Specification of systems with parameterised events: an institution-independent approach

Rolf Hennicker, Alexander Knapp

Angaben zur Veröffentlichung / Publication details:

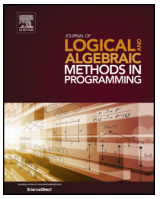
Hennicker, Rolf, and Alexander Knapp. 2022. "Specification of systems with parameterised events: an institution-independent approach." *Journal of Logical and Algebraic Methods in Programming* 128: 100791. https://doi.org/10.1016/j.jlamp.2022.100791.

Journal of Logical and Algebraic Methods in Programming 128 (2022) 100791

Contents lists available at ScienceDirect

Journal of Logical and Algebraic Methods in Programming

www.elsevier.com/locate/jlamp



Specification of systems with parameterised events: An institution-independent approach

Rolf Hennicker [a,*], Alexander Knapp [b,*]

[a] Ludwig-Maximilians-Universität München, Munich, Germany
[b] Universität Augsburg, Augsburg, Germany

ARTICLE INFO

Article history:
Received 27 March 2022
Accepted 5 July 2022
Available online 13 July 2022

Dedicated to Luis Soares Barbosa

ABSTRACT

Event-based systems operate in an environment that signals events upon which the system reacts. Besides the control flow of such systems also their data flow is of major importance. We present an event/data-based institution which is generic in the underlying data state institution. The logic is based on previous developments [14,8] and is now extended to take into account event parameters, quantification over data, and non-deterministic choice of arguments in an institution-independent way. We show that the resulting framework forms again an institution.

1. Introduction

Event-based systems are an important kind of modern computing systems. They live typically in an environment which signals events upon which the system reacts in certain ways. Such systems are strongly stateful in the sense that not any event is meaningful at any time and the same event can cause different reactions depending on the current system state.

There is a considerable amount of approaches in the literature dealing with formal methods for the development of event-based systems. Some of them propose constructive formalisms that aim at the description of concrete designs or implementations, like e.g., Event-B [1,4], symbolic transition systems [18], and UML behavioural and protocol state machines [17,10]. On the other hand, there are logical formalisms to express desired properties of event-based systems. Among them are temporal logics integrating state- and event-based styles [21], and various kinds of modal logics, like dynamic logic [7] or the modal μ-calculus with data and time [6]. The gap between logics and constructive specification is usually filled by checking whether *the* model of a constructive specification satisfies certain logical formulæ.

Instead of analysing a concrete design in a "post mortem" fashion it may, however, be useful to proceed in a series of refinement steps, where each step represents gradual design decisions such that errors can be detected earlier in the development process. This method has been extensively studied in the area of abstract data types and for developing functional programs. It was consolidated in the seminal book of Sannella and Tarlecki [20]. There it is argued that in the theory of concurrency a "commonly accepted solution still seems to be outstanding" (p. 157).

1.1. Some bits of history

The last statement was a major motivation to organise in 2015 a meeting in Aveiro where Luis, Alexandre (Madeira), Manuel (Martins), and Rolf met to work on an appropriate logic supporting both, abstract specifications of properties of

* Corresponding authors.
E-mail addresses: hennicke@ifi.lmu.de (R. Hennicker), knapp@isse.de (A. Knapp).

https://doi.org/10.1016/j.jlamp.2022.100791

event-based systems, such as safety and liveness, and concrete designs formulated in *the same* logic. Then refinement may be expressed by logical implication (as already suggested in TLA [11] which is, however, not event-based). The fruitful discussions in the group led to the idea to use dynamic logic style specifications, with regular expressions of events as modalities, for the description of abstract system properties and to complement them by integrating state variables, binders, and jumps known from hybrid logics [2]. The latter allow to characterise logically concrete, recursive process structures. The outcome of the investigations led to the so-called "dynamic logic with binders" [13] which was formalised in [14] as an institution in the sense of Goguen and Burstall [5]. It has been shown that this logic is well suited for the rigorous development of reactive component systems by applying appropriate implementation constructors (like parallel composition, relabelling, etc.) in refinement steps. This is reflected by the semantic models of the logic which are reachable transition systems with initial states where the transitions are labelled by (atomic) events. In this formalism states play the role of control states which determine the enabledness of events.

Dynamic logic with binders was the basis for several further developments in cooperation with our Portuguese colleagues. Most importantly it was extended to take into account data in [9] leading to the event/data-based logic $\mathcal{E}^{\downarrow}$. The states of a transition system are now configurations γ with an associated data state $\omega(\gamma)$ modelling the data administrated by a system. In $\mathcal{E}^{\downarrow}$-logic events e are augmented by state transition predicates ψ leading to (atomic) event expressions of the form $e/\!\!/\psi$. They can be combined to complex event expressions λ according to the operators of dynamic logic for sequential composition $\lambda_1;\lambda_2$, alternative $\lambda_1+\lambda_2$, and iteration λ^*. Event expressions λ are used as modalities in event/data formulæ according to the grammar

$$\varrho ::= \mathsf{true} \mid \varphi \mid \langle\lambda\rangle\varrho \mid \neg\varrho \mid \varrho \vee \varrho \mid \ldots \text{ (hybrid features omitted here)}$$

$\langle\lambda\rangle$ is the usual diamond operator expressing possibility and its dual, the box operator $[\lambda]$ expressing necessity, is defined as usual by $[\lambda]\varrho = \neg\langle\lambda\rangle\neg\varrho$. In the grammar, φ is a data state formula to be evaluated w.r.t. the data state $\omega(\gamma)$ of a configuration γ. Transition predicates ψ occurring in event expressions $e/\!\!/\psi$ specify the admissible effects of an event on data states. They are evaluated w.r.t. *two* data states, the data state $\omega(\gamma)$ of the source configuration γ before the event occurred and the data state $\omega(\gamma')$ of the configuration γ' after the event has happened.

Example 1. As a simple example we consider a counter with upper bound. Its data states are determined by the current value of the counter, represented by an attribute $\mathsf{val} : \mathsf{Nat}$, and the value of the upper bound represented by an attribute $\mathsf{max} : \mathsf{Nat}$. Both values are natural numbers. There is only one event inc; it is supposed to increment the counter value by 1. To express that in a configuration where the counter value is smaller than max an increment is possible, we use the formula

$$\mathsf{val} < \mathsf{max} \rightarrow \langle \mathsf{inc}/\!\!/\mathsf{val}' = \mathsf{val} + 1\rangle \mathsf{true} \tag{1}$$

where val' denotes the new value of the counter. We can more generally require that "whenever the counter value is smaller than the upper bound an increment is possible." This progress property is specified by formula (2):

$$[(\mathsf{inc}/\!\!/\mathsf{true})^*](\mathsf{val} < \mathsf{max} \rightarrow \langle \mathsf{inc}/\!\!/\mathsf{val}' = \mathsf{val} + 1\rangle \mathsf{true}) \tag{2}$$

Note that the expression $[(\mathsf{inc}/\!\!/\mathsf{true})^*]$ ranges over all reachable states of the counter, since inc is the only event considered here. □

$\mathcal{E}^{\downarrow}$-logic did rely on a particular realisation of data states modelled by assignments of values to attributes. In [8] we have developed a generic institution for event/data-based logic, denoted by $\mathcal{E}^{\downarrow}(\vec{\mathcal{D}})$, which is parametrised by an underlying data state institution $\vec{\mathcal{D}}$. Any concrete data institution which satisfies the amalgamation property can be used as a basis to instantiate $\mathcal{E}^{\downarrow}(\vec{\mathcal{D}})$. It is then even possible to change — during system development — a data state institution, usually from a poorer to a more expressive one, by means of an institution co-morphism.

1.2. Parameterising events

The approaches considered so far have a serious limitation: events are just names and do not support parameters. This excludes a lot of applications, like withdrawing a certain amount of money from an account or, considering our previous example, incrementing a counter by different values. In the latter case, we would like to add a parameter of type Nat to the event inc and represent this parameterised event by a term like $\mathsf{inc}(\mathsf{x})$.

To specify system properties involving parameterised events the quantification of parameters must be clarified. For instance, for a modal formula like $\langle \mathsf{inc}(\mathsf{x})/\!\!/\mathsf{val}' = \mathsf{val} + \mathsf{x}\rangle \mathsf{true}$ it should be made clear whether the existence of an increment transition is required *for all* valuations of x or only *for some* valuation of x. In model-theoretic approaches parameters are usually implicitly universally quantified. In the dynamic logic of [6] one can choose universal and existential quantification as desired. The recent logic for specifying properties of UML state machines in [19] proposes two kinds of the modal diamond operator, one for universal and one for existential interpretation of event parameters. We must, however, also consider cases where quantification is not directly bound to modal operators but can occur at various places in an event/data

formula. For instance, for adjusting formula (1) from above to parameterised increments the parameter x must be bound early enough as done in formula (1a):

$$\forall x\,.\,(\mathsf{val} + x \leq \mathsf{max} \rightarrow \langle \mathsf{inc}(x) /\!\!/ \mathsf{val}' = \mathsf{val} + x\rangle \mathsf{true}) \tag{1a}$$

We also want to express the progress property (2) from above in the parameterised case. An attempt would be to use the formula

$$\forall x\,.\,[(\mathsf{inc}(x) /\!\!/ \mathsf{true})^*]\,\forall x\,.\,(\mathsf{val} + x \leq \mathsf{max} \rightarrow \langle \mathsf{inc}(x) /\!\!/ \mathsf{val}' = \mathsf{val} + x\rangle \mathsf{true}) \tag{2a}$$

Unfortunately this formula does only subsume progress in configurations which are reachable by an arbitrary number of increments each one using the *same value* for x. In the general case we would need, in each step of the iteration, a non-deterministic choice for the instances of (the "first") x. Note that this cannot be expressed by the choice operator "+" of dynamic logic if the domains of parameter values are infinite. As a solution we propose atomic event/data actions like $\mathsf{inc}(\textbf{any}\ x) /\!\!/ \mathsf{true}$ where **any** x ranges over all values of x for which the transition predicate, here true, is satisfied. Then the desired progress property is expressed by the formula (2b):

$$[(\mathsf{inc}(\textbf{any}\ x) /\!\!/ \mathsf{true})^*]\forall x\,.\,(\mathsf{val} + x \leq \mathsf{max} \rightarrow \langle \mathsf{inc}(x) /\!\!/ \mathsf{val}' = \mathsf{val} + x\rangle \mathsf{true}) \tag{2b}$$

where the box together with the iteration refers to all states that are reachable with arbitrary arguments of inc.

In this paper we will omit hybrid operators (like binders and jumps) from $\mathcal{E}^{\downarrow}(\vec{\mathcal{D}})$ and consider its dynamic logic version denoted by $\mathcal{E}(\vec{\mathcal{D}})$. Indeed hybrid features have no impact on parameterised events and our study could be easily enlarged to include them. We propose an extension of $\mathcal{E}(\vec{\mathcal{D}})$-logic to $\mathcal{E}_{\mathrm{p}}(\vec{\mathcal{D}})$-logic which allows us to treat parameterised events, quantification, and non-deterministic choice of arguments in an institution-independent way. In particular, we will show that this extension forms again an institution.

Structure of the paper. After our personal tribute to Luis we start with a summary of basic definitions for institutions in Sect. 2. Then we present the ideas and the essential formal ingredients (signatures, structures, sentences and satisfaction relations) of our concepts in Sect. 3 to Sect. 5. More technical details, like signature morphisms, sentence translations, reduct functors, and proofs, are provided in the appendices. We finish with some concluding remarks in Sect. 6.

1.3. Personal note

Luis and Rolf know each other since 2005 when Luis was the PC chair of the International Workshop on Formal Aspects of Component Software (FACS) at the United Nations University in Macao where Rolf gave a talk. Both came into contact because of their common interest in the formal specification and development of component systems. In 2010 Luis invited Rolf to become an external consultant of the national Portuguese MONDRIAN project coordinated by Luis. This was the origin of a closer collaboration between the two underpinned by several scientific meetings in connection with project workshops and a further FACS workshop organised by Luis in autumn 2010 in Guimarães. The MONDRIAN project dealt, in particular, with dynamically reconfigurable components. This topic was further investigated in the PhD thesis of Alexandre Madeira, supervised by Luis and Manuel António Martins, where Rolf acted as an external examiner in 2013. A discussion on refinement notions in this context led to a first scientific publication with Luis and Rolf as co-authors [15]. This was followed by further fruitful meetings and the common publications [13,14]. In the meanwhile Alexander (Knapp) joined the collaboration and as an outcome of reciprocal scientific visits in Aveiro, Braga, and Munich the common work with our Portuguese colleagues was successfully continued resulting in currently ten peer-reviewed publications, among them four journal papers.

We would like to thank Luis very cordially for his initiatives which brought all these results on the way. He is not only an outstanding scientist but also a very friendly and open minded colleague. We are looking forward to further inspiring meetings and collaborations with him.

2. Basic notions of institutions

The notion of an institution has been introduced in [5]. It is an abstract concept which formalises requirements for logical systems used in rigorous, formal program development. An institution clearly distinguishes between syntax provided in terms of signatures Σ and associated sets of Σ-sentences, semantics in terms of Σ-structures, and satisfaction relations $M \models_\Sigma \varphi$ between Σ-structures M and Σ-sentences φ.[1] Satisfaction is defined for semantic structures without the consideration of further semantic concepts like valuations of variables. Intuitively this means that Σ-sentences should be closed formulæ without free variables. The family of satisfaction relations has to obey the so-called satisfaction condition which expresses the idea that truth is invariant under "change of notation". For the categorical terminology used in the sequel of this paper we refer the reader to the book by Mac Lane [12] (where, however, functional order of morphism composition $g \circ f$ is used instead of the diagrammatic order $f; g$ often employed here).

[1] Differently to the original terminology models are called structures here to avoid ambiguity when talking about models of a specification.

Institution. An *institution* $(\mathbb{S}, Str, \mathrm{Sen}, \models)$ consists of

- a category $\mathbb{S}$ whose objects are called *signatures* and arrows *signature morphisms*;
- a functor $Str : \mathbb{S}^{\mathrm{op}} \to \mathrm{Cat}$, giving for each signature Σ a category whose objects are called Σ-*structures*, and whose arrows are called Σ-*(structure) morphisms*; each arrow $\sigma : \Sigma \to \Sigma'$ in $\mathbb{S}$ (i.e., $\sigma : \Sigma' \to \Sigma$ in $\mathbb{S}^{\mathrm{op}}$), is mapped to a functor $Str(\sigma) : Str(\Sigma') \to Str(\Sigma)$ called *reduct functor*, whose effect is to cast a structure of Σ' as a structure of Σ; when $M = Str(\sigma)(M')$ we say that M is the σ-reduct of M';
- a functor $\mathrm{Sen} : \mathbb{S} \to \mathrm{Set}$, giving for each signature a set whose elements are called *sentences* over that signature; each arrow $\sigma : \Sigma \to \Sigma'$ in $\mathbb{S}$ is mapped to a *sentence translation function* $\mathrm{Sen}(\sigma) : \mathrm{Sen}(\Sigma) \to \mathrm{Sen}(\Sigma')$;
- a family $\models = (\models_\Sigma \subseteq |Str(\Sigma)| \times \mathrm{Sen}(\Sigma))_{\Sigma \in |\mathbb{S}|}$ of *satisfaction relations* determining, for each signature Σ, satisfaction of Σ-sentences by Σ-structures (where $|Str(\Sigma)|$ denotes the objects of the category $Str(\Sigma)$)

such that for each signature morphism $\sigma : \Sigma \to \Sigma'$ in $\mathbb{S}$, the *satisfaction condition*

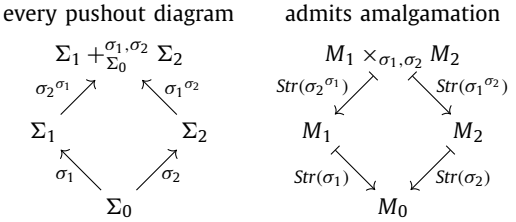
$$Str(\sigma)(M') \models_\Sigma \varphi \iff M' \models_{\Sigma'} \mathrm{Sen}(\sigma)(\varphi)$$

holds for each $M' \in |Str(\Sigma')|$ and $\varphi \in \mathrm{Sen}(\Sigma)$; graphically,

$$\begin{array}{ccc} \Sigma & Str(\Sigma) & \xrightarrow{\models_\Sigma} \mathrm{Sen}(\Sigma) \\ {\scriptstyle\sigma}\downarrow & {\scriptstyle Str(\sigma)}\uparrow & \downarrow{\scriptstyle \mathrm{Sen}(\sigma)} \\ \Sigma' & Str(\Sigma') & \xrightarrow{\models_{\Sigma'}} \mathrm{Sen}(\Sigma') \end{array}$$

Amalgamation property. Amalgamation will be used later to construct the union of data states and variable valuations. An institution $(\mathbb{S}, Str, \mathrm{Sen}, \models)$ has the *amalgamation property* [20, Def. 4.4.12] if all pushouts in $\mathbb{S}$ exist and

every pushout diagram admits amalgamation



In more detail, $(\Sigma_1 +_{\Sigma_0}^{\sigma_1,\sigma_2} \Sigma_2, (\sigma_{3-i}{}^{\sigma_i} : \Sigma_i \to \Sigma_1 +_{\Sigma_0}^{\sigma_1,\sigma_2} \Sigma_2)_{1 \le i \le 2})$ is a *pushout* along $\sigma_1 : \Sigma_0 \to \Sigma_1$ and $\sigma_2 : \Sigma_0 \to \Sigma_2$ in $\mathbb{S}$ if $\sigma_1; \sigma_2{}^{\sigma_1} = \sigma_2; \sigma_1{}^{\sigma_2}$ and, furthermore, for all $\Sigma' \in |\mathbb{S}|$ and $\sigma_i' : \Sigma_i \to \Sigma'$, $1 \le i \le 2$ satisfying $\sigma_1; \sigma_1' = \sigma_2; \sigma_2'$, there is a unique $\sigma : \Sigma_1 +_{\Sigma_0}^{\sigma_1,\sigma_2} \Sigma_2 \to \Sigma'$ with $\sigma_i' = \sigma_{3-i}{}^{\sigma_i}; \sigma$, $1 \le i \le 2$. Such a pushout admits *amalgamation* if

- for any two structures $M_i \in |Str(\Sigma_i)|$ such that $Str(\sigma_1)(M_1) = M_0 = Str(\sigma_2)(M_2)$, there exists a unique structure $M_1 \times_{\sigma_1,\sigma_2} M_2 \in |Str(\Sigma_1 +_{\Sigma_0}^{\sigma_1,\sigma_2} \Sigma_2)|$ such that $Str(\sigma_{3-i}{}^{\sigma_i})(M_1 \times_{\sigma_1,\sigma_2} M_2) = M_i$ for $1 \le i \le 2$; and
- for any two morphisms $\mu_i : M_i \to N_i$ in $Str(\Sigma_i)$ such that $Str(\sigma_1)(\mu_1) = Str(\sigma_2)(\mu_2)$, there is a unique morphism $\mu_1 \times_{\sigma_1,\sigma_2} \mu_2 : M_1 \times_{\sigma_1,\sigma_2} M_2 \to N_1 \times_{\sigma_1,\sigma_2} N_2$ such that $Str(\sigma_{3-i}{}^{\sigma_i})(\mu_1 \times_{\sigma_1,\sigma_2} \mu_2) = \mu_i$ for $1 \le i \le 2$.

As shown in [8] reducts preserve amalgamations.

Example 2. The following institution of *ground equational logic* $gEQ = (\mathbb{S}^{gEQ}, Str^{gEQ}, \mathrm{Sen}^{gEQ}, \models^{gEQ})$ will be used in our examples to model data states: $\mathbb{S}^{gEQ}$ consists of the many-sorted signatures $\Sigma = (S, F)$ with S-sorted function symbols F, where function symbols without arguments are called constants, and the signature morphisms $\sigma = (\sigma_S, \sigma_F) : \Sigma \to \Sigma'$ that preserve the sorting of the function symbols. A Σ-structure in $Str^{gEQ}(\Sigma)$ is a Σ-algebra $\mathfrak{A} = ((s^{\mathfrak{A}})_{s \in S}, (f^{\mathfrak{A}})_{f \in F})$ interpreting the sorts by non-empty sets and the function symbols by (total) functions; a Σ-structure morphism is a Σ-algebra homomorphism $h = (h_s : s^{\mathfrak{A}_1} \to s^{\mathfrak{A}_2})_{s \in S}$. The reduct of a Σ'-algebra $\mathfrak{A}'$ along $\sigma : \Sigma \to \Sigma'$ yields the Σ-algebra $\mathfrak{A}'|\sigma = ((\sigma_S(s)^{\mathfrak{A}'})_{s \in S}, (\sigma_F(f)^{\mathfrak{A}'})_{f \in F})$, and the reduct of a Σ'-algebra homomorphism $h' : \mathfrak{A}_1' \to \mathfrak{A}_2'$ the Σ-algebra homomorphism $h'|\sigma = (h'_{\sigma_S(s)})_{s \in S}$, such that $Str^{gEQ}(\sigma) = -|\sigma : Str^{gEQ}(\Sigma') \to Str^{gEQ}(\Sigma)$. A signature Σ induces a sort-indexed family of terms t, defined inductively, and the set of sentences $\mathrm{Sen}^{gEQ}(\Sigma)$ given by the grammar

$$\varphi ::= t_1 = t_2 \mid \mathrm{true} \mid \neg\varphi \mid \varphi_1 \vee \varphi_2 \,.$$

For a signature morphism $\sigma : \Sigma \to \Sigma'$, the term translation and sentence translation $\mathrm{Sen}^{gEQ}(\sigma) : \mathrm{Sen}^{gEQ}(\Sigma) \to \mathrm{Sen}^{gEQ}(\Sigma')$ are defined inductively as preserving the term and sentence structure, respectively. The evaluation of a term t of sort s over

a Σ-algebra $\mathfrak{A}$ is written $t^{\mathfrak{A}}$ and yields an element of $s^{\mathfrak{A}}$. The *satisfaction relation* $\mathfrak{A} \models^{gEQ}_{\Sigma} \varphi$ for a sentence $\varphi \in \mathrm{Sen}^{gEQ}(\Sigma)$ is inductively given by

- $\mathfrak{A} \models^{gEQ}_{\Sigma} t_1 = t_2$ iff $(t_1)^{\mathfrak{A}} = (t_2)^{\mathfrak{A}}$;
- $\mathfrak{A} \models^{gEQ}_{\Sigma}$ true;
- $\mathfrak{A} \models^{gEQ}_{\Sigma} \neg\varphi$ iff not $\mathfrak{A} \models^{gEQ}_{\Sigma} \varphi$;
- $\mathfrak{A} \models^{gEQ}_{\Sigma} \varphi_1 \vee \varphi_2$ iff $\mathfrak{A} \models^{gEQ}_{\Sigma} \varphi_1$ or $\mathfrak{A} \models^{gEQ}_{\Sigma} \varphi_2$.

The institution *gEQ* has the amalgamation property [20]. □

3. Motivating discussions

Our goal is to introduce parameterised events, and consequently quantification of parameters and variables in $\mathcal{E}(\vec{\mathcal{D}})$-logic, i.e., to make them available for arbitrary data state institutions satisfying the amalgamation property. The resulting logic is called $\mathcal{E}_{\mathrm{p}}(\vec{\mathcal{D}})$-logic. In Sect. 1.2 we have already summarised some issues related to the introduction of parameterised events. In this section we will provide some motivations and discussions for designing an appropriate framework to achieve our goals.

The first question to be solved is how parameters can be represented in an institution-independent way. This question is related to the treatment of open formulæ in an arbitrary institution $\mathfrak{I}$. Given a signature Σ, according to [20, Sect. 4.4.2] any pair (φ, ξ), where $\xi : \Sigma \to \hat{\Sigma}$ is a signature morphism and φ is a $\hat{\Sigma}$-sentence, is an open Σ-formula. (The underlying intuition stems from first-order logic where variables can be considered as constants which are added to a signature Σ thus yielding signature $\hat{\Sigma}$.) For any Σ-structure M and open Σ-formula (φ, ξ), a valuation is a $\hat{\Sigma}$-expansion $\hat{M}$ of M, i.e., the reduct of $\hat{M}$ along ξ is M, and (φ, ξ) is said to hold in M under valuation $\hat{M}$ if $\hat{M} \models_{\hat{\Sigma}} \varphi$.

Based on these notions, Sannella and Tarlecki consider universally closed Σ-formulæ $\forall \xi \,.\, \varphi$ where (φ, ξ) is an open Σ-formula. $\forall \xi \,.\, \varphi$ holds in a Σ-structure M if $\hat{M} \models_{\hat{\Sigma}} \varphi$ holds *for all* $\hat{\Sigma}$-expansions $\hat{M}$ of M. Then, for a collection $\mathcal{X}$ of signature morphisms in $\mathfrak{I}$, Sannella and Tarlecki define the institution $\mathfrak{I}^{\forall(\mathcal{X})}$ such that, for any signature Σ, its set of Σ-sentences is the disjoint union of the Σ-sentences of $\mathfrak{I}$ with the collection of all universal closures $\forall \xi \,.\, \varphi$ of open Σ-formulæ (φ, ξ) such that $\xi \in \mathcal{X}$.

Unfortunately we cannot simply use this approach in our setting. One reason is that we would like to have nested quantifications which are not supported by $\mathfrak{I}^{\forall(\mathcal{X})}$. But there is a more demanding problem, since event/data formulæ specify dynamic behaviours with changing states. In this context, we have to relate data states of pre- and post-configurations such that the values of parameters stay the same while the underlying data states may change.

For our "counter" example consider, e.g., the formula (1a) from Sect. 1.2.

$$\forall \mathsf{x} \,.\, (\mathsf{val} + \mathsf{x} \leq \mathsf{max} \rightarrow \langle \mathsf{inc}(\mathsf{x}) /\!\!/ \mathsf{val}' = \mathsf{val} + \mathsf{x} \rangle \mathsf{true}) \tag{1a}$$

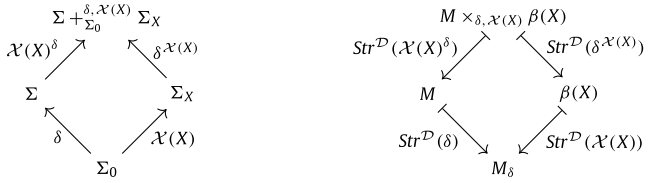
Assume we have an event/data transition system and a configuration γ. Following the lines from above a valuation of x would be an expansion $\hat{\omega}$ of the data state $\omega(\gamma)$ where x is a constant interpreted by some natural number. If $\mathsf{val} + \mathsf{x} \leq \mathsf{max}$ holds in $\hat{\omega}$ then there must exist an increment transition to a configuration γ' such that the transition predicate $\mathsf{val}' = \mathsf{val} + \mathsf{x}$ holds. This must be checked on the basis of the two data states $\omega(\gamma)$ and $\omega(\gamma')$. So we would need two expansions, one for $\omega(\gamma)$ and one for $\omega(\gamma')$ such that x is interpreted by the same value in both extensions. Another example is the formula

$$\forall \mathsf{x} \,.\, [(\mathsf{inc}(\mathsf{x}) /\!\!/ \mathsf{true}); (\mathsf{inc}(\mathsf{x}) /\!\!/ \mathsf{true})] \mathsf{false} \tag{3}$$

which expresses that incrementing the counter twice in a row with *the same* value is not allowed. Again for evaluation of the formula data states would change but the value of the parameter should stay the same.

Though not dealing with parameters of events (modalities resp.) the treatment of quantification in the "hybridisation" method in [16,3] could show a workaround since there not only expansions of states but expansions of whole transition systems are considered when evaluating quantified formulæ. This approach needs not only extensions on the level of transition systems but also, additionally to the models of a hybrid signature, subclasses of so-called constrained models to express in an institution-independent way that variables cannot change their value after a transition. Moreover, the general hybridisation method does not rely on transition predicates to express desired properties of pre- and post-data states of transitions which we believe are a convenient tool for the specification of event-based systems with changing data states. Moreover, the treatment of non-deterministic arguments would not be possible since the hybridisation method does not support parameterised modalities. Related to parameterised events, but not institution-independent, are the typed modalities proposed in [22] for dynamic networks of interactions.

In our approach we propose to go a different way, still taking the idea of modelling variables in an institution independent way by signature morphisms and we will consider event parameters as (logical) variables a well. We will, however, ensure that the interpretation of variables is separated from the data states of configurations as far as possible. This means that source and target data states of transitions (in an event/data transition system) as well as variables will only rely on a common primitive part representing, e.g., basic data types.



(a) pushout (b) amalgamation

Fig. 1. Open data state formulæ.

4. Data state institutions and open formulæ

We assume given an arbitrary base institution $\mathcal{D}$ satisfying the amalgamation property and construct on top of it a *data state institution* $\vec{\mathcal{D}}$ such that signatures in $\vec{\mathcal{D}}$ are signature morphisms $\delta : \Sigma_0 \to \Sigma$ in $\mathcal{D}$, δ-structures in $\vec{\mathcal{D}}$ are those Σ-structures M in $\mathcal{D}$ which are expansions of a common Σ_0-structure M_δ in $\mathcal{D}$, and δ-sentences are just the Σ-sentences in $\mathcal{D}$. The satisfaction relation $\models^{\vec{\mathcal{D}}}_\delta$ is $\models^{\mathcal{D}}_\Sigma$. As detailed in Appendix B, the class of signatures in $\vec{\mathcal{D}}$ can be chosen as a subcategory of the class of signature morphisms in $\mathcal{D}$. In fact, one could choose a fixed base signature Σ_0 such that all signatures in $\vec{\mathcal{D}}$ have the same domain Σ_0 and then the reducts of all structures in $\vec{\mathcal{D}}$ are a fixed Σ_0-structure in $\mathcal{D}$; see the data state institution $gEQ_{\Sigma_0,\mathfrak{A}_0}$ in Example 3.

Let $\delta : \Sigma_0 \to \Sigma$ be a signature in $\vec{\mathcal{D}}$. A collection $\mathcal{X}$ of "variables" for δ consists of a countable set $|\mathcal{X}|$ of variable names together with a signature $\mathcal{X}(x) : \Sigma_0 \to \Sigma_x$ in $\vec{\mathcal{D}}$ attached to each $x \in |\mathcal{X}|$ (note that several $x \in |\mathcal{X}|$ may be equipped with the same signature). For every finite subset $X \subseteq |\mathcal{X}|$ we assume a canonically chosen colimit signature Σ_X in $\mathcal{D}$ with morphism $\mathcal{X}(X) : \Sigma_0 \to \Sigma_X$ constructed by a finite iteration of pushouts over the family $(\mathcal{X}(x) : \Sigma_0 \to \Sigma_x)_{x\in X}$ such that $\Sigma_\emptyset = \Sigma_0$.

An *open* δ-formula over $\mathcal{X}$ is then a pair, written $\varphi(X)$, such that X is a finite subset of $|\mathcal{X}|$ and φ is a sentence over the pushout signature $\Sigma +^{\delta,\mathcal{X}(X)}_{\Sigma_0} \Sigma_X$ in $\mathcal{D}$ shown in Fig. 1a, also abbreviated as $\Sigma +_{\Sigma_0} \Sigma_X$. The set of open δ-formulæ over $\mathcal{X}$ is denoted by $\mathrm{Frm}^{\vec{\mathcal{D}}}(\delta, \mathcal{X})$.

Let M_δ be the common Σ_0-structure for which all δ-structures are expansions. A *valuation* for the variables $\mathcal{X}$ is a function β which maps any $x \in |\mathcal{X}|$ to a $\mathcal{X}(x)$-structure of $\vec{\mathcal{D}}$ (i.e., a Σ_x-structure $\beta(x)$ in $\mathcal{D}$ whose reduct $Str^{\mathcal{D}}(\mathcal{X}(x))(\beta(x))$ along $\mathcal{X}(x)$ is M_δ).

We can now define satisfaction $M, \beta \models^{\vec{\mathcal{D}}}_{\beta,\mathcal{X}} \varphi(X)$ of an open δ-formula $\varphi(X)$ by a δ-structure M w.r.t. a valuation β. First, for a finite subset $X \subseteq |\mathcal{X}|$ we consider the limit $\beta(X)$ of Σ_x-structures $\beta(x)$ in $\mathcal{D}$ with morphism $Str^{\mathcal{D}}(\mathcal{X}(X))$ mapping $\beta(\emptyset)$ to M_δ and constructed by a finite iteration of amalgamations over the family $(Str^{\mathcal{D}}(\mathcal{X}(x)) : Str^{\mathcal{D}}(\Sigma_x) \to Str^{\mathcal{D}}(\Sigma_0))_{x\in X}$. The satisfaction $M, \beta \models^{\vec{\mathcal{D}}}_{\beta,\mathcal{X}} \varphi(X)$ is then defined as the satisfaction of the $(\Sigma +_{\Sigma_0} \Sigma_X)$-sentence φ by the amalgamated union of M and $\beta(X)$ in $\mathcal{D}$ shown in Fig. 1b, i.e.,

$$M, \beta \models^{\vec{\mathcal{D}}}_{\delta,\mathcal{X}} \varphi(X) \quad \text{if, and only if,} \quad M \times_{\delta,\mathcal{X}(X)} \beta(X) \models^{\mathcal{D}}_{\Sigma+_{\Sigma_0}\Sigma_X} \varphi \tag{A}$$

Note that the satisfaction relation of $\vec{\mathcal{D}}$ is still $\models^{\vec{\mathcal{D}}}_\delta = \models^{\mathcal{D}}_\Sigma$ as explained above. But the treatment of variables and the satisfaction definition $M, \beta \models^{\vec{\mathcal{D}}}_{\delta,\mathcal{X}} \varphi(X)$ involving valuations and open formulæ will be the basis for the satisfaction relation of the event/data logic with parameterised events and variables when sentences of the logic will be closed formulæ.

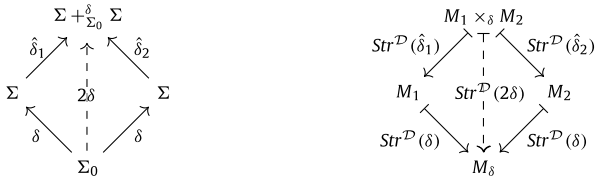
Our constructions can be lifted to the definition of open transition predicates ψ and for their evaluation w.r.t. two data states M and M'. The idea is to use an additional component for pushouts and for amalgamations such that target data states can be reflected. The important point in our construction is that valuations of variables still remain independent from concrete data states.

We formalise transition predicates and their interpretations by using a so-called 2-data state institution $2\vec{\mathcal{D}}$ constructed on top of $\vec{\mathcal{D}}$ (and hence $\mathcal{D}$). Signatures in $2\vec{\mathcal{D}}$ are the same as in $\vec{\mathcal{D}}$ but structures of $2\vec{\mathcal{D}}$ are pairs (M_1, M_2) intended to represent pre- and post-data states of transitions. Thus, for each signature $\delta : \Sigma_0 \to \Sigma$ in $2\vec{\mathcal{D}}$ (and hence $\vec{\mathcal{D}}$), δ-structures in $2\vec{\mathcal{D}}$ are pairs (M_1, M_2) of δ-structures M_1, M_2 in $\vec{\mathcal{D}}$.

Properties of such data state pairs (without considering variables yet) can be specified by the sentences of $2\vec{\mathcal{D}}$ which are built according to a pushout construction in $\mathcal{D}$. For each signature $\delta : \Sigma_0 \to \Sigma$ in $2\vec{\mathcal{D}}$ (and hence $\vec{\mathcal{D}}$ and hence a signature morphism in $\mathcal{D}$), we assume given a specifically chosen pushout of δ with itself in $\mathcal{D}$, as illustrated in Fig. 2a.

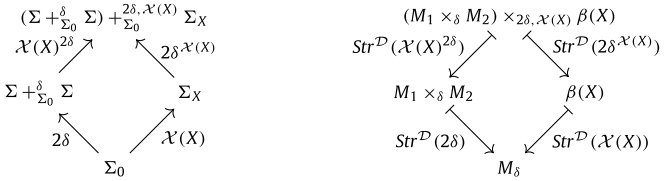
Then the set of δ-sentences in $2\vec{\mathcal{D}}$ is defined as the set of $(\Sigma +^{\delta}_{\Sigma_0} \Sigma)$-sentences in $\mathcal{D}$. The intuition behind this construction is that $(\Sigma +^{\delta}_{\Sigma_0} \Sigma)$-sentences may contain copies of the symbols in "$\Sigma \setminus \Sigma_0$" which can be separately evaluated in pre- and post-data states. For instance, in Example 1, $\mathsf{val}' = \mathsf{val} + 1$ is a transition predicate containing the symbol val (to be evaluated in a pre-data state) and its copy val' (to be evaluated in a post-data state).

The satisfaction relation in $2\vec{\mathcal{D}}$ for a δ-structure (M_1, M_2) in $2\vec{\mathcal{D}}$, such that M_δ is the common reduct of M_1 and M_2 along δ, relies on the construction of an amalgamation of M_1 and M_2 in $\mathcal{D}$ as shown in Fig. 2b.



(a) Pushout (b) Amalgamation

Fig. 2. 2-data state signatures and structures.



(a) Pushout (b) Amalgamation

Fig. 3. Open 2-data state formulæ.

Then, for any δ-sentence ψ in $2\vec{\mathcal{D}}$, i.e., ψ is a $(\Sigma +^{\delta}_{\Sigma_0} \Sigma)$-sentence in $\mathcal{D}$, the $2\vec{\mathcal{D}}$-satisfaction relation is defined by

$$(M_1, M_2) \models^{2\vec{\mathcal{D}}}_{\delta} \psi \quad \text{if, and only if,} \quad M_1 \times_\delta M_2 \models^{\mathcal{D}}_{\Sigma +^{\delta}_{\Sigma_0} \Sigma} \psi$$

Considering M_1 and M_2 as pre- and post-data states respectively, we see that the base signature Σ_0 in $\delta : \Sigma_0 \to \Sigma$ determines that part of data states whose interpretation, given by M_δ, has to be kept invariant while interpretations of the remaining part are flexible.

The treatment of variables for two data state formulæ extends the one for one data state formulæ from above as follows: Let $\delta : \Sigma_0 \to \Sigma$ be a signature in $2\vec{\mathcal{D}}$ (and hence in $\vec{\mathcal{D}}$) and let $\mathcal{X}$ be a collection of variables for δ with associated signatures $\mathcal{X}(x) : \Sigma_0 \to \Sigma_x$ for each $x \in |\mathcal{X}|$. The set of open 2-state formulæ over δ and $\mathcal{X}$, denoted by $\mathrm{Frm}^{2\vec{\mathcal{D}}}(\delta, \mathcal{X})$, is defined as the set of open state formulæ over 2δ and $\mathcal{X}$, i.e. $\mathrm{Frm}^{2\vec{\mathcal{D}}}(\delta, \mathcal{X}) = \mathrm{Frm}^{\vec{\mathcal{D}}}(2\delta, \mathcal{X})$. This means that an *open* 2-state δ-formula is a pair, written $\psi(X)$, such that X is a finite subset of $|\mathcal{X}|$ and ψ is a sentence over the pushout signature $(\Sigma +^{\delta}_{\Sigma_0} \Sigma) +^{2\delta,\mathcal{X}(X)}_{\Sigma_0} \Sigma_X$ in $\mathcal{D}$, shorter written as $(\Sigma +_{\Sigma_0} \Sigma) +_{\Sigma_0} \Sigma_X$ and shown in Fig. 3a.

The *satisfaction* of an open 2-state δ-formula $\psi(X) \in \mathrm{Frm}^{2\vec{\mathcal{D}}}(\delta, \mathcal{X})$ by a δ-structure (M_1, M_2) in $2\vec{\mathcal{D}}$ w.r.t. a valuation β for $\mathcal{X}$ is defined by

$$(M_1, M_2), \beta \models^{2\vec{\mathcal{D}}}_{\delta,\mathcal{X}} \psi(X) \quad \text{if, and only if,} \quad M_1 \times_\delta M_2, \beta \models^{\vec{\mathcal{D}}}_{2\delta,\mathcal{X}} \psi(X)$$

By (A) the latter is the same as $(M_1 \times_\delta M_2) \times_{2\delta,\mathcal{X}(X)} \beta(X) \models^{\mathcal{D}}_{\Sigma +_{\Sigma_0} \Sigma +_{\Sigma_0} \Sigma_X} \psi$ where the amalgamated union is shown in Fig. 3b.

Example 3. We want to express $\mathsf{val} + \mathsf{x} \leq \mathsf{max}$ as a data state formula and $\mathsf{val}' = \mathsf{val} + \mathsf{x}$ as a 2-data state formula over the base institution *gEQ* of ground equational logic. As the base many-sorted signature Σ_0 we choose a signature which has the primitive sorts Nat and Bool and contains function symbols $+ : \mathsf{Nat}\ \mathsf{Nat} \to \mathsf{Nat}$ and $\leq : \mathsf{Nat}\ \mathsf{Nat} \to \mathsf{Bool}$ (both used in infix notation) and usual constants like $\mathsf{true} : \mathsf{Bool}$. The accompanying base algebra $\mathfrak{A}_0$ interprets Nat as the natural numbers, Bool as the Booleans and the function symbols as expected. We thus obtain a data state institution $gEQ_{\Sigma_0,\mathfrak{A}_0}$. For capturing val and max as data state attributes we consider the many-sorted signature Σ which extends Σ_0 by the two constants $\mathsf{val} : \mathsf{Nat}$ and $\mathsf{max} : \mathsf{Nat}$. Hence the structures $Str^{gEQ_{\Sigma_0,\mathfrak{A}_0}}(\delta)$ for $\delta : \Sigma_0 \to \Sigma$ are Σ-algebras containing the fixed interpretation $\mathfrak{A}_0$ together with arbitrary natural numbers for val and max. Moreover, the δ-structures of $2gEQ_{\Sigma_0,\mathfrak{A}_0}$ are pairs of Σ-algebras $(\mathfrak{A}_1, \mathfrak{A}_2)$ sharing $\mathfrak{A}_0$. For finally adding the variable x we consider the variables $\mathcal{X}$ with $|\mathcal{X}| = \{\mathsf{x}\}$ and the inclusion $\mathcal{X}(\mathsf{x}) : \Sigma_0 \to \Sigma_\mathsf{x}$ where Σ_x shows an additional constant $\mathsf{x} : \mathsf{Nat}$. The combined signature $(\Sigma +^{\delta}_{\Sigma_0} \Sigma) +_{\Sigma_0} \Sigma_\mathsf{x}$ then contains the shared Σ_0 and constants $\mathsf{val} : \mathsf{Nat}$, $\mathsf{val}' : \mathsf{Nat}$, $\mathsf{max} : \mathsf{Nat}$, $\mathsf{max}' : \mathsf{Nat}$, and $\mathsf{x} : \mathsf{Nat}$. Thus, the equation $\mathsf{val}' = \mathsf{val} + \mathsf{x}$ is an open formula in $\mathrm{Frm}^{2gEQ_{\Sigma_0,\mathfrak{A}_0}}(\delta, \mathcal{X})$. Checking whether it is satisfied involves a δ-structure in $2gEQ_{\Sigma_0,\mathfrak{A}_0}$ which is determined by values for val, val′, max, max′ and a valuation $\beta(\mathsf{x})$ involving a single natural number for x. Similarly, $\mathsf{val} + \mathsf{x} \leq \mathsf{max}$, abbreviating the equation $(\mathsf{val} + \mathsf{x} \leq \mathsf{max}) = \mathsf{true}$, is an open formula in $\mathrm{Frm}^{gEQ_{\Sigma_0,\mathfrak{A}_0}}(\delta, \mathcal{X})$ and its satisfaction is checked w.r.t. a δ-structure in $gEQ_{\Sigma_0,\mathfrak{A}_0}$ determined by values for val, max and a valuation $\beta(\mathsf{x})$. The formula $\mathsf{max}' = \mathsf{max}$ contains no free variables and therefore its satisfaction in a δ-structure in $2gEQ_{\Sigma_0,\mathfrak{A}_0}$ depends only on the values of max and max′ while the valuation $\beta(\mathsf{x})$ is irrelevant. In the examples considered here we could have added the equation $\mathsf{max}' = \mathsf{max}$ in the transition predicates to express that the upper bound of the counter cannot change. □

5. Event/data-based logic with parameterised events

In the sequel we assume given an arbitrary base institution $\mathcal{D}$ satisfying the amalgamation property and the data institution $\vec{\mathcal{D}}$ and the 2-data institution $2\vec{\mathcal{D}}$ as defined in Sect. 4 over $\mathcal{D}$. In contrast to the previous sections signatures in $\mathcal{D}$ will be denoted by $\Delta_0, \Delta, \ldots$ instead of $\Sigma_0, \Sigma, \ldots$ since notations with Σ will be reserved for event/data signatures.

A crucial ingredient of any event/data-based system are the events which may occur at a certain instance of time and may change the computation state as well as the data state of a system. In $\mathcal{E}(\vec{\mathcal{D}})$-logic, signatures consist of an event part, determined by a set of events, and a data part, determined by a signature of $\vec{\mathcal{D}}$. In $\mathcal{E}_{\mathrm{p}}(\vec{\mathcal{D}})$-logic, we extend the event part by allowing parametrised events and we adjust the other notions like structures, sentences, and satisfaction accordingly.

In this section we introduce the essential ingredients of $\mathcal{E}_{\mathrm{p}}(\vec{\mathcal{D}})$-logic. Elaborated details and proofs are given in Appendix C.

Parameterised events. Let $\delta : \Delta_0 \to \Delta$ be a signature in $\vec{\mathcal{D}}$. A parameterised event over δ consists of an event name e and a finite sequence of parameter types, each type being represented by a signature in $\vec{\mathcal{D}}$. More formally, an *event signature* $E = (|E|, \xi)$ over δ consists of a set of *event names* $|E|$ and a *parameter types* map ξ such that for all $e \in |E|$, $\xi(e) = \xi(e)_1, \ldots, \xi(e)_n$ is a sequence of signatures $\xi(e)_i : \Delta_0 \to \Delta_{e,i}$ in $\vec{\mathcal{D}}$.

Event occurrences provide concrete parameter values for events. They are formalised by structures of the signatures of the parameter types. Thus, an *event occurrence* for an $e \in |E|$ with parameter types $\xi(e)_i : \Delta_0 \to \Delta_{e,i}$ for $i = 1, \ldots, n$ is a pair $e(\mathrm{B})$ such that $\mathrm{B} = \mathrm{B}_1, \ldots, \mathrm{B}_n$ is a sequence of $\xi(e)_i$-structures in $\vec{\mathcal{D}}$. The set of event occurrences for an event signature E is denoted by $\hat{E}$.

Event/data signature. An *event/data signature* (ed signature, for short) $\Sigma = (E, \delta : \Delta_0 \to \Delta)$ consists of a data signature $\delta : \Delta_0 \to \Delta$ in $\vec{\mathcal{D}}$ and an event signature E over δ.

Structures of $\mathcal{E}_{\mathrm{p}}(\vec{\mathcal{D}})$. Any ed signature Σ determines a class of semantic Σ-structures, called *event/data transition systems*, which are transition systems with sets of initial states and, differently from $\mathcal{E}(\vec{\mathcal{D}})$, event occurrences as labels on transitions. The states of our transition systems are called *configurations*. There is a data state operator ω which assigns to each configuration γ a data state $\omega(\gamma)$ which is a $\vec{\mathcal{D}}$-structure. It is related to the function M in hybridised logics [16] which maps states to structures of an underlying base institution. Different configurations can have the same data states but different enabled event occurrences. Hence configurations may carry control flow information. Since we are interested here in describing (properties of) reactive systems which start their executions in some initial configurations, the state space of our semantic models is restricted to reachable configurations only.

A Σ-*event/data transition system* (Σ-*edts*) $M = (\Gamma, R, \Gamma_0, \omega)$ over an ed signature $\Sigma = (E, \delta : \Delta_0 \to \Delta)$ consists of

- a set of *configurations* Γ;
- a family of *transition relations* $R = (R_{\hat{e}} \subseteq \Gamma \times \Gamma)_{\hat{e} \in \hat{E}}$;
- a non-empty set of *initial configurations* $\Gamma_0 \subseteq \Gamma$ and
- a *data state operator* $\omega : \Gamma \to |Str^{\vec{\mathcal{D}}}(\delta)|$,

such that Γ is *reachable* via R, i.e., for all $\gamma \in \Gamma$ there are $\gamma_0 \in \Gamma_0$, $n \geq 0$, $\hat{e}_1, \ldots, \hat{e}_n \in \hat{E}$, and $(\gamma_i, \gamma_{i+1}) \in R_{\hat{e}_{i+1}}$ for all $0 \leq i < n$ with $\gamma_n = \gamma$.

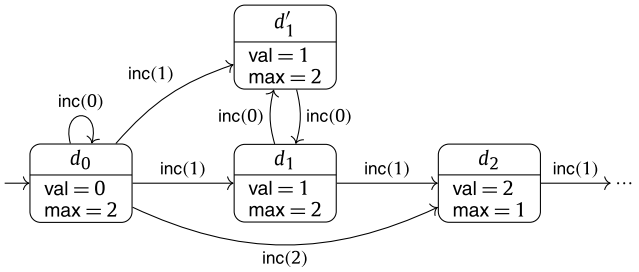
We write $\Gamma(M)$ for Γ, $R(M)$ for R, $\Gamma_0(M)$ for Γ_0 and $\omega(M)$ for ω. The class of all Σ-edts is denoted by $Str^{\mathcal{E}_{\mathrm{p}}(\vec{\mathcal{D}})}(\Sigma)$.

Example 4. Fig. 4 gives pictorial representations of Σ-event/data transition systems for the counter signature described in Example 3. The configurations γ are shown as rounded rectangles, their data states $\omega(\gamma)$ as valuations in the lower state compartment. The initial configurations are indicated by an arrow pointer. The arguments for the parameterised event inc which, in fact, are Σ_{x}-algebras are represented by the chosen value.

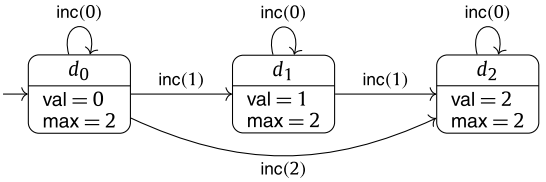
In Fig. 4a, only a few sample transitions are shown explicitly. The transition system is not "extensional": There are several configurations, like d_1 and d_1' for the same data state. By contrast, Fig. 4b gives the complete picture for a counter up to $\mathsf{max} = 2$, where this upper limit is kept constant. □

Event/data actions. Before defining the formulæ (sentences resp.) of our logic we must first provide an appropriate representation of parameterised events. We assume given an ed signature $\Sigma = (E, \delta)$ and a collection $\mathcal{X}$ of variables for δ. As usual in programming, we present the parameters of events by variables. Given an event name $e \in |E|$ with parameter types $\xi(e) = \xi(e)_1, \ldots, \xi(e)_n$, an *event term* is an expression $e(\mathbf{any}\ X)$ such that $X = x_1, \ldots, x_n$ is a list of variables in $\mathcal{X}$ with types $\mathcal{X}(x_i) = \xi(e)_i$ for $i = 1, \ldots, n$. The term $\mathbf{any}\ X$ allows us to express non-deterministic choice of actual event parameters as discussed in Sect. 1.2. More examples for the use of $\mathbf{any}\ X$ and convenient abbreviations are given in Example 5.

We denote the set of event terms over Σ and $\mathcal{X}$ by $E(\Sigma, \mathcal{X})$. Then atomic event/data actions over Σ have the form $e(\mathbf{any}\ X) /\!\!/ \psi(Y)$ with $e(\mathbf{any}\ X) \in E(\Sigma, \mathcal{X})$ an event term and $\psi(Y)$ a data state transition predicate formalised as an open formula in $2\vec{\mathcal{D}}$. Often X and Y coincide. Following the dynamic logic style we also use complex, structured actions formed



(a) Non-extensional transition system (incomplete).



(b) Extensional transition system (max constant, all transitions shown).

Fig. 4. "Counter" event/data transition systems.

over atomic event/data actions by union "+" (expressing non-deterministic choice), sequential composition ";", and iteration "*". The set $\Lambda(\Sigma, \mathcal{X})$ of Σ-*event/data actions* (Σ-*ed actions*) is given by the grammar

$$\lambda ::= e(\mathbf{any}\ X) /\!\!/ \psi(Y) \mid \lambda_1 + \lambda_2 \mid \lambda_1; \lambda_2 \mid \lambda^*$$

where $e(X) \in E(\Sigma, \mathcal{X})$ and $\psi(Y) \in \mathrm{Frm}^{2\vec{\mathcal{D}}}(\delta, \mathcal{X})$. The *free variables* $\mathrm{fvar}(\lambda)$ of an action λ are defined inductively by $\mathrm{fvar}(e(\mathbf{any}\ X) /\!\!/ \psi(Y)) = Y \setminus X$, $\mathrm{fvar}(\lambda_1 + \lambda_2) = \mathrm{fvar}(\lambda_1) \cup \mathrm{fvar}(\lambda_2) = \mathrm{fvar}(\lambda_1; \lambda_2)$, and $\mathrm{fvar}(\lambda^*) = \mathrm{fvar}(\lambda)$.

Event/data formulæ. For an event/data signature $\Sigma = (E, \delta)$ and data variables $\mathcal{X}$ for δ the set $\mathrm{Frm}^{\mathcal{E}_\mathrm{p}(\vec{\mathcal{D}})}(\Sigma, \mathcal{X})$ of *event/data formulæ* supporting parameterised events and quantification of data variables is defined by the grammar:

$$\varrho ::= \mathrm{true} \mid \varphi(X) \mid \langle\lambda\rangle\varrho \mid \exists x.\, \varrho \mid \neg\varrho \mid \varrho \vee \varrho$$

where $\varphi(X) \in \mathrm{Frm}^{\vec{\mathcal{D}}}(\delta, \mathcal{X})$, $\lambda \in \Lambda(\Sigma, \mathcal{X})$, and $x \in |\mathcal{X}|$. The modal box operator $[\lambda]\varrho$ stands for $\neg[\lambda]\neg\varrho$, universal quantification $\forall x.\, \varrho$ for $\neg\exists x.\, \neg\varrho$, and other boolean connectives like $\wedge$, $\rightarrow$, $\leftrightarrow$, etc. and the constant false are derived as usual.

The *free variables* $\mathrm{fvar}(\varrho)$ of an ed formula ϱ are defined inductively by $\mathrm{fvar}(\mathrm{true}) = \emptyset$, $\mathrm{fvar}(\varphi(X)) = X$, $\mathrm{fvar}(\langle\lambda\rangle\varrho) = \mathrm{fvar}(\lambda) \cup \mathrm{fvar}(\varrho)$, $\mathrm{fvar}(\exists x.\, \varrho) = \mathrm{fvar}(\varrho) \setminus \{x\}$, $\mathrm{fvar}(\neg\varrho) = \mathrm{fvar}(\varrho)$, and $\mathrm{fvar}(\varrho_1 \vee \varrho_2) = \mathrm{fvar}(\varrho_1) \cup \mathrm{fvar}(\varrho_2)$. An event/data formula ρ is *closed*, if $\mathrm{fvar}(\rho) = \emptyset$. For any ed signature Σ, the set of closed ed formulæ over Σ forms the *event/data sentences* (ed sentences) of $\mathcal{E}_\mathrm{p}(\vec{\mathcal{D}})$-logic, denoted by $\mathrm{Sen}^{\mathcal{E}_\mathrm{p}(\vec{\mathcal{D}})}(\Sigma)$.

Satisfaction relation of $\mathcal{E}_\mathrm{p}(\vec{\mathcal{D}})$. To define the satisfaction relation of $\mathcal{E}_\mathrm{p}(\vec{\mathcal{D}})$-logic we must first, as usual in dynamic logic, provide interpretations of the actions $\Lambda(\Sigma, \mathcal{X})$ in a Σ-edts M. Since actions may contain variables, such interpretations must be defined relative to a variable valuation β. Moreover, special care has to be taken about event terms. Let $e(\mathbf{any}\ X)$ be an event term with $X = x_1, \ldots, x_n$ and β_X a valuation for the variables in X. We then denote the event occurrence $e(\beta_X(x_i)_{1 \le i \le n})$ by $e(\beta_X)$. The set of valuations for X is denoted B_X.

For the interpretation of event/data actions in a Σ-*edts* M w.r.t. a variable valuation β over $\mathcal{X}$, we define the family of relations $(R(M, \beta)_\lambda \subseteq \Gamma(M) \times \Gamma(M))_{\lambda \in \Lambda(\Sigma, \mathcal{X})}$ by:

- $R(M, \beta)_{e(\mathbf{any}\ X) /\!\!/ \psi(Y)} = \{(\gamma, \gamma') \in R(M)_{e(\beta_X)} \mid \beta_X \in \mathrm{B}_X, (\omega(M)(\gamma), \omega(M)(\gamma')), \beta\{X \mapsto \beta_X\} \models^{2\vec{\mathcal{D}}}_{\delta(\Sigma), \mathcal{X}} \psi\}$,
  $(\beta\{X \mapsto \beta_X\})(x) = \beta_X(x)$ for $x \in X$, $(\beta\{X \mapsto \beta_X\})(x') = \beta(x')$ for $x' \notin X$,
- $R(M, \beta)_{\lambda_1 + \lambda_2} = R(M, \beta)_{\lambda_1} \cup R(M, \beta)_{\lambda_2}$,
- $R(M, \beta)_{\lambda_1; \lambda_2} = R(M, \beta)_{\lambda_1}; R(M, \beta)_{\lambda_2}$,
- $R(M, \beta)_{\lambda^*} = (R(M, \beta)_\lambda)^*$.

For instance, $R(M, \beta)_{e(\mathbf{any}\ X) /\!\!/ \psi(Y)}$ relates all configurations for which there is a transition labelled with an event occurrence $e(\beta_X)$ such that the (open) transition predicate $\psi(Y)$ is satisfied according to $2\vec{\mathcal{D}}$ by the source and target data states under valuation $\beta\{X \mapsto \beta_X\}$. This shows that for the variables occurring in X not the valuation β is relevant but the

valuation used for the actual parameters in the event occurrence $e(\beta_X)$. Therefore, **any** X binds the variables in X for the evaluation of $\psi(Y)$.

Given an ed signature Σ and a Σ-edts M, the *satisfaction* of an event/data formula $\varrho \in \mathrm{Frm}^{\mathcal{E}_\mathrm{p}(\vec{\mathcal{D}})}(\Sigma, \mathcal{X})$ is inductively defined w.r.t. configurations $\gamma \in \Gamma(M)$ and data variable valuations β:

- $M, \gamma, \beta \models^{\mathcal{E}_\mathrm{p}(\vec{\mathcal{D}})}_{\Sigma,\mathcal{X}}$ true;
- $M, \gamma, \beta \models^{\mathcal{E}_\mathrm{p}(\vec{\mathcal{D}})}_{\Sigma,\mathcal{X}} \varphi(X)$ iff $\omega(M)(\gamma), \beta \models^{\vec{\mathcal{D}}}_{\delta(\Sigma),\mathcal{X}} \varphi(X)$;
- $M, \gamma, \beta \models^{\mathcal{E}_\mathrm{p}(\vec{\mathcal{D}})}_{\Sigma,\mathcal{X}} \langle\lambda\rangle\varrho$ iff $M, \gamma', \beta \models^{\mathcal{E}_\mathrm{p}(\vec{\mathcal{D}})}_{\Sigma,\mathcal{X}} \varrho$
  for some $\gamma' \in \Gamma(M)$ with $(\gamma, \gamma') \in R(M, \beta)_\lambda$;
- $M, \gamma, \beta \models^{\mathcal{E}_\mathrm{p}(\vec{\mathcal{D}})}_{\Sigma,\mathcal{X}} \exists x\,.\,\varrho$ iff $M, \beta\{x \mapsto \mathrm{B}_x\}, \gamma \models^{\mathcal{E}_\mathrm{p}(\vec{\mathcal{D}})}_{\Sigma,\mathcal{X}} \varrho$
  for some $\mathrm{B}_x \in |Str^{\vec{\mathcal{D}}}(\mathcal{X}(x))|$
  where $(\beta\{x \mapsto \mathrm{B}_x\})(x) = \mathrm{B}_x$ and $(\beta\{x \mapsto \mathrm{B}_x\})(x') = \beta(x')$ for $x \neq x'$;
- $M, \gamma, \beta \models^{\mathcal{E}_\mathrm{p}(\vec{\mathcal{D}})}_{\Sigma,\mathcal{X}} \neg\varrho$ iff $M, \gamma, \beta \not\models^{\mathcal{E}_\mathrm{p}(\vec{\mathcal{D}})}_{\Sigma,\mathcal{X}} \varrho$;
- $M, \gamma, \beta \models^{\mathcal{E}_\mathrm{p}(\vec{\mathcal{D}})}_{\Sigma,\mathcal{X}} \varrho_1 \vee \varrho_2$ iff $M, \gamma, \beta \models^{\mathcal{E}_\mathrm{p}(\vec{\mathcal{D}})}_{\Sigma,\mathcal{X}} \varrho_1$ or $M, \gamma, \beta \models^{\mathcal{E}_\mathrm{p}(\vec{\mathcal{D}})}_{\Sigma,\mathcal{X}} \varrho_2$.

The satisfaction of data state formulæ $\varphi(X)$ relies on the satisfaction of open formulæ in $\vec{\mathcal{D}}$. Similarly, the satisfaction of diamond formulæ $\langle e(\mathbf{any}\, X) /\!\!/ \psi(Y)\rangle\varrho$ relies on the satisfaction of open formulæ in $2\vec{\mathcal{D}}$. But note that the variables in X are bound by **any**. It expresses that it is possible to execute in the current configuration and current variable valuation all event occurrences $\hat{e}$ for which $\psi(Y)$ holds such that ϱ is satisfied in the subsequent configuration under the same data variable valuation.

As usual, for closed formulæ ϱ, i.e. for Σ-sentences of $\mathcal{E}_\mathrm{p}(\vec{\mathcal{D}})$, variable valuations are irrelevant and M satisfies ϱ, written $M \models^{\mathcal{E}_\mathrm{p}(\vec{\mathcal{D}})}_{\Sigma} \varrho$, if $M, \gamma, \beta \models^{\mathcal{E}^{\downarrow}(\vec{\mathcal{D}})} \varrho$ for all initial configurations and arbitrary valuation β. The family of satisfaction relations of $\mathcal{E}_\mathrm{p}(\vec{\mathcal{D}})$ is given by $\models^{\mathcal{E}_\mathrm{p}(\vec{\mathcal{D}})}_{\Sigma}$ for all ed signatures Σ.

The proofs to show that $\mathcal{E}_\mathrm{p}(\vec{\mathcal{D}})$-logic forms an institution are given in Appendix C.

Specifications of event/data-based systems. For any data state institution $\mathcal{D}$, the sentences of $\mathcal{E}_\mathrm{p}(\vec{\mathcal{D}})$-logic can be used to specify properties of event/data-based systems and thus to write system specifications.

A *specification* $Sp = (\Sigma, Ax)$ in $\mathcal{E}_\mathrm{p}(\vec{\mathcal{D}})$ consists of an ed-signature Σ and a set of *axioms* $Ax \subseteq \mathrm{Sen}^{\mathcal{E}_\mathrm{p}(\vec{\mathcal{D}})}(\Sigma)$. We write $\Sigma(Sp)$ for Σ and $Ax(Sp)$ for Ax. The *semantics of* Sp is given by the pair $(\Sigma(Sp), \mathrm{Mod}^{\mathcal{E}_\mathrm{p}(\vec{\mathcal{D}})}(Sp))$ where

$$\mathrm{Mod}^{\mathcal{E}_\mathrm{p}(\vec{\mathcal{D}})}(Sp) = \{M \in |Str^{\mathcal{E}_\mathrm{p}(\vec{\mathcal{D}})}(\Sigma(Sp))| \mid \forall \varrho \in Ax(Sp)\,.\, M \models^{\mathcal{E}_\mathrm{p}(\vec{\mathcal{D}})}_{\Sigma(Sp)} \varrho\}\,.$$

The $\Sigma(Sp)$-edts in $\mathrm{Mod}^{\mathcal{E}_\mathrm{p}(\vec{\mathcal{D}})}(Sp)$ are called *models* of Sp and $\mathrm{Mod}^{\mathcal{E}_\mathrm{p}(\vec{\mathcal{D}})}(Sp)$ is the *model class* of Sp.

Example 5. We complete the counter specification with parameterised events started in Sect. 1.2. The basic data state institution is ground equational logic *gEQ*; see Example 2. Our first axiom is the sentence

$$[(\mathsf{inc}(\mathbf{any}\ \mathsf{x}) /\!\!/ \mathsf{true})^*]\,\forall \mathsf{x}\,.\,(\mathsf{val} + \mathsf{x} \leq \mathsf{max} \rightarrow \langle \mathsf{inc}(\mathsf{x}) /\!\!/ \mathsf{val}' = \mathsf{val} + \mathsf{x}\rangle \mathsf{true}) \tag{Ax1}$$

already shown in Sect. 1.2. Therein the expression $\mathsf{inc}(\mathsf{x}) /\!\!/ \mathsf{val}' = \mathsf{val} + \mathsf{x}$ is a shorthand notation for the atomic event/data action $\mathsf{inc}(\mathbf{any}\,\hat{\mathsf{x}}) /\!\!/ \hat{\mathsf{x}} = \mathsf{x} \wedge \mathsf{val}' = \mathsf{val} + \mathsf{x}$ where $\hat{\mathsf{x}}$ is a variable different from x and bound by **any** (in particular, $\mathsf{inc}(\mathsf{x})$ in this expression is not meant to be an event term). Note that we cannot move the precondition in the transition predicate and use

$$\forall \mathsf{x}\,.\,\langle \mathsf{inc}(\mathbf{any}\,\hat{\mathsf{x}}) /\!\!/ \hat{\mathsf{x}} = \mathsf{x} \wedge \mathsf{val} + \mathsf{x} \leq \mathsf{max} \wedge \mathsf{val}' = \mathsf{val} + \mathsf{x}\rangle \mathsf{true})$$

instead of

$$\forall \mathsf{x}\,.\,(\mathsf{val} + \mathsf{x} \leq \mathsf{max} \rightarrow \langle \mathsf{inc}(\mathbf{any}\,\hat{\mathsf{x}}) /\!\!/ \hat{\mathsf{x}} = \mathsf{x} \wedge \mathsf{val}' = \mathsf{val} + \mathsf{x}\rangle \mathsf{true})$$

The former would require that (in the current configuration) for any value of x there must exist a transition such that the precondition $\mathsf{val} + \mathsf{x} \leq \mathsf{max}$ is satisfied, which is certainly not intended.

Next, we specify the safety property "whenever the counter value has reached the upper bound no further increment is possible" by our second axiom:

$$[(\mathsf{inc}(\mathbf{any}\ \mathsf{x}) /\!\!/ \mathsf{true})^*]\,(\mathsf{val} = \mathsf{max} \rightarrow [\mathsf{inc}(\mathbf{any}\,\mathsf{x}) /\!\!/ \mathsf{true}]\mathsf{false}) \tag{Ax2}$$

The third axiom expresses "whenever the counter value is smaller than the upper bound besides incrementing the counter value nothing else can happen."

$$[(\mathsf{inc}(\mathbf{any}\ \mathsf{x})/\!\!/\mathsf{true})^*]\,\forall \mathsf{x}\,.\,(\mathsf{val}+\mathsf{x}\leq \mathsf{max}\rightarrow[\mathsf{inc}(\mathsf{x})/\!\!/\neg(\mathsf{val}'=\mathsf{val}+\mathsf{x})]\mathsf{false}) \tag{Ax3}$$

Similarly as above, the expression $\mathsf{inc}(\mathsf{x})/\!\!/\neg(\mathsf{val}'=\mathsf{val}+\mathsf{x})$ is a shorthand notation for the atomic event/data action $\mathsf{inc}(\mathbf{any}\,\hat{\mathsf{x}})/\!\!/\hat{\mathsf{x}}=\mathsf{x}\wedge\neg(\mathsf{val}'=\mathsf{val}+\mathsf{x})$.

For these axioms the Σ-edts of Fig. 4b is a model, but Fig. 4a is not, since, e.g., d'_1 lacks an outgoing $\mathsf{inc}(1)$-transition. □

6. Conclusions

We have extended the dynamic logic variant of the event/data-based institution introduced in [8] to take into account parameterised events in the context of an arbitrary underlying data state institution which satisfies the amalgamation property. At a first glance such an extension might look like an easy task; it turned out, however, that there are several design decisions to be made in order to treat the extension independently of the underlying data state institution and at the same time being able to express subtle aspects like non-deterministic choice of arguments. We have not considered here hybrid features like in [8], but an extension to include them would indeed be straightforward since the hybrid features are not concerned by event parameters. Then the whole machinery developed in [8] for stepwise refinement from abstract property specifications to concrete recursive process structures (representable by hybrid features) would be applicable. The crucial ideas of the treatment of parameterised events in our logic were illustrated by a small example; a larger case study for validation should follow next. Moreover, we want to extend the approach by taking into account a differentiation between input and output events and thus to be able to model composition of components with synchronous or asynchronous message passing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Appendix A. Institutions

Institution gEQ of Example 2. A *many-sorted signature* $\Sigma=(S,F)$ consists of sets of sorts S and function symbols F; the latter have argument sorts and a result sort, a function symbol without arguments is a constant. A *many-sorted signature morphism* $\sigma=(\sigma_S,\sigma_F):\Sigma\rightarrow\Sigma'$ maps the sorts and function symbols of Σ to those of Σ' such that the sorting of function symbols is transferred. Many-sorted signatures and signature morphisms form the *category of signatures* $\mathbb{S}^{gEQ}$.

A Σ-*algebra* $\mathfrak{A}$ for a many-sorted signature $\Sigma=(S,F)$ consists of non-empty *carrier sets* $s^{\mathfrak{A}}$ for each sort s and *functions* $f^{\mathfrak{A}}:s_1^{\mathfrak{A}}\times\ldots\times s_n^{\mathfrak{A}}\rightarrow s^{\mathfrak{A}}$ for each $f\in F$ with argument sorts $s_1,\ldots,s_n$ and result sort s; a Σ-*algebra homomorphism* $h:\mathfrak{A}_1\rightarrow\mathfrak{A}_2$ is given by a family of functions $(h_s:s^{\mathfrak{A}_1}\rightarrow s^{\mathfrak{A}_2})_{s\in S}$ such that $h_s(f^{\mathfrak{A}_1}(a_1,\ldots,a_n))=f^{\mathfrak{A}_2}(h_{s_1}(a_1),\ldots,h_{s_n}(a_n))$ for all $f\in F$ and all $a_i\in s_i^{\mathfrak{A}_1}$. Σ-algebras and Σ-algebra homomorphisms form the category $Str^{gEQ}(\Sigma)$. For a many-sorted signature morphism $\sigma=(\sigma_S,\sigma_F):\Sigma\rightarrow\Sigma'$ the *reduct* of a Σ'-algebra $\mathfrak{A}'$ is the Σ-algebra $\mathfrak{A}'|\sigma$ with $s^{\mathfrak{A}'|\sigma}=\sigma_S(s)^{\mathfrak{A}'}$ for each $s\in S$ and $f^{\mathfrak{A}'|\sigma}=\sigma_F(f)^{\mathfrak{A}'}$ for each $f\in F$; the *reduct* of a Σ'-algebra homomorphism $h':\mathfrak{A}'_1\rightarrow\mathfrak{A}'_2$ is the Σ-algebra homomorphism $h'|\sigma:\mathfrak{A}'_1|\sigma\rightarrow\mathfrak{A}'_2|\sigma$ with $(h'|\sigma)_s=h'_{\sigma_S(s)}$ for each $s\in S$. The *structures functor* $Str^{gEQ}:(\mathbb{S}^{gEQ})^{\mathrm{op}}\rightarrow\mathrm{Cat}$ maps Σ to the category $Str^{gEQ}(\Sigma)$ and $\sigma:\Sigma\rightarrow\Sigma'$ to the functor $-|\sigma$.

The S-indexed family of *terms* $\mathscr{T}(\Sigma)$ over the many-sorted signature $\Sigma=(S,F)$ is inductively given by $f(t_1,\ldots,t_n)\in\mathscr{T}(\Sigma)_s$ for $f\in F$ with arguments sorts $s_1,\ldots,s_n$ and result sort s and $t_i\in\mathscr{T}(\Sigma)_{s_i}$ (for constants, the parentheses are omitted). The set of *sentences* $\mathrm{Sen}^{gEQ}(\Sigma)$ is given by the grammar

$$\varphi ::= t_1=t_2 \mid \mathsf{true} \mid \neg\varphi \mid \varphi_1\vee\varphi_2\,,$$

where $t_1,t_2\in\mathscr{T}(\Sigma)_s$ for some $s\in S$. Term translation $\mathscr{T}(\sigma)=(\mathscr{T}(\sigma)_s:\mathscr{T}(\Sigma)_s\rightarrow\mathscr{T}(\Sigma')_s)_{s\in S}:\mathscr{T}(\Sigma)\rightarrow\mathscr{T}(\Sigma')$ and sentence translation $\mathrm{Sen}^{gEQ}(\sigma):\mathrm{Sen}^{gEQ}(\Sigma)\rightarrow\mathrm{Sen}^{gEQ}(\Sigma')$ along a signature morphism $\sigma:\Sigma\rightarrow\Sigma'$ preserve the term and sentence structure: $\mathscr{T}(\sigma)_s(f(t_1,\ldots,t_n))=\sigma_F(f)(\mathscr{T}(\sigma)_{s_1}(t_1),\ldots,\mathscr{T}(\sigma)_{s_n}(t_n))$ and $\mathrm{Sen}^{gEQ}(\sigma)(t_1=t_2)=(\mathscr{T}(\sigma)_s(t_1)=\mathscr{T}(\sigma)_s(t_2))$, etc. The *sentence functor* $\mathrm{Sen}^{gEQ}:\mathbb{S}^{gEQ}\rightarrow\mathrm{Set}$ maps Σ to the sentences over Σ and $\sigma:\Sigma\rightarrow\Sigma'$ to the sentence translation along σ.

For a Σ-algebra $\mathfrak{A}$, the *term evaluation* $(-)^{\mathfrak{A}}=((-)^{\mathfrak{A}}_s:\mathscr{T}(\Sigma)_s\rightarrow s^{\mathfrak{A}})_{s\in S(\Sigma)}$ is inductively given by $(f(t_1,\ldots,t_n))^{\mathfrak{A}}_s=f^{\mathfrak{A}}((t_1)^{\mathfrak{A}}_{s_1},\ldots,(t_n)^{\mathfrak{A}}_{s_n})$. The *satisfaction relation* $\mathfrak{A}\models^{gEQ}_{\Sigma}\varphi$ for a sentence $\varphi\in\mathrm{Sen}^{gEQ}(\Sigma)$ is inductively given by

- $\mathfrak{A}\models^{gEQ}_{\Sigma}t_1=t_2$ iff $(t_1)^{\mathfrak{A}}=(t_2)^{\mathfrak{A}}$;
- $\mathfrak{A}\models^{gEQ}_{\Sigma}\mathsf{true}$;
- $\mathfrak{A}\models^{gEQ}_{\Sigma}\neg\varphi$ iff not $\mathfrak{A}\models^{gEQ}_{\Sigma}\varphi$;
- $\mathfrak{A}\models^{gEQ}_{\Sigma}\varphi_1\vee\varphi_2$ iff $\mathfrak{A}\models^{gEQ}_{\Sigma}\varphi_1$ or $\mathfrak{A}\models^{gEQ}_{\Sigma}\varphi_2$.

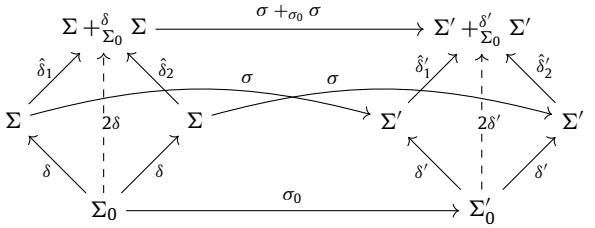
Appendix B. (2-)data state institutions and open formulæ

Data state institutions. Let $\mathcal{D}$ be an institution enjoying the amalgamation property. A *data state institution* $\vec{\mathcal{D}} = (\mathbb{S}^{\vec{\mathcal{D}}}, Str^{\vec{\mathcal{D}}}, \mathrm{Sen}^{\vec{\mathcal{D}}}, \models^{\vec{\mathcal{D}}})$ over $\mathcal{D}$ consists of

- a signature category $\mathbb{S}^{\vec{\mathcal{D}}}$ which is a subcategory of the arrow category $(\mathbb{S}^{\mathcal{D}})^{\to}$, where the objects are morphisms $\delta : \Sigma_0 \to \Sigma$ in $\mathcal{D}$, and which is closed under pushouts, i.e., if $(\Sigma_1 +^{\delta_1,\delta_2}_{\Sigma_0} \Sigma_2, (\delta_{3-i}{}^{\delta_i} : \Sigma_i \to \Sigma_1 +^{\delta_1,\delta_2}_{\Sigma_0} \Sigma_2)_{1\le i\le 2})$ is a pushout of $\delta_1 : \Sigma_0 \to \Sigma_1$ and $\delta_2 : \Sigma_0 \to \Sigma_2$ in $\mathbb{S}^{\mathcal{D}}$ and $\delta_1, \delta_2 \in |\mathbb{S}^{\vec{\mathcal{D}}}|$, then $\delta_1; \delta_2{}^{\delta_1} = \delta_2; \delta_1{}^{\delta_2} \in |\mathbb{S}^{\vec{\mathcal{D}}}|$.
- a structures functor $Str^{\vec{\mathcal{D}}} : (\mathbb{S}^{\vec{\mathcal{D}}})^{\mathrm{op}} \to \mathrm{Cat}$ which yields
  1. for every $\vec{\mathcal{D}}$-signature $\delta : \Sigma_0 \to \Sigma \in |\mathbb{S}^{\vec{\mathcal{D}}}|$ a non-empty subcategory of $Str^{\mathcal{D}}(\Sigma)$ such that there is an $M_\delta \in |Str^{\mathcal{D}}(\Sigma_0)|$ with $Str^{\mathcal{D}}(\delta)(M) = M_\delta$ for all $M \in |Str^{\vec{\mathcal{D}}}(\delta)|$ and $Str^{\mathcal{D}}(\delta)(\mu) = 1_{M_\delta}$ for all $\mu : M_1 \to M_2$ in $Str^{\vec{\mathcal{D}}}(\delta)$;
  2. for every signature morphism $(\sigma_0, \sigma) : (\delta : \Sigma_0 \to \Sigma) \to (\delta' : \Sigma'_0 \to \Sigma')$ in $\mathbb{S}^{\vec{\mathcal{D}}}$ a reduct functor $Str^{\vec{\mathcal{D}}}(\sigma_0, \sigma) : Str^{\vec{\mathcal{D}}}(\delta') \to Str^{\vec{\mathcal{D}}}(\delta)$ such that for every $M' \in |Str^{\vec{\mathcal{D}}}(\delta')| \subseteq |Str^{\mathcal{D}}(\Sigma')|$ and every $\mu' : M'_1 \to M'_2$ in $Str^{\vec{\mathcal{D}}}(\delta')$: $Str^{\vec{\mathcal{D}}}(\sigma_0, \sigma)(M') = Str^{\mathcal{D}}(\sigma)(M')$ and $Str^{\vec{\mathcal{D}}}(\sigma_0, \sigma)(\mu') = Str^{\mathcal{D}}(\sigma)(\mu')$;
- the sentence functor $\mathrm{Sen}^{\vec{\mathcal{D}}} : \mathbb{S}^{\vec{\mathcal{D}}} \to \mathrm{Set}$ with $\mathrm{Sen}^{\vec{\mathcal{D}}}(\delta) = \mathrm{Sen}^{\mathcal{D}}(\Sigma)$ for each signature $\delta : \Sigma_0 \to \Sigma \in |\mathbb{S}^{\vec{\mathcal{D}}}|$ and $\mathrm{Sen}^{\vec{\mathcal{D}}}(\sigma_0, \sigma) = \mathrm{Sen}^{\mathcal{D}}(\sigma)$ for each $\vec{\mathcal{D}}$-signature morphism $(\sigma_0, \sigma) : (\delta : \Sigma_0 \to \Sigma) \to (\delta' : \Sigma'_0 \to \Sigma')$;
- the satisfaction relations $(\models^{\vec{\mathcal{D}}}_\delta \subseteq |Str^{\vec{\mathcal{D}}}(\delta)| \times \mathrm{Sen}^{\vec{\mathcal{D}}}(\delta))_{\delta \in |\mathbb{S}^{\vec{\mathcal{D}}}|}$ with $M \models^{\vec{\mathcal{D}}}_\delta \varphi$ if, and only if, $M \models^{\mathcal{D}}_\Sigma \varphi$ for each $\delta : \Sigma_0 \to \Sigma$, $M \in |Str^{\vec{\mathcal{D}}}(\delta)|$, and $\varphi \in \mathrm{Sen}^{\vec{\mathcal{D}}}(\delta)$.

2-data state institutions. Let $\vec{\mathcal{D}} = (\mathbb{S}^{\vec{\mathcal{D}}}, Str^{\vec{\mathcal{D}}}, \mathrm{Sen}^{\vec{\mathcal{D}}}, \models^{\vec{\mathcal{D}}})$ be a data state institution over $\mathcal{D}$. The *2-data state institution* $2\vec{\mathcal{D}} = (\mathbb{S}^{2\vec{\mathcal{D}}}, Str^{2\vec{\mathcal{D}}}, \mathrm{Sen}^{2\vec{\mathcal{D}}}, \models^{2\vec{\mathcal{D}}})$ over $\vec{\mathcal{D}}$ consists of

- the signature category $\mathbb{S}^{2\vec{\mathcal{D}}} = \mathbb{S}^{\vec{\mathcal{D}}}$;
- the structures functor $Str^{2\vec{\mathcal{D}}} : (\mathbb{S}^{2\vec{\mathcal{D}}})^{\mathrm{op}} \to \mathrm{Cat}$ mapping each $\delta \in |\mathbb{S}^{2\vec{\mathcal{D}}}| = |\mathbb{S}^{\vec{\mathcal{D}}}|$ to the cartesian product of categories $Str^{\vec{\mathcal{D}}}(\delta) \times Str^{\vec{\mathcal{D}}}(\delta)$ and each signature morphism $(\sigma_0, \sigma) : \delta \to \delta'$ in $\mathbb{S}^{2\vec{\mathcal{D}}} = \mathbb{S}^{\vec{\mathcal{D}}}$ to the cartesian product of functors $Str^{\vec{\mathcal{D}}}(\sigma_0, \sigma) \times Str^{\vec{\mathcal{D}}}(\sigma_0, \sigma) : Str^{\vec{\mathcal{D}}}(\delta') \times Str^{\vec{\mathcal{D}}}(\delta') \to Str^{\vec{\mathcal{D}}}(\delta) \times Str^{\vec{\mathcal{D}}}(\delta)$;
- the sentence functor $\mathrm{Sen}^{2\vec{\mathcal{D}}} : \mathbb{S}^{2\vec{\mathcal{D}}} \to \mathrm{Set}$ defined as follows: For each signature $\delta : \Sigma_0 \to \Sigma \in |\mathbb{S}^{2\vec{\mathcal{D}}}| = |\mathbb{S}^{\vec{\mathcal{D}}}|$, we assume given a specifically chosen pushout of δ with itself in $\mathbb{S}^{\mathcal{D}}$, denoted by $(\Sigma +^{\delta}_{\Sigma_0} \Sigma, (\hat{\delta}_i : \Sigma \to \Sigma +^{\delta}_{\Sigma_0} \Sigma)_{1\le i\le 2})$, as illustrated in the diagram below at the left hand side. Then the set $\mathrm{Sen}^{2\vec{\mathcal{D}}}(\delta)$ of *$2\vec{\mathcal{D}}$-sentences* is $\mathrm{Sen}^{\vec{\mathcal{D}}}(2\delta) = \mathrm{Sen}^{\mathcal{D}}(\Sigma +^{\delta}_{\Sigma_0} \Sigma)$. For each signature morphism $(\sigma_0, \sigma) : (\delta : \Sigma_0 \to \Sigma) \to (\delta' : \Sigma'_0 \to \Sigma')$ in $\mathbb{S}^{2\vec{\mathcal{D}}} = \mathbb{S}^{\vec{\mathcal{D}}}$ the sentence translation function $\mathrm{Sen}^{2\vec{\mathcal{D}}}(\sigma) : \mathrm{Sen}^{2\vec{\mathcal{D}}}(\delta) \to \mathrm{Sen}^{2\vec{\mathcal{D}}}(\delta')$ is $\mathrm{Sen}^{\vec{\mathcal{D}}}(\sigma_0, \sigma +_{\sigma_0} \sigma) = \mathrm{Sen}^{\mathcal{D}}(\sigma +_{\sigma_0} \sigma)$ where $\sigma +_{\sigma_0} \sigma : \Sigma +^{\delta}_{\Sigma_0} \Sigma \to \Sigma' +^{\delta'}_{\Sigma'_0} \Sigma'$ is the unique signature morphism in $\mathbb{S}^{\mathcal{D}}$ such that the following diagram commutes:



- the satisfaction relations $(\models^{2\vec{\mathcal{D}}}_\delta \subseteq |Str^{2\vec{\mathcal{D}}}(\delta)| \times \mathrm{Sen}^{2\vec{\mathcal{D}}}(\delta))_{\delta \in |\mathbb{S}^{\vec{\mathcal{D}}}|}$ defined by $(M_1, M_2) \models^{2\vec{\mathcal{D}}}_\delta \psi$ if, and only if, $M_1 \times_\delta M_2 \models^{\mathcal{D}}_{\Sigma +^{\delta}_{\Sigma_0} \Sigma} \psi$ for $\delta : \Sigma_0 \to \Sigma \in |\mathbb{S}^{2\vec{\mathcal{D}}}|$, $(M_1, M_2) \in |Str^{2\vec{\mathcal{D}}}(\delta)| = |Str^{\vec{\mathcal{D}}}(\delta)| \times |Str^{\vec{\mathcal{D}}}(\delta)|$, and $\psi \in \mathrm{Sen}^{2\vec{\mathcal{D}}}(\delta) = \mathrm{Sen}^{\mathcal{D}}(\Sigma +^{\delta}_{\Sigma_0} \Sigma)$.

Variables. *Variables* $\mathcal{X}$ for a $\delta : \Sigma_0 \to \Sigma \in |\mathbb{S}^{\vec{\mathcal{D}}}|$ consist of a countable set $|\mathcal{X}|$ of names with a signature $\mathcal{X}(x) : \Sigma_0 \to \Sigma_x \in |\mathbb{S}^{\vec{\mathcal{D}}}|$ attached to each $x \in |\mathcal{X}|$. For every finite subset $X \subseteq |\mathcal{X}|$ we assume a canonically chosen colimit, i.e., iterated pushout, signature Σ_X in $\mathcal{D}$ with signature $\mathcal{X}(X) : \Sigma_0 \to \Sigma_X \in |\mathbb{S}^{\vec{\mathcal{D}}}|$ such that $\Sigma_\emptyset = \Sigma_0$ and $\mathcal{X}(\emptyset) = 1_{\Sigma_0}$. The variables $\mathcal{X}$

are *translated* along a signature morphism $(\sigma_0, \sigma) : (\delta : \Sigma_0 \to \Sigma) \to (\delta' : \Sigma_0' \to \Sigma')$ in $\mathbb{S}^{\vec{\mathcal{D}}}$ to the variables $\mathcal{X}^{\sigma_0}$ for δ' given by $|\mathcal{X}^{\sigma_0}| = |\mathcal{X}|$ and $\mathcal{X}^{\sigma_0}(x) = \mathcal{X}(x)^{\sigma_0} : \Sigma_0' \to \Sigma_0' +^{\sigma_0, \mathcal{X}(x)}_{\Sigma_0} \Sigma_x$. By amalgamation in $\mathcal{D}$ the following holds:

Lemma 1. *Let $(\sigma_0, \sigma) : \delta \to \delta'$ in $\mathbb{S}^{\vec{\mathcal{D}}}$. Then $M'_x \in |Str^{\vec{\mathcal{D}}}(\mathcal{X}^{\sigma_0}(x))|$ if, and only if, there is a $M_x \in |Str^{\vec{\mathcal{D}}}(\mathcal{X}(x))|$ with $M'_x = M_{\delta'} \times_{\sigma_0, \mathcal{X}(x)} M_x$.*

A *valuation* for $\mathcal{X}$ over δ is given by a map β from $|\mathcal{X}|$ to the structures of $\vec{\mathcal{D}}$ such that $\beta(x) \in |Str^{\vec{\mathcal{D}}}(\mathcal{X}(x))|$. For every finite subset $X \subseteq |\mathcal{X}|$, the structure $\beta(X)$ is the limit, i.e., iterated amalgamation, of $(\beta(x))_{x \in X}$ in $\vec{\mathcal{D}}$ such that $\beta(\emptyset) = M_\delta$. The *reduct* of a valuation β' for $\mathcal{X}^{\sigma_0}$ over δ' along $(\sigma_0, \sigma) : \delta \to \delta'$ is given by $Str^{\vec{\mathcal{D}}}(\sigma_0, \mathcal{X})(\beta')(x) = Str^{\mathcal{D}}(\sigma_0{}^{\mathcal{X}(x)})(\beta'(x))$ for $x \in |\mathcal{X}|$.

Open formulæ. The open $\vec{\mathcal{D}}$*-formulæ* $\mathrm{Frm}^{\vec{\mathcal{D}}}(\delta, \mathcal{X})$ and the open $2\vec{\mathcal{D}}$*-formulæ* $\mathrm{Frm}^{2\vec{\mathcal{D}}}(\delta, \mathcal{X})$ over δ and $\mathcal{X}$ are given by

$$\mathrm{Frm}^{\vec{\mathcal{D}}}(\delta, \mathcal{X}) = \{\varphi(X) \mid X \subseteq |\mathcal{X}| \text{ finite}, \varphi \in \mathrm{Sen}^{\mathcal{D}}(\Sigma +^{\delta, \mathcal{X}(X)}_{\Sigma_0} \Sigma_X)\} ,$$

$$\mathrm{Frm}^{2\vec{\mathcal{D}}}(\delta, \mathcal{X}) = \mathrm{Frm}^{\vec{\mathcal{D}}}(2\delta, \mathcal{X}) .$$

The translation of formulæ along $(\sigma_0, \sigma) : (\delta : \Sigma_0 \to \Sigma) \to (\delta' : \Sigma_0' \to \Sigma')$ is given by the functions $\mathrm{Frm}^{\vec{\mathcal{D}}}(\sigma_0, \sigma, \mathcal{X}) : \mathrm{Frm}^{\vec{\mathcal{D}}}(\delta, \mathcal{X}) \to \mathrm{Frm}^{\vec{\mathcal{D}}}(\delta', \mathcal{X}^{\sigma_0})$ and $\mathrm{Frm}^{2\vec{\mathcal{D}}}(\sigma_0, \sigma, \mathcal{X}) : \mathrm{Frm}^{2\vec{\mathcal{D}}}(\delta, \mathcal{X}) \to \mathrm{Frm}^{2\vec{\mathcal{D}}}(\delta', \mathcal{X}^{\sigma_0})$ defined by

$$\mathrm{Frm}^{\vec{\mathcal{D}}}(\sigma_0, \sigma, \mathcal{X})(\varphi(X)) = (\mathrm{Sen}^{\mathcal{D}}(\sigma +_{\sigma_0} -^{\sigma_0})(\varphi))(X)$$

for the unique $\sigma +_{\sigma_0} -^{\sigma_0} : \Sigma +^{\delta, \mathcal{X}(X)}_{\Sigma_0} \Sigma_X \to \Sigma' +^{\delta', \mathcal{X}^{\sigma_0}(X)}_{\Sigma_0'} \Sigma_{X^{\sigma_0}}$ and

$$\mathrm{Frm}^{2\vec{\mathcal{D}}}(\sigma_0, \sigma, \mathcal{X})(\psi(X)) = \mathrm{Frm}^{\vec{\mathcal{D}}}(\sigma_0, \sigma +_{\sigma_0} \sigma, \mathcal{X})(\psi(X)) .$$

The *satisfaction* of a $\vec{\mathcal{D}}$-formula $\varphi(X) \in \mathrm{Frm}^{\vec{\mathcal{D}}}(\delta, \mathcal{X})$ over a structure $M \in |Str^{\vec{\mathcal{D}}}(\delta)|$ and a valuation β for $\mathcal{X}$ is given by

$$M, \beta \models^{\vec{\mathcal{D}}}_{\delta, \mathcal{X}} \varphi(X) \iff M \times_{\delta, \mathcal{X}(X)} \beta(X) \models^{\mathcal{D}}_{\Sigma +^{\delta, \mathcal{X}(X)}_{\Sigma_0} \Sigma_X} \varphi ;$$

likewise, the *satisfaction* of a $2\vec{\mathcal{D}}$-formula $\psi(Y) \in \mathrm{Frm}^{2\vec{\mathcal{D}}}(\delta, \mathcal{X})$ over a structure $(M_1, M_2) \in |Str^{2\vec{\mathcal{D}}}(\delta)|$ and a valuation β for $\mathcal{X}$ is defined by

$$(M_1, M_2), \beta \models^{2\vec{\mathcal{D}}}_{\delta, \mathcal{X}} \psi(Y) \iff M_1 \times_\delta M_2, \beta \models^{\vec{\mathcal{D}}}_{2\delta, \mathcal{X}} \psi(Y) .$$

Lemma 2. *Let $(\sigma_0, \sigma) : \delta \to \delta'$ in $\mathbb{S}^{\vec{\mathcal{D}}}$, $\mathcal{X}$ variables for δ, and β' a valuation for $\mathcal{X}^{\sigma_0}$ over δ'.*

1. *For each $M' \in |Str^{\vec{\mathcal{D}}}(\delta')|$ and $\varphi(X) \in \mathrm{Frm}^{\vec{\mathcal{D}}}(\delta, \mathcal{X})$ it holds that*

$$M', \beta' \models^{\vec{\mathcal{D}}}_{\delta', \mathcal{X}^{\sigma_0}} \mathrm{Frm}^{\vec{\mathcal{D}}}(\sigma_0, \sigma, \mathcal{X})(\varphi(X)) \iff Str^{\vec{\mathcal{D}}}(\sigma_0, \sigma)(M'), Str^{\vec{\mathcal{D}}}(\sigma_0, \mathcal{X})(\beta') \models^{\vec{\mathcal{D}}}_{\delta, \mathcal{X}} \varphi(X) .$$

2. *For each $(M'_1, M'_2) \in |Str^{2\vec{\mathcal{D}}}(\delta')|$ and $\psi(Y) \in \mathrm{Frm}^{2\vec{\mathcal{D}}}(\delta, \mathcal{X})$ it holds that*

$$(M'_1, M'_2), \beta' \models^{2\vec{\mathcal{D}}}_{\delta', \mathcal{X}^{\sigma_0}} \mathrm{Frm}^{2\vec{\mathcal{D}}}(\sigma_0, \sigma, \mathcal{X})(\psi(Y)) \iff Str^{2\vec{\mathcal{D}}}(\sigma_0, \sigma)(M'_1, M'_2), Str^{\vec{\mathcal{D}}}(\sigma_0, \mathcal{X})(\beta') \models^{2\vec{\mathcal{D}}}_{\delta, \mathcal{X}} \psi(Y) .$$

Appendix C. Event/data institution $\mathcal{E}_{\mathbf{p}}(\vec{\mathcal{D}})$

C.1. Event/data signatures

Parameterised events. For a $\delta : \Delta_0 \to \Delta \in |\mathbb{S}^{\vec{\mathcal{D}}}|$, an *event signature* $E = (|E|, \xi)$ over δ consists of a set of *events names* $|E|$ and a *parameter types* map $\xi : |E| \to |\mathbb{S}^{\vec{\mathcal{D}}}|^*$ with $\xi(e)_i = \Delta_0 \to \Delta_{e,i}$ for $1 \le i \le |\xi(e)|$; we write $\xi(E)$ for ξ. A *parameterised event* is given by an $e \in |E|$ and its parameter types $\xi(e)$. For a $(\vartheta_0, \vartheta) : \delta \to (\delta' : \Delta_0' \to \Delta')$ in $\mathbb{S}^{\vec{\mathcal{D}}}$ and event signatures E over δ and E' over δ', an *event signature morphism* $\eta : E \to E'$ for (ϑ_0, ϑ) is given by a function $\eta : |E| \to |E'|$ such that $\xi(E')(\eta(e))_i = (\xi(E)(e)_i)^{\vartheta_0} : \Delta_0' \to \Delta_0' +^{\vartheta_0, \xi(E)(e)_i}_{\Delta_0} \Delta_{e,i}$ where $\xi(E)(e)_i : \Delta_0 \to \Delta_{e,i}$. An *event occurrence* for an $e \in |E|$ with $\xi(E)(e) = \xi_1, \ldots, \xi_n$ is a pair $e(\mathrm{B})$ with $\mathrm{B} = (\mathrm{B}_i \in |Str^{\vec{\mathcal{D}}}(\xi_i)|)_{1 \le i \le n}$. The set of event occurrences for an event signature E is denoted by $\hat{E}$.

Event/data signatures. An *event/data signature* $\Sigma = (E, \delta : \Delta_0 \to \Delta)$ consists of a data signature $\delta : \Delta_0 \to \Delta$ in $|\mathbb{S}^{\vec{\mathcal{D}}}|$ and an event signature E over δ. We write $E(\Sigma)$ for E and $\delta(\Sigma)$ for δ. The event occurrences $\hat{E}$ over Σ are denoted by $\hat{E}(\Sigma)$.

An *event/data signature morphism* $\sigma = (\eta, \vartheta)$ from Σ to Σ' is given by a data signature morphism $\vartheta : \delta(\Sigma) \to \delta(\Sigma')$ in $\mathbb{S}^{\vec{\mathcal{D}}}$ and an event signature morphism $\eta : E(\Sigma) \to E(\Sigma')$ for ϑ. We write $E(\sigma)$ for η and $\delta(\sigma)$ for ϑ.

Event/data signatures and their morphisms form a category denoted by $\mathbb{S}^{\mathcal{E}_{\mathrm{p}}(\vec{\mathcal{D}})}$.

C.2. Event/data structures

A Σ-*event/data transition system* $M = (\Gamma, R, \Gamma_0, \omega)$ over an event/data signature Σ consists of

- a set of *configurations* Γ;
- a family of *transition relations* $R = (R_{\hat{e}} \subseteq \Gamma \times \Gamma)_{\hat{e} \in \hat{E}(\Sigma)}$;
- a non-empty set of *initial configurations* $\Gamma_0 \subseteq \Gamma$; and
- a *data state labelling* $\omega : \Gamma \to |Str^{\vec{\mathcal{D}}}(\delta(\Sigma))|$,

such that Γ is *reachable* via R, i.e., for all $\gamma \in \Gamma$ there are $\gamma_0 \in \Gamma_0$, $n \geq 0$, $\hat{e}_1, \ldots, \hat{e}_n \in \hat{E}(\Sigma)$, and $(\gamma_i, \gamma_{i+1}) \in R_{\hat{e}_{i+1}}$ for all $0 \leq i < n$ with $\gamma_n = \gamma$. We write $\Gamma(M)$ for Γ, $R(M)$ for R, $\Gamma_0(M)$ for Γ_0, and $\omega(M)$ for ω.

Let $\sigma : \Sigma \to \Sigma'$ be an event/data signature morphism and M' a Σ'-event/data structure. The σ-*reduct* of M' is the Σ-event/data structure $M'|\sigma$ such that

- $\Gamma(M'|\sigma)$ and $R(M'|\sigma) = (R(M'|\sigma)_{\hat{e}})_{\hat{e} \in \hat{E}(\Sigma)}$ are defined inductively with base $\Gamma_0(M') \subseteq \Gamma(M'|\sigma)$ and step cases for all $\gamma', \gamma'' \in \Gamma(M')$, $e \in E(\Sigma)$, if $\gamma' \in \Gamma(M'|\sigma)$ and $(\gamma', \gamma'') \in R(M')_{E(\sigma)(e)(\mathrm{B}')}$, then $\gamma'' \in \Gamma(M'|\sigma)$ and $(\gamma', \gamma'') \in R(M'|\sigma)_{e(\mathrm{B}'|\sigma)}$ where $(\mathrm{B}'|\sigma)_i = Str^{\mathcal{D}}(\delta(\sigma)_0{}^{\xi(e)_i})(\mathrm{B}'_i)$ for $1 \leq i \leq |\xi(e)|$;
- $\Gamma_0(M'|\sigma) = \Gamma_0(M')$; and
- $\omega(M'|\sigma)(\gamma') = Str^{\vec{\mathcal{D}}}(\delta(\sigma))(\omega(M')(\gamma'))$ for all $\gamma' \in \Gamma(M'|\sigma)$.

For each $\Sigma \in |\mathbb{S}^{\mathcal{E}_{\mathrm{p}}(\vec{\mathcal{D}})}|$, we denote the discrete category (class) of all Σ-event/data structures by $Str^{\mathcal{E}_{\mathrm{p}}(\vec{\mathcal{D}})}(\Sigma)$; and for each $\sigma : \Sigma \to \Sigma'$ in $\mathbb{S}^{\mathcal{E}_{\mathrm{p}}(\vec{\mathcal{D}})}$ the reduct functor (function) $-|\sigma : Str^{\mathcal{E}_{\mathrm{p}}(\vec{\mathcal{D}})}(\Sigma') \to Str^{\mathcal{E}_{\mathrm{p}}(\vec{\mathcal{D}})}(\Sigma)$ by $Str^{\mathcal{E}_{\mathrm{p}}(\vec{\mathcal{D}})}(\sigma)$, such that $Str^{\mathcal{E}_{\mathrm{p}}(\vec{\mathcal{D}})} : (\mathbb{S}^{\mathcal{E}_{\mathrm{p}}(\vec{\mathcal{D}})})^{\mathrm{op}} \to \mathrm{Cat}$ forms a functor.

C.3. Event/data formulæ and sentences

Event/data actions. For an event/data signature Σ and variables $\mathcal{X}$ for $\delta(\Sigma)$, the *event/data actions* $\Lambda(\Sigma, \mathcal{X})$ are given by

$$\lambda ::= e(\mathbf{any}\ X) /\!\!/ \psi(Y) \mid \lambda_1 + \lambda_2 \mid \lambda_1; \lambda_2 \mid \lambda^*$$

where $e \in |E(\Sigma)|$ and $X = x_1, \ldots, x_n \subseteq |\mathcal{X}|$ such that $\mathcal{X}(x_i) = \xi(E(\Sigma))(e)_i$ for $1 \leq i \leq |\xi(E(\Sigma))(e)|$, and $\psi(Y) \in \mathrm{Frm}^{2\vec{\mathcal{D}}}(\delta, \mathcal{X})$; we call an expression $e(\mathbf{any}\ X)$ an *event term*. The *free variables* $\mathrm{fvar}(\lambda)$ of an action λ are defined inductively by $\mathrm{fvar}(e(\mathbf{any}\ X) /\!\!/ \psi(Y)) = Y \setminus X$, $\mathrm{fvar}(\lambda_1 + \lambda_2) = \mathrm{fvar}(\lambda_1) \cup \mathrm{fvar}(\lambda_2) = \mathrm{fvar}(\lambda_1; \lambda_2)$, and $\mathrm{fvar}(\lambda^*) = \mathrm{fvar}(\lambda)$.

The *event/data action translation* $\Lambda(\sigma, \mathcal{X}) : \Lambda(\Sigma, \mathcal{X}) \to \Lambda(\Sigma', \mathcal{X}^{\delta(\sigma)_0})$ along an event/data signature morphism $\sigma : \Sigma \to \Sigma'$ is recursively given by

- $\Lambda(\sigma, \mathcal{X})(e(\mathbf{any}\ X) /\!\!/ \psi(Y)) = E(\sigma)(e)(\mathbf{any}\ X) /\!\!/ \mathrm{Frm}^{2\vec{\mathcal{D}}}(\delta(\sigma), \mathcal{X})(\psi(Y))$,
- $\Lambda(\sigma, \mathcal{X})(\lambda_1 + \lambda_2) = \Lambda(\sigma, \mathcal{X})(\lambda_1) + \Lambda(\sigma, \mathcal{X})(\lambda_2)$,
- $\Lambda(\sigma, \mathcal{X})(\lambda_1; \lambda_2) = \Lambda(\sigma, \mathcal{X})(\lambda_1); \Lambda(\sigma, \mathcal{X})(\lambda_2)$,
- $\Lambda(\sigma, \mathcal{X})(\lambda^*) = \Lambda(\sigma, \mathcal{X})(\lambda)^*$.

Event/data formulæ. For an event/data signature Σ and variables $\mathcal{X}$ for $\delta(\Sigma)$, the *event/data formulæ* $\mathrm{Frm}^{\mathcal{E}_{\mathrm{p}}(\vec{\mathcal{D}})}(\Sigma, \mathcal{X})$ are given by

$$\varrho ::= \mathrm{true} \mid \varphi(X) \mid \langle\lambda\rangle\varrho \mid \exists x . \varrho \mid \neg\varrho \mid \varrho_1 \vee \varrho_2$$

where $\varphi(X) \in \mathrm{Frm}^{\vec{\mathcal{D}}}(\delta(\Sigma), \mathcal{X})$, $\lambda \in \Lambda(\Sigma, \mathcal{X})$, and $x \in |\mathcal{X}|$. The *free variables* $\mathrm{fvar}(\varrho)$ of a formula ϱ are defined inductively by $\mathrm{fvar}(\mathrm{true}) = \emptyset$, $\mathrm{fvar}(\varphi(X)) = X$, $\mathrm{fvar}(\langle\lambda\rangle\varrho) = \mathrm{fvar}(\lambda) \cup \mathrm{fvar}(\varrho)$, $\mathrm{fvar}(\exists x . \varrho) = \mathrm{fvar}(\varrho) \setminus \{x\}$, $\mathrm{fvar}(\neg\varrho) = \mathrm{fvar}(\varrho)$, and $\mathrm{fvar}(\varrho_1 \vee \varrho_2) = \mathrm{fvar}(\varrho_1) \cup \mathrm{fvar}(\varrho_2)$.

The *event/data formulæ translation* $\mathrm{Frm}^{\mathcal{E}_{\mathrm{p}}(\vec{\mathcal{D}})}(\sigma, \mathcal{X}) : \mathrm{Frm}^{\mathcal{E}_{\mathrm{p}}(\vec{\mathcal{D}})}(\Sigma, \mathcal{X}) \to \mathrm{Frm}^{\mathcal{E}_{\mathrm{p}}(\vec{\mathcal{D}})}(\Sigma', \mathcal{X})$ along $\sigma : \Sigma \to \Sigma'$ is recursively given by

- $\mathrm{Frm}^{\mathcal{E}_{\mathrm{p}}(\vec{\mathcal{D}})}(\sigma, \mathcal{X})(\varphi(X)) = \mathrm{Frm}^{\vec{\mathcal{D}}}(\delta(\sigma), \mathcal{X})(\varphi(X))$,

- $\mathrm{Frm}^{\mathcal{E}_{\mathrm{p}}(\vec{\mathcal{D}})}(\sigma,\mathcal{X})(\langle\lambda\rangle\varrho) = \langle\Lambda(\sigma)(\lambda,\mathcal{X})\rangle\mathrm{Frm}^{\mathcal{E}_{\mathrm{p}}(\vec{\mathcal{D}})}(\sigma,\mathcal{X})(\varrho)$,
- $\mathrm{Frm}^{\mathcal{E}_{\mathrm{p}}(\vec{\mathcal{D}})}(\sigma,\mathcal{X})(\exists x.\,\varrho) = \exists x.\,\mathrm{Frm}^{\mathcal{E}_{\mathrm{p}}(\vec{\mathcal{D}})}(\sigma,\mathcal{X})(\varrho)$,
- $\mathrm{Frm}^{\mathcal{E}_{\mathrm{p}}(\vec{\mathcal{D}})}(\sigma,\mathcal{X})(\neg\varrho) = \neg\mathrm{Frm}^{\mathcal{E}_{\mathrm{p}}(\vec{\mathcal{D}})}(\sigma,\mathcal{X})(\varrho)$,
- $\mathrm{Frm}^{\mathcal{E}_{\mathrm{p}}(\vec{\mathcal{D}})}(\sigma,\mathcal{X})(\varrho_1\vee\varrho_2) = \mathrm{Frm}^{\mathcal{E}_{\mathrm{p}}(\vec{\mathcal{D}})}(\sigma)(\varrho_1,\mathcal{X})\vee\mathrm{Frm}^{\mathcal{E}_{\mathrm{p}}(\vec{\mathcal{D}})}(\sigma)(\varrho_2,\mathcal{X})$.

Event/data sentences. The set of Σ-*event/data sentences* $\mathrm{Sen}^{\mathcal{E}_{\mathrm{p}}(\vec{\mathcal{D}})}(\Sigma) = \{\varrho\in\mathrm{Frm}^{\mathcal{E}_{\mathrm{p}}(\vec{\mathcal{D}})}(\Sigma,\mathcal{X}) \mid \mathrm{fvar}(\varrho)=\emptyset\}$ consists of all Σ-event/data formulæ without free variables. The *event/data sentence translation* $\mathrm{Sen}^{\mathcal{E}_{\mathrm{p}}(\vec{\mathcal{D}})}(\sigma) : \mathrm{Sen}^{\mathcal{E}_{\mathrm{p}}(\vec{\mathcal{D}})}(\Sigma)\to\mathrm{Sen}^{\mathcal{E}_{\mathrm{p}}(\vec{\mathcal{D}})}(\Sigma')$ along an event/data signature morphism $\sigma:\Sigma\to\Sigma'$ is defined as $\mathrm{Sen}^{\mathcal{E}_{\mathrm{p}}(\vec{\mathcal{D}})}(\sigma) = \mathrm{Frm}^{\mathcal{E}_{\mathrm{p}}(\vec{\mathcal{D}})}(\sigma,\mathcal{X})$.

C.4. Event/data satisfaction relation

Interpretation. For an event/data signature Σ and variables $\mathcal{X}$, let $e(\mathbf{any}\ X)$ be an event term with $X = x_1,\ldots,x_n\subseteq|\mathcal{X}|$ and β_X a valuation for X. We then denote the event occurrence $e(\beta_X(x_i)_{1\le i\le n})$ by $e(\beta_X)$. The set of valuations for X are denoted B_X.

For a Σ-event/data structure M and a valuation β of the variables $\mathcal{X}$, the *interpretation* $(R(M,\beta)_\lambda\subseteq\Gamma(M)\times\Gamma(M))_{\lambda\in\Lambda(\Sigma,\mathcal{X})}$ of event/data actions is given by

- $R(M,\beta)_{e(\mathbf{any}\,X)/\!\!/\psi(Y)} = \{(\gamma,\gamma')\in R(M)_{e(\beta_X)} \mid \beta_X\in\mathrm{B}_X, (\omega(M)(\gamma),\omega(M)(\gamma')),\beta\{X\mapsto\beta_X\}\models^{2\vec{\mathcal{D}}}_{\delta(\Sigma),\mathcal{X}}\psi\}$,
  $(\beta\{X\mapsto\beta_X\})(x) = \beta_X(x)$ for $x\in X$, $(\beta\{X\mapsto\beta_X\})(x') = \beta(x')$ for $x\notin X$,
- $R(M,\beta)_{\lambda_1+\lambda_2} = R(M,\beta)_{\lambda_1}\cup R(M,\beta)_{\lambda_2}$,
- $R(M,\beta)_{\lambda_1;\lambda_2} = R(M,\beta)_{\lambda_1}; R(M,\beta)_{\lambda_2}$,
- $R(M,\beta)_{\lambda^*} = (R(M,\beta)_\lambda)^*$.

For a $\sigma:\Sigma\to\Sigma'$, we abbreviate $Str^{\vec{\mathcal{D}}}(\delta(\sigma)_0,\mathcal{X})(\beta')$ by $\beta'|\sigma$.

Lemma 3. *Let $\sigma:\Sigma\to\Sigma'$ be an event/data signature morphism, M' a Σ'-event/data structure, β' a valuation for $\mathcal{X}^{\delta(\sigma)_0}$, and $\gamma'_1\in\Gamma(M'|\sigma)\subseteq\Gamma(M')$. Then*

1. *for all $\gamma'_2\in\Gamma(M')$ and all $\lambda = e(\mathbf{any}\,X)/\!\!/\psi(Y)\in\Lambda(\Sigma,\mathcal{X})$, it holds that $(\gamma'_1,\gamma'_2)\in R(M'|\sigma,\beta'|\sigma)_\lambda$ if, and only if, $(\gamma'_1,\gamma'_2)\in R(M',\beta')_{\Lambda(\sigma,\mathcal{X})(\lambda)}$;*
2. *for all $\lambda\in\Lambda(\Sigma,\mathcal{X})$, $\{\gamma'_2\in\Gamma(M'|\sigma)\mid(\gamma'_1,\gamma'_2)\in R(M'|\sigma,\beta'|\sigma)_\lambda\} = \{\gamma'_2\in\Gamma(M')\mid(\gamma'_1,\gamma'_2)\in R(M',\beta')_{\Lambda(\sigma,\mathcal{X})(\lambda)}\}$.*

Satisfaction relation. Given an event/data signature Σ and a Σ-event/data structure M, the *satisfaction* of an event/data formula $\varrho\in\mathrm{Frm}^{\mathcal{E}_{\mathrm{p}}(\vec{\mathcal{D}})}(\Sigma,\mathcal{X})$ is inductively defined w.r.t. configurations $\gamma\in\Gamma(M)$ and valuations β:

- $M,\gamma,\beta\models^{\mathcal{E}_{\mathrm{p}}(\vec{\mathcal{D}})}_{\Sigma,\mathcal{X}}$ true,
- $M,\gamma,\beta\models^{\mathcal{E}_{\mathrm{p}}(\vec{\mathcal{D}})}_{\Sigma,\mathcal{X}}\varphi(X)$ iff $\omega(M)(\gamma),\beta\models^{\vec{\mathcal{D}}}_{\delta(\Sigma),\mathcal{X}}\varphi(X)$,
- $M,\gamma,\beta\models^{\mathcal{E}_{\mathrm{p}}(\vec{\mathcal{D}})}_{\Sigma,\mathcal{X}}\langle\lambda\rangle\varrho$ iff $M,\gamma',\beta\models^{\mathcal{E}_{\mathrm{p}}(\vec{\mathcal{D}})}_{\Sigma,\mathcal{X}}\varrho$
  for some $\gamma'\in\Gamma(M)$ with $(\gamma,\gamma')\in R(M,\beta)_\lambda$,
- $M,\gamma,\beta\models^{\mathcal{E}_{\mathrm{p}}(\vec{\mathcal{D}})}_{\Sigma,\mathcal{X}}\exists x.\,\varrho$ iff $M,\beta\{x\mapsto\mathrm{B}_x\},\gamma\models^{\mathcal{E}_{\mathrm{p}}(\vec{\mathcal{D}})}_{\Sigma,\mathcal{X}}\varrho$
  for some $\mathrm{B}_x\in|Str^{\vec{\mathcal{D}}}(\mathcal{X}(x))|$
  where $(\beta\{x\mapsto\mathrm{B}_x\})(x) = \mathrm{B}_x$ and $(\beta\{x\mapsto\mathrm{B}_x\})(x') = \beta(x')$ for $x\neq x'$,
- $M,\gamma,\beta\models^{\mathcal{E}_{\mathrm{p}}(\vec{\mathcal{D}})}_{\Sigma,\mathcal{X}}\neg\varrho$ iff $M,\gamma,\beta\not\models^{\mathcal{E}_{\mathrm{p}}(\vec{\mathcal{D}})}_{\Sigma,\mathcal{X}}\varrho$,
- $M,\gamma,\beta\models^{\mathcal{E}_{\mathrm{p}}(\vec{\mathcal{D}})}_{\Sigma,\mathcal{X}}\varrho_1\vee\varrho_2$ iff $M,\gamma,\beta\models^{\mathcal{E}_{\mathrm{p}}(\vec{\mathcal{D}})}_{\Sigma,\mathcal{X}}\varrho_1$ or $M,\gamma,\beta\models^{\mathcal{E}_{\mathrm{p}}(\vec{\mathcal{D}})}_{\Sigma,\mathcal{X}}\varrho_2$.

Lemma 4. *Let $\sigma:\Sigma\to\Sigma'$ be an event/data signature morphism, $\mathcal{X}$ variables for $\delta(\Sigma)$, and M' a Σ'-event/data structure. For all $\gamma'\in\Gamma(M'|\sigma)\subseteq\Gamma(M')$, all valuations β' for $\mathcal{X}^{\delta(\sigma)_0}$ over $\delta(\Sigma')$, and all $\varrho\in\mathrm{Frm}^{\mathcal{E}_{\mathrm{p}}(\vec{\mathcal{D}})}(\Sigma,\mathcal{X})$ it holds that*

$$M'|\sigma,\gamma',\beta'|\sigma\models^{\mathcal{E}_{\mathrm{p}}(\vec{\mathcal{D}})}_{\Sigma,\mathcal{X}}\varrho \iff M',\gamma',\beta'\models^{\mathcal{E}_{\mathrm{p}}(\vec{\mathcal{D}})}_{\Sigma',\mathcal{X}^{\delta(\sigma)_0}}\mathrm{Frm}^{\mathcal{E}_{\mathrm{p}}(\vec{\mathcal{D}})}(\sigma,\mathcal{X})(\varrho)\,.$$

Proof. We apply induction on the structure of Σ-event/data formulæ. We only consider the cases $\varphi(X)$, $\langle\lambda\rangle\varrho$, and $\exists x.\,\varrho$; true, negation, and disjunction are straightforward.

Case $\varphi(X)$:

$$
\begin{aligned}
& M'|\sigma, \gamma', \beta'|\sigma \models^{\mathcal{E}_{\mathrm{p}}(\vec{\mathcal{D}})}_{\Sigma,\mathcal{X}} \varphi(X) \\
\Leftrightarrow \quad & \{ \text{ def. } \models^{\mathcal{E}_{\mathrm{p}}(\vec{\mathcal{D}})} \} \\
& \omega(M'|\sigma)(\gamma'), \beta'|\sigma, \models^{\vec{\mathcal{D}}}_{\delta(\Sigma),\mathcal{X}} \varphi(X) \\
\Leftrightarrow \quad & \{ \text{ def. } \omega(M'|\sigma) \} \\
& Str^{\vec{\mathcal{D}}}(\delta(\sigma))(\omega(M')(\gamma')), Str^{\vec{\mathcal{D}}}(\delta(\sigma)_0, \mathcal{X})(\beta') \models^{\vec{\mathcal{D}}}_{\delta(\Sigma),\mathcal{X}} \varphi(X) \\
\Leftrightarrow \quad & \{ \text{ Lemma 2 } \} \\
& \omega(M')(\gamma'), \beta' \models^{\vec{\mathcal{D}}}_{\delta(\Sigma'),\mathcal{X}^{\delta(\sigma)_0}} \mathrm{Frm}^{\vec{\mathcal{D}}}(\delta(\sigma), \mathcal{X})(\varphi(X)) \\
\Leftrightarrow \quad & \{ \text{ def. } \models^{\mathcal{E}_{\mathrm{p}}(\vec{\mathcal{D}})} \} \\
& M', \gamma', \beta' \models^{\mathcal{E}_{\mathrm{p}}(\vec{\mathcal{D}})}_{\Sigma',\mathcal{X}^{\delta(\sigma)_0}} \mathrm{Frm}^{\vec{\mathcal{D}}}(\delta(\sigma), \mathcal{X})(\varphi(X)) \\
\Leftrightarrow \quad & \{ \text{ def. } \mathrm{Frm}^{\mathcal{E}_{\mathrm{p}}(\vec{\mathcal{D}})}(\sigma) \} \\
& M', \gamma', \beta' \models^{\mathcal{E}_{\mathrm{p}}(\vec{\mathcal{D}})}_{\Sigma',\mathcal{X}^{\delta(\sigma)_0}} \mathrm{Frm}^{\mathcal{E}_{\mathrm{p}}(\vec{\mathcal{D}})}(\sigma, \mathcal{X})(\varphi(X))
\end{aligned}
$$

Case $\langle\lambda\rangle\varrho$:

$$
\begin{aligned}
& M'|\sigma, \gamma', \beta'|\sigma \models^{\mathcal{E}_{\mathrm{p}}(\vec{\mathcal{D}})}_{\Sigma,\mathcal{X}} \langle\lambda\rangle\varrho \\
\Leftrightarrow \quad & \{ \text{ def. } \models^{\mathcal{E}_{\mathrm{p}}(\vec{\mathcal{D}})} \} \\
& M'|\sigma, \gamma'', \beta'|\sigma \models^{\mathcal{E}_{\mathrm{p}}(\vec{\mathcal{D}})}_{\Sigma,\mathcal{X}} \varrho \quad \text{for some } \gamma'' \in \Gamma(M'|\sigma) \text{ with } (\gamma', \gamma'') \in R(M'|\sigma, \beta'|\sigma)_\lambda \\
\Leftrightarrow \quad & \{ \text{ Lemma 3(2) } \} \\
& M'|\sigma, \gamma'', \beta'|\sigma \models^{\mathcal{E}_{\mathrm{p}}(\vec{\mathcal{D}})}_{\Sigma,\mathcal{X}} \varrho \quad \text{for some } \gamma'' \in \Gamma(M'|\sigma) \subseteq \Gamma(M') \text{ with } (\gamma', \gamma'') \in R(M', \beta')_{\Lambda(\sigma,\mathcal{X})(\lambda)} \\
\Leftrightarrow \quad & \{ \text{ I.H. by Lemma 3(2) for "}\Leftarrow\text{" } \} \\
& M', \gamma'', \beta' \models^{\mathcal{E}_{\mathrm{p}}(\vec{\mathcal{D}})}_{\Sigma',\mathcal{X}^{\delta(\sigma)_0}} \mathrm{Frm}^{\mathcal{E}_{\mathrm{p}}(\vec{\mathcal{D}})}(\sigma, \mathcal{X})(\varrho) \quad \text{for some } \gamma'' \in \Gamma(M') \text{ with } (\gamma', \gamma'') \in R(M', \beta')_{\Lambda(\sigma,\mathcal{X})(\lambda)} \\
\Leftrightarrow \quad & \{ \text{ def. } \models^{\mathcal{E}_{\mathrm{p}}(\vec{\mathcal{D}})} \} \\
& M', \gamma', \beta' \models^{\mathcal{E}_{\mathrm{p}}(\vec{\mathcal{D}})}_{\Sigma',\mathcal{X}^{\delta(\sigma)_0}} \langle\Lambda(\sigma, \mathcal{X})(\lambda)\rangle \mathrm{Frm}^{\mathcal{E}_{\mathrm{p}}(\vec{\mathcal{D}})}(\sigma, \mathcal{X})(\varrho) \\
\Leftrightarrow \quad & \{ \text{ def. } \mathrm{Frm}^{\mathcal{E}_{\mathrm{p}}(\vec{\mathcal{D}})}(\sigma) \} \\
& M', \gamma', \beta' \models^{\mathcal{E}_{\mathrm{p}}(\vec{\mathcal{D}})}_{\Sigma',\mathcal{X}^{\delta(\sigma)_0}} \mathrm{Frm}^{\mathcal{E}_{\mathrm{p}}(\vec{\mathcal{D}})}(\sigma, \mathcal{X})(\langle\lambda\rangle\varrho)
\end{aligned}
$$

Case $\exists x \,.\, \varrho$:

$$
\begin{aligned}
& M'|\sigma, \gamma', \beta'|\sigma \models^{\mathcal{E}_{\mathrm{p}}(\vec{\mathcal{D}})}_{\Sigma,\mathcal{X}} \exists x \,.\, \varrho \\
\Leftrightarrow \quad & \{ \text{ def. } \models^{\mathcal{E}_{\mathrm{p}}(\vec{\mathcal{D}})} \} \\
& M'|\sigma, \gamma', (\beta'|\sigma)\{x \mapsto \mathrm{B}_x\} \models^{\mathcal{E}_{\mathrm{p}}(\vec{\mathcal{D}})}_{\Sigma,\mathcal{X}} \varrho \quad \text{for some } \mathrm{B}_x \in |Str^{\vec{\mathcal{D}}}(\mathcal{X}(x))| \\
\Leftrightarrow \quad & \{ \text{ Lemma 1 } \} \\
& M'|\sigma, \gamma', (\beta'\{x \mapsto \mathrm{B}'_x\})|\sigma \models^{\mathcal{E}_{\mathrm{p}}(\vec{\mathcal{D}})}_{\Sigma,\mathcal{X}} \varrho \quad \text{for some } \mathrm{B}'_x \in |Str^{\vec{\mathcal{D}}}(\mathcal{X}^{\delta(\sigma)_0}(x))| \\
\Leftrightarrow \quad & \{ \text{ I.H. } \} \\
& M', \gamma', \beta'\{x \mapsto \mathrm{B}'_x\} \models^{\mathcal{E}_{\mathrm{p}}(\vec{\mathcal{D}})}_{\Sigma',\mathcal{X}^{\delta(\sigma)_0}} \varrho \quad \text{for some } \mathrm{B}'_x \in |Str^{\vec{\mathcal{D}}}(\mathcal{X}^{\delta(\sigma)_0}(x))| \\
\Leftrightarrow \quad & \{ \text{ def. } \models^{\mathcal{E}_{\mathrm{p}}(\vec{\mathcal{D}})} \} \\
& M', \gamma', \beta' \models^{\mathcal{E}_{\mathrm{p}}(\vec{\mathcal{D}})}_{\Sigma',\mathcal{X}^{\delta(\sigma)_0}} \exists x \,.\, \mathrm{Frm}^{\mathcal{E}_{\mathrm{p}}(\vec{\mathcal{D}})}(\sigma, \mathcal{X})(\varrho)
\end{aligned}
$$

$$\Leftrightarrow \qquad \{ \text{ def. } \mathrm{Frm}^{\mathcal{E}_{\mathrm{p}}(\vec{\mathcal{D}})} \}$$

$$M', \gamma', \beta' \models^{\mathcal{E}_{\mathrm{p}}(\vec{\mathcal{D}})}_{\Sigma', \mathcal{X}^{\delta(\sigma)_0}} \mathrm{Frm}^{\mathcal{E}_{\mathrm{p}}(\vec{\mathcal{D}})}(\sigma, \mathcal{X})(\exists x \,.\, \varrho) \quad \square$$

For sentences $\varrho \in \mathrm{Sen}^{\mathcal{E}_{\mathrm{p}}(\vec{\mathcal{D}})}(\Sigma)$, we define $M \models^{\mathcal{E}_{\mathrm{p}}(\vec{\mathcal{D}})}_{\Sigma} \varrho$ iff $M, \gamma_0, \beta \models^{\mathcal{E}_{\mathrm{p}}(\vec{\mathcal{D}})}_{\Sigma, \mathcal{X}} \varrho$ for all $\gamma_0 \in \Gamma_0(M)$ and all valuations β for $\mathcal{X}$.

Corollary 1. *For all $\sigma : \Sigma \to \Sigma'$ in $\mathbb{S}^{\mathcal{E}_{\mathrm{p}}(\vec{\mathcal{D}})}$, $M' \in |Str^{\mathcal{E}_{\mathrm{p}}(\vec{\mathcal{D}})}(\Sigma')|$, and $\varrho \in \mathrm{Sen}^{\mathcal{E}_{\mathrm{p}}(\vec{\mathcal{D}})}(\Sigma)$,*

$$Str^{\mathcal{E}_{\mathrm{p}}(\vec{\mathcal{D}})}(\sigma)(M') \models^{\mathcal{E}_{\mathrm{p}}(\vec{\mathcal{D}})}_{\Sigma} \varrho \iff M' \models^{\mathcal{E}_{\mathrm{p}}(\vec{\mathcal{D}})}_{\Sigma'} \mathrm{Sen}^{\mathcal{E}_{\mathrm{p}}(\vec{\mathcal{D}})}(\sigma)(\varrho) \,.$$

Corollary 2. $(\mathbb{S}^{\mathcal{E}_{\mathrm{p}}(\vec{\mathcal{D}})}, Str^{\mathcal{E}_{\mathrm{p}}(\vec{\mathcal{D}})}, \mathrm{Sen}^{\mathcal{E}_{\mathrm{p}}(\vec{\mathcal{D}})}, \models^{\mathcal{E}_{\mathrm{p}}(\vec{\mathcal{D}})})$ *forms an institution.*

References

[1] Jean-Raymond Abrial, Modeling in Event-B: System and Software Engineering, Cambridge University Press, 2013.
[2] Torben Braüner, Hybrid Logic and Its Proof-Theory, Applied Logic Ser., Springer, 2010.
[3] Răzvan Diaconescu, Alexandre Madeira, Encoding hybridized institutions into first-order logic, Math. Struct. Comput. Sci. 26 (5) (2016) 745–788.
[4] Marie Farrell, Rosemary Monahan, James F. Power, An institution for event-B, in: Phillip James, Markus Roggenbach (Eds.), Rev. Sel. Papers 23rd IFIP WG 1.3 Intl. Ws. Recent Trends in Algebraic Development Techniques, in: Lect. Notes Comp. Sci., vol. 10644, Springer, 2017, pp. 104–119.
[5] Joseph A. Goguen, Rod M. Burstall, Institutions: abstract model theory for specification and programming, J. ACM 39 (1992) 95–146.
[6] Jan Friso Groote, Mohammad Reza Mousavi, Modeling and Analysis of Communicating Systems, MIT Press, 2014.
[7] David Harel, Dexter Kozen, Jerzy Tiuryn, Dynamic Logic, MIT Press, 2000.
[8] Rolf Hennicker, Alexander Knapp, Alexandre Madeira, Hybrid dynamic logic institutions for event/data-based systems, Form. Asp. Comput. 33 (6) (2021) 1209–1248.
[9] Rolf Hennicker, Alexandre Madeira, Alexander Knapp, A hybrid dynamic logic for event/data-based systems, in: Reiner Hähnle, Wil M.P. van der Aalst (Eds.), Proc. 22nd Intl. Conf. Fundamental Approaches to Software Engineering, in: Lect. Notes Comp. Sci., vol. 11424, Springer, 2019, pp. 79–97.
[10] Alexander Knapp, Till Mossakowski, Markus Roggenbach, Martin Glauer, An institution for simple UML state machines, in: Alexander Egyed, Ina Schaefer (Eds.), Proc. 18th Intl. Conf. Fundamental Approaches to Software Engineering, in: Lect. Notes Comp. Sci., vol. 9033, Springer, 2015, pp. 3–18.
[11] Leslie Lamport, Specifying Systems: The TLA+ Language and Tools for Hardware and Software Engineers, Addison-Wesley, 2003.
[12] Saunders Mac Lane, Categories for the Working Mathematician, 2nd edition, Springer, 1998.
[13] Alexandre Madeira, Luís Soares Barbosa, Rolf Hennicker, Manuel A. Martins, Dynamic logic with binders and its application to the development of reactive systems, in: Augusto Sampaio, Farn Wang (Eds.), Proc. 13th Intl. Conf. Theoretical Aspects of Computing (ICTAC 2016), in: Lect. Notes Comp. Sci., vol. 9965, 2016, pp. 422–440.
[14] Alexandre Madeira, Luís Soares Barbosa, Rolf Hennicker, Manuel A. Martins, A logic for the stepwise development of reactive systems, Theor. Comput. Sci. 744 (2018) 78–96.
[15] Alexandre Madeira, Manuel A. Martins, Luís Soares Barbosa, Rolf Hennicker, Refinement in hybridised institutions, Form. Asp. Comput. 27 (2) (2015) 375–395.
[16] Manuel A. Martins, Alexandre Madeira, Răzvan Diaconescu, Luís Soares Barbosa, Hybridization of institutions, in: Andrea Corradini, Bartek Klin, Corina Cîrstea (Eds.), Proc. 4th Intl. Conf. Algebra and Coalgebra in Computer Science, in: Lect. Notes Comp. Sci., vol. 6859, Springer, 2011, pp. 283–297.
[17] OMG (Object Management Group), Unified Modeling Language, Standard formal/17-12-05, OMG, 2017.
[18] Pascal Poizat, Jean-Claude Royer, A formal architectural description language based on symbolic transition systems and modal logic, J. Univers. Comput. Sci. 12 (12) (2006) 1741–1782.
[19] Tobias Rosenberger, Alexander Knapp, Markus Roggenbach, An institutional approach to communicating UML state machines, in: Einar Broch Johnsen, Manuel Wimmer (Eds.), Proc. 25th Intl. Conf. Fundamental Approaches to Software Engineering, in: Lect. Notes Comp. Sci., vol. 13241, Springer, 2022, pp. 205–224, https://doi.org/10.1007/978-3-030-99429-7_12.
[20] Donald Sannella, Andrzej Tarlecki, Foundations of Algebraic Specification and Formal Software Development, EATCS Monographs in Theoretical Computer Science, Springer, 2012.
[21] Maurice H. ter Beek, Alessandro Fantechi, Stefania Gnesi, Franco Mazzanti, An action/state-based model-checking approach for the analysis of communication protocols for service-oriented applications, in: Stefan Leue, Pedro Merino (Eds.), Rev. Sel. Papers 12th Intl. Ws. Formal Methods for Industrial Critical Systems, in: Lect. Notes Comp. Sci., vol. 4916, Springer, 2008, pp. 133–148.
[22] Ionut Tutu, Claudia Elena Chirita, José Luiz Fiadeiro, Dynamic reconfiguration via typed modalities, in: Marieke Huisman, Corina S. Pasareanu, Naijun Zhan (Eds.), Proc. 24th Intl. Symp. Formal Methods (FM 2021), in: Lect. Notes Comp. Sci., vol. 13047, Springer, 2021, pp. 599–615.