Science of Computer Programming 126 (2016) 73–93

Contents lists available at ScienceDirect

Science of Computer Programming

www.elsevier.com/locate/scico

Proof theory for hybrid(ised) logics

Renato Neves [a,*], Alexandre Madeira [a], Manuel A. Martins [b], Luis S. Barbosa [a]

[a] HASLab (INESC TEC) & Universidade do Minho, Portugal
[b] CIDMA, Dep. Mathematics, Universidade de Aveiro, Portugal

ARTICLE INFO

Article history:
Received 24 April 2015
Received in revised form 3 March 2016
Accepted 3 March 2016
Available online 14 March 2016

Keywords:
Hybrid logic
Decidability
Completeness
Tableau systems
Hilbert calculus

ABSTRACT

Hybridisation is a systematic process along which the characteristic features of hybrid logic, both at the syntactic and the semantic levels, are developed on top of an arbitrary logic framed as an institution. In a series of papers this process has been detailed and taken as a basis for a specification methodology for reconfigurable systems. The present paper extends this work by showing how a proof calculus (in both a Hilbert and a tableau based format) for the hybridised version of a logic can be systematically generated from a proof calculus for the latter. Such developments provide the basis for a complete proof theory for hybrid(ised) logics, and thus pave the way to the development of (dedicated) proof support.

1. Introduction

1.1. Motivation and context

This paper is part of a broader research agenda on the use of hybrid logic [1,2] as a formal basis for the specification of reconfigurable systems [3]. Those are characterised by the ability to adapt (or reconfigure) their behaviour in response to contextual changes, switching from one mode of operation to another. Such systems are ubiquitous in the Information Society, from service-oriented applications that change services in accordance to the network traffic level, to controllers embedded in modern cars whose driving contexts switch from economy to added power whenever the 'sports mode' is selected.

The formal specification of a reconfigurable system is often a challenge: whatever logic the software engineer finds useful to define the system's behavioural requirements, it may not be suitable to relate the different contexts in which they hold and express the reconfiguration dynamics. The approach proposed in [4] makes explicit the *labelled transition structure* that typically underlies a reconfigurable system. Each of its states corresponds to a specific configuration, or mode of operation, specified in an appropriate logic. Arrows, on the other hand, relate two possible configurations and exhibit in their labels the event that triggers the change. Thus, while this transition structure can be specified in (some variant of) modal logic, the description of the possible behaviours of concrete configurations requires logics that better suit the nature of the software system at hands. For example, continuous systems advocate the use of topological logics in the specification of each local configuration, whereas probabilistic systems are better handled through logics that embed some fragment of probability theory. Thus, our previous work [4] proposes that the specification of *reconfigurable* systems should be divided into two different levels:

* Corresponding author.
E-mail addresses: rjneves@inescporto.pt (R. Neves), amadeira@inescporto.pt (A. Madeira), martins@ua.pt (M.A. Martins), lsb@di.uminho.pt (L.S. Barbosa).

http://dx.doi.org/10.1016/j.scico.2016.03.001
0167-6423/

- *globally* the reconfiguration dynamics is represented by a transition structure described in hybrid logic, a logic that adds to modal reasoning the ability to pinpoint individual states, which in this context, represent configurations;
- *locally* each state is endowed with a structure that models, in a suitable logic, the specification of the associated configuration.

Therefore, to address both dimensions together in a single logical setting, the features of hybrid logic are developed on top of whatever logic is used for the local specification of each configuration. The logic used locally becomes *hybridised*, a specific procedure for combination of logics developed in A. Madeira's doctoral thesis [5], referred to as the *hybridisation* process.

The logic used locally depends, as expected, on the application requirements. Typical candidates are equational, partial algebra or first-order logic (*FOL*), but one may equally resort to multivalued logics or even to hybrid logic itself equipping, in this last case, each state with another (local) transition system. Verification resorts to a parametrised translation to *FOL* (developed in [6,7] and further extended in [8]), but at the cost of losing decidability and adding extra complexity.

The work reported here, extending as discussed below previous results introduced in the original conference paper [9], paves the way to an alternative approach in which verification can be carried on at the level of the hybridised logic itself. Even if a number of further questions has to be addressed to make it a pragmatic alternative, namely in what concerns complexity and possible circumventing heuristics, the paper introduces a first contribution. In brief, the hybridisation method is extended so that not only the logic is hybridised but also its calculus is systematically enriched into a calculus for the hybridised logic. Moreover, the latter is shown to be sound and complete whenever the calculus associated to the underlying, base logic is.

This programme, sketched in reference [9] as an Hilbert style calculus, is detailed here and extended to the generation of a tableau system for the hybridised logic. Actually, Hilbert calculi, although simple and versatile, are not amenable to effective computational support. On the other hand, tableau systems [2,10], able to systematically decompose sentences until contradictions are found, are well-known for their impressive computational power; in particular for the class of modal logics, where hybrid(ised) logics live. We believe this development is a first step towards dedicated proof support for a broad spectrum of hybrid(ised) logics.

Hybrid logic, with its ability to explicitly refer to local states in a transition structure, proved to be a powerful tool to specify reconfigurations [4,11]. Other, less standard extensions of modal logic, namely swap logic [12,13] in which reconfigurations steps can be reverted or erased at evaluation time, may complement this view with other, interesting possibilities.

1.2. Contributions and roadmap

The paper's starting point is to recast the hybridisation method in the context of the theory of institutions with proofs [14], which makes possible the development of the whole framework at a general level. Then, it simplifies the generation of the Hilbert calculus originally proposed in [9], and, as a main contribution, introduces the corresponding tableau version. Besides the theoretical relevance of these results, from a pragmatic point of view they pave the way to the development of effective tool support for the verification of reconfigurable systems within the approach proposed in [4].

A clarification is in order at this point. As the attentive reader may notice, most of the results presented here could be formulated out of the institutional setting. We believe, however, the latter provides an abstract framework in which the hybridisation process can be discussed in full generality. Actually, the level of generality that the notion of institution achieves, is one of the reasons for its success. Such was also the path initiated in [4] and kept in this paper, which can be considered as one of its follow-ups.

The theory of institutions (see [14] for an extensive account) was motivated by the need to abstract from the particular details of each individual logic and characterise generic issues, such as satisfaction and combination of logics, in very general terms. In computer science, this lead to the development of a solid *institution-independent specification theory*, on which, structuring and parameterisation mechanisms, required to scale up software specification methods, are defined 'once and for all', irrespective of the concrete logic used in each application domain. This explains why institutions proved effective and resilient as witnessed by the wide number of logics formalised in this way. Examples range from the usual logics in classical mathematical logic (propositional, equational, first order, etc.), to the ones underlying specification and programming languages or used for describing particular systems from different domains. Well-known examples include *probabilistic logics* [15], *quantum logics* [16], *hidden and observational logics* [17–19], *coalgebraic logics* [20], as well as logics for reasoning about *process algebras* [21], *functional* [22,23] and *imperative programing languages* [22].

The remainder of the paper is organised as follows: Section 2 provides the relevant background on institutions with proofs and revisits the hybridisation method in this setting. Section 3 presents the generation of Hilbert calculi, and discusses decidability and completeness of hybrid(ised) logics. Section 4 introduces the corresponding tableau version. Finally, Section 5 concludes and provides pointers for future work.

2. Background

2.1. Institutions with proofs

The generic character of the hybridisation process is due to its rendering in the context of the theory of institutions [24]. The notion of an institution formalises the essence of a logical system by encompassing syntax, semantics and satisfaction. Formally,

Definition 1. An institution is a tuple $(Sign^I, Sen^I, Mod^I, (\models^I_\Sigma)_{\Sigma \in |Sign^I|})$, where

- $Sign^I$ is a category whose objects are signatures and arrows signature morphisms,
- $Sen^I : Sign^I \to \mathbb{S}et$, is a functor that, for each signature $\Sigma \in |Sign^I|$, returns a set of sentences over Σ,
- $Mod^I : (Sign^I)^{op} \to \mathbb{C}at$, is a functor that, for each signature $\Sigma \in |Sign^I|$, returns a category whose objects are models over Σ,
- $\models^I_\Sigma \subseteq |Mod^I(\Sigma)| \times Sen^I(\Sigma)$, or simply $\models$, if the context is clear, is a satisfaction relation such that, for each signature morphism $\varphi : \Sigma \to \Sigma'$,

$$Mod^I(\varphi)(M') \models^I_\Sigma \rho \text{ iff } M' \models^I_{\Sigma'} Sen^I(\varphi)(\rho), \text{ for any}$$

$M' \in |Mod^I(\Sigma')|$ and $\rho \in Sen^I(\Sigma)$. Graphically,

$$\begin{array}{ccc} \Sigma & & Mod^I(\Sigma) \xrightarrow{\models^I_\Sigma} Sen^I(\Sigma) \\ {\scriptstyle\varphi}\downarrow & & \uparrow{\scriptstyle Mod^I(\varphi)} \qquad \downarrow{\scriptstyle Sen^I(\varphi)} \\ \Sigma' & & Mod^I(\Sigma') \xrightarrow[\models^I_{\Sigma'}]{} Sen^I(\Sigma') \end{array}$$

Intuitively, this property claims that satisfaction is preserved under change of notation. In order to build up the reader's intuition, let us recall some typical examples.

Example 1. Many sorted first-order logic (FOL).

- Signatures. $Sign^{\text{FOL}}$ is a category whose objects are triples (S, F, P), consisting of a set of sort symbols S, a family, $F = (F_{w \to s})_{w \in S^*, s \in S}$, of function symbols indexed by their arity, and a family, $P = (P_w)_{w \in S^*}$, of relational symbols also indexed by their arity.
  A signature morphism is a triple $(\varphi_{st}, \varphi_{op}, \varphi_{rl}) : (S, F, P) \to (S', F', P')$ such that if $\sigma \in F_{w \to s}$, then $\varphi_{op}(\sigma) \in F'_{\varphi_{st}(w) \to \varphi_{st}(s)}$, and if $\pi \in P_w$ then $\varphi_{rl}(\pi) \in P'_{\varphi_{st}(w)}$.
- Sentences. For each signature $(S, F, P) \in |Sign^{\text{FOL}}|$, $Sen^{\text{FOL}}(S, F, P)$ is the smallest set generated by the grammar below

$$\rho \ni \neg\rho \mid \rho \wedge \rho \mid t = t \mid \pi(X) \mid \forall x : s \,.\, \rho'$$

  where t is a term with the syntactic structure $\sigma(X)$ for $\sigma \in F_{w \to s}$ and X a list of terms compatible with the arity of σ. $\pi \in P_w$ and X is a list of terms compatible with the arity of π. Finally, $\rho' \in Sen^{FOL}(S, F \uplus \{x\}_{\to s}, P)$. $Sen^I(\varphi)$, for φ a signature morphism, is a function that, given a sentence $\rho \in Sen^I(S, F, P)$, replaces the signature symbols in ρ according to φ.
- Models. For each signature $(S, F, P) \in |Sign^{\text{FOL}}|$, $Mod^{\text{FOL}}(S, F, P)$ is the category with only identity arrows and whose objects are models with a carrier set $|M_s|$, for each $s \in S$, a function $M_\sigma : |M_w| \to |M_s|$, for each $\sigma_{w \to s} \in F_{w \to s}$, and a relation $M_\pi \subseteq |M_w|$, for each $\pi \in P_w$.
- Satisfaction. Satisfaction of sentences by models is the usual Tarskian satisfaction.

▲

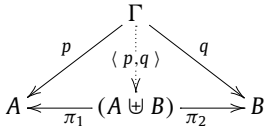
Example 2. Equational (EQ) and propositional (PL) logics.
The institution EQ is the sub-institution of FOL in which sentences are restricted to those of the type $\forall \overline{x} : \overline{s} \,.\, t = t'$. Institution PL is the sub-institution of FOL in which signatures with no empty set of sorts are discarded.

▲

Other examples of institutions include the algebraic specification language CASL [25], many-valued logics [26,27], and the relational-based language Alloy [28].

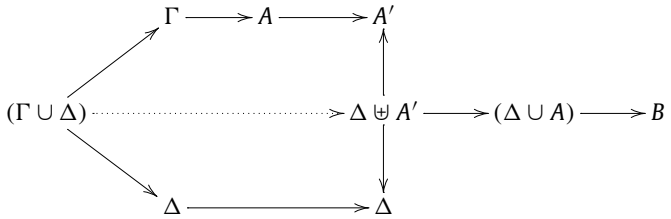
However, the classic notion of an institution, does not includes an abstract structure to represent the associated logic calculus. The problem was addressed in [29] with the introduction of π-*institutions*, and, more recently, with the notion of an *institution with proofs* [14].

Definition 2. An institution with proofs adds to the original definition of an institution, a functor $Prf^I : Sign^I \to \mathbb{C}at$ such that, for each $\Sigma \in |Sign^I|$, $Prf^I(\Sigma)$ (called the category of Σ-proofs) has subsets of $Sen^I(\Sigma)$ (*i.e.* $|Prf^I(\Sigma)| = \mathcal{P}(Sen^I(\Sigma))$) as objects, and the corresponding proofs as arrows. The latter are preserved along signature morphisms. In addition, for $A, B \in |Prf^I(\Sigma)|$, if $A \subseteq B$ then arrow $B \longrightarrow A$ exists; if $A \cap B = \emptyset$ and $\Gamma \in |Prf^I(\Sigma)|$ has arrows $p : \Gamma \longrightarrow A$ and $q : \Gamma \longrightarrow B$, then there is a unique proof arrow $\langle p, q \rangle$ that makes the following diagram to commute:



For the sake of simplicity, when a singleton set of sentences is presented in a proof arrow, we will drop the curly brackets. Also, observe that the restrictions imposed to the proof arrows force upon Prf^I the following properties, which are typical of most proof systems:

1. *Reflexivity* (if $A \in \Gamma$, then $\Gamma \vdash A$) follows from the fact that $\{A\} \subseteq \Gamma$ and, therefore, $\Gamma \longrightarrow A$.
2. *Monotonicity* (if $\Gamma \vdash A$ and $\Gamma \subseteq \Delta$ then $\Delta \vdash A$), follows from composition of proofs, where $\Delta \longrightarrow \Gamma$ is given by inclusion and $\Gamma \longrightarrow A$ by the assumption.
3. *Transitivity* (if $\Gamma \vdash A$ and $\{\Delta, A\} \vdash B$ then $\Gamma \cup \Delta \vdash B$), follows from the product of disjoint sets, reflexivity and monotonicity,



where $A' = A - (A \cap \Delta)$.

Functor Prf^I distinguishes different proof arrows between the same pair of objects. In this work, however, we force the category $Prf^I(\Sigma)$ to be thin (*i.e.* each pair of objects to have at most one arrow). Such a restriction allows a clear focus on entailment systems,[1] and trivialises the uniqueness property of arrow $\langle p, q \rangle$.

In the sequel we use notation $A \vdash^I B$ to say that arrow $A \to B$ is in $Prf^I(\Sigma)$, and expression $\vdash^I B$ as an abbreviation of $\emptyset \vdash^I B$. Conversely, we use $A \nvdash^I B$ to negate $A \vdash^I B$. On the semantic side, we say that a sentence $\rho \in Sen^I(\Sigma)$ is Σ-*valid* (or simply, valid) if for each model $M \in |Mod^I(\Sigma)|$, $M \models^I_\Sigma \rho$. Usually we prefix such sentences by $\models^I_\Sigma$ or, simply by $\models^I$ or just $\models$.

Definition 3. Let I be an institution with a proof system Prf^I. We say that Prf^I is *sound* and *complete* if, for any signature $\Sigma \in |Sign^I|$ and sentence $\rho \in Sen^I(\Sigma)$,

$$\vdash^I \rho \text{ iff } \models^I \rho$$

Specifically, sound if $\vdash^I \rho$ entails $\models^I \rho$ and complete if $\models^I \rho$ entails $\vdash^I \rho$.

A property equivalent to soundness and completeness arises from the following definitions.

Definition 4. (See [30].) An institution I is called *Boolean complete* if it has all semantic Boolean connectives. More formally, if given a signature $\Sigma \in |Sign^I|$,

- for any sentence $\rho \in Sen^I(\Sigma)$, there is a sentence $\neg\rho \in Sen^I(\Sigma)$ such that for any model $M \in |Mod^I(\Sigma)|$, $M \models \rho$ iff $M \not\models \neg\rho$,

[1] Typically, in an entailment system $\Gamma \vdash A$ means that Γ derives (or entails) A.

- for any sentences $\rho, \rho' \in Sen^I(\Sigma)$, there is a sentence $\rho \wedge \rho' \in Sen^I(\Sigma)$ such that for any model $M \in |Mod^I(\Sigma)|$, $M \models \rho \wedge \rho'$ iff $M \models \rho$ and $M \models \rho'$.

Note that the Boolean connectives are unique up to semantic equivalence. Then, negation makes possible to state that, given an institution I and signature $\Sigma \in |Sign^I|$, for any sentence $\rho \in Sen^I(\Sigma)$,

ρ is *unsatisfiable* iff $\neg\rho$ is valid.

As usual, $\rho \vee \rho'$ denotes $\neg(\neg\rho \wedge \neg\rho')$ and $\rho \rightarrow \rho'$ denotes $\neg(\rho \wedge \neg\rho')$. Sentence $\rho \wedge \neg\rho$, denoted by $\bot$, is such that no model in $|Mod^I(\Sigma)|$ satisfies it. Symbol $\top$ represents the negation of $\bot$. Finally,

Theorem 1. *Consider a Boolean complete institution with proofs I, such that Prf^I contains the* double negation introduction rule *and, its inverse, the* double negation elimination. *Then the following statements are equivalent.*

1. *Prf^I is sound and complete,* i.e. *for any $\rho \in Sen^I(\Sigma)$, $\vdash^I \rho$ iff $\models^I \rho$.*
2. *For any sentence $\rho \in Sen^I(\Sigma)$, ρ is satisfiable iff $\nvdash^I \neg\rho$.*

Proof. Follows from:

- $1. \Rightarrow 2.$

$$
\begin{array}{ll}
 & \rho \text{ is satisfiable} \\
\equiv & \{ \text{ Definition of satisfiability } \} \\
 & \not\models^I \neg\rho \\
\equiv & \{ \text{ 1. } \} \\
 & \nvdash^I \neg\rho
\end{array}
$$

- $2. \Rightarrow 1.$

$$
\begin{array}{ll}
 & \vdash^I \rho \\
\equiv & \{ \text{ Double negation rules } \} \\
 & \vdash^I \neg(\neg\rho) \\
\equiv & \{ \text{ 2. } \} \\
 & \neg\rho \text{ is unsatisfiable} \\
\equiv & \{ \text{ Definition of satisfiability } \} \\
 & \models^I \rho \quad \square
\end{array}
$$

2.2. Hybridisation revisited

This subsection reviews the basics of the hybridisation process with the global modality. Document [6] reports a version of hybridisation where universal quantification over worlds and polyadic modalities are also considered.

Let $Sign^{\mathcal{H}}$ be the category $\mathbb{S}et \times \mathbb{S}et$ whose objects are pairs (Nom, Λ), where Nom denotes a set of nominal symbols and Λ a set of modality symbols.

Definition 5. Given an institution $I = (Sign^I, Sen^I, Mod^I, \models^I)$ its hybridised version $\mathcal{H}I = (Sign^{\mathcal{H}I}, Sen^{\mathcal{H}I}, Mod^{\mathcal{H}I}, \models^{\mathcal{H}I})$ is defined as follows:

- $Sign^{\mathcal{H}I} = Sign^{\mathcal{H}} \times Sign^I$,
- given a signature $(\Delta, \Sigma) \in |Sign^{\mathcal{H}I}|$, $Sen^{\mathcal{H}I}(\Delta, \Sigma)$ is the least set generated by

$$\rho \ni \barwedge\rho \mid \rho \barwedge \rho \mid i \mid @_i\rho \mid \langle\lambda\rangle\rho \mid A\ \rho \mid \psi$$

for i a nominal, λ a modality, $\psi \in Sen^I(\Sigma)$. We use non standard Boolean connectives symbols $(\barwedge, \barwedge)$ in order to distinguish them from the Boolean connectives that a base logic may have. In general, note that the set of symbols introduced by the hybridisation method is disjoint from the set of symbols in the base institution I. Also, define $[\lambda]\rho \equiv \barwedge\langle\lambda\rangle\barwedge\rho$, $E\ \rho \equiv \barwedge A \barwedge\rho$ and $\rho \Rightarrow \rho' \equiv \barwedge(\rho \barwedge \barwedge\rho')$. Letter ψ stands for a sentence of the base logic.

- Given a signature $(\Delta, \Sigma) \in |Sign^{\mathcal{H}I}|$, a model $M \in |Mod^{\mathcal{H}I}(\Delta, \Sigma)|$ is a triple (W, R, m) such that,
  - W is a nonempty set of worlds,
  - R is a family of relations indexed by the modality symbols Λ, *i.e.* for each $\lambda \in \Lambda$, $R_\lambda \subseteq W \times W$,
  - $m : W \to |Mod^I(\Sigma)|$.

  Also, for each $i \in Nom$, $M_i \in W$.
- Given a signature $(\Delta, \Sigma) \in |Sign^{\mathcal{H}I}|$, a model $M = (W, R, m) \in |Mod^{\mathcal{H}I}(\Delta, \Sigma)|$ and a sentence $\rho \in Sen^{\mathcal{H}I}(\Delta, \Sigma)$, the satisfaction relation is defined as,

$$M \models^{\mathcal{H}I}_{(\Delta,\Sigma)} \rho \text{ iff } M \models^w \rho, \text{ for all } w \in W$$

where,

$$\begin{array}{lll}
M \models^w \neg\rho & \text{iff} & M \not\models^w \rho \\
M \models^w \rho \barwedge \rho' & \text{iff} & M \models^w \rho \text{ and } M \models^w \rho' \\
M \models^w i & \text{iff} & M_i = w \\
M \models^w @_i \rho & \text{iff} & M \models^{M_i} \rho \\
M \models^w \psi & \text{iff} & m(w) \models^I_\Sigma \psi \\
M \models^w A\ \rho & \text{iff} & \text{for all } v \in W,\ M \models^v \rho \\
M \models^w \langle\lambda\rangle\rho & \text{iff} & \text{there is some } v \in W \text{ such that } (w, v) \in R_\lambda \text{ and } M \models^v \rho.
\end{array}$$

Actually, if the base institution I is Boolean complete, due to the equivalences $\psi \wedge \psi' \equiv \psi \barwedge \psi'$, $\neg\psi \equiv \barneg\psi$, it is possible to collapse the Boolean connectives $\wedge$, $\barwedge$, and also $\neg$, $\barneg$ (*cf.* [8]). Thus, the grammar of the hybridised logic becomes,

$$\rho \ni \neg\rho \mid \rho \wedge \rho \mid i \mid @_i\rho \mid \langle\lambda\rangle\rho \mid A\ \rho \mid \psi.$$

In the sequel, since it turns proofs simpler and more intuitive, we assume that all hybridised logics adopt this approach.

Example 3. Hybridised propositional logic ($\mathcal{H}PL$)

- Signatures are pairs $(\Delta, \Sigma) \in |Sign^{\mathcal{H}PL}|$ where Σ is a set of propositional symbols. It is assumed that this set and the set of nominals are disjoint.
- Sentences are generated by the grammar

$$\rho \ni i \mid p \mid \neg\rho \mid \rho \wedge \rho \mid @_i\rho \mid \langle\lambda\rangle\rho \mid A\ \rho$$

  where i is a nominal and p a propositional symbol.
- Models are Kripke structures (W, R) (where for each $\lambda \in \Lambda$, $R_\lambda \subseteq W \times W$) equipped with a function $m : W \to |Mod^I(\Sigma)|$ that makes each world to correspond to a propositional model (*i.e.* a subset of Σ).

▲

When the only signatures considered are those that possess exactly one modality symbol, $\mathcal{H}PL$ coincides with classical hybrid propositional logic with global modality (which is known to be decidable and have a complete calculus). In this case symbols $[\lambda]$, $\langle\lambda\rangle$ are replaced, respectively, by $\Box$ and $\Diamond$.

3. Generation of an Hilbert calculus for the hybridised logic

3.1. The method

This section introduces a refined version of the method for generation of an Hilbert calculus for the hybridised logic (originally proposed in [9]) in which the collapse of Boolean connectives is taken into consideration. This new formulation simplifies the whole process and contributes to smaller proofs. Thus, consider an institution I with a proof system Prf^I. For any signature $(\Delta, \Sigma) \in |Sign^{\mathcal{H}I}|$, the category $Prf^{\mathcal{H}I}(\Delta, \Sigma)$ is generated by the axioms and rules stated in Fig. 1. Note that their schematic form guarantees that the proof arrows are preserved along signature morphisms.

Let us see some examples of Hilbert calculi, generated through this process, at work.

Example 4. To show that $[\lambda](\forall \overline{x} : \overline{s}\,.\, t = t)$ is a theorem in $\mathcal{H}EQ$ one starts with

$$\vdash^{EQ} \ \forall \overline{x} : \overline{s}\,.\, t = t$$

Axioms	
All instances of classical tautologies for $\neg, \rightarrow$	(CT)
$@_i(\rho \rightarrow \rho') \leftrightarrow (@_i\rho \rightarrow @_i\rho')$	$(Dist)$
$@_i\bot \rightarrow \bot$	$(\bot)$
$@_i@_j\rho \rightarrow @_j\rho$	$(Scope)$
$@_i i$	(Ref)
$(i \wedge \rho) \rightarrow @_i\rho$	$(Intro)$
$([\lambda]\,\rho \wedge \langle\lambda\rangle i) \rightarrow @_i\rho$	$([\lambda]_E)$
$A\,\rho \rightarrow @_i\,\rho$	(A_E)
ψ, for all $\vdash^I \psi$	$(\uparrow)$

Rules	
$\vdash^{\mathcal{H}I} \rho, \vdash^{\mathcal{H}I} \rho \rightarrow \rho'$ entails $\vdash^{\mathcal{H}I} \rho'$	(MP)
if $\vdash^{\mathcal{H}I} \rho$ then $\vdash^{\mathcal{H}I} @_i\rho$	$(@_I)$
if $\vdash^{\mathcal{H}I} @_i\rho$ then $\vdash^{\mathcal{H}I} \rho$	$(@_E)^\star$
if $\vdash^{\mathcal{H}I} (\rho \wedge \langle\lambda\rangle i) \rightarrow @_i\rho'$ then $\vdash^{\mathcal{H}I} \rho \rightarrow [\lambda]\rho'$	$([\lambda]_I)^\star$
if $\vdash^{\mathcal{H}I} \rho \rightarrow @_i\rho'$ then $\vdash^{\mathcal{H}I} \rho \rightarrow A\,\rho'$	$(A_I)^\star$

where annotation * denotes condition '*if i does not occur free neither in ρ nor ρ'*'.

Fig. 1. Axioms and rules for $Prf^{\mathcal{H}I}$ (based on the Hilbert calculus introduced in [2]).

and proceeds

$$\vdash^{\mathcal{H}I} \forall\,\overline{x}:\overline{s}\,.\,t = t \qquad (\uparrow)$$
$$\vdash^{\mathcal{H}I} @_i\,(\forall\,\overline{x}:\overline{s}\,.\,t = t) \qquad (@_I)$$
$$\vdash^{\mathcal{H}I} (\top \wedge \langle\lambda\rangle i) \rightarrow @_i\,(\forall\,\overline{x}:\overline{s}\,.\,t = t) \qquad (CT)$$
$$\vdash^{\mathcal{H}I} \top \rightarrow [\lambda](\forall\,\overline{x}:\overline{s}\,.\,t = t) \qquad ([\lambda]_I)$$
$$\vdash^{\mathcal{H}I} [\lambda](\forall\,\overline{x}:\overline{s}\,.\,t = t) \qquad (CT)$$

▲

Example 5. Sentence $\Box(p \rightarrow q) \rightarrow (\Box p \rightarrow \Box q)$ is an instance of theorem K of classic hybrid propositional logic; let us prove it through the generated Hilbert calculus of $\mathcal{H}PL$. First one notes that,

$$\vdash^{\mathcal{H}I} (\Box p \wedge \Diamond i) \rightarrow @_i p \qquad (\Box_E)$$
$$\vdash^{\mathcal{H}I} (\Box(p \rightarrow q) \wedge \Box p \wedge \Diamond i) \rightarrow (\Box(p \rightarrow q) \wedge @_i p \wedge \Diamond i) \qquad (CT)$$

Then,

$$\vdash^{\mathcal{H}I} (\Box(p \rightarrow q) \wedge \Diamond i) \rightarrow @_i(p \rightarrow q) \qquad (\Box_E)$$
$$\vdash^{\mathcal{H}I} (\Box(p \rightarrow q) \wedge \Diamond i) \rightarrow (@_i p \rightarrow @_i q) \qquad (Dist)$$
$$\vdash^{\mathcal{H}I} (\Box(p \rightarrow q) \wedge \Diamond i \wedge @_i p) \rightarrow @_i q \qquad (CT)$$

Both cases lead to theorem,

$$\vdash^{\mathcal{H}I} (\Box(p \rightarrow q) \wedge \Box p \wedge \Diamond i) \rightarrow @_i q \qquad (MP, CT)$$
$$\vdash^{\mathcal{H}I} (\Box(p \rightarrow q) \wedge \Box p) \rightarrow \Box q \qquad (\Box_I)$$
$$\vdash^{\mathcal{H}I} \Box(p \rightarrow q) \rightarrow (\Box p \rightarrow \Box q) \qquad (CT)$$

▲

Note that it is straightforward to generalise the property above to any hybridised logic.

3.2. Soundness and completeness

We shall now show that, under certain conditions, any generated Hilbert calculus is sound and complete whenever such is the case for the corresponding base calculus. For this assume, in the sequel, that the logic to be hybridised is Boolean complete.

Theorem 2 *(Soundness). Consider an institution I with a sound proof system Prf^I. Then, for any signature $(\Delta, \Sigma) \in |Sign^{\mathcal{H}I}|$ and sentence $\rho \in Sen^{\mathcal{H}I}(\Delta, \Sigma)$,*

$$\vdash^{\mathcal{H}I} \rho \text{ entails } \models^{\mathcal{H}I} \rho$$

Proof. The result follows from the analysis of each rule and axiom in $Prf^{\mathcal{H}I}$. In particular, for axiom $(\uparrow)$ we have

$$\begin{array}{ll} & \vdash^I \psi \\ \Rightarrow & \quad \{ \vdash^I \text{ is sound } \} \\ & \models^I \psi \\ \Rightarrow & \quad \{ \text{ Definition of } \models^{\mathcal{H}I} \} \\ & \models^{\mathcal{H}I} \psi \end{array}$$

The proof of the remaining cases is straightforward. □

On the other hand, the proof of completeness requires some preliminaries.

Definition 6. Consider a Boolean complete institution I. For any signature $(\Delta, \Sigma) \in |Sign^{\mathcal{H}I}|$, a given sentence $\rho \in Sen^{\mathcal{H}I}(\Delta, \Sigma)$ is *basic* iff $sb(\rho) = \{\rho\}$ where $sb(\varphi) = \bigcup_{k>0} sb_k(\varphi)$ for

$$\begin{array}{l} sb_0(\varphi) = \varphi \\ sb_{k+1}(\varphi) = \{\varphi' : \heartsuit\varphi' \in sb_k(\varphi) \text{ for } \heartsuit \in \{\neg, @_i, \langle\lambda\rangle, A\}\} \\ \cup \{\varphi_1, \varphi_2 : \varphi_1 \wedge \varphi_2 \in sb_k(\varphi)\} \text{ for any } k > 0 \end{array}$$

Definition 7. Consider a signature $(\Delta, \Sigma) \in |Sign^{\mathcal{H}I}|$, $\rho \in Sen^{\mathcal{H}I}(\Delta, \Sigma)$ and let $B_\rho = \{\psi_1, \ldots, \psi_n\} \subseteq Sen^I(\Sigma)$ be the set of maximal base sentences in ρ that are basic. Then, Ω_ρ denotes the set of sentences such that for each $a \in 2^{B_\rho}$

$$(\chi_1 \wedge \cdots \wedge \chi_n) \in \Omega_\rho \subseteq Sen^I(\Sigma)$$

where

$$\chi_i = \begin{cases} \psi_i & \text{if } \psi_i \in a \\ \neg\psi_i & \text{otherwise} \end{cases}$$

Lemma 1. *Assume that $\Omega_\rho \not\equiv \emptyset$. Then, for any model $M \in |Mod^I(\Sigma)|$, M satisfies exactly one of the sentences in Ω_ρ.*

Proof. First observe that for any different $\chi, \chi' \in \Omega_\rho$ at least one clause in χ appears negated in χ'. This entails that M can never satisfy χ and χ' at the same time (conjunction and negation properties). Now, if $M \not\models \chi$, then there is a sentence $\chi' \in \Omega_\rho$ that negates all clauses leading to $M \not\models \chi$, and, therefore, $M \models \chi'$. □

Definition 8. Consider function $\sigma : Sen^{\mathcal{H}I}(\Delta, \Sigma) \to Sen^{\mathcal{H}PL}(\Delta, P)$ where $P = \{\pi_\psi \mid \psi \in Sen^I(\Sigma)\}$, such that

$$\begin{array}{lcl} \sigma(\neg\rho) & = & \neg\sigma(\rho) \\ \sigma(\rho \wedge \rho') & = & \sigma(\rho) \wedge \sigma(\rho') \\ \sigma(i) & = & i \\ \sigma(@_i\rho) & = & @_i\sigma(\rho) \\ \sigma(\langle\lambda\rangle\rho) & = & \langle\lambda\rangle\sigma(\rho) \\ \sigma(A\ \rho) & = & A\ \sigma(\rho) \\ \sigma(\psi) & = & \pi_\psi, \text{ if } \psi \text{ is basic} \end{array}$$

Intuitively, this means that function σ replaces the basic sentences of the input $\rho \in Sen^{\mathcal{H}I}(\Delta, \Sigma)$ by propositional symbols.

Lemma 2. *For any signature* $(\Delta, \Sigma) \in |Sign^{\mathcal{H}I}|$, $\rho \in Sen^{\mathcal{H}I}(\Delta, \Sigma)$

$$\not\vdash^{\mathcal{H}I} \rho \text{ entails } \not\vdash^{\mathcal{H}PL} \sigma(\rho)$$

or equivalently,

$$\vdash^{\mathcal{H}PL} \sigma(\rho) \text{ entails } \vdash^{\mathcal{H}I} \rho$$

Proof. Observe that rules and axioms in $Prf^{\mathcal{H}PL}$ also hold for $Prf^{\mathcal{H}I}$, and that $\sigma(\rho), \rho$ are structurally the same. This implies that if $\vdash^{\mathcal{H}PL} \sigma(\rho)$, then, whichever rules and axioms were used before, one may reproduce the process using the same rules and axioms, thus arriving at $\vdash^{\mathcal{H}I} \rho$. □

Definition 9. Let $\Omega^{\star}_{\rho} = \{\chi \in \Omega_{\rho} \mid \vdash^{I} \neg\chi\}$ and consider function $\eta : Sen^{\mathcal{H}I}(\Delta, \Sigma) \to Sen^{I}(\Sigma)$ such that

$$\eta(\rho) = \begin{cases} \bigwedge\{\neg\chi \mid \chi \in \Omega^{\star}_{\rho}\} & \text{if } \Omega^{\star}_{\rho} \neq \emptyset \\ \top & \text{otherwise} \end{cases}$$

Lemma 3. *The sentence* $A\ \eta(\rho)$ *is a theorem, or in symbols* $\vdash^{\mathcal{H}I} A\ \eta(\rho)$.

Proof. Since $\vdash^{I} \eta(\rho)$ one has that $\vdash^{\mathcal{H}I} \eta(\rho)$. Then, by rule (A_I), $\vdash^{\mathcal{H}I} A\ \eta(\rho)$. □

Lemma 4. *Consider a signature* $(\Delta, \Sigma) \in |Sign^{\mathcal{H}I}|$, *a sentence* $\rho \in Sen^{\mathcal{H}I}(\Delta, \Sigma)$ *and a model* $M \in |Mod^{\mathcal{H}PL}(\Delta, P)|$ *such that*

$$M \models^{w} A\ \sigma(\eta(\rho))$$

for some $w \in W$. *Given any* $\chi \in \Omega_{\rho}$, *if* $\sigma(\chi)$ *is satisfied at some world of* M, *then* χ *is satisfiable.*

Proof. If χ is unsatisfiable then, because Prf^{I} is complete, condition $\vdash^{I} \neg\chi$ holds, implying that $\neg\chi$ is a clause of $\eta(\rho)$ and $\sigma(\neg\chi)$ a clause of $\sigma(\eta(\rho))$. Therefore, since $M \models^{w} A\ \sigma(\eta(\rho))$, no world of M can point to a model that satisfies $\sigma(\chi)$. □

Definition 10. An institution I has the *explicit satisfaction property*, if for any signature $\Sigma \in |Sign^{I}|$ and sentence $\rho \in Sen^{I}(\Sigma)$, satisfiability of ρ entails the existence of a model $M \in |Mod^{I}(\Sigma)|$ such that $M \models^{I}_{\Sigma} \rho$.

This last property holds for the most common logics used in software specification, *e.g.*, propositional, fuzzy, equational, partial and first-order. In the following theorem assume that the base institution has the explicit satisfaction property.

Theorem 3 *(Completeness). Consider signature* $(\Delta, \Sigma) \in |Sign^{\mathcal{H}I}|$ *and sentence* $\rho \in Sen^{\mathcal{H}I}(\Delta, \Sigma)$

If $\not\vdash^{\mathcal{H}I} \neg\rho$ *then* ρ *is satisfiable*

Proof. Start with the observation

$$\begin{array}{ll} & \not\vdash^{\mathcal{H}I} \neg\rho \\ \Rightarrow & \{ \text{(MP) and Lemma 3} \} \\ & \not\vdash^{\mathcal{H}I} \neg(\rho \wedge A\ \eta(\rho)) \\ \Rightarrow & \{ \text{Lemma 2} \} \\ & \not\vdash^{\mathcal{H}PL} \sigma(\neg(\rho \wedge A\ \eta(\rho))) \\ \Rightarrow & \{ \text{Definition of } \sigma \} \\ & \not\vdash^{\mathcal{H}PL} \neg(\sigma(\rho \wedge A\eta(\rho))) \end{array}$$

Thus, by Theorem 1 and since $Prf^{\mathcal{H}PL}$ is complete, there is a model $M = (W, R, m) \in |Mod^{\mathcal{H}PL}(\Delta, P)|$ such that

$$M \models^{w} \sigma(\rho) \wedge A\ \sigma(\eta(\rho))$$

for some $w \in W$.

Next we build a model for ρ. Let $M' = (W, R, m')$ where for any $w \in W$ $m'(w)$ is a model for χ where $\sigma(\chi)$ is satisfied at $m(w)$—recall Lemmas 1 and 4 and the fact that I has the explicit satisfaction property. To finish the proof, it remains to show that $M' \models^{w} \rho$. This is proved by induction on the subformulas of ρ. For any sentence $\psi \in B_{\rho}$

$$
\begin{array}{ll}
& M, w \models \sigma(\psi) \\
\equiv & \quad \{ \text{ Definition of } \models \ \} \\
& m(w) \models \pi_\psi \\
\equiv & \quad \{ \ m'(w) \text{ satisfies some } \chi \text{ in which } \psi \text{ is present } \} \\
& m'(w) \models \psi \\
\equiv & \quad \{ \text{ Definition of } \models \} \\
& M', w \models \psi
\end{array}
$$

The remaining cases offer no difficulty. □

3.3. Decidability

Decidability is a key property on developing a new logic. Indeed, not only it is a central element in proof theory, but has also practical implications in the theory of software validation and verification.

The machinery used above to prove completeness, provides an interesting opportunity to discuss the decidability of hybrid(ised) logics. More concretely, progressing through slight changes in the definition of function η, one can show that if a logic is decidable then its hybridised version also is. This subsection reports on such a result. Recall our assumption that all base logics are Boolean complete. Then,

Lemma 5. *Consider signature* $(\Delta, \Sigma) \in |Sign^{\mathcal{H}I}|$ *and sentence* $\rho \in Sen^{\mathcal{H}I}(\Delta, \Sigma)$. *For any* $\chi \in \Omega_\rho$, $\sigma(\chi)$ *is satisfiable.*

Proof. Unsatisfaction of $\sigma(\chi)$ may only come from one of the following cases:

- a clause of $\sigma(\chi)$ is unsatisfiable;
- two clauses of $\sigma(\chi)$ contradict each other.

Clearly, a single clause of $\sigma(\chi)$ – a proposition – is always satisfiable. Then, note that, according to definition of χ, a clause in $\sigma(\chi)$ is π_{ψ_i} or $\neg\pi_{\psi_i}$ and any other π_{ψ_j} or $\neg\pi_{\psi_j}$. Since their corresponding propositional symbols differ, it is clear that they never clash. □

Theorem 4. *Consider a signature* $(\Delta, \Sigma) \in |Sign^{\mathcal{H}I}|$, *and sentence* $\rho \in Sen^{\mathcal{H}I}(\Delta, \Sigma)$. *If* ρ *is satisfiable* $\sigma(\rho)$ *also is.*

Proof. Start with the assumption that ρ is satisfiable which means that there is a model $(W, R, m) = M \in |Mod^{\mathcal{H}I}(\Delta, \Sigma)|$ such that $M \models^w \rho$ for some $w \in W$. From M define a model $M' = (W, R, m') \in |Mod^{\mathcal{H}PL}(\Delta, P)|$ such that for any $w \in W$, $\chi \in \Omega_\rho$, if $m(w) \models \chi$ then $m'(w) \models \sigma(\chi)$ (Lemmas 1 and 5). To finish the proof, it remains to show that $M' \models^w \sigma(\rho)$, which is done by induction on the subformulas of ρ. In particular, for any $v \in W$, $\psi \in B_\rho$,

$$
\begin{array}{ll}
& M \models^v \psi \\
\equiv & \quad \{ \text{ Definition of } \models^I \ \} \\
& m(v) \models \psi \\
\Rightarrow & \quad \{ \ m(v) \text{ satisfies some } \chi \in \Omega_\rho \text{ of which } \psi \text{ is a clause } \} \\
& m'(v) \models \sigma(\psi) \\
\equiv & \quad \{ \text{ Definition of } \models^{\mathcal{H}PL} \ \} \\
& M' \models^v \sigma(\psi)
\end{array}
$$

The remaining cases are straightforward. □

Next, we redefine function η.

Definition 11. Consider an institution I corresponding to a decidable logic, *i.e.*, with an effective decision procedure Sat^I. Then, let $\Omega^\star_\rho = \{\chi \in \Omega_\rho \mid Sat^I(\chi) \text{ is unsat } \}$ and

$$\eta(\rho) = \begin{cases} \bigwedge\{\neg\chi \mid \chi \in \Omega^{\star}_{\rho}\} & \text{if } \Omega^{\star}_{\rho} \neq \emptyset \\ \top & \text{otherwise} \end{cases}$$

Lemma 6. *Consider a signature* $(\Delta, \Sigma) \in |Sign^{\mathcal{HI}}|$*, a sentence* $\rho \in Sen^{\mathcal{HI}}(\Delta, \Sigma)$ *and a model* $M \in |Mod^{\mathcal{HPL}}(\Delta, P)|$ *such that*

$$M \models^{w} A\,\sigma(\eta(\rho))$$

for some $w \in W$*. Given any* $\chi \in \Omega_{\rho}$ *if* $\sigma(\chi)$ *is satisfied at some world of* M*, then* χ *is satisfiable.*

Proof. If χ is unsatisfiable, $\neg\chi$ is a clause of $\eta(\rho)$. Hence, since $M \models^{w} A\,\sigma(\eta(\rho))$, no world of M satisfies $\sigma(\chi)$. □

Theorem 5. *Assume that* I *has the explicit satisfaction property. Then, consider a signature* $(\Delta, \Sigma) \in |Sign^{\mathcal{HI}}|$ *and a sentence* $\rho \in Sen^{\mathcal{HI}}(\Delta, \Sigma)$*. If* $\sigma(\rho \wedge A\,\eta(\rho))$ *is satisfiable then so is* ρ*.*

Proof. Start with the assumption that $\sigma(\rho \wedge A\,\eta(\rho))$ is satisfiable which means that there is a model $M = (W, R, m) \in |Mod^{\mathcal{HPL}}(\Delta, P)|$ such that

$$M \models^{w} \sigma(\rho) \wedge A\,\sigma(\eta(\rho))$$

for some $w \in W$. From model M we define a model $M' = (W, R, m') \in |Mod^{\mathcal{HI}}(\Delta, \Sigma)|$ such that for any $w \in W$, $m'(w)$ is a model for $\chi \in \Omega_{\rho}$ where $m(w) \models \sigma(\chi)$ (recall Lemmas 1 and 6 and the fact that the explicit satisfaction property holds for I). To finish the proof, it remains to show that $(W, R, m') \models^{w} \rho$, which is done by induction on the structure of ρ. For any sentence $\psi \in B_{\rho}$, any $v \in W$,

$$\begin{aligned}
& M \models^{v} \sigma(\psi) \\
\equiv \quad & \{ \text{ Definition of } \models^{I} \ \} \\
& m(v) \models \sigma(\psi) \\
\Rightarrow \quad & \{ \ m'(v) \text{ satisfies some } \chi \text{ of which } \psi \text{ is a clause, definition of } m' \ \} \\
& m'(v) \models \psi \\
\equiv \quad & \{ \text{ Definition of } \models^{\mathcal{HI}} \ \} \\
& M' \models^{v} \psi
\end{aligned}$$

The remaining cases are straightforward. □

Corollary 1. *Together, Theorems 4 and 5 entail that given a signature* $(\Delta, \Sigma) \in |Sign^{\mathcal{HI}}|$*, and sentence* $\rho \in Sen^{\mathcal{HI}}(\Delta, \Sigma)$*,*

ρ *is satisfiable iff* $\sigma(\rho \wedge A\,\eta(\rho))$ *is satisfiable.*

Since $\mathcal{HPL}$ is decidable and the equivalence above holds, it is possible to use the decision procedure of $\mathcal{HPL}$ to show the (un)satisfiability of ρ. This approach defines an effective decision procedure for $\mathcal{HI}$, and thus shows that the latter is decidable, which leads to the expected result

Corollary 2. *If* I *is decidable then* $\mathcal{HI}$ *is also decidable.*

Moreover, note that the strategy that underlies the proof of Theorem 5 paves the way to a *constructive* decision algorithm for $\mathcal{HI}$; *i.e.*, a decision algorithm that in the case of the input sentence ρ being satisfiable, provides a witnessing model. For validation purposes, this model may serve as a counter-example of some property (about the system) that is put to test.

Technically, such an algorithm relies on constructive decision algorithms for both I and $\mathcal{HPL}$—the latter has at least one prover that meets this requirement [31]. Then, as indicated in the proof, through a $\mathcal{HPL}$ decision procedure, one extracts a Kripke frame for the input sentence in which suitable models of I are 'attached' (given by the constructive decision algorithm for I). Note, however, that the algorithm may be computationally hard: for example, it may happen that in order to define $\eta(\rho)$, the decision algorithm for I must be executed 2^{n} times where $n = |B_{\rho}|$.

$$\frac{@_i\neg\rho}{\neg@_i\rho}\ (\neg) \qquad \rho \notin Sen^I(\Sigma)$$

$$\frac{\neg@_i\psi}{@_i\neg\psi}\ (\neg\downarrow) \qquad \psi \in Sen^I(\Sigma)$$

$$\frac{@_i(\rho\wedge\rho')}{@_i\rho,@_i\rho'}\ (\wedge) \qquad \rho,\rho' \notin Sen^I(\Sigma)$$

$$\frac{@_i\psi,@_i\psi'}{@_i(\psi\wedge\psi')}\ (\wedge\downarrow) \qquad \psi,\psi' \in Sen^I(\Sigma)$$

$$\frac{\neg@_i\neg\rho}{@_i\rho}\ (\neg\neg)$$

$$\frac{\neg@_i(\rho\wedge\rho')}{\neg@_i\rho \mid \neg@_i\rho'}\ (\neg\wedge)$$

$$\frac{@_i@_j\rho}{@_j\rho}\ (@)$$

$$\frac{\neg@_i@_j\rho}{\neg@_j\rho}\ (\neg@)$$

$$\frac{@_i\mathrm{E}\rho}{@_j\rho}\ (\mathrm{E}) \qquad j \text{ is fresh}$$

$$\frac{\neg@_i\mathrm{E}\rho}{\neg@_l\rho}\ (\neg\mathrm{E}) \qquad l \in Nom$$

$$\frac{@_i\langle\lambda\rangle\rho}{@_k\rho,@_i\langle\lambda\rangle k}\ (\langle\lambda\rangle) \qquad k \text{ is fresh},\ \rho \notin Nom$$

$$\frac{\neg@_i\langle\lambda\rangle\rho,@_i\langle\lambda\rangle l}{\neg@_l\rho}\ (\neg\langle\lambda\rangle) \qquad l \in Nom$$

$$\frac{}{@_i i}\ (\mathrm{R}) \qquad i \in Nom$$

$$\frac{@_i j,@_i\rho}{@_j\rho}\ (\mathrm{N1}) \qquad \rho \in Sen^I(\Sigma)\cup Nom$$

$$\frac{@_i k,@_i\langle\lambda\rangle j}{@_k\langle\lambda\rangle j}\ (\mathrm{N2}) \qquad j \in Nom$$

Fig. 2. The tableau $\mathcal{T}^{\mathcal{H}I}$ (based on the tableau system for hybrid logic in [2]).

4. Generation of a tableau for the hybridised logic

4.1. The method

Let us now discuss how to generate a tableau for the hybridised logic, in complement to the generation of an (Hilbert) calculus discussed in the last section. Actually, prone to computational support, tableau systems offer to the software engineer automatic methods of verification, whereas Hilbert calculi, despite simple and versatile, often require intensive human assistance for non trivial proofs. Another key feature of tableau systems is their ability to provide counter-examples when some wrong statement about the system is put to test. This helps the engineer to locate flawed designs, and, overall, turns the validation process more agile.

Tableau systems are driven by a set of rules, but, differently from other families of proof systems, they cater for the possibility of executions paths to diverge. Actually, when validating a sentence, tableau systems tend to open a number of execution paths, also called branches, each of them is expected to be examined, through sentence decomposition, until contradictions are exposed or no further rules can be applied. If the former case occurs the branch closes; otherwise, it is said to become saturated.

Generally speaking, when checking the validity of a sentence its negation is fed to a suitable tableau: if all branches close – which means that all possibilites have contradictions – the negated sentence is unsatisfiable and therefore the assertion (*i.e.*, the original sentence) is found valid. On the other hand, if some branch saturates one can, in principle, extract a model for the negated sentence that serves as a counter-example of the assertion being tested. A detailed account on tableau systems can be found for example, in references [10] and [2], the latter specialised on tableau systems for hybrid logic.

Let I be a Boolean complete institution with proofs. The tableau system for its hybridisation, $\mathcal{T}^{\mathcal{H}I}$, is driven by the set of rules in Fig. 2. Before letting a branch to saturate, an extra test is added: each sentence of the type $@_i\psi$, where $\psi \in Sen^I(\Sigma)$, must be satisfiable (this is checked through functor Prf^I). The branch closes if it fails the test; otherwise it becomes saturated.

Since the rules in $\mathcal{T}^{\mathcal{H}I}$ only cater for sentences of the type $@_i\rho, \neg@_i\rho$, a given input sentence ϕ is replaced, at the beginning, by $@_0\phi$ where 0 is a fresh nominal, *i.e.*, the root sentence is prefixed by $@_0$. Note that the process preserves satisfiability.

The next example illustrates the mechanisms of $\mathcal{T}^{\mathcal{H}I}$.

Example 6. Recall rule (*Dist*), introduced in the previous section; it states that $@_i(\rho \to \rho') \to (@_i\rho \to @_i\rho')$. Thus, instantiating to classical hybrid propositional logic, one gets:

$$
\begin{aligned}
& @_i(p \to q) \to (@_i p \to @_i q) \\
\equiv\ & (@_i(p \to q) \wedge @_i p) \to @_i q \\
\equiv\ & \neg((@_i(p \to q) \wedge @_i p) \wedge \neg @_i q) \\
\equiv\ & \neg((@_i\neg(p \wedge \neg q) \wedge @_i p) \wedge \neg @_i q)
\end{aligned}
$$

Then, its negation, $@_i\neg(p \wedge \neg q) \wedge @_i p \wedge \neg @_i q$, is fed to the tableau which computes

$@_0(@_i\neg(p \wedge \neg q) \wedge @_i p \wedge \neg @_i q)$	
$@_0@_i\neg(p \wedge \neg q), @_0@_i p, @_0\neg @_i q$	$(\wedge)$
$@_i\neg(p \wedge \neg q), @_i p, \neg @_0@_i q$	$(@, \neg)$
$@_i(\neg(p \wedge \neg q) \wedge p), \neg @_i q$	$(\wedge\downarrow, \neg@)$
$@_i(\neg(p \wedge \neg q) \wedge p \wedge \neg q)$	$(\neg\downarrow, \wedge\downarrow)$

Now, as defined above, the tableau resorts to a prover of the base logic (that corresponds to Prf^I) to check the satisfiability of sentence $\neg(p \wedge \neg q) \wedge p \wedge \neg q$. For example, the tableau system of propositional logic, driven by the rules,

$$\frac{p \wedge q}{p, q}\ (\wedge) \qquad \frac{\neg\neg p}{p}\ (\neg\neg)$$

$$\frac{\neg(p \wedge q)}{\neg p \mid \neg q}\ (\neg\wedge)$$

leads to

$\neg(p \wedge \neg q) \wedge p \wedge \neg q$		
$\neg(p \wedge \neg q), p, \neg q$		$(\wedge)$
$\neg p, p, \neg q$	$q, p, \neg q$	$(\neg\wedge)$
$\times$	$\times$	

Therefore, the test fails, the branch closes, and the unsatisfiability of the input is disclosed. This means that sentence $@_i(p \to q) \to @_i p \to @_i q$ is indeed valid.

▲

4.2. Soundness and completeness

This section shows that any tableau system generated as explained above, is sound and complete whenever the corresponding proof system for the base logic is as well.

To prove that a tableau system is sound, it usually suffices to show that each rule preserves satisfiability.

Theorem 6 *(Soundness). Given an institution I with a sound proof system Prf^I, the tableau system $\mathcal{T}^{\mathcal{H}I}$ is* sound*; i.e., given any signature $(\Delta, \Sigma) \in |Sign^{\mathcal{H}I}|$, and sentence $\rho \in Sen^{\mathcal{H}I}(\Delta, \Sigma)$, ρ being satisfiable entails that any tableau for ρ has at least one branch that does not close.*

Proof. Let us start by showing that rules $(\wedge\downarrow), (\neg\downarrow)$ preserve satisfiability. For any signature $(\Delta, \Sigma) \in |Sign^{\mathcal{H}I}|$, model $M \in |Mod^{\mathcal{H}I}(\Delta, \Sigma)|$ and base sentences $\psi_1, \psi_2 \in Sen^I(\Sigma)$,

$$
\begin{aligned}
& M \models^w @_i\psi_1 \text{ and } M \models^w @_i\psi_2 \\
\equiv\ & \quad \{\ \text{Definition of } \models^{\mathcal{H}}\ \} \\
& M \models^{M_i} \psi_1 \text{ and } M \models^{M_i} \psi_2 \\
\equiv\ & \quad \{\ \text{Definition } \models^{\mathcal{H}}\ \}
\end{aligned}
$$

$$
\begin{aligned}
& m(M_i) \models \psi_1 \text{ and } m(M_i) \models \psi_2 \\
\equiv \quad & \{ \text{ Definition } \models^I \ \} \\
& m(M_i) \models \psi_1 \wedge \psi_2 \\
\equiv \quad & \{ \text{ Definition } \models^{\mathcal{H}} \ \} \\
& M \models^{M_i} \psi_1 \wedge \psi_2 \\
\equiv \quad & \{ \text{ Definition } \models^{\mathcal{H}} \ \} \\
& M \models^{w} @_i(\psi_1 \wedge \psi_2)
\end{aligned}
$$

For rule $(\neg \downarrow)$,

$$
\begin{aligned}
& M \models^{w} \neg @_i \psi_1 \\
\equiv \quad & \{ \text{ Definition of } \models^{\mathcal{H}} \ \} \\
& M \not\models^{w} @_i \psi_1 \\
\equiv \quad & \{ \text{ Definition of } \models^{\mathcal{H}} \ \} \\
& M \not\models^{M_i} \psi_1 \\
\equiv \quad & \{ \text{ Definition } \models^{\mathcal{H}} \ \} \\
& m(M_i) \not\models \psi_1 \\
\equiv \quad & \{ \text{ Definition } \models^{I} \ \} \\
& m(M_i) \models \neg \psi_1 \\
\equiv \quad & \{ \text{ Definition } \models^{\mathcal{H}} \ \} \\
& M \models^{M_i} \neg \psi_1 \\
\equiv \quad & \{ \text{ Definition } \models^{\mathcal{H}} \ \} \\
& M \models^{w} @_i \neg \psi_1
\end{aligned}
$$

The remaining cases are proved in a similar way. Then, to finish the proof, note that Prf^I is sound and therefore the test that regards satisfiability of base sentences only closes branches with contradictions; more concretely, branches with some unsatisfiable sentence of the type $@_i \psi$ where $\psi \in Sen^I(\Sigma)$. □

We now consider completeness.

Theorem 7. *Consider an institution I with a complete proof system Prf^I and the explicit satisfaction property. Then, the tableau system $\mathcal{T}^{\mathcal{H}I}$ is* complete, i.e.*, given any signature $(\Delta, \Sigma) \in |Sign^{\mathcal{H}I}|$, and sentence $\rho \in Sen^{\mathcal{H}I}(\Delta, \Sigma)$, if some branch saturates for ρ, then ρ is satisfiable.*

Proof. Suppose that some branch saturates for ρ. Then, we are able to build model $(W, R, m) \in |Mod^{\mathcal{H}I}(\Delta, \Sigma)|$, as follows

- $W = (N / \sim)$, where N denotes the set of nominals that occur in the branch, and $\sim$ is the equivalence relation generated by the sentences in the branch of the type $@_i j$. Note that rules (R) and (N1) guarantee that $\sim$ is an equivalence relation,
- for any $n \in Nom$, $M_n = [n]$, where $[n]$ denotes the equivalence class of n,
- for any $\lambda \in \Lambda$, $w, v \in W$, $(w, v) \in R_\lambda$ iff there is some nominal $n \in Nom$ such that $n \sim v$ and sentence $@_w \langle \lambda \rangle n$ occurs in the branch,
- for any $w \in W$, $m(w)$ is a model of $|Mod^I(\Sigma)|$ for a sentence $\chi \in Sen^I(\Sigma)$ where $@_w \chi$ is a sentence that occurs in the branch's leaf (Prf^I is complete and I has the explicit satisfaction property). If no such sentence exists, $m(w)$ is a model for $\top$.

It remains to show that there is some $w \in W$ such that $(W, R, m) \models^w \rho$. We prove this by showing that the following statements are true

- if $@_i \varphi$ occurs in the branch then $M \models^w @_i \varphi$,

- if $\neg @_i \varphi$ occurs in the branch then $M \not\models^w @_i \varphi$,

for any sentence $\varphi \in Sen^{\mathcal{H}I}(\Delta, \Sigma)$. This is done by induction on the sentence's structure. In particular,

- $@_i j$

$@_i j$ occurs in the branch
$\Rightarrow$ { Definition of $\sim$ }
$i \sim j$
$\Rightarrow$ { Definition of M }
$M_i = M_j$
$\Rightarrow$ { Definition of $\models^{\mathcal{H}}$ }
$M \models^w @_i j$

- $\neg @_i j$

$\neg @_i j$ occurs in the branch
$\Rightarrow$ { Definition of $\sim$ }
$i \nsim j$
$\Rightarrow$ { Definition of M }
$M_i \neq M_j$
$\Rightarrow$ { Definition of $\models^{\mathcal{H}}$ }
$M \not\models^w @_i j$
$\Rightarrow$ { Definition of $\models^{\mathcal{H}}$ }
$M \models^w \neg @_i j$

- $@_i \psi$

$@_i \psi$ occurs in the branch
$\Rightarrow$ { Application of rule $(\wedge \downarrow)$ }
ψ is a clause of some sentence $@_i \chi$ in the branch's leaf where
$\chi \in Sen^I(\Sigma)$
$\Rightarrow$ { Application of rule (N1), definition of M ($M_i = [i]$) }
$M(M_i) \models \psi$
$\Rightarrow$ { Definition of $\models^{\mathcal{H}}$ }
$M \models^w @_i \psi$

- $\neg @_i \psi$

$\neg @_i \psi$ occurs in the branch
$\Rightarrow$ { Application of rule $(\neg \downarrow)$ }
$\neg \psi$ is a clause of some sentence $@_i \chi$ in the branch's leaf where
$\chi \in Sen^I(\Sigma)$
$\Rightarrow$ { Application of rule (N1), definition of M ($M_i = [i]$) }

$M(M_i) \models \neg\psi$

$\Rightarrow$ { Definition of $\models^I$ }

$M(M_i) \not\models \psi$

$\Rightarrow$ { Definition of $\models^{\mathcal{H}}$ }

$M \not\models @_i\psi$

$\Rightarrow$ { Definition of $\models^{\mathcal{H}}$ }

$M \models^w \neg@_i\psi$

- $@_i\langle\lambda\rangle\rho$

$@_i\langle\lambda\rangle\rho$ occurs in the branch

$\Rightarrow$ { Application of rule ($\langle\lambda\rangle$) }

$@_k\rho, @_i\langle\lambda\rangle k$ occur in the branch

$\Rightarrow$ { Induction hypothesis }

$M \models^w @_k\rho$ and $@_i\langle\lambda\rangle k$ occurs in the branch

$\Rightarrow$ { Application of rule (N2), definition of M }

$M \models^w @_k\rho$ and $(M_i, M_k) \in R_\lambda$

$\Rightarrow$ { Definition of $\models^{\mathcal{H}}$ }

$M \models^{M_i} \langle\lambda\rangle\rho$

$\Rightarrow$ { Definition of $\models^{\mathcal{H}}$ }

$M \models^w @_i\langle\lambda\rangle\rho$

- $\neg@_i\langle\lambda\rangle\rho$

$\neg@_i\langle\lambda\rangle\rho$ occurs in the branch

$\Rightarrow$ { Definition of M and rule $\neg\langle\lambda\rangle$ }

for any $v \in W$ such that $(M_i, v) \in R_\lambda$, $\neg@_v\rho$

$\Rightarrow$ { Induction hypothesis }

for any $v \in W$ such that $(M_i, v) \in R_\lambda$, $M \models^w \neg@_v\rho$

$\Rightarrow$ { Definition of $\models^{\mathcal{H}}$ }

for any $v \in W$ such that $(M_i, v) \in R_\lambda$, $M \not\models^v \rho$

$\Rightarrow$ { Duality between existential and universal quantification }

there is no $v \in W$ such that $(M_i, v) \in R_\lambda$ and $M \models^v \rho$

$\Rightarrow$ { Definition of $\models^{\mathcal{H}}$ }

$M \not\models^{M_i} \langle\lambda\rangle\rho$

$\Rightarrow$ { Definition of $\models^{\mathcal{H}}$ }

$M \not\models @_i\langle\lambda\rangle\rho$

$\Rightarrow$ { Definition of $\models^{\mathcal{H}}$ }

$M \models \neg@_i\langle\lambda\rangle\rho$

The remaining cases are straightforward. □

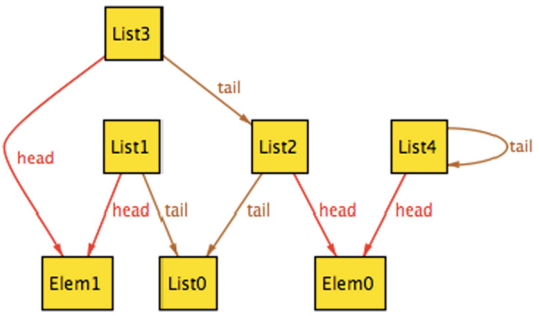


Fig. 3. Several instances of a list in Alloy.

4.3. An illustration in $\mathcal{H}$Alloy—the reconfigurable buffers

Increasingly popular both in industry and academia, Alloy [32] is a lightweight model finder for software design whose language is based on single sorted relational logic extended with a transitive closure operator—hence its motto: *everything is a relation*. Adding to this, Alloy has the ability to automatically validate specifications with respect to bounded domains, and, moreover, to graphically depict counter-examples of flawed assertions.

In order be able to hybridise Alloy specifications, to capture reconfigurable systems, but also, in a wider perspective, to 'connect' it to a vast network of logics and provers [33]—Neves et al. [34,28] introduced an institution for Alloy along with suitable translations to (variants of) first-order and second-order logics. This makes possible not only to hybridise Alloy but also to verify the corresponding specifications in powerful provers such as SPASS [35] and LEO-II [36].

Here, however, our focus is the development of dedicated tool support for $\mathcal{H}$Alloy, based on the tableau generation method. Thus, this section illustrates the potentialities of the method through an example of $\mathcal{T}^{\mathcal{H}\text{Alloy}}$ at work. The case study concerns the specification of a reconfigurable buffer, addressed in documents [8,37] through hybridised *partial* logic.

Consider a buffer that stores and pops out client requests. In general, the store and pop operations follow the FIFO strategy. However, when client requests increase, the buffer adapts by starting to behave as a LIFO system. A question that is typically asked in this context is the following: *once known the expected behaviour for its different settings, is it possible to discern the current execution mode?* To answer this question, we start by defining in Alloy the notion of a buffer as a list, *i.e.*, a set `List` equipped with the following relations

```
head : List → Elem
tail : List → List
```

where for each $\mathtt{l} \in \mathtt{List}$, its head and tail ($\mathtt{l} \cdot \mathtt{head}$, $\mathtt{l} \cdot \mathtt{tail}$) have at most cardinality one. Recall that operator $\cdot$ denotes relation composition. Then, it is necessary to force exactly one empty list to exist, and any other to have its `head` and `tail` well-defined.

```
one l : List | l · head ⊆ ∅
one l : List | l · tail ⊆ ∅
one l : List | l · head ⊆ ∅ and l · tail ⊆ ∅
```

At this stage, Alloy can already provide several instances of a list. For example, Fig. 3 depicts lists: $\mathtt{List0} = []$, $\mathtt{List1} = [b]$, $\mathtt{List2} = [a]$, $\mathtt{List3} = [b, a]$ and $\mathtt{List4} = [a, a, a, \ldots]$ where $\mathtt{Elem0} = a$ and $\mathtt{Elem1} = b$.

The next step is to define the `pop` relation

```
pop : List → List
```

and the possible execution modes. In particular, we state that the system has only two possible execution modes

```
FIFO ≡ ¬LIFO
```

and define the behaviour of `pop` at `FIFO` and `LIFO` as

```
@FIFO

   all l : List | ¬ l · tail = empty →
                      (l · pop) · head = l · head and
                      (l · pop) · tail = (l · tail) · pop
   all l : List | l · tail = empty → l · pop = empty

@LIFO

   all l : List | ¬ l = empty → l · pop = l · tail
   all l : List | l = empty → l · pop = empty
```

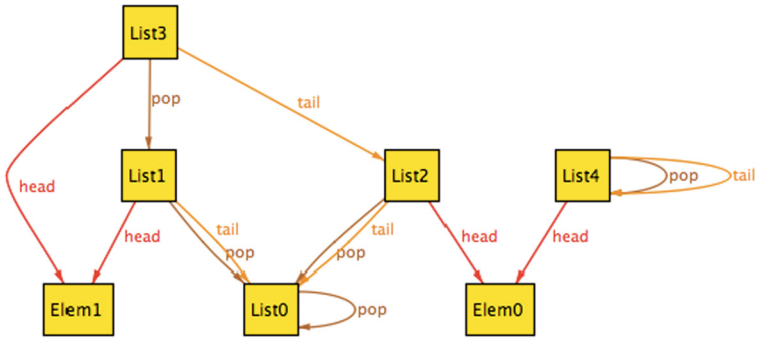


Fig. 4. Examples of pop in action at state FIFO.

Let us denote the axiomatics of pop at FIFO by $@_{\texttt{FIFO}}\psi_1$ and at LIFO by $@_{\texttt{LIFO}}\psi_2$. Alloy can also show the behaviour of pop at FIFO or at LIFO; Fig. 4 shows the behaviour of pop at FIFO with the lists mentioned above: $\texttt{pop}([]) = [], \texttt{pop} = ([b]) = [], \texttt{pop}([a]) = [], \texttt{pop}([b, a]) = [b]$ and $\texttt{pop}([a, a, a, \dots]) = [a, a, a, \dots]$.

We are now ready to answer our original question. Clearly, in models with just the empty list, singleton lists and lists with only element repetition, it is impossible to observe and distinguish the current execution mode. Indeed, in these cases pop at FIFO behaves as pop at LIFO. But what happens in the case of a list whose first element is different from the second? Formally,

```
φ1 ≡ some l : List | ¬ (l · tail) · head = l · head and
                     ¬ (l · tail) = empty
```

It turns out that, when such a condition is true, for any Alloy model with no more than four elements and fifty lists it is possible to distinguish the current execution mode with the test

```
φ2 ≡ all l : List | ¬ l = empty → l · pop = l · tail
```

Indeed, as tableau $\mathcal{T}^{\mathcal{H}\text{Alloy}}$ proves the validity of the sentence below, it also proves that the proposition holds.

$$((\texttt{FIFO} \vee \texttt{LIFO}) \wedge @_{\texttt{FIFO}}\psi_1 \wedge @_{\texttt{LIFO}}\psi_2 \wedge \phi_1) \rightarrow (\phi_2 \rightarrow \texttt{LIFO})$$
$$\equiv ((\texttt{FIFO} \vee \texttt{LIFO}) \wedge @_{\texttt{FIFO}}\psi_1 \wedge @_{\texttt{LIFO}}\psi_2 \wedge \phi_1 \wedge \phi_2) \rightarrow \texttt{LIFO}$$
$$\equiv \neg((\texttt{FIFO} \vee \texttt{LIFO}) \wedge @_{\texttt{FIFO}}\psi_1 \wedge @_{\texttt{LIFO}}\psi_2 \wedge \phi_1 \wedge \phi_2 \wedge \neg\texttt{LIFO})$$
$$\equiv \neg(\neg(\neg\texttt{FIFO} \wedge \neg\texttt{LIFO}) \wedge @_{\texttt{FIFO}}\psi_1 \wedge @_{\texttt{LIFO}}\psi_2 \wedge \phi_1 \wedge \phi_2 \wedge \neg\texttt{LIFO})$$

Its negation, $\neg(\neg\texttt{FIFO} \wedge \neg\texttt{LIFO}) \wedge @_{\texttt{FIFO}}\psi_1 \wedge @_{\texttt{LIFO}}\psi_2 \wedge \phi_1 \wedge \phi_2 \wedge \neg\texttt{LIFO}$, is fed to the tableau which calculates

$@_0(\neg(\neg\texttt{FIFO} \wedge \neg\texttt{LIFO}) \wedge @_{\texttt{FIFO}}\psi_1 \wedge @_{\texttt{LIFO}}\psi_2 \wedge \phi_1 \wedge \phi_2 \wedge \neg\texttt{LIFO})$	
$@_0\neg(\neg\texttt{FIFO} \wedge \neg\texttt{LIFO}), @_{\texttt{FIFO}}\psi_1, @_{\texttt{LIFO}}\psi_2, @_0\phi_1, @_0\phi_2, @_0\neg\texttt{LIFO}$	$(\wedge, @)$
$@_0\neg(\neg\texttt{FIFO} \wedge \neg\texttt{LIFO}), @_{\texttt{FIFO}}\psi_1, @_{\texttt{LIFO}}\psi_2, @_0\phi_1, @_0\phi_2, \neg@_0\texttt{LIFO}$	$(\neg)$
$@_0\neg(\neg\texttt{FIFO} \wedge \neg\texttt{LIFO}), @_{\texttt{FIFO}}\psi_1, @_{\texttt{LIFO}}\psi_2, @_0(\phi_1 \wedge \phi_2), \neg@_0\texttt{LIFO}$	$(\wedge\downarrow)$

Then,

$@_0\texttt{FIFO}$	$@_0\texttt{LIFO}, \neg@_0\texttt{LIFO}$	$(\neg\wedge)$
$@_{\texttt{FIFO}}\psi_1, @_{\texttt{FIFO}}(\phi_1 \wedge \phi_2)$	$\times$	(N1)
$@_{\texttt{FIFO}}(\psi_1 \wedge \phi_1 \wedge \phi_2)$	$\times$	$(\wedge\downarrow)$
$\times$	$\times$	No model for $\psi_1 \wedge \phi_1 \wedge \phi_2$.

Alloy cannot find a model up to four elements and fifty lists for $\psi_1 \wedge \phi_1 \wedge \phi_2$, which means that, whenever no base model exceeds these domains, our assertion is valid.

5. Conclusions and future work

Despite the major advantages of working in a single logical setting, current software complexity often forces the engineer to use multiple logics in the specification of a single software system. Hence, it comes as no surprise the emergence of several mechanisms for combining logics (*e.g.* [38–42]). From a computer science point of view, the programme is even broader because, as Goguen and Meseguer wrote in [43],

"The right way to combine various programming paradigms is to discover their underlying logics, combine them, and then base a language upon the combined logic."

Indeed, the hybridisation method can be more broadly understood as a specific way of combining logics at model theoretical level. Actually, it classifies as *a tool for simplifying problems involving heterogeneous reasoning* [44], a common ingredient to this family of methods according to the corresponding entry in the *Stanford Encyclopedia of Philosophy*.

Hybridisation is, thus, an asymmetric combination of logics in the sense that specific features of hybrid logic are developed 'on top' of another logic. As mentioned in the Introduction, this follows exactly the same steps, and to a certain extent extends, previous work by R. Diaconescu and P. Stefaneas [38] on 'modalisation' of institutions, which endows systematically institutions with Kripke semantics for standard modalities. R. Fajardo and M. Finger introduced in [45] an alternative method to modalise logics, and proved preservation of both completeness and decidability of the source logics. Other examples, in a similar research line, include the 'temporalisation' of logics introduced by M. Finger and D. Gabbay in [39] and the more recent 'probabilisation' of logics introduced by P. Baltazar in [40]. The work of A. Costa Leite on what he calls *paraconsistentization of logics* [46] goes in a similar direction investigating how the paraconsistent counterpart of an arbitrary logic can be obtained.

This sort of approaches were generalised by C. Caleiro, A. Sernadas and C. Sernadas in [41], in a method called *parameterization*. In brief, a logic is parametrized by another one if the atomic part of the former is replaced by the latter. The recent method of *importing logics* suggested by J. Rasga, A. Sernadas and C. Sernadas [47] aims at formalising this kind of asymmetric combinations resorting to a graph-theoretic approach.

From a wider perspective, combination of logics is increasingly recognised as a relevant research domain, driven not only by philosophical enquiry on the nature of logics or strict mathematical questions, but also from applications in computer science and artificial intelligence. The first methods appeared in the context of modal logics. This includes *fusion* of the underlying languages [48], pioneered by M. Fitting, in a 1969 paper combining alethic and deontic modalities [49], and *product of logics* [50]. Both approaches can be characterised as symmetric. Product, for example, amounts to pairing the Kripke semantics, i.e., the accessibility relations, of both logics. With a wider scope of application, i.e. beyond modal logics, *fibring* [51] was originally proposed by D. Gabbay, and contains fusion as a particular case. From a syntactic point of view the language of the resulting logic is freely generated from the signatures of the combined logics, symbols from both of them appearing intertwined in an arbitrary way.

Reference [52] offers an excellent roadmap for the several variants of fibring in the literature. A particularly relevant evolution was the work of A. Sernadas and his collaborators resorting to universal constructions from category theory to characterise different patterns of connective sharing, as documented in [53]. In the simplest case, where no constraint is imposed by sharing, fibring is the least extension of both logics over the coproduct of their signatures, which basically amounts to a coproduct of logics. This approach, usually referred to as *algebraic fibring*, makes heavy use of categorial constructions as a source of genericity to provide more general and wide applicable methods.

The use of the theory of institutions [54] as a foundation for hybridisation, as described in this paper, or for modalisation in [38], has a similar motivation: going categorial is going generic. Actually, a proper setting to discuss the generation of new logics form old, and to identify the sort of properties preserved or reflected along such a process, always requires a generic definition of what a logic and a logic system is.

Serving as a framework for the specification of reconfigurable systems, the hybridisation method has been extended to include equivalence and refinement [11], initial semantics [55], and, on the verification side, suitable translations to first-order logic [6,8]. Recently, hybridisation was implemented [7] in the HETS platform [33]. However, contrary to what happened, for example with *temporalisation* [39] and *probabilisation* [40], the proof theory for hybrid(ised) logics was only recently discussed in reference [9]. The very particular case of equational hybrid logic was addressed in [56].

Document [9] gave the first step in this line of research by showing how an Hilbert calculus for the hybridised version of a logic can be systematically generated from a calculus for the latter. The current paper goes one step beyond by simplifying the previous method, and providing a similar process for generating tableau systems. Such developments form a sound basis for a complete proof theory, which, from a pragmatic point of view, paves the way to dedicated proof support for a broad spectrum of hybrid(ised) logics.

Actually, the next natural step in this direction is to 'extract' the algorithms developed in this paper and implement them in the HETS platform where provers of different logics can communicate with each other (and consequently where hybridisation's potentialities are maximised). Then, a comparison with the strategy of using the parametrised translation to first-order logic will be in order.

The completeness results that this paper reports rely on the assumption that the base institution has the explicit satisfaction property. Although prevalent for the logics used in software specification, such a property does not hold in hybridised logics. It can, however, be regained by relaxing the satisfaction definition into

$$M \models^{\mathcal{H}I}_{(\Delta,\Sigma)} \rho \text{ iff } M \models^{w} \rho, \text{ for some } w \in W$$

where M is the typical model of an hybridised logic and ρ a compatible sentence. This means that in a multiple hybridisation scenario [57], soundness and completeness of the corresponding calculi, as well as decidability, can still be obtained.

Complexity issues of the hybridised logics, although out of the scope of this paper, cannot be ignored if this work is to be taken as a basis for the construction of a computational proof tool. In this context the techniques proposed in [58,59], which underly the Sibyl prover, should be explored.

Acknowledgements

The authors are grateful to Torben Bräuner for helpful, inspiring discussions, and to the anonymous referees for their detailed comments.

This work is funded by ERDF—European Regional Development Fund, through the COMPETE Programme, and by National Funds through Fundação para a Ciência e a Tecnologia (FCT) within project PTDC/EEI-CTP/4836/2014. Moreover, the first and the second authors are sponsored by FCT grants SFRH/BD/52234/2013 and SFRH/BPD/103004/2014, respectively. M. Martins is also supported by the EU FP7 Marie Curie PIRSES-GA-2012-318986 project GeTFun: Generalizing Truth-Functionality and FCT project UID/MAT/04106/2013 through CIDMA. L. Barbosa is further supported by FCT in the context of SFRH/BSAB/113890/2015.

References

[1] C. Areces, B. ten Cate, Hybrid logics, in: P. Blackburn, F. Wolter, J. van Benthem (Eds.), Handbook of Modal Logics, Elsevier, 2006.
[2] T. Braüner, Proof-theory of propositional hybrid logic, in: Hybrid Logic and Its Proof-Theory, 2011.
[3] R. Szepesia, H. Ciocârlie, An overview on software reconfiguration, Theory Appl. Math. Comput. Sci. 1 (2011) 74–79.
[4] A. Madeira, J.M. Faria, M.A. Martins, L.S. Barbosa, Hybrid specification of reactive systems: an institutional approach, in: G. Barthe, A. Pardo, G. Schneider (Eds.), Software Engineering and Formal Methods, SEFM 2011, Montevideo, Uruguay, November 14–18, 2011, in: Lecture Notes in Computer Science, vol. 7041, Springer, 2011, pp. 269–285.
[5] A. Madeira, Foundations and techniques for software reconfigurability (an institution-independent approach to specifying and reasoning about reconfigurable systems), Ph.D. thesis, Minho, Aveiro and Porto Universities, July 2013 (MAP-i Doctoral Programme).
[6] M.A. Martins, A. Madeira, R. Diaconescu, L.S. Barbosa, Hybridization of institutions, in: A. Corradini, B. Klin, C. Cîrstea (Eds.), Algebra and Coalgebra in Computer Science, CALCO 2011, Winchester, UK, August 30–September 2, 2011, in: Lecture Notes in Computer Science, vol. 6859, Springer, 2011, pp. 283–297.
[7] R. Neves, A. Madeira, M.A. Martins, L.S. Barbosa, Hybridisation at work, in: R. Heckel, S. Milius (Eds.), Algebra and Coalgebra in Computer Science, CALCO 2013, Warsaw, Poland, September 3–6, 2013, in: Lecture Notes in Computer Science, vol. 8089, Springer, 2013, pp. 340–345.
[8] R. Diaconescu, A. Madeira, Encoding hybridized institutions into first-order logic, Mathematical Structures in Computer Science FirstView (2015) 1–44.
[9] R. Neves, M.A. Martins, L.S. Barbosa, Completeness and decidability results for hybrid(ised) logics, in: C. Braga, N. Martí-Oliet (Eds.), Formal Methods: Foundations and Applications, SBMF 2014, Maceió, AL, Brazil, September 29–October 1, 2014, in: Lecture Notes in Computer Science, vol. 8941, 2015, pp. 146–161.
[10] R. Goré, Tableau methods for modal and temporal logics, in: M. D'Agostino, D. Gabbay, R. Hähnle, J. Posegga (Eds.), Handbook of Tableau Methods, Springer, Netherlands, 1999, pp. 297–396.
[11] A. Madeira, M. Martins, L. Barbosa, R. Hennicker, Refinement in hybridised institutions, Form. Asp. Comput. 27 (2) (2015) 375–395.
[12] C. Areces, R. Fervari, G. Hoffmann, Swap logic, Log. J. IGPL 22 (2) (2014) 309–332.
[13] C. Areces, R. Fervari, G. Hoffmann, Relation-changing modal operators, Log. J. IGPL 23 (4) (2015) 601–627.
[14] R. Diaconescu, Institution-Independent Model Theory, Birkhäuser, Basel, 2008.
[15] C. Beierle, G. Kern-Isberner, Looking at probabilistic conditionals from an institutional point of view, in: G. Kern-Isberner, W. Rödder, F. Kulmann (Eds.), Conditionals, Information, and Inference, Revised Selected Papers, WCII 2002, Hagen, Germany, May 13–15, 2002, in: Lecture Notes in Computer Science, vol. 3301, Springer, 2005, pp. 162–179.
[16] C. Caleiro, P. Mateus, A. Sernadas, C. Sernadas, Quantum institutions, in: K. Futatsugi, J.-P. Jouannaud, J. Meseguer (Eds.), Algebra, Meaning, and Computation, Essays Dedicated to Joseph A. Goguen on the Occasion of His 65th Birthday, in: Lecture Notes in Computer Science, vol. 4060, Springer, 2006, pp. 50–64.
[17] R. Burstall, R. Diaconescu, Hiding and behaviour: an institutional approach, in: W. Roscoe (Ed.), A Classical Mind: Essays in Honour of C.A.R. Hoare, Prentice-Hall, 1994, pp. 75–92.
[18] M. Bidoit, R. Hennicker, Constructor-based observational logic, J. Log. Algebraic Program. 67 (1–2) (2006) 3–51.
[19] M.A. Martins, D. Pigozzi, Behavioural reasoning for conditional equations, Math. Struct. Comput. Sci. 17 (5) (2007) 1075–1113.
[20] C. Cîrstea, An institution of modal logics for coalgebras, J. Log. Algebraic Program. 67 (1–2) (2006) 87–113.
[21] T. Mossakowski, M. Roggenbach, Structured CSP—a process algebra as an institution, in: J.L. Fiadeiro, P.-Y. Schobbens (Eds.), Recent Trends in Algebraic Development Techniques, Revised Selected Papers, WADT 2006, La Roche en Ardenne, Belgium, June 1–3, 2006, in: Lecture Notes in Computer Science, vol. 4409, Springer, 2006, pp. 92–110.
[22] D. Sannella, A. Tarlecki, Foundations of Algebraic Specification and Formal Software Development, EATCS Monographs on Theoretical Computer Science, Springer, 2012.
[23] L. Schröder, T. Mossakowski, HasCasl: integrated higher-order specification and program development, Theor. Comput. Sci. 410 (12–13) (2009) 1217–1260.
[24] J.A. Goguen, R.M. Burstall, Institutions: abstract model theory for specification and programming, J. ACM 39 (1992) 95–146.
[25] T. Mossakowski, A. Haxthausen, D. Sannella, A. Tarlecki, CASL: The common algebraic specification language: semantics and proof theory, Comput. Inform. 22 (2003) 285–321.
[26] R. Diaconescu, Institutional semantics for many-valued logics, Fuzzy Sets Syst. 218 (2013) 32–52.
[27] J. Agustí-Cullell, F. Esteva, P. Garcia, L. Godo, Formalizing multiple-valued logics as institutions, in: B. Bouchon-Meunier, R. Yager, L. Zadeh (Eds.), Uncertainty in Knowledge Bases, in: Lecture Notes in Computer Science, vol. 521, Springer, Berlin Heidelberg, 1991, pp. 269–278.
[28] R. Neves, A. Madeira, M. Martins, L. Barbosa, An institution for Alloy and its translation to second-order logic, in: T. Bouabana-Tebibel, S.H. Rubin (Eds.), Integration of Reusable Systems, extended versions of the best papers which were presented at IEEE International Conference on Information Reuse and Integration and IEEE International Workshop on Formal Methods Integration, San Francisco, CA, USA, August 2013, in: Advances in Intelligent Systems and Computing, vol. 263, Springer, 2014.
[29] J. Fiadeiro, A. Sernadas, Structuring theories on consequence, in: D. Sannella, A. Tarlecki (Eds.), Recent Trends in Data Type Specification, in: Lecture Notes in Computer Science, vol. 332, Springer, Berlin Heidelberg, 1988, pp. 44–72.
[30] R. Diaconescu, Quasi–Boolean encodings and conditionals in algebraic specification, J. Log. Algebraic Program. 79 (2) (2010) 174–188.

[31] G. Hoffmann, C. Areces, Htab: a terminating tableaux system for hybrid logic, Electron. Notes Theor. Comput. Sci. 231 (2009) 3–19.
[32] D. Jackson, Software Abstractions: Logic, Language, and Analysis, The MIT Press, 2006.
[33] T. Mossakowski, C. Maeder, K. Lüttich, The heterogeneous tool set, Hets, in: O. Grumberg, M. Huth (Eds.), Tools and Algorithms for the Construction and Analysis of Systems, TACAS 2007, Braga, Portugal, March 24–April 1, 2007, in: Lecture Notes in Computer Science, vol. 4424, Springer, 2007, pp. 519–522.
[34] R. Neves, A. Madeira, M.A. Martins, L.S. Barbosa, Giving alloy a family, in: C. Zhang, J. Joshi, E. Bertino, B. Thuraisingham (Eds.), Proceedings of 14th IEEE International Conference on Information Reuse and Integration, IEEE Press, 2013, pp. 512–519.
[35] C. Weidenbach, D. Dimova, A. Fietzke, R. Kumar, M. Suda, P. Wischnewski, SPASS version 3.5, in: R.A. Schmidt (Ed.), Automated Deduction, CADE-22, Montreal, Canada, August 2–7, 2009, in: Lecture Notes in Artificial Intelligence, vol. 5663, Springer, 2009, pp. 140–145.
[36] C. Benzmüller, F. Theiss, L. Paulson, A. Fietzke, LEO-II—a cooperative automatic theorem prover for higher-order logic, in: A. Armando, P. Baumgartner, G. Dowek (Eds.), Automated Reasoning, IJCAR 2008, Sydney, Australia, August 12–15, 2008, in: LNCS, vol. 5195, Springer, 2008, pp. 162–170.
[37] A. Madeira, R. Neves, M.A. Martins, L.S. Barbosa, When even the interface evolves…, in: H. Wang, R. Banach (Eds.), Theoretical Aspects of Software Engineering, TASE 2013, Birmingham, UK, 1–3 July, 2013, IEEE Press, 2013, pp. 79–82.
[38] R. Diaconescu, P. Stefaneas, Ultraproducts and possible worlds semantics in institutions, Theor. Comput. Sci. 379 (1–2) (2007) 210–230.
[39] M. Finger, D. Gabbay, Adding a temporal dimension to a logic system, J. Log. Lang. Inf. 1 (3) (1992) 203–233.
[40] P. Baltazar, Probabilization of logics: completeness and decidability, Log. Univers. 7 (4) (2013) 403–440.
[41] C. Caleiro, C. Sernadas, A. Sernadas, Parameterisation of logics, in: J.L. Fiadeiro (Ed.), Recent Trends in Algebraic Development Techniques, Selected Papers: WADT '98, Lisbon, Portugal, April 2–4, 1998, in: Lecture Notes in Computer Science, vol. 1589, Springer, 1998, pp. 48–62.
[42] J. Rasga, A. Sernadas, C. Sernadas, Importing logics: soundness and completeness preservation, Stud. Log. 101 (1) (2013) 117–155.
[43] J.A. Goguen, J. Meseguer, Models and equality for logical programming, in: H. Ehrig, R. Kowalski, G. Levi, U. Montanari (Eds.), Theory and Practice of Software Development, TAPSOFT'87, Pisa, Italy, March 23–27, 1987, in: Lecture Notes in Computer Science, vol. 250, Springer, Berlin Heidelberg, 1987, pp. 1–22.
[44] W. Carnielli, M.E. Coniglio, Combining logics, in: E.N. Zalta (Ed.), The Stanford Encyclopedia of Philosophy, winter 2011 edition, 2011.
[45] R. Fajardo, M. Finger, Non-normal modalisation, in: P. Balbiani, N. Suzuki, F. Wolter, M. Zakharyaschev (Eds.), Advances in Modal Logic 4, Papers from the Fourth Conference on Advances in Modal Logic, Held in Toulouse, France in October 2002, King's College Publications, 2002, pp. 83–96.
[46] A. Costa-Leite, Interactions of metaphysical and epistemic concepts, Ph.D. thesis, Universite de Neuchatel, Switzerland, 2007.
[47] J. Rasga, A. Sernadas, C. Sernadas, Importing logics, Stud. Log. 100 (3) (2012) 545–581.
[48] R.H. Thomason, Combinations of tense and modality, in: D. Gabbay, F. Guenthner (Eds.), Handbook of Philosophical Logic. Volume II. Extensions of Classical Logic, Springer, Netherlands, 1984, pp. 135–165.
[49] M. Fitting, Logics with several modal operators, Theoria 35 (3) (1969) 259–266.
[50] K. Segerberg, Two-dimensional modal logic, J. Philos. Log. 2 (1) (1973) 77–96.
[51] D. Gabbay, Fibred semantics and the weaving of logics. Part 1, J. Symb. Log. 61 (4) (1996) 1057–1120.
[52] C. Caleiro, A. Sernadas, C. Sernadas, Fibring logics: past, present and future, in: S.N. Artëmov, H. Barringer, A.S. d'Avila Garcez, L.C. Lamb, J. Woods (Eds.), We Will Show Them! Essays in Honour of Dov Gabbay, Volume One, College Publications, 2005, pp. 363–388.
[53] A. Sernadas, C. Sernadas, C. Caleiro, Fibring of logics as a categorial construction, J. Log. Comput. 9 (2) (1999) 149–179.
[54] R.M. Burstall, J.A. Goguen, The semantics of CLEAR, a specification language, in: D. Bjørner (Ed.), Abstract Software Specifications, 1979 Copenhagen Winter School, January 22–February 2, 1979, in: Lecture Notes in Computer Science, vol. 86, Springer, 1980, pp. 292–332.
[55] R. Diaconescu, Quasi-varieties and initial semantics for hybridized institutions, J. Log. Comput. (2013).
[56] L.S. Barbosa, M.A. Martins, M. Carreteiro, A Hilbert-style axiomatisation for equational hybrid logic, J. Log. Lang. Inf. 23 (1) (2014) 31–52.
[57] A. Madeira, R. Neves, M. Martins, L. Barbosa, Introducing hierarchical hybrid logic, in: Short Papers Presented at Advances in Modal Logic 2014, 2014, pp. 74–78.
[58] S. Cerrito, M.C. Mayer, An efficient approach to nominal equalities in hybrid logic tableaux, J. Appl. Non-Class. Log. 20 (1–2) (2010) 39–61.
[59] S. Cerrito, M.C. Mayer, A tableau based decision procedure for an expressive fragment of hybrid logic with binders, converse and global modalities, J. Autom. Reason. 51 (2) (2013) 197–239.