Foundations and techniques for software reconfigurability

Alexandre Leite de Castro Madeira

UMinho | 2013

Universidade do Minho
Escola de Engenharia

Alexandre Leite de Castro Madeira

Foundations and techniques for software reconfigurability

Programa de Doutoramento em Informática das Universidades do Minho, de Aveiro e do Porto

universidade de aveiro

Universidade do Minho

U.PORTO

June 2013

Universidade do Minho
Escola de Engenharia

Alexandre Leite de Castro Madeira

Foundations and techniques for software reconfigurability

Programa de Doutoramento em Informática das Universidades do Minho, de Aveiro e do Porto

universidade de aveiro

Universidade do Minho

U.PORTO

This work was realized under supervision of:
Luís Soares Barbosa
and
Manuel A. Maritns

June 2013

This dissertation was supported by the Fundação para a Ciência e Tecnologia (FCT) and Critical Software S.A., under BDE grant under the contract SFRH/BDE/33650/2009 and by the MONDRIAN Project (FCT) under the contract PTDC/EIA-CCO/108302/2008.

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 focus 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. These specifications are modeled by 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-structutres*, or *probabilistic spaces* or even in *configurations-as-Kripke-structutres*, 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 J. Goguen and R. 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.

RESUMO

O termo *reconfigurável* é usado para sistemas de software que se comportam de forma diferente em diferentes modos de operação (frequentemente chamados de configurações) comutando entre eles, ao longo do seu ciclo de vida. Estes sistemas, que evoluem em resposta a estímulos externos e internos, estão por toda a parte, desde sistemas de *e-Health* ou sistemas integrados de *e-Governement*, às redes de sensores, das aplicações domésticas aos complexos sistemas distribuidos, dos sistemas críticos de missão ao software de computação paralela.

Existem duas abordagens formais para captar requisitos deste tipo de sistemas: uma focada no comportamento e evolução; e outra focada nos dados e respectivas transformações. Segundo o primeiro paradigma, os sistemas reconfiguráveis são abordados por (alguma variante) de máquinas-de-estados, correspondendo, cada um dos seus estados, a uma configuração que o sistema possa assumir. A outra abordagem, orientada aos dados, especifica as funcionalidades do sistema em função de relações de input-output, que modelam operações nos dados. Uma especificação apresenta uma teoria numa lógica adequada, expressa sobre uma assinatura que capta a sua interface sintática. A sua semântica consiste na classe de álgebras, ou estruturas de primeira ordem, que modelam a teoria especificada.

A observação de que, a cada momento, os serviços oferecidos por um sistema reconfigurável possam depender do estado da sua evolução, sugere-nos que ambas as dimensões (dados e comportamento) estejam interligados e devam ser combinados. Em particular, cada nó do sistema de transição, que descreve o espaço de reconfigurabilidade, pode ser dotado de uma estrutura local onde as funcionalidades do sistema, na respectiva configuração, são modeladas. Esta é a ideia base da especificação *configurações-como-modelos-locais*. Tecnicamente, as especificações são modeladas por máquinas de estados estruturadas, onde cada estado denota uma estrutura complexa, ao invés de um conjunto.

Uma especificação para este tipo de sistemas deve ser adequada à expressão de asserções acerca da dinâmica de transições, assim como, ao nível local de cada configuração particular. Isto leva-nos à adopção de *lógica híbrida*, que adiciona, mecanismos para referir estados específicos à expressividade modal dos sistemas de transição, como *lingua franca* para um método adequado de especificação.

Por outro lado, aplicações podem requerer lógicas específicas para descrever as suas configurações. Por exemplo, requisitos expressos por equações devem ser modelados numa perspectiva *configurações-como-álgebras*. Dependendo da sua natureza, podemos considerar *configurações-como-estruturas de primeira ordem*, ou *configurações-como-espaços probabilísticos* ou mesmo

configurações-como-estruturas de Kripke quando usadas, localmente, lógica de primeira ordem, lógica fuzzy, ou lógica modal respectivamente.

O objectivo da tese é desenvolver os fundamentos para um método de especificação baseado nestes princípios. Por forma a acomodar todas estas possibilidades, a abordagem é desenvolvida sob fundamentos muito genéricos. Ao invés de comprometer a abordagem com uma lógica híbrida particular, partimos da escolha da lógica específica para especificar requisitos ao nível (estáctico) local. Esta lógica é então tomada como lógica de *base*, sobre a qual os mecanismos da lógica híbrida, tanto ao nível sintáctico (i.e., modalidades, nominais, etc.) como ao semântico (i.e., mundos possíveis), são desenvolvidos. Este processo, que chamamos de *hibridização,* é uma das principais contribuições técnicas da tese. A generalidade do método resulta do seu desenvolvimento no contexto da *teoria das instituições* de J. Goguen e R. Burstall. As peincipais contribuições da tese são:

- um método para hibridizar instituições arbitrárias; o que pode ser entendido como uma fonte de lógicas para suportar especificações configurações-como-modelos-locais arbitrárias
- um método para transportar codificações de uma instituição nas apresentações de primeira ordem (tecnicamente comorfismos), em codificações da sua hibridização em apresentações em primeira ordem; este resultado abre o caminho para a introdução do suporte de prova automático para uma ampla classe de lógicas híbridas;
- caracterização de relações de bissimulação e de refinamento para modelos de hibridizações genéricas. Isto oferece relações canónicas, que preservam satisfação, para identificar e relacionar modelos;
- um método de especificação para sistemas reconfiguráveis com dois estágios, baseado numa estrutura de transição *global*, onde o espaço de reconfigurações do sistema é modelado; e numa especificação *local* das configurações expressa numa lógica escolhida como adequada, aos requisitos a tratar;
- um conjunto de técnicas adicionais para assistir o processo de especificação e de verificação de requisitos de sistemas reconfiguráveis com suporte parcial de ferramentas.

ACKNOWLEDGMENTS

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 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. Messenguer 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 $\mathcal{HI}$-signatures 44
3.1.2 $\mathcal{HI}$-sentences fuctor 45
3.1.3 $\mathcal{HI}$-models functor 48
3.1.4 The satisfaction relation 50
3.1.5 Base logic versus its hybridisation 54
3.2 Examples 56
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 82
4.3 Conservativity 89
5 BISIMULATION AND REFINEMENT NOTIONS FOR HYBRIDISED LOGICS 95
5.1 Bisimulation for hybridised Logics 96
5.2 Refinements for generic hybridised logics 101

II TECHNIQUES 107
6 HYBRIDISATION FOR THE WORKING SOFTWARE ENGINEER 109
6.1 Motivation 109
6.2 The approach 111
6.3 Dealing with heterogenous requirements 124
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

LIST OF FIGURES

Figure 1 Proof Scheme 54
Figure 2 Refinement in $\mathcal{H}MVL_{\mathrm{L}}$. 104
Figure 3 Refinement in $\mathcal{H}EQ$. 105
Figure 4 Specification of reconfigurable systems: The approach. 112
Figure 5 Specification of reconfigurable systems: The method. 113
Figure 6 A plastic buffer. 114
Figure 7 Plastic Buffer *PAR* signature. 118
Figure 8 Plastic Buffer $\mathcal{H}PAR$ Specification. 120
Figure 9 The 'plastic buffer' specification translated into *FOL*. 125
Figure 10 A $\mathcal{H}TRM$ specification. 126
Figure 11 Hierarchical representation of the pump boot. 128
Figure 12 A specification in $\mathcal{H}^2PL$. 129
Figure 13 The reconfigurable calculator in $\mathcal{H}CASL$. 135
Figure 14 A specification parsed by the generic engine. 135

1 INTRODUCTION

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 [DM13].

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, DM13, 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

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], μ-*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 latter 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 latter. 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* [BKI05], *quantic logics* [CMSS06], *hiding and observational logics* [BD94, BH06, Mar06], *coalgebraic logics* [Cô6], *functional* [ST12, SM09] and *imperative programing 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 the each specification logic (e.g. [Mos04, Tar03, DT11]). 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 another [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 $\Box$ and $\Diamond$ 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 date 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, Vou10] 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 specification 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 *lax coalgebra* [CGRH01] which incorporates algebraic structure in both labels and states.

There are two ways to mathematically represent transition systems: either as *graphs* or as *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[Roz97], notably the double pushout approach [HMP01], have been extensibly used in modelling the evolution of dynamic systems. Typical applications emerge in the areas of mobile processes, from the π-calculus [MPW92] and its variants [SW03] to the latter work of Robin Milner in bigraphs [CMS07, Mil09], architectural evolution [WF02, BBG+08] or coordination of software services [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 [BFH+12], the use of feature enriched transition systems [CCS+13] or well-structured transition systems [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. [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 coalge-

bra, 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], emboding 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 includes 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

2

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 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

$$\rho := \mathsf{Prop} \mid \neg\rho \mid \Box\rho \mid \Diamond\rho \mid \rho \otimes \rho$$

where $\otimes \in \{\vee, \wedge, \Rightarrow\}$. 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 $M = (M_w : \mathsf{Prop} \to \{\top, \bot\})_{w \in |W|}$ 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., $f : \mathsf{Prop} \to \mathcal{P}(|W|)$ (cf. [Bla00]). Of course both definitions are equivalent making, $f(p) = \{w \in |W| | M_w(p) = \top\}$.

The satisfaction relation $\models^{MPL}_{\mathsf{Prop}}$ is inductively defined as follows:

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

where

- $(M, W) \models^{w} p$ iff $M_w(p) = \top$;
- $(M, W) \models^{w} \rho \vee \rho'$ iff $(M, W) \models^{w} \rho$ or $(M, W) \models^{w} \rho'$ and similarly for the connectives $\{\wedge, \Rightarrow\}$;
- $(M, W) \models^{w} \neg\rho$ iff $(M, W) \not\models^{w} \rho$;
- $(M, W) \models^{w} \Diamond\rho$ iff there is some $w' \in |W|$ such that $(w, w') \in W_R$ 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 latter case, signatures are pairs (Prop, Λ) where Λ 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 $i \in \mathrm{Nom}$, state the validity of ρ at the world named by i. Hence, the (propositional) hybrid logic sentences are defined by the grammar:

$$\rho := \mathrm{Prop} \mid \mathrm{Nom} \mid @_i\rho \mid \neg\rho \mid \Box\rho \mid \Diamond\rho \mid \rho \otimes \rho$$

$i \in \mathrm{Nom}$ and $\otimes \in \{\vee, \wedge, \Rightarrow\}$.

The models (M, W) are defined as in the standard modal case, with nominals interpreted as constants over W. The satisfaction relation $\models^{\mathcal{H}PL}_{(\mathrm{Prop},\mathrm{Nom})}$ extends $\models^{MPL}_{\mathrm{Prop}}$ 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 (using $i \Rightarrow \neg\Box i$). 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

– $(M, W) \models_{(\mathrm{Prop},\mathrm{Nom})} (\forall i)\rho$ iff for any $\{i\}$-expansion (M', W') of (M, W), $(M', W') \models_{(\mathrm{Prop},\mathrm{Nom}+\{i\})} \rho$

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

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 ϵ 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 ϵ. 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 bellow some basic definitions [Lan98, AHS90, Awo06].

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

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

$$\mathbb{C}(A, B) = \{f \in \mathbb{C} | \mathrm{dom}(f) = A \textit{ and } \mathrm{cod}(f) = B\}$$

is a set,

- *for any objects* $A, B, C \in \mathbb{C}$, *a* composition *map*

$$\_;\_ : \mathbb{C}(A, B) \times \mathbb{C}(B, C) \to \mathbb{C}(A, C),$$

such that the following diagram commutes

$$\begin{tikzcd} A \arrow[r, "f"] \arrow[dr, "f;g"'] & B \arrow[d, "g"] \arrow[dr, "g;h"] & \\ & C \arrow[r, "h"'] & D \end{tikzcd}$$

- *an* identity *map* $1 : |\mathbb{C}| \to \mathbb{C}$ *such that for each* $A \in \mathbb{C}$, $1_A \in \mathbb{C}(A, A)$, *the following diagram commutes*

$$\begin{tikzcd} A \arrow[r, "f"] \arrow[d, "1_A"'] \arrow[dr, "f"] & B \arrow[d, "1_B"] \\ A \arrow[r, "f"] & B \end{tikzcd}$$

The category $\mathbb{S}\mathrm{et}$ takes as objects sets and as arrows functions with the usual functional composition. The category [1] $\mathbb{C}\mathrm{AT}$, known as the *category of categories*, takes as objects categories and, as arrows, functors between them:

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

1. *associates objects to objects* $|F| : |\mathbb{A}| \to |\mathbb{B}|$, *and*
2. *arrows to arrows,* $F_{A,B} : \mathbb{A}(A, B) \to \mathbb{B}(F(A), F(B))$ *such that*
   - $F(1_A) = 1_{FA}$,
   - $F(f; g) = Ff; Fg$.

We say that a category $\mathbb{A}$ is a subcategory of $\mathbb{B}$, in symbols $\mathbb{A} \subseteq \mathbb{B}$, when

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

1 Strictly speaking, this is only a 'quasi-category' living in a higher set-theoretic universe (cf. [Lan98]);

- the composition is $\mathbb{A}$ is the restriction of the composition in $\mathbb{B}$ to $|\mathbb{A}|$.

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

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

$$\begin{tikzcd} F(A) \arrow[r, "\tau(A)"] \arrow[d, "Ff"'] & G(A) \arrow[d, "Gf"] \\ F(B) \arrow[r, "\tau(B)"'] & G(B) \end{tikzcd} \tag{1}$$

We write $\begin{tikzcd} \mathbb{A} \arrow[r, bend left, "F"] \arrow[r, bend right, "G"'] \arrow[r, phantom, "\Downarrow\tau"] & \mathbb{B} \end{tikzcd}$, *or simply,* $\tau : F \Rightarrow G$ *to say that* τ *is a natural trasformation between* F *and* G.

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

- *the diagram* $\begin{tikzcd} A \arrow[r, "q"] \arrow[d, "p"'] & Y \arrow[d, "g"] \\ X \arrow[r, "f"'] & B. \end{tikzcd}$ *commutes, and*

- *for any other commuting square* $\begin{tikzcd} Z \arrow[r, "s"] \arrow[d, "r"'] & Y \arrow[d, "g"] \\ X \arrow[r, "f"'] & B. \end{tikzcd}$ *, there is a unique morphism* $h : Z \to A$ *making the following diagram to comute*

$$\begin{tikzcd} Z \arrow[dr, dotted, "h"] \arrow[drr, bend left, "s"] \arrow[ddr, bend right, "r"'] & & \\ & A \arrow[r, "q"] \arrow[d, "p"'] & Y \arrow[d, "g"] \\ & X \arrow[r, "f"'] & B. \end{tikzcd}$$

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

- *the diagram* $\begin{tikzcd} B \arrow[r, "g"] \arrow[d, "f"'] & Y \arrow[d, "s"] \\ X \arrow[r, "t"'] & A \end{tikzcd}$ *commutes and,*

- *for any other commutative diagram* $\begin{tikzcd} B \arrow[r, "g"] \arrow[d, "f"'] & Y \arrow[d, "u"] \\ X \arrow[r, "v"'] & Z \end{tikzcd}$ *, there is a unique arrow* $h : A \to Z$ *commuting the following diagram*

$$\begin{tikzcd} B \arrow[r, "g"] \arrow[d, "f"'] & Y \arrow[d, "s"] \arrow[ddr, bend left, "u"] & \\ X \arrow[r, "t"'] \arrow[drr, bend right, "v"'] & A \arrow[dr, dotted, "h"] & \\ & & Z. \end{tikzcd}$$

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 $\mathbf{I}$ in $\mathbb{A}$ consists of a functor $D : \mathbf{I} \to \mathbb{A}$. Category $\mathbf{I}$ is called *index of the diagram* and D_i used to denote the image $D(i)$, for $i \in \mathbf{I}$. A *cone for a diagram* D consists of an object $A \in \mathbb{A}$ to-

gether with a family of arrows $\big(C_i : A \to D_i\big)_{i \in \mathbf{I}}$ such that, for any $f \in \mathbf{I}(i,j)$, the triangle

$$\begin{tikzcd} & A \arrow[dl, "C_i"'] \arrow[dr, "C_j"] & \\ D_i \arrow[rr, "Cf"'] & & D_j \end{tikzcd}$$

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

$$\begin{tikzcd} D_i \arrow[rr, "Cf"] \arrow[dr, "C_i"'] & & D_j \arrow[dl, "C_j"] \\ & A & \end{tikzcd}$$

commutes.

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

$$\begin{tikzcd} A' \arrow[rr, dotted, "Z"] \arrow[dr, "C'_i"'] & & A \arrow[dl, "C_i"] \\ & D_i & \end{tikzcd}$$

commutes, for any $i \in |\boldsymbol{I}|$.
A colimit *for a diagram* $D : \boldsymbol{I} \to \mathbb{A}$*, or a* D*-colimit for short, consists of a cocone* $\big(C_i : D_i \to A\big)_{i \in |\boldsymbol{I}|}$ *such that, for any other cocone* $\big(C'_i : D_i \to A'\big)_{i \in |\boldsymbol{I}|}$*, there is an unique arrow* $Z \in \mathbb{A}(A, A')$ *such that the diagram*

$$\begin{tikzcd} & D_i \arrow[dl, "C'_i"'] \arrow[dr, "C_i"] & \\ A' & & A \arrow[ll, dotted, "Z"] \end{tikzcd}$$

commutes, for any $i \in |\boldsymbol{I}|$.

Finally we observe that a pullback in a category $\mathbb{A}$ is a limit for a diagram $D : \mathbf{I} \to \mathbb{A}$ when $\mathbf{I}$ is the category completely defined by the objects and arrows $i \longrightarrow j \longleftarrow k$. Analogously, a pushout in $\mathbb{A}$

can be seen as a colimit of a diagram $D : \mathbf{I} \to \mathbb{A}$ for $\mathbf{I}$ the category completely defined by $i \longleftarrow j \longrightarrow k$. Moreover, final and initial objects can also be seen respectively as the limit and the colimit for $D : \mathbf{I} \to \mathbb{A}$, where $\mathbf{I}$ is the empty category, i.e., the category with an empty set of objects.

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

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

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

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

$$\mathcal{I} = \left(\mathrm{Sign}^{\mathcal{I}}, \mathrm{Sen}^{\mathcal{I}}, \mathrm{Mod}^{\mathcal{I}}, (\models^{\mathcal{I}}_{\Sigma})_{\Sigma \in |\mathrm{Sign}^{\mathcal{I}}|}\right)$$

consists of

1. *a category* $\mathrm{Sign}^{\mathcal{I}}$ *whose objects are called* signatures*,*
2. *a functor* $\mathrm{Sen}^{\mathcal{I}} : \mathrm{Sign}^{\mathcal{I}} \to \mathbb{S}\mathrm{et}$ *giving for each signature the set od* sentences *over that signature,*
3. *a functor* $\mathrm{Mod}^{\mathcal{I}} : (\mathrm{Sign}^{\mathcal{I}})^{\mathrm{op}} \to \mathbb{C}\mathrm{AT}$*, giving for each signature* Σ *the category of* Σ-models*, and the respective* Σ-(models) homomorphisms*, and*
4. *a relation* $\models^{\mathcal{I}}_{\Sigma} \subseteq |\mathrm{Mod}^{\mathcal{I}}(\Sigma)| \times \mathrm{Sen}^{\mathcal{I}}(\Sigma)$ *for each* $\Sigma \in |\mathrm{Sign}^{\mathcal{I}}|$*, called the* satisfaction relation*,*

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

$$M' \models^{\mathcal{I}}_{\Sigma'} \mathrm{Sen}^{\mathcal{I}}(\varphi)(\rho) \ \textit{iff}\ \mathrm{Mod}^{\mathcal{I}}(\varphi)(M') \models^{\mathcal{I}}_{\Sigma} \rho \tag{2}$$

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

$$\begin{tikzcd}
\Sigma \arrow[d, "\varphi"'] & \mathrm{Mod}^{\mathcal{I}}(\Sigma) \arrow[r, dash, "\models^{\mathcal{I}}_{\Sigma}"] & \mathrm{Sen}^{\mathcal{I}}(\Sigma) \arrow[d, "\mathrm{Sen}^{\mathcal{I}}(\varphi)"] \\
\Sigma' & \mathrm{Mod}^{\mathcal{I}}(\Sigma') \arrow[u, "\mathrm{Mod}^{\mathcal{I}}(\varphi)"] \arrow[r, dash, "\models^{\mathcal{I}}_{\Sigma'}"'] & \mathrm{Sen}^{\mathcal{I}}(\Sigma')
\end{tikzcd}$$

As usual, $\mathcal{M}^*$ denotes the set of sentences $\{\rho \in \mathrm{Sen}(\Sigma) \mid M \models_{\Sigma} \rho\}$ and for any $\Gamma \subseteq \mathrm{Sen}(\Sigma)$, Γ^* denotes the class of models $\{M \in |\mathrm{Mod}(\Sigma)| \mid M \models_{\Sigma} \rho\}$ 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

$$\mathrm{Mod}^{\mathcal{I}} : (\mathrm{Sign}^{\mathcal{I}}) \to \mathbb{C}\mathrm{AT}^{\mathrm{op}}$$

(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

$$\mathrm{Mod}^{\mathcal{I}} : (\mathrm{Sign}^{\mathcal{I}}) \to \mathbf{Set}^{\mathrm{op}}.$$

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

○

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

ignatures are triples (S, F, P) consisting of

- a set of sort symbols S,
- a family $F = \{F_{\underline{ar} \to s} \mid \underline{ar} \in S^*, s \in S\}$ of sets of function symbols indexed by arities $\underline{ar}$ (for the arguments) and sorts s (for the results), and
- a family $P = \{P_{\underline{ar}} \mid \underline{ar} \in S^*\}$ 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 $\varphi^{st} : S \to S'$,

- a family of functions $\varphi^{op} = \{\varphi^{op}_{\underline{ar}\to s} : F_{\underline{ar}\to s} \to F'_{\varphi^{st}(\underline{ar})\to\varphi^{st}(s)} \mid \underline{ar} \in S^*, s \in S\}$, and
- a family of functions $\varphi^{rl} = \{\varphi^{rl}_{\underline{ar}\to s} : P_{\underline{ar}} \to P'_{\varphi^{st}(\underline{ar})} \mid \underline{ar} \in S^*, s \in S\}$.

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 σ as a function M_σ from the product of the interpretations of the argument sorts to the interpretation of the target sort, and each relation symbol π as a subset M_π 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., $h_s(M_\sigma(m)) = M'_\sigma(h_{\underline{ar}}(m))$ for each $\sigma \in F_{\underline{ar}\to s}$ and each $m \in M_{\underline{ar}}$, and
- $h_{\underline{ar}}(m) \in M'_\pi$ if $m \in M_\pi$ (i.e. $h_{\underline{ar}}(M_\pi) \subseteq M'_\pi$) for each relation $\pi \in P_{\underline{ar}}$ and each $m \in M_{\underline{ar}}$.

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

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

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, Sen^{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 $(F + X)_{\underline{ar} \to s} = F_{\underline{ar} \to s}$ when $\underline{ar}$ is non-empty and $(F + X)_{\to s} = F_{\to s} \cup \{(x, s, (S, F, P)) \mid (x, s, (S, F, P)) \in X\}$ and we let $\varphi' : (S, F + X, P) \to (S', F' + X^{\varphi}, P')$ be the canonical extension of φ 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 $\varphi : (S, F, P) \to (S', F', P')$, i.e., $\mathrm{Sen}^{FOL}(\varphi) : \mathrm{Sen}^{FOL}(S, F, P) \to \mathrm{Sen}^{FOL}(S', F', P')$, replaces symbols of (S, F, P) by the respective φ-images in (S', F', P'). More precisely, $\varphi^{\mathrm{trm}} : T_{(S,F)} \to T_{(S',F')}$ is defined by

$$\varphi^{\mathrm{trm}}(f(t_1, \ldots, t_n)) = \varphi^{\mathrm{op}}(f)(\varphi^{\mathrm{trm}}(t_1), \ldots, \varphi^{\mathrm{trm}}(t_n)).$$

Then,

- $\mathrm{Sen}^{FOL}(\varphi)(t \approx t') = \varphi^{\mathrm{trm}}(t) \approx \varphi^{\mathrm{trm}}(t')$;
- $\mathrm{Sen}^{FOL}(\varphi)(\pi(t_1, \ldots, t_n)) = \varphi\mathrm{rl}(\pi)(\varphi^{\mathrm{trm}}(t_1), \ldots, \varphi^{\mathrm{trm}}(t_n))$;
- $\mathrm{Sen}^{FOL}(\varphi)(\neg\rho) = \neg\mathrm{Sen}^{FOL}(\varphi)(\rho)$;
- $\mathrm{Sen}^{FOL}(\varphi)(\rho \odot \rho') = \mathrm{Sen}^{FOL}(\varphi)(\rho) \odot \mathrm{Sen}^{FOL}(\varphi)(\rho')$, where $\odot \in \{\vee, \wedge, \to\}$;
- $\mathrm{Sen}^{FOL}(\varphi)(\forall X\, \rho) = \forall X^{\varphi}\, \mathrm{Sen}^{FOL}(\varphi')(\rho)$, where $X^{\varphi} = \{(x, \varphi^{\mathrm{st}}(s), (S', F', P')) | (x, s, (S, F, P))\}$, and φ' canonically extends φ by mapping each $(x, s, (S, F, P))$ to $(x, \varphi^{\mathrm{st}}(s), (S', F', P'))$.

$$\begin{tikzcd}
(S, F, P) \arrow[r, "\varphi"] \arrow[d] & (S, F', P') \arrow[d] \\
(S, F \uplus X, P) \arrow[r, "\varphi'"'] & (S, F' \uplus X^{\varphi}, P')
\end{tikzcd}$$

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 $\pi(t_1, \ldots, t_n)$.
- $M \models_{(S,F,P)} \rho_1 \wedge \rho_2$ 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 $\exists$.

$\circ$

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 $(\forall X)\, t = t'$. $\circ$

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. $\circ$

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 $(\emptyset, \emptyset, \mathsf{Prop})$, where Prop is a set of (0-ary predicate) symbols called propositions. We denote propositional signatures $(\emptyset, \emptyset, \mathsf{Prop})$ simply by Prop. The signature morphisms are functions between sets of symbols and hence, Sign^{PL} coincides with $\mathbb{S}\mathrm{et}$. Models of Prop are sets X such that $X \subseteq \mathsf{Prop}$ or, equivalently, functions $M : \mathsf{Prop} \to \{\top, \bot\}$. Sentences are the usual propositional logic formulæ

$$\rho := \mathsf{Prop} \mid \rho \vee \rho \mid \rho \wedge \rho \mid \rho \Rightarrow \rho \mid \neg\rho$$

The satisfaction relation is defined as usual:

- $M \models^{PL}_{\mathsf{Prop}} p$ iff $M(p) = \top$, for any $p \in \mathsf{Prop}$;
- $M \models^{PL}_{\mathsf{Prop}} \rho \vee \rho'$ iff, $M \models^{PL}_{\mathsf{Prop}} \rho$ or $M \models^{PL}_{\mathsf{Prop}} \rho'$,

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 $(S, \mathsf{TF}, \mathsf{PF})$, where TF is a family of sets of *total* function symbols and PF is a family of sets of *partial* function symbols such that $\mathsf{TF}_{\underline{ar}\to s} \cap \mathsf{PF}_{\underline{ar}\to s} = \emptyset$ for each arity $\underline{ar}$ 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 PF are interpreted as partial, rather than total, functions. For any $\sigma \in \mathsf{PF}_{\underline{ar}\to s}$, $\mathsf{dom}(A_\sigma) = \{a \in A_{\underline{ar}} \mid A_\sigma(a) \text{ defined}\}$ is the domain of the operation σ. A *partial algebra homomorphism* $h : A \to B$ is a family of (total) functions $\{h_s : A_s \to B_s \mid s \in S\}$, indexed by the set of sorts S in the signature, such that $h_s(A_\sigma(a)) = B_\sigma(h_{\underline{ar}}(a))$ for each function symbol $\sigma \in \mathsf{TF}_{\underline{ar}\to s} \cup \mathsf{PF}_{\underline{ar}\to s}$ and each tuple of arguments $a \in A_{\underline{ar}}$ for which $A_\sigma(a)$ is defined.

Sentences have three kinds of atoms: *definedness* $\mathsf{df}(t)$, *strong* equality $t = t'$, and *existential* equality $t \stackrel{e}{=} t'$. 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:

- $A \models^{PA}_{(S,TF,PT)} \mathsf{df}(t)$ iff A_t is defined;
- $A \models^{PA}_{(S,TF,PT)} t = t'$ iff either both are defined and $A_t = At'$ or both are undefined;
- $A \models^{PA}_{(S,TF,PT)} t \stackrel{e}{=} t'$ 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 $\mathsf{df}(t)$

is equivalent to $t \overset{e}{=} t$ and that $t = t'$ is equivalent to $(t \overset{e}{=} t') \vee (\neg \mathsf{df}(t) \wedge \neg \mathsf{df}(t'))$.

$\circ$

Example 2.3.8 (Multi-valued and Fuzzy Logics) Multi-valued logics replace the two-elements set of true values $\{\mathtt{true}, \mathtt{false}\}$, 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 bellow.

A *residuated lattice* is a structure $L = (\mathbf{L}, \leqslant, \wedge, \vee, \top, \bot, \star)$, where

- $(\mathbf{L}, \wedge, \vee, \top, \bot)$ is a lattice ordered by $\leqslant$, with support $\mathbf{L}$, with (binary) infimum and supremum, $\wedge$ and $\vee$, and bigest and smallest elements, $\top$ and $\bot$;
- $\star$ is an associative and commutative binary operation such that, for any elements $x, y, z \in L$:
  - $x \star \top = \top \star x = x$;
  - $y \leqslant z$ implies that $(x \star y) \leqslant (x \star z)$;
  - there exists an element $x \Rightarrow z$ such that
  $$y \leqslant (x \Rightarrow z) \text{ iff } x \star y \leqslant z.$$

  The residuated lattice L is complete if any subset $S \subseteq \mathbf{L}$ has infimum and supremum denoted by $\bigwedge S$ and $\bigvee S$, respectively.

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

- $\mathrm{Sign}^{MVL_L} = \mathrm{Sign}^{FOL}$;
- Sentences of $\mathrm{Sen}^{MVL_L}(S, F, P)$ are pairs (ρ, p) where
  - p is an element of L, and
  - ρ is a 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 $\{\Rightarrow \wedge, \vee, \top, \bot, \star\}$ and quantifications $(\forall X)$ and $(\exists X)$ for X a finite set of variables.

- a model $M \in |\mathrm{Mod}^{MVL_L}(S, F, P)|$ consists
  - of an algebra (S, F),
  - for each $\pi \in P_{\underline{ar}}$, of a function $M_\pi : M_{\underline{ar}} \to \mathbf{L}$.

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

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

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

  where $M \models \rho$ is inductively defined as follows:
  - for any relational atom $\pi(t_1, \ldots, t_n)$, $(M \models \pi(t_1, \ldots, t_n)) = M_\pi(M_{t_1}, \ldots, M_{t_n})$;
  - $(M \models \top) = \top$;
  - $(M \models \bot) = \bot$;
  - $(M \models \rho_1 \odot \rho_2) = (M \models \rho_1) \odot (M \models \rho_2)$, for $\odot \in \{\wedge, \vee, \Rightarrow, \star\}$;
  - $(M \models (\forall X)\rho) = \bigwedge \{M' \models \rho | M'\restriction_{(S,F,P)} = M\}$;
  - $(M \models (\exists X)\rho) = \bigvee \{M' \models \rho | M'\restriction_{(S,F,P)} = M\}$.

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 $\wedge$ 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 Σ and any Σ-sentences e_1*

and e_2 *there exists a* Σ*-sentence* e *such that* $e^* = e_1^* \cap e_2^*$. *Usually* e *is denoted by* $e_1 \wedge e_2$.

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

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

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

Note, for instance that *FOL* has all of the mentioned semantical connectives and quantifications, *PL* all the connectives. Moreover, MVL_L has and *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*

$$\begin{tikzcd} A & A_1 \arrow[l, "F_1"'] \\ A_2 \arrow[u, "F_2"] & A' \arrow[l, "G_2"] \arrow[u, "G_1"'] \end{tikzcd} \tag{3}$$

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 $\mathrm{Mod}: \mathrm{Sign}^{\mathrm{op}} \to \mathbf{CAT}$ *a commuting square of signature morphisms*

$$\begin{tikzcd} \Sigma \arrow[r, "\varphi_1"] \arrow[d, "\varphi_2"'] & \Sigma_1 \arrow[d, "\theta_1"] \\ \Sigma_2 \arrow[r, "\theta_2"'] & \Sigma' \end{tikzcd} \tag{4}$$

is a (weak) amalgamation square for Mod *when*

$$\begin{tikzcd} \mathrm{Mod}(\Sigma) & \mathrm{Mod}(\Sigma_1) \arrow[l, "\mathrm{Mod}(\varphi_1)"'] \\ \mathrm{Mod}(\Sigma_2) \arrow[u, "\mathrm{Mod}(\varphi_2)"] & \mathrm{Mod}(\Sigma') \arrow[l, "\mathrm{Mod}(\theta_2)"] \arrow[u, "\mathrm{Mod}(\theta_1)"'] \end{tikzcd} \tag{5}$$

is a (weak) amalgamation square.

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

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* $\mathrm{Mod}' \subseteq \mathrm{Mod}: \mathrm{Sign}^{\mathrm{op}} \to \mathbb{C}\mathrm{AT}$ reflects (weak) amalgamation *when each pushout square in* Sign *that is a (weak) amalgamation square for* Mod *is a (weak) amalgamation square for* Mod' *too.*

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

$$\begin{tikzcd} \mathrm{Mod}_1(\Sigma) & \mathrm{Mod}_2(\Sigma) \arrow[l, "\beta_\Sigma"'] \\ \mathrm{Mod}_1(\Sigma') \arrow[u, "\mathrm{Mod}_1(\chi)"] & \mathrm{Mod}_2(\Sigma') \arrow[l, "\beta_{\Sigma'}"] \arrow[u, "\mathrm{Mod}_2(\chi)"'] \end{tikzcd}$$

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 β 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, DM13], 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 *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

- $M \models^{FOL}_{(S,F,P)} (\forall X)\rho$ when $M' \models^{FOL}_{(S,F+X,P)} \rho$ for any $(S, F+X, P)$-expansion M' 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 $\mathcal{I}$ could be formalised as

- $M \models^{\mathcal{I}}_{\Sigma} (\forall\chi)\rho$ when $M' \models^{\mathcal{I}}_{\Sigma'} \rho$ for any χ-expansion M' of M, i.e., for any $M' \in \mathrm{Mod}^{\mathcal{I}}(\Sigma')$ such that $\mathrm{Mod}^{\mathcal{I}}(\chi)(M') = M$.

Moreover, and in order to enforce the satisfaction condition (and the functoriality of $\mathrm{Sen}^{\mathcal{I}}$), we have to study the transformation of quantification spaces under signature morphisms. In the *FOL* example we have $\mathrm{Sen}^{FOL}(\varphi)(\forall X\,\rho) = (\forall X^{\varphi})\,\mathrm{Sen}^{FOL}(\varphi')(\rho)$, where we take X^{φ} and φ' to defining the following pushout:

$$\begin{tikzcd}
(S, F, P) \arrow[r, "\varphi"] \arrow[d, "X"] & (S, F', P') \arrow[d, "X^{\varphi}"] \\
(S, F \uplus X, P) \arrow[r, "\varphi'"'] & (S, F' \uplus X^{\varphi}, P')
\end{tikzcd}$$

Thus, $X^{\varphi} = \{(x, \varphi_{st}(s), (S', F', P')) | (x, s, (S, F, P)) \in X\}$, and φ' is the canonical extension of φ, 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* Sign *a subclass of arrows* $\mathcal{D} \subseteq \mathrm{Sign}$ *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*

$$\begin{tikzcd} \Sigma \arrow[r, "\varphi"] \arrow[d, "\chi"'] & \Sigma_1 \arrow[d, "\chi(\varphi)"] \\ \Sigma' \arrow[r, "\varphi[\chi]"'] & \Sigma'_1 \end{tikzcd} \tag{6}$$

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*

$$\begin{tikzcd} \Sigma \arrow[r, "\varphi"] \arrow[d, "\chi"'] & \Sigma_1 \arrow[r, "\theta"] \arrow[d, "\chi(\varphi)"] & \Sigma_2 \arrow[d, "\chi(\varphi)(\theta)"] \\ \Sigma' \arrow[r, "\varphi[\chi]"'] & \Sigma'_1 \arrow[r, "\theta[\chi(\varphi)]"'] & \Sigma'_2 \end{tikzcd} \tag{7}$$

$\varphi[\chi]; \theta[\chi(\varphi)] = (\varphi; \theta)[\chi]$ *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* Sign *is* adequate *for a functor* $\mathrm{Mod} : \mathrm{Sign}^{op} \to \mathbb{CAT}$ *when the designated pushouts mentioned above are weak amalgamation squares for* Mod.

Example 2.3.9 (Standard quantification for *FOL*) X-expansions of *FOL*-signatures, i.e., morphisms $\chi : (S, F, P) \hookrightarrow (S, F + X, P)$, constitute adequate quantification spaces for Mod^{FOL}. In this case, for each $\varphi : (S, F, P) \to (S', F', P')$, we take as designated pushout the X^{φ}-expansions morphisms, and the respective φ', 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 Mod^{FOL} follows from the fact that Mod^{FOL} preserves all finite limits (cf. [Dia08]). ◦

Example 2.3.10 (Infinitary quantification for *FOL*) Another adequate quantification space for Mod^{FOL} considers *infinite* blocks of variables instead of finite ones. For each infinite set of variables X, we take X^{φ} and φ' 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 $\underline{ar} \in S^*$ and $s \in S$. Then, to any block X of second order variables it corresponds a signature extension $\chi : (S, F, P) \to (S, F + X^{op}, P + X^{rl})$ 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 latter realise 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

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

consists of

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

such that the following satisfaction condition *holds*

$$M' \models^{\mathcal{I}'}_{\Phi(\Sigma)} \alpha_\Sigma(e) \ \textit{iff}\ \beta_\Sigma(M') \models^{\mathcal{I}}_{\Sigma} e$$

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

Occasionally, for simplifying notation, we may denote a particular comorphisms $(\Phi, \alpha, \beta) : \mathcal{I} \to \mathcal{I}'$ just by $\mathcal{I}2\mathcal{I}'$.

Theorem 2.3.1 *Given a comorphism, for any set* $\Gamma \subseteq \mathrm{Sen}^{\mathcal{I}}(\Sigma)$ *and sentence* $\rho \in \mathrm{Sen}^{\mathcal{I}}(\Sigma)$,

$$\Gamma \models^{\mathcal{I}}_{\Sigma} \rho \ \textit{implies}\ \alpha_\Sigma(\Gamma) \models^{\mathcal{I}'}_{\Phi(\Sigma)} \alpha_\Sigma(\rho).$$

Proof.

$$\begin{array}{ll}
 & \Gamma \models^{\mathcal{I}}_{\Sigma} \rho \\
\Leftrightarrow & \quad \{\text{by definition}\} \\
 & \text{for any } M \in \mathrm{Mod}^{\mathcal{I}}(\Sigma), \\
 & M \models^{\mathcal{I}}_{\Sigma} \Gamma \text{ implies } M \models^{\mathcal{I}}_{\Sigma} \rho \\
\Rightarrow & \quad \{\text{instantiation}\} \\
 & \text{for any } M' \in \mathrm{Mod}^{\mathcal{I}'}(\Sigma'), \\
 & \beta(M') \models^{\mathcal{I}}_{\Sigma} \Gamma \text{ implies } \beta(M') \models^{\mathcal{I}}_{\Sigma} \rho \\
\Leftrightarrow & \quad \{\text{by hypothesis}\} \\
 & \text{for any } M' \in \mathrm{Mod}^{\mathcal{I}'}(\Sigma'), \\
 & M' \models^{\mathcal{I}'}_{\Phi(\Sigma)} \alpha(\Gamma) \text{ implies } M' \models^{\mathcal{I}'}_{\Phi(\Sigma)} \alpha(\rho) \\
\Leftrightarrow & \quad \{\text{by definition}\} \\
 & \alpha(\Gamma) \models^{\mathcal{I}'}_{\Phi(\Sigma)} \alpha(\rho)
\end{array}$$

□

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

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* $\Gamma \subseteq \mathrm{Sen}^{\mathcal{I}}(\Sigma)$ *and sentence* $\rho \in \mathrm{Sen}^{\mathcal{I}}(\Sigma)$,

$$\Gamma \models^{\mathcal{I}}_{\Sigma} \rho \textit{ iff } \alpha_{\Sigma}(\Gamma) \models^{\mathcal{I}'}_{\Phi(\Sigma)} \alpha_{\Sigma}(\rho).$$

Proof. The surjectivity of β entails the converse direction for the (second) $\Rightarrow$-implication step in the proof of Theorem 2.3.1. □

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

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

- $\alpha_{\mathtt{Prop}}(p) = p$, for $p \in \mathtt{Prop}$, and
- $\alpha_{\mathtt{Prop}}(\rho \otimes \rho') = \alpha_{\mathtt{Prop}}(\rho) \otimes \alpha_{\mathtt{Prop}}(\rho')$ for any $\otimes \in \{\vee, \wedge, \Rightarrow\}$ and $\alpha_{\mathtt{Prop}}(\neg\rho) = \neg\alpha_{\mathtt{Prop}}(\rho)$;

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

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

The comorphism is obviously conservative. ○

2.3.7 *Presentations*

We introduced, in this sub-section the notion of *institution of presentations* over a base institution $\mathcal{I}$.

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

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

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

Fact 2.3.3 *For any institution $\mathcal{I}$, $\mathcal{I}^{\mathrm{pres}}$ 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 $\mathcal{I}$ into a 'simpler' one $\mathcal{I}'$. In such encodings the structural complexity cost is shifted to the mapping Φ 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 $\mathcal{I} \to \mathcal{I}'^{\text{pres}}$ 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*^{pres}) Literature reports a number of possible encodings of *PA* to *FOL*^{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*^{pres}, where partial operations are encoded as total ones. In all of them comorphisms preserve different model theoretic properties, as discussed in [DM13].

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

1. Φ maps each signature $(S, \mathsf{TF}, \mathsf{PF})$ to the presentation

$$\big((S, \mathsf{TF} + \mathsf{PF}, (\mathsf{Def}_s)_{s \in S}), \Gamma_{(S,\mathsf{TF},\mathsf{PF})}\big)$$

   where $\Gamma_{(S,\mathsf{TF},\mathsf{PF})}$ axiomatizes the definability of terms through the new predicates $(\mathsf{Def}_s)_{s \in S}$ as follows:

$$\Gamma_{(S,\mathsf{TF},\mathsf{PF})} =$$
$$\{(\forall X)\mathsf{Def}_s(\sigma(X)) \Rightarrow \mathsf{Def}_{\underline{ar}}(X) \mid \sigma \in (\mathsf{TF} + \mathsf{PF})_{\underline{ar} \to s}, \underline{ar} \in S^*, s \in S\} \cup$$
$$\{(\forall X)\mathsf{Def}_{\underline{ar}}(X) \Rightarrow \mathsf{Def}_s(\sigma(X)) \mid \sigma \in \mathsf{TF}_{\underline{ar} \to s}, \underline{ar} \in S^*, s \in S\}$$

   (where $\mathsf{Def}_{\underline{ar}}(X)$ denotes $\bigwedge_{(x\,:\,s) \in X}(\mathsf{Def}_s(x))$).

2. $\alpha_{(S,\mathsf{TF},\mathsf{PF})}$ is recursively defined as follows:
   - $\alpha(t \overset{e}{=} t') = \mathsf{Def}_s(t) \wedge (t = t')$;
   - $\alpha((\forall X)\rho) = (\forall X)(\mathsf{Def}_{\underline{ar}}(X) \Rightarrow \alpha(\rho))$;
   - α commutes with boolean connectives $\wedge$, $\vee$, $\Rightarrow$, etc.

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

$\circ$

3

HYBRIDISATION OF INSTITUTIONS

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 [DM13].

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 $\mathcal{I}$ the method builds an institution $\mathcal{HI}$ 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 chapter 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 $\mathcal{I} = (\mathrm{Sign}^{\mathcal{I}}, \mathrm{Sen}^{\mathcal{I}}, \mathrm{Mod}^{\mathcal{I}}, (\models^{\mathcal{I}}_{\Sigma})_{\Sigma \in |\mathrm{Sign}^{\mathcal{I}}|})$ with a designated quantification space $\mathcal{D}^{\mathcal{I}} \subseteq \mathrm{Sign}^{\mathcal{I}}$. 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 a class of institutions, called the *hybridisations of* $\mathcal{I}$.

3.1.1 *The category of $\mathcal{HI}$-signatures*

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

$$\mathrm{Sign}^{\mathcal{HI}} = \mathrm{Sign}^{\mathcal{I}} \times \mathrm{Sign}^{REL}$$

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

General category theory entails:

Proposition 3.1.1 Sign^{REL} *has small co-limits.*

Proof. As Sign^{REL} is isomorphic to the power category $\mathbb{S}\mathrm{et}^{\mathbb{N}}$. As $\mathbb{S}\mathrm{et}$ has small co-limits it follows that Sign^{REL} has small co-limits. □

Corollary 3.1.1 *The projection* $\mathrm{Sign}^{\mathcal{HI}} \to \mathrm{Sign}^{\mathcal{I}}$ *lifts small co-limits.*

Proof. Let us suppose that C is a small co-limit in $\mathrm{Sign}^{\mathcal{I}}$. Then, take a small colimit C' in Sign^{REL} over the underlying diagram of C. Since the pair (C, C') is a small co-limit in $\mathrm{Sign}^{\mathcal{I}} \times \mathrm{Sign}^{\mathrm{REL}} = \mathrm{Sign}^{\mathcal{HI}}$, the projection (the proper C) is also a small co-limit.

□

3.1.2 $\mathcal{HI}$-*sentences fuctor*

Let us fix a quantification space $\mathcal{D}^{\mathcal{HI}}$ for $\mathrm{Sign}^{\mathcal{HI}}$ such that for each $\chi \in \mathcal{D}^{\mathcal{HI}}$ its projection χ_{Sign} to $\mathrm{Sign}^{\mathcal{I}}$ belongs to $\mathcal{D}^{\mathcal{I}}$. The quantification space $\mathcal{D}^{\mathcal{HI}}$ is a parameter of the hybridisation process. Whenever $\mathcal{D}^{\mathcal{HI}}$ 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 $\Delta = (\Sigma, \mathrm{Nom}, \Lambda)$. The set of sentences $\mathrm{Sen}^{\mathcal{HI}}(\Delta)$ is the least set such that

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

Whenever χ is a simple extension with variables it can be abbreviated to quantifications by the corresponding variables. For example when χ is an extension of $(\Sigma, \mathrm{Nom}, \Lambda)$ 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 $\mathrm{Sen}^{\mathcal{I}}(\Sigma) \hookrightarrow \mathrm{Sen}^{\mathcal{HI}}(\Sigma, \mathrm{Nom}, \Lambda)$. 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 $\mathcal{HI}$-sentences:

Let $\varphi = (\varphi_{\mathrm{Sign}}, \varphi_{\mathrm{Nom}}, \varphi_{\mathrm{MS}}) : (\Sigma, \mathrm{Nom}, \Lambda) \to (\Sigma', \mathrm{Nom}', \Lambda')$ be a morphim of $\mathcal{HI}$-signatures.

The translation $\mathrm{Sen}^{\mathcal{HI}}(\varphi)$ is defined as follows:

- $\mathrm{Sen}^{\mathcal{HI}}(\varphi)(\rho) = \mathrm{Sen}^{\mathcal{I}}(\varphi_{\mathrm{Sign}})(\rho)$ for any $\rho \in \mathrm{Sen}^{\mathcal{I}}(\Sigma)$;
- $\mathrm{Sen}^{\mathcal{HI}}(\varphi)(i) = \varphi_{\mathrm{Nom}}(i)$;
- $\mathrm{Sen}^{\mathcal{HI}}(\varphi)(\neg\rho) = \neg\mathrm{Sen}^{\mathcal{HI}}(\varphi)(\rho)$;
- $\mathrm{Sen}^{\mathcal{HI}}(\varphi)(\rho \star \rho') = \mathrm{Sen}^{\mathcal{HI}}(\varphi)(\rho) \star \mathrm{Sen}^{\mathcal{HI}}(\varphi)(\rho'), \star \in \{\vee, \wedge, \Rightarrow\}$;
- $\mathrm{Sen}^{\mathcal{HI}}(\varphi)(@_i\rho) = @_{\varphi_{\mathrm{Nom}}(i)}\mathrm{Sen}^{\mathcal{HI}}(\rho)$;
- $\mathrm{Sen}^{\mathcal{HI}}(\varphi)([\lambda](\rho_1, \ldots, \rho_n)) = [\varphi_{\mathrm{MS}}(\lambda)](\mathrm{Sen}^{\mathcal{HI}}(\rho_1), \ldots, \mathrm{Sen}^{\mathcal{HI}}(\rho_n))$;
- $\mathrm{Sen}^{\mathcal{HI}}(\varphi)(\langle\lambda\rangle(\rho_1, \ldots, \rho_n)) = \langle\varphi_{\mathrm{MS}}(\lambda)\rangle(\mathrm{Sen}^{\mathcal{HI}}(\rho_1), \ldots, \mathrm{Sen}^{\mathcal{HI}}(\rho_n))$;
- $\mathrm{Sen}^{\mathcal{HI}}(\varphi)\big((\forall\chi)\rho\big) = (\forall\chi(\varphi))\mathrm{Sen}^{\mathcal{HI}}(\varphi[\chi])(\rho)$;
- $\mathrm{Sen}^{\mathcal{HI}}(\varphi)\big((\exists\chi)\rho\big) = (\exists\chi(\varphi))\mathrm{Sen}^{\mathcal{HI}}(\varphi[\chi])(\rho)$;

Proposition 3.1.2 $\mathrm{Sen}^{\mathcal{HI}}$ *is a functor* $\mathrm{Sign}^{\mathcal{HI}} \to \mathcal{S}et$.

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

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

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

$$\begin{array}{ll}
 & \mathrm{Sen}^{\mathcal{HI}}(\varphi);\mathrm{Sen}^{\mathcal{HI}}(\theta)(\rho) \\
= & \quad \{\text{ defn. of composition}\} \\
 & \mathrm{Sen}^{\mathcal{HI}}(\theta)\big(\mathrm{Sen}^{\mathcal{HI}}(\varphi)(\rho)\big) \\
= & \quad \{\text{ defn of } \mathrm{Sen}^{\mathcal{HI}} \text{ and } \rho \in \mathrm{Sen}^{\mathcal{I}}(\Sigma)\} \\
 & \mathrm{Sen}^{\mathcal{HI}}(\theta)\big(\mathrm{Sen}^{\mathcal{I}}(\varphi_{\mathrm{Sign}})(\rho)\big) \\
= & \quad \{\text{ defn of } \mathrm{Sen}^{\mathcal{HI}} \text{ and } \mathrm{Sen}^{\mathcal{I}}(\varphi_{\mathrm{Sign}})(\rho) \in \mathrm{Sen}^{\mathcal{I}}(\Sigma')\} \\
 & \mathrm{Sen}^{\mathcal{I}}(\theta_{\mathrm{Sign}})\big(\mathrm{Sen}^{\mathcal{I}}(\varphi_{\mathrm{Sign}})(\rho)\big) \\
= & \quad \{\text{ since } \mathrm{Sen}^{\mathcal{I}} \text{ is a functor}\} \\
 & \mathrm{Sen}^{\mathcal{I}}(\theta_{\mathrm{Sign}};\varphi_{\mathrm{Sign}})(\rho) \\
= & \quad \{\text{ defn. of } \mathrm{Sen}^{\mathcal{HI}} \text{ and } \rho \in \mathrm{Sen}^{\mathcal{I}}(\Sigma)\}
\end{array}$$

$$\mathrm{Sen}^{\mathcal{HI}}(\theta;\varphi)(\rho)$$

For sentences $\rho = @_i\rho$ we have:

$$\begin{aligned}
& \mathrm{Sen}^{\mathcal{HI}}(\varphi);\mathrm{Sen}^{\mathcal{HI}}(\theta)(@_i\rho) \\
= & \quad \{\text{ defn. of composition}\} \\
& \mathrm{Sen}^{\mathcal{HI}}(\theta)\big(\mathrm{Sen}^{\mathcal{HI}}(\varphi)(@_i\rho)\big) \\
= & \quad \{\text{ defn of } \mathrm{Sen}^{\mathcal{HI}}\} \\
& \mathrm{Sen}^{\mathcal{HI}}(\theta)\big(@_{\varphi_{\mathrm{Nom}}(i)}\mathrm{Sen}^{\mathcal{HI}}(\varphi)(\rho)\big) \\
= & \quad \{\text{ defn of } \mathrm{Sen}^{\mathcal{HI}}\} \\
& @_{\theta_{\mathrm{Nom}}(\varphi_{\mathrm{Nom}}(i))}\mathrm{Sen}^{\mathcal{HI}}(\theta)\big(\mathrm{Sen}^{\mathcal{HI}}(\varphi)(\rho)\big) \\
= & \quad \{\text{ I.H. and defn of composition}\} \\
& @_{\varphi_{\mathrm{Nom}};\theta_{\mathrm{Nom}}(i)}\mathrm{Sen}^{\mathcal{HI}}(\theta;\varphi)(\rho) \\
= & \quad \{\text{ defn of } \mathrm{Sen}^{\mathcal{HI}}\} \\
& \mathrm{Sen}^{\mathcal{HI}}(\varphi;\theta)(@_i\rho)
\end{aligned}$$

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 $\rho\star\rho', \star\in\{\wedge,\vee,\Rightarrow\}$, as well as of $\neg\rho$ comes directly from the induction hypothesis and the definition of $\mathrm{Sen}^{\mathcal{HI}}$. Finally, for the sentences $(\forall\chi)\rho \in \mathrm{Sen}^{\mathcal{HI}}(\Sigma,\mathrm{Nom},\Lambda)$, we have:

$$\begin{aligned}
& \mathrm{Sen}^{\mathcal{HI}}(\varphi);\mathrm{Sen}^{\mathcal{HI}}(\theta)((\forall\chi)\rho) \\
= & \quad \{\text{ defn. of composition}\} \\
& \mathrm{Sen}^{\mathcal{HI}}(\theta)\big(\mathrm{Sen}^{\mathcal{HI}}(\varphi)((\forall\chi)\rho)\big) \\
= & \quad \{\text{ defn of } \mathrm{Sen}^{\mathcal{HI}}\ \} \\
& \mathrm{Sen}^{\mathcal{HI}}(\theta)\big((\forall\chi(\varphi))\mathrm{Sen}^{\mathcal{HI}}(\varphi[\chi])(\rho)\big) \\
= & \quad \{\text{ defn of } \mathrm{Sen}^{\mathcal{HI}}\ \} \\
& (\forall(\chi(\varphi)\theta))\mathrm{Sen}^{\mathcal{HI}}(\theta[\chi(\varphi)])(\mathrm{Sen}^{\mathcal{HI}}(\varphi[\chi]))(\rho) \\
= & \quad \{\text{ I.H.}\} \\
& (\forall(\chi(\varphi)\theta))(\mathrm{Sen}^{\mathcal{HI}}(\varphi[\chi];\theta[\chi(\varphi)]))(\rho) \\
= & \quad \{\text{ (7) of Definition 2.3.6}\} \\
& \mathrm{Sen}^{\mathcal{HI}}(\varphi;\theta)((\forall\chi)\rho)
\end{aligned}$$

□

3.1.3 $\mathcal{HI}$-*models functor*

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

- W is a (Nom, Λ)-model in *REL*;
- M is a function $|W| \to |\mathrm{Mod}^{\mathcal{I}}(\Sigma)|$.

The carrier set $|W|$ is *the set of states of* (M, W); $\{W_n \mid n \in \mathrm{Nom}\}$ 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 Λ. We denote $M(w)$ simply by M_w.

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

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

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

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

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

Reducts of $\mathcal{HI}$-models

Let $\Delta = (\Sigma, \mathrm{Nom}, \Lambda)$ and $\Delta' = (\Sigma', \mathrm{Nom}', \Lambda')$ be two $\mathcal{HI}$-signatures, $\varphi = (\varphi_{\mathrm{Sign}}, \varphi_{\mathrm{Nom}}, \varphi_{\mathrm{MS}})$ a morphism between Δ and Δ' and (M', W') a Δ'-model. The *reduct* of (M', W') along φ, denoted by $\mathrm{Mod}^{\mathcal{HI}}(\varphi)(M', W')$, is the Δ-model (M, W) where

- W is the $(\varphi_{\mathrm{Nom}}, \varphi_{\mathrm{MS}})$-reduct of W'; i.e.
  - $|W| = |W'|$;
  - for any $n \in \mathrm{Nom}, W_n = W'_{\varphi_{\mathrm{Nom}}(n)}$;
  - for any $\lambda \in \Lambda$, $W_\lambda = W'_{\varphi_{\mathrm{MS}}(\lambda)}$;

  and
- for any $w \in |W|$, $M_w = \mathrm{Mod}^{\mathcal{I}}(\varphi_{\mathrm{Sign}})(M'_w)$.

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

$$\mathrm{Mod}^{\mathcal{HI}}(\varphi)(h') : \mathrm{Mod}^{\mathcal{HI}}(\varphi)(M'_1, W'_1) \to \mathrm{Mod}^{\mathcal{HI}}(\varphi)(M'_2, W'_2)$$

defined by $(h'_{\mathrm{st}}, h'_{\mathrm{mod}}; \mathrm{Mod}^{\mathcal{I}}(\varphi))$.

Fact 3.1.2 $\mathrm{Mod}^{\mathcal{HI}}$ *is a functor.*

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

Proof. Given a pushout of $\mathcal{HI}$-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 *REL*; this is because the projection functor $\mathrm{Sign}^{\mathcal{HI}} = \mathrm{Sign}^{\mathcal{I}} \times \mathrm{Sign}^{REL} \to \mathrm{Sign}^{REL}$ preserves pushouts, hence the underlying square of signature morphisms in *REL* 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 $M' : |W| \to |\mathrm{Mod}^{\mathcal{I}}(\Sigma')|$ is the function determined by the (weak) pullback property of the result of applying functor $|\mathrm{Mod}^{\mathcal{I}}(-)|$ to the underlying square of signature morphisms in $\mathcal{I}$.

□

Corollary 3.1.2 *If $\mathcal{D}^{\mathcal{I}}$ is adequate for* $\mathrm{Mod}^{\mathcal{I}}$ *then $\mathcal{D}^{\mathcal{HI}}$ is adequate for* $\mathrm{Mod}^{\mathcal{HI}}$.

Proof. Any designated square Sq in the quantification space $\mathcal{D}^{\mathcal{HI}}$ is a pushout by definition and since the projection functor $\mathrm{Sign}^{\mathcal{HI}} = \mathrm{Sign}^{\mathcal{I}} \times \mathrm{Sign}^{REL} \to \mathrm{Sign}^{\mathcal{I}}$ preserves pushouts, it follows that the underlying square Sq_0 of Sq consisting of $\mathcal{I}$ signature morphisms, is also a pushout. Because of this and also because the projection of $\mathcal{D}^{\mathcal{HI}}$ to $\mathcal{I}$ is included in $\mathcal{D}^{\mathcal{I}}$, it follows that Sq_0 is isomorphic to a designated square in the quantification space $\mathcal{D}^{\mathcal{I}}$. This, by the adequacy property of $\mathcal{D}^{\mathcal{I}}$ with respect to $\mathrm{Mod}^{\mathcal{I}}$, is a weak amalgamation square. Hence Sq_0 is a weak amalgamation square, which by Theorem 3.1.1, implies that Sq is adequate for $\mathrm{Mod}^{\mathcal{HI}}$. □

3.1.4 *The satisfaction relation*

For any $(M, W) \in |\mathrm{Mod}^{\mathcal{HI}}(\Sigma, \mathrm{Nom}, \Lambda)|$ and for any $w \in |W|$,

- $(M, W) \models^w \rho$ iff $M_w \models^{\mathcal{I}} \rho$; when $\rho \in \mathrm{Sen}^{\mathcal{I}}(\Sigma)$,
- $(M, W) \models^w i$ iff $W_i = w$; when $i \in \mathrm{Nom}$,
- $(M, W) \models^w \rho \vee \rho'$ iff $(M, W) \models^w \rho$ or $(M, W) \models^w \rho'$,
- $(M, W) \models^w \rho \wedge \rho'$ iff $(M, W) \models^w \rho$ and $(M, W) \models^w \rho'$,
- $(M, W) \models^w \rho \Rightarrow \rho'$ iff $(M, W) \models^w \rho$ implies that $(M, W) \models^w \rho'$,
- $(M, W) \models^w \neg\rho$ iff $(M, W) \not\models^w \rho$,
- $(M, W) \models^w [\lambda](\xi_1, \ldots, \xi_n)$ iff for any $(w, w_1, \ldots, w_n) \in W_\lambda$ we have that $(M, W) \models^{w_i} \rho_i$ for some $1 \leqslant i \leqslant n$.
- $(M, W) \models^w \langle\lambda\rangle(\xi_1, \ldots, \xi_n)$ iff there exists $(w, w_1, \ldots, w_n) \in W_\lambda$ such that and $(M, W) \models^{w_i} \xi_i$ for any $1 \leqslant i \leqslant n$.
- $(M, W) \models^w @_j\rho$ iff $(M, W) \models^{W_j} \rho$,
- $(M, W) \models^w (\forall\chi)\rho$ iff $(M', W') \models^w \rho$ for any (M', W') such that $\mathrm{Mod}^{\mathcal{HI}}(\chi)(M', W') = (M, W)$,
- $(M, W) \models^w (\exists\chi)\rho$ iff $(M', W') \models^w \rho$ for some (M', W') such that $\mathrm{Mod}^{\mathcal{HI}}(\chi)(M', W') = (M, W)$, and

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 $\mathcal{D}^{\mathcal{I}}$ is adequate for* $\mathrm{Mod}^{\mathcal{I}}$. *Let* $\Delta = (\Sigma, \mathrm{Nom}, \Lambda)$ *and* $\Delta' = (\Sigma', \mathrm{Nom}', \Lambda')$ *be two $\mathcal{HI}$-signatures and* $\varphi : \Delta \to \Delta'$ *a morphism of signatures. For any* $\rho \in \mathrm{Sen}^{\mathcal{HI}}(\Delta)$, $(M', W') \in |\mathrm{Mod}^{\mathcal{HI}}(\Delta')|$, *and* $w \in |W|$,

$$\mathrm{Mod}^{\mathcal{HI}}(\varphi)(M', W')) \models^{w} \rho \textit{ iff } (M', W') \models^{w} \mathrm{Sen}^{\mathcal{HI}}(\varphi)(\rho).$$

Proof. Let us denote $\mathrm{Mod}^{\mathcal{HI}}(\varphi)(M', W') = \mathrm{Mod}^{\mathcal{HI}}(\varphi)(M', W')$ by (M, W). The proof is by recursion on the structure of the sentence ρ

1. $\rho = i$ for some $i \in \mathrm{Nom}$

$$\begin{array}{ll} & (M, W) \models^{w} i \\ \Leftrightarrow & \quad \{ \text{ defn. of } \models^{w} \} \\ & W_i = w \\ \Leftrightarrow & \quad \{ \text{ defn of reduct, } W'_{\varphi_{\mathrm{Nom}}(i)} = W_i \} \\ & (M', W') \models^{w} \varphi_{\mathrm{Nom}}(i) \\ \Leftrightarrow & \quad \{ \text{ defn of } \mathrm{Sen}^{\mathcal{HI}}(\varphi) \} \\ & (M', W') \models^{w} \mathrm{Sen}^{\mathcal{HI}}(\varphi)(i) \end{array}$$

2. $\rho \in \mathrm{Sen}^{\mathcal{I}}(\Sigma)$

$$\begin{array}{ll} & (M, W) \models^{w} \rho \\ \Leftrightarrow & \quad \{ \text{ defn. of } \models^{w} \} \\ & \mathrm{Mod}^{\mathcal{I}}(\varphi_{\mathrm{Sign}})(M'_w) = M_w \models^{\mathcal{I}} \rho \\ \Leftrightarrow & \quad \{ \text{ by the Satisfaction Condition in } \mathcal{I} \} \\ & M'_w \models \mathrm{Sen}^{\mathcal{I}}(\varphi_{\mathrm{Sign}})(\rho) \end{array}$$

$$\Leftrightarrow \quad \{\text{ defn. of } \models^w\}$$
$$(M', W') \models^w \mathrm{Sen}^{\mathcal{I}}(\varphi_{\mathrm{Sign}})(\rho)$$
$$\Leftrightarrow \quad \{\text{ defn of } \mathrm{Sen}^{\mathcal{HI}}(\varphi)\}$$
$$(M', W') \models^w \mathrm{Sen}^{\mathcal{HI}}(\varphi)(\rho)$$

3. $\rho = \xi \vee \xi'$ for some $\xi, \xi' \in \mathrm{Sen}^{\mathcal{HI}}(\Delta)$

$$(M, W) \models^w \xi \vee \xi'$$
$$\Leftrightarrow \quad \{\text{ defn. of } \models^w\}$$
$$(M, W) \models^w \xi \text{ or } (M, W) \models^w \xi'$$
$$\Leftrightarrow \quad \{\text{ I.H.}\}$$
$$(M', W') \models^w \mathrm{Sen}^{\mathcal{HI}}(\varphi)(\xi) \text{ or}$$
$$(M', W') \models^w \mathrm{Sen}^{\mathcal{HI}}(\varphi)(\xi')$$
$$\Leftrightarrow \quad \{\text{ defn. of } \models^w\}$$
$$(M', W') \models^w \mathrm{Sen}^{\mathcal{HI}}(\varphi)(\xi \vee \xi')$$

The proofs for the cases when $\rho = \xi \wedge \xi'$, $\rho = \xi \Rightarrow \xi'$, $\rho = \neg\xi$, etc. are analogous.

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

$$(M, W) \models^w [\lambda](\xi_1, \ldots, \xi_n)$$
$$\Leftrightarrow \quad \{\text{ defn. of } \models^w\}$$
for any $(w, w_1, \ldots, w_n) \in W_\lambda$ there is some $k \in \{1, \ldots, n\}$ such that $(M, W) \models^{w_k} \xi_k$
$$\Leftrightarrow \quad \{\text{ by reduct defn, } W_\lambda = W'_{\varphi_{\mathrm{MS}}}(\lambda)\}$$
for any $(w, w_1, \ldots, w_n) \in W'_{\varphi_{\mathrm{MS}}(\lambda)}$ there is some $k \in \{1, \ldots, n\}$ such that $(M, W) \models^{w_k} \xi_k$
$$\Leftrightarrow \quad \{\text{ I.H.}\}$$
for any $(w, w_1, \ldots, w_n) \in W'_{\varphi_{\mathrm{MS}}(\lambda)}$ there is some $k \in \{1, \ldots, n\}$ such that

$$
\begin{array}{ll}
 & (M', W') \models^{w_k} \mathrm{Sen}^{\mathcal{HJ}}(\varphi)(\xi_k) \\
\Leftrightarrow & \quad \{\text{ defn. of } \models^{w}\} \\
 & (M', W') \models^{w} [\varphi_{\mathrm{MS}}(\lambda)](\mathrm{Sen}^{\mathcal{HJ}}(\varphi)(\xi_1), \ldots, \mathrm{Sen}^{\mathcal{HJ}}(\varphi)(\xi_n)) \\
\Leftrightarrow & \quad \{\text{ defn. of } \mathrm{Sen}^{\mathcal{HJ}}(\varphi)\} \\
 & (M', W') \models^{w} \mathrm{Sen}^{\mathcal{HJ}}(\varphi)([\lambda](\xi_1, \ldots, \xi_n))
\end{array}
$$

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

5. $\rho = @_i\xi$ for some $\xi \in \mathrm{Sen}^{\mathcal{HJ}}(\Delta)$, $i \in \mathrm{Nom}$

$$
\begin{array}{ll}
 & (M, W) \models^{w} @_i\xi \\
\Leftrightarrow & \quad \{\text{ defn. of } \models^{w}\} \\
 & (M, W) \models^{W_i} \xi \\
\Leftrightarrow & \quad \{\text{I.H.}\} \\
 & (M', W') \models^{W_i} \mathrm{Sen}^{\mathcal{HJ}}(\varphi)(\xi) \\
\Leftrightarrow & \quad \{\text{ by reduct defn, } W_i = W'_{\varphi_{\mathrm{Nom}}(i)}\} \\
 & (M', W') \models^{W'_{\varphi_{\mathrm{Nom}}(i)}} \mathrm{Sen}^{\mathcal{HJ}}(\varphi)(\xi) \\
\Leftrightarrow & \quad \{\text{ defn. of } \models^{w}\} \\
 & (M', W') \models^{w} @_{\varphi_{\mathrm{Nom}}(i)}\mathrm{Sen}^{\mathcal{HJ}}(\varphi)(\xi) \\
\Leftrightarrow & \quad \{\text{ defn. of } \mathrm{Sen}^{\mathcal{HJ}}(\varphi)\} \\
 & (M', W') \models^{w} \mathrm{Sen}^{\mathcal{HJ}}(\varphi)(@_i\xi)
\end{array}
$$

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

Figure 1 depicts the proof scheme. Hence,

$$
\begin{array}{ll}
 & (M, W) \models^{w} (\forall\chi)\xi \\
\Leftrightarrow & \quad \{\text{ defn. of } \models^{w}\} \\
 & \text{for all } (M_1, W_1) \text{ such that } \mathrm{Mod}^{\mathcal{HJ}}(\chi)(M_1, W_1) = (M, W), \\
 & (M_1, W_1) \models^{w} \xi \\
\Leftrightarrow & \quad \{\text{ by Corollary 3.1.2, } \mathcal{D}^{\mathcal{HJ}} \text{ is adequate for } \mathrm{Mod}^{\mathrm{C}}\}
\end{array}
$$

$$
\begin{tikzcd}
(M,W) & & (M',W') \arrow[ll, "\mathrm{Mod}^{\mathcal{HI}}(\varphi)"'] \\
& \Delta \arrow[r, "\varphi"] \arrow[d, "\chi"'] & \Delta' \arrow[d, "\chi(\varphi)"] \\
& \Delta_1 \arrow[r, "\varphi[\chi]"'] & \Delta'_1 \\
(M_1,W_1) \arrow[uu, "\mathrm{Mod}^{\mathcal{HI}}(\chi)"] & & (M'_1,W'_1) \arrow[ll, "\mathrm{Mod}^{\mathcal{HI}}(\varphi[\chi])"] \arrow[uu, "\mathrm{Mod}^{\mathcal{HI}}(\chi(\varphi))"']
\end{tikzcd}
$$

Figure 1: Proof Scheme

$$
\begin{array}{ll}
 & \text{for all } (M'_1,W'_1) \text{ such that} \\
 & \mathrm{Mod}^{\mathcal{HI}}(\chi(\varphi))(M'_1,W'_1) = (M',W'), \\
 & \mathrm{Mod}^{\mathcal{HI}}(\varphi[\chi])(M'_1,W'_1) \models^w \xi \\
\Leftrightarrow & \quad \{\text{I.H.}\} \\
 & \text{for all } (M'_1,W'_1) \text{ such that} \\
 & \mathrm{Mod}^{\mathcal{HI}}(\chi(\varphi))(M'_1,W'_1) = (M',W'), \\
 & (M'_1,W'_1) \models^w \mathrm{Sen}^{\mathcal{HI}}(\varphi[\chi])(\xi) \\
\Leftrightarrow & \quad \{\text{ defn. of } \models^w\} \\
 & (M',W') \models^w (\forall\chi(\varphi))\mathrm{Sen}^{\mathcal{HI}}(\varphi[\chi])(\xi) \\
\Leftrightarrow & \quad \{\text{defn of } \mathrm{Sen}^{\mathcal{HI}}(\varphi)\} \\
 & (M',W') \models^w \mathrm{Sen}^{\mathcal{HI}}(\varphi)((\forall\chi)\xi)
\end{array}
$$

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

□

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

Corollary 3.1.3 (The Satisfaction Condition) *$\mathcal{HI}$ is an institution.*

3.1.5 *Base logic versus its hybridisation*

In hybridised institutions we may have to consider, at the level of the sentences of the base institution, 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 $\mathcal{HI}$ be an hybridisation of $\mathcal{I}$ and let us denote the Boolean connectives and the quantifiers in the base institution $\mathcal{I}$ by* $\textcircled{\wedge}$, $\textcircled{\vee}$, $\textcircled{\Rightarrow}$, $\textcircled{\neg}$, *and* $\textcircled{\forall}$, $\textcircled{\exists}$, *respectively. For any* $(\Sigma, \mathrm{Nom}, \Lambda)$*-model* (M, W), $w \in |W|$, *sentences* $\rho, \rho' \in \mathrm{Sen}^{\mathcal{I}}(\Sigma)$ *of the base institution and for each* $\chi \in \mathcal{D}$

- $(M, W) \models^{w} \rho \star \rho'$ *iff* $(M, W) \models^{w} \rho \textcircled{\star} \rho'$ *for* $\star \in \{\wedge, \vee, \Rightarrow\}$,
- $(M, W) \models^{w} \neg\rho$ *iff* $(M, W) \models^{w} \textcircled{\neg}\, \rho$,
- $(M, W) \models^{w} (\textcircled{\forall}\chi)\rho$ *implies* $(M, W) \models^{w} (\forall(\chi, 1_{\mathrm{Nom}}, 1_{\Lambda}))\rho$, *and*
- $(M, W) \models^{w} (\exists(\chi, 1_{\mathrm{Nom}}, 1_{\Lambda}))\rho$ *implies* $(M, W) \models^{w} (\textcircled{\exists}\chi)\rho$.

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

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

- $\Phi(\Sigma) = (\Sigma, \{i\}, \emptyset)$,
- $\alpha_{\Sigma}(\rho) = @_i\rho$, *and*
- $\beta_{\Sigma}(M, W) = M_{W_i}$.

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

$$
\begin{array}{ll}
 & \beta(M, W) \models^{\mathcal{I}} \rho \\
\Leftrightarrow & \quad \{\text{ defn. of } \beta\} \\
 & M_{W_i} \models^{\mathcal{I}} \rho \\
\Leftrightarrow & \quad \{\text{ defn of } \models^{\mathcal{HI}}\} \\
 & (M, W) \models^{\mathcal{HI}} @_i\rho
\end{array}
$$

$$\Leftrightarrow \qquad \{\text{ defn of } \alpha\}$$
$$(M, W) \models^{\mathcal{HI}} \alpha(\rho)$$

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

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 characterisation of first-order encodings introduced in the next chapter.

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

In the sequel, an hybridisation without constrained models, i.e., with $\mathrm{Mod}^{\mathsf{C}} = \mathrm{Mod}^{\mathcal{HI}}$ is called *free hybridisation* of $\mathcal{I}$. Observe that the satisfaction relation $\models^{\mathcal{HI}^{\mathsf{C}}}$ of a hybridisation $\mathcal{HI}^{\mathsf{C}}$ with properly constrained models is not necessarily the restriction of $\models^{\mathcal{HI}}$. This is justified by the fact that the satisfaction relation for quantifiers relies upon expansions with respect to $\mathrm{Mod}^{\mathsf{C}}$ and not to $\mathrm{Mod}^{\mathcal{HI}}$.

3.2 EXAMPLES

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 $\mathcal{D}^{\mathcal{HI}}$, and
3. the constrained models (Mod^{C}).

Example 3.2.1 ($\mathcal{H}PL$) Applying the quantifier-free version of the hybridisation method to PL and fixing $\Lambda_2 = \{\lambda\}$ and $\Lambda_n = \emptyset$ for each $n \neq 2$, we obtain the institution of "standard" *hybrid propositional logic* (without state quantifiers). The category of signatures is $\mathrm{Sign}^{\mathcal{H}PL} = \mathbf{Set} \times \mathbf{Set}$ with objects denoted by (P, Nom) and morphisms by $(\varphi_{\mathrm{Sign}}, \varphi_{\mathrm{Nom}})$; sentences are the usual hybrid propositional formulas, i.e., modal formulas closed by Boolean connectives, $[\lambda]$ denoted by $\Box$, $\langle\lambda\rangle$ denoted by $\Diamond$, and operators $@_i$, for $i \in \mathrm{Nom}$; models consist of pairs (M, W) where W consists of a carrier set $|W|$, interpretations $W_i \in |W|$ for each $i \in \mathrm{Nom}$, and a binary relation $W_\lambda \subseteq |W| \times |W|$, and for each $w \in |W|$, M_s is a propositional model, i.e., a function $M_w : P \to \{\top, \bot\}$ which is equivalent to a subset $M_w \subseteq P$. Note that, by Fact 3.1.3, we do not need to make a distinction between the Boolean connectives at the level of PL and those 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_λ is a reflexive relation; a preorder, and an equivalence relation, respectively (e.g. [BdRV01]). By considering an arbitrary set of modalities Λ, instead of a single λ, we set the "multi-modal hybrid propositional logic".

Similarly, but considering $\mathrm{Nom} = \emptyset$, we obtain the standard (non-hybrid) modal logics T, $S4$ and $S5$ (e.g. [BdRV01, GO07]).

A challenging issue concerns finding suitable quantification spaces to capture versions of hybrid propositional logic. One choice is the quantifier-free version in which $\mathcal{D}^{\mathcal{H}PL}$ consists only of identities. However, it would be interesting, along the hybridisation process, to capture a quantifier such as $\mathbf{E}$, where $\mathbf{E}\rho$ means that "ρ *is true in some state of the model*" [AB01]. Considering as quantification spaces the exten-

sions of signatures with nominal symbols, paves the way to expressing the following properties:

$(M, W) \models^w ((\forall i)i) \Leftrightarrow \rho$	iff	ρ is satisfied at w iff w is unique in (M, W)
$(M, W) \models^w (\exists i)@_i\rho$	iff	$(M, W) \models \mathbf{E}\rho$

A *block of nominal variables* X for a $\mathcal{H}PL$ signature (P, Nom) is a finite set of nominal variables of the form (x, P, Nom) (as in the case of *FOL* variables, x is the name and (P, Nom) the qualification of the variable) such that $(x, P, \mathrm{Nom}), (x', P, \mathrm{Nom}) \in X$ implies $x = x'$. Then $\mathcal{D}^{\mathcal{H}PL}$ may be defined by of the signature extensions with blocks of nominal variables, i.e. $(P, \mathrm{Nom}) \hookrightarrow (P, \mathrm{Nom} \cup X)$. For any signature morphism $\varphi : (P, \mathrm{Nom}) \to (P', \mathrm{Nom}')$ and X a block of nominal variables for (P, Nom) we define

$$X^\varphi = \{(x, P', \mathrm{Nom}') \mid (x, P, \mathrm{Nom}) \in X\}.$$

Then $\chi(\varphi)$ is the extension $(P', \mathrm{Nom}') \hookrightarrow (P', \mathrm{Nom}' \cup X^\varphi)$ and $\varphi[\chi]$ is the canonical extension of φ that maps each (x, P, Nom) to (x, P', Nom').

When we combine this quantification with the constraineds 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.

$\circ$

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

$$\mathrm{Sign}^{TRM} \times \mathrm{Sign}^{REL} \cong \mathrm{Sign}^{REL}.$$

Since $\mathrm{Sen}^{TRM}(*) = \emptyset$, we have that $\mathrm{Sen}^{\mathcal{H}TRM}(*, \mathrm{Nom}, \Lambda)$ is the set of sentences built up from nominals in Nom by the application of

modalities in Λ and boolean connectives. This kind of formulas has been called by *pure hybrid formulas* of [BdRV01, Ind07].The models of $\mathrm{Mod}^{\mathcal{H}TRM}(*, \mathrm{Nom}, \Lambda)$ consist of relational structures (W, M), where M is a constant function $M_w = *$, for any $w \in |W|$. This fragment of $\mathcal{H}PL$ has been studied in, e.g., [Ind07], namely concerning completness. ○

Example 3.2.3 ($\mathcal{H}^2PL$) Let us consider the quantified free hybridisation of $\mathcal{H}PL$ (of Example 3.2.1), say $\mathcal{H}^2PL$. Hence, signatures are of form $\big((P, \mathrm{Nom}', \Lambda'), \mathrm{Nom}, \Lambda\big)$, for $(P, \mathrm{Nom}', \Lambda') \in \mathrm{Sign}^{\mathcal{H}PL}$. In order to prevent potential ambiguities, we represent explicitly the natural embedding of the base institutions into its hybridisation: i.e., for any $\rho \in \mathrm{Sign}^{\mathcal{H}PL}(P, \mathrm{Nom}'\Lambda')$, an atomic sentence $\rho \in \mathrm{Sign}^{\mathcal{H}^2PL}\big((P, \mathrm{Nom}', \Lambda'), \mathrm{Nom}, \Lambda\big)$ is represented by $\underline{\rho}$. Thus is no ambiguity in sentences like $i \wedge \underline{i}$ or $@_i\underline{@_i\rho}$. By Fact 3.1.3 we do not make any distinction between the Boolean connectives at the levels of $\mathcal{H}PL$ and $\mathcal{H}^2PL$. In particular, we have that $\underline{\rho \vee \rho'}$ is equivalent to $\underline{\rho} \vee \underline{\rho'}$and similarly for the other connectives. The models of $\mathrm{Mod}^{\mathcal{H}^2PL}$ are, hence, *"Kripke structures of Kripke structures"*. To represent that we adopt again the same "underline" convention: we denote models in $\mathrm{Mod}^{\mathcal{H}^2PL}\big((P, \mathrm{Nom}, \Lambda'), \mathrm{Nom}, \Lambda\big)$ by pairs (M, W) and, for any $w \in W$, models M_w in $\mathrm{Mod}^{\mathcal{H}PL}(P, \mathrm{Nom}', \Lambda')$ by $(\underline{W_w}, \underline{M_w})$.

Let us consider also the institution $\mathcal{H}^2PL'$ by constraining the models to those with sharing of "sub-states universe", i.e., to models (M, W) such that $w, w' \in |W|$, $|W_w| = |W_{w'}|$. The reflection condition of Definition 3.1.1 holds directly from the $\mathcal{H}PL$-reduct definition: for any pushout square of signature morphisms in $\mathrm{Sign}^{\mathcal{H}^2PL}$

$$\begin{tikzcd} \Delta \arrow[r, "\varphi_1"] \arrow[d, "\varphi_2"'] & \Delta_1 \arrow[d, "\theta_1"] \\ \Delta_2 \arrow[r, "\theta_2"'] & \Delta' \end{tikzcd}$$

let us consider a constrained $(\Sigma_k, \mathrm{Nom}_k, \Lambda_k)$-model (M_k, W_k) for each $k \in \{1, 2\}$ such that $\mathrm{Mod}^C(\varphi_1)(M_1, W_1) = \mathrm{Mod}^C(\varphi_2)(M_2, W_2)$. We take the amalgamation $(M', W') = (M_1, W_1) \otimes (M_2, W_2)$ according to Theorem 3.1.1. By definition of reduct we have that $|W'| = |W_k|$ and that, for any $w \in |W'|$, $\mathrm{Mod}^{\mathcal{H}PL}(\theta_{\mathrm{Sign}_k})(\underline{W'_w}, \underline{M'_w}) =$

$(\underline{W_{kw}}, \underline{M_{kw}})$ and, hence $|\underline{W'_w}| = |\underline{W_{kw}}|$. Since (M_k, W_k) are constrained models, we have that $|\underline{W_{kw}}| = |\underline{W_{kw'}}|$ for any $w, w' \in |W_k|(= |W'|)$. Therefore $|W'_w| = |W'_{w'}|$ for any $w, w' \in |W'|$, i.e., (M', W') is a constrained model.

Similarly, we can construct other logics by applying successively the hybridisation method. Each one of these applications induces a new "layer of hybridisation". The conventions used for $\mathcal{H}^2PL$ can be extended to other levels of hybridisation $\mathcal{H}^nPL$ (for $n > 2$). ○

Example 3.2.4 ($\mathcal{H}FOL$) Applying of the hybridisation method to *FOL* leads to the *first-order hybrid logic* of [Bra05, 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 connectives 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 $\mathcal{H}REL'$ Δ-models (M, W) and (M', W'), a model homomorphism $h : (M, W) \to (M', W')$ 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 con-

dition of Definition 3.1.1 holds directly from the $\mathcal{H}REL$-reduct definition: for any pushout square of signature morphisms in $\mathrm{Sign}^{\mathcal{H}REL}$

$$\begin{tikzcd} \Delta \arrow[r, "\varphi_1"] \arrow[d, "\varphi_2"'] & \Delta_1 \arrow[d, "\theta_1"] \\ \Delta_2 \arrow[r, "\theta_2"'] & \Delta' \end{tikzcd}$$

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

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

We take the amalgamation $(M', W') = (M_1, W_1) \otimes (M_2, W_2)$ according to Theorem 3.1.1. Therefore, for any $w \in W_k$, $(M_k)_w = \mathrm{Mod}^{REL}(\theta_{k\mathrm{Sign}})(M_w)$ 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
$\mathrm{Mod}^{REL}(\theta_{k\mathrm{Sign}})(M_{w'})) = (M_k)_{w'}$), 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".

∘

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.
- $\mathsf{Sen}^{FOLR}(S, S_0, F, F_0, P, P_0)$ consists of those sentences in $\mathsf{Sen}^{FOL}(S, F, P)$ that contains only quantifiers over rigid variables,
- $\mathsf{Mod}^{FOLR}(S, S_0, F, F_0, P, P_0) = \mathsf{Mod}^{FOL}(S, F, P)$, and
- the satisfaction relation in *FOLR* is induced canonically from *FOL*, i.e. $\models^{FOLR}_{(S,S_0,F,F_0,P,P_0)} = \models^{FOL}_{(S,F,P)}$.

We let $\mathcal{H}FOLR$ be the hybridisation of *FOLR* with quantifications over nominal, modalities, and *rigid FOL* variables.

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

$$\begin{array}{ccc} (\Sigma, \mathsf{Nom}, \Lambda) & \xrightarrow{\varphi_1} & (\Sigma_1, \mathsf{Nom}_1, \Lambda_1) \\ {\scriptstyle \varphi_2}\downarrow & & \downarrow{\scriptstyle \theta_1} \\ (\Sigma_2, \mathsf{Nom}_2, \Lambda_2) & \xrightarrow[\theta_2]{} & (\Sigma', \mathsf{Nom}', \Lambda') \end{array}$$

let us consider a constrained $(\Sigma_k, \mathsf{Nom}_k, \Lambda_k)$-model (M_k, W_k) for each $k \in \{1, 2\}$ such that $\mathsf{Mod}^C(\varphi_1)(M_1, W_1) = \mathsf{Mod}^C(\varphi_2)(M_2, W_2)$.

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 Σ' and any $i, j \in |W'|$. By Proposition 3.1.1 we have that

$$\begin{tikzcd} \Sigma \arrow[r, "(\varphi_1)_{Sign}"] \arrow[d, "(\varphi_2)_{Sign}"'] & \Sigma_1 \arrow[d, "(\theta_1)_{Sign}"] \\ \Sigma_2 \arrow[r, "(\theta_2)_{Sign}"'] & \Sigma' \end{tikzcd}$$

is a pushout square of *FOLR* signature morphisms. Note that the set of rigid symbols in Σ' is the union of the translations of the rigid symbols from both Σ_1 and Σ_2 through $(\theta_1)_{Sign}$ and $(\theta_2)_{Sign}$. This means that there exists $k \in \{1, 2\}$ and x_k rigid symbol of Σ_k such that $x = \theta_k(x_k)$. It follows that $(M'_i)_x = ((M_k)_i)_{x_k}$ and $(M'_j)_x = ((M_k)_j)_{x_k}$, 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 $\mathcal{H}FOLR$ with models with "increasing domains", constraining the models where, for any $w, w' \in W$, $\lambda \in \Lambda$, $(w, w') \in R_\lambda$, $(M_w)_s \subseteq (M_w)_{s'}$, 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 Δ_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 *MFOL* 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

$(S, S_0, \mathsf{TF}, \mathsf{TF}_0, \mathsf{PF}, \mathsf{PF}_0)$ where $(S_0, \mathsf{TF}_0, \mathsf{TF}_0)$ is a sub-signature of 'rigid symbols' for a *PA* signature $(S, \mathsf{TF}, \mathsf{PF})$, 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 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 $\mathrm{Mod}^{\mathsf{C}}$ defined by $(M, W) \in |\mathrm{Mod}^{\mathsf{C}}(\Sigma, \mathrm{Nom}, \Lambda)|$ 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 σ in PF_0, we have that $\mathsf{dom}((M_i)_\sigma) = \mathsf{dom}((M_j)_\sigma)$. The reflection condition for $\mathrm{Mod}^{\mathsf{C}}$ 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, unlikely the base case where each sentence is tagged by a L-value, we have now more structured expressions involving different L-values (e.g. $(\rho, p) \wedge (\rho', p')$).

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 $\mathrm{Mod}^{\mathsf{C}}$ 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.

4

FIRST-ORDER ENCODINGS

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 $\mathcal{H}PL$ 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 $\mathcal{H}FOL2FOL$ (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 $\mathcal{H}$*FOL*2*FOL* to lift $\mathcal{I}$2*FOL* to $\mathcal{H}\mathcal{I}$2*FOL*. Thus, all the signatures and sentences targeted by $\mathcal{I}$2*FOL* become parametric on states and the remaining sentences are treated as in $\mathcal{H}$*FOL*2*FOL*. 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}$2*FOL*.

As discussed in [DM13], 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 [DM13], extends it to presentations, constrained and quantified models. The characterisation of conservativity also appeared in [DM13].

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,
- $[F]_{\underline{ar} \to s} = \begin{cases} F_{\underline{ar}' \to s} & \textit{for any } s \in S, \underline{ar}' \in S^* \textit{ such that } \underline{ar} = (ST)\underline{ar}' \\ \emptyset & \textit{for the other cases;} \end{cases}$
- $[P]_{\underline{ar}} = \begin{cases} P_{\underline{ar}'} & \textit{for any } \underline{ar}' \in S^* \setminus S \textit{ such that } \underline{ar} = (ST)\underline{ar}'; \\ \emptyset, & \textit{for the other cases.} \end{cases}$

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

- $[\varphi]^{st}(ST) = ST$,
- $[\varphi]^{st}(s) = \varphi^{st}(s)$ *for any* $s \in S$,
- $[\varphi]^{op}_{(ST)\underline{ar}' \to s}(\sigma) = \varphi^{op}_{\underline{ar}' \to s}(\sigma)$ *for any* $\sigma \in F_{\underline{ar}' \to s}$, *and*
- $[\varphi]^{rl}_{(ST)\underline{ar}'}(\pi) = \varphi^{rl}_{\underline{ar}'}(\pi)$ *for any* $\pi \in P_{\underline{ar}'}$.

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*

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

is defined by

- $[t = t']^x = ([t]^x = [t']^x)$ *where* $[\sigma(t_1, \ldots, t_n)]^x = \sigma(x, [t_1]^x, \ldots, [t_n]^x)$;
- $[\pi(t)]^x = \pi(x, [t]^x)$;
- $[\rho_1 \star \rho_2]^x = [\rho_1]^x \star [\rho_2]^x$, *for* $\star \in \{\vee, \wedge, \Rightarrow\}$;
- $[\neg\rho]^x = \neg[\rho]^x$;

- $[(\forall Y)\rho]^x = (\forall Y)([\rho]^x)_Y$ *where* $([\rho]^x)_Y$ *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* $D_F = \{D_\sigma \mid \sigma \in F_{\underline{ar}\to s}, \underline{ar} \in S^*, s \in S\}$, *where*

- *For any* $s \in S$, D_s *is a new designated relation symbol with arity* $(ST)s$*;*
- *For any* $\sigma \in F_{s_1 \ldots s_n \to s}$, D_σ *is the Horn sentence*

$$(\forall y)(\forall x_1, \ldots, x_n) \bigwedge_{1 \leqslant i \leqslant n} D_{s_i}(y, x_i) \Rightarrow D_s(y, \sigma(y, x_1, \ldots, x_n))$$

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_σ assures the definability of σ 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* σ *in* F, $(M'|_w)_\sigma(m) = M'_\sigma(w, m)$*;*
- *for each* π *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 $\sigma \in F_{\underline{ar}\to s}$ and each $m \in (M'|_w)_{\underline{ar}}$, $(M'|_w)_\sigma(m) \in (M'|_w)_s$, relies upon the condition $M' \models D_F$.

Notation 4.1.2 *For any* (S, F, P)-*sentence* ρ, *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* ρ. *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* ρ, *and any* $w \in M'_{ST}$, *if* $M' \models V(\rho)$ *then*

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

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

Proof. The proof is by induction on the structure of ρ.

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

$$(M'|_w)_t = (M'^w)_{[t]^x}, \text{ for any term } t \tag{9}$$

which is proved by induction on t as follows:

$$\begin{array}{ll}
 & (M'|_w)_{\sigma(t_1,\ldots,t_n)} \\
= & \quad \{\text{defn of evaluation of terms}\} \\
 & (M'|_w)_\sigma((M'|_w)_{t_1}, \ldots, (M'|_w)_{t_n}) \\
= & \quad \{\text{defn of } M'|_w\} \\
 & M'_\sigma(w, (M'|_w)_{t_1}, \ldots, (M'|_w)_{t_n}) \\
= & \quad \{\text{because } M'^w_x = w \text{ and defn of term evaluation}\} \\
 & M'^w_{\sigma(x,[t_1]^x,\ldots,[t_n]^x)} \\
= & \quad \{\text{defn of } [_]^x\} \\
 & M'^w_{[\sigma(t_1,\ldots,t_n)]^x}
\end{array}$$

2. When ρ is $\pi(t_1, \ldots, t_n)$:

$$M'|_w \models \pi(t_1, \ldots, t_n)$$

$$
\begin{aligned}
\Leftrightarrow \quad & \{\text{ defn of } FOL\text{-satisfaction}\} \\
& ((M'|_w)_{t_1}, \ldots, (M'|_w)_{t_n}) \in (M'|_w)_\pi \\
\Leftrightarrow \quad & \{\text{ defn of } (M'|_w)_\pi \text{ and by (9)}\} \\
& (w, M'^w_{[t_1]^x}, \ldots, M'^w_{[t_n]^x}) \in M'_\pi \\
\Leftrightarrow \quad & \{\text{ because } M'^w_x = w\} \\
& M'^w \models \pi(x, [t_1]^x, \ldots, [t_n]^x)
\end{aligned}
$$

3. When ρ is $\xi_1 \star \xi_2$ for $\star \in \{\wedge, \vee, \Rightarrow\}$ or ρ is $\neg\xi$, the proof reduces to an immediate application of the recursion hypothesis.

4. When ρ is $(\forall Y)\xi$:

   $M'|_w \models (\forall Y)\xi$ iff $M'' \models \xi$ for any $(S, F + x, P)$-expansion M'' of $M'|_w$, and

   $M'^w \models (\forall Y)([\xi]^x)_Y$ iff $N'^w \models ([\xi]^x)_Y$ for any $([S], [F] + Y + x, [P])$-expansion N'^w 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 $([S], [F] + Y + x, [P])$-expansions N'^w of M'^w and $(S, F + Y, P)$-expansions M'' of $M'|_w$ given by $M''_y = N'^w_y$ 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 $([S], [F + Y] + x, [P])$-expansion N''^w of M'^w by $N''^w_y(m) = N'^w_y$ for all $m \in M'_{ST}$ and each $y \in Y$. Furthermore
     $$N''^w \models [\xi]^x \text{ if and only if } N'^w \models ([\xi]^x)_Y. \tag{10}$$
   - $M'' = N''|_w$.

   The induction hypothesis entails $M'' \models \xi$ iff $N''^w \models [\xi]^x$. By (10) it follows that $M'' \models \xi$ iff $N'^w \models ([\xi]^x)_Y$.

□

4.1.2 *The standard translation lifting*

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

- χ_{Nom} are finite extensions, and
- χ_{MS} are identities.

Given any comorphism

$$(\Phi, \alpha, \beta) : \mathcal{I} \to \mathit{FOL}^{\mathrm{pres}}$$

such that $\Phi(\mathcal{D}^{\mathcal{I}}) \subseteq \mathcal{D}^{\mathit{FOL}}$, we define another comorphism

$$(\Phi'^{C}, \alpha', \beta'^{C}) : (\mathrm{Sign}^{\mathcal{HI}}, \mathrm{Sen}^{\mathcal{HI}}, \mathrm{Mod}^{C}, \models) \to \mathit{FOL}^{\mathrm{pres}}$$

in two steps:

1. We define a functor $\Phi' : \mathrm{Sign}^{\mathcal{HI}} \to \mathrm{Sign}^{\mathit{FOL}^{\mathrm{pres}}}$, a natural transformation $\alpha' : \mathrm{Sen}^{\mathcal{HI}} \Rightarrow \Phi'; \mathrm{Sen}^{\mathit{FOL}^{\mathrm{pres}}}$, and a natural transformation $\beta' : \Phi'^{op}; \mathrm{Mod}^{\mathit{FOL}^{\mathrm{pres}}} \Rightarrow \mathrm{Mod}^{\mathcal{HI}}$.
2. We extend the definitions of Φ' and β' to Φ'^{C} and β'^{C} respectively. Then we prove the Satisfaction Condition for $(\Phi'^{C}, \alpha', \beta'^{C})$.

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 $\mathcal{HI}$ signature $(\Sigma, \mathrm{Nom}, \Lambda)$, let*

$$\Phi'(\Sigma, \mathrm{Nom}, \Lambda) = \\ ([S_\Sigma], [F_\Sigma] + \overline{\mathrm{Nom}}, (D_s)_{s \in S_\Sigma} + [P_\Sigma] + \overline{\Lambda}, \overline{\Gamma_\Sigma} \cup D_{F_\Sigma})$$

where

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

- $(\overline{\mathrm{Nom}})_{\underline{\mathrm{ar}}\to s} = \begin{cases} \mathrm{Nom} & \textit{when } \underline{\mathrm{ar}} = \emptyset, s = \mathrm{ST}, \\ \emptyset & \textit{for the other cases}; \end{cases}$
- $(\overline{\Lambda})_{\underline{\mathrm{ar}}} = \begin{cases} \Lambda_n & \textit{when } \underline{\mathrm{ar}} = (\mathrm{ST})^n, n \in \mathbb{N} \\ \emptyset & \textit{for the other cases}; \end{cases}$
- $\overline{\Gamma_\Sigma} = \{\forall x\ [\gamma]^x \mid \gamma \in \Gamma_\Sigma\} \cup V(\Gamma_\Sigma)$.

The translation of signature morphisms by Φ' extend translations via Φ as follows:

Definition 4.1.5 (Translation of the signature morphisms) *For any $\mathcal{HI}$ signature morphism*

$$\varphi = (\varphi_{\mathrm{Sign}}, \varphi_{\mathrm{Nom}}, \varphi_{\mathrm{MS}}) : \ (\Sigma_1, \mathrm{Nom}_1, \Lambda_1) \to (\Sigma_2, \mathrm{Nom}_2, \Lambda_2)$$

the FOL^{pres} signature morphism $\Phi'(\varphi)$ is defined as follows:

- $\Phi'(\varphi)_{\mathrm{Sign}}$ *is the extension of*

  $$[\Phi(\varphi_{\mathrm{Sign}})] : \ ([S_{\Sigma_1}], [F_{\Sigma_1}], [P_{\Sigma_1}]) \to ([S_{\Sigma_2}], [F_{\Sigma_2}], [P_{\Sigma_2}])$$

  given by

  - $\Phi'(\varphi)^{\mathrm{op}}_{\mathrm{Sign}}(D_s) = D_{\Phi(\varphi_{\mathrm{Sign}})^{\mathrm{st}}(s)}$ *for each sort* $s \in S_{\Sigma_1}$
- $\Phi'(\varphi)_{\mathrm{Nom}}(n) = \varphi_{\mathrm{Nom}}(n)$ *for each* $n \in \mathrm{Nom}_1$, *and*
- $\Phi'(\varphi)_{\mathrm{MS}}(\lambda) = \varphi_{\mathrm{MS}}(\lambda)$ *for each* $\lambda \in \Lambda_1$.

Fact 4.1.1 $\Phi'(\varphi)$ *of Definition 4.1.5 is a presentation morphism*

$$\Phi'(\Sigma_1, \mathrm{Nom}_1, \Lambda_1) \to \Phi'(\Sigma_2, \mathrm{Nom}_2, \Lambda_2).$$

Next definition establishes the translation of the sentences. The key idea is to use the principle underlying the standard translation $\mathcal{H}$*FOL2FOL* to lift $\mathcal{I}$*2FOL* into $\mathcal{HI}$*2FOL*. For this purpose, the translation of the basic $\mathcal{I}$-sentences by $\mathcal{I}$*2FOL* is followed by the application of the state-parametrisation procedure, extending the standard translation principle in all the hybrid features (nominals and modalities) accordantly:

Definition 4.1.6 (Translation of sentences) *The translation of the sentences is given by* $\alpha'_{(\Sigma,\mathrm{Nom},\Lambda)}(\rho) = (\forall x)\alpha'^{x}_{(\Sigma,\mathrm{Nom},\Lambda)}(\rho)$, *where*

$$\begin{aligned}&\alpha'^{x}_{(\Sigma,\mathrm{Nom},\Lambda)} : \ \mathrm{Sen}^{\mathcal{H}\mathcal{I}}(\Sigma,\mathrm{Nom},\Lambda)\\ &\quad \to \mathrm{Sen}^{FOL}([S_\Sigma],[F_\Sigma]+\overline{\mathrm{Nom}}+x,(D_s)_{s\in S}+[P_\Sigma]+\overline{\Lambda})\end{aligned}$$

with x *a constant of sort* ST, *is defined*

- *for each* $\rho \in \mathrm{Sen}^{\mathcal{I}}(\Sigma)$, $\alpha'^{x}(\rho) = [\alpha_\Sigma(\rho)]^x$
- $\alpha'^{x}(i) = (i = x)$, $i \in \mathrm{Nom}$;
- $\alpha'^{x}(@_i\rho) = \alpha'^{i}(\rho)$;
- $\alpha'^{x}([\lambda](\rho_1,\ldots,\rho_n)) =$
  $\forall y_1,\ldots,y_n \left(\lambda(x,y_1,\ldots,y_n) \Rightarrow \bigvee_{1\leqslant i\leqslant n} \alpha'^{y_i}(\rho_i)\right)$;
- $\alpha'^{x}(\langle\lambda\rangle(\rho_1,\ldots,\rho_n)) =$
  $\exists y_1,\ldots,y_n \left(\lambda(x,y_1,\ldots,y_n) \wedge \bigwedge_{1\leqslant i\leqslant n} \alpha'^{y_i}(\rho_i)\right)$;
- $\alpha'^{x}(\rho_1 \star \rho_2) = \alpha'^{x}(\rho_1) \star \alpha'^{x}(\rho_2)$, $\star \in \{\vee,\wedge,\Rightarrow\}$;
- $\alpha'^{x}(\neg\rho) = \neg\alpha'^{x}(\rho)$;
- $\alpha'^{x}_{\Delta}((\forall\chi)\rho) = (\forall i)(\forall Y)(\alpha'^{x}_{\Delta'+i}(\rho))_Y$, *where*
  - $\chi : \ \Delta \to \Delta' + i$ *with* $\Delta = (\Sigma,\mathrm{Nom},\Lambda)$, $\Delta' = (\Sigma',\mathrm{Nom},\Lambda)$,
  - $\Phi(\Sigma') = \Phi(\Sigma) + Y$ *(since by hypothesis* $\Phi(\chi_{\mathrm{Sign}}) \in \mathcal{D}^{FOL}$*), and*
  - $(\alpha'^{x}_{\Delta'+i}(\rho))_Y$ *is the result of replacing in* $\alpha'^{x}_{\Delta'+i}(\rho)$ *all occurrences of* $y(i)$ *by* y *for each* y *in* Y.

From the naturality of α it follows:

Fact 4.1.2 α' *is natural transformation.*

We turn now to the translation of models. The idea of recovering a model $(M,W) \in \mathrm{Mod}^{\mathcal{I}}(\Delta)$ from a model $M \in \mathrm{Mod}^{FOL^{\mathrm{pres}}}(\Phi'(\Delta))$ is rather natural.The relational part of the model is directly extracted from the {ST}-component of M (with the respective constants and predicates). Then, the (local) $\mathcal{I}$-models are built, state by state, applying the base models translation of $\mathcal{I}2FOL$ to the respective slice. Formally:

Definition 4.1.7 (Translation of models) *For any $\mathcal{HI}$ signature $(\Sigma, \mathrm{Nom}, \Lambda)$ and any $\Phi'(\Sigma, \mathrm{Nom}, \Lambda)$-model M', define*

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

where

- *W is the reduct $M'\restriction_{(\{ST\},\overline{\mathrm{Nom}},\overline{\Lambda})}$, i.e. $|W| = M'_{ST}$, $W_i = M'_i$ for each $i \in \mathrm{Nom}$, and $W_\lambda = M'_\lambda$, for each λ in Λ, and*
- *$M : |W| \to |\mathrm{Mod}^{\mathcal{I}}(\Sigma)|$ is defined, for each $w \in |W|$, by $M_w = \beta_\Sigma(M'|_w)$, where $M'|_w$ abbreviates $(M'\restriction_{([S_\Sigma],[F_\Sigma],[P_\Sigma])})|_w$.*

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 $M' \models V(\Gamma_\Sigma) \cup D_{F_\Sigma}$ 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 $M' \models \overline{\Gamma_\Sigma}$, we have $M' \models (\forall x)[\gamma]^x$, for each $\gamma \in \Gamma_\Sigma$. Hence $M'^w \models [\gamma]^x$ for each $\gamma \in \Gamma_\Sigma$. □

Having defined the functor Φ' and the natural transformations α' and β', 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 $C : \mathrm{Sign}^{\mathcal{HI}} \to \mathrm{Sign}^{FOL^{pres}}$ such that the constraints of $\mathrm{Mod}^C(\Delta)$ could be taken into account in the axiomatization of $C(\Delta)$. Given this functor, we extend Φ' and β' to Φ'^C and β'^C accordantly.Thus,

Definition 4.1.8 *A functor C is* compatible with Φ' *when the diagram below commutes*

$$\begin{tikzcd}
\mathrm{Sign}^{\mathcal{HI}} \arrow[r, "\Phi'"] \arrow[d, "C"'] & \mathrm{Sign}^{FOL^{pres}} \arrow[d, "U"] \\
\mathrm{Sign}^{FOL^{pres}} \arrow[r, "U"'] & \mathrm{Sign}^{FOL}
\end{tikzcd}$$

where U denotes the forgetful functor.

For C compatible with Φ', let

- Φ'^{C} *denote the functor that represents the componentwise union of the corresponding presentations, i.e.* $\Phi'^{C}(\Delta)$ *is the union of* $\Phi'(\Delta)$ *and* $C(\Delta)$*, and*
- $\beta'^{C} : \Phi'^{C}; \mathrm{Mod}^{FOL^{\mathrm{pres}}} \Rightarrow \mathrm{Mod}^{\mathcal{HI}}$ *denote the corresponding (componentwise) restriction of* β'.

Next theorem characterises the conditions on C for which the intended comorphism $\mathcal{HI}2FOL$ 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 Mod^{C};
- 2 assures the definability of the *FOL*-quantifications resulting from the translation of basic sentences;
- 3 assures that the adequacy of χ_{Sign} is not lost on the restriction of β'^{C}; and
- 4 imposes that quantifications of χ_{Sign} are in fact first-order quantifications.

Hence,

Theorem 4.1.1 *Assume a functor* C *compatible with* Φ' *such that*

1. *For each* $\mathcal{HI}$*-signature* Δ *and each* $M' \in |\mathrm{Mod}^{FOL^{\mathrm{pres}}}(\Phi'^{C}(\Delta))|$, $\beta'_{\Delta}(M') \in |\mathrm{Mod}^{C}(\Delta)|$.
2. *For any* $\mathcal{HI}$*-signature* $\Delta = (\Sigma, \mathrm{Nom}, \Lambda)$ *and for any* Σ*-sentence* ξ,
$$\Phi'^{C}(\Delta) \models V(\alpha_{\Sigma}(\xi)). \tag{11}$$
3. *Each signature morphism* $\chi \in \mathcal{D}^{\mathcal{HI}}$ *with* $\chi_{\mathrm{Nom}} = 1_{\mathrm{Nom}}$ *is adequate for* β'^{C}.
4. *For each signature morphism* $(\chi : \Delta \to \Delta') \in \mathcal{D}^{\mathcal{HI}}$ *such that* $\Phi(\chi_{\mathrm{Sign}}) : \Phi(\Sigma) \to \Phi(\Sigma) + Y$ *for* Y *set of FOL variables,*
$$\Phi'^{C}(\Delta') \models \{(\forall z_1, z_2) y(z_1) = y(z_2) \mid y \in Y\}. \tag{12}$$

Then, for any $\Delta = (\Sigma, \mathrm{Nom}, \Lambda) \in |\mathrm{Sign}^{\mathcal{HJ}}|$, $\rho \in \mathrm{Sen}^{\mathcal{HJ}}(\Delta)$, $M' \in |\mathrm{Mod}^{FOL^{\mathrm{pres}}}(\Phi'^{C}(\Delta))|$ *and* $w \in M'_{ST}$,

$$\beta'^{C}_{\Delta}(M') \models^{w}_{\Delta} \rho \ \textit{ if and only if } \ M'^{w} \models_{\Phi'(\Delta)+x} \alpha'^{x}_{\Delta}(\rho), \tag{13}$$

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

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

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

$$\begin{array}{ll}
 & (M, W) \models^{w}_{\Delta} i \\
\Leftrightarrow & \quad \{\text{ defn. of } \models^{w}_{\Delta}\} \\
 & W_i = w \\
\Leftrightarrow & \quad \{\text{ defn } \beta' \text{ and of } M'^{w}\ \} \\
 & M'_i (= M'^{w}_{i}) = M'^{w}_{x} \\
\Leftrightarrow & \quad \{\text{ defn } FOL^{\mathrm{pres}}\text{-satisfaction}\} \\
 & M'^{w} \models_{\Phi'(\Delta)+x} i = x \\
\Leftrightarrow & \quad \{\text{ defn of } \alpha'^{x}\} \\
 & M'^{w} \models_{\Phi'(\Delta)+x} \alpha'^{x}(i)
\end{array}$$

2. If $\rho \in \mathrm{Sen}^{\mathcal{J}}(\Sigma)$:

$$\begin{array}{ll}
 & (M, W) \models^{w}_{\Delta} \rho \\
\Leftrightarrow & \quad \{\text{ defn. of } \models^{w}_{\Delta}\} \\
 & M_w \models^{\mathcal{J}} \rho \\
\Leftrightarrow & \quad \{\text{ defn. of } \beta'\} \\
 & \beta_{\Sigma}(M'|_w) \models_{\Sigma} \rho \\
\Leftrightarrow & \quad \{\text{ by the satisfaction condition of } (\Phi, \alpha, \beta)\} \\
 & M'|_w \models_{\Phi(\Sigma)} \alpha_{\Sigma}(\rho)
\end{array}$$

$$
\begin{array}{ll}
\Leftrightarrow & \{\text{ by (11) and Lemma 4.1.1}\} \\
& M'^{w} \models_{\Phi'(\Delta)+x} [\alpha_{\Sigma}(\rho)]^{x} \\
\Leftrightarrow & \{\text{ defn of } \alpha'^{x}\} \\
& M'^{w} \models_{\Phi'(\Delta)+x} \alpha'^{x}(\rho) \\
\Leftrightarrow & \{\text{ by (11) and Lemma 4.1.1}\} \\
& M'^{w} \models_{\Phi'(\Delta)+x} [\alpha_{\Sigma}(\rho)]^{x} \\
\Leftrightarrow & \{\text{ defn } \alpha'^{x}\} \\
& M'^{w} \models_{\Phi'(\Delta)+x} \alpha'^{x}(\rho)
\end{array}
$$

3. If $\rho = @_i\xi$:

$$
\begin{array}{ll}
& (M, W) \models^{w}_{\Delta} @_i\xi \\
\Leftrightarrow & \{\text{ defn. o } \models_{\Delta}\} \\
& (M, W) \models^{W_i}_{\Delta} \xi \\
\Leftrightarrow & \{\text{ I.H.}\} \\
& M'^{W_i} \models_{\Phi'(\Delta)+x} \alpha'^{x}(\xi) \\
\Leftrightarrow & \{\text{ since } M'^{W_i}_{x} = W_i = M'_i\} \\
& M' \models_{\Phi'(\Delta)} \alpha'^{i}(\xi) \\
\Leftrightarrow & \{\text{ satisfaction condition in } FOL\} \\
& M'^{w} \models_{\Phi'(\Delta)+x} \alpha'^{i}(\xi) \\
\Leftrightarrow & \{\text{ defn of } \alpha'^{x}\} \\
& M'^{w} \models_{\Phi'(\Delta)+x} \alpha'^{x}(@_i\xi)
\end{array}
$$

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

$$
\begin{array}{ll}
& (M, W) \models^{w}_{\Delta} [\lambda](\xi_1, \ldots, \xi_n) \\
\Leftrightarrow & \{\text{ defn. of } \models_{\Delta}\}
\end{array}
$$

for any $(w, w_1, \ldots, w_n) \in W_\lambda$ there exists $1 \leqslant i \leqslant n$ such that $(M, W) \models^{w_i} \xi_i$

$\Leftrightarrow$ $\{$ I.H.$\}$

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

$\Leftrightarrow$ $\{$ defn of $\models_{\Phi'(\Delta)}\}$

for all $w_1, \ldots, w_n$

$M'^{ww_1 \ldots w_n} \models_{\Phi'(\Delta)+x+y_1+\cdots+y_n} \lambda(x, y_1, \ldots, y_n)$

$\Rightarrow \bigvee_{1 \leqslant i \leqslant n} \alpha'^{y_i}(\xi_i)$

$\Leftrightarrow$ $\{$ Rule of Generalization in *FOL*$\}$

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

$\Leftrightarrow$ $\{$ defn. of $\alpha'^{x}\}$

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

5. If $\rho = \xi \vee \xi'$:

$(M, W) \models^{w}_{\Delta} \xi \vee \xi'$

$\Leftrightarrow$ $\{$ defn. of $\models_{\Delta}\}$

$(M, W) \models^{w}_{\Delta} \xi$ or $(M, W) \models^{w}_{\Delta} \xi'$

$\Leftrightarrow$ $\{$ I.H.$\}$

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

$\Leftrightarrow$ $\{$ defn of $\models^{FOL^{\text{pres}}}\}$

$M'^{w} \models_{\Phi'(\Delta)+x} \alpha'^{x}(\xi) \vee \alpha'^{x}(\xi')$

$\Leftrightarrow$ $\{$ defn of $\alpha'^{x}\}$

$M'^{w} \models_{\Phi'(\Delta)+x} \alpha'^{x}(\xi \vee \xi')$

The proofs for cases $\rho = \xi \wedge \xi'$, $\rho = \xi \Rightarrow \xi'$, $\rho = \neg\xi$, etc. are analogous.

6. If $\rho = (\forall\chi : \ \Delta \to \Delta' + i)\xi$:

$$
\begin{array}{ll}
 & (M, W) \models^{w}_{\Delta} (\forall\chi)\xi \\
\Leftrightarrow & \quad \{ \text{ defn. of } \models_{\Delta} \} \\
 & (N, W') \models^{w}_{\Delta'+i} \xi \\
 & \text{for any } \chi\text{-expansion } (N, W') \text{ of } (M, W) \\
\Leftrightarrow & \quad \{ \text{ by Lemma 4.1.3 (above) and induction hypothesis} \} \\
 & N'^{w} \models_{\Phi'(\Delta')+i+\chi} \alpha'^{x}(\xi) \\
 & \text{for any } \Phi'^{C}(\chi)\text{-expansion } N' \text{ of } M' \\
\Leftrightarrow & \quad \{ \text{ by Fact 4.1.3} \} \\
 & (N')^{w}_{Y} \models_{\Phi'(\Delta)+Y+i+\chi} (\alpha'^{x}(\xi))_{Y} \\
 & \text{for any } (N')_{Y} \text{ a } \Phi'(\Delta) + Y + i\text{-expansion of } M' \\
\Leftrightarrow & \quad \{ \models_{FOL} \} \\
 & M'^{w} \models_{\Phi'(\Delta)+\chi} (\forall i)(\forall Y)(\alpha'^{x}(\xi))_{Y} \\
\Leftrightarrow & \quad \{ \text{ defn of } \alpha'^{x} \} \\
 & M'^{w} \models_{\Phi'(\Delta)+\chi} \alpha'^{x}((\forall\chi)\xi)
\end{array}
$$

□

Fact 4.1.3 *Let us denote by* Y *the block of variables such that*

$$\Phi(\chi_{\mathrm{Sign}}) : \ \Phi(\Sigma) \to \Phi(\Sigma') = \Phi(\Sigma) + Y.$$

Note that by (12) the model reduct $\mathrm{Mod}^{FOL^{\mathrm{pres}}}(\Phi'^{C}(\Delta + i) + Y) \to \mathrm{Mod}^{FOL^{\mathrm{pres}}}(\Phi'^{C}(\Delta' + i))$ *corresponding to the second order substitution* $\Phi'(\Delta' + i) \to \Phi'(\Delta + i) + Y$ *that maps each term* $y(i)$ *(with* y *in* Y *and* $i \in \overline{\mathrm{Nom}}$*) to* y*, is a bijection. Let* $(\_)_{Y}$ *denote the inverse of this bijection.*

Lemma 4.1.3 χ *is adequate for* β'^{C}.

Proof. It is very easy to see that any signature with a nominal is adequate, hence the signature extension $\Delta' \to \Delta' + i$ is adequate for β'^{C}. By hypothesis we know that $(\chi_{\mathrm{Sign}}, 1_{\mathrm{Nom}}, 1_{\wedge})$ is adequate for β'^{C}, hence by Fact 2.3.1 we get that χ, which is the composition of these signature morphisms, is adequate for β'^{C} too. □

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*

$$(\mathrm{Sign}^{\mathcal{HI}}, \mathrm{Sen}^{\mathcal{HI}}, \mathrm{Mod}^{C}, \models) \to \mathit{FOL}^{\mathrm{pres}},$$

i.e. for any $\Delta \in |\mathrm{Sign}^{\mathcal{HI}}|$, $\rho \in \mathrm{Sen}^{\mathcal{HI}}(\Delta)$ and $M' \in |\mathrm{Mod}^{\mathit{FOL}^{\mathrm{pres}}}((\Phi'^{C})(\Delta))|$,

$$\beta'^{C}_{\Delta}(M') \models_{\Delta} \rho \textit{ if and only if } M' \models_{\Phi'^{C}(\Delta)} \alpha'_{\Delta}(\rho).$$

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 (Φ, α, β) is the canonical embedding of *PL* into *FOL* presented on Example 2.3.14. This means the D's and the Γ's are empty. The quantification space for the hybridisation consists of extensions with nominal variables. Hence, for any signature $(\mathrm{Prop}, \mathrm{Nom})$,

$$\Phi'(\mathrm{Prop}, \mathrm{Nom}) = (\{ST\}, \overline{\mathrm{Nom}}, [\mathrm{Prop}] + \lambda).$$

Moreover, for any sentence $\rho \in \mathrm{Sign}^{\mathcal{H}PL}(\mathrm{Prop}, \mathrm{Nom})$,

$$\alpha'_{(\mathrm{Prop},\mathrm{Nom})}(\rho) = (\forall x)\alpha^{x}_{(\mathrm{Prop},\mathrm{Nom})}(\rho),$$

where

- for each $\rho \in \mathrm{Sen}^{\mathcal{H}PL}(\mathrm{Prop})$, $\alpha'^{x}_{(\mathrm{Prop},\mathrm{Nom})}(\rho) = [\alpha_{\mathrm{Prop}}(\rho)]^{x}$
- $\alpha'^{x}_{(\mathrm{Prop},\mathrm{Nom})}(i) = (i = x)$, $i \in \mathrm{Nom}$;
- $\alpha'^{x}_{(\mathrm{Prop},\mathrm{Nom})}(@_i\rho) = \alpha'^{i}_{(\mathrm{Prop},\mathrm{Nom})}(\rho)$;

- $\alpha'^{x}_{(\mathrm{Prop},\mathrm{Nom})}([\lambda]\rho = (\forall y)\,\lambda(x,y) \Rightarrow \alpha'^{y}_{(\mathrm{Prop},\mathrm{Nom})}(\rho)$;
- $\alpha'^{x}_{(\mathrm{Prop},\mathrm{Nom})}(\rho_1 \star \rho_2) = \alpha'^{x}_{(\mathrm{Prop},\mathrm{Nom})}(\rho_1) \star \alpha'^{x}_{(\mathrm{Prop},\mathrm{Nom})}(\rho_2), \star \in \{\vee, \wedge, \Rightarrow\}$;
- $\alpha'^{x}_{(\mathrm{Prop},\mathrm{Nom})}(\neg\rho) = \neg\alpha'^{x}_{(\mathrm{Prop},\mathrm{Nom})}(\rho)$;
- $\alpha'^{x}_{(\mathrm{Prop},\mathrm{Nom})}((\forall i)\rho) = (\forall i)\alpha'^{x}_{(\mathrm{Prop},\mathrm{Nom}+\{i\})}(\rho)$.

Observe that, as expected, the transformation α' coincides with the so-called "standard translation" for the propositional hybrid logic (e.g. [Bla00, Bra10]).

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

- $W = M'\restriction_{(\{ST\},\overline{\mathrm{Nom}},\lambda)}$;
- for any $w \in W$,
  $M_w = \beta_{\mathrm{Prop}}((M'\restriction_{[\mathrm{prop}]})|_w) = (M'\restriction_{[\mathrm{prop}]})|_w$, i.e., for any $p \in \mathrm{Prop}$, $(M_w)_p$ 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_λ is reflexive. The functor C is such that each $C(P,\mathrm{Nom})$ is the presentation containing the sentence $(\forall x)\lambda(x,x)$. Note that Φ'^{C} maps any signature (P,Nom) to the *FOL*-presentation $((\{ST\},\overline{\mathrm{Nom}},[P]+\lambda),(\forall x)\lambda(x,x))$. Again, conditions (11) and (12) of Theorem 4.1.1 are vacuously satisfied, as well as the adequacy condition for β'^{C} (of the same theorem). ○

Example 4.2.2 ($\mathcal{H}$*TRM2FOL*) Let us consider the comorphism *TRM2FOL* of Example 2.3.13. Then the method yields

$$\Phi'(*,\mathrm{Nom},\Lambda) = \big(\{St\},\overline{\mathrm{Nom}},p_* : ST+\overline{\Lambda}),\emptyset\big).$$

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

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

where

- $\alpha'^{x}(i) = (i = x)$, $i \in \mathrm{Nom}$;
- $\alpha'^{x}(@_i\rho) = \alpha'^{i}(\rho)$;
- $\alpha'^{x}([\lambda](\rho_1, \ldots, \rho_n)) =$
  $\forall y_1, \ldots, y_n \left(\lambda(x, y_1, \ldots, y_n) \Rightarrow \bigvee_{1 \leqslant i \leqslant n} \alpha'^{y_i}(\rho_i)\right)$;
- $\alpha'^{x}(\langle\lambda\rangle(\rho_1, \ldots, \rho_n)) =$
  $\exists y_1, \ldots, y_n \left(\lambda(x, y_1, \ldots, y_n) \wedge \bigwedge_{1 \leqslant i \leqslant n} \alpha'^{y_i}(\rho_i)\right)$;
- $\alpha'^{x}(\rho_1 \star \rho_2) = \alpha'^{x}(\rho_1) \star \alpha'^{x}(\rho_2)$, $\star \in \{\vee, \wedge, \Rightarrow\}$;
- $\alpha'^{x}(\neg\rho) = \neg\alpha'^{x}(\rho)$;

The models translation, $M' \in \mathrm{Mod}^{\mathcal{H}TRM}(\Phi'(*, \mathrm{Nom}, \Lambda))$,

$$\beta'_{(\mathrm{Prop},\mathrm{Nom})}(M') = (M, W)$$

is obtained as follows:

- $W = M'\restriction_{(\{\mathrm{ST}\},\overline{\mathrm{Nom}},\lambda)}$;
- for any $w \in W$, $M_w = *$

Since we have no D_s neither Γ 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 ($\mathcal{H}^2PL2FOL$) Let us now consider the case of $\mathcal{H}^2PL$ presented in Example 3.2.3. We take, as base comorphism, the comorphism between the free hybridisation of *PL* into *FOL* Example 4.2.1. Hence, we have $\Phi(\mathrm{Prop}, \mathrm{Nom}') = (\{\mathrm{ST}'\}, \overline{\mathrm{Nom}'}, [\mathrm{Prop}] + \{\overline{\lambda}\})$. We denote the sort of states with a prime, ST', in order to distinguish it from the sort ST introduced in the hybridisation process.

Thus, $\Phi'((\mathrm{Prop}, \mathrm{Nom}'), \mathrm{Nom}, \Lambda) =$

$$([\{\mathrm{ST}'\}], [\overline{\mathrm{Nom}'}] + \overline{\mathrm{Nom}}, D_{\mathrm{ST}'} + [[\mathrm{Prop}] + \{\overline{\lambda}\}], D_{[\overline{\mathrm{Nom}'}]+\overline{\mathrm{Nom}}}).$$

As expected we have now a sort of states for each level of hybridisation $[\{\mathrm{ST}'\}] = \{\mathrm{ST}, \mathrm{ST}'\}$. Predicate $D_{\mathrm{ST}'}$ plays the role of a "sub-state-relation". A nominal i of the basis signature has now a different interpretation for each "super-state" of ST (by $i : \mathrm{ST} \to \mathrm{ST}'(\in [\overline{\mathrm{Nom}'}])$) and similarly for the base modality λ (by a predicate $\lambda : \mathrm{ST} \times \mathrm{ST}' \times \mathrm{ST}'$).

In order to get condition (11) of Theorem 4.1.1 fulfilled, since for any $\rho \in \mathrm{Sen}^{\mathcal{H}PL}(\mathrm{Prop}, \mathrm{Nom}')$, $\alpha_{(\mathrm{Prop},\mathrm{Nom})}(\rho)$ is ST'-universally quantified, we have to take

$$C((\mathrm{Prop}, \mathrm{Nom}'), \mathrm{Nom}, \Lambda) = \{(\forall x, y) D_{\mathrm{ST}'}(x, y)\}. \tag{14}$$

This implies directly $C((\mathrm{Prop}, \mathrm{Nom}'), \mathrm{Nom}, \Lambda) \models D_{\overline{[\mathrm{Nom}']}+\overline{\mathrm{Nom}}}$. Since the hybridisation is free, the adequacy condition for β'^{C} of Theorem 4.1.1 trivially holds. Moreover, because we admit, at the hybridisation level, quantification over the nominals, condition (12) of Theorem 4.1.1 is vacuously satisfied. Thus

$$\Phi'^{C}((\mathrm{Prop}, \mathrm{Nom}'), \mathrm{Nom}, \Lambda) =$$

$$\big(([\{\mathrm{ST}'\}], \overline{[\mathrm{Nom}']} + \overline{\mathrm{Nom}}, D_{\mathrm{ST}'} + [[\mathrm{Prop}] + \overline{\{\lambda\}}]), (\forall x, y) D_{\mathrm{ST}'}(x, y)\big). \tag{15}$$

Regarding the constraint version $\mathcal{H}^2PL'$, the sharing of the substates universe can be specified by $(\forall x, y, z) D_{\mathrm{ST}'}(x, z) \Leftrightarrow D_{\mathrm{ST}'}(y, z)$. Again, since it is a consequence of (14), $\Phi'^{C}((\mathrm{Prop}, \mathrm{Nom}'), \mathrm{Nom}, \Lambda)$ is still defined as in (15).

∘

Example 4.2.4 Let us now consider the free hybridisation of *FOL* with only quantification over nominal variables. The base comorphism (Φ, α, β) is the identity, hence all Γ's are empty. Hence,

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

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

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

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

$$\Phi'^{C}((S, F, P), \mathrm{Nom}, \Lambda) =$$

$$([S], [F] + \mathrm{Nom}, (D_s)_{s \in S} + [P] + \overline{\Lambda}, \{(\forall x, y) D_s(x, y) \mid s \in S\}).$$

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 β'^{C} (of the same theorem).

The variant of this example in which the base institution is the quantifier-free fragment of *FOL* rather that the whole of *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

$$C((S, F, P), Nom, \Lambda) \models \{(\forall z_1, z_2) y(z_1) = y(z_2) \mid y \in F_{\to s}, s \in S\}.$$

However because the hybridisation is free there is no way to get the adequacy condition for β'^{C}. Hence in this case it is not possible to build the encoding comorphism.

Example 4.2.5 When encoding $\mathcal{H}REL'$ (of Example 3.2.5) the base comorphism (Φ, α, β) is the canonical embedding of *REL* into *FOL* determined by the embedding of *REL* signatures as *FOL* signatures. Hence all Γ's are empty. Thus,

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

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: $\Phi'^{C}((C, P), Nom, \Lambda) =$
$((\{ST, \star\}, [C] + Nom, \{D_\star\} + [P] + \overline{\Lambda}), \{(\forall x, y) D_\star(x, y)\} \cup$
$\{(\forall x, y)\sigma(x) = \sigma(y) \mid \sigma \in C\})$. It remains to check the adequacy condition for β'^{C}, which is a very easy task. Let Δ denote the $\mathcal{H}REL$ signature $((C, P), Nom, \Lambda)$. For any block Y of variables for the *REL* signature (C, P), any $\mathcal{H}REL'$-model (N, W) for $\Delta + Y$, and any $\Phi'^{C}(\Delta)$-model M' such that $(N, W)\restriction_\Delta = \beta'(M')$,

$$\begin{tikzcd}
Mod^{\mathcal{H}REL'}(\Delta) & Mod^{FOL^{pres}}(\Phi'^{C}(\Delta)) \arrow[l, "\beta'^{C}_{\Delta}"'] \\
Mod^{\mathcal{H}REL'}(\Delta + Y) \arrow[u] & Mod^{FOL^{pres}}(\Phi'^{C}(\Delta + Y)) \arrow[l, "\beta'^{C}_{\Delta+Y}"] \arrow[u]
\end{tikzcd}$$

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

$$(\forall x, y, z)\big[\lambda(x, y) \Rightarrow ((D_\star(x, z) \Rightarrow D_\star(y, z))\big].$$

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.

○

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 (Φ, α, β) is defined as follows:

1. Φ is the forgetful functor $\mathrm{Sign}^{FOLR} \to \mathrm{Sign}^{FOL}$ 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 $\mathrm{Sen}^{FOLR}(S, S_0, F, F_0, P, P_0) \subseteq \mathrm{Sen}^{FOL}(S, F, P)$ (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 $\mathrm{Mod}^{FOL}(S, F, P)$.

This is a comorphism mapping signatures to signatures. Hence all Γ's are empty. Thus,

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

The specification of the model constraints requires that $C\big((S, S_0, F, F_0, P, P_0)\big)$ 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 σ in F_0, $(\forall x, y, Z)\sigma(x, Z) = \sigma(y, Z)$, and

3. for each π in P_0, $(\forall x, y, Z)\pi(x, Z) \Leftrightarrow \pi(y, Z)$.

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 $(\forall x, y)D_s(x, y)$ 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 C that specifies the constraints. Finally, the adequacy condition for β'^C is checked as in Example 4.2.5.

○

Example 4.2.7 In the case of the encoding of $\mathcal{H}PAR'$ (see Example 3.2.7) the base comorphism (Φ, α, β) extends canonically the first encoding comorphism $PA \to FOL^{\text{pres}}$, presented in Example 2.3.15, say $(\Phi_0, \alpha_0, \beta_0)$, to a comorphism $PAR \to FOL^{\text{pres}}$ making

- $\Phi(S, S_0, TF, TF_0, PF, PF_0) = \Phi_0(S, TF, PF)$,
- $\alpha = \alpha_0$ and
- $\beta = \beta_0$.

The encoding to FOL^{pres} obtained as an instance of the general encoding presented above yields $\Phi'(S, S_0, TF, TF_0, PF, PF_0) = \big(([S], [TF + PF] + \overline{Nom}, (D_s)_{s\in S} + [(Def_s)_{s\in S}] + \overline{\Lambda}), D_{TF+PF} \cup \overline{\Gamma_{(S,TF,PF)}}\big)$. For any $\mathcal{H}PAR$ signature $((S, S_0, TF, TF_0, PF, PF_0), Nom, \Lambda)$, the rigidification constraint on S_0 and TF_0 are axiomatized by $\{(\forall x, z)D_s(x, z) \mid s \in S_0\}$ and $\{(\forall x, y, Z)\sigma(x, Z) = \sigma(y, Z) \mid \sigma \text{ in } TF_0\}$ respectively. The constraints on the definability of PF_0 must be axiomatized, for any $\sigma \in (PF_0)_{\underline{ar}\to s}, \underline{ar} \in S^*, s \in S$ as

$$Def_s(x, (\sigma(x, Z)) \Leftrightarrow Def_s(y, (\sigma(y, Z)).$$

Hence,

$$\begin{aligned}
&C((S, S_0, TF, TF_0, PF, PF_0), Nom, \Lambda) = \\
&\{(\forall x, z)D_s(x, z) \mid s \in S_0\} \cup \\
&\{(\forall x, y, Z)\sigma(x, Z) = \sigma(y, Z) \mid \sigma \text{ in } TF_0\} \cup \\
&\{(\forall x, y, Z)Def_s(x, (\sigma(x, Z)) \Leftrightarrow \\
&Def_s(y, (\sigma(y, Z)) \mid \sigma \in (PF_0)_{\underline{ar}\to s}, \underline{ar} \in S^*, s \in S\}
\end{aligned}$$

Note that the first component in the definition of C covers also condition (11) of Theorem 4.1.1 while condition (12) of Theorem 4.1.1 is entailed by the second component of C. Finally, the adequacy condition for β'^{C} is checked 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 $(\Phi, \alpha, \beta) : \mathcal{I} \to FOL^{\text{pres}}$ to comorphism $(\Phi'^{C}, \alpha', \beta') : \mathcal{HI}^{C} \to FOL^{\text{pres}}$. We assume the conditions and notation of Theorem 4.1.1 above.

Proposition 4.3.1 *Let us assume, for each $\mathcal{I}$-signature Σ, a mapping*

$$\delta_\Sigma : |\text{Mod}^{\mathcal{I}}(\Sigma)| \to |\text{Mod}^{FOL^{\text{pres}}}(\Phi(\Sigma))|$$

such that for each Σ-model A, $\beta_\Sigma(\delta_\Sigma(A)) = A$. For each $\mathcal{HI}$ signature $\Delta = (\Sigma, \text{Nom}, \Lambda)$ and each model $(M, W) \in |\text{Mod}^{\mathcal{HI}}(\Delta)|$ 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*

$$\delta_\Sigma(M_w)_s = \delta_\Sigma(M_{w'})_s. \tag{16}$$

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

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 λ in Λ,
- 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 $\sigma \in (F_\Sigma)_{\underline{ar} \to v}$,
  $$M'_\sigma(w, m) = \begin{cases} \delta_\Sigma(M_w)_\sigma(m), & \text{when } m \in \delta_\Sigma(M_w)_{\underline{ar}}; \\ \text{any } y \in \delta_\Sigma(M_w)_v, & \text{otherwise.} \end{cases}$$
  Note that the correctness of this definition relies upon our basic hypothesis that the *FOL*-models have non-empty carriers.

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

Now we have to prove that M' satisfies the sentences of $\Phi'(\Delta)$. That $M' \models D_{F_\Sigma}$ follows immediately from the definitions of M'_{D_s} and of M'_σ. Also from the hypothesis (16) 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

$$w \in |W|,\ M'|_w = \delta_\Sigma(M_w). \tag{17}$$

Since $\delta_\Sigma(M_w) \models \Gamma_\Sigma$, from (17) and Lemma 4.1.1 it follows that $M'^w \models \{[\gamma]^x \mid \gamma \in \Gamma_\Sigma\}$, (where M'^w denotes the expansion of M' to the signature extended with the constant x such that $M'^w_x = w$). From the latter relation we deduce that $M' \models \overline{\Gamma_\Sigma}$.

That $\beta'(M') = (M, W)$ may be noted immediately with the help of the relation (17). Hence we define $\delta'_\Delta(M, W) = M'$. □

Corollary 4.3.1 *Any comorphism as in Corollary 4.1.1, such that for each constrained model* $(M, W) \in |\mathrm{Mod}^C(\Delta)|$,

1. (M, W) *satisfies condition (16) of Proposition 4.3.1, and*
2. $\delta'_\Delta(M, W) \models C(\Delta)$

is conservative.

Example 4.3.1 The encoding of S4 hybrid propositional logic of Example 4.2.1 is conservative according to Corollary 4.3.1, as follows:

– δ_Σ are identities,

– condition (16) of Proposition 4.3.1 is vacuously fulfilled (all Γ'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:*

- δ_* *is the identity;*
- *since all* Γ*'s are empty, the condition (16) of Proposition 4.3.1 is vacuously fulfilled.*

○

Example 4.3.3 ($\mathcal{H}^2PL2FOL$) Let as characterize the conservativity of $\mathcal{H}^2PL2FOL$ of Example 4.2.3. Consider therefore the map

$$\delta_{(\mathtt{Prop},\mathrm{Nom}')} : \mathrm{Mod}^{\mathcal{H}PL}(\mathtt{Prop}, \mathrm{Nom}') \to \mathrm{Mod}^{FOL}(\{\mathrm{ST}'\}, \overline{\mathrm{Nom}'}, [\mathrm{Prop}] + \lambda)$$

where $\delta_{(\mathtt{Prop},\mathrm{Nom}')}(M, W) = M'$ and

- $M'_{\mathrm{ST}} = W$,
- for any $p \in [\mathtt{Prop}]$, $w \in M'_{\mathrm{ST}}$, $M'_p(w)$ iff $(M_w)_p$,
- for any $i \in \overline{\mathrm{Nom}'}$, $M'_i = W_i$ and
- $M'_\lambda = W_\lambda$.

Hence we have $\beta_{(\mathtt{Prop},\mathrm{Nom}')}(M') = (M_1, W_1)$ (*cf.* Example 4.2.3), where

- $W_1 = M'_{\mathrm{ST}} = W$;
- for any $p \in \mathtt{Prop}$, $(M_{1w})_p$ iff $w \in M'_p$ iff $(M_w)_p$;
- for any $i \in \mathrm{Nom}$, $(W_1)_i = M_i = W_i$ and
- $(W_1)_\lambda = M'_\lambda = W_\lambda$.

Therefore $(M_1, W_1) = (M, W)$, i.e.,

$$\delta_{(\mathtt{Prop},\mathrm{Nom}')}(\beta_{(\mathtt{Prop},\mathrm{Nom}')}(M, W)) = (M, W).$$

Moreover, condition (16) of Proposition 4.3.1 is vacuously fulfilled (all Γ's are empty). Finally, we observe that there are models (M, W) failing to verify $\delta'_\Delta(M, W) \models (\forall x, y)D_{\mathrm{ST}'}(x, y)$. By this reason, the comorphism is *not* conservative.

For the case $\mathcal{H}^2PL'$, we have by the definition of δ' (cf. proof of Proposition 4.3.1) that $\delta'(M, W)_{D_{\mathrm{ST}'}} = \{(w, m) \mid m \in \delta_\Sigma(M_w)_{\mathrm{ST}'}\}$. Since we have sharing on the sub-states universe, this is equal to $|M'_{\mathrm{ST}}| \times |M'_{\mathrm{ST}'}|$. Hence $\delta'_\Delta(M, W) \models (\forall x, y)D_{\mathrm{ST}'}(x, y)$ and, therefore, $\mathcal{H}^2PL'2FOL$ is conservative. ○

Example 4.3.4 The encoding of the quantifier free hybridisation of *FOL* of Example 4.2.4 is *not* conservative. Although the condition (16) of Proposition 4.3.1 is vacuously fulfilled (the Γ'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. δ_Σ are identities,
2. condition (16) of Proposition 4.3.1 is vacuously fulfilled (all Γ'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 $\mathcal{H}REL'_0 2FOL^{pres}$ 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 δ', that for any $w \in W$, $|(M'_{D_\star}|_w)| = |M_w|$. Hence

$$|(M'_{D_\star}|_w)| = |M_w| \subseteq |M_{w'}| = |(M'_{D_\star}|_{w'})|.$$

Therefore, for any m, $M'_{D_\star}(w, m) \subseteq M'_{D_\star}(w', m)$. Hence,

$$M' \models (\forall x, y, z)[\lambda(x, y) \Rightarrow (D_\star(x, z) \Rightarrow D_\star(y, z))]$$

○

Example 4.3.6 The encoding of the hybridisation of *FOLR* of Example 4.2.6 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 4.2.7. 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 $s \in S$, $\delta_\Sigma(M)_s = M_s \cup \{\perp\}$ where $\perp$ is a new element; and $\delta_\Sigma(M)_{Def_s} = M_s$
- for any $\sigma \in (TF + PF)_{\underline{ar} \to s}$,

$$\delta_\Sigma(M)_\sigma(m) = \begin{cases} M_\sigma(m), & \text{if } m \in \delta_\Sigma(M)_{Def_{\underline{ar}}} \text{ and} \\ & M_\sigma(m) \text{ is defined} \\ \perp, & \text{otherwise.} \end{cases}$$

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

The condition (16) of Proposition 4.3.1 is satisfied as follows. For each $(M, W) \in |\mathrm{Mod}^{\mathcal{H}PAR'}(\Delta)|$, each rigid sort s (since all quantifications with first order variables are over rigid sorts) and any $w, w' \in |W|$, we have

$$\begin{aligned} \delta_\Sigma(M_w)_s &= (M_w)_s \cup \{\perp\} && \text{(definition of } \delta) \\ &= (M_{w'})_s \cup \{\perp\} && ((M_w)_s = (M_{w'})_s \text{ because } s \text{ is rigid }) \\ &= \delta_\Sigma(M_{w'})_s && \text{(definition of } \delta). \end{aligned}$$

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 $\delta'_\Delta(M, W)_s = \delta_\Sigma(M_w)_s$ 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 σ in TF_0, $\delta'_\Delta(M, W) \models (\forall x, y, Z)\sigma(x, Z) = \sigma(y, Z)$ holds because of the following facts:
   - for each rigid sort s and each $w \in |W|$, $\delta'_\Delta(M, W)_s = \delta_\Sigma(M_w)_s$;

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

3. For all $\underline{ar} \in S^*$, $s \in S$ and $\sigma \in (\mathsf{PF}_0)_{\underline{ar} \to s}$,

$$\delta'_\Delta(M, W) \models (\forall x, y, Z)\mathsf{Def}_s(x, (\sigma(x, Z)) \Leftrightarrow \mathsf{Def}_s(y, (\sigma(y, Z))$$

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

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

and $\delta'_\Delta(M, W)_\sigma(w', m) = \delta_\Sigma(M_{w'})_\sigma(m)$. 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 σ, i.e. because $(M_w)_\sigma$ and $(M_{w'})_\sigma$ have the same domain.

$\circ$

5

BISIMULATION AND REFINEMENT NOTIONS FOR HYBRIDISED LOGICS

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 purposed and characterized in universal coalgebra [AM89, Rut00]. But the concept also arose independently in 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 focussing 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 an hybridised institution $\mathcal{HI}$, 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* $M, M' \in \mathrm{Mod}^{\mathcal{I}}(\Sigma)$ *and* Sen' *be a subfunctor of* $\mathrm{Sen}^{\mathcal{I}}$. *Models* M *and* M' *are elementarily equivalent with respect to sentences in* $\mathrm{Sen}'(\Sigma)$, *in symbols* $M \equiv^{\mathrm{Sen}'} M'$, *if for any* $\rho \in \mathrm{Sen}'(\Sigma)$

$$M \models^{\mathcal{I}} \rho \Leftrightarrow M' \models^{\mathcal{I}} \rho. \tag{18}$$

If only the implication $\Rightarrow$ *of (18) holds in the right hand side of the above equivalence we write* $M \gg_{\varphi}^{\mathrm{Sen}'} M'$.

Note the role of φ 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 $M \equiv_{\varphi}^{\mathrm{Sen}'} M'$ whenever $M \equiv^{\mathrm{Sen}'} \mathrm{Mod}^{\mathcal{I}}(\varphi)(M')$ for a given $\varphi \in \mathrm{Sign}^{\mathcal{I}}(\Sigma, \Sigma')$, $M \in \mathrm{Mod}^{\mathcal{I}}(\Sigma)$ and $M' \in \mathrm{Mod}^{\mathcal{I}}(\Sigma')$. Models M and M' are said to be (φ, Sen')-elementarily equivalent.

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

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

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 $\mathcal{HI}$ be the quantifier-free hybridisation of the institution $\mathcal{I}$ and $\varphi \in \mathrm{Sign}^{\mathcal{HI}}(\Delta, \Delta')$ a signature morphism. Let Sen' be a subfunctor of $\mathrm{Sen}^{\mathcal{I}}$. A (φ, Sen')-*bisimulation between models $(M, W) \in \mathrm{Mod}^{\mathcal{HI}}(\Delta)$ and $(M', W') \in \mathrm{Mod}^{\mathcal{HI}}(\Delta')$ *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 $wB_\varphi w'$, then for each $k \in \{1, \ldots, n\}$ there is a $w'_k \in |W'|$ such that $w_k B_\varphi w'_k$ and $(w', w'_1, \ldots, w'_n) \in W'_{\varphi_{\mathrm{MS}}(\lambda)}$.

(ii) For any $\lambda \in \Lambda_n$ if $(w', w'_1, \ldots, w'_n) \in W'_{\varphi_{\mathrm{MS}}(\lambda)}$ and $wB_\varphi w'$, 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 $wB_\varphi w'$, and for any $i \in \mathrm{Nom}$,

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

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

(v) for any $wB_\varphi w'$, $M_w \equiv^{\mathrm{Sen}'}_{\varphi_{\mathrm{Sign}}} M'_{w'}$.

*We say that (M, W) and (M', W') are (φ, Sen')-*bisimilar*, and write $(M, W) \rightleftharpoons_\varphi (M', W')$, if there is a (φ, Sen')-bisimulation B_φ between them. Whenever φ is the identity we simply talk of a bisimulation B and bisimilarity $\rightleftharpoons$.*

Next theorem establishes that, for quantifier-free hybridisations, the (local)-hybrid satisfaction $\models^{\mathcal{HI}}$ is invariant under (φ, Sen)-bisimulations:

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

$$(M, W) \models^w \rho \text{ iff } (M', W') \models^{w'} \mathrm{Sen}^{\mathcal{HI}}(\varphi)(\rho). \tag{19}$$

Proof. The proof is by induction on the structure of the sentences.

1. $\rho = i$ for some $i \in \mathrm{Nom}$:

$$
\begin{aligned}
& (M, W) \models^{w} i \\
\Leftrightarrow \quad & \{ \text{ defn. of } \models^{w} \} \\
& W_i = w \\
\Leftrightarrow \quad & \{ \text{ (iii) of Definition 5.1.2 } \} \\
& W'_{\varphi(i)} = w' \\
\Leftrightarrow \quad & \{ \text{ defn. of } \models^{w'} \} \\
& (M', W') \models^{w'} \varphi_{\mathrm{Nom}}(i) \\
\Leftrightarrow \quad & \{ \text{ defn of } \mathrm{Sen}^{\mathcal{HJ}}(\varphi) \} \\
& (M', W') \models^{w'} \mathrm{Sen}^{\mathcal{HJ}}(\varphi)(i)
\end{aligned}
$$

2. $\rho \in \mathrm{Sen}^{\mathcal{J}}(\Sigma)$:

$$
\begin{aligned}
& (M, W) \models^{w} \rho \\
\Leftrightarrow \quad & \{ \text{ defn. of } \models^{w} \} \\
& M_w \models^{\mathcal{J}} \rho \\
\Leftrightarrow \quad & \{ \text{ by hypothesis } M_w \equiv_{\varphi_{\mathrm{Sign}}} M'_{w'} + \text{Corollary 5.1.1} \} \\
& M'_{w'} \models \mathrm{Sen}^{\mathcal{J}}(\varphi_{\mathrm{Sign}})(\rho) \\
\Leftrightarrow \quad & \{ \text{ defn. of } \models^{w'} \} \\
& (M', W') \models^{w'} \mathrm{Sen}^{\mathcal{J}}(\varphi_{\mathrm{Sign}})(\rho) \\
\Leftrightarrow \quad & \{ \text{ defn of } \mathrm{Sen}^{\mathcal{HJ}}(\varphi) \} \\
& (M', W') \models^{w'} \mathrm{Sen}^{\mathcal{HJ}}(\varphi)(\rho)
\end{aligned}
$$

3. $\rho = \xi \vee \xi'$ for some $\xi, \xi' \in \mathrm{Sen}^{\mathcal{HJ}}(\Delta)$:

$$
\begin{aligned}
& (M, W) \models^{w} \xi \vee \xi' \\
\Leftrightarrow \quad & \{ \text{ defn. of } \models^{w} \} \\
& (M, W) \models^{w} \xi \text{ or } (M, W) \models^{w} \xi' \\
\Leftrightarrow \quad & \{ \text{ I.H.} \} \\
& (M', W') \models^{w'} \mathrm{Sen}^{\mathcal{HJ}}(\varphi)(\xi) \text{ or} \\
& (M', W') \models^{w'} \mathrm{Sen}^{\mathcal{HJ}}(\varphi)(\xi')
\end{aligned}
$$

$$\Leftrightarrow \quad \{\text{ defn. of } \models^{w}\}$$
$$(M', W') \models^{w'} \mathrm{Sen}^{\mathcal{HI}}(\varphi)(\xi \vee \xi')$$

The proofs for cases $\rho = \xi \wedge \xi'$, $\rho = \xi \Rightarrow \xi'$, $\rho = \neg\xi$, etc. are analogous.

4. $\rho = [\lambda](\xi_1, \ldots, \xi_n)$ for some $\xi_1, \ldots, \xi_n \in \mathrm{Sen}^{\mathcal{HI}}(\Delta)$, $\lambda \in \Lambda_{n+1}$:

$$(M, W) \models^{w} [\lambda](\xi_1, \ldots, \xi_n)$$
$$\Leftrightarrow \quad \{\text{ defn. of } \models^{w}\}$$
for any $(w, w_1, \ldots, w_n) \in W_\lambda$ there is some $k \in \{1, \ldots, n\}$ such that $(M, W) \models^{w_k} \xi_k$
$$\Leftrightarrow \quad \{*\}$$
for any $(w', w'_1, \ldots, w'_n) \in W'_{\varphi_{\mathrm{MS}}(\lambda)}$ there is some $p \in \{1, \ldots, n\}$ such that $(M', W') \models^{w'_p} \mathrm{Sen}^{\mathcal{HI}}(\varphi)(\xi_p)$
$$\Leftrightarrow \quad \{\text{ defn. of } \models^{w'}\}$$
$$(M', W') \models^{w'} [\varphi_{\mathrm{MS}}(\lambda)](\mathrm{Sen}^{\mathcal{HI}}(\varphi)(\xi_1), \ldots, \mathrm{Sen}^{\mathcal{HI}}(\varphi)(\xi_n))$$
$$\Leftrightarrow \quad \{\text{ defn. of } \mathrm{Sen}^{\mathcal{HI}}(\varphi)\}$$
$$(M', W') \models^{w'} \mathrm{Sen}^{\mathcal{HI}}(\varphi)([\lambda](\xi_1, \ldots, \xi_n))$$

For the step marked with * we proceed as follows. Supposing $(w', w'_1, \ldots, w'_n) \in W'_{\varphi_{\mathrm{MS}}(\lambda)}$ with $w B_\varphi w'$, 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. $(M', W') \models^{w'_p} \mathrm{Sen}^{\mathcal{HI}}(\varphi)(\xi_p)$. 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 $\xi \in \mathrm{Sen}^{\mathcal{HI}}(\Delta)$ and $i \in \mathrm{Nom}$:

$$
\begin{aligned}
& (M, W) \models^{w} @_i \xi \\
\Leftrightarrow \quad & \{ \text{ defn. of } \models^{w} \} \\
& (M, W) \models^{W_i} \xi \\
\Leftrightarrow \quad & \{ \text{ I.H. and clause (iv) of Definition 5.1.2} \} \\
& (M', W') \models^{W'_{\varphi_{\mathrm{Nom}}(i)}} \mathrm{Sen}^{\mathcal{HI}}(\varphi)(\xi) \\
\Leftrightarrow \quad & \{ \text{ defn. of } \models^{w} \} \\
& (M', W') \models^{w} @_{\varphi_{\mathrm{Nom}}(i)} \mathrm{Sen}^{\mathcal{HI}}(\varphi)(\xi) \\
\Leftrightarrow \quad & \{ \text{ defn. of } \mathrm{Sen}^{\mathcal{HI}}(\varphi) \} \\
& (M', W') \models^{w} \mathrm{Sen}^{\mathcal{HI}}(\varphi)(@_i \xi)
\end{aligned}
$$

□

As a direct consequence of the previous theorem we get the following characterisation of the preservation of (global) satisfaction, $\models^{\mathcal{HI}}$, under φ-bisimilarity:

Corollary 5.1.2 *On the conditions of Theorem 5.1.1, let* $(M, W) \rightleftharpoons_{\varphi} (M', W')$ *witnessed by a total and surjective bisimulation. Then,*

$$(M, W) \models^{\mathcal{HI}} \rho \textit{ iff } (M', W') \models^{\mathcal{HI}} \mathrm{Sen}^{\mathcal{HI}}(\varphi)(\rho). \tag{20}$$

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 $\varphi = \mathrm{id}$ and $\mathrm{Sen}' = \mathrm{Sen}^{\mathcal{I}}$. Let consider the models

$$(M, W), (M', W') \in |\mathrm{Mod}^{\mathcal{H}PL}(P, \mathrm{Nom}, \{\lambda\})|.$$

B is a bisimulation between (M, W) and (M', W') if

(i) for any $(w, w_1) \in W_\lambda$ with $w B w'$, there is a $w'_1 \in |W'|$ such that $w_1 B w'_1$ and $(w_1, w'_1) \in W'_\lambda$;

(ii) for any $(w', w'_1) \in W'_\lambda$ with $w B w'$, there is a $w_1 \in |W|$ such that $w_1 B w'_1$ and $(w_1, w'_1) \in W'_\lambda$;

(iii) for any $i \in \mathrm{Nom}$, $w B w'$, $w = W_i$ iff $w' = W'_i$;

(iv) for any $i \in \mathrm{Nom}$, $W_i B W'_i$;

(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 $\mathcal{H}EQ$) Consider now the instantiation of 5.1.2 for $\mathcal{H}EQ$. All one has to do is to replace condition (iv) in Definition 5.1.2 by its instantiation for algebras: two algebras are equationaly 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 $\mathcal{HI}$ be the quantified-free hybridisation of an institution $\mathcal{I}$, $\varphi \in \mathrm{Sign}^{\mathcal{HI}}(\Delta, \Delta')$ a signature morphism and Sen' a subfunctor of $\mathrm{Sen}^{\mathcal{I}}$. A (φ, Sen')-refinement of $(M, W) \in \mathrm{Mod}^{\mathcal{HI}}(\Delta)$ by $(M', W') \in \mathrm{Mod}^{\mathcal{HI}}(\Delta')$ consists of a non-empty relation $R_\varphi^{\mathrm{Sen}'} \subseteq |W| \times |W'|$ such that, for any $w R_\varphi^{\mathrm{Sen}'} w'$,*

(f.i) for any $i \in \mathrm{Nom}$, if $W_i = w$ then $W'_{\varphi_{\mathrm{Nom}}(i)} = w'$.

(f.ii) $M_w \gg_{\varphi}^{\mathrm{Sen}'} M'_{w'}$.

(f.iii) *for any* $i \in \mathrm{Nom}$, $W_i \, R_{\varphi}^{\mathrm{Sen}'} \, W'_{\varphi_{\mathrm{Nom}}(i)}$.

(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 a* $w'_k \in |W'|$ *such that* $w_k R_\varphi w'_k$ *and* $(w', w'_1, \ldots, w'_n) \in W'_{\varphi_{\mathrm{MS}}(\lambda)}$.

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 R_φ-refinement from a Δ-hybrid model (M, W) with $|W| = \{w\}$ and $W_\lambda = \emptyset$ for $\lambda \in \Lambda_n$ to any other Δ'-hybrid model (M', W') such that $\mathrm{Mod}^{\mathcal{HJ}}(\varphi_{\mathrm{Sign}})(M'_{w'}) = M_w$ 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 R_φ-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 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 $M_w = \mathrm{Mod}(\varphi)(M'_{w'})$. In particular, an id-refinement implies the equality of realisations of related states (since, the implication "$M_w \models^{PL} \neg p$ then $M'_{w'} \models^{PL} \neg p$" is equivalent to the implication "$M'_{w'} \models^{PL} p$ then $M_w \models^{PL} p$". 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 $\mathsf{Sen}'(\mathtt{Prop})$ the set of propositional sentences without negations.

Given an institution $\mathcal{I} = (\mathsf{Sign}^{\mathcal{I}}, \mathsf{Sen}^{\mathcal{I}}, \mathsf{Mod}^{\mathcal{I}}, (\models_\Sigma)_{\Sigma \in |\mathsf{Sign}^{\mathcal{I}}|})$ and a sentences subfunctor $\mathsf{Sen}' \subseteq \mathsf{Sen}^{\mathcal{I}}$, we denote by $\mathcal{HI}'$ the hybridisation of the institution $\mathcal{I}' = (\mathsf{Sign}^{\mathcal{I}}, \mathsf{Sen}', \mathsf{Mod}^{\mathcal{I}}, (\models_\Sigma)_{\Sigma \in |\mathsf{Sign}^{\mathcal{I}}|})$.

Definition 5.2.2 (Sen'**-Positive Existencial sentences)** *The* Sen'*-positive existencial sentences of a signature* $\Delta \in |\mathsf{Sign}^{\mathcal{HI}}|$ *are given by a subfunctor* $\mathsf{Sen}^{\mathcal{HI}}_{+} \subseteq \mathsf{Sen}^{\mathcal{HI}'}$ *defined recursivly for each signature* Δ *as* $\mathsf{Sen}^{\mathcal{HI}'}(\Delta)$ *but excluding both negations and box modalities. For each signature morphism* $\varphi : \Delta \to \Delta'$, $\mathsf{Sen}^{\mathcal{HI}}_{+}(\varphi)$ *is the restriction of* $\mathsf{Sen}^{\mathcal{HI}'}(\varphi)$ *to* $\mathsf{Sen}^{\mathcal{HI}}_{+}(\Delta)$.

Theorem 5.2.1 *Let* $\mathcal{HI}$ *be the quantifier free hybridisation of an institution* $\mathcal{I}$, Sen' *a subfunctor of* $\mathsf{Sen}^{\mathcal{I}}$, $\varphi \in \mathsf{Sign}^{\mathcal{HI}}(\Delta, \Delta')$ *a signature morphism,* $(M, W) \in \mathsf{Mod}^{\mathcal{HI}}(\Delta)$ *and* $(M', W') \in \mathsf{Mod}^{\mathcal{HI}}(\Delta')$. *Suppose that* (M', W') *is a refinement of* (M, W) *witnessed by the relation* $R^{\mathsf{Sen}'}_{\varphi}$. *Then, for any* $w R^{\mathsf{Sen}'}_{\varphi} w'$ *and* $\rho \in \mathsf{Sen}^{\mathcal{HI}}_{+}(\Delta)$,

$$(M, W) \models^{w} \rho \textit{ implies that } (M', W') \models^{w'} \mathsf{Sen}^{\mathcal{HI}}(\varphi)(\rho).$$

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,

$$\begin{array}{ll}
 & (M, W) \models^{w} \langle\lambda\rangle(\xi_1, \ldots, \xi_n) \\
\Leftrightarrow & \quad \{ \text{ defn. of } \models^{w} \} \\
 & \text{there exists } (w, w_1, \ldots, w_n) \in W_\lambda \\
 & \text{such that } (M, W) \models^{w_k} \xi_k \text{ for any } k \in \{1, \ldots, n\} \\
\Rightarrow & \quad \{ \text{ By (f.iii), we have } w_k R_\varphi w'_k \text{ for any } k \in \{1, \ldots, n\} + \text{I.H. } \} \\
 & \text{there exists } (w', w'_1, \ldots, w'_n) \in W'_{\varphi_{MS}(\lambda)} \\
 & \text{such that } (M', W') \models^{w'_k} \xi_k \text{ for any } k \in \{1, \ldots, n\} \\
\Leftrightarrow & \quad \{ \text{ defn. of } \models^{w'} \} \\
 & (M', W') \models^{w'} \langle\varphi_{MS}(\lambda)\rangle(\mathsf{Sen}^{\mathcal{HI}}(\varphi)(\xi_1), \ldots, \mathsf{Sen}^{\mathcal{HI}}(\varphi)(\xi_n))
\end{array}$$

$$\Leftrightarrow \qquad \{\text{ defn. of } \mathrm{Sen}^{\mathcal{HJ}}(\varphi)\}$$
$$(M', W') \models^{w'} \mathrm{Sen}^{\mathcal{HJ}}(\varphi)(\langle\lambda\rangle(\xi_1, \ldots, \xi_n))$$

□

Corollary 5.2.1 *In the conditions of Theorem 5.2.1, for any* $\rho \in \mathrm{Sen}^{\mathcal{HJ}}_{+}(\Delta)$, *if* R_φ *is surjective, then*

$$(M, W) \models \rho \textit{ implies that } (M', W') \models \mathrm{Sen}^{\mathcal{HJ}}(\varphi)(\rho).$$

The following examples illustrate refinement situations in this setting.

Example 5.2.1 (Refinement in $\mathcal{H}MVL_L$) Figure 2 illustrates an example of a Sen'-refinement in $\mathcal{H}MVL_{L_4}$, for L_4 represented in Figure 2. Consider $\mathrm{Sen}' \subseteq \mathrm{Sen}^{\mathcal{J}}$ restricting the base sentences to propositions, i.e., $\mathrm{Sen}'(\mathrm{LProp}) = \{(p, l) | p \in \mathrm{LProp} \text{ and } l \in L_4\}$. Con-

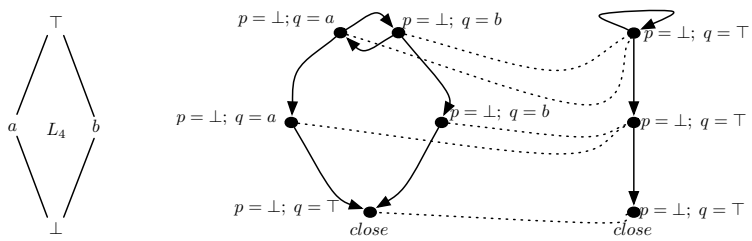

Figure 2: Refinement in $\mathcal{H}MVL_L$.

ditions *(f.i)* and *(f.iii)* of Definition 5.2.1 are obviously satisfied. In what concerns the verification of condition *(f.ii)* for which $(p, l) \in \mathrm{Sen}'(\mathrm{LProp})$, $M_w \models^{MVL_{L_4}}_{\mathrm{LProp}} (p, l) \Rightarrow M'_{w'} \models^{MVL_{L_4}}_{\mathrm{LProp}} (p, l)$, it is sufficient to see that, $(M_w \models p) \leqslant (M'_{w'} \models p)$, $p \in \mathrm{LProp}$. ∘

Example 5.2.2 (Refinement in $\mathcal{H}EQ$) Consider a store system abstractly modelled as the initial algebra A of the $((S, F), \Gamma)$ where
$S = \{mem, elem\}$, $F_{mem\,elem \to mem} = \{write\}$, $F_{mem \to mem} = \{del\}$ and $F_{\underline{ar} \to s} = \emptyset$ otherwise and $\Gamma = \{del(write(m, e)) = m\}$. 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 $\mathtt{shift}$. Note that the abstract model can be seen as the $\big((S,F),\emptyset,\{\mathtt{shift}\}\big)$-hybrid model $\mathcal{M} = (M, W)$, taking $|W| = \{\star\}$, $W_{\mathtt{shift}} = \emptyset$ and $M_\star = A$. Then, we take the inclusion morphism $\varphi_{\mathsf{Sign}} : (S,F) \hookrightarrow (S,F')$ where F' extends F with $F_{\mathtt{mem}\to\mathtt{elem}} = \mathtt{read}$ and $F_{\mathtt{mem}} = \{\mathtt{empty}\}$. For the envisaged refinement let us consider the model $\mathcal{M}' = (M', W')$ where $W' = \{s_1, s_2\}$ and $W'_{\mathtt{shift}} = \{(s_1, s_2), (s_2, s_1)\}$ and where M_{s_1} and M_{s_2} are the initial algebras of the equations presented in Figure 3. It

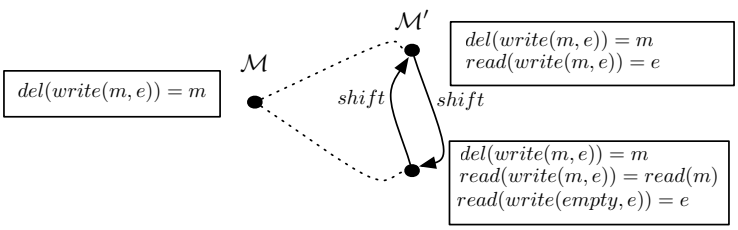

Figure 3: Refinement in $\mathcal{H}EQ$.

is not difficult to see that $R = \{(\star, s_1), (\star, s_2)\}$ is a φ-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

HYBRIDISATION FOR THE WORKING SOFTWARE ENGINEER

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 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 $\mathcal{I}$ stands for the logic used locally to specify individual configurations and $\mathcal{HI}$ 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*:

reconfigurations as transitions, configurations as local models

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*.

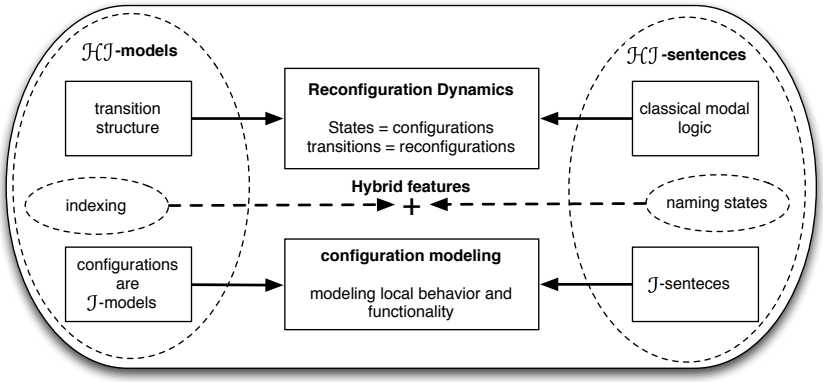

Figure 4: Specification of reconfigurable systems: The approach.

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) $\mathcal{I}$.

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**.

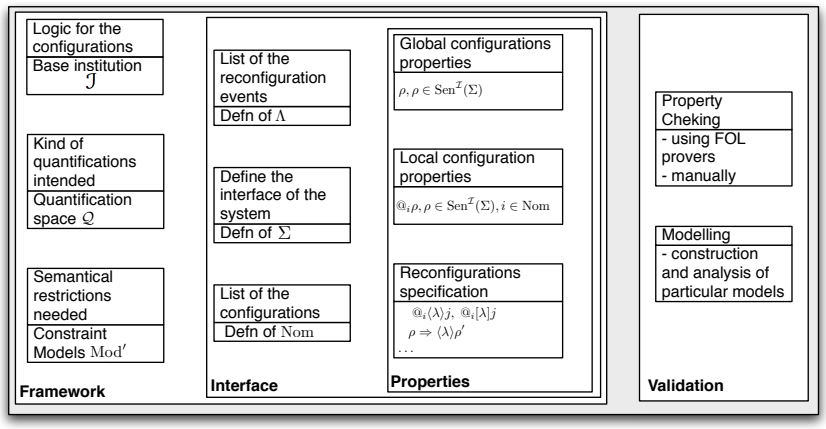

Figure 5: Specification of reconfigurable systems: The method.

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,

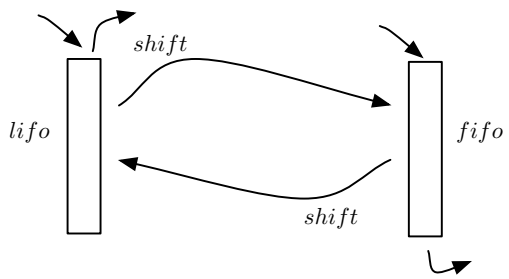

Figure 6: A plastic buffer.

- 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 $\mathcal{I}$ 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 Λ). Moreover, a (local) $\mathcal{I}$-signature for the specification of individual configurations (i.e., the component Σ) 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:

spec SPECNAME **in H**BASEINST =
  Nom
    list of nominal symbols
  Modal
    list of modality symbols
  BaseSig
    signature of the base institution
  Axioms
    set of axioms

SPECNAME is the specification identifier, and BASEINST stands for the base institution $\mathcal{I}$ adopted when fixing the specification framework. The fields **Nom** and **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 **BaseSig** is the declaration of the $\mathcal{I}$-signature Σ. 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 **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 *fifo* (for the queue mode) and *lifo* (for the stack mode), respectively. The event responsible for triggering a reconfiguration is denoted by the modality

symbol *shift*. This makes up the global, *REL*-based component of the hybrid signature. Basically, its role is to express the system's reconfiguration semantics.

The local interface is a *PAR*-signature with the following components:

- a sort *mem* of stacks/queues;
- a sort *elem* of elements;
- a total operation *write* to denote the 'push/enqueue' operation;
- a constant *new* to denote the empty buffer;
- a partial operation *read* to denote the 'top/front' operation;
- a partial operation 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, *elem* and *mem* are declared to be rigid;
- operation *write* is also rigid and, since it is total, it has the same effect in both configurations;
- operations *read* and *del* play different roles in each individual configuration. However, both are declared as rigid because they are partial and in $\mathcal{H}PAR$ 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 *read* and *del* are defined on non-empty stacks/queues.

The local, *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 **R** is used to mark the rigid components:

Nom

fifo
lifo

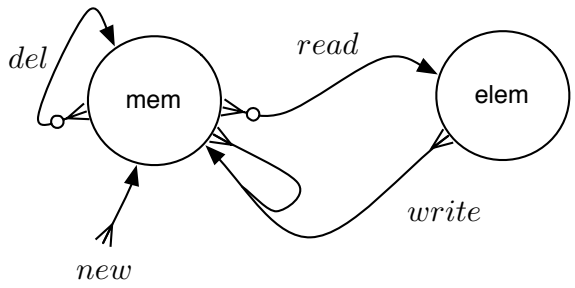

Figure 7: Plastic Buffer *PAR* signature.

Modal
  shift : 1
BaseSig
  sorts *mem* **R**
    elem **R**
  ops *new* : $\rightarrow$ *mem* **R**
    write : *mem* $\times$ *elem* $\rightarrow$ *mem* **R**
      del : *mem* $\rightarrow$? *mem* **R**
      read : *mem* $\rightarrow$? *elem* **R**

$\circ$

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) $\mathcal{J}$-sentences;

2. the local properties, i.e. relative to specific configurations, which are expressed by tagging $\mathcal{J}$-sentences with the satisfaction operator @ ($@_{\mathtt{i}}\rho$ is used to express that property ρ holds in the configuration named by $\mathtt{i}$);

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

$$\textcircled{\forall}\, e : elem;\ \textcircled{\forall}\, m : mem;$$
$$\bullet\ @_{lifo} del(write(m, e)) \stackrel{e}{=} m$$
$$\bullet\ @_{lifo} read(write(m, e)) \stackrel{e}{=} e$$

Similarly, for the local specification of configuration *fifo*,

$$\textcircled{\forall}\, e : elem;\ \textcircled{\forall}\, m : mem;$$
$$\bullet\ @_{fifo} read(write(new, e)) \stackrel{e}{=} e$$
$$\bullet\ @_{fifo} del(write(new, e)) \stackrel{e}{=} new$$
$$\bullet\ @_{fifo}\ read(m) \stackrel{e}{=} read(m) \Rightarrow read(write(m, e)) \stackrel{e}{=} read(m)$$
$$\bullet\ @_{fifo}\ read(m) \stackrel{e}{=} read(m) \Rightarrow del(write(m, e)) \stackrel{e}{=} write(del(m), e)$$

Finally, the reconfiguration dynamics is given by

$$\bullet\ @_{fifo}\langle shift\rangle lifo \wedge @_{lifo}\langle shift\rangle fifo$$

expressing configuration alternation triggered by event *shift*. The complete specification is depicted in Figure 8.

Note that the quantification is *local* which means it is placed at the level of the base institution (hence the use of notation $\textcircled{\forall}$; see Fact 3.1.3). For example the first sentence in the specification reads as $@_{lifo}(\textcircled{\forall}\, e)(\textcircled{\forall}\, m) del(write(m, e)) \stackrel{e}{=} m$.

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

$$lifo \vee fifo$$

spec PLASTICBUFFER **in** **H**PAR =
  nom
    fifo
    lifo
  modal
    shift : 1
  BaseSig
    sorts *mem* **R**
      elem **R**
    ops *new* : $\rightarrow$ *mem* **R**
      write : *mem* $\times$ *elem* $\rightarrow$ *mem* **R**
        del : *mem* $\rightarrow$? *mem* **R**
        read : *mem* $\rightarrow$? *elem* **R**

  $\textcircled{\forall}$ *e* : *elem*; $\textcircled{\forall}$ *m* : *mem*;

  %(stack properties)**%**

  • $@_{lifo} del(write(m, e)) \stackrel{e}{=} m$
  • $@_{lifo} read(write(m, e)) \stackrel{e}{=} e$

  %(queue properties)**%**

  • $@_{fifo} read(write(new, e)) \stackrel{e}{=} e$
  • $@_{fifo} del(write(new, e)) \stackrel{e}{=} new$
  • $@_{fifo}\ read(m) \stackrel{e}{=} read(m) \Rightarrow read(write(m, e)) \stackrel{e}{=} read(m)$
  • $@_{fifo}\ read(m) \stackrel{e}{=} read(m) \Rightarrow del(write(m, e)) \stackrel{e}{=} write(del(m), e)$

  %(reconfigurations spec)**%**

  • $@_{fifo} \langle shift \rangle lifo \wedge @_{lifo} \langle shift \rangle fifo$

end

Figure 8: Plastic Buffer $\mathcal{H}PAR$ Specification.

as an axiom. Similarly

$$\neg @_{lifo}\, fifo$$

or, equivalently, $\neg @_{fifo}\, lifo$, avoids confusion.

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 $\mathcal{H}PL$ 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+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 (M, W) for the specification of a 'plastic buffer' discussed above. The global, *REL* component of the signature is interpreted as follows:

- $|W| = \{\text{sfifo}, \text{slifo}\}$;
- $W_{lifo} = \text{slifo}$ and $W_{fifo} = \text{sfifo}$;
- $W_{shift} = \{(\text{sfifo}, \text{slifo}), (\text{slifo}, \text{sfifo})\}$.

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 ϵ stands for the empty sequence.

- $(M_{\text{sfifo}})_{elem} = (M_{\text{slifo}})_{elem} = A$;
- $(M_{\text{sfifo}})_{mem} = (M_{\text{slifo}})_{mem} = A^*$;
- $(M_{\text{slifo}})_{new} = (M_{\text{slifo}})_{new} = \epsilon$;
- $(M_{\text{sfifo}})_{write}(L, a) = (M_{\text{slifo}})_{write}(L, a) = L.a$;
- $(M_{\text{slifo}})_{del}(L) = J$ if $L = J.a$ for $J \in A^*$, $a \in A$ and is undefined otherwise;
- $(M_{\text{sfifo}})_{del}(L) = J$ if $L = a.J$ for $J \in A^*$, $a \in A$ and is undefined otherwise;
- $(M_{\text{slifo}})_{read}(L) = a$ if $L = J.a$, for $J \in A^*$, $a \in A$ and is undefined otherwise;
- $(M_{\text{sfifo}})_{read}(L) = a$ if $L = a.J$, for $J \in A^*$, $a \in A$ and is undefined otherwise;

One may now conjecture whether the following properties are valid.

$$(\textcircled{\forall}\, e, e', m, m')$$
$$[\mathit{shift}](m' = \mathit{write}(e, m)) \Leftrightarrow (m' = \mathit{write}(e, m)) \tag{21}$$

$$(\textcircled{\forall}\, e, e', m, m')$$
$$\langle \mathit{shift} \rangle (m' = \mathit{write}(e, m)) \Leftrightarrow (m' = \mathit{write}(e, m)) \tag{22}$$

$$(\textcircled{\forall}\, m, m', m_1, m_2)(m' = \mathit{del}(m_1)) \wedge [\mathit{shift}](m_1 = \mathit{del}(m))$$
$$\Leftrightarrow ([\mathit{shift}](m' = \mathit{del}(m_2)) \wedge (m_2 = \mathit{del}(m)) \tag{23}$$

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' $\mathcal{H}PAR'$ specification into *FOL*, following closely the procedure discussed in Example 4.2.7.

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* → *mem*;
    write : *ST* × *mem* × *elem* → *mem*;
    read : *ST* × *mem* → *elem*;
    del : *ST* × *mem* → *mem*
  preds *shift* : *ST* × *ST*;
    Def_mem : *ST* × *mem*;
    Def_elem : *ST* × *elem*;
    D_mem : *ST* × *mem*;
    D_elem : *ST* × *elem*

The $\overline{\Gamma_{(\mathsf{S},\mathsf{TF},\mathsf{PF})}}$:

$\forall\ e : elem;\ w : ST;\ m : mem$
$\bullet\ Def\_mem(w, new(w))$
$\bullet\ Def\_mem(w, m) \wedge Def\_elem(w, e) \Leftrightarrow Def\_mem(w, write(w, m, e))$
$\bullet\ Def\_mem(w, del(w, m)) \Rightarrow Def\_mem(w, m)$
$\bullet\ Def\_elem(w, read(w, m)) \Rightarrow Def\_mem(w, m)$
$\bullet\ D\_mem(w, m)$
$\bullet\ D\_elem(w, e)$

The specification of the $\mathsf{D}_{\mathsf{TF}+\mathsf{PF}}$-sentences is redundant as they are all consequences of $\mathsf{V}(\Gamma)$ (i.e. the previous two sentences). Thus, they can be safely skipped. The same happens for the first two sentences determined by the constraint functor C. The other sentences are as follows:

$\forall\ e : elem;\ w, v : ST;\ m : mem$
$\bullet\ new(w) = new(v)$
$\bullet\ write(w, m, e) = write(v, m, e)$
$\bullet\ Def\_mem(w, del(w, m)) \Leftrightarrow Def\_mem(v, del(v, m))$
$\bullet\ Def\_elem(w, read(w, m)) \Leftrightarrow Def\_elem(v, read(v, m))$

Finally, Fig. 9 depicts the complete the translation.

Using Theorem 2.3.2 combined with the conclusion of Example 4.3.7 property (22) is shown to be a consequence of the 'plastic buffer' specification. We performed this verification in the SPASS [WDF+09] automatic prover for first order logic. On the other hand, (21) and (23) do *not* hold. Both were disproved by Darwin [BFT06], another first order prover with (limited) disproving capabilities. However, the right-left implication of (21) was proved by SPASS.

6.3 DEALING WITH HETEROGENOUS 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, $\mathcal{H}EQ$.

$\forall\, e : elem;\ m : mem;\ w : ST;$

- $(Def\_mem\ (lifo,m) \wedge Def\_elem\ (lifo,e)) \Rightarrow$
  $(Def\_mem\ (lifo,m) \wedge del(lifo,write(lifo,m,e)) = m))$
- $(Def\_mem\ (lifo,m) \wedge Def\_elem\ (lifo,e)) \Rightarrow$
  $(Def\_elem\ (lifo,e) \wedge read(lifo,write(lifo,m,e)) = e));$
- $Def\_elem\ (fifo,e) \Rightarrow$
  $(Def\_elem\ (fifo,e) \wedge read(fifo,write(fifo,new(fifo),e)) = e));$
- $Def\_elem\ (fifo,e) \Rightarrow (Def\_mem\ (fifo,new(fifo)) \wedge$
  $del(fifo,write(fifo,new(fifo),e)) = new(fifo)));$
- $(Def\_mem\ (fifo,m) \wedge Def\_elem\ (fifo,e)) \Rightarrow$
  $(Def\_elem\ (fifo,read(fifo,m)) \wedge read(fifo,m) = read(fifo,m) \Rightarrow$
  $Def\_elem\ (fifo,read(fifo,m)) \wedge read(fifo,write(fifo,m,e)) =$
  $read(fifo,m));$
- $(Def\_mem\ (fifo,m) \wedge Def\_elem\ (fifo,e)) \Rightarrow$
  $(Def\_elem\ (read(fifo,m)) \wedge read(fifo,m) = read(fifo,m) \Rightarrow$
  $Def\_mem\ (fifo,write(fifo,del(fifo,m),e)) \wedge$
  $del(fifo,write(fifo,m,e)) = write(fifo,del(fifo,m),e));$

$\exists\, y, v : ST;$

- $(shift(fifo,y) \wedge (y = lifo)) \wedge (shift(lifo,v) \wedge (v = fifo)).$

end

Figure 9: The ‘plastic buffer’ specification translated into *FOL*.

spec SWITCH **in** $\mathbf{H}TRM =$
  Nom
    On; Off
  Modal
    $TurnOff$:**1**; $TurnOn$:**1**
  BaseSig
    $*$
  Axioms
  $On \vee Off$
  $@_{On}\langle TurnOff\rangle Off \wedge @_{On}[TurnOff]Off$
  $@_{Off}\langle TurnOn\rangle On \wedge @_{Off}[TurnOn]On$
end

Figure 10: A $\mathcal{H}TRM$ specification.

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 bomb. The most trivial one is the specification of the 'pump switch':

(R1) 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) 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

$$Off \wedge ready \Leftrightarrow \langle TurnOn \rangle On.$$

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, $\mathcal{H}^2PL$ is the suitable specification framework for the double scenario. To illustrate this consider the following additional requirements for the boot operation of the insulin pump:

(R3) When on the pump can be suspended or run normally

(R4) The pump can be turned off when running normally or suspended

It is clear from requirement (R3) that configuration *On* has two possible (sub)-states, identified by *Nor* and *Sus*, respectively. Moreover, we may now embody the check on sensors mentioned in (R2) as a (sub)-state of *Off*. We denote it by the nominal *Test* and call *Dead* the state where the system is inactive. A local event *?ready* will trigger the execution of the sensors' diagnostic. For technical reasons, we collect all of these components in a (unique) $\mathcal{H}PL$ base signature. However we may declare which of these states are reachable in which configuration (e.g. $@_{On}(\underline{Sus} \wedge \underline{Nor})$). Note that this sort of hybridisation allows the specification of transitions from sub-states of a configuration into sub-states of another. For instance, requirement (R2) is captured by

$$\langle TurnOn \rangle true \Rightarrow Off \wedge \underline{Test},$$

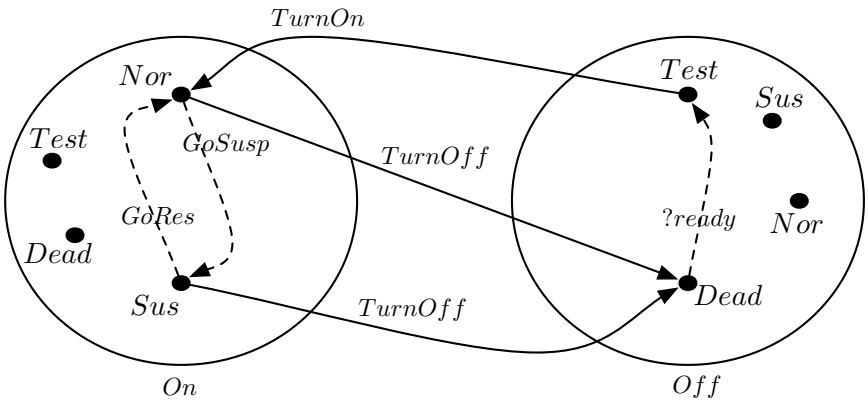

Figure 11: Hierarchical representation of the pump boot.

which can be read as follows: *"if a transition through event TurnOn is possible, the device is in state Test of configuration Off"*. Here the symbol $\mathtt{true}$ stands for a tautology of the $\mathcal{H}^2PL$. For instance, it can abbreviate $@_{\mathtt{i}}\mathtt{i}$ for some $\mathtt{i} \in \{On, Off\}$. Similarly, the last axiom of Figure 12 expresses requirement (R4).

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 *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

> *(R5) 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 $\mathcal{H}PL$-sentence

$$\neg full \Rightarrow \langle after \rangle alarm.$$

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.

spec HIERCHSWITCH **in** $\mathbf{H}\mathcal{H}TRM =$
Nom
$On; Off$
Modal
$TurnOn{:}\mathbf{1}; TurnOff{:}\mathbf{1}$
BaseSig
$\underline{\mathrm{Nom}} = \{Nor, Sus, Test, Dead\}$
$\underline{\Lambda} = \{GoSusp : \mathbf{1}, GoRes : \mathbf{1}, ?ready : \mathbf{1}\}$
Axioms
$@_{On}(\underline{Sus} \vee \underline{Nor})$
$@_{Off}(\underline{Dead} \vee \underline{Test})$
$@_{On}\underline{@_{Nor}\langle GoSusp\rangle Sus \wedge @_{Sus}\langle Res\rangle Nor}$
$\langle TurnOn\rangle \mathbf{true} \Rightarrow Off \wedge \underline{Test}$
$On \wedge (\underline{Nor} \vee \underline{Sus}) \Leftrightarrow \langle TurnOff\rangle(Off \wedge \underline{Dead})$
end

Figure 12: A specification in $\mathcal{H}^2PL$.

To go in this direction, let us consider the hybridisation of $MVL_{\mathbf{3}}$, the multi-valued logic on the residuated lattice **3** with truth-values: $\{\top, u, \bot\}$. Hence, we may capture (R5) in a more realistic way by

$$(full, \top) \wedge \langle after\rangle(full, u) \Rightarrow \langle after\rangle\langle next\rangle(alarm, \top)$$

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 $\mathcal{H}MVL_{\mathsf{L}}$ 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"*.

$$(full, 1) \wedge \langle next\rangle(full, 0.5) \Rightarrow \langle next\rangle\langle next\rangle(alarm, 0.7).$$

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.

	Base Logic	$\mathcal{I}$ form.	Model Constraints	Quantifications	$\mathcal{HI}$	$\mathcal{I}2FOL$	$\mathcal{HI}2FOL$
$\mathcal{H}TRM$	–	Ex 2.3.1	free	free	Ex. 3.2.2	Ex 2.3.13	Ex 4.2.2
$\mathcal{H}PL$	Propositional L.	Ex 2.3.5	free	free/states	Ex. 3.2.1	Ex 2.3.14	Ex 4.2.1
$\mathcal{H}^2PL$	Hybrid Prop. L.	Ex 2.3.6	sharing of univ.	free	Ex. 3.2.3	Ex 4.2.1	Ex 4.2.3
$\mathcal{H}REL$	Relational L.	Ex 2.3.4	free	cts. ext.	Ex 3.2.5	Ex 4.2.5	not enc.
$\mathcal{H}REL'$	–	Ex 3.2.5	sharing of univ. and const.	cts. ext.	Ex 3.2.5	Ex 4.2.5	Ex 4.2.5
$\mathcal{H}FOL$	First-order L.	Ex 2.3.2	free	cts. ext.	Ex 3.2.4	Ex 4.2.6	not enc.
$\mathcal{H}FOL'$	–	[MFMB11]	sharing of univ.	cts. ext.	[MFMB11]	[MFMB11]	[MFMB11]
$\mathcal{H}FOLR$	–	Ex 3.2.4	sharing of univ., op. and pred.	rig. cts. ext.	Ex 3.2.4	Ex 4.2.6	Ex 4.2.6
$\mathcal{H}PAR$	–	Ex 3.2.7	sharing of univ., op. and pred.	rig. total cts ext.	Ex 3.2.7	Ex 4.2.7	Ex 4.2.7
$\mathcal{H}MVL_{\mathrm{L}}$	Multi-valued L.	Ex 2.3.8	free	const. ext	Ex 3.2.8		
$\mathcal{H}\mathtt{Alloy}$	Alloy Spec L.	[NMMB13a]	sharing univ. and const.	const. ext.		[NMMB13a]	

Table 1: Hybridisations.

7

COMPLEMENTARY TECHNICS

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 pragmatical 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, DM13]) 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 $\mathcal{H}CASL$ [1] to *CASL* was also defined. Thus, assisted proof support for $\mathcal{H}CASL$ becomes available for free. $\mathcal{H}CASL$ 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:

```
CFro'=HFor | ... ;
HFor=@n CFor' | <m> CFor' | [m] CFor' | Here n | ! n CFor' | ? n CFor';
```

where *n* is a nominal, *m* a modality, and the last two cases denote, respectively, universal and existential quantification over nominals. Figure 13 depicts a code fragment of a $\mathcal{H}CASL$ specification — the specification of the *reconfigurable calculator*, from Example 7.3.1, in the $\mathcal{H}EQ$ 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 = m # n
(n # m) # p = n # (m # p)
n<=m => 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 13: 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 14: 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 14 illustrates how the specification in $\mathcal{H}^2PL$ 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 we start with

- a set of relevant configurations named by the set of nominals Nom;
- and a family of modalities Λ to trigger reconfigurations.

Suppose, however, that in the place of a unique (static) interface (S, F), we consider

- a family $(S^i, F^i)_{i \in \mathrm{Nom}}$ of local signatures, indexed by the set of nominals.

The technique proceeds by building a presentation

$$\big(((S, TF, PF), \mathrm{Nom}, \Lambda), \Gamma\big) \in |\mathrm{Sign}^{\mathcal{H}PA^{\mathrm{pres}}}|$$

in the institution $\mathcal{H}PA^{\mathrm{pres}}$ of presentations over $\mathcal{H}PA$, 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 $i \in \mathrm{Nom}$) and those which concern only to a specific state. These two sets of operations define a (global) $\mathcal{H}PA$-signature:

$$S = \bigcup_{i \in \mathrm{Nom}} S^i$$
$$TF_{\underline{ar} \to w} = \big\{\sigma | \sigma \in \bigcap_{i \in \mathrm{Nom}} F^i\big\}$$
$$PF_{\underline{ar} \to w} = \big\{\sigma | \sigma \in (F^i)_{\underline{ar} \to w} \setminus TF_{\underline{ar} \to w}, i \in \mathrm{Nom}\big\}$$

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:

$$\begin{aligned}\Gamma = \quad & \{@_i(\forall X)\mathsf{df}(\sigma(X)) | \sigma \in (\mathsf{F}^i)_{\underline{ar}\to w} \cap \mathsf{PF}_{\underline{ar}\to w}, i \in \mathrm{Nom}\} \cup \\ & \{\neg @_i(\exists X)\mathsf{df}(\sigma(X)) | \sigma \in \mathsf{PF}_{\underline{ar}\to w} \setminus (\mathsf{F}^i)_{\underline{ar}\to w}, i \in \mathrm{Nom}\}\end{aligned}$$

Therefore, one ends up with a presentation

$$\big(\big((S, \mathbb{F}, \mathbb{PF}), \mathrm{Nom}, \Lambda\big), \Gamma\big)$$

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 $t \stackrel{e}{=} t'$ 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 $@_i(t \stackrel{e}{=} t')$. 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 *sets* (when the application may abstract over order and repetitions), and finally

iv) an algebra of *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 (*e.g.*, insertion clearly differs from one state to the other), but also a change at the *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 $\mathrm{Nom} = \{\mathsf{OM}, \mathsf{Om}, \mathsf{oM}, \mathsf{om}\}$ 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

$$\Lambda = \{\mathsf{goto\_OM}, \mathsf{goto\_Om}, \mathsf{goto\_oM}, \mathsf{goto\_om}\}.$$

Consider now the local interfaces. For $(S^{\mathsf{om}}, F^{\mathsf{om}})$ choose the usual signature of Sets comprising the set of sorts $S^{\mathsf{om}} = \{\mathsf{Elem}, \mathsf{Store}, \mathsf{Bool}\}$ and operation symbols $F^{\mathsf{om}}{}_{\mathsf{Store}} = \{\mathsf{empty}\}$, $F^{\mathsf{om}}{}_{\mathsf{Elem}\times\mathsf{Store}\to\mathsf{Bool}} = \{\mathsf{is\_in}\}$; $F^{\mathsf{om}}{}_{\mathsf{Elem}\times\mathsf{Store}\to\mathsf{Store}} = \{\mathsf{insert}\}$; and $F^{\mathsf{om}}{}_{\underline{\mathsf{ar}}\to s} = \emptyset$ for the other arities. Clearly, $(S^{\mathsf{Om}}, F^{\mathsf{Om}}) = (S^{\mathsf{om}}, F^{\mathsf{om}})$. The remaining cases need to deal with multiplicities; therefore signatures have to be enriched with new operations. Hence, $(S^{\mathsf{oM}}, F^{\mathsf{oM}})$ can be defined as $S^{\mathsf{oM}} = S^{\mathsf{om}} \uplus \{\mathsf{Nat}\}$ and $F^{\mathsf{oM}}{}_{\mathsf{Elem}\times\mathsf{Store}\to\mathsf{Nat}} = \{\mathsf{mult}\}$ and $F^{\mathsf{oM}}{}_{\underline{\mathsf{ar}}\to s} = F^{\mathsf{om}}{}_{\underline{\mathsf{ar}}\to s}$ for the other arities. Again, $(S^{\mathsf{OM}}, F^{\mathsf{OM}}) = (S^{\mathsf{oM}}, F^{\mathsf{oM}})$. Therefore, the following "global" partial signature is defined: $\big(((S, \mathsf{TF}, \mathsf{PF}), \mathrm{Nom}, \Lambda), \Gamma\big)$ taking $S = \bigcup_{i\in\mathrm{Nom}} S^i = S^{\mathsf{OM}}$, $\mathsf{TF} = F^{\mathsf{om}}$ and $\mathsf{PF}_{\mathsf{Elem}\times\mathsf{Store}\to\mathsf{Nat}} = \{\mathsf{mult}\}$ and $\mathsf{PF}_{\underline{\mathsf{ar}}\to s} = \emptyset$ for the other arities. On its turn, Γ is defined by the sentences

$@_i(\forall s)(\forall e)\,\mathsf{df}(\mathsf{mult}(e, s))$, for $i \in \{\mathsf{oM}, \mathsf{OM}\}$

$\neg @_i(\forall s)(\forall e)\,\mathsf{df}(\mathsf{mult}(e, s))$, for $i \in \{\mathsf{om}, \mathsf{Om}\}$.

In this setting, we may now proceed with the specification of the global properties, as for example,

$(\forall e : \mathsf{elem})\,\mathsf{is\_in}(e, \mathsf{empty}) = \mathsf{False}$

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 om:

$$@_{\mathsf{om}}(\forall e, e')(\forall s)\mathsf{insert}(e', \mathsf{insert}(e, s)) = \mathsf{insert}(e, \mathsf{insert}(e', s))$$

$$@_{\mathsf{om}}(\forall e)(\forall s)\, \mathsf{insert}(e, \mathsf{insert}(e, s)) = \mathsf{insert}(e, s)$$

On the other hand, the specification of mult in configuration oM is introduced as

$$@_{\mathsf{oM}}(\forall e, e')(\forall s)\neg e = e' \Rightarrow \mathsf{mult}(e, \mathsf{insert}(e', s)) = \mathsf{mult}(e, s)$$

$$@_{\mathsf{oM}}(\forall e)\mathsf{mult}(e, \mathsf{empty}) = 0.$$

Finally, we have to specify the possible reconfigurations. For this, one may use sentences as direct as

$$@_{\mathsf{om}}\langle \mathsf{goto\_OM} \rangle \mathsf{OM}$$

stating that a reconfiguration from om to OM is possible, or opt for more elaborated forms as *e.g.*,

$$(\forall e, e')(\forall s)\, \mathsf{insert}(e', \mathsf{insert}(e, s)) \;=\; \mathsf{insert}(e, \mathsf{insert}(e', s))$$
$$\Rightarrow \langle \mathsf{goto\_Om} \rangle \mathsf{Om}.$$

The latter states the system can evolve to configuration Om (through the event $\mathsf{goto\_Om}$) 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 *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 a arbitrary hybridised institution $\mathcal{HI}$ 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 λ can be seen as an event (denoted by) λ' such that its realisation $R_{\lambda'} \subseteq R_\lambda$ satisfies the guard.

Suppose, for example, one wants to specify a transition identified by λ and guarded by a sentence ρ from a state called i into a state called j. Assuming that λ' is a sub-modality of λ, i.e., λ' is such that

$$(\forall i, j)@_i\langle\lambda'\rangle j \Rightarrow @_i\langle\lambda\rangle j,$$

this can be expressed by

$$@_i\rho \Rightarrow (\exists\lambda')@_i[\lambda']j.$$

Therefore the intended guarded transition can be specified by the sentence

$$@_i\rho \Rightarrow \big[(\exists\lambda')\big((\forall i, j)(@_i\langle\lambda'\rangle j \Rightarrow @_i\langle\lambda\rangle j) \wedge @_i[\lambda']j\big)\big].$$

Of course, if ρ is a quantified sentence, the particular Kripke substructure corresponding to λ' becomes defined on the respective expansions of the model. Each one of these substructures specify an *enabling* relation.

Abbreviating $\lambda' \subseteq \lambda \stackrel{\text{abr}}{=} (\forall i, j)@_i\langle\lambda'\rangle j \Rightarrow @_i\langle\lambda\rangle i$, we write

$$@_i\rho \Rightarrow (\exists\lambda')\big(\lambda' \subseteq \lambda \wedge @_i[\lambda']j\big).$$

Along the same lines we may express requirements involving patterns like "property ρ enables a reconfiguration into j"

$$\rho \Rightarrow (\exists \lambda')\big(\lambda' \subseteq \lambda \wedge [\lambda']j\big),$$

or, more generally, "property ρ enables a move to configurations satisfying property ρ'", by

$$\rho \Rightarrow (\exists \lambda')\big(\lambda' \subseteq \lambda \wedge [\lambda']\rho'\big).$$

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 $\mathtt{sum}$ where the $\star$ is interpreted as addition, and a mode $\mathtt{mult}$ where $\star$ behaves as multiplication. There is also an event $\mathtt{shift}$ 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 $(\mathrm{Nat}_\star, \{\mathtt{sum}, \mathtt{mult}\}, \{\mathtt{shift}\})$, where $\mathrm{Nat}_\star$ 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 $\mathtt{shift}$ is a unary modality symbol. In such a setting we are able to relate modes through their local properties. For instance,

$$(\forall m : \mathtt{nat}) \star (m, 0) = 0 \Rightarrow [\mathtt{shift}](\forall m : \mathtt{nat}) \star (m, 0) = m$$

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

$$\chi : ((S, S_0, F, F_0, P, P_0), \mathrm{Nom}, \Lambda) \hookrightarrow ((S, S_0, F, F_0 + X, P, P_0), \mathrm{Nom}, \Lambda)$$

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 $\mathsf{Nat}_\star$ above to make both the sort and the successor function rigid, say into a signature $\mathsf{NatR}_\star$, we are able to specify multiplication by means of sums:

$$\begin{aligned}&(\forall m, n : \mathsf{nat})(\exists w, z : \mathsf{nat})\\ &\quad @_{\mathsf{mult}}\big(\star(\mathsf{suc}(n), m) = z \wedge \star(n, m) = w \wedge [\mathsf{shift}]\star(w, m) = z\big)\end{aligned}$$

Note that the variable z plays the role of a "communicating bridge" between two different modes, or configurations, of this simple system.

$\circ$

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 mult-mode is enabled when $a \geqslant b$. Otherwise, the only enabled reconfiguration is to the sum-mode.

As mentioned above, enabling transitions are particular sub-relations of the accessibility relation. For instance, a reconfiguration into the mult-mode can be specified by

$$(\forall a, b : \mathsf{nat})\ a \geqslant b \Rightarrow (\exists \mathsf{shift}')\big(\mathsf{shift}' \subseteq \mathsf{shift} \wedge [\mathsf{shift}']\mathsf{mult}\big).$$

Reconfigurations into the sum-mode can be specified, analogously, by

$$(\forall a, b : \mathsf{nat})\ a < b \Rightarrow (\exists \mathsf{shift}')\big(\mathsf{shift}' \subseteq \mathsf{shift} \wedge [\mathsf{shift}']\mathsf{sum}\big).$$

For instance, let us consider the hybrid structure $\mathcal{M} = (M, R)$ where M_{mult} and M_{sum} are realized by the natural numbers with the usual multiplication and sum operations respectively, $|R| = \{s_\times, s_+\}$, $R_{mult} = s_\times$ and $R_{sum} = s_+$. Assume R_{shift} as the universal relation over $|R|$. Consider also a $\{a, b\}$-expansion of $\mathcal{M}$, say $\mathcal{M}^{a \geqslant b}$, such that $M^{a \geqslant b} \models a \geqslant b$, i.e., $M_a^{a \geqslant b} \geqslant M_b^{a \geqslant b}$. Finally, consider a $shift'$-expansion $\mathcal{M}^{a \geqslant b, shift'}$ such that $R_{shift'}^{a \geqslant b, shift'} = \{(s_+, s_\times), (s_\times, s_\times)\}$. Since, $R_{shift'} \subseteq R_{shift}$ and $\mathcal{M}^{a \geqslant b, shift'} \models [shift']mult$, we have

$$\mathcal{M}^{a \geqslant b} \models (\exists shift')\big[shift' \subseteq shift \wedge [shift']mult\big].$$

Similarly, for $\{a, b\}$-expansions where $M_a^{a<b} < M_b^{a<b}$, we have

$$\mathcal{M}^{a<b} \models (\exists shift')\big[shift \subseteq shift' \wedge [shift']mult\big].$$

Therefore,

$$\mathcal{M} \models (\forall a, b : nat) a \geqslant b \Rightarrow (\exists shift')\big[shift \subseteq [shift'] \wedge [shift]mult\big].$$

Analogously, taking $R_{shift'}^{a<b, shift'} = \{(s_+, s_+), (s_\times, s_+)\}$ we have

$$\mathcal{M} \models (\forall a, b : nat) a < b \Rightarrow (\exists shift')\big[shift \subseteq shift' \wedge [shift']mult\big].$$

Hence $\mathcal{M}$ is a model for the given specification. ∘

8

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 specfication method introduced in the thesis. For instance, one could extend the lifting of comorphisms from $\mathcal{I}2FOL^{\text{pres}}$ to $\mathcal{HI}2FOL^{\text{pres}}$, into the lifting of comorphisms $\mathcal{HI}2CASL^{\text{pres}}$ to $\mathcal{HI}2CASL^{\text{pres}}$ 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, DT11]), 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, resorting, for example, to real, industrial case studies, remains to be done.

EPILOGUE

há um rio nesta água

Ana

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, focussed on *hidden algebra* and *observational equality* [Mad08]. Latter 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, Vou10] as a suitable model for reconfigurable systems. A few initial ideas were presented in [MMB11].

BIBLIOGRAPHY

[AB01] Carlos Areces and Patrick Blackburn. Bringing them all together. *J. Log. Comput.*, 11(5):657–669, 2001.

[ABK+02] Egidio Astesiano, Michel Bidoit, Hélène Kirchner, Bernd Krieg-Brückner, Peter D. Mosses, Donald Sannella, and Andrzej Tarlecki. CASL: the common algebraic specification language. *Theor. Comput. Sci.*, 286(2):153–196, 2002.

[ABM99] Carlos Areces, Patrick Blackburn, and Maarten Marx. A road-map on complexity for hybrid logics. In J. Flum and M. Rodríguez-Artalejo, editors, *13th Intern. Workshop Computer Science Logic (CSL'99, Madrid, Spain, September 20-25, 1999)*, volume 1683 of *Lecture Notes in Computer Science*, pages 307–321. Springer, 1999.

[ACEGG90] Jaume Agustí-Cullell, Francesc Esteva, Pere Garcia, and Lluis Godo. Formalizing multiple-valued logics as institutions. In B. Bouchon-Meunier, R. R. Yager, and L. A. Zadeh, editors, *3rd Intern. Conf. on Information Processing and Management of Uncertainty in Knowledge-Based Systems (IPMU 90, Paris, France, July 2-6, 1990)*, volume 521 of *Lecture Notes in Computer Science*, pages 269–278. Springer, 1990.

[ACP06] Horacio L. Arló-Costa and Eric Pacuit. First-order classical modal logic. *Studia Logica*, 84(2):171–210, 2006.

[AHS90] Jiri Adamek, Horst Herrlich, and George E. Strecker. *Abstract and concrete categories : The joy of cats*. Pure and Applied Mathematics. John Wiley & Sons, New York, 1990.

[AILS07] Luca Aceto, Anna Ingólfdótir, Kim G. Larsen, and Jiri Srba. *Reactive Systems: Modelling, specification and verification*. Cambridge University Press, 2007.

[AM89] Peter Aczel and Nax Paul Mendler. A final coalgebra theorem. In D. H. Pitt, D. E. Rydeheard, P. Dybjer, A. M. Pitts, and A. Poigné, editors, *Category Theory and Computer Science (Manchester, UK, September 5-8, 1989)*, volume 389 of *Lecture Notes in Computer Science*, pages 357–365. Springer, 1989.

[AM02] Nazareno Aguirre and Tom S. E. Maibaum. A temporal logic approach to the specification of reconfigurable component-based systems. In *17th IEEE Intern. Conf. on Automated Software Engineering (ASE 2002, 23-27 September 2002, Edinburgh, Scotland, UK)*, pages 271–274, 2002.

[Arb04] Farhad Arbab. Reo: a channel–based coordination model for component composition. *Mathematical Structures in Comp. Sci.*, 14(3):329–366, 2004.

[AtC07] C. Areces and B. ten Cate. Hybrid logics. In P. Blackburn, F. Wolter, and J. van Benthem, editors, *Handbook of Modal Logic*, Studies in Logic and Practical Reasoning (volume 3), pages 822–868. Elsevier, 2007.

[Awo06] Steve Awodey. *Category Theory*. Oxford Logic Guides. Oxford University Press, 2006.

[Bal10] Pedro Baltazar. *Probabilization of Logic Systems*. PhD thesis, Instituto Superior Técnico de Lisboa, 2010.

[BB06] Thomas Bolander and Torben Braüner. Tableau-based decision procedures for hybrid logic. *J. Log. Comput.*, 16(6):737–763, 2006.

[BBG+08] Roberto Bruni, Antonio Bucchiarone, Stefania Gnesi, Dan Hirsch, and Alberto Lluch-Lafuente. Graph-based design and analysis of dynamic software architectures.

In P. Degano, R. De Nicola, and J. Meseguer, editors, *Concurrency, Graphs and Models (Essays Dedicated to Ugo Montanari on the Occasion of His 65th Birthday)*, volume 5065 of *Lecture Notes in Computer Science*, pages 37–56. Springer, 2008.

[BBS88] David B. Benson and Ofer Ben-Shachar. Bisimulation of automata. *Inf. Comput.*, 79(1):60–83, 1988.

[BCL+06] Eric Bruneton, Thierry Coupaye, Matthieu Leclercq, Vivien Quéma, and Jean-Bernard Stefani. The Fractal component model and its support in java. *Softw., Pract. Exper.*, 36(11-12):1257–1284, 2006.

[BD94] Rod Burstall and Răzvan Diaconescu. Hiding and behaviour: an institutional approach. In William Roscoe, editor, *A Classical Mind: Essays in Honour of C.A.R. Hoare*, pages 75–92. Prentice-Hall, 1994.

[BdP06] Torben Braüner and Valeria de Paiva. Intuitionistic hybrid logic. *J. Applied Logic*, 4(3):231–255, 2006.

[BdRV01] Patrick Blackburn, Maarten de Rijke, and Yde Venema. *Modal Logic*. Number 53 in Cambridge Tracts in Theoretical Computer Science. Cambridge University Press, 2001.

[BFH+12] Howard Barringer, Yliès Falcone, Klaus Havelund, Giles Reger, and David E. Rydeheard. Quantified event automata: Towards expressive and efficient runtime monitors. In D. Giannakopoulou and D. Mery, editors, *Formal Methods (FM-12, Paris, France, August 27-31, 2012)*, volume 7436 of *Lecture Notes in Computer Science*, pages 68–84. Springer, 2012.

[BFT06] Peter Baumgartner, Alexander Fuchs, and Cesare Tinelli. Implementing the model evolution calculus. *International Journal on Artificial Intelligence Tools*, 15(1):21–52, 2006.

[BG80] Rod M. Burstall and Joseph A. Goguen. The semantics of CLEAR, a specification language. In D. Bjørner, editor, *Abstract Software Specifications (1979 Copenhagen Winter School, January 22 - February 2, 1979)*, volume 86 of *Lecture Notes in Computer Science*, pages 292–332. Springer, 1980.

[BH06] Michel Bidoit and Rolf Hennicker. Constructor-based observational logic. *J. Log. Algebr. Program.*, 67(1-2):3–51, 2006.

[BH08] Michel Bidoit and Rolf Hennicker. An algebraic semantics for contract-based software components. In J. Meseguer and G. Rosu, editors, *Algebraic Methodology and Software Technology (AMAST 2008 - Urbana, IL, USA, July 28-31, 2008*, volume 5140 of *Lecture Notes in Computer Science*, pages 216–231. Springer, 2008.

[BHW11] Sebastian S. Bauer, Rolf Hennicker, and Martin Wirsing. Interface theories for concurrency and data. *Theor. Comput. Sci.*, 412(28):3101–3121, 2011.

[Bjø06a] Dines Bjørner. *Software Engineering 2: Specification of Systems and Languages*. Monographs in Theoretical Computer Science, an EATCS Series. Springer Verlag, 2006.

[Bjø06b] Dines Bjørner. *Software Engineering (volumes 1, 2 and 3)*. Monographs in Theoretical Computer Science, an EATCS Series. Springer Verlag, 2006.

[BKI05] Christoph Beierle and Gabriele Kern-Isberner. Looking at probabilistic conditionals from an institutional point of view. In G. Kern-Isberner, W. Rödder, and F. Kulmann, editors, *Conditionals, Information, and Inference (Revised Selected Papers of WCII 2002, Hagen, Germany, May 13-15, 2002)*, volume 3301 of *Lecture Notes in Computer Science*, pages 162–179. Springer, 2005.

[Bla00] Patrick Blackburn. Representation, reasoning, and relational structures: a hybrid logic manifesto. *Logic Journal of IGPL*, 8(3):339–365, 2000.

[Bla06] Patrick Blackburn. Arthur Priorand hybrid logic. *Synthese*, 150(3):329–372, 2006.

[BLLM08] Roberto Bruni, Alberto Lluch-Lafuente, and Ugo Montanari. Style-based architectural reconfigurations. *Bulletin of the EATCS*, 94:161–180, 2008.

[Bor99] Tomasz Borzyszkowski. Moving specification structures between logical systems. In J. L. Fiadeiro, editor, *Recent Trends in Algebraic Development Techniques (Revised Selected Papers of WADT '98, Lisbon, Portugal, April, 2-4, 1998)*, volume 1589 of *Lecture Notes in Computer Science*, pages 16–30. Springer, 1999.

[Bor02] Tomasz Borzyszkowski. Logical systems for structured specifications. *Theoretical Computer Science*, 286(2):197–245, 2002.

[BPT12] Peter Baumgartner, Björn Pelzer, and Cesare Tinelli. Model evolution with equality - revised and implemented. *J. Symb. Comput.*, 47(9):1011–1045, 2012.

[Bra05] Torben Braüner. Natural deduction for first-order hybrid logic. *Journal of Logic, Language and Information*, 14(2):173–198, 2005.

[Bra10] Torben Brauner. *Hybrid Logic and its Proof-Theory*. Applied Logic Series. Springer, 2010.

[BS03] Egon Börger and Robert F. Stärk. *Abstract State Machines. A Method for High-Level System Design and Analysis*. Springer, 2003.

[BT98] Patrick Blackburn and Miroslava Tzakova. Hybrid completeness. *Logic Journal of the IGPL*, 6(4):625–650, 1998.

[BVB07] Patrick Blackburn and Johan Van Benthem. Modal logic: a semantic perspective. In P. Blackburn, F. Wolter, and J. van Benthem, editors, *Handbook of Modal Logic*, Studies in Logic and Practical Reasoning (volume 3), pages 1–82. Elsevier, 2007.

[BW00] Manfred Broy and Martin Wirsing. Algebraic state machines. In T. Rus, editor, *Algebraic Methodology and Software Technology (AMAST 2000, Iowa City, Iowa, USA, May 20-27, 2000)*, volume 1816 of *Lecture Notes in Computer Science*, pages 89–188. Springer, 2000.

[CÔ6] Corina Cîrstea. An institution of modal logics for coalgebras. *J. Log. Algebr. Program.*, 67(1-2):87–113, 2006.

[CAPM10] Pablo F. Castro, Nazareno Aguirre, Carlos Gustavo Lopez Pombo, and T. S. E. Maibaum. Towards managing dynamic reconfiguration of software systems in a categorical setting. In A.Cavalcanti, D. Deharbe, M.-C. Gaudel, and J. Woodcock, editors, *Theoretical Aspects of Computing (ICTAC 2010, Natal, Rio Grande do Norte, Brazil, September 1-3, 2010*, volume 6255 of *Lecture Notes in Computer Science*, pages 306–321. Springer, 2010.

[CCK11] Yongzhi Cao, Guoqing Chen, and Etienne E. Kerre. Bisimulations for fuzzy-transition systems. *IEEE T. Fuzzy Systems*, 19(3):540–552, 2011.

[CCS+13] Andreas Classen, Maxime Cordy, Pierre-Yves Schobbens, Patrick Heymans, Axel Legay, and Jean-Francois Raskin. Featured transition systems: Foundations for verifying variability-intensive systems and their application to ltl model checking. *IEEE Transactions on Software Engineering*, 99(PrePrints):1, 2013.

[Cen98] M. Victoria Cengarle. The temporal logic institution. Technical Report Technical Report 9805, Ludwig-

Maximilians-Universität München, Institut für Informatik, November 1998.

[CG08] Mihai Codescu and Daniel Găină. Birkhoff completeness in institutions. *Logica Universalis*, 2(2):277–309, 2008.

[CGRH01] Andrea Corradini, Martin Große-Rhode, and Reiko Heckel. A coalgebraic presentation of structured transition systems. *Theor. Comput. Sci.*, 260(1-2):27–55, 2001.

[Cla08] Dave Clarke. A basic logic for reasoning about connector reconfiguration. *Fundam. Inform.*, 82(4):361–390, 2008.

[CM92] Andrea Corradini and Ugo Montanari. An algebraic semantics for structured transition systems and its applications to logic programs. *Theor. Comput. Sci.*, 103(1):51–106, 1992.

[CMS07] Giovanni Conforti, Damiano Macedonio, and Vladimiro Sassone. Static BiLog: a unifying language for spatial structures. In W. Penczek and G. Rozenberg, editors, *Half a century of inspirational research, honouring the scientific influence of Antoni Mazurkiewicz*, volume 80 of *Fundamenta Informaticae*, pages 91–110. IOS Press, 2007.

[CMSS06] Carlos Caleiro, Paulo Mateus, Amílcar Sernadas, and Cristina Sernadas. Quantum institutions. In K. Futatsugi, J.-P. Jouannaud, and J. Meseguer, editors, *Algebra, Meaning, and Computation, Essays Dedicated to Joseph A. Goguen on the Occasion of His 65th Birthday*, volume 4060 of *Lecture Notes in Computer Science*, pages 50–64. Springer, 2006.

[CR97] Gerardo Costa and Gianna Reggio. Specification of abstract dynamic-data types: A temporal logic approach. *Theor. Comput. Sci.*, 173(2):513–554, 1997.

[CS02] Giulianella Coleti and Romano Scozzafava. *Probabilistic Logic in a Coherent Setting*. Trends in Logic, Studia Logica Library (volume 15). Kluwer Academic Publishers, 2002.

[CSS99] Carlos Caleiro, Cristina Sernadas, and Amílcar Sernadas. Parameterisation of logics. In J. L. Fiadeiro, editor, *Recent Trends in Algebraic Development Techniques (Revised Selected Papers of WADT '98, Lisbon, Portugal, April, 2-4, 1998)*, volume 1589 of *Lecture Notes in Computer Science*, pages 48–62. Springer, 1999.

[CZZ12] Roberto Di Cosmo, Stefano Zacchiroli, and Gianluigi Zavattaro. Towards a formal component model for the cloud. In G. Eleftherakis, M. Hinchey, and M. Holcombe, editors, *Software Engineering and Formal Methods (SEFM 2012, Thessaloniki, Greece, October 1-5, 2012)*, volume 7504 of *Lecture Notes in Computer Science*, pages 156–171. Springer, 2012.

[DF02] Răzvan Diaconescu and Kokichi Futatsugi. Logical foundations of CafeOBJ. *Theor. Comput. Sci.*, 285:289–318, 2002.

[DG07] Manfred Droste and Paul Gastin. Weighted automata and weighted logics. *Theor. Comput. Sci.*, 380(1-2):69–86, June 2007.

[Dia98] Răzvan Diaconescu. Extra theory morphisms for institutions: Logical semantics for multi-paradigm languages. *Applied Categorical Structures*, 6(4):427–453, 1998.

[Dia02] Răzvan Diaconescu. Grothendieck institutions. *Applied Categorical Structures*, 10(4):383–402, 2002.

[Dia08] Răzvan Diaconescu. *Institution-independent Model Theory*. Studies in Universal Logic. Birkhäuser Basel, 2008.

[Dia09] Răzvan Diaconescu. An encoding of partial algebras as total algebras. *Inf. Process. Lett.*, 109(23-24):1245–1251, 2009.

[Dia10a] Răzvan Diaconescu. Quasi-boolean encodings and conditionals in algebraic specification. *J. Log. Algebr. Program.*, 79(2):174–188, 2010.

[Dia10b] Răzvan Diaconescu. Three decades of institution theory. In J.-Y. Béziau, editor, *An anthology of universal logic (from Paul Hertz to Dov Gabbay)*, Studies in Universal Logic, pages 309–322. Birkhäuser Basel, 2010.

[Dia11] Răzvan Diaconescu. On quasi-varieties of multiple valued logic models. *Math. Log. Q.*, 57(2):194–203, 2011.

[Dia12a] Răzvan Diaconescu. Borrowing interpolation. *J. Log. Comput.*, 22(3):561–586, 2012.

[Dia12b] Răzvan Diaconescu. Interpolation for predefined types. *Mathematical Structures in Computer Science*, 22(1):1–24, 2012.

[Dia13] Răzvan Diaconescu. Quasi-varieties and initial semantics in hybridized institutions. *Journal of Logic and Computation*, 2013. in print.

[DKL10] Julien Dormoy, Olga Kouchnarenko, and Arnaud Lanoix. Using temporal logic for dynamic reconfigurations of components. In L. S. Barbosa and M. Lumpe, editors, *Formal Aspects of Component Software (Revised Selected Papers of FACS 2010, Guimaraes, Portugal, October 14-16, 2010)*, volume 6921 of *Lecture Notes in Computer Science*, pages 200–217. Springer, 2010.

[DLN07] Stéphane Demri, Ranko Lazić, and David Nowak. On the freeze quantifier in constraint ltl: Decidability and complexity. *Inf. Comput.*, 205(1):2–24, 2007.

[DM13] Răzvan Diaconescu and Alexandre Madeira. Encoding hybridized institutions into first order logic. (Submited), 2013.

[DMV90] Rocco De Nicola, Ugo Montanari, and Frits W. Vaandrager. Back and forth bisimulations. In J. C. M. Baeten and J. W. Klop, editors, *Theories of Concurrency: Unification and Extension (CONCUR'90, Amsterdam, The Netherlands, August 27-30, 1990)*, volume 458 of *Lecture Notes in Computer Science*, pages 152–165. Springer, 1990.

[Dob10] Ernst-Erich Doberkart. *Stochastic Coalgebraic Logic*. Monographs in Theoretical Computer Science, an EATCS Series. Springer, 2010.

[DP06] Răzvan Diaconescu and Marius Petria. Abstract beth definability in institutions. *J. Symb. Log.*, 71(3):1002–1028, 2006.

[DS07] Răzvan Diaconescu and Petros S. Stefaneas. Ultraproducts and possible worlds semantics in institutions. *Theor. Comput. Sci.*, 379(1-2):210–230, 2007.

[DT11] Răzvan Diaconescu and Ionut Tutu. On the algebra of structured specifications. *Theor. Comput. Sci.*, 412(28):3145–3174, 2011.

[EM85] Hartmut Ehrig and Bernd Mahr. *Fundamentals of Algebraic Specification 1: Equations and Initial Semantics*. Monographs in Theoretical Computer Science, an EATCS Series. Springer Verlag, 1985.

[FF02] Rogerio Fajardo and Marcelo Finger. Non-normal modalisation. In P. Balbiani, N.-Y. Suzuki, F. Wolter, and M. Zakharyaschev, editors, *Advances in Modal Logic 4 (Toulouse, France, October 2002)*, pages 83–96, 2002.

[FG92] Marcelo Finger and Dov Gabbay. Adding a temporal dimension to a logic adding a temporal dimension to a

logic system. *Journal of Logic, Language and Information*, 1(3), 1992.

[FL13] Jose Luiz Fiadeiro and Antonia Lopes. A model for dynamic reconfiguration in service-oriented architectures. *Software and System Modeling*, 12(2):349–367, 2013.

[FMFR11] Yliès Falcone, Laurent Mounier, Jean-Claude Fernandez, and Jean-Luc Richier. Runtime enforcement monitors: composition, synthesis, and enforcement abilities. *Formal Methods in System Design*, 38(3):223–262, 2011.

[FS01] Alain Finkel and Ph. Schnoebelen. Well-structured transition systems everywhere! *Theor. Comput. Sci.*, 256(1-2):63–92, 2001.

[GB92] Joseph A. Goguen and Rod M. Burstall. Institutions: Abstract model theory for specification and programming. *J. ACM*, 39(1):95–146, 1992.

[GGM76] V. Giarratana, F. Gimona, and Ugo Montanari. Observability concepts in abstract data type specifications. In A. W. Mazurkiewicz, editor, *Mathematical Foundations of Computer Science (MFCS, Gdansk, Poland, September 6-10, 1976)*, volume 45 of *Lecture Notes in Computer Science*, pages 576–587. Springer, 1976.

[GKS10] Daniel Götzmann, Mark Kaminski, and Gert Smolka. Spartacus: A tableau prover for hybrid logic. *Electr. Notes Theor. Comput. Sci.*, 262:127–139, 2010.

[GM87] Joseph A. Goguen and Jose Meseguer. Models and equality for logical programming. In H. Ehrig, R. A. Kowalski, G. Levi, and U. Montanari, editors, *Intern. Joint Conference on Theory and Practice of Software Development (TAPSOFT, Vol. 1, Pisa, Italy, March 23-27, 1987)*, volume 250 of *Lecture Notes in Computer Science*, pages 1–22. Springer, 1987.

[GM00] Joseph A. Goguen and Grant Malcolm. A hidden agenda. *Theor. Comput. Sci.*, 245(1):55–101, 2000.

[GO07] V. Goranko and M. Otto. Model theory of modal logic. In P. Blackburn, F. Wolter, and J. van Benthem, editors, *Handbook of Modal Logic*, pages 249–329. Elsevier, 2007.

[Gor96] Valentin Goranko. Hierarchies of modal and temporal logics with reference pointers. *Journal of Logic, Language and Information*, 5(1):1–24, 1996.

[Got01] Siegfried Gottwald. *A Treatise on Many-Valued Logics*. Studies in Logic and Computation (volume 9). Research Studies Press, 2001.

[GR02] Joseph A. Goguen and Grigore Rosu. Institution morphisms. *Formal Asp. Comput.*, 13(3-5):274–307, 2002.

[Grä79] George Grätzer. *Universal Algebra*. New York, Heidelberg, Berlin: Springer-Verlag, 1979.

[Gro] O. M. G. Group. UML Specification, Version 2.0.

[Gur94] Yuri Gurevich. Evolving algebras. In *IFIP Congress (1)*, pages 423–427, 1994.

[HA09] Guillaume Hoffmann and Carlos Areces. Htab: a terminating tableaux system for hybrid logic. *Electr. Notes Theor. Comput. Sci.*, 231:3–19, 2009.

[Har87] David Harel. Statecharts: A visual formalism for complex systems. *Sci. Comput. Program.*, 8(3):231–274, 1987.

[HB98] Rolf Hennicker and Michel Bidoit. Observational logic. In Armando Martin Haeberer, editor, *AMAST*, volume 1548 of *Lecture Notes in Computer Science*, pages 263–277. Springer, 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, Nancy, France, 9-12 September 2008)*, pages 111–132. College Publications, 2008.

[Hen96] Thomas A. Henzinger. The theory of hybrid automata. In *11th Annual IEEE Symposium on Logic in Computer Science (LICS'96, New Brunswick, New Jersey, USA, July 27-30, 1996)*, pages 278–292, 1996.

[HMP01] Annegret Habel, Jürgen Müller, and Detlef Plump. Double-pushout graph transformation revisited. *Mathematical Structures in Computer Science*, 11(5):637–688, 2001.

[Hod93] Wilfrid Hodges. *Model Theory*. Cambridge University Press, 1993.

[Hod97] Wilfrid Hodges. *A shorter model theory*. Cambridge University Press, 1997.

[Hod10] Ian Hodkinson. The bounded fragment and hybrid logic with polyadic modalities. *Review of Symbolic Logic*, 3:279–286, 2010.

[HRWW11] Rolf Haenni, Jan-Willem Romeijn, Gregory Wheeler, and Jon Williamsont. *Probabilistic Logics and Probabilistic Networks*. Synthese Library: Studies in Epistemology, Logic, Methodology, and Philosophy of Science (volume 350). Springer, 2011.

[HS08] Christian Hoareau and Ichiro Satoh. Hybrid logics and model checking: A recipe for query processing in location-aware environments. In *22nd Intern. Conf. on Advanced Information Networking and Applications (AINA 2008, GinoWan, Okinawa, Japan, March 25-28, 2008)*, pages 130–137. IEEE Computer Society, 2008.

[Ind07] Andrzej Indrzejczak. Modal hybrid logic. *Logic and Logical Philosophy*, 16:147–257, 2007.

[Jac11] Daniel Jackson. *Software Abstractions (Logic, Language, and Analysis)*. MIT Press, 2nd edition, 2011.

[JSHS96] Ralf Jungclaus, Gunter Saake, Thorsten Hartmann, and Cristina Sernadas. TROLL - a language for object-oriented specification of information systems. *ACM Trans. Inf. Syst.*, 14(2):175–211, 1996.

[KLSW09] Ahmet Kara, Martin Lange, Thomas Schwentick, and Volker Weber. On the hybrid extension of CTL and CTL+. *CoRR*, abs/0906.2541, 2009.

[KMLA11] Christian Krause, Ziyan Maraikar, Alexander Lazovik, and Farhad Arbab. Modeling dynamic reconfigurations in reo using high-level replacement systems. *Sci. Comput. Program.*, 76(1):23–36, 2011.

[Lan98] Saunders M. Lane. *Categories for the Working Mathematician (Graduate Texts in Mathematics)*. Springer, 2nd edition, 1998.

[Lan09] Martin Lange. Model checking for hybrid logic. *J. of Logic, Lang. and Inf.*, 18(4):465–491, 2009.

[LSVW95] Nancy A. Lynch, Roberto Segala, Frits W. Vaandrager, and Henri B. Weinberg. Hybrid i/o automata. In R. Alur, T. A. Henzinger, and E. D. Sontag, editors, *Hybrid Systems III: Verification and Control (DIMACS/SYCON Workshop, October 22-25, 1995, Ruttgers University, New Brunswick, NJ, USA)*, volume 1066 of *Lecture Notes in Computer Science*, pages 496–510. Springer, 1995.

[Mad08] Alexandre Madeira. Algebraic approach to the observational equality. Master's thesis, Dep Matemática, Universidade de Aveiro, 2008.

[Mar06] Manuel A. Martins. Behavioral institutions and refinements in generalized hidden logics. *J. UCS*, 12(8):1020–1049, 2006.

[McC10] W. McCune. Prover9 and Mace4. `www.cs.unm.edu/~mccune/prover9/`, 2005–2010.

[MDT09] Till Mossakowski, Răzvan Diaconescu, and Andrzej Tarlecki. What is a logic translation. *Logica Universalis*, 3(1):95–124, 2009.

[Mes89] Jose Meseguer. General logics. In H. D. Ebbinghaus, J. Fernandez-Prida, M. Garrido, D. Lascar, and M. Rodriguez-Artalejo, editors, *Logic Colloqium'87*, Studies in Logic and the Foundations of Mathematics (volume 129), pages 275–330. Elsevier, 1989.

[Mes92] Jose Meseguer. Conditioned rewriting logic as a united model of concurrency. *Theor. Comput. Sci.*, 96(1):73–155, 1992.

[Mes12] Jose Meseguer. Twenty years of rewriting logic. *J. Log. Algebr. Program.*, 81(7-8):721–781, 2012.

[MFMB11] Alexandre Madeira, José M. Faria, Manuel A. Martins, and Luís Soares Barbosa. Hybrid specification of reactive systems: An institutional approach. In G. Barthe, A. Pardo, and G. Schneider, editors, *Software Engineering and Formal Methods (SEFM 2011, Montevideo, Uruguay, November 14-18, 2011)*, volume 7041 of *Lecture Notes in Computer Science*, pages 269–285. Springer, 2011.

[MHST03] Till Mossakowski, Anne Haxthausen, Donald Sannella, and Andrzej Tarlecki. CASL: The common algebraic specification language: Semantics and proof theory. *Computing and Informatics*, 22:285–321, 2003.

[Mil80] R. Milner. *A Calculus of Communicating Systems*, volume 92 of *Lecture Notes in Computer Science*. Springer, 1980.

[Mil89] R. Milner. *Communication and Concurrency*. Series in Computer Science. Prentice-Hall, 1989.

[Mil09] Robin Milner. *The Space and Motion of Communicating Agents*. Cambridge University Press, 2009.

[MMB09a] Manuel A. Martins, Alexandre Madeira, and Luís S. Barbosa. Refinement by interpretation in a general setting. *Electron. Notes Theor. Comput. Sci.*, 259:105–121, 2009.

[MMB09b] Manuel A. Martins, Alexandre Madeira, and Luís S. Barbosa. Refinement via interpretation. In D. V. Hung and P. Krishnan, editors, *Seventh IEEE International Conference on Software Engineering and Formal Methods (SEFM 2009, Hanoi, Vietnam, 23-27 November 2009)*, pages 250–259. IEEE Computer Society, 2009.

[MMB11] Alexandre Madeira, Manuel A. Martins, and Luís S. Barbosa. Models as arrows: the role of dialgebras. In *Models of Computation in Context (CiE'11- Computability in Europe, Sofia, Bulgaria, 27 June - 2 July)*, pages 144–153. St. Kliment Ohridski University Press, 2011.

[MMB12] Manuel A. Martins, Alexandre Madeira, and Luis S. Barbosa. Towards a semantics for infinitary equational hybrid logic. In *Proceedings of AiML'12: Advances in Modal Logic*, pages 42–46, 2012.

[MMB13a] Alexandre Madeira, Manuel A. Martins, and Luís S. Barbosa. Bisimilarity and refinement for hybrid(ised) logics. In *2013 Refinement Workshop*, volume (to appear) of *Electronic Proceedings in Theoretical Computer Science*, 2013.

[MMB13b] Manuel A. Martins, Alexandre Madeira, and Luís Soares Barbosa. A coalgebraic perspective on logical interpretations. *Studia Logica*, 101(4):783–825, 2013.

[MMCL13] Till Mossakowski, Christian Maeder, Mihai Codescu, and Dominik Lucke. HETS User Guide - Version 0.99. Technical report, DFKI Lab Bremen, April 2013.

[MMDB11a] Manuel A. Martins, Alexandre Madeira, Răzvan Diaconescu, and Luís Soares Barbosa. Hybridization of institutions. In A. Corradini, B. Klin, and C. Cîrstea, editors, *Algebra and Coalgebra in Computer Science (CALCO 2011, Winchester, UK, August 30 - September 2, 2011)*, volume 6859 of *Lecture Notes in Computer Science*, pages 283–297. Springer, 2011.

[MMDB11b] Manuel A. Martins, Alexandre Madeira, Răzvan Diaconescu, and Luís Soares Barbosa. Hybridization of institutions (extended report). Technical Report TR-HASLab:05:2011, HASLab - INESC TEC, Universidade do Minho, October 2011.

[MML07] Till Mossakowski, Christian Maeder, and Klaus Lüttich. The heterogeneous tool set, Hets. In O. Grumberg and M. Huth, editors, *Tools and Algorithms for the Construction and Analysis of Systems (TACAS 2007 - Braga, Portugal, March 24 - April 1, 2007)*, volume 4424 of *Lecture Notes in Computer Science*, pages 519–522. Springer, 2007.

[MNMB13] Alexandre Madeira, Renato Neves, Manuel A. Martins, and Luís S. Barbosa. When even the interface evolves... In *7th Intern.l Symp. on Theoretical Aspects of Software Engineering (TASE 2013)*, volume (to appear). IEEE Press, 2013.

[Mos96] Till Mossakowski. Different types of arrow between logical frameworks. In F. Meyer auf der Heide and B. Monien, editors, *Automata, Languages and Programming (ICALP96, Paderborn, Germany, 8-12 July 1996)*, volume 1099 of *Lecture Notes in Computer Science*, pages 158–169. Springer, 1996.

[Mos02] Till Mossakowski. Relating CASL with other specification languages: the institution level. *Theor. Comput. Sci.*, 286(2):367–475, 2002.

[Mos03] Till Mossakowski. Foundations of heterogeneous specification. In M. Wirsing, D.Pattinson, and R. Hennicker, editors, *Recent Trends in Algebraic Development Techniques (Revised Selected Papers of WADT 2002, Frauenchiemsee, Germany, September 24-27, 2002)*, volume 2755 of *Lecture Notes in Computer Science*, pages 359–375. Springer, 2003.

[Mos04] Peter D. Mosses. *CASL Reference Manual, The Complete Documentation of the Common Algebraic Specification Language*, volume 2960 of *Lecture Notes in Computer Science*. Springer, 2004.

[MPW92] R. Milner, J. Parrow, and D. Walker. A calculus of mobile processes (parts I and II). *Information and Computation*, 100(1):1–77, 1992.

[MR06] Till Mossakowski and Markus Roggenbach. Structured CSP - a process algebra as an institution. In J. L. Fiadeiro and P.-Y. Schobbens, editors, *Recent Trends in Algebraic Development Techniques (Revised Selected Papers of WADT 2006, La Roche en Ardenne, Belgium, June 1-3, 2006)*, volume 4409 of *Lecture Notes in Computer Science*, pages 92–110. Springer, 2006.

[Nev13] Renato Neves. Proof support for hybridised logics. Master's thesis, Dep. Informatica, Universidade do Minho, 2013.

[NMMB13a] Renato Neves, Alexandre Madeira, Manuel A. Martins, and Luís S. Barbosa. From modelling to verification: Giving alloy a family, 2013. (submitted).

[NMMB13b] Renato Neves, Alexandre Madeira, Manuel A. Martins, and Luís S. Barbosa. Hybridisation at work. In *CALCO*

TOOLS, volume (to appear) of *Lecture Notes in Computer Science*. Springer, 2013.

[NW05] Hung T. Nguyen and Elbert A. Walker. *A first course in fuzzy logic (3. ed.)*. Chapman&Hall/CRC Press, 2005.

[Par81] David Park. Concurrency and automata on infinite sequences. In P. Deussen, editor, *Theoretical Computer Science (5th GI-Conference, Karlsruhe, Germany, March 23-25, 1981)*, volume 104 of *Lecture Notes in Computer Science*, pages 167–183. Springer, 1981.

[Pla10] Andre Platzer. *Logical Analysis of Hybrid Systems: Proving Theorems for Complex Dynamics*. Springer, 2010.

[Pri67] Arthur N. Prior. *Past, Present and Future*. Oxford University Press, 1967.

[PT91] Solomon Passy and Tinko Tinchev. An essay in combinatory dynamic logic. *Inf. Comput.*, 93(2):263–332, 1991.

[PW07] Björn Pelzer and Christoph Wernhard. System description: E-KRHyper. In F. Pfenning, editor, *Automated Deduction (CADE-21, Bremen, Germany, July 17-20, 2007)*, volume 4603 of *Lecture Notes in Computer Science*, pages 508–513. Springer, 2007.

[PZ01] Erik Poll and Jan Zwanenburg. From algebras and coalgebras to dialgebras. *Electr. Notes Theor. Comput. Sci.*, 44(1):289–307, 2001.

[Rei85] Horst Reichel. Behavioral program specification. In David H. Pitt, Samson Abramsky, Axel Poigné, and David E. Rydeheard, editors, *CTCS*, volume 240 of *Lecture Notes in Computer Science*, pages 390–411. Springer, 1985.

[RMMB11] César Jesus Rodrigues, Manuel A. Martins, Alexandre Madeira, and Luís Soares Barbosa. Refinement by interpretation in π-institutions. *EPTCS*, 55:53 – 64, 2011.

[Roz97] Gregori Rozenberg. *Handbook of Graph Grammars and Computing by Graph Transformation, Vol 1: Foundations*. World Scientific, 1997.

[RS11] Horia Ciocarlie Robert Szepesia. An overview on software reconfiguration. *Theory and Applications of Math. and Comp. Sci.*, 1:74–79, 2011.

[RSS12] João Rasga, Amílcar Sernadas, and Cristina Sernadas. Importing logics. *Studia Logica*, 100(3):545–581, 2012.

[Rut00] J. J. M. M. Rutten. Universal coalgebra: a theory of systems. *Theor. Comput. Sci.*, 249(1):3–80, 2000.

[RV02] Alexandre Riazanov and Andrei Voronkov. The design and implementation of VAMPIRE. *AI Commun.*, 15(2-3):91–110, 2002.

[San09] Davide Sangiorgi. On the origins of bisimulation and coinduction. *ACM Trans. Program. Lang. Syst.*, 31(4), 2009.

[Sch02] Setphan Schulz. E – a brainiac theorem prover. *Journal of AI Communications*, 15(2/3):111–126, 2002.

[Seg95] Roberto Segala. A compositional trace-based semantics for probabilistic automata. In I. Lee and S. A. Smolka, editors, *Concurrency Theory (CONCUR'95 - Philadelphia, PA, USA, August 21-24, 1995)*, volume 962 of *Lecture Notes in Computer Science*, pages 234–248. Springer, 1995.

[SM09] Lutz Schröder and Till Mossakowski. HasCasl: Integrated higher-order specification and program development. *Theor. Comput. Sci.*, 410(12-13):1217–1260, 2009.

[Sok11] Ana Sokolova. Probabilistic systems coalgebraically: A survey. *Theor. Comput. Sci.*, 412(38):5095–5110, 2011.

[ST08] Donald Sannella and Andrzej Tarlecki. Observability concepts in abstract data type specification, 30 years later. In P. Degano, R. De Nicola, and J. Meseguer, editors, *Concurrency, Graphs and Model (Essays Dedicated to Ugo Montanari on the Occasion of His 65th Birthday)*, volume 5065 of *Lecture Notes in Computer Science*, pages 593–617. Springer, 2008.

[ST10] Jeremy Sproston and Angelo Troina. Simulation and bisimulation for probabilistic timed automata. In K. Chatterjee and T. A. Henzinger, editors, *Formal Modeling and Analysis of Timed Systems (FORMATS 2010, Klosterneuburg, Austria, September 8-10, 2010)*, volume 6246 of *Lecture Notes in Computer Science*, pages 213–227. Springer, 2010.

[ST12] Donald Sannella and Andrzej Tarlecki. *Foundations of Algebraic Specification and Formal Software Development*. Monographs on Theoretical Computer Science, an EATCS Series. Springer, 2012.

[SV01] Ulrike Sattler and Moshe Y. Vardi. The hybrid μ-calculus. In R. Goré, A. Leitsch, and T. Nipkow, editors, *Automated Reasoning (IJCAR 2001 Siena, Italy, June 18-23, 2001)*, volume 2083 of *Lecture Notes in Computer Science*, pages 76–91. Springer, 2001.

[SW03] Davide Sangiorgi and David Walker. *The π-calculus: A Theory of Mobile Processes*. Cambridge University Press, 2003.

[Tar98] Andrzej Tarlecki. Towards heterogeneous specifications. In *Frontiers of Combining Systems (FroCoS'98)*, Applied Logic Series, pages 337–360. Kluwer Academic Publishers, 1998.

[Tar03] Andrzej Tarlecki. Abstract specification theory: an overview. In M. Broy and M. Pizka, editors, *Models, Algebras, and Logics of Engineering Software*, volume

191 of *NATO Science Series, Computer and Systems Sciences, VOL 191*, pages 43–79. IOS Press, 2003.

[tC05] Balder David ten Cate. *Model Theory for Extended Modal Languages*. PhD thesis, Institute for Logic, Language and Computation Universiteit van Amsterdam, 2005.

[TCCD10] Gabriel Tamura, Rubby Casallas, Anthony Cleve, and Laurence Duchien. Qos contract-aware reconfiguration of component architectures using e-graphs. In *Formal Aspects of Component Software (Revised Selected Papers of FACS 2010, Guimaraes, Portugal, October 14-16, 2010)*, volume 6921 of *Lecture Notes in Computer Science*, pages 34–52. Springer, 2010.

[tCF05] Balder ten Cate and Massimo Franceschet. On the complexity of hybrid logics with binders. In C.-H. Luke Ong, editor, *Computer Science Logic (CSL 2005 - Oxford, UK, August 22-25, 2005*, volume 3634 of *Lecture Notes in Computer Science*, pages 339–354. Springer, 2005.

[vB83] Johan van Betntham. *Modal Logic and Classic Logic*. Humanities Press, 1983.

[vE02] Jan van Eijck. Hylotab-tableau-based theorem proving for hybrid logics. Technical report, CWI, Amsterdam, 2002.

[Vou10] George Voutsadakis. Universal dialgebra: Unifying universal algebra and coalgebra. *Far East Journal of Mathematical Sciences*, 44(1), 2010.

[vRHWS08] M. Birna van Riemsdijk, Rolf Hennicker, Martin Wirsing, and Andreas Schroeder. Service specification and matchmaking using description logic. In J. Meseguer and G. Rosu, editors, *Algebraic Methodology and Software Technology (AMAST 2008 - Urbana, IL, USA, July*

28-31, 2008, volume 5140 of *Lecture Notes in Computer Science*, pages 392–406. Springer, 2008.

[WDF+09] Christoph Weidenbach, Dilyana Dimova, Arnaud Fietzke, Rohit Kumar, Martin Suda, and Patrick Wischnewski. SPASS version 3.5. In R. A. Schmidt, editor, *Automated Deduction (CADE-22, Montreal, Canada, August 2-7, 2009*, volume 5663 of *Lecture Notes in Computer Science*, pages 140–145. Springer, 2009.

[Web09] Volker Weber. On the complexity of branching-time logics. In E. Grädel and R. Kahle, editors, *Computer Science Logic (CSL 2009, Coimbra, Portugal, September 7-11, 2009)*, volume 5771 of *Lecture Notes in Computer Science*, pages 530–545. Springer, 2009.

[WF02] Michel Wermelinger and José Luiz Fiadeiro. A graph transformation approach to software architecture reconfiguration. *Sci. Comput. Program.*, 44(2):133–155, 2002.

[Wir90] M. Wirsing. Algebraic specification. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science (volume B)*, pages 673–788. Elsevier - MIT Press, 1990.

[YWM06] Zheng Yu, Ian Warren, and Bruce A. MacDonald. Dynamic reconfiguration for robot software. In *IEEE Intern. Conf. on Automation Science and Engineering (CASE, Shanghai, China, 7-10 October 2006)*, pages 292–297. IEEE, 2006.

[ZJJR11] Yi Zhang, Raoul Jetley, Paul L Jones, and Arnab Ray. Generic safety requirements for developing safe insulin pump software. *J Diabetes Sci Technol*, 5(6):1403–19, 2011.

[ZL12] Zhikun Zhao and Wei Li. Specifying dynamic software architectures with dynamic description logic. *Jour. of Software*, 7(1):169–175, 2012.