A Heterogeneous Approach to Service-Oriented Systems Specification[*]

Alexander Knapp
Institute of Computer Science
University of Augsburg
knapp@informatik.uni-augsburg.de

Grzegorz Marczyński
Institute of Informatics
University of Warsaw
gmarc@mimuw.edu.pl

Martin Wirsing
Ludwig-Maximilians-Universität München
wirsing@ifi.lmu.de

Artur Zawłocki
Institute of Informatics
University of Warsaw
zawlocki@mimuw.edu.pl

ABSTRACT

Service-oriented architecture (SOA) is a relatively new approach to software system development. It divides system functionality to independent, loosely coupled, interoperable services. In this paper we propose a new heterogeneous specification approach for SOA systems where a heterogeneous structured specification consists of a number of specifications of individual services written in a "local" logic and where the specification of their interactions is separately described in a "global" logic. A main feature of our global logic is the possibility of describing the dynamic change of service communications over time. Our approach is based on the theory of institutions: we show that both logics form institutions and that these institutions are connected by an institution comorphism. We illustrate our approach by a simple scenario of an e-university management system and show the power of the heterogeneous specification approach by a compositional refinement of the scenario.

Categories and Subject Descriptors

F.3.1 [**Theory of Computation**]: Logics and Meanings of Programs—*Specifying and Verifying and Reasoning about Programs*

1. INTRODUCTION

Service-oriented architecture (SOA, [1]) is a software construction paradigm where system functions are separated into a number of encapsulated components. These components, called *services*, are independent entities capable to communicate with each other through their public interfaces. Services are *loosely-coupled* and their cooperation structure may change dynamically, i.e. their communication configuration may change at any time.

In this paper we propose a new formal approach for specifying the behaviour of SOA systems in a declarative way. This is in contrast to many current SOA design and specification languages such as BPEL, BPMN, UML4SOA or SCA where behaviour is typically specified by state machines or workflow-oriented notations. An exception is SRML which also uses declarative temporal logic specifications ([2]). We believe that such declarative logical specifications complement the more concrete operational approaches; they may e.g. serve as goal-oriented requirement specifications and be used for proving properties of operational models. Another advantage of our approach is that it is based on the general framework of (heterogeneous) institutions ([3, 4, 5]) which allows us to integrate our service specifications easily and in a mathematically well-defined way with other languages such as UML ([6]).

We introduce institutions suitable for specifying SOA systems in which services are classified by *roles* (or *interfaces*), the interaction structure of services may change over time, and the number of services may be not known in advance. Our approach is illustrated with an example scenario based on the SENSORIA e-learning case study ([7]).

The scenario describes an E-course Management System with the *student service* acting as a representation of a student to other services of the system. When a (human) student enters its data to the portal (a user-interface of student service that is irrelevant here), the student service transfers these data to a *course manager* with a request for all matching courses. In consequence, the course manager contacts all available *course providers* (one provider represents one course) and gathers their replies. Then it selects only those courses that match the student's curriculum and university regulations. Eventually it sends the answer to the student service, that lets the student choose from a given selection. Finally, on the student's request, the student service registers the student to selected courses.

To describe such scenarios we propose two different log-

[*]This research was supported by the EC 6th Framework project SENSORIA: Software Engineering for Service-Oriented Overlay Computers (IST-2005-016004); project web site: `http://www.sensoria-ist.eu/`.

ical systems formalised as institutions. In the *local logic* one can specify the behaviour of individual services, such as a course manager or a course provider. The services may themselves be composed of other services and components; however, in the local language they can be described only as "monolithic" entities, with no indication in which part of the service the actions occur. In contrast to this, the *global logic* allows one to specify the choreography of a SOA system, i.e. the common behaviour of co-operating services. It provides means to refer to individual system components as well as to describe the system configuration and its evolution in time. We use a synchronisation predicate for modelling the communication connections of the service partners; this communication structure and thus the system configuration can change dynamically (i.e. it may be different for each configuration). Another feature of the global logic is quantification over components which allows one to write specifications for systems with an unbounded number of services.

We relate these two institutions via an institution comorphism ([8]) that gives grounds for the heterogeneous specification ([5]) of SOA systems, where the specification of a whole system is a heterogeneous structured specification that consists of a number of specifications of individual services in the local logic and the specification of their interactions separately described in the global one. It also gives the possibility for compositionally constructing specifications using well-defined algebraic specification operators and for expressing refinements of specifications on both levels. Last not least it provides another example for heterogeneous semantics of modelling languages and can be seen as further step in building heterogeneous semantics of multi-modelling languages ([6, 9]).

The paper is organised as follows. In Section 2 we introduce the local logic and formalise it as an institution. In Section 3 we do the same for the global logic. Then, in Section 4, we relate both logics using a map between institutions and introduce the notion of *heterogeneous service specifications*. In Section 5 we survey the related work and in Section 6 we provide concluding remarks.

2. LOCAL LOGIC

Our starting point is a logic for specifying the behaviour of individual components of SOA. We call it the *local logic* to contrast it with its extension presented in Sect. 3, intended to describe the behaviour of the system as a whole.

In the paper we use a relatively expressive temporal language to facilitate writing example specifications, both on the local and on the global level. Our local logic is a variant of the full computation tree logic CTL* ([10]) enriched with past temporal modalities. Unlike in the usual interpretation of CTL* over Kripke structures, atomic formulae will refer to transitions instead of states. Except for this detail, syntax and semantics of formulae is standard. In Sect. 6 we briefly comment on basing the local logic on a more restricted language, such as LTL or CTL ([10]).

Definition 1. Let A be a set. The formulae of the logic aPCTL$^*(A)$ (for action-based CTL* with past temporal operators and atomic propositions from A) are defined by the following grammar, where $a \in A$:

$$\varphi ::= \bot \mid \varphi_1 \Rightarrow \varphi_2 \mid a \mid \mathsf{X}\,\varphi \mid \mathsf{Y}\,\varphi \mid \varphi_1\,\mathsf{U}\,\varphi_2 \mid \varphi_1\,\mathsf{S}\,\varphi_2 \mid \mathsf{A}\,\varphi$$

Here, X (in the next step), U (until) and A (for every path) are standard operators of CTL*, and Y (in the previous step) and S (since) are past temporal operators (see e.g. [11]).

We recall several standard notions. A *labelled transition system* (an *lts* for short) $T = \langle A, S, r, \rightarrow\rangle$ consists of a set A of *actions*, a set S of *states*, an *initial state* $r \in S$ and a *transition relation* $\rightarrow\, \subseteq S \times A \times S$. As usual, we write $s \xrightarrow{a} t$ for $\langle s, a, t\rangle \in\, \rightarrow$. A *path* in T is a possibly infinite sequence $\pi = s_0 \xrightarrow{a_1} s_1 \xrightarrow{a_2} s_2 \ldots$ of consecutive transitions. A *run* is a maximal path starting in the initial state. The number of transitions in π is denoted by $|\pi| \leq \omega$. For finite $n \leq |\pi|$, by $\pi|_n$ we denote the prefix $s_0 \xrightarrow{a_1} s_1 \ldots s_{n-1} \xrightarrow{a_n} s_n$ of π.

Definition 2. Let A be a set and let $T = \langle A, S, r, \rightarrow\rangle$ be an lts. The satisfaction relation $\models$ is defined by structural induction on formulae of aPCTL$^*(A)$.

Let $\pi = s_0 \xrightarrow{a_1} s_1 \xrightarrow{a_2} \ldots$ be a run and let $i \leq |\pi|$:

- $\pi, i \models \bot$ does not hold;
- $\pi, i \models \varphi_1 \Rightarrow \varphi_2$ iff $\pi, i \not\models \varphi_1$ or $\pi, i \models \varphi_2$;
- $\pi, i \models a$ iff $i < |\pi|$ and $a_{i+1} = a$;
- $\pi, i \models \mathsf{X}\,\varphi$ iff $i < |\pi|$ and $\pi, (i+1) \models \varphi$;
- $\pi, i \models \mathsf{Y}\,\varphi$ iff $i > 0$ and $\pi, (i-1) \models \varphi$;
- $\pi, i \models \varphi_1\,\mathsf{U}\,\varphi_2$ iff there exists a finite j with $i \leq j \leq |\pi|$ such that $\pi, j \models \varphi_2$ and $\pi, i' \models \varphi_1$ for every i' with $i \leq i' < j$;
- $\pi, i \models \varphi_1\,\mathsf{S}\,\varphi_2$ iff there exists j with $j \leq i$ such that $\pi, j \models \varphi_2$ and $\pi, i' \models \varphi_1$ for every i' with $j < i' \leq i$;
- $\pi, i \models \mathsf{A}\,\varphi$ iff for every run π' such that $\pi|_i = \pi'|_i$, $\pi', i \models \varphi$.

φ is satisfied in T, written $T \models_{\mathrm{aPCTL}^*} \varphi$, if $\pi, 0 \models \varphi$ holds for every run π of T.

2.1 Institution $\mathbb{L}$ for Local Logic

An institution ([3]) $\mathbb{I} = \langle \mathbf{Sign}, \mathbf{Mod}, \mathbf{Sen}, \langle\models_\Sigma\rangle_{\Sigma\in|\mathbf{Sign}|}\rangle$ consists of: a category $\mathbf{Sign}$ of *signatures*; a functor $\mathbf{Mod} : \mathbf{Sign}^{op} \rightarrow \mathbf{Class}$ giving a class of Σ-*models* for each $\Sigma \in |\mathbf{Sign}|$; a functor $\mathbf{Sen} : \mathbf{Sign} \rightarrow \mathbf{Set}$ giving a set of Σ-*sentences* for each $\Sigma \in |\mathbf{Sign}|$; a family $\{\models_\Sigma\}_{\Sigma\in|\mathbf{Sign}|}$ of *satisfaction relations*, where $\models_\Sigma\, \subseteq \mathbf{Mod}(\Sigma) \times \mathbf{Sen}(\Sigma)$ for each $\Sigma \in |\mathbf{Sign}|$. The components of $\mathbb{I}$ are subject to the following *satisfaction condition*: for every signature morphism $\sigma : \Sigma \rightarrow \Sigma'$, Σ'-model M' and Σ-sentence φ,

$$\mathbf{Mod}(\sigma)(M') \models_\Sigma \varphi \iff M' \models_{\Sigma'} \mathbf{Sen}(\sigma)(\varphi) .$$

For a morphism $\sigma : \Sigma \rightarrow \Sigma'$ in $\mathbf{Sign}$, $\mathbf{Sen}(\sigma)$ is called the σ-*translation* map and $\mathbf{Mod}(\sigma)$ the σ-*reduct* functor.[1] A *specification* $Sp = \langle\Sigma, Ax\rangle$ in $\mathbb{I}$ consists of a signature $\Sigma \in |\mathbf{Sign}|$ and a set of sentences $Ax \subseteq \mathbf{Sen}(\Sigma)$. A model $M \in \mathbf{Mod}(\Sigma)$ *satisfies* Sp, written as $M \models Sp$, if $M \models_\Sigma \varphi$ for all $\varphi \in Ax$. A specification $Sp' = \langle\Sigma', Ax'\rangle$ with a signature morphism $\sigma : \Sigma \rightarrow \Sigma'$ *refines* Sp, written as $Sp \rightsquigarrow Sp'$, if $\mathbf{Mod}(\sigma)(M') \models Sp$ for all M' with $M' \models Sp'$.

Standard examples of institutions include propositional, equational and first-order logic (single- and many-sorted),

[1]A more standard definition requires **Mod** to be a **Cat**-valued functor. However, for our purposes morphisms of models are not relevant and thus it is sufficient to consider model classes.

see e.g. [3]. From the point of view of behaviour specification, more interesting examples of institutions are those for temporal and modal logics. Numerous examples of such institutions exist, we give a brief overview in Section 5.

As the category of signatures of the institution for local logic $\mathbb{L}$ we take simply **Set**. Elements of a signature set $\mathcal{A}$ are called *action names*. For a set $\mathcal{A}$, an $\mathcal{A}$-sentence is a formula φ of aPCTL$^*(\mathcal{A})$, and for a map $f : \mathcal{A} \to \mathcal{B}$, the f-translation of φ is a formula $f(\varphi)$ of aPCTL$^*(\mathcal{B})$ obtained by replacing each name $a \in \mathcal{A}$ in φ with $f(a) \in \mathcal{B}$. The translation is clearly functorial and thus we obtain the sentence functor of the local institution $\mathbf{Sen}^{\mathbb{L}} : \mathbf{Set} \to \mathbf{Set}$. The model functor is defined as follows.

Definition 3. Let $\mathcal{A}$ be a set. A *local $\mathcal{A}$-model* $M = \left\langle T, \langle a^M \rangle_{a\in\mathcal{A}} \right\rangle$ consists of an lts $T = \langle A, S, r, \to \rangle$ and, for every $a \in \mathcal{A}$, an action $a^M \in A$.

Let $f : \mathcal{A} \to \mathcal{B}$ be a map and let $M = \left\langle T, \langle b^M \rangle_{b\in\mathcal{B}} \right\rangle$ be a local $\mathcal{B}$-model. The *f-reduct* of M is a local $\mathcal{A}$-model $M|_f = \left\langle T, \langle a^{M|_f} \rangle_{a\in\mathcal{A}} \right\rangle$ where, for every $a \in \mathcal{A}$, $a^{M|_f} = f(a)^M$. These data define the model functor $\mathbf{Mod}^{\mathbb{L}} : \mathbf{Set}^{op} \to \mathbf{Class}$.

Note that we do not require that the action names in a signature $\mathcal{A}$ be the same as the ones in the lts T underlying a model. Instead, each name in $\mathcal{A}$ is interpreted as an action in T. A model may thus contain transitions which are *unobservable*, i.e., not labelled with any of a^M. This allows us to define model reduct in a natural way and makes the proof of the satisfaction condition straightforward.

Definition 4. Let $\mathcal{A}$ be a set, $M = \left\langle T, \langle a^M \rangle_{a\in\mathcal{A}} \right\rangle$ be a local $\mathcal{A}$-model and let φ be an $\mathcal{A}$-sentence. By $(\varphi)^M$ we denote the translation of φ along the map $a \mapsto a^M$. Then

$$M \models^{\mathbb{L}}_{\mathcal{A}} \varphi \quad \text{iff} \quad T \models_{\text{aPCTL}^*} (\varphi)^M .$$

At this point we have all components of $\mathbb{L}$ at hand. It remains to verify whether the satisfaction condition holds.

Proposition 1. $\mathbb{L} = \langle \mathbf{Set}, \mathbf{Mod}^{\mathbb{L}}, \mathbf{Sen}^{\mathbb{L}}, \models^{\mathbb{L}} \rangle$ *is an institution; for every map $f : \mathcal{A} \to \mathcal{B}$ in* **Set**, $M \in \mathbf{Mod}^{\mathbb{L}}(\mathcal{B})$ *and* $\varphi \in \mathbf{Sen}^{\mathbb{L}}(\mathcal{A})$,

$$M|_f \models^{\mathbb{L}}_{\mathcal{A}} \varphi \quad \textit{iff} \quad M \models^{\mathbb{L}}_{\mathcal{B}} f(\varphi).$$

The proof uses the fact that the reduct operation essentially amounts to action relabelling—the idea of the proof is the same as for the institution in [12] and the technical details can be found in the extended version of this paper [13].

2.2 Local Specification of E-Course Management Example

Using the local logic we specify some aspects of the example scenario. We start with the local specification of a student service—a component representing a student using the e-course registration system. In the sequel, we refer to this component simply as *student*. Its observable actions are

$$\mathcal{A}_S = \{\textit{ask}!, \textit{answer}?, \textit{choose}!, \textit{register}!\}$$

representing a request for a list of available courses (*ask*!), receiving the list (*answer*?), choosing courses from the list (*choose*!), and registering to the chosen courses (*register*!). For ease of readability, actions initiated by the student (e.g., requests to external services) end with "!", while names of the actions for which it is a receiving party (e.g., responses from those services) end with "?".

In the axioms we use a number of derived operators. These include the standard temporal operators F (finally) and G (globally), their past counterparts P (sometimes in the past) and H (always in the past), as well as some non-standard operators parametrised with sets of action names.

$$\begin{aligned}
\mathsf{F}\,\varphi &\equiv \top\,\mathsf{U}\,\varphi & \mathsf{X}_{\mathcal{A}}\,\varphi &\equiv \mathsf{X}\left((\neg \textstyle\bigvee_{a\in\mathcal{A}} a)\,\mathsf{U}\,\varphi\right)\\
\mathsf{P}\,\varphi &\equiv \top\,\mathsf{S}\,\varphi & \mathsf{Y}_{\mathcal{A}}\,\varphi &\equiv \mathsf{Y}\left((\neg \textstyle\bigvee_{a\in\mathcal{A}} a)\,\mathsf{S}\,\varphi\right)\\
\mathsf{G}\,\varphi &\equiv \neg\mathsf{F}\,\neg\varphi & \varphi_1\,\mathsf{U}_{\mathcal{A}}\,\varphi_2 &\equiv \left(\varphi_1 \vee \neg \textstyle\bigvee_{a\in\mathcal{A}} a\right)\,\mathsf{U}\,\varphi_2\\
\mathsf{H}\,\varphi &\equiv \neg\mathsf{P}\,\neg\varphi & \varphi_1\,\mathsf{S}_{\mathcal{A}}\,\varphi_2 &\equiv \left(\varphi_1 \vee \neg \textstyle\bigvee_{a\in\mathcal{A}} a\right)\,\mathsf{S}\,\varphi_2
\end{aligned}$$

The axioms for the student component read as follows:

(1) $\mathsf{G}\,(\textit{ask}! \Rightarrow \mathsf{X}_{\mathcal{A}_S}\,\textit{answer}?)$ (3) $\mathsf{G}\,(\textit{choose}! \Rightarrow \mathsf{Y}_{\mathcal{A}_S}\,\textit{answer}?)$
(2) $\mathsf{G}\,(\textit{answer}? \Rightarrow \mathsf{Y}_{\mathcal{A}_S}\,\textit{ask}!)$ (4) $\mathsf{G}\,(\textit{register}! \Rightarrow \mathsf{Y}_{\mathcal{A}_S}\,\textit{choose}!)$

Axiom (1) states that after requesting a list of courses (*ask*!) the student expects a reply (*answer*?). By axiom (2), a reply may be observed only as the next action after the request is sent. We use $\mathsf{X}_{\mathcal{A}_S}$ and $\mathsf{Y}_{\mathcal{A}_S}$ here, rather than X and Y, since transitions labelled with *ask*! and *answer*? may be separated by some transitions not labelled with any name from $\mathcal{A}_S$; the exact number of those transitions is considered an "implementation detail" that should be left unspecified. This is achieved by using $\mathsf{X}_{\mathcal{A}_s}$ which skips over transitions not labelled with elements of $\mathcal{A}_S$. The remaining axioms specify that choosing a course may occur only as the next action after the reply is received and that registering may only occur as the next action after choosing a course.

From the description of the e-course management in the Introduction we may infer two additional services: *course* and *management* with respective alphabets:

$$\begin{aligned}
\mathcal{A}_C &= \{\textit{request}?, \textit{reply}!, \textit{register}?\}\\
\mathcal{A}_M &= \{\textit{ask}?, \textit{answer}!, \textit{request}!, \textit{reply}?, \textit{select}!\}
\end{aligned}$$

where *request*? and *reply*! are the request for an offer from a course provider and its response and *select*! represents the selection of a course performed by the course management for a student. The specifications for *course* and *management* in the local logic are given in Appendix A.

3. GLOBAL LOGIC

At the global level we describe the choreography of services. We first describe the structures that will serve as models for assemblies of interacting service components and then introduce a global choreography description language that extends the local logic (see Sect. 2) with component variables, quantification over components and atomic formulae for describing synchronisations of actions of components.

Models at the global level represent communities of interacting components; these communities may change their communication configuration over time. We start with a family $\{T_n\}_{n\in N}$ of lts's, where N is an arbitrary set. Each $T_n = \langle A_n, S_n, r_n, \to_n \rangle$ represents one component of the modelled system. For $n \in N$ and $a \in A_n$, by $n.a$ we denote the corresponding element of the disjoint union $\coprod_{n\in N} A_n$.

Definition 5. A *global state* of $\{T_n\}_{n\in N}$ is a tuple $\langle s_n \rangle_{n\in N}$ of states, with $s_n \in S_n$, for every $n \in N$. A *global step* of

$\{T_n\}_{n \in N}$ is a triple

$$\big\langle \langle s_n \rangle_{n \in N}, A, \langle s'_n \rangle_{n \in N} \big\rangle$$

where $\langle s_n \rangle_{n \in N}$ and $\langle s'_n \rangle_{n \in N}$ are global states of $\{T_n\}_{n \in N}$ and $A \subseteq \coprod_{n \in N} A_n$ is such that $A \neq \emptyset$ and, for every $n \in N$,

- either $A_n \cap A = \emptyset$ and $s_n = s'_n$,
- or $A_n \cap A = \{a\}$ for some a such that $s_n \xrightarrow{a}_n s'_n$.

Thus a global step represents a *collective* action performed by a subset of components. We assume two actions may occur together only if they are *synchronised*; independent actions performed concurrently by different components will be interleaved.

Formally, possible collective actions are represented by equivalence classes of a *synchronisation relation* for $\{T_n\}_{n \in N}$, i.e., any equivalence relation $\sim$ on $\coprod_{n \in N} A_n$ such that, for every $A \in (\coprod_{n \in N} A_n)/\sim$ and $n \in N$, $A \cap A_n$ has at most one element. The set of all synchronisation relations for $\{T_n\}_{n \in N}$ is denoted by $\mathit{Sync}(\{T_n\}_{n \in N})$.

In order to model typical scenarios in service-oriented computing we have to allow for dynamic changes in the system configuration. For this we introduce a structure in which system configurations, as given by synchronisation relations, can vary from state to state.

Definition 6. A *global frame* $G = \big\langle N, \{T_n\}_{n \in N}, S, r, \rightarrow \big\rangle$ consists of

- a set (possibly infinite) N of *components*,
- for every $n \in N$, an lts $T_n = \langle A_n, S_n, r_n, \rightarrow_n \rangle$,
- a set $S \subseteq \big(\prod_{n \in N} S_n\big) \times \mathit{Sync}(\{T_n\}_{n \in N})$ of *global configurations*,
- an element r of S, called the *initial global configuration*,
- a relation $\rightarrow \subseteq S \times \mathscr{P}(\coprod_{n \in N} A_n) \times S$, called the *global transition relation*,

which together satisfy the following conditions

(1) $r = \big\langle \langle r_n \rangle_{n \in N}, \sim_r \big\rangle$, for some $\sim_r \in \mathit{Sync}(\{T_n\}_{n \in N})$;

(2) for every $\langle s, \sim \rangle, \langle s', \sim' \rangle \in S$ and $A \subseteq \coprod_{n \in N} A_n$, if $\langle s, \sim \rangle \xrightarrow{A} \langle s', \sim' \rangle$ then $s \xrightarrow{A} s'$ is a global step of $\{T_n\}_{n \in N}$ (see Def. 5) and $A \in (\coprod_{n \in N} A_n)/\sim$.

(3) for every $\langle s, \sim \rangle \in S$, $s' \in \prod_{n \in N} S_n$ and $A \subseteq \coprod_{n \in N} A_n$, if $s \xrightarrow{A} s'$ is a global step of $\{T_n\}_{n \in N}$ and $A \in (\coprod_{n \in N} A_n)/\sim$ then there exists $\sim' \in \mathit{Sync}(\{T_n\}_{n \in N})$ such that $\langle s, \sim \rangle \xrightarrow{A} \langle s', \sim' \rangle$.

By (1), in the global initial configuration all the components are in their initial states. By (2), all transitions enabled in a global configuration are compatible with the synchronisation in this state. Finally, (3) is a *progress condition* saying that if all components involved in a collective action may proceed locally then the collective action is enabled. Note that in this case the collective action may lead to several global configurations with different synchronisation relations.

3.1 Institution $\mathbb{G}$ for Global Logic

The global logic extends the local logic with component variables, quantification over components and atomic formulae for describing synchronisation of actions. Using component variables we will be able to say, e.g., that an action occurs at a particular component. Formulae of the global logic are classified by *global signatures*.

Definition 7. A *global signature* $\Gamma = \langle \mathcal{C}, \langle \mathcal{A}_c \rangle_{c \in \mathcal{C}} \rangle$ consists of a set $\mathcal{C}$ of *classes*[2] and, for each $c \in \mathcal{C}$, a set $\mathcal{A}_c$ of *actions of class* c, such that $\mathcal{A}_c \cap \mathcal{A}_d = \emptyset$ when $c \neq d$.

A *global signature morphism* $\gamma : \langle \mathcal{C}, \langle \mathcal{A}_c \rangle_{c \in \mathcal{C}} \rangle \rightarrow \langle \mathcal{C}', \langle \mathcal{A}'_c \rangle_{c \in \mathcal{C}'} \rangle$ is a tuple $\gamma = \langle \kappa, \langle \alpha_c \rangle_{c \in \mathcal{C}} \rangle$, where $\kappa : \mathcal{C} \rightarrow \mathcal{C}'$ and, for every $c \in \mathcal{C}$, $\alpha_c : \mathcal{A}_c \rightarrow \mathcal{A}'_{\kappa(c)}$. Global signatures with global signature morphisms form a category $\mathbf{Sign}^{\mathbb{G}}$.

Recall that a local $\mathcal{A}$-model is an lts enriched with an interpretation of symbols from $\mathcal{A}$ (see Def. 3). Similarly, a global Γ-model will consist of a global frame (see Def. 6) and an interpretation of symbols from Γ in this structure. As in $\mathbb{L}$, the reduct functor in the global institution does not affect underlying structures (i.e., global frames) but only changes the interpretation part of models.

Definition 8. A *global model* $M = \big\langle G, \big\langle c^M \big\rangle, \big\langle a_n^M \big\rangle \big\rangle$ for a global signature $\Gamma = \langle \mathcal{C}, \langle \mathcal{A}_c \rangle_{c \in \mathcal{C}} \rangle$ (a Γ-*model*) consists of

- a global frame $G = \big\langle N, \{T_n\}_{n \in N}, S, r, \rightarrow \big\rangle$,
- for every $c \in \mathcal{C}$, a non-empty set $c^M \subseteq N$,
- for every $c \in \mathcal{C}$, $a \in \mathcal{A}_c$, $n \in c^M$, an action $a_n^M \in A_n$.

The class of all global Γ-models is denoted by $\mathbf{Mod}^{\mathbb{G}}(\Gamma)$.

Definition 9. Let $\gamma : \Gamma \rightarrow \Delta$ be a signature morphism, with $\gamma = \big\langle \kappa, \langle \alpha_c \rangle_{c \in \mathcal{C}} \big\rangle$, and let $M = \big\langle G, \big\langle c^M \big\rangle, \big\langle a_n^M \big\rangle \big\rangle$ be a Δ-model. The γ-reduct of M is a Γ-model

$$M|_\gamma = \big\langle G, \langle c^{M|_\gamma} \rangle, \langle a_n^{M|_\gamma} \rangle \big\rangle$$

where

- for every $c \in \mathcal{C}$, $c^{M|_\gamma} = \kappa(c)^M$,
- for every $c \in \mathcal{C}$, $n \in c^M$ and $a \in \mathcal{A}_c$, $a_n^{M|_\gamma} = \alpha_c(a)_n^M$.

In the sequel we assume a functor $\mathcal{V}$ giving, for every signature $\Gamma = \big\langle \mathcal{C}, \langle \mathcal{A}_c \rangle_{c \in \mathcal{C}} \big\rangle$, a $\mathcal{C}$-sorted set $\mathcal{V}(\Gamma)$ and, for every signature morphism $\gamma = \big\langle \kappa, \langle \alpha_c \rangle_{c \in \mathcal{C}} \big\rangle : \Gamma \rightarrow \Delta$, a tuple of maps $\big\langle \mathcal{V}(\gamma)_c : \mathcal{V}(\Gamma)_c \rightarrow \mathcal{V}(\Gamma)_{\kappa(c)} \big\rangle_{c \in \mathcal{C}}$ such that the induced map $\coprod_{c \in \mathcal{C}} \mathcal{V}(\gamma)_c : \coprod_{c \in \mathcal{C}} \mathcal{V}(\Gamma)_c \rightarrow \coprod_{c \in \mathcal{C}} \mathcal{V}(\Delta)_{\kappa(c)}$ is an injection. An element $k \in \mathcal{V}(\Gamma)_c$ is a *variable of class* c.

Definition 10. Let $\Gamma = \langle \mathcal{C}, \langle \mathcal{A}_c \rangle_{c \in \mathcal{C}} \rangle$ be a global signature. The formulae of the *global logic* over Γ are generated by the following grammar:

$$\begin{aligned}\varphi \ ::= \ & \bot \mid \varphi_1 \Rightarrow \varphi_2 \mid \mathsf{X}\,\varphi \mid \mathsf{Y}\,\varphi \mid \varphi_1 \,\mathsf{U}\, \varphi_2 \mid \varphi_1 \,\mathsf{S}\, \varphi_2 \mid \mathsf{A}\,\varphi \mid \\ & \forall\, k{:}c\ \varphi \mid k_1^{c_1} = k_2^{c_2} \mid k^c.a \mid k_1^{c_1}.a_1 \sim k_2^{c_2}.a_2\end{aligned}$$

where $c \in \mathcal{C}$, $k \in \bigcup \mathcal{V}(\Gamma)_c$ and $a \in \bigcup \mathcal{A}_c$. Additionally, we require that a formula is well-typed, i.e., in every $k : c$ and k^c we have $k \in \mathcal{V}(\Gamma)_c$, in every $k_1^{c_1} = k_2^{c_2}$ we have $c_1 = c_2$ and in every $k^c.a$ we have $a \in \mathcal{A}_c$.

[2]In the sense of object-oriented programming.

The notions of free and bound variables are standard. We omit class names in bound occurrences of variables, writing e.g. $k_1.a_1 \sim k_2.a_2$ instead of $k_1^{c_1}.a_1 \sim k_2^{c_2}.a_2$, whenever c_1 and c_2 may be inferred from the context.

As usual, a *sentence* is a formula with no free variables. For a global signature morphism $\gamma = \langle \kappa, \langle \alpha_c \rangle_{c \in \mathcal{C}} \rangle : \Gamma \to \Delta$ the translation of a Γ-sentence φ to a Δ-sentence $\gamma(\varphi)$ is defined in a straightforward way: class names are translated using κ, variables are translated using $\mathcal{V}(\gamma)_c$ and action names are translated using α_c. All connectives are preserved. It is easy to verify that the translation is functorial, that is, $\gamma; \delta(\varphi) = \delta(\gamma(\varphi))$ for every global $\delta : \Delta \to \Upsilon$. This defines the sentence functor $\mathbf{Sen}^{\mathbb{G}} : \mathbf{Sign}^{\mathbb{G}} \to \mathbf{Set}$.

Let M be a model with a frame $G = \langle N, \{T_n\}_{n \in N}, S, r, \to \rangle$. Formulae of the global logic will express global behaviour of the system represented by M. Formally, this behaviour is represented by the lts $LTS(M) = \langle \mathscr{P}(\coprod A_n), S, r, \to \rangle$ and thus the satisfaction relation refers to runs in this lts.

An *M-valuation* ξ is a $\mathcal{C}$-sorted map $\langle \xi_c : \mathcal{V}(\Gamma)_c \to c^M \rangle_{c \in \mathcal{C}}$.

Definition 11. Let $\Gamma = \langle \mathcal{C}, \langle A_c \rangle_{c \in \mathcal{C}} \rangle$ be a signature and let $M = \langle G, \langle c^M \rangle, \langle a_n^M \rangle \rangle$ be a Γ-model. For $k \in \mathcal{V}$, $c \in \mathcal{C}$, $a \in \mathcal{A}_c$ and an M-valuation ξ we let $(k^c.a)_\xi^M$ denote the action $\xi_c(k).a_{\xi_c(k)}^M$. We define the relation $\models_M$ by structural induction on formulae as follows: for every run $\rho = \langle s_0, \sim_0 \rangle \xrightarrow{[a_1]_{\sim_0}} \langle s_1, \sim_1 \rangle \xrightarrow{[a_2]_{\sim_1}} \dots$ in $LTS(M)$, finite $i \leq |\rho|$ and an M-valuation ξ,

- $\rho, i, \xi \models_M \mathsf{X}\,\varphi$ iff $i < |\rho|$ and $\rho, (i+1), \xi \models_M \varphi$;
- $\rho, i, \xi \models_M \mathsf{Y}\,\varphi$ iff $i > 0$ and $\rho, (i-1), \xi \models_M \varphi$;
- $\rho, i, \xi \models_M \varphi_1 \,\mathsf{U}\, \varphi_2$ iff there exists a finite j such that $i \leq j \leq |\rho|$ and $\rho, j, \xi \models_M \varphi_2$, and for every i', if $i \leq i' < j$ then $\rho, i', \xi \models_M \varphi_1$.
- $\rho, i, \xi \models_M \varphi_1 \,\mathsf{S}\, \varphi_2$ iff there exists j such that $j \leq i$ and $\rho, j, \xi \models_M \varphi_2$, and for every i', if $j < i' \leq i$ then $\rho, i', \xi \models_M \varphi_1$.
- $\rho, i, \xi \models_M \mathsf{A}\,\varphi$ iff for every run ρ' such that $\rho|_i = \rho'|_i$, $\rho', i, \xi \models_M \varphi$.
- $\rho, i, \xi \models_M k_1^{c_1} = k_2^{c_2}$ iff $\xi_{c_1}(k_1) = \xi_{c_2}(k_2)$;
- $\rho, i, \xi \models_M k^c.a$ iff $i < |\rho|$ and $(k^c.a)_\xi^M \in [a_{i+1}]_{\sim_i}$;
- $\rho, i, \xi \models_M k_1^{c_1}.a_1 \sim k_2^{c_2}.a_2$ iff $(k_1^{c_1}.a_1)_\xi^M \sim_i (k_2^{c_2}.a_2)_\xi^M$;
- $\rho, i, \xi \models_M \forall k\!:\!c\,\varphi$ iff for every $n \in c^M$, $\rho, i, \xi[n/k^c] \models_M \varphi$;

Standard clauses for propositional connectives are omitted.

M satisfies a formula φ, written $M \models_\Gamma \varphi$, if $\rho, 0, \xi \models_M \varphi$ holds for every run ρ in $LTS(M)$ and every M-valuation ξ.

Note that the clauses for the temporal operators follow Def. 2. Quantification and variable equality are interpreted as in the first-order many-sorted logic. A formula $k.a$ states that the next collective action along the current path involves the action a of the component referred to by k. A formula $k_1.a_1 \sim k_2.a_2$ states that in the current global state the action a_1 of the component k_1 is synchronised with the action a_2 of the component k_2. Note that $k_1.a_1 \sim k_2.a_2$ is strictly stronger than $\mathsf{A}\,(k_1.a_1 \Leftrightarrow k_2.a_2)$ and strictly weaker than $\mathsf{E}\,(k_1.a_1 \wedge k_2.a_2)$. In fact, action synchronisation cannot be expressed in terms of other operators.

Proposition 2. *$\mathbb{G} = \langle \mathbf{Sign}^{\mathbb{G}}, \mathbf{Mod}^{\mathbb{G}}, \mathbf{Sen}^{\mathbb{G}}, \models^{\mathbb{G}} \rangle$ is an institution; for every morphism $\gamma : \Gamma \to \Delta$ in $\mathbf{Sign}^{\mathbb{G}}$, $M \in \mathbf{Mod}^{\mathbb{G}}(\Delta)$ and $\varphi \in \mathbf{Sen}^{\mathbb{G}}(\Gamma)$,*

$$M|_\gamma \models_\Gamma^{\mathbb{G}} \varphi \iff M \models_\Delta^{\mathbb{G}} \gamma(\varphi) \ .$$

The proof can be found in [13].

3.2 Global Specification of E-Course Management Example

We exemplify the global logic by specifying a choreography for the e-course management system. First, we introduce some convenient abbreviations: given a global signature $\langle \mathcal{C}, \langle \mathcal{A}_c \rangle_{c \in \mathcal{C}} \rangle$ and a variable k^c, for some $c \in \mathcal{C}$ with finite $\mathcal{A}_c$, we define the following derived operators, corresponding to $\mathsf{X}_\mathcal{A}$ and $\mathsf{Y}_\mathcal{A}$ in the local logic (see Sect. 2.2):

$$\mathsf{X}_k^{\mathcal{A}_c}\,\varphi \equiv \mathsf{X}\,\neg(\bigvee_{a \in \mathcal{A}_c} k^c.a)\,\mathsf{U}\,\varphi, \qquad \mathsf{Y}_k^{\mathcal{A}_c}\,\varphi \equiv \mathsf{Y}\,\neg(\bigvee_{a \in \mathcal{A}_c} k^c.a)\,\mathsf{S}\,\varphi$$

For the local logic, in Sect. 2.2, we have identified the three services *student*, *course* and *management*. The global signature of the e-course management specification is therefore $\langle \mathcal{C}, \langle \mathcal{A}_c \rangle_{c \in \mathcal{C}} \rangle$ where $\mathcal{C} = \{\mathbf{student}, \mathbf{course}, \mathbf{mngm}\}$ and the sets of actions for component classes are given as in Sect. 2.2.

As an initial condition we require that everything starts with a student asking for courses:

$$\exists s\colon \mathbf{student} \cdot s.\mathit{ask}!$$

We ensure that a student asking for courses is connected to a management service by requiring their *ask* actions to be synchronised; the same must hold true for their *answer* actions when the management service replies to the student:

$$\begin{aligned}&\forall s\colon \mathbf{student} \cdot \mathsf{G}\,(s.\mathit{ask}! \Rightarrow \exists m\colon \mathbf{mngm}\cdot \\ &\qquad s.\mathit{ask}! \sim m.\mathit{ask}? \wedge \mathsf{F}\,(s.\mathit{answer}? \wedge m.\mathit{answer}!))\end{aligned}$$

Later on the management service will have gathered all available courses and go on to select suitable ones; we require that when doing the selection at least one course is indeed available, i.e., connected to the management service; this is again expressed by requiring synchronisation:

$$\begin{aligned}&\forall m\colon \mathbf{mngm} \cdot \mathsf{G}\,((\mathsf{Y}_m^{\mathcal{A}_M}\, m.\mathit{select}!) \Rightarrow \\ &\qquad \exists c\colon \mathbf{course} \cdot m.\mathit{request}! \sim c.\mathit{request}?)\end{aligned}$$

A more complete specification is given in Appendix A.

4. HETEROGENEOUS SPECIFICATION OF SERVICE SYSTEMS

One of the advantages of institutions is that they provide a framework in which various specification-building operations can be given semantics independent of any particular institution ([3, 14]). This can be extended to a heterogeneous setting where the overall specification is built out of components coming from different institutions ([5]). A prerequisite for such a heterogeneous approach is the ability to "translate" specifications between different institutions, just like they can be translated along signature morphisms within a single institution. For this, institution morphisms and comorphisms are introduced ([8]). Here we will only need the latter notion.

Given institutions $\mathbb{I} = \langle \mathbf{Sign}, \mathbf{Mod}, \mathbf{Sen}, \models \rangle$ and $\mathbb{I}' = \langle \mathbf{Sign}', \mathbf{Mod}', \mathbf{Sen}', \models' \rangle$, an *institution comorphism* from $\mathbb{I}$ to $\mathbb{I}'$ is a triple $\langle \Phi, \alpha, \beta \rangle$ where $\Phi : \mathbf{Sign} \to \mathbf{Sign}'$ is a functor and $\alpha : \mathbf{Sen} \to \Phi; \mathbf{Sen}'$ and $\beta : \Phi; \mathbf{Mod}' \to \mathbf{Mod}$ are natural transformations such that the following condition holds for every $\Sigma \in \mathbf{Sign}$, $M' \in \mathbf{Mod}'(\Phi(\Sigma))$ and $\varphi \in \mathbf{Sen}(\Sigma)$:

$$M' \models'_{\Phi(\Sigma)} \alpha_\Sigma(\varphi) \quad \text{iff} \quad \beta_\Sigma(M') \models_\Sigma \varphi \ .$$

Intuitively, a comorphism from $\mathbb{I}$ to $\mathbb{I}'$ shows how $\mathbb{I}$ can be *represented* in $\mathbb{I}'$: for every signature Σ in $\mathbb{I}$, $\Phi(\Sigma)$ is the corresponding signature in $\mathbb{I}'$; for every Σ-sentence φ, $\alpha_\Sigma(\varphi)$ is the corresponding $\Phi(\Sigma)$-sentence; from every $\Phi(\Sigma)$-model M' we can retrieve the corresponding Σ-model $\beta_\Sigma(M)$.

In our case, it is more straightforward to relate a single global model M to a whole set of local models, each corresponding to one component of M. In order to fit this approach into the framework of institution comorphisms we introduce a "powerset" version of the local institution as follows: Signatures and sentences of $\mathscr{P}(\mathbb{L})$ are those of $\mathbb{L}$; for every signature $\mathcal{A}$, $\mathbf{Mod}^{\mathscr{P}(\mathbb{L})}(\mathcal{A})$ is the class of all sets of local $\mathcal{A}$-models; for every $\alpha : \mathcal{A} \to \mathcal{B}$ and $\mathbb{M} \subseteq \mathbf{Mod}^{\mathscr{P}(\mathbb{L})}(\mathcal{B})$, $\mathbf{Mod}^{\mathscr{P}(\mathbb{L})}(\alpha)(\mathbb{M})$ is obtained by applying $\mathbf{Mod}^{\mathbb{L}}(\alpha)$ to each element of $\mathbb{M}$; finally, $\mathbb{M} \models^{\mathscr{P}(\mathbb{L})}_{\mathcal{A}} \varphi$ holds if $M \models^{\mathbb{L}}_{\mathcal{A}} \varphi$ holds for every $M \in \mathbb{M}$.

Now, a link between $\mathbb{G}$ and $\mathbb{L}$ can be formalised as a comorphism from $\mathscr{P}(\mathbb{L})$ to $\mathbb{G}$. Each global model M in $\mathbb{G}$ will be mapped to a set of local models, each obtained by "projecting" M onto one of its components. Formally, for a global frame $G = \langle N, \{T_n\}_{n\in N}, S, r, \to\rangle$ and $n \in N$ we define an lts $G|_n = \langle A_n \cup \{\tau\}, S, r, \to_n\rangle$ where τ is a name not in A_n and $\to_n$ is defined as follows:

$$\begin{aligned}\to_n \ &= \big\{\langle s, a, t\rangle \mid s \xrightarrow{A} t, A_n \cap A = \{a\}\big\} \\ &\cup \big\{\langle s, \tau, t\rangle \mid s \xrightarrow{A} t, A_n \cap A = \emptyset\big\}.\end{aligned}$$

PROPOSITION 3. *The following data form an institution comorphism $\langle \Gamma, \alpha, \beta\rangle : \mathscr{P}(\mathbb{L}) \to \mathbb{G}$:*

- *for every set $\mathcal{A}$, $\Gamma(\mathcal{A}) = \langle\{*\}, \{* \mapsto \mathcal{A}\}\rangle$, where $*$ is a "dummy" class name;*
- *for every set $\mathcal{A}$ and a sentence $\varphi \in$ aPCTL$^*(\mathcal{A})$, $\alpha_\mathcal{A}(\varphi)$ is $\forall k{:}*\, k.\varphi$, where $k.\varphi$ is a formula obtained by replacing each occurrence of $a \in \mathcal{A}$ in φ by $k.a$;*
- *for every set $\mathcal{A}$ and a $\Gamma(\mathcal{A})$-model M with a frame G, $\beta_\mathcal{A}(M)$ is the set $\{M|_n \mid n \in *^M\}$, where $M|_n = \big\langle G|_n, \langle a^{M|_n}\rangle_{a\in\mathcal{A}}\big\rangle$ and $a^{M|_n} = \alpha^M_n(a)$, for each $a \in \mathcal{A}$.*

For the proof see [13].

4.1 Heterogeneous Service Specifications

The institution comorphism from the local to the global level allows us to combine specifications of service components in the local and the global logic into a single specification. Let $\mathcal{C}$ be a set of service classes with a local signature $\mathcal{A}_c$ and a local specification Sp_c for each $c \in \mathcal{C}$. Let Sp be a global specification over the global signature $\Gamma = \big\langle \mathcal{C}, \langle \mathcal{A}_c\rangle_{c\in\mathcal{C}}\big\rangle$. Then

$$\big\langle \langle c : Sp_c\rangle_{c\in\mathcal{C}}, Sp\big\rangle$$

is a *heterogeneous service specification.* Its signature is Γ and its class of *models* M is given by

$$\{M \in \mathbf{Mod}(\Gamma) \mid M \models Sp \wedge \forall c \in \mathcal{C} \,.\, M \models \alpha_{\mathcal{A}_c}(Sp_c)\{* \mapsto c\}\},$$

where $\alpha_{\mathcal{A}_c}$ embeds specifications of the local logic into the global logic and $\{* \mapsto c\}$ substitutes the class name $*$ by c in the specification Sp_c for adapting the service classes. In particular, the notion of specification refinement (see Sect. 2.1) transfers to heterogeneous service specifications.

For the e-course management example, denoting the local specifications for *student*, *course*, and *management* (see Sect. 2.2) by Sp_S, Sp_C, and Sp_M, respectively; and letting the global choreography specification for the e-course management (see Sect. 3.2) be Sp_E the overall service specification for the e-course management service system is

$$\big\langle \langle \mathbf{student} : Sp_S,\ \mathbf{course} : Sp_C,\ \mathbf{mngm} : Sp_M\rangle,\ Sp_E\big\rangle.$$

4.2 Refinement of Specification of E-Course Management

Suppose we want to refine the description of our example scenario by splitting the course selection process into two phases: the management service first selects the continuations of previously taken courses, and then selects other feasible courses taking into account their time and place.

The refinement concerns only the component class **mngm**, thus we just refine its local specification. We extend the signature to $\mathcal{A}_{M1} = \mathcal{A}_M \cup \{\mathit{sel\_continuation!}, \mathit{sel\_feasible!}\}$. We extend the local specification of the service component **mngm** into a specification Sp_{M1} adding the axiom

$$\begin{gathered}\mathsf{G}\,(\mathit{reply?} \wedge \neg \mathsf{X}_{\mathcal{A}_M}\, \mathit{reply?} \Rightarrow \\ \mathsf{X}_{\mathcal{A}_{M1}}\,(\mathit{sel\_continuation!} \wedge \mathsf{X}_{\mathcal{A}_{M1}}\,(\mathit{sel\_feasible!} \wedge \mathsf{F}\,\mathit{select!}))\end{gathered}$$

which basically says that the two new actions occur after the last *reply?* and before *select!*. Notice that since in Sp_M we used $\mathsf{X}_{\mathcal{A}_M}$ and $\mathsf{Y}_{\mathcal{A}_M}$, the axioms still hold even though we put some actions between the last *reply?* and *select!*. It is so, because none of new actions belongs to $\mathcal{A}_M$. Thus we have $Sp_M \rightsquigarrow Sp_{M1}$. This refinement relation extends to the overall specifications Sp_{ecm} and $\langle\langle \mathbf{student} : Sp_S, \mathbf{course} : Sp_C, \mathbf{mngm} : Sp_{M1}\rangle, Sp_{E1}\rangle$ where Sp_{E1} is Sp_E but with its signature extended by the new actions for **mngm**.

5. RELATED WORK

The theory of institutions ([3]) and its subsequent development into a powerful framework for distributed heterogeneous specifications ([15, 4, 16, 5]) provide the mathematical foundations for our approach. A recent application of heterogeneous specifications is [6] where institutions for several types of UML diagrams are related with comorphisms.

Various institutions for temporal and modal logics are directly related to our work. In [17] several such institutions are presented in a uniform way as so-called *β-institutions* (i.e., institutions for behaviour). Our local institution can also be fit into this approach. The institution of [12] resembles our local institution and uses lts's as models, but its logic contains also first-order constructs. An institution for CTL* and Kripke structures is given in [18].

Institutions aside, our local logic is related to other action-based variants of CTL or CTL*, such as ACTL* ([19]). Differences are mostly syntactic, since action-based temporal logics usually use action names as *modalities*, e.g., a formula $\langle a\rangle\varphi$ means that the action a is enabled and after it occurs, φ holds. Recently, such temporal logics have been used in the context of service-oriented computing. Examples include UCTL ([20])—an action and state-based logic which was originally designed for expressing properties of UML statecharts, SocL ([21])—a variant of UCTL used for expressing properties of the service-oriented process calculus COWS ([22]), and μUCTL (UCTL extended with fixed point operators) for specifying asynchronous service interactions ([2]), in connection with SRML ([23]). The distributed state temporal logic of [24] is an extension of Unity for formalising policies; it includes an operator to deal with events and modalities to localise properties of system components. Similarly, in [25] atomic formulae expressing event occurrence are localised at particular agents, represented by identifiers from a fixed set. However, none of these temporal logics contains quantification over component instances nor allows for a direct specification of configuration, as is possible in our global logic.

The concept of wires in SRML may be regarded as corresponding to our synchronisation relation; wiring can also be dynamic and change at runtime ([23]), but the semantic construction involves discovery and binding and can be seen as an implementation of the synchronisation relation.

6. CONCLUSION AND FUTURE WORK

We have proposed a formal approach to the specification of SOA systems. We employ the theory of institutions ([3]) to define two temporal logics. The first logic is used to specify the individual SOA components, the second logic is used to specify changes to their communication patterns. We connect the two logics by an institution comorphism ([8]) that yields a heterogeneous specifications setting ([5]) where a SOA system can be specified on two levels. Its individual components can be specified in the local logic, whereas their interactions can be separately described on the global level.

As for the temporal part of both the local and the global logic, our choice of CTL* extended with S and Y is rather arbitrary. In particular, past operators have been added purely for convenience in writing example properties. Atomic predicates evaluated in states could have been used for this purpose equally well, but this would lead to slightly more complicated definitions of a signature and a model. If needed, institutions for action-based variants of LTL or CTL can be obtained by restricting the syntax of the local logic appropriately. On the other hand, the language can be extended, e.g., with fixed-point operators, without affecting the model part of the institution. This flexibility is due to the fact that reducing a model along a signature morphism does not alter the transition system underlying a model and thus the satisfaction condition is easy to establish.

On the other hand, using an action-based logic is a natural choice since, at the global level, we use synchronisation of actions to model component interactions. It should be straightforward to enrich the logics with state predicates or first-order constructs for describing data, both on the local and on the global level. A more interesting topic of further work is to generalise the way we build the global logic over the local one to a construction parametrised by an arbitrary temporal logic institution, possibly satisfying some additional requirements.

In our model, components communicate in a purely synchronous manner. This choice may look strange in the context of services that are naturally "loosely-coupled". However, asynchronous communication can be implemented by synchronous channels and, since we allow communication topology to change, the coupling of the components is not that strict at all. Moreover, since global specifications may allow an unlimited number of interacting components, it is fairly easy to model discovery and binding of new services. Nevertheless, it would be interesting to develop a similar formalism based on asynchronous communication in future.

Our specification methodology promotes the separate specification of components in a local logic and confines the global specification to the description of their common behaviour; it thus provides an example of *structured specifications*. However, the methodology currently does not reflect the common practical situation where the components are implemented independently according to local specifications and then used as parameters to the global construction that implements their coordination working consistently for any given set of actual parameters; this would amount to *architectural specifications*. Usually the prerequisite for such specifications is the existence of amalgamation in the institution used for specifications. Fortunately, both of our institutions meet this requirement and we plan to extend our framework to architectural specifications as well.

Acknowledgements.

We would like to thank María Victoria Cengarle and Andrzej Tarlecki for comments and suggestions.

7. REFERENCES

[1] M. Wirsing, A. Clark, S. Gilmore, M. Hölzl, A. Knapp, N. Koch, and A. Schroeder, "Semantic-Based Development of Service-Oriented Systems," in *FORTE'06*, vol. 4229 of *LNCS*, pp. 24–45, Springer, 2006.

[2] J. Abreu, L. Bocchi, J. L. Fiadeiro, and A. Lopes, "Specifying and Composing Interaction Protocols for Service-Oriented System Modelling," in *FORTE'07*, vol. 4574 of *LNCS*, pp. 358–373, Springer, 2007.

[3] J. A. Goguen and R. M. Burstall, "Institutions: Abstract Model Theory for Specification and Programming," *J. ACM*, vol. 39, no. 1, pp. 95–146, 1992.

[4] T. Mossakowski, "Heterogeneous Specification and the Heterogeneous Tool Set." Habilitation thesis, Universität Bremen, 2005.

[5] T. Mossakowski and A. Tarlecki, "Heterogeneous logical environments for distributed specifications," in *WADT'08*, LNCS, Springer, 2008.

[6] M. V. Cengarle, A. Knapp, A. Tarlecki, and M. Wirsing, "A Heterogeneous Approach to UML Semantics," in *Festschrift for Ugo Montanari*, vol. 5065 of *LNCS*, pp. 383–402, Springer, 2008.

[7] Matthias Hölz, "Requirements Modelling and Analysis of Selected Scenarios: University Management and E-Learning Case Study," 2007. Deliverable D8.4.a of the Sensoria project.

[8] J. A. Goguen and G. Rosu, "Institution Morphisms," *Formal Asp. Comput.*, vol. 13, no. 3–5, pp. 274–307, 2002.

[9] A. Boronat, A. Knapp, J. Meseguer, and M. Wirsing, "What is a multi-modeling language?," in *WADT'08*, vol. 5486 of *LNCS*, pp. 71–87, Springer, 2009.

[10] E. A. Emerson, "Temporal and Modal Logic," in *Handbook of Theoretical Computer Science, Volume B*, pp. 995–1072, Elsevier, 1990.

[11] F. Laroussinie and P. Schnoebelen, "A Hierarchy of Temporal Logics with Past," in *STACS'94*, vol. 775 of *LNCS*, pp. 47–58, Springer, 1994.

[12] M. V. Cengarle, "The Temporal Logic institution," Tech. Rep. 9805, Ludwig-Maximilians-Universität München, Institut für Informatik, 1998.

[13] A. Knapp, G. Marczyński, M. Wirsing, and A. Zawłocki, "A Heterogeneous Approach to Service-Oriented Systems Specification (Extended Version)," tech. rep., Institute of Informatics, University of Warsaw, 2009. www.mimuw.edu.pl/~zawlocki/papers/sac2010.pdf.

[14] D. Sannella and A. Tarlecki, "Specifications in an Arbitrary Institution," *Information and Computation*, vol. 76, pp. 165–210, 1988.

[15] A. Tarlecki, "Moving between Logical Systems," in *WADT'95*, vol. 1130 of *LNCS*, pp. 478–502, Springer, 1996.

[16] R. Diaconescu, *Institution-Independent Model Theory*. Birkhäuser, 2008.

[17] J. L. Fiadeiro and J. F. Costa, "Institutions for Behaviour Specification," in *WADT'94*, vol. 906 of *LNCS*, pp. 273–289, Springer, 1994.

[18] M. Palomino, J. Meseguer, and N. Martí-Oliet, "A Categorical Approach to Simulations," in *CALCO'05*, vol. 3629 of *LNCS*, pp. 313–330, 2005.

[19] R. De Nicola and F. Vaandrager, "Action versus State-based Logics for Transition systems," in *LITP Spring School*, vol. 469 of *LNCS*, pp. 407–419, Springer, 1990.

[20] M. H. ter Beek, A. Fantechi, S. Gnesi, and F. Mazzanti, "An Action/State-Based Model-Checking Approach for the Analysis of Communication Protocols for Service-Oriented Applications," in *FMICS'07*, vol. 4916 of *LNCS*, pp. 133–148, 2007.

[21] A. Fantechi, S. Gnesi, A. Lapadula, F. Mazzanti, R. Pugliese, and F. Tiezzi, "A Model Checking Approach for Verifying COWS Specifications," in *FASE'08*, vol. 4961 of *LNCS*, pp. 230–245, Springer, 2008.

[22] A. Lapadula, R. Pugliese, and F. Tiezzi, "A Calculus for Orchestration of Web Services," in *ESOP'07*, vol. 4421 of *LNCS*, pp. 33–47, 2007.

[23] J. L. Fiadeiro, A. Lopes, and L. Bocchi, "A Formal Approach to Service-Oriented Architecture," in *WS-FM'06*, vol. 4184 of *LNCS*, pp. 193–213, Springer, 2006.

[24] C. Montangero, S. Reiff-Marganiec, and L. Semini, "Logic-based Detection of Conflicts in APPEL Policies," in *Int. Symp. Fund. Software Engineering*, vol. 4767 of *LNCS*, pp. 257–271, Springer, 2007.

[25] K. Lodaya, M. Mukund, R. Ramanujam, and P. S. Thiagarajan, "Models and Logics for True Concurrency," Tech. Rep. IMSc/90/12, Madras, 1990.

APPENDIX

A. SPECIFICATIONS OF E-COURSE MANAGEMENT SYSTEM

The example scenario describes an e-course management system. The *student service* acts as a representation of a student to other services of the system. It transfers the data entered by the (human) student to a *course manager service* and requests a list of matching courses. The course manager then contacts all available *course providers* (each provider represents one course) and gathers their replies. Then it selects the courses that match the student's curriculum and university regulations and sends the list to the student service, which in turn lets the student choose from the list. Finally, on the student's request, the student service registers the student to the selected courses.

The local specification Sp_S of a student service consists of the set of actions $\mathcal{A}_S = \{\mathit{ask}!, \mathit{answer}?, \mathit{choose}!, \mathit{register}!\}$ and the four axioms given in Sect. 2.2. The local specification Sp_C of the **course** service consists of the set of actions $\mathcal{A}_C = \{\mathit{request}?, \mathit{reply}!, \mathit{register}?\}$ and the following axioms:

(1) $\mathsf{G}\,(\mathit{request}? \Rightarrow \mathsf{X}_{\mathcal{A}_C}\, \mathit{reply}!)$ (2) $\mathsf{G}\,(\mathit{reply}! \Rightarrow \mathsf{Y}_{\mathcal{A}_C}\, \mathit{request}?)$
(3) $\mathsf{G}\,(\mathit{register}? \Rightarrow \mathsf{Y}_{\mathcal{A}_C}\, \mathit{reply}!)$

The local specification Sp_M of the **mngm** service has $\mathcal{A}_M = \{\mathit{ask}?, \mathit{answer}!, \mathit{request}!, \mathit{reply}?, \mathit{select}!\}$ as the set of actions and the following axioms:

(1) $\mathsf{G}\,(\mathit{ask}? \Rightarrow \mathsf{X}_{\mathcal{A}_M}\, \mathit{request}!)$
(2) $\mathsf{G}\,(\mathit{request}! \Rightarrow (\mathsf{Y}_{\mathcal{A}_M}\, \mathit{ask}?) \wedge (\mathsf{X}_{\mathcal{A}_M}\, \mathit{reply}?))$
(3) $\mathsf{G}\,(\mathit{reply}? \Rightarrow (\mathit{reply}?\, \mathsf{S}_{\mathcal{A}_M}\, \mathit{request}!) \wedge (\mathsf{F}\, \mathit{select}!))$
(4) $\mathsf{G}\,(\mathit{select}! \Rightarrow (\mathsf{Y}_{\mathcal{A}_M}\, \mathit{reply}?) \wedge (\mathsf{X}_{\mathcal{A}_M}\, \mathit{answer}!))$
(5) $\mathsf{G}\,(\mathit{answer}! \Rightarrow \mathsf{Y}_{\mathcal{A}_M}\, \mathit{select}!)$

The signature of the global specification is $\left\langle \mathcal{C}, \langle \mathcal{A}_c \rangle_{c\in\mathcal{C}} \right\rangle$ where $\mathcal{C} = \{\mathbf{student}, \mathbf{course}, \mathbf{mngm}\}$ and the actions for each class are given above. The axioms of the global specification include the three axioms discussed in Sect. 3.2 and the following additional ones:

(1) $\forall m\colon \mathbf{mngm} \cdot \mathsf{G}\,(m.\mathit{request}! \Rightarrow$
$(\exists c\colon \mathbf{course} \cdot m.\mathit{request}! \sim c.\mathit{request}?) \wedge$
$(\forall c\colon \mathbf{course} \cdot m.\mathit{request}! \sim c.\mathit{request}? \Rightarrow$
$\mathsf{F}\,(m.\mathit{reply}? \wedge c.\mathit{reply}!)))$

(2) $\mathsf{G}\,\forall m\colon \mathbf{mngm}, c_1, c_2\colon \mathbf{course} \cdot$
$c_1.\mathit{request}? \sim m.\mathit{request}! \sim c_2.\mathit{request}? \wedge c_1 \neq c_2 \Rightarrow$
$\mathsf{G}\, c_1.\mathit{reply}! \not\sim c_2.\mathit{reply}!$

(3) $\forall m\colon \mathbf{mngm}, c\colon \mathbf{course} \cdot \mathsf{G}\,(m.\mathit{request}! \wedge c.\mathit{request}? \Rightarrow$
$(m.\mathit{request}! \sim c.\mathit{request}?)\, \mathsf{U}\, m.\mathit{select}!)$

(4) $\forall m\colon \mathbf{mngm}, c\colon \mathbf{course}, s\colon \mathbf{student} \cdot$
$\mathsf{G}\,(((\mathsf{Y}_m^{\mathcal{A}_M}\, m.\mathit{select}!) \wedge m.\mathit{request}! \sim c.\mathit{request}?) \Rightarrow$
$(\neg s.\mathit{register}!)\, \mathsf{U}\, (s.\mathit{register}! \sim c.\mathit{register}?))$

(5) $\forall m\colon \mathbf{mngm}, c\colon \mathbf{course}, s\colon \mathbf{student} \cdot$
$\mathsf{G}\,(((\mathsf{Y}_m^{\mathcal{A}_M}\, m.\mathit{select}!) \wedge m.\mathit{request}! \sim c.\mathit{request}?) \Rightarrow$
$\mathsf{E}\,\mathsf{F}\,((\mathsf{Y}_s^{\mathcal{A}_S}\, s.\mathit{choose}!) \wedge s.\mathit{register}! \sim c.\mathit{register}?))$

(6) $\forall c\colon \mathbf{course}, s\colon \mathbf{student} \cdot \mathsf{G}\,((s.\mathit{register}! \wedge c.\mathit{register}?) \Rightarrow$
$\exists m\colon \mathbf{mngm} \cdot \mathsf{P}\,(\mathsf{P}\,(\mathsf{Y}_m^{\mathcal{A}_M}\, m.\mathit{select}!) \wedge$
$m.\mathit{request}! \sim c.\mathit{request}?) \wedge$
$((\mathsf{Y}_s^{\mathcal{A}_S}\, s.\mathit{choose}!) \wedge s.\mathit{register}! \sim c.\mathit{register}?))$

For the discussion of the additional axioms the reader is referred to the extended version of this paper ([13]).