An Institutional Approach to Communicating UML State Machines

Tobias Rosenberger[1,2], Alexander Knapp[3](✉), and Markus Roggenbach[1]

[1] Swansea University, Swansea, U. K.
{t.rosenberger.971978, m.roggenbach}@swansea.ac.uk
[2] VERIMAG, Université Grenoble Alpes, Grenoble, France
[3] Universität Augsburg, Augsburg, Germany
knapp@informatik.uni-augsburg.de

Abstract We present a new approach on how to provide institution-based semantics for communicating UML state machines in form of a hybrid modal logic $\mathcal{M}^{\downarrow}_{\mathcal{D}}$. A theoroidal comorphism maps $\mathcal{M}^{\downarrow}_{\mathcal{D}}$ into the CASL institution. This allows for symbolic reasoning on communicating UML state machines.

1 Introduction

In line with a long-standing line of research [5,6,15,4], we set out on a general programme to bring together multi-view system specification with UML diagrams and heterogeneous specification and verification based on institution theory, giving the different system views both a joint semantics and richer tool support. Institutions, a formal notion of a logic, are a principled way of creating such joint semantics. They make moderate assumptions about the data constituting a logic, give uniform notions of well-behaved translations between logics and, given a graph of such translations, automatically give rise to a joint institution.

UML state machines are an object-based variant of Harel statecharts. Within the UML, state machines are a central means to specify system behaviour. In previous work [16], an institutional semantics for UML state machines was provided that allowed for symbolic reasoning. Such symbolic reasoning can be of advantage as, in principle, it allows to verify properties of UML state machines with large or infinite state spaces. Here, we extend this work in order to cater for communication.

A typical scenario for such communication is the interaction between a `User`, an `ATM`, and a `Bank` in order to authenticate the `User` as legitimate owner of a bank card by checking an entered PIN. Figure 1 depicts a UML modelling for this scenario. In brief, the system consists of the `ATM` and the `Bank`, where we consider `User` interaction as an external communication. The scenario begins with the `User` entering a bank card and a PIN. The `ATM` requests their verification by the `Bank`. The `Bank` checks validity of the card/PIN combination and communicates the result to the `ATM`. We model the validity check as internal, non-deterministic choice made by the `Bank`. In case of a positive result, the `ATM` will return the card to the `User`. In case of a negative result, the `User` is given a

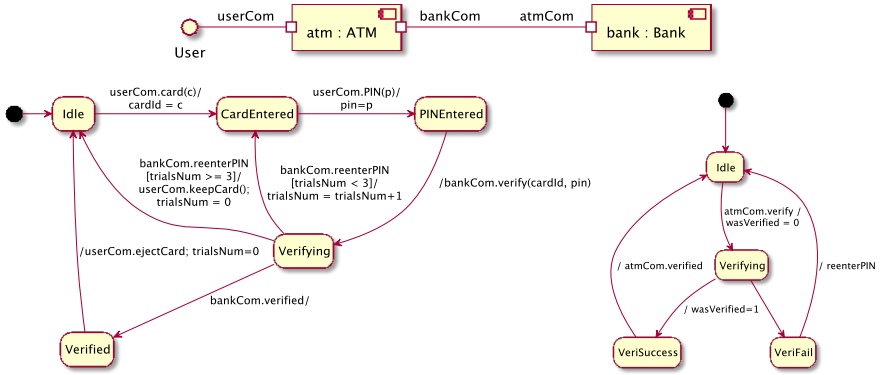

Figure 1. UML diagrams for the `ATM` example (implicit completion events omitted): Composite structure diagram: top; state machine: left `ATM`, right `Bank`.

second and third chance to enter a correct PIN. After the third verification failure, the `ATM` will keep the card. A typical question on this model is whether the `ATM` will consider the verification successful only if the `Bank` has already come to the same conclusion. To answer such questions, one needs to take into account the behaviour of all state machines involved as well as how they can communicate via the ports and connectors as specified by a composite structure diagram.

Closest to our approach are the works [6,4]. Both these papers address the topic of communicating state machines, however, both fail to provide institutions of state machines as reported in [15,16]. Learning from the reason for this shortcoming, rather than capturing UML state machines directly as an institution, [16] builds up a new logic in which UML state machines can be embedded. Here, we extend this logic for communication. In particular, we treat UML event pools as part of composite structure diagrams rather than of state machines. State machines are seen as a completely open system, which is (partially) closed by 'wiring up' in a communication structure. Overall, this leads to a separation of concerns: event pools and transitions can be analysed independently.

A number of authors give formal semantics to communicating state machines, however with a purpose different from symbolic analysis of UML. The Object Management Group provides an executable semantics of UML Composite Structures [14]. Their objective is to provide an interpreter for the executable subset fUML of the UML. Dragomir [12] define transformations from composite structure diagrams to communicating extended timed automata for the purpose of simulation, static analysis and model-checking. Mazzanti et al. [8] provide a UML model checker that also covers composite structure diagrams. A quite comprehensive formal semantics has been provided by Liu et al. [7], again with the main purpose of supporting model checking.

In Section 2, we recall the notion of an institution and sketch the CFOL$^{=}$ institution of CASL, which we use for specifying data. In Section 3, we extend the hybrid modal logic $\mathcal{M}^{\downarrow}_{\mathcal{D}}$ [16] to cater also for output by adding the notion of messages (in [16] with input only). For structures and formulae this requires us to introduce relativisations with

regards to a set of outputs. We show that the extended logic $\mathcal{M}^{\downarrow}_{\mathcal{D}}$ is an institution, can be embedded into CASL via a theoroidal comorphism, and allows for "borrowing" of CASL theorem proving support. In Section 4, we show how to embed simple UML state machines with output into the extended logic $\mathcal{M}^{\downarrow}_{\mathcal{D}}$. In Section 5, we provide an institution for simple UML composite structures by enriching our extended logic $\mathcal{M}^{\downarrow}_{\mathcal{D}}$ with elements capturing connectors and event queues. Again, "borrowing" of CASL theorem proving support is possible. Finally, in Section 6, we demonstrate that our approach allows for automated theorem proving.

2 Background on Institutions and CASL

Institutions are an abstract formalisation of the notion of logical systems combining signatures, structures, sentences, and satisfaction under the slogan "truth is invariant under change of notation" [3]. Institutions can be related in different ways by institution (forward) (co-)morphisms, where a so-called theoroidal institution comorphism covers a particular case of encoding a "poorer" logic into a "richer" one. The algebraic specification language CASL [11] uses an institution of first-order logic at its basic specification level, where mainly signature items and axiom sentences are listed. On its structured specifications level, CASL offers institution-independent combination mechanisms to build larger specifications in a hierarchical and modular fashion. We use CASL's basic institution CFOL$^{=}$ of first-order logic with equality and sort generation constraints [9] and construct a theoroidal institution comorphism from our hybrid modal logic institution $\mathcal{M}^{\downarrow}_{\mathcal{D}}$ to CFOL$^{=}$.

2.1 Institutions and Theoroidal Institution Comorphisms

An *institution* $\mathcal{I} = (\mathbb{S}^{\mathcal{I}}, \mathit{Str}^{\mathcal{I}}, \mathrm{Sen}^{\mathcal{I}}, \models^{\mathcal{I}})$ consists of (i) a category of *signatures* $\mathbb{S}^{\mathcal{I}}$; (ii) a contravariant *structures functor* $\mathit{Str}^{\mathcal{I}} \colon (\mathbb{S}^{\mathcal{I}})^{\mathrm{op}} \to \mathrm{Cat}$, where Cat is the category of (small) categories; (iii) a *sentence functor* $\mathrm{Sen}^{\mathcal{I}} \colon \mathbb{S}^{\mathcal{I}} \to \mathrm{Set}$, where Set is the category of sets; and (iv) a family of *satisfaction relations* $\models^{\mathcal{I}}_{\Sigma} \subseteq |\mathit{Str}^{\mathcal{I}}(\Sigma)| \times \mathrm{Sen}^{\mathcal{I}}(\Sigma)$ indexed over $\Sigma \in |\mathbb{S}^{\mathcal{I}}|$, such that the following *satisfaction condition* holds for all $\sigma \colon \Sigma \to \Sigma'$ in $\mathbb{S}^{\mathcal{I}}$, $\varphi \in \mathrm{Sen}^{\mathcal{I}}(\Sigma)$, and $M' \in |\mathit{Str}^{\mathcal{I}}(\Sigma')|$:

$$\mathit{Str}^{\mathcal{I}}(\sigma)(M') \models^{\mathcal{I}}_{\Sigma} \varphi \iff M' \models^{\mathcal{I}}_{\Sigma'} \mathrm{Sen}^{\mathcal{I}}(\sigma)(\varphi) \,.$$

$\mathit{Str}^{\mathcal{I}}(\sigma)$ is called the *reduct* functor, $\mathrm{Sen}^{\mathcal{I}}(\sigma)$ the *translation* function.

A *theory presentation* $T = (\Sigma, \Phi)$ in the institution $\mathcal{I}$ consists of a signature $\Sigma \in |\mathbb{S}^{\mathcal{I}}|$, also denoted by $\mathit{Sig}(T)$, and a set of sentences $\Phi \subseteq \mathrm{Sen}^{\mathcal{I}}(\Sigma)$. Its *model class* $\mathrm{Mod}^{\mathcal{I}}(T)$ is the class $\{M \in \mathit{Str}^{\mathcal{I}}(\Sigma) \mid M \models^{\mathcal{I}}_{\Sigma} \varphi \text{ f. a. } \varphi \in \Phi\}$ of the Σ-structures satisfying the sentences in Φ. A *theory presentation morphism* $\sigma \colon (\Sigma, \Phi) \to (\Sigma', \Phi')$ is given by a signature morphism $\sigma \colon \Sigma \to \Sigma'$ such that $M' \models^{\mathcal{I}}_{\Sigma'} \mathrm{Sen}^{\mathcal{I}}(\sigma)(\varphi)$ for all $\varphi \in \Phi$ and $M' \in \mathrm{Mod}^{\mathcal{I}}(\Sigma', \Phi')$. Theory presentations in $\mathcal{I}$ and their morphisms form the category $\mathrm{Pres}^{\mathcal{I}}$.

A *theoroidal institution comorphism* $\nu = (\nu^{\mathit{Sig}}, \nu^{\mathrm{Mod}}, \nu^{\mathrm{Sen}}) \colon \mathcal{I} \to \mathcal{I}'$ consists of a functor $\nu^{\mathit{Sig}} \colon \mathbb{S}^{\mathcal{I}} \to \mathrm{Pres}^{\mathcal{I}'}$ inducing the functor $\nu^{\mathbb{S}} = \nu^{\mathit{Sig}}; \mathit{Sig} \colon \mathbb{S}^{\mathcal{I}} \to \mathbb{S}^{\mathcal{I}'}$ on

signatures, a natural transformation $\nu^{\mathrm{Mod}} \colon (\nu^{Sig})^{\mathrm{op}}; \mathrm{Mod}^{\mathcal{I}'} \dot{\to} Str^{\mathcal{I}}$ on models and structures, and a natural transformation $\nu^{\mathrm{Sen}} \colon \mathrm{Sen}^{\mathcal{I}} \to \nu^{\mathbb{S}}; \mathrm{Sen}^{\mathcal{I}'}$ on sentences, such that for all $\Sigma \in |\mathbb{S}^{\mathcal{I}}|$, $M' \in |\mathrm{Mod}^{\mathcal{I}'}(\nu^{Sig}(\Sigma))|$, and $\varphi \in \mathrm{Sen}^{\mathcal{I}}(\Sigma)$ the following *satisfaction condition* holds:

$$\nu^{\mathrm{Mod}}_{\Sigma}(M') \models^{\mathcal{I}}_{\Sigma} \varphi \iff M' \models^{\mathcal{I}'}_{\nu^{\mathbb{S}}(\Sigma)} \nu^{\mathrm{Sen}}(\Sigma)(\varphi) \,.$$

A theory presentation (Σ, Φ) over the institution $\mathcal{I}$ is *translated* via a theoroidal institution comorphism $\nu \colon \mathcal{I} \to \mathcal{I}'$ into the theory presentation $\nu^{\mathrm{Pres}}(\Sigma, \Phi) = (\Sigma_\nu, \Phi_\nu \cup \nu^{\mathrm{Sen}}_{\Sigma}(\Phi))$ over $\mathcal{I}'$ where $\nu^{Sig}(\Sigma) = (\Sigma_\nu, \Phi_\nu)$ and $\nu^{\mathrm{Sen}}_{\Sigma}(\Phi) = \{\nu^{\mathrm{Sen}}_{\Sigma}(\varphi) \mid \varphi \in \Phi\}$.

2.2 CASL and the Institution CFOL$^{=}$

At the level of basic CASL specifications, CFOL$^{=}$ offers declarations of *sorts*, *operations*, and *predicates* with given argument and result sorts. Formally, this defines a *many-sorted signature* $\Sigma = (S, F, P)$ with a set S of sorts, a $S^* \times S$-sorted families $F = (F_{w,s})_{w\, s \in S^+}$ of *total function symbols*, and family $P = (P_w)_{w \in S^*}$ of *predicate symbols*. Using these symbols, one may then write axioms in first-order logic with equality. Moreover, one can specify *data types*, each given by a list of data constructors and, optionally, selectors. Data types may be declared to be *generated* or *free*. Generatedness amounts to an implicit higher-order induction axiom and intuitively states that all elements of the data types are reachable by constructor terms ("no junk"); freeness additionally requires that all these constructor terms are distinct ("no confusion"). Basic CASL specifications denote the class of all algebras which fulfil the declared axioms, i.e., CASL has loose semantics. More formally, for CFOL$^{=}$ a *many-sorted Σ-structure* M consists of a non-empty carrier set s^M for each $s \in S$, a total function $f^M \colon w^M \to s^M$ for each function symbol $f \in F_{w,s}$ and a predicate p^M for each predicate symbol $p \in P_w$. A *many-sorted Σ-sentence* is a closed many-sorted first-order formula over Σ or a sort generation constraint.

3 The Hybrid Modal Logic $\mathcal{M}^{\downarrow}_{\mathcal{D}}$ for Event/Data Systems

The logic $\mathcal{M}^{\downarrow}_{\mathcal{D}}$ is a hybrid modal logic for specifying and reasoning about event/data-based reactive systems. The modal part of the logic allows to handle transitions between system configurations where the modalities describe guarded configuration moves based on input and output events with arguments, i.e., messages, and the corresponding effects on data. The hybrid part of the logic allows to bind control states of system configurations and to jump to configurations with such control states explicitly. $\mathcal{M}^{\downarrow}_{\mathcal{D}}$ with its signatures, sentences, and structures forms an institution. Furthermore, $\mathcal{M}^{\downarrow}_{\mathcal{D}}$ can be translated into CASL via a theoroidal institution comorphism.

We extend the logic and the comorphism of [16] by including output. A modal formula $\langle\!| i : \phi \mathbin{/\!/} [O]_N : \psi |\!\rangle \varrho$ now says that in the current configuration an input message according to i can be accepted if precondition state predicate ϕ holds and that, in response, output messages according to $[O]_N$ and satisfying the transition predicate ψ can be produced such that ϱ holds afterwards. The messages frame $[O]_N$ tells that besides outputs from O also additional messages according to N can be sent. This relativisation allows $\mathcal{M}^{\downarrow}_{\mathcal{D}}$

to specify the "cone of messages above O" in a finite and, in particular, institution-compatible way that also is extensible into a theoroidal institution comorphism from $\mathcal{M}^{\downarrow}_{\mathcal{D}}$ to CASL. We furthermore demonstrate that for pure $\mathcal{M}^{\downarrow}_{\mathcal{D}}$-invariants the comorphism leads to simpler CASL proof obligations that are easier to automate in theorem proving.

For the inclusion of *data* in $\mathcal{M}^{\downarrow}_{\mathcal{D}}$, we assume given a consistent, monomorphic CASL specification Dt. The interpretation of the sorts $S(Dt)$ of Dt represents the different kinds of data, like the integers or lists of integers. Requiring Dt to be monomorphic fixes these carrier sets as there is, up to isomorphism, a single model $\mathcal{D}$ of Dt. We also use open formulæ $\mathscr{F}^{\text{CASL}}_{Sig(Dt),X}$ over sorted variables $X = (X_s)_{s\in S(Dt)}$ and their satisfaction relation $\mathcal{D}, \beta \models^{\text{CASL}}_{Sig(Dt),X} \varphi$ for a variable valuation $\beta\colon X \to \mathcal{D}$, i.e., $\beta = (\beta_s\colon X_s \to s^{\mathcal{D}})_{s\in S(Dt)}$.

3.1 Data States and Transitions

A *data signature* A consists of a finite set of *attributes* $|A|$ and a sorting $s(A)\colon |A| \to S(Dt)$. A *data signature morphism* from a data signature A to a data signature A' is a function $\alpha\colon |A| \to |A'|$ such that $s(A)(a) = s(A')(\alpha(a))$ for all $a \in |A|$. We sometimes identify A with the $S(Dt)$-sorted family $(s(A)^{-1}(s))_{s\in S(Dt)}$.

A *data state* ω for a data signature A is given by an attribute valuation $\omega\colon A \to \mathcal{D}$, i.e., $\omega(a) \in s(A)(a)^{\mathcal{D}}$ for $a \in |A|$; in particular, $\Omega(A) = \mathcal{D}^A$ is the set of A-data states. The *state predicates* $\mathscr{F}^{\mathcal{D}}_{A,X}$ are the formulæ in $\mathscr{F}^{\text{CASL}}_{Sig(Dt),A\cup X}$, taking A as well as an additional $S(Dt)$-indexed family X as variables. A state predicate $\phi \in \mathscr{F}^{\mathcal{D}}_{A,X}$ is to be interpreted over an A-data state ω and variable valuation $\beta\colon X \to \mathcal{D}$ and we define the *satisfaction relation* $\models^{\mathcal{D}}$ by

$$\omega, \beta \models^{\mathcal{D}}_{A,X} \phi \iff \mathcal{D}, \omega \cup \beta \models^{\text{CASL}}_{Sig(Dt),A\cup X} \phi\,.$$

The α-*reduct* of an A'-data state $\omega'\colon A' \to \mathcal{D}$ along a data signature morphism $\alpha\colon A \to A'$ is given by the A-data state $\omega'|\alpha\colon A \to \mathcal{D}$ with $(\omega'|\alpha)(a) = \omega'(\alpha(a))$ for every $a \in |A|$. The *state predicate translation* $\mathscr{F}^{\mathcal{D}}_{\alpha,X}\colon \mathscr{F}^{\mathcal{D}}_{A,X} \to \mathscr{F}^{\mathcal{D}}_{A',X}$ along $\alpha\colon A \to A'$ is given by the CASL-formula translation $\mathscr{F}^{\text{CASL}}_{Sig(Dt),\alpha\cup 1_X}$ along the substitution $\alpha \cup 1_X$. Reduct and translation fulfil the following *satisfaction condition* due to the general substitution lemma for CASL:

$$\omega'|\alpha, \beta \models^{\mathcal{D}}_{A,X} \phi \iff \omega', \beta \models^{\mathcal{D}}_{A',X} \mathscr{F}^{\mathcal{D}}_{\alpha,X}(\phi)\,.$$

A *data transition* (ω, ω') for a data signature A is a pair of A-data states; in particular, $\Omega^2(A) = (\mathcal{D}^A)^2$ is the set of A-data transitions. It holds that $(\mathcal{D}^A)^2 \cong \mathcal{D}^{2A}$, where $2A = A \uplus A$ and we assume that no attribute in A ends in a prime $\prime$ and all attributes in the second summand are adorned with an additional prime. The *transition predicates* $\mathscr{F}^{2\mathcal{D}}_{A,X}$ are the formulæ $\mathscr{F}^{\mathcal{D}}_{2A,X}$. The satisfaction relation $\models^{2\mathcal{D}}$ for a transition predicate $\psi \in \mathscr{F}^{2\mathcal{D}}_{A,X}$, data transition $(\omega, \omega') \in \Omega^2(A)$, and valuation $\beta\colon X \to \mathcal{D}$ is defined as

$$(\omega, \omega'), \beta \models^{2\mathcal{D}}_{A,X} \psi \iff \omega + \omega', \beta \models^{\mathcal{D}}_{2A,X} \psi$$

where $\omega + \omega' \in \Omega(2A)$ with $(\omega + \omega')(a) = \omega(a)$ and $(\omega + \omega')(a') = \omega'(a)$.

The α-*reduct* of an A'-data transition (ω', ω'') along a data signature morphism $\alpha\colon A \to A'$ is given by the A-data transition $(\omega', \omega'')|\alpha = (\omega'|\alpha, \omega''|\alpha)$. The *transition predicate translation* $\mathscr{F}^{2\mathcal{D}}_{\alpha,X}$ along α is given by $\mathscr{F}^{\mathcal{D}}_{2\alpha,X}$ with $2\alpha\colon 2A \to 2A'$ defined by $2\alpha(a) = \alpha(a)$ and $2\alpha(a') = \alpha(a)'$. Like for data states, reduct and translation fulfil the following *satisfaction condition*:

$$(\omega', \omega'')|\alpha, \beta \models^{2\mathcal{D}}_{A,X} \psi \iff (\omega', \omega''), \beta \models^{2\mathcal{D}}_{A',X} \mathscr{F}^{2\mathcal{D}}_{\alpha,X}(\psi)\,.$$

3.2 Events and Messages

An *event signature* E consists of a finite set of events $|E|$ and a map $\overline{s}(E)\colon |E| \to S(Dt)^*$ assigning to each $e \in |E|$ its list of parameter sorts. An *event signature morphism* $\eta\colon E \to E'$ is a function $\eta\colon |E| \to |E'|$ such that $\overline{s}(E)(e) = \overline{s}(E')(\eta(e))$ for all $e \in |E|$. We write $e(X)$ for $e \in |E|$ and $\overline{s}(E)(e) = s_1, \ldots, s_n$ when choosing n different *parameters* $X = x_1, \ldots, x_n$, and also $e(X) \in E$ in this case; when $f = e(X)$, we write $X(f)$ for X and we furthermore lift this notation to sets and lists of events. We sometimes identify the parameter list X with the $S(Dt)$-sorted family $(\{x_i \mid s_i = s\})_{s \in S(Dt)}$ and write $\overline{s}(E)(e)(x_i)$ for s_i.

A *message* $e(\beta)$ over an event signature E is given by an event $e(X) \in E$ with its parameters X instantiated by a parameter valuation $\beta\colon X \to \mathcal{D}$ such that $\beta(x) \in s^{\mathcal{D}}$ for $\overline{s}(E)(e)(x) = s$; the set of all messages over an event signature E is denoted by $\hat{E}(E)$. When $\hat{e} = e(\beta) \in \hat{E}(E)$, we write $\beta(\hat{e})$ for β, and when $e(X) \in E$ and $\beta\colon Y \to \mathcal{D}$ for $X \subseteq Y$, we write $e(\beta)$ for $e(\beta\restriction X)$; both notations are furthermore lifted to sets and lists.

The set of *shufflings* $\hat{F}_1 \parallel \hat{F}_2$ of two message lists $\hat{F}_1$ and $\hat{F}_2$ is inductively given by

$$\hat{F} \parallel \varepsilon = \{\hat{F}\} = \varepsilon \parallel \hat{F}\,,$$
$$(\hat{f} :: \hat{F}_1) \parallel \hat{F}_2 = \{\hat{f} :: \hat{F} \mid \hat{F} \in \hat{F}_1 \parallel \hat{F}_2\} = \hat{F}_1 \parallel (\hat{f} :: \hat{F}_2)\,.$$

An event signature morphism $\eta\colon E \to E'$ is lifted to a message $e(\beta) \in \hat{E}(E)$ by setting $\hat{E}(\eta)(e(\beta)) = \eta(e)(\beta) \in \hat{E}(E')$ and also to sets and lists of messages.

3.3 Event/Data Signatures

An *event/data signature* Σ consists of *input* and *output* event signatures $I(\Sigma)$ and $O(\Sigma)$, and a data signature $A(\Sigma)$. An *event/data signature morphism* $\sigma\colon \Sigma \to \Sigma'$ consists of an input event signature morphism $I(\sigma)\colon I(\Sigma) \to I(\Sigma')$, an output event signature morphism $O(\sigma)\colon O(\Sigma) \to O(\Sigma')$, and a data signature morphism $A(\sigma)\colon A(\Sigma) \to A(\Sigma')$. We lift the event signatures and signature morphisms to messages by writing $\hat{I}(\Sigma)$ for $\hat{E}(I(\Sigma))$, $\hat{O}(\Sigma)$ for $\hat{E}(O(\Sigma))$, $\hat{I}(\sigma)$ for $\hat{E}(I(\sigma))$, and $\hat{O}(\sigma)$ for $\hat{E}(O(\sigma))$.

The category of $\mathcal{M}^{\downarrow}_{\mathcal{D}}$-*signatures* $\mathbb{S}^{\mathcal{M}^{\downarrow}_{\mathcal{D}}}$ consists of the event/data signatures and signature morphisms.

3.4 Event/Data Structures

A *configuration* $\gamma = (c, d)$ consists of a *control state* c from some set of control states C and a *data state* d from some set of data states D. Given a data signature A the data state

of γ may be *labelled* by a map ω such that $\omega(d) \in \Omega(A)$. For a set of configurations Γ we write $C(\Gamma)$ for its set of control states.

A Σ*-event/data structure* $M = (\Gamma, R, \Gamma_0, \omega)$ over an event/data signature Σ consists of a set of *configurations* $\Gamma \subseteq C \times D$, a family of *transition relations* $R = (R_{\hat{\imath},\hat{O}} \subseteq \Gamma \times \Gamma)_{\hat{\imath}\in\hat{I}(\Sigma),\hat{O}\in\hat{O}(\Sigma)^*}$, and a non-empty set of *initial configurations* $\Gamma_0 \subseteq \Gamma$ such that Γ is *reachable* from Γ_0 via R, i.e., for all $\gamma \in \Gamma$ there are $\gamma_0 \in \Gamma_0$, $n \geq 0$, $\hat{\imath}_1, \ldots, \hat{\imath}_n \in \hat{I}(\Sigma)$, $\hat{O}_1, \ldots, \hat{O}_n \in \hat{O}(\Sigma)^*$, and $(\gamma_k, \gamma_{k+1}) \in R_{\hat{\imath}_{k+1},\hat{O}_{k+1}}$ for all $0 \leq k < n$ with $\gamma_n = \gamma$; and a *data state labelling* $\omega\colon D \to \Omega(A(\Sigma))$.

We write $c(M)(\gamma) = c$ and $\omega(M)(\gamma) = \omega(d)$ for $\gamma = (c, d) \in \Gamma$, $\Gamma(M)$ for Γ, $C(M)$ for $\{c(M)(\gamma) \mid \gamma \in \Gamma(M)\}$, $R(M)$ for R, $\Gamma_0(M)$ for Γ_0, $C_0(M)$ for $C(\Gamma_0)$, and $\Omega_0(M)$ for $\{\omega(M)(\gamma_0) \mid \gamma_0 \in \Gamma_0\}$.

The above definition restricts structures to reachable ones only. Although an $\mathcal{M}^{\downarrow}_{\mathcal{D}}$-sentence will hold in an event/data structure if it is satisfied in all its initial states, the modal and hybrid operators of $\mathcal{M}^{\downarrow}_{\mathcal{D}}$ will allow for expressing that a certain property holds in all (reachable) states of the structure.

The σ*-reduct* of a Σ'-event/data structure M' along the event/data signature morphism $\sigma\colon \Sigma \to \Sigma'$ is the Σ-event/data structure $M'|\sigma$ such that

- $\Gamma(M'|\sigma) \subseteq \Gamma(M')$ as well as $R(M'|\sigma) = (R(M'|\sigma)_{\hat{\imath},\hat{O}})_{\hat{\imath}\in\hat{I}(\Sigma),\hat{O}\in\hat{O}(\Sigma)^*}$ are inductively defined by $\Gamma_0(M') \subseteq \Gamma(M'|\sigma)$ and, for all $\gamma', \gamma'' \in \Gamma(M')$, $\hat{\imath} \in \hat{I}(\Sigma)$, and $\hat{O} \in \hat{O}(\Sigma)^*$, if $\gamma' \in \Gamma(M'|\sigma)$ and $(\gamma', \gamma'') \in R(M')_{\hat{I}(\sigma)(\hat{\imath}),\hat{O}(\sigma)(\hat{O})}$, then $\gamma'' \in \Gamma(M'|\sigma)$ and $(\gamma', \gamma'') \in R(M'|\sigma)_{\hat{\imath},\hat{O}}$;
- $\Gamma_0(M'|\sigma) = \Gamma_0(M')$; and
- $\omega(M'|\sigma)(\gamma') = (\omega(M')(\gamma'))|A(\sigma)$ for all $\gamma' \in \Gamma(M'|\sigma)$.

This σ-reduct keeps exactly those transitions that are a direct image along σ. It would also be possible to additionally keep transitions that show a super-list of the outputs that can be reached by σ. When moving to $\mathcal{M}^{\downarrow}_{\mathcal{D}}$-sentences, however, it turns out to be impossible to fix a particular list of outputs.

Given sets of input events $J \subseteq I(\Sigma)$ and output events $N \subseteq O(\Sigma)$, we denote by $\Gamma^{J,N}(M, \gamma)$ and $\Gamma^{J,N}(M)$, respectively, the set of configurations of a Σ-event/data structure M that are J, N-reachable from a configuration $\gamma \in \Gamma(M)$ and from an initial configuration $\gamma_0 \in \Gamma_0(M)$, respectively. Here a $\gamma_n \in \Gamma(M)$ is J, N*-reachable in* M *from* a $\gamma_1 \in \Gamma(M)$ if there are $n \geq 1$, $\hat{\imath}_2, \ldots, \hat{\imath}_n \in \hat{I}(J)$, $\hat{O}_2, \ldots, \hat{O}_n \in \hat{O}(N)^*$, and $(\gamma_i, \gamma_{i+1}) \in R(M)_{\hat{\imath}_{k+1},\hat{O}_{k+1}}$ for all $1 \leq k < n$.

The Σ-event/data structures form the discrete category $Str^{\mathcal{M}^{\downarrow}_{\mathcal{D}}}(\Sigma)$ of $\mathcal{M}^{\downarrow}_{\mathcal{D}}$*-structures* over Σ. For each $\sigma\colon \Sigma \to \Sigma'$ in $\mathbb{S}^{\mathcal{M}^{\downarrow}_{\mathcal{D}}}$ the σ*-reduct functor* $Str^{\mathcal{M}^{\downarrow}_{\mathcal{D}}}(\sigma)\colon Str^{\mathcal{M}^{\downarrow}_{\mathcal{D}}}(\Sigma') \to Str^{\mathcal{M}^{\downarrow}_{\mathcal{D}}}(\Sigma)$ is given by $Str^{\mathcal{M}^{\downarrow}_{\mathcal{D}}}(\sigma)(M') = M'|\sigma$.

3.5 Event/Data Formulæ and Sentences

The Σ*-event/data formulæ* $\mathscr{F}^{\mathcal{M}^{\downarrow}_{\mathcal{D}}}_{\Sigma,S}$ over an event/data signature Σ and a set of *state variables* S are inductively defined by

- φ — data state sentence $\varphi \in \mathscr{F}^{\mathcal{D}}_{A(\Sigma),\emptyset}$ holds in the current configuration;

- s — the control state of the current configuration is $s \in S$;
- $\downarrow s \,.\, \varrho$ — calling the current control state s, formula $\varrho \in \mathscr{F}^{\mathcal{M}^{\downarrow}_{\mathcal{D}}}_{\Sigma, S \uplus \{s\}}$ holds (s is turned into a fresh variable by adding to by disjoint union to the set of state variables);
- $(@^{J,N} s)\varrho$ — in all configurations with control state $s \in S$ that are J, N-reachable, formula $\varrho \in \mathscr{F}^{\mathcal{M}^{\downarrow}_{\mathcal{D}}}_{\Sigma, S}$ holds (relativised "jump");
- $\Box^{J,N} \varrho$ — in all configurations that are J, N-reachable from the current configuration formula $\varrho \in \mathscr{F}^{\mathcal{M}^{\downarrow}_{\mathcal{D}}}_{\Sigma, S}$ holds (relativised "globally");
- $\langle i \mathbin{/\!\!/} [O]_N : \psi \rangle \varrho$ — in the current configuration there are valuations $\beta \colon X(i) \to \mathcal{D}$, $\beta' \colon X(O) \to \mathcal{D}$, and a transition for the incoming message $i(\beta) \in \hat{I}(\Sigma)$ and the outgoing messages $\hat{O}' \in O(\beta') \parallel \hat{N}$ for $O(\beta') \in \hat{O}(\Sigma)^*$, $\hat{N} \in \hat{O}(N)^*$ such that $\beta \cup \beta'$ satisfies transition formula $\psi \in \mathscr{F}^{2\mathcal{D}}_{A(\Sigma), X(i) \cup X(O)}$ and $\varrho \in \mathscr{F}^{\mathcal{M}^{\downarrow}_{\mathcal{D}}}_{\Sigma, S}$ holds afterwards;
- $\langle\!\langle i : \phi \mathbin{/\!\!/} [O]_N : \psi \rangle\!\rangle \varrho$ — in the current configuration for all valuations $\beta \colon X(i) \to \mathcal{D}$ satisfying state formula $\phi \in \mathscr{F}^{\mathcal{D}}_{A(\Sigma), X(i)}$ there are a valuation $\beta' \colon X(O) \to \mathcal{D}$ and a transition for the incoming message $i(\beta) \in \hat{I}(\Sigma)$ and the outgoing messages $\hat{O}' \in O(\beta') \parallel \hat{N}$ for $O(\beta') \in \hat{O}(\Sigma)^*$, $\hat{N} \in \hat{O}(N)^*$ such that $\beta \cup \beta'$ satisfies transition formula $\psi \in \mathscr{F}^{2\mathcal{D}}_{A(\Sigma), X(i) \cup X(O)}$ and $\varrho \in \mathscr{F}^{\mathcal{M}^{\downarrow}_{\mathcal{D}}}_{\Sigma, S}$ holds afterwards;
- $\neg \varrho$ — in the current configuration $\varrho \in \mathscr{F}^{\mathcal{M}^{\downarrow}_{\mathcal{D}}}_{\Sigma, S}$ does not hold;
- $\varrho_1 \vee \varrho_2$ — in the current configuration $\varrho_1 \in \mathscr{F}^{\mathcal{M}^{\downarrow}_{\mathcal{D}}}_{\Sigma, S}$ or $\varrho_2 \in \mathscr{F}^{\mathcal{M}^{\downarrow}_{\mathcal{D}}}_{\Sigma, S}$ hold.

We write $(@s)\varrho$ for $(@^{I(\Sigma), O(\Sigma)} s)\varrho$, $\Box \varrho$ for $\Box^{I(\Sigma), O(\Sigma)} \varrho$, $[i \mathbin{/\!\!/} [O]_N : \psi]\varrho$ for $\neg\langle i \mathbin{/\!\!/} [O]_N : \psi \rangle \neg \varrho$, and true for $\downarrow s \,.\, s$; we write O for $[O]_\emptyset$.

Two different kinds of relativisations are used in $\mathcal{M}^{\downarrow}_{\mathcal{D}}$-formulæ: For the jump operator $(@^{J,N} s)\varrho$ and the globally operator $\Box^{J,N} \varrho$ the subsets of input events $J \subseteq I(\Sigma)$ and output events $N \subseteq O(\Sigma)$ restrict the referable states in an $\mathcal{M}^{\downarrow}_{\mathcal{D}}$-structure to those that are J, N-reachable. On the other hand, $[O]_N$ specifies that besides messages from O additional messages for events in $N \subseteq O(\Sigma)$ can be mixed into the output, such that, in particular, $[O]_\emptyset$ requires exactly O. Since the set of output events is assumed to be finite, $[O]_N$ can be used to specify message lists of arbitrary length with finitely many formulæ. Moreover, the syntactic information in both kinds of relativisations is kept through a translation to another $\mathcal{M}^{\downarrow}_{\mathcal{D}}$-signature.

Let $\sigma \colon \Sigma \to \Sigma'$ be an event/data signature morphism. The *event/data formulæ translation* $\mathscr{F}^{\mathcal{M}^{\downarrow}_{\mathcal{D}}}_{\sigma, S} \colon \mathscr{F}^{\mathcal{M}^{\downarrow}_{\mathcal{D}}}_{\Sigma, S} \to \mathscr{F}^{\mathcal{M}^{\downarrow}_{\mathcal{D}}}_{\Sigma', S}$ along σ is recursively given by

- $\mathscr{F}^{\mathcal{M}^{\downarrow}_{\mathcal{D}}}_{\sigma, S}(\varphi) = \mathscr{F}^{\mathcal{D}}_{A(\sigma), \emptyset}(\varphi)$;
- $\mathscr{F}^{\mathcal{M}^{\downarrow}_{\mathcal{D}}}_{\sigma, S}(s) = s$;
- $\mathscr{F}^{\mathcal{M}^{\downarrow}_{\mathcal{D}}}_{\sigma, S}(\downarrow s \,.\, \varrho) = \downarrow s \,.\, \mathscr{F}^{\mathcal{M}^{\downarrow}_{\mathcal{D}}}_{\sigma, S \uplus \{s\}}(\varrho)$;
- $\mathscr{F}^{\mathcal{M}^{\downarrow}_{\mathcal{D}}}_{\sigma, S}((@^{J,N} s)\varrho) = (@^{I(\sigma)(J), O(\sigma)(N)} s)\mathscr{F}^{\mathcal{M}^{\downarrow}_{\mathcal{D}}}_{\sigma, S}(\varrho)$;
- $\mathscr{F}^{\mathcal{M}^{\downarrow}_{\mathcal{D}}}_{\sigma, S}(\Box^{J,N} \varrho) = \Box^{I(\sigma)(J), O(\sigma)(N)} \mathscr{F}^{\mathcal{M}^{\downarrow}_{\mathcal{D}}}_{\sigma, S}(\varrho)$;
- $\mathscr{F}^{\mathcal{M}^{\downarrow}_{\mathcal{D}}}_{\sigma, S}(\langle i \mathbin{/\!\!/} [O]_N : \psi \rangle \varrho) =$
  $\langle I(\sigma)(i) \mathbin{/\!\!/} [O(\sigma)(O)]_{O(\sigma)(N)} : \mathscr{F}^{2\mathcal{D}}_{A(\sigma), X(i) \cup X(O)}(\psi) \rangle \mathscr{F}^{\mathcal{M}^{\downarrow}_{\mathcal{D}}}_{\sigma, S}(\varrho)$;

- $\mathscr{F}^{\mathcal{M}^{\downarrow}_{\mathcal{D}}}_{\sigma,S}(\langle\!|i : \phi /\!\!/ [O]_N : \psi|\!\rangle \varrho) =$
  $\langle\!|I(\sigma)(i) : \mathscr{F}^{\mathcal{D}}_{A(\sigma),X(i)}(\phi) /\!\!/ [O(\sigma)(O)]_{O(\sigma)(N)} : \mathscr{F}^{2\mathcal{D}}_{A(\sigma),X(i)\cup X(O)}(\psi)|\!\rangle \mathscr{F}^{\mathcal{M}^{\downarrow}_{\mathcal{D}}}_{\sigma,S}(\varrho)$;
- $\mathscr{F}^{\mathcal{M}^{\downarrow}_{\mathcal{D}}}_{\sigma,S}(\neg\varrho) = \neg\mathscr{F}^{\mathcal{M}^{\downarrow}_{\mathcal{D}}}_{\sigma,S}(\varrho)$;
- $\mathscr{F}^{\mathcal{M}^{\downarrow}_{\mathcal{D}}}_{\sigma,S}(\varrho_1 \vee \varrho_2) = \mathscr{F}^{\mathcal{M}^{\downarrow}_{\mathcal{D}}}_{\sigma,S}(\varrho_1) \vee \mathscr{F}^{\mathcal{M}^{\downarrow}_{\mathcal{D}}}_{\sigma,S}(\varrho_2)$.

The set $\mathrm{Sen}^{\mathcal{M}^{\downarrow}_{\mathcal{D}}}(\Sigma)$ of *Σ-event/data sentences* is given by $\mathscr{F}^{\mathcal{M}^{\downarrow}_{\mathcal{D}}}_{\Sigma,\emptyset}$, the *event/data sentence translation* $\mathrm{Sen}^{\mathcal{M}^{\downarrow}_{\mathcal{D}}}(\sigma)\colon \mathrm{Sen}^{\mathcal{M}^{\downarrow}_{\mathcal{D}}}(\Sigma) \to \mathrm{Sen}^{\mathcal{M}^{\downarrow}_{\mathcal{D}}}(\Sigma')$ by $\mathscr{F}^{\mathcal{M}^{\downarrow}_{\mathcal{D}}}_{\sigma,\emptyset}$.

3.6 Satisfaction Relation for $\mathcal{M}^{\downarrow}_{\mathcal{D}}$

Let Σ be an event/data signature, M a Σ-event/data structure, S a set of state variables, $v\colon S \to C(M)$ a state variable assignment, and $\gamma \in \Gamma(M)$. The *satisfaction relation* for event/data formulæ is inductively given by

- $M, v, \gamma \models^{\mathcal{M}^{\downarrow}_{\mathcal{D}}}_{\Sigma,S} \varphi$ iff $\omega(M)(\gamma), \emptyset \models^{\mathcal{D}}_{A(\Sigma),\emptyset} \varphi$;
- $M, v, \gamma \models^{\mathcal{M}^{\downarrow}_{\mathcal{D}}}_{\Sigma,S} s$ iff $v(s) = c(M)(\gamma)$;
- $M, v, \gamma \models^{\mathcal{M}^{\downarrow}_{\mathcal{D}}}_{\Sigma,S} \downarrow s\,.\,\varrho$ iff $M, v\{s \mapsto c(M)(\gamma)\}, \gamma \models^{\mathcal{M}^{\downarrow}_{\mathcal{D}}}_{\Sigma,S\uplus\{s\}} \varrho$;
- $M, v, \gamma \models^{\mathcal{M}^{\downarrow}_{\mathcal{D}}}_{\Sigma,S} (@^{J,N}s)\varrho$ iff $M, v, \gamma' \models^{\mathcal{M}^{\downarrow}_{\mathcal{D}}}_{\Sigma,S} \varrho$ for all $\gamma' \in \Gamma^{J,N}(M)$ with $c(M)(\gamma') = v(s)$;
- $M, v, \gamma \models^{\mathcal{M}^{\downarrow}_{\mathcal{D}}}_{\Sigma,S} \Box^{J,N}\varrho$ iff $M, v, \gamma' \models^{\mathcal{M}^{\downarrow}_{\mathcal{D}}}_{\Sigma,S} \varrho$ for all $\gamma' \in \Gamma^{J,N}(M, \gamma)$;
- $M, v, \gamma \models^{\mathcal{M}^{\downarrow}_{\mathcal{D}}}_{\Sigma,S} \langle i /\!\!/ [O]_N : \psi\rangle\varrho$ iff there are valuations $\beta\colon X(i) \to \mathcal{D}, \beta'\colon X(O) \to \mathcal{D}$, output messages $\hat{O}' \in O(\beta') \parallel \hat{N}$ with $\hat{N} \in \hat{O}(N)^*$, and a configuration $\gamma' \in \Gamma(M)$ such that $(\gamma, \gamma') \in R(M)_{i(\beta),\hat{O}'}$, $(\omega(M)(\gamma), \omega(M)(\gamma')), \beta \cup \beta' \models^{2\mathcal{D}}_{A(\Sigma),X(i)\cup X(O)} \psi$, and $M, v, \gamma' \models^{\mathcal{M}^{\downarrow}_{\mathcal{D}}}_{\Sigma,S} \varrho$;
- $M, v, \gamma \models^{\mathcal{M}^{\downarrow}_{\mathcal{D}}}_{\Sigma,S} \langle\!|i : \phi /\!\!/ [O]_N : \psi|\!\rangle\varrho$ iff for all valuations $\beta\colon X(i) \to \mathcal{D}$ that satisfy $\omega(M)(\gamma), \beta \models^{\mathcal{D}}_{A(\Sigma),X(i)} \phi$ there are a valuation $\beta'\colon X(O) \to \mathcal{D}$, output messages $\hat{O}' \in O(\beta') \parallel \hat{N}$ with $\hat{N} \in \hat{O}(N)^*$, and a configuration $\gamma' \in \Gamma(M)$ such that $(\gamma, \gamma') \in R(M)_{i(\beta),\hat{O}'}$, $(\omega(M)(\gamma), \omega(M)(\gamma')), \beta \cup \beta' \models^{2\mathcal{D}}_{A(\Sigma),X(i)\cup X(O)} \psi$, and $M, v, \gamma' \models^{\mathcal{M}^{\downarrow}_{\mathcal{D}}}_{\Sigma,S} \varrho$;
- $M, v, \gamma \models^{\mathcal{M}^{\downarrow}_{\mathcal{D}}}_{\Sigma,S} \neg\varrho$ iff $M, v, \gamma \not\models^{\mathcal{M}^{\downarrow}_{\mathcal{D}}}_{\Sigma,S} \varrho$;
- $M, v, \gamma \models^{\mathcal{M}^{\downarrow}_{\mathcal{D}}}_{\Sigma,S} \varrho_1 \vee \varrho_2$ iff $M, v, \gamma \models^{\mathcal{M}^{\downarrow}_{\mathcal{D}}}_{\Sigma,S} \varrho_1$ or $M, v, \gamma \models^{\mathcal{M}^{\downarrow}_{\mathcal{D}}}_{\Sigma,S} \varrho_2$.

For a $\Sigma \in |\mathbb{S}^{\mathcal{M}^{\downarrow}_{\mathcal{D}}}|$, an $M \in |Str^{\mathcal{M}^{\downarrow}_{\mathcal{D}}}(\Sigma)|$, and a $\rho \in \mathrm{Sen}^{\mathcal{M}^{\downarrow}_{\mathcal{D}}}(\Sigma)$ the *satisfaction relation* $M \models^{\mathcal{M}^{\downarrow}_{\mathcal{D}}}_{\Sigma} \rho$ holds if, and only if, $M, \emptyset, \gamma_0 \models^{\mathcal{M}^{\downarrow}_{\mathcal{D}}}_{\Sigma,\emptyset} \rho$ for all $\gamma_0 \in \Gamma_0(M)$.

Theorem 1. $(\mathbb{S}^{\mathcal{M}^{\downarrow}_{\mathcal{D}}}, Str^{\mathcal{M}^{\downarrow}_{\mathcal{D}}}, \mathrm{Sen}^{\mathcal{M}^{\downarrow}_{\mathcal{D}}}, \models^{\mathcal{M}^{\downarrow}_{\mathcal{D}}})$ *is an institution.*

from *Basic/StructuredDatatypes* **get** LIST, SET % import finite lists and sets
spec $\text{TRANS}_\Sigma = Dt$
then free type $\text{InEvt} ::= I(\Sigma)$
free type $\text{OutEvt} ::= O(\Sigma)$
then LIST[sort OutEvt] **and** SET[sort InEvt] **and** SET[sort OutEvt]
then sort Ctrl
free type $\text{Conf} ::= \text{conf}(\text{c} : \text{Ctrl}; A(\Sigma))$
preds $\text{init} : \text{Conf}$;
$\text{trans} : \text{Conf} \times \text{InEvt} \times \mathit{List}[\text{OutEvt}] \times \text{Conf}$
$\cdot \exists g : \text{Conf} \cdot \text{init}(g)$ % there is some initial configuration
then free { **pred** $\text{reachable} : \mathit{Set}[\text{InEvt}] \times \mathit{Set}[\text{OutEvt}] \times \text{Conf} \times \text{Conf}$
$\forall g, g', g'' : \text{Conf}; J : \mathit{Set}[\text{InEvt}]; N : \mathit{Set}[\text{OutEvt}]; i : \text{InEvt}; O : \mathit{List}[\text{OutEvt}]$
$\cdot \text{reachable}(J, N, g, g)$
$\cdot \text{reachable}(J, N, g, g') \wedge i \in J \wedge O \subseteq N \wedge \text{trans}(g', i, O, g'') \Rightarrow$
$\text{reachable}(J, N, g, g'')$ }
then preds $\text{reachable}(J : \mathit{Set}[\text{InEvt}], N : \mathit{Set}[\text{OutEvt}], g : \text{Conf}) \Leftrightarrow$
$\exists g_0 : \text{Conf} \cdot \text{init}(g_0) \wedge \text{reachable}(J, N, g_0, g)$;
$\text{reachable}(g : \text{Conf}) \Leftrightarrow \text{reachable}(I(\Sigma), O(\Sigma), g)$
then pred $\text{mixed} : \mathit{List}[\text{OutEvt}] \times \mathit{Set}[\text{OutEvt}] \times \mathit{List}[\text{OutEvt}]$
$\forall o, o' : \text{OutEvt}; O, O' : \mathit{List}[\text{OutEvt}]; N : \mathit{Set}[\text{OutEvt}]$
$\cdot \text{mixed}(O, N, O)$
$\cdot \text{mixed}(o :: O, N, o :: O')$ **if** $\text{mixed}(O, N, O')$
$\cdot \text{mixed}(O, N, o' :: O')$ **if** $\text{mixed}(O, N, O') \wedge o' \in N$
end

Figure 2. Frame for translating $\mathcal{M}^{\downarrow}_{\mathcal{D}}$ into CASL.

3.7 A Theoroidal Comorphism from $\mathcal{M}^{\downarrow}_{\mathcal{D}}$ to CASL

We define a theoroidal comorphism from $\mathcal{M}^{\downarrow}_{\mathcal{D}}$ to CASL. The construction mainly follows the standard translation of modal logics to first-order logic [1] and extends the scheme of [16] by outputs.

The basis is a representation of $\mathcal{M}^{\downarrow}_{\mathcal{D}}$-signatures and the frame given by $\mathcal{M}^{\downarrow}_{\mathcal{D}}$-structures as a CASL-specification as shown in Fig. 2. The signature translation

$$\nu^{Sig} : \mathbb{S}^{\mathcal{M}^{\downarrow}_{\mathcal{D}}} \to \text{Pres}^{\text{CASL}}$$

maps an $\mathcal{M}^{\downarrow}_{\mathcal{D}}$-signature Σ to the CASL-theory presentation given by TRANS_Σ and an $\mathcal{M}^{\downarrow}_{\mathcal{D}}$-signature morphism to the corresponding theory presentation morphism. TRANS_Σ first of all covers the events according to $I(\Sigma)$ and $O(\Sigma)$ with types InEvt and OutEvt, and the configurations with type Conf showing a single constructor conf for the control state from Ctrl and a data state given by assignments to the attributes from $A(\Sigma)$. Furthermore, TRANS_Σ sets the frame for describing reachable transition systems with a set of initial configurations, a transition relation, and reachability predicates, where the specification of reachable uses CASL's "structured free" construct to ensure reachability to be inductively defined. Finally, a predicate mixed is included for representing the shufflings of a list of outputs with some additional output events.

The model translation

$$\nu^{\mathrm{Mod}}_{\Sigma} \colon \mathrm{Mod}^{\textsc{Casl}}(\nu^{Sig}(\Sigma)) \to Str^{\mathcal{M}^{\downarrow}_{\mathcal{D}}}(\Sigma)$$

then can rely on this encoding. In particular, for a model $M' \in \mathrm{Mod}^{\textsc{Casl}}(\nu^{Sig}(\Sigma))$, there are bijective maps $\iota_{M',\mathrm{Conf}} \colon \mathrm{Conf}^{M'} \cong \mathrm{Ctrl}^{M'} \times \Omega(A(\Sigma))$ for the configurations as well as $\iota_{M',\mathrm{InEvt}} \colon \mathrm{InEvt}^{M'} \cong \hat{I}(\Sigma)$ and $\iota_{M',\mathrm{OutEvt}} \colon \mathrm{OutEvt}^{M'} \cong \hat{O}(\Sigma)$ for the messages. Moreover, $\mathrm{mixed}^{M'}(\iota^{-1}_{M',\mathrm{OutEvt}}(\hat{O}), \iota^{-1}_{M',\mathrm{OutEvt}}(N), \iota^{-1}_{M',\mathrm{OutEvt}}(\hat{O}'))$ if, and only if, $\hat{O}' \in \hat{O} \parallel \hat{N}'$ with $\hat{N}' \in N^*$. The $\mathcal{M}^{\downarrow}_{\mathcal{D}}$-structure resulting from a CASL-model M' of $\textsc{Trans}_{\Sigma}$ can thus be defined by

- $\Gamma(\nu^{\mathrm{Mod}}_{\Sigma}(M')) = \iota^{-1}_{M',\mathrm{Conf}}(\{g^{M'} \in \mathrm{Conf}^{M'} \mid \mathrm{reachable}^{M'}(g^{M'})\})$
- $R(\nu^{\mathrm{Mod}}_{\Sigma}(M'))_{\hat{\imath},\hat{O}} = \{(\gamma,\gamma') \in \Gamma(\nu^{\mathrm{Mod}}_{\Sigma}(M')) \times \Gamma(\nu^{\mathrm{Mod}}_{\Sigma}(M')) \mid \mathrm{trans}^{M'}(\iota_{M',\mathrm{Conf}}(\gamma), \iota^{-1}_{M',\mathrm{InEvt}}(\hat{\imath}), \iota^{-1}_{M',\mathrm{OutEvt}}(\hat{O}), \iota_{M',\mathrm{Conf}}(\gamma'))\}$
- $\Gamma_0(\nu^{\mathrm{Mod}}_{\Sigma}(M')) = \{\gamma \in \Gamma(\nu^{\mathrm{Mod}}_{\Sigma}(M')) \mid \mathrm{init}^{M'}(\iota_{M',\mathrm{Conf}}(\gamma))\})$
- $\omega(\nu^{\mathrm{Mod}}_{\Sigma}(M')) = \{(c,\omega) \in \Gamma(\nu^{\mathrm{Mod}}_{\Sigma}(M')) \mapsto \omega\}$

For $\mathcal{M}^{\downarrow}_{\mathcal{D}}$-sentences, we first define a formula translation

$$\nu^{\mathscr{F}}_{\Sigma,S,g} \colon \mathscr{F}^{\mathcal{M}^{\downarrow}_{\mathcal{D}}}_{\Sigma,S} \to \mathscr{F}^{\textsc{Casl}}_{\nu^{\mathbb{S}}(\Sigma),S\cup\{g\}}$$

which, mimicking the standard translation, takes a variable $g : \mathrm{Conf}$ as a parameter that records the "current configuration" and also uses a set S of state names for the control states. The translation embeds the data state and 2-data state formulæ using the substitution $A(\Sigma)(g) = \{a \mapsto a(g) \mid a \in A(\Sigma)\}$ for replacing the attributes $a \in A(\Sigma)$ by the accessors $a(g)$. The translation of $\mathcal{M}^{\downarrow}_{\mathcal{D}}$-formulæ then reads

- $\nu^{\mathscr{F}}_{\Sigma,S,g}(\varphi) = \mathscr{F}^{\textsc{Casl}}_{\nu^{\mathbb{S}}(\Sigma),A(\Sigma)(g)}(\varphi)$
- $\nu^{\mathscr{F}}_{\Sigma,S,g}(s) = (s = \mathrm{c}(g))$
- $\nu^{\mathscr{F}}_{\Sigma,S,g}(\downarrow s \,.\, \varrho) = \exists s : \mathrm{Ctrl} \,.\, s = \mathrm{c}(g) \wedge \nu^{\mathscr{F}}_{\Sigma,S\uplus\{s\},g}(\varrho)$
- $\nu^{\mathscr{F}}_{\Sigma,S,g}((@^{J,N}s)\varrho) = \forall g' : \mathrm{Conf} \,.\, (\mathrm{c}(g') = s \wedge \mathrm{reachable}(J,N,g')) \Rightarrow \nu^{\mathscr{F}}_{\Sigma,S,g'}(\varrho)$
- $\nu^{\mathscr{F}}_{\Sigma,S,g}(\Box^{J,N}\varrho) = \forall g' : \mathrm{Conf} \,.\, \mathrm{reachable}(J,N,g,g') \Rightarrow \nu^{\mathscr{F}}_{\Sigma,S,g'}(\varrho)$
- $\nu^{\mathscr{F}}_{\Sigma,S,g}(\langle i \mathbin{/\!\!/} [O]_N : \psi\rangle\varrho) = \exists X : \overline{s}(I(\Sigma))(i); X' : \overline{s}(O(\Sigma))(O); O' : \mathit{List}[\mathrm{OutEvt}]; g' : \mathrm{Conf} \,.\, \mathrm{mixed}(O(X'),N,O') \wedge \mathrm{trans}(g,i(X),O',g') \wedge \mathscr{F}^{\textsc{Casl}}_{\nu^{\mathbb{S}}(\Sigma),A(\Sigma)(g)\cup A(\Sigma)(g')\cup 1_{X\cup X'}}(\psi) \wedge \nu^{\mathscr{F}}_{\Sigma,S,g'}(\varrho)$
- $\nu^{\mathscr{F}}_{\Sigma,S,g}(\langle\!|i : \phi \mathbin{/\!\!/} [O]_N : \psi|\!\rangle\varrho) = \forall X : \overline{s}(I(\Sigma))(i) \,.\, \mathscr{F}^{\textsc{Casl}}_{\nu^{\mathbb{S}}(\Sigma),A(\Sigma)(g)\cup 1_X}(\phi) \Rightarrow \exists X' : \overline{s}(O(\Sigma))(O); O' : \mathit{List}[\mathrm{OutEvt}]; g' : \mathrm{Conf} \,.\, \mathrm{mixed}(O(X'),N,O') \wedge \mathrm{trans}(g,i(X),O',g') \wedge \mathscr{F}^{\textsc{Casl}}_{\nu^{\mathbb{S}}(\Sigma),A(\Sigma)(g)\cup A(\Sigma)(g')\cup 1_{X\cup X'}}(\psi) \wedge \nu^{\mathscr{F}}_{\Sigma,S,g'}(\varrho)$
- $\nu^{\mathscr{F}}_{\Sigma,S,g}(\neg\varrho) = \neg\nu^{\mathscr{F}}_{\Sigma,S,g}(\varrho)$
- $\nu^{\mathscr{F}}_{\Sigma,S,g}(\varrho_1 \vee \varrho_2) = \nu^{\mathscr{F}}_{\Sigma,S,g}(\varrho_1) \vee \nu^{\mathscr{F}}_{\Sigma,S,g}(\varrho_2)$

Building on the translation of formulæ, the sentence translation

$$\nu^{\mathrm{Sen}}_{\Sigma} \colon \mathrm{Sen}^{\mathcal{M}^{\downarrow}_{\mathcal{D}}}(\Sigma) \to \mathrm{Sen}^{\textsc{Casl}}(\nu^{\mathbb{S}}(\Sigma))$$

only has to require additionally that evaluation starts in an initial state:

– $\nu^{\mathrm{Sen}}_{\Sigma}(\rho) = \forall g : \mathrm{Conf} \,.\, \mathrm{init}(g) \Rightarrow \nu^{\mathscr{F}}_{\Sigma,\emptyset,g}(\rho)$

Theorem 2. *$(\nu^{Sig}, \nu^{\mathrm{Mod}}, \nu^{\mathrm{Sen}})$ is a theoroidal comorphism from $\mathcal{M}^{\downarrow}_{\mathcal{D}}$ to* CASL.

For a CASL-proof of an *$\mathcal{M}^{\downarrow}_{\mathcal{D}}$-invariant* $\Box\varphi$ such that φ has to hold in every reachable configuration, the full generality of the reachable predicate can sometimes be avoided by replacing the proof obligation $\forall g : \mathrm{Conf} \,.\, \mathrm{reachable}(g) \Rightarrow \mathscr{F}^{\mathrm{CASL}}_{\nu^{\mathbb{S}}(\Sigma),A(\Sigma)(g)}(\varphi)$ by the usual stepwise induction scheme that only requires to demonstrate the invariant to hold in all initial configurations and that it is preserved by every transition. Moreover, the $\mathcal{M}^{\downarrow}_{\mathcal{D}}$-state formula φ can be generalised into a CASL-invariant.

Proposition 1. *Let (Σ, P) be a theory presentation in $\mathcal{M}^{\downarrow}_{\mathcal{D}}$ and $(\nu^{\mathbb{S}}(\Sigma), \Phi)$ a theory presentation in* CASL *such that $\mathrm{Mod}^{\mathrm{CASL}}(\nu^{\mathrm{Pres}}(\Sigma, P)) \subseteq \mathrm{Mod}^{\mathrm{CASL}}(\nu^{\mathbb{S}}(\Sigma), \Phi)$. Let $inv^{\mathrm{CASL}}(g) \in \mathscr{F}^{\mathrm{CASL}}_{\nu^{\mathbb{S}}(\Sigma),\{g\}}$ be a* CASL*-formula with a single free variable g and $inv^{\mathcal{M}^{\downarrow}_{\mathcal{D}}} \in \mathscr{F}^{\mathcal{D}}_{A(\Sigma),\emptyset}$ an $\mathcal{M}^{\downarrow}_{\mathcal{D}}$-state formula, such that*

(I0) $\forall g : \mathrm{Conf} \,.\, inv^{\mathrm{CASL}}(g) \Rightarrow \mathscr{F}^{\mathrm{CASL}}_{\nu^{\mathbb{S}}(\Sigma),A(\Sigma)(g)}(inv^{\mathcal{M}^{\downarrow}_{\mathcal{D}}})$

(I1) $\forall g : \mathrm{Conf} \,.\, \mathrm{init}(g) \Rightarrow inv^{\mathrm{CASL}}(g)$

(I2) $\forall g, g' : \mathrm{Conf}; i \in \mathrm{InEvt}; O \in \mathit{List}[\mathrm{OutEvt}] \,.$
$inv^{\mathrm{CASL}}(g) \wedge \mathrm{trans}(g, i, O, g') \Rightarrow inv^{\mathrm{CASL}}(g')$

hold in every model $M' \in \mathrm{Mod}^{\mathrm{CASL}}(\nu^{\mathbb{S}}(\Sigma), \Phi)$. Then $\nu^{\mathrm{Mod}}_{\Sigma}(M') \models^{\mathcal{M}^{\downarrow}_{\mathcal{D}}}_{\Sigma} \Box inv^{\mathcal{M}^{\downarrow}_{\mathcal{D}}}$ for all models $M' \in \mathrm{Mod}^{\mathrm{CASL}}(\nu^{\mathrm{Pres}}(\Sigma, P))$.

4 Simple UML State Machines with Outputs

UML state machines [13, Ch. 14] provide means to specify the reactive behaviour of objects or component instances. These entities hold an internal data state, typically given by a set of attributes or properties as specified in a static structure, and shall react to event occurrences like incoming messages by firing different transitions in different control states. Such transitions may have a guard depending on event arguments and the internal state and may change, as an effect, the internal control and data state of the entity as well as send out messages on their own. Beyond such "simple" means for specifying reactive entities, UML state machines offer also more advanced modelling constructs, like hierarchical states or compound transitions, which, however, we defer to future work.

In our formal account, extending again [16], a *simple UML state machine with outputs* U uses an event/data signature $\Sigma(U)$ for its input and output events as well as its attributes and consists of a finite set of *control states* $C(U)$; a finite set of *transition specifications* $T(U)$ of the form $(c, \phi, i(X), o_1(X_1), \ldots, o_m(X_m), \psi, c')$ with

– *source* and *target* control states $c, c' \in C(U)$,
– *input* event $i(X) \in I(\Sigma(U))$ and *output* events $o_1(X_1), \ldots, o_m(X_m) \in O(\Sigma(U))$ such that $X \cap \bigcup_{1 \leq k \leq m} X_m = \emptyset$,
– *precondition* state predicate $\phi \in \mathscr{F}^{\mathcal{D}}_{A(\Sigma(U)),X}$, and

– *postcondition* transition predicate $\psi \in \mathscr{F}^{2\mathcal{D}}_{A(\Sigma(U)),X\cup\bigcup_{1\leq k\leq m} X_k}$;

an *initial control state* $c_0(U) \in C(U)$; and an *initial state predicate* $\varphi_0(U) \in \mathscr{F}^{\mathcal{D}}_{A(\Sigma(U)),\emptyset}$, such that $C(U)$ is *syntactically reachable*, i.e., for every $c \in C(U) \setminus \{c_0(U)\}$ there are $(c_0(U), \phi_1, i_1, O_1, \psi_1, c_1), \ldots, (c_{n-1}, i_n, O_n, \psi_n, c_n) \in T(U)$ with $n > 0$ and $c_n = c$. The constraint of syntactic reachability is only introduced to simplify semantic and algorithmic constructions on simple UML state machines with output.

A $\Sigma(U)$-event/data structure M is a *model* of a simple UML state machine U with output, $M \in \mathrm{Mod}^{\mathcal{M}^{\downarrow}_{\mathcal{D}}}(U)$, if $C(U) \subseteq C(M)$ up to a bijective renaming, $C_0(M) = \{c_0(U)\}$, $\Omega_0(M) \subseteq \{\omega \in |\Omega(A(\Sigma(U)))| \mid \omega \models^{\mathcal{D}}_{A(\Sigma(U)),\emptyset} \varphi_0(U)\}$, and if the following holds for all $(c, d) \in \Gamma(M)$:

– for all transition specifications $(c, \phi, i, O, \psi, c') \in T(U)$ and $\beta\colon X(i) \to \mathcal{D}$ with $\omega(M)(d), \beta \models^{\mathcal{D}}_{A(\Sigma(U)),X(i)} \phi$, there is a $\beta'\colon X(O) \to \mathcal{D}$ and a pair $((c, d), (c', d')) \in R(M)_{i(\beta),O(\beta')}$ such that $(\omega(M)(d), \omega(M)(d')), \beta \cup \beta' \models^{2\mathcal{D}}_{A(\Sigma(U)),X(i)\cup X(O)} \psi$;
– for all pairs $((c, d), (c', d')) \in R(M)_{i(\beta),O(\beta')}$ there is some transition specification $(c, \phi, i, O, \psi, c') \in T(U)$ such that $\omega(M)(d), \beta \models^{\mathcal{D}}_{A(\Sigma(U)),X(i)} \phi$ and also $(\omega(M)(d), \omega(M)(d')), \beta \cup \beta' \models^{2\mathcal{D}}_{A(\Sigma(U)),X(\{i\}\cup O)} \psi$.

The last requirement that all transitions in a model are due to transition specifications does not cover the requirement of *input enabledness* for UML state machines: An event for which currently no transition can fire is discarded. This behaviour can be added by a syntactical transformation extending the set of transition specifications by self-loops with empty outputs for all situations where some event is not accepted.

In UML, completion events are produced whenever a state completes its internal behaviour and such events have always to be prioritised in event processing; the reaction to a completion event is indicated by a transition without a triggering event. For the simple machines with output described here, where states do not show internal behaviour, the only use of completion events is to let a machine make progress autonomously without external input. For using this feature, the machine's event/data signature has to be extended by such events and the transition specifications have to take completions into account. Still, the prioritisation cannot be covered by a single state machine alone, as it has no event processing discipline of its own.

Extending the characterisation algorithm in [16] with outputs, it can be shown that $\mathcal{M}^{\downarrow}_{\mathcal{D}}$ is expressive enough to capture the model class of a simple UML state machine with output U by a single sentence ϱ_U such that $M \in \mathrm{Mod}^{\mathcal{M}^{\downarrow}_{\mathcal{D}}}(U)$ if, and only if, $M \models^{\mathcal{M}^{\downarrow}_{\mathcal{D}}}_{\Sigma(U)} \varrho_U$. The simplest case is a single transition specification $(c, \phi, i, O, \psi, c')$: By requiring $(@c)\langle\!|i : \phi /\!/ O : \psi|\!\rangle c'$ it can be ensured that a model indeed shows a transition from control state c to the control state c' for the input event i with precondition ϕ satisfied which outputs O with ψ satisfied. For requiring that such a transition for input i and output O is only offered when the precondition ϕ and the transition condition ψ hold, a formula $(@c)[i /\!/ O : \neg\phi \vee \neg\psi]\mathrm{false}$ has to be added. For ensuring that no other output than O can be produced, on the one hand $(@c)[i /\!/ O' : \mathrm{true}]\mathrm{false}$ for every $O' \neq O$ that is at most the length of O has to be added and on the other hand $(@c)[i /\!/ [O']_{O(\Sigma)} : \mathrm{true}]\mathrm{false}$ for every O' with length one more than O.

Reasoning over a simple UML state machine with output U in CASL via the translation of U's characterising sentence along the theoroidal comorphism of Thm. 2 will involve some not fully transpicuous axioms due to the necessary exclusion of some behaviour using formulæ like $(@c)[i /\!\!/ [O']_{O(\Sigma)} : \mathrm{true}]\mathrm{false}$. It is therefore sometimes advantageous to directly use the requirements for M being a model of U to obtain another characterisation of the trans predicate in the CASL presentation for the comorphism, which then can be favourably combined with Prop. 1 for proving invariants:

Proposition 2. *Let U be a simple UML state machine with output and let $M' \in \mathrm{Mod}^{\mathrm{CASL}}(\nu^{Sig}(\Sigma(U)))$ such that $\nu^{\mathrm{Mod}}_{\Sigma(U)}(M') \in \mathrm{Mod}^{\mathcal{M}^{\downarrow}_{\mathcal{D}}}(U)$. Then*

$$
\begin{aligned}
M' \models^{\mathrm{CASL}}_{\nu^{\mathbb{S}}(\Sigma)} \forall g : \mathrm{Conf} \,.\, \mathrm{reachable}(g) \Rightarrow \\
\Big(\forall g' : \mathrm{Conf}; i_* : \mathrm{InEvt}; O_* : \mathit{List}[\mathrm{OutEvt}] \,.\, \mathrm{trans}(g, i_*, O_*, g') \iff \\
\bigvee_{(c,\phi,i,O,\psi,c') \in T(U)} \exists X : \overline{s}(I(\Sigma))(i); X' : \overline{s}(O(\Sigma))(O) \,. \\
\mathrm{c}(g) = c \wedge \mathscr{F}^{\mathrm{CASL}}_{\nu^{\mathbb{S}}(\Sigma), A(\Sigma)(g) \cup 1_X}(\phi) \wedge i_* = i(X) \wedge O_* = O(X') \wedge \\
\mathscr{F}^{\mathrm{CASL}}_{\nu^{\mathbb{S}}(\Sigma), A(\Sigma)(g) \cup A(\Sigma)(g') \cup 1_{X \cup X'}}(\psi) \wedge \mathrm{c}(g') = c' \Big) \,.
\end{aligned}
$$

5 Simple UML Composite Structures

A UML composite structure [13, Ch. 11] specifies the internal structure of a class or component and its collaborations. For our purposes, a composite structure is given by class or component instances, its so-called *parts*, that can communicate through their attached *ports* specifying provided and required interfaces and being linked by *connectors*. All connectors are assumed to be binary and each part to be equipped with a state machine for describing its behaviour.

A *composite structure signature* Δ over $\mathcal{M}^{\downarrow}_{\mathcal{D}}$ consists of a set $\mathit{Cmp}(\Delta)$ of *parts* c each equipped with an $\mathcal{M}^{\downarrow}_{\mathcal{D}}$-signature $\Sigma(\Delta, c)$ for its input and output events and internal attributes; a set $\mathit{Prt}(\Delta)$ of *ports* p each showing a part $\mathit{cmp}(\Delta)(p) \in \mathit{Cmp}(\Delta)$ as well as an $\mathcal{M}^{\downarrow}_{\mathcal{D}}$-signature $\Sigma(\Delta, p)$ without attributes (i.e., $A(\Sigma(\Delta, p)) = \emptyset$) for its *provided* (input) and *required* (output) events; and a symmetric binary relation $\mathit{Con}(\Delta) \subseteq \mathit{Prt}(\Delta) \times \mathit{Prt}(\Delta)$ of *connectors* such that

- for each part $c \in \mathit{Cmp}(\Delta)$, the input and output events of $\Sigma(\Delta, c)$ are the provided and required events of c's ports prefixed with the port name, i.e., for $F \in \{I, O\}$, $F(\Sigma(\Delta, c)) = \bigcup_{p \in \mathit{cmp}(\Delta)^{-1}(c)} \{p.f \mid f \in F(\Sigma(\Delta, p))\}$;
- for each part $c \in \mathit{Cmp}(\Delta)$, the attributes of $\Sigma(\Delta, c)$ are all prefixed with c, i.e., if $a \in A(\Sigma(\Delta, c))$, then $a = c.a_*$;
- for each connection $(p, p') \in \mathit{Con}(\Delta)$, the required events of port p are provided by p', i.e., $O(\Sigma(\Delta, p)) \subseteq I(\Sigma(\Delta, p'))$.

We say that port $p \in \mathit{Prt}(\Delta)$ is *open* in Δ if there is no $p' \in \mathit{Prt}(\Delta)$ such that $(p, p') \in \mathit{Con}(\Delta)$; otherwise p is *connected*.

A *composite structure signature morphism* $\delta\colon \Delta \to \Delta'$ over $\mathcal{M}^{\downarrow}_{\mathcal{D}}$ consists of a function $Cmp(\delta)\colon Cmp(\Delta) \to Cmp(\Delta')$ mapping parts, together with an $\mathcal{M}^{\downarrow}_{\mathcal{D}}$-signature morphism $\Sigma(\delta, c)\colon \Sigma(\Delta, c) \to \Sigma(\Delta', Cmp(\delta)(c))$ for each $c \in Cmp(\Sigma)$; a function $Prt(\delta)\colon Prt(\Delta) \to Prt(\Delta')$ mapping ports, together with an $\mathcal{M}^{\downarrow}_{\mathcal{D}}$-signature morphism $Prt(\delta)(p)\colon \Sigma(\Delta, p) \to \Sigma(\Delta', Prt(\delta)(p))$, preserving

- the part owning each port p, i.e., $Cmp(\delta)(cmp(\Delta)(p)) = cmp(\Delta')(Prt(\delta)(p))$;
- the connections, i.e., if $(p, p') \in Con(\Delta)$, then $(Prt(\delta)(p), Prt(\delta)(p')) \in Con(\Delta')$.

The category of $\mathrm{cs}(\mathcal{M}^{\downarrow}_{\mathcal{D}})$-*signatures* $\mathbb{S}^{\mathrm{cs}(\mathcal{M}^{\downarrow}_{\mathcal{D}})}$ consists of the composite structure signatures and signature morphisms over $\mathcal{M}^{\downarrow}_{\mathcal{D}}$.

For an $\mathrm{cs}(\mathcal{M}^{\downarrow}_{\mathcal{D}})$-signature Δ, a Δ-*composite structure structure* (sic!) over $\mathcal{M}^{\downarrow}_{\mathcal{D}}$ is a family $\mathscr{C} \in (\mathscr{C}(c) \in |Str^{\mathcal{M}^{\downarrow}_{\mathcal{D}}}(\Sigma(\Delta, c))|)_{c \in Cmp(\Delta)}$ consisting of an $\mathcal{M}^{\downarrow}_{\mathcal{D}}$-structure for each part c. The δ-*reduct* $\mathscr{C}'|\delta$ of a Δ'-composite structure structure $\mathscr{C}'$ over $\mathcal{M}^{\downarrow}_{\mathcal{D}}$ along a composite structure signature morphism $\delta\colon \Delta \to \Delta'$ is computed component-wise as $(\mathscr{C}'(Cmp(\delta)(c))|\Sigma(\delta, c))_{c \in Cmp(\Delta)}$. The Δ-composite structure structures form the discrete category $Str^{\mathrm{cs}(\mathcal{M}^{\downarrow}_{\mathcal{D}})}(\Delta)$ of $\mathrm{cs}(\mathcal{M}^{\downarrow}_{\mathcal{D}})$-*structures* over Δ. For each signature morphism $\delta\colon \Delta \to \Delta'$ in $\mathbb{S}^{\mathrm{cs}(\mathcal{M}^{\downarrow}_{\mathcal{D}})}$ the δ-*reduct functor* $Str^{\mathrm{cs}(\mathcal{M}^{\downarrow}_{\mathcal{D}})}(\delta)\colon Str^{\mathrm{cs}(\mathcal{M}^{\downarrow}_{\mathcal{D}})}(\Delta') \to Str^{\mathrm{cs}(\mathcal{M}^{\downarrow}_{\mathcal{D}})}(\Delta)$ is given by $Str^{\mathrm{cs}(\mathcal{M}^{\downarrow}_{\mathcal{D}})}(\delta)(\mathscr{C}') = \mathscr{C}'|\delta$.

In UML, state machines organised in a composite structure communicate with each other by sending messages which are stored in event pools. A state machine draws a message from its event pool, which is typically implemented as an event queue, and reacts to this message by firing one of its enabled transitions or by discarding it when no transition is enabled. This communication scheme is obtained for a Δ-composite structure structure $\mathscr{C}$ over $\mathcal{M}^{\downarrow}_{\mathcal{D}}$ by constructing an overall $\mathcal{M}^{\downarrow}_{\mathcal{D}}$-structure over an $\mathcal{M}^{\downarrow}_{\mathcal{D}}$-signature that reflects the parts, the ports, and the connections in its events and attributes, but includes explicit event queues as additional attributes. The overall $\mathcal{M}^{\downarrow}_{\mathcal{D}}$-structure over this queue-based $\mathcal{M}^{\downarrow}_{\mathcal{D}}$-signature then implements the selection of an event from a part's event queue, the reactions of this part to this event, and the distribution of the produced messages to the connected parts.

Formally, we construct a functor $\Sigma_{\mathrm{q}}\colon \mathbb{S}^{\mathrm{cs}(\mathcal{M}^{\downarrow}_{\mathcal{D}})} \to \mathbb{S}^{\mathcal{M}^{\downarrow}_{\mathcal{D}}}$ on signatures that assigns to a composite structure signature Δ the *queue-based* event/data signature $\Sigma_{\mathrm{q}}(\Delta) = \bigcup_{c \in Cmp(\Delta)}(\Sigma(\Delta, c) \cup \{\mathrm{q}_c : \hat{I}(\Sigma(\Delta, c))^*\})$ and to a composite structure signature morphism the canonically corresponding event/data signature morphism. For a composite structure signature Δ and a part $c \in Cmp(\Delta)$ there is a natural signature embedding $\eta^{\mathrm{q}}_{\Delta,c}\colon \Sigma(\Delta, c) \to \Sigma_{\mathrm{q}}(\Delta)$.

For a Δ-composite structure structure $\mathscr{C}$ we construct an overall $\Sigma_{\mathrm{q}}(\Delta)$-event/data structure $M_{\mathscr{C}}$ as follows: An overall configuration of $M_{\mathscr{C}}$ consists, for each part $c \in Cmp(\Delta)$, of an *event queue* $q(c) \in \hat{I}(\Sigma(\Delta, c))^*$ stored in the attribute q_c and a part configuration $\gamma(c) \in \Gamma(\mathscr{C}(c))$; initially, all parts are in some of their initial configurations and all event queues are empty. In an overall configuration $(q(c), \gamma(c))_{c \in Cmp(\Delta)}$ an overall transition to another overall configuration $(q'(c), \gamma'(c))_{c \in Cmp(\Delta)}$ *reacts* to some $\hat{\imath} \in \hat{I}(\Sigma_{\mathrm{q}}(\Delta))$ and *outputs* some $\hat{O} \in \hat{O}(\Sigma_{\mathrm{q}}(\Delta))^*$. This $\hat{\imath}$ can either instantiate some provided event $i \in I(\Sigma(\Delta, p_*))$ of some of the open ports $p_* \in Prt(\Delta)$ with

$c_* = cmp(\Delta)(p)$, or it is the head of the event queue of some $c_* \in Cmp(\Delta)$ such that $i \in I(\Sigma(\Delta, c_*))$. In the latter case, $\hat{\imath}$ is removed from the event queue of c_*. In both cases, the reaction of part c_* is any transition $(\gamma(c_*), \gamma'_*) \in R(\mathscr{C}(c))_{\hat{\imath},\hat{O}}$ and overall $\gamma' = \gamma\{c_* \mapsto \gamma'_*\}$. Finally, all outputs $p.\hat{o} \in \hat{O}$ such that $(p, p') \in Con(\Delta)$ and $cmp(\Delta)(p') = c'$ are appended to the respective event queue of part c'. This defines a natural transformation $Str_{\mathrm{q}}^{\mathrm{cs}(\mathcal{M}^{\downarrow}_{\mathcal{D}})} \colon Str^{\mathrm{cs}(\mathcal{M}^{\downarrow}_{\mathcal{D}})} \dot{\to} \Sigma_{\mathrm{q}}; Str^{\mathcal{M}^{\downarrow}_{\mathcal{D}}}$ with $Str_{\mathrm{q},\Delta}^{\mathrm{cs}(\mathcal{M}^{\downarrow}_{\mathcal{D}})}(\mathscr{C}) = M_{\mathscr{C}}$.

Theorem 3. $(\mathbb{S}^{\mathrm{cs}(\mathcal{M}^{\downarrow}_{\mathcal{D}})}, Str^{\mathrm{cs}(\mathcal{M}^{\downarrow}_{\mathcal{D}})}, \mathrm{Sen}^{\mathrm{cs}(\mathcal{M}^{\downarrow}_{\mathcal{D}})}, \models^{\mathrm{cs}(\mathcal{M}^{\downarrow}_{\mathcal{D}})})$ *with* $\mathrm{Sen}^{\mathcal{M}^{\downarrow}_{\mathcal{D}}} = \Sigma_{\mathrm{q}}; \mathrm{Sen}^{\mathcal{M}^{\downarrow}_{\mathcal{D}}}$ *and* $\mathscr{C} \models_{\Delta}^{\mathrm{cs}(\mathcal{M}^{\downarrow}_{\mathcal{D}})} \varrho$ *if, and only if,* $Str_{\mathrm{q},\Delta}^{\mathrm{cs}(\mathcal{M}^{\downarrow}_{\mathcal{D}})}(\mathscr{C}) \models_{\Sigma_{\mathrm{q}}(\Delta)}^{\mathcal{M}^{\downarrow}_{\mathcal{D}}} \varrho$ *is an institution.*

$\mathrm{cs}(\mathcal{M}^{\downarrow}_{\mathcal{D}})$ inherits the event/data formulæ of $\mathcal{M}^{\downarrow}_{\mathcal{D}}$ and the underlying $\mathcal{D}$, though extended by queue attributes. In particular, we have for a part $c \in Cmp(\Delta)$ that a transition sentence $\langle\!| i : \phi \mathbin{/\!\!/} O : \psi |\!\rangle \varrho$ (in the current configuration there are valuations and a transition for the incoming message and the outgoing messages such that these valuations satisfy transition formula ψ and ϱ holds afterwards) locally formulated for this part can be faithfully transferred to the global composite structure, abbreviating the embedding $\eta^{\mathrm{q}}_{\Delta,c}$ to η,

$$\begin{aligned}\langle\!| \eta(i) : \mathscr{F}^{\mathcal{D}}_{A(\eta),X(i)}(\phi) \wedge (\mathrm{hd}(\mathrm{q}_c) = I(\eta)(i) \vee \mathrm{open}_{\Delta,c}(I(\eta)(i))) \mathbin{/\!\!/}\\ O(\eta)(O) : \mathscr{F}^{2\mathcal{D}}_{A(\eta),X(i)\cup X(O)}(\psi) \wedge\\ \textstyle\bigwedge_{a \in A(\Sigma_{\mathrm{q}}(\Delta)) \setminus (A(\Sigma(\Delta,c)) \cup \{\mathrm{q}_c \mid c \in Cmp(\Delta)\})} a = a' \wedge\\ \mathrm{dist}_{\Delta,c}(I(\eta)(i), O(\eta)(O), (\mathrm{q}_c, \mathrm{q}'_c)_{c \in Cmp(\Delta)}) |\!\rangle \mathrm{Sen}^{\mathcal{M}^{\downarrow}_{\mathcal{D}}}(\eta)(\varrho)\,,\end{aligned}$$

where hd yields the head of a queue, open checks whether the part's port for the event is open, the frame condition $a = a'$ ranges over all attributes not pertaining to c or the queues, dist removes the input and distributes the outputs to the queues.

6 Verification Example: Communication between `User`, `ATM` and `Bank`

We applied[4] the technique set out in this paper to the example from the introduction concerning a typical interaction between a `User`, an `ATM` component and a `Bank` component.

We formalised the state machines for the `Bank` and the `ATM` as well as their communication in CASL. We then set out to show a safety property (by means of a stronger invariant) on this system by inductive verification, as justified by Prop. 1. We first tried to show the preservation of said invariant using fully automatic provers connected to Hets [10], the main tool suite for verification based on CASL and institution theory. However, no inductive automated provers are currently connected to Hets. Therefore, handling freely generated datatype would require manual intervention to add suitable induction schemes — defeating our goal of automation. Instead we utilised the interactive theorem prover

[4] Full specifications and proofs accessible at: https://rosento.github.io/2021-paper-composite/

KIV [2]. This prover supports algebraic specifications similar to CASL and offers extensive heuristics for inductive proofs. KIV's heuristics fully automatically discharged all proof obligations in our experiments. The translation of the CASL specifications into KIV is straightforward.

With our process clarified, we can now state the safety property we will prove:

```
safe-def: safe(g) ↔ (ctrl(caConf(g)) = Verified → wasVerified(cbConf(g)) = 1);
used for: s, ls;
```

The above introduces an axiom `safe-def` defining the predicate `safe` and marks the axiom for use as a simplifier rule (`s`) and a local simplifier rule (`ls`) for the KIV system.

The predicate `safe` ranges over a type of system configurations, each consisting of the `ATM` configuration (`caConf`) and queue, as well as the bank configuration (`cbConf`) and queue. The machine configurations in turn consist of the control state and attributes. The safety predicate holds in a configuration *iff* should the `ATM` be in control state `Verified`, the bank attribute `wasVerified` has the value 1.

The behaviours of `Bank` and `ATM` are defined in the form of an initial state predicate and a transition predicate. For space reasons we show only one transition:

```
atmTrans-def: atmTrans(atmConf(sa1, c1, p1, t1), in, out, atmConf(sa2, c2, p2, t2))
  ↔ ∃ c : CardId, p : Pin . ...
       ∨ (   sa1 = CardEntered
          ∧ in = msg(userCom, PIN(p)) ∧ out = (msg(atmCompl, PINEnteredCompl) +l [])
          ∧ p2 = p ∧ sa2 = PINEntered ∧ c2 = c1 ∧ t2 = t1)
       ∨ ...; used for: s, ls;
```

The `ATM` transitions from one configuration to another, receiving an input event and sending out a list of messages. Each `ATM` configuration consists of (in that order) the control state, the card id to be verified, the PIN to be verified and the counter for the number of verification attempts. We give the definition of the transition predicate by a disjunction of the conditions of all syntactic transitions, including the control state before, the input event, the output list, variables to be set, the control state after and variables to remain unchanged. Given these machine predicates and a predicate `dist` to encode connectors, we can then define the transition predicate for the overall system:

```
trans-def: trans(conf(ca1, qa1, cb1, qb1), in, out, conf(ca2, qa2, cb2, qb2))
  ↔   dist(out, qa1, qa2, qb1, qb2)
    ∧ (   (atmTrans(ca1, in, out, ca2) ∧ cb2 = cb1 )
        ∨ (bankTrans(cb1, in, out, cb2) ∧ ca2 = ca1)); used for: s, ls;
```

Initially, the queues are empty and the machines are in their initial configurations.

Having thus defined the machines, we turn to verification and define an invariant strong enough to show both its own preservation and our safety property. The idea is to control the queues' status that allows us to enter the `Verified` state on the `ATM` or to reset the `wasVerified` attribute. In essence the invariant can be syntactically read off from the composite structure.

```
invar-def: invar(conf(ca, qa, cb, qb)) ↔ ∃ x.
    (ctrl(ca) = Idle         ∧ ctrl(cb) = Idle         ∧ qa = empty ∧ qb = empty)
  ∨ (ctrl(ca) = CardEntered  ∧ ctrl(cb) = Idle         ∧ qa = empty ∧ qb = empty)
  ∨ (ctrl(ca) = PINEntered   ∧ ctrl(cb) = Idle         ∧ qa = enq(x, empty) ∧ qb = empty)
  ∨ (ctrl(ca) = Verifying    ∧ ctrl(cb) = Idle         ∧ qa = empty ∧ qb = enq(x, empty))
  ∨ (ctrl(ca) = Verifying    ∧ ctrl(cb) = Verifying    ∧ qa = empty ∧ qb = enq(x, empty))
  ∨ (ctrl(ca) = Verifying    ∧ ctrl(cb) = VeriSuccess ∧
                qa = empty ∧ qb = enq(x, empty) ∧ wasVerified(cb) = 1)
```

```
  ∨ (ctrl(ca) = Verifying   ∧ ctrl(cb) = VeriFail    ∧ qa = empty ∧ qb = enq(x,empty))
  ∨ (ctrl(ca) = Verifying   ∧ ctrl(cb) = Idle        ∧
                qa = enq(msg(bankCom, reenterPIN), empty) ∧ qb = empty)
  ∨ (ctrl(ca) = Verifying   ∧ ctrl(cb) = Idle        ∧
                qa = enq(msg(bankCom, verified), empty) ∧ qb = empty ∧ wasVerified(cb) = 1)
  ∨ (ctrl(ca) = Verified    ∧ ctrl(cb) = Idle        ∧
                qa = enq(x, empty) ∧ qb = empty ∧ wasVerified(cb) = 1); used for: s, ls;
```

Note that we can mostly ignore attribute values, as well as all distinctions between queue elements unrelated to our verification task. We can then formulate lemmas to the effect that this invariant does in fact imply the safety property, that it is satisfied in any legal initial configurations and that it is preserved by all transitions. These lemmas are as follows, again limited to one example for the transitions:

```
Safe: invar(g) → safe(g);
Init: init(g) → invar(g);
...
Trans6: g1 = conf(atmConf(Verifying, c, p, t), qa, cb, qb)
      ∧ qa ≠ empty ∧ top(qa) = msg(atmCom, verified)
      ∧ g2 = conf(atmConf(Verified, c, p, t),
                  enq(msg(atmCompl, VerifiedCompl), deq(qa)), cb, qb)
      ∧ invar(g1) → invar(g2);
```

Formulating separate lemmas for each transition instead of one lemma using the transition predicate helps us avoid a combinatorial explosion in the theorem prover.

Providing our specification to KIV with all definitions marked as simplifier rules and activating the heuristics mode “PL heuristics + structural induction”, each of our lemmas is proved without noticeable delay, i.e., the verification of the invariant is successful and does not pose any difficulty to the prover.

7 Conclusion

We have developed two new institutions extending the hybrid modal logic $\mathcal{M}^{\downarrow}_{\mathcal{D}}$ [16]. One institution caters for simple UML state machines with outputs, an extension of it captures simple UML composite structure diagrams. Besides providing formal semantics for communicating UML state machines, via comorphisms these institutions provide a bridge towards theorem proving for UML. Through an elementary example we could demonstrate that, thanks to our framework, effective automated theorem proving for communicating UML state machines is possible.

Future work will be on proof automation. In particular we plan to implement the translations from UML into extended $\mathcal{M}^{\downarrow}_{\mathcal{D}}$, the institution comorphisms from extended $\mathcal{M}^{\downarrow}_{\mathcal{D}}$ to CASL, and possibly the link from Hets to KIV. Yet another important aspect is to implement analyses of the composite structure and its state machines with a view to automatically generate lemmas for automated theorem proving. In terms of our general research programme, the next topic to tackle are UML interactions and how they relate or refine to UML state machines. Going beyond the UML, it would be interesting to consider a truly heterogeneous framework, in which composite structure diagrams connect not only UML state machines, but also components specified in languages such as TLA or Event-B.

References

1. Blackburn, P., de Rijke, M., Venema, Y.: Modal Logic, Cambridge Tracts in Theoretical Computer Science, vol. 53. Cambridge University Press (2001)
2. Ernst, G., Pfähler, J., Schellhorn, G., Haneberg, D., Reif, W.: KIV: Overview and VerifyThis Competition. Intl. J. Softw. Tools Technol. Transfer **17**(6), 677–694 (2015)
3. Goguen, J.A., Burstall, R.M.: Institutions: Abstract Model Theory for Specification and Programming. J. ACM **39**, 95–146 (1992)
4. Knapp, A., Mossakowski, T.: UML Interactions Meet State Machines — An Institutional Approach. In: Bonchi, F., König, B. (eds.) Proc. 7th Intl. Conf. Algebra and Coalgebra in Computer Science (CALCO 2017). LIPIcs, vol. 72, pp. 15:1–15:15 (2017)
5. Knapp, A., Mossakowski, T., Roggenbach, M.: Towards an Institutional Framework for Heterogeneous Formal Development in UML — A Position Paper. In: De Nicola, R., Hennicker, R. (eds.) Software, Services, and Systems — Essays Dedicated to Martin Wirsing on the Occasion of His Retirement from the Chair of Programming and Software Engineering, Lect. Notes Comp. Sci., vol. 8950, pp. 215–230. Springer (2015)
6. Knapp, A., Mossakowski, T., Roggenbach, M., Glauer, M.: An Institution for Simple UML State Machines. In: Egyed, A., Schaefer, I. (eds.) Proc. 18th Intl. Conf. Fundamental Approaches to Software Engineering (FASE 2015). Lect. Notes Comp. Sci., vol. 9033, pp. 3–18. Springer (2015)
7. Liu, S., Liu, Y., Étienne André, Choppy, C., Sun, J., Wadhwa, B., Dong, J.S.: A Formal Semantics for Complete UML State Machines with Communications. In: Johnsen, E.B., Petre, L. (eds.) Proc. 10th Intl. Conf. Integrated Formal Methods (IFM 2013). Lect. Notes Comp. Sci., vol. 7940, pp. 331–346. Springer (2013)
8. Mazzanti, F., Ferrari, A., Spagnolo, G.O.: The KandISTI/UMC Online Open-Access Verification Framework. ERCIM News **109** (2017)
9. Mossakowski, T.: Relating CASL with Other Specification Languages: The Institution Level. Theo. Comp. Sci. **286**(2), 367–475 (2002)
10. Mossakowski, T., Maeder, C., Lüttich, K.: The Heterogeneous Tool Set. In: Grumberg, O., Huth, M. (eds.) Proc. 13th Intl. Conf. Tools and Algorithms for the Construction and Analysis of Systems (TACAS 2007). Lect. Notes Comp. Sci., vol. 4424, pp. 519–522. Springer (2007)
11. Mosses, P.D.: CASL Reference Manual — The Complete Documentation of the Common Algebraic Specification Language, Lect. Notes Comp. Sci., vol. 2960. Springer (2004)
12. Ober, I., Dragomir, I.: Unambiguous UML Composite Structures: The OMEGA2 Experience. In: Cerná, I., Gyimóthy, T., Hromkovic, J., Jeffery, K.G., Královic, R., Vukolic, M., Wolf, S. (eds.) Proc. 37th Conf. Current Trends in Theory and Practice of Computer Science (SOFSEM 2011), Lect. Notes Comp. Sci., vol. 6543, pp. 418–430. Springer (2011)
13. Object Management Group: Unified Modeling Language. Standard formal/2017-12-05, OMG (2017), https://www.omg.org/spec/UML/2.5.1
14. Object Management Group: Precise Semantics of UML Composite Structures. Standard formal/2019-02-01, OMG (2019), https://www.omg.org/spec/PSCS/1.2
15. Rosenberger, T.: Relating UML State Machines and Interactions in an Institutional Framework. Master's thesis, Universität Augsburg, Ludwig-Maximilians-Universität München, Technische Universität München (2017)
16. Rosenberger, T., Bensalem, S., Knapp, A., Roggenbach, M.: Institution-based Encoding and Verification of Simple UML State Machines in CASL/SPASS. In: Roggenbach, M. (ed.) Rev. Sel. Papers 25th Intl. Ws. Recent Trends in Algebraic Development Techniques (WADT 2020). Lect. Notes Comp. Sci., vol. 12669, pp. 120–141. Springer (2020)