Proof theory for hybrid(ised) logics

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

[a] *HASLab (INESC TEC) & Univ. Minho, Portugal*
[b] *CIDMA – Dep. Mathematics, Univ. Aveiro, Portugal*

Abstract

The prevalent ability of software systems to adapt lead to a formal description technique based on hybridisation — a process that systematically develops the ingredients of hybrid logic on top of whatever logic is found useful for property specification. The engineer, however, looks not only for logics able to capture the requirements of the system at hands, but also for suitable proof support for reasoning upon them. This calls for an enriched version of hybridisation where both the logic and the corresponding calculus are hybridised. In a previous paper, the authors addressed this issue by showing how an Hilbert calculus for the hybridised version of a logic can be systematically generated from a calculus for the latter. This paper provides a simpler version of this process and goes a step forward by characterising a complementary (tableau based) technique, with increased computational support. Such developments provide the basis for a complete proof theory for hybrid(ised) logics, and thus offer (dedicated) proof support for the working software engineer.

Keywords:
Hybrid logic, decidability, completeness, tableau systems, Hilbert calculus

1. Introduction

1.1. Motivation and context

This paper is motivated by a specific class of software systems that are able to adapt (or reconfigure) their behaviour whenever context switches so

Email addresses: rjneves@inescporto.pt (Renato Neves), amadeira@inescporto.pt (Alexandre Madeira), martins@ua.pt (Manuel A. Martins), lsb@di.uminho.pt (Luís S. Barbosa)

demand. Such systems, often called *reconfigurable*, take a central place in software development. In fact, more and more they influence our daily lives, from service–oriented applications that change their services in accordance with the traffic level to controllers embedded in cars that favour power over economy when the 'sports mode' is engaged.

The formal specification of a reconfigurable system is often a challenge: whatever logic the software engineer finds useful to define possible behavioural requirements, it may not be suitable to relate the different contexts in which they hold and express the reconfiguration dynamics. Actually, underlying any reconfigurable system lies a *labelled transition system* whose states correspond to configurations and characterise a specific behaviour; arrows relate two possible configurations and are labelled by the event that triggers the switch. Thus, while this transition system may be specified in (some variant of) modal logic, to describe the possible behaviours of concrete configurations requires a logic that suits the nature of the software system at hands. For example, continuous systems advocate topological logics, and probabilistic systems are better handled through logics that embed some fragment of probability theory. Such a view hints at a framework, proposed in [1], for the specification of *reconfigurable* systems:

- *globally* the system's dynamics is represented by a transition structure described in hybrid logic – an extension of modal logic with enough expressive power to pinpoint specific states but without losing decidability or increasing complexity.

- *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 the one used for the local specification of configurations, thus giving rise to an *hybridised* logic.

The logic used locally, *i.e.* the one to be hybridised, depends 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 [2, 3] and further extended in [4]), but at the cost of losing decidability and adding extra complexity.

Our SBMF paper [5] introduced an alternative approach: not only the logic is hybridised but also its calculus is systematically enriched into a Hilbert calculus for the hybridised logic. Moreover, it showed that the latter is sound and complete whenever the corresponding base calculus is. Such development is a first step towards dedicated proof support for a broad spectrum of hybrid(ised) logics.

Hilbert calculi, however, although simple and versatile, are not amenable to effective computational support, and so constrain the practical relevance of the former results. Therefore, this paper introduces a similar procedure but generating a tableau system instead. Tableau systems [6, 7], able to systematically decompose sentences until contradictions are found, are well-known for their impressive computational results, in particular for the class of modal logics where hybrid(ised) logics live.

1.2. Contributions and roadmap

The paper starts by recasting the hybridisation method in the theory of institutions with proofs, which makes possible the development of the whole framework. Then, it simplifies the generation of Hilbert calculi originally proposed in [5], and introduces the 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.

The remainder of the paper is organised as follows: Section 2 provides the relevant background. Section 3 presents the generation of Hilbert calculi and discusses decidability of hybrid(ised) logics. Section 4 introduces the corresponding tableau version. Finally, Section 5 concludes.

2. Background

2.1. Institutions

The generic character of the hybridisation process is due to its rendering in the context of the theory of institutions [8]. The notion of 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,

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

Intuitively, the property above tells that satisfaction is preserved under change of notation. In order to build up the reader's intuition, let us analyse 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 in this category 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 object $(S, F, P) \in |Sign^{\text{FOL}}|$, $Sen^{\text{FOL}}(S, F, P)$ 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 of sorts 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 under the mapping corresponding to φ.

- MODELS. For each signature $(S, F, P) \in |Sign^{\mathrm{FOL}}|$, $Mod^{\mathrm{FOL}}(S, F, P)$ 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 $\sigma_{w \to s} \in F_{w \to s}$; 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 logic (EQ)
The institution EQ is the sub-institution of FOL in which sentences are restricted to those of the type $\forall \overline{x} : \overline{s} \,.\, t = t'$

▲

Example 3. Propositional logic (PL)
Institution PL is the sub–institution of FOL in which signatures with no empty set of sorts are discarded.

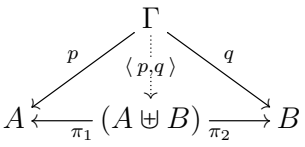
▲

Other examples of institutions underlie the algebraic specification language CASL [9], many–valued logics [10, 11], and the relational–based language ALLOY [12].

However, the classic notion of an institution, does not include an abstract structure to represent associated logic calculi. The problem was addressed in [13] with the introduction of π–*institutions*, and, more recently, in [14] with the notion of an *institution with proofs*, a more general version of the previous work.

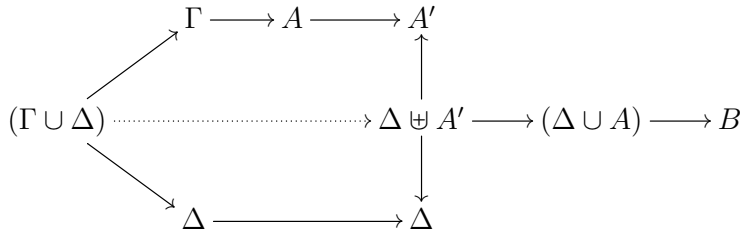
Definition 2. An institution with proofs adds to the original definition of an institution, a functor $Prf^I : Sign^I \to \mathbb{C}at$ such that, for each $\Sigma \in |Sign^I|$, $Prf^I(\Sigma)$ (called the category of Σ–proofs) has subsets of $Sen^I(\Sigma)$

(*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 arrows $\langle p, q \rangle$ that makes the diagram to commute.



For the sake of simplicity, when a singleton set of sentences is presented in a proof arrow, we may drop the curly brackets. Also, observe that the restrictions imposed to the proof arrows force Prf^I to follow the basic properties of a proof system:

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), which provides a clear focus on entailment systems [1], and trivialises the uniqueness property of arrow $\langle p, q \rangle$.

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

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 $A \vdash^I B$. In 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 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. (From [15]) 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 sentence $\neg\rho \in Sen^I(\Sigma)$ such that for any model $M \in |Mod^I(\Sigma)|$, $M \models \rho$ iff $M \not\models \neg\rho$,
- for any sentences $\rho, \rho' \in Sen^I(\Sigma)$, there is 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.

Definition 5. Then, negation makes possible to state that, given an institution I and signature $\Sigma \in |Sign^I|$, for any sentence $\rho \in Sen^I(\Sigma)$,

$$\rho \text{ is } \textit{unsatisfiable} \text{ iff } \neg\rho \text{ is valid.}$$

As usual, $\rho \vee \rho'$ denotes $\neg(\neg\rho \wedge \neg\rho')$ and $\rho \to \rho'$ denotes $\neg(\rho \wedge \neg\rho')$. Also, we have sentence $\rho \wedge \neg\rho$, denoted by $\bot$, that no model in $|Mod^I(\Sigma)|$ satisfies. 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 $\not\vdash^I \neg\rho$.

Proof. Follows from:

- 1. $\Rightarrow$ 2.

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

- 2. $\Rightarrow$ 1.

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

□

2.2. Hybridisation revisited

This subsection reviews the basis of the hybridisation process with the global modality. Document [2] reports a version of hybridisation where universal quantification over worlds and polyadic modalities are also considered.

Definition 6. The category $Sign^{\mathcal{H}}$ is 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 7. 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

- $Sign^{\mathcal{H}\mathcal{I}} = Sign^{\mathcal{H}} \times Sign^{\mathcal{I}}$,

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

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

for i a nominal, λ a modality, $\psi \in Sen^{\mathcal{I}}(\Sigma)$. We use non standard Boolean connectives symbols $(\neg, \barwedge)$ in order to distinguish them from the Boolean connectives that a base logic may have. Also, define $[\lambda]\rho \equiv \neg\langle\lambda\rangle\neg\rho$, $E\,\rho \equiv \neg A \neg\rho$ and $\rho \Rightarrow \rho' \equiv \neg(\rho \barwedge \neg\rho')$. We will use ψ to symbolise a sentence of the base logic.

- given a signature $(\Delta, \Sigma) \in |Sign^{\mathcal{HI}}|$, a model $M \in |Mod^{\mathcal{HI}}(\Delta, \Sigma)|$ is a triple (W, R, m) such that,
  - W is a non–empty 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 $(\Delta, \Sigma) \in |Sign^{\mathcal{HI}}|$, a model $M = (W, R, m) \in |Mod^{\mathcal{HI}}(\Delta, \Sigma)|$ and a sentence $\rho \in Sen^{\mathcal{HI}}(\Delta, \Sigma)$, the satisfaction relation is defined as,

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

where,

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

$(w, v) \in R_\lambda$ and $M \models^v \rho$

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.* [4]). Thus, the grammar of the hybridised logic becomes,

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

Since it turns proofs simpler and more intuitive, we assume that all hybridised logics adopt this approach.

Example 4. Hybridised propositional logic ($\mathcal{H}PL$)

- SIGNATURES are pairs $(\Delta, \Sigma) \in |Sign^{\mathcal{H}PL}|$ where Σ is a set of propositional symbols.
- 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 correspond to a propositional model (a subset of Σ that denotes the propositions that are true).

▲

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 (the latter is decidable and has 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 [5], but 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 Figure 3.1. Note that their schematic form guarantees that the proof arrows are preserved along signature morphisms.

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 $\vdash^{\mathcal{H}I} \big(\rho \wedge \langle\lambda\rangle i\big) \rightarrow @_i\rho'$ 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 symbol * denotes condition *if i does not occur free neither in ρ nor ρ'*.

Figure 1: Axioms and rules for $Prf^{\mathcal{H}I}$ (based on the Hilbert calculus for hybrid logic reported in [6]).

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

Example 5. To show that $[\lambda](\forall\, \overline{x} : \overline{s}\, .\, t = t)$ it is a theorem in $\mathcal{H}EQ$ one starts with

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

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 6. Sentence $\Box(p \rightarrow q) \rightarrow (\Box p \rightarrow \Box q)$ is an instance of the famous 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

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

Proof for the remaining cases is straightforward. □

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

Definition 8. 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

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

Definition 9. 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 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 $\Omega_\rho \not\subseteq \emptyset$. 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 $M \models \chi'$ (again by the conjunction and negation properties.)

□

Definition 10. 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 exist in $Prf^{\mathcal{H}I}$, and that $\sigma(\rho), \rho$ are structurally the same. This implicates that if $\vdash^{\mathcal{H}PL} \sigma(\rho)$, then, whichever rules and axioms were used before, one can imitate the process using the same rules and axioms, thus arriving at $\vdash^{\mathcal{H}I} \rho$. □

Definition 11. Let $\Omega^{\star}_{\rho} = \{\chi \in \Omega_{\rho} \mid \vdash^{I} \neg\chi\}$ and consider the 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, due to rule (A_I), concludes $\vdash^{\mathcal{H}I} A\,\eta(\rho)$. □

Lemma 4. Consider a signature $(\Delta, \Sigma) \in |Sign^{\mathcal{H}I}|$, sentence $\rho \in Sen^{\mathcal{H}I}(\Delta, \Sigma)$ and 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 12. 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 in 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 $\rho \in Sen^{\mathcal{H}I}(\Sigma)$

$$\text{If } \nvdash^{\mathcal{H}I} \neg\rho \text{ then } \rho \text{ is satisfiable}$$

Proof. Start with the observation

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

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 $M' \models^w \rho$, which is done through induction on the subformulas of ρ. For any sentence $\psi \in B_\rho$

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

The remaining cases offer no difficulty. □

3.3. Decidability

The decidability property is a famous concept, traditionally studied under the light of any newly developed logic. Indeed, it is a central element in proof

theory, but it also takes a paradigmatic role in the validation of software systems: a logic is decidable iff it has a decision procedure that can always decide about the validity of any of its sentences. In other words, if the logic is decidable, then the software engineer can always obtain yes–or–no answers to queries that regard properties of the system at hands.

The machinery used before to achieve a proof of completeness provides an interesting opportunity to discuss the decidability of hybrid(ised) logics. More concretely, 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 such 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 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 model M define model $M' = (W, R, m') \in |Mod^{\mathcal{H}PL}(\Delta, \Sigma)|$ 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 through induction on the subformulas of ρ. For any $v \in W$, $\psi \in B_\rho$,

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

The remaining cases are straightforward. □

Next, we redefine function η.

Definition 13. Consider a decidable institution I 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 $(\Delta, \Sigma) \in |Sign^{\mathcal{HI}}|$, sentence $\rho \in Sen^{\mathcal{HI}}(\Delta, \Sigma)$ and 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 ρ also is.

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 $M' = (W, R, m') \in |Mod^{\mathcal{H}I}(\Delta, \Sigma)|$ 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 that I has the explicit satisfaction property). To finish the proof, it remains to show that $(W, R, m') \models^w \rho$, which is done through induction on the structure of ρ. For any sentence $\psi \in B_\rho$, any $v \in W$,

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

The remaining cases are straightforward. □

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

$$\rho \text{ is satisfiable iff } \sigma(\rho \wedge A \neg\eta(\rho)) \text{ is satisfiable.}$$

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

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

Moreover, note that the strategy that underlies the proof of Theorem 5 paves the way for a *constructive* decision algorithm of $\mathcal{H}I$; *i.e.*, a decision algorithm that in the case of the input sentence ρ being satisfiable says not only that it is, but also provides a model that shows. In the task of validation, this model may serve as a counter–example of some property (about the system) that is put to test.

Technically, to construct such an algorithm one also needs to have constructive decision algorithms for both I and $\mathcal{H}PL$ – the latter has at least one prover that meets this requirement [16]. Then, as indicated in the proof,

through a $\mathcal{H}PL$ decision procedure, one extracts a Kripke frame for the input sentence in which suitable models of I are 'attached' given its example decision algorithm for I. Note, however, that the algorithm may be computationally hard: for example, in order to define $\eta(\rho)$ the decision algorithm for I must be executed 2^n times where $n = |B_\rho|$.

4. Generation of a tableau for the hybridised logic

4.1. The method

The current section is devoted to the generation of a tableau for the hybridised logic, which, as already mentioned, complements the method of (Hilbert) calculi generation 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 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 being examined, through sentence decomposition, until contradictions are exposed or no further rules can be applied. If the former case occurs the branch closes, and in the latter it becomes 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) 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. For a more detailed account on the mechanisms that underlie tableau systems, check, for example, document [7]; but also, [6], which specialises 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 Figure 2. Note that 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 7. 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,

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Figure 2: The tableau $\mathcal{T}^{\mathcal{H}I}$ (based on the tableau system for hybrid logic reported in [6]).

leads to

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

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.

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

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 $(\Delta, \Sigma) \in |Sign^{\mathcal{H}I}|$, 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 $\psi_1, \psi_2 \in Sen^I(\Sigma)$

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

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

Regarding rule $(\neg\downarrow)$,

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

Document [6] shows the proof the for remaining rules. To finish the proof, 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 $\psi \in Sen^I(\Sigma)$.

□

Next, we show that $\mathcal{T}^{\mathcal{H}I}$ is complete. Being based on the tableau system for hybrid logic reported in [6], $\mathcal{T}^{\mathcal{H}I}$ has a completeness proof similar to the one for the former, though simpler as termination is not addressed in this paper.

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 $(\Delta, \Sigma) \in |Sign^{\mathcal{H}I}|$, 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 $(W, R, m) \in |Mod^{\mathcal{H}I}(\Delta, \Sigma)|$, defined as

- $W = (N \,/\, \sim)$ where N denotes the set of nominals that occur in the branch, and $\sim$ the equivalence relation generated by the sentences in the branch of the type $@_i j$ (Rules (R) and (N1) guarantee that $\sim$ is an equivalence relation).
- for any $n \in Nom$, $M_n = [n]$, where $[n]$ denotes the class representative 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 $\varphi \in Sen^{\mathcal{H}I}(\Delta, \Sigma)$. Such 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 $\sim$ }

$i \not\sim j$

$\Rightarrow$ { 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 $\neg\langle\lambda\rangle$ lead to }

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. □

*4.3. An illustration in $\mathcal{H}$*ALLOY *– the reconfigurable buffers*

Increasingly popular both in industry and academia, ALLOY [17] is a lightweight model finder for software design whose language is 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 [18] – Neves *et al* [19, 12] introduced an institution for ALLOY along with suitable translations to (variants of) first–order and second–order logics. This makes possible to hybridise ALLOY but also

to verify the corresponding specifications in powerful provers such as SPASS [20] and LEO-II [21].

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 [4, 22] 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, when it is possible to discern the current execution mode?* To answer such a 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

$$\texttt{head} : \texttt{List} \to \texttt{Elem}$$
$$\texttt{tail} : \texttt{List} \to \texttt{List}$$

where for each $\texttt{l} \in \texttt{List}$, its head and tail ($\texttt{l} \cdot \texttt{head}$, $\texttt{l} \cdot \texttt{tail}$) has 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.

$$\texttt{one l} : \texttt{List} \mid \texttt{l} \cdot \texttt{head} \subseteq \emptyset$$
$$\texttt{one l} : \texttt{List} \mid \texttt{l} \cdot \texttt{tail} \subseteq \emptyset$$
$$\texttt{one l} : \texttt{List} \mid \texttt{l} \cdot \texttt{head} \subseteq \emptyset \texttt{ and l} \cdot \texttt{tail} \subseteq \emptyset$$

At this stage, ALLOY can already provide several instances of a list. For example, Figure 3 denotes the lists: $\texttt{List0} = []$, $\texttt{List1} = [b]$, $\texttt{List2} = [a]$, $\texttt{List3} = [b, a]$ and $\texttt{List4} = [a, a, a, \dots]$ where $\texttt{Elem0} = a$ and $\texttt{Elem1} = b$.

The next step is to define the `pop` relation

$$\texttt{pop} : \texttt{List} \to \texttt{List}$$

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

$$\texttt{FIFO} \vee \texttt{LIFO}$$

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

$$@_{\texttt{FIFO}}$$

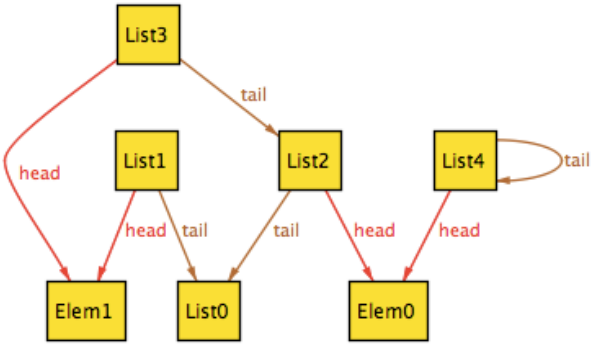


Figure 3: Several instances of a list in ALLOY

```
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
```

$@_{\texttt{LIFO}}$

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

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`; Figure 4 shows the behaviour of `pop` at `FIFO` with the lists mentioned above. It tells: $\texttt{pop}([]) = [], \texttt{pop} = ([b]) = [], \texttt{pop}([a]) = [], \texttt{pop}([b, a]) = [b]$ 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? Or formally, when

```
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

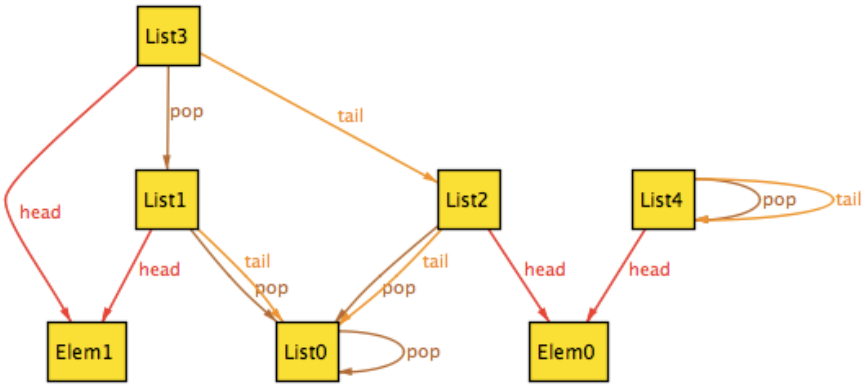


Figure 4: Examples of pop in action at state FIFO.

no more than four elements and fifty lists it is possible to distinguish the current execution mode with the test

$$\phi_2 \equiv \texttt{all l : List | } \neg \texttt{ l = empty} \rightarrow \texttt{l} \cdot \texttt{pop} = \texttt{l} \cdot \texttt{tail}$$

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

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

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,

$@_0\mathtt{FIFO}$	$@_0\mathtt{LIFO}, \neg@_0\mathtt{LIFO}$	$(\neg\wedge)$
$@_{\mathtt{FIFO}}\psi_1, @_{\mathtt{FIFO}}(\phi_1 \wedge \phi_2)$	$\times$	(N1)
$@_{\mathtt{FIFO}}(\psi_1 \wedge \phi_1 \wedge \phi_2)$	$\times$	$(\wedge\downarrow)$
$\times$	$\times$	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, the 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.* [23, 24, 25, 26, 27]), but also what Goguen and Meseguer wrote in [28]

> *"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."*

and the manifesto [29] on combination of logics.

Serving as a framework for the specification of reconfigurable systems, the hybridisation method systematically adds the features of hybrid logics on top of whatever logic is found useful in specification – and is, therefore, a technique for combining logics *asymmetrically*. Its formal introduction, in document [2], was ensued by further developments such as equivalence and refinement notions in hybrid(ised) logics [30], definition of initial semantics in this context [31], and, in the verification side, by suitable translations to first–order logic [2, 4]. Recently, hybridisation was implemented [3] in the HETS platform [18] and illustrations of its potentialities provided in [32]. However, contrary to what happened, for example with *temporalisation* [24] and *probabilisation* [25], proof theory for hybrid(ised) logics was yet to be explored.

Document [5] 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 went one

step beyond by simplifying the previous method, and providing a similar process that generates tableau systems instead. Such developments form a sound basis for a complete proof theory for hybrid(ised) logics, which, from a paradigmatic point of view, paves the way for 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 shall ensue.

The completeness results that this paper reports had always present the assumption that the base institution has the explicit satisfaction property. Although prevalent in logics used in software specification, such 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 (*cf.* [33]) of a logic soundness and completeness of the corresponding calculi, as well as decidability, can be obtained.

On a different note, other results in the literature abstract the combination of logics pattern by considering the "top logic" itself arbitrary. Such is the case of what is called *parametrisation* of logics in [26] by C. Caleiro, A. Sernadas and C. Sernadas. Similarly, the recent method of *importing* logics suggested by J. Rasga, A. Sernadas and C. Sernadas [27] aims at formalising this kind of asymmetric combinations resorting to a graph-theoretic approach. In both cases some decidability and completeness results are given. It should be interesting to see in which ways the hybridisation method relates to these approaches.

Acknowledgements. We would like to acknowledge Torben Bräuner due to helpful discussions between him and the authors about the matter of this paper.

This work is funded by ERDF - European Regional Development Fund, through the COMPETE Programme, and by National Funds through Fundação para a Ciência e Tecnologia (FCT) within project FCOMP-01-0124-FEDER-028923. Also, by project NORTE-07-0124-FEDER-000060, financed

by the North Portugal Regional Operational Programme (ON.2 - O Novo Norte), under the National Strategic Reference Framework (NSRF), through the European Regional Development Fund (ERDF), and by National Funds through FCT. Moreover, the first author is sponsored by FCT grant SFRH/BD/52234/2013. Finally, M. Martins is also supported by FCT project UID/MAT/04106/2013 at CIDMA and the EU FP7 Marie Curie PIRSES-GA-2012-318986 project GeTFun: Generalizing Truth-Functionality.

[1] 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), Vol. 7041 of Lecture Notes in Computer Science, Springer, 2011, pp. 269–285.

[2] 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), Vol. 6859 of Lecture Notes in Computer Science, Springer, 2011, pp. 283–297.

[3] R. Neves, A. Madeira, M. A. Martins, L. S. Barbosa, Hybridisation at work, in: CALCO TOOLS, Vol. 8089 of Lecture Notes in Computer Science, Springer, 2013.

[4] R. Diaconescu, A. Madeira, Encoding hybridized institutions into first-order logic, Mathematical Structures in Computer Science FirstView (2015) 1–44. doi:10.1017/S0960129514000383.
URL http://journals.cambridge.org/article_S0960129514000383

[5] 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: Brazilian Symp. on Formal Methods, SBMF 2014. Proceedings, Vol. 8941 of Lecture Notes in Computer Science, 2015, pp. 146–161.

[6] T. Braüner, Proof-Theory of Propositional Hybrid Logic, Hybrid Logic and its Proof-Theory, 2011.

[7] 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. doi:10.1007/978-94-017-1754-0_6.
URL http://dx.doi.org/10.1007/978-94-017-1754-0_6

[8] J. A. Goguen, R. M. Burstall, Institutions: abstract model theory for specification and programming, J. ACM 39 (1992) 95–146.

[9] T. Mossakowski, A. Haxthausen, D. Sannella, A. Tarlecki, CASL: The common algebraic specification language: Semantics and proof theory, Computing and Informatics 22 (2003) 285–321.

[10] R. Diaconescu, Institutional semantics for many-valued logics, Fuzzy Sets Syst. 218 (2013) 32–52. doi:10.1016/j.fss.2012.11.015.
URL http://dx.doi.org/10.1016/j.fss.2012.11.015

[11] 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, Vol. 521 of Lecture Notes in Computer Science, Springer Berlin Heidelberg, 1991, pp. 269–278. doi:10.1007/BFb0028112.
URL http://dx.doi.org/10.1007/BFb0028112

[12] R. Neves, A. Madeira, M. Martins, L. Barbosa, An institution for alloy and its translation to second-order logic, in: Integration of Reusable Systems, Springer, 2014, pp. 45–75.

[13] J. Fiadeiro, A. Sernadas, Structuring theories on consequence, in: D. Sannella, A. Tarlecki (Eds.), Recent Trends in Data Type Specification, Vol. 332 of Lecture Notes in Computer Science, Springer Berlin Heidelberg, 1988, pp. 44–72. doi:10.1007/3-540-50325-0-3.
URL http://dx.doi.org/10.1007/3-540-50325-0-3

[14] R. Diaconescu, Institution-independent Model Theory, Birkhäuser Basel, 2008.

[15] R. Diaconescu, Quasi-boolean encodings and conditionals in algebraic specification, J. Log. Algebr. Program. 79 (2) (2010) 174–188.

[16] G. Hoffmann, C. Areces, Htab: a terminating tableaux system for hybrid logic, Electr. Notes Theor. Comput. Sci. 231 (2009) 3–19.

[17] D. Jackson, Software Abstractions: Logic, Language, and Analysis, The MIT Press, 2006.

[18] 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), Vol. 4424 of Lecture Notes in Computer Science, Springer, 2007, pp. 519–522.

[19] 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 intergration, IEEE Press, 2013, pp. 512–519.

[20] C. Weidenbach, D. Dimova, A. Fietzke, R. Kumar, M. Suda, P. Wischnewski, SPASS version 3.5, in: R. A. Schmidt (Ed.), Proceedings of the 22nd International Conference on Automated Deduction, CADE 2009, Vol. 5663 of Lecture Notes in Artificial Intelligence, Springer, 2009, pp. 140–145.

[21] 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, 4th International Joint Conference, IJCAR 2008, Sydney, Australia, August 12-15, 2008, Proceedings, Vol. 5195 of LNCS, Springer, 2008, pp. 162–170.
URL `www.ags.uni-sb.de/ chris/papers/C26.pdf`

[22] A. Madeira, R. Neves, M. A. Martins, L. S. Barbosa, When even the interface evolves..., in: H. Wang, R. Banach (Eds.), Proceedings of 7th Internl Symp. on Theoretical Aspects of Software Engineering (TASE 2013), IEEE Press, 2013, pp. 79–82.

[23] R. Diaconescu, P. Stefaneas, Ultraproducts and possible worlds semantics in institutions, Theor. Comput. Sci. 379 (1-2) (2007) 210–230. doi:10.1016/j.tcs.2007.02.068.
URL `http://dx.doi.org/10.1016/j.tcs.2007.02.068`

[24] M. Finger, D. Gabbay, Adding a temporal dimension to a logic system, Journal of Logic, Language and Information 1 (3) (1992) 203–233. doi:10.1007/BF00156915.
URL http://dx.doi.org/10.1007/BF00156915

[25] P. Baltazar, Probabilization of logics: Completeness and decidability, Logica Universalis 7 (4) (2013) 403–440. doi:10.1007/s11787-013-0087-8.
URL http://dx.doi.org/10.1007/s11787-013-0087-8

[26] C. Caleiro, C. Sernadas, A. Sernadas, Parameterisation of logics, in: WADT, 1998, pp. 48–62.

[27] J. Rasga, A. Sernadas, C. Sernadas, Importing logics: Soundness and completeness preservation, Studia Logica 101 (1) (2013) 117–155. doi:10.1007/s11225-011-9363-x.
URL http://dx.doi.org/10.1007/s11225-011-9363-x

[28] J. A. Goguen, J. Meseguer, Models and equality for logical programming, in: H. Ehrig, R. Kowalski, G. Levi, U. Montanari (Eds.), TAPSOFT '87, Vol. 250 of Lecture Notes in Computer Science, Springer Berlin Heidelberg, 1987, pp. 1–22. doi:10.1007/BFb0014969.
URL http://dx.doi.org/10.1007/BFb0014969

[29] P. Blackburn, M. de Rijke, Why combine logics?, Studia Logica 59 (1) (1997) 5–27. doi:10.1023/A:1004991115882.

[30] A. Madeira, M. Martins, L. Barbosa, R. Hennicker, Refinement in hybridised institutions, Formal Aspects of Computing 27 (2) (2015) 375–395. doi:10.1007/s00165-014-0327-6.
URL http://dx.doi.org/10.1007/s00165-014-0327-6

[31] R. Diaconescu, Quasi-varieties and initial semantics for hybridized institutions, Journal of Logic and Computation.doi:10.1093/logcom/ext016.

[32] A. Madeira, R. Neves, M. A. Martins, L. S. Barbosa, Hybridisation for the working software engineer: a formal approach to reconfigurable systems, submitted to a journal.

[33] A. Madeira, R. Neves, M. Martins, L. Barbosa, Introducing hierarchical hybrid logic, in: Advances in Modal Logic 2014, 2014, pp. 74 – 78.