Modal Interface Automata: A Theory for Heterogeneous Specification of Parallel Systems

Dissertation
zur Erlangung des Doktorgrades
an der Fakultät für Angewandte Informatik
der Universität Augsburg

vorgelegt von
Ferenc Péter Bujtor

Oktober 2018

Erstgutachter:	Prof. Dr. Walter Vogler
Zweitgutachter:	Prof. Dr. Alexander Knapp
Tag der Mündlichen Prüfung:	12. Oktober 2018

Thanks / Danksagungen

I would like to thank Prof. Dr. Walter Vogler, my mentor, coauthor and Ph.D. advisor. Thanks to you, I learned what a proper proof is and how to write one.

I would like to thank my parents, who made it possible for me to study and learn. Thanks to you I made it this far.

I would like to thank my love Alina, who always supported me. Thanks to you I can stay motivated and happy.

Ich danke Prof. Dr. Walter Vogler, meinem Mentor, Koautor und Doktorvater. Dank Dir habe ich gelernt, was ein sauberer Beweis ist und wie man ihn führt.

Ich danke meinen Eltern, die es mir ermöglicht haben zu lernen und zu studieren. Dank Euch habe ich es soweit gebracht.

Ich danke meiner Liebsten Alina, die mich stets unterstützt hat. Dank Dir kann ich motiviert und glücklich bleiben.

Abstract

This thesis presents the latest incarnation of the interface theory of *Modal Interface Automata (MIA)* which unifies several different types of specification mechanisms. It allows for stepwise and aspect-oriented specifications and provides structural operators, like parallel composition and hiding, logical operators like conjunction and disjunction, and even temporal logic operators like Universal and Existential Next-operators and an Unless-operator.

MIA combines the modalities of (disjunctive) Modal Transition Systems (dMTSs) with the input/output handling of Interface Automata (IA). Thus, we examine the fundamentals of these two specification theories considering different aspects of possible refinement relations before presenting the MIA framework.

We examine failure semantics and failure/divergence semantics for dMTSs and their justification via testing scenarios. We also examine the fundamental design decisions of IA: the pruning built into parallel composition and the optimistic approach of the refinement. The resulting refinement relations are compared to the more traditional simulation relations on which the refinement of MIA is built. Further insights from these examinations are also incorporated into the interface theory that is MIA.

Contents

Chapter 1

Introduction

Interface Theories [26, 22, 24, 42, 51, 59] are tools for modelling and analysing concurrent systems in terms of their communications. They also support the component-based design of concurrent systems and offer a semantic framework for, e.g., software contracts [3] and web services [9]. The interface specifications of such a theory describe at the same time assumptions towards the environment for the correct functioning of a component and the behaviour of said component.

In [26], de Alfaro and Henzinger proposed that an interface theory should provide a model for the input/output behaviour of components, a (parallel/structural) composition operator, usually accompanied with a notion of compatibility, to build complex components, and a hierarchy for abstraction/implementation, i.e. a *refinement relation*. These should feature *independent implementability*, i.e. the refinement should be *compositional*: Given two compatible interfaces P and Q, if P can be refined to P' and Q to Q', then the composition $P \parallel Q$ can be refined to $P' \parallel Q'$. Formally, this means the refinement relation should not only be a preorder, but a precongruence for parallel composition and in fact for all other relevant operators, as well.

When using a refinement relation (formally a preorder), a specification P refines another specification Q ($P \sqsubseteq Q$) if it is more specific and/or better in some sense – or at least equal. This is a generalization from using equivalence relations, as is often done, e.g. for process algebras [57, 39, 37] and creates a formal hierarchy from abstract to more concrete specifications. Thus, it allows for a stepwise design process starting from an abstract specification and making small local design decisions one after the other while always preserving previous properties. There is a large spectrum of possible refinements (and equivalence relations), as has been examined in several papers (cf. e.g. [68]). A simple one would be *Labelled Transition Systems (LTS)* with simulation, which for example implies language inclusion (cf. Sect. 1.1

below for a formal definition). Thus, properties like 'cannot perform trace *abb*' are preserved during refinement. This might be desirable, e.g. in the context of error-avoidance.

Once a theory provides means to specify simple specifications and to refine them, it is natural to require that an implementation should satisfy several specifications. Thus, more recent interface theories also feature a conjunction operator. With it one can specify different aspects of a system separately and conjoin these simpler *aspect specifications* to a global one. This works even better if the framework also provides a means of alphabet extension. Then, when creating the aspect specifications a designer need only concern himself with the actions regarding the aspect in question, which is quite a natural desire (cf. e.g. [36, 59] and the input aspect of [27]).

From a more formal point of view, the conjunction $P \wedge Q$ of two specifications P and Q is refined exactly by the specifications refining both P and Q. Should there be no common refinement, the conjunction is naturally undefined; in this case P and Q are considered *inconsistent*. From a mathematical point of view the conjunction is the greatest lower bound of the refinement and thus unique (up to equivalence) if it exists. Therefore, when designing an interface theory, the problem is not in defining what conjunction should be, but in finding an operator, i.e. a construction, yielding a representative of the equivalence class. The difficulty is that a conjunction $P \wedge Q$ cannot always be represented as a specification of the theory, even if P and Q have common refinements; it is for example impossible for Modal Transition Systems [45] with modal refinement (cf. [7] or Prop. 2.45 later on).

As mentioned by De Nicola et al. in [29], an equivalence-based approach has the major disadvantage that even the most abstract specifications tend to be too concrete. This may be mitigated by using refinement (with conjunction and aspect specifications) instead, but the authors use a different solution: They combine LTS with temporal logics, in particular ACTL, a transition based variant of CTL on LTSs to allow for more abstract specifications of properties. Temporal logics provide compact and more intuitive specifications for system properties [38, 37, 56] and are particularly suited to deal with notions of necessity, possibility and eventuality [29].

In this thesis, we present the latest incarnation of *Modal Interface Automata (MIA)* (first presented in [12]). It is an interface theory built on de Alfaro and Henzinger's *Interface Automata (IA)* [27] for input/output behaviour and on an extension of Larsen's *Modal Transition Systems (MTS)* [45] for refinement. Additionally, it provides alphabet extension for better aspect orientation. Furthermore, it features the usual structural operators, like parallel composition and hiding, as well as the logical operators, conjunction and disjunction, and even a safety fragment of ACTL making MIA a truly heterogeneous interface theory.

De Alfaro and Henzinger's theory, Interface Automata [25, 26, 27], fits their requirements for an interface theory. It features LTS with disjoint input- and output-alphabets and an alternating simulation as refinement, where during refinement inputs can be added and outputs removed. Its handling of input/output behaviour and communication errors is the basis for several more advanced interface theories [4, 42, 51, 59]. The main assumption is that any output sent by one component must be immediately accepted by a receiving component, otherwise a fatal error occurs. The intuition is that any specified behaviour is better than a fatal error. This main assumption is, in fact, not new; it has been presented in the deterministic context of speed independent circuits in [32].

The main problem of IA is that it only aims for error prevention. Thus, adding an input or removing an output during refinement is always allowed. This leads to a *blackhole process* [51], which accepts all inputs and never sends outputs, to be the optimal system. In practice, however, such a system is quite useless. There have been several approaches to remedy this. One is to forbid or at least consider the introduction of *quiescence* (cf. [22, 24]), the absence of outputs, as in ioco-testing [64].

The solution we will use, is to combine IA with *disjunctive Modal Transition Systems (dMTS)* [51], an extension of MTS as introduced by Larsen [45]. Several such approaches exist [4, 42, 51, 53, 59]. MTS features two types of transitions: Prescribed *must*-transitions and permitted *may*-transitions. This allows to prescribe outputs. dMTS are an extension of MTS in that a must-transition (with one label) can lead to a set of targets requiring only one to be implemented. They are also a specialization of DMTS [44], where must-transitions with different labels can be grouped together in a similar manner. Using dMTS is necessary and sufficient to allow for a conjunction operator: MTS would not allow for one (cf. [51]) while DMTS would be unnecessarily complicated.

There have been several combinations of IA and MTS before. To the best of our knowledge, the first was *IOMTS* [42] by Larsen et al.; however their compositionality result was not correct. *Modal I/O Automata (MIO)* [4] by Bauer et al. incorporate quite different design decisions from MIA, and thus also different from the ideas of IA. Bauer et al.'s research focuses more on notions of compatibility. MIA can be seen as an improvement of *Modal Interfaces (MI)* [59], due to similar design decisions. In contrast to MI, MIA allow for nondeterminism and features an associative parallel composition, which is far from being a given when dealing with errors and pruning. We will go into more detail in the chapter pertaining to MIA.

The rest of this thesis is structured as follows: We will first examine and compare several refinement relations for MTS (and dMTS) in Chapter 2. This combines the results and insights of [19] and [15] (extended abstracts were published as [17] and [14]). We examine test-based refinements for avoidance of deadlocks and divergences, as well as may- and two versions of must-testing [30]. These lead to variations of Failure- and Failure/Divergence-Semantics (cf. e.g. [69, 65]). We compare them among each other and to the traditional alternating simulations known as modal refinement and observational modal refinement.

In Chapter 3 we will scrutinize two basic design decisions of IA: taking an optimistic view and integrating pruning into parallel composition. For this, we examine *Error-IO-Transition-Systems (EIO)*, which are essentially IA with explicit error states. These considerations have been published in [16] as an extended abstract and fully in [18]. We will consider three different notions of error-freedom for EIOs: For the hyper-optimistic approach the EIO is still considered error-free if it cannot reach any error by internal actions. For the optimistic approach it must not reach errors via locally controlled actions (internal and output); for the pessimistic approach it must not have any reachable error. Furthermore, we do not assume pruning during parallel composition as correct, but instead use the traditional parallel composition for LTS while marking error states as such. Our conclusions will in fact confirm the design decisions of IA: It is appropriate to take an optimistic view, where a component is only considered erroneous, if no environment can prevent it from running into an error. Furthermore, we show that pruning (which in IA is built into the parallel composition) is an equivalence-transformation on EIO, i.e. we prove it correct.

After these considerations, the choices that went into the design of MIA are no longer only design decisions, but consequences of more basic and less disputable decisions. For a more detailed overview of these topics we refer to the introductions of the respective chapters.

We will finally examine MIA in Chapter 4. This chapter is based on [13] (extended abstract [12]), which first introduced MIA, and on [21] (extended abstract [20]), which added temporal logics. However, contrary to these papers, we will here work with a modified version of the refinement relation, as will be discussed in the chapter itself. Thus, most proofs and several constructions have been adjusted; this work is yet to be published. As mentioned before, MIA combines the modalities of (d)MTS with the input/output handling of IA. We will discuss the details and motivations for this combination as well as variations found in the literature. We go on to introduce the structural and logical operators (including temporal logics). Finally, we will also introduce an improved alphabet extension allowing for better aspect-oriented specification; this, too, is yet to be published.

1.1 Notation and Basic Definitions

To introduce notational conventions we recall some standard definitions. We also define LTS, its parallel composition, failure semantics and simulation, which we will build upon in later chapters.

We assume a set Σ of visible actions and an additional internal action τ; we write Σ_τ for $\Sigma \cup \{\tau\}$ and the analog for other alphabets, i.e. subsets of Σ. We let a range over Σ and α over Σ_τ. Similarly, when distinguishing between input and output alphabets I and O (we sometimes use I/O as a shorthand), we let i, o and ω range over I, O and O_τ. For $A \subseteq \Sigma_\tau$, we denote with A^* and A^ω the sets of finite and infinite words over A respectively. The empty word is denoted by ε.

For $v \in \Sigma^*$ and $w \in \Sigma^* \cup \Sigma^\omega$ we write $v \sqsubseteq w$ if v is a *prefix* of w, and $v \sqsubset w$ if additionally $v \neq w$. Some $w \in \Sigma^*$ is prefix-minimal in some set $L \subseteq \Sigma^*$ if $w \in L$ and for all $v \in L$: $v \sqsubseteq w$ implies $v = w$.

Definition 1.1. *A* Labeled Transition System (LTS) *is a tuple* $(P, \mathcal{A}, \longrightarrow, p_0)$ *where*

1. P *is a set of* states*,*
2. $p_0 \in P$ *is the* initial state*,*
3. $\mathcal{A} \subseteq \Sigma$ *is the* alphabet*,*
4. $\longrightarrow\, \subseteq P \times \mathcal{A}_\tau \times P$ *is the* transition *relation.* □

For notational convenience, we denote an LTS (and the other systems considered in the other chapters) with its state space, i.e. we denote $(P, A, \longrightarrow, p_0)$ by P. We write $\mathfrak{P}(P)$ ($\mathfrak{P}_{fin}(P)$) for the (finite) powerset of P. As a shorthand we write $p_{01}, \mathcal{A}_1$ etc. for components of a system P_1, $p_{02}, \mathcal{A}_2$ etc. for P_2 and so on. We also use this notation for semantics and e.g. write $\mathcal{F}_1$ for $\mathcal{F}(P_1)$ as defined later on.

We write $p \xrightarrow{\alpha} p'$ for $(p, \alpha, p') \in\, \longrightarrow$ and $p \xrightarrow{\alpha}$ if $\exists p'.\, p \xrightarrow{\alpha} p'$. Extending transitions to sequences $w \in \Sigma_\tau^*$, we write $p \xrightarrow{w} p'$ ($p \xrightarrow{w}$) whenever there is a *run* $p \xrightarrow{\alpha_1} p_1 \xrightarrow{\alpha_2} \cdots p_{n-1} \xrightarrow{\alpha_n} p'$ ($p \xrightarrow{\alpha_1} p_1 \xrightarrow{\alpha_2} \cdots p_{n-1} \xrightarrow{\alpha_n}$) with $w = \alpha_1 \cdots \alpha_n$. Similarly, we also use $p \xrightarrow{w}$ for *infinite sequences* $w \in \Sigma_\tau^\omega$.

Furthermore, $w|_B$ denotes the action sequence obtained from w by deleting all actions not in $B \subseteq \Sigma_\tau$ and $\hat{w}$ denotes $w|_\Sigma$. We write $p \overset{w}{\Longrightarrow} p'$ for

$w \in \Sigma^*$ if $\exists w' \in \Sigma_\tau^* : w'|_\Sigma = w \wedge p \xrightarrow{w'} p'$, and $p \overset{w}{\Longrightarrow}$ if $p \overset{w}{\Longrightarrow} p'$ for some p'. Sometimes, we refer to a run $p \overset{w}{\Longrightarrow} p'$, thereby fixing a run giving rise to $p \overset{w}{\Longrightarrow} p'$ for later reference.

For may-transitions of MTS, dMTS and MIA we use analogous notation with dashed arrows. Where disjunctive must-transitions are involved, the above definition for weak transitions and transition sequences does not work. We will comment on this and provide solutions in the respective chapters and use the above only for may-transitions.

In figures, we will analogously use dashed arrows for may-transitions and solid (possibly split) ones for (disjunctive) must-transitions, which also imply the underlying may-transitions. When distinguishing between inputs and outputs, we write $a?$ for inputs and $a!$ for outputs.

The following definitions of parallel composition on LTS and traces are quite standard. We will mostly use parallel composition without immediate hiding of synchronising actions. In the following definition, the set of synchronising actions is a parameter of the parallel composition operator. In the chapters on EIOs and MIA, where we distinguish inputs and outputs, we will use parallel composition with full synchronization, i.e. synchronization on all common actions.

Definition 1.2. *The* parallel composition *of two LTS $P_1 = (P_1, \mathcal{A}_1, \longrightarrow_1, p_{10})$ and $P_2 = (P_2, \mathcal{A}_2, p_{20}, \longrightarrow_2, p_{20})$ with synchronizing set $A \subseteq \Sigma$ is defined as the LTS $P_1 \parallel_A P_2 = (P_1 \times P_2, (p_{10}, p_{20}), \mathcal{A}_1 \cup \mathcal{A}_2, \longrightarrow_{12})$, where*

$$\begin{aligned}\longrightarrow_{12} = &\left\{\big((p_1,p_2),\alpha,(p_1',p_2)\big) \mid p_1 \xrightarrow{\alpha}_1 p_1', \alpha \in \Sigma_\tau \setminus A\right\}\\ \cup &\left\{\big((p_1,p_2),\alpha,(p_1,p_2')\big) \mid p_2 \xrightarrow{\alpha}_2 p_2', \alpha \in \Sigma_\tau \setminus A\right\}\\ \cup &\left\{\big((p_1,p_2),a,(p_1',p_2')\big) \mid p_1 \xrightarrow{a}_1 p_1', p_2 \xrightarrow{a}_2 p_2', a \in A\right\}\end{aligned}$$

The parallel compositions of $u, v \in \Sigma^*$ is the set $u \parallel_A v$ defined as follows: $w \in u \parallel_A v$ iff $u = u_1 u_2 \cdots u_n$, $v = v_1 v_2 \cdots v_n$ and $w = w_1 w_2 \cdots w_n$ with each $u_i, v_i \in \Sigma \cup \{\varepsilon\}$ and $w_i \in \Sigma$ such that for every i we have $u_i = v_i = w_i \in A$ or $\varepsilon \neq u_i = w_i \notin A \wedge v_i = \varepsilon$, or $\varepsilon \neq v_i = w_i \notin A \wedge u_i = \varepsilon$.

Intuitively, as is usual, the parallel composition tracks the states of both component systems in the components of its states. Synchronizing actions must be performed simultaneously by both, while non-synchronising and internal actions are performed separately leaving the other component in the same state.

Parallel composition can also be applied to traces.

Definition 1.3 (Parallel Composition on Traces)**.** *Given two composable LTSs P_1, P_2, $w_1 \in \mathcal{A}_1, w_2 \in \mathcal{A}_2, W_1 \subseteq \mathcal{A}_1^*$ and $W_2 \subseteq \mathcal{A}_2^*$, we define*

- $w_1 \parallel w_2 = \{w \in (\mathcal{A}_1 \cup \mathcal{A}_2)^* \mid \ w|_{\mathcal{A}_1} = w_1 \wedge w|_{\mathcal{A}_2} = w_2\}$
- $w_1 \mid w_2 = \{w|_{\mathcal{A}_{12}} \ \mid \ w \in w_1 \parallel w_2\}$
- $W_1 \parallel W_2 = \bigcup\{w_1 \parallel w_2 \ \mid \ w_1 \in W_1 \wedge w_2 \in W_2\}$
- $W_1 \mid W_2 = \bigcup\{w_1 \mid w_2 \ \mid \ w_1 \in W_1 \wedge w_2 \in W_2\}$

In the following chapters we will examine several domains of specifications, e.g. dMTS and consider a subset, e.g. LTS to be implementations. On such a domain we can formally define refinements:

Definition 1.4. *Let $\mathcal{S}$ be a class of* systems *and $\mathcal{I} \subseteq \mathcal{S}$ a sub-class of* implementations*. A* refinement *is a preorder, i.e. a reflexive and transitive binary relation, on $\mathcal{S}$.*

Given a refinement relation $\sqsubseteq_R$*, the set of* R-implementations *of a system P is $impl_R(P) = \{S \mid S \sqsubseteq_R P, S \text{ is an implementation}\}$.*

We say that P thorough-R-refines *MTS P' if R-impl$(P) \subseteq R$-impl(P'). We call R* thorough *if R-refinement and thorough-R-refinement coincide.*

A simple and straightforward notion of refinement is language inclusion. For our purposes the language (called basic language here) is the set of possible traces.

Definition 1.5. *The* basic language *of an LTS P is $L(P) = \{w \in \mathcal{A}^* \mid p_0 \overset{w}{\Longrightarrow}\}$.*

P language-refines another LTS Q if $L(P) \subseteq L(Q)$.

For most purposes, however, this refinement is too coarse. It does not reflect the nondeterminism and choice inherent in an LTS at all.

Failure semantics (cf. e.g. [65, 69, 70]) makes finer distinctions than language inclusion. The idea is to enrich traces with a set of actions yielding so called refusal pairs (w, X). The refusal set X is a set actions that the system can refuse, all at once, after performing the trace w.

Definition 1.6. *The* failure semantics *of an LTS P is $\mathcal{F}(P) = \{(w, X) \mid p_0 \overset{w}{\Longrightarrow} p, \forall a \in X : p \overset{a}{\not\Longrightarrow}\}$.*

P failure- *(or $\mathcal{F}$-)*refines *P', written $P \sqsubseteq_{\mathcal{F}} P'$, if $\mathcal{F}(P) \subseteq \mathcal{F}(P')$. P and P' are* failure- *(or $\mathcal{F}$-)*equivalent*, written $P =_{\mathcal{F}} P'$, if $\mathcal{F}(P) \sqsubseteq_{\mathcal{F}} \mathcal{F}(P')$ and $\mathcal{F}(P') \sqsubseteq_{\mathcal{F}} \mathcal{F}(P)$.*

We call $(w, X) \in \mathcal{F}(P)$ a refusal pair *and X a* refusal set*; we say that the respective p* can refuse *X.*

This semantics reflects some degree of nondeterminism. For example, if $\mathcal{F}(P)$ contains $(w, \{a\})$ and $(w, \{b\})$ but not $(w, \{a, b\})$, we can infer that w can be performed in at least two different ways leading to two different states.

Failure semantics describes the potential deadlocks of a system. The refusal pair (w, X) implies that there is a state p, which can be reached by w and which can refuse each action in X. If all actions can be refused, i.e. we have (w, Σ), then the state or at least a state reached after w is a *deadlock*, i.e cannot perform any visible action. This is particularly interesting, when considering parallel composition. The following well known lemma (cf. e.g. 3.2.4. in [70]) shows how the failure semantics of a parallel composition can be obtained from the semantics of the components and implies that $\sqsubseteq_{\mathcal{F}}$ is a precongruence for $\|_A$. Furthermore, it also reflects how deadlocks can appear in an LTS.

Lemma 1.7. *For all LTS P and Q and all $A \subseteq \Sigma$: $\mathcal{F}(P \|_A Q) = \{(w, X) \mid (u, Y) \in \mathcal{F}(P), (v, Z) \in \mathcal{F}(Q), w \in u \|_A v, X \subseteq (Y \cap Z) \cup (A \cap (Y \cup Z))\}$.*

Intuitively, for $(w, X) \in \mathcal{F}(P \|_A Q)$, P and Q synchronize on two traces as described above. If both components refuse an action a afterwards, then so does $P \|_A Q$. For $a \in A$, the components have to synchronize on a; hence $P \|_A Q$ refuses a already if one component does. Consequently a deadlock, $(w, \Sigma) \in \mathcal{F}(P_1 \|_A P_2)$, can only appear, if both P_1 and P_2 can perform w to states such that all non-synchronising actions are refused by both ($Y \cap Z$ above) and synchronizing actions are refused by at least one of the components ($A \cap (Y \cup Z)$ above).

Because of this close connection to deadlocks, failure refinement arises if one tests systems for deadlock freedom considering P better than Q if P is deadlock free in all compositions in which Q is. This will be more thoroughly examined and explained in Chapter 2.

Another well known concept of refinement is simulation: Given two LTS P and Q, the idea is to match states of P to corresponding states of Q. p can match q if q has at least the same transitions as p, each with matching behavior afterwards. For the weak simulation, q is allowed to use internal transitions for matching, as well.

Definition 1.8. *Let P, Q be LTSs. A relation $\mathcal{R} \subseteq P \times Q$ is a* (strong) simulation relation *if for all $(p, q) \in \mathcal{R}$:*

$$p \xrightarrow{\alpha} p' \textit{ implies } \exists q' : q \xrightarrow{\alpha} q' \textit{ and } (p', q') \in \mathcal{R}.$$

Similarly $\mathcal{R} \subseteq P \times Q$ is a weak simulation relation *if for all $(p, q) \in \mathcal{R}$:*

$p \xrightarrow{\alpha} p'$ *implies* $\exists q' : q \overset{\hat{\alpha}}{\Longrightarrow} q'$ *and* $(p', q') \in \mathcal{R}$.

We write $p \sqsubseteq_{sim} q$ *(*$\sqsubseteq_{w\text{-}sim}$*) and say that* p is (weakly) simulated by q *if there exists a (weak) simulation relation* $\mathcal{R}$ *such that* $(p, q) \in \mathcal{R}$.

Furthermore, we extend this notion to LTSs and write $P \sqsubseteq_{sim} Q$ *if* $p_0 \sqsubseteq_{sim} q_0$ *– analogously for* $\sqsubseteq_{w\text{-}sim}$. □

As usual, $\sqsubseteq_{sim}$ is the greatest simulation relation and a preorder, i.e. $\sqsubseteq_{sim}$ is a refinement relation. The analog is true for $\sqsubseteq_{\text{w-sim}}$.

Strong and weak simulation, as well as failure semantics are compositional, i.e. the refinements are precongruences w.r.t. parallel composition. We do not present the proofs here, since the results are well known and corollaries of results in later chapters.

Proposition 1.9. *Given LTS* P', P *and* Q *with* $P' \sqsubseteq P$*, we have:*

- $P' \sqsubseteq_{\mathcal{F}} P$ *implies* $P' \parallel_A Q \sqsubseteq_{\mathcal{F}} P \parallel_A Q$,
- $P' \sqsubseteq_{sim} P$ *implies* $P' \parallel_A Q \sqsubseteq_{sim} P \parallel_A Q$,
- $P' \sqsubseteq_{w\text{-}sim} P$ *implies* $P' \parallel_A Q \sqsubseteq_{w\text{-}sim} P \parallel_A Q$,

As mentioned at the beginning, this property is highly desirable in any refinement relation. It is key to a component based approach, since it allows one to replace a component by a finer/better one thereby refining/improving the whole system. In short, it allows for independent implementability.

Simulation-based refinement relations are often quite intuitive and capture more of the nondeterminism of a system than trace-based refinement relations, like $\mathcal{F}$-refinement. Indeed, strong simulation implies $\mathcal{F}$-refinement which in turn implies language inclusion. It is also known that simulation-based refinements are, in general, cheaper to decide than refinements based on trace inclusion. However, while simulations are intuitive, they tend to be too strict and may sometimes reject implementations for unintuitive reasons. In our fundamental examinations of (d)MTS and EIO we pursue refinement relations that only reject implementations due to specific and well-defined reasons. Such reasons involve the introduction of an unwanted property, like a deadlock, either immediately or, more usually, in some environment. Rejection of the latter is necessary to ensure that the refinement is a precongruence. These pursuits lead us to trace-based characterizations, which we consider to be optimal from this point of view. For MIAs, however, we use a simulation-based refinement which seems more appropriate for several reasons detailed in the introduction of the Chapter 4.

Chapter 2

Notions of Refinement for MTS and dMTS

2.1 Introduction

The most important issue when refining a specification in a step-wise system development is that any *desirable property* satisfied by the specification *is preserved*. As mentioned before, in a component-based approach, it is equally important that the refinement relation is a *precongruence* for parallel composition: If we refine a component specification P to P', then replacing P by P' in an overall specification $P \parallel Q$ should lead to a refinement $P' \parallel Q$ – and the respective desirable property (or properties) is preserved. We are convinced that, in such a setting, the *optimal* refinement notion is the one that does not reject any P' as refinement of P unless this is necessary to achieve these two goals. In other words, it is optimal to use the coarsest precongruence, denote it by $\sqsubseteq$, that guarantees preservation of the desirable property. Efficiency to decide whether $P' \sqsubseteq P$ is of secondary importance; if there is a finer, more efficiently decidable precongruence, one can still choose to work with this as an under-approximation of $\sqsubseteq$ (cf. [65]) – the price is of course that one restricts oneself to a subset of the set of reasonable refinements.

Testing in the sense of De Nicola and Hennessy [30] is an approach that usually leads to such an optimal refinement relation. A specification P *satisfies* a test system T (you might think of it as a user) if $P \parallel T$ has the desired property under consideration, say deadlock freedom.[1] Then P' refines P according to the *testing preorder* $\sqsubseteq_d$, if P' satisfies at least all test systems (users) that P satisfies. If $\sqsubseteq_d$ is a precongruence, it is usually the

[1] In [30], the property is based on reaching some success state of T.

optimal one as explained above.

The problem is of course that $\sqsubseteq_d$ is based on all possible test systems; to understand and e.g. decide it, we need a characterization of $P' \sqsubseteq_d P$ that just refers to the behaviour of P and P'. For LTS, such a characterization is given by $\mathcal{F}(P') \subseteq \mathcal{F}(P)$ (cf. Def. 1.6).

We will start this chapter by generalizing this deadlock testing to MTS and then to dMTS. As mentioned in Chapter 1, an MTS has two kinds of transitions: may-transitions, which describe permitted behaviour, and must-transitions, describing required behaviour. dMTS additionally allow for disjunctive must-transitions targeting several states and require that at least one of them is implemented. Contrary to DMTS (introduced in [44]), where any may-transitions originating from a single state can be grouped together into a must-transition, dMTS also requires them to have the same label. We will, however, not examine DMTS here for several reasons. Firstly, despite having found quite some interest over the years (see e.g. [6] for a recent publication), DMTS are difficult to handle; in particular, parallel composition on DMTSs has only been defined 20 years after their invention [7]. The more common definition found in [6] involves translating the two DMTSs into Nondeterministic Acceptance Automata, building their parallel composition and translating the result back into a DMTS. Secondly, dMTS suffice for our purposes. We base MIA on them (cf. Ch. 4) rather than MTS to provide a conjunction operator. Again DMTS would be unnecessarily complicated. Finally, it will turn out, that, already for dMTS, failure refinement is not a precongruence.

When defining the test-setting for MTS (and dMTS afterwards), we immediately encounter a difficulty: It is not so clear what a deadlock in an MTS is – does a may-transition save us from a deadlock? Since in MTS-settings LTSs are often considered *implementations*, we solve this by not checking for deadlock freedom on MTS, but on all its implementations instead. There the notion of deadlock is rather clear. Thus, to make this work, we have to associate each MTS P with a set of implementations.

However, since we are trying to define a refinement- (and thus implementation-) relation, it is not immediately clear what implementations we should associate with P. The standard refinement relation for MTS and (dMTS) is based on an alternating simulation, we will call it *as-refinement*[2]; with this refinement relation, MTSs are more expressive than LTSs. So we stick with the traditional concept of as-implementations; since as-refinement

[2] Usually, this relation is called modal refinement; this does not appear very suitable, since there are certainly very different refinement relations for *Modal* Transition Systems, as we will demonstrate.

is so strict, it seems indubitable that these really are implementations of P.

Since the characterization of the refinement defined by our test-setting is quite intricate for dMTS and will turn out not to be precongruence, we will start with examining only MTS. We characterise the refinement with a kind of failure semantics extending the $\mathcal{F}$-semantics of LTS. The characterization answers our question about deadlocks showing that may-transitions do not save us from a deadlock; intuitively, this is because there is an implementation where the may-transition is omitted and thus a deadlock occurs. We show the *robustness* of our test-scenario by running it again, this time using $\mathcal{F}$-implementations instead of as-implementations: The resulting refinement is the same. Finally, we show that our refinement is thorough in contrast to as-refinement.

We proceed to do the same examinations for dMTS. The characterization is technically far more intricate than the one for MTS, which it nevertheless extends. It uses the concept of so-called τ-choices to deal with disjunctive τ-must-transitions and ready sets, a concept complementary to refusal sets; we will offer some ideas how the process of determining the semantics of a dMTS can be improved, thereby exhibiting a connection to stable-failure semantics, cf. [8] or [57, Sect. 9.4]. It turns out that most problems arise from divergences and disjunctive τ-must-transitions. Indeed, as preparatory results, Sorokin, the second author of [15], already showed in his bachelor's thesis [62] that failure semantics for deadlock-testing is much easier to handle, when we do not allow the internal action τ on disjunctive transitions or require divergence freedom (as discussed later on) and use stable failures. It should also be mentioned that Valmari [66] has considered deadlock testing on LTS with a slightly different understanding of deadlock. In his setting, the testing preorder can be characterized with stable-failure semantics. In these cases, the refinement is also a precongruence, as it is for MTS.

Since in general the refinement fails to be a precongruence on dMTS, it is just an over-approximation for the desired optimal precongruence from above, complemented by as-refinement as under-approximation. We will also show that, with this semantics, dMTS are not more expressive than MTS. Therefore, we will not pursue the failure semantics further for dMTS, but only for MTS.

We proceed to examine conjunction for our $\mathcal{F}$-semantics on MTS. As mentioned in Chapter 1, conjunction is an important operator if one wants to describe various aspects of a specification separately and then combine them into one overall specification.

Obviously, conjunction depends on the refinement notion. Conjunction for MTSs w.r.t. as-refinement has already been defined in [46], but the result usually violates *syntactic consistency*; the standard condition of syntactic consistency requires each must-transition to have an underlying may-transition as well, so that all required behaviour is permitted. In other words, the conjunction operator is not closed on MTS. In fact, a conjunction w.r.t. as-refinement does not always exist in MTS. In a setting without τ, [35] presents a characterization when it exists and how to construct it in this case. But the conjunction of MTSs exists in the class of dMTS as shown for general MTSs w.r.t. a weak form of as-refinement in [51]. Before that, it was shown that the conjunction for the τ-free case exists on the larger class of DMTS (c.f. [7]). The latter two conjunction operators are closed on their respective classes.

We will present a conjunction w.r.t. $\mathcal{F}$-refinement. This operator is quite different from the ones found in the literature, and it demonstrates that in our setting the conjunction of two MTSs is always an MTS again – provided the two have a common $\mathcal{F}$-refinement; otherwise, conjunction is necessarily undefined. This conjunction on MTS is very interesting, since MTS are conceptionally much easier than dMTS or DMTS. As an aside, we also prove the existence of a conjunction operator for dMTS with $\mathcal{F}$-semantics.

However, intuitively, $\mathcal{F}$-refinement can be regarded as too generous. It ignores the interplay of may- and must-transitions to a large degree and treats the MTS in question as *action modal*: when refining a state p with several a-may-transitions, it only matters whether p has an a-must-transition or not; regardless to which state it leads, p can be implemented by elevating any a-may-transition to a must-transition. Essentially it would suffice to use only one type of transitions and annotate each state with a set of actions that have to be present.

Hence, we suggest a way to make $\mathcal{F}$-refinement stricter by combining it with a refinement based on may-testing [30]. So far, the success of a test meant that the system in parallel with the test environment works without deadlocking. In the may- and must-testing of De Nicola and Hennessy success means that a successful state of the test environment may or must be reached eventually.[3] We develop may-testing for MTS with our implementation-based testing scenario and characterize the resulting preorder as reverse language inclusion based on must-transitions. Adding this to $\mathcal{F}$-refinement makes finer distinctions w.r.t. must-transitions. Conjunction, however, cannot be defined on MTS for this stricter refinement.

In the second half of this chapter, we have a look at deadlock/divergence testing, where test satisfaction means deadlock and divergence freedom. Di-

[3] The concepts of may- and must-testing are not to be confused with the may- and must-transitions of MTS.

vergence (an infinite internal computation) is often regarded as undesirable because it can block any communication with the environment. In addition, we have seen that most problems and complications with dMTS and $\mathcal{F}$-refinement stem from divergences.

Indeed, the idea of failures is best known as part of the, as we will call it, *traditional* failure/divergence semantics, which arises from the must-testing preorder of [30] – cf. [28] – and was developed as a denotational semantics for CSP in [11]. The explicit treatment of *divergence traces* in the latter is needed to make denotational semantics work in the presence of hiding.

Our results regarding deadlock and divergence are not just generalizations, but already new for LTS. It seems that so far such a deadlock/divergence testing has only been mentioned once in [70], essentially in an LTS setting; but the characterization claimed in [70, Thm. 3.2.6.i) 3rd and 4th item] is wrong. We correct this here, characterizing the testing preorder with a widely unknown semantics that combines failures with an unusual treatment of traces leading to divergence; these traces and their continuations are called divergence traces. So far, this semantics was presumably only studied in [69] to solve a problem from [5]; previous results about this semantics are very limited, so we regard it as still new – in particular, since we formulate it here for the first time on the basis of stable failures. This time, the deadlock/divergence preorder is not only robust (in the sense $\mathcal{F}$-semantics was shown to be), it is also a precongruence, it can be seen as the coarsest precongruence as described above, and the characterizing semantics is fairly easy to handle.

In the traditional semantics, the failure part is flooded with divergence traces in the sense that it contains any (w, X) where w is a divergence trace. In our new semantics, there is an additional flooding: action a can also be in X because wa is a divergence trace. This way, our semantics leads to a coarser, hence better refinement; it is strictly coarser than the traditional refinement already on LTS. We show that the new refinement is also a precongruence for hiding, and it can be justified by a variation of De Nicola's and Hennessy's must-testing as well.

The rest of this chapter is structured as follows. The second section introduces basic definitions and notations. The third deals with deadlock-testing and failure semantics for MTS and dMTS; it also includes its combinations with may-testing and conjunction on MTS. The new deadlock-divergence-testing and failure-divergence semantics are treated in the fourth section, while the fifth gives an overview of the refinement relations we examined. We conclude the chapter with a brief summary in the sixth section.

2.2 Definitions and the Concept of Testing

We start by formally defining MTS, dMTS and their traditional refinement relations, the strong and weak alternating simulation. (The latter were originally defined as modal and observational modal refinement.)

Definition 2.1. *A* disjunctive modal transition system (dMTS) *is a tuple* $(P, p_0, \longrightarrow, \dashrightarrow)$ *where*

1. *P is a set of* states*,*
2. *$p_0 \in P$ is the* initial state*,*
3. *$\longrightarrow \subseteq P \times \Sigma_\tau \times (\mathfrak{P}_{fin}(P) \setminus \{\emptyset\})$ is the* (disjunctive) must-transition *relation,*
4. *$\dashrightarrow \subseteq P \times \Sigma_\tau \times P$ is the* may-transition *relation.*

As is usual for (d)MTS, we require syntactic consistency*, i.e. $p \xrightarrow{\alpha} P'$ implies $\forall p' \in P' : p \overset{\alpha}{\dashrightarrow} p'$.*

We identify $p \xrightarrow{\alpha} \{p'\}$ with $p \xrightarrow{\alpha} p'$. If all must-transitions are of this kind, i.e. $\longrightarrow \subseteq P \times \Sigma \times \{\{p\} \mid p \in P\}$, then P is a modal transition system (MTS)*. If $\longrightarrow = \dashrightarrow$, P is (essentially) a LTS, which we also consider to be an* implementation*. The* alphabet *of P is the set collecting all visible actions occurring on may-transitions of P.*

Intuitively, e.g. $q \xrightarrow{a} \{q_1, q_2\}$ means that we must have an a-transition from q to q_1 or q_2. If we additionally have $q \xrightarrow{a} \{q_2, q_3\}$, then either there is at least an a-transition to q_2 or at least an a-transitions to q_1 and one to q_3. This does not exclude the possibility of implementing all three a-transitions. Analogously, $q \overset{a}{\dashrightarrow} q_1$ means that an a to q_1 is allowed.

Note that the meaning of $\Longrightarrow$ on dMTS is not obvious; a weak transition ($\overset{\alpha}{\Longrightarrow}$) is defined in [52], and we will use a slightly different definition in Chapter 4 later on. Here, we will use $\Longrightarrow$ only for LTS or MTS. Consequently, the same holds for weak as-implementation (see definition below), which is based on $\Longrightarrow$. If an MTS is known to be an implementation (an LTS), we write all transitions as must-transitions.

Reachability is based on may-transitions, i.e. if $p_0 \overset{w}{\Longrightarrow} p$, w is called a *trace* and p is a *reachable* state of P. A state p is *must-stable*, if $p \overset{\tau}{\not\longrightarrow}$ and *stable* if $p \overset{\tau}{\not\dashrightarrow}$.

Definition 2.2. *Let P, Q be dMTSs. A relation $\mathcal{R} \subseteq P \times Q$ is a* (strong) alternating refinement relation (as-relation) *if for all $(p, q) \in \mathcal{R}$:*

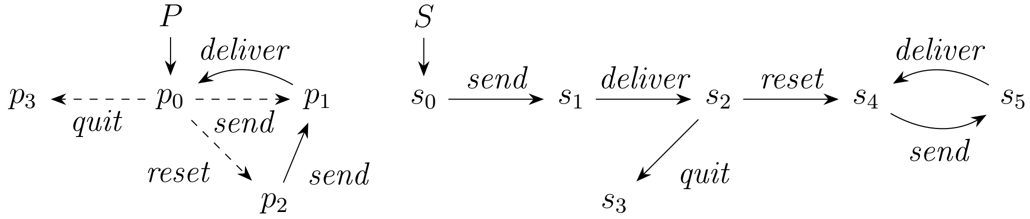

Figure 2.1: An example MTS P and one of its as-implementations S.

1. *$q \xrightarrow{\alpha} Q'$ implies $\exists P' : p \xrightarrow{\alpha} P'$ and $\forall p' {\in} P' : \exists q' {\in} Q' : (p', q') \in \mathcal{R}$,*

2. *$p \overset{\alpha}{\dashrightarrow} p'$ implies $\exists q' : q \overset{\alpha}{\dashrightarrow} q'$ and $(p', q') \in \mathcal{R}$,*

Similarly, if P and Q are MTSs, $\mathcal{R} \subseteq P \times Q$ is a weak alternating refinement relation (was-relation) *if for all $(p, q) \in \mathcal{R}$:*

1. *$q \xrightarrow{\alpha} q'$ implies $\exists P' : p \overset{\hat{\alpha}}{\Longrightarrow} p'$ and $(p', q') \in \mathcal{R}$,*

2. *$p \overset{\alpha}{\dashrightarrow} p'$ implies $\exists q' : q \overset{\hat{\alpha}}{\Longrightarrow} q'$ and $(p', q') \in \mathcal{R}$,*

We write $p \sqsubseteq_{as} q$ ($\sqsubseteq_{w\text{-}as}$) and say that p (strongly) as-refines *q (was-refines) if there exists an as-refinement (was-refinement) relation $\mathcal{R}$ such that $(p, q) \in \mathcal{R}$. Furthermore, we extend this notion to dMTSs (only MTSs for was-refinement) and write $P \sqsubseteq_{as} Q$ if $p_0 \sqsubseteq_{as} q_0$.*

As usual, $\sqsubseteq_{as}$ is the greatest as-relation and a preorder, i.e. $\sqsubseteq_{as}$ is a refinement relation. Recall that for a *refinement relation* $\sqsubseteq_R$, the set of *R-implementations* of a dMTS P is $impl_R(P) = \{S \mid S \sqsubseteq_R P, S \text{ is an implementation}\}$.

As an example consider Fig. 2.1. First observe that S is an implementation, since all may-transitions are must-transitions as well. It is an as-refinement, and hence an as-implementation, of P ($S \sqsubseteq_{as} P$, $S \in impl_{as}(P)$ respectively) because of the relation $R = \{(s_0, p_0), (s_1, p_1), (s_2, p_0), (s_3, p_3), (s_4, p_2), (s_5, p_1), (s_4, p_0)\}$. Note that as-refinement allows for unfolding of cycles and at each unfolding different choices can be made w.r.t. the same may-transition. In our example the cycle p_0, p_1 is unfolded to s_0, s_1, s_2; s_0 implements only the *send*-transition, while s_2 implements only the *reset*- and the *quit*-transition. Another interesting point is that s_4 implements both, p_0 and p_2, as can be seen from R.

For further small examples regarding as-refinement, see Fig. 2.2 and Fig. 2.5 below.

It is obvious that, on the one hand, our dMTS-setting with as-refinement is a generalization of the MTS-setting with as-refinement (which in turn generalizes LTS with bisimulation equivalence). On the other hand, a disjunctive must-transition in a dMTS describes a choice between transitions with the same action and different target states; so a dMTS is a special DMTS (cf. e.g. [44, 7]), where a disjunctive must-transition describes a choice between transitions with different actions and different target states.

2.3 Deadlock-Testing

The idea of testing a system P as proposed in [30] (and briefly mentioned in Sect. 1.1) is that P is put in parallel with a test system, and then the overall system is checked for a desirable property like deadlock freedom ($\nexists p \forall a \in \Sigma.\, p \overset{a}{\not\Longrightarrow}$, cf. Sect. 1.1). If these systems are MTSs or even dMTSs, it is not obvious what deadlock in the presence of may-transitions really means. Therefore, we propose the following modification: Only LTSs are considered as test systems, and such a test system is not composed with P itself, but with all its implementations. When all these parallel compositions are deadlock free, P passes the test. Then, as usual, P is considered a refinement of P' iff P passes at least all the tests passed by P'. Intuitively, any user/test environment that communicates well, i.e. in a deadlock-free manner, with (all implementations of) P' will also communicate well with P. Thus replacing P' by P will not introduce a deadlock in any environment. This testing preorder captures what behaviour of a dMTS is relevant if we use it as a component and want to develop deadlock-free systems.

Thus, our testing scenario uses parallel composition and the notion of deadlock-freedom only for implementations. This allows us to study what these should mean for dMTSs, instead of prescribing them. The only problem with this idea is that it is not clear what an implementation of P is – in particular, since the approach is about finding an adequate refinement and, thus, implementation notion. Since it is a commonly accepted and strict notion of implementation, we use as-implementations; it seems undoubtable that these really should be implementations of P.

As we show after the characterization results, we could have opted for a range of weaker refinement (and resulting implementation) notions than $\sqsubseteq_{as}$. Refinement notions that allow additional actions in a refinement (e.g. weak-alphabet-refinement of [34]) are out of the scope of this chapter.

Definition 2.3. *A* test (T, A) *is a pair consisting of an implementation* T *and a set* $A \subseteq \Sigma$. *For a refinement relation* $\sqsubseteq_r$, *a dMTS* P r-deadlock-satisfies *a test* (T, A) *if* $S \parallel_A T$ *is deadlock free for each* $S \in \mathit{impl}_r(P)$.

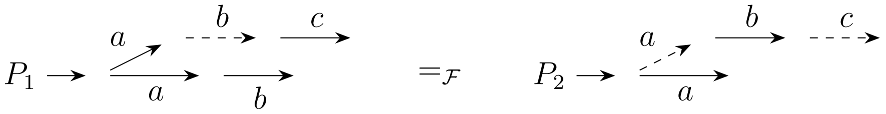

Figure 2.2: $\mathcal{F}$-equivalent systems P_1 and P_2

Then, we write $P\ sat_r^d\ (T, A)$.
A dMTS P r-deadlock-refines P' ($P \leq_r^d P'$)*, if for all tests* (T, A) :

$$P'\ sat_r^d\ (T, A) \implies P\ sat_r^d\ (T, A)$$

This testing preorder is intuitively well justified but hard to work with, since it refers to all possible test environments, i.e. an infinite number of tests. We will characterize this preorder by failure-set inclusion, which only refers to the systems that are compared. For ease of understanding, we will first develop this characterization for MTS and then extend it to the more intricate one for dMTS.

2.3.1 F-Semantics for MTS

Definition 2.4. *The* failure semantics *of an MTS* P *is* $\mathcal{F}(P) = \{(w, X) \mid p_0 \stackrel{w}{\Longrightarrow} p, \forall a \in X : p \stackrel{a}{\not\Longrightarrow}\}$. p failure- *(or* $\mathcal{F}$*-)*refines P'*, written* $P \sqsubseteq_{\mathcal{F}} P'$*, if* $\mathcal{F}(P) \subseteq \mathcal{F}(P')$. P *and* P' *are* failure- *(or* $\mathcal{F}$*-)*equivalent*, written* $P =_{\mathcal{F}} P'$*, if* $\mathcal{F}(P) \sqsubseteq_{\mathcal{F}} \mathcal{F}(P')$ *and* $\mathcal{F}(P') \sqsubseteq_{\mathcal{F}} \mathcal{F}(P)$. *We call* $(w, X) \in \mathcal{F}(P)$ *a refusal pair,* X *a* refusal set *and say that the respective* p can refuse X.

This definition generalizes failure semantics of LTS found e.g. in [70, 65] and Sect. 1.1. Intuitively a refusal pair (w, X) represents a partial deadlock (locking out all actions in X), which can occur after trace w. Observe that the trace is based on may-transitions, while the refusal set is based on must-transitions.

For a small example consider P_1 and P_2 in Fig. 2.2, where $\{(\varepsilon, \{b, c\})$, $(a, \{a, b, c\}), (ab, \{a, b, c\}), (abc, \{a, b, c\})\}$ are the elements of $\mathcal{F}(P_1) = \mathcal{F}(P_2)$ with maximal refusal sets; e.g. $(a, \{b\})$ is in $\mathcal{F}(P_1)$ due to the upper a-transition and in $\mathcal{F}(P_2)$ due to the lower one. So both, P_1 and P_2 are $\mathcal{F}$-refinements of each other, but neither as- (or was-)refines the other.

As another example, consider Fig. 2.1. There, we have $(\mathit{send\ deliver\ reset\ send\ deliver}, \{\mathit{reset}, \mathit{deliver}\}) \in \mathcal{F}(P) \cap \mathcal{F}(S)$, whereas $(\mathit{send\ deliver}, \{\mathit{reset}\})$ is only in $\mathcal{F}(P)$. For $A = \Sigma \setminus \{\omega\}$ and T in Fig. 2.3, (T, A) tests for the absence of $(\mathit{send\ deliver}, \{\mathit{reset}\})$: $S \parallel_A T$ is isomorphic to T which, obviously, cannot deadlock. Nevertheless, P does not pass (T, A), due to $S' \in \mathit{impl}_{as}(P)$

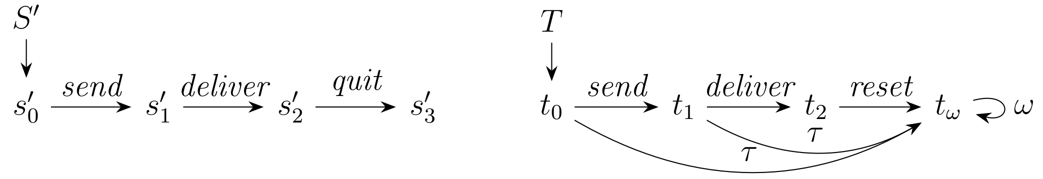

Figure 2.3: Another implementation P' and a test T.

in Fig. 2.3 (among others); the resulting $S' \parallel_A T$ is isomorphic to T without the *reset*-transition and thus can reach the deadlock (s'_2, t_2).

The next two lemmas prepare the main result of this section, which characterizes our refinement notion. The first one deals with the additional twist of our testing scenario that we consider implementations of the MTS under test. It shows that the $\mathcal{F}$-semantics of an MTS is the collection of all refusal pairs of its implementations.

Lemma 2.5. $\mathcal{F}(P) = \bigcup_{S \in \mathit{impl}_{as}(P)} \mathcal{F}(S)$

Proof. '$\supseteq$' For any $S \in \mathit{impl}_{as}(P)$, $(w, X) \in \mathcal{F}(S)$ implies $s_0 \overset{w}{\Longrightarrow} s \wedge \forall a \in X : s \overset{a}{\not\Longrightarrow}$. Because of $S \sqsubseteq_{w\text{-}as} P$, we conclude that $p_0 \overset{w}{\Longrightarrow} p$ for some p with $s \sqsubseteq_{w\text{-}as} p$ by Def. 2.2.2. Now Def. 2.2.1 implies $\forall a \in X : p \overset{a}{\not\Longrightarrow}$. By definition this means $(w, X) \in \mathcal{F}(P)$. '$\subseteq$' is equivalent to $(w, X) \in \mathcal{F}(P) \implies \exists S \in \mathit{impl}_{as}(P) : (w, X) \in \mathcal{F}(S)$. We give a construction for such an S.

Let $(w, X) \in \mathcal{F}(P)$ due to $p_0 \overset{v}{\dashrightarrow} p$ with $w = \widehat{v}$; let n be the length of v. To construct S, we unfold P to some degree using $n+1$ copies of P.

We construct $S = (S, (p_0, 0), \longrightarrow_S, \longrightarrow_S)$ where:

- $S = P \times \{0, 1, \ldots, n\}$
- $\longrightarrow_S = \{((p, i), \alpha, (p', i+1)) \mid p \overset{\alpha}{\dashrightarrow}_P p', 0 \leq i < n\} \cup$
  $\{(p, n), \alpha, (p', n)) \mid p \overset{\alpha}{\longrightarrow}_P p'\}$

Now $S \sqsubseteq_{as} P$ due to $R = \{((p, i), p) \mid 0 \leq i \leq n\}$. In particular, the run $p_0 \overset{\alpha_1}{\dashrightarrow} p_1 \overset{\alpha_2}{\dashrightarrow} \cdots \overset{\alpha_n}{\dashrightarrow} p_n = p$ in P with $v = \alpha_1 \cdots \alpha_n$ is simulated in S by $(p_0, 0) \xrightarrow{\alpha_1} (p_1, 1) \xrightarrow{\alpha_2} \cdots \xrightarrow{\alpha_n} (p_n, n)$, and $p = p_n$ and (p_n, n) have the same must-transitions. □

The second lemma exhibits a characteristic test for a refusal pair. In fact, LTS T in Fig. 2.3 is the characteristic test for $(\mathit{send}\ \mathit{deliver}, \{\mathit{reset}\})$.

Lemma 2.6. *(cf. [55, 70]) Let $w = w_1 \cdots w_n \in \Sigma^*$, $X \subseteq \Sigma$ and $T(w, X)$ as depicted in Fig. 2.4. For all implementations S where an action ω does not occur, the system $S \parallel_{\Sigma \setminus \{\omega\}} T(w, X)$ can deadlock iff $(w, X) \in \mathcal{F}(S)$.*

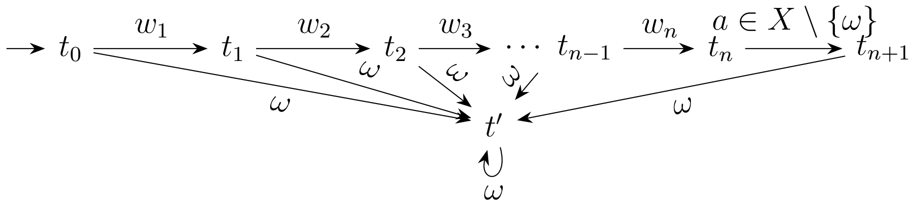

Figure 2.4: $T(w, X)$ with $w = w_1 \cdots w_n$.

Proof. Since $T(w, X)$ can almost always reach t' and perform arbitrarily many ωs, the compositions can deadlock iff $s_0 \overset{w}{\Longrightarrow} s$ where s can refuse X. □

Theorem 2.7. *P as-deadlock refines P' if and only if $P \sqsubseteq_{\mathcal{F}} P'$.*

Proof. '⇒': Take $(w, X) \in \mathcal{F}(P)$ and choose a fresh ω not occurring in P or P'. Let $A = \Sigma \setminus \{\omega\}$. By Lemma 2.5 we know that there is some as-implementation S of P with $(w, X) \in \mathcal{F}(S)$. Therefore, $S \parallel_A T(w, X)$ can deadlock by Lemma 2.6 and thus P does not satisfy $(T(w, X), A)$. Now our premise implies that P' does not satisfy $(T(w, X), A)$ either and, therefore, there must be an as-implementation S' of P' such that $S' \parallel_A T(w, X)$ can deadlock. Again by Lemma 2.6 we get $(w, X) \in \mathcal{F}(S')$ and $\mathcal{F}(S') \subseteq \mathcal{F}(P')$ by Lemma 2.5.

'⇐': Let $P \sqsubseteq_{\mathcal{F}} P'$. We prove that P refines P'; by contraposition, we assume for some test (T, A) that $\neg P\, sat_{\text{as}}(T, A)$. Thus there must be an as-implementation S of P such that $S \parallel_A T$ can deadlock, i.e. $(w, \Sigma) \in \mathcal{F}(S \parallel_A T)$. Since these are LTS, this implies $\exists (u, Y) \in \mathcal{F}(S) \subseteq \mathcal{F}(P) \subseteq \mathcal{F}(P'), (v, Z) \in \mathcal{F}(T) : w \in u \parallel_A v$ and $\Sigma \subseteq (Y \cap Z) \cup (A \cap (Y \cup Z))$ by Lemma 1.7. By Lemma 2.5 we know that there must be some as-implementation S' of P' with $(u, Y) \in \mathcal{F}(S')$. Again by Lemma 1.7 we get $(w, \Sigma) \in \mathcal{F}(S' \parallel_A T)$. This means that $S' \parallel_A T$ can deadlock and $\neg P'\, sat_{\text{as}}(T, A)$. □

Our $\mathcal{F}$-refinement is strictly coarser than both, strong and weak, as-refinement. Obviously, this means $impl_{\mathcal{F}}(P) \subseteq impl_{w\text{-}as}(P) \subseteq impl_{as}(P)$ for all MTS P. We do not have equality since otherwise the thorough refinements would coincide; see Sect. 2.5 later on. This shows that alternating simulations disallow some implementations that are perfectly correct from our semantical viewpoint.

Proposition 2.8. *$\mathcal{F}$-refinement is strictly coarser than weak as-refinement which in turn is strictly coarser than as-refinement, i.e. $P \sqsubseteq_{as} P' \implies P \sqsubseteq_{w\text{-}as} P' \implies P \sqsubseteq_{\mathcal{F}} P'$, but $P \sqsubseteq_{as} P' \not\Longleftarrow P \sqsubseteq_{w\text{-}as} P' \not\Longleftarrow P \sqsubseteq_{\mathcal{F}} P'$.*

Proof. The fact that weak as-refinement is strictly coarser than as-refinement can easily be seen: An alternating simulation is also a weak one, by definition. On the other hand an MTS P which can only perform $p_0 \xrightarrow{\tau a}$ was-refines an MTS Q which can only perform $q_0 \xrightarrow{a}$, but not strongly.

So let us consider $\mathcal{F}$-refinement and weak as-refinement.

‘$\Rightarrow$’: $(w, X) \in \mathcal{F}(P)$ implies $p_0 \overset{w}{\Longrightarrow} p \wedge \forall a \in X : p \overset{a}{\not\Longrightarrow}$. Because of $P \sqsubseteq_{w\text{-}as} P'$, we conclude that $p'_0 \overset{w}{\Longrightarrow} p'$ for some p' with $p \sqsubseteq_{w\text{-}as} p'$ by Def. 2.2.2. Now Def. 2.2.1 implies $\forall a \in X : p' \overset{a}{\not\Longrightarrow}$. By definition this means $(w, X) \in \mathcal{F}(P')$.

‘$\not\Leftarrow$’: We can consider systems without must-transitions, where $\mathcal{F}$-inclusion equates to language inclusion (arbitrary X can be refused) and alternating simulation to simulation. It is well known that simulation is stricter than language inclusion. □

We have already seen that $\mathcal{F}$-refinement does not imply as- or weak as-refinement. As in the standard LTS-setting, the situation is different for deterministic MTS. An MTS is *deterministic* if it is τ-free (has no τ-may-transitions) and the may-transition relation is deterministic, i.e. $\forall a \in \Sigma : p \overset{a}{\dashrightarrow} p' \wedge p \overset{a}{\dashrightarrow} p'' \implies p' = p''$.

Proposition 2.9. *For an MTS P and a deterministic MTS Q, we have $P \sqsubseteq_{\mathcal{F}} Q \Leftrightarrow P \sqsubseteq_{w\text{-}as} Q$. If, in addition, P is τ-free, then $P \sqsubseteq_{\mathcal{F}} Q \Leftrightarrow P \sqsubseteq_{as} Q$.*

Proof. ‘$\Leftarrow$’ is implied by Proposition 2.8 for arbitrary MTSs.

‘$\Rightarrow$’: For a deterministic Q, $\mathcal{R} = \{(p, q) \mid \exists w \in \Sigma^* : p_0 \overset{w}{\Longrightarrow} p', q_0 \overset{w}{\dashrightarrow} q'\}$ is a weak alternating simulation: For any $(p, q) \in \mathcal{R}$ with $p \overset{a}{\dashrightarrow} p'$ we know that $(wa, \emptyset) \in \mathcal{F}(P)$ and by assumption $(wa, \emptyset) \in \mathcal{F}(Q)$. Therefore, and since Q is deterministic, $q_0 \overset{w}{\dashrightarrow} q \overset{a}{\dashrightarrow}$. For any $(p, q) \in \mathcal{R}$ with $q \xrightarrow{a} q'$ we know that $(w, \{a\}) \notin \mathcal{F}(Q)$ and by assumption $(w, \{a\}) \notin \mathcal{F}(P)$. Therefore $p \overset{a}{\Longrightarrow}$. In both cases the states reached are related by $\mathcal{R}$ again.

If, in addition, also P has no τ-transition, weak and strong alternating simulations coincide. □

We developed our testing approach, choosing the implementations of an MTS to be defined according to as-refinement; this approach justified $\mathcal{F}$-refinement between MTSs. In this sense as-refinement supports $\mathcal{F}$-refinement and not itself. In Cor. 2.12 below, we show that using implementations based on $\mathcal{F}$-refinement again leads to $\mathcal{F}$-refinement, thus $\mathcal{F}$-refinement supports itself.

Definition 2.10. *An MTS P $\mathcal{F}$-*satisfies *a test (T, A) if $S \parallel_A T$ is deadlock free for all $S \in \mathit{impl}_{\mathcal{F}}(P)$. We then write $P\, \mathit{sat}_{\mathcal{F}}(T, A)$.*

We can observe that testing based on $\mathcal{F}$-implementations coincides with the one based on as-implementations already at the stage of test-satisfaction. This implies the first main result of this section in Cor. 2.12 below.

Proposition 2.11. *For an MTS P and a test (T, A), we have $P\, \mathit{sat}_{\mathcal{F}}(T, A) \Leftrightarrow P\, \mathit{sat}_{\mathrm{as}}(T, A)$.*

Proof. '$\Rightarrow$': By Proposition 2.8 every as-refinement is an $\mathcal{F}$-refinement.
'$\Leftarrow$': $\neg P\, \mathit{sat}_{\mathcal{F}}(T, A)$ implies the existence of an $\mathcal{F}$-implementation S, such that $S \parallel_A T$ can deadlock. This is because of some $(w, \Sigma) \in \mathcal{F}(S \parallel_A T)$, which implies the existence of an appropriate $(v, X) \in \mathcal{F}(S)$ by Proposition 2.29. Then $(v, X) \in \mathcal{F}(P)$ by definition of $\mathcal{F}$-refinement and, by Lemma 2.5, there is an $S' \in \mathit{impl}_{\mathit{as}}(P)$ with $(v, X) \in \mathcal{F}(S')$. Therefore, $S' \parallel_A T$ can also deadlock because of (w, Σ) and thus $\neg P\, \mathit{sat}_{\mathrm{as}}(T, A)$. □

Corollary 2.12. *$P \sqsubseteq_{\mathcal{F}} P'$ if and only if for all tests (T, A) :
$P'\, \mathit{sat}_{\mathcal{F}}(T, A) \implies P\, \mathit{sat}_{\mathcal{F}}(T, A)$.*

Proof. Immediate from Thm. 2.7 and Proposition 2.11. □

This corollary shows that $\mathcal{F}$-refinement supports itself in our testing scenario. Actually, we could also associate each MTS P with any set $\mathit{impl}(P)$ of implementations satisfying $\mathit{impl}_{\mathit{as}}(P) \subseteq \mathit{impl}(P) \subseteq \mathit{impl}_{\mathcal{F}}(P)$ and define test satisfaction w.r.t. these sets. This *impl*-satisfaction is implied by $\mathcal{F}$- and implies as-satisfaction by choice, so all three coincide by Prop. 2.11. Hence, analogously to Cor. 2.12, also *impl* supports $\mathcal{F}$-refinement. For example, we could use $\mathit{impl}_{\mathit{w\text{-}as}}(p)$ – the set of weak-as-implementations of P – or we could work with implementations according to a branching-version of weak-as-refinement as argued for in [34]. For the definition of refinement, we could even use an implication like: $P'\, \mathit{sat}_{\mathrm{as}}(T, A) \implies P\, \mathit{sat}_{\mathcal{F}}(T, A)$ with the same result.

At the beginning of Sect. 2.3, we discussed the use of as-implementations in Def. 2.3. The above consideration shows that our choice is not so crucial, and that we can claim quite some generality for our approach.

Thoroughness is another, quite prominent (and conceptually easier) notion of a refinement supporting itself.

Here, we have the same picture as above: as-refinement is not thorough (see e.g.[43]), but $\mathcal{F}$-refinement is, as we show below in Thm. 2.14 with the help of Lemma 2.13 – just as Lemma 2.5 was needed in the proof of Thm. 2.7.

Lemma 2.13. $\mathcal{F}(P) = \bigcup_{S \in \mathit{impl}_{\mathit{w\text{-}as}}(P)} \mathcal{F}(S) = \bigcup_{S \in \mathit{impl}_{\mathcal{F}}(P)} \mathcal{F}(S)$

Proof. 1.'$\subseteq$': $(w,X) \in \mathcal{F}(P) \implies \exists S \in \mathit{impl}_{as}(P) : (w,X) \in \mathcal{F}(S)$ by Lemma 2.5. Since by Proposition 2.8 $\mathit{impl}_{as}(P) \subseteq \mathit{impl}_{w\text{-}as}(P)$, we are done.
2.'$\subseteq$': Immediate since by Proposition 2.8 $\mathit{impl}_{w\text{-}as}(P) \subseteq \mathit{impl}_{\mathcal{F}}(P)$.
3. The inclusion of the last in the first set is implied by the definition of $\sqsubseteq_{\mathcal{F}}$. □

Theorem 2.14. *$P \sqsubseteq_{\mathcal{F}} P' \Leftrightarrow \mathit{impl}_{\mathcal{F}}(P) \subseteq \mathit{impl}_{\mathcal{F}}(P')$*

Proof. '$\Rightarrow$': Let S be an $\mathcal{F}$-implementation of P, then $S \sqsubseteq_{\mathcal{F}} P \sqsubseteq_{\mathcal{F}} P'$ by our premise. Thus S is an $\mathcal{F}$-implementation of P'.

'$\Leftarrow$': Take an arbitrary $(w,X) \in \mathcal{F}(P)$. By Lemma 2.13 we know that there is an $\mathcal{F}$-implementation S of P with $(w,X) \in \mathcal{F}(S)$. By our premise, S is also an $\mathcal{F}$-implementation of P' and thus, again by Lemma 2.13, $(w,X) \in \mathcal{F}(P')$. □

2.3.2 F-Semantics for dMTS

We now turn our attention to dMTS in general. Again, our aim is to show that $\leq^d_{as}$ and $\leq^d_{\mathcal{F}}$ coincide, and to characterize them with a kind of failure semantics on dMTS. For this, we first need to further generalize the $\mathcal{F}$-semantics, which we already extended to MTS above.

We could use the same definition as before for dMTS, but its meaning, of course, would depend on the notion of $\overset{a}{\Longrightarrow}$ for dMTS. With $\overset{a}{\Longrightarrow}$ defined as in [52], the resulting semantics characterizes $\leq^d_{as}$ only for dMTS without disjunctive τ-transitions as shown in [62].

To obtain a suitable semantics for general dMTS, we now describe an alternative definition and generalize it to dMTS; this will lead to the desired results. In order to differentiate between the 'traditional' failure semantics (for LTS and MTS) and our new one (for dMTS), we will denote the former by $\mathcal{F}_t$ for the remainder of this section.

The key concept for our alternative is that of a ready set. With the usual definition, the *ready set* of some state p of an MTS P is simply defined as $\mathit{ready}_t(p) = \{a \in \Sigma \mid p \overset{a}{\Longrightarrow}\}$. The complement of $\mathit{ready}_t(p)$ (and each subset of the complement) is a refusal set for traces w with $p_0 \overset{w}{\Longrightarrow} p$. Thus, given a trace w, to determine all X with $(w,X) \in \mathcal{F}_t(P)$, it suffices to consider the states with minimal ready sets among the states p with $p_0 \overset{w}{\Longrightarrow} p$.

For dMTS we define $\mathit{readies}(p)$ as a set of sets of actions. The sets of actions are ready sets of implementations of p, and all the minimal ones among these ready sets are contained in $\mathit{readies}(p)$. Each $X \in \mathit{readies}(p)$ – called a *ready* of p – arises from a τ-choice function f; the idea is that $f(p)$ describes a minimal way to implement the disjunctive τ-must-transitions starting from p.

Definition 2.15. *Let P be a dMTS. A τ-*choice *f is a function that selects, for each $p \in P$, a minimal subset $f(p) \subseteq P$ such that $\forall p \xrightarrow{\tau} P' : \exists p' \in P' \cap f(p)$. We say that f* reaches a state p_n from p_1 *if $p_1 \overset{\tau}{\dashrightarrow} p_2 \overset{\tau}{\dashrightarrow} \cdots p_n$ such that $p_{i+1} \in f(p_i)$ for $i = 1, \ldots, n-1$.*

The set of readies of a state p_1 is $readies(p_1) = \{Z \subseteq \Sigma \mid \exists \tau\text{-choice } f : a \in Z \Leftrightarrow f$ reaches some p_n from p_1 with $p_n \xrightarrow{a}\}$

The failure semantics *of P is defined as $\mathcal{F}(P) = \{(w, X) \mid \exists p_0 \overset{w}{\Longrightarrow} p : \exists Z \in readies(p) : X \cap Z = \emptyset\}$. We say that (w, X) 'is justified by' or 'in $\mathcal{F}(P)$ due to' the state p, the τ-choice f and the resulting Z in the definition of $\mathcal{F}(P)$. Further, P $\mathcal{F}$-*refines *another dMTS P' ($P \sqsubseteq_{\mathcal{F}} P'$) if $\mathcal{F}(P) \subseteq \mathcal{F}(P')$.*

Note that, in this definition, the run $p_0 \overset{w}{\Longrightarrow} p$ might use a τ-transition not selected by f, e.g. because it is a may-transition that is not underlying any must-transition.

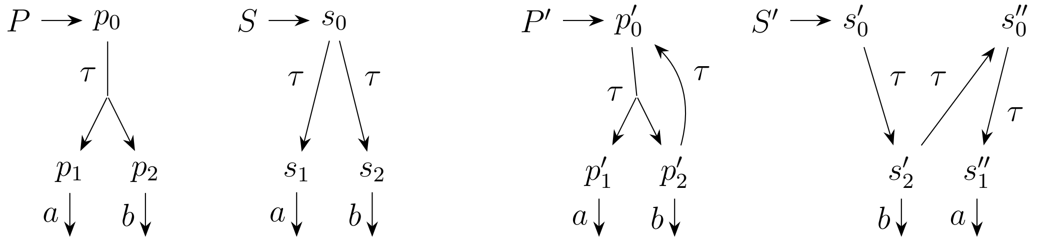

Figure 2.5: Two dMTSs P and P' with two as-implementations S and S'

The small examples in Fig. 2.5 illustrate these points. For dMTS P, there are only two τ-choices, which choose the left or the right τ-branch, yielding $readies(p_0) = \{\{a\}, \{b\}\}$. We ignore the as-implementation S, which would lead us to add $\{a, b\}$ to $readies(p_0)$; note that this additional ready would not change $\mathcal{F}(P)$ according to Def. 2.15, since $\{a\} \subseteq \{a, b\}$. Also for dMTS P', there are only two τ-choices, yielding $readies(p'_0) = \{\{a\}, \{b\}\}$. In the as-implementation S', p'_0 is implemented by s'_0 and s''_0, and both of these choose one τ-branch. Again, we do not consider this as-implementation, since it is an unfolding with additional states.

A τ-choice describes an implementation only as far as τ-transitions are concerned: some τ-may-transitions are turned into τ-must-transitions, others are deleted – and the states are not modified. Thus, our approach is far from exhaustively searching through all implementations of P. In particular, for finite P, all sets $readies(p)$ can be computed by exhaustively searching through all τ-choices. Once we have characterized the testing preorder $\leq^d_{as}$, we will sketch how the $\mathcal{F}$-semantics can be used to decide it.

As an aside, we mention that some choice-functions – very different from our τ-choices – are defined in [33]. They are used for a simulation-style refinement (instead of a denotational-style characterization) and choose one 'branch' from *every* disjunctive transition. For instance for two equally labelled transitions from some state q to $\{1, 2\}$, $\{2, 3\}$ respectively, 1 and 2 could be chosen, which is not minimal.

It is easy to see that we have indeed extended the previous $\mathcal{F}_t$-semantics to dMTS. Thus, the usual characterization for deadlock freedom of LTSs carries over to the (intentionally) new $\mathcal{F}$-semantics.

Proposition 2.16. *For every MTS P, we have $\mathcal{F}(P) = \mathcal{F}_t(P)$.*

Proof. Since P has no disjunctive τ-transitions, the only possible τ-choice selects the only target state for each τ-must-transition. This implies that for every $p \in P$: $\mathit{readies}(p) = \{\{a \mid p \xrightarrow{\tau^* a}\}\} = \{\mathit{ready}_t(p)\}$. Thus, for any state p with $p_0 \overset{w}{\Longrightarrow} p$, the sets X with $X \cap \{a \mid p \xrightarrow{\tau^* a}\} = \emptyset$ are exactly the sets where $p \overset{a}{\not\Longrightarrow}$ for all $a \in X$.

□

The next lemma is a key-result for connecting the $\mathcal{F}$-semantics to our implementation-oriented testing. It is essentially the extension of Lem. 2.13. It also deals with the more complicated aspects of the $\mathcal{F}$-semantics for dMTS.

Lemma 2.17. *1. $\mathcal{F}(P) = \bigcup_{S \in \mathit{impl}_{as}(P)} \mathcal{F}(S)$*

2. $\mathcal{F}(P) = \bigcup_{S \in \mathit{impl}_{\mathcal{F}}(P)} \mathcal{F}(S)$

Proof. 1.'$\subseteq$': Let $(w, X) \in \mathcal{F}(P)$ be justified by p_1, f and Z and $p_0 \overset{v}{\dashrightarrow} p_1$ with $w = \hat{v}$. As in the MTS-case (cf. 2.5, we construct an as-implementation S by unfolding P to the level n, where n is the length of v. For this unfolding, we use $n + 1$ copies of P.

Let $S = (S, (p_0, 0), \longrightarrow_S, \longrightarrow_S)$ where

- $S = P \times \{0, 1, \ldots, n\}$
- $\longrightarrow_S = \{((p, i), \alpha, (p', i + 1)) \mid p \overset{\alpha}{\dashrightarrow}_P p', 0 \le i < n\} \cup$
  $\{((p, n), \alpha, (p', n)) \mid p \xrightarrow{\alpha}_P P', p' \in P', p' \in f(p) \vee \alpha \neq \tau\}$

Now $S \sqsubseteq_{as} P$ due to $\mathcal{R} = \{((p, i), p) \mid p \in P, 0 \le i \le n\}$ and $(p_0, 0) \overset{w}{\Longrightarrow} (p_1, n)$.

Clearly, $(p_1, n) \overset{\varepsilon}{\Longrightarrow} (p, n)$ iff f reaches p from p_1. Thus, $Z = \mathit{ready}_t(p_1, n)$ and $(w, X) \in \mathcal{F}_t(S) = \mathcal{F}(S)$.

1.'$\supseteq$': Consider $S \in \mathit{impl}_{as}(P)$ according to as-relation $\mathcal{R}$ and $(w, X) \in \mathcal{F}_t(S)$ due to some $s_0 \overset{w}{\Longrightarrow} s_1$. Clearly, we have some p_1 with $p_0 \overset{w}{\Longrightarrow} p_1$ and $(s_1, p_1) \in \mathcal{R}$. Let P_τ be defined as the smallest set containing p_1 and fulfilling $p \in P_\tau \wedge p \xrightarrow{\tau} P' \implies P' \subseteq P_\tau$, and let S_τ be defined analogously from s_1. Let $Z_S = \{a \in \Sigma \mid \exists s \in S_\tau : s \xrightarrow{a}\} = \mathit{ready}_t(s_1)$. Thus $X \cap Z_S = \emptyset$ by choice of s_1.

In the following, we will call $p \in P_\tau$ *matched (by s)* if there is some $s \in S_\tau$ such that $(s, p) \in \mathcal{R}$. If $p \xrightarrow{\tau} P'$ for such a p, some $p' \in P'$ is also matched – due to Def. 2.2.1 – by some s' with $s \xrightarrow{\tau} s'$. Hence, there is a τ-choice f for P with the following property: For every matched $p \in P_\tau$ and every transition $p \xrightarrow{\tau} P'$, all $p' \in P' \cap f(p)$ are also matched. Note that p_1 is matched by s_1.

Let $Z_P \in \mathit{readies}(p_1)$ be the ready resulting from this f. Consider some $a \in Z_P$ due to $p_1 \overset{\tau}{\dashrightarrow} p_2 \overset{\tau}{\dashrightarrow} \cdots p_n \xrightarrow{a}$ as in the definition of $\mathit{readies}(p_1)$. Since p_1 is matched, so is p_n by choice of f – by s_n, say. Hence, $s_n \in S_\tau$ and $s_n \xrightarrow{a}$, i.e. $a \in Z_S$. Therefore $Z_P \subseteq Z_S$, $X \cap Z_P = \emptyset$, $(w, X) \in \mathcal{F}(P)$ and we are done.

2.'$\subseteq$': Any as-implementation is also an $\mathcal{F}$-implementation by '1.$\supseteq$', and we are done by '1.$\subseteq$'. 2.'$\supseteq$'follows immediately from $S \in \mathit{impl}_{\mathcal{F}}(P)$. □

In the previous section, we proved a result for MTSs, analogous to Prop. 2.17, using that as-refinement implies $\mathcal{F}$-refinement. Here, it seems easier to proceed the other way round.

Corollary 2.18. *For dMTSs P and Q, $P \sqsubseteq_{as} Q \implies P \sqsubseteq_{\mathcal{F}} Q$. This implication is strict.*

Proof. For each $(w, X) \in \mathcal{F}(P)$, there is $S \in \mathit{impl}_{as}(P)$ with $(w, X) \in \mathcal{F}(S)$ by Prop. 2.17. By transitivity of $\sqsubseteq_{as}$ we know that $S \in \mathit{impl}_{as}(Q)$ and $(w, X) \in \mathcal{F}(Q)$, again by Prop. 2.17.

The strictness of this implication already holds for MTS 2.8. Indeed it can already be seen on LTS, where as-refinement equals bisimilarity. □

After these preparations we will prove that $\mathcal{F}$-refinement characterizes $\leq^d_{as}$. To achieve this, we first make three points, summed up in the lemma below; the first two already appeared previously. The first point describes, how the $\mathcal{F}$-semantics of two LTSs are related to the $\mathcal{F}$-semantics of their parallel composition (Lem. 1.7).

The second point describes a characteristic test system $T(w, X)$, which only deadlocks if the tested implementation's $\mathcal{F}$-semantics contains (w, X) (Lemma 2.6).

Finally, we observe that the same tests are satisfied, no matter whether we use $\mathcal{F}$-implementations or as-implementations.

Lemma 2.19. *1. For LTSs P and Q and $A \subseteq \Sigma$: $\mathcal{F}(P \parallel_A Q) = \{(w, X) \mid (u, Y) \in \mathcal{F}(P), (v, Z) \in \mathcal{F}(Q), w \in u \parallel_A v, X \subseteq (Y \cap Z) \cup (A \cap (Y \cup Z)) =: \mathcal{F}(P) \parallel_A \mathcal{F}(Q)\}$.*

2. Let $w = w_1 \cdots w_n \in \Sigma^*$, $X \subseteq \Sigma$ and $T(w, X)$ be as depicted in Fig. 2.4. For each implementation S whose alphabet does not contain an action ω, the LTS $S \parallel_{\Sigma \setminus \{\omega\}} T(w, X)$ can deadlock if and only if $(w, X) \in \mathcal{F}(S)$.

3. For each dMTS P and test (T, A), $P \ sat^d_{\mathcal{F}} \ (T, A) \Leftrightarrow P \ sat^d_{as} \ (T, A)$.

Proof. Since the first two claims have been proven before, we only show the third: $\neg P \ sat^d_{\mathcal{F}} \ (T, A) \Leftrightarrow \neg P \ sat^d_{as} \ (T, A)$.
'$\Leftarrow$': If P fails the test due to the as-implementation S, then S is also an $\mathcal{F}$-implementation by Cor. 2.18.
'$\Rightarrow$': $\neg P \ sat^d_{\mathcal{F}} \ (T, A)$ implies the existence of an $\mathcal{F}$-implementation $S_{\mathcal{F}}$, such that $S_{\mathcal{F}} \parallel_A T$ can deadlock. This means that there is some $(w, \Sigma) \in \mathcal{F}(S_{\mathcal{F}} \parallel_A T)$ (cf. Lem. 1.7), which implies the existence of appropriate $(u, Y) \in \mathcal{F}(S_{\mathcal{F}})$ and $(v, Z) \in \mathcal{F}(T)$ by Lemma 2.19.1. Then $(u, Y) \in \mathcal{F}(P)$ by definition of $\mathcal{F}$-implementation and, by Prop. 2.17, there is an $S_{as} \in impl_{as}(P)$ with $(u, Y) \in \mathcal{F}(S_{as})$. Therefore, $S_{as} \parallel_A T$ can also deadlock because of (w, Σ) and thus $\neg P \ sat^d_{as} \ (T, A)$. □

With this we can prove our main characterization result. The proof works essentially in the same way as previously in the MTS-case.

Theorem 2.20. *For dMTS, as-deadlock-refinement $\leq^d_{as}$ and $\mathcal{F}$-refinement $\sqsubseteq_{\mathcal{F}}$ coincide.*

Proof. '$\subseteq$': Consider dMTSs P and P' with $P \leq^d_{as} P'$ and take $(w, X) \in \mathcal{F}(P)$. We choose a fresh ω not occurring in P or P' and let $A = \Sigma \setminus \{\omega\}$. By Prop. 2.17, we know that there is some as-implementation S of P with $(w, X) \in \mathcal{F}(S)$. Therefore, $S \parallel_{\Sigma \setminus \{\omega\}} T(w, X)$ can deadlock by Lemma 2.19.2 and thus P does not satisfy $(T(w, X), A)$. Now our premise implies that P' does not satisfy $(T(w, X), A)$ either and, therefore, there must be an as-implementation S' of P' such that $S' \parallel_{\Sigma \setminus \{\omega\}} T(w, X)$ can deadlock. By Lemma 2.19.2 and Prop. 2.17, we get $(w, X) \in \mathcal{F}(S') \subseteq \mathcal{F}(P')$.

'$\supseteq$': Let $P \sqsubseteq_{\mathcal{F}} P'$. To prove $P \leq^d_{as} P'$, we consider some test (T, A) with $\neg P \ sat^d_{as} \ (T, A)$. Thus, there must be an as-implementation S of P such that $S \parallel_A T$ can deadlock, i.e. $(w, \Sigma) \in \mathcal{F}(S \parallel_A T)$. By Lemma 2.19.1 this implies $\exists (u, Y) \in \mathcal{F}(S) \subseteq \mathcal{F}(P) \subseteq \mathcal{F}(P'), (v, Z) \in \mathcal{F}(T) : w \in u \parallel_A v$ and $\Sigma \subseteq (Y \cap Z) \cup (A \cap (Y \cup Z))$. By Prop. 2.17, we know that there is some as-implementation S' of P' with $(u, Y) \in \mathcal{F}(S')$. Again by Lemma 2.19.1 we get $(w, \Sigma) \in \mathcal{F}(S' \parallel_A T)$. This means that $S' \parallel_A T$ can deadlock and $\neg P' \ sat^d_{as} \ (T, A)$. □

This characterization does not only give more insight into the testing preorder, it also has an impact on decidability. Since the definition of $\leq_{as}^{d}$ refers to arbitrary tests, it is not clear that $\leq_{as}^{d}$ is decidable for finite dMTS, but $\sqsubseteq_{\mathcal{F}}$ is – by the following standard method [37]:

When comparing two finite dMTS, we can assume that Σ is finite. From each of the dMTSs, say P, we construct a finite automaton by just considering the may-transitions and adding a new, unique final state *fail*. From each *readies*(p), we determine the refusal sets X that p contributes according to Def. 2.15, and we add for each X an X-labelled transition from p to *fail*. The automaton accepts exactly the words wX with $(w, X) \in \mathcal{F}(P)$, and the automaton is finite, since Σ is finite. Hence $\sqsubseteq_{\mathcal{F}}$ can be decided by checking inclusion for regular languages. This method can be made more efficient by considering the fewer readies Z instead of the refusal sets X and matching wZ in the first automaton by wZ' in the other, provided $Z' \subseteq Z$; recall that a smaller Z' also justifies all refusal sets justified by the larger Z.

Compared to a setting with LTS (or even MTS) only, we have the additional costs of inspecting all τ-choices. We will show how to limit these costs, thereby also gaining additional insight into the $\mathcal{F}$-semantics. First, consider the graph on P with edges pp' whenever $p \xrightarrow{\tau} P'$ and $p' \in P'$. Call the weakly connected components of this graph the weak components of P. Each τ-choice of P is a combination of local τ-choices, one for each of these weak components; such a local τ-choice is sufficient to determine *readies*(p) for all states p of the respective weak component. Thus, it is much more efficient to consider all local τ-choices instead of their combinations, whose number can be exponential in the number of weak components. This idea should be combined with the following ideas, which we describe for the undecomposed P.

Recall that a τ-choice f reaches a state p_n from p_1 if $p_1 \overset{\tau}{\dashrightarrow} p_2 \overset{\tau}{\dashrightarrow} \cdots p_n$ such that $p_{i+1} \in f(p_i)$ for $i = 1, \ldots, n-1$. Each such p_n contributes its 'must-enabled' $a \in \Sigma$ to the $Z \in$ *readies*(p_1) arising from f. Thus, having fewer reachable states gives a smaller, i.e. more informative, ready. A simple way to achieve this concerns transitions $p_i \xrightarrow{\tau} P'$ with $p_i \in P'$: We can assume that f chooses p_i to cover this transition; any other choice can only lead to additional states being reached from p_1. We can enforce the desirable choice by simply removing the transition $p_i \xrightarrow{\tau} P'$. This will usually make *readies*(p_1) smaller, but for each missing Z, there is still some $Z' \subseteq Z$. Thus, the modified *readies*(p_1) may differ from the original one, but it still gives rise to the same $\mathcal{F}$-semantics. In short, before determining $\mathcal{F}(P)$, we remove all transitions $p \xrightarrow{\tau} P'$ with $p \in P'$.

The next idea for increasing efficiency is more subtle and more interesting.

If a τ-choice f reaches a must-stable state p_n from some $p_1 \neq p_n$, the resulting $Z \in \mathit{readies}(p_1)$ is irrelevant for $\mathcal{F}(P)$, because $\mathit{readies}(p_n) = \{\mathit{ready}_t(p_n)\}$ and $\mathit{ready}_t(p_n) \subseteq Z$. The latter means that each (w, X) with $p_0 \overset{w}{\Longrightarrow} p_1$ and $X \cap Z = \emptyset$ also arises from $p_0 \overset{w}{\Longrightarrow} p_n$ (!) and $X \cap \mathit{ready}_t(p_n) = \emptyset$.

This observation has a consequence for divergence-free dMTS in general, i.e. also for infinite ones. For each p_1 reachable in such a dMTS and each τ-choice f, f can reach a must-stable state from p_1. Otherwise there would be an infinite run from p_1 along the τ-transitions selected by f. Hence it is sufficient to consider the stable-failure semantics, e.g. defined on LTS in [8]. We generalize it to dMTS as follows:

Definition 2.21. *The* stable-failure semantics *of a dMTS P is:*
$\mathcal{F}_{st}(P) = \{(w, X) \mid p_0 \overset{w}{\Longrightarrow} p, p \text{ is must-stable, and } \forall a \in X : \; p \overset{a}{\not\longrightarrow}\}$

Proposition 2.22. *For a divergence-free dMTS P, $\mathcal{F}(P) = \mathcal{F}_{st}(P)$.*

Now we return to finite dMTSs. To make use of the second idea from above, the plan is to determine $\mathcal{F}(P)$ from the stable-failure semantics and from a restricted set of τ-choices, defined on a separate dMTS $TC(P)$, cf. Thm. 2.23.

$TC(P)$ has state set P, initial state p_0 and, initially, the τ-must-transitions of P; at all stages, it only has the τ-may-transitions required by syntactic consistency. We will reduce $TC(P)$ and, in the end, we will consider the τ-choices of $TC(P)$ to obtain the *reduced readies* $\mathit{rr}(p_1) \subseteq \mathit{readies}(p_1)$ in P for all p_1 still in $TC(P)$ – in the same way as $\mathit{readies}(p_1)$ is obtained in Def. 2.15. These τ-choices satisfy the defining condition of a τ-choice for all these p_1, but they are not defined for the other states of P. Finally, we will add $\{(w, X) \mid \exists p_0 \overset{w}{\Longrightarrow} p : p \in TC(P) \text{ and } \exists Z \in \mathit{rr}(p) : X \cap Z = \emptyset\}$ to $\mathcal{F}_{st}(P)$.

The set $\mathit{readies}(p)$ for a must-stable p is already taken account of in $\mathcal{F}_{st}(P)$. Furthermore, p's contribution to the readies of the other states is also not needed: As argued above, if some τ-choice f reaches p from some p_1, then the resulting $Z \in \mathit{readies}(p_1)$ is irrelevant – i.e. it should not be considered. Consequently, we remove p from $TC(P)$ and replace each $p' \xrightarrow{\tau} P'$ with $p \in P'$ by $p' \xrightarrow{\tau} P' \setminus \{p\}$. This forces a τ-choice to choose a different state from P', i.e. the f just discussed is indeed not considered anymore.

To sum up, the first modification of $TC(P)$ is that we remove all must-stable states. In the intermediate stages of the construction, also empty target states are allowed. In fact, and this is an invariant in the construction, a transition $p' \xrightarrow{\tau} \emptyset$ indicates that any τ-choice f of P must reach a must-stable state from p'. Again, with the argument from above, if f reaches p'

from some p (possibly $= p'$), the resulting $Z \in \mathit{readies}(P)$ is not relevant. Hence, iteratively, we remove each p' with $p' \xrightarrow{\tau} \emptyset$.

Finally, for all p_1 in $TC(P)$, we determine $\mathit{rr}(p_1)$ on P analogously to the definition of $\mathit{readies}(p_1)$, but with f being a τ-choice for $TC(P)$. This gives the following result:

Theorem 2.23. *Let P be a finite dMTS and let $TC(P)$ and $\mathit{rr}(p_1)$ for $p_1 \in TC(P)$ be obtained by the algorithm above, then $\mathcal{F}(P) = \{(w, X) \mid \exists p_1 \in TC(P), p_0 \overset{w}{\Longrightarrow} p_1, Z \in \mathit{rr}(p_1) : X \cap Z = \emptyset\} \cup \mathcal{F}_{st}(P)$.*

As an example consider Fig. 2.6. The two τ-choices reaching p_7 from p_0 give rise to $\{a, b, c, d\}, \{a, b, c, e\} \in \mathit{readies}(p_1)$. They are not needed because of $\mathit{readies}(p_8) = \{\{d\}\}$ and $\mathit{readies}(p_9) = \{\{e\}\}$. The method removes p_8 and p_9, and hence p_7 from $TC(P)$. Now there is just one τ-choice and $\mathit{rr}(p_0) = \{\{a, b, f\}\}$.

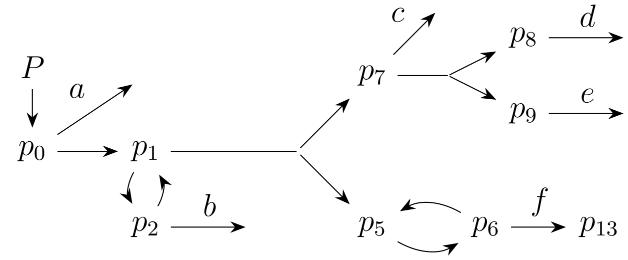

Figure 2.6: Unlabelled transitions are the τ-transitions forming $TC(P)$.

Further optimizations are possible. Consider a τ-choice on $TC(P)$ and the graph G with the states of $TC(P)$ as vertices and edges pp' whenever $p \overset{\tau}{\dashrightarrow} p'$ in $TC(P)$ and $p' \in f(p)$. Assume an SCC (strongly connected component) C of G is reachable from some p_1, but not vice versa. Then, we can invoke the argument from above with C playing the role of the stable state p_n: $Z \in \mathit{rr}(p_1)$ arising from f is irrelevant due to Z' arising from f for each $p \in C$ (all these p have the same Z').

To use this observation, we compute the $\mathit{rr}(p)$ with an outer loop over all τ-choices f for $TC(P)$. For each f, we obtain in an inner loop the contribution Z for each $p \in TC(P)$.

For this inner loop, we compute the SCCs described above in linear time. Call an SCC a *leaf*, if there is no edge leaving it. For each leaf C, we choose one $p \in C$; we determine the Z resulting from f for p and add it to $\mathit{rr}(p)$. The resulting sets $\mathit{rr}(p)$ will in general be smaller than the original ones, but they still make Thm. 2.23 true.

For the only τ-choice in the reduced $TC(P)$ of Fig. 2.6 obtained above, $\{p_5, p_6\}$ is an SCC and a leaf. We finally obtain w.l.o.g.: $rr(p_5) = \{\{f\}\}$ and $rr(p_0) = rr(p_1) = rr(p_2) = rr(p_6) = \emptyset$.

Although it is a bit complex to extract the $\mathcal{F}$-semantics from a dMTS, we have characterized the deadlock-oriented testing preorder with the same kind of $\mathcal{F}$-semantics as in the MTS-case; we make this more precise below. Furthermore, we have the same robustness results. The first concerns thoroughness; the if-part of the following statement follows from Prop. 2.17, the only-if part holds for every refinement relation $\sqsubseteq_r$.

Proposition 2.24. *$\mathcal{F}$-refinement is thorough, i.e. for dMTS P and P', $P \sqsubseteq_{\mathcal{F}} P'$ if and only if $impl_{\mathcal{F}}(P) \subseteq impl_{\mathcal{F}}(P')$.*

The second robustness result concerns the question how our testing preorder depends on the choice to use as-implementations. The next proposition is immediately implied by Lemma 2.19.3.

Proposition 2.25. *The testing preorders $\leq^d_{as}$ and $\leq^d_{\mathcal{F}}$ coincide.*

Again, as for MTS, the same result holds for any $\leq^d_r$ in place of $\leq^d_{\mathcal{F}}$ due to Lemma 2.19.3, provided $\sqsubseteq_r$ lies between $\sqsubseteq_{\mathcal{F}}$ and $\sqsubseteq_{as}$.

2.3.3 Compositionality

In a testing approach, systems are compared as components in the same parallel contexts. Therefore, a testing preorder usually is a precongruence for parallel composition, i.e. it supports modular refinement (cf. [18, Thm. 17] for a general theorem). It is therefore surprising that this does not hold for $\leq^d_{as}$ on dMTSs. This can only happen because, in contrast to usual testing scenarios, our test systems are only drawn from the subclass of LTSs, and dMTSs are only checked on their implementations. It is maybe just a pleasant surprise that $\leq^d_{as}$ *is* a precongruence on MTSs [19].

First, we present the definition of parallel composition for dMTS (cf. e.g. [51]). Then, we will show that this parallel composition is compositional w.r.t. $\mathcal{F}$-refinement on MTS, but not on dMTS.

Definition 2.26. *The* parallel composition *of two dMTSs $P_1 = (P_1, p_{10}, \longrightarrow_1, \dashrightarrow_1)$ and $P_2 = (P_2, p_{20}, \longrightarrow_2, \dashrightarrow_2)$ with synchronizing set $A \subseteq \Sigma$ is defined as the dMTS $P_1 \parallel_A P_2 = (P_1 \times P_2, (p_{10}, p_{20}), \longrightarrow_{12}, \dashrightarrow_{12})$, where*

$$\begin{aligned} \longrightarrow_{12} = {} & \{\big((p_1, p_2), \alpha, P'_1 \times \{p_2\}\big) \mid p_1 \xrightarrow{\alpha}_1 P'_1, \alpha \in \Sigma_\tau \setminus A\} \\ & \cup \{\big((p_1, p_2), \alpha, \{p_1\} \times P'_2\big) \mid p_2 \xrightarrow{\alpha}_2 P'_2, \alpha \in \Sigma_\tau \setminus A\} \\ & \cup \{\big((p_1, p_2), a, P'_1 \times P'_2\big) \mid p_1 \xrightarrow{a}_1 P'_1, p_2 \xrightarrow{a}_2 P'_2, a \in A\} \end{aligned}$$

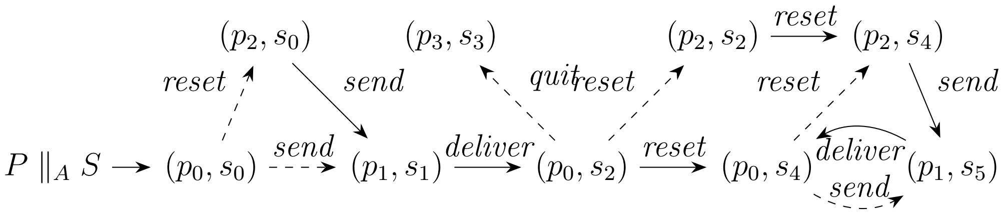

Figure 2.7: $P \parallel_A S$ with $A = \Sigma \setminus \{\mathit{reset}\}$.

$$\begin{aligned}
\dashrightarrow_{12} = {} & \{((p_1,p_2),\alpha,(p_1',p_2)) \mid p_1 \overset{\alpha}{\dashrightarrow}_1 p_1', \alpha \in \Sigma_\tau \setminus A\} \\
\cup {} & \{((p_1,p_2),\alpha,(p_1,p_2')) \mid p_2 \overset{\alpha}{\dashrightarrow}_2 p_2', \alpha \in \Sigma_\tau \setminus A\} \\
\cup {} & \{((p_1,p_2),a,(p_1',p_2')) \mid p_1 \overset{a}{\dashrightarrow}_1 p_1', p_2 \overset{a}{\dashrightarrow}_2 p_2', a \in A\}
\end{aligned}$$

Note that the parallel composition of two MTS again yield an MTS.

It is commonly known that the following lemma holds for LTS. Since it only concerns may-transitions, it also holds for dMTS, and we will use it without explicitly referencing it.

Lemma 2.27. *For dMTSs P_1, P_2 we have in $P_1 \parallel_A P_2$: $(p_1, p_2) \overset{w}{=\Rightarrow} (p_1', p_2')$ if and only if $\exists u, v \in \Sigma^* : p_1 \overset{u}{=\Rightarrow} p_1', p_2 \overset{v}{=\Rightarrow} p_2'$ and $w \in u \parallel_A v$.*

To illustrate how modalities are treated in $\parallel_A$, consider the MTS $P \parallel_{\Sigma\setminus\{\mathit{reset}\}} S$ shown in Fig. 2.7 with P and S from Fig. 2.1. Note that the *send*-may-transition of p_0 and the *send*-must-transition of s_0 result in a may-transition only. Another interesting point is that (p_2, s_2) has no outgoing *send*-transition, although p_2 does have a *send*-must-transition.

To prove the result for MTS, we will use the following standard lemma for parallel composition; see e.g. 3.1.7 in [70], which has a similar proof.

Lemma 2.28. *Consider MTS P and Q with $p \in P$ and $q \in Q$; further let $w \in \Sigma^*$.*

1. *$\exists u, v \in \Sigma^* : w \in u \parallel_A v \wedge p \overset{u}{=\Rightarrow} p' \wedge q \overset{v}{=\Rightarrow} q'$ iff $(p, q) \overset{w}{=\Rightarrow} (p', q')$ in $P \parallel_A Q$*

2. *$\exists u, v \in \Sigma^* : w \in u \parallel_A v \wedge p \overset{u}{\Longrightarrow} p' \wedge q \overset{v}{\Longrightarrow} q'$ iff $(p, q) \overset{w}{\Longrightarrow} (p', q')$ in $P \parallel_A Q$*

The next proposition is a rather simple modification of Lem. 1.7. It implies that $\sqsubseteq_{\mathcal{F}}$ is a precongruence for $\parallel_A$. We give a proof to point out the details regarding modalities.

Proposition 2.29. *For all MTS P and Q and all $A \subseteq \Sigma$: $\mathcal{F}(P \parallel_A Q) = \{(w,X) \mid (u,Y) \in \mathcal{F}(P), (v,Z) \in \mathcal{F}(Q), w \in u \parallel_A v, X \subseteq (Y \cap Z) \cup (A \cap (Y \cup Z))\}$.*

Proof. '$\subseteq$ ': Let $(w,X) \in \mathcal{F}(P \parallel_A Q)$; then there is p with $(p_0,q_0) \overset{w}{\dashrightarrow} (p,q)$ and $\forall a \in X : (p,q) \overset{a}{\not\Longrightarrow}$. By Lemma 2.28.1 there must be $u, v \in \Sigma^*$ such that $p_0 \overset{u}{\Longrightarrow} p$, $q_0 \overset{v}{\Longrightarrow} q$ and $w \in u \parallel_A v$. Let $Y = \{a \in \Sigma \mid p \overset{a}{\not\Longrightarrow}\}$ and $Z = \{a \in \Sigma \mid q \overset{a}{\not\Longrightarrow}\}$. Together with the definition of $\parallel_A$, Lemma 2.28.2 shows that for $a \in X \cap A$ we must have $a \in Y$ or $a \in Z$, and for $a \in X \setminus A$ we must have $a \in Y$ and $a \in Z$.

'$\supseteq$ ': Let (w,X) be an element of the r.h.s. and let $p_0 \overset{u}{\Longrightarrow} p$, $q_0 \overset{v}{\Longrightarrow} q$ be such, that $w \in u \parallel_A v$, $p \overset{a}{\not\Longrightarrow}$ for all $a \in Y$ and $q \overset{a}{\not\Longrightarrow}$ for all $a \in Z$. By Lemma 2.28.1 we have $(p_0,q_0) \overset{w}{\Longrightarrow} (p,q)$. Consider $a \in X$. If $a \in A$ then $a \in Y \cup Z$ (since $Y \cap Z \subseteq Y \cup Z$), which implies $p \overset{a}{\not\Longrightarrow}$ or $q \overset{a}{\not\Longrightarrow}$. By Lemma 2.28.2 we get $(p,q) \overset{a}{\not\Longrightarrow}$. If $a \notin A$ then $a \in Y \cap Z$, which implies $p \overset{a}{\not\Longrightarrow}$ and $q \overset{a}{\not\Longrightarrow}$. By Lemma 2.28.2 we get again $(p,q) \overset{a}{\not\Longrightarrow}$ and thus (w,X) is in the l.h.s. □

Corollary 2.30. *On MTS $\sqsubseteq_{\mathcal{F}}$ is a precongruence for $\parallel_A$, i.e. $P' \sqsubseteq_{\mathcal{F}} P$ implies $P' \parallel_A Q \sqsubseteq_{\mathcal{F}} P \parallel_A Q$.*

Having achieved the desired compositionality result for MTS, we proceed to the counterexample for dMTSs: Figure 2.8 shows two $\mathcal{F}$-equivalent dMTs P and P', essentially because both can initially refuse either a, b or c, but not more than one of them; in particular, when a τ-choice for P picks τ-transitions from p_1 to p_2 and from p_2 to p_1, then p_3 becomes unreachable and c can be refused. However $P \parallel_A P$ is not equivalent to $P \parallel_A P'$ for $A = \{a,b,c\}$. The former has (ε, Σ) in its $\mathcal{F}$-semantics (the justifying τ-choice is also illustrated in Fig. 2.8), while the latter has not. This example indicates that the precongruence problem is related to divergence. And indeed, $\sqsubseteq_{\mathcal{F}}$ is a precongruence for divergence-free dMTS; this result has been shown in [62], and we will obtain it here as a corollary in Sect. 2.4, where we test for divergence as well as deadlock.

Another standard way to proceed would be to look for the coarsest precongruence contained in $\sqsubseteq_{\mathcal{F}}$, cf. e.g. the problem with bisimilarity and choice in CCS [57]. It is not at all clear how this coarsest precongruence for $\sqsubseteq_{\mathcal{F}}$ could be characterized. Besides the interest that understanding the deadlock-testing preorder has in itself, we can regard $\mathcal{F}$-refinement as an overapproximation of the coarsest precongruence, accompanied by as-refinement as an underapproximation. That as-refinement is unnecessarily fine is already clear on LTS, where as-refinement is bisimulation equivalence.

In the light of this negative result, it seems little attractive to generalize our approach to full DMTSs. In this context, it should be noted that it took 20 years to define a parallel composition for DMTS [7]. This definition is quite indirect and hard to handle, and it seems that hardly any results concerning this parallel composition are known.

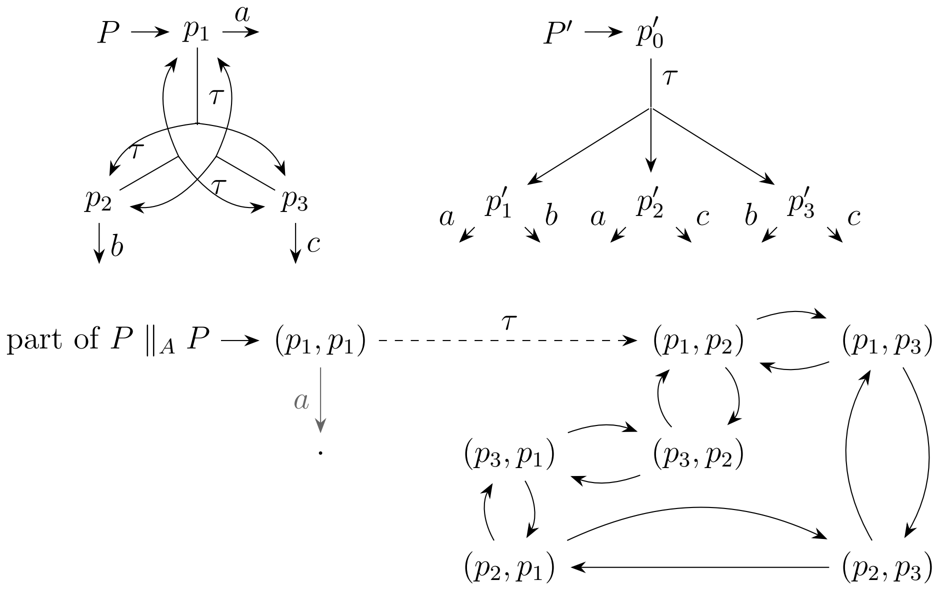

Figure 2.8: Equivalent systems P, P'; the problematic $(\varepsilon, \Sigma) \in \mathcal{F}(P \parallel_A P)$ is due to (p_1, p_2) and the τ-choice which is shown for the reachable part (must-τ-labels omitted).

The considerable effort needed to develop our approach for dMTS resulted in a very standard failure semantics. This is demonstrated by the fact that LTS are already as expressive as dMTS in our approach.

To show this, we use the standard concept of $\mathcal{F}$-consistent sets. The conditions, FC1 – FC3, are known from the classical failure semantics [11]. We will prove that the $\mathcal{F}$-semantics of a dMTS is $\mathcal{F}$-consistent and that each $\mathcal{F}$-consistent set can be represented as an LTS.

Definition 2.31. *An $\mathcal{F}$-set $M \subseteq \Sigma^* \times \mathfrak{P}(\Sigma)$ is called* $\mathcal{F}$-consistent*, if it satisfies*

FC1) $(w, X) \in M \implies \forall Y \subseteq X : (w, Y) \in M$

FC2) $(w, X) \in M \implies \forall v \sqsubseteq w : (v, \emptyset) \in M$

FC3) $(w, X) \in M \land \forall a \in Y : (wa, \emptyset) \notin M \implies (w, X \cup Y) \in M$

Note that FC3 corresponds to syntactic consistency. If $(w, X \cup Y) \notin M$ then, roughly, some $a \in Y$ cannot be refused after w, it must be possible in the sense of $\overset{a}{\Longrightarrow\!\!\!\!\!/}$. Hence, a is allowed after w, wa is a possible trace and $(wa, \emptyset) \in M$.

Proposition 2.32. *1. For a dMTS P, $\mathcal{F}(P)$ is $\mathcal{F}$-consistent.*

2. For each $\mathcal{F}$-consistent set[4] *$M \neq \emptyset$, there is an LTS P with $\mathcal{F}(P) = M$.*

Proof. 1.: For FC1 and FC2 consider some $(w, X) \in \mathcal{F}(P)$. By definition, there is some p and with $p_0 \overset{w}{\Longrightarrow} p$ and some $Z \in \mathit{readies}(p)$ with $Z \cap X = \emptyset$. Obviously, we have $Z \cap Y = \emptyset$ for all subsets $Y \subseteq X$; furthermore, for all prefixes $v \sqsubseteq w$, we have some p' with $p_0 \overset{v}{\Longrightarrow} p'$ and $\forall Z' \in \mathit{readies}(p') : Z' \cap \emptyset = \emptyset$.
For FC3 consider $(w, X) \in \mathcal{F}(P)$ due to some $p_0 \overset{w}{\Longrightarrow} p$, τ-choice f and resulting set $Z \in \mathit{readies}(p)$, and consider some Y with $\forall a \in Y : (wa, \emptyset) \notin \mathcal{F}(P)$. This means for any $a \in Y$ that $p \overset{a}{\not\Longrightarrow}$, hence $a \notin Z$. Therefore $Z \cap (X \cup Y) = \emptyset$, and we are done.

2.: We define an LTS P as follows: $P = \{w \mid (w, \emptyset) \in M\} \cup M$, $p_0 = \varepsilon$ (where $\varepsilon \in P$ by FC2 since $M \neq \emptyset$), $w \xrightarrow{a} wa$ if $(wa, \emptyset) \in M$, $w \xrightarrow{\tau} (w, X)$ if $(w, X) \in M$, and $(w, X) \xrightarrow{a} wa$ if $a \notin X$ and $(wa, \emptyset) \in M$.

To show that $\mathcal{F}(P) = M$ we use the traditional-style definition from Def. 2.4, which for divergence-free MTS/LTS (like P) is the same as $\mathcal{F}_{st}$, as we have proven in Prop. 2.22.

Each $(w, X) \in M$ is represented by the state (w, X), which is reached by $p_0 \xrightarrow{w} w \xrightarrow{\tau} (w, X)$.

For the converse, observe that a state (w, Y) may also refuse actions $a \notin Y$. Consider some $(w, X) \in \mathcal{F}_{st}(P)$. Since the w-states of P are not must-stable, (w, X) is justified by some state (w, Y), where possibly $X \neq Y$. For each $a \in X$, we have $(w, Y) \overset{a}{\not\longrightarrow}$ and, by construction, $a \in Y$ or $(wa, \emptyset) \notin M$. From this and $(w, Y) \in M$, we get that $(w, Y \cup X) \in M$ by FC3, and $(w, X) \in M$ by FC1. □

This translation of each dMTS P to an $\mathcal{F}$-equivalent LTS $\overline{P}$ allows to inherit positive results from an LTS- or MTS-setting, e.g. for *conjunction*: In the next section, we will show that a conjunction operator exists for MTS with $\mathcal{F}$-refinement, i.e. for any two MTSs P and P', a third one $P \land P'$ exists such that $Q \sqsubseteq_{\mathcal{F}} P \land P'$ if and only if $Q \sqsubseteq_{\mathcal{F}} P$ and $Q \sqsubseteq_{\mathcal{F}} P'$ for each MTS

[4] To satisfy the 'fresh-action assumption' for dMTS, one should require here that there always be a fresh action not occurring on any trace in the $\mathcal{F}$-set.

Q – the characteristic property when applying conjunction to specifications. Combining the results, the conjunction of dMTSs P and P' can be defined as $\overline{P} \wedge \overline{P'}$.

This idea does not work for parallel composition, since this operator already exists on dMTS. For P and P' shown in Fig. 2.8, $\overline{P} \parallel_A \overline{P}$ and $\overline{P} \parallel_A \overline{P'}$ are $\mathcal{F}$-equivalent, since $\sqsubseteq_{\mathcal{F}}$ is a precongruence for parallel composition on MTS. Thus, parallel composition on dMTS differs from the one on MTS.

Clearly, the above translation is only conceptual; in particular, it might translate a finite dMTS into an infinite LTS. It could be that some dMTSs are more compact than $\mathcal{F}$-equivalent LTSs.

2.3.4 Conjunction

As already explained in the introduction, conjunction is an important operator if one wants to describe various aspects of a specification with separate systems and, then, to combine these into one overall specification, which allows exactly for all common refinements of the former ones. In the following, we will define a conjunction operator on MTS w.r.t. $\mathcal{F}$-refinement. Because of the arguments at the end of the previous section, we will not consider general dMTS for this. It should be noted that in our setting the conjunction of two MTS can be represented as an MTS. It is not necessary to resort to a larger and more complicated class of MTS like DMTS or dMTS, as is the case for MTS with as-refinement [51, 53]. To prove the operator correct, we will show that the $\mathcal{F}$-refinements of $P_1 \wedge P_2$ are exactly the common $\mathcal{F}$-refinements of P_1 and P_2. This also means that, naturally, $P_1 \wedge P_2$ can only be defined if such a common $\mathcal{F}$-refinement exists.

On the level of the $\mathcal{F}$-semantics, conjunction corresponds more or less to intersection. But actually, such an intersection is often not representable by an MTS: the sets $\mathcal{F}(P)$ with P an MTS are $\mathcal{F}$-consistent as shown in the previous section, whereas their intersections often are not.

We start by showing how to obtain the greatest $\mathcal{F}$-consistent set contained in an $\mathcal{F}$-set. As we will explain after Def. 2.35, this gives a first conjunction operator, which is easier to understand and prove correct than the second one further below.

Definition 2.33. *For an $\mathcal{F}$-set M, $GFC(M)$ is defined stepwise as follows:*

1. $M_1 := M \setminus \{(w, X) \mid \exists Y \subseteq X : (w, Y) \notin M\}$

2. $M_2 := M_1 \setminus \{(w, X) \mid \exists v \sqsubseteq w : (v, \emptyset) \notin M_1\}$

3. *$GFC(M) := M_2 \setminus (FIS(M_2) \cup \{(w,X) \mid \exists v \sqsubset w : (v,\emptyset) \in FIS(M_2)\})$, where the $\mathcal{F}$*-inconsistency set *$FIS(M_2)$ is defined as the least set satisfying: If*
   $(*)$ $(w,X) \in M_2 \wedge (\exists Y \forall a \in Y : ((wa,\emptyset) \notin M_2 \vee (wa,\emptyset) \in FIS(M_2)) \wedge (w, X \cup Y) \notin M_2)$,
   then $(w,X) \in FIS(M_2)$.

In steps 1 – 3, we remove elements of M that lead to violation of FC1 – FC3, respectively. This is obvious for the first two steps. Note that one must ensure that later removals do not introduce violations of conditions enforced earlier. The iteratively constructed $FIS(M_2)$ collects those (w,X) that lead to a violation of FC3.

As an example, consider a case where M_2 contains $(ab,\emptyset)$ and $(aba,\emptyset)$, but none of $(ab,\{a\})$, $(aba,\{c\})$, or $(abac,\emptyset)$. Regarding the definition of $FIS(M_2)$ as a fixed point iteration, we initialize it as $\emptyset$ and then see immediately that $(aba,\emptyset)$ satisfies $(*)$ for $Y = \{c\}$. After adding this pair to $FIS(M_2)$, $(ab,\emptyset)$ also satisfies $(*)$ for $Y = \{a\}$.

In the end, $FIS(M_2)$ is removed from M_2; since this can introduce new violations of FC2, the respective refusal pairs are removed as well.

Because of (3), it is not so obvious that $GFC(M)$ is well defined. Hence, we prove now that there exists a least set satisfying the if-statement. For an $\mathcal{F}$-set $F \subseteq M_2$ and $(w,X) \in M_2$, we say that F *derives* (w,X), whenever $(*)$ is true provided $F \subseteq FIS(M_2)$. A *FIS-derivation tree* for (w,X) is a (possibly infinite) directed tree without infinite paths where the vertices are elements of M_2, (w,X) is the root and for each vertex (v,Z):
– (v,Z) in place of (w,X) makes $(*)$ true under the assumption that $FIS(M_2) = \emptyset$
or
– some minimal F derives (v,Z) and F is the set of children of (v,Z).

We argue that $FIS(M_2)$ is the set FIS of all (w,X) with a FIS-derivation tree. On the one hand, by Noetherian induction, all these (w,X) must be in $FIS(M_2)$. On the other hand, consider some (w,X) that makes $(*)$ true with FIS in place of $FIS(M_2)$. Let F be the set of all $(wa,\emptyset) \in FIS$ that are needed to make the disjunction in $(*)$ true. Then, we get a FIS-derivation tree for (w,X) by making each $(wa,\emptyset) \in F$ a child of the new root (w,X) and adding a FIS-derivation tree for each of these $(wa,\emptyset)$. Thus, $(w,X) \in FIS$ and the defining if-statement for $FIS(M_2)$ is satisfied.

Since $FIS(M_2)$ is the least set with this property, $FIS(M_2) = FIS$.

Proposition 2.34. *The greatest $\mathcal{F}$-consistent $\mathcal{F}$-set M' contained in some $\mathcal{F}$-set M satisfies $M' = GFC(M)$.*

Proof. Clearly, refusal pairs removed in Step 1 of Def. 2.33 cannot be in M' and M_1 satisfies FC1. Also, the refusal pairs removed in Step 2 cannot be in M'. Since, for some w, all (w, X) are removed if some (w, X) is, M_2 still satisfies FC1. If $(w, X) \in M_1$, $w' \sqsubseteq w$ and $(w', \emptyset)$ is removed due to some $v \sqsubseteq w'$, then (w, X) is also removed due to v, hence M_2 also satisfies FC2.

For Step 3, observe that $(*)$ is monotonic in $\mathit{FIS}(M_2)$: If (w, X) satisfies $(*)$ due to Y for some set in place of $\mathit{FIS}(M_2)$, then it also does for a larger set. Hence, $\mathit{FIS}(M_2)$ is a least fixed point and defined.

Further, by transfinite induction, no (w, X) added to $\mathit{FIS}(M_2)$ can be in M' due to FC3. Also note: (1) If $(w, X) \in M_2$ and for some $X' \subseteq X$, (w, X') is added to $\mathit{FIS}(M_2)$ due to Y, then (w, X) is also added because of Y. The reason is that $(w, X \cup Y) \in M_2$ would imply $(w, X' \cup Y) \in M_2$ since M_2 satisfies FC1.

By construction, $M_2 \setminus \mathit{FIS}(M_2)$ satisfies FC3. Removing (w, X) due to some $v \sqsubset w$ satisfying $(v, \emptyset) \in \mathit{FIS}(M_2)$ removes all refusal pairs (w, Z); hence it can only influence FC3 if $(w, X) = (ua, \emptyset)$. But in this case $v \sqsubseteq u$; if $u = v$, all (u, X) are removed from M_2 by (1); for $v \sqsubset u$ this is clear anyway. In conclusion, $\mathit{GFC}(M)$ satisfies FC3.

Now (1) shows that FC1 is preserved in $\mathit{GFC}(M)$, since all (w, Y) are removed if some (w, X) is removed due to $v \sqsubset w$. FC2 is preserved by construction and FC1, and we are done. □

For the discussion of conjunction, logical consistency is an important notion.

Definition 2.35. *MTSs P_1 and P_2 are* logically consistent *if they have a common refinement.*

The conjunction of MTSs P_1 and P_2 has to be refined exactly by the common refinements of P_1 and P_2. If P_1 and P_2 are logically consistent, the $\mathcal{F}$-semantics of any common refinement is an $\mathcal{F}$-consistent set contained in $\mathcal{F}(P_1)$ and $\mathcal{F}(P_2)$; thus, this set is contained in $\mathit{GFC}(\mathcal{F}(P_1) \cap \mathcal{F}(P_2))$, and the latter is non-empty for logically consistent P_1 and P_2. For finite alphabets, we now have an MTS with $\mathcal{F}$-semantics equal to $\mathit{GFC}(\mathcal{F}(P_1) \cap \mathcal{F}(P_2))$ by Prop. 2.32, and we could take this as $P_1 \wedge P_2$. Therefore, the conjunction exists for logically consistent P_1 and P_2 in this case. But the construction to prove Prop. 2.32 usually builds an infinite LTS and is, therefore, not a real algorithm. We will improve on this now; see the last statement of Thm. 2.37.

Definition 2.36. *For MTSs P_1 and P_2, let $P_1 \|_{may} P_2$ denote the MTS $(P_1, p_{01}, \emptyset, \dashrightarrow_1) \|_\Sigma (P_2, p_{02}, \emptyset, \dashrightarrow_2)$. The (possibly infinite)* powerset-MTS *R of $P_1 \|_{may} P_2$ is defined by:*

- *The initial state is* $r_0 = \{(p_1,p_2) \mid (p_{01},p_{02}) \overset{\varepsilon}{\Longrightarrow} (p_1,p_2) \textit{ in } P_1 \|_{may} P_2\} =: Q_0$.

- *For* $a \in \Sigma$ *and* $Q \subseteq P_1 \times P_2$*:* $Q \overset{a}{\dashrightarrow} Q'$ *if* $Q' = \{(p_1',p_2') \mid \exists (p_1,p_2) \in Q : (p_1,p_2) \overset{a}{\Longrightarrow} (p_1',p_2') \textit{ in } P_1 \|_{may} P_2\} \neq \emptyset$

- $\longrightarrow_R = \emptyset$

- *The states of* R *are all* Q *reachable from* Q_0*, and* $\dashrightarrow_R$ *is restricted accordingly.*

For a state (p_1,p_2) *in* $Q \in R$*, define* $\textit{refuse}(p_1,p_2) = \{a \in \Sigma \mid p_1 \overset{a}{\not\Longrightarrow} \wedge\, p_2 \overset{a}{\not\Longrightarrow}\}$*. Now we let the* inconsistency set $IS(R)$ *be the least subset of* R *such that* $Q \in IS(R)$ *if each* $(p_1,p_2) \in Q$ *satisfies:*
$(**)\ \exists a \in \Sigma : a \notin \textit{refuse}(p_1,p_2) \wedge (\forall Q' : Q \overset{a}{\dashrightarrow} Q' \textit{ implies } Q' \in IS(R))$

If $Q_0 \in IS(R)$*,* $P_1 \wedge P_2$ *is undefined (and there is no common implementation of* P_1 *and* P_2*, see below). Otherwise, the* conjunction $P_1 \wedge P_2$ *has*

- $\{Q \mid Q \textit{ is reachable in } R \setminus IS(R)\} \cup \{(p_1,p_2,Q) \mid Q \textit{ is reachable in } R \setminus IS(R), (p_1,p_2) \in Q \textit{ does not satisfy } (**)\}$ *as state set and* Q_0 *as initial state;*

- *the may-transitions of* R *and the following must-transitions (with the underlying may-transitions):*

  - $Q \xrightarrow{\tau} (p_1,p_2,Q)$
  - $\forall a \notin \textit{refuse}(p_1,p_2) : (p_1,p_2,Q) \xrightarrow{a} Q'$ *for the* Q' *satisfying* $Q \overset{a}{\dashrightarrow} Q'$*.*
    Observe that the latter transition exists since (p_1,p_2) *does not satisfy* $(**)$*.*

For finite P_1 and P_2, Def. 2.36 presents an algorithm to determine $P_1 \wedge P_2$ (if it exists). The next theorem states that $P_1 \wedge P_2$ is indeed the conjunction of P_1 and P_2.

The core of the algorithm is the construction of the iteratively increasing set $IS(R)$ (a fixed-point iteration), which corresponds to an iterative reduction of the state set of $P_1 \wedge P_2$. Constructions like this are known from the literature: one identifies a logic inconsistency and removes a corresponding state; this can create a new inconsistency, and so the removal has to be repeated. This is called backward propagation of inconsistency in [49] and pruning in [59].

The approach of [59] works with deterministic MTS. A parallel product like $P_1 \|_{may} P_2$ above is constructed as an MTS that might violate syntactic consistency. An inconsistency is a state (i.e. a state pair from $P_1 \times P_2$) that is required, but not allowed to perform some a. Removal of this state deletes further may-transitions, possibly leading to new inconsistencies and so on. Not surprisingly, there are some similarities with our construction, but also important differences that make our construction much more subtle.

Condition $a \notin \mathit{refuse}(p_1, p_2)$ says that (p_1, p_2), considered on its own, is required to perform a while the next conjunct says that Q is not allowed to do a, implying that a is required but not allowed for (p_1, p_2). Now, if indeed some a is required but not allowed for (p_1, p_2), this is not a problem (a relevant inconsistency) in itself, provided some other state pair in Q and hence also Q may do an a. And even if the latter does not hold, the problem only has an effect if all state pairs in Q satisfy $(**)$. Another subtle point is that they may satisfy $(**)$ due to different actions a. Also see the examples below.

Note that addition of some Q' to $IS(R)$ essentially removes Q' and its incoming may-transitions from R: when checking the second conjunct in $(**)$, may-transitions leading to Q' are considered as non-existent. To see that $IS(R)$ is well defined, one can use a similar argumentation as for $\mathit{FIS}(M_2)$ in Def. 2.33. Instead of refusal pairs in M_2, one has to consider states $Q \in R$. For $\mathit{FIS}(M_2)$ we use FIS-derivation trees, which are directed trees where the vertices are the elements of M_2. Here, one has to use IS-derivation DAGs, which are rooted directed acyclic graphs where the vertices are elements of R.

Regarding $P_1 \wedge P_2$, each state Q represents some trace of the conjunction; each $(p_1, p_2) \in Q$ gives rise to a refusal set for these traces, and the states (p_1, p_2, Q) represent the respective refusal pairs.

Theorem 2.37. *$P_1 \wedge P_2$ is defined iff P_1 and P_2 are logically consistent. In this case, $\mathcal{F}(P_1 \wedge P_2)$ is the greatest $\mathcal{F}$-consistent subset of $\mathcal{F}(P_1) \cap \mathcal{F}(P_2)$, and an MTS S is a common $\mathcal{F}$-refinement of P_1 and P_2 iff $S \sqsubseteq_{\mathcal{F}} P_1 \wedge P_2$.*

Furthermore, $\sqsubseteq_{\mathcal{F}}$ is a precongruence w.r.t. $\wedge$ in the following sense: If P_1 and P_2 are logically consistent and $P_1 \sqsubseteq_{\mathcal{F}} P_1'$, then P_1' and P_2 are logically consistent and $P_1 \wedge P_2 \sqsubseteq_{\mathcal{F}} P_1' \wedge P_2$.

If P_1 and P_2 are finite, then $P_1 \wedge P_2$ is also finite.

Proof. The last statement is obvious. Now, we observe that $\mathcal{F}(P_1) \cap \mathcal{F}(P_2) = \{(w, X) \mid (p_{01}, p_{02}) \overset{w}{\Longrightarrow} (p_1, p_2)$ in $P_1 \|_{may} P_2$ and p_i can refuse X in P_i for $i = 1, 2\}$ and that this set satisfies FC1 and FC2. We can also write this set as $\{(w, X) \mid Q_0 \overset{w}{\dashrightarrow} Q$ in $R \wedge \exists (p_1, p_2) \in Q : p_i$ can refuse X for $i = 1, 2\}$.

Thus, the general idea is to represent just all (w, X) with $Q_0 \overset{w}{\dashrightarrow} Q$ and $\exists(p_1, p_2) \in Q : X = \mathit{refuse}(p_1, p_2)$ in an MTS (and then all (w, Y) with $Y \subseteq X$ are represented as well by FC1). But as we have seen in Def. 2.33 Step 3, it might be necessary to remove some refusal pairs to achieve $\mathcal{F}$-consistency and represent $G := \mathit{GFC}(\mathcal{F}(P_1) \cap \mathcal{F}(P_2))$.

The construction of $IS(R)$ removes those w such that all (w, X) (or equivalently: $(w, \emptyset)$) must be removed. For this, we first note that $(**)$ in Def. 2.36 is monotonic in $IS(R)$, thus $IS(R)$ is a least fixed point and defined.

Next, we prove by transfinite induction that, for all $Q \in IS(R)$, $Q_0 \overset{w}{\dashrightarrow} Q$ implies $(w, \emptyset) \notin G$. For this, consider the set Y of all actions a that make $(**)$ true for some $(p_1, p_2) \in Q$ at the stage when Q is added to $IS(R)$. We have $(w, Y) \notin \mathcal{F}(P_1) \cap \mathcal{F}(P_2)$, because otherwise, for $i = 1, 2$, there exists some p_i justifying $(w, Y) \in \mathcal{F}(P_i)$; for this $(p_1, p_2) \in Q$, we have $\forall a \in Y : a \in \mathit{refuse}(p_1, p_2)$, contradicting that Q is added to $IS(R)$. Furthermore, $(wa, \emptyset) \notin G$ for each $a \in Y$, either because $\neg Q_0 \overset{wa}{\dashrightarrow}$ or by induction applied to the only Q' with $Q_0 \overset{wa}{\dashrightarrow} Q'$. Now, by FC3, $(w, \emptyset) \in G$ would imply $(w, Y) \in G \subseteq \mathcal{F}(P_1) \cap \mathcal{F}(P_2)$, a contradiction.

Observe that we have seen now that $Q_0 \in IS(R)$ implies that G is empty. If P_1 and P_2 had a common $\mathcal{F}$-refinement S, the $\mathcal{F}$-consistent set $\mathcal{F}(S)$ would be contained in G, i.e. $P_1 \wedge P_2$ undefined implies that P_1 and P_2 are not consistent, showing one implication of the first claim in the theorem. So assume now that $Q_0 \notin IS(R)$.

Because of the statement just shown by transfinite induction, it would suffice to represent just all (w, X) with $Q_0 \overset{w}{\dashrightarrow} Q$ in $Q \backslash IS(R)$ and $\exists(p_1, p_2) \in Q : X = \mathit{refuse}(p_1, p_2)$. But this might still be too large; if some (p_1, p_2) satisfies $(**)$ due to a, then (p_1, p_2) cannot justify $(w, X \cup \{a\})$ and (w, X) might violate FC3. We will now argue that we do not have to consider such a pair (p_1, p_2) when representing all $(w, X') \in G$. More technically, we will show that each $(w, X') \in G$ with $X' \subseteq \mathit{refuse}(p_1, p_2)$ is represented by some (p_1', p_2') not satisfying $(**)$, i.e. $X' \subseteq \mathit{refuse}(p_1', p_2')$.

Consider some $Q_0 \overset{w}{\dashrightarrow} Q$ in $R \setminus IS(R)$ and $Z = \{a \in \Sigma \mid \neg Q \overset{a}{\dashrightarrow}$ in $R \setminus IS(R)\}$, i.e. $(wa, \emptyset) \notin G$ for all $a \in Z$. Note that Q was not added to $IS(R)$, so $Z \subseteq \mathit{refuse}(p_1', p_2')$ for some $(p_1', p_2') \in Q$ – namely exactly for those (p_1', p_2') that do not satisfy $(**)$.

Further, consider some $(p_1, p_2) \in Q$ satisfying $(**)$. Then $(w, X') \in G$ for some $X' \subseteq \mathit{refuse}(p_1, p_2)$ implies $(w, X' \cup Z) \in G$ by FC3, which must be justified by some $(p_1', p_2') \in Q$ with $X' \cup Z \subseteq \mathit{refuse}(p_1', p_2')$. Such a (p_1', p_2') does not satisfy $(**)$ and it is therefore sufficient that in $P_1 \wedge P_2$ exactly the respective sets $\mathit{refuse}(p_1', p_2')$ are represented explicitly by (p_1', p_2', Q).

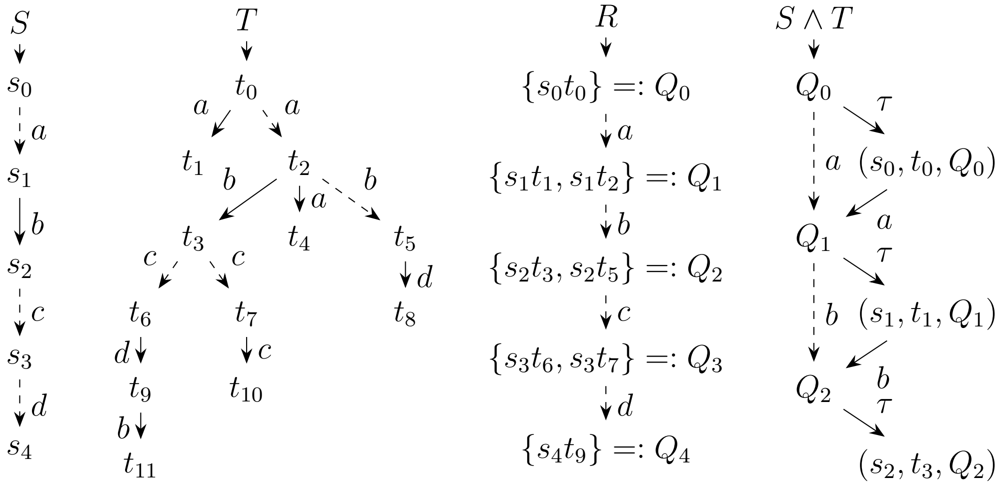

Figure 2.9: An example for conjunction

In conclusion, we have shown that $\mathcal{F}(P_1 \wedge P_2)$ can be obtained by omitting refusal pairs from $\mathcal{F}(P_1) \cap \mathcal{F}(P_2)$; hence, if $P_1 \wedge P_2$ is defined, it demonstrates the consistency of P_1 and P_2, finishing the proof of the first claim. At the same time, the omitted refusal pairs are not in G. Hence, $G \subseteq \mathcal{F}(P_1 \wedge P_2) \subseteq \mathcal{F}(P_1) \cap \mathcal{F}(P_2)$, and $G = \mathcal{F}(P_1 \wedge P_2)$ follows since $\mathcal{F}(P_1 \wedge P_2)$ is $\mathcal{F}$-consistent and G is the greatest $\mathcal{F}$-consistent set in $\mathcal{F}(P_1) \cap \mathcal{F}(P_2)$. The second claim is now obvious, and also the third claim follows since $\mathcal{F}(S)$ is $\mathcal{F}$-consistent.

For the precongruence claim, see the general proof in [51] Cor. 3.8. □

We give a small example in Fig. 2.9 to illustrate some aspects of our conjunction; we write S and T instead of P_1 and P_2 for ease of readability: First, we show how inconsistent states are pruned and how the inconsistency can propagate backwards. For this, recall the condition for a state pair to be considered inconsistent: $(**)$ $\exists a \in \Sigma : a \notin \mathit{refuse}(s_1, s_2) \wedge (\forall Q' : Q \overset{a}{\dashrightarrow} Q'$ implies $Q' \in \mathit{IS}(R))$

Consider R's state Q_4. For its only state pair (s_4, t_9), we have $b \notin \mathit{refuse}(s_4, t_9)$ since $t_9 \overset{b}{\Longrightarrow}$, and $Q_4 \overset{b}{\not\dashrightarrow}$ due to $s_4 \overset{b}{\not\Longrightarrow}$. Thus, (s_4, t_9) is inconsistent and Q_4 is added to $\mathit{IS}(R)$ and will be removed.

Similarly in Q_3, the state pair (s_3, t_7) satisfies $(**)$ because of $c \notin \mathit{refuse}(s_3, t_7)$ due to $t_7 \overset{c}{\Longrightarrow}$ and $Q_3 \overset{c}{\not\dashrightarrow}$. Its other state pair, (s_3, t_6), also satisfies the inconsistency criterion $(**)$, but for the different action d and only because Q_4 has been added to $\mathit{IS}(R)$: $d \notin \mathit{refuse}(s_3, t_6)$, and $Q_3 \overset{d}{\dashrightarrow} Q$ implies $Q = Q_4 \in \mathit{IS}(R)$. Thus, all state pairs of Q_3 satisfy $(**)$ and Q_3 is

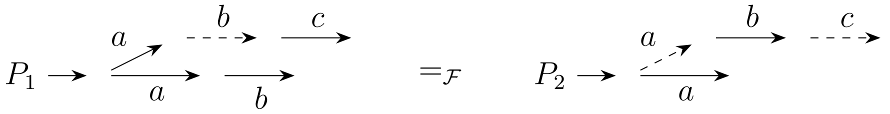

Figure 2.10: $\mathcal{F}$-refinement is action-modal

added to $IS(R)$. In other words, the inconsistency of Q_4 propagates backwards to render Q_3 inconsistent as well.

Next, (s_2, t_5) satisfies $(**)$ due to d, but $\mathit{refuse}(s_2, t_3) = \Sigma$ – and Q_2 'survives' although it contains some inconsistent state pair.

Second, we show some subtle effects. It is possible that some $(s, t) \in Q \in S \wedge T$ satisfies $(**)$, but some $Q \overset{a}{\dashrightarrow} Q'$ exists only due to (s, t) – i.e. for all other $(s', t') \in Q$, $s' \overset{a}{\not\dashrightarrow}$ or $t' \overset{a}{\not\dashrightarrow}$. It is important that such an a-may-transition is not removed from Q together with (s, t). In our example, this happens for (s_1, t_2), which satisfies $(**)$ due to a. The transition $Q_1 \overset{b}{\dashrightarrow} Q_2$ only exists because of $(s_1, t_2) \overset{b}{\dashrightarrow} (s_2, t_3)$. But the removal of the b-may-transition from R would lead to a wrong result: an MTS consisting of just two must-transitions $\xrightarrow{a}$ followed by $\xrightarrow{b}$ is a common $\mathcal{F}$-implementation of S and T, but would not be covered without the b-may-transition.

Additionally, (s_2, t_3) is only in Q_2 because of (s_1, t_2); had we identified (s_1, t_2) as inconsistent early and not generated (s_2, t_3), Q_2 would be removed. Even more subtly, (s_1, t_1) would satisfy $(**)$ due to b without the b-may-transition of Q_1, and consequently Q_1 would be removed as well.

2.3.5 May-Testing

It is maybe a bit surprising that consideration of deadlock freedom leads to a refinement notion that ignores the interplay of may- and must-transitions to a large degree. For instance, the initial states in Fig. 2.10 demonstrate the following: if $p \xrightarrow{a}$ for some state p and some $a \in \Sigma$, it does not matter which a-transitions from p are must-transitions, as long as there is at least one. Thus, despite their appearance, MTSs with $\mathcal{F}$-refinement are only *action-modal* transition systems [50] w.r.t. visible actions: it would suffice to have just one type of visible transitions, and to attach to each state a subset of Σ describing those actions that are to be treated as a must.

To distinguish the MTSs in Fig. 2.10, one can recall the approach of De Nicola and Hennessy in [30]: may- and must-testing. Note that this has nothing to do with the modalities of MTS, but instead with the reachability requirement for success during a test. In both approaches, test systems have a special *success action* ω and states that can perform ω are considered *success states.* In must-testing, a test is passed if the the parallel composition reaches a successful state in every maximal run; in may-testing it suffices to have one such successful run. Traditionally, it is required for the success actions to be performed in the latter, but this is obviously an equivalent requirement.

Must-testing is somewhat similar to our testing based on deadlock freedom and we will recall it later on (cf. Def. 2.48). De Nicola and Hennessy also examined the intersection of the two refinements thereby creating a refinement that is stricter than either. In this section, we will adapt may-testing to our approach and use it to make $\mathcal{F}$-refinement strict. Since the characterization again uses $\Longrightarrow$, but is otherwise not very intriguing, we will consider it formally for MTS while commenting on the extension to dMTS.

Definition 2.38. *In the rest of this section, a* test *is an implementation T and $\omega \in \Sigma$ is a special action, which is only allowed in tests, but not in MTS in general.*
An MTS P may-satisfies *a test T, $P\, sat_{may\text{-}as}\, T$, if the following holds: for each $S \in impl_{as}(P)$, some run from the initial state of $S \parallel_{\Sigma \setminus \{\omega\}} T$ reaches a* successful *state (s,t) with $(s,t) \xrightarrow{\omega}$. P* may-refines *P' if for all tests T: $P'\, sat_{may\text{-}as}\, T \implies P\, sat_{may\text{-}as}\, T$.*

Definition 2.39. *For an MTS P, the* must-language *of P is the set $mustL(P) = \{w \mid p_0 \overset{w}{\Longrightarrow}\}$ of* must-traces *of P. We write $P \sqsubseteq_{mustL} P'$ if $mustL(P) \supseteq mustL(P')$ (note the inverse inclusion).*

Before we prove that this reverse must-language inclusion indeed characterises may-refinement, we relate it to other refinements we examined.

Proposition 2.40. *$\sqsubseteq_{mustL}$ is strictly coarser than as-refinement; it is neither coarser nor finer than $\mathcal{F}$-refinement.*

Proof. It is easy to see that $\sqsubseteq_{as} \subset \sqsubseteq_{mustL}$. For P_3 and P_4 in Fig. 2.11, we have $P_3 \sqsubseteq_{mustL} P_4$, but not $P_3 \sqsubseteq_{\mathcal{F}} P_4$ and, hence, neither $P_3 \sqsubseteq_{as} P_4$. For P_1 and P_2 from Fig. 2.10, we have $P_2 \sqsubseteq_{\mathcal{F}} P_1$ but not $P_2 \sqsubseteq_{mustL} P_1$ (whereas $P_1 \sqsubseteq_{mustL} P_2$). □

Our next aim is to show that $\sqsubseteq_{mustL}$ characterizes may-refinement. With Lemma 2.28, one easily sees:

Proposition 2.41. *For all MTS P_1 and P_2 and all $A \subseteq \Sigma$, $mustL(P_1 \parallel_A P_2) = \{w \mid w \in u \parallel_A v \text{ for some } u \in mustL(P_1), v \in mustL(P_2)\}$. $\sqsubseteq_{mustL}$ is a precongruence for $\parallel_A$.*

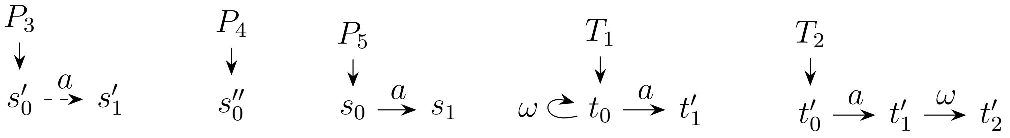

Figure 2.11: MTSs used in some proofs

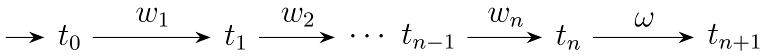

Figure 2.12: $T(w)$

Lemma 2.42. *Let $w = w_1 \cdots w_n \in \Sigma^*$ and $T(w)$ as depicted in Fig. 2.12. For all implementations S, the system $S \parallel_{\Sigma\setminus\{\omega\}} T(w)$ can reach some successful state iff $w \in mustL(S)$.*

Proof. $T(w)$ can reach t_n only by performing w and in the parallel composition this can only be done if S can perform it as well, i.e. $w \in mustL(S)$. □

Theorem 2.43. *P may-refines P' iff $P \sqsubseteq_{mustL} P'$.*

Proof. '⇒': If $w \in mustL(P')$, then $w \in mustL(S')$ for each as-implementation S' of P'. Hence, $P' \, sat_{may\text{-}as} \, T(w)$ by Lemma 2.42, and this also holds for P. Consider the as-implementation S of P that is obtained from P by removing all may-transitions that are not also must-transitions. By Lemma 2.42, we have $w \in mustL(S) = mustL(P)$.

'⇐': Consider a test T with $P' \, sat_{may\text{-}as} \, T$, and consider the as-implementation S' of P' obtained from P' as S is from P above. In $S' \parallel_{\Sigma\setminus\{\omega\}} T$, we have some $(s'_0, t_0) \overset{w}{\Longrightarrow} (s', t) \overset{\omega}{\longrightarrow}$ where ω does not occur in w. By Lemma 2.28, we have $s'_0 \overset{w}{\Longrightarrow} s'$ and $t_0 \overset{w}{\Longrightarrow} t \overset{\omega}{\longrightarrow}$. Hence, $w \in mustL(S') = mustL(P') \subseteq mustL(P)$. Since each as-implementation S of P satisfies $S \sqsubseteq_{mustL} P$, we have $w \in mustL(S)$ and we can combine a respective run with $t_0 \overset{w}{\Longrightarrow} t \overset{\omega}{\longrightarrow}$ to see that a successful state (s, t) can be reached in $S \parallel_{\Sigma\setminus\{\omega\}} T$ as required. □

We can now combine may-refinement with $\mathcal{F}$-refinement resulting in the *$mL\mathcal{F}$-refinement* $\sqsubseteq_{mL\mathcal{F}} := \sqsubseteq_{mustL} \cap \sqsubseteq_{\mathcal{F}}$. This is a finer preorder than $\mathcal{F}$-refinement alone, and it can distinguish e.g. between the two MTSs of Fig. 2.10. Since both, $\sqsubseteq_{mustL}$ and $\sqsubseteq_{\mathcal{F}}$ are precongruences for $\parallel_A$, so is $\sqsubseteq_{mL\mathcal{F}}$. In addition $\sqsubseteq_{mustL}$ turns out to be thorough, as we show in the following proposition. Together with the thoroughness of $\mathcal{F}$-refinement this means that $\sqsubseteq_{mL\mathcal{F}}$ is also thorough.

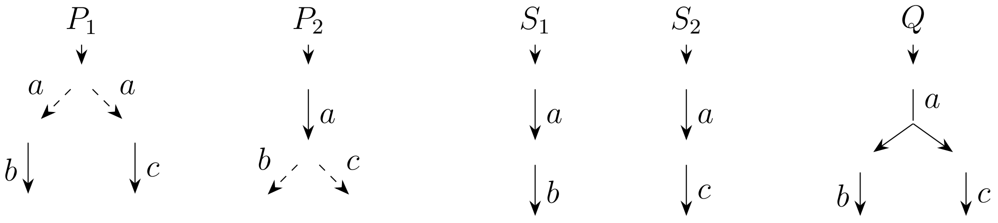

Figure 2.13: $mL\mathcal{F}$-conjunction counterexample and their dMTS-conjunction Q

Proposition 2.44. *The preorder $\sqsubseteq_{mustL}$ is thorough, i.e. $P \sqsubseteq_{mustL} Q \Leftrightarrow$ mustL-impl$(P) \subseteq$ mustL-impl(Q). Hence, $\sqsubseteq_{mL\mathcal{F}}$ is thorough.*

Proof. We only show the first claim, where '$\Rightarrow$' is analogous to the proof of Thm. 2.14. For '$\Leftarrow$', consider the *mustL*-implementation S of P obtained by deleting all may-transitions of P that are not must-transitions. Now $mustL(P) = mustL(S) \supseteq mustL(Q)$. □

When $\mathcal{F}$-refinement is restricted to $mL\mathcal{F}$-refinement, the closed conjunction operator is lost. In fact, the following proposition shows the non-existence of conjunction also for e.g. as- and w-as-refinement, since $\sqsubseteq_{as} \subset \sqsubseteq_{w\text{-}as} \subset \sqsubseteq_{mL\mathcal{F}}$ (cf. Proposition 2.74). The proof uses the example from [7, 51], where [7] proved the claim for as-refinement, whereas [51] considered a different version of weak alternating simulation, strictly between as- and w-as-refinement.

Proposition 2.45. *There exist MTSs P_1 and P_2 without a conjunction for any refinement $\sqsubseteq_R$ with $\sqsubseteq_{as} \subseteq \sqsubseteq_R \subseteq \sqsubseteq_{mL\mathcal{F}}$, i.e. there is no MTS Q such that for all MTS S: $S \sqsubseteq_R P_1 \wedge S \sqsubseteq_R P_2 \Leftrightarrow S \sqsubseteq_R Q$.*

Proof. Consider the MTSs presented in Fig. 2.13. The systems S_1 and S_2 are both as-refinements of P_1 and of P_2. Thus, they are also common $\sqsubseteq_R$-refinements of P_1 and P_2. Hence any conjunction Q of P_1 and P_2 would have to satisfy $S_i \sqsubseteq_R Q$ implying $S_i \sqsubseteq_{mL\mathcal{F}} Q$ for $i = 1, 2$. This leads to $mustL(Q) \subseteq mustL(S_1) \cap mustL(S_2) = \{\varepsilon, a\}$.

Similarly $Q \sqsubseteq_R P_2$ implies $mustL(Q) \supseteq mustL(P_2) = \{\varepsilon, a\}$ and we can conclude that $mustL(Q) = \{\varepsilon, a\}$. Therefore, there exists q with $q_0 \overset{a}{\Longrightarrow} q$ such that $q \overset{d}{\not\Longrightarrow}$ for any $d \in \Sigma$, which implies $(a, \Sigma) \in \mathcal{F}(Q)$. But since $(a, \Sigma) \notin \mathcal{F}(P_1)$, we get $Q \not\sqsubseteq_{mL\mathcal{F}} P_1$, contradicting $Q \sqsubseteq_R P_1$. □

As another variation, one could consider to replace the universal quantification by an existential one in the definitions of satisfaction, i.e. Defs. 2.3 and 2.38. The resulting two refinement preorders turn out not to be plausible at all. Some strange effects can be seen in the examples in Fig. 2.11. In the first case, P_3 satisfies $(T_1, \{a\})$ due to its implementation P_4. Hence, P_5 does not refine P_3. In the second case, P_3 satisfies $(T_2, \{a\})$ due to its implementation P_5. Hence, P_4 does not refine P_3.

2.4 Deadlock/Divergence-Testing

Besides deadlocks, also divergence is often regarded as undesirable: an infinite internal computation in a component prevents any communication with the environment; furthermore, if a parallel system is run on one processor, an infinite internal computation in one component can block the whole system. Analogously to the previous section, we now develop a testing approach where satisfaction depends on deadlock and divergence freedom. As already stated in the introduction, it seems that so far such a deadlock/divergence testing has only been mentioned once in [70], essentially in an LTS-setting and with an incorrect characterization.

In this section, we will explore deadlock/divergence-testing and a new version of failure/divergence-semantics which results from it and also compare it to the more traditional semantics arising from must-testing. We will not go into the details of traditional must-testing here; we just give some results for comparison, referring to [19] for the details.

2.4.1 Test setting and the $\mathcal{FD}$-Refinement

A technical benefit to be expected in the (still) new testing scenario is that the characterizing semantics for dMTS might be less complicated than the $\mathcal{F}$-semantics, and one could hope to thereby solve the precongruence problem we encountered in the Sect. 2.3.3, since both problems were related to divergence. Another problem with $\mathcal{F}$-refinement is that it fails to be a precongruence for hiding, already on LTS. This is the reason that there is a special treatment of divergence in the well-known denotational failures/divergence semantics for CSP in [11]. For comparison, we first define this *traditional semantics* (operationally, cf. [28]), and we immediately define it on MTS as in [19].

We start with a few definitions including the traditional failure/divergence-semantics.

Definition 2.46. *A state p of a dMTS is* divergent*, if it can perform an* *infinite sequence of τ-transitions, i.e. $p \overset{\tau^\omega}{\dashrightarrow}$. If P does not have a divergent state p, we call it* divergence free.

Note that our notion of deadlock just concerns visible behaviour; a deadlock p can be divergent at the same time.

Definition 2.47. *For any set $L \subseteq \Sigma^*$ of traces we define*

1. *$cont(L) = \{wu \mid w \in L, u \in \Sigma^*\}$; L is* continuation-closed *if $L = cont(L)$.*
2. *${}^\nabla L = \{(w, X) \mid w \in L, X \subseteq \Sigma\}$*

Definition 2.48. *The* traditional failure-divergence semantics *of an MTS P is given by:*

1. *$\mathcal{D}(P) = cont(\{w \mid p_0 \overset{w}{\Longrightarrow} p, p \text{ is divergent}\})$, the set of* divergence traces *of P*
2. *$\mathcal{F}_{\mathcal{D}t}(P) = \mathcal{F}_t(P) \ \cup \ {}^\nabla\mathcal{D}(P)$*

*P $\mathcal{FD}t$-*refines *P', written $P \sqsubseteq_{\mathcal{FD}t} P'$, if $\mathcal{D}(P) \subseteq \mathcal{D}(P')$ and $\mathcal{F}_{\mathcal{D}t}(P) \subseteq \mathcal{F}_{\mathcal{D}t}(P')$.*

Must-testing [28, 30] is suitable to justify this semantics on LTS – for image-finite specifications, to be precise. We generalized this result to MTS in [19]. A must-test is satisfied if a successful state (marked by a special success action) of the test is always reached; cf. Sect. 2.4.3 for a new variant, where the action must be performed.

Our general approach to testing aims for a refinement relation which ensures that unwanted behaviour is not introduced during refinement. Considering deadlock and divergence as unwanted, we will arrive at a different failure/divergence semantics and an $\mathcal{FD}$-refinement, which already on LTS differs from the traditional one. In fact, $\mathcal{FD}$-refinement is coarser and therefore, in our view, better, than the traditional one. Again, this alternative semantics has to our best knowledge only appeared in one publication: It is used in [69] to solve a problem posed in [5], namely to characterize an equivalence based on maximal traces and a parallel composition where synchronized actions are hidden. Nothing else seems to have been known so far – except that the implication in Prop. 2.52 below has already been proven for LTS in [69].

Definition 2.49. *For a refinement relation $\sqsubseteq_r$, a dMTS P r-*deadlock-divergence-satisfies *a test (T, A) if $S \parallel_A T$ is deadlock and divergence free*

for each $S \in impl_r(P)$. Then, we write $P\ sat_r^{dd}\ (T, A)$.
*A dMTS P r-*deadlock-divergence-refines *P', $P \leq_r^{dd} P'$, if for all tests (T, A) :*

$$P'\ sat_r^{dd}\ (T, A) \implies P\ sat_r^{dd}\ (T, A)$$

Similar as above, we will show that $\leq_{as}^{dd}$ and $\leq_{\mathcal{FD}}^{dd}$ coincide, and we will characterize them with the following (fairly) new variant of failure-divergence semantics.

Definition 2.50. *Let P be a dMTS. Its* failure-divergence semantics *is given by:*

1. *$\mathcal{D}(P)$ is defined as for MTS in Def. 2.48.*
2. *$\mathcal{F}_{\mathcal{D}}(P) = \{(w, X \cup X') \mid (w, X) \in \mathcal{F}_{st}(P), \forall a \in X' :\ wa \in \mathcal{D}(P)\} \quad \cup\ {}^{\nabla}\mathcal{D}(P)$*

*We say that P $\mathcal{FD}$-*refines *another dMTS P', $P \sqsubseteq_{\mathcal{FD}} P'$, if $\mathcal{F}_{\mathcal{D}}(P) \subseteq \mathcal{F}_{\mathcal{D}}(P')$ and $\mathcal{D}(P) \subseteq \mathcal{D}(P')$.*

We will also use the flooded language*: $L_{\mathcal{D}}(P) = \{w \mid \exists X \subseteq \Sigma : (w, X) \in \mathcal{F}_{\mathcal{D}}(P)\}$.*

The new semantics shows, whether an LTS S can deadlock or diverge: It is easy to see, that S is divergence free if and only if $\mathcal{D}(S) = \emptyset$. If it is, then S can deadlock if and only if there is some $(w, \Sigma) \in \mathcal{F}_{\mathcal{D}}(S)$. Actually, S can deadlock or diverge if and only if $(w, \Sigma) \in \mathcal{F}_{\mathcal{D}}(S)$. Another pleasing property is that $\mathcal{F}_{\mathcal{D}}(P)$ is always $\mathcal{F}$-consistent, whereas $\mathcal{F}_{st}(P)$ might violate FC2.

Lemma 2.51.

1. *An LTS S can deadlock or diverge if and only if $(w, \Sigma) \in \mathcal{F}_{\mathcal{D}}(S)$ for some $w \in \Sigma^*$.*
2. *For an MTS P: $w \notin \mathcal{D}(P) \wedge (w, X) \in \mathcal{F}_t(P) \implies (w, X) \in \mathcal{F}_{st}(P)$*
3. *For a dMTS P:*
   (a) *$\mathcal{F}_{\mathcal{D}}(P)$ is $\mathcal{F}$-consistent.*
   (b) *If $p_0 \overset{w}{\Longrightarrow}$, then $w \in \mathcal{D}(P) \vee (w, \emptyset) \in \mathcal{F}_{st}(P)$, which is equivalent to $w \in L_{\mathcal{D}}(P)$.*
   (c) *If $\forall a \in \overline{X} : wa \in \mathcal{D}(P)$ and $(w, X_0) \in \mathcal{F}_{\mathcal{D}}(P)$, then $(w, X_0 \cup \overline{X}) \in \mathcal{F}_{\mathcal{D}}(P)$.*

Proof. 1. If S can diverge, then there is some divergent state s reachable by some $w \in \Sigma^*$. By definition $w \in \mathcal{D}(S)$ and thus $(w, \Sigma) \in {}^{\nabla}\mathcal{D}(S) \subseteq \mathcal{F}_{\mathcal{D}}(S)$.

Otherwise, if S can deadlock, then there is some state s, reachable by some $w \in \Sigma^*$, that can refuse all actions in Σ. Since S is divergence free, we have $s \overset{\varepsilon}{\Longrightarrow} s'$ for some (must-)stable s', which can still refuse all actions. Thus $(w, \Sigma) \in \mathcal{F}_{st}(S) \subseteq \mathcal{F}_{\mathcal{D}}(S)$.

Let $(w, \Sigma) \in \mathcal{F}_{\mathcal{D}}(S)$ and assume that S cannot diverge. Then $(w, \Sigma) \in \mathcal{F}_{st}(S)$ and S can deadlock.

2. $(w, X) \in \mathcal{F}_t(P)$ implies $p_0 \overset{w}{=\!\!\Rightarrow} p$ for some suitable $p \in P$. Since p cannot diverge by assumption, there is a stable (and thus must-stable) state p' with $p \overset{\varepsilon}{\Longrightarrow} p'$. Now $p' \overset{a}{\not\longrightarrow}$ for all $a \in X$, otherwise $p \overset{\varepsilon}{\Longrightarrow} p' \overset{a}{\longrightarrow}$ would contradict $a \in X$ with $(w, X) \in \mathcal{F}_t(P)$.

3a. Easy. For FC2 note that $w \in \mathcal{D}(P)$ or $(w, X) \in \mathcal{F}_{st}(P)$ implies that $v \in \mathcal{D}(P)$ or $(v, \emptyset) \in \mathcal{F}_{st}(P)$ for all $v \sqsubseteq w$, since after performing v the system can must-stabilize or diverge as argued in 2.

3b. We have $p_0 \overset{w}{=\!\!\Rightarrow} p$ for some $p \in P$. If p diverges, then $w \in \mathcal{D}(P)$ and we are done. Otherwise, $p \overset{\varepsilon}{=\!\!\Rightarrow} p'$ for some must-stable p', as above. Therefore $p_0 \overset{w}{=\!\!\Rightarrow} p'$ and $(w, \emptyset) \in \mathcal{F}_{st}(P)$. The second statement is easy once we note that $(w, X) \in \mathcal{F}_{\mathcal{D}}(P)$ implies $(w, \emptyset) \in \mathcal{F}_{\mathcal{D}}(P)$ by FC2.

3c. If $w \in \mathcal{D}(P)$, then $(w, X_0 \cup \overline{X}) \in {}^{\nabla}\mathcal{D}(P) \subseteq \mathcal{F}_{\mathcal{D}}(P)$ and we are done. Otherwise $(w, X_0) \in \mathcal{F}_{\mathcal{D}}(P)$ implies some $(w, X) \in \mathcal{F}_{st}(P)$ and some X' such that $\forall a \in X' : wa \in \mathcal{D}(P)$ with $X \cup X' = X_0$. Since $\forall a \in \overline{X} : wa \in \mathcal{D}(P)$, we get $\forall a \in X' \cup \overline{X} : wa \in \mathcal{D}(P)$ and thus $(w, X_0 \cup \overline{X}) \in \mathcal{F}_{\mathcal{D}}(P)$. □

To get further acquainted with the new refinement, we compare it to the traditional version. Since the latter has not been examined for dMTS, we compare them on MTS. We show that our new refinement is strictly coarser, i.e. it does not imply the traditional one but is implied by the latter.

To see the first claim, consider LTSs P and P' from Fig. 2.14. We have $\mathcal{D}(P) = \emptyset$ and $\mathcal{D}(P') = a\Sigma^*$. Furthermore, we have $\mathcal{F}_{\mathcal{D}}(P) = \{(\varepsilon, X) \mid X \subseteq \Sigma\} = \mathcal{F}_{\mathcal{D}t}(P)$ and $\mathcal{F}_{\mathcal{D}}(P') \supseteq \{(\varepsilon, X) \mid X \subseteq \Sigma\}$, but $(\varepsilon, \Sigma) \notin \mathcal{F}_{\mathcal{D}t}(P')$. Thus $P \sqsubseteq_{\mathcal{F}\mathcal{D}} P'$, but $P \not\sqsubseteq_{\mathcal{F}\mathcal{D}t} P'$.

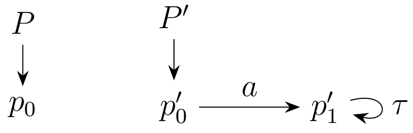

Figure 2.14: LTSs P and P' proving that $P \sqsubseteq_{\mathcal{F}\mathcal{D}} P' \implies P \sqsubseteq_{\mathcal{F}\mathcal{D}t} P'$ does not hold in general.

Proposition 2.52. *For MTSs P and P' we have $P \sqsubseteq_{\mathcal{F}\mathcal{D}t} P' \implies P \sqsubseteq_{\mathcal{F}\mathcal{D}} P'$. The reverse implication does not hold in general, not even for LTSs.*

Proof. It only remains to show the implication. Given $P \sqsubseteq_{\mathcal{F}\mathcal{D}t} P'$, we have $\mathcal{D}(P) \subseteq \mathcal{D}(P')$. Consider $(w, X_0) \in \mathcal{F}_{\mathcal{D}}(P)$. If $w \in \mathcal{D}(P')$ we are done; otherwise, $w \notin \mathcal{D}(P)$ and we have $X_0 = X \cup X'$ with $(w, X) \in \mathcal{F}_{st}(P) \subseteq \mathcal{F}_t(P) \subseteq \mathcal{F}_{\mathcal{D}t}(P) \subseteq \mathcal{F}_{\mathcal{D}t}(P')$ and $wa \in \mathcal{D}(P)$ for all $a \in X'$. Thus, $(w, X) \in \mathcal{F}_t(P')$ and from Lemma 2.51.2 we get $(w, X) \in \mathcal{F}_{st}(P')$. Recalling $\mathcal{D}(P) \subseteq \mathcal{D}(P')$, we conclude $(w, X \cup X') \in \mathcal{F}_{\mathcal{D}}(P')$ and we are done. □

Intuitively, the $\mathcal{F}_{\mathcal{D}t}$- and the $\mathcal{F}_{\mathcal{D}}$-semantics agree that divergence is catastrophic and that failure information is important for deadlock-avoidance. For the LTSs in Fig. 2.14, the $\mathcal{F}_{\mathcal{D}t}$-semantics sees the difference that P can refuse a initially, while P' surely offers it. In principle, this is very relevant if e.g. a is a start signal for another component. But here, it will hardly make a difference since the overall system runs into a catastrophe after a. Hence, we believe that our new semantics is practically more relevant.

The next step is to show that our semantics is compositional for parallel composition.

Theorem 2.53. *For all dMTS P and Q and all $A \subseteq \Sigma$:*

1. $\mathcal{D}(P \parallel_A Q) = \mathit{cont}\big(\mathcal{D}(P) \parallel_A L_{\mathcal{D}}(Q) \;\cup\; L_{\mathcal{D}}(P) \parallel_A \mathcal{D}(Q)\big)$

2. $\mathcal{F}_{\mathcal{D}}(P \parallel_A Q) = {}^{\nabla}\mathcal{D}(P \parallel_A Q) \;\cup$
   $\{(w, X \cup X') \mid (w, X) \in \mathcal{F}_{\mathcal{D}}(P) \parallel_A \mathcal{F}_{\mathcal{D}}(Q), \forall a \in X' : wa \in \mathcal{D}(P \parallel_A Q)\}$

Proof. 1. Note that a state (p, q) is divergent, if and only if p is divergent or q is divergent (possibly both), because τs are not synchronized and no new τ can arise during parallel composition. Since both sides are continuation-closed, it suffices to show that each prefix-minimal w in one side is contained in the other.

'$\subseteq$': Let w be prefix-minimal in $\mathcal{D}(P \parallel_A Q)$, thus $(p_0, q_0) \overset{w}{=\!\!\Rightarrow} (p, q)$ for some divergent (p, q). As just noted, this implies that p or q is divergent – say, p is. We also know that $p_0 \overset{u}{=\!\!\Rightarrow} p$ and $q_0 \overset{v}{=\!\!\Rightarrow} q$ such that $w \in u \parallel_A v$. Then $u \in \mathcal{D}(P)$ and $v \in L_{\mathcal{D}}(Q)$ by Lemma 2.51.3b.

'$\supseteq$': Consider a prefix-minimal element w in the r.h.s., say $w \in u \parallel_A v$ for some $u \in \mathcal{D}(P)$ and $v \in L_{\mathcal{D}}(Q)$. By choice of w, u must be prefix-minimal in $\mathcal{D}(P)$, implying $p_0 \overset{u}{=\!\!\Rightarrow} p$ for some divergent p. Otherwise, u would have a proper prefix in $\mathcal{D}(P)$, which together with a prefix of v would lead to a proper prefix of w in the r.h.s. By Lemma 2.51.3b, $v \in \mathcal{D}(Q)$ and the same argument shows that $q_0 \overset{v}{=\!\!\Rightarrow} q$, or we have $(v, \emptyset) \in \mathcal{F}_{st}(Q)$ implying the same. Thus $(p_0, q_0) \overset{w}{=\!\!\Rightarrow} (p, q)$ and (p, q) is divergent.

2.'$\subseteq$': Consider $(w, X_0) \in \mathcal{F}_{\mathcal{D}}(P \parallel_A Q)$. If $w \in {}^{\nabla}\mathcal{D}(P \parallel_A Q)$, we are done. So assume otherwise, i.e. $X_0 = X \cup X'$, $(w, X) \in \mathcal{F}_{st}(P \parallel_A Q)$ and

$X' \subseteq \{a \mid wa \in \mathcal{D}(P \parallel_A Q)\}$. Since the flooding with X' is also performed on the r.h.s., it remains to show that $(w, X) \in \mathcal{F}_\mathcal{D}(P) \parallel_A \mathcal{F}_\mathcal{D}(Q)$

Assume $(w, X) \in \mathcal{F}_{st}(P \parallel_A Q)$ due to $(p_0, q_0) \overset{w}{\Longrightarrow} (p, q)$ with (p, q) must-stable. Thus, there are u, v such that $p_0 \overset{u}{\Longrightarrow} p$, $q_0 \overset{v}{\Longrightarrow} q$, p and q are must-stable and $w \in u \parallel_A v$. Choose $Y = \{a \mid p \overset{a}{\not\longrightarrow}\}$ and $Z = \{a \mid q \overset{a}{\not\longrightarrow}\}$.

By the definition of $\parallel_A$ on $\mathcal{F}$-sets, we only have to check that $X \subseteq (Y \cap Z) \cup (A \cap (Y \cup Z))$. For $a \in X$ we have $(p, q) \overset{a}{\not\longrightarrow}$. If $a \in A$, we get that $p \overset{a}{\not\longrightarrow}$ or $q \overset{a}{\not\longrightarrow}$ and thus $a \in A \cap (Y \cup Z)$. If $a \notin A$, we get that $p \overset{a}{\not\longrightarrow}$ and $q \overset{a}{\not\longrightarrow}$ and thus $a \in Y \cap Z$.

'$\supseteq$': The first set, ${}^\nabla\mathcal{D}(P \parallel_A Q)$, is obviously contained in the l.h.s., so we examine some $(w, X \cup X')$ with $(w, X) \in \mathcal{F}_\mathcal{D}(P) \parallel_A \mathcal{F}_\mathcal{D}(Q)$ and $\forall a \in X' : wa \in \mathcal{D}(P \parallel_A Q)\}$. Due to Lemma 2.51.3c, we only have to show that (w, X) is contained in the l.h.s.

By the definition of $\parallel_A$, we have $(u, Y) \in \mathcal{F}_\mathcal{D}(P)$ and $(v, Z) \in \mathcal{F}_\mathcal{D}(Q)$ with $w \in u \parallel_A v$ and $X \subseteq (Y \cap Z) \cup (A \cap (Y \cup Z))$. If $u \in \mathcal{D}(P)$ or $v \in \mathcal{D}(Q)$, then $(w, X) \in {}^\nabla\mathcal{D}(P \parallel_A Q)$ because of the first claim and we are done.

So we have $p_0 \overset{u}{\Longrightarrow} p$ and $q_0 \overset{v}{\Longrightarrow} q$ for some must-stable p and q and $(p_0, q_0) \overset{w}{\Longrightarrow} (p, q)$ with (p, q) being must-stable. For $a \in X$ we consider the cases $a \in A$ and $a \notin A$:

Take $a \in X \setminus A$. Then $a \in Y \cap Z$ and we have $p \overset{a}{\not\longrightarrow}$ or $ua \in \mathcal{D}(P)$ and we have $q \overset{a}{\not\longrightarrow}$ or $va \in \mathcal{D}(Q)$. There are two possibilities: if $p \overset{a}{\not\longrightarrow}$ and $q \overset{a}{\not\longrightarrow}$, then $(p, q) \overset{a}{\not\longrightarrow}$. Otherwise $p_0 \overset{ua}{\Longrightarrow} p'$ with p' divergent or $q_0 \overset{va}{\Longrightarrow} q'$ with q' divergent, possibly both – say the first. Then $(p_0, q_0) \overset{wa}{\Longrightarrow} (p', q)$ and (p', q) is divergent, hence $wa \in \mathcal{D}(P \parallel_A Q)$. In any case, $a \in X$ is justified on the l.h.s.

Take $a \in X \cap A$. By definition $a \in A \cap (Y \cup Z)$, thus we have $p \overset{a}{\not\longrightarrow}$ or $ua \in \mathcal{D}(P)$ or we have $q \overset{a}{\not\longrightarrow}$ or $va \in \mathcal{D}(Q)$; possibly we have some combination. Again there are two possibilities: if $p \overset{a}{\not\longrightarrow}$ or $q \overset{a}{\not\longrightarrow}$, then $(p, q) \overset{a}{\not\longrightarrow}$. Otherwise, say $p_0 \overset{ua}{\Longrightarrow} p'$ with p' divergent and $q_0 \overset{v}{\Longrightarrow} q \overset{a}{\dashrightarrow} q'$; then also $(p_0, q_0) \overset{wa}{\Longrightarrow} (p', q')$ with (p', q') divergent. Thus we are done. □

As usual, compositionality implies that the respective refinement is a precongruence. For divergence-free dMTS P and Q, $\mathcal{D}(P)$ is empty and $\mathcal{F}_\mathcal{D}(P)$ reduces to $\mathcal{F}_{st}(P)$ and similarly for Q; according to Thm. 2.53, also $\mathcal{D}(P \parallel_A Q)$ is empty and $\mathcal{F}_\mathcal{D}(P \parallel_A Q)$ is the parallel composition of $\mathcal{F}_{st}(P)$ and $\mathcal{F}_{st}(Q)$. With Prop. 2.22, this implies the precongruence result announced in the previous section.

Corollary 2.54. *1. $\sqsubseteq_{\mathcal{FD}}$ is a precongruence for $\parallel_A$, i.e. $P \sqsubseteq_{\mathcal{FD}} P' \implies P \parallel_A Q \sqsubseteq_{\mathcal{FD}} P' \parallel_A Q$.*

2. *For divergence-free dMTS, $\sqsubseteq_{\mathcal{F}}$ is a precongruence for parallel composition.*

It is interesting to note, that the calculation of $\mathcal{F}_{\mathcal{D}}(P \parallel_A Q)$ is simpler if P and Q synchronize on their common actions rather than on some arbitrary set. In this case, the flooding with X' is not needed. Formally:

Proposition 2.55. *Let P and Q be dMTSs and A be the intersection of their alphabets. Then, $\mathcal{F}_{\mathcal{D}}(P \parallel_A Q) = {}^{\nabla}\mathcal{D}(P \parallel_A Q) \cup (\mathcal{F}_{\mathcal{D}}(P) \parallel_A \mathcal{F}_{\mathcal{D}}(Q))$.*

Proof. '$\supseteq$' is implied by Thm. 2.53.2, since ${}^{\nabla}\mathcal{D}(P \parallel_A Q) \cup \mathcal{F}_{\mathcal{D}}(P) \parallel_A \mathcal{F}_{\mathcal{D}}(Q)$ is included in the r.h.s. there.

'$\subseteq$': Consider $(w, X_0) \in \mathcal{F}_{\mathcal{D}}(P \parallel_A Q)$. If $w \in \mathcal{D}(P \parallel_A Q)$, we are done. So assume otherwise, i.e. $X_0 = X \cup X'$, $(w, X) \in \mathcal{F}_{st}(P \parallel_A Q)$ and $X' \subseteq \{a \mid wa \in \mathcal{D}(P \parallel_A Q)\}$.

Assume $(w, X) \in \mathcal{F}_{st}(P \parallel_A Q)$ due to $(p_0, q_0) \overset{w}{\Longrightarrow} (p, q)$ with (p, q) must-stable. Thus, there are u, v such that $p_0 \overset{u}{\Longrightarrow} p$, $q_0 \overset{v}{\Longrightarrow} q$, p and q are must-stable and $w \in u \parallel_A v$. Note that u and v are unique: Due to the synchronization on common actions, they are the projections of w to the alphabets of P and Q respectively. Choose $Y = \{a \mid p \overset{a}{\not\longrightarrow}\} \cup \{a \mid ua \in \mathcal{D}(P)\}$ and $Z = \{a \mid q \overset{a}{\not\longrightarrow}\} \cup \{a \mid va \in \mathcal{D}(Q)\}$.

By the definition of $\parallel_A$ on $\mathcal{F}$-sets, we have to check that $X_0 \subseteq (Y \cap Z) \cup (A \cap (Y \cup Z))$. For $a \in X$, we proceed as in the previous proof: We have $(p, q) \overset{a}{\not\longrightarrow}$. If $a \in A$, we get that $p \overset{a}{\not\longrightarrow}$ or $q \overset{a}{\not\longrightarrow}$ and thus $a \in A \cap (Y \cup Z)$. If $a \notin A$, we get that $p \overset{a}{\not\longrightarrow}$ and $q \overset{a}{\not\longrightarrow}$ and thus $a \in Y \cap Z$.

For $a \in X'$, we have $wa \in \mathcal{D}(P \parallel_A Q)$. Since $w \notin \mathcal{D}(P \parallel_A Q)$, there must be $(p_0, q_0) \overset{w}{\Longrightarrow} (p', q') \overset{a}{\dashrightarrow} (p_d, q_d)$ where (p_d, q_d) is divergent while (p', q') is not. Hence p_d or q_d is divergent (possibly both), but neither p' nor q' is. Due to the uniqueness of u and v, we have $p_0 \overset{u}{\Longrightarrow} p'$, $q_0 \overset{v}{\Longrightarrow} q'$. The following two subcases remain:

If $a \in A$, we get that $p' \overset{a}{\dashrightarrow} p_d$ and $q' \overset{a}{\dashrightarrow} q_d$. W.l.o.g. we assume p_d to be divergent. This implies $ua \in \mathcal{D}(P)$ and thus $a \in A \cap (Y \cup Z)$.

If $a \notin A$, we get that $p' \overset{a}{\dashrightarrow} p_d$ or $q' \overset{a}{\dashrightarrow} q_d$, say the former. Since a is not a common action, we have $q \overset{a}{\not\longrightarrow}$ implying $a \in Z$. Furthermore, p_d diverges since $q_d = q'$ does not. Thus $ua \in \mathcal{D}(P)$, implying $a \in Y$ and $a \in Y \cap Z$. □

To see that the flooding with X' is needed in general when determining $\mathcal{F}_{\mathcal{D}}(P \parallel_A Q)$, consider P and Q from Fig. 2.15. They do not synchronize at all in $P \parallel_\emptyset Q$, even though they have the action a in common. We have $(\varepsilon, \{a\}) \in \mathcal{F}_{\mathcal{D}}(P \parallel_\emptyset Q)$, but $\varepsilon \notin \mathcal{D}(P \parallel_\emptyset Q)$ and $(\varepsilon, \{a\}) \notin \mathcal{F}_{\mathcal{D}}(P) \parallel_\emptyset \mathcal{F}_{\mathcal{D}}(Q)$, because ε is only associated to refusal sets $\subseteq \Sigma \setminus \{a\}$ in $\mathcal{F}_{\mathcal{D}}(Q)$.

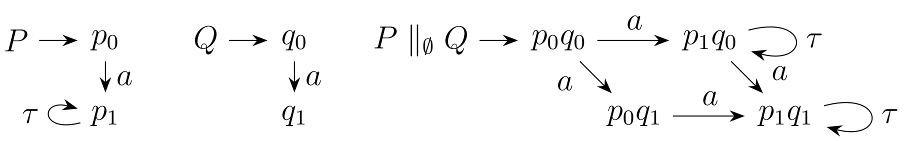

Figure 2.15: LTSs P, Q and $P \parallel_\emptyset Q$

Now we show that $\sqsubseteq_{\mathcal{FD}}$ coincides with the test preorders based on as- or $\mathcal{FD}$-implementations. We do not examine testing based on $\mathcal{F}$-implementations, since $\mathcal{F}$-refinement can freely introduce divergences, so every dMTS would have implementations that diverge initially and thus fail all tests. Again we need some lemmata. The first is the key result connecting the $\mathcal{FD}$-semantics to testing.

Lemma 2.56. *Let P and P' be dMTSs.*

1. $P \sqsubseteq_{as} P' \implies P \sqsubseteq_{\mathcal{FD}} P'$.
2. *(a)* $\mathcal{D}(P) = \bigcup_{S \in impl_{as}(P)} \mathcal{D}(S) = \bigcup_{S \in impl_{\mathcal{FD}}(P)} \mathcal{D}(S)$
   (b) $\mathcal{F}_\mathcal{D}(P) = \bigcup_{S \in impl_{as}(P)} \mathcal{F}_\mathcal{D}(S) = \bigcup_{S \in impl_{\mathcal{FD}}(P)} \mathcal{F}_\mathcal{D}(S)$

Proof. 1. Take a $w \in \mathcal{D}(P)$. We have $p_0 \overset{w}{\Longrightarrow} p \overset{\tau^\omega}{\dashrightarrow}$. By $p_0 \sqsubseteq_{as} p'_0$, each of these transitions must be matched and thus $p'_0 \overset{w}{\Longrightarrow} p' \overset{\tau^\omega}{\dashrightarrow}$ justifying $w \in \mathcal{D}(P')$.

Take an element of $\mathcal{F}_\mathcal{D}(P)$. If it is in ${}^\nabla\mathcal{D}(P)$, we are done by the above. So consider $(w, X \cup X') \in \mathcal{F}_\mathcal{D}(P)$ with each $a \in X'$ satisfying $wa \in \mathcal{D}(P)$ and with $(w, X) \in \mathcal{F}_{st}(P)$ being justified by some p with $p_0 \overset{w}{\Longrightarrow} p$. This must be matched by some $p'_0 \overset{w}{\Longrightarrow} p'$ with $p \sqsubseteq_{as} p'$. Since p is must-stable, so is p' by definition of an as-relation. For all $a \in X$ we have $p \overset{a}{\not\longrightarrow}$ and thus $p' \overset{a}{\not\longrightarrow}$, so p' justifies $(w, X) \in \mathcal{F}_{st}(P')$. Since $wa \in \mathcal{D}(P) \subseteq \mathcal{D}(P')$ for all $a \in X'$, we have $(w, X \cup X') \in \mathcal{F}_\mathcal{D}(P')$ and are done.

2. We have to check three inclusions for each subcase. For both (a) and (b), the second inclusion follows from (1) and the third from the definition of $\sqsubseteq_{\mathcal{FD}}$. So we prove the first:

2a. For $\mathcal{D}(P) \subseteq \bigcup_{S \in impl_{as}(P)} \mathcal{D}(S)$ consider $S = (P, p_0, \dashrightarrow_P, \dashrightarrow_P)$ with $\mathcal{D}(P) = \mathcal{D}(S)$.

2b. For $\mathcal{F}_\mathcal{D}(P) \subseteq \bigcup_{S \in impl_{as}(P)} \mathcal{F}_\mathcal{D}(S)$ consider $(w, X_0) \in \mathcal{F}_\mathcal{D}(P)$. If $w \in \mathcal{D}(P)$ we are done by (a); otherwise, $X_0 = X \cup X'$ with $(w, X) \in \mathcal{F}_{st}(P)$ justified by $\overline{p}$ and $\forall a \in X' : wa \in \mathcal{D}(P)$. We construct the implementation by implementing all may-transitions and almost duplicating the state $\overline{p}$ with

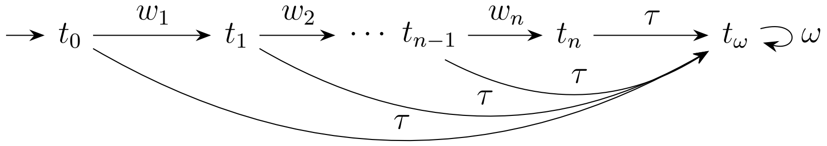

Figure 2.16: $T_\tau(w) = w_1 \dots w_n$

s: $S = (P \uplus \{s\}, p_0, \longrightarrow_S, \longrightarrow_S)$ with $\longrightarrow_S = \dashrightarrow_P \cup \{(\overline{p}, \tau, s)\} \cup \{(s, a, p') \mid \overline{p} \xrightarrow{a} P', p' \in P'\}$. S is an as-implementation of P due to the relation $id_P \cup \{(s, \overline{p})\}$, since $\overline{p}$ is must-stable in P, though possibly not in S.

Clearly, $p_0 \overset{w}{\Longrightarrow}_S \overline{p} \xrightarrow{\tau} s$ and s justifies $(w, X) \in \mathcal{F}_{st}(S)$. Since S has all transitions of P, we have $\mathcal{D}(P) = \mathcal{D}(S)$ implying $wa \in \mathcal{D}(S)$ for all $a \in X'$. Therefore $(w, X \cup X') \in \mathcal{F}_{\mathcal{D}}(S)$. □

The second lemma shows how we can test for an element of the $\mathcal{D}$- or $\mathcal{F}_{\mathcal{D}}$-sets. Note that Part 2b is unusually weak. Also note that the previous results in this section, except for Lem. 2.51, are about properties of the $\mathcal{FD}$-semantics and not concerned with testing.

Lemma 2.57. *Let S be an LTS, ω an action not in the alphabet of S and $A = \Sigma \setminus \{\omega\}$. For $T(w, X)$ refer to Fig. 2.4 in Sect. 2.3.2; for $T_\tau(w)$ see Fig 2.16.*

1. *$w \in \mathcal{D}(S)$ if and only if $S \parallel_A T_\tau(w)$ can diverge, which is equivalent to $S \parallel_A T_\tau(w)$ can diverge or deadlock.*
2. (a) *If $(w, X) \in \mathcal{F}_{\mathcal{D}}(S)$, then $S \parallel_A T(w, X)$ can deadlock or diverge.*

   (b) *If $S \parallel_A T(w, X)$ can deadlock or diverge, then $w \in \mathcal{D}(S)$, $(w, X) \in \mathcal{F}_{st}(S)$ or $\exists a \in X : wa \in \mathcal{D}(S)$.*

Proof. 1. For the latter equivalence, observe that by construction of T and $\omega \notin A$, $S \parallel_A T_\tau(w)$ can never deadlock.

For the if-part, we have S such that $S \parallel_A T_\tau(w)$ can diverge. Since $T(w)$ cannot diverge, there is some divergent s such that $s_0 \overset{v}{\Longrightarrow} s$ with $v \sqsubseteq w$, implying $w \in \mathcal{D}(S)$.

For the only-if-part, we have S with $w \in \mathcal{D}(S)$; hence, for some $i \leq n$ and $v = w_1 \dots w_i$, $s_0 \overset{v}{\Longrightarrow} s$ for some divergent s. One can easily see that $S \parallel_A T_\tau(w)$ can diverge, because of the run $(s_0, t_0) \overset{v}{\Longrightarrow} (s, t_i)$ and (s, t_i) being divergent.

2a. Since $(w, X \cup \{\omega\}) \in \mathcal{F}_{\mathcal{D}}(S)$ and $(w, (\Sigma \setminus X) \cup \{\omega\}) \in \mathcal{F}_{\mathcal{D}}(T(w, X))$ we have $(w, \Sigma) \in \mathcal{F}_{\mathcal{D}}(S \parallel_A T(w, X))$ by Thm. 2.53 and the definition of $\parallel_A$ on

$\mathcal{F}$-sets: ω is refused by both and each other $a \in \Sigma$ by one of the components. Hence we are done by Lemma 2.51.1.

2b. If $S \parallel_A T(w, X)$ can diverge in some state (s, t), then it diverges due to s. If $t = t'$, it already diverges in some (s, t_i), so it does so in any case. If $i \leq n$, then $w \in \mathcal{D}(S)$. Otherwise, $i = n+1$ and there is $a \in X : wa \in \mathcal{D}(S)$.

If $S \parallel_A T(w, X)$ cannot diverge, it can deadlock. This is only possible in some (s, t_n), since all other states can perform ω. It also means $s \overset{a}{\not\Longrightarrow}$ for all $a \in X$. Thus $(w, X) \in \mathcal{F}_{st}(S)$ by Lemma 2.51.2. □

Finally, the following lemma extracts a common part of several proofs; we will use it three times.

Lemma 2.58. *Let S be an $\mathcal{FD}$-implementation of a dMTS P and (T, A) a test. If $S \parallel_A T$ can deadlock or diverge, then there is some as-implementation S' of P such that $S' \parallel_A T$ can deadlock or diverge.*

Proof. If $S \parallel_A T$ can diverge, there is some prefix-minimal $w \in \mathcal{D}(S \parallel_A T)$. By Thm. 2.53, we get that $w \in \mathcal{D}(S) \parallel_A L_{\mathcal{D}}(T) \cup L_{\mathcal{D}}(S) \parallel_A \mathcal{D}(T)$. If w is in the first subset, we have some $u \in \mathcal{D}(S) \subseteq \mathcal{D}(P)$ and $v \in L_{\mathcal{D}}(T)$ such that $w \in u \parallel_A v$. By 2.56.2. there is some as-implementation S' of P with $u \in \mathcal{D}(S')$. Thus we get $w \in \mathcal{D}(S') \parallel_A L_{\mathcal{D}}(T)$, i.e. $S' \parallel_A T$ can diverge.

If w is in the second subset, we have some $(u, \emptyset) \in \mathcal{F}_{\mathcal{D}}(S) \subseteq \mathcal{F}_{\mathcal{D}}(P)$ and $v \in \mathcal{D}(T)$ such that $w \in u \parallel_A v$. By 2.56.2, there is $S' \in \mathit{impl}_{as}(P)$ with $(u, \emptyset) \in \mathcal{F}_{\mathcal{D}}(S')$, i.e. $u \in L_{\mathcal{D}}(S')$. Thus we get $w \in L_{\mathcal{D}}(S') \parallel_A \mathcal{D}(T)$, i.e. $S' \parallel_A T$ can diverge.

If $S \parallel_A T$ cannot diverge, it can deadlock and $\mathcal{D}(S \parallel_A T) = \emptyset$. This means, by Lemma 2.51.1 and Thm. 2.53, that there is some $(w, \Sigma) \in \mathcal{F}_{\mathcal{D}}(S) \parallel_A \mathcal{F}_{\mathcal{D}}(T)$, i.e. there are $(u, Y) \in \mathcal{F}_{\mathcal{D}}(S) \subseteq \mathcal{F}_{\mathcal{D}}(P)$ and $(v, Z) \in \mathcal{F}_{\mathcal{D}}(T)$ with $(w, X) \in (u, Y) \parallel_A (v, Z)$. By 2.56.2 we find some $S' \in \mathit{impl}_{as}(P)$ with $(u, Y) \in \mathcal{F}_{\mathcal{D}}(S')$. Thus $(w, \Sigma) \in \mathcal{F}_{\mathcal{D}}(S') \parallel_A \mathcal{F}_{\mathcal{D}}(T) \subseteq \mathcal{F}_{\mathcal{D}}(S' \parallel_A T)$ implying (again by Lemma 2.51.1) that $S' \parallel_A T$ can deadlock or diverge. □

Theorem 2.59. *For all dMTSs, as-deadlock-divergence-refinement and $\sqsubseteq_{\mathcal{FD}}$ coincide.*

Proof. 'P as-deadlock-divergence-refines $P' \Rightarrow P \sqsubseteq_{\mathcal{FD}} P'$':
For $\mathcal{D}$-inclusion, the argument is quite the same as for must-testing in [19], since divergence only depends on may-transitions. We present it only for completeness. Consider a prefix-minimal $w \in \mathcal{D}(P)$. By Lemma 2.56.2, there is some as-implementation S of P with $w \in \mathcal{D}(S)$. Due to Lemma 2.57.1, $S \parallel_A T_\tau(w)$ can diverge. Thus P fails the test, and, by our premise, so does P'. This implies the existence of some $S' \in \mathit{impl}_{as}(P')$ such that $S' \parallel_A T_\tau(w)$

can deadlock or diverge. Due to Lemma 2.57.1 $w \in \mathcal{D}(S')$ and, finally, $w \in \mathcal{D}(P')$ by Lemma 2.56.2.

Now consider $(w, X_0) \in \mathcal{F}_{\mathcal{D}}(P)$. If $w \in \mathcal{D}(P')$ we are done, so let $w \notin \mathcal{D}(P')$. By Lemma 2.56.2 there is an implementation $S \in \mathit{impl}_{as}(P)$ with $(w, X_0) \in \mathcal{F}_{\mathcal{D}}(S)$. Let $X = X_0 \setminus \{a \mid wa \in \mathcal{D}(P')\}$ – note the use of P'. Then, we have $(w, X) \in \mathcal{F}_{\mathcal{D}}(S)$ by $\mathcal{F}$-consistency of $\mathcal{F}_{\mathcal{D}}(S)$. By Lemma 2.57.2a, $\neg P \ \mathit{sat}_{as}^{dd} \ (T(w, X), A)$ for the respective test due to S. By assumption $\neg P' \ \mathit{sat}_{as}^{dd} \ (T(w, X), A)$, which by definition means that the test fails due to some $S' \in \mathit{impl}_{as}(P')$. By Lemma 2.57.2b we get $w \in \mathcal{D}(S')$, $(w, X) \in \mathcal{F}_{st}(S')$ or $\exists a \in X : wa \in \mathcal{D}(S')$. The first and the last possibility cannot hold by $w \notin \mathcal{D}(P')$, choice of X and (due to Lemma 2.56.2) $\mathcal{D}(S') \subseteq \mathcal{D}(P')$. So we have $(w, X) \in \mathcal{F}_{st}(S') \subseteq \mathcal{F}_{\mathcal{D}}(S') \subseteq \mathcal{F}_{\mathcal{D}}(P')$. With Lemma 2.51.3c we get $(w, X_0) \in \mathcal{F}_{\mathcal{D}}(P')$ and we are done.

'P as-deadlock-divergence-refines $P' \Leftarrow P \sqsubseteq_{\mathcal{FD}} P'$':
We prove the as-deadlock-divergence-refinement by contraposition and assume $\neg P \ \mathit{sat}_{as}^{dd} \ (T, A)$ for some test (T, A). Thus, there must be an as-implementation S of P where $S \parallel_A T$ can diverge or deadlock. By Lem. 2.56.1 S is an $\mathcal{FD}$-implementation of P and, by transitivity, of P'. Therefore, by Lemma 2.58, there is some $S' \in \mathit{impl}_{as}(P')$ such that $S' \parallel_A T$ can deadlock or diverge, implying $\neg P' \ \mathit{sat}_{as}^{dd} \ (T, A)$. □

2.4.2 Robustness, Expressivity and Precongruence Results

Lemma 2.60. *For each test (T, A), $P \ \mathit{sat}_{\mathcal{FD}}^{dd} \ (T, A) \Leftrightarrow P \ \mathit{sat}_{as}^{dd} \ (T, A)$.*

Proof. We prove this by showing $\neg P \ \mathit{sat}_{\mathcal{FD}}^{dd} \ (T, A) \Leftrightarrow \neg P \ \mathit{sat}_{as}^{dd} \ (T, A)$.

'$\Rightarrow$': $\neg P \ \mathit{sat}_{\mathcal{FD}}^{dd} \ (T, A)$ implies the existence of an $\mathcal{FD}$-implementation S, such that $S \parallel_A T$ can deadlock or diverge. By Lemma 2.58, there is also an $S' \in \mathit{impl}_{as}(P)$ such that $S' \parallel_A T$ can deadlock or diverge and thus $\neg P \ \mathit{sat}_{as}^{dd} \ (T, A)$.

'$\Leftarrow$': Follows similarly from Lem. 2.56.1, since each as-implementation is an $\mathcal{FD}$-implementation. □

This immediately implies the following theorem, which shows that, analogously to deadlock testing, deadlock/divergence testing is robust: We can base it on as- or $\mathcal{FD}$-implementations or any notion in between. Using Lemma 2.56, we can also prove that $\mathcal{FD}$-refinement is thorough.

Theorem 2.61. *For all dMTSs, as-deadlock-divergence-refinement and $\mathcal{FD}$-deadlock-divergence-refinement coincide. $\mathcal{FD}$-refinement is thorough.*

According to the general ideas described in the introduction, we should show that $\mathcal{FD}$-refinement is the coarsest precongruence for the operators $\|_A$ preserving deadlock and divergence freedom. For this, we have to generalize the notion of deadlock and divergence freedom for dMTS.

Theorem 2.62. *We say that a dMTS P can deadlock or diverge, if there is some $S \in impl_{as}(P)$ that can. This is equivalent to $(w, \Sigma) \in \mathcal{F}_\mathcal{D}(P)$. With this notion, $\mathcal{FD}$-refinement is the coarsest precongruence for the operators $\|_A$ that preserves deadlock and divergence freedom.*

Proof. The first claim follows from Lemma 2.51.1 and 2.56.2. For the second, let P and P' be arbitrary dMTSs and $A = \Sigma \setminus \{\omega\}$.

Assume P can deadlock or diverge and $P \sqsubseteq_{\mathcal{FD}} P'$. Then $(w, \Sigma) \in \mathcal{F}_\mathcal{D}(P) \subseteq \mathcal{F}_\mathcal{D}(P')$, i.e. P' can deadlock or diverge as well. So, $\mathcal{FD}$-refinement preserves deadlock and divergence freedom.

Now consider some precongruence $\sqsubseteq$ preserving deadlock and divergence freedom; let $P \sqsubseteq P'$. To check $P \sqsubseteq_{\mathcal{FD}} P'$, firstly consider $w \in \mathcal{D}(P)$. There exists some $S \in impl_{as}(P)$ with $w \in \mathcal{D}(S)$ by Lemma 2.56.2 and $S \|_A T_\tau(w)$ can deadlock or diverge by Lemma 2.57.1. Since as-refinement is a precongruence for parallel composition [40], we get $S \|_A T_\tau(w) \in impl_{as}(P \|_A T_\tau(w))$. By the properties of $\sqsubseteq$, we have $P \|_A T_\tau(w) \sqsubseteq P' \|_A T_\tau(w)$ and some $S' \in impl_{as}(P' \|_A T_\tau(w))$ can deadlock or diverge.

Considering $T_\tau(w)$ and A, we see that S' cannot deadlock, since $t \overset{\omega}{\Longrightarrow}$ for any state (p', t) of $P' \|_A T_\tau(w)$. Thus, we further see that some $v \sqsubseteq w$ and $i \geq 0$ with $v\omega^i \in \mathcal{D}(S') \subseteq \mathcal{D}(P' \|_A T_\tau(w))$ (again using 2.56.2). Because of our choice of A, we get that $v \in \mathcal{D}(P')$ since $T_\tau(w)$ cannot diverge; hence, $w \in \mathcal{D}(P')$.

Secondly, consider some $(w, X_0) \in \mathcal{F}_\mathcal{D}(P)$. If $w \in \mathcal{D}(P')$, we are done, so assume $w \notin \mathcal{D}(P')$. Let $X = X_0 \setminus \{a \mid wa \in \mathcal{D}(P')\}$; $(w, X) \in \mathcal{F}_\mathcal{D}(P)$ by $\mathcal{F}$-consistency. There exists some $S \in impl_{as}(P)$ with $(w, X) \in \mathcal{F}_\mathcal{D}(S)$ by Lemma 2.56.2. $S \|_A T(w, X)$ can deadlock or diverge by Lemma 2.57.2a. Since as-refinement is a precongruence for parallel composition, we get $S \|_A T(w, X) \in impl_{as}(P \|_A T(w, X))$. By the properties of $\sqsubseteq$, we have $P \|_A T(w, X) \sqsubseteq P' \|_A T(w, X)$. Thus, with $P \|_A T(w, X)$, also $P' \|_A T(w, X)$ and some $S' \in impl_{as}(P' \|_A T(w, X))$ can deadlock or diverge.

Observe that $P' \|_A T(w, X)$ cannot diverge: Otherwise, since only P' can give rise to a divergence, a divergence state of $P' \|_A T(w, X)$ can be reached without performing ω. This implies that $w \in \mathcal{D}(P')$ or $wa \in \mathcal{D}(P')$ for some $a \in X$, a contradiction. Thus, S' cannot diverge either, due to as-refinement.

Hence, S' can deadlock – and this in a state matching some (p', t_n) of $P' \|_A T(w, X)$ since all other states enforce an ω-transition. We conclude that (p', t_n) is a deadlock and $(w, \Sigma) \in \mathcal{F}_\mathcal{D}(P' \|_A T(w, X))$. Since $\mathcal{D}(P' \|_A$

$T(w,X)) = \emptyset$, Thm. 2.53 now gives $(w,X) \in \mathcal{F}_\mathcal{D}(P')$, and this implies $(w,X_0) \in \mathcal{F}_\mathcal{D}(P')$ with 2.51.3c. □

We will show that our new $\sqsubseteq_{\mathcal{FD}}$ is a precongruence for hiding, like the traditional $\sqsubseteq_{\mathcal{FD}t}$. For this, we have to restrict ourselves slightly to image-finite dMTS.

Definition 2.63. *A dMTS P is* image-finite *if for all states p and all $\alpha \in \Sigma_\tau$ there are only finitely many transitions $p \overset{\alpha}{\dashrightarrow} p'$.*

Definition 2.64. *Given a dMTS P and a set of actions $A \subseteq \Sigma$, the* hiding *of A in P is the dMTS P/A obtained from P by replacing all transition labels $a \in A$ by τ.*

It is known that failure inclusion is not a precongruence with respect to hiding (already on LTS), as shown by the standard example in Fig. 2.17; cf. e.g. [60]. One can see that P_1 and P_2 are equivalent w.r.t. our $\mathcal{F}$-refinement, since both can refuse $\{b\}$ or $\{c\}$ after any number of as. After hiding a, however, $P_1/\{a\}$ can refuse $\{b\}$ or $\{c\}$ initially, whereas $P_2/\{a\}$ cannot refuse any of them.

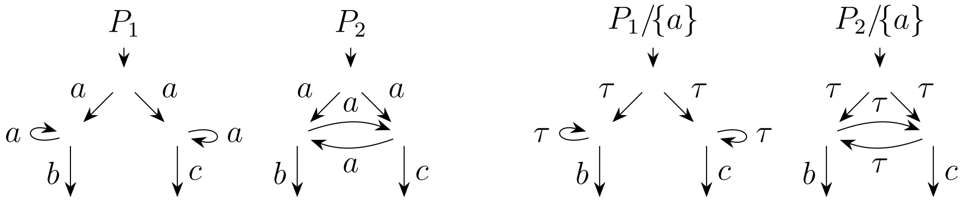

Figure 2.17: Failure semantics is not compositional for hiding

The following lemma exhibits the use of image-finiteness. It has been proven e.g. in [19] for MTS, but holds here as well since it only concerns may-transitions.

Lemma 2.65. *If P is an image-finite dMTS, then for all infinite sequences $\sigma \in \Sigma^\omega$ such that $p_0 \overset{\sigma'}{=\!\!\Rightarrow}$ and $\sigma' \notin \mathcal{D}(P)$ for each $\sigma' \sqsubset \sigma$, we have $p_0 \overset{\sigma}{=\!\!\Rightarrow}$.*

Theorem 2.66. *Let P be an image-finite dMTS and $A \subseteq \Sigma$ be a set of actions.*

1. *$\mathcal{D}(P/A) = cont(\{w|_{\Sigma\setminus A} \mid w \in \mathcal{D}(P)\} \cup \{w|_{\Sigma\setminus A} \mid \exists \sigma \in A^\omega : \forall \sigma' \sqsubset \sigma : (w\sigma', \emptyset) \in \mathcal{F}_\mathcal{D}(P)\})$*

2. $\mathcal{F}_\mathcal{D}(P/A) = \{(w|_{\Sigma\setminus A}, X \cup X') \mid (w, X \cup A) \in \mathcal{F}_\mathcal{D}(P), \forall a \in X' : (w|_{\Sigma\setminus A})a \in \mathcal{D}(P/A)\} \cup {}^\nabla\mathcal{D}(P/A)$.

Proof. 1. This has been proven for MTS in [19] based on $\sqsubseteq_{\mathcal{FD}t}$. The present proof only differs in Case '$\supseteq$'(b). We will refer to $\{w|_{\Sigma\setminus A} \mid w \in \mathcal{D}(P)\}$ and $\{w|_{\Sigma\setminus A} \mid \exists \sigma \in A^\omega : \forall \sigma' \sqsubset \sigma : (w\sigma', \emptyset) \in \mathcal{F}_\mathcal{D}(P)\}$ as the first and second set on the r.h.s. respectively.

'$\subseteq$': Take a prefix minimal $v \in \mathcal{D}(P/A)$, i.e. $p_0 \overset{v}{\Longrightarrow}_{P/A} p$ with p divergent in P/A. Then there is some w with $w|_{\Sigma\setminus A} = v$ and $p_0 \overset{w}{\Longrightarrow}_P p$. There are two possibilities: There is some $u \in A^*$ (possibly $u = \varepsilon$), such that $p \overset{u}{\Longrightarrow}_P p'$ and p' is divergent in P. Then $wu \in \mathcal{D}(P)$, and $v = (wu)|_{\Sigma\setminus A}$ is contained in the first set of the r.h.s. Otherwise, there has to be an infinite sequence $\sigma \in A^\omega$ such that $p \overset{\sigma}{\Longrightarrow}_P$. In this case, v is clearly in the second set of the r.h.s. The inclusion follows since the r.h.s. is closed under continuation.

'$\supseteq$': It suffices to consider elements of the first or second set on the r.h.s., since $\mathcal{D}(P/A)$ is closed under continuation.

(a) Take a prefix minimal element w of $\mathcal{D}(P)$. Then $p_0 \overset{w}{\Longrightarrow}_P p$ for some divergent p. This implies $p_0 \overset{w|_{\Sigma\setminus A}}{\Longrightarrow}_{P/A} p$ and p is still divergent in P/A, i.e. $w|_{\Sigma\setminus A} \in \mathcal{D}(P/A)$. For each continuation wv of w, $(wv)|_{\Sigma\setminus A} = w|_{\Sigma\setminus A} v|_{\Sigma\setminus A}$ is a continuation of $w|_{\Sigma\setminus A}$, hence also in $\mathcal{D}(P/A)$.

(b) Let $w \in \Sigma^*$ and $\sigma \in A^\omega$ be such that, for all finite prefixes σ' of σ, we have $(w\sigma', \emptyset) \in \mathcal{F}_\mathcal{D}(P)$. If some $w\sigma' \in \mathcal{D}(P)$, then we are done by (a). Otherwise, by Lemma 2.51.3b, we have $(w\sigma', \emptyset) \in \mathcal{F}_{st}(P)$ and $p_0 \overset{w\sigma'}{\Longrightarrow}_P$ for each $\sigma' \sqsubset \sigma$ (and $p_0 \overset{w'}{\Longrightarrow}_P$ for each $w' \sqsubseteq w$). By Lemma 2.65 we get that $p_0 \overset{w\sigma}{\Longrightarrow}_P$. This implies $p_0 \overset{(w\sigma)|_{\Sigma\setminus A}}{\Longrightarrow}_{P\setminus A}$, ending in an infinite sequence of τ-transitions. Hence $(w\sigma)|_{\Sigma\setminus A} = w|_{\Sigma\setminus A} \in \mathcal{D}(P/A)$.

2. '$\subseteq$': Take some $(v, X_0) \in \mathcal{F}_\mathcal{D}(P/A)$. If $(v, X_0) \in {}^\nabla\mathcal{D}(P/A)$ we are done since the latter is contained in the r.h.s.; so consider $X_0 = X \cup X'$, $(v, X) \in \mathcal{F}_{st}(P/A)$ and $X' \subseteq \{a \mid va \in \mathcal{D}(P/A)\}$.

We only have to show that $(w, X \cup A) \in \mathcal{F}_\mathcal{D}(P)$ for some w with $w|_{\Sigma\setminus A} = v$. $(v, X) \in \mathcal{F}_{st}(P/A)$ is justified by some p must-stable in P/A with $p_0 \overset{v}{\Longrightarrow}_{P/A} p$ and $p \overset{a}{\not\longrightarrow}_{P/A}$ for all $a \in X$. From this we can deduce that $p_0 \overset{w}{\Longrightarrow}_P p$ for some w with $w|_{\Sigma\setminus A} = v$. Since p is must-stable in P/A, we have $p \overset{a}{\not\longrightarrow}_P$ for all $a \in A$. The same holds for $a \in X \setminus A$, since the a-transitions for these a are the same in P and P/A. Since hiding does not remove any τ-transitions, p is also must-stable in P. Hence, $(w, X \cup A) \in \mathcal{F}_{st}(P) \subseteq \mathcal{F}_\mathcal{D}(P)$.

'$\supseteq$': Since ${}^\nabla\mathcal{D}(P/A)$ is contained in the l.h.s., we consider some $(v, X \cup X')$ with $v = w|_{\Sigma\setminus A}$, $(w, X \cup A) \in \mathcal{F}_\mathcal{D}(P)$ and $X' \subseteq \{a \mid va \in \mathcal{D}(P/A)\}$. We can assume that $va \notin \mathcal{D}(P/A)$ for all $a \in X$, since for a violating a, we can replace X by $X \setminus \{a\}$ and X' by $X' \cup \{a\}$. This leaves $(v, X \cup X')$ unchanged and the other statements remain true. We show $(v, X) \in \mathcal{F_D}(P/A)$, and then $(v, X \cup X') \in \mathcal{F_D}(P/A)$ follows by Lemma 2.51.3c.

$(w, X \cup A) \in \mathcal{F_D}(P)$ implies $w \in \mathcal{D}(P)$ – in which case we are done by 1. – or $p_0 \overset{w}{\Longrightarrow}_P p$ for some must-stable p with $X \cup A = Y \cup Y'$, $p \overset{a}{\not\longrightarrow}_P$ for all $a \in Y$ and $wa \in \mathcal{D}(P)$ for all $a \in Y'$.

Consider some $a \in Y'$. From $wa \in \mathcal{D}(P)$ we deduce by 1. $v(a|_{\Sigma \setminus A}) = (wa)|_{\Sigma \setminus A} \in \mathcal{D}(P/A)$. For $a \in X \setminus A$ we would get $va \in \mathcal{D}(P/A)$ contradicting our choice of X and X'. Thus $a \in A$ and $v \in \mathcal{D}(P/A)$ and we are done. Therefore, we can assume that $Y' = \emptyset$.

From $p_0 \overset{w}{\Longrightarrow}_P p$ we obtain $p_0 \overset{v}{\Longrightarrow}_{P/A} p$. Furthermore, p is must-stable in P/A, since it is must-stable in P and $p \overset{a}{\not\longrightarrow}_P$ for all $a \in A \subseteq Y$, since $p \overset{a}{\not\longrightarrow}_P$ for all $a \in Y$. Lastly, $p \overset{a}{\not\longrightarrow}_{P/A}$ for all $a \in X \subseteq Y$. Hence $(v, X) \in \mathcal{F}_{st}(P/A) \subseteq \mathcal{F_D}(P/A)$. □

Corollary 2.67. *For image-finite dMTSs P_1 and P_2 and a set of actions $A \subseteq \Sigma$, we have $P_1 \sqsubseteq_{\mathcal{FD}} P_2 \implies P_1/A \sqsubseteq_{\mathcal{FD}} P_2/A$.*

Finally, concerning expressiveness, we find that dMTSs are not more expressive than LTS also w.r.t. $\mathcal{FD}$-semantics.

Proposition 2.68. *For a dMTS P, we have $P =_{\mathcal{FD}} S$ for the following LTS S:*

- *$S = P \cup \{\overline{p} \mid p \in P, p$ stable$\}$ and $s_0 = p_0$.*
- *$\longrightarrow_S = \dashrightarrow_S = \dashrightarrow_P \cup \{(p, \tau, \overline{p}) \mid p, \overline{p} \in S\} \cup \{(\overline{p}, a, p') \mid \overline{p} \in S, p \xrightarrow{a}_P P', p' \in P'\}$*

Proof. We have $\mathcal{D}(P) = \mathcal{D}(S)$ since both systems have the same divergent runs on the states and transitions of P and the new states $\overline{p}$ allow for no new traces. The $(w, X) \in \mathcal{F_D}(P)$ justified by some stable p are exactly the $(w, X) \in \mathcal{F_D}(S)$ justified by $\overline{p}$, since they have the same outgoing must-transitions. Furthermore, any $(w, X) \in \mathcal{F_D}(S)$ justified by some stable p, is also justified by the corresponding $\overline{p}$, since $\overline{p}$ can refuse at least all actions p can. □

2.4.3 Action-Must-Testing

We will now show that $\mathcal{FD}$-refinement can also be justified by a new version of must-testing. In the traditional variant, a test environment has success-states, marked by outgoing ω-transitions for the special action ω. A test is satisfied if some success-state is reached on each maximal run; we will call this *state-must-testing*. Applying it to MTS, again by testing all implementations, yields the $\mathcal{FD}t$-semantics as we have shown in [19]. Presumably, applying state-must-testing to dMTS will yield an analogous semantics. Here, we will examine *action-must-testing*, where ω must be *performed* on all maximal runs.

For the remainder of this section, a test is an implementation T. As a *general assumption*, the special action ω is only allowed in tests, and A denotes the synchronization set $\Sigma \setminus \{\omega\}$.

Definition 2.69. *A dMTS P* as-action-must-satisfies *T if, for each $S \in impl_{as}(P)$, every* maximal run *(infinite or ending in a stable deadlock) of $S \parallel_A T$ contains the* success action *ω. Then, we write $P \ sat^m_{as} \ T$. A dMTS P* as-action-must-refines *P', $P \leq^m_{as} P'$, if for all tests T : $P' \ sat^m_{as} \ T \implies P \ sat^m_{as} \ T$.*

*We define $\mathcal{FD}$-*action-must-satisfaction *$sat^m_{\mathcal{FD}}$ and $\mathcal{FD}$-*action-must-refinement *$\leq^m_{\mathcal{FD}}$ analogously with $S \in impl_{\mathcal{FD}}(P)$.*

We will show that action-must-refinement is characterized by $\mathcal{FD}$-semantics and thus equivalent to deadlock-divergence-refinement. For this, we can use the lemmata from the deadlock-divergence-testing section. We mainly need analogous results for Lemma 2.57 and Lemma 2.58.

Lemma 2.70. *Let S be an LTS. For $T(w, X)$ and $T_\tau(w)$ refer to Fig. 2.4 and 2.16; let $C = S \parallel_A T(w, X)$.*

1. *$w \in \mathcal{D}(S)$ if and only if $S \parallel_A T_\tau(w)$ has a maximal run that does not contain ω.*
2. *(a) If $(w, X) \in \mathcal{F}_{\mathcal{D}}(S)$, then C has a maximal run that does not contain ω.*

   (b) If C has a maximal run that does not contain ω, then $w \in \mathcal{D}(S)$, $(w, X) \in \mathcal{F}_{st}(S)$ or $\exists a \in X : wa \in \mathcal{D}(S)$.

Proof. 1. First consider $w \in \mathcal{D}(S)$; hence, for some $i \leq n$ and $v = w_1 \dots w_i$, $s_0 \overset{v}{\Longrightarrow} s$ for some divergent s. One can easily see that $S \parallel_A T_\tau(w)$ has the maximal run $(s_0, t_0) \overset{v}{\Longrightarrow} (s, t_i) \overset{\tau^\omega}{\Longrightarrow}$, which does not contain ω.

Second, $S \parallel_A T_\tau(w)$ has a maximal run that does not contain ω. By construction of $T_\tau(w)$ this can only be because of a divergence; a deadlock or an infinite visible run are impossible. Therefore, $w \in D(S)$ follows from Lem. 2.57.1.

2a. By Lemma 2.57.2a, C can deadlock in some state (s, t) or diverge. In the latter case, it has to be S that performs infinitely many τs after a prefix of w or some wa with $a \in X \setminus \{\omega\}$, and some respective infinite run of C does not contain ω.

If C does not diverge, we must have $t = t_n$ by construction of $T(w, X)$ and ω has not been performed before. Since C is divergence free, we can assume s to be stable; thus, the run reaching (s, t) is maximal.

2b. The maximal run has at most $n + 1$ visible actions, hence C can deadlock or diverge; we are done by Lemma 2.57.2b. □

Lemma 2.71. *Let S be an $\mathcal{FD}$-implementation of an image-finite dMTS P and T a test. If $S \parallel_A T$ has a maximal run that does not contain ω, then there is some as-implementation S' of P such that $S' \parallel_A T$ has a maximal run that does not contain ω.*

Proof. We distinguish the following three possibilities: The run ends in an infinite τ-sequence, it reaches a stable deadlock, or it performs some infinite $w \in A^\omega$.

(α) In case of an infinite τ-sequence, there is some $w \in \mathcal{D}(S \parallel_A T)$ – let it be prefix-minimal w.l.o.g. By Thm. 2.53, we get that $w \in \mathcal{D}(S) \parallel_A L_\mathcal{D}(T) \cup L_\mathcal{D}(S) \parallel_A \mathcal{D}(T)$. If w is in the first subset, we have some $u \in \mathcal{D}(S) \subseteq \mathcal{D}(P)$ and $v \in L_\mathcal{D}(T)$ such that $w \in u \parallel_A v$ (and actually $u = v = w$). By Lemma 2.56.2 there is some as-implementation S' of P with $u \in \mathcal{D}(S')$. Thus we get $w \in \mathcal{D}(S') \parallel_A L_\mathcal{D}(T)$, i.e. $S' \parallel_A T$ can perform $w\tau^\omega$.

If w is in the second subset, then (by $\mathcal{F}$-consistency) we have $(u, \emptyset) \in \mathcal{F}_\mathcal{D}(S) \subseteq \mathcal{F}_\mathcal{D}(P)$ and $v \in \mathcal{D}(T)$ such that $w \in u \parallel_A v$. By Lemma 2.56.2, there is $S' \in \mathit{impl}_{as}(P)$ with $(u, \emptyset) \in \mathcal{F}_\mathcal{D}(S')$, i.e. $u \in L_\mathcal{D}(S')$. Thus we get $w \in L_\mathcal{D}(S') \parallel_A \mathcal{D}(T)$, i.e. $S' \parallel_A T$ can again perform $w\tau^\omega$.

(β) In case of a stable deadlock, we have $(s_0, t_0) \overset{w}{\Longrightarrow} (s, t)$ and $(s, t) \overset{\alpha}{\not\longrightarrow}$ for all $\alpha \in \Sigma_\tau$. Thus we have runs $s_0 \overset{w}{\Longrightarrow} s$ and $t_0 \overset{w}{\Longrightarrow} t$ with s and t stable. Choose $Z = \{a \in \Sigma \mid t \overset{a}{\not\longrightarrow}\}$. Due to s, there is Y_0 such that $(w, Y_0) \in \mathcal{F}_\mathcal{D}(S) \subseteq \mathcal{F}_\mathcal{D}(P)$, $Y_0 \cup Z = \Sigma$ and $Y_0 \cap Z = \{\omega\}$ w.l.o.g. (otherwise set $Y_0 := (Y_0 \setminus Z) \cup \{\omega\}$). By Lemma 2.56.2, there is $S' \in \mathit{impl}_{as}(P)$ with $(w, Y_0) \in \mathcal{F}_\mathcal{D}(S')$. If for some $v \sqsubseteq w$ we have $s'_0 \overset{v}{\Longrightarrow} s'$ with s' divergent, we are done as in (α), since $v \in L_\mathcal{D}(T)$. So assume $(w, Y_0) \in \mathcal{F}_\mathcal{D}(S')$ due to some stable s' (i.e. $s'_0 \overset{w}{\Longrightarrow} s'$). Let $Y \cup Y' = Y_0$ such that $s' \overset{a}{\not\longrightarrow}$ for all $a \in Y$ and $wa \in \mathcal{D}(S')$ for all $a \in Y'$. Consider $a \in Y'$ (thus $a \neq \omega$); then, by choice of Y_0, there is an infinite run $(s'_0, t_0) \overset{wa\tau^\omega}{\Longrightarrow}$ not containing ω. Only the case $Y' = \emptyset$ remains. Now $(s'_0, t_0) \overset{w}{\Longrightarrow} (s', t)$, (s', t) is stable, and $s' \overset{\omega}{\not\longrightarrow}$, $t \overset{\omega}{\not\longrightarrow}$ and each other action is blocked by s' or t.

(γ) In the final case, $(s_0, t_0) \overset{w}{\Longrightarrow}$ with $w \in A^\omega$. This is the only case where image-finiteness is needed. If $v \in \mathcal{D}(S \parallel_A T)$ for some $v \sqsubset w$, we are done by (α), so assume otherwise. Since ω never occurs in w, $s_0 \overset{w}{\Longrightarrow}$

and $t_0 \stackrel{w}{\Longrightarrow}$. Therefore, for every prefix w' of w, we have $(w', \emptyset) \in \mathcal{F}_D(S) \subseteq \mathcal{F}_D(P)$. Let S' be the as-implementation of P with the same state set and $\longrightarrow_{S'} = \dashrightarrow_{S'} = \dashrightarrow_P$, which is also image-finite; then each $(w', \emptyset) \in \mathcal{F}_D(S')$. Should any such $(w', \emptyset)$ be in ${}^{\nabla}\mathcal{D}(S')$, we are done as in ($\alpha$) (first subcase). Otherwise, we get that $s'_0 \stackrel{w}{\Longrightarrow}$ by Lemma 2.65. Combining this with the run $t_0 \stackrel{w}{\Longrightarrow}$ from above, we get an infinite run $(s'_0, t_0) \stackrel{w}{\Longrightarrow}$ that does not contain ω. □

Theorem 2.72. *For a dMTS P and an image-finite dMTS P', $P \leq^m_{as} P'$ and $P \sqsubseteq_{\mathcal{FD}} P'$ are equivalent.*

Proof. '$P \leq^m_{as} P' \Rightarrow P \sqsubseteq_{\mathcal{FD}} P'$': Consider a prefix-minimal $w \in \mathcal{D}(P)$. By Lemma 2.56.2, there is some as-implementation S of P with $w \in \mathcal{D}(S)$. Due to Lemma 2.70.1, $S \parallel_A T_\tau(w)$ has a maximal run not containing ω. Thus P fails the test, and, by our premise, so does P'. This implies the existence of some $S' \in \mathit{impl}_{as}(P')$ such that $S' \parallel_A T_\tau(w)$ has a maximal run not containing ω. Due to Lemma 2.70.1 $w \in \mathcal{D}(S')$ and, finally, $w \in \mathcal{D}(P')$ by Lemma 2.56.2.

Also the case '$(w, X_0) \in \mathcal{F}_{\mathcal{D}}(P)$' works as in the proof of Thm. 2.59, using 2.70 in place of 2.56.

'$P \leq^m_{as} P' \Leftarrow P \sqsubseteq_{\mathcal{FD}} P'$': We prove the as-action-must-refinement by contraposition and assume $\neg P \;\; \mathit{sat}^m_{as} \;\; T$ for some test T. Thus, there must be an as-implementation S of P where $S \parallel_A T$ has a maximal run not containing ω. By Lemma 2.56.1 S is an $\mathcal{FD}$-implementation of P and, by transitivity, of P'. Therefore, by Lemma 2.71, there is some $S' \in \mathit{impl}_{as}(P')$ such that $S' \parallel_A T$ has a maximal run not containing ω, implying $\neg P' \;\; \mathit{sat}^m_{as} \;\; T$. This is the only place where the image-finiteness of P' is needed. □

Finally, we show that action-must-testing based on as-implementations coincides with that based on $\mathcal{FD}$-implementations.

Proposition 2.73. *For each test T and image-finite dMTS P, we have $P \;\; \mathit{sat}^m_{\mathcal{FD}} \;\; T \Leftrightarrow P \;\; \mathit{sat}^m_{as} \;\; T$. For image-finite dMTSs P and P', $P \leq^m_{as} P'$ and $P \leq^m_{\mathcal{FD}} P'$ coincide.*

Proof. The coincidence of $\leq^m_{as}$ and $\leq^m_{\mathcal{FD}}$ immediately follows from the first claim. We prove the latter by showing $\neg P \;\; \mathit{sat}^m_{\mathcal{FD}} \;\; T \Leftrightarrow \neg P \;\; \mathit{sat}^m_{as} \;\; T$.

'$\Rightarrow$': $\neg P \;\; \mathit{sat}^m_{\mathcal{FD}} \;\; T$ implies the existence of an $\mathcal{FD}$-implementation S, such that $S \parallel_A T$ has a maximal run not containing ω. By Lemma 2.71, there is also an $S' \in \mathit{impl}_{as}(P)$ such that $S' \parallel_A T$ has a maximal run not containing ω and thus $\neg P \;\; \mathit{sat}^m_{as} \;\; (T, A)$.

'$\Leftarrow$': Follows, since each as-implementation is an $\mathcal{FD}$-implementation. □

2.5 Overview

In Fig. 2.18, we present an overview regarding strong and weak as-simulation and $\mathcal{F}$-refinement and its variants, as well as the thorough refinements based on them. The arrows show the significant implications. If an arrow is neither shown nor implied by transitivity, the implication does not hold. The examinations of $\sqsubseteq_{mL\mathcal{F}}$ and $\sqsubseteq_{\mathcal{FD}t}$ have formally only been examined for MTS. Disregarding them, the results hold on dMTS in general.

$$\begin{tikzcd}
\text{(strong) as} \arrow[r] \arrow[d] & \text{th. (strong) as} \arrow[r] \arrow[dr] & \mathcal{FD}t \Leftrightarrow \text{th. } \mathcal{FD}t \arrow[r] & \mathcal{FD} \Leftrightarrow \text{th. } \mathcal{FD} \\
\text{weak as} \arrow[r] & \text{th. weak as} \arrow[r] & \sqsubseteq_{mL\mathcal{F}} \Leftrightarrow \text{th. } \sqsubseteq_{mL\mathcal{F}} \arrow[r] & \mathcal{F} \Leftrightarrow \text{th. } \mathcal{F}
\end{tikzcd}$$

Figure 2.18: The implications between the refinement notions (th. = thorough)

The strictness of the implication between strong and thorough strong as-refinement had already been shown e.g. in [43] with the MTSs S and T in Fig. 2.19, where $P \sqsubseteq_{as} Q$ fails even though $impl_{as}(P) \subseteq impl_{as}(Q)$.

The two most remarkable results in the next proposition are the following (MTS from Fig. 2.19): We observed that U is a weak as-implementation of P, but not of Q, which proves that thorough as-refinement does not imply the thorough weak one. Furthermore, it was presumably believed that P and Q also show that thorough weak as-refinement does not imply weak as-refinement, since the two MTSs have no τ-transitions. Our U demonstrates the flaw in this reasoning: its τ-transition makes a difference. Rolf Hennicker [private communication] found a suitable modification Q', also shown in Fig. 2.19, to really refute the implication: P is a thorough weak as-refinement of Q', but not a weak as-refinement.

Proposition 2.74. *The implications shown in Fig. 2.18 and their reflexive-transitive closure are the only implications between these refinements.*

Proof. It is clear from the definition that as-refinement implies weak as-refinement. Also, all refinement relations imply their thorough counterparts,

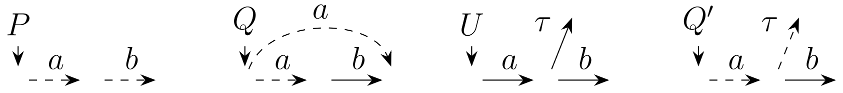

Figure 2.19: MTSs distinguishing the thorough refinements

analogously to '⇒' in the proof of Thm. 2.14. By the same theorem and Prop. 2.44, we know that $\mathcal{F}$- and $mL\mathcal{F}$-refinement are thorough. That $\mathcal{FD}t$-refinement is also thorough can be seen by a proof analogous to the one of Thm. 2.14, using an analogue of Lemma 2.13 (cf. [19] for details). We have shown it for $\mathcal{FD}$-refinement in Thm. 2.61. For the following, we ignore these four thorough variants.

In order to prove that the two thorough as-refinements imply $mL\mathcal{F}$-refinement, we show that they imply both, may- and $\mathcal{F}$-refinement: For $\mathcal{F}$-refinement, let us consider MTSs P and Q where P is a thorough strong as-refinement of Q. Applying Lem. 2.5 twice, we get $\mathcal{F}(P) = \bigcup_{P \in impl_{as}(P)} \mathcal{F}(S)$ $\subseteq \bigcup_{S \in impl_{as}(Q)} \mathcal{F}(S) = \mathcal{F}(Q)$. Lemma 2.13 and an analogous argument show the implication for the weak case. For may-refinement we argue similarly to the proof of Prop. 2.44: we consider the strong and weak as-implementation S of P obtained by removing all may-transitions that are not must-transitions. On the one hand we have $mustL(P) = mustL(S)$; on the other hand, by repeated application of Def. 2.2.1, we get $mustL(S) \supseteq mustL(Q)$.

As we have just seen, thorough strong as-refinement implies $\mathcal{F}$-refinement, and it refines $\mathcal{D}$-inclusion with an analogous argument using an analogue of Lemma 2.5 (again cf. [19]). In Prop. 2.52 we have proven $P \sqsubseteq_{\mathcal{FD}t} P' \implies P \sqsubseteq_{\mathcal{FD}} P'$. Obviously $mL\mathcal{F}$-refinement implies $\mathcal{F}$-refinement. Thus we have all implications.

From Prop. 2.40 we already know that $\mathcal{F}$-refinement does not imply may-refinement and, thus, it does not imply $mL\mathcal{F}$-refinement either. From Prop. 2.52 we also know that $\sqsubseteq_{\mathcal{FD}}$ does not imply $\sqsubseteq_{\mathcal{FD}t}$. Furthermore, we know from the examples in Figs. 2.17 and 2.20 that $\mathcal{FD}$-refinement and $\mathcal{F}$-refinement are incomparable. Therefore $\sqsubseteq_{\mathcal{F}}$ implies none of the other refinements and, in turn, is not implied by $\sqsubseteq_{\mathcal{F}D}$; we are done with it.

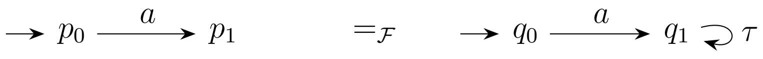

Figure 2.20: Failure equivalent LTS despite divergence

From $\sqsubseteq_{\mathcal{F}D}$ not implying $\sqsubseteq_{\mathcal{F}}$, we can deduce that it does not imply any of the other refinements. Since in Fig. 2.20 $Q \sqsubseteq_{w\text{-}as} P$, but not $Q \sqsubseteq_{\mathcal{F}D} P$, it follows that:

(a) $\sqsubseteq_{w\text{-}as}$ does not imply $\sqsubseteq_{\mathcal{F}D}$.

Hence, there is no arrow from weak as, th. weak as or $\sqsubseteq_{mL\mathcal{F}}$ to $\mathcal{FD}$, and we can remove $\mathcal{FD}$ from consideration. (a) also implies that $\sqsubseteq_{mL\mathcal{F}}$ does not imply thorough as-refinement. We proved before stating the proposition that:

(b) thorough as-refinement does not imply thorough weak as-refinement.

This implies that $\sqsubseteq_{mL\mathcal{F}}$ does not imply thorough weak as-refinement and that thorough as-refinement implies neither strong nor weak as-refinement.

Observe that from (a) we also know that neither weak as-refinement, nor thorough weak as-refinement imply thorough as-refinement or as-refinement. Finally, we have also shown before the proposition, that thorough weak as-refinement does not imply weak as-refinement. □

2.6 Conclusion

In this section, we presented a new refinement relation for MTS arising from deadlock testing and demonstrated its generality and its advantages with several results: Our $\mathcal{F}$-refinement arises in numerous variants of our testing scenario and, in contrast to as-refinement, it supports itself in this scenario and is thorough; it also is a precongruence for parallel composition.

Generalizing our testing approach to dMTS, we again characterized the preorder by a standard failure semantics; in particular, this allows to decide the preorder. Unfortunately, it failed to be a precongruence for parallel composition.

Returning to MTS, we also presented a conjunction operator and outlined how one can be achieved for dMTS, as well.

To make $\mathcal{F}$-refinement more discriminating, we combined it with *mustL*-inclusion, arising from may-testing. But, we have shown that no conjunction operator on MTS exists for the resulting $mL\mathcal{F}$-refinement or any refinement notions between it and as-refinement.

To achieve a precongruence wrt. hiding and to solve the problems with $\mathcal{F}$-refinement on dMTS we studied the new deadlock/divergence testing and characterized it with a new failures-divergence refinement. This is coarser and, hence, better than the standard variant. Our refinement is robust and a precongruence for parallel composition and (for image-finite systems) also for hiding. Furthermore, we have shown that this refinement also arises from action-must-testing, a variant of must-testing, where a success-action has to be executed, not only be executable.

Finally we gave an overview of the refinement relations we considered throughout the section.

These testing approaches can be generalized to DMTS, where disjunctive transitions can have different labels [44]. DMTS come with an alternating-simulation-style refinement, but it is a strong simulation, where no additional τ-transitions can be used while matching transitions. No weak simulation exists. A somewhat unconventional definition of parallel composition for

τ-less DMTS can be found in [6]. It involves translating the DMTSs into Nondeterministic Acceptance Automata (NAA), constructing their parallel composition and translating the result back into a DMTS. The key step corresponds to transforming a disjunctive into a conjunctive normal form, where the latter corresponds to disjunctive must-transitions.

However, since this is a conservative extension of the dMTS setting, the refinement resulting from deadlock testing cannot be a precongruence for parallel composition. We see little point in pursuing this line of thought. Nevertheless, future work might include examining must-testing or deadlock/divergence-testing for DMTS, where we would expect analogous results.

Chapter 3

Interface Automata – Error Handling and Pruning

3.1 Introduction

In this chapter we will examine the fundamentals of interface automata as introduced by de Alfaro and Henzinger e.g. in [27]. They are an abstract description of the communication behaviour of a system or component in terms of input and output actions. Based on this behavioural type, one can study whether two systems are compatible if put in parallel, and one can define a refinement for specifications. Compatibility, it this case, is defined as some notion or error-freedom. As mentioned in Chapter 1, it is essential for such a setting that the refinement relation is a precongruence for parallel composition; in particular, if we refine two compatible specifications, it must be guaranteed that the refined specifications are compatible again.

A basic intuition here is that outputs are under the control of the respective system: if one component in a composition provides an output for another, the latter must synchronize by performing the same action as input; if this is not possible, the whole system might malfunction – such a catastrophic error state has to be avoided. In contrast to the I/O-automata of [54], interface automata are not input enabled. Instead, a missing input in a state corresponds to the requirement that an environment must not send this input to this state.

There are two essential design decisions in the approach of [27] that we will scrutinize in this chapter. First, the approach is optimistic: an error state is not a problem, if it cannot be reached in a helpful environment, i.e. an environment that prevents the system from running into an error. This is reflected in the details of parallel composition, where from a standard product automaton all states are removed that can reach an error state just by locally controlled, i.e. output and internal, actions (often called *pruning*). Although this definition has some intuitive justification, its details appear somewhat arbitrary. This is also the case for the second decision to take some alternating simulation as refinement relation. Actually, the same authors used a slightly different relation for a slightly larger class of automata in the earlier [25]; no real argument is given for the change.

Here, we will work out to what degree these design decisions can be *justified* from some more basic and, hopefully, more agreeable ideas. We model components as labelled transition systems (LTSs) with disjoint input and output actions and an internal action, quite like the interface automata of [27]. So as not to exclude any possibilities prematurely, our LTS have explicit error states. For these Error-IO-Transition Systems (EIO), we consider a standard parallel composition where, additionally, error states occur as described above; a composed system also reaches an error state if one of the components reaches one. It should be mentioned, that in contrast to the previous chapter, we use parallel composition with immediate hiding, as is usual for interface automata. The setting has been examined for parallel composition without hiding by Schlosser in the bachelor's thesis [61].

An indisputable requirement for a refinement relation is that an error-free specification should only be refined by an error-free system. This can be understood as a basic refinement relation, which is parametric in the exact meaning of error-free: in the optimistic view, error-free means that no error state can be reached by locally controlled actions only; in the pessimistic view (cf. e.g. [4]), a system is error-free only if no error state is reachable at all.

As mentioned before, for modular reasoning, which is at the heart of the approach under study, the refinement relation $\sqsubseteq$ must be a precongruence: if a component of a parallel composition is replaced by a refinement, the composition itself gets refined, i.e. $P_1 \sqsubseteq P_2$ implies $P_1 \mid Q \sqsubseteq P_2 \mid Q$. Since the basic relations fail to be precongruences in each case, we will characterize (or at least approximate) the respective coarsest precongruence for parallel composition that is contained in the basic relation. Such a *fully abstract* precongruence is optimal for preserving error-freeness, since it does not distinguish components unnecessarily.

This approach is somewhat similar to the testing approach defined in the previous chapter, but not the same. The refinement relations in the previous chapter were defined by the interaction of their implementations with test-environments; a test was passed if no undesirable state was reachable and a refinement had to pass all tests of the specification. The result was not always a precongruence. Here, we start with a very coarse relation that only requires preservation of error-freedom – if the specification has no reachable undesirable state then neither may the refinement. Then we find the coarsest precongruence in this relation. This, of course, guarantees that the refinement relation is *compositional*, i.e. a precongruence; Nevertheless, we will have to examine how exactly the refined automata behaves in parallel with environments, in particular when composed with more than one system.

In the optimistic case, the precongruence can be characterized as (component-wise) inclusion for a pair of trace sets; the definition of one of these uses pruning on traces. With this characterization we can prove that, essentially, each EIO is equivalent w.r.t. the precongruence to one without error states, where the latter can be obtained by pruning the former almost as in [27]. Thus, we can work with EIO without error states, i.e. with interface automata and (almost) with the parallel composition of [27], but our pruning is *proven* to be correct.

While this justifies the first design decision in [27], our precongruence shows that alternating simulation is unnecessarily strict. This is not really new. A setting with input and outputs where unexpected inputs lead to errors has been studied long before [27] for speed-independent (thus asynchronous) circuits by Dill in [32]. The difference is that Dill does not start from an operational model as we do (in particular, there is no parallel composition for LTS), but on a semantic level with pairs of trace sets; he requires these pairs to be input enabled. On this semantic level, he also uses pruning; a normalized form of his pairs coincides with our pairs. Essentially, the full abstraction result can also be found in [22], though for a slightly different parallel composition and only for a congruence. Since that paper starts from a declarative approach, our presentation and proofs are more direct, and they prepare the reader for the succeeding sections.

In [22], EIO (called Logic IOLTS there) are seen as an alternative framework to interface automata, and an error state is actually added to normalize an EIO. In this chapter, we see error states only as a tool to study interface automata and would prefer to remove them in the end[1]; with this view, we discovered a subtle point about pruning. Interface automata in [27] are deterministic w.r.t. input actions. Since we do not require this here, our pruning is a bit *different* from the one in [27]. In fact, the interface automata in [25] are also not input deterministic, but pruning used there is *the same* as the one in [27]. As a consequence, Theorem 1 of [25] claiming associativity for

[1] But they will prove invaluable for MIA in the next chapter. The universal state u there is essentially one state representing all error states. It cannot be removed and is necessary for associativity of parallel composition.

parallel composition is wrong; in our setting, it is easily proven.

It might seem that we have actually prescribed pruning in our optimistic approach since we consider only locally reachable errors as relevant and pruning removes exactly those states that can reach an error locally. To fortify the justification of pruning, we briefly describe the results for two other approaches. For a more detailed examination, we refer to [18].

The first is the 'hyper-optimistic' approach, where only internally reachable errors are relevant. This *more generous* notion of error-free leads to a slightly *stricter* precongruence. We present a characterization that is again based on pruning; the new idea is to extend traces with a set of outputs removed during pruning. This is an interesting precongruence but, compared to our first one, it looks unnecessarily complicated.

The other approach is the pessimistic approach, where every reachable error is relevant, as advocated e.g. in [4]. For this case, we only present an approximation of the fully abstract precongruence: we describe a precongruence contained in the respective basic relation, which is based on three trace sets and again employs pruning. We sketch how one might get the fully abstract precongruence, but this will be technically so involved as to make it unattractive. Even without a characterization, we can show that the fully abstract precongruence is again *stricter* than the optimistic one, although the notion of being error-free is *less generous.*

The next section will explain some basic notions and definitions. We examine the optimistic approach in Section 3.3 including a comparison to IA. In Section 3.4 we summarize the results of the hyper-optimistic and pessimistic approaches. Finally, in Section 3.5, we conclude with a comparison and give arguments why we will prefer the optimistic variant in the future.

3.2 Definitions and Notation

First we define our scenario. In interface automata, internal actions have names. For two automata to be composable, the internal action names of one must be disjoint from all action names of the other; hence, standard α-conversion for the names of internal actions is not fully supported.[2] To improve this, our EIOs have just one internal, unobservable action τ. Furthermore, they have as additional component a set of error states; such states can be created in a parallel composition.

[2] Strictly speaking, α-conversion in a refinement step is allowed in [27], since there is no requirement regarding the alphabets of internal actions. But this is certainly an oversight since, as an effect, refinement does not preserve compatibility, because it does not even preserve composability; cf. Proposition 3.13.

Definition 3.1 (Error-IO-Transition-System)**.** *An* Error-IO-Transition-System (EIO) *is defined as a tuple* $(P, I, O, \longrightarrow, p_0, E)$*, where*

- P *– a set of states*
- I, O *– disjoint sets of (visible) input and output actions*
- $\longrightarrow \subseteq P \times (I \cup O \cup \{\tau\}) \times P$ *– a transition relation*
- $p_0 \in P$ *– an initial state*
- $E \subseteq P$ *– a set of* error *states*

The actions *of* P *are* $\mathcal{A} := I \cup O$*, and its signature is* $Sig(P) = (I, O)$*. We call* P closed*, if* $\mathcal{A} = \emptyset$*.*

In a parallel composition, all common actions are synchronized and then immediately hidden. Two EIOs can only be composed, if their input and output actions fit together, i.e. the EIOs have neither common inputs nor common outputs. A state of the composition is an error state if one component is in an error state (*inherited* error) or if one component sends an output to the other one, which is not ready to receive it (*new* error).

Definition 3.2 (Parallel Composition)**.** *Two EIOs* P_1, P_2 *are* composable *if* $I_1 \cap I_2 = \emptyset = O_1 \cap O_2$*. The* parallel composition without hiding *is defined for two composable EIOs as* $P_1 \parallel P_2 = (P_1 \times P_2, I, O, \longrightarrow, p_0, E)$*, where*

- $I = (I_1 \backslash O_2) \cup (I_2 \backslash O_1)$
- $O = O_1 \cup O_2$
- $p_0 = (p_{01}, p_{02})$

Furthermore, with $Synch(P_1, P_2) = (I_1 \cap O_2) \cup (I_2 \cap O_1)$ *being the set of synchronized actions, we define*

$$\begin{aligned}
&\longrightarrow = \\
&\{\big((p_1,p_2),\alpha,(p_1',p_2)\big) \mid (p_1,\alpha,p_1') \in \longrightarrow_1, \alpha \in (\mathcal{A}_1 \cup \{\tau\}) \backslash Synch(P_1,P_2)\} \cup \\
&\{\big((p_1,p_2),\alpha,(p_1,p_2')\big) \mid (p_2,\alpha,p_2') \in \longrightarrow_2, \alpha \in (\mathcal{A}_2 \cup \{\tau\}) \backslash Synch(P_1,P_2)\} \cup \\
&\{\big((p_1,p_2),\alpha,(p_1',p_2')\big) \mid (p_1,\alpha,p_1') {\in} \longrightarrow_1, (p_2,\alpha,p_2') {\in} \longrightarrow_2, \alpha {\in} Synch(P_1,P_2)\}
\end{aligned}$$

$$\begin{aligned}
&E = \\
&(P_1 \times E_2) \cup (E_1 \times P_2) && \text{‘inherited errors’} \\
&\cup \{(p_1,p_2) \mid \exists a \in O_1 \cap I_2 : p_1 \xrightarrow{a} \wedge p_2 \stackrel{a}{\not\longrightarrow}\} && \text{‘new errors’} \\
&\cup \{(p_1,p_2) \mid \exists a \in I_1 \cap O_2 : p_1 \stackrel{a}{\not\longrightarrow} \wedge p_2 \xrightarrow{a}\}
\end{aligned}$$

The parallel composition (with hiding) $P_1 \mid P_2$ *differs only in the definition of its outputs and its transition function. We define* $O = (O_1 \backslash I_2) \cup$

$(O_2 \backslash I_1)$ *and* $\longrightarrow$ *as above, but with the third set consisting of transitions* $\big((p_1, p_2), \tau, (p_1', p_2')\big)$.

We introduce P_{12} *as shorthand for* $P_1 \,|\, P_2$ *and similarly for its components and semantics. We call an EIO* P *a* partner *of an EIO* P' *if their parallel composition is closed, i.e. if they have dual signatures* $Sig(P) = (I, O)$ *and* $Sig(P') = (O, I)$.

For our results and proofs, we also define $|$ and $\|$ as parallel composition on traces with and without hiding respectively. This corresponds to Def. 1.3 and extends it to parallel composition with hiding.

Definition 3.3 (Parallel Composition on Traces). *Given two composable EIOs* P_1, P_2, $w_1 \in \mathcal{A}_1, w_2 \in \mathcal{A}_2, W_1 \subseteq \mathcal{A}_1^*$ *and* $W_2 \subseteq \mathcal{A}_2^*$, *we define*

- $w_1 \parallel w_2 = \{w \in (\mathcal{A}_1 \cup \mathcal{A}_2)^* \mid \ w|_{\mathcal{A}_1} = w_1 \wedge w|_{\mathcal{A}_2} = w_2\}$
- $w_1 \,|\, w_2 = \{w|_{\mathcal{A}_{12}} \ \mid \ w \in w_1 \parallel w_2\}$
- $W_1 \parallel W_2 = \bigcup\{w_1 \parallel w_2 \ \mid \ w_1 \in W_1 \wedge w_2 \in W_2\}$
- $W_1 \,|\, W_2 = \bigcup\{w_1 \,|\, w_2 \ \mid \ w_1 \in W_1 \wedge w_2 \in W_2\}$

For a short example consider Fig. 3.1. Here and in other figures, we will mark an error state by a box. We have two systems P_1 and P_2 and their parallel composition (with hiding). P_1 represents a simple user interface. It can be given a message (*msg*), which it passes on to a *buffer.* This buffering can be *cancel*led, but if the buffering has already taken place, or no message was given, we encounter an error. However, the system can recover from the error, if it is given the *recover* command. P_2 is a simple buffer with capacity two; it can *send* the message on.

In their parallel composition the error in P_1 (state p_{1e}) is inherited and appears in states $(p_{1e}, p_{20}), (p_{1e}, p_{21})$ and (p_{1e}, p_{22}). A new error is created in (p_{11}, p_{22}): here p_{11} could perform the output *buffer*, but p_{22} is not ready to receive it. In the parallel composition of [27] such an error state would be removed as part of the composition, together with all its in- and outgoing edges. In contrast, we leave it for later examination. The error states can have outgoing edges, thus a system could recover from an error. However, we consider an error to be catastrophic; hence, such a recovery will be useless in the end.

Note further that an input without corresponding output does not produce an error. This is the case, for example, in the initial state (p_{10}, p_{20}). The component state p_{20} is ready to receive a *buffer* signal, but since p_{10} does not send the signal, this behaviour simply disappears.

We will base our semantics on traces that can lead to error states. In this context, we will use a pruning function, which removes all output actions from the end of a trace. We also define a function for arbitrary continuation of traces; generalizing this to trace sets, gives the *continuation* or *suffix closure*.

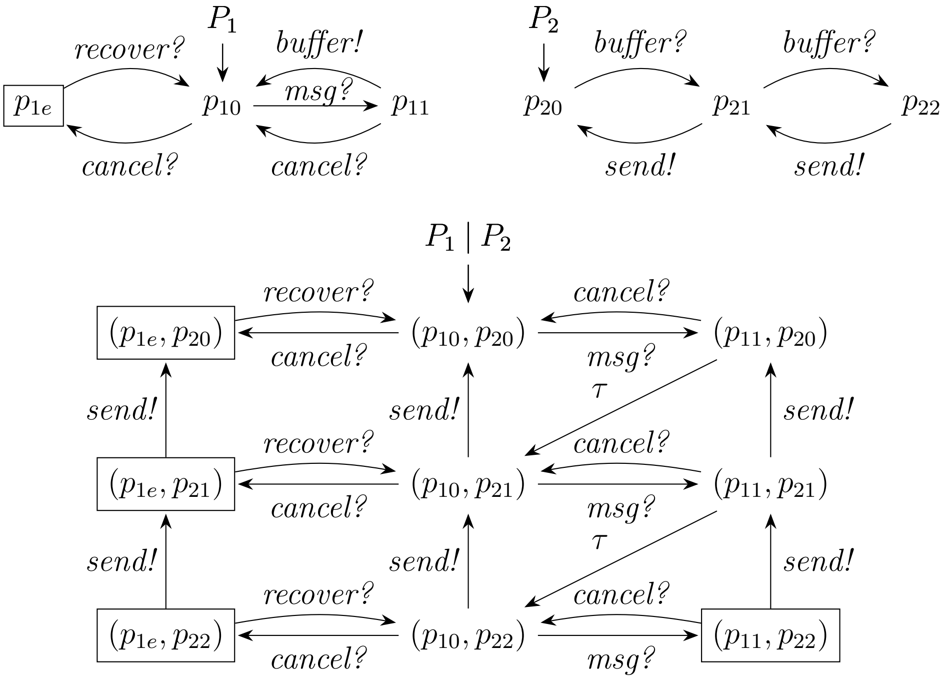

Figure 3.1: An example for parallel composition

Definition 3.4 (Pruning and Continuation Functions)**.** *For an EIO P, we define*

- *prune* : $\mathcal{A}^* \to \mathcal{A}^*$, $w \mapsto u$, *where* $w = uv$, $u = \varepsilon \vee u \in \mathcal{A}^* \cdot I$ *and* $v \in O^*$
- *cont* : $\mathcal{A}^* \to \mathfrak{P}(\mathcal{A}^*)$, $w \mapsto \{wu \mid u \in \mathcal{A}^*\}$
- *cont* : $\mathfrak{P}(\mathcal{A}^*) \to \mathfrak{P}(\mathcal{A}^*)$, $L \mapsto \bigcup\{cont(w) \mid w \in L\}$

For composable EIOs P_1 and P_2, consider a run of their parallel composition $P_1 \parallel P_2$ that justifies statement $(p_1, p_2) \overset{w}{\Longrightarrow} (p_1, p_2)$ for $w \in \mathcal{A}^*$. It is well known and not difficult to see that such a run can be projected to runs of P_1 and P_2, passing through all the first, second respective, components of the states of the composed run. These *projected runs* justify $p_i \overset{w_i}{\Longrightarrow} p_i$

with $w|_{\mathcal{A}_i} = w_i$, $i = 1, 2$. Vice versa, any two runs of P_1 and P_2 justifying $p_i \overset{w_i}{\Longrightarrow} p_i$ with $w|_{\mathcal{A}_i} = w_i$, $i = 1, 2$, are projections of a unique run of $P_1 \parallel P_2$ justifying $(p_1, p_2) \overset{w}{\Longrightarrow} (p_1, p_2)$. From this, the first claim of the next lemma follows.

Each run of $P_1 \parallel P_2$ corresponds to one of $P_1 \mid P_2$ – simply replace some actions by τ. We also call the projected runs of the former the *projections* of the latter. In such a case, we also say that the $p_i \overset{w_i}{\Longrightarrow} p_i$, $i = 1, 2$, are the *projections* of $(p_1, p_2) \overset{w'}{\Longrightarrow} (p_1, p_2)$ in $P_1 \mid P_2$, where $w' \in w_1 \mid w_2$.[3] These considerations justify the second claim of the next lemma.

Lemma 3.5 (Basic Language of Composition). *For two composable EIOs P_1 and P_2, we have*

1. $L(P_1 \parallel P_2) = L(P_1) \parallel L(P_2)$
2. $L(P_1 \mid P_2) = L(P_1) \mid L(P_2)$.

Returning to the example of Fig. 3.1, it is easy to see that *msg buffer msg cancel* is in $L(P_1)$ and *buffer send* is in $L(P_2)$. Composing them according to Def. 3.3 yields the set $\{$*msg msg cancel send*, *msg msg send cancel*, *msg send msg cancel*$\} \subseteq L(P_1) \mid L(P_2)$.

3.3 Optimistic Approach: Local Errors

We are now ready to consider some basic refinement relations. We will use variations of the notation '$P \sqsubseteq^B Q$' to denote that P in some basic sense is an implementation of, i.e. refines, the specification Q.

3.3.1 Precongruence

In this section, we will start with a variant based on *local* (i.e. internal and output) actions. We consider the following requirement: if a specification is error-free in the sense that it cannot reach an error state by local actions only, then any implementation must be error-free as well. This is an optimistic view: it only considers processes to be faulty, if they can run into an error on their own, i.e. using only local actions. Formally:

Definition 3.6 (Local Basic Relation). *An* error is locally reachable *in an EIO P, if $\exists w \in O^* : p_0 \overset{w}{\Longrightarrow} p \in E$. For EIOs P and Q with the same*

[3] This is a slight abuse of language, since these projections have additional actions and are not really unique; the possible differences do not matter.

signature, we write $P \sqsubseteq_{loc}^{B} Q$, whenever an error is locally reachable in P only if an error is locally reachable in Q.

We let $\sqsubseteq_{loc}^{c}$ denote the fully abstract *precongruence with respect to $\sqsubseteq_{loc}^{B}$ and $|$, i.e. the coarsest precongruence with respect to $|$ that is contained in $\sqsubseteq_{loc}^{B}$.*

In order to characterize this coarsest precongruence, we will need several trace sets. Of course, we are interested in those traces that can reach an error state, the so-called strict error traces. Furthermore, consider an EIO that can perform a trace w such that input a is not possible in the state reached. If the environment allows this state to be reached by providing the necessary inputs and then performs a as an output, a new error state arises in the composition. Thus, we are also interested in the sequence wa and call it a missing-input trace.

Definition 3.7 (Error Traces)**.** *We define the following trace sets for an EIO P:*

- strict error traces*: $StT(P) = \{w \in \mathcal{A}^* \mid p_0 \overset{w}{\Longrightarrow} p \in E\}$*
- pruned error traces*: $PrT(P) = \{prune(w) \mid w \in StT(P)\}$*
- missing-input traces*: $MIT(P) = \{wa \in \mathcal{A}^* \mid p_0 \overset{w}{\Longrightarrow} p \wedge a \in I \wedge p \overset{a}{\not\longrightarrow}\}$*

The characterization we are looking for will be provided by the following local error semantics; the intuitions are as follows. Errors arise in a composition because a component cannot accept some input after a trace or because it performs a strict error trace; in the latter case, the error is already unavoidable if the error state can be reached by local actions only. Hence, we consider the trace sets PrT and MIT in the definition of ET below. But as already explained above, the other component must take part in such problematic behaviour, hence we are also interested in the basic language of a component.

If *along* an action sequence an error *can* occur, it does not matter whether the sequence can be performed at all, and if so, whether it leads to an error state. Thus, we want to obliterate this information about such a sequence; for this purpose, we close the set of problematic traces under continuation, and we also include this extended set in the language; this technique of flooding is well known e.g. in the context of failures semantics [11].

It will turn out that we can characterize $\sqsubseteq_{loc}^{c}$ as component-wise set inclusion for pairs $(ET(P), EL(P))$, and $\sqsubseteq_{loc}$ denotes this relation.

Definition 3.8 (Local Error Semantics)**.** *Let P be an EIO.*

- *The set of* error traces *of P is* $ET(P) = \mathit{cont}(PrT(P)) \cup \mathit{cont}(MIT(P))$*;*
- *the* flooded language *of P is* $EL(P) = L(P) \cup ET(P)$*.*

For two EIOs P and Q with the same signature, we write

$$P \sqsubseteq_{loc} Q \text{ if } ET(P) \subseteq ET(Q) \text{ and } EL(P) \subseteq EL(Q)$$

and we call P and Q local-error equivalent*, $P =_{loc} Q$, if $P \sqsubseteq_{loc} Q$ and $Q \sqsubseteq_{loc} P$.*

We illustrate some aspects of this semantics with a small example. Consider the EIO P from Fig. 3.2. This system can receive some data via *import*. Import does not always work properly, however, and sometimes the subsequent *calc* command is not accepted and no *result* is returned. Alternatively the data can be entered *manual*ly. This causes the system to crash after sending a *crashed* output.

The local error semantics of this EIO is $ET(P) = \{\mathit{import}\} \cdot I \cdot \mathcal{A}^* \cup \{\mathit{manual}\} \cdot \mathcal{A}^*$ and $EL(P) = ET(P) \cup \{\mathit{import}\}$. Observe that, according to the semantics, p_5 is as good as an error state, since it can reach one via an output. An environment cannot stop P from reaching an error, as soon as it sends *manual*. On the other hand, consider the behaviour after *import*. The fact that *calc* can sometimes be accepted is irrelevant, since it might just as well produce an error.

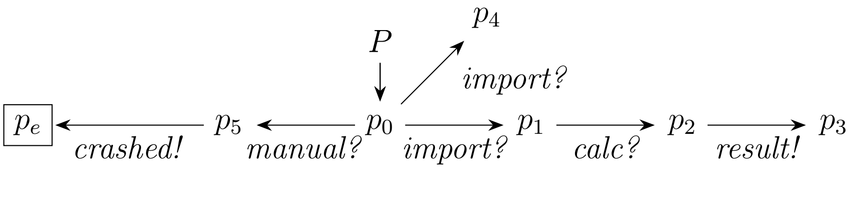

Figure 3.2: An example for the local error semantics

For the characterization result, it is crucial that the local error semantics is compositional – which is desirable anyway.

Theorem 3.9 (Local Error Semantics for Composition)**.** *For two composable EIOs P_1, P_2 and $P_{12} = P_1 \mid P_2$ we have:*

1. $ET_{12} = \mathit{cont}\Big(\mathit{prune}\Big(\big(ET_1 \mid EL_2\big) \cup \big(EL_1 \mid ET_2\big)\Big)\Big)$
2. $EL_{12} = \big(EL_1 \mid EL_2\big) \cup ET_{12}$

Proof. 1.a) '$\subseteq$':
Since both sides are closed under *cont*, it suffices to consider a prefix-minimal element w of ET_{12}. This means w is in MIT_{12} or in PrT_{12}.

First we consider the case, that $w \in MIT_{12}$:
We know that $w = xa$ with $(p_{01}, p_{02}) \overset{x}{\Longrightarrow} (p_1, p_2) \overset{a}{\not\longrightarrow}$, $a \in I_{12}$. Since $a \in I_{12}$, it holds that $a \in I_1 \uplus I_2$ and $a \notin O_1 \cup O_2$. Let w.l.o.g. $a \in I_1$. Thus by projection we get $p_{01} \overset{x_1}{\Longrightarrow} p_1 \overset{a}{\not\longrightarrow}$ and $p_{02} \overset{x_2}{\Longrightarrow}$ (i.e. $x_2 \in L_2$) with $x \in x_1 \mid x_2$. Thus we know that $x_1 a \in ET_1$ and $x_2 \in L_2 \subseteq EL_2$, and it follows that $w \in (x_1 \mid x_2) \cdot \{a\} \subseteq x_1 a \mid x_2 \subseteq ET_1 \mid EL_2$, which is contained in the r.h.s. set.

Now we get to the second case: $w \in PrT_{12}$
In this case we know that there exists $u \in O_{12}^*$ such that $(p_{01}, p_{02}) \overset{w}{\Longrightarrow} (p_1, p_2) \overset{u}{\Longrightarrow} (p'_1, p'_2)$ with $(p'_1, p'_2) \in E_{12}$ and $w = prune(wu)$.

By projection we get $p_{01} \overset{w_1}{\Longrightarrow} p_1 \overset{u_1}{\Longrightarrow} p'_1$ and $p_{02} \overset{w_2}{\Longrightarrow} p_2 \overset{u_2}{\Longrightarrow} p'_2$ with $w \in w_1 \mid w_2$ and $u \in u_1 \mid u_2$. Since $(p'_1, p'_2) \in E_{12}$ it follows that either (p'_1, p'_2) is an inherited error due to $p'_1 \in E_1$ or $p'_2 \in E_2$, or it is a new error due to some $a \in O_1 \cap I_2$ with $p'_1 \overset{a}{\longrightarrow} \wedge p'_2 \overset{a}{\not\longrightarrow}$ or some $a \in I_1 \cap O_2$ with $p'_1 \overset{a}{\not\longrightarrow} \wedge p'_2 \overset{a}{\longrightarrow}$.

If it is an inherited error, then let $p'_1 \in E_1$ w.l.o.g. Thus, we know that $w_1 u_1 \in StT_1 \subseteq ET_1$. Because of $p_{02} \overset{w_2 u_2}{\Longrightarrow}$, we get $w_2 u_2 \in L_2 \subseteq EL_2$. Hence $wu \in ET_1 \mid EL_2$ and $w = prune(wu)$ is in the r.h.s. set.

If (p'_1, p'_2) is a new error, let w.l.o.g. $a \in I_1 \cap O_2$ with $p'_1 \overset{a}{\not\longrightarrow} \wedge p'_2 \overset{a}{\longrightarrow}$. Thus we know that $w_1 u_1 a \in MIT_1 \subseteq ET_1$ and $w_2 u_2 a \in L_2 \subseteq EL_2$. By definition of $\mid$ we know that $w_1 u_1 a \mid w_2 u_2 a = w_1 u_1 \mid w_2 u_2$ and thus we are done as above.

1.b) '$\supseteq$':
It should be noted that $P_1 \parallel P_2$ an $P_1 \mid P_2$ have the same states, error states and input actions. Consequently, using the *prune*-function on some trace of $P_1 \parallel P_2$ yields $v = \varepsilon$ or v ending with some $b \in I_{12} = I_{P_1 \mid P_2}$.

Again it suffices to consider a prefix-minimal element x. For such an x it holds that:
$x \in prune\Big(\big(ET_1 \mid EL_2\big) \cup \big(ET_2 \mid EL_1\big)\Big)$
Since x is the result of the *prune* function, we consider $xy \in \big(ET_1 \mid EL_2\big) \cup \big(ET_2 \mid EL_1\big)$ with $y \in O_{12}^*$. W.l.o.g we assume $xy \in ET_1 \mid EL_2$, i.e. there is $w_1 \in ET_1$ and $w_2 \in EL_2$ with $xy \in w_1 \mid w_2$. We also get $w \in w_1 \parallel w_2$ such that $w|_{\mathcal{A}_{12}} = xy$.

Below, we will treat several cases, and in each case we will show that there is some $v \in PrT(P_1 \parallel P_2) \cup MIT(P_1 \parallel P_2)$ which is a prefix of w and either ends with an input action of $P_1 \mid P_2$ or is ε. In both cases $v|_{\mathcal{A}_{12}}$ is a prefix of x. In case of $v|_{\mathcal{A}_{12}} = \varepsilon$, the latter is obvious. Otherwise $v|_{\mathcal{A}_{12}}$ ends with some input action $b \in I_{12}$ and it has to be a prefix of xy by construction of w. Since $y \in O_{12}^*$, this $v|_{\mathcal{A}_{12}}$ has to be a prefix of x. Therefore x has a prefix in $PrT(P_1 \mid P_2) \cup MIT(P_1 \mid P_2)$ and we are done.

Let v_1 be the shortest prefix of w_1 that is in $PrT_1 \cup MIT_1$. If $w_2 \in L_2$, let $v_2 = w_2$; otherwise, let v_2 be the shortest prefix of w_2 that is in $PrT_2 \cup MIT_2$. Every action of v_1 and v_2 has its corresponding action in w. We now assume that $v_2 = w_2 \in L_2$ or the last action of v_1 is before or the same as the last action of v_2. Otherwise, $v_2 \in PrT_2 \cup MIT_2$ ends before v_1 and this is analogous to the case where v_1 ends before v_2. (Note that the case $v_2 = w_2 \in L_2$ is needed to cover the situation where w_2 ends before v_1, but is not an error trace.)

If $v_1 = \varepsilon$, then choose $v_2' = v' = \varepsilon$.

If $v_1 \neq \varepsilon$, then v_1 by choice ends with some $a \in I_1$, i.e. $v_1 = v_1'a$. Let v' be the prefix of w that ends with the last action of v_1 and let $v_2' = v'|_{\mathcal{A}_2}$. If $v_2 \in L_2 \cup PrT_2$, then v_2' is a prefix of v_2. If $v_2 \in MIT_2$ then it ends with some $b \in I_2$, i.e. $b \neq a$; according to the above assumption, in this case v_1 must end before v_2 and v_2' is a proper prefix of v_2.

In all cases (including the case $v_1 = \varepsilon$), we get (*) $p_{02} \overset{v_2'}{\Longrightarrow}$. Furthermore, $v_2' = v'|_{\mathcal{A}_2}$ is a prefix of v_2, and $v' \in v_1 \parallel v_2'$ is a prefix of w. Now we have to consider two cases:

First we consider the case, that $v_1 \in MIT_1$ (and $v_1 \neq \varepsilon$ in this case):
In this case we have $p_{01} \overset{v_1'}{\Longrightarrow} p_1 \overset{a}{\not\longrightarrow}$ and we let $v' = v''a$. We have to consider two subcases:

1. If a is not a synchronizing action, i.e. $a \notin \mathcal{A}_2$, then by (*) $p_{02} \overset{v_2'}{\Longrightarrow} p_2$ with $v'' \in v_1' \parallel v_2'$. Therefore $(p_{01}, p_{02}) \overset{v''}{\Longrightarrow} (p_1, p_2) \overset{a}{\not\longrightarrow}$ with $a \in I_{12}$. Thus we can choose $v := v''a = v' \in MIT(P_1 \parallel P_2)$.

2. If $a \in \mathcal{A}_2$, then $a \in O_2$ and $v_2' = v_2''a$. By (*) $p_{02} \overset{v_2''}{\Longrightarrow} p_2 \overset{a}{\longrightarrow}$ with $v'' \in v_1' \parallel v_2''$. Thus, $(p_{01}, p_{02}) \overset{v''}{\Longrightarrow} (p_1, p_2)$ with $p_1 \overset{a}{\not\longrightarrow}$, $a \in I_1$, $p_2 \overset{a}{\longrightarrow}$ and $a \in O_2$; hence $(p_1, p_2) \in E_{12}$. In this case we choose $v := prune(v'') \in PrT(P_1 \parallel P_2)$.

The second case is $v_1 \in PrT_1$ (where we might have $v_1 = \varepsilon$).
In this case $\exists u_1 \in O_1^* : p_{01} \overset{v_1}{\Longrightarrow} p_1 \overset{u_1}{\Longrightarrow} p_1'$ with $p_1' \in E_1$.
Again $p_{02} \overset{v_2'}{\Longrightarrow} p_2$, this time with $(p_{01}, p_{02}) \overset{v'}{\Longrightarrow} (p_1, p_2)$. We have two subcases depending on 'how long' p_2 can 'take part' in u_1.

1. There is some $u_2 \in (O_1 \cap I_2)^*$ and some $c \in (O_1 \cap I_2)$ such that $u_2 c$ is a prefix of $u_1|_{I_2}$ with $p_2 \overset{u_2}{\Longrightarrow} p_2' \overset{c}{\not\longrightarrow}$.

   Consider the prefix $u_1' c$ of u_1 with $u_1' c|_{I_2} = u_2 c$. We know that $p_1 \overset{u_1'}{\Longrightarrow} p_1'' \overset{c}{\longrightarrow}$. Then $u_1' \in u_1' \parallel u_2$ and $(p_1, p_2) \overset{u_1'}{\Longrightarrow} (p_1'', p_2') \in E_{12}$, i.e. we get a new error.

   We can choose $v := \mathit{prune}(v' u_1') \in \mathit{PrT}(P_1 \parallel P_2)$, which is a prefix of v', since $u_1' \in O_1^*$.

2. Otherwise we have $p_2 \overset{u_2}{\Longrightarrow} p_2'$ with $u_2 = u_1|_{I_2}$. Then $u_1 \in u_1 \parallel u_2$ and $(p_1, p_2) \overset{u_1}{\Longrightarrow} (p_1', p_2') \in E_{12}$. This is an error inherited from P_1. Therefore we can again choose $v := \mathit{prune}(v' u_1) \in \mathit{PrT}(P_1 \parallel P_2)$, which again is a prefix of v'.

2. Observe that: $L_i \subseteq EL_i$ and $ET_i \subseteq EL_i$. For better readability, we start from the right hand side of the equation:

$$\begin{aligned}
&(EL_1 \mid EL_2) \cup ET_{12} = \\
&\big((L_1 \cup ET_1) \mid (L_2 \cup ET_2)\big) \cup ET_{12} = \\
&\underbrace{(L_1 \mid ET_2)}_{\subseteq ET_{12} \quad (1)} \cup \underbrace{(ET_1 \mid L_2)}_{\subseteq ET_{12} \quad (1)} \cup (L_1 \mid L_2) \cup \underbrace{(ET_1 \mid ET_2)}_{\subseteq ET_{12} \quad (1)} \cup ET_{12} = \\
&(L_1 \mid L_2) \cup ET_{12} \overset{3.5.2}{=} L_{12} \cup ET_{12} = EL_{12} \qquad \square
\end{aligned}$$

The above theorem implies that parallel composition is *monotonic* w.r.t. $\sqsubseteq_{loc}$, i.e. increasing the error traces and the flooded language of a component increases the respective sets of the parallel composition. Note that $|$, *cont* and *prune* are defined elementwise. Thus $\sqsubseteq_{loc}$ is a precongruence, and next we will show that it is the coarsest one. For this, we will construct a test environment U for each relevant trace w of P that reveals that w is also a suitable trace of Q. The proof technique is similar to the one used in the previous chapter, e.g. for the proof of Thm. 2.7

Lemma 3.10. *For two EIOs P and Q with the same signature, we have: if $U \mid P \sqsubseteq_{loc}^{B} U \mid Q$ for all partners U of P, then $P \sqsubseteq_{loc} Q$.*

Proof. Since P and Q have the same signature, we will write I for $I_P = I_Q$ and O for $O_P = O_Q$ throughout this proof. P and Q have the same partners U; $I_U = O$ and $O_U = I$ for each of these.

Note that $\varepsilon \in ET(P)$ signifies that an error is locally reachable in P, since this can only result from $\varepsilon \in \mathit{PrT}(P)$. We have to show the following inclusions to get $P \sqsubseteq_{loc} Q$:

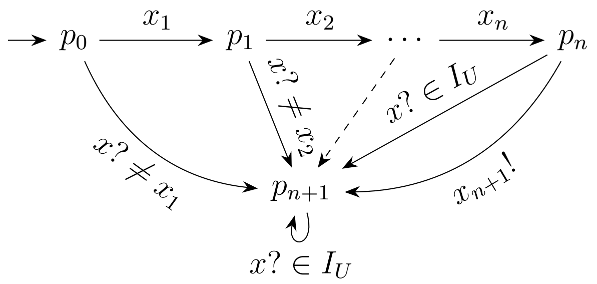

Figure 3.3: $x? \neq x_i$ indicates all $x \in I_U \backslash \{x_i\}$, $x_{n+1}!$ indicates $x_{n+1} \in O_U$

- $ET(P) \subseteq ET(Q)$
- $EL(P) \subseteq EL(Q)$

For the first inclusion we consider a prefix minimal element $w \in ET(P)$. It suffices to show that w or any of its prefixes is in $ET(Q)$. The construction for $w = \varepsilon$ is a simple variant of the next one. If $w = \varepsilon$, then an error state is locally reachable in P. Let U consist of the initial state only with a loop for each $x \in I_U$; P can reach the same error state in $P \,|\, U$ internally. Thus, $Q \,|\, U$ must also have a locally reachable error state. By definition of U, this can only be inherited from Q. Thus, Q must have an error state that is reachable by internal and output actions only, i.e. $\varepsilon \in PrT(Q)$.

So we assume that $w = x_1 \cdots x_n x_{n+1} \in \mathcal{A}^+$ with $n \geq 0$ and $x_{n+1} \in I$. We consider the following partner U; see Fig. 3.3.

- $P_U = \{p_0, p_1, \ldots p_{n+1}\}$
- $p_{0U} = p_0$
- $E_U = \emptyset$
- $\longrightarrow_U = \{(p_i, x_{i+1}, p_{i+1}) \mid 0 \leq i \leq n\} \cup \{(p_i, x, p_{n+1}) \mid x \in I_U \backslash \{x_{i+1}\}, 0 \leq i \leq n\} \cup \{(p_{n+1}, x, p_{n+1}) \mid x \in I_U\}$

For w we can distinguish two cases. Both will lead to $\varepsilon \in StT(P \mid U)$.
If $w \in MIT(P)$, then in $P \parallel U$ we have $(p_{0P}, p_{)} \overset{x_1 \cdots x_n}{\Longrightarrow} (p', p_n)$ with $p' \overset{x_{n+1}}{\not\longrightarrow}$ and $p_n \overset{x_{n+1}}{\longrightarrow}$. Therefore $(p', p_n) \in E_{P|U}$ and $\varepsilon \in StT(P \mid U)$.
If $w \in PrT(P)$, then in $P \parallel U$ we have $(p_{0P}, p_{0U}) \overset{w}{\Longrightarrow} (p'', p_{n+1}) \overset{u}{\Longrightarrow} (p', p_{n+1})$ with some $u \in O^*$ and $p' \in E_P$. The latter implies $(p', p_{n+1}) \in E_{P|U}$, and again $\varepsilon \in StT(P \mid U)$.

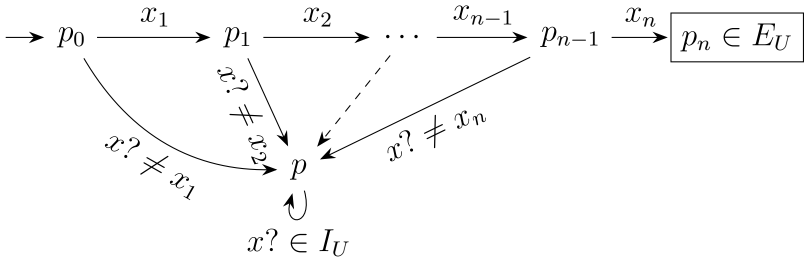

Figure 3.4: $x? \neq x_i$ indicates all $x \in I_U \backslash \{x_i\}$

Since we now know that $\varepsilon \in StT(P|U)$, we also know from $P|U \sqsubseteq_{loc}^{B} Q|U$ that there is a locally reachable error in $Q \mid U$ as well.
There are two kinds of error states $Q \mid U$ can have: new or inherited. Since each state of U enables every $x \in O = I_U$, a locally reachable new error has to be one where U enables an output $a \in O_U$ which is not enabled in Q. By construction p_{n+1} enables no outputs, therefore such a new error state has to be of the form (p', p_i) with $i \leq n$, $p' \overset{x_i+1}{\not\longrightarrow}$ and $x_{i+1} \in O_U = I$. Thus, by projection $p_{0Q} \overset{x_1 \cdots x_i}{\Longrightarrow} p' \overset{x_i+1}{\not\longrightarrow}$ and therefore $x_1 \cdots x_{i+1} \in MIT(Q) \subseteq ET(Q)$ is a prefix of w and we are done.
If the locally reachable error is due to an inherited error state, then by projection U has performed some $x_1 \cdots x_i u$ with $u \in I_U^* = O^*$ (possibly $i = 0$) and hence so has Q. With this, Q has reached some state in E_Q. Therefore $prune(x_1 \cdots x_i u) = prune(x_1 \cdots x_i) \in StT(Q)$. Again this is a prefix of w and in $ET(Q)$ and we are done.

For the second inclusion it suffices to show that $L(P) \backslash ET(P) \subseteq EL(Q)$, because of the first inclusion and the definition of EL.
For this we consider a $w \in L(P) \backslash ET(P)$ and show that it is in $EL(Q)$. If $w = \varepsilon$ we are done, since ε always is in $EL(Q)$. Therefore we consider $w = x_1 \cdots x_n$ with $n \geq 1$ and construct a partner U (illustrated in Fig. 3.4) with:

- $P_U = \{p, p_0, p_1, \ldots p_n\}$

- $p_{0U} = p_0$

- $E_U = \{p_n\}$

- $\longrightarrow_U = \{(p_i, x_{i+1}, p_{i+1}) \mid 0 \leq i < n\} \cup \{(p_i, x, p) \mid x \in I_U \backslash \{x_{i+1}\}, 0 \leq i \leq n\} \cup \{(p, x, p) \mid x \in I_U\}$

Because of $p_{0P} \overset{w}{\Longrightarrow} p$ we know that $P\,|\,U$ has a locally reachable error. Thus, because of $P\,|\,U \sqsubseteq^B_{loc} Q\,|\,U$, Q also has to have a locally reachable error state. Firstly, this could be a new error because of some $x_i \in O_U$ and $p_{0Q} \overset{x_1\cdots x_{i-1}}{\Longrightarrow} p' \overset{x_i}{\not\longrightarrow}$. In this case $x_1 \cdots x_i \in MIT(Q)$ and thus $w \in EL(Q)$. Note, that outputs of U are only enabled along this trace. Therefore there are no other outputs of U, which could lead to a new error.
Secondly it could be a new error due to some $a \in O_Q$, which U could not match. But the only state of U in which not all inputs are enabled is p_n, which already is an error state. If this state is reachable in $Q \mid U$, then the composed EIO has an inherited error and thus $w \in L(Q) \subseteq EL(Q)$.
Thirdly it can be an error inherited from U. Since the only state in E_U is p_n and all actions are synchronized, this is only possible if $p_{0Q} \overset{x_1\cdots x_n}{\Longrightarrow}$. In this case $w \in L(Q)$ and we are done.
Finally, the error could have been inherited from Q. In this case $p_{0Q} \overset{x_1\cdots x_i u}{\Longrightarrow} p' \in E_Q$, for some $i \geq 0$ and $u \in O^*$. This means that $x_1 \cdots x_i u \in StT(Q)$ and thus $prune(x_1 \cdots x_i u) = prune(x_1 \cdots x_i) \in PrT(Q) \subseteq EL(Q)$. Hence again $w \in EL(Q)$ and we are done. □

Theorem 3.11 (Full Abstractness for Local Error Semantics)**.** *Let P and Q be two EIOs with the same signature. Then $P \sqsubseteq^c_{loc} Q \Leftrightarrow P \sqsubseteq_{loc} Q$; in particular, $\sqsubseteq_{loc}$ is a precongruence.*

Proof. As noted above, Theorem 3.9 implies that $\sqsubseteq_{loc}$ is a precongruence. As mentioned in the proof of Lemma 3.10 above, $\varepsilon \in ET(P)$ signifies that an error is locally reachable in P. Hence, $P \sqsubseteq_{loc} Q$ implies that $\varepsilon \in ET(Q)$ whenever $\varepsilon \in ET(P)$, and thus also that $P \sqsubseteq^B_{loc} Q$. This proves the reverse implication.

For the implication, assume $P \sqsubseteq^c_{loc} Q$; by definition of $\sqsubseteq^c_{loc}$, this implies $P \mid U \sqsubseteq^c_{loc} Q \mid U$ and then $P \mid U \sqsubseteq^B_{loc} Q \mid U$ for all EIOs U composable with P in general and all partners of P in particular. Thus, we are done by Lemma 3.10. □

3.3.2 Comparison to Interface Automata

We will show now that, up to local-error equivalence, we can essentially work with EIOs without error states. Such EIOs are exactly the *interface automata* of [27], if they additionally are *input-deterministic*: if $p \xrightarrow{a} p'$ and $p \xrightarrow{a} p''$ for some $a \in I$, then $p' = p''$. The only difference is that, in a setting with EIOs without error states, we do not have EIOs anymore that show an error initially.

Theorem 3.12 (Removing Error States). *Let P be an EIO, and let $prune(P)$ be obtained from P by removing the* illegal *states in $illegal(P) = \{p \in P \mid$ an error state is reachable from p by local actions$\}$, their in- and out-going transitions and all transitions $p \xrightarrow{a} p'$ where $p \xrightarrow{a} p''$ with $p'' \in illegal(P)$ for some $a \in I$. If $p_0 \notin illegal(P)$, $prune(P)$ is an EIO and local-error equivalent to P.*

Proof. We assume $p_0 \notin illegal(P)$; then the claim about $prune(P)$ being an EIO is obvious.

For $P \sqsubseteq_{loc} prune(P)$, consider some $w \in PrT(P)$ and a suitable underlying run $p_0 \xrightarrow{a_1} p_1 \xrightarrow{a_2} \cdots p_n$. Observe that p_n is an illegal state and missing in $prune(P)$. So let p_i be the first state on the run such that $p_i \xrightarrow{a_{i+1}} p_{i+1}$ is missing in $prune(P)$. This means that p_i is not illegal, a_{i+1} is an input, and some p with $p_i \xrightarrow{a_{i+1}} p$ is illegal. This implies that some prefix of w is in $MIT(prune(P))$, and $w \in ET(prune(P))$.

For $wa \in MIT(P)$, we argue similarly. Either some suitable run underlying w is still in $prune(P)$ and $wa \in MIT(prune(P))$, or some transition of the run is missing in $prune(P)$ and wa has a prefix in $MIT(prune(P))$. Thus, $ET(P) \subseteq ET(prune(P))$.

Analogously for $w \in L(P)$, either some run underlying w is still in $prune(P)$ and $w \in L(prune(P))$, or some transition of the run is missing and w has a prefix in $MIT(prune(P))$. Thus, $EL(P) \subseteq EL(prune(P))$.

For $ET(prune(P)) \subseteq ET(P)$, we just consider $wa \in MIT(prune(P))$, where $a \in I$. Consider a suitable run $p_0 \dots p$ underlying w in $prune(P)$. Either, p has no a-transition in P and $wa \in MIT(P)$, or $p \xrightarrow{a} p'$ for some illegal p' and $wa \in PrT(P)$. Thus, $ET(prune(P)) \subseteq ET(P)$.

For $w \in L(prune(P))$, each run underlying w is still in P and $w \in L(prune(P))$. Thus, $EL(P) \subseteq EL(P)$. □

The respective pruning in the definition of parallel composition in [27] only removes transitions from legal to illegal states. (Since then the illegal states are unreachable, they can be removed as well.) The additional removal of transitions $p \xrightarrow{a} p'$ as described in the theorem is obviously redundant in case of input determinism.

Thus, according to Theorem 3.12, we could work with EIOs without error states; whenever we put such EIOs in parallel, we have to normalize the result, i.e. we take $prune(P_1 \mid P_2)$ as parallel composition. We only have to make sure that this is well-defined: we call EIOs P_1 and P_2 *compatible*, if the initial state of $P_1 \mid P_2$ is not illegal, and we only apply the new parallel composition to compatible P_1 and P_2. For this, we have:

Proposition 3.13. *If Q and Q' are compatible EIOs and $P \sqsubseteq_{loc} Q$, then also P and Q' are compatible.*

Proof. If P and Q' are not compatible, then $\varepsilon \in ET(P \mid Q')$. Now $ET(P \mid Q') \subseteq ET(P|Q)$ by Theorem 3.9, hence also P and Q are not compatible. $\square$

Thus, also on the level of transition systems, pruning as introduced in [27] is justified according to our approach. But the refinement relation based on alternating simulation in [27] is somewhat arbitrarily too strict, as we will show below. To our best knowledge, alternating simulation as refinement relation has first been considered for modal transition systems [46]; see [42] for a comparison to the setting of interface automata.[4]

Since the refinement relation of [27] is a precongruence, one might believe that, due to our coarsest precongruence result, this refinement should directly imply $\sqsubseteq_{loc}$. This is not really so obvious: we have considered parallel components that are not interface automata (due to violation of input determinism), and this could have forced us to be too strict w.r.t. alternating simulation. The next proposition shows that actually the implication holds. It also shows that alternating simulation is unnecessarily strict if one is only interested in avoiding local errors.

Definition 3.14. *For EIOs P_1 and P_2 with the same signature, an* alternating simulation relation *from P_1 to P_2 is some $\mathcal{R} \subseteq P_1 \times P_2$ with $(p_{01}, p_{02}) \in \mathcal{R}$ such that for all $(p_1, p_2) \in \mathcal{R}$ we have:*

1. *If $p_2 \xrightarrow{a} p_2'$ and $a \in I_1$, then $p_1 \xrightarrow{a} p_1'$ and $(p_1', p_2') \in \mathcal{R}$.*
2. *If $p_1 \xrightarrow{a} p_1'$ and $a \in O_1$, then $p_2 \Longrightarrow\xrightarrow{a} p_2'$ and $(p_1', p_2') \in \mathcal{R}$.*
3. *If $p_1 \xrightarrow{\tau} p_1'$, then $p_2 \Longrightarrow p_2'$ and $(p_1', p_2') \in \mathcal{R}$.*

Thus, implementation P_1 must match a prescribed input immediately, while an output or τ is allowed for P_1 if P_2 can match it using internal steps.

Proposition 3.15. *If there exists some alternating simulation relation $\mathcal{R}$ for interface automata P_1 and P_2, then $P_1 \sqsubseteq_{loc} P_2$. This implication is strict.*

Proof. Since interface automata do not have error states, we just have to consider $wa \in MIT(P_1)$ with $a \in I$ for $ET(P_1) \subseteq ET(P_2)$. Take a suitable run in P_1 underlying w, and build up a matching run in P_2 as follows. To start with, the initial states are related according to $\mathcal{R}$. Each output or internal

[4] A concept called alternating simulation also appears in [2], but it is applied to so-called alternating transition systems. It is accompanied by some alternating trace containment, but we have been unable to see any connection.

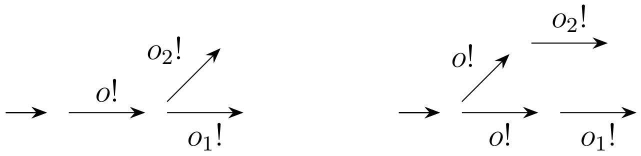

Figure 3.5: Local-error equivalent automata, unnecessarily distinguished by alternating refinement

transition in P_1 can be matched according to 3.14.2 or 3.14.3, reaching related states again. If the runs have reached $(p_1, p_2) \in \mathcal{R}$ so far and the next transition is $p_1 \xrightarrow{a} p_1'$ with $a \in I_1$, then either p_2 does not have an a-transition and a prefix of w is in $MIT(P_2)$, or $p_2 \xrightarrow{a} p_2'$ and $(p_1', p_2') \in \mathcal{R}$ due to input determinism. If the run in P_1 ends and we have reached $(p_1, p_2) \in \mathcal{R}$, then p_2 cannot have an a-transition due to 3.14.1 and $wa \in MIT(P_2)$.

The treatment of $w \in L(P_1)$ is analogous, except that we do not have to consider a missing action after w at the end.

To conclude that alternating simulations are really unnecessarily strict, consider the interface automata in Fig. 3.5. They have no inputs and are local-error equivalent since there are no error traces and the basic languages are the same. But there exists no alternating refinement relation from the first to the second, since whichever way the second interface automaton matches o, it will forbid one of o_1 or o_2 afterwards. □

Next, we will show associativity for parallel composition. As mentioned in the introduction, Theorem 1 of [25] claiming this associativity is wrong there due to an error in the definition of pruning; a proof of such a theorem is a bit tricky when composition involves pruning, which is not the case in our setting.

To demonstrate the problem, Fig. 3.6 shows three systems P, Q and R. Their respective signatures are $I_P = \{i\}, O_P = \{o\}, I_Q = \{o\}, O_Q = \emptyset, I_R = \emptyset, O_R = \{i, x\}$. The system $Q \,|\, R$ is isomorphic to R, the only difference being its signature, since it now has o as an input. P and $Q \,|\, R$ are not compatible in [25], since the initial state of their parallel composition would be illegal – as it is in our setting. $(P \,|\, Q) \,|\, R$ however is a valid system according to [25], as depicted in Fig. 3.6.

The problem arises when constructing $P \mid Q$ as shown in Fig. 3.6: the removal of illegal states as introduced in [25] removes only one of P's input transitions – namely the one that leads to an error state – but leaves the other in place. This problem disappears in later work of de Alfaro and Henzinger on interface automata, where the automata are required to be

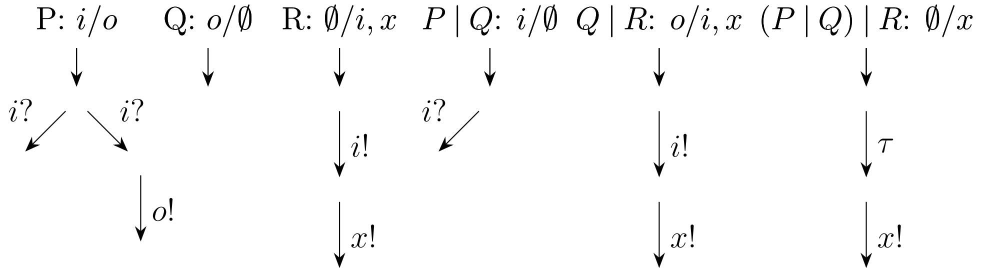

Figure 3.6: Counter example for associativity in [25]

input deterministic. In our approach, $P \mid Q$ has an i-transition to an error state, and thus the initial state of $(P|Q)|R$ has a τ-transition to an error state and is illegal itself. Explained another way: our pruning of $P \mid Q$ removes all input transitions and therefore $(P \mid Q) \mid R$ has an error state as initial state.

The problem also demonstrates the danger when one develops an unorthodox definition justified with informal intuitive arguments only. In this chapter, pruning on EIOs is *proven correct* in Theorem 3.12, and this proof would fail with some incorrect definition of pruning.

In our setting with error states associativity is easy, because the two systems are easily seen to be isomorphic. Hence, associativity holds for any sensible equivalence on EIOs.

Theorem 3.16. *For pairwise composable EIOs P_1, P_2 and P_3, $P_1 \mid (P_2 \mid P_3)$ and $(P_1 \mid P_2) \mid P_3$ are isomorphic and in particular local-error equivalent.*

Proof. Both EIOs are isomorphic to an EIO with state set $P_1 \times P_2 \times P_3$, (p_{01}, p_{02}, p_{03}) as initial state, signature as that of $P_1|(P_2|P_3)$ and the following transitions and error states.

Transitions:

- $(p_1, p_2, p_3) \xrightarrow{\alpha} (p'_1, p_2, p_3)$ if $p_1 \xrightarrow{\alpha} p'_1$ and $\alpha \in (\mathcal{A}_1 \cup \{\tau\}) \backslash (\mathcal{A}_2 \cup \mathcal{A}_3)$ and similarly for transitions derived from P_2 and P_3 instead of P_1

  and

- $(p_1, p_2, p_3) \xrightarrow{\tau} (p'_1, p'_2, p_3)$ if $p_1 \xrightarrow{a} p'_1$ and $p_2 \xrightarrow{a} p'_2$ for some $a \in \mathcal{A}_1 \cap \mathcal{A}_2$ and similarly for transitions derived from other pairs instead of P_1 and P_2. Clearly, if a visible action belongs to two alphabets, it must be an input of one EIO and an output of another by assumption. Also by assumption, a visible action cannot be common to all EIOs:

such an action would have to be a common input or a common output of two systems.

Error states: (p_1, p_2, p_3) is an error state

- if $p_1 \in E_1$ and similarly for E_2 or E_3 instead of E_1

  or

- if $p_1 \xrightarrow{a} p_1'$ and $\neg p_2 \xrightarrow{a}$ for some $a \in O_1 \cap I_2$ or similarly for one of the other five pairs instead of (P_1, P_2).

Observe that, in the latter item, (p_2, p_3) possibly is not an error state and $(p_1, (p_2, p_3)))$ is a new one, while (p_1, p_2) is an error state and $((p_1, p_2), p_3)$ is an inherited one. □

According to this theorem, also other equivalences in this chapter make $|$ associative, and commutativity is obvious in the sense that composability is symmetric and that $P \mid P'$ and $P' \mid P$ are isomorphic for composable P and P'. In this context, it is useful to mention the following result, giving a sufficient condition for a fully abstract precongruence. Although we regard it as some form of folklore – see e.g. [70, Sect. 3.2] for similar considerations –, we do not know whether exactly this result has been published before. Theorem 3.17 is independent of our particular setting with EIOs and $|$, and we will adapt it below to our setting.

Theorem 3.17. *Let $\leq^B$ be a preorder on a set M such that some binary operation $\circ$ is commutative and associative for the related equivalence $=^B$, and there exists $Nil \in M$ with $Nil \circ P =^B P$ for all $P \in M$. Let preorder $\leq$ satisfy for all $P, Q \in M$: $P \leq Q$ iff $U \circ P \leq^B U \circ Q$ for all $U \in M$. Then $\leq$ is $\leq^c$, the fully abstract precongruence for $\leq^B$ and $\circ$.*

Proof. First, we observe that, for all $P, Q \in M$, $P \leq Q \Rightarrow P =^B Nil \circ P \leq^B Nil \circ Q =^B Q$. Hence, $\leq$ is contained in $\leq^B$.

Second, for all $P, Q \in M$: $P \leq^c Q \Rightarrow \forall U : U \circ P \leq^c U \circ Q \Rightarrow \forall U : U \circ P \leq^B U \circ Q \Rightarrow P \leq Q$. Hence, $\leq^c$ is contained in $\leq$.

Third, we have to show that $\leq$ is a precongruence; then, with the first observation, $\leq$ is also contained in $\leq^c$. Consider arbitrary $P, Q, V \in M$ with $P \leq Q$. We have to show that $V \circ P \leq V \circ Q$. $P \leq Q \Rightarrow \forall U : (U \circ V) \circ P \leq^B (U \circ V) \circ Q \Rightarrow \forall U : U \circ (V \circ P) \leq^B U \circ (V \circ Q)$ by associativity $\Rightarrow V \circ P \leq V \circ Q$.

Similarly, one can show $P \circ V \leq Q \circ V$, using commutativity in addition to associativity. □

We now adapt this theorem to our setting, where we can choose Nil as an EIO with just one state (the initial one), no transitions and empty alphabet. The interesting point is that our operator $|$ is only defined on composable systems, i.e. it is a partial operator. Consequently, associativity is also only given for pairwise composable systems. For the adapted theorem, our basic relation has to support α-conversion and to respect isomorphism.

Definition 3.18 (Bijective Relabelling, Sensible Relation)**.** *A* bijective relabelling function *for an EIO P is a bijective function $f : I \cup O \to \mathcal{A}'$. The* bijective relabelling *of an EIO is the EIO $f(P) = (P, f(I), f(O), \longrightarrow', p_0, E)$ where $\longrightarrow' = \{(p, f(a), p') \mid (p, a, p') \in \longrightarrow\}$.*

We call a relation $\leq$ between EIOs sensible *if it relates only EIOs with the same signature, respects isomorphism, i.e. for all isomorphic EIOs P and P' we have $P = P'$ ($P \leq P'$ and $P' \leq P$), and it also respects bijective relabelling in the sense that $P \leq P'$ implies $f(P) \leq f(P')$.*

Lemma 3.19. *Let $\sqsubseteq^B$ be a sensible preorder on EIOs. Then $|$ is commutative and associative in the sense of Theorem 3.16 for the related equivalence $=^B$. Let preorder $\sqsubseteq$ satisfy for all EIOs P and Q: $P \sqsubseteq Q$ iff $U \mid P \sqsubseteq^B U \mid Q$ for all composable U. Then $\sqsubseteq$ respects bijective relabelling.*

Proof. The first claim is obvious by the definition of sensible and Theorem 3.16. Let f be a bijective relabelling for P (and hence for Q as well). Assuming $P \sqsubseteq Q$ we have to show $f(P) \sqsubseteq f(Q)$, which is equivalent to $\forall U$ composable with $f(P) : \; U \,|\, f(P) \sqsubseteq^B U \,|\, f(Q)$. Consider such an EIO U. We define a bijective relabelling g for U as follows: $g(a) := f^{-1}(a)$ if $a \in \mathcal{A}_{f(P)}$ and otherwise $g(a)$ is some fresh a' where $a' \notin \mathcal{A}_P$. Thus by construction $f(P) \,|\, U$ differs from $P \,|\, g(U)$ only by bijective relabelling; analogously for Q: common actions are relabelled according to f^{-1}, as are the actions specific to P. All actions of U that are not actions of $f(P)$ are mapped to actions of $g(U)$ that are not actions of P.

Thus we have $U \mid f(P) \sqsubseteq^B U \mid f(Q) \Leftrightarrow g(U) \mid P| \sqsubseteq^B g(U) \mid Q$. Since $P \sqsubseteq Q$ implies $\forall V$ composable with $P : \; V \mid P \sqsubseteq^B V \mid Q$, this also holds for $g(U)$ and thus we are done. □

Theorem 3.20. *Let $\sqsubseteq^B$ be a sensible preorder on EIOs. Let preorder $\sqsubseteq$ satisfy for all EIOs P and Q: $P \sqsubseteq Q$ iff $U \mid P \sqsubseteq^B U \mid Q$ for all composable U. Then $\sqsubseteq$ is $\sqsubseteq^c$, the fully abstract precongruence for $\sqsubseteq^B$ and $|$.*

Proof. Note that there exists an EIO Nil that is composable with all EIOs and satisfies $Nil \mid P =^B P$, to wit: $Nil = (\{p_0\}, \emptyset, \emptyset, \emptyset, p_0, \emptyset)$. Now the first and second point are analogous to the proof of Theorem 3.17, restricting U to EIOs composable with P.

For the third point, we consider EIOs P and Q with $P \sqsubseteq Q$ and an EIO V composable with P (and hence also with Q), and we want to shown that $V \mid P \sqsubseteq V \mid Q$. By assumption this is equivalent to $U \mid (V \mid P) \sqsubseteq^B U \mid (V \mid Q)$ for all U composable with $V \mid P$; so consider such a U.

If U is also composable with V and to P, then $U \mid V$ is composable with P. Thus we get $P \sqsubseteq Q \Rightarrow (U|V)|P \sqsubseteq^B (U|V)|Q \Rightarrow U|(V|P) \sqsubseteq^B U|(V|Q)$ by associativity.

If not, then this must be because of an $a \in \mathcal{A}_U \cap \mathit{Synch}(V, P)$, otherwise U wouldn't be composable with $V \mid P$ either. For this case, we relabel all occurrences of each such action a in V, P and Q with a fresh action a' not in $\mathcal{A}_U$. We call the resulting EIOs V', P' and Q'. This bijective relabelling can be seen as α-conversion since $V \mid P$ and $V \mid Q$ are identical to $V' \mid P'$ and $V' \mid Q'$ respectively. Now, by construction, U, V' and P' are pairwise composable, hence we have associativity in the sense of Theorem 3.16. By Lemma 3.19 we also get $P' \sqsubseteq Q'$. Thus we have $P' \sqsubseteq Q' \Rightarrow (U \mid V') \mid P' \sqsubseteq^B (U \mid V') \mid Q' \Rightarrow U \mid (V' \mid P') \sqsubseteq^B U \mid (V' \mid Q') \Rightarrow U \mid (V \mid P) \sqsubseteq^B U \mid (V \mid Q)$.

Now that we have $U \mid (V \mid P) \sqsubseteq^B U \mid (V \mid Q)$ for all U composable with $V \mid P$, we have $V \mid P \sqsubseteq V \mid Q$ and are done.

Similarly, one can show $P \mid V \sqsubseteq Q \mid V$, using commutativity in addition to associativity. □

This theorem tells us that, for proving that $\sqsubseteq_{loc}$ is fully abstract, it suffices to consider parallel compositions with two components instead of arbitrary ones. In the present setting, we get an even simpler characterization from our earlier results, considering only partners as defined in Definition 3.2.

Corollary 3.21. *For EIOs P and Q, we have $P \sqsubseteq_{loc} Q$ if and only if $U \mid P \sqsubseteq^B_{loc} U \mid Q$ for all partners U.*

Proof. The implication is implied by Theorem 3.11; the reverse implication is Lemma 3.10. □

This result is relevant for the following reason: when a (possibly composed) system is finally put to use, it is composed in parallel with a user, resulting in a closed system. In other words, a user is a partner U, and $U \mid P \sqsubseteq^B_{loc} U \mid Q$ means that the user is as happy with P as she was with Q. For some people, a relation with such a characterization is of foremost interest (see e.g. [63] where a "happy" partner is called a strategy), even though it is not necessarily a precongruence.

The corollary will also be essential for proving below that the third precongruence $\sqsubseteq^c_{act}$ is strictly finer than $\sqsubseteq_{loc}$.

3.4 The Hyper-Optimistic and the Pessimistic Approach

It might seem that, above, we were able to justify pruning (of local actions) just because we concentrated on locally reachable errors. To obtain an even *better justification* for pruning, we briefly examine two other approaches: the hyper-optimistic and the pessimistic. For a more elaborate discussion and formal proofs, we refer to [18].

The first focuses on errors reached by internal actions only. The view that only such errors count is even more optimistic than our first one, since errors reachable by output actions are no longer considered dangerous. The new idea for the resulting semantics is that each error trace is annotated with a set of output actions; traces are pruned again and the set of a pruned trace contains the output actions that are needed to reach an error state, i.e. that occur in the unpruned trace. The intuition for this is: if the system performs the trace while synchronizing with another one on the given output actions, then an error state can be reached internally afterwards. If some action o of these actions is not synchronized, the error state is only reached by performing the still visible o and, hence, not significant.

Our base relation is now defined as:

Definition 3.22 (Internal Basic Relation)**.** *For an EIO P, an* error is internally reachable*, if $\varepsilon \in StT(P)$. For EIOs P and Q with the same signature, we write $P \sqsubseteq^B_{int} Q$ whenever an error is internally reachable in P only if an error is internally reachable in Q. We denote the fully abstract precongruence with respect to $\sqsubseteq^B_{int}$ and $|$ by $\sqsubseteq^c_{int}$.*

Definition 3.23 (Function out, Error Pair)**.** *Given an EIO P, we define $out : \mathcal{A}^* \to \mathfrak{P}(O)$ such that $out(w)$ consists of all output actions in w. An* error pair *over a signature (I, O) is a pair $(w, X) \in (I \cup O)^* \times \mathfrak{P}(O)$ with $out(w) \subseteq X$.*

Given two composable EIOs P_1, P_2, we define for an error pair (w, X) over (I_1, O_1) and $v \in \mathcal{A}_2^*$:
$(w, X) \parallel v = \{(z, Y) \ | \ z \in w \parallel v, Y = X \cup out(v)\}$
$(w, X) \,|\, v = \{(z|_{\mathcal{A}_{12}}, Y \cap \mathcal{A}_{12}) \ | \ (z, Y) \in (w, X) \parallel v\}$

It is easy to see that these sets consist of error pairs over the signatures of $P_1 \parallel P_2$ and $P_{12} = P_1 \,|\, P_2$ respectively. On error pairs over some (I, O), we define the following prefix relation and the following functions, where we generalize cont to sets as in Definition 3.4:

- $(w, X) \sqsubseteq (v, Y)$ *if* $w \sqsubseteq v$ *and* $X \subseteq Y$

- *$prune(w, X) := (prune(w), X)$ (an error pair again)*

- *$cont(w, X) := \{(v, Y) \mid (w, X) \sqsubseteq (v, Y)\}$ (consisting of error pairs)*

Definition 3.24 (Sets of Error Pairs)**.** *We define the following sets of error pairs for an EIO P:*

- strict error pairs*: $StP(P) = \{(w, X) \mid w \in StT(P), out(w) = X\}$*

- pruned error pairs*: $PrP(P) = \{prune(w, X) \mid (w, X) \in StP(P)\}$*

- missing-input pairs*: $MIP(P) = \{(w, X) \mid w \in MIT(P), out(w) = X\}$*

It is easy to see that these sets indeed consist of error pairs over (I, O), and that they are an enhanced version of similar sets defined in the previous section. It will turn out that we can characterize $\sqsubseteq^c_{int}$ as component-wise set inclusion for pairs $(EP(P), EPL(P))$, where the latter is the basic language of P flooded with a set of traces derived from $EP(P)$; we also introduce a sign for this relation.

Definition 3.25 (Internal Error Semantics)**.** *For an EIO P:*

- *the set of* error pairs *of P is $EP(P) = cont(PrP(P)) \cup cont(MIP(P))$;*

- *the set of* error pair traces *of P is $EPT(P) = \{w \mid (w, out(w)) \in EP(P)\}$;*

- *the flooded language of P, called* error pair language*, is $EPL(P) = L(P) \cup EPT(P)$.*

For two EIOs P and Q with the same signature, we write

$$P \sqsubseteq_{int} Q \text{ if } EP(P) \subseteq EP(Q) \text{ and } EPL(P) \subseteq EPL(Q).$$

It can be proven that $\sqsubseteq_{int}$ is the coarsest (i.e. fully abstract) precongruence for $\sqsubseteq^B_{int}$ and $|$. Two points are notable: although outputs do not play a special role for $\sqsubseteq^B_{int}$, pruning of outputs on traces is again essential for our characterization. Since the concept of error-freeness underlying $\sqsubseteq^B_{int}$ is more liberal than the one for $\sqsubseteq^B_{loc}$, it is maybe surprising that the resulting precongruence is strictly finer.

Proposition 3.26. *The internal precongruence $\sqsubseteq_{int}$ is strictly finer than the local precongruence $\sqsubseteq_{loc}$, i.e. for all EIOs P and Q with the same signature, $P \sqsubseteq_{int} Q$ implies $P \sqsubseteq_{loc} Q$, but not the other way round.*

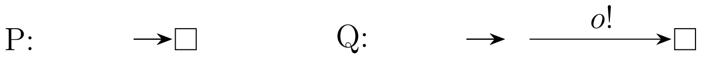

Figure 3.7: $P \sqsubseteq_{loc} Q$, but $P \not\sqsubseteq_{int} Q$

Proof. For the implication, inclusion of the ET-sets follows from the inclusion of the EP-sets, since the former can be obtained from the latter by projection of each pair in $EP(P)$ to the first component. This is not necessarily true for the EL- and the EPL-sets. Consider some $w \in EL(P)$. If $w \in ET(P)$, we are done by the first part. If $w \in L(P)$, then $w \in EPL(P) \subseteq EPL(Q)$; if $w \in L(Q)$, we are done – and otherwise, $w \in EPT(Q)$ is the projection of a pair in $EP(Q)$ and, hence, in $ET(Q)$.

For the second claim, consider P and Q in Fig. 3.7 with $I = \emptyset$ and $O = \{o\}$.

We have $P \sqsubseteq_{loc} Q$ since $ET(P) = ET(Q) = \mathcal{A}^*$ and $EL(P) = EL(Q) = \mathcal{A}^*$. But $P \not\sqsubseteq_{int} Q$ since $(\varepsilon, \emptyset) \in EP(P) = \mathcal{A}^* \times \{\emptyset, \{o\}\}$ but $(\varepsilon, \emptyset) \notin EP(Q) = \mathcal{A}^* \times \{\{o\}\}$. The second part can also be seen from $P \not\sqsubseteq_{int}^{B} Q$, since Q is error free in the internal sense, but P is not. □

Finally, we turn to the pessimistic approach, which has already been discussed in the literature e.g. in [4], and consider only those systems error free that do not have any reachable error states.

Definition 3.27 (Pessimistic Basic Relation)**.** *An* error is reachable *in an EIO P, if StT(P)* $\neq \emptyset$. *For EIOs P and Q with the same signature, we write* $P \sqsubseteq_{act}^{B} Q$*, whenever an error is reachable in P only if an error is reachable in Q. We denote the fully abstract precongruence with respect to* $\sqsubseteq_{act}^{B}$ *and* $|$ *by* $\sqsubseteq_{act}^{c}$.

A pessimistic person might argue that systems with an error should just not be used at all (such a view is presumably taken in [4]), and that it does not make sense to distinguish between erroneous systems, as we will do with $\sqsubseteq_{act}^{c}$ (since this is finer than $\sqsubseteq_{loc}^{c}$, see below). This version of pessimism has the severe disadvantage that parallel composition is not associative in general. Consider P, Q and R and their inputs and outputs as given in Fig. 3.8. $P \mid Q$ has a reachable error, so $(P \mid Q) \mid R$ is not considered – in contrast to $P \mid (Q \mid R)$, since $Q \mid R$ and $P \mid (Q \mid R)$ are completely error free and consist just of the initial state.

It is easy to see that our local error semantics does not suffice to characterize $\sqsubseteq_{act}^{c}$, since it does not differentiate between a missing input and an input leading to an error state as seen in Fig. 3.9. Since $P \not\sqsubseteq_{act}^{B} Q$, we cannot have $P \sqsubseteq_{act}^{c} Q$.

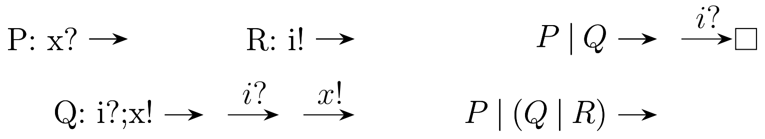

Figure 3.8: counterexample for associativity

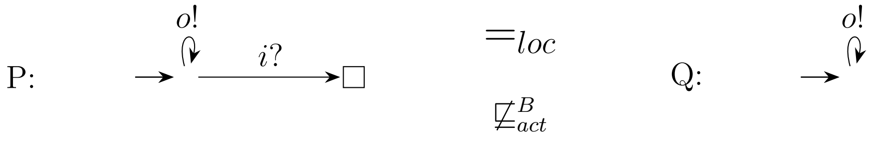

Figure 3.9: $\sqsubseteq_{loc}^{c}$ does not imply $\sqsubseteq_{act}^{B}$

We will now present an extension to the local error refinement that is a precongruence for this pessimistic approach, albeit not the coarsest one. CPT deals with the real errors and is based on pruning of outputs again; MIC additionally considers the missing-input traces, this time without closing under continuation; another subtle point is that both, MIC and L, are flooded with CPT.

Definition 3.28 (Pessimistic Error Semantics). *Let P be an EIO.*

- *The set of* continued pruned traces *of P is:* $CPT(P) = cont(prune(StT(P)))$*;*
- *the set of* flooded missing-input traces *of P is:* $MIC(P) = MIT(P) \cup CPT(P)$*;*
- *the* CPT-flooded language *of P is:* $LCP(P) = L(P) \cup CPT(P)$*.*

For two EIOs P and Q with the same signature, we write $P \sqsubseteq_{act} Q$ *if and only if* $CPT(P) \subseteq CPT(Q)$*,* $MIC(P) \subseteq MIC(Q)$ *and* $LCP(P) \subseteq LCP(Q)$*.*

Again, it can be shown that parallel composition is monotonic w.r.t. $\sqsubseteq_{act}$, i.e. $\sqsubseteq_{act}$ is a precongruence. However, it is not fully abstract regarding | and $\sqsubseteq_{act}^{B}$ as illustrated by the following example. We will also sketch (without proof) how we think one can get the coarsest refinement. The two EIOs P and Q with $I = \{a, b\}$ and $O = \emptyset$, depicted in Fig. 3.10, are not seen as equivalent by our precongruence:

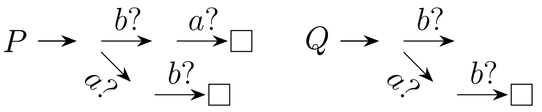

Figure 3.10: First example for coarsest refinement

$Q \sqsubseteq_{act} P$, but $P \not\sqsubseteq_{act} Q$ because ba is contained in $CPT(P)$ but not in $CPT(Q)$. Yet $P \sqsubseteq^c_{act} Q$ holds, since there is no environment U such that $P \mid U \not\sqsubseteq^B_{act} Q \mid U$, cf. Theorem 3.20.

To prove this, we consider all possibilities for an environment U. If U is not error free, then both $P \mid U$ and $Q \mid U$ have an error: since both, P and Q, have no output actions, they cannot prevent U performing a run ending with an error state; they can only cause the error to appear earlier. Hence, we can concentrate on error free environments.

If $a, b \notin O_U$, then no synchronization takes place and both P and Q can reach an error by performing ab autonomously. If $a \in O_U$ and $b \notin O_U$, it might be that U never performs a and neither $P \mid U$ nor $Q \mid U$ can reach an error. If a is performed, then afterwards b can be performed autonomously by P or Q respectively, leading to an error. The case that only $b \in O_U$ is analogous.

Also in the case $a, b \in O_U$, no difference between $P \mid U$ and $Q \mid U$ can be observed, since U not performing one a followed by b or vice versa would prevent all errors; otherwise, an error occurs in both $P \mid U$ and $Q \mid U$. The only slight and unobservable difference appears if U performs ba. Then $P \mid U$ has an inherited and $Q \mid U$ a new error.

This example also highlights the difficulty of characterizing $\sqsubseteq^c_{act}$. It appears that under certain circumstances a missing-input trace (like ba for Q in the example above) has to be added to the set CPT. This appears to be the case if the missing-input trace and some error trace are the same when projecting the missing action away. The next example in Fig. 3.11 illustrates the need for this to be done iteratively. The trace ba should be included in CPT but only because of $ab \in CPT$. Then, one can argue analogously to the above that $cbac$ has to be added as well, but only because ba has been added before.

Hopefully, these considerations have convinced the reader that a characterization of the coarsest refinement will be overly complicated, and that it is not worth the effort to work it out in detail. But although we do not have a characterization of $\sqsubseteq^c_{act}$, we can nevertheless compare it to the local precongruence $\sqsubseteq_{loc}$ using Corollary 3.10; in contrast to the previous section,

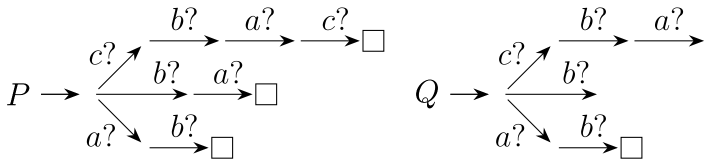

Figure 3.11: Second example for coarsest refinement

we have made the notion of error-freeness much *stricter*, but it turns out that again this leads to a finer precongruence.

Theorem 3.29. *The coarsest pessimistic precongruence $\sqsubseteq^c_{act}$ is strictly finer than the local precongruence $\sqsubseteq_{loc}$, i.e. $\sqsubseteq^c_{act} \subsetneq \sqsubseteq_{loc}$. Hence, this also applies for $\sqsubseteq^c_{act}$.*

Proof. $\sqsubseteq^c_{act} \subseteq \sqsubseteq_{loc}$:
By definition of $\sqsubseteq^c_{act}$, $P \sqsubseteq^c_{act} Q \Rightarrow \forall U : U \mid P \sqsubseteq^B_{act} U \mid Q \Rightarrow$ for all partners U of P, $U \mid P \sqsubseteq^B_{act} U \mid Q$. This is equivalent to $P \sqsubseteq_{loc} Q$ by Corollary 3.21, since $U \mid P$ and $U \mid Q$ are closed, and $\sqsubseteq^B_{act}$ and $\sqsubseteq^B_{loc}$ coincide on closed systems.

$\sqsubseteq^c_{act} \not\supseteq \sqsubseteq_{loc}$:
This follows from the two systems in Fig. 3.9, since $\sqsubseteq^c_{act}$ is finer than $\sqsubseteq^B_{act}$. □

3.5 Conclusion

To study the foundations of interface automata, we have chosen a variant with explicit error states and a standard parallel composition extended according to the characteristic idea: an output that is not expected by the recipient creates an error. To determine an appropriate refinement relation, we started from the basic idea that an error-free specification can only have error-free implementations and then considered the respective coarsest precongruence respecting this basic requirement. We have done this for three variants of error-freeness and characterized or at least approximated the precongruence with an essentially trace-style semantics.

We have started with the optimistic view that errors only count if they are reached locally. Our result here shows that the simulation-style refinement of [27] is unnecessarily strict, but that the pruning integrated into the parallel composition of [27] is justified. Then, we briefly looked at a hyper-optimistic version (only internally reachable errors count) and a pessimistic version (each reachable error has to be avoided). Surprisingly, both variants lead to a stricter precongruence, and both the semantics are also based on the same idea of pruning outputs (justifying it further). Since they are more complicated, one might prefer the local variant for its simplicity.

More intuitively, we believe that it also is based on the right concept. At the heart of interface automata is the emergence of errors, and this relies on the idea that each system controls its outputs and internal actions; so a locally reachable error can indeed not be prevented by the environment. The hyper-optimistic view is less intuitive, but at least it served to show that output pruning does not rely so much on the idea that only locally reachable errors count. Note that the error pairs used to describe the respective fully abstract precongruence might remind one of the failure semantics in the previous chapter; but the meaning of error pairs is very different, and this new idea might also be helpful elsewhere. The pessimistic view has the plausibility of controlling the worst case; but this is actually a misconception: comparing a state where input i is missing with a state where it leads to an error state, we see that both just formulate the same requirement for the environment: the environment must take this state into account and avoid producing i – there is no difference at all. Put another way, input transitions are only taken if the input is provided; for the two states mentioned nothing bad will happen without input i being produced. There are cases, where the pessimistic approach may be more appropriate, for example when studying assemblies or closed systems. However, then all three approaches coincide and the optimistic one has the advantage of simplicity.

A final argument concerns the approach we described at the end of Section 3.3.2; assume we call P better than Q if each *user* (partner) U encounters an error with P only if the same can happen with Q. There are three variants for what it means to encounter an error, but they all agree for closed systems like $P\,|\,U$ and $Q\,|\,U$. Hence, there is only one meaning for "better-than", and this is the precongruence $\sqsubseteq_{loc}$ of the first variant due to Corollary 3.21.

All three of our semantics are language- (i.e. trace-)based; in particular, $\sqsubseteq_{loc}$ is conceptually easy to decide for finite-state EIOs with automata-theoretic methods. It is often argued that simulation-type refinement notions should be used because they are more efficient to check (i.e. in low polynomial time), while language-based notions are usually PSPACE-hard; see e.g. [25]. We believe that one should not mix up the search for a refinement notion that reflects the semantical issues of interest with complexity issues. Once the semantically most adequate refinement is determined, one can still approximate it with a stricter notion that can be checked efficiently. If P is a refinement of Q in the stricter sense, it can safely be used in place of Q. If it is not, one can still decide whether one wants to invest greater effort in order to find out whether the use of P would really destroy some relevant properties. More detailed arguments of this kind are brought forward in [65].

Chapter 4

Modal Interface Automata

4.1 Introduction

We now come to the main contribution of this thesis: the interface theory of *Modal Interface Automata* (MIA).

In this chapter we present MIA and its supported operations. Essentially, MIAs are state machines with disjoint input and output alphabets, as in Interface Automata (cf. Chapter 3), and two transition relations, *may* and *must*, as in disjunctive Modal Transition Systems (dMTS) (cf. Chapter 2). Unlike previous versions of MIA [52, 53] and also unlike other similar theories, we introduce a special *universal state* u capturing arbitrary behaviour. It essentially is the set of error states of EIOs (cf. Chapter 3) compressed into one state. But, in contrast to error states u cannot be removed in general. It can, however, be replaced by different notations for arbitrary behaviour after an input may-transition (cf. [41, 48]). Unlike previous versions of MIA, we allow for multiple (in fact infinitely many) initial states. Intuitively, they are understood disjunctively, similar to the targets of disjunctive must-transitions: each initial state of a refinement has to match at least one initial state of the specification. The reason for this extension, next to a more intuitive and simpler disjunction operator, is the introduction of temporal logics. The technical reasons will be presented later on.

Unlike the previous chapters, the refinement relation we use is simulation based. We have examined the coarsest, most optimal refinement relations for avoiding new errors and deadlocks. However, while being optimal for their respective purposes, they are not as intuitive and not as easily decidable as a simulation type refinements. In particular the modalities of dMTS are lost to a large degree when using trace-based semantics, like the $\mathcal{F}$-semantics, as we have shown.

Thus, while our simulation is finer than the trace based relations we could have used, it still preserves relevant properties like error or deadlock freedom, while capturing modalities neatly and intuitively. Furthermore, it can be decided more efficiently. Last but not least, simulation based refinements are well accepted in the community.

The refinement used here is based on the weak as-refinement (aka observational modal refinement) of dMTS, but it treats inputs slightly differently: When an input must-transition $q \xrightarrow{i} Q'$ is to be matched by a state p, only trailing τ-transitions are allowed: we require $p \xrightarrow{i}\overset{\varepsilon}{\Longrightarrow} P'$, not $p \overset{i}{\Longrightarrow} P'$. The reason for this is error avoidance: If q guarantees immediate acceptance of the signal i, it cannot be refined to some p which would produce an error when receiving i; recall that we require immediate acceptance of signals, as in Chapter 3. Output must-transitions can be matched using leading and/or trailing τ-transitions, i.e. just like with usual weak as-refinement.

The refinement used for MIA in [13] and [20] also forbids using leading τs for matching input may-transitions. Here we lift this restriction. We rightly believed that this would lead to complications with the pruning during parallel composition; however, we overestimated the challenges we described in [13, Remark 9]: While removal of input transitions has to propagate further back, there is no need to propagate it forward as well. Furthermore, pruning during hiding is not a technical necessity. Nevertheless, the latter is more intuitive, than the usual hiding and thus we will explore it as an alternative hiding operator.

Our theory supports a number of operators. It, of course, features a parallel composition operator (without immediate hiding), but also relabelling, restriction (of inputs) and two hiding (of outputs) operators. These we collectively refer to as structural operators[1].

MIA also features the logical operators of conjunction and disjunction. As mentioned before, the former is only possible since we base MIA on dMTS rather than MTS. The construction of disjunction is simpler than for the refinement used in [13], since we use multiple initial states; we will elaborate on this later on.

Following the ideas of [29] and [50] we add temporal logics to MIA – in particular a safety fragment of ACTL. With this our interface theory provides Existential and Universal Next-operators (collectively known as HML-operators), as well as Always- and Unless-operators.

The Existential Next-operator $\langle\alpha\rangle\varphi$ requires the existence of an α-transition with certain behaviour afterwards specified by φ. Similarly, the Uni-

[1] In [13] we also presented a quotienting operator, but since it was mainly the contribution of Fendrich, it is omitted from this thesis.

versal Next-operator $[\alpha]\varphi$ requires φ to be satisfied after any α-transition. To extend expressiveness, we will use sets of labels as parameters instead of α for both HML-operators. The Always-operator $\Box\varphi$ requires φ to be satisfied in any reachable state; we will also parametrize this operator with a set of labels ($\Box_B\varphi$) to require φ only in states reached by actions in B. The Unless-operator $\varphi \mathsf{W} \psi$ intuitively requires φ to hold unless at some point ψ holds at least once. It is an extension of the Always-operator, which therefore will only be a stepping stone that eases understanding of the rather involved Unless-operator. We also parametrize the Unless-operator $\varphi {}_B\mathsf{W}_C \psi$: B, as with the Always-operator restricts requirements to states reached by actions in B. C strengthens the escape requirement: φ now has to hold (in all states reached by actions in B) unless an action in C in performed and ψ holds afterwards. All of these parametrizations are generalizations.

We embed these operators into the framework by giving constructions and showing that satisfaction coincides with refinement. Thus, the operators can freely be applied not only to formulae but to arbitrary MIAs again resulting in MIAs. To use temporal logics, we have to restrict MIAs slightly, requiring initial τ-closure: Each state reachable by internal transitions should also be an initial state. This requirement seems intuitive to us, but is technically only required for our ACTL operators, so we will not require it in the rest of the chapter. Requiring initial τ-closure throughout, would necessitate only a slight adjustment in the definitions and proofs (cf. [20]).

In the context of temporal logics and refinement often a Hennessy-Milner-style characterization is considered desirable. We show that our ACTL logics does not provide such a characterization, but we define a slightly different logic which does. We consider it, however, to be of secondary importance, since, in contrast to ACTL, it cannot be embedded into the theory.

Essentially, all these operators are on an equal footing and can be combined arbitrarily. We can use MIAs as primitives of logical formulae and transitions of MIAs can lead to formulae (technically to MIAs representing formulae).

When specifying different aspects of a system separately, it is often beneficial to concentrate on only some actions of the overall system. For example, when designing a communications interface for a web-service, a designer specifying interactions with a user should not need to concern himself with actions only pertaining to e.g. maintenance.

Since the overall system has to satisfy the aspect specification, one has to define a setting where a refinement might have more actions, i.e. a larger alphabet than the specification Q. Furthermore, for conjunction, one has to deal with operands that have different alphabets. For both, the question is:

what does Q specify regarding *foreign* actions, actions outside its scope? The answer mostly found in literature [13, 53, 59] is that Q allows but disregards foreign actions keeping all its requirements, permissions and prohibitions in place.

From a technical viewpoint, the refinement relation has to be extended to allow for alphabet extension. This can be defined directly or by defining an alphabet extension operator $[Q]_P$ which extends the alphabet of Q to the one of P in a canonical way; then $P \sqsubseteq Q$ can be checked as before. For state based settings, like MIA, such an alphabet extension operator usually means adding some kind of loop: may-loops were argued for in [53], which dealt with a previous incarnation of MIA. In the deterministic, trace-based setting of Modal Interfaces [59], the authors used a mechanism corresponding to may- and must-loops, naming the results weak and strong extensions. However, their theory uses the former for conjunction and the latter for parallel composition. This usage of separate refinement relations for different operators renders their theory incoherent.

In Section 4.2.2, after MIAs have been formally defined, we will discuss two other intuitive ideas on how to deal with foreign actions, other than adding may-loops. We will incorporate flexible mechanism into MIA that allows to specify various intuitions about the impact of foreign actions, and in such a way that different ideas might apply to different states: we introduce special *new actions* ν_I, ν_O acting as placeholders for foreign actions. Refinement without alphabet extension will treat them as it does any other labels of may-transitions. With these a designer can specify which kind of alphabet extension he desires at this state, giving different degrees of importance and different meaning to different states instead of forcing a uniform understanding on all states vis-a-vis foreign actions and other aspect specifications. We believe ours to be a very flexible and elegant solution to the problem, which also generalizes and unifies several previous approaches.

In Chapter 1, we briefly mentioned some previous work concerning combinations of IA and MTS. Assuming the knowledge of previous chapters and some intuition about MIA, we now go into more detail.

The first to examine relations between IA and MTS were Larsen et al. in [42]. They examined the similarities between the two approaches, which already had similar refinement relations: alternating simulations. They provided a translation from IA to MTS: input transitions became must-transitions and output transitions became may-transitions. In addition, to be able to freely add input transitions during refinement (as is allowed in IA), they introduced a dedicated state which allowed for arbitrary behaviour – we came to call this a *true-state*; a missing input of IA became a may-transition to the true-state. A true-state is very similar to the universal state we use in MIA. However, while the intention is the same, there is a key difference between the two concepts: the universal state retains its special property after parallel composition. This is very important for associativity of parallel composition (cf. Modal Interfaces below).

Larsen et al. went on to define IOMTS, a combination of IA and MTS featuring an IA-style refinement and an MTS-style parallel composition. However, their claim of compositionality, or rather its restricted version of independent implementability, was wrong as has been shown in [59].

In the same latter, Raclet et al. present the deterministic setting of *Modal Interfaces (MI)* and attempt to repair the flaw of IOMTS. However, their result fails to show precongruence, as the claim compares two different composition operators. Furthermore, their claim of associativity is flawed. Their treatment of errors (while being trace-based) corresponds to a true-state. Since this, unlike a universal state, loses its special status during parallel composition, the order of compositions is not irrelevant. MIA, as presented here[2], can be seen as a nondeterministic extension of MI. We also correct the associativity flaw using the universal state.

As mentioned above, Raclet et al. also examine two versions of alphabet extension for MI: weak extension corresponds to the addition of may-loops at each state and strong extension to adding must-loops. Their theory uses the former for conjunction and the latter for parallel composition. This usage of separate refinement relations for different operators renders MI an incoherent theory.

To improve on IOMTS and MI, Lüttgen and Vogler developed the first two incarnations of Modal Interface Automata (let us call them *MIA-1* and *MIA-2* respectively) in [51] and [53][3]. Their aim was not only to find a combination of IA and MTS into which IA can be embedded, but also to provide conjunction operators for IA, MTS and their combinations. To achieve this for the setting including modality they introduced and used dMTS instead of MTS, since, as we have seen in Chapter 2, no conjunction is possible for traditional MTS.

The refinement relation used for MIA-1 and MIA-2 is based more on the one for IA rather than MTS. Therefore, there is no forbidding inputs in the MIA-1 setting and there are consequently no *pure input may-transitions*, i.e. input may-transitions without an accompanying must-transition. Inputs are either must, or missing and thus underspecified; in case of the latter an input transition can be introduced arbitrarily. Furthermore, input determinism is

[2] and already the version of [13]

[3] The second paper was co-authored by Fendrich.

required. It should be noted, that MIA-1, as well as IA, IOMTS and MI, (mostly) lack internal transitions.

For MIA-2, the authors introduce internal transitions, in particular allowing for internal disjunctive must-transitions and weak must-transitions. As we mentioned in Chapter 2, this is far from being as straightforward as weak may-transitions.

The authors consider alphabet extension in this setting and illustrate the occurring problems: allowing foreign inputs to knock out the specification is, in general, undesired and the addition of input must-transitions would force specification of undesired inputs. Since no pure input may-transitions are allowed in this setting, the options are exhausted. They conclude that for outputs may-loops are a solution, but in general they leave the problem open.

They go on to examine a pessimistic version of MIA-2 where they can allow for pure input may-transitions and present, for the first time, conjunction and disjunction operators for a nondeterministic pessimistic setting. In this pessimistic setting they also achieve alphabet extension by adding may-loops.

We base the refinement relation of our MIA framework here on the refinement of MTS rather than IA. Thus inputs can be forbidden, pure input may-transitions are possible and we use the universal state to symbolize underspecification. Formalizing underspecification is necessary for the IA-style parallel composition and its pruning, as we have seen in Chapter 3 and will see here. The pure may-transitions allow for an elegant alphabet extension operator. In [13], as mentioned above, we also added may loops to achieve a persistent alphabet extension operator, thus solving the open problem mentioned above. Here, we go even further with the mechanism of new-action labels, as explained above.

In [47] and [48], Luthmann et al. use the third MIA-model, first presented in [12], to examine how modalities can be added to ioco testing (cf. [64]). They use a different notation for essentially the same model: They swap the meanings of a nonexistent input transition and an input transition leading to a universal state. Thus, nonexistent transitions can be added arbitrarily, while forbidden transitions have to lead to a special state.

A similar notational variant was examined by Kühbacher in his Bachelor's thesis [41], where he also removed the universal state. Instead of input transitions leading to the universal state, he annotated each state with a set of inputs, which were allowed to be implemented arbitrarily.

Modal I/O Automata (MIO) [4] by Bauer et al. incorporate quite different design decisions than MIA; these are also different from the ideas of IA. The authors define a pessimistic setting and consider all compositions that can produce an error to be incompatible. We have argued against this in Chapter 3. Furthermore, they examine the effects of weaker notions of compatibility not requiring signals to be accepted immediately.

Thus, while MIO might at first glance look similar to MIA (they both mix modalities with inputs and outputs), they are actually quite different approaches.

The rest of this chapter is structured as follows. In the next section we formally define MIA, weak transition relations and the refinement relation. We also discuss the need for multiple initial states and the ideas behind the new actions used for alphabet extension. We discuss the structural, logical and temporal logic operators in Sect. 4.3, 4.4 and 4.5 respectively. The latter also includes discussions on HML-characterizations of our refinement. In Sect. 4.6 we introduce alphabet extension and extend the refinement and the operators accordingly.

4.2 Modal Interface Automata: The Setting

We begin by formally defining MIAs.

Definition 4.1 (Modal Interface Automata)**.** *A* Modal Interface Automaton *(MIA) is a tuple* $(P, I, O, \longrightarrow, \dashrightarrow, P_0, u)$*, where*

- P *is the set of* states *containing the possibly empty set of* initial states P_0 *and the* universal state u*,*
- $I \subseteq \Sigma$ *and* $O \subseteq \Sigma$ *are disjoint alphabets of* input *and* output actions*;* $\mathcal{A} =_{df} I \cup O$ *is called the* alphabet *and we define* $\mathcal{A}_\tau =_{df} \mathcal{A} \cup \{\tau\}$*,*
- $\longrightarrow \subseteq P \times \mathcal{A}_\tau \times (\mathfrak{P}(P) \setminus \{\emptyset\})$ *is the* disjunctive must-transition *relation, with* $\mathfrak{P}(P)$ *being the powerset of* P*,*
- $\dashrightarrow \subseteq P \times (\mathcal{A}_\tau \cup \{\nu_I, \nu_O\}) \times P$ *is the* may-transition *relation.*

We require the following conditions:

1. *For all* $\alpha \in \mathcal{A}_\tau$*,* $p \xrightarrow{\alpha} P'$ *implies* $\forall p' {\in} P'.\, p \overset{\alpha}{\dashrightarrow} p'$ (syntactic consistency)*,*
2. u *appears in transitions only as the target state of input may-transitions* (sink condition)*.*

Condition 1 states that whatever is required should be allowed; this syntactic consistency is a natural and standard condition (see [45]). Regarding Condition 2, recall that we use u to express that an input is optional in some state, with arbitrary behaviour afterwards. Note that there might very well be ordinary states without any outgoing transitions for some input i; in other words, a MIA does not have to be *input-enabled* like the IO-Automata in [54].

Observe that, as with dMTS in Chapter 2, our disjunctive must-transitions have a single label, in contrast to Disjunctive MTS (DMTS) [44]. In the context of MTS, this is sufficient for intuitively and compactly representing (a) conjunction, as shown in [52], and (b) parallel composition, which would otherwise require an indirect definition via, e.g., Acceptance Automata [1, 58], as suggested in [6]. Our restriction to single labels does not seem to restrict the expressible sets of implementations, i.e., τ-free labelled transition systems (LTS), as studied by Fecher and Schmidt [33] and Beneš et al. [6], due to allowing arbitrary sets of initial states in MIAs. The main reason for using sets of initial states instead of single initial states (as in previous versions of MIA) are temporal operators. We will go into more detail after the definition of refinement.

We define the *τ-closure* of a set of states P' as $\{p' \mid \exists p \in P'.\, p \overset{\varepsilon}{=\!\Rightarrow} p'\}$. We write p_0 for P_0 if $P_0 = \{p_0\}$ and $P[P']$ for the MIA P with the τ-closure of P' as the set of initial states instead of P_0 (or $P[p']$ if $P' = \{p'\}$).

We now define *weak* must- and may-transition relations that abstract from transitions labelled by τ, as is needed for MIA refinement. It is an alternative, more general definition than the one presented in [53]. In [53] and [12], we have failed to notice that our conjunction operator applied to infinite MIAs can result in infinite target sets of disjunctive must-transitions (Rule (Must) in Def. 4.33; see p. 141 for an example of this). Consequently, we now allow such target sets in Def. 4.1 above. As a consequence, we modify also the definition of weak transitions; in order to derive adequate weak must-transitions, they are built up back-to-front.

Definition 4.2 (Weak Transition Relations). *For an arbitrary MIA P, we define* weak *must- and may-transition relations, $\Longrightarrow$ and $=\!\Rightarrow$ respectively, as the smallest relations satisfying the following conditions, where we write $P' \overset{\hat{\alpha}}{\Longrightarrow} P''$ as a shorthand for $\forall p \in P' \, \exists P_p .\, p \overset{\hat{\alpha}}{\Longrightarrow} P_p$ and $P'' = \bigcup_{p \in P'} P_p$:*

1. *$p \overset{\varepsilon}{\Longrightarrow} \{p\}$ for all $p \in P$,*

2. *$p \xrightarrow{\tau} P'$ and $P' \overset{\hat{\alpha}}{\Longrightarrow} P''$ implies $p \overset{\hat{\alpha}}{\Longrightarrow} P''$,*

3. *$p \xrightarrow{a} P'$ and $P' \overset{\varepsilon}{\Longrightarrow} P''$ implies $p \overset{a}{\Longrightarrow} P''$,*

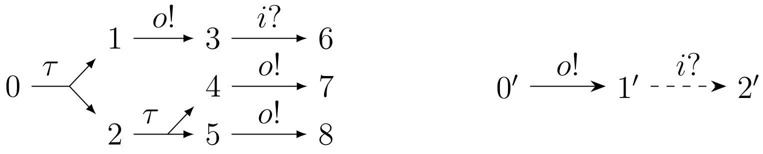

Figure 4.1: Examples of weak transitions and refinement.

4. $p \overset{\varepsilon}{=\!\Rightarrow} p$,

5. $p \overset{\varepsilon}{=\!\Rightarrow} p'' \overset{\tau}{\dashrightarrow} p'$ *implies* $p \overset{\varepsilon}{=\!\Rightarrow} p'$,

6. $p \overset{\varepsilon}{=\!\Rightarrow} p'' \overset{\alpha}{\dashrightarrow} p''' \overset{\varepsilon}{=\!\Rightarrow} p'$ *implies* $p \overset{\alpha}{=\!\Rightarrow} p'$.

We write $\xrightarrow{a}\xRightarrow{\varepsilon}$ *for transitions that are built up according to Case 3 and call them* trailing-weak *must-transitions. Similarly,* $\overset{a}{\dashrightarrow}\overset{\varepsilon}{=\!\Rightarrow}$ *stands for trailing-weak may-transitions.*

For examples of weak transitions, consider the MIA on the left-hand side of Fig. 4.1. By applying Def. 4.2.1 and 4.2.2, any τ-transition is also a weak ε-transition. Similarly, every a-transition is also a weak a-transition by Def. 4.2.1 and 4.2.3. Transition $2 \xrightarrow{\tau} \{4,5\}$ can be extended to $2 \xRightarrow{o} \{7,8\}$ by applying Def. 4.2.2. Hence, $0 \xrightarrow{\tau} \{1,2\}$ extends to $0 \xRightarrow{o} \{3,7,8\}$. Observe that our weak must-transitions correspond to standard weak transitions of LTS in the case that only must-transitions with a single target state are used.

When reasoning about weak must-transitions, e.g., in Lems. 4.3 and 4.23 below, we consider a derivation of a weak must-transition according to Def. 4.2 as a tree and each node as being larger than the nodes from which it is derived. Although the tree might be infinitely branching, larger-than is a Noetherian partial order. Hence, one can apply (Noetherian, transfinite) induction on the derivation of a weak must-transition.

Lemma 4.3. *Consider an arbitrary MIA P.*

(a) $p \xRightarrow{\varepsilon} \bar{P}$ *and* $\bar{P} \xRightarrow{\hat{\alpha}} P'$ *implies* $p \xRightarrow{\hat{\alpha}} P'$,

(b) $p \xRightarrow{a} \bar{P}$ *and* $\bar{P} \xRightarrow{\varepsilon} P'$ *implies* $p \xRightarrow{a} P'$.

Proof. (a) We proceed by induction on the definition of $p \xRightarrow{\varepsilon} \bar{P}$. Regarding Def. 4.2.1, the claim is trivial. Now assume that $p \xRightarrow{\varepsilon} \bar{P}$ is due to Def. 4.2.2, i.e., we have $p \xrightarrow{\tau} P''$ and, for each $p'' \in P''$, there is some $\bar{P}_{p''}$ with $p'' \xRightarrow{\varepsilon} \bar{P}_{p''}$ and $\bar{P} = \bigcup_{p'' \in P''} \bar{P}_{p''}$. By premise $\bar{P} \xRightarrow{\hat{\alpha}} P'$, some $P_{\bar{p}}$ exists for each $\bar{p} \in \bar{P}$

such that $\bar{p} \overset{\hat{\alpha}}{\Longrightarrow} P_{\bar{p}}$ and $P' = \bigcup_{\bar{p}\in\bar{P}} P_{\bar{p}}$. For each $p'' \in P''$, $P'_{p''} =_{df} \bigcup_{\bar{p}\in\bar{P}_{p''}} P_{\bar{p}}$ satisfies $\bar{P}_{p''} \overset{\hat{\alpha}}{\Longrightarrow} P'_{p''}$ and, by induction hypothesis, $p'' \overset{\hat{\alpha}}{\Longrightarrow} P'_{p''}$. By Def. 4.2.2, this implies $p \overset{\hat{\alpha}}{\Longrightarrow} \bigcup_{p''\in P''} P'_{p''}$. This target set is clearly the union of some $P_{\bar{p}}$ with $\bar{p} \in \bar{P}$; moreover, each $\bar{p} \in \bar{P}$ is in some $\bar{P}_{p''}$, and the target set covers $P'_{p''} \supseteq P_{\bar{p}}$. Hence, the target set is P' and we are done. The case of Def. 4.2.3 does not apply.

(b) Similarly to (a), we apply induction on the derivation of $p \overset{a}{\Longrightarrow} \bar{P}$. Case 1 of Def. 4.2 does not apply. Case 2 is shown as above, observing that we need $p'' \overset{a}{\Longrightarrow}$ twice, $\bar{p} \overset{\varepsilon}{\Longrightarrow}$, $\bar{P}_{p''} \overset{\varepsilon}{\Longrightarrow}$ and $p \overset{a}{\Longrightarrow}$. Case 3 is also similar to Case 2 in (a), except that all weak transitions not originating in p are labelled ε, and we use (a) instead of the induction hypothesis. □

Now we define our simulation-based refinement relation. It is a weak alternating simulation that is conceptually similar to the observational modal refinement found, e.g., in [40]. A notable aspect, originating from IA [27], is that inputs-must-transitions have to be matched immediately, i.e., only trailing τs are allowed. Intuitively, this is because of the requirement that a signal sent from one system must be received immediately; otherwise, it is considered an error (communication mismatch). Since one wishes not to introduce new errors during refinement, a refined system must immediately provide all specified inputs.

We treat the universal state u as completely underspecified, i.e., any state refines it; this is only possible since u is not an ordinary state. Recall that we have an i-may-transition from some state p to u to express that, like in the IA-approach, p can be refined by a state with an i-transition followed by arbitrary behaviour. We define our refinement preorder for MIAs with common input and output alphabets here and relax this restriction in Sect. 4.6.

The new actions ν_I and ν_O are, for now, treated like normal inputs and outputs. They will play a special role when we extend this definition in Def. 4.63. For convenience (use in definitions and proofs), we define the refinement in two steps.

Definition 4.4 (MIA Refinement)**.** *Let P, Q be MIAs with common input/output alphabets. A relation $\mathcal{R} \subseteq P \times Q$ is a* MIA-refinement relation *if for all $(p, q) \in \mathcal{R}$ with $q \neq u_Q$:*

(i) $p \neq u_P$,

(ii) $q \overset{i}{\longrightarrow} Q'$ *implies*
$$\exists P'.\, p \overset{i}{\longrightarrow}\overset{\varepsilon}{\Longrightarrow} P' \text{ and } \forall p'{\in}P'\ \exists q'{\in}Q'.\, (p', q') \in \mathcal{R},$$

(iii) $q \xrightarrow{\omega} Q'$ *implies*

$$\exists P'.\, p \overset{\hat{\omega}}{\Longrightarrow} P' \text{ and } \forall p'{\in}P'\ \exists q'{\in}Q'.\, (p', q') \in \mathcal{R},$$

(iv) $p \overset{\alpha}{\dashrightarrow} p'$ *implies*

$$\exists q'.\, q \overset{\hat{\alpha}}{=\!\!\Rightarrow} q' \text{ and } (p', q') \in \mathcal{R}.$$

We write $p \sqsubseteq q$ *and say that* p MIA-refines q *(or that* p *matches* q*) if there exists a MIA-refinement relation* $\mathcal{R}$ *such that* $(p, q) \in \mathcal{R}$*, and we let* $p \sqsupseteq\!\sqsubseteq q$ *stand for* $p \sqsubseteq q$ *and* $q \sqsubseteq p$ *and call it* MIA-equivalence*.*

We lift the definition to MIAs by defining $p \sqsubseteq Q$ *if* $\exists q_0 \in Q_0.\, p \sqsubseteq q_0$ *and* $P \sqsubseteq Q$ *if* $\forall p_0 \in P_0.\, p_0 \sqsubseteq Q$*. We call* P MIA-equivalent *to* Q *(*$P \sqsupseteq\!\sqsubseteq Q$*) if* $P \sqsubseteq Q$ *and* $Q \sqsubseteq P$*.*

Item (i) ensures that the universal state does not refine any other state. Intuitively, Item (iv) says that $p \overset{\alpha}{\dashrightarrow} p'$ is allowed by q, since a matching behaviour is specified in Q. As far as can be observed, $q \overset{\hat{\alpha}}{=\!\!\Rightarrow} q'$ shows the same behaviour, but allows additional unobservable τ-transitions. In case $\alpha = \tau$, the standard usage of $\hat{\alpha}$ allows that the matching behaviour might be empty and $q = q'$. Note that this also covers new actions.

Conversely, Item (iii) ensures that $q \xrightarrow{\alpha} Q'$ is implemented by p. The $\forall$-quantified formula ensures that the choice of target states in Q' is narrowed down by the matching P'. Item (ii) deals with inputs in a similar way, except that only trailing τs are allowed. This fits to the intuition behind parallel composition in [12, 25]: whenever a signal i is sent to a component Q in state q, a communication mismatch occurs if q cannot accept it immediately. An i-must-transition originating from q guarantees that the signal will be accepted, i.e. it is safe for another component to send it. Since refinement should not introduce new communication mismatches, any refinement p of q also has to immediately accept the signal, i.e. p needs to have an outgoing i-must-transition.

An example of a refinement can be found in Fig. 4.1, where the left MIA refines the right one due to the refinement relation $\{(0, 0'), (1, 0'), (2, 0'), (4, 0'), (5, 0'), (3, 1'), (7, 1'), (8, 1'), (6, 2')\}$. Observe how the refined states 3 and 7 (and 8) of state $1'$ implement the outgoing $i?$-may-transition differently.

For another example, consider the weak transition $p_0 \overset{o}{\Longrightarrow} P' =_{df} \{p'_1, p'_2, \ldots\}$ of 'finite but unbounded depth' in Fig. 4.2, which arises from our back-to-front definition (cf. Def. 4.2). This weak transition is intuitively justified, since each target of the disjunctive τ-must-transition guarantees o. Technically, each p_i refines q_1 and, hence, p_0 refines q_0 according to Def. 4.4. Therefore, $p_0 \overset{o}{\Longrightarrow} P'$ is needed to make Prop. 4.5 (iii) below true for $q_0 \overset{o}{\Longrightarrow} q'_1$.

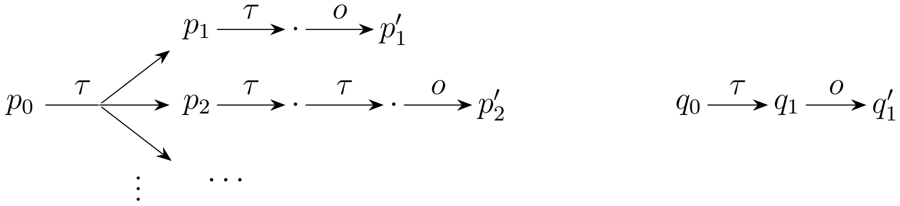

Figure 4.2: Example of refining a weak transition.

As we show next, Lem. 4.3 allows us to replace the transition in the premises of (ii) and (iii) above by a trailing weak and a weak one, respectively; the analogous replacement in (iv) is standard. This result is needed for proving that $\sqsubseteq$ is a preorder.

Proposition 4.5. *Let $\mathcal{R} \subseteq P \times Q$ be a MIA-refinement relation for MIAs P and Q, and let $(p, q) \in \mathcal{R}$ with $q \neq u_Q$. Then, the following generalizations of Def. 4.4(ii)–(iv) hold:*

(ii) $q \xrightarrow{i}\overset{\varepsilon}{\Longrightarrow} Q'$ implies $\exists P'.\, p \xrightarrow{i}\overset{\varepsilon}{\Longrightarrow} P'$ and $\forall p'{\in}P'\,\exists q'{\in}Q'.\,(p', q') \in \mathcal{R}$,

(iii) $q \overset{\hat{\omega}}{\Longrightarrow} Q'$ implies $\exists P'.\, p \overset{\hat{\omega}}{\Longrightarrow} P'$ and $\forall p'{\in}P'\,\exists q'{\in}Q'.\,(p', q') \in \mathcal{R}$,

(iv) $p \overset{\hat{\alpha}}{\Longrightarrow} p'$ implies $\exists q'.\, q \overset{\hat{\alpha}}{\Longrightarrow} q'$ and $(p', q') \in \mathcal{R}$.

Proof. The proof of Part (iv) is standard; the proof of Part (ii) is very similar to that of Part (iii), although the third case below is not relevant for Part (ii); thus, we focus on proving Part (iii) concerning weak disjunctive transitions. We proceed by induction on the definition of $q \overset{\hat{\omega}}{\Longrightarrow} Q'$:

- Let $\omega = \tau$ and $Q' = \{q\}$. Then, we choose $P' =_{df} \{p\}$.

- Let $q \xrightarrow{\tau} \bar{Q}$ and $\forall \bar{q} \in \bar{Q}\, \exists Q_{\bar{q}}.\, \bar{q} \overset{\hat{\omega}}{\Longrightarrow} Q_{\bar{q}}$ with $Q' = \bigcup_{\bar{q}\in\bar{Q}} Q_{\bar{q}}$ according to Def. 4.2.2. By assumption, a weak transition $p \overset{\varepsilon}{\Longrightarrow} \bar{P}$ with $\forall \bar{p} \in \bar{P}$ $\exists \bar{q} \in \bar{Q}.\, (\bar{p}, \bar{q}) \in \mathcal{R}$ exists. Choosing for each $\bar{p} \in \bar{P}$ a suitable $\bar{q}$, we get some $P_{\bar{p}}$ such that $\bar{p} \overset{\hat{\omega}}{\Longrightarrow} P_{\bar{P}}$ and $\forall p' \in P_{\bar{p}}\, \exists q' \in Q_{\bar{q}}.\, (p', q') \in \mathcal{R}$ by induction hypothesis. By Lem. 4.3(a), we obtain $p \overset{\hat{\omega}}{\Longrightarrow} P' =_{df} \bigcup_{\bar{p}\in\bar{P}} P_{\bar{p}}$.

- Let $q \overset{\hat{\omega}}{\Longrightarrow} Q'$ due to Def. 4.2.3, i.e., $\hat{\omega} = o$, $q \xrightarrow{o} \bar{Q}$, $\forall \bar{q} \in \bar{Q}.\, \bar{q} \overset{\varepsilon}{\Longrightarrow} Q_{\bar{q}}$ and $Q' = \bigcup_{\bar{q}\in\bar{Q}} Q_{\bar{q}}$. The proof then proceeds as in the previous case, using Lem. 4.3(b). □

Corollary 4.6. *MIA refinement $\sqsubseteq$ is a preorder and the largest MIA-refinement relation.*

Proof. Reflexivity immediately follows from the fact that the identity relation on states is a MIA-refinement relation. For transitivity one shows that the composition of two MIA-refinement relations is again a MIA-refinement relation, using Prop. 4.5 and following the lines of [57]. The second claim follows since MIA-refinement relations are easily seen to be closed under union. □

4.2.1 MIA-refinement with multiple initial states

Traditionally, in a setting with single initial states (p_0 or q_0 respectively) one defines that $P \sqsubseteq Q$ if $p_0 \sqsubseteq q_0$. In the context of the HML-operators $[a]$ and $\langle a \rangle$, we want to specify systems that after an action a behave like a MIA Q. We expect this specification to be obtained from Q by adding some a-may-transition targeting the initial state. This might fail with the traditional definition of $P \sqsubseteq Q$ if the initial state of Q has a must-transition in conflict with a τ-may-transition.

As an example, consider MIAs Q and Q' in Fig. 4.3. One would expect Q' to behave like Q after performing a. However, Q' can be refined by P', because p'_1 can be matched with q'_2. This way, after a the b-must-transition is completely ignored. However no refinement of Q can do this.

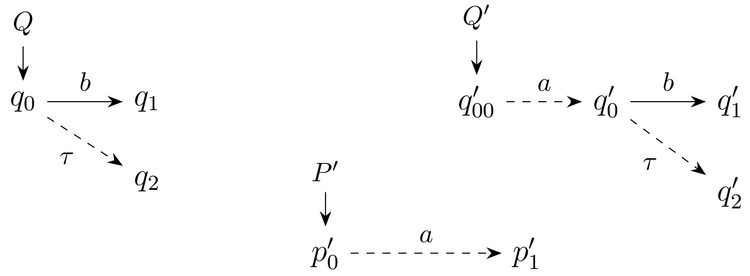

Figure 4.3: Treatment of initial states during refinement.

A way to solve this in a setting with single initial states is to define an *initially loose MIA-refinement* where $P \sqsubseteq_L Q$ if $p_0 \sqsubseteq q$ for some q with $q_0 \overset{\varepsilon}{\Longrightarrow} q$. In our example, $P'[p'_1] \sqsubseteq_L Q$, since $p'_1 \sqsubseteq q_2$. However, with this refinement, we cannot define the disjunction of two MIAs with the characteristic property of disjunction, namely $P \vee Q \sqsubseteq_L R$ iff $P \sqsubseteq_L R$ and $Q \sqsubseteq_L R$ for any MIA R:

Consider the MIAs in Fig. 4.4; for this paragraph, ignore the MIA $P \vee Q$ shown. Since $P \sqsubseteq_L R$ and $Q \sqsubseteq_L R$ by $p_0 \sqsubseteq r_p$ and $q_0 \sqsubseteq r_q$, any MIA $P \vee Q$ would have to refine R, i.e. the initial state pq_0 of $P \vee Q$ would have to match r_0, r_p or r_q. If it matched r_p, it would not allow o', also not from a state pq' with $pq_0 \overset{\varepsilon}{\Longrightarrow} pq'$ (since none of these states can have an o'-may-transition); thus, Q would not refine $P \vee Q$, contradicting the characteristic property for $R \sqsubseteq_L P \vee Q$. Analogously, pq_0 cannot match r_q. Finally, pq_0 cannot match r_0 either, since then a MIA consisting of a single $\overline{o}$-must-transition would refine $P \vee Q$, which would be absurd.

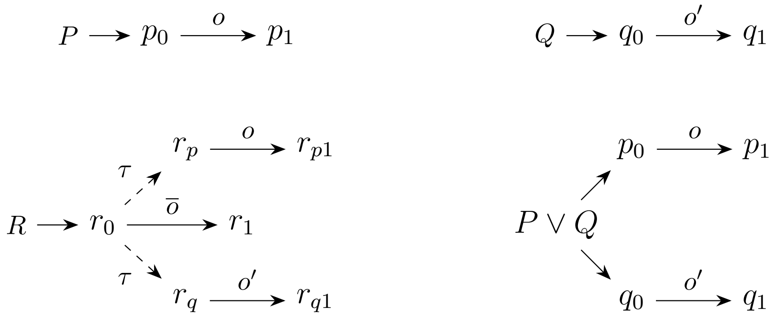

Figure 4.4: Disjunction problem.

Usually it is easy to define disjunction for systems with several initial states, as in Def. 4.1 (cf. e.g. [6]). If we define $P \sqsubseteq'_L Q$ by $\forall p_0 \in P_0 \; \exists q_0 \in Q_0, q \in Q. \, (q_0 \overset{\varepsilon}{\Longrightarrow} q$ and $p_0 \sqsubseteq q)$, then the disjunction $P \vee Q$ can be defined as the disjoint union of the states, initial states and transitions of P and Q, barring the universal states, which have to be merged into one (cf. $P \vee Q$ in Fig. 4.4 and the formal definition below).

This refinement works fine for our purposes; but it is technically easier to avoid weak transitions, so we prefer an alternative (but equivalent) definition. We allow for several initial states, but require the set of initial states to be closed under τ-may-transitions. However, this *initial τ-closure* will only be necessary for the temporal operators presented in Sect. 4.5. In the rest we will simply allow for multiple initial states.

Note that this refinement coincides with the one presented in [12] on the domain of MIAs where both are applicable. This domain consists of MIAs with single initial states that lack outgoing τ-transitions. We identify two further special types of MIAs:

Definition 4.7 (Universal and Empty MIAs)**.** *A MIA P is called* empty *if it has no initial state ($P_0 = \emptyset$). A MIA P is called* universal *if it has its universal state among its initial states ($u \in P_0$).*

It is easy to see that all empty MIAs are equivalent, as are all universal ones. We will use $\mathfrak{F} = (\{u\}, I, O, \emptyset, \emptyset, \emptyset, u)$ and $\mathfrak{U} = (\{u\}, I, O, \emptyset, \emptyset, \{u\}, u)$ as representatives for these classes.

An empty MIA stands for logical inconsistency and arises from the conjunction of two logically inconsistent (i.e. contradictory) MIAs. In our logic, $\mathfrak{F}$ will play the role of false. A universal MIA arises from the parallel composition of incompatible systems, i.e. it represents an immediate communication mismatch. We consider such a mismatch to be erroneous and fatal. Thus, any specification is better than, i.e. a refinement of $\mathfrak{U}$. Nevertheless, $\mathfrak{U}$ will not play the role of true for reasons detailed at the beginning of Sect. 4.5.

4.2.2 Aspect-Oriented Specification

As mentioned above, the main question for aspect-oriented specification is: given an aspect specification Q that is only concerned with some actions of the overall specification, what does it specify regarding foreign actions? This is relevant when checking whether some overall implementation P with a larger alphabet satisfies Q, i.e. whether $P \sqsubseteq Q$. To encompass such cases, the notion of refinement is extended in [13] and [59] by defining an *alphabet extension operator* $[Q]_P$. It extends the alphabet of Q to that of P while adding transitions for the new actions; then $P \sqsubseteq [Q]_P$ is required. With this, the alphabet extension of Q shows explicitly, what requirements Q has regarding foreign actions. It is also possible to define an extended version of refinement directly. This was, in part, done in [27], where refinement allowed for alphabet modification: inputs could be added and outputs could be removed during refinement. We will present both variants for the new alphabet extension later on.

There are several intuitive possibilities what an aspect specification could require regarding foreign actions, each suited for different circumstances. We will discuss three possibilities now: persistent, knockout and forbidding. The first of these is the one usually found in the literature, while the other two are rare or do not appear at all (to the best of our knowledge). Since we aim for strong expressiveness, we will eventually propose a flexible solution that combines the advantages and possibilities of all three concepts.

We will also consider technical aspects of each version, in particular, how it can be integrated into our existing notion of refinement derived from [13]. Since this is simulation based, each state of a refinement P has to be matched by an appropriate state of its specification Q, such that all requirements and prohibitions are preserved with appropriate future behaviour

The version of alphabet extension that has been used in [13] (and in [59] for a trace-based setting) is *persistent alphabet extension.* Intuitively, a MIA allows but disregards foreign actions keeping all its requirements, permissions and prohibitions in place. Technically, a state p refining q may perform an action foreign to q reaching a state p', but p' must essentially match q again.

This notion is useful e.g. for safety properties that should always hold, no matter what actions occur in the overall system. As an example consider the specification P in Fig. 4.5 with alphabet $\mathcal{A} = \{in, out\}$. It only allows *in* and *out* to alternate starting with *in.* One could also look at it as initially prohibiting *out* and, after *in*, prohibiting *in.* Under persistent alphabet extension it allows the use of any foreign action without lifting these prohibitions.

Thus R (Fig. 4.5) is a refinement of P: r_0 matches p_0. After the initial *in* transition r_1 matches p_1. So must r_2, since the foreign a-transition between them is ignored by P. R' however is not a refinement of P, as r_1' would have to match p_1, which prohibits *in*-transitions.

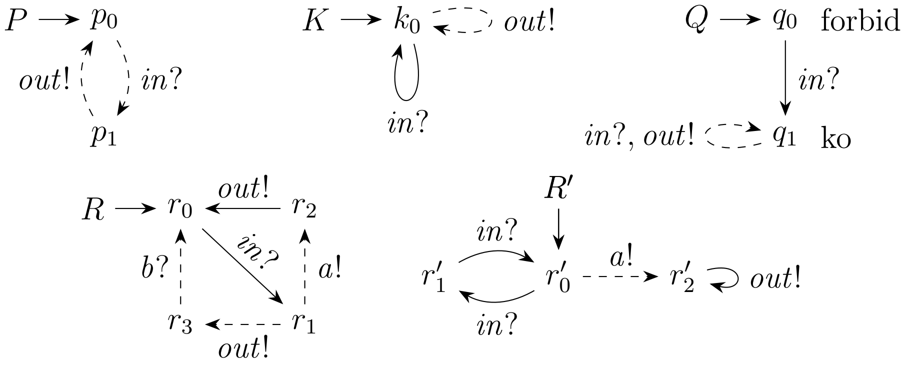

Figure 4.5: Examples of alphabet extensions: may/must-transitions are denoted by dashed/solid arrows; in figures, inputs/outputs are marked by ?/! after the label.

Another possibility is that after an unforeseen action, the specification can be ignored, i.e. foreign actions lift all requirements of the specification. Since a foreign action effectively knocks out the specification, we speak of *knockout alphabet extension.* It appears as half of the alphabet modification in [27], where foreign inputs can be added arbitrarily; outputs can only be removed since this fits the setting more easily. Technically, any transition in the refinement labelled with a foreign input need not be matched at all, thus allowing arbitrary subsequent behaviour

Knockout extension can be used to specify properties like '*in* should be enabled as long as only actions *in* and *out* occur', without having to list all possible (and maybe unknown) actions that can lift the requirement. K

in Fig. 4.5 describes this property and R' satisfies it, i.e. R' refines K: r'_0 and r'_1 both match k_0, since they enable *in*. The a-transition, however need not be matched in K and thus r'_2 can prohibit *in*-transitions. R however does not refine K, since r_1 would be required to match k_0, but lacks an *in*-must-transition.

Both versions are reasonable and we can imagine a setting where one would like to combine them, e.g. to require persistent behaviour for one communication channel (*in* and *out*) and knockout behaviour for another channel (*in'* and *out'*). To achieve this, one would introduce state predicates that would allow a designer to specify a state as persistent or as one that can be knocked out.

Continuing along this line of thinking, one could desire additional choices, for example to completely prohibit all foreign actions. A setting where all states are understood as *forbidding* would be rather useless: while technically one could extend the alphabet of a MIA, it would be impossible to use these new actions. However, in combination with a knockout-predicate, forbidding can be used to specify that a system should not do anything until it receives the *in*-signal. This is depicted as Q in Fig. 4.5, with 'forbid' and 'ko' denoting the state predicates. R satisfies this requirement, as it initially only allows for an *in*-transition. However, R' does not satisfy Q, since an a-transition is allowed in r'_0, but forbidden in q_0 as a foreign action.

In this chapter, we propose an even more flexible definition of alphabet extension: instead of using state predicates, we treat foreign actions explicitly with special ν-transitions, differentiating between new input (ν_I) and new output (ν_O) transitions. This will allow us to model all three presented versions of alphabet extension and more. Essentially, P in the persistent, K in the knockout and Q in the combined setting would look like P', K' and Q' in Fig. 4.6. We will go into this in further detail and present a more elaborate in Sect. 4.6.4 after the technical part.

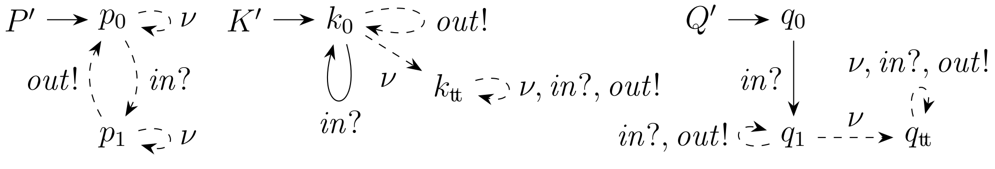

Figure 4.6: Alphabet extension with ν standing for ν_I and ν_O.

4.3 Structural Operators – Parallel Composition

Interface Automata (IA) [26, 27] are equipped with an interleaving parallel operator, where an action occurring as an input in one interface is synchronized with the same action occurring as an output in some other interface; the synchronized action is hidden, i.e., labelled by τ. Since our work builds upon Modal Interfaces (MI) [59] we instead consider here a parallel composition, where the synchronization of an interface's output action involves all concurrently running interfaces that have the action as input. Moreover, we include a separate operator for hiding outputs (cf. [54]). This properly generalises the binary communication of IA to multicast in MIA.

4.3.1 Parallel Composition

We present a parallel operator $\|$ on MIA in the same way as we did in [52, 53], except that common actions are not hidden immediately. Parallel composition is defined in two stages, similarly as in IA. First, a standard product $\otimes$ between two MIAs is introduced. Then, errors are identified, i.e., states where an output is not matched by an appropriate input, and all states from which reaching an error cannot be prevented are *pruned*, i.e., removed.

Recall that, according to our intuition, outputs are under the control of one component. Therefore we exclude MIAs sharing output actions from parallel composition.

Definition 4.8 (Parallel Product)**.** *MIAs P_1, P_2 are* composable *if* $O_1 \cap O_2 = \emptyset$*. For such MIAs we define the* product $P_1 \otimes P_2 = ((P_1 \times P_2) \cup \{u_{12}\}, I, O, \longrightarrow, \dashrightarrow, P_{01} \times P_{02}, u_{12})$*, where u_{12} is a fresh state, $I =_{df} (I_1 \cup I_2) \setminus (O_1 \cup O_2)$ and $O =_{df} O_1 \cup O_2$, and where $\longrightarrow$ and $\dashrightarrow$ are the least relations satisfying the following rules:*

(PMust1) $(p_1, p_2) \xrightarrow{\alpha} P_1' \times \{p_2\}$ *if* $p_1 \xrightarrow{\alpha} P_1'$ *and* $\alpha \notin A_2$
(PMust2) $(p_1, p_2) \xrightarrow{\alpha} \{p_1\} \times P_2'$ *if* $p_2 \xrightarrow{\alpha} P_2'$ *and* $\alpha \notin A_1$
(PMust3) $(p_1, p_2) \xrightarrow{a} P_1' \times P_2'$ *if* $p_1 \xrightarrow{a} P_1'$, $p_2 \xrightarrow{a} P_2'$ *and* $a \in A_1 \cap A_2$
(PMay1) $(p_1, p_2) \overset{\alpha}{\dashrightarrow} (p_1', p_2)$ *if* $p_1 \overset{\alpha}{\dashrightarrow} p_1'$ *and* $\alpha \notin A_2$
(PMay2) $(p_1, p_2) \overset{\alpha}{\dashrightarrow} (p_1, p_2')$ *if* $p_2 \overset{\alpha}{\dashrightarrow} p_2'$ *and* $\alpha \notin A_1$
(PMay3) $(p_1, p_2) \overset{a}{\dashrightarrow} (p_1', p_2')$ *if* $p_1 \overset{a}{\dashrightarrow} p_1'$, $p_2 \overset{a}{\dashrightarrow} p_2'$ *and* $a \in A_1 \cap A_2$

Note that ν_I and ν_O are treated by rules (PMay1) and (PMay2).
From the parallel product, parallel composition is obtained by pruning, i.e., one removes errors and states leading up to them via local actions, so called *illegal* states. This also cuts all input transitions leading to an illegal state.

In [16] we showed that de Alfaro and Henzinger have defined pruning in an inappropriate way in [25], such that associativity is violated. We remedied this by cutting not only an i-transition from some state p to an illegal state, but also all other i-transitions from p. However, since we allow for leading τ-transitions in the refinement of input-may-transitions, we also have to cut i-transitions from states that can reach p via τ-transitions. We discuss this further after the definition. Not only did we prove that this is correct, the solution is also intuitive since, this way, p describes the requirement that a helpful environment must not produce input i. This requirement is described in input-deterministic settings like [27] without any remedy.

Now, in [25, 16], p can be refined by a state with an i-transition and arbitrary behaviour afterwards. As explained above, we express this by introducing an i-may-transition to the universal state. This construction is necessary to achieve compositionality and associativity for parallel composition; see Fig. 10 in [52] for the compositionality flaw in IOMTS [42] and Fig. 4.8 for the associativity problem in MI [59], respectively.

Definition 4.9 (Parallel Composition)**.** *Given a parallel product $P_1 \otimes P_2$, a state (p_1, p_2) is a* new error *if there is some $a \in A_1 \cap A_2$ such that (a) $a \in O_1$, $p_1 \overset{a}{\dashrightarrow}$ and $p_2 \overset{a}{\not\longrightarrow}$, or (b) $a \in O_2$, $p_2 \overset{a}{\dashrightarrow}$ and $p_1 \overset{a}{\not\longrightarrow}$. It is an* inherited error *if one of its components is a universal state, i.e., if it is of the form (u_1, p_2) or (p_1, u_2).*

We define the set $E \subseteq P_1 \times P_2$ of illegal *states as the least set such that $(p_1, p_2) \in E$ if (i) (p_1, p_2) is a new or inherited error or (ii) $(p_1, p_2) \overset{\omega}{\dashrightarrow} (p_1', p_2')$ and $(p_1', p_2') \in E$.*

The parallel composition *$P_1 \parallel P_2$ is obtained from $P_1 \otimes P_2$ by* pruning *illegal states as follows: first, if there is a state $(p_1, p_2) \notin E$ with $(p_1, p_2) \overset{i}{\dashrightarrow} (p_1', p_2') \in E$ for some $i \in I \cup \{\nu_I\}$, then all must- and may-transitions labelled i and starting at a state $(\overline{p_1}, \overline{p_2})$ with $(\overline{p_1}, \overline{p_2}) \overset{\varepsilon}{\Longrightarrow} (p_1, p_2)$ are removed, and a single transition $(\overline{p_1}, \overline{p_2}) \overset{i}{\dashrightarrow} u_{12}$ is added to each. Second, if an initial state is illegal, it is removed from the set of initial states and u_{12} is added. In this case P_1 and P_2 are called* incompatible *and $P_1 \parallel P_2$ is a universal MIA, otherwise they are called* compatible*. Finally, all unreachable states except for u_{12} and all their incoming and outgoing transitions are removed. Note that this contains all states in E. If $(p_1, p_2) \in P_1 \parallel P_2$, we write $p_1 \parallel p_2$ and call p_1 and p_2* compatible*.*

Observe that the parallel composition of MIAs results in a well-defined MIA. Firstly, this is true for the parallel product; in particular, u_{12} does not have any transitions. Secondly, pruning guarantees that all target sets of must-transitions are non-empty, and it preserves syntactic consistency and the sink condition. As an aside, even if we would not have required the sink condition in Def. 4.1, it would be enforced when applying parallel composition. Due to the universality of u, $P_1 \parallel P_2$ is universally refineable if P_1 and P_2 are incompatible. Observe also that, in $P \otimes Q$, a transition from a legal state to an illegal one must be an input transition, which is removed. Hence, for compatible P_1 and P_2, all illegal states are removed as well. All transitions of $P_1 \parallel P_2$ exist in $P_1 \otimes P_2$, except for transitions to u_{12}.

Before proving associativity and compositionality for $\parallel$, we explain how and why the pruning in Def. 4.9 differs from previous versions of MIA.

In [13] we used a simpler pruning, which only replaces i-transitions originating from a state $(p_1, p_2) \notin E$ with $(p_1, p_2) \overset{i}{\dashrightarrow} (p_1', p_2') \in E$. This *cutting* did not propagate backwards over τ-paths. Let us denote this old parallel composition as $\overline{\|}$. While $\overline{\|}$ is compositional for the refinement of [13], it is not so for our current refinement[4]:

Consider P and Q from Fig. 4.7 and observe that $P \sqsubseteq Q$. Also consider $R =_{df} (\{r_0, u_R\}, \{d\}, \emptyset, \emptyset, \emptyset, \{r_0\}, u_R)$, a MIA without any transitions and with d as input and only action. The old parallel compositions $Q \overline{\|} R$ and $P \overline{\|} R$ are shown, and it is easy to see that $P \overline{\|} R \not\sqsubseteq Q \overline{\|} R$ since $q_0 \overline{\|} r_0 \xrightarrow{i} \overline{q} \overline{\|} r_0$ cannot be matched as $p_0 \overline{\|} r_0$ does not have any i-must-transition. The new parallel composition (cf. Def. 4.9) cuts all i-transitions of $P \otimes Q$. The resulting $Q \parallel R$ is also depicted and in fact MIA-equivalent to $P \overline{\|} R = P \parallel R$.

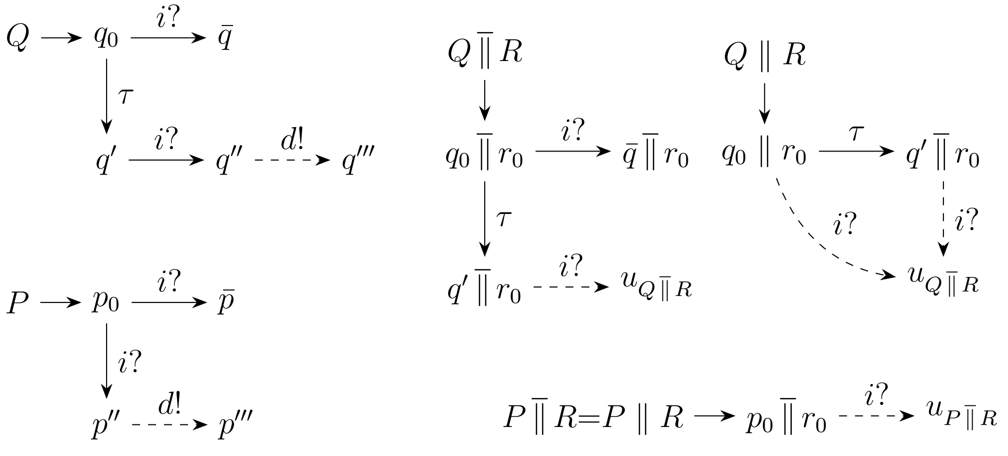

Figure 4.7: Illustration of pruning, where $A_P = A_Q = \{i\}/\{d\}$ and $A_R = \{d\}/\emptyset$.

This pruning is also intuitively sensible: consider a system S intending

[4] In [13, Rem. 9], we claimed that 'defining a general fix is much more involved since backward *and* forward propagation along τs is necessary'. This is actually not the case.

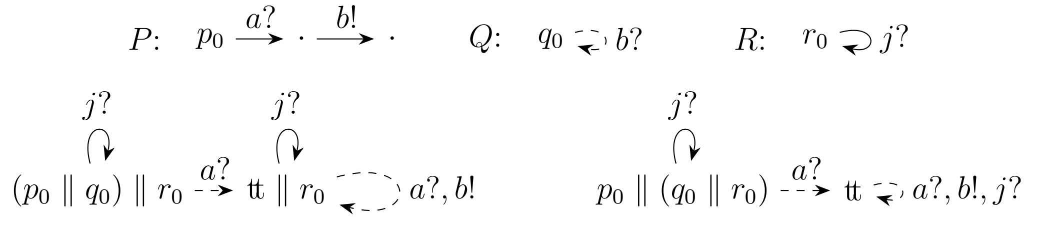

Figure 4.8: Differences between our state u and tt in [59], where $A_P = \{a\}/\{b\}$, $A_Q = \{b\}/\emptyset$ and $A_R = \{j\}/\emptyset$.

to send the signal i to $Q \mathrel{\overline{\|}} R$. It cannot know whether $Q \mathrel{\overline{\|}} R$ is still in state $q_0 \mathrel{\overline{\|}} r_0$, where it could accept the signal, or if it has already switched to state $q' \mathrel{\overline{\|}} r_0$ such that a mismatch occurs. So S is incompatible with $Q \mathrel{\overline{\|}} R$ as well as with $Q \parallel R$. For a system not intending to send i at this point, there is no difference either. Thus, a user cannot see a difference between $Q \mathrel{\overline{\|}} R$ and $Q \parallel R$.

In [59], Raclet et al. use a similar approach to pruning, but without an explicit universal state. Instead, when pruning illegal states, they introduce a state we denote as tt, which almost behaves like our universal state. By construction, this state has only input may-transitions as incoming transitions. Furthermore, it has a may-loop for every action of the parallel composition so that it can be refined by any state, much like our universal state (see Def. 4.4(i)). But tt behaves differently in a parallel composition. To see this, consider the MIAs P, Q, R in Fig. 4.8, where we construct $(P \parallel Q) \parallel R$ according to [59]. Since tt is an ordinary state, it is combined with r_0 inheriting the j-must-loop. In our approach, the combination with r_0 is an inherited error, and the target state just has a j-may-loop.

More importantly, there is the severe problem that parallel composition in [59] is not associative. Consider $P \parallel (Q \parallel R)$, also shown in Fig. 4.8, which is not equivalent according to $\sqsupseteq\sqsubseteq$ (and the equivalence in [59]) to $(P \parallel Q) \parallel R$, due to the j-must-loop at tt $\parallel r_0$. Note that our example does not rely on the multicast aspect of our parallel composition; it works just as well for the classic IA parallel composition.

We now prove that our parallel composition is indeed associative, starting with two lemmas.

Lemma 4.10. *If P, Q are composable MIAs, $p \parallel q \in P \parallel Q$, $o \in O_{P\|Q}$ and $i \in I_{P\|Q}$, then:*

1. $p \parallel q \overset{o}{\dashrightarrow}$ *iff* $p \overset{o}{\dashrightarrow}$ *and* $o \in O_P$, *or* $q \overset{o}{\dashrightarrow}$ *and* $o \in O_Q$.

2. *If $p \overset{i}{\not\longrightarrow}$ and $i \in I_P$ or if $q \overset{i}{\not\longrightarrow}$ and $i \in I_Q$, then $p \parallel q \overset{i}{\not\longrightarrow}$. The reverse implication does not hold in general.*

Proof. 1. Implication "⇒" is obvious. If implication "⇐" were false, then (p, q) would be a new error or $(p, q) \overset{o}{\dashrightarrow} (p', q')$ in $P \otimes Q$ with $p' \parallel q'$ undefined. Both would render (p, q) illegal and $p \parallel q$ undefined, leading to a contradiction.

2. This implication is also obvious, but the reverse implication does not hold since the must-transition of $p \parallel q$ might have been cut during pruning. □

Lemma 4.11. *Given three MIAs P_1, P_2 and P_3, we have:*

1. *$(P_1 \parallel P_2) \parallel P_3$ is defined iff P_1, P_2 and P_3 are pairwise composable iff $P_1 \parallel (P_2 \parallel P_3)$ is defined as well.*

2. *$(P_1 \parallel P_2) \parallel P_3$ is equal to S obtained from applying pruning in one step to $(P_1 \otimes P_2) \otimes P_3$ (up to the name of the respective universal state). For this purpose, a state $((p_1, p_2), p_3)$ is a new error if, for some $i \neq j$ with $i, j \in \{1, 2, 3\}$, there is some $a \in A_i \cap A_j$ such that $a \in O_i$, $p_i \overset{a}{\dashrightarrow}$ and $p_j \overset{a}{\not\longrightarrow}$; it is an inherited one, if $p_i = u_i$ for some $i \in \{1, 2, 3\}$.*

Proof. 1. is easy. 2. For reasons of readability we use P, Q, R instead of P_1, P_2, P_3 and write (p, q, r) for $((p, q), r)$. Let E_{PQR} denote the illegal states of $(P \otimes Q) \otimes R$ as defined above when constructing S. We denote the illegal states of $P \otimes Q$ and $(P \parallel Q) \otimes R$ by E_{PQ} and $E_{(P\parallel Q)\otimes R}$ respectively. Furthermore, let Err_{PQR}, Err_{PQ} and $Err_{(P\parallel Q)\otimes R}$ be the errors of the respective systems. We also say that two states p and q *produce an error*, if (p, q) is an error due to $p \overset{a}{\dashrightarrow}$ and $q \overset{a}{\not\longrightarrow}$ while $a \in O_P \cap I_Q$ or vice versa.

Our first aim is to show that $E_{PQR} = (E_{PQ} \times R) \cup (E_{(P\parallel Q)\otimes R} \setminus (\{u_{P\parallel Q}\} \times R))$.

Part "⊆" We prove that $(p, q, r) \in E_{PQR}$ is contained in the r.h.s. by induction on the length of a local transition sequence from (p, q, r) to an error in Err_{PQR}. For the base case, we show $Err_{PQR} \subseteq (E_{PQ} \times R) \cup (E_{(P\parallel Q)\otimes R} \setminus (\{u_{P\parallel Q}\} \times R))$.

Consider $(p, q, r) \in Err_{PQR}$. If (p, q) is illegal in $P \otimes Q$ (this covers the cases that p or q is universal or that p and q produce an error), then $(p, q, r) \in E_{PQ} \times R$. Otherwise, $r = u_R$ and $(p, q, r) \in Err_{(P\parallel Q)\otimes R} \setminus (\{u_{P\parallel Q}\} \times R) \subseteq E_{(P\parallel Q)\otimes R} \setminus (\{u_{P\parallel Q}\} \times R)$, or r produces the error with p or q (or possibly both). W.l.o.g. let p and r produce the error because $p \overset{a}{\dashrightarrow}$ and $r \overset{a}{\not\longrightarrow}$ for some $a \in O_P \cap I_R$ or because $p \overset{a}{\not\longrightarrow}$ and $r \overset{a}{\dashrightarrow}$ for some $a \in I_P \cap O_R$. By

Lem. 4.10.1, this leads to $p \parallel q \overset{a}{\dashrightarrow}$ and $r \overset{a}{\not\longrightarrow}$ or, by Lem. 4.10.2, to $p \parallel q \overset{a}{\not\longrightarrow}$ and $r \overset{a}{\dashrightarrow}$. Again, $(p, q, r) \in \mathit{Err}_{(P\parallel Q)\otimes R} \setminus (\{u_{P\parallel Q}\} \times R)$.

For the induction step, consider $(p, q, r) \in E_{PQR}$ such that $(p, q, r) \overset{\omega}{\dashrightarrow} (p', q', r') \in E_{PQR}$ and $(p', q', r') \in (E_{PQ} \times R) \cup (E_{(P\parallel Q)\otimes R} \setminus (\{u_{P\parallel Q}\} \times R))$ by induction hypothesis. By the argument at the beginning of the base case, we can assume that $p \parallel q$ is defined and, thus, $(p \parallel q, r)$ exists in $(P \parallel Q) \otimes R$. Thus, if $(p', q', r') \in E_{(P\parallel Q)\otimes R} \setminus (\{u_{P\parallel Q}\} \times R))$, then $(p, q, r) \in E_{(P\parallel Q)\otimes R} \setminus (\{u_{P\parallel Q}\} \times R))$ by the definition of E.

Finally, consider $(p', q', r') \in E_{PQ} \times R$. If the ω-transition is only performed by r, then $(p', q', r') = (p, q, r')$ and, thus, $(p, q) \in E_{PQ}$, contradicting that (p, q) is not illegal. Otherwise, if $\omega \in O_{P\otimes Q} \cup \{\tau\}$, then $(p, q) \overset{\omega}{\dashrightarrow} (p', q') \in E_{PQ}$ and $(p, q) \in E_{PQ}$, a contradiction. Thus, $\omega \in I_{P\otimes Q}$ and r performs ω as an output since, overall, it is an output. As $(p, q) \overset{\omega}{\dashrightarrow} (p', q') \in E_{PQ}$, this input transition is cut when pruning $P \otimes Q$, implying $p \parallel q \overset{\omega}{\not\longrightarrow}$. This shows again that $(p, q, r) \in \mathit{Err}_{(P\parallel Q)\otimes R} \setminus (\{u_{P\parallel Q}\} \times R)$.

Part "$\supseteq$" We show that $(E_{PQ} \times R) \cup (E_{(P\parallel Q)\otimes R} \setminus (\{u_{P\parallel Q}\} \times R)) \subseteq E_{PQR}$.

First, we establish $E_{PQ} \times R \subseteq E_{PQR}$: We prove that $(p, q, r) \in E_{PQ} \times R$ is contained in E_{PQR} by induction on the length of a local transition sequence from (p, q) to an error in Err_{PQ}. In the base case $(p, q) \in \mathit{Err}_{PQ}$, we have that p and q produce an error or one of them is an error state. In either case $(p, q, r) \in \mathit{Err}_{PQR} \subseteq E_{PQR}$. For the induction step, consider some $(p, q) \overset{\omega}{\dashrightarrow} (p', q') \in E_{PQ}$ where, by induction hypothesis, $\{(p', q')\} \times R \subseteq E_{PQR}$. If $\omega \notin A_R$, then $(p, q, r) \overset{\omega}{\dashrightarrow} (p', q', r) \in E_{PQR}$, and we are done. If $\omega \in A_R$, then we must have $\omega \in I_R$. Now, either $(p, q, r) \in \mathit{Err}_{PQR}$ or $(p, q, r) \overset{\omega}{\dashrightarrow} (p', q', r') \in E_{PQR}$ for some r', and in either case we are done.

Second, we establish $E_{(P\parallel Q)\otimes R} \setminus (\{u_{P\parallel Q}\} \times R) \subseteq E_{PQR}$. We prove that $(p, q, r) \in E_{(P\parallel Q)\otimes R} \setminus (\{u_{P\parallel Q}\} \times R)$ is contained in E_{PQR} by induction on the length of a local transition sequence from $(p \parallel q, r)$ to an error in $\mathit{Err}_{(P\parallel Q)\otimes R}$. In the base case $(p \parallel q, r) \in \mathit{Err}_{(P\parallel Q)\otimes R} \setminus (\{u_{P\parallel Q}\} \times R)$, we have that $r = u_R$ and, thus, $(p, q, r) \in \mathit{Err}_{PQR} \subseteq E_{PQR}$, or that $p \parallel q$ and r produce an error. The latter means either $p \parallel q \overset{a}{\dashrightarrow}$ and $r \overset{a}{\not\longrightarrow}$ for some $a \in (O_P \cup O_Q) \cap I_R$, implying $p \overset{a}{\dashrightarrow}$ and $a \in O_P$ or $q \overset{a}{\dashrightarrow}$ and $a \in O_Q$ by Lem. 4.10.1, and hence $(p, q, r) \in \mathit{Err}_{PQR} \subseteq E_{PQR}$; or $p \parallel q \overset{a}{\not\longrightarrow}$ and $r \overset{a}{\dashrightarrow}$ for some $a \in (I_P \cup I_Q) \cap O_R$. Here, $p \parallel q \overset{a}{\not\longrightarrow}$ can have several reasons. We might have $p \overset{a}{\not\longrightarrow}$ and $a \in I_P$, or $q \overset{a}{\not\longrightarrow}$ and $a \in I_Q$, and in both cases $(p, q, r) \in \mathit{Err}_{PQR}$ due to $r \overset{a}{\dashrightarrow}$. Otherwise, $(p, q) \overset{a}{\longrightarrow}$ and $(p, q) \overset{\varepsilon}{\Longrightarrow}\overset{a}{\dashrightarrow} (p', q') \in E_{PQ}$; in this case, $(p, q, r) \overset{\varepsilon}{\Longrightarrow}\overset{a}{\dashrightarrow} (p', q', r') \in E_{PQ} \times R \subseteq E_{PQR}$ by the above, implying

$(p, q, r) \in E_{PQR}$ since $a \in O_{(P \otimes Q) \otimes R}$. For the induction step, consider some $(p \parallel q, r) \overset{\omega}{\dashrightarrow} (p' \parallel q', r') \in E_{(P \parallel Q) \otimes R}$; since $(p', q', r') \in E_{PQR}$ by induction hypothesis, we are done with the '$\supseteq$'-case and, thus, with establishing the desired equality.

Denoting the universal state of $(P \parallel Q) \parallel R$ and S by u, we now show that the state space $(P \times Q \times R) \setminus E_{PQR} \cup \{u\}$ of S coincides with the one of $(P \parallel Q) \parallel R$ (up to the name of the universal state). The states of $(P \parallel Q) \parallel R$ are:

$$\begin{aligned} &(((P \times Q) \setminus E_{PQ} \cup \{u_{P \parallel Q}\}) \times R) \setminus E_{(P \parallel Q) \otimes R} \cup \{u\} \\ &= ((P \times Q \times R) \setminus (E_{PQ} \times R) \cup (\{u_{P \parallel Q}\} \times R)) \setminus E_{(P \parallel Q) \otimes R} \cup \{u\} \\ &= \underbrace{(P \times Q \times R) \setminus ((E_{PQ} \times R) \cup E_{(P \parallel Q) \otimes R})}_{=(P \times Q \times R) \setminus E_{PQR}} \cup \underbrace{(\{(u_{P \parallel Q}\} \times R) \setminus E_{(P \parallel Q) \otimes R}}_{=\emptyset} \cup \{u\}) \end{aligned}$$

For the last step, note that $(P \times Q \times R) \cap (\{u_{P \parallel Q}\} \times R) = \emptyset$.

Finally, we prove that the transitions of $(P \parallel Q) \parallel R$ and S are the same. For transitions to u, consider $(p \parallel q) \parallel r \overset{i}{\dashrightarrow} u$ for some $i \in I_{(P \parallel Q) \parallel R}$. This transition exists iff $(p \parallel q, r) \overset{i}{\dashrightarrow}$ and $(p \parallel q, r) \overset{\varepsilon}{\Longrightarrow} \overset{i}{\dashrightarrow} (t, r') \in E_{(P \parallel Q) \otimes R}$ for some t and r'. Now, either $t = p' \parallel q'$ for some p' and q', and we have $(t, r') \in E_{(P \parallel Q) \otimes R} \setminus (\{u_{P \parallel Q}\} \times R)$; or $(p \parallel q, r) \overset{i}{\dashrightarrow} (u_{P \parallel Q}, r')$, which holds iff $(p, q) \overset{i}{\dashrightarrow}$ and $(p, q) \overset{\varepsilon}{\Longrightarrow} \overset{i}{\dashrightarrow} (p', q') \in E_{PQ}$ and either $r \overset{i}{\dashrightarrow} r'$ or $i \notin A_R$ and $r = r'$. This is equivalent to $(p, q, r) \overset{i}{\dashrightarrow}$ and $(p, q, r) \overset{\varepsilon}{\Longrightarrow} \overset{i}{\dashrightarrow} (p', q', r') \in E_{PQ} \times R$. Both cases together show: $(p \parallel q) \parallel r \overset{i}{\dashrightarrow} u$ iff $(p, q, r) \overset{i}{\dashrightarrow}_{P \otimes Q \otimes R}$ and $(p, q, r) \overset{\varepsilon}{\Longrightarrow} \overset{i}{\dashrightarrow}_{P \otimes Q \otimes R} (p', q', r') \in E_{PQR}$ iff $(p, q, r) \overset{i}{\dashrightarrow}_S u$ in S.

For transitions between the states of S, which are also the states of $(P \parallel Q) \parallel R$, observe that these are exactly the transitions inherited from $(P \otimes Q) \otimes R$ minus all i-transitions from any s with $s \overset{i}{\dashrightarrow} u$. In $(P \parallel Q) \parallel R$, all transitions are inherited indirectly from $(P \otimes Q) \otimes R$; if $s \overset{i}{\dashrightarrow} u$, s clearly has no other i-transitions.

It remains for us to show that no a-transition from some state $s \in S$ is missing, if $s \overset{a}{\not\dashrightarrow} u$. Assume the contrary, namely that a transition $s = (p, q, r) \overset{a}{\dashrightarrow}_{P \otimes Q \otimes R} (p', q', r')$ of S is missing in $(P \parallel Q) \parallel R$ although $s \overset{a}{\not\dashrightarrow} u$. This can only be due to pruning; recall that $(p \parallel q) \parallel r$ and $(p' \parallel q') \parallel r'$ are states of $(P \parallel Q) \parallel R$.

If $(p, q) \overset{a}{\not\dashrightarrow}_{P \otimes Q}$, then $a \notin A_P \cup A_Q$, and the missing transition was lost when pruning $(P \parallel Q) \otimes R$, contradicting $s \overset{a}{\not\dashrightarrow} u$. Thus, $(p, q) \overset{a}{\dashrightarrow}_{P \otimes Q} (p', q')$.

If $p \parallel q \overset{a}{\not\dashrightarrow} p' \parallel q'$, then we have $p \parallel q \overset{a}{\dashrightarrow} u_{P\parallel Q}$ and $(p \parallel q, r)$ is illegal if $a \in O_R$ or $(p \parallel q) \parallel r \overset{a}{\dashrightarrow} u$, a contradiction in both cases. Thus, $(p \parallel q, r) \overset{a}{\dashrightarrow} (p' \parallel q', r')$ in $(P \parallel Q) \otimes R$. Again in this case, the transition was lost when pruning $(P \parallel Q) \otimes R$, a contradiction. □

This lemma immediately implies the desired associativity:

Theorem 4.12. *Parallel composition is associative in the sense that, for MIAs P, Q and R, if $(P \parallel Q) \parallel R$ is defined, then $P \parallel (Q \parallel R)$ is defined and both are isomorphic, and vice versa.*

Now we proceed to show that MIA refinement is compositional w.r.t. parallel composition, which essentially means that $P_1 \sqsubseteq Q_1$ implies $P_1 \parallel P_2 \sqsubseteq Q_1 \parallel P_2$ for all MIAs P_1, Q_1 and P_2. The proof requires the following two lemmas:

Lemma 4.13 (Illegal states and refinement)**.** *For MIAs P_1, P_2 and Q_1, let E_P be the E-set of $P_1 \otimes P_2$ and E_Q be the one of $Q_1 \otimes P_2$. Further, let $p_1 \in P_1$, $p_2 \in P_2$ and $q_1 \in Q_1$ such that $p_1 \sqsubseteq q_1$. Then, $(p_1, p_2) \in E_P$ implies $(q_1, p_2) \in E_Q$.*

Proof. Let I_1/O_1 be the alphabets of P_1 and Q_1, let I_2/O_2 be the alphabets of P_2, and let I/O be the alphabets of the products. The proof is by induction on the length of a path from (p_1, p_2) to an error of $P_1 \otimes P_2$:

(Base) Let (p_1, p_2) be an error.

- Let $p_1 \overset{a}{\dashrightarrow}$ with $a \in O_1 \cap I_2$ and $p_2 \overset{a}{\not\longrightarrow}$. Then, for some q_1', we have $q_1 \overset{\varepsilon}{\Longrightarrow} q_1' \overset{a}{\dashrightarrow}$ by $p_1 \sqsubseteq q_1$; hence, $(q_1, p_2) \overset{\varepsilon}{\Longrightarrow} (q_1', p_2) \in E_Q$ and $(q_1, p_2) \in E_Q$ as well.
- Let $p_2 \overset{a}{\dashrightarrow}$ with $a \in O_2 \cap I_1$ and $p_1 \overset{a}{\not\longrightarrow}$. If $q_1 \xrightarrow{a}$, we have a contradiction to $p_1 \sqsubseteq q_1$; otherwise, (q_1, p_2) is an error since $a \in I_1 \cap O_2$.
- If $p_1 = u_{P_1}$, then $q_1 = u_{Q_1}$ because of $p_1 \sqsubseteq q_1$, and thus $(q_1, p_2) \in E_Q$.
- Case $p_2 = u_{P_2}$ is obvious.

(Step) For a shortest path from state (p_1, p_2) to an error, consider the first transition $(p_1, p_2) \overset{\omega}{\dashrightarrow} (p_1', p_2') \in E_P$, where $\omega \in O \cup \{\tau\}$. The transition is due to either Rule (PMay1), (PMay2) or (PMay3). The treatment of (PMay3) is easier than in [13] since, there, ω can be in O_1 or in $I_1 \cap O_2$. In all cases we find some $q_1' \in Q_1$ such that (q_1', p_2') is locally reachable from (q_1, p_2) and $p_1' \sqsubseteq q_1'$. The latter implies $(q_1', p_2') \in E_Q$ by induction hypothesis.

(PMay1) $p_1 \overset{\omega}{\dashrightarrow} p_1'$, $p_2 = p_2'$, $\omega \notin A_2$. Due to $p_1 \sqsubseteq q_1$, there is a q_1' such that $q_1 \overset{\hat{\omega}}{\Longrightarrow} q_1'$ and $p_1' \sqsubseteq q_1'$, and $(q_1, p_2) \overset{\hat{\omega}}{\Longrightarrow} (q_1', p_2)$ by applications of (PMay1). By induction hypothesis, $(q_1', p_2) \in E_Q$ and, therefore, $(q_1, p_2) \in E_Q$.

(PMay2) $p_1 = p_1'$, $p_2 \overset{\omega}{\dashrightarrow} p_2'$ and $\omega \notin A_1$. Using (PMay2) we obtain $(q_1, p_2) \overset{\omega}{\dashrightarrow} (q_1, p_2')$, so that $(q_1, p_2') \in E_Q$ by induction hypothesis. Hence, $(q_1, p_2) \in E_Q$, too.

(PMay3) $\omega = o$, $p_1 \overset{o}{\dashrightarrow} p_1'$ and $p_2 \overset{o}{\dashrightarrow} p_2'$ with $o \in A_1 \cap A_2$. Note that o is an output for the product and one of its components, but an input for the other. By $p_1 \sqsubseteq q_1$ we have $q_1 \overset{\varepsilon}{\Longrightarrow} q_1'' \overset{o}{\dashrightarrow} q_1''' \overset{\varepsilon}{\Longrightarrow} q_1'$ for some q_1', q_1'', q_1''' with $p_1' \sqsubseteq q_1'$. Therefore, we get $(q_1, p_2) \overset{\varepsilon}{\Longrightarrow} (q_1'', p_2) \overset{o}{\dashrightarrow} (q_1''', p_2') \overset{\varepsilon}{\Longrightarrow} (q_1', p_2')$ via (PMay1) and (PMay3). By induction hypothesis, $(q_1', p_2') \in E_Q$ and, hence, $(q_1, p_2) \in E_Q$, too. □

The next lemma generalises the synchronization according to Rule (PMust3) to weak transitions:

Lemma 4.14 (Weak Must-Transitions)**.** *Let P, Q be composable MIAs.*

1. *For $\alpha \notin A_Q$, $p \overset{\hat{\alpha}}{\Longrightarrow} P'$ and $q \in Q$ implies $(p, q) \overset{\hat{\alpha}}{\Longrightarrow} P' \times \{q\}$ in $P \otimes Q$.*
2. *If $p \overset{a}{\Longrightarrow} P'$ (or $p \xrightarrow{a}\overset{\varepsilon}{\Longrightarrow} P'$) and $q \xrightarrow{a} Q'$ for some $a \in A_P \cap A_Q$, then $(p, q) \overset{a}{\Longrightarrow} P' \times Q'$ (or $(p, q) \xrightarrow{a}\overset{\varepsilon}{\Longrightarrow} P' \times Q'$) in $P \otimes Q$.*

Proof. Claim 1: Clearly, the mapping $P \to P \times \{q\} : p \mapsto (p, q)$ is an isomorphism if we only consider must-transitions labelled with the given α or τ and states in $P \times \{q\}$ in $P \otimes Q$.

Claim 2: By induction on the definition of $p \overset{a}{\Longrightarrow} P'$. In Case 2 of Def. 4.2, we have $p \xrightarrow{\tau} \bar{P}$ and a suitable $\bar{p} \overset{a}{\Longrightarrow} P_{\bar{p}}$ for each $\bar{p} \in \bar{P}$, such that $P' = \bigcup_{\bar{p} \in \bar{P}} P_{\bar{p}}$. Then, $(p, q) \xrightarrow{\tau} \bar{P} \times \{q\}$ due to (PMust1), and $(\bar{p}, q) \overset{a}{\Longrightarrow} P_{\bar{p}} \times Q'$ by induction hypothesis; this yields $(p, q) \overset{a}{\Longrightarrow} P' \times Q'$ due to Def. 4.2.2. In Case 3 (the only one for the variant concerning $\xrightarrow{a}\overset{\varepsilon}{\Longrightarrow}$), we have $p \xrightarrow{a} \bar{P}$ and a suitable $\bar{p} \overset{\varepsilon}{\Longrightarrow} P_{\bar{p}}$ for each $\bar{p} \in \bar{P}$ such that $P' = \bigcup_{\bar{p} \in \bar{P}} P_{\bar{p}}$. Then, $(p, q) \xrightarrow{a} \bar{P} \times Q'$ by (PMust3) and, for each $(\bar{p}, q') \in \bar{P} \times Q'$, we get $(\bar{p}, q') \overset{\varepsilon}{\Longrightarrow} P_{\bar{p}} \times \{q'\}$ by Claim 1, hence $\bar{P} \times Q' \overset{\varepsilon}{\Longrightarrow} P' \times Q'$. By Def. 4.2.3 we obtain $(p, q) \overset{a}{\Longrightarrow} P' \times Q'$. □

The proof for the following compositionality theorem has many similarities to the corresponding proof in [5] (where leading τs are not allowed when matching input-may-transitions), but the details are more involved due to the more intricate definition of pruning. It is remarkable that our new pruning is needed to deal with must-transitions (Subcases (PMust1) and (PMust3) in Case (ii)). This might be surprising since cutting propagates backwards over τ-may-transitions only.

Theorem 4.15 (Compositionality of Parallel Composition)**.** *Let P_1, P_2 and Q_1 be MIAs and $P_1 \sqsubseteq Q_1$. Assume that Q_1 and P_2 are composable, then:*

1. *P_1 and P_2 are composable; they are compatible if Q and P_2 are.*
2. *$P_1 \parallel P_2 \sqsubseteq Q_1 \parallel P_2$.*

Proof. Part 1 is trivial and follows from Lem. 4.13 respectively. We denote the universal state of $P_1 \parallel P_2$ and $Q_1 \parallel P_2$ by u_P and u_Q, respectively. E_P stands for the E-set of $P_1 \otimes P_2$ and E_Q for the one of $Q_1 \otimes P_2$, as in Lem. 4.13. To establish Part 2, we prove that

$$\mathcal{R} =_{df} \{(p_1 \parallel p_2,\ q_1 \parallel p_2) \mid p_1 \sqsubseteq q_1\} \cup ((P_1 \parallel P_2) \times \{u_Q\})$$

is a MIA-refinement relation by checking the conditions of Def. 4.4. Then, we are done: If Q_1 and P_2 are incompatible, the initial states of $P_1 \parallel P_2$ are matched by u_Q. Otherwise, P_1 and P_2 are also compatible by Part 1. Each initial state of $P_1 \parallel P_2$ has the form $p_{01} \parallel p_{02}$ with $p_{01} \in P_{01}$ and $p_{02} \in P_{02}$. By $P_1 \sqsubseteq Q_1$ there is a matching $q_{01} \in Q_{01}$ for p_{01} and thus $(p_{01} \parallel p_{02}, q_{01} \parallel p_{02}) \in \mathcal{R}$.

For the second subset, the check is trivial; so consider some $(p_1 \parallel p_2, q_1 \parallel p_2) \in \mathcal{R}$:

(i) Obvious, since $p_1 \parallel p_2 \neq u_P$.

(ii) Let $q_1 \parallel p_2 \xrightarrow{i} \bar{Q}$ due to either Rule (PMust1), (PMust2) or (PMust3). Recall that $(q_1, p_2) \xrightarrow{i} \bar{Q}$ in $Q_1 \otimes P_2$ as well and that $\bar{Q}$ only consists of legal states.

(PMust1) $q_1 \xrightarrow{i} Q'_1$ and $\bar{Q} = Q'_1 \times \{p_2\}$. Then, by $p_1 \sqsubseteq q_1$, there is a $P'_1 \subseteq P_1$ such that $p_1 \xrightarrow{i} \overset{\varepsilon}{\Longrightarrow} P'_1$ and $\forall p'_1 {\in} P'_1\, \exists q'_1 {\in} Q'_1.\, p'_1 \sqsubseteq q'_1$. Now, $(p_1, p_2) \xrightarrow{i} \overset{\varepsilon}{\Longrightarrow} P'_1 \times \{p_2\}$ in $P_1 \otimes P_2$ by repeated application of Rule (PMust1) and since $i \notin A_2$. For every $(p'_1, p_2) \in P'_1 \times \{p_2\}$, we have a suitable $(q'_1, p_2) \in Q'_1 \times \{p_2\}$; moreover, $(p'_1, p_2) \notin E_P$ by Lem. 4.13 since $(q'_1, p_2) \notin E_Q$. Thus, we have $(p'_1 \parallel p_2, q'_1 \parallel p_2) \in \mathcal{R}$.

It remains for us to show that $(p_1, p_2) \xrightarrow{i}\overset{\varepsilon}{\Longrightarrow} P'_1 \times \{p_2\}$ also exists in $P_1 \parallel P_2$, i.e., that no state (p''_1, p_2) along this weak transition is pruned. More generally, let us consider any $\bar{p}_1$ and p''_1 with $p_1 \overset{i}{\dashrightarrow} \bar{p}_1 \overset{\varepsilon}{\Longrightarrow} p''_1$, implying $(p_1, p_2) \overset{i}{\dashrightarrow} (\bar{p}_1, p_2) \overset{\varepsilon}{\Longrightarrow} (p''_1, p_2)$. Because of $p_1 \overset{i}{\dashrightarrow} \bar{p}_1$ and $p_1 \sqsubseteq q_1$, there must be some $\bar{q}_1$ with $q_1 \overset{i}{\Longrightarrow} \bar{q}_1$ which implies $(q_1, p_2) \overset{i}{\Longrightarrow} (\bar{q}_1, p_2)$, and $\bar{p}_1 \sqsubseteq \bar{q}_1$. If $(\bar{q}_1, p_2) \in E_Q$, then the i-transition on $(q_1, p_2) \overset{i}{\Longrightarrow} (\bar{q}_1, p_2)$ would lead to an illegal state and all outgoing i-transitions from $q_1 \parallel p_2$ would have been pruned, contradicting our assumptions. This argument depends on our new pruning; without the backwards propagation of cutting (as e.g. in [13]) it would not work. Thus, $(\bar{p}_1, p_2) \notin E_P$ by Lem. 4.13, which implies by definition of E_P that $(p''_1, p_2) \notin E_P$, too.

(PMust2) $p_2 \xrightarrow{i} P'_2$ and $\bar{Q} = \{q_1\} \times P'_2$. Then, $(p_1, p_2) \xrightarrow{i} \bar{P} = \{p_1\} \times P'_2$ according to (PMust2) and since $i \notin A_1$. For $(p_1, p'_2) \in \bar{P}$, we get $(p_1, p'_2) \notin E_P$ due to Lem. 4.13, because $(q_1, p'_2) \notin E_Q$. Thus, $p_1 \parallel p_2 \xrightarrow{i} \bar{P}$ and, for every $p_1 \parallel p'_2 \in \bar{P}$, we have $q_1 \parallel p'_2 \in \bar{Q}$ with $(p_1 \parallel p'_2, q_1 \parallel p'_2) \in \mathcal{R}$.

(PMust3) $q_1 \xrightarrow{i} Q'_1$, $p_2 \xrightarrow{i} P'_2$ and $\bar{Q} = Q'_1 \times P'_2$. (Note that $i \in I_1 \cap I_2$.) Then, by $p_1 \sqsubseteq q_1$, there is a set $P'_1 \subseteq P_1$ such that $p_1 \xrightarrow{i}\overset{\varepsilon}{\Longrightarrow} P'_1$ and $\forall p'_1{\in}P'_1\, \exists q'_1{\in}Q'_1.\, p'_1 \sqsubseteq q'_1$. By Lem. 4.14 we get $(p_1, p_2) \xrightarrow{i}\overset{\varepsilon}{\Longrightarrow} P'_1 \times P'_2$. As in Case (PMust1) above, the states in $P'_1 \times P'_2$ are matched by $\bar{Q}$ according to $\mathcal{R}$.

Again, similarly to Case (PMust1), we have to show that $(p_1, p_2) \xrightarrow{i}\overset{\varepsilon}{\Longrightarrow} P'_1 \times P'_2$ also exists in $P_1 \parallel P_2$, i.e., no state (p''_1, p'_2) along this weak transition is pruned. More generally, let us consider any $\bar{p}_1$ and p''_1 with $p_1 \overset{i}{\dashrightarrow} \bar{p}_1 \overset{\varepsilon}{\Longrightarrow} p''_1$ and some p'_2 with $p_2 \overset{i}{\dashrightarrow} p'_2$, implying $(p_1, p_2) \overset{i}{\dashrightarrow} (\bar{p}_1, p'_2) \overset{\varepsilon}{\Longrightarrow} (p''_1, p'_2)$. Because of $p_1 \overset{i}{\dashrightarrow} \bar{p}_1$ and $p_1 \sqsubseteq q_1$, there must be some $\bar{q}_1$ with $q_1 \overset{i}{\Longrightarrow} \bar{q}_1$, which implies $(q_1, p_2) \overset{i}{\Longrightarrow} (\bar{q}_1, p'_2)$, and $\bar{p}_1 \sqsubseteq \bar{q}_1$. If $(\bar{q}_1, p'_2) \in E_Q$, then all outgoing i-transitions from $q_1 \parallel p_2$ would have been pruned, contradicting our assumptions as above; thus $(p''_1, p'_2) \notin E_P$ follows.

(iii) Let $q_1 \parallel p_2 \xrightarrow{\omega} \bar{Q}$ due to either (PMust1), (PMust2) or (PMust3). Again the transition and the states exist in $Q_1 \otimes P_2$.

(PMust1) $q_1 \xrightarrow{\omega} Q'_1$, $\omega \notin A_2$ and $\bar{Q} = Q'_1 \times \{p_2\}$. Then, by $p_1 \sqsubseteq q_1$, there exists $P'_1 \subseteq P_1$ such that $p_1 \overset{\hat{\omega}}{\Longrightarrow} P'_1$ and $\forall p'_1{\in}P'_1\, \exists q'_1{\in}Q'_1.\, p'_1 \sqsubseteq q'_1$. Now, $(p_1, p_2) \overset{\hat{\omega}}{\Longrightarrow} P'_1 \times \{p_2\}$ according to (PMust1) and since $\omega \notin A_2$. Because $p_1 \parallel p_2$ is defined, this also holds for all pairs along this weak transition by the definition of E_P. For $p'_1 \in P'_1$ we have a suitable $q'_1 \in Q'_1$ such that, for the arbitrary $p'_1 \parallel p_2$, we may also infer $(p'_1 \parallel p_2, q'_1 \parallel q_2) \in \mathcal{R}$.

(PMust2) $p_2 \xrightarrow{\omega}_{P_2} P'_2$, $\omega \notin A_1$ and $\bar{Q} = \{q_1\} \times P'_2$. In this case we obtain that $(p_1, p_2) \xrightarrow{\omega} \bar{P} = \{p_1\} \times P'_2$ by (PMust2) and $\omega \notin A_1$. For $(p_1, p'_2) \in \bar{P}$ we get $(p_1, p'_2) \notin E_P$ due to Lem. 4.13 since $(q_1, p'_2) \notin E_Q$. Thus, $p_1 \parallel p_2 \xrightarrow{\omega} \bar{P}$ and therefore also $p_1 \parallel p_2 \overset{\hat{\omega}}{\Longrightarrow} \bar{P}$. For $(p_1, p'_2) \in \bar{P}$, we also have $(p_1 \parallel p'_2, q_1 \parallel p'_2) \in \mathcal{R}$.

(PMust3) $\omega = o$, $q_1 \xrightarrow{o} Q'_1$, $p_2 \xrightarrow{o} P'_2$ for some action $o \in (O_1 \cap I_2) \cup (I_1 \cap O_2)$, and $\bar{Q} = Q'_1 \times P'_2$. By $p_1 \sqsubseteq q_1$, there exists some $P'_1 \subseteq P_1$ with $p_1 \overset{o}{\Longrightarrow} P'_1$ (possibly $p_1 \xrightarrow{o}\overset{\varepsilon}{\Longrightarrow} P'_1$, if $o \in I_1$) such that $\forall p'_1{\in}P'_1$ $\exists q'_1{\in}Q'_1.\, p'_1 \sqsubseteq q'_1$. Now, $(p_1, p_2) \overset{o}{\Longrightarrow} P'_1 \times P'_2$ by Lem. 4.14 and, as in Case (PMust1) above, all pairs along this weak transition exist in $P_1 \times P_2$. Hence, $p_1 \parallel p_2 \overset{o}{\Longrightarrow} P'_1 \times P'_2$ and, for all $p'_1 \parallel p'_2 \in P'_1 \times P'_2$, we have some $q' \in Q'$ such that $(p'_1 \parallel p'_2, q'_1 \parallel p'_2) \in \mathcal{R}$.

(iv) First, consider $p_1 \parallel p_2 \overset{i}{\dashrightarrow} u_P$ due to pruning, i.e. $(p_1, p_2) \overset{\varepsilon}{=\!\!\Rightarrow} (p''_1, p''_2) \overset{i}{\dashrightarrow} (p'_1, p'_2) \in E_P$ and $(p_1, p_2) \overset{i}{\dashrightarrow}$ in $P_1 \otimes P_2$.

(PMay1) $i \notin A_2$, i.e. $p_1 \overset{\varepsilon}{=\!\!\Rightarrow} p''_1 \overset{i}{\dashrightarrow} p'_1$ and $p'_2 = p''_2 = p_2$. By $p_1 \sqsubseteq q_1$ and Prop. 4.5, we have some q'_1 with $q_1 \overset{i}{=\!\!\Rightarrow} q'_1$ and $p'_1 \sqsubseteq q'_1$. With (PMay1) we get $(q_1, p_2) \overset{i}{=\!\!\Rightarrow} (q'_1, p_2)$, since we have $(q'_1, p_2) \in E_Q$ by Lem. 4.13. Therefore, $q_1 \parallel p_2 \overset{i}{=\!\!\Rightarrow} u_Q$ by pruning.

(PMay2) $i \notin A_1$, i.e. $p_2 \overset{\varepsilon}{=\!\!\Rightarrow}_{P_2} p''_2 \overset{i}{\dashrightarrow} p'_2$ and $p'_1 = p''_1 = p_1$. Then, $(q_1, p_2) \overset{i}{=\!\!\Rightarrow} (q_1, p'_2)$ by (PMay2). Since $(p_1, p'_2) = (p'_1, p'_2) \in E_P$, we get $(q_1, p'_2) \in E_Q$ by Lem. 4.13. Hence, $q_1 \parallel p_2 \overset{i}{=\!\!\Rightarrow} u_Q$ by pruning.

(PMay3) $p_1 \overset{i}{\dashrightarrow} p'_1$ and $p_2 \overset{i}{\dashrightarrow} p'_2$ for some action $i \in I_1 \cap I_2$. Due to $p_1 \sqsubseteq q_1$, we get $q_1 \overset{i}{=\!\!\Rightarrow} q'_1$ for some q'_1 such that $p'_1 \sqsubseteq q'_1$. Hence, $(q_1, p_2) \overset{i}{=\!\!\Rightarrow} (q'_1, p'_2)$ by Rules (PMay1) and (PMay3). Lem. 4.13 yields $(q'_1, p'_2) \in E_Q$. Therefore, $q_1 \parallel p_2 \overset{i}{=\!\!\Rightarrow} u_Q$ by pruning.

Second, we consider $p_1 \parallel p_2 \overset{\alpha}{\dashrightarrow} p'_1 \parallel p'_2$, due to one of the Rules (PMay1) through (PMay3).

(PMay1) $p_1 \overset{\alpha}{\dashrightarrow} p'_1$, $\alpha \notin A_2$ and $p'_2 = p_2$. By $p_1 \sqsubseteq q_1$, we have $q_1 \overset{\hat{\alpha}}{\Longrightarrow} q'_1$ for some q'_1 such that $p'_1 \sqsubseteq q'_1$. Hence, $(q_1, p_2) \overset{\hat{\alpha}}{\Longrightarrow} (q'_1, p_2)$ by repeated application of (PMay1) and since $\alpha \notin A_2$. Note that $(q_1, p_2) \notin E_Q$, since $q_1 \parallel p_2$ is defined by assumption. Therefore, if any state along $(q_1, p_2) \overset{\hat{\alpha}}{\Longrightarrow} (q'_1, p_2)$ is in E_Q, then we have $\alpha \in I$ and $(q_1, p_2) \overset{\alpha}{\Longrightarrow} u_Q$. Then, we are done by $(p_1 \parallel p_2, u_Q) \in \mathcal{R}$; otherwise, we have $q_1 \parallel p_2 \overset{\hat{\alpha}}{\Longrightarrow} q'_1 \parallel p_2$ with $(p'_1 \parallel p_2, q'_1 \parallel p_2) \in \mathcal{R}$.

(PMay2) $p_2 \overset{\alpha}{\dashrightarrow} p'_2$, $\alpha \notin A_1$ and $p'_1 = p_1$. Then, $(q_1, p_2) \overset{\alpha}{\dashrightarrow} (q_1, p'_2)$ by (PMay2) and due to $p_1 \sqsubseteq q_1$. If the latter state (q_1, p'_2) were in E_Q, then we either have $\alpha \in I$ and $(q_1, p_2) \overset{\alpha}{\dashrightarrow} u_Q$, or $(q_1, p_2) \in E_Q$. In the first case we are done and in the second case $q_1 \parallel p_2$ would not be defined. Thus, we have $q_1 \| p_2 \overset{\alpha}{\dashrightarrow} q_1 \| p'_2$ and $(p_1 \| p'_2, q_1 \| p'_2) \in \mathcal{R}$.

(PMay3) $\alpha = a$, $p_1 \overset{a}{\dashrightarrow} p'_1$ and $p_2 \overset{a}{\dashrightarrow} p'_2$ for some action $a \in A_1 \cap A_2$. Due to $p_1 \sqsubseteq q_1$, we get $q_1 \overset{\varepsilon}{\Longrightarrow} q''_1 \overset{a}{\dashrightarrow} q'''_1 \overset{\varepsilon}{\Longrightarrow} q'_1$ for some $q'_1, q''_1, q'''_1 \in Q$ such that $p'_1 \sqsubseteq q'_1$. Now, we obtain $(q_1, p_2) \overset{\varepsilon}{\Longrightarrow} (q''_1, p_2) \overset{a}{\dashrightarrow} (q'''_1, p'_2) \overset{\varepsilon}{\Longrightarrow} (q'_1, p'_2)$ by (PMay1) and (PMay3). Hence, $q_1 \parallel p_2 \overset{a}{\Longrightarrow} q'_1 \parallel p'_2$ and $(p'_1 \parallel p'_2, q'_1 \parallel p'_2) \in \mathcal{R}$ or $q_1 \parallel p_2 \overset{a}{\Longrightarrow} u_Q$, as in Case (PMay1) above. □

We close this subsection on parallel composition with a discussion of legal environments as introduced for IA in [27]. Intuitively, a legal (or helpful) environment for a composition $P \otimes Q$ is a MIA V that prevents $P \otimes Q$ from running into an error. In the final application, the parallel composition is embedded in such a legal environment, which may for example represent a user.

In [27], the concept of legal environments is used to justify the pruning, by showing that two systems are compatible if and only if there is a legal environment for them. The intuition is that two IA are only compatible if there is an environment that can use them without producing errors. If there is no such environment, the parallel composition may as well be undefined. Note that, two IAs are incompatible if their parallel composition is not defined due to the initial state being removed by pruning. Correspondingly, two MIAs are incompatible if the initial state of the parallel composition contains u due to pruning.

Definition 4.16 (Legal Environment)**.** *A* legal environment *for MIAs P and Q is a MIA V with:*

1. V is composable with $P \otimes Q$,

2. *$I_{(P \otimes Q) \otimes V} = \emptyset$,*

3. *The reachable states of $(P \otimes Q) \otimes V$ contain neither new nor inherited errors in the sense of Lem. 4.11.*

Note that, since $(P \otimes Q) \otimes V$ only has locally controlled actions, all reachable errors are locally reachable. Usually, in frameworks with binary communication, an environment is defined to have the outputs of $P \otimes Q$ as inputs and vice versa; thus, their composition is closed. Here, such a signature results in the product only having output and internal actions, which is natural for multicasting such as ours. One can then close the system with hiding all outputs. To generalize our additional result in Prop. 4.18, we only require that composition with an environment results in a system without inputs.

This notion of legal environment provides a characterization of compatibility:

Proposition 4.17. *MIAs P and Q are compatible if and only if there exists a legal environment for them.*

Proof. '$\Rightarrow$': If P and Q are compatible, then $P \otimes Q$ has no locally reachable errors. Composing it with a MIA V that accepts all inputs (via must-loops at the initial state) but provides no outputs, yields only those states that are locally reachable from (p_0, q_0). Thus V is a legal environment for P and Q.

'$\Leftarrow$': Assume, towards a contradiction, that P and Q are not compatible, i.e. $P \otimes Q$ has a locally reachable error in (p, q). Then, for any MIA V, $(P \otimes Q) \otimes V$ either has a reachable state $((p, q), v)$ resulting in an inherited error, or there is a first output transition on the path to (p, q) which V prevents by not providing the corresponding input transition. This results in a locally reachable new error in $(P \otimes Q) \otimes V$. Either way V is no legal environment. □

In terms of justification, we can do better than this. We show that pruning only removes behaviour from $P \otimes Q$ that is never reached in any legal environment, i.e. $(P \otimes Q) \otimes V = (P \parallel Q) \otimes V$. Our result shows that pruning does not change the behaviour when the compostion is used properly, i.e. in a legal environment. Note that our result is actually more general, since we do not need Condition 2 on Def. 4.16.

Proposition 4.18. *For MIAs P and Q and a corresponding legal environment V, we have $(P \otimes Q) \otimes V = (P \parallel Q) \otimes V$ (up to the names of the respective universal states).*

Proof. Due to Def. 4.16, the pruning in Lem. 4.11 does not change $(P \otimes Q) \otimes V$ and its universal state is unreachable. Furthermore, it equals $(P \parallel Q) \parallel V$ (up to name of the universal state). Since the universal state is unreachable in $(P \parallel Q) \parallel V$, pruning of $(P \parallel Q) \otimes V$ left the MIA unchanged and the claim follows. □

4.3.2 Universal States in Input/Output Approaches

States that, like our universal state u, represent arbitrary behaviour and have only input transitions as ingoing transitions date back at least to the thesis of Dill [32]. His work is focussed on a trace-based semantics, consisting of a set of ordinary traces and a set of so-called failure traces. The latter deal with behaviour resulting from communication mismatches. LTS-representations of the semantics have a special state that has arbitrary behaviour due to looping transitions for all actions. This state *completes* the LTS by making the other states input-enabled, and it is not an ordinary state since it is the only one representing the failure traces. The notion of input-enabledness is purely syntactic: a state s is input-enabled if it has an outgoing i-transition (must or may in case of modalities), for each input i. Considering an LTS in [32], an i-transition from s to the special state indicates that s cannot safely receive this input. This is just the same as a missing input in IA, and the LTS really *is* an IA. The representation based on a special state is just more convenient for a trace-based semantics expressing that, as in standard IA, a missing input can always be added in a refinement step.

A similar completion can be found in a process-algebraic setting in [31], where it is called *demonic*: if a process p does not have an i-transition according to the standard operational rules, then there are additional rules that give p an i-transition to a process having arbitrary behaviour, essentially due to loops. The latter process is an ordinary process, and communication mismatches are not considered. A variant of demonic completion is also used in [67] to achieve compositionality for parallel composition in the *ioco*-approach to conformance testing; this approach also disregards communication mismatches. The suggestion is to apply the completion to the specification first, each time the ioco-implementation relation is checked. That the completion uses ordinary states makes sense only because ioco does not support stepwise refinement and assumes that implementations are always input-enabled. Applying the suggested solution in IA would force each refinement to be input-enabled, violating the very idea of IA.

This problem with ordinary universal states in an IA-approach can be fixed as in [22, 16], where universal states are called *error states*. The *semantics* and the special treatment of error states in these papers is similar to the one in [32], but error states do not necessarily complete an IA and do not need loops. They arise in case of a communication mismatch in a parallel composition, just as in this chapter. The problem with ordinary universal states vanishes when modalities are added, since input-transitions to the universal state and the loops at this state can be declared to be of may-modality. Completion with this idea is used in [42] when translating IA with their refinement relation to MTSs. There, an input may-transition always expresses that the respective input is allowed in a refinement, but at present the input cannot be received safely.

As already discussed above, an ordinary state $\mathbb{tt}$ with may-loops is inserted during parallel composition in [59] as target for input transitions that have been cut due to pruning. This way, a precongruence is achieved, and it works fine for refinement that $\mathbb{tt}$ is regarded as an ordinary state. However, parallel composition is not associative this way; to avoid this problem, we insert u during parallel composition *and* give it a special treatment in refinement. It is important to note that we do *not* perform completion, i.e., for some ordinary state, an input can also be forbidden in all refinements, in accordance with the MTS-view.

4.3.3 Hiding, Restriction and Relabelling

We now turn to relabelling, hiding and restriction, which are common operators in process algebra [39, 57]. While relabelling is rather straightforward, we define hiding only for outputs and restriction only for inputs. The intuition behind this is that both operators block communication channels. Since outputs are under the control of the system, they are still performed, even if the signal no longer reaches the environment. It still can represent internal communication within components of the system. Inputs however can only be performed as a result of an outside stimulus. Once the signal is blocked the system cannot spontaneously decide to receive it. The same concept appeared in the IA-like setting of [23], albeit as a combined operator.

Definition 4.19 (Relabelling). *Let $P = (P, I, O, \longrightarrow_P, \dashrightarrow_P, p_0, u)$ be a MIA. We call a function $f : \Sigma \to \Sigma$ a* relabelling function*. If $f(I) \cap f(O) = \emptyset$, then the* relabelling of *P is defined as $P[f_I, f_O] =_{df} (P, f(I), f(O), \longrightarrow_{P[f_I,f_O]}, \dashrightarrow_{P[f_I,f_O]}, p_0, u)$ where all transition labels a are replaced by $f(a)$.*

Note that when relabelling some a to b, both can be foreign actions, but neither can be ν_I or ν_O.

Definition 4.20 (Restriction). *Given a MIA $P = (P, I, O, \longrightarrow_P, \dashrightarrow_P, P_0, u)$ and a set $L \subseteq \Sigma \setminus O$. Then,* restricting *$L$ in P yields the MIA $P \setminus L =_{df}$*

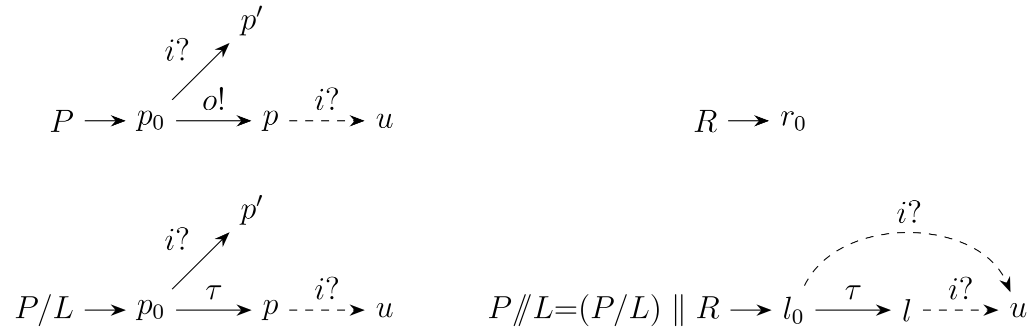

Figure 4.9: Problem with standard hiding. $L = \{o\}$ and $\mathcal{A}_R = \emptyset$.

$(P, I \backslash L, O, \longrightarrow_{P \backslash L}, \dashrightarrow_{P \backslash L}, P_0, u)$*, where all transitions with a label contained in L are deleted.*

Definition 4.21 (Hiding)**.** *Given a MIA* $P = (P, I, O, \longrightarrow_P, \dashrightarrow_P, P_0, u)$ *and a set* $L \subseteq \Sigma \setminus I$*. Then,* hiding L *in* P *is the MIA* $P/L =_{df} (P, I, O \setminus L, \longrightarrow_{P/L}, \dashrightarrow_{P/L}, P_0, u)$*, where all transition labels* $o \in L$ *are replaced by* τ*.*

The definition of hiding is natural and standard, but a usual equivalence fails, namely $(P/L) \parallel R \sqsupseteq\sqsubseteq (P \parallel R)/L$ for $L \cap \mathcal{A}_R = \emptyset$. Consider P and R from Fig. 4.9. $P \parallel R$ and P are equal up to the names of states, hence so are $(P \parallel R)/L$ and P/L. Here, $(P/L) \parallel R$ (also depicted) is not MIA-equivalent to $(P \parallel R)/L$, since the first is missing an initial i-must-transition.

A solution for this is to perform during hiding the same normalization as results from our pruning during parallel composition: for $P/\!\!/L$ we replace all labels in L by τ and identify states p that then can reach the universal state using τ-transitions and one final input i ($p \overset{\varepsilon}{\Longrightarrow}\overset{i}{\dashrightarrow} u_P$). From such a state we replace all outgoing i-transitions (if present) by a single one to the universal state (cf. Fig. 4.9).

Definition 4.22 (Hiding with pruning)**.** *Given a MIA* $P = (P, I, O, \longrightarrow_P, \dashrightarrow_P, P_0, u)$ *and a set* $L \subseteq \Sigma \backslash I$*. Then,* P norm-hiding L *is the MIA* $P/\!\!/L =_{df} (P, I, O \setminus L, \longrightarrow_{P/\!\!/L}, \dashrightarrow_{P/\!\!/L}, P_0, u)$*, where the transitions are changed as follows: from each state* p *with* $p \overset{w}{\Longrightarrow}\overset{i}{\dashrightarrow} u_P$ *and* $p \overset{i}{\dashrightarrow}$ *(with* $w \in L^*$ *and* $i \in I$*), all outgoing* i*-transitions are removed and one transition* $p \overset{i}{\dashrightarrow}_{P/\!\!/L} u_P$ *is added. Each label* $o \in L$ *is replaced by* τ*.*

Observe that all four of the above operators yield well-defined MIAs; in particular, the sink condition is preserved by hiding (with and without pruning) since $L \cap I = \emptyset$.

Lemma 4.23 (Weak Must-Transitions under Hiding)**.** *Let P be a MIA, $L \cap I = \emptyset$ and $o \in L \cap O$. If $p \overset{o}{\Longrightarrow}_P P'$, then $p \overset{\varepsilon}{\Longrightarrow}_{P/L} P'$ and $p \overset{\varepsilon}{\Longrightarrow}_{P/\!\!/L} P'$.*

Proof. By induction on the definition of $p \overset{o}{\Longrightarrow}_P P'$. If $p \overset{o}{\Longrightarrow}_P P'$ is due to Def. 4.2.3, then the claim is obvious. Otherwise, $p \overset{o}{\Longrightarrow}_P P'$ is due to some $p \overset{\tau}{\longrightarrow}_P \bar{P}$ and $\bar{P} \overset{o}{\Longrightarrow}_P P'$ according to Def. 4.2.2. By induction hypothesis, we have $\bar{p} \overset{\varepsilon}{\Longrightarrow}_{P/L} P_{\bar{p}}$ for each $\bar{p} \in \bar{P}$ and $P' = \bigcup_{\bar{p} \in \bar{P}} P_{\bar{p}}$. By Def. 4.2.2, we obtain $p \overset{\varepsilon}{\Longrightarrow}_{P/L} P'$. The same holds for $P/\!\!/L$, since it differs from P/L only on input transitions. □

As desired, MIA-refinement is a precongruence w.r.t. relabelling, restriction and hiding:

Proposition 4.24. *Let P, Q be MIAs with $P \sqsubseteq Q$ and L be an appropriate set, then:*

1. *$P[f_I, f_O] \sqsubseteq Q[f_I, f_O]$ for an appropriate relabelling function f.*

2. *$P \setminus L \sqsubseteq Q \setminus L$.*

3. *$P/L \sqsubseteq Q/L$.*

4. *$P/\!\!/L \sqsubseteq Q/\!\!/L$.*

Proof. The proofs for 1. and 2. are straightforward and 3. is very similar to but simpler than 4. We show the last explicitly:

Since $P \sqsubseteq Q$, there is a MIA-refinement relation $\mathcal{R}'$. We show that $\mathcal{R} = \mathcal{R}' \cup \{(p, u_{Q/\!\!/L})\}$ is a MIA-refinement relation. We consider some $(p, q) \in \mathcal{R}$ with $q \neq u_{Q/\!\!/L}$ (i.e. in the first subset) and check the points of Def. 4.4.

(i) $q \neq u_{Q/\!\!/L}$ implies $q \neq u_Q$ by construction. Due to $p \sqsubseteq q$ we get $p \neq u_P$ and again by construction $p \neq u_{P/\!\!/L}$.

(ii) $q \overset{i}{\longrightarrow}_{Q/\!\!/L} Q'$: by construction $q \overset{i}{\longrightarrow} Q'$ and thus, there exists a P' matching Q' with $p \overset{i}{\longrightarrow}\overset{\varepsilon}{\Longrightarrow}_{P/\!\!/L} P'$. If $p \overset{i}{\longrightarrow}_{P/\!\!/L}$ then $p \overset{i}{\longrightarrow}\overset{\varepsilon}{\Longrightarrow}_{P/\!\!/L} P'$ and we are done. Otherwise the i-transition was removed during pruning due to $p \overset{w}{\Longrightarrow}\overset{i}{\dashrightarrow}_P u_P$ with $w \in L^*$. Because of $p \sqsubseteq q$ we get $q \overset{w}{\Longrightarrow}\overset{i}{\dashrightarrow}_Q u_Q$. Thus all i-transitions originating from q are removed during pruning and $q \overset{i}{\not\longrightarrow}_{Q/\!\!/L}$ which contradict our assumptions for this case.

(iii) $q \xrightarrow{\omega}_{Q/\!\!/L} Q'$: if $\omega \in O \setminus L$ or $\omega = \tau$ and $q \xrightarrow{\tau}_{Q} Q'$ the claim can be shown analogously to (ii) without the possibility of pruning. Otherwise, we have $q \xrightarrow{\tau}_{Q/\!\!/L} Q'$ due to $q \xrightarrow{o}_{Q} Q'$ with $o \in L$. Then there is some P' matching Q' in $\mathcal{R}$ and $p \overset{o}{\Longrightarrow} P'$. By Lem. 4.23 we get $p \overset{\varepsilon}{\Longrightarrow}_{P/\!\!/L} P'$ and are done.

(iv) $p \overset{\alpha}{\dashrightarrow}_{P/\!\!/L} p'$: the first possibility is that $\alpha \in \mathcal{A} \cup \{\nu_I, \nu_O\}$ or $\alpha = \tau$ and $p \overset{\tau}{\dashrightarrow}_{P} p'$; either way $p \overset{\alpha}{\dashrightarrow}_{P} p'$. Then $q \overset{\hat{\alpha}}{\Longrightarrow}_{Q} q'$ with $(p', q') \in \mathcal{R}$. If the respective α-transition is pruned during hiding, then $q \overset{\alpha}{\Longrightarrow}_{Q/\!\!/L} u_{Q/\!\!/L}$ and we are done since $(p', u_{Q/\!\!/L}) \in \mathcal{R}$. Otherwise, we have $q \overset{\hat{\alpha}}{\Longrightarrow}_{Q/\!\!/L} q'$ and $(p', q') \in \mathcal{R}$. The second possibility is that $\alpha = \tau$ and $p \overset{\tau}{\dashrightarrow}_{P/\!\!/L} p'$ due to $p \overset{o}{\dashrightarrow}_{P} p'$ and $o \in L$. Then $q \overset{o}{\Longrightarrow}_{Q} q'$ with $(p', q') \in \mathcal{R}'$ and thus $q \overset{\tau}{\Longrightarrow}_{Q/\!\!/L} q'$ and $(p', q') \in \mathcal{R}$. □

MIA-refinement indeed satisfies the law described above for our modified hiding. The completely new proof is fairly hard.

Proposition 4.25. *Let P, Q be MIAs with $P \sqsubseteq Q$ and $L \subseteq \Sigma \setminus (I_P \cup \mathcal{A}_Q)$. Then $(P/\!\!/L) \parallel Q$ is equal to $(P \parallel Q)/\!\!/L$.*

Proof. Observe that the parallel products $(P/\!\!/L) \otimes Q$ and $P \otimes Q$ have the same states. The set of illegal states is also the same: firstly, no error is caused by $o \in L$ since, by assumption, o is not synchronising. Secondly, pruning during $/\!\!/$ at worst removes input must-transitions from P, thus all errors in the second product are also errors in the first. The additional errors of $(P/\!\!/L) \otimes Q$ can only arise from a state (p, q) where an $a \in I_P \cap O_Q$ was cut at p and $q \overset{a}{\dashrightarrow}$. This implies $p \overset{w}{\Longrightarrow} p' \overset{i}{\dashrightarrow} u$ for some $w \in (L \cap O_P)^*$. Since $i \in O_Q$, (p', q) is an error and (p, q) is an illegal state of $P \otimes Q$. Thirdly, the definition of illegal states treats outputs and τs the same, thus, it is not affected by hiding. Thus, and since hiding does not change the set of states, the states of the final systems $(P/\!\!/L) \parallel Q$ and $(P \parallel Q)/\!\!/L$ are the same.

Consider some (p, q) appearing in both final systems $(P/\!\!/L) \parallel Q$ and $(P \parallel Q)/\!\!/L$ and an α-transition of (p, q) in either system. For ease of notation we denote the states of $P/\!\!/L$ by p_L and of $(P \parallel Q)/\!\!/L$ by $p \parallel q_L$ in the remainder of this proof; we also will write l for an illegal state and l_L for the same state after hiding.

If the transition is internal due to hiding some $o \in O_P \cap L$, it is a non-synchronising output of $P \parallel Q$ and thus the same in both final systems. Similarly, if $\alpha = \tau$ and not due to hiding or if α is a non-synchronising output of either system, it is not changed by hiding and treated the same by both parallel compositions. Synchronising outputs of P are also treated the same, since cutting only concerns inputs (and illegal states are not considered any more). The latter argument is also true for a synchronising output of Q, i.e. $a \in O_Q \cap I_P$, if it is not cut at p during hiding of P. If cut, we have $p \overset{w}{\Longrightarrow} p' \overset{a}{\dashrightarrow} u$ with $w \in L^*$ and $p \overset{a}{\dashrightarrow}$; also $q \overset{a}{\dashrightarrow} q'$, otherwise there is no transition in the final systems. Then, on the one hand $p_L \overset{a}{\dashrightarrow} u$ and, hence $(p_L, q) \in E$, which is a contradiction. On the other hand, $(p, q) \overset{w}{\Longrightarrow} (p', q) \overset{a}{\dashrightarrow} (u, q')$ in $P \otimes Q$, thus $(p, q) \in E$, a contradiction again.

Inputs of the final systems that are not cut during any pruning procedure are the same. Thus, it only remains to consider the inputs of the final systems that are cut during parallel composition or during hiding in either system.

Consider an input i cut at $p \parallel q_L$ during hiding of $(P \parallel Q)/\!\!/L$, i.e. $p \parallel q_L \overset{i}{\dashrightarrow} u$ due to $p \parallel q \overset{w}{\Longrightarrow} p' \parallel q' \overset{i}{\dashrightarrow} u$ with some $w \in L^*$ and $p \parallel q \overset{i}{\dashrightarrow}$. Considering backward propagation of cutting, we then have $p \parallel q \overset{w}{\Longrightarrow} p' \parallel q' \overset{\varepsilon}{\Longrightarrow} p'' \parallel q'' \overset{i}{\dashrightarrow} l$ for some illegal state $l \in E$ and $p \parallel q \overset{i}{\dashrightarrow}$, implying $p_L \parallel q \overset{\varepsilon}{\Longrightarrow} p'_L \parallel q' \overset{\varepsilon}{\Longrightarrow} p''_L \parallel q'' \overset{i}{\dashrightarrow} l_L$ (where l_L might also be an inherited error due to cutting while hiding) and $p_L \parallel q \overset{i}{\dashrightarrow}$. Thus all i-transitions of $p_L \parallel q$ are cut during parallel composition and we get $p_L \parallel q \overset{i}{\dashrightarrow} u$.

If i is cut at $p \parallel q$ during parallel composition of $P \parallel Q$, we have $p \parallel q \overset{\varepsilon}{\Longrightarrow} p'' \parallel q'' \overset{i}{\dashrightarrow} l$ for some illegal state $l \in E$ and $p \parallel q \overset{i}{\dashrightarrow}$. Similarly to the above, we get $p_L \parallel q \overset{\varepsilon}{\Longrightarrow} p''_L \parallel q'' \overset{i}{\dashrightarrow} l_L$ and $p_L \parallel q \overset{i}{\dashrightarrow}$. Again, all i-transitions of $p_L \parallel q$ are cut and $p_L \parallel q \overset{i}{\dashrightarrow} u$.

Now consider an input of P cut during $P/\!\!/L$. We have already covered $i \in I_P \cap O_Q$ showing that the states (p_L, q) and (p, q) are illegal and thus removed in the final systems. Otherwise, $p \overset{w}{\Longrightarrow} p' \overset{i}{\dashrightarrow} u$ with $w \in L^*$ and $p \overset{i}{\dashrightarrow}$ and again $q \overset{i}{\dashrightarrow} q'$. Then on the one hand $p_L \overset{i}{\dashrightarrow} u$ and hence, during parallel composition, i-transitions of (p_L, q) are cut. On the other hand, $p \parallel q \overset{w}{\Longrightarrow} p' \parallel q \overset{i}{\dashrightarrow} u$ and $p \parallel q \overset{i}{\dashrightarrow}$, thus all i-transitions of $p \parallel q$ are cut during pruning.

Finally, consider an input i cut at $p_L \parallel q$ during parallel composition of $(P/\!\!/L) \parallel Q$. This implies $p_L \parallel q \overset{\varepsilon}{\Longrightarrow} p'_L \parallel q' \overset{i}{\dashrightarrow} l_L$ and $p_L \parallel q \overset{i}{\dashrightarrow}$. If $i \in I_P$ is cut at p'_L during $P/\!\!/L$ it is cut at p_L as well, and this is covered by the previous case; otherwise, we deduce $p \parallel q \overset{\varepsilon}{\Longrightarrow} p' \parallel q' \overset{i}{\dashrightarrow} l$ and $p \parallel q \overset{i}{\dashrightarrow}$. Thus, all i-transitions of $p \parallel q$ are cut and $p \parallel q \overset{i}{\dashrightarrow} u$, just as with $p_L \parallel q$. □

4.3.4 Parallel Composition with Hiding

We now turn our attention to parallel composition with immediate hiding on synchronized actions, thereby enforcing binary communication. This parallel composition is used by de Alfaro and Henzinger for Interface Automata (IA) in [25, 27]. We show here that the standard IA parallel composition can be expressed via our multicast parallel composition and hiding.

Definition 4.26 (Parallel Product and Composition with Hiding)**.** *MIAs P_1 and P_2 are* H-composable *if $O_1 \cap O_2 = \emptyset = I_1 \cap I_2$. We then define the* product with hiding *in the same way as the parallel product in Def. 4.8, except for $O =_{df} (O_1 \cup O_2) \setminus (I_1 \cup I_2)$ and a change of Rules (PMust3) and (PMay3):*

(PMust3') $(p_1, p_2) \xrightarrow{\tau} P_1' \times P_2'$ *if* $p_1 \xrightarrow{a} P_1'$ *and* $p_2 \xrightarrow{a} P_2'$ *for some* a*,*
(PMay3') $(p_1, p_2) \overset{\tau}{\dashrightarrow} (p_1', p_2')$ *if* $p_1 \overset{a}{\dashrightarrow} p_1'$ *and* $p_2 \overset{a}{\dashrightarrow} p_2'$ *for some* a*.*

From this parallel product with hiding, we get the parallel composition with hiding *$P_1 \mid P_2$ by the same pruning procedure as in Def. 4.9.*

It can easily be seen that the parallel product with hiding can be expressed by our parallel product without hiding and the hiding operator. Pruning does not change this, since it treats outputs and internal actions equally.

Proposition 4.27. *Let P_1, P_2 be H-composable MIAs and $S = A_1 \cap A_2$ be the set of synchronising actions. Then, $P_1 \mid P_2 = (P_1 \parallel P_2)/S$.*

Associativity is a natural property of parallel composition, so one would expect that $(P \mid Q) \mid R = P \mid (Q \mid R)$ for some suitable equivalence $=$ (e.g., equality up to isomorphism) provided that one side is defined. This law looks much less natural if we rewrite it according to Prop. 4.27; it is wrong in the version of $\mid$ in [25]. Here, associativity can be proved from Thm. 4.12 and the following proposition.

Proposition 4.28. *For composable MIAs P and Q we have the following laws, where $=$ means that the respective MIAs are identical (up to the naming of the respective universal states in Part (iii)).*

(i) $P/L = P$ *if* $A_P \cap L = \emptyset$*.*

(ii) $P/L/L' = P/(L \cup L')$ *if* $L \cap I_P = L' \cap I_P = \emptyset$*.*

(iii) $(P \parallel Q)/L = (P/L) \parallel (Q/L)$ *if* $A_P \cap A_Q \cap L = \emptyset$*.*

Proof. Parts (i) and (ii) are straightforward. We thus focus on proving Part (iii). $P \otimes Q$ and $P/L \otimes Q/L$ are the same due to the condition $A_P \cap A_Q \cap L$, except that transition labels $o \in L$ in the former are replaced by τ in the latter; observe that (PMust3) and (PMay3) are never applicable to $o \in L$ by assumption, and the other rules work for $o \in L$ and τ in the same way. Also by assumption, the same states are considered as errors in both products. As a consequence and since pruning makes no difference between output- and τ-transitions, it deletes the same states in both systems and the same input transitions get redirected to the respective universal states of the parallel compositions. Finally, applying hiding to $P \parallel Q$ for the first system makes the MIAs identical. □

Using this proposition we may now prove the associativity of $|$.

Proposition 4.29. *Parallel composition with hiding is associative in the sense, that for pairwise H-composable MIAs P, Q and R, if $(P \mid Q) \mid R$ is defined, then $P \mid (Q \mid R)$ is defined as well and both are isomorphic, and vice versa.*

Proof. Let P, Q, R be pairwise H-composable MIAs. We set $S_{PQ} =_{df} A_P \cap A_Q$, $A_{PQ} =_{df} (A_P \cup A_Q) \setminus S_{PQ}$, etc. and let $S_{PQR} =_{df} S_{PQ} \cup S_{PR} \cup S_{QR}$. Note that $(*)$ $S_{PQ} \cap A_R = \emptyset$ since, otherwise A_R would contain an action that is an input in one of P and Q and an output in the other, contradicting H-composability of R with one of the other MIAs. Furthermore, $(**)$ $S_{PQ} \cup (A_{PQ} \cap A_R) = S_{PQ} \cup (((A_P \cup A_Q)/S_{PQ}) \cap A_R) \overset{(*)}{=} S_{PQ} \cup (((A_P \cup A_Q) \cap A_R)/S_{PQ}) = S_{PQ} \cup ((A_P \cup A_Q) \cap A_R) = S_{PQ} \cup (A_P \cap A_R) \cup (A_Q \cap A_R) = S_{PQR}$. We now obtain:

$$\begin{aligned}
(P \mid Q) \mid R &= ((P \parallel Q)/S_{PQ} \parallel R)/(A_{PQ} \cap A_R) && \text{(Prop. 4.27)}\\
&= ((P \parallel Q)/S_{PQ} \parallel R/S_{PQ})/(A_{PQ} \cap A_R) && \text{(Prop. 4.28(i) and } (*))\\
&= ((P \parallel Q) \parallel R)/S_{PQ}/(A_{PQ} \cap A_R) && \text{(Prop. 4.28(iii) and } (*))\\
&= ((P \parallel Q) \parallel R)/S_{PQR} && \text{(Prop. 4.28(ii) and } (**))\\
&= (P \parallel (Q \parallel R))/S_{PQR} && \text{(Thm. 4.12)}\\
&= P \mid (Q \mid R) && \text{(symmetrically)} \quad \square
\end{aligned}$$

4.4 Logical Operators – Conjunction and Disjunction

The main feature of our heterogeneous setting is that it features not only structural, but also logical operators. It supports not only conjunction and disjunction, but also temporal logics. We start with disjunction, which, as mentioned before, is rather straightforward, since we allow for multiple initial states: Assuming w.l.o.g. that the disjuncts have the same universal state, the disjunction is simply the union of the disjuncts.

Definition 4.30 (Disjunction)**.** *MIAs P and Q are u-*disjoint*, if their state spaces are disjoint, except for their common universal state, i.e. $P \cap Q = \{u_P\} = \{u_Q\}$.*

Two given MIAs are w.l.o.g. u-disjoint MIAs $P = (P, I, O, \longrightarrow_P, \dashrightarrow_P, P_0, u)$ and $Q = (Q, I, O, \longrightarrow_Q, \dashrightarrow_Q, Q_0, u)$, and we define $P \vee Q$ as $(P \cup Q, I, O, \longrightarrow_P \uplus \longrightarrow_Q, \dashrightarrow_P \uplus \dashrightarrow_Q, P_0 \cup Q_0, u)$.

The proof that $\vee$ is indeed or and compositional can be taken from [20].

Theorem 4.31 ($\vee$ is or)**.** *For all MIAs P, Q and R, we have $P \vee Q \sqsubseteq R$ if and only if $P \sqsubseteq R$ and $Q \sqsubseteq R$.*

Proof. By definition, $P \vee Q \sqsubseteq R$ means that for each $p_0 \in P_0$ and each $q_0 \in Q_0$ there exist $r_0, r_0' \in R_0$ such that $p_0 \sqsubseteq r_0$ and $q_0 \sqsubseteq r_0'$, which is equivalent to $P \sqsubseteq R$ and $Q \sqsubseteq R$. □

Corollary 4.32 (Compositionality of $\vee$)**.** *MIA-refinement is compositional w.r.t. disjunction. $P_1 \sqsubseteq Q_1$ and $P_2 \sqsubseteq Q_2$ implies $P_1 \vee P_2 \sqsubseteq Q_1 \vee Q_2$ for all MIAs P_1, P_2, Q_1 and Q_2.*

Besides parallel composition and quotienting, conjunction is one of the most important operators of interface theories. It allows one to specify different perspectives of a system separately, from which an overall specification can be determined by conjunctive composition. More formally, the conjunction should be the coarsest specification that refines the given perspective specifications, i.e., it should characterise the greatest lower bound of the refinement preorder. Similarly to parallel composition, the construction is defined in two steps: first a conjunctive product $P \& Q$ is calculated via operational rules; the treatment of multiple initial states is straightforward. In a second step inconsistencies are identified and removed. The remaining states (p, q) of $P \wedge Q$ are written as $p \wedge q$. This construction may remove all initial states if P and Q are inconsistent, i.e. have no common refinement, resulting in an empty MIA.

Definition 4.33 (Conjunctive Product)**.** *Consider MIAs $(P, I, O, \longrightarrow_P, \dashrightarrow_P, P_0, u_P)$ and $(Q, I, O, \longrightarrow_Q, \dashrightarrow_Q, Q_0, u_Q)$ with common alphabets. The* conjunctive product *is defined as $P \& Q =_{df} (P \times Q, I, O, \longrightarrow, \dashrightarrow, P_0 \times Q_0)$, $(u_P, u_Q))$ by the following operational transition rules and their symmetric counterparts:*

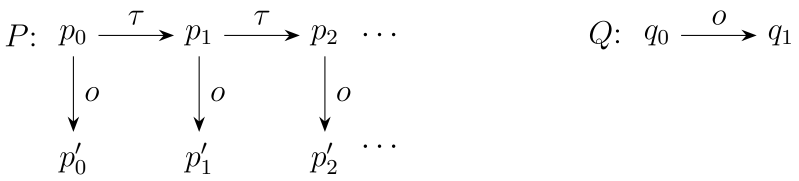

Figure 4.10: Example of a conjunction leading to a transition with an infinite target set.

$$
\begin{array}{lll}
\text{(Must1)} & (p,q) \xrightarrow{\alpha} & \\
 & \quad \{(p',q') \,|\, p' \in P',\ q \overset{\hat{\alpha}}{\Longrightarrow}_Q q'\} & \textit{if } p \xrightarrow{\alpha}_P P',\ q \overset{\hat{\alpha}}{\Longrightarrow}_Q,\ q \neq u_Q \\
\text{(UMust1)} & (p,u_Q) \xrightarrow{\alpha} P' \times \{u_Q\} & \textit{if } p \xrightarrow{\alpha}_P P' \\
\text{(May1)} & (p,q) \overset{\alpha}{\dashrightarrow} (p',q') & \textit{if } p \overset{\alpha}{\Longrightarrow}_P p' \textit{ and } q \overset{\hat{\alpha}}{\Longrightarrow}_Q q' \\
\text{(UMay1)} & (p,u_Q) \overset{\alpha}{\dashrightarrow} (p',u_Q) & \textit{if } p \overset{\alpha}{\dashrightarrow}_P p'
\end{array}
$$

We refer to Rules (Must1) and its symmetric counterpart (Must2) collectively as (Must)-rules and analogously for the other rules. Note that ν_I and ν_O are also treated by the (May)- and (UMay)-rules. Note also that this definition is similar to the one in [53], except for the treatment of inputs and the universal state.

The weak transitions in the (Must)-Rules may lead to disjunctive transitions with infinite target sets, which were prohibited in [12]. For example, consider the conjunction of the MIAs P and Q depicted in Fig. 4.10. Infinitely many weak o-transitions start from p_0, yielding an infinite disjunctive o-transition at $p_0 \wedge q_0$ due to Rule (OMust2). We addressed this issue by generalising MIAs (cf. Def. 4.1) to allow infinite target sets of must-transitions and by adapting Def. 4.2 accordingly. Note that the aforementioned problem arises only for an infinite-state MIA (cf. P in Fig. 4.10) and is not a problem in practice, where MIAs are expected to be finite state.

The list of rules is short compared to the ones in previous versions (e.g. [13]). This is because we previously had to give separate rules for inputs and outputs for both may- and must- transitions and another one for τ-may-transitions. The relaxation of the refinement relation makes these separate treatments unnecessary. This may be surprising in the case of must-transitions, but note that the condition for adding a must-transition depends on weak may-transitions, which were also treated differently depending on whether they were labeled with an input or an output.

The conjunctive product is inherently different from the parallel product: single transitions are defined through weak transitions, e.g., as in Rules (Must), (May), and τ-transitions synchronise by Rule (May). Furthermore, as given by Rules (UMust) and (UMay), the universal states are neutral elements for the conjunctive product, whereas they are absorbing for the parallel product.

The following definition deals with inconsistencies: these arise if one conjunct requires an action that the other prohibits. Again, this definition is shorter than in previous versions of MIA, for the same reasons: for item (F1) we had to differentiate between inputs and outputs, requiring $q \overset{a}{\dashrightarrow}$ for the first and $q \overset{a}{\Longrightarrow}$ for the latter. Now it is $q \overset{a}{\Longrightarrow}$ for both.

Definition 4.34 (Conjunction)**.** *Given a conjunctive product $P\&Q$, the set $F \subseteq P \times Q$ of (logically)* inconsistent states *is defined as the least set satisfying the following rules for all $p \neq u_P$ and $q \neq u_Q$:*

(F1) $p \xrightarrow{a}_P$ *and not* $q \overset{a}{\Longrightarrow}_Q$ *implies* $(p,q) \in F$
(F2) *not* $p \overset{a}{\Longrightarrow}_P$ *and* $q \xrightarrow{a}_Q$ *implies* $(p,q) \in F$
(F3) $(p,q) \xrightarrow{\alpha} K'$ *and* $K' \subseteq F$ *implies* $(p,q) \in F$

The conjunction $P \wedge Q$ is obtained by deleting all states $(p,q) \in F$ from $P\&Q$. This also removes any may- or must-transition exiting a deleted state and any may-transition entering a deleted state; in addition, deleted states are removed from targets of disjunctive must-transitions. We write $p \wedge q$ for state (p,q) of $P \wedge Q$; all such states are defined *– and consistent – by construction. Note that, if $P_0 \times Q_0 \subseteq F$, then the conjunction of P and Q is an empty MIA, i.e. inconsistent.*

As mentioned, disjunctive transitions are essential to the conjunction operator. The (Must)-Rules generate them, even if the conjuncts themselves do not have any. To illustrate this and their treatment in the presence of inconsistencies, we recall an example and its discussion from [51, Fig. 5] in Fig. 4.11. There, it is presented for dMTS not MIA, but we can easily adapt it by designating all actions as outputs. Observe that, while P and Q have no truly disjunctive must-transitions, the a transitions yield one with three targets in $P\&Q$. Since $(4,6)$ is inconsistent (due to d) and the inconsistency propagates back to $(3,5)$ over c, both are removed. $p \wedge q$ remains, however, since $(3,5)$ was not the only target of its disjunctive transition.

Next, we prove that conjunction as defined above is the greatest lower bound w.r.t. MIA refinement. To this end, we introduce the notion of a witness as in [53]:

Definition 4.35 (Witness)**.** *A* witness *W of $P\&Q$ is a subset of $P \times Q$ such that the following conditions hold for all $(p,q) \in W$:*

(W1) $p \xrightarrow{a}_P$ *implies* $q \overset{a}{\Longrightarrow}_Q$ *or* $q = u_Q$
(W2) $q \xrightarrow{a}_Q$ *implies* $p \overset{a}{\Longrightarrow}_P$ *or* $p = u_P$
(W3) $(p,q) \xrightarrow{\alpha} K'$ *implies* $K' \cap W \neq \emptyset$

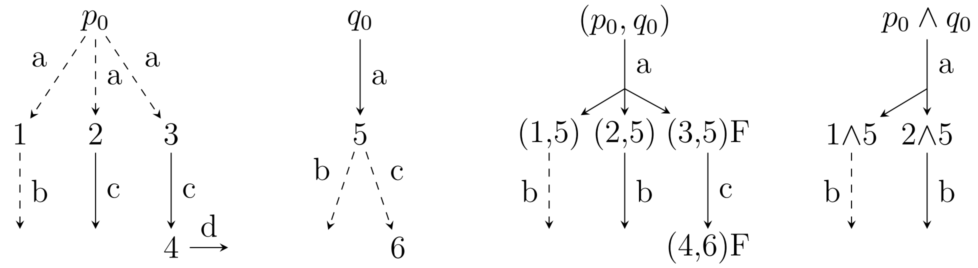

Figure 4.11: Disjunctive transitions during conjunction

Intuitively, a witness is a set of state pairs that ensure that none of them justifies (F1), (F2) or (F3) and, thus, the set witnesses their consistency.

Lemma 4.36 (Concrete Witness)**.** *Let P, Q and R be MIAs with common alphabets.*

1. *For any witness W of $P\&Q$, we have $F \cap W = \emptyset$.*

2. *The set $\{(p,q) \in P \times Q \mid \exists r \in R.\, r \sqsubseteq p$ and $r \sqsubseteq q\}$ is a witness of $P\&Q$.*

Proof. While the first statement of the lemma is quite obvious, we prove here that $W =_{df} \{(p,q) \in P \times Q \mid \exists r \in R.\, r \sqsubseteq p \text{ and } r \sqsubseteq q\}$ is a witness of $P\&Q$:

(W1) $p \xrightarrow{a}_P P'$ implies $r \overset{a}{\Longrightarrow}_R R'$ for some R' by $r \sqsubseteq p$. Choose some $r' \in R'$. Then, $r \overset{a}{=\!\Rightarrow}_R r'$ by syntactic consistency, and $q \overset{a}{=\!\Rightarrow}_Q$ or $q = u_Q$ by $r \sqsubseteq q$.

(W2) Analogous to (W1).

(W3) Consider $(p,q) \in W$ due to r. If $(p,q) \xrightarrow{\alpha} K'$ by (Must1), then, $p \xrightarrow{\alpha}_P P'$, $q \overset{\hat{\alpha}}{=\!\Rightarrow}$, $q \neq u_Q$ and $K' = \{(p',q') \mid p' {\in} P',\, q \overset{\hat{\alpha}}{=\!\Rightarrow}_Q q'\}$. By $r \sqsubseteq p$ and $p \neq u_P$, we get some $R' \subseteq R$ with $r \overset{\hat{\alpha}}{\Longrightarrow}_R R'$ and $\forall r' {\in} R'\, \exists p' {\in} P'.\, r' \sqsubseteq p'$. Choose $r' {\in} R'$; now, $r \overset{\hat{\alpha}}{=\!\Rightarrow}_R r'$ due to syntactic consistency, and $q \overset{\hat{\alpha}}{=\!\Rightarrow}_Q q'$ with $r' \sqsubseteq q'$ by $r \sqsubseteq q$. Thus, we have $p' \in P'$ and q' such that $(p',q') \in K' \cap W$ due to r'.

Otherwise, if $(p, u_Q) \xrightarrow{\alpha} K'$ by (UMust1), then, $p \xrightarrow{\alpha}_P P'$ and $K' = P' \times \{u_Q\}$. As before, by $r \sqsubseteq p$ and $p \neq u_P$, we get some $R' \subseteq R$ with $r \overset{\hat{\alpha}}{\Longrightarrow}_R R'$ and $\forall r' {\in} R'\, \exists p' {\in} P'.\, r' \sqsubseteq p'$. Choose $r' {\in} R'$; now, $r \overset{\hat{\alpha}}{=\!\Rightarrow}_R r'$ due to syntactic consistency, and obviously $r' \sqsubseteq u_Q$. Thus, we have $p' \in P'$ and $q' = u_Q$ such that $(p',q') \in K' \cap W$ due to r'. □

On the basis of this lemma we can now establish the desired greatest lower bound result for $\wedge$, which implies the compositionality of $\sqsubseteq$ w.r.t. $\wedge$ (cf. [53]).

Theorem 4.37 ($\wedge$ is and)**.** *For MIAs P, Q and R, we have:*

1. *There is some r with $r \sqsubseteq p$ and $r \sqsubseteq q$ if and only if $p \wedge q$ is defined.*
2. *If $p \wedge q$ is defined, then $r \sqsubseteq p$ and $r \sqsubseteq q$ if and only if $r \sqsubseteq p \wedge q$.*
3. *$R \sqsubseteq P$ and $R \sqsubseteq Q$ if and only if $R \sqsubseteq P \wedge Q$.*

Proof. 1. '$\Rightarrow$': We assume some r with $r \sqsubseteq p$ and $r \sqsubseteq q$. By Lem. 4.36.2, there exists a witness W containing (p, q). Thus, Lem. 4.36.1 implies $(p, q) \notin F$, i.e. $p \wedge q$ is defined.

1. and 2. '$\Leftarrow$': For reasons of symmetry, it suffices to show that $\mathcal{R} =_{df} \{(r, p) \mid \exists q. r \sqsubseteq p \wedge q\}$ is a MIA-refinement relation. Then, in particular, 1. '$\Leftarrow$' follows by choosing $r = p \wedge q$.

Note that (UMust1) and (UMay1) produce an isomorphic copy of P. The refinement conditions for states $(r, p) \in \mathcal{R}$ due to $q = u_Q$ hold by definition of $\mathcal{R}$, and we can ignore these rules in the rest of this proof.

We check the conditions of Def. 4.4 for some $(r, p) \in \mathcal{R}$ due to q, where $p \neq u_P$:

- $p \neq u_P$ implies $p \wedge q \neq u_P \wedge u_Q$. By $r \sqsubseteq p \wedge q$, we have $r \neq u_R$.
- Let $p \xrightarrow{\alpha}_P P'$; then, $q \overset{\hat{\alpha}}{=\!\Rightarrow}_Q$. For $\alpha \neq \tau$, this is because, otherwise, $p \wedge q$ would not be defined due to (F1); otherwise it is trivial. Hence, by (Must1), $p \wedge q \xrightarrow{\alpha} \{p' \wedge q' \mid p' \in P', q \overset{\hat{\alpha}}{=\!\Rightarrow}_Q q', p' \wedge q' \text{ defined}\}$. By $r \sqsubseteq p \wedge q$, we get $r \overset{\hat{\alpha}}{\Longrightarrow}_R R'$ ($r \xrightarrow{\alpha}\overset{\varepsilon}{\Longrightarrow} R'$ in case of $\alpha \in I$) such that $\forall r' {\in} R' \, \exists p' {\wedge} q'. \, p' \in P'$, $q \overset{\hat{\alpha}}{=\!\Rightarrow}_Q q'$ and $r' \sqsubseteq p' {\wedge} q'$. Thus, $\forall r' {\in} R' \, \exists p' {\in} P'. \, (r', p') \in \mathcal{R}$.
- $r \overset{\alpha}{\dashrightarrow}_R r'$ implies $\exists p' \wedge q'. \, p \wedge q \overset{\hat{\alpha}}{=\!\Rightarrow} p' \wedge q'$ and $r' \sqsubseteq p' \wedge q'$. The contribution of p in this weak transition sequence (cf. (May), (UMay1) and (UMay2)) gives $p \overset{\hat{\alpha}}{=\!\Rightarrow}_P p'$, and, thus, we have $(r', p') \in \mathcal{R}$ due to q'.

2. '$\Rightarrow$': Here, we show that $\mathcal{R} =_{df} \{(r, p \wedge q) \mid r \sqsubseteq p \text{ and } r \sqsubseteq q\}$ is a MIA-refinement relation. By Claim 1, $p \wedge q$ is defined whenever $r \sqsubseteq p$ and $r \sqsubseteq q$. As above, the conditions of Def. 4.4 are clear if $p = u_P$ or $q = u_Q$. We verify the remaining.

- If $p \wedge q \neq u_P \wedge u_Q$, then w.l.o.g. $p \neq u_P$. By $r \sqsubseteq p$, we also have $r \neq u_R$.

- Let $p \wedge q \xrightarrow{\alpha} S'$; w.l.o.g., this is due to $p \xrightarrow{\alpha}_P P'$ and $S' = \{p' \wedge q' \mid p' \in P',\ q \overset{\hat{\alpha}}{=\!\!\Rightarrow}_Q q',\ p' \wedge q' \text{ defined}\}$. Because of $r \sqsubseteq p$, we have $r \overset{\hat{\alpha}}{\Longrightarrow}_R R'$ (and $r \xrightarrow{\alpha}\overset{\varepsilon}{\Longrightarrow} R'$ in case of $\alpha \in I$) so that $\forall r' {\in} R'\, \exists p' {\in} P'.\ r' \sqsubseteq p'$. Consider some arbitrary $r' \in R'$ and the respective $p' \in P'$. Then, $r \overset{\hat{\alpha}}{=\!\!\Rightarrow}_R r'$ by syntactic consistency and, due to $r \sqsubseteq q$ and Prop. 4.5, there exists some q' with $q \overset{\hat{\alpha}}{=\!\!\Rightarrow}_Q q'$ and $r' \sqsubseteq q'$. Thus, $p' \wedge q' \in S'$ and $(r', p' \wedge q') \in \mathcal{R}$.

- Let $r \overset{\alpha}{\dashrightarrow}_R r'$, and consider $p \overset{\hat{\alpha}}{=\!\!\Rightarrow}_P p'$ and $q \overset{\hat{\alpha}}{=\!\!\Rightarrow}_Q q'$ satisfying $r' \sqsubseteq p'$ and $r' \sqsubseteq q'$. Thus, $p \wedge q \overset{\alpha}{\dashrightarrow} p' \wedge q'$ by (May) or $p \wedge q = p' \wedge q'$ and either way $(r', p' \wedge q') \in \mathcal{R}$.

3. By definition, $R \sqsubseteq P \wedge Q$ means $\forall r_0 \in R_0\ \exists p_0 \wedge q_0 \in P_0 \wedge Q_0.\ r_0 \sqsubseteq p_0 \wedge q_0$. By 2. this is equivalent to $\forall r_0 \in R_0\ \exists p_0 \wedge q_0 \in P_0 \wedge Q_0.\ (r_0 \sqsubseteq p_0$ and $r_0 \sqsubseteq q_0)$. The latter is equivalent to $\forall r_0 \in R_0\ \exists p_0 \in P_0.\ r_0 \sqsubseteq p_0$ and $\forall r_0 \in R_0\ \exists q_0 \in Q_0.\ r_0 \sqsubseteq q_0$; recall for the reverse implication that by 1. each required $p_0 \wedge q_0$ is present in the conjunction, since there is an r_0 refining both p_0 and q_0. By definition, the last two formulae mean $R \sqsubseteq P$ and $R \sqsubseteq Q$ and we are done. □

Corollary 4.38. *MIA refinement is compositional w.r.t. conjunction.*

Clearly, conjunction is commutative. Furthermore, any conjunction operator that satisfies the statement of Thm. 4.37 for some preorder $\sqsubseteq$ is associative.

Lemma 4.39. *Let P, Q, R and S be MIAs.*

1. *$P \wedge (Q \wedge R)$ is defined iff $(P \wedge Q) \wedge R$ is defined.*

2. *If $P \wedge (Q \wedge R)$ is defined, then $S \sqsubseteq P \wedge (Q \wedge R)$ iff $S \sqsubseteq (P \wedge Q) \wedge R$.*

Proof. 1. Thm. 4.37.1–3 imply that $P \wedge (Q \wedge R)$ is defined iff $\exists S.\ S \sqsubseteq P$ and $S \sqsubseteq Q \wedge R$ iff $\exists S.\ S \sqsubseteq P$ and $S \sqsubseteq Q$ and $S \sqsubseteq R$ iff $\exists S.\ S \sqsubseteq P \wedge Q$ and $S \sqsubseteq R$ iff $(P \wedge Q) \wedge R$ is defined. Claim 2 follows directly from multiple applications of Thm. 4.37.3. □

As a consequence of Lem. 4.39 we obtain strong associativity of conjunction.

Theorem 4.40 (Associativity of Conjunction)**.** *Conjunction is associative in the sense that, if one of $P \wedge (Q \wedge R)$ and $(P \wedge Q) \wedge R$ is defined, then both are defined and $P \wedge (Q \wedge R) \sqsupseteq\sqsubseteq (P \wedge Q) \wedge R$.*

4.5 Temporal Operators – ACTL

As reasoned for at the end of Section 4.2, we will require initial τ-closure for MIAs in this section. Like MIAs, our formulae have the same fixed alphabets I and O for now. As we have shown in [21], our operators preserve this property. We will work with single sets of parameters instead of explicitly partitioning them into inputs and outputs, since, with fixed I and O, the partitioning is set. When allowing for alphabet extension in Sect. 4.6 we will be more specific with this.

We define satisfaction so that universal MIAs do not satisfy any formulae. In fact, the formula $\mathtt{tt}$ can be understood as 'is not a universal MIA'. An intuitive reason for this is that $\mathfrak{U}$ stands for a communication mismatch and one will hardly want to specify such a deadly error. Another reason is that $\langle o \rangle P$ should specify that a system can perform an o-must-transition and behave like P afterwards. For a universal MIA P, this allows to produce an error after o, and such a system should be regarded as an error itself from the perspective of pruning a parallel composition. Hence, the system does not have an o-must-transition in contrast to the idea of $\langle o \rangle P$.

A universal MIA not satisfying any formula fits our goal that P satisfies a formula φ if and only if P refines the MIA-representation of φ. The latter is never universal; since a universal MIA only refines universal ones, it does not need to satisfy any formula. Additionally, the universal state u is in a way an artefact: One can define MIA and MIA-refinement equivalently without u but with a different addition describing arbitrary implementability of specific inputs (cf. [41, 48]).

We now define our variant of ACTL.

Definition 4.41 (ACTL)**.** *Let $C \subseteq B \subseteq \mathcal{A}$, $B_\varepsilon \subseteq \mathcal{A} \cup \{\varepsilon\}$ and $P \not\sqsupseteq\sqsubseteq \mathfrak{U}$ be a MIA.* ACTL-formulae φ *are given by:*

$\varphi ::= \mathtt{tt} \mid \mathtt{ff} \mid [B]\varphi \mid \langle B_\varepsilon \rangle \varphi \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \varphi\, {}_B\mathsf{W}_C\, \varphi$.

The semantics of ACTL is defined inductively as follows:

$P \models \mathtt{tt}$ *always*

$P \models \mathtt{ff}$ *if* $P_0 = \emptyset$

$P \models [B]\varphi$ *if* $\forall p_0 \in P_0\ \forall a \in B\ \forall p'.$
$(p_0 \overset{a}{\dashrightarrow} p'$ *implies* $P[p'] \models \varphi)$

$P \models \langle B_\varepsilon \rangle \varphi$ *if* $\forall p_0 \in P_0\ \exists \alpha \in B_\varepsilon, P' \subseteq P.\ P[P'] \models \varphi$ *and*
either $\alpha \in I$ *and* $p_0 \xrightarrow{\alpha}\overset{\varepsilon}{\Longrightarrow} P'$
or $\alpha \in O \cup \{\varepsilon\}$ *and* $p_0 \overset{\alpha}{\Longrightarrow} P'$

$P \models \varphi \wedge \psi$ *if* $P \models \varphi$ *and* $P \models \psi$

$P \models \varphi \vee \psi$ *if* $\forall p_0 \in P_0\ P[p_0] \models \varphi$ *or* $P[p_0] \models \psi$

$P \models \varphi_B \mathsf{W}_C \psi$ *if* $\forall p_0 \in P_0\ \forall p_1, p_2, \ldots, p_n \in P\ \forall \alpha_1, \alpha_2, \ldots, \alpha_n \in B \cup \{\tau\}.$
$(p_0 \overset{\alpha_1}{\dashrightarrow} p_1 \overset{\alpha_2}{\dashrightarrow} p_2 \cdots \overset{\alpha_n}{\dashrightarrow} p_n$ *implies*
$((\exists 1 \leq i \leq n.\, \alpha_i \in C$ *and* $P[p_i] \models \psi)$ *or* $P[p_n] \models \varphi))$

We define, $\mathcal{L}_A(P) =_{df} \{\varphi \mid P \models \varphi, \varphi$ *is an ACTL-formula*$\}$ *as the set of ACTL-formulae that the MIA* P *satisfies. We write* $[a]$ *or* $\langle\alpha\rangle$ *if* a *or* α *is the only element in the respective set.*

Before discussing the details and choices of our semantics, we present a lemma to ease understanding. It is easy to show by structural induction. For the first item, induction is only needed for $\varphi \wedge \psi$. It implies the second and third item.

Lemma 4.42. *Let* P *be a MIA and* φ *be a formula.*

1. $P \models \varphi$ *if and only if* $\forall p_0 \in P_0.\, P[p_0] \models \varphi$.

2. $P \models \varphi \wedge \psi$ *if and only if* $\forall p_0 \in P_0.\, P[p_0] \models \varphi$ *and* $\forall p_0 \in P_0.\, P[p_0] \models \psi$.

3. $P \models \varphi$ *implies* $P[P_0'] \models \varphi$ *for all* $P_0' \subseteq P_0$.

The first two operators tt and ff are standard and rather intuitive. For $[B]$, using $\overset{a}{\Longrightarrow}$ instead of $\overset{a}{\dashrightarrow}$ would look more natural (and more restrictive), since all ways of performing a should be covered. But, in fact, this is an equivalent definition: Clearly the version with $\overset{a}{\Longrightarrow}$ implies our condition. Vice versa, assume $p_0 \overset{a}{\Longrightarrow} p'$ due to $p_0 \overset{\varepsilon}{\Longrightarrow} p_0' \overset{a}{\dashrightarrow} p \overset{\varepsilon}{\Longrightarrow} p'$; since $p_0' \in P_0$ (due to initial τ-closure), we require $P[p] \models \varphi$ above and p' is initial in $P[p]$. By Lem. 4.42.1, $P[p'] \models \varphi$ holds also in our setting.

For $\langle B_\varepsilon \rangle$, we have to define the operator by using weak transitions since these properties are supposed to be retained during refinement: Consider MIAs consisting of $q_0 \xrightarrow{o} q'$ and $p_0 \xrightarrow{\tau} p \xrightarrow{o} p'$ respectively. They are MIA-equivalent, but only Q would satisfy $\langle o \rangle$tt if leading τ-transitions were not permitted for outputs in the definition.

A similar argument can be made for trailing τ-transitions for both inputs and outputs: Consider MIAs P and Q from Fig. 4.12 and $\varphi = \langle a \rangle \langle i \rangle$tt. We have $P \sqsupseteq\sqsubseteq Q$ and both satisfy φ. But if not permitting trailing τs in the definition of $\langle B_\varepsilon \rangle$, only Q would satisfy φ.

If one only defines $[b]\varphi$ and $\langle b \rangle \varphi$ for single actions b, one cannot derive $[B]\varphi$ or $\langle B_\varepsilon \rangle \varphi$ in general. The first is impossible for infinite B and the second already for finite B. We will go into more detail later in Prop. 4.57. A formula like $\langle \mathcal{A} \rangle$tt is useful to describe deadlock freedom; see our main example in Sect. 4.5.2.

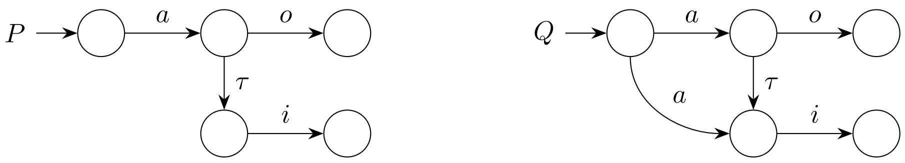

Figure 4.12: MIAs used in the arguments for trailing τ-transitions in Def. 4.41 and the power of the $\langle B_\varepsilon\rangle$-operator.

We allow for $\langle\varepsilon\rangle\varphi$, which is standard, but not for $[\varepsilon]\varphi$, since it would be redundant. One would define $P \models [\varepsilon]\varphi$ as $\forall p_0 \in P_0 .\, p_0 \overset{\varepsilon}{\Longrightarrow} p$ implies $P[p] \models \varphi$. Since $p \in P_0$, this is already required for $P \models \varphi$ by Lem. 4.42.1, hence $[\varepsilon]\varphi$ would be equivalent to φ.

In contrast to $\wedge$, we cannot define $P \models \varphi \vee \psi$ by $P \models \varphi$ or $P \models \psi$ without changing its meaning. For an example consider again the MIAs in Fig. 4.4. By our definition $P \vee Q \models \langle o\rangle\mathtt{tt} \vee \langle o'\rangle\mathtt{tt}$, but neither $P \vee Q \models \langle o\rangle\mathtt{tt}$ nor $P \vee Q \models \langle o'\rangle\mathtt{tt}$. Our definition is necessary for our translation of formulae into MIAs; we present a formal argument after Cor. 4.52.

To explain the intuition behind the Unless-operator we say that a *state p satisfies* φ if $P[p] \models \varphi$. Intuitively, $\varphi_B\mathsf{W}_C\psi$ means that φ must hold in all states reachable by actions $b \in B$ unless an action $c \in C$ is performed and ψ holds afterwards. In particular, this means that $P \models \psi$ does not imply $P \models \varphi_B\mathsf{W}_C\psi$, as it would usually do, since the Unless-operator requires a c-transition to lead to a state satisfying ψ. A variant satisfying this implication can be expressed by $\psi \vee (\varphi_{\mathcal{A}}\mathsf{W}_{\mathcal{A}}\psi)$ (cf. Def. 4.43). Note that $C \subseteq B$ is not a restriction: Taking any transition labelled with some $a \notin B$ can end the requirement for φ without having to satisfy ψ afterwards. For a better understanding, we refer to the brief example after the following definition.

We can derive some further typical operators, like Always or an unparametrized Unless.

Definition 4.43 (ACTL derived operators)**.** *Let $B \subseteq \mathcal{A}$ and $P \not\sqsupseteq\!\sqsubseteq \mathfrak{U}$ be a MIA.*

$P \models en(B)$ *if* $p \models \langle B\rangle\mathtt{tt}$
i.e. $\forall p_0 \in P_0\, (\exists i \in B \cap I .\, p_0 \xrightarrow{i}\overset{\varepsilon}{\Longrightarrow})$ *or* $(\exists o \in B \cap O .\, p_0 \overset{o}{\Longrightarrow})$

$P \models dis(B)$ *if* $P \models [B]\mathtt{ff}$
i.e. $\forall p_0 \in P_0\ \forall a \in B .\, p_0 \overset{a}{\dashrightarrow}\!\!\!\!/\ $

$P \models \Box_B\varphi$ *if* $P \models \varphi_B\mathsf{W}_\emptyset\mathtt{ff}$
i.e. $\forall p_1, p_2, \ldots, p_n\ \forall \alpha_1, \alpha_2, \ldots, \alpha_n \in B \cup \{\tau\}$.
$(p_0 \overset{\alpha_1}{\dashrightarrow} p_1 \overset{\alpha_2}{\dashrightarrow} p_2 \cdots \overset{\alpha_n}{\dashrightarrow} p_n$ *implies* $P[p_n] \models \varphi)$

$$\begin{array}{ll} P \models \Box\varphi & \textit{if } P \models \Box_{\mathcal{A}}\varphi \\ & \textit{i.e. } \forall p_1, p_2, \dots, p_n.\ (p_0 \overset{\alpha_1}{\dashrightarrow} p_1 \overset{\alpha_2}{\dashrightarrow} p_2 \cdots \overset{\alpha_n}{\dashrightarrow} p_n)\ \textit{implies } P[p_n] \models \varphi) \\ P \models \varphi \mathsf{W} \psi & \textit{if } P \models \psi \vee (\varphi_{\mathcal{A}} \mathsf{W}_{\mathcal{A}} \psi) \\ & \textit{i.e. } \forall p_1, p_2, \dots, p_n\ \forall \alpha_1, \alpha_2, \dots, \alpha_n \in \mathcal{A}_\tau. \\ & \quad (p_0 \overset{\alpha_1}{\dashrightarrow} p_1 \overset{\alpha_2}{\dashrightarrow} p_2 \cdots \overset{\alpha_n}{\dashrightarrow} p_n\ \textit{implies} \\ & \quad (P[p_n] \models \varphi\ \textit{or}\ \exists 0 \leq i \leq n.\, P[p_i] \models \psi)) \end{array}$$

Note that $en(B)$ is defined on sets $B \subseteq \mathcal{A}$ and, thus, does not allow for $en(\varepsilon)$. Of course, the formula $\langle B_\varepsilon \rangle \mathtt{tt}$ with $\varepsilon \in B_\varepsilon$ is defined, but it is equivalent to $\mathtt{tt}$ since $p_0 \overset{\varepsilon}{\Longrightarrow} p_0$ is always true.

As an example we can easily formalize that a system is ready to accept a signal i as an input by $en(i)$. A more interesting specification is that the system should be allowed to operate, i.e. accept inputs and send outputs, while remaining capable of receiving i until doing so. This can be formalized by $en(i)_{\mathcal{A}} \mathsf{W}_{\{i\}} \mathtt{tt}$. It requires i to remain enabled, and the only way to escape this requirement is actually taking an i-transition, i.e. receiving the signal i. This highly desirable property could not be specified without the parameter C in our definition of unless.

4.5.1 Embedding in MIA

For a truly heterogeneous specification method, we now define how to apply the logical connectives to arbitrary non-universal MIAs. In particular, this will also give us a translation from a formula to a MIA. We will show compatibility of satisfaction with refinement, in the sense that $P \models \varphi$ iff $P \sqsubseteq \overline{\varphi}$, where $\overline{\varphi}$ is a MIA-representation of φ. Conjunction and disjunction are already defined on MIAs, so we start by defining $\mathtt{tt}$, $\mathtt{ff}$ and the Next-operators.

Definition 4.44 (Temporal operators on MIAs: $\mathtt{tt}$, $\mathtt{ff}$, $\langle B_\varepsilon \rangle$ and $[B]$)**.** *Let $Q \not\sqsupseteq\!\sqsubseteq \mathfrak{U}$ be a MIA. $\mathtt{ff}$ is represented by the empty MIA $\mathfrak{F}$. The representation of $\mathtt{tt}$ is depicted in Fig. 4.13, as are constructions for $\langle B_\varepsilon \rangle Q$ (for $B = \{a, b, \varepsilon\}$) and $[B]Q$.*

A state labelled by Q_0 means that Q is reached here: An incoming may-transition represents a bundle of may-transitions, one for each initial state of Q (and each $a \in B$), and an incoming must-transition is one disjunctive must-transition to Q_0. The depicted state u_Q is the universal state of Q and of $\langle B_\varepsilon \rangle Q$ or $[B]Q$ respectively.

The MIA for $\langle B_\varepsilon \rangle$ has states $\{\mathtt{tt}_0\} \cup \{x_{\alpha 0} \mid \alpha \in B_\varepsilon\} \cup Q$. The second subset is the set of its initial states.

$\mathtt{tt}$ is the most permissive non-universal MIA; it needs a universal state to be completely arbitrary after an input. Note that $\mathtt{tt}$ is embedded in the

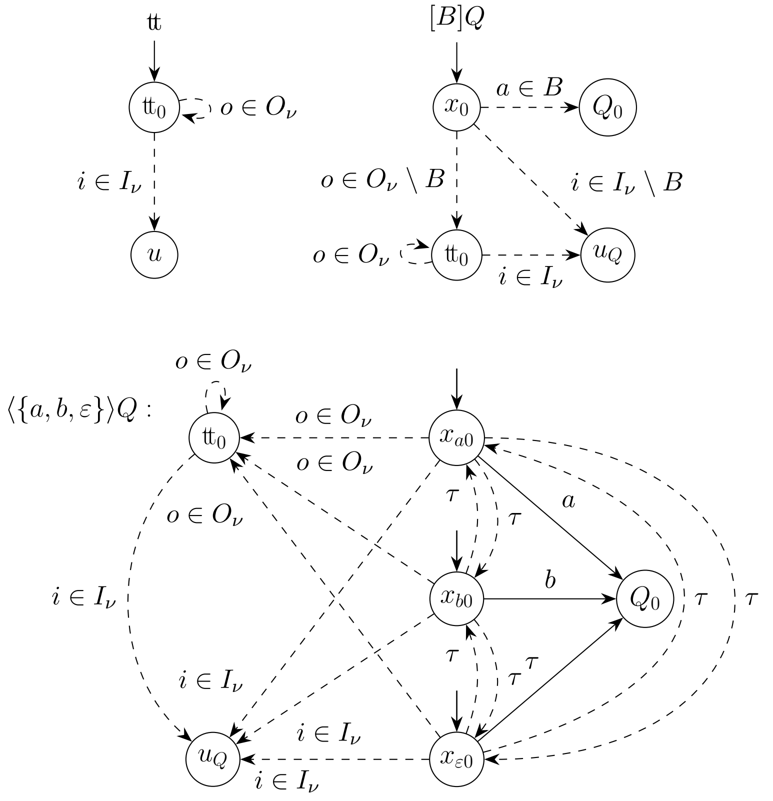

Figure 4.13: MIA constructions for $\mathtt{tt}$, $[B]$ and $\langle\{a, b, \varepsilon\}\rangle$. The must-transitions are disjunctive. I_ν and O_ν stand for $I \cup \{\nu_I\}$ and $O \cup \{\nu_O\}$ respectively.

$[B]Q$ and $\langle B_\varepsilon\rangle Q$. The MIA $[B]Q$ specifies that Q must be refined after any $a \in B$. For other actions, it behaves like $\mathtt{tt}$. Similarly, the initial states of $\langle B_\varepsilon\rangle Q$ behave like $\mathtt{tt}$, but additionally require together that at least one of the $\alpha \in B_\varepsilon$ be implemented.

Before we define the construction for the Unless-operator, we first present one for the Always-operator. This is technically not necessary since Always is a derived operator, but we will use it as part of the definition of the Unless-operator and we believe it eases understanding to see this construction on its own. The basic idea of the construction is to build up conjunctions of states while adding q_0 after each transition. It is already used in [50], albeit in a simpler setting with a very specific treatment of τs and absence of action-sets as indices.

We define $\Box_B Q$ in two steps, similar to our construction of the conjunction: first we define a 'Proto-Always', then we remove logical inconsistencies.

The idea behind this construction is that a set of states $\{q_1, \dots, q_n\}$ will behave like the conjunction of the states, except that after each transition labelled with some $a \in B$ another copy of some $q_0 \in Q_0$ is added. This ensures that Q is refined by every state of the new MIA. Apart from this, the transitions are defined similarly to the transitions of the conjunctive product. As mentioned before, the universal state is neutral w.r.t. conjunction, cf. (UMay1), and thus we remove it from sets $\{q_0, \dots, q_n\}$. In conjunctions $\bigwedge\{q'_1, \dots, q'_n\}$, as introduced after Thm. 4.37, it is neutral by definition of the conjunction and thus we can remove it as well if we want, except if it is the only conjunct.

Taking a transition labelled by some $a \notin B$ ends adding q_0, and the target is a conjunction $\bigwedge\{q'_1, \dots, q'_n\}$, which ensures the remaining requirements for the states that have been passed. This conjunction may be undefined, i.e. non-existent, since its elements can be inconsistent. In this case, no may-transition is added in (MayK) below. Similarly, undefined $\bigwedge\{q'_1, \dots, q'_n\}$ are not added to the target set in (IMustK) or (OMustK). In an extreme case, this may give $\emptyset$ as target set. Since the removal of inconsistencies propagates backward over must-transitions, we need to retain this information. Therefore, we allow must-transitions to lead to $\emptyset$. Otherwise, the Proto-Always is a MIA, and we call it a pseudo-MIA. Note that syntactic consistency vacuously holds for transitions to $\emptyset$.

Observe that for $\{q'_1, \dots, q'_n, q_0\}$ and $\bigwedge\{q'_1, \dots, q'_n\}$ in the targets, some of the elements of the sets may be listed more than once; hence the size of a set might be shrunk after a transition. For states $\bigwedge\{q'_1, \dots, q'_n\}$, we import the respective iterated conjunction MIA.

Definition 4.45 (Temporal operators on MIAs: Proto-Always)**.** *Let $B \subseteq \mathcal{A}$ and Q be a non-universal MIA. The* Proto-Always *is the pseudo-MIA $\overline{\Box_B Q} = (\overline{Q} \mathbin{\dot\cup} K \mathbin{\dot\cup} \{\emptyset\}, I, O, \longrightarrow_\Box, \dashrightarrow_\Box, \{S \mid \exists q_0 \in Q_0.\, \{q_0\} \overset{\varepsilon}{\Longrightarrow}_\Box S\}, \{u_Q\})$, where $\overline{Q} = \mathfrak{P}_{\mathit{fin}}(Q) \setminus \{\emptyset\}$ is the set of non-empty finite subsets of Q and $K = \{\bigwedge\{q_1, \dots, q_n\} \mid \{q_1, \dots, q_n\} \subseteq Q, \bigwedge\{q_1, \dots, q_n\}$ is defined$\}$ the set of all (finite) conjunctions of states of Q. The sets $\longrightarrow_\Box$ and $\dashrightarrow_\Box$ contain all transitions of the conjunctions and the ones given by the following rules:*

(IMust) $\{q_1, \dots, q_n\} \overset{i}{\longrightarrow}_\Box$

$\{\{q'_1, \dots, q'_n, q_0\} \setminus \{u\} \mid q'_l \in Q'_l,\ q_0 \in Q_0, \forall k \in \{1, \dots, n\} \setminus \{l\}.\, q_k \overset{i}{\Longrightarrow} q'_k\}$

if $i \in B$, $l \in \{1, \dots, n\}$ *and* $q_l \overset{i}{\longrightarrow} Q'_l$

(IMustK) $\{q_1, \dots, q_n\} \overset{i}{\longrightarrow}_\Box$

$\{\bigwedge\{q'_1, \dots, q'_n\} \mid q'_l \in Q'_l,\ \forall k \in \{1, \dots, n\} \setminus \{l\}.\, q_k \overset{i}{\Longrightarrow} q'_k\}$

if $i \notin B$*,* $l \in \{1, \dots, n\}$ *and* $q_l \xrightarrow{i} Q'_l$

(OMust) $\{q_1, \dots, q_n\} \xrightarrow{\omega}_{\Box}$

$\{\{q'_1, \dots, q'_n, q_0\} \mid q'_l \in Q'_l, q_0 \in Q_0, \forall k \in \{1, \dots, n\} \setminus \{l\}.\, q_k \overset{\hat{\omega}}{\Longrightarrow} q'_k\}$

if $\omega \in B \cup \{\tau\}$*,* $l \in \{1, \dots, n\}$ *and* $q_l \xrightarrow{\omega} Q'_l$

(OMustK) $\{q_1, \dots, q_n\} \xrightarrow{o}_{\Box}$

$\{\bigwedge\{q'_1, \dots, q'_n\} \mid q'_l \in Q'_l, \forall k \in \{1, \dots, n\} \setminus \{l\}.\, q_k \overset{o}{\Longrightarrow} q'_k\}$

if $o \notin B$*,* $l \in \{1, \dots, n\}$ *and* $q_l \xrightarrow{o} Q'_l$

(May) $\{q_1, \dots, q_n\} \overset{\alpha}{\dashrightarrow}_{\Box} \{q'_1, \dots, q'_n, q_0\} \setminus \{u\}$

if $\alpha \in B \cup \{\tau\}$*,* $q_0 \in Q_0$*,* $(\forall k \in \{1, \dots, n\}.\, q_k \overset{\hat{\alpha}}{\Longrightarrow} q'_k)$*,*

and $\exists k \in \{1, \dots, n\}.\, q_k \overset{\alpha}{\Longrightarrow}$

(MayK) $\{q_1, \dots, q_n\} \overset{a}{\dashrightarrow}_{\Box} \bigwedge\{q'_1, \dots, q'_n\}$

if $a \notin B$ *and* $\forall k \in \{1, \dots, n\}.\, q_k \overset{a}{\Longrightarrow} q'_k$

(MayN) $\{q_1, \dots, q_n\} \overset{\nu}{\dashrightarrow}_{\Box} \bigwedge\{q'_1, \dots, q'_n\}$

if $\nu \notin \{\nu_I, \nu_O\}$ *and* $\forall k \in \{1, \dots, n\}.\, q_k \overset{\nu}{\Longrightarrow} q'_k$

Indeed, $\overline{\Box_B Q}$ is a pseudo-MIA. Note that, for $\Box Q$, the rules ending in K are never used and can be omitted, as well as the set K.

We iteratively determine the set of inconsistent states of the Proto-Always. We consider states in $\overline{Q}$ to be inconsistent if there is one element requiring an action and another forbidding it. As in the conjunction construction in [12], the latter means $\overset{o}{\not\Longrightarrow}$ for outputs and $\overset{i}{\not\dashrightarrow}$ for inputs. It suffices to check for states, where one element has an a-must-transition, but the state has no strong a-may-transition: by construction of the Proto-Always (cf. (May) and (MayK)), $\overline{q} \overset{o}{\Longrightarrow}$ if and only if $\overline{q} \overset{o}{\dashrightarrow}$. Having identified such immediate inconsistencies, we consider states that have a must-transition to a set of inconsistent states (in particular, the empty set) to be inconsistent as well. After removing all these inconsistent states, we have the desired MIA $\Box_B Q$.

Definition 4.46 (Temporal operators on MIAs: Always)**.** *Given a Proto-Always* $\overline{\Box_B Q}$*, the set* $F \subseteq \overline{\Box_B Q}$ *of (logically)* inconsistent states *is defined as the least set satisfying the following conditions for all* $\overline{q} \in \overline{Q}$*:*

(1) $(\exists q_i \in \overline{q}.\, q_i \xrightarrow{a}) \wedge \overline{q} \overset{a}{\not\dashrightarrow}$ *implies* $\overline{q} \in F$

(2) $\overline{q} \xrightarrow{\alpha} \overline{Q}' \subseteq F$ *implies* $\overline{q} \in F$ *(especially for* $\overline{Q}' = \emptyset$*)*

The MIA $\Box_B Q$ *is obtained by deleting all states in* F *from* $\overline{\Box_B Q}$*. This also removes any may- or must-transitions exiting and any may-transitions entering a deleted state; in addition, deleted states are removed from targets of disjunctive must-transitions.*

For later use we present the following lemma. Intuitively, it states that a state $\{q_1, \dots, q_n\}$ is finer or equal to the conjunction $q_1 \wedge q_2 \wedge \ldots \wedge, q_n$.

Lemma 4.47. *Let P and $Q \not\sqsupseteq\!\sqsubseteq \mathfrak{U}$ be MIAs and $\overline{q} \in \Box_B Q$. We write $q \in \bigwedge\{q_1, \dots, q_n\}$ if $q \in \{q_1, \dots, q_n\}$. If $p \sqsubseteq \overline{q}$ then for all $q_i \in \overline{q}$ we have $p \sqsubseteq q_i$. The reverse implication does not hold.*

Proof. We partition the state space $\Box_B Q$ into $\overline{Q} \dot\cup K$ as in Def. 4.45. To prove the implication, we show that $\mathcal{R} = \{(p, q_l) \mid p \in P, \exists \overline{q} \in \overline{Q}.\, q_l \in \overline{q} \text{ and } p \sqsubseteq \overline{q}\} \cup \sqsubseteq$ is a MIA-refinement relation. The correctness of $\sqsubseteq$ is obvious. We take a pair (p, q_l) due to $\overline{q} = \{q_1, \dots, q_n\}$ from the first subset and check the items of Def. 4.4. We have $q_l \neq u_Q$ by construction and the assumption on Q.

(i) Since $q_l \neq u_Q$, we have $\overline{q} \neq \{u_Q\}$. Thus, $p \neq u_P$, since $p \sqsubseteq \overline{q}$.

(ii) $q_l \xrightarrow{i} Q'_l$. If not $\forall q_k \in \overline{q}.\, q_k \overset{i}{\Longrightarrow}$, then $\overline{q} \overset{i}{\not\dashrightarrow}$ (cf. (May) and (MayK)) and $\overline{q}$ is logically inconsistent implying $\overline{q} \notin \Box_B Q$. So we assume otherwise and get $\overline{q} \xrightarrow{i}$.

If $i \notin B$, we get $\overline{q} \xrightarrow{i} \overline{Q}' = \{\bigwedge\{q'_1, \dots, q'_l, \dots, q'_n\} \mid q'_l \in Q'_l, \forall k \in \{1, \dots, n\} \backslash \{l\}.\, q_k \overset{i}{\Longrightarrow} q'_k\}$ by (IMustK). Since $p \sqsubseteq \overline{q}$, we have $p \xrightarrow{i}\overset{\varepsilon}{\Longrightarrow} P'$ for some P' with $\forall p' \in P'\ \exists \overline{q}' \in \overline{Q}'.\, p' \sqsubseteq \overline{q}'$. For the respective $q'_l \in \overline{q}'$, we have $p' \sqsubseteq q'_l$ by Thm. 4.37.2.

If $i \in B$, we get $\overline{q} \xrightarrow{i} \overline{Q}' = \{\{q'_1, \dots, q'_l, \dots, q'_n, q_0\} \setminus \{u\} \mid q'_l \in Q'_l, \forall k \in \{1, \dots, n\} \setminus \{l\}.\, p_k \overset{i}{\Longrightarrow} p'_k\}$ by (IMust). Since $p \sqsubseteq \overline{q}$, we have $p \xrightarrow{i}\overset{\varepsilon}{\Longrightarrow} P'$ for some P' with $\forall p' \in P'\ \exists \overline{q}' \in \overline{Q}'.\, p' \sqsubseteq \overline{q}'$. Since $p' \sqsubseteq \overline{q}'$, each pair (p', q'_k) with $q'_k \in \overline{q}'$ is in the first subset. This includes (p', q'_l) for some $q'_l \in Q'_l$ in particular; observe that q'_l is indeed in $\overline{q}'$ since $q'_l \neq u_Q$ is not possible for a target of $q_l \xrightarrow{i} Q'_l$.

(iii) $q_l \xrightarrow{\omega} Q'$ is analogous.

(iv) $p \overset{\alpha}{\dashrightarrow} p'$ implies $\exists \overline{q}'.\, \overline{q} \overset{\alpha}{\Longrightarrow} \overline{q}'$ and $p' \sqsubseteq \overline{q}'$. This transition exists because of (May) or (MayK). If we have $\alpha = \tau$ and $q_l \in \overline{q}'$ then (p', q_l) is in the first subset of $\mathcal{R}$, no matter if $\overline{q} = \overline{q}'$ or $\overline{q} \overset{\tau}{\Longrightarrow} \overline{q}'$. Otherwise, we have $q_l \overset{\alpha}{\Longrightarrow} q'_l$ and $q'_l \in \overline{q}'$ or $q'_l = u_Q$ and are done due to the first subset, Thm. 4.37.2 or Def. 4.4.

To show that the reverse does not hold in general consider a MIA $Q = (\{q_0, q_1, u\}, I, O, \dashrightarrow, \longrightarrow, \{q_0\}, u)$ with $o \in O$ and the transitions $q_0 \xrightarrow{o} q_1$ (including the underlying may-transition) and $q_1 \overset{o}{\dashrightarrow} q_1$. Obviously, $q_0 \sqsubseteq q_0$ and thus, q_0 refines all q_l in $\{q_0\}$, the initial state of $\Box_{\mathcal{A}} Q$. However, $\{q_0\} \xrightarrow{o} \{q_1, q_0\} \xrightarrow{o} \{q_1, q_0\}$; the second transition cannot be matched in Q, thus $q_0 \not\sqsubseteq \{q_0\}$. □

The Unless construction essentially creates all states and transitions the Always construction does. In addition, there is the possibility of taking a C-transition while finally adding a conjunct from R_0 instead of Q_0. For must-transitions with $a \in C$ this means also adding some r_0 to the target state. Like the Proto-Always, the Proto-Unless is a pseudo MIA, i.e. it can have must-transitions targeting the empty state.

Definition 4.48 (Temporal operators on MIAs: Proto-Unless)**.** *Let $C \subseteq B \subseteq \mathcal{A}$ and Q and R be MIAs. The* Proto-Unless *is the pseudo-MIA $\overline{Q_B \mathsf{W}_C R} = (\overline{Q} \uplus K \uplus K' \uplus \{\emptyset\}, I, O, \longrightarrow_W, \dashrightarrow_W, \{S \mid \exists q_0 \in Q_0. \{q_0\} \overset{\varepsilon}{\Longrightarrow}_W S\}, \{u_Q\})$, where $\overline{Q} = \mathfrak{P}_{fin}(Q) \backslash \{\emptyset\}$ is the powerset of Q, $K = \{\bigwedge\{q_1, \dots, q_n\} \mid \{q_1, \dots, q_n\} \subseteq Q,\ \bigwedge\{q_1, \dots, q_n\}$ is defined $\}$ the set of all possible consistent conjunctions of states of Q and $K' = \{\bigwedge\{q_1, \dots, q_n, r\} \mid \{q_1, \dots, q_n\} \subseteq Q,\ r \in R,\ \bigwedge\{q_1, \dots, q_n, r\}$ is defined$\}$ the set of all possible consistent conjunctions of states of Q and one of R. The sets $\longrightarrow_W$ and $\dashrightarrow_W$ contain all transitions of the conjunctions, all transitions of the Proto-Always* except for must-transitions with labels in C, *and the ones given by the following rules:*

(IMustW) $\{q_1, \dots, q_n\} \xrightarrow{i}_W$
$\{\{q'_1, \dots, q'_n, q_0\} \setminus \{u\} \mid q'_l \in Q'_l, q_0 \in Q_0, \forall k \in \{1, \dots, n\} \setminus \{l\}. q_k \overset{i}{\Longrightarrow} q'_k\}$
$\cup \{ \bigwedge\{q'_1, \dots, q'_n, r_0\} \mid q'_l \in Q'_l, r_0 \in R_0, \forall k \in \{1, \dots, n\} \setminus \{l\}. q_k \overset{i}{\Longrightarrow} q'_k\}$
if $i \in C$, $l \in \{1, \dots, n\}$ *and* $q_l \xrightarrow{i} Q'_l$
(OMustW) $\{q_1, \dots, q_l, \dots, q_n\} \xrightarrow{o}_W$
$\{\{q'_1, \dots, q'_n, q_0\} \mid q'_l \in Q'_l, q_0 \in Q_0, \forall k \in \{1, \dots, n\} \setminus \{l\}. q_k \overset{o}{\Longrightarrow} q'_k\}$
$\cup \{\bigwedge\{q'_1, \dots, q'_n, r_0\} \mid q'_l \in Q'_l, r_0 \in R_0, \forall k \in \{1, \dots, n\} \setminus \{l\}. q_k \overset{o}{\Longrightarrow} q'_k\}$
if $o \in C$, $l \in \{1, \dots, n\}$ *and* $q_l \xrightarrow{o} Q'_l$
(MayW) $\{q_1, \dots, q_n\} \overset{a}{\dashrightarrow}_W \bigwedge\{q'_1, \dots, q'_n, r_0\}$
if $a \in C$, $r_0 \in R_0$ *and* $\forall k \in \{1, \dots, n\}. q_k \overset{a}{\Longrightarrow} q'_k$

Definition 4.49 (Temporal operators on MIAs: Unless)**.** *Given a Proto-Unless $\overline{Q_B \mathsf{W}_C R}$, we arrive at the Unless MIA $Q_B \mathsf{W}_C R$ by applying the same pruning as for Always in Def. 4.46.*

Similarly to Always, we present a lemma stating that the states of the Unless-MIA behave like a conjunction of the component states or finer.

Lemma 4.50. *Let P and $Q \sqsupset\!\!\!\not\sqsubseteq \mathfrak{U} \sqsupset\!\!\!\not\sqsubseteq R$ be MIAs, $C \subseteq B \subseteq \mathcal{A}$ and $\overline{q} \in Q_B \mathsf{W}_C R$. We write $q \in \bigwedge\{q_1, \dots, q_n\}$ if $q \in \{q_1, \dots, q_n\}$. If $p \sqsubseteq \overline{q}$ then, for all $q_i \in \overline{q}$, we have $p \sqsubseteq q_i$. (The reverse implication does not hold.)*

Proof. We partition the state space $P_B \mathsf{W}_C Q$ into $\overline{Q} \uplus K \uplus K'$ as in Def. 4.48. To prove the implication, we show that $\mathcal{R} = \{(p, q_l) \mid p \in P, \exists \overline{q} \in \overline{Q}. q_l \in$

$\overline{q}$ and $p \sqsubseteq \overline{q}\} \cup \sqsubseteq$ is a MIA-refinement relation. The correctness of $\sqsubseteq$ is obvious. We take a pair (p, q_l) due to $\overline{q} = \{q_1, \ldots, q_n\}$ from the first subset and check the items of Def. 4.4. We have $q_l \neq u_Q$ by construction and the assumption on Q.

(i) Since $q_l \neq u_Q$, we have $\overline{q} \neq \{u_Q\}$. Thus, $p \neq u_P$, since $p \sqsubseteq \overline{q}$.

(ii) $q_l \xrightarrow{i} Q'_l$. If not $\forall q_k \in \overline{q}.\, q_k \overset{i}{\Longrightarrow}$, then $\overline{q} \overset{i}{\dashrightarrow}\!\!\!\!/\;$ (cf. (May), (MayK) and (MayW)) and $\overline{q}$ is logically inconsistent implying $\overline{q} \notin P_B \mathsf{W}_C Q$. So we assume otherwise and get $\overline{q} \xrightarrow{i}$.

The case $i \notin C$, has been shown as part of the proof of Lem. 4.47. It remains to examine the case $i \in C$. We get $\overline{q} \xrightarrow{i} \overline{Q}'$ with $\overline{Q}' = \{\{q'_1, \ldots, q'_n, q_0\} \setminus \{u_Q\} \mid q'_l \in Q'_l, q_0 \in Q_0, \forall k \in \{1, \ldots, n\} \setminus \{l\}.\, q_k \overset{i}{\Longrightarrow} q'_k\} \cup \{\bigwedge\{q'_1, \ldots, q'_n, r_0\} \mid q'_l \in Q'_l, r_0 \in R_0, \forall k \in \{1, \ldots, n\} \setminus \{l\}.\, q_k \overset{i}{\Longrightarrow} q'_k\}$ by (IMustW). Since $p \sqsubseteq \overline{q}$, we have $p \xrightarrow{i} \overset{\varepsilon}{\Longrightarrow} P'$ for some P' with $\forall p' \in P'\ \exists \overline{q}' \in \overline{Q}'.\, p' \sqsubseteq \overline{q}'$. Since $p' \sqsubseteq \overline{q}'$, each pair (p', q'_k) with $q'_k \in \overline{q}'$ is in the first subset of $\mathcal{R}$ if $\overline{q}'$ is in the first subset of $\overline{Q}'$ and in the second subset of $\mathcal{R}$ if $\overline{q}'$ is in the second subset of $\overline{Q}'$. This includes (p', q'_l) for some $q'_l \in Q'_l$ in particular.

(iii) $q_l \xrightarrow{\omega} Q'$ is analogous.

(iv) $p \overset{\alpha}{\dashrightarrow} p'$ implies $\exists \overline{q}'.\, \overline{q} \overset{\alpha}{\Longrightarrow} \overline{q}'$ and $p' \sqsubseteq \overline{q}'$. If we have $\omega = \tau$ and $q_l \in \overline{q}'$ then (p', q_l) is in the first subset of $\mathcal{R}$, no matter if $\overline{q} = \overline{q}'$ or $\overline{q} \overset{\tau}{\Longrightarrow} \overline{q}'$. Otherwise, this transition exists because of (May), (MayK) or (MayW), and in either case, we have $q_l \overset{\alpha}{\Longrightarrow} q'_l$ and $q'_l \in \overline{q}'$ or $q'_l = u_Q$ and are done due to the first subset, Thm. 4.37.2 or Def. 4.4.

The reverse implication does not hold, already for $\Box Q = Q_{\mathcal{A}} \mathsf{W}_{\emptyset} \mathrm{ff}$ as shown in Lem. 4.47. □

Now we have shown how to translate each logical operator into an operator on MIAs. In particular, we can translate each formula φ into a MIA, which we now also denote by φ.

To achieve the result that satisfaction and refinement coincide, we show more generally that the logical operators applied to MIAs fit the definition of satisfaction. For conjunction, we know this already.

Theorem 4.51 (Temporal MIA operators vs. formulae)**.** *For non-universal MIAs P, Q and R and for sets $C \subseteq B \subseteq \mathcal{A}$ and $B_\varepsilon \subseteq \mathcal{A} \cup \{\varepsilon\}$, we have*

$$\begin{array}{ll}
P \sqsubseteq \mathtt{tt} & \textit{always}\\
P \sqsubseteq \mathtt{ff} & \textit{iff}\ \ P_0 = \emptyset\\
P \sqsubseteq [B]Q & \textit{iff}\ \ \forall p_0 \in P_0\ \forall a \in B\ \forall p'.\ (p_0 \overset{a}{\dashrightarrow} p' \textit{ implies } P[p'] \sqsubseteq Q)\\
P \sqsubseteq \langle B_\varepsilon \rangle Q & \textit{iff}\ \ \forall p_0 \in P_0\ \exists \alpha \in B_\varepsilon, P'.\, P[P'] \sqsubseteq Q \textit{ and}\\
 & \qquad\quad \textit{either } \alpha \in I \textit{ and } p_0 \xrightarrow{\alpha}\overset{\varepsilon}{\Longrightarrow} P'\\
 & \qquad\quad \textit{or } \alpha \in O \cup \{\varepsilon\} \textit{ and } p_0 \overset{\alpha}{\Longrightarrow} P'\\
P \sqsubseteq Q \wedge R & \textit{iff}\ \ P \sqsubseteq Q \textit{ and } P \sqsubseteq R\\
P \sqsubseteq Q \vee R & \textit{iff}\ \ \forall p_0 \in P_0\ P[p_0] \sqsubseteq Q \textit{ or } P[p_0] \sqsubseteq R\\
P \sqsubseteq \Box_B Q & \textit{iff}\ \ \forall p_0 \in P_0\ \forall p_1, p_2, \ldots, p_n \in P\ \forall \alpha_1, \alpha_2, \ldots, \alpha_n \in B \cup \{\tau\}.\\
 & \qquad (p_0 \overset{\alpha_1}{\dashrightarrow} p_1 \overset{\alpha_2}{\dashrightarrow} \cdots p_n \textit{ implies } P[p_n] \sqsubseteq Q)\\
P \sqsubseteq Q_B \mathsf{W}_C R & \textit{iff}\ \ \forall p_0 \in P_0\ \forall p_1, p_2, \ldots, p_n \in P\\
 & \forall \alpha_1, \alpha_2, \ldots, \alpha_n \in B \cup \{\tau\}.\\
 & \qquad (p_0 \overset{\alpha_1}{\dashrightarrow} p_1 \overset{\alpha_2}{\dashrightarrow} p_2 \cdots \overset{\alpha_n}{\dashrightarrow} p_n \textit{ implies}\\
 & \qquad ((\exists 1 \leq i \leq n.\, \alpha_i \in C \textit{ and } P[p_i] \sqsubseteq R) \textit{ or } P[p_n] \sqsubseteq Q))
\end{array}$$

Proof. The case $\mathtt{ff}$ is obvious. The case $\wedge$ has been proven in Thm. 4.37. For the next three cases refer to Fig. 4.13 for the respective meaning of labels i, o and a and for names of states of the respective right-hand MIA; note that they mostly include ν_I or ν_O.

Case $\mathtt{tt}$:
'$\Rightarrow$' is obvious. For '$\Leftarrow$' we show that $\mathcal{R}_{\mathtt{tt}} = \{(p, \mathtt{tt}_0) \mid p \in P \backslash \{u_P\}\} \cup \{(p, u) \mid p \in P\}$ is a MIA-refinement relation. The pairs in the second subset are obviously correct. For the first subset recall that u is only reachable by input may-transitions. Thus for $p \overset{\omega}{\dashrightarrow} p'$ the pair $(p', \mathtt{tt}_0)$ is contained in the first subset and for $p \overset{i}{\dashrightarrow} p'$ (p', u) is contained in the second. It is easy to see that for each $p_0 \in P_0$ the pair $(p_0, \mathtt{tt}_0)$ is contained in the first subset (recall that $p_0 \neq u_P$ by assumption).

Case $[B]Q$:
'$\Rightarrow$': Consider some $p_0 \in P_0$ and $p' \in P$ with $p_0 \overset{a}{\dashrightarrow} p'$ for some $a \in B$. Since $P \sqsubseteq [B]Q$, we know that $p_0 \sqsubseteq x_0$. Thus, p' can only be matched by some q in the τ-closure of some state in Q_0, i.e. $q \in Q_0$. Then, all p'' with $p' \overset{\varepsilon}{\Longrightarrow} p''$ are matched by some element of Q_0. Thus, $P[p'] \sqsubseteq Q$.

'$\Leftarrow$': We show that $\mathcal{R} = \{(p_0, x_0) \mid p_0 \in P_0\} \cup \{(p, q) \mid p \in P, q \in Q, p \sqsubseteq q\} \cup \mathcal{R}_{\mathtt{tt}}$ is a MIA-refinement relation. The last subset is correct, as argued in case $\mathtt{tt}$. Obviously, the second subset is correct as well. Since x_0 has no must-transitions, we only have to consider the may-transitions of some $p_0 \in P_0$. Each o- and i-transition (with $o \notin B$ and $i \notin B$ respectively) of p_0 is matched by $x_0 \overset{o}{\dashrightarrow} \mathtt{tt}_0$ or $x_0 \overset{i}{\dashrightarrow} u$ respectively. The resulting pairs are in $\mathcal{R}_{\mathtt{tt}}$. Each τ-transition of p_0 can only lead to an initial state $p_0' \in P_0$. This is matched by x_0 not taking any transition and (p_0', x_0) is in the first subset. Each a-transition of p_0 with $a \in B$ leads, by assumption, to some p' with $P[p'] \sqsubseteq Q$, i.e. for each p' exists a $q_0 \in Q_0$ with $p' \sqsubseteq q_0$. Thus, (p', q_0) is contained in the second subset and we are done.

Case $\langle B_\varepsilon \rangle Q$:
'$\Rightarrow$': Consider some $p_0 \in P_0$. Since $P \sqsubseteq \langle B_\varepsilon \rangle Q$, we know that $p_0 \sqsubseteq x_{\alpha 0}$ for some $\alpha \in B_\varepsilon$. Due to the must-transition of $x_{\alpha 0}$, there exists some P' such that for each $p' \in P'$ there is a $q_0 \in Q_0$ with $p' \sqsubseteq q_0$ and with $p_0 \overset{\alpha}{\Longrightarrow} P'$ if $\alpha \in O \cup \{\varepsilon\}$ or with $p_0 \xrightarrow{\alpha}\overset{\varepsilon}{\Longrightarrow} P'$ if $\alpha \in I$. Furthermore, each p'' in the τ-closure of P' must match some $q_0' \in Q_0$, since Q_0 is τ-closed. Thus, $P[P'] \sqsubseteq Q$.

'$\Leftarrow$': We show that $\mathcal{R} = \{(p_0, x_{\alpha 0}) \mid p_0 \in P_0, \alpha \in B_\varepsilon, P[P'] \sqsubseteq Q, \alpha \in I$ and $p_0 \xrightarrow{\alpha}\overset{\varepsilon}{\Longrightarrow} P'$ or $\alpha \in O$ and $p_0 \overset{\alpha}{\Longrightarrow} P'$ or $\alpha = \varepsilon$ and $p_0 \overset{\varepsilon}{\Longrightarrow} P'\} \cup \{(p, q) \mid p \in P, q \in Q, p \sqsubseteq q\} \cup \mathcal{R}_{\mathtt{tt}}$ is a MIA-refinement relation. The last two subsets are correct, as argued in the case $[B]Q$ above. The first components of the pairs in the first subset contain all $p_0 \in P_0$. Consider such a p_0 of a pair $(p_0, x_{\alpha 0})$. Each a-may-transition of p_0 (including a-transitions with $a \in B_\varepsilon$) can be matched by a transition from $x_{\alpha 0}$ to $\mathtt{tt}_0$ or u, and the resulting pairs are contained in $\mathcal{R}_{\mathtt{tt}}$. A τ-transition of p_0 can only lead to an initial state $p_0' \in P_0$. This is matched by $x_{\alpha 0} \overset{\varepsilon}{\Longrightarrow} x_{\beta 0}$ (possibly equal to $x_{\alpha 0}$) such that $(p_0', x_{\beta 0})$ is in the first subset. The α-must-transition $x_{\alpha 0} \xrightarrow{\alpha} Q_0$ is matched by the weak α-must-transition to P' that p_0 must have by assumption. Note that the type of weak transition fits the refinement for $\alpha \in I$ and for $\alpha \in O \cup \{\tau\}$. Since $P[P'] \sqsubseteq Q$, we have for each $p' \in P'$ some $q_0 \in Q_0$ with $p' \sqsubseteq q_0$. The pairs (p', q_0) are in the second subset and we are done.

Case $Q \vee R$:
'$\Rightarrow$': Consider some $p_0 \in P_0$. There is some $qr_0 \in Q_0 \cup R_0$ with $p_0 \sqsubseteq qr_0$; w.l.o.g. $qr_0 \in Q_0$. Since each p_0' in the τ-closure of p_0 can be matched by some q_0' in the τ-closure of qr_0, we have $P[p_0] \sqsubseteq Q$ and are done.

'$\Leftarrow$': Consider some $p_0 \in P_0$. We have $P[p_0] \sqsubseteq Q$ or $P[p_0] \sqsubseteq R$, say the former. Thus there is some $q_0 \in Q_0$ with $p_0 \sqsubseteq q_0$. Since q_0 is an initial state of $Q \vee R$, we are done.

Case $\Box_B Q$:
'$\Rightarrow$': Consider some $p_0, p_1, \ldots, p_n$ and $\alpha_1, \ldots, \alpha_n$ as in the claim. Since $P \sqsubseteq \Box_B Q$ and all $\alpha_l \in B \cup \{\tau\}$, the may-transitions must be matched by $\Box_B Q$ using transitions resulting from (May). Thus p_n is matched by some $\{q_1, \ldots, q_n, q_0\}$ for some $q_0 \in Q_0$. By Lem. 4.47 this implies $p_n \sqsubseteq q_0$.

If $p_n \overset{\varepsilon}{\Longrightarrow} p$, we extend the path of the α_i and get $p_n \sqsubseteq q_0$ for the same reason. Thus $P[p_n] \sqsubseteq Q$.

'$\Leftarrow$': We partition the state space $\Box_B Q$ into $\overline{Q} \,\dot\cup\, K$ as in Def. 4.45. Recall that we write $q \in \bigwedge\{q_1, \dots, q_n\}$ if $q \in \{q_1, \dots, q_n\}$ for ease of notation.

We show that $\mathcal{R} = \{(p, \overline{q}) \mid \overline{q} \in \overline{Q}, \exists p_0 \in P_0, w \in B^*.\, p_0 \overset{w}{\Longrightarrow} p$ and $\forall q_k \in \overline{q}.\, p \sqsubseteq q_k\} \cup \sqsubseteq$ is a MIA-refinement relation.

To see that this suffices, choose $p_0 \in P_0$. By assumption with $n = 0$, we have $P[p_0] \sqsubseteq Q$, i.e. $\exists q_0 \in Q_0.\, p_0 \sqsubseteq q_0$. For $\overline{q}_0 = \{q_0\}$, we have $(p_0, \overline{q}_0)$ in the first subset of $\mathcal{R}$. Hence, $\forall p_0 \in P_0\ \exists \overline{q_0} \in (\Box_B Q)_0.\, (p_0, \overline{q_0}) \in \mathcal{R}$.

The correctness of $\sqsubseteq$ is obvious. We take a pair $(p, \overline{q})$ due to p_0 and w from the first subset and check the items of Def. 4.4; assume $\overline{q} \neq \{u_Q\}$.

(i) Since $\overline{q} \neq \{u_Q\}$, there is some $q_k \in \overline{q}$ with $q_k \neq u$. Now $p \sqsubseteq q_k$ implies $p \neq u_P$.

(ii) $\overline{q} \xrightarrow{i} \overline{Q}'$ is analogous to the next case.

(iii) Consider $\overline{q} \xrightarrow{\omega} \overline{Q}'$ with $\overline{q} = \{q_1, \dots, q_l, \dots, q_n\}$, $q_l \xrightarrow{\omega} Q'_l$ and, depending on whether $\omega \in B \cup \{\tau\}$ or not, we have
1. $\overline{Q}' = \{\{q'_1, \dots, q'_n, q_0\} \mid q'_l \in Q'_l, q_0 \in Q_0, \forall k \in \{1, \dots, n\} \setminus \{l\}.\, q_k \overset{\hat{\omega}}{\Longrightarrow} q'_k\}$, or
2. $\overline{Q}' = \{\bigwedge\{q'_1, \dots, q'_n\} \mid q'_l \in Q'_l, \forall k \in \{1, \dots, n\} \setminus \{l\}.\, q_k \overset{\hat{\omega}}{\Longrightarrow} q'_k\}$.

Since $\forall q_k \in \overline{q}.\, p \sqsubseteq q_k$, we have $p \sqsubseteq q_l$ and therefore there exists some $p \overset{\hat{\omega}}{\Longrightarrow} P'$ matching $q_l \xrightarrow{\omega} Q'_l$. Consider some $p' \in P'$ and $q'_l \in Q'_l$ with $p' \sqsubseteq q'_l$. Because of syntactic consistency, we have $p \overset{\hat{\omega}}{\Longrightarrow} p'$. Again by $\forall q_k \in \overline{q}.\, p \sqsubseteq q_k$ and Prop. 4.5(iv) we get $\forall k \in \{1, \dots, n\} \setminus \{l\}\ \exists q'_k.\, q_k \overset{\hat{\omega}}{\Longrightarrow} q'_k \wedge p' \sqsubseteq q'_k$. Thus, in Case 2 we are done, since we have a $\overline{q}' = \bigwedge\{q'_1, \dots, q'_n\} \in \overline{Q_l}'$ with $\forall q'_k \in \overline{q}.\, p' \sqsubseteq q'_k$ and thus $p' \sqsubseteq \overline{q}'$ by Thm. 4.37.2.

For Case 1, we also have $p_0 \overset{w}{\Longrightarrow} p \overset{\hat{\omega}}{\Longrightarrow} p'$ with $w\hat{\omega} \in B^*$. Thus $P[p'] \sqsubseteq Q$, implying $\exists q_0 \in Q_0.\, p' \sqsubseteq q_0$. Again, we have a $\overline{q}' = \{q'_1, \dots, q'_n, q_0\} \in \overline{Q_l}'$ with $\forall q'_k \in \overline{q}.\, p' \sqsubseteq q'_k$ and thus $(p', \overline{q}') \in \mathcal{R}$.

(iv) Consider $p \overset{\alpha}{\dashrightarrow} p'$. From $\forall q_k \in \overline{q}.\, p \sqsubseteq q_k$ we get $\forall q_k \in \overline{q}\ \exists q'_k.\, q_k \overset{\hat{\alpha}}{\Longrightarrow} q'_k \wedge p' \sqsubseteq q'_k$. If $\forall k.\, q_k = q'_k$, we have $(p', \overline{q})$ in the first subset of $\mathcal{R}$ and are done. Otherwise, by construction of $\Box_B Q$, we get $\overline{q} \overset{\alpha}{\dashrightarrow}$.

If $\alpha \in B \cup \{\tau\}$, then $\overline{q} \overset{\alpha}{\dashrightarrow} \overline{q}' = \{q'_1, \dots, q'_n, q_0\}$ for each $q_0 \in Q_0$ by (May). We already established $p' \sqsubseteq q'_k$ for all $q'_k \in \{q'_1, \dots, q'_n\}$. We now have $p_0 \overset{w}{\Longrightarrow} p \overset{\alpha}{\dashrightarrow} p'$ with $w\alpha \in B^*$. Therefore, $P[p'] \sqsubseteq Q$ implying $\exists q_0 \in Q_0.\, p' \sqsubseteq q_0$. Thus, $p' \sqsubseteq q$ for all $q \in \{q'_1, \dots, q'_n, q_0\}$ implying $(p', \overline{q}') \in \mathcal{R}$.

If $\alpha \notin B \cup \{\tau\}$, then $\overline{q} \overset{\alpha}{\dashrightarrow} c' = \bigwedge\{q'_1, \ldots, q'_n\}$ for each $q_0 \in Q_0$ by (MayK) . As we established $p' \sqsubseteq q'_k$ for all $q'_k \in c'$, we get $p' \sqsubseteq c'$.

Case $Q_B \mathsf{W}_C R$:
'$\Rightarrow$': Consider some $p_0, p_1, \ldots, p_n$ and $\alpha_1, \ldots, \alpha_n$ as in the claim. Since $P \sqsubseteq Q_B \mathsf{W}_C R$ and all $\alpha_j \in B \cup \{\tau\}$, the may-transitions must be matched by $Q_B \mathsf{W}_C R$ using transitions resulting from (May), or there is a transition $p_{i-1} \overset{\alpha_i}{\dashrightarrow} p_i$ with $\alpha_i \in C$ due to (MayW). In the first case, p_n is matched by some $\{q_1, \ldots, q_m, q_0\}$ for some $q_0 \in Q_0$; by Lem. 4.50 this implies $p_n \sqsubseteq q'_0$. If $p_n \overset{\varepsilon}{\Longrightarrow} p$, we extend the path of the α_j and get $p_n \sqsubseteq q_0 \in Q_0$ for the same reason. Thus $P[p_n] \sqsubseteq Q$. In the second case, p_i is matched by some $\{q_1, \ldots, q_m, r_0\}$ for some $r_0 \in R_0$; by Lem. 4.50 this implies $p_i \sqsubseteq r_0$. If $p_i \overset{\varepsilon}{\Longrightarrow} p'$, then by Prop. 4.5 there is some $r_0 \overset{\varepsilon}{\Longrightarrow} r'_0$ with $p' \sqsubseteq r'_0$ and $r'_0 \in R_0$ by initial τ-closure of R. Thus, $P[p_i] \sqsubseteq R$.

'$\Leftarrow$': We partition the state space $Q_B \mathsf{W}_C R$ into $\overline{Q} \mathbin{\dot\cup} K \mathbin{\dot\cup} K'$ as in Def. 4.48 and again write $q \in \bigwedge\{q_1, \ldots, q_m\}$ if $q \in \{q_1, \ldots, q_m\}$ for ease of notation.

We show that $\mathcal{R} = \{(p_n, \overline{q}) \mid \overline{q} \in \overline{Q}, \exists p_0 \in P_0, \alpha_1, \ldots, \alpha_n \in B.\, p_0 \overset{\alpha_1}{\dashrightarrow} p_1 \overset{\alpha_2}{\dashrightarrow} \cdots p_{n-1} \overset{\alpha_n}{\dashrightarrow} p_n$ and $\forall q_k \in \overline{q}.\, p_n \sqsubseteq q_k$ and there is no $i \in \{1, \ldots, n\}$ such that $\alpha_i \in C$ and $P[p_i] \sqsubseteq R\} \cup \sqsubseteq$ is a MIA-refinement relation.

To see that this suffices, choose $p_0 \in P_0$. By assumption with $n = 0$, we have $P[p_0] \sqsubseteq Q$, i.e. $\exists q_0 \in Q_0.\, p_0 \sqsubseteq q_0$. For $\overline{q}_0 = \{q_0\}$, we have $(p_0, \overline{q}_0)$ in the first subset of $\mathcal{R}$. Hence, $\forall p_0 \in P_0\ \exists \overline{q_0} \in (Q_B \mathsf{W}_C R)_0.\, (p_0, \overline{q_0}) \in \mathcal{R}$.

The correctness of $\sqsubseteq$ is obvious. So we take a pair $(p_n, \overline{q})$ due to p_0 and $w = \alpha_1 \cdots \alpha_n$ from the first subset and check the items of Def. 4.4; assume $\overline{q} \neq \{u_Q\}$.

(i) Since $\overline{q} \neq \{u_Q\}$, there is some $q_k \in \overline{q}$ with $q_k \neq u_Q$. Now $p \sqsubseteq q_k$ implies $p \neq u_P$ with.

(ii) $\overline{q} \overset{i}{\longrightarrow} \overline{Q}'$ is analogous to the next case.

(iii) Consider $\overline{q} \overset{\omega}{\longrightarrow} \overline{Q}'$ with $\overline{q} = \{q_1, \ldots, q_l, \ldots, q_m\}$, $q_l \overset{\omega}{\longrightarrow} Q'_l$ and, depending on whether $\omega \in B \cup \{\tau\}$ or not, we have 1. $\overline{Q}' = \{\{q'_1, \ldots, q'_m, q_0\} \mid q'_l \in Q'_l, q_0 \in Q_0, \forall k \in \{1, \ldots, m\} \backslash \{l\}.\, q_k \overset{\hat{\omega}}{\Longrightarrow} q'_k\} \cup \{\bigwedge\{q'_1, \ldots, q'_m, r_0\} \mid \omega \in C, q'_l \in Q'_l, r_0 \in R_0, \forall k \in \{1, \ldots, m\} \setminus \{l\}.\, q_k \overset{\omega}{\Longrightarrow} q'_k\}$ or

2. $\overline{Q}' = \{\bigwedge\{q'_1, \ldots, q'_m\} \mid q'_l \in Q'_l, \forall k \in \{1, \ldots, m\} \setminus \{l\}.\, q_k \overset{\hat{\omega}}{\Longrightarrow} q'_k\}$.

Since $\forall q_k \in \overline{q}.\, p \sqsubseteq q_k$, we have $p \sqsubseteq q_l$ and therefore there exists some $p \overset{\hat{\omega}}{\Longrightarrow} P'$ matching $q_l \overset{\omega}{\longrightarrow} Q'_l$. Consider some $p' \in P'$ and $q'_l \in Q'_l$ with $p' \sqsubseteq q'_l$. Because of syntactic consistency, we have $p \overset{\hat{\omega}}{\Longrightarrow} p'$.

Again by $\forall q_k \in \overline{q}.\, p \sqsubseteq q_k$ and Prop. 4.5 we get $\forall k \in \{1, \ldots, m\} \setminus \{l\}\ \exists q'_k.\, q_k \overset{\hat{\omega}}{\Longrightarrow} q'_k \wedge p' \sqsubseteq q'_k$. Thus, in Case 2 we are done, since we have a $\overline{q}' = \bigwedge\{q'_1, \ldots, q'_m\} \in \overline{Q}'$ with $\forall q'_k \in \overline{q}'.\, p' \sqsubseteq q'_k$ and thus $p' \sqsubseteq \overline{q}'$.

For Case 1, we also have $p_0 \overset{\alpha_1\cdots\alpha_n}{\Longrightarrow} p_n \overset{\beta_1}{\dashrightarrow} p_{n+1} \overset{\beta_2}{\dashrightarrow} \cdots \overset{\beta_l}{\dashrightarrow} p_{n+l} = p'$ with $\widehat{\beta_1\beta_2\cdots\beta_l} = \hat{\omega} \in B \cup \{\varepsilon\}$. Now, by the initial assumption, there are two possibilities: The first is the existence of some p_{n+i} and $\beta_i \in C$ with $1 \leq i \leq l$ and $P[p_{n+i}] \sqsubseteq R$. In this case, $\exists r_0 \in R_0.\, p_{n+i} \sqsubseteq r_0$ and by Prop. 4.5 and initial τ-closure of R, we get $\exists r'_0 \in R_0.\, p' \sqsubseteq r'_0$. Thus, we have $\overline{q}' = \bigwedge\{q'_1, \ldots, q'_m, r'_0\} \in \overline{Q}'$ with $\forall q' \in \overline{q}'.\, p' \sqsubseteq q'$ (possibly $q' = r'_0$) implying $p' \sqsubseteq \overline{q}'$. Otherwise, $P[p'] \sqsubseteq Q$, in which case we have a $q'_0 \in Q_0$ with $p' \sqsubseteq q'_0$; thus, $\overline{q}' = \{q'_1, \ldots, q'_m, q'_0\} \in \overline{Q}'$ and $\forall q'_k \in \overline{q}.\, p' \sqsubseteq q'_k$ implying $(p', \overline{q}') \in \mathcal{R}$.

(iv) Consider $p_n \overset{\alpha}{\dashrightarrow} p'$. From $\forall q_k \in \overline{q}.\, p_n \sqsubseteq q_k$ we get $\forall q_k \in \overline{q}\ \exists q'_k.\, q_k \overset{\hat{\alpha}}{\Longrightarrow} q'_k$ with $p' \sqsubseteq q'_k$. If $\forall k.\, q_k = q'_k$, we are done. Otherwise, by construction of $Q_B \mathsf{W}_C R$, we get $\overline{q} \overset{\alpha}{\dashrightarrow}$ by (May), (MayK) or (MayW).

If $\alpha \in B \cup \{\tau\}$, then $\overline{q} \overset{\alpha}{\dashrightarrow} \{q'_1, \ldots, q'_m, q_0\}$ for each $q_0 \in Q_0$ by (May) or, if in addition $\alpha \in C$, then also $\overline{q} \overset{\alpha}{\dashrightarrow} \bigwedge\{q'_1, \ldots, q'_m, r_0\}$ for each $r_0 \in R_0$, by (MayW). We already established $p' \sqsubseteq q'_k$ for all $q'_k \in \{q'_1, \ldots, q'_m\}$. We now have $p_0 \overset{\alpha_1}{\dashrightarrow} p_1 \cdots p_{n-1} \overset{\alpha_n}{\dashrightarrow} p_n \overset{\alpha}{\dashrightarrow} p'$ with $\alpha_1, \ldots, \alpha_n, \alpha \in B$. By assumption, 1. $\alpha \in C$ and $P[p'] \sqsubseteq R$ or, otherwise, we have 2. $P[p'] \sqsubseteq Q$. In the second case $P[p'] \sqsubseteq Q$ implies $\exists q_0 \in Q_0.\, p' \sqsubseteq q_0$. Thus, $p' \sqsubseteq q$ for all $q \in \overline{q}' = \{q'_1, \ldots, q'_m, q_0\}$ implying $(p', \overline{q}') \in \mathcal{R}$. In the first case, $P[p'] \sqsubseteq R$ implies $p' \sqsubseteq q$ for all $q \in \overline{q}' = \bigwedge\{q_1, \ldots, q_m, r_0\}$ for some $r_0 \in R_0$. and thus $p' \sqsubseteq \overline{q}'$.

If $\alpha \notin B \cup \{\tau\}$, then $\overline{q} \overset{\alpha}{\dashrightarrow} \overline{q}' = \bigwedge\{q'_1, \ldots, q'_m\}$ by (MayK). As we established $p' \sqsubseteq q'_k$ for all $q'_k \in \overline{q}'$ and we get $p' \sqsubseteq \overline{q}'$. □

Now the desired and central result is just a direct corollary. Then, transitivity of $\sqsubseteq$ shows that refinement preserves satisfaction.

Corollary 4.52 (Compatibility). *1. $P \models \varphi$ if and only if $P \sqsubseteq \varphi$.*

2. $P \sqsubseteq Q$ and $Q \models \varphi$ imply $P \models \varphi$.

With our main result achieved, we can further discuss some aspects of our logic, particularly the lack of negation and the non-standard way disjunction is defined.

In a setting where refinement preserves satisfaction, classic negation cannot be expressed in the logics. Given $P \sqsubseteq Q$ and not $Q \models \varphi$, we would have

$Q \models \neg\varphi$, $P \models \neg\varphi$ and not $P \models \varphi$. Thus, all such P and Q would satisfy the same formulae, which hardly makes sense. This is already remarked in [10]. In fact, since all non-universal MIA satisfy and, hence, refine $\mathtt{tt}$, all these MIAs would satisfy the same formulae, and the logics would be useless.

We can now formally show that defining disjunction as $P \models \varphi \vee' \psi$ if $P \models \varphi$ or $P \models \psi$ would prevent us from achieving our main result. Consider the formulae $\varphi_o = \langle o \rangle \mathtt{tt}$ and $\varphi_{o'} = \langle o' \rangle \mathtt{tt}$. For our main result there have to be MIA representations R_o and $R_{o'}$ with $P \models \varphi_o$ if and only if $P \sqsubseteq R_o$ for any MIA P (analogously for o'). Now consider again MIAs P and Q from Fig. 4.4. Since $P \models \varphi_o$ and $Q \models \varphi_{o'}$, we must have $P \sqsubseteq R_o$ and $Q \sqsubseteq R_{o'}$ if our main result holds. Disjunction on MIAs must satisfy Thm. 4.31 and Cor. 4.32, so we get $P \vee Q \sqsubseteq R_o \vee R_{o'}$. However, according to the above definition, $P \vee Q \not\models \varphi_o \vee' \varphi_{o'}$, since φ_o fails due to q_0 and $\varphi_{o'}$ due to p_0.

With the characterization result, we can also easily prove that our operators are precongruences:

Theorem 4.53 (Precongruences)**.** *The temporal operators are precongruences: For all non-universal MIAs P,Q and R,* $P \sqsubseteq Q$ *implies* $\langle a \rangle P \sqsubseteq \langle a \rangle Q$, $[a]P \sqsubseteq [a]Q$, $P_B \mathsf{W}_C R \sqsubseteq Q_B \mathsf{W}_C R$ *and* $R_B \mathsf{W}_C P \sqsubseteq R_B \mathsf{W}_C Q$.

Proof. We prove this for $[a]$. The other cases are analogous.

We first show: for some MIA R, $R \sqsubseteq [a]P$ and $P \sqsubseteq Q$ implies $R \sqsubseteq [a]Q$. $R \sqsubseteq [a]P$ implies $\forall r_0 \in R_0 \, \forall r'. \, (r_0 \overset{a}{\dashrightarrow} r'$ implies $R[r'] \sqsubseteq P)$ by Thm. 4.51. Since $P \sqsubseteq Q$ and transitivity, we get $\forall r_0 \in R_0 \, \forall r'. \, (r_0 \overset{a}{\dashrightarrow} r'$ implies $R[r'] \sqsubseteq Q)$. Again by Thm. 4.51 we get $R \sqsubseteq [a]Q$.

Now we choose $R = [a]P$ and are done. □

4.5.2 ACTL Example

To show how our heterogeneous specifications can be used, we present a small example inspired by the one in [50]. For ease of understanding and better readability, we omit ν-transitions, since they are irrelevant for this example.

We first develop a specification C for a channel that receives messages via the input *in* and forwards them via the output *out*; $\mathcal{A} = \{in, out\}$. The channel should behave like the lossy channel P in Fig. 4.14 but also satisfy the formula $\psi = \Box[in][in](en(out) \wedge dis(in))$. This formula ensures that no two messages are lost in a row. The MIA-representation of ψ and the MIA that ψ is constructed from are shown in Fig. 4.14; some reductions are applied here: the various copies of $\mathtt{tt}_0$ added according to Fig. 4.13 are merged as well as $\mathtt{tt}_0 \wedge \mathtt{tt}_0$ and $\mathtt{tt}_0$. The resulting MIA ψ turns out to be rather intuitive: In the state $\psi_0 = \{s_0\}$, no *in* or *out* has been performed yet. In $\psi_1 = \{s_1, s_0\}$,

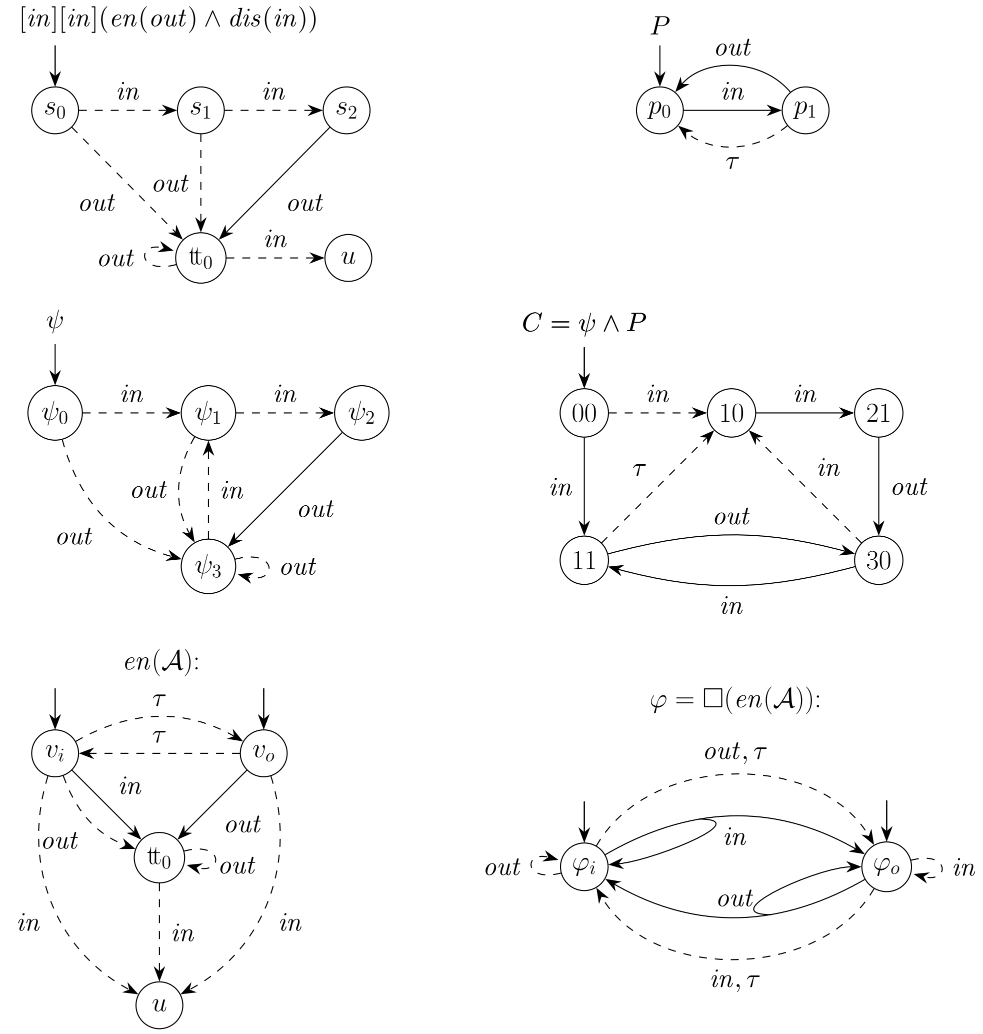

Figure 4.14: MIAs and some construction steps for the example. ν-transitions are omitted

the last action was an *in* and the one before was not, i.e. another *in* can be performed, but *out* is also enabled. $\psi_2 = \{s_2, s_1, s_0\}$ means that the last two actions were *in*, thus the next one must be *out*. Finally, $\psi_3 = \{\mathit{tt}_0, x_0\}$ is similar to ψ_0 in that no *in* has been performed since the last *out*. For specification $C = P \wedge \psi$, note that the state 20 is inconsistent and thus has been removed.

We finally demonstrate how to verify that C is deadlock-free, i.e. it can always perform *out* or receive *in*. This property can be formalized as $\varphi = \Box(en(\mathcal{A}))$, the MIA-representation of which is depicted in Fig. 4.13 as well. To show that $C \models \varphi$, it suffices to check $C \sqsubseteq \varphi$. By $\mathcal{R} = \{(00, \varphi_i), (10, \varphi_i), (21, \varphi_o), (11, \varphi_o), (30, \varphi_i)\}$ the property holds. This also proves that any implementation of C, e.g. some channel $C' \sqsubseteq C$, also satisfies the property φ, i.e. $C' \models \varphi$. Furthermore, we also know that if some complex system S has C as its component, e.g. $S = (C \parallel T)/\{out\}$, and that S satisfies some φ', then its refinement $S' = (C' \parallel T)/\{out\}$ also satisfies φ'.

To show that possible implementations of C include some MIA D, rather than proving $D \sqsubseteq C$ directly, one could also establish $D \sqsubseteq P$ and $D \models \psi$ (or equivalently $D \sqsubseteq \psi$).

4.5.3 Laws and Expressivity

Our setting and our logical operators satisfy several desirable, and often expected, laws. The following laws of propositional logic, except for distributivity, hold in all lattices where ff and tt are bottom and top elements, i.e. they follow from Thm. 4.31 and Thm. 4.37.

Proposition 4.54 (Laws of propositional logic)**.** *The following laws hold for MIAs* $P \not\sqsupseteq\!\sqsubseteq \mathfrak{U} \not\sqsupseteq\!\sqsubseteq Q$*:*

Idempotence: $P \wedge P \sqsupseteq\sqsubseteq P$ *and* $P \vee P \sqsupseteq\sqsubseteq P$
Absorption: $P \wedge (P \vee Q) \sqsupseteq\sqsubseteq P$ *and* $P \vee (P \wedge Q) \sqsupseteq\sqsubseteq P$
Neutral Elements: $P \vee \mathit{ff} \sqsupseteq\sqsubseteq P$ *and* $P \wedge \mathit{tt} \sqsupseteq\sqsubseteq P$
Null Elements: $P \wedge \mathit{ff} \sqsupseteq\sqsubseteq \mathit{ff}$ *and* $P \vee \mathit{tt} \sqsupseteq\sqsubseteq \mathit{tt}$
False/True: $\mathit{ff} \sqsubseteq P$ *and* $P \sqsubseteq \mathit{tt}$
And/Or: $P \wedge Q \sqsubseteq P$ *and* $P \sqsubseteq P \vee Q$
Lower/Upper Bound: $P \wedge Q \sqsupseteq\sqsubseteq P \quad \Leftrightarrow \quad P \vee Q \sqsupseteq\sqsubseteq Q \quad \Leftrightarrow \quad P \sqsubseteq Q$

Furthermore,
Distributivity I: $P \wedge (Q \vee R) \sqsupseteq\sqsubseteq (P \wedge Q) \vee (P \wedge R)$
Distributivity II: $P \vee (Q \wedge R) \sqsupseteq\sqsubseteq (P \vee Q) \wedge (P \vee R)$

Proof. We show Distributivity I explicitly, by showing that $S \sqsubseteq P \wedge (Q \vee R)$ if and only if $S \sqsubseteq (P \wedge Q) \vee (P \wedge R)$. Distributivity II can be shown similarly.

Assume $S \sqsubseteq P \wedge (Q \vee R)$. By Thm. 4.37.3, $S \sqsubseteq P$ and $S \sqsubseteq Q \vee R$. Then, by definition of refinement and disjunction, there is for each $s_0 \in S_0$ a $p_0 \in P_0$ with $s_0 \sqsubseteq p_0$ and some $q_0 \in Q_0 \cup R_0$ (say $\in Q_0$) with $s_0 \sqsubseteq q_0$. Then, by Thm. 4.37.1 and 2, the state $p_0 \wedge q_0$ is consistent, an initial state of $P \wedge Q$ and thus of $(P \wedge Q) \vee (P \wedge R)$, and it is refined by s_0. Similarly, assuming $S \sqsubseteq (P \wedge Q) \vee (P \wedge R)$, each s_0 refines some p_0 and some q_0 or r_0 and thus $S \sqsubseteq P$ and $S \sqsubseteq Q \vee R$. □

More interestingly, we also have several laws for the temporal operators.

Proposition 4.55 (Laws of temporal logic)**.** *The following laws hold for* $B, C \subseteq \mathcal{A}$, $B_\varepsilon, C_\varepsilon \subseteq \mathcal{A} \cup \{\varepsilon\}$, *and MIAs* $P \not\sqsupseteq\!\!\not\sqsubseteq \mathfrak{U} \not\sqsupseteq\!\!\not\sqsubseteq Q$*:*

(1) $[B](P \wedge Q) \sqsupseteq\!\sqsubseteq [B]P \wedge [B]Q$

(2) $\Box_B(P \wedge Q) \sqsupseteq\!\sqsubseteq \Box_B P \wedge \Box_B Q$

(3) $\Box_B P \sqsupseteq\!\sqsubseteq P \wedge [B]\Box_B P$

(4) $\mathit{en}(B) \wedge \mathit{dis}(B) \sqsupseteq\!\sqsubseteq \mathrm{ff}$

(5) $\mathit{dis}(B) \wedge [C]P \sqsupseteq\!\sqsubseteq \mathit{dis}(B)$ *if* $C \subseteq B$

(6) $[B]P \wedge [C]P \sqsupseteq\!\sqsubseteq [B \cup C]P$

(7) $\langle B_\varepsilon \rangle P \vee \langle C_\varepsilon \rangle P \sqsubseteq \langle B_\varepsilon \cup C_\varepsilon \rangle P$ *the reverse is not true in general*

Proof. We prove these equivalences by showing that both sides have the same refinements. Since these refinements include both sides themselves, we are done. We will make heavy use of the characterization given by Thm. 4.51. To be closer to this theorem, we use P as variable for the refinements showing e.g. $P \sqsubseteq [B](Q \wedge R) \Leftrightarrow P \sqsubseteq [B]Q \wedge [B]R$ for (1). For reasons of readability we omit sets where these are clear from the representatives, e.g. writing $\forall p_0$ instead of $\forall p_0 \in P_0$.

(1): $P \sqsubseteq [B](Q \wedge R)$ is equivalent to $\forall p_0 \forall a \in B \forall p'.\, p_0 \overset{a}{\dashrightarrow} p'$ implies $(P[p'] \sqsubseteq Q$ and $P[p'] \sqsubseteq R)$ by Thm. 4.51.3 and 5. By logical transformations we get $(\forall p_0 \forall a \in B \forall p'.\, p_0 \overset{a}{\dashrightarrow} p'$ implies $P[p'] \sqsubseteq Q)$ and $(\forall p_0 \forall a \in B \forall p'.\, p_0 \overset{a}{\dashrightarrow} p'$ implies $P[p'] \sqsubseteq R)$, which again by Thm. 4.51.3 and 5 is equivalent to $P \sqsubseteq [B]Q \wedge [B]R$.

(2) Can be proven similarly to (1) using Thm. 4.51.7 instead of 3.

(3): Applying Thm. 4.51.7 to the left hand side, and Thm. 4.51.5 and 3 (and the definition of refinement) to the right hand side, it remains to show the following equivalence:

$\forall p_0\, \forall p_1, \dots, p_n\, \forall \alpha_1, \dots, \alpha_n \in B \cup \{\tau\}.\ (p_0 \overset{\alpha_1}{\dashrightarrow} p_1 \overset{\alpha_2}{\dashrightarrow} \cdots p_n$ implies $P[p_n] \sqsubseteq Q)$
is equivalent to

$\forall p_0.\, P[p_0] \sqsubseteq Q$ and $\forall p_0 \forall a \in B \forall p'.\ p_0 \overset{a}{\dashrightarrow} p'$ implies $P[p'] \sqsubseteq \Box_B Q$.

The case $\widehat{\alpha_1 \cdots \alpha_n} = \varepsilon$ (including $n = 0$) in the upper statement corresponds to the first conjunct in the lower statement. For all other cases note that, for k the lowest index where $\alpha_k \neq \tau$, we have $p_{k-1} \in P_0$ by initial τ-closure. Therefore this is covered by the second conjunct of the lower statement.

(4): Applying Thm. 4.51.5, 1-4 to the left hand side yields
$\forall p_0 \; \exists a \in B$ such that either $a \in I$ and $p_0 \xrightarrow{a}\overset{\varepsilon}{\Longrightarrow} P'$ or $a \in O$ and $p_0 \overset{a}{\Longrightarrow} P'$
and
$\forall p_0 \forall a \in B. p_0 \overset{a}{\not\dashrightarrow}$.

Recalling initial τ-closure for the output-case, we see that any p_0 would violate syntactic consistency. Thus $P_0 = \emptyset$, which means $P_0 \sqsupseteq\sqsubseteq \mathrm{ff}$.

(5): Applying Thm. 4.51.5, 3, 2 to the left hand side yields $\forall p_0 \forall a \in B. \, p_0 \overset{a}{\not\dashrightarrow}$ and $\forall p_0 \forall a \in C \forall p'. \, (p_0 \overset{a}{\dashrightarrow} p'$ implies $P[p'] \sqsubseteq Q)$. Since $C \subseteq B$, the final implication holds by ex falso quodlibet. Thus only the first part remains which is equivalent to $P \sqsubseteq \mathit{dis}(B)$.

(6): Obvious from the definition.

(7): For readability and brevity, we write that p *does* α if $\exists P'. \, P[P'] \sqsubseteq Q$ and either $\alpha \in I$ and $p \xrightarrow{\alpha}\overset{\varepsilon}{\Longrightarrow} P'$ or $\alpha \in O \cup \{\varepsilon\}$ and $p \overset{\alpha}{\Longrightarrow} P'$.

Applying Thm. 4.51.6 and 4 to $P \sqsubseteq \langle B_\varepsilon \rangle Q \vee \langle C_\varepsilon \rangle Q$, we get $\forall p_0. \big((\forall p_0'. \, p_0 \overset{\varepsilon}{\Longrightarrow} p_0'$ implies $\exists \alpha \in B_\varepsilon. \, p_0'$ does $\alpha)$ or $(\forall p_0'. \, p_0 \overset{\varepsilon}{\Longrightarrow} p_0'$ implies $\exists \alpha \in C_\varepsilon. \, p_0'$ does $\alpha)\big)$. This implies, but is not equivalent to $(\forall p_0 \exists \alpha \in B_\varepsilon \cup C_\varepsilon. \, p_0$ does $\alpha)$. By Thm. 4.51.4, we get $P \sqsubseteq \langle B_\varepsilon \cup C_\varepsilon \rangle Q$.

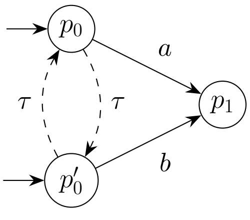

Figure 4.15: MIA P used in the proof of Prop. 4.57.

To see that the reverse of (7) does not hold, consider the MIA P from Fig. 4.15 with $P \models \langle a, b \rangle \mathrm{tt}$. However, $P \not\models \langle a \rangle \mathrm{tt} \vee \langle b \rangle \mathrm{tt}$, since otherwise all states in the τ-closure (e.g. of p_0) would have to have an a-must-transition or all would have to have a b-must-transition. □

Items (1), (4) and (5) can also be proven very nicely by considering the constructions. For (1) the resulting MIAs are isomorphic. For (4) all initial states of the conjunctive product of $\mathit{en}(B)$ and $\mathit{dis}(B)$ are immediately inconsistent and thus removed. For (5) the already conjunctive product of $\mathit{dis}(B)$ and $[C]P$ is isomorphic to $\mathit{dis}(B)$.

Let us now consider the expressiveness of $[B]$ and $\langle B_\varepsilon\rangle$ compared to $[\alpha]$ and $\langle\alpha\rangle$. It is easy to see that $[B]$ does not add expressiveness compared to $[\alpha]$ if B is finite, because of Prop. 4.55(6). With $\langle B_\varepsilon\rangle$, however, we have increased expressiveness already in the finite case. We show this for a fragment of our logic, where the Unless-Operator is excluded and $\langle B_\varepsilon\rangle$ is restricted to a single action or ε. We take this opportunity to also introduce another fragment which will be used in the next section.

Definition 4.56. *An ACTL-formula is called* Unless-free*, if the Unless-Operator does not appear in it. The* Unless-free-fragment $\mathfrak{Wf}$ *of ACTL consists of all Unless-free formulae. We define* $\mathcal{L}_{\mathfrak{Wf}}(P) =_{df} \{\varphi \mid P \models \varphi, \varphi \textit{ is Unless-free}\}$.

Further, we define the HML-fragment $\mathfrak{H}$ *of ACTL as the fragment of ACTL which contains only Unless-free formulae and where the* $[B]$ *and* $\langle B_\varepsilon\rangle$ *operators only occur with* $|B| = 1$ *and* $|B_\varepsilon| = 1$ *respectively. We define* $\mathcal{L}_{\mathfrak{H}}(P) =_{df} \{\varphi \mid P \models \varphi, \varphi \textit{ is in } \mathfrak{H}\}$.

Note that the Always-Operator does also not appear in formulae of either fragment, since it is a shorthand using Unless.

Proposition 4.57 (Expressivity: $\langle B_\varepsilon\rangle$ vs. $\langle\alpha\rangle$)**.** *No formula in* $\mathfrak{Wf}$ *with* $|B_\varepsilon| = 1$ *for every occurrence of* $\langle B_\varepsilon\rangle$ *is equivalent to* $\langle a, b\rangle\mathtt{tt}$.

Proof. Consider the MIA P from Fig. 4.15 and a 'deadlock'-MIA D which has a single non-universal initial state d_0 and no transitions. It is easy to see that $P \models \langle a, b\rangle\mathtt{tt}$ and $D \not\models \langle a, b\rangle\mathtt{tt}$. We show by structural induction that for any formula $\varphi \in \mathfrak{Wf}$ (with $|B_\varepsilon| = 1$ for every occurrence of $\langle B_\varepsilon\rangle$) we have $P \not\models \varphi$ or $D \models \varphi$, so that the desired equivalence for φ is violated by P or by D.

For $\varphi = \mathtt{tt}$ we have $D \models \varphi$ and for $\varphi = \mathtt{ff}$ we have $P \not\models \varphi$. For any action c and any formula ψ, we have $P \not\models \langle c\rangle\psi$; in particular, $P \not\models \langle a\rangle\psi$, since $p_0' \stackrel{a}{\not\longrightarrow}$. Similarly, $D \models [B]\psi$, since $d_0 \stackrel{c}{\not\dashrightarrow}$ for all $c \in B$ for any $B \subseteq \mathcal{A}$. Now, consider $\varphi = \langle\varepsilon\rangle\psi$. Since none of the initial states under consideration has any outgoing τ-must-transition (and by Lem. 4.42.1), we have $P \models \varphi \Leftrightarrow P \models \psi$ and $D \models \varphi \Leftrightarrow D \models \psi$. Thus, by induction, $P \not\models \varphi$ or $D \models \varphi$.

Now we only have to consider disjunctions and conjunctions of formulae we already considered. By applying the distributivity laws of Prop. 4.54, any combination can be written as a disjunction φ of conjunctions. Since each initial state of P is in the τ-closure of the other, $P \models \varphi$ if and only if $P \models \psi$ for some disjunct ψ of φ. By definition, $P \models \psi$ if and only if $P \models \theta$ for all conjuncts θ of ψ. Assume $P \models \varphi$ and apply the induction hypothesis to each

θ; this gives $D \models \theta$ for all conjuncts θ of ψ. Thus, $D \models \psi$ and since it has only one initial state, we get $D \models \varphi$ and are done. □

4.5.4 HML-Characerization

In this section we examine in how far a HML-style characterization can be applied to our setting. As we will show, refinement for MIAs is not characterized by inclusion of ACTL-formulae. There are, however, interesting variants that are worth examining.

For this, we return to the whole setting by lifting the requirement for initial τ-closure. The characterization requires image-finiteness, as usual.

Definition 4.58 (Image Finiteness and τ-Freedom)**.** *A MIA P is* image finite *if P_0 is finite and for all $\alpha \in \mathcal{A}_\tau$ and all states $p \in P$ the set $\{p' \mid p \overset{\alpha}{\Longrightarrow} p'\}$ is finite. P is called* τ-free *if it has no τ-transitions, i.e. $p \overset{\alpha}{\dashrightarrow} p'$ implies $\alpha \neq \tau$.*

We present a state-based HML-style-logic and characterise MIA-refinement with it. In a τ-free setting, this logic is a subset of ACTL, but otherwise they are incomparable. We do not have a translation to MIAs and indeed cannot have one. However, satisfaction is still retained under refinement (cf. Cor. 4.52.2).

For the HML-operators we use different notation to highlight the difference to ACTL. We still exclude $\mathfrak{U}$ and u from satisfying any formulae, but we will include them in our considerations for characterising MIA-refinement.

Definition 4.59 (Hennessy-Milner-Logic)**.** *Let $\alpha \in \mathcal{A} \cup \{\varepsilon\}$, P be a MIA and $p \neq u$ one of its states.* (Classic) HML-formulae *φ are given by:*

$\varphi ::= \mathtt{tt} \mid \mathtt{ff} \mid [[\alpha]]\varphi \mid \langle\langle\alpha\rangle\rangle\varphi \mid \varphi \wedge \varphi \mid \varphi \vee \varphi.$

The semantics of classic HML-formulae is defined inductively as follows:

$p \models \mathtt{tt}$	*always*
$p \models \mathtt{ff}$	*never*
$p \models [[\alpha]]\varphi$	*if* $\forall p' \in P$ *the following hold:* $p \overset{\alpha}{\Longrightarrow} p'$ *implies* $p' \models \varphi$
$p \models \langle\langle\alpha\rangle\rangle\varphi$	*if* $\exists P' \subseteq P.\, (\forall p' \in P'.\, p' \models \varphi)$ *and* *either* $\alpha \in I$ *and* $p_0 \xrightarrow{\alpha}\overset{\varepsilon}{\Longrightarrow} P'$ *or* $\alpha \in O \cup \{\varepsilon\}$ *and* $p_0 \overset{\alpha}{\Longrightarrow} P'$
$p \models \varphi \wedge \psi$	*if* $p \models \varphi$ *and* $p \models \psi$
$p \models \varphi \vee \psi$	*if* $p \models \varphi$ *or* $p \models \psi$

We lift these definitions to MIAs by defining $P \models \varphi$ if $\forall p_0 \in P_0.\, p_0 \models \varphi$. We define $\mathcal{L}_{HML}(p) =_{df} \{\varphi \mid p \models \varphi, \varphi$ is a HML-formula$\}$ as the set of HML-formulae that p satisfies and $\mathcal{L}_{HML}(P)$ analogously.

Observe that the HML-semantics above indeed coincides with the one for $\mathfrak{H}$ on τ-free MIAs, i.e. $\langle\langle\alpha\rangle\rangle$ corresponds to $\langle\alpha\rangle$ and $[[\alpha]]$ to $[\alpha]$. As mentioned, the different notation is in order to highlight the difference in meaning, since in general the semantics do not coincide.

Proposition 4.60 (HML-characterization)**.** *Let P and Q be image finite MIAs. Then*

$$P \sqsubseteq Q \quad \Longleftrightarrow \quad \mathcal{L}_{HML}(Q) \subseteq \mathcal{L}_{HML}(P).$$

Proof. '$\Longrightarrow$': First we prove the claim for states p and q and then generalise it to MIAs.

Assume $p \sqsubseteq q$ and $q \models \varphi$ for some HML-formula φ. Note that this implies $q \neq u_Q$ and thus $p \neq u_P$. Using structural induction, we show $p \models \varphi$ for each possible form of φ.

- $\varphi = \mathtt{tt}$ or $\varphi = \mathtt{ff}$: It is obvious that $p \models \mathtt{tt}$ holds and that assuming $q \models \mathtt{ff}$ immediately leads to a contradiction.

- $\varphi = [[\alpha]]\psi$: We assume $\alpha \in O \cup \{\varepsilon\}$; the case $\alpha \in I$ is similar. Consider some $p \overset{\alpha}{=\!\!\Rightarrow} p'$. Since $p \sqsubseteq q$, there must be some $q \overset{\alpha}{=\!\!\Rightarrow} q'$ with $p' \sqsubseteq q'$. By $q \models [[\alpha]]\psi$, we get $q' \models \psi$. By assumption, $p' \models \psi$ and thus $p \models [[\alpha]]\psi$.

- $\varphi = \langle\langle\alpha\rangle\rangle\psi$: Since $q \models \langle\langle\alpha\rangle\rangle\psi$, there is some $q \overset{\alpha}{\Longrightarrow} Q'$ (if $\alpha \in O \cup \{\varepsilon\}$, otherwise $q \xrightarrow{\alpha}\overset{\varepsilon}{\Longrightarrow} Q'$) with $\forall q' \in Q'.\, q' \models \psi$. Since $p \sqsubseteq q$, we get $p \overset{\alpha}{\Longrightarrow} P'$ with $\forall p' \in P' \exists q' \in Q'.\, p' \sqsubseteq q'$, by applying Prop. 4.5.(iii) or (ii). By assumption, we have $\forall p' \in P'.\, p' \models \psi$ and thus $p \models \langle\langle\alpha\rangle\rangle\psi$.

- $\varphi = \psi_1 \wedge \psi_2$: By definition, $q \models \varphi$ implies $q \models \psi_1$ and $q \models \psi_2$. By assumption, we get $p \models \psi_1$ and $p \models \psi_2$ and again by definition $p \models \varphi$.

- $\varphi = \psi_1 \vee \psi_2$: Similar.

Now that we have $p \sqsubseteq q \Longrightarrow \mathcal{L}_{HML}(q) \subseteq \mathcal{L}_{HML}(p)$, assume $P \sqsubseteq Q$ and $Q \models \varphi$. Then $\forall q_0 \in Q_0.\, q_0 \models \varphi$ and $\forall p_0 \in P_0 \exists q_0 \in Q_0.\, p_0 \sqsubseteq q_0$. Thus $\forall p_0 \in P_0.\, p_0 \models \varphi$ and we are done.

'$\Longleftarrow$': As a first step, we show that $\forall \varphi.\, Q \models \varphi \implies P \models \varphi$ implies $\forall p_0 \in P_0 \exists q_0 \in Q_0 \forall \psi.\, q_0 \models \psi \implies p_0 \models \psi$ by contraposition: Assume the existence of a $p_0 \in P_0$ such that for each $q_{0i} \in Q_0 = \{q_{01}, q_{02}, \ldots, q_{0n}\}$ (which is finite by our assumption of image finiteness) there is a formula ψ_i with $q_{0i} \models \psi_i$ but $p_0 \not\models \psi_i$. Then $\varphi = \psi_1 \vee \psi_2 \vee \cdots \vee \psi_n$ is such that $Q \models \varphi$ by definition of disjunction but $P \not\models \varphi$, due to p_0.

As a second step, we show that $\forall p_0 \in P_0 \exists q_0 \in Q_0 \forall \psi .\, q_0 \models \psi \implies p_0 \models \psi$ implies $\forall p_0 \in P_0 \exists q_0 \in Q_0 .\, p_0 \sqsubseteq q_0$ by showing that

$$\mathcal{R} = \{(p, q) \mid \mathcal{L}_{HML}(q) \subseteq \mathcal{L}_{HML}(p)\}$$

is a MIA-refinement relation. For this, consider a pair $(p, q) \in \mathcal{R}$ and assume towards a contradiction that one of the items of Def. 4.4 is violated; note that this implies $q \neq u_Q$. We check each item and show a contradiction:

We consider items (ii) and (iii) together and write the input case in brackets.

(i) $p = u_P$ implies $p \not\models \mathtt{tt}$, which implies $q \not\models \mathtt{tt}$ by $\mathcal{L}_{HML}(q) \subseteq \mathcal{L}_{HML}(p)$. Thus $q = u_Q$ which contradicts $q \neq u_Q$.

(ii), (iii) Assume $q \xrightarrow{\alpha} Q'$ and $\forall P' .\, p \overset{\alpha}{\Longrightarrow} P'$ ($\forall P' .\, p \xrightarrow{\alpha}\overset{\varepsilon}{\Longrightarrow} P'$ if $\alpha \in I$) implies $\exists p' {\in} P' \; \forall q' {\in} Q' .\, (p', q') \notin \mathcal{R}$. Let $\overline{P'} = \{P'_1, \ldots, P'_n\} = \{P' \mid p \overset{\alpha}{\Longrightarrow} P'\}$ ($\{P' \mid p \xrightarrow{\alpha}\overset{\varepsilon}{\Longrightarrow} P'\}$ if $\alpha \in I$); note that this set is finite by our assumption of image finiteness.

Fix a $P'_k \in \overline{P'}$. There exists $p'_k \in P'_k$ such that for all $q'_l \in Q' = \{q'_1, \ldots, q'_m\}$ there exists a HML-formula $\psi_{k,l}$ such that $p'_k \not\models \psi_{k,l}$ and $q'_l \models \psi_{k,l}$, since $(p'_k, q'_l) \notin \mathcal{R}$. Therefore, by definition of disjunction, $p'_k \not\models \psi_{k,1} \vee \cdots \vee \psi_{k,m}$, but for all $q'_l \in Q'$, $q'_l \models \psi_{k,1} \vee \cdots \vee \psi_{k,m}$.

Thus, for $\varphi = \langle\langle a \rangle\rangle\big((\psi_{1,1} \vee \cdots \vee \psi_{1,m}) \wedge \cdots \wedge (\psi_{n,1} \vee \cdots \vee \psi_{n,m})\big)$ we have $q \models \varphi$ and $p \not\models \varphi$.

(iv) Assume $p \overset{\alpha}{\dashrightarrow} p'$ and $\forall q' .\, q \overset{\alpha}{=\!\Rightarrow} q'$ implies $(p', q') \notin \mathcal{R}$. Let $\overline{Q} = \{q'_1, \ldots, q'_m\} = \{q' \mid q \overset{\alpha}{=\!\Rightarrow} q'\}$; note that this set is finite by our assumption of image finiteness.

There exists for each $q'_k \in \overline{Q}$ a HML-formula ψ_k such that $q'_k \models \psi_k$ but $p' \not\models \psi_k$, since $(p', q'_k) \notin \mathcal{R}$. Thus, for $\varphi = [[a]](\psi_1 \vee \cdots \vee \psi_m)$ we have $q \models \varphi$ and $p \not\models \varphi$. □

This characterization for MIAs is also applicable to subsets such as the MIAs with initial τ-closure or single initial states (cf. [12]). Furthermore, dMTSs and MTSs can also be seen as subsets if we consider all visible actions to be outputs; recall that dMTS and MTS only have single initial states and

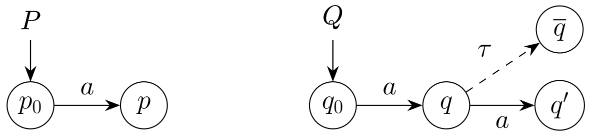

Figure 4.16: ACTL-formulae give no HML-characterization for MIA-refinement in general.

must-transitions only have single target states in MTSs. For the latter, a HML-style characterization has already been presented in [45] with a different proof.

We now return to examining MIAs. The implication of Prop. 4.60 holds for all ACTL. Furthermore, the reverse implication holds for τ-free MIAs, since then HML- and $\mathfrak{H}$-formulae coincide, as mentioned above. For MIAs with τ, however, even requiring inclusion of all Unless-free-formulae is insufficient, as we show in the following.

Proposition 4.61 (HML-characterization)**.** *For image-finite MIAs P and Q we have:*

1. *$P \sqsubseteq Q$ implies $\mathcal{L}_A(Q) \subseteq \mathcal{L}_A(P)$, hence $\mathcal{L}_{\mathfrak{H}}(Q) \subseteq \mathcal{L}_{\mathfrak{H}}(P)$,*
2. *if P and Q are τ-free, $\mathcal{L}_{\mathfrak{H}}(Q) \subseteq \mathcal{L}_{\mathfrak{H}}(P)$ implies $P \sqsubseteq Q$,*
3. *$\mathcal{L}_{\mathfrak{Wf}}(Q) \subseteq \mathcal{L}_{\mathfrak{Wf}}(P)$ does not imply $P \sqsubseteq Q$ in general.*

Proof. 1. Take φ with $Q \models \varphi$. Applying Cor. 4.52 we get $Q \sqsubseteq \varphi$. By transitivity and the same corollary, we have $P \models \varphi$.

2. Since on τ-free MIAs HML-formulae are essentially a subset of ACTL-formulae, the claim is a corollary of Prop. 4.60.

3. Consider MIAs P and Q from Fig. 4.16; clearly, $P \not\sqsubseteq Q$. We will argue, that there is no Unless-free formula φ distinguishing P and Q in the sense that $Q \models \varphi$ and $P \not\models \varphi$. Similarly to the proof of Prop. 4.57, we will show by structural induction, that for all $\varphi \in \mathfrak{Wf}$ we have $Q \not\models \varphi$ or $P \models \varphi$. Note that $p \sqsubseteq \overline{q}$ and thus $P[p] \sqsubseteq Q[q]$ implying $(*) : \forall \varphi.\, Q[q] \models \varphi \implies P[p] \models \varphi$.

For $\varphi = \mathtt{tt}$ we have $P \models \varphi$ and for $\varphi = \mathtt{ff}$ we have $Q \not\models \varphi$.

We have $P \models [B]\psi$ if $a \notin B$, since then $p_0 \overset{c}{\dashrightarrow}\!\!\!\!\!/\;$ for all $c \in B$. If $a \in B$, assume $Q \models \varphi$ (otherwise, we are done). Then $Q[q] \models \psi$ and, by $(*)$, we get $P[p] \models \psi$. Thus $P \models \varphi$ by definition of $[B]$, and we are done.

Similarly, $Q \models \langle B_\varepsilon \rangle \psi$ can only hold because of $\varepsilon \in B_\varepsilon$ or $a \in B_\varepsilon$, since q_0 has only one outgoing must-transition, which is labelled with a. In the first case, $Q \models \psi$ by $q_0 \overset{\varepsilon}{\Longrightarrow} q_0$ and $Q = Q[q_0]$. Then $P \models \psi$ by induction and $P \models \langle B_\varepsilon \rangle \psi$ because of $\varepsilon \in B_\varepsilon$. In the second case, $Q \models \langle B_\varepsilon \rangle \psi$ holds because of $Q[q] \models \psi$, which implies $P[p] \models \psi$ by $(*)$. Now, $P \models \varphi$ due to $a \in B_\varepsilon$ and we are done.

Finally, it remains to consider disjunctions and conjunctions of formulae we already considered. By applying the distributivity laws of Prop. 4.54, any combination can be written as a disjunction φ of conjunctions. We can again assume $Q \models \varphi$ and, by the definition of $\vee$, there is a disjunct ψ of φ with $Q[q_0] \models \psi$. Since $Q = Q[q_0]$, we have $Q \models \psi$. By definition of $\wedge$, this implies that $Q \models \theta$ for all conjuncts θ of ψ. Applying the induction hypothesis to each θ, we get that $P \models \theta$ for all conjuncts θ of ψ. Thus $P \models \varphi$ and also $P \models \psi$, since it has only one initial state. □

4.6 Aspect-Oriented Specification – Alphabet Extension

We now extend the definition of refinement to allow for alphabet extension by using an alphabet extension operator. We also present a direct, simulation-based characterization, which can be seen as an alternative definition. While the definition based on alphabet extension is more useful for proofs, the direct version can be more efficient and is more space-saving when implemented. This is particularly true for large differences in alphabets of P and Q.

Definition 4.62 (ν-Alphabet Extension Operator and ν-Refinement)**.** *Given a MIA $(P, I, O, \longrightarrow, \dashrightarrow, P_0, u)$ and disjoint action sets I' and O' satisfying $I' \cap \mathcal{A} = \emptyset = O' \cap \mathcal{A}$, the* alphabet extension of P by I' and O' *is given by $[P]_{I',O'} =_{df} (P, I \dot\cup I', O \dot\cup O', \longrightarrow, \dashrightarrow', P_0, u)$ with $\dashrightarrow' =_{df} \dashrightarrow \cup \{(p, i, p') \mid i \in I', p \overset{\nu_I}{\dashrightarrow} p'\} \cup \{(p, o, p') \mid o \in O', p \overset{\nu_O}{\dashrightarrow} p'\}$. We often write $[p]_{I',O'}$ for p as state of $[P]_{I',O'}$, and conveniently use $[p]$ in case I', O' are understood from the context. We also abbreviate $[p]_{I_Q \setminus I_P, O_Q \setminus O_P}$ by $[p]_Q$; the same notation is used for P in place of p.*

Given MIAs P and Q with $I_P \supseteq I_Q$ and $O_P \supseteq O_Q$, we write $p \sqsubseteq_\nu q$ and say that p ν-refines *q (or that p matches q) if $p \sqsubseteq [q]_P$; We let $p \sqsupseteq\sqsubseteq_\nu q$ stand for $p \sqsubseteq_\nu q$ and $q \sqsubseteq_\nu p$ and call it* ν-equivalence. *The same notations are used for MIAs instead of states.*

It is quite obvious that $\sqsubseteq_\nu$ extends $\sqsubseteq$. It is also easy to see that it behaves according to out intuition: foreign actions are allowed exactly where specified by ν_I and ν_O transitions.

Definition 4.63 (Simulation based ν-Refinement)**.** *Let P, Q be MIAs with $I_P \supseteq I_Q$ and $O_P \supseteq O_Q$. A relation $\mathcal{R} \subseteq P \times Q$ is a* ν-refinement relation *if, for all $(p, q) \in \mathcal{R}$ with $q \neq u_Q$, the conditions (i)-(iii) of Def. 4.4 hold and:*

(iv) $p \overset{\alpha}{\dashrightarrow} p'$ *and* $\alpha \in \mathcal{A}_Q \cup \{\tau, \nu_I, \nu_O\}$ *implies*
$\exists q'.\, q \overset{\hat{\alpha}}{\Longrightarrow} q'$ *and* $(p', q') \in \mathcal{R}$.

(v) $p \overset{\alpha}{\dashrightarrow} p'$ *and* $\alpha \notin \mathcal{A}_Q \cup \{\tau, \nu_I, \nu_O\}$ *implies*
$\exists q'.\, (p', q') \in \mathcal{R}$ *and either* $\alpha \in I_P$ *and* $q \overset{\nu_I}{\Longrightarrow} q'$ *or* $\alpha \in O_P$ *and* $q \overset{\nu_O}{\Longrightarrow} q'$.

We write $p \sqsubseteq'_\nu q$ if there exists a ν-refinement relation $\mathcal{R}$ such that $(p, q) \in \mathcal{R}$. Furthermore, we define $P \sqsubseteq'_\nu Q$ if $\forall p_0 \in P_0 \exists q_0 \in Q_0.\, p_0 \sqsubseteq'_\nu q_0$.

Theorem 4.64 (Characterization of ν-Refinement)**.** *For MIAs P and Q with $I_P \supseteq I_Q$ and $O_P \supseteq O_Q$ we have $P \sqsubseteq_\nu Q$ if and only if $P \sqsubseteq'_\nu Q$.*

Proof. We show that MIA-refinement relations for P and $[Q]_P$ and ν-refinement relations for P and Q coincide. Since conditions (i) – (iii) for these relations coincide and alphabet extension only concerns may-transitions, we only have to compare the remaining conditions. Since alphabet extension only concerns foreign actions, Cond. (iv) in Def. 4.63 corresponds to the subcase of (iv) in Def. 4.4 for $\alpha \in \mathcal{A}_Q \cup \{\tau, \nu_I, \nu_O\}$. To see that Cond. (v) corresponds to the remaining subcase, observe that $q'' \overset{\alpha}{\dashrightarrow} q'''$ with e.g. $\alpha \in I_P \setminus \mathcal{A}_Q$ exists in $[Q]_P$ if and only if $q'' \overset{\nu_I}{\dashrightarrow} q'''$ does in Q.

□

4.6.1 Extending Logical Operators

Disjunction works for ν-refinement with the same definition, results and proofs as presented previously in Sect. 4.4. We proceed to lift our conjunction operator to conjuncts with dissimilar alphabets.

Definition 4.65 (Lifting Conjunction by alphabet extension)**.** *Let P, Q be MIAs, $p \in P$ and $q \in Q$ such that $I_P \cap O_Q = \emptyset = I_Q \cap O_P$. Then, $p \wedge_\nu q =_{df} [p]_Q \wedge [q]_P$ and similarly for $P \wedge_\nu Q$.*

It is obvious that this definition coincides with the previous definition of $\wedge$ on MIAs with common alphabets. Therefore we will write & for $\&_\nu$ and $\wedge$ for $\wedge_\nu$.

To be able to lift our main result, Thm. 4.37, it is sufficient to establish that the alphabet extension operation is a homomorphism for conjunction. The proof of Thm. 4.67 follows exactly the line of argument in [53].

Lemma 4.66. *Let P with $p \in P$ and Q with $q \in Q$ be MIAs with common alphabets. Consider the alphabet extensions by some I' and O'. Then:*

(a) *p and q are consistent iff $[p]$ and $[q]$ are.*

(b) *Given consistency, $[P \wedge Q] \sqsupseteq\sqsubseteq [P] \wedge [Q]$ and thus $[P \wedge Q] \sqsupseteq\sqsubseteq_\nu [P] \wedge [Q]$.*

Proof. For proving (a), consider the mapping $\beta : (p, q) \mapsto ([p], [q])$, which is a bijection between $P\&Q$ and $[P]\&[Q]$. We have $(p, q) \in F_{P\&Q}$ due to $a \in \mathcal{A}$ and (F1) or (F2) iff $([p], [q]) \in F_{[P]\&[Q]}$ due to $a \in \mathcal{A}$ and (F1) or (F2). Observe that (F1) and (F2) never apply to $([p], [q])$ and $a \in I' \cup O'$, since there are no must-transitions labelled a. For the same reason, Rules (Must1), (UMust1) and their symmetric counterparts are never applicable for a and, thus, β is an isomorphism regarding must-transitions; hence, (F3) is applicable exactly in the corresponding cases according to β. Therefore, β is also a bijection between $F_{P\&Q}$ and $F_{[P]\&[Q]}$.

For (b), we can regard β also as a bijection between $[P \wedge Q]$ and $[P] \wedge [Q]$. We show that β and β^{-1} are ν-refinement relations. For this, note that β relates initial states exactly to initial states. We consider a pair $([p \wedge q], [p] \wedge [q]) \in \beta$ and check the conditions of Def. 4.63: Cond. (i) is trivial. Conds. (ii) and (iii) are clear, because β is still an isomorphism on must-transitions. The same holds for β^{-1}, and we only have to check (iv).

For the following observe that (νMay)-rules are not applicable in the construction of either conjunction. We let ν stand for ν_I in case $\alpha \in I'$ and for ν_O in case $\alpha \in O'$.

- *"$\sqsubseteq$":* Regarding Cond. (iv), we only have to consider $\alpha \in I' \cup O'$ and $[p \wedge q] \overset{\alpha}{\dashrightarrow} [p' \wedge q']$, since the other transitions are not modified by alphabet extension. This transition exists because of $p \wedge q \overset{\nu}{\dashrightarrow} p' \wedge q'$, which exists due to Rule (May1) or (UMay1) w.l.o.g. In the first case, we have $p \overset{\nu}{\Longrightarrow} p'$ and $q \overset{\nu}{\Longrightarrow} q'$. This implies $[p] \overset{\alpha}{\Longrightarrow} [p']$ and $[q] \overset{\alpha}{\Longrightarrow} [q']$. In the second case, $p \overset{\nu}{\Longrightarrow} p'$ and $q = u_Q = q'$; the former statement implying $[p] \overset{\alpha}{\Longrightarrow} [p']$. Either way, we get $[p] \wedge [q] \overset{\alpha}{\dashrightarrow} [p'] \wedge [q']$ by Rule (May1) or (UMay1) respectively. This matches $[p \wedge q] \overset{\alpha}{\dashrightarrow} [p' \wedge q']$ as we needed to show.

- *"$\sqsupseteq$":* To establish Cond. (iv) consider $[p] \wedge [q] \overset{\alpha}{\dashrightarrow} [p'] \wedge [q']$ with $\alpha \in I' \cup O'$. If this transition exists due to (May1), we have $p \overset{\nu}{\Longrightarrow} p'$ and $q \overset{\nu}{\Longrightarrow} q'$. We get $p \wedge q \overset{\nu}{\dashrightarrow} p' \wedge q'$ due to (May1) implying $[p \wedge q] \overset{\alpha}{\dashrightarrow} [p' \wedge q']$. If the transition is due to (UMay1), we have $q = u_Q = q'$ and $[p] \overset{\alpha}{\Longrightarrow} [p']$. Following the same line of argumentation as above, we get $p \overset{\nu}{\Longrightarrow} p'$, then $p \wedge q \overset{\nu}{\dashrightarrow} p' \wedge q'$ (by (UMay1)), and finally $[p \wedge q] \overset{\alpha}{\dashrightarrow} [p' \wedge q']$. □

Having proven the lemma in out new setting, the characteristic property of conjunction follows as in [13]. We present the proof for completeness.

Theorem 4.67 ($\wedge$ is And). *Let P, Q and R be MIAs such that $I_P \cap O_Q = \emptyset = I_Q \cap O_P$, $I_R \supseteq I_P \cup I_Q$ and $O_R \supseteq O_P \cup O_Q$. Then, $R \sqsubseteq_\nu P$ and $R \sqsubseteq_\nu Q$ iff $R \sqsubseteq_\nu P \wedge Q$.*

Proof. Recall that we denote by $[\cdot]_P$ an extension with the additional actions of P, and similarly for Q and R. Also note that, in the context of this theorem, $[[P]_Q]_R = [P]_R$ and $[[Q]_P]_R = [Q]_R$. Furthermore, note that Thm. 4.37 is applicable to MIAs with common alphabets.

We reason as follows:

$$\begin{array}{lll}
 & R \sqsubseteq_\nu P \text{ and } R \sqsubseteq_\nu Q & \\
\text{iff} & R \sqsubseteq_\nu [P]_R \text{ and } R \sqsubseteq_\nu [Q]_R & \text{(by Thm. 4.64)} \\
\text{iff} & R \sqsubseteq_\nu [P]_R \wedge [Q]_R & \text{(by Thm. 4.37)} \\
\text{iff} & R \sqsubseteq_\nu [[P]_Q \wedge [Q]_P]_R & \text{(by Lem. 4.66 and note above)} \\
\text{iff} & R \sqsubseteq_\nu P \wedge Q & \text{(by Defs. 4.62 and 4.65)}
\end{array}$$ □

Alternatively, the conjunction can be constructed directly, instead of first constructing the alphabet extensions of the conjuncts.

Definition 4.68 (Conjunction for ν-refinement). *Given MIAs $(P, I_P, O_P, \longrightarrow_P, \dashrightarrow_P, P_0, u_P)$ and $(Q, I_Q, O_Q, \longrightarrow_Q, \dashrightarrow_Q, Q_0, u_Q)$. The* ν-conjunctive product *is defined as $P \&_\nu Q =_{df} (P \times Q, I_P \cup I_Q, O_P \cup O_Q, \longrightarrow, \dashrightarrow, P_0 \times Q_0), (u_P, u_Q))$ with the following operational transition rules and their symmetric counterparts:*

(Must1) $(p,q) \xrightarrow{\alpha} \{(p',q') \,|\, p' \in P',\ q \overset{\hat{\alpha}}{\Longrightarrow}_Q q'\}$ *if* $p \xrightarrow{\alpha}_P P',\ q \overset{\hat{\alpha}}{\Longrightarrow}_Q$
for $\alpha \in (\mathcal{A}_P \cap \mathcal{A}_Q) \cup \{\tau, \nu_I, \nu_O\}$

(νMust1) $(p,q) \xrightarrow{\alpha} \{(p',q') \,|\, p' \in P',\ q \overset{\alpha}{\Longrightarrow}_Q q'\}$
if $p \xrightarrow{\alpha}_P P'$ *and* $q \overset{\nu_I}{\Longrightarrow}_Q q'$ *for* $\alpha \in I_P \setminus I_Q$
or if $p \xrightarrow{\alpha}_P P'$ *and* $q \overset{\nu_O}{\Longrightarrow}_Q q'$ *for* $\alpha \in O_P \setminus O_Q$

(UMust1) $(p, u_Q) \xrightarrow{\alpha} P' \times \{u_Q\}$ *if* $p \xrightarrow{\alpha}_P P'$

(May1) $(p,q) \overset{\alpha}{\dashrightarrow} (p',q')$ *if* $p \overset{\alpha}{\Longrightarrow}_P p'$ *and* $q \overset{\hat{\alpha}}{\Longrightarrow}_Q q'$
for $\alpha \in (\mathcal{A}_P \cap \mathcal{A}_Q) \cup \{\tau, \nu_I, \nu_O\}$

(νMay1) $(p,q) \overset{\alpha}{\dashrightarrow} (p',q')$ *if* $p \overset{\alpha}{\Longrightarrow}_P p'$ *and* $q \overset{\nu_I}{\Longrightarrow}_Q q'$ *for* $\alpha \in I_P \setminus I_Q$
or if $p \overset{\alpha}{\Longrightarrow}_P p'$ *and* $q \overset{\nu_O}{\Longrightarrow}_Q q'$ *for* $\alpha \in O_P \setminus O_Q$

(UMay1) $(p, u_Q) \overset{\alpha}{\dashrightarrow} (p', u_Q)$ *if* $p \overset{\alpha}{\dashrightarrow}_P p'$

From this conjunctive product $P \&_\nu Q$ we get the conjunction $P \wedge_\nu Q$ by identifying and deleting inconsistent states as described in Def. 4.34.

It is easy to see that this definition coincides with the previous one using alphabet extension, since (νMay1) treats actions foreign to P as (May) does after applying alphabet extension. The same is true for (νMust) and (Must). Again the direct construction requires less space when implemented and is thus more efficient.

4.6.2 Extending Structural Operators

It is easy to show that compositionality of parallel composition as in Thm. 4.15 also holds for the extended refinement relation as long as alphabet extension does not yield new communications. As has already been demonstrated in [13], these might result in an error and, therefore, must be disallowed. Technically, if $a \in (\mathcal{A}_1 \setminus \mathcal{A}_Q) \cap \mathcal{A}_2$, then $P_1 \parallel P_2$ might have a new error if P_1 performs $a \in O_1$ or cannot perform $a \in I_1$.

Theorem 4.69 (Compositionality of Parallel Composition)**.** *Let P_1, P_2, Q be MIAs such that Q and P_2 are composable and $P_1 \sqsubseteq_\nu Q$. Assume further that, for $I' =_{df} I_1 \setminus I_Q$ and $O' =_{df} O_1 \setminus O_Q$, we have $(I' \cup O') \cap \mathcal{A}_2 = \emptyset$. Then:*

1. *P_1 and P_2 are composable.*
2. *$P_1 \parallel P_2 \sqsubseteq_\nu Q \parallel P_2$. Thus P_1 and P_2 are compatible if Q and P_2 are.*

Proof. It is easy to see that the MIAs $[Q]_{I',O'}$ and P_2 are composable due to $(I' \cup O') \cap \mathcal{A}_2 = \emptyset$. Furthermore, $[Q]_{I',O'} \otimes P_2$ is isomorphic to $[Q \otimes P_2]_{I',O'}$ via mapping $[q] \otimes p_2 \mapsto [q \otimes p_2]$. This is because of (PMay1) in the definition of $\otimes$, since we only add "fresh" may-transitions to each $q \in Q$. The mapping also respects errors: new may-transitions with label $o \in O'$ cannot create new errors since $o \notin I_2$, and no new $i \in I'$ has to have a must-transition since $i \notin O_2$. Thus, for each $q_0 \in Q_0$ and $p_{02} \in P_{02}$ that are compatible, $[q_0]$ and p_{02} are compatible as well; moreover, $p_{01} \sqsubseteq_\nu [q_0]$. Now, the result follows from Thm. 4.15. The last statement follows since a universal state only refines a universal state. □

It is also easy to see that the generalized $\sqsubseteq_\nu$ is a precongruence for hiding and restriction as well. Similarly to parallel composition, where no new communications may arise, we have to require that L does not hide actions that are added during refinement. Otherwise, new τ-transitions appear parallel to ν_O-transitions which may be impossible to match by Q.

Proposition 4.70. *Let P and Q be MIAs such that $P \sqsubseteq_\nu Q$ and $L \subseteq \Sigma$.*

1. *If Q/L is defined and $L \cap (I_P \setminus I_Q) = \emptyset$, then P/L is defined. If in addition $L \cap (O_P \setminus O_Q) = \emptyset$, then $P/L \sqsubseteq_\nu Q/L$.*

2. *If $Q/\!\!/L$ is defined and $L \cap (I_P \setminus I_Q) = \emptyset$, then $P/\!\!/L$ is defined. If in addition $L \cap (O_P \setminus O_Q) = \emptyset$, then $P/\!\!/L$ is defined and $P/\!\!/L \sqsubseteq_\nu Q/\!\!/L$.*

3. *If $Q \setminus L$ is defined and $L \cap (O_P \setminus O_Q) = \emptyset$, then $P \setminus L$ is defined and $P \setminus L \sqsubseteq_\nu Q \setminus L$.*

Proof. 1. Q/L being defined implies $L \cap I_Q = \emptyset$. With $L \cap (I_P \setminus I_Q)$ we have $L \cap I_P = \emptyset$, thus P/L is defined.

We have $P \sqsubseteq [Q]_P$ and, due to Prop. 4.24, $P/L \sqsubseteq [Q]_P/L$. The latter, $[Q]_P/L$, is identical to $[Q/L]_{P/L}$, since hiding does not affect transitions added by the alphabet extension operator due to $L \cap (O_P \setminus O_Q) = \emptyset$ and since $O_{P/L} \cap L = \emptyset$ by construction, which ensures that labels in L are not reintroduced.

2. Can be shown analogously. Additionally, observe: if a foreign input $i \in I_P$ in $[Q]_P$ is cut during $/\!\!/L$, then the parallel ν_I-transition is cut as well. The latter is also cut when computing $Q/\!\!/L$, so a parallel i-transition to u is introduced for $[Q/\!\!/L]_{P/\!\!/L}$.

3. That $P \setminus L$ is defined can be shown analogously to the above.

We have $P \sqsubseteq [Q]_P$ and, due to Prop. 4.24, $P \setminus L \sqsubseteq [Q]_P \setminus L$. The latter, $[Q]_P \setminus L$, is identical to $[Q \setminus L]_{P \setminus L}$: input transitions added by $[\cdot]_P$ and removed by $\setminus L$ in the first MIA are not added by $[\cdot]_{P \setminus L}$ in the second. □

4.6.3 Extending Temporal Operators

Finally, we extend the temporal side of our framework to also support alphabet extension. Recall that similarly to MIAs, formulae are considered to have alphabets, even though this did not play a role in Sect. 4.5 and was largely omitted. We require a quite intuitive consistency of notation: Labels appearing in parameters of φ should be contained in its alphabets; each parameter B has to be partitioned into inputs and outputs, i.e. $B = B_I/B_O$; no label appears as input and as output in any parameter or MIA that is an argument of the formula. Syntactically, the formula remain the same. The extension of the operators is quite straightforward and intuitive. The constructions remain the same, only the proofs have to deal with the ν-cases. Therefore, we will only show the extension for the universal next-operator and conjunction. The other operators are extended analogously.

Definition 4.71 (Extended ACTL). *Let alphabets B and B_ε be partitioned into inputs and outputs B_I/B_O – B_ε may additionally contain ε. Let C be partitioned analogously into C_I/C_O. The syntax of ACTL-formulae is* *the same as in Def. 4.41, but with the following additional requirements for alphabets: If φ has alphabets I/O and φ has alphabets I'/O' then*

- *$[B]\psi$ and $\langle B_\varepsilon\rangle\psi$ have alphabets $(I \cup B_I)/(O \cup B_O)$.*
- *$\varphi \wedge \psi$ and $\varphi \vee \psi$ have alphabets $(I \cup I')/(O \cup O')$.*
- *$\varphi_B \mathsf{W}_C \psi$ has alphabets $(I \cup I' \cup B_I \cup C_I)/(O \cup O' \cup B_O \cup C_O)$.*

The semantics of ACTL-formulae remains the same as in Def. 4.41.

The constructions presented in Sect. 4.5 remain unchanged, as they already deal with ν_I and ν_O in the desired manner. Intuitively, the constructions for temporal operators deal with actions contained in their parameter B and C in a certain way and with all other actions in some 'default'way. Since new actions cannot be part of the parameters, treating ν_I and ν_O in the default manner yields the desired result.

It should be noted that this definition does not allow ν_I and ν_O in the parameters of the temporal operators. This could be extended such that ν_I and ν_O would again be treated like may transitions. They could be allowed in the parameter B of $[B]$, $\Box_B$ and ${}_B\mathsf{W}_C$ and even in the parameter C of the latter. The constructions would have to be adjusted slightly to allow for this. However, it is unreasonable to allow these labels in $\langle B_\varepsilon\rangle$. Technically, the construction would no longer work, as ν-labels are not allowed on must-transitions which would be part of the construction. More intuitively, it would be strange and counter-intuitive for a specification to require a system to perform an action that is not included in the specification itself.

With these extended operators the analog of Thm. 4.51 hold for refinement with alphabet extension. Again, we formulate the claim and show the proofs only for $[B]P$ and $P \wedge Q$. The other cases are analogous. The compatibility and precongruence results of Cor. 4.52 and Thm. 4.53 follow, as before.

Theorem 4.72 (Extended Temporal MIA operators vs. formulae)**.** *For non-universal MIAs P and Q and for sets $B \subseteq \mathcal{A}$ and $B_\varepsilon \subseteq \mathcal{A} \cup \{\varepsilon\}$, we have*

3) $P \sqsubseteq_\nu [B]Q$ iff $\forall p_0 \in P_0\, \forall a \in B\, \forall p'.\ (p_0 \overset{a}{\dashrightarrow} p'$ implies $P[p'] \sqsubseteq_\nu Q)$
5) $P \sqsubseteq_\nu Q \wedge R$ iff $P \sqsubseteq_\nu Q$ and $P \sqsubseteq_\nu R$

The other operators are extended analogously from Thm. 4.51.

Proof. The case $\wedge$ has been proven in Thm. 4.67.

Case $[B]Q$:
'$\Rightarrow$': as in the proof of Thm. 4.51.

'⇐': Again, we only have to argue for the first subset. Since x_0 has no must-transitions, we only have to consider the may-transitions of some $p_0 \in P_0$. Each o- and i-transition (with $o \notin B$ and $i \notin B$ respectively including $i, o \notin \mathcal{A}_{[B]Q}$) of p_0 is matched by $x_0 \overset{o}{\dashrightarrow} \mathtt{tt}_0$ or $x_0 \overset{i}{\dashrightarrow} u$ respectively. The resulting pairs are in $\mathcal{R}_{\mathtt{tt}}$. The remainder can again be shown, as in the proof of Thm. 4.51. □

4.6.4 Examples

In this section, we will present a meaningful example that illustrates the flexibility of our approach. But first, we will have a quick look at the design patterns persistent, knockout and forbidding alphabet extension introduced in Sect. 4.2.2, adding a technical, but important detail for the knockout case.

The implementation of persistent alphabet extension is straightforward: each state that should behave persistently is equipped with ν_I and ν_O may-transition-loops. For this the alphabet extension operator produces the same results as the one in [13], which added may loops for all foreign actions to all states. Forbidding alphabet extension is similarly easy; such a states has no outgoing ν_I or ν_O transitions. It is easy to see that these two constructions produce the desired results.

The knockout pattern is more subtle. After performing a foreign action, a refinement should be allowed to behave arbitrarily. However, a state equipped with may-loops for all actions in $\mathcal{A} \cup \{\nu_I, \nu_O\}$ as in Fig. 4.6 is not sufficient, since it cannot be refined by u. We also cannot use the universal state instead, since outputs must not lead to u. The solution is to have a ν_I transition to u and a ν_O transition to a state $\mathtt{tt}$ as shown in Fig. 4.17: it is equipped with a may loop for each output, foreign or not, and with a may-transition to u for each input. This generalises the construction of true in [20] by adding ν_I and ν_O transitions.

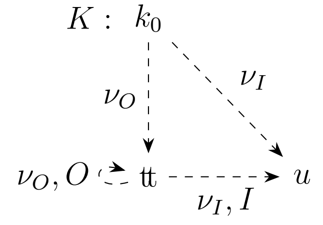

Figure 4.17: KO alphabet extension explicitly translated.

Now we come to the announced meaningful example, which does not fit into the above design patterns. In this example separately specified subrou-

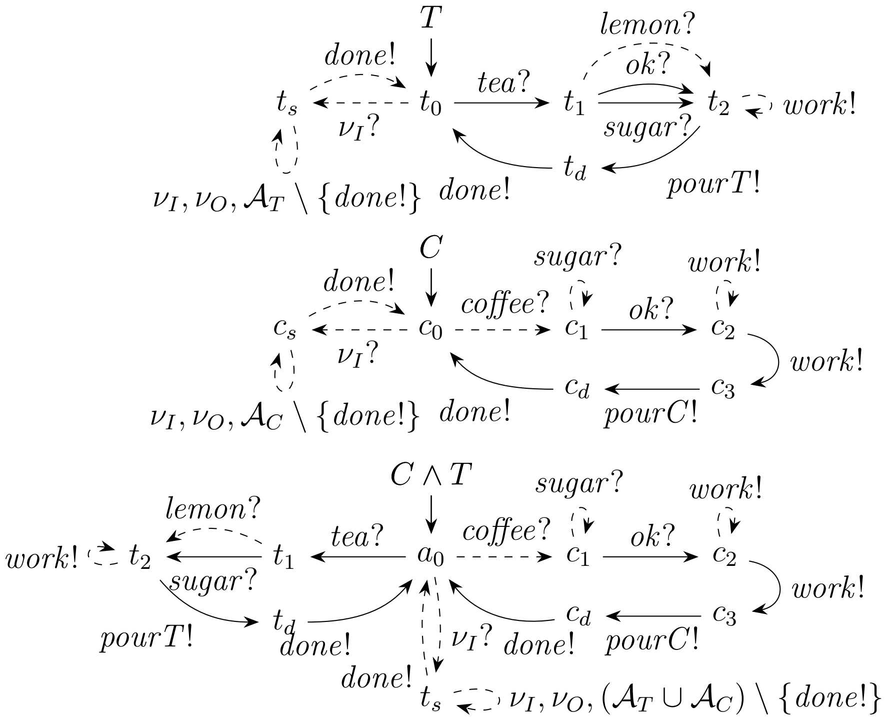

Figure 4.18: Drink dispenser.

tines are combined in a repeating choice. Concretely, we give two aspect specifications of a machine that dispenses several beverages, tea and coffee for now; others can easily be added. T and C in Fig. 4.18 specify how to handle the requests for these two beverages.

T specifies that a request for tea has to be accepted. Afterwards, sugar and possibly lemon has to be offered, and the offer can be refused by *ok* etc. After pouring the tea, the machine has to return to its initial state with *done*. C describes that, similarly, one can possibly request coffee and specifies the subsequent behaviour. Note that the procedures use common actions like *sugar*, and separate actions like *lemon*. Each of these specifications, say T, can initially be suspended by foreign inputs. Intuitively, T makes no requirements then, while the behaviour in progress is specified elsewhere. When the latter finishes with *done*, T again requires that a request for tea has to be accepted. The conjunction of T and C also presented in Fig. 4.18 shows that the combination of the aspect specifications works as intended. An essential point is that the tea-aspect can be devized without knowing which other beverages will be considered now or in the future, and it is no problem to combine any number of such aspect specifications.

This cannot be achieved by designing states as persistent, knockout and forbidding. Designating the initial state t_0 as persistent would lead to *tea* necessarily being enabled after *coffee* in the conjunction – something not possible for usual drink dispensers. If t_0 were specified as knockout, *coffee* would cancel T completely: after the first choice of coffee, all further choices would automatically be the same. With t_0 being forbidding, *coffee* would be disallowed when tea is on offer.

4.7 Conclusion

We have presented the interface theory of Modal Interface Automata, a framework that allows for heterogeneous specification of parallel processes. We defined structural, logical and temporal operators in a coherent way so that they can be combined freely.

We resolved the conflict between unspecified inputs being allowed in interface theories derived from de Alfaro and Henzinger's Interface Automata [27] but forbidden in Modal Transition Systems [45] by introducing a special universal state. This also allowed us to achieve compositionality and associativity for our parallel composition operator.

We presented a variant of ACTL for MIA and have shown how to apply the logical operators to MIA in a suitable way. We illustrated the utility of these temporal operators with an example and showed that sets of actions increase expressiveness of our logic in a meaningful way. Furthermore, we presented a list of desirable, and often expected, laws involving both propositional and temporal operators. For our interpretation of ACTL on MIAs, we proved that the Unless-free fragment is not suitable to characterise refinement. We also had a look at a mainly state-based interpretation of HML, which is standard except for the difference between inputs and outputs. This setting allows to characterise refinement by inclusion of satisfied formulae – also for dMTS – but it is not suitable to support heterogeneous specification, as we have shown.

We improved on the previous version of MIA [13] by relaxing the refinement relation and introduced a new, transition-based alphabet extension, which provides a high degree of flexibility while still allowing for aspect based specification. In particular the concepts of persistent, knockout and forbidding alphabet extension can easily be modelled, but the framework also allows for custom built patterns. It also allows for alphabet extension of ACTL-formulae. We demonstrated the utility of a custom pattern with an example of a repeated choice between separately specified subroutines.

Future work may include a more fundamental examination of MIA and its possible refinement relations as done for dMTS and EIO in the previous chapters. Furthermore, there are plans on implementing tool support for MIA.

Bibliography

[1] Luis M. Alonso and Ricardo Peña. Acceptance automata: A framework for specifying and verifying TCSP parallel systems. In *PARLE 1991*, volume 506 of *LNCS*, pages 75–91. Springer 506, 1991.

[2] Rajeev Alur, Thomas A. Henzinger, Orna Kupferman, and Moshe Y. Vardi. Alternating refinement relations. In *CONCUR 1998*, LNCS 1466, pages 163–178. Springer, 1998.

[3] Sebastian S. Bauer, Alexandre David, Rolf Hennicker, Kim Guldstrand Larsen, Axel Legay, Ulrik Nyman, and Andrzej Wasowski. Moving from specifications to contracts in component-based design. In *FASE 2012*, LNCS 7212, pages 43–58. Springer, 2012.

[4] Sebastian S. Bauer, Philip Mayer, Andreas Schroeder, and Rolf Hennicker. On weak modal compatibility, refinement, and the MIO workbench. In *TACAS 2010*, LNCS 6015, pages 175–189. Springer, 2010.

[5] Bernd Baumgarten. On internal and external characterizations of PT-net building block behaviors. In *Advances in Petri Nets 1988*, LNCS 340, pages 44–61. Springer, 1987.

[6] Nikola Benes, Benoît Delahaye, Uli Fahrenberg, Jan Kretínský, and Axel Legay. Hennessy-Milner logic with greatest fixed points as a complete behavioural specification theory. In *CONCUR 2013*, LNCS 8052, pages 76–90. Springer, 2013.

[7] Nikola Beneš, Ivana Černá, and Jan Křetínský. Modal transition systems: Composition and LTL model checking. In *ATVA 2011*, LNCS 6996, pages 228–242. Springer, 2011.

[8] Jan A. Bergstra, Jan W. Klop, and Ernst-Rüdiger Olderog. Failures without chaos: a new process semantics for fair abstraction. In *Formal Description of Programming Concepts III*, pages 77–103. North-Holland, 1987.

[9] Dirk Beyer, Arindam Chakrabarti, Thomas A. Henzinger, and Sanjit A. Seshia. An application of web-service interfaces. In *ICWS 2007*, pages 831–838. IEEE Computer Society, 2007.

[10] Gérard Boudol and Kim Guldstrand Larsen. Graphical versus logical specifications. *Theor. Comput. Sci.*, 106(1):3–20, 1992.

[11] Stephen D. Brookes, C. A. R. Hoare, and Andrew W. Roscoe. A theory of communicating sequential processes. *J. ACM*, 31(3):560–599, 1984.

[12] Ferenc Bujtor, Sascha Fendrich, Gerald Lüttgen, and Walter Vogler. Nondeterministic modal interfaces. In *SOFSEM 2015*, LNCS 8939, pages 152–163. Springer, 2015.

[13] Ferenc Bujtor, Sascha Fendrich, Gerald Lüttgen, and Walter Vogler. Nondeterministic modal interfaces. *Theor. Comput. Sci.*, 642:24–53, 2016.

[14] Ferenc Bujtor, Lev Sorokin, and Walter Vogler. Testing preorders for dMTS: Deadlock- and the new deadlock/divergence-testing. In *ACSD 2015*, pages 60–69, 2015.

[15] Ferenc Bujtor, Lev Sorokin, and Walter Vogler. Testing preorders for dMTS: Deadlock- and the new deadlock/divergence-testing. *ACM Trans. Embed. Comput. Syst.*, 16(2):41:1–41:28, December 2016.

[16] Ferenc Bujtor and Walter Vogler. Error-pruning in interface automata. In *SOFSEM 2014*, LNCS 8327, pages 162–173. Springer, 2014.

[17] Ferenc Bujtor and Walter Vogler. Failure semantics for modal transition systems. In *ACSD 2014*, pages 42–51. IEEE, 2014.

[18] Ferenc Bujtor and Walter Vogler. Error-pruning in interface automata. *Theor. Comput. Sci.*, 597:18–39, 2015.

[19] Ferenc Bujtor and Walter Vogler. Failure semantics for modal transition systems. *ACM Trans. Embedded Comput. Syst.*, 14(4):67:1–67:30, 2015.

[20] Ferenc Bujtor and Walter Vogler. ACTL for modal interface automata. In *ACSD 2016*, pages 1–10, 2016.

[21] Ferenc Bujtor and Walter Vogler. ACTL for modal interface automata. *Theor. Comput. Sci.*, 693:13–34, 2017.

[22] Taolue Chen, Chris Chilton, Bengt Jonsson, and Marta Z. Kwiatkowska. A compositional specification theory for component behaviours. In *ESOP 2012*, LNCS 7211, pages 148–168. Springer, 2012.

[23] Chris Chilton, Bengt Jonsson, and Marta Kwiatkowska. An algebraic theory of interface automata. Technical Report RR-13-02, DCS, 2013.

[24] Chris J. Chilton. *An Algebraic Theory of Componentised Interaction*. PhD thesis, Department of Computer Science, University of Oxford, UK, 2013.

[25] Luca de Alfaro and Thomas A. Henzinger. Interface automata. In *FSE 2001*, pages 109–120. ACM, 2001.

[26] Luca de Alfaro and Thomas A. Henzinger. Interface theories for component-based design. In *EMSOFT 2001*, LNCS 2211, pages 148–165. Springer, 2001.

[27] Luca de Alfaro and Thomas A. Henzinger. Interface-based design. In *Engineering Theories of Software-Intensive Systems*, NATO Science Series, pages 83–104. Springer 195, 2005.

[28] Rocco De Nicola. Extensional equivalences for transition systems. *Acta Inf.*, 24(2):211–237, 1987.

[29] Rocco De Nicola, Alessandro Fantechi, Stefania Gnesi, and Gioia Ristori. An action-based framework for verifying logical and behavioural properties of concurrent systems. *Computer Networks and ISDN Systems*, 25(7):761–778, 1993.

[30] Rocco De Nicola and Matthew Hennessy. Testing equivalences for processes. *Theor. Comput. Sci.*, 34:83–133, 1984.

[31] Rocco De Nicola and Roberto Segala. A process algebraic view of input/output automata. *Theor. Comput. Sci.*, 138(2):391–423, 1995.

[32] David L. Dill. *Trace theory for automatic hierarchical verification of speed-independent circuits.* ACM distinguished dissertations. MIT Press, 1989.

[33] Harald Fecher and Heiko Schmidt. Comparing disjunctive modal transition systems with an one-selecting variant. *J. Log. Algebr. Program.*, 77(1-2):20–39, 2008.

[34] Dario Fischbein, Víctor A. Braberman, and Sebastián Uchitel. A sound observational semantics for modal transition systems. In *ICTAC 2009*, LNCS 5684, pages 215–230. Springer, 2009.

[35] Dario Fischbein and Sebastián Uchitel. On correct and complete strong merging of partial behaviour models. In *SIGSOFT FSE*, pages 297–307, 2008.

[36] Mihaela Gheorghiu, Dimitra Giannakopoulou, and Corina S. Pasareanu. Refining interface alphabets for compositional verification. In *TACAS 2007*, LNCS 4424, pages 292–307. Springer, 2007.

[37] Matthew Hennessy. *Algebraic theory of processes*. MIT Press, 1988.

[38] Matthew Hennessy and Robin Milner. Algebraic laws for nondeterminism and concurrency. *J. ACM*, 32(1):137–161, 1985.

[39] C. A. R. Hoare. *Communicating Sequential Processes*. Prentice-Hall, 1985.

[40] Hans Hüttel and Kim Guldstrand Larsen. The use of static constructs in a modal process logic. In *Logic at Botik*, LNCS 363, pages 163–180. Springer, 1989.

[41] C. Kühbacher. Implizite und verbotene Eingaben bei modalen Interface-Automaten. Bachelor's thesis, Univ. Augsburg, Germany, 2015.

[42] Kim G. Larsen, Ulrik Nyman, and Andrzej Wąsowski. Modal I/O automata for interface and product line theories. In *ESOP 2007*, LNCS 4421, pages 64–79. Springer, 2007.

[43] Kim G. Larsen, Ulrik Nyman, and Andrzej Wąsowski. On modal refinement and consistency. In *CONCUR 2007*, LNCS 4703, pages 105–119. Springer, 2007.

[44] Kim G. Larsen and Liu Xinxin. Equation solving using modal transition systems. In *LICS 1990*, pages 108–117. IEEE Computer Society, 1990.

[45] Kim Guldstrand Larsen. Modal specifications. In *Automatic Verification Methods for Finite State Systems*, LNCS 407, pages 232–246. Springer, 1989.

[46] Kim Guldstrand Larsen. Proof systems for satisfiability in hennessy-milner logic with recursion. *Theor. Comput. Sci.*, 72(2&3):265–288, 1990.

[47] Lars Luthmann, Stephan Mennicke, and Malte Lochau. Towards an I/O conformance testing theory for software product LLine based on modal interface automata. In *FMSPLE 2015*, pages 1–13, 2015.

[48] Lars Luthmann, Stephan Mennicke, and Malte Lochau. Compositionality, decompositionality and refinement in input/output conformance testing. In *FACS 2016*, pages 54–72, 2016.

[49] Gerald Lüttgen and Walter Vogler. Conjunction on processes: Full abstraction via ready-tree semantics. *Theor. Comput. Sci.*, 373(1-2):19–40, 2007.

[50] Gerald Lüttgen and Walter Vogler. Safe reasoning with logic LTS. *Theor. Comput. Sci.*, 412(28):3337–3357, 2011.

[51] Gerald Lüttgen and Walter Vogler. Modal interface automata. *Logical Methods in Computer Science*, 9(3), 2013.

[52] Gerald Lüttgen and Walter Vogler. Richer interface automata with optimistic and pessimistic compatibility. *ECEASST*, 66, 2013.

[53] Gerald Lüttgen, Walter Vogler, and Sascha Fendrich. Richer interface automata with optimistic and pessimistic compatibility. *Acta Inf.*, 52(4-5):305–336, 2015.

[54] Nancy A. Lynch. *Distributed Algorithms.* Morgan Kaufmann, 1996.

[55] Michael G. Main. Trace, failure and testing equivalences for communicating processes. *International Journal of Parallel Programming*, 16(5):383–400, 1987.

[56] Zohar Manna and Amir Pnueli. The anchored version of the temporal framework. In *Linear Time, Branching Time and Partial Order in Logics and Models for Concurrency, School/Workshop, Noordwijkerhout, The Netherlands, May 30 - June 3, 1988, Proceedings*, LNCS 354, pages 201–284. Springer, 1988.

[57] Robin Milner. *Communication and concurrency.* Prentice Hall, 1989.

[58] Jean-Baptiste Raclet. Residual for component specifications. *Electr. Notes Theor. Comput. Sci.*, 215:93–110, 2008.

[59] Jean-Baptiste Raclet, Eric Badouel, Albert Benveniste, Benoît Caillaud, Axel Legay, and Roberto Passerone. A modal interface theory for component-based design. *Fundamenta Informaticae*, 108(1-2):119–149, 2011.

[60] Arend Rensink and Walter Vogler. Fair testing. *Inf. Comput.*, 205(2):125–198, 2007.

[61] Christoph Franz Schlosser. Eio-automaten mit parallelkomposition ohne internalisierung. B.Sc. thesis, Universität Augsburg, 2012.

[62] Lev Sorokin. $\mathcal{F}$-semantik für disjunktive modale transitionssysteme. B.Sc. thesis, Universität Augsburg, April 2014.

[63] Christian Stahl, Peter Massuthe, and Jan Bretschneider. Deciding substitutability of services with operating guidelines. In *ToPNoC II*, LNCS 5460, pages 172–191. Springer, 2009.

[64] Jan Tretmans. Conformance testing with labelled transition systems: Implementation relations and test generation. *Computer Networks and ISDN Systems*, 29(1):49–79, 1996.

[65] Antti Valmari. Failure-based equivalences are faster than many believe. In *Structures in Concurrency Theory*, Workshops in Computing, pages 326–340. Springer, 1995.

[66] Antti Valmari. The weakest deadlock-preserving congruence. *IPL*, 53(6):341–346, 1995.

[67] Machiel van der Bijl, Arend Rensink, and Jan Tretmans. Compositional testing with ioco. In *Formal Approaches to Software Testing (FATES)*, volume 2931 of *LNCS*, pages 86–100. Springer, 2004.

[68] Rob J. van Glabbeek. The linear time - branching time spectrum II. In *CONCUR 1993*, LNCS 715, pages 66–81. Springer, 1993.

[69] Walter Vogler. Failures semantics and deadlocking of modular Petri nets. *Acta Inf.*, 26(4):333–348, 1989.

[70] Walter Vogler. *Modular Construction and Partial Order Semantics of Petri Nets. (LNCS 625).* LNCS 625. Springer, 1992.