Logica Universalis

Check for
updates

Introducing *H*, an Institution-Based Formal Specification and Verification Language

Răzvan Diaconescu

Abstract. This is a short survey on the development of the formal specification and verification language *H* with emphasis on the scientific part. *H* is a modern highly expressive language solidly based upon advanced mathematical theories such as the internalisation of *Kripke semantics* within *institution theory*.

Mathematics Subject Classification. Primary 03C95; Secondary 03C98, 68N30.

Keywords. Institution theory, modal logic, formal specification.

1. Introduction

H is a language for formal specification and verification that has emerged out of a theoretical effort spread over a period of more than 25 years. It has been provisionally implemented as a prototype running system during 2017–2018 [8] and following this implementation a number of succesful case studies have been reported [8,40]. *H* is designed as a two-component system:

- *Hspec*–the specification language. This is an institution-based language in the sense that is parameterised over a variety of base logic systems captured as institutions (in the sense of the *institution theory* of Goguen and Burstall [21]). The role of the base logics refer to the specification of the data part of the system. For the dynamics part, *Hspec* employs the essentials of modal logic. This design is based upon the understanding that the essential ingredients of modal logic (both at the syntactic and at the semantic level) are independent of the base level logic. The fact that *Hspec* is parameterised by base logics gives it unparalleled specification expressivity and power since the most appropriate logic for the data part may be chosen. Moreover the list of base logics is open, any new logic may be added when convenient.
- *Hver*–a collection of verification tools and methods for specifications developed with *Hspec*. At this moment *Hver* contains only one such method

which is based on translations to first order logic. However in the future we plan to extend *Hver* with other methods and tools.

A web page of *H* is currently maintained at http://imar.ro/~diacon/forver/forver.html.

The *H concept*, the core of *H* represented by its scientific foundation, is based on developing an abstract Kripke semantics within the institution theory of Goguen and Burstall [21]. This is the core of *H*, from which the vision of *H* had emerged, its development being the result of a sustained mathematical effort reported in a series of papers from which the most representative are [14,19,20]. In this way *H* may be a typical example of a formal method that has emerged out of initially purely theoretically motivated work.

In this paper we survey the development of *H*, with emphasis on its most important aspects. This goes as follows:

1. In the first part, which is also the main part of the survey, we present the mathematical foundations of *H* and its basic design. As mentioned above this consists essentially of the internalisation of Kripke semantics in abstract institutions. This concept has been developed very gradually over many years and in our paper we will survey the most important moments of this development. We believe that presenting the *H* concept in this way has several benefits for the reader, including the possibility to understand the flow of ideas behind the *H* concept. It may be quite difficult if we chose to present only the end result of this rather complex process.

2. The next section is dedicated to the current implementation of *H*. This is only a temporary implementation that is based on the Hets system [28]. In long term, our vision for *H* is to have an independent implementation.

3. In the last section we discuss very briefly some case studies that have been formally specified and verified with *H*.

The readership must be familiar with some very basic category theory concepts, which are now quite commonly used in some areas of formal methods. Some familiarity with institution theory and modal logic may be also quite helpful.

On the practical implications of Universal Logic Universal Logic is renown as a general approach to logic and due to its generality and higher abstraction level not many fully understand its applicative power. However, people working in institution theory—a model theoretic branch of Universal Logic—know well about the crucial role played by institution theory in computing and programming, especially in the area of formal specification. For example, it has been a very effective tool for designing new logic-based specification languages and for developing the theory of specification uniformly at a general level. In this respect the case of *H* is quite special. Unlike any other language that has benefited from institution theory, *H embedds directly* abstract institutions in its definition and this reflects also at the methodological level. The unique specification and verification power of *H* is a consequence of this explicit use of abstract institutions.

2. The H Concept

The broader scientific context of the H concept is the theory of institutions of Goguen and Burstall [21]. The narrower context is the development of Kripke semantics within abstract institutions [14, 19, 20]. In this section we first give a very brief presentation of institution theory, and then survey the development process of the institution-theoretic Kripke semantics.

2.1. Institutions

The model theory oriented axiomatic approach by Goguen and Burstall to the notion of a logical system [21] that is based on the notion of *institution* has started a line of important developments of adequately abstract and general approaches to the foundations of software specifications and formal system development (see [35]) as well as a modern version of very abstract model theory (see [11]). One of the main original motivations for introducing institution theory was to respond to the explosion in the population of logics in use in computing almost four decades ago, a situation that continues today perhaps at an accelerated pace. These days the concept of institution lies at the foundations of several formal modern specification languages and environments such as Maude [7], CASL [3] or CafeOBJ [17], Hets [28] etc. In the area of formal specification and verification the contribution of the institution-theoretic approach to modularity and heterogeneity are priceless. Let us recall the notorious concept of institution:
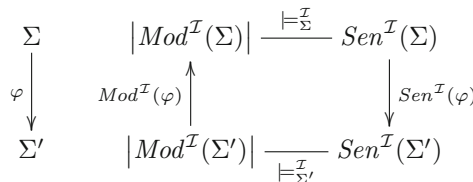

An *institution* $\mathcal{I} = \left(Sign^{\mathcal{I}}, Sen^{\mathcal{I}}, Mod^{\mathcal{I}}, \models^{\mathcal{I}} \right)$ consists of
- a category $Sign^{\mathcal{I}}$ whose objects are called *signatures*,
- a sentence functor $Sen^{\mathcal{I}} : Sign^{\mathcal{I}} \rightarrow \mathbf{Set}$ defining for each signature a set whose elements are called *sentences* over that signature and defining for each signature morphism a *sentence translation* function,
- a model functor $Mod^{\mathcal{I}} : (Sign^{\mathcal{I}})^{\mathrm{op}} \rightarrow \mathbf{CAT}$ defining for each signature Σ the category $Mod^{\mathcal{I}}(\Sigma)$ of Σ-*models* and Σ-model homomorphisms, and for each signature morphism φ the *reduct* functor $Mod^{\mathcal{I}}(\varphi)$,
- for every signature Σ, a binary Σ-*satisfaction relation* $\models_{\Sigma}^{\mathcal{I}} \subseteq |Mod^{\mathcal{I}}(\Sigma)| \times Sen^{\mathcal{I}}(\Sigma)$,

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

$$M' \models_{\Sigma'}^{\mathcal{I}} Sen^{\mathcal{I}}(\varphi)(\rho) \text{ if and only if } Mod^{\mathcal{I}}(\varphi)(M') \models_{\Sigma}^{\mathcal{I}} \rho \qquad (1)$$

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

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

The literature (e.g. [11,35]) shows myriads of logical systems from computing or from mathematical logic captured as institutions. In fact, an informal thesis underlying institution theory is that any 'logic' may be captured by the above definition. While this should be taken with a grain of salt, it certainly applies to any logical system based on satisfaction between sentences and models of any kind. In concrete institutions, typically the signatures are structured collections of symbols, the sentences are inductively defined from atoms by using sentence building operators, the sentence translations (along signature morphisms) rename symbols, the models interpret the symbols of the signatures as sets and functions, the reducts "forget" interpretations of some symbols, and the satisfaction is defined inductively on the structure of the sentences in Tarski's style [39].

Here we refrain from presenting examples of logical systems captured as institutions since the institution theory literature abounds of such examples. Instead let us just point out that the process of defining particular logical systems as institutions is not necessarily a trivial one since one may have to reconsider and give a serious fresh thought to concepts such as signature morphisms, variables, quantifiers, etc. This rethinking of various concepts may have to do very much with the intended applications, such as formal specification. For example, from the specification perspective the concept of signature morphism has to be much more general than what is usually employed in conventional logic, in order for the mathematics to work the variables require a kind of qualifications that are inspired from the practice of specification languages, etc. Some of these issues have been discussed in extenso in [13].

2.2. Kripke Semantics in Institutions

The semantics for modal logics, known as *Kripke semantics* was introduced in [24]. The origin of the development of Kripke semantics in institutions—often refereed to as 'modalization of institutions'—lies in some research undertaken within the group of the late Professor Joseph Goguen at Oxford in the early nineties regarding institutions for modal logics. First there was the realisation of the fact that the model amalgamation properties in modal logic institutions are a direct consequence of the respective properties in the base logics, such as propositional or first order logic. From there it followed the idea that each modal logic institution has an underlying simpler base institution and that the Kripke models may be defined uniformly on the basis of the models in a base institution. However it took over a decade to see the first paper on this published [20], mainly due to a rather complicated refereeing process.

In [20]—which may be considered the seminal paper for the H concept—we have introduced the first version of Kripke semantics in abstract institutions first by considering a "base" institution $\mathcal{I}$ and then by building a "modal" institution $\mathcal{HI}$ on top of $\mathcal{I}$. This construction has several components:

1. An extension of the syntax of $\mathcal{I}$. While the signatures stay the same, new sentences are built from the sentences of $\mathcal{I}$ by iteration of sentences building operators such as the usual Boolean operators, quantifiers, and modalities.

2. Kripke models built from the models of $\mathcal{I}$.
3. The definition of a modal satisfaction relation between the Kripke models and the new sentences.

Now let us review these three components of the construction of $\mathcal{HI}$ one by one.

2.2.1. The Syntax of $\mathcal{HI}$. For any signature Σ, the set $Sen^{\mathcal{HI}}(\Sigma)$ of the Σ-sentences of the "modal" institution $\mathcal{HI}$ is the least set closed under the following operations:

- $Sen^{\mathcal{I}}(\Sigma) \subseteq Sen^{\mathcal{HI}}(\Sigma)$;
- $\rho \star \rho' \in Sen^{\mathcal{HI}}(\Sigma)$ for any $\rho, \rho' \in Sen^{\mathcal{HI}}(\Sigma)$ and any $\star \in \{\vee, \wedge, \Rightarrow\}$,
- $\neg\rho \in Sen^{\mathcal{HI}}(\Sigma)$, for any $\rho \in Sen^{\mathcal{HI}}(\Sigma)$,
- $[\lambda](\rho_1, \ldots, \rho_n), \langle\lambda\rangle(\rho_1, \ldots, \rho_n) \in Sen^{\mathcal{HI}}(\Sigma)$,
  for any $\lambda \in \Lambda_{n+1}, \rho_i \in Sen^{\mathcal{HI}}(\Sigma)$, $i \in \{1, \ldots, n\}$;
- $(\forall\chi)\rho, (\exists\chi)\rho \in Sen^{\mathcal{HI}}(\Sigma)$, for any $\rho \in Sen^{\mathcal{HI}}(\Sigma')$ and $\chi: \Sigma \to \Sigma' \in \mathcal{D}$;

Some explanations are necessary:

- The first condition says that each sentence of the base institution becomes automatically a sentence of the "modal" institution.
- The second and the third conditions do the Boolean connectors on the sentences of the "modal" institution. Note that the sentences of the base institution may also involve Boolean connectors, in this case it is important to distinguish between the Boolean connectors at the base level and at the modal level since in general their effects may be differ.
- The fourth condition introduces modalities as sentences building operators. Here Λ_n means the set of modalities of arity n, which may be thought just as relation symbols. For now the modalities are *not* considered part of the signatures, they are rather fixed.
- Quantifiers are considered in the institution-theoretic manner, via designated signature morphisms (see for example [11] for details). Conventional concrete quantifiers would correspond to those signature morphisms that are in fact extensions of signatures with variables. So, not each signature morphism may be used in quantifiers, those that are designated for such use form a so-called *quantification space* which is the $\mathcal{D}$ from the last condition above. This concept represents an axiomatic approach to quantifiers that considers coherence properties with respect to translations along signature morphisms; it has been defined first in [12] and given this name in [14].
- Like with the Boolean connectors we have to carefully distinguish between quantifiers the level of $\mathcal{HI}$ and those that come with the sentences of the base institution as their effects may differ.
- The general institution-theoretic feature of the quantifiers, namely that they support higher-order quantification (up to what the concrete concept of signature supports) applies also here. So, depending on how we chose $\mathcal{D}$ we may have first order, or second order, or even higher order quantifiers.

2.2.2. Kripke Models. The models of $\mathcal{HI}$ are Kripke models defined on the basis of the models of the base institution $\mathcal{I}$:

Given a signature Σ, a *Kripke model* (M, W) consists of
- a set $|W|$ – called the set of the "possible worlds";
- for each $\lambda \in \Lambda_n$, a relation $W_\lambda \subseteq |W|^n$; and
- a mapping $M : |W| \to |Mod^{\mathcal{I}}(\Sigma)|$.

So, for each $w \in W$, M_w is a model of the base institution $\mathcal{I}$. Moreover, $W = (|W|, (W_\lambda)_{\lambda \in \Lambda})$ is called the *Kripke frame* of (M, W).

However in order for the quantifications to work properly, usually the models M_w have to share something. For example in the concrete case of first order modal logic it is quite common to require that the first order logic models M_w that are part of a Kripke model share their underlying sets and the interpretations of the variables. At the level of the abstract institutions this condition has been expressed in a general way in [20] as

---

$\beta_\Sigma(M_w) = \beta_\Sigma(M_v)$ for all $w, v \in |W|$.

---

where $\beta_\Sigma : Mod(\Sigma) \to Dom(\Sigma)$ is a functor satisfying some rather mild technical conditions (we omit them here).[1]

2.2.3. The "modal" Satisfaction. The satisfaction relation that relate the syntax of the "modal institution" $\mathcal{HI}$ to its semantics is defined by following the usual institution theoretic definitions and in Tarski's style by recursion on the structure of the sentences, the recursion base being the satisfaction in the base institution $\mathcal{I}$. For each Kripke model (M, W) and each $w \in |W|$ we define a "local" satisfaction relation as follows:

- $(M, W) \models^w \rho$ iff $M_w \models^{\mathcal{I}} \rho$; when $\rho \in Sen^{\mathcal{I}}(\Sigma)$,
- $(M, W) \models^w \rho \wedge \rho'$ iff $(M, W) \models^w \rho$ and $(M, W) \models^w \rho'$, and similarly for the other Boolean connectors;
- $(M, W) \models^w [\lambda](\xi_1, \ldots, \xi_n)$ iff for any $(w, w_1, \ldots, w_n) \in W_\lambda$ we have that $(M, W) \models^{w_i} \rho_i$ for some $1 \le i \le n$.
- $(M, W) \models^w \langle\lambda\rangle(\xi_1, \ldots, \xi_n)$ iff there exists $(w, w_1, \ldots, w_n) \in W_\lambda$ such that and $(M, W) \models^{w_i} \xi_i$ for any $1 \le i \le n$.
- $(M, W) \models^w (\forall \chi)\rho$ iff $(M', W') \models^w \rho$ for any (M', W') such that $Mod(\chi)(M', W') = (M, W)$,
- $(M, W) \models^w (\exists \chi)\rho$ iff $(M', W') \models^w \rho$ for some (M', W') such that $Mod(\chi)(M', W') = (M, W)$.

Under these definition in [20] it has been proved that $\mathcal{HI}$ is an institution where the satisfaction $(M, W) \models \rho$ is defined on the basis of the "local" satisfaction

---

[1]In examples $Dom : Sign^{\mathrm{op}} \to \mathbf{CAT}$ is always a functor that is "weaker" than Mod in the sense that it interprets less structure.

by $(M, W) \models^w \rho$ for all $w \in |W|$. Moreover, the adequacy of this construction has been tested against some deep model theoretic results including a very general "modal" ultraproducts theorem and its compactness consequences. Although in [20] we have not used multi-modalities (i.e. the relations from Λ) but instead used the more familiar $\Box$ and $\Diamond$, this difference is insignificant, being just a matter of form.

Note that $\mathcal{HI}$ in fact represents a class of institutions rather than a single institution because of the several parameters involved in its construction. Besides the base institution $\mathcal{I}$ of course, we also have the modalities Λ, the quantification space $\mathcal{D}$ and the sharing functor β. From this perspective a notation such as $\mathcal{HI}(\Lambda, \mathcal{D}, \beta)$ appears as more appropriate, however this is rather heavy so we usually stick to the simpler version when the involved parameters are clear.

The usual modal logic institutions arise immediately as examples of $\mathcal{HI}$. For instance modal propositional logic arises when considering propositional logic as base institution (eventually stripped of the Boolean connectors) and with $\mathcal{D}$ being trivial, while first order modal logic arises when considering atomic first order logic as the base institution (i.e. first order logic stripped off the Boolean connectors and off the quantifiers) and $\mathcal{D}$ consisting of the extensions of the signatures with appropriate variables. However the potential of the construction of $\mathcal{HI}$ goes much beyond that of known examples of modal logics because it frees modal logic from its conventional base. For example, at the base level it is possible to have partial functions with various kinds of sharing (an interesting one from an [19] would consider the sharing only of the definition domains of the partial functions). A more intriguing example is given by the possibility to iterate this construction for a number of times, obtaining hierarchical modal logics.

The construction of the "modal institution" of [20] is quite emblematic for all other developments in the area and constitutes the very basis for *Hspec* as both the syntax and the semantics of *Hspec* are based on this construction, but subject to some further additions that will be presented below.

2.3. Adding Nominals

An important development in the area of institution-theoretic Kripke semantics is the extension of the theory of [20] with the ingredients of the so-called "hybrid logic". Hybrid logics [4] are a brand of modal logics that provides appropriate syntax for the Kripke semantics in a simple and very natural way through the so-called *nominals*. Historically, hybrid logic was introduced in [33] and further developed in works such as [2,6,32] etc. The name "hybrid logics" was coined by Blackburn, but we consider this an uninspired choice leading to confusions because of at least two reasons. On the one hand this name does not suggest in any way the reality, namely that "hybrid logics" is a sub-brand of the modal logics. In fact the difference between the two is rather minor because technically it boils down only to a simple syntactic addition, whilst they share the same semantics. On the other hand the term "hybrid" has a clear meaning in ordinary language, which is difficult to relate to the

corresponding brand of logics. In spite of all these considerations, the terminology "hybrid logics" is already established in the literature, and even the name H owes to it.

The presence of nominals brings in several advantages from the point of view of formal specification and verification, such as the possibility of explicit reference to specific states of the model and a better more uniform proof theory. All these specification benefits have called for an extension of the original theory of [20], a first attempt in this direction being [26]. That had been technically a rather straightforward enterprise, which is briefly presented below.

2.3.1. Upgrading the Signatures. At the level of the signatures of $\mathcal{HI}$ we add the nominals, so after this addition a signature consists of a pair (Nom, Σ) where Nom is a set (of nominal symbols) and Σ is a signature of the base institution $\mathcal{I}$. This had been a good moment to include also the modalities Λ in the signatures, a move that is specification oriented. When specifying dynamics of systems it is necessary to have user defined modalities. Therefore a signature in $\mathcal{HI}$ is now a triple $(\mathrm{Nom}, \Lambda, \Sigma)$.

2.3.2. Upgrading the Sentences. The collection of the sentence building operators gets expanded with:

- Nom $\subseteq Sen^{\mathcal{HI}}(\mathrm{Nom}, \Lambda, \Sigma)$;
- $@_i\rho \in Sen^{\mathcal{HI}}(\mathrm{Nom}, \Lambda, \Sigma)$ for any $\rho \in Sen^{\mathcal{HI}}(\mathrm{Nom}, \Lambda, \Sigma)$ and $i \in \mathrm{Nom}$;

Then there is the issue of upgrading the quantification by allowing quantifications over the nominals. For this we have to consider $\mathcal{D}$ as a quantification space for the upgraded signatures, but one which does not have any effect on the modalities. Thus the quantification building operators get upgraded to:

- $(\forall\chi)\rho, (\exists\chi)\rho \in Sen^{\mathcal{HI}}(\mathrm{Nom}, \Lambda, \Sigma)$, for any $\rho \in Sen^{\mathcal{HI}}(\mathrm{Nom}', \Lambda, \Sigma')$ and $\chi : (\mathrm{Nom}, \Lambda, \Sigma) \to (\mathrm{Nom}', \Lambda, \Sigma') \in \mathcal{D}$;

2.3.3. Upgrading the Semantics. The upgrade of the concept of Kripke models is very simple, just interpret the new syntactic entities by extending W with interpretations for the nominals. So for each $i \in \mathrm{Nom}$ we have a designated element $W_i \in |W|$.

2.3.4. Upgrading the Satisfaction Relation. This upgrade adds the semantics of the new building operators as follows:

- $(M, W) \models^w i$ iff $W_i = w$; when $i \in \mathrm{Nom}$,
- $(M, W) \models^w @_j\rho$ iff $(M, W) \models^{W_j} \rho$.

We can see that the upgrade of the construction of $\mathcal{HI}$ from [20] in the direction of nominals is technically very straightforward. This is one of the reasons the paper [26] may be considered as only a minor contribution to the general development of the H concept. But there are other more serious reasons for

this evaluation. Due to being a conference paper—and therefore being quite rushed and suffering from severe space limitations—the authors had to scrap some crucial features of the original construction from [20], such as the sharing at the level of the Kripke models. One of the dramatic consequences of this simplification—called the "free hybridisation"—was that the quantification became nonfunctional, thus reducing a lot the specification power of the formalism. However these shortcomings have been corrected in the journal paper [14], which may be considered as the first paper addressing the extension of [20] with nominals in a proper way.

2.4. More General Constraints

In [14] an ultimate very general axiomatic approach to the *constraints* on Kripke models had been proposed. This approach captures a wide variety of constraints, such as various sharing constraints or constraints on the shape of the Kripke frames (such as reflexivity, transitivity, etc.). It is for instance more general and more accommodating than the sharing constraints defined in [20]. Let us recall from [14]:

A *constrained $\mathcal{HI}$-model functor* is a sub-functor $Mod^C \subseteq Mod^{\mathcal{HI}}$ such that it reflects weak amalgamation for the designated pushout squares corresponding to the quantification space $\mathcal{D}^{\mathcal{I}}$ at the level of the base institution $\mathcal{I}$ (that is obtained by "forgetting" the nominals part from $\mathcal{D}^{\mathcal{HI}}$). The models in Mod^C are called *constrained $\mathcal{HI}$-models*.

We omit here detailed explanations concerning the technical condition on weak amalgamation as the interested reader may consult [14] or [19]. Informally, the meaning of the reflection (of weak amalgamation) condition is that in the case of the designated pushout squares used in quantifications the amalgamation of constrained models yields a constrained model. The role of this condition, which is rather mild in the applications, is to ensure that the constrained models support smoothly the quantifications. At the end we get a 'sub-institution' of $\mathcal{HI}$ with constrained Kripke models that is denoted $\mathcal{HI}^C$.

2.5. *Hspec* and $\mathcal{HI}^C$

The definition of *Hspec* sticks closely to the construction of $\mathcal{HI}^C$, being just a realisation of this construction as a specification language. The following ideas underlie the definition of *Hspec*:

- The syntax of *Hspec* comes on two layers. The "upper" layer follows the definition of the signatures and of the sentences of $\mathcal{HI}^C$, which become *Hspec* declarations. The "lower" layer follows the definition of the signatures and of the sentences of the base institution $\mathcal{I}$, which is the most important parameter the respective specification. In principle there is almost absolute freedom about the "lower" layer, in practice however we have to commit to something concrete, usually to something that already exists in the realm of current specification languages. For example CASL [3] may be used in many situations because its underlying institution is a

rather complex one which includes many logical features, such as Boolean connectors, first order quantifications, partial functions, etc.

- The semantics of a *Hspec* specification is the class of the constrained $\mathcal{HI}^C$ models (Kripke models) that satisfy the axioms declared in the respective specification.
- In *Hspec*, currently the constraints on the Kripke models are specified in two ways. Either by "rigidity" declarations for the syntactic entities (sorts, operations, relations, etc.) that are meant to be interpreted uniformly across the base institution models in a Kripke model, or else by specific axioms in other cases (such as various properties of the Kripke frame, but not only). The constraint axioms do not appear in the specifications as they are part of the definition of all Kripke models and therefore are common to all specifications; they are declared when defining the respective logic/institution.

Let us see how these ideas are realised in the case of a concrete example of a *Hspec* specification. The following *Hspec* specification is that of a reconfigurable calculator for natural numbers with a binary operation that in one state is sum and in the other one is multiplication, an example which is discussed in [25].

```
spec Nat =
 logic : RigidCASL

 rigid sort Nat
 rigid op 0 : Nat
 rigid op suc : Nat −> Nat
 op X : Nat ∗ Nat −> Nat
end

spec Calc =
 logic :
  HRigidCASLC
 data :
  Nat
 configuration :
  nominals mult , sum
  modality shift : 2
  . mult \/ sum
  . @ sum : <shift> mult /\ [ shift ] mult
  . @ mult : <shift> sum /\ [ shift ] sum
  . @ mult : not sum
  . @ sum : forall m : Nat . X(m, 0) = m
  . @ sum : forall m, n : Nat . X(m, suc(n)) = suc (X(m, n))
  . @ mult :
     forallH m : Nat . forallH n : Nat
       . existsH x : Nat . existsH y : Nat
       . X(m, n) = x /\ X(m, suc(n)) = y /\ <shift> X(x, m) = y
end
```

The first specification, at the level of the base institution, declares the data of the natural numbers together with the binary operation that will change modes that can be either interpreted as addition or as multiplication. This uses the CASL logic (essentially first order logic with partial functions). Rigidity constraints are also specified at this stage in order to prepare for the Kripke models in which the base models share their underlying sets and some of the operations. The rigidity declarations do not have any semantic effect at the level of the data (`Nat`), however they will have an effect at the level of the hybridisation. This latter aspect, although does not introduce any error, is a little "unclean". It is not a kind of implementation shortcut, it rather comes from a small gap in the theory. Currently, when building a hybridisation $\mathcal{HI}^C$, the signatures of the base institution are preserved. Since rigidity of sorts and operations are in fact declarations at the level of the base institution signatures, they have to be there already in order to specify sharing constraints in the hybridisation. A possible solution to this is to go more abstract about the signatures of $\mathcal{HI}^C$ by specifying them abstractly together with a projection functor to the signatures of $\mathcal{I}$ (the base institution) that may be subject to some axioms, in the style of how frame and nominals extractions are defined in [15].

The second specification is at the level of a hybridisation, which in this case is `HRigidCASLC` (rigid sorts, rigid total functions and the domain of each rigid partial function are interpreted uniformly). Its definition does not appear in this specification as it resides in a library, being a predefined entity of *Hspec*. However in this particular example we do not use any partial functions. The data `Nat` is imported and nominals and the modality are declared. In the case of the modality note its arity 2. This is the part that declares the respective $\mathcal{HI}^C$ signature. Then follows a series of axioms mainly regarding the dynamics of the system. For example, the first axiom says that the Kripke frames have at most two elements. Note the two levels of quantifiers, `forall` is at the level of the base institution while `forallH` and `existsH` are quantifiers at the level of the hybridised institution. Because the base is a kind of fully fledged kind of first order logic, its hybridisation $\mathcal{HI}^C$ differs substantially from the classical first order hybrid logics.

Regarding the syntax of the *Hspec* specifications, the current parser of *Hspec* allows for two different styles of syntax:

1. One more faithful to the mathematical foundations, like in the example, and
2. another one that is more "engineering oriented".

For example in the latter one, the application of a modality `[m] e` is written `After m, always e`.

Now a short note on the structured specifications in *Hspec*. It is not necessary to discuss them in detail here as they just follow the institution-theoretic approach (e.g. [16,18,35]). The fundamental technical results supporting structured specifications in *Hspec* are the existence of pushouts of signature morphisms and model amalgamation in $\mathcal{HI}^C$, results that follow rather easily from the definitions.

2.6. Encoding into First-Order Logic

The current version of *Hver* contains only one method and tool that is based upon a mathematical result that constitutes the main achievement in [19]. That result represents an extension of the traditional translation of modal logic to first order logic [41] (for the hybrid variant [5]) to encodings of abstract hybridised institutions into first order logic.

That encoding uses the mathematical notion of *comorphism* [22,27,29, 37,38], which is an important concept of institution theory. From the perspective of the mathematical structure, comorphisms are just 'homomorphisms of institutions'. So they are mappings between institutions that preserve the mathematical structure of institutions.

An *institution comorphism* $(\Phi, \alpha, \beta) : \mathcal{I} \to \mathcal{I}'$ consists of

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

such that the following *satisfaction condition* holds

$$M' \models'_{\Phi(\Sigma)} \alpha_\Sigma(e) \ \text{ iff } \ \beta_\Sigma(M') \models_\Sigma e$$

for each signature $\Sigma \in |Sign|$, for each $\Phi(\Sigma)$-model M', and each Σ-sentence e.

While the α represents the translation of the syntax, the β represents the translation of the semantics. The *satisfaction condition* ensures the mutual coherence between these translations.

Although comorphisms generally express an embedding relationship between institutions, they can also be used for 'encoding' a 'more complex' institution $\mathcal{I}$ into a 'simpler' one $\mathcal{I}'$. In such encodings the structural complexity cost is shifted to the mapping Φ on the signatures, thus Φ maps signatures of $\mathcal{I}$ to *theories* of $\mathcal{I}'$ rather than signatures. This is why in the literature these are sometimes [22,29] called 'theoroidal' comorphisms. A *theory* in $\mathcal{I}$ is just a specification in $\mathcal{I}$, i.e. a signature Σ plus a set E of Σ-sentences. Technically speaking a 'theoroidal' comorphism is in fact an ordinary comorphism when we replace the institution $\mathcal{I}'$ with the *institution of its theories* $\mathcal{I}'^{\mathrm{th}}$. This is achieved through a general construction that can be applied to absolutely any institution, in which the signatures of $\mathcal{I}'^{\mathrm{th}}$ are the theories of $\mathcal{I}$. The details of this construction may be found in may places in the literature, such as in [11] (but under the name of the institution of 'presentations').

Due to the generality of the construction of $\mathcal{HI}^C$, including its parameters $\mathcal{D}$ and the constraint sub-functor on models, the definition of the general encoding of $\mathcal{HI}^C$ into first order logic is technically rather complex. Therefore we omit it here (for the details the interested reader has to refer to [19]) and instead we present briefly its main idea. The basis of the construction of the comorphism $\mathcal{HI}^C \to \mathcal{FOL}^{\mathrm{th}}$ (where $\mathcal{FOL}$ is the institution of first order

logic in its many sorted form) is a given encoding from the base institution to $\mathcal{FOL}$, i.e. a comorphism $\mathcal{I} \to \mathcal{FOL}^{\text{th}}$. This is considered abstractly, so it may vary, and in this way it constitutes the main parameter of this construction. In practice these comorphisms may be well established translations. Then under some technical conditions—mainly about quantifiers and model constraints— that are commonly satisfied in the applications, the comorphism $\mathcal{I} \to \mathcal{FOL}^{\text{th}}$ is "lifted" to a comorphism $\mathcal{HI}^C \to \mathcal{FOL}^{\text{th}}$.

2.6.1. How *Hver* Works in Principle. Suppose that we have a specification in *Hspec*, which corresponds to a theory (Δ, E) in some $\mathcal{HI}^C$. Suppose that we have a property e that we have to check; this means that we have to prove that $E \models_\Sigma e$ (which means that any model of E also satisfies e). In order to establish $E \models e$ we have to perform the following steps:

1. We translate E and e by using the comorphism $\mathcal{HI}^C \to \mathcal{FOL}^{\text{th}}$; this yields $\tilde{E}$ and $\tilde{e}$ in $\mathcal{FOL}$. Usually $\tilde{E}$ includes both the syntactic translation $\alpha(E)$ of E and the sentences of the theory $\Phi(\Delta)$. Obviously in the case of $\tilde{e}$ it is not necessary to include the latter sentences, so $\tilde{e} = \alpha(e)$.
2. If we had a theorem prover for $\mathcal{FOL}$ then this step would not be necessary. But unfortunately, at least up to my knowledge, all major first order logic theorem provers work with the *unsorted* version of first order logic. So we translate again both $\tilde{E}$ and $\tilde{e}$ along a well known comorphism that encodes many sorted first order logic into unsorted first order logic (the details of this comorphism may be found for example in [11]). We thus arrive at $\bar{E}$ and $\bar{e}$.
3. Now we employ a first order theorem prover and attempt to prove that $\bar{E} \models \bar{e}$.
4. If we are successful with the previous task then we may conclude that $E \models e$. However this move backwards is not straightforward. It holds as a consequence of an important property of comorphisms, namely that of *conservativity*:

> An institution comorphism $(\Phi, \alpha, \beta) : \mathcal{I} \to \mathcal{I}'$ is *conservative* when for each Σ-model M in $\mathcal{I}$, there exists a $\Phi(\Sigma)$-model M in $\mathcal{I}$ such that $M = \beta_\Sigma(M)$.

In [19] we have shown that under some technical conditions that are usually satisfied in the applications, the conservativity of the comorphism $\mathcal{HI}^C \to \mathcal{FOL}^{\text{th}}$ is inherited from the conservativity of the base comorphism $\mathcal{I} \to \mathcal{FOL}^{\text{th}}$. In order to complete the argument we still need that the encoding of $\mathcal{FOL}$ to unsorted first order logic is conservative, which indeed is.

In practice all the translations and the proofs are performed automatically using tools. We will see more about this later on.

3. The Current H Implementation

3.1. ForVer and Hets

In the year 2017 the author of this paper won funding for the project proposal *Formal Verification of Reconfigurable Systems* (acronym: *ForVer*) under a new funding scheme of a Romanian government agency for funding of research that was dedicated to experimental-demonstrative projects. That competition was highly competitive, with a succes rate of about 5%, and the reviewers of the project proposals were selected from the international scientific community. The goal of *ForVer* was to realise the long term vision of H and of the science behind it as a running prototype. Then the project hired Mihai Codescu for programming the prototype.[2]

The basic plan for this implementation of H was to rely on Hets [28]. Hets is a tool for heterogeneous multi-logic specification and modeling of software systems and ontology development. In both these fields, there are a large number of logics and languages in use, each better suited for a different task or providing a better support for a different aspect of a complex system. Instead of trying to integrate the features of all these logics into a single formalism, the paradigm of heterogenous multi-logic specification is to integrate all logics by the means of a so-called Grothendieck construction over a graph of logics and their translations (captured as institutions and institution comorphisms, respectively). Thus, for each logic we can make use of its dedicated syntax(es) and proof tools. The specifier has the freedom to choose the logic that suits best the problem to be solved, offers best tool support and according to the degree of familiarity with a certain specification language. Hets provides an implementation of this paradigm. Because of the multi-logic feature of H and also because of *Hver* is based logic encodings (translations), Hets appeared as suitable for a smooth implementation of a first prototype for H.

3.1.1. Grothendieck Institutions. As mentioned above the foundation of Hets are the so-called *Grothendieck institutions*, which represents the ultimate theoretical answer to the problem of heterogeneous multi-logic specification. Instead of presenting the rather intricate technicalities of this concept let us review how it was developed. This theory has been initially developed gradually within the context of the CafeOBJ [17], which was the first heterogeneous specification language. A first attempt to address this heterogeneity institution theoretically was in [9]. Then the late Professor Martin Hofmann, while writing a review for this publication, suggested a construction on institutions similar to the famous construction by Alexandre Grothendieck on categories [23] originating from algebraic geometry. In the year 2000 this suggestion was realised in [10], but that was based on the original concept of homomorphism of institutions, the *institution morphisms* of [21], which is somehow dual to the concept of institution comorphisms that was discussed above. A few years

---

[2] "Programming" here is meant is a broader sense that includes the design of the architecture of the implementation and the writing of the code.

later it was realised [30] that some crucial proporties of Grothendieck institutions may be obtained more smoothly by the same construction but based on institution comorphisms rather than institution morphisms. The Hets system is based on the later version of Grothendieck institutions.

3.2. The Hets Implementation of H

Note that this is an open implementation that supports further enhancements. Here we only review very briefly its main components, the details of the implementation being outside the scope of the paper. For the interested readers, a more detailed description of this implementation may be found in [8].

1. Syntactic support for the declarations of the parameters of the hybridisation process. The considered parameters of the hybridisation are:
   - The base institution. This is specified by using its internal Hets name, based on a Hets qualification mechanism it is possible to select also a sub-institution of an institution already implemented in Hets.
   - The quantifier symbols. These may be nominal symbols or classes of symbols that are specific to the quantifications in the base institutions (such as constants, rigid constants, total constants).
   - The constraints on the Kripke models. These are specified through a fixed grammar that cover two different kinds of constraints: on the Kripke frames and on the interpretations of symbols in the local models.
2. Generic method for generating new instances of the Hets class **Logic**.
3. Support for structured specifications. The core specification structuring operators of *Hspec* consist of unions and specification translations alongs signature morphisms (which are symbols renaming). Other structuring mechanisms, such as imports for example, are derived on the basis of these. These are supported on the basis of a correspondence between the structured *Hspec* specifications and DOL [31], the structuring language supported in Hets.
4. Support for *Hver*. There is a special declarative syntax for this that takes as parameters the base theoroidal comorphism (from the base institution $\mathcal{I}$ to $\mathcal{FOL}$) and the name of the respective hybridised logic $\mathcal{HI}^C$ and a generic method analyses these definitions and generates corresponding Haskell code. The compilation of this code makes the new encoding available for the verification process, where the translation $\bar{E} \models \bar{e}$ of a goal $E \models e$ is passed to one of the first order logic theorem provers of choice, such as SPASS [42], Vampire [34], E [36].

4. H at Work

A number of succesful case studies with H have been already reported. In this section we present very briefly a couple of them.

4.1. A Steam Boiler Control

The problem of this case study is a very notorious benchmark in formal methods [1]. The case study with H, developed by Mihai Codescu and Ionuţţ Ţuţu, has been reported only in the *ForVer* project documents. The H code is available at https://ontohub.org/forver/Sbcs.dol.

The H specification of the boiler control system illustrates almost all of features of H. The base institution $\mathcal{I}$ is many-sorted first order logic, the hybridised institution $\mathcal{HI}^C$ has quantifications over nominals and rigid constants, and the constraints are given by the sharing of the domains and of the interpretations of rigid symbols.

The properties that have been verified include changes of modes when an event takes place and that in all states of the system the expected functionality takes place. In the H formalisation, the system has five modes (nominals) and nine events (modalities).

4.2. A Bike-Sharing System Design

This case study has been reported in [40] and the H code is available at https://ontohub.org/forver/BSS.dol.

It is based on a double hybridisation (hybridisation iterated twice) the base level for the first level hybridisation being the (atomic fragment) of propositional logic.

- The first level hybridisation has quantifications over nominals and one constraint, namely that the interpretation of one of the modalities ('parent') is a forest (a set of disjoint trees).
- The second level of hybridisation admits quantifications over first level nominals (called "actors") as well as quantifications over second level nominals (called "configurations"). There is a sharing constraint: the first level Kripke models in a second level Kripke model share the same underlying set of "actors".

Since at the verification stage this modelling leads to some timeout problems (due to a huge number of sentences obtained by the encoding in $\mathcal{FOL}$), the first level of the hybridisation has been encoded in an institution of relations.

The first order theorem prover employed by this case study is SPASS.

5. H in the Future

There are several directions that we see with respect to the future evolution of H.

- When conditions allow there should be a new implementation of H that is independent of Hets. The reasons for this are manifold. For example Hets is a rather big system and H relates only to a small part of Hets. Such big systems are prone to errors that may easily affect the functionality of H. Moreover H maintainers have little control on the evolution of Hets.

- *Hver* should be enhanced with more tools and methods. For example a direct tool based on proof systems at the level of hybridised institutions (so *not* by translation) would be an welcome enhancement of *Hver*.
- Adding new base institutions and constraints to the current Hets implementation of H.
- Some slight upgrades of the foundations may be necessary in order to accomodate certain specification methodologies. For instance, we have already discussed the issue of rigidity declarations at the level of the base institutions which may be solved by considering 'projection' functors from the categories of the signatures of the hybridised institutions to the categories of the signatures of the base institution.
- More larger case studies should be developed with the aim to finally have H as an industrial tool.

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

References

[1] Abrial, J.-R., Börger, E., Langmaack, H. (eds.): Formal Methods for Industrial Applications – Specifying and Programming the Steam Boiler Control, volume 1165 of LNCS. Springer, New York (1996)

[2] Areces, C., Blackburn, P., Delany, S.R.: Bringing them all together. J. Log. Comput. **11**, 657–669 (2001)

[3] Astesiano, E., Bidoit, M., Kirchner, H., Krieg-Brückner, B., Mosses, P., Sannella, D., Tarlecki, A.: CASL: the common algebraic specification language. Theor. Comput. Sci. **286**(2), 153–196 (2002)

[4] Blackburn, P.: Representation, reasoning, and relational structures: a hybrid logic manifesto. Log. J. IGPL **8**(3), 339–365 (2000)

[5] Blackburn, P., Seligman, J.: Hybrid languages. J. Log. Lang. Inf. **4**(3), 251–272 (1995)

[6] Braüner, T.: Hybrid Logic and its Proof-Theory, Volume 37 of Applied Logic Series. Springer, New York (2011)

[7] Clavel, M., Durán, F., Eker, S., Lincoln, P., Martí-Oliet, N., Meseguer, J., Talcott, C.: All About Maude—A High-Performance Logical Framework. Lecture Notes in Computer Science, vol. 4350. Springer, New York (2007)

[8] Codescu, M.: Hybridisation of institutions in Hets. In: CALCO 2019, 8th Conference on Algebra and Coalgebra in Computer Science (2019)

[9] Diaconescu, R.: Extra theory morphisms for institutions: logical semantics for multi-paradigm languages. Appl. Categ. Struct., **6**(4), 427–453 (1998). A preliminary version appeared as JAIST Technical Report IS-RR-97-0032F in 1997

[10] Diaconescu, R.: Grothendieck institutions. Appl. Categ. Struct., **10**(4), 383–402 (2002). Preliminary version appeared as IMAR Preprint 2-2000, ISSN 250-3638, (February 2000)

[11] Diaconescu, R.: Institution-Independent Model Theory. Birkhäuser, Basel (2008)

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

[13] Diaconescu, R.: From universal logic to computer science, and back. In: Ciobanu, G., Méry, D. (ed.) Theoretical Aspects of Computing—ICTAC 2014, Volume 8687 of Lecture Notes in Computer Science. Springer, New York (2014)

[14] Diaconescu, R.: Quasi-varieties and initial semantics in hybridized institutions. J. Log. Comput. **26**(3), 855–891 (2016)

[15] Diaconescu, R.: Implicit Kripke semantics and ultraproducts in stratified institutions. J. Log. Comput. **27**(5), 1577–1606 (2017)

[16] Diaconescu, R., Ţuţu, I.: On the algebra of structured specifications. Theor. Comput. Sci. **412**(28), 3145–3174 (2011)

[17] Diaconescu, R., Futatsugi, K. : CafeOBJ Report: The Language, Proof Techniques, and Methodologies for Object-Oriented Algebraic Specification, Volume 6 of AMAST Series in Computing. World Scientific, Singapore (1998)

[18] Diaconescu, R., Goguen, J., Stefaneas, P.: Logical support for modularisation. In: Huet, G., Plotkin, G. (eds.) Logical Environments, Cambridge, 1993, pp. 83–130. Proceedings of a Workshop held in Edinburgh, Scotland (1991)

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

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

[21] Goguen, J., Burstall, R.: Institutions: abstract model theory for specification and programming. J. Assoc. Comput. Mach. **39**(1), 95–146 (1992)

[22] Goguen, J., Roşu, G.: Institution morphisms. Form. Asp. Comput. **13**, 274–307 (2002)

[23] Grothendieck, A.: Catégories fibrées et descente. In: Revêtements étales et groupe fondamental, Séminaire de Géométrie Algébraique du Bois-Marie 1960/61, Exposé VI. Institut des Hautes Études Scientifiques, 1963. Reprinted in Lecture Notes in Mathematics, Volume 224, pp. 145–94. Springer, New York (1971)

[24] Kripke, S.: A completeness theorem in modal logic. J. Symb. Log. **24**, 1–15 (1959)

[25] Madeira, A.: Foundations and techniques for software reconfigurability. PhD thesis, Universidades do Minho, Aveiro and Porto (Joint MAP-i Doctoral Programme) (2014)

[26] Martins, M.-A., Madeira, A., Diaconescu, R., Barbosa, L.: Hybridization of institutions. In: Corradini, A., Klin, B., Cîrstea, C. (eds.) Algebra and Coalgebra in Computer Science, Volume 6859 of Lecture Notes in Computer Science, pp. 283–297. Springer, New York (2011)

[27] Meseguer, J.: General logics. In: Ebbinghaus, H.-D., et al. (ed.) Proceedings, Logic Colloquium, 1987, pp. 275–329. North-Holland (1989)

[28] Mossakowski, T., Maeder, C., Lütich, K.: The heterogeneous tool set. Lect. Notes Comput. Sci. **4424**, 519–522 (2007)

[29] Mossakowski, T.: Different types of arrow between logical frameworks. In: Meyer auf der Heide, F., Monien, B. (eds.) Proceedings of ICALP 96, Volume 1099 of Lecture Notes in Computer Science, pp. 158–169. Springer, New York (1996)

[30] Mossakowski, Till: Comorphism-based Grothendieck logics. In K. Diks and W. Rytter, editors, Mathematical foundations of computer science, volume 2420 of Lecture Notes in Computer Science, pages 593–604. Springer, (2002)

[31] Mossakowski, T., Codescu, M., Neuhaus, F., Kutz, O.: The distributed ontology, modeling and specification language - dol. In: Buchsbaum, A., Koslow, A. (eds.) The Road to Universal Logic. Birkhauser, Cham (2015)

[32] Passy, S., Tinchev, T.: An essay in combinatory dynamic logic. Inf. Comput. **93**(2), 263–332 (1991)

[33] Prior, A.N.: Past, Present and Future. Oxford University Press, Oxford (1967)

[34] Riazonov, A., Voronkov, A.: The design and implementation of VAMPIRE. AI Commun. **15**(2–3), 91–110 (2002)

[35] Sannella, D., Tarlecki, A.: Foundations of Algebraic Specifications and Formal Software Development. Springer, New York (2012)

[36] Schulz, S.: System description: E 1.8. In: Proceedings of the 19th conference on Logic Programming and Autamated Reasoning (LPAR), Volume 8312 of LNCS, pp. 477–483 (2013)

[37] Tarlecki, A.: Moving between logical systems. In: Haveraaen, M., Owe, O., Dahl, O.-J. (eds.) Recent Trends in Data Type Specification, Volume 1130 of Lecture Notes in Computer Science, pp. 478–502. Springer, New York (1996)

[38] Tarlecki, A.: Towards heterogeneous specifications. In: Gabbay, D., van Rijke, M. (eds.) Proceedings, International Conference on Frontiers of Combining Systems (FroCoS'98), pp. 337–360. Research Studies Press (2000)

[39] Tarski, A.: The semantic conception of truth. Philos. Phenomenol. Res. **4**, 13–47 (1944)

[40] Ţuţu, I., Chiriţă, C.E., Lopes, A., Fiadeiro, J.L.: Logical support for bike-sharing system design. In: From Software Engineering to Formal Methods and Tools, and Back, Volume 11865 of Lecture Notes in Computer Science. Springer, New York (2019)

[41] van Bentham, J.: Modal Logic and Classical Logic. Humanities Press, New York (1988)

[42] Weidenbach, C., Dimova, D., Fietzke, A., Kumar, R., Suda, M, Wischnewski, P.: SPASS version 3.5. In: Automated Deduction, Volume 5663 of LNCS, pp. 140–145 (2009)

Răzvan Diaconescu
Simion Stoilow Institute of Mathematics of the Romanian Academy
Bucharest
Romania
e-mail: `Razvan.Diaconescu@ymail.com`