Hybrid Specification of Reactive Systems: An Institutional Approach*

Alexandre Madeira[1,2,3], José M. Faria[1],
Manuel A. Martins[3], and Luís S. Barbosa[2]

[1] Critical Software S.A., Portugal
[2] Department of Informatics, Minho University
[3] Department of Mathematics, University of Aveiro

Abstract. This paper introduces a rigorous methodology for requirements specification of systems that react to external stimulus by evolving through different operational modes. In each mode different functionalities are provided. Starting from a classical state-machine specification, the envisaged methodology interprets each state as a different mode of operation endowed with an algebraic specification of the corresponding functionality. Specifications are given in an expressive variant of hybrid logic which is, at a later stage, translated into first-order logic to bring into scene suitable tool support. The paper's main contribution is to provide rigorous foundations for the method, framing specification logics as institutions and the translation process as a comorphism between them.

1 Introduction

Motivation. The successful development and deployment of safety-critical, reactive systems, from the early concept and system definition phases, down to implementation and validation, poses a number of challenges that engineers must overcome. From the outset, there are two basic approaches to formally capture requirements for this sort of software: one emphasizes *behaviour* and its evolution; the other focus on *data* and their transformations.

Within the first paradigm, reactive systems are typically specified through (some variant of) *state-machines*. Such models capture system's evolution in terms of event occurrence and its impact in the system internal state configuration. Automata theory, and its more recent, abstract rendering in coalgebraic terms, provide a suitable formalism for both specification and analysis. Crucial notions of bisimulation, minimization and invariant, among others, play a fundamental, long established role in this framework. In the dual, data-oriented

* This research was partially supported by FCT (the Portuguese Foundation for Science and Technology) under contract PTDC/EIA-CCO/ 108302/2008 (the MONDRIAN project) and CIDMA at University of Aveiro. A. Madeira and J. M. Faria worked under contracts SFRH/BDE/ 33650/2009 and SFRH / BDE / 51049 / 2010, two PhD grants jointly supported by FCT and *Critical Software S.A., Portugal*. M. Martins was further supported by the project *Nociones de Completud*, reference FFI2009-09345 (Spain).

G. Barthe, A. Pardo, and G. Schneider (Eds.): SEFM 2011, LNCS 7041, pp. 269–285, 2011.

approach the system's functionality is given in terms of input-output relations modeling operations on *data*. A specification is a theory in a suitable logic, expressed over a signature, which captures its syntactic interface. Its semantics is a class of concrete algebras acting as models for the specified theory [5,18].

In practice, however, both approaches are interconnected: the functionality offered by the system, at each moment, may depend on the stage of its evolution. Such is typically the case of complex, reactive, reconfigurable software.

This paper explores such a interconnection. Starting from a classical state-machine specification, the methodology illustrated in the sequel goes a step further: different states are interpreted as different *modes* of operation and each of them is equipped with an algebraic specification (over the system's interface) of the corresponding functionality. Technically, specifications become *structured* state-machines, states denoting *algebras*, rather than *sets*.

The following paragraph sums up the envisaged approach. It should be remarked this has been developed in a concrete, industrial context — that of a leading, portuguese IT company, whose mission includes the production of formally certified software for critical systems. Such a context makes effective, but sound tool support a must. As discussed in the sequel, rigorous foundations also (may) lead to fulfill this objective.

Approach and paper outline. The approach proposed in the paper is sketched in Figure 1. The upper plane sums up the envisaged methodology. The block on the left hand side represents the specification framework, structured in two stages, as explained below. The annotation on top — *Hybrid logic* — states the underlying logic. The block on the right concerns verification and analysis of hybrid specifications suitably translated to first order logic (FOL). The translation itself is depicted as a *comorphism* between the two logic systems in presence: hybrid logic, chosen for its expressive power, first order, to benefit from existent verification support. Hybrid logic [2] plays a fundamental role here given its ability to make explicit references, through special symbols called *nominals*, to specific states within a model.

The lower plane of Figure 1 refers to the methodology foundations. Actually, a basic property to require from a specification formalism is its ability to be framed as an *institution* [7,4]. This is not a formal idiosyncrasy: institutions, as abstract, general representations of logical systems, provide modular structuring and parameterization mechanisms which are defined 'once and for all' abstracting from the concrete particularities of the each specification logic [24]. Moreover, several current specification formalisms, notably, CafeOBJ [5], Casl [18] and Hets [20] were designed to take advantage of such a general framework.

Moreover institutions provide a systematic way to relate logics and transport results from one to another [17], which means that a theorem prover for the latter can be used to reason about specifications written in the former. This is achieved through a special class of maps between institutions, referred to as *comorphisms*, as depicted in Figure 1.

The rest of the paper is organized around two main sections: one on the *methodology* (sections 2) and another on *foundations* (section 3). Section 4

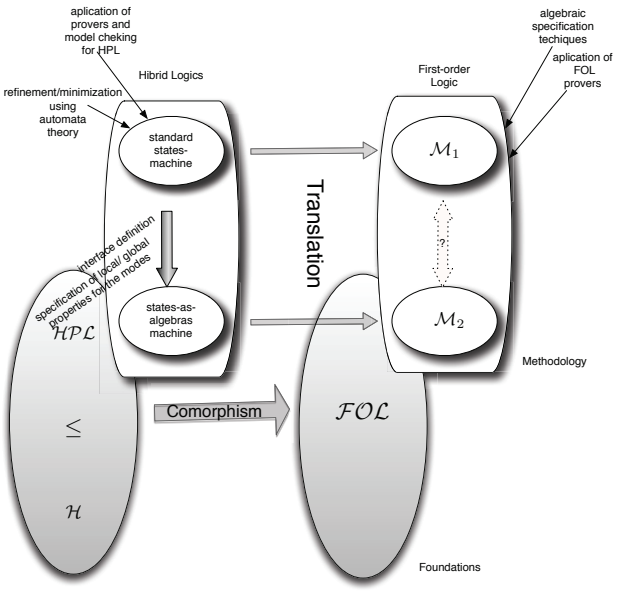

Fig. 1. The approach

discusses current work on suitable tool support based on HETS [20]. Section 5 concludes and provides a few pointers for future work. The reader may found the detailed proofs of all formal claims of the paper on the technical report [13].

2 A Specification Methodology

As stated above, the paper proposes a methodology to the specification and analysis of reactive systems which is intended to be effectively used in an industrial context. The methodology has the following stages, which will be detailed later in the paper:

I (I.1) Express the requirements in *hybrid propositinal logic* (HPL), identifying states and transitions to build a first state-machine; (I.2) Specify local properties as propositions; At this stage, traditional technics of state machine analysis/refinement may be applied, and available reasoning tools for HPL used (see Section 2.1).

II (II.1)Define the actual system's interface through the set of (external) services offered. Technically, this is supported by the definition of a (multi-sorted) first-order signature. (II.2) Express, whenever possible, the attributes of the first machine as functional properties over this signature.

III Translate both specifications into FOL, providing a common ground for testing and verification.

In the sequel the methodology is illustrated in a number of specification fragments of an *automatic cruise control* (ACC) system. The example, small but

self-contained, is taken from [9], where the overall requirements are summarized as follows:

> *"The mode class CruiseControl contains four modes, Off, Inactive, Cruise, and Override. At any given time, the system must be in one of these modes. Turning the ignition on causes the system to leave Off mode and enter Inactive mode, while turning the cruise control level to const when the brake is off and the engine running causes the system to enter Cruise mode. (...) Once cruise control has been invoked, the system uses the automobile's actual speed to determine whether to set the throttle to accelerate or decelerate the automobile, or to maintain the current speed (...)To override cruise control (i.e., enter Override), the driver turns the lever to off or applies the brake".*

2.1 Hybrid Specifications (Stage I)

The requirements for the cruise control system example can be captured by the state machine depicted in Figure 2. This section introduces its specification in propositional hybrid logic (HPL). Such a presentation has the advantage of being compact, unambiguous and closer to the input format of typical verification engines.

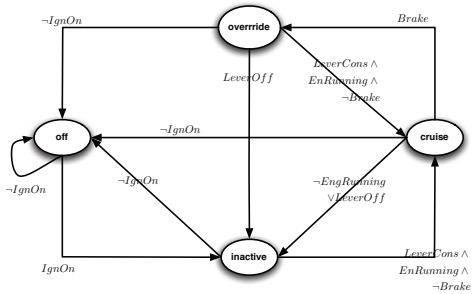

Fig. 2. State-machine of the system

The set of HPL formulas is defined by the following grammar:

$$\varphi, \psi ::= p \mid i \mid \neg\varphi \mid [\lambda]\varphi \mid @_i\varphi \mid \varphi \wedge \psi \mid \varphi \vee \psi \mid \varphi \Rightarrow \psi \tag{1}$$

where λ ranges over a set Λ of modal operators. Models of this logic are state-machines with an additional function state : Nom $\rightarrow S$ which assigns to each nominal a state. This allows explicit reference to particular states in a specification. Thus, models are tuples $\mathcal{P} = \langle S, \text{state}, (R_\lambda)_{\lambda\in\Lambda}, (P_s)_{s\in S}\rangle$ where S is a set of states, $R_\lambda \subseteq S \times S$ is the accessibility relation associated to the modality λ and P_s : Prop $\rightarrow \{\top, \bot\}$ is the function that assigns the propositions on the state $s \in S$. The satisfaction relation is defined as in standard modal logic (e.g. $\mathcal{P} \models^s p$ iff $P_s(p) = \top$; $\mathcal{P} \models^s [\lambda]\varphi$ iff $\mathcal{P} \models^{s'} \varphi$ for any s' such that $(s, s') \in R_\lambda$) adding the following cases related to nominals:

- $\mathcal{P} \models^s @_i\varphi$ iff $\mathcal{P} \models^{\text{state}(i)} \varphi$;
- $\mathcal{P} \models^s i$ iff state$(i) = s$.

Moreover, we abbreviate formulas $\neg[\lambda]\neg\varphi$ and $\langle\lambda\rangle\varphi \wedge [\lambda]\varphi$ to $\langle\lambda\rangle$ and $\langle\lambda\rangle^\circ\varphi$, respectively.

For the running example, a modality $\{next\}$ is introduced to denote the state-machine accessibility relation. Nominals in $\{off, inactive, override, cruise\}$ correspond to the operation modes mentioned in the requirements. Finally, a set of propositions is considered — one for each label in Figure 2. With such signature, transitions are specified as follows:

- $(T_1)@_{off}(\ IgnOn \Rightarrow \langle next\rangle^\circ inactive\ )$
- $(T_2)\neg\ IgnOn \Rightarrow \langle next\rangle^\circ off$
- $(T_3)@_{inactive}(LeverCons \wedge IgnOn \wedge \neg\ Brake \Rightarrow \langle next\rangle^\circ cruise)$
- $(T_4)@_{cruise}(\neg\ EngRunning \vee LeverOff \Rightarrow \langle next\rangle^\circ inactive)$
- $(T_5)@_{cruise}(\ Brake \Rightarrow \langle next\rangle^\circ override)$
- $(T_6)@_{override}(LeverCons \wedge IgnOn) \wedge EngRunning \wedge \neg\ Brake \Rightarrow \langle next\rangle^\circ cruise)$

Local properties can also be expressed resorting to the satisfaction operator $@_i$, for each nominal i, to reference the corresponding state. For instance, the requirement that the engine controls speed decelerating the car if the *speed is high* and maintaining it when it is considered *adequate* is modelled by

- $(L^1_{cruise})@_{cruise}(\ IgnOn \wedge EngRunning \wedge HighSpeed \Rightarrow decel)$
- $(L^2_{cruise})@_{cruise}(IgnOn \wedge EngRunning \wedge AdmissibleSpeed \Rightarrow mantain)$

Finally, admissibility properties, concerning propositions, are also captured. For instance, the fact that *the lever cannot be switched in more than one position at each time*, and similarly for the acceleration and speed modes, is expressed as

- $(A_1)LeverOff \Leftrightarrow \neg\ LeverCons$
- ...
- $(A_4)HighSpeed \Rightarrow \neg\ CruiseSpeed \wedge \neg\ LowSpeed$

2.2 States-as-Algebras Models (Stage II)

The logic. The second stage in the methodology equips each state of the underlying state-machine with an *algebra*, more precisely a first-order structure, to model its local functionality. Therefore, hybrid structures are enriched with a family of first-order structures indexed by the set of states, i.e., they become structures

$$\mathcal{M} = \langle S, state, (R_\lambda)_{\lambda\in\Lambda}, (P_s)_{s\in S}, (A_s)_{s\in S}\rangle$$

where first-order structures in the family $(A_s)_{s\in S}$ are defined over the same signature and universe, say A. Each A_s models the system's behaviour at state $s \in S$.

Definition 1. *Let Σ a first-order signature and X a set of variables for it,* Nom, Prop *and Λ three disjoint sets of nominals, propositions and modalities respectively. The set of* hybrid equational formulas *is defined by the following grammar:*

$$\varphi, \psi ::= p \,|\, i \,|\, t \approx t' \,|\, P(\bar{t}) \,|\, \neg\varphi \,|\, \varphi \star \psi \,|\, [\lambda]\varphi \,|\, @_i\varphi \,|\, \forall x\varphi \tag{2}$$

where $\star \in \{\vee, \wedge, \Rightarrow\}$, p is a proposition, i is a nominal, $t \approx t'$ is a Σ-equation over X, $x \in X$, P is a Σ-predicate of type $s_1, \ldots, s_n$ where $\bar{t} := t_1, \ldots, t_n$ and $t_i \in (T_\Sigma(X))_{s_i}$.

An assignment for $M = \langle S, state, (R_\lambda)_{\lambda\in\Lambda}, (P_s)_{s\in S}, (A_s)_{s\in S}\rangle$ consists of a (sorted-set) function $g : X \to A$, where A is the carrier set of the first-order structures of $\mathcal{M}$ and X is a set of variables. We write $g \sim^x g'$ if for any variable $y \neq x$, $g(y) = g'(y)$. Note that the assignment $g : X \to A$ induces an S-family of assignments $g^s : T_\Sigma(X) \to A$ defined, for any $x \in X$, by $g^s(x) = g(x)$ and, for each term $f(t_1, \ldots, t_n)$, by $g^s(f(t_1, \ldots, t_n)) = f^{A_s}(g^s(t_1), \ldots, g^s(t_n))$.

Definition 2. *Let $\mathcal{M} = \langle S, state, (R_\lambda)_{\lambda\in\Lambda}, (P_s)_{s\in S}, (A_s)_{s\in S}\rangle$ be an hybrid structure. For any assignment $g : X \to A$, the* satisfaction relation *is recursively defined as follows:*

- $\mathcal{M}, g \models^s i$ *if* $state(i) = s$;
- $\mathcal{M}, g \models^s p$ *if* $P_s(p) = \top$;
- $\mathcal{M}, g \models^s t \approx t'$ *if* $A_s \models t \approx t'[g]$ *i.e., if* $g^s(t) = g^s(t')$;
- $\mathcal{M}, g \models^s Q(t_1, \ldots, t_n)$ *if* $A_s \models Q(t_1, \ldots, t_n)[g]$, *i.e., if* $Q^{A_s}(g^s(t_1), \ldots, g^s(t_n))$;
- $\mathcal{M}, g \models^s \rho \vee \rho'$ *if* $\mathcal{M} \models^s \rho$ *or* $\mathcal{M} \models^s \rho'$; *and similarly for the remaining boolean connectives;*
- $\mathcal{M}, g \models^s \forall x \rho$ *if, for any assignment* $g' : X \to A$, *if* $g \sim^x g'$, *one has* $\mathcal{M}, g' \models^s \rho$;
- $\mathcal{M}, g \models^s [\lambda]\rho$ *if, for any* $s' \in S$ *such that* $(s, s') \in R_\lambda$, *one has* $\mathcal{M} \models^{s'} \rho$.

We write $\mathcal{M} \models^s \rho$ when for any assignment $g : X \to A$, $\mathcal{M}, g \models^s \rho$ and $\mathcal{M}, g \models \rho$ when for any $s \in S$, $\mathcal{M}, g \models^s \rho$.

In order to model the system's functionality, as provided by the car artifact, we resort to a classical algebraic specification. This entails the need for introducing data types able to support the envisaged notions of *time*, *speed* and *acceleration*. In the running example integer numbers, with the usual operations and predicates $\{+, \leq, \geq, <, >\}$, can do the job.

spec TimeSort =Int **with sort** *Int* $\mapsto$ *time*, **ops** $0 \mapsto$ *init*, *suc* $\mapsto$ *after* **end**
spec SpeedSort =Int **with sort** *Int* $\mapsto$ *speed* **end**
spec AcellSort =Int **with sort** *Int* $\mapsto$ *accel* **end**

Thus, the operation *Pedal* models the accelerations applied by the driver at each moment. On the other hand, *Automatic* captures accelerations applied on the engine by the ACC, and *CurrentSpeed* records the current speed. Finally, constant $MaxCruiseSpeed$ represents the maximum speed allowed on the ACC mode:

spec ACCSign =
  TimeSort **and** SpeedSort **and** AcellSort
then ops *Pedal* : *time* $\to$ *accel*;
  Automatic : *time* $\to$ *accel*;
  Speed : *speed* $\times$ *accel* $\to$ *speed*;
  CurrentSpeed : *time* $\to$ *speed*;
  MaxCruiseSpeed : *speed*

There are properties that globally hold, in all the configurations of the system. For instance,

$\forall\ s : speed;\ a : accel;\ t : time$
• $(G_1)\ Speed(s,\ a) \geq 0$
• $(G_2)\ CurrentSpeed(t) = 0 \wedge Pedal(t) \geq 0 \Rightarrow CurrentSpeed(after(t)) \geq 0$
• $(G_3)\ Pedal(t) \geq 0 \Leftrightarrow CurrentSpeed(t) < CurrentSpeed(after(t))$
• $(G_4)\ Speed(s,\ a) = s \Leftrightarrow a = 0$
• $(G_5)\ CurrentSpeed(after(t)) = Speed(CurrentSpeed(t), Pedal(t))$

Local properties. Differently from the properties above, local requirements hold only at particular configurations. Let us explore some of them. First, in state off, it is required that speed and acceleration are null and no other operations in the interface react:

$\forall\ t : time;\ s : speed;\ a : accel$
• $(L^1_{off})\ @_{off}\ CurrentSpeed(t) = 0$
• $(L^2_{off})\ @_{off}\ Speed(s,\ a) = 0$

In state *inactive*, the speed and acceleration depend on the accelerations automatically introduced in the system, i.e,

$\forall\ s : speed;\ a : accel$
• $(L^1_{inactive})\ @_{inactive}\ Speed(s,\ a) = s + a$

$\forall\ t{:}\ time;\ s : speed;\ a : accel$
• $(L^{1'}_{cruise})\ @_{cruise}[CurrentSpeed(t) > MaxCruiseSpeed \Rightarrow Automatic(after(t)) < 0]$
• $(L^{2'}_{cruise})\ @_{cruise}[CurrentSpeed(t) \leq MaxCruiseSpeed \Leftrightarrow Automatic(after(t)) = 0]$
• $(L^3_{cruise})@_{cruise}\ Speed(s,\ a) = s + a$
• $(L^4_{cruise})@_{cruise}\ Pedal(t) \geq 0 \Rightarrow Pedal(t) = Automatic(t)$

An interesting feature in this example is that properties local to states *override* and off do coincide. The system's behaviour on both states only differs in what concerns the definition of the allowed transitions. The latter are dealt as follows.

Transitions specification. To specify state transitions we simply resort to the state-machine built in Stage I, through axioms $(T_1), \ldots, (T_n)$ from Section 2.1. However, some propositions may now be expressed by means of algebraic properties of local states. For instance, we may replace (T_4) by

$\forall\ t{:}\ time;$
• $(T_{4'})@_{cruise}[CurrentSpeed(t) = 0 \Rightarrow \langle next\rangle^{\circ}(inactive \wedge CurrentSpeed(after(t)) = 0)]$
• $(T_{4''})@_{cruise}[LeverOff \Rightarrow \langle next\rangle^{\circ} inactive].$

Furthermore, the fact that when ACC is activated by transition T_6, the speed should to be maintained, is captured by

$\forall\ t{:}\ time;\ \forall\ s{:}\ speed$
• $(T_{6'})@_{override}[(LeverCons \wedge CurrentSpeed(t) = s\ \wedge\ s \geq 0) \Rightarrow$
$\langle next\rangle^{\circ}(cruise \wedge CurrentSpeed(after(t)) = s)].$

3 Foundations

3.1 Going "institutional"

Dealing with the sort of specifications produced in Stages I and II above, entails the need for a *uniform* specification framework in which both equational properties of data types, modal properties of transitions and local properties of states

can be expressed and verified. The canonical way to do it is through the notion of an *institution* [7,4], as an abstract representation of a logical system, encompassing syntax, semantics and satisfaction. Let us recall the formal definition:

Definition 3 (Institution). *An* institution $\left(\mathrm{Sign}^{\mathcal{I}}, \mathrm{Sen}^{\mathcal{I}}, \mathrm{Mod}^{\mathcal{I}}, (\models^{\mathcal{I}}_{\Sigma})_{\Sigma \in |\mathrm{Sign}^{\mathcal{I}}|}\right)$ *consists of*

- *a category* $\mathrm{Sign}^{\mathcal{I}}$ *whose objects are called* signatures.
- *a functor* $\mathrm{Sen}^{\mathcal{I}} : \mathrm{Sign}^{\mathcal{I}} \rightarrow \mathsf{Set}$ *giving for each signature a set whose elements are called* sentences *over that signature.*
- *a functor* $\mathrm{Mod}^{\mathcal{I}} : (\mathrm{Sign}^{\mathcal{I}})^{op} \rightarrow \mathsf{CAT}$*, giving for each signature* Σ *a category whose objects are* Σ-models*, and whose arrows the corresponding* Σ-morphisms*, and*
- *a* satisfaction relation $\models^{\mathcal{I}}_{\Sigma} \subseteq |\mathrm{Mod}^{\mathcal{I}}(\Sigma)| \times \mathrm{Sen}^{\mathcal{I}}(\Sigma)$

such that for each morphism $\varphi : \Sigma \rightarrow \Sigma' \in \mathrm{Sign}^{\mathcal{I}}$ *, the satisfaction condition*

$$M' \models^{\mathcal{I}}_{\Sigma'} \mathrm{Sen}^{\mathcal{I}}(\varphi)(\rho) \; iff \; \mathrm{Mod}^{\mathcal{I}}(\varphi)(M') \models^{\mathcal{I}}_{\Sigma} \rho \tag{3}$$

holds for each $M' \in |Mod^{\mathcal{I}}(\Sigma')|$ *and* $\rho \in \mathrm{Sen}^{\mathcal{I}}(\Sigma)$*.*

A well known example of institution is the institution of first order logic, denoted in the sequel by $\mathcal{FOL}$ (see [4] for a detailed account). Institutions provide a suitable setting to do *abstract specification theory* [24], structuring any kind of specifications through combinators which are institution-independent, i.e. not tied to a specific logic system. In CASL [18], for example, such combinators allow the construction of basic specifications, by defining a signature and a set of sentences, the union of specifications, and the derivation and translation of specifications along signature morphisms. The use of this set of (abstract) combinators, makes possible to approach, in a uniform way and trough the same theory, systems expressed in completely different logics.

Therefore, our first aim concerning *foundations* is to prove that the proposed specification formalism may be framed on this big picture of institution theory. Let start by collecting the necessary ingredients to define a suitable institution $\mathcal{H}$.

Category $\mathrm{SIGN}^{\mathcal{H}}$: Signatures are tuples $\langle(\Sigma, X), \mathrm{Nom}, \mathrm{Prop}, \Lambda\rangle$ where Σ is a first-order logic signature, X is a set of first-order variables and Nom, Prop and Λ are (disjoint) sets of symbols of nominals, propositions and modalities. Signature morphisms

$$\langle(\Sigma, X), \mathrm{Nom}, \mathrm{Prop}, \Lambda\rangle \xrightarrow{\varphi} \langle(\Sigma', X'), \mathrm{Nom}', \mathrm{Prop}', \Lambda'\rangle$$

are tuples $\varphi = (\varphi_{\mathrm{Sig}}, \varphi_{\mathrm{Nom}}, \varphi_{\mathrm{Prop}}, \varphi_{\mathrm{MS}})$ where $\varphi_{\mathrm{Nom}} : \mathrm{Nom} \rightarrow \mathrm{Nom}'$, $\varphi_{\mathrm{Prop}} : \mathrm{Prop} \rightarrow \mathrm{Prop}'$ and $\varphi_{\mathrm{MS}} : \Lambda \rightarrow \Lambda'$ are functions and $\varphi_{\mathrm{Sig}} : (\Sigma, X) \rightarrow (\Sigma', X')$ is a morphism in $\mathcal{FOL}$, i.e., a tuple $\varphi_{\mathrm{Sig}} = (\varphi^{sort}_{\mathrm{Sig}}, \varphi^{op}_{\mathrm{Sig}}, \varphi^{pred}_{\mathrm{Sig}}, \varphi^{var}_{\mathrm{Sig}})$

- for any operation $f \in \Sigma_{s_1 \ldots s_n, s}$, $\varphi^{op}_{\mathrm{Sig}}(f) \in \Sigma'_{\varphi^{sort}_{\mathrm{Sig}}(s_1) \ldots \varphi^{sort}_{\mathrm{Sig}}(s_n), \varphi^{sort}_{\mathrm{Sig}}(s)}$;

- for any predicate $Q \in \Sigma_{s_1 \ldots s_n}$, $\varphi^{pred}_{\mathrm{Sig}}(Q) \in \Sigma'_{\varphi^{sort}_{\mathrm{Sig}}(s_1)\ldots\varphi^{sort}_{\mathrm{Sig}}(s_n)}$;
- for any variable $x \in X_s$, $\varphi^{var}_{\mathrm{Sig}}(x) \in X'_{\varphi^{sort}_{\mathrm{Sig}}(s)}$.

Functor $\mathrm{SEN}^{\mathcal{H}}$: This functor maps a signature $\Delta = \langle(\Sigma, X), \mathrm{Nom}, \mathrm{Prop}, \Lambda\rangle$ into the set of hybrid sentences, i.e., on the subset of bonded-variables formulas of Definition 1, and a morphism

$$\langle(\Sigma, X), \mathrm{Nom}, \mathrm{Prop}, \Lambda\rangle \xrightarrow{\varphi} \langle(\Sigma', X'), \mathrm{Nom}', \mathrm{Prop}', \Lambda'\rangle$$

into the sentence translation

$$\mathrm{Sen}^{\mathcal{H}}(\langle(\Sigma, X), \mathrm{Nom}, \mathrm{Prop}, \Lambda\rangle) \xrightarrow{\mathrm{Sen}^{\mathcal{H}}(\varphi)} \mathrm{Sen}^{\mathcal{H}}(\langle(\Sigma', X'), \mathrm{Nom}', \mathrm{Prop}', \Lambda'\rangle)$$

recursively defined as follows

- $\mathrm{Sen}^{\mathcal{H}}(\varphi)(\rho) = \mathrm{Sen}^{\mathcal{FOL}}(\varphi_{\mathrm{Sig}})(\rho)$ for any $\rho \in \mathrm{Sen}^{\mathcal{FOL}}(\Sigma)$;
- $\mathrm{Sen}^{\mathcal{H}}(\varphi)(i) = \varphi_{\mathrm{Nom}}(i)$, $i \in \mathrm{Nom}$;
- $\mathrm{Sen}^{\mathcal{H}}(\varphi)(p) = \varphi_{\mathrm{Prop}}(p)$, $p \in \mathrm{Prop}$;
- $\mathrm{Sen}^{\mathcal{H}}(\varphi)(t \approx t') = \varphi^{term}(t) \approx \varphi^{term}(t)$, where $\varphi^{term} : T_{\Sigma}(X) \to T_{\Sigma'}(X')$ is a function recursively defined as follows
  - $\varphi^{term}(x) = \varphi^{var}_{\mathrm{Sig}}(x)$ for $x \in X$;
  - $\varphi^{term}(f(t_1, \ldots, t_n)) = \varphi^{op}_{\mathrm{Sig}}(f)(\varphi^{term}(t_1), \ldots, \varphi^{term}(t_n))$, for any $f \in \Sigma_{s_1 \ldots s_n, s}$, $t_i \in (T_{\Sigma}(X))_{s_i}$.
- $\mathrm{Sen}^{\mathcal{H}}(\varphi)(Q(t_1, \ldots, t_n)) = \varphi^{pred}_{\mathrm{Sig}}(Q)(\varphi^{term}(t_1), \ldots, \varphi^{term}(t_n))$;
- $\mathrm{Sen}^{\mathcal{H}}(\varphi)(\neg\rho) = \neg\mathrm{Sen}^{\mathcal{H}}(\varphi)(\rho)$;
- $\mathrm{Sen}^{\mathcal{H}}(\varphi)(\rho \odot \rho') = \mathrm{Sen}^{\mathcal{H}}(\varphi)(\rho) \odot \mathrm{Sen}^{\mathcal{H}}(\varphi)(\rho')$, $\odot \in \{\vee, \wedge, \to\}$;
- $\mathrm{Sen}^{\mathcal{H}}(\varphi)(@_i\rho) = @_{\varphi_{\mathrm{Nom}}(i)}\mathrm{Sen}^{\mathcal{H}}(\varphi)(\rho)$;
- $\mathrm{Sen}^{\mathcal{H}}(\varphi)([\lambda]\rho) = [\varphi_{\mathrm{MS}}(\lambda)]\mathrm{Sen}^{\mathcal{H}}(\varphi)(\rho)$;
- $\mathrm{Sen}^{\mathcal{H}}(\varphi)(\forall x\rho) = \forall\varphi^{var}_{\mathrm{Sig}}(x)\mathrm{Sen}^{\mathcal{H}}(\varphi)(\rho)$.

Functor $\mathrm{Mod}^{\mathcal{H}}$: This functor maps each signature $\langle(\Sigma, X), \mathrm{Nom}, \mathrm{Prop}, \Lambda\rangle$ to a category whose models are the hybrid structures $\mathcal{M} = \langle S, state, (R_\lambda)_{\lambda\in\Lambda}, (P_s)_{s\in S}, (A_s)_{s\in S}\rangle$ defined above. Morphisms between models $\langle S, state, (R_\lambda)_{\lambda\in\Lambda}, (P_s)_{s\in S}, (A_s)_{s\in S}\rangle$ and $\langle S', state', (R'_\lambda)_{\lambda\in\Lambda}, (P'_s)_{s\in S'}, (A'_s)_{s\in S'}\rangle$ consists of pairs (h_{st}, h_{mod}) such that

- h_{mod} is an S-family $\left(h_{mod_s} : A_s \to A'_{h_{st}(s)}\right)_{s\in S}$ of first-order structures morphisms;
- $P_s(p) = P'_{h_{st}(s)}(\varphi_{\mathrm{Prop}}(p))$;
- $h_{st} : S \to S'$ is a function such that
  - $(s, s') \in R_\lambda$ implies that $(h_{st}(s), h_{st}(s')) \in R'_\lambda$,
  - $state'(n) = h_{st}(state(n))$,

Functor $\mathrm{Mod}^{\mathcal{H}}$ maps each morphism

$$\langle(\Sigma, X), \mathrm{Nom}, \mathrm{Prop}, \Lambda\rangle \xrightarrow{\varphi} \langle(\Sigma', X'), \mathrm{Nom}', \mathrm{Prop}', \Lambda'\rangle$$

into the reduct functor

$$\mathrm{Mod}^{\mathcal{H}}(\langle(\Sigma, X), \mathrm{Nom}, \mathrm{Prop}, \Lambda\rangle) \xleftarrow{\mathrm{Mod}^{\mathcal{H}}(\varphi)} \mathrm{Mod}^{\mathcal{H}}(\langle(\Sigma', X'), \mathrm{Nom}', \mathrm{Prop}', \Lambda'\rangle)$$

that maps each $\langle(\Sigma', X'), \mathrm{Nom}', \mathrm{Prop}', \Lambda'\rangle$-model $\langle S', state', (R'_\lambda)_{\lambda\in\Lambda'}, (P'_s)_{s\in S'}, (A'_s)_{s\in S'}\rangle$ into the $\langle\Sigma, \mathrm{Nom}, \mathrm{Prop}, \Lambda\rangle$-model $\langle S, state, (R_\lambda)_{\lambda\in\Lambda}, (P_s)_{s\in S}, (A_s)_{s\in S}\rangle$ such that

- $S = S'$;
- $state(n) = state'(\varphi_{\mathrm{Nom}}(n))$ for any $n \in \mathrm{Nom}$;
- $R_\lambda = R'_{\varphi_{\mathrm{MS}}(\lambda)}$ for any $\lambda \in \Lambda$;
- $A_s = \mathrm{Mod}^{\mathcal{FOL}}(\varphi_{\mathrm{Sig}})(A'_s)$ for any $s \in S$, where $\mathrm{Mod}^{\mathcal{FOL}}(\varphi_{\mathrm{Sig}})$, the reduct notion on the institution of first-order logic, consists of the classical reduct notion on first-order structures;
- $P_s(p) = P'_s(\varphi_{\mathrm{Prop}}(p))$ for any $p \in \mathrm{Prop}$

Satisfaction $\models^{\mathcal{H}}$: Satisfaction is the restriction of Definition 2 to sentences.

Theorem 1. *Let $\Delta = ((\Sigma, X), \mathrm{Nom}, \mathrm{Prop}, \Lambda)$ and Δ' two $\mathcal{H}$-signatures and $\varphi : \Delta \to \Delta'$ a morphism of signatures. For any $\rho \in \mathrm{Sen}^{\mathcal{H}}(\Delta)$, $\mathcal{M}' = \langle S', state', R_{\Lambda'}, (P'_s)_{s\in S'}, (A'_s)_{s\in S'}\rangle \in |\mathrm{Mod}^{\mathcal{H}}(\Delta')|$, and $s \in S$,*

$$\mathrm{Mod}^{\mathcal{H}}(\varphi)(\mathcal{M}'), g \models^s \rho \text{ iff } \mathcal{M}', g' \models^s \mathrm{Sen}^{\mathcal{H}}(\varphi)(\rho).$$

where, for any $x \in X$, $g(x) = g'(\varphi^{var}_{\mathrm{Sig}}(x))$.

Proof. The proof is done by induction on the structures of sentences.

The satisfaction condition for $\mathcal{H}$ follows from a well known fact, which states that satisfaction of a formula only depends on assignment of free variables. Actually,

Corollary 1 (Satisfaction condition). *Let $\Delta = ((\Sigma, X), \mathrm{Nom}, \mathrm{Prop}, \Lambda)$ and Δ' be two $\mathcal{H}$-signatures and $\varphi : \Delta \to \Delta'$ a morphism of signatures. For any $\rho \in \mathrm{Sen}^{\mathcal{H}}(\Delta)$, $\mathcal{M}' = \langle S', state', R_{\Lambda'}, (P'_s)_{s\in S'}, (A'_s)_{s\in S'}\rangle \in |\mathrm{Mod}^{\mathcal{H}}(\Delta')|$,*

$$\mathrm{Mod}^{\mathcal{H}}(\varphi)(\mathcal{M}') \models \rho \text{ iff } \mathcal{M}' \models \mathrm{Sen}^{\mathcal{H}}(\varphi)(\rho).$$

Therefore,

Corollary 2. $(\mathrm{Sign}^{\mathcal{H}}, \mathrm{Sen}^{\mathcal{H}}, \mathrm{Mod}^{\mathcal{H}}, \models^{\mathcal{H}})$ *is an institution.*

Finally, observe that models, language and satisfaction presented on Section 2.1 also constitute an institution. This institution is similarly defined, by forgetting the first-order signature from hybrid signatures, the state-family of first-order structures from models and the equations and quantifications from sentences. By obvious reasons, we call this the *institution of propositional hybrid logic* and write $\mathcal{HPL}$.

3.2 Translating to $\mathcal{FOL}$ (Stage III)

Stage III in the envisaged methodology was not discussed in section 2. Actually, from a methodological point of view it is rather straightforward: a translation of specifications to a well-known first order setting. Technically, however, this can be stated in a very precise way as a *comorphism*. Comorphims play, at the institutional level, the role of logical translations, lifting specifications expressed within different institutions to a common level [17]. Therefore, any tools, namely proof assistants, available at the target institution, can be borrowed by the source one. Formaly,

Definition 4 (Comorphism). *Given institutions* $\mathcal{I} = (\text{Sign}, \text{Sen}, \text{Mod}, \models)$ *and* $\mathcal{I}' = (\text{Sign}', \text{Sen}', \text{Mod}', \models')$ *a* comorphism $(\Phi, \alpha, \beta) : \mathcal{I} \to \mathcal{I}'$ *consists of*

1. *a functor* $\Phi : \text{Sign} \to \text{Sign}'$,
2. *a natural transformation* $\alpha : \text{Sen} \Rightarrow \Phi; \text{Sen}'$, *and*
3. *a natural transformation* $\beta : \Phi^{op}; \text{Mod}' \Rightarrow \text{Mod}$

such that the following satisfaction condition *holds*

$$M' \models'_{\Phi(\Sigma)} \alpha_\Sigma(\rho) \quad \textit{iff} \quad \beta_\Sigma(M') \models_\Sigma \rho$$

for each signature $\Sigma \in |\text{Sign}|$, $\Phi(\Sigma)$*-model* M', *and* Σ*-sentence* ρ.

The comorphism is conservative *whenever, for each* Σ*-model* M *in* $\mathcal{I}$, *there exists a* $\Phi(\Sigma)$*-model* M' *in* $\mathcal{I}'$ *such that* $M = \beta_\Sigma(M')$.

Note that the comorfism conservativeness is necessary to borrow institutions proof support since it entails that $\Gamma \models_\Sigma \rho$ iff $\alpha_\Sigma(\Gamma) \models'_{\Phi(\Sigma)} \alpha(\rho)$, when we just have the left-right implication on its absence.

In this sub-section, we establish a comorphism from $\mathcal{H}$ into $\mathcal{FOL}$. The translation procedure is based on the addition of a special sort to represent states. Hence, in order to 'collapse' every local state algebra in a unique structure, the signature of all operations and predicates is enriched with an argument of this sort. Moreover, nominals are regarded as constants over ST, modalities as usual first-order relations and propositions as unary predicates over ST. For that we have a functor

$$\begin{aligned} \Phi : \quad & \text{Sign}^{\mathcal{H}} \longrightarrow \text{Sign}^{\mathcal{FOL}} \\ & \langle(\Sigma, X), \text{Nom}, \text{Prop}, \Lambda\rangle \longmapsto \big(\langle S^\Sigma + \{\text{ST}\}, \overline{F^\Sigma} + \overline{\text{Nom}}, \overline{P^\Sigma} + \overline{\text{Prop}} + \overline{\Lambda}\rangle, \bar{X}\big), \end{aligned}$$

where $\Sigma = (S^\Sigma, F^\Sigma, P^\Sigma)$ and

- $\overline{F_\Sigma} = \begin{cases} (\overline{F_\Sigma})_{\text{ST}w \to s} = (F_\Sigma)_{w \to s}, \text{ for any } s \in S_\Sigma, w \in S_\Sigma^* \\ \emptyset, \text{ for the other cases} \end{cases}$;
- $\overline{P_\Sigma} = \begin{cases} (\overline{P_\Sigma})_{\text{ST}w} = (P_\Sigma)_w, \text{ for any } w \in S_\Sigma^*; \\ \emptyset, \text{ for the other cases} \end{cases}$
- $\overline{\text{Nom}} = \{c_i : \ \to \text{ST} \mid i \in \text{Nom}\}$;
- $\overline{\text{Prop}} = \{\bar{p} : \ \text{ST} \mid p \in \text{Prop}\}$;
- $\overline{\Lambda} = \{\lambda : \ \text{ST}^n \mid \lambda \in \Lambda_n\}$.
- $\overline{X} = \begin{cases} \overline{X}_{sort} = X_{sort}, \text{ for any } sort \in S^\Sigma; \\ \overline{X}_{\text{ST}} = \{w, v\} \end{cases}$

Natural transformation $\beta : \Phi^{op}; \text{Mod}^{\mathcal{FOL}} \Rightarrow \text{Mod}^{\mathcal{H}}$ maps each first-order structure $(M; M_{\bar{F}} + M_{\overline{\text{Nom}}}; M_{\bar{P}} + M_{\overline{\text{Prop}}} + M_{\bar{\Lambda}}) \in \text{Mod}(\langle S_\Sigma + \{\text{ST}\}, \overline{F_\Sigma} + \overline{\text{Nom}}, \overline{P_\Sigma} + \overline{\text{Prop}} + \overline{\Lambda}\rangle)$ into

$$\langle S, state, R_\Lambda, (P_s)_{s \in S}, (A_s)_{s \in S}\rangle \xleftarrow{\beta_{b\langle F, \text{Nom}, \text{Prop}, \Lambda\rangle}} \langle M; M_{\bar{F}} + M_{\overline{\text{Nom}}}; M_{\bar{P}} + M_{\overline{\text{Prop}}} + M_{\bar{\Lambda}}\rangle \ ,$$

where for any $i \in \mathrm{Nom}$, $state(i) = c_i^M$, for any $\lambda \in \Lambda$, $R_\lambda = R_\lambda^M$. Moreover, A_s, $s \in S$ is a first-order structure whose carrier set is A^{S_Σ}; functions $f \in F^\Sigma_{s_1 \ldots s_n, s}$ and predicates $Q \in P^\Sigma_{s_1,\ldots,s_n}$ are defined for each $u_i \in U$, $i \leq n$, by $f^{A_s}(u_1, \ldots, u_n) = \bar{f}^M(s, u_1, \ldots, u_n)$ and $Q^{A_s}(u_1, \ldots, u_n) = \bar{P}^M(s, u_1, \ldots, u_n)$ respectively. The family $(P_s)_{s \in S}$, is defined, for each s as $P_s(p) = \top$ iff $\bar{p}^M(s)$.

Natural transformation $\alpha : \mathrm{Sen}^{\mathcal{H}} \Rightarrow \Phi; \mathrm{Sen}^{\mathcal{FOL}}$ is defined for each $(F, \mathrm{Nom}, \Lambda)$-sentence by $\alpha(\rho) = (\forall w)\alpha_w(\rho)$, where w is a variable of ST and α_w is recursively defined as follows:

$$
\begin{array}{rll}
\alpha_w(t \approx t') = \mathcal{T}_w(t) \approx \mathcal{T}_w(t') & & t, t' \in (T_\Sigma(x))_s, s \in S^\Sigma \\
\alpha_w(Q(t_1, \ldots, t_n)) = \bar{Q}(w, \mathcal{T}_w(t_1), \ldots, \mathcal{T}_w(t_n)) & & Q \in P^\Sigma_{s_1,\ldots,s_n}, t_i \in (T_\Sigma(X))_{s_i} \\
\alpha_w(i) = c_i \approx w, & & i \in \mathrm{Nom} \\
\alpha_w(p) = \bar{p}(w), & & p \in \mathrm{Prop} \\
\alpha_w(@_i\rho) = \alpha_{c_i}(\rho), & & \\
\alpha_w([\lambda]\rho) = (\forall v)[(w, v) \in R_\lambda \rightarrow \alpha_v(\rho)], & & \lambda \in \Lambda \\
\alpha_w(\neg\rho) = \neg\alpha_w(\rho) & & \\
\alpha_w(\rho \odot \rho') = \alpha_w(\rho) \odot \alpha_w(\rho'), & & \odot \in \{\vee, \wedge, \rightarrow\} \\
\alpha_w(\forall x \rho) = \forall x\ \alpha_w(\rho) & & x \in X
\end{array}
$$

where $\mathcal{T}_w : T_\Sigma(X) \rightarrow T_{\overline{\Sigma}}(\overline{X})$, for $\overline{\Sigma} = (\overline{S^\Sigma}, \overline{F^\Sigma}, \overline{P^\Sigma})$, defined for each variable $x \in X$, $\mathcal{T}_w(x) = x$ and for each $f(t_1, \ldots, t_n) \in T_\Sigma(X)$ by $\mathcal{T}_w(f(t_1, \ldots, t_n)) = \bar{f}(w, \mathcal{T}_w(t_1), \ldots, \mathcal{T}_w(t_n))$.

Theorem 2. *Let $\Delta \in |\mathrm{SIGN}^{\mathcal{H}}|$, $\rho \in \mathrm{SEN}^{\mathcal{H}}$ and $M' \in \mathrm{Mod}^{\mathcal{FOL}}(\Phi(\Delta))$. Then, for α and β defined as above, for any $s \in S$ and any assignment $g : \bar{X} \rightarrow A$ such that whenever $g(w) = s$, we have that*

$$\beta_\Delta(M'), g\restriction_X \models^s_{\mathcal{H}} \rho \text{ iff } M', g \models^{\mathcal{FOL}}_{\Phi(\Delta)} \alpha_w(\rho). \tag{4}$$

Proof. The proof is done by induction on the structures of sentences.

As direct consequence we have the general satisfaction condition for comorphisms:

Corollary 3 (Comorphism satisfaction condition). *Let $\Delta \in |\mathrm{SIGN}^{\mathcal{H}}|$, $\rho \in \mathrm{SEN}^{\mathcal{H}}$ and $M' \in \mathrm{Mod}^{\mathcal{FOL}}(\Phi(\Delta))$. Then, for α and β defined as above we have that,*

$$\beta_\Delta(M') \models^{\mathcal{H}}_\Delta \rho \text{ iff } M' \models^{\mathcal{FOL}}_{\Phi(\Delta)} \alpha_\Delta(\rho). \tag{5}$$

Moreover it is conservative: this is directly entailed by the assumption that states have constant domains. It is straitforward to see that, we may define a comorphism from $\mathcal{HPL}$ into $\mathcal{FOL}$ from the presented one. This is achieved by forgetting the first-order components of the signatures and models and by restricting α to the hybrid propositional formulas.

Recalling our running example, we end up with the signature

ops

$Speed^* : st^* \times speed \times accel \rightarrow speed$; $Pedal^* : st^* \times time \rightarrow accel$;...

pred

$next : st^* \times st^*$; $IgnOn^* : st^*$; ...

Note that, now, global properties are universally quantified, and local properties take as state argument the respective nominal. For instance, global properties (G_1) and (G_2) are translated into

$\forall\ s : speed;\ w : st^*;\ a : accel; t : time$
$\bullet\ (G_{1^*})\ \geq^*(w\ , Speed^*(w,\ s,\ a),\ 0^*(w))$
$\bullet\ (G_{2^*})\ CurrentSpeed^*(w,t) = 0^*(w) \wedge \geq^*(w,\ Pedal^*(w,t),\ 0^*(w)).$

and local properties (L^1_{off}) and (L^4_{cruise}), into

$\forall\ t : time$
$\bullet\ (L^{1^*}_{off})\ CurrentSpeed^*(off,t) = 0^*(off)$
$\bullet\ (L^{4^*}_{cruise})\ \geq^*(cruise, Pedal^*(cruise,t), 0^*(cruise)) \Rightarrow Pedal(cruise,t) = Automatic^*(cruise,t).$

For instance, transition (T_1) is expressed by

$\bullet (T_{1*})\ IgnOn(off) \Rightarrow$
$[(\forall w : st^*)\ (off, w) \in next \Rightarrow inactive = w \wedge (\exists w' : st^*)\ (off, w') \in next \Rightarrow inactive = w']$,
i.e.,
$\bullet IgnOn(off) \Rightarrow (off, inactive) \in next.$

4 Tool Support

A central ingredient for the successful integration of a formal method in the industrial practice is the existence of effective tool support.

Certainly hybrid specifications produces in Stage I of our methodology can be anchored on recent implementations of logical calculus for HPL (see e.g. HTab [11], HyLoTab [25] and Spartacus [8]). Moreover, model checking for HPL models is also an active research issue (e.g. [12,10]).

Our focus is, however, a different, somehow more standard, one: hybrid specifications are translated to $\mathcal{FOL}$ through a suitable comorphism. This solution provides a uniform first order logical framework for analysis and verification supporting the whole methodology. Moreover, to the best of our knowledge, richer versions of hybrid logic, as required at Stage II, lack effective tool support, which makes our approach by translation the only option available. Beyond the conceptual support of institutions theory and the structured specification methodology offered by CASL, we have effective computational tools, to support our sort of specification. On this perspective HETS-*heterogeneous tools set* [20] deserves a special attention.

Using a metaphor of [19], HETS may be seen as a "motherboard" where different "expansion cards" can be plugged. These pieces are individual logics (with their particular analyzers and proof tools) as well as logic translations. To make them *compatible*, logics have to be formalized as institutions and, the corresponding translations, as comorphisms. Therefore, the integration of the hybrid specifications on the HETS framework is legitimate, since all formal requirements (e.g., that institutions exist, that a comorphism can be defined, etc.) are provided in the present work. HETS already integrates parsers, static analyzers and provers for a wide set of individual logics, and manages heterogeneous proofs resorting to the so-called graphs of logics, i.e., graphs whose nodes are logics and, whose edges, are comorphisms between them.

Furthermore, and directly relevant to our methodology, HETS provides a rich support for $\mathcal{FOL}$, and consequently, for $\mathcal{H}$ and $\mathcal{HPL}$. For instance, provers SoftFOL, Spass, MathServe Broker, among others, are already "pluged" into HETS [19], and therefore, all of them provide effective to our specification methodology (see Figure 3). Moreover, we are also able to take advantage of a number of "borrowed" provers from other institutions through comorphisms with source in $\mathcal{FOL}$.

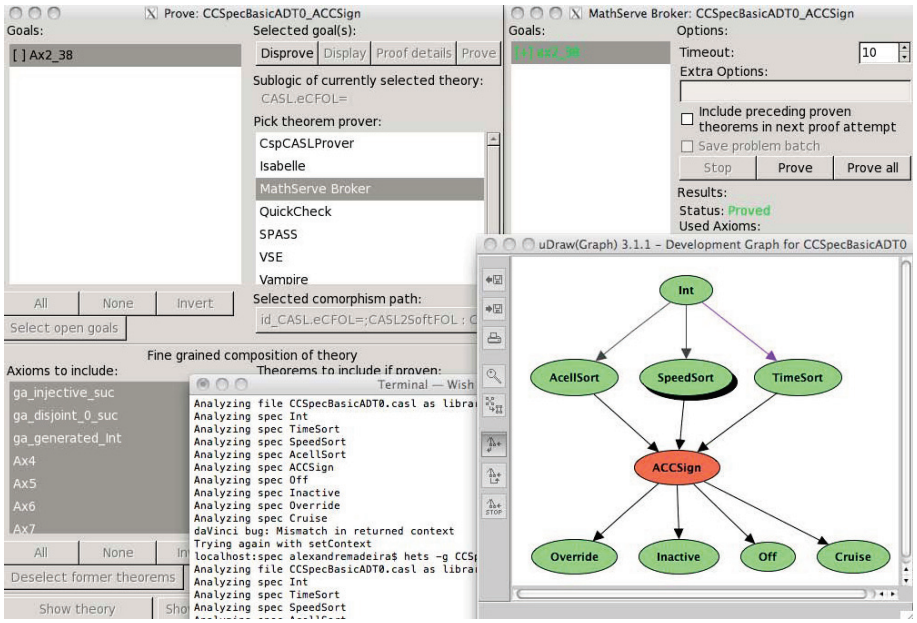

Fig. 3. HETS session

An open issue at this level concerns *verification*. So far model checking of hybrid structures is restricted to propositional hybrid logic [6,12]. The combination of traditional algebraic specification tools, like first-order provers and rewriting engines (e.g. CafeObj [5]), together with provers and model checkers for hybrid logics (e.g. [1,6]) may broaden the scope of application.

5 Conclusions

The paper introduced a rigorous methodology for requirements specification of reactive systems, flexible enough to capture the existence of different operational modes at each stage of evolution. Variants of hybrid logic provided the right conceptual framework to develop such specifications. At a later stage, such specifications are translated into first-order logic to bring into scene suitable tool support. The paper's main contribution was to provide rigorous foundations for the method, framing specification logics as institutions and the translation process as a comorphism between them.

A lot of work remains to be done. From an experimental point of view, we are conducting case studies with different size and complexity to assess the methodology.

Another line of research is concerned with establishing a precise comparison with approaches to specification with a similar purpose. For instance, many (variations) of state machines may be represented as hybrid models. Moreover, some structured state-machines, such as ASM (Abstract State Machines) [3] can also be represented as our states-as-algebras models. An interesting aspect to explore, is whether the institutions constructed here may provide an uniform platform to reason, in a property-oriented perspective, about these model-oriented specifications. Moreover, recent theoretical developments from the authors justifies to look to the presented methodology in a more broad sense: it proofs in [16] that the hybridization idea presented above can be extended to arbitrary institutions. Trough this result it would be worth to consider, on place of the first-order structures, other kind of semantical models such algebras, temporal frames or even Haskel modules, since all of these structures are objects of some particular institution.

Last but not least, *refinement*. At stage III FOL is used as a common language to support reasoning and verification on models built on stages I and II. It is, therefore, expectable to find a way to use this common platform to formally relate these models. In particular, it would be important to formally assure that requirements specified on the first stage are not violated on the second one. This entails the need for a rigorous formalization of the intuitive arrow "?" of figure 1. A natural candidate to do this job, is the classical concept of refinement from algebraic specifications (e.g. [23]). Throughout this notion, a specification SP refines a specification SP_0 over the same signature, if all the properties satisfied by SP_0 are also satisfied by SP. More generally, when specification signatures are related by a morphism, a translation of properties is in order wrt to the signature morphism.

In general, however, this refinement relation is not adequate. For instance, as suggested on stage II, it is expectable to map a proposition of the state-machine into an equation on the respective states-as-algebras model. These formulas are represented in $\mathcal{FOL}$ by a predicate and an equation, respectively, which cannot be related through signature morphisms (which only relate predicates with predicates and equations with equations). Less conventional approaches to refinement may help to overcome this sort of situations. A possibility we are currently investigating is to resort to logical interpretations, instead of signature morphisms, to direct refinement as studied by the authors in [15,14,22]. Interpretations are multi-functions between the specifications formulas which preserve and reflect logical consequence.

There are others specification frameworks also based on modal versions of first-order logic. For instance, in [21] it is defined a logic (for hybrid systems) based on a dynamical version of first-order logic (over $\mathbb{R}$) with nominals. It is important to note that the semantical paradigm of those approaches is quite different for the proposed here; namely, as usual, they deal with states as values

of system variables on of given moment of execution, evaluated in an unique first-order structure. In our work, it corresponds not to a set of values, but to state-families of first-order structures, modeling the behaviour of all the system functionalities.

References

1. Areces, C., Heguiabehere, J.: Hylores: A hybrid logic prover based on direct resolution. In: Proceedings of Advances in Modal Logic, AiML 2002 (2002)
2. Blackburn, P.: Representation, reasoning, and relational structures: a hybrid logic manifesto. Logic Journal of IGPL 8(3), 339–365 (2000)
3. Börger, E., Stärk, R.: Abstract state machines: A method for high-level system design and analysis. Springer, Heidelberg (2003)
4. Diaconescu, R.: Institution-independent Model Theory. Birkhäuser, Basel (2008)
5. Diaconescu, R., Futatsugi, K.: Logical foundations of CafeOBJ. Theor. Comput. Sci. 285(2), 289–318 (2002)
6. Franceschet, M., de Rijke, M.: Model checking for hybrid logics (with an application to semistructured data). Journal of Applied Logic 4(3), 279–304 (2006)
7. Goguen, J.A., Burstall, R.M.: Institutions: abstract model theory for specification and programming. J. ACM 39, 95–146 (1992)
8. Götzmann, D., Kaminski, M., Smolka, G.: Spartacus: A tableau prover for hybrid logic. Electr. Notes Theor. Comput. Sci. 262, 127–139 (2010)
9. Heitmeyer, C.L., Kirby, J., Labaw, B.G.: The SCR Method for Formally Specifying, Verifying, and Validating Requirements: Tool Support. In: ICSE, pp. 610–611 (1997)
10. Hoareau, C., Satoh, I.: Hybrid logics and model checking: A recipe for query processing in location-aware environments. In: AINA, pp. 130–137. IEEE Computer Society, Los Alamitos (2008)
11. Hoffmann, G., Areces, C.: Htab: a terminating tableaux system for hybrid logic. Electr. Notes Theor. Comput. Sci. 231, 3–19 (2009)
12. Lange, M.: Model checking for hybrid logic. J. of Logic, Lang. and Inf. 18(4), 465–491 (2009)
13. Madeira, A., Faria, J.M., Martins, M.A., Barbosa, L.S.: Hybrid specification of reactive systems: An institutional approach (extended version). Technical Report CCTC-11-03, University of Minho (July 2011)
14. Martins, M.A., Madeira, A., Barbosa, L.S.: Refinement by interpretation in a general setting. Electron. Notes Theor. Comput. Sci. 259, 105–121 (2009)
15. Martins, M.A., Madeira, A., Barbosa, L.S.: Refinement via interpretation. In: Hung, D.V., Krishnan, P. (eds.) SEFM, pp. 250–259. IEEE Computer Society (2009)
16. Martins, M.A., Madeira, A., Diaconescu, R., Barbosa, L.S.: Hybridization of institutions. In: Corradini, A., Klin, B., Cîrstea, C. (eds.) CALCO 2011. LNCS, vol. 6859, pp. 283–297. Springer, Heidelberg (2011)
17. Mossakowski, T.: Foundations of heterogeneous specification. In: Wirsing, M., Pattinson, D., Hennicker, R. (eds.) WADT 2003. LNCS, vol. 2755, pp. 359–375. Springer, Heidelberg (2003)
18. Mossakowski, T., Haxthausen, A., Sannella, D., Tarlecki, A.: CASL: The common algebraic specification language: Semantics and proof theory. Computing and Informatics 22, 285–321 (2003)

19. Mossakowski, T., Maeder, C., Codescu, M., Lucke, D.: Hets user guide - version 0.97. Technical report, DFKI Lab Bremen (March 2011), http://www.informatik.uni-bremen.de/agbkb/forschung/formal_methods/CoFI/hets/index_e.htm
20. Mossakowski, T., Maeder, C., Lüttich, K.: The heterogeneous tool set, HETS. In: Grumberg, O., Huth, M. (eds.) TACAS 2007. LNCS, vol. 4424, pp. 519–522. Springer, Heidelberg (2007)
21. Platzer, A.: Towards a hybrid dynamic logic for hybrid dynamic systems. Electron. Notes Theor. Comput. Sci. 174, 63–77 (2007)
22. Rodrigues, C.J., Martins, M.A., Madeira, A., Barbosa, L.S.: Refinement by interpretation in π-institutions. EPTCS 55, 53–64 (2011)
23. Sannella, D.: Algebraic specification and program development by stepwise refinement (Extended abstract). In: Bossi, A. (ed.) LOPSTR 1999. LNCS, vol. 1817, pp. 1–9. Springer, Heidelberg (2000)
24. Tarlecki, A.: Abstract specification theory: An overview. In: Broy, M., Pizka, M. (eds.) Models, Algebras, and Logics of Engineering Software. NATO Science Series, Computer and Systems Sciences, vol. 191, pp. 43–79. IOS Press, Amsterdam (2003)
25. van Eijck, J.: Hylotab-tableau-based theorem proving for hybrid logics. Technical report, CWI (2002), http://homepages.cwi.nl/~jve/#Publications