Universidade de Aveiro
Ano 2023

JULIANA ANDREIA MARQUES PRAZERES DA CUNHA

PARA UMA TEORIA DE ESPECIFICAÇÃO PARA TRANSIÇÕES PARACONSISTENTES

TOWARDS A SPECIFICATION THEORY FOR PARACONSISTENT TRANSITIONS

Universidade de Aveiro
Ano 2023

JULIANA ANDREIA MARQUES PRAZERES DA CUNHA

PARA UMA TEORIA DE ESPECIFICAÇÃO PARA TRANSIÇÕES PARACONSISTENTES

TOWARDS A SPECIFICATION THEORY FOR PARACONSISTENT TRANSITIONS

Dissertação apresentada à Universidade de Aveiro para cumprimento dos requisitos necessários à obtenção do grau de Mestre em Matemática e Aplicações, realizada sob a orientação científica do Doutor Alexandre Leite de Castro Madeira, Professor Auxiliar do Departamento de Matemática da Universidade de Aveiro, e coorientação do Doutor Luís Soares Barbosa, Professor Catedrático do Departamento de Informática da Universidade do Minho.

o júri

presidente

Prof. Doutor Eugénio Alexandre Miguel Rocha
Professor Associado da Universidade de Aveiro

vogais

Doutor Leandro Rafael Moreira Gomes
Investigador Pós-Doc, University of Lille

Prof. Doutor Alexandre Leite de Castro Madeira
Professor Auxiliar da Universidade de Aveiro (orientador)

agradecimentos / acknowledgements

Ao meu orientador professor Alexandre Madeira que me propôs este tema e sem saber o que me esperava percorri um caminho que me fez aprender imenso. Agradeço-lhe principalmente por todo o tempo que me dedicou, por toda a compreensão, por todo o rigor nos desafios que me apresentou e pelos seus constantes encorajamentos que me deram força e entusiasmo para continuar.

Ao meu coorientador Professor Luís Barbosa por todas as sugestões e comentários.

Aos meus pais por me terem sempre encorajado a continuar os meus estudos. Às minhas gatas Minnie e Cuca que foram as minhas companheiras durante 20 anos e que deixaram muitas saudades. Ao meu avô Fernando pelo seu incondicional apoio e por ter partilhado comigo tantas memórias, principalmente do meu pai, no que foram os nossos almoços ao domingo. À minha avó Estrela por toda a sua beleza, gentileza e por todos os abraços apertados que me deu.
Ao meu namorado Leandro por ser o meu maior apoio, a minha maior fonte de felicidade e inspiração mas acima de tudo agradeço-lhe por todo o seu amor.

Jamais existirão palavras que consigam expressar a imensa gratidão e carinho que sinto por todos. Muito obrigada!

palavras-chave Teoria das Categorias, Especificação, Sistemas de transição, Instituições, Paraconsistência, Lógica dinâmica, Lógica híbrida, Lógica modal.

resumo Existem muitas situações em que é necessário expressar e racionar sobre informação, num sistema, que ou se reforça ou se contradiz. Tal necessidade é uma das maiores motivações das *lógicas paraconsistentes*, que possibilitam a ocorrência de inconsistências sem o colapso de todo o sistema.

Esta dissertação apresenta duas lógicas, formalizadas como instituições, para racionar sob transições paraconsistentes de sistemas e os seus correspondentes processos. Nesse sentido procura desenvolver uma teoria de especificação, de uma maneira composicional e estruturada, para transições paraconsistentes.

keywords Category Theory, Specification, Transition Systems, Institutions, Paraconsistency, Dynamic logic, Hybrid logic, Modal logic.

abstract There are many situations where one needs to express and reason about information in a system that either reinforces or contradicts each other. Such need is one of the main motivations behind *paraconsistent logics*, that allow inconsistencies to occur without producing the collapse of the whole system.

This dissertation presents two logics, framed as institutions, to reason about paraconsistent transition systems and the corresponding processes. This leads to the development of a specification theory, approached in a compositional and structured manner, of paraconsistent transitions.

Contents

Contents **i**

List of Figures **1**

1 Introduction **3**

2 Background **11**
2.1 Category Theory 11
2.2 Institutions 19
2.3 Specification Theory 30

3 An institution for paraconsistent transition systems **35**
3.1 (Metric) Twisted algebras 35
3.2 The signatures 39
3.3 The models 40
3.4 The sentences 43
3.5 The satisfaction relation 45

4 An institution for paraconsistent processes **53**
4.1 The signatures 54
4.2 The models 54
4.3 The sentences 57
4.4 The satisfaction relation 59

5 Specification theory **63**
5.1 Structured Specification 63
5.2 Behavioural abstraction 73
5.2.1 Relating models 73
Similarity 73
Bisimilarity 77
5.2.2 Behavioural abstraction specifications 81

6 Conclusions and future work **89**

References **91**

List of Figures

1.1 Lattice $\mathfrak{FOUR}$ in [CMB22] 5
1.2 The vagueness-inconsistency square [CMB22]. 6

Chapter 1

Introduction

Principle of explosion and the triviality of logics
We say that a set of sentences in a logic is *trivial* [GW07] if it contains all sentences.
A classical logic, whose semantics is bivalent, demands that each formula is either true or false. In these logics if a set of sentences Φ is inconsistent i.e. let A be some formula, Φ contains both A and its negation $\neg A$ then, any formula can be derived from Φ. If any formula can be deducted from Φ then the set of consequences of Φ is trivial. This is called the *Principle of Explosion*, which states that from contradictory premises any conclusion can be derived.
Any classical logic satisfies the Principle of Explosion. Thus, classical logics become trivial in the presence of a contradiction and are said to be explosive. A problem arises when it is possible to consider different situations where inconsistent information exists. An even bigger problem arises when we want to infer conclusions from a set of inconsistent sentences, since classical logics can not be used for this purpose.

Some scenarios where inconsistent information may arise
One scenario where inconsistent information may be present is when data is collected trough a sensor. This is a very common practice in engineering to monitor systems. For instance, in drainage systems [WWN14] data regarding water levels in drainage, water discharge and rainfall conditions is collected from sensors to monitor the system. However, sensors have hardware limitations which limits the accuracy in measuring to a certain degree. Moreover, measurements themselves can be affected by outside conditions. It comes as no surprise that when using several sensors the information collected by one sensor may be contradictory to the information collected by another.
In medical practice it is also not uncommon that two physicians have contradictory diagnosis, both of them are relevant to be taken in consideration. In article [MGR15] a study was done to explore reasons for discrepancy between physicians and neuropsychologists diagnosis of dementia.
In Science one can also find contradictions. Perhaps the most recognized example is in Physics with the Bohr's model of atoms. This models considers electrons moving in orbitals around the nucleus of the atom. According to Maxwell's equations, any particle that is moving in a trajectory that is not a straight line experiences acceleration and consequently radiates energy. Yet, atoms do not radiate, at least not constantly, energy. We refer to [dCK03] for a further read into the topic.
Another scenario can be found in weather forecasting, which resorts to models for predicating

the weather. On average such models produce more accurate results whenever the difference between the current time and the time for which the forecast is being made decreases. Thus, whenever predictions contradict prior ones, forecasters need to make a decision to update or not this prediction to the public [BJ20]. Such changes can affect user's trust and given that users nowadays have multiple sources of weather forecast it would not be an hard decision to stop consulting a source that appears inconsistent. This is a great example to show that sometimes we have to reason among inconsistencies and the decisions we make lead to consequences.
Another interesting study to mention is [Bar32] where participants were asked to read the tale "War of the Ghosts". After that, at different periods their memory of the tale was tested. The study showed that longer intervals between hearing the tale and reproducing it led to recollections getting shorter as well as details being simplified to make the story more familiar. As details were more likely to be remembered if the story was interesting or familiar to the participant, it appears that often memory seems intertwine with inconsistency.
Another good example of this is witnesses testifying at a trial, each witness contributes with some information and even thought the information provided by each person is self-consistent, the group as a whole may have inconsistencies. That is, some witness may provide information that is contradictory with the information provided by another, for example disagreeing on the physical aspect of the criminal which may lead to wrongful convictions. It is not necessarily the case that witnesses are lying but that their testimony is a result of their perception [DBPM04]. Naturally, one wonders how can law enforcement, in the presence of contradictory information draw conclusions? How can it reason among inconsistencies?
This is not the only example of possible contradictory information that can be found in a courtroom as laws themselves may also contain contradictions. Suppose that a Parliament enacts two laws:

1. No child under the age of ten years is guilty of any offence.

2. Any person who has a knife with them on school premises is guilty of an offence.

If a child age 9 takes a knife to school, will he be found guilty of any offence? When such inconsistent laws are found usually either the jury rules one over the other or a change of more basic legislation may occur. We refer to [Wal99] for a more in depth reference regarding this topic.

Reasoning among inconsistencies: What is paraconsistent logic?
Paraconsistent logics can be valuable tools to deal with the scenarios described above, since instead of perceiving inconsistent information as an anomaly, they perceive it as potentially informative. It offers a flexible and controlled logical system to reason among contradictory information and draw reasonable conclusions. A formal definition of a *paraconsistent logic* can be found in [CM17],

> "A logic is paraconsistent iff it is not the case that for all formulas A, B, $\{A, \neg A\} \vdash B$"

Where $\vdash$ is an inference relation, thus we write $\{A\} \vdash B$ whenever a formula B can be derived from a formula A.
This definition of paraconsistent logic boils down to the denial of the Principle of Explosion. Consequently, paraconsistent logics are suitable to deal with inconsistent set of sentences,

without leading to the collapse of the whole system, i.e. without trivializing logic consequence. An influential contribution in paraconsistent logics was the work of the polish logician Jaśkowsi in 1948, who proposed a propositional paraconsistent logic that he called discursive logic. The main idea that motivated his creation was to imagine a group of people engaged in a discussion where everyone contributes with information. Does this sound familiar? Witnesses at a trial is probably an intuitive example (that we presented above) that describes the motivation behind Jaśkowsi logic.
Since Jaśkowsi work a lot more of research has been done on this topic. The work of Florencio Asenjo, Newton da Costa, Walter Carnielli and others granted paraconsistent logics to become a well established field of study. We refer to [GW07] for a fuller historical reference.

The paraconsistent logic used in this work
This dissertation's starting point is the paraconsistent logical system formalized in [CMB22], that generalises Belnap-Dunn four-valued logic [Bel77], whose semantics is given by the four element lattice $\mathfrak{FOUR}$:

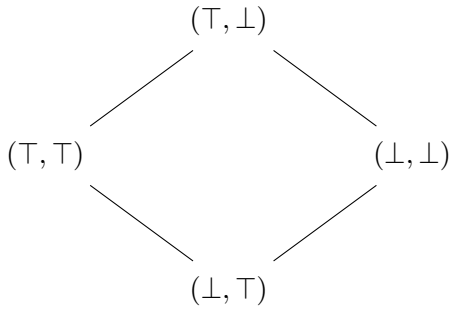


Figure 1.1: Lattice $\mathfrak{FOUR}$ in [CMB22]

In $\mathfrak{FOUR}$ propositions are interpreted as "true", "false", "neither true or false" or as "both true and false". Such results in the valuation of propositions being a pair (a, b) with $a, b \in \{\top, \bot\}$ such that pairs $(\top, \bot)$ and $(\bot, \top)$ represent consistent information, while pair $(\top, \top)$ represents inconsistent information and pair $(\bot, \bot)$ represents vague information. The relation between elements in Figure 1.1 is a truth ordering where $(\bot, \top) \leqslant (\top, \top) = (\bot, \bot) \leqslant (\top, \bot)$. However, other plausible orders of the elements can be admitted.

The logical system presented in [CMB22], extends $\mathfrak{FOUR}$ to a modal version, with two accessibility relations, one called *positive* and the other *negative*. Moreover, instead of interpreting weights in a billatice over $\{\top, \bot\}$, in [CMB22] the aim is to generalize the Belnap-Dunn four-valued logic to allow weights to be taken from a larger class of residuated lattices over a set A of possible truth values.
Therefore, valuation of propositions and semantics in the logical system used consist of a pair of weights over a set A, such that the positive accessibility relation weights the possibility of a transition being present while the negative (although not an actual negative value) weights the possibility of being absent. In classical bivalent logics such weights are complementary. However as stated before, in a number of real situations weights are not complementary and thus both relations should be formally taken into consideration.

Consider, without loss of generality, that weights for both transitions come from a resid-

uated lattice over the real $[0, 1]$ interval. Then, the two accessibility relations jointly express a scenario of:

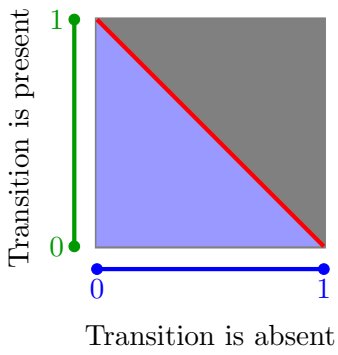


Figure 1.2: The vagueness-inconsistency square [CMB22].

- *inconsistency*, when the positive and negative weights are contradictory, that is, they sum to some value greater than 1 (corresponding to the upper triangle in Fig. 1.2 filled in grey);
- *vagueness*, when the sum is less than 1 (corresponding to the lower, periwinkle triangle in Fig. 1.2);
- *strict consistency*, when the sum is exactly 1, which means that the measures of the factors enforcing or preventing a transition are complementary, corresponding to the red line in the figure.

For the sake of giving a formal meaning to the square in Figure 1.2, the semantics of the logic system used is also endowed with a metric [CMB22] over the lattice carrier. This metric defines a distance between two weights and thus, it is able to assess the level of vagueness or inconsistency of a given pair. For example, a pair of weights that represents consistent information, i.e. in which the distance between the weights is 0, is expected to be in the red line of Figure 1.2.

Applications of paraconsistent logics

Recall the examples provided before where contradictory information may exist. In all examples a paraconsistent logic can be considered as a tool to model the corresponding scenario. The reference [TLAdSF11] is part of a series of research projects in which a robot is designed. The aim of their studies is to observe how a robot achieves a certain point in an environment with obstacles (for example, walls). This robot has a sensor on each side that covers in total a 90° degree angle. A paraconsistent logic comes into play here when one sensor detects an obstacle in front of the robot while the other does not. In this situations the robot can stop and turn 45° or it may take another decision like going ahead.

A paraconsistent hybrid logic has been studied in [CM14] to model the health care flow of a patient. Usually, when a patient enters an hospital he starts by being consulted at the triage station, followed by the observation by a physician. The next step might be medication which could have no effect and the patient may be observed again, which can result in the realization

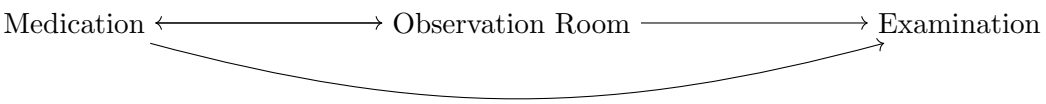
of exams that can be inconclusive and thus a new observation is required and the pathway continues until the patient is either cured or dead. This pathway accounts for inconsistencies at different stages.
One can not forget to mention the application aim of the logical system we will use in this work, proposed in [CMB22]. The goal was to model quantum circuits where a phenomena called *decoherence*, that can be seen as a loss of information from the circuit through interaction with the environment, can happen. Decoherence is not an exact measure, usually there exist time intervals where the circuit is said to lose coherence. Thus, a transition in a model consists of two accessibility relations, one describing the possibility of the system remaining coherent and the other the possibility of it becoming unstable.
Finally, Newton da Costa and Carnielli article [CC86] proposes a paraconsistent deontic logic. Deontic logics are often used to model laws since they can express concepts as obligation, prohibition, right and wrong. Other paraconsistent deontic logics have been proposed, for example in [Con07]. It is important to note that neither of these logics can remove inconsistency in a legal repository but they are helpful tools to identify contradictory laws leading to the decision of either removing some of them or modifying them to avoid conflicts.

Beyond these examples, a lot more can be referred where paraconsistent logics are used to model and reason. For instance, monitoring devices of electrical systems [CSdSF22], valuation of ecosystem services [Afa19] and artificial intelligence [dSFLA10].

Specification theory
As seen so far, the applications of paraconsistent logics are vast. However, we have not pointed out the fact that any attempt in modelling a problem must start with a description of the task that the system is supposed to perform. For example, in [CM14], where a health care flow of a patient is modelled there exists a set Δ of sentences in the underlying logic used. This set represents properties that must be satisfied in every model. To illustrated the concept, consider Δ to be the set

$$\{@_{Obs.room}(\diamond Medication \vee \diamond Examination), @_{Medication}(\diamond Examination \vee \diamond Obs.room)\} \tag{1.1}$$

The first sentence says that we expect patients at the observation room may need to take medication; or to carry out an examination. The second sentence means that if a patient is taking medication he might need to go back to the observation room; or undertake an examination for some reason, e.g. because there may be a need to reevaluate his dosage of medication.
The term *specification* [ST12] is thus used to define a formal object that characterizes a particular system. In the previous example, Δ is a specification that characterizes the class of all possible pathway cares.
Upon the decision of the specification the goal is to build a concrete model/system such that it satisfies the tasks described in the specification. This system will be seen as a correct realisation of the problem we pretend to model. For instance, the following Kripke frame is a system that models health care flow of patients for Δ equal to (1.1).



The usual method to verify if a model satisfies a specification involves checking if all the tasks in the specification are satisfied through an exhaustive, but finite, testing of the model.

Such approach has the advantage of certainty that the program performs all the demanded tasks, this is called *correctness* [ST12]. However, for problems of realistic size, full proofs of correctness are usually unfeasible. In addition, specifications can be too abstract, complex or large to be perfectly formalized.
Consequently, another approach is taken to build correct specifications through refinements. As described in [ST12], "Each refinement step captures a single design decision, for instance a choice between several functions that satisfy the specification, between several algorithms that implement the same function, or between several ways of efficiently representing a given data". Hence, when building a large/complex specification it is possible to decompose the problem into smaller/simpler specifications that are possible to define. Later these smaller units can be refined more and even combined to achieve the complex specification we desire until its correspondence to a concrete model becomes obvious.

The build up: our work
The challenge of our work is to connect paraconsistent logic and specification theory. Thus, we start by using a paraconsistent logic that is capable of modelling problems where inconsistent information occurs, for instance in quantum circuits [CMB22]. However, our goal is not to explore these applications. Actually, all relevant constructions of the logical system used are parametric in a class of residuated lattices, thus admitting different instances, according to the structure of the truth values domain, to better suit each modelling problem at hand.
Thus, later on, we define a *specification theory*, associated with the logic used. Technically, the price to be paid to support these developments consists of framing the logic used as an (many-valued) institution. Specifications can be viewed as correct realisations of some problem we pretend to model and as stated, they may not have a straightforward construction.

Contributions

- In this dissertation we extend the modal logic for paraconsistent transition systems presented in [CMB22] to the multi-modal case. We also introduce another logic for paraconsistent transition processes, that aside from being modal also embodies dynamic and hybrid logic features. Then, both logics are framed in the categorical framework of (many-valued) institutions [ACEGG91].

- Our main contribution lies on setting up a specification theory for paraconsistent transitions, for which two parts were explored: structured specifications and abstraction specifications.

  - We start by adjusting the semantics of standard structured specification operators *à la CASL* [MHST03] to perform structured specification of paraconsistent transition systems. These results were reported in our paper [CMBnt]. Moreover, we also define similar structured specification operators for paraconsistent transition processes. This approach builds on previous work documented in [JGMB21] where a similar agenda is proposed for the specification of fuzzy transition systems.
  - We also present a palette of constructors defined to combine models over different signatures into a model of another signature. These constructors are based on work documented in [MBHM18] where a dynamic and hybrid logic, $D^{\downarrow}$, is proposed to reason about reactive systems.

- Finally, we introduce abstraction specifications, following the approach introduced in [JGMB21, MBHM18].

Outline
The remaining of the thesis is organized as follows:

Chapter 2 provides the background needed for this dissertation. Recalling some necessary definitions as well as setting the notation used in *Category Theory*, *Theory of Institutions* and *Structured Specification*.

Chapter 3 starts with Section 3.1 where an introduction to *(Metric) Twisted algebras* is given, the definition of a residuated lattice and the definition of a twisted-structure, denoted by $\mathscr{A}$, are also recalled. Finally the section ends with a list of properties, regarding *iMTL*-algebras and twisted algebras, and their respective proofs. For the following sections in this chapter a twisted algebra $\mathscr{A}$ is fixed and the focus is on defining all the necessary ingredients to formalize a logic for paraconsistent transitition systems, $\boldsymbol{S}(\mathscr{A})$, as a many-valued institution. All of the ingredients are framed in terms of category theory (either as a category or as a functor) and the satisfaction condition (for a many-valued institution) is proved. Throughout the chapter some important results are explored and examples are provided to illustrate concepts.

Chapter 4 introduces another logic, $\boldsymbol{P}(\mathscr{A})$, for paraconsistent processes also formalized as a many-valued institution. We revisit all the definitions and results presented in the previous chapter but adapting them to suit the new logic. The main differences between $\boldsymbol{S}(\mathscr{A})$ and $\boldsymbol{P}(\mathscr{A})$ are: combining modalities with regular expressions of actions, typical of dynamic logic, the introduction of binders $\downarrow x$ and $@_x$ operator, typical of hybrid logic [1], and finally the satisfaction relation is now evaluated from an initial state rather than globally.

Chapter 5 divided in two parts, is devoted to specification theory for paraconsistent transitions.
In the first part, Section 5.1, $\boldsymbol{S}(\mathscr{A})$ and $\boldsymbol{P}(\mathscr{A})$ are equipped with standard structured specification operators *à la CASL* [MHST03] that are re-built on top of these institutions. The first concept to be introduced is the one of *flat specification* a basic tool to create simple and small specifications. Then structured specifications operators are defined to build (more complex) specifications. Examples of these constructions are provided to illustrate the different operators. In addition, the definition of *simple implementation* is given alongside the results of *vertical* and *horizontal composition*. The rest of the first part follows the lines of the work documented in [MBHM18]. We address some of the constructors as well as some of the examples presented in this reference, slightly modifying them to suit our logic for processes $\boldsymbol{P}(\mathscr{A})$.
On the second part of this chapter, Section 5.2, the notion of similarity and bisimilarity are defined for $\boldsymbol{S}(\mathscr{A})$ and $\boldsymbol{P}(\mathscr{A})$. Results with respect to these relations are presented in Theorem 10 and Theorem 11 showing the preservations of the satisfaction relations between two related models through similarity and bisimilarity, respectively. Notions of *abstract implementation* and *abstractor implementation* ***w.r.t.*** *an equivalence relation* $\equiv$ are given as well as a result for vertical composition.

[1] It is worth noting that nominals are not added, only state variables are used as parameters

Chapter 6 presents the main conclusions that emerged from this dissertation and some ideas for future research.

Chapter 2

Background

The purpose of this chapter is to give a brief introduction to *Category Theory* in Section 2.1, *Institutions* in Section 2.2 and finally *Specification Theory* in Section 2.3. We focus on presenting main concepts and results that are required for the developments in later chapters.

It is worth noting that some of the results or examples throughout this chapter correspond to worked out exercises proposed in some references. When this is the case the reference will be properly mentioned and we provide a plausible solution for that exercise.

2.1 Category Theory

In 1945 Samuel Eilenberg and Saunders Mac Lane paper "General Theory of Natural Equivalence" [EM45] introduced the concept of category, functor and natural transformations, initially to simplify aspects of algebraic topology.
In [Awo10], Steve Awodey describes Category Theory as:

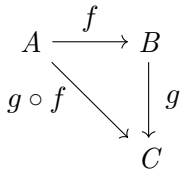
> "It's the abstract theory of functions. Functions are everywhere! And everywhere that functions are, there are categories"

Thus, Category Theory rouse in popularity due to its wide range of applications and due to being a powerful, and at the same time, simple framework to reason about objects and the relations between them. Fundamentally, it does not matter what the objects that we study are and their particular nuances, what matters is how they relate. This mindset provides us with generalization and abstraction, typical of Category Theory. Hence, each collection of objects comes equipped with a notion of morphism that fulfills two requirements: it has identities and it is associative.
The definitions, examples and results on this section can be found in [Awo10] and [ST12, Chapter 3].

Definition 1. *A category* K *consists of:*

- *a collection* $|\mathsf{K}|$ *of* K*-objects*
- *for* $A, B \in |\mathsf{K}|$*, a collection* $\mathsf{K}(A, B)$ *of* K*-morphisms from* A *to* B*.*
- *for* $f \in \mathsf{K}(A, B)$ *we write* $f : A \to B$ *to indicate that* $A = dom(f)$ *(domain of* f*) and* $B = cod(f)$ *(codomain of* f*)*

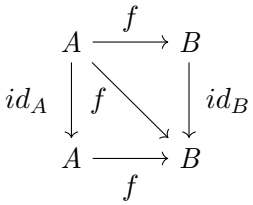
- *for $A, B, C \in |\mathsf{K}|$, a composition operator $_ \circ _ : \mathsf{K}(A,B) \times \mathsf{K}(B,C) \to \mathsf{K}(A,C)$.*



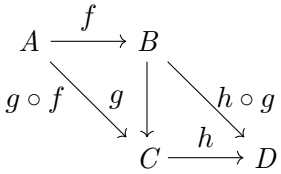
- *for each $A \in |\mathsf{K}|$, a morphism $id_A \in \mathsf{K}(A,A)$ called the identity morphism of A*

Required to satisfy:

1. Existence of identities: *for any $f \in \mathsf{K}(A,B)$, $f \circ id_A = f = id_B \circ f$, i.e. the following diagram commutes*



2. Associative of composition: *for any $f \in \mathsf{K}(A,B)$, $g \in \mathsf{K}(B,C)$ and $h \in \mathsf{K}(C,D)$, $h \circ (g \circ f) = (h \circ g) \circ f$, i.e the following diagram commutes:*



Examples of categories are presented next to illustrate the concept.

Example 1. *[Awo10]* [***Monoid Category***]
The category has exactly one object X and morphisms are elements of X. The set X equipped with a binary operation $_\cdot_ : X \times X \to X$ and an identity element $id \in X$ is a monoid $\langle X, \cdot, id \rangle$ such that $(x \cdot y) \cdot z = x \cdot (y \cdot z)$ and $id \cdot x = x = x \cdot id$ for all $x, y, z \in X$.

Example 2. *[Awo10]* [Set, ***category of sets***]
Let the objects in the category be sets and the morphisms be functions with explicit domain and codomain. Given sets A, B, C and functions $f : A \to B$, $g : B \to C$ we have that $g \circ f : A \to C$ such that for any $a \in A$, $(g \circ f)(a) = g(f(a))$.
By the definition of the composition operator and the identity function we have that:

$$(f \circ 1_A)(a) = f(1_A(a)) = f(a) = 1_B(\underbrace{f(a)}_{\in B}) = (1_B \circ f)(a)$$

Proving the existence of identities.
Using again the definition of the composition operator we have:

$$(h \circ (g \circ f))(a) = h((g \circ f)(a)) = h(g(f(a)))$$

$$((h \circ g) \circ f)(a) = (h \circ g)(f(a)) = h(g(f(a)))$$

Therefore, $(h \circ (g \circ f))(a) = ((h \circ g) \circ f)(a)$*, proving the associativity of the composition operator.*

Example 3. *[Awo10, Chapter 1, Exercise 1]* [Rel ***category***]
Let sets be objects and binary relations morphisms, that is, $f : A \to B$ *is a subset* $f \subseteq A \times B$*. The identity relation is:*

$$id_A = \{(a, a) \in A \times A \,|\, a \in A\} \subseteq A \times A$$

Given $g \subseteq B \times C$ *the composition operator* $f \cdot g$*, is defined by:*

$$(a, c) \in f \cdot g \Leftrightarrow \exists b,\ (a, b) \in f \,\&\, (b, c) \in g$$

Analogously, for any $(a, c) \in A \times B$

$$(a, c) \in id_A \cdot f \Leftrightarrow \exists b\ (a, b) \in id_A \,\&\, (b, c) \in f$$

By the definition of id_A*, if* $(a, b) \in id_A$ *then* $a = b$ *therefore* $(a, c) \in f$*.*
Similarly,

$$(a, c) \in f \cdot id_B \Leftrightarrow \exists b\ (a, b) \in f \,\&\, (b, c) \in id_B$$

By the definition of id_B *we have that* $b = c$ *so* $(a, c) \in f$*. Thus, proving* $id_A \cdot f = f = f \cdot id_B$*. Consider now a relation* $h \subseteq C \times D$*. To prove the associativity of the composition operator we use the definition of "*$\cdot$*":*

$$\begin{aligned}
(a, d) \in (f \cdot g) \cdot h &\Leftrightarrow \exists c,\ (a, c) \in (f \cdot g) \,\&\, (c, d) \in h \\
&\Leftrightarrow \exists c\, (\exists b,\ (a, b) \in f \,\&\, (b, c) \in g) \,\&\, (c, d) \in h \\
&\Leftrightarrow \exists b, c,\ (a, b) \in f \,\&\, (b, c) \in g \,\&\, (c, d) \in h \\
&\Leftrightarrow \exists b,\ (a, b) \in f \,\&\, (\exists c, (b, c) \in g \,\&\, (c, d) \in h) \\
&\Leftrightarrow \exists b,\ (a, b) \in f \,\&\, (b, d) \in (g \cdot h) \\
&\Leftrightarrow (a, d) \in f \cdot (g \cdot h)
\end{aligned}$$

The following two examples of categories can be found in [ST12] as Example 3.1.12 (category of algebraic signatures, Sign) and Example 3.1.9 (category of Σ-algebras, Alg(Σ)) with the objects and morphisms of the categories explicitly stated. We detailed these examples by proving that Sign and Alg(Σ) are categories.

Example 4. [Sign, ***category of signatures***]
A signature *is a pair* $\Sigma = \langle S, \Omega \rangle$*, where* S *is a set of sort names and* Ω *is a* $(S^* \times S)$*-sorted set of operation names with* S^* *being a finite sequence of elements of* S*. We write* $f : s_1, ..., s_n \to s \in \Sigma$ *to denote* $f \in \Omega_{s_1,..,s_n,s}$*, i.e. that* f *is the name of an* n*-ary operation that takes arguments of sort* $s_1, .., s_n$ *and produces a result of sort* s*.*

Let $\Sigma' = \langle S', \Omega' \rangle$ *be another signature. A* signature morphism $\sigma : \Sigma \to \Sigma'$ *between* Σ *and* Σ' *is a pair* $\sigma = \langle \sigma_{sorts}, \sigma_{ops} \rangle$ *such that:*

- $\sigma_{sorts} : S \to S'$

- σ_{ops} *is a family of functions that respects the arities of operation names, i.e. it is a family:*

$$\sigma_{ops} = \langle \sigma_{w,s} : \Omega_{w,s} \to \Omega'_{\sigma^*_{sorts}(w),\sigma_{sorts}(s)} \rangle_{w \in S^*,\, s \in S}$$

Such that for any $w = s_1...s_n \in S^*$, $\sigma^*_{sorts}(w) = \sigma_{sorts}(s_1)...\sigma_{sorts}(s_n)$

The composition of signature morphisms σ *and* $\sigma' : \Sigma' \to \Sigma''$ *is defined as* $(\sigma' \circ \sigma) = \sigma'(\sigma)$ *or more precisely* $(\sigma' \circ \sigma)_{sorts} = \sigma'_{sorts}(\sigma_{sorts})$ *and* $(\sigma' \circ \sigma)_{w,s} = \sigma'_{\sigma * _{sorts}(w),\sigma_{sorts}(s)}(\sigma_{w,s})$.

If we consider objects to be signatures and morphisms to be signature morphisms, Sign *is a category. We have that:*

$$(\sigma \circ id_\Sigma) = \sigma = (id_{\Sigma'} \circ \sigma)$$

Also, for any signature morphism $\sigma'' : \Sigma'' \to \Sigma'''$ *and for any sort or operation* x*, using the definition of composition of morphisms:*

$$(\sigma'' \circ (\sigma' \circ \sigma))(x) = (\sigma''(\sigma' \circ \sigma(x))) = \sigma''(\sigma'(\sigma(x))) = (\sigma'' \circ \sigma')(\sigma(x)) = ((\sigma'' \circ \sigma') \circ \sigma(x))$$

Example 5. [Alg(Σ), ***category of*** Σ***-algebras***]
A Σ-algebra A *consists of:*

- *an* S*-sorted set* $|A|$ *of carriers*

- *for each* $f : s_1 \times ... \times s_n \to s \in \Sigma$*, a operation* $(f : s_1 \times ... \times s_n \to s)_A : |A|_{s_1} \times ... \times |A|_{s_n} \to |A|_s$

If s *is a sort in* Σ *then* $|A|_s$ *is the carrier set of sort* s *in* A*, that is, the universe of possible data values of sort* s*. If* $f : s_1 \times ... \times s_n \to s$ *in* Σ *then the operation* $(f : s_1 \times ... \times s_n \to s)_A$ *is a function on the corresponding carrier sets of* A*, that will be abbreviated to* f_A.

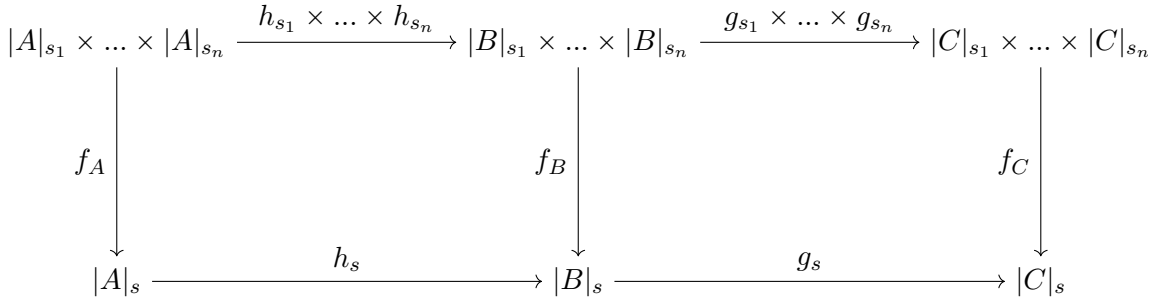
Given any Σ*-algebras* A *and* B*. A* Σ*-homomorphism* $h : A \to B$ *is an* S*-sorted function* $h : |A| \to |B|$ *that respects operations in* Σ*, that is, for any* $f : s_1 \times ... \times s_n \to s \in \Sigma$ *and for* $a_1 \in |A|_{s_1}, .., a_n \in |A|_{s_n}$*, we have that* $h_s(f_A(a_1, ..., a_n)) = f_B(h_{s_1}, ..., h_{s_n})$.

Let the objects of the category be Σ*-algebras and the morphisms* Σ*-homomorphisms with explicit given domain and codomain algebras.*

A homomorphism between algebras is a function between carriers that respect the behaviour of the operators. We have already proven in Example 2 that Set *is a category, we only have left to prove that the identity homomorphism and the composition of homorphisms preserve the behaviour of the operators. For any operation* $f : s_1 \times ... \times s_n \to s \in \Sigma$ *and* $a_1 \in |A|_{s_1} \times ... \times |A|_{s_n}$*,*

$$id_S(f_A(a_1, ..., a_n)) = f_A(a_1, ..., a_n) = f_A(id_{s_1}(a_1), ..., id_{s_n}(a_n))$$

Given Σ*-homomorphisms* $h : A \to B$ *and* $g : B \to C$ *we are left to show that the following diagram commutes:*



$$\begin{aligned} g_s(h_s(f_A(a_1,...,a_n)))) &= \{\textit{def. of } \Sigma\textit{-homomorphism}\} \\ & \quad g_s(f_B(h_{s_1}(a_1),...,h_{s_n}(a_n))) \\ &= \{\textit{def. of } \Sigma\textit{-homomorphism}\} \\ & \quad f_C(g_{s_1}(h_{s_1}(a_1)),...,g_{s_1}(h_{s_1}(a_n))) \end{aligned}$$

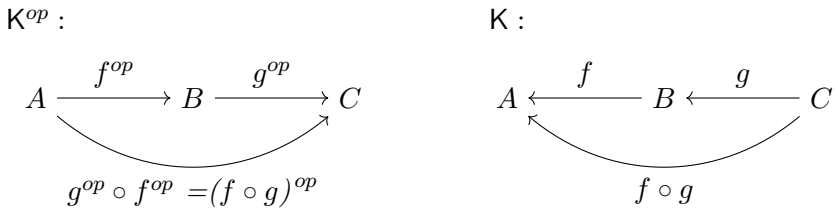
Thus, we conclude that $\mathsf{Alg}(\Sigma)$ *is a category.*

The dual concept of a category is an *opposite category* where objects are the same but morphisms are "reversed".

Definition 2. *The* opposite category *of a category* K *is the category* K^{op} *where:*

- $|\mathsf{K}^{op}| = |\mathsf{K}|$*, objects in* K^{op} *are the same as in* K
- $\mathsf{K}^{op}(A,B) = \mathsf{K}(B,A)$ *for all* $A, B \in |\mathsf{K}^{op}|$
- *for* $f \in \mathsf{K}^{op}(A,B)$ *(i.e.* $f \in \mathsf{K}(B,A)$*) and* $g \in \mathsf{K}^{op}(B,C)$ *(i.e.* $g \in \mathsf{K}(C,B)$*),* $f \circ g \in \mathsf{K}(C,A)$

It is useful to have a different notation to distinguish a morphism in K *from the same one in* K^{op}*. Thus, for any* $f \in \mathsf{K}(B,A)$ *we write* $f^{op} \in \mathsf{K}^{op}(A,B)$*. The following diagram illustrates composition:*



Lemma 1. *[ST12][Exercise 3.1.23] For a category* K*,* K^{op} *is a category.*

Proof. Let K be a category then for any $A, B \in |\mathsf{K}|$ and morphisms $f \in K(B,A)$, $g \in K(C,B)$ and $h \in K(D,C)$ we have:

1. The existence of identities $f \circ id_B = f = id_A \circ f$
2. Composition is associative: $(f \circ g) \circ h = f \circ (g \circ h)$

Using the definition of K^{op} the proprieties above translate to:

1. $(f \circ id_B)^{op} = f^{op} = (id_A \circ f)^{op}$ which is the same as writing $id_B^{op} \circ f^{op} = f^{op} = f^{op} \circ id_A^{op}$. Proving the existence of identities in K^{op}

2. $((f \circ g) \circ h)^{op} = h^{op} \circ (f \circ g)^{op} = h^{op} \circ g^{op} \circ f^{op} = (g \circ h)^{op} \circ f^{op} = (f \circ (g \circ h))^{op}$. Proving the associativity of the composition operator in K^{op}.

□

The following corollary is a consequence of Lemma 1 plus the fact that Sign is a category, Example 4.

Corollary 1. Sign^{op}*, the opposite category of algebraic signatures, is a category.*

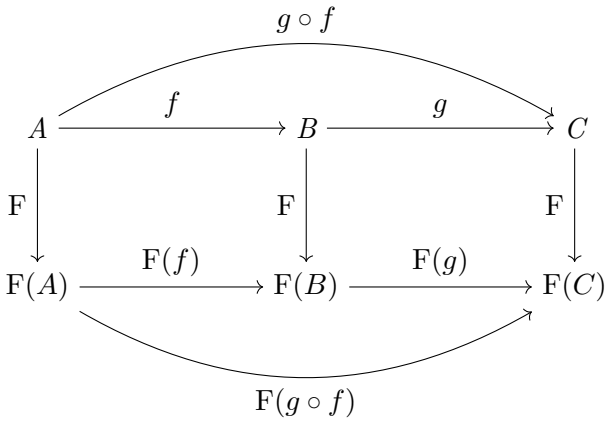
One of the simplest ways of constructing a new category is by selecting objects from another category, K, and the morphisms between those objects, consequently constructing a *subcategory* of K.

Definition 3. *A category* K' *is a subcategory of* K*, in symbols* $\mathsf{K}' \subseteq \mathsf{K}$*, if* $|\mathsf{K}'| \subseteq |\mathsf{K}|$ *and* $\mathsf{K}'(A, B) \subseteq \mathsf{K}(A, B)$ *for all* $A, B \in |\mathsf{K}'|$*, with composition and identities in* K' *as in* K*.*

Another important notion in Category Theory is that of a *functor*. A functor consists of a mapping between two categories such that identities and composition are preserved.

Definition 4. *A functor* $\mathrm{F} : \mathsf{K}_1 \to \mathsf{K}_2$ *between categories* K_1 *and* K_2 *is a map which:*

- *Associates objects of* K_1 *to objects of* K_2*:* $|\mathrm{F}| : |\mathsf{K}_1| \to |\mathsf{K}_2|$
- *for each* $A, B \in |\mathsf{K}_1|$*, a function* $\mathrm{F}_{A,B} : \mathsf{K}_1(A, B) \to \mathsf{K}_2(\mathrm{F}(A), \mathrm{F}(B))$ *such that:*
  1. F *preserves identities:* $\mathrm{F}(id_A) = id_{\mathrm{F}(A)}$ *for all objects* $A \in |\mathsf{K}_1|$
  2. F *preserves composition: for all morphisms* $f : A \to B$ *and* $g : B \to C$ *in* K_1*,* $\mathrm{F}(g \circ f) = \mathrm{F}(g) \circ \mathrm{F}(f)$



Another example of a category is the category of all categories, CAT, defined as:

Definition 5. *The objects in* CAT *are categories and the morphisms between objects are functors such that if* $\mathrm{F} : \mathsf{K}_1 \to \mathsf{K}_2$ *and* $\mathrm{G} : \mathsf{K}_2 \to \mathsf{K}_3$ *are functors, then* $\mathrm{G} \circ \mathrm{F} : \mathsf{K}_1 \to \mathsf{K}_3$ *is a functor defined as:*

- $|G \circ F| = |G| \circ |F|$
- $(G \circ F)(A, B) = G(F(A), F(B)) \circ F(A, B)$ *for all* $A, B \in \mathsf{K}_1$

Let us consider a few examples of functors.

Example 6. *[Awo10]* [***Monoid functor***]
Let $\mathsf{M} = \langle X, \cdot_M, id_M \rangle$ *and* $\mathsf{N} = \langle Y, \cdot_N, id_N \rangle$ *be two monoid categories, the mapping* $\mathrm{F} : \mathsf{M} \to \mathsf{N}$ *is a functor such that for all* $m, n \in X$*:*

- $\mathrm{F}(id_M) = id_{\mathrm{F}(X)} = id_N$
- $\mathrm{F}(m \cdot_M n) = \mathrm{F}(m) \cdot_N \mathrm{F}(n)$

The following example of a functor (σ-reduct of an algebra, $Mod(\sigma)$) can be found in [ST12] as Example 3.4.29 with the components of the functor specified in the reference. We provide in the example a formal proof that $\mathsf{Mod}(\sigma)$ is indeed a functor.

Example 7. [***Reduct algebra functor***]
Let A' *be a* Σ'*-algebra. Given a signature morphism* $\sigma : \Sigma \to \Sigma'$*, the* σ*-reduct algebra of* A' *is a* Σ*-algebra* $\mathsf{Mod}(\sigma)(A')$ *defined as:*

- $|\mathsf{Mod}(\sigma)(A')|_s = |A'|_{\sigma(s)}$ *for all* $s \in S$
- *for all* $f : s_1 \times ... \times s_n \to s$ *in* Σ*,*

$$f_{A'|_\sigma} : |\mathsf{Mod}(\sigma)(A')|_{s_1} \times ... \times |\mathsf{Mod}(\sigma)(A')|_{s_n} \to |\mathsf{Mod}(\sigma)(A')|_s = \sigma(f)_{A'} : |A'|_{\sigma(s_1)} \times ... \times |A'|_{\sigma(s_n)} \to |A'|_{\sigma(s)}$$

Given any Σ'*-homomorphism* $h' : A' \to B'$ *the* σ*-reduct homomorphism,*

$$\mathsf{Mod}(\sigma)(h') : \mathsf{Mod}(\sigma)(A') \to \mathsf{Mod}(\sigma)(B')$$

is an S*-sorted function* $\mathsf{Mod}(\sigma)(h') : |\mathsf{Mod}(\sigma)(A')| \to |\mathsf{Mod}(\sigma)(B')|$ *such that* $(\mathsf{Mod}(\sigma)(h'))_s = h'_{\sigma(s)}$ *for all* $s \in S$*.*

The mapping $\mathsf{Mod}(\sigma) : \mathsf{Alg}(\Sigma') \to \mathsf{Alg}(\Sigma)$ *is a functor that takes each* Σ'*-algebra* A' *to its reduct* $\mathsf{Mod}(\sigma)(A')$ *(in* $\mathsf{Alg}(\Sigma)$*) and each* Σ'*-homomorphism* h' *to its* σ*-reduct* $\mathsf{Mod}(\sigma)(h')$*.*

$$\begin{array}{ccc} \mathsf{Mod}(\sigma)(A') & \xleftarrow{\mathsf{Mod}(\sigma)} & A' \\ \Big\downarrow{\scriptstyle \mathsf{Mod}(\sigma)(h')} & & \Big\downarrow{\scriptstyle h} \\ \mathsf{Mod}(\sigma)(B') & \xleftarrow{\mathsf{Mod}(\sigma)} & B' \end{array}$$

The identity homomorphism $id_{A'}$ *is mapped to its reduct* $id_{\mathsf{Mod}(\sigma)(A')}$*, that is ,* $\mathsf{Mod}(\sigma)(id_{A'}) = id_{\mathsf{Mod}(\sigma)(A')}$*.*

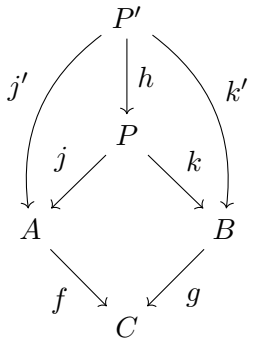
For any $f : A' \to B'$ and $g : B' \to C'$, Σ'-homomorphisms defined as $(g \circ f)_s = g_s \circ f_s$ for any $s \in S$

$$\begin{aligned}
\mathsf{Mod}(\sigma)(g \circ f)_s =& \{\textit{def. of } \sigma\textit{-reduct homomorphism}\} \\
& (g \circ f)_{\sigma(s)} \\
=& \{\textit{composition of homomorphism}\} \\
& g_{\sigma(s)} \circ f_{\sigma(s)} \\
=& \{\textit{def. of } \Sigma\textit{-reduct homomorphism}\} \\
& (\mathsf{Mod}(\sigma)(g))_s \circ (\mathsf{Mod}(\sigma)(f))_s
\end{aligned}$$

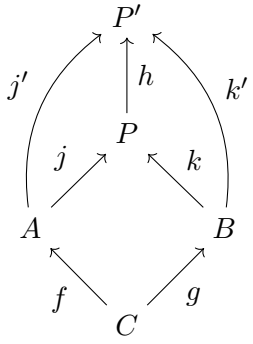
So, $Mod : \mathsf{Sign}^{op} \to \mathsf{CAT}$ is a functor between the opposite category of algebraic signatures and the category of all categories.

Finally we present the dual concepts of *pullback* and *pushout*, where K stands for an arbitrary category.

Definition 6. *[ST12] A pullback of two morphisms with the same codomain, $f : A \to C$ and $g : B \to C$, is an object $P \in |\mathsf{K}|$ and a pair of morphisms $j : P \to A$ and $k : P \to B$ such that $f \circ j = g \circ k$, and such that for any object $P' \in |\mathsf{K}|$ and pair of morphisms $j' : P' \to A$ and $k' : P' \to B$ satisfying $f \circ j' = g \circ k'$ there is exactly one morphism $h : P' \to P$ such that the following diagram commutes:*



Definition 7. *A pushout of two morphisms with the same domain, $f : C \to A$ and $g : C \to B$, is an object $P \in |\mathsf{K}|$ together with a pair of morphisms $j : A \to P$ and $k : B \to P$ such that $j \circ f = k \circ g$, and such that for any object $P' \in |\mathsf{K}|$ and pair of morphisms $j' : A \to P'$ and $k' : B \to P'$ satisfying $j' \circ f = k' \circ g$ there is exactly one morphism $h : P \to P'$ such that the following diagram commutes:*



Pushouts are useful for combining structures, in particular algebraic signatures. Given two signatures and two signature morphisms with the same domain, the pushout puts together the signatures while keeping the ability to identify the parts coming from the common source and the new parts disjoint.

2.2 Institutions

In [BG80] Goguen and Burstall paper entitled "The semantics of clear, a specification language" presented for the first time the concept of an *institution*, initially called a language. For a more in depth historical reference on institutions we refer to [Dia12].

Informally, an institution defines abstractly a logic system by describing the kinds of signatures, the models and the satisfaction relation between models and sentences. In fact, the concept is formulated in terms of category theory affording us with generalization and abstraction. All entities (signatures, sentences and models) are not only required to be specified but their morphisms as well, so we can abstract from the exact details of the objects and focus on their relationships.

This section first introduces the definition of institution and some of its important properties. We illustrate the concept with the institutions of Propositional, Equational, First order and Modal logics. These examples are just a small collection of many logics that have been formalized as institutions after Goguen and Burstall paper.

Definition 8. *[GB92] An institution I is a tuple*

$$I = (\mathsf{Sign}_I, \mathsf{Sen}_I, \mathsf{Mod}_I, \models_I)$$

consisting of

- *a category Sign_I, whose objects are called signatures*
- *a functor $\mathsf{Sen}_I : \mathsf{Sign}_I \to \mathsf{Set}$ giving a set of Σ-sentences for each signature $\Sigma \in |\mathsf{Sign}_I|$. For each signature morphism $\sigma : \Sigma \to \Sigma'$ the function*

  $$\mathsf{Sen}_I(\sigma) : \mathsf{Sen}_I(\Sigma) \to \mathsf{Sen}_I(\Sigma')$$

  translates Σ-sentences to Σ'-sentences
- *a functor $\mathsf{Mod}_I : \mathsf{Sign}_I^{op} \to \mathsf{CAT}$ assigns to each signature Σ the category of Σ-models. For each signature morphism $\sigma : \Sigma \to \Sigma'$ the functor*

  $$\mathsf{Mod}_I(\sigma) : \mathsf{Mod}_I(\Sigma') \to \mathsf{Mod}_I(\Sigma)$$

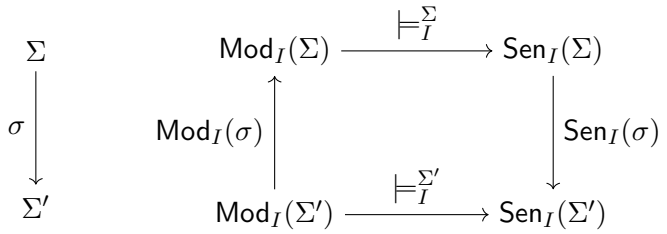
  translates Σ'-models to Σ-models
- *a satisfaction relation $\models_I^{\Sigma} \subseteq |\mathsf{Mod}_I(\Sigma)| \times \mathsf{Sen}_I(\Sigma)$ determines the satisfaction of Σ-sentences by Σ-models for each signature $\Sigma \in |\mathsf{Sign}_I|$.*

Satisfaction must be preserved under change of signature, that is, for any signature morphism $\sigma : \Sigma \to \Sigma'$, for any $\varphi \in \mathsf{Sen}_I(\Sigma)$ and $M' \in |\mathsf{Mod}_I(\Sigma')|$

$$\left(M' \models_I^{\Sigma'} \mathsf{Sen}_I(\sigma)(\varphi)\right) \Leftrightarrow \left(\mathsf{Mod}_I(\sigma)(M') \models_I^{\Sigma} \varphi\right) \tag{2.1}$$

(Satisfaction condition)

Showing that "truth is invariant under change of notation".
Graphically,



In later Chapter 3 and Chapter 4 we will formalise many-valued logics as institutions, in these cases the equivalence on the satisfaction condition (2.1) can be replaced by an equality (c.f. [ACEGG91]):

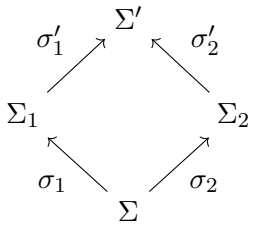
$$\Big(M' \models_I^{\Sigma'} \mathsf{Sen}_I(\sigma)(\varphi)\Big) = \big(\mathsf{Mod}_I(\sigma)(M') \models_I^{\Sigma} \varphi\big) \tag{2.2}$$

In conclusion, if we are dealing with institutions where a sentence is either satisfied or not, the satisfaction condition (2.1) needs to be proved. Otherwise, if we are dealing with many-valued institutions, for example in the case of fuzzy logics where a sentence is satisfied by some degree [JGMB21], the satisfaction condition (2.2) is the one that needs to be proved.

Notation: When there is no confusion we will omit the subscript I when referring to components of an institution as well as the subscript Σ on the satisfaction relation, that is, $I = (\mathsf{Sign}, \mathsf{Sen}, \mathsf{Mod}, \models)$. The functor $\mathsf{Mod}(\sigma) : \mathsf{Mod}(\Sigma') \to \mathsf{Mod}(\Sigma)$ is called the σ-reduct functor and it will be often denoted by $-|_\sigma$

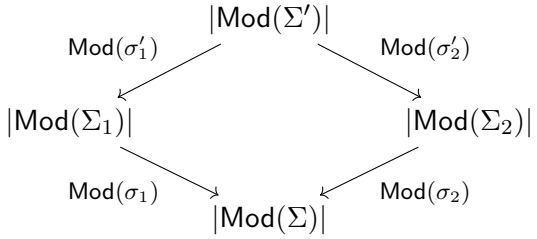
Next we present a property that will be particularly useful for the upcoming examples in this section.

Definition 9. *Consider an institution $I = (\mathsf{Sign}_I, \mathsf{Sen}_I, \mathsf{Mod}_I, \models_I)$ with $\mathsf{Sign}_I = \mathsf{Sign}$, defined in Example 4, and the following pushout in* Sign*:*



The diagram above is an amalgamation square if and only if: for any Σ_1-algebra M_1 and Σ_2-algebra M_2 such that $M_1|_{\sigma_1} = M_2|_{\sigma_2}$, there exists a unique Σ'-algebra M', called the amalgamation of M_1 and M_2, such that $M'|_{\sigma'_1} = M_1$ and $M'|_{\sigma'_2} = M_2$. In the absence of uniqueness of M' we say that is a weak amalgamation square*.*

Note that the reduct functor, defined in Example 7, and denoted by Mod maps the above pushout in the following pullback:



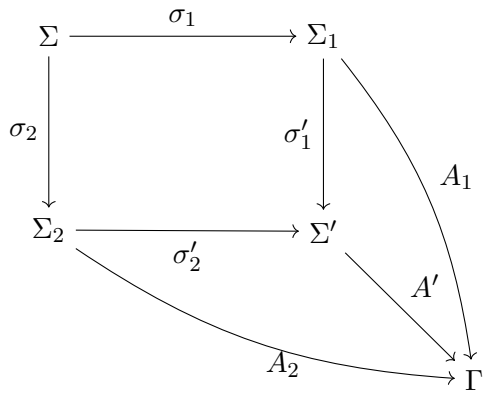
An institution has the *amalgamation property* if and only if each pushout of signatures is an amalgamation square.

Lemma 2. *A pushout in* Sign*, the category of algebraic signatures, is an amalgamation square.*

The proof is adapted from the one found in [Dia08] for institution **FOL** that will be later introduced in this chapter.

Proof. Consider the following pushout in the category Sign, where $\Gamma = \langle S^\Gamma, \Omega^\Gamma \rangle$ is an *hyper*-signature defined as:

- $S^\Gamma = \mathsf{Set}$, class of all sets
- For any sets $s_1, ..., s_n, s$, $\Omega^\Gamma_{s_1 ... s_n, s} = \mathsf{Set}(s_1 \times ... \times s_n, s)$ is the set of all functions $s_1 \times ... \times s_n \to s$



Notice that a Σ-algebra A ($A \in |\mathsf{Mod}(\Sigma)|$) is just a signature morphism $\Sigma \to \Gamma$.
Given a Σ_1-algebra A_1 and a Σ_2-algebra A_2 (that is, $\Sigma_1 \xrightarrow{A_1} \Gamma$ and $\Sigma_2 \xrightarrow{A_2} \Gamma$) such that $A_1|_{\sigma_1} = A_2|_{\sigma_2}$, according to the definition of pushout, there exists a unique signature morphism $A' : \Sigma' \to \Gamma$, equivalent to saying that A' is a Σ'-algebra, such that the above diagram commutes. Hence,

$$A_1 = A' \circ \sigma'_1 \text{ and } A_2 = A' \circ \sigma'_2$$

Thus,

$$A_1|_{\sigma'_1} = A' \text{ and } A_2|_{\sigma'_2} = A'$$

□

All of the examples of institutions we present in this section have the amalgamation property. The proof can be done by adapting the one above to suit the different types of signatures. For examples **EQ**, equational logic, and **FOL**, first order logic, the property is particularly useful for proving that satisfaction is preserved under change of signature.

We end this section with a few examples of logics formulated as institutions: **PL**, propositional logic; **EQ**, equational logic; **FOL**, first order logic and **ML**, modal logic. For other examples we refer to [Dia08]. The description of each component of the institutions **PL**, **EQ** and **FOL**, provided as examples, can be found in [ST12]. We focus on filling a few missing details as well as presenting the proof for the satisfaction condition. The formalization of the institution **ML** was proposed as an exercise while writing this chapter.

Example 8. [***Propositional logic, PL***]
Institution ***PL*** *is defined by:*

- *Signatures are* $\mathrm{Prop} \in |\mathsf{Set}|$*, a set of symbols called propositions.*
- *For a signature* Prop*, the set of sentences is given by the following grammar:*

$$\varphi := \bot \mid p \mid \neg\varphi \mid \varphi \to \varphi$$

with $\top = \neg\bot$*,* $(\varphi \wedge \varphi') = \neg(\varphi \to \neg\varphi')$*,* $(\varphi \vee \varphi') = (\neg\varphi \to \varphi')$ *and with* $(\varphi \leftrightarrow \varphi') = ((\varphi \to \varphi') \wedge (\varphi' \to \varphi))$*.*
For any signature morphism $\sigma : \mathrm{Prop} \to \mathrm{Prop}'$*,* $\mathsf{Sen}_{PL}(\sigma)$ *translates sentences with symbols in* Prop *to sentences with symbols in* Prop′*, preserving the propositional connectives. For any* $p \in Prop$ *and* φ*,* φ' Prop*-sentences,* $\mathsf{Sen}_{PL}(\sigma)$ *is recursively defined as:*

  - $\mathsf{Sen}_{PL}(\sigma)(\bot) = \bot$
  - $\mathsf{Sen}_{PL}(\sigma)(p) = \sigma(p)$
  - $\mathsf{Sen}_{PL}(\sigma)(\neg\varphi) = \neg\, \mathsf{Sen}_{PL}(\sigma)(\varphi)$
  - $\mathsf{Sen}_{PL}(\sigma)(\varphi \to \varphi') = \mathsf{Sen}_{PL}(\sigma)(\varphi) \to \mathsf{Sen}_{PL}(\sigma)(\varphi')$

- *Models of signature* Prop *are sets* $X \subseteq \mathrm{Prop}$*, or equivalently, are functions* $M : \mathrm{Prop} \to \{\bot, \top\}$*. Given models* M_1 *and* M_2*, their morphism are inclusions of sets, i.e. for* $M_1, M_2 : \mathrm{Prop} \to \{\bot, \top\}$ *there is a morphism from model* M_1 *to model* M_2 *if for all* $p \in \mathrm{Prop}, M_2(p) = \top$ *when* $M_1(p) = \top$*.*
For any signature morphism $\sigma : \mathrm{Prop} \to \mathrm{Prop}'$*,*

$$\mathsf{Mod}_{PL}(\sigma) : \mathsf{Mod}_{PL}(\mathrm{Prop}') \to \mathsf{Mod}_{PL}(\mathrm{Prop})$$

maps any model $M' : \mathrm{Prop}' \to \{\bot, \top\}$ *to* $M' \circ \sigma : \mathrm{Prop} \to \{\bot, \top\}$*.*
Clearly, for any $p \in \mathrm{Prop}$*,* $(M'(\sigma(p)) = \top)$ *iff* $(\mathsf{Mod}(\sigma)(M')(p) = \top)$

- *The satisfaction relation is defined with* $M \models \varphi$ *standing for "*φ *is satisfied by model* M*" and* $M \not\models \varphi$ *for "*φ *is not satisfied by model* M*".*

  - $M \models^{\mathrm{Prop}}_{\boldsymbol{PL}} p$ *iff* $M(p) = \top$*, for any* $p \in \mathrm{Prop}$
  - $M \not\models^{\mathrm{Prop}}_{\boldsymbol{PL}} \bot$

– $M \models^{\mathrm{Prop}}_{\boldsymbol{PL}} (\neg\varphi)$ *iff* $\neg(M \models^{\mathrm{Prop}}_{\boldsymbol{PL}} \varphi) = M \not\models^{\mathrm{Prop}}_{\boldsymbol{PL}} \varphi$

– $M \models^{\mathrm{Prop}}_{\boldsymbol{PL}} (\varphi \to \varphi')$ *iff* $(M \models^{\mathrm{Prop}}_{\boldsymbol{PL}} \varphi \Rightarrow M \models^{\mathrm{Prop}}_{\boldsymbol{PL}} \varphi')$

The proof of the satisfaction condition is made by induction over the propositional connectives. Cases $p \in Prop$ and $\top$ are trivial:

$$(M' \models^{\mathrm{Prop}'}_{\boldsymbol{PL}} \mathsf{Sen}(\sigma)(\top)) = (M' \models^{\mathrm{Prop}'}_{\boldsymbol{PL}} \top) = (Mod(\sigma)(M') \models^{\mathrm{Prop}}_{\boldsymbol{PL}} \top)$$

$$\begin{aligned}
M' \models^{\mathrm{Prop}'}_{\boldsymbol{PL}} \mathsf{Sen}(\sigma)(p) =& \{\textit{def. of } \mathsf{Sen}(\sigma)\} \\
& M' \models^{\mathrm{Prop}'}_{\boldsymbol{PL}} \sigma(p) \\
=& \{\textit{def. of } \models\} \\
& M'(\sigma(p)) = \top \textit{ for } \sigma(p) \in \mathrm{Prop}' \\
=& \{\textit{def. of } \mathsf{Mod}(\sigma)\} \\
& Mod(\sigma)(M')(p) = \top \textit{ for } p \in \mathrm{Prop} \\
=& \{\textit{def. of } \models\} \\
& Mod(\sigma)(M') \models^{\mathrm{Prop}}_{\boldsymbol{PL}} p
\end{aligned}$$

For $\varphi, \varphi' \in \mathsf{Sen}_{PL}(Prop)$.

$$\begin{aligned}
M' \models^{\mathrm{Prop}'}_{\boldsymbol{PL}} \mathsf{Sen}(\sigma)(\neg\,\varphi) =& \{\textit{def. of } \mathsf{Sen}(\sigma)\} \\
& M' \models^{\mathrm{Prop}'}_{\boldsymbol{PL}} \neg\, \mathsf{Sen}(\sigma)(\varphi) \\
=& \{\textit{def. of } \models\} \\
& \neg(M' \models^{\mathrm{Prop}'}_{\boldsymbol{PL}} \mathsf{Sen}(\sigma)(\varphi)) \\
=& \{\textit{Induction Hypothesis}\} \\
& \neg(Mod(\sigma)(M') \models^{\mathrm{Prop}}_{\boldsymbol{PL}} \varphi) \\
=& \{\textit{def. of } \models\} \\
& Mod(\sigma)(M') \models^{\mathrm{Prop}}_{\boldsymbol{PL}} \neg\,\varphi
\end{aligned}$$

$$\begin{aligned}
M' \models^{\mathrm{Prop}'}_{\boldsymbol{PL}} \mathsf{Sen}(\sigma)(\varphi \to \varphi') =& \{\textit{def. of } \mathsf{Sen}(\sigma)\} \\
& (M' \models^{\mathrm{Prop}'}_{\boldsymbol{PL}} \mathsf{Sen}(\sigma)(\varphi)) \Rightarrow (M' \models^{\mathrm{Prop}'}_{\boldsymbol{PL}} \mathsf{Sen}(\sigma)(\varphi')) \\
=& \{\textit{Induction Hypothesis twice}\} \\
& (Mod(\sigma)(M') \models^{\mathrm{Prop}}_{\boldsymbol{PL}} \varphi) \Rightarrow (Mod(\sigma)(M') \models^{\mathrm{Prop}}_{\boldsymbol{PL}} \varphi') \\
=& \{\textit{def. of } \models\} \\
& Mod(\sigma)(M') \models^{\mathrm{Prop}}_{\boldsymbol{PL}} (\varphi \to \varphi')
\end{aligned}$$

Example 9. [***Equational logic, EQ***]
Given any signature $\Sigma = \langle S, \Omega \rangle$, defined in Example 4, a Σ-equation $(\forall X)\, t = t'$ involves:

- *a finite set of variables X, such that, X_s denotes variables of sort s. We write $(x : s)_\Sigma$ to denote a variable x in Σ of sort s,*

- t, t' *are* Σ*-terms,* $|T_{\Sigma,s}(X)|$ *for some* $s \in S$,

where $T_\Sigma(X)$ *is the* Σ*-algebra of terms generated by* Σ *using variables* X*, and* $T_{\Sigma,s}(X)$ *is the set of terms of sort* s *inductively defined as follows:*

- *if* x *is a variable of sort* s *then* $x \in T_{\Sigma,s}(X)$
- *if* $c :\rightarrow s$ *an 0-ary operation in* Ω *then* $c \in T_{\Sigma,s}(X)$
- *if* $f : s_1 \times ... \times s_n \rightarrow s$ *an operation in* Ω *and* t_1 *in* $T_{\Sigma,s_1}(X)$*,...,* tn *in* $T_{\Sigma,s_n}(X)$ *terms then* f"$(t_1, ..., t_n)$" $\in T_{\Sigma,s}(X)$ [1]

$T_\Sigma(X)$ *basically builds complex terms from simpler terms. We will assume that variables are disjoint from operation symbols in the signature.*
The ***EQ*** *institution is defined by:*

- *Signatures are objects* $\Sigma \in |\mathsf{Sign}|$*, the category of algebraic signatures. A signature morphism* $\sigma : \Sigma \rightarrow \Sigma'$ *between* $\Sigma = \langle S, \Omega \rangle$ *and* $\Sigma' = \langle S', \Omega' \rangle$ *is defined by a pair of functions* $\sigma_{sorts} : S \rightarrow S'$ *and*

$$\sigma_{ops} = \langle \sigma_{w,s} : \Omega_{w,s} \rightarrow \Omega'_{\sigma * _{sorts}(w), \sigma_{sorts}(s)} \rangle_{w \in S^*, s \in S}$$

  with $\sigma_{sorts}(s_1...s_n) = \sigma_{sorts}(s_1)...\sigma_{sorts}(s_n)$

- *Sentences are universally quantified equations* $(\forall X)\, t = t'$*. For each signature morphism* $\sigma : \Sigma \rightarrow \Sigma'$*,* Σ*-equations are translated into* Σ'*-equations. So* $\mathsf{Sen}(\sigma)((\forall X)\, t = t')$ *is mapped to* $(\forall X')\, \mathsf{Sen}(\sigma')(t = t')$*, where:*
  - *For any variable* x *in* Σ *of sort* s*,* x' *is a variable in* Σ' *of sort* $\sigma(s)$ *and* $x' \in X'$*, where*

$$X' = \{\, (x' : \sigma(s))_{\Sigma'} \mid (x : s)_\Sigma \,\}$$

  - $\sigma' : \langle S, \Omega \uplus X \rangle \rightarrow \langle S', \Omega' \uplus X' \rangle$ *canonically extends* σ *by translating variables* $(x : s)_\Sigma$ *to* $(x' : \sigma(s))_{\Sigma'}$*.*

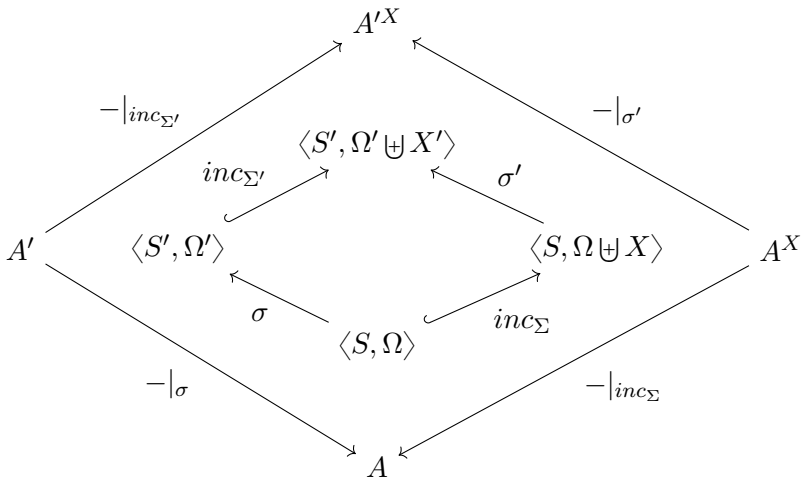
- *Given a signature* Σ*, the category* $\mathsf{Mod}(\Sigma)$ *is the category* $\mathsf{Alg}(\Sigma)$ *of* Σ*-algebras and* Σ*-homomorphisms, defined in Example 5. For a* Σ'*-algebra* A' *and signature morphism* $\sigma : \Sigma \rightarrow \Sigma'$*,* $A'|_\sigma$ *is the* σ*-reduct of algebra* A' *such that* $A'_{\sigma(s)} = (A'|_\sigma)_s$ *for any sort* $s \in S$ *and similarly for any operation in* Ω *and for any* Σ*-term* t*:* $A'_{\sigma(t)} = (A'|_\sigma)_t$*. Hence,* $\mathsf{Mod} : \mathsf{Sign}^{op} \rightarrow \mathsf{CAT}$ *is a functor.*

- *For each signature* $\Sigma = \langle S, \Omega \rangle$*, the satisfaction relation, as in [MMBH15], is the Tarskian satisfaction defined recursively as:*
  - $A \models^\Sigma_{\boldsymbol{EQ}} t = t'$ *when* $A_t = A_{t'}$
  - $A \models^\Sigma_{\boldsymbol{EQ}} (\forall X)\, t = t'$ *when* $A' \models^{\langle S, \Omega \uplus X \rangle}_{\boldsymbol{EQ}} t = t'$ *for any* inc_Σ*-expansion* A' *of* A*. where* $inc_\Sigma : \langle S, \Omega \rangle \hookrightarrow \langle S, \Omega \uplus X \rangle$ *is an inclusion morphism that enriches* $\langle S, \Omega \rangle$ *with the set of variables* X

  Note that saying that A' *is an* inc_Σ*-expansion of* A *is the same as saying* $A'|_{inc_\Sigma} = A$*.*

[1] Quotation indicates that is not an evaluation and that terms are words.

Consider the following diagram where $inc_\Sigma : \Sigma \hookrightarrow \langle S, \Omega \uplus X \rangle$, $inc_{\Sigma'} : \Sigma' \hookrightarrow \langle S', \Omega' \uplus X' \rangle$ *are inclusion morphisms:*



A pushout of inc_Σ *and* σ *is the signature* $\langle S', \Omega' \uplus X' \rangle$ *and the two morphisms* $inc_{\Sigma'}$ *and* σ'. *For any pushout in* Sign *by the amalgamation property, Lemma 2, for a* Σ*-algebra* A *such that* $A'|_\sigma = A = A^X|_{inc_\Sigma}$ *there is an unique* $\langle S', \Omega' \uplus X' \rangle$*-algebra* A'^X *such that* $A'^X|_{\sigma'} = A^X$ *and* $A'^X|_{inc_{\Sigma'}} = A'$.
Thus,

$$
\begin{aligned}
A' \models^{\Sigma'}_{\boldsymbol{EQ}} \mathsf{Sen}(\sigma)((\forall X)\, t = t') = & \{\textit{def. of } \mathsf{Sen}\} \\
& A' \models^{\Sigma'}_{\boldsymbol{EQ}} (\forall X')\, \mathsf{Sen}(\sigma')(t = t') \\
= & \{\textit{def. of } \models\} \\
& A'^X \models^{\langle S', \Omega' \uplus X' \rangle}_{\boldsymbol{EQ}} \mathsf{Sen}(\sigma')(t = t') \textit{ for any } inc_{\Sigma'}\textit{-exp. } A'^X \textit{ of } A' \\
= & \{\textit{Induction hypothesis}\} \\
& A'^X|_{\sigma'} \models^{\langle S', \Omega' \uplus X' \rangle}_{\boldsymbol{EQ}} t = t' \textit{ for any } inc_{\Sigma'}\textit{-expansion } A'^X \textit{ of } A' \\
= & \{\textit{by Lemma 2 and } A^X|_{inc_\Sigma} = A\} \\
& A^X \models^{\langle S, \Omega \uplus X \rangle}_{\boldsymbol{EQ}} t = t' \textit{ for any } inc_\Sigma\textit{-expansion } A^X \textit{ of } A \\
= & \{\textit{def. of } \models \textit{ and } A = A'|_\sigma\} \\
& A'|_\sigma \models^{\Sigma}_{\boldsymbol{EQ}} (\forall X)\, t = t'
\end{aligned}
$$

Example 10. [***First order logic with equality, FOL***]
Institution ***FOL*** *consists of:*

- *Signatures are triples* $\Theta = \langle S, \Omega, \Pi \rangle$ *where* S *is a set of sort names,* $\Omega = \langle \Omega_{w,s} \rangle_{w \in S^*, s \in S}$ *is a* $(S^* \times S)$*-sorted set of operation names and* $\Pi = \langle \Pi_w \rangle_{w \in S^*}$ *is a family of sets of predicate symbols.*
  A signature morphism $\theta : \langle S, \Omega, \Pi \rangle \to \langle S', \Omega', \Pi' \rangle$ *involves three functions:*
  - $\theta_{sorts} : S \to S'$
  - $\theta_{ops} = \langle \theta_{w,s} : \Omega_{w,s} \to \Omega'_{\theta^*_{sorts}(w), \theta_{sorts}(s)} \rangle_{w \in S^*, s \in S}$

– $\theta_{preds} = \langle (\theta_{preds})_w : \Pi_w \to \Pi'_{\theta^*_{sorts}(w)} \rangle_{w \in S^*}$

for any $w = s_1...s_n \in S^*$, $\theta^*_{sorts}(w) = \theta_{sorts}(s_1), ..., \theta_{sorts}(s_n)$.

- *Sentences for a signature* $\Theta = \langle S, \Omega, \Pi \rangle$ *are first order formulas built from equational and predicate atoms using the Boolean connectives* $(\neg, \vee, \wedge, \to)$ *and quantifiers* $(\forall, \exists)$. *Equational atoms are of the form* $t = t'$ *where* t *and* t' *are* $\langle S, \Omega \rangle$*-terms and predicate atoms are of the form* $\pi(t_1, .., t_n)$ *where* $\pi \in \Pi_{t_1,...,t_n}$ *and* $t_1, ..., t_n$ *are terms of sort* $s_1, ..., s_n$ *respectively. Thus, sentences are given by the following grammar:*

$$\varphi := \ t = t' \mid \pi(t_1, ., , , t_n) \mid \neg\varphi \mid \varphi \to \varphi \mid (\forall X)\, \varphi$$

with $\exists\varphi = \neg\forall\neg\varphi$ *plus all the abbreviates taken in Example 8, and* X *is a set of variables, defined in Example 9.*
Given a signature morphism $\theta : \langle S, \Omega, \Pi \rangle \to \langle S', \Omega', \Pi' \rangle$, *sentence translation is defined for Boolean connectives as in Example 8 and:*

– $\mathsf{Sen}(\theta)(t = t') = \ (\theta(t) = \theta(t'))$
– $\mathsf{Sen}(\theta)(\pi(t_1, ..., t_n)) = \ \theta_{preds}(\pi)(\theta(t_1), ..., \theta(t_n))$
– $\mathsf{Sen}(\theta)((\forall X)\, \varphi) = \ (\forall X')\, \mathsf{Sen}(\theta')(\varphi)$

with X' *defined in Example 9 and* θ' *canonically extending* θ *by translating variables* $(x : s)_\Theta$ *to* $(x' : \theta(s))_{\Theta'}$.

- *A first order structure* M *for a signature* $\langle S, \Omega, \Pi \rangle$ *consists of:*

– *a carrier set* $|M|_s$ *for each* $s \in S$
– *a function* $f_M : |M|_{s_1} \times ... \times |M|_{s_n} \to |M|_s$ *for each operation* $f \in \Omega_{s_1...s_n,s}$
– $\pi_M \subseteq |M|_{s_1} \times ... \times |M|_{s_n}$ *for each predicate* $\pi \in \Pi_{s_1...s_n}$.

We write $\pi_M(a_1, ..., a_n)$ *for* $\langle a_1, ..., a_n \rangle \in \pi_M$
For two structures M *and* N *over the same signature, a morphism* $h : M \to N$ *is a family of functions* $h = \langle h_s : |M|_s \to |N|_s \rangle_{s \in S}$ *that preserves operations and predicates, i.e. for* $\pi \in \Pi_{s_1,...,s_n}$ *and* $a_1 \in |M|_{s_1}$, ..., $a_n \in |M|_{s_n}$, *if* $\pi_M(a_1, ..., a_n)$ *then* $\pi_N(h_{s_1}(a_1), ..., h_{s_n}(a_n))$.
For a $\langle S', \Omega', \Pi' \rangle$*-structure* M', *its* θ*-reduct is a* $\langle S, \Omega, \Pi \rangle$*-structure* $M'|_\theta$ *such that* $M'_{\theta(s)} = (M'|_\theta)_s$ *for any* $s \in S$, *and similarly for any operation, predicate symbol or term.*

- *For a signature* $\Theta = \langle S, \Omega, \Pi \rangle$ *and a* Θ*-structure* M *the satisfaction relation is defined recursively as*

– $M \models^\Theta_{\boldsymbol{FOL}} t = t'$ *when* $M_t = M_{t'}$, *where* M_t *is the interpretation of term* t *in* M *defined recursively* $M_{f(t_1,...,t_n)} = M_f(M_{t_1}, ..., M_{t_n})$ *for* $f \in \Omega$
– $M \models^\Theta_{\boldsymbol{FOL}} \pi(t_1, ..., t_n)$ *when* $M_\pi(M_{t_1}, ..., M_{t_n})$
– $M \models^\Theta_{\boldsymbol{FOL}} \neg\varphi$ *when* $M \not\models^\Theta_{\boldsymbol{FOL}} \varphi$
– $M \models^\Theta_{\boldsymbol{FOL}} \varphi \to \varphi'$ *when* $M \models^\Theta_{\boldsymbol{FOL}} \varphi \Rightarrow M \models^\Theta_{\boldsymbol{FOL}} \varphi'$

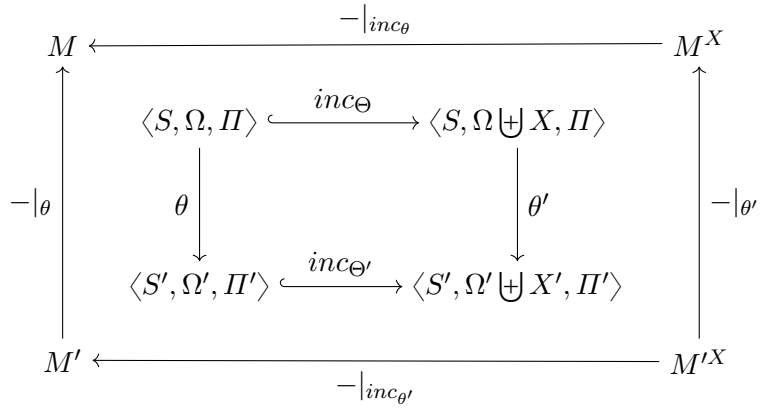
– $M \models^{\Theta}_{\mathbf{FOL}} (\forall X)\, \varphi$ *when* $M' \models^{\Theta'}_{\mathbf{FOL}} \varphi$ *for any* $\Theta' = \langle S, \Omega \biguplus X, \Pi \rangle$*-expansion* M' *of* M

For signatures $\Theta = \langle S, \Omega, \Pi \rangle$, $\Theta' = \langle S', \Omega', \Pi' \rangle$, *a signature morphism* $\theta : \Theta \to \Theta'$ *and a* Θ'*-structure* M' *the proof of the satisfaction condition is done by induction over the connectives. The proofs for cases* $\neg$ *and* $\to$ *are like the one presented for the case of the* ***PL*** *institution, Example 8. For the remaining connectives the proofs go as follows:*

$$
\begin{aligned}
M' \models^{\Theta'}_{\mathbf{FOL}} Sen(\theta)(t = t') = & \{\textit{def. of Sen}\} \\
& M' \models^{\Theta'}_{\mathbf{FOL}} \theta(t) = \theta(t') \\
= & \{\textit{def. of} \models\} \\
& M'_{\theta(t)} = M'_{\theta(t')} \\
= & \{\textit{def. of } \theta\textit{-reduct}\} \\
& (M'|_\theta)_t = (M'|_\theta)_{t'} \\
= & \{\textit{def. of} \models\} \\
& M'|_\theta \models^{\Theta}_{\mathbf{FOL}} t = t'
\end{aligned}
$$

$$
\begin{aligned}
M' \models^{\Theta'}_{\mathbf{FOL}} Sen(\theta)(\pi(t_1, ..., t_n)) = & \{\textit{def. of Sen}\} \\
& M' \models^{\Theta'}_{\mathbf{FOL}} \theta_{preds}(\pi)(\theta(t_1), ..., \theta(t_n)) \\
= & \{\textit{def. of} \models\} \\
& M'_{\theta_{preds}(\pi)}(M'_{\theta(t_1)}, ..., M'_{\theta(t_n)}) \\
= & \{\textit{def. of } \theta\textit{-reduct: } M'_{\theta_{preds}(\pi)} = (M'|_\theta)_\pi\textit{, similarly for terms}\} \\
& (M'|_\theta)_\pi((M'|_\theta)_{t_1}, ..., (M'|_\theta)_{t_n}) \\
= & \{\textit{def. of} \models\} \\
& M'|_\theta \models^{\Theta}_{\mathbf{FOL}} \pi(t_1, ..., t_n)
\end{aligned}
$$

Consider the expansions $inc_\Theta : \langle S, \Omega, \Pi \rangle \hookrightarrow \langle S, \Omega \biguplus X, \Pi \rangle$ *and* $inc_{\Theta'} : \langle S', \Omega', \Pi' \rangle \hookrightarrow \langle S', \Omega' \biguplus X', \Pi' \rangle$ *and the signature morphism* θ' *that canonically extends* θ *by mapping variables of sort* s *to* $\theta(s)$. *A pushout of* inc_θ *and* θ *is the signature* $\langle S', \Omega' \uplus X', \Pi' \rangle$ *and the two morphisms* $inc_{\theta'}$ *and* θ'. *First order structures have the amalgamation property for a* Θ*-algebra* M *such that* $M'|_\theta = M = M^X|_{inc_\theta}$*; then there is an unique* $\langle S', \Omega' \uplus X', \Pi \rangle$*-structure* M'^X *such that* $M'^X|_{\theta'} = M^X$ *and* $M'^X|_{inc_{\theta'}} = M'$.
The proof for the amalgamation property in ***FOL*** *is like the one for Lemma 2 but the hypersignature is replaced by* $\Gamma = \langle S^\Gamma, \Omega^\Gamma, \Pi^\Gamma \rangle$ *where* S^Γ *and* Ω^Γ *are defined exactly the same and, for any sets* $s_1, ..., s_n$, $\Pi^\Gamma_{s_1,...,s_n} = \mathscr{P}(s_1, ..., s_n)$ *i.e. all subsets of* $s_1 \times ... \times s_n$.



$$
\begin{aligned}
M' \models^{\Theta'}_{\textbf{\textit{FOL}}} \mathsf{Sen}(\theta)((\forall X)\,\varphi) = & \{\textit{def. of Sen}\} \\
& M' \models^{\Theta'}_{\textbf{\textit{FOL}}} (\forall X')\,\mathsf{Sen}(\theta')(\varphi) \\
= & \{\textit{def. of } \models\} \\
& M'^X \models^{\langle S', \Omega' \uplus X', \Pi' \rangle}_{\textbf{\textit{FOL}}} Sen(\theta')(\varphi) \textit{ for any } \Theta'\textit{-exp. } M'^X \textit{ of } M' \\
= & \{\textit{Induction hypthesis}\} \\
& M'^X|_{\theta'} \models^{\langle S', \Omega' \uplus X', \Pi' \rangle}_{\textbf{\textit{FOL}}} \varphi \textit{ for any } \Theta'\textit{-expansion } M'^X \textit{ of } M' \\
= & \{\textit{by Lemma 2 and } M^X|_{inc_\Theta} = M\} \\
& M^X \models^{\langle S, \Omega \uplus X, \Pi \rangle}_{\textbf{\textit{FOL}}} \varphi \textit{ for any } \Theta\textit{-expansion } M^X \textit{ of } M \\
= & \{\textit{def. of } \models \textit{ and } M'|_\theta = M\} \\
& M'|_\theta \models^{\Theta}_{\textbf{\textit{FOL}}} (\forall X)\,\varphi
\end{aligned}
$$

Even thought the concept of *sub-institution* is not explored in this dissertation we still want to briefly discuss it. It is easy to see that ***PL*** institution is similar to ***FOL*** by restricting signatures to those with an empty set of sorts hence, ***PL*** signatures are sets that consist of 0-ary predicate symbols. Institution ***EQ*** is also similar to ***FOL*** by dropping the predicate symbols and sentences just being universally quantified equations. Therefore, ***PL*** and ***EQ*** can be framed as a *sub-institution* of ***FOL***.

Example 11. [***Modal Logic, ML***]

- *A signature is a pair* $\Sigma = \langle \mathrm{Act}, \mathrm{Prop} \rangle$ *where* $\mathrm{Prop} \in |\mathsf{Set}|$ *is a set of symbols usually called propositions and* $\mathrm{Act} \in |\mathsf{Set}|$ *is a set of action symbols. A signature morphism* $\sigma : \Sigma \to \Sigma'$ *is a pair of functions* $\sigma_{\mathrm{Prop}} : \mathrm{Prop} \to \mathrm{Prop}'$ *and* $\sigma_{\mathrm{Act}} : \mathrm{Act} \to \mathrm{Act}'$.

- *The set of sentences is given by the grammar:*

$$\varphi := \bot \mid p \mid \neg\varphi \mid \varphi \to \varphi \mid [a]\,\varphi$$

  with all the abbreviations taken from Example 8. Also, $\langle a \rangle$*, for any* $a \in \mathrm{Act}$ *is the dual connective of* $[a]$*, that is,* $\langle a \rangle \varphi = \neg\,[a]\,\neg\,\varphi$*. Sentence translation for Boolean connectives and propositions is defined as in Example 8 and:*

$$Sen(\sigma)([a]\,\varphi) = [\,\sigma_{\mathrm{Act}}(a)\,]\,Sen(\sigma)(\varphi)$$

- *Models are triples* $M = (W, R, V)$ *where* W *is a set of states,* $R = (R_a : W \times W \to \{0,1\})_{a \in \mathrm{Act}}$ *is an* Act*-indexed family of binary relations on* W *also called accessibility relation (we write* $R_a(v,u)$ *if state* v *relates to states* u *through action* a*) and* $V : \mathrm{Prop} \to \mathscr{P}(W)$ *is a valuation function that for a given proposition assigns a set of states in which the proposition holds.*
  Let $\sigma : \mathrm{Prop} \to \mathrm{Prop}'$ *be a signature morphism and* $M' = (W', R', V')$ *be a* Prop'*-model. The* σ*-reduct of* M' *is* $Mod(\sigma)(M') = (W, R, V)$ *a* Prop*-model where* $W = W'$*,* $R_a(u,v) = R'_{\sigma_{\mathrm{Act}}(a)}(u,v)$ *for any* $a \in \mathrm{Act}$ *and* $w \in V(p)$ *iff* $w \in V'(\sigma_{\mathrm{Prop}}(p))$ *for any* $p \in \mathrm{Prop}$*.*

- *For any state* $w \in W$*,* $p \in \mathrm{Prop}$ *and* $a \in \mathrm{Act}$ *the satisfaction relation is defined by:*
  - $M, w \models^{\mathrm{Prop}}_{\boldsymbol{ML}} p$ *iff* $w \in V(p)$
  - $M, w \not\models^{\mathrm{Prop}}_{\boldsymbol{ML}} \bot$
  - $M, w \models^{\mathrm{Prop}}_{\boldsymbol{ML}} \neg\varphi$ *iff* $M, w \not\models^{\mathrm{Prop}}_{\boldsymbol{ML}} \varphi$
  - $M, w \models^{\mathrm{Prop}}_{\boldsymbol{ML}} \varphi \to \varphi'$ *iff* $(M, w \models^{\mathrm{Prop}}_{\boldsymbol{ML}} \varphi) \Rightarrow (M, w \models^{\mathrm{Prop}}_{\boldsymbol{ML}} \varphi')$
  - $M, w \models^{\mathrm{Prop}}_{\boldsymbol{ML}} [a]\,\varphi$ *iff (for all* $v \in W$*,* $R_a(w,v) \Rightarrow M, v \models^{\mathrm{Prop}}_{\boldsymbol{ML}} \varphi$*)*

where "$M, w \models \varphi$*" means that* φ *holds in state* w *of model* M *and "*$M, w \not\models \varphi$*" means that sentence* φ *does not hold in state* w *of* M*.*
Proof of the satisfaction relation is given by induction over the connectives. Case $\bot$ *is trivial. For the case of a symbol* $p \in \mathrm{Prop}$*,*

$$
\begin{aligned}
M', w \models^{\mathrm{Prop}'}_{\boldsymbol{ML}} Sen(\sigma)(p) =& \{\textit{def of } Sen(\sigma)\} \\
& M', w \models^{\mathrm{Prop}'}_{\boldsymbol{ML}} \sigma_{\mathrm{Prop}}(p) \\
=& \{\textit{def of } \models\} \\
& w \in V'(\sigma_{\mathrm{Prop}}(p)) \\
=& \{\textit{def of } \sigma\textit{-reduct}\} \\
& w \in V(p) \\
=& \{\textit{def of } \models\} \\
& Mod(\sigma)(M'), w \models^{\mathrm{Prop}}_{\boldsymbol{ML}} p
\end{aligned}
$$

Proof for connectives $\neg$ *and* $\to$ *are similar to Example 8.*
For connective $[a]\,\varphi$ *the proof is the following:*

$$
\begin{aligned}
M', w \models^{\mathrm{Prop}'}_{\boldsymbol{ML}} Sen([a]\,\varphi) =& \{\textit{def of } Sen(\sigma)\} \\
& M', w \models^{\mathrm{Prop}'}_{\boldsymbol{ML}} [\sigma_{\mathrm{Act}}(a)]\, Sen(\sigma)(\varphi) \\
=& \{\textit{def of } \models\} \\
& \textit{for all } v \in W',\ R'_{\sigma_{\mathrm{Act}}(a)}(w,v) \Rightarrow M', v \models^{\mathrm{Prop}'}_{\boldsymbol{ML}} Sen(\sigma)(\varphi) \\
=& \{\textit{def. } \sigma\textit{-reduct (W=W', } R_a(w,v) = R'_{\sigma_{\mathrm{Act}}(a)}(w,v)\textit{) and by the I.H.}\} \\
& \textit{for all } v \in W,\ R_a(w,v) \Rightarrow Mod(\sigma)(M'), v \models^{\mathrm{Prop}}_{\boldsymbol{ML}} \varphi \\
=& \{\textit{def of } \models\} \\
& Mod(\sigma)(M'), w \models^{\mathrm{Prop}}_{\boldsymbol{ML}} [a]\,\varphi
\end{aligned}
$$

2.3 Specification Theory

This section is divided in three parts: *structured specification*, *constructor implementation* and *abstract specification*. The main goal of this section is to make the reader familiar with the classical definitions of specification theory that can be found in [ST12] as well as provide some context to the concepts presented. We also refer to [Sri90] as a simple introductory overview of the topic.

First it is important to define what a specification is. A specification is a description of a signature Σ and some class of Σ-algebras[2].
The purpose of *algebraic specification* is to model systems as algebras and specifications will amount to characterisations of classes of algebras. For example, if we want to model a group of people such that they are related by "Friendship" the class of algebras that model this system is possibly infinite. Aside from this fact, our goal is not to have any algebra model our system but to have an algebra model our system *and* satisfying some basic requirements of a "Friendship", for example, "Friendship" is a reflexive relation ("I am friends with myself") and it is symmetric ("If I am her friend then she is my friend"). Imposing on the class of algebras such requirements will result in a more restricted class: now any algebra in the class not only models our system of people but it also specifies that "Friendship" is a reflexive and symmetric relation.

In summary, algebraic specification is a technique to model systems as algebras. In addition, instead of taking a often impractical approach of listing every algebra associated to a signature it takes a more viable approach of describing the functional behaviour of the algebras in the class by listing the proprieties they must verify. Thus, the syntax of an algebraic specification encompasses:

- A signature Σ which provides names for the components of an algebra
- Set of axioms, Σ-sentences, which describe proprieties that should be satisfied by any algebra in the class so specified

We say that an algebra A models a specification, which is equivalent to say that it is a correct realization of the specification. if for any $\varphi \in \Phi$, $A \models \varphi$. Therefore, the semantics of an algebraic specification is associated with the signature and the satisfaction relation between algebras and sentences.

For this entire section concepts are defined in an arbitrary but fixed institution, since there is no mention of the syntax and semantic of a particular logical system. This approach allows this theory to be applicable to a wide range of logical systems that can be framed as institutions, abstracting from the nuances of a certain logic and focusing on building complex specifications that model a specific envisaged system.

Structured Specification

One of the other main concerns of algebraic specification is being able to build specifications in a structured manner. As specifications get more complex they may have a larger number of sorts, operation symbols and axioms which makes them harder to reason about. For those reasons, trying to find an algebra that models such a large and abstract system can be a complicated, if not an impossible task. Hence, there is a need for larger specifications to be

[2]We will use "algebra" and "model" interchangeably since an algebra can be seen as a model for a signature

built and extended from smaller ones (called flat specifications) through specification-building operators. In this way, larger specifications can be understood more easily as we can reason about the proprieties that they have by understanding their individual components and the operations between them.
For each specification SP each specification-building operator presented will define a signature $Sig(SP)$ and a class of models $Mod(SP)$.

As previously mentioned, we will start with the concept of a flat specification, that consists of a signature and a set of axioms. Since usually these simple specifications are the starting point to build new ones through composition.

Definition 10. *A* flat specification*, also called a presentation, is a pair* $\langle \Sigma, \Phi \rangle$ *where* Σ *is a signature and* $\Phi \subseteq \mathsf{Sen}(\Sigma)$*.* $Mod(\langle \Sigma, \Phi \rangle)$ *is the class of all models of specification* $\langle \Sigma, \Phi \rangle$*.*

Definition 11. ***Union***
Given two specifications SP *and* SP'*, both over the signature* Σ*, then*

- $Sig(SP \cup SP') = \Sigma$
- $Mod(SP \cup SP') = Mod(SP) \cap Mod(SP')$

This operator allows us to combine specifications with the restriction that they have to be both over the same signature.
It is easy to see that if SP and SP' are flat specifications with axioms Φ and Φ', respectively, then $Mod[SP \cup SP'] = Mod[\langle \Sigma, \Phi \cup \Phi' \rangle]$. The class of models that satisfies two flat specification over the same signature corresponds to all models that satisfy both sets of axioms. Notice, that combining specifications SP and SP' will result in a smaller class of algebras, that is, $Mod(SP \cup SP') \subseteq Mod(SP) \cup Mod(SP')$.

Definition 12. ***Translation***
Let SP *be a* Σ*-specification, that is* $Sig(SP) = \Sigma$*, and* $\sigma : \Sigma \to \Sigma'$ *a signature morphism, then*

- $Sig(SP \textbf{ with } \sigma) = \Sigma'$
- $Mod(SP \textbf{ with } \sigma) = \{M' \in \mathsf{Mod}(\Sigma') \mid M'|_\sigma \in Mod(SP)\}$

If SP is a flat specification then SP **with** σ has the same models as $\langle \Sigma', \sigma(\Phi) \rangle$, where sentence translation is under the sentence functor $\mathsf{Sen}(\sigma) : \mathsf{Sen}(\Sigma) \to \mathsf{Sen}(\Sigma')$.
Translation is a basic renaming operator that is particularly important when combined with **Union** introduced in Definition 11. Consider the inclusion morphisms $i : \Sigma \to \Sigma \cup \Sigma'$ and $i' : \Sigma' \to \Sigma \cup \Sigma'$. We can define a "sum" operator [Sri90] that allows us to combine specifications over different signatures:

$$SP + SP' := (SP \textbf{ with } i) \cup (SP' \textbf{ with } i') \tag{2.3}$$

where,

- $Sig(SP + SP') = \Sigma \cup \Sigma'$
- $Mod(SP + SP') = \{M \mid M \in \mathsf{Mod}(\Sigma \cup \Sigma'),\ M|_i \in Mod(SP) \text{ and } M|_{i'} \in Mod(SP')\}$

Definition 13. ***Hiding***
If SP' is a Σ'-specification and $\sigma : \Sigma \to \Sigma'$ a signature morphism, then

- $Sig(SP' \textbf{ hide via } \sigma) = \Sigma$
- $Mod(SP' \textbf{ hide via } \sigma) = \{M'|_\sigma | \, M' \in Mod(SP')\}$

Hiding is usually used to hide auxiliary sorts and operations.

Definition 14. *Given specifications SP and SP', with $Sig(SP) = Sig(SP')$ we say that SP' implements or refines SP, in symbols, $SP \rightsquigarrow SP'$ if, $Mod(SP') \subseteq Mod(SP)$.*

Conceptually, for two specifications SP and SP' over the same signature, if SP' refines SP then SP' is more concrete (it satisfies at least all the requirements of SP). Implementations lead to a methodology of stepwise refinement where successive implementations of a specification are developed so that every step produces a more concrete specification than the previous one.

The next theorem shows that the simple implementation relation is transitive. This theorem can be found in [ST12, Proposition 7.13] and we provide a proof for it.

Theorem 1. [***Vertical Composition***] *If $SP \rightsquigarrow SP'$ and $SP' \rightsquigarrow SP''$ then $SP \rightsquigarrow SP''$.*

Proof. Since $SP \rightsquigarrow SP'$ and $SP' \rightsquigarrow SP''$, by definition of $\rightsquigarrow$, $Sig(SP) = Sig(SP') = Sig(SP'')$. Also by the definition of $\rightsquigarrow$ and hypothesis, $Mod(SP') \subseteq Mod(SP)$ and $Mod(SP'') \subseteq Mod(SP')$. Trivially, $Mod(SP'') \subseteq Mod(SP') \subseteq Mod(SP)$. Therefore, $SP \rightsquigarrow SP''$. □

The next theorem can be found in [ST12, Proposition 7.16] with a slightly different presentation. In the reference the theorem is presented for an arbitrary operator **sbo** between a Σ-specification and a Σ'-specification and the horizontal composition is a consequence from the fact that **sbo** is monotone. For specifications SP over Σ and SP' over Σ' the operator **sbo** is said to be monotone when, $Mod(SP) \subseteq Mod(SP')$, then $Mod(\textbf{sbo}(SP)) \subseteq Mod(\textbf{sbo}(SP'))$. Such presentation will not be explored in this dissertation although it is possible to prove that **Union**, **Translation** and **Hiding** operators are monotone.

Theorem 2. [***Horizontal Composition***] *If $SP_1 \rightsquigarrow SP_1'$ and $SP_2 \rightsquigarrow SP_2'$ then,*

1. $(SP_1 \cup SP_2) \rightsquigarrow (SP_1' \cup SP_2')$
2. $(SP_1 \textbf{ with } \sigma) \rightsquigarrow (SP_1' \textbf{ with } \sigma)$
3. $(SP_1 \textbf{ hide via } \sigma) \rightsquigarrow (SP_1' \textbf{ hide via } \sigma)$

Constructor implementations

So far, for one specification to implement another they both need to have the same signature. Such definition of implementation can be too strict since some practices in software development often require implementation decisions (introducing new features or reusing previously defined features) which results in dealing with different signatures along the development process. The notion of a function called *constructor* that transforms models into other models, possible with different signatures, and of *constructor implementation* are the tools to be used in this scenarios.

Definition 15. *Given signatures* $\Sigma_1, ..., \Sigma_n, \Sigma$, *a* constructor *is a function*

$$k : \mathsf{Mod}(\Sigma_1) \times ... \times \mathsf{Mod}(\Sigma_n) \to \mathsf{Mod}(\Sigma)$$

For a constructor k *and a set of constructors* $k_i : \mathsf{Mod}(\Sigma_i^1) \times ... \times \mathsf{Mod}(\Sigma_i^{k_i}) \to \mathsf{Mod}(\Sigma_i)$ *for* $1 \leqslant i \leqslant n$, *the constructor*

$$k(k_1, ..., k_n) : \mathsf{Mod}(\Sigma_1^1) \times ... \times \mathsf{Mod}(\Sigma_1^{k_1}) \times ... \times \mathsf{Mod}(\Sigma_n^1) \times ... \times \mathsf{Mod}(\Sigma_n^{k_n}) \to \mathsf{Mod}(\Sigma)$$

is obtained by the usual composition of functions.

It is possible to check that all the specification-building operators defined before in this chapter are constructors.

The idea behind the following definition of a constructor implementation is that a specification SP can be built from several specifications such that applying a construction on top of them leads to a model of SP, that by definition satisfies the requirements of SP.

Definition 16. *Given specifications* $SP, SP_1, ..., SP_n$ *and a constructor*

$$k : \mathsf{Mod}(Sig(SP_1)) \times ... \times Mod(Sig(SP_n)) \to \mathsf{Mod}(Sig(SP))$$

$(SP_1, ..., SP_n)$ *is a constructor implementation via* k *of* SP, *in symbols* $SP \rightsquigarrow_k (SP_1, ..., SP_n)$ *if for all* $M_i \in Mod(SP_i)$, $k(M_1, ..., M_n) \in Mod(SP)$.
We say that the implementation involves a decomposition if $n > 1$.

Abstract Specification

Often in software development some model does not satisfy exactly the requirements of a specification due to certain implementation details, which are hidden from the observer of the system. In this situations a model may still satisfy requirements abstractly if it exhibits the desired observable behaviour. Abstractor implementations emerge to define what precisely means two models being identical from an observational perspective. This will be expressed by the use of an equivalence relation $\equiv$ between models, over the same signature. We write $M \equiv M'$ when a model M is equivalent to a model M', with reference to the relation $\equiv$.

Definition 17. *Given an equivalence relation* $\equiv \subseteq \mathsf{Mod}(\Sigma) \times \mathsf{Mod}(\Sigma)$ *between models of signature* Σ, *then*

- $Sig($**abstractor** SP **w.r.t.** $\equiv) = \Sigma$
- $Mod($**abstractor** SP **w.r.t.** $\equiv) =$
  $= \{M \mid M \in \mathsf{Mod}(\Sigma)$ *and* $M \equiv N$ *for some* $N \in Mod(SP)\}$

The closure of $Mod(SP)$ under $\equiv$ is $Mod($**abstractor** SP **w.r.t.** $\equiv)$.

Definition 18. *Consider* SP *and* SP' *specifications over the same signature* Σ *and* $\equiv \subseteq \mathsf{Mod}(\Sigma) \times \mathsf{Mod}(\Sigma)$ *an equivalence relation. We say that* SP' *implements abstractly* SP *with respect to* $\equiv$, *in symbols* $SP \rightsquigarrow^{\equiv} SP'$ *whenever*

$$Mod(SP') \subseteq Mod(\textbf{abstractor } SP \textbf{ w.r.t } \equiv)$$

Constructor and abstractor implementation can also be combined:

Definition 19. *[MBHM18] Let $SP, SP_1, .., SP_n$ be specifications over signatures Σ, $\Sigma_1, ...,$ Σ_n, respectively. Consider a constructor*

$$k : \mathsf{Mod}(\Sigma_1) \times ... \times \mathsf{Mod}(\Sigma_n) \to \mathsf{Mod}(\Sigma)$$

*and an equivalence relation $\equiv \subseteq \mathsf{Mod}(\Sigma) \times \mathsf{Mod}(\Sigma)$. We say that $(SP_1, ..., SP_n)$ is an abstractor implementation of SP via k w.r.t $\equiv$, in symbols $SP \rightsquigarrow_k^{\equiv} (SP_1, ..., SP_n)$ if for all $M_i \in Mod(SP_i)$, $k(M_1, .., M_n) \in Mod($**abstractor** SP **w.r.t.** $\equiv)$*

Finally, we recall [MBHM18, Theorem 3], whose proof can be found in the reference, regarding composition of abstractor implementations via a constructor.

Theorem 3. *Consider specifications $SP, SP_1, .., SP_n$ over signatures $\Sigma, \Sigma_1, .., \Sigma_n$, respectively, a contructor*

$$k : \mathsf{Mod}(\Sigma_1) \times ... \times \mathsf{Mod}(\Sigma_n) \to \mathsf{Mod}(\Sigma)$$

and an equivalence relation $\equiv \subseteq \mathsf{Mod}(\Sigma) \times \mathsf{Mod}(\Sigma)$ such that $SP \rightsquigarrow_k^{\equiv} (SP_1, ..., SP_n)$.
For each $i \in \{1, .., n\}$, let $SP_i \rightsquigarrow_{k_i}^{\equiv_i} (SP_i^1, ..., SP_i^{r_i})$ with specification $SP_i^1, .., SP_i^{r_i}$ over signatures $\Sigma_i^1, .., \Sigma_i^{r_i}$, respectively. Consider constructors

$$k_i : \mathsf{Mod}(\Sigma_i^1) \times ... \times \mathsf{Mod}(\Sigma_i^{r_i}) \to \mathsf{Mod}(\Sigma_i)$$

and equivalences $\equiv_i \subseteq \mathsf{Mod}(\Sigma_i) \times \mathsf{Mod}(\Sigma_i)$.
Suppose that k preserves the abstractions $\equiv_i$, that is, for each $M_i, N_i \in \mathsf{Mod}(\Sigma_i)$ such that $M_i \equiv_i N_i$, $k(M_1, ..., M_n) \equiv k(N_1, .., N_n)$. Then,

$$SP \rightsquigarrow_{k(k_1,..,k_n)}^{\equiv} (SP_1^1, .., SP_1^{r_1}, ..., SP_n^1, ..., SP_n^{r_n})$$

Chapter 3

An institution for paraconsistent transition systems

This chapter starts with the characterization of (Metric) twisted algebras in Section 3.1, such structures amount to a class of residuated lattices over a set of possible truth values A. The focus from Section 3.2 to Section 3.5 is to properly formalize all the necessary ingredients: signatures, models, sentences and satisfaction relation (alongside the proof of the satisfaction condition) of institution $\boldsymbol{S}(\mathscr{A})$ parametric to some fix twisted algebra $\mathscr{A}$.

3.1 (Metric) Twisted algebras

The positive and negative relations proposed in [CMB22] and presented again in this dissertation take values in a truth space that is defined in terms of a residuated lattice. Hence, this section introduces the necessary concepts to define an algebraic structure over which we will define our logics for paraconsistent systems $\boldsymbol{S}(\mathscr{A})$, in this chapter, and for paraconsistent processes $\boldsymbol{P}(\mathscr{A})$, in Chapter 4.

A nonempty set A together with two binary operations $\vee$ and $\wedge$ on A (read "join" and "meet", respectively) is called a lattice [BS81], in symbols $\langle A, \wedge, \vee \rangle$, if it satisfies the following identities:

Commutative laws:

- $x \vee y \approx y \vee x$
- $x \wedge y \approx y \wedge x$

Idempotent laws:

- $x \vee x \approx x$
- $x \wedge x \approx x$

Associative laws:

- $x \vee (y \vee z) \approx (x \vee y) \vee z$
- $x \wedge (y \wedge z) \approx (x \wedge y) \wedge z$

Absorption laws:

- $x \approx x \vee (x \wedge y)$
- $x \approx x \wedge (x \vee y)$

One of the properties of lattices is that any pairs of elements $x, y \in A$, have a supremum $x \vee y$ and an infimum $x \wedge y$.

A residuated lattice $\langle A, \sqcap, \sqcup, 1, 0, \odot, \rightharpoonup, e\rangle$, over a nonempty set A, is an algebraic structure where

- $\langle A, \sqcap, \sqcup\rangle$ is a lattice.
- $\langle A, \odot, e\rangle$ is a monoid, i.e. set A is equipped with a binary operation $\odot$ and an identity element e.
- $\odot$ has a right adjoint $\rightharpoonup$, called the residuum. Thus,

$$a \odot b \leqslant c \text{ iff } b \leqslant a \rightharpoonup c$$

We will, however, focus on a particular class of residuated lattices in which the lattice meet ($\sqcap$) and monoidal composition ($\odot$) coincide. Thus the adjunction above is stated as

$$a \sqcap b \leqslant c \text{ iff } b \leqslant a \rightharpoonup c$$

If any nonempty subset of A has a supremum and an infimum then a residuated lattice over A is said to be complete [Coh12]. We shall only focus on complete residuated lattices that are bounded by a maximal element 1 and a minimal element 0 and that are *integral*, which means that $1 = e$. The following Lemma will be needed in the sequel.

Lemma 3. *[BEGR11] Let $\langle A, \sqcap, \sqcup, 1, 0, \odot, \rightharpoonup, e\rangle$ be a complete residuated lattice and I a finite set. Then,*

$$b \rightharpoonup \left(\bigsqcap_{i\in I} a_i\right) = \bigsqcap_{i\in I}(b \rightharpoonup a_i) \tag{3.1}$$

$$\left(\bigsqcup_{i\in I} a_i\right) \rightharpoonup b = \bigsqcap_{i\in I}(a_i \rightharpoonup b) \tag{3.2}$$

where $\bigsqcup$ and $\bigsqcap$ are the distributed versions of $\sqcup$ and $\sqcap$, respectively.

The binary relation $\leqslant$ mentioned so far is defined on a set A of possible truth values by

$$a \leqslant b \text{ iff } a = a \sqcap b$$

Trivially, $\leqslant$ is a total order relation [BS81] since the following conditions hold for any $a, b \in A$:

- $a \leqslant a$ (reflexivity)
- $a \leqslant b$ and $b \leqslant a$ implies $a = b$ (antisymmetry)
- $a \leqslant b$ and $b \leqslant c$ implies $a \leqslant c$ (transitivity)
- $a \leqslant b$ or $b \leqslant a$

Additionally, we will enforce a pre-linearity condition,

$$(a \rightharpoonup b) \sqcup (b \rightharpoonup a) = 1 \tag{3.3}$$

A residuated lattice obeying prelinearity is known as a MTL-algebra [EG01]. With a slight abuse of nomenclature, the designation iMTL-algebra, from *integral MTL-algebra*, will be used in the sequel for the class of semantic structures considered, i.e. prelinear, complete and integral residuated lattices such that $\sqcap$ and $\odot$ coincide.

Example 12. [***iMTL-algebras***]

- *the Boolean algebra* $\mathbf{2} = \langle \{\bot, \top\}, \wedge, \vee, \top, \bot, \rightarrow \rangle$
- $\mathbf{3} = \langle \{\top, u, \bot\}, \wedge_3, \vee_3, \top, \bot, \rightarrow_3 \rangle$*, where*

$\wedge_3$	$\bot$	u	$\top$
$\bot$	$\bot$	$\bot$	$\bot$
u	$\bot$	u	u
$\top$	$\bot$	u	$\top$

$\vee_3$	$\bot$	u	$\top$
$\bot$	$\bot$	u	$\top$
u	u	u	$\top$
$\top$	$\top$	$\top$	$\top$

$\rightarrow_3$	$\bot$	u	$\top$
$\bot$	$\top$	$\top$	$\top$
u	$\bot$	$\top$	$\top$
$\top$	$\bot$	u	$\top$

- $\ddot{\boldsymbol{G}} = \langle [0,1], \min, \max, 0, 1, \rightarrow \rangle$*, with implication defined as* $a \rightarrow b = \begin{cases} 1 & if\, a \leqslant b \\ b & otherwise \end{cases}$

We will focus on iMTL-algebras $\mathbf{A}$ whose carrier A supports a metric space (A, d), for a suitable choice of a metric d, i.e. $d\colon A \times A \rightarrow \mathbb{R}^+$ such that $d(x, y) = 0$ iff $x = y$ and $d(x, y) \leqslant d(x, z) + d(z, y)$. Such metric is particularly important to give a concrete meaning to the *vagueness-inconsistency* square depicted in Figure 1.2. As stated, this metric d is able to access the level of vagueness or inconsistency of a given pair of weights. For example, for any pair $(x, y) \in A \times A$ whenever $d(x, y) = 0$ the pair (x, y) represents consistent information, otherwise the pair represents information with some level of vagueness or inconsistency.

Example 13.

- *for* $\boldsymbol{2}$ *a suitable metric is* $d(x, y) = \begin{cases} 0 & if\, x = y \\ 1 & otherwise \end{cases}$
- *for* $\boldsymbol{3}$ *a suitable metric is*

d	$\bot$	u	$\top$
$\bot$	0	1	2
u	1	0	1
$\top$	2	1	0

- *For* $\ddot{\boldsymbol{G}}$ *a suitable metric is* $d(x, y) = \sqrt{(x - y)^2}$

In order to operate with pairs of truth weights reference [CMB22] introduced the notion of $\mathbf{A}$-*twisted algebra*. This algebraic structure, will play a crucial role in the semantics of our institution and it consists of an enrichment of a twist-structure [Kra98] with a metric. The latter is relevant to interpret the consistency operator of the logic:

Definition 20. *[CMB22] Given a iMTL-algebra* $\boldsymbol{A} = \langle A, \sqcap, \sqcup, 1, 0, \rightharpoonup \rangle$ *enriched with a metric* d*.* $\mathscr{A} = \langle A \times A, \boxplus, \boxplus, \Longrightarrow, /\!/, D \rangle$ *is a* $\boldsymbol{A}$*-twisted algebra defined for any* (a, b), $(c, d) \in A \times A$ *as:*

- $(a, b) \boxplus (c, d) = (a \sqcap c, b \sqcup d)$
- $(a, b) \boxplus (c, d) = (a \sqcup c, b \sqcap d)$
- $(a, b) \Rightarrow (c, d) = (a \rightharpoonup c, a \sqcap d)$

- $/\!\!/(a,b) = (b,a)$
- $D((a,b),(c,d)) = \sqrt{d(a,c)^2 + d(b,d)^2}$

The order in $\boldsymbol{A}$ *is lifted to* $\mathscr{A}$ *as* $(a,b) \preceq (c,d)$ *iff* $a \leqslant c$ *and* $b \geqslant d$.

We finish this subsection by listing and proving a few properties that will be used in future chapters.

Property 1. Operators $\sqcap$ and $\sqcup$ are monotone, i.e. if $a \leqslant c$ and $b \leqslant d$ then, $(a \sqcap b) \leqslant (c \sqcap d)$, similarly for $\sqcup$.

Proof. We give the proof for $\sqcup$ as the one for $\sqcap$ is similar. Let $a \leqslant c$ and $b \leqslant d$ if

- $(a \sqcup b) = a$ and $(c \sqcup d) = c$ then by hypothesis $(a \sqcup b) = a \leqslant c = (c \sqcup d)$.
- $(a \sqcup b) = a$ and $(c \sqcup d) = d$ so $c \leqslant d$ by hypothesis $(a \sqcup b) = a \leqslant c \leqslant d = (c \sqcup d)$.
- $(a \sqcup b) = b$ and $(c \sqcup d) = c$ so we have that $(a \sqcup b) = b \leqslant d \leqslant c = (c \sqcup d)$.
- $(a \sqcup b) = b$ and $(c \sqcup d) = d$ then by hypothesis $(a \sqcup b) \leqslant (c \sqcup d)$

That is, if $a \leqslant c$ and $b \leqslant d$ then $(a \sqcup b) \leqslant (c \sqcup d)$ always holds. □

Property 2. [MNM16, Lemma 1.(38)] If $a \leqslant b$ then $(c \rightharpoonup a) \leqslant (c \rightharpoonup b)$.

Proof. We start by noting that for any $a, b, c \in A$, $(c \rightharpoonup a) \leqslant (c \rightharpoonup (a \sqcup b))$ holds in any residuated lattice $\langle A, \sqcap, \sqcup, 1, 0, \odot, \rightharpoonup, e\rangle$ [Koz94, Table 3.(19)].
Given that $a \leqslant b$, we have that $(a \sqcup b) = b$ thus, $(c \rightharpoonup a) \leqslant (c \rightharpoonup (a \sqcup b))$ is equivalent to $(c \rightharpoonup a) \leqslant (c \rightharpoonup b)$. □

Property 3. [MNM16, Lemma 1.(39)] If $a \leqslant b$ then $(b \rightharpoonup c) \leqslant (a \rightharpoonup c)$.

Proof. A consequence of a complete residuated lattice [BEGR11] is: Since $a \leqslant b$ then $(a \sqcup b) = b$ so $b \rightharpoonup c$ is equal to $(a \sqcup b) \rightharpoonup c$. Using 3.2, we have that $(a \sqcup b) \rightharpoonup c$ is equal to $(a \rightharpoonup c) \sqcap (b \rightharpoonup c)$. Thus,

$$((a \sqcup b) \rightharpoonup c) = (a \rightharpoonup c) \sqcap (b \rightharpoonup c) \leqslant (a \rightharpoonup c)$$

□

Property 4. If $a \leqslant c$ and $b \geqslant d$ then, $(a \rightharpoonup b) \geqslant (c \rightharpoonup d)$

Proof. Since $a \leqslant c$ by Property 3. we have that $(c \rightharpoonup d) \leqslant (a \rightharpoonup d)$ and since $d \leqslant b$ by Property 2. we have that $(a \rightharpoonup d) \leqslant (a \rightharpoonup b)$. Hence, $(c \rightharpoonup d) \leqslant (a \rightharpoonup d) \leqslant (a \rightharpoonup b)$, that is, $(a \rightharpoonup b) \geqslant (c \rightharpoonup d)$. □

Property 5. Operators $\sqcap\!\!\sqcap$ and $\sqcup\!\!\sqcup$ are monotone, i.e. if $(a,b) \preccurlyeq (c,d)$ and $(a',b') \preccurlyeq (c',d')$ then $(a,b) \sqcap\!\!\sqcap (a',b') \preccurlyeq (c,d) \sqcap\!\!\sqcap (c',d')$, similarly for $\sqcup\!\!\sqcup$.

Proof. We give the proof for operator $\sqcap\!\!\sqcap$ as the one for $\sqcup\!\!\sqcup$ is alike. Let $(a, b) \preccurlyeq (c, d)$ and $(a', b') \preccurlyeq (c', d')$, that is, using the definition of $\preccurlyeq$

$$(a \leqslant c \text{ and } a' \leqslant c') \text{ and } (b \geqslant d \text{ and } b' \geqslant d')$$

Using Property 1.,

$$(a \sqcap a') \leqslant (c \sqcap c') \text{ and } (b \sqcup b') \geqslant (d \sqcup d')$$

Using the definition of $\preccurlyeq$,

$$(a \sqcap a', b \sqcup b') \preccurlyeq (c \sqcap c', d \sqcup d')$$

Finally, using definition of $\sqcap\!\!\sqcap$,

$$(a, b) \sqcap\!\!\sqcap (b, b') \preccurlyeq (c, d) \sqcap\!\!\sqcap (c', d')$$

□

Property 6. Relation $\preccurlyeq$ is transitive, i.e. if $(a, b) \preccurlyeq (c, d)$ and $(c, d) \preccurlyeq (e, f)$ then $(a, b) \preccurlyeq (e, f)$.

Proof. Let $(a, b) \preccurlyeq (c, d)$ and $(c, d) \preccurlyeq (e, f)$. Using the definition of $\preccurlyeq$,

$$a \leqslant c \text{ and } b \geqslant d \text{ and } c \leqslant e \text{ and } d \geqslant f$$

Which is equivalent to

$$a \leqslant c \leqslant e \text{ and } b \geqslant d \geqslant f$$

Hence, $a \leqslant e$ and $b \geqslant f$, by the definition of $\preccurlyeq$ that is the same as writing $(a, b) \preccurlyeq (e, f)$. □

Let us fix a given twisted algebra $\mathscr{A}$. In the following sections we will introduce the ingredients for an institution $\boldsymbol{S}(\mathscr{A}) = (\mathsf{Sign}_{\boldsymbol{S}(\mathscr{A})}, \mathsf{Sen}_{\boldsymbol{S}(\mathscr{A})}, \mathsf{Mod}_{\boldsymbol{S}(\mathscr{A})}, \models_{\boldsymbol{S}(\mathscr{A})})$ that will be abbreviated to $\boldsymbol{S} = (\mathsf{Sign}_{\boldsymbol{S}}, \mathsf{Sen}_{\boldsymbol{S}}, \mathsf{Mod}_{\boldsymbol{S}}, \models_{\boldsymbol{S}})$ whenever the choice of $\mathscr{A}$ is not essential.

3.2 The signatures

Definition 21. *A signature is defined as a tuple* $\langle \mathrm{Act}, \mathrm{Prop} \rangle$ *where* Act *is a set of action symbols and* Prop *is a set of propositions.*
Let $\Sigma = \langle \mathrm{Act}, \mathrm{Prop} \rangle$ *and* $\Sigma' = \langle \mathrm{Act}', \mathrm{Prop}' \rangle$ *be signatures. A signature morphism* $\sigma : \Sigma \to \Sigma'$ *is a pair of functions* $\sigma_{\mathrm{Prop}} : \mathrm{Prop} \to \mathrm{Prop}'$ *and* $\sigma_{\mathrm{Act}} : \mathrm{Act} \to \mathrm{Act}'$.

Signature morphisms can be seen as a change of notation, that is, a change of names of the proposition and action symbols.

Lemma 4. *Signatures* $\langle \mathrm{Act}, \mathrm{Prop} \rangle$ *and their morphisms form a category,* $\mathsf{Sign}_{\boldsymbol{S}}$.

Proof. Morphisms between two signatures in $\mathsf{Sign}_{\boldsymbol{S}}$ are functions between sets such that identities exist and the composition of functions is associative, as expected. □

Notice that $\mathsf{Sign}_{\boldsymbol{S}}$ is a sub-category of Sign, the category of algebraic signatures, defined in Example 4. For any signature $\langle \mathrm{Act}, \mathrm{Prop} \rangle$, the set Act can be seen as a set of sorts and the set Prop as a set of 0-ary operations. Hence, for any signature $\Sigma \in |\mathsf{Sign}_{\boldsymbol{S}}|$ then $\Sigma \in |\mathsf{Sign}|$. For any signatures $\Sigma = \langle \mathrm{Act}, \mathrm{Prop} \rangle$, $\Sigma' = \langle \mathrm{Act}', \mathrm{Prop}' \rangle \in |\mathsf{Sign}_{\boldsymbol{S}}|$, any morphism from Σ to Σ' in the collection $\mathsf{Sign}_{\boldsymbol{S}}(\Sigma, \Sigma')$ is also in the collection $\mathsf{Sign}(\Sigma, \Sigma')$.

3.3 The models

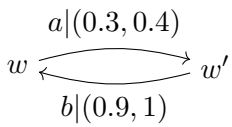
Definition 22. *Let $\Sigma = \langle \mathrm{Act}, \mathrm{Prop} \rangle$ be a signature. A Σ-paraconsistent transition system, abbreviated to PTS, is a tuple $M = (W, R, V)$ such that:*

- *W is a nonempty set of states,*
- *$R = (R_a \colon W \times W \to A \times A)_{a \in \mathrm{Act}}$ is an Act-indexed family of relations R_a for each $a \in \mathrm{Act}$. For any pair of states $(w_1, w_2) \in W \times W$ relation R_a assigns a pair $(\Vdash, \nVdash) \in A \times A$ where $\Vdash$ represents the evidence degree of the transition from w_1 to w_2 occurring through action a and $\nVdash$ represents the evidence degree of the transition being prevented from occurring.*
- *$V : W \times \mathrm{Prop} \to A \times A$ is a valuation function, that assigns to a proposition $p \in \mathrm{Prop}$ at a given state w a pair $(\Vdash, \nVdash) \in A \times A$ such that $\Vdash$ is the truth degree of p holding in w and $\nVdash$ the degree of not holding*

We write $(w, w', \Vdash, \nVdash) \in R_a$ to represent a transition from state w to w' through action a such that $R_a(w, w') = (\Vdash, \nVdash)$.

For any pair $(\Vdash, \nVdash) \in A \times A$, $(\Vdash, \nVdash)^+$ denotes $\Vdash$ and $(\Vdash, \nVdash)^-$ denotes $\nVdash$.

Example 14. *Consider $\boldsymbol{S}(\ddot{\boldsymbol{G}})$ and the signature $\Sigma = \langle \{a, b\}, \{p\} \rangle$. The following model $M = (W, R, V)$ is a PTS-model over Σ with $V(w, q) = (0, 1)$ and $V(w', q) = (1, 0.2)$.*



Note that $R_a(w, w')$ represents vague information, $V(w, q)$ represents consistent information and $V(w', q)$, and $R_b(w', w)$ represent paraconsistent information.

Definition 23. *A morphism connecting two $\langle \mathrm{Act}, \mathrm{Prop} \rangle$-PTS$s$ (W, R, V) and (W', R', V') is a function $h : W \to W'$, such that*

- *for each $a \in \mathrm{Act}$, $R_a(w_1, w_2) \preccurlyeq R'_a(h(w_1), h(w_2))$, and*
- *for any $p \in \mathrm{Prop}$, $w \in W$, $V(w, p) \preccurlyeq V'(h(w), p)$.*

We say that M and M' are isomorphic, in symbols $M \cong M'$, whenever there are morphisms $h : M \to M'$ and $h^{-1} : M' \to M$ such that $h' \circ h = id_{W'}$ $h \circ h' = id_W$.

Lemma 5. *$\langle \mathrm{Act}, \mathrm{Prop} \rangle$-PTSs and their corresponding morphisms form a category denoted by $\mathsf{Mod}_{\boldsymbol{S}}$*

Proof. Morphisms between two PTS-models can be seen as a function between sets of states, such that, identities exist and the composition of functions is associative, satisfying the categorical properties. □

Definition 24. *Let $\sigma : \langle \mathrm{Act}, \mathrm{Prop} \rangle \to \langle \mathrm{Act}', \mathrm{Prop}' \rangle$ be a signature morphism and $M' = (W', R', V')$ a $\langle \mathrm{Act}', \mathrm{Prop}' \rangle$-PTS. The σ-reduct of M' is $M|_\sigma = (W, R, V)$, a $\langle \mathrm{Act}, \mathrm{Prop} \rangle$-PTS, where*

- $W = W'$,
- *for* $p \in \text{Prop}$, $w \in W$, $V(w,p) = V'(w,\sigma(p))$, *and*
- *for* $w, v \in W$ *and* $a \in \text{Act}$, $R_a(w,v) = R'_{\sigma(a)}(w,v)$.

Lemma 6. *Reducts preserve morphisms.*

Proof. Let $\sigma : \langle \text{Act}, \text{Prop} \rangle \to \langle \text{Act}', \text{Prop}' \rangle$ be a signature morphism and $h : W \to W'$ a morphism between models $M = (W, R, V)$ and $M' = (W', R', V')$. The σ-reduct of M is the model $M|_\sigma = (W|_\sigma, R|_\sigma, V|_\sigma)$, and the σ-reduct of M' is the model $M'|_\sigma = (W'|_\sigma, R'|_\sigma, V'|_\sigma)$. We want to show that there exists a morphism from $W|_\sigma$ to $W'|_\sigma$.

$$\begin{array}{ccc} M = (W,R,V) & \xrightarrow{\quad h \quad} & M' = (W',R',V') \\ \Big\downarrow -|_\sigma & & \Big\downarrow -|_\sigma \\ M|_\sigma = (W|_\sigma, R|_\sigma, V|_\sigma) & \xrightarrow{\quad h \quad} & M'|_\sigma = (W'|_\sigma, R'|_\sigma, V'|_\sigma) \end{array}$$

Notice that, by the definition of σ-reduct for any $w \in W$ then $w \in W|_\sigma$, similarly for any $w' \in W'$ then $w' \in W'|_\sigma$. Moreover, a morphism h between states W and W' is a morphism between $W|_\sigma$ and $W'|_\sigma$. We only have to prove that morphism $h : M|_\sigma \to M'|_\sigma$ preserves the source valuation and transition functions. Thus, for each $a \in Act$ and $w, v \in W|_\sigma$

$$\begin{aligned} (R|_\sigma)_a(w,v) =& \{\text{def. of } \sigma\text{-reduct}\} \\ & R_{\sigma(a)}(w,v) \\ \preccurlyeq& \{\text{def. of morphism}\} \\ & R'_{\sigma(a)}(h(w),h(v)) \\ =& \{\text{def. of } \sigma\text{-reduct}\} \\ & (R'|_\sigma)_a(h(w),h(v)) \end{aligned}$$

Similarly, for each $p \in Prop$ and $w, v \in W|_\sigma$

$$\begin{aligned} V|_\sigma(w,p) =& \{\text{def. of } \sigma\text{-reduct}\} \\ & V(w,\sigma(p)) \\ \preccurlyeq& \{\text{def. of morphism}\} \\ & V'(h(w),\sigma(p)) \\ =& \{\text{def. of } \sigma\text{-reduct}\} \\ & V'|_\sigma(h(w),p) \end{aligned}$$

□

Each signature morphism $\sigma : \Sigma \to \Sigma'$ ($\Sigma, \Sigma' \in |\mathsf{Sign}_{\boldsymbol{S}}|$) defines a functor $\mathsf{Mod}_{\boldsymbol{S}}(\sigma) : \mathsf{Mod}_{\boldsymbol{S}}(\Sigma') \to \mathsf{Mod}_{\boldsymbol{S}}(\Sigma)$ that maps PTS and morphisms to the corresponding reducts. More generally, this lifts to a functor, $\mathsf{Mod}_{\boldsymbol{S}} : (\mathsf{Sign}_{\boldsymbol{S}})^{op} \to \mathsf{CAT}$, mapping each signature to the category of its models, and each signature morphism to its reduct functor.

Lemma 7. $\mathsf{Mod}_{\boldsymbol{S}} : (\mathsf{Sign}_{\boldsymbol{S}})^{op} \to \mathsf{CAT}$ *is a functor.*

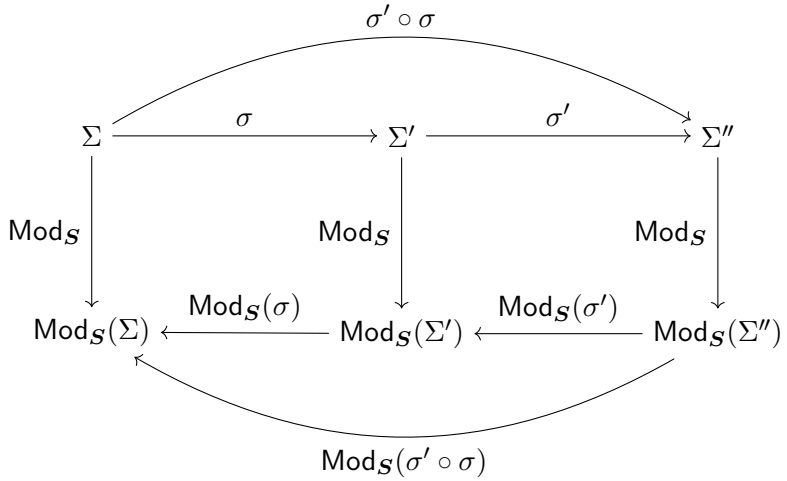
Proof. $(\mathsf{Sign}_{\boldsymbol{S}})^{op}$ is a category because by Lemma 4 we know that $\mathsf{Sign}_{\boldsymbol{S}}$ is a category and the opposite category of $\mathsf{Sign}_{\boldsymbol{S}}$ is a category, by Lemma 2. CAT is the category of all categories, Definition 5. $\mathsf{Mod}_{\boldsymbol{S}}$ is a functor between category $(\mathsf{Sign}_{\boldsymbol{S}})^{op}$ and CAT that maps a signature to the category of its models and each model morphism to its reduct.
Let $\mathsf{Mod}_{\boldsymbol{S}}(\Sigma)$ denote the category of Σ-models for any $\Sigma \in |\mathsf{Sign}_{\boldsymbol{S}}|$. For a signature morphism $\sigma : \Sigma \to \Sigma'$, defines the functor $\mathsf{Mod}_{\boldsymbol{S}}(\sigma) : \mathsf{Mod}_{\boldsymbol{S}}(\Sigma') \to \mathsf{Mod}_{\boldsymbol{S}}(\Sigma)$.
It is easy to check that $\mathsf{Mod}_{\boldsymbol{S}}$ preserves identities. Given the identity morphisms $id_\Sigma : \Sigma \to \Sigma$ and $id_{\mathsf{Mod}_{\boldsymbol{S}}(\Sigma)} : \mathsf{Mod}_{\boldsymbol{S}}(\Sigma) \to \mathsf{Mod}_{\boldsymbol{S}}(\Sigma)$ then,

$$\mathsf{Mod}_{\boldsymbol{S}}(id_\Sigma) = id_{\mathsf{Mod}_{\boldsymbol{S}}(\Sigma)}$$

We are left to prove that $\mathsf{Mod}_{\boldsymbol{S}}$ preserves composition, i.e. given any signature morphisms $\sigma : \Sigma \to \Sigma'$ and $\sigma' : \Sigma' \to \Sigma''$,

$$\mathsf{Mod}_{\boldsymbol{S}}(\sigma' \circ \sigma) = \mathsf{Mod}_{\boldsymbol{S}}(\sigma') \circ \mathsf{Mod}_{\boldsymbol{S}}(\sigma)$$

The following diagram illustrates composition:



We will prove that given a Σ''-model M'' its σ'-reduct is M' and M' σ-reduct is a Σ-model M **which is the same as saying** that M is the $(\sigma' \circ \sigma)$-reduct of M''.

Let $M'' = (W'', R'', V'')$ be a Σ''-model, the σ'-reduct of M'' is the model $M' = (W', R', V')$ in $\mathsf{Mod}_{\boldsymbol{S}}(\Sigma')$ such that, by the definition of reduct:

- $W' = W$
- $V'(w, p) = V''(w, \sigma'(p))$
- $R'_a(w, v) = R''_{\sigma'(a)}(w, v)$

Now let $M = (W, R, V)$ be the σ-reduct of M'. Then, by the definition of reduct:

- $W = W'$
- $V(w, p) = V'(w, \sigma(p))$

– $R_a(w,v) = R'_{\sigma(a)}(w,v)$

Since M' is the σ'-reduct of M'' we can write:

– $W = W' = W''$

– $V(w,p) = V'(w,\sigma(p)) = V''(w,\sigma'(\sigma(p))) = V''(w,(\sigma' \circ \sigma)(p))$

– $R_a(w,v) = R'_{\sigma(a)}(w,v) = R''_{\sigma'(\sigma(a))}(w,v) = R''_{(\sigma' \circ \sigma)(a)}(w,v)$

Thus, M is the $(\sigma' \circ \sigma)$-reduct of model M''. □

3.4 The sentences

Once characterised models let us define its syntax and the satisfaction relation.

Definition 25. *Given a signature* $\Sigma = \langle \mathrm{Act}, \mathrm{Prop} \rangle \in |\mathsf{Sign}_{\boldsymbol{S}}|$ *the set* $\mathsf{Sen}_{\boldsymbol{S}}(\Sigma)$ *of sentences is given by the following grammar*

$$\varphi ::= p \mid \bot \mid \neg\varphi \mid \varphi \to \varphi \mid \varphi \vee \varphi \mid \varphi \wedge \varphi \mid [a]\varphi \mid \langle a \rangle \varphi \mid \cancel{[a]}\varphi \mid \cancel{\langle a \rangle}\varphi \mid \circ\varphi$$

with $p \in \mathrm{Prop}$ *and* $a \in \mathrm{Act}$. *Note that* $\top = \neg\bot$ *and* $\varphi_1 \leftrightarrow \varphi_2 = (\varphi_1 \to \varphi_2) \wedge (\varphi_2 \to \varphi_1)$

Notice that unlike classical modal logic presented in Example 11 abbreviations in this paraconsistent modal logic can not be done. For example, in a paraconsistent setting it might not be true that $\langle a \rangle \varphi = \neg[a]\neg\varphi$.

Each signature morphism $\sigma : \Sigma \to \Sigma'$ between signatures $\Sigma = \langle \mathrm{Act}, \mathrm{Prop} \rangle$ and $\Sigma' = \langle \mathrm{Act}', \mathrm{Prop}' \rangle$ induces a sentence translation scheme $\mathsf{Sen}_{\boldsymbol{S}}(\sigma) : \mathsf{Sen}_{\boldsymbol{S}}(\Sigma) \to \mathsf{Sen}_{\boldsymbol{S}}(\Sigma')$ recursively defined as follows:

- $\mathsf{Sen}_{\boldsymbol{S}}(\sigma)(p) = \sigma_{\mathrm{Prop}}(p)$
- $\mathsf{Sen}_{\boldsymbol{S}}(\sigma)(\bot) = \bot$
- $\mathsf{Sen}_{\boldsymbol{S}}(\sigma)(\neg\varphi) = \neg\mathsf{Sen}_{\boldsymbol{S}}(\sigma)(\varphi)$
- $\mathsf{Sen}_{\boldsymbol{S}}(\sigma)(\varphi \to \varphi') = \mathsf{Sen}_{\boldsymbol{S}}(\sigma)(\varphi) \to \mathsf{Sen}_{\boldsymbol{S}}(\sigma)(\varphi')$
- $\mathsf{Sen}_{\boldsymbol{S}}(\sigma)(\varphi \vee \varphi') = \mathsf{Sen}_{\boldsymbol{S}}(\sigma)(\varphi) \vee \mathsf{Sen}_{\boldsymbol{S}}(\sigma)(\varphi')$
- $\mathsf{Sen}_{\boldsymbol{S}}(\sigma)(\varphi \wedge \varphi') = \mathsf{Sen}_{\boldsymbol{S}}(\sigma)(\varphi) \wedge \mathsf{Sen}_{\boldsymbol{S}}(\sigma)(\varphi')$
- $\mathsf{Sen}_{\boldsymbol{S}}(\sigma)([a]\,\varphi) = [\sigma_{Act}(a)]\,\mathsf{Sen}_{\boldsymbol{S}}(\sigma)(\varphi)$
- $\mathsf{Sen}_{\boldsymbol{S}}(\sigma)(\langle a \rangle \varphi) = \langle \sigma_{Act}(a) \rangle\, \mathsf{Sen}_{\boldsymbol{S}}(\sigma)(\varphi)$
- $\mathsf{Sen}_{\boldsymbol{S}}(\sigma)(\cancel{[a]}\,\varphi) = \cancel{[\sigma_{Act}(a)]}\,\mathsf{Sen}_{\boldsymbol{S}}(\sigma)(\varphi)$
- $\mathsf{Sen}_{\boldsymbol{S}}(\sigma)(\cancel{\langle a \rangle}\varphi) = \cancel{\langle \sigma_{Act}(a) \rangle}\,\mathsf{Sen}_{\boldsymbol{S}}(\sigma)(\varphi)$
- $\mathsf{Sen}_{\boldsymbol{S}}(\sigma)(\circ\varphi) = \circ\,\mathsf{Sen}_{\boldsymbol{S}}(\sigma)(\varphi)$

which entails a functor $\mathsf{Sen}_{\boldsymbol{S}} : \mathsf{Sign}_{\boldsymbol{S}} \to \mathsf{Set}$ mapping each signature to the set of its sentences, and each signature morphism to the corresponding translation of sentences.

Lemma 8. $\mathsf{Sen}_{\boldsymbol{S}} : \mathsf{Sign}_{\boldsymbol{S}} \to \mathsf{Set}$ *is a functor.*

Proof. As proven before, Lemma 4, $\mathsf{Sign}_{\boldsymbol{S}}$ is a category. $\mathsf{Sen}_{\boldsymbol{S}}$ is a functor between category $\mathsf{Sign}_{\boldsymbol{S}}$ and Set such that for any signature $\Sigma \in |\mathsf{Sign}_{\boldsymbol{S}}|$, $\mathsf{Sen}_{\boldsymbol{S}}(\Sigma)$ stands for the set of all Σ-sentences. For any signature morphism from Σ to Σ' function:

$$\mathsf{Sen}_{\boldsymbol{S}} : \mathsf{Sign}_{\boldsymbol{S}}(\Sigma, \Sigma') \to \mathsf{Sen}_{\boldsymbol{S}}(\mathsf{Sen}_{\boldsymbol{S}}(\Sigma), \mathsf{Sen}_{\boldsymbol{S}}(\Sigma'))$$

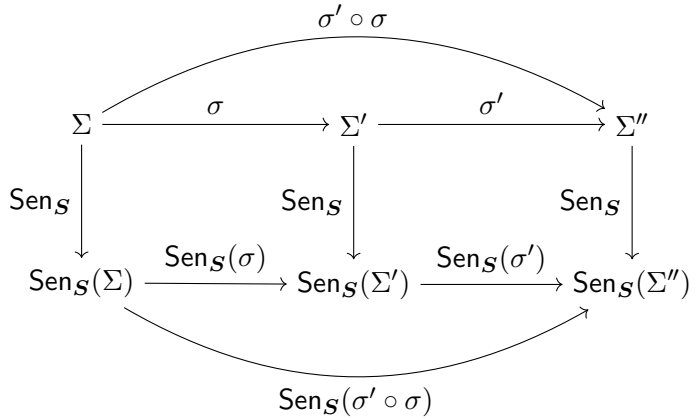
maps that morphism to another that translates sentences in $\mathsf{Sen}_{\boldsymbol{S}}(\Sigma)$ to sentences in $\mathsf{Sen}_{\boldsymbol{S}}(\Sigma')$. Given the identity morphisms $id_\Sigma : \Sigma \to \Sigma$ and $id_{\mathsf{Sen}_{\boldsymbol{S}}(\Sigma)} : \mathsf{Sen}_{\boldsymbol{S}}(\Sigma) \to \mathsf{Sen}_{\boldsymbol{S}}(\Sigma)$ then functor $\mathsf{Sen}_{\boldsymbol{S}}$ preserves identities:

$$\mathsf{Sen}_{\boldsymbol{S}}(id_\Sigma) = id_{\mathsf{Sen}_{\boldsymbol{S}}(\Sigma)}$$

We are left to prove that $\mathsf{Sen}_{\boldsymbol{S}}$ preserves composition, that is, given signature morphisms $\sigma : \Sigma \to \Sigma'$ and $\sigma' : \Sigma' \to \Sigma''$ between signatures in $|\mathsf{Sign}_{\boldsymbol{S}}|$ then,

$$\mathsf{Sen}_{\boldsymbol{S}}(\sigma' \circ \sigma) = \mathsf{Sen}_{\boldsymbol{S}}(\sigma') \circ \mathsf{Sen}_{\boldsymbol{S}}(\sigma)$$

Composition of $\mathsf{Sen}_{\boldsymbol{S}}$ is illustrated by the following diagram:



The proof is done by induction over the structure of sentences. Cases $\bot$ and p (operation in Σ) are trivial.

$$\mathsf{Sen}_{\boldsymbol{S}}(\sigma' \circ \sigma)(\bot) = \bot = \mathsf{Sen}_{\boldsymbol{S}}(\sigma')(\bot) =$$
$$\mathsf{Sen}_{\boldsymbol{S}}(\sigma')(\mathsf{Sen}_{\boldsymbol{S}}(\sigma)(\bot)) = (\mathsf{Sen}_{\boldsymbol{S}}(\sigma') \circ \mathsf{Sen}_{\boldsymbol{S}}(\sigma))(\bot)$$

and,

$$\mathsf{Sen}_{\boldsymbol{S}}(\sigma' \circ \sigma)(p) = (\sigma' \circ \sigma)(p) = \sigma'(\sigma(p)) = \mathsf{Sen}_{\boldsymbol{S}}(\sigma')(\sigma(p)) =$$
$$\mathsf{Sen}_{\boldsymbol{S}}(\sigma')(\mathsf{Sen}_{\boldsymbol{S}}(\sigma)(p)) = (\mathsf{Sen}_{\boldsymbol{S}}(\sigma') \circ \mathsf{Sen}_{\boldsymbol{S}}(\sigma))(p)$$

We present the proof for connective $\to$ the ones for connectives $\neg$, $\wedge$, $\vee$ and $\circ$ are similar.

$$\begin{aligned}
\mathsf{Sen}_{\boldsymbol{S}}(\sigma' \circ \sigma)(\varphi \to \varphi') =& \{\text{def. of } \mathsf{Sen}_{\boldsymbol{S}}\} \\
& \mathsf{Sen}_{\boldsymbol{S}}(\sigma' \circ \sigma)(\varphi) \to \mathsf{Sen}_{\boldsymbol{S}}(\sigma' \circ \sigma)(\varphi') \\
=& \{\text{induction hypothesis}\} \\
& (\mathsf{Sen}_{\boldsymbol{S}}(\sigma') \circ \mathsf{Sen}_{\boldsymbol{S}}(\sigma))(\varphi) \to (\mathsf{Sen}_{\boldsymbol{S}}(\sigma') \circ \mathsf{Sen}_{\boldsymbol{S}}(\sigma))(\varphi') \\
=& \{\text{def. of } \mathsf{Sen}_{\boldsymbol{S}}\} \\
& (\mathsf{Sen}_{\boldsymbol{S}}(\sigma') \circ \mathsf{Sen}_{\boldsymbol{S}}(\sigma))(\varphi \to \varphi')
\end{aligned}$$

For connective $[a]\varphi$ the proof is as follows, it is similar for the connectives $\langle a\rangle\varphi$, $[\not a]\varphi$, $\langle\not a\rangle\varphi$.

$$\begin{aligned}
\mathsf{Sen}_{\boldsymbol{S}}(\sigma' \circ \sigma)([a]\varphi) =& \{\text{def. of } \mathsf{Sen}_{\boldsymbol{S}}\} \\
& [(\sigma' \circ \sigma)(a)]\, \mathsf{Sen}_{\boldsymbol{S}}(\sigma' \circ \sigma)(\varphi) \\
& \{\text{induction hypothesis}\} \\
& [\sigma'(\sigma(a))]\ (\mathsf{Sen}_{\boldsymbol{S}}(\sigma') \circ \mathsf{Sen}_{\boldsymbol{S}}(\sigma))(\varphi) \\
=& \{\text{def. of } \mathsf{Sen}_{\boldsymbol{S}}\} \\
& \mathsf{Sen}_{\boldsymbol{S}}(\sigma')\ ([\sigma(a)]\, \mathsf{Sen}_{\boldsymbol{S}}(\sigma)(\varphi)) \\
=& \{\text{def. of } \mathsf{Sen}_{\boldsymbol{S}}\} \\
& \mathsf{Sen}_{\boldsymbol{S}}(\sigma')\ (\mathsf{Sen}_{\boldsymbol{S}}(\sigma)([a]\varphi)) = (\mathsf{Sen}_{\boldsymbol{S}}(\sigma') \circ \mathsf{Sen}_{\boldsymbol{S}}(\sigma))\ ([a]\varphi)
\end{aligned}$$

□

3.5 The satisfaction relation

Definition 26. *Given a signature* $\Sigma = \langle \mathrm{Act}, \mathrm{Prop}\rangle \in |\mathsf{Sign}_{\boldsymbol{S}}|$*, and a* Σ*-PTS model* $M = (W, R, V)$*, the satisfaction relation*

$$\models_{\boldsymbol{S}} :\ \mathsf{Mod}_{\boldsymbol{S}}(\Sigma) \times \mathsf{Sen}_{\boldsymbol{S}}(\Sigma) \to A \times A$$

is defined by

$$(M \models_{\boldsymbol{S}} \varphi) = \prod_{w \in W} (M, w \models_{\boldsymbol{S}} \varphi)$$

Relation $\models_{\boldsymbol{S}}$ *is recursively defined as follows*

- $(M, w \models_{\boldsymbol{S}} p) = V(w, p)$
- $(M, w \models_{\boldsymbol{S}} \bot) = (0, 1)$
- $(M, w \models_{\boldsymbol{S}} \neg\varphi) = /\!/(M, w \models_{\boldsymbol{S}} \varphi)$
- $(M, w \models_{\boldsymbol{S}} \varphi \to \varphi') = (M, w \models_{\boldsymbol{S}} \varphi) \Rightarrow (M, w \models_{\boldsymbol{S}} \varphi')$
- $(M, w \models_{\boldsymbol{S}} \varphi \vee \varphi') = (M, w \models_{\boldsymbol{S}} \varphi) \amalg (M, w \models_{\boldsymbol{S}} \varphi')$
- $(M, w \models_{\boldsymbol{S}} \varphi \wedge \varphi') = (M, w \models_{\boldsymbol{S}} \varphi) \prod (M, w \models_{\boldsymbol{S}} \varphi')$
- $(M, w \models_{\boldsymbol{S}} [a]\varphi) = (\, [a^+](M, w, \varphi^+), \langle a^+\rangle(M, w, \varphi^-)\,)$
- $(M, w \models_{\boldsymbol{S}} \langle a\rangle\varphi) = (\, \langle a^+\rangle(M, w, \varphi^+), [a^+](M, w, \varphi^-)\,)$
- $(M, w \models_{\boldsymbol{S}} [\not a]\varphi) = (\, \langle a^-\rangle(M, w, \varphi^-), [a^-](M, w, \varphi^+)\,)$
- $(M, w \models_{\boldsymbol{S}} \langle\not a\rangle\varphi) = (\, [a^-](M, w, \varphi^-), \langle a^-\rangle(M, w, \varphi^+)\,)$
- $(M, w \models_{\boldsymbol{S}} \circ\varphi) = \begin{cases} (1,0) & if\ (M, w \models_{\boldsymbol{S}} \varphi) \in \Delta_C \\ (0,1) & otherwise \end{cases}$

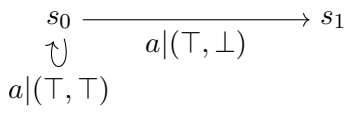
where

- $[a^+](M,w,\varphi^*) = \bigsqcap_{w' \in W} (R_a^+(w,w') \rightharpoonup (M,w' \models_{\boldsymbol{S}} \varphi)^*)$
- $[a^-](M,w,\varphi^*) = \bigsqcap_{w' \in W} (R_a^-(w,w') \rightharpoonup (M,w' \models_{\boldsymbol{S}} \varphi)^*)$
- $\langle a^+\rangle(M,w,\varphi^*) = \bigsqcup_{w' \in W} (R_a^+(w,w') \sqcap (M,w' \models_{\boldsymbol{S}} \varphi)^*)$
- $\langle a^-\rangle(M,w,\varphi^*) = \bigsqcup_{w' \in W} (R_a^-(w,w') \sqcap (M,w' \models_{\boldsymbol{S}} \varphi)^*)$
- $\Delta_C = \{(a,b) \mid D((a,b),(0,0)) \leqslant D((a,b),(1,1))\}$

with $* \in \{^+, ^-\}$

Hence φ is valid if, for any $w \in W$, $(M, w \models_{\boldsymbol{S}} \varphi) = (1,0)$.

Example 15. *Consider* ***2*** *the underlying iMTL-algebra, a signature* $\langle\{a\},\{p,q\}\rangle$ *and a paraconsistent transition system* $M = (\{s_0,s_1\},R,V)$ *depicted in the figure bellow:*

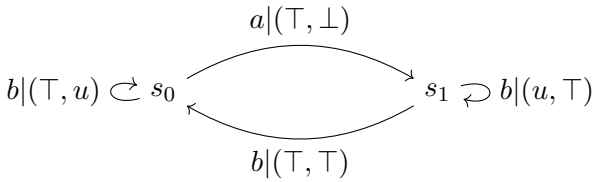


where $V(s_0,p) = (\top,\bot)$, $V(s_0,q) = (\bot,\top)$, $V(s_1,p) = (\top,\top)$ *and* $V(s_1,q) = (\bot,\bot)$.

$$
\begin{aligned}
M, s_0 \models_{\boldsymbol{S}} \not\langle a\rangle p \vee q =& (M, s_0 \models_{\boldsymbol{S}} \not\langle a\rangle p) \sqcup (M, s_0 \models_{\boldsymbol{S}} q) \\
=& ([a^-](M,s_0,p^-), \langle a^-\rangle(M,s_0,p^+)) \sqcup V(s_0,q) \\
=& \Big((R_a^-(s_0,s_0) \rightharpoonup (M,s_0 \models p)^-) \sqcap (R_a^-(s_0,s_1) \rightharpoonup (M,s_1 \models p)^-), \\
& \quad (R_a^-(s_0,s_0) \sqcap (M,s_0 \models p)^+) \sqcup (R_a^-(s_0,s_1) \sqcap (M,s_1 \models p)^+) \Big) \sqcup (\bot,\top) \\
=& \Big((\top \rightharpoonup \bot) \sqcap (\bot \rightharpoonup \top), (\top \sqcap \top) \sqcup (\bot \sqcap \top) \Big) \sqcup (\bot,\top) \\
=& (\bot \sqcap \top, \top \sqcup \bot) \sqcup (\bot,\top) = (\bot,\top) \sqcup (\bot,\top) = (\bot \sqcup \bot, \top \sqcap \top) \\
=& (\bot,\top)
\end{aligned}
$$

At state s_0 *the sentence* $\not\langle a\rangle p \vee q$ *holds with evidence degree* $\bot$ *and does not hold with evidence degree* $\top$.

Example 16. *Consider* ***3*** *the underlying iMTL-algebra and the following* $\langle\{a,b\},\{p,q,r\}\rangle$ *paraconsistent transition* $M = (\{s_0,s_1\},R,V)$ *depicted in the figure bellow. Where* $V(s_0,p) = (\top,\top), V(s_0,q) = V(s_1,p) = (\bot,u)$, $V(s_0,r) = V(s_1,r) = (u,u)$, $V(s_1,q) = (\bot,\bot)$



$$\begin{aligned}(M, s_0 \models_{\boldsymbol{S}} r \to (p \vee q)) =& (M, s_0 \models_{\boldsymbol{S}} r) \Rightarrow (M, s_0 \models_{\boldsymbol{S}} (p \vee q)) \\ =& V(s_0, r) \Rightarrow ((M, s_0 \models_{\boldsymbol{S}} p) \amalg (M, s_0 \models_{\boldsymbol{S}} q)) \\ =& V(s_0, r) \Rightarrow (V(s_0, p) \amalg V(s_0, q)) \\ =& (u, u) \Rightarrow ((\top, \top) \amalg (\bot, u)) \\ =& (u, u) \Rightarrow (\top \sqcup \bot, \top \sqcap u) \\ =& (u, u) \Rightarrow (\top, u) \\ =& (u \rightharpoonup \top, u \sqcap u) = (\top, u)\end{aligned}$$

At state s_0 the sentence $r \to (p \vee q)$ has a evidence degree $\top$ of holding and the evidence degree in which it does not hold is u.

Notice that,

$$\begin{aligned}\langle b^+ \rangle (M, s_1, p^+) =& \bigsqcup_{s \in R_b[s_1]} (R_b^+(s_1, s) \sqcap (M, s \models_{\boldsymbol{S}} p)^-) \\ =& (R_b^+(s_1, s_1) \sqcap (M, s_1 \models_{\boldsymbol{S}} p)^-) \sqcup (R_b^+(s_1, s_0) \sqcap (M, s_0 \models_{\boldsymbol{S}} p)^-) \\ =& \Big((u, \top)^+ \sqcap (\top, u)^-\Big) \sqcup \Big((\top, \top)^+ \sqcap (\top, \top)^-\Big) \\ =& (u \sqcap u) \sqcup (\top \sqcap \top) = \top\end{aligned}$$

$$\begin{aligned}[b^+](M, s_1, p^-) =& \bigsqcap_{s \in R_b[s_1]} (R_b^+(s_1, s) \rightharpoonup (M, s \models_{\boldsymbol{S}} p)^-) \\ =& (R_b^+(s_1, s_1) \rightharpoonup (M, s_1 \models_{\boldsymbol{S}} p)^-) \sqcap (R_b^+(s_1, s_0) \rightharpoonup (M, s_0 \models_{\boldsymbol{S}} p)^-) \\ =& (u \rightharpoonup u) \sqcap (\top \rightharpoonup \top) = \top\end{aligned}$$

Therefore,

$$(M, s_1 \models_{\boldsymbol{S}} \langle b \rangle p) = (\langle b^+ \rangle (M, s_1, p^+)\,,\, [b^+](M, s_1, p^-)) = (\top, \top)$$

That is, in state s_1 the sentence $\langle b \rangle p$ has evidence degree $\top$ of holding and evidence degree $\top$ of not holding.

Theorem 4. *Let $\Sigma = \langle \mathrm{Act}, \mathrm{Prop} \rangle$ and $\Sigma' = \langle \mathrm{Act}', \mathrm{Prop}' \rangle$ be signatures in $\mathsf{Sign}_{\boldsymbol{S}}$ and $\sigma : \Sigma \to \Sigma'$ be a signature morphism. For any model $M' \in \mathsf{Mod}_{\boldsymbol{S}}(\Sigma')$ and formula $\varphi \in \mathsf{Sen}_{\boldsymbol{S}}(\Sigma)$,*

$$\big(M'|_\sigma \models_{\boldsymbol{S}} \varphi\big) \;=\; \big(M' \models_{\boldsymbol{S}} \mathsf{Sen}_{\boldsymbol{S}}(\sigma)(\varphi)\big) \tag{3.4}$$

Proof. According to the definition of $\models_{\boldsymbol{S}}$ it is enough to prove that for any $w \in W$,

$$\big(M'|_\sigma, w \models_{\boldsymbol{S}} \varphi\big) \;=\; \big(M', w \models_{\boldsymbol{S}} \mathsf{Sen}_{\boldsymbol{S}}(\sigma)(\varphi)\big)$$

Notice that w is in the model $M'|_\sigma$ and M' by the definition of a σ-reduct.
The proof is by induction over the structure of sentences. To simplify notation we will write $\sigma(p)$ instead of $\sigma_{\mathrm{Prop}}(p)$ for any $p \in \mathrm{Prop}$ and $\sigma(a)$ instead of $\sigma_{\mathrm{Act}}(a)$ for any $a \in \mathrm{Act}$, as well as omit the subscript $\boldsymbol{S}$.
The case of $\bot$ is trivial, by the definition of $\models$ and Sen we have that

$$(M'|_\sigma, w \models \bot) = (M', w \models \mathsf{Sen}(\sigma)(\bot) = \bot) = (0, 1)$$

Given a proposition $p \in \mathrm{Prop}$ observe that

$$\begin{aligned}
M', w \models \mathsf{Sen}(\sigma)(p) =& \{\text{definition of } \mathsf{Sen}\} \\
& M', w \models \sigma(p) \\
=& \{\text{definition of } \models\} \\
& V'(w, \sigma(p)) \\
=& \{\text{definition of } \sigma\text{-reduct}\} \\
& V(w, p) \\
=& \{\text{definition of } \models\} \\
& M'|_\sigma, w \models p
\end{aligned}$$

Given any sentences φ and φ' and any $a \in \mathrm{Act}$ we have

$$\begin{aligned}
M', w \models \mathsf{Sen}(\sigma)(\neg\varphi) =& \{\text{definition of } \mathsf{Sen}\} \\
& M', w \models \neg\mathsf{Sen}(\sigma)(\varphi) \\
=& \{\text{definition of } \models\} \\
& /\!\!/ (M', w \models \mathsf{Sen}(\sigma)(\varphi)) \\
=& \{\text{induction hypothesis}\} \\
& /\!\!/ (M'|_\sigma, w \models \varphi) \\
=& \{\text{definition of } \models\} \\
& M'|_\sigma, w \models \neg\varphi
\end{aligned}$$

$$\begin{aligned}
M', w \models \mathsf{Sen}(\sigma)(\varphi \wedge \varphi') =& \{\text{definition of } \mathsf{Sen}\} \\
& M', w \models \mathsf{Sen}(\sigma)(\varphi) \wedge \mathsf{Sen}(\sigma)(\varphi') \\
=& \{\text{definition of } \models\} \\
& (M', w \models \mathsf{Sen}(\sigma)(\varphi)) \sqcap (M', w \models \mathsf{Sen}(\sigma)(\varphi')) \\
=& \{\text{induction hypothesis}\} \\
& (M'|_\sigma, w \models \ \varphi) \sqcap (M'|_\sigma, w \models \varphi') \\
=& \{\text{definition of } \models\} \\
& M'|_\sigma, w \models (\varphi \wedge \varphi')
\end{aligned}$$

$$\begin{aligned}
M', w \models \mathsf{Sen}(\sigma)(\varphi \vee \varphi') =& \{\text{definition of } \mathsf{Sen}\} \\
& (M', w \models \mathsf{Sen}(\sigma)(\varphi) \vee \mathsf{Sen}(\sigma)(\varphi')) \\
=& \{\text{definition of } \models\} \\
& ((M', w \models \mathsf{Sen}(\sigma)(\varphi)) \sqcup (M', w \models \mathsf{Sen}(\sigma)(\varphi'))) \\
=& \{\text{induction hypothesis}\} \\
& ((M'|_\sigma, w \models \ \varphi) \sqcup (M'|_\sigma, w \models \varphi')) \\
=& \{\text{definition of } \models\} \\
& M'|_\sigma, w \models (\varphi \vee \varphi')
\end{aligned}$$

$$
\begin{aligned}
M', w \models \mathsf{Sen}(\sigma)(\varphi \to \varphi') =& \{\text{definition of } \mathsf{Sen}\} \\
& M', w \models \mathsf{Sen}(\sigma)(\varphi) \to \mathsf{Sen}(\sigma)(\varphi') \\
=& \{\text{definition of } \models\} \\
& (M', w \models \mathsf{Sen}(\sigma)(\varphi)) \Rightarrow (M', w \models \mathsf{Sen}(\sigma)(\varphi')) \\
=& \{\text{induction hypothesis}\} \\
& (M'|_\sigma, w \models \ \varphi) \Rightarrow (M'|_\sigma, w \models \varphi') \\
=& \{\text{definition of } \models\} \\
& M'|_\sigma, w \models (\varphi \to \varphi')
\end{aligned}
$$

For case $[a]\varphi$ the proof follows:

$$
\begin{aligned}
& M', w \models \mathsf{Sen}(\sigma)([a]\,\varphi) \\
=& \{\text{definition of } \mathsf{Sen}\} \\
& M', w \models [\sigma(a)]\,\mathsf{Sen}(\sigma)(\varphi) \\
=& \{\text{definition of } \models\} \\
& ([\sigma(a)^+](M', w, \mathsf{Sen}(\sigma)(\varphi)^+)\,, \langle\sigma(a)^+\rangle(M', w, \mathsf{Sen}(\sigma)(\varphi)^-)) \\
=& \{\text{def. of } [a^+] \text{ and } \langle a^+\rangle\} \\
& \left(\prod_{w' \in W'} (R'^+_{\sigma(a)}(w, w') \rightharpoonup (M', w' \models \mathsf{Sen}(\sigma)(\varphi))^+),\ \bigsqcup_{w' \in W'} (R'^+_{\sigma(a)}(w, w') \sqcap (M', w' \models \mathsf{Sen}(\sigma)(\varphi))^-)\right) \\
=& \{(\text{step } \star)\} \\
& \left(\prod_{w' \in W} (R^+_a(w, w') \rightharpoonup (M'|_\sigma, w \models \varphi)^+),\ \bigsqcup_{w' \in W} (R^+_a(w, w') \sqcap (M'|_\sigma, w \models \varphi)^-)\right) \\
=& \{\text{def. } [a^+] \text{ and } \langle a^+\rangle\} \\
& ([a^+](M'|_\sigma, w, \varphi^+)\,, \langle a^+\rangle(M'|_\sigma, w, \varphi^-)) \\
=& \{\text{definition of } \models\} \\
& M'|_\sigma, w \models [a]\,\varphi
\end{aligned}
$$

(step $\star$) By the definition of σ-reduct we have that $W = W'$ and for any $w, w' \in W$ $R'_{\sigma(a)}(w, w') = R_a(w, w') = (\mathit{tt}, \mathit{ff})$ for some $\mathit{tt}, \mathit{ff} \in A$, that is, there is a transition in M' between state w and w' through action $\sigma(a)$ (equivalent to writing $(w, w', \mathit{tt}, \mathit{ff}) \in R'_{\sigma(a)}$) and there is a transition in $M'|_\sigma$ between state w and w' through action a (equivalent to writing $(w, w', \mathit{tt}, \mathit{ff}) \in R_a$). Also, by the induction hypothesis, $(M', w \models \mathsf{Sen}(\sigma)(\varphi)) = (M'|_\sigma, w \models \varphi)$, which is equivalent to

$$
\begin{aligned}
& \left((M', w \models \mathsf{Sen}(\sigma)(\varphi))^+, (M', w \models \mathsf{Sen}(\sigma)(\varphi))^-\right) \\
& = \left((M'|_\sigma, w \models \varphi)^+, (M'|_\sigma, w \models \varphi)^-\right)
\end{aligned}
\tag{3.5}
$$

Therefore, by (3.5), we have that $(M', w \models \mathsf{Sen}(\sigma)(\varphi))^+ = (M'|_\sigma, w \models \varphi)^+$ as well as $(M', w \models \mathsf{Sen}(\sigma)(\varphi))^- = (M'|_\sigma, w \models \varphi)^-$.

For case $\langle a \rangle \varphi$ the proof is:

$$
\begin{aligned}
& M', w \models \mathsf{Sen}(\sigma)(\langle a \rangle \varphi) \\
=& \{\text{definition of } \mathsf{Sen}\} \\
& M', w \models \langle \sigma(a) \rangle \mathsf{Sen}(\sigma)(\varphi) \\
=& \{\text{definition of } \models\} \\
& (\langle \sigma(a)^+ \rangle (M', w, \mathsf{Sen}(\sigma)(\varphi)^+), [\sigma(a)^+](M', w, \mathsf{Sen}(\sigma)(\varphi)^-)) \\
=& \{\text{def. of } [a^+] \text{ and } \langle a^+ \rangle\} \\
& \left(\bigsqcup_{w' \in W'} (R'^{+}_{\sigma(a)}(w, w') \sqcap (M', w' \models \mathsf{Sen}(\sigma)(\varphi))^+), \prod_{w' \in W'} (R'^{+}_{\sigma(a)}(w, w') \rightharpoonup (M', w' \models \mathsf{Sen}(\sigma)(\varphi))^-) \right) \\
=& \{\text{analogous to (step } \star)\} \\
& \left(\bigsqcup_{w' \in W} (R^+_a(w, w') \sqcap (M'|_\sigma, w' \models \varphi)^+), \prod_{w' \in W} (R^+_a(w, w') \rightharpoonup (M'|_\sigma, w' \models \varphi)^-) \right) \\
=& \{\text{def. of } [a^+] \text{ and } \langle a^+ \rangle\} \\
& (\langle a^+ \rangle (M'|_\sigma, w, \varphi^+)\,,\, [a^+](M'|_\sigma, w, \varphi^-)) \\
=& \{\text{definition of } \models\} \\
& M'|_\sigma, w \models \langle a \rangle \varphi
\end{aligned}
$$

For case $\cancel{[a]}\varphi$ the proof follows:

$$
\begin{aligned}
& M', w \models \mathsf{Sen}(\sigma)(\cancel{[a]}\, \varphi) \\
=& \{\text{definition of } \mathsf{Sen}\} \\
& M', w \models \cancel{[\sigma(a)]}\, \mathsf{Sen}(\sigma)(\varphi) \\
=& \{\text{definition of } \models\} \\
& (\langle \sigma(a)^- \rangle (M', w, \mathsf{Sen}(\sigma)(\varphi)^-), [\sigma(a)^-](M', w, \mathsf{Sen}(\sigma)(\varphi)^+)) \\
=& \{\text{def. of } [a^-] \text{ and } \langle a^- \rangle\} \\
& \left(\bigsqcup_{w' \in W'} (R'^{-}_{\sigma(a)}(w, w') \sqcap (M', w' \models \mathsf{Sen}(\sigma)(\varphi))^-), \prod_{w' \in W'} (R'^{-}_{\sigma(a)}(w, w') \rightharpoonup (M', w' \models \mathsf{Sen}(\sigma)(\varphi))^+) \right) \\
=& \{\text{analogous to (step } \star)\} \\
& (\langle a^- \rangle (M'|_\sigma, w, \varphi^-)\,,\, [a^-](M'|_\sigma, w, \varphi^+)) \\
=& \{\text{definition of } \models\} \\
& M'|_\sigma, w \models \cancel{[a]}\, \varphi
\end{aligned}
$$

For case $\cancel{\langle a \rangle}\varphi$ the proof is:

$$
\begin{aligned}
& M', w \models \mathsf{Sen}(\sigma)(\cancel{\langle a \rangle}\varphi) \\
=& \{\text{definition of } \mathsf{Sen}\} \\
& M', w \models \cancel{\langle \sigma(a) \rangle}\, \mathsf{Sen}(\sigma)(\varphi) \\
=& \{\text{definition of } \models\} \\
& ([\sigma(a)^-](M', w, \mathsf{Sen}(\sigma)(\varphi)^-)\,,\, \langle \sigma(a)^- \rangle (M', w, \mathsf{Sen}(\sigma)(\varphi)^+)) \\
=& \{\text{def. of } [a^-] \text{ and } \langle a^- \rangle\}
\end{aligned}
$$

$$\Big(\bigsqcap_{w'\in W'}(R'^{-}_{\sigma(a)}(w,w') \rightharpoonup (M',w' \models \mathsf{Sen}(\sigma)(\varphi))^{-}),\ \bigsqcup_{w'\in W'}(R'^{-}_{\sigma(a)}(w,w') \sqcap (M',w' \models \mathsf{Sen}(\sigma)(\varphi))^{+})\Big)$$

$=$ {analogous to (step $\star$)}

$([a^{-}](M'|_{\sigma},w,\varphi^{-})\,,\langle a^{-}\rangle(M'|_{\sigma},w,\varphi^{+}))$

$=$ {definition of $\models$}

$M'|_{\sigma},w \models \not{\langle a\rangle}\varphi$ |

Finally, for case $\circ\varphi$ the proof is:

$M',w \models \mathsf{Sen}(\sigma)(\circ\,\varphi)$

$=$ {definition of Sen}

$M',w \models \circ\,\mathsf{Sen}(\sigma)(\varphi)$

$=$ {definition of $\models$}

$$\begin{cases}(1,0) & if\ (M',w \models \mathsf{Sen}(\sigma)(\varphi)) \in \Delta_C\\ (0,1) & otherwise\end{cases}$$

$=$ {induction hypothesis}

$$\begin{cases}(1,0) & if\ (M'|_{\sigma},w \models \varphi) \in \Delta_C\\ (0,1) & otherwise\end{cases}$$

$=$ {def. of $\models$}

$M'|_{\sigma},w \models \circ\,\varphi$

□

The next Theorem is a consequence of this entire chapter where every ingredient of institution $\boldsymbol{S}(\mathscr{A})$ is properly formalized in terms of category theory and the satisfaction condition, for many-valued institutions, is provided.

Theorem 5. *$\boldsymbol{S}(\mathscr{A}) = (\mathsf{Sign}_{\boldsymbol{S}(\mathscr{A})},\mathsf{Sen}_{\boldsymbol{S}(\mathscr{A})},\mathsf{Mod}_{\boldsymbol{S}(\mathscr{A})},\models_{\boldsymbol{S}(\mathscr{A})})$ is an institution, for a fixed twisted algebra $\mathscr{A}$.*

Hence, we framed $\boldsymbol{S}$ as an institution. Such abstraction is necessary to avoid the specific syntax of the logic and focus on building larger specifications in Chapter 5.

Chapter 4

An institution for paraconsistent processes

In this chapter we explore a logic for paraconsistent processes, similar to the one discussed in the previous chapter, with the addition of binders and operator @, modalities indexed by regular expressions and satisfaction of sentences being evaluated at an initial state rather than globally.

The binder and @ operators are taken from hybrid logic, that was first introduced by the work of Arthur Prior in the 1960s. The binder operator is an important tool to express recursive patterns (e.g. loop transitions) and to name states that can be later referred to with the operator @ thus, allowing evaluation at a specific state in a model. We will, however, omit nominals, also common in hybrid logic [Bra10], which are propositional symbols of a new sort such that in a model each nominal is true at exactly one state. The only consequence of not having nominals in our logic is that we can not refer to states that have not been previously visited. This approach is similar to the one taken in [MBHM18].
Hybrid logic allows us to express some abstract requirements as, for example,

> ***Reflexivity*:** $\downarrow x.\langle a\rangle x$ reads as "there is a loop transition of action a in the current state"
>
> ***Symmetry:*** $\downarrow x.\langle a\rangle \downarrow y \rightharpoonup @_y\langle a\rangle x$ reads as "at the current state x if an action a occurs we reach state y then at state y an action a can occur to state x"
>
> ***Deterministic transition:*** $\downarrow x.\langle a\rangle\top \rightharpoonup (\langle a\rangle \downarrow y.\, @_x\,[a]\, y)$ reads as "if action a is possible to occur at the current state x then state y is reached and no other"

We take from dynamic logic the notion of regular modalities, i.e. modalities are now indexed by regular expressions of actions rather than by atomic actions only. This enables us to express abstract proprieties [MBHM18] such as:

> ***Liveness:*** $[\mathrm{Act}^*; a]\langle \mathrm{Act}^*; b\rangle \top$ reads "after the occurrence of an action a, an action b can be eventually realised"
>
> ***Liveness:*** $[\mathrm{Act}^*; a; (-b)^*]\langle \mathrm{Act}^*; b\rangle \top$ reads "after the occurrence of an action a, an occurrence of an action b is eventually possible if it has not occurred before" with $-b$ standing for the set $\mathrm{Act}\backslash\{b\}$.

Deadlock avoidance: $[\text{Act}^*]\langle\text{Act}\rangle\top$ reads "after the occurrence of any action, it is possible to read another action"

This combination of hybrid and dynamic logic to the paraconsistent modal logic of Chapter 3 results in a more expressive logic, able to capture abstract and complex requirements like the ones mentioned above.

Now that we have discussed the main differences with respect to the logic formalized in the previous chapter, let us revisit each component of $\boldsymbol{S}(\mathscr{A})$ institution and define a new institution $\boldsymbol{P}(\mathscr{A}) = (\mathsf{Sign}_{\boldsymbol{P}(\mathscr{A})}, \mathsf{Sen}_{\boldsymbol{P}(\mathscr{A})}, \mathsf{Mod}_{\boldsymbol{P}(\mathscr{A})}, \models_{\boldsymbol{P}(\mathscr{A})})$ for paraconsistent processes. As before, we will fix a twisted algebra $\mathscr{A}$, and abbreviate institution $\boldsymbol{P}(\mathscr{A})$, whenever $\mathscr{A}$ is not essential, to $\boldsymbol{P} = (\mathsf{Sign}_{\boldsymbol{P}}, \mathsf{Sen}_{\boldsymbol{P}}, \mathsf{Mod}_{\boldsymbol{P}}, \models_{\boldsymbol{P}})$.

4.1 The signatures

Definition 27. *Signatures are pairs* $\langle \text{Act}, \text{Prop}\rangle$ *where* Prop *and* Act *are sets of propositions and action symbols, respectively.*

That is, $\mathsf{Sign}_{\boldsymbol{P}} = \mathsf{Sign}_{\boldsymbol{S}}$ hence $\mathsf{Sign}_{\boldsymbol{P}}$ is a category, Lemma 4.
The set of actions Act will induce a set of structured actions, $Str(\text{Act})$, defined by:

$$\alpha := a \mid \alpha;\alpha \mid \alpha + \alpha \mid \alpha^*$$

where $a \in \text{Act}$.
Let $\Sigma, \Sigma' \in \mathsf{Sign}_{\boldsymbol{P}}$ be signatures. A signature morphism $\sigma : \Sigma \to \Sigma'$ involves two functions $\sigma_{\text{Prop}} : \text{Prop} \to \text{Prop}'$ and $\sigma_{\text{Act}} : \text{Act} \to \text{Act}'$, where σ_{Act} extends to $Str(\text{Act})$ as follows:

- $\widehat{\sigma}_{\text{Act}}(a) = \sigma_{\text{Act}}(a)$
- $\widehat{\sigma}_{\text{Act}}(\alpha;\alpha') = \widehat{\sigma}_{\text{Act}}(\alpha);\widehat{\sigma}_{\text{Act}}(\alpha')$
- $\widehat{\sigma}_{\text{Act}}(\alpha+\alpha') = \widehat{\sigma}_{\text{Act}}(\alpha) + \widehat{\sigma}_{\text{Act}}(\alpha')$
- $\widehat{\sigma}_{\text{Act}}(\alpha^*) = \widehat{\sigma}_{\text{Act}}(\alpha)^*$

$$\begin{array}{ccc} \text{Act} & \xrightarrow{\sigma_{\text{Act}}} & \text{Act}' \\ \downarrow & & \downarrow \\ Str(\text{Act}) & \xrightarrow{\widehat{\sigma}_{\text{Act}}} & Str(\text{Act}') \end{array}$$

for all $a \in \text{Act}$ and $\alpha, \alpha' \in Str(\text{Act})$.

4.2 The models

Definition 28. *For a signature* $\Sigma = \langle \text{Act}, \text{Prop}\rangle$*, a paraconsistent transition processes is a tuple* $M = (W, w_0, R, V)$*, where* $(W, R, V) \in \mathsf{Mod}_{\boldsymbol{S}}(\Sigma)$*, Definition 22, and* $w_0 \in W$ *is called the initial state of* M*.*

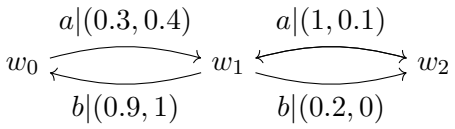
The interpretation of $\alpha \in Str(\text{Act})$ in a model $M = (W, w_0, R, V)$ extends the relation R to a relation $\widehat{R}$ defined for any states $w, w' \in W$ as:

- $\widehat{R}_a(w, w') = R_a(w, w')$ for $a \in \text{Act}$
- $\widehat{R}_{\alpha+\alpha'}(w, w') = \widehat{R}_\alpha(w, w') \sqcup \widehat{R}_{\alpha'}(w, w')$

- $\widehat{R}_{\alpha;\alpha'}(w,w') = \bigsqcup\!\!\bigsqcup_{v\in W}\left(\widehat{R}_\alpha(w,v) \sqcap \widehat{R}_{\alpha'}(v,w')\right)$
- $\widehat{R}_{\alpha *}(w,w') = \bigsqcup\!\!\bigsqcup_{i\geqslant 0} \widehat{R}^i_\alpha(w,w')$. Where,
  - $\widehat{R}^0_\alpha = \begin{cases}(1,0) & \text{if } w = w'\\ (0,1) & \text{otherwise}\end{cases}$
  - $\widehat{R}^{k+1}_\alpha(w,w') = (\widehat{R}^k_\alpha ; \widehat{R}_\alpha)(w,w')$

where $\bigsqcap\!\!\bigsqcap$ and $\bigsqcup\!\!\bigsqcup$ are the distributed versions of $\sqcap$ and $\sqcup$, respectively.

Example 17. *Consider $\boldsymbol{P}(\ddot{\boldsymbol{G}})$ and the signature $\Sigma = \langle\{a,b\},\{p\}\rangle$. The following model $M = (W, w_0, R, V)$ is a paraconsistent transition process over Σ with $V(w_0,q) = (0,1)$, $V(w_1,q) = (1,0.2)$ and $V(w_2,q) = (0.3,0.3)$.*



Notice that, $\widehat{R}_{a+b}(w_1,w_2) = R_a(w_1,w_2) \sqcup R_b(w_1,w_2) = (1,0.1) \sqcup (0.2,0) = (1,0)$
The evidence degree of going from state w_1 to state w_2 through an action a or an action b is 1 while the evidence degree of not being able to go from w_1 to w_2 (neither through action a neither through b) is 0. Note that in this case the pair of weights $(1,0)$ represents consistent information.
On the other hand,

$$\begin{aligned}\widehat{R}_{a;b}(w_1,w_1) =& \bigsqcup\!\!\bigsqcup_{v\in W}(R_a(w_1,v) \sqcap R_b(v,w_1)) = R_a(w_1,w_2) \sqcap R_b(w_2,w_1)\\ =&(1,0.1) \sqcap (0.2,0) = (0.2,0.1)\end{aligned}$$

$$\begin{aligned}\widehat{R}_{b;a}(w_1,w_1) =& \bigsqcup\!\!\bigsqcup_{v\in W}(R_b(w_1,v) \sqcap R_a(v,w_1)) = R_b(w_1,w_0) \sqcap R_a(w_0,w_1)\\ =&(0.9,1) \sqcap (0.3,0.4) = (0.3,1)\end{aligned}$$

Thus,

$$\widehat{R}_{(a;b)+(b;a)}(w_1,w_1) = \widehat{R}_{a;b}(w_1,w_1) \sqcup \widehat{R}_{b;a}(w_1,w_1) = (0.2,0.1) \sqcup (0.3,1) = (0.3,0.1)$$

The evidence degree from which transitioning from w_1 back to w_1 by action a followed by action b $(a;b)$ or by an action b followed by an action a $(b;a)$ may occur is 0.3 and the evidence degree of being prevented from occurring is 0.1. Note that $\widehat{R}_{(a;b)+(b;a)}(w_1,w_1)$ represents a scenario where the information is vague.

Definition 29. *A morphism connecting two paraconsistent processes $M = (W, w_0, R, V)$ and $M = (W', w'_0, R', V')$ is a function $h : W \to W'$ like the one defined for $\boldsymbol{S}$, Definition 23, plus the condition that $h(w_0) = w'_0$.*

Note that a morphism between two processes is a model morphism of two paraconsistent systems in institution $\boldsymbol{S}$ such that the initial state is preserved.

Lemma 9. *Models and their morphisms in institution* $\boldsymbol{P}$ *form a category.*

Proof. Morphisms between two paraconsistent processes can be seen as a function between sets of states, such that, identities exist and the composition of functions is associative, satisfying the category properties. □

Definition 30. *For any signature morphism* $\sigma : \langle \text{Act}, \text{Prop} \rangle \to \langle \text{Act}', \text{Prop}' \rangle$ *and* $M = (W', w_0', R', V')$ *a* $\langle \text{Act}', \text{Prop}' \rangle$*-process, the* σ*-reduct of* M' *is a* $\langle \text{Act}, \text{Prop} \rangle$*-process* $M|_\sigma = (W, w_0, R, V)$ *such that*

- $w_0 = w_0'$
- $W \backslash \{w_0\} = W' \backslash \{w_0'\}$
- *for* $p \in \text{Prop}$, $w \in W$, $V(w, p) = V'(w, \sigma(p))$
- *for* $w, v \in W$ *and* $a \in \text{Act}$, $R_a(w, v) = R'_{\sigma(a)}(w, v)$.

Lemma 10. *Reducts in institution* $\boldsymbol{P}$ *preserve morphisms.*

Proof. Let $\sigma : \langle \text{Act}, \text{Prop} \rangle \to \langle \text{Act}', \text{Prop}' \rangle$ be a signature morphism between signatures in $\mathsf{Sign}_{\boldsymbol{P}}$ and $h : W \to W'$ a morphism between a Σ-process $M = (W, w_0, R, V)$ and a Σ'-process $M' = (W', w_0', R', V')$.
The σ-reduct of M is the process $M|_\sigma = (W, w_0, R|_\sigma, V|_\sigma)$ and the σ-reduct of M' is the process $M'|_\sigma = (W', w_0', R'|_\sigma, V'|_\sigma)$.
By definition of σ-reduct, h is also a morphism between the states of $M|_\sigma$ and $M'|_\sigma$ and, by definition of morphism, this preserves the initial state, i.e. $h(w_0) = h(w_0')$. We are left to check that the morphism between reducts preserves the source valuation and the transition functions. Note that a process in $\boldsymbol{P}$ is a model in $\boldsymbol{S}$ (with one state being called the initial state), consequently the remaining of the proof follows exactly like the one done for institution $\boldsymbol{S}$ in Lemma 6. □

A functor $\mathsf{Mod}_{\boldsymbol{P}}$ is a mapping from each signature to the category of its processes and each signature morphism to its reduct functor.

Lemma 11. $\mathsf{Mod}_{\boldsymbol{P}} : (\mathsf{Sign}_{\boldsymbol{P}})^{op} \to \mathsf{CAT}$ *is a functor.*

Proof. As stated before, $\mathsf{Sign}_{\boldsymbol{P}}$ is a category and its opposite category is a category, Lemma 1. For each signature the category CAT of its models is a category, Lemma 9. $\mathsf{Mod}_{\boldsymbol{P}}$ is a functor between these two categories.
Let $\Sigma = \langle \text{Act}, \text{Prop} \rangle$ and $\Sigma' = \langle \text{Act}', \text{Prop}' \rangle$ in $|\mathsf{Sign}_{\boldsymbol{P}}|$ be signatures and $\sigma : \Sigma \to \Sigma'$ a signature morphism. $\mathsf{Mod}_{\boldsymbol{P}}(\Sigma)$ stands for the category of Σ-models and the function $\mathsf{Mod}_{\boldsymbol{P}}(\sigma) : \mathsf{Mod}_{\boldsymbol{P}}(\Sigma') \to \mathsf{Mod}_{\boldsymbol{P}}(\Sigma)$ maps a signature morphism to its reduct.

Consider the identity morphisms $id_\Sigma : \Sigma \to \Sigma$ and $id_{\mathsf{Mod}_{\boldsymbol{P}}(\Sigma)} : \mathsf{Mod}_{\boldsymbol{P}}(\Sigma) \to \mathsf{Mod}_{\boldsymbol{P}}(\Sigma)$. It is easy to check that $\mathsf{Mod}_{\boldsymbol{P}}$ preserves identities:

$$\mathsf{Mod}_{\boldsymbol{S}}(id_\Sigma) = id_{\mathsf{Mod}_{\boldsymbol{S}}(\Sigma)}$$

. Consider now a signature $\Sigma'' \in |\mathsf{Sign}_{\boldsymbol{P}}|$ and a signature morphism $\sigma' : \Sigma \to \Sigma''$. Let $M'' = (W'', w_0'', R'', V'')$ be a Σ''-process such that $M' = (W', w_0', R', V')$ is its σ'-reduct and $M = (W, w_0, R, V)$ is the σ-reduct of M'. Applying the definition of reduct, we have that for any $p \in \text{Prop}$, $a \in \text{Act}$ and any states $w, v \in W$

- $w_0 = w_0' = w_0''$
- $W \backslash \{w_0\} = W' \backslash \{w_0'\} = W'' \backslash \{w_0''\}$
- $V(w,p) = V'(w,\sigma(p)) = V''(w,\sigma'(\sigma(p))) = V''(w,(\sigma' \circ \sigma)(p))$
- $R_a(w,v) = R'_{\sigma(a)}(w,v) = R''_{\sigma'(\sigma(a))}(w,v) = R''_{(\sigma' \circ \sigma)(a)}(w,v)$

Thus M is the $(\sigma' \circ \sigma)$-reduct of M''. Moreover, for any Σ''-process M'',

$$\mathsf{Mod}_{\boldsymbol{S}}(\sigma' \circ \sigma)(M'') = (\mathsf{Mod}_{\boldsymbol{S}}(\sigma') \circ \mathsf{Mod}_{\boldsymbol{S}}(\sigma))(M'')$$

□

4.3 The sentences

Formulas are given by the following grammar:

$$\varphi := p \mid \bot \mid \neg\varphi \mid \varphi \to \varphi \mid \varphi \vee \varphi \mid \varphi \wedge \varphi \mid [\alpha]\,\varphi \mid \langle\alpha\rangle\,\varphi \mid \cancel{[\alpha]}\,\varphi \mid \cancel{\langle\alpha\rangle}\,\varphi \mid \circ\,\varphi \mid x \mid \downarrow x.\varphi \mid @_x\varphi$$

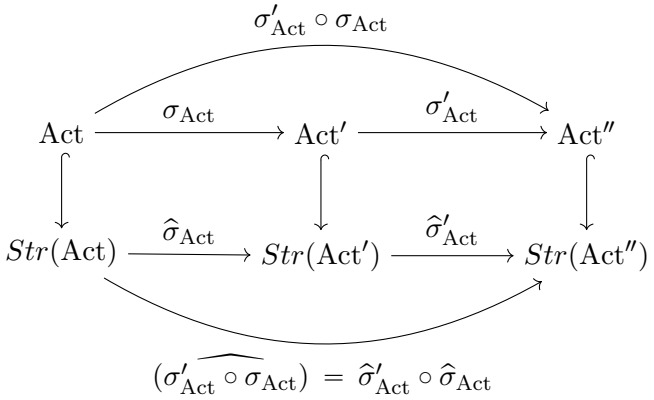
where $p \in \mathit{Prop}$, $\alpha \in \mathit{Str}(\mathrm{Act})$ and $x \in X$, with X being an infinite set of variables.
Sentence translation is the same as for $\boldsymbol{S}$ except for the modalities indexed by regular expressions, the addition of binders and operator @:

- $\mathsf{Sen}_{\boldsymbol{P}}(\sigma)([\alpha]\varphi) = [\widehat{\sigma}_{\mathrm{Act}}(\alpha)]\,\mathsf{Sen}_{\boldsymbol{P}}(\sigma)(\varphi)$
- $\mathsf{Sen}_{\boldsymbol{P}}(\sigma)(\langle\alpha\rangle\varphi) = \langle\widehat{\sigma}_{\mathrm{Act}}(\alpha)\rangle\,\mathsf{Sen}_{\boldsymbol{P}}(\sigma)(\varphi)$
- $\mathsf{Sen}_{\boldsymbol{P}}(\sigma)(\cancel{[\alpha]}\varphi) = \cancel{[\widehat{\sigma}_{\mathrm{Act}}(\alpha)]}\,\mathsf{Sen}_{\boldsymbol{P}}(\sigma)(\varphi)$
- $\mathsf{Sen}_{\boldsymbol{P}}(\sigma)(\cancel{\langle\alpha\rangle}\varphi) = \cancel{\langle\widehat{\sigma}_{\mathrm{Act}}(\alpha)\rangle}\,\mathsf{Sen}_{\boldsymbol{P}}(\sigma)(\varphi)$
- $\mathsf{Sen}_{\boldsymbol{P}}(\sigma)(x) = x$
- $\mathsf{Sen}_{\boldsymbol{P}}(\sigma)(\downarrow x.\varphi) = \downarrow x.\mathsf{Sen}_{\boldsymbol{P}}(\sigma)(\varphi)$
- $\mathsf{Sen}_{\boldsymbol{P}}(\sigma)(@_x\varphi) = @_x\mathsf{Sen}_{\boldsymbol{P}}(\sigma)(\varphi)$

which lifts a functor $\mathsf{Sen}_{\boldsymbol{P}}$ that maps each siganture to the set of its sentences, and each signature morphism to the corresponding translation of sentences.

Lemma 12. $\mathsf{Sen}_{\boldsymbol{P}} : \mathsf{Sign}_{\boldsymbol{P}} \to \mathsf{Set}$ *is a functor.*

Proof. As proven before, $\mathsf{Sign}_{\boldsymbol{P}}$ and Set are categories. $\mathsf{Sen}_{\boldsymbol{P}}$ is a functor between the two categories.
The proof that $\mathsf{Sen}_{\boldsymbol{P}}$ preservers identities and composition is equal to the one done for institution $\boldsymbol{S}$, Lemma 8, except for $[\alpha]\varphi$, $\langle\alpha\rangle\varphi$, $\cancel{[\alpha]}\varphi$ and $\cancel{\langle\alpha\rangle}\varphi$ that require minimal changes.
It's worth noticing that for any $\Sigma, \Sigma', \Sigma'' \in |\mathsf{Sign}_{\boldsymbol{P}}|$ and signature morphisms $\sigma : \Sigma \to \Sigma'$ and $\sigma' : \Sigma' \to \Sigma''$ we have the next commutative diagram:



For $[\alpha]\varphi$ the proof that functor $\mathsf{Sen}_{\boldsymbol{P}}$ preserves composition follows, with σ_{Act} abbreviated to σ to simplify notation.

$$
\begin{aligned}
\mathsf{Sen}_{\boldsymbol{P}}(\sigma' \circ \sigma)([\alpha]\varphi) =& \{\text{def. of } \mathsf{Sen}_{\boldsymbol{P}}\} \\
& [(\widehat{\sigma' \circ \sigma})(\alpha)]\, \mathsf{Sen}_{\boldsymbol{P}}(\sigma' \circ \sigma)(\varphi) \\
=& \{\text{induction hypothesis}\} \\
& [\widehat{\sigma}'(\widehat{\sigma}(\alpha))]\ (\mathsf{Sen}_{\boldsymbol{P}}(\sigma') \circ \mathsf{Sen}_{\boldsymbol{P}}(\sigma))(\varphi) \\
=& \{\text{def. of } \mathsf{Sen}_{\boldsymbol{P}}\} \\
& \mathsf{Sen}_{\boldsymbol{P}}(\sigma')\ ([\widehat{\sigma}(\alpha)]\, \mathsf{Sen}_{\boldsymbol{P}}(\sigma)(\varphi)) \\
=& \{\text{def. of } \mathsf{Sen}_{\boldsymbol{P}}\} \\
& \mathsf{Sen}_{\boldsymbol{P}}(\sigma')(\mathsf{Sen}_{\boldsymbol{P}}(\sigma)([\alpha]\varphi)) = (\mathsf{Sen}_{\boldsymbol{P}}(\sigma') \circ \mathsf{Sen}_{\boldsymbol{P}}(\sigma))([\alpha]\varphi)
\end{aligned}
$$

The proof for the remaining modal operators is similar.
The case of a variable x is trivial since $\mathsf{Sen}_{\boldsymbol{P}}(\sigma' \circ \sigma)(x) = x$ by the definition of $\mathsf{Sen}_{\boldsymbol{P}}$ and

$$(\mathsf{Sen}_{\boldsymbol{P}}(\sigma') \circ \mathsf{Sen}_{\boldsymbol{P}}(\sigma))(x) = \mathsf{Sen}_{\boldsymbol{P}}(\sigma')(\mathsf{Sen}_{\boldsymbol{P}}(\sigma)(x)) = \mathsf{Sen}_{\boldsymbol{P}}(\sigma')(x) = x$$

Therefore, $\mathsf{Sen}_{\boldsymbol{P}}(\sigma' \circ \sigma)(x) = (\mathsf{Sen}_{\boldsymbol{P}}(\sigma') \circ \mathsf{Sen}_{\boldsymbol{P}}(\sigma))(x)$.
For $\downarrow x.\, \varphi$ the proof is:

$$
\begin{aligned}
\mathsf{Sen}_{\boldsymbol{P}}(\sigma' \circ \sigma)(\downarrow x.\, \varphi) =& \{\text{def. of } \mathsf{Sen}_{\boldsymbol{P}}\} \\
& \downarrow x.\, \mathsf{Sen}_{\boldsymbol{P}}(\sigma' \circ \sigma)(\varphi) \\
=& \{\text{induction hypothesis}\} \\
& \downarrow x.\, (\mathsf{Sen}_{\boldsymbol{P}}(\sigma') \circ \mathsf{Sen}_{\boldsymbol{P}}(\sigma))(\varphi) \\
=& \{\text{def. of } \mathsf{Sen}_{\boldsymbol{P}}\} \\
& (\mathsf{Sen}_{\boldsymbol{P}}(\sigma') \circ \mathsf{Sen}_{\boldsymbol{P}}(\sigma))(\downarrow x.\, \varphi)
\end{aligned}
$$

For the operator $@_x$ the proof follows similarly. □

4.4 The satisfaction relation

Definition 31. *Let* $\Sigma = \langle \mathrm{Act}, \mathrm{Prop} \rangle$ *be a signature,* $M = (W, w_0, R, V) \in \mathsf{Mod}_{\boldsymbol{P}}(\Sigma')$ *and* $\varphi \in \mathsf{Sen}_{\boldsymbol{P}}(\Sigma)$ *a sentence. Then,*

$$(M \models_{\boldsymbol{P}} \varphi) \;=\; \prod_{g \in W^X} (M, g, w_0 \models_{\boldsymbol{P}} \varphi)$$

where $g : X \to W$ *is a valuation function such that, for* $x \in X$, $g[x \mapsto w]$ *denotes the valuation given by* $g[x \mapsto w](x) = w$ *and* $g[x \mapsto w](y) = g(y)$ *for any* $y \neq x \in X$. *If* φ *is a formula without free variables then the valuation function* g *is irrelevant, that is,* $(M, g, w \models_{\boldsymbol{P}} \varphi) = (M, w \models_{\boldsymbol{P}} \varphi)$.
Thus, $\models_{\boldsymbol{P}}$ *is recursively defined as* $\models_{\boldsymbol{S}}$, *Definition 26, except for the modal operators and the new hybrid operators.*

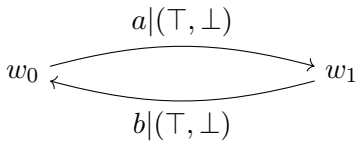
- $(M, g, w \models_{\boldsymbol{P}} [\alpha]\varphi) = (\,[\alpha^+](M, g, w, \varphi^+), \langle \alpha^+ \rangle (M, g, w, \varphi^-)\,)$
- $(M, g, w \models_{\boldsymbol{P}} \langle \alpha \rangle \varphi) = (\,\langle \alpha^+ \rangle (M, g, w, \varphi^+), [\alpha^+](M, g, w, \varphi^-)\,)$
- $(M, g, w \models_{\boldsymbol{P}} \cancel{[\alpha]}\varphi) = (\,\langle \alpha^- \rangle (M, g, w, \varphi^-), [\alpha^-](M, g, w, \varphi^+)\,)$
- $(M, g, w \models_{\boldsymbol{P}} \cancel{\langle \alpha \rangle}\varphi) = (\,[\alpha^-](M, g, w, \varphi^-), \langle \alpha^- \rangle (M, g, w, \varphi^+)\,)$
- $(M, g, w \models_{\boldsymbol{P}} x) = (1, 0)$ *iff* $g(x) = w$
- $(M, g, w \models_{\boldsymbol{P}\downarrow} x.\varphi) = (M, g[x \mapsto w], w \models_{\boldsymbol{P}} \varphi)$
- $(M, g, w \models_{\boldsymbol{P}} @_x \varphi) = (M, g, g(x) \models_{\boldsymbol{P}} \varphi)$

and

- $[\alpha^+](M, g, w, \varphi^*) = \prod_{w' \in W} (\widehat{R}^+_\alpha(w, w') \rightharpoonup (M, g, w' \models_{\boldsymbol{P}} \varphi)^*)$
- $[\alpha^-](M, g, w, \varphi^*) = \prod_{w' \in W} (\widehat{R}^-_\alpha(w, w') \rightharpoonup (M, g, w' \models_{\boldsymbol{P}} \varphi)^*)$
- $\langle \alpha^+ \rangle (M, g, w, \varphi^*) = \bigsqcup_{w' \in W} (\widehat{R}^+_\alpha(w, w') \sqcap (M, g, w' \models_{\boldsymbol{P}} \varphi)^*)$
- $\langle \alpha^- \rangle (M, g, w, \varphi^*) = \bigsqcup_{w' \in W} (\widehat{R}^-_\alpha(w, w') \sqcap (M, g, w' \models_{\boldsymbol{P}} \varphi)^*)$

with $\alpha \in Str(\mathrm{Act})$.

Example 18. *Consider* $\boldsymbol{P}(\mathbf{2})$, *let* $\Sigma = \langle \{a, b\}, \varnothing \rangle$ *be a signature with the set of propositions empty. Consider the following model* $M = (W, w_0, R, V)$ *where all transitions represent consistent information.*



Since we are in a case where the information of the model is consistent, we expect that the valuation of $\downarrow x.\langle a;b\rangle x$ *at* w_0 *represents consistent information. Moreover, we can expect by looking at* M *that the result of the valuation is going to be* $(\top,\bot)$ *because it is possible from* w_0 *to go through action* a*, to state* w_1*, and then, through action* b*, to return to state* w_0*. Notice that,*

$$\widehat{R}_{a;b}(w_0,w_0) = R_a(w_0,w_1) \sqcap R_b(w_1,w_0) = (\top,\bot)$$

$$\widehat{R}_{a;b}(w_0,w_1) = \bigsqcup_{v\in W}(R_a(w_0,v) \sqcap R_b(v,w_1)) = (\bot,\top)$$

Thus,

$$\begin{aligned}
&M,g,w_0 \models_{\boldsymbol{P}} \downarrow x.\langle a;b\rangle x\\
=&M,g[x\mapsto w_0],w_0 \models_{\boldsymbol{P}} \langle a;b\rangle x\\
=&\Big(\langle (a;b)^+\rangle (M,g,w_0,x^+)\,,\,[(a;b)^+](M,g,w_0,x^-)\Big)\\
=&\left(\bigsqcup_{w'\in W}(\widehat{R}^+_{a;b}(w_0,w')\sqcap (M,g,w'\models_{\boldsymbol{P}} x)^+)\,,\,\prod_{w'\in W}(\widehat{R}^+_{a;b}(w_0,w')\to (M,g,w'\models_{\boldsymbol{P}} x)^-)\right)\\
=&\Big(\widehat{R}^+_{a;b}(w_0,w_0)\sqcap (M,g,w_0\models_{\boldsymbol{P}} x)^+\,,\,\widehat{R}^+_{a;b}(w_0,w_0)\to (M,g,w_0\models_{\boldsymbol{P}} x)^-\Big)\\
=&(\top\sqcap\top,\top\to\bot) = (\top,\bot)
\end{aligned}$$

which aligns with our expectations.

With the previous example we emphasized the importance of remembering that paraconsistent logic $\boldsymbol{S}(\mathscr{A})$ and $\boldsymbol{P}(\mathscr{A})$ are meant to generalize models in which the information is consistent.

Theorem 6. *Let* $\Sigma=\langle \mathrm{Act},\mathrm{Prop}\rangle$ *and* $\Sigma'=\langle \mathrm{Act}',\mathrm{Prop}'\rangle$ *be signatures in* $\mathsf{Sign}_{\boldsymbol{P}}$ *and* $\sigma:\Sigma\to\Sigma'$ *be a signature morphism. For any model* $M'=(W',w'_0,R',V')\in \mathsf{Mod}_{\boldsymbol{P}}(\Sigma')$ *and a formula* $\varphi\in \mathsf{Sen}_{\boldsymbol{P}}(\Sigma)$*,*

$$\big(M'|_\sigma \models_{\boldsymbol{P}} \varphi\big) = \big(M' \models_{\boldsymbol{P}} \mathsf{Sen}_{\boldsymbol{P}}(\sigma)(\varphi)\big) \tag{4.1}$$

Proof. According to the definition of $\models_{\boldsymbol{P}}$ we want to prove that for $w'_0\in W'$ and for any valuation $g:X\to W'$, $\big(M'|_\sigma,g,w'_0\models_{\boldsymbol{P}}\varphi\big) = \big(M',g,w'_0\models_{\boldsymbol{S}}\mathsf{Sen}_{\boldsymbol{S}}(\sigma)(\varphi)\big)$.
Recall the proof of the satisfaction condition for institution $\boldsymbol{S}$, Theorem 4, where we show that for any $w\in W$ $\big(M'|_\sigma,w\models_{\boldsymbol{S}}\varphi\big) = \big(M',w\models_{\boldsymbol{S}}\mathsf{Sen}_{\boldsymbol{S}}(\sigma)(\varphi)\big)$.
Consider now $\Sigma,\Sigma'\in|\mathsf{Sign}_{\boldsymbol{P}}|$ and a signature morphism $\sigma:\Sigma\to\Sigma'$. For any model $M'=(W',w'_0,R',V')\in\mathsf{Mod}_{\boldsymbol{P}}(\Sigma')$ and a Σ-sentence, and for any $w\in W'$

$$\big(M'|_\sigma,g,w\models_{\boldsymbol{P}}\varphi\big) = \big(M',g,w\models_{\boldsymbol{P}}\mathsf{Sen}_{\boldsymbol{S}}(\sigma)(\varphi)\big) \tag{4.2}$$

The proof of this statement is done by induction over the structure of sentences. Cases $\bot$, $p\in\mathrm{Prop}$, $\wedge$, $\vee$, $\to$ and $\circ$ are like the ones for institution $\boldsymbol{S}$ (with the valuation function being irrelevant for these cases). Modal operators require some changes to cater for being now indexed by regular expressions. We follow with the proof for connective $[\alpha]\varphi$ (as the

remaining modal connectives are analogous). Like before we write $\sigma(p)$ instead of $\sigma_{\mathrm{Prop}}(p)$ for any $p \in \mathrm{Prop}$, and $\sigma(a)$ instead of $\sigma_{\mathrm{Act}}(a)$ for any $a \in \mathrm{Act}$. We also omit the subscript $\boldsymbol{P}$.

$$
\begin{aligned}
& M', g, w \models \mathsf{Sen}(\sigma)([\alpha]\,\varphi) \\
=& \{\text{definition of } \mathsf{Sen}\} \\
& M', g, w \models [\widehat{\sigma}(\alpha)]\,\mathsf{Sen}(\sigma)(\varphi) \\
=& \{\text{definition of } \models\} \\
& ([\widehat{\sigma}(\alpha)^+](M', g, w, \mathsf{Sen}(\sigma)(\varphi)^+)\,, \langle\widehat{\sigma}(\alpha)^+\rangle(M', g, w, \mathsf{Sen}(\sigma)(\varphi)^-)) \\
=& \{\text{def. of } [\alpha^+] \text{ and } \langle\alpha^+\rangle\}
\end{aligned}
$$

$$
\begin{aligned}
& \Big(\prod_{w' \in W'} (\widehat{R}'^{+}_{\widehat{\sigma}(\alpha)}(w, w') \rightharpoonup (M', g, w' \models \mathsf{Sen}(\sigma)(\varphi))^+),\ \bigsqcup_{w' \in W'} (\widehat{R}'^{+}_{\widehat{\sigma}(\alpha)}(w, w') \sqcap (M', g, w' \models \mathsf{Sen}(\sigma)(\varphi))^-)\Big) \\
=& \{(\text{step } \star\star)\} \\
& \Big(\prod_{w' \in W} (\widehat{R}^{+}_{\alpha}(w, w') \rightharpoonup (M'|_\sigma, g, w \models \varphi)^+),\ \bigsqcup_{w' \in W} (\widehat{R}^{+}_{\alpha}(w, w') \sqcap (M'|_\sigma, g, w \models \varphi)^-)\Big) \\
=& \{\text{def. } [\alpha^+] \text{ and } \langle\alpha^+\rangle\} \\
& ([\alpha^+](M'|_\sigma, g, w, \varphi^+)\,, \langle\alpha^+\rangle(M'|_\sigma, g, w, \varphi^-)) \\
=& \{\text{definition of } \models\} \\
& M'|_\sigma, g, w \models [\alpha]\,\varphi
\end{aligned}
$$

(step $\star\star$) By the definition of reduct we have that $W = W'$ and for any $w, w' \in W$, $\widehat{R}'_{\widehat{\sigma}(\alpha)}(w, w') = \widehat{R}_\alpha(w, w') = (\mathit{tt}, \mathit{ff})$ for some $\mathit{tt}, \mathit{ff} \in A$. Also by the induction hypothesis, $(M', g, w \models \mathsf{Sen}(\sigma)(\varphi)) = (M'|_\sigma, g, w \models \varphi)$, which is equivalent to writing:

$$
\begin{aligned}
& \Big((M', g, w \models \mathsf{Sen}(\sigma)(\varphi))^+, (M', g, w \models \mathsf{Sen}(\sigma)(\varphi))^-)\Big) \\
&= \Big((M'|_\sigma, g, w \models \varphi)^+, (M'|_\sigma, g, w \models \varphi)^-\Big)
\end{aligned}
\tag{4.3}
$$

Therefore, by (4.3), we have that $M', g, w \models \mathsf{Sen}(\sigma)(\varphi))^+ = (M'|_\sigma, g, w \models \varphi)^+$ as well as $(M', g, w \models \mathsf{Sen}(\sigma)(\varphi))^- = (M'|_\sigma, g, w \models \varphi)^-$.
For the case of state variables we have that $M', g, w \models \mathsf{Sen}(\sigma)(x)$ is either $(1, 0)$ or $(0, 1)$. If it is equal to $(1, 0)$ then,

$$
\begin{aligned}
(M', g, w \models \mathsf{Sen}(\sigma)(x)) = (1, 0) \Leftrightarrow & \{\text{def. of } \mathsf{Sen}\} \\
& (M', g, w \models x) = (1, 0) \\
\Leftrightarrow & \{\text{induction hypothesis}\} \\
& (M'|_\sigma, g, w \models x) = (1, 0)
\end{aligned}
$$

otherwise,

$$
\begin{aligned}
(M', g, w \models \mathsf{Sen}(\sigma)(x)) = (0, 1) \Leftrightarrow & \{\text{def. of } \mathsf{Sen}\} \\
& (M', g, w \models x) = (0, 1) \\
\Leftrightarrow & \{\text{induction hypothesis}\} \\
& (M'|_\sigma, g, w \models x) = (0, 1)
\end{aligned}
$$

The case for binder is as follows:

$$\begin{aligned}
M', g, w \models \mathsf{Sen}(\sigma)(\downarrow x.\varphi) =& \{\text{def. of } \mathsf{Sen}\} \\
& M', g, w \models\downarrow x.\mathsf{Sen}(\sigma)(\varphi) \\
=& \{\text{def. of } \models\} \\
& M', g[x \mapsto w], w \models \mathsf{Sen}(\sigma)(\varphi) \\
=& \{\text{induction hypothesis}\} \\
& M'|_\sigma, g[x \mapsto w], w \models \varphi \\
=& \{\text{def. of } \models\} \\
& M'|_\sigma, g, w \models\downarrow x.\varphi
\end{aligned}$$

For operator $@_x$ we have that:

$$\begin{aligned}
M', g, w \models \mathsf{Sen}(\sigma)(@_x.\varphi) =& \{\text{def. of } \mathsf{Sen}\} \\
& M', g, w \models @_x.\mathsf{Sen}(\sigma)(\varphi) \\
=& \{\text{def. of } \models\} \\
& M', g, g(x) \models \mathsf{Sen}(\sigma)(\varphi) \\
=& \{\text{induction hypothesis}\} \\
& M'|_\sigma, g, g(x) \models \varphi \\
=& \{\text{def. of } \models\} \\
& M'|_\sigma, g, w \models @_x\varphi
\end{aligned}$$

In conclusion we have proven Equation 4.2 for institution $\boldsymbol{P}$. Moreover, the equation still holds when w is replaced by $w'_0 \in W'$ (which by the definition of reduct is equal to the initial state of $M'|_\sigma$). Consequently, $\big(M'|_\sigma \models_{\boldsymbol{P}} \varphi\big) = \big(M' \models_{\boldsymbol{P}} \mathsf{Sen}_{\boldsymbol{P}}(\sigma)(\varphi)\big)$. □

The next Theorem is a consequence of this entire chapter where every ingredient of institution $\boldsymbol{P}(\mathscr{A})$ is properly formalized in terms of category theory and the satisfaction condition, for many-valued institutions, is provided.

Theorem 7. $\boldsymbol{P}(\mathscr{A}) = (\mathsf{Sign}_{\boldsymbol{P}(\mathscr{A})}, \mathsf{Sen}_{\boldsymbol{P}(\mathscr{A})}, \mathsf{Mod}_{\boldsymbol{P}(\mathscr{A})}, \models_{\boldsymbol{P}(\mathscr{A})})$ *is an institution, for a fixed twisted algebra* $\mathscr{A}$*.*

Chapter 5

Specification theory

This chapter is one of the main contribution of this dissertation: a specification theory for paraconsistent logics.

For such contribution framing $\boldsymbol{S}(\mathscr{A})$ and $\boldsymbol{P}(\mathscr{A})$ as institutions, for any fixed twisted algebra $\mathscr{A}$, plays a crucial role, since the specification theory presented in the Background Section 2.3 is institution-independent.

Hence, after formalizing $\boldsymbol{S}$ and $\boldsymbol{P}$ as institutions it is possible to adapt the specification theory in Section 2.3 in order to cater for paraconsistency in a system's description.

Notice that the specification theory we present for institutions $\boldsymbol{S}$ and $\boldsymbol{P}$ generalizes the one presented in Section 2.3 for an arbitrary institution. Whenever pairs of weights are complementary we expect that the concepts and results of this chapter coincide with the ones in that Section.

Chapter 5 is divided in two sections. The first one focuses on defining specification building-operators and their compositional relationships alongside a palette of constructors to combine specifications. The second section presents the relations *similarity* and *bisimilarity*, found in [CMB22], to compare models. Such relations define what it means for two paraconsistent models to be indistinguishable from the observer's point of view. This allow us to define behavioural abstraction specifications.

To begin with let us define what a *paraconsistent specification* is.

Definition 32. *A paraconsistent* specification *is a pair* $SP = (Sig(SP), Mod(SP))$ *where* $Sig(SP)$ *is a signature in* Sign*, category of algebraic signatures, and* $Mod(SP)$ *is a mapping*

$$Mod(SP) : \mathsf{Mod}(Sig(SP)) \to A \times A$$

For a model $M \in \mathsf{Mod}(Sig(SP))$ *such that* $Mod(SP)(M) = (\mathbb{tt}, \mathbb{ff})$*, the value* $\mathbb{tt}$ *represents the evidence degree that* M *is a model of specification* SP *and* $\mathbb{ff}$ *the evidence degree that* M *is not a model of* SP*.*

From now one, any mentioned specification is a paraconsistent one.

5.1 Structured Specification

A great tool to build specifications are *flat specifications*. This tool is used whenever the requirements are easily expressed in a set of sentences in our logic.

Definition 33. *If $\Sigma \in |\mathsf{Sign}|$ is a signature and $\Phi \subseteq |\mathsf{Sen}(\Sigma)|$ is a set of Σ-sentences, often called axioms, then $SP = (\Sigma, \Phi)$ is a flat specification. Consequently,*

- $Sig(SP) = \Sigma$
- $Mod(SP)(M) = \overqbar{\prod}_{\varphi \in \Phi} (M \models \varphi)$

The **Union** operator is used whenever we pretend to combine two specifications, over the same signature, the result is a specification $SP \cup SP'$.

Definition 34. *Let SP and SP' be two specifications over the same signature, $\Sigma \in |\mathsf{Sign}|$. Then $SP \cup SP'$ is*

- $Sig(SP \cup SP') = \Sigma$
- $Mod(SP \cup SP')(M) = Mod(SP)(M) \sqcap Mod(SP')(M)$

If $SP_1 = (\Sigma, \Phi_1)$ and $SP_2 = (\Sigma, \Phi_2)$ are flat specifications, then

$$Mod((\Sigma, \Phi_1) \cup (\Sigma, \Phi_2))(M) = Mod(\Sigma, \Phi_1 \cup \Phi_2)(M)$$

The next two operators take a morphism between signatures as a parameter. **Translation** is a tool to translate a specification according to a morphism, while **Hiding** is usually used to hide certain details of a specification via a morphism.

Definition 35. *If SP is a Σ-specification and $\sigma : \Sigma \to \Sigma'$ is a signature morphism then,*

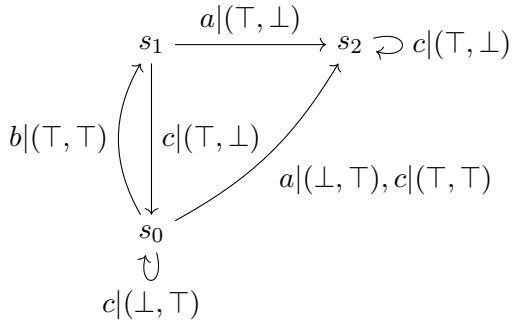
- $Sig(SP \textbf{ with } \sigma) = \Sigma'$
- $Mod(SP \textbf{ with } \sigma)(M') = Mod(SP)(M'|_\sigma)$, *note that M' is a Σ'-model.*

Definition 36. *If SP' is a Σ'-specification and $\sigma : \Sigma \to \Sigma'$ signature morphism then,*

- $Sig(SP' \textbf{ hide via } \sigma) = \Sigma$
- $Mod(SP' \textbf{ hide via } \sigma)(M) = \left(\underqbar{\bigsqcup}_{N \in M^\sigma} Mod(SP')(N) \right)$

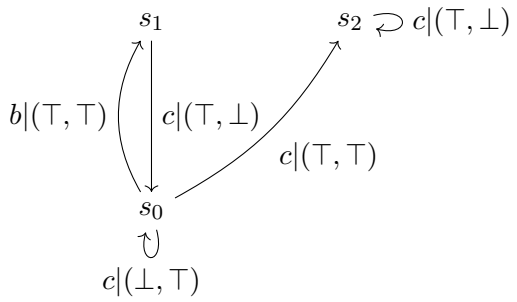
where M^σ is the class of all σ-expansions of M, i.e. $M^\sigma = \{N \in \mathsf{Mod}(\Sigma') | N|_\sigma = M\}$.

Example 19. *Consider $\boldsymbol{S(2)}$, that is, weights are either $\top$ or $\bot$. Given the signature $\Sigma = \langle \{b, c\}, \{p, q\} \rangle$, the specification $SP = \langle \Sigma, \Phi \rangle$ where $\Phi = \{\langle c \rangle \top,\ p \vee q,\ q \to \langle c \rangle p\}$ and the inclusion morphism $\sigma : \langle \{b, c\}, \{p, q\} \rangle \to \langle \{a, b, c\}, \{p, q\} \rangle$. Let $M' = (\{s_0, s_1, s_2\}, R', V')$ be a $\langle \{a, b, c\}, \{p, q\} \rangle$-transition model depicted below.*



where $V'(s_0, p) = V'(s_0, q) = V'(s_1, q) = (\top, \top)$, $V'(s_1, p) = (\bot, \bot)$, $V'(s_2, p) = (\top, \bot)$ *and* $V'(s_2, q) = (\bot, \top)$.

The following Σ*-model* $M'|_\sigma = (W, R, V)$ *is the* σ*-reduct of* M'*:*



By the definition of σ*-reduct,* $W = W'$ *and* $V(s, r) = V'(s, r)$ *for any* $s \in W$ *and* $r \in \{p, q\}$. *Then,*

- $Sig(SP \textbf{ with } \sigma) = \langle \{a, b, c\}, \{p, q\} \rangle$
- $Mod(SP \textbf{ with } \sigma)(M') = Mod(SP)(M'|_\sigma) = \overline{\prod}_{\varphi \in \Phi} (M'|_\sigma \models \varphi)$

Notice that for sentence $\langle c \rangle \top$*:*

$$\begin{aligned} M'|_\sigma, s_0 \models \langle c \rangle \top =& (\langle c^+ \rangle (M'|_\sigma, s_0, \top)^+, [c^+](M'|_\sigma, s_0, \top^-)) \\ =& (R_c^+(s_0, s_2) \wedge (M, s_2 \models \top)^+, R_c^+(s_0, s_0) \wedge (M'|_\sigma, s_0 \models \top)^-) \\ =& (\top \to \top, \bot \wedge \bot) \;=\; (\top, \bot) \end{aligned}$$

Similarly, we find that $(M'|_\sigma, s_1 \models \langle c \rangle \top) \;=\; (\top, \bot) \;=\; M'|_\sigma, s_2 \models \langle c \rangle \top$.

Thus, the evidence degree of model $M'|_\sigma$ *satisfying* $\langle c \rangle \top$ *is* $\top$ *and the evidence degree of not satisfying is* $\bot$.

$$(M'|_\sigma \models \langle c \rangle \top) = \left(\overline{\prod}_{s \in W} (M'|_\sigma, s \models \langle c \rangle \top) \right) \;=\; (\top, \bot)$$

This is a case where the pair of weights represents consistent information we are completely certain that at any state it is possible to go to another state trough action c.
For sentence $p \vee q$*, we get*

$$\begin{aligned} (M'|_\sigma \models p \vee q) =& \left(\overline{\prod}_{s \in W} (M'|_\sigma, s \models p \vee q) \right) \\ =& \left(\overline{\prod}_{s \in W} ((M'|_\sigma, s \models p) \amalg (M'|_\sigma, s \models q)) \right) \\ =& ((\top, \top) \amalg (\top, \top)) \overline{\prod} ((\bot, \bot) \amalg (\top, \top)) \overline{\prod} ((\top, \bot) \amalg (\bot, \top)) \\ =& (\top \vee \top, \top \wedge \top) \overline{\prod} (\bot \vee \top, \bot \wedge \top) \overline{\prod} (\top \vee \bot, \bot \wedge \top) \\ =& (\top, \top) \overline{\prod} (\top, \bot) \overline{\prod} (\top, \bot) \\ =& (\top \wedge \top \wedge \top, \top \vee \bot \vee \bot) = (\top, \top) \end{aligned}$$

In this case the pair represents paraconsistent information: the evidence degree of $p \vee q$ *being satisfied and not satisfied by model* $M'|_\sigma$ *is* $\top$.

For sentence $q \to \langle c \rangle p$:

$$\begin{aligned}
&M'|_\sigma, s_0 \models q \to \langle c \rangle p \\
=&(M'|_\sigma, s_0 \models q) \Rightarrow (M'|_\sigma, s_0 \models \langle c \rangle p) \\
=&(\top, \top) \Rightarrow (\langle c^+ \rangle (M, s_0, p^+), [c^+](M, s_0, p^-)) \\
=&(\top, \top) \Rightarrow (R_c^+(s_0, s_2) \wedge (M'|_\sigma, s_2 \models p)^+), R_c^+(s_0, s_2) \to (M'|_\sigma, s_2 \models p)^-) \\
=&(\top, \top) \Rightarrow (R_c^+(s_0, s_2) \wedge V^+(s_2, p), R_c^+(s_0, s_2) \to V^-(s_2, p)) \\
=&(\top, \top) \Rightarrow (\top \wedge \top, \top \to \bot) \\
=&(\top, \top) \Rightarrow (\top, \bot) \\
=&(\top, \bot)
\end{aligned}$$

Similarly, we have that $(M'|_\sigma, s_1 \models q \to \langle c \rangle p) = (\top, \top)$ *and* $(M'|_\sigma, s_2 \models q \to \langle c \rangle p) = (\top, \bot)$. *Therefore,*

$$(M'|_\sigma \models q \to \langle c \rangle p) = (\top, \bot) \sqcap (\top, \top) \sqcap (\top, \bot) = (\top, \top)$$

The sentence can be read as: whenever q holds there is a transition trough action c to a state where p holds. The evidence degree of model $M'|_\sigma$ satisfying such a requirement and not satisfying it is $\top$.

In conclusion,

$$\begin{aligned}
Mod(SP)(M'|_\sigma) =&(M'|_\sigma \models [c] \top) \sqcap (M'|_\sigma \models p \vee q) \sqcap (M'|_\sigma \models q \to \langle c \rangle p) \\
=&(\top, \bot) \sqcap (\top, \top) \sqcap (\top, \top) \\
=&(\top \wedge \top \wedge \top, \bot \vee \top \vee \top) = (\top, \top) \\
=&Mod(SP \textbf{ with } \sigma)(M')
\end{aligned}$$

The evidence degree of M' satisfying specification SP **with** *σ, that is, the specification SP translated via σ, is $\top$. The evidence degree of not satisfying is $\top$. Notice that in this case we have* paraconsistent *information.*

As a side note, since satisfaction evaluation in $\boldsymbol{S}$ is global, specifications in this institution are quite strong. That is due to the fact that any requirements of the specification is to hold in every state. In such cases the pair of weights $Mod(SP)$ tend to collapse to $(0, 1)$.

Example 20. *Consider $\boldsymbol{S}(\ddot{\boldsymbol{G}})$, $\Sigma = \langle \{a\}, \{p, q\} \rangle$ and the following paraconsistent transition model $M = (\{s_0, s_1\}, R, V)$ depicted below.*

$$s_0 \xrightarrow{\quad a|(1, 0.3) \quad} s_1 \circlearrowright a|(0.1, 0.2)$$

With $V(s_0, p) = (0.4, 0.6)$, $V(s_0, q) = (0.9, 0.2)$ and $V(s_1, p) = (0.3, 0.7)$, $V(s_1, q) = (0.5, 0.5)$. Let $SP = (SP_1 \cup SP_2)$ be a Σ-specification such that $SP_1 = (\Sigma, \{[a]p, \langle a \rangle q\})$ and $SP_2 = (\Sigma, \{p \to \neg q, \langle a \rangle q \to (p \vee \neg q)\})$. Then,

$$\begin{aligned}
Mod(SP)(M) =&Mod(SP_1 \cup SP_2)(M) \\
=&Mod(SP_1)(M) \sqcap Mod(SP_2)(M) \\
=&\left(\bigsqcap_{\varphi \in \Phi_1} M \models \varphi \right) \sqcap \left(\bigsqcap_{\varphi \in \Phi_2} M \models \varphi \right)
\end{aligned}$$

where $\Phi_1 = \{[a]p, \langle a\rangle q\}$ *and* $\Phi_2 = \{p \to \neg q, \langle a\rangle q \to (p \vee \neg q)\}$.

$$\begin{aligned}\left(\underset{s\in W}{\sqcap} M, s \models [a]p\right) =& (M, s_0 \models [a]p) \sqcap (M, s_1 \models [a]p)\\ =& ([a^+](M, s_0, p^+), \langle a^+\rangle(M, s_0, p^-)) \sqcap ([a^+](M, s_1, p^+), \langle a^+\rangle(M, s_1, p^-))\\ =& (R_a^+(s_0, s_1) \to (M, s_1 \models p)^+, \min(R_a^+(s_0, s_1), (M, s_1 \models p)^-)) \sqcap\\ & (R_a^+(s_1, s_1) \to (M, s_1 \models p)^+, \min(R_a^+(s_1, s_1), (M, s_1 \models p)^-))\\ =& (1 \to 0.3, \min(1, 0.7)) \sqcap (0.1 \to 0.3, \min(0.1, 0.7))\\ =& (0.3, 0.7) \sqcap (1, 0.1) = (\min(0.3, 1), \max(0.7, 0.1)) = (0.3, 0.7)\end{aligned}$$

$$\begin{aligned}\left(\underset{s\in W}{\sqcap} M, s \models \langle a\rangle q\right) =& (M, s_0 \models \langle a\rangle q) \sqcap (M, s_1 \models \langle a\rangle q)\\ =& (\langle a^+\rangle(M, s_0, q^+), [a^+](M, s_0, q^-)) \sqcap (\langle a^+\rangle(M, s_1, q^+), [a^+](M, s_1, q^-))\\ =& (\min(R_a^+(s_0, s_1), (M, s_1 \models q)^+), R_a^+(s_0, s_1) \to (M, s_1 \models q)^-) \sqcap\\ & (\min(R_a^+(s_1, s_1), (M, s_1 \models q)^+), R_a^+(s_1, s_1) \to (M, s_1 \models q)^-)\\ =& (\min(1, 0.5), 1 \to 0.5) \sqcap (\min(0.1, 0.5), 0.1 \to 0.5)\\ =& (0.5, 0.5) \sqcap (0.1, 1) = (\min(0.5, 0.1), \max(0.5, 1)) = (0.1, 1)\end{aligned}$$

Therefore,

$$Mod(SP_1)(M) = (0.3, 0.7) \sqcap (0.1, 1) = (\min(0.3, 0.1), \max(0.7, 1)) = (0.1, 1)$$

Also,

$$\begin{aligned}\left(\underset{s\in W}{\sqcap} M, s \models p \to \neg q\right) =& (M, s_0 \models p \to \neg q) \sqcap (M, s_1 \models p \to \neg q)\\ =& ((M, s_0 \models p) \Rightarrow /\!\!/(M, s_0 \models q)) \sqcap ((M, s_1 \models p) \Rightarrow /\!\!/(M, s_1 \models q))\\ =& ((0.4, 0.6) \Rightarrow (0.2, 0.9)) \sqcap ((0.3, 0.7) \Longrightarrow (0.5, 0.5))\\ =& (0.4 \to 0.2, \min(0.4, 0.9)) \sqcap (0.3 \to 0.5, \min(0.3, 0.5))\\ =& (0.2, 0.4) \sqcap (1, 0.3) = (\min(0.2, 1), \max(0.4, 0.3)) = (0.2, 0.4)\end{aligned}$$

$$\begin{aligned}\left(\underset{s\in W}{\sqcap} M, s \models \langle a\rangle q \to (p \vee \neg q)\right) =& (M, s_0 \models \langle a\rangle q \to (p \vee \neg q)) \sqcap (M, s_1 \models \langle a\rangle q \to (p \vee \neg q))\\ =& \Big((M, s_0 \models \langle a\rangle q) \Rightarrow (M, s_0 \models p \vee \neg q)\Big) \sqcap\\ & \Big((M, s_1 \models \langle a\rangle q) \Rightarrow (M, s_1 \models p \vee \neg q)\Big)\\ =& \Big((0.5, 0.5) \Rightarrow (M, s_0 \models p \sqcup /\!\!/(M, s_0 \models q))\Big) \sqcap\\ & \Big((0.1, 1) \Rightarrow (M, s_1 \models p \sqcup /\!\!/(M, s_1 \models q))\Big)\\ =& \Big((0.5, 0.5) \Rightarrow (\max(0.4, 0.2), \min(0.6, 0.9))\Big) \sqcap\end{aligned}$$

$$
\begin{aligned}
&\Big((0.1,1) \Rightarrow (\max(0.3,0.5), \min(0.7,0.5))\Big) \\
=&\Big((0.5,0.5) \Rightarrow (0.4,0.6)\Big) \sqcap \Big((0.1,1) \Rightarrow (0.5,0.5)\Big) \\
=&(0.5 \to 0.4, \min(0.5,0.6)) \sqcap (0.1 \to 0.5, \min(0.1,0.5)) \\
=&(0.4,0.5) \sqcap (1,0.1) \;=\; (0.4,0.5)
\end{aligned}
$$

That is,

$$Mod(SP_2)(M) = (0.2,0.4) \sqcap (0.4,0.5) = (\min(0.2,0.4), \max(0.4,0.5)) = (0.2,0.5)$$

Hence,

$$
\begin{aligned}
Mod(SP)(M) =& Mod(SP_1)(M) \sqcap Mod(SP_2)(M) \\
=&(0.1,1) \sqcap (0.2,0.5) = (\min(0.1,0.2), \max(1,0.5)) = (0.1,1)
\end{aligned}
$$

In conclusion, the evidence degree of model M satisfying specification SP is 0.1 *and the evidence degree of not satisfying is* 1*. Since the sum of the degrees is bigger than 1, we are in a case where there is* inconsistency*.*
It is worth noting that since SP_1 and SP_2 are flat specifications $Mod(SP_1 \cup SP_2)(M) = Mod(\Sigma, \Phi_1 \cup \Phi_2)(M)$.

Definition 37. *We write $SP \rightsquigarrow SP'$ when a specification SP'* implements *another specification SP, if:*

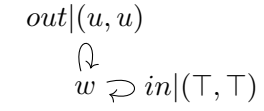
- $Sig(SP') = Sig(SP) = \Sigma$
- $Mod(SP')(M) \preccurlyeq Mod(SP)(M)$*, for all* $M \in \mathsf{Mod}(\Sigma)$

The following example is adapted from [MBHM18] and illustrates the concept of implementation of specifications.

Example 21. *Let $\boldsymbol{S}(\mathbf{3})$ and $\langle \mathrm{Act}, \varnothing \rangle$ a signature where the set of propositions is empty and the set of actions is* $\mathrm{Act} = \{in, out\}$ *with $\{in\}$ standing for the input of a text file and $\{out\}$ standing for the output of a zip-file.*
This example considers a file compressing service working only with text files. Starting with a loose specification SP_0 whose requirements are, at every state:

(0.1) $[in]\langle out\rangle\top$*, whenever a text file is received for compression a corresponding zip-file will be output*

(0.2) $\langle \mathrm{Act}\rangle\top$*, the system should never terminate*



Let $M_0 = (W, R, V)$ be a model where $W = \{w\}$ and $R_{in}(w, w) = (\top, \bot)$, $R_{out}(w, w) = (u, u)$, that is M_0 models a text compression service where it's uncertain when an output of zip-files happens. Trivially $(M, w \models \langle in \rangle \top) = (\top, \bot)$, that is, $(M \models \langle \text{Act} \rangle \top) = (\top, \bot)$.

$$
\begin{aligned}
Mod(SP)(M) =& (M, w \models [in]\langle out \rangle \top) \between (M, w \models \langle \text{Act} \rangle \top) \\
=& \left(R_{in}^{+} \to (M, w \models \langle out \rangle \top)^{+}, R_{in}^{+} \wedge (M, w \models \langle out \rangle \top)^{-} \right) \between (\top, \bot) \\
=& \left(\top \to (R_{out}^{+} \wedge (M, w \models \top)^{+}), \top \wedge (R_{out}^{+} \to (M, w \models \top)^{-}) \right) \between (\top, \bot) \\
=& \left(\top \to (u \wedge \top), \top \wedge (u \to \bot) \right) \between (\top, \bot) \\
=& \left(\top \to u, \top \wedge \bot \right) \between (\top, \bot) = (u \wedge \top, \bot \vee \bot) = (u, \bot)
\end{aligned}
$$

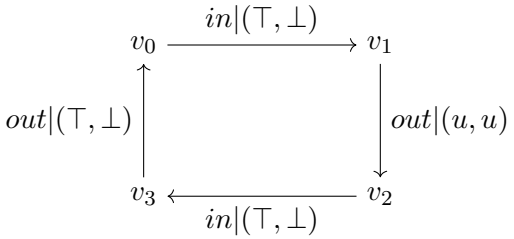
As stated, SP_0 is a very loose specification that would allow to start a computation with an output action and to have output actions without previous inputs. Because of that we will now consider a more concrete specification. Specification SP_1 will require that at every state:

(1.1) $\langle in \rangle \top \wedge [out] \bot$, at the beginning only an input action is allowed

(1.2) $[in](\langle out \rangle \top \wedge [in] \bot)$, after every input action the next action must be an output action

(1.3) $[out](\langle in \rangle \top \wedge [out] \bot)$, whenever there is an output action the system must go on with an input

It is easy to see that axiom (1.2) and (1.3) guarantee non termination, which corresponds to axiom (0.2), and that axiom (0.1) is represented in SP_1 by axiom (1.2). Note that axiom (1.2) can be rewritten as two axioms: (1.2.1) $[in]\langle out \rangle \top$ and (1.2.2) $[in][in] \bot$. Hence, for any model M in our signature, $Mod(SP_1)(M) \preccurlyeq Mod(SP_0)(M)$ since SP_0 is a looser specification we expect models to satisfy the requirements of SP_0 with more certainty than they satisfy the requirements of SP_1 and we expect models to not satisfy the requirements of SP_1 with more certainty than the ones from SP_0. That is $SP_0 \rightsquigarrow SP_1$.
Let M_1 be the model represented below where every output action in state v_3 happens and every output action after state v_1 is uncertain to happen or not.



One can check that $Mod(SP_1)(M_0) = (\bot, \top) \preccurlyeq Mod(SP_0)(M_0)$, as expected.
Also, $Mod(SP_1)(M_1) = (u, \bot) = Mod(SP_0)(M_1)$ thus $Mod(SP_1)(M_1) \preccurlyeq Mod(SP_0)(M_1)$

Theorem 8. [***Vertical Composition***] *Given specification SP, SP' and SP'' over the same signature Σ, such that $SP \rightsquigarrow SP'$ and $SP' \rightsquigarrow SP''$ then $SP \rightsquigarrow SP''$*

Proof. By hypothesis, $Sig(SP) = Sig(SP') = Sig(SP'')$, so we only are left to prove that for any model $M \in \mathsf{Mod}(\Sigma)$, $Mod(SP)(M) \preccurlyeq Mod(SP'')(M)$, which is trivial because of the transitivity property of $\preccurlyeq$.
Let $M \in \mathsf{Mod}(\Sigma)$ be any Σ-model such that $Mod(SP)(M) = (\mathit{tt}, \mathit{ff})$, $Mod(SP')(M) = (\mathit{tt}', \mathit{ff}')$ and $Mod(SP'')(M) = (\mathit{tt}'', \mathit{ff}'')$. By hypothesis $SP \rightsquigarrow SP'$ and $SP' \rightsquigarrow SP''$, that is,

$$(\mathit{tt}, \mathit{ff}) \preccurlyeq (\mathit{tt}', \mathit{ff}') \text{ and } (\mathit{tt}', \mathit{ff}') \preccurlyeq (\mathit{tt}'', \mathit{ff}'')$$

Using Property 6.:

$$\mathit{tt} \leqslant \mathit{tt}'' \text{ and } \mathit{ff} \geqslant \mathit{ff}''$$

Therefore, for any $M \in \mathsf{Mod}(\Sigma)$, $Mod(SP)(M) \preccurlyeq Mod(SP'')(M)$. □

Theorem 9. [***Horizontal Composition***] *Let SP_1, SP_1' and SP_2, SP_2' be specifications over the same signature, such that $SP_1 \rightsquigarrow SP_1'$ and $SP_2 \rightsquigarrow SP_2'$. Then,*

1. $(SP_1 \cup SP_2) \rightsquigarrow (SP_1' \cup SP_2')$
2. $(SP_1 \textbf{ with } \sigma) \rightsquigarrow (SP_1' \textbf{ with } \sigma)$
3. $(SP_1 \textbf{ hide via } \sigma) \rightsquigarrow (SP_1' \textbf{ hide via } \sigma)$

Proof. Let $\Sigma = \langle \mathrm{Act}, \mathrm{Prop} \rangle$ and $\Sigma' = \langle \mathrm{Act}', \mathrm{Prop}' \rangle$ be signatures and SP_1, SP_1', SP_2 and SP_2' be Σ-specifications.
For statement 1: using the definition of the union operator and hypothesis, $Sig(SP_1 \cup SP_2) = \Sigma = Sig(SP_1' \cup SP_2')$. Also, for any model $M \in \mathsf{Mod}(\Sigma)$

$$\begin{aligned}
Mod(SP_1 \cup SP_2)(M) =& \{\text{def. of } \cup\} \\
& Mod(SP_1)(M) \sqcap Mod(SP_2)(M) \\
\preccurlyeq& \{\text{since } SP_1 \rightsquigarrow SP_1', SP_2 \rightsquigarrow SP_2' \text{ and Property 5.}\} \\
& Mod(SP_1')(M) \sqcap Mod(SP_2')(M) \\
=& \{\text{def. of } \cup\} \\
& Mod(SP_1' \cup SP_2')(M)
\end{aligned}$$

For statement 2: let $\sigma : \langle \mathrm{Act}, \mathrm{Prop} \rangle \to \langle \mathrm{Act}', \mathrm{Prop}' \rangle$ be a signature morphism. Then, $Sig(SP_1 \textbf{ with } \sigma) = \Sigma' = Sig(SP_1' \textbf{ with } \sigma)$. And for any model $M \in \mathsf{Mod}(\Sigma)$

$$\begin{aligned}
Mod(SP_1 \textbf{ with } \sigma)(M) =& \{\text{def. of } \textbf{with } \sigma\} \\
& Mod(SP_1)(M|_\sigma) \\
\preccurlyeq& \{SP_1 \rightsquigarrow SP_1'\} \\
& Mod(SP_1')(M|_\sigma) \\
=& \{\text{def. of } \textbf{with } \sigma\} \\
& Mod(SP_1' \textbf{ with } \sigma)(M)
\end{aligned}$$

For statement 3: let $\sigma : \langle \mathrm{Act}', \mathrm{Prop}' \rangle \to \langle \mathrm{Act}, \mathrm{Prop} \rangle$ be a signature morphism and

$$Sig(SP_1 \textbf{ hide via } \sigma) = \Sigma' = Sig(SP_1' \textbf{ hide via } \sigma)$$

For any model $M \in \mathsf{Mod}(\Sigma')$:

$$\begin{aligned}
\mathsf{Mod}(SP_1 \textbf{ hide via } \sigma)(M) =& \{\text{def. of } \textbf{hide via } \sigma\} \\
& \left(\coprod_{N \in M^\sigma} Mod(SP_1)(N) \right) \\
\preccurlyeq& \{SP_1 \rightsquigarrow SP_1' \text{ and Property 5.}\} \\
& \left(\coprod_{N \in M^\sigma} Mod(SP_1')(N) \right) \\
=& \{\text{def. of } \textbf{hide via } \sigma\} \\
& \mathsf{Mod}(SP_1' \textbf{ hide via } \sigma)(M)
\end{aligned}$$

where $M^\sigma = \{N \in Mod(\Sigma) | N|_\sigma = M\}$. □

Definition 38. *Let $SP, SP_1, ..., SP_n$ be specifications over signatures $\Sigma, \Sigma_1, ..., \Sigma_n$, respectively, and $k : \mathsf{Mod}(\Sigma_1) \times ... \times \mathsf{Mod}(\Sigma_n) \to \mathsf{Mod}(\Sigma)$ a constructor. We say that $(SP_1, ..., SP_n)$ is a constructor implementation via k of SP, in symbols $SP \rightsquigarrow_k (SP_1, ..., SP_n)$ if for any $M_i \in \mathsf{Mod}(\Sigma_i)$*

$$\coprod_{i=1}^{n} Mod(SP_i)(M_i) \preccurlyeq Mod(SP)(k(M_1, ..., M_n))$$

We say that the implementation involves decomposition if $n > 1$.

We follow this section by redefining constructors in [MBHM18] to suit our paraconsistent logics.

Definition 39. *Let $\Sigma = \langle \mathrm{Act}, \mathrm{Prop} \rangle$ and $\Sigma' = \langle \mathrm{Act}', \mathrm{Prop}' \rangle$ be signatures such that $\mathrm{Prop} \subseteq \mathrm{Prop}'$ and $\mathrm{Act} \subseteq \mathrm{Act}'$. The signature extension constructor is $k_{ext} : \mathsf{Mod}(\Sigma) \to \mathsf{Mod}(\Sigma')$.*
For institution $\boldsymbol{S}$, let $M = (W, R, V) \in \mathsf{Mod}_{\boldsymbol{S}}(\Sigma)$ be a model. The constructor is defined as $k_{ext}(M) = (W, R', V')$ where for any $w \in W$:

- $R'_a[w] = \begin{cases} R_a[w] & \text{if } a \in \mathrm{Act} \\ \{(w, w', 0, 1) \text{ for all } w' \in W\} & \text{otherwise} \end{cases}$
- $V'(w, p) = \begin{cases} V(w, p) & \text{if } p \in \mathrm{Prop} \\ (0, 1) & \text{otherwise} \end{cases}$

For institution $\boldsymbol{P}$, let $M = (W, w_0, R, V) \in \mathsf{Mod}_{\boldsymbol{P}}(\Sigma)$ be a process. Then, $k_{ext}(M) = (W, w_0, R', V')$ with R' and V' defined above.

Definition 40. *Let $\Sigma_1 = \langle \mathrm{Act}, \mathrm{Prop} \rangle$ and $\Sigma_2 = \langle \mathrm{Act}', \mathrm{Prop}' \rangle$ be signatures. The parallel composition constructor is $k_\otimes : \mathsf{Mod}(\Sigma_1) \times \mathsf{Mod}(\Sigma_2) \to \mathsf{Mod}(\Sigma^\otimes)$ where $\Sigma^\otimes = \langle \mathrm{Act} \cup \mathrm{Act}', \mathrm{Prop} \cup \mathrm{Prop}' \rangle$.*
For institution $\boldsymbol{S}$, for any models $M = (W, R, V) \in \mathsf{Mod}_{\boldsymbol{S}}(\Sigma_1)$ and $M' = (W', R', V') \in \mathsf{Mod}_{\boldsymbol{S}}(\Sigma_2)$ the parallel composition is the model $M \otimes M' = (W^\otimes, R^\otimes, V^\otimes)$:

- $W^\otimes = W \times W'$
- *for any $(w, w') \in W^\otimes$*

– *if* $a \in \mathrm{Act} \cap \mathrm{Act}'$, $R_a(w,v) = (\alpha,\beta)$ *and* $R'_a(w',v') = (\alpha',\beta')$, *then*

$$(v,v') \in W^{\otimes} \text{ and } R_a^{\otimes}((w,w'),(v,v')) = (\alpha \sqcap \alpha', \beta \sqcup \beta')$$

– *if* $a \in \mathrm{Act} \backslash \mathrm{Act}'$, $R_a(w,v) = (\alpha,\beta)$, *then*

$$(v,w') \in W^{\otimes} \text{ and } R_a^{\otimes}((w,w'),(v,w')) = (\alpha,\beta)$$

– *if* $a \in \mathrm{Act}' \backslash \mathrm{Act}$, $R'_a(w',v') = (\alpha',\beta')$, *then*

$$(w,v') \in W^{\otimes} \text{ and } R_a^{\otimes}((w,w'),(w,v')) = (\alpha',\beta')$$

– *for any* $(w,w') \in W^{\otimes}$

  – *if* $p \in \mathrm{Prop} \cap \mathrm{Prop}'$, $V^{\otimes}((w,w'),p) = V(w,p) \sqcap\!\!\sqcap V'(w',p)$
  – *if* $p \in \mathrm{Prop} \backslash \mathrm{Prop}'$, $V^{\otimes}((w,w'),p) = V(w,p)$
  – *if* $p \in \mathrm{Prop}' \backslash \mathrm{Prop}$, $V^{\otimes}((w,w'),p) = V'(w',p)$

For institution $\boldsymbol{P}$, *the parallel composition of models* $M = (W, w_0, R, V) \in \mathsf{Mod}_{\boldsymbol{P}}(\Sigma)$ *and* $M' = (W', w'_0, R', V') \in \mathsf{Mod}_{\boldsymbol{P}}(\Sigma')$ *is the model* $M \otimes M' = (W^{\otimes}, (w_0, w'_0), R^{\otimes}, V^{\otimes})$ *with* $W^{\otimes}$, $R^{\otimes}$ *and* $V^{\otimes}$ *defined above.*

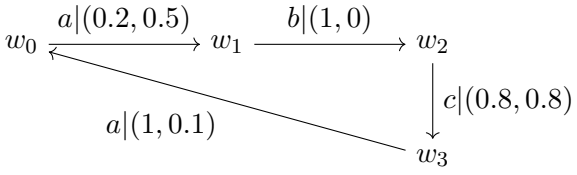
Other definitions of parallel constructor could be given. For instance, whenever an action a is in $\mathrm{Act} \cap \mathrm{Act}'$, $R_a(w,v) = (\alpha,\beta)$ and $R'_a(w',v') = (\alpha',\beta')$ then

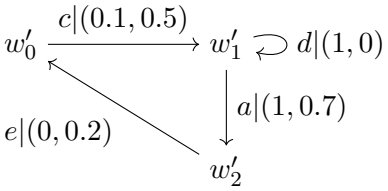
$$(v,v') \in W^{\otimes} \text{ and } R_a^{\otimes}((w,w'),(v,v')) = (\alpha \sqcap \alpha', \beta \sqcup \beta')$$

$$(v,w') \in W^{\otimes} \text{ and } R_a^{\otimes}((w,w'),(v,w')) = (\alpha,\beta)$$

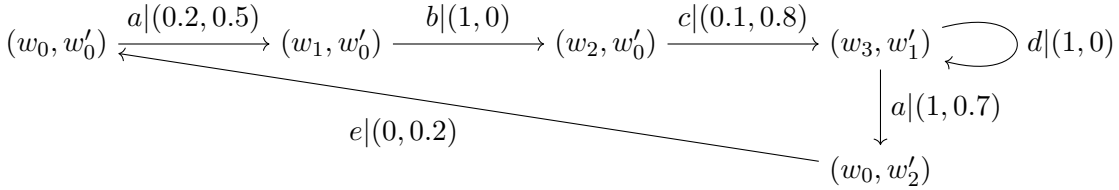
$$(w,v') \in W^{\otimes} \text{ and } R_a^{\otimes}((w,w'),(w,v')) = (\alpha',\beta')$$

Example 22. *Consider* $\boldsymbol{P}(\ddot{G})$ *and signatures* $\Sigma = \langle \{a,b,c\}, \varnothing \rangle$ *and* $\Sigma' = \langle \{c,d,e,a\}, \varnothing \rangle$. *Let the following left model be in signature* Σ *and the right one be in* Σ'.





The product of both models is the following one over signature $\langle \{a,b,c,d,e\}, \varnothing \rangle$.



Definition 41. *A signature morphism* $\sigma : \Sigma \to \Sigma'$ *between signatures* $\Sigma = \langle \mathrm{Act}, \mathrm{Prop} \rangle$ *and* $\Sigma' = \langle \mathrm{Act}', \mathrm{Prop}' \rangle$ *defines a constructor* $k_\sigma : \mathsf{Mod}(\Sigma') \to \mathsf{Mod}(\Sigma)$ *that maps any model* $M' \in \mathsf{Mod}(\Sigma')$ *to its reduct* $k_\sigma(M') = M'|_\sigma$.

If σ is bijective then k_σ is a relabelling constructor; if σ is injective then k_σ is a restriction constructor.

Definition 42. *Let* $\Sigma = \langle \mathrm{Act}, \mathrm{Prop} \rangle$, $\Sigma' = \langle \mathrm{Act}', \mathrm{Prop}' \rangle$ *be signatures and* $\Sigma'_D = \langle D, \mathrm{Prop}' \rangle$ *also be a signature such that* $D \subseteq Str(\mathrm{Act}')$ *is a finite subset and* $f : \Sigma \to \Sigma'_D$ *is a signature morphism. Then, the constructor* $k_f : \mathsf{Mod}(\Sigma'_D) \to \mathsf{Mod}(\Sigma)$ *maps a paraconsistent process* $M' \in \mathsf{Mod}(\Sigma'_D)$ *to its reduct* $\mathsf{Mod}(f)(M')$.

Note that the action refinement constructor described above is only defined for institution $\boldsymbol{P}(\mathscr{A})$ whose modalities are indexed by regular expressions.

5.2 Behavioural abstraction

Often models do not satisfy precisely each requirement of a specification. However they do satisfy the requirements up to some abstraction typically encoding an observation of the system, which for the problem at hands, might be sufficient.

This section is divided in two subsections. In the first one we present two binary relations to compare models. The next subsection focuses on defining behavioural abstraction specifications as well as presenting and proving some theorem regarding compositionality.

5.2.1 Relating models

We recall definitions of similarity and bisimilarity presented in [CMB22] generalized to suit multi-modalities.

Similarity

Definition 43. *Let* $\Sigma = \langle \mathrm{Act}, \mathrm{Prop} \rangle$ *be a signature and* $M = (W, R, V)$ *and* $M' = (W', R', V')$ *be* Σ*-models. A relation* $S \subseteq W \times W'$ *is a* simulation *if, for any* $p \in \mathrm{Prop}$, $a \in \mathrm{Act}$ *and* $(w, w') \in S$

- $R_a(w, v) = (\alpha, \beta)$ *then exists* $v' \in W'$, α', $\beta' \in A$ *such that* $R'_a(w', v') = (\alpha', \beta')$ *and* $(v, v') \in S$ *and* $(\alpha, \beta) \preccurlyeq (\alpha', \beta')$
- $V(w, p) \preccurlyeq V'(w', p)$

The next result generalizes [CMB22, Lemma 7] to suit multi-modalities.

Theorem 10. *Consider a signature* $\Sigma = \langle \mathrm{Act}, \mathrm{Prop} \rangle$ *and* Σ*-models* $M = (W', R', V')$, $M' = (W', R', V')$. *If* $S \subseteq W \times W'$ *is a simulation between models* M *and* M' *then, for any* $(w, w') \in S$ *and* $\varphi \in \mathsf{Sen}^+_{\boldsymbol{S}}(\Sigma)$

$$\Big(M, w \models_{\boldsymbol{S}} \varphi\Big) \preccurlyeq \Big(M', w' \models_{\boldsymbol{S}} \varphi\Big)$$

Where $\mathsf{Sen}^+_{\boldsymbol{S}}(\Sigma) (\subseteq \mathsf{Sen}_{\boldsymbol{S}}(\Sigma))$ *are all the sentences given by the following grammar:*

$$\varphi := \ p \,|\, \bot \,|\, \top \,|\, \varphi \wedge \varphi \,|\, \varphi \vee \varphi \,|\, \langle a \rangle \varphi$$

Proof. The proof is done by induction over the structure of sentences in $\mathsf{Sen}^{+}_{\boldsymbol{S}}(\Sigma)$. Cases $p \in \mathrm{Prop}$, $\bot$ and $\top$ are trivial:

$$(M, w \models_{\boldsymbol{S}} p) = V(w,p) \preccurlyeq V'(w',p) = (M', w' \models_{\boldsymbol{S}} p)$$

$$(M, w \models_{\boldsymbol{S}} \top) = (1,0) = (M, w \models_{\boldsymbol{S}} \top) \text{ hence } (M, w \models_{\boldsymbol{S}} \top) \preccurlyeq (M', w' \models_{\boldsymbol{S}} \top)$$

$$(M, w \models_{\boldsymbol{S}} \bot) = (0,1) = (M, w \models_{\boldsymbol{S}} \bot) \text{ hence } (M, w \models_{\boldsymbol{S}} \bot) \preccurlyeq (M', w' \models_{\boldsymbol{S}} \bot)$$

We next present the proof for "$\wedge$", which is similar to the proof for "$\vee$" with "$\wedge$" being replaced by "$\vee$" and "$\sqcap\!\!\sqcap$" with "$\sqcup\!\!\sqcup$".

$$\begin{aligned}
(M, w \models_{\boldsymbol{S}} \varphi \wedge \varphi') =& \{\text{def. of } \models_{\boldsymbol{S}}\} \\
& (M, w \models_{\boldsymbol{S}} \varphi) \sqcap\!\!\sqcap (M, w \models_{\boldsymbol{S}} \varphi') \\
\preccurlyeq& \{\text{induction hypothesis and Property 5.}\} \\
& (M', w' \models_{\boldsymbol{S}} \varphi) \sqcap\!\!\sqcap (M', w' \models_{\boldsymbol{S}} \varphi') \\
=& \{\text{def. of } \models_{\boldsymbol{S}}\} \\
& (M', w' \models_{\boldsymbol{S}} \varphi \wedge \varphi')
\end{aligned}$$

For any $a \in \mathrm{Act}$

$$\begin{aligned}
& (M, w \models_{\boldsymbol{S}} \langle a \rangle \varphi) \\
=& \{\text{def. of } \models_{\boldsymbol{S}}\} \\
& (\langle a^{+} \rangle (M, w, \varphi^{+}), [a^{+}](M, w, \varphi^{-})) \\
=& \{\text{def. of } \langle a^{+} \rangle \text{ and } [a^{+}]\} \\
& \left(\bigsqcup_{v \in W} (R^{+}_{a}(w,v) \wedge (M, v \models_{\boldsymbol{S}} \varphi)^{+}), \prod_{v \in W} (R^{+}_{a}(w,v) \rightharpoonup (M, v \models_{\boldsymbol{S}} \varphi)^{-}) \right) \\
\preccurlyeq& \{(\text{step } \star\star) + \text{Property 1.} + \text{Property 4.}\} \\
& \left(\bigsqcup_{v' \in W'} (R'^{+}_{a}(w',v') \wedge (M', v' \models_{\boldsymbol{S}} \varphi)^{+}), \prod_{v' \in W'} (R'^{+}_{a}(w',v') \rightharpoonup (M', v' \models_{\boldsymbol{S}} \varphi)^{-}) \right) \\
=& \{\text{def. of } \models_{\boldsymbol{S}}\} \\
& (M', w' \models_{\boldsymbol{S}} \langle a \rangle \varphi)
\end{aligned}$$

(step $\star\star$) Since $(w, w') \in S$ using the definition of simulation we have that for any $v \in W$ such that $R_a(w,v) = (\alpha, \beta)$ (i.e. $v \in R_a[w]$) there exists a $v' \in W'$, $\alpha', \beta' \in A$ such that $(v, v') \in S$ and $R'_a(w', v') = (\alpha', \beta')$ (i.e. $v' \in R'_a[w']$) with $(\alpha, \beta) \preccurlyeq (\alpha', \beta')$. Thus, $R^{+}_{a}(w,v) \preccurlyeq R'^{+}_{a}(w',v')$ and $R^{-}_{a}(w,v) \succcurlyeq R'^{-}_{a}(w',v')$. Similarly, using the induction hypothesis we have that $(M, w \models_{\boldsymbol{S}} \varphi)^{+} \preccurlyeq (M', w' \models_{\boldsymbol{S}} \varphi)^{+}$ and $(M, w \models_{\boldsymbol{S}} \varphi)^{-} \succcurlyeq (M', w' \models_{\boldsymbol{S}} \varphi)^{-}$. □

The following example illustrates a paraconsistent transition system in institution $\boldsymbol{S}$ where for some formulas Theorem 10 fails. More examples can be found in [CMB22].

Example 23. *Consider $\boldsymbol{S}(\ddot{\boldsymbol{G}})$ and the paraconsistent transition system $M = (\{w\}, R, V)$ and $M' = (\{w'\}, R', V')$ over the signature $\langle \{a\}, \{p, q\} \rangle$ where $R_a(w,w) = (0, 0.4) \preccurlyeq (1, 0.4) = R'_a(w', w')$ and $V(w,p) = (0.3, 0.8) \preccurlyeq (0.6, 0.7) = V'(w', p)$ and $V(w,q) = (0.1, 0.4) \preccurlyeq (0.2, 0.4) = V'(w', q)$.*



It is obvious that $S = \{(w, w')\}$ *is a simulation. We follow the example by presenting formulas* $\neg p$, $p \to q$, $[a]p$, $[\not a]p$ *and* $\langle \not a \rangle p$ *for which preservation of Theorem 10 does not hold, as expected.*

- *For* $\boldsymbol{\neg p}$ *we have that* $(M, w \models_{\boldsymbol{S}} \neg p) = /\!\!/V(w, p) = (0.8, 0.3)$ *and* $(M', w' \models_{\boldsymbol{S}} \neg p) = /\!\!/V'(w', p) = (0.7, 0.6)$. *That is,* $(M, w \models_{\boldsymbol{S}} \neg p) \not\preccurlyeq (M', w' \models_{\boldsymbol{S}} \neg p)$.

- *For* $\boldsymbol{p \to q}$*:*

$$(M, w \models_{\boldsymbol{S}} p \to q)^- = \min(V^+(w, p), V^-(w, q)) = \min(0.3, 0.4) = 0.3$$

$$(M', w' \models_{\boldsymbol{S}} p \to q)^- = \min(V'^+(w', p), V'^-(w', q)) = \min(0.60.4) = 0.4$$

  Since $0.3 < 0.4$ *then* $(M, w \models_{\boldsymbol{S}} p \to q) \not\preccurlyeq (M', w' \models_{\boldsymbol{S}} p \to q)$.

- *For* $\boldsymbol{[a]p}$*:*

$$(M, w \models_{\boldsymbol{S}} [a]p)^+ = (R_a^+(w, w) \to V^+(w, p)) = (0 \to 0.3) = 1$$

$$(M', w' \models_{\boldsymbol{S}} [a]p)^+ = (R_a'^+(w', w') \to V'^+(w', p)) = (1 \to 0.6) = 0.6$$

  Given that $1 > 0.6$ *we have that* $(M, w \models_{\boldsymbol{S}} [a]p) \not\preccurlyeq (M', w' \models_{\boldsymbol{S}} [a]p)$.

- *For* $\boldsymbol{[\not a]p}$*:*

$$(M, w \models_{\boldsymbol{S}} [\not a]p)^- = (R_a^-(w, w) \to (M, w \models_{\boldsymbol{S}} p)^+) = (0.4 \to 0.3) = 0.3$$

$$(M', w' \models_{\boldsymbol{S}} [\not a]p)^- = (R_a'^-(w', w') \to (M', w' \models_{\boldsymbol{S}} p)^+) = (0.4 \to 0.7) = 1$$

  Thus, $(M, w \models_{\boldsymbol{S}} [\not a]p) \not\preccurlyeq (M', w' \models_{\boldsymbol{S}} [\not a]p)$.

- *For* $\boldsymbol{\langle \not a \rangle p}$*:*

$$(M, w \models_{\boldsymbol{S}} \langle \not a \rangle p)^- = (\langle a^- \rangle (M, w, p^+)) = (R_a^-(w, w) \sqcap (M, w \models_{\boldsymbol{S}} p)^+) = (0.4 \sqcap 0.3) = 0.3$$

$$(M', w' \models_{\boldsymbol{S}} \langle \not a \rangle p)^+ = (R_a'^-(w', w') \sqcap (M', w' \models_{\boldsymbol{S}} p)^+) = (0.4 \sqcap 0.6) = 0.4$$

  Since $0.3 < 0.4$ *we have that* $(M, w \models_{\boldsymbol{S}} \langle \not a \rangle p) \not\preccurlyeq (M', w' \models_{\boldsymbol{S}} \langle \not a \rangle p)$.

Lemma 13. *Let* $M = (W, R, V)$, $M' = (W', R', V')$ *and* $M'' = (W'', R'', V'')$ *be models over a signature* $\Sigma = \langle \mathrm{Act}, \mathrm{Prop} \rangle$ *and* $S \subseteq W \times W'$, $S' : W' \subseteq W''$ *be two simulations.*
Then the composition of S *and* S' *is a simulation defined as:*

$$S' \circ S := \{(w, v) |\ \textit{exists } u \in W' \textit{ such that } (w, u) \in S \textit{ and } (u, v) \in S'\}$$

Proof. We will show that $(S' \circ S)$ is a simulation. Let $(w, v) \in (S' \circ S)$ and $u \in W'$ such that $(w, u) \in S$ and $(u, v) \in S'$. For any $p \in \text{Prop}$:

$$\begin{aligned} V(w,p) &\leqslant \{\text{def. of simulation } S\} \\ & V'(u,p) \\ &\leqslant \{\text{def. of simulation } S'\} \\ & V''(v,p) \end{aligned}$$

For any $a \in \text{Act}$, if $R_a(w, w') = (\alpha, \beta)$ then, by hypothesis, $(w, u) \in S$. So, using the definition of simulation S, we know that:

$$\text{exists } u' \in W' \text{ such that } (w', u') \in S \text{ and } R_a(w, w') \leqslant R'_a(u, u')$$

If $R'_a(u, u') = (\alpha', \beta')$ by hypothesis $(u, v) \in S$ so using the definition of simulation S' we have that:

$$\text{exists } v' \in W'' \text{ such that } (u', v') \in S' \text{ and } R'_a(u, u') \leqslant R''_a(v, v')$$

Therefore, if $R_a(w, w') = (\alpha, \beta)$ then there exists $v' \in W''$ such that $(w', v') \in (S' \circ S)$ and, by Property 6., $R_a(w, w') \leqslant R''_a(v, v')$. □

Definition 44. *Let $M \in |\mathsf{Mod}_{\boldsymbol{S}}(\Sigma)|$ for some signature $\Sigma \in |\mathsf{Sign}_{\boldsymbol{S}}|$.* Similarity *relation for model M is defined as:*

$$\lesssim \coloneqq \{(w, v) \mid (w, v) \in S \textit{ for some simulation } S \subseteq W \times W\}$$

Two states $w \in W$ and $v \in W'$ are similar, written as $w \lesssim v$, if there is a simulation $S \subseteq W \times W'$ and $(w, v) \in S$.

Lemma 14. *Similarity is a pre-order relation (reflexive and transitive).*

Proof. Let $M = (W, R, V)$ be a Σ-model. Then, for any states $w, v, u \in W$:

- $\lesssim$ is reflexive, that is $w \lesssim w$.
  The identity relation is a simulation, that is, $id \subseteq \lesssim$. Therefore, for any state $w \in W$ there exists a simulation id such that $(w, w) \in id$ (i.e. $w \lesssim w$).

- $\lesssim$ is transitive, that is, if $w \lesssim v$ and $v \lesssim u$ then $w \lesssim u$.
  If $w \lesssim v$ then there is a simulation S_1 such that $(w, v) \in S_1$ and if $v \lesssim u$ then there is a simulation S_2 such that $(v, u) \in S_2$. Let the simulation $S = S_2 \circ S_1$ be the composition of simulation S_1 and S_2 defined as:

  $$S_2 \circ S_1 = \{(w, u) \mid \text{exists } v \in W \text{ such that } (w, v) \in S_1 \text{ and } (v, u) \in S_2\}$$

  Given Lemma 13, $S_2 \circ S_1$ is a simulation and $(w, u) \in S$. By hypothesis there exists $u \in W'$ such that $(w, u) \in S_1$ and $(v, u) \in S_2$. Therefore, $w \lesssim u$.

□

Note that $\lesssim$ is not a partial order relation [BS81], i.e. reflexive, transitive and antisymmetric. The following example illustrates a simulation $S \subseteq \lesssim$ where $w \lesssim v$ and $v \lesssim w$ but $w \neq v$.

Example 24. *Let M be the model represented below over the signature $\langle\{a\},\{p\}\rangle$ with $V(w,p)$ equal to $V(v,p)$.*

$$a|(1,1) \circlearrowright w \xleftarrow{\quad a|(1,1) \quad} v \circlearrowright a|(1,1)$$

$S = \{(w,w), (v,v), (w,v), (v,w)\}$ is a simulation such that $(w,v) \in S$ and $(v,w) \in S$. However $w \neq v$.

Bisimilarity

Definition 45. *Let $\Sigma = \langle \mathrm{Act}, \mathrm{Prop}\rangle$ be a signature and $M = (W,R,V)$, $M' = (W',R',V')$ be Σ-models. A relation $B \subseteq W \times W'$ is a* bisimulation *between models M and M' if for any $(w,w') \in B$:*

- ***(Atom)*** *for any $p \in \mathrm{Prop}$, $V(w,p) = V'(w',p)$*
- ***(Zig)*** *for any $v \in W$ such that $R_a(w,v) = (\alpha,\beta)$, there is $v' \in W'$ such that $R'_a(w',v') = (\alpha,\beta)$ and $(v,v') \in B$*
- ***(Zag)*** *for any $v' \in W'$ such that $R'_a(w',v') = (\alpha,\beta)$, there is $v \in W$ such that $R_a(w,v) = (\alpha,\beta)$ and $(v,v') \in B$*

If there is a bisimulation B such that $(w,w') \in B$ for some $w \in W$, $w' \in W'$, we say that w and w' are bisimilar states and write $w \sim w'$.

Theorem 11. *Consider the signature $\Sigma = \langle \mathrm{Act}, \mathrm{Prop}\rangle$ and Σ-models $M = (W',R',V')$ and $M' = (W',R',V')$. If $B \subseteq W \times W'$ is a bisimulation between models $M = (W,R,V)$ and $M' = (W',R',V')$ then, for any $(w,w') \in S$ and $\varphi \in \mathsf{Sen}_{\boldsymbol{S}}(\Sigma)$*

$$\Big(M, w \models_{\boldsymbol{S}} \varphi\Big) \;=\; \Big(M', w' \models_{\boldsymbol{S}} \varphi\Big)$$

Proof. The proof is done by induction over the structure of sentences. Case $\bot$ is equal to the one presented for Theorem 10. Case $p \in \mathrm{Prop}$ is a direct consequence of the definition of bisimulation, for any $(w,w') \in B$ then, $(M, w \models_{\boldsymbol{S}} p) = V(w,p) = V'(w',p) = (M', w' \models_{\boldsymbol{S}} p)$. For cases $\neg$, $\wedge$, $\vee$, $\rightarrow$ and $\circ$ the proof is similar to some presented before where we start by using the definition of $\models_{\boldsymbol{S}}$ followed by the induction hypothesis and finally we use again the definition of $\models_{\boldsymbol{S}}$.
For cases $\langle a\rangle\varphi$, $[a]\varphi$, $\langle\!\!\not a\!\!\rangle\varphi$ and $[\not a]\varphi$ the proofs are similar: we illustrate all the cases by proving the theorem for $\langle a\rangle\varphi$:

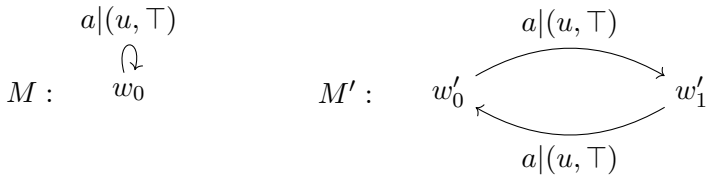
$$\begin{aligned}
&(M, w \models_{\boldsymbol{S}} \langle a\rangle\varphi)\\
=&\{\text{def. of } \models_{\boldsymbol{S}}\}\\
&(\langle a^+\rangle(M,w,\varphi^+), [a^+](M,w,\varphi^-))\\
=&\{\text{def. of } \langle a^+\rangle \text{ and } [a^+]\}\\
&\Big(\bigsqcup_{v\in W}(R_a^+(w,v) \sqcap (M, v \models_{\boldsymbol{S}} \varphi)^+), \bigsqcap_{v\in W}(R_a^+(w,v) \rightharpoonup (M, v \models_{\boldsymbol{S}} \varphi)^-)\Big)\\
=&\{\text{def. of bisimilarity } \textbf{(Zig)} + \textbf{(Zag)}\text{, the induction hypothesis and Property 1.}\}
\end{aligned}$$

$$\Big(\bigsqcup_{v' \in W'} (R_a'^{+}(w', v') \sqcap (M', v' \models_{\boldsymbol{S}} \varphi)^{+}), \bigsqcap_{v' \in W'} (R_a'^{+}(w', v') \rightharpoonup (M', v' \models_{\boldsymbol{S}} \varphi)^{-})\Big)$$
$$= \{\text{def. of } \models_{\boldsymbol{S}}\}$$
$$(M', w' \models_{\boldsymbol{S}} \langle a \rangle \varphi)$$

□

Theorem 11 of model invariance holds for models in institution $\boldsymbol{S}$, as proven above, but fails for models in institution $\boldsymbol{P}$. The following example, adapted from [JGMB21, Example 1], illustrates this fact.

Example 25. *Consider the models M and M' presented below over the signature $\langle \{a\}, \{p\} \rangle$ where $V(w, p) = (\top, \top)$ for any $w \in \{w_0, w_0', w_1'\}$.*



It is clear that $B = \{(w_0, w_0'), (w_0, w_1')\}$ is a bisimulation. However for $(w_0, w_0') \in S$ and sentence $\downarrow x.\langle a \rangle x$,

$$\begin{aligned} M, g, w_0 \models_{\boldsymbol{P}\downarrow} x.\langle a \rangle x &= M, g[x \mapsto w_0], w_0 \models_{\boldsymbol{P}} \langle a \rangle x \\ &= (\langle a^{+} \rangle (M, g, w_0, x^{+}),\ [a^{+}](M, g, w_0, x^{-})) \\ &= (u \wedge (M, g, w_0 \models_{\boldsymbol{P}} x)^{+},\ u \rightarrow (M, g, w_0 \models_{\boldsymbol{P}} x)^{-}) \\ &= (u \wedge \top, u \rightarrow \bot) = (u, \bot) \end{aligned}$$

and,

$$\begin{aligned} M', g, w_0' \models_{\boldsymbol{P}\downarrow} x.\langle a \rangle x &= M', g[x \mapsto w_0'], w_0' \models_{\boldsymbol{P}} \langle a \rangle x \\ &= (u \wedge (M', g, w_0' \models_{\boldsymbol{P}} x)^{+},\ u \rightarrow (M', g, w_0' \models_{\boldsymbol{P}} x)^{-}) \\ &= (u \wedge \bot, u \rightarrow \top) = (\bot, \top) \end{aligned}$$

That is, $(M, w_0 \models_{\boldsymbol{P}\downarrow} x.\langle a \rangle x) \neq (M', w_0' \models_{\boldsymbol{P}\downarrow} x.\langle a \rangle x)$.

Lemma 15. *Let $M = (W, R, V)$, $M' = (W', R', V')$ be $\langle \mathrm{Act}, \mathrm{Prop} \rangle$-models and $B \subseteq W \times W'$ a bisimulation. Then, the converse of B, written B°, is a bisimulation defined as:*

$$B^{\circ} := \{(w', w) \mid (w, w') \in B\}$$

Proof. For any bisimulation B between models M and M' such that for $w \in W$ and $w' \in W'$ we have that $(w, w') \in B$ then $(w', w) \in B^{\circ}$. We want to prove that B° is a bisimulation:

- for any $p \in \mathrm{Prop}$ since B is a bisimulation we have that $V(w, p) = V'(w', p)$. Thus, from the definition of **(Atom)** of B it follows **(Atom)** of B°.

- for any $v' \in W'$ such that $R'_a(w', v') = (\alpha, \beta)$, since B is a bisimulation by **(Zag)** we have that

$$\begin{aligned}&\text{exists } v \in W \text{ such that } R_a(w, v) = (\alpha, \beta) \text{ and } (v, v') \in B\\ &\Leftrightarrow \text{exists } v \in W \text{ such that } R_a(w, v) = (\alpha, \beta) \text{ and } (v', v) \in B^\circ\end{aligned}$$

  So from using **(Zag)** of B it follows **(Zig)** of B°.

- for any $v \in W$ such that $R_a(w, v) = (\alpha, \beta)$ since B is a bisimulation by **(Zig)** we have that

$$\begin{aligned}&\text{exists } v' \in W' \text{ such that } R'_a(w', v') = (\alpha, \beta) \text{ and } (v, v') \in B\\ &\Leftrightarrow \text{exists } v' \in W' \text{ such that } R'_a(w', v') = (\alpha, \beta) \text{ and } (v', v) \in B^\circ\end{aligned}$$

  From **(Zig)** of B it follows **(Zag)** of B°.

□

Lemma 16. *Let $M = (W, R, V)$, $M' = (W', R', V')$ and $M'' = (W'', R'', V'')$ be Σ-models, with $\Sigma = \langle \mathrm{Act}, \mathrm{Prop} \rangle \in |\mathsf{Sign}_{\mathcal{S}}|$. Given simulations $B \subseteq W \times W'$ and $B' \subseteq W' \times W''$, their composition is a bisimulation defined as:*

$$B' \circ B = \{(w, w'') | \textit{exists } w' \textit{ such that } (w, w') \in B \textit{ and } (w', w'') \in B'\}$$

Proof. We will prove that $(B' \circ B)$ is a bisimulation. Let B and B' be bisimulations and $(w, w'') \in (B' \circ B)$ such that there exists $w' \in W'$, $(w, w') \in B$ and $(w', w'') \in B'$ then,

- For any $p \in \mathrm{Prop}$ since B is a bisimulation and $(w, w') \in B$ then $V(w, p) = V'(w', p)$ also since B' is a bisimulation and $(w', w'') \in B'$ then $V'(w'p) = V''(w'', p)$. Thus, for any $(w, w'') \in (B' \circ B)$ we have that $V(w, p) = V''(w'', p)$.

- For any $a \in \mathrm{Act}$, if $R_a(w, v) = (\alpha, \beta)$, since B is a bisimulation and, by hypothesis, $(w, w') \in B$, then by **(Zig)** of B

$$\text{exists } v' \in W' \text{ such that } R'_a(w', v') = (\alpha, \beta) \text{ and } (v, v') \in B$$

  For any $R'_a(w', v') = (\alpha, \beta)$ because B' is a bisimulation and by hypothesis $(w', w'') \in B'$, using **(Zig)** of B'

$$\text{exists } v'' \in W'' \text{ such that } R''_a(w'', v'') = (\alpha, \beta) \text{ and } (v', v'') \in B'$$

  Therefore, for any $(w, w'') \in (B' \circ B)$ if $R_a(w, v) = (\alpha, \beta)$ then there exists a $v'' \in W''$ such that $R''_a(w'', v'') = (\alpha, \beta)$ and $(v, v'') \in (B' \circ B)$.

- For any $a \in \mathrm{Act}$ if $R''_a(w'', v'') = (\alpha, \beta)$, since B' is a bisimulation and $(w', w'') \in B'$

$$\text{exists } v' \in W' \text{ such that } R'_a(w', v') = (\alpha, \beta) \text{ and } (v', v'') \in B'$$

  Since $(w, w') \in B$ if $R'_a(w, v) = (\alpha, \beta)$ then by **(Zag)** of B

$$\text{exists } v \in W \text{ such that } R_a(w, v) = (\alpha, \beta) \text{ and } (v, v') \in B$$

  Therefore, for $(w, w'') \in (B' \circ B)$ if $R''_a(w'', v'') = (\alpha, \beta)$ then there exists $v \in W$ such that $R_a(w, v) = (\alpha, \beta)$ and $(v, v'') \in (B' \circ B)$.

□

Definition 46. *Let M be Σ-model for some signature Σ.* Bisimilarity *relation for model M is defined as:*

$$\sim_M := \{(w,v) |\, (w,v) \in B \text{ for some bisimulation } B \subseteq W \times W\}$$

A consequence of Lemma 15 and Lemma 16 is the following Lemma.

Lemma 17. *Bisimilarity $\sim_M$ is a equivalence relation (reflexive, symmetric and transitive).*

Proof. Let $M = (W, R, V)$ be a Σ-model and $\sim_M := \{(w,v) |\, (w,v) \in B$ for some $B \subseteq W \times W\}$ a relation. Then, for any states $w, v, u \in W$:

- $\sim_M$ is reflexive, that is, $w \sim_M w$
  The identity relation is a bisimulation, i.e. $id \subseteq \sim$. Therefore, for any state $w \in W$ there is a bisimulation, id, such that $w \sim_M w$.

- $\sim_M$ is symmetric, that is, if $w \sim_M v$ then $v \sim_M w$
  This is a consequence of Lemma-15. If there is a bisimulation B such that $(w,v) \in B$ then B° is a bisimulation such that $(v,w) \in B^\circ$.

- $\sim_M$ is transitive, that is, if $w \sim_M u$ and $u \sim_M v$ then $w \sim_M v$
  If $w \sim_M u$ then there is a bisimulation B_1 such that $(w,u) \in B_1$ and if $u \sim_M v$ then there is a bisimulation B_2 such that $(u,v) \in B_2$. Let $B_2 \circ B_1$ be the composition of B_1 and B_2, by Lemma 16, $B_2 \circ B_1$ is a bisimulation. Given that $(w,u) \in B_1$ and $(u,v) \in B_2$, by hypothesis, we have that $(w,v) \in (B_2 \circ B_1)$ thus, there is a bisimulation such that $w \sim_M v$.

□

Definition 47. *Given two models $M = (W, R, V)$ and $M' = (W', R', V')$ over the same signature. We say that M and M' are* behaviourally equivalent*, in symbols $M \equiv M'$, if and only if there is a bisimulation $B \subseteq W \times W'$, such that:*

- *for any $w \in W$ exists $w' \in W'$ such that $(w, w') \in B$*
- *for any $w' \in W'$ exists $w \in W$ such that $(w, w') \in B$*

Two processes $P = (W, w_0, R, V)$ and $P' = (W', w_0', R', V')$ are behavioural equivalent, $P \equiv P'$, if and only if $w_0 \sim w_0'$.

Corollary 2. *$\equiv$ is an equivalence relation (reflexive, symmetric and transitive).*

Proof. Let M, M', M'' be Σ-models with set of states W, W', W'', respectively, and P, P', P'' be Σ-processes with initial state w_0, w_0', w_0'', respectively.

- $\equiv$ is reflexive, that is, $M \equiv M$
  The identity relation, id, is a bisimulation such that for any state $w \in W$, $(w,w) \in id$. Therefore, any model M is behaviourally equivalent to itself. Similarly, for processes the identity relation, id, is a bisimulation so $w_0 \sim w_0$.

- $\equiv$ is symmetric, that is, if $M \equiv M'$ then $M' \equiv M$
  If $M \equiv M'$ then there is a bisimulation $B \subseteq W \times W'$ such that
  - for any $w \in W$, exists $w' \in W'$ such that $(w, w') \in B$ and
  - for any $w' \in W'$, exists $w \in W$ such that $(w, w') \in B$

  Consider the converse of B, denoted by B° and defined in Definition 15. The above conditions can be rewritten as:
  - for any $w \in W$, exists $w' \in W'$ such that $(w', w) \in B^\circ$ and
  - for any $w' \in W'$, exists $w \in W$ such that $(w', w) \in B^\circ$

  Hence, $M' \equiv M$.
  For processes, if $P \equiv P'$ then $w_0 \sim w_0'$, that is, there is a bisimulation B such that $(w_0, w_0') \in B$. Using the definiton of converse of B, we have that, there is a bisimulation B° such that $(w_0', w_0) \in B^\circ$, i.e. $w_0' \sim w_0$. Consequently, $P' \equiv P$.

- $\equiv$ is transitive, that is, if $M \equiv M'$ and $M' \equiv M''$ then $M \equiv M''$.
  If $M \equiv M'$ then there is a bisimulation $B \subseteq W \times W'$ such that
  - for any $w \in W$, exists $w' \in W'$ such that $(w, w') \in B$ and
  - for any $w' \in W'$, exists $w \in W$ such that $(w, w') \in B$

  And, if $M' \equiv M''$ then there is a bisimulation $B' \subseteq W' \times W''$ such that
  - for any $w' \in W'$, exists $w'' \in W''$ such that $(w', w'') \in B'$ and
  - for any $w'' \in W''$, exists $w' \in W'$ such that $(w', w'') \in B'$

  Using the definition of composition of bisimulations, Definition 16, there is a bisimulation $B' \circ B \subseteq W \times W''$ where
  - for any $w \in W$, exists $w'' \in W''$ such that $(w, w'') \in B' \circ B$ and
  - for any $w'' \in W''$, exists $w \in W$ such that $(w, w'') \in B' \circ B$

  Hence, $M \equiv M''$.
  For processes, if $P \equiv P'$ then there exists a bisimulation B such that $(w_0, w_0') \in B$ and if $P' \equiv P''$ then there exists a bisimulation B' such that $(w_0', w_0'') \in B'$. Using the definition of composition of bisimulations, there exists a bisimulation $B' \circ B$ such that (w_0, w_0''), that is, $w_0 \sim w_0''$. In conclusion, $P \equiv P''$.

□

5.2.2 Behavioural abstraction specifications

After defining what it means for models to be equivalent w.r.t. to a relation we define behavioural abstraction specifications and see how they can be combined with constructors.

Definition 48. *Let $M = (W, R, V)$ be a $\langle \text{Act}, \text{Prop} \rangle$-model in institution $\boldsymbol{S}$. The* quotient *of M with respect to $\sim_M$ is $M/{\sim_M} = (W/{\sim_M},\ R/{\sim_M},\ V/{\sim_M})$ with:*

- $W/{\sim_M} = \{[w]_{\sim_M} \mid w \in W\}$ *where* $[w]_{\sim_M} = \{v \in W \mid w \sim_M v\}$

- *For any* $a \in \text{Act}$, $(R/\sim_M)_a \colon W/\sim_M \times W/\sim_M \to A \times A$ *is defined as*

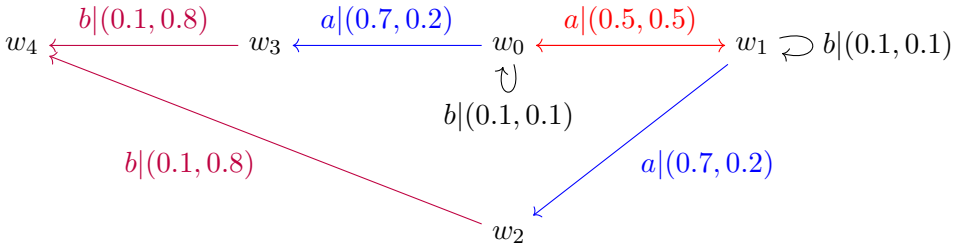
$$(R/\sim_M)_a\,([u]_{\sim_M}, [v]_{\sim_M}) \;=\; \bigsqcup_{\substack{u' \in [u]_{\sim_M} \\ v' \in [v]_{\sim_M}}} R_a(u', v')$$

- $V/\sim_M \colon W/\sim_M \times \text{Prop} \to A \times A$ *is given by* $V/\sim_M\,([w]_{\sim_M}, p) = V(w, p)$

For a $\langle \text{Act}, \text{Prop} \rangle$*-process* $P = (W, w_0, R, V)$ *the quotient with respect to* $\sim_M$ *is the process* $P/\sim_M = (W/\sim_M, [w_0]_{\sim_M}, R/\sim_M,\ V/\sim_M)$.

The next example defines the quotient of a model, w.r.t $\sim_M$, thus reducing the cardinality of its state space.

Example 26. *Let* $\ddot{\boldsymbol{G}}$ *be the underlying algebra,* $M = (\{w_0, w_1, w_2, w_3, w_4\}, R, V)$ *a model represented below over signature* $\langle \{a, b\}, \{p\} \rangle$



With $V(w_0, p) = V(w_1, p) = (1, 1)$, $V(w_2, p) = V(w_3, p) = (0, 0.8)$ *and* $V(w_4, p) = (0.8, 0.3)$. *Note that:*

- $[w_0]_{\sim_M} = [w_1]_{\sim_M} = \{w_0, w_1\}$. *Thus, for any* $i, j \in \{0, 1\}$ *and* $x \in \{a, b\}$,

$$R_x([w_i]_{\sim_M}, [w_j]_{\sim_M}) = R_x(w_i, w_j)$$

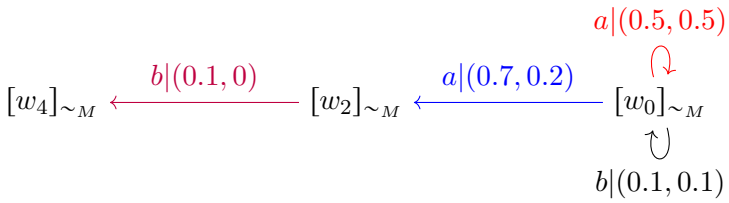
  and $V([w_0]_{\sim_M}, p) = (1, 1)$.

- $[w_2]_{\sim_M} = [w_3]_{\sim_M} = \{w_2, w_3\}$ *and* $V([w_2]_{\sim_M}, p) = (0, 0.8)$
- $[w_4]_{\sim_M} = \{w_4\}$ *and* $V([w_4]_{\sim_M}, p) = (0.8, 0.3)$

Finally we have that $R_a([w_0]_{\sim_M}, [w_2]_{\sim_M}) = R_a(w_0, w_2) = (0.7, 0.2)$ *and*

$$R_b([w_2]_{\sim_M}, [w_4]_{\sim_M}) = R_b(w_2, w_4) = (0.1, 0.8)$$

Accordingly, model $M/\sim_M$ *is:*



The notion of a quotient can be used to define a **behaviour** operator.

Definition 49. *Let SP be a Σ-specification and $\sim_M$ an equivalence relation between states of a Σ-model M. Then,*

- $Sig(\textbf{behaviour } SP) = \Sigma$
- $Mod(\textbf{behaviour } SP)(M) = Mod(SP)(M/\sim_M)$

Definition 50. *Let SP be a Σ-specification and $\equiv$ an equivalence relation between Σ-models. Then,*

- $Sig(\textbf{abstractor } SP \textbf{ w.r.t } \equiv) = \Sigma$
- $Mod(\textbf{abstractor } SP)(M) = \bigsqcup_{N \in [M]_\equiv} Mod(SP)(N)$

where $[M]_\equiv = \{N \in \mathsf{Mod}(\Sigma) \mid N \equiv M\}$.

Definition 51. *Let SP, SP' be Σ-specifications and $\equiv \subseteq Mod(\Sigma) \times Mod(\Sigma)$ be an equivalence relation. We write $SP \rightsquigarrow^{\equiv} SP'$ when SP' is a* simple abstractor implementation *of SP, that is, for any $M \in \mathsf{Mod}(\Sigma)$*

$$Mod(SP')(M) \preccurlyeq Mod(\textbf{abstractor } SP)(M)$$

The next definition combines abstractor implementation with constructor for paraconsistent specifications, generalizing [MBHM18, Definition 5].

Definition 52. *Let SP, SP_1, ..., SP_n be specifications over signatures Σ, Σ_1, ..., Σ_n, respectively, $k : \mathsf{Mod}(\Sigma_1) \times ... \times \mathsf{Mod}(\Sigma_n) \to \mathsf{Mod}(\Sigma)$ a constructor and $\equiv \subseteq \mathsf{Mod}(\Sigma) \times \mathsf{Mod}(\Sigma)$ an equivalence relation.*
We say that $(SP_1, ..., SP_n)$ is an abstractor implementation *of SP, in symbols $SP \rightsquigarrow_k^{\equiv} (SP_1, ..., SP_n)$ if for any $M_i \in \mathsf{Mod}(\Sigma_i)$,*

$$\bigsqcap_{i=1}^{n} Mod(SP_i)(M_i) \preccurlyeq Mod(\textbf{abstractor } SP)(k(M_1, ..., M_n))$$

Definition 53. *Let $k : \mathsf{Mod}(\Sigma_1) \times ... \times \mathsf{Mod}(\Sigma_n) \to \mathsf{Mod}(\Sigma)$ be a constructor and, for each $1 \leqslant i \leqslant n$, $\equiv_i$ an equivalence relation between Σ_i-models and $\equiv$ an equivalence relation between Σ-models. We say that a constructor k* preserves abstractions *$\equiv_i$ if for any M_i, $N_i \in \mathsf{Mod}(\Sigma_i)$ such that $M_i \equiv_i N_i$, $k(M_1, .., M_n) \equiv k(N_1, ..., N_n)$.*

Theorem 12. *The signature extension constructor k_{ext}, Definition 39, preserves behavioural equivalences, that is, if $M \equiv N$ then $k_{ext}(M) \equiv k_{ext}(N)$*

Proof. Let $\Sigma = \langle \mathrm{Act}, \mathrm{Prop} \rangle$ and $\Sigma' = \langle \mathrm{Act}', \mathrm{Prop}' \rangle$ be signatures with $\mathrm{Prop} \subseteq \mathrm{Prop}'$ and $\mathrm{Act} \subseteq \mathrm{Act}'$ and $M = (W_M, R_M, V_M)$ and $N = (W_N, R_N, V_N)$ be Σ-models such that there is a bisimulation B between them, that is $M \equiv N$. We want to show that B is also a bisimulation between $k_{ext}(M) = (W, R'_M, V'_M)$ and $k_{ext}(N) = (W_N, R'_N, V'_N)$, i.e. $k_{ext}(M) \equiv k_{ext}(N)$.
By the definition of bisimulation for any $a \in \mathrm{Act}$ and $p \in \mathrm{Prop}$ we have conditions **(Atom)**, **(Zig)** and **(Zag)** so we only have to concern ourselves with the propositions and actions that are not in signature Σ. For any $p \in \mathrm{Prop}' \backslash \mathrm{Prop}$ and $w \in W_M$, $v \in W_v$ such that $(w, v) \in B$ by definition of signature extension constructor $V'_M(w, p) = (0, 1) = V'_N(v, p)$, **(Atom)**. For any $a \in \mathrm{Act}' \backslash \mathrm{Act}$ and $w, w' \in W_M$ and $v, v' \in W_v$ such that $(w, v) \in B$ and $(v, v') \in B$ then by definition of the signature extension constructor we have $(R'_M)_a(w, w') = (0, 1)$ and $(R'_N)_a(v, v') = (0, 1)$, **(Zig)**+**(Zag)**. □

Theorem 13. *The parallel composition constructor, Definition 40, preserves behavioural equivalences, that is, if $M \equiv M'$ and $N \equiv N'$ then $M \otimes N \equiv M' \otimes N'$*

Proof. Let $M = (W_M, R_M, V_M)$, $M' = (W'_M, R'_M, V'_M)$ be Σ-models and $N = (W_N, R_N, V_N)$, $N' = (W'_N, R'_N, V'_N)$ be Σ'-models with $B_M \subseteq W_M \times W'_M$ and $B_N \subseteq W_N \times W'_N$ bisimulations. Consider $\sim \subseteq (W_M \times W_N) \times (W'_M \times W'_N)$ a relation between states of $M \otimes N = (W^{\otimes}, R^{\otimes}, V^{\otimes})$ and $M' \otimes N' = (W'^{\otimes}, R'^{\otimes}, V'^{\otimes})$ where $(w_1, w_2) \sim (w'_1, w'_2)$ if $(w_1, w'_1) \in B_M$ and $(w_2, w'_2) \in B_N$.
To show that $M \otimes N \equiv M' \otimes N'$ we have to prove that $\sim$ is a bisimulation. Without loss of generality, we will assume that $(w_1, w'_1) \in B_M$ and $(w_2, w'_2) \in B_N$, that is, $(w_1, w_2) \sim (w'_1, w'_2)$.

- for any $p \in \text{Prop} \cap \text{Prop}'$ and $(w_1, w_2) \in W^{\otimes}$ using the definition of parallel composition constructor
$$V^{\otimes}((w_1, w_2), p) = V_M(w_1, p) \barwedge V_N(w_2, p)$$
and for any $(w'_1, w'_2) \in W'^{\otimes}$,
$$V'^{\otimes}((w'_1, w'_2), p) = V'_M(w'_1, p) \barwedge V'_N(w'_2, p)$$
By hypothesis B_M is a bisimulation thus, if $(w_1, w'_1) \in B_M$ by **(Atom)**, $V_M(w_1, p) = V'_M(w'_1, p)$, similarly if $(w_2, w'_2) \in B_N$ then $V_N(w_2, p) = V'_N(w'_2, p)$. Therefore,
$$V^{\otimes}((w_1, w_2), p) = V'^{\otimes}((w'_1, w'_2), p)$$
If $p \in \text{Prop} \backslash \text{Prop}'$, then $V^{\otimes}((w_1, w_2), p) = V_M(w_1, p)$ and $V'^{\otimes}((w'_1, w'_2), p) = V'_M(w'_1, p)$, by hypothesis B_M is a bisimulation thus, $V^{\otimes}((w_1, w_2), p) = V'^{\otimes}((w'_1, w'_2), p)$. The case for $p \in \text{Prop}' \backslash \text{Prop}$ is similar.

- for any $a \in \text{Act} \cap \text{Act}'$ if $(R_M)_a(w_1, v_1) = (\alpha_1, \beta_1)$ and $(R_N)_a(w_2, v_2) = (\alpha_2, \beta_2)$, by the definition of parallel composition, $R^{\otimes}_a((w_1, v_1), (w_2, v_2)) = (\alpha_1 \sqcap \alpha_2, \beta_1 \sqcup \beta_2)$.
Using the definition of **(Zig)** of B_M, for $(w_1, w'_1) \in B_M$
$$\text{exists } v'_1 \in W'_M \text{ such that } (R'_M)_a(w'_1, v'_1) = (\alpha_1, \beta_1) \text{ and } (v_1, v'_1) \in B_M$$
Similarly, for $(w_2, w'_2) \in B_N$
$$\text{exists } v'_2 \in W'_N \text{ such that } (R'_N)_a(w'_2, v'_2) = (\alpha_2, \beta_2) \text{ and } (v_2, v'_2) \in B_N$$
For any v_1, v'_1 such that $(v_1, v'_1) \in B_M$ and $(v_2, v'_2) \in B_N$, by definition of $\sim$, $(v_1, v_2) \sim (v'_1, v'_2)$.
Also if $(R'_M)_a(w'_1, v'_1) = (\alpha_1, \beta_1)$ and $(R'_N)_a(w'_2, v'_2) = (\alpha_2, \beta_2)$ then, by the definition of $\otimes$, $R'^{\otimes}((w'_1, v'_1), (w'_2, v'_2)) = (\alpha_1 \sqcap \alpha_2, \beta_1 \sqcup \beta_2)$.
In conclusion, we proved that for $(w_1, w_2) \sim (w'_1, w'_2)$ if $R^{\otimes}_a((w_1, v_1), (w_2, v_2)) = (\alpha_1 \sqcap \alpha_2, \beta_1 \sqcup \beta_2)$ using definitions: **(Zig)** of B_M, **(Zig)** of B_N and $\otimes$, then
$$\text{exists } (v'_1, v'_2) \in W'^{\otimes} \text{ st } R'^{\otimes}((w'_1, v'_1), (w'_2, v'_2)) = (\alpha_1 \sqcap \alpha_2, \beta_1 \sqcup \beta_2) \text{ and } (v_1, v_2) \sim (v'_1, v'_2)$$
Case $a \in \text{Act} \backslash \text{Act}'$ is similar and proved resorting to the definition of **(Zig)** of B_M followed by the definition of $\otimes$, while case $a \in \text{Act}' \backslash \text{Act}$ uses the definition of **(Zig)** of B_N followed by $\otimes$.

– for any $a \in \text{Act} \cap \text{Act}'$ if $(R'_M)_a(w'_1, v'_1) = (\alpha_1, \beta_1)$ and $(R'_N)_a(w'_2, v'_2) = (\alpha_2, \beta_2)$ then $R'^{\otimes}_a((w'_1, v'_1), (w'_2, v'_2)) = (\alpha_1 \sqcap \alpha_2, \beta_1 \sqcup \beta_2)$.
By **(Zag)** of B_M, for $(w_1, w'_1) \in B_M$

$$\text{exists } v_1 \in W_M \text{ such that } (R_M)_a(w_1, v_1) = (\alpha_1, \beta_1) \text{ and } (v_1, v'_1) \in B_M$$

By **(Zag)** of B_N, for $(w_2, w'_2) \in B_N$

$$\text{exists } v_2 \in W_N \text{ such that } (R_N)_a(w_2, v_2) = (\alpha_2, \beta_2) \text{ and } (v_2, v'_2) \in B_N$$

Once again, for any v_1, v'_1 such that $(v_1, v'_1) \in B_M$ and $(v_2, v'_2) \in B_N$ then $(v_1, v_2) \sim (v'_1, v'_2)$. Furthermore, if $(R_M)_a(w_1, v_1) = (\alpha_1, \beta_1)$ and $(R_N)_a(w_2, v_2) = (\alpha_2, \beta_2)$ then,

$$R^{\otimes}_a((w_1, v_1), (w_2, v_2)) = (\alpha_1 \sqcap \alpha_2, \beta_1 \sqcup \beta_2)$$

We have proved that for $(w_1, w_2) \sim (w'_1, w'_2)$ if $R'^{\otimes}_a((w'_1, v'_1), (w'_2, v'_2)) = (\alpha_1 \sqcap \alpha_2, \beta_1 \sqcup \beta_2)$ by **(Zag)** of B_M, **(Zag)** of B_N and $\otimes$,

exists $(v_1, v_2) \in W^{\otimes}$ st $R^{\otimes}_a((w_1, v_1), (w_2, v_2)) = (\alpha_1 \sqcap \alpha_2, \beta_1 \sqcup \beta_2)$ and $(v_1, v_2) \sim (v'_1, v'_2)$

Case $a \in \text{Act} \backslash \text{Act}'$ is proved by using the definition of **(Zag)** of B_M followed by the definition of $\otimes$ and case $a \in \text{Act}' \backslash \text{Act}$ uses the definition of **(Zag)** of B_N followed by $\otimes$.

□

Theorem 14. *The relabelling, reduct and restriction constructors, Definition 41, preserve behavioural equivalences, that is, for any $\sigma : \Sigma \to \Sigma'$ and $M, N \in \mathsf{Mod}(\Sigma')$ if $M \equiv N$ then $M|_\sigma \equiv N|_\sigma$*

Proof. Let $M = (W, R, V)$ and $N = (W', R', V')$ be $\langle \text{Act}', \text{Prop}' \rangle$-models and $\sigma : \langle \text{Act}, \text{Prop} \rangle \to \langle \text{Act}', \text{Prop}' \rangle$ a signature morphism. Assume that $M \equiv N$, i.e. there is a bisimulation $B \subseteq W \times W'$.
The σ-reduct of M is $M|_\sigma = (W|_\sigma, R|_\sigma, V|_\sigma)$, where

- $W|_\sigma = W$
- for $w, v \in W$ and $a \in \text{Act}$, $(R|_\sigma)_a(w, v) = R_{\sigma(a)}(w, v)$
- for $w \in W$ and $p \in \text{Prop}$, $(V|_\sigma)(w, p) = V(w, \sigma(p))$

The σ-reduct of N is $N|_\sigma = (W'|_\sigma, R'|_\sigma, V'|_\sigma)$, where

- $W'|_\sigma = W'$
- for $w, v \in W'$ and $a \in \text{Act}$, $(R'|_\sigma)_a(w, v) = R'_{\sigma(a)}(w, v)$
- for $w \in W'$ and $p \in \text{Prop}$, $(V'|_\sigma)(w, p) = V'(w, \sigma(p))$

We will show that $B \subseteq W|_\sigma \times W'|_\sigma$ is a bisimulation between the states of $M|_\sigma$ and $N|_\sigma$, i.e. $M|_\sigma \equiv N|_\sigma$.
Given that B is a bisimulation we have that, for any $w, v \in W$ and $w', v' \in W'$ if $(w, w') \in B$,

- For any $p \in \text{Prop}$, by **(Atom)**, $V(w, \sigma(p)) = V'(w', \sigma(p))$, which using the definition of σ-reduct is the same as writing, $(V|_\sigma)(w, p) = (V'|_\sigma)(w', p)$.

- For any $a \in \text{Act}$, by **(Zig)**, if $R_{\sigma(a)}(w, v) = (\alpha, \beta)$ then there exists $v' \in W'$ such that $R'_{\sigma(a)}(w', v') = (\alpha, \beta)$ and $(v, v') \in B$, using the definition of σ-reduct is the same as writing, if $(R|_\sigma)_a(w, v) = (\alpha, \beta)$ then there exists $v' \in W'_\sigma$ such that $(R'|_\sigma)_a(w', v') = (\alpha, \beta)$ and $(v, v') \in B$.

- For $a \in \text{Act}$, by **(Zag)** and by the definition of σ-reduct, if $(R'|_\sigma)_a(w', v') = (\alpha, \beta)$ then there exists $v \in W|_\sigma$ such that $(R|_\sigma)_a(w, v) = (\alpha, \beta)$ and $(v, v') \in B$

□

Theorem 15. *The action refinement constructor k_f, Definition 42, preserves behavioural equivalence, that is, for any $f : \Sigma \to \Sigma'_D$ and paraconsistent processes $M, N \in \mathsf{Mod}(\Sigma'_D)$ if $M \equiv N$ then $k_f(M) \equiv k_f(N)$*

Proof. The proof is similar to the one for Theorem 14, with "σ" being replaced by "f" and slight alterations since the action refinement constructor is defined only for institution $\boldsymbol{P}$. □

The notion of a constructor preserving abstraction of some equivalence relations, Definition 53, is particularly important for the next theorem regarding vertical composition of abstractor specifications with constructors. The theorem is a generalization of [MBHM18, Theorem 3] for paraconsistent specifications.

Theorem 16. *Consider specifications SP, SP_1, ..., SP_n over signatures Σ, Σ_1, ..., Σ_n respectively, a constructor*

$$k : \mathsf{Mod}(\Sigma_1) \times ... \times \mathsf{Mod}(\Sigma_n) \to \mathsf{Mod}(\Sigma)$$

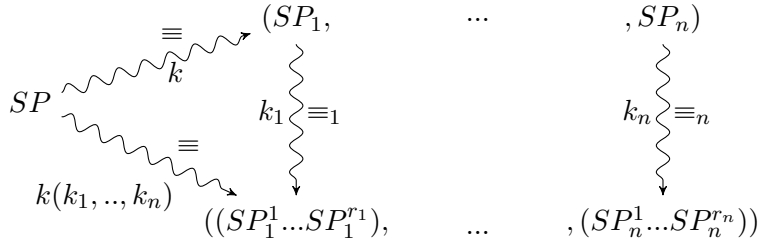
and an equivalence relation $\equiv \subseteq \mathsf{Mod}(\Sigma) \times \mathsf{Mod}(\Sigma)$ such that $SP \rightsquigarrow_k^{\equiv} (SP_1, ..., SP_n)$.
For each $i \in \{1, ..., n\}$ let $SP_i \rightsquigarrow_{k_i}^{\equiv_i} (SP_i^1, ..., SP_i^{r_i})$ with specifications $SP_i^1, ..., SP_i^{r_i}$ over signatures $\Sigma_i^1, ..., \Sigma_i^{r_i}$ respectively, constructors:

$$k_i : \mathsf{Mod}(\Sigma_i^1) \times ... \times \mathsf{Mod}(\Sigma_i^{r_i}) \to \mathsf{Mod}(\Sigma_i)$$

and equivalence relations $\equiv_i \subseteq \mathsf{Mod}(\Sigma_i) \times \mathsf{Mod}(\Sigma_i)$.
Suppose that k preserves the abstractions $\equiv_i$. Then,

$$SP \rightsquigarrow_{k(k_1,...,k_n)}^{\equiv} (SP_1^1, ..., SP_1^{r_1}, ..., SP_n^1, ..., SP_n^{r_n})$$

With $k(k_1, ..., k_n) : \mathsf{Mod}(\Sigma_1^1) \times ... \times \mathsf{Mod}(\Sigma_1^{r_1}) \times ... \times \mathsf{Mod}(\Sigma_n^1) \times ... \times \mathsf{Mod}(\Sigma_n^{r_n}) \to \mathsf{Mod}(\Sigma)$ being a constructor defined by the composition of constructors k_i and k.
Graphically,



Proof. For each $1 \leqslant i \leqslant n$ and for all $1 \leqslant j \leqslant r_i$, let $M_i^j \in \mathsf{Mod}(\Sigma_i^j)$.
For each i, by hypothesis, $SP_i \rightsquigarrow_{k_i}^{\equiv_i} (SP_i^1, ..., SP_i^{r_i})$, using the definition of abstractor implementation, Definition 52, and the definition of abstractor operator, Definition 50,

$$\begin{aligned}\prod_{j=1}^{r_i} Mod(SP_i^j)(M_i^j) &\preccurlyeq Mod(\textbf{abstractor } SP_i \textbf{ w.r.t } \equiv_i)(k_i(M_i^1, ..., M_i^{r_i})) \\ &= \bigsqcup_{N_i \in [k_i(M_i^1,...,M_i^{r_i})]_{\equiv_i}} Mod(SP_i)(N_i)\end{aligned}$$

Without loss of generality, for each i, let M_i be a Σ_i-model such that

$$\begin{aligned}\prod_{j=1}^{r_i} Mod(SP_i^j)(M_i^j) &\preccurlyeq Mod(\textbf{abstractor } SP_i \textbf{ w.r.t } \equiv_i)(k_i(M_i^1, ..., M_i^{r_i})) \\ &= Mod(SP_i)(M_i)\end{aligned} \tag{5.1}$$

By definition of M_i, we have that $M_i \in [k_i(M_i^1, ..., M_i^{r_i})]_{\equiv_i}$, that is, $M_i \equiv_i k_i(M_i^1, ..., M_i^{r_i})$.
Since k preserves abstraction $\equiv_i$,

$$k(M_1, ..., M_n) \equiv k(k_1(M_1^1, ..., M_1^{r_1}), ..., k_n(M_n^1, .., M_n^{r_n}))$$

Using the definition of composition of constructors, the equation above can be rewritten as:

$$k(M_1, ..., M_n) \equiv k(k_1, .., k_n)(M_1^1, ..., M_1^{r_1}, ..., M_n^1, .., M_n^{r_n}) \tag{5.2}$$

By hypothesis, $SP \rightsquigarrow_k^{\equiv} (SP_1, ..., SP_n)$, using Definition 52,

$$\begin{aligned}\prod_{i=1}^{n} Mod(SP_i)(M_i) &\preccurlyeq Mod(\textbf{abstractor } SP \textbf{ w.r.t } \equiv)(k(M_1, ..., M_n)) \\ &= \bigsqcup_{N \in [k(M_1,...,M_n)]_{\equiv}} Mod(SP)(N)\end{aligned} \tag{5.3}$$

By Equation 5.2, we have that:

$$k(k_1, .., k_n)(M_1^1, ..., M_1^{r_1}, ..., M_n^1, .., M_n^{r_n}) \in [k(M_1, ..., M_n)]_{\equiv} \tag{5.4}$$

Since $\equiv$ is transitive, Corollary 2, in Equation 5.3, $[k(M_1, ..., M_n)]_{\equiv}$ can be replaced with $[k(k_1, .., k_n)(M_1^1, ..., M_1^{r_1}, ..., M_n^1, .., M_n^{r_n})]_{\equiv}$.
Considering Equation 5.1, we know that for each i,

$$\prod_{j=1}^{r_i} Mod(SP_i^j)(M_i^j) \preccurlyeq Mod(SP_i)(M_i)$$

Since $\prod$ is monotone, Property 5.

$$\prod_{i=1}^{n} \prod_{j=1}^{r_i} Mod(SP_i^j)(M_i^j) \preccurlyeq \prod_{i=1}^{n} Mod(SP_i)(M_i) \tag{5.5}$$

With 5.4 and Equation 5.5 we can rewrite Equation 5.3 as:

$$\begin{aligned}\prod_{i=1}^{n} \prod_{j=1}^{r_i} Mod(SP_i^j)(M_i^j) \preccurlyeq & \prod_{N \in [k(k_1,..,k_n)(M_1^1,...,M_1^{r_1},...,M_n^1,..,M_n^{r_n})]_{\equiv}} Mod(SP)(N) \\ = & Mod(\textbf{abst. } SP \textbf{ w.r.t } \equiv)(k(k_1, .., k_n)(M_1^1, ..., M_1^{r_1}, ..., M_n^1, .., M_n^{r_n}))\end{aligned}$$

Thus, $SP \rightsquigarrow (SP_1^1 ... SP_1^{r_1}, ..., SP_n^1, ..., SP_n^{r_n})$. □

Chapter 6

Conclusions and future work

This dissertation started by generalizing, for multi-modalities, a modal paraconsistent logic introduced in [CMB22] that we denoted by $\boldsymbol{S}$. Then $\boldsymbol{S}$ served as the starting point to the introduction of $\boldsymbol{P}$ in Chapter 4, a logic to deal with paraconsistent processes where models have an initial state from which its behaviour unfolds. In $\boldsymbol{P}$ there is also a combination of modalities with regular expressions of actions and a combination of modal and hybrid connectives. Consequently, logic $\boldsymbol{P}$ is suitable to express not only more abstract and complex properties than $\boldsymbol{S}$, but also turns out to be closer to program development in Computer Science.

Another contribution of this dissertation was the formalization of these two paraconsistent logics as many-valued institutions. Such formalization was done to enable the introduction in Chapter 5 of a (paraconsistent) specification theory.

As seen in the Chapter 1, the applications of paraconsistent logics are extensive. Any problem in which inconsistent information can be found as well as a need to sensibly draw conclusions from such information can be modelled within some paraconsistent logic.

Since there are no doubts regarding paraconsistent logic relevancy, this thesis developed a specification theory based on an algebraic approach documented in [ST12]. Algebraic specification is an approach in which algebras are used to model systems and a list of properties (axioms) is used to describe a class of algebras. Having said that, listing such properties might be a complicated task which leads us to the development of structured and behavioural abstraction specifications. Such specifications can be combined in order to build formally more complex and concrete specifications.

Therefore, well know *CASL*-like operators were introduced so that paraconsistent specifications could be built in a structured manner. We also presented a palette of constructors defined to combine models from different signatures into a model of another signature. This constructors were based on work documented in [MBHM18].

Similarity and bisimilarity relations defined in [CMB22] were redefined here, with a few modifications. Such relations equipped us with the notion of what it means for two models to be behaviourally equivalent. This leads us to define behavioural abstract specifications. Instead of precisely satisfying each requirement these specifications focus on satisfying the requirement up to some abstraction.

Refinement results were also provided throughout Chapter 5 so that paraconsistent specifications could be refined leading to stepwise developments where, in every step, specifications get more concrete.

All of the work done in Chapter 5 provides us with numerous combinations to build specifications for systems with consistent, vague and, more importantly, inconsistent information.

From this dissertation, a number of topics for future research can be explored.

- Redefining bisimulation as a paraconsistent relation itself, that is, as a relation $B : W \times W \to A \times A$. This will result in states w and v of some model to be related by a pair of weights. One of the weights would represent the evidence degree of w and v being bisimilar and the other weight the evidence degree of not being so. Given that information in a paraconsistent model contains uncertainty and inconsistencies it seems natural to approach bisimilarity as a paraconsistent relation. Doing such work would lead to a less strong notion of bisimulation than the one used in Section 5.2.1. A similar agenda can be found in [Fan15] where a fuzzy bisimulation is defined for Gödel modal logic.
  Given such characterization, we intend to revisit behavioural abstract specifications found in Chapter 5.2.2 and redefining them for a notion of paraconsistent bisimulation.

- Exploration of the *Normal form theorem* for our paraconsistent institutions and specifications. The theorem reads as follows

  Theorem 17. *[ST12, Theorem 5.6.10] Let **INS** be semi-exact institution. For every specification SP built from flat specifications using union, translate and hiding, there is an equivalent specification of the form (SP, Φ)* **hide via** *σ (where Φ is finite provided the flat specifications involved in SP are finite).*

  This result is quite powerful as it states that specification built from flat specifications combined with specification-building operators can be reduced to a standard form. Once a specification is normalized it may be harder to understand it. However it may be helpful in other ways, for example, to prove equivalence of specifications.

- Another line of research worth to be explored in the future is the framing of the presented logics, in the light of the asymmetric combination of logics, where the features of a specific logic are developed on top of another one (see e.g. [NMBM16]). More precisely, we intend to introduce a a systematic way to build paraconsistent modal logics on top of a (base) logic, used to represent the states. The idea is to it following the lines of the temporalization [FG92], probabilization [Bal13] and of the hybridization of logics [MMDB11]. Finally, we intend to relate the achieved methods to convert classic into paraconsistent logics introduced in [dSCLD16].

References

[ACEGG91] J. Agustí-Cullell, F. Esteva, P. Garcia, and Ll Godo. Formalizing multiple-valued logics as institutions. In Bernadette Bouchon-Meunier, Ronald R. Yager, and Lotfi A. Zadeh, editors, *Uncertainty in Knowledge Bases*, pages 269–278. Springer, 1991.

[Afa19] Juan Afanador. Arguing ecosystem values with paraconsistent logics. *CoRR*, abs/1911.06367, 2019.

[Awo10] Steve Awodey. *Category Theory*. Oxford Logic Guides. OUP Oxford, 2010.

[Bal13] Pedro Baltazar. Probabilization of logics: Completeness and decidability. *Logica Universalis*, 7(4):403–440, 2013.

[Bar32] Frederic Bartlett. *Remembering: A Study in Experimental and Social Psychology*. Cambridge University Press, 1932.

[BEGR11] Félix Bou, Francesc Esteva, Lluís Godo, and Ricardo Oscar Rodríguez. On the minimum many-valued modal logic over a finite residuated lattice. *J. Log. Comput.*, 21(5):739–790, 2011.

[Bel77] Nuel D. Belnap. A useful four-valued logic. In J. Michael Dunn and George Epstein, editors, *Modern Uses of Multiple-Valued Logic*, volume 2 of *Modern Uses of Multiple-Valued Logic*, pages 5–37, Dordrecht, 1977. Springer Netherlands.

[BG80] Rod M. Burstall and Joseph A. Goguen. The semantics of clear, a specification language. In Dines Bjørner, editor, *Abstract Software Specifications, Proceedings of 1979 Copenhagen Winter School on Abstract Software Specification*, volume 86 of *Lecture Notes in Computer Science*, pages 292—332. Springer, 1980.

[BJ20] Jessica N. Burgeno and Susan L. Joslyn. The impact of weather forecast inconsistency on user trust. *Weather, Climate, and Society*, 12:679—-694, 2020.

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

[BS81] Stanley Burris and Hanamantagouda P. Sankappanavar. *A course in universal algebra*, volume 78 of *Graduate texts in mathematics*. Springer, 1981.

[CC86] Newton C. A. Costa and Walter A. Carnielli. On paraconsistent deontic logic. *Philosophia*, 16(3-4):293–305, 1986.

[CM14] Diana Costa and Manuel A. Martins. Inconsistencies in health care knowledge. In *16th IEEE International Conference on e-Health Networking, Applications and Services, Healthcom 2014, Natal-RN, Brazil, October 15-18, 2014*, pages 37–42. IEEE, 2014.

[CM17] Diana Costa and Manuel A. Martins. Paraconsistency in hybrid logic. *J. Log. Comput.*, 27(6):1825–1852, 2017.

[CMB22] Ana Cruz, Alexandre Madeira, and Luís Soares Barbosa. A logic for paraconsistent transition systems. In Andrzej Indrzejczak and Michal Zawidzki, editors, *10th International Conference on Non-Classical Logics. Theory and Applications*, volume 358 of *EPTCS*, pages 270–284, 2022.

[CMBnt] Juliana Cunha, Alexandre Madeira, and Luís Soares Barbosa. Structured specification of paraconsistent transition systems. Fundamentals of Software Engineering. Lecture Notes in Computer Science, (in print).

[Coh12] Paul Moritz Cohn. *Universal Algebra*, volume 6 of *Mathematics and Its Applications*. Springer Netherlands, 2012.

[Con07] Marcelo Coniglio. Logics of deontic inconsistency. *CLE e-Prints*, 7(4), 2007.

[CSdSF22] Hyghor Miranda Côrtes, Paulo Eduardo Santos, and João Inácio da Silva Filho. Monitoring electrical systems data-network equipment by means of fuzzy and paraconsistent annotated logic. *Expert Syst. Appl.*, 187:115865, 2022.

[DBPM04] Kenneth Deffenbacher, Brian Bornstein, Steven Penrod, and E. Kiernan McGorty. A meta-analytic review of the effects of high stress on eyewitness memory. *Law and Human Behavior*, 28:687—-706, 2004.

[dCK03] Newton C. A. da Costa and Décio Krause. Remarks on the applications of paraconsistent logic to physics. (avaliable at http://philsci-archive.pitt.edu/1566/), December 2003.

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

[Dia12] Răzvan Diaconescu. *Three decades of institution theory*. Studies in Universal Logic. Springer Basel, 2012.

[dSCLD16] Edelcio G. de Souza, Alexandre Costa-Leite, and Diogo H. B. Dias. On a paraconsistentization functor in the category of consequence structures. *Journal of Applied Non-Classical Logics*, 26(3):240–250, 2016.

[dSFLA10] João Inácio da Silva Filho, Germano Lambert-Torres, and Jair Minoro Abe. *Uncertainty Treatment Using Paraconsistent Logic - Introducing Paraconsistent Artificial Neural Networks*, volume 211 of *Frontiers in Artificial Intelligence and Applications*. IOS Press, 2010.

[EG01] Francesc Esteva and Lluis Godo. Monoidal t-norm based logic: Towards a logic for left-continuous t-norms. *Fuzzy Sets and Systems*, 124:271–288, 12 2001.

[EM45] Samuel Eilenberg and Saunders MacLane. General theory of natural equivalences. *Transactions of the American Mathematical Society*, 58(2):231–294, 1945.

[Fan15] Tuan-Fang Fan. Fuzzy bisimulation for gödel modal logic. *IEEE Trans. Fuzzy Syst.*, 23(6):2387–2396, 2015.

[FG92] Marcelo Finger and Dov M. Gabbay. Adding a temporal dimension to a logic system. *Journal of Logic, Language and Information*, 1(3):203–233, 1992.

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

[GW07] Dov M. Gabbay and John Woods, editors. *The Many Valued and Nonmonotonic Turn in Logic*, volume 8 of *Handbook of the History of Logic*. Elsevier, 2007.

[JGMB21] Manisha Jain, Leandro Gomes, Alexandre Madeira, and Luís Soares Barbosa. Towards a specification theory for fuzzy modal logic. In *International Symposium on Theoretical Aspects of Software Engineering, TASE 2021, Shanghai, China, August 25-27, 2021*, pages 175–182. IEEE, 2021.

[Koz94] Dexter Kozen. On action algebras. In J. van Eijck and A. Visser, editors, *Logic and Information Flow*, pages 78–88. MIT Press, 1994.

[Kra98] Marcus Kracht. On extensions of intermediate logics by strong negation. *Journal of Philosophical Logic*, 27(1):49–73, 1998.

[MBHM18] Alexandre Madeira, Luís S. Barbosa, Rolf Hennicker, and Manuel A. Martins. A logic for the stepwise development of reactive systems. *Theoretical Computer Science*, 744:78–96, 2018.

[MGR15] Chris MacKnight, Janice Graham, and Kenneth Rockwood. Factors associated with inconsistent diagnosis of dementia between physicians and neuropsychologists. *Journal of the American Geriatrics Society*, 47:1294–1299, 2015.

[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.

[MMBH15] Alexandre Madeira, Manuel A. Martins, Luís Soares Barbosa, and Rolf Hennicker. Refinement in hybridised institutions. *Formal Aspects Comput.*, 27(2):375–395, 2015.

[MMDB11] Manuel A. Martins, Alexandre Madeira, Razvan Diaconescu, and Luís Soares Barbosa. Hybridization of institutions. In Andrea Corradini, Bartek Klin, and Corina Cîrstea, editors, *Algebra and Coalgebra in Computer Science - 4th International Conference, CALCO 2011, Winchester, UK, August 30 - September 2, 2011. Proceedings*, volume 6859 of *Lecture Notes in Computer Science*, pages 283–297. Springer, 2011.

[MNM16] Alexandre Madeira, Renato Neves, and Manuel A. Martins. An exercise on the generation of many-valued dynamic logics. *J. Log. Algebraic Methods Program.*, 85(5):1011–1037, 2016.

[NMBM16] Renato Neves, Alexandre Madeira, Luís Soares Barbosa, and Manuel A. Martins. Asymmetric combination of logics is functorial: A survey. In Phillip James and Markus Roggenbach, editors, *Recent Trends in Algebraic Development Techniques - 23rd IFIP WG 1.3 International Workshop, WADT 2016, Gregynog, UK, September 21-24, 2016, Revised Selected Papers*, volume 10644 of *Lecture Notes in Computer Science*, pages 39–55. Springer, 2016.

[Sri90] Yellamraju V. Srinivas. Algebraic specification: syntax, semantics, structure. Technical Report 90-15, University of California, Irvine, June 1990.

[ST12] Donald Sannella and Andrzej Tarlecki. *Foundations of Algebraic Specification and Formal Software Development*. Monographs in Theoretical Computer Science. An EATCS Series. Springer, 2012.

[TLAdSF11] Cláudio Rodrigo Torres, Germano Lambert-Torres, Jair Minoro Abe, and João Inácio da Silva Filho. The sensing system for the autonomous mobile robot emmy III. In *FUZZ-IEEE 2011, IEEE International Conference on Fuzzy Systems, Taipei, Taiwan, 27-30 June, 2011, Proceedings*, pages 2928–2933. IEEE, 2011.

[Wal99] Jeremy Waldron. *Law and Disagreement*. Oxford University Press, 1999.

[WWN14] Retno Tri Wahyuni, Yusmar Palapa Wijaya, and Dini Nurmalasari. Design of wireless sensor network for drainage monitoring system. *Innovative Systems Design and Engineering*, 5:6–13, 2014.