https://doi.org/10.1007/s00165-021-00550-7
The Author(s) © 2021
Formal Aspects of Computing

Formal Aspects
of Computing

Hybrid dynamic logic institutions for event/data-based systems

Rolf Hennicker[a], Alexander Knapp[b], and Alexander Madeira[c] [1]

[a]Ludwig-Maximilians-Universität München, Munich, Germany
[b]Universität Augsburg, Augsburg, Germany
[c]CIDMA, Mathematics Dep. of U. Aveiro, Aveiro, Portugal

Abstract. We propose $\mathcal{E}^{\downarrow}(\vec{\mathcal{D}})$-logic as a formal foundation for the specification and development of event-based systems with data states. The framework is presented as an institution in the sense of Goguen and Burstall and the logic itself is parametrised by an underlying institution $\vec{\mathcal{D}}$ whose structures are used to model data states. $\mathcal{E}^{\downarrow}(\vec{\mathcal{D}})$-logic is intended to cover a broad range of abstraction levels from abstract requirements specifications up to constructive specifications. It uses modal diamond and box operators over complex actions adopted from dynamic logic. Atomic actions are pairs $e /\!\!/ \psi$ where e is an event and ψ a state transition predicate capturing the allowed reactions to the event. To write concrete specifications of recursive process structures we integrate (control) state variables and binders of hybrid logic. The semantic interpretation relies on event/data transition systems. For the presentation of constructive specifications we propose operational event/data specifications allowing for familiar, diagrammatic representations by state transition graphs. We show that $\mathcal{E}^{\downarrow}(\vec{\mathcal{D}})$-logic is powerful enough to characterise the semantics of an operational specification by a single $\mathcal{E}^{\downarrow}(\vec{\mathcal{D}})$-sentence. Thus the whole (formal) development process for event/data-based systems relies on $\mathcal{E}^{\downarrow}(\vec{\mathcal{D}})$-logic and its semantics as a common basis. It is supported by a variety of implementation constructors which can express, among others, event refinement and parallel composition. Due to the genericity of the approach, it is also possible to change a data state institution during system development when needed. All steps of our formal treatment are illustrated by a running example.

1. Introduction

Event-based systems are an important kind of software systems that are open to the environment to react to certain events. A crucial characteristic of such system is that their reaction to events will differ over time due to their internal state; in particular, not any event may be meaningful at any time. Hence the control flow of the system is significant and should be modelled by appropriate means. On the other hand components administer data which may change upon the occurrence of an event. Thus also the specification of admissible data changes caused by events plays a major role.

There is quite a lot of literature on modelling and specifying event-based systems. Several approaches, often underpin ned by graphical notations, provide formalisms aiming at being constructive enough to suggest particular designs or implementations, like e.g., Event-B [Abr13, FMP17], symbolic transition systems [PR06], and UML behavioural and protocol state machines [OMG17, KMRG15]. On the other hand, there are logical formalisms

Correspondence to: Alexander Knapp. E-mail: knapp@informatik.uni-augsburg.de

[1] A. Madeira is supported by the European Regional Development Fund through the Operational Programme for Competitiveness and Internationalisation - COMPETE 2020 and by National Funds through the Portuguese funding agency FCT - within the Projects POCI-01-0145-FEDER-029946 and UID/MAT/04106/2019.

Published online: 29 July 2021

to express desired properties of event-based systems. Among them are temporal logics integrating state and event-based styles [tBFGM08], and various kinds of modal logics involving data, like first-order dynamic logic [HKT00] or the modal μ-calculus with data and time [GM14]. The gap between logics and constructive specification is usually filled by checking whether *the* model of a constructive specification satisfies certain logical formulæ.

In this paper we are interested in investigating a logic which is capable to express properties of event/data-based systems on various abstraction levels in a common formalism. For this purpose we follow ideas of [MBHM18], but there data states, effects of events on them and constructive operational specifications (see below) were not considered. The advantage of an expressive logic is that we can split the transition from system requirements to system implementation into a series of gradual refinement steps which are more easy to understand, to verify, and to adjust when certain aspects of the system are to be changed or when a product line of similar products has to be developed.

To that end we propose $\mathcal{E}^{\downarrow}(\vec{\mathcal{D}})$-logic, a dynamic logic enriched with features of hybrid logic and parametrised by an underlying logic $\vec{\mathcal{D}}$ for data states. The dynamic part uses diamond and box operators, $\langle\lambda\rangle\varrho$ and $[\lambda]\varrho$ resp., over structured actions λ adopted from dynamic logic [HKT00]. Atomic actions are of the form $e \mathbin{/\!\!/} \psi$ with e an event and ψ a state transition predicate specifying the admissible effects of e on the data. Using sequential composition, union, and iteration we obtain complex actions that, in connection with the modal operators, can be used to specify required and forbidden behaviours. In particular, if E is a finite set of events, though data is infinite we are able to capture all states of the system, reachable from initial ones by events of E, and thus to express liveness and safety properties, the latter by sentences of the form $[\boldsymbol{E}^*]\varrho$. But $\mathcal{E}^{\downarrow}(\vec{\mathcal{D}})$-logic is also powerful enough to specify concrete, recursive process structures by integrating state variables x, binders $\downarrow x\,.\,\varrho$ and jumps $(@^F x)\varrho$ from hybrid logic [Bra10] with the subtle difference that our state variables are used to denote control states only and that we relativise jumps by a set F of events.

An axiomatic specification $Sp = (\Sigma, Ax)$ in $\mathcal{E}^{\downarrow}(\vec{\mathcal{D}})$ is given by an event/data signature $\Sigma = (E, \delta)$ with a set E of events and a data signature δ to model data states, and a set of $\mathcal{E}^{\downarrow}(\vec{\mathcal{D}})$-sentences Ax, called axioms, describing desired system properties. For the semantic interpretation we use event/data transition systems (edts). Their states are reachable configurations (c, d) showing a control state c that records the current state of execution, and a data state d. Transitions between configurations are labelled by events. The semantics of a specification Sp is “loose” in the sense that it consists of *all* edts satisfying the axioms of the specification. Such structures are called models of Sp. Loose semantics allows us to define a simple refinement notion: Sp_1 refines to Sp_2 if the model class of Sp_2 is included in the model class of Sp_1. We may also say that Sp_2 is an implementation of Sp_1.

Our refinement process starts typically with axiomatic specifications whose axioms involve only the dynamic part of the logic. Hybrid features will successively be added in refinements when specifying more concrete behaviours by introducing variables for control states, variable binders and jumps. Aiming at a concrete design, the use of an axiomatic specification style may, however, become cumbersome since we have to state explicitly also all negative cases, what the system should not do. For a convenient presentation of constructive specifications we propose operational event/data specifications, which are a kind of symbolic transition systems equipped (again) with a model class semantics in terms of edts. We will show that $\mathcal{E}^{\downarrow}(\vec{\mathcal{D}})$-logic, by use of the hybrid features, is powerful enough to characterise the semantics of an operational specification. Therefore, we do not really move outside $\mathcal{E}^{\downarrow}(\vec{\mathcal{D}})$-logic when refining axiomatic by operational specifications. Moreover, since several constructive notations in the literature, including (essential parts of) Event-B, symbolic transition systems, and UML protocol state machines, can be expressed as operational specifications, $\mathcal{E}^{\downarrow}(\vec{\mathcal{D}})$-logic provides a logical umbrella under which event/data-based systems can be developed.

In order to consider more complex kinds of refinements we take up an idea of Sannella and Tarlecki [ST88, ST12] who have proposed the notion of constructor implementation. This is a generic notion applicable to specification formalisms based on signatures and their semantic structures. As both are available in the context of $\mathcal{E}^{\downarrow}(\vec{\mathcal{D}})$-logic, we complement our approach by introducing a couple of constructors, among them event refinement and parallel composition. For the latter we provide a powerful refinement criterion relying on a relationship between syntactic and semantic parallel composition. We show under which additional assumptions our criterion is even a necessary condition which sharpens a corresponding result in [HMK19]. The logic and the use of the implementation constructors will be illustrated by a running example. We have also implemented a small, prototypical tool[2] that allows to check $\mathcal{E}^{\downarrow}(\vec{\mathcal{D}})$-formulæ on finite-state edts and the refinement of finite-state operational specifications using constructor implementations such that the running example can be reproduced.

[2] Available at https://bitbucket.org/knappale/edhl/.

This paper is a significant extension of the conference paper [HMK19]. We provide a formalisation of our specification theory for event data-based systems as an "institution". The notion of an institution has been proposed by Goguen and Burstall in [GB92] as an abstract concept to capture the essential ingredients that a logical system should provide when being used in formal software development. Since then the ideas of an institution became quite popular and many different logical systems have been presented as institutions; see [ST12] for an overview. Formalising a logical system as an institution has several advantages: It provides a better insight in the used concepts with a clear structure concerning syntax (in terms of signatures and sentences), semantics (in terms of mathematical structures) and the relationship between the two in terms of a satisfaction relation. Having an institution supports the process of software development by formalising system extensions, reducts and component-wise development by means of signature morphisms. It also allows to reuse abstract results that are valid in arbitrary institutions concerning, e.g., structured specifications and refinement. All this does not come for free but with a proof obligation: The logical system at hand must guarantee the "satisfaction condition" which expresses that validity of logical sentences is invariant under change of context or notation. This is an important requirement for modular software design.

We present $\mathcal{E}^{\downarrow}(\vec{\mathcal{D}})$-logic as a generic institution which is parametrised by an underlying data state institution $\vec{\mathcal{D}}$. In particular, we show that for $\mathcal{E}^{\downarrow}(\vec{\mathcal{D}})$ the satisfaction condition holds. The proof relies, besides on the satisfaction condition for $\vec{\mathcal{D}}$, (a) on the data state labelling for configurations of event/data transition systems and (b) on a generalisation of the jump operator $(@x)\varrho$ of hybrid logic to a relativised jump operator $(@^{F}x)\varrho$ which moves the state of evaluation to all configurations whose control state is denoted by x *and* which are reachable by an explicitly stated set F of events. This way the scope of the events usable for "jumping" remains under control of the formula and thus is independent of the context in which the formula is used. Both, (a) and (b) were not incorporated in [HMK19] and the logic there could not be reused as it is to get an institution.

Due to the genericity of $\mathcal{E}^{\downarrow}(\vec{\mathcal{D}})$-logic, any concrete data state institution satisfying a few assumptions, most importantly the amalgamation property, can be used to instantiate $\mathcal{E}^{\downarrow}(\vec{\mathcal{D}})$. Of course, this improves significantly the applicability of $\mathcal{E}^{\downarrow}(\vec{\mathcal{D}})$. An important consequence is also that during system development the data state institution can be changed if different, usually more expressive, constructs are needed when moving towards an implementation. As a tool for sound migration of data state institutions we use institution comorphisms. How this works is exemplified by our running example.

Being parametric in the underlying data state institution requires a proper institutional treatment of pairs of data states representing pre- and post-states of transitions. For this purpose, we introduce the novel concept of a 2-data state institution $2\vec{\mathcal{D}}$ over $\vec{\mathcal{D}}$ which, by itself, is a new research result that should be applicable to institutionalise other kinds of pre/post-condition style specification formats. The formalisation of $2\vec{\mathcal{D}}$ uses pushouts and amalgamations. The latter allow to combine two data states, a pre- and a post-state, over which state transition predicates can be interpreted. These predicates are formalised as sentences of the corresponding pushout signature. The base signature of such pushouts models "rigid" symbols, i.e., symbols whose interpretation remains unchanged when moving from one data state to another.

Related to the genericity of $\mathcal{E}^{\downarrow}(\vec{\mathcal{D}})$-logic are approaches which deal with "temporalisation" [FG92], "modalisation" [FF02, DS07] and "hybridisation" [MMDB11, DM16]. In these papers the idea is to extend an arbitrary base logic or institution with temporal, modal, or hybrid features and thus to be able to work out the characteristic features of the respective logical extensions and to study preservation of certain properties. The motivation of our work is not to find yet another hybridisation process but to provide support for system development. As a consequence, we consider (a) only models reachable from initial states, (b) operational specifications, and (c) implementation constructors useful in a refinement methodology. In this context, an important point for us is to be able to express safety and liveness properties by navigation through all reachable states and to be able to express concrete process structures as well by using operational specifications, which can be equivalently expressed in our logic by binders and (relativised) jumps. A relevant point is also that our 2-data state institutions allow to relate logically pre- and post-states of transitions on the basis of pushout constructions and amalgamated unions. Therefore neither quantification of hybrid logical formulæ is needed as in [MMDB11] nor a particular "rigidification" of symbols as in [DM16].

Outline. The remainder of this paper is structured as follows: In Sect. 2 we recall the notion of an institution and some related concepts. Then, in Sect. 3, we consider institutions for data states and their transitions. The constituents of the generic $\mathcal{E}^{\downarrow}(\vec{\mathcal{D}})$-logic institution are introduced in Sect. 4. A significant part of this section concerns the proof of the satisfaction condition. In Sect. 5 we consider axiomatic as well as operational specifications of event/data-based systems and demonstrate the expressiveness of $\mathcal{E}^{\downarrow}(\vec{\mathcal{D}})$-logic. Refinement of both types of specifications using several implementation constructors is considered in Sect. 6. Section 7 provides some concluding remarks.

Readers who are more interested in the stepwise (hierarchical) formalisation of institutions for data states ($\vec{\mathcal{D}}$), data state transitions ($2\vec{\mathcal{D}}$), and the hybrid dynamic logic $\mathcal{E}^{\downarrow}(\vec{\mathcal{D}})$ built on top of these can concentrate on Sections 2 to 4; readers who want to focus on the usage of $\mathcal{E}^{\downarrow}(\vec{\mathcal{D}})$ as a methodology for the stepwise development of event/data-based systems can put their attention to Sections 4 to 6 with some occasional references to Sections 2 and 3.

2. Institutions

The concept of an institution has been introduced in [GB92]. It formalises some basic ingredients that a logical system should provide when it is used as a specification framework in program development. The notion relies on a clear separation between syntax (signatures, sentences) and semantics (models) such that models and sentences are related by a satisfaction relation. Differently to the original terminology models will be called structures to avoid ambiguity when we talk about models of a specification later on. For the categorical terminology used in the sequel of this paper we refer the reader to the book by Mac Lane [Mac98] (where, however, functional order of morphism composition $g \circ f$ is used instead of the diagrammatic order $f;\ g$ often employed here).

Definition 1 An *institution* $(\mathbb{S}, Str, \mathrm{Sen}, \models)$ consists of

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

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

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

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

$$\begin{tikzcd}
\Sigma \arrow[d, "\sigma"'] & Str(\Sigma) \arrow[r, dash, "\models_\Sigma"] & \mathrm{Sen}(\Sigma) \arrow[d, "\mathrm{Sen}(\sigma)"] \\
\Sigma' & Str(\Sigma') \arrow[u, "Str(\sigma)"] \arrow[r, dash, "\models_{\Sigma'}"] & \mathrm{Sen}(\Sigma')
\end{tikzcd}$$

□

Example 1 We sketch *propositional logic* and *many-sorted first-order logic with equality* as institutions $Prop$ and $FO^{=}$, respectively (for detailed expositions see, e.g., [ST12]).

(a) Propositional logic $Prop = (\mathbb{S}^{Prop}, Str^{Prop}, \mathrm{Sen}^{Prop}, \models^{Prop})$.
The *category of propositional signatures* $\mathbb{S}^{Prop}$ has sets P of *propositional variables* as objects and functions $\pi : P \to P'$ as morphisms.
The *propositional structures functor* $Str^{Prop} : (\mathbb{S}^{Prop})^{\mathrm{op}} \to \mathrm{Cat}$ maps each signature $P \in |\mathbb{S}^{Prop}|$ to the category $Str^{Prop}(P)$ with functions $\mu : P \to \mathbb{B} = \{\mathit{ff}, \mathit{tt}\}$ as objects and $h : \mu_1 \to \mu_2$ a (unique) morphism from $\mu_1 : P \to \mathbb{B}$ to $\mu_2 : P \to \mathbb{B}$ if $\{p \mid \mu_1(p) = \mathit{tt}\} \subseteq \{p \mid \mu_2(p) = \mathit{tt}\}$; and each signature morphism $\pi : P \to P'$ to the reduct functor $Str^{Prop}(\pi) : Str^{Prop}(P') \to Str^{Prop}(P)$ defined by $Str^{Prop}(\pi)(\mu') = \pi;\ \mu'$ for each $\mu' \in |Str^{Prop}(P')|$ such that indeed $Str^{Prop}(\pi)(h') : Str^{Prop}(\pi)(\mu'_1) \to Str^{Prop}(\pi)(\mu'_2)$ for $h' : \mu'_1 \to \mu'_2$ in $Str^{Prop}(P')$.
The *propositional sentences functor* $\mathrm{Sen}^{Prop} : \mathbb{S}^{Prop} \to \mathrm{Set}$ maps each signature $P \in |\mathbb{S}^{Prop}|$ to the set $\mathrm{Sen}^{Prop}(P)$ defined by the grammar

$$\rho ::= p \mid \mathrm{true} \mid \neg\rho \mid \rho_1 \vee \rho_2 \mid \rho_1 \wedge \rho_2 \mid \rho_1 \rightarrow \rho_2 \mid \rho_1 \leftrightarrow \rho_2$$

for $p \in P$; and each morphism $\pi : P \to P'$ to the sentence translation $\mathrm{Sen}^{Prop}(\sigma) : \mathrm{Sen}^{Prop}(P) \to \mathrm{Sen}^{Prop}(P')$ replacing each propositional variable p by $\sigma(p)$.

Finally, for each signature $P \in |\mathbb{S}^{Prop}|$, each structure $\mu \in |Str^{Prop}(P)|$, and each sentence $\rho \in \mathrm{Sen}^{Prop}(P)$ the propositional satisfaction relation $\mu \models_P^{Prop} \rho$ is defined inductively as usual:

- $\mu \models_P^{Prop} p$ iff $\mu(p) = tt$;
- $\mu \models_P^{Prop}$ true;
- $\mu \models_P^{Prop} \neg\rho$ iff not $\mu \models_P^{Prop} \rho$;
- $\mu \models_P^{Prop} \rho_1 \vee \rho_2$ iff $\mu \models_P^{Prop} \rho_1$ or $\mu \models_P^{Prop} \rho_2$;

and similarly for the other connectives, such that the satisfaction condition is fulfilled.

(b) First-order logic with equality $FO = (\mathbb{S}^{FO^=}, Str^{FO^=}, \mathrm{Sen}^{FO^=}, \models^{FO^=})$.

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

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

For constructing terms and formulæ over a many-sorted signature $\Sigma = (S, F)$ an S-indexed family of *variables* X is assumed. The $S(\Sigma)$-indexed family of *terms* $\mathscr{T}(\Sigma, X)$ is inductively given by $x \in \mathscr{T}(\Sigma, X)_s$ for $x \in X_s$ and $f(t_1, \ldots, t_n) \in \mathscr{T}(\Sigma, X)_s$ for $f \in F$ with arguments sorts $s_1, \ldots, s_n$ and result sort s and $t_i \in \mathscr{T}(\Sigma, X)_{s_i}$. The set of *formulæ* $\mathscr{F}(\Sigma, X)$ is given by the grammar

$$\varphi ::= t_1 = t_2 \mid \text{true} \mid \neg\varphi \mid \varphi_1 \vee \varphi_2 \mid \forall\, x : s \,.\, \varphi\,,$$

where in $\forall\, x : s \,.\, \varphi$ the variable x is *bound* by the quantifier. Term and formulæ translation along a many-sorted signature morphism $\sigma : \Sigma \to \Sigma'$ preserve the term and formulæ structure as well as unbound, *free* variables. The *sentence functor* $\mathrm{Sen}^{FO^=} : \mathbb{S}^{FO^=} \to \mathrm{Set}$ maps Σ to the *sentences* over Σ, i.e., the formulæ over Σ that show no free variables, and $\sigma : \Sigma \to \Sigma'$ to the formula translation along σ.

For a Σ-algebra $\mathfrak{A}$ and a *valuation* $\beta = (\beta_s : X_s \to s^{\mathfrak{A}})_{s \in S(\Sigma)}$ the *term evaluation* $\beta^* = (\beta^*_s : \mathscr{T}(\Sigma, X)_s \to s^{\mathfrak{A}})_{s \in S(\Sigma)}$ is inductively given by $\beta^*_s(x) = \beta_s(x)$ and $\beta^*_s(f(t_1, \ldots, t_n)) = f^{\mathfrak{A}}(\beta^*_{s_1}(t_1), \ldots, \beta^*_{s_n}(t_n))$. *Formula satisfaction* $\mathfrak{A}, \beta \models_\Sigma^{FO^=} \varphi$ for a formula $\varphi \in \mathscr{F}(\Sigma, X)$ is inductively given by

- $\mathfrak{A}, \beta \models_\Sigma^{FO^=} t_1 = t_2$ iff $\beta^*(t_1) = \beta^*(t_2)$;
- $\mathfrak{A}, \beta \models_\Sigma^{FO^=}$ true;
- $\mathfrak{A}, \beta \models_\Sigma^{FO^=} \neg\varphi$ iff not $\mathfrak{A}, \beta \models_\Sigma^{FO^=} \varphi$;
- $\mathfrak{A}, \beta \models_\Sigma^{FO^=} \varphi_1 \vee \varphi_2$ iff $\mathfrak{A}, \beta \models_\Sigma^{FO^=} \varphi_1$ or $\mathfrak{A}, \beta \models_\Sigma^{FO^=} \varphi_2$;
- $\mathfrak{A}, \beta \models_\Sigma^{FO^=} \forall\, x : s \,.\, \varphi$ iff $\mathfrak{A}, \beta\{x : s \mapsto a\} \models_\Sigma^{FO^=} \varphi$ for all $a \in s^{\mathfrak{A}}$,
  where $\beta\{x : s \mapsto a\}(x) = a$ and $\beta\{x : s \mapsto a\}(y) = \beta(y)$ for $y \neq x$.

Finally, the satisfaction relation $\mathfrak{A} \models_\Sigma^{FO^=} \varphi$ for a Σ-algebra $\mathfrak{A} \in |Str^{FO^=}(\Sigma)|$ and a sentence $\varphi \in \mathrm{Sen}^{FO^=}(\Sigma)$ is defined by $\mathfrak{A}, \beta \models_\Sigma^{FO^=} \varphi$ for an arbitrary valuation β (as φ shows no free variables). □

Let us now recall some useful properties of institutions.

Amalgamation Property

The amalgamation property will be used later to construct the union of two data states (intuitively pre- and post-states) over which state transition predicates, formalised as sentences over a pushout signature, can be interpreted.

An institution $(\mathbb{S}, \mathit{Str}, \mathrm{Sen}, \models)$ has the *amalgamation property* [ST12, Def. 4.4.12] if all pushouts in $\mathbb{S}$ exist and every pushout diagram

$$\begin{tikzcd}
& \Sigma_1 +^{\sigma_1,\sigma_2}_{\Sigma_0} \Sigma_2 & \\
\Sigma_1 \arrow[ur, "\hat\sigma_1"] & & \Sigma_2 \arrow[ul, "\hat\sigma_2"'] \\
& \Sigma_0 \arrow[ul, "\sigma_1"] \arrow[ur, "\sigma_2"'] &
\end{tikzcd}$$

admits amalgamation:

$$\begin{tikzcd}
& M_1 \times_{\sigma_1,\sigma_2} M_2 \arrow[dl, mapsto, "\mathit{Str}(\hat\sigma_1)"'] \arrow[dr, mapsto, "\mathit{Str}(\hat\sigma_2)"] & \\
M_1 \arrow[dr, mapsto, "\mathit{Str}(\sigma_1)"'] & & M_2 \arrow[dl, mapsto, "\mathit{Str}(\sigma_2)"] \\
& M_0 &
\end{tikzcd}$$

In more detail, $(\Sigma_1 +^{\sigma_1,\sigma_2}_{\Sigma_0} \Sigma_2, (\hat\sigma_i : \Sigma_i \to \Sigma_1 +^{\sigma_1,\sigma_2}_{\Sigma_0} \Sigma_2)_{1 \le i \le 2})$ is a *pushout* along $\sigma_1 : \Sigma_0 \to \Sigma_1$ and $\sigma_2 : \Sigma_0 \to \Sigma_2$ in $\mathbb{S}$ if $\sigma_1;\ \hat\sigma_1 = \sigma_2;\ \hat\sigma$ and, furthermore, for all $\Sigma' \in |\mathbb{S}|$ and $\sigma'_i : \Sigma_i \to \Sigma'$, $1 \le i \le 2$ satisfying $\sigma_1;\ \sigma'_1 = \sigma_2;\ \sigma'_2$, there is a unique $\sigma : \Sigma_1 +^{\sigma_1,\sigma_2}_{\Sigma_0} \Sigma_2 \to \Sigma'$ with $\sigma'_i = \hat\sigma_i;\ \sigma$, $1 \le i \le 2$. Such a pushout admits *amalgamation* if

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

Example 2 In the propositional logic institution Prop of Ex. 1(a) a pushout along two propositional signature morphisms $\pi_1 : P_0 \to P_1$ and $\pi_2 : P_0 \to P_2$ in $\mathbb{S}^{\mathit{Prop}}$ is obtained as follows: Let $P_1 \uplus P_2 = \{(1, p) \mid p \in P_1\} \cup \{(2, p) \mid p \in P_2)\}$ be the disjoint union of P_1 and P_2 and let $\sim^{\pi_1,\pi_2}_{P_0} \subseteq (P_1 \uplus P_2)^2$ be the smallest equivalence relation containing $\{(1, \pi_1(p)), (2, \pi_2(p)) \mid p \in P_0\}$. Then $P_1 +^{\pi_1,\pi_2}_{P_0} P_2 = (P_1 \uplus P_2)/\sim^{\pi_1,\pi_2}_{P_0}$ and $\hat\pi_i : P_i \to P_1 +^{\pi_1,\pi_2}_{P_0} P_2$ with $\hat\pi_i(p) = [(i, p)]_{\sim^{\pi_1,\pi_2}_{P_0}}$ for $p \in P_i$, $1 \le i \le 2$. The corresponding amalgamation $\mu_1 \times_{\pi_1,\pi_2} \mu_2 : P_1 +^{\pi_1,\pi_2}_{P_0} P_2 \to \mathbb{B}$ of two propositional structures $\mu_1 : P_1 \to \mathbb{B}$ and $\mu_2 : P_2 \to \mathbb{B}$ with $\mathit{Str}^{\mathit{Prop}}(\pi_1)(\mu_1) = \pi_1;\ \mu_1 = \mu_0 = \pi_2;\ \mu_2 = \mathit{Str}^{\mathit{Prop}}(\pi_2)(\mu_2)$ for $\mu_0 : P_0 \to \mathbb{B}$ is given by $(\mu_1 \times_{\pi_1,\pi_2} \mu_2)([(i, p)]_{\sim^{\pi_1,\pi_2}_{P_0}}) = \mu_i(p)$.

This construction can be generalised and adapted for the institution $\mathit{FO}^=$ of many-sorted first-order logic with equality in Ex. 1(b) showing that both Prop and $\mathit{FO}^=$ satisfy the amalgamation property [ST12]. □

Lemma 1 (Reducts preserve amalgamations) *Let* $(\Sigma_1 +^{\sigma_1,\sigma_2}_{\Sigma_0} \Sigma_2, (\hat\sigma_i : \Sigma_i \to \Sigma_1 +^{\sigma_1,\sigma_2}_{\Sigma_0} \Sigma_2)_{1 \le i \le 2})$ *be a pushout of* $\sigma_i : \Sigma_0 \to \Sigma_i$ *for* $1 \le i \le 2$ *in* $\mathbb{S}$ *and* $(\Sigma'_1 +^{\sigma'_1,\sigma'_2}_{\Sigma'_0} \Sigma'_2, (\hat\sigma'_i : \Sigma'_i \to \Sigma'_1 +^{\sigma'_1,\sigma'_2}_{\Sigma'_0} \Sigma'_2)_{1 \le i \le 2})$ *a pushout of* $\sigma'_i : \Sigma'_0 \to \Sigma'_i$ *for* $1 \le i \le 2$ *in* $\mathbb{S}$*. Let* $\tau_i : \Sigma_i \to \Sigma'_i$ *for* $0 \le i \le 2$ *be signature morphisms such that the following diagram commutes*

$$\begin{tikzcd}
& \Sigma_1 +^{\sigma_1,\sigma_2}_{\Sigma_0} \Sigma_2 \arrow[rrr, "\tau_1 +_{\tau_0} \tau_2"] & & & \Sigma'_1 +^{\sigma'_1,\sigma'_2}_{\Sigma'_0} \Sigma'_2 & \\
\Sigma_1 \arrow[ur, "\hat\sigma_1"] \arrow[rrr, bend left, "\tau_1"] & & \Sigma_2 \arrow[ul, "\hat\sigma_2"'] \arrow[rrr, bend left, "\tau_2"] & \Sigma'_1 \arrow[ur, "\hat\sigma'_1"] & & \Sigma'_2 \arrow[ul, "\hat\sigma'_2"'] \\
& \Sigma_0 \arrow[ul, "\sigma_1"] \arrow[ur, "\sigma_2"'] \arrow[rrr, "\tau_0"] & & & \Sigma'_0 \arrow[ul, "\sigma'_1"] \arrow[ur, "\sigma'_2"'] &
\end{tikzcd}$$

with $\tau_1 +_{\tau_0} \tau_2$ *the unique signature morphism from* $\Sigma_1 +^{\sigma_1,\sigma_2}_{\Sigma_0} \Sigma_2$ *to* $\Sigma'_1 +^{\sigma'_1,\sigma'_2}_{\Sigma'_0} \Sigma'_2$ *resulting from the pushout property due to* $\sigma_i;\ \tau_i = \tau_0;\ \sigma'_i$ *for* $1 \le i \le 2$ *and* $\sigma_1;\ \tau_1;\ \hat\sigma'_1 = \sigma_2;\ \tau_2;\ \hat\sigma'_2$*. Let* $M'_1 \in |\mathit{Str}(\Sigma'_1)|$ *and* $M'_2 \in |\mathit{Str}(\Sigma'_2)|$ *such that* $\mathit{Str}(\sigma'_1)(M'_1) = \mathit{Str}(\sigma'_2)(M'_2)$*. Then*

$$\mathit{Str}(\tau_1 +_{\tau_0} \tau_2)(M'_1 \times_{\sigma'_1,\sigma'_2} M'_2) = \mathit{Str}(\tau_1)(M'_1) \times_{\sigma_1,\sigma_2} \mathit{Str}(\tau_2)(M'_2)\ .$$

Proof. By the uniqueness of amalgamations it suffices to show for $1 \le i \le 2$ that

$$\mathit{Str}(\hat\sigma_i)(\mathit{Str}(\tau_1 +_{\tau_0} \tau_2)(M'_1 \times_{\sigma'_1,\sigma'_2} M'_2)) = \mathit{Str}(\tau_i)(M'_i)\ .$$

From the commuting diagram we have that

$$\hat{\sigma}_i;\ (\tau_1 +_{\tau_0} \tau_2) = \tau_i;\ \hat{\sigma}'_i$$

and hence by the functorial property of Str

$$Str(\tau_1 +_{\tau_0} \tau_2);\ Str(\hat{\sigma}_i) = Str(\hat{\sigma}'_i);\ Str(\tau_i)\,.$$

Since $M'_1 \times_{\sigma'_1,\sigma'_2} M'_2$ is the amalgamation of M'_1 and M'_2 it holds $Str(\hat{\sigma}'_i)(M'_1 \times_{\sigma'_1,\sigma'_2} M'_2) = M'_i$; thus

$$Str(\hat{\sigma}_i)(Str(\tau_1 +_{\tau_0} \tau_2)(M'_1 \times_{\sigma'_1,\sigma'_2} M'_2)) = Str(\tau_i)(Str(\hat{\sigma}'_i)(M'_1 \times_{\sigma'_1,\sigma'_2} M'_2)) = Str(\tau_i)(M'_i)\,.$$ □

Closure under Boolean connectives

An institution $(\mathbb{S}, Str, \mathrm{Sen}, \models)$ is *closed under boolean connectives* (see [Dia08, Sect. 5.1] and [ST12, Ex. 4.1.41]) if for each $\Sigma \in |\mathbb{S}|$ the following holds:

- there is a sentence $\mathrm{true} \in \mathrm{Sen}(\Sigma)$ with $M \models_\Sigma \mathrm{true}$ for all $M \in |Str(\Sigma)|$;
- for each sentence $\varphi \in \mathrm{Sen}(\Sigma)$ there is a sentence $\neg\varphi \in \mathrm{Sen}(\Sigma)$ with $M \models_\Sigma \neg\varphi$ if, and only if, $M \not\models_\Sigma \varphi$ for all $M \in |Str(\Sigma)|$;
- for all sentences $\varphi_1, \varphi_2 \in \mathrm{Sen}(\Sigma)$ there is a sentence $\varphi_1 \vee \varphi_2 \in \mathrm{Sen}(\Sigma)$ with $M \models_\Sigma \varphi_1 \vee \varphi_2$ if, and only if, $M \models_\Sigma \varphi_1$ or $M \models_\Sigma \varphi_2$ for all $M \in |Str(\Sigma)|$.

In any institution that is closed under boolean connectives all other binary boolean connectives $\wedge$, $\rightarrow$, $\leftrightarrow$, etc. and the constant false can be adequately represented.

Example 3 Institutions $Prop$ and $FO^=$ from Ex. 1 are closed under boolean connectives by definition. □

Institution comorphisms

During system development 'it is sometimes useful to switch from one institution to another. To do this *institution comorphisms* [GR02] are an appropriate tool. They allow to express a kind of embedding of a "poorer" source into a "richer" target logic [MDT09]. Formally, an institution comorphism $\nu = (\nu^{\mathbb{S}}, \nu^{Str}, \nu^{\mathrm{Sen}}) : (\mathbb{S}^{\mathcal{I}}, Str^{\mathcal{I}}, \mathrm{Sen}^{\mathcal{I}}, \models^{\mathcal{I}}) \rightarrow (\mathbb{S}^{\mathcal{I}'}, Str^{\mathcal{I}'}, \mathrm{Sen}^{\mathcal{I}'}, \models^{\mathcal{I}'})$ consists of a functor $\nu^{\mathbb{S}} : \mathbb{S}^{\mathcal{I}} \rightarrow \mathbb{S}^{\mathcal{I}'}$, a natural transformation $\nu^{Str} : (\nu^{\mathbb{S}})^{\mathrm{op}};\ Str^{\mathcal{I}'} \dot{\rightarrow} Str^{\mathcal{I}}$, and a natural transformation $\nu^{\mathrm{Sen}} : \mathrm{Sen}^{\mathcal{I}} \dot{\rightarrow} \nu^{\mathbb{S}};\ \mathrm{Sen}^{\mathcal{I}'}$, such that for all $\Sigma \in\mid \mathbb{S}^{\mathcal{I}} \mid$, $M' \in |Str^{\mathcal{I}'}(\nu^{\mathbb{S}}(\Sigma))|$, and $\varphi \in \mathrm{Sen}^{\mathcal{I}}(\Sigma)$ the following *satisfaction condition* holds:

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

Example 4 There is an institution comorphism ν from $Prop$ to $FO^=$:

The signature functor $\nu^{\mathbb{S}} : \mathbb{S}^{Prop} \rightarrow \mathbb{S}^{FO^=}$ maps a propositional signature P to the (many-sorted) signature $\Sigma_P = (\{\mathrm{Bool}\}, \{\mathrm{tt} : \mathrm{Bool}\} \cup \{p : \mathrm{Bool} \mid p \in P\})$ with a single sort Bool and a constant tt (different from all propositional variables in P) for true as well as constants for the propositional variables, and a propositional signature morphism $\pi : P \rightarrow P'$ to the many-sorted signature morphism $(\pi_S, \pi_F) : \Sigma_P \rightarrow \Sigma_{P'}$ with $\pi_S(\mathrm{Bool}) = \mathrm{Bool}$, $\pi_F(\mathrm{tt}) = \mathrm{tt}$, and $\pi_F(p) = \pi(p)$.

For each $P \in |\mathbb{S}^{Prop}|$, the structures functor $\nu^{Str}_P : Str^{FO^=}(\Sigma_P) \rightarrow Str^{Prop}(P)$ maps a Σ_P-algebra $\mathfrak{A} \in |Str^{FO^=}(\Sigma_P)|$ to $\mu_{\mathfrak{A}} : P \rightarrow \mathbb{B}$ with $\mu_{\mathfrak{A}}(p) = tt$ if, and only if, $p^{\mathfrak{A}} = \mathrm{tt}^{\mathfrak{A}}$, and an algebra homomorphism $h : \mathfrak{A}_1 \rightarrow \mathfrak{A}_2$ to the unique propositional inclusion morphism $\mu_h : \mu_{\mathfrak{A}_1} \rightarrow \mu_{\mathfrak{A}_2}$ as $p^{\mathfrak{A}_1} = \mathrm{tt}^{\mathfrak{A}_1}$ implies $p^{\mathfrak{A}_2} = h(p^{\mathfrak{A}_1}) = h(\mathrm{tt}^{\mathfrak{A}_1}) = \mathrm{tt}^{\mathfrak{A}_2}$. Then $\nu^{Str} : \nu^{\mathbb{S}};\ Str^{FO^=} \dot{\rightarrow} Str^{Prop}$ obviously is a natural transformation.

For each $P \in |\mathbb{S}^{Prop}|$, define the function $(-)^{=}_{P} : \mathrm{Sen}^{Prop}(P) \rightarrow \mathrm{Sen}^{FO^=}(\Sigma_P)$ inductively by

- $(p)^{=}_{P} = (p = \mathrm{tt})$;
- $(\neg\rho)^{=}_{P} = \neg(\rho)^{=}_{P}$;
- $(\rho_1 \vee \rho_2)^{=}_{P} = (\rho_1)^{=}_{P} \vee (\rho_1)^{=}_{P}$;

and similarly for the other connectives. Then $\nu^{\mathrm{Sen}} : \mathrm{Sen}^{Prop}(P) \rightarrow \nu^{\mathbb{S}};\ \mathrm{Sen}^{FO^=}$ with $\nu^{\mathrm{Sen}}(\rho) = (\rho)^{=}_{P}$ obviously is a natural transformation.

Finally, the satisfaction condition

$$\mu_{\mathfrak{A}} \models_{P}^{Prop} \rho \iff \mathfrak{A} \models_{\Sigma_P}^{FO^=} (\rho)_{P}^{=}$$

can be checked by structural induction over ρ. □

Lemma 2 *Let $\nu : \mathcal{I} \to \mathcal{I}'$ be an institution comorphism. Let $(\Sigma_1 +_{\Sigma_0}^{\sigma_1,\sigma_2} \Sigma_2, (\hat{\sigma}_i : \Sigma_i \to \Sigma_1 +_{\Sigma_0}^{\sigma_1,\sigma_2} \Sigma_2)_{1\leq i\leq 2})$ be a pushout of $\sigma_i : \Sigma_0 \to \Sigma_i$ for $1 \leq i \leq 2$ in $\mathbb{S}^{\mathcal{I}}$ admitting amalgamation such that $(\nu^{\mathbb{S}}(\Sigma_1) +_{\nu^{\mathbb{S}}(\Sigma_0)}^{\nu^{\mathbb{S}}(\sigma_1),\nu^{\mathbb{S}}(\sigma_2)} \nu^{\mathbb{S}}(\Sigma_2), (\nu^{\mathbb{S}}(\hat{\sigma}_i) : \nu^{\mathbb{S}}(\Sigma_i) \to \nu^{\mathbb{S}}(\Sigma_1) +_{\nu^{\mathbb{S}}(\Sigma_0)}^{\nu^{\mathbb{S}}(\sigma_1),\nu^{\mathbb{S}}(\sigma_2)} \nu^{\mathbb{S}}(\Sigma_2))_{1\leq i\leq 2})$ is a pushout of $\nu^{\mathbb{S}}(\sigma_i) : \nu^{\mathbb{S}}(\Sigma_0) \to \nu^{\mathbb{S}}(\Sigma_i)$ for $1 \leq i \leq 2$ in $\mathbb{S}^{\mathcal{I}'}$ admitting amalgamation. Let $M_1' \in |Str^{\mathcal{I}'}(\nu^{\mathbb{S}}(\Sigma_1))|$ and $M_2' \in |Str^{\mathcal{I}'}(\nu^{\mathbb{S}}(\Sigma_2))|$ such that $Str^{\mathcal{I}'}(\nu^{\mathbb{S}}(\sigma_1))(M_1') = Str^{\mathcal{I}'}(\nu^{\mathbb{S}}(\sigma_2))(M_2')$. Then*

$$\nu^{Str}_{\Sigma_1 +_{\Sigma_0}^{\sigma_1,\sigma_2} \Sigma_2}(M_1' \times_{\nu^{\mathbb{S}}(\sigma_1),\nu^{\mathbb{S}}(\sigma_2)} M_2') = \nu^{Str}_{\Sigma_1}(M_1') \times_{\sigma_1,\sigma_2} \nu^{Str}_{\Sigma_2}(M_2')\ .$$

Proof. By the uniqueness of amalgamations it suffices to show for $1 \leq i \leq 2$ that

$$Str^{\mathcal{I}}(\hat{\sigma}_i)(\nu^{Str}_{\Sigma_1 +_{\Sigma_0}^{\sigma_1,\sigma_2} \Sigma_2}(M_1' \times_{\nu^{\mathbb{S}}(\sigma_1),\nu^{\mathbb{S}}(\sigma_2)} M_2')) = \nu^{Str}_{\Sigma_i}(M_i')\ .$$

As $\nu^{\mathbb{S}}$ preserves the pushout it follows from the naturality of ν^{Str} that

$$\nu^{Str}_{\Sigma_1 +_{\Sigma_0}^{\sigma_1,\sigma_2} \Sigma_2};\ Str^{\mathcal{I}}(\hat{\sigma}_i) = Str^{\mathcal{I}'}(\nu^{\mathbb{S}}(\hat{\sigma}_i));\ \nu^{Str}_{\Sigma_i}\ .$$

Since $M_1' \times_{\nu^{\mathbb{S}}(\sigma_1),\nu^{\mathbb{S}}(\sigma_2)} M_2'$ is the amalgamation of M_1' and M_2' it holds that $Str^{\mathcal{I}'}(\nu^{\mathbb{S}}(\hat{\sigma}_i))(M_1' \times_{\nu^{\mathbb{S}}(\sigma_1),\nu^{\mathbb{S}}(\sigma_2)} M_2') = M_i'$; thus

$$Str^{\mathcal{I}}(\hat{\sigma}_i)(\nu^{Str}_{\Sigma_1 +_{\Sigma_0}^{\sigma_1,\sigma_2} \Sigma_2}(M_1' \times_{\nu^{\mathbb{S}}(\sigma_1),\nu^{\mathbb{S}}(\sigma_2)} M_2')) = \nu^{Str}_{\Sigma_i}(Str^{\mathcal{I}'}(\nu^{\mathbb{S}}(\hat{\sigma}_i))(M_1' \times_{\nu^{\mathbb{S}}(\sigma_1),\nu^{\mathbb{S}}(\sigma_2)} M_2')) = \nu^{Str}_{\Sigma_i}(M_i')\ .$$ □

3. Institutions for data states and transitions

Data institutions are used to model data states as well as data state changes which are typical in event/data-based systems. We are interested in a generic approach which can be instantiated by concrete data institutions needed in particular application domains.

General assumption. Throughout this paper we assume given an arbitrary institution $\mathcal{D} = (\mathbb{S}^{\mathcal{D}}, Str^{\mathcal{D}}, \mathrm{Sen}^{\mathcal{D}}, \models^{\mathcal{D}})$ which is closed under boolean connectives and satisfies the amalgamation property.

We proceed in two steps: First, in Sect. 3.1, we show how to construct "data state institutions" on the basis of $\mathcal{D}$ to model single data states (in terms of structures) and to specify properties of them (in terms of sentences). To formalise transitions from one data state to another (and their properties) we introduce so-called "2-data state institutions" in Sect. 3.2. Structures of a 2-data state institution are pairs of structures of a data state institution modelling pre- and post-states of transitions.

3.1. Data state institutions

In principle, an institution $\mathcal{D}$ with the assumed properties could already serve as a formal framework for data states. In general, however, it is useful to take not all signatures of $\mathcal{D}$ but to select only particular ones. For instance, one may be interested to admit only signatures which are extensions of some (base) signature having a unique interpretation. To treat this intuitively simple idea in a formal way we need some additional technicalities.

First, the category of signatures of a data state institution over $\mathcal{D}$, which we call $\vec{\mathcal{D}}$, should be a subcategory of the arrow category $(\mathbb{S}^{\mathcal{D}})^{\to}$. This means that signatures in $\vec{\mathcal{D}}$ are signature morphisms $\delta : \Sigma_0 \to \Sigma$ in $\mathcal{D}$ and signature morphisms $(\sigma_0, \sigma) : (\delta : \Sigma_0 \to \Sigma) \to (\delta' : \Sigma_0' \to \Sigma')$ in $\vec{\mathcal{D}}$ are pairs of signature morphisms $\sigma_0 : \Sigma_0 \to \Sigma_0'$ and $\sigma : \Sigma \to \Sigma'$ in $\mathcal{D}$ such that $\sigma_0;\ \delta' = \delta;\ \sigma$. In order to facilitate compositions, we also require that this subcategory of $(\mathbb{S}^{\mathcal{D}})^{\to}$ is closed under pushouts.

Moreover, the form of signatures in a data state institution $\vec{\mathcal{D}}$ has a semantic counterpart concerning the structures functor $\mathit{Str}^{\vec{\mathcal{D}}}$. We require that for each $\vec{\mathcal{D}}$-signature $\delta : \Sigma_0 \to \Sigma$ the (base) signature Σ_0 has a unique interpretation in all structures of $\mathit{Str}^{\vec{\mathcal{D}}}(\delta)$. The sentences and satisfaction relations of $\vec{\mathcal{D}}$ are those inherited from $\mathcal{D}$.

Definition 2 A *data state institution* $\vec{\mathcal{D}} = (\mathbb{S}^{\vec{\mathcal{D}}}, \mathit{Str}^{\vec{\mathcal{D}}}, \mathrm{Sen}^{\vec{\mathcal{D}}}, \models^{[\vec{\mathcal{D}}]})$ over $\mathcal{D}$ consists of the following parts:

- A category $\mathbb{S}^{\vec{\mathcal{D}}}$ of $\vec{\mathcal{D}}$-signatures which is a subcategory of the arrow category $(\mathbb{S}^{\mathcal{D}})^{\to}$ and which is closed under pushouts, i.e., if $(\Sigma_1 +_{\Sigma_0}^{\delta_1,\delta_2} \Sigma_2, (\hat{\delta}_i : \Sigma_i \to \Sigma_1 +_{\Sigma_0}^{\delta_1,\delta_2} \Sigma_2)_{1\leq i\leq 2})$ is a pushout of $\delta_1 : \Sigma_0 \to \Sigma_1$ and $\delta_2 : \Sigma_0 \to \Sigma_2$ in $\mathbb{S}^{\mathcal{D}}$ and $\delta_1, \delta_2 \in |\mathbb{S}^{\vec{\mathcal{D}}}|$, then $\delta_1;\ \hat{\delta}_1 = \delta_2;\ \hat{\delta}_2 \in |\mathbb{S}^{\vec{\mathcal{D}}}|$.
- A functor $\mathit{Str}^{\vec{\mathcal{D}}} : (\mathbb{S}^{\vec{\mathcal{D}}})^{\mathrm{op}} \to \mathrm{Cat}$ which yields
  1. for every $\vec{\mathcal{D}}$-signature $\delta : \Sigma_0 \to \Sigma \in |\mathbb{S}^{\vec{\mathcal{D}}}|$ a non-empty subcategory of $\mathit{Str}^{\mathcal{D}}(\Sigma)$ such that there is an $M_\delta \in |\mathit{Str}^{\mathcal{D}}(\Sigma_0)|$ with $\mathit{Str}^{\mathcal{D}}(\delta)(M) = M_\delta$ for all $M \in |\mathit{Str}^{\vec{\mathcal{D}}}(\delta)|$ and $\mathit{Str}^{\mathcal{D}}(\delta)(\mu) = 1_{M_\delta}$ for all $\mu : M_1 \to M_2$ in $\mathit{Str}^{\vec{\mathcal{D}}}(\delta)$;
  2. for every signature morphism $(\sigma_0, \sigma) : (\delta : \Sigma_0 \to \Sigma) \to (\delta' : \Sigma_0' \to \Sigma')$ in $\mathbb{S}^{\vec{\mathcal{D}}}$ a reduct functor $\mathit{Str}^{\vec{\mathcal{D}}}(\sigma_0, \sigma) : \mathit{Str}^{\vec{\mathcal{D}}}(\delta') \to \mathit{Str}^{\vec{\mathcal{D}}}(\delta)$ such that

     ($*$) for every $M' \in |\mathit{Str}^{\vec{\mathcal{D}}}(\delta')| \subseteq |\mathit{Str}^{\mathcal{D}}(\Sigma')|$ and every $\mu' : M_1' \to M_2'$ in $\mathit{Str}^{\vec{\mathcal{D}}}(\delta')$:
     $\mathit{Str}^{\vec{\mathcal{D}}}(\sigma_0, \sigma)(M') = \mathit{Str}^{\mathcal{D}}(\sigma)(M')$ and $\mathit{Str}^{\vec{\mathcal{D}}}(\sigma_0, \sigma)(\mu') = \mathit{Str}^{\mathcal{D}}(\sigma)(\mu')$.
- The functor $\mathrm{Sen}^{\vec{\mathcal{D}}} : \mathbb{S}^{\vec{\mathcal{D}}} \to \mathrm{Set}$, giving for each signature $\delta : \Sigma_0 \to \Sigma \in |\mathbb{S}^{\vec{\mathcal{D}}1}|$ the set $\mathrm{Sen}^{\mathcal{D}}(\Sigma)$ of Σ-sentences in $\mathcal{D}$, i.e., $\mathrm{Sen}^{\vec{\mathcal{D}}}(\delta) = \mathrm{Sen}^{\mathcal{D}}(\Sigma)$, and for each $\vec{\mathcal{D}}$-signature morphism $(\sigma_0, \sigma) : (\delta : \Sigma_0 \to \Sigma) \to (\delta' : \Sigma_0' \to \Sigma')$ the sentence translation function $\mathrm{Sen}^{\mathcal{D}}(\sigma) : \mathrm{Sen}^{\mathcal{D}}(\Sigma) \to \mathrm{Sen}^{\mathcal{D}}(\Sigma')$ in $\mathcal{D}$, i.e., $\mathrm{Sen}^{\vec{\mathcal{D}}}(\sigma_0, \sigma) : \mathrm{Sen}^{\vec{\mathcal{D}}}(\delta) \to \mathrm{Sen}^{\vec{\mathcal{D}}}(\delta')$ is $\mathrm{Sen}^{\mathcal{D}}(\sigma) : \mathrm{Sen}^{\mathcal{D}}(\Sigma) \to \mathrm{Sen}^{\mathcal{D}}(\Sigma')$.
- For each signature $\delta : \Sigma_0 \to \Sigma \in |\mathbb{S}^{\vec{\mathcal{D}}}|$ the satisfaction relation $\models_\delta^{\vec{\mathcal{D}}} \subseteq |\mathit{Str}^{\vec{\mathcal{D}}}(\delta)| \times \mathrm{Sen}^{\vec{\mathcal{D}}}(\delta)$ is given by the satisfaction relation $\models_\Sigma$ in $\mathcal{D}$, i.e., for each $M \in |\mathit{Str}^{\vec{\mathcal{D}}}(\delta)| \subseteq |\mathit{Str}^{\mathcal{D}}(\Sigma)|$ and $\varphi \in \mathrm{Sen}^{\vec{\mathcal{D}}}(\delta) = \mathrm{Sen}^{\mathcal{D}}(\Sigma)$: $M \models_\delta^{\vec{\mathcal{D}}} \varphi$ if, and only if, $M \models_\Sigma^{\mathcal{D}} \varphi$. □

Remark 1. A data institution $\vec{\mathcal{D}}$ over $\mathcal{D}$ is indeed an institution. The satisfaction condition follows from condition * taking into account that the satisfaction relation in $\vec{\mathcal{D}}$ is inherited from $\mathcal{D}$ and that $\mathcal{D}$ is an institution. More precisely, let $(\sigma_0, \sigma) : (\delta : \Sigma_0 \to \Sigma) \to (\delta' : \Sigma_0' \to \Sigma')$ be a signature morphism in $\mathbb{S}^{\vec{\mathcal{D}}}$ and $M' \in |\mathit{Str}^{\vec{\mathcal{D}}}(\delta')|$, $\varphi \in \mathrm{Sen}^{\vec{\mathcal{D}}}(\delta)$. Then

$$\begin{aligned}
& \mathit{Str}^{\vec{\mathcal{D}}}(\sigma_0, \sigma)(M') \models_\delta^{\vec{\mathcal{D}}} \varphi \\
\Leftrightarrow \quad & \quad \{ \text{ cond. (*), def. } \models^{\vec{\mathcal{D}}} \} \\
& \mathit{Str}^{\mathcal{D}}(\sigma)(M') \models_\Sigma^{\mathcal{D}} \varphi \\
\Leftrightarrow \quad & \quad \{ \text{ sat. cond. for } \mathcal{D} \} \\
& M' \models_{\Sigma'}^{\mathcal{D}} \mathrm{Sen}^{\mathcal{D}}(\sigma)(\varphi) \\
\Leftrightarrow \quad & \quad \{ \text{ def. } \models^{\vec{\mathcal{D}}}, \mathrm{Sen}^{\vec{\mathcal{D}}} \} \\
& M' \models_{\delta'}^{\vec{\mathcal{D}}} \mathrm{Sen}^{\vec{\mathcal{D}}}(\sigma_0, \sigma)(\varphi)
\end{aligned}$$

Moreover, $\vec{\mathcal{D}}$ inherits closure under boolean connective from $\mathcal{D}$. □

Remark 2. A data institution $\vec{\mathcal{D}}$ over $\mathcal{D}$ is uniquely defined once a subcategory $\mathbb{S}^{\vec{\mathcal{D}}}$ of the arrow category $(\mathbb{S}^{\mathcal{D}})^{\to}$ and a functor $\mathit{Str}^{\vec{\mathcal{D}}} : (\mathbb{S}^{\vec{\mathcal{D}}})^{\mathrm{op}} \to \mathrm{Cat}$ satisfying the above conditions are selected. According to condition * there is only one way to define the reduct functor $\mathit{Str}^{\vec{\mathcal{D}}}(\sigma_0, \sigma)$ for signature morphisms $(\sigma_0, \sigma) : (\delta : \Sigma_0 \to \Sigma) \to (\delta' : \Sigma_0' \to \Sigma')$ in $\mathbb{S}^{\vec{\mathcal{D}}}$. In fact $\mathit{Str}^{\vec{\mathcal{D}}}(\sigma_0, \sigma)$ works like the reduct functor $\mathit{Str}^{\mathcal{D}}(\sigma)$ in $\mathcal{D}$. In concrete examples the critical part is to check that for every $M' \in |\mathit{Str}^{\vec{\mathcal{D}}}(\delta')|$ indeed $\mathit{Str}^{\mathcal{D}}(\sigma)(M') \in |\mathit{Str}^{\vec{\mathcal{D}}}(\delta)|$ holds and that for every morphism $\mu' : M_1' \to M_2'$ in $\mathit{Str}^{\vec{\mathcal{D}}}(\delta')$ it holds that $\mathit{Str}^{\mathcal{D}}(\sigma)(\mu')$ is a morphism in $\mathit{Str}^{\vec{\mathcal{D}}}(\delta)$. □

Example 5 (a) The propositional logic institution Prop of Ex. 1(a) is readily usable as data state institution as no specific base propositional signature has to be selected that would need a fixed interpretation. Technically, we can turn Prop into the data state institution $\mathit{Prop}_\emptyset$ as follows: For the category of signatures we choose $\mathbb{S}^{\mathit{Prop}_\emptyset}$ as the subcategory of $(\mathbb{S}^{\mathit{Prop}})^\rightarrow$ consisting of all propositional signature morphisms $\delta_P : \emptyset \rightarrow P$ for $P \in |\mathbb{S}^{\mathit{Prop}}|$ as objects and all $(\emptyset, \pi) : \delta_P \rightarrow \delta_{P'}$ with $\pi : P \rightarrow P'$ in $\mathbb{S}^{\mathit{Prop}}$ as morphisms. In fact, $\mathbb{S}^{\mathit{Prop}_\emptyset}$ is isomorphic to $\mathbb{S}^{\mathit{Prop}}$ and also closed under pushouts. For the structures functor we set $\mathit{Str}^{\mathit{Prop}_\emptyset}(\delta_P) = \mathit{Str}^{\mathit{Prop}}(P)$ and $\mathit{Str}^{\mathit{Prop}_\emptyset}((\emptyset, \pi) : \delta_P \rightarrow \delta_{P'}) = \mathit{Str}^{\mathit{Prop}}(\pi)$ which directly satisfies (*).

(b) A first data state institution $\mathit{Attr}^{\mathit{FO}^=}_{\Sigma_\mathrm{b},\mathfrak{A}_\mathrm{b}}$ over the institution $\mathit{FO}^=$ of many-sorted first-order logic with equality, see Ex. 1(b), considers sorted attributes over a base signature Σ_b and a fixed algebra $\mathfrak{A}_\mathrm{b}$.

We fix a many-sorted base signature $\Sigma_\mathrm{b} \in |\mathbb{S}^{\mathit{FO}^=}|$ providing sorts and function symbols for primitive data types like booleans, integers, etc., and we fix a (standard) interpretation given by a Σ_b-algebra $\mathfrak{A}_\mathrm{b} \in |\mathit{Str}^{\mathit{FO}^=}(\Sigma_\mathrm{b})|$. In particular, we assume that Σ_b contains a sort Bool and constants tt : Bool, ff : Bool such that $\mathfrak{A}_\mathrm{b}$ interprets Bool by $\mathbb{B}$ with $\mathrm{tt}^{\mathfrak{A}_\mathrm{b}} = \mathit{tt}$ and $\mathrm{ff}^{\mathfrak{A}_\mathrm{b}} = \mathit{ff}$. An *attribute* over Σ_b is a constant function symbol, denoted by $a : s$, whose sort s belongs to the sorts of Σ_b. Any set A of attributes over Σ_b defines an *attribute signature* Σ_A which is a many-sorted signature with subsignature Σ_b such that Σ_A and Σ_b have the same sorts and the function symbols of Σ_A extend the function symbols of Σ_b by the attributes A.

For the signature category $\mathbb{S}^{\mathit{Attr}^{\mathit{FO}^=}_{\Sigma_\mathrm{b},\mathfrak{A}_\mathrm{b}}}$ of $\mathit{Attr}^{\mathit{FO}^=}_{\Sigma_\mathrm{b},\mathfrak{A}_\mathrm{b}}$ we take as objects all the inclusion signature morphisms $\iota : \Sigma_\mathrm{b} \hookrightarrow \Sigma_A$ in $\mathit{FO}^=$ where Σ_A is an attribute signature over Σ_b. These inclusion signature morphisms are also closed under pushouts which correspond to disjoint unions. As signature morphisms $(\sigma_0, \sigma) : (\iota : \Sigma_\mathrm{b} \hookrightarrow \Sigma_A) \rightarrow (\iota' : \Sigma_\mathrm{b} \hookrightarrow \Sigma_{A'})$ in $\mathit{Attr}^{\mathit{FO}^=}_{\Sigma_\mathrm{b},\mathfrak{A}_\mathrm{b}}$ we take $\sigma_0 = 1_{\Sigma_\mathrm{b}}$ and all signature morphisms $\sigma : \Sigma_A \rightarrow \Sigma_{A'}$ in $\mathit{FO}^=$ such that the restriction of σ to Σ_b is the identity on Σ_b. Hence there is a one-to-one correspondence between the signature morphisms in $\mathbb{S}^{\mathit{Attr}^{\mathit{FO}^=}_{\Sigma_\mathrm{b},\mathfrak{A}_\mathrm{b}}}$ and the set of sort-preserving mappings from A to A'. Obviously, $1_{\Sigma_\mathrm{b}};\ \iota' = \iota;\ \sigma$ holds and the category $\mathbb{S}^{\mathit{Attr}^{\mathit{FO}^=}_{\Sigma_\mathrm{b},\mathfrak{A}_\mathrm{b}}}$ of signatures in $\mathit{Attr}^{\mathit{FO}^=}_{\Sigma_\mathrm{b},\mathfrak{A}_\mathrm{b}}$ is a subcategory of the arrow category $(\mathbb{S}^{\mathit{FO}^=})^\rightarrow$.

For the structures functor $\mathit{Str}^{\mathit{Attr}^{\mathit{FO}^=}_{\Sigma_\mathrm{b},\mathfrak{A}_\mathrm{b}}}$ we take for each $\iota : \Sigma_\mathrm{b} \hookrightarrow \Sigma_A \in |\mathbb{S}^{\mathit{Attr}^{\mathit{FO}^=}_{\Sigma_\mathrm{b},\mathfrak{A}_\mathrm{b}}}|$ the category whose objects are all Σ_A-algebras $\mathfrak{A} \in |\mathit{Str}^{\mathit{FO}^=}(\Sigma_A)|$ such that $\mathfrak{A}|\iota = \mathfrak{A}_\mathrm{b}$ and whose morphisms $h : \mathfrak{A}_1 \rightarrow \mathfrak{A}_2$ are all Σ_A-algebra homomorphisms such that $h|\iota = 1_{\mathfrak{A}_\mathrm{b}}$. Hence there is a one-to-one correspondence between the class of $\mathit{Attr}^{\mathit{FO}^=}_{\Sigma_\mathrm{b},\mathfrak{A}_\mathrm{b}}$-structures with signature $\iota : \Sigma_\mathrm{b} \hookrightarrow \Sigma_A$ and the class of all valuations mapping attributes $(a : s) \in A$ to values of sort s in the algebra $\mathfrak{A}_\mathrm{b}$.

For each signature morphism $(1_{\Sigma_\mathrm{b}}, \sigma) : (\iota : \Sigma_\mathrm{b} \hookrightarrow \Sigma_A) \rightarrow (\iota' : \Sigma_\mathrm{b} \hookrightarrow \Sigma_{A'})$ in $\mathbb{S}^{\mathit{Attr}^{\mathit{FO}^=}_{\Sigma_\mathrm{b},\mathfrak{A}_\mathrm{b}}}$ the corresponding reduct functor $\mathit{Str}^{\mathit{Attr}^{\mathit{FO}^=}_{\Sigma_\mathrm{b},\mathfrak{A}_\mathrm{b}}}(1_{\Sigma_\mathrm{b}}, \sigma) : \mathit{Str}^{\mathit{Attr}^{\mathit{FO}^=}_{\Sigma_\mathrm{b},\mathfrak{A}_\mathrm{b}}}(\iota') \rightarrow \mathit{Str}^{\mathit{Attr}^{\mathit{FO}^=}_{\Sigma_\mathrm{b},\mathfrak{A}_\mathrm{b}}}(\iota)$ maps every algebra $\mathfrak{A}' \in |\mathit{Str}^{\mathit{Attr}^{\mathit{FO}^=}_{\Sigma_\mathrm{b},\mathfrak{A}_\mathrm{b}}}(\iota')| \subseteq |\mathit{Str}^{\mathit{FO}^=}(\Sigma_{A'})|$ to $\mathfrak{A}'|\sigma \in |\mathit{Str}^{\mathit{FO}^=}(\Sigma_A)|$ and every $h' : \mathfrak{A}'_1 \rightarrow \mathfrak{A}'_2$ in $\mathit{Str}^{\mathit{Attr}^{\mathit{FO}^=}_{\Sigma_\mathrm{b},\mathfrak{A}_\mathrm{b}}}(\iota')$ to $h'|\sigma : \mathfrak{A}'_1|\sigma \rightarrow \mathfrak{A}'_2|\sigma$ in $\mathit{Str}^{\mathit{FO}^=}(\Sigma_A)$. Then $\mathit{Str}^{\mathit{Attr}^{\mathit{FO}^=}_{\Sigma_\mathrm{b},\mathfrak{A}_\mathrm{b}}}(1_{\Sigma_\mathrm{b}}, \sigma)$ is well-defined: By the functorial property of the reduct functor in $\mathit{FO}^=$ and since $1_{\Sigma_\mathrm{b}};\ \iota' = \iota;\ \sigma$, we obtain that $(\mathfrak{A}'|\sigma)|\iota = (\mathfrak{A}'|\iota')|1_{\Sigma_\mathrm{b}} = \mathfrak{A}'|\iota' = \mathfrak{A}_\mathrm{b}$, i.e., $\mathfrak{A}'|\sigma \in |\mathit{Str}^{\mathit{Attr}^{\mathit{FO}^=}_{\Sigma_\mathrm{b},\mathfrak{A}_\mathrm{b}}}(\iota)|$; it follows similarly that $h'|\sigma$ is a morphism in $\mathit{Str}^{\mathit{Attr}^{\mathit{FO}^=}_{\Sigma_\mathrm{b},\mathfrak{A}_\mathrm{b}}}(\iota)$.

For each $\iota : \Sigma_\mathrm{b} \hookrightarrow \Sigma_A$ the sentences in $\mathrm{Sen}^{\mathit{Attr}^{\mathit{FO}^=}_{\Sigma_\mathrm{b},\mathfrak{A}_\mathrm{b}}}(\iota)$ are the first-order Σ_A-sentences in $\mathit{FO}^=$ and the satisfaction relation in $\mathit{Attr}^{\mathit{FO}^=}_{\Sigma_\mathrm{b},\mathfrak{A}_\mathrm{b}}$ is the first-order satisfaction relation.

(c) A second data state institution $\mathit{Attr}^{=}_{\Sigma_\mathrm{b},\mathfrak{A}_\mathrm{b}}$ over $\mathit{FO}^=$ is defined just as $\mathit{Attr}^{\mathit{FO}^=}_{\Sigma_\mathrm{b},\mathfrak{A}_\mathrm{b}}$ but omits quantification in the sentences, i.e., $\mathbb{S}^{\mathit{Attr}^{=}_{\Sigma_\mathrm{b},\mathfrak{A}_\mathrm{b}}} = \mathbb{S}^{\mathit{Attr}^{\mathit{FO}^=}_{\Sigma_\mathrm{b},\mathfrak{A}_\mathrm{b}}}$ and, for each $(\iota : \Sigma_\mathrm{b} \hookrightarrow \Sigma_A)$ in $\mathbb{S}^{\mathit{Attr}^{=}_{\Sigma_\mathrm{b},\mathfrak{A}_\mathrm{b}}}$, the sentences in $\mathrm{Sen}^{\mathit{Attr}^{=}_{\Sigma_\mathrm{b},\mathfrak{A}_\mathrm{b}}}(\iota)$ are the quantifier-free Σ_A-sentences in $\mathit{FO}^=$. The satisfaction relation in $\mathit{Attr}^{=}_{\Sigma_\mathrm{b},\mathfrak{A}_\mathrm{b}}$ is the first-order satisfaction relation but restricted to quantifier-free sentences.

Other variants of attribute-based institutions are also possible, like, for instance, allowing only finite sets of attributes in the signatures. □

Comorphisms between data state institutions

To get an institution comorphism between data state institutions it is sufficient to provide a comorphism between their underlying base institutions if this comorphism satisfies some specific conditions: Let $\vec{\mathcal{D}}$ be a data state institution over $\mathcal{D}$, $\vec{\mathcal{D}}'$ a data state institution over $\mathcal{D}'$, and let $\nu : \mathcal{D} \to \mathcal{D}'$ be an institution comorphism such that

- $\nu^{\mathbb{S}}(\delta) \in |\mathbb{S}^{\vec{\mathcal{D}}'}|$ for all $\delta \in |\mathbb{S}^{\vec{\mathcal{D}}}|$ and $(\nu^{\mathbb{S}}(\sigma_0), \nu^{\mathbb{S}}(\sigma)) : \nu^{\mathbb{S}}(\delta) \to \nu^{\mathbb{S}}(\delta')$ in $\mathbb{S}^{\vec{\mathcal{D}}'}$ for all $(\sigma_0, \sigma) : \delta \to \delta'$ in $\mathbb{S}^{\vec{\mathcal{D}}}$;
- $\nu^{\mathit{Str}}_{\Sigma}(M') \in |\mathit{Str}^{\vec{\mathcal{D}}}(\delta)|$ for all $(\delta : \Sigma_0 \to \Sigma) \in |\mathbb{S}^{\vec{\mathcal{D}}}|$ and $M' \in |\mathit{Str}^{\vec{\mathcal{D}}'}(\nu^{\mathbb{S}}(\delta))|$, and $\nu^{\mathit{Str}}_{\Sigma}(\mu')$ in $\mathit{Str}^{\vec{\mathcal{D}}}(\delta)$ for all $(\delta : \Sigma_0 \to \Sigma) \in |\mathbb{S}^{\vec{\mathcal{D}}}|$ and $\mu' : M'_1 \to M'_2$ in $\mathit{Str}^{\vec{\mathcal{D}}'}(\nu^{\mathbb{S}}(\delta))$.

Define $\vec{\nu}^{\mathbb{S}}(\delta) = \nu^{\mathbb{S}}(\delta)$ and $\vec{\nu}^{\mathbb{S}}(\sigma_0, \sigma) = (\nu^{\mathbb{S}}(\sigma_0), \nu^{\mathbb{S}}(\sigma))$, $\vec{\nu}^{\mathit{Str}}_{\delta:\Sigma_0\to\Sigma} = \nu^{\mathit{Str}}_{\Sigma}$, and $\vec{\nu}^{\mathrm{Sen}}_{\delta:\Sigma_0\to\Sigma} = \nu^{\mathrm{Sen}}_{\Sigma}$. Then $\vec{\nu}^{\mathit{Str}}$ and $\vec{\nu}^{\mathrm{Sen}}$ are natural transformations and for all $(\delta : \Sigma_0 \to \Sigma) \in |\mathbb{S}^{\vec{\mathcal{D}}}|$, $M' \in |\mathit{Str}^{\vec{\mathcal{D}}'}(\vec{\nu}^{\mathbb{S}}(\delta))|$, and $\varphi \in \mathrm{Sen}^{\vec{\mathcal{D}}}(\delta)$ it holds that:

$$\begin{aligned}
& \vec{\nu}^{\mathit{Str}}_{\delta}(M') \models^{\vec{\mathcal{D}}}_{\delta} \varphi \\
\Leftrightarrow \quad & \quad \{ \text{ def. } \vec{\nu}^{\mathit{Str}} \ \} \\
& \nu^{\mathit{Str}}_{\Sigma}(M') \models^{\mathcal{D}}_{\Sigma} \varphi \\
\Leftrightarrow \quad & \quad \{ \text{ sat. cond. } \nu \ \} \\
& M' \models^{\mathcal{D}}_{\nu^{\mathbb{S}}(\Sigma)} \nu^{\mathrm{Sen}}_{\Sigma}(\varphi) \\
\Leftrightarrow \quad & \quad \{ \text{ def. } \models^{\vec{\mathcal{D}}'}, \vec{\nu}^{\mathrm{Sen}} \ \} \\
& M' \models^{\vec{\mathcal{D}}'}_{\vec{\nu}^{\mathbb{S}}(\delta)} \vec{\nu}^{\mathrm{Sen}}_{\delta}(\varphi)
\end{aligned}$$

Thus the institution comorphism $\nu : \mathcal{D} \to \mathcal{D}'$ can be *lifted* to the institution comorphism $\vec{\nu} : \vec{\mathcal{D}} \to \vec{\mathcal{D}}'$.

Example 6 The institution comorphism $\nu : \mathit{Prop} \to \mathit{FO}^{=}$ of Ex. 4 maps a Prop-signature P to $\Sigma_P = (\{\mathrm{Bool}\}, \{\mathrm{tt} : \mathrm{Bool}\} \cup \{p : \mathrm{Bool} \mid p \in P\})$ such that $\Sigma_{\mathrm{b}} \not\hookrightarrow \Sigma_P$ as Σ_{b} (see Ex. 5(b)) shows at least the additional function symbol ff. Thus the first condition on signatures for lifting an institution comorphism to a comorphism between data state institutions is violated. However, we may consider a subtly changed institution comorphism $\nu_{\mathrm{b}} : \mathit{Prop} \to \mathit{FO}^{=}$ that maps P to $\Sigma_{\mathrm{b},P} = \Sigma_{\mathrm{b}} \cup (\{\mathrm{Bool}\}, \{p : \mathrm{Bool} \mid p \in P\})$ but otherwise is defined as the comorphism in Ex. 4. Then an institution comorphism $\vec{\nu}_{\mathrm{b}} : \mathit{Prop}_{\emptyset} \to \mathit{Attr}^{=}_{\Sigma_{\mathrm{b}},\mathfrak{A}_{\mathrm{b}}}$ from the data state institution $\mathit{Prop}_{\emptyset}$ to the data state institution $\mathit{Attr}^{=}_{\Sigma_{\mathrm{b}},\mathfrak{A}_{\mathrm{b}}}$ (see Ex. 5(c)) can be obtained by lifting ν_{b} to the extended data signatures since $(\nu_{\mathrm{b}})^{\mathbb{S}}(\emptyset) = \Sigma_{\mathrm{b}}$ and $(\nu_{\mathrm{b}})^{\mathit{Str}}_{P}(\mathfrak{A}')$ is an object and $(\nu_{\mathrm{b}})^{\mathit{Str}}_{P}(h')$ a morphism in $\mathit{Str}^{\mathit{Prop}_{\emptyset}}(\delta_P : \emptyset \to P) = \mathit{Str}^{\mathit{Prop}}(P)$. □

3.2. 2-Data state institutions

We introduce 2-data state institutions as a formal framework to model data state transitions. The basic idea is that structures of a 2-data state institution are pairs (M_1, M_2) of structures of an underlying data state institution representing pre- and post-states of a transition. The data state institution and the 2-data state institution rely both on the base institution $\mathcal{D}$. Properties of pre-/post-state pairs can be specified by sentences of a 2-data state institution which are built according to a pushout construction in the underlying institution $\mathcal{D}$. The satisfaction relation in a 2-data state institution relies on the construction of an amalgamation of M_1 and M_2 in $\mathcal{D}$. The existence of such pushouts and amalgamations is guaranteed since $\mathcal{D}$ satisfies the amalgamation property.

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

- The category $\mathbb{S}^{2\vec{\mathcal{D}}}$ of $2\vec{\mathcal{D}}$-signatures is the category $\mathbb{S}^{\vec{\mathcal{D}}}$.
- The structures functor $\mathit{Str}^{2\vec{\mathcal{D}}} : (\mathbb{S}^{2\vec{\mathcal{D}}})^{\mathrm{op}} \to \mathrm{Cat}$ maps each $\delta \in |\mathbb{S}^{2\vec{\mathcal{D}}}| = |\mathbb{S}^{\vec{\mathcal{D}}}|$ to the cartesian product of categories $\mathit{Str}^{\vec{\mathcal{D}}}(\delta) \times \mathit{Str}^{\vec{\mathcal{D}}}(\delta)$ and each signature morphism $(\sigma_0, \sigma) : \delta \to \delta'$ in $\mathbb{S}^{2\vec{\mathcal{D}}} = \mathbb{S}^{\vec{\mathcal{D}}}$ to the cartesian product of functors $\mathit{Str}^{\vec{\mathcal{D}}}(\sigma_0, \sigma) \times \mathit{Str}^{\vec{\mathcal{D}}}(\sigma_0, \sigma) : \mathit{Str}^{\vec{\mathcal{D}}}(\delta') \times \mathit{Str}^{\vec{\mathcal{D}}}(\delta') \to \mathit{Str}^{\vec{\mathcal{D}}}(\delta) \times \mathit{Str}^{\vec{\mathcal{D}}}(\delta)$. Hence, for each $\delta : \Sigma_0 \to \Sigma$, structures in $|\mathit{Str}^{2\vec{\mathcal{D}}}(\delta)|$ are pairs of structures $M_1, M_2 \in |\mathit{Str}^{\vec{\mathcal{D}}}(\delta)|$, and reducts of $2\vec{\mathcal{D}}$-structures (M_1, M_2) are computed pairwise.

- The sentence functor $\mathrm{Sen}^{2\vec{\mathcal{D}}} : \mathbb{S}^{2\vec{\mathcal{D}}} \to \mathrm{Set}$ is defined as follows: For each signature $\delta : \Sigma_0 \to \Sigma \in |\mathbb{S}^{2\vec{\mathcal{D}}}| = |\mathbb{S}^{\vec{\mathcal{D}}}|$, we assume given a specifically chosen pushout of δ with itself in $\mathbb{S}^{\mathcal{D}}$, denoted by $(\Sigma +^{\delta}_{\Sigma_0} \Sigma, (\hat{\delta}_i : \Sigma \to \Sigma +^{\delta}_{\Sigma_0} \Sigma)_{1 \leq i \leq 2})$, as illustrated in

$$\begin{tikzcd}
& \Sigma +^{\delta}_{\Sigma_0} \Sigma & \\
\Sigma \arrow[ur, "\hat{\delta}_1"] & & \Sigma \arrow[ul, "\hat{\delta}_2"'] \\
& \Sigma_0 \arrow[ul, "\delta"] \arrow[ur, "\delta"'] \arrow[uu, dashed, "2\delta"] &
\end{tikzcd}$$

Then the set $\mathrm{Sen}^{2\vec{\mathcal{D}}}(\delta)$ of $2\vec{\mathcal{D}}$-*sentences* is $\mathrm{Sen}^{\vec{\mathcal{D}}}(2\delta) = \mathrm{Sen}^{\mathcal{D}}(\Sigma +^{\delta}_{\Sigma_0} \Sigma)$. For each signature morphism $(\sigma_0, \sigma) : (\delta : \Sigma_0 \to \Sigma) \to (\delta' : \Sigma'_0 \to \Sigma')$ in $\mathbb{S}^{2\vec{\mathcal{D}}} = \mathbb{S}^{\vec{\mathcal{D}}}$ the sentence translation function $\mathrm{Sen}^{2\vec{\mathcal{D}}}(\sigma) : \mathrm{Sen}^{2\vec{\mathcal{D}}}(\delta) \to \mathrm{Sen}^{2\vec{\mathcal{D}}}(\delta')$ is $\mathrm{Sen}^{\vec{\mathcal{D}}}(\sigma_0, \sigma +_{\sigma_0} \sigma) = \mathrm{Sen}^{\mathcal{D}}(\sigma +_{\sigma_0} \sigma)$ where $\sigma +_{\sigma_0} \sigma : \Sigma +^{\delta}_{\Sigma_0} \Sigma \to \Sigma' +^{\delta'}_{\Sigma'_0} \Sigma'$ is the unique signature morphism in $\mathbb{S}^{\mathcal{D}}$ such that the following diagram commutes:

$$\begin{tikzcd}
& \Sigma +^{\delta}_{\Sigma_0} \Sigma \arrow[rrr, "\sigma +_{\sigma_0} \sigma"] & & & \Sigma' +^{\delta'}_{\Sigma_0} \Sigma' & \\
\Sigma \arrow[ur, "\hat{\delta}_1"] \arrow[rrr, bend left, "\sigma"] & & \Sigma \arrow[ul, "\hat{\delta}_2"'] \arrow[rrr, bend left, "\sigma"] & \Sigma' \arrow[ur, "\hat{\delta}'_1"] & & \Sigma' \arrow[ul, "\hat{\delta}'_2"'] \\
& \Sigma_0 \arrow[ul, "\delta"] \arrow[ur, "\delta"'] \arrow[uu, dashed, "2\delta"] \arrow[rrr, "\sigma_0"] & & & \Sigma'_0 \arrow[ul, "\delta'"] \arrow[ur, "\delta'"'] \arrow[uu, dashed, "2\delta'"] &
\end{tikzcd}$$

Note that such a signature morphism exists due to the pushout property of $\Sigma +^{\delta}_{\Sigma_0} \Sigma$.

- The satisfaction relations in $2\vec{\mathcal{D}}$ are defined as follows: For each $2\vec{\mathcal{D}}$-signature $\delta : \Sigma_0 \to \Sigma \in |\mathbb{S}^{2\vec{\mathcal{D}}}| = |\mathbb{S}^{\vec{\mathcal{D}}}|$ together with the pushout diagram above, and for any $(M_1, M_2) \in |Str^{2\vec{\mathcal{D}}}(\delta)| = |Str^{\vec{\mathcal{D}}}(\delta)| \times |Str^{\vec{\mathcal{D}}}(\delta)|$ the amalgamation property for $\mathcal{D}$ yields the amalgamation $M_1 \times_\delta M_2 \in |Str^{\mathcal{D}}(\Sigma +^{\delta}_{\Sigma_0} \Sigma)|$ as illustrated in the subsequent diagram, since $Str^{\mathcal{D}}(\delta)(M_1) = M_\delta = Str^{\mathcal{D}}(\delta)(M_2)$ as required for structures in $|Str^{\vec{\mathcal{D}}}(\delta)|$:

$$\begin{tikzcd}
& M_1 \times_\delta M_2 \arrow[dl, mapsto, "Str^{\mathcal{D}}(\hat{\delta}_1)"'] \arrow[dr, mapsto, "Str^{\mathcal{D}}(\hat{\delta}_2)"] & \\
M_1 \arrow[dr, mapsto, "Str^{\mathcal{D}}(\delta)"'] & & M_2 \arrow[dl, mapsto, "Str^{\mathcal{D}}(\delta)"] \\
& M_\delta &
\end{tikzcd}$$

Then, for any $\psi \in \mathrm{Sen}^{2\vec{\mathcal{D}}}(\delta)$, i.e., $\psi \in \mathrm{Sen}^{\mathcal{D}}(\Sigma +^{\delta}_{\Sigma_0} \Sigma)$, the $2\vec{\mathcal{D}}$-satisfaction relation is given by

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

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

To show that $2\vec{\mathcal{D}}$ is indeed an institution we must prove the satisfaction condition.

Theorem 1 (Satisfaction condition of $2\vec{\mathcal{D}}$) *Let* $(\sigma_0, \sigma) : (\delta : \Sigma_0 \to \Sigma) \to (\delta' : \Sigma'_0 \to \Sigma')$ *be a signature morphism in* $\mathbb{S}^{2\vec{\mathcal{D}}}$, $(M'_1, M'_2) \in |Str^{2\vec{\mathcal{D}}}(\delta')|$, *and* $\psi \in \mathrm{Sen}^{2\vec{\mathcal{D}}}(\delta)$. *Then*

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

Proof. Using the satisfaction condition for $\mathcal{D}$, we get

$$\begin{array}{ll}
 & \mathit{Str}^{2\vec{\mathcal{D}}}(\sigma_0, \sigma)(M_1', M_2') \models^{2\vec{\mathcal{D}}}_{\delta} \psi \\
\Leftrightarrow & \quad \{ \text{ def. } \mathit{Str}^{2\vec{\mathcal{D}}} \} \\
 & (\mathit{Str}^{\vec{\mathcal{D}}}(\sigma_0, \sigma)(M_1'), \mathit{Str}^{\vec{\mathcal{D}}}(\sigma_0, \sigma)(M_2')) \models^{2\vec{\mathcal{D}}}_{\delta} \psi \\
\Leftrightarrow & \quad \{ \mathit{Str}^{\vec{\mathcal{D}}}, \text{ def. } \models^{2\vec{\mathcal{D}}} \} \\
 & \mathit{Str}^{\mathcal{D}}(\sigma)(M_1') \times_{\delta} \mathit{Str}^{\mathcal{D}}(\sigma)(M_2') \models^{\mathcal{D}}_{\Sigma +^{\delta}_{\Sigma_0} \Sigma} \psi \\
\Leftrightarrow & \quad \{ \text{ Lem. } 1 \} \\
 & \mathit{Str}^{\mathcal{D}}(\sigma +_{\sigma_0} \sigma)(M_1' \times_{\delta'} M_2') \models^{\mathcal{D}}_{\Sigma +^{\delta}_{\Sigma_0} \Sigma} \psi \\
\Leftrightarrow & \quad \{ \text{ sat. cond. for } \mathcal{D} \} \\
 & M_1' \times_{\delta'} M_2' \models^{\mathcal{D}}_{\Sigma' +^{\delta'}_{\Sigma_0'} \Sigma'} \mathrm{Sen}^{\mathcal{D}}(\sigma +_{\sigma_0} \sigma)(\psi) \\
\Leftrightarrow & \quad \{ \text{ def. } \mathrm{Sen}^{2\vec{\mathcal{D}}} \} \\
 & M_1' \times_{\delta'} M_2' \models^{\mathcal{D}}_{\Sigma' +^{\delta'}_{\Sigma_0'} \Sigma'} \mathrm{Sen}^{2\vec{\mathcal{D}}}(\sigma_0, \sigma)(\psi) \\
\Leftrightarrow & \quad \{ \text{ def. } \models^{2\vec{\mathcal{D}}} \} \\
 & (M_1', M_2') \models^{2\vec{\mathcal{D}}}_{\delta'} \mathrm{Sen}^{2\vec{\mathcal{D}}}(\sigma_0, \sigma)(\psi)
\end{array}$$

□

Again, as $\vec{\mathcal{D}}$ does from $\mathcal{D}$, $2\vec{\mathcal{D}}$ inherits closure under boolean connective from $\vec{\mathcal{D}}$.

Example 7 (a) For obtaining the propositional 2-data state institution $2\mathit{Prop}_{\emptyset}$ from the data state institution $\mathit{Prop}_{\emptyset}$ of Ex. 5(a) we choose the specific pushout signature $P +^{\delta_P}_{\emptyset} P$ in $\mathbb{S}^{\mathit{Prop}}$ for a signature morphism $\delta_P : \emptyset \to P$ in $\mathbb{S}^{\mathit{Prop}_{\emptyset}}$ to contain all propositional variables from P together with primed copies p' of them (assuming that P does not already contain primed propositional variables). δ_P-sentences in $2\mathit{Prop}_{\emptyset}$, also called *state transition predicates*, are then $(P +^{\delta_P}_{\emptyset} P)$-sentences in Prop, like, e.g., $p' \leftrightarrow p$ or $p \vee q \to q'$. The amalgamated union $\mu_1 \times_{\delta_P} \mu_2$ for a $2\mathit{Prop}_{\emptyset}$-structure (μ_1, μ_2) consisting of two functions $\mu_i : P \to \mathbb{B}$ interprets all propositional variables $p \in P$ using μ_1 and all primed propositional variables p' by μ_2 for the non-primed propositional variables p.

(b) Similarly to $2\mathit{Prop}_{\emptyset}$, the construction of a 2-data state institution $2\mathit{Attr}^{=}_{\Sigma_b, \mathfrak{A}_b}$ of equational attributes from $\mathit{Attr}^{=}_{\Sigma_b, \mathfrak{A}_b}$ defined in Ex. 5(c) can choose as the pushout signature $\Sigma_A +^{\iota}_{\Sigma_b} \Sigma_A$ for $\iota : \Sigma_b \hookrightarrow \Sigma_A$ the $FO^{=}$-signature that contains all symbols of Σ_A together with primed copies $a' : s$ of all attributes $a : s$ in A; possible state transition predicates here then are $a' < a$ or $a' = a + 1$. The amalgamated union $\mathfrak{A}_1 \times_{\iota} \mathfrak{A}_2$ for a $2\mathit{Attr}^{=}_{\Sigma_b, \mathfrak{A}_b}$-structure $(\mathfrak{A}_1, \mathfrak{A}_2)$ consisting of two Σ_A-algebras such that $\mathfrak{A}_1|\iota = \mathfrak{A}_b = \mathfrak{A}_2|\iota$ interprets all symbols $a : s$ in the signature Σ_A like $\mathfrak{A}_1$ does and all primed attributes $a' : s$ by the values of the non-primed attributes $a : s$ in $\mathfrak{A}_2$. Note that $\mathfrak{A}_1$ and $\mathfrak{A}_2$ have the same interpretation for all symbols in the base signature Σ_b which therefore cannot be changed in data state transitions. □

Comorphisms between 2-data state institutions

We show that under certain conditions institution comorphisms between 2-data state institutions can be obtained from comorphisms between their underlying base institutions. Let $\nu : \mathcal{D} \to \mathcal{D}'$ be an institution comorphism that can be lifted to an institution comorphism $\vec{\nu} : \vec{\mathcal{D}} \to \vec{\mathcal{D}}'$ such that $\nu^{\mathbb{S}}$ preserves the specific choices of pushouts in $\mathbb{S}^{\mathcal{D}}$ and $\mathbb{S}^{\mathcal{D}'}$ made for the construction of $2\vec{\mathcal{D}}$ and $2\vec{\mathcal{D}}'$. Define $(\vec{\nu}^2)^{\mathbb{S}} = \vec{\nu}^{\mathbb{S}}$, $(\vec{\nu}^2)^{\mathit{Str}} = \vec{\nu}^{\mathit{Str}} \times \vec{\nu}^{\mathit{Str}}$, and $(\vec{\nu}^2)^{\mathrm{Sen}}_{\delta : \Sigma_0 \to \Sigma} = \vec{\nu}^{\mathrm{Sen}}_{\Sigma +^{\delta}_{\Sigma_0} \Sigma}$. Then $(\vec{\nu}^2)^{\mathit{Str}}$ and $(\vec{\nu}^2)^{\mathrm{Sen}}$ are natural transformations and for all $(\delta : \Sigma_0 \to \Sigma) \in |\mathbb{S}^{2\vec{\mathcal{D}}}|$, $(M_1', M_2') \in |\mathit{Str}^{2\vec{\mathcal{D}}'}((\vec{\nu}^2)^{\mathbb{S}}(\delta))|$, and $\psi \in \mathrm{Sen}^{2\vec{\mathcal{D}}}(\delta)$ it holds that

$$\begin{array}{ll}
 & (\vec{\nu}^2)^{\mathit{Str}}_{\delta}(M_1', M_2') \models^{2\vec{\mathcal{D}}}_{\delta} \psi \\
\Leftrightarrow & \quad \{ \text{ def. } (\vec{\nu}^2)^{\mathit{Str}} \} \\
 & (\vec{\nu}^{\mathit{Str}}_{\delta}(M_1'), \vec{\nu}^{\mathit{Str}}_{\delta}(M_2')) \models^{2\vec{\mathcal{D}}}_{\delta} \psi
\end{array}$$

$$\Leftrightarrow \qquad \{\ \vec{\nu}^{Str}, \text{def. } \models^{2\vec{\mathcal{D}}}\ \}$$

$$\nu^{Str}_{\Sigma}(M'_1) \times_{\delta} \nu^{Str}_{\Sigma}(M'_2) \models^{\mathcal{D}}_{\Sigma +^{\delta}_{\Sigma_0} \Sigma} \psi$$

$$\Leftrightarrow \qquad \{\ \text{Lem. } 2\ \}$$

$$\nu^{Str}_{\Sigma +^{\delta}_{\Sigma_0} \Sigma}(M'_1 \times_{\nu^{\mathbb{S}}(\delta)} M'_2) \models^{\mathcal{D}}_{\Sigma +^{\delta}_{\Sigma_0} \Sigma} \psi$$

$$\Leftrightarrow \qquad \{\ \text{sat. cond. } \nu\ \}$$

$$M'_1 \times_{\nu^{\mathbb{S}}(\delta)} M'_2 \models^{\mathcal{D}'}_{\Sigma +^{\delta}_{\Sigma_0} + \Sigma} \nu^{\mathrm{Sen}}_{\Sigma +^{\delta}_{\Sigma_0} \Sigma}(\psi)$$

$$\Leftrightarrow \qquad \{\ \text{def. } (\vec{\nu}^2)^{\mathrm{Sen}}\ \}$$

$$M'_1 \times_{\nu^{\mathbb{S}}(\delta)} M'_2 \models^{\mathcal{D}'}_{\Sigma +^{\delta}_{\Sigma_0} + \Sigma} (\vec{\nu}^2)^{\mathrm{Sen}}_{\delta}(\psi)$$

$$\Leftrightarrow \qquad \{\ \text{def. } \models^{2\vec{\mathcal{D}}'}\ \}$$

$$(M'_1, M'_2) \models^{2\vec{\mathcal{D}}'}_{(\vec{\nu}^2)^{\mathbb{S}}(\delta)} (\vec{\nu}^2)^{\mathrm{Sen}}_{\delta}(\psi)\ .$$

Thus the institution comorphism $\vec{\nu} : \vec{\mathcal{D}} \to \vec{\mathcal{D}}'$ obtained from $\nu : \mathcal{D} \to \mathcal{D}'$ can be further *lifted* to the institution comorphism $\vec{\nu}^2 : 2\vec{\mathcal{D}} \to 2\vec{\mathcal{D}}'$.

Example 8 The institution comorphism $\vec{\nu}_{\mathrm{b}}$ of Ex. 6 obtained as a lifting from the institution comorphism ν_{b} can be further lifted to an institution comorphism $\vec{\nu}^2_{\mathrm{b}} : 2Prop_{\emptyset} \to 2Attr^{\equiv}_{\Sigma_{\mathrm{b}},\mathfrak{A}_{\mathrm{b}}}$ since, using the notation of Ex. 6, ν_{b} maps $P +^{\delta_P}_{\emptyset} P$ to $\Sigma_{\mathrm{b},P} +^{\iota_P}_{\Sigma_{\mathrm{b}}} + \Sigma_{\mathrm{b},P}$ with $\iota_P : \Sigma_{\mathrm{b}} \hookrightarrow \Sigma_{\mathrm{b},P}$. □

Particular sentences of 2-data state institutions

We finish this section by considering some particular sentences of 2-data state institutions. First, the set of sentences of the 2-institution $2\vec{\mathcal{D}}$ contains translations of the sentences of $\mathcal{D}$ that refer either to the first or to the second component of a $2\vec{\mathcal{D}}$-structure (M_1, M_2). Such sentence translations are satisfied by (M_1, M_2) if, and only if, their untranslated versions are satisfied by the respective component.

Proposition 1 For each $2\vec{\mathcal{D}}$-signature $\delta : \Sigma_0 \to \Sigma$, sentence $\varphi \in \mathrm{Sen}^{\mathcal{D}}(\Sigma)$ and $i \in \{1, 2\}$ it holds that

$$(M_1, M_2) \models^{2\vec{\mathcal{D}}}_{\delta} \mathrm{Sen}^{\mathcal{D}}(\hat{\delta}_i)(\varphi) \iff M_i \models^{\mathcal{D}}_{\Sigma} \varphi\ .$$

Proof. Expanding the satisfaction relation for $2\vec{\mathcal{D}}$ and using the satisfaction condition in $\mathcal{D}$ we have

$$(M_1, M_2) \models^{2\vec{\mathcal{D}}}_{\delta} \mathrm{Sen}^{\mathcal{D}}(\hat{\delta}_i)(\varphi)$$

$$\Leftrightarrow \qquad \{\ \text{def. } \models^{2\vec{\mathcal{D}}}\ \}$$

$$M_1 \times_{\delta} M_2 \models^{\mathcal{D}}_{\Sigma +^{\delta}_{\Sigma_0} \Sigma} \mathrm{Sen}^{\mathcal{D}}(\hat{\delta}_i)(\varphi)$$

$$\Leftrightarrow \qquad \{\ \text{sat. cond. for } \mathcal{D}\ \}$$

$$Str^{\mathcal{D}}(\hat{\delta}_i)(M_1 \times_{\delta} M_2) \models^{\mathcal{D}}_{\Sigma} \varphi$$

$$\Leftrightarrow \qquad \{\ \text{def. amalg. prop. for } \mathcal{D}\ \}$$

$$M_i \models^{\mathcal{D}}_{\Sigma} \varphi$$

□

Example 9 Let $2Attr^{\equiv}_{\Sigma_{\mathrm{b}},\mathfrak{A}_{\mathrm{b}}}$ be the 2-data state institution over the attribute data state institution $Attr^{\equiv}_{\Sigma_{\mathrm{b}},\mathfrak{A}_{\mathrm{b}}}$, let $\iota : \Sigma_{\mathrm{b}} \hookrightarrow \Sigma_A$ be a signature of $2Attr^{\equiv}_{\Sigma_{\mathrm{b}},\mathfrak{A}_{\mathrm{b}}}$ and $\varphi \in \mathrm{Sen}^{FO^=}(\Sigma_A)$. Then $\mathrm{Sen}^{FO^=}(\hat{\iota}_1)(\varphi)$, $\mathrm{Sen}^{FO^=}(\hat{\iota}_2)(\varphi)$, and $\mathrm{Sen}^{FO^=}(\hat{\iota}_1)(\varphi) \leftrightarrow \mathrm{Sen}^{FO^=}(\hat{\iota}_2)(\varphi)$ are sentences in $\mathrm{Sen}^{2Attr^{\equiv}_{\Sigma_{\mathrm{b}},\mathfrak{A}_{\mathrm{b}}}}(\iota)$. For each $(\mathfrak{A}_1, \mathfrak{A}_2) \in |Str^{2Attr^{\equiv}_{\Sigma_{\mathrm{b}},\mathfrak{A}_{\mathrm{b}}}}(\iota)|$ we obtain:

$$(\mathfrak{A}_1, \mathfrak{A}_2) \models^{2Attr^{\equiv}_{\Sigma_{\mathrm{b}},\mathfrak{A}_{\mathrm{b}}}}_{\iota} (\mathrm{Sen}^{FO^=}(\hat{\iota}_1)(\varphi) \leftrightarrow \mathrm{Sen}^{FO^=}(\hat{\iota}_2)(\varphi))$$

$$\Leftrightarrow \qquad \{\ \text{def. } \leftrightarrow\ \}$$

$$(\mathfrak{A}_1, \mathfrak{A}_2) \models^{2Attr^{\equiv}_{\Sigma_{\mathrm{b}},\mathfrak{A}_{\mathrm{b}}}}_{\iota} \mathrm{Sen}^{FO^=}(\hat{\iota}_1)(\varphi) \quad \text{iff} \quad (\mathfrak{A}_1, \mathfrak{A}_2) \models^{2Attr^{\equiv}_{\Sigma_{\mathrm{b}},\mathfrak{A}_{\mathrm{b}}}}_{\iota} \mathrm{Sen}^{FO^=}(\hat{\iota}_2)(\varphi)$$

$$\Leftrightarrow \qquad \{\ \text{Prop. } 1\ \}$$

$$\mathfrak{A}_1 \models^{FO^=}_{\Sigma_A} \varphi \quad \text{iff} \quad \mathfrak{A}_2 \models^{FO^=}_{\Sigma_A} \varphi\ .$$

Hence, validity of the sentence $\mathrm{Sen}^{FO^=}(\hat{\iota}_1)(\varphi) \leftrightarrow \mathrm{Sen}^{FO^=}(\hat{\iota}_2)(\varphi)$ in a $2Attr^{=}_{\Sigma_\mathrm{b},\mathfrak{A}_\mathrm{b}}$-structure $(\mathfrak{A}_1, \mathfrak{A}_2)$ expresses that the sentence φ holds in $\mathfrak{A}_1$ if, and only if, it holds in $\mathfrak{A}_2$. □

Sentences over a signature $\delta : \Sigma_0 \to \Sigma$ in the $2\vec{\mathcal{D}}$-institution take into account two $\vec{\mathcal{D}}$-structures M_1 and M_2 at the same time. Semantically it is already ensured that the common part $Str^{\mathcal{D}}(\delta)(M_1) = Str^{\mathcal{D}}(\delta)(M_2)$ remains unchanged. It will, however, be useful to be able to express that certain additional parts of M_1 and M_2 are identical. In particular, this will be needed when we define the (syntactic) parallel composition of operational specifications in Sect. 5.2.

Definition 3 (Invariance sentence) Let $\delta_0 : \Sigma_0 \to \Sigma_1$ be a $\vec{\mathcal{D}}$-signature and $\sigma_1 : \Sigma_1 \to \Sigma$ a $\mathcal{D}$-signature morphism such that $\delta = \delta_0;\ \sigma_1$ is a $\vec{\mathcal{D}}$-signature. A sentence $\psi \in \mathrm{Sen}^{2\vec{\mathcal{D}}}(\delta)$ is an *invariance sentence* w.r.t. (δ_0, σ_1) if for all $(M_1, M_2) \in |Str^{2\vec{\mathcal{D}}}(\delta)|$, $(M_1, M_2) \models^{2\vec{\mathcal{D}}}_{\delta} \psi$ if, and only if, $Str^{\mathcal{D}}(\sigma_1)(M_1) = Str^{\mathcal{D}}(\sigma_1)(M_2)$. □

Example 10 Let $\iota : \Sigma_\mathrm{b} \hookrightarrow \Sigma_A$ be a signature in $2Attr^{=}_{\Sigma_\mathrm{b},\mathfrak{A}_\mathrm{b}}$ and $A_\mathrm{f} \subseteq A$ be a finite subset of attributes. Then the sentence $\bigwedge_{a \in A_\mathrm{f}} a' = a$ in $2Attr^{=}_{\Sigma_\mathrm{b},\mathfrak{A}_\mathrm{b}}$ is an invariance sentence w.r.t. $(\iota_0 : \Sigma_\mathrm{b} \hookrightarrow \Sigma_{A_\mathrm{f}}, \iota_1 : \Sigma_{A_\mathrm{f}} \hookrightarrow \Sigma_A)$. □

In general, we cannot expect that $\mathrm{Sen}^{2\vec{\mathcal{D}}}$ provides invariance sentences. In this case we extend, for each $2\vec{\mathcal{D}}$ signature $\delta : \Sigma_0 \to \Sigma$ and for each split $\delta_0 : \Sigma_0 \to \Sigma_1$ and $\sigma_1 : \Sigma_1 \to \Sigma$ with $\delta = \delta_0;\ \sigma_1$ the set $\mathrm{Sen}^{2\vec{\mathcal{D}}}(\delta)$ of $2\vec{\mathcal{D}}$-sentences by the special invariance sentence $\mathrm{id}_{(\delta_0,\sigma_1)}$ for which satisfaction is defined as follows:

$$(M_1, M_2) \models^{2\vec{\mathcal{D}}}_{\delta} \mathrm{id}_{(\delta_0,\sigma_1)} \iff Str^{\mathcal{D}}(\sigma_1)(M_1) = Str^{\mathcal{D}}(\sigma_1)(M_2)\ .$$

Note that $\mathrm{id}_{(\mathrm{id}_{\Sigma_0},\delta)}$ simply expresses true.

Using a general, though "intuitively somewhat artificial way of dealing with the translation of sentences" [ST12, p. 184], an institution extending $2\vec{\mathcal{D}}$ by such invariance sentences can be obtained directly [ST12, Ex. 4.1.46]. However, if certain additional pushouts are chosen specifically, invariance sentences can be translated along signature morphisms in $2\vec{\mathcal{D}}$ as follows: Let $(\sigma_0, \sigma) : (\delta : \Sigma_0 \to \Sigma) \to (\delta' : \Sigma'_0 \to \Sigma')$ be a signature morphism in $2\vec{\mathcal{D}}$, i.e., a pair of signature morphisms $\sigma_0 : \Sigma_0 \to \Sigma'_0$ and $\sigma : \Sigma \to \Sigma'$ in $\mathcal{D}$ such that $\sigma_0;\ \delta' = \delta;\ \sigma$. Then the invariance sentence $\mathrm{id}_{(\delta_0,\sigma_1)} \in \mathrm{Sen}^{2\vec{\mathcal{D}}}(\delta)$ is translated to the invariance sentence $\mathrm{id}_{(\hat{\sigma}_0,\sigma'_1)} \in \mathrm{Sen}^{2\vec{\mathcal{D}}}(\delta')$ according to the following diagram where the left quadrilateral is a pushout diagram and σ'_1 is the unique morphism from the pushout signature to Σ':

$$\begin{tikzcd}
\Sigma_0 \arrow[rr, "\delta"] \arrow[dr, "\delta_0"] \arrow[ddd, "\sigma_0"'] & & \Sigma \arrow[ddd, "\sigma"] \\
& \Sigma_1 \arrow[ur, "\sigma_1"'] \arrow[d, "\hat{\delta}_0"'] & \\
& \Sigma'_0 +^{\sigma_0,\delta_0}_{\Sigma_0} \Sigma_1 \arrow[dr, "\sigma'_1"] & \\
\Sigma'_0 \arrow[ur, "\hat{\sigma}_0"] \arrow[rr, "\delta'"'] & & \Sigma'
\end{tikzcd}$$

It is in order to obtain a proper mapping $\mathrm{id}_{(\delta_0,\sigma_1)} \mapsto \mathrm{id}_{(\hat{\sigma}_0,\sigma'_1)}$ that the pushout $\Sigma'_0 +^{\sigma_0,\delta_0}_{\Sigma_0} \Sigma_1$ has to be fixed. Then the next lemma shows that the satisfaction condition of $2\vec{\mathcal{D}}$ still holds if it is extended by invariance sentences; so it remains an institution.

Lemma 3 (Satisfaction condition for invariance sentences) *Let* $(\sigma_0, \sigma) : (\delta : \Sigma_0 \to \Sigma) \to (\delta' : \Sigma'_0 \to \Sigma')$ *be a signature morphism in* $\mathbb{S}^{2\vec{\mathcal{D}}}$, $(M'_1, M'_2) \in |Str^{2\vec{\mathcal{D}}}(\delta')|$, *and* $\mathrm{id}_{(\delta_0,\sigma_1)} \in \mathrm{Sen}^{2\vec{\mathcal{D}}}(\delta)$ *be an invariance sentence w.r.t.* (δ_0, σ_1). *Then*

$$(M'_1, M'_2) \models^{2\vec{\mathcal{D}}}_{\delta'} \mathrm{Sen}^{2\vec{\mathcal{D}}}(\sigma_0, \sigma)(\mathrm{id}_{(\delta_0,\sigma_1)}) \iff Str^{2\vec{\mathcal{D}}}(\sigma_0, \sigma)(M'_1, M'_2) \models^{2\vec{\mathcal{D}}}_{\delta} \mathrm{id}_{(\delta_0,\sigma_1)}\ .$$

Proof. As illustrated in the following diagram, let $M_i = Str^{\mathcal{D}}(\sigma)(M'_i)$, $M'_i| = Str^{\mathcal{D}}(\sigma'_1)(M'_i)$, $M_i| = Str^{\mathcal{D}}(\sigma_1)(M_i)$ for $1 \le i \le 2$, $Str^{\mathcal{D}}(\delta')(M'_1) = M'_0 = Str^{\mathcal{D}}(\delta')(M'_2)$, and $Str^{\mathcal{D}}(\delta)(M_1) = M_0 = Str^{\mathcal{D}}(\delta)(M_2)$ such that in particular

$Str^{\mathcal{D}}(\sigma_0)(M_0') = M_0$:

$$\begin{tikzcd}
M_0 & & M_1, M_2 \arrow[ll, mapsto, "Str^{\mathcal{D}}(\delta)"'] \arrow[dl, mapsto, "Str^{\mathcal{D}}(\sigma_1)"'] \\
& M_1|, M_2| \arrow[ul, mapsto, "Str^{\mathcal{D}}(\delta_0)"'] & \\
& M_1'|, M_2'| \arrow[u, mapsto, "Str^{\mathcal{D}}(\hat{\delta}_0)"] \arrow[dl, mapsto, "Str^{\mathcal{D}}(\hat{\sigma}_0)"'] & \\
M_0' \arrow[uuu, mapsto, "Str^{\mathcal{D}}(\sigma_0)"] & & M_1', M_2' \arrow[ll, mapsto, "Str^{\mathcal{D}}(\delta')"] \arrow[ul, mapsto, "Str^{\mathcal{D}}(\sigma_1')"'] \arrow[uuu, mapsto, "Str^{\mathcal{D}}(\sigma)"']
\end{tikzcd}$$

Let first $(M_1', M_2') \models^{2\vec{\mathcal{D}}}_{\delta'} \mathrm{Sen}^{2\vec{\mathcal{D}}}(\sigma_0, \sigma)(\mathrm{id}_{(\delta_0,\sigma_1)})$ hold. Then $M_1'| = M_2'|$, i.e., $Str^{\mathcal{D}}(\sigma_1')(M_1') = Str^{\mathcal{D}}(\sigma_1')(M_2')$. Thus $Str^{\mathcal{D}}(\hat{\delta}_0)(Str^{\mathcal{D}}(\sigma_1')(M_1')) = Str^{\mathcal{D}}(\hat{\delta}_0)(Str^{\mathcal{D}}(\sigma_1')(M_2'))$ and, using $\sigma_1;\ \sigma = \hat{\delta}_0;\ \sigma_1'$, we hence obtain that also $Str^{\mathcal{D}}(\sigma_1)(Str^{\mathcal{D}}(\sigma)(M_1')) = Str^{\mathcal{D}}(\sigma_1)(Str^{\mathcal{D}}(\sigma)(M_2'))$, i.e., $M_1| = M_2|$.

Conversely, let $Str^{2\vec{\mathcal{D}}}(\sigma_0, \sigma)(M_1', M_2') \models^{2\vec{\mathcal{D}}}_{\delta} \mathrm{id}_{(\delta_0,\sigma_1)}$ hold. Then $M_1| = M_2|$, i.e., $Str^{\mathcal{D}}(\sigma_1)(Str^{\mathcal{D}}(\sigma)(M_1')) = Str^{\mathcal{D}}(\sigma_1)(Str^{\mathcal{D}}(\sigma)(M_2'))$. We have to prove that $M_1'| = M_2'|$, i.e., $Str^{\mathcal{D}}(\sigma_1')(M_1') = Str^{\mathcal{D}}(\sigma_1')(M_2')$. By $\sigma_1;\ \sigma = \hat{\delta}_0;\ \sigma_1'$ we have $Str^{\vec{\mathcal{D}}}(\hat{\delta}_0)(M_1'|) = Str^{\vec{\mathcal{D}}}(\hat{\delta}_0)(M_2'|)$, and from $Str^{\vec{\mathcal{D}}}(\delta')(M_1') = M_0' = Str^{\vec{\mathcal{D}}}(\delta')(M_2')$ and $\delta' = \hat{\sigma}_0;\ \sigma_1'$ it follows that $Str^{\vec{\mathcal{D}}}(\hat{\sigma}_0)(M_1'|) = M_0' = Str^{\vec{\mathcal{D}}}(\hat{\sigma}_0)(M_2'|)$. Thus by the amalgamation property we obtain $M_1'| = Str^{\vec{\mathcal{D}}}(\hat{\sigma}_0)(M_1'|) \times_{\sigma_0,\delta_0} Str^{\vec{\mathcal{D}}}(\hat{\delta}_0)(M_1'|) = Str^{\vec{\mathcal{D}}}(\hat{\sigma}_0)(M_2'|) \times_{\sigma_0,\delta_0} Str^{\vec{\mathcal{D}}}(\hat{\delta}_0)(M_2'|) = M_2'|$. □

4. Generic $\mathcal{E}^{\downarrow}(\vec{\mathcal{D}})$-logic

We present $\mathcal{E}^{\downarrow}(\vec{\mathcal{D}})$-logic which forms an event/data institution for any underlying data state institution $\vec{\mathcal{D}}$. Hence $\mathcal{E}^{\downarrow}(\vec{\mathcal{D}})$-logic is a generic logic parametrised by $\vec{\mathcal{D}}$. It is an extension of dynamic logic with binders [MBHM18] taking into account data. Like its data-less predecessor in [MBHM18], $\mathcal{E}^{\downarrow}(\vec{\mathcal{D}})$ is intended to support the whole development process for event/data-based systems from abstract requirements specifications down to concrete designs specifying the (recursive) structure of processes. For the former we use constructs from dynamic logic [HKT00] which allow us to integrate complex actions into modal formulæ with diamond and box operators. For the latter we use constructs from hybrid logic [Bra10] which allow us to refer (and thus to jump) to computation states of transition systems by means of state variables.

We assume given an arbitrary data state institution $\vec{\mathcal{D}}$ as considered in Sect. 3.1 and the 2-data state institution $2\vec{\mathcal{D}}$ as constructed in Sect. 3.2. We illustrate $\mathcal{E}^{\downarrow}(\vec{\mathcal{D}})$-logic by specifying properties of an ATM which will be our running example further on. In the examples of the section we use $Prop_{\emptyset}$ (see Ex. 5(a)) as data state institution, i.e., we work in $\mathcal{E}^{\downarrow}(Prop_{\emptyset})$. In further refinement steps later on we will switch to the more expressive data state institution $Attr^{\equiv}_{\Sigma_{\mathfrak{b}},\mathfrak{A}_{\mathfrak{b}}}$.

4.1. Signatures of $\mathcal{E}^{\downarrow}(\vec{\mathcal{D}})$

A crucial ingredient of any event/data-based system are the events which may occur at a certain instance of time and may change the computation state as well as the data state of a system. In $\mathcal{E}^{\downarrow}(\vec{\mathcal{D}})$-logic signatures consist of an event part, determined by a set of events, and a data part, determined by a signature of $\vec{\mathcal{D}}$.

Definition 4 An *event/data signature* (*ed signature* for short) $\Sigma = (E, \delta : \Delta_0 \to \Delta)$ consists of a set of *events* E and a data signature $\delta : \Delta_0 \to \Delta$ in $|\mathbb{S}^{\vec{\mathcal{D}}}|$. We write $E(\Sigma)$ for E and $\delta(\Sigma)$ for δ.

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

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

Example 11 We consider a (rather simplified) ATM application. We start with its exposition in $\mathcal{E}^{\downarrow}(\mathit{Prop}_{\emptyset})$ based on the propositional logic institution Prop and its corresponding data state institution $\mathit{Prop}_{\emptyset}$; see Ex. 5(a).

A first relevant set of events for our ATM is $E_0 = \{\mathsf{insertCard}, \mathsf{enterPIN}, \mathsf{ejectCard}\}$. For the data part of the ATM, we use the propositional variable check. Our first ed signature for the ATM system thus is $\Sigma_0 = (E_0, \delta_P : \emptyset \to P)$ where $P = \{\mathsf{check}\}$.

If we wish to be able to cancel transactions, we extend E_0 with the event cancel and get the larger ed signature $\Sigma_1 = (E_1, \delta_P : \emptyset \to P)$ with $E_1 = E_0 \cup \{\mathsf{cancel}\}$. Then $\sigma_0 = (\eta_0, 1_{\delta_P}) : \Sigma_0 \to \Sigma_1$ with $\eta_0 : E_0 \hookrightarrow E_1$ the inclusion is an ed signature morphism. (For shortening the presentation we omit further events like withdrawing money, etc.)□

4.2. Structures of $\mathcal{E}^{\downarrow}(\vec{\mathcal{D}})$

Any ed signature Σ determines a class of semantic structures, called *event/data transition systems*, which are transition systems with sets of initial states and events as labels on transitions. Since we are interested here in describing (properties of) reactive systems which start their executions in some initial states, the state space of our semantic models will be restricted to reachable states only. To capture the dynamic and the data aspect of event/based systems, the states of our transition systems are pairs $\gamma = (c, d)$, called *configurations*, where c is a *control state* recording the current execution state and the *data state* d is labelled by a $\vec{\mathcal{D}}$-structure $\omega(d)$ over $\delta(\Sigma)$. The usefulness of the data state labelling will be detailed in Ex. 17 in connection with the satisfaction condition of institutions. It is related to the function M in hybridised logics [MMDB11] which maps states to structures of an underlying base institution.

Definition 5 A Σ-*event/data transition system* (Σ-*edts*) $M = (\Gamma, R, \Gamma_0, \omega)$ over an ed signature Σ consists of

- a set of *configurations* $\Gamma \subseteq C \times D$ of pairs of *control states* from a set C and *data states* from a set D;
- a family of *transition relations* $R = (R_e \subseteq \Gamma \times \Gamma)_{e \in E(\Sigma)}$;
- a non-empty set of *initial configurations* $\Gamma_0 \subseteq \{c_0\} \times D_0 \subseteq \Gamma$ with a unique *initial control state* $c_0 \in C$; and
- a *data state labelling* $\omega : D \to |\mathit{Str}^{\vec{\mathcal{D}}}(\delta(\Sigma))|$,

such that Γ is *reachable* via R, i.e., for all $\gamma \in \Gamma$ there are $\gamma_0 \in \Gamma_0$, $n \geq 0$, $e_1, \ldots, e_n \in E(\Sigma)$, and $(\gamma_i, \gamma_{i+1}) \in R_{e_{i+1}}$ for all $0 \leq i < n$ with $\gamma_n = \gamma$.

We write $\omega(M)$ for ω, $c(M)(\gamma)$ for c and $\omega(M)(\gamma)$ for $\omega(M)(d)$ if $\gamma = (c, d) \in \Gamma$, $\Gamma(M)$ for Γ, $C(M)$ for $\{c(M)(\gamma) \mid \gamma \in \Gamma(M)\}$, $\Omega(M)$ for $\{\omega(M)(\gamma) \mid \gamma \in \Gamma(M)\}$, $R(M)$ for R, $\Gamma_0(M)$ for Γ_0, $c_0(M)$ for c_0, and $\Omega_0(M)$ for $\{\omega(M)(\gamma_0) \mid \gamma_0 \in \Gamma_0(M)\}$.

A Σ-*edts morphism* $h : M_1 \to M_2$ is given by a function $h : \Gamma(M_1) \to \Gamma(M_2)$ such that

- $c(M_2)(h(\gamma_1)) = c(M_2)(h(\gamma_1'))$ for all $\gamma_1, \gamma_1' \in \Gamma(M_1)$ with $c(M_1)(\gamma_1) = c(M_1)(\gamma_1')$;
- $h(\gamma_{1,0}) \in \Gamma_0(M_2)$ for all $\gamma_{1,0} \in \Gamma_0(M_1)$; and
- $(h(\gamma_1), h(\gamma_1')) \in R(M_2)_e$ for all $(\gamma_1, \gamma_1') \in R(M_1)_e$ and $e \in E(\Sigma)$. □

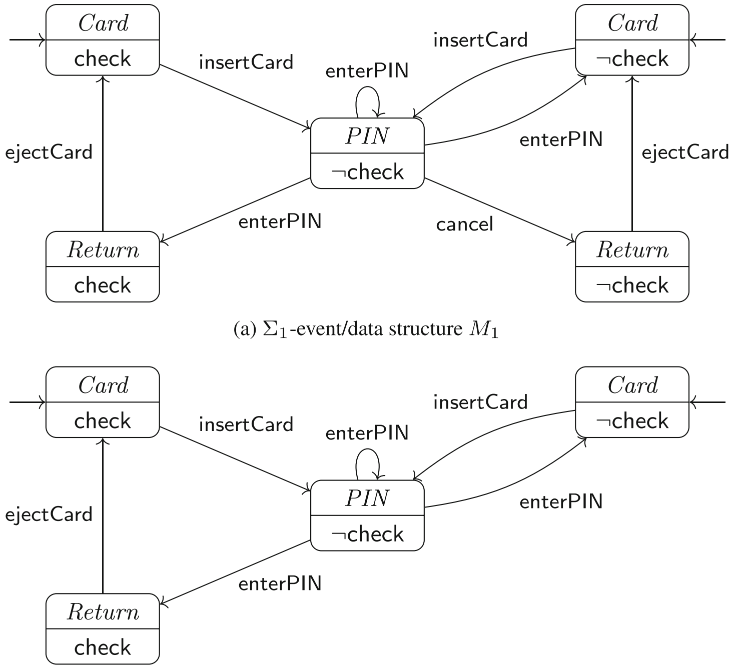
For any ed signature Σ, the class of Σ-edts and their morphisms form a category denoted by $\mathit{Str}^{\mathcal{E}^{\downarrow}(\vec{\mathcal{D}})}(\Sigma)$.

A Σ-edts M is *extensional* if $\omega(M)$ is injective, i.e., $\omega(M)(\gamma_1) = \omega(M)(\gamma_2)$ implies $\gamma_1 = \gamma_2$; in this case the data states and the data labels can be identified.

Example 12 Continuing Ex. 11, a Σ_1-edts M_1 in $\mathit{Str}^{\mathcal{E}^{\downarrow}(\mathit{Prop}_{\emptyset})}(\Sigma_1)$ is shown in Fig. 1a. Its control states are Card, PIN, and Return. The data states are check and ¬check standing for the functions $\mu_{\mathsf{check}}, \mu_{\neg\mathsf{check}} : \{\mathsf{check}\} \to \mathbb{B}$ with $\mu_{\mathsf{check}}(\mathsf{check}) = \mathit{tt}$ and $\mu_{\neg\mathsf{check}}(\mathsf{check}) = \mathit{ff}$. The data labelling ω is just the identity such that M_1 is extensional. The control state Card is initial with two initial data states, check and ¬check. There is no configuration with control state PIN and data state check in M_1 since such a configuration is not reachable. There is a non-deterministic choice for enterPIN when the control state is PIN. The first alternative going to configuration $(\mathit{Return}, \mathsf{check})$ models the situation where a correct PIN has been entered. The other alternatives either allow to repeat entering an incorrect PIN or stop the process if sufficiently often an incorrect PIN has been entered. In the latter case the card is kept.

□

Thus we have already defined the first part of the (contravariant) structures functor $\mathit{Str}^{\mathcal{E}^{\downarrow}(\vec{\mathcal{D}})} : (\mathbb{S}^{\mathcal{E}^{\downarrow}(\vec{\mathcal{D}})})^{\mathrm{op}} \to \mathrm{Cat}$ mapping each ed signature Σ to the category of Σ-edts.



(a) Σ_1-event/data structure M_1

(b) Σ_0-event/data structure $M_0 = \mathit{Str}^{\mathcal{E}^{\downarrow}(\vec{\mathcal{D}})}(\sigma_0)(M_1)$ (reduct of M_1 along σ_0)

Fig. 1. Sample edts M_1 and M_0 for an ATM

It remains to define, for any signature morphism $\sigma : \Sigma \to \Sigma'$ in $\mathbb{S}^{\mathcal{E}^{\downarrow}(\vec{\mathcal{D}})}$, the reduct functor $\mathit{Str}^{\mathcal{E}^{\downarrow}(\vec{\mathcal{D}})}(\sigma) : \mathit{Str}^{\mathcal{E}^{\downarrow}(\vec{\mathcal{D}})}(\Sigma') \to \mathit{Str}^{\mathcal{E}^{\downarrow}(\vec{\mathcal{D}})}(\Sigma)$ that sends Σ'-edts and their morphisms to their reducts. Special care has to be taken because of the reachability property of event/data transition systems: for any Σ'-edts M' its reduct along σ must be a Σ-edts with reachable configurations. Therefore we propose an inductive definition for reducts of edts. Of course, the reducts of the data parts of configurations will be the reducts of data states as provided by the data state institution $\vec{\mathcal{D}}$.

Definition 6 Let $\sigma : \Sigma \to \Sigma'$ be an ed signature morphism and M' a Σ'-edts. The σ-*reduct* of M' is the Σ-edts $M'|\sigma$ such that

- $\Gamma(M'|\sigma)$ and $R(M'|\sigma) = (R(M'|\sigma)_e)_{e \in E(\Sigma)}$ are inductively defined by $\Gamma_0(M') \subseteq \Gamma(M'|\sigma)$ and, for all $\gamma', \gamma'' \in \Gamma(M')$, $e \in E(\Sigma)$, if $\gamma' \in \Gamma(M'|\sigma)$ and $(\gamma', \gamma'') \in R(M')_{E(\sigma)(e)}$, then $\gamma'' \in \Gamma(M'|\sigma)$ and $(\gamma', \gamma'') \in R(M'|\sigma)_e$;
- $\Gamma_0(M'|\sigma) = \Gamma_0(M')$; and
- $\omega(M'|\sigma)(\gamma') = \mathit{Str}^{\vec{\mathcal{D}}}(\delta(\sigma))(\omega(M')(\gamma'))$ for all $\gamma' \in \Gamma(M'|\sigma)$. □

Note that in particular $\Gamma(M'|\sigma) \subseteq \Gamma(M')$, $c(M'|\sigma)(\gamma') = c(M')(\gamma')$ for all $\gamma' \in \Gamma(M'|\sigma)$, and therefore $C(M'|\sigma) \subseteq C(M')$.

Example 13 Continuing Ex. 12, the reduct M_0 of the Σ_1-edts M_1 in Fig. 1a along the ed signature morphism $\sigma_0 : \Sigma_0 \to \Sigma_1$ of Ex. 11 is shown in Fig. 1b. It does not contain the configuration $(\mathit{Return}, \neg\mathsf{check})$ any more, as Σ_0 does not show the event cancel and thus $(\mathit{Return}, \neg\mathsf{check})$ becomes unreachable. □

Lemma 4 *Let $\sigma : \Sigma \to \Sigma'$ in $\mathbb{S}^{\mathcal{E}^{\downarrow}(\vec{\mathcal{D}})}$ and $h' : M'_1 \to M'_2$ be a Σ'-edts morphism in $\mathit{Str}^{\mathcal{E}^{\downarrow}(\vec{\mathcal{D}})}(\Sigma')$. Then $h'|\sigma : \Gamma(M'_1|\sigma) \to \Gamma(M'_2|\sigma)$ with $(h'|\sigma)(\gamma'_1) = h'(\gamma'_1)$ is a Σ-edts morphism from $M'_1|\sigma$ to $M'_2|\sigma$.*

Proof. The well-definedness of $h'|\sigma$ follows directly from the inductive definition of reducts. □

For any $\sigma : \Sigma \to \Sigma'$ in $\mathbb{S}^{\mathcal{E}^{\downarrow}(\vec{\mathcal{D}})}$, the map $\mathit{Str}^{\mathcal{E}^{\downarrow}(\vec{\mathcal{D}})}(\sigma) : \mathit{Str}^{\mathcal{E}^{\downarrow}(\vec{\mathcal{D}})}(\Sigma') \to \mathit{Str}^{\mathcal{E}^{\downarrow}(\vec{\mathcal{D}})}(\Sigma)$ that sends Σ'-edts and their morphisms to their reducts is a functor. This lifts to the desired functor $\mathit{Str}^{\mathcal{E}^{\downarrow}(\vec{\mathcal{D}})} : (\mathbb{S}^{\mathcal{E}^{\downarrow}(\vec{\mathcal{D}})})^{\mathrm{op}} \to \mathrm{Cat}$ mapping each ed signature to the category of its structures and each ed signature morphism to its reduct functor.

4.3. Sentences of $\mathcal{E}^{\downarrow}(\vec{\mathcal{D}})$

Actions. Atomic actions over an ed signature Σ are given by expressions of the form $e /\!\!/ \psi$ with $e \in E(\Sigma)$ an event and $\psi \in \mathrm{Sen}^{2\vec{\mathcal{D}}}(\delta(\Sigma))$ a state transition predicate formalised as a sentence in the 2-data state institution $2\vec{\mathcal{D}}$; see Sect. 3.2. The intuition is that the occurrence of the event e causes a state transition in accordance with ψ, i.e., the pre- and post-data states satisfy ψ. Thus ψ specifies the possible effects of e. Following the ideas of dynamic logic we also use complex, structured actions formed over atomic actions by union "+" (expressing alternatives), sequential composition "; " and iteration "*".

Definition 7 Let $\Sigma = (E, \delta : \Delta_0 \to \Delta)$ be an ed signature. The set $\Lambda(\Sigma)$ of Σ-*event/data actions* (Σ-*ed actions*) is given by the grammar

$$\lambda ::= e /\!\!/ \psi \ \mid \ \lambda_1 + \lambda_2 \ \mid \ \lambda_1;\ \lambda_2 \ \mid \ \lambda^*$$

where $e \in E$ and $\psi \in \mathrm{Sen}^{2\vec{\mathcal{D}}}(\delta)$. □

We use the following shorthand notations for ed actions taking into account that $2\vec{\mathcal{D}}$ comprises the sentence true. For an event $e \in E$, we also write e for the atomic action $e /\!\!/ \mathrm{true}$, and for a finite subset $F = \{e_1, \dots, e_k\} \subseteq E$, we also write $\{e_1, \dots, e_k\}$ or simply $\boldsymbol{F}$ to denote the complex action $e_1 + \dots + e_k$. In particular, if E is finite we write $\boldsymbol{E}$ for the composed action obtained by combining with "+" all elements of E. This captures the choice of all possible events with arbitrary effects. Moreover, if E is finite we write $-\{e_1, \dots, e_k\}$ for the composed action obtained by combining with "+" all elements of $E \setminus \{e_1, \dots, e_k\}$ and briefly $-e$ for $-\{e\}$.

Example 14 For the ATM signature Σ_0 in Ex. 11, a Σ_0-ed action $\mathsf{enterPIN} /\!\!/ \mathsf{check}'$ expresses that after $\mathsf{enterPIN}$ has occurred the propositional variable check is true. The action $\boldsymbol{E}_0^*;\ \mathsf{insertCard}$ here abbreviates $(\mathsf{insertCard} /\!\!/ \mathrm{true} + \mathsf{enterPIN} /\!\!/ \mathrm{true} + \mathsf{ejectCard} /\!\!/ \mathrm{true})^*;\ \mathsf{insertCard} /\!\!/ \mathrm{true}$ and means an arbitrary sequence of event occurrences and data transitions ending in an occurrence of $\mathsf{insertCard}$. □

Let $\sigma : \Sigma \to \Sigma'$ be an ed signature morphism. The *event/data action translation* $\Lambda(\sigma) : \Lambda(\Sigma) \to \Lambda(\Sigma')$ along σ is recursively given by

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

Formulæ and sentences. The logical formulæ of $\mathcal{E}^{\downarrow}(\vec{\mathcal{D}})$ are a combination of features from dynamic logic [HKT00] and hybrid logic [Bra10]. From dynamic logic we use modalities filled with regular expressions of actions. In our context the atomic actions are event/data actions $e /\!\!/ \psi$ as introduced above. As usual a diamond formula $\langle \lambda \rangle \varrho$ expresses that it is possible to execute λ in the current state such that ϱ is satisfied in the subsequent state; the derived box modality $[\lambda]\varrho$ expresses that whenever λ is executed in the current state then it is necessary that ϱ holds in any subsequent state. From hybrid logic we use state variables which can be bound to the current control state by the binder operator $\downarrow x \,.\, \varrho$ for further reference in formula ϱ, and the jump operator $(@^F x)\varrho$ which moves the state of evaluation for ϱ to the state bound by x.

The binder operator was first studied by [Gor94]. In contrast to the classical approach we are dealing here with configurations which are pairs of control and data state. We claim that state variables should refer to the current point of control flow of a system and that binders and jumps should provide means to model the control flow. For instance, a sentence like $\downarrow x \,.\, \langle \mathsf{inc} /\!\!/ \mathsf{c}' = \mathsf{c} + 1 \rangle x$ should model a looping behaviour where an attribute c is constantly incremented by 1. Thus x should not be bound to a configuration including a data state. Therefore, variables in $\mathcal{E}^{\downarrow}(\vec{\mathcal{D}})$-formulæ denote just control states and not configurations; i.e., for the hybrid part of our logic, data states are disregarded. This will be reflected in the satisfaction of $\mathcal{E}^{\downarrow}(\vec{\mathcal{D}})$-formulæ below. Another variation concerns the jump operator $(@^F x)\varrho$ which parametrises the jump operator $(@x)\varrho$ of hybrid logic by a set F of events. It has the effect that ϱ will be evaluated in all configurations having the control state determined by x *and* being reachable with events from F. We will illustrate later, in Ex. 18, that this relativisation is useful to get the satisfaction condition of an institution for $\mathcal{E}^{\downarrow}(\vec{\mathcal{D}})$-logic.

$\mathcal{E}^{\downarrow}(\vec{\mathcal{D}})$ retains from hybrid logic the use of binders and jumps, but omits free nominals. Thus sentences, i.e., formulæ without free variables, become restricted to express properties of configurations reachable from the initial ones.

In the following of this paper we always assume given a countably infinite set X of control state variables.

Definition 8 The set $\mathscr{F}(\Sigma)$ of Σ-*event/data formulæ* over an ed signature Σ is defined by the following grammar

$$\varrho ::= \varphi \mid x \mid {\downarrow} x \,.\, \varrho \mid (@^{F} x)\varrho \mid \langle\lambda\rangle\varrho \mid \neg\varrho \mid \varrho_1 \vee \varrho_2$$

where $\varphi \in \mathrm{Sen}^{\vec{\mathcal{D}}}(\delta(\Sigma))$, $x \in X$, $F \subseteq E(\Sigma)$, $\lambda \in \Lambda(\Sigma)$. The set of free variables of a formula is defined as usual with ${\downarrow}x$ being the unique operator binding variables. A Σ-*event/data sentence* (Σ-*ed sentence*) is a Σ-event/data formula without free variables. The set $\mathrm{Sen}^{\mathcal{E}^{\downarrow}(\vec{\mathcal{D}})}(\Sigma)$ consists of all Σ-ed sentences. □

We write $[\lambda]\varrho$ for $\neg\langle\lambda\rangle\neg\varrho$ and we use the usual boolean connectives. Furthermore, we express the (unrelativised) jump operator $(@x)\varrho$ of hybrid logic by $(@^{E(\Sigma)}x)\varrho$.

Using additionally the shorthand notations for actions, we can specify safety properties with $[\boldsymbol{E}^*]\varrho$; deadlock freedom is expressed by $[E^*]\langle\boldsymbol{E}\rangle\mathrm{true}$. Liveness properties, like "whenever an event e has happened, an event e' can eventually occur", can be expressed by $[\boldsymbol{E}^*;\ e]\langle\boldsymbol{E}^*;\ e'\rangle\mathrm{true}$. We can also express that an event e' must never occur when an event e has happened before with $[\boldsymbol{E}^*;\ e;\ \boldsymbol{E}^*;\ e']\mathrm{false}$. Of course, events e and e' standing for $e /\!\!/ \mathrm{true}$ and $e' /\!\!/ \mathrm{true}$ could be more generally replaced by $e /\!\!/ \psi$ and $e' /\!\!/ \psi'$. These kinds of properties are suited for abstract requirements specifications. They use only the dynamic logic fragment of $\mathcal{E}^{\downarrow}(\vec{\mathcal{D}})$.

Example 15 Continuing Ex. 14, we can express some abstract properties required for an ATM using ed signature Σ_0 of Ex. 11:

(0.1) "Whenever a card has been inserted, a correct PIN can eventually be entered."

$$[\boldsymbol{E}_{\boldsymbol{0}}^*;\ \mathsf{insertCard}]\langle\boldsymbol{E}_{\boldsymbol{0}}^*;\ \mathsf{enterPIN} /\!\!/ \mathsf{check}'\rangle\mathrm{true}$$

(0.2) "Whenever a correct PIN has been entered, the card can eventually be ejected."

$$[\boldsymbol{E}_{\boldsymbol{0}}^*;\ (\mathsf{enterPIN} /\!\!/ \mathsf{check}')]\langle\boldsymbol{E}_{\boldsymbol{0}}^*;\ \mathsf{ejectCard}\rangle\mathrm{true}$$

(0.3) "A card cannot be ejected if it was not inserted before."

$$[(-\mathsf{insertCard})^*;\ \mathsf{ejectCard}]\mathrm{false}$$

Note that the complex action $-\mathsf{insertCard}$ in (0.3) ranges over all events of Σ_0 except $\mathsf{insertCard}$. □

The logic $\mathcal{E}^{\downarrow}(\vec{\mathcal{D}})$ is also suited to directly express process structures and, thus, the implementation of abstract requirements. The binder operator is crucial for this. The ability to give names to visited control states, together with the modal features to express transitions, makes possible a precise description of the whole dynamics of a process structure in a single sentence. This will be significant in Sect. 5.3. Binders allow to express recursive patterns, namely loop transitions from the current to some already visited control state. Actually, this kind of properties cannot be specified in the absence of a feature to refer to specific control states in a model, as in standard modal logic. For example, sentence

$${\downarrow}x_0 \,.\, \big(\langle e\rangle x_0 \wedge \langle f\rangle {\downarrow} x_1 \,.\, (\langle e\rangle x_0 \wedge \langle f\rangle x_1)\big)$$

specifies process structures with two states represented by x_0 and x_1. Event e loops in x_0, event f moves to x_1 and loops in x_1 while e moves back to x_0. To model that this is the only allowed behaviour one could expand the sentence within the scope of x_0 and x_1 by

$$(@x_0)([e]x_0 \wedge [f]x_1) \wedge (@x_1)([e]x_0 \wedge [f]x_1)$$

Clearly, structures like this can also involve specific data properties which will be illustrated later in our development.

To get the sentence functor of an institution it remains to define sentence translation. Let $\sigma : \Sigma \to \Sigma'$ be an ed signature morphism. The *event/data formulæ translation* $\mathscr{F}(\sigma) : \mathscr{F}(\Sigma) \to \mathscr{F}(\Sigma')$ along σ is recursively given by

- $\mathscr{F}(\sigma)(\varphi) = \mathrm{Sen}^{\vec{\mathcal{D}}}(\delta(\sigma))(\varphi)$;
- $\mathscr{F}(\sigma)(x) = x$;
- $\mathscr{F}(\sigma)({\downarrow}x \,.\, \varrho) = {\downarrow}x \,.\, \mathscr{F}(\sigma)(\varrho)$;
- $\mathscr{F}(\sigma)((@^{F} x)\varrho = (@^{E(\sigma)(F)} x)\mathscr{F}(\sigma)(\varrho)$;

- $\mathscr{F}(\sigma)(\langle\lambda\rangle\varrho) = \langle\Lambda(\sigma)(\lambda)\rangle\mathscr{F}(\sigma)(\varrho)$;
- $\mathscr{F}(\sigma)(\neg\varrho) = \neg\mathscr{F}(\sigma)(\varrho)$;
- $\mathscr{F}(\sigma)(\varrho_1 \vee \varrho_2) = \mathscr{F}(\sigma)(\varrho_1) \vee \mathscr{F}(\sigma)(\varrho_2)$.

The *event/data sentence translation* $\mathrm{Sen}^{\mathcal{E}^{\downarrow}(\vec{\mathcal{D}})}(\sigma) : \mathrm{Sen}^{\mathcal{E}^{\downarrow}(\vec{\mathcal{D}})}(\Sigma) \to \mathrm{Sen}^{\mathcal{E}^{\downarrow}(\vec{\mathcal{D}})}(\Sigma')$ along the ed signature morphism σ is defined as $\mathrm{Sen}^{\mathcal{E}^{\downarrow}(\vec{\mathcal{D}})}(\sigma)(\varrho) = \mathscr{F}(\sigma)(\varrho)$ for each $\varrho \in \mathrm{Sen}^{\mathcal{E}^{\downarrow}(\vec{\mathcal{D}})}(\Sigma)$.

Mapping each signature $\Sigma \in |\, \mathbb{S}^{\mathcal{E}^{\downarrow}(\vec{\mathcal{D}})}|$ into the set of event/data sentences $\mathrm{Sen}^{\mathcal{E}^{\downarrow}(\vec{\mathcal{D}})}(\Sigma)$ and each ed signature morphism $\sigma : \Sigma \to \Sigma'$ in $\mathbb{S}^{\mathcal{E}^{\downarrow}(\vec{\mathcal{D}})}$ into the event/data sentence translation $\mathrm{Sen}^{\mathcal{E}^{\downarrow}(\vec{\mathcal{D}})}(\sigma)$ defines a functor $\mathrm{Sen}^{\mathcal{E}^{\downarrow}(\vec{\mathcal{D}})} : \mathbb{S}^{\mathcal{E}^{\downarrow}(\vec{\mathcal{D}})} \to \mathrm{Set}$.

4.4. Satisfaction relation of $\mathcal{E}^{\downarrow}(\vec{\mathcal{D}})$

To define the satisfaction relation for $\mathcal{E}^{\downarrow}(\vec{\mathcal{D}})$-logic we must first, as usual in dynamic logic, provide an interpretation of the actions $\Lambda(\Sigma)$ over a Σ-edts M as the family of relations $(R(M)_\lambda \subseteq \Gamma(M) \times \Gamma(M))_{\lambda\in\Lambda(\Sigma)}$ defined by

- $R(M)_{e/\!\!/\psi} = \{(\gamma, \gamma') \in R(M)_e \mid (\omega(M)(\gamma), \omega(M)(\gamma')) \models^{2\vec{\mathcal{D}}}_{\delta(\Sigma)} \psi\}$,
- $R(M)_{\lambda_1+\lambda_2} = R(M)_{\lambda_1} \cup R(M)_{\lambda_2}$, i.e., union of relations,
- $R(M)_{\lambda_1;\lambda_2} = R(M)_{\lambda_1};\ R(M)_{\lambda_2}$, i.e., sequential composition of relations,
- $R(M)_{\lambda^*} = (R(M)_\lambda)^*$, i.e., reflexive-transitive closure of relations.

The satisfaction of state formulæ φ relies on the satisfaction relation of the underlying data state institution $\vec{\mathcal{D}}$. Similarly, satisfaction of diamond formulæ $\langle\lambda\rangle\varrho$ relies, for $\lambda = e/\!\!/\psi$, on the satisfaction relation of the 2-data institution $2\vec{\mathcal{D}}$ (via the relations $R(M)_{e/\!\!/\psi}$). To define satisfaction of formulæ involving the jump operator we use the following (generalised) reachability notion:

Let Σ be an ed signature, $F \subseteq E(\Sigma)$, and M a Σ-edts. A configuration $\gamma \in \Gamma(M)$ is *F-reachable* in M if there are $\gamma_0 \in \Gamma_0(M)$, $n \geq 0$, $e_1, \ldots, e_n \in F$, and $(\gamma_i, \gamma_{i+1}) \in R(M)_{e_{i+1}}$ for all $0 \leq i < n$ with $\gamma_n = \gamma$. The set of F-reachable configurations of M is denoted by $\Gamma^F(M)$.

Definition 9 Given an ed signature Σ and a Σ-edts M, the *satisfaction* of an event/data formula $\varrho \in \mathscr{F}(\Sigma)$ is inductively defined w.r.t. *valuations* $v : X \to C(M)$ mapping control state variables to control states, and configurations $\gamma \in \Gamma(M)$:

- $M, v, \gamma \models^{\mathcal{E}^{\downarrow}(\vec{\mathcal{D}})}_{\Sigma} \varphi$ iff $\omega(M)(\gamma) \models^{\vec{\mathcal{D}}}_{\delta(\Sigma)} \varphi$;
- $M, v, \gamma \models^{\mathcal{E}^{\downarrow}(\vec{\mathcal{D}})}_{\Sigma} x$ iff $c(M)(\gamma) = v(x)$;
- $M, v, \gamma \models^{\mathcal{E}^{\downarrow}(\vec{\mathcal{D}})}_{\Sigma} {\downarrow}x\,.\,\varrho$ iff $M, v\{x \mapsto c(M)(\gamma)\}, \gamma \models^{\mathcal{E}^{\downarrow}(\vec{\mathcal{D}})}_{\Sigma} \varrho$;
- $M, v, \gamma \models^{\mathcal{E}^{\downarrow}(\vec{\mathcal{D}})}_{\Sigma} (@^F x)\varrho$ iff $M, v, \gamma' \models^{\mathcal{E}^{\downarrow}(\vec{\mathcal{D}})}_{\Sigma} \varrho$ for all $\gamma' \in \Gamma^F(M)$ with $c(M)(\gamma') = v(x)$;
- $M, v, \gamma \models^{\mathcal{E}^{\downarrow}(\vec{\mathcal{D}})}_{\Sigma} \langle\lambda\rangle\varrho$ iff $M, v, \gamma' \models^{\mathcal{E}^{\downarrow}(\vec{\mathcal{D}})}_{\Sigma} \varrho$ for some $\gamma' \in \Gamma(M)$ with $(\gamma, \gamma') \in R(M)_\lambda$;
- $M, v, \gamma \models^{\mathcal{E}^{\downarrow}(\vec{\mathcal{D}})}_{\Sigma} \neg\varrho$ iff $M, v, \gamma \not\models^{\mathcal{E}^{\downarrow}(\vec{\mathcal{D}})}_{\Sigma} \varrho$;
- $M, v, \gamma \models^{\mathcal{E}^{\downarrow}(\vec{\mathcal{D}})}_{\Sigma} \varrho_1 \vee \varrho_2$ iff $M, v, \gamma \models^{\mathcal{E}^{\downarrow}(\vec{\mathcal{D}})}_{\Sigma} \varrho_1$ or $M, v, \gamma \models^{\mathcal{E}^{\downarrow}(\vec{\mathcal{D}})}_{\Sigma} \varrho_2$.

A Σ-event/data sentence $\varrho \in \mathrm{Sen}^{\mathcal{E}^{\downarrow}(\vec{\mathcal{D}})}(\Sigma)$ is *satisfied* in M, denoted by $M \models^{\mathcal{E}^{\downarrow}(\vec{\mathcal{D}})}_{\Sigma} \varrho$, if $M, v, \gamma_0 \models^{\mathcal{E}^{\downarrow}(\vec{\mathcal{D}})}_{\Sigma} \varrho$ for all $\gamma_0 \in \Gamma_0(M)$ and an arbitrary valuation v (which is anyway irrelevant since sentences do not contain free variables). □

Example 16 Continuing Ex. 15, let us check that the statements (0.1) to (0.3) are satisfied in the Σ_0-edts M_0 depicted in Fig. 1b: For (0.1), whenever an insertCard has happened, configuration (PIN, ¬check) is entered. Then it is (immediately) possible to enter enterPIN with the effect that check holds. For (0.2), whenever an enterPIN has happened such that check holds, configuration ($Return$, check) is reached. Then ejectCard is (immediately) possible. Finally, (0.3) is obviously satisfied by M_0, since its first event is insertCard. □

4.5. Satisfaction condition for $\mathcal{E}^{\downarrow}(\vec{\mathcal{D}})$

The idea of the satisfaction condition is to ensure that satisfaction is invariant under change of notation. Formally, a change of notation is expressed by a signature morphism $\sigma : \Sigma \to \Sigma'$. In the context of $\mathcal{E}^{\downarrow}(\vec{\mathcal{D}})$-logic the satisfaction condition requires that for any Σ'-edts M' and for any Σ-sentence ϱ the reduct of M' along σ satisfies ϱ if, and only if, M' satisfies the sentence translation of ϱ w.r.t. σ. For the proof we use the following lemmas. The first lemma will be used to prove the satisfaction condition for sentences involving the diamond modality. It is crucial here that the 2-data institution $2\vec{\mathcal{D}}$ satisfies the satisfaction condition for state transition predicates, i.e., sentences in $2\vec{\mathcal{D}}$.

Lemma 5 Let $\sigma : \Sigma \to \Sigma'$ be an ed signature morphism, M' a Σ'-edts, and $\gamma_1' \in \Gamma(M'|\sigma) \subseteq \Gamma(M')$. Then

1. $(\gamma_1', \gamma_2') \in R(M'|\sigma)_{e/\!\!/\psi}$ if, and only if, $(\gamma_1', \gamma_2') \in R(M')_{\Lambda(\sigma)(e/\!\!/\psi)}$ for all $\gamma_2' \in \Gamma(M')$ and $e/\!\!/\psi \in \Lambda(\Sigma)$;
2. $\{\gamma_2' \in \Gamma(M'|\sigma) \mid (\gamma_1', \gamma_2') \in R(M'|\sigma)_\lambda\} = \{\gamma_2' \in \Gamma(M') \mid (\gamma_1', \gamma_2') \in R(M')_{\Lambda(\sigma)(\lambda)}\}$ for all $\lambda \in \Lambda(\Sigma)$.

Proof. For (1) let $\gamma_2' \in \Gamma(M')$ and $e/\!\!/\psi \in \Lambda(\Sigma)$:

$$\begin{aligned}
& (\gamma_1', \gamma_2') \in R(M'|\sigma)_{e/\!\!/\psi} \\
\Leftrightarrow\quad & \{ \text{ def. } R(M'|\sigma)_{e/\!\!/\psi} \ \} \\
& (\gamma_1', \gamma_2') \in R(M'|\sigma)_e \wedge (\omega(M'|\sigma)(\gamma_1'), \omega(M'|\sigma)(\gamma_2')) \models^{2\vec{\mathcal{D}}}_{\delta(\Sigma)} \psi \\
\Leftrightarrow\quad & \{ \text{ def. } R(M'|\sigma)_e \text{ and } \gamma_1' \in \Gamma(M'|\sigma) \ \} \\
& (\gamma_1', \gamma_2') \in R(M')_{E(\sigma)(e)} \wedge (\omega(M'|\sigma)(\gamma_1'), \omega(M'|\sigma)(\gamma_2')) \models^{2\vec{\mathcal{D}}}_{\delta(\Sigma)} \psi \\
\Leftrightarrow\quad & \{ \text{ def. } \omega(M'|\sigma) \ \} \\
& (\gamma_1', \gamma_2') \in R(M')_{E(\sigma)(e)} \wedge (Str^{\vec{\mathcal{D}}}(\delta(\sigma))(\omega(M')(\gamma_1')), Str^{\vec{\mathcal{D}}}(\delta(\sigma))(\omega(M')(\gamma_2'))) \models^{2\vec{\mathcal{D}}}_{\delta(\Sigma)} \psi \\
\Leftrightarrow\quad & \{ \text{ def. } Str^{2\vec{\mathcal{D}}} \ \} \\
& (\gamma_1', \gamma_2') \in R(M')_{E(\sigma)(e)} \wedge Str^{2\vec{\mathcal{D}}}(\delta(\sigma))(\omega(M')(\gamma_1'), \omega(M')(\gamma_2')) \models^{2\vec{\mathcal{D}}}_{\delta(\Sigma)} \psi \\
\Leftrightarrow\quad & \{ \text{ sat. cond. for } 2\vec{\mathcal{D}} \ \} \\
& (\gamma_1', \gamma_2') \in R(M')_{E(\sigma)(e)} \wedge (\omega(M')(\gamma_1'), \omega(M')(\gamma_2')) \models^{2\vec{\mathcal{D}}}_{\delta(\Sigma')} \mathrm{Sen}^{2\vec{\mathcal{D}}}(\delta(\sigma))(\psi) \\
\Leftrightarrow\quad & \{ \text{ def. } R(M')_{\Lambda(\sigma)(e/\!\!/\psi)} \ \} \\
& (\gamma_1', \gamma_2') \in R(M')_{\Lambda(\sigma)(e/\!\!/\psi)}
\end{aligned}$$

For (2) we apply induction on the structure of Σ-actions λ where the base case $\lambda = e/\!\!/\psi$ follows directly from (1). □

The next lemma is needed to get the satisfaction condition for sentences involving the relativised jump operator.

Lemma 6 Let $\sigma : \Sigma \to \Sigma'$ be an ed signature morphism, $F \subseteq E(\Sigma)$, and M' a Σ'-edts. Then, for all $\gamma' \in \Gamma(M')$, $\gamma' \in \Gamma^F(M'|\sigma)$ if, and only if, $\gamma' \in \Gamma^{E(\sigma)(F)}(M')$.

Proof. Let $\gamma' \in \Gamma(M')$. By induction, it holds that $\gamma' \in \Gamma^{E(\sigma)(F)}(M')$ if, and only if, there are $\gamma_0', \ldots, \gamma_n' \in \Gamma(M')$ and $e_1, \ldots, e_n \in F$ with $n \geq 0$, $\gamma_0' \in \Gamma_0(M')$, $(\gamma_i', \gamma_{i+1}') \in R(M')_{E(\sigma)(e_{i+1})}$ for all $0 \leq i < n$, and $\gamma' = \gamma_n'$. Now, $\gamma' \in \Gamma_0(M'|\sigma)$ if, and only if, $\gamma' \in \Gamma_0(M')$; and, if $\gamma' \in \Gamma(M'|\sigma)$, $\gamma'' \in \Gamma(M')$, and $e \in E(\Sigma)$, then $(\gamma', \gamma'') \in R(M'|\sigma)_e$ if, and only if $(\gamma', \gamma'') \in R(M')_{E(\sigma)(e)}$. Thus by induction, $\gamma' \in \Gamma^F(M'|\sigma)$ if, and only if, $\gamma' \in \Gamma^{E(\sigma)(F)}(M')$. □

Finally, the next lemma is formulated for formulæ, possibly involving free variables, in order to be able to perform induction on the structure of formulæ. The satisfaction condition for sentences stated in the subsequent corollary is a direct consequence.

Lemma 7 Let $\sigma : \Sigma \to \Sigma'$ be an ed signature morphism and M' a Σ'-edts. For all $\varrho \in \mathscr{F}(\Sigma)$, all $\gamma' \in \Gamma(M'|\sigma) \subseteq \Gamma(M')$, and all $v : X \to C(M'|\sigma) \subseteq C(M')$ it holds that

$$M'|\sigma, v, \gamma' \models^{\mathcal{E}^{\downarrow}(\vec{\mathcal{D}})}_{\Sigma} \varrho \iff M', v, \gamma' \models^{\mathcal{E}^{\downarrow}(\vec{\mathcal{D}})}_{\Sigma'} \mathscr{F}(\sigma)(\varrho)\,.$$

Proof. We apply induction on the structure of Σ-event/data formulæ. We only consider the cases φ, x, $\downarrow x\,.\,\varrho$, $(@^{F}x)\varrho$, and $\langle\lambda\rangle\varrho$; negation and disjunction are straightforward.

Case φ:

$$
\begin{array}{ll}
 & M'|\sigma, v, \gamma' \models^{\mathcal{E}^{\downarrow}(\vec{\mathcal{D}})}_{\Sigma} \varphi \\
\Leftrightarrow & \quad \{\text{ def. } \models^{\mathcal{E}^{\downarrow}(\vec{\mathcal{D}})} \} \\
 & \omega(M'|\sigma)(\gamma') \models^{\vec{\mathcal{D}}}_{\delta(\Sigma)} \varphi \\
\Leftrightarrow & \quad \{\text{ def. } \omega(M'|\sigma) \} \\
 & Str^{\vec{\mathcal{D}}}(\delta(\sigma))(\omega(M')(\gamma')) \models^{\vec{\mathcal{D}}}_{\delta(\Sigma)} \varphi \\
\Leftrightarrow & \quad \{\text{ sat. cond. for } \vec{\mathcal{D}} \} \\
 & \omega(M')(\gamma') \models^{\vec{\mathcal{D}}}_{\delta(\Sigma')} \mathrm{Sen}^{\vec{\mathcal{D}}}(\delta(\sigma))(\varphi) \\
\Leftrightarrow & \quad \{\text{ def. } \models^{\mathcal{E}^{\downarrow}(\vec{\mathcal{D}})} \} \\
 & M', v, \gamma' \models^{\mathcal{E}^{\downarrow}(\vec{\mathcal{D}})}_{\Sigma'} \mathrm{Sen}^{\vec{\mathcal{D}}}(\delta(\sigma))(\varphi) \\
\Leftrightarrow & \quad \{\text{ def. } \mathscr{F}(\sigma) \} \\
 & M', v, \gamma' \models^{\mathcal{E}^{\downarrow}(\vec{\mathcal{D}})}_{\Sigma'} \mathscr{F}(\sigma)(\varphi)
\end{array}
$$

Case x:

$$
\begin{array}{ll}
 & M'|\sigma, v, \gamma' \models^{\mathcal{E}^{\downarrow}(\vec{\mathcal{D}})}_{\Sigma} x \\
\Leftrightarrow & \quad \{\text{ def. } \models^{\mathcal{E}^{\downarrow}(\vec{\mathcal{D}})} \} \\
 & v(x) = c(M'|\sigma)(\gamma') \\
\Leftrightarrow & \quad \{\text{ def. } |\sigma \} \\
 & v(x) = c(M')(\gamma') \\
\Leftrightarrow & \quad \{\text{ def. } \models^{\mathcal{E}^{\downarrow}(\vec{\mathcal{D}})} \} \\
 & M', v, \gamma' \models^{\mathcal{E}^{\downarrow}(\vec{\mathcal{D}})}_{\Sigma'} x \\
\Leftrightarrow & \quad \{\text{ def. } \mathscr{F}(\sigma) \} \\
 & M', v, \gamma' \models^{\mathcal{E}^{\downarrow}(\vec{\mathcal{D}})}_{\Sigma'} \mathscr{F}(\sigma)(x)
\end{array}
$$

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

$$
\begin{array}{ll}
 & M'|\sigma, v, \gamma' \models^{\mathcal{E}^{\downarrow}(\vec{\mathcal{D}})}_{\Sigma} \downarrow x\,.\,\varrho \\
\Leftrightarrow & \quad \{\text{ def. } \models^{\mathcal{E}^{\downarrow}(\vec{\mathcal{D}})} \} \\
 & M'|\sigma, v\{x \mapsto c(M'|\sigma)(\gamma')\}, \gamma' \models^{\mathcal{E}^{\downarrow}(\vec{\mathcal{D}})}_{\Sigma} \varrho \\
\Leftrightarrow & \quad \{\text{ def. } c(M'|\sigma) \text{ and I. H. } \} \\
 & M', v\{x \mapsto c(M')(\gamma')\}, \gamma' \models^{\mathcal{E}^{\downarrow}(\vec{\mathcal{D}})}_{\Sigma'} \mathscr{F}(\sigma)(\varrho) \\
\Leftrightarrow & \quad \{\text{ def. } \models^{\mathcal{E}^{\downarrow}(\vec{\mathcal{D}})} \} \\
 & M', v, \gamma' \models^{\mathcal{E}^{\downarrow}(\vec{\mathcal{D}})}_{\Sigma'} \downarrow x\,.\,\mathscr{F}(\sigma)(\varrho) \\
\Leftrightarrow & \quad \{\text{ def. } \mathscr{F}(\sigma) \} \\
 & M', v, \gamma' \models^{\mathcal{E}^{\downarrow}(\vec{\mathcal{D}})}_{\Sigma'} \mathscr{F}(\sigma)(\downarrow x\,.\,\varrho)
\end{array}
$$

Case $(@^F x)\varrho$:

$$M'|\sigma, v, \gamma' \models_{\Sigma}^{\mathcal{E}^{\downarrow}(\vec{\mathcal{D}})} (@^F x)\varrho$$

$\Leftrightarrow$ $\{$ def. $\models^{\mathcal{E}^{\downarrow}(\vec{\mathcal{D}})}$ $\}$

$$M'|\sigma, v, \gamma'' \models_{\Sigma}^{\mathcal{E}^{\downarrow}(\vec{\mathcal{D}})} \varrho \quad \text{for all } \gamma'' \in \Gamma^F(M'|\sigma) \text{ with } c(M'|\sigma')(\gamma'') = v(x)$$

$\Leftrightarrow$ $\{$ Lem. 6 $\}$

$$M'|\sigma, v, \gamma'' \models_{\Sigma}^{\mathcal{E}^{\downarrow}(\vec{\mathcal{D}})} \varrho \quad \text{for all } \gamma'' \in \Gamma^F(M'|\sigma) = \Gamma^{E(\sigma)(F)}(M') \text{ with } c(M'|\sigma)(\gamma'') = v(x)$$

$\Leftrightarrow$ $\{$ def. $c(M'|\sigma)$ and I. H. $\}$

$$M', v, \gamma'' \models_{\Sigma'}^{\mathcal{E}^{\downarrow}(\vec{\mathcal{D}})} \mathscr{F}(\sigma)(\varrho) \quad \text{for all } \gamma'' \in \Gamma^{E(\sigma)(F)}(M') \text{ with } c(M')(\gamma'') = v(x)$$

$\Leftrightarrow$ $\{$ def. $\models^{\mathcal{E}^{\downarrow}(\vec{\mathcal{D}})}$ $\}$

$$M', v, \gamma' \models_{\Sigma'}^{\mathcal{E}^{\downarrow}(\vec{\mathcal{D}})} (@^{E(\sigma)(F)} x)\mathscr{F}(\sigma)(\varrho)$$

$\Leftrightarrow$ $\{$ def. $\mathscr{F}(\sigma)$ $\}$

$$M', v, \gamma' \models_{\Sigma'}^{\mathcal{E}^{\downarrow}(\vec{\mathcal{D}})} \mathscr{F}(\sigma)((@^F x)\varrho)$$

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

$$M'|\sigma, v, \gamma' \models_{\Sigma}^{\mathcal{E}^{\downarrow}(\vec{\mathcal{D}})} \langle\lambda\rangle\varrho$$

$\Leftrightarrow$ $\{$ def. $\models^{\mathcal{E}^{\downarrow}(\vec{\mathcal{D}})}$ $\}$

$$M'|\sigma, v, \gamma'' \models_{\Sigma}^{\mathcal{E}^{\downarrow}(\vec{\mathcal{D}})} \varrho \quad \text{for some } \gamma'' \in \Gamma(M'|\sigma) \text{ with } (\gamma', \gamma'') \in R(M'|\sigma)_{\lambda}$$

$\Leftrightarrow$ $\{$ Lem. 5(2) $\}$

$$M'|\sigma, v, \gamma'' \models_{\Sigma}^{\mathcal{E}^{\downarrow}(\vec{\mathcal{D}})} \varrho \quad \text{for some } \gamma'' \in \Gamma(M'|\sigma) \subseteq \Gamma(M') \text{ with } (\gamma', \gamma'') \in R(M')_{\Lambda(\sigma)(\lambda)}$$

$\Leftrightarrow$ $\{$ I. H. by Lem. 5(2) for "$\Leftarrow$" $\}$

$$M', v, \gamma'' \models_{\Sigma'}^{\mathcal{E}^{\downarrow}(\vec{\mathcal{D}})} \mathscr{F}(\sigma)(\varrho) \quad \text{for some } \gamma'' \in \Gamma(M') \text{ with } (\gamma', \gamma'') \in R(M')_{\Lambda(\sigma)(\lambda)}$$

$\Leftrightarrow$ $\{$ def. $\models^{\mathcal{E}^{\downarrow}(\vec{\mathcal{D}})}$ $\}$

$$M', v, \gamma' \models_{\Sigma'}^{\mathcal{E}^{\downarrow}(\vec{\mathcal{D}})} \langle\Lambda(\sigma)(\lambda)\rangle\mathscr{F}(\sigma)(\varrho)$$

$\Leftrightarrow$ $\{$ def. $\mathscr{F}(\sigma)$ $\}$

$$M', v, \gamma' \models_{\Sigma'}^{\mathcal{E}^{\downarrow}(\vec{\mathcal{D}})} \mathscr{F}(\sigma)(\langle\lambda\rangle\varrho) \qquad \square$$

Corollary 1 (Satisfaction condition for $\mathcal{E}^{\downarrow}(\vec{\mathcal{D}})$) *For any ed signature morphism $\sigma : \Sigma \to \Sigma'$ in $\mathbb{S}^{\mathcal{E}^{\downarrow}(\vec{\mathcal{D}})}$, $M' \in| Str^{\mathcal{E}^{\downarrow}(\vec{\mathcal{D}})}(\Sigma')|$, and $\varrho \in \mathrm{Sen}^{\mathcal{E}^{\downarrow}(\vec{\mathcal{D}})}(\Sigma)$,*

$$Str^{\mathcal{E}^{\downarrow}(\vec{\mathcal{D}})}(\sigma)(M') \models_{\Sigma}^{\mathcal{E}^{\downarrow}(\vec{\mathcal{D}})} \varrho \iff M' \models_{\Sigma'}^{\mathcal{E}^{\downarrow}(\vec{\mathcal{D}})} \mathrm{Sen}^{\mathcal{E}^{\downarrow}(\vec{\mathcal{D}})}(\sigma)(\varrho) .$$

Proof. By unfolding the definitions we obtain

$$Str^{\mathcal{E}^{\downarrow}(\vec{\mathcal{D}})}(\sigma)(M') \models_{\Sigma}^{\mathcal{E}^{\downarrow}(\vec{\mathcal{D}})} \varrho$$

$\Leftrightarrow$ $\{$ def. $\models^{\mathcal{E}^{\downarrow}(\vec{\mathcal{D}})}$ for sentences, $Str^{\mathcal{E}^{\downarrow}(\vec{\mathcal{D}})}(\sigma)(M') = M'|\sigma$ $\}$

$$M'|\sigma, v, \gamma'_0 \models_{\Sigma}^{\mathcal{E}^{\downarrow}(\vec{\mathcal{D}})} \varrho \quad \text{for all } \gamma'_0 \in \Gamma_0(M'|\sigma) = \Gamma_0(M') \text{ and some valuation } v : X \to C(M'|\sigma)$$

$\Leftrightarrow$ $\{$ Lem. 7 $\}$

$$M', v, \gamma'_0 \models_{\Sigma'}^{\mathcal{E}^{\downarrow}(\vec{\mathcal{D}})} \mathscr{F}(\sigma)(\varrho) \quad \text{for all } \gamma'_0 \in \Gamma_0(M') \text{ and some valuation } v : X \to C(M'|\sigma) \subseteq C(M')$$

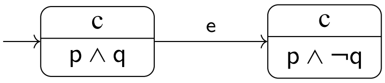
$\Leftrightarrow$ $\{$ def. $\models^{\mathcal{E}^{\downarrow}(\vec{\mathcal{D}})}$ for sentences $\}$

$$M' \models_{\Sigma'}^{\mathcal{E}^{\downarrow}(\vec{\mathcal{D}})} \mathrm{Sen}^{\mathcal{E}^{\downarrow}(\vec{\mathcal{D}})}(\sigma)(\varrho) \qquad \square$$

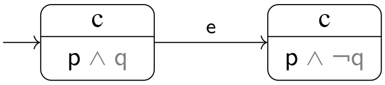
As an immediate consequence we obtain:

Theorem 2 $\mathcal{E}^{\downarrow}(\vec{\mathcal{D}}) = (\mathbb{S}^{\mathcal{E}^{\downarrow}(\vec{\mathcal{D}})}, Str^{\mathcal{E}^{\downarrow}(\vec{\mathcal{D}})}, \mathrm{Sen}^{\mathcal{E}^{\downarrow}(\vec{\mathcal{D}})}, \models^{\mathcal{E}^{\downarrow}(\vec{\mathcal{D}})})$ *is an institution for each data state institution $\vec{\mathcal{D}}$.* □

Example 17 This example illustrates the usefulness of the data state labelling ω in the definition of ed structures to get the satisfaction condition. We work in the institution $\mathcal{E}^{\downarrow}(\mathit{Prop}_{\emptyset})$. Let $E = E' = \{\mathsf{e}\}$ and $P = \{\mathsf{p}\}$, $P' = \{\mathsf{p}, \mathsf{q}\}$. Thus we have two ed signatures $\Sigma = (E, \delta_P : \emptyset \to P)$ and $\Sigma' = (E', \delta_{P'} : \emptyset \to P')$, and an ed signature morphism σ from Σ to Σ' whose event part is the identity and whose data part is the inclusion. The following Σ'-edts M' has two configurations being pairs $(\mathsf{c}, \mathsf{p} \wedge \mathsf{q})$ and $(\mathsf{c}, \mathsf{p} \wedge \neg\mathsf{q})$ with the same control state c. The data state $\mathsf{p} \wedge \mathsf{q}$ represents the function $\mu_{\mathsf{p}\wedge\mathsf{q}}$ with $\mu_{\mathsf{p}\wedge\mathsf{q}}(\mathsf{p}) = \mu_{\mathsf{p}\wedge\mathsf{q}}(\mathsf{q}) = \mathit{tt}$ and $\mathsf{p} \wedge \neg\mathsf{q}$ represents the function $\mu_{\mathsf{p}\wedge\neg\mathsf{q}}$ with $\mu_{\mathsf{p}\wedge\neg\mathsf{q}}(\mathsf{p}) = \mathit{tt}$ and $\mu_{\mathsf{p}\wedge\neg\mathsf{q}}(\mathsf{q}) = \mathit{ff}$. The data state labelling $\omega(M')$ is the identity.



The reduct of M' along σ is the following Σ-edts $M'|\sigma$:

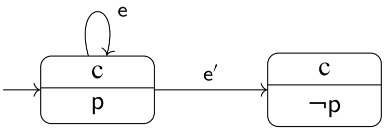


$M'|\sigma$ has the same configurations as M' but the data state labelling $\omega(M'|\sigma)$ restricts both data states $\mathsf{p} \wedge \mathsf{q}$ and $\mathsf{p} \wedge \neg\mathsf{q}$ to p, pictorally represented by shadowing q and $\neg\mathsf{q}$ respectively. Intuitively this means that q is hidden but the data states of the configurations are still different; only their labellings coincide (i.e. $M'|\sigma$ is not extensional). This has the effect that both edts satisfy the Σ-sentence $\langle\mathsf{e}\rangle\mathsf{true} \wedge [\mathsf{e};\ \mathsf{e}]\mathsf{false}$, i.e.,

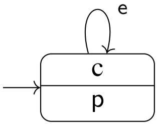
$$M' \models_{\Sigma'}^{\mathcal{E}^{\downarrow}(\mathit{Prop}_{\emptyset})} \langle\mathsf{e}\rangle\mathsf{true} \wedge [\mathsf{e};\ \mathsf{e}]\mathsf{false} \qquad \text{and} \qquad M'|\sigma \models_{\Sigma}^{\mathcal{E}^{\downarrow}(\mathit{Prop}_{\emptyset})} \langle\mathsf{e}\rangle\mathsf{true} \wedge [\mathsf{e};\ \mathsf{e}]\mathsf{false}\ .$$

Without using a data state labelling, $M'|\sigma$ would just have a single configuration (c, p) where the event e is either enabled, leading to a loop, or not. In each case $M'|\sigma$ would not satisfy the above sentence but M' does. Hence, the satisfaction condition would be violated. □

Example 18 This example illustrates why the relativisation of the jump operator is needed to get the satisfaction condition. We work again in the institution $\mathcal{E}^{\downarrow}(\mathit{Prop}_{\emptyset})$. Let $E = \{\mathsf{e}\}$ and $E' = \{\mathsf{e}, \mathsf{e}'\}$ be two sets of events and $P = P' = \{\mathsf{p}\}$. Thus we have two ed signatures $\Sigma = (E, \delta_P : \emptyset \to P)$ and $\Sigma' = (E', \delta_{P'} : \emptyset \to P')$ and an ed signature morphism σ from Σ to Σ' whose event part is the inclusion and whose data part is the identity. Consider the following extensional Σ'-edts M' with just one control state c and with configurations (c, p) and $(\mathsf{c}, \neg\mathsf{p})$:



The σ-reduct of M' has no e'-transition; it is just



Assume that we would use the unrelativised jump operator $(@x)$ in $\varrho = {\downarrow}x\,.\,(@x)\mathsf{p}$ referring to *all* configurations whose control state is bound by x and not only to those reachable by $E = \{e\}$. In M' this would include the configuration $(\mathsf{c}, \neg\mathsf{p})$ which is not reachable in $M'|\sigma$. Thus we have $M' \not\models_{\Sigma'}^{\mathcal{E}^{\downarrow}(\mathit{Prop}_{\emptyset})} \varrho$, but $M'|\sigma \models_{\Sigma}^{\mathcal{E}^{\downarrow}(\mathit{Prop}_{\emptyset})} \varrho$. Thus the satisfaction condition would be violated. Using the relativised jump operator $(@^E x)$ we have, however,

$$M' \models_{\Sigma'}^{\mathcal{E}^{\downarrow}(\mathit{Prop}_{\emptyset})} {\downarrow}x\,.\,(@^E x)\mathsf{p} \qquad \text{and} \qquad M'|\sigma \models_{\Sigma}^{\mathcal{E}^{\downarrow}(\mathit{Prop}_{\emptyset})} {\downarrow}x\,.\,(@^E x)\mathsf{p}\ . \qquad \square$$

Example 19 Continuing Ex. 16, consider again the Σ_1-edts M_1 shown in Fig. 1a and its σ_0-reduct M_0 shown in Fig. 1b. As shown in Ex. 16, M_0 satisfies the sentences (0.1) to (0.3). Hence, by the satisfaction condition for $\mathcal{E}^{\downarrow}(\mathit{Prop}_{\emptyset})$, M_1 satisfies the sentences (0.1) to (0.3) as well (after applying the trivial sentence translation by inclusion). Conversely, one could also show first that M_1 satisfies the sentences (0.1) to (0.3) and deduce, by using the satisfaction condition, that M_0 satisfies (0.1) to (0.3). □

4.6. Institution comorphisms for $\mathcal{E}^{\downarrow}(\vec{\mathcal{D}})$

Let $\nu : \mathcal{D} \to \mathcal{D}'$ be an institution comorphism. We know that it can be lifted to an institution comorphism $\vec{\nu} : \vec{\mathcal{D}} \to \vec{\mathcal{D}}'$ on data state institutions which in turn can be lifted to an institution comorphism $\vec{\nu}^2 : 2\vec{\mathcal{D}} \to 2\vec{\mathcal{D}}'$ on 2-data state institutions. We can then provide a final lifting into an institution comorphism $\vec{\nu}^{\downarrow} : \mathcal{E}^{\downarrow}(\vec{\mathcal{D}}) \to \mathcal{E}^{\downarrow}(\vec{\mathcal{D}}')$ on event/data institutions as follows:

- For event/data signatures, $\vec{\nu}^{\mathbb{S}}$ is applied to the data signature: Define $(\vec{\nu}^{\downarrow})^{\mathbb{S}}(E, \delta) = (E, \vec{\nu}^{\mathbb{S}}(\delta))$ and $(\vec{\nu}^{\downarrow})^{\mathbb{S}}(\eta, \vartheta) = (\eta, \vec{\nu}^{\mathbb{S}}(\vartheta))$.
- For event/data structures, $\vec{\nu}^{Str}$ is applied to the data labels of configurations: Define $(\vec{\nu}^{\downarrow})^{Str}_{\Sigma}(\Gamma', R', \Gamma'_0, \omega') = (\Gamma', R', \Gamma'_0, \omega)$ with $\omega(\gamma') = \vec{\nu}^{Str}_{\delta(\Sigma)}(\omega'(\gamma'))$ for all $\gamma' \in \Gamma'$, and $(\vec{\nu}^{\downarrow})^{Str}_{\Sigma}(h) = h$.
- For event/data sentences, $\vec{\nu}^{\mathrm{Sen}}$ is lifted to event/data actions via $\vec{\nu}^{\Lambda}_{\Sigma} : \Lambda(\Sigma) \to \Lambda((\vec{\nu}^{\downarrow})^{\mathbb{S}}(\Sigma))$ with $\vec{\nu}^{\Lambda}_{\Sigma}(e \mathbin{/\!\!/} \psi) = e \mathbin{/\!\!/} (\vec{\nu}^2)^{\mathrm{Sen}}_{\delta(\Sigma)}(\psi)$ etc., and to event/data formulæ via $\vec{\nu}^{\mathscr{F}}_{\Sigma} : \mathscr{F}(\Sigma) \to \mathscr{F}((\vec{\nu}^{\downarrow})^{\mathbb{S}}(\Sigma))$ with $\vec{\nu}^{\mathscr{F}}_{\Sigma}(\varphi) = \vec{\nu}^{\mathrm{Sen}}_{\delta(\Sigma)}(\varphi)$, $\vec{\nu}^{\mathscr{F}}_{\Sigma}(\langle\lambda\rangle\varrho) = \langle\vec{\nu}^{\Lambda}_{\Sigma}(\lambda)\rangle\vec{\nu}^{\mathscr{F}}_{\Sigma}(\varrho)$, etc., which then defines $(\vec{\nu}^{\downarrow})^{\mathrm{Sen}}_{\Sigma}$ as $\vec{\nu}^{\mathscr{F}}_{\Sigma}$ on sentences.

The satisfaction condition is routinely checked by structural induction on $\mathcal{E}^{\downarrow}(\vec{\mathcal{D}})$-formulæ.

Example 20 We can lift the institution comorphisms $\vec{\nu}_{\mathrm{b}} : \mathit{Prop}_{\emptyset} \to \mathit{Attr}^{\equiv}_{\Sigma_{\mathrm{b}},\mathfrak{A}_{\mathrm{b}}}$ and $\vec{\nu}^2_{\mathrm{b}} : 2\mathit{Prop}_{\emptyset} \to 2\mathit{Attr}^{\equiv}_{\Sigma_{\mathrm{b}},\mathfrak{A}_{\mathrm{b}}}$ for data state institutions in Ex. 6 and Ex. 8 to an institution comorphism $\vec{\nu}^{\downarrow}_{\mathrm{b}} : \mathcal{E}^{\downarrow}(\mathit{Prop}_{\emptyset}) \to \mathcal{E}^{\downarrow}(\mathit{Attr}^{\equiv}_{\Sigma_{\mathrm{b}},\mathfrak{A}_{\mathrm{b}}})$ which changes the underlying data state institution of our event/data logic by moving from propositional logic data states to attribute-based data states. □

5. Specifications of event/data-based systems

For specifying event/data-based systems in $\mathcal{E}^{\downarrow}(\vec{\mathcal{D}})$ we consider two specification styles: An axiomatic specification uses sentences as axioms to express requirements. This textual style is complemented by operational specifications with a graphical representation. Operational specifications also offer a (syntactic) parallel composition operator. We show that finitary operational specifications can be axiomatised such that we do not leave $\mathcal{E}^{\downarrow}(\vec{\mathcal{D}})$-logic.

5.1. Axiomatic specifications of event/data-based systems

Sentences of any $\mathcal{E}^{\downarrow}(\vec{\mathcal{D}})$-logic can be used to specify properties of event/data-based systems and thus to write system specifications in an axiomatic way.

Definition 10 An *axiomatic specification* $Sp = (\Sigma, Ax)$ in $\mathcal{E}^{\downarrow}(\vec{\mathcal{D}})$ consists of a signature $\Sigma \in |\, \mathbb{S}^{\mathcal{E}^{\downarrow}(\vec{\mathcal{D}})} \,|$ and a set of *axioms* $Ax \subseteq \mathrm{Sen}^{\mathcal{E}^{\downarrow}(\vec{\mathcal{D}})}(\Sigma)$. We write $\Sigma(Sp)$ for Σ and $Ax(Sp)$ for Ax.

The *semantics of Sp* is given by the pair $(\Sigma(Sp), \mathrm{Mod}^{\mathcal{E}^{\downarrow}(\vec{\mathcal{D}})}(Sp))$ where

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

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

Example 21 We continue with the ATM example considered from Ex. 11 onward. Now we provide a first axiomatic specification that will be gradually extended and refined later on in Sect. 6.

Our first specification is $Sp_0 = (\Sigma_0, Ax_0)$, where $\Sigma_0 = (E_0, \delta_P)$ and Ax_0 requires the properties (0.1–0.3) described in Ex. 15. The Σ_0-edts M_0 shown in Fig. 1b is a model of Sp_0; it satisfies the required axioms as demonstrated in Ex. 16. □

5.2. Operational specifications

Operational specifications are introduced as a means to specify in a constructive style the properties of event/data-based systems.

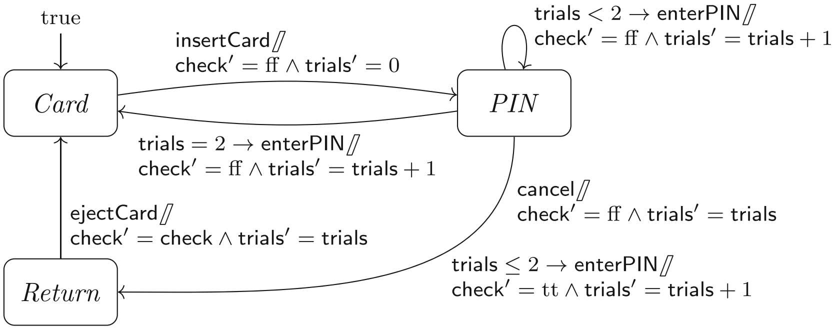

true
insertCard //
check′ = ff ∧ trials′ = 0
trials < 2 → enterPIN //
check′ = ff ∧ trials′ = trials + 1
Card
PIN
trials = 2 → enterPIN //
check′ = ff ∧ trials′ = trials + 1
cancel //
check′ = ff ∧ trials′ = trials
ejectCard //
check′ = check ∧ trials′ = trials
trials ≤ 2 → enterPIN //
check′ = tt ∧ trials′ = trials + 1
Return

Fig. 2. Operational specification *ATM*

They are not appropriate for writing abstract requirements for which axiomatic specifications should be used. Though $\mathcal{E}^{\downarrow}(\vec{\mathcal{D}})$-logic is able to specify concrete process structures as well, cf. Sect. 4.3, operational specifications provide a convenient representation of desired behaviours while undesired behaviours are automatically excluded and need not be explicitly formalised like in the declarative, axiomatic specification style. Operational specifications allow a graphic representation close to well-known formalisms in the literature, like UML protocol state machines, cf. [OMG17, KMRG15]. Nevertheless, as will be shown in Sect. 5.3, finite operational specifications can be characterised by a sentence in $\mathcal{E}^{\downarrow}(\vec{\mathcal{D}})$-logic. Therefore, $\mathcal{E}^{\downarrow}(\vec{\mathcal{D}})$-logic is still the common basis of our development approach.

Transitions in an operational specification are tuples $(c, \varphi, e, \psi, c')$ with c a source control state, φ a precondition, e an event, ψ a state transition predicate specifying the possible effects of the event e, and c' a target control state. In the semantic models an event must be enabled whenever the respective source data state satisfies the precondition (condition (1) in Def. 11). Thus isolating preconditions has a semantic consequence that is not expressible by transition predicates only. The effect of the event must respect ψ; no other transitions are allowed, i.e., any semantic transition must be justified by a syntactic one (condition (2) in Def. 11).

Definition 11 An *operational specification* $O = (\Sigma, C, T, (c_0, \varphi_0))$ in $\mathcal{E}^{\downarrow}(\vec{\mathcal{D}})$ is given by a signature $\Sigma \in |\mathbb{S}^{\mathcal{E}^{\downarrow}(\vec{\mathcal{D}})}|$, a set of *control states* C, a *transition relation specification* $T \subseteq C \times \mathrm{Sen}^{\vec{\mathcal{D}}}(\delta(\Sigma)) \times E(\Sigma) \times \mathrm{Sen}^{2\vec{\mathcal{D}}}(\delta(\Sigma)) \times C$, an *initial control state* $c_0 \in C$, and an *initial state predicate* $\varphi_0 \in \mathrm{Sen}^{\vec{\mathcal{D}}}(\delta(\Sigma))$, such that C is *syntactically reachable*, i.e., for every $c \in C \setminus \{c_0\}$ there are $(c_0, \varphi_1, e_1, \psi_1, c_1), \ldots, (c_{n-1}, \varphi_n, e_n, \psi_n, c_n) \in T$ with $n > 0$ such that $c_n = c$. We write $\Sigma(O)$ for Σ, etc.

A Σ-edts M is a *model* of O if $C \subseteq C(M)$ up to a bijective renaming, $c_0(M) = c_0$, $\Omega_0(M) \subseteq \{\omega \in |Str^{\vec{\mathcal{D}}}(\delta(\Sigma))| \mid \omega \models^{\vec{\mathcal{D}}}_{\delta(\Sigma)} \varphi_0\}$, and if the following conditions hold for all $(c, d) \in \Gamma(M)$:

1. for all $(c, \varphi, e, \psi, c') \in T$ with $\omega(M)(d) \models^{\vec{\mathcal{D}}}_{\delta(\Sigma)} \varphi$, there is some $((c, d), (c', d')) \in R(M)_e$ with $(\omega(M)(d), \omega(M)(d')) \models^{2\vec{\mathcal{D}}}_{\delta(\Sigma)} \psi$;
2. for all $((c, d), (c', d')) \in R(M)_e$ there is some $(c, \varphi, e, \psi, c') \in T$ with $\omega(M)(d) \models^{\vec{\mathcal{D}}}_{\delta(\Sigma)} \varphi$ and $(\omega(M)(d), \omega(M)(d')) \models^{2\vec{\mathcal{D}}}_{\delta(\Sigma)} \psi$.

The class of all models of O is denoted by $\mathrm{Mod}^{\mathcal{E}^{\downarrow}(\vec{\mathcal{D}})}(O)$. The *semantics* of O is given by the pair $(\Sigma(O), \mathrm{Mod}^{\mathcal{E}^{\downarrow}(\vec{\mathcal{D}})}(O))$ where $\Sigma(O) = \Sigma$. □

Example 22 We construct an operational specification ATM for the ATM example using $Attr^{=}_{\Sigma_b, \mathfrak{A}_b}$ as data state institution; see Ex. 5(c). ATM will reappear in Sect. 6 in a refinement chain for the implementation of Sp_0. The ed signature of ATM is $\Sigma_{ATM} = (E_1, \iota : \Sigma_b \hookrightarrow \Sigma_A)$ where $E_1 = \{\mathsf{insertCard}, \mathsf{enterPIN}, \mathsf{ejectCard}, \mathsf{cancel}\}$, Σ_b is the base signature of $Attr^{=}_{\Sigma_b, \mathfrak{A}_b}$, and Σ_A is the attribute signature induced by the set of attributes $A = \{\mathsf{check} : \mathsf{Bool}, \mathsf{trials} : \mathsf{Int}\}$. The integer-valued attribute trials is used to count the number of the attempts to enter a correct PIN (with the same card). Specification ATM is graphically presented in Fig. 2. The initial control state is $Card$ and the initial state predicate is true. If no precondition is explicitly indicated, it is true. □

Operational specifications can be composed by a syntactic parallel composition operator which synchronises shared events. Two ed signatures Σ_1 and Σ_2 are *composable* if $\delta(\Sigma_1) : \Delta_0 \to \Delta_1$ and $\delta(\Sigma_2) : \Delta_0 \to \Delta_2$ (for the same Δ_0); their parallel composition is given by $\Sigma_1 \otimes \Sigma_2$ with $E(\Sigma_1 \otimes \Sigma_2) = E(\Sigma_1) \cup E(\Sigma_2)$ and $\delta(\Sigma_1 \otimes \Sigma_2) : \Delta_0 \to \Delta_1 +^{\delta(\Sigma_1),\delta(\Sigma_2)}_{\Delta_0} \Delta_2$ defined by the pushout diagram

$$\begin{tikzcd}
& \Delta_1 +^{\delta(\Sigma_1),\delta(\Sigma_2)}_{\Delta_0} \Delta_2 & \\
\Delta_1 \arrow[ur, "\hat{\delta}(\Sigma_1)"] & & \Delta_2 \arrow[ul, "\hat{\delta}(\Sigma_2)"'] \\
& \Delta_0 \arrow[ul, "\delta(\Sigma_1)"] \arrow[ur, "\delta(\Sigma_2)"'] \arrow[uu, dashed, "\delta(\Sigma_1 \otimes \Sigma_2)"] &
\end{tikzcd}$$

Let us abbreviate $\Delta_1 +^{\delta(\Sigma_1),\delta(\Sigma_2)}_{\Delta_0} \Delta_2$ by Δ. A state predicate $\varphi_i \in \mathrm{Sen}^{\vec{\mathcal{D}}}(\delta(\Sigma_i)) = \mathrm{Sen}^{\mathcal{D}}(\Delta_i)$ can be considered as state predicate $\mathrm{Sen}^{\mathcal{D}}(\hat{\delta}(\Sigma_i))(\varphi_i) \in \mathrm{Sen}^{\vec{\mathcal{D}}}(\delta(\Sigma_1 \otimes \Sigma_2)) = \mathrm{Sen}^{\mathcal{D}}(\Delta)$ for $1 \le i \le 2$. Using the satisfaction condition for $\mathcal{D}$ it holds that

$$M \models^{\mathcal{D}}_{\Delta} \mathrm{Sen}^{\mathcal{D}}(\hat{\delta}(\Sigma_i))(\varphi_i) \iff \mathit{Str}^{\mathcal{D}}(\hat{\delta}(\Sigma_i)) \models^{\mathcal{D}}_{\Delta_i} \varphi_i \,.$$

For a similar construction for transition predicates $\psi_i \in \mathrm{Sen}^{2\vec{\mathcal{D}}}(\delta(\Sigma_i))$ we note that by the uniqueness of pushouts in $\mathcal{D}$ up to isomorphism, there are uniquely determined $\mathcal{D}$-signature morphisms

$$2\hat{\delta}(\Sigma_i) : \Delta_i +^{\delta(\Sigma_i)}_{\Delta_0} \Delta_i \to \Delta +^{\delta(\Sigma_1 \otimes \Sigma_2)}_{\Delta_0} \Delta \quad \text{for } 1 \le i \le 2.$$

With these morphisms a transition predicate $\psi_i \in \mathrm{Sen}^{2\vec{\mathcal{D}}}(\delta(\Sigma_i)) = \mathrm{Sen}^{\mathcal{D}}(\Delta_i +^{\delta(\Sigma_i)}_{\Delta_0} \Delta_i)$ can be considered as transition predicate $\mathrm{Sen}^{\mathcal{D}}(2\hat{\delta}(\Sigma_i))(\psi_i) \in \mathrm{Sen}^{2\vec{\mathcal{D}}}(\delta(\Sigma_1 \otimes \Sigma_2)) = \mathrm{Sen}^{\mathcal{D}}(\Delta +^{\delta(\Sigma_1 \otimes \Sigma_2)}_{\Delta_0} \Delta)$ for $1 \le i \le 2$. We abbreviate $\mathrm{Sen}^{\mathcal{D}}(\hat{\delta}(\Sigma_i))(\varphi_i)$ by $\hat{\varphi}_i$ and $\mathrm{Sen}^{\mathcal{D}}(2\hat{\delta}(\Sigma_i))(\psi_i)$ by $\hat{\psi}_i$.

Moreover, let $\mathrm{id}_{\Delta(\Sigma_i)}$ denote the invariance sentence $\mathrm{id}_{(\delta(\Sigma_i),\hat{\delta}(\Sigma_i))} \in \mathrm{Sen}^{2\vec{\mathcal{D}}}(\delta(\Sigma_1 \otimes \Sigma_2))$ for $1 \le i \le 2$ which requires that composite states do not change on the Δ_i part.

Definition 12 Let O_1 and O_2 be operational specifications such that $\Sigma(O_1)$ and $\Sigma(O_2)$ are composable. The *parallel composition* of O_1 and O_2 is given by the operational specification $O_1 \parallel O_2 = (\Sigma(O_1) \otimes \Sigma(O_2), C, T, (c_0, \varphi_0))$ with $c_0 = (c_0(O_1), c_0(O_2))$, $\varphi_0 = \hat{\varphi}_{0,1} \wedge \hat{\varphi}_{0,2}$ for $\varphi_{0,1} = \varphi_0(O_1)$, $\varphi_{0,2} = \varphi_0(O_2)$, and C and T are inductively defined by $c_0 \in C$ and

- for $e_1 \in E(\Sigma_1) \setminus E(\Sigma_2)$, $c_1, c_1' \in C(O_1)$, and $c_2 \in C(O_2)$, if $(c_1, c_2) \in C$ and $(c_1, \varphi_1, e_1, \psi_1, c_1') \in T(O_1)$, then $(c_1', c_2) \in C$ and $((c_1, c_2), \hat{\varphi}_1, e_1, \hat{\psi}_1 \wedge \mathrm{id}_{\Delta(\Sigma(O_2))}, (c_1', c_2)) \in T$;
- for $e_2 \in E(\Sigma_2) \setminus E(\Sigma_1)$, $c_2, c_2' \in C(O_2)$, and $c_1 \in C(O_1)$, if $(c_1, c_2) \in C$ and $(c_2, \varphi_2, e_2, \psi_2, c_2') \in T(O_2)$, then $(c_1, c_2') \in C$ and $((c_1, c_2), \hat{\varphi}_2, e_2, \hat{\psi}_2 \wedge \mathrm{id}_{\Delta(\Sigma(O_1))}, (c_1, c_2')) \in T$;
- for $e \in E(\Sigma_1) \cap E(\Sigma_2)$, $c_1, c_1' \in C(O_1)$, and $c_2, c_2' \in C(O_2)$, if $(c_1, c_2) \in C$, $(c_1, \varphi_1, e, \psi_1, c_1') \in T(O_1)$, and $(c_2, \varphi_2, e, \psi_2, c_2') \in T(O_2)$, then $(c_1', c_2') \in C$ and $((c_1, c_2), \hat{\varphi}_1 \wedge \hat{\varphi}_2, e, \hat{\psi}_1 \wedge \hat{\psi}_2, (c_1', c_2')) \in T$. □

Note that joint moves with e cannot become inconsistent due to composability of ed signatures. An example for parallel composition of operational specifications is shown in Fig. 4.

5.3. Expressiveness of $\mathcal{E}^{\downarrow}(\vec{\mathcal{D}})$-logic

We show that the semantics of an operational specification O with finitely many control states can be characterised by a single $\mathcal{E}^{\downarrow}(\vec{\mathcal{D}})$-sentence ϱ_O, i.e., an edts M is a model of O iff $M \models^{\mathcal{E}^{\downarrow}(\vec{\mathcal{D}})}_{\Sigma(O)} \varrho_O$. Using Alg. 1, such a characterising sentence is

$$\varrho_O = {\downarrow} c_0 \,.\, \varphi_0 \wedge \mathrm{sen}(c_0, \mathit{Im}_O(c_0), C(O), \{c_0\}) \,,$$

where $c_0 = c_0(O)$ and $\varphi_0 = \varphi_0(O)$. The algorithm follows closely the procedure in [MBHM18] for characterising a finite structure by a sentence of $\mathcal{D}^{\downarrow}$-logic.

Algorithm 1 Constructing a sentence from an operational specification

```
Require: $O \equiv$ finite operational specification
         $Im_O(c) = \{(\varphi, e, \psi, c') \mid (c, \varphi, e, \psi, c') \in T(O)\}$ for $c \in C(O)$
         $Im_O(c, e) = \{(\varphi, \psi, c') \mid (c, \varphi, e, \psi, c') \in T(O)\}$ for $c \in C(O)$, $e \in E(\Sigma(O))$
function sen($c, I, V, B$)        ▷ $c$: state, $I$: image to visit, $V$: states to visit, $B$: bound states
    if $I \neq \emptyset$ then
        $(\varphi, e, \psi, c') \leftarrow$ choose $I$
        if $c' \in B$ then
            return $(@c)\varphi \rightarrow \langle e /\!\!/ \psi\rangle(c' \wedge \mathrm{sen}(c, I \setminus \{(\varphi, e, \psi, c')\}, V, B))$
        else
            return $(@c)\varphi \rightarrow \langle e /\!\!/ \psi\rangle(\downarrow c' \,.\, \mathrm{sen}(c, I \setminus \{(\varphi, e, \psi, c')\}, V, B \cup \{c'\}))$
    else
        $V \leftarrow V \setminus \{c\}$
        if $V \neq \emptyset$ then
            $c' \leftarrow$ choose $B \cap V$
            return $\mathrm{fin}(c) \wedge \mathrm{sen}(c', Im_O(c'), V, B)$
        else
            return $\mathrm{fin}(c) \wedge \bigwedge_{c_1 \in C(O), c_2 \in C(O) \setminus \{c_1\}} \neg(@c_1)c_2$
function fin($c$)
    return $(@c)\bigwedge_{e \in E(\Sigma(O))} \bigwedge_{P \subseteq Im_O(c,e)}$
        $[e /\!\!/ \big(\bigwedge_{(\varphi,\psi,c') \in P}(\varphi \wedge \psi)\big) \wedge \neg\big(\bigvee_{(\varphi,\psi,c') \in Im_O(c,e) \setminus P}(\varphi \wedge \psi)\big)]\big(\bigvee_{(\varphi,\psi,c') \in P} c'\big)$
```

A call $\mathrm{sen}(c, I, V, B)$ performs a recursive breadth-first traversal through the operational specification O starting from c, where I holds the unprocessed quadruples (φ, e, ψ, c') of transitions outgoing from c, V the remaining states to visit, and B the set of already bound states. The function first requires the existence of each outgoing transition of I, provided its precondition holds, in the resulting formula, binding any newly reached state. Then it requires that no other transitions with source state c exist using calls to fin. Having visited all states in V, it finally adds the requirement that all states in $C(O)$ are pairwise different.

It is $\mathrm{fin}(c)$ where this algorithm mainly deviates from [MBHM18]: In order to ensure that no other transitions from c exist than those specified in O, $\mathrm{fin}(c)$ produces the requirement that at state c, for every event e and for every subset P of the transitions outgoing from c, whenever an e-transition can be done with the combined effect of P but not adhering to any of the effects of the currently not selected transitions, the e-transition must have one of the states as its target that are target states of P. The rather complicated formulation is due to possibly overlapping preconditions where for a single event e the preconditions of two different transitions may be satisfied simultaneously. For a state c, where all outgoing transitions for the same event have (semantically) disjoint preconditions, the $\mathcal{E}^{\downarrow}(\vec{\mathcal{D}})$-formula returned by $\mathrm{fin}(c)$ is equivalent to

$$(@c) \bigwedge_{e \in E(\Sigma(O))} \bigwedge_{(\varphi,\psi,c') \in Im_O(c,e)} [e /\!\!/ \varphi \wedge \psi] c' \wedge \\ [e /\!\!/ \neg\big(\bigvee_{(\varphi,\psi,c') \in Im_O(c,e)} (\varphi \wedge \psi)\big)]\mathrm{false}\ .$$

In both cases the actions involve a conjunction of a state predicate φ and a transition predicate ψ that indeed can be combined faithfully using Prop. 1.

Example 23 We show the first few steps of representing the operational specification ATM of Fig. 2 as an $\mathcal{E}^{\downarrow}(\vec{\mathcal{D}})$-sentence ϱ_{ATM}. This top-level sentence is

$$\downarrow Card \,.\, \mathrm{true} \wedge \mathrm{sen}(Card, \{(\mathrm{true}, \mathsf{insertCard}, \mathsf{check}' \\ = \mathsf{ff} \wedge \mathsf{trials}' = 0, \mathit{PIN})\}, \{Card, \mathit{PIN}, Return\}, \{Card\})\ .$$

The first call of $\mathrm{sen}(Card, \ldots)$ explores the single outgoing transition from $Card$ to PIN, adds PIN to the bound states, and hence expands to

$$(@Card)\mathrm{true} \rightarrow \langle \mathsf{insertCard} /\!\!/ \mathsf{check}' = \mathsf{ff} \wedge \mathsf{trials}' = 0\rangle \downarrow \mathit{PIN}\,. \\ \mathrm{sen}(Card, \emptyset, \{Card, \mathit{PIN}, Return\}, \{Card, \mathit{PIN}\})\ .$$

Now all outgoing transitions from Card have been explored and the next call of $\mathrm{sen}(\mathit{Card}, \emptyset, \ldots)$ removes Card from the set of states to be visited, resulting in

$$\begin{aligned}&\mathrm{fin}(\mathit{Card}) \wedge \mathrm{sen}(\mathit{PIN}, \{(\mathsf{trials} < 2, \mathsf{enterPIN}, \ldots), (\mathsf{trials} = 2, \mathsf{enterPIN}, \ldots),\\ &\qquad\qquad\qquad\qquad (\mathsf{trials} \leq 2, \mathsf{enterPIN}, \ldots), (\mathsf{true}, \mathsf{cancel}, \ldots)\},\\ &\qquad\qquad \{\mathit{PIN}, \mathit{Return}\}, \{\mathit{Card}, \mathit{PIN}\})\ .\end{aligned}$$

As there is only a single outgoing transition from Card, the special case of disjoint preconditions applies for the finalisation call, and

$$\begin{aligned}&(@\mathit{Card})[\mathsf{insertCard}/\!\!/\mathsf{check}' = \mathsf{ff} \wedge \mathsf{trials}' = 0]\mathit{PIN} \wedge\\ &\qquad [\mathsf{insertCard}/\!\!/\mathsf{check}' = \mathsf{tt} \vee \mathsf{trials}' \neq 0]\mathsf{false} \wedge\\ &\qquad [\mathsf{enterPIN}/\!\!/\mathsf{true}]\mathsf{false} \wedge [\mathsf{cancel}/\!\!/\mathsf{true}]\mathsf{false} \wedge [\mathsf{ejectCard}/\!\!/\mathsf{true}]\mathsf{false}\end{aligned}$$

is the result of $\mathrm{fin}(\mathit{Card})$. □

6. Constructor implementations and refinement in $\mathcal{E}^{\downarrow}(\vec{\mathcal{D}})$

The semantics of a specification is loose and allows usually several different models for its realisation. A refinement step is therefore understood as a restriction of the model class of an abstract specification. Following the terminology of Sannella and Tarlecki [ST88, ST12], we will call a specification which refines another one an *implementation*. Formally, a specification Sp' is a *simple implementation* of a specification Sp over the same signature, in symbols $\mathit{Sp} \rightsquigarrow \mathit{Sp}'$, whenever $\Sigma(\mathit{Sp}) = \Sigma(\mathit{Sp}')$ and $\mathrm{Mod}(\mathit{Sp}) \supseteq \mathrm{Mod}(\mathit{Sp}')$. This implementation notion is, however, too simple for many practical applications. It requires the same signature for specification and implementation and does not support the process of constructing an implementation. Therefore, Sannella and Tarlecki have proposed the notion of constructor implementation which is a generic notion applicable to specification formalisms which are based on institutions. We will reuse the ideas in the context of $\mathcal{E}^{\downarrow}(\vec{\mathcal{D}})$-logic, of course staying generic w.r.t. the underlying data institution.

6.1. Constructor implementations

We apply the notion of a constructor implementation [ST88, ST12] to $\mathcal{E}^{\downarrow}(\vec{\mathcal{D}})$-logic and extend it to take into account a change of institutions. For signatures $\Sigma_1, \ldots, \Sigma_n, \Sigma \in |\ \mathbb{S}^{\mathcal{E}^{\downarrow}(\vec{\mathcal{D}})}\ |$, a *constructor* κ from $(\Sigma_1, \ldots, \Sigma_n)$ to Σ is a (total) function $\kappa : |\ \mathit{Str}^{\mathcal{E}^{\downarrow}(\vec{\mathcal{D}})}(\Sigma_1)\ | \times \ldots \times |\ \mathit{Str}^{\mathcal{E}^{\downarrow}(\vec{\mathcal{D}})}(\Sigma_n)\ | \to |\ \mathit{Str}^{\mathcal{E}^{\downarrow}(\vec{\mathcal{D}})}(\Sigma)\ |$. Given a constructor κ from $(\Sigma_1, \ldots, \Sigma_n)$ to Σ and a set of constructors κ_i from $(\Sigma_i^1, \ldots, \Sigma_i^{k_i})$ to Σ_i, $1 \leq i \leq n$, the constructor $(\kappa_1, \ldots, \kappa_n);\ \kappa$ from $(\Sigma_1^1, \ldots, \Sigma_1^{k_1}, \ldots, \Sigma_n^1, \ldots, \Sigma_n^{k_n})$ to Σ is obtained by the usual composition of functions. The following definitions apply to both axiomatic and operational specifications since the semantics of both is given in terms of ed signatures and model classes of edts. In particular, the implementation notion allows to implement axiomatic specifications by operational specifications.

Definition 13 Given specifications $\mathit{Sp}, \mathit{Sp}_1, \ldots, \mathit{Sp}_n$ and a constructor κ from $(\Sigma(\mathit{Sp}_1), \ldots, \Sigma(\mathit{Sp}_n))$ to $\Sigma(\mathit{Sp})$, the tuple $\langle \mathit{Sp}_1, \ldots, \mathit{Sp}_n \rangle$ is a *constructor implementation via* κ of Sp, in symbols $\mathit{Sp} \rightsquigarrow_{\kappa} \langle \mathit{Sp}_1, \ldots, \mathit{Sp}_n \rangle$, if for all $M_i \in \mathrm{Mod}^{\mathcal{E}^{\downarrow}(\vec{\mathcal{D}})}(\mathit{Sp}_i)$ we have $\kappa(M_1, \ldots, M_n) \in \mathrm{Mod}^{\mathcal{E}^{\downarrow}(\vec{\mathcal{D}})}(\mathit{Sp})$. The implementation involves a *decomposition* if $n > 1$. □

It may sound strange to call the case $n > 1$ a decomposition since κ composes structures. But indeed the idea is to split the implementation of a specification into parts (the decomposition) and then to justify that the implementation is correct by showing that those parts composed by the n-ary constructor κ satisfy the requirements of the original specification. In the sequel all constructors apart from parallel composition will have arity $n = 1$. The notion of simple implementation is also captured by the above definition if we choose the identity constructor.

Using an institutional approach in system development has also the advantage that one can work in heterogeneous logical environments where several institutions may be used depending on different kinds of properties, views, and aspects to be expressed by the formalism. This can even be supported by tools, like the heterogeneous tool set HETS [MML07]. A particularly relevant case is the change of institution when moving from a higher abstraction level to a more concrete one. The need for such a change may be motivated by the need of richer expressiveness when moving towards an implementation. For that purpose we will use institution comorphisms; cf.

Sect. 2. The next definition incorporates institution comorphisms in the definition of constructor implementation. It is a particular application of the notion of a generalised constructor in [ST12].

Definition 14 Let $\vec{\nu}^{\downarrow} : \mathcal{E}^{\downarrow}(\vec{\mathcal{D}}) \to \mathcal{E}^{\downarrow}(\vec{\mathcal{D}}')$ be a lifted comorphism between event/data institutions; see Sect. 4.6. Given a specification Sp over $\mathcal{E}^{\downarrow}(\vec{\mathcal{D}})$, specifications $Sp_1, \dots, Sp_n$ over $\mathcal{E}^{\downarrow}(\vec{\mathcal{D}}')$ and a constructor κ from $(\Sigma(Sp_1), \dots, \Sigma(Sp_n))$ to $(\vec{\nu}^{\downarrow})^{\mathbb{S}}(\Sigma(Sp))$, the tuple $\langle Sp_1, \dots, Sp_n\rangle$ is a *constructor implementation via* κ *and* $\vec{\nu}^{\downarrow}$ of Sp, in symbols $Sp \rightsquigarrow_{\kappa}^{\vec{\nu}^{\downarrow}} \langle Sp_1, \dots, Sp_n\rangle$, if for all $M_i' \in \mathrm{Mod}^{\mathcal{E}^{\downarrow}(\vec{\mathcal{D}}')}(Sp_i)$ we have $(\vec{\nu}^{\downarrow})^{Str}_{\Sigma(Sp)}(\kappa(M_1', \dots, M_n')) \in \mathrm{Mod}^{\mathcal{E}^{\downarrow}(\vec{\mathcal{D}})}(Sp)$. The implementation involves a *decomposition* if $n > 1$. □

We now introduce a set of constructors in the context of $\mathcal{E}^{\downarrow}(\vec{\mathcal{D}})$-signatures and edts. The constructors will be illustrated by a refinement chain of implementation constructions starting with the abstract ATM specification Sp_0 in Ex. 21. The first three specifications are developed in $\mathcal{E}^{\downarrow}(Prop_{\emptyset})$. Then we change the data state institution and continue with $\mathcal{E}^{\downarrow}(Attr^{=}_{\Sigma_{\flat},\mathfrak{A}_{\flat}})$. From ATM onwards we use operational specifications:

$$Sp_0 \rightsquigarrow_{\kappa_{\sigma_0}} Sp_1 \rightsquigarrow Sp_2 \rightsquigarrow_{\kappa_\sigma}^{\vec{\nu}_{\flat}^{\downarrow}} ATM \rightsquigarrow_{\kappa_{\otimes};\ \kappa_{\alpha}} \langle ATM', CC\rangle \ .$$

6.2. Reduct constructors and institution change

Reduct constructors allow us to move from one signature to another one via a signature morphism σ. The refinement is correct if the σ-reducts of the models of the more concrete specification are models of the abstract specification. Depending on the form of σ a variety of implementation constructions can be expressed. If σ is bijective we obtain a one-to-one renaming; if σ is just injective the target signature is larger and the reduct, going in the converse direction, hides the added details when constructing abstract models from concrete ones. This kind of semantic constructions going in the converse direction led to the name "constructor implementation".

Definition 15 Let $\sigma : \Sigma \to \Sigma'$ be an ed signature morphism. The *reduct constructor* κ_σ from Σ' to Σ maps any $M' \in\mid Str^{\mathcal{E}^{\downarrow}(\vec{\mathcal{D}})}(\Sigma')\mid$ to its reduct $\kappa_\sigma(M') = M'|\sigma$. Whenever $E(\sigma)$ is a bijective function, κ_σ is an *(event) relabelling constructor*; if $E(\sigma)$ is injective, κ_σ is an *(event) restriction constructor*. □

The following characterisation of implementation correctness for reduct constructors is a direct consequence of the satisfaction condition of $\mathcal{E}^{\downarrow}(\vec{\mathcal{D}})$. It shows that for implementing an axiomatic specification Sp via a reduct constructor it is sufficient to check that the (syntactically translated) axioms of Sp hold in the concrete specification.

Theorem 3 *Let* $Sp = (\Sigma, Ax)$ *be an axiomatic specification,* Sp' *a specification, and* κ_σ *from* $\Sigma(Sp')$ *to* Σ *a reduct constructor via* $\sigma : \Sigma \to \Sigma(Sp')$*. Then,* $Sp \rightsquigarrow_{\kappa_\sigma} Sp'$ *if, and only if,* $M' \models^{\mathcal{E}^{\downarrow}(\vec{\mathcal{D}})}_{\Sigma(Sp')} \mathrm{Sen}^{\mathcal{E}^{\downarrow}(\vec{\mathcal{D}})}(\sigma)(\varrho)$ *for all* $M' \in \mathrm{Mod}^{\mathcal{E}^{\downarrow}(\vec{\mathcal{D}})}(Sp')$ *and all* $\varrho \in Ax$*.* □

Example 24 Consider the specification Sp_0 from Ex. 21. We provide a refinement of this specification by adding the possibility to cancel an ATM transaction. For this purpose we use a specification $Sp_1 = (\Sigma_1, Ax_1)$ with the ed signature $\Sigma_1 = (E_1, \delta_P)$ containing the cancel event; cf. Ex. 11. The axioms of Sp_1 are the following sentences, which are similar to the axioms of Sp_0 but take into account cancel.

(1.1) "Whenever a card has been inserted, a correct PIN can be entered and also the transaction can be cancelled."

$$[\boldsymbol{E}_1^*;\ \mathsf{insertCard}](\langle \mathsf{enterPIN} /\!\!/ \mathsf{check}'\rangle \mathsf{true} \wedge \langle \mathsf{cancel}\rangle \mathsf{true})$$

(1.2) "Whenever either a correct PIN has been entered or the transaction has been cancelled, the card can be ejected."

$$[\boldsymbol{E}_1^*;\ (\mathsf{enterPIN} /\!\!/ \mathsf{check}') + \mathsf{cancel}]\langle \mathsf{ejectCard}\rangle \mathsf{true}$$

(1.3) "A card cannot be ejected if it was not inserted before."

$$[(-\mathsf{insertCard})^*;\ \mathsf{ejectCard}]\mathsf{false}$$

The Σ_1-edts M_1 shown in Fig. 1a is a model of Sp_1. To formally justify that specification Sp_1 is a correct implementation of Sp_0 we use the reduct constructor κ_{σ_0} associated to the ed signature (inclusion) morphism $\sigma_0 : \Sigma_0 \to \Sigma_1$ described in Ex. 11. In fact, κ_{σ_0} is an event restriction constructor.

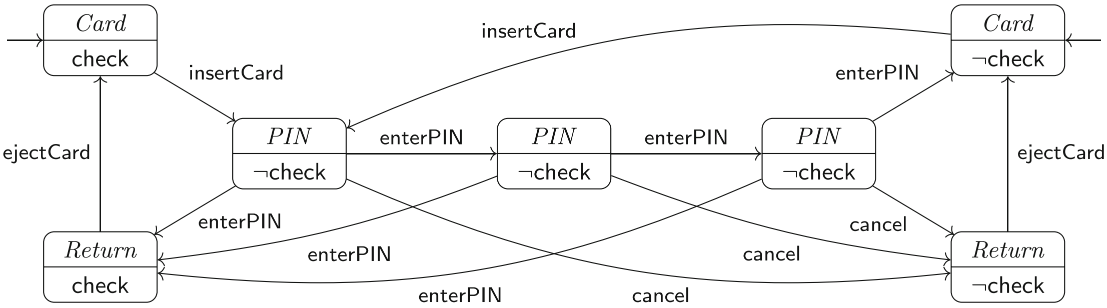


Fig. 3. A model M_2 of Sp_2

To check that $Sp_0 \rightsquigarrow_{\kappa_{\sigma_0}} Sp_1$ is a constructor implementation it is sufficient, by Thm. 3, to show that for all $M_1 \in \mathrm{Mod}^{\mathcal{E}^{\downarrow}(Prop_\emptyset)}(Sp_1)$ it holds that M_1 satisfies the axioms (0.1–0.3) of Sp_0. Axiom (0.1) is an obvious consequence of axiom (1.1) of Sp_1 since the latter strengthens (0.1). Similarly, axioms (1.2) and (1.3) of Sp_1 strengthen (0.2) and (0.3) respectively. Note that in the axioms of Sp_1, $\boldsymbol{E_1}$ ranges over all events of Σ_1 and hence includes cancel. The complex action $-$insertCard in (1.3) ranges over all events of Σ_1 except insertCard. □

Example 25 In a second refinement step we provide a simple implementation $Sp_1 \rightsquigarrow Sp_2$ such that the specification $Sp_2 = (\Sigma_1, Ax_2)$ has the same signature as Sp_1. The axioms of Sp_2 are the sentences (2.1–2.4) below which have already a constructive flavour. Axioms (2.1) to (2.3) specify that *only* the desired behaviour should happen. Axioms (2.3) and (2.4) use binders and state variables from hybrid logic to specify a loop. Axiom (2.4) deals, additionally to the previous specifications, with the situation when an incorrect PIN has been entered too often.

(2.1) "At the beginning and whenever the control state of the beginning is reached, a card can be inserted with the effect that check is false, and nothing else is possible."

$$\downarrow x_0 \,.\, (@x_0)\langle \mathsf{insertCard} /\!\!/ \neg\mathsf{check}'\rangle \mathsf{true} \wedge [\mathsf{insertCard} /\!\!/ \mathsf{check}']\mathsf{false} \wedge [-\mathsf{insertCard}]\mathsf{false}$$

"Whenever the control state of the beginning is reached" is expressed by the jump operator $@x_0$.

(2.2) "Whenever a card has been inserted, a correct and an incorrect PIN can be entered and also the transaction can be cancelled; but nothing else."

$$[\boldsymbol{E}_1^*;\ \mathsf{insertCard}](\langle \mathsf{enterPIN} /\!\!/ \mathsf{check}'\rangle \mathsf{true} \wedge \langle \mathsf{enterPIN} /\!\!/ \neg\mathsf{check}'\rangle \mathsf{true} \wedge \langle \mathsf{cancel}\rangle \mathsf{true} \wedge [-\{\mathsf{enterPIN}, \mathsf{cancel}\}]\mathsf{false})$$

(2.3) "Whenever either a correct PIN has been entered or the transaction has been cancelled, the card can be ejected and the ATM starts at the control state from the beginning. Nothing else is possible then."

$$\downarrow x_0 \,.\, [\boldsymbol{E}_1^*;\ (\mathsf{enterPIN} /\!\!/ \mathsf{check}') + \mathsf{cancel}](\langle \mathsf{ejectCard}\rangle x_0 \wedge [\mathsf{ejectCard}]x_0 \wedge [-\mathsf{ejectCard}]\mathsf{false})$$

(2.4) "Whenever an incorrect PIN has been entered three times in a row the ATM goes back to the initial control state." Hence the current card is not ejected; it is kept.

$$\downarrow x_0 \,.\, [\boldsymbol{E}_1^*;\ (\mathsf{enterPIN} /\!\!/ \neg\mathsf{check}')^3]x_0$$

It can easily be checked that all models of Sp_2 must satisfy the axioms (1.1–1.3) of Sp_1, i.e., $Sp_1 \rightsquigarrow Sp_2$ holds: Axiom (1.1) is obviously a consequence of the stronger axiom (2.2). Similarly, (1.2) is a consequence of the stronger axiom (2.3). Axiom (1.3) is a consequence of (2.1) which requires that the first event is always insertCard. Note that the Σ_1-edts M_1 shown in Fig. 1a is not a model of Sp_2 since it does not satisfy (2.4). A model M_2 of Sp_2 is shown in Fig. 3. Since $Sp_1 \rightsquigarrow Sp_2$, M_2 is also a model of Sp_1 and, since $Sp_0 \rightsquigarrow_{\kappa_{\sigma_0}} Sp_1$ holds, the σ_0-reduct of M_2 is a model of Sp_0. This reduct removes from M_2 all cancel transitions, the configuration $(Return, \neg\mathsf{check})$, and its outgoing ejectCard transition. □

During the development process it may turn out that the data state institution at hand is not expressive enough to model more concrete solutions. In such cases a change of institution is necessary. Then one can continue with constructors in the new institution as considered in Def. 14. If the constructor is a reduct constructor, the next theorem, generalising Thm. 3, is helpful to prove implementation correctness with institution change. It is a direct consequence of the satisfaction conditions of event/data institutions and institution comorphisms.

Theorem 4 *Let $\vec{\nu}^{\downarrow} : \mathcal{E}^{\downarrow}(\vec{\mathcal{D}}) \to \mathcal{E}^{\downarrow}(\vec{\mathcal{D}}')$ be a lifted comorphism between event/data institutions $\mathcal{E}^{\downarrow}(\vec{\mathcal{D}})$ and $\mathcal{E}^{\downarrow}(\vec{\mathcal{D}}')$. Let $Sp = (\Sigma, Ax)$ be an axiomatic specification over $\mathcal{E}^{\downarrow}(\vec{\mathcal{D}})$, Sp' a specification over $\mathcal{E}^{\downarrow}(\vec{\mathcal{D}}')$, and κ_σ from $\Sigma(Sp')$ to $(\vec{\nu}^{\downarrow})^{\mathbb{S}}(\Sigma)$ a reduct constructor via $\sigma : (\vec{\nu}^{\downarrow})^{\mathbb{S}}(\Sigma) \to \Sigma(Sp')$. Then, $Sp \rightsquigarrow^{\vec{\nu}^{\downarrow}}_{\kappa_\sigma} Sp'$ if, and only if, $M' \models^{\mathcal{E}^{\downarrow}(\vec{\mathcal{D}}')}_{\Sigma(Sp')} \mathrm{Sen}^{\mathcal{E}^{\downarrow}(\vec{\mathcal{D}}')}(\sigma)((\vec{\nu}^{\downarrow})^{\mathrm{Sen}}_{\Sigma}(\varrho))$ for all $M' \in \mathrm{Mod}^{\mathcal{E}^{\downarrow}(\vec{\mathcal{D}}')}(Sp')$ and all $\varrho \in Ax$.* □

Example 26 We continue the development of the ATM example. Axiom (2.4) of our last specification Sp_2 in Ex. 25 has required that whenever an incorrect PIN has been entered three times in a row, then the ATM moves back to its initial control state. In a solution-oriented specification it should therefore be possible to count the number of trials such that after a third trial with an incorrect PIN the control of the ATM is again at the beginning. Obviously, in the data institution $Prop_\emptyset$ we are not able to count. Therefore we switch to the data institution $Attr^{=}_{\Sigma_\mathrm{b},\mathfrak{A}_\mathrm{b}}$, defined in Ex. 5(c), where we have assumed that the underlying base signature Σ_b and its algebra interpretation support integers and thus counting. Hence we come up with the ed signature $\Sigma_A TM = (E_1, \iota : \Sigma_\mathrm{b} \hookrightarrow \Sigma_A)$ constructed in Ex. 22 and we consider the operational specification ATM in Fig. 2.

We want to show that $Sp_2 \rightsquigarrow^{\vec{\nu}^{\downarrow}_\mathrm{b}}_{\kappa_\sigma} ATM$ holds, i.e., ATM is a constructor implementation of Sp_2 via the institution comorphism $\vec{\nu}^{\downarrow}_\mathrm{b} : \mathcal{E}^{\downarrow}(Prop_\emptyset) \to \mathcal{E}^{\downarrow}(Attr^{=}_{\Sigma_\mathrm{b},\mathfrak{A}_\mathrm{b}})$ changing the data state institution from propositional logic to attribute-based data states (see Ex. 20) and using the reduct constructor κ_σ determined by the signature morphism $\sigma = (1_{E_1}, \vartheta) : (\vec{\nu}^{\downarrow}_\mathrm{b})^{\mathbb{S}}(\Sigma(Sp_2)) \to \Sigma_{ATM}$ with ϑ being the inclusion of attribute signatures. In detail, $\Sigma(Sp_2) = \Sigma_1 = (E_1, \delta_P : \emptyset \to \{\mathsf{check}\})$ in $Prop_\emptyset$ and therefore $(\vec{\nu}^{\downarrow}_\mathrm{b})^{\mathbb{S}}(\Sigma(Sp_2)) = (E_1, \iota_1 : \Sigma_\mathrm{b} \hookrightarrow \Sigma_{A_1})$ where Σ_{A_1} is the attribute signature induced by $A_1 = \{\mathsf{check} : \mathsf{Bool}\}$. The corresponding sentence translation maps any occurrence of the propositional atom check to the equation $\mathsf{check} = \mathsf{tt}$ and any occurrence of check' to $\mathsf{check}' = \mathsf{tt}$ and is extended to $\mathcal{E}^{\downarrow}(Prop_\emptyset)$-formulæ inductively; in particular, $\neg\mathsf{check}$ is translated to $\neg(\mathsf{check} = \mathsf{tt})$ and similarly for $\neg\mathsf{check}'$. From this it should be clear how the axioms of Sp_2 are translated to $Attr^{=}_{\Sigma_\mathrm{b},\mathfrak{A}_\mathrm{b}}$. For instance, axiom (2.4) is translated to

$$\downarrow x_0 \,.\, [\boldsymbol{E}_{\mathbf{1}}^{*};\ (\mathsf{enterPIN} \mathbin{/\!\!/} \neg(\mathsf{check}' = \mathsf{tt}))^3]x_0$$

By Thm. 4 it is sufficient to check that the translations of the axioms of Sp_2 are satisfied by all models of ATM. Concerning the first axiom (2.1) of Sp_2, we can associate the state variable x_0 with the control state $Card$ of ATM. Then only $\mathsf{insertCard} \mathbin{/\!\!/} \mathsf{check}' = \mathsf{ff} \wedge \mathsf{trials}' = 0$ is possible. Hence, $\mathsf{insertCard} \mathbin{/\!\!/} \mathsf{check}' = \mathsf{ff}$ and thus $\mathsf{insertCard} \mathbin{/\!\!/} \neg(\mathsf{check}' = \mathsf{tt})$ is possible, $\mathsf{insertCard} \mathbin{/\!\!/} \mathsf{check}' = \mathsf{tt}$ is not possible, and also no other event is possible. Thus the translated version of (2.1) holds in any model of ATM. For axiom (2.2) we observe that whenever a card has been inserted, control state PIN is reached in ATM. Then only transitions as required by (2.2) are possible in ATM. Concerning (2.3), after a correct PIN has been entered or if the transaction has been cancelled, control state $Return$ is reached. Then only ejectCard moving to the initial control state is possible. Finally, let us consider axiom (2.4) of Sp_2. Entering a PIN is only possible in control state PIN. If three times in a row an incorrect PIN has been entered, inevitably the initial control state $Card$ is reached again. Thus the translations of all axioms of Sp_2 are satisfied by ATM.□

A further refinement technique for reactive systems (see, e.g., [GR00]), is the implementation of simple events by complex events, like their sequential composition. To formalise this as a constructor we use *composite events* $\Theta(E)$ over a given set of events E, given by the grammar $\theta ::= e \mid \theta + \theta \mid \theta;\ \theta \mid \theta^*$ with $e \in E$. For any (E, δ)-edts M, additionally to the atomic events $e \in E$, the non-atomic events are *interpreted* by the relations $R(M)_{\theta_1+\theta_2} = R(M)_{\theta_1} \cup R(M)_{\theta_2}$, $R(M)_{\theta_1;\,\theta_2} = R(M)_{\theta_1};\ R(M)_{\theta_2}$, and $R(M)_{\theta^*} = (R(M)_\theta)^*$. Let $\Sigma = (E, \delta)$ be an ed signature, D a subset of $\Theta(E)$ including E, M a Σ-edts, and $\Sigma^D = (D, \delta)$. Then M induces a unique Σ^D-edts M^D whose relations for $\theta \in D$ are defined on top of M as indicated above.

Definition 16 Let Σ, Σ' be ed signatures, D' a subset of $\Theta(E(\Sigma'))$ including $E(\Sigma')$, $\Sigma'^{D'} = (D', \delta(\Sigma'))$, and $\alpha : \Sigma \to \Sigma'^{D'}$ an ed-signature morphism. The *event refinement* constructor κ_α from Σ' to Σ maps any $M' \in |\,Str^{\mathcal{E}^{\downarrow}(\vec{\mathcal{D}})}(\Sigma')|$ to the reduct of $M'^{D'}$ along α, that is to $(M'^{D'})|\alpha \in |\,Str^{\mathcal{E}^{\downarrow}(\vec{\mathcal{D}})}(\Sigma)|$. □

The event refinement constructor will be applied in Ex. 28.

6.3. Parallel composition constructor

Finally, we consider a semantic, synchronous parallel composition constructor that allows for decomposition of implementations into components which synchronise on shared events. The idea is to split the implementation of a specification into parts (the decomposition) and then to justify that the implementation is correct by showing that the parallel, synchronous product of the parts satisfies the requirements of the original specification.

Given two composable ed signatures Σ_1 and Σ_2 (see Sect. 5.2 for the composition of ed signatures), the *parallel composition* $\gamma_1 \otimes \gamma_2$ of two configurations $\gamma_1 = (c_1, d_1)$, $\gamma_2 = (c_2, d_2)$ is given by $((c_1, c_2), (d_1, d_2))$ of the pair of control states and the pair of data states, and lifted to two sets of configurations Γ_1 and Γ_2 by $\Gamma_1 \otimes \Gamma_2 = \{\gamma_1 \otimes \gamma_2 \mid \gamma_1 \in \Gamma_1,\ \gamma_2 \in \Gamma_2\}$.

Definition 17 Let Σ_1, Σ_2 be composable ed signatures. The *parallel composition constructor* $\kappa_\otimes$ from (Σ_1, Σ_2) to $\Sigma_1 \otimes \Sigma_2$ maps any $M_1 \in\mid Str^{\mathcal{E}^\downarrow(\vec{\mathcal{D}})}(\Sigma_1)\mid$, $M_2 \in\mid Str^{\mathcal{E}^\downarrow(\vec{\mathcal{D}})}(\Sigma_2)\mid$ to $M_1 \otimes M_2 = (\Gamma, R, \Gamma_0, \omega) \in\mid Str^{\mathcal{E}^\downarrow(\vec{\mathcal{D}})}(\Sigma_1 \otimes \Sigma_2)\mid$, where $\Gamma_0 = \Gamma_0(M_1) \otimes \Gamma_0(M_2)$, $\omega(\gamma_1 \otimes \gamma_2) = \omega(M_1)(\gamma_1) \times_{\delta(\Sigma_1),\delta(\Sigma_2)} \omega(M_2)(\gamma_2)$, and Γ and $R = (R_e)_{e \in E(\Sigma_1) \cup E(\Sigma_2)}$ are inductively defined by $\Gamma_0 \subseteq \Gamma$ and

- for all $e_1 \in E(\Sigma_1) \setminus E(\Sigma_2)$, $\gamma_1, \gamma_1' \in \Gamma(M_1)$, and $\gamma_2 \in \Gamma(M_2)$, if $\gamma_1 \otimes \gamma_2 \in \Gamma$ and $(\gamma_1, \gamma_1') \in R(M_1)_{e_1}$, then $\gamma_1' \otimes \gamma_2 \in \Gamma$ and $(\gamma_1 \otimes \gamma_2, \gamma_1' \otimes \gamma_2) \in R_{e_1}$;
- for all $e_2 \in E(\Sigma_2) \setminus E(\Sigma_1)$, $\gamma_2, \gamma_2' \in \Gamma(M_2)$, and $\gamma_1 \in \Gamma(M_1)$, if $\gamma_1 \otimes \gamma_2 \in \Gamma$ and $(\gamma_2, \gamma_2') \in R(M_2)_{e_2}$, then $\gamma_1 \otimes \gamma_2' \in \Gamma$ and $(\gamma_1 \otimes \gamma_2, \gamma_1 \otimes \gamma_2') \in R_{e_2}$;
- for all $e \in E(\Sigma_1) \cap E(\Sigma_2)$, $\gamma_1, \gamma_1' \in \Gamma(M_1)$, and $\gamma_2, \gamma_2' \in \Gamma(M_2)$, if $\gamma_1 \otimes \gamma_2 \in \Gamma$, $(\gamma_1, \gamma_1') \in R(M_1)_e$, and $(\gamma_2, \gamma_2') \in R(M_2)_e$, then $\gamma_1' \otimes \gamma_2' \in \Gamma$ and $(\gamma_1 \otimes \gamma_2, \gamma_1' \otimes \gamma_2') \in R_e$. □

An obvious question is how the semantic parallel composition constructor is related to the syntactic parallel composition of operational specifications. The following proposition shows that semantic parallel composition is included in the semantics of syntactic parallel composition.

Proposition 2 *Let O_1, O_2 be operational specifications with composable signatures. Then*

$$\mathrm{Mod}^{\mathcal{E}^\downarrow(\vec{\mathcal{D}})}(O_1) \otimes \mathrm{Mod}^{\mathcal{E}^\downarrow(\vec{\mathcal{D}})}(O_2) \subseteq \mathrm{Mod}^{\mathcal{E}^\downarrow(\vec{\mathcal{D}})}(O_1 \parallel O_2)\,,$$

where $\mathrm{Mod}^{\mathcal{E}^\downarrow(\vec{\mathcal{D}})}(O_1) \otimes \mathrm{Mod}^{\mathcal{E}^\downarrow(\vec{\mathcal{D}})}(O_2) = \{M_1 \otimes M_2 \mid M_1 \in \mathrm{Mod}^{\mathcal{E}^\downarrow(\vec{\mathcal{D}})}(O_1),\ M_2 \in \mathrm{Mod}^{\mathcal{E}^\downarrow(\vec{\mathcal{D}})}(O_2)\}$.

Proof. Let $O_i = (\Sigma_i, C_i, T_i, (c_{i,0}, \varphi_{i,0}))$ for $i \in \{1, 2\}$, $\Sigma = \Sigma_1 \otimes \Sigma_2$, and $O = O_1 \parallel O_2 = (\Sigma, C, T, (c_0, \varphi_0))$. For an $\omega \in |Str^{\vec{\mathcal{D}}}(\delta(\Sigma))|$ abbreviate $Str^{\vec{\mathcal{D}}}(\hat{\delta}(\Sigma_i))(\omega)$ by $\omega|\Sigma_i$ for $i \in \{1, 2\}$; for $\omega_1 \in |Str^{\vec{\mathcal{D}}}(\delta(\Sigma_1))|$ and $\omega_2 \in |Str^{\vec{\mathcal{D}}}(\delta(\Sigma_2))|$ abbreviate $\omega_1 \times_{\delta(\Sigma_1),\delta(\Sigma_2)} \omega_2$ by $\omega_1 \times \omega_2$. Let $M_i \in \mathrm{Mod}^{\mathcal{E}^\downarrow(\vec{\mathcal{D}})}(O_i)$ for $i \in \{1, 2\}$ and $M = M_1 \otimes M_2$; we prove that $M \in \mathrm{Mod}^{\mathcal{E}^\downarrow(\vec{\mathcal{D}})}(O)$.

For the initialisation condition, we have $c_0(M) = (c_{1,0}, c_{2,0}) = c_0(O_1 \parallel O_2)$ and $\omega_1 \times \omega_2 \models^{\vec{\mathcal{D}}}_{\delta(\Sigma)} \hat{\varphi}_{0,1} \wedge \hat{\varphi}_{0,2} = \varphi_0(O_1 \parallel O_2)$ for all $\omega_1 \times \omega_2 \in \Omega_0(M)$ as required.

For condition (1) of Def. 11, let $\gamma = (c_1, d_1) \otimes (c_2, d_2) \in \Gamma(M)$ and $((c_1, c_2), \varphi, e, \psi, (c_1', c_2')) \in T(O)$ with $\omega(M)(\gamma) \models^{\vec{\mathcal{D}}}_{\delta(\Sigma)} \varphi$ be given.

Case $e \in E(\Sigma_1) \setminus E(\Sigma_2)$: Then $(c_1, \varphi_{1,e}, e, \psi_{1,e}, c_1') \in T(O_1)$ with $\varphi = \hat{\varphi}_{1,e}$, $\psi = \hat{\psi}_{1,e} \wedge \mathrm{id}_{\Delta(\Sigma_2)}$, and it holds that $(\omega(M)(\gamma))|\Sigma_1 \models^{\vec{\mathcal{D}}}_{\Delta(\Sigma_1)} \varphi_{1,e}$. As $M_1 \in \mathrm{Mod}^{\mathcal{E}^\downarrow(\vec{\mathcal{D}})}(O_1)$ and $(c_1, d_1) \in \Gamma(M_1)$, there is a $((c_1, d_1), (c_1', d_1')) \in R(M_1)_e$ such that $(\omega(M_1)(d_1), \omega(M_1)(d_1')) \models^{2\vec{\mathcal{D}}}_{\delta(\Sigma_1)} \psi_{1,e}$. Let $\gamma' = (c_1', d_1') \otimes (c_2, d_2)$; then it holds that $\omega(M)(\gamma') = \omega(M_1)(d_1') \times \omega(M_2)(d_2)$ and $(\omega(M)(\gamma), \omega(M)(\gamma')) \models^{2\vec{\mathcal{D}}}_{\delta(\Sigma)} \psi$. Then $(\gamma, \gamma') \in R(M)_e$, since $\gamma \in \Gamma(M)$.

Case $e \in E(\Sigma_2) \setminus E(\Sigma_1)$: Symmetric to $e \in E(\Sigma_1) \setminus E(\Sigma_2)$.

Case $e \in E(\Sigma_1) \cap E(\Sigma_2)$: Then $(c_1, \varphi_{1,e}, e, \psi_{1,e}, c_1') \in T(O_1)$, $(c_2, \varphi_{2,e}, e, \psi_{2,e}, c_2') \in T(O_2)$ with $\varphi = \hat{\varphi}_{1,e} \wedge \hat{\varphi}_{2,e}$, $\psi = \hat{\psi}_{1,e} \wedge \hat{\psi}_{2,e}$, and $\omega(M)(\gamma)|\Sigma_1 \models^{\vec{\mathcal{D}}}_{\delta(\Sigma_1)} \varphi_{1,e}$, $\omega(M)(\gamma)|\Sigma_2 \models^{\vec{\mathcal{D}}}_{\delta(\Sigma_2)} \varphi_{2,e}$. Since $M_i \in \mathrm{Mod}^{\mathcal{E}^\downarrow(\vec{\mathcal{D}})}(O_i)$ and $(c_i, d_i) \in \Gamma(M_i)$, there are $((c_i, d_i), (c_i', d_i')) \in R(M_i)_e$ such that $(\omega(M_i)(d_i), \omega(M_i)(d_i')) \models^{2\vec{\mathcal{D}}}_{\delta(\Sigma_i)} \psi_{i,e}$ for $i \in \{1, 2\}$. Let $\gamma' = (c_1', d_1') \otimes (c_2', d_2')$; then it holds that $\omega(M)(\gamma') = \omega(M_1)(d_1') \times \omega(M_2)(d_2')$ and $(\omega(M)(\gamma), \omega(M)(\gamma')) \models^{2\vec{\mathcal{D}}}_{\delta(\Sigma)} \psi$. Then $(\gamma, \gamma') \in R(M)_e$ since $\gamma \in \Gamma(M)$.

For condition (2) of Def. 11, we now show that for all $e \in E(\Sigma)$ and $((c_1, d_1) \otimes (c_2, d_2), (c_1', d_1') \otimes (c_2', d_2')) \in R(M)_e$ there is some $((c_1, c_2), \varphi, e, \psi, (c_1', c_2')) \in T(O)$ such that $\omega(M)(d_1, d_2) \models^{\vec{\mathcal{D}}}_{\delta(\Sigma)} \varphi$ and $(\omega(M)(d_1, d_2),$

$\omega(M)(d_1', d_2')) \models^{2\vec{\mathcal{D}}}_{\delta(\Sigma)} \psi$ by induction over the reachability of $\Gamma(M)$: Let $(c_1, d_1) \otimes (c_2, d_2) \in \Gamma(M)$ with $((c_{1,i}, d_{1,i}) \otimes (c_{2,i}, d_{2,i}), (c_{1,i+1}, d_{1,i+1}) \otimes (c_{2,i+1}, d_{2,i+1})) \in R(M)_{e_{i+1}}$ such that there are $((c_{1,i}, c_{2,i}), \varphi_{i+1}, e_{i+1}, \psi_{i+1}, (c_{1,i+1}, c_{2,i+1})) \in T(O)$ satisfying $\omega(M)(d_{1,i}, d_{2,i}) \models^{\vec{\mathcal{D}}}_{\delta(\Sigma)} \varphi_i$ and $(\omega(M)(d_{1,i}, d_{2,i}), \omega(M)(d_{1,i+1}, d_{2,i+1})) \models^{2\vec{\mathcal{D}}}_{\delta(\Sigma)} \psi_{i+1}$ for $0 \leq i < n$, $(c_{1,0}, d_{1,0}) \otimes (c_{2,0}, d_{2,0}) \in \Gamma_0(M)$, and $(c_{1,n}, d_{1,n}) \otimes (c_{2,n}, d_{2,n}) = (c_1, d_1) \otimes (c_2, d_2)$. Let $((c_1, d_1) \otimes (c_2, d_2), (c_1', d_1') \otimes (c_2', d_2')) \in R(M)_e$.

Case $e \in E(\Sigma_1) \setminus E(\Sigma_2)$: Then $((c_1, d_1), (c_1', d_1')) \in R(M_1)_e$ and $c_2' = c_2$, $d_2' = d_2$. Since $M_1 \in \mathrm{Mod}^{\mathcal{E}^{\downarrow}(\vec{\mathcal{D}})}(O_1)$, there is some $(c_1, \varphi_1, e, \psi_1, c_1') \in T(O_1)$ such that $(\omega(M)(d_1, d_2))|\Sigma_1 \models^{\vec{\mathcal{D}}}_{\delta(\Sigma_1)} \varphi_1$ and also $((\omega(M)(d_1, d_2))|\Sigma_1, (\omega(M)(d_1', d_2'))|\Sigma_1) \models^{2\vec{\mathcal{D}}}_{\delta(\Sigma_1)} \psi_1$. By induction hypothesis, $(c_1, c_2) \in C(O)$ and hence $((c_1, c_2), \hat{\varphi}_1, e, \hat{\psi}_1 \wedge \mathrm{id}_{\Delta(\Sigma_2)}, (c_1', c_2)) \in T(O)$, where $\omega(M)(d_1, d_2) \models^{\vec{\mathcal{D}}}_{\delta(\Sigma)} \hat{\varphi}_1$ and $(\omega(M)(d_1, d_2), \omega(M)(d_1', d_2') \models^{2\vec{\mathcal{D}}}_{\delta(\Sigma)} \hat{\psi}_1 \wedge \mathrm{id}_{\Delta(\Sigma_2)}$.

Case $e \in E(\Sigma_2) \setminus E(\Sigma_1)$: Symmetric to $e \in E(\Sigma_1) \setminus E(\Sigma_2)$.

Case $e \in E(\Sigma_1) \cap E(\Sigma_2)$: Then $((c_1, d_1), (c_1', d_1')) \in R(M_1)_e$ and $((c_2, d_2), (c_2', d_2')) \in R(M_2)_e$. Since $M_i \in \mathrm{Mod}^{\mathcal{E}^{\downarrow}(\vec{\mathcal{D}})}(O_i)$, there are some $(c_i, \varphi_i, e, \psi_i, c_i') \in T(O_i)$ such that $(\omega(M)(d_1, d_2))|\Sigma_i \models^{\vec{\mathcal{D}}}_{\delta(\Sigma_i)} \varphi_i$ holds and also $((\omega(M)(d_1, d_2))|\Sigma_i, \omega(M)(d_1', d_2'))|\Sigma_i) \models^{2\vec{\mathcal{D}}}_{\delta(\Sigma_i)} \psi_i$ for $i \in \{1, 2\}$. By induction hypothesis, $(c_1, c_2) \in C(O)$ and hence $((c_1, c_2), \hat{\varphi}_1 \wedge \hat{\varphi}_2, e, \hat{\psi}_1 \wedge \hat{\psi}_2, (c_1', c_2')) \in T(O)$, where $\omega(M)(d_1, d_2) \models^{\vec{\mathcal{D}}}_{\delta(\Sigma)} \hat{\varphi}_1 \wedge \hat{\varphi}_2$ and $(\omega(M)(d_1, d_2), \omega(M)(d_1', d_2')) \models^{2\vec{\mathcal{D}}}_{\delta(\Sigma)} \hat{\psi}_1 \wedge \hat{\psi}_2$. □

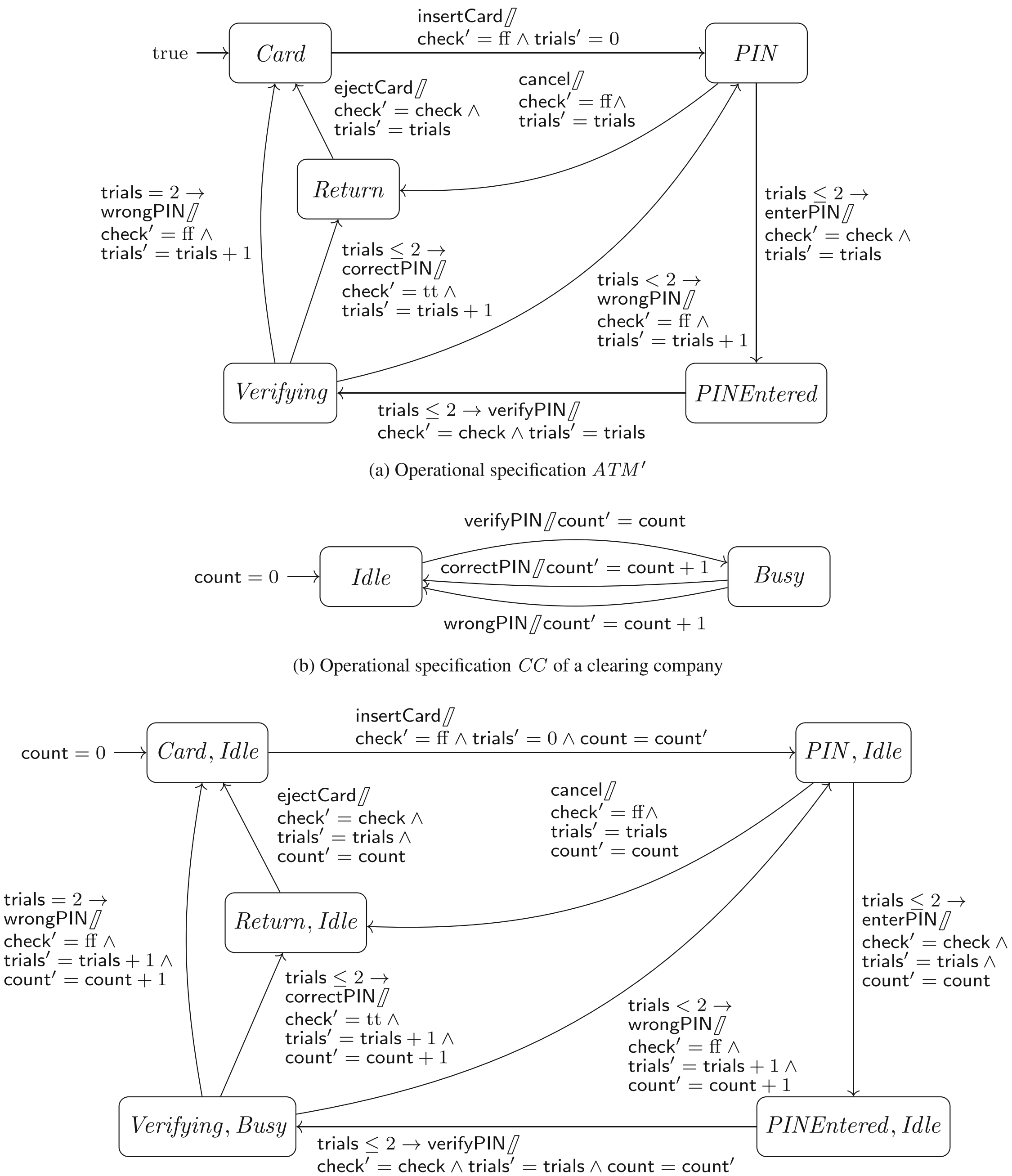
Example 27 The converse of Prop. 2, i.e., $\mathrm{Mod}^{\mathcal{E}^{\downarrow}(\vec{\mathcal{D}})}(O_1 \parallel O_2) \subseteq \mathrm{Mod}^{\mathcal{E}^{\downarrow}(\vec{\mathcal{D}})}(O_1) \otimes \mathrm{Mod}^{\mathcal{E}^{\downarrow}(\vec{\mathcal{D}})}(O_2)$, does, in general, not hold: Consider the ed signature $\Sigma = (E, \iota)$ in $\mathcal{E}^{\downarrow}(Attr^{=}_{\Sigma_b, \mathfrak{A}_b})$ with $E = \{e\}$, and ι representing the empty set of attributes. Consider also the operational specifications $O_i = (\Sigma, C_i, T_i, (c_{i,0}, \varphi_{i,0}))$ for $i \in \{1, 2\}$ with $C_1 = \{c_{1,0}\}$, $T_1 = \{(c_{1,0}, \mathrm{true}, e, \mathrm{false}, c_{1,0})\}$, $\varphi_{1,0} = \mathrm{true}$; and $C_2 = \{c_{2,0}\}$, $T_2 = \emptyset$, $\varphi_{2,0} = \mathrm{true}$. Obviously, $\mathrm{Mod}^{\mathcal{E}^{\downarrow}(\vec{\mathcal{D}})}(O_1) = \emptyset$ since there is a transition in T_1 with precondition true and transition predicate false which is not satisfiable. Therefore, $\mathrm{Mod}^{\mathcal{E}^{\downarrow}(\vec{\mathcal{D}})}(O_1) \otimes \mathrm{Mod}^{\mathcal{E}^{\downarrow}(\vec{\mathcal{D}})}(O_2) = \emptyset$. On the other hand, $\mathrm{Mod}^{\mathcal{E}^{\downarrow}(\vec{\mathcal{D}})}(O_2)$ has a single model showing just the initial configuration w.r.t. O_2 but no transition. By definition of the syntactic parallel composition of operational specifications, $O_1 \parallel O_2$ has no transition since $T_2 = \emptyset$ and the only event is the shared event e. Therefore, $\mathrm{Mod}^{\mathcal{E}^{\downarrow}(\vec{\mathcal{D}})}(O_1 \parallel O_2) = \{M\}$ with M showing just the initial configuration w.r.t. $O_1 \parallel O_2$. □

The next theorem shows the usefulness of the syntactic parallel composition operator for proving implementation correctness when a (semantic) parallel composition constructor is involved. The theorem is a direct consequence of Prop. 2 and Def. 13.

Theorem 5 *Let Sp be an (axiomatic or operational) specification, O_1, O_2 operational specifications with composable signatures, and κ an implementation constructor from $\Sigma(O_1) \otimes \Sigma(O_2)$ to $\Sigma(Sp)$: If $Sp \rightsquigarrow_{\kappa} O_1 \parallel O_2$, then $Sp \rightsquigarrow_{\kappa_{\otimes};\kappa} \langle O_1, O_2 \rangle$.*

Proof. Let $Sp \rightsquigarrow_{\kappa} O_1 \parallel O_2$ hold, i.e., $\kappa(M) \in \mathrm{Mod}^{\mathcal{E}^{\downarrow}(\vec{\mathcal{D}})}(Sp)$ for all $M \in \mathrm{Mod}^{\mathcal{E}^{\downarrow}(\vec{\mathcal{D}})}(O_1 \parallel O_2)$. Let $M_1 \in \mathrm{Mod}^{\mathcal{E}^{\downarrow}(\vec{\mathcal{D}})}(O_1)$ and $M_2 \in \mathrm{Mod}^{\mathcal{E}^{\downarrow}(\vec{\mathcal{D}})}(O_2)$. Then $\kappa_{\otimes}(M_1, M_2) = M_1 \otimes M_2 \in \mathrm{Mod}^{\mathcal{E}^{\downarrow}(\vec{\mathcal{D}})}(O_1 \parallel O_2)$ by Prop. 2, and therefore $\kappa(\kappa_{\otimes}(M_1, M_2)) \in \mathrm{Mod}^{\mathcal{E}^{\downarrow}(\vec{\mathcal{D}})}(Sp)$. Thus $Sp \rightsquigarrow_{\kappa_{\otimes};\kappa} \langle O_1, O_2 \rangle$. □

Example 28 We finish the refinement chain for the ATM specifications by applying a decomposition into two parallel components. The operational specification ATM of Fig. 2 (and Ex. 26) describes the interface behaviour of an ATM interacting with a user. For a concrete realisation, however, an ATM will also interact internally with other components, like, e.g., a clearing company which supports the ATM for verifying PINs. Our last refinement step hence realises the specification ATM by two parallel components, represented by the operational specification ATM' in Fig. 4a and the operational specification CC of a clearing company in Fig. 4b. Models of both specifications communicate (via shared events) when a model of ATM' sends a verification request verifyPIN to a model of CC. The clearing company model can answer with correctPIN or wrongPIN and then the ATM'-model continues following its specification. For the implementation construction we use the parallel composition constructor



(a) Operational specification ATM'

(b) Operational specification CC of a clearing company

(c) Syntactic parallel composition $ATM' \parallel CC$

Fig. 4. Operational specifications ATM', CC and their parallel composition

$\kappa_{\otimes}$ from $(\Sigma(ATM'), \Sigma(CC))$ to $\Sigma(ATM') \otimes \Sigma(CC)$

which synchronises the models of ATM' and CC on shared events. The signature of CC consists of the events shown on the transitions in Fig. 4b. Moreover, there is one integer-valued attribute count counting the number of verification tasks performed. The signature of ATM' extends $\Sigma(ATM)$ by the events verifyPIN, correctPIN and wrongPIN. The ed signature $\Sigma(ATM') \otimes \Sigma(CC)$ is therefore $(\hat{E}, \hat{\iota} : \Sigma_{\mathrm{b}} \hookrightarrow \Sigma_{\hat{A}})$ with

$$\hat{E} = \{\mathsf{insertCard}, \mathsf{enterPIN}, \mathsf{ejectCard}, \mathsf{cancel}, \mathsf{verifyPIN}, \mathsf{correctPIN}, \mathsf{wrongPIN}\}$$

and $\Sigma_{\hat{A}}$ is the attribute signature induced by the set of attributes $\hat{A} = \{\mathsf{check} : \mathsf{Bool}, \mathsf{trials} : \mathsf{Int}, \mathsf{count} : \mathsf{Int}\}$.

So far, we can apply $\kappa_\otimes$ to any models $M_1 \in \mathrm{Mod}^{\mathcal{E}^\downarrow(Attr^{=}_{\Sigma_b,\mathfrak{A}_b})}(ATM')$ and $M_2 \in \mathrm{Mod}^{\mathcal{E}^\downarrow(Attr^{=}_{\Sigma_b,\mathfrak{A}_b})}(CC)$ obtaining the $(\hat{E}, \hat{\imath})$-edts $M_1 \otimes M_2$. Our final goal is, however, to get an implementation of the specification ATM. For this purpose we compose $\kappa_\otimes$ with an event refinement constructor (cf. Def. 16)

$$\kappa_\alpha \quad \text{from} \quad \Sigma(ATM') \otimes \Sigma(CC) \quad \text{to} \quad \Sigma(ATM)\,.$$

We consider $\alpha : \Sigma(ATM) \to (\Sigma(ATM') \otimes \Sigma(CC))^D, \theta = \mathsf{enterPIN};\ \mathsf{verifyPIN};\ (\mathsf{correctPIN} + \mathsf{wrongPIN})$ and $D = \hat{E} \cup \{\theta\}$. The event part of α maps $\mathsf{enterPIN}$ to θ; for the other events α is the identity and for the attributes the inclusion. Finally, we have to prove the refinement relation

$$ATM \rightsquigarrow_{\kappa_\otimes;\ \kappa_\alpha} \langle ATM', CC\rangle\,.$$

For doing this, we rely on the syntactic parallel composition $ATM' \parallel CC$ shown in Fig. 4c, and on Thm. 5. It is easy to see that $ATM \rightsquigarrow_{\kappa_\alpha} ATM' \parallel CC$. In fact, all transitions for event $\mathsf{enterPIN}$ in Fig. 2 are split into several transitions in Fig. 4c according to the event refinement defined by α. For instance, the loop transition from PIN to PIN with precondition $\mathsf{trials} < 2$ in Fig. 2 is split into the cycle from $(PIN, Idle)$ via $(PINEntered, Idle)$ and $(Verifying, Busy)$ back to $(PIN, Idle)$ in Fig. 4c. Thus, we have $ATM \rightsquigarrow_{\kappa_\alpha} ATM' \parallel CC$ and can apply Thm. 5 such that we get $ATM \rightsquigarrow_{\kappa_\otimes;\ \kappa_\alpha} \langle ATM', CC\rangle$.

Let us note again that we have implemented a small, prototypical tool that allows to check $\mathcal{E}^\downarrow(\vec{\mathcal{D}})$-formulæ on finite-state edts and the refinement of finite-state operational specifications using constructor implementations such that this running example can be reproduced. □

We conclude this section by showing that the condition in Theorem 5 relying on Prop. 2 is even necessary under some additional assumptions. First, for the converse of Prop. 2 to hold we have to consider well-formed operational specifications O_1 and O_2 only: An operational specification O with signature Σ is *well-formed*, if for all its transitions $(c, \varphi, e, \psi, c')$ and all $\omega \in |Str^{\vec{\mathcal{D}}}(\delta(\Sigma))|$ the following holds: If $\omega \models^{\vec{\mathcal{D}}}_{\delta(\Sigma)} \varphi$, then there exists an $\omega' \in |Str^{\vec{\mathcal{D}}}(\delta(\Sigma))|$ such that $(\omega, \omega') \models^{2\vec{\mathcal{D}}}_{\delta(\Sigma)} \psi$. Note that the counterexample Ex. 27 to the converse inclusion of Prop. 2 indeed involves an operational specification, namely O_1, that is not well-formed. Second, we restrict the model class of $O_1 \parallel O_2$ to extensional models since only an extensional model of the parallel composition of two (well-formed) operational specifications can be split into models of the single operational specifications. Finally, an isomorphism, i.e., a structure morphism which is bijective on the configurations, has to be allowed for:

Proposition 3 Let O_1 and O_2 be well-formed operational specifications with composable signatures. Let $M \in \mathrm{Mod}^{\mathcal{E}^\downarrow(\vec{\mathcal{D}})}(O_1 \parallel O_2)$ be extensional. Then there is an $M_1 \otimes M_2 \in \mathrm{Mod}^{\mathcal{E}^\downarrow(\vec{\mathcal{D}})}(O_1) \otimes \mathrm{Mod}^{\mathcal{E}^\downarrow(\vec{\mathcal{D}})}(O_2)$ such that M and $M_1 \otimes M_2$ are isomorphic.

Proof. Let $O_i = (\Sigma_i, C_i, T_i, (c_{i,0}, \varphi_{i,0}))$ for $i \in \{1, 2\}$, $\Sigma = \Sigma_1 \otimes \Sigma_2$, and $O = O_1 \parallel O_2 = (\Sigma, C, T, (c_0, \varphi))$; for $\omega_1 \in |Str^{\vec{\mathcal{D}}}(\delta(\Sigma_1))|$ and $\omega_2 \in |Str^{\vec{\mathcal{D}}}(\delta(\Sigma_2))|$ abbreviate $\omega_1 \times_{\delta(\Sigma_1),\delta(\Sigma_2)} \omega_2$ by $\omega_1 \times \omega_2$.

Let $M \in \mathrm{Mod}^{\mathcal{E}^\downarrow(\vec{\mathcal{D}})}(O)$ be extensional, such that data states and data labels are identified. For a $\gamma = ((c_1, c_2), \omega_1 \times \omega_2) \in \Gamma(M)$ define $\pi_i(\gamma) = (c_i, \omega_i)$ for $i \in \{1, 2\}$. Define the Σ_i-edts $M_i = (\Gamma_i, R_i, \Gamma_{i,0})$ with $\Gamma_i = \{\pi_i(\gamma) \mid \gamma \in \Gamma(M)\}$, $R_{i,e_i} = \{(\pi_i(\gamma), \pi_i(\gamma')) \mid (\gamma, \gamma') \in R(M)_{e_i}\}$ for $e_i \in E(\Sigma_i)$, and $\Gamma_{i,0} = \{\pi_i(\gamma_0) \mid \gamma_0 \in \Gamma_0(M)\}$. Then $M_1 \otimes M_2 \cong M$: It is $\Gamma_0(M_1 \otimes M_2) \cong \Gamma_0(M)$ by definition; for each $e \in E(\Sigma)$ the inclusion $R(M_1 \otimes M_2)_e \subseteq R(M)_e$ up to isomorphism follows by induction on the syntactic reachability of $O_1 \parallel O_2$ and the reverse inclusion $R(M)_e \subseteq R(M_1 \otimes M_2)_e$ up to isomorphism by induction on the reachability of M and using the extensionality of M. However, $M_i \notin \mathrm{Mod}^{\mathcal{E}^\downarrow(\vec{\mathcal{D}})}(O_i)$ for $i \in \{1, 2\}$, in general, as there may be a configuration $((c_1, c_2), \omega_1 \times \omega_2) \in \Gamma(M)$ with a transition $(c_1, \varphi_1, e, \psi_1, c_1') \in T(O_1)$ for a shared $e \in E(\Sigma_1) \cap E(\Sigma_2)$ such that $\omega_1 \models^{\vec{\mathcal{D}}}_{\delta(\Sigma_1)} \varphi_1$ but no transition $(c_2, \varphi_2, e, \psi_2, c_2') \in T(O_2)$ with $\omega_2 \models^{\vec{\mathcal{D}}}_{\delta(\Sigma_2)} \varphi_2$, and, similarly, with 1 and 2 switched (by contrast, transitions of O_1 or O_2 for non-shared events always have a counterpart in M_1 or M_2).

For reconstructing these "lost" transitions define their set as

$$\begin{aligned} I_{i,e}((c_1, c_2), \omega_1 \times \omega_2) = \{(c_i, \varphi_i, e, \psi_i, c_i') \in T(O_i) \mid \omega_i \models^{\vec{\mathcal{D}}}_{\delta(\Sigma_i)} \varphi_i \wedge{} \\ \forall (c_{3-i}, \varphi_{3-i}, e, \psi_{3-i}, c_{3-i}') \in T(O_{3-i})\,.\,\omega_{3-i} \not\models^{\vec{\mathcal{D}}}_{\delta(\Sigma_{3-i})} \varphi_{3-i}\} \end{aligned}$$

for all $((c_1, c_2), \omega_1 \times \omega_2) \in \Gamma(M)$, $e \in E(\Sigma_1) \cap E(\Sigma_2)$, and $i \in \{1, 2\}$. Also define semantic counterparts by

$$K^{(1)}_{i,e} = \{((c_i, \omega_i), (c'_i, \omega'_i)) \mid \exists\, \gamma \in \Gamma(M), (c_i, \varphi_i, e_i, \psi_i, c'_i) \in I_{i,e}(\gamma) \,.\, \pi_i(\gamma) = (c_i, \omega_i) \wedge (\omega_i, \omega'_i) \models^{2\vec{\mathcal{D}}}_{\delta(\Sigma_i)} \psi_i\}$$

for all $e \in E(\Sigma_1) \cap E(\Sigma_2)$ and $i \in \{1, 2\}$; these semantic counterparts are bound to be non-empty due to the well-formedness of O_i. Define the Σ_i-edts $M^{(1)}_i = (\Gamma^{(1)}_i, R^{(1)}_i, \Gamma^{(1)}_{i,0})$ with $R^{(1)}_{i,e} = R_{i,e}$ for $e \in E(\Sigma) \setminus (E(\Sigma_1) \cap E(\Sigma_2))$ and $R^{(1)}_{i,e} = R_{i,e} \cup K^{(1)}_{i,e}$ for $e \in E(\Sigma_1) \cap E(\Sigma_2)$, and $\Gamma^{(1)}_i = \Gamma_i \cup \{\gamma'_i \mid (\gamma_i, \gamma'_i) \in \bigcup_{e_i \in E(\Sigma_i)} R^{(1)}_{e_i}\}$ and $\Gamma^{(1)}_{i,0} = \Gamma_{i,0}$. Again it holds that $M^{(1)}_1 \otimes M^{(1)}_2 \cong M$ as the added semantic relations do not contribute to the parallel composition. However, still $M^{(1)}_i \notin \mathrm{Mod}^{\mathcal{E}^{\downarrow}(\vec{\mathcal{D}})}(O_i)$ for $i \in \{1, 2\}$, in general, as there may be a configuration $(c_i, \omega_i) \in \Gamma(M^{(1)}_i) \setminus \Gamma(M_i)$ with a transition $(c_i, \varphi_i, e_i, \psi_i, c'_i) \in T(O_i)$ for some $e_i \in E(\Sigma_i)$ such that $\omega_i \models^{\vec{\mathcal{D}}}_{\delta(\Sigma_i)} \varphi_i$ but no $((c_i, \omega_i), (c'_i, \omega'_i)) \in R(M^{(1)}_i)_{e_i}$; again by the well-formedness of O_i such a successor (c'_i, ω'_i) must exist.

For adding these missing relations let $M^{(0)}_i = M_i$ for $i \in \{1, 2\}$ and define recursively

$$\begin{aligned}
K^{(n+1)}_{i,e_i} &= \{((c_i, \omega_i), (c'_i, \omega'_i)) \mid (c_i, \omega_i) \in \Gamma(M^{(n)}) \setminus \Gamma(M^{(n-1)}) \wedge \\
&\qquad\qquad \exists (c_i, \varphi_i, e_i, \psi_i, c'_i) \in T(O_i) \,.\, \omega_i \models^{\vec{\mathcal{D}}}_{\delta(\Sigma_i)} \varphi_i \wedge (\omega_i, \omega'_i) \models^{2\vec{\mathcal{D}}}_{\delta(\Sigma_i)} \psi_i\} \\
R^{(n+1)}_{i,e_i} &= R^{(n)}_{i,e_i} \cup K^{(n+1)}_{i,e_i} \\
M^{(n+1)}_i &= (\Gamma(M^{(n)}_i) \cup \{\gamma'_i \mid (\gamma_i, \gamma'_i) \in \textstyle\bigcup_{e_i \in E(\Sigma_i)} R^{(n+1)}_{i,e_i}\}, R^{(n+1)}_i, \Gamma_{i,0})
\end{aligned}$$

for all $e_i \in E(\Sigma_i)$, $i \in \{1, 2\}$, and $n \geq 1$. As the added semantic relations again do not contribute to the parallel composition it holds that $M^{(n)}_1 \otimes M^{(n)}_2 \cong M$ for all $n \geq 0$. Finally, for $M^{\infty}_i = (\bigcup_{0 \leq n} \Gamma(M^{(n)}_i), \bigcup_{0 \leq n} R(M^{(n)}_i), \Gamma_{i,0})$ for $i \in \{1, 2\}$ it holds that both $M^{\infty}_1 \otimes M^{\infty}_2 \cong M$ and $M_i \in \mathrm{Mod}^{\mathcal{E}^{\downarrow}(\vec{\mathcal{D}})}(O_i)$ for $i \in \{1, 2\}$. □

7. Conclusions

We have presented a novel logic, called $\mathcal{E}^{\downarrow}(\vec{\mathcal{D}})$-logic, for the rigorous formal development of event-based systems incorporating changing data states. This logic is formalised as a generic institution which can be instantiated by underlying data state institutions $\vec{\mathcal{D}}$. Due to its genericity, $\mathcal{E}^{\downarrow}(\vec{\mathcal{D}})$ is applicable for different data models and supports migration from one data state institution to another one during system development. The parametric approach of $\mathcal{E}^{\downarrow}(\vec{\mathcal{D}})$ follows the lines of previous approaches for tailoring logics to specific formalisms describing properties of dynamic systems [FG92, DS07, MMDB11]; see Sect. 1 for the relationship to $\mathcal{E}^{\downarrow}(\vec{\mathcal{D}})$. Our approach supports the full development process for event/data-based systems ranging from abstract requirements specifications, expressible by the dynamic logic features, to constructive specifications of implementations, expressible by the hybrid part of the logic.

The temporal logic of actions (TLA [Lam03]) supports also stepwise refinement where state transition predicates are considered as actions. In contrast to TLA we model also the events which cause data state transitions. For writing concrete specifications we have proposed an operational specification format capturing (at least parts of) similar formalisms, like Event-B [Abr13], symbolic transition systems [PR06], and UML protocol state machines [OMG17]. A significant difference to Event-B machines is that we distinguish between control and data states, the former being encoded as data in Event-B. An institution-based semantics of Event-B has been proposed in [FMP17] which coincides with our semantics of operational specifications for the special case of deterministic data state transition predicates. Similarly, our semantics of operational specifications coincides with the unfolding of symbolic transition systems in [PR06] if we instantiate our generic data domain with algebraic specifications of data types (and consider again only deterministic transition predicates). The syntax of UML protocol state machines is about the same as the one of operational event/data specifications. As a consequence, all of the aforementioned concrete specification formalisms (and several others) would be appropriate candidates for integration into a development process based on $\mathcal{E}^{\downarrow}(\vec{\mathcal{D}})$-logic.

There remain several interesting tasks for future research. First, our events do not support parameters which is obviously desired in an extension of current $\mathcal{E}^{\downarrow}(\vec{\mathcal{D}})$. Of course, the parameters must also be formalised on an institution-independent level. Secondly, we believe that the investigation of the 2-data state institution provides an interesting basis for a (generic) formalisation of pre/post-condition specification styles like OCL [OMG14]. It would also be interesting to work out a formal relationship between our 2-data state approach and the hybridisation process in [DM16] using rigidity constraints. Another issue concerns the separation of events into inputs and outputs as in I/O-automata [Lyn03]. Then also communication compatibility, see [dAH01] for interface automata without and [MCM09] with data, as well as [MCM09, BHW11] for interface theories with data, would become relevant when applying the parallel composition constructor. $\mathcal{E}^{\downarrow}(\vec{\mathcal{D}})$-logic is not equipped with a proof system for deriving consequences of specifications which is a working package for its own. This would also support the proof of refinement steps which is currently achieved by purely semantic reasoning. A proof system for $\mathcal{E}^{\downarrow}(\vec{\mathcal{D}})$-logic must cover dynamic and hybrid logic parts at the same time, like the proof system in [MBHM18], which, however, does not consider data states, and the recent calculus of [BP18], which extends differential dynamic logic but does not handle events and reactions to events. Both proof systems could be appropriate candidates for incorporating the features of $\mathcal{E}^{\downarrow}(\vec{\mathcal{D}})$-logic. On the other hand, what concerns operational specifications we have already implemented a tool for the verification of refinements supporting our constructors for implementations; see the reference in Sect. 1. An integration into HETS would allow for a combination of $\mathcal{E}^{\downarrow}(\vec{\mathcal{D}})$ with other provers and heterogeneous institutions.

Acknowledgements

We are very grateful to the anonymous reviewers of the submitted version of this paper for their thorough reading and valuable comments leading to major improvements and even new development ideas.

Funding Open Access funding enabled and organized by Projekt DEAL.

References

[Abr13] Abrial J-R (2013) Modeling in event-B: system and software engineering. Cambridge University Press

[BHW11] Bauer SS, Hennicker R, Wirsing M (2011) Interface theories for concurrency and data. Theor Comput Sci 412(28):3101–3121

[BP18] Bohrer B, Platzer A (2018) A hybrid, dynamic logic for hybrid-dynamic information flow. In: Dawar A, Grädel E (eds) Proc 33rd ann ACM/IEEE symp logic in computer science, pp 115–124. ACM

[Bra10] Braüner T (2010) Hybrid logic and its proof-theory. Appl Log Ser. Springer

[dAH01] de Alfaro L, Henzinger TA (2001) Interface automata. In: Min Tjoa A, Gruhn V (eds) Proc 8th Europ software engineering conf 9th ACM SIGSOFT intl. symp. foundations of software engineering, pp 109–120. ACM

[Dia08] Diaconescu R (2008) Institution-independent model theory. Studies in universal logic. Birkhäuser

[DM16] Diaconescu R, Madeira A (2016) Encoding hybridized institutions into first-order logic. Math Struct Comput Sci 26(5):745–788

[DS07] Diaconescu R, Stefaneas PS (2007) Ultraproducts and possible worlds semantics in institutions. Theor Comput Sci 379(1–2):210–230

[FF02] Fajardo R, Finger M (2002) Non-normal modalisation. In: Balbiani P, Suzuki N-Y, Wolter F, Zakharyaschev M (eds) Advances in modal logic 4, papers from the fourth conference on "advances in modal logic," held in Toulouse, France, 30 September–2 October 2002, pp 83–96. King's College Publications

[FG92] Finger M, Gabbay DM (1992) Adding a temporal dimension to a logic system. J Log Lang Inform 1(3):203–233

[FMP17] Farrell M, Monahan R, Power JF (2017) An institution for event-B. In: James P, Roggenbach M (eds) Rev sel papers 23rd IFIP WG 1.3 intl ws recent trends in algebraic development techniques, vol 10644 of lect notes comp sci, pp 104–119. Springer

[GB92] Goguen JA, Burstall RM (1992) Institutions: abstract model theory for specification and programming. J ACM 39:95–146

[GM14] Groote JF, Mousavi MR (2014) Modeling and analysis of communicating systems. MIT Press

[Gor94] Goranko V (1994) Temporal logic with reference pointers. In: Proc 1st intl conf temporal logic, vol 827 of lect notes comp sci, pp 133–148. Springer

[GR00] Gorrieri R, Rensink A (2000) Action refinement. In: Bergstra JA, Ponse A, Smolka SA (eds) Handbook of process algebra, pp 1047–1147. Elsevier

[GR02] Goguen JA, Rosu G (2002) Institution morphisms. Formal Asp Comput 13(3–5):274–307

[HKT00] Harel D, Kozen D, Tiuryn J (2000) Dynamic logic. MIT Press

[HMK19] Hennicker R, Madeira A, Knapp A (2019) A hybrid dynamic logic for event/data-based systems. In: Hähnle R, van der Aalst WMP (eds) Proc 22nd intl conf fundamental approaches to software engineering, vol 11424 of lect notes comp sci, pp 79–97. Springer

[KMRG15] Knapp A, Mossakowski T, Roggenbach M, Glauer M (2015) An institution for simple UML state machines. In: Egyed A, Schaefer I (eds) Proc 18th intl conf fundamental approaches to software engineering, vol 9033 of lect notes comp. sci, pp 3–18. Springer

[Lam03] Lamport L (2003) Specifying systems: the TLA+ language and tools for hardware and software engineers. Addison-Wesley

[Lyn03] Lynch NA (2003) Input/output automata: basic, timed, hybrid, probabilistic, dynamic, ... In: Amadio RM, Lugiez D (eds) Proc 14th intl conf concurrency theory, vol 2761 of lect notes comp sci, pp 187–188. Springer

[Mac98] Mac Lane S (1998) Categories for the working mathematician, 2nd edn. Springer

[MBHM18] Madeira A, Barbosa LS, Hennicker R, Martins MA (2018) A logic for the stepwise development of reactive systems. Theor Comput Sci 744:78–96

[MCM09] Mouelhi S, Chouali S, Mountassir H (2009) Refinement of interface automata strengthened by action semantics. Electron Notes Theor Comput Sci 253(1):111–126

[MDT09] Mossakowski T, Diaconescu R, Tarlecki A (2009) What is a logic translation? Logica Universalis 3(1):95–124

[MMDB11] Martins MA, Madeira A, Diaconescu R, Barbosa LS (2011) Hybridization of institutions. In: Corradini A, Klin B, Cîrstea C (eds) Proc 4th intl conf algebra and coalgebra in computer science, vol 6859 of lect notes comp sci, pp 283–297. Springer

[MML07] Mossakowski T, Maeder C, Lüttich K (2007) The heterogeneous tool set (Hets). In: Beckert B (ed) Proc 4th intl verification ws, vol 259 of CEUR ws proc CEUR-WS.org

[OMG14] OMG (Object Management Group) (2014) Object constraint language. Standard formal/14-02-03. OMG

[OMG17] OMG (Object Management Group) (2017) Unified modeling language. Standard formal/17-12-05. OMG

[PR06] Poizat P, Royer J-C (2006) A formal architectural description language based on symbolic transition systems and modal logic. J Univ Comput Sci 12(12):1741–1782

[ST88] Sannella D, Tarlecki A (1988) Toward formal development of programs from algebraic specifications: implementations revisited. Acta Inf 25(3):233–281

[ST12] Sannella D, Tarlecki A (2012) Foundations of algebraic specification and formal software development. EATCS Monographs in Theoretical Computer Science. Springer

[tBFGM08] ter Beek MH, Fantechi A, Gnesi S, Mazzanti F (2008) An action/state-based model-checking approach for the analysis of communication protocols for service-oriented applications. In: Leue S, Merino P (eds) Sel papers 12th intl ws formal methods for industrial critical systems, vol 4916 of lect notes comp sci, pp 133–148. Springer

Received 12 December 2019

Accepted in revised form 4 May 2021 by James Woodcock