ALEXANDRE MADEIRA

FOUNDATIONS AND TECHNIQUES FOR SOFTWARE RECONFIGURABILITY
An institution-independent approach for specifying and reasoning about reconfigurable systems
FOUNDATIONS AND TECHNIQUES FOR SOFTWARE RECONFIGURABILITY
An institution-independent approach for specifying and reasoning about reconfigurable systems

ALEXANDRE MADEIRA

Departamento de Informática
Escola de Engenharia
Universidade do Minho
June 2013
Alexandre Madeira: Foundations and techniques for software reconfigurability. An institution-independent approach for specifying and reasoning about reconfigurable systems, © June 2013
ABSTRACT

The qualifier reconfigurable is used for software systems which behave differently in different modes of operation (often called configurations) and commute between them along their lifetime. Such systems, which evolve in response to external or internal stimulus, are everywhere: from e-Health or e-Government integrated services to sensor networks, from domestic appliances to complex systems distributed and collaborating over the web, from safety or mission-critical applications to massive parallel software.

There are two basic approaches to formally capture requirements of this sort of systems: one emphasizes behaviour and its evolution; the other focuses on data and their transformations. Within the first paradigm, reconfigurable systems are regarded as (some variant of) state-machines whose states correspond to the different configurations they may assume. On the other hand, in data-oriented approaches the system’s functionality is specified in terms of input-output relations modelling operations on data. A specification presents a theory in a suitable logic, expressed over a signature which captures its syntactic interface. Its semantics is a class of concrete algebras or relational structures, acting as models for the specified theory.

The observation that whatever services a reconfigurable system may offer, at each moment, may depend on the stage of its evolution, suggests that both dimensions (data and behaviour) are interconnected and should be combined. In particular, each node in the transition system which describes a reconfiguration space, may be endowed with a local structure modelling the functionality of the respective configuration. This is the basic insight of a configurations-as-local-models specification style. Technically, specifications become structured state-machines, states denoting complex structures, rather than sets.

A specification for this sort of system should be able to make assertions both about the transition dynamics and, locally, about each particular configuration. This leads to the adoption of hybrid logic, which adds to the modal description of transition structures the ability to refer to specific states, as the lingua franca for a suitable specification method.

On the other hand, specific applications may require specific logics to describe their configurations. For example, requirements expressed equationally lead to a configurations-as-algebras perspective. But depending on their nature one could also naturally end up in configurations-as-relational-structures, or probabilistic spaces or even in configurations-as-Kripke-structures, if first-order, fuzzy or modal logic is locally used.

The aim of this thesis is to develop the foundations for a specification method based on these principles. To subsume all the possibilities above our approach builds on very general grounds. Therefore, instead of committing to a particular version of hybrid logic, we start by choosing a specific logic for expressing requirements at the configuration (static) level. This is later taken as the base logic on top of which the characteristic features of hybrid logic, both at the level of syntax (i.e. modalities, nominals, etc.) and of the semantics (i.e. possible worlds), are developed. This process is called hybridisation and is one of the main technical contributions of this thesis. To be completely general, it is framed in the context of the theory of...
institutions of Goguen and Burstall, each logic (base and hybridised) being treated abstractly as an institution.

In this setting the thesis’ contributions are the following:

- A method to hybridise arbitrary institutions; this can be understood as a source of logics to support arbitrary configurations-as-local-models specifications.

- A method to lift encodings (technically, comorphisms) from an institution to a presentation in first-order logic, into encodings from its hybridisation to a presentation in first-order logic; this result paves the way to the introduction of suitable automatised proof support for a wide range of hybridised logics.

- Suitable characterisations of bisimulation and refinement for models of (generic) hybridisations, which provide canonical, satisfaction preserving relations to identify and relate models.

- A two-stage specification method for reconfigurable systems based on a global transition structure to capture the system’s reconfiguration space, and a local specification of configurations in whatever logic is found expressive enough for the requirements at hands.

- A set of additional technics to assist the process of specifying and verifying requirements for reconfigurable systems, with partial tool support.
PUBLICATIONS

Some ideas and figures have appeared previously in the following publications:

Put your publications from the thesis here. The packages multibib or bibtopic etc. can be used to handle multiple different bibliographies in your document.
ACKNOWLEDGEMENTS

First of all, I would like to express my sincerely gratitude to my supervisors Prof. Luís S. Barbosa and Prof. Manuel A. Martins. Thank you very much for showing me how interesting can be Logic and Computer Science, and how easily they can live together. Thank you very much for all the support, patience and friendship along this pleasant journey. It was a real privilege working with you.

I would also like to thank to Prof. Răzvan Diaconescu as the external member of my supervising committee. I am grateful for showing me the beauty of working at the abstract level of the theory of institutions, which provided the foundations of this thesis. For the kindness and hospitality during my visit to Sinaia, I would like to express my special gratitude to you and your family.

Thank you very much to Critical Software S.A. for all the support to this project by providing all the conditions for the development of the present work.

I had the opportunity to discuss on this research with a lot of people. Dirk Hoffman, Pedro Nora, Renato Neves, José Faria, George Voutsadakis, Torben Brauner, Ian Hodkinson, Andrzej Tarlecki and Till Mossakovski, all of them, had impact at some particular point of the present work.

Thank you very much to José Faria, Joaquim Tojal and Filipe Pedrosa, for giving me all the conditions to “make accounts with letters” during your hard and substantial work to grow up the ambitious project of Educed.

A special thanks to my parents for all the support since the very beginning.

The modal nature of life, and its irreversible reconfiguration through some events, was the most significative lesson learned during this PhD time. This was shown by means of the introduction of an institution where our own reconfigurable essence, as well as the concreteness of the infinite were proven. My gratitude goes to Susana for the support given in my absence, making the development of this work possible. But mainly, because she is part of the canonical model of this institution, also composed by myself and by its prime members: our daughters. Ana e Clara, muito obrigado por terem nascido durante este processo!
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.

J. Goguen and J. Meseguer in [GM87]
# CONTENTS

List of Figures xv

1 INTRODUCTION 1
   1.1 Problem and motivation 1
   1.2 The approach 2
   1.3 Summary of contributions and roadmap 3
   1.4 Context and related work 6

i FOUNDATIONS 15

2 BACKGROUND 17
   2.1 Hybrid logic(s) 17
   2.2 Category theory 20
   2.3 Institutions 25
      2.3.1 Definition and examples 26
      2.3.2 Examples 27
      2.3.3 Internal logic 33
      2.3.4 Amalgamation 34
      2.3.5 Quantification spaces 36
      2.3.6 Comorphisms 38
      2.3.7 Presentations 41

3 HYBRIDISATION OF INSTITUTIONS 43
   3.1 The hybridisation method 44
      3.1.1 The category of $\mathcal{HJ}$-signatures 44
      3.1.2 $\mathcal{HJ}$-sentences functor 45
      3.1.3 $\mathcal{HJ}$-models functor 48
      3.1.4 The satisfaction relation 50
      3.1.5 Base logic versus its hybridisation 55
   3.2 Examples 57

4 FIRST-ORDER ENCODINGS 67
   4.1 Hybridising FOL-encodings 69
      4.1.1 States parametrisation 69
      4.1.2 The standard translation lifting 73
   4.2 Examples 83
   4.3 Conservativity 91

5 BISIMULATION AND REFINEMENT NOTIONS FOR HYBRIDISED LOGICS 97
   5.1 Bisimulation for hybridised Logics 98
   5.2 Refinements for generic hybridised logics 103
CONTENTS

ii TECHNIQUES 109

6 HYBRIDISATION FOR THE WORKING SOFTWARE ENGINEER 111
   6.1 Motivation 111
   6.2 The approach 113
   6.3 Dealing with heterogenous requirements 128

7 COMPLEMENTARY TECHNICS 133
   7.1 Validation 133
   7.2 Evolving interfaces 136
   7.3 Enabling reconfigurations 140

8 CONCLUSIONS AND FURTHER WORK 145
   8.1 Concluding 145

Epilogue 149
Bibliography 151
<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Figure 1</td>
<td>Proof Scheme</td>
<td>54</td>
</tr>
<tr>
<td>Figure 2</td>
<td>Refinement in $\mathcal{H}MV_{L}$</td>
<td>106</td>
</tr>
<tr>
<td>Figure 3</td>
<td>Refinement in $\mathcal{H}E_{Q}$</td>
<td>107</td>
</tr>
<tr>
<td>Figure 4</td>
<td>Specification of reconfigurable systems: The approach.</td>
<td>114</td>
</tr>
<tr>
<td>Figure 5</td>
<td>Specification of reconfigurable systems: The method.</td>
<td>115</td>
</tr>
<tr>
<td>Figure 6</td>
<td>A plastic buffer.</td>
<td>115</td>
</tr>
<tr>
<td>Figure 7</td>
<td>Plastic Buffer $\mathcal{H}PAR$ signature.</td>
<td>120</td>
</tr>
<tr>
<td>Figure 8</td>
<td>Plastic Buffer $\mathcal{H}PAR$ Specification.</td>
<td>122</td>
</tr>
<tr>
<td>Figure 9</td>
<td>The 'plastic buffer' specification translated into $\mathcal{FOL}$.</td>
<td>127</td>
</tr>
<tr>
<td>Figure 10</td>
<td>A $\mathcal{H}TRM$ specification.</td>
<td>129</td>
</tr>
<tr>
<td>Figure 11</td>
<td>Hierarchical representation of the pump boot.</td>
<td>130</td>
</tr>
<tr>
<td>Figure 12</td>
<td>The reconfigurable calculator in $\mathcal{H}CASL$.</td>
<td>135</td>
</tr>
<tr>
<td>Figure 13</td>
<td>A specification parsed by the generic engine.</td>
<td>135</td>
</tr>
</tbody>
</table>
1.1 PROBLEM AND MOTIVATION

The qualifier reconfigurable is used for systems whose execution modes, and not only the values stored in their internal memory, may change in response to the continuous interaction with the environment. Such systems behave differently in different modes of operation, or configurations, and commute between them along their lifetime.

At present such is more the norm than the exception in software systems whose components are frequently reconfigured — a typical, everyday example is offered by cloud based applications that elastically react to client demands. Reconfigurability, together with related issues like self-adaptation or context-awarness, become a main research topic [RS11], in the triple perspective of foundations, methods and technologies.

In safety-critical systems, the quality of dynamic reconfigurations is vital. In modern cars, for example, hundreds of electronic control units must operate in different modes, depending on the current situation – such as driving on a highway or finding a parking spot. Switching between these modes is a typical example of a dynamic reconfiguration.

As it happens with any other class of inherently complex software, the project of reconfigurable systems requires both expressive modelling notations and mathematically sound methods for development and verification. In safety or mission-critical applications systems’ reconfigurations, often occurring at runtime, need to be specified with special care to make explicit the scope of their application and guarantee the absence of unintended side effects – for example, to enforce that changing the air conditioning settings does not influence the correct operation of the brakes. In many cases the use of sound, i.e., precise and rigorous engineering principles, which in Computer Science are known as formal methods [Bjø06b] is no more an option: not only
to make systems safer by guaranteeing they behave as expected, but also to achieve higher quality and efficiency in software design.

Such is the standpoint for this thesis which aims at contributing to the foundations of a rigorous methodology to the specification and analysis of reconfigurable systems.

From the outset we sought to develop methods and technics both sound, i.e. framed on precise logic foundations, and generic, i.e. applicable to a wide range of reconfigurable systems. In particular, they should be independent of whatever specific logic is used to model the systems’ individual configurations, giving maximum freedom to the Software Engineer.

Although the emphasis of the thesis is on foundations, and its main contribution a method for hybridising logics, the latter paves the way to the development of specification techniques for reconfigurable software. Moreover, and from the outset, the thesis was co-supported by a leading, Portuguese IT company, whose mission includes the production of formally certified software for critical systems. This provided a challenging context for what was originally planed as a thesis-on-theory, namely motivating its methodological counterpart and the search for suitable tool support.

1.2 THE APPROACH

From the very beginning the motto

```
reconfigurations as transitions, configurations as local models
```

emerged as an expressive characterisation of what we were trying to achieve. On the one hand, it embodies a two-layered abstraction between a local specification stage (that of the individual configurations of a system) and a global one (concerning the dynamics of reconfiguration). On the other hand, regarding reconfigurations as transitions suggests some sort of modal logic as the language to express them. Finally, identifying configurations, or operation modes, as local models emphasises that no special restrictions should be put to their specification. I.e. our methods must be prepared to deal with whatever logics are used to formally describe a system’s local configurations.
In a sense to became clear along the thesis, our approach can be described in a rather straightforward way: models for reconfigurable software are structured transition systems described within appropriate logical systems. Their states are the individual configurations with whatever structure they have to bear in concrete applications. Transitions correspond to the admissible reconfigurations.

This view determined the two other choices on the formal setting for the thesis: that of hybrid logic [Bla00, Bra10, AtC07] as the basic language to express evolution (through modalities) and locality (through nominals), and that of the theory of institutions [BG80, Dia08] as the right abstraction to combine the former with the specific logics used to describe each local configuration.

Actually, the thesis builds on the essential principles of the theory of institutions. First of all, as a very general formalisation of what a logical system is, institutions abstract the very essence of hybrid logic from specific syntactic and semantic details of each of its versions. Secondly, the theory provides the general constructions, or more precisely the right framework to develop the general constructions, to combine the global hybrid language with the specification logics used locally.

1.3 SUMMARY OF CONTRIBUTIONS AND ROADMAP

Contributions

The contributions of this thesis span from a foundational to a methodological level.

The first aims at developing the foundations of a specification method for reconfigurable systems, combining whatever logic is used to describe the system’s individual configurations with a hybrid language able to express reconfigurations, the latter encodes into modalities whatever triggers their occurrence, and provides means to name and talk about each local configuration. Instead of committing to a particular version of hybrid logic, we start by choosing a specific logic for expressing requirements at the configuration (static) level. This is later taken as the base logic on top of which the characteristic features of hybrid logic, both at the level of syntax (i.e. modalities, nominals, etc.)
and of the semantics (i.e. possible worlds), are developed. The process is called hybridisation and constitutes the main technical contributions of the thesis.

These results are presented in the first part of the thesis. Most of them are already published in references [MMDB11a, MMB13a]. The complete proofs for results in [MMDB11a] are documented in [MMDB11b]. Further results appear in a current journal submission [DM].

The specific contributions at this level are:

- A method to hybridise arbitrary institutions, which we call the hybridisation process.

- A method to lift encodings (technically, comorphisms) from an institution to a presentation in first-order logic, into encodings from its hybridisation again to a presentation in first-order logic. Note this result paves the way to the introduction of suitable automatised proof support for verification of specifications written in a wide range of hybridised logics.

- A characterisation of both bisimulation and refinement notions for models of (generic) hybridisations, which provides canonical, satisfaction preserving relations to identify and relate models.

The second level of contributions has a methodological nature, resorting to the hybridisation process to propose a rigorous but flexible specification method for reconfigurable systems. Again, most of the results are already published in Formal Methods conferences [MFMB11, MNMB13, NMMB13b].

The methodological contributions of the thesis are:

- A two-stage specification method for reconfigurable systems based on a global transition structure to capture the system’s reconfiguration space, and a local specification of configurations in whatever logic is found expressive enough for the specific application requirements.
• A set of complementary technics to assist the process of specifying and verifying requirements for reconfigurable systems. This includes

  – An incorporation of the hybridisation process into the HETS [MML07] framework to provide effective, computer assisted support for validating specifications.

  – A technique to cater for non homogenous interfaces in the local configurations of a system. This was found a relevant requirement in practice, which is, however, not fully accommodated in the general method.

  – A technique to handle reconfigurations which are triggered by events depending on actual values in the state variables of a local configuration.

Roadmap

As mentioned above the thesis is organised in two main parts. The first one is concerned with foundations, the second with the methodological level.

The foundational part starts in Chapter 2 with a review of the necessary background on hybrid logic, category theory and theory of institutions which provides the technical context for the developments to follow.

Chapter 3 contains the main results on the hybridisation process. Apart its technical specificity, this process sheds light on the generic pattern of hybridisation and provides a ‘source of logics’ for specifying software reconfigurability, as discussed in the second part of the thesis. Its development extends the work [DS07] where it was shown how arbitrary institutions can be ‘modalised’.

Chapter 4 is devoted to the construction of first-order encodings of hybridised institutions. In particular, given an institution ‘encodable’ in presentations in first-order logic, it proposes a systematic construction of a similar encoding for its hybridised institution. Most results in Chapters 3 and 4 appear in [MMDB11a, MMDB11b, DM, Dia13].

The foundational part of the thesis closes in Chapter 5 with the study of suitable equivalence and order relations to compare models of
specifications in hybridised logics. Two such relations are considered: a notion of bisimulation, to relate models with indistinguishable behaviours, and a notion of refinement to connect abstract models to their realisations. The main results of this chapter appeared in [MMB13a].

The methodological part of the thesis comprises Chapters 6 and 7. Chapter 6 introduces and illustrates a specification method for reconfigurable systems based on the hybridisation process discussed in Part I. Through hybridisation one is able to weave together, in a precise mathematical sense, the dynamics of reconfiguration with the local specifications. Hybrid features add to modal logic descriptions the power to refer with a surgical’ precision to the latter. On the other hand, the institution-independent character of the whole approach supports the genericity of the method and allows for transporting specifications and proofs from one logic to another, to take advantage, for example, of suitable tool support. A version of the method introduced in this chapter was published in [MFMB11].

Chapter 7 is devoted to the presentation of three techniques which complement or enrich the specification method. They include a computer-assisted validation of specifications in hybridised logics based on HETS, as well as specific techniques to deal with varying interfaces at the local level and with reconfigurations triggered by specific values of local state variables. Part of this chapter is based on references [MNMB13, NMMB13b].

Besides Parts I and II, the thesis includes this introductory chapter as well as a final concluding one. This Introduction will close with a brief review of the state of the art relevant to both Parts I and II. Such a review is by no means complete, conveying just a personal roadmap through related work.

1.4 CONTEXT AND RELATED WORK

*Hybrid logics*

Modal logics (e.g. [AtC07, BdRV01]) has been recognised as a very relevant area of formal logic with respect to applications. The “possible worlds semantics” they entail, provides a natural framework to deal
with patterns involving evolution and change. This justifies their ubiquity in Computer Science. There is, however, a well-known limitation in standard modal logic: its expressive power is not enough to name or to explicitly refer to specific states of the underlying Kripke structure. Therefore, there is no way to assert the equality between two particular states or to specify properties assigned to particular states within a model.

Hybrid logic [Ind07, Bla00, Bra10, AtC07] addresses this problem through the introduction of propositional symbols of a new sort, called nominals. Each nominal is true at exactly one possible state. The sentences are then enriched in two directions. On the one hand, nominals are used as simple sentences holding exclusively in the state they name. On the other hand, explicit reference to states is provided by sentences $@i \rho$, stating the validity of $\rho$ at the state named $i$. This justifies the “hybrid nature” of nominals: they are simultaneously states in the models and sentences of the language.

The literature abounds in richer versions of hybrid logic. This includes hybrid versions of first order logic (e.g. [Bra10]), many-valued hybrid logic [HBB08], intuitionistic logic [BdP06], computation tree logic (CTL) [Web09, KLSW09], linear temporal logic (LTL) [DLN07], $\mu$-calculus [SV01], among many others.

Historically, hybrid logic was introduced by A. Prior in his book [Pri67]. However, its seminal ideas emerged by the end of the 50’s, in a discussion of C.A. Meredith (cf.[Bla00]). The theme was later revisited, in the school of Sofia, by S. Passy and T. Tinchev (e.g. [PT91]). It achieved global interest within the modal logic community on the 90’s, being contributed by P. Blackburn, C. Areces, B. ten Cate, T. Braüner, T. Bolander, among many others (e.g. [AB01, tCF05, Bra10, BB06]). This lifted the status of Hybrid Logic to an independent branch of modern logic. For an historical account we suggest [Bla00, Bra10] as well as [Bla06] for a comparison with the original perspective of A. Prior.

Institutions and combination of logics

An institution is a categorical formalisation of what a logical system is, encompassing syntax, semantics and satisfaction. The concept was
suggested by Goguen and Burstall, by the end of the seventies, in order to “formalise the formal notion of logical systems”, and in response to the “population explosion among the logical systems used in Computing Science” [GB92]. More precisely, institutions were first introduced in [BG80] (where they were called simply ‘languages’), being the well known, seminal paper [GB92] published much later. Reference [Dia10b] presents an historical perspective on the origins and further developments of the topic, both in Computer Science and mathematical logic.

The universal character and resilience of institutions is witnessed by the wide set of logics formalised and dealt within this framework. Examples range from the standard classical logics, to the most unconventional ones as well as to logical systems underlying modern specification and programming languages. Beyond the examples presented in the thesis, a lot of other logics and specification paradigms can be approached along the institutional lines. Well known examples include process algebras [MR06], temporal logics [Cen98], probabilistic logics [BK105], quantic logics [CMSS06], hiding and observational logics [BD94, BH06, Mar06], coalgebraic logics [C06], functional [ST12, SM09] and imperative programming languages [ST12], higher-order logic [Bor99] among many others.

The original motivation for institutions was the need to abstract from particular details of the individual logics by creating a theory to enlighten and characterise many aspects that “are completely independent of what underlying logic is chosen” [GB92]. This was the motto for the development of a solid institution-independent specification theory, upon which, the modular structuring and parameterisation mechanisms used in algebraic specification were defined ‘once and for all’, abstracting from the concrete particularities of each specification logic (e.g. [Mos04, Tar03, Dc11]). A fruitful recent trend of the field is the institution-independent perception that most of the technics and methods of the classical model theory can be defined at the abstract level of arbitrary institutions [Dia08].

The ability to relate different logics is a decisive issue in formal logic. This was also a driven force on the development of the theory of institutions (e.g. [MDT09]). In particular, institutions provide a systematic way to relate logics and transport results from one to an-
other [Mos03], which means that a theorem prover for the latter can be used to reason about theorems of the former. These translations pave the way for the heterogeneous specification paradigm where different institutions are combined and used along a specification process [Dia02, Mos02, Dia98, DF02, MML07].

The combination of logics is an active research topic in modern logic. Particularly, the kind of combinations where the particular features of a logic are combined ‘on top’ of another one have been considered in several contexts. The hybridisation process proposed in this thesis (as in [MMDB11a]), extending the previous work by R. Diaconescu and P. Stefaneas [DS07] on ‘modalisation’ of institutions, which endows systematically institutions with Kripke semantics (w.r.t. the standard □ and ◇ modalities), can be regarded as one of these combinations. R. Fajardo and M. Finger also presented in [FF02] a method to modalise logics, and proved that both completeness and decidability of the source logics are preserved. The ‘temporalisation’ of logics introduced by M. Finger and D. Gabbay in [FG92] and the recent ‘probabilisation’ of logics introduced by P. Baltazar in [Bal10] are other remarkable examples on this kind of combinations.

Other proposals in the literature abstract the combination pattern by considering the ‘top logic’ itself arbitrary. Such is the case of what is called parametrisation of logics in [CSS99] by C. Caleiro, A. Sernadas and C. Sernadas. In brief, a logic is parametrized by another one if an atomic part of the first is replaced by the second. Therefore, the method distinguishes a parameter to fill (the atomic part), a parametrised logic (the ‘top’ logic) and a parameter logic (the logic inserted within). The recent method of importing logics suggested by J. Rasga, A. Sernadas and C. Sernadas [RSS12] aims at formalising this kind of asymmetric combinations resorting to a graph-theoretic approach.

Specification of reconfigurable systems

As explained earlier in this chapter, we intend to explore hybridised logics to frame a general approach to the specification of reconfigurable systems. By general we mean independent of whatever logic one finds suitable to describe the system’s individual configurations.
The rationale underlying our approach seeks to combine two basic dimensions in systems specification: one which emphasises behaviour and its evolution, another focused on data and their transformations. To be able to cope, within a single formalism, with both data structuring and prescription of functionality, as well as with specification and analysis of (externally observable) behaviour remains a main challenge for Software Engineering.

Behaviour is typically specified through (some variant of) state machines. Such models capture evolution in terms of event occurrences and their impact in the system’s internal state configuration (see e.g. [AILS07]). Data types and services upon them, on the other hand, are often presented as theories in suitable logics, over a signature which offers a syntactic interface to the system. Semantics is, then, given by a class of concrete algebras acting as models of the specified theory (see e.g. [MHST03]).

The dichotomy between data and behaviour is mirrored in a proper mathematical duality: both initial algebras and final coalgebras provide abstract descriptions of those computational structures. Moreover, as universal properties, they entail definitional and proof principles, i.e., a basis for the development of program calculi directly based on (actually driven by) type specifications. The notion of a dialgebra [PZ01, You10] which embodies an algebra and a coalgebra over a common state space, conceptually unifies both structures, at the price of loosing crucial properties, for example the existence of final/initial models, i.e. of canonical representatives of behaviour.

Our starting point is that these two dimensions are interconnected: the functionality offered by a reconfigurable system, at each moment, may depend on the stage of its evolution. In [MFMB11] the reconfiguration dynamics is modelled as a transition system, whose nodes are interpreted as the different configurations it may assume. Therefore, each of such nodes is endowed with an algebra, or even a first-order structure, to formally characterise the semantics of the services offered in the corresponding configuration. Technically, models of reconfigurable systems are given as structured state-machines whose states denote algebras, rather than sets.

Structured transition systems [CM92] are, therefore, the semantic structure underlying the approach proposed in this thesis to the spec-
ification of reconfigurable systems. They are usually obtained by extending the bare structure of (sets of) states and transitions with further elements in either of them (e.g., structured labels, weights, functions, algebraic structure on states, etc). A quite general characterisation was proposed by R. Heckel and A. Corradini through the concept of \textit{lax coalgebra} \cite{CGRH01} which incorporates algebraic structure in both labels and states.

There are two ways to mathematically represent transition systems: either as \textit{graphs} or as \textit{coalgebras} by regarding the transition relation as a function from states to some collection of states whose shape is determined by a suitable endofunctor. It is not surprising that the literature on formal models of software reconfiguration explores both these paths. Graph rewriting techniques\cite{Roz97}, notably the double pushout approach \cite{HMP01}, have been extensively used in modelling the evolution of dynamic systems. Typical applications emerge in the areas of mobile processes, from the $\pi$-calculus \cite{MPW92} and its variants \cite{SW03} to the latter work of Robin Milner in bigraphs \cite{CMS07,Mil09}, architectural evolution \cite{WF02,BBG+08} or coordination of software services \cite{KMLA11}, among many others.

Representing reconfiguration as transition systems described by coalgebras or, more often, in terms of their relational counterpart, has also been considered in the literature. The approaches are more diverse than in the graph-oriented trend and often endowed with a particular variant of a modal logic. Some examples include the separate specification of a second transition structure to monitor the basic one, often called a reconfiguration manager, as e.g. in \cite{BFH+12}, the use of feature enriched transition systems \cite{CCS+13} or well-structured transition systems \cite{FS01}. The latter incorporates an order on (an infinite) state space, compatible with the transitions, with interesting decidability results. The specification of contracts or interaction conflicts in either states or transitions, in the context of the well known design-by-contract formalism, also provides mechanisms to talk about reconfigurations (see e.g. \cite{TCCD10,FMFR11,BHW11,CZZ12}).

In both cases a critical ingredient to incorporate the reconfigurability dimension in specifications is the ability to add structure, typically algebraic structure, to the transition system modelling the system’s behaviour. In a sense such is also the path taken in this thesis. But the
combination of what, after Rutten’s seminal work on universal coalgebra, are called algebraic and coalgebraic structures, has a long trace in Computer Science.

A first landmark was the whole research trend on *behavioural satisfaction*, early references being [GGM76] and [Rei85]. Hidden-sort algebra [GM00], embodying a fundamental distinction between visible values and internal states, which can only be observed in an indirect way, is an example of a behavioural formalism whose development was triggered, from the outset, by research on the foundations of object-oriented programming. Specification with *coherent hidden algebras* [DF02] and *observational algebras* [HB98] are remarkable approaches in this line. Reference [ST08] provides a comprehensive account of the area.

Another research direction, somehow closer to the approach proposed in this thesis, seeks to combine explicitly algebraic specifications with state-based structures. Also motivated by the emergence of object orientation in the 80’s, the specification language TROOL [JSHS96] is a paradigmatic example. Objects, defined by attributes and evolving in response to events, are described as abstract data types and their evolution as linear processes specified in a temporal logic. Reference [CR97] introduces a logic which combines many-sorted first-order logic with branching-time combinators, with both initial and loose semantics.

Introduced by M. Broy and M. Wirsing in [BW00], *algebraic state machines* take algebraic specifications as states, an idea which was also present in Y. Gurevich seminal work on *evolving algebras* [Gur94], posteriorly renamed to *abstract states machines* [BS03]. These machines, aiming at modelling arbitrary computational processes, consists of transition systems where each state has a structure of an algebra. The initial state consists of a particular algebra and each transition in a command that triggers an update on the current algebra. Hence, the set of transitions can be regarded as an abstract (imperative) program to be executed over the assigned initial state. The impact of abstract state machines in formal modelling cannot be understated. Several key ideas were borrowed by, and later incorporated in, popular model-oriented formalisms, namely the B method. A recent manifestation of this *states as algebras* perspective appears in the work of M. Bidoit
and R. Hennicker in [BH08] as a semantic foundation for the contract-based design of software components. This perspective developed into a whole approach to software architecture based on a two layered semantics (at the interface and internal levels) with precise notions of composition and refinement.

As the reader will be able to appreciate in the second part of this thesis, our own perspective has several points of contact with these approaches based on structured transition systems. Note, however, that in our models a state does not correspond to a configuration of variables over a unique, common, fixed first-order structure, but to a specific structure modelling the configuration behavior and functionality. Technically, we resort to rigid variables for non rigid operations, in contrast to other, more disseminated approaches where rigid operations act upon non rigid variables. On the other hand our specifications are always axiomatic and expressed in a logic which results from the hybridisation of the logic found suitable for each application to capture its possible configurations.

The use of different, often domain-tailored logics to specify reconfigurations in software constitutes a wide and heterogenous landscape in which our own approach fits in. We mention some examples, for illustration but with no pretension of exhaustibility. An important one is J. Meseguer’s rewriting logic [Mes92] and its MAUDE realisation, a language whose dynamics is based on the concurrent transformation of a ‘soup’ of objects and messages. A detailed overview, including references to modelling evolution and self-adaptability, is provided in [Mes12]. Logic based formalisms are also common in specifying reconfiguration in component-based paradigms, dynamic software architectures and coordination schemes. An example of the first is given by the work of O. Kouchnarenko and her collaborators in which reconfigurations are specified in a temporal pattern logic [DKL10] in the context FRACtAL [BCL+06], a paradigmatic component model. The work of T. Maibaum [AM02, CAPM10] and J. Fiadeiro [FL13], as well as of the Pisa or the Munchen Schools (see e.g. [BLLM08] or [vRHWS08], the latter work developed in an institutional framework), among many others [ZL12], exemplify applications to dynamic architectures. Finally, on the coordination side, the work of D. Clarke on what is called reconfiguration logic [Cla08] expresses evolution of
REO [Arb04] connectors in formulas of a modal logic evaluated over constrained automata.

If dynamic reconfigurations increase the availability and the reliability of component-based systems by allowing their architectures to evolve at run-time, there is a number of other domains in Computing where reconfigurability, and the search for suitable formal methods, are emerging as a crucial issue. These include areas in the intersection of discrete and continuous behaviour, namely in what concerns sensor networks [vRHWS08] and robotics [YWM06]. Whether the approach introduced in the thesis scales, on the one hand, to these areas, and to real, industrial cases, on the other only time and effort will tell. The study of hybridisation of logics to deal with probabilistic [CS02, Dob10, HRWW11] and continuous [Pla10] reasoning would be the next, natural step to take.
Part I

FOUNDATIONS
BACKGROUND

This chapter provides a brief introduction to three main topics in Mathematics which underlie the thesis contribution: hybrid logic (in Section 2.1), as a main language to express properties of reconfigurable software; category theory (Section 2.2), which provides the relevant mathematical constructions grounding the development of the thesis; the theory of institutions (Section 2.3), a categorical abstract theory in which the hybridisation process is framed. It should be remarked, however, that only the main concepts and results relevant to the thesis are recalled here. The interested reader is referred to specialised references in each of those areas for a detailed account.

2.1 HYBRID LOGIC(S)

Modal logic (ML) is an active branch of modern logic with a wide range of applications in Computer Science (cf. [BVB07]). Its simplest (propositional) version is briefly presented as follows: the signatures are sets of symbols, say Prop, and the sentences are defined by the grammar

\[ \rho := \text{Prop} \mid \neg \rho \mid \Box \rho \mid \Diamond \rho \mid \rho \otimes \rho \]

where \( \otimes \in \{ \lor, \land, \Rightarrow \} \). Models for a signature Prop consists of a Kripke frame \( W \) and a proposition assignment function \( M \). I.e., a model consists of a pair \((M, W)\) where \( W \) is a relational structure over a set \( |W| \), called the set of worlds or the state space, \( W_R : |W| \times |W| \) is a binary relation, called accessibility relation, and \( M \) is a family of functions \( M = (M_w : \text{Prop} \to \{ \top, \perp \})_{w \in |W|} \) called the propositional assignment. Note that usually, a proposition assignment is defined as a function from the set of propositions into the powerset of worlds, i.e., \( f : \text{Prop} \to \mathcal{P}(|W|) \) (cf. [Bla00]). Of course both definitions are equivalent making, \( f(p) = \{ w \in |W| | M_w(p) = \top \} \).
The satisfaction relation $\models_{\text{mpl}}^\text{Prop}$ is inductively defined as follows:

$$(M, W) \models_{\text{mpl}}^\text{Prop} \rho \text{ iff for any } w \in |W|, (M, W) \models^w \rho,$$

where

- $(M, W) \models^w p \text{ iff } M_w(p) = \top$;

- $(M, W) \models^w \rho \lor \rho' \text{ iff } (M, W) \models^w \rho \text{ or } (M, W) \models^w \rho'$ and similarly for the connectives $\land, \Rightarrow$;

- $(M, W) \models^w \neg \rho \text{ iff } (M, W) \not\models^w \rho$;

- $(M, W) \models^w \Box \rho \text{ iff there is some } w' \in |W| \text{ such that } (w, w') \in W_R \text{ and } M \models^{w'} \rho$; and

There are many variants and extensions to this basic modal language in the literature. For instance, we may consider logics with multi-modalities, i.e., with more than one modality or, more generally, with polyadic modalities (i.e., $n$-ary modalities) [BdRV01]. In the later case, signatures are pairs $(\text{Prop}, \Lambda)$ where $\Lambda$ is an arity-indexed family of modality symbols (where $\lambda \in \Lambda_{n+1}$ is interpreted as a $n$-ary modality). Moreover, in some versions, worlds may carry further structure, for instance, algebraic or first-order structures (cf. [ACP06] or [DS07] for a detailed account). Other classes of logics are obtained by restricting the semantics to particular classes of models. These are the cases of the well known logics $T$, $S4$, and $S5$ whose models are restricted to those in which $W_R$ is, respectively a reflexive, preorder, or equivalence relation (e.g. [BdRV01]).

There is, however, a typical limitation in standard modal logic: it lacks expressive power to name or to explicitly mention specific states in a model. Therefore, there is no way to assert the equality between two particular worlds or the existence of a transition between them. Moreover, there is a number of properties, for example irreflexivity of the underlying accessibility relation, that can not be axiomatised by the same reason (cf. [Ind07] for a list of simple properties that are not expressible in ML). Hybrid logic [Ind07, Bla00, Bra10, AtC07] overcomes this limitation by introducing a new kind of symbols Nom, called nominals, to make explicitly reference to states in models. Sentences are then enriched in two directions. On the one hand, each
nominal is used as a simple sentence holding exclusively in the world it names; on the other hand, sentences $@_i \rho$, for $i \in \text{Nom}$, state the validity of $\rho$ at the world named by $i$. Hence, the (propositional) hybrid logic sentences are defined by the grammar:

$$\rho := \text{Prop} | \text{Nom} | @_i \rho | \neg \rho | \Box \rho | \Diamond \rho | \rho \otimes \rho$$

$i \in \text{Nom}$ and $\otimes \in \{\lor, \land, \Rightarrow\}$.

The models $(M, W)$ are defined as in the standard modal case, with nominals interpreted as constants over $W$. The satisfaction relation $\models^{(PL,\text{Nom})}_{\text{Prop}}$ extends $\models^{\text{MPL}}_{\text{Prop}}$ with

- $(M, W) \models^w_i$ iff $W_i = w$;
- $(M, W) \models^w@_i \rho$ iff $(M, W) \models^{W_i} \rho$,

where $W_i$ denotes the world named by nominal $i$. This makes possible to express properties such as: the equality between the worlds denoted $i$ and $j$ (with $@_i j$), that a property holds on a world named $i$ (with $@_i \rho$), that there is a transition between the worlds named by $i$ and $j$ (with $@_i \Diamond j$) or the irreflexivity on the underlying accessibility relation in a word named by $i$ (using $i \Rightarrow \neg \Diamond i$). The first sentence above plays the role of the “the classical theory of equality” on modal logic. This is another reason for calling hybrid logic “hybrid” (cf. [Bla00]).

An attractive aspect of this logic is the fact that, the increased expressive power has no cost on the complexity of the proof methods (cf. [ABM99]).

Other very expressive variants of the propositional hybrid consider quantification over worlds (e.g. [BT98]): $(\forall i) \rho$, whose semantics is

- $(M, W) \models^{(\text{Prop}, \text{Nom})}_{\text{Prop}} (\forall i) \rho$ iff for any $\{i\}$-expansion $(M’, W’)$ of $(M, W)$, $(M’, W’) \models^{\text{Prop}, \text{Nom} + \{i\}}_{\text{Prop}} \rho$

Binding world-variables to worlds is a common feature in many hybrid logics. This is done through the introduction of the binder operator $\downarrow$ (originally suggested in [Gor96]). The semantics of $\downarrow i \rho$ can be given by abbreviating $\downarrow i \rho \abr (\forall i) (i \Rightarrow \rho)$ or, equivalently, to $(\exists i) (i \land \rho)$.

Multi-modalities and polyadic modalities are naturally extended from the (pure) modal case to the hybrid one [Bla00, Hod10].
2.2 CATEGORY THEORY

Roughly speaking, categories deal with arrows and their composition, in the same sense that sets deal with elements, their aggregation and membership. An arrow is an abstraction of the familiar notion of a function in set theory or of a homomorphism in algebra. Depicted as an arrow connecting two objects $X$ and $Y$, called its source and target, respectively, it may be thought of as a transformation, or, simply, a link between them. The sources and targets of all the arrows in a category, form the class of its objects. If the same object is both the target of an arrow $f$ and the source of another arrow $g$, $f$ and $g$ are said to be composable. Arrow composition is thus a partial operation and what the axioms for a category say is that arrows and arrow composition form a sort of generalised monoid. This builds up the basis for a very general mathematical framework around the notions of functoriality (the coherent transformation of both objects and arrows), naturality (an abstract counterpart of the notion of polymorphism in type theory) and universality. The latter is not only a main issue in category theory, but also a pervasive topic in mathematics. In brief, an entity $e$ is universal among a family of ‘similar’ entities if it is the case that every other entity in the family can be reduced or traced back to $e$. Moreover, the dual of an universal is still an universal, which adds duality as a fourth main ingredient of the categorial framework. Several examples of universal constructions are reviewed below and extensively used in this thesis.

In order to make the text self-contained and to fix the notation, we revisit below some basic definitions [Lan98, AHS90, Awo06].

**Definition 2.2.1 (Category)** A category $\mathcal{C}$ consists of

- a class of objects $|\mathcal{C}|$;
- a class of arrows (or morphisms) denoted by $\mathcal{C}$;
- two maps $\text{dom}, \text{cod} : \mathcal{C} \rightarrow |\mathcal{C}|$ giving the domain and codomain of each arrow such that, for each $A, B \in |\mathcal{C}|$,
  $$\mathcal{C}(A, B) = \{f \in \mathcal{C}|\text{dom}(f) = A \text{ and } \text{cod}(f) = B\}$$

is a set,
• for any objects \( A, B, C \in C \), a composition map

\[
\circ : C(A, B) \times C(B, C) \to C(A, C),
\]

such that the following diagram commutes

\[
\begin{array}{ccc}
A & \xrightarrow{f} & B \\
\downarrow{f;g} & & \downarrow{g;h} \\
C & \xrightarrow{h} & D
\end{array}
\]

• an identity map \( 1 : |C| \to C \) such that for each \( A \in C \), \( 1_A \in C(A, A) \), the following diagram commutes

\[
\begin{array}{ccc}
A & \xrightarrow{f} & B \\
\downarrow{1_A} & & \downarrow{1_B} \\
A & \xrightarrow{f} & B
\end{array}
\]

The category \( \text{Set} \) takes as objects sets and as arrows functions with the usual functional composition. The category \(^1\) \( \text{CAT} \), known as the category of categories, takes as objects categories and, as arrows, functors between them:

**Definition 2.2.2 (Functor)** A functor \( F : \mathcal{A} \to \mathcal{B} \) between categories \( \mathcal{A} \) and \( \mathcal{B} \) is a map which

1. associates objects to objects \( |F| : |\mathcal{A}| \to |\mathcal{B}| \), and

2. arrows to arrows, \( F_{A,B} : \mathcal{A}(A, B) \to \mathcal{B}(F(A), F(B)) \) such that

- \( F(1_A) = 1_{F(A)} \),
- \( F(f; g) = Ff; Fg \).

We say that a category \( \mathcal{A} \) is a subcategory of \( \mathcal{B} \), in symbols \( \mathcal{A} \subseteq \mathcal{B} \), when

- \( |\mathcal{A}| \subseteq |\mathcal{B}| \),
- for any \( A, B \in \mathcal{A} \), \( \mathcal{A}(A, B) \subseteq \mathcal{B}(A, B) \) and

\(^1\) Strictly speaking, this is only a ‘quasi-category’ living in a higher set-theoretic universe (cf. [Lan98]).
• the composition is $A$ is the restriction of the composition in $B$ to $|A|$. 

We define the dual of a category $C$, denoted by $C^{\text{op}}$, as the category obtained by reversing the arrows in $C$, i.e., $|C^{\text{op}}| = C$ and $C^{\text{op}}(A, B) = C(B, A)$. Moreover, the dual of a functor $F : A \to B$ is a functor $F^{\text{op}} : A^{\text{op}} \to B^{\text{op}}$ such that, for any $A \in A$, $F^{\text{op}}(A) = F(A)$ and $F^{\text{op}}(B \to A) = F(A \to B)$ (cf. [AHS90]). A contravariant functor from the category $A$ to $B$ is a functor $F : A^{\text{op}} \to B$, mapping arrows $f \in A(A, B)$ into arrows $Ff \in B(FB, FA)$. It is easy to see that $(C^{\text{op}})^{\text{op}} = C$ and $(F^{\text{op}})^{\text{op}} = F$.

**Definition 2.2.3 (Natural Transformation)** Let $F, G : A \to B$ be two functors. A natural transformation $\tau$ between $F$ and $G$ consists of a family of arrows $(\tau_a : Fa \to Ga)_{a \in |A|}$ such that for any $f \in A(A, B)$, $\tau(A); Gf = Ff; \tau(B)$, i.e., the following diagram commutes:

$$
\begin{array}{ccc}
F(A) & \xrightarrow{\tau(A)} & G(A) \\
\downarrow Ff & & \downarrow Gf \\
F(B) & \xrightarrow{\tau(B)} & G(B)
\end{array}
$$

We write $\mathcal{A} \xrightarrow{\tau} \mathcal{B}$, or simply, $\tau : F \Rightarrow G$ to say that $\tau$ is a natural transformation between $F$ and $G$.

**Definition 2.2.4 (Pullback and Pushout)** A pullback for two arrows $f \in C(X, B)$ and $g \in C(Y, B)$ consists of a triple $(A, p, q)$ such that

- the diagram $A \xrightarrow{q} Y$ commutes, and

$$
\begin{array}{ccc}
A & \xrightarrow{q} & Y \\
p & \downarrow & \downarrow g \\
X & \xrightarrow{f} & B.
\end{array}
$$
• for any other commuting square \( Z \xrightarrow{s} Y \), there is a unique morphism \( h : Z \to A \) making the following diagram to commute

\[
\begin{array}{c}
Z \\
\downarrow^s & \\
A \xrightarrow{q} Y \\
\downarrow^r & \\
X \xrightarrow{f} B.
\end{array}
\]

Dually, a pushout for \( f \in C(B, X) \) and \( g \in C(B, Y) \), consists of a triple \((A, s, t)\) such that

• the diagram \( B \xrightarrow{g} Y \) commutes and,

\[
\begin{array}{c}
B \\
\downarrow^g & \\
X \xrightarrow{f} A
\end{array}
\]

• for any other commutative diagram \( B \xrightarrow{g} Y \), there is a unique arrow \( h : A \to Z \) commuting the following diagram

\[
\begin{array}{c}
B \xrightarrow{g} Y \\
\downarrow^f & \\
X \xrightarrow{t} A \\
\downarrow^s & \\
\downarrow^u & \\
X \xrightarrow{v} Z
\end{array}
\]

Pullbacks and pushouts, which are dual to each other, can be understood as particular cases of a limit and, a colimit respectively. In order to introduce them, we first define the notions of a diagram and a cone. A diagram of type \( I \) in \( A \) consists of a functor \( D : I \to A \). Category \( I \) is called index of the diagram and \( D_i \) used to denote the image \( D(i) \), for \( i \in I \). A cone for a diagram \( D \) consists of an object \( A \in A \) to-
gether with a family of arrows \((C_i : A \to D_i)_{i \in I}\) such that, for any \(f \in I(i, j)\), the triangle

\[
\begin{array}{c}
A \\
\downarrow^{c_i} \\
D_i \\
\downarrow^{c_f} \\
D_j \\
\end{array}
\]

commutes. Dually, a cocone for a diagram \(D\) consists of an object \(A \in \mathcal{A}\) together with a family of arrows \((C_i : D_i \to A)_{i \in I}\) such that, for any \(f \in I(i, j)\), the triangle

\[
\begin{array}{c}
D_i \\
\downarrow^{c_f} \\
A \\
\downarrow^{c_i} \\
D_j \\
\end{array}
\]

commutes.

**Definition 2.2.5 (Limits and Colimits)** A limit for a diagram \(D : I \to \mathcal{A}\), or a \(D\)-diagram for short, consists of a cone \((C_i : A \to D_i)_{i \in I}\) such that, for any other cone \((C_i' : A' \to D_i)_{i \in I}\) there is an unique arrow \(Z \in \mathcal{A}(A', A)\) such that the diagram

\[
\begin{array}{c}
A' \quad \quad \quad \quad \quad \quad \quad Z \\
\downarrow^{c_i'} \\
D_i \\
\downarrow^{c_i} \\
A \\
\end{array}
\]

commutes, for any \(i \in I\).

A colimit for a diagram \(D : I \to \mathcal{A}\), or a \(D\)-colimit for short, consists of a cocone \((C_i : D_i \to A)_{i \in I}\) such that, for any other cocone \((C_i' : D_i \to A')_{i \in I}\), there is an unique arrow \(Z \in \mathcal{A}(A, A')\) such that the diagram

\[
\begin{array}{c}
A' \quad \quad \quad \quad \quad \quad \quad Z \\
\downarrow^{c_i'} \\
D_i \\
\downarrow^{c_i} \\
A \\
\end{array}
\]

commutes, for any \(i \in I\).

Finally we observe that a pullback in a category \(\mathcal{A}\) is a limit for a diagram \(D : I \to \mathcal{A}\) when \(I\) is the category completely defined by the objects and arrows \(i \longrightarrow j \leftarrow k\). Analogously, a pushout in \(\mathcal{A}\)
can be seen as a colimit of a diagram \( D : I \to A \) for \( I \) the category completely defined by \( \xymatrix{ i & j \ar[l] \ar[r] & k } \). Moreover, final and initial objects can also be seen respectively as the limit and the colimit for \( D : I \to A \), where \( I \) is the empty category, i.e., the category with an empty set of objects.

We say that a category has (co)limits of type \( D : I \to A \) when every \( D \)-diagram has a (co)limit in \( A \).

**Definition 2.2.6 (Preservation and lifting of limits by a functor)** A functor \( F : A \to B \) preserves limits for a diagram \( D : I \to A \) if, for any \( D \)-limit \( (C_i : A \to D_i)_{i \in |I|} \), \( (FC_i : FA \to FD_i)_{i \in |I|} \) is a limit for the diagram \( D; F \).

A functor \( F : A \to B \) lifts limits for a diagram \( D : I \to A \) if, for any \( D; F \)-limit \( (C'_i : B \to D'_i)_{i \in |I'|} \), there exists a \( D \)-limit \( (C_i : A \to D_i)_{i \in |I|} \) such that for any \( i \in |I| \), \( FC_i = C'_i \).

The preservation and lifting of colimits is defined analogously.

## 2.3 Institutions

As stated in the previous chapter, an institution consists of a categorical formalisation of a logical system, encompassing syntax, semantics and satisfaction. We introduce, in this section, basic definitions and notations used in this thesis. For a complete reference on the issue we suggest [Dia08].

The section is organised as follows: the definition of institution and respective notation is introduced in Section 2.3.1, followed by a set of illustrative examples in Section 2.3.2. Then, the notions of amalgamation and the institutional-independent treatment of quantification are exposed in Section 2.3.4 and Section 2.3.5. Finally, the last two sections are concerned with the presentation of how to relate and encode institutions: the notion of comorphism is presented in Section 2.3.6, followed by its theoroidal version in Section 2.3.7.
2.3.1 **Definition and examples**

**Definition 2.3.1 (Institution)** An institution

\[ J = (\text{Sign}^I, \text{Sen}^I, \text{Mod}^I, (\models^I)_{\Sigma \in |\text{Sign}^I|}) \]

consists of

1. a category \( \text{Sign}^I \) whose objects are called signatures,
2. a functor \( \text{Sen}^I : \text{Sign}^I \to \text{Set} \) giving for each signature the set of sentences over that signature,
3. a functor \( \text{Mod}^I : (\text{Sign}^I)^{\text{op}} \to \text{CAT} \), giving for each signature \( \Sigma \) the category of \( \Sigma \)-models, and the respective \( \Sigma \)-(models) homomorphisms, and
4. a relation \( \models^I_{\Sigma} \subseteq |\text{Mod}^I(\Sigma)| \times |\text{Sen}^I(\Sigma)| \) for each \( \Sigma \in |\text{Sign}^I| \), called the satisfaction relation,

such that for each morphism \( \varphi : \Sigma \to \Sigma' \in \text{Sign}^I \), the satisfaction condition

\[ M' \models^I_{\Sigma} \text{Sen}^I(\varphi)(\rho) \iff \text{Mod}^I(\varphi)(M') \models^I_{\Sigma} \rho \quad (2) \]

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

\[
\begin{array}{ccc}
\Sigma & \xrightarrow{\varphi} & \Sigma' \\
\text{Mod}^I(\Sigma) & \xrightarrow{\models^I_{\Sigma}} & \text{Sen}^I(\Sigma) \\
\text{Mod}^I(\varphi) & \xrightarrow{\models^I_{\Sigma}} & \text{Sen}^I(\varphi) \\
\Sigma' & \xrightarrow{\models^I_{\Sigma'}} & \text{Sen}^I(\Sigma')
\end{array}
\]

As usual, \( M^* \) denotes the set of sentences \( \{ \rho \in \text{Sen}(\Sigma) \mid M \models_{\Sigma} \rho \} \) and for any \( \Gamma \subseteq \text{Sen}(\Sigma) \), \( \Gamma^* \) denotes the class of models \( \{ M \in |\text{Mod}(\Sigma)| \mid M \models_{\Sigma} \rho \} \) for each \( \rho \in \Gamma \). Moreover, \( \Gamma' \models \Gamma \) means that \( \Gamma'^* \subseteq \Gamma^* \).

The attentive reader may wonder about some subtle variants of the definition that can be found in the literature. For instance, item (3) appears sometimes as

\[ \text{Mod}^I : (\text{Sign}^I) \to \text{CAT}^{\text{op}} \]

(see e.g. [GB92]). However, by duality, both formalizations are equivalent. Moreover, in the original version of the concept in [BG80], the
semantics of an institution is simply characterised by a set of models (instead of a category), given by

\[ \text{Mod}^I : \text{Sign}^I \to \text{Set}^{\text{op}}. \]

The new, more general, definition was latter justified by the authors in [GB92].

2.3.2 Examples

We present, on this section, a number of logics defined as institutions.

**Example 2.3.1 (TRM)** We define the institution \( \text{TRM} \) as follows: \( \text{Sign}^{\text{TRM}} \) is the terminal category, i.e., the category whose class of objects consists of the singleton set \( \{ \ast \} \) and, whose morphisms is the identity \( 1_{\ast}(\ast) = \ast \). The sentences functor \( \text{Sen}^{\text{TRM}} \) sends object \( \ast \) into the empty set \( \emptyset \) and, morphism \( 1_{\ast} \), into the empty function. The models functor \( \text{Mod}^{\text{TRM}} \) sends the signature \( \ast \) to the terminal category. Since the set of sentences is empty, the satisfaction condition trivially holds.

\[ \circ \]

**Example 2.3.2 (FOL)** Let us define \( \text{FOL} \), the institution of first order logic with equality in its many sorted form.

*Signatures* are triples \( (S, F, P) \) consisting of

- a set of sort symbols \( S \),
- a family \( F = \{ F_{\text{ar}} : s \mid \text{ar} \in S^*, s \in S \} \) of sets of function symbols indexed by arities \( \text{ar} \) (for the arguments) and sorts \( s \) (for the results), and
- a family \( P = \{ P_{\text{ar}} \mid \text{ar} \in S^* \} \) of sets of relation (predicate) symbols indexed by arities.

*Signature morphisms* map the three components in a compatible way. This means that a signature morphism \( \varphi : (S, F, P) \to (S', F', P') \) consists of

- a function \( \varphi^{\text{st}} : S \to S' \),
• a family of functions $\varphi^{\text{op}} = \{ \varphi^{\text{op}}_{\text{ar} \to s} : F_{\text{ar} \to s} \to F'_{\varphi^{\text{op}}(\text{ar}) \to \varphi^{\text{st}}(s)} \mid \text{ar} \in S^*, s \in S \}$, and

• a family of functions $\varphi^{\text{rl}} = \{ \varphi^{\text{rl}}_{\text{ar} \to s} : P_{\text{ar} \to s} \to P'_{\varphi^{\text{rl}}(\text{ar})} \mid \text{ar} \in S^*, s \in S \}$.

Models $M$ for a signature $(S, F, P)$ are first order structures interpreting each sort symbol $s$ as a set $M_s$, each function symbol $\sigma$ as a function $M_\sigma$ from the product of the interpretations of the argument sorts to the interpretation of the target sort, and each relation symbol $\pi$ as a subset $M_\pi$ of the product of the interpretations of the argument sorts. By $|M|$ we denote $(M_s)_{s \in S}$ and call it the universe of $M$ or the carrier set(s) of $M$. In order to avoid the existence of empty interpretations for sorts, we assume that each signature has at least one constant (i.e. function symbol with empty arity) for each sort. A model homomorphism $h : M \to M'$ is an indexed family of functions $\{ h_s : M_s \to M'_s \mid s \in S \}$ such that

• $h$ is an $(S, F)$-algebra homomorphism $M \to M'$, i.e., $h_s(M_\sigma(m)) = M'_\sigma(h_{\text{ar}}(m))$ for each $\sigma \in F_{\text{ar} \to s}$ and each $m \in M_{\text{ar}}$, and

• $h_{\text{ar}}(m) \in M'_\pi$ if $m \in M_\pi$ (i.e. $h_{\text{ar}}(M_\pi) \subseteq M'_\pi$) for each relation $\pi \in P_{\text{ar}}$ and each $m \in M_{\text{ar}}$.

where $h_{\text{ar}} : M_{\text{ar}} \to M'_{\text{ar}}$ is the canonical component-wise extension of $h$, i.e. $h_{\text{ar}}(m_1, \ldots, m_n) = (h_{s_1}(m_1), \ldots, h_{s_n}(m_n))$ for $\text{ar} = s_1 \ldots s_n$ and $m_i \in M_{s_i}$ for $1 \leq i \leq n$.

For each signature morphism $\varphi$, the reduct $M'_{\mid \varphi}$ of a model $M'$ is defined by $(M'_{\mid \varphi})_x = M'_{\varphi(x)}$ for each sort, function, or relation symbol $x$ from the domain signature of $\varphi$.

Sentences are the usual first order formulas built from equational and relational atoms by iterative application of Boolean connectives and quantifiers. The variables are disjoint from the constants of the signature, $\text{Sen}^{\text{FOL}}$ is functorial and there is no overloading of variables (which in certain situations would cause a failure of the Satisfaction Condition). This is achieved by considering that a variable for $(s, F, P)$ is a triple of the form $(x, s, (S, F, P))$ where $x$ is the name of the variable and $s \in S$ is the sort of the variable and that two different variables in $X$ have different names. Variables $(x, s, (S, F, P))$
are abbreviated by their name \(x\) or by their name and sort qualification like in \((x : s)\). Then let \((S, F + X, P)\) be the extension of \((S, F, P)\) such that \((F + X)_{ar \rightarrow s} = F_{ar \rightarrow s}\) when \(ar\) is non-empty and \((F + X)_{s \rightarrow s} = F_{s \rightarrow s} \cup \{(x, s, (S, F, P)) \mid (x, s, (S, F, P)) \in X\}\) and we let \(\varphi' : (S, F + X, P) \rightarrow (S', F' + X^\varphi, P')\) be the canonical extension of \(\varphi\) that maps each variable \((x, s, (S, F, P))\) to \((x, \varphi(s), (S', F', P'))\).

To simplify notation, instead of \((S, F + X, P)\) as above one may also write \((S, F, P) + X\) and when \(X\) is a singleton, i.e. \(X = \{x\}\), simply write \(x\) instead of \(X\). We may also extend these conventions to other institutions.

Sentence translations along a signature morphism \(\varphi : (S, F, P) \rightarrow (S', F', P')\), i.e., \(Sen_{FOL}(\varphi) : Sen_{FOL}(S, F, P) \rightarrow Sen_{FOL}(S', F', P')\), replaces symbols of \((S, F, P)\) by the respective \(\varphi\)-images in \((S', F', P')\). More precisely, \(\varphi^{\text{trm}} : T_{(S, F)} \rightarrow T_{(S', F')}\) is defined by

\[\varphi^{\text{trm}}(f(t_1, \ldots, t_n)) = \varphi^{\text{op}}(f)(\varphi^{\text{trm}}(t_1), \ldots, \varphi^{\text{trm}}(t_n)).\]

Then,

- \(\varphi^{FOL}(\varphi)(t \approx t') = \varphi^{\text{trm}}(t) \approx \varphi^{\text{trm}}(t')\);
- \(\varphi^{FOL}(\varphi)(\pi(t_1, \ldots, t_n)) = \varphi^{\text{rl}}(\pi)(\varphi^{\text{trm}}(t_1), \ldots, \varphi^{\text{trm}}(t_n))\);
- \(\varphi^{FOL}(\varphi)(\neg \rho) = \neg \varphi^{FOL}(\varphi)(\rho)\);
- \(\varphi^{FOL}(\varphi)(\rho \circ \rho') = \varphi^{FOL}(\varphi)(\rho) \circ \varphi^{FOL}(\varphi)(\rho')\), where \(\circ \in \{\lor, \land, \rightarrow\}\);
- \(\varphi^{FOL}(\varphi)(\forall X \rho) = \forall X^\varphi \varphi^{FOL}(\varphi)(\rho)\), where \(X^\varphi = \{(x, \varphi^{\text{st}}(s), (S', F', P')) | (x, s, (S, F, P))\}\), and \(\varphi'\) canonically extends \(\varphi\) by mapping each \((x, s, (S, F, P))\) to \((x, \varphi^{\text{st}}(s), (S', F', P'))\).

\[
\begin{array}{ccc}
(S, F, P) & \xrightarrow{\varphi} & (S, F', P') \\
\downarrow & & \downarrow \\
(S, F \cup X, P) & \xrightarrow{\varphi'} & (S, F' \cup X^\varphi, P')
\end{array}
\]

The satisfaction of sentences by models is the usual Tarskian satisfaction defined recursively on the structure of the sentences as follows:
• $M \models_{(S,F,P)} t = t'$ when $M_t = M_{t'}$, where $M_t$ denotes the interpretation of the $(S,F)$-term $t$ in $M$ defined recursively by $M_\sigma(t_1, \ldots, t_n) = M_\sigma(M_{t_1}, \ldots, M_{t_n})$.

• $M \models_{(S,F,P)} \pi(t_1, \ldots, t_n)$ when $(M_{t_1}, \ldots, M_{t_n}) \in M_\pi$, for each relational atom $\pi(t_1, \ldots, t_n)$.

• $M \models_{(S,F,P)} \rho_1 \land \rho_2$ when $M \models_{(S,F,P)} \rho_1$ and $M \models_{(S,F,P)} \rho_2$, and similarly for the other Boolean connectives.

• $M \models_{(S,F,P)} (\forall X) \rho$ when $M' \models_{(S,F+X,P)} \rho$ for any $(S,F+X,P)$-expansion $M'$ of $M$, and similarly for $\exists$.

Example 2.3.3 (ALG and EQ) The institution ALG is obtained by FOL by discarding the relational symbols and the corresponding interpretations in models. The institution EQ is defined as the sub-institution of ALG where the sentences are universally quantified equations $(\forall X) t = t'$.

Example 2.3.4 (REL) The institution REL is the sub-institution of single-sorted first-order logic with signatures having only constants and predicates symbols.

Example 2.3.5 (PL) The institution PL (of propositional logic) is the fragment of FOL determined by signatures with empty set of sort symbols: Signatures are of form $(\emptyset, \emptyset, \text{Prop})$, where Prop is a set of (0-ary predicate) symbols called propositions. We denote propositional signatures $(\emptyset, \emptyset, \text{Prop})$ simply by Prop. The signature morphisms are functions between sets of symbols and hence, Sign$^{PL}$ coincides with Set. Models of Prop are sets $X$ such that $X \subseteq \text{Prop}$ or, equivalently, functions $M : \text{Prop} \to \{\top, \bot\}$. Sentences are the usual propositional logic formulæ

$$\rho := \text{Prop} \mid \rho \lor \rho \mid \rho \land \rho \mid \rho \Rightarrow \rho \mid \neg \rho$$

The satisfaction relation is defined as usual:

• $M \models^{PL}_{\text{Prop}} \rho$ iff $M(\rho) = \top$, for any $\rho \in \text{Prop}$;

• $M \models^{PL}_{\text{Prop}} \rho \lor \rho'$ iff, $M \models^{PL}_{\text{Prop}} \rho$ or $M \models^{PL}_{\text{Prop}} \rho'$,
and similarly for the other connectives.

**Example 2.3.6 ((Propositional) hybrid logic)** The *propositional modal logic* and *the propositional hybrid logic* mentioned in Section 2.1 constitutes two institutions. Their complete definition can be consulted in Example 3.2.1 as particular instances of the application of the *hybridisation method* discussed in Section 3.1.

**Example 2.3.7 (PA)** Consider now the institution *PA*, of partial algebras, which underlies the specification language CASL [ABK+02].

A *partial* algebraic signature is a tuple \((S, \overline{F}, \overline{P})\), where \(\overline{F}\) is a family of sets of *total* function symbols and \(\overline{P}\) is a family of sets of *partial* function symbols such that \(\overline{F}_{ar\rightarrow s} \cap \overline{P}_{ar\rightarrow s} = \emptyset\) for each arity \(ar\) and each sort \(s\). In order to avoid empty carriers, as in the case of *FOL*, we assume there exists at least one *total* constant for each sort. Signature morphisms map the three components in a compatible way.

A partial algebra is an ordinary algebra (i.e. a *FOL* model without relations) in which the function symbols in \(\overline{P}\) are interpreted as partial, rather than total, functions. For any \(\sigma \in \overline{F}_{ar\rightarrow s}\), \(\text{dom}(A_\sigma) = \{a \in A_{ar} \mid A_\sigma(a) \text{ defined}\}\) is the domain of the operation \(\sigma\). A partial algebra homomorphism \(h : A \rightarrow B\) is a family of (total) functions \(\{h_s : A_s \rightarrow B_s \mid s \in S\}\), indexed by the set of sorts \(S\) in the signature, such that \(h_s(A_\sigma(a)) = B_\sigma(h_{ar}(a))\) for each function symbol \(\sigma \in \overline{F}_{ar\rightarrow s} \cup \overline{P}_{ar\rightarrow s}\) and each tuple of arguments \(a \in A_{ar}\) for which \(A_\sigma(a)\) is defined.

Sentences have three kinds of atoms: *definedness* \(\text{df}(t)\), *strong equality* \(t = t'\), and *existential equality* \(t \models t'\). They are formed from these atoms by Boolean connectives and quantification over total variables (i.e variables that are always defined). For satisfaction, we have that:

- \(A \models_{(S,\overline{F},\overline{P})} \text{df}(t)\) iff \(A_t\) is defined;
- \(A \models_{(S,\overline{F},\overline{P})} t = t'\) iff either both are defined and \(A_t = A_{t'}\) or both are undefined;
- \(A \models_{(S,\overline{F},\overline{P})} t \models t'\) iff \(A_t\) and \(A_{t'}\) are defined and \(A_t = A_{t'}\).

The satisfaction relation is defined for the composition with boolean connectives and quantifiers as is Example 2.3.2. Notice that \(\text{df}(t)\)
is equivalent to \( t \equiv t \) and that \( t = t' \) is equivalent to \((t \equiv t') \lor (\neg \text{df}(t) \land \neg \text{df}(t'))\).

---

**Example 2.3.8 (Multi-valued and Fuzzy Logics)**  Multi-valued logics replace the two-elements set of true values \{true, false\}, structured as a Boolean algebra, by other sets structured as *complete residuated lattices* (cf. [Got01] for an overview).

Multi-valued logics were first formalised as institutions in [ACEGG90], being [Dia11] a recent reference. We follow the latter in the presentation below.

A *residuated lattice* is a structure \( L = (L, \leq, \land, \lor, \top, \bot, \star) \), where

- \((L, \land, \lor, \top, \bot)\) is a lattice ordered by \( \leq \), with support \( L \), with (binary) infimum and supremum, \( \land \) and \( \lor \), and biggest and smallest elements, \( \top \) and \( \bot \);
- \( \star \) is an associative and commutative binary operation such that, for any elements \( x, y, z \in L \):
  - \( x \star \top = \top \star x = x \);
  - \( y \leq z \) implies that \((x \star y) \leq (x \star z)\);
  - there exists an element \( x \Rightarrow z \) such that \( y \leq (x \Rightarrow z) \) iff \( x \star y \leq z \).

The residuated lattice \( L \) is complete if any subset \( S \subseteq L \) has infimum and supremum denoted by \( \land S \) and \( \lor S \), respectively.

Given a complete residuated lattice \( L \), we define the institution \( MVL_L \) as follows.

- \( \text{Sign}^{MVL_L} = \text{Sign}^{\text{FOL}} \);
- Sentences of \( \text{Sen}^{MVL_L}(S, F, P) \) are pairs \((\rho, p)\) where
  - \( p \) is an element of \( L \), and
  - \( \rho \) is generated from relational atoms of \( (S, F, P) \), i.e.
    expressions \( \pi(t_1, \ldots, t_n) \), for \( \pi \in P_{s_1 \ldots s_n} \) and \( t_i \) an algebraic term of sort \( s_i \), with the (extended) set of connectives \( \{ \Rightarrow, \land, \lor, \top, \bot, \star \} \) and quantifications \((\forall X)\) and \((\exists X)\) for \( X \) a finite set of variables.
• a model $M \in \text{Mod}^{MVL_L}(S, F, P)$ consists
  – of an algebra $(S, F)$,
  – for each $\pi \in P_{\text{ar}}$, of a function $M_\pi : M_{\text{ar}} \to L$.

Morphisms between models $M$ and $N$ are algebra homomorphisms such that for any $\pi \in R_{\text{ar}}$, $M_\pi(m) \leq N_\pi(h_{\text{ar}}(m))$.

• For any $M \in \text{Mod}^{MVL_L}(S, F, P)$ and for any $(\rho, p) \in \text{Sen}^{MVL_L}(S, F, P)$ the satisfaction relation is given by

$$M \models^{MVL_L}_{(S,F,P)} (\rho, p) \text{ iff } p \leq (M \models \rho)$$

where $M \models \rho$ is inductively defined as follows:

- for any relational atom $\pi(t_1, \ldots, t_n)$, $(M \models \pi(t_1, \ldots, t_n)) = M_\pi(M_{t_1}, \ldots, M_{t_n})$;
- $(M \models T) = T$;
- $(M \models \bot) = \bot$;
- $(M \models \rho_1 \circ \rho_2) = (M \models \rho_1) \circ (M \models \rho_2)$, for $\circ \in \{\land, \lor, \Rightarrow, \star\}$;
- $(M \models (\forall X)\rho) = \bigwedge \{ M' \models \rho | M'|_{(S,F,P)} = M \}$;
- $(M \models (\exists X)\rho) = \bigvee \{ M' \models \rho | M'|_{(S,F,P)} = M \}$.

This institution captures a number of multi-valued logics in the literature. For instance, taking $L$ as the Łukasiewicz arithmetic lattice over the closed interval $[0, 1]$, where $x \star y = 1 - \max\{0, x + y - 1\}$ (and $x \Rightarrow y = \min\{1, 1 - x + y\}$), corresponds to the standard predicate fuzzy logic. On the other hand, considering signatures with the empty set of sorts, results in propositional fuzzy logic (e.g. [NW05]). Note that the 2-boolean truth values with the usual disjunction and $\land$ as the usual conjunction constitutes a residuated lattice $B$. In this case, $MVL_B$ coincides with $FOL$ without equality.

2.3.3 Internal logic

**Definition 2.3.2 (Internal logic, [DS07])** An institution $\mathcal{I}$ has (semantic) conjunctions when for each signature $\Sigma$ and any $\Sigma$-sentences $e_1$
and $e_2$ there exists a $\Sigma$-sentence $e$ such that $e^* = e_1^* \cap e_2^*$. Usually $e$ is denoted by $e_1 \land e_2$.

$I$ has (semantic) implications when for each $e_1$ and $e_2$ as above there exists $e$ such that $e^* = (\text{Mod}(\Sigma) - e_1^*) \cup e_2^*$. Usually $e$ is denoted $e_1 \Rightarrow e_2$.

$I$ has (semantic) existential $D$-quantifications for a class $D$ of signature morphisms, if for each $\chi : \Sigma \to \Sigma' \in D$ and $\Sigma'$-sentence $e'$ there exists a $\Sigma$-sentence $e$ such that $e^* = \text{Mod}(\chi)(e'^*)$. Usually $e$ is denoted by $(\exists \chi)e'$.

In the same style we may extend this list to other Boolean connectives disjunction ($\lor$), negation ($\neg$), equivalence ($\equiv$) and semantic universal quantifications ($(\forall \chi)e'$).

Note, for instance that $\text{FOL}$ has all of the mentioned semantical connectives and quantifications, $\text{PL}$ all the connectives. Moreover, $\text{MVL}_L$ has and $\text{EQ}$ has no conjuction.

2.3.4 Amalgamation

The notion of amalgamation plays a central role on the theory of institutions:

**Definition 2.3.3 (Amalgamation property)** A commuting square of functors

\[
\begin{array}{c}
A \\
\downarrow F_2 \\
A_2
\end{array}
\quad
\begin{array}{c}
A_1 \\
\downarrow G_1 \\
A'
\end{array}
\]

is a weak amalgamation square if and only if; for each $M_1 \in |A_1|$ and $M_2 \in |A_2|$ such that $F_1(M_1) = F_2(M_2)$, there exists a $M' \in |A'|$ such that $G_1(M') = M_1$ and $G_2(M') = M_2$. When $M'$ is required to be unique, the square is called amalgamation square. The object $M'$ is called an amalgamation of $M_1$ and $M_2$ and, if unique, denoted by $M_1 \otimes_{F_1,F_2} M_2$. 
For any functor $\text{Mod} \colon \text{Sign}^{\text{op}} \to \text{CAT}$ a commuting square of signature morphisms

$$\begin{array}{ccc}
\Sigma & \xrightarrow{\varphi_1} & \Sigma_1 \\
\downarrow{\varphi_2} & & \downarrow{\theta_1} \\
\Sigma_2 & \xrightarrow{\theta_2} & \Sigma'
\end{array}$$  \hspace{1cm} (4)

is a (weak) amalgamation square for $\text{Mod}$ when

$$\begin{array}{ccc}
\text{Mod}(\Sigma) & \xrightarrow{\text{Mod}(\varphi_1)} & \text{Mod}(\Sigma_1) \\
\text{Mod}(\varphi_2) & \downarrow{\text{Mod}(\theta_1)} & \downarrow{\text{Mod}(\theta_2)} \\
\text{Mod}(\Sigma_2) & \xleftarrow{\text{Mod}(\theta_2)} & \text{Mod}(\Sigma')
\end{array}$$  \hspace{1cm} (5)

is a (weak) amalgamation square.

We say that an institution $I$ has the (weak) amalgamation property when each pushout square of signature morphisms is a (weak) amalgamation square for the model functor $\text{Mod}^I$.

The existence of amalgamation is a very important, albeit common, institutional property. In particular, all the institutions discussed above have amalgamation (cf. [Dia08]).

This sub-section closes with two technical definitions used in the sequel:

**Definition 2.3.4** A sub-functor $\text{Mod}' \subseteq \text{Mod} : \text{Sign}^{\text{op}} \to \text{CAT}$ reflects (weak) amalgamation when each pushout square in $\text{Sign}$ that is a (weak) amalgamation square for $\text{Mod}$ is a (weak) amalgamation square for $\text{Mod}'$ too.

**Definition 2.3.5** For any functors $\text{Mod}_1, \text{Mod}_2 : \text{Sign} \to \text{CAT}^{\text{op}}$ and any natural transformation $\beta : \text{Mod}_2 \to \text{Mod}_1$ we say that $(\chi : \Sigma \to \Sigma') \in \text{Sign}$ is adequate for $\beta$ if and only if the following square is a weak amalgamation square:

$$\begin{array}{ccc}
\text{Mod}_1(\Sigma) & \xrightarrow{\beta_\Sigma} & \text{Mod}_2(\Sigma) \\
\downarrow{\text{Mod}_1(\chi)} & & \downarrow{\text{Mod}_2(\chi)} \\
\text{Mod}_1(\Sigma') & \xleftarrow{\beta_{\Sigma'}} & \text{Mod}_2(\Sigma')
\end{array}$$
From the rather straightforward fact that the glueing of (weak) amalgamation squares yields a (weak) amalgamation square we obtain that:

**Fact 2.3.1** Adequate morphisms for \( \beta \) are closed under composition.

### 2.3.5 Quantification spaces

While first-order quantification is well known, other less standard patterns of quantification arise in the literature, for example second-order and infinitary quantifications. However, to understand abstractly the essence of what a quantification is at an institution-independent level is a challenging issue.

In the present thesis, as in our previous work [MMDB11a, DM], we approach this issue by means of *quantification spaces*, a concept recently introduced in [Dia10a]. Just to motivate the approach, let us revisit the Example 2.3.2, on which variables are dealt through the *Lemma on constants of first-order logic* (e.g. [Hod93]). This means that a set of variables \( X \) is regarded as a set of constants (of an \( X \)-expanded signature), and

\[
- M \models_{FOL(S,F,P)} (\forall X) \rho \quad \text{when} \quad M' \models_{FOL(S,F+X,P)} (\forall X) \rho \quad \text{for any } (S,F+X,P)-\text{expansion } M' \text{ of } M.
\]

A generalisation of this pattern can be achieved if we consider a morphism \( \chi : \Sigma \to \Sigma' \) to play the role of the inclusion morphism \( (S,F,P) \hookrightarrow (S,F+X,P) \). Hence, an universal quantification in an arbitrary institution \( I \) could be formalised as

\[
- M \modelsI (\forall \chi) \rho \quad \text{when} \quad M' \modelsI (\forall \chi) \rho \quad \text{for any } \chi\text{-expansion } M' \text{ of } M,
\]

i.e., for any \( M' \in \text{Mod}_I(\Sigma') \) such that \( \text{Mod}_I(\chi)(M') = M \).

Moreover, and in order to enforce the satisfaction condition (and the functoriality of \( \text{Sen}_I^{\Sigma} \)), we have to study the transformation of quantification spaces under signature morphisms. In the *FOL* example we have

\[
\text{Sen}_{FOL}(\phi)(\forall X \rho) = (\forall X^{\phi}) \text{Sen}_{FOL}(\phi')(\rho),
\]

where we take \( X^{\phi} \) and \( \phi' \) to defining the following pushout:

\[
\begin{array}{ccc}
(S,F,P) & \xrightarrow{\phi} & (S,F',P') \\
\downarrow{\chi} & & \downarrow{\chi^{\phi}} \\
(S,F \uplus X,P) & \xrightarrow{\phi'} & (S,F' \uplus X^{\phi},P')
\end{array}
\]
Thus, $X^\phi = \{(x, \varphi_{st}(s), (S', F', P'))| (x, s, (S, F, P)) \in X\}$, and $\varphi'$ is the canonical extension of $\varphi$, mapping each $(x, s, (S, F, P))$ to $(x, \varphi_{st}(s), (S', F', P'))$. This construction is abstracted in the following definition as a designated pushout:

**Definition 2.3.6 (Quantification space)** For any category $\text{Sign}$ a subclass of arrows $\mathcal{D} \subseteq \text{Sign}$ is called a quantification space if, for any $(\chi : \Sigma \to \Sigma') \in \mathcal{D}$ and $\varphi : \Sigma \to \Sigma_1$, there is a designated pushout

$$
\begin{array}{ccc}
\Sigma & \xrightarrow{\varphi} & \Sigma_1 \\
\downarrow{\chi} & & \downarrow{\chi(\varphi)} \\
\Sigma' & \xrightarrow{\varphi[\chi]} & \Sigma_1'
\end{array}
$$

(6)

with $\chi(\varphi) \in \mathcal{D}$ and such that the ‘horizontal’ composition of such designated pushouts is again a designated pushout, i.e. for the pushouts in the following diagram

$$
\begin{array}{ccc}
\Sigma & \xrightarrow{\varphi} & \Sigma_1 \xrightarrow{\theta} \Sigma_2 \\
\downarrow{\chi} & & \downarrow{\chi(\varphi)(\theta)} \\
\Sigma' & \xleftarrow{\varphi[\chi]} & \Sigma_1' \xleftarrow{\theta[\chi(\varphi)]} \Sigma_2'
\end{array}
$$

(7)

$\varphi[\chi]; \theta[\chi(\varphi)] = (\varphi; \theta)[\chi]$ and $\chi(\varphi)(\theta) = \chi(\varphi; \theta)$, and such that $\chi(1\Sigma) = \chi$ and $1_{\Sigma}[\chi] = 1_{\Sigma'}$.

We say that a quantification space $\mathcal{D}$ for $\text{Sign}$ is adequate for a functor $\text{Mod} : \text{Sign}^{\text{op}} \to \text{CAT}$ when the designated pushouts mentioned above are weak amalgamation squares for $\text{Mod}$.

**Example 2.3.9 (Standard quantification for FOL)** $X$-expansions of FOL-signatures, i.e., morphisms $\chi : (S, F, P) \to (S, F + X, P)$, constitute adequate quantification spaces for $\text{Mod}^{\text{FOL}}$. In this case, for each $\varphi : (S, F, P) \to (S', F', P')$, we take as designated pushout the $X^\varphi$-expansions morphisms, and the respective $\varphi'$, as defined above. It is easy to note that these define pushout squares fulfilling the properties of Definition 2.3.6, and the adequacy for $\text{Mod}^{\text{FOL}}$ follows from the fact that $\text{Mod}^{\text{FOL}}$ preserves all finite limits (cf. [Dia08]).

**Example 2.3.10 (Infinitary quantification for FOL)** Another adequate quantification space for $\text{Mod}^{\text{FOL}}$ considers infinite blocks of variables instead of finite ones. For each infinite set of variables $X$, we take $X^\varphi$ and $\varphi'$ as in the previous example.
Example 2.3.11 (Standard quantification for $PA$) Similarly, we may take $X$-expansions of total constants in $PA$-signatures, i.e., morphisms $\chi : (S, TF, PF) \hookrightarrow (S, TF + X, PF)$, as adequate quantification spaces for $PA$.

Example 2.3.12 (Taking SOL from FOL) To obtain second-order logic, we quantify the (quantified-free version of) $FOL$ with blocks of second order variables, i.e., blocks of the form $(x, (w, s), (S, F, P))$ (function variables) or of the form $(x, w, (S, F, P))$ (relation variables) where $ar \in S^*$ and $s \in S$. Then, to any block $X$ of second order variables it corresponds a signature extension $\chi : (S, F, P) \rightarrow (S, F + X^{op}, P + X^{rl})$ where $X$ is split as $X^{op} \cup X^{rl}$, with $X^{op}$ the function variables and $X^{rl}$ the relation variables, and where $F + X^{op}$ and $P + X^{rl}$ extend in the obvious way the definition of $F + X$ from Example 2.3.2.

Note that these definitions also apply to $REL$, $EQ$, $MLV_L$ and $ALG$.

2.3.6 Comorphisms

As stated before, the ability to relate logics is a main topic in the theory of institutions. There are two main kinds of structure preserving mappings between institutions: morphisms [GB92] and comorphisms [GR02, Mes89, Tar98]. The former are adequate for expressing ‘forgetful’ operations from a ‘more complex’ institution to a structurally ‘simpler’ one. The later realises the intuition of ‘embedding’ a ‘simpler’ institution into a ‘more complex’ one. In this thesis a heavy use is made of comorphism.

Definition 2.3.7 (Comorphism) An institution comorphism

$$(\Phi, \alpha, \beta) : \mathcal{I} \rightarrow \mathcal{I}'$$

consists of

1. a functor $\Phi : \text{Sign}^{\mathcal{I}} \rightarrow \text{Sign}^{\mathcal{I}'}$,

2. a natural transformation $\alpha : \text{Sen}^{\mathcal{I}} \Rightarrow \Phi ; \text{Sen}^{\mathcal{I}'}$, and

3. a natural transformation $\beta : \Phi^{op} ; \text{Mod}^{\mathcal{I}'} \Rightarrow \text{Mod}^{\mathcal{I}}$
such that the following satisfaction condition holds
\[ M' \models_{\Phi(\Sigma)} \alpha_{\Sigma}(e) \iff \beta_{\Sigma}(M') \models_{\Sigma} \rho \]
for each signature $\Sigma \in \langle \text{Sign} \rangle$, for each $\Phi(\Sigma)$-model $M'$, and each $\Sigma$-sentence $e$.

Occasionally, for simplifying notation, we may denote a particular co-morphisms $(\Phi, \alpha, \beta) : I \to I'$ just by $I \to I'$.

**Theorem 2.3.1** Given a comorphism, for any set $\Gamma \subseteq \text{Sen}^I(\Sigma)$ and sentence $\rho \in \text{Sen}^I(\Sigma)$,
\[ \Gamma \models_{\Sigma} \rho \text{ implies } \alpha_{\Sigma}(\Gamma) \models_{\Phi(\Sigma)} \alpha_{\Sigma}(\rho). \]

**Proof.**
\[
\begin{align*}
\Gamma & \models_{\Sigma} \rho \\
\iff \quad & \{ \text{by definition} \} \\
& \text{for any } M \in \text{Mod}^I(\Sigma), \\
& M \models_{\Sigma} \Gamma \text{ implies } M \models_{\Sigma} \rho \\
\implies \quad & \{ \text{instantiation} \} \\
& \text{for any } M' \in \text{Mod}^{I'}(\Sigma'), \\
& \beta(M') \models_{\Sigma} \Gamma \text{ implies } \beta(M') \models_{\Sigma} \rho \\
\iff \quad & \{ \text{by hypothesis} \} \\
& \text{for any } M' \in \text{Mod}^{I'}(\Sigma'), \\
& M' \models_{\Phi(\Sigma)} \alpha(\Gamma) \text{ implies } M' \models_{\Phi(\Sigma)} \alpha(\rho) \\
\iff \quad & \{ \text{by definition} \} \\
& \alpha(\Gamma) \models_{\Phi(\Sigma)} \alpha(\rho)
\end{align*}
\]

Next definition singles out a class of comorphisms for which the converse of Theorem 2.3.1 also holds

**Definition 2.3.8 (Conservative comorphism)** Let $(\Phi, \alpha, \beta) : I \to I'$ be a comorphism. We say that $(\Phi, \alpha, \beta)$ is conservative whenever, for each $\Sigma$-model $M$ in $I$, there exists a $\Phi(\Sigma)$-model $M'$ in $I'$ such that $M = \beta_{\Sigma}(M')$. 

Conservativeness is a very useful property since, when it holds, results about the source institution may be proven by the proof system of the target institution in a sound and complete way:

**Theorem 2.3.2** Given a conservative comorphism, for any set \( \Gamma \subseteq \text{Sen}^I(\Sigma) \) and sentence \( \rho \in \text{Sen}^I(\Sigma) \),

\[
\Gamma \vDash^I \rho \text{ iff } \alpha_\Sigma(\Gamma) \vDash^I_{\Phi(\Sigma)} \alpha_\Sigma(\rho).
\]

**Proof.** The surjectivity of \( \beta \) entails the converse direction for the (second) \( \Rightarrow \)-implication step in the proof of Theorem 2.3.1. \( \square \)

**Example 2.3.13** (TRM2FOL) There is an obvious way to define a comorphism from TRM to any other institution \( J \). Consider, for example, the case of FOL. The functor \( \Phi \) maps the signature * into the FOL signature \((\emptyset, \emptyset, P^*)\), \( P^*_\emptyset = \{ * \} \) and \( P^*_{\text{ar}} = \emptyset \) for \( \text{ar} \neq \emptyset \). The natural transformation \( \beta \) maps each \( M \in \text{Mod}^{\text{FOL}}((\emptyset, \emptyset, P^*)) \) into the model \( * \in \text{Mod}^{\text{TRM}}(*) \) and \( \alpha \) is simply the empty function. Since there are no sentences, the satisfaction condition of the comorphism holds trivially. Moreover, since \( \beta \) is surjective this comorphism is conservative. \( \circ \)

**Example 2.3.14** (PL2FOL) The base comorphism \((\Phi, \alpha, \beta)\) is the canonical embedding of PL into FOL determined by the embedding of PL signatures into FOL signatures: for any propositional signature \( \text{Prop} \in \text{Sign}^{\text{PL}} \), \( \Phi(\text{Prop}) = (\emptyset, \emptyset, P) \), where \( P_{\emptyset} = \text{Prop} \) and \( P_{\text{ar}} = \emptyset \), for \( \text{ar} \neq \emptyset \). The translation of sentences is inductively defined by

- \( \alpha_{\text{Prop}}(p) = p \), for \( p \in \text{Prop} \), and
- \( \alpha_{\text{Prop}}(\rho \otimes \rho') = \alpha_{\text{Prop}}(\rho) \otimes \alpha_{\text{Prop}}(\rho') \) for any \( \otimes \in \{ \lor, \land, \Rightarrow \} \) and \( \alpha_{\text{Prop}}(\neg \rho) = \neg \alpha_{\text{Prop}}(\rho) \);

and, the translation of models \( \beta(M) \), for \( M \in \text{Mod}^{\text{FOL}}((\emptyset, \emptyset, \text{Prop})) \), is given by

- for any \( p \in \text{Prop} \), \( \beta(M)_p = M_p \);

The comorphism is obviously conservative. \( \circ \)
2.3.7 Presentations

We introduced, in this sub-section the notion of institution of presentations over a base institution $I$.

**Definition 2.3.9 (Presentations)** A presentation in $I$ is a pair $(\Sigma, E)$ consisting of a signature $\Sigma$ and a set $E$ of $\Sigma$-sentences. A presentation morphism $\varphi : (\Sigma, E) \to (\Sigma', E')$ is a signature morphism $\varphi : \Sigma \to \Sigma'$ such that $E' \models \text{Sign}_I(\varphi)(E)$.

**Fact 2.3.2** Presentation morphisms are closed under composition given by composition of the signature morphisms.

This fact paves the way for the general construction given in the following definition.

**Definition 2.3.10 (The institution of presentations)** Let $I = (\text{Sign}, \text{Sen}, \text{Mod}, \models)$ be an institution. The institution of presentations of $I$, denoted by \( I^{\text{pres}} = (\text{Sign}^{\text{pres}}, \text{Sen}^{\text{pres}}, \text{Mod}^{\text{pres}}, \models^{\text{pres}}) \), is defined by

- $\text{Sign}^{\text{pres}}$ is the category $\text{Pres}$ of presentations of $I$,
- $\text{Sen}^{\text{pres}}(\Sigma, E) = \text{Sen}(\Sigma)$,
- $\text{Mod}^{\text{pres}}(\Sigma, E)$ is the full subcategory of $\text{Mod}(\Sigma)$ of those models which satisfy $E$, and
- for each $(\Sigma, E)$-model $M$ and $\Sigma$-sentence $e$, $M \models_{\text{pres}} e$ if and only if $M \models_{\Sigma} e$.

**Fact 2.3.3** For any institution $I$, $I^{\text{pres}}$ is also an institution.

Comorphisms into this special kind of institutions are particularly useful: rather than expressing an embedding between institutions, they can be used for ‘encoding’ a ‘more complex’ institution $I$ into a ‘simpler’ one $I'$. In such encodings the structural complexity cost is shifted to the mapping $\Phi$ on the signatures. This is especially useful for borrowing methods from the target institution to the source one (e.g. [Dia08, Dia12a, Dia12b, DP06]). This kind of encodings are sometimes called “theoroidal comorphisms” [Mos96, GR02]. There are
many well known illustrations of encodings $J \rightarrow J^{\text{pres}}$ in the literature, many of them mentioned in [Dia08]; reference [Mos02] reports an exhaustive investigation on encodings into the institution CASL.

**Example 2.3.15 (PA2FOL$^{\text{pres}}$)** Literature reports a number of possible encodings of PA to FOL$^{\text{pres}}$. A classical example is the *relational encoding* where partial functions are encoded into first-order relations (e.g. [Dia08, Mos96]). More recently [Dia09] introduced the "quasi-boolean encoding". We present in the sequel an operational encoding of PA into FOL$^{\text{pres}}$, where partial operations are encoded as total ones. In all of them comorphisms preserve different model theoretic properties, as discussed in [DM].

The comorphism $(\Phi, \alpha, \beta) : PA \rightarrow FOL^{\text{pres}}$ is defined as follows:

1. $\Phi$ maps each signature $(S, TF, PF)$ to the presentation
   $$( (S, TF + PF, (\text{Def}_s)_{s \in S}), \Gamma_{(S,TF,PF)} )$$
   where $\Gamma_{(S,TF,PF)}$ axiomatizes the definability of terms through the new predicates $(\text{Def}_s)_{s \in S}$ as follows:
   $$\Gamma_{(S,TF,PF)} = \{(\forall X)\text{Def}_s(X) \Rightarrow \text{Def}_s(\text{Def}_s(X)) \mid \sigma \in (TF + PF)_{ar \mapsto s}, ar \in S^*, s \in S\} \cup$$
   $$\{(\forall X)\text{Def}_s(X) \Rightarrow \text{Def}_s(\text{Def}_s(X)) \mid \sigma \in TF_{ar \mapsto s}, ar \in S^*, s \in S\}$$
   (where $\text{Def}_s(X)$ denotes $\bigwedge_{x \in X}(\text{Def}_s(x))$).

2. $\alpha_{(S,TF,PF)}$ is recursively defined as follows:
   - $\alpha(t \equiv t') = \text{Def}_s(t) \land (t = t')$;
   - $\alpha((\forall X)\rho) = (\forall X)(\text{Def}_s(X) \Rightarrow \alpha(\rho))$;
   - $\alpha$ commutes with boolean connectives $\land, \lor, \Rightarrow$, etc.

3. $\beta_{(S,TF,PF)}$ maps any $((S, TF + PF, (\text{Def}_s)_{s \in S}), \Gamma_{(S,TF,PF)})$ model $M$ to the partial algebra $\beta(M)$ where:
   - for any $s \in S$, $\beta(M)_s = M_{\text{Def}_s}$;
   - for any $\sigma \in TF_{ar \mapsto s}$, $\beta(M)_\sigma = M_\sigma$;
   - for any $\sigma \in PF_{ar \mapsto s}$, $\beta(M)_\sigma$ consists of the restriction of $M_\sigma$ to $M_{\text{Def}_s}$ such that $\text{dom}(\beta(M)_\sigma) = \{x \in M_{\text{Def}_s} \mid M_\sigma(x) \in M_{\text{Def}_s}\}$. 

$\diamondsuit$
This chapter details the first original contribution of the thesis: an institution-independent method to hybridise arbitrary logics. The method is not only a technical construction formulated in an institutional, thus generic, setting. It also sheds light on the generic pattern of hybridisation and, offers a “source of logics” useful for specifying software reconfigurability, as discussed in the second part of the thesis. Most of results reported here appeared in [MMDB11a] and [DM].

The starting point for the development of the method was centred in the observation that the “modalisation” of a logic can be done at an institution-independent level [DS07], and on the experimental demand for hybrid extensions of modal logics.

Quoting [AB01],

“(...)Strictly speaking, not all modal logics are hybrid, but certainly any modal logics can be hybridised, and in our view many of them should be (...)”

Concretely, the proposed method extends to arbitrary institutions Kripke semantics, for multi-(polyadic)-modalities, as well as nominals and local satisfaction operators. Technically, given a base institution \( I \) the method builds an institution \( HI \) corresponding to its hybridisation. It takes as a parameter the quantization scheme intended (beyond the quantification of the base institution, we may also consider quantifications over the nominals and modalities).

At the end of the Section 3.1 we mentioned the concept of constrained model recently introduced by R. Diaconescu in [Dia13]. This notion, that can be taken as a third parameter of the method, plays a crucial role on the remaining of the thesis, namely in the Chapter 4.

The remaining of the Chapter is organised as follows: we start with the exposition of the method in Section 3.1. The construction of the hybridisation of an institution is presented and the section closes with the proof of the Satisfaction Condition for the hybridised institution.
Then, a brief discussion about how connectives are inherited from the base institution is made in Section 3.1.5. Finally, a number of examples of hybridisations are discussed in Section 3.2.

3.1 THE HYBRIDISATION METHOD

Let us consider an institution \( \mathcal{I} = (\text{Sign}^\mathcal{I}, \text{Sen}^\mathcal{I}, \text{Mod}^\mathcal{I}, (\models^\mathcal{I}_\Sigma)_{\Sigma \subseteq \text{Sign}^\mathcal{I}}) \) with a designated quantification space \( \mathcal{D}^\mathcal{I} \subseteq \text{Sign}^\mathcal{I} \). This will be referred to as the base institution. Below we introduce a method to enrich \( \mathcal{I} \) with modalities and nominals, and define a suitable semantics for this enrichment. Moreover, it is shown that the outcome still defines an institution called the hybridisation of \( \mathcal{I} \).

3.1.1 The category of \( \mathcal{I} \)-hybrid signatures

The category of \( \mathcal{I} \)-hybrid signatures, denoted by \( \text{Sign}^{I\mathcal{I}} \), is defined as the following direct (Cartesian) product of categories:

\[
\text{Sign}^{I\mathcal{I}} = \text{Sign}^\mathcal{I} \times \text{Sign}^{REL}
\]

The REL-signatures are denoted by \((\text{Nom}, \Lambda)\), where \(\text{Nom}\) is a set of constants called nominals and \(\Lambda\) is a \(\mathbb{N}\)-family of sets of relational symbols called modalities.

General category theory entails:

**Proposition 3.1.1** \(\text{Sign}^{REL}\) has small co-limits.

**Proof.** As \(\text{Sign}^{REL}\) is isomorphic to the power category \(\text{Set}^{\mathbb{N}}\). As \(\text{Set}\) has small co-limits it follows that \(\text{Sign}^{REL}\) has small co-limits. \(\Box\)

**Corollary 3.1.1** The projection \(\text{Sign}^{I\mathcal{I}} \to \text{Sign}^\mathcal{I}\) lifts small co-limits.

**Proof.** Let us suppose that \(C\) is a small co-limit in \(\text{Sign}^\mathcal{I}\). Then, take a small colimit \(C'\) in \(\text{Sign}^{REL}\) over the underlying diagram of \(C\). Since the pair \((C, C')\) is a small co-limit in \(\text{Sign}^\mathcal{I} \times \text{Sign}^{REL} = \text{Sign}^{I\mathcal{I}}\), the projection \(\text{the proper} C\) is also a small co-limit. \(\Box\)
3.1.2 \( \mathcal{H}3 \)-sentences functor

Let us fix a quantification space \( \mathcal{D}^{3\mathcal{C}} \) for \( \text{Sign}^{3\mathcal{C}} \) such that for each \( \chi \in \mathcal{D}^{3\mathcal{C}} \) its projection \( \chi_{\text{Sign}} \) to \( \text{Sign}^{3} \) belongs to \( \mathcal{D}^{3} \). The quantification space \( \mathcal{D}^{3\mathcal{C}} \) is a parameter of the hybridisation process. Whenever \( \mathcal{D}^{3\mathcal{C}} \) consists of identities we say the hybridisation is quantifier-free. Note that a quantifier-free hybridisation does not necessarily imply the absence of ‘local’ quantification, i.e., quantification placed at the level of base institution \( \mathcal{I} \).

Let \( \Delta = (\Sigma, \text{Nom}, \Lambda) \). The set of sentences \( \text{Sen}^{3\mathcal{C}}(\Delta) \) is the least set such that

- \( \text{Nom} \subseteq \text{Sen}^{3\mathcal{C}}(\Delta) \);
- \( \text{Sen}^{3}(\Sigma) \subseteq \text{Sen}^{3\mathcal{C}}(\Delta) \);
- \( \rho \ast \rho' \in \text{Sen}^{3\mathcal{C}}(\Delta) \) for any \( \rho, \rho' \in \text{Sen}^{3\mathcal{C}}(\Delta) \) and any \( \ast \in \{\lor, \land, \Rightarrow\} \),
- \( \neg \rho \in \text{Sen}^{3\mathcal{C}}(\Delta) \), for any \( \rho \in \text{Sen}^{3\mathcal{C}}(\Delta) \),
- \( @_i \rho \in \text{Sen}^{3\mathcal{C}}(\Delta) \) for any \( \rho \in \text{Sen}^{3\mathcal{C}}(\Delta) \) and \( i \in \text{Nom} \);
- \( \langle \lambda \rangle(\rho_1, \ldots, \rho_n) \in \text{Sen}^{3\mathcal{C}}(\Delta) \), for any \( \lambda \in \Lambda_{n+1}, \rho_i \in \text{Sen}^{3\mathcal{C}}(\Delta) \), \( i \in \{1, \ldots, n\} \);
- \( \langle \lambda \rangle(\rho_1, \ldots, \rho_n) \in \text{Sen}^{3\mathcal{C}}(\Delta) \), for any \( \lambda \in \Lambda_{n+1}, \rho_i \in \text{Sen}^{3\mathcal{C}}(\Delta) \), \( i \in \{1, \ldots, n\} \);
- \( (\forall \chi) \rho, (\exists \chi) \rho \in \text{Sen}^{3\mathcal{C}}(\Delta) \), for any \( \rho \in \text{Sen}^{3\mathcal{C}}(\Delta') \) and \( \chi : \Delta \to \Delta' \in \mathcal{D}^{3\mathcal{C}} \).

Whenever \( \chi \) is a simple extension with variables it can be abbreviated to quantifications by the corresponding variables. For example when \( \chi \) is an extension of \( (\Sigma, \text{Nom}, \Lambda) \) with a nominal variable \( i \), instead of \( (\forall \chi) \rho \) we write \( (\forall i) \rho \).

In some cases, it is important to represent explicitly the embedding of the sentences of the base institution into its hybridisation, i.e., the embedding \( \text{Sen}^{3}(\Sigma) \hookrightarrow \text{Sen}^{3\mathcal{C}}(\Sigma, \text{Nom}, \Lambda) \). Such is the case of recursive hybridisation (cf. Example 3.2.3) where, for example, it may not be clear how to identify if a sentence \( i \) comes from the base or from the hybridised level.
Translations of $\mathcal{H}I$-sentences:

Let $\phi = (\varphi_{\text{Sign}}, \varphi_{\text{Nom}}, \varphi_{\text{MS}}) : (\Sigma, \text{Nom}, \Lambda) \to (\Sigma', \text{Nom}', \Lambda')$ be a morphim of $\mathcal{H}I$-signatures.

The translation $\text{Sen}^{\mathcal{H}I}(\phi)$ is defined as follows:

- $\text{Sen}^{\mathcal{H}I}(\phi)(\rho) = \text{Sen}^\mathcal{I}(\varphi_{\text{Sign}})(\rho)$ for any $\rho \in \text{Sen}^\mathcal{I}(\Sigma)$;
- $\text{Sen}^{\mathcal{H}I}(\phi)(i) = \varphi_{\text{Nom}}(i)$;
- $\text{Sen}^{\mathcal{H}I}(\phi)(\neg \rho) = \neg \text{Sen}^{\mathcal{H}I}(\phi)(\rho)$;
- $\text{Sen}^{\mathcal{H}I}(\phi)(\rho \ast \rho') = \text{Sen}^{\mathcal{H}I}(\phi)(\rho) \ast \text{Sen}^{\mathcal{H}I}(\phi)(\rho')$, $\ast \in \{\lor, \land, \Rightarrow\}$;
- $\text{Sen}^{\mathcal{H}I}(\phi)(\otimes_i \rho) = @_{\varphi_{\text{Nom}}(i)} \text{Sen}^{\mathcal{H}I}(\phi)(\rho)$;
- $\text{Sen}^{\mathcal{H}I}(\phi)(\lambda)(\rho_1, \ldots, \rho_n) = [\varphi_{\text{MS}}(\lambda)](\text{Sen}^{\mathcal{H}I}(\phi)(\rho_1), \ldots, \text{Sen}^{\mathcal{H}I}(\phi)(\rho_n))$;
- $\text{Sen}^{\mathcal{H}I}(\phi)(\lambda)(\rho_1, \ldots, \rho_n) = \langle \varphi_{\text{MS}}(\lambda) \rangle(\text{Sen}^{\mathcal{H}I}(\phi)(\rho_1), \ldots, \text{Sen}^{\mathcal{H}I}(\phi)(\rho_n))$;
- $\text{Sen}^{\mathcal{H}I}(\phi)((\forall \chi) \rho) = (\forall \chi(\phi)) \text{Sen}^{\mathcal{H}I}(\phi)(\chi)(\rho)$;
- $\text{Sen}^{\mathcal{H}I}(\phi)((\exists \chi) \rho) = (\exists \chi(\phi)) \text{Sen}^{\mathcal{H}I}(\phi)(\chi)(\rho)$;

Proposition 3.1.2 $\text{Sen}^{\mathcal{H}I}$ is a functor $\text{Sign}^{\mathcal{H}I} \to \text{Set}$.

Proof. The proof is by induction on the structure of sentences. For that, let us suppose the following signature morphisms in $\text{Sign}^{\mathcal{H}I}$:

$$(\Sigma, \text{Nom}, \Lambda) \xrightarrow{\varphi} (\Sigma', \text{Nom}', \Lambda') \xrightarrow{\theta} (\Sigma'', \text{Nom}'', \Lambda'')$$

Hence, for sentences $\rho \in \text{Sen}^\mathcal{I}(\Sigma)$ we have:

$$\text{Sen}^{\mathcal{H}I}(\phi); \text{Sen}^{\mathcal{H}I}(\theta)(\rho)$$

$$= \{ \text{defn. of composition} \}$$

$$\text{Sen}^{\mathcal{H}I}(\theta)(\text{Sen}^{\mathcal{H}I}(\phi)(\rho))$$

$$= \{ \text{defn of Sen}^{\mathcal{H}I} \text{ and } \rho \in \text{Sen}^\mathcal{I}(\Sigma) \}$$

$$\text{Sen}^{\mathcal{H}I}(\theta)(\text{Sen}^\mathcal{I}(\varphi_{\text{Sign}})(\rho))$$

$$= \{ \text{defn of Sen}^{\mathcal{H}I} \text{ and } \text{Sen}^\mathcal{I}(\varphi_{\text{Sign}})(\rho) \in \text{Sen}^\mathcal{I}(\Sigma') \}$$

$$\text{Sen}^\mathcal{I}(\theta_{\text{Sign}})(\text{Sen}^\mathcal{I}(\varphi_{\text{Sign}})(\rho))$$

$$= \{ \text{since Sen}^\mathcal{I} \text{ is a functor} \}$$

$$\text{Sen}^\mathcal{I}(\theta_{\text{Sign}}; \varphi_{\text{Sign}})(\rho)$$

$$= \{ \text{defn. of Sen}^{\mathcal{H}I} \text{ and } \rho \in \text{Sen}^\mathcal{I}(\Sigma) \}$$
\[ \text{Sen}^{3Cl}(\theta; \varphi)(\rho) \]

For sentences \( \rho = \@_i \rho \) we have:

\[
\text{Sen}^{3Cl}(\varphi) ; \text{Sen}^{3Cl}(\theta)(\@_i \rho) \\
= \{ \text{defn. of composition} \} \\
\text{Sen}^{3Cl}(\theta)(\text{Sen}^{3Cl}(\varphi)(\@_i \rho)) \\
= \{ \text{defn of Sen}^{3Cl} \} \\
\text{Sen}^{3Cl}(\theta)(\varphi_{\text{Nom}}(i) \text{Sen}^{3Cl}(\varphi)(\rho)) \\
= \{ \text{defn of Sen}^{3Cl} \} \\
\@_{\varphi_{\text{Nom}}(\theta_{\text{Nom}}(i))} \text{Sen}^{3Cl}(\theta)(\text{Sen}^{3Cl}(\varphi)(\rho)) \\
= \{ \text{I.H. and defn of composition} \} \\
@_{\varphi_{\text{Nom}}(\theta_{\text{Nom}}(i))} \text{Sen}^{3Cl}(\theta; \varphi)(\rho) \\
= \{ \text{defn of Sen}^{3Cl} \} \\
\text{Sen}^{3Cl}(\varphi; \theta)(@_i \rho)
\]

The proof for sentences \( \langle \lambda \rangle (\rho_1, \ldots, \rho_n) \) is analogous and the proof for sentences \( i \) follows directly from functional composition. The proof for sentences \( \rho \star \rho' \), \( \star \in \{ \land, \lor, \rightarrow \} \), as well as of \( \neg \rho \) comes directly from the induction hypothesis and the definition of Sen\(^{3Cl}\). Finally, for the sentences \( \langle \forall \chi \rangle \rho \in \text{Sen}^{3Cl}(\Sigma, \text{Nom}, \Lambda) \), we have:

\[
\text{Sen}^{3Cl}(\varphi) ; \text{Sen}^{3Cl}(\theta)(\langle \forall \chi \rangle \rho) \\
= \{ \text{defn. of composition} \} \\
\text{Sen}^{3Cl}(\theta)(\text{Sen}^{3Cl}(\varphi)(\langle \forall \chi \rangle \rho)) \\
= \{ \text{defn of Sen}^{3Cl} \} \\
\text{Sen}^{3Cl}(\theta)(\langle \forall \chi(\varphi) \rangle \text{Sen}^{3Cl}(\varphi[\chi])(\rho)) \\
= \{ \text{defn of Sen}^{3Cl} \} \\
\langle \forall (\chi(\varphi) \theta) \rangle \text{Sen}^{3Cl}(\theta[\chi(\varphi)])(\text{Sen}^{3Cl}(\varphi[\chi])(\rho)) \\
= \{ \text{I.H.} \} \\
\langle \forall (\chi(\varphi) \theta) \rangle \text{Sen}^{3Cl}(\varphi[\chi]; \theta[\chi(\varphi)])(\rho) \\
= \{ (7) \text{ of Definition 2.3.6} \} \\
\text{Sen}^{3Cl}(\varphi; \theta)(\langle \forall \chi \rangle \rho)
\]
3.1.3 $\mathcal{H}\mathcal{I}$-models functor

$(\Sigma, \text{Nom}, \Lambda)$-models are pairs $(M, W)$ where

- $W$ is a $(\text{Nom}, \Lambda)$-model in $REL$;
- $M$ is a function $|W| \rightarrow |\text{Mod}^2(\Sigma)|$.

The carrier set $|W|$ is the set of states of $(M, W)$; $\{W_n \mid n \in \text{Nom}\}$ represents the interpretation of nominals of Nom, whereas relations $\{W_\lambda \mid \lambda \in \Lambda_n, n \in \mathbb{N}\}$ represent the interpretation of modalities of $\Lambda$. We denote $M(w)$ simply by $M_w$.

A $(\Sigma, \text{Nom}, \Lambda)$-model homomorphism $h : (M, W) \rightarrow (M', W')$ consists of a pair aggregating

- a $(\text{Nom}, \Lambda)$-model homomorphism in $REL$, $h_{\text{st}} : |W| \rightarrow |W'|$ such that, for $i \in \text{Nom}$, $W'_i = h_{\text{st}}(W_i)$; and, for any $w_1, \ldots, w_n \in |W|$, $\lambda \in \Lambda_n$, and $(w_1, \ldots, w_n) \in W_\lambda$, $(h_{\text{st}}(w_1), \ldots, h_{\text{st}}(w_n)) \in W'_\lambda$.
- a natural transformation $h_{\text{mod}} : M \Rightarrow h_{\text{st}} M'$; note that $h_{\text{mod}}$ is a $|W|$-indexed family of $\Sigma$-model homomorphisms $h_{\text{mod}} = \{(h_{\text{mod}})_w : M_w \rightarrow M'_{h_{\text{st}}(w)} \mid w \in |W|\}$. In the sequel we often abbreviate $(h_{\text{mod}})_w$ by $h_w$.

The composition of $\mathcal{H}\mathcal{I}$-model homomorphisms is defined canonically by

$$h; h' = (h_{\text{st}}; h'_{\text{st}}, h_{\text{mod}}; h'_{\text{mod}}).$$

**Fact 3.1.1** Let $\Delta$ be any $\mathcal{H}\mathcal{I}$-signature. Then $\Delta$-models together with their homomorphisms constitute a category, denoted by $\text{Mod}^{\mathcal{H}\mathcal{I}}(\Delta)$.

**Reducts of $\mathcal{H}\mathcal{I}$-models**

Let $\Delta = (\Sigma, \text{Nom}, \Lambda)$ and $\Delta' = (\Sigma', \text{Nom}', \Lambda')$ be two $\mathcal{H}\mathcal{I}$-signatures, $\varphi = (\varphi_{\text{Sign}}, \varphi_{\text{Nom}}, \varphi_{\text{MS}})$ a morphism between $\Delta$ and $\Delta'$ and $(M', W')$ a $\Delta'$-model. The reduct of $(M', W')$ along $\varphi$, denoted by $\text{Mod}^{\mathcal{H}\mathcal{I}}(\varphi)(M', W')$, is the $\Delta$-model $(M, W)$ where
• \( W \) is the \((\varphi_{\text{Nom}}, \varphi_{\text{MS}})\)-reduct of \( W' \); i.e.
  - \(|W| = |W'|\);
  - for any \( n \in \text{Nom} \), \( W_n = W'_{\varphi_{\text{Nom}}(n)} \);
  - for any \( \lambda \in \Lambda \), \( W_\lambda = W'_{\varphi_{\text{MS}}(\lambda)} \);

and
• for any \( w \in |W| \), \( M_w = \text{Mod}^J(\varphi_{\text{Sign}})(M'_w) \).

For any \( \varphi : \Delta \to \Delta' \), and for any \( h' : (M'_1, W'_1) \to (M'_2, W'_2) \in \text{Mod}^{\mathcal{J}}(\Delta') \), \( \text{Mod}^{\mathcal{J}}(\varphi)(h') \) is the \( \text{Mod}^{\mathcal{J}}(\Delta) \) morphism

\[
\text{Mod}^J(\varphi)(h') : \text{Mod}^J(\varphi)(M'_1, W'_1) \to \text{Mod}^J(\varphi)(M'_2, W'_2)
\]

defined by \((h'_\text{str}, h'_\text{mod}; \text{Mod}^J(\varphi))\).

**Fact 3.1.2** \( \text{Mod}^{\mathcal{J}} \) is a functor.

**Theorem 3.1.1** A pushout square of \( \mathcal{J} \)-signature morphisms is a (weak) amalgamation square (for \( \text{Mod}^{\mathcal{J}} \)) if the underlying square of signature morphisms in \( \mathcal{I} \) is a (weak) amalgamation square.

**Proof.** Given a pushout of \( \mathcal{J} \)-signature morphisms as in Definition 2.3.3, the amalgamation of \((M_1, W_1) \) and \((M_2, W_2) \) is \((M', W') \) is defined as follows:

• \( W' \) is an amalgamation of \( W_1 \) and \( W_2 \) in \( \text{REL} \); this is because the projection functor \( \text{Sign}^{\mathcal{J}} = \text{Sign}^J \times \text{Sign}^{\text{REL}} \to \text{Sign}^{\text{REL}} \) preserves pushouts, hence the underlying square of signature morphisms in \( \mathcal{I} \) is a pushout.

• Let \(|W|\) be the carrier set of \( W_1, W_2 \) and \( W \) (by definition of reduct note all of them share the same carrier set). Then \( M' : |W| \to |\text{Mod}^J(\Sigma')| \) is the function determined by the (weak) pullback property of the result of applying functor \(|\text{Mod}^J(-)|\) to the underlying square of signature morphisms in \( J \).

**Corollary 3.1.2** If \( \mathcal{D}^J \) is adequate for \( \text{Mod}^J \) then \( \mathcal{D}^{\mathcal{J}} \) is adequate for \( \text{Mod}^{\mathcal{J}} \).
Proof. Any designated square \( \text{Sq} \) in the quantification space \( \mathcal{D}^{\mathcal{I}|\mathcal{J}} \) is a pushout by definition and since the projection functor \( \text{Sign}^{\mathcal{I}|\mathcal{J}} = \text{Sign}^\mathcal{J} \times \text{Sign}^{\mathcal{I}} \rightarrow \text{Sign}^\mathcal{J} \) preserves pushouts, it follows that the underlying square \( \text{Sq}_0 \) of \( \text{Sq} \) consisting of \( \mathcal{I} \) signature morphisms, is also a pushout. Because of this and also because the projection of \( \mathcal{D}^{\mathcal{I}|\mathcal{J}} \) to \( \mathcal{I} \) is included in \( \mathcal{D}^\mathcal{J} \), it follows that \( \text{Sq}_0 \) is isomorphic to a designated square in the quantification space \( \mathcal{D}^\mathcal{J} \). This, by the adequacy property of \( \mathcal{D}^\mathcal{J} \) with respect to \( \text{Mod}^\mathcal{J} \), is a weak amalgamation square. Hence \( \text{Sq}_0 \) is a weak amalgamation square, which by Theorem 3.1.1, implies that \( \text{Sq} \) is adequate for \( \text{Mod}^{\mathcal{I}|\mathcal{J}} \).

\[
\begin{align*}
3.1.4 & \quad \text{The satisfaction relation} \\
\text{For any } & (M, W) \in |\text{Mod}^{\mathcal{I}|\mathcal{J}}(\Sigma, \text{Nom}, \Lambda)| \text{ and for any } w \in |W|, \\
\bullet & \quad (M, W) \models^w \rho \iff M, w \models^\mathcal{J} \rho; \text{ when } \rho \in \text{Sen}^\mathcal{J}(\Sigma), \\
\bullet & \quad (M, W) \models^w i \iff W_i = w; \text{ when } i \in \text{Nom}, \\
\bullet & \quad (M, W) \models^w \rho \lor \rho' \iff (M, W) \models^w \rho \text{ or } (M, W) \models^w \rho', \\
\bullet & \quad (M, W) \models^w \rho \land \rho' \iff (M, W) \models^w \rho \text{ and } (M, W) \models^w \rho', \\
\bullet & \quad (M, W) \models^w \rho \Rightarrow \rho' \iff (M, W) \models^w \rho \text{ implies that } (M, W) \models^w \rho', \\
\bullet & \quad (M, W) \models^w \neg \rho \iff (M, W) \not\models^w \rho, \\
\bullet & \quad (M, W) \models^w [\lambda](\xi_1, \ldots, \xi_n) \iff \text{for any } w, w_1, \ldots, w_n \in |W| \text{ such that } (w, w_1, \ldots, w_n) \in W_\lambda \text{ we have that } (M, W) \models^wi \rho_i \text{ for some } 1 \leq i \leq n, \\
\bullet & \quad (M, W) \models^w \langle \lambda \rangle(\xi_1, \ldots, \xi_n) \iff \text{there exists } w, w_1, \ldots, w_n \in |W| \text{ such that } (w, w_1, \ldots, w_n) \in W_\lambda \text{ and } (M, W) \models^wi \xi_i \text{ for any } 1 \leq i \leq n, \\
\bullet & \quad (M, W) \models^w \@_i \rho \iff (M, W) \models^W \rho, \\
\bullet & \quad (M, W) \models^w (\forall \chi) \rho \iff (M', W') \models^w \rho \text{ for any } (M', W') \text{ such that } \text{Mod}^{\mathcal{I}|\mathcal{J}}(\chi)(M', W') = (M, W), \\
\bullet & \quad (M, W) \models^w (\exists \chi) \rho \iff (M', W') \models^w \rho \text{ for some } (M', W') \text{ such that } \text{Mod}^{\mathcal{I}|\mathcal{J}}(\chi)(M', W') = (M, W), \text{ and}
\end{align*}
\]
We write \((M, W) \models \rho\) iff \((M, W) \models^w \rho\) for any \(w \in |W|\).

As expected, there is a semantical equivalence between sentences \(\langle \lambda \rangle (\rho_1, \ldots, \rho_n)\) and \(\neg [\lambda] (\neg \rho_1, \ldots, \neg \rho_n)\). It is also interesting to note that if the quantification space allows quantifications over nominal variables, then the binder operator \(\downarrow\) that appears in many works on hybrid logic, e.g. [Gor96, Bra10], is redundant since sentences of the form \((\downarrow i) \rho\) are semantically equivalent to \((\forall i)(i \Rightarrow \rho)\).

### The Satisfaction Condition

**Theorem 3.1.2** Assume \(\mathcal{T}^3\) is adequate for \(\text{Mod}^3\). Let \(\Delta = (\Sigma, \text{Nom}, \Lambda)\) and \(\Delta' = (\Sigma', \text{Nom'}, \Lambda')\) be two \(3\mathcal{C}\)-signatures and \(\varphi : \Delta \rightarrow \Delta'\) a morphism of signatures. For any \(\rho \in \text{Sen}^3\mathcal{C}(\Delta)\), \((M', W') \in |\text{Mod}^3\mathcal{C}(\Delta')|\), and \(w \in |W|\),

\[
\text{Mod}^3\mathcal{C}(\varphi)(M', W') \models^w \rho \text{ iff } (M', W') \models^w \text{Sen}^3\mathcal{C}(\varphi)(\rho).
\]

**Proof.** Let us denote \(\text{Mod}^3\mathcal{C}(\varphi)(M', W') = \text{Mod}^3\mathcal{C}(\varphi)(M', W')\) by \((M, W)\). The proof is by recursion on the structure of the sentence \(\rho\)

1. \(\rho = i\) for some \(i \in \text{Nom}\)

\[
(M, W) \models^w i
\]

\[
\Leftrightarrow \quad \{ \text{defn. of } \models^w \}
\]

\[
W_i = w
\]

\[
\Leftrightarrow \quad \{ \text{defn of reduct, } W'_{\varphi_{\text{Nom}}(i)} = W_i \}
\]

\[
(M', W') \models^w \varphi_{\text{Nom}}(i)
\]

\[
\Leftrightarrow \quad \{ \text{defn of } \text{Sen}^3\mathcal{C}(\varphi) \}
\]

\[
(M', W') \models^w \text{Sen}^3\mathcal{C}(\varphi)(i)
\]

2. \(\rho \in \text{Sen}^3(\Sigma)\)

\[
(M, W) \models^w \rho
\]

\[
\Leftrightarrow \quad \{ \text{defn. of } \models^w \}
\]

\[
\text{Mod}^3(\varphi_{\text{Sign}})(M'_w) = M_w \models^3 \rho
\]
\( \Leftrightarrow \{ \text{by the Satisfaction Condition in } \mathcal{I} \} \)

\[ M'_w \models \text{Sen}^j(\varphi_{\text{Sign}})(\rho) \]

\( \Leftrightarrow \{ \text{defn. of } \models \} \)

\[ (M',W') \models^w \text{Sen}^j(\varphi_{\text{Sign}})(\rho) \]

\( \Leftrightarrow \{ \text{defn of } \text{Sen}^{3\mathcal{I}}(\varphi) \} \)

\[ (M',W') \models^w \text{Sen}^{3\mathcal{I}}(\varphi)(\rho) \]

3. \( \rho = \xi \lor \xi' \) for some \( \xi, \xi' \in \text{Sen}^{3\mathcal{I}}(\Delta) \)

\[ (M,W) \models^w \xi \lor \xi' \]

\( \Leftrightarrow \{ \text{defn. of } \models \} \)

\[ (M,W) \models^w \xi \lor (M,W) \models^w \xi' \]

\( \Leftrightarrow \{ \text{I.H.} \} \)

\[ (M',W') \models^w \text{Sen}^{3\mathcal{I}}(\varphi)(\xi) \lor (M',W') \models^w \text{Sen}^{3\mathcal{I}}(\varphi)(\xi') \]

\( \Leftrightarrow \{ \text{defn of } \models \} \)

\[ (M',W') \models^w \text{Sen}^{3\mathcal{I}}(\varphi)(\xi \lor \xi') \]

The proofs for the cases when \( \rho = \xi \land \xi', \rho = \xi \Rightarrow \xi', \rho = \neg \xi, \) etc. are analogous.

4. \( \rho = [\lambda](\xi_1, \ldots, \xi_n) \) for some \( \xi_1, \ldots, \xi_n \in \text{Sen}^{3\mathcal{I}}(\Delta), \lambda \in \Lambda_{n+1} \)

\[ (M,W) \models^w [\lambda](\xi_1, \ldots, \xi_n) \]

\( \Leftrightarrow \{ \text{defn of } \models \} \)

for any \( (w,w_1, \ldots, w_n) \in W_{\lambda} \) there is some \( k \in \{1, \ldots, n\} \) such that \( (M,W) \models_{w_k} \xi_k \)

\( \Leftrightarrow \{ \text{by reduct defn, } W_{\lambda} = W_{\varphi_{\text{MS}}(\lambda)} \} \)

for any \( (w,w_1, \ldots, w_n) \in W'_{\varphi_{\text{MS}}(\lambda)} \) there is some \( k \in \{1, \ldots, n\} \) such that \( (M,W) \models_{w_k} \xi_k \)

\( \Leftrightarrow \{ \text{I.H.} \} \)
for any \((w, w_1, \ldots, w_n) \in W'_{\varphi_{\mathcal{MS}}(\lambda)}\) there is some 
k \in \{1, \ldots, n\} such that 

\[
(M', W') \models_{w_k} \text{Sen}^{\mathfrak{3C}}(\varphi)(\xi_k)
\]

\[
\iff 
\{ \text{defn. of } \models_{w} \}
\]

\[
(M', W') \models_{w} [\varphi_{\mathcal{MS}}(\lambda)](\text{Sen}^{\mathfrak{3C}}(\varphi)(\xi_1), \ldots, \text{Sen}^{\mathfrak{3C}}(\varphi)(\xi_n))
\]

\[
\iff 
\{ \text{defn. of } \text{Sen}^{\mathfrak{3C}}(\varphi) \}
\]

\[
(M', W') \models_{w} \text{Sen}^{\mathfrak{3C}}(\varphi)([\lambda](\xi_1, \ldots, \xi_n))
\]

The proof for sentences of form \(\rho = \langle \lambda \rangle(\xi_1, \ldots, \xi_n)\) for some 
\(\xi_1, \ldots, \xi_n \in \text{Sen}^{\mathfrak{3C}}(\Delta), \lambda \in \Lambda_{n+1}\) is analogous.

5. \(\rho = @_i \xi\) for some \(\xi \in \text{Sen}^{\mathfrak{3C}}(\Delta), i \in \text{Nom}\)

\[
(M, W) \models_{w} @_i \xi
\]

\[
\iff 
\{ \text{defn. of } \models_{w} \}
\]

\[
(M, W) \models_{W_i} \xi
\]

\[
\iff 
\{ \text{I.H.} \}
\]

\[
(M', W') \models_{W_i} \text{Sen}^{\mathfrak{3C}}(\varphi)(\xi)
\]

\[
\iff 
\{ \text{by reduct defn, } W_i = W'_{\varphi_{\text{Nom}(i)}} \}
\]

\[
(M', W') \models_{W'_{\varphi_{\text{Nom}(i)}}} \text{Sen}^{\mathfrak{3C}}(\varphi)(\xi)
\]

\[
\iff 
\{ \text{defn. of } \models_{w} \}
\]

\[
(M', W') \models_{w} @_i \xi
\]

\[
\iff 
\{ \text{defn. of } \text{Sen}^{\mathfrak{3C}}(\varphi) \}
\]

\[
(M', W') \models_{w} \text{Sen}^{\mathfrak{3C}}(\varphi)(@_i \xi)
\]

6. \(\rho = (\forall \chi : \Delta \rightarrow \Delta_1)\xi\)

Figure 1 depicts the proof scheme. Hence,

\[
(M, W) \models_{w} (\forall \chi)\xi
\]

\[
\iff 
\{ \text{defn. of } \models_{w} \}
\]

for all \((M_1, W_1)\) such that \(\text{Mod}^{\mathfrak{3C}}(\chi)(M_1, W_1) = (M, W),\)
HYBRIDISATION OF INSTITUTIONS

\[ (M, W) \xleftarrow{\text{Mod}^{\mathcal{C}}(\varphi)} (M', W') \]

\[ \begin{array}{c}
\Delta \xrightarrow{\varphi} \Delta' \\
\chi \quad \chi(\varphi) \\
\Delta_1 \xrightarrow{\varphi[\chi]} \Delta'_1
\end{array} \]

\[ (M_1, W_1) \xleftarrow{\text{Mod}^{\mathcal{C}}(\varphi[\chi])} (M'_1, W'_1) \]

Figure 1: Proof Scheme

\[(M_1, W_1) \models^w \xi\]

\[\iff \{ \text{by Corollary 3.1.2, } \mathcal{D}^{\mathcal{C}} \text{ is adequate for } \text{Mod}^{\mathcal{C}} \}\]

for all \((M'_1, W'_1)\) such that

\[\text{Mod}^{\mathcal{C}}(\chi(\varphi))(M'_1, W'_1) = (M', W'),\]

\[\text{Mod}^{\mathcal{C}}(\varphi[\chi])(M'_1, W'_1) \models^w \xi,\]

\[\iff \{ \text{I.H.} \}\]

for all \((M'_1, W'_1)\) such that

\[\text{Mod}^{\mathcal{C}}(\chi(\varphi))(M'_1, W'_1) = (M', W'),\]

\[(M'_1, W'_1) \models^w \text{Sen}^{\mathcal{C}}(\varphi[\chi])(\xi)\]

\[\iff \{ \text{defn. of } \models^w \}\]

\[(M', W') \models^w (\forall \chi(\varphi))\text{Sen}^{\mathcal{C}}(\varphi[\chi])(\xi)\]

\[\iff \{ \text{defn of } \text{Sen}^{\mathcal{C}}(\varphi) \}\]

\[(M', W') \models^w \text{Sen}^{\mathcal{C}}(\varphi)(((\forall \chi)\xi))\]

When \(\rho = (\exists \chi : \Delta \rightarrow \Delta_1)\xi\), the proof is analogous.

\[\Box\]

Note that in the quantifier-free situation, i.e. when \(\mathcal{D}^{\mathcal{C}}\) is trivial, the adequacy assumption of Theorem 3.1.2 is not needed.

**Corollary 3.1.3 (The Satisfaction Condition)** \(\mathcal{H}^{\mathcal{C}}\) is an institution.
3.1.5 Base logic versus its hybridisation

In hybridised institutions we may have to consider, if the base institution has boolean connectives, two sets of Boolean connectives, those of the hybridised level and those of the base institution. The following simple result, recently presented in [Dia13], allows us to ignore the distinction between these two sets of Boolean connectives. The result also captures the general relationship between quantification at the base and at the hybridised level.

Fact 3.1.3 ([Dia13]) Let $\mathcal{H} \mathcal{I}$ be an hybridisation of $\mathcal{I}$ and let us denote the Boolean connectives and the quantifiers in the base institution $\mathcal{I}$ by $\wedge, \vee, \Rightarrow, \neg$, and $\forall, \exists$, respectively. For any $(\Sigma, \text{Nom}, \Lambda)$-model $(M, W)$, $w \in |W|$, sentences $\rho, \rho' \in \text{Sen}^I(\Sigma)$ of the base institution and for each $\chi \in \mathcal{D}$

- $(M, W) \models^w \rho \star \rho'$ iff $(M, W) \models^w \rho \star \rho$ for $\star \in \{\land, \lor, \Rightarrow\}$,
- $(M, W) \models^w \neg \rho$ iff $(M, W) \models^w \neg\rho$,
- $(M, W) \models^w (\forall \chi) \rho$ implies $(M, W) \models^w (\forall (\chi, 1_{\text{Nom}}, 1_{\Lambda})) \rho$, and
- $(M, W) \models^w (\exists \chi, \text{Nom}, 1_{\Lambda})) \rho$ implies $(M, W) \models^w (\exists \chi) \rho$.

One may legitimately wonder about the existence of a canonical embedding of the base institution $\mathcal{I}$ into its hybridisation $\mathcal{H} \mathcal{I}$. That relationship is formalised on the following result:

Theorem 3.1.3 Let $\mathcal{I}$ an institution and $\mathcal{H} \mathcal{I}$ its quantifier-free hybridisation. Then, there is a conservative comorphism $(\Phi, \alpha, \beta) : \mathcal{I} \rightarrow \mathcal{H} \mathcal{I}$ where

- $\Phi(\Sigma) = (\Sigma, \{i\}, 0)$,
- $\alpha(\rho) = @_i \rho$, and
- $\beta(\Sigma, \text{Nom}, \Lambda) = M_{W_i}$.

Proof. Let us consider a $\Phi(\Sigma)$-model $(M, W)$, and a $\Sigma$-sentence $\varphi$. Then,

$$\beta(M, W) \models^I \varphi$$
\[
\begin{align}
&\Leftrightarrow \{ \text{defn. of } \beta \} \\
&M_{W_i} \models^j \rho \\
&\Leftrightarrow \{ \text{defn of } \models^{\mathcal{H}^j} \} \\
&(M, W) \models^{\mathcal{H}^j} \@_i \rho \\
&\Leftrightarrow \{ \text{defn of } \alpha \} \\
&(M, W) \models^{\mathcal{H}^j} \alpha(\rho)
\end{align}
\]

This proofs the satisfaction condition for the comorphism. Surjectivity of \( \beta \) comes from the observation that for any \( \Sigma \)-model \( M \) we may take a \( \Phi(\Sigma) \) model \((M, W)\) with \( M_{W_i} = M \). Therefore, the comorphism is conservative.

The notion of a constrained model, recently introduced by R. Diaconescu in [Dia13], plays a crucial role on the remaining of the thesis. In particular, it accommodates ‘rigidity’ of first-order variables, a fundamental notion for the characterization of first-order encodings introduced in the next chapter.

**Definition 3.1.1 ([Dia13])** A constrained \( \mathcal{H}^j \)-model functor is a subfunctor \( \text{Mod}^C \subseteq \text{Mod}^{\mathcal{H}^j} \) (i.e., \( \text{Mod}^C(\Delta) \) is a subcategory of \( \text{Mod}^{\mathcal{H}^j}(\Delta) \), \( \Delta \in |\text{Sign}^{\mathcal{H}^j}| \)) such that it reflects weak amalgamation. Models in \( \text{Mod}^C(\Delta) \) are called constrained \( \mathcal{H}^j \)-models.

Informally, the meaning of the reflection requirement in Definition 3.1.1 is that, pushout squares of signature morphisms, the amalgamation of constrained models yields a constrained model.

The choice of a constrained model functor is, therefore, the third parameter for the hybridisation method:

**Theorem 3.1.4 ([Dia13])** \( (\text{Sign}^{\mathcal{H}^j}, \text{Sen}^{\mathcal{H}^j}, \text{Mod}^C, \models) \) is an institution.

In the sequel, an hybridisation without constrained models, i.e., with \( \text{Mod}^C = \text{Mod}^{\mathcal{H}^j} \) is called free hybridisation of \( \mathcal{J} \). Observe that the satisfaction relation \( \models^{\mathcal{H}^j \mathcal{C}} \) of a hybridisation \( \mathcal{H}^j \mathcal{C} \) with properly constrained models is not necessarily the restriction of \( \models^{\mathcal{H}^j} \). This is justified by the fact that the satisfaction relation for quantifiers relies upon expansions with respect to \( \text{Mod}^C \) and not to \( \text{Mod}^{\mathcal{H}^j} \).
3.2 EXAM P L S

A myriad of examples of hybridised logics may be generated through the hybridisation process described in the previous sections. They arise by considering different instances of the three parameters of this process:

1. the base institution $\mathcal{I}$
2. the quantification space $\mathcal{D}^{\mathcal{I}_J}$, and
3. the constrained models $(\text{Mod}^C)$.

Example 3.2.1 (Hybrid propositional logic $\mathcal{H}'PL$) Applying the quantifier-free version of the hybridization method described above to $PL$ and fixing $\Lambda_2 = \{\lambda\}$ and $\Lambda_n = \emptyset$ for each $n \neq 2$, we obtain the institution $\mathcal{H}'PL$ of the “standard” hybrid propositional logic (without state quantifiers): the category of signatures is $\text{Sign}^{\mathcal{H}'PL} = \text{Set} \times \text{Set}$ with objects denoted by $(P, \text{Nom})$ and morphisms by $(\varphi_{\text{Sign}}, \varphi_{\text{Nom}})$; sentences are the usual hybrid propositional formulas, i.e., modal formulas closed by Boolean connectives, $[\lambda]$ denoted $\Box$, $\langle \lambda \rangle$ denoted $\Diamond$, and by the operator $@_i$, $i \in \text{Nom}$; models consists of pairs $(M, W)$ where $W$ consists of a carrier set $|W|$, interpretations $W_i \in |W|$ for each $i \in \text{Nom}$, and a binary relation $W_\lambda \subseteq |W| \times |W|$, and for each $w \in |W|$, $M_w$ is a propositional model, i.e., a function $M_w : P \rightarrow \{0, 1\}$ which is equivalent to a subset $M_w \subseteq P$. Note that by virtue of Fact 3.1.3 we do not need to make a distinction between the Boolean connectives at the level of $PL$ and at the level of $\mathcal{H}'PL$.

The T, S4, and S5 versions of hybrid propositional logic are obtained by constraining the models of $\mathcal{H}'PL$ to those models $(M, W)$ for which $W_\lambda$ is reflexive, preorder, and equivalence, respectively.

When we relax to arbitrary sets of modalities $\Lambda$ rather than only $\lambda$, we obtain the “multi-modal hybrid propositional logic”.

A challenging issue concerns finding suitable quantification spaces to capture versions of hybrid propositional logic. One choice is the quantifier-free version in which $\mathcal{D}^{\mathcal{H}'PL}$ would consist only of identities.

---

1 Note that by fixing $\Lambda$ to only one symbol of arity 2 means the restriction to a subcategory of the $\mathcal{H}PL$-signatures, i.e. $\text{Sign}^{\mathcal{H}'PL} \subseteq \text{Sign}^{\mathcal{H}PL}$. Then $\mathcal{H}'PL$ is the ‘sub-institution’ of $\mathcal{H}PL$ determined by this restriction of the signatures.
However, it would be interesting, along the hybridization process, to capture a quantifier such as $E$, where $E \rho$ means that “$\rho$ is true in some state of the model” [AB01]. Considering as a quantification space the extensions of signatures with nominal symbols, paves the way to express the following properties:

| $\mathcal{M}, \mathcal{W} \models^w (\forall i)i \leftrightarrow \rho$ | iff | $\rho$ is satisfied at $w$ iff $w$ is unique in $\mathcal{M}, \mathcal{W}$ |
| $\mathcal{M}, \mathcal{W} \models^w (\exists i)i \rho$ | iff | $(\mathcal{M}, \mathcal{W}) \models E \rho$ |

Let us denote this modification of $\mathcal{H}'PL$ by $\mathcal{H}'PLn$. A block of nominal variables $X$ for a $\mathcal{H}'PLn$ signature $(P, Nom)$ is a finite set nominal variables of the form $(x, P, Nom)$ (like in the case of FOL variables, $x$ is the name and $(P, Nom)$ the qualification of the variable) such that $(x, P, Nom), (x', P, Nom) \in X$ implies $x = x'$. Then $\mathcal{D}^{\mathcal{H}'PLn}$ may be defined as consisting of the signature extensions with blocks of nominal variables, i.e. $(P, Nom) \hookrightarrow (P, Nom \cup X)$. For any signature morphism $\varphi : (P, Nom) \to (P', Nom')$ and $X$ block of nominal variables for $(P, Nom)$ we define $X^\varphi = \{(x, P', Nom') \mid (x, P, Nom) \in X\}$. Then $\chi(\varphi)$ is the extension $(P', Nom') \hookrightarrow (P', Nom' \cup X^\varphi)$ and $\varphi[\chi]$ is the canonical extension of $\varphi$ that maps each $(x, P, Nom)$ to $(x, P', Nom')$.

When we combine this quantification with the constraints T, S4, S5, etc., it becomes necessary to establish the adequacy condition for the constrained model sub-functor. However, in this case, this is almost trivial since we may consider $\mathcal{D}^{PL}$ (the quantification space at the level of the base institution) as being trivial. Furthermore it is also immediate that the amalgamation of constrained models is still a constrained model.

**Example 3.2.2 ($\mathcal{H}(TRM)$)** Let us consider the free-hybridisation of the institution $TRM$ of Example 2.3.1. The signature category corresponds to

$$\text{Sign}^{TRM} \times \text{Sign}^{REL} \cong \text{Sign}^{REL}.$$
Since \( \text{Sen}^{\text{TRM}}(\ast) = \emptyset \), we have that \( \text{Sen}^{\mathcal{H}^\text{TRM}}(\ast, \text{Nom}, \Lambda) \) is the set of sentences built up from nominals in \( \text{Nom} \) by the application of modalities in \( \Lambda \) and boolean connectives. This kind of formulas has been called by pure hybrid formulas in [BdRV01, Ind07]. The models of \( \text{Mod}^{\mathcal{H}^\text{TRM}}(\ast, \text{Nom}, \Lambda) \) consist of relational structures \((W, M)\), where \( M \) is a constant function \( M_w = \ast, \) for any \( w \in |W| \). This fragment of \( \mathcal{H}^{\text{PL}} \) has been studied in, e.g., [Ind07], namely concerning completeness.

Example 3.2.3 (Double layered hybridization \( \mathcal{H}'\mathcal{H}'\text{PL} \)) An institution for specification of hierarchical state transition systems is achieved by a double layered hybridization of \( \text{PL} \) (but it could also be extended to any other base institution instead of \( \text{PL} \)). This means a hybridization of \( \mathcal{H}'\mathcal{H}'\text{PL} \) (or of any of its variants from Ex. 3.2.1). Let us denote by \( \mathcal{H}'\mathcal{H}'\text{PL} \) the quantifier-free hybridization of \( \mathcal{H}'\text{PL} \). The models of this institution are “Kripke structures of Kripke structures”.

Thus the \( \mathcal{H}'\mathcal{H}'\text{PL} \) signatures are triples \((P, \text{Nom}^0, \text{Nom}^1)\) with \( \text{Nom}^0 \) and \( \text{Nom}^1 \) denoting the nominals of the first and second layer of hybridization, respectively. In order to prevent potential ambiguities, in general we tag the symbols of the respective layers of hybridization by the superscripts 0 (for the first layer) and 1 (for the second layer). This convention should include nominals and connectors (\( \Diamond, \land, \) etc.); however by Fact 3.1.3 in the case of the Boolean connectors we may skip this. For instance, the expression \( @_j^0 k^0 \land \Box^1 \rho \) is a sentence of \( \mathcal{H}'\mathcal{H}'\text{PL} \) where the symbols \( k \) and \( j \) represent nominals of the first and second level of hybridization and \( \rho \) a \( \text{PL} \) sentence. On the other hand, according to this tagging convention the expression \( @_0^1 k^1 \land \Box^1 \rho \) would not parse. Our tagging convention is extended also to \( \mathcal{H}'\mathcal{H}'\text{PL} \) models: a \((P, \text{Nom}^0, \text{Nom}^1)\)-model is denoted by \((M^1, W^1)\) where for any \( w \in |W| \) the models \( M^1_w \) are denoted by \((W^0_w, M^0_w)\).

We may consider also quantified versions of the double hybridization of \( \text{PL} \), and there are several variants of those depending on the existence of the quantifiers at each layer. Let us denote these generically by \( \mathcal{H}'(\mathcal{H}'\text{PL}x)y \) where \( x \) is either empty or \( n \) and \( y \) is either empty, \( n^0, n^1 \), or \( n^{01} \). The absence of \( x \) and/or \( y \) means the absence of quantification at the base and/or the upper level, respectively. The superscripts 0 and 1 tagged to \( n \) denote the existence of quantifica-
tion at the upper level with nominals variables from the lower and upper level of hybridization, respectively. For example \( H'(H'PLn)_{n^0} \) would have all possible quantifications, such as \((\forall^0i^0), (\forall^1i^0), (\forall^1i^1)\) while \( H'(H'PL)_{n^1} \) would have only those of the form \((\forall^1i^1)\).

Let \( H'(H'PL)' \) denote the double hybridization obtained by constraining the models \((M^1, W^1)\) to those such that for any \(w, w' \in |W^1|\) and any \(i^0 \in \text{Nom}^0\) we have that \(|W_w^0| = |W_{w'}^0|\) and \((W_w^0)_{i^0} = (W_{w'}^0)_{i^0} \).

The quantified variants of \( H'(H'PL)' \) would require the reflection condition of Def. 3.1.1 that holds directly from the \( H'PL \) reduct definition: for any pushout square of signature morphisms in \( \text{Sign}^{H'PL} \)

\[
\begin{array}{ccc}
\Delta & \xrightarrow{\varphi_1} & \Delta_1 \\
\varphi_2 & \downarrow & \downarrow \theta_1 \\
\Delta_2 & \xrightarrow{\varphi_2} & \Delta'
\end{array}
\]

let us consider a constrained \((\Sigma_k, \text{Nom}_k, \Lambda_k)\)-model \((M_k^1, W_k^1)\) for each \(k \in \{1, 2\}\) such that \(\text{Mod}^C(\varphi_1)(M_k^1, W_k^1) = \text{Mod}^C(\varphi_2)(M_k^1, W_k^1)\). We take the amalgamation \((M', W') = (M_1^1, W_1^1) \otimes (M_2^1, W_2^1)\) according to Theorem 3.1.1. Regards to the sharing on the substates universe, we have by definition of reduct we have that \(|W_1^1| = |W_k^1|\) and that, for any \(w \in |W'|\), \(\text{Mod}^{HPL}(\theta_{\text{Sign}_k})(W_w^0, M_w^0) = (W_{W_w}^0, M_{W_w}^0)\) and, hence \(|W_w^0| = |W_{W_w}^0|\). Since \((M_k^1, W_k^1)\) are constrained models, we have that \(|W_{W_w}^0| = |W_{W_{W_w}}^0|\) for any \(w, w' \in |W'|\). The proof for the sharing in the interpretation of \(\text{Nom}^0\) symbols is analogous. Therefore \(|W_{W_w}^1| = |W_{W_{W_w}}^1|\) for any \(w, w' \in |W'|\), i.e., \((M', W')\) is a constrained model.

The layered hybridization construction (together with its associated notational conventions) may be iterated to higher layers of hybridizations, e.g. \( H^3PL, H^4PL, ... \) The convention on quantified versions carries also forward to such higher layers of hybridization.

**Example 3.2.4 (H'FOL)** Applying of the hybridisation method to FOL leads to the first-order hybrid logic of \([\text{Bra05}, \text{Bra10}]\). For that we have to restrict the (hybrid) signatures to singleton sets of unary modalities (i.e. \(\Lambda_2 = \{\lambda\}\) and \(\Lambda_n = \emptyset\) for the other naturals) and the base signatures to the one-sorted case. As quantification space we take first-order and nominal variables expansions. Note that, as stated, binder connec-
tives now implicitly defined through quantification over nominals. As in the case of $\mathcal{H}(PL)$, by the Fact 3.1.3, we do not need to make a distinction between the Boolean connectives at the level of $FOL$ and those at the level of $\mathcal{H}FOL$. Moreover, because the carriers of the $FOL$ models are non-empty we may easily show that in this case the implications of Fact 3.1.3 about quantifiers may be turned into equivalences. Hence it is also not necessary to distinguish between quantifiers at the base $FOL$ level and at the hybridised $\mathcal{H}FOL$ level.

Example 3.2.5 (Predefined sharing in $\mathcal{H}(REL)$) Let $\mathcal{H}(REL)'$ be the hybridisation of $REL$ that constrained the models of $\mathcal{H}(REL)$ to those models $(M, W)$ such that $\{M_i \mid i \in |W|\}$ share the same universe (underlying set) and the same interpretation of constants. The sharing is also extended to model homomorphisms: for all $\mathcal{H}(REL)'$-models $(M, W)$ and $(M', W')$, a model homomorphism $h : (M, W) \rightarrow (M', W')$ in $\mathcal{H}(REL)$ belongs to $\mathcal{H}(REL)'$ if and only if $h_i = h_j$ for all $i, j \in |W|$. Note that the amalgamation of models preserves sharing. The reflection condition of Definition 3.1.1 holds directly from the $\mathcal{H}(REL)$-reduct definition: for any pushout square of signature morphisms in $Sign^{\mathcal{H}(REL)}$

\[
\begin{array}{ccc}
\Delta & \xrightarrow{\varphi_1} & \Delta_1 \\
\downarrow{\varphi_2} & & \downarrow{\vartheta_1} \\
\Delta_2 & \xrightarrow{\vartheta_2} & \Delta'
\end{array}
\]

let us consider a constrained $\{\Sigma_k, \text{Nom}_k, \Lambda_k\}$-model $(M_k, W_k)$, for each $k \in \{1, 2\}$, such that

$\text{Mod}^C(\varphi_1)(M_1, W_1) = \text{Mod}^C(\varphi_2)(M_2, W_2)$.

We take the amalgamation $(M', W') = (M_1, W_1) \otimes (M_2, W_2)$ according to Theorem 3.1.1. Therefore, for any $w \in W_k$, $(M_k)_w = \text{Mod}^{REL}(\vartheta_{k\text{Sign}})(M_w)$ and, hence, $|(M_k)_w| = |M_w|$. Since $(M_k, W_k)$ is a constrained model, we also have that, for any $w, w' \in W_k$, $|(M_k)_w| = |(M_k)_{w'}|$. By definition of reduct we have $W_k = W$ (and since $\text{Mod}^{REL}(\vartheta_{k\text{Sign}})(M_{w'}) = (M_k)_{w'}$, we have $|M_w| = |M_{w'}|$, for any $w, w' \in W'$. Hence the reflection condition of Definition 3.1.1 is fulfilled.
\(\mathcal{D}^{\mathcal{H}(REL)'}\) consists of signature extensions with FOL variables (for the states), with nominal variables (in the style of \(\mathcal{D}^{\mathcal{H}(PL)}\) of Example 3.2.1) and with variables for modalities.

Note that, as in \(\mathcal{H}(FOL)\), in \(\mathcal{H}(REL)'\) we also do not need to distinguish between the Boolean connectives and the quantifiers at the base and at the hybridised level.

An interesting variant of \(\mathcal{H}(REL)\), say \(\mathcal{H}(REL)''\), is achieved if we constrain the models with “increasing domains”, i.e., with the preservation of universes by accessible relations, in the sense that for any \(w, w' \in W\), \((w, w') \in W^\lambda\) implies that \(|M_w| \subseteq |M_{w'}|\). This is an usual assumption taken in many modal versions of the first-order logic (cf. [Bra10]). The verification of the reflection condition of Definition 3.1.1 can be made exactly as above.

Let us define \(\mathcal{H}(REL)_0\) as the free-hybridisation of the quantifier-free fragment of \(REL\), with \(\mathcal{D}^{\mathcal{H}(REL)_0}\) consisting just of nominal expansions. We denote by \(\mathcal{H}(REL)'_0\) the institution achieved by constraining \(\mathcal{H}(REL)_0\) to “increasing domains”.

\[\circ\]

**Example 3.2.6 \((\mathcal{H}(FOLR)' and \mathcal{H}(EQ)')\)** Example 3.2.5 above may be considered an example of ‘predefined’ or ‘default’ sharing since the interpretation of all constants is shared. However in formal specification applications it is also important to consider ‘user defined’ sharing, in which one has the possibility to define by hand the entities to be shared.

Consider the FOLR defined below as the base institution:

- **Sign\(^{FOLR}\)** is the category of MFOL signatures of [DS07]: its objects are tuples \((S, S_0, F, F_0, P, P_0)\) where \((S_0, F_0, P_0)\) and \((S, F, P)\) are FOL signatures such that \((S_0, F_0, P_0)\) is a sub-signature of \((S, F, P)\); the symbols of \((S_0, F_0, P_0)\) are called ‘rigid’, and signature morphisms \(\varphi : (S, S_0, F, F_0, P, P_0) \to (S', S'_0, F', F'_0, P', P'_0)\) are just FOL signature morphisms \((S, F, P) \to (S', F', P')\) that map rigid symbols to rigid symbols.

- **Sen\(^{FOLR}\)** \((S, S_0, F, F_0, P, P_0)\) consists of those sentences in \(\text{Sen}^{FOL}(S, F, P)\) that contains only quantifiers over rigid variables,
We let $\mathcal{H}(\text{FOLR})$ be the hybridisation of FOLR with quantifications over nominal, modalities, and rigid FOL variables.

For $\mathcal{H}(\text{FOLR}')$, consider the constrained model sub-functor $\text{Mod}^C$ such that $(M, W) \in |\text{Mod}^C(\Sigma, \text{Nom}, \Lambda)|$ if and only if for all $i, j \in |W|$ and each rigid symbol $x$ in $\Sigma$, $(M_i)_x = (M_j)_x$. For any pushout square of signature morphisms in $\text{Sign}^{\text{FOLR}}$

\[
\begin{array}{c}
\Sigma, \text{Nom}, \Lambda \\
\downarrow \theta_2 \\
\Sigma_2, \text{Nom}_2, \Lambda_2
\end{array} \quad \xrightarrow{\varphi_2} \quad \begin{array}{c}
\Sigma_1, \text{Nom}_1, \Lambda_1 \\
\downarrow \theta_1 \\
\Sigma', \text{Nom}', \Lambda'
\end{array}
\]

let us consider a constrained $(\Sigma_k, \text{Nom}_k, \Lambda_k)$-model $(M_k, W_k)$ for each $k \in \{1, 2\}$ such that $\text{Mod}^C(\varphi_1)(M_1, W_1) = \text{Mod}^C(\varphi_2)(M_2, W_2)$. We take the amalgamation $(M', W') = (M_1, W_1) \otimes (M_2, W_2)$ according to Theorem 3.1.1. Then we consider any rigid symbol $x$ of $\Sigma'$ and any $i, j \in |W'|$. By Proposition 3.1.1 we have that

\[
\begin{array}{c}
\Sigma \\
\downarrow \theta_2 | \text{Sign} \\
\Sigma_2
\end{array} \quad \xrightarrow{\varphi_2 | \text{Sign}} \quad \begin{array}{c}
\Sigma_1 \\
\downarrow \varphi_1 | \text{Sign} \\
\Sigma
\end{array} \quad \xrightarrow{\theta_1 | \text{Sign}} \quad \begin{array}{c}
\Sigma' \\
\downarrow \theta_2 | \text{Sign} \\
\Sigma
\end{array}
\]

is a pushout square of FOLR signature morphisms. Note that the set of rigid symbols in $\Sigma'$ is the union of the translations of the rigid symbols from both $\Sigma_1$ and $\Sigma_2$ through $(\theta_1 | \text{Sign})$ and $(\theta_2 | \text{Sign})$. This means that there exists $k \in \{1, 2\}$ and $x_k$ rigid symbol of $\Sigma_k$ such that $x = \theta_k(x_k)$. It follows that $(M'_i)_x = ((M_k)_i)_x$ and $(M'_j)_x = ((M_k)_j)_x$, hence $(M'_i)_x = (M'_j)_x$ since $(M_k)_i x_k = (M_k)_j x_k$ (because $(M_k, W_k)$ is a constrained model). This proves that $(M', W')$ is a constrained model, which gives the reflection condition of Definition 3.1.1.

In analogy with Example 3.2.5, it is natural to consider a version of $\mathcal{H}(\text{FOLR})$ with models with “increasing domains”, constraining the models where, for any $w, w' \in W$, $\lambda \in \Lambda$, $(w, w') \in R_\lambda$, $(M_w)_s \subseteq
HYBRIDISATION OF INSTITUTIONS

$(M_s, W)_{s'}$, for any $s \in S$. However, in this case we have no way to establish the reflection condition for the constrained models. This happens because we just have, by reduct, the preservation of universes on the sorts of $\Delta_k$, where $k$ is such that $\theta_k(\lambda_k) = \lambda$.

There are some interesting first-order modal logics captured by suitable versions of $\mathcal{H}FOLR'$. The first order modal logic institution $MFOLO$ of [DS07] is such an example. In [MFMB11] we suggest a version of the hybrid first order logic as a suitable logic for the specification of reconfigurable systems. That logic can be generated as a variant of $\mathcal{H}FOLR'$ taking the rigidification on the entire set of sorts and quantification over first-order logic variables.

Example 3.2.7 (User defined sharing in Hybrid Partial Algebra) Let $PAR$ be a rigid version institution of the partial algebras of Example 2.3.7 which is defined similarly to $FOLR$, the rigid version of $FOL$ in Example 3.2.6. This means that we consider signatures of the form $(\Sigma, S_0, TF_0, PF_0)$ where $(S_0, TF_0, PF_0)$ is a sub-signature of ‘rigid symbols’ for a $PA$ signature $(\Sigma, TF, PF)$, etc. We skip here the details that replicate the corresponding details from the definition of $FOLR$. Let $\mathcal{H}PAR$ be the hybridisation of $PAR$ with quantifications over nominals, modalities, and $PAR$ variables (i.e. rigid total variables). We consider $\mathcal{D}^{\mathcal{H}PAR}$ to consist of signature extensions with total rigid (first-order) variables. The amalgamation property of $PA$ entails the adequacy of $\mathcal{D}^{PAR}$ for $\text{Mod}^{PAR}$. From Corollary 3.1.2 it follows that $\mathcal{D}^{\mathcal{H}PAR}$ is adequate for $\mathcal{H}PAR$.

We denote by $\mathcal{H}PAR'$ the hybridisation obtained by constraining the model sub-functor to $\text{Mod}^{C}$ defined by $(M, W) \in |\text{Mod}^{C}(\Sigma, Nom, \Lambda)|$ if and only if

- rigid sorts and total functions share the same interpretations in all the states, and

- rigid partial functions share domains.

This means that for all $i, j \in |W|$, for each symbol $x$ in $S_0$ or $TF_0$, $(M_i)_x = (M_j)_x$ and, for any $\sigma$ in $PF_0$, we have that $\text{dom}((M_i)_\sigma) = \text{dom}((M_j)_\sigma)$. The reflection condition for $\text{Mod}^{C}$ is established in this case as it was in Example 3.2.6.
Example 3.2.8 ($\mathcal{H}(MVL)$) A very interesting logic can be obtained with the hybridisation of $MVL_L$ of Example 2.3.8 (for a fixed complete residuated lattice $L$). Beyond to the expressivity introduced by the hybrid features (and the respective Kripke semantics), new expressivity patterns arise at the single level. For instance, unlike the base case where each sentence is tagged by a $L$-value, we have now more structured expressions involving different $L$-values (e.g. $(\rho, p) \land (\rho', p')$).

As in the $\mathcal{H}(REL)$ example we may consider a version of $\mathcal{H}(MVL_L)$ constraining the models to those that share the interpretation of universes and constants. For this case, the reflection condition for $\text{Mod}^C$ came as in Example 3.2.5 (since the notions of reduct and signature morphism are exactly the same).

A main feature of this approach concerns the possibility of building encodings of hybridised logics into $FOL$. This is an extremely relevant step in the quest for suitable tool support for engineering methodologies based on hybridisation. Under what circumstances this can be done and how, is the subject of the following chapter.
A main technical aim of the hibridisation process introduced in the previous chapter is to provide the necessary infrastructure to transport specifications from a logical system to another, better equipped in terms of effective proof support. This chapter contributes in this direction through the systematic characterisation of encodings of hibridised institutions into the institution of many sorted first-order logic $FOL$. In particular, for any institution ‘encodable’ in presentations in $FOL$, we suggest a method to construct an encoding from its hybridisation to $FOL$. Therefore, a wide variety of computer assisted provers for $FOL$ can be ‘borrowed’ to reason about specifications in the new, hybridised logics.

Actually, the wide variety of formal verification tools for $FOL$ can be ‘borrowed’ as valuable resources to specify and verify systems described in the languages of encodable hybridisations.

Technically such encodings are achieved by extending the classical standard translation of modal logic into the (one-sorted) first order logic [vB83], more precisely, for its hybrid version [Bla00], to the encodings of hybridised institutions into $FOL$. We call this process “hybridisation of encodings”.

The general idea of the standard translation from $\mathcal{HPL}$ into the (one-sorted) first-order logic, is to consider a sort to denote the state space, where nominals are interpreted as constants, modalities as binary relations, and propositions as unary predicates (where $p(w)$ means that the proposition $p$ holds at state $w$). The idea underlying the standard translation $\mathcal{HFL2FOL}$ (e.g. [Bra10]), is to extend this encoding by considering a new universe $ST$ as an extra sort in the signature, and “flattening” the universes, operations and predicates of the (local) $FOL$-models to an unique (global) $FOL$-model. Local functions and predicates become parametric over states, and the state universes distinguished with a sort-family of definability predicates. Intuitively, whenever $m$ belongs to the universe of $w$, $\pi(w, m)$ and $\sigma(w, m) = b$
means that $\pi(m)$ and $\sigma(m) = b$ hold in the state $w$. Moreover, restricting this global model $M$ to the local universes, operations and predicates of a fixed word $w$, we obtain a “slice of $M$”, say $M|_w$, that consists of a local FOL-model representing (and coinciding with) $M_w$.

The method is based on the application of state-parametrisation construction of $\mathcal{H}FOL2FOL$ to lift $\mathcal{I}2FOL$ to $\mathcal{H}\mathcal{I}2FOL$. Thus, all the signatures and sentences targeted by $\mathcal{I}2FOL$ become parametric on states and the remaining sentences are treated as in $\mathcal{H}FOL2FOL$. A slice $M|_w$ corresponds now to the “FOL-interpretation” of the local $\mathcal{I}$-model $M_w$, which can be recovered using $\mathcal{I}2FOL$.

As discussed in [DM], this process can be understood, on the perspective of logic combination, as a combination of logic encodings, between the standard translation of hybrid logic into FOL and other encodings into FOL.

Such encodings are required to be conservative ‘theoroidal comorphisms’ [Mos96, GR02], i.e., ordinary comorphisms into presentations, mapping signatures to theories (cf. Section 2.3.7). Conservativity, i.e., the surjectivity of the models translation, is a sufficient condition to use such maps as actual encodings. In particular, it is necessary in order to borrow from FOL proof resources in a sound and complete way. This entails the need for an abstract characterisation of conservativity which is done in the sequel.

The first steps in defining a method for generating first-order encodings in hybridised institutions were presented in [MMDB11a]. The present version following recent work documented in [DM], extends it to presentations, constrained and quantified models. The characterisation of conservativity also appeared in [DM].

Subsection 4.1.1 introduces some preliminaries for the presentation of the encoding, namely, the state-parametrisation and the corresponding slicing procedures. The remaining components of the encoding are presented in Subsection 4.1.2. The illustration of the method is done in Subsection 4.2 where a number of encodings are generated as instantiations of the method. Finally, Subsection 4.3 introduces a characterisation of conservativity. The examples are revisited and their conservativity discussed.
4.1 HYBRIDISING FOL-ENCODINGS

4.1.1 States parametrisation

Notation 4.1.1 For any FOL-signature \((S, F, P)\) we denote by \(([S], [F], [P])\) the following FOL-signature:

- \([S] = S \cup \{ST\}\), where \(ST\) is a designated sort not in \(S\),
- \([F]_{ar \to s} = \begin{cases} \mathcal{F}_{ar' \to s} & \text{for any } s \in S, ar' \in S^* \text{ such that } ar = (ST)ar' \\
\emptyset & \text{for the other cases;} \end{cases}\)
- \([P]_{ar} = \begin{cases} \mathcal{P}_{ar'} & \text{for any } ar' \in S^* \setminus S \text{ such that } ar = (ST)ar' \\
\emptyset, & \text{for the other cases.} \end{cases}\)

For any morphism of FOL signatures \(\phi : (S, F, P) \to (S', F', P')\) let \([\phi] : ([S], [F], [P]) \to ([S'], [F'], [P'])\) be a morphism of FOL signatures defined as follows:

- \([\phi]_{st}(ST) = ST,\)
- \([\phi]_{st}(s) = \phi_{st}(s)\) for any \(s \in S,\)
- \([\phi]_{op}^{ST}_{ar' \to s}(\sigma) = \phi_{op}^{ST}_{ar' \to s}(\sigma)\) for any \(\sigma \in \mathcal{F}_{ar' \to s},\)
- \([\phi]_{rl}^{ST}_{ar'}(\pi) = \phi_{rl}^{ST}_{ar'}(\pi)\) for any \(\pi \in \mathcal{P}_{ar'}\).

The translation defined above consists of the state-parametrisation of FOL-sentences.

Definition 4.1.1 For any FOL-signature \((S, F, P)\) and any new constant \(x\) of sort \(ST\), the translation

\[ [\_]^x_{(S,F,P)} : \text{Sen}^{FOL}(S, F, P) \to \text{Sen}^{FOL}([S], [F] + x, [P]) \]

is defined by

- \([t = t']^x = ([t]^x = [t']^x)\) where \([\sigma(t_1, \ldots, t_n)]^x = \sigma(x, [t_1]^x, \ldots, [t_n]^x);\)
- \([\pi(t)]^x = \pi(x, [t]^x);\)
- \([p_1 \star p_2]^x = [p_1]^x \star [p_2]^x,\) for \(\star \in \{\lor, \land, \Rightarrow\};\)
- \([-p]^x = -[p]^x;\)
• \([\forall Y]\rho^x = (\forall Y)([\rho]^x)_{Y}\) where \([\rho]^x\) is the result of replacing in \([\rho]^x\) all occurrences of \(y(z)\) by \(y\) for each \(y\) in \(Y\).

Although for all the other cases the definition is as expected, the clause involving quantification may require further explanation. It goes as follows: without the suggested replacement, variables would be translated into unary-functions instead of to constants, i.e., into second-order variables in the place of first-order ones.

Next definition introduces definability predicates:

**Definition 4.1.2** For a FOL-signature \((S, F, P)\) let define \(D_F = \{D_\sigma \mid \sigma \in F_{ar \rightarrow s}, \text{ar} \in S^*, s \in S\}\), where

- For any \(s \in S\), \(D_s\) is a new designated relation symbol with arity \((ST)s\);
- For any \(\sigma \in F_{s_1 \ldots s_n \rightarrow \text{ar}}\), \(D_\sigma\) is the Horn sentence
  \[
  (\forall y)(\forall x_1, \ldots, x_n) \bigwedge_{1 \leq i \leq n} D_{s_i}(y, x_i) \Rightarrow D_s(y, \sigma(y, x_1, \ldots, x_n))
  \]

Bearing in mind that the intuitive meaning of \(D_s(w, m)\) is that “\(m\) belongs to the \(s\)-carrier of the state \(w\)”, clause \(D_\sigma\) assures the definability of \(\sigma\) with respect to “these universes”. Next definition formalises the notion of a “slice” suggested above:

**Definition 4.1.3** For any FOL-signature \((S, F, P)\), any \(([S], [F], [P])\)-model \(M'\) such that \(M' \models D_F\) and for any \(w \in M'_{ST}\) the \((S, F, P)\)-model \(M'_{|w}\) is defined as follows:

- for each \(s \in S\), \((M'_{|w})_s = \{m \in M'_s \mid (w, m) \in M'_D_s\}\);
- for each \(\sigma\) in \(F\), \((M'_{|w})_\sigma(m) = M'_\sigma(w, m)\);
- for each \(\pi\) in \(P\), \(m \in (M'_{|w})_\pi\) iff \((w, m) \in M'_\pi\).

Note the correctness of the definition of \(M'_{|w}\), i.e. the fact that for each \(\sigma \in F_{ar \rightarrow s}\) and each \(m \in (M'_{|w})_{ar}\), \((M'_{|w})_\sigma(m) \in (M'_{|w})_s\), relies upon the condition \(M' \models D_F\).

**Notation 4.1.2** For any \((S, F, P)\)-sentence \(\rho\), we denote by \(V(\rho)\) the set of all sentences \((\forall x, y)D_s(x, y)\) for \(s\) any sort of a variable in a quantification that occurs in \(\rho\). For any set \(E\) of sentences, \(V(E)\) denotes \(\cup\{V(\rho) \mid \rho \in E\}\).
Next lemma establishes the relation between the state-parametrisation and the slice procedure whenever the axiomatisation of definability $D_F$ is verified:

**Lemma 4.1.1** For any FOL-signature $(S, F, P)$, any $([S], [F], [P])$-model $M'$ with $M' \models D_F$, any $(S, F, P)$-sentence $\rho$, and any $w \in M'_{ST}$, if $M' \models V(\rho)$ then

$$M'|_w \models (S, F, P) \rho \text{ if and only if } M'^w \models ([S], [F] + x, [P]) [\rho]^x \quad (8)$$

where $M'^w$ denotes the expansion of $M'$ to $([S], [F] + x, [P])$ defined by $M'^w = w$.

**Proof.** The proof is by induction on the structure of $\rho$.

1. When $\rho$ is $t = t'$ the lemma is an immediate consequence of the following relation

$$M'|_w |_{t} = (M'^w)_{[t]} x,$$  

for any term $t$  

which is proved by induction on $t$ as follows:

$$
(M'|_w)_{\sigma(t_1, \ldots, t_n)} \\
= \{ \text{defn of evaluation of terms} \} \\
(M'|_w)_{\sigma((M'|_w)_{t_1}, \ldots, (M'|_w)_{t_n})} \\
= \{ \text{defn of } M'|_w \} \\
M'_{\sigma(w, (M'|_w)_{t_1}, \ldots, (M'|_w)_{t_n})} \\
= \{ \text{because } M'^w = w \text{ and defn of term evaluation} \} \\
M'^w_{\sigma(x, [t_1]^x, \ldots, [t_n]^x)} \\
= \{ \text{defn of } [\cdot]^x \} \\
M'^w_{\sigma(t_1, \ldots, t_n))^x}
$$

2. When $\rho$ is $\pi(t_1, \ldots, t_n)$:

$$M'|_w \models \pi(t_1, \ldots, t_n)$$
\[
(M'_{|w})_{t_1}, \ldots, (M'_{|w})_{t_n} \in (M'_{|w})_{\pi} \\
\Leftrightarrow \{ \text{defn of } (M'_{|w})_{\pi} \text{ and by (9)} \} \\
(\omega, M'_{[t_1]}_{x_1}, \ldots, M'_{[t_n]}_{x_n}) \in M'_{\pi} \\
\Leftrightarrow \{ \text{because } M'_{\pi} = \omega \} \\
M'_{\pi} = \pi(\omega, [t_1]_{x_1}, \ldots, [t_n]_{x_n})
\]

3. When \( \rho \) is \( \xi_1 \star \xi_2 \) for \( \star \in \{ \land, \lor, \Rightarrow \} \) or \( \rho \) is \( \neg \xi \), the proof reduces to an immediate application of the recursion hypothesis.

4. When \( \rho \) is \( (\forall Y)\xi \):

\[
M'_{|w} \models (\forall Y)\xi \text{ iff } M'' \models \xi \text{ for any } (S, F + x, P)\text{-expansion } M'' \text{ of } M'_{|w}, \text{ and} \\
M''_{|w} \models (\forall Y)(([\xi]_{x})_{Y} \text{ iff } N''_{|w} \models ([\xi]_{x})_{Y} \text{ for any } ([S], [F] + Y + x, [P])\text{-expansion } N''_{|w} \text{ of } M''_{|w}.
\]

The proof proceeds by showing the equivalence between the right hand sides of the two equivalences above. This follows from the following facts:

- There is a canonical bijection between \( ([S], [F] + Y + x, [P])\text{-expansion } N''_{|w} \text{ and } (S, F + Y, P)\text{-expansion } M'' \text{ of } M'_{|w} \) given by \( M''_{|w} = N''_{|w} \) for each \( y \in Y \). This relies upon equality \( (M'_{|w})_{s} = M'_{s} \) which follows from \( M' \models V(\rho) \).

- Each \( N''_{|w} \) as above determines an \( ([S], [F] + Y + x, [P])\text{-expansion } N''_{|w} \text{ by } N''_{|w}(m) = N''_{|w} \) for all \( m \in M'_{ST} \) and each \( y \in Y \). Furthermore

\[
N''_{|w} \models [\xi]_{x} \text{ if and only if } N''_{|w} \models ([\xi]_{x})_{Y}. \quad (10)
\]

- \( M'' = N''_{|w} \).

The induction hypothesis entails \( M'' \models \xi \text{ iff } N''_{|w} \models [\xi]_{x} \). By (10) it follows that \( M'' \models \xi \text{ iff } N''_{|w} \models ([\xi]_{x})_{Y} \).
### 4.1.2 The standard translation lifting

Let \((\text{Sign}^\mathcal{H}_I, \text{Sen}^\mathcal{H}_I, \text{Mod}^\mathcal{C}_I, \models)\) be a hybridisation of an institution \(I\) such that, for all \(\chi \in \mathcal{D}^\mathcal{H}_I\),

- \(\chi_{\text{Nom}}\) are finite extensions, and
- \(\chi_{\text{MS}}\) are identities.

Given any comorphism
\[
(\Phi, \alpha, \beta) : I \rightarrow FOL^\text{pres}
\]
such that for each \(\phi : \Sigma \rightarrow \Sigma'\) in \(\mathcal{D}^I\) we have that

- the underlying \(FOL\) signature morphism of \(\Phi(\phi)\) is in \(\mathcal{D}^{FOL}\), and
- the difference between the presentations \(\Phi(\Sigma')\) and \(\Phi(\Sigma)\) consists of a finite set \(\Gamma_\phi\) of sentences,

we define another comorphism
\[
(\Phi^C, \alpha', \beta^C) : (\text{Sign}^\mathcal{H}_I, \text{Sen}^\mathcal{H}_I, \text{Mod}^\mathcal{C}_I, \models) \rightarrow FOL^\text{pres}
\]
in two steps:

1. We define a functor \(\Phi' : \text{Sign}^\mathcal{H}_I \rightarrow \text{Sign}^{FOL^\text{pres}}\), a natural transformation \(\alpha' : \text{Sen}^\mathcal{H}_I \Rightarrow \Phi';\text{Sen}^{FOL^\text{pres}}\), and a natural transformation \(\beta' : \Phi'^\text{op};\text{Mod}^{FOL^\text{pres}} \Rightarrow \text{Mod}^\mathcal{H}_I\).

2. We extend the definitions of \(\Phi'\) and \(\beta'\) to \(\Phi'^C\) and \(\beta'^C\) respectively. Then we prove the Satisfaction Condition for \((\Phi'^C, \alpha'^C, \beta'^C)\).

Let us start at step (1), with the translation of signatures. As mentioned before, beyond the introduction of state-parameter, we have to transform nominals and modalities into standard constants and predicates over \(ST\). This construction is formalized as follows:

**Definition 4.1.4 (Translation of signatures)** For any \(\mathcal{H}_I\) signature \((\Sigma, \text{Nom}, \Lambda)\), let

\[
\Phi'(\Sigma, \text{Nom}, \Lambda) =
\]

\[
([S_\Sigma], [F_\Sigma] + \overline{\text{Nom}}, (D_s)_{s \in S_\Sigma} + [P_\Sigma] + [\Lambda], F_\Sigma \cup D_{F_\Sigma})
\]

where
• \( \Phi(\Sigma) = ((S_\Sigma, F_\Sigma, P_\Sigma), \Gamma_\Sigma) \), where \((S_\Sigma, F_\Sigma, P_\Sigma)\) is a FOL-signature and \(\Gamma_\Sigma\) is a set of \((S_\Sigma, F_\Sigma, P_\Sigma)\)-sentences;

• \( (\text{Nom})_{\text{ar} \rightarrow s} = \begin{cases} \text{Nom} & \text{when } \text{ar} = \emptyset, s = \text{ST}, \\ \emptyset & \text{for the other cases} \end{cases} \)

• \( (\exists)_{\text{ar}} = \begin{cases} \Lambda_n & \text{when } \text{ar} = (\text{ST})^n, n \in \mathbb{N} \\ \emptyset & \text{for the other cases} \end{cases} \)

• \( T_\Sigma = \{ \forall x [\gamma]^x | \gamma \in \Gamma_\Sigma \} \cup V(\Gamma_\Sigma) \).

The translation of signature morphisms by \( \Phi' \) extend translations via \( \Phi \) as follows:

**Definition 4.1.5 (Translation of the signature morphisms)** For any \( \mathcal{H} \mathcal{I} \) signature morphism
\( \varphi = (\varphi_{\text{Sign}}, \varphi_{\text{Nom}}, \varphi_{\text{MS}}) : (\Sigma_1, \text{Nom}_1, \Lambda_1) \rightarrow (\Sigma_2, \text{Nom}_2, \Lambda_2) \) the FOL\textsuperscript{pres} signature morphism \( \Phi'(\varphi) \) is the extension of
\[ \lceil \Phi(\varphi_{\text{Sign}}) \rceil : ([S_{\Sigma_1}], [F_{\Sigma_1}], [P_{\Sigma_1}]) \rightarrow ([S_{\Sigma_2}], [F_{\Sigma_2}], [P_{\Sigma_2}]) \]
defined by

• \( \Phi'(\varphi)^{\text{op}}(n) = \varphi_{\text{Nom}}(n) \) for each \( n \in \text{Nom}_1 \), and

• \( \Phi'(\varphi)^{\text{rl}}(D_s) = D_{\Phi(\varphi_{\text{Sign}})^{\text{st}}(s)} \) for each sort \( s \in S_{\Sigma_1} \),

• \( \Phi'(\varphi)^{\text{rl}}(\lambda) = \varphi_{\text{MS}}(\lambda) \) for each \( \lambda \in \Lambda_1 \).

**Fact 4.1.1** \( \Phi'(\varphi) \) of Definition 4.1.5 is a presentation morphism
\[ \Phi'(\Sigma_1, \text{Nom}_1, \Lambda_1) \rightarrow \Phi'(\Sigma_2, \text{Nom}_2, \Lambda_2) \).

Next definition establishes the translation of the sentences. The key idea is to use the principle underlying the standard translation \( \mathcal{H}\text{FOL}\mathcal{L}\text{FOL} \) to lift \( \mathcal{I}\text{2FOL} \) into \( \mathcal{H}\mathcal{I}\text{2FOL} \). For this purpose, the translation of the basic \( \mathcal{I} \)-sentences by \( \mathcal{I}\text{2FOL} \) is followed by the application of the state-parametrisation procedure, extending the standard translation principle in all the hybrid features (nominals and modalities) accordingly. For sake of clarity, and without any loss of generality, we have treated, in the following translation, the quantification by nominals and by base institution signature morphisms separately. Moreover, we have considered single variables instead of finite blocks of variables:
Definition 4.1.6 (Translation of sentences) The translation of the sentences is given by \( \alpha'_{\Sigma,\text{Nom},\Lambda}(\rho) = (\forall x)\alpha^x_{\Sigma,\text{Nom},\Lambda}(\rho) \), where

\[
\alpha^x_{\Sigma,\text{Nom},\Lambda} : \text{Sen}^{\text{FCF}}(\Sigma, \text{Nom}, \Lambda) \\
\rightarrow \text{Sen}^{\text{FOL}}([S_{\Sigma}], [F_{\Sigma}] + \overline{\text{Nom}} + x_\ast (D_s)_{s \in S} + [P_{\Sigma}] + \overline{\Lambda})
\]

with \( x \) a constant of sort \( \text{ST} \), is defined by

- \( \alpha^x(i) = (i = x) \), \( i \in \text{Nom} \);
- for each \( \rho \in \text{Sen}^2(\Sigma) \), \( \alpha^x(\rho) = [\alpha_x(\rho)]^x \);
- \( \alpha^x(p_1 \cdot p_2) = \alpha^x(p_1) \ast \alpha^x(p_2) \), \( \ast \in \{\vee, \wedge, \Rightarrow\} \);
- \( \alpha^x(\neg \rho) = \neg \alpha^x(\rho) \);
- \( \alpha^x(\@_i \rho) = \alpha^i(\rho) \);
- \( \alpha^x([\lambda](\rho_1, \ldots, \rho_n)) = \forall y_1, \ldots, y_n \ (\lambda(x, y_1, \ldots, y_n) \Rightarrow \bigvee_{1 \leq i \leq n} \alpha^y_i(\rho_i)) \);
- \( \alpha^x(\lambda(\rho_1, \ldots, \rho_n)) = \exists y_1, \ldots, y_n \ (\lambda(x, y_1, \ldots, y_n) \wedge \bigwedge_{1 \leq i \leq n} \alpha^y_i(\rho_i)) \);
- \( \alpha^x_{\Sigma,\text{Nom},\Lambda}((\forall i)\rho) = (\forall i)\alpha^x_{\Sigma,\text{Nom} + t,\Lambda}(\rho) \) and \( \alpha^x_{\Sigma,\text{Nom},\Lambda}((\exists i)\rho) = (\exists i)\alpha^x_{\Sigma,\text{Nom} + t,\Lambda}(\rho) \) for \( \rho \in \text{Sen}^{\text{FCF}}(\Sigma, \text{Nom} + i, \Lambda) \);
- \( \alpha^x_{\Delta}((\forall \chi)\rho) = (\forall y)(\Gamma_{\text{XSign}} \cup \{D_y\}) \Rightarrow \alpha^x_{\Delta}(\rho) \) and \( \alpha^x_{\Delta}((\exists \chi)\rho) = (\exists y)(\Gamma_{\text{XSign}} \cup \{D_y\}) \Rightarrow \alpha^x_{\Delta}(\rho) \) for \( \rho \in \text{Sen}^{\text{FCF}}(\Delta') \) (where \( \chi = (\chi_{\text{Sign}}, 1_{\text{Nom}}, 1_{\Lambda}) : \Delta = (\Sigma, \text{Nom}, \Lambda) \rightarrow \Delta' = (\Sigma', \text{Nom}, \Lambda) \) and \( \Phi(\chi_{\text{Sign}}) \) extends the signature of \( \Phi(\Sigma) \) with the variable \( y \) and the presentation \( \Phi(\Sigma) \) with the finite set of sentences \( \Gamma_{\chi_{\text{Sign}}} \)).

From the naturality of \( \alpha \) it follows:

Fact 4.1.2 \( \alpha' \) is natural transformation.

We turn now to the translation of models. The idea of recovering a model \( \langle M, W \rangle \in \text{Mod}^3(\Delta) \) from a model \( M \in \text{Mod}^{\text{FOL-pres}}(\Phi'((\Delta))) \) is rather natural. The relational part of the model is directly extracted from the \( \text{[ST]} \)-component of \( M \) (with the respective constants and predicates). Then, the (local) \( \mathcal{J} \)-models are built, state by state, applying the base models translation of \( \mathcal{J}2\text{FOL} \) to the respective slice. Formally:
Definition 4.1.7 (Translation of models) For any $\mathcal{H}\mathcal{I}$ signature $(\Sigma, \text{Nom}, \Lambda)$ and any $\Phi'(\Sigma, \text{Nom}, \Lambda)$-model $M'$, define

$$\beta'_{(\Sigma, \text{Nom}, \Lambda)}(M') = (M, W)$$

where

- $W$ is the reduct $M'|_{((\text{ST}), \text{Nom}, \Lambda)}$, i.e. $|W| = M'|_{\text{ST}}$, $W_i = M'_i$ for each $i \in \text{Nom}$, and $W_\lambda = M'_\lambda$, for each $\lambda \in \Lambda$, and
- $M' : |W| \to |\text{Mod}^\mathcal{I}(\Sigma)|$ is defined, for each $w \in |W|$, by $M'_w = \beta_\Sigma(M'|_w)$, where $M'|_w$ abbreviates $(M'|_{(\Sigma, \text{IF}, \text{FP})})|_w$.

As expected:

Lemma 4.1.2 Definition 4.1.7 is correct, in the sense that for each $w \in |W|$, $M'|_w \models \Gamma_\Sigma$.

Proof. Since $M' \models V(\Gamma_\Sigma) \cup \text{DF}_\Sigma$ we may apply the conclusion of Lemma 4.1.1 from the right to the left for each $\gamma \in \Gamma_\Sigma$. In order to do this note that, because $M' \models \Gamma_\Sigma$, we have $M' \models (\forall x)[\gamma]^x$, for each $\gamma \in \Gamma_\Sigma$. Hence $M'_w \models [\gamma]^x$ for each $\gamma \in \Gamma_\Sigma$. \hfill $\Box$

Having defined the functor $\Phi'$ and the natural transformations $\alpha'$ and $\beta'$, we have to adjust them to the constrained models. In particular, the semantical restrictions of a constrained models functor should be reflected on the semantics of the respective encoding. For this we resort to a functor $C : \text{Sign}^{\mathcal{H}\mathcal{I}} \to \text{Sign}^{\text{FOLpres}}$ such that the constraints of $\text{Mod}^C(\Delta)$ could be taken into account in the axiomatization of $C(\Delta)$. Given this functor, we extend $\Phi'$ and $\beta'$ to $\Phi'/C$ and $\beta'/C$ accordantly.

Thus,

Definition 4.1.8 A functor $C$ is compatible with $\Phi'$ when the diagram below commutes

$$\begin{array}{ccc}
\text{Sign}^{\mathcal{H}\mathcal{I}} & \xrightarrow{\Phi'} & \text{Sign}^{\text{FOLpres}} \\
C \downarrow & & \downarrow \text{U} \\
\text{Sign}^{\text{FOLpres}} & \xrightarrow{\text{U}} & \text{Sign}^{\text{FOL}}
\end{array}$$

where $\text{U}$ denotes the forgetful functor.

For $C$ compatible with $\Phi'$, let
4.1 HYBRIDISING FOL-ENCODINGS

- \( \Phi^{IC} \) denote the functor that represents the componentwise union of the corresponding presentations, i.e. \( \Phi^{IC}(\Delta) \) is the union of \( \Phi'(\Delta) \) and \( C(\Delta) \), and

- \( \beta^{IC} : \Phi^{IC}; \text{Mod}^{FOL_{pres}} \Rightarrow \text{Mod}^{\mathcal{H}2\mathcal{F}OL} \) denote the corresponding (componentwise) restriction of \( \beta' \).

Next theorem characterises the conditions on \( C \) for which the intended comorphism \( \mathcal{H}2\mathcal{F}OL \) holds. The inherent technicality of the conditions is justified with the high level of abstraction achieved, which allows a proper treatment of some complex issues, such as quantifications. In particular,

- **1** assures the suitability of \( C \) w.r.t \( \text{Mod}^{C} \);
- **2** assures the definability of the FOL-quantifications resulting from the translation of basic sentences;
- **3** assures that the adequacy of \( \chi_{\text{Sign}} \) is not lost on the restriction of \( \beta^{IC} \); and that the quantifications of \( \chi_{\text{Sign}} \) are in fact first-order quantifications.

Hence,

**Theorem 4.1.1** Assume a functor \( C \) compatible with \( \Phi' \) such that

1. For each \( \mathcal{H}2\mathcal{F}OL \)-signature \( \Delta \) and each \( M' \in |\text{Mod}^{FOL_{pres}}(\Phi^{IC}(\Delta))| \), \( \beta_{\Delta}^{I}(M') \in |\text{Mod}^{C}(\Delta)| \).

2. For any \( \mathcal{H}2\mathcal{F}OL \)-signature \( \Delta = (\Sigma, \text{Nom}, \Lambda) \) and for any \( \Sigma \)-sentence \( \xi \),

\[
\Phi^{IC}(\Delta) \models V(\alpha_{\Sigma}(\xi)).
\]  

(11)

3. Each signature morphism \( (\chi : \Delta \rightarrow \Delta') \in D^{\mathcal{H}2\mathcal{F}OL} \) with \( \chi_{\text{Nom}} = 1_{\text{Nom}} \)

- is adequate for \( \beta^{IC} \); and
- satisfies

\[
C(\Delta') \models C(\Delta) \cup \{ (\forall z_{1}, z_{2}) y(z_{1}) = y(z_{2}) \mid y \in Y \}. \quad (12)
\]

(where the signature of \( \Phi(\chi_{\text{Sign}}) \) adds the finite block of variables \( Y \) to the signature of \( \Phi(\Sigma) \)).
Then, for any $\Delta = (\Sigma, \text{Nom}, \Lambda) \in |\text{Sign}^{3G}|$, $\rho \in \text{Sen}^{3G}(\Delta)$, $M' \in |\text{Mod}^{\text{FOLpres}}(\Phi^C(\Delta))|$ and $w \in M'_{ST}$,

$$\beta^C_\Delta(M') \models^w \rho \text{ if and only if } M'^w \models_{\Phi^C(\Delta)+x} \alpha^x_\Delta(\rho),$$

(13)

where (as in Lemma 4.1.1) $M'^w$ denotes the expansion of $M'$ to $\Phi^C(\Delta)+x$ defined by $M'^w_{ST} = w$.

**Proof.** The proof is by induction on the structure of $\rho$. Let us denote $\beta_\Delta(M')$ by $(M, W)$.

1. If $\rho = i$, for some $i \in \text{Nom}$

$$(M, W) \models^w_i \iff \{ \text{ defn. of $\models^w_i$} \}
W_i = w
\iff \{ \text{ defn $\beta'$ and of $M'^w$} \}
M'_i(= M'^w_i) = M'^w
\iff \{ \text{ defn FOLpres-satisfaction} \}
M'^w \models_{\Phi^C(\Delta)+x} i = x
\iff \{ \text{ defn of $\alpha^x$} \}
M'^w \models_{\Phi^C(\Delta)+x} \alpha^x(i)$$

2. If $\rho \in \text{Sen}^3(\Sigma)$:

$$(M, W) \models^w \rho \iff \{ \text{ defn. of $\models^w$} \}
M_w \models^w \rho
\iff \{ \text{ defn. of $\beta'$} \}
\beta_\Sigma(M'_w) \models_\Sigma \rho
\iff \{ \text{ by the satisfaction condition of $(\Phi, \alpha, \beta)$} \}
M'_w \models_{\Phi(\Sigma)} \alpha_\Sigma(\rho)$$
\[\Longleftrightarrow \{ \text{by (11) and Lemma 4.1.1}\}\]
\[M^\mathcal{W} \models \Phi_{(\Delta)+x}\left[\alpha_\Sigma(\rho)\right]_{x}\]
\[\Longleftrightarrow \{ \text{defn of } \alpha^x\}\]
\[M^\mathcal{W} \models \Phi_{(\Delta)+x}\alpha^\mathcal{W}(\rho)\]
\[\Longleftrightarrow \{ \text{by (11) and Lemma 4.1.1}\}\]
\[M^\mathcal{W} \models \Phi_{(\Delta)+x}\left[\alpha_\Sigma(\rho)\right]_{x}\]
\[\Longleftrightarrow \{ \text{defn } \alpha^x\}\]
\[M^\mathcal{W} \models \Phi_{(\Delta)+x}\alpha^\mathcal{W}(\rho)\]

3. If \(\rho = \mathcal{W}_i\xi\):

\[(M, W) \models_{\Delta}^\mathcal{W} \mathcal{W}_i\xi\]
\[\Longleftrightarrow \{ \text{defn. of } \models_{\Delta}\}\]
\[(M, W) \models_{\Delta}^\mathcal{W} \xi\]
\[\Longleftrightarrow \{ \text{I.H.}\}\]
\[M^{\mathcal{W}_i} \models \Phi_{(\Delta)+x}\alpha^\mathcal{W}(\xi)\]
\[\Longleftrightarrow \{ \text{since } M_{\mathcal{W}_i}^{\mathcal{W}_i} = W_i = M_{\emptyset}\}\]
\[M' \models \Phi_{(\Delta)}\alpha^\mathcal{W}(\xi)\]
\[\Longleftrightarrow \{ \text{satisfaction condition in FOL}\}\]
\[M^\mathcal{W} \models \Phi_{(\Delta)+x}\alpha^\mathcal{W}(\xi)\]
\[\Longleftrightarrow \{ \text{defn of } \alpha^x\}\]
\[M^\mathcal{W} \models \Phi_{(\Delta)+x}\alpha^\mathcal{W}(\xi)\]

4. If \(\rho = [\lambda](\xi_1, \ldots, \xi_n)\) with \(\lambda \in \Lambda_{n+1}\):

\[(M, W) \models_{\Delta}^\mathcal{W} [\lambda](\xi_1, \ldots, \xi_n)\]
\[\Longleftrightarrow \{ \text{defn. of } \models_{\Delta}\}\]
for any \((w, w_1, \ldots, w_n) \in W\) there exists \(1 \leq i \leq n\) such that \((M, W) \models^{w_i} \xi_i\)

\[\Leftrightarrow \quad \{ \text{I.H.} \}\]

for any \((w, w_1, \ldots, w_n) \in W\) there exists \(1 \leq i \leq n\) such that \(M^{w_i} \models_{\Phi'(\Delta)+y_i}^{\alpha'^{\lambda}(\xi_i)}\)

\[\Leftrightarrow \quad \{ \text{defn of } \models_{\Phi'(\Delta)}^{\lambda(\xi)} \}\]

for all \(w_1, \ldots, w_n\)

\(M^{nw_1 \ldots w_n} \models_{\Phi'(\Delta)+x}^{\lambda(x, y_1, \ldots, y_n)} \\bigvee_{1 \leq i \leq n} \alpha'^{y_i}(\xi_i)\)

\[\Leftrightarrow \quad \{ \text{Rule of Generalization in FOL} \}\]

\(M^{nw} \models_{\Phi'(\Delta)+x}^{\lambda(x, y_1, \ldots, y_n)} \\forall y_1, \ldots, y_n \lambda(x, y_1, \ldots, y_n) \Rightarrow \\bigvee_{1 \leq i \leq n} \alpha'^{y_i}(\xi_i)\)

\[\Leftrightarrow \quad \{ \text{defn. of } \alpha'^{\lambda} \}\]

\(M^{nw} \models_{\Phi'(\Delta)+x}^{\alpha'^{(\lambda([\xi_1, \ldots, \xi_n]))}(\xi)}\)

5. If \(\rho = \xi \lor \xi'\):

\((M, W) \models_{\Delta}^{w} \xi \lor \xi'\)

\[\Leftrightarrow \quad \{ \text{defn. of } \models_{\Delta}^{w} \}\]

\((M, W) \models_{\Delta}^{w} \xi\) or \((M, W) \models_{\Delta}^{w} \xi'\)

\[\Leftrightarrow \quad \{ \text{I.H.} \}\]

\(M^{nw} \models_{\Phi'(\Delta)+x}^{\alpha'^{(\lambda)(\xi)}}\) or \(M^{nw} \models_{\Phi'(\Delta)+x}^{\alpha'^{(\lambda)(\xi')}}\)

\[\Leftrightarrow \quad \{ \text{defn of } \models_{\text{FOLpres}}^{w} \}\]

\(M^{nw} \models_{\Phi'(\Delta)+x}^{\alpha'^{(\lambda)(\xi)}} \lor \alpha'^{(\lambda)(\xi')}\)

\[\Leftrightarrow \quad \{ \text{defn of } \alpha'^{\lambda} \}\]

\(M^{nw} \models_{\Phi'(\Delta)+x}^{\alpha'^{(\lambda)(\xi \lor \xi')}}\)
The proofs for cases $\rho = \xi \land \xi'$, $\rho = \xi \Rightarrow \xi'$, $\rho = \neg \xi$, etc. are analogous.

6. If $\rho = (\forall i)\xi$:

\[(M, W) \models^w_\Delta (\forall i)\xi,\]

\[\iff\]

\[\{ \text{defn. of } \models_\Delta \} \]

\[(M, W') \models^w_{\Delta+i} \xi \text{ for each}\]

\[(\text{Nom }+ i, \Lambda)\text{-expansion } W' \text{ of } W\]

\[\iff\]

\[\{ \text{I.H.} \} \]

\[N^w \models \Phi'(\Delta)+x \alpha^\Delta_{\Delta+i}(\xi) \text{ for each}\]

\[(\Phi'(\Delta) + i)\text{-expansion } N \text{ of } M'\]

\[\iff\]

\[\{ \text{defn. of } \models_\Delta \} \]

\[M^w \models \Phi'(\Delta)+x \alpha^\Delta_{\Delta+i}(\xi)\]

\[\iff\]

\[\{ \text{defn. of } \alpha^\Delta_i \} \]

\[M^w \models \Phi'(\Delta)+x \alpha^\Delta_{\Delta+i}(\forall i)\xi\]

7. If $\rho = (\forall \chi)\xi$:

Let $\chi : \Delta = (\Sigma, \text{Nom}, \Lambda) \rightarrow \Delta' = (\Sigma', \text{Nom}, \Lambda)$. We have to prove that

\[(M, W) \models^w_\Delta (\forall \chi)\xi,\]

if and only if

\[M^w \models \Phi'(\Delta)+x (\forall y)(\overline{\Gamma_{\chi \text{Sign}} \cup \{D_y\}} \Rightarrow \alpha^\Delta_{\Delta+i}(\xi))_y.\]

\[\Rightarrow\]

Let $N'$ be any $(\Phi'(\Delta) + x)$-expansion of of $M'$. We have to prove that

\[N^w \models \Phi'(\Delta)+x+y (\overline{\Gamma_{\chi \text{Sign}} \cup \{D_y\}} \Rightarrow \alpha^\Delta_{\Delta+i}(\xi))_y\]

which is equivalent to proving that

\[N''^w \models \Phi'(\Delta')+x (\overline{\Gamma_{\chi \text{Sign}} \cup \{D_y\}} \Rightarrow \alpha^\Delta_{\Delta'i}(\xi))\]
where $N''$ is the $\Phi'(\Delta')$-expansion of $M'$ defined by $N''(z) = N''_y$. Let us assume that $N^{nw} \models \Gamma_{X_{\text{Sign}}} \cup \{D_y\}$. We have the following:

- $N'' \models \Gamma_{X_{\text{Sign}}} \cup \{D_y\}$ because $\Gamma_X$ and $D_y$ are $\Phi'(\Delta')$-sentences;
- $N'' \models \Phi'^C(\Delta)$ because $N''$ is an expansion of $M'$ and $M' \models \Phi'^C(\Delta)$; and
- $N'' \models (\forall z_1, z_2) y(z_1) = y(z_2)$.

From these three satisfactions and from (12) it follows that $N'' \models \Phi'^C(\Delta')$. Consequently

$$\beta'^C_{\Delta'}(N'') \models x = \beta'^C_{\Delta'}(N'' \models \Phi'(\chi)) = \beta'^C_{\Delta'}(M') = (M, W)$$

hence $\beta'^C_{\Delta'}(N'') \models \xi$. By the induction hypothesis it follows that $N^{nw} \models \alpha'^x_{\Delta'}(\xi)$.

$\Leftarrow$ Let $(N, W)$ be a $\chi$-expansion of $(M, W)$. We have to prove that $(N, W) \models^w \xi$. By the adequacy hypothesis there exists a $\Phi'^C(\Delta')$-model $N''$ such that $(N, W) = \beta'^C_{\Delta'}(N')$ and $M' = N'' \models \Phi'^C(\chi)$. By the induction hypothesis is suffices to prove that $N^{nw} \models \Phi'(\Delta') + x \alpha'^x_{\Delta'}(\xi)$.

We let $N'$ be the $(\Phi'(\Delta) + y)$-expansion of $M'$ defined by $N'_y = N''(z)$. This definition is correct because $N' \models C(\Delta')$ which by (12) implies that $N''(z)$ is invariant with respect to $z$. Since $N^{nw}$ is an expansion of $M^{nw}$, by hypothesis it follows that

$$N^{nw} \models \Phi'(\Delta) + y + x \ (\Gamma_{X_{\text{Sign}}} \cup \{D_y\} \Rightarrow \alpha'^x_{\Delta'}(\xi))_y.$$  

which implies

$$N^{nw} \models \Phi'(\Delta') + x \ (\Gamma_{X_{\text{Sign}}} \cup \{D_y\} \Rightarrow \alpha'^x_{\Delta'}(\xi)).$$

The desired conclusion follows now from the fact that $N''$, as a $\Phi'^C(\Delta')$-model, satisfies the condition of the implication above.

Finally:
Corollary 4.1.1 (Satisfaction condition for \((\Phi'^C, \alpha', \beta'^C)\)) Under the conditions of Theorem 4.1.1 above, \((\Phi'^C, \alpha', \beta'^C)\) is comorphism
\[
(\text{Sign}^{\mathcal{C}}, \text{Sen}^{\mathcal{C}}, \text{Mod}^C, \models) \to \text{FOL}^{\text{pres}},
\]
i.e. for any \(\Delta \in \text{Sign}^{\mathcal{C}}, \rho \in \text{Sen}^{\mathcal{C}}(\Delta)\) and \(M' \in \text{Mod}^{\text{FOL}^{\text{pres}}}(\langle \Phi'^C(\Delta) \rangle)\),
\[
\beta'^C_{\Delta}(M') \models_{\Delta} \rho \text{ if and only if } M' \models_{\Phi'^C(\Delta)} \alpha'_\Delta(\rho).
\]

4.2 EXAMPLES

We illustrate the method of the previous section with a number of examples.

Example 4.2.1 (\(\mathcal{H}PL2FOL\)) Consider the case of \(\mathcal{H}PL\) discussed in Example 3.2.1. The base comorphism \((\Phi, \alpha, \beta)\) is the canonical embedding of \(PL\) into \(FOL\) presented on Example 2.3.14. This means the \(D\)'s and the \(\Gamma\)'s are empty. The quantification space for the hybridisation consists of extensions with nominal variables. Hence, for any signature \((\text{Prop}, \text{Nom})\),
\[
\Phi'(\text{Prop}, \text{Nom}) = ([\text{ST}], \overline{\text{Nom}}, [\text{Prop}] + \lambda).
\]
Moreover, for any sentence \(\rho \in \text{Sign}^{\mathcal{H}PL}(\text{Prop}, \text{Nom})\),
\[
\alpha'_{\text{Prop,Nom}}(\rho) = (\forall x)\alpha_{\text{Prop,Nom}}^x(\rho),
\]
where
- for each \(\rho \in \text{Sen}^{\mathcal{H}PL}(\text{Prop})\), \(\alpha_{\text{Prop,Nom}}^\chi(\rho) = [\alpha_{\text{Prop}}(\rho)]^\chi\)
- \(\alpha_{\text{Prop,Nom}}^\chi(i) = (i = x), i \in \text{Nom};\)
- \(\alpha_{\text{Prop,Nom}}^\chi(\text{at} \rho) = \alpha_{\text{Prop,Nom}}^\eta(\rho);\)
- \(\alpha_{\text{Prop,Nom}}^\chi([\lambda] \rho = (\forall y) \lambda(x, y) \Rightarrow \alpha_{\text{Prop,Nom}}^\eta(\rho);\)
- \(\alpha_{\text{Prop,Nom}}^\chi(\rho_1 \star \rho_2) = \alpha_{\text{Prop,Nom}}^\chi(\rho_1) \star \alpha_{\text{Prop,Nom}}^\chi(\rho_2), \star \in \{\lor, \land, \Rightarrow\};\)
- \(\alpha_{\text{Prop,Nom}}^\chi(\lnot \rho) = -\alpha_{\text{Prop,Nom}}^\chi(\rho);\)
- \(\alpha_{\text{Prop,Nom}}^\chi((\forall i) \rho) = (\forall i)\alpha_{\text{Prop,Nom}+\{i\}}^\chi(\rho).\)
Observe that, as expected, the transformation $\alpha'$ coincides with the so-called “standard translation” for the propositional hybrid logic (e.g. [Bla00, Bra10]).

The models translation, $M' \in \text{Mod}^{\mathcal{PL}}(\Phi'(\text{Prop}, \text{Nom})) = (\text{Mod}^{\mathcal{PL}}(\{ST\}, \overline{\text{Nom}}, \text{[Prop]} + \lambda)), \beta'_{\text{Prop,Nom}}(M') = (M, W)$ is obtained as follows:

- $W = M'|_{\langle\{ST\}, \overline{\text{Nom}}, \lambda\rangle}$;
- for any $w \in W$, $M_w = \beta_{\text{Prop}}((M'|_{\text{Prop}})|w) = (M'|_{\text{Prop}})|w$, i.e., for any $p \in \text{Prop}, (M_w)_p$ iff $w \in M'_p$.

Hence, conditions (11) and (12) of Theorem 4.1.1 are vacuously satisfied.

Let us consider now the case of T-hybrid propositional logic (see Example 3.2.1). In this case we have to constrain the models that the accessible relation $W_\lambda$ is reflexive. The functor $C$ is such that each $C(\text{P}, \text{Nom})$ is the presentation containing the sentence $(\forall x)\lambda(x, x)$. Note that $\Phi'^C$ maps any signature $(\text{P}, \text{Nom})$ to the FOL-presentation $(\langle\{ST\}, \overline{\text{Nom}}, \text{[P]} + \lambda\rangle, (\forall x)\lambda(x, x))$. Again, conditions (11) and (12) of Theorem 4.1.1 are vacuously satisfied, as well as the adequacy condition for $\beta'^C$ (of the same theorem).

**Example 4.2.2 ($\mathcal{HTRM}2\mathcal{FOL}$)** Let us consider the comorphism $\mathcal{HTRM}2\mathcal{FOL}$ of Example 2.3.13. Then the method yields

$$\Phi'(\ast, \text{Nom}, \Lambda) = (\langle\text{St}\rangle, \overline{\text{Nom}}, p, : \text{ST} + \overline{\Lambda}), \emptyset).$$

Moreover, for any sentence $\rho \in \text{Sign}^{\mathcal{HTRM}}(\ast, \text{Nom}, \Lambda),$

$$\alpha'_{\ast,\text{Nom},\Lambda}(\rho) = (\forall x)\alpha^x(\rho),$$

where

- $\alpha^x(i) = (i = x), i \in \text{Nom};$
- $\alpha^x(@_i\rho) = \alpha^x_i(\rho);$  
- $\alpha^x(\langle\lambda\rangle(\rho_1, \ldots, \rho_n)) = (\forall y_1, \ldots, y_n (\lambda(x, y_1, \ldots, y_n) \Rightarrow \bigvee_{1 \leq i \leq n} \alpha^y_i(\rho_i)));$
- $\alpha^x(\langle\lambda\rangle(\rho_1, \ldots, \rho_n)) = (\exists y_1, \ldots, y_n (\lambda(x, y_1, \ldots, y_n) \wedge \bigwedge_{1 \leq i \leq n} \alpha^y_i(\rho_i))).$
4.2 Examples

- \( \alpha^\land (\rho_1 \ast \rho_2) = \alpha^\land (\rho_1) \ast \alpha^\land (\rho_2), \ast \in \{\lor, \land, \Rightarrow\} \);
- \( \alpha^\land (\neg \rho) = \neg \alpha^\land (\rho) \);

The models translation, \( M' \in \text{Mod}^{\mathcal{H}^\land \mathcal{H}^\land \mathcal{P}L}(\Phi'(\ast, \text{Nom}, \land)) \),

\[
\beta'_{(\text{Prop, Nom})}(M') = (M, W)
\]
is obtained as follows:

- \( W = M'|_{([\text{ST}], \overline{\text{Nom}}, \lambda)} \);
- for any \( w \in W \), \( M_w = * \)

Since we have no \( D_s \) neither \( \Gamma \) and since this is a quantifier-free hybridisation, we have that all the conditions of Theorem 4.1.1 trivially hold.

Example 4.2.3 In the case of the encoding of \( \mathcal{H}^\land \mathcal{H}^\land \mathcal{P}L \) (from Example 3.2.3) the base comorphism is the embedding of the free hybridisation of \( \mathcal{P}L \) into \( \mathcal{F}O\mathcal{L} \); hence (see Example 4.2.1) we have

\[
\Phi(P, \text{Nom}^0) = (\langle \text{ST}^0 \rangle, \overline{\text{Nom}}^0, [P] + \lambda^0)
\]

(we use \( \lambda^0 \) and \( \lambda^1 \) to distinguish the relations underlying \( \Box^0 \) and \( \Box^1 \) respectively). Thus \( \Phi'(P, \text{Nom}^0, \text{Nom}^1) = \)

\[
(\langle \text{ST}^0, \text{ST}^1 \rangle, \overline{\text{Nom}}^0 + \overline{\text{Nom}}^1, D_{\text{ST}^0} + [[P]] + \lambda^0 + \lambda^1, D_{\overline{\text{Nom}}^0}).
\]

As expected we have now a sort of states for each level of hybridisation, i.e. \( \langle \text{ST}^0 \rangle = \langle \text{ST}^0, \text{ST}^1 \rangle \). The predicate \( D_{\text{ST}^0} : \langle \text{ST}^1 \rangle(\text{ST}^0) \) plays the role of a “sub-state-relation”. A nominal \( i^0 \) of \( \text{Nom}^0 \) is interpreted as an operation \( i^0 : \text{ST}^1 \rightarrow \text{ST}^0 \) (of \( \overline{\text{Nom}}^0 \)) and similarly the base modality \( \lambda^0 \) by a predicate \( \lambda^0 : \langle \text{ST}^1 \rangle(\text{ST}^0)(\text{ST}^0) \). Also \( D_{\overline{\text{Nom}}^0} = \{(\forall y)D_{\text{ST}^0}(y, i^0(y)) \mid i^0 \in \text{Nom}^0\} \). In order to get the condition (11) of Theorem 4.1.1 fulfilled, since for any \( \rho \in \text{Sen}^{\mathcal{H}^\land \mathcal{H}^\land \mathcal{P}L}(P, \text{Nom}^0) \) the sentence \( \alpha_{(P, \text{Nom}^0)}(\rho) \) is \( \text{ST}^0 \)-quantified, we take

\[
C(P, \text{Nom}^0, \text{Nom}^1) = \{(\forall x, y)D_{\text{ST}^0}(x, y)\}.
\]

(14)

Note that since \( C(P, \text{Nom}^0, \text{Nom}^1) \models D_{\overline{\text{Nom}}^0} \) we may define

\[
\Phi^C(P, \text{Nom}^0, \text{Nom}^1) = (\langle \text{ST}^0, \text{ST}^1 \rangle, \overline{\text{Nom}}^0 + \overline{\text{Nom}}^1, D_{\text{ST}^0} + [[P]] +
\]

\[
\text{ST}^0 + \text{ST}^1 + \overline{\text{Nom}}^0 + \overline{\text{Nom}}^1 + \lambda^0 + \lambda^1 + D_{\overline{\text{Nom}}^0}.
\]
\[ \lambda^0 + \lambda^1, \{(\forall x, y)D_{ST^0}(x, y)\} \]. Because of the absence of quantifications, the adequacy condition on \( \beta^C \) and the condition (12) of Theorem 4.1.1 hold trivially.

With respect to the quantified versions of \( \mathcal{H}' \mathcal{H}'PL \) the situation is as follows.

- The condition (12) of Theorem 4.1.1 holds trivially for the quantifications \((\forall n^0)\) and \((\forall n^1)\). (In the former case \( \chi \) is identity and in the latter case \( \chi_{\text{Sign}} \) is identity.)

- In the presence of quantifications \((\forall n^0)\), \( \chi_{\text{Sign}} \) adds \( n^0 \) to the \( \mathcal{H}PL \) signature and consequently \( \Phi(\chi_{\text{Sign}}) \) adds \( n^0 \) as a new constant of sort \( ST^0 \). The condition (12) of Theorem 4.1.1 can be fulfilled only for the constrained versions from the \( \mathcal{H}'(\mathcal{H}'PL'x) \) family. In these cases \( C \) has to reflect the sharing of sub-states domains by the sentences \( \{(\forall x, y, z)D_{ST^0}(x, z) \iff D_{ST^0}(y, z)\} \) and the sharing of the interpretations of the base nominals by the sentences \( \{(\forall x, y)i(x) = i(y) \mid i \in \overline{\text{Nom}^0}\} \). Since the former are consequences of (14), we have that
  \[
  C((S, F, P), \text{Nom}, \Lambda) = \{(\forall x, y)D_{s}(x, y) \mid s \in S\}. 
  \]

\[ \circ \]

**Example 4.2.4** Let us now consider the free hybridisation of \( FOL \) with only quantification over nominal variables. The base comorphism \((\Phi, \alpha, \beta)\) is the identity, hence all \( \Gamma \)'s are empty. Hence,

\[
\Phi'((S, F, P), \text{Nom}, \Lambda) = ([S], [F] + \text{Nom}, (D_s)_{s \in S} + [P] + \overline{\Lambda}, D_F).
\]

In order to get the condition (11) of Theorem 4.1.1 fulfilled we define

\[
C((S, F, P), \text{Nom}, \Lambda) = \{(\forall x, y)D_s(x, y) \mid s \in S\}.
\]

Note that \( C((S, F, P), \text{Nom}, \Lambda) \models D_F \). Hence we may write

\[
\Phi'^C((S, F, P), \text{Nom}, \Lambda) = \\
([S], [F] + \text{Nom}, (D_s)_{s \in S} + [P] + \overline{\Lambda}, \{(\forall x, y)D_s(x, y) \mid s \in S\}).
\]
Because in this case only quantifications over nominal variables are allowed the condition (12) of Theorem 4.1.1 is vacuously fulfilled and so is the adequacy condition for $\beta^C$ (of the same theorem).

The variant of this example in which the base institution is the quantifier-free fragment of $\text{FOL}$ rather than the whole of $\text{FOL}$, has all $C$’s empty, and hence $\Phi^C = \Phi'$.

The version of the above variant which considers quantification with first order variables at the level of the hybridisation, requires, to get condition (12) of Theorem 4.1.1 fulfilled, that

$$C((S,F,P),\text{Nom},\Lambda) \models [(\forall z_1, z_2) y(z_1) = y(z_2) \mid y \in F \rightarrow s, s \in S].$$

However because the hybridisation is free there is no way to get the adequacy condition for $\beta^C$. Hence in this case it is not possible to build the encoding comorphism.

**Example 4.2.5** When encoding $\hat{\text{HREL}}'$ (of Example 3.2.5) the base comorphism $(\Phi, \alpha, \beta)$ is the canonical embedding of $\text{REL}$ into $\text{FOL}$ determined by the embedding of $\text{REL}$ signatures as $\text{FOL}$ signatures. Hence all $\Gamma$’s are empty. Thus,

$$\Phi'((C,P),\text{Nom},\Lambda) = (([\text{ST},\star], [C] + \text{Nom}, [D_\star] + [P] + \overline{\Lambda}), D_C).$$

The sharing of the underlying universe requires all $C$’s to contain

$$(\forall x, y, z) D_\star(x, z) \Leftrightarrow D_\star(y, z).$$

However in order to get condition (11) of Theorem 4.1.1 fulfilled $(\forall x, y) D_\star(x, y)$ is also needed. Since the latter sentence implies the former and also implies $D_C$, we can do only with $(\forall x, y) D_\star(x, y)$. Finally, the sharing of the interpretations of constants requires

$$(\forall x, y) \sigma(x) = \sigma(y) \mid \sigma \in C).$$

This also meets (12) of Theorem 4.1.1. Hence: $\Phi^C((C,P),\text{Nom},\Lambda) =$

$$(([\text{ST},\star], [C] + \text{Nom}, [D_\star] + [P] + \overline{\Lambda}), (\forall x, y) D_\star(x, y) \cup
(\forall x, y) \sigma(x) = \sigma(y) \mid \sigma \in C).)$$

It remains to check the adequacy condition for $\beta^C$, which is a very easy task. Let $\Delta$ denote the $\hat{\text{HREL}}$ signature $((C,P),\text{Nom},\Lambda)$. For any block $Y$ of variables for the $\text{REL}$ sig-
nature \((C, P)\), any \(\mathcal{H}REL'\)-model \((N, W)\) for \(\Delta + Y\), and any \(\Phi'^C(\Delta)\)-model \(M'\) such that \((N, W)|_\Delta = \beta'(M')\),

\[
\begin{array}{c}
\text{Mod}^{\mathcal{H}REL'}(\Delta) \xleftarrow{\beta'^C_\Delta} \text{Mod}^{FOL^{pres}}(\Phi'^C(\Delta)) \\
\text{Mod}^{\mathcal{H}REL'}(\Delta + Y) \xleftarrow{\beta'^C_{\Delta + Y}} \text{Mod}^{FOL^{pres}}(\Phi'^C(\Delta + Y))
\end{array}
\]

the amalgamation of \(M'\) and \((N, W)\) is the \(\Phi'^C(\Delta + Y)\)-expansion \(N'\) of \(M'\) defined by \(N'_y(z) = (N_w)_y \in M_s\), for any \(z \in M'_{ST} = |W|\) and any \(w \in |W|\). This definition does not depend on \(s\) because the underlying universe and the interpretation of the constants are shared. Note also that \(N'\) satisfies indeed the sentences of \(\Phi'^C(\Delta + Y)\) since, by the satisfaction condition in \(FOL\), it satisfies the sentences of \(\Phi'^C(\Delta)\) and also \((\forall z_1, z_2) y(z_1) = y(z_2)\) for each \(y \in Y\).

Let us also recall \(\mathcal{H}REL'_0\) in Example 3.2.5. In this case, the “increase of the domains” should be axiomatized, for any \(\lambda \in \Lambda\), as

\[
(\forall x, y, z) [\lambda(x, y) \Rightarrow ((D_\ast(x, z) \Rightarrow D_\ast(y, z))].
\]

Since we have no basic quantification, neither any constraint on the universes or constants sharing, conditions (11) and (12) of Theorem 4.1.1 are trivially fulfilled.

\[\circ\]

Example 4.2.6 In the case of the encoding of \(\mathcal{H}FOLR'\) (see Example 3.2.6) the quantification space \(\mathcal{D}^{\mathcal{H}FOLR}\) consists of extensions with nominal variables and rigid first-order variables. The base comorphism \((\Phi, \alpha, \beta)\) is defined as follows:

1. \(\Phi\) is the forgetful functor \(\text{Sign}^{FOLR} \rightarrow \text{Sign}^{FOL}\) that maps a signature \((S, S_0, F, F_0, P, P_0)\) to \((S, F, P)\),

2. \(\alpha_{(S, S_0, F, F_0, P, P_0)}\) is the inclusion \(\text{Sen}^{FOLR}(S, S_0, F, F_0, P, P_0) \subseteq \text{Sen}^{FOL}(S, F, P)\) (the difference is given by the quantification which in \(FOLR\) is restricted to the rigid symbols), and

3. \(\beta_{(S, S_0, F, F_0, P, P_0)}\) is the identity on \(\text{Mod}^{FOL}(S, F, P)\).
This is a comorphism mapping signatures to signatures. Hence all $\Gamma$’s are empty. Thus,

$$\Phi'((S, S_0, F, F_0, P, P_0)) = ([S], [F] + \overline{\text{Nom}}, D + [P] + \overline{\Lambda}, D_F)$$

The specification of the model constraints requires that $C((S, S_0, F, F_0, P, P_0))$ contains the following sentences:

1. for each $s \in S_0$, $(\forall x, y, z)D_s(x, z) \iff D_s(y, z)$,
2. for each $\sigma$ in $F_0$, $(\forall x, y, Z)\sigma(x, Z) = \sigma(y, Z)$, and
3. for each $\pi$ in $P_0$, $(\forall x, y, Z)\pi(x, Z) \iff \pi(y, Z)$.

Note that these already cover condition (12) of Theorem 4.1.1. For condition (11) of Theorem 4.1.1 we have to add also the sentences $(\forall x, y, z)D_s(x, y)$ for each $s \in S_0$, which are stronger than

$$(\forall x, y, z)D_s(x, z) \iff D_s(y, z).$$

All these together define the functor $C$ that specifies the constraints. Finally, the adequacy condition for $\beta'^C$ is checked as in Example 4.2.5.

**Example 4.2.7** In the case of the encoding of $\mathcal{H}FOLR'$ (see Example 3.2.6) the quantification space $D^{\mathcal{H}FOLR}$ consists of extensions with nominal variables and rigid first-order variables. The base comorphism $(\Phi, \alpha, \beta)$ is defined as follows:

1. $\Phi$ is the forgetful functor $\text{Sign}^{FOLR} \to \text{Sign}^{FOL}$ that maps a signature $(S, S_0, F, F_0, P, P_0)$ to $(S, F, P)$,
2. $\alpha_{(S, S_0, F, F_0, P, P_0)}$ is the inclusion $\text{Sen}^{FOLR}(S, S_0, F, F_0, P, P_0) \subseteq \text{Sen}^{FOL}(S, F, P)$ (the difference is given by the quantification which in $FOLR$ is restricted to the rigid symbols), and
3. $\beta_{(S, S_0, F, F_0, P, P_0)}$ is the identity on $\text{Mod}^{FOL}(S, F, P)$.

This is a comorphism mapping signatures to signatures, hence the $\Gamma$’s are empty. Thus

$$\Phi'((S, S_0, F, F_0, P, P_0), \text{Nom}, \Lambda) = ([S], [F] + \overline{\text{Nom}}, D + [P] + \overline{\Lambda}, D_F)$$

The specification of the model constraints requires that $C((S, S_0, F, F_0, P, P_0))$ contains the following sentences:
1. for each \( s \in S_0 \), \((\forall x, y, z)D_s(x, z) \Leftrightarrow D_s(y, z)\),
2. for each \( \sigma \) in \( F_0 \), \((\forall x, y, Z)\sigma(x, Z) = \sigma(y, Z)\), and
3. for each \( \pi \) in \( P_0 \), \((\forall x, y, Z)\pi(x, Z) \Leftrightarrow \pi(y, Z)\).

Note that these already cover the condition \((12)\) of Theorem 4.1.1. For the condition \((11)\) of Theorem 4.1.1 we have to add also the sentences \((\forall x, y)D_s(x, y)\) for each \( s \in S_0 \), which are stronger than \((\forall x, y, z)D_s(x, z) \Leftrightarrow D_s(y, z)\). All these together define the functor that specifies the constraints. Finally, the adequacy condition for \( \beta^C \) may be checked easily in the same way as in Example 4.2.5; therefore we omit this here.

**Example 4.2.8** In the case of the encoding of \( \mathcal{H} \text{PAR}' \) (see Example 3.2.7) the base comorphism \((\Phi, \alpha, \beta)\) extends canonically the first encoding comorphism \( PA \to \text{FOL}^\text{pres} \), presented in Example 2.3.15, say \((\Phi_0, \alpha_0, \beta_0)\), to a comorphism \( \text{PAR} \to \text{FOL}^\text{pres} \) making

1. \( \Phi \) maps each signature \((S, S_0, \text{TF}, \text{TF}_0, \text{PF}, \text{PF}_0)\) to the presentation \((\langle S, \text{TF} + \text{PF}, (\text{Def}_s)_{s \in S} \rangle, \Gamma_{(S, \text{TF}, \text{PF})})\) where \( \Gamma_{(S, \text{TF}, \text{PF})} \) axiomatizes the definability of terms through the new predicates \((\text{Def}_s)_{s \in S}\) as follows:
   \[
   \Gamma_{(S, \text{TF}, \text{PF})} = \\
   \{ (\forall X)\text{Def}_s(\sigma(X)) \Rightarrow \text{Def}_s(X) \mid \sigma \in (\text{TF} + \text{PF})_{\text{ar} \to s}, \text{ar} \in S^*, s \in S \} \\
   \cup \\
   \{ (\forall X)\text{Def}_s(X) \Rightarrow \text{Def}_s(\sigma(X)) \mid \sigma \in \text{TF}_{\text{ar} \to s}, \text{ar} \in S^*, s \in S \}
   \]
   (where \( \text{Def}_s(X) \) denotes \( \bigwedge_{(x : s) \in X} (\text{Def}_s(x)) \).)
2. \( \alpha_{(S, S_0, \text{TF}, \text{TF}_0, \text{PF}, \text{PF}_0)} \) is recursively defined as follows:
   - \( \alpha(t = t') = \text{Def}_s(t) \land (t = t') \);
   - \( \alpha((\forall X)\rho) = (\forall X)(\text{Def}_s(X) \Rightarrow \alpha(\rho)) \);
   - \( \alpha \) commutes with boolean connectives \( \land, \lor, \Rightarrow, \) etc.
3. \( \beta_{(S, S_0, \text{TF}, \text{TF}_0, \text{PF}, \text{PF}_0)} \) maps any \((\langle S, \text{TF} + \text{PF}, (\text{Def}_s)_{s \in S} \rangle, \Gamma_{(S, \text{TF}, \text{PF})})\) model \( M \) to the partial algebra \( \beta(M) \) where:
   - for any \( s \in S \), \( \beta(M)_s = M_{\text{Def}_s} \);
4.3 CONSERVATIVITY

- for any \( \sigma \in \text{TF}_{\text{ar}\rightarrow s} \), \( \beta(M)_\sigma(m) = M_\sigma(m) \);
- for any \( \sigma \in \text{PF}_{\text{ar}\rightarrow s} \), \( \beta(M)_\sigma \) consists of the restriction of \( M_\sigma \) to \( M_{\text{Def}_{\text{ar}}} \) such that \( \text{dom}(\beta(M)_\sigma) = \{ x \in M_{\text{Def}_{\text{ar}}} \mid M_\sigma(x) \in M_{\text{Def}_s} \} \).

The encoding to \( \text{FOL}^{\text{pres}} \) obtained as instance of the general encoding presented above yields

\[
\Phi'((S, S_0, \text{TF}, \text{TF}_0, \text{PF}, \text{PF}_0), \text{Nom}, \Lambda) = ((|S|, \text{TF} + \text{PF} + \text{Nom}, (D_s)_{s \in S} + [(\text{Def}_s)_{s \in S}] + \text{\Lambda}), D_{\text{TF} + \text{PF}} \cup \Gamma_{(S, \text{TF}, \text{PF})}).
\]

For any \( \mathcal{H}\text{PAR} \) signature \( ((S, S_0, \text{TF}, \text{TF}_0, \text{PF}, \text{PF}_0), \text{Nom}, \Lambda) \),

\[
C((S, S_0, \text{TF}, \text{TF}_0, \text{PF}, \text{PF}_0), \text{Nom}, \Lambda) = \\
\{ (\forall x, z) D_s(x, z) \mid s \in S_0 \} \cup \\
\{ (\forall x, y, Z) \sigma(x, Z) = \sigma(y, Z) \mid \sigma \text{ in } \text{TF}_0 \} \cup \\
\{ (\forall x, y, Z) \text{Def}_s(x, (\sigma(x, Z)) \leftrightarrow \text{Def}_s(y, (\sigma(y, Z))) \mid \sigma \in (\text{PF}_0)_{\text{ar}\rightarrow s, \text{ar} \in S^*, s \in S} \}
\]

Note that the first component in the definition of \( C \) covers both the condition (11) of Theorem 4.1.1 and the condition on the interpretation of the rigid sorts while the condition (12) of Theorem 4.1.1 is entailed by the second component of \( C \). Finally, the adequacy condition for \( \beta'^C \) may be checked easily in the same way as in Example 4.2.5.  

4.3 CONSERVATIVITY

In this section we give a general method to lift the conservativity property from the base comorphism \( (\Phi, \alpha, \beta) : I \rightarrow \text{FOL}^{\text{pres}} \) to comorphism \( (\Phi'^C, \alpha', \beta') : \text{I}'\mathcal{C} \rightarrow \text{FOL}^{\text{pres}} \). We assume the conditions and notation of Theorem 4.1.1 above.

**Proposition 4.3.1** Let us assume, for each \( I \)-signature \( \Sigma \), a mapping

\[
\delta_\Sigma : |\text{Mod}^I(\Sigma)| \rightarrow |\text{Mod}^{\text{FOL}^{\text{pres}}}(\Phi(\Sigma))|
\]

\[\text{Here } M_{\text{Def}_{\text{ar}}} \text{ denotes } M_{\text{Def}_{s_1}} \times \cdots \times M_{\text{Def}_{s_n}} \text{ for } \text{ar} = s_1 \ldots s_n.\]
such that for each $\Sigma$-model $A$, $\beta_{\Sigma}(\delta_{\Sigma}(A)) = A$. For each $\mathcal{H}\mathcal{I}$ signature $\Delta = (\Sigma, \text{Nom}, \Lambda)$ and each model $(M, W) \in |\text{Mod}^{\mathcal{H}\mathcal{I}}(\Delta)|$ such that, for any sort of a variable occurring in a quantification of some sentence in $\Phi(\Sigma)$, and any $w, w' \in |W|$, we have that

$$\delta_{\Sigma}(M_w)_s = \delta_{\Sigma}(M_{w'})_s.$$  \hspace{1cm} (15)

Then, there exists a $\Phi'(\Delta)$-model $\delta'_{\Delta}(M, W)$ such that $\beta'(\delta'_{\Delta}(M, W)) = (M, W)$.

**Proof.** Let $\Phi(\Sigma) = ((S_{\Sigma}, F_{\Sigma}, P_{\Sigma}), \Gamma_{\Sigma})$. We define a $\Phi'(\Delta)$-model $M'$ as follows:

- $M'_{ST} = |W|$,  
- $M'_i = W_i$ for each $i \in \text{Nom}$,
- $M'_\lambda = W_\lambda$ for each modality symbol $\lambda$ in $\Lambda$,
- for each $s \in S_{\Sigma}$, $M'_s = \bigcup_{w \in |W|} \delta_{\Sigma}(M_w)_s$ and $M'_{D_s} = \{ (w, m) \mid m \in \delta_{\Sigma}(M_w)_s \}$,
- for each $\sigma \in (F_{\Sigma})_{ar \rightarrow v}$,
  $$M'_{\sigma}(w, m) = \begin{cases} 
\delta_{\Sigma}(M_w)_{\sigma}(m), & \text{when } m \in \delta_{\Sigma}(M_w)_{ar'}; \\
\text{any } y \in \delta_{\Sigma}(M_w)_{v}, & \text{otherwise}.
\end{cases}$$

Note that the correctness of this definition relies upon our basic hypothesis that the FOL-models have non-empty carriers.

- for each $\pi$ in $P_{\Sigma}$, $M'_{\pi} = \{ (w, m) \mid m \in \delta_{\Sigma}(M_w)_{\pi} \}$.

Now we have to prove that $M'$ satisfies the sentences of $\Phi'(\Delta)$. That $M' \models D_{\Sigma}$ follows immediately from the definitions of $M'_{D_s}$ and of $M'_\sigma$. Also from the hypothesis (15) we have that $M' \models V(\Gamma_{\Sigma})$. For each $w \in |W|$ we let $M'_{|w}$ be defined as in Definition 4.1.7 and Lemma 4.1.1. Note that for each $w \in |W|$, $M'_{|w} = \delta_{\Sigma}(M_w)$.

Since $\delta_{\Sigma}(M_w) \models \Gamma_{\Sigma}$, from (16) and Lemma 4.1.1 it follows that $M^{\text{exp}}_{|w} = \{ [y]_x \mid \gamma \in \Gamma_{\Sigma} \}$, (where $M^{\text{exp}}_{|w}$ denotes the expansion of $M'$ to the signature extended with the constant $x$ such that $M^{\text{exp}}_{|w} = w$). From the latter relation we deduce that $M' \models \Gamma_{\Sigma}$.

That $\beta'(M') = (M, W)$ may be noted immediately with the help of the relation (16). Hence we define $\delta'_{\Delta}(M, W) = M'$. \hfill $\Box$
Corollary 4.3.1. Any comorphism as in Corollary 4.1.1, such that for each constrained model \((M, W) \in |\text{Mod}^C(\Delta)|\),

1. \((M, W)\) satisfies condition (15) of Proposition 4.3.1, and

2. \(\delta'_\Delta(M, W) \models C(\Delta)\)

is conservative.

Example 4.3.1. The encoding of \(\mathcal{H}PL\) hybrid propositional logic of Example 4.2.1 is conservative according to Corollary 4.3.1, as follows:

- \(\delta_\Sigma\) are identities,
- condition (15) of Proposition 4.3.1 is vacuously fulfilled (all \(\Gamma\)'s are empty), and
- obviously, for each S4 \(\mathcal{H}PL\) model \((M, W)\),

\[\delta'_\Delta(M, W) \models (\forall x)\lambda(x, x).\]

Example 4.3.2 (\(\mathcal{H}TRM2FOL\)). The comorphism \(\mathcal{H}TRM2FOL\) is conservative according to Corollary 4.3.1:

- \(\delta_*\) is the identity;
- since all \(\Gamma\)'s are empty, the condition (15) of Proposition 4.3.1 is vacuously fulfilled.

Example 4.3.3. Let us characterize the conservativity of encodings discussed in Example 4.2.3. For that, let us rename to \(\delta^0\) the output \(\delta'\) of Proposition 4.3.1 applied to the encoding \(\mathcal{H}'PL \rightarrow FOL\) (see Example 4.2.3)

\[\delta^0_{(p, Nom^0)} : \text{Mod}^{\mathcal{H}'PL}(p, Nom^0) \rightarrow \text{Mod}^{FOL}([ST^0], \overline{Nom^0}, [P] + \lambda^0).\]

Then by applying again Proposition 4.3.1 with \(\delta^0\) in the role of \(\delta\) we obtain \(\delta^1\) in the role of \(\delta'\):

\[\delta^1_{(p, Nom^0, Nom^1)} : \text{Mod}^{\mathcal{H}'PL}(p, Nom^0, Nom^1) \rightarrow \text{Mod}^{FOL}([ST^0, ST^1], [Nom^0] + Nom^1, \overline{D_{ST^0} + [P] + \lambda^0 + \lambda^1}).\]
Note however that there are models \((M, W)\) such that \(\delta_\Delta'(M, W) \not\models (\forall x, y)D^\circ_{ST_0}(x, y)\), hence for the encoding of \(\mathcal{H}'\mathcal{H}'PL\). Proposition 4.3.1 does not get the conservativity property. But if we consider the constrained cases \(\mathcal{H}'[\mathcal{H}'PL'x]y\) with the sharing of the substates universes and of the base nominals then we get the conservativity property through Proposition 4.3.1 and Corollary 4.3.1.

Example 4.3.4 The encoding of the quantifier free hybridisation of \(FOL\) of Example 4.2.4 is not conservative. Although the condition (15) of Proposition 4.3.1 is vacuously fulfilled (the \(\Gamma\)'s are empty) the example fails in the condition introduced by Corollary 4.3.1 since there are models \((M, W)\) such that \(\delta_\Delta'(M, W) \not\models (\forall x, y)D_s(x, y)\).

However the variant of the example that takes the quantifier free fragment of \(FOL\) as a base institution is conservative because in that case the \(C\)'s are empty (see Example 4.2.4) and thus the condition introduced by Corollary 4.3.1 is vacuously fulfilled.

Example 4.3.5 The encoding of the hybridisation \(\mathcal{H}REL'\) of \(REL\) of Example 4.2.5 is conservative according to Corollary 4.3.1 because:

1. \(\delta_\Sigma\) are identities,
2. condition (15) of Proposition 4.3.1 is vacuously fulfilled (all \(\Gamma\)'s are empty), and
3. for each \(\mathcal{H}REL'\) model \((M, W)\), \(\delta_\Delta'(M, W)\) satisfies \(C(\Delta)\) since for all \(w, w' \in |W|\),

\[
\delta_\Sigma(M_w)_x = (M_w)_x = (M_{w'})_x = \delta_\Sigma(M_{w'})_x
\]

for each sort symbol or constant \(x\).

The conservativity of comorphism \(\mathcal{H}REL'_02FOL^{pres}\) can be established considering points 1 and 2 as above, and observing that for each \(\mathcal{H}REL'_0\) model \((M, W)\), \(\delta_\Delta'(M, W)\) satisfies \(C(\Delta)\) as follows: by the constraint definition we have that, for all \(w, w' \in |W|\), and for any \(\lambda \in \Lambda\) such that \((w, w') \in W_\lambda\), \(|M_w| \subseteq |M_{w'}|\). Moreover, by the definition of \(\delta'\), that for any \(w \in W\), \(|M'_D_\lambda|_w\) = \(|M_w|\). Hence

\[
(|M'_D_\lambda|_w) = |M_w| \subseteq |M_{w'}| = |(M'_D_\lambda|_w')|.
\]
Therefore, for any \( m, M_D'(w, m) \subseteq M_D'(w', m) \). Hence,

\[
M' \models (\forall x, y, z)[\lambda(x, y) \Rightarrow (D_s(x, z) \Rightarrow D_s(y, z))]
\]

Example 4.3.6 The encoding of the hybridisation of FOLR of Example 4.2.7 is conservative according to Corollary 4.3.1 by arguments similar to those expressed in Example 4.3.5 above.

Example 4.3.7 Consider now the encoding of the hybridisation \( \mathcal{H}PAR' \) in Example 4.2.8. This is conservative according to Corollary 4.3.1. Actually, for each \( \Sigma = (S_0, S, TF_0, TF, PF_0, PF) \)-model \( M, \delta_\Sigma(M) \) is defined as follows:

- for any \( s \in S \), \( \delta_\Sigma(M)_s = M_s \cup \{\bot\} \) where \( \bot \) is a new element; and \( \delta_\Sigma(M)_{\text{Def} s} = M_s \)
- for any \( \sigma \in (\text{TF} + \text{PF})_{\text{ar} \rightarrow s} \),

\[
\delta_\Sigma(M)_{\sigma}(m) = \begin{cases} 
M_{\sigma}(m), & \text{if } m \in \delta_\Sigma(M)_{\text{Def} \sigma} \text{ and } M_{\sigma}(m) \text{ is defined} \\
\bot, & \text{otherwise.}
\end{cases}
\]

It is easy to check that \( \delta_\Sigma(M) \models [S, \text{TF}, \text{PF}] \) and that \( \beta_\Sigma(\delta_\Sigma(M)) = M \).

The condition (15) of Proposition 4.3.1 is satisfied as follows. For each \( (M, W) \in |\text{Mod}^{\mathcal{H}PAR'}(\Delta)| \), each rigid sort \( s \) (since all quantifications with first order variables are over rigid sorts) and any \( w, w' \in |W| \), we have

\[
\delta_\Sigma(M_w)_s = (M_w)_s \cup \{\bot\} \quad \text{(definition of \( \delta \))}
= (M_w')_s \cup \{\bot\} \quad ((M_w)_s = (M_w')_s \text{ because } s \text{ is rigid })
= \delta_\Sigma(M_w')_s \quad \text{(definition of \( \delta \)).}
\]

The justification that for each \( \mathcal{H}PAR' \) model \( (M, W) \), \( \delta_\Delta'(M, W) \models C(\Delta) \) goes as follows:

1. for each \( s \in S_0 \), \( \delta_\Delta'(M, W) \models (\forall x, z)D_s(x, z) \) means that for each \( w \in |W| \) and each \( m \in \delta_\Delta'(M, W)_s \), \((w, m) \in \delta_\Delta'(M, W)_{D_s} \) which, according to the definition of \( \delta_\Delta'(M, W)_{D_s} \) from the proof
of Proposition 4.3.1 means $m \in \delta_\Sigma(M_w)_s$. But $\delta'_\Lambda(M, W)_s = \delta_\Sigma(M_w)_s$ because $s$ is rigid (which according to an argument above implies that for all $w, w' \in |W|$, $\delta_\Sigma(M_w)_s = \delta_\Sigma(M_{w'})_s$).

2. for each $\sigma$ in $\text{TF}_0$, $\delta'_\Lambda(M, W) \models (\forall x, y, Z)\sigma(x, Z) = \sigma(y, Z)$ holds because of the following facts:

- for each rigid sort $s$ and each $w \in |W|$, $\delta'_\Lambda(M, W)_s = \delta_\Sigma(M_w)_s$;
- since $\sigma$ is rigid and total, for each $w, w' \in |W|$, $(M_w)_\sigma = (M_{w'})_\sigma$;
- for each $w \in |W|$, $\delta'_\Lambda(M, W)_\sigma(w, m) = \delta_\Sigma(M_w)_\sigma(m)$ because $\delta'_\Lambda(M, W)_s = \delta_\Sigma(M_w)_s$.

3. For all $\text{ar} \in S^*$, $s \in S$ and $\sigma \in (\text{TF}_0)_{\text{ar}\rightarrow s}$,

$$\delta'_\Lambda(M, W) \models (\forall x, y, Z)\text{Def}_s(x, (\sigma(x, Z))) \iff \text{Def}_s(y, (\sigma(y, Z)))$$

which means that for all $w, w' \in |W|$, $\delta'_\Lambda(M, W)_\sigma(w, m) \in \delta_\Sigma(M_w)_{\text{Def}_s} = (M_w)_s$ if and only if $\delta'_\Lambda(M, W)_\sigma(w', m) \in \delta_\Sigma(M_{w'})_{\text{Def}_s} = (M_{w'})_s$. But $(M_w)_s = (M_{w'})_s$ and

$$\delta'_\Lambda(M, W)_\sigma(w, m) = \delta_\Sigma(M_w)_\sigma(m)$$

and $\delta'_\Lambda(M, W)_\sigma(w', m) = \delta_\Sigma(M_{w'})_\sigma(m)$. Thus the property is equivalent to the fact that $(M_w)_\sigma(m)$ is defined if and only if $(M_{w'})_\sigma(m)$ is defined, which holds by the rigidity of $\sigma$, i.e. because $(M_w)_\sigma$ and $(M_{w'})_\sigma$ have the same domain.

$\circ$
We introduce in this chapter the definition and the characterisation of suitable relations to compare and relate models of hybridised logics. There are two important relations to consider: behavioural equivalence, relating models with indistinguishable behaviours and refinement, relating abstract with more concrete models preserving all of its abstract properties. The choice of bisimilarity to base behavioural equivalence of models and similarity to base refinement seems quite standard as a fine grained approach to observation based methods for system’s comparison. The notion of bisimulation and the associated conductive proof method were originated in concurrency theory due to the seminal work of David Park [Par81] and Robin Milner [Mil89] in the quest for an appropriate definition of observational equivalence for communicating processes as understood in the CCS calculus [Mil80, Mil89].

The notion is now pervasive in Computer Science as the standard behavioural equivalence for a wide range of transition systems. Specific formalizations range from the case of classical automata and labeled transition systems [BBS88, DMV90] to the theory of the most sophisticated processes models as probabilistic timed automata or fuzzy transition systems [ST10, CCK11]. A very abstract unifying notion of bisimulation was proposed and characterized in universal coalgebra [AM89, Rut00]. But the concept also arose independently of modal logic as a refinement of notions of homomorphism between algebraic models — see [San09] for an extensive historical account, including its role in the development of (non well-founded) set theory.

This Chapter is organized as follows: Subsection 5.1 introduces a general notion of bisimulation for hybridised logics and characterizes the preservation of logic satisfaction under it. Then Subsection 5.2 follows a similar path, but focusing on refinement witnessed by a simulation relation.
5.1 BISIMULATION FOR HYBRIDISED LOGICS

Intuitively a bisimulation relates states which exhibit the “same” (observable) information and preserves this property along transitions. Thus, to define a general notion of bisimulation over Kripke structures whose states are models of whatever base logic was chosen for specifications, we have to make precise what the “same” information actually means. In a previous work [MMB12] on infinitary equational hybrid logic we required from bisimilar states that the generated varieties of each local algebra be identical. In the more general setting of hybridised institution \( \mathcal{H} \), the corresponding notion is elementary equivalence for first-order logic (e.g. [Hod97]). We therefore explore this possibility, requiring (local) \( \mathcal{I} \)-models of bisimilar states to be elementarily equivalent. Formally,

**Definition 5.1.1** Let \( M, M' \in \text{Mod}^\mathcal{I}(\Sigma) \) and \( \text{Sen}' \) be a subfunctor of \( \text{Sen}^\mathcal{I}(\Sigma) \). Models \( M \) and \( M' \) are elementarily equivalent with respect to sentences in \( \text{Sen}'(\Sigma) \), in symbols \( M \equiv_{\text{Sen}'} M' \), if for any \( \rho \in \text{Sen}'(\Sigma) \)

\[
M \models^\mathcal{I} \rho \iff M' \models^\mathcal{I} \rho.
\]

If only the implication \( \Rightarrow \) of (17) holds in the right hand side of the above equivalence we write \( M \gg_{\text{Sen}'} M' \).

Note the role of \( \phi \) above: as a signature morphism it captures the possible change of notation from a specification to another.

Under the institution theory motto — truth is invariant under change of notation — we write \( M \equiv_{\phi, \text{Sen}'} M' \) whenever \( M \equiv_{\text{Sen}'} \text{Mod}^\mathcal{I}(\phi)(M') \) for a given \( \phi \in \text{Sign}^\mathcal{I}(\Sigma, \Sigma') \), \( M \in \text{Mod}^\mathcal{I}(\Sigma) \) and \( M' \in \text{Mod}^\mathcal{I}(\Sigma') \). Models \( M \) and \( M' \) are said to be \( (\phi, \text{Sen}') \)-elementarily equivalent.

Resorting to the satisfaction condition in \( \mathcal{I} \), the following characterisation of \( (\phi, \text{Sen}') \)-elementary equivalence pops out:

**Corollary 5.1.1** On the conditions of Definition 5.1.1, \( M \equiv_{\phi, \text{Sen}'} M' \) iff for any \( \rho \in \text{Sen}'(\Sigma) \), \( M \models_{\Sigma}^\mathcal{I} \rho \iff M' \models_{\Sigma}^\mathcal{I} \text{Sen}'(\phi)(\rho) \).

However, its pertinence becomes clear in refinement situations, as discussed in the next section, where it may accommodate many forms...
of interface enrichment or adaptation (e.g. through the introduction of auxiliary operations).

**Definition 5.1.2** Let $\cal I$ be the quantifier-free hybridisation of the institution $\cal J$ and $\varphi \in \text{Sign}^{\cal I}(\Delta, \Delta')$ a signature morphism. Let $\text{Sen}'$ be a subfunctor of $\text{Sen}$ $^J$. A $(\varphi, \text{Sen}')$-bisimulation between models $(M, W) \in \text{Mod}^{\cal I}(\Delta)$ and $(M', W') \in \text{Mod}^{\cal I}(\Delta')$ is a non-empty relation $B_\varphi \subseteq |W| \times |W'|$ such that

(i) For any $\lambda \in \Lambda_n$, if $(w, w_1, \ldots, w_n) \in W_\lambda$ and $w B_\varphi w'$, then for each $k \in \{1, \ldots, n\}$ there is a $w'_k \in |W'|$ such that $w_k B_\varphi w'_k$ and $(w', w'_1, \ldots, w'_n) \in W'_{\varphi\text{MS}(\lambda)}$.

(ii) For any $\lambda \in \Lambda_n$ if $(w', w'_1, \ldots, w'_n) \in W'_{\varphi\text{MS}(\lambda)}$ and $w B_\varphi w'$, then for each $k \in \{1, \ldots, n\}$ there is a $w_k \in |W|$, such that $w_k B_\varphi w'_k$ and $(w, w_1, \ldots, w_n) \in W_\lambda$.

(iii) for any $w B_\varphi w'$, and for any $i \in \text{Nom}$,

$$W_i = w \text{ iff } W'_{\varphi\text{Nom}(i)} = w'.$$

(iv) for any $i \in \text{Nom}$, $W_i B_\varphi W'_{\varphi\text{Nom}(i)}$.

(v) for any $w B_\varphi w'$, $M_w \equiv^{\text{Sign}}_{\varphi\text{Sen}} M'_w$.

We say that $(M, W)$ and $(M', W')$ are $(\varphi, \text{Sen}')$-bisimilar, and write $(M, W) \equiv^\varphi (M', W')$, if there is a $(\varphi, \text{Sen}')$-bisimulation $B_\varphi$ between them. Whenever $\varphi$ is the identity we simply talk of a bisimulation $B$ and bisimilarity $\equiv$.

Next theorem establishes that, for quantifier-free hybridisations, the (local)-hybrid satisfaction $|=^{\cal I}$ is invariant under $(\varphi, \text{Sen})$-bisimulations. The Theorem generalises the well known result bisimulation invariance result in the hybrid propositional logic (e.g. [BVB07]).

**Theorem 5.1.1** Let $\cal I$ be a quantified-free hybridisation of the institution $\cal J$ and $\varphi \in \text{Sign}^{\cal I}(\Delta, \Delta')$ a signature morphism. Let $B_\varphi \subseteq |W| \times |W'|$ be a $(\varphi, \text{Sen})$-bisimulation. Then, for any $w B_\varphi w'$ and for any $\rho \in \text{Sen}^{\cal I}(\Delta)$,

$$(M, W) \models^w \rho \text{ iff } (M', W') \models^{w'} \text{Sen}^{\cal I}(\varphi)(\rho).$$
Proof. The proof is by induction on the structure of the sentences.

1. \( \rho = i \) for some \( i \in \text{Nom} \):

\[
(M, W) \models^w i \\
\iff \{ \text{defn. of } \models^w \}
\]

\( W_i = w \)

\[
\iff \{ \text{ (iii) of Definition 5.1.2 } \}
\]

\[
W'_{\varphi(i)} = w'
\]

\[
\iff \{ \text{defn of } \models^w \}
\]

\[
(M', W') \models^{w'} \varphi_{\text{Nom}}(i)
\]

\[
\iff \{ \text{defn of } \text{Sen}^{3CJ}(\varphi) \}
\]

\[
(M', W') \models^{w'} \text{Sen}^{3CJ}(\varphi)(i)
\]

2. \( \rho \in \text{Sen}_3(\Sigma) \):

\[
(M, W) \models^w \rho
\]

\[
\iff \{ \text{defn. of } \models^w \}
\]

\[
M_w \models^3 \rho
\]

\[
\iff \{ \text{by hypothesis } M_w \equiv \varphi_{\text{Sign}}, M'_w, + \text{Corollary 5.1.1} \}
\]

\[
M'_{w'} \models^3 (\varphi_{\text{Sign}})(\rho)
\]

\[
\iff \{ \text{defn of } \models^w \}
\]

\[
(M', W') \models^{w'} \text{Sen}^3(\varphi_{\text{Sign}})(\rho)
\]

\[
\iff \{ \text{defn of } \text{Sen}^{3CJ}(\varphi) \}
\]

\[
(M', W') \models^{w'} \text{Sen}^{3CJ}(\varphi)(\rho)
\]

3. \( \rho = \xi \lor \xi' \) for some \( \xi, \xi' \in \text{Sen}^{3CJ}(\Delta) \):

\[
(M, W) \models^w \xi \lor \xi'
\]

\[
\iff \{ \text{defn. of } \models^w \}
\]

\[
(M, W) \models^w \xi \text{ or } (M, W) \models^w \xi'
\]

\[
\iff \{ \text{I.H.} \}
\]

\[
(M', W') \models^{w'} \text{Sen}^{3CJ}(\varphi)(\xi) \text{ or }
\]
5.1 Simulation for Hybridised Logics

\[(M', W') \models^{w'} \text{Sen}^{3CJ}(\varphi)(\xi') \]
\[\iff \{ \text{defn. of } \models^{w} \} \]
\[(M', W') \models^{w'} \text{Sen}^{3CJ}(\varphi)(\xi \lor \xi') \]

The proofs for cases \(\rho = \xi \land \xi', \rho = \xi \Rightarrow \xi', \rho = \neg \xi, \text{ etc.} \) are analogous.

4. \(\rho = [\lambda](\xi_1, \ldots, \xi_n) \) for some \(\xi_1, \ldots, \xi_n \in \text{Sen}^{3CJ}(\Delta), \lambda \in \Lambda_{n+1}: \)

\[(M, W) \models^{w} [\lambda](\xi_1, \ldots, \xi_n) \]
\[\iff \{ \text{defn. of } \models^{w} \} \]
for any \((w, w_1, \ldots, w_n) \in W_{\lambda} \) there is some \(k \in \{1, \ldots, n\} \) such that \((M, W) \models^{w_k} \xi_k \)
\[\iff \{ \ast \} \]
for any \((w', w'_1, \ldots, w'_n) \in W'_{\phi_{\text{MS}}(\lambda)} \) there is some \(p \in \{1, \ldots, n\} \) such that \((M', W') \models^{w'_p} \text{Sen}^{3CJ}(\varphi)(\xi_p) \)
\[\iff \{ \text{defn. of } \text{Sen}^{3CJ}(\varphi) \} \]
\[(M', W') \models^{w'} [\phi_{\text{MS}}(\lambda)] \text{Sen}^{3CJ}(\varphi)(\xi_1, \ldots, \text{Sen}^{3CJ}(\varphi)(\xi_n)) \]
\[\iff \{ \text{defn. of } \text{Sen}^{3CJ}(\varphi) \} \]
\[(M', W') \models^{w'} \text{Sen}^{3CJ}(\varphi)([\lambda](\xi_1, \ldots, \xi_n)) \]

For the step marked with \(\ast\) we proceed as follows. Supposing \((w', w'_1, \ldots, w'_n) \in W'_{\phi_{\text{MS}}(\lambda)} \) with \(wB_{\varphi} w'\), we have by clause (ii) of Definition 5.1.2 that there are \(w_k \), with \(k \in \{1, \ldots, n\} \), such that \((w, w_1, \ldots, w_n) \in W_{\lambda} \). By hypothesis, \((M, W) \models^{w_p} \xi_p \) for some \(p \in \{1, \ldots, n\} \). Moreover, by I.H. \((M', W') \models^{w'_p} \text{Sen}^{3CJ}(\varphi)(\xi_p) \). Clause (i) of Definition 5.1.2 entails the converse implication. The proof for sentences of form \(\rho = \langle \lambda \rangle(\xi_1, \ldots, \xi_n) \) is analogous.

5. \(\rho = @_i \xi \) for some \(\xi \in \text{Sen}^{3CJ}(\Delta) \) and \(i \in \text{Nom}: \)
\[(M, W) \models^{w} @i, \xi \]
\[\Leftrightarrow \{ \text{defn. of } \models^{w} \} \]
\[(M, W) \models^{W, i, \xi} \]
\[\Leftrightarrow \{ \text{I.H. and clause (iv) of Definition 5.1.2} \}
\[(M', W') \models^{W'_{\phi_{\text{Nom}}^{(i)}}} \text{Sen}^{3\mathcal{C}}(\varphi)(\xi) \]
\[\Leftrightarrow \{ \text{defn. of } \models^{w} \} \]
\[(M', W') \models^{w @\varphi_{\text{Nom}}^{(i)}} \text{Sen}^{3\mathcal{C}}(\varphi)(\xi) \]
\[\Leftrightarrow \{ \text{defn. of } \text{Sen}^{3\mathcal{C}}(\varphi) \}
\[(M', W') \models^{w} \text{Sen}^{3\mathcal{C}}(\varphi)(@i, \xi) \]

\[\square\]

As a direct consequence of the previous theorem we get the following characterisation of the preservation of (global) satisfaction, $\models^{3\mathcal{C}}$, under $\varphi$-bisimilarity:

**Corollary 5.1.2** On the conditions of Theorem 5.1.1, let $(M, W) \models_{\varphi} (M', W')$ witnessed by a total and surjective bisimulation. Then,

\[(M, W) \models^{3\mathcal{C}} \rho \text{ iff } (M', W') \models^{3\mathcal{C}} \text{Sen}^{3\mathcal{C}}(\varphi)(\rho). \quad (19)\]

**Example 5.1.1 (Bisimulation in \mathcal{H}PL)** Let us instantiate Definition 5.1.2 for the \mathcal{H}PL case (cf. Example 3.2.1), considering $\varphi = \text{id}$ and $\text{Sen}' = \text{Sen}^{\mathcal{H}}$. Let consider the models

\[(M, W), (M', W') \in \text{Mod}^{3\mathcal{H}PL}(P, \text{Nom}, \{\lambda\}).\]

B is a bisimulation between $(M, W)$ and $(M', W')$ if

(i) for any $(w, w_1) \in W_\lambda$ with $wBw'$, there is a $w'_1 \in |W'|$ such that $w_1Bw'_1$ and $(w_1, w'_1) \in W'_\lambda$;

(ii) for any $(w', w'_1) \in W'_\lambda$ with $wBw'$, there is a $w_1 \in |W|$ such that $w_1Bw'_1$ and $(w_1, w'_1) \in W'_\lambda$;

(iii) for any $i \in \text{Nom}$, $wBw'$, $w = W_i$ iff $w' = W'_i$;
(iv) for any \( i \in \text{Nom} \), \( W_i B W'_i \);

(v) \( M_w \equiv M'_w \), i.e., bisimilar states satisfy the same sentences.

Note that condition (v) is equivalent to say that bisimilar states have assigned the same set of propositions (for any \( p \in P \), \( M_w(p) = \top \) iff \( M'_w(p) = \top \)). As expected, this definition corresponds exactly to standard bisimulation for propositional hybrid logic (see, e.g. [tC05, Defn 4.1.1]).

The definition of bisimulation computed in the previous example, can also capture the case of propositional modal logic: just consider pure modal signatures (i.e., with an empty set of nominals), as condition (iii) and (iv) is trivially satisfied. Moreover, instantiating Theorem 5.1.1 we get the classical result on the preservation of modal satisfaction by bisimulation (cf. [BVB07]).

**Example 5.1.2 (Bisimulation for \( \mathcal{HEQ} \))** Consider now the instantiation of 5.1.2 for \( \mathcal{HEQ} \). All one has to do is to replace condition (iv) in Definition 5.1.2 by its instantiation for algebras: two algebras are equationally equivalent if the respective generated varieties coincides [Grä79].

5.2 **Refinements for Generic Hybridised Logics**

A standard notion of refinement for Kripke models, and in general for transition systems, is based on *simulations*. On the one hand, it entails preservation (but not reflection) of transitions, from the abstract to the concrete system. On the other hand, at each local state, preservation of the original properties along local refinement. In the context of the approach, this amounts to our following characterization:

**Definition 5.2.1** Let \( \mathcal{HJ} \) be the quantified-free hybridisation of an institution \( J \), \( \varphi \in \text{Sign}^{|\Delta, \Delta'|} \) a signature morphism and \( \text{Sen'} \) a sub-functor of \( \text{Sen} \). A \( (\varphi, \text{Sen'}) \)-refinement of \( (M, W) \in \text{Mod}^{|\Delta|}(\Delta) \) by \( (M', W') \in \text{Mod}^{|\Delta'|}(\Delta') \) consists of a non-empty relation \( R_{\varphi \text{Sen'}} \subseteq |W| \times |W'| \) such that, for any \( w R_{\varphi \text{Sen'}} w' \),

\[ (f.i) \text{ for any } i \in \text{Nom}, \text{ if } W_i = w \text{ then } W'_i_{\varphi \text{Nom}(i)} = w'. \]
(f.ii) \( M_w \gg^\text{Sen'}_\varphi M'_{w'} \).

(f.iii) for any \( i \in \text{Nom} \), \( W_i \mathcal{R}_{\varphi}^\text{Sen'} W'_{\varphi_{\text{Nom}(i)}} \).

(f.iv) For any \( \lambda \in \Lambda_n \), if \((w, w_1, \ldots, w_n) \in W_\lambda \) then for each \( k \in \{1, \ldots, n\} \) there is an \( w'_k \in |W'| \) such that \( w_k \mathcal{R}_\varphi w'_k \) and \((w', w'_1, \ldots, w'_n) \in W'_{\varphi_{\text{MS}}(\lambda)} \).

Is (hybrid) satisfaction preserved by refinement? On a first attempt, it is natural to accept a positive answer which, although intuitive, is wrong. Actually, not all hybrid sentences can be preserved along a refinement chain. Note on the proof of Theorem 5.1.1, that the preservation of hybrid satisfaction of sentences \( [\lambda](\xi_1, \ldots, \xi_n) \) is entailed by condition (ii) of Definition 5.1.1, but the latter is stated on the opposite direction to refinement. As a simple counter-example, define a \( \mathcal{R}_\varphi \)-refinement from a \( \Delta \)-hybrid model \((M, W)\) with \( |W| = \{w\} \) and \( W_\lambda = \emptyset \) for \( \lambda \in \Lambda_n \) to any other \( \Delta' \)-hybrid model \((M', W')\) such that \( \text{Mod}^{\varphi_{\text{Sign}}}(\lambda)(M'_{w'}) = M_w \) for some \( w' \in |W'| \). Sentence \( [\lambda](\xi_1, \ldots, \xi_n) \), which trivially holds in the state \( w \) of \((M, W)\), may fail to be satisfied in the \( \mathcal{R}_\varphi \)-related state \( w' \) of \((M', W')\). Sentences like \( \neg \xi \) provide another counter-example. The reason is that, by hypothesis, preservation is only assumed on the refinement direction and, of course, non satisfaction in one direction, does not imply non satisfaction in the other. Therefore, differently from the bisimulation case, the preservation of the satisfaction under refinement does not hold for all the hybrid sentences. Actually, the ‘boxed’ and negated sentences are exactly the cases in which this may fail.

Finally, a note regarding parameter \( \text{Sen'} \) in condition (f.ii). First of all note that the “unrestricted” implication of clause (f.ii) in Definition 5.2.1 is very strong: it often implies the converse implication as well. For instance, in \( \mathcal{H}(PL) \), the condition holds if and only if \( M_w = \text{Mod}(\varphi)(M'_{w'}) \). In particular, an \( \text{id} \)-refinement implies the equality of realisations of related states (since, the implication “\( M_w \Vdash_{PL} \neg p \) then \( M'_{w'} \Vdash_{PL} \neg p \)” is equivalent to the implication “\( M'_{w'} \Vdash_{PL} p \) then \( M_w \Vdash_{PL} p \)”.

Hence, \( M_w = M'_{w'} \). It seems reasonable to somehow weaken this condition to yield a strict inclusion. One way to do this is to restrict the focus to a subset of the sentences in the base institution. In the example mentioned above this will correspond to
exclude PL negations, which amounts to take as $\text{Sen}'(\text{Prop})$ the set of propositional sentences without negations.

Given an institution $\mathcal{J} = (\text{Sign}^\vartheta, \text{Sen}^\vartheta, \text{Mod}^\vartheta, (\models_{\Sigma})_{\Sigma \in \text{Sign}^\vartheta})$ and a sentences subfunctor $\text{Sen}' \subseteq \text{Sen}^\vartheta$, we denote by $\mathcal{H}\mathcal{C}'$ the hybridisation of the institution $\mathcal{J}' = (\text{Sign}^\vartheta, \text{Sen}', \text{Mod}^\vartheta, (\models_{\Sigma})_{\Sigma \in \text{Sign}^\vartheta})$.

**Definition 5.2.2 (Sen'-Positive Existential sentences)** The Sen'-positive existential sentences of a signature $\Delta \in \lbrack \text{Sign}^{\mathcal{H}\mathcal{C}\vartheta} \rbrack$ are given by a subfunctor $\text{Sen}^{\mathcal{H}\mathcal{C}\vartheta} \subseteq \text{Sen}^{\mathcal{H}\mathcal{C}\vartheta}'$ defined recursively for each signature $\Delta$ as $\text{Sen}^{\mathcal{H}\mathcal{C}\vartheta}'(\Delta)$ but excluding both negations and box modalities. For each signature morphism $\varphi : \Delta \rightarrow \Delta'$, $\text{Sen}^{\mathcal{H}\mathcal{C}\vartheta}'(\varphi)$ is the restriction of $\text{Sen}^{\mathcal{H}\mathcal{C}\vartheta}'(\varphi)$ to $\text{Sen}^{\mathcal{H}\mathcal{C}\vartheta}(\Delta)$.

**Theorem 5.2.1** Let $\mathcal{H}\mathcal{C}$ be the quantifier free hybridisation of an institution $\mathcal{J}$, $\text{Sen}'$ a subfunctor of $\text{Sen}^\vartheta$, $\varphi \in \text{Sign}^{\mathcal{H}\mathcal{C}}(\Delta, \Delta')$ a signature morphism, $(M, W) \in \text{Mod}^{\mathcal{H}\mathcal{C}}(\Delta)$ and $(M', W') \in \text{Mod}^{\mathcal{H}\mathcal{C}}(\Delta')$. Suppose that $(M', W')$ is a refinement of $(M, W)$ witnessed by the relation $R_{\text{Sen}^\vartheta}'$. Then, for any $wR_{\text{Sen}^\vartheta}' w'$ and $\rho \in \text{Sen}^{\mathcal{H}\mathcal{C}}(\Delta)$,

$$(M, W) \models_{w} \rho \text{ implies that } (M', W') \models_{w'} \text{Sen}^{\mathcal{H}\mathcal{C}}(\varphi)(\rho).$$

**Proof.** The proof is by induction on the structure of the existential positive sentences and comes directly from the proof of Theorem 5.1.1, taking the right to left implication. Preservation of base sentences follows exactly the same proof since the induction hypothesis is precisely about the Sen’ sentences. What remains to be proved is the case $\rho = \langle \lambda \rangle(\xi_{1}, \ldots, \xi_{n})$. We have,

$$(M, W) \models_{w} \langle \lambda \rangle(\xi_{1}, \ldots, \xi_{n}) \iff \begin{cases} \text{defn. of } \models_{w} \\ \text{there exists } (w, w_{1}, \ldots, w_{n}) \in W_{\lambda} \\
\text{such that } (M, W) \models_{w_{k}} \xi_{k} \text{ for any } k \in \{1, \ldots, n\} \\
\Rightarrow \begin{cases} \text{By (f.iii), we have } w_{k}R_{\varphi}w' \text{ for any } k \in \{1, \ldots, n\} + \text{I.H.} \\
\text{there exists } (w', w'_{1}, \ldots, w'_{n}) \in W'_{\varphi_{\text{Mod}}(\lambda)} \\
\text{such that } (M', W') \models_{w'} \xi_{k} \text{ for any } k \in \{1, \ldots, n\} \\
\iff \begin{cases} \text{defn. of } \models_{w'} \\
(M', W') \models_{w'} \langle \varphi_{\text{Mod}}(\lambda) \rangle(\text{Sen}^{\mathcal{H}\mathcal{C}}(\varphi)(\xi_{1}), \ldots, \text{Sen}^{\mathcal{H}\mathcal{C}}(\varphi)(\xi_{n})) \end{cases} \end{cases} \end{cases}$$
Corollary 5.2.1 In the conditions of Theorem 5.2.1, for any \( \rho \in \text{Sen}^{\mathcal{CJ}}(\Delta) \), if \( R_\varphi \) is surjective, then

\[
(M, W) \models \rho \text{ implies that } (M', W') \models \text{Sen}^{\mathcal{CJ}}(\varphi)(\rho).
\]

The following examples illustrate refinement situations in this setting.

Example 5.2.1 (Refinement in \( \mathcal{H} \text{MVL} \)) Figure 2 illustrates an example of a Sen’-refinement in \( \mathcal{H} \text{MVL}_{L_4} \), for \( L_4 \) represented in Figure 2. Let consider \( \Delta = (\emptyset, \emptyset, \text{LProp}) \) such that \( \text{LProp}_{\text{Ar}} = \emptyset \) for any \( \text{Ar} \neq \emptyset \). Consider \( \text{Sen}' \subseteq \text{Sen}' \) restricting the base sentences to propositions, i.e., \( \text{Sen}'(\Delta) = \{((p, l) | p \in \text{LProp} \text{ and } l \in L_4) \} \). Conditions (f.ii)

\[
\text{and (f.iii) of Definition 5.2.1 are obviously satisfied. In what concerns the verification of condition (f.iii) for which } (p, l) \in \text{Sen}'(\text{LProp}),
\]

\[
M_w \models_{\text{MVL}_{L_4}} (p, l) \Rightarrow M'_w \models_{\text{LProp}} (p, l), \text{ it is sufficient to see that,}
\]

\[
(M_w \models p) \leq (M'_w \models p), p \in \text{LProp}.
\]

Example 5.2.2 (Refinement in \( \mathcal{H} \text{EQ} \)) Consider a store system abstractly modelled as the initial algebra \( A \) of the \( ((S, F), \Gamma) \) where

\[
S = \{\text{mem}, \text{elem}\}, F_{\text{mem,elem} \rightarrow \text{mem}} = \{\text{write}\}, F_{\text{mem} \rightarrow \text{mem}} = \{\text{del}\} \text{ and } F_{\text{Ar} \rightarrow s} = \emptyset \text{ otherwise and } \Gamma = \{\text{del}(\text{write}(m,e)) = m\}. \]

Suppose one intends to refine this structure into a read function configurable in two different modes: in one of them it reads the first element
in the store, in the other the last. Reconfiguration between the two execution modes is enforced by an external event shift. Note that the abstract model can be seen as the \((S,F,\emptyset,\{\text{shift}\})\)-hybrid model \(M = (M,W)\), taking \(|W| = \{\ast\}\), \(W_{\text{shift}} = \emptyset\) and \(M_\ast = A\). Then, we take the inclusion morphism \(\varphi_{\text{Sign}} : (S,F) \hookrightarrow (S,F')\) where \(F'\) extends \(F\) with \(F_{\text{mem} \rightarrow \text{elem}} = \text{read}\) and \(F_{\text{mem}} = \{\text{empty}\}\). For the envisaged refinement let us consider the model \(M' = (M',W')\) where \(W' = \{s_1,s_2\}\) and \(W'_{\text{shift}} = \{(s_1,s_2),(s_2,s_1)\}\) and where \(M_{s_1}\) and \(M_{s_2}\) are the initial algebras of the equations presented in Figure 3. It is not difficult to see that \(R = \{(\ast,s_1),(\ast,s_2)\}\) is a \(\varphi\)-refinement relation: conditions \((f.i)\) and \((f.iii)\) of Definition 5.2.1 are trivially fulfilled and, condition \((f.ii)\) is a direct consequence of initiallity.
Part II

TECHNIQUES
6.1 Motivation

As a complex artefact software has to meet requirements formulated and verified at different levels of abstraction. As mentioned in the introduction of this thesis, a basic distinction is drawn between behavioural (dynamic) and data (static) aspects. Another one distinguishes between the local and global levels. This is particularly relevant in the project of reconfigurable systems where the former encompasses the services and operations provided in each configuration, and the latter concerns the overall system dynamics leading from one configuration to another.

Actually, what a software component may offer at each stage depends on its own evolution and history. It typically acts as an evolving structure which may change from one mode of operation to another, entailing corresponding updates in what counts, at each mode or stage, as a valid description of its behaviour. For example, a component in a sensor network may be unable to restart a particular equipment if in an alarm stage of operation, but not in a normal one. On the other hand, the way it computes the result of sensoring a number of hardware control devices may change from one mode to another (changing, for example, the palette of weights used to compute a weighted sum of measurements).

This entails the need to consider in the project of reconfigurable software two different levels of specification. The global view is that of a transition system whose transitions are triggered by events that enforce a move from a configuration to another. Modal logics provide the standard language to express properties and reason about this sort of structures. On the other hand, the specification of each configuration may resort to whatever formalism fits better the problem domain. Equation logic, as well as its partial or hidden, variants are typical choices in classical algebraic specification [EM85, Wir90, ST12]. But less com-
mon alternatives are equally useful, for example, multivalued logic to express a certain degree of vagueness in the problem, or modal logic itself when each configuration is described by a local transition system, for instance by a process algebra expression. Reference [Bjø06a], the second volume of Dines Bjørner monumental treatise on Software Engineering, offers a detailed discussion on the choice of specification logics for different problem domains.

As stated in the Introduction, this thesis aims at building foundations for a specification method in which these two levels are not only made explicit and juxtaposed, but formally interrelated. The key to achieve such a goal is the hybridisation process characterised in Chapters 3 to 5. Actually, through the hybridisation of whatever is taken as a local specification formalism, the relation between the local and global levels becomes internalised in the logic itself. This leads to a specification method for reconfigurable systems whose development is documented in reference [MFMB11]. The second part of this thesis, encompassing Chapters 6 and 7, is devoted to its introduction and illustration. The overall approach is presented in this chapter. Tool support and additional techniques are discussed in Chapter 7.

Note that at each specification stage one needs to consider the mathematical structures suitable to model systems’ components; the languages in which such models can be described and, finally, the relationship between the (semantic) structures and the (syntactic) formulation of requirements as sentences in the specification language. In a quite canonical way, the three aspects can be put under the common umbrella of an institution [GB92] which, as an abstract representation of a logical system, encompasses syntax, semantics and satisfaction, and provides ways to relate, compare and combine specification logics. As argued below, it is remarkable how the theory of institutions not only provides a standard, mathematically solid basis for the approach discussed here, but also paves the way to suitable tool support. A discussion on the use of the HETS platform [MMCL13] for validating our specifications is included in Chapter 7.
6.2 THE APPROACH

As suggested by the title of this chapter, the approach proposed in the sequel to the specification of reconfigurable systems builds upon the hybridisation process introduced in Part I. This is done at two levels:

- First of all the introduction of hybrid features on top of the modal language used to specify the overall transition structure, makes possible to refer to individual configurations in an explicit way, leading to more flexible and precise specifications. For example, nominals and the corresponding satisfaction operators give the specifier a “surgical” precision in talking about the system’s configurations.

- On the other hand, through hybridisation of whatever logic is chosen to specify individual configurations, we get a single, powerful logic weaving together local and global aspects to reason about the system. The two conceptual levels mentioned above get unified through the use of a common logic which bears in its own structure local and global means of expression.

- Finally, the institution-based construction used provides a precise way to transport specifications and proofs from a logic to another to seek for appropriate tool support for the method.

These three aspects characterise the methodological contribution of the thesis and, in our opinion, distinguish it from other proposals in the literature.

The approach is depicted in Figure 4, where $I$ stands for the logic used locally to specify individual configurations and $\mathcal{H}I$ is its hybridisation. The former will be referred to as the base logic, to be consistent with the terminology already used in Part I. The picture details our motto:

\begin{center}
reconfigurations as transitions, configurations as local models
\end{center}

This is made concrete in a rather straightforward way. Models for reconfigurable software are structured transition systems: states corresponds to the possible configurations, and transitions to the admissible reconfigurations. The novelty is that the approach is parametric on
how configurations are modelled, or, more precisely, on the base logic. It is exactly in this sense that we claim the method discussed in this chapter to be institution independent.

The upper part of Figure 4 refers to the global level of a specification; the lower one to the local description of configurations. The line in the middle emphasises the role of hybrid features: in a formula they provide a way to name evaluation states, whereas in a model they index the relevant configuration. Actually, each configuration is characterised by a local model capturing its functionality and behaviour. Hybrid models \((M, W)\), discussed in Chapter 3, seem appropriate for this role. Recall \(W\) is a model in \(REL\) interpreting nominals and modalities, whereas \(M\) associates to each state the corresponding (local) model of the configuration it stands for. Of course, the kind of structures these local models possess is determined by the choice of the base logic (institution) \(J\).

We shall now outline the main steps of the specification method proposed and illustrate its application through a small but detailed example. Section 6.3, on its turn, elaborates on the choice of a base logic to meet specific application requirements.

The method depicted in Figure 5, is divided in four phases:

- definition of the specification framework,
- interface description,
- specification of properties and
- analysis and validation.
Example 6.2.1 (A plastic buffer) Our tour through the different stages of the specification method will be illustrated with the following example.

A 'plastic buffer', depicted in Figure 6, is a versatile data structure with two distinct modes of execution: in one of them it behaves like a stack; in the other like a queue. The reconfiguration is triggered by an event 'shift' which may abstract some sort of internal condition, e.g. an increase on the the difference between the rates associated to the incoming and outcoming data streams.
Definition of the specification framework

In this first stage the Software Engineer is supposed to fix the specification framework by generating the specification language through hybridisation of a base logic. This entails the need to instantiating the triple parameter of the hybridisation method by choosing

- the institution representing the base logic suitable to specify individual configurations of the problem at hands,

- the quantification space, and

- the relevant constrained models

These choices are the crucial ones as they fix the working institution and therefore constrains all subsequent development. In particular, the choice of the base institution \( I \) needs to be made only after most of the problem informal requirements were given and clearly understood. One may, for example, specify configurations as multialgebras to cope with non determinism, or with multi-valued logic to deal with uncertainty. Another possibility is partial equational logic to deal with exceptions, or observational logics when the local level encapsulates hidden spaces. Moreover, as mentioned above, in some cases each configuration can be regarded as a transition system itself and a modal language in then in order.

The other two parameters play also relevant roles. The decision about what kind of quantification is to be allowed has a direct impact on the expressiveness of the framework. On the other hand, enforcing additional constraints upon the models provides the technical support to deal with sharing (e.g. of data, operations or both) across configurations. Such constraints may also tune the accessibility relation which expresses the reconfigurations dynamics (imposing, for example, a reflexive or an ordered structure). Both issues are closely related: for example, global quantification over the universes of individual configurations requires the previous choice of a suitable constrained model. In practice the following kinds of quantification are find useful:

- **Quantification over the base institution**, i.e. over the domains of system’s configurations. This can be
– *global*, if a shared universe for all the configurations is chosen and the quantified variables are assumed to be rigid. Actually, this is the typical way to ‘relate computations of’ or to ‘communicate values across’ different configurations.

– *or local*, i.e., ranging over each particular configuration domain. Technically, this is obtained by taking a quantifier-free hybridisation with respect to the first component of the quantification morphisms.

- **Quantification over nominals**, which makes possible to express properties about the systems global state space. This is particularly useful, for instance, to express the existence of configurations satisfying a given requirement.

- **Quantification over modalities**, a rather powerful form of quantification useful to express enabling/disabling of reconfigurations, discussed later in Subsection 7.2.

As usual, when facing these choices the specifier should take into account the compromise between expressiveness and formal tractability. For instance, quantification over modalities should not be used if the aim is to take advantage of the first-order encodings discussed in Chapter 4 to obtain suitable tool support for validating the specifications. Similarly, the relationships between the choice of constrained models and that of quantification spaces cannot be overlooked (cf. Theorem 4.1.1).

**Example 6.2.2 (A plastic buffer)** A specification framework for the ‘plastic buffer’ example must take into consideration the partiality inherent to the reading operations (undefined whenever the buffer is empty). This suggests the adoption of a version of *PA* as the base institution. On the other hand, as the data handled in both configurations is the same, i.e., their definition shared, rigidifications over signatures should be enforced. The combination of these two aspects recommends the choice of *PAR* introduced in Example 3.2.7 as the base institution to fix the specification framework.
Interface description

At this stage all the relevant ‘vocabulary’ to specify the intended system is declared. This includes the enumeration of all configurations to be considered (i.e., the component Nom) and the events triggering reconfigurations (i.e., the component \( \Lambda \)). Moreover, a (local) \( \mathcal{I} \)-signature for the specification of individual configurations (i.e., the component \( \Sigma \)) is required. In a sense, the latter can be understood as the actual interface of the system since the services and functions offered at each configuration must be declared at this stage.

In order to represent hybrid signatures in an uniform way we resort to a terse notation, close to that of the specification language CASL [ABK⁺02]. In particularly, we use the following header:

\[
\text{spec } \text{SpecName in HBaseInst} =
\begin{align*}
\text{Nom} & \quad \text{list of nominal symbols} \\
\text{Modal} & \quad \text{list of modality symbols} \\
\text{BaseSig} & \quad \text{signature of the base institution} \\
\text{Axioms} & \quad \text{set of axioms}
\end{align*}
\]

\text{SpecName} is the specification identifier, and \text{BaseInst} stands for the base institution \( \mathcal{I} \) adopted when fixing the specification framework. The fields \text{Nom} and \text{Modal} are used for the declaration of nominal and modality symbols, respectively. Arities of modalities are given by natural numbers and are presented in front of the respective symbol (e.g. event:\(k\) means that the modality event has arity \(k\)). The parameter \text{BaseSig} is the declaration of the \( \mathcal{I} \)-signature \( \Sigma \). Since the structure of those signatures is specific for each hybridisation, there is no way to fix the syntax for this declaration. Finally, the field \text{Axioms} contains the specification expressed in axioms of the chosen logic.

Example 6.2.3 (A plastic buffer) The ‘plastic buffer’ has to commute between two different configurations denoted by nominals \( \text{fifo} \) (for the queue mode) and \( \text{lifo} \) (for the stack mode), respectively. The event responsible for triggering a reconfiguration is denoted by the modality
symbol \textit{shift}. This makes up the global, \textit{REL}-based component of the hybrid signature. Basically, its role is to express the system’s reconfiguration semantics.

The local interface is a \textit{PAR}-signature with the following components:

- a sort \textit{mem} of stacks/queues;
- a sort \textit{elem} of elements;
- a total operation \textit{write} to denote the ‘push/enqueue’ operation;
- a constant \textit{new} to denote the empty buffer;
- a partial operation \textit{read} to denote the ‘top/front’ operation;
- a partial operation \textit{del} to denote the ‘pop/dequeue’ operation.

On its turn rigidification is discussed as follows:

- since the buffer stores data of the same type in both configurations, \textit{elem} and \textit{mem} are declared to be rigid;
- operation \textit{write} is also rigid and, since it is total, it has the same effect in both configurations;
- operations \textit{read} and \textit{del} play different roles in each individual configuration. However, both are declared as rigid because they are partial and in \mathcal{HPAR} this means that while their interpretation might differ from one configuration to another, their domains remain fixed (cf. Example 3.2.7). In this case both \textit{read} and \textit{del} are defined on non-empty stacks/queues.

The local, \textit{PAR}-based component of the hybrid signature is depicted in Figure 7 in an ADJ-like diagram. Partiality and rigidity of operations is marked by a circle and a ramified source, respectively. The hybrid signature is given as follows, where symbol \textbf{R} is used to mark the rigid components:

\begin{verbatim}
Nom
 fifo
 lifo
\end{verbatim}
Modal

\textit{shift} : 1

BaseSig

\textbf{sorts} mem R

elem R

\textbf{ops} new : \rightarrow mem R

write : mem \times elem \rightarrow mem R

del : mem \rightarrow? mem R

read : mem \rightarrow? elem R

Specification of properties

Properties, both local and global, are introduced at this stage. They correspond to requirements placed at different levels of abstraction. In particular, the Software Engineer has to consider

1. the global properties, i.e., properties holding in all the configurations, which are expressed through (atomic) \( I \)-sentences;

2. the local properties, i.e. relative to specific configurations, which are expressed by tagging \( I \)-sentences with the satisfaction operator \( @ \) (\( @_i \rho \) is used to express that property \( \rho \) holds in the configuration named by \( i \));

3. and, finally, the reconfiguration dynamics whose specification resorts to the modal features introduced in the hybridisation process.
Example 6.2.4 (A plastic buffer) For the ‘plastic buffer’ example our starting point is the classical specification of queues and stacks in partial equational logic. Then satisfaction operators are used to assign each set of local properties to the corresponding configuration. For the local specification of configuration lifo we use

\[\forall e : \text{elem}; \forall m : \text{mem};\]
\[\bullet \ @_{\text{lifo}} \text{del}(\text{write}(m, e)) \equiv m\]
\[\bullet \ @_{\text{lifo}} \text{read}(\text{write}(m, e)) \equiv e\]

Similarly, for the local specification of configuration fifo,

\[\forall e : \text{elem}; \forall m : \text{mem};\]
\[\bullet \ @_{\text{fifo}} \text{read}(\text{write}(\text{new}, e)) \equiv e\]
\[\bullet \ @_{\text{fifo}} \text{del}(\text{write}(\text{new}, e)) \equiv \text{new}\]
\[\bullet \ @_{\text{fifo}} \text{read}(m) \equiv \text{read}(m) \Rightarrow \text{read}(\text{write}(m, e)) \equiv \text{read}(m)\]
\[\bullet \ @_{\text{fifo}} \text{read}(m) \equiv \text{read}(m) \Rightarrow \text{del}(\text{write}(m, e)) \equiv \text{write}(\text{del}(m), e)\]

Other properties could also be considered to avoid ‘anomalous’ models. Actually, as it always happens in loose semantics, the specification admits other models behind the ones in which all configurations are distinguished and suitably identified by nominals. Such ‘junk configurations’ exist, as well as one may have to deal with ‘confusion on the configurations’ whenever different nominals identify the same state. Both situations can be avoided by considering some additional axioms. In particular, the first case is ruled out by including

\[\text{lifo} \lor \text{fifo}\]

as an axiom. Similarly

\[\neg @_{\text{lifo}} \text{fifo}\]

or, equivalently, \(\neg @_{\text{fifo}} \text{lifo}\), avoids confusion.

Finally, the reconfiguration dynamics is given by

\[\bullet @_{\text{fifo}}(\text{shift}) \text{lifo} \land @_{\text{lifo}}(\text{shift}) \text{fifo}\]

expressing configuration alternation triggered by event shift. The complete specification is depicted in Figure 8.
spec PlasticBuffer in HPAR =

nom  
      fifo  
      lifo
mod
      shift : 1
BaseSig
  sorts mem R  
      elem R
  ops new : → mem R
      write : mem × elem → mem R
      del : mem → ? mem R
      read : mem → ? elem R

\forall e : elem; \forall m : mem;

• @lifo.del(write(m, e)) e = m  
• @lifo.read(write(m, e)) e = e

% (stack properties)%

• @fifo.read(write(new, e)) e = e
• @fifo.del(write(new, e)) e = new
• @fifo.read(m) e = read(m) ⇒ read(write(m, e)) e = read(m)
• @fifo.read(m) e = read(m) ⇒ del(write(m, e)) e = write(del(m), e)

% (reconfigurations spec)%

• fifo \lor lifo lifo
• @fifo(shift).lifo \land @lifo(shift).fifo

end

Figure 8: Plastic Buffer \( \mathcal{H} \text{PAR} \) Specification.
Note that the quantification is *local* which means it is placed at the level of the base institution (hence the use of notation \(\forall\); see Fact 3.1.3). For example the first sentence in the specification reads as \(\@_{lifo}(\forall e)(\forall m)\text{del}(\text{write}(m,e)) = m\).

### Analysis and validation

The construction and the analysis of particular models of a specification is a most relevant step in the design process. In a sense, it can be understood as a high level implementation of the specification, a first prototype acting as a proof-of-concept for the system. Once a model is available, its validation, i.e. the systematic verification of the specified properties, becomes crucial. Although this can be done with ‘paper and pencil’, the availability of computational proof-support tools is a necessary condition for the methodology to be considered a viable alternative in the software industry.

There is a number of provers available for propositional hybrid logics. They can of course be of use when dealing with hybrid(ised) propositional logic. Among the implementations of logical calculi for HPL we single out HTAB [HA09], HyLoTAB [vE02] and SPARTACUS [GKS10]). Other works, for example [Lan09, HS08], study model checking procedures for hybrid propositional models.

Unfortunately, propositional hybrid logic has a limited use in the specification of reconfigurable systems. Actually, it only suits the case in which states have no structure, i.e. when the local description of individual configurations is irrelevant. On the other hand, and to the best of our knowledge, there is no effective tool support for richer hybrid logics. The method proposed in this thesis, considering first order encodings of hybridised logics emerges as a promising solution to support a wide class of specifications. Such encodings in FOL presentations were introduced in Chapter 4. Whenever they are supported by the base institution (and in practice they often are as most logics used in specifications admit them), there are a number of tools for validating the (translated) specification. For example, SPASS [WDF\textsuperscript{+}09], VAMPIRE [RV02], PROVER9 [McC10] and DARWIN [BPT12] are well
known examples of successful automatic theorem provers. Model finding technics, as implemented, for example, in MACE4 [McC10], can also help in checking consistency or disproving statements (by the generation of suitable counter-examples).

**HETS** — the *Heterogeneous Tool Set* [MML07, MMCL13], provides a flexible interface to interrelate logics and the corresponding support tools. In particular, most of the *FOL*-provers mentioned above are already accessible from **HETS**. The integration of the hybridisation process proposed in this thesis into the **HETS** framework is discussed in Section 7.1.

**Example 6.2.5 (A model for the plastic buffer)** Let us consider a model $\langle M, W \rangle$ for the specification of a ‘plastic buffer’ discussed above. The global, REL component of the signature is interpreted as follows:

- $|W| = \{\text{sfifo, slifo}\}$;
- $W_{\text{lifo}} = \text{slifo}$ and $W_{\text{fifo}} = \text{sfifo}$;
- $W_{\text{shift}} = \{(\text{sfifo, slifo}), (\text{slifo, sfifo})\}$.

A buffer over an arbitrary data type $A$ can be implemented by a sequence of $A$ values, i.e. an element of $A^*$. The sequence concatenation is denoted by $\_\_\_\_$ and $\epsilon$ stands for the empty sequence.

- $(M_{\text{sfifo}})_{\text{elem}} = (M_{\text{slifo}})_{\text{elem}} = A$;
- $(M_{\text{sfifo}})_{\text{mem}} = (M_{\text{slifo}})_{\text{mem}} = A^*$;
- $(M_{\text{slifo}})_{\text{new}} = (M_{\text{slifo}})_{\text{new}} = \epsilon$;
- $(M_{\text{sfifo}})_{\text{write}}(L, a) = (M_{\text{slifo}})_{\text{write}}(L, a) = L.a$;
- $(M_{\text{slifo}})_{\text{del}}(L) = J$ if $L = J.a$ for $J \in A^*$, $a \in A$ and is undefined otherwise;
- $(M_{\text{sfifo}})_{\text{del}}(L) = J$ if $L = a.J$ for $J \in A^*$, $a \in A$ and is undefined otherwise;
- $(M_{\text{slifo}})_{\text{read}}(L) = a$ if $L = J.a$, for $J \in A^*$, $a \in A$ and is undefined otherwise;
One may now conjecture whether the following properties are valid.

\[(\forall e, e', m, m')\]
\[\langle \text{shift} \rangle (m' = \text{write}(e, m)) \iff (m' = \text{write}(e, m)) \quad (21)\]

\[(\forall m, m', m_1, m_2) (m' = \text{del}(m_1)) \land \langle \text{shift} \rangle (m_1 = \text{del}(m))
\iff ([\text{shift}](m' = \text{del}(m_2)) \land (m_2 = \text{del}(m)) \quad (22)\]

To proceed we start building the FOL encoding of the specification.

**Example 6.2.6 (A first-order encoding)** We shall now build the encoding of the ‘plastic buffer’ $\mathcal{HPAR}$ specification into FOL, following closely the procedure discussed in Example 4.2.8.

The starting point is, of course, the definition of the signature:

**logic** CASL.FOL

**spec** PLASTICBUFFERFOL =

- **sorts** $ST$;
  - $mem$;
  - $elem$
- **ops** $fifo: ST$;
  - $lifo: ST$;
  - $new: ST \rightarrow mem$;
  - $write: ST \times mem \times elem \rightarrow mem$;
  - $read: ST \times mem \rightarrow elem$;
  - $del: ST \times mem \rightarrow mem$
- **preds** $shift: ST \times ST$;
  - $Def_mem: ST \times mem$;
The formal verification of properties (20), (21) and (22) is performed as follows:

1. We translate the properties by using the $\mathcal{H}PAR'$ instance of our general encoding (as given in Ex. 4.2.8);

2. By the conservativity property of the encoding of $\mathcal{H}PAR'$ (see Ex. 4.3.7), according to the general result of Fact 2.3.2 it is enough to prove that the translations of the properties are a consequence of the translation of the specification. The corresponding proofs are performed by using SPASS [WDF+09] automatic first order logic prover through the Hets system [MMCL13].
∀ e : elem; m : mem
- Def_mem(lifo, m) ∧ Def_elem(lifo, e) 
  ⇒ Def_mem(lifo, m) ∧ del(lifo, write(lifo, m, e)) = m
- Def_mem(lifo, m) ∧ Def_elem(lifo, e) 
  ⇒ Def_elem(lifo, e) ∧ read(lifo, write(lifo, m, e)) = e
- Def_elem(fifo, e) 
  ⇒ Def_elem(fifo, e) ∧ read(fifo, write(fifo, new(fifo), e)) = e
- Def_elem(fifo, e) 
  ⇒ Def_mem(fifo, new(fifo)) ∧ del(fifo, write(fifo, new(fifo), e)) = new(fifo)
- Def_mem(fifo, m) ∧ Def_elem(fifo, e) 
  ⇒ Def_elem(fifo, read(fifo, m)) ∧ read(fifo, m) = read(fifo, m) 
  ⇒ Def_elem(fifo, read(fifo, m)) ∧ read(fifo, write(fifo, m, e)) = read(fifo, m)
- Def_mem(fifo, m) ∧ Def_elem(fifo, e) 
  ⇒ Def_elem(fifo, read(fifo, m)) ∧ read(fifo, m) = read(fifo, m) 
  ⇒ Def_mem(fifo, write(fifo, del(fifo, m), e)) ∧ del(fifo, write(fifo, m, e)) = write(fifo, del(fifo, m), e)

∀ w : ST • w = fifo ∨ w = lifo % (no_junk)

∃ y, z : ST • (shift(fifo, y) ∧ y = lifo) ∧
(shift(lifo, z) ∧ z = fifo)

end

Figure 9: The ‘plastic buffer’ specification translated into FOL.
6.3 DEALING WITH HETEROGENEOUS REQUIREMENTS

An approach to the specification of reconfigurable systems based on hybridisation was introduced in the previous section. Its application was illustrated with a small example based on PAR-hybridisation. Similar illustrations, corresponding to different case studies and hybridising different logics appeared in [MFMB11] and [MMB13a]. The former relies on hybridised first order logic, $\mathcal{H}FOL$, the latter on hybridised equational logic, $\mathcal{H}EQ$.

In this section we intend to further explore the institution-independent nature of the proposed method. We proceed by looking at some fragments of another case study in the design of reconfigurable software: the specification of an insulin infusion pump based on a collection of requirements collected in [ZJJR11]. The complete case study will appear in a forthcoming MSc dissertation [Nev13]. Its interest for our purposes lies in the heterogeneity of the requirements which entails the need for adopting different base institutions to specify different fragments of the overall system. We shall discuss some of these cases.

The first class of requirements concerns only the relationship between different modes of operation of the insulin pump. The most trivial one is the specification of the ‘pump switch’:

\[(R1)\text{ The pump can be turned on or off} \]

We may identify each of these modes as configurations On and Off through which the system commutes executing events TurnOn and TurnOff. Since there is no further local information, the hybridisation of TRM is enough to express the system, as shown in Figure 10.

An usual practice, however, is to characterise the configurations through a set of propositions. To tune our framework to this case, propositional logic, $PL$, is chosen as the base institution. For example, suppose the specification of the pump switch above is enriched with the requirement that

\[(R2)\text{ In order to be turned on, a check of the sensors integrity should be fulfilled}\]

Since no further information on what a check is is given, we abstract it as a proposition $ready$, assumed to be true whenever the system passes
the test, and false otherwise. The resulting specification is formulated as in Figure 10 but considering a proposition ready as the base signature and replacing the second axiom by

$$\text{Off} \land \text{ready} \iff \langle \text{TurnOn} \rangle \text{On}.$$ 

In some cases a transition system is also adequate to model individual configurations themselves. From a global point of view this leads to transition structures of transition structures, a scenario which is not so uncommon in software design. Actually, it underlies methodologies such as Hierarchical State-Machines in UML [Gro] or David Harel’s statecharts [Har87]. In our own approach, if $\mathcal{H}PL$ is taken as the adequate institution to specify transition structures, $\mathcal{H}'\mathcal{H}'PL$ is the suitable specification framework for the double scenario. To illustrate this suppose the following additional requirement:

(R3) When on the pump can be suspended or run normally

It is clear from requirement (R3) that configuration $\text{On}$ (denoted in this hierarchical setting by $\text{On}^1$) has two possible (sub)-states, identified by $\text{Nor}^0$ and $\text{Sus}^0$, that can be achieved by the (internal) events $\text{TurnNor}^0$ and $\text{TurnSus}^0$ respectively. This can be collected in the $\mathcal{H}TRM$ base signature

$$(\ast, \{\text{Nor}^0, \text{Sus}^0\}, \{\text{TurnNor}^0 : 1, \text{TurnSus}^0 : 1\}).$$

<table>
<thead>
<tr>
<th><strong>spec</strong></th>
<th><strong>SWITCH in HTRM =</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Nom</strong></td>
<td>$\text{On: Off}$</td>
</tr>
<tr>
<td><strong>Modal</strong></td>
<td>$\text{TurnOff:1; TurnOn:1}$</td>
</tr>
<tr>
<td><strong>BaseSig</strong></td>
<td>$\ast$</td>
</tr>
<tr>
<td><strong>Axioms</strong></td>
<td>$\text{On } \lor \text{ Off}$</td>
</tr>
<tr>
<td></td>
<td>$\neg \text{On} \langle \text{TurnOff} \rangle \text{Off} \land \neg \text{Off} \langle \text{TurnOn} \rangle \text{On}$</td>
</tr>
</tbody>
</table>
Hence, we may express (R3) with
\[ \@_{On^1} \left( [TurnSus^0]Sus^0 \land [TurnNor^0]Nor^0 \right), \]
expressing the expected ‘internal inactivity’ of Off^1 with
\[ \@_{Off^1} \left( \neg (TurnSus^0)\text{true} \land \neg (TurnNor^0)\text{true} \right), \]
where the symbol \text{true} stands for a tautology as, for instance, \@i\text{true} for some \(i \in \{Sus^0, Nor^0\}\).

Systems whose configurations evolve temporally can be captured in our approach by taking a linear discretisation of time as the reconfiguration space. Configurations are then time instants and a modality \text{after} relates them chronologically. Nominals, of course, refer to such instants. As one would expect, taking time into the picture is independent of the base institution chosen. For example, requirement

\textit{(R4) If the reservoir is found to be empty then the alarm must be activated;}

can be captured in this framework. A simple formalisation would be the \(\mathcal{HPL}\)-sentence
\[ \neg \text{full} \Rightarrow (\text{after})\text{alarm}. \]

For a more realistic specification one may like to take into account other information, for example the level of confidence assigned to sensors. Multi-valued logics, as the natural setting to deal with vagueness and degrees of certainty, are good candidates for this sort of requirements.
To go in this direction, let us consider the hybridisation of $MVL_3$, the multi-valued logic on the residuated lattice $3$ with truth-values: $\{\top, u, \bot\}$. Hence, we may capture (R4) in a more realistic way by

$$(\text{full}, \top) \land (\text{after}) (\text{full}, u) \Rightarrow (\text{after}) (\text{next}) (\text{alarm}, \top)$$

which reads "if the reservoir is full at given a moment, but its level is unknown in the following one, then an alarm should be triggered subsequently". It is also possible to assume continuous degrees of true by using $\mathcal{H}MVL_L$ for $L$ the Lukasiewicz arithmetic lattice. This would allow us to express properties like "if the reservoir is full at a given a moment, and full with a degree of confidence of 0.5 in the following one, then the alarm should be triggered subsequently with a confidence of 0.7".

$$(\text{full}, 1) \land (\text{next}) (\text{full}, 0.5) \Rightarrow (\text{next}) (\text{next}) (\text{alarm}, 0.7).$$

This small example gives a flavour of what can be achieved by combining different hybridised logics to approach different classes of requirements. Actually, to whatever logic one may think of as appropriate for the project of reconfigurable software, corresponds a suitable institution to be taken as the base of a hybridisation process. Besides the cases illustrated in this chapter, one may also mention the institutions corresponding to hiding and observational logics [BD94, BH06], the logics of functional FPL and imperative programming introduced in IMP [ST12], or HasCASL [SM09]. At the time of writing we are developing an institution to capture the logic underlying Alloy [Jac11] descriptions and studying its hybridisation [NMMB13a].

Table 1 enumerates the list of hybridisations which are discussed in this thesis or in the author’s current work. For each of them the base logic, quantification scheme and additional model constraints are pointed out, as well as the existence of a an encoding to $FOL$ and a pointer to the relevant result.
<table>
<thead>
<tr>
<th>$\mathcal{H}$</th>
<th>Base Logic</th>
<th>$J$ form.</th>
<th>Model Constraints</th>
<th>Quantifications</th>
<th>$\mathcal{H}_1$</th>
<th>$\mathcal{H}_2\text{FOL}$</th>
<th>$\mathcal{H}_2\text{FOL}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\mathcal{H}_{TRM}$</td>
<td>–</td>
<td>Ex 2.3.1</td>
<td>free</td>
<td>free</td>
<td>Ex. 3.2.2</td>
<td>Ex 2.3.13</td>
<td>Ex 4.2.2</td>
</tr>
<tr>
<td>$\mathcal{H}_{PL}$</td>
<td>Propositional L.</td>
<td>Ex 2.3.5</td>
<td>free</td>
<td>free</td>
<td>Ex. 3.2.1</td>
<td>Ex 2.3.14</td>
<td>Ex 4.2.1</td>
</tr>
<tr>
<td>$\mathcal{H}_{2PL}$</td>
<td>Hybrid Prop. L.</td>
<td>Ex 2.3.6</td>
<td>sharing of univ.</td>
<td>free</td>
<td>Ex. 3.2.3</td>
<td>Ex 4.2.1</td>
<td>Ex 4.2.3</td>
</tr>
<tr>
<td>$\mathcal{H}_{REL}$</td>
<td>Relational L.</td>
<td>Ex 2.3.4</td>
<td>free</td>
<td>cts. ext.</td>
<td>Ex 3.2.5</td>
<td>Ex 4.2.5</td>
<td>Ex 4.2.5</td>
</tr>
<tr>
<td>$\mathcal{H}_{REL}'$</td>
<td>–</td>
<td>Ex 3.2.5</td>
<td>sharing of univ. and const.</td>
<td>cts. ext.</td>
<td>Ex 3.2.5</td>
<td>Ex 4.2.5</td>
<td>Ex 4.2.5</td>
</tr>
<tr>
<td>$\mathcal{H}_{FOL}$</td>
<td>First-order L.</td>
<td>Ex 2.3.2</td>
<td>free</td>
<td>cts. ext.</td>
<td>Ex 3.2.4</td>
<td>Ex 4.2.7</td>
<td>not enc.</td>
</tr>
<tr>
<td>$\mathcal{H}_{FOLR}$</td>
<td>–</td>
<td>Ex 3.2.4</td>
<td>sharing of univ., op. and pred.</td>
<td>rigid. cts. ext.</td>
<td>Ex 3.2.4</td>
<td>Ex 4.2.7</td>
<td>Ex 4.2.7</td>
</tr>
<tr>
<td>$\mathcal{H}_{PAR}$</td>
<td>–</td>
<td>Ex 3.2.7</td>
<td>sharing of univ., op. and pred.</td>
<td>rigid. total cts ext.</td>
<td>Ex 3.2.7</td>
<td>Ex 4.2.8</td>
<td>Ex 4.2.8</td>
</tr>
<tr>
<td>$\mathcal{H}_{MVVL}$</td>
<td>Multi-valued L.</td>
<td>Ex 2.3.8</td>
<td>free</td>
<td>const. ext.</td>
<td>Ex 3.2.8</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\mathcal{H}_{Alloy}$</td>
<td>Alloy Spec L.</td>
<td>[NMMB13a]</td>
<td>sharing univ. and const.</td>
<td>const. ext.</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 1: Hybridisations.
This chapter introduces a number of techniques which complement or enrich the specification method discussed in Chapter 6. Section 7.1 discusses the incorporation of the hybridisation process into the HETS framework. This is most relevant, from a pragmatically point of view, in order to provide computer based support to reason about specifications written in hybridised logics.

Sections 7.2 and 7.2, on the other hand, discuss possible extensions to the specification method. The former aims at accommodating systems in which the interfaces of local configurations vary. The latter discusses the case in which reconfigurations are triggered by events depending on actual values of local state variables.

The qualifier *complementary* emphasises not only their role in assisting the specification process, but also their still exploratory character. Actually, their full development in a complete general setting is still part of our current research.

### 7.1 Validation

A central ingredient for the successful integration of a formal methodology in the industrial practice is the existence of effective computational tools to assist and support the specification and validation processes.

**HETS, the heterogeneous tool set** [MML07] emerges as a good candidate for the job. Using a metaphor of [MMCL13], HETS may be seen as a “motherboard” where different “expansion cards” can be plugged. These pieces are individual logics (with their particular analysers and proof tools) as well as logic translations. HETS already integrates parsers, static analysers and provers for a wide range of individual logics and manages heterogeneous proofs resorting to the so-called graphs of logics.
As a generic framework, ‘agnostic’ with respect to specific logics, HETS combines well with the institution-independent character of the methodology proposed in this thesis. Moreover, the common grounds on institution theory make this combination easier. HETS is also endowed with a very rich support to specification in FOL, a quite relevant aspect in order to take advantage of the general encodings developed in Chapter 4. In particular, the theorem provers SPASS [WDF+09], VAMPIRE [RV02], EPROVER [Sch02], E-KRHYPER [PW07] and Darwin [BFT06] are already plugged to the framework. Therefore, the use of HETS as a suitable interface to FOL provers emerged as a complement, and possibly an alternative, to the carrying out proofs and building encodings manually (as in [MFMB11, MNMB13, DM]) when dealing with specifications written in hybridised logics.

As usual, the adoption of computer based support increases the methods’s applicability in practice. A first move into this direction was to expand HETS’ graph-of-logics incrementally, hybridisation by hybridisation and comorphism by comorphism. Moreover, since both the hybridisation method and the lifting of first-order encodings are effective procedures, it is legitimate to take a more ambitious approach by directly implementing them over the HETS’ graph-of-logics. This section discusses some of these steps recently presented in [NMMB13b].

As a first step we integrated the hybridisation of CASL [MHST03] in HETS. A comorphism from the resulting H\textit{CASL}¹ to CASL was also defined. Thus, assisted proof support for H\textit{CASL} becomes available for free. H\textit{CASL} specifications add to the usual ones in CASL a declaration of nominals and modalities. Sentences include the typical hybrid machinery as well as quantification over nominals. Thus, the corresponding grammar is extended as follows:

\[
\text{CFro'} = \text{HFor} | \ldots ; \\
\text{\textit{HFor}} = \@n \text{CFro'} | <m> \text{CFro'} | \[m] \text{CFro'} | \text{Here } n | ! n \text{CFro'} | ? n \text{CFro'};
\]

where \( n \) is a nominal, \( m \) a modality, and the last two cases denote, respectively, universal and existential quantification over nominals. Figure 12 depicts a code fragment of a H\textit{CASL} specification — the specification of the \textit{reconfigurable calculator}, from Example 7.3.1, in the \textit{H\textit{EQ}} fragment of this logic.

¹ where a constraint over models enforcing sharing of universes and quantification over rigid variables is assumed.
logic Hybrid
spec Reconf_Calc =
HNat then
nominals
Sum,Mul
modalities
Shift
ops
x_op : Nat * Nat -> Nat
%% global properties
forall n,m,p : Nat
n ≠ m ⇒ n # m = m # n
(n # m) # p = n # (m # p)
x_op(n,p) = x_op(m,p)
%% configurations properties
forall n,m : Nat
  @Sum x_op(n,c) = n
  @Sum suc(n) = x_op (n,suc(c))
  @Mul x_op(n,c) = c
  @Mul x_op(n,suc(c)) = n
%% reconfigurability properties
   Here Sum /\ Here Mul
   @Sum(<Shift> Here Mul /\ [Shift] Here Mul)
   @Mul(<Shift> Here Sum /\ [Shift] Here Sum)

Figure 12: The reconfigurable calculator in $\mathcal{H}$CASL.

logic Hybridise
spec X =
baselogic Hybridise
Basic_Spec {
baselogic Propositional
Basic_Spec { props p }
nominal Test,Off,Nor,Sus
modality Sus,Res,NotOk,Activate
Test \ / Off \ / Nor \ / Sus
}
nominal On,Off
modality TurnOn,TurnOff
%% The possible super states
On \ / Off;
%% Defining and restricting relations between the
%% sub states in the super state On
@On { @Nor <Sus>* Sus /\ @Sus <Res>* Nor };
@On { ( <Sus> true -> Nor /\ <Res> true -> Sus ) /\ not ( <NotOk> true \ / <Activate> true ) };
%% Defining and restricting relations between
%% the sub states in the super state Off
@Off { @Test <NotOk>* Off /\ @Off <Activate>* Test };
@Off { ( <NotOk> true -> Test /\ <Activate> true -> Off ) /\ not ( <Res> true \ / <Sus> true ) };
%% Defining and restricting relations between the
%% super and sub states
Off \ / {Test} <-> ( <TurnOn>* (On \ / (Nor)) )
<TurnOn> true -> Off \ / (Test);
On \ / {Nor} <-> ( <TurnOff>* (Off \ / (Off)) )
<TurnOn> true -> Off \ / (Off)

Figure 13: A specification parsed by the generic engine.
A few steps were also taken into the more ambitious direction of a direct implementation of the hybridisation method. In particular, we implemented a generic parser for hybridising institutions already supported by HETS. Currently, *propositional* logic, CASL and CoCASL have been fully hybridised and made available in the platform. Additionally, any previously hybridised logic, can again be hybridised. In each case the resulting hybridised specifications composes a specification in the base logic with the declaration of nominals and modalities and the sentences enriched with hybrid properties. Figure 13 illustrates how the specification in $\mathcal{H}_2^2PL$ discussed in the previous chapter looks like in this setting. Note that the underline notation is replaced here by wrapping the sentences between curly brackets (c.f. 3.2.3).

Other features of HETS can be explored in the context of the methodology proposed here. For instance, the model finder associated to DARWIN, already integrated in HETS, may be used as a consistency checker for hybridised specifications. Moreover, available encodings of FOL into HasCASL, a specification language for functional programs, open further perspectives for validating specifications in hybridised logics (see [MMCL13]).

### 7.2 Evolving Interfaces

The approach to the specification of reconfigurable systems introduced in the previous chapter assumes that all configurations share the same signature, i.e., the interface provided at any local state is fixed. Or, to put it in yet another way, that the system’s interface is invariant with respect to the reconfiguration process.

In practice, however, this may be a too strong assumption. Actually, not only the realisation of a service may change from a configuration to another, but also the set of services provided may itself vary. In other words, sometimes in reconfigurable systems the local interfaces may evolve as well.

Although taking up this challenge in a completely general setting would require a substantial review of the method, a partial answer can be obtained by exploring the generated (hybrid) languages. This section proposes a technique, published in [MNMB13], to deal with in-
terface reconfiguration whenever the local specifications are given in \( EQ \). We want to allow not only a possibly different algebra in each state, but also different algebras over different signatures. Technically, this is achieved through the introduction of (hybrid) partial algebra specifications to “simulate” the intended, independent (hybrid) equational ones. Note, however, that, even resorting to partial specifications, models will always be (total) algebras with respect to the corresponding local interface.

As in chapter 6, let us suppose

- a set of relevant configurations named by the set of nominals \( \text{Nom} \);
- and a family of modalities \( \Lambda \) to trigger reconfigurations.

Suppose, however, that in the place of a unique (static) interface \((S,F)\), we consider

- a family \((S^i,F^i)_{i \in \text{Nom}}\) of local signatures, indexed by the set of nominals.

The technique proceeds by building a presentation

\[
((S, TF, PF), \text{Nom}, \Lambda, \Gamma) \in \text{Sign}_{\mathcal{HPA}_{\text{pres}}}
\]

in the institution \( \mathcal{HPA}_{\text{pres}} \) of presentations over \( \mathcal{HPA} \), where all this information can be considered, and the hybridisation process discussed in Subsection 6.2 applied.

The first step is to define a signature \((S, TF, PF)\) in \( PA \) able to capture all the possible interfaces. Thus, operations are split into the ones which are globally defined (i.e., presented in any \((S^i,F^i)\), for \( i \in \text{Nom} \)) and those which concern only to a specific state. These two sets of operations define a (global) \( \mathcal{HPA} \)-signature:

\[
S = \bigcup_{i \in \text{Nom}} S^i
\]

\[
\text{TF}_{ar \rightarrow s} = \{ \sigma | \sigma \in \bigcap_{i \in \text{Nom}} F^i_{ar \rightarrow s} \}
\]

\[
\text{PF}_{ar \rightarrow s} = \{ \sigma | \sigma \in (F^i)_{ar \rightarrow s} \setminus \text{TF}_{ar \rightarrow s}, i \in \text{Nom} \}
\]

Thus, we recovered an unique base signature to proceed with the specification. However, the information about which of those operations are
defined in which configuration has yet to be considered. This is done by the following axioms:

\[ \Gamma = \{ @_i(\forall X)df(\sigma(X))|\sigma \in (F^i)_{ar\to s} \cap PF_{ar\to s}, i \in \text{Nom}\} \cup \{ \neg @_i(\exists X)df(\sigma(X))|\sigma \in PF_{ar\to s} \setminus (F^i)_{ar\to s}, i \in \text{Nom}\} \]

Therefore, one ends up with a presentation

\[ ((S, \overline{F}, PF), \text{Nom}, \Lambda, \Gamma) \]

collecting all the intended information on the interfaces. The specification method can be safely applied from this point on. In broad terms, we are going to simulate local, total functions with global, partial ones. This entails the need for adopting strong equality to specify “global properties” of operations defined in a specific configuration.

For instance, any existential equation \( t = t' \) involving operations in PF is inconsistent because it fails on configurations where these operations are not defined. Of course, this is not the case of existential equations prefixed by satisfaction operators, i.e., of sentences of form \( @_i(t = t') \). But, in general, this is not enough: all operations must be “locally”-total or “completely”-undefined.

**Example 7.2.1**

Suppose that, in the context of a client server architecture, a buffering component is required to store and manage incoming messages from different clients. Depending on the server’s execution mode, i.e., on its current configuration, issues like the order in which calls have arrive or the number of repeated messages may, or may not, be relevant. Therefore, the shape of the buffering component may vary, typically being determined by an external manager.

A model for this component comprises four kinds of configurations endowed with, respectively,

i) an algebra of sequences (for configurations where both order and multiplicities are relevant issues),

ii) an algebra of multi sets (when the order may be left out),
iii) an algebra of \textit{sets} (when the application may abstract over order and repetitions), and finally

iv) an algebra of \textit{repetition free sequences} (to cater only for the messages’ order).

Going from one configuration to another involves not only a change in the way a service is realised (\textit{e.g.}, insertion clearly differs from one state to the other), but also a change at the \textit{interface} level. For example, an operation to count the number of replicated messages does not make sense if sets are used as a local model.

We start by defining a set $\text{Nom} = \{\text{OM, Om, oM, om}\}$ of nominals, where the capitalised letters correspond to the relevance of order and multiplicity issues (for instance, Om refers to a configuration where order, but not multiplicity, is the relevant issue). Then, for the reconfigurations events, take a set of modal symbols $\Lambda = \{\text{goto}_\text{OM, goto}_\text{Om, goto}_\text{oM, goto}_\text{om}\}$.

Consider now the local interfaces. For $(S^\text{om}, F^\text{om})$ choose the usual signature of Sets comprising the set of sorts $S^\text{om} = \{\text{Elem, Store, Bool}\}$ and operation symbols $F^\text{om}_\text{Store} = \{\text{empty}\}$, $F^\text{om}_\text{Elem} \times \text{Store} \rightarrow \text{Bool} = \{\text{is}_\text{in}\}$; $F^\text{om}_\text{Elem} \times \text{Store} \rightarrow \text{Store} = \{\text{insert}\}$; and $F^\text{om}_\text{ar} \rightarrow \text{s} = \emptyset$ for the other arities. Clearly, $(S^\text{om}, F^\text{om}) = (S^\text{om}, F^\text{om})$. The remaining cases need to deal with multiplicities; therefore signatures have to be enriched with new operations. Hence, $(S^\text{OM}, F^\text{OM})$ can be defined as $S^\text{OM} = S^\text{om} \uplus \{\text{Nat}\}$ and $F^\text{OM}_\text{Elem} \times \text{Store} \rightarrow \text{Nat} = \{\text{mult}\}$ and $F^\text{OM}_\text{ar} \rightarrow \text{s} = F^\text{om}_\text{ar} \rightarrow \text{s}$ for the other arities. Again, $(S^\text{OM}, F^\text{OM}) = (S^\text{OM}, F^\text{OM})$. Therefore, the following “global” partial signature is defined: $(\langle S, TF, PF, \text{Nom, } \Lambda, \Gamma \rangle)$ taking $S = \bigcup_{i \in \text{Nom}} S^i = S^\text{OM}$, $TF = F^\text{om}$ and $PF_{\text{Elem} \times \text{Store} \rightarrow \text{Nat}} = \{\text{mult}\}$ and $PF_{\text{ar} \rightarrow \text{s}} = \emptyset$ for the other arities. On its turn, $\Gamma$ is defined by the sentences

$$\forall i (\forall s) (\forall e) \text{df}(\text{mult}(e, s)),$$ for $i \in \{\text{OM, OM}\}$

$$\neg \forall i (\forall s) (\forall e) \text{df}(\text{mult}(e, s)),$$ for $i \in \{\text{om, Om}\}$.

In this setting, we may now proceed with the specification of the global properties, as for example,

$$(\forall e : \text{elem}) \text{ is}_\text{in}(e, \text{empty}) = \text{False}$$
For the local properties one resorts to the hybrid satisfaction operator. This allows, for example, to record the fact that ordering and the multiple insertion are irrelevant for the configuration \( \text{om} \):

\[
@_{\text{om}} (\forall e, e') (\forall s) \text{insert}(e', \text{insert}(e, s)) = \text{insert}(e, \text{insert}(e', s))
\]

\[
@_{\text{om}} (\forall e) (\forall s) \text{insert}(e, \text{insert}(e, s)) = \text{insert}(e, s)
\]

On the other hand, the specification of \( \text{mult} \) in configuration \( \text{oM} \) is introduced as

\[
@_{\text{oM}} (\forall e, e') (\forall s) \neg e = e' \Rightarrow \text{mult}(e, \text{insert}(e', s)) = \text{mult}(e, s)
\]

\[
@_{\text{oM}} (\forall e) \text{mult}(e, \text{empty}) = 0.
\]

Finally, we have to specify the possible reconfigurations. For this, one may use sentences as direct as

\[
@_{\text{om}} (\text{goto}_\text{OM}) \text{OM}
\]

stating that a reconfiguration from \( \text{om} \) to \( \text{OM} \) is possible, or opt for more elaborated forms as \( e.g. \),

\[
(\forall e, e') (\forall s) \text{insert}(e', \text{insert}(e, s)) = \text{insert}(e, \text{insert}(e', s))
\]

\[
\Rightarrow (\text{goto}_\text{OM}) \text{OM}.
\]

The latter states the system can evolve to configuration \( \text{OM} \) (through the event \( \text{goto}_\text{OM} \)) from any other configuration where the order of insertion is irrelevant.

We conclude here the illustration of the specification method extended to accommodate the presence of different interfaces (i.e., algebraic signatures) in different configuration states. Notice, however, that a number of details were not considered here; for example, a definition of the natural numbers and the booleans should be included (and all signatures extended accordingly).

### 7.3 Enabling Reconfigurations

In the specification method introduced in chapter 6 the Kripke structure which captures the reconfiguration dynamics is \( \text{global} \). This means that reconfiguration events, encoded into modalities, are placed at a level of abstraction different from that of configurations themselves.
The approach is, therefore, unsuitable to model situations in which what triggers a reconfiguration (and, therefore, a transition in the underlying Kripke structure) depends on local state values, i.e. it is a function of parameters recorded in the variables of the current configuration.

This entails the need for complementing the definition of the Kripke structure with information about which transitions are enabled or disabled in response to current values of local state variables. Actually, hybrid models are unable to capture directly this type of transitions.

The solution put forward in this section is based on the use of a form of quantification over modalities. To go in this direction, let us suppose for the remaining of the section that we are working into an arbitrary hybridised institution $\mathcal{H}$ with a quantification space which allows expansions of sets of nominals and of modality symbols.

Our standpoint is that the set of enabled transitions, for a given valuation of local state variables, defines a substructure of the Kripke structure. More precisely, the enabling of a (guarded) event denoted by $\lambda$ can be seen as an event (denoted by) $\lambda'$ such that its realisation $R_{\lambda'} \subseteq R_\lambda$ satisfies the guard.

Suppose, for example, one wants to specify a transition identified by $\lambda$ and guarded by a sentence $\rho$ from a state called $i$ into a state called $j$. Assuming that $\lambda'$ is a sub-modality of $\lambda$, i.e., $\lambda'$ is such that

$$(\forall i, j)@i(\lambda')j \Rightarrow @i(\lambda)j,$$

this can be expressed by

$$@i\rho \Rightarrow (\exists \lambda')@i[\lambda']j.$$

Therefore the intended guarded transition can be specified by the sentence

$$@i\rho \Rightarrow [(\exists \lambda')(\forall i, j)(@i(\lambda')j \Rightarrow @i(\lambda)j) \land @i[\lambda']j].$$

Of course, if $\rho$ is a quantified sentence, the particular Kripke substructure corresponding to $\lambda'$ becomes defined on the respective expansions of the model. Each one of these substructures specify an enabling relation.

Abbreviating $\lambda' \subseteq \lambda \equiv (\forall i, j)@i(\lambda')j \Rightarrow @i(\lambda)j$, we write

$$@i\rho \Rightarrow (\exists \lambda')(\lambda' \subseteq \lambda \land @i[\lambda']j).$$
Along the same lines we may express requirements involving patterns like “property $\rho$ enables a reconfiguration into $j$”

$$\rho \Rightarrow (\exists \lambda') (\lambda' \subseteq \lambda \land [\lambda']j),$$

or, more generally, “property $\rho$ enables a move to configurations satisfying property $\rho'$”, by

$$\rho \Rightarrow (\exists \lambda') (\lambda' \subseteq \lambda \land [\lambda']\rho').$$

At first sight, the reader may wonder whether it is suitable to resort to existential quantifications to characterise the semantic model. Actually, the enabling relations, which characterise the Kripke substructures, are not part of the model but just of its extensions. However, the basic idea is well-known in standard algebraic specification: for example, the existence of an inverse for sum in the classical specification of the integers may be expressed as $$(\forall n)(\exists n') n + n' = 0.$$  

An example will illustrate the technique suggested in this section.

**Example 7.3.1** Consider the following requirements:

An ‘adaptable calculator’ offers a single binary operation over the natural numbers, denoted by $\star$. The system has two different modes of execution: a mode `sum` where the $\star$ is interpreted as addition, and a mode `mult` where $\star$ behaves as multiplication. There is also an event `shift` to trigger the reconfiguration of the system from one mode to the other.

$\mathcal{H}FOL$ equipped with the trivial quantification space is a suitable institution for the specification of reconfigurable systems whose modes can be seen as “non-communicating worlds”. Consider then this institution with the signature $(\mathbb{Nat}, \star, \{\text{sum}, \text{mult}\}, \{\text{shift}\})$, where $\mathbb{Nat}$ is the standard one-sorted signature of the natural numbers, with the zero constant and the successor function, enriched with a binary operation $\star$, and `shift` is a unary modality symbol. In such a setting we are able to relate modes through their local properties. For instance,

$$(\forall m : \text{nat}) \star (m, 0) = 0 \Rightarrow [\text{shift}] (\forall m : \text{nat}) \star (m, 0) = m$$

However, in the absence of global quantification (over rigid variables), these properties may not be related “by values”.


On the other hand, a suitable logic to specify systems whose modes are regarded as “communicating worlds” is $\mathcal{H}FOLR'$ with quantifications over rigid variables, i.e., taking as a quantification space, the morphism inclusions

$$\chi : ((S, S_0, F, F_0, P, P_0), \text{Nom}, \Lambda) \hookrightarrow ((S, S_0, F, F_0 + X, P, P_0), \text{Nom}, \Lambda)$$

With such an extension it becomes possible to specify how the system evolves from one mode to another by relating computations between them. For instance, adapting signature $\text{Nat}^\star$ above to make both the sort and the successor function rigid, say into a signature $\text{Nat}R^\star$, we are able to specify multiplication by means of sums:

$$(\forall m, n : \text{nat})(\exists w, z : \text{nat}) @_{\text{mult}}(\star (\text{suc}(n), m) = z \land \star(n, m) = w \land [\text{shift}] \star (w, m) = z)$$

Note that the variable $z$ plays the role of a “communicating bridge” between two different modes, or configurations, of this simple system.

Let us now propose a possible refinement of our running example.

**Example 7.3.2** Consider the following new requirement for the ‘adaptable calculator’:

The calculator’s evolution from one mode to another is triggered by the comparison of the values of two internal variables, say $a$ and $b$. In particular, a reconfiguration to the $\text{mult}$-mode is enabled when $a \geq b$. Otherwise, the only enabled reconfiguration is to the $\text{sum}$-mode.

As mentioned above, enabling transitions are particular sub-relations of the accessibility relation. For instance, a reconfiguration into the $\text{mult}$-mode can be specified by

$$(\forall a, b : \text{nat}) a \geq b \Rightarrow (\exists \text{shift}')(\text{shift}' \subseteq \text{shift} \land [\text{shift}']_{\text{mult}}).$$

Reconfigurations into the $\text{sum}$-mode can be specified, analogously, by

$$(\forall a, b : \text{nat}) a < b \Rightarrow (\exists \text{shift}')(\text{shift}' \subseteq \text{shift} \land [\text{shift}']_{\text{sum}}).$$
For instance, let us consider the hybrid structure \( M = (M, R) \) where
\( M_{\text{mult}} \) and \( M_{\text{sum}} \) are realized by the natural numbers with the usual multiplication and sum operations respectively, \( |R| = \{s_x, s_+\} \), \( R_{\text{mult}} = s_x \) and \( R_{\text{sum}} = s_+ \). Assume \( R_{\text{shift}} \) as the universal relation over \( |R| \).

Consider also a \( \{a, b\} \)-expansion of \( M \), say \( M_{a \geq b} \), such that \( M_{a \geq b} \models a \geq b \), i.e., \( M_a^{\geq b} \supseteq M_b^{a \geq b} \). Finally, consider a shift’-expansion \( M_{a \geq b, \text{shift}' } \) such that \( R_{\text{shift}'} = \{(s_+, s_x), (s_x, s_+)\} \). Since, \( R_{\text{shift}'} \subseteq R_{\text{shift}} \) and \( M_{a \geq b, \text{shift}'} \models [\text{shift}']_{\text{mult}} \), we have

\[
M_{a \geq b} \models (\exists \text{shift}')[\text{shift}' \subseteq \text{shift} \land [\text{shift}']_{\text{mult}}].
\]

Similarly, for \( \{a, b\} \)-expansions where \( M_a^{< b} < M_b^{< b} \), we have

\[
M_{a < b} \models (\exists \text{shift}')[\text{shift} \subseteq \text{shift}' \land [\text{shift}']_{\text{mult}}].
\]

Therefore,

\[
M \models (\forall a, b : \text{nat}) a \geq b \Rightarrow (\exists \text{shift}')[\text{shift} \subseteq [\text{shift}'] \land [\text{shift}']_{\text{mult}}].
\]

Analogously, taking \( R_{\text{shift}'}^{a < b} = \{(s_+, s_+), (s_x, s_+)\} \) we have

\[
M \models (\forall a, b : \text{nat}) a < b \Rightarrow (\exists \text{shift}')[\text{shift} \subseteq \text{shift}' \land [\text{shift}']_{\text{mult}}].
\]

Hence \( M \) is a model for the given specification. \( \circ \)
CONCLUSIONS AND FURTHER WORK

8.1 CONCLUDING

On concluding the thesis it may be appropriate to recall its driving force: the development of foundations and techniques for the formal specification of reconfigurable systems.

The approach proposed was, from the outset, summed up in the motto configurations-as-local-models, reconfigurations-as-transitions. The message was clear: we wanted to understand the dynamics of software reconfiguration in terms of a transition structure whose states, representing the systems’ individual configurations, are endowed with specific, often complex, local specifications. Furthermore, the method was intended to be generic, i.e. independent of whatever logic was found suitable to express the operational requirements of each configuration.

To achieve the envisaged level of generality the whole approach was developed in an institution-independent way. In particular, we defined a method to build institutions which combines a base logic with the hybrid logics features. Along the thesis this was called the hybridisation process. To provide suitable tool support for validating specifications, we proposed a way to ’generate first-order encodings’ for hybridised logics starting from a first-order encoding of the base institution, whenever it exists. Generic notions of bisimulation and refinement for hybrid models were also studied.

This foundational work made possible the development of a rigorous, but flexible, specification method for reconfigurable software. This, again due to the institutional framework adopted, embodies a methodological principle: build the right tool for each job. Or, in other words, tune the method (and the corresponding tools) to whatever logic is found most appropriate for specifying the system’s local layer.

In this context, a few topics for future work are enumerated below.
Further work

Extending the encodings. There is a number of adjustments that can be considered to increase the applicability of the specification method introduced in the thesis. For instance, one could extend the lifting of comorphisms from $\text{I}_2\text{FOL}^{\text{pres}}$ to $\mathcal{H}\text{I}_2\text{FOL}^{\text{pres}}$, into the lifting of comorphisms $\mathcal{H}\text{I}_2\text{CASL}^{\text{pres}}$ to $\mathcal{H}\text{I}_2\text{CASL}^{\text{pres}}$ straightforwardly. This simple extension would lead to a significative enlargement of the class of hybridisations with proof support in HETS (see [Mos02] for an extensive list of institutions encodable in CASL, part of them already integrated into the HETS framework). Furthermore the extension of first order to second order encodings should be considered. Recall, for example, that quantification over modalities, in which a proper treatment of ‘specification enabling’ can be formulated, has second order nature.

Hybridisation for quantitative reasoning. Specification frameworks for quantitative reasoning, dealing for example with weighted or probabilistic transition systems, emerged recently as a main challenge for Software Engineers. This witnesses a shift from classical models of computation, such as labeled transition systems, to similar structures where quantities can be handled. Examples include weighted [DG07], hybrid [Hen96, LSVW95] or probabilistic [Seg95] automata, as well as their coalgebraic rendering (e.g. [Sok11]).

An interesting topic to pursue is taking up this ‘quantitative’ challenge within the context of the hybridisation process itself. The simplest move in such a direction proceeds by instantiation. In this case quantitative reasoning is just reflected and expressed at the local level of concrete, specific configurations. A complementary path may focus on generalising the underlying semantic structures, replacing the REL-component in models by coalgebras over suitable categories of probability distributions, metric, or topological spaces.

Calculus. Comparing the calculus for hybrid propositional logic in reference [Bra10] with the one for hybrid first-order logic in [Bra05], a common structure pops out: both “share” rules involving sentences with nominals and satisfaction operators (i.e., formulas of a “hybrid nature”) and have specific rules to reason about “atomic sentences” that
come from the base institution. Hence, it makes sense to consider the development of a general proof calculus for hybrid institutions built on top of the calculus for the corresponding base institution, in the style of [Bor02, CG08].

**Structured specifications and initial semantics.** Hybridised logics represent a class of institutions. Therefore, all mechanisms used in *abstract specification theory* (e.g. [Tar03, Dc11]), developed for arbitrary institutions, can and should be applied here. For example, dealing with structured specification procedures is essential to lift our approach to modular specifications.

Specifications used in the thesis typically adopt a loose semantics. Whenever possible, on the other hand, the use of an initial semantics, to construct canonical models, provides all the advantages well motivated in the literature. Revisiting this work with the initial semantics paradigm in mind is an interesting direction to pursue. A recent paper, [Dia13], may shed some light on the necessary discussion.

**Industrial assessment.** Although the industrial context where this thesis was developed provided a number of examples and challenges to the envisaged specification method, a more systematic assessment, ressorting, for example, to real, industrial case studies, remains to be done.
Formal specification and development of complex systems has been a major issue in the author’s research. This dates back to his first academic dissertation, in the context of a MSc degree in Mathematics, focused on *hidden algebra* and *observational equality* [Mad08]. Later it was again a major motivation to pursue a PhD project blending Mathematics and Computer Science.

The project was partially supported by an industrial grant offered by *Critical Software*, a Portuguese, leading IT company devoted to the development of safety and mission-critical software. This provided an interesting context for the thesis, contributing for a deeper understanding of the challenges and needs inherent to the rigorous design of reliable systems. In particular, it was determinant in the choice of reconfigurable systems as a technical target for the thesis and an engineering challenge to the author’s mathematical background. Later, the author became part of a team who launched a new spin-off company directed to formal software development.

The author’s interest on the theory of institutions goes back to a visit of Răzvan Diaconescu, his external supervisor, to Portugal in June of 2010, as an advisor for the *Mondrian* project. In particular, the work on the institution-independent account of hybridisation that underlies the thesis started during this visit. This was also the kick-off for the line of research documented here.

However, as often happens in a long term exploratory project, a number of other research directions were explored and omitted from the final version of the thesis, because they fall out of its scope. This final remark closes pointing out two main branches crossed along the way:

- The first was the work on an alternative notion of refinement of algebraic specifications taking as witnesses *logical interpretations* rather than the more usual (surjective) homomorphisms.
First formalised for specifications in equational logic [MMB09b], the method was generalised to k-deductive systems in [MMB09a], and further to Π-institutions [RMMB11]. The study of the correspondence between logical interpretations and morphisms of a certain kind of coalgebras, so that usual coalgebraic constructions, such as simulations and bisimulations, could be used to explore interpretations between (abstract) logics, was worked out and published in [MMB13b].

- A second branch of research, although of a preliminary nature, was an initial attempt to explore dialgebras [PZ01, You10] as a suitable model for reconfigurable systems. A few initial ideas were presented in [MMB11].


[BBG⁺08] Roberto Bruni, Antonio Bucchiarone, Stefania Gnesi, Dan Hirsch, and Alberto Lluch-Lafuente. Graph-based design and analysis of dynamic software architectures.


Maximilians-Universität München, Institut für Informatik, November 1998.


[HBB08] Jens Hansen, Thomas Bolander, and Torben Braüner. Many-valued hybrid logic. In C. Areces and R. Goldblatt, editors, Advances in Modal Logic (7th AiML,
BIBLIOGRAPHY


[Mos03] Till Mossakowski. Foundations of heterogeneous specification. In M. Wirsing, D. Pattinson, and R. Hen-


