

MINIMUM MODELS: REASONING AND AUTOMATION

A DISSERTATION SUBMITTED TO THE UNIVERSITY OF MANCHESTER
FOR THE DEGREE OF MASTER OF SCIENCE
IN THE FACULTY OF ENGINEERING AND PHYSICAL SCIENCES

2009

By
Jesús Héctor Domínguez Sánchez
School of Mathematics

Contents

| | |
|---|-----------|
| Abstract | 4 |
| Declaration | 5 |
| Copyright | 6 |
| Acknowledgements | 7 |
| 1 Introduction | 8 |
| 2 Minimum models | 13 |
| 2.1 Motivation | 13 |
| 2.2 Minimum models | 14 |
| 2.2.1 Models and satisfaction | 14 |
| 2.2.2 Observations and minimum models | 15 |
| 2.2.3 Characterising minimum models | 16 |
| 2.3 Minimum model examples | 17 |
| 2.3.1 A blocks world example | 18 |
| 2.3.2 A boolean circuit example | 20 |
| 3 Minimum model reasoning | 26 |
| 3.1 \mathcal{O} -closedness and \mathcal{O} -independence | 26 |
| 3.2 A restricted proof system | 27 |
| 3.3 Discussion: Difficulty of minimum model reasoning | 33 |
| 3.4 A proof system that relaxes \mathcal{O} -closedness | 45 |
| 3.5 A class of unrestricted proof systems | 58 |
| 3.5.1 Discussion on recursive admissible relations | 60 |

| | | |
|----------|--|------------|
| 4 | Examples in Isabelle | 76 |
| 4.1 | An overview of Isabelle | 76 |
| 4.2 | Implementing Υ^M using an example theory | 78 |
| 4.3 | Implementing $\Upsilon^{M^2}(\mathcal{I}, \gamma)$ using an example theory | 96 |
| 5 | Conclusions | 113 |
| | Bibliography | 121 |

Abstract

This report presents a series of proof systems for reasoning in a class of models, called minimum models. The report describes the semantic concept of minimum models and provides examples. Soundness, completeness and effectiveness for each of the presented proof systems is discussed, as well as the main difficulties involved when building a proof system for minimum model reasoning. The report presents implementations in the proof assistant Isabelle of two of the described proof systems by means of toy Blocks World theories.

Declaration

No portion of the work referred to in this dissertation has been submitted in support of an application for another degree or qualification of this or any other university or other institute of learning.

Copyright

- i. Copyright in text of this dissertation rests with the author. Copies (by any process) either in full, or of extracts, may be made **only** in accordance with instructions given by the author. Details may be obtained from the appropriate Graduate Office. This page must form part of any such copies made. Further copies (by any process) of copies made in accordance with such instructions may not be made without the permission (in writing) of the author.
- ii. The ownership of any intellectual property rights which may be described in this dissertation is vested in the University of Manchester, subject to any prior agreement to the contrary, and may not be made available for use by third parties without the written permission of the University, which will prescribe the terms and conditions of any such agreement.
- iii. Further information on the conditions under which disclosures and exploitation may take place is available from the Head of the School of Mathematics.

Acknowledgements

I would like to thank David Rydeheard and Howard Barringer for the opportunity they gave me in carrying out this project on a computer science topic and for their guidance in selecting the route the dissertation took.

I would like to thank the Msc. in Mathematical Logic teaching staff, as they provided me with all the formal training I needed to carry out this dissertation. I hope this dissertation, although with its humble results, is at the level of their standards.

Finally, but not less important, I would like to thank CONACYT for the opportunity they gave me in studying abroad. Without their financial support, studying at the University of Manchester could have been very difficult for me.

Chapter 1

Introduction

This report describes a series of proof systems for reasoning in a class of models, called minimum models. Implementations in Isabelle of some those proof systems are presented.

The main concept on this report is that of a minimum model. The concept of a minimum model, as presented in [1] and [2], appears naturally when modelling systems whose states can be described by positive “facts” or “observations” (many applications in Artificial Intelligence use this kind of representation for states, see [6], [9], [18]) and where observations not stated or not deducible from the rules of the system are intended not to hold. This interpretation of “absence as negation” is very close to the idea of circumscription in Artificial Intelligence, for example [8], [15]. The concept of minimum model seems to be a very natural way of representing this “absence as negation” interpretation on the state descriptions.

On this report, the concept of minimum model that will be used is the one presented in [1] and [2]. The report will explain the concept, together with examples on its usage. As stated in [2], other authors, for example [5], [15], [20] and [17], have considered orders on models and minimum, or minimal, models in similar contexts, but the logical settings and definitions of order differ from those presented in [1] and [2].

After the main concept is presented, the report starts the exploration on how the semantic concept of minimum model entailment could be mapped into its proof theory equivalent. Three proof systems are presented that capture minimum model entailment under certain conditions. The first proof system is a very restricted proof system that works for ideal state descriptions, i.e. states where all positive observations are specified. Even though its simplicity, the first proof

system serves the purpose of pointing out where the main difficulty resides when building a proof system for minimum model reasoning that is sound, complete and effective, for unrestricted state descriptions and general theories. The report discusses these difficulties in detail and then goes to describe two more proof systems that try to compromise between how restrictive is the state description (by generalising the state description to non-ideal cases) and which properties of the proof system are still attainable (soundness, completeness and effectiveness) when the restriction on the state description is relaxed; but taking into account that every proof system must conserve at least soundness and effectiveness, as we are interested in its correctness and automation as its most important properties.

The need to implement the proof system on a computer led us to the task of selecting a suitable proof environment to implement it. I now present a short overview of the area of automated reasoning together with short descriptions of some of the available provers.

The objective of automated reasoning is the construction of computer programs that assist in solving problems that require reasoning [23]. Reasoning in this context is understood as *logical* reasoning, i.e. logical reasoning is *not* concerned with conclusions that have a good chance of being true when the facts are true, like in common-sense reasoning. There are two main branches in automated reasoning. The first branch is called automated theorem proving which is concerned with the construction of fully automated theorem provers, i.e. without the intervention of humans while searching for a proof. The second branch is called interactive theorem proving which is a combination of automation with human control and guidance, i.e. the ideal is to leave the subtle parts of the proof to humans, and leave the obvious or tedious parts to the automated prover.

Some provers in the automated paradigm are the following:

- Prover9 [10] is an automated theorem prover for first-order and equational logic. It is a successor of the Otter prover. Prover9 uses the inference techniques of binary resolution, hyperresolution, UR-resolution, and binary paramodulation. Other features are conversion from first-order formulas to clauses, weighting, term ordering, forward and back demodulation, answer literals, forward and back subsumption, evaluable functions and predicates and the hints strategy.
- Vampire [16] is an automatic theorem prover for first-order classical logic

developed at the Computer Science department of the University of Manchester. It is a resolution and paramodulation-based theorem prover. It implements several versions of binary resolution and superposition with literal selection for logics with and without equality. A number of standard redundancy criteria and simplification techniques are used for pruning the search space: subsumption, tautology deletion, simplification by unit equalities, branch demodulation and subsumption resolution.

Some provers in the interactive paradigm are the following (this account of both provers can be found in [7]):

- PVS Verification System [19] is being developed at SRI International Computer Science Laboratory. PVS is written in Lisp. PVS has been applied in serious applications, for example, the specification and design of fault-tolerant flight control systems. PVS implements classical typed higher-order logic, extended with predicate subtypes and dependent types. PVS has built-in types with hard-coded standard operations on them. Type constructors are available to build complex types, i.e. function types, product types, records (labelled products) and recursively defined abstract data types. PVS has a parametrised module system: a specification is divided in theories and each theory can be parametrised with both types and values. Many of the language constructs in PVS are hard-coded into the language and the prover.

PVS represents theorems using the sequent calculus. Every sub-goal consists of a list of assumptions and a list of conclusions. PVS has commands for carrying out one step in a proof and automated tools that combine various proof steps for solving the goal. The language for proof strategies in PVS contains constructs for sequencing, backtracking, branching, let-binding and recursion. Powerful decision procedures were added to PVS; however, the procedures are part of the kernel, which makes the kernel large and complex; this produces soundness problems sometimes.

- Isabelle [13] is being developed in Cambridge, UK, and in Munich. Isabelle is written in ML. Among the applications of Isabelle are: formalising mathematics (including semantics), logical investigations, program development, specification languages, and verification of programs or systems. Isabelle

has a meta-logic, which is a fragment of higher-order logic. Formulae in the meta-logic are built using implication \implies , universal quantification \bigwedge and equality \equiv . Object logics are represented in the meta-logic. Examples of object logics are first-order logic, Zermelo-Fraenkel set theory and (typed) higher-order logic.

The specification language of Isabelle is inspired by functional programming languages. A specification is divided in theories but the specification of theories does not allow parametrisation. Isabelle has constructs for defining complex types, i.e. function types, product types and recursively defined abstract data types. Isabelle automatically generates induction principles for each declared recursive datatype. Also, Isabelle allows inductive and coinductive function definitions. The syntax in Isabelle is extensible.

In Isabelle, every goal consists of a list of assumptions and one conclusion. The basic proof method of Isabelle is resolution with higher-order unification. Resolution can be applied by using one-step commands in the proof or by combining tactics. Isabelle has also automated tools for solving goals. The language for proof strategies (tactics) in Isabelle is ML, which is a complete programming language: there are functions for sequencing, backtracking, branching and these functions can be combined to program more complex proof strategies.

In this report, we will use Isabelle to implement the proof systems that will be presented later on this report. The reason is that the presence of a meta-logic in Isabelle allows the representation of non-standard inference rules (i.e. inference rules that are not part of a classical first-order logic proof system), as all the proof systems in this report will require the inclusion of a non-standard inference rule. Most of the automated theorem provers do not allow this kind of extension as they work mostly on first-order logic; the interactive theorem provers are usually the ones that allow this kind of extension. Another requirement is that we need access to the low level programming library in order to build complex proof strategies, Isabelle is very flexible on this respect as it uses the programming language ML to code general proof strategies; this requirement will be necessary as we will need to represent a decision procedure inside one of the proof systems on this report.

The report is structured as follows. Chapter 2 provides a mathematical definition of minimum models, a characterisation of minimum models in terms of

classical entailment, a characterisation of minimum model entailment in terms of classical entailment and some examples on the usage of minimum models. Chapter 3 describes proof systems for minimum model reasoning. The chapter introduces various techniques for defining proof systems. One technique is assuming some property on the state descriptions, like being closed with respect to the observations derivable from the theory (Section 3.2). Another technique is allowing interactions of observations in a part of the theory that has some nice property, like being decidable (Section 3.4). Two more techniques involve the analysis of the form of the sentences in the theory and searching for finite models of the theory (Section 3.5). The chapter also discusses at Section 3.3 the difficulties involved in constructing a proof system for minimum model reasoning that is sound, complete and effective, when we drop any restrictions on the state descriptions and allow general theories. Chapter 4 presents implementations in the proof assistant Isabelle of the proof systems of Sections 3.2 and 3.4 by means of toy versions of the Blocks World Theory. The chapter describes the technical details involved in the Isabelle implementation of the proof systems. Also, the chapter shows examples of minimum model proofs in the toy Blocks World theory, carried out in Isabelle. Finally, Chapter 5 summarizes the achievements of the research, explains the biggest problems of the approach taken on this report for building the proof systems, and provides an evaluation of the work done on this report and some areas of improvement as future work.

Chapter 2

Minimum models

This chapter describes the main concept on this report: minimum models. Section 2.1 provides a motivation for the usage of minimum models. Section 2.2 defines the concept mathematically and provides some mathematical results that will be useful later on. Finally, Section 2.3 presents some examples on the usage of minimum models.

2.1 Motivation

A common way of representing the state of some system is to specify which facts or observations hold in the system. The observations are usually first-order formulae, in a very simple form, like atomic formulae. Such descriptions are commonly used in applications of Artificial Intelligence [6], [9], [18].

Models of these state descriptions are not simply standard models of the first-order formulae in the state description. The reason is that these state descriptions are intended to be a complete description, in the sense that independent observations about the system that are not included in the description are assumed not to hold. Therefore, a standard model cannot capture this idea because a standard model requires that the negation of the observations that do not hold in the system be included in the state description in order for the state to capture completely the observations that hold in the standard model, or otherwise, there could be a lot of standard models that satisfy the state description but where observations that do not hold in the system are actually true in the standard model. This idea is related to the “absence as negation” interpretation and to the notion of “circumscription” in Artificial Intelligence, for example [8], [15].

Why not just add the negations of the observations that do not hold to the state description? If the set of observations that do not hold is finite, we could just add those negative observations to the state description so as to circumscribe the state description completely (as we would do in a standard interpretation), but in general, the set of observations that do not hold could be infinite or very large, and so it may not be practical to do such thing when a simple state description suffices. Another reason for not specifying negative observations is that applications of systems whose states are representable by facts usually involve the state description being changed dynamically due to the execution of actions. These actions can be stated in a revision-based logic, in which the actions are described by stating the observations which are added to or removed from the state description, based on some pre-condition [1], [6]. So, if the state has all the negations of the observations that do not hold, the description of the actions in the revision-based logic could become very complex, limiting the usefulness of the representation. And finally, the situation becomes even more complex if there is a theory (i.e. the rules of the world) relating observational facts.

The notion of minimum model that will be presented in this section is an approach to capture this logical account on observational descriptions of a system, and the “absence as negation” interpretation under the presence of a theory relating observational facts.

2.2 Minimum models

The account of minimum models that will be presented in this section (Subsections 2.2.1, 2.2.2 and 2.2.3) is exactly as the one found in the short note [2]. I include the account as it is in the note because I will require all these results in the examples of Section 2.3 and to prove results in Chapter 3, and also, because the account of minimum models given in [2] is not standard in the literature.

2.2.1 Models and satisfaction

Let \mathcal{L} be a typed, first-order signature. We will denote by $Sen(\mathcal{L})$ the set of sentences built from the signature \mathcal{L} .

Set-theoretic models are standard:

Definition 2.1. *Let \mathcal{L} be a first-order, typed signature. An \mathcal{L} -model α is an*

allocation of a set $\alpha(T)$ for each type T of \mathcal{L} . A function $\alpha(f) : \alpha(T_1) \times \dots \times \alpha(T_n) \rightarrow \alpha(T)$ for each function symbol $f : T_1 \times \dots \times T_n \rightarrow T$ of \mathcal{L} , a relation $\alpha(r) \subseteq \alpha(T_1) \times \dots \times \alpha(T_n)$ for each predicate symbol $r : T_1 \times \dots \times T_n$ of \mathcal{L} , and an element $\alpha(c) \in \alpha(T)$ for each constant c of type T in \mathcal{L} .

The equality symbol is treated as equality of elements in models. If the signature allows enumeration types, these are interpreted strictly: Each declared constant c of an enumeration type T denotes a distinct element of $\alpha(T)$ and this exhausts $\alpha(T)$ i.e. for all x in $\alpha(T)$, there is a constant c of type T such that $\alpha(c) = x$.

The definition of the interpretation of sentences $\psi \in \text{Sen}(\mathcal{L})$ in a model is standard. A first-order sentence ψ is satisfied in a model α , written $\alpha \models \psi$ if and only if the interpretation of ψ in α is true. We extend this to sets of sentences Ψ , i.e. $\alpha \models \Psi$ if and only if for all $\psi \in \Psi$, $\alpha \models \psi$.

For a set of sentences Ψ and sentence ψ , we write $\Psi \models \psi$ if and only if for all \mathcal{L} -models α , $\alpha \models \Psi \implies \alpha \models \psi$.

For first-order, typed theory W , we say α is a model of W (or α is a W -model) when, for all $\psi \in W$, $\alpha \models \psi$. We write $\Psi \models_W \psi$ if and only if for all W -models α , if $\alpha \models \Psi$ then $\alpha \models \psi$. For typed theories with equality, the standard rules for equality (equivalence and substitution) are assumed. For types described as enumeration types, the distinctness and exhaustiveness of the enumerations are assumed in the theory.

2.2.2 Observations and minimum models

Let \mathcal{L} be a typed, first-order signature. Let \mathcal{O} be a set of \mathcal{L} -sentences which we interpret as a collection of “possible observations or facts” about a model (\mathcal{O} may be empty).

Definition 2.2. Let $W \subseteq \text{Sen}(\mathcal{L})$ be a theory and $\Delta \subseteq \mathcal{O}$ a state description. A W -model α satisfies Δ iff $\alpha \models \Delta$. Let $\text{Mod}_W(\Delta)$ be the class of all W -models that satisfy Δ .

We define a pre-order \lesssim on $\text{Mod}_W(\Delta)$ by

$$\alpha \lesssim \beta \text{ iff } \forall \varphi \in \mathcal{O} \quad \alpha \models \varphi \implies \beta \models \varphi$$

Minimum models in $\text{Mod}_W(\Delta)$ will be those models $\alpha \in \text{Mod}_W(\Delta)$ such that

for all models $\beta \in \text{Mod}_W(\Delta)$, $\alpha \lesssim \beta$, that is:

$$\forall \varphi \in \mathcal{O} \quad \alpha \models \varphi \implies \forall \beta \in \text{Mod}_W(\Delta) \quad \beta \models \varphi$$

Minimum models are “observationally equivalent”, i.e. for two minimum models α and β ,

$$\forall \varphi \in \mathcal{O} \quad \alpha \models \varphi \text{ iff } \beta \models \varphi$$

However, minimum models may not be isomorphic and they may not exist - it depends on the theory W , the choice of observations \mathcal{O} and the set Δ (see Corollary 2.1).

We introduce another satisfaction relation, \models_W , defined using minimum models:

Definition 2.3. Let $W \subseteq \text{Sen}(\mathcal{L})$ be a theory, $\mathcal{O} \subseteq \text{Sen}(\mathcal{L})$ a set of observations, $\Delta \subseteq \mathcal{O}$ a state description, and $\psi \in \text{Sen}(\mathcal{L})$. Write,

$$\Delta \models_W \psi$$

iff for all α minimum in $\text{Mod}_W(\Delta)$, $\alpha \models \psi$.

Note that $\Delta \models_W \psi \implies \Delta \models \psi$.

2.2.3 Characterising minimum models

We now consider the existence of minimum models, first beginning with a characterisation theorem.

Theorem 2.1 (Characterising minimum models). Let $W \subseteq \text{Sen}(\mathcal{L})$ be a theory, $\mathcal{O} \subseteq \text{Sen}(\mathcal{L})$ a set of observations and $\Delta \subseteq \mathcal{O}$ a state description. Define $\mathcal{T}(\Delta) \subseteq \text{Sen}(\mathcal{L})$, by:

$$\begin{aligned} \mathcal{T}(\Delta) &= \Delta \cup \{\neg\varphi \mid \varphi \in \mathcal{O} \text{ and } \neg(\forall \beta \in \text{Mod}_W(\Delta) \quad \beta \models \varphi)\} \\ &= \Delta \cup \{\neg\varphi \mid \varphi \in \mathcal{O} \text{ and } \Delta \not\models_W \varphi\} \end{aligned}$$

Then, $\alpha \in \text{Mod}_W(\Delta)$ is minimum iff $\alpha \models \mathcal{T}(\Delta)$.

Proof. If $\alpha \in \text{Mod}_W(\Delta)$ is minimum, then for all $\varphi \in \mathcal{O}$,

$$\alpha \models \varphi \implies \Delta \models_W \varphi$$

(by definition of minimum). Hence, by contrapositive, for all $\varphi \in \mathcal{O}$,

$$\Delta \not\models_W \varphi \implies \alpha \models \neg\varphi$$

Thus $\alpha \models \mathcal{T}(\Delta)$.

Conversely, suppose $\alpha \models \mathcal{T}(\Delta)$. Then for all $\varphi \in \mathcal{O}$, if $\alpha \models \varphi$ then $\Delta \models_W \varphi$ (since, if not i.e. $\Delta \not\models_W \varphi$ then $\alpha \models \neg\varphi$ - contradiction). Hence, for all $\varphi \in \mathcal{O}$, if $\alpha \models \varphi$ then $\forall \beta \in \mathbf{Mod}_W(\Delta) \beta \models \varphi$. Thus α is minimum. \square

A direct consequence of the last theorem is the following corollary.

Corollary 2.1 (Existence of minimum models). *Consider a theory $W \subseteq \text{Sen}(\mathcal{L})$ and $\Delta \subseteq \mathcal{O}$ a state description where $\mathcal{O} \subseteq \text{Sen}(\mathcal{L})$ is a set of observations. A minimum model in $\mathbf{Mod}_W(\Delta)$ exists iff $\mathcal{T}(\Delta)$ is W -consistent, i.e. there is a model $\gamma \in \mathbf{Mod}_W(\Delta)$ with $\gamma \models \mathcal{T}(\Delta)$.*

We now characterise the satisfaction relation \approx in terms of \models .

Corollary 2.2 (Characterising satisfaction for minimum models). *Consider a theory $W \subseteq \text{Sen}(\mathcal{L})$ and state description $\Delta \subseteq \mathcal{O}$ where $\mathcal{O} \subseteq \text{Sen}(\mathcal{L})$ is a set of observations. Define $\mathcal{T}(\Delta)$ as above (Theorem 2.1). Then, for any \mathcal{L} -sentence ψ , we have*

$$\Delta \approx_W \psi \text{ iff } \mathcal{T}(\Delta) \models_W \psi$$

Proof. Assume $\Delta \approx_W \psi$, i.e. for all $\alpha \in \mathbf{Mod}_W(\Delta)$ minimum, $\alpha \models \psi$. But, by Theorem 2.1, α is minimum iff $\alpha \models \mathcal{T}(\Delta)$. Thus, for all $\alpha \in \mathbf{Mod}_W(\Delta)$ with $\alpha \models \mathcal{T}(\Delta)$, $\alpha \models \psi$, i.e. $\mathcal{T}(\Delta) \models_W \psi$.

Conversely, assume $\mathcal{T}(\Delta) \models_W \psi$, i.e. for all $\alpha \in \mathbf{Mod}_W(\Delta)$ if $\alpha \models \mathcal{T}(\Delta)$ then $\alpha \models \psi$. But α is minimum iff $\alpha \models \mathcal{T}(\Delta)$. Hence $\Delta \approx_W \psi$ as required. \square

Observe that Corollary 2.2 justifies the “absence as negation” interpretation, as $\mathcal{T}(\Delta)$ is adding the negations of the observations not entailed from the theory and the state description.

2.3 Minimum model examples

This section will present two examples, one based on a version of a blocks world and another one representing a boolean circuit. Both examples show the usefulness of minimum models to capture the “absence as negation” interpretation in

the presence of a theory describing the rules of the world, when modelling systems whose states can be described by positive observations.

2.3.1 A blocks world example

The following example is a combination of examples taken from [2] and [1]. A Blocks World consists of a finite collection of blocks (in which we think all the blocks are of the same size) which may be placed on top each other to form towers or may reside on a table. The blocks can have a color, and we will assume that it is either red or green (see Figure 2.1). We will have two distinct entities, blocks and tables, and we will call “object” to any entity that is either a block or a table. There will be a predicate $on(x, y)$ stating that block x is directly on object y , a predicate $red(a)$ stating that block a is red, and predicate $green(a)$ stating that block a is green.

We will also have *abstraction* predicates, or properties that can be deduced from the observations or facts in the state of the world under the rules imposed to the world (the rules of the world will be represented by a first-order theory). We will use the abstraction predicate $above(x, y)$ to represent the fact that the block x can be found somewhere in the blocks that are on top the object y .

We present now the theory W , where *Block* and *Table* are enumeration types and *Object* is the type of their union. We will use four blocks and one table.

TYPES

$$Block \stackrel{\text{def}}{=} \{A, B, C, D\} \quad Table \stackrel{\text{def}}{=} \{T\} \quad Object \stackrel{\text{def}}{=} Block \cup Table$$

OBSERVATION PREDICATES

$$on : Block \times Object \quad red : Block \quad green : Block$$

ABSTRACTION PREDICATES

$$above : Block \times Object$$

AXIOMS

$$\forall x : Block \neg on(x, x)$$

$$\forall x : Block \forall y : Object \forall z : Object \ on(x, y) \wedge on(x, z) \rightarrow y = z$$

$$\forall x : Block \forall y : Block \forall z : Block \ on(y, x) \wedge on(z, x) \rightarrow y = z$$

$$\forall x : Block \forall y : Object \ on(x, y) \rightarrow above(x, y)$$

$$\forall x : Block \forall y : Block \forall z : Object \ above(x, y) \wedge above(y, z) \rightarrow above(x, z)$$

$$\forall x : Block \forall y : Block \ above(x, y) \rightarrow \neg above(y, x)$$

$$\forall x: Block \neg green(x) \leftrightarrow red(x)$$

The first axiom is stating that blocks cannot be put on top themselves. The second is saying that the same block cannot be put on two distinct objects at the same time. The third is saying that two distinct blocks cannot be put on the same block at the same time. The fourth introduces the abstraction *above*. The fifth says that the abstraction *above* is transitive. The sixth is saying that the abstraction *above* cannot be inverted between pairs of blocks. The last one is stating that the color of a block is mutually exclusive: either green or red but not both.

The set of observations \mathcal{O} will be the set of atomic sentences formed from observation predicates, in other words, a state description will have sentences stating how blocks are stacked on objects (i.e. blocks or tables) and the color of the blocks. For example, Figure 2.1 shows four blocks A , B , C and D and table T . The letter in the upper left corner of each block indicates the color: R for red and G for green. The state description for Figure 2.1 is

$$\Delta = \{ on(A, B), on(C, D), on(B, T), on(D, T), \\ red(A), green(B), green(C), red(D) \}$$

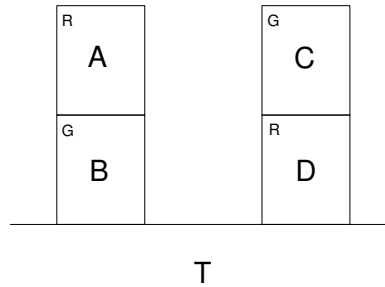


Figure 2.1: A Blocks World state.

As described in Section 2.1, for Δ to be a full account of the state of the world, it needs to include the negations of the observations that do not hold. Since here we are stating only the positive facts, we need to take the position that any observation that is not deducible from our state description Δ and the rules of the world W , is false, i.e. we need to assume that the world is “closed”, which is what we have in mind when we “draw” a state of the world as in Figure 2.1. For example, when we look at the picture, B is not on C and D is not on A , also

A is not on the table T . And indeed, we have $\Delta \not\#_W on(B, C)$; $\Delta \not\#_W on(D, A)$; and $\Delta \not\#_W on(A, T)$, as the “model” in Figure 2.1 satisfies $W \cup \Delta$ but falsifies the three sentences. Therefore, according to Theorem 2.1, every minimum model of $\Delta \cup W$ should satisfy $\neg on(B, C)$, $\neg on(D, A)$, and $\neg on(A, T)$ (equivalently: $\Delta \approx_W \neg on(B, C)$, $\Delta \approx_W \neg on(D, A)$ and $\Delta \approx_W \neg on(A, T)$), in other words, every minimum model satisfies the negations of the observations that are not being represented in Figure 2.1.

Figure 2.1 also shows that $green(A)$ does not hold, and indeed we have $\Delta \not\#_W green(A)$. Therefore, by Theorem 2.1 and Corollary 2.2, $\Delta \approx_W \neg green(A)$. But, observe that $red(A) \in \Delta$, therefore, by using the last axiom in W , we get classically $\Delta \vDash_W \neg green(A)$ which implies $\Delta \approx_W \neg green(A)$ as everything that holds under classical entailment also holds under minimum model entailment (see Definition 2.3).

What happens if we remove $green(B)$ from the state description Δ to get the state Δ_2 ? Then, it follows that both $\Delta_2 \not\#_W green(B)$ and $\Delta_2 \not\#_W red(B)$ and therefore, $\neg green(B) \in \mathcal{T}(\Delta_2)$ and $\neg red(B) \in \mathcal{T}(\Delta_2)$ by Theorem 2.1 and so, $\mathcal{T}(\Delta_2)$ will be inconsistent with respect to the last axiom in W . From here, by Corollary 2.1 there is no minimum model for Δ_2 . So, the existence of a minimum model depends on the theory W (i.e. if the last axiom is not in W then $\mathcal{T}(\Delta_2)$ will not be inconsistent), the set of possible observations \mathcal{O} (i.e. if the predicate $green$ is not an observation and therefore, we need to remove all its appearances in Δ_2 , then $\mathcal{T}(\Delta_2)$ will not be inconsistent) and the state description Δ (i.e. if $green(B) \in \Delta_2$ then $\mathcal{T}(\Delta_2)$ will not be inconsistent).

2.3.2 A boolean circuit example

The following example was inspired by an example in [18]. The differences are: the example in [18] uses an untyped theory, the axioms of the theory that will be presented in here are almost different to the ones in [18], the concept ontology differs from the one presented in here, and the example in [18] is using standard models.

Imagine you have a circuit board with one hundred inputs and one hundred outputs. Imagine you have a stock of one hundred NOR gates (negated OR) that you can attach to the circuit board and wire them in the board, such that they can form a boolean circuit taking inputs from the inputs in the circuit board and giving outputs to the outputs in the circuit board (see Figure 2.2). Each NOR

gate has two inputs, which we will call “input connection points”. Also, each NOR gate has an output, which we will call “output connection point”. Every input connection point together with the outputs of the circuit board will be called “signal receivers”. Every output connection point together with the inputs of the circuit board will be called “signal senders”. A “carrier” will be either a signal receiver or a signal sender. Every carrier has a “signal” which we will interpret as absence of electrical current (0) or presence of electrical current (1).

We will use the predicate $connected(a, b)$ to represent the fact that there is a wire from the signal sender a to the signal receiver b . The function $signal(a)$ will represent the signal value (0 or 1) of the carrier a . The function $first_in(a)$ will represent the first input connection point of the gate a , $second_in(a)$ will represent the second input connection point of the gate a , and $gate_out(a)$ will represent the output connection point of the gate a .

Also, we will assume that every input of the circuit board has attached a switch, such that when the switch is on, the signal on the input will be 1, and conversely when the switch is off, the signal in the input will be 0. We will use the predicate $switch_on(a)$ to represent the fact that the input to the circuit board a has its switch on.

Now, we present the theory W , where NOR , $Input$, $Output$ and $Signal_Value$ are enumeration types, and Sig_Sender , $Sig_Receiver$ and $Carrier$ are union types.

TYPES

$$\begin{aligned}
NOR &\stackrel{\text{def}}{=} \{NG_1, \dots, NG_{100}\} & Input &\stackrel{\text{def}}{=} \{I_1, \dots, I_{100}\} \\
Output &\stackrel{\text{def}}{=} \{O_1, \dots, O_{100}\} & In_Conn_Point & \quad Out_Conn_Point \\
Sig_Sender &\stackrel{\text{def}}{=} Input \cup Out_Conn_Point \\
Sig_Receiver &\stackrel{\text{def}}{=} Output \cup In_Conn_Point \\
Carrier &\stackrel{\text{def}}{=} Sig_Sender \cup Sig_Receiver \\
Signal_Value &\stackrel{\text{def}}{=} \{0, 1\}
\end{aligned}$$

OBSERVATION PREDICATES

$$connected : Sig_Sender \times Sig_Receiver \quad switch_on : Input$$

FUNCTIONS

$$\begin{aligned}
signal &: Carrier \rightarrow Signal_Value \\
first_in &: NOR \rightarrow In_Conn_Point \\
second_in &: NOR \rightarrow In_Conn_Point
\end{aligned}$$

$$gate_out : NOR \rightarrow Out_Conn_Point$$

AXIOMS

$$\forall x:Sig_Sender \forall y:Sig_Receiver \text{ connected}(x, y) \rightarrow signal(x) = signal(y)$$

$$\forall x:NOR \text{ signal}(gate_out(x)) = 1 \leftrightarrow$$

$$[signal(first_in(x)) = 0 \wedge signal(second_in(x)) = 0]$$

$$\forall x:In_Conn_Point \exists g:NOR [first_in(g) = x \vee second_in(g) = x]$$

$$\forall x:Out_Conn_Point \exists g:NOR [gate_out(g) = x]$$

$$\forall x:Input \text{ switch_on}(x) \leftrightarrow signal(x) = 1$$

The first axiom is stating that if a signal sender and a signal receiver are connected by a wire, then they have the same signal. The second is stating that the signal of the output connection point of a NOR gate is 1 if and only if both of its input connection points have signal 0. The third is stating that every input connection point belongs to a gate and similarly the fourth is saying that every output connection point belongs to a gate. The last one is stating the condition that an input has signal 1 if and only if its switch is on.

The set of observations \mathcal{O} will be the set of atomic sentences formed from observation predicates, in other words, a state description will have sentences stating how gates are wired in the circuit board and which inputs have their switch turned on. For example, the state description of Figure 2.2 will be (where an input to the circuit board with a black fill means that its switch is on):

$$\begin{aligned} \Delta = \{ & \text{connected}(I_1, first_in(NG_1)), \text{connected}(I_2, first_in(NG_2)), \\ & \text{connected}(I_3, second_in(NG_1)), \text{connected}(I_4, second_in(NG_2)), \\ & \text{connected}(gate_out(NG_1), first_in(NG_3)), \\ & \text{connected}(gate_out(NG_2), second_in(NG_3)), \\ & \text{connected}(gate_out(NG_3), O_1), \\ & \text{switch_on}(I_1) \} \end{aligned}$$

Again, as described in Section 2.1, for Δ to be a full account of the state of the system, it needs to include the negations of the observations that do not hold. Intuitively, when we describe the state of some system, we state the positive observations and we assume that the world is “closed”, in the sense that everything that is not stated or implied by the rules of the world, is false. In our case, when we state Δ above, we have in mind the model represented in Figure 2.2, as the model represented by the figure satisfies all the “rules” imposed by

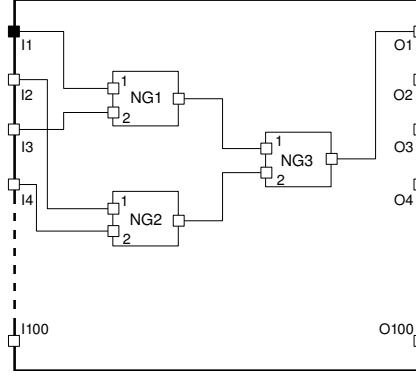


Figure 2.2: A boolean circuit state.

the theory W and observations that are not represented in the figure, do not hold. For example, when we see the picture, I_3 does not have its switch turned on, also, there is no wiring between, say I_1 and an input connection point with a non-existent gate in the picture, like $first_in(NG_7)$. And indeed, we have $\Delta \not\models_W switch_on(I_3)$ and $\Delta \not\models_W connected(I_1, first_in(NG_7))$ as the “model” in Figure 2.2 satisfies $W \cup \Delta$ but falsifies the two sentences. Therefore, according to Theorem 2.1, every minimum model of $\Delta \cup W$ should satisfy $\neg switch_on(I_3)$ and $\neg connected(I_1, first_in(NG_7))$ (equivalently, $\Delta \models_W \neg switch_on(I_3)$, and $\Delta \models_W \neg connected(I_1, first_in(NG_7))$), in other words, every minimum model satisfies the negations of the observations that are not being represented in Figure 2.2.

This property of minimum models produces the effect that when we restrict entailment of statements from a state description Δ with respect to minimum models only, we can assert more statements than classically possible. For example, in Figure 2.2 we can show that, given Δ above, every minimum model satisfies $signal(O_1) = 0$ which is what we would expect under the intuitive understanding of the gates: Since inputs I_2 , I_3 and I_4 do not have their switches turned on, their signal is 0. Since input I_1 has its switch turned on, its signal is 1. Then, following the semantics of the NOR gate, NG_3 will have as inputs 0 and 1, and therefore O_1 will be 0. A more detailed proof of the statement $signal(O_1) = 0$ under minimum model entailment involves the following observation:

- The sentence $\forall x : Carrier\ signal(x) = 0 \vee signal(x) = 1$ is classically provable from $W \cup \Delta$: By universal instantiation on the axiom of the enumeration type *Signal_Value* we get $signal(x) = 0 \vee signal(x) = 1$ where x

is a variable of type *Carrier*. The result follows by universal generalization, as x does not appear free in $W \cup \Delta$.

Let us do the case for NG_1 :

- $\Delta \vDash_W \neg \text{switch_on}(I_3)$ follows by Theorem 2.1 and Corollary 2.2 as $\Delta \not\vDash_W \text{switch_on}(I_3)$ (just take the “model” represented in Figure 2.2 as it satisfies $W \cup \Delta$ but falsifies $\text{switch_on}(I_3)$). Therefore, $\Delta \vDash_W \text{signal}(I_3) \neq 1$ by the last axiom in W and so, $\Delta \vDash_W \text{signal}(I_3) = 0$ by the observation above. Then, by the first axiom of W , $\Delta \vDash_W \text{signal}(\text{second_in}(NG_1)) = 0$ as $\text{connected}(I_3, \text{second_in}(NG_1))$.
- Since $\text{connected}(I_1, \text{first_in}(NG_1))$, by the first axiom $\Delta \vDash_W \text{signal}(I_1) = \text{signal}(\text{first_in}(NG_1))$. Since $\text{switch_on}(I_1)$ then $\text{signal}(I_1) = 1$ and therefore $\Delta \vDash_W \text{signal}(\text{first_in}(NG_1)) = 1$. So, $\Delta \vDash_W \text{signal}(\text{first_in}(NG_1)) \neq 0$ by $1 \neq 0$ in the enumeration axioms in type *Signal_Value*. And therefore, by the second axiom $\Delta \vDash_W \text{signal}(\text{gate_out}(NG_1)) \neq 1$ and so, $\Delta \vDash_W \text{signal}(\text{gate_out}(NG_1)) = 0$ by the observation above (i.e. $\forall x : \text{Carrier } \text{signal}(x) = 0 \vee \text{signal}(x) = 1$).

The cases for NG_2 and NG_3 are similar. $\Delta \vDash_W \text{signal}(\text{gate_out}(NG_2)) = 1$ will hold, and finally we will get $\Delta \vDash_W \text{signal}(\text{gate_out}(NG_3)) = 0$ and therefore $\Delta \vDash_W \text{signal}(O_1) = 0$ by the first axiom as $\text{connected}(\text{gate_out}(NG_3), O_1)$.

Although, observe that classically $\Delta \not\vDash_W \text{signal}(O_1) = 0$ as the following standard model \mathfrak{A} shows (by switching on the switch of input I_2 in Figure 2.2):

DOMAINS

$$\begin{aligned}
\text{NOR}^{\mathfrak{A}} &= \{g_1, \dots, g_{100}\} & \text{Input}^{\mathfrak{A}} &= \{i_1, \dots, i_{100}\} \\
\text{Output}^{\mathfrak{A}} &= \{o_1, \dots, o_{100}\} & \text{In_Conn_Point}^{\mathfrak{A}} &= \{ip_1, \dots, ip_{200}\} \\
\text{Out_Conn_Point}^{\mathfrak{A}} &= \{op_1, \dots, op_{100}\} \\
\text{Sig_Sender} &= \text{Input} \cup \text{Out_Conn_Point} \\
\text{Sig_Receiver} &= \text{Output} \cup \text{In_Conn_Point} \\
\text{Carrier} &= \text{Sig_Sender} \cup \text{Sig_Receiver} & \text{Signal_Value} &= \{\perp, \top\}
\end{aligned}$$

RELATIONS

$$\begin{aligned}
\text{connected}^{\mathfrak{A}} &= \{(i_1, ip_1), (i_2, ip_3), (i_3, ip_2), (i_4, ip_4), \\
&\quad (op_1, ip_5), (op_2, ip_6), (op_3, o_1)\} \\
\text{switch_on}^{\mathfrak{A}} &= \{i_1, i_2\}
\end{aligned}$$

FUNCTIONS

$$\begin{aligned}
& \text{signal}^{\mathfrak{A}}(i_1) = \top, \text{signal}^{\mathfrak{A}}(i_2) = \top, \text{signal}^{\mathfrak{A}}(ip_1) = \top, \\
& \text{signal}^{\mathfrak{A}}(ip_2) = \perp, \text{signal}^{\mathfrak{A}}(ip_3) = \top, \text{signal}^{\mathfrak{A}}(ip_4) = \perp, \\
& \text{signal}^{\mathfrak{A}}(op_1) = \perp, \text{signal}^{\mathfrak{A}}(op_2) = \perp, \text{signal}^{\mathfrak{A}}(ip_5) = \perp, \\
& \text{signal}^{\mathfrak{A}}(ip_6) = \perp, \text{signal}^{\mathfrak{A}}(op_3) = \top, \text{signal}^{\mathfrak{A}}(o_1) = \top, \\
& \text{signal}^{\mathfrak{A}}(ip_j) = \perp \text{ for } j \geq 7, \text{signal}^{\mathfrak{A}}(op_j) = \top \text{ for } j \geq 4, \\
& \text{signal}^{\mathfrak{A}}(i_j) = \perp \text{ for } j \geq 3, \text{signal}^{\mathfrak{A}}(o_j) = \perp \text{ for } j \geq 2, \\
& \text{first_in}^{\mathfrak{A}}(g_j) = ip_{2j-1}, \text{second_in}^{\mathfrak{A}}(g_j) = ip_{2j}, \\
& \text{gate_out}^{\mathfrak{A}}(g_j) = op_j
\end{aligned}$$

CONSTANTS

$$NG_j^{\mathfrak{A}} = g_j, I_j^{\mathfrak{A}} = i_j, O_j^{\mathfrak{A}} = o_j, 0^{\mathfrak{A}} = \perp, 1^{\mathfrak{A}} = \top$$

Then, \mathfrak{A} satisfies $W \cup \Delta$ but it does not satisfy $\text{signal}(O_1) = 0$ as $\text{signal}^{\mathfrak{A}}(o_1) = \top = 1^{\mathfrak{A}}$.

What if we change Δ by adding the sentence $\text{switch_on}(I_2)$ to get Δ_2 ? A similar analysis will show that $\Delta_2 \approx_W \text{signal}(O_1) = 1$. If we remove $\text{switch_on}(I_1)$ from Δ_2 to get Δ_3 we can check that $\Delta_3 \approx_W \text{signal}(O_1) = 0$. And finally, if we remove all switch-related predicates to get Δ_4 , we can check that $\Delta_4 \approx_W \text{signal}(O_1) = 0$. Therefore, the circuit of Figure 2.2 is implementing an AND gate under minimum model entailment, where the AND gate is using I_1 and I_2 as inputs. This is the expected behaviour when we look at picture 2.2, under the rules imposed by the theory W . Of course, we could also change the connections in Δ or add more gates and see what are the output signals of the circuit under minimum model entailment.

These examples showed that the minimum model approach is useful to capture the intuitive idea of the ‘‘absence as negation’’ interpretation under the restrictions imposed by a theory, when we look at systems whose states can be described by positive observations, instead of an approach that uses standard models.

Chapter 3

Minimum model reasoning

This chapter describes proof systems for minimum model reasoning. Section 3.1 introduces some definitions that will be used along this chapter. Section 3.2 introduces a proof system which is intended to work under ideal state descriptions, i.e. states where all positive observations that hold in the system are stated. Soundness, completeness and effectiveness results for the restricted proof system are shown. Section 3.3 discusses the difficulties involved in constructing a proof system for minimum model reasoning that is sound, complete and effective, when we drop any restrictions on the state descriptions and allow general theories. The proof system of Section 3.4 relaxes the “ideal” restriction on the state descriptions by allowing interactions between observations in a part of the theory that is decidable with respect to observations. The decidable part of the theory is selected entirely based on the form of the sentences. Soundness, completeness and effectiveness results for the proof system are shown. Finally, Section 3.5 describes a class of proof systems that drop any restriction on the state descriptions under the expense of sacrificing full completeness. For this class of proof systems two approaches are discussed: a syntactical approach which verifies the form of the sentences in the theory and a semantic approach that searches for finite models of the theory but falsify the observation given as query. Soundness and effectiveness results are shown for this class of proof systems.

3.1 \mathcal{O} -closedness and \mathcal{O} -independence

The following definitions will be used along this chapter. $Form(\mathcal{L})$ will denote the set of well-formed formulae of the typed, first-order signature \mathcal{L} . Similarly,

$Sen(\mathcal{L})$ will denote the set of well-formed sentences of the typed, first-order signature \mathcal{L} .

Definition 3.1. *Let $\mathcal{O} \subseteq Sen(\mathcal{L})$ be a set of possible observations and $W \subseteq Sen(\mathcal{L})$ a theory of \mathcal{L} . Let $\Delta \subseteq \mathcal{O}$ be a finite state description. Δ is \mathcal{O} -closed with respect to W if $\{\psi \mid \psi \in \mathcal{O} \text{ and } \Delta \vDash_W \psi\} \subseteq \Delta$.*

Intuitively, \mathcal{O} -closedness means that the set already contains all the observations in \mathcal{O} entailed from the set by the theory W .

Definition 3.2. *Let $\mathcal{O} \subseteq Sen(\mathcal{L})$ be a set of possible observations and $W \subseteq Sen(\mathcal{L})$ a theory of \mathcal{L} . W is \mathcal{O} -independent if for every finite subset $\Delta \subseteq \mathcal{O}$ which is W -consistent, Δ is \mathcal{O} -closed with respect to W .*

Intuitively, \mathcal{O} -independence puts a restriction on the theory W in such a way that the theory is not able to entail a positive observation which is not already in the state description. This remark is clearer if the contrapositive form of \mathcal{O} -closedness is used in the definition of \mathcal{O} -independence:

Remark 3.1. *Let $\mathcal{O} \subseteq Sen(\mathcal{L})$ and $W \subseteq Sen(\mathcal{L})$ a theory of \mathcal{L} . W is \mathcal{O} -independent if for every finite subset $\Delta \subseteq \mathcal{O}$ which is W -consistent, the following property holds:*

Δ is \mathcal{O} -closed with respect to W , which is equivalent to (by definition of the \subseteq relation):

$$\forall \psi \in \mathcal{O} \quad \Delta \vDash_W \psi \implies \psi \in \Delta$$

which is equivalent to (by contrapositive):

$$\forall \psi \in \mathcal{O} \quad \psi \notin \Delta \implies \Delta \not\vDash_W \psi$$

In this sense, \mathcal{O} -independence is a stronger form of \mathcal{O} -closedness.

3.2 A restricted proof system

In this section we will present a proof system for minimum model reasoning assuming certain restrictions on the state descriptions (subsets of the set of possible observations) or in the theory, namely, state descriptions are \mathcal{O} -closed with respect to the theory or the theory is \mathcal{O} -independent. Later on we will relax those restrictions and discuss the consequences of dropping the restrictions.

The intuition behind these restrictions rely on our intuitive understanding of what an *ideal* state description is: it should include all directly measurable observations (i.e. the state description is \mathcal{O} -closed) or the theory should not be able to prove an observation that has not been already stated in the state description (i.e. the theory is \mathcal{O} -independent). What happens if the state description is not ideal? For example, a situation where it is impossible to measure all relevant observations or a situation where there are dependencies between observations and so it is not necessary to specify all of them. These form of state descriptions will be explored in later sections.

Let Υ be a sound and complete set of inference rules in a natural deduction style for classical first-order logic with equality. A general inference rule in Υ looks like the following, where $\Gamma, \Gamma_i \subseteq Form(\mathcal{L})$ for $1 \leq i \leq n$ and $\psi, \psi_i \in Form(\mathcal{L})$ for $1 \leq i \leq n$:

$$\frac{\Gamma_1 \mid_W \psi_1 \quad \dots \quad \Gamma_n \mid_W \psi_n}{\Gamma \mid_W \psi}$$

The notation $\Gamma \mid_W \psi$ is a shorthand for $W \cup \Gamma \mid \psi$ where $W \subseteq Sen(\mathcal{L})$ is a fixed theory. Accordingly, the Reflexivity Axiom Schema has the form:

$$\frac{\psi \in \Gamma \text{ or } \psi \in W}{\Gamma \mid_W \psi}$$

As in the standard case for first-order logic, a *proof* will be a finite sequence of sequents $\Gamma_1 \mid_W \psi_1 \dots \Gamma_n \mid_W \psi_n$ where each $\Gamma_i \subseteq Form(\mathcal{L})$ is finite and each $\psi_i \in Form(\mathcal{L})$. Also, each sequent $\Gamma_i \mid_W \psi_i$ in a proof is either an instance of an axiom or the conclusion of an instance of an inference rule of the following form, where $j_t < i$ for $1 \leq t \leq n$ and each $\Gamma_{j_t} \mid_W \psi_{j_t}$ is a sequent occurring previously in the proof:

$$\frac{\Gamma_{j_1} \mid_W \psi_{j_1} \quad \dots \quad \Gamma_{j_n} \mid_W \psi_{j_n}}{\Gamma_i \mid_W \psi_i}$$

Similarly, the notation $\Gamma \vdash_W \psi$ means that there exists a proof $\Gamma_1 \mid_W \psi_1 \dots \Gamma_n \mid_W \psi_n$ such that $\Gamma_n \subseteq \Gamma$ and $\psi_n = \psi$.

Let Υ^M be the deduction system obtained from Υ in the following way:

Each inference rule of Υ is changed to:

$$\frac{\Theta, \Lambda, \Gamma_1 \mid_W^M \psi_1 \quad \dots \quad \Theta, \Lambda, \Gamma_n \mid_W^M \psi_n}{\Theta, \Lambda, \Gamma \mid_W^M \psi}$$

Where $\Lambda \subseteq Sen(\mathcal{L})$, $\Theta \subseteq Sen(\mathcal{L})$ and $\Lambda \subseteq \Theta$ are place-holders that can be instantiated by any sets that satisfy the conditions. We shall interpret Θ as a place-holder for the possible observations, Λ as a place-holder for state descriptions, W as a place-holder for theories and Γ as the *working set* of the sequent. The Reflexivity Axiom Schema is changed to:

$$\frac{\psi \in \Gamma \text{ or } \psi \in W \text{ or } \psi \in \Lambda}{\Theta, \Lambda, \Gamma \mid_W^M \psi}$$

And the Minimum Axiom Schema is added:

$$\frac{\psi \in \Theta \quad \psi \notin \Lambda}{\Theta, \Lambda, \Gamma \mid_W^M \neg\psi} \quad (*)$$

Intuitively, Λ , Θ and W act as parameters that are carried over along the inference rules, and the only places where they are used are in the Minimum Axiom Schema and the Reflexivity Axiom Schema.

A *proof* will be a finite sequence of sequents $\Theta, \Lambda, \Gamma_1 \mid_W^M \psi_1 \dots \Theta, \Lambda, \Gamma_n \mid_W^M \psi_n$ where each $\Gamma_i \subseteq Form(\mathcal{L})$ is finite and each $\psi_i \in Form(\mathcal{L})$. Also, each sequent $\Theta, \Lambda, \Gamma_i \mid_W^M \psi_i$ in a proof is either an instance of an axiom or the conclusion of an instance of an inference rule of the following form, where $j_t < i$ for $1 \leq t \leq n$ and each $\Theta, \Lambda, \Gamma_{j_t} \mid_W^M \psi_{j_t}$ is a sequent occurring previously in the proof:

$$\frac{\Theta, \Lambda, \Gamma_{j_1} \mid_W^M \psi_{j_1} \quad \dots \quad \Theta, \Lambda, \Gamma_{j_n} \mid_W^M \psi_{j_n}}{\Theta, \Lambda, \Gamma_i \mid_W^M \psi_i}$$

Similarly, the notation $\Theta, \Lambda, \Gamma \vdash_W^M \psi$ means that there exists a proof $\Theta, \Lambda, \Gamma_1 \mid_W^M \psi_1 \dots \Theta, \Lambda, \Gamma_n \mid_W^M \psi_n$ such that $\Gamma_n \subseteq \Gamma$ and $\psi_n = \psi$.

Given a theory $W \subseteq Sen(\mathcal{L})$, possible observations $\mathcal{O} \subseteq Sen(\mathcal{L})$ and state description $\Delta \subseteq \mathcal{O}$, we will denote by $\mathcal{T}(\Delta)$ the set $\Delta \cup \{\neg\psi \mid \psi \in \mathcal{O} \text{ and } \Delta \not\approx_W \psi\}$ as was defined in Theorem 2.1.

Theorem 3.1 (Soundness for \mathcal{O} -closedness in Υ^M). *Let $\mathcal{O} \subseteq Sen(\mathcal{L})$ be a set of possible observations and $W \subseteq Sen(\mathcal{L})$ a theory of \mathcal{L} . Then, for every $\psi \in Sen(\mathcal{L})$ and every finite, \mathcal{O} -closed $\Delta \subseteq \mathcal{O}$ with respect to W :*

$$\mathcal{O}, \Delta, \emptyset \vdash_W^M \psi \implies \Delta \approx_W \psi$$

Proof. Suppose $\mathcal{O}, \Delta, \emptyset \vdash_W^M \psi$. By Corollary 2.2 and the completeness theorem for first-order logic, it suffices to show that $\mathcal{T}(\Delta) \vdash_W \psi$. By assumption, there is a proof $\mathcal{O}, \Delta, \Gamma_1 \vdash_W^M \psi_1 \dots \mathcal{O}, \Delta, \Gamma_n \vdash_W^M \psi_n$ such that $\Gamma_n = \emptyset$ and $\psi_n = \psi$.

Claim: $\mathcal{O}, \Delta, \Gamma_i \vdash_W^M \psi_i \implies \mathcal{T}(\Delta) \cup \Gamma_i \vdash_W \psi_i$ for $1 \leq i \leq n$. By induction on the length of the proof, we consider each axiom and inference rule in Υ^M . It suffices to show the case for the Minimum Axiom Schema.

So, for $\mathcal{O}, \Delta, \Gamma_i \vdash_W^M \psi_i$ to be an instance of the Minimum Axiom Schema, $\psi_i = \neg\phi$ for some $\phi \in \mathcal{O}$ such that $\phi \notin \Delta$. Since Δ is \mathcal{O} -closed, by definition $\phi \notin \{\psi \mid \psi \in \mathcal{O} \text{ and } \Delta \vDash_W \psi\}$ which means that $\Delta \not\vDash_W \phi$ (as $\phi \in \mathcal{O}$). This in turn implies that $\neg\phi \in \mathcal{T}(\Delta) = \Delta \cup \{\neg\psi \mid \psi \in \mathcal{O} \text{ and } \Delta \not\vDash_W \psi\}$ (i.e. $\psi_i \in \mathcal{T}(\Delta)$), and therefore $\mathcal{T}(\Delta) \cup \Gamma_i \vdash_W \psi_i$.

This proves the claim.

By the claim, it follows that $\mathcal{T}(\Delta) \cup \Gamma_n \vdash_W \psi_n$ which is equivalent to $\mathcal{T}(\Delta) \vdash_W \psi$ (as $\Gamma_n = \emptyset$ and $\psi_n = \psi$). \square

What is the minimum property a finite subset $\Delta \subseteq \mathcal{O}$ needs to have in order to guarantee soundness in the proof system Υ^M ? The following theorem shows that.

Theorem 3.2 (Characterising soundness in Υ^M). *Let $\mathcal{O} \subseteq \text{Sen}(\mathcal{L})$, $W \subseteq \text{Sen}(\mathcal{L})$ a theory of \mathcal{L} and $\Delta \subseteq \mathcal{O}$ be finite. Then, $\forall \psi \in \text{Sen}(\mathcal{L}) \mathcal{O}, \Delta, \emptyset \vdash_W^M \psi \implies \Delta \vDash_W \psi$ IF AND ONLY IF there is no minimum model in $\text{Mod}_W(\Delta)$ or Δ is \mathcal{O} -closed with respect to W .*

Proof. \implies . Suppose $\forall \psi \in \text{Sen}(\mathcal{L}) \mathcal{O}, \Delta, \emptyset \vdash_W^M \psi \implies \Delta \vDash_W \psi$. Suppose there is a minimum model in $\text{Mod}_W(\Delta)$. We will show that Δ is \mathcal{O} -closed with respect to W .

Let $\phi \in \{\psi \mid \psi \in \mathcal{O} \text{ and } \Delta \vDash_W \psi\}$. Then, $\phi \in \mathcal{O}$ and $\Delta \vDash_W \phi$. Suppose that $\phi \notin \Delta$. Then, $\mathcal{O}, \Delta, \emptyset \vdash_W^M \neg\phi$ is an instance of the Minimum Axiom Schema, and therefore $\mathcal{O}, \Delta, \emptyset \vdash_W^M \neg\phi$. Therefore, by assumption of soundness, $\Delta \vDash_W \neg\phi$ and $\mathcal{T}(\Delta) \vdash_W \neg\phi$ by Corollary 2.2. But $\Delta \vDash_W \phi$ which implies that $\mathcal{T}(\Delta) \vDash_W \phi$, so $\mathcal{T}(\Delta) \vdash_W \phi$ by completeness of first-order logic. So, $\mathcal{T}(\Delta) \vdash_W \perp$. But, since there is a minimum model in $\text{Mod}_W(\Delta)$, $\mathcal{T}(\Delta)$ is W -consistent (by Corollary 2.1) which contradicts $\mathcal{T}(\Delta) \vdash_W \perp$. Therefore $\phi \in \Delta$ as required.

\Leftarrow . Suppose there is no minimum model in $\text{Mod}_W(\Delta)$ or Δ is \mathcal{O} -closed. The case for \mathcal{O} -closed Δ follows directly by Theorem 3.1. For the other case, since there is no minimum model in $\text{Mod}_W(\Delta)$, then trivially, for any $\psi \in \text{Sen}(\mathcal{L})$,

$\Delta \vDash_W \psi$. Therefore $\forall \psi \in \text{Sen}(\mathcal{L}) \ \mathcal{O}, \Delta, \emptyset \vdash_W^M \psi \implies \Delta \vDash_W \psi$ since the conclusion does not depend on the hypothesis of the implication. \square

A direct consequence of the last theorem, is the following corollary.

Corollary 3.1 (Soundness for \mathcal{O} -independence in Υ^M). *Let $\mathcal{O} \subseteq \text{Sen}(\mathcal{L})$ and $W \subseteq \text{Sen}(\mathcal{L})$ an \mathcal{O} -independent theory of \mathcal{L} . Then, for every $\psi \in \text{Sen}(\mathcal{L})$ and every finite $\Delta \subseteq \mathcal{O}$:*

$$\mathcal{O}, \Delta, \emptyset \vdash_W^M \psi \implies \Delta \vDash_W \psi$$

Proof. Let $\Delta \subseteq \mathcal{O}$ be finite and W be \mathcal{O} -independent. We want to show that either, there is no minimum model in $\text{Mod}_W(\Delta)$ or Δ is \mathcal{O} -closed. Suppose there is a minimum model in $\text{Mod}_W(\Delta)$. This implies that Δ is W -consistent. As W is \mathcal{O} -independent, and Δ is finite and W -consistent, Δ is \mathcal{O} -closed by definition.

By Theorem 3.2, soundness holds. \square

The following theorem shows that completeness does not require any restriction on the state descriptions.

Theorem 3.3 (Completeness in Υ^M). *Let $\mathcal{O} \subseteq \text{Sen}(\mathcal{L})$ and $W \subseteq \text{Sen}(\mathcal{L})$ a theory of \mathcal{L} . Then, for every $\psi \in \text{Sen}(\mathcal{L})$ and every finite $\Delta \subseteq \mathcal{O}$:*

$$\mathcal{O}, \Delta, \emptyset \vdash_W^M \psi \iff \Delta \vDash_W \psi$$

Proof. Suppose $\Delta \vDash_W \psi$. By Corollary 2.2 and the completeness theorem for first-order logic, $\mathcal{T}(\Delta) \vdash_W \psi$ holds. Then, there is a proof $\Gamma_1 \mid_W \psi_1 \dots \Gamma_n \mid_W \psi_n$ such that $\Gamma_n \subseteq \mathcal{T}(\Delta)$, $\psi_n = \psi$ and each Γ_i is finite.

Claim: $\Gamma_i \mid_W \psi_i \implies \mathcal{O}, \Delta, \Gamma_i - \mathcal{T}(\Delta) \vdash_W^M \psi_i$ for $1 \leq i \leq n$. By induction on the length of the proof, we consider each axiom and inference rule in Υ . It suffices to show the case for the Reflexivity Axiom Schema.

So, for $\Gamma_i \mid_W \psi_i$ to be an instance of the Reflexivity Axiom Schema, either $\psi_i \in \Gamma_i$ or $\psi_i \in W$ must hold. If $\psi_i \in W$ the result follows trivially by the Reflexivity Axiom Schema in Υ^M . Now, for the case when $\psi_i \in \Gamma_i$, either $\psi_i \notin \mathcal{T}(\Delta)$ or $\psi_i \in \mathcal{T}(\Delta)$. If $\psi_i \notin \mathcal{T}(\Delta)$ then $\psi_i \in \Gamma_i - \mathcal{T}(\Delta)$ and the result follows trivially by the Reflexivity Axiom Schema in Υ^M . If $\psi_i \in \mathcal{T}(\Delta)$ then either $\psi_i \in \Delta$ or $\psi_i \in \{\neg\psi \mid \psi \in \mathcal{O} \text{ and } \Delta \not\vDash_W \psi\}$. If $\psi_i \in \Delta$ the result follows trivially by the Reflexivity Axiom Schema in Υ^M . For the other case, $\psi_i = \neg\phi$ for some $\phi \in \mathcal{O}$ such that $\Delta \not\vDash_W \phi$. If ϕ were in Δ , then $\Delta \vDash_W \phi$ which is a contradiction. It

follows that $\phi \notin \Delta$ and so $\mathcal{O}, \Delta, \Gamma_i - \mathcal{T}(\Delta) \mid_W^M \neg\phi$ is an instance of the Minimum Axiom Schema (as Γ_i is finite and so $\Gamma_i - \mathcal{T}(\Delta)$ is also finite) which is equivalent to $\mathcal{O}, \Delta, \Gamma_i - \mathcal{T}(\Delta) \mid_W^M \psi_i$ and this last result is a proof for $\mathcal{O}, \Delta, \Gamma_i - \mathcal{T}(\Delta) \vdash_W^M \psi_i$.

This proves the claim.

By the claim, it follows that $\mathcal{O}, \Delta, \Gamma_n - \mathcal{T}(\Delta) \vdash_W^M \psi_n$ which is equivalent to $\mathcal{O}, \Delta, \emptyset \vdash_W^M \psi$ (as $\Gamma_n \subseteq \mathcal{T}(\Delta)$ and $\psi_n = \psi$). \square

A simple example of a theory suitable for this proof system is the following:

Example 3.1. *We will consider a toy Blocks World Theory. Let theory W , where the types *Block* and *Table* are enumeration types:*

TYPES

$$Block \stackrel{\text{def}}{=} \{A, B\} \quad Table \stackrel{\text{def}}{=} \{T\} \quad Object \stackrel{\text{def}}{=} Block \cup Table$$

OBSERVATION PREDICATES

$$on : Block \times Object \quad red : Block \quad green : Block$$

ABSTRACTION PREDICATES

$$above : Block \times Object$$

AXIOMS

$$\forall x : Block \forall y : Object \quad on(x, y) \rightarrow above(x, y)$$

$$\forall x : Block \forall y : Block \forall z : Object \quad above(x, y) \wedge above(y, z) \rightarrow above(x, z)$$

$$\forall x : Block \quad \neg green(x) \leftrightarrow red(x)$$

The set of observations \mathcal{O} will be the set of atomic sentences without equality that can be formed from observation predicates in signature \mathcal{L} .

Now, let Δ be a finite state description. By Theorem 3.2 we only need to worry for those state descriptions that guarantee the existence of a minimum model with respect to W . So, we will assume that Δ is W -consistent. Also, observe that given some W -consistent Δ , $W \cup \Delta$ will not be able to prove an observation that is not stated in Δ . The reason is that the first two axioms cannot prove observation statements and neither negated color statements. The two sentences can prove negated *on* statements, but those are still not useful to prove negated color statements in W or more observation statements in W . The third axiom can prove positive color statements, but only if we can prove a negated color statement which cannot be done from Δ in theory W . Also, the third axiom can prove negated color statements, but those are not useful to prove more observation

statements in theory W . Therefore, any W -consistent Δ is \mathcal{O} -closed with respect to W or equivalently W is \mathcal{O} -independent.

Therefore, W can be used in Υ^M with any finite state description $\Delta \subseteq \mathcal{O}$ (including state descriptions that do not have minimum models, as Theorem 3.2 still guaranties soundness for those state descriptions).

Now, we will show an example derivation in Υ^M . Let $\Delta = \{on(B, A), on(A, T), red(A), green(B)\}$. We will show a proof of $\forall x: Block \neg on(x, x)$.

1. $\mathcal{O}, \Delta, \emptyset \mid_W^M \neg on(A, A)$ (By Minimum Axiom Schema as: $on(A, A) \in \mathcal{O}$ and $on(A, A) \notin \Delta$).
2. $\mathcal{O}, \Delta, \emptyset \mid_W^M \neg on(B, B)$ (By Minimum Axiom Schema as: $on(B, B) \in \mathcal{O}$ and $on(B, B) \notin \Delta$).
3. $\mathcal{O}, \Delta, \emptyset \mid_W^M \forall x: Block x = A \vee x = B$ (By Reflexivity Axiom Schema as: it is a type enumeration axiom).
4. $\mathcal{O}, \Delta, \emptyset \mid_W^M x = A \vee x = B$ (By universal specification).
5. $\mathcal{O}, \Delta, \{x = A\} \mid_W^M \neg on(A, A)$ (By Monotonicity and 1).
6. $\mathcal{O}, \Delta, \{x = A\} \mid_W^M x = A$ (By Reflexivity Axiom Schema).
7. $\mathcal{O}, \Delta, \{x = A\} \mid_W^M \neg on(x, x)$ (By Equality Substitution in 5 and 6).
8. $\mathcal{O}, \Delta, \emptyset \mid_W^M x = A \rightarrow \neg on(x, x)$ (By implication introduction and 7).
9. $\mathcal{O}, \Delta, \{x = B\} \mid_W^M \neg on(B, B)$ (By Monotonicity and 2).
10. $\mathcal{O}, \Delta, \{x = B\} \mid_W^M x = B$ (By Reflexivity Axiom Schema).
11. $\mathcal{O}, \Delta, \{x = B\} \mid_W^M \neg on(x, x)$ (By Equality Substitution in 9 and 10).
12. $\mathcal{O}, \Delta, \emptyset \mid_W^M x = B \rightarrow \neg on(x, x)$ (By implication introduction and 11).
13. $\mathcal{O}, \Delta, \emptyset \mid_W^M \neg on(x, x)$ (By disjunction elimination with 4, 8, and 12).
14. $\mathcal{O}, \Delta, \emptyset \mid_W^M \forall x: Block \neg on(x, x)$ (By generalization with 13: as x is not free in $W \cup \Delta$).

3.3 Discussion: Difficulty of minimum model reasoning

Theorem 3.2 suggest that if state descriptions are not \mathcal{O} -closed (assuming there exist a minimum model under that state description), the proof system Υ^M needs to be changed in order to preserve soundness.

An obvious question that arises is: Does there exist an effective proof system which is still sound and complete for a general finite state description? It turns

out that the answer depends on the existence of certain kind of sets. Before stating the result, we state some remarks:

Remark 3.2. *Technically speaking, the concept of recursiveness applies to subsets of natural numbers, but as long as there exists some effective coding of a general set A of objects to a set of natural numbers A' , we can talk of the general set A as being recursive as long as A' is recursive. In what follows I will assume that there exists some coding function of formulae in a typed, first-order signature into natural numbers (this is the so called Gödel number of the formula). Therefore, I will speak of sets of formulae as being recursive without specifying Gödel numbers for its elements.*

Remark 3.3. *In the same way as formulae can be coded into natural numbers, proof expressions (as proofs are finite and a proof expression is a finite sequence of symbols) can be coded into natural numbers. I will assume that such coding exists without further mention. As a remark, a proof expression is a sequence of the form $\Gamma_1 \mid \psi_1 \dots \Gamma_n \mid \psi_n$ such that for $1 \leq i \leq n$, $\Gamma_i \subseteq \text{Form}(\mathcal{L})$ are finite and $\psi_i \in \text{Form}(\mathcal{L})$ for some typed, first-order signature \mathcal{L} (here, I am also assuming that finite sets are being coded as finite sequences).*

Remark 3.4. *Given a proof system Υ^X and sets $\Lambda_1, \dots, \Lambda_n \subseteq \text{Sen}(\mathcal{L})$ that serve as parameters in proof expressions of the system Υ^X , I will denote as $\text{Prov}_{\Lambda_1, \dots, \Lambda_n}$ the set*

$$\{(\Omega, \psi) \mid \Omega \text{ is a proof expression of the formula } \psi \text{ in the system } \Upsilon^X\}$$

Then, it is said that a proof system Υ^X is effective if $\text{Prov}_{\Lambda_1, \dots, \Lambda_n}$ is a recursive set.

And now the theorem.

Theorem 3.4. *IF there exists a typed, countable, first-order signature \mathcal{J} , recursive set $R \subseteq \text{Sen}(\mathcal{J})$, recursive set $\mathcal{P} \subseteq \text{Sen}(\mathcal{J})$ and finite $\Lambda \subseteq \mathcal{P}$ such that there is a minimum model in $\text{Mod}_R(\Lambda)$ and $\mathcal{P} \cap \{\psi \in \text{Sen}(\mathcal{J}) \mid \Lambda \not\models_R \psi\}$ is NOT recursively enumerable THEN*

There is no proof system Υ^X such that for any typed, countable, first-order signature \mathcal{L} , any $W \subseteq \text{Sen}(\mathcal{L})$, any $\mathcal{O} \subseteq \text{Sen}(\mathcal{L})$ and any finite $\Delta \subseteq \mathcal{O}$:

- *If W and \mathcal{O} are recursive sets, then Υ^X is effective (i.e. the set $\text{Prov}_{W, \mathcal{O}, \Delta}$ is recursive where W , \mathcal{O} and Δ act as parameters of proof expressions of*

the form $\mathcal{O}, \Delta, \Gamma_1 \mid_W^X \psi_1 \dots \mathcal{O}, \Delta, \Gamma_n \mid_W^X \psi_n$ such that for $1 \leq i \leq n$, $\Gamma_i \subseteq \text{Form}(\mathcal{L})$ are finite and $\psi_i \in \text{Form}(\mathcal{L})$.

- Υ^X is sound and complete with respect to minimum model semantics (i.e. for any $\psi \in \text{Sen}(\mathcal{L})$: $\mathcal{O}, \Delta, \emptyset \vdash_W^X \psi \iff \Delta \approx_W \psi$)

Proof. Suppose the hypothesis of the theorem holds and suppose such proof system exists.

As \mathcal{P} and R are recursive, the set $\text{Prov}_{R, \mathcal{P}, \Lambda}$ is recursive by assumption of the existence of the proof system. Therefore, the set $\text{NegProv}_{R, \mathcal{P}, \Lambda} = \{(\Omega, \psi) \mid \psi \in \text{Sen}(\mathcal{J}) \text{ and } (\Omega, \neg\psi) \in \text{Prov}_{R, \mathcal{P}, \Lambda}\}$ is also recursive (as $\text{Sen}(\mathcal{J})$ is recursive). Then, the set $\{\psi \mid \exists \Omega (\Omega, \psi) \in \text{NegProv}_{R, \mathcal{P}, \Lambda}\} = \{\psi \in \text{Sen}(\mathcal{J}) \mid \mathcal{P}, \Lambda, \emptyset \vdash_R^X \neg\psi\}$ is recursively enumerable (recursively enumerable sets can be formed from recursive sets by using existential quantifiers). This implies that $\mathcal{P} \cap \{\psi \in \text{Sen}(\mathcal{J}) \mid \mathcal{P}, \Lambda, \emptyset \vdash_R^X \neg\psi\}$ is a recursively enumerable set (as \mathcal{P} is recursive and therefore recursively enumerable, and recursively enumerable sets are closed under intersection).

Claim: For any typed, countable, first-order signature \mathcal{K} , any $H \subseteq \text{Sen}(\mathcal{K})$, any $\mathcal{Q} \subseteq \text{Sen}(\mathcal{K})$ and any finite $\Phi \subseteq \mathcal{Q}$ such that there exists a minimum model in $\text{Mod}_H(\Phi)$:

$$\forall \phi \in \mathcal{Q} \quad \mathcal{Q}, \Phi, \emptyset \vdash_H^X \neg\phi \iff \Phi \not\models_H \phi$$

Proof of the claim. Let $\phi \in \mathcal{Q}$.

\implies . Suppose $\neg\phi \notin \{\neg\psi \mid \psi \in \mathcal{Q} \text{ and } \Phi \not\models_H \psi\}$. Then, $\phi \notin \mathcal{Q}$ or $\Phi \models_H \phi$. Then, $\Phi \models_H \phi$ (as $\phi \in \mathcal{Q}$). Which implies that $\mathcal{T}(\Phi) \models_H \phi$. Now, by assumption, $\mathcal{Q}, \Phi, \emptyset \vdash_H^X \neg\phi$, which implies that $\mathcal{T}(\Phi) \models_H \neg\phi$ (As Υ^X is sound, and by Corollary 2.2). Therefore $\mathcal{T}(\Phi) \models_H \perp$. But this contradicts the existence of a minimum model in $\text{Mod}_H(\Phi)$, as $\mathcal{T}(\Phi)$ must be H -consistent by Corollary 2.1.

So, $\neg\phi \in \{\neg\psi \mid \psi \in \mathcal{Q} \text{ and } \Phi \not\models_H \psi\}$. Therefore, $\Phi \not\models_H \phi$.

\impliedby . Suppose $\Phi \not\models_H \phi$. As $\phi \in \mathcal{Q}$, then $\neg\phi \in \{\neg\psi \mid \psi \in \mathcal{Q} \text{ and } \Phi \not\models_H \psi\}$ and so $\neg\phi \in \mathcal{T}(\Phi)$. Therefore $\mathcal{T}(\Phi) \models_H \neg\phi$. As Υ^X is complete and by Corollary 2.2, $\mathcal{Q}, \Phi, \emptyset \vdash_H^X \neg\phi$ holds.

This proves the claim.

By last claim and hypothesis of the theorem, $\forall \phi \in \mathcal{P} \quad \mathcal{P}, \Lambda, \emptyset \vdash_R^X \neg\phi \iff \Lambda \not\models_R \phi$ which is a rephrased form of $\mathcal{P} \cap \{\psi \in \text{Sen}(\mathcal{J}) \mid \mathcal{P}, \Lambda, \emptyset \vdash_R^X \neg\psi\} =$

$\mathcal{P} \cap \{\psi \in \text{Sen}(\mathcal{J}) \mid \Lambda \not\vdash_R \phi\}$. This implies that $\mathcal{P} \cap \{\psi \in \text{Sen}(\mathcal{J}) \mid \Lambda \not\vdash_R \phi\}$ is a recursively enumerable set (as $\mathcal{P} \cap \{\psi \in \text{Sen}(\mathcal{J}) \mid \mathcal{P}, \Lambda, \emptyset \vdash_R^X \neg\psi\}$ is recursively enumerable) which contradicts hypothesis of the theorem. Therefore no such proof system Υ^X exists. \square

It is not obvious if the sets described in the hypothesis of Theorem 3.4 exist. If they do not exist, then the theorem is not claiming anything. Even if those sets do not exist and assuming that a proof system exists, it will be very difficult to come up with one that is effective, sound and complete (for general state descriptions and general theories) because of the following:

Let $W \subseteq \text{Sen}(\mathcal{L})$ be a theory for a typed, first-order signature \mathcal{L} and $\Delta \subseteq \mathcal{O}$ a finite state description where $\mathcal{O} \subseteq \text{Sen}(\mathcal{L})$ is a set of possible observations. Suppose we are given an effective, sound and complete proof system Υ^X . Suppose that $\neg\psi \in \text{Sen}(\mathcal{L}) - (W \cup \Delta)$. Now, suppose that the only classical proof of $\neg\psi$ from $\mathcal{T}(\Delta) \cup W$ is a one-step proof that applies the reflexivity axiom schema. This means that $\neg\psi \in \{\neg\phi \mid \phi \in \mathcal{O} \text{ and } \Delta \not\vdash_W \phi\}$ and so $\Delta \not\vdash_W \psi$. Also, since the proof system is complete, $\neg\psi$ can be proved in Υ^X . This implies that the proof system Υ^X is able to decide *implicitly* that $\Delta \not\vdash_W \psi$ (since the proof system is effective). The difficulty in designing an effective, sound and complete proof system is then the need to decide over *non-entailment* (i.e. elements of the set $\{\neg\phi \mid \phi \in \mathcal{O} \text{ and } \Delta \not\vdash_W \phi\}$), even if it is not done in an explicit way inside the proof system. *And it is known that the set of sentences non-entailed by a theory is not even recursively enumerable in general.*

What about proof systems that are effective and sound but incomplete? Take any effective and sound proof system for first-order logic. That system will be effective and sound with respect to minimum model semantics (as $\Delta \vdash_W \psi \implies \Delta \approx_W \psi$ for any $\psi \in \text{Sen}(\mathcal{L})$). Of course, such system will be useless unless it is able to prove sentences that hold under minimum model semantics but cannot be proved classically. So, we are interested in systems that have some “degree of completeness” with respect to minimum model semantics.

It appears that a useful, minimal degree of completeness that a proof system could have is to be able to prove sentences in the set $\{\neg\phi \mid \phi \in \mathcal{O} \text{ and } \Delta \not\vdash_W \phi\}$. Let’s call a system with that property *minimally complete*. If an effective, sound and minimally complete proof system Υ^X exists, we could use it to *try* to see if a sentence of the form $\neg\psi$ is in the set $\{\neg\phi \mid \phi \in \mathcal{O} \text{ and } \Delta \not\vdash_W \phi\}$ and therefore the sentence can be used as an assumption inside a classical proof system in order to

prove minimum model statements (by Corollary 2.2: $\Delta \models_W \beta \iff \Delta \cup \{\neg\phi \mid \phi \in \mathcal{O} \text{ and } \Delta \not\models_W \phi\} \models_W \beta$). The problem with this approach is that if the sentence $\neg\psi$ is not in $\{\neg\phi \mid \phi \in \mathcal{O} \text{ and } \Delta \not\models_W \phi\}$, the proof system Υ^X could give no answer, as entailment is in general not recursive (even if Υ^X is effective). But nevertheless, the answer depends on the existence on the same kind of sets as in Theorem 3.4 as the following theorem shows:

Theorem 3.5. *IF there exists a typed, countable, first-order signature \mathcal{J} , recursive set $R \subseteq \text{Sen}(\mathcal{J})$, recursive set $\mathcal{P} \subseteq \text{Sen}(\mathcal{J})$ and finite $\Lambda \subseteq \mathcal{P}$ such that there is a minimum model in $\text{Mod}_R(\Lambda)$ and $\mathcal{P} \cap \{\psi \in \text{Sen}(\mathcal{J}) \mid \Lambda \not\models_R \psi\}$ is NOT recursively enumerable THEN*

There is no proof system Υ^X such that for any typed, countable, first-order signature \mathcal{L} , any $W \subseteq \text{Sen}(\mathcal{L})$, any $\mathcal{O} \subseteq \text{Sen}(\mathcal{L})$ and any finite $\Delta \subseteq \mathcal{O}$:

- *If W and \mathcal{O} are recursive sets, then Υ^X is effective (i.e. the set $\text{Prov}_{W,\mathcal{O},\Delta}$ is recursive where W , \mathcal{O} and Δ act as parameters of proof expressions of the form $\mathcal{O}, \Delta, \Gamma_1 \mid_W^X \psi_1 \dots \mathcal{O}, \Delta, \Gamma_n \mid_W^X \psi_n$ such that for $1 \leq i \leq n$, $\Gamma_i \subseteq \text{Form}(\mathcal{L})$ are finite and $\psi_i \in \text{Form}(\mathcal{L})$).*
- *Υ^X is sound (i.e. for any $\psi \in \text{Sen}(\mathcal{L})$: $\mathcal{O}, \Delta, \emptyset \vdash_W^X \psi \implies \Delta \models_W \psi$) and minimally complete (i.e. for any $\psi \in \{\neg\phi \mid \phi \in \mathcal{O} \text{ and } \Delta \not\models_W \phi\}$: $\mathcal{O}, \Delta, \emptyset \vdash_W^X \psi$).*

Proof. The argument is identical as that of Theorem 3.4. So, it suffices to show the required claim:

Claim: For any typed, countable, first-order signature \mathcal{K} , any $H \subseteq \text{Sen}(\mathcal{K})$, any $\mathcal{Q} \subseteq \text{Sen}(\mathcal{K})$ and any finite $\Phi \subseteq \mathcal{Q}$ such that there exists a minimum model in $\text{Mod}_H(\Phi)$:

$$\forall \phi \in \mathcal{Q} \quad \mathcal{Q}, \Phi, \emptyset \vdash_H^X \neg\phi \iff \Phi \not\models_H \phi$$

Proof of the claim. Let $\phi \in \mathcal{Q}$.

\implies . Identical as in the proof of the corresponding claim of Theorem 3.4.

\impliedby . Suppose $\Phi \not\models_H \phi$. As $\phi \in \mathcal{Q}$, then $\neg\phi \in \{\neg\psi \mid \psi \in \mathcal{Q} \text{ and } \Phi \not\models_H \psi\}$. As Υ^X is minimally complete, $\mathcal{Q}, \Phi, \emptyset \vdash_H^X \neg\phi$ holds.

This proves the claim.

The rest of the argument is identical as in the proof of Theorem 3.4. □

So, assuming the sets of the hypothesis of Theorem 3.5 exist and we still want effectiveness and soundness (for general state descriptions and general theories), the best we can do is to build a proof system that can prove strict subsets of $\{\neg\phi \mid \phi \in \mathcal{O} \text{ and } \Delta \not\equiv_W \phi\}$.

It appears that the results of Theorems 3.4 and 3.5 depend on the fact that the set \mathcal{O} of possible observations can contain any kind of sentence. We could think that by restricting the form of the observations we could be able to build a proof system with the required properties. The fact is that by restricting only the form of the observations to a class of sentences that at least allows sentences of the form $P(c)$ where P is an unary predicate symbol and c is a constant symbol, we still have the same kind of problems as Theorem 3.6 will show. But before proving the theorem, we require the following lemma:

Lemma 3.1. *Let class $Sen = \{\psi \mid \psi \text{ is a sentence in a typed, countable, first-order signature}\}$. Let class $\mathcal{C} \subseteq Sen$ such that $\{P(c) \mid P \text{ is an unary predicate symbol, } c \text{ is a constant symbol and } P(c) \in Sen\} \subseteq \mathcal{C}$.*

Let \mathcal{J} be a typed, countable, first-order signature. Let $R \subseteq Sen(\mathcal{J})$ be a recursive set. Let $\mathcal{P} \subseteq Sen(\mathcal{J})$ be a recursive set, and let $\Lambda \subseteq \mathcal{P}$ be finite. IF there is a minimum model in $\mathbf{Mod}_R(\Lambda)$ and $\mathcal{P} \cap \{\psi \in Sen(\mathcal{J}) \mid \Lambda \not\equiv_R \psi\}$ is NOT recursively enumerable THEN

There exists a typed, countable, first-order signature \mathcal{J}' , recursive set $R' \subseteq Sen(\mathcal{J}')$, recursive set $\mathcal{P}' \subseteq Sen(\mathcal{J}')$ such that $\mathcal{P}' \subseteq \mathcal{C}$ and finite $\Lambda' \subseteq \mathcal{P}'$ such that there is a minimum model in $\mathbf{Mod}_{R'}(\Lambda')$ and $\mathcal{P}' \cap \{\psi \in Sen(\mathcal{J}') \mid \Lambda' \not\equiv_{R'} \psi\}$ is not recursively enumerable.

Proof. Suppose the hypothesis. Let τ be a new type not in signature \mathcal{J} . Let Q be a new unary predicate symbol not in \mathcal{J} such that Q has type signature τ . Let M be a recursive countably infinite set of new constant symbols not in \mathcal{J} such that each one of them has type signature τ , and let $m : \mathbb{N} \rightarrow M$ be a corresponding recursive bijection that enumerates the elements of M . Let \mathcal{J}' be the signature \mathcal{J} together with type τ , the symbol Q and the constants in M .

Let $\mathcal{A} = \{Q \cdot (\cdot b \cdot) \mid b \in M\}$ where \cdot denotes symbol concatenation. Therefore $\mathcal{A} \subseteq Sen(\mathcal{J}')$ and \mathcal{A} is recursive. Let $a : \mathbb{N} \rightarrow \mathcal{A}$ be defined as $a(i) = Q \cdot (\cdot m(i) \cdot)$ where \cdot denotes symbol concatenation and m is the function defined above. Then, a is a recursive bijection that enumerates the elements of \mathcal{A} . Also, observe that as \mathcal{A} is contained in the class $\{P(c) \mid P \text{ is an unary predicate symbol, } c \text{ is a constant symbol and } P(c) \in Sen\}$, it follows that $\mathcal{A} \subseteq \mathcal{C}$.

Let $s : \mathbb{N} \rightarrow \text{Sen}(\mathcal{J})$ be a recursive bijection that enumerates the elements of $\text{Sen}(\mathcal{J})$ (It exists as $\text{Sen}(\mathcal{J})$ is countably infinite and recursive).

Let $\mathcal{P}' = \{a(s^{-1}(\psi)) \mid \psi \in \mathcal{P}\}$. Let $\Lambda' = \{a(s^{-1}(\psi)) \mid \psi \in \Lambda\}$. Let $R' = R \cup \{a(s^{-1}(\psi)) \cdot \leftrightarrow \cdot \psi \mid \psi \in \mathcal{P}\}$ where \cdot denotes symbol concatenation. Observe that by construction Λ' is finite, $a(s^{-1}(\mathcal{P})) = \mathcal{P}'$, $s(a^{-1}(\mathcal{P}')) = \mathcal{P}$, $R' = R \cup \{a(s^{-1}(\psi)) \cdot \leftrightarrow \cdot \psi \mid \psi \in \mathcal{P}\} = R \cup \{\psi \cdot \leftrightarrow \cdot s(a^{-1}(\psi)) \mid \psi \in \mathcal{P}'\}$, and $\Lambda' = a(s^{-1}(\Lambda)) \subseteq a(s^{-1}(\mathcal{P})) = \mathcal{P}'$ (as $\Lambda \subseteq \mathcal{P}$ and a, s are bijections). So, $\mathcal{P}' \subseteq \mathcal{A} \subseteq \mathcal{C}$, $\Lambda' \subseteq \mathcal{P}'$, $R' \subseteq \text{Sen}(\mathcal{J}')$. Also, it can be easily checked that \mathcal{P}' and R' are recursive sets as \mathcal{P} , R and \mathcal{A} are recursive sets and a, s are recursive bijections.

Claim 1: $\forall \psi \in \mathcal{P}, \Lambda \models_R \psi$ (in signature \mathcal{J}) $\iff \Lambda' \models_{R'} a(s^{-1}(\psi))$ (in signature \mathcal{J}')

Proof of the claim. Let $\psi \in \mathcal{P}$.

\implies . Let \mathfrak{A} be an \mathcal{J}' -structure that satisfies $\Lambda' \cup R'$. We want to show that $\mathfrak{A} \models a(s^{-1}(\psi))$. First, we show that $\mathfrak{A} \models \Lambda$. Let $\phi \in \Lambda$. Then $a(s^{-1}(\phi)) \in \Lambda'$ and so $\mathfrak{A} \models a(s^{-1}(\phi))$ as $\mathfrak{A} \models \Lambda' \cup R'$. Also $\mathfrak{A} \models a(s^{-1}(\phi)) \cdot \leftrightarrow \cdot \phi$ as $a(s^{-1}(\phi)) \cdot \leftrightarrow \cdot \phi \in R'$ by construction. Therefore $\mathfrak{A} \models \phi$. So, $\mathfrak{A} \models \Lambda$ and also $\mathfrak{A} \models R$ as $R \subseteq R'$. As $\Lambda \cup R$ does not mention symbols Q and the constants in M , the restriction of \mathfrak{A} to signature \mathcal{J} (i.e. $\mathfrak{A} \upharpoonright \mathcal{J}$) also satisfies $\Lambda \cup R$. So, $\mathfrak{A} \upharpoonright \mathcal{J} \models \psi$ (as $\Lambda \models_R \psi$ by assumption). As ψ does not mention symbols Q and the constants in M (as $\psi \in \mathcal{P}$), $\mathfrak{A} \models \psi$, and as $\mathfrak{A} \models a(s^{-1}(\psi)) \cdot \leftrightarrow \cdot \psi$ ($a(s^{-1}(\psi)) \cdot \leftrightarrow \cdot \psi \in R'$ by construction and $\mathfrak{A} \models \Lambda' \cup R'$) $\mathfrak{A} \models a(s^{-1}(\psi))$ as required.

\impliedby . Let \mathfrak{A} be a \mathcal{J} -structure that satisfies $\Lambda \cup R$. We want to show that $\mathfrak{A} \models \psi$. Let T be a countably infinite set which is not a domain in \mathfrak{A} . Let $t : \mathbb{N} \rightarrow T$ be a bijection that enumerates the elements of T .

Let the \mathcal{J}' -structure \mathfrak{A}^+ be the structure \mathfrak{A} augmented with interpretations for the extra symbols in signature \mathcal{J}' in the following way: the domain of type τ is T , for $i \in \mathbb{N} : [m(i)]^{\mathfrak{A}^+} = t(i)$ (i.e. the interpretation of the i th constant symbol in M is the i th element of T), and $Q^{\mathfrak{A}^+} = \{t(i) \mid i \in \mathbb{N} \text{ and } \mathfrak{A} \models s(i)\}$.

First, we show that $\mathfrak{A}^+ \models \Lambda'$. Let $\phi \in \Lambda'$, then $\phi = a(s^{-1}(\gamma))$ where $\gamma \in \Lambda$. Therefore $\mathfrak{A} \models \gamma$ (as $\mathfrak{A} \models \Lambda \cup R$). So, $t(s^{-1}(\gamma)) \in Q^{\mathfrak{A}^+}$ and therefore $\mathfrak{A}^+ \models Q \cdot ([m(s^{-1}(\gamma))]) = a(s^{-1}(\gamma)) = \phi$ (as $\mathfrak{A}^+ \models Q \cdot ([m(s^{-1}(\gamma))]) \iff [m(s^{-1}(\gamma))]^{\mathfrak{A}^+} = t(s^{-1}(\gamma)) \in Q^{\mathfrak{A}^+}$).

Second, we show that $\mathfrak{A}^+ \models R'$. As $\mathfrak{A} \models \Lambda \cup R$ and R does not mention symbols Q and the constants in M , then $\mathfrak{A}^+ \models R$. Let $\phi \in R' - R$. Then, $\phi = a(s^{-1}(\gamma)) \cdot \leftrightarrow$

$\cdot\gamma$ where $\gamma \in \mathcal{P}$. But $\mathfrak{A}^+ \models a(s^{-1}(\gamma)) \leftrightarrow \cdot\gamma$ if and only if $(\mathfrak{A}^+ \models a(s^{-1}(\gamma)))$ iff $\mathfrak{A}^+ \models \gamma$. So, let $\mathfrak{A}^+ \models a(s^{-1}(\gamma)) = Q \cdot ([m(s^{-1}(\gamma))])$. So, $[m(s^{-1}(\gamma))]^{\mathfrak{A}^+} = t(s^{-1}(\gamma)) \in Q^{\mathfrak{A}^+}$ and therefore $\mathfrak{A} \models s(s^{-1}(\gamma)) = \gamma$. As γ does not mention symbols Q and the constants in M (as $\gamma \in \mathcal{P}$), $\mathfrak{A}^+ \models \gamma$. For the other way around, let $\mathfrak{A}^+ \models \gamma$. As γ does not mention symbols Q and the constants in M (as $\gamma \in \mathcal{P}$) then $\mathfrak{A} \models \gamma$. So, $t(s^{-1}(\gamma)) \in Q^{\mathfrak{A}^+}$ and therefore $\mathfrak{A}^+ \models Q \cdot ([m(s^{-1}(\gamma))]) = a(s^{-1}(\gamma))$ (as $\mathfrak{A}^+ \models Q \cdot ([m(s^{-1}(\gamma))]) \iff [m(s^{-1}(\gamma))]^{\mathfrak{A}^+} = t(s^{-1}(\gamma)) \in Q^{\mathfrak{A}^+}$).

Therefore $\mathfrak{A}^+ \models \Lambda' \cup R'$ and $\mathfrak{A}^+ \models a(s^{-1}(\psi))$ as $\Lambda' \models_{R'} a(s^{-1}(\psi))$ by assumption. Now, $\mathfrak{A}^+ \models a(s^{-1}(\psi)) \leftrightarrow \cdot\psi$ because $a(s^{-1}(\psi)) \leftrightarrow \cdot\psi \in R'$ by construction and $\mathfrak{A}^+ \models R'$ and so $\mathfrak{A}^+ \models \psi$. As ψ does not mention symbols Q and the constants in M then $\mathfrak{A} \models \psi$ as required.

This proves Claim 1.

Claim 2: There is a \mathcal{J}' -structure which is a minimum model in $\text{Mod}_{R'}(\Lambda')$.

Proof of the claim. By assumption of the hypothesis of the theorem, there is a minimum model in $\text{Mod}_R(\Lambda)$. By Corollary 2.1 there is a \mathcal{J} -structure \mathfrak{A} that satisfies $\mathcal{T}(\Lambda) \cup R$. We want to show that there is a \mathcal{J}' -structure that satisfies $\mathcal{T}(\Lambda') \cup R'$. Let \mathfrak{A}^+ be the \mathcal{J}' -structure that is constructed from the structure \mathfrak{A} in an identical way as the one constructed in the second part of the proof of Claim 1. So, $\mathfrak{A}^+ \models \Lambda' \cup R'$ as shown in the second part of the proof of Claim 1. It suffices to show that $\mathfrak{A}^+ \models \{\neg\psi \mid \psi \in \mathcal{P}' \text{ and } \Lambda' \not\models_{R'} \psi\}$. Let $\phi \in \{\neg\psi \mid \psi \in \mathcal{P}' \text{ and } \Lambda' \not\models_{R'} \psi\}$, then $\phi = \neg\psi$ for $\psi \in \mathcal{P}'$ and $\Lambda' \not\models_{R'} \psi$. Since $\psi \in \mathcal{P}'$ then $\psi = a(s^{-1}(\gamma))$ where $\gamma \in \mathcal{P}$. So, $\Lambda' \not\models_{R'} \psi = a(s^{-1}(\gamma))$ and therefore, by Claim 1, $\Lambda \not\models_R \gamma$. This implies that $\neg\gamma \in \mathcal{T}(\Lambda)$ as $\gamma \in \mathcal{P}$ and so $\mathfrak{A} \models \neg\gamma$ (as $\mathfrak{A} \models \mathcal{T}(\Lambda) \cup R$). As γ does not mention symbols Q and the constants in M (because $\gamma \in \mathcal{P}$) then $\mathfrak{A}^+ \models \neg\gamma$ and so $\mathfrak{A}^+ \not\models \gamma$. Now, $\mathfrak{A}^+ \models a(s^{-1}(\gamma)) \leftrightarrow \cdot\gamma$ because $a(s^{-1}(\gamma)) \leftrightarrow \cdot\gamma \in R'$ by construction and $\mathfrak{A}^+ \models R'$. Therefore $\mathfrak{A}^+ \not\models a(s^{-1}(\gamma))$ and so $\mathfrak{A}^+ \models \neg \cdot a(s^{-1}(\gamma)) = \neg\psi = \phi$ as required.

This proves Claim 2.

Claim 3: The set $\mathcal{P}' \cap \{\psi \in \text{Sen}(\mathcal{J}') \mid \Lambda' \not\models_{R'} \psi\}$ is not recursively enumerable.

Proof of the claim. Suppose it is. Then, there is a recursive function $h : \mathbb{N} \rightarrow \text{Sen}(\mathcal{J}')$ such that $\text{ran } h = \mathcal{P}' \cap \{\psi \in \text{Sen}(\mathcal{J}') \mid \Lambda' \not\models_{R'} \psi\}$. Let $f : \mathbb{N} \rightarrow \text{Sen}(\mathcal{J})$ be defined as $f(i) = s(a^{-1}(h(i)))$. Note that f is well-defined because $\text{ran } h \subseteq \mathcal{P}' \subseteq \mathcal{A}$. Also, f is a recursive function (as a, s are recursive bijections and h is

recursive). I'll show that $\text{ran } f = \mathcal{P} \cap \{\psi \in \text{Sen}(\mathcal{J}) \mid \Lambda \not\equiv_R \psi\}$.

Case: $\text{ran } f \subseteq \mathcal{P} \cap \{\psi \in \text{Sen}(\mathcal{J}) \mid \Lambda \not\equiv_R \psi\}$. Let $\phi \in \text{ran } f$, then there is $i \in \mathbb{N}$ such that $f(i) = \phi$ and so $s(a^{-1}(h(i))) = \phi$ which implies that $h(i) = a(s^{-1}(\phi))$. Therefore $a(s^{-1}(\phi)) \in \text{ran } h$ and so $a(s^{-1}(\phi)) \in \mathcal{P}'$ (which implies that $\phi \in \mathcal{P}$ by construction and also $\phi \in \text{Sen}(\mathcal{J})$ as $\mathcal{P} \subseteq \text{Sen}(\mathcal{J})$) and $\Lambda' \not\equiv_{R'} a(s^{-1}(\phi))$. Now, by Claim 1, $\Lambda \not\equiv_R \phi$ and therefore $\phi \in \mathcal{P} \cap \{\psi \in \text{Sen}(\mathcal{J}) \mid \Lambda \not\equiv_R \psi\}$ as required.

Case: $\mathcal{P} \cap \{\psi \in \text{Sen}(\mathcal{J}) \mid \Lambda \not\equiv_R \psi\} \subseteq \text{ran } f$. Let $\phi \in \mathcal{P}$, $\phi \in \text{Sen}(\mathcal{J})$ and $\Lambda \not\equiv_R \phi$. By Claim 1, $\Lambda' \not\equiv_{R'} a(s^{-1}(\phi))$. Also, $a(s^{-1}(\phi)) \in \mathcal{P}'$ as $\phi \in \mathcal{P}$. So, $a(s^{-1}(\phi)) \in \mathcal{P}' \cap \{\psi \in \text{Sen}(\mathcal{J}') \mid \Lambda' \not\equiv_{R'} \psi\}$ and therefore $a(s^{-1}(\phi)) \in \text{ran } h$ which implies that there is $i \in \mathbb{N}$ such that $h(i) = a(s^{-1}(\phi))$ and so $s(a^{-1}(h(i))) = \phi$ which is equivalent to $f(i) = \phi$. Therefore $\phi \in \text{ran } f$ as required.

Therefore, $\mathcal{P} \cap \{\psi \in \text{Sen}(\mathcal{J}) \mid \Lambda \not\equiv_R \psi\}$ is recursively enumerable as it is the range of recursive function f , but this contradicts hypothesis of the theorem, and so $\mathcal{P}' \cap \{\psi \in \text{Sen}(\mathcal{J}') \mid \Lambda' \not\equiv_{R'} \psi\}$ is not recursively enumerable.

This proves Claim 3.

By Claims 2 and 3, there exists a typed, countable, first-order signature \mathcal{J}' , recursive set $R' \subseteq \text{Sen}(\mathcal{J}')$, recursive set $\mathcal{P}' \subseteq \text{Sen}(\mathcal{J}')$ such that $\mathcal{P}' \subseteq \mathcal{C}$ and finite $\Lambda' \subseteq \mathcal{P}'$ such that there is a minimum model in $\text{Mod}_{R'}(\Lambda')$ and $\mathcal{P}' \cap \{\psi \in \text{Sen}(\mathcal{J}') \mid \Lambda' \not\equiv_{R'} \psi\}$ is not recursively enumerable. \square

Lemma 3.1 shows that if we are given a class of sentences that at least allows statements of the form $P(c)$ where P is a unary predicate symbol and c is a constant symbol, and we are given a theory together with a set of observations that are not in the class, we can transform the set of observations in such a way that they will conform to the restriction of being in the class and we can also transform the theory in such a way that the result will be an “equivalent” theory that has the same properties: like having a minimum model and having a non-recursively enumerable set of non-entailed observations.

The intuition behind the transformation is based on the ability to index the sentences using constant symbols, so that for each sentence there is a unique constant symbol, and then by introducing a unary predicate Q such that $Q(c)$ intuitively reads as “The sentence with index c holds” we can force the set of observations to be inside the class without altering the meaning of the observations. Similarly, the theory is transformed by adding statements of the form “The sentence with index c holds if and only if ψ ” where ψ is the actual sentence with

index c , so that the transformation preserves the meaning of the theory.

Now, the promised theorem follows from Lemma 3.1:

Theorem 3.6. *Let class $Sen = \{\psi \mid \psi \text{ is a sentence in a typed, countable, first-order signature}\}$. Let class $\mathcal{C} \subseteq Sen$ such that $\{P(c) \mid P \text{ is an unary predicate symbol, } c \text{ is a constant symbol and } P(c) \in Sen\} \subseteq \mathcal{C}$.*

IF there exists a typed, countable, first-order signature \mathcal{J} , recursive set $R \subseteq Sen(\mathcal{J})$, recursive set $\mathcal{P} \subseteq Sen(\mathcal{J})$ and finite $\Lambda \subseteq \mathcal{P}$ such that there is a minimum model in $\text{Mod}_R(\Lambda)$ and $\mathcal{P} \cap \{\psi \in Sen(\mathcal{J}) \mid \Lambda \not\vdash_R \psi\}$ is NOT recursively enumerable THEN

There is no proof system Υ^X such that for any typed, countable, first-order signature \mathcal{L} , any $W \subseteq Sen(\mathcal{L})$, any $\mathcal{O} \subseteq Sen(\mathcal{L})$ such that $\mathcal{O} \subseteq \mathcal{C}$ and any finite $\Delta \subseteq \mathcal{O}$:

- *If W and \mathcal{O} are recursive sets, then Υ^X is effective (i.e. the set $\text{Prov}_{W,\mathcal{O},\Delta}$ is recursive where W , \mathcal{O} and Δ act as parameters of proof expressions of the form $\mathcal{O}, \Delta, \Gamma_1 \vdash_W^X \psi_1 \dots \mathcal{O}, \Delta, \Gamma_n \vdash_W^X \psi_n$ such that for $1 \leq i \leq n$, $\Gamma_i \subseteq \text{Form}(\mathcal{L})$ are finite and $\psi_i \in \text{Form}(\mathcal{L})$).*
- *Υ^X is sound (i.e. for any $\psi \in Sen(\mathcal{L})$: $\mathcal{O}, \Delta, \emptyset \vdash_W^X \psi \implies \Delta \approx_W \psi$) and minimally complete (i.e. for any $\psi \in \{\neg\phi \mid \phi \in \mathcal{O} \text{ and } \Delta \not\vdash_W \phi\}$: $\mathcal{O}, \Delta, \emptyset \vdash_W^X \psi$)*

Proof. Assume the hypothesis. By Lemma 3.1, there exists a typed, countable, first-order signature \mathcal{J}' , recursive set $R' \subseteq Sen(\mathcal{J}')$, recursive set $\mathcal{P}' \subseteq Sen(\mathcal{J}')$ such that $\mathcal{P}' \subseteq \mathcal{C}$ and finite $\Lambda' \subseteq \mathcal{P}'$ such that there is a minimum model in $\text{Mod}_{R'}(\Lambda')$ and $\mathcal{P}' \cap \{\psi \in Sen(\mathcal{J}') \mid \Lambda' \not\vdash_{R'} \psi\}$ is not recursively enumerable.

From here, the proof is identical to the proof of Theorem 3.5 by assuming that the proof system Υ^X exists and using the sets \mathcal{P}' , Λ' and R' instead of sets \mathcal{P} , Λ and R after proving the following claim:

Claim: For any typed, countable, first-order signature \mathcal{K} , any $H \subseteq Sen(\mathcal{K})$, any $\mathcal{Q} \subseteq Sen(\mathcal{K})$ such that $\mathcal{Q} \subseteq \mathcal{C}$ and any finite $\Phi \subseteq \mathcal{Q}$ such that there exists a minimum model in $\text{Mod}_H(\Phi)$:

$$\forall \phi \in \mathcal{Q} \quad \mathcal{Q}, \Phi, \emptyset \vdash_H^X \neg\phi \iff \Phi \not\vdash_H \phi$$

Proof of the claim. Identical to the proof of the corresponding claim in Theorem 3.5. \square

Theorem 3.6 shows that restricting the form of the observations alone does not solve the problem. The reason is that, as was shown in Lemma 3.1, the elements in the set of observations can be moved in and out of it by adding a proper interaction (i.e. “if and only if” sentences) to the theory. So, the key point is that *not only the form of the observations needs to be restricted but also the interactions of the observations in the theory.*

One approach for restricting interactions between observations and the theory is by building observations which are not sentences in a typed, first-order signature \mathcal{L} , but expressions from another syntactical universe \mathcal{U} . The theory is still a set of sentences in \mathcal{L} and therefore the trick that was done on Lemma 3.1 cannot be done, as expressions of the form $o \leftrightarrow s$, where o is an observation and s is a sentence, are not well-formed sentences. In this kind of setting, the original theory needs to be partitioned in two: one part A as a set of expressions in the syntactical universe \mathcal{U} and the second part B as a set of sentences in \mathcal{L} . A proof system in this kind of setting must have: inference rules for reasoning in the theory A of \mathcal{U} -expressions, inference rules for reasoning in the theory B of \mathcal{L} -sentences, and inference rules that allow the translation of statements in \mathcal{U} to sentences in \mathcal{L} , and/or sentences in \mathcal{L} to statements in \mathcal{U} .

The problem with this approach is that as soon as the proof system is able to translate statements from one syntactical universe to the other and vice versa, the same kind of trick as the one appearing in Lemma 3.1 can be done: instead of adding expressions of the form $o \leftrightarrow s$, where o is an observation and s is a sentence (as this clearly cannot be done), we add expressions of the form $o' \leftrightarrow s$ to theory B , where o' is the translated observation into a sentence, and the ability of the proof system to translate back and forth will allow the required trick. This implies that if we want to sever the interaction between observations and theory, the proof system in this kind of setting must be “unidirectional” (i.e. translation in one direction or in the other, but not both) or not fully bidirectional.

This unidirectional restriction implies that the proof system is incomplete: Suppose that theory A proves \mathcal{U} -statement α , and requires \mathcal{U} -statement β to prove \mathcal{U} -statement γ . Suppose that theory B is able to prove \mathcal{L} -sentence β' given \mathcal{L} -sentence α' (where β' and α' are the corresponding translations into \mathcal{L} -sentences of the \mathcal{U} -statements α and β). If the proof system were bidirectional, the system

would be able to prove \mathcal{U} -statement γ : First, as α holds, translate α to α' , then conclude β' using theory B , translate β' to β , then conclude γ using theory A . If the proof system can only translate from \mathcal{U} -statements to \mathcal{L} -sentences, then it cannot prove γ .

On this report, I will not explore further the idea of having a different syntactical universe for the observations. It is not clear to me at this point how expressive unidirectional proof systems are, and this could be a subject worth to explore as future work. Another reason is they require changing the definition of satisfaction over sets of observations by typed, first-order structures, as observations are no longer sentences. Also, it seems to me that whatever can be represented in those kind of systems, can be emulated in a pure sentential approach by adding extra functions and predicates, but this is a claim that requires testing. As such, I will try to explore what can be done inside a pure sentential approach.

Another approach for restricting interactions between observations and the theory is by allowing interactions only on a subset A of the theory by assuming that the observational closure of the state description with respect to A cannot prove observations that are not already stated in the closure when we consider the full theory. The part A of the theory is selected in such a way that it has *nice* properties, like being decidable. This is the approach followed in Section 3.4.

Unfortunately, using this approach we cannot drop the restriction on the closure of the state description, as that would imply an unsound proof system with respect to minimum model reasoning. The reason is that minimum model reasoning requires non-entailment testing over the entire theory in order to conclude that the negation of an observation can be entailed, so if only a part of the theory is used to test for non-entailment, the part of the theory that was left-out could still entail the observation, and so the proof system could be proving contradictions when it should not.

To overcome that restriction on the state descriptions, Section 3.5 explores a class of proof systems that do not require assumptions on the state descriptions, but the price to pay is that they are incomplete proof systems and for some proof systems in the class, effectiveness will be restricted as well.

3.4 A proof system that relaxes \mathcal{O} -closedness

This section describes a proof system whose main idea is to decide non-entailment of observations only based on a decidable part of the theory. The decision of what part of the theory is decidable will be based entirely on the form of the sentences in the theory. So, in order to achieve this, a class of sentences \mathcal{I} is defined such that *the class defines a decidable fragment of first-order logic*. We will require that the theory be finite, the set of observations be quantifier-free sentences and we will define the decidable part of a theory W to be $\gamma(W \pitchfork \mathcal{I})$, where $W \pitchfork \mathcal{I} = \{S \subseteq W^U \mid PNF(\bigwedge S) \in \mathcal{I}\}$ (where W^U is an equivalent theory to W but in an untyped, first-order signature, and PNF is a function that computes a Prenex Normal Form of the sentence) and γ is a choice function that selects one subset among the subsets in $W \pitchfork \mathcal{I}$. Also, note that when we say that W^U is an untyped equivalent theory to W we mean: for every $\psi \in Sen(\mathcal{L})$ we have $W \vdash \psi \iff W^U \vdash \psi^U$ where ψ^U is the untyped form of the sentence ψ and \mathcal{L} is the typed signature. The reason for defining the decidable part in this way, is because all of the decidable fragments of first-order logic that will be presented in this section can only test satisfiability of a *single* sentence in Prenex Normal Form expressed in an *untyped* fragment of first-order logic. So, we require that the conjunction of the sentences in the part of the theory we are interested in, be a sentence that can be expressed in the fragment. The choice function is required, as there could exist more than one subset S of W that satisfies $PNF(\bigwedge S) \in \mathcal{I}$.

The restriction on quantifier-free sentences is not completely arbitrary as an intuitive idea of an observation is something that can be measured directly by an action that does not involve iteration of measurements. This statement is clearer if we see the case for the universal quantifier: To test if observation $\forall x : \tau P(x)$ holds, we iterate over the elements of the domain of type τ and measure property P for each element. This provides the justification. We will denote by $QFSen(\mathcal{L})$ the set of quantifier-free sentences of some signature \mathcal{L} . All these considerations will allow testing of non-entailment of observations.

This approach also has the advantage of allowing us to relax a bit the \mathcal{O} -closed condition that was required in the proof system of Section 3.2. Now, we will require that the observational deductive closure of the state description with respect to $\gamma(W \pitchfork \mathcal{I})$, be \mathcal{O} -closed with respect to W , where W is the theory and \mathcal{I} is the class mentioned above. So, if there exist sentences in the theory W that express dependencies between observations, as long as the dependencies are in

the class \mathcal{I} , the proof system will be able to handle state descriptions that are not completely specified (i.e. non-ideal state descriptions).

So, let \mathcal{I} be a class of sentences that defines a decidable fragment of first-order logic and γ a choice function. Let $\Upsilon^{M^2}(\mathcal{I}, \gamma)$ be the proof system Υ^M of Section 3.2 but with the Minimum Axiom Schema changed to:

$$\frac{\psi \in \Theta \quad \Lambda \not\vdash_{\gamma(W \cap \mathcal{I})} \psi}{\Theta, \Lambda, \Gamma \mid_W^{M^2} \neg \psi} \quad (*)$$

Where $\not\vdash$ is the classical un-provability relation of first-order logic.

Examples of classes \mathcal{I} that define a decidable fragment of first-order logic [with equality] can be found in [3]. The classes of sentences presented in [3] require that every sentence in the class is in Prenex Normal Form, there are no constant symbols in the sentences of the class, and there are no predicate symbols with arity 0 (except the nullary predicates *true* and *false*).

The restriction on the constant symbols is not really necessary, as given one of those classes \mathcal{I} , we can construct another class \mathcal{I}' such that it “emulates” constant symbols. For this emulation to work, we require that the number of constant symbols to emulate be finite, say N , and that these symbols be ordered in a finite sequence. Then, sentences in the class will be those sentences in Prenex Normal Form *without constant symbols* but that satisfy the regular expression $\exists^N \lambda$ where \exists^N means N instances of existential quantifiers at the front and λ is a formula that satisfies all the restrictions in the class \mathcal{I} . The idea is to identify the i -th variable in the existential quantifier with the i -th constant symbol in the finite sequence. The intuition behind this is that constants behave quite similar to “global” individuals inside theories. And so, if λ represents a theory, $\exists^N \lambda$ is stating that there exists N individuals that can be referenced globally inside the theory λ . Also, decidable classes in [3] allow adding existential quantifiers to the front of a sentence in Prenex Normal Form and so, the construction that is being done at the class \mathcal{I}' will be defining a decidable fragment of first-order logic if \mathcal{I} is defining one in the first place. Therefore, from now on, I will assume that decidable classes presented in this section implicitly allow constants.

Some examples of decidable classes in [3] follows. The following two classes require that sentences in the class mention only one and the same function symbol, and this function symbol needs to have arity 1:

- Any Prefix is allowed. Predicate symbols have arity 1. Equality symbol IS allowed. (Rabin 1969)

- Prefix satisfies the regular expression: $\exists^*\forall\exists^*$. Equality symbol IS allowed. (Shelah 1977)

The following classes are also decidable as they have the *finite model property*, and whenever function symbols are allowed, there are no restrictions in the number of function symbols allowed:

- Prefix satisfies the regular expression: $\exists^*\forall^*$. Sentence has no function symbols. Equality symbol IS allowed. (Bernays, Schönfinkel 1928)
- Prefix satisfies the regular expression: $\exists^*\forall^2\exists^*$. Sentence has no function symbols. Equality symbol is NOT allowed. (Gödel 1932, Kalmár 1933, Schütte 1934)
- Any Prefix is allowed. Predicate and function symbols have arity 1. Equality symbol is NOT allowed. (Löb 1967, Gurevich 1969)
- Prefix satisfies the regular expression: $\exists^*\forall\exists^*$. Equality symbol is NOT allowed. (Gurevich 1973)
- Prefix satisfies the regular expression: \exists^* . Equality symbol IS allowed. (Gurevich 1976)

Remark 3.5. *A fragment \mathcal{I} of first-order logic has the finite model property, if for every satisfiable sentence σ expressed in the fragment \mathcal{I} we have that σ is satisfiable by a finite model.*

Remark 3.6. *Satisfiability in a fragment \mathcal{I} with the finite model property is decidable. Given a sentence ψ in the fragment, the following decision procedure test satisfiability of the sentence:*

Test in parallel: $\vdash \neg\psi$ and “there is a finite model that satisfies ψ ”

If first condition ends first: Output NO

If second condition ends first: Output YES

The procedure always ends. The reasons are:

- *The relation \vdash is always recursively enumerable (no matter if we are using a fragment of first-order logic or not).*

- The relation “there is a finite model that satisfies ψ ” is also recursively enumerable, just enumerate in some order all finite structures of the signature used by the sentence.
- If $\not\vdash \neg\psi$, then the first condition will never end, but the second condition will end: $\not\vdash \neg\psi$ implies that there is a model that does not satisfy $\neg\psi$ or equivalently, there is a model that satisfies ψ and therefore, as ψ is in the fragment, by the finite model property it follows that there is a finite model that satisfies ψ .
- If $\vdash \neg\psi$ then the first condition will end but the second condition will not: $\vdash \neg\psi$ implies that every model satisfies $\neg\psi$, or equivalently, every model does not satisfy ψ , and in particular every finite model will not satisfy ψ .

Now, an example on the usage of a decidable fragment:

Example 3.2. Let theory W be, where τ is the only type in the theory:

CONSTANTS

$0 : \tau \quad 1 : \tau$

OBSERVATION PREDICATES

$O_1 : \tau \times \tau \quad O_2 : \tau \times \tau$

ABSTRACTION PREDICATES

$A : \tau$

AXIOMS

$\forall x : \tau [O_1(x, x) \rightarrow \exists y : \tau O_2(y, x)]$

$\forall x : \tau [O_1(x, 1) \rightarrow A(0)]$

$\exists x : \tau O_2(x, 1) \wedge A(x)$

The set of observations is the set of atomic sentences without equality formed from observation predicates. Take the class \mathcal{I} with prefix $\exists^*\forall^2\exists^*$, where equality symbol is not allowed and no function symbols are allowed. What are the biggest sets in $W \cap \mathcal{I}$ that contain the type restrictions?

First, we need to transform the theory into its untyped form, for that, we introduce a new predicate symbol τ representing the type:

$$\begin{array}{l}
\tau(0) \quad \tau(1) \quad \forall x \tau(x) \\
\forall x [\tau(x) \rightarrow O_1(x, x) \rightarrow \exists y [\tau(y) \wedge O_2(y, x)]] \\
\forall x [\tau(x) \rightarrow O_1(x, 1) \rightarrow A(0)] \\
\exists x [\tau(x) \wedge O_2(x, 1) \wedge A(x)]
\end{array}$$

From this, after taking the conjunction of the sentences on each subset and computing the Prenex Normal Form of this conjunction, it should be clear that the biggest sets in $W \uplus \mathcal{I}$ that contain the type restrictions are:

$$\begin{array}{ll}
\tau(0) & \tau(0) \\
\tau(1) & \tau(1) \\
\forall x \tau(x) & \forall x \tau(x) \\
\forall x [\tau(x) \rightarrow O_1(x, x) \rightarrow \exists y [\tau(y) \wedge O_2(y, x)]] & \forall x [\tau(x) \rightarrow O_1(x, 1) \rightarrow A(0)] \\
\exists x [\tau(x) \wedge O_2(x, 1) \wedge A(x)] & \exists x [\tau(x) \wedge O_2(x, 1) \wedge A(x)]
\end{array}$$

The entire theory is not in the class, because the Prenex Normal Form of the conjunction of the sentences in the theory will have three universal quantifiers and we are only allowed two. Which subset should the choice function γ select? One criteria could be: select the one that guaranties the \mathcal{O} -closedness for state descriptions. In our case, the set on the left would be a better choice as we will be leaving out the sentence $\forall x [\tau(x) \rightarrow O_1(x, 1) \rightarrow A(0)]$ which can only conclude abstraction statements and negated observation statements which are not useful to prove more observation statements in the entire theory, and so the \mathcal{O} -closedness property will be guaranteed.

Returning to the proof system $\Upsilon^{M^2}(\mathcal{I}, \gamma)$, a *proof* will be a finite sequence of sequents $\Theta, \Lambda, \Gamma_1 \mid_W^{M^2} \psi_1 \dots \Theta, \Lambda, \Gamma_n \mid_W^{M^2} \psi_n$ where each $\Gamma_i \subseteq \text{Form}(\mathcal{L})$ is finite and each $\psi_i \in \text{Form}(\mathcal{L})$. Also, each sequent $\Theta, \Lambda, \Gamma_i \mid_W^{M^2} \psi_i$ in the proof is either an instance of an axiom or the conclusion of an instance of an inference rule of the following form, where $j_t < i$ for $1 \leq t \leq n$ and each $\Theta, \Lambda, \Gamma_{j_t} \mid_W^{M^2} \psi_{j_t}$ is a sequent occurring previously in the proof:

$$\frac{\Theta, \Lambda, \Gamma_{j_1} \mid_W^{M^2} \psi_{j_1} \quad \dots \quad \Theta, \Lambda, \Gamma_{j_n} \mid_W^{M^2} \psi_{j_n}}{\Theta, \Lambda, \Gamma_i \mid_W^{M^2} \psi_i}$$

Similarly, the notation $\Theta, \Lambda, \Gamma \vdash_W^{M^2} \psi$ means that there exists a proof $\Theta, \Lambda, \Gamma_1 \mid_W^{M^2} \psi_1 \dots \Theta, \Lambda, \Gamma_n \mid_W^{M^2} \psi_n$ such that $\Gamma_n \subseteq \Gamma$ and $\psi_n = \psi$.

Given a theory $W \subseteq \text{Sen}(\mathcal{L})$, possible observations $\mathcal{O} \subseteq \text{Sen}(\mathcal{L})$ and state description $\Delta \subseteq \mathcal{O}$, we will denote by $\mathcal{T}(\Delta)$ the set $\Delta \cup \{\neg\psi \mid \psi \in \mathcal{O} \text{ and } \Delta \not\#_W \psi\}$

as was defined in Theorem 2.1. In addition, given a decidable class \mathcal{I} and choice function γ , we will denote by Δ^* the set $\{\psi \in \mathcal{O} \mid \Delta \vDash_{\gamma(W \cap \mathcal{I})} \psi\}$.

Theorem 3.7 (Soundness for \mathcal{O} -closedness in $\Upsilon^{M2}(\mathcal{I}, \gamma)$). *Let $\mathcal{O} \subseteq QFSen(\mathcal{L})$ and $W \subseteq Sen(\mathcal{L})$ a finite theory of \mathcal{L} . Let W^U be the corresponding equivalent theory in the untyped signature \mathcal{L}^U . Then, for every $\psi \in Sen(\mathcal{L})$ and every finite $\Delta \subseteq \mathcal{O}$, such that Δ^* is \mathcal{O} -closed with respect to W :*

$$\mathcal{O}, \Delta, \emptyset \vdash_W^{M2} \psi \implies \Delta \vDash_W \psi$$

Proof. Suppose $\mathcal{O}, \Delta, \emptyset \vdash_W^{M2} \psi$. By Corollary 2.2 and the completeness theorem for first-order logic, it suffices to show that $\mathcal{T}(\Delta) \vdash_W \psi$. By assumption, there is a proof $\mathcal{O}, \Delta, \Gamma_1 \mid_W^{M2} \psi_1 \dots \mathcal{O}, \Delta, \Gamma_n \mid_W^{M2} \psi_n$ such that $\Gamma_n = \emptyset$ and $\psi_n = \psi$.

Claim: $\mathcal{O}, \Delta, \Gamma_i \mid_W^{M2} \psi_i \implies \mathcal{T}(\Delta) \cup \Gamma_i \vdash_W \psi_i$ for $1 \leq i \leq n$. By induction on the length of the proof, we consider each axiom and inference rule in $\Upsilon^{M2}(\mathcal{I}, \gamma)$. It suffices to show the case for the Minimum Axiom Schema.

So, for $\mathcal{O}, \Delta, \Gamma_i \mid_W^{M2} \psi_i$ to be an instance of the Minimum Axiom Schema, $\psi_i = \neg\phi$ for some $\phi \in \mathcal{O}$ such that $\Delta \not\vDash_{\gamma(W \cap \mathcal{I})} \phi$, and so $\Delta \not\vDash_{\gamma(W \cap \mathcal{I})} \phi$ by completeness of first-order logic (observe that here we switched to signature \mathcal{L}^U , as Δ and $\phi \in \mathcal{O}$ are quantifier-free sentences, they are well-formed in \mathcal{L}^U ; this will be a common technique in this section: as long as we work with observations, as they are quantifier-free, it is safe to move between signatures \mathcal{L} and \mathcal{L}^U as long as the form of the observation is not changed when working in signature \mathcal{L}^U). Therefore $\phi \notin \Delta^*$. Since Δ^* is \mathcal{O} -closed with respect to W , by definition $\phi \notin \{\psi \mid \psi \in \mathcal{O} \text{ and } \Delta^* \vDash_W \psi\}$ which means that $\Delta^* \not\vDash_W \phi$ (as $\phi \in \mathcal{O}$). So, $\Delta \not\vDash_W \phi$ (if not, $\Delta \vDash_W \phi$ would imply $\Delta^* \vDash_W \phi$ by monotonicity which is a contradiction). This in turn implies that $\neg\phi \in \mathcal{T}(\Delta) = \Delta \cup \{\neg\psi \mid \psi \in \mathcal{O} \text{ and } \Delta \not\vDash_W \psi\}$ (i.e. $\psi_i \in \mathcal{T}(\Delta)$), and therefore $\mathcal{T}(\Delta) \cup \Gamma_i \vdash_W \psi_i$.

This proves the claim.

By the claim, it follows that $\mathcal{T}(\Delta) \cup \Gamma_n \vdash_W \psi_n$ which is equivalent to $\mathcal{T}(\Delta) \vdash_W \psi$ (as $\Gamma_n = \emptyset$ and $\psi_n = \psi$). \square

What is the minimum property a finite subset $\Delta \subseteq \mathcal{O}$ needs to have in order to guarantee soundness in the proof system $\Upsilon^{M2}(\mathcal{I}, \gamma)$? The following theorem shows that.

Theorem 3.8 (Characterising soundness in $\Upsilon^{M2}(\mathcal{I}, \gamma)$). *Let $\mathcal{O} \subseteq QFSen(\mathcal{L})$,*

$W \subseteq \text{Sen}(\mathcal{L})$ a finite theory of \mathcal{L} and $\Delta \subseteq \mathcal{O}$ be finite. Let W^U be the corresponding equivalent theory in the untyped signature \mathcal{L}^U . Then, $\forall \psi \in \text{Sen}(\mathcal{L}) \ \mathcal{O}, \Delta, \emptyset \vdash_W^{M2} \psi \implies \Delta \approx_W \psi$ IF AND ONLY IF there is no minimum model in $\text{Mod}_W(\Delta)$ or Δ^* is \mathcal{O} -closed with respect to W .

Proof. \implies . Suppose $\forall \psi \in \text{Sen}(\mathcal{L}) \ \mathcal{O}, \Delta, \emptyset \vdash_W^{M2} \psi \implies \Delta \approx_W \psi$. Suppose there is a minimum model in $\text{Mod}_W(\Delta)$. We'll show that Δ^* is \mathcal{O} -closed with respect to W .

Let $\phi \in \{\psi \mid \psi \in \mathcal{O} \text{ and } \Delta^* \vDash_W \psi\}$. Then, $\phi \in \mathcal{O}$ and $\Delta^* \vDash_W \phi$. We'll prove by contradiction that $\phi \in \Delta^*$, so suppose that $\phi \notin \Delta^*$, therefore by definition $\phi \notin \{\psi \in \mathcal{O} \mid \Delta \vDash_{\gamma(W \cap \mathcal{I})} \psi\}$ which means that $\Delta \not\vDash_{\gamma(W \cap \mathcal{I})} \phi$ (as $\phi \in \mathcal{O}$) which by completeness of first-order logic $\Delta \not\vDash_{\gamma(W \cap \mathcal{I})} \phi$. Then, $\mathcal{O}, \Delta, \emptyset \vdash_W^{M2} \neg\phi$ is an instance of the Minimum Axiom Schema, and therefore $\mathcal{O}, \Delta, \emptyset \vdash_W^{M2} \neg\phi$. Therefore, by assumption of soundness, $\Delta \approx_W \neg\phi$ and $\mathcal{T}(\Delta) \vdash_W \neg\phi$ by Corollary 2.2.

We know that $\Delta^* \vDash_W \phi$ and we want to show that $\mathcal{T}(\Delta) \vDash_W \phi$. Let α be an \mathcal{L} -structure such that $\alpha \vDash \mathcal{T}(\Delta) \cup W$, we want to show that $\alpha \vDash \phi$. As $\alpha \vDash \mathcal{T}(\Delta) \cup W$, $\alpha \vDash \Delta \cup W$. Therefore α satisfies the deductive closure of $\Delta \cup W$. First, we'll show that $\alpha \vDash \Delta^*$. Let $\gamma \in \Delta^* = \{\psi \in \mathcal{O} \mid \Delta \vDash_{\gamma(W \cap \mathcal{I})} \psi\}$. So, $\Delta \vDash_{\gamma(W \cap \mathcal{I})} \gamma$, and $\Delta \vDash_{W^U} \gamma$ by monotonicity (as $\gamma(W \cap \mathcal{I}) \subseteq W^U$). But W^U is equivalent to W and $\gamma \in \mathcal{O} \subseteq QFSen(\mathcal{L})$ and so $\Delta \vDash_W \gamma$. Therefore, γ is in the deductive closure of $\Delta \cup W$ and so $\alpha \vDash \gamma$ as required. So, $\alpha \vDash \Delta^*$ and therefore $\alpha \vDash \phi$ as required (as $\Delta^* \vDash_W \phi$ and $\alpha \vDash W$).

Therefore $\mathcal{T}(\Delta) \vDash_W \phi$ and so $\mathcal{T}(\Delta) \vdash_W \phi$ by completeness of first-order logic. So, $\mathcal{T}(\Delta) \vdash_W \perp$ (as previously was shown that $\mathcal{T}(\Delta) \vdash_W \neg\phi$). But, since there is a minimum model in $\text{Mod}_W(\Delta)$, $\mathcal{T}(\Delta)$ is W -consistent (by Corollary 2.1) which contradicts $\mathcal{T}(\Delta) \vdash_W \perp$. Therefore $\phi \in \Delta^*$ as required.

\Leftarrow . Suppose there is no minimum model in $\text{Mod}_W(\Delta)$ or Δ^* is \mathcal{O} -closed with respect to W . The case for \mathcal{O} -closed Δ^* follows directly by Theorem 3.7. For the other case, since there is no minimum model in $\text{Mod}_W(\Delta)$, then trivially, for any $\psi \in \text{Sen}(\mathcal{L})$, $\Delta \approx_W \psi$. Therefore $\forall \psi \in \text{Sen}(\mathcal{L}) \ \mathcal{O}, \Delta, \emptyset \vdash_W^{M2} \psi \implies \Delta \approx_W \psi$ since the conclusion does not depend on the hypothesis of the implication. \square

The following theorem shows that completeness does not require any restriction on the state descriptions.

Theorem 3.9 (Completeness for $\Upsilon^{M2}(\mathcal{I}, \gamma)$). *Let $\mathcal{O} \subseteq QFSen(\mathcal{L})$ and $W \subseteq \text{Sen}(\mathcal{L})$ a finite theory of \mathcal{L} . Let W^U be the corresponding equivalent theory in*

the untyped signature \mathcal{L}^U . Then, for every $\psi \in \text{Sen}(\mathcal{L})$ and every finite $\Delta \subseteq \mathcal{O}$:

$$\mathcal{O}, \Delta, \emptyset \vdash_W^{M2} \psi \iff \Delta \vDash_W \psi$$

Proof. Suppose $\Delta \vDash_W \psi$. By Corollary 2.2 and the completeness theorem for first-order logic, $\mathcal{T}(\Delta) \vdash_W \psi$ holds. Then, there is a proof $\Gamma_1 \mid_W \psi_1 \dots \Gamma_n \mid_W \psi_n$ such that $\Gamma_n \subseteq \mathcal{T}(\Delta)$, $\psi_n = \psi$ and each Γ_i is finite.

Claim: $\Gamma_i \mid_W \psi_i \implies \mathcal{O}, \Delta, \Gamma_i - \mathcal{T}(\Delta) \vdash_W^{M2} \psi_i$ for $1 \leq i \leq n$. By induction on the length of the proof, we consider each axiom and inference rule in Υ . It suffices to show the case for the Reflexivity Axiom Schema.

So, for $\Gamma_i \mid_W \psi_i$ to be an instance of the Reflexivity Axiom Schema, either $\psi_i \in \Gamma_i$ or $\psi_i \in W$ must hold. If $\psi_i \in W$ the result follows trivially by the Reflexivity Axiom Schema in $\Upsilon^{M2}(\mathcal{I}, \gamma)$. Now, for the case when $\psi_i \in \Gamma_i$, either $\psi_i \notin \mathcal{T}(\Delta)$ or $\psi_i \in \mathcal{T}(\Delta)$. If $\psi_i \notin \mathcal{T}(\Delta)$ then $\psi_i \in \Gamma_i - \mathcal{T}(\Delta)$ and the result follows trivially by the Reflexivity Axiom Schema in $\Upsilon^{M2}(\mathcal{I}, \gamma)$. If $\psi_i \in \mathcal{T}(\Delta)$ then either $\psi_i \in \Delta$ or $\psi_i \in \{\neg\psi \mid \psi \in \mathcal{O} \text{ and } \Delta \not\vDash_W \psi\}$. If $\psi_i \in \Delta$ the result follows trivially by the Reflexivity Axiom Schema in $\Upsilon^{M2}(\mathcal{I}, \gamma)$. For the other case, $\psi_i = \neg\phi$ for some $\phi \in \mathcal{O}$ such that $\Delta \not\vDash_W \phi$. But W^U is equivalent to W and $\phi \in \mathcal{O} \subseteq QFSen(\mathcal{L})$ which implies that $\phi \in QFSen(\mathcal{L}^U)$ and so $\Delta \not\vDash_{W^U} \phi$. If $\Delta \vDash_{\gamma(W \cap \mathcal{I})} \phi$ were true, then $\Delta \vDash_{W^U} \phi$ would be true by monotonicity, which is a contradiction, therefore $\Delta \not\vDash_{\gamma(W \cap \mathcal{I})} \phi$ and so $\mathcal{O}, \Delta, \Gamma_i - \mathcal{T}(\Delta) \mid_W^{M2} \neg\phi$ is an instance of the Minimum Axiom Schema (as Γ_i is finite and so $\Gamma_i - \mathcal{T}(\Delta)$ is also finite) which is equivalent to $\mathcal{O}, \Delta, \Gamma_i - \mathcal{T}(\Delta) \mid_W^{M2} \psi_i$ and this last result is a proof for $\mathcal{O}, \Delta, \Gamma_i - \mathcal{T}(\Delta) \vdash_W^{M2} \psi_i$.

This proves the claim.

By the claim, it follows that $\mathcal{O}, \Delta, \Gamma_n - \mathcal{T}(\Delta) \vdash_W^{M2} \psi_n$ which is equivalent to $\mathcal{O}, \Delta, \emptyset \vdash_W^{M2} \psi$ (as $\Gamma_n \subseteq \mathcal{T}(\Delta)$ and $\psi_n = \psi$). \square

Effectiveness follows:

Theorem 3.10 (Effectiveness for $\Upsilon^{M2}(\mathcal{I}, \gamma)$). *Let \mathcal{L} be a typed, first-order signature with a finite number of constant symbols. Let recursive $\mathcal{O} \subseteq QFSen(\mathcal{L})$, recursive $W \subseteq Sen(\mathcal{L})$ be a finite theory of \mathcal{L} , and finite $\Delta \subseteq \mathcal{O}$. If class \mathcal{I} is recursive and the choice function γ is recursive, then $\Upsilon^{M2}(\mathcal{I}, \gamma)$ is effective (i.e, the set $Prov_{W, \mathcal{O}, \Delta}$ is recursive where W , \mathcal{O} and Δ act as parameters of proof expressions of the form $\mathcal{O}, \Delta, \Gamma_1 \mid_W^{M2} \psi_1 \dots \mathcal{O}, \Delta, \Gamma_n \mid_W^{M2} \psi_n$ such that for $1 \leq i \leq n$, $\Gamma_i \subseteq Form(\mathcal{L})$ are finite and $\psi_i \in Form(\mathcal{L})$)*

Proof. It suffices to show that the condition of the Minimum Axiom Schema in $\Upsilon^{M^2}(\mathcal{I}, \gamma)$ is recursive.

For $\psi \in \text{Form}(\mathcal{L})$: $\psi \in \mathcal{O}$ is a recursive condition as \mathcal{O} is recursive. So, first test if $\psi \in \mathcal{O}$, if not, the Minimum Axiom Schema does not apply, if yes, then we know that ψ is a quantifier-free sentence. Now, do the following:

It should be clear that transforming W into W^U is recursive and W^U will be finite: for each type τ add a unary predicate τ . If type τ is an enumeration type, add sentences indicating that all constants in the type are distinct and add a sentence indicating that any object satisfies predicate τ if and only if the object is equal to one of the constants in the type (One example of this transformation is shown in Example 3.3). If type τ is a union type, add a sentence indicating that any object satisfies predicate τ if and only if the object satisfies one of the predicates of the types in the union (see Example 3.3). If the type is neither an enumeration nor a union type, then for each constant c of type τ , add the sentence $\tau(c)$. Now, add a sentence indicating that every object is in one of the non-union types (see Example 3.3). Finally, for each sentence in W , substitute any occurrence of $\forall v:\tau$ (where v is a variable and τ is a type) by $\forall v \tau(v) \rightarrow$, and any occurrence of $\exists v:\tau$ by $\exists v \tau(v) \wedge$.

As W^U is finite, it has a finite number of subsets: for each subset S of W^U , build the sentence $\bigwedge S$ and test if $PNF(\bigwedge S) \in \mathcal{I}$. There exists an algorithm for computing a Prenex Normal Form of a sentence, constructing $\bigwedge S$ is recursive as S is finite and testing membership in \mathcal{I} is recursive as \mathcal{I} is recursive. So, the set $W \cap \mathcal{I}$ is recursive and $\gamma(W \cap \mathcal{I})$ will be computable as γ is recursive.

Now, build the sentence $\phi = PNF([\bigwedge \Delta] \wedge [\bigwedge \gamma(W \cap \mathcal{I})] \wedge [\neg\psi])$ whose construction is recursive as Δ is finite, $\gamma(W \cap \mathcal{I})$ is recursive and PNF is recursive.

Now, observe that by definition, $PNF(\bigwedge \gamma(W \cap \mathcal{I})) \in \mathcal{I}$. Also, as $\bigwedge \Delta$ and $\neg\psi$ are quantifier-free sentences, this means that ϕ will have the same Prefix as $PNF(\bigwedge \gamma(W \cap \mathcal{I}))$ and so $\phi \in \mathcal{I}$.

Finally, as \mathcal{I} defines a decidable fragment of first-order logic, for any sentence in \mathcal{I} we can decide if the sentence is satisfiable. So, test if ϕ is satisfiable. If yes, then there is a model that satisfies $\Delta \cup \gamma(W \cap \mathcal{I})$ but does not satisfy ψ , and therefore $\Delta \not\models_{\gamma(W \cap \mathcal{I})} \psi$. If no, then every model satisfies $\neg\phi$ which is equivalent to saying that $([\bigwedge \Delta] \wedge [\bigwedge \gamma(W \cap \mathcal{I})]) \rightarrow \psi$ is valid and so $\Delta \vdash_{\gamma(W \cap \mathcal{I})} \psi$. \square

One suitable theory for this proof system is the one shown in the following example:

Example 3.3. We will consider a toy Blocks World Theory. Let theory W , where the types *Block* and *Table* are enumeration types:

TYPES

$$\text{Block} \stackrel{\text{def}}{=} \{A, B\} \quad \text{Table} \stackrel{\text{def}}{=} \{T\} \quad \text{Object} \stackrel{\text{def}}{=} \text{Block} \cup \text{Table}$$

OBSERVATION PREDICATES

$$\begin{aligned} \text{on} &: \text{Block} \times \text{Object} & \text{under} &: \text{Object} \times \text{Block} & \text{red} &: \text{Block} \\ \text{green} &: \text{Block} \end{aligned}$$

ABSTRACTION PREDICATES

$$\text{above} : \text{Block} \times \text{Object}$$

AXIOMS

$$\begin{aligned} \forall x: \text{Block} \forall y: \text{Object} \text{on}(x, y) &\rightarrow \text{above}(x, y) \\ \forall x: \text{Block} \forall y: \text{Block} \forall z: \text{Object} \text{above}(x, y) \wedge \text{above}(y, z) &\rightarrow \text{above}(x, z) \\ \forall x: \text{Block} \neg \text{green}(x) \leftrightarrow \text{red}(x) & \\ \forall x: \text{Block} \text{under}(T, x) \rightarrow \text{green}(x) & \\ \forall x: \text{Block} \forall y: \text{Object} \text{on}(x, y) \leftrightarrow \text{under}(y, x) & \end{aligned}$$

The set of observations \mathcal{O} will be the set of atomic sentences without equality that can be formed from observation predicates in signature \mathcal{L} .

First, we will transform the theory into its untyped form. For that, we introduce unary predicates *Block*, *Table* and *Object* to represent the types. Call the untyped theory W^U .

TYPE PREDICATES

$$\text{Block} : \text{unary} \quad \text{Table} : \text{unary} \quad \text{Object} : \text{unary}$$

OBSERVATION PREDICATES

$$\text{on} : \text{binary} \quad \text{under} : \text{binary} \quad \text{red} : \text{unary} \quad \text{green} : \text{unary}$$

ABSTRACTION PREDICATES

$$\text{above} : \text{binary}$$

ENUMERATION TYPE AXIOMS

$$\begin{aligned} \neg A &= B \\ \forall x \text{Block}(x) &\leftrightarrow [x = A \vee x = B] \\ \forall x \text{Table}(x) &\leftrightarrow x = T \\ \forall x \text{Object}(x) &\leftrightarrow [\text{Block}(x) \vee \text{Table}(x)] \end{aligned}$$

$$\forall x \text{ Block}(x) \vee \text{Table}(x)$$

AXIOMS

$$\forall x [\text{Block}(x) \rightarrow \forall y [\text{Object}(y) \rightarrow \text{on}(x, y) \rightarrow \text{above}(x, y)]]$$

$$\forall x [\text{Block}(x) \rightarrow \forall y [\text{Block}(y) \rightarrow \forall z [\text{Object}(z) \rightarrow$$

$$\text{[above}(x, y) \wedge \text{above}(y, z) \rightarrow \text{above}(x, z)]]]]$$

$$\forall x [\text{Block}(x) \rightarrow [\neg \text{green}(x) \leftrightarrow \text{red}(x)]]$$

$$\forall x [\text{Block}(x) \rightarrow \text{under}(T, x) \rightarrow \text{green}(x)]$$

$$\forall x [\text{Block}(x) \rightarrow \forall y [\text{Object}(y) \rightarrow [\text{on}(x, y) \leftrightarrow \text{under}(y, x)]]]$$

We will define our decidable class \mathcal{I} to be very restrictive, as it will allow only enumeration axioms and simple interactions between observations. This will show that \mathcal{I} does not need to be one of the full decidable fragments presented at the start of this section.

Define the class \mathcal{K} to be the class of atomic formulae without equality, formed from observation predicates in the signature of the theory W^U . Define the class \mathcal{QFT} of quantifier-free formulae with equality, formed from type predicates in the signature of the theory W^U . Define the class \mathcal{AT} of atomic formulae without equality, formed from type predicates in the signature of the theory W^U . Define \mathcal{I} to be the smallest class of sentences such that it has sentences in Prenex Normal Form with the form $\exists^* \forall^* \phi$ where ϕ is a quantifier-free formula with the form $\bigwedge C_i$ where each conjunct C_i satisfies one or more of the following conditions:

- $C_i \in \mathcal{K}$.
- $C_i \in \mathcal{QFT}$.
- $C_i = \lambda \rightarrow \tau \rightarrow \dots \rightarrow \eta \rightarrow [\alpha \rightarrow \beta]$ where λ , τ and η and any other condition in the implication before the last implication (i.e. the bracketed implication) are in \mathcal{AT} but α and β are in \mathcal{K} .
- $C_i = \lambda \rightarrow \tau \rightarrow \dots \rightarrow \eta \rightarrow [\alpha \leftrightarrow \beta]$ where λ , τ and η and any other condition in the implication before the bracketed “if and only if” are in \mathcal{AT} but α and β are in \mathcal{K} .

Observe that \mathcal{I} defines a decidable class as it is contained in the class with prefix $\exists^* \forall^*$, equality allowed, no function symbols (see list of decidable classes at the start of this section).

What are the sentences of W^U that are left out by the biggest set in $W \cap \mathcal{I}$?
The following sentences will be left out:

$$\begin{aligned} & \forall x [Block(x) \rightarrow \forall y [Object(y) \rightarrow on(x, y) \rightarrow above(x, y)]] \\ & \forall x [Block(x) \rightarrow \forall y [Block(y) \rightarrow \forall z [Object(z) \rightarrow \\ & \qquad \qquad \qquad [above(x, y) \wedge above(y, z) \rightarrow above(x, z)]]]] \\ & \forall x [Block(x) \rightarrow [\neg green(x) \leftrightarrow red(x)]] \end{aligned}$$

The reasons are: the first two sentences contain abstraction predicates and the last one contains a negated atomic formula in the if and only if part. So, the class \mathcal{I} is big enough to allow type enumeration restrictions, observation statements and some simple interactions between observables, namely, restriction interactions $\alpha \rightarrow \beta$ and renaming interactions $\alpha \leftrightarrow \beta$. Now, define $\gamma(W \cap \mathcal{I})$ to be the biggest set among $W \cap \mathcal{I}$, i.e. it will be the set of all sentences in W^U except the three sentences given above.

Now, let Δ be a finite state description. By Theorem 3.8 we only need to worry for those state descriptions that guarantee the existence of a minimum model with respect to the entire theory W . So, we will assume that Δ is W -consistent. Also, observe that given some W -consistent Δ , $\gamma(W \cap \mathcal{I})$ will not be able to prove a negated color statement, just by the form of the sentences in $\gamma(W \cap \mathcal{I})$. This means that after we add the three sentences we left out, $\Delta^* = \{\psi \in \mathcal{O} \mid \Delta \models_{\gamma(W \cap \mathcal{I})} \psi\}$ will be \mathcal{O} -closed with respect to W . The reason is that the first two sentences that were left out (see sentences above) cannot prove observation statements and neither negated color statements. The two sentences can prove negated on statements, but those are still not useful to prove negated color statements in W or more observation statements in W . The third sentence can prove positive color statements, but only if we can prove a negated color statement which cannot be done from Δ in theory $W - \{\forall x: Block \neg green(x) \leftrightarrow red(x)\}$. Also, the third sentence can prove negated color statements, but those are not useful to prove more observation statements in theory W .

Therefore, W can be used in $\Upsilon^{M^2}(\mathcal{I}, \gamma)$ with any finite state description $\Delta \subseteq \mathcal{O}$.

Now, we will show an example derivation in this setting. Let $\Delta = \{on(B, A), on(A, T), red(B)\}$. The big difference with Example 3.1 is that Δ is not an ideal state description: observe that we are leaving out the observation $green(A)$ and still $\Upsilon^{M^2}(\mathcal{I}, \gamma)$ will not prove $\neg green(A)$ contrary to what the proof system of

Section 3.2 will do, as $\text{green}(A) \notin \Delta$ that proof system will conclude $\neg\text{green}(A)$ which is clearly a contradiction.

We will show a proof of $\exists x : \text{Block } \neg\text{under}(T, x) \wedge \neg\text{green}(x)$. Observe that we can't use the axiom $\forall x : \text{Block } \neg\text{green}(x) \leftrightarrow \text{red}(x)$ to conclude $\neg\text{green}(B)$ from $\text{red}(B)$ as this axiom was left out from $\gamma(W \cap \mathcal{I})$, so we will rely on the Minimum Axiom Schema to conclude $\neg\text{green}(B)$:

1. $\mathcal{O}, \Delta, \emptyset \mid_W^{M^2} \neg\text{under}(T, B)$ (By Minimum Axiom Schema as: $\text{under}(T, B) \in \mathcal{O}$ and $\Delta \not\vdash_{\gamma(W \cap \mathcal{I})} \text{under}(T, B)$ where here, we are assuming that some algorithm is telling us this, as $\gamma(W \cap \mathcal{I})$ is decidable).

2. $\mathcal{O}, \Delta, \emptyset \mid_W^{M^2} \neg\text{green}(B)$ (By Minimum Axiom Schema as: $\text{green}(B) \in \mathcal{O}$ and $\Delta \not\vdash_{\gamma(W \cap \mathcal{I})} \text{green}(B)$).

3. $\mathcal{O}, \Delta, \emptyset \mid_W^{M^2} \neg\text{under}(T, B) \wedge \neg\text{green}(B)$ (By conjunction introduction using 1 and 2).

4. $\mathcal{O}, \Delta, \emptyset \mid_W^{M^2} \exists x : \text{Block } \neg\text{under}(T, x) \wedge \neg\text{green}(x)$ (By existential introduction in 3).

Another proof of the same sentence goes like this:

1. $\mathcal{O}, \Delta, \emptyset \mid_W^{M^2} \neg\text{green}(B)$ (By Minimum Axiom Schema as: $\text{green}(B) \in \mathcal{O}$ and $\Delta \not\vdash_{\gamma(W \cap \mathcal{I})} \text{green}(B)$).

2. $\mathcal{O}, \Delta, \emptyset \mid_W^{M^2} \forall x : \text{Block } \text{under}(T, x) \rightarrow \text{green}(x)$ (By Reflexivity Axiom Schema: as it is a sentence in W).

3. $\mathcal{O}, \Delta, \emptyset \mid_W^{M^2} \text{under}(T, B) \rightarrow \text{green}(B)$ (By universal specification with 2).

4. $\mathcal{O}, \Delta, \emptyset \mid_W^{M^2} \neg\text{green}(B) \rightarrow \neg\text{under}(T, B)$ (By contrapositive equivalence with 3).

5. $\mathcal{O}, \Delta, \emptyset \mid_W^{M^2} \neg\text{under}(T, B)$ (By Modus Ponens with 1 and 4).

6. $\mathcal{O}, \Delta, \emptyset \mid_W^{M^2} \neg\text{under}(T, B) \wedge \neg\text{green}(B)$ (By conjunction introduction using 5 and 1).

7. $\mathcal{O}, \Delta, \emptyset \mid_W^{M^2} \exists x : \text{Block } \neg\text{under}(T, x) \wedge \neg\text{green}(x)$ (By existential introduction in 6).

3.5 A class of unrestricted proof systems

The main disadvantage of the proof systems in Sections 3.2 and 3.4 is that if the state description is not \mathcal{O} -closed (Theorem 3.2) or the observational closure of the state description with respect to a part of the theory is not \mathcal{O} -closed with respect to the full theory (Theorem 3.8) then the systems become unsound. Applications of minimum model reasoning involve changing the theory dynamically [1]. As the theory changes, it could be very hard to test if the \mathcal{O} -closedness condition still holds after the changes in the theory. This does not mean that the proof systems described on past sections are completely useless, but certainly their applicability is limited. This suggests future research on what changes in a theory preserve the \mathcal{O} -closedness on each of the proof systems already presented.

This section takes another route and presents a class of proof systems which have no restrictions on the state descriptions Δ . The proof systems are sound and have some “degree of completeness” in the sense of being able to proof elements of the set $\mathcal{T}(\Delta)$ when the theory has some specific form, but the price to pay is that limited completeness only applies for some very limited cases as we will see later on.

Let \Vdash be a relation in $\mathcal{P}(\text{Sen}) \times \mathcal{P}(\text{Sen}) \times \text{Sen}$ where class $\text{Sen} = \{\psi \mid \psi \text{ is a sentence in a typed, countable, first-order signature}\}$ and $\mathcal{P}(\text{Sen})$ is the power class of Sen . Let us call \Vdash *admissible* if for all $\Delta \subseteq \text{Sen}$, $W \subseteq \text{Sen}$, $\varphi \in \text{Sen}$: $\Delta \Vdash_W \varphi \implies \Delta \not\vdash_W \varphi$. If in addition, \Vdash is also recursive we will say that \Vdash is a recursive admissible relation.

For every admissible \Vdash relation, let $\Upsilon^{M3}(\Vdash)$ be the proof system Υ^M of Section 3.2 but with the Minimum Axiom Schema changed to:

$$\frac{\psi \in \Theta \quad \Lambda \Vdash_W \psi}{\Theta, \Lambda, \Gamma \mid_W^{M3} \neg\psi} \quad (*)$$

So that every \Vdash is defining a distinct proof system $\Upsilon^{M3}(\Vdash)$.

A *proof* will be a finite sequence of sequents $\Theta, \Lambda, \Gamma_1 \mid_W^{M3} \psi_1 \dots \Theta, \Lambda, \Gamma_n \mid_W^{M3} \psi_n$ where each $\Gamma_i \subseteq \text{Form}(\mathcal{L})$ is finite and each $\psi_i \in \text{Form}(\mathcal{L})$. Also, each sequent $\Theta, \Lambda, \Gamma_i \mid_W^{M3} \psi_i$ in the proof is either an instance of an axiom or the conclusion of an instance of an inference rule of the following form, where $j_t < i$ for $1 \leq t \leq n$ and each $\Theta, \Lambda, \Gamma_{j_t} \mid_W^{M3} \psi_{j_t}$ is a sequent in the proof:

$$\frac{\Theta, \Lambda, \Gamma_{j_1} \mid_W^{M3} \psi_{j_1} \quad \dots \quad \Theta, \Lambda, \Gamma_{j_n} \mid_W^{M3} \psi_{j_n}}{\Theta, \Lambda, \Gamma_i \mid_W^{M3} \psi_i}$$

Similarly, the notation $\Theta, \Lambda, \Gamma \vdash_W^{M3} \psi$ means that there exists a proof $\Theta, \Lambda, \Gamma_1 \mid_W^{M3} \psi_1 \dots \Theta, \Lambda, \Gamma_n \mid_W^{M3} \psi_n$ such that $\Gamma_n \subseteq \Gamma$ and $\psi_n = \psi$.

Given a theory $W \subseteq \text{Sen}(\mathcal{L})$, possible observations $\mathcal{O} \subseteq \text{Sen}(\mathcal{L})$ and state description $\Delta \subseteq \mathcal{O}$, we will denote by $\mathcal{T}(\Delta)$ the set $\Delta \cup \{\neg\psi \mid \psi \in \mathcal{O} \text{ and } \Delta \not\#_W \psi\}$ as was defined in Theorem 2.1.

Theorem 3.11 (Soundness for $\Upsilon^{M3}(\Vdash)$). *Let $\mathcal{O} \subseteq \text{Sen}(\mathcal{L})$ and $W \subseteq \text{Sen}(\mathcal{L})$ a theory of \mathcal{L} . Then, for every $\psi \in \text{Sen}(\mathcal{L})$ and every finite $\Delta \subseteq \mathcal{O}$:*

$$\mathcal{O}, \Delta, \emptyset \vdash_W^{M3} \psi \implies \Delta \Vdash_W \psi$$

Proof. Suppose $\mathcal{O}, \Delta, \emptyset \vdash_W^{M3} \psi$. By Corollary 2.2 and the completeness theorem for first-order logic, it suffices to show that $\mathcal{T}(\Delta) \vdash_W \psi$. By assumption, there is a proof $\mathcal{O}, \Delta, \Gamma_1 \mid_W^{M3} \psi_1 \dots \mathcal{O}, \Delta, \Gamma_n \mid_W^{M3} \psi_n$ such that $\Gamma_n = \emptyset$ and $\psi_n = \psi$.

Claim: $\mathcal{O}, \Delta, \Gamma_i \mid_W^{M3} \psi_i \implies \mathcal{T}(\Delta) \cup \Gamma_i \vdash_W \psi_i$ for $1 \leq i \leq n$. By induction on the length of the proof, we consider each axiom and inference rule in $\Upsilon^{M3}(\Vdash)$. It suffices to show the case for the Minimum Axiom Schema.

So, for $\mathcal{O}, \Delta, \Gamma_i \mid_W^{M3} \psi_i$ to be an instance of the Minimum Axiom Schema, $\psi_i = \neg\phi$ for some $\phi \in \mathcal{O}$ such that $\Delta \Vdash_W \phi$. Since \Vdash is admissible, $\Delta \not\#_W \phi$ and so $\Delta \#_W \phi$. This in turn implies that $\neg\phi \in \mathcal{T}(\Delta) = \Delta \cup \{\neg\psi \mid \psi \in \mathcal{O} \text{ and } \Delta \not\#_W \psi\}$ (i.e. $\psi_i \in \mathcal{T}(\Delta)$), and therefore $\mathcal{T}(\Delta) \cup \Gamma_i \vdash_W \psi_i$.

This proves the claim.

By the claim, it follows that $\mathcal{T}(\Delta) \cup \Gamma_n \vdash_W \psi_n$ which is equivalent to $\mathcal{T}(\Delta) \vdash_W \psi$ (as $\Gamma_n = \emptyset$ and $\psi_n = \psi$). \square

Effectiveness follows trivially as long as \Vdash is a recursive admissible relation. So, I only state the theorem without proof:

Theorem 3.12 (Effectiveness for $\Upsilon^{M3}(\Vdash)$). *Let recursive $\mathcal{O} \subseteq \text{Sen}(\mathcal{L})$, recursive $W \subseteq \text{Sen}(\mathcal{L})$ be a theory of \mathcal{L} , and finite $\Delta \subseteq \mathcal{O}$. If \Vdash is recursive then $\Upsilon^{M3}(\Vdash)$ is effective (i.e, the set $\text{Prov}_{W, \mathcal{O}, \Delta}$ is recursive where W , \mathcal{O} and Δ act as parameters of proof expressions of the form $\mathcal{O}, \Delta, \Gamma_1 \mid_W^{M3} \psi_1 \dots \mathcal{O}, \Delta, \Gamma_n \mid_W^{M3} \psi_n$ such that for $1 \leq i \leq n$, $\Gamma_i \subseteq \text{Form}(\mathcal{L})$ are finite and $\psi_i \in \text{Form}(\mathcal{L})$)*

The heart of the class $\Upsilon^{M3}(\Vdash)$ is then how the relation \Vdash is defined. The following section explores such issue.

3.5.1 Discussion on recursive admissible relations

One way of defining a recursive admissible relation is by means of a decision procedure, given a theory W , finite state description Δ and sentence ψ . If $W \cup \Delta$ is decidable, then the non-provability relation in $W \cup \Delta$ provides a trivial recursive admissible relation. Another trivial case is when we are given a finite theory W , a finite state description Δ , a sentence ψ and a recursive class of sentences \mathcal{I} that defines a decidable fragment of first-order logic. Just define the following decision procedure:

```

if  $\Delta \subseteq QFSen$  and  $\psi \in QFSen$  and  $PNF(\wedge W^U) \in \mathcal{I}$  then
  Let  $\lambda = PNF([\wedge \Delta] \wedge [\wedge W^U] \wedge \neg\psi)$ 
  if  $\lambda$  is satisfiable then
    Output YES
  else
    Output NO
  end if
else
  Output NO
end if

```

The procedure shows that, as in Section 3.4, we are only interested in quantifier-free observations Δ and quantifier-free query ψ . Also, PNF is a function that computes a Prenex Normal Form of a sentence and W^U is the equivalent untyped theory of the theory W (i.e. for every $\psi \in Sen(\mathcal{L})$ we have $W \vdash \psi \iff W^U \vdash \psi^U$ where ψ^U is the untyped form of the sentence ψ and \mathcal{L} is the typed signature).

So, for each of the classes that defines a decidable fragment of first-order logic \mathcal{I} described in Section 3.4, we can build a corresponding recursive admissible relation $\Vdash^{\mathcal{I}}$. The problem with this approach is that if there exist a sentence in $W \cup \Delta$ whose Prenex Normal Form is not in the class \mathcal{I} , the procedure will always answer NO and so the Minimum Axiom Schema will never be used in any proof, and therefore the corresponding proof system $\Upsilon^{M3}(\Vdash^{\mathcal{I}})$ will be equivalent to the classical first-order logic proof system Υ .

So, we want a decision procedure that can say YES in cases when the theory W is not completely inside a class \mathcal{I} . One way of doing this is by “patching” the decision procedure by adding an extra class \mathcal{K} to the sentences that are left out,

i.e. we require that $PNF(\bigwedge[W^U - \gamma(W \cap \mathcal{I})]) \in \mathcal{K}$ (where again, as in the start of Section 3.4, $W \cap \mathcal{I} = \{S \subseteq W^U \mid \bigwedge S \in \mathcal{I}\}$, W^U is the equivalent untyped form of the theory W , γ is a choice function that selects a set among $W \cap \mathcal{I}$ and PNF is a function that computes a Prenex Normal Form of a sentence). The patching needs to make sure that when adding the sentences that were left out, non-provability is still decidable. One way of doing this is by making sure that when we add the left-out sentences, they don't add more quantifier-free provable facts (as we are only interested in quantifier-free queries). One simple example [and completely useless for most cases]: take class \mathcal{I} to be sentences whose Prefix of their Prenex Normal Form satisfies the regular expression $\exists^*\forall^*$ and no function symbols allowed. Therefore, \mathcal{I} is a decidable class (see Section 3.4). Take \mathcal{K} to be the class of sentences which satisfy the regular expression $\forall^*\exists^* \alpha$ where α is a quantifier-free formula with no function symbols.

Continuing with the example, given a finite theory W , finite Δ and sentence ψ , define the following procedure:

```

Let  $L = W^U - \gamma(W \cap \mathcal{I})$ 
Let  $C = \gamma(W \cap \mathcal{I})$ 
Let  $\mathcal{L}$  be the typed signature of  $W$ 
if  $\Delta \subseteq QFSen(\mathcal{L})$  and  $\psi \in QFSen(\mathcal{L})$  and  $PNF(\bigwedge L) \in \mathcal{K}$  then
  for all  $\phi \in L$  do
    Let  $\lambda = PNF([\bigwedge \Delta] \wedge [\bigwedge C] \wedge \neg\phi)$ 
    if  $\lambda$  is satisfiable then
      Output NO
    end if
  end for
  Let  $\eta = PNF([\bigwedge \Delta] \wedge [\bigwedge C] \wedge \neg\psi)$ 
  if  $\eta$  is satisfiable then
    Output YES
  else
    Output NO
  end if
else
  Output NO
end if

```

Observe that by construction of λ inside the FOR loop, the Prenex Normal Form of $\neg\phi$ will be in the class \mathcal{I} (as the Prenex Normal Form of ϕ is in \mathcal{K}) and therefore λ will be in \mathcal{I} and so, we can test its satisfiability. Also, as ψ is quantifier-free, we can test satisfiability of η in the last IF. As W is finite, Δ is finite, \mathcal{I} and \mathcal{K} are recursive classes, and construction of W^U is recursive, the procedure is recursive.

Now, in order to show that the procedure defines an admissible relation, we need to show that whenever the procedure outputs YES, $\Delta \not\vdash_W \psi$ holds. As the procedure outputs YES, this means that Δ contains quantifier-free sentences and ψ is quantifier-free. Also $PNF(\bigwedge L) \in \mathcal{K}$ where $L = W^U - \gamma(W \cap \mathcal{I})$. Also, for every $\phi \in L$ we have $PNF([\bigwedge \Delta] \wedge [\bigwedge \gamma(W \cap \mathcal{I})] \wedge \neg\phi)$ is NOT satisfiable or equivalently $\Delta \vdash_{\gamma(W \cap \mathcal{I})} \phi$. This implies that L is in the deductive closure of $C \cup \Delta$ and so $W^U \cup \Delta = C \cup L \cup \Delta$ is in the deductive closure of $\gamma(W \cap \mathcal{I}) \cup \Delta$ which implies that the deductive closure of $W^U \cup \Delta$ is contained [in fact, equal] in the deductive closure of $\gamma(W \cap \mathcal{I}) \cup \Delta$. Also, we know that $\eta = PNF([\bigwedge \Delta] \wedge [\bigwedge C] \wedge \neg\psi)$ is satisfiable or equivalently $\Delta \not\vdash_{\gamma(W \cap \mathcal{I})} \psi$ and so, ψ will not be in the deductive closure of $W^U \cup \Delta$ or equivalently $\Delta \not\vdash_{W^U} \psi$. But W^U and W are equivalent and $\psi \in QFSen(\mathcal{L})$ therefore $\Delta \not\vdash_W \psi$.

The procedure in this example is not very useful because it will only output YES when the sentences that were left out are already provable from the part of the theory that is in the class, as this will guarantee that no new quantifier-free statements will be provable when we add the sentences that were left out. Even though its limited usefulness, this example shows that it is possible to build a procedure that will output YES even if the theory is not completely inside the class. A more intelligent approach could analyse the form of sentences in both $\gamma(W \cap \mathcal{I})$ and $W^U - \gamma(W \cap \mathcal{I})$ and not only rely on their decidability property. Also, instead of general quantifier-free sentences we could use only atomic observations and try to come up with a recursive test [but maybe incomplete] for \mathcal{O} -closedness in W^U .

It is not clear to me how useful these kind of extensions can be or if there is a way to measure that usefulness or if these extensions actually exist. Also, it is uncertain how complex the decision procedure can become and still defining a recursive admissible relation. All this difficulties arise because non-entailment is not “nice” in the sense that it is a global property, unlike entailment, we require

the entire theory and search all provable possibilities inside the theory, to be sure that something is not provable.

A more interesting approach for generating admissible relations consist on searching for finite models of the theory that falsify the sentence. But for this to define a recursive admissible relation, we require an extra condition. The condition we will impose I will call the *finite representation property*:

Definition 3.3. *Let $\mathcal{C} \subseteq \text{Sen}(\mathcal{L})$ be a non-empty set of sentences in a typed, first-order, countable signature \mathcal{L} . Let $\mathcal{O} \subseteq \mathcal{C}$ and $W \subseteq \text{Sen}(\mathcal{L})$ a finite theory of \mathcal{L} . Let $\Delta \subseteq \mathcal{O}$ be finite. Define the class $\text{FMod}_W(\Delta)$ to be the class of \mathcal{L} -models α that satisfy $W \cup \Delta$ where α has a finite domain for each type. We say that $\Delta \cup W$ has the \mathcal{C} -finite representation property (\mathcal{C} -frp, for short) if for every $\psi \in \mathcal{C}$: $\Delta \vDash_W \psi \iff$ for every $\alpha \in \text{FMod}_W(\Delta)$, $\alpha \vDash \psi$.*

Depending on what the set \mathcal{C} contains, we will say that a theory has the $QFSen(\mathcal{L})$ -frp when $\mathcal{C} = QFSen(\mathcal{L})$ and similarly for the set of atomic sentences $ATSen(\mathcal{L})$. The frp intuitively means that whatever the theory can express (i.e. entails), it is representable in every finite model of the theory (hence the name finite representation). If we are given a finite theory plus finite state description with the \mathcal{C} -frp and the set \mathcal{C} is recursive then the relation defined by the following procedure is recursive and admissible:

```

if  $\Delta \subseteq \mathcal{C}$  and  $\psi \in \mathcal{C}$  then
  Test in parallel:  $\Delta \vdash_W \psi$  and  $\Delta \perp_W \psi$ 
  If first condition ends first: Output NO
  If second condition ends first: Output YES
else
  Output NO
end if

```

The notation $\Delta \perp_W \psi$ means that there is a finite model that satisfies $W \cup \Delta$ but does not satisfy ψ . The procedure always ends. The reasons are:

- The relation \vdash is always recursively enumerable.
- The relation \perp is also recursively enumerable when $\Delta \cup W$ is finite, just enumerate in some order all finite structures of the signature used by the

theory W and test each one of them if it satisfies $\Delta \cup W$ but does not satisfy ψ .

- If $\Delta \not\vdash_W \psi$, then the first condition will never end, but the second condition will end: by the \mathcal{C} -frp: $\Delta \not\vdash_W \psi$ implies that there is a finite model that satisfies $\Delta \cup W$ but does not satisfy ψ , i.e. $\Delta \perp_W \psi$.
- If $\Delta \vdash_W \psi$ then the first condition will end but the second condition will not: by the \mathcal{C} -frp: $\Delta \vdash_W \psi$ implies that every finite model that satisfies $\Delta \cup W$ also satisfies ψ , i.e. $\Delta \perp_W \psi$ does NOT hold.

A trivial example of a theory with the \mathcal{C} -frp (for any set \mathcal{C}) is a theory that only uses finite enumeration types, as the theory will only have finite models. Also, if the Prenex Normal Form of the conjunction of a theory W plus a quantifier-free state description Δ is in a class defining a fragment of first-order logic with the finite model property, then $W \cup \Delta$ has the $QFSen(\mathcal{L})$ -frp (see classes in Section 3.4). But a theory that allows infinite models can also have the \mathcal{C} -frp. One of those cases, and one of the most interesting, is when \mathcal{C} has only atomic sentences and more specifically on those theories that look like a database:

- There is a finite number of constants, functions, predicates and types in the signature.
- The observations are atomic sentences without equality.
- For every function on every model of the theory we have: each element in the image of the function can be represented by a constant in the signature.
- The theory ensures a finite closure of the state description with respect to atomic sentences without equality and without function symbols.

Observe that the last condition always holds, as it follows from the first condition. Also, later on, we will see that the second condition is not necessary, as long as the observations are quantifier-free sentences. Even if those conditions are redundant, I will left them there, as it makes obvious the analogy with the database as follows:

- The condition on the finite number of constants, functions, predicates and types is required as databases only allow a finite range of values for each type and a finite number of tables.

- Observations of the form $P(x_1, \dots, x_n)$ mean that x_1, \dots, x_n is an entry in the table represented by the symbol P .
- The condition on the functions is justified as databases use functions such that when evaluated, the resulting value can be represented by an expression that does not have function symbols (i.e. a constant symbol) before being saved as an entry on a table.
- The condition on the finite closure of the state description with respect to atomic sentences without equality and without function symbols means that each table represented by each predicate symbol has a finite number of entries (i.e. the tables in the database are finite), the condition that there is no function symbols is because in a database, the function is “evaluated” before being inserted as an entry in one column of the table, this is always ensured by the previous condition as there exists a constant representing each value of every function when it is “evaluated”.

We will require the following remark about observations that are atomic sentences, before one example of a database-like theory.

Remark 3.7. *For the case when $\mathcal{C} = ATSen(\mathcal{L})$. Given a theory W and a state description Δ where observations are in \mathcal{C} , one way to prove that $\Delta \cup W$ has the $ATSen(\mathcal{L})$ -frp is by showing that for a general model \mathfrak{A} of $\Delta \cup W$, we can find a suitable model \mathfrak{B} in the class $FMod_W(\Delta)$ such that there is an homomorphism from \mathfrak{B} to \mathfrak{A} . As homomorphisms preserve atomic sentences (i.e. if there is an homomorphism from \mathfrak{B} to \mathfrak{A} , then for every atomic sentence ψ : if $\mathfrak{B} \models \psi$ then $\mathfrak{A} \models \psi$) the $ATSen(\mathcal{L})$ -finite representation property follows trivially.*

And now the example.

Example 3.4. *Let W be the toy Blocks World Theory:*

TYPES

Color Block

CONSTANTS

$c_1, \dots, c_{10} : Block$ $red : Color$ $green : Color$

OBSERVATION PREDICATES

$on : Block \times Block$

ABSTRACTION PREDICATES

$$\textit{above} : \textit{Block} \times \textit{Block}$$

FUNCTIONS

$$f : \textit{Block} \rightarrow \textit{Color}$$

AXIOMS

$$\forall x : \textit{Block} \forall y : \textit{Block} \textit{on}(x, y) \rightarrow \textit{above}(x, y)$$

$$\forall x : \textit{Block} \forall y : \textit{Block} \forall z : \textit{Block} \textit{above}(x, y) \wedge \textit{above}(y, z) \rightarrow \textit{above}(x, z)$$

$$\forall x : \textit{Block} x \neq c_{10} \rightarrow f(x) = \textit{red}$$

$$f(c_{10}) = \textit{green}$$

where the set of observations \mathcal{O} is the set of atomic sentences without equality that can be formed from observation predicates. An example of a state description is $\Delta = \{\textit{on}(c_3, c_2), \textit{on}(c_2, c_1), \textit{on}(c_{10}, c_3)\}$.

Models of theory W allow an infinite domain for types \textit{Block} and \textit{Color} as we are not stating that the constants in the signature are enumerating the type \textit{Block} or the type \textit{Color} . Also, observe that as the function f is completely specified inside the theory, each element of the image of any function on any model that interprets f will be represented by a constant of type \textit{Color} . We will show that, given any finite state description Δ , $W \cup \Delta$ has the $ATSen(\mathcal{L})$ -frp where \mathcal{L} is the signature of theory W given above.

Informal proof: I will follow the suggested strategy described in Remark 3.7. Let \mathfrak{A} be a model of $\Delta \cup W$. We need to build a model \mathfrak{B} such that the domains for types \textit{Block} and \textit{Color} are finite, \mathfrak{B} satisfies $\Delta \cup W$, and there is an homomorphism from \mathfrak{B} to \mathfrak{A} . So, let Δ be a finite state description and let \mathfrak{A} be a model of $\Delta \cup W$.

Build the model \mathfrak{B} as follows: the domain of type \textit{Block} will be the interpretations of the 10 constant symbols in the structure \mathfrak{A} . The domain of type \textit{Color} will be the interpretations of \textit{red} and \textit{green} in the structure \mathfrak{A} . Then, define $\textit{on}^{\mathfrak{B}} = \textit{on}^{\mathfrak{A}} \cap (\textit{Block}^{\mathfrak{B}} \times \textit{Block}^{\mathfrak{B}})$ and similarly $\textit{above}^{\mathfrak{B}} = \textit{above}^{\mathfrak{A}} \cap (\textit{Block}^{\mathfrak{B}} \times \textit{Block}^{\mathfrak{B}})$. Now, define $c_i^{\mathfrak{B}} = c_i^{\mathfrak{A}}$ for $1 \leq i \leq 10$ and $\textit{red}^{\mathfrak{B}} = \textit{red}^{\mathfrak{A}}$, $\textit{green}^{\mathfrak{B}} = \textit{green}^{\mathfrak{A}}$. Finally, define $f^{\mathfrak{B}} = f^{\mathfrak{A}} \upharpoonright \textit{Block}^{\mathfrak{B}}$.

Observe that for $f^{\mathfrak{A}} \upharpoonright \textit{Block}^{\mathfrak{B}}$ to be well-defined we need to show that $\text{ran } f^{\mathfrak{A}} \subseteq \textit{Color}^{\mathfrak{B}}$ and this is where the third database-like condition comes into play: as every element in the image of $f^{\mathfrak{A}}$ can be represented by a constant of type \textit{Color} and $\textit{Color}^{\mathfrak{B}}$ contains all the interpretations in \mathfrak{A} of all constants of type \textit{Color} , the

result follows. But let's do the proof with all the details. Let $x \in \text{ran} f^{\mathfrak{A}}$, then there is $y \in \text{Block}^{\mathfrak{A}}$ such that $f^{\mathfrak{A}}(y) = x$. Now, $y = c_{10}^{\mathfrak{A}}$ or $y \neq c_{10}^{\mathfrak{A}}$. Suppose $y = c_{10}^{\mathfrak{A}}$. We know that $\mathfrak{A} \models [f(c_{10}) = \text{green}]$ and so $f^{\mathfrak{A}}(c_{10}^{\mathfrak{A}}) = f^{\mathfrak{A}}(y) = \text{green}^{\mathfrak{A}} = x$. But $\text{green}^{\mathfrak{A}} = \text{green}^{\mathfrak{B}}$ by construction and so $x \in \text{Color}^{\mathfrak{B}}$. For the other case, $y \neq c_{10}^{\mathfrak{A}}$. We know that $\mathfrak{A} \models [\forall z : \text{Block } z \neq c_{10} \rightarrow f(z) = \text{red}]$ and so $f^{\mathfrak{A}}(y) = \text{red}^{\mathfrak{A}} = x$ and so, again, $x \in \text{Block}^{\mathfrak{B}}$.

Continuing with the example, the inclusion functions $\text{id}_1 : \text{Block}^{\mathfrak{B}} \rightarrow \text{Block}^{\mathfrak{A}}$ and $\text{id}_2 : \text{Color}^{\mathfrak{B}} \rightarrow \text{Color}^{\mathfrak{A}}$ will provide the required homomorphism (as \mathfrak{B} is just a finite structure that is taking a finite information from \mathfrak{A} and behaves like \mathfrak{A} on that finite information). [As a side note, the homomorphism is actually an embedding, as \mathfrak{B} is actually a substructure of \mathfrak{A}] We have the homomorphism and a finite structure, but still \mathfrak{B} could not satisfy $W \cup \Delta$, so we need to verify that \mathfrak{B} actually satisfies $W \cup \Delta$.

The case for $\mathfrak{B} \models \Delta$ is obvious by construction, as $\mathfrak{A} \models \Delta$. The case for $\mathfrak{B} \models W$ will be exemplified with the first and third axioms (as the second and fourth are similar).

We want to show that $\mathfrak{B} \models [\forall x : \text{Block } \forall y : \text{Block } \text{on}(x, y) \rightarrow \text{above}(x, y)]$. So, suppose $x, y \in \text{Block}^{\mathfrak{B}}$ such that $(x, y) \in \text{on}^{\mathfrak{B}}$. Therefore, by construction $(x, y) \in \text{on}^{\mathfrak{A}}$ and as $\mathfrak{A} \models [\forall x : \text{Block } \forall y : \text{Block } \text{on}(x, y) \rightarrow \text{above}(x, y)]$ then $(x, y) \in \text{above}^{\mathfrak{A}}$, but $(x, y) \in \text{Block}^{\mathfrak{B}} \times \text{Block}^{\mathfrak{B}}$ and so, by construction $(x, y) \in \text{above}^{\mathfrak{B}}$ as required.

We want to show that $\mathfrak{B} \models [\forall x : \text{Block } x \neq c_{10} \rightarrow f(x) = \text{red}]$. So, suppose $x \in \text{Block}^{\mathfrak{B}}$ such that $x \neq c_{10}^{\mathfrak{B}}$. Therefore, by construction $x \neq c_{10}^{\mathfrak{A}}$ and as $\mathfrak{A} \models [\forall x : \text{Block } x \neq c_{10} \rightarrow f(x) = \text{red}]$ it follows that $f^{\mathfrak{A}}(x) = \text{red}^{\mathfrak{A}}$, but $x \in \text{Block}^{\mathfrak{B}}$ and therefore $f^{\mathfrak{A}}(x) = [f^{\mathfrak{A}} \upharpoonright \text{Block}^{\mathfrak{B}}](x) = f^{\mathfrak{B}}(x)$ and so $f^{\mathfrak{B}}(x) = \text{red}^{\mathfrak{A}} = \text{red}^{\mathfrak{B}}$ as required.

Therefore, by Remark 3.7, $W \cup \Delta$ has the $ATSen(\mathcal{L})$ -frp.

Observe that the theory of the example cannot be represented in any decidable fragment of Section 3.4, but it is still decidable at least when we want to query atomic sentences, as it has the $ATSen(\mathcal{L})$ -frp. Let us do a more complex example:

Example 3.5. We will use a modified version of the theory of total orders. Let W be:

TYPES

$$Number \quad NULL \quad Value \stackrel{\text{def}}{=} Number \cup NULL$$

CONSTANTS

$$1, \dots, 100 : Number \quad null : NULL$$

OBSERVATION PREDICATES

$$\leq : Value \times Value$$

ABSTRACTION PREDICATES

$$domain : Value$$

FUNCTIONS

$$max : Value \times Value \rightarrow Value$$

AXIOMS

$$1 \neq 2 \quad \dots \quad 1 \neq 100 \quad 1 \neq null$$

$$2 \neq 3 \quad \dots \quad 2 \neq 100 \quad 2 \neq null$$

$$\vdots \quad \quad \quad \vdots$$

$$99 \neq 100 \quad 99 \neq null \quad 100 \neq null$$

$$\forall x : Number [x \leq x]$$

$$\forall x : Number \forall y : Number [x \leq y \wedge y \leq x \rightarrow x = y]$$

$$\forall x : Number \forall y : Number \forall z : Number [x \leq y \wedge y \leq z \rightarrow x \leq z]$$

$$\forall x : Number \forall y : Number [x \leq y \vee y \leq x]$$

$$\forall x : Number [null \not\leq x \wedge x \not\leq null]$$

$$\forall x : Value [domain(x) \leftrightarrow [x = 1 \vee x = 2 \vee \dots \vee x = 100]]$$

$$\forall x : Value \forall y : Value [domain(x) \wedge domain(y) \rightarrow [x \leq y \rightarrow max(x, y) = y]]$$

$$\forall x : Value \forall y : Value [domain(x) \wedge domain(y) \rightarrow [x \not\leq y \rightarrow max(x, y) = x]]$$

$$\forall x : Value \forall y : Value [\neg(domain(x) \wedge domain(y)) \rightarrow max(x, y) = null]$$

Intuitively, the function max is computing the biggest of its arguments. We are using the abstraction predicate $domain$ to specify the valid domain of the function max so that, when the arguments to max are not in the valid domain, max outputs the constant $null$. Also, the function max is completely specified: any pair of inputs to max will be mapped to a constant, either to a constant of type $Number$ or to the constant $null$. We will define the set of observations \mathcal{O} to be the set of all atomic sentences without equality that can be formed from observation predicates. Therefore, an example of a state description will be $\Delta = \{5 \leq 2, 10 \leq max(5, 20), max(null, max(1, 2)) \leq 6\}$ which is obviously

inconsistent with respect to W .

Also, observe that W allows infinite models. We will show that given some finite state description Δ , $W \cup \Delta$ has the $ATSen(\mathcal{L})$ -frp where \mathcal{L} is the signature given above.

I will follow the suggested strategy described in Remark 3.7. Let Δ be a finite state description and let \mathfrak{A} be a model of $\Delta \cup W$.

Build the model \mathfrak{B} as follows: the domain of type `Number` will be the interpretations of the 100 constant symbols in the structure \mathfrak{A} . The domain of type `NULL` will be the interpretation of `null` in the structure \mathfrak{A} and the domain of type `Value` will be the union of `Number` and `NULL`. Then, define $\leq^{\mathfrak{B}} = \leq^{\mathfrak{A}} \cap (Value^{\mathfrak{B}} \times Value^{\mathfrak{B}})$ and similarly $domain^{\mathfrak{B}} = domain^{\mathfrak{A}} \cap Value^{\mathfrak{B}}$. Now, define $i^{\mathfrak{B}} = i^{\mathfrak{A}}$ for $1 \leq i \leq 100$ and $null^{\mathfrak{B}} = null^{\mathfrak{A}}$. Finally, define $max^{\mathfrak{B}} = max^{\mathfrak{A}} \upharpoonright (Value^{\mathfrak{B}} \times Value^{\mathfrak{B}})$.

Again, we need to show that $ran\ max^{\mathfrak{A}} \subseteq Value^{\mathfrak{B}}$ for $max^{\mathfrak{A}} \upharpoonright (Value^{\mathfrak{B}} \times Value^{\mathfrak{B}})$ to be well-defined. Let $x \in ran\ max^{\mathfrak{A}}$, then there is $y, z \in Value^{\mathfrak{A}}$ such that $max^{\mathfrak{A}}(y, z) = x$. Now, both $y \in domain^{\mathfrak{A}}$ and $z \in domain^{\mathfrak{A}}$ or not. If both are, now either $y \leq^{\mathfrak{A}} z$ or not. If $y \leq^{\mathfrak{A}} z$ and as $\mathfrak{A} \models [\forall x : Value\ \forall y : Value\ [domain(x) \wedge domain(y) \rightarrow [x \leq y \rightarrow max(x, y) = y]]]$ we have $max^{\mathfrak{A}}(y, z) = z = x$. But $z \in domain^{\mathfrak{A}}$ and as $\mathfrak{A} \models [\forall x : Value\ [domain(x) \leftrightarrow [x = 1 \vee x = 2 \vee \dots \vee x = 100]]]$ we have that $z = i^{\mathfrak{A}}$ for some $1 \leq i \leq 100$. Therefore $x \in Value^{\mathfrak{B}}$ as $i^{\mathfrak{A}} \in Value^{\mathfrak{B}}$ for every $1 \leq i \leq 100$. The case for $y \not\leq^{\mathfrak{A}} z$ is similar. Now, for the case when either $y \notin domain^{\mathfrak{A}}$ or $z \notin domain^{\mathfrak{A}}$, as $\mathfrak{A} \models [\forall x : Value\ \forall y : Value\ [\neg(domain(x) \wedge domain(y)) \rightarrow max(x, y) = null]]]$ it follows that $max^{\mathfrak{A}}(y, z) = null^{\mathfrak{A}} = x$ and therefore $x \in Value^{\mathfrak{B}}$ as well.

Continuing with the example, the inclusion functions $id_1 : Number^{\mathfrak{B}} \rightarrow Number^{\mathfrak{A}}$, $id_2 : NULL^{\mathfrak{B}} \rightarrow NULL^{\mathfrak{A}}$ and $id_3 : Value^{\mathfrak{B}} \rightarrow Value^{\mathfrak{A}}$ will provide the required homomorphism [As a side note, actually it is an embedding, as it can be shown that \mathfrak{B} is a substructure of \mathfrak{A}]. We have the homomorphism and a finite structure, but still \mathfrak{B} could not satisfy $W \cup \Delta$, so we need to verify that \mathfrak{B} actually satisfies $W \cup \Delta$.

The case for $\mathfrak{B} \models \Delta$ is obvious by construction, as $\mathfrak{A} \models \Delta$. The case for $\mathfrak{B} \models W$ will be exemplified with the fourth, fifth and eighth axioms (as all the others have similar proofs).

We want to show that $\mathfrak{B} \models [\forall x : Number\ \forall y : Number\ [x \leq y \vee y \leq x]]$. So, suppose $x, y \in Number^{\mathfrak{B}}$. Then, $x, y \in Number^{\mathfrak{A}}$ and as $\mathfrak{A} \models [\forall x : Number\ \forall y :$

Number $[x \leq y \vee y \leq x]$, we have $x \leq^{\mathfrak{A}} y \vee y \leq^{\mathfrak{A}} x$. It follows by construction that $x \leq^{\mathfrak{B}} y \vee y \leq^{\mathfrak{B}} x$ as $(x, y), (y, x) \in \text{Number}^{\mathfrak{B}} \times \text{Number}^{\mathfrak{B}}$.

We want to show that $\mathfrak{B} \models [\forall x: \text{Number} [\text{null} \not\leq x \wedge x \not\leq \text{null}]]$. So, suppose $x \in \text{Number}^{\mathfrak{B}}$. Then, $x \in \text{Number}^{\mathfrak{A}}$ and as $\mathfrak{A} \models [\forall x: \text{Number} [\text{null} \not\leq x \wedge x \not\leq \text{null}]]$, we have $\text{null}^{\mathfrak{A}} \not\leq^{\mathfrak{A}} x$ and $x \not\leq^{\mathfrak{A}} \text{null}^{\mathfrak{A}}$. It follows that $(\text{null}^{\mathfrak{A}}, x) \notin [\leq^{\mathfrak{A}} \cap (\text{Value}^{\mathfrak{B}} \times \text{Value}^{\mathfrak{B}})]$ and $(x, \text{null}^{\mathfrak{A}}) \notin [\leq^{\mathfrak{A}} \cap (\text{Value}^{\mathfrak{B}} \times \text{Value}^{\mathfrak{B}})]$ so $(\text{null}^{\mathfrak{A}}, x) \notin \leq^{\mathfrak{B}}$ and $(x, \text{null}^{\mathfrak{A}}) \notin \leq^{\mathfrak{B}}$ by construction or equivalently $\text{null}^{\mathfrak{B}} \not\leq^{\mathfrak{B}} x$ and $x \not\leq^{\mathfrak{B}} \text{null}^{\mathfrak{B}}$.

We want to show that $\mathfrak{B} \models [\forall x: \text{Value} \forall y: \text{Value} [\text{domain}(x) \wedge \text{domain}(y) \rightarrow [x \not\leq y \rightarrow \text{max}(x, y) = x]]]$. So, suppose $x, y \in \text{Value}^{\mathfrak{B}}$ such that $x \in \text{domain}^{\mathfrak{B}}$, $y \in \text{domain}^{\mathfrak{B}}$ and $x \not\leq^{\mathfrak{B}} y$. By construction, $x \in \text{domain}^{\mathfrak{A}}$, $y \in \text{domain}^{\mathfrak{A}}$ and $x \not\leq^{\mathfrak{A}} y$ (as $(x, y) \in (\text{Value}^{\mathfrak{B}} \times \text{Value}^{\mathfrak{B}})$). As $\mathfrak{A} \models [\forall x: \text{Value} \forall y: \text{Value} [\text{domain}(x) \wedge \text{domain}(y) \rightarrow [x \not\leq y \rightarrow \text{max}(x, y) = x]]]$ it follows that $\text{max}^{\mathfrak{A}}(x, y) = x$. But $(x, y) \in \text{Value}^{\mathfrak{B}} \times \text{Value}^{\mathfrak{B}}$ and therefore $\text{max}^{\mathfrak{A}}(x, y) = [\text{max}^{\mathfrak{A}} \upharpoonright (\text{Value}^{\mathfrak{B}} \times \text{Value}^{\mathfrak{B}})](x, y) = \text{max}^{\mathfrak{B}}(x, y)$ and so $\text{max}^{\mathfrak{B}}(x, y) = x$ as required.

Therefore, by Remark 3.7, $W \cup \Delta$ has the $ATSen(\mathcal{L})$ -frp.

So, it seems that our database-like conditions are strong enough to guarantee the $ATSen(\mathcal{L})$ -frp, but wait, What if we add the following axiom, to make the total order *dense* in the last example?:

$$\begin{aligned} \forall x: \text{Number} \forall y: \text{Number} [x \leq y \wedge x \neq y \rightarrow \\ \exists z: \text{Number} z \neq x \wedge z \neq y \wedge x \leq z \wedge z \leq y] \end{aligned}$$

Then, following the same procedure, we will get a finite structure \mathfrak{B} such that there is an homomorphism to \mathfrak{A} . The structure \mathfrak{B} will satisfy the theory, except the new axiom! The new axiom produces the nasty effect that the theory W will only allow infinite models. So, our database-like conditions are not complete after all, in the sense that they do not imply the $ATSen(\mathcal{L})$ -frp: the conditions ensure the existence of a finite model in the signature \mathcal{L} and of an homomorphism from the finite model to a general model of $\Delta \cup W$, but the conditions do not ensure that the constructed finite model will satisfy $\Delta \cup W$, therefore there is missing criteria that has to do with the form of the theory.

What is wrong with the new axiom? Apart from producing the effect of not allowing finite models, the key property is: the new axiom is introducing an anonymous individual in the existential quantifier (remember the type *Number* is not an enumeration type). If we want to keep going with our database analogy,

introducing an anonymous individual is forbidden in a database: anonymous individuals play the same role as anonymous values in a function, as they cannot be evaluated before being inserted as an entry in a table. So, we require that every time an existentially quantified individual is introduced in an axiom, we require that the individual be represented by some constant symbol, i.e. the individual is NOT anonymous.

All these considerations lead to a revision of the criteria for database-like theories, this time we will only state the required conditions:

Definition 3.4. *Let \mathcal{L} be a typed, first-order signature. We will say that \mathcal{L} is finite if has a finite number of constants, a finite number of predicates, a finite number of functions and a finite number of types. Let $W \subseteq \text{Sen}(\mathcal{L})$. We will say that W is database-like if the following conditions are met:*

1. *For every function symbol $f \in \mathcal{L}$ of type $\tau_1 \times \dots \times \tau_n \rightarrow \tau$: for every \mathcal{L} -model \mathfrak{A} of W , we have that for every $x \in \text{ran } f^{\mathfrak{A}}$, there is a constant symbol $c \in \mathcal{L}$ of type τ such that $x = c^{\mathfrak{A}}$.*
2. *For every sub-formula in W of the form $\exists v : \tau \lambda$ where $\lambda \in \text{Form}(\mathcal{L})$ and τ is a type in \mathcal{L} : for every \mathcal{L} -model \mathfrak{A} of W and every interpretation functions $h^{\tau_i} : \text{Var}(\mathcal{L}, \tau_i) \rightarrow \tau_i^{\mathfrak{A}}$, where τ_i is the i -th type, we have that IF $\mathfrak{A} \models \exists v : \tau \lambda[h^{\tau_1}, \dots, h^{\tau}, \dots, h^{\tau_n}]$ THEN there is $x \in \tau^{\mathfrak{A}}$ such that $\mathfrak{A} \models \lambda[h^{\tau_1}, \dots, \langle h'^{\tau} \mid h'^{\tau}(v) \mapsto x \rangle, \dots, h^{\tau_n}]$ and there is a constant $c \in \mathcal{L}$ of type τ such that $x = c^{\mathfrak{A}}$.*

Where I am denoting by $\text{Form}(\mathcal{L})$ the set of formulae in signature \mathcal{L} . $\text{Var}(\mathcal{L}, \tau_i)$ is the set of variables with type τ_i in signature \mathcal{L} . $\tau^{\mathfrak{A}}$ denotes the domain of type τ in structure \mathfrak{A} . And the notation $\langle h'^{\tau} \mid h'^{\tau}(v) \mapsto z \rangle$ means that $\langle h'^{\tau} \mid h'^{\tau}(v) \mapsto z \rangle$ is the function identical to h^{τ} except for input v : v will be mapped to z .

Observe that theories that only have enumeration types, are trivially database-like. Also, I will concentrate on the case of the $QFSen(\mathcal{L})$ -frp instead of the $ATSen(\mathcal{L})$ -frp as all previous examples suggest that we can actually strengthen the form of the observations to quantifier-free sentences. Another reason is that the $ATSen(\mathcal{L})$ -frp follows trivially from the $QFSen(\mathcal{L})$ -frp. Now, the main result of this section:

Theorem 3.13. *Let \mathcal{L} be a typed, finite, first-order signature. Let $\mathcal{O} \subseteq QFSen(\mathcal{L})$ and $W \subseteq Sen(\mathcal{L})$ a finite theory. Let finite state description $\Delta \subseteq \mathcal{O}$. If W is database-like then $W \cup \Delta$ has the $QFSen(\mathcal{L})$ -frp.*

Proof. Suppose W is database-like. We will prove the direction \Leftarrow of the $QFSen(\mathcal{L})$ -frp, as the direction \Rightarrow is trivial. So, assume that for every \mathcal{L} -structure α in $FMod_W(\Delta)$ we have that $\alpha \models \eta$ where $\eta \in QFSen(\mathcal{L})$. We want to show $\Delta \models_W \eta$. We will follow a similar strategy as in the previous examples. Let \mathfrak{A} be a model of $W \cup \Delta$.

Now, build structure \mathfrak{B} as follows: For each type $\tau \in \mathcal{L}$, define its domain $\tau^{\mathfrak{B}}$ in \mathfrak{B} to be the interpretations in \mathfrak{A} of all constants in \mathcal{L} with type τ . If after doing this, $\tau^{\mathfrak{B}}$ is still empty (in the case there are no constants in \mathcal{L} of type τ), then as \mathfrak{A} is a structure and so $\tau^{\mathfrak{A}} \neq \emptyset$, and so there is $z \in \tau^{\mathfrak{A}}$, define $\tau^{\mathfrak{B}} = \{z\}$.

For each predicate symbol $P \in \mathcal{L}$ of type signature $\tau_1 \times \dots \times \tau_n$, define $P^{\mathfrak{B}} = P^{\mathfrak{A}} \cap (\tau_1^{\mathfrak{B}} \times \dots \times \tau_n^{\mathfrak{B}})$.

Similarly, for each constant symbol $c \in \mathcal{L}$, define $c^{\mathfrak{B}} = c^{\mathfrak{A}}$.

Finally, each function symbol $f \in \mathcal{L}$ of type signature $\tau_1 \times \dots \times \tau_n \rightarrow \tau$ define $f^{\mathfrak{B}} = f^{\mathfrak{A}} \upharpoonright (\tau_1^{\mathfrak{B}} \times \dots \times \tau_n^{\mathfrak{B}})$. Again, for $f^{\mathfrak{B}} = f^{\mathfrak{A}} \upharpoonright (\tau_1^{\mathfrak{B}} \times \dots \times \tau_n^{\mathfrak{B}})$ to be well-defined, we need to show that $ran f^{\mathfrak{A}} \subseteq \tau^{\mathfrak{B}}$. This follows directly from the first database-like condition: Let $x \in ran f^{\mathfrak{A}}$, by the first database-like condition, there is a constant $c \in \mathcal{L}$ of type τ such that $x = c^{\mathfrak{A}}$. But $c^{\mathfrak{A}} \in \tau^{\mathfrak{B}}$ by construction, so $x \in \tau^{\mathfrak{B}}$ as required.

Observe that \mathfrak{B} is a finite structure as \mathcal{L} has a finite number of constants. Also, observe that structure \mathfrak{B} is a substructure of \mathfrak{A} by construction.

Property 1: As \mathfrak{B} is a substructure of \mathfrak{A} this means that the following property holds: For all $\phi \in QFForm(\mathcal{L})$ and every interpretation functions $h^{\tau_i} : Var(\mathcal{L}, \tau_i) \rightarrow \tau_i^{\mathfrak{B}}$, where τ_i is the i -th type in \mathcal{L} , we have: $\mathfrak{B} \models \phi[h^{\tau_1}, \dots, h^{\tau_n}] \iff \mathfrak{A} \models \phi[h^{\tau_1}, \dots, h^{\tau_n}]$. Where we are denoting by $QFForm(\mathcal{L})$ the set of quantifier-free formulae in signature \mathcal{L} , by $Var(\mathcal{L}, \tau_i)$ the set of variables with type τ_i in signature \mathcal{L} , and by $\tau_i^{\mathfrak{B}}$ the domain of type τ_i in structure \mathfrak{B} .

By property 1, it follows that $\mathfrak{B} \models \Delta$ as $\mathfrak{A} \models \Delta$ and $\Delta \subseteq \mathcal{O} \subseteq QFSen(\mathcal{L})$. In order to show that $\mathfrak{B} \models W$, let $\psi \in W$. Transform ψ to its Prenex Normal Form, then, it will have the form $Q^* \lambda$ where λ is a quantifier-free formula and Q is either \exists or \forall . Now, we prove the following claim:

Claim 1: For all interpretation functions $h^{\tau_i} : Var(\mathcal{L}, \tau_i) \rightarrow \tau_i^{\mathfrak{B}}$, where τ_i is

the i -th type in \mathcal{L} : IF $\mathfrak{A} \models Q^* \lambda[h^{\tau_1}, \dots, h^{\tau_n}]$ THEN $\mathfrak{B} \models Q^* \lambda[h^{\tau_1}, \dots, h^{\tau_n}]$.

Proof: By induction on the length of the string Q^* . The base case is when there are no quantifiers. It holds trivially by Property 1, as λ will be a quantifier-free sentence. Now, the inductive step:

Case $Q^* \lambda = \forall v : \tau \omega$. We assume the property for ω :

“For all interpretation functions $h^{\tau_i} : \text{Var}(\mathcal{L}, \tau_i) \rightarrow \tau_i^{\mathfrak{B}}$, where τ_i is the i -th type in \mathcal{L} : IF $\mathfrak{A} \models \omega[h^{\tau_1}, \dots, h^{\tau_n}]$ THEN $\mathfrak{B} \models \omega[h^{\tau_1}, \dots, h^{\tau_n}]$ ”.

Now, suppose $h^{\tau_i} : \text{Var}(\mathcal{L}, \tau_i) \rightarrow \tau_i^{\mathfrak{B}}$ are interpretation functions. Suppose $\mathfrak{A} \models \forall v : \tau \omega[h^{\tau_1}, \dots, h^{\tau_n}]$. We want to show that $\mathfrak{B} \models \forall v : \tau \omega[h^{\tau_1}, \dots, h^{\tau_n}]$. So, let arbitrary $z \in \tau^{\mathfrak{B}}$ we want to show $\mathfrak{B} \models \omega[h^{\tau_1}, \dots, \langle h'^{\tau} \mid h'^{\tau}(v) \mapsto z \rangle, \dots, h^{\tau_n}]$. As $z \in \tau^{\mathfrak{B}}$ then $z \in \tau^{\mathfrak{A}}$ (as \mathfrak{B} is a substructure of \mathfrak{A}). Since $\mathfrak{A} \models \forall v : \tau \omega[h^{\tau_1}, \dots, h^{\tau_n}]$, then $\mathfrak{A} \models \omega[h^{\tau_1}, \dots, \langle h'^{\tau} \mid h'^{\tau}(v) \mapsto z \rangle, \dots, h^{\tau_n}]$. Then, by using the interpretation functions $h^{\tau_1}, \dots, \langle h'^{\tau} \mid h'^{\tau}(v) \mapsto z \rangle, \dots, h^{\tau_n}$ in the inductive hypothesis, we will get $\mathfrak{B} \models \omega[h^{\tau_1}, \dots, \langle h'^{\tau} \mid h'^{\tau}(v) \mapsto z \rangle, \dots, h^{\tau_n}]$ as required.

[Remember: The notation $\langle h'^{\tau} \mid h'^{\tau}(v) \mapsto z \rangle$ means that $\langle h'^{\tau} \mid h'^{\tau}(v) \mapsto z \rangle$ is the function identical to h^{τ} except for input v : v will be mapped to z].

Case $Q^* \lambda = \exists v : \tau \omega$. We assume the property for ω :

“For all interpretation functions $h^{\tau_i} : \text{Var}(\mathcal{L}, \tau_i) \rightarrow \tau_i^{\mathfrak{B}}$, where τ_i is the i -th type in \mathcal{L} : IF $\mathfrak{A} \models \omega[h^{\tau_1}, \dots, h^{\tau_n}]$ THEN $\mathfrak{B} \models \omega[h^{\tau_1}, \dots, h^{\tau_n}]$ ”.

Now, suppose $h^{\tau_i} : \text{Var}(\mathcal{L}, \tau_i) \rightarrow \tau_i^{\mathfrak{B}}$ are interpretation functions. Suppose $\mathfrak{A} \models \exists v : \tau \omega[h^{\tau_1}, \dots, h^{\tau_n}]$. We want to show that $\mathfrak{B} \models \exists v : \tau \omega[h^{\tau_1}, \dots, h^{\tau_n}]$. Since $\mathfrak{A} \models \exists v : \tau \omega[h^{\tau_1}, \dots, h^{\tau_n}]$, then, by the second database-like condition, there is $z \in \tau^{\mathfrak{A}}$ such that $\mathfrak{A} \models \omega[h^{\tau_1}, \dots, \langle h'^{\tau} \mid h'^{\tau}(v) \mapsto z \rangle, \dots, h^{\tau_n}]$ and there is a constant $c \in \mathcal{L}$ of type τ such that $z = c^{\mathfrak{A}}$. At this point, we need to be careful, as the function $\langle h'^{\tau} \mid h'^{\tau}(v) \mapsto z \rangle$ has in its image the element z which may not be in $\tau^{\mathfrak{B}}$ and therefore we could not use the induction hypothesis. But, by construction $c^{\mathfrak{A}} \in \tau^{\mathfrak{B}}$ and therefore $z \in \tau^{\mathfrak{B}}$. So, we can use the inductive hypothesis with the interpretation functions $h^{\tau_1}, \dots, \langle h'^{\tau} \mid h'^{\tau}(v) \mapsto z \rangle, \dots, h^{\tau_n}$ to get $\mathfrak{B} \models \omega[h^{\tau_1}, \dots, \langle h'^{\tau} \mid h'^{\tau}(v) \mapsto z \rangle, \dots, h^{\tau_n}]$, and therefore there is $z \in \tau^{\mathfrak{B}}$ such that $\mathfrak{B} \models \omega[h^{\tau_1}, \dots, \langle h'^{\tau} \mid h'^{\tau}(v) \mapsto z \rangle, \dots, h^{\tau_n}]$ and so $\mathfrak{B} \models \exists v : \tau \omega[h^{\tau_1}, \dots, h^{\tau_n}]$ by definition of satisfaction of existential.

This proves Claim 1.

Since $\mathfrak{A} \models W$ and $\psi \in W$ and its Prenex Normal Form $Q^* \lambda$ is equivalent to ψ , then $\mathfrak{A} \models Q^* \lambda$. By Claim 1, it follows that $\mathfrak{B} \models Q^* \lambda$ or equivalently $\mathfrak{B} \models \psi$ (Observe that it does not matter which interpretation functions for the variables we choose, as $Q^* \lambda$ is a sentence). From here, it follows that $\mathfrak{B} \models W$.

Therefore, we have built a finite \mathcal{L} -structure \mathfrak{B} that satisfies $W \cup \Delta$. By assumption of the \Leftarrow direction of the $QFSen(\mathcal{L})$ -frp, we have that $\mathfrak{B} \models \eta$. But, η is a quantifier-free sentence and \mathfrak{B} is a substructure of \mathfrak{A} , therefore, by property 1: $\mathfrak{A} \models \eta$ as required. \square

Ways to check in a recursive manner the first condition of the database-like criteria is to make sure that every function in the theory either maps to an enumeration type OR if this is not the case, to check if the function is being built using a primitive recursive construction form. Primitive recursive functions have the nice property that they are totally defined and they can be recognized just by the form of their formal expression [as long as we have a means to define how a formal expression of a primitive recursive function will look like in a first-order theory, but this can be done in principle]. Observe that in Example 3.5, I am defining function *max* by cases, which is a valid primitive recursive construction. Note that I am not claiming that *max* is a primitive recursive function, but only that it is built using a primitive recursive construction: definition by cases. This is a problem worth exploring as future work.

Ways to check in a recursive manner the second condition of the database-like criteria is to make sure that every existential either introduces a variable v in an enumeration type OR if this is not the case, to check that after the variable v there is a clause of the form $\bigvee v = t_i$ where the t_i 's are *closed* terms. Since the t_i 's are closed, by the first database-like condition, in any model of the sentence, there will be a constant representing the closed term and so v will be represented by some constant in the disjunction.

After we have defined recursive tests of the database-like conditions (the ones I gave on the previous two paragraphs are examples of how this could be done), call this recursive function *testDatabaseCond*, then we can define a recursive admissible relation as follows:

```

if  $\Delta \subseteq QFSen(\mathcal{L})$  and  $\psi \in QFSen(\mathcal{L})$  and  $testDatabaseCond(W)$  then
  Test in parallel:  $\Delta \vdash_W \psi$  and  $\Delta \perp_W \psi$ 
  If first condition ends first: Output NO
  If second condition ends first: Output YES
else
  Output NO
end if

```

Again, the notation $\Delta \perp_W \psi$ means that there is a finite model that satisfies $W \cup \Delta$ but does not satisfy ψ .

The recursive property follows: if $testDatabaseCond(W)$ outputs YES [and all other conditions are satisfied], by Theorem 3.13, the theory will have the $QFSen(\mathcal{L})$ -frp and so, the procedure will finish.

Tasks to explore as future work involve how can we express a database in a first-order language (as well as the restrictions normally used in a database), as this will give us means to build recursive tests for the database-like criteria (maybe more general tests than the ones I am presenting in here); how can we represent primitive recursive functions in a first-order language and how can we recognize just by checking the syntax of the theory that a function is being built from this primitive recursive constructions; and finally, but not less important, the problem regarding the expressive power of a database-like theory.

Chapter 4

Examples in Isabelle

This chapter presents a general overview of the generic theorem prover and proof assistant Isabelle. It also presents two implementations of toy versions of the Blocks World Theory, in order to exemplify the main ideas of the proof systems of Sections 3.2 and 3.4 and the technical details required for implementing those proof systems in Isabelle.

4.1 An overview of Isabelle

Isabelle is a generic theorem prover and a generic system for building deductive systems, with a special focus on interactive theorem proving in higher-order logics [21]. Proofs in Isabelle are performed by a generalization of resolution, using higher-order unification.

Object-logics are represented in Isabelle by means of a *theory* file. A theory is a named collection of types, functions, and theorems, much like a module in a programming language [12]. The general format of a theory T is

```
theory T
imports B1 ... Bn
begin
declarations, definitions, and proofs
end
```

where B1 ... Bn are the names of existing theories that T is based on and `declarations, definitions, and proofs` represents the newly introduced concepts (types, functions etc.) and proofs about them.

For example, the theory `HOL` (representing classical higher-order logic) is built on top the theory `Pure`. `Pure` defines the meta-logic of Isabelle, which is an intuitionistic higher-order logic with implication (`==>`) and universal quantifier (`!!`). `HOL` builds on top, by defining the classical logical operators: universal quantifier (`ALL`), existential quantifier (`EX`), implication (`-->`), equality (`=`), and (`&`), or (`|`), negation (`~`), bi-conditional (`<->`). In this way, `HOL` can express the rules of inference of classical logic by using the meta-logic operators in `Pure`.

For example, the Modus Ponens inference rule is expressed as $(P \rightarrow Q) \Rightarrow P \Rightarrow Q$ which can be abbreviated as $[| P \rightarrow Q; P |] \Rightarrow Q$. The symbols `P` and `Q` are variables, and `P --> Q` is a well-formed expression in `HOL` which is using the implication operator (`-->`) in `HOL`. Observe that the operator `==>` is the implication operator in `Pure`. In our example, the Modus Ponens rule interpreted in `Pure` is saying: Whenever the expression `P --> Q` holds and the expression `P` holds, the expression `Q` holds. This shows an example of how `Pure` acts like a meta-logic.

A more interesting example is the case of the universal introduction inference rule. In classical logic, we can introduce the universal quantifier using variable x in some proof step as long as x does not appear free in our set of assumptions. In `HOL`, the universal introduction inference rule is represented as $(!!x. P x) \Rightarrow (ALL x. P x)$. Observe that `!!` is the universal quantifier in `Pure` and `ALL` is the universal quantifier in `HOL`, so that when the expression is interpreted in `Pure`, it reads: If for all x , `P x` holds, then the expression `ALL x. P x` holds, and so there is no need to stipulate that x must be not free, as it is already bounded by the meta-universal quantifier in `Pure`!

This property of `Pure` to act as a meta-logic is what will allow us to represent the Minimum Axiom Schema in Sections 4.2 and 4.3. And also, we will reuse the classical operators in `HOL`, the inference rules in `HOL` and the operators and inference rules in theory `Set` which is a representation of typed set theory built on top theory `HOL`.

Isabelle was programmed using Standard ML, therefore Isabelle provides constructions for programming functions in ML and connect them with Isabelle. This is the property we will exploit in Section 4.3 for representing a decision procedure in Isabelle.

The central notion in Isabelle is that of a theorem. All axioms in the definitions inside a theory file are available as theorems. The general form of a theorem is `A`

$\implies B \implies \dots \implies C \implies D$ (i.e. an expression using the implication operator in theory *Pure*) which can be abbreviated as $[| A; B; \dots; C|] \implies D$.

Theorems are combined by resolution. For example, given theorems $A_1 \implies \dots \implies A_m \implies A$ and $B_1 \implies \dots \implies B_k \implies \dots \implies B_n \implies B$ and a substitution s such that $s(A) = s(B_k)$, then the following theorem can be obtained $s(B_1 \implies \dots \implies A_1 \implies \dots \implies A_m \implies \dots \implies B_n \implies B)$. Isabelle computes the substitution s by higher-order unification [11]. Theorem $B_1 \implies \dots \implies B_k \implies \dots \implies B_n \implies B$ can be read as an intermediate proof state in a backward proof where B is the overall goal and the B_i are the subgoals yet to be proved. In this case, resolution with theorem $A_1 \implies \dots \implies A_m \implies A$ (i.e. if we interpret the theorem as an inference rule) corresponds to an “instantiation” of the inference rule where the hypotheses of the rule will be added as sub-goals to be proved.

In Isabelle, proofs can be performed by applying one rule at a time (by making use of the resolution step described in the last paragraph), or in large tactic steps. Tactics combine arbitrary algorithmic sequences of resolution steps and other transformations into a single function. Since Isabelle represents proof states as theorems (see paragraph above on the interpretation of theorem $B_1 \implies \dots \implies B_k \implies \dots \implies B_n \implies B$ as an intermediate step in a backward proof), tactics are functions from theorems to sequences of theorems, to allow for backtracking (in Section 4.3 we will see an example of how Isabelle represents proof states as theorem expressions). Backtracking can occur over the choice of rule and also over the choice of unifier (because of higher-order unification). Tactics can be written from scratch or can be assembled from existing tactics with tacticals like `THEN`, `ORELSE`, `REPEAT`, `DEPTH FIRST`, `BEST FIRST`, etc. [11]. Tacticals are simply functions from tactics to tactics.

All these constructions will be explained in more detail in the following sections.

4.2 Implementing Υ^M using an example theory

In this section we will take the Blocks World theory of Example 3.1 of Section 3.2 to exemplify how the proof system Υ^M of Section 3.2 could be implemented in Isabelle and how we could carry out some proofs in Υ^M . We will extend the example to four blocks instead of two.

Also, we will not represent the sequents $\mathcal{O}, \Delta, \Gamma \mid_W^M \psi$ of Υ^M as such in Isabelle. Why not? Doing such thing would imply that we will not be able to use all the built-in rules and automatic tactics in the theory `HOL` of Isabelle unless we do some extra work: In order to represent the sequents, we need to introduce a relation symbol \mid that relates the sets \mathcal{O} , Δ , Γ , W and the formula ψ , and then define all rules of inference using that relation symbol, and if we want to use the automatic tools, we will need to setup the tools to recognize expressions including the relation symbol \mid . This approach is followed in some implementations of logics, for example [4]. Instead of following that route, we will use the available inference rules and automatic tools in the theory `HOL`. Remember that the proof system Υ^M has all the classical inference rules but we are adding only the Minimum Axiom Schema (see axiom schema $(*)$ of Section 3.2), so that it makes sense to reuse all available classical inference rules in Isabelle. So, we will follow some conventions regarding the sequent $\mathcal{O}, \Delta, \Gamma \mid_W^M \psi$:

- The sets \mathcal{O} and Δ will be represented by external sets. These external sets will be referenced only in the Minimum Axiom Schema.
- Isabelle uses a “working set” of formulae while doing a proof. This working set will represent our Γ in the sequent. The working set is used to discharge goals that can be proved by assumption.
- W will be represented externally by providing a list of axioms, so that whenever we want to use a sentence of W in a proof, we will need to add the sentence to the working set.
- Even though, Δ is represented externally using a set, we will add at the start of every proof the elements of Δ to the working set.

Having established the conventions we will follow, we start the definitions in Isabelle:

```
theory BWUpsilonM1
imports Main
begin

datatype Block = A | B | C | D
datatype Table = T
```

```
types
```

```
Object = "Block + Table"
```

```
datatype Formula = on Block Object | green Block | red Block |
                  above Block Object |
                  And Formula Formula
```

```
consts
```

```
Obs      :: "Formula set"
```

```
Delta    :: "Formula set"
```

The theory `Main` contains the theory `HOL` and many other useful theories and constructions, like sets, datatypes, definitions of functions, etc. So, we are saying to Isabelle to import all those standard theories. After that, we define the main types, `Block` and `Table` to be datatypes. Datatypes are useful to represent data structures or syntactic constructions: each entry consists of a constructor name followed by the types of the parameters, if there are no parameters after the constructor name, the constructor will represent a *constant*. Internally, Isabelle creates axioms stating that every object that belongs to the datatype will be equal to one of the declared constructions, and also axioms stating that the constructor names are distinct. This will allow Isabelle to automatically handle equality comparison of expressions in the datatype. In this sense, datatypes are useful to represent enumeration types as we do in the example.

Next comes the `types` declaration, useful to declare “compound” or non-atomic types. The type `Object` is the disjoint union of the types `Block` and `Table`. Isabelle will handle the disjoint union by introducing two unary functions: `Inr` will tag an object of type `Table` into the type `Object` and `Inl` will tag an object of type `Block` into the type `Object`. We will see an example of their usage later on.

Next comes the declaration of the syntax of our formulae. This will allow us to form sets of formulae as we will require to do later on, when we build the set of observations. Also, this will allow equality testing of formulae when we want to test if some formula is in some set. For simplicity reasons, I am including only

atomic formulae and one example of a more complex formulae, i.e. the **And** case. Observe that datatypes allow inductive definitions as we are doing in the declaration of the **And** case. For example, the declaration `on Block Object` means that well-formed expressions involving `on`, will have as parameters expressions of type `Block` followed by an expression of type `Object`: take for example the sentence $on(A, T)$ which will be represented by the expression `on A (Inr T)` which is a well-formed expression according to the datatype declaration (remember that `Inr` and `Inl` are the functions that handle disjoint unions). As another example, the formula $[on(A, B) \wedge above(B, T)] \wedge red(A)$ will be represented as `And (And (on A (Inl B)) (above B (Inr T))) (red A)`. Later, we will see that we will not require to use the **And** case, as all of our observations are atomic sentences without equality.

Finally, comes the `consts` declaration section for declaring constants. We will represent the state description Δ (`Delta`) and our set of observations \mathcal{O} (`Obs`) as constants so that when we declare the Minimum Axiom Schema, we can make a reference to them (see later). Both of them have types of “sets of formulae”.

Now we add the following declarations to Isabelle:

```
fun coercion :: "Formula => bool"    ("@ _ ") where
"@ (And x y)  = ((@ x) & (@ y))"

defs
Obs_def :
"Obs == {t. EX w::Block. EX x::Object. EX y::Block. EX z::Block.
          (t = (on w x) | t = (red y) | t = (green z))}"
Delta_def :
"Delta == {r. (r = (on A (Inl B)) | r = (on B (Inr T)) |
              r = (on C (Inl D)) | r = (red A) |
              r = (green C))}"
```

In order to use the inference rules and the logical connectives in the theory `HOL`, we need to coerce our syntactic constructions declared in the datatype `Formula` into the `bool` type of the theory `HOL`. The function `coercion` does exactly this: the function will “tag” `Formula` expressions. The tagging can be intuitively read as “If `form` is a `Formula` (a syntactic expression without meaning) then `coercion form` is a formula in `HOL` (the expression has a truth value when

interpreted as a formula in HOL)". Compound syntactic expressions, like `And`, will be decomposed using the corresponding boolean operator in HOL. In our case we only need to declare the case for `And` (the corresponding logical operator in HOL is `&`), as the `Formula` datatype only declared the `And` case. The decomposition is done inductively until we arrive to a non-and case, i.e. predicates, in that case the function just tags the predicate without further transformation. The construction ("`@ _`") is an abbreviation, so that, for example, instead of writing `coercion (on A (Inl B))` we can write `@ (on A (Inl B))`.

Next comes the definitions of the symbols declared in the `consts` section. `Obs` is the set of all atomic sentences without equality that can be formed from observation predicates (see Example 3.1 in Section 3.2 for a description on the signature of the theory). This is the notation used to build sets by comprehension in the theory `Set` of Isabelle, fairly similar to the standard notation in mathematics. After the variable `t`, we specify a first-order formula that every element in the set must satisfy. This formula is built using the logical operators in HOL: The notation `EX x::t` denotes the existential quantifier with variable `x` of type `t` and the operator `|` is the logical or. In a similar way, the state description `Delta` is built. Observe that in here, the equality symbol means identity over syntactical expressions, remember that a `Formula` acquires meaning in the logic only if it is tagged with the coercion function `@`.

Now we add the axioms of the theory together with the declaration of the Minimum Axiom Schema:

```
axioms

ax1: "ALL x::Block. ALL y::Object.
      (@ (on x y)) --> (@ (above x y))"
ax2: "ALL x::Block. ALL y::Block. ALL z::Object.
      ((@ (above x (Inl y))) & (@ (above y z)))
      --> (@ (above x z))"
ax3: "ALL x::Block. (~ (@ (green x))) <-> (@ (red x))"

axioms

min_rule: "[| P : Obs; P ~: Delta |] ==> ~(@ P)"
```

The first three axioms are just a translation of the axioms of Example 3.1

into the notation of the operators in HOL. `ALL` is the universal quantifier, `-->` is the implication operator, `~` is the not operator and `<->` is the iff operator. Here we need to tag the predicates with `@` as they are now embedded inside the HOL operators.

The next axiom is actually a rule of inference, it is the Minimum Axiom Schema of Υ^M (see axiom schema $(*)$ of Section 3.2). As was explained in Section 4.1, rules of inference are represented in Isabelle using the notation `[| A; B; ... F; G |] ==> T` where `A`, `B`, ..., `F`, `G` are the premises of the rule and `T` is the conclusion. For example, the Modus Ponens rule is represented in Isabelle as `[| P --> Q; P |] ==> Q`. Here, `P` and `Q` stand as variables so that the rule of inference can be instantiated with any two expressions of type `bool`. The notation `P:Obs` means set membership testing (i.e. $P \in \mathcal{O}$) and `P~:Delta` means $P \notin \Delta$. In our case, `P` will act as a variable also, but it can only be instantiated by expressions of type `Formula` as the function `@` requires as argument a `Formula` expression and sets `Obs` and `Delta` are sets of expressions of type `Formula`. This tagging is required because the `min_rule` (i.e. Minimum Axiom Schema) is connecting a syntactical world (we need to test if an expression is in some sets) with the logical world in HOL.

These are all the required constructions, now it is time to start some proofs in Isabelle. Let us do a simple classical proof to exemplify the main ideas when doing a proof in Isabelle. To start a proof, we need to use the reserved word `lemma` or `theorem` (both words are used interchangeably in Isabelle) followed by an expression of the form `[| A; B; ... F; G |] ==> T`, where `A`, `B`, ..., `F`, `G` are our assumptions and `T` is the statement we want to prove. Observe that the notation is identical to the notation used for inference rules as they have exactly the same semantic status: We want to prove that given the assumptions, the conclusion holds, exactly as what an inference rule means. The difference is in their ontological status, a rule of inference is an axiom, while a lemma is something that must be proved first.

The “working set” that was mentioned in the conventions at the start of this section will be precisely the set of statements that are between the brackets `[| A; B; ... F; G |]` as we can add elements inside the brackets during a proof, as long as the elements we want to add were declared in the `axioms` section (see above). Also, by the conventions, we will need to add the elements of Δ in the working set right at the start.

Let us prove the sentence $\exists x: \text{Object above}(A, x)$ from the given $\Delta = \{\text{on}(A, B), \text{on}(B, T), \text{on}(C, D), \text{red}(A), \text{green}(C)\}$ (Δ was declared in the constant declaration section, see above). We write in Isabelle:

```
lemma "[| @ (on A (Inl B)); @ (on B (Inr T)); @ (on C (Inl D));
        @ (red A); @ (green C) |]
        ==> EX x::Object. @ (above A x)"
```

Now, after executing that line, Isabelle enters proof mode. Isabelle responds:

```
goal (1 subgoal):
1. [| @ on A (Inl B) ; @ on B (Inr T) ; @ on C (Inl D) ;
     @ red A ; @ green C  |]
   ==> EX x. @ above A x
```

As was explained in Section 4.1 when describing resolution of theorems, Isabelle works backwards: suppose that your current statement to proof is $[| A; \dots; C |] \implies B$ (in Isabelle terminology, the statement to proof is called a “sub-goal”), and you want to apply a rule of inference $[| R; \dots; V |] \implies T$. Isabelle will try to unify the conclusion B of the the current sub-goal with the conclusion T of the inference rule and if unification succeeds, Isabelle will add the instantiated conditions $R; \dots; V$ of the rule as new subgoals to prove. In our example, we can apply the existential introduction rule: $[| P \ x \ |] \implies \text{EX } z. P \ z$ as the conclusion of the rule $\text{EX } z. P \ z$ can be unified with the conclusion $\text{EX } x. @ \text{ above } A \ x$ of our current sub-goal. Note that P and x in the rule of inference act as free variables that can be substituted freely with any expression of the right type. How can $\text{EX } z. P \ z$ be unified with $\text{EX } x. @ \text{ above } A \ x$? Isabelle uses higher order unification due to the fact that Isabelle is implementing the logical constructs in a typed lambda calculus, so that the expression $\text{EX } x. @ \text{ above } A \ x$ can be seen as the expression $\text{EX } x. (\% \ z. @ \text{ above } A \ z) \ x$ where $\%$ is the lambda λ operator in Isabelle, so that when P is substituted by $(\% \ z. @ \text{ above } A \ z)$ in the rule of inference, the conclusion of the current sub-goal will unify with the conclusion of the rule. The new sub-goal becomes $(\% \ z. @ \text{ above } A \ z) \ ?a$ which is equivalent to $@ \text{ above } A \ ?a$ by β -reduction, where $?a$ is an unknown, as the variable x in the condition of the rule was not unified with anything. Unknowns are carried over in the proof until they can be unified with another expression.

Now, to apply the rule of existential introduction, we execute the command `apply(rule exI)` where `exI` is the identifier for the existential introduction rule of inference and `rule` is a tactic that tells Isabelle to unify the conclusion of the given rule with the conclusion of the current goal and add the subgoals that result to the current proof state. As explained in Section 4.1, a tactic in Isabelle is a computable function that maps proof states to sequences of proof states. Isabelle has many tactics that can be used in proof mode (see Isabelle manual [21]) and later, in Section 4.3 we will have to build a tactic to represent the decision procedure that will be required in Example 3.3 of Section 3.4.

Now, continuing with the example, Isabelle will respond:

```
> apply(rule exI)
```

```
goal (1 subgoal):
```

1. [| @ on A (Inl B) ; @ on B (Inr T) ; @ on C (Inl D) ;
 @ red A ; @ green C |]
 ==> @ above A ?x3

Next, we will add the axioms of the theory to the working set, as they will be needed later on. For that, we apply the tactic `insert` and provide the identifiers of the axioms, as they were declared in the `axioms` section (see declarations above):

```
> apply(insert ax1 ax2 ax3)
```

```
goal (1 subgoal):
```

1. [| @ on A (Inl B) ; @ on B (Inr T) ; @ on C (Inl D) ;
 @ red A ; @ green C ; ALL x y.
 @ on x y --> @ above x y ;
 ALL x y z. @ above x (Inl y) & @ above y z
 --> @ above x z ;
 ALL x. (~ @ green x) = @ red x |]
 ==> @ above A ?x3

Now, we apply the universal quantification elimination rule: [| ALL z. P z ;
 P x ==> R |] ==> R:

```
> apply(rule allE)
```

```
goal (2 subgoals):
```

1. [| @ on A (Inl B) ; @ on B (Inr T) ; @ on C (Inl D) ;
@ red A ; @ green C ;
ALL x y. @ on x y --> @ above x y ;
ALL x y z. @ above x (Inl y) & @ above y z
--> @ above x z ;
ALL x. (~ @ green x) = @ red x |]
==> ALL x. ?P5 x
2. [| @ on A (Inl B) ; @ on B (Inr T) ; @ on C (Inl D) ;
@ red A ; @ green C ;
ALL x y. @ on x y --> @ above x y ;
ALL x y z. @ above x (Inl y) & @ above y z
--> @ above x z ;
ALL x. (~ @ green x) = @ red x ; ?P5 ?x5 |]
==> @ above A ?x3

Observe that by the form of the `allE` rule, Isabelle introduced `?P5 ?x5` inside the working set of subgoal 2. This means that when we prove the first subgoal, `?P5` will be resolved automatically in the second subgoal. The first subgoal can be solved by assumption, by unifying it with the axiom `ALL x y. @ on x y --> @ above x y` in the working set. The unification becomes obvious if we transform the expression into its λ form: `ALL x. (% z. ALL y. @ on z y --> @ above z y) x`. In order to discharge the first subgoal by assumption we use the `assumption` tactic. The `assumption` tactic searches the working set, starting with the first statement, until it finds one statement that unifies with the conclusion of the current sub-goal, if this succeeds, the sub-goal is discharged.

```
> apply(assumption)
```

```
goal (1 subgoal):
```

1. [| @ on A (Inl B) ; @ on B (Inr T) ; @ on C (Inl D) ;
@ red A ; @ green C ;
ALL x y. @ on x y --> @ above x y ;
ALL x y z. @ above x (Inl y) & @ above y z

```

--> @ above x z ;
ALL x. (~ @ green x ) = @ red x ;
ALL y. @ on ?x5 y --> @ above ?x5 y []
==> @ above A ?x3

```

Observe how ?P5 was resolved. The expression ?P5 ?x5 in the working set of the second subgoal (which now is the only subgoal) is now ALL y. @ on ?x5 y --> @ above ?x5 y.

```
> apply(rule allE)
```

```
goal (2 subgoals):
```

```

1. [| @ on A (Inl B) ; @ on B (Inr T) ; @ on C (Inl D) ;
    @ red A ; @ green C ;
    ALL x y. @ on x y --> @ above x y ;
    ALL x y z. @ above x (Inl y) & @ above y z
    --> @ above x z ;
    ALL x. (~ @ green x ) = @ red x ;
    ALL y. @ on ?x5 y --> @ above ?x5 y []
    ==> ALL x. ?P8 x
2. [| @ on A (Inl B) ; @ on B (Inr T) ; @ on C (Inl D) ;
    @ red A ; @ green C ;
    ALL x y. @ on x y --> @ above x y ;
    ALL x y z. @ above x (Inl y) & @ above y z
    --> @ above x z ;
    ALL x. (~ @ green x ) = @ red x ;
    ALL y. @ on ?x5 y --> @ above ?x5 y ; ?P8 ?x8 []
    ==> @ above A ?x3

```

Now, we want to unify the first subgoal with the expression ALL y. @ on ?x5 y --> @ above ?x5 y in the working set, so we apply `assumption` tactic.

```
> apply(assumption)
```

```
goal (1 subgoal):
```

```

1. [| @ on A (Inl B) ; @ on B (Inr T) ; @ on C (Inl D) ;
    @ red A ; @ green C ;

```

```

ALL x y. @ on x y --> @ above x y ;
ALL x y z. @ above x (Inl y) & @ above y z
                --> @ above x z ;
ALL x. (~ @ green x ) = @ red x ;
ALL y. @ on ?x5 y --> @ above ?x5 y ;
ALL y. @ on ?x8 y --> @ above ?x8 y  []
==> @ above A ?x3

```

But, the `assumption` tactic selected the axiom `ALL x y. @ on x y --> @ above x y` again! To solve this, we use the command `back` to cycle through the distinct unifications allowed in the `assumption` tactic until we obtain the desired one.

```

> back
> back
> back

```

goal (1 subgoal):

```

1. [| @ on A (Inl B) ; @ on B (Inr T) ; @ on C (Inl D) ;
    @ red A ; @ green C ;
    ALL x y. @ on x y --> @ above x y ;
    ALL x y z. @ above x (Inl y) & @ above y z
                --> @ above x z ;
    ALL x. (~ @ green x ) = @ red x ;
    ALL y. @ on ?x5 y --> @ above ?x5 y ;
    @ on ?x5 ?x8 --> @ above ?x5 ?x8  []
    ==> @ above A ?x3

```

Now, modus Ponens rule: `[| P --> Q; P |] ==> Q`.

```

> apply(rule mp)

```

goal (2 subgoals):

```

1. [| @ on A (Inl B) ; @ on B (Inr T) ; @ on C (Inl D) ;
    @ red A ; @ green C ;
    ALL x y. @ on x y --> @ above x y ;
    ALL x y z. @ above x (Inl y) & @ above y z

```

```

--> @ above x z ;
ALL x. (~ @ green x ) = @ red x ;
ALL y. @ on ?x5 y --> @ above ?x5 y ;
@ on ?x5 ?x8 --> @ above ?x5 ?x8 []
==> ?P11 --> @ above A ?x3
2. [| @ on A (Inl B) ; @ on B (Inr T) ; @ on C (Inl D) ;
@ red A ; @ green C ;
ALL x y. @ on x y --> @ above x y ;
ALL x y z. @ above x (Inl y) & @ above y z
--> @ above x z ;
ALL x. (~ @ green x ) = @ red x ;
ALL y. @ on ?x5 y --> @ above ?x5 y ;
@ on ?x5 ?x8 --> @ above ?x5 ?x8 []
==> ?P11

```

The first sub-goal can be discharged by assumption, by unifying it with `@ on ?x5 ?x8 --> @ above ?x5 ?x8` in the working set. And notice how almost all unknowns will be resolved.

```
> apply(assumption)
```

```
goal (1 subgoal):
```

```

1. [| @ on A (Inl B) ; @ on B (Inr T) ; @ on C (Inl D) ;
@ red A ; @ green C ;
ALL x y. @ on x y --> @ above x y ;
ALL x y z. @ above x (Inl y) & @ above y z
--> @ above x z ;
ALL x. (~ @ green x ) = @ red x ;
ALL y. @ on A y --> @ above A y ;
@ on A ?x3 --> @ above A ?x3 []
==> @ on A ?x3

```

Finally we apply `assumption`, as the current sub-goal can be unified with `@ on A (Inl B)` in the working set. Isabelle will respond that there are no more subgoals and so we apply the command `done` to close proof mode.

```
> apply(assumption)
```

```
goal:
No subgoals!
```

```
> done
```

Isabelle provides various tactics that implement automatic search for proofs (see [21] for a list of tactics that use the classical reasoner of Isabelle). They are useful to prove subgoals in a complex proof, but they will usually fail if the goal is too “complex” to prove. That is why Isabelle is a proof assistant, but Isabelle has means to connect with fast automated theorem provers, like vampire, and also, if there is a known algorithm to decide statements in your theory, Isabelle provides means to connect the decision procedure into the rich formalism in Isabelle (this is what we will do in Section 4.3).

For example, the proof we just did, can be done in one line using the tactic `blast` after we have inserted the axioms of the theory into the working set, but I wanted to do all the steps to exemplify the main concepts in Isabelle.

```
> lemma "[| @ (on A (Inl B)); @ (on B (Inr T)); @ (on C (Inl D));
          @ (red A); @ (green C) |]
          ==> EX x::Object. @ (above A x)"
> apply(insert ax1 ax2 ax3)
> apply(blast)
```

```
goal:
No subgoals!
```

```
> done
```

Now, let us do a proof of the sentence $\exists y: \text{Object} \neg \text{on}(A, y)$ using the Minimum Axiom Schema. I will skip the details until we require to use the `min_rule` (i.e. the Minimum Axiom Schema) as declared in the `axioms` section.

```
> lemma "[| @ (on A (Inl B)); @ (on B (Inr T)); @ (on C (Inl D));
          @ (red A); @ (green C) |]
          ==> EX y::Object. ~ @ (on A y)"
> apply(insert ax1 ax2 ax3)
```

```
> apply(rule exI)
```

```
goal (1 subgoal):
```

```
1. [| @ on A (Inl B) ; @ on B (Inr T) ; @ on C (Inl D) ;
    @ red A ; @ green C ;
    ALL x y. @ on x y --> @ above x y ;
    ALL x y z. @ above x (Inl y) & @ above y z
    --> @ above x z ;
    ALL x. (~ @ green x ) = @ red x  |]
==> ~ @ on A ?y3
```

At this point, we are required to prove the negation of an observation. The conclusion of the current sub-goal unifies with the conclusion of the `min_rule`, so we apply the rule.

```
> apply(rule min_rule)
```

```
goal (2 subgoals):
```

```
1. [| @ on A (Inl B) ; @ on B (Inr T) ; @ on C (Inl D) ;
    @ red A ; @ green C ;
    ALL x y. @ on x y --> @ above x y ;
    ALL x y z. @ above x (Inl y) & @ above y z
    --> @ above x z ;
    ALL x. (~ @ green x ) = @ red x  |]
==> on A ?y3 : Obs

2. [| @ on A (Inl B) ; @ on B (Inr T) ; @ on C (Inl D) ;
    @ red A ; @ green C ;
    ALL x y. @ on x y --> @ above x y ;
    ALL x y z. @ above x (Inl y) & @ above y z
    --> @ above x z ;
    ALL x. (~ @ green x ) = @ red x  |]
==> on A ?y3 ~: Delta
```

Now, we need to “unfold” the definition of the set of observations, for that, we use the tactic `unfold`. Then we apply `blast` for the membership testing to be done automatically. We do a similar thing for `Delta`.

```
> apply(unfold Obs_def)
> apply(blast)
```

```
goal (1 subgoal):
```

```
1. [| @ on A (Inl B) ; @ on B (Inr T) ; @ on C (Inl D) ;
    @ red A ; @ green C ;
    ALL x y. @ on x y --> @ above x y ;
    ALL x y z. @ above x (Inl y) & @ above y z
    --> @ above x z ;
    ALL x. (~ @ green x) = @ red x  |]
    ==> on A ?y3 ~: Delta
```

```
> apply(unfold Delta_def)
> apply(blast)
```

```
goal:
```

```
No subgoals!
```

```
> done
```

Observe that the conclusion `on A ?y3 ~:Delta` of the sub-goal has an unknown, this means that an enumeration needs to be done over the type `Object` to check if one of those substitutions is not in `Delta`. `blast` does the enumeration automatically.

Finally, we will prove that the current Δ does not have minimum models by showing that we can prove any formula from Δ . Observe that Δ is classically consistent with the theory, so it is impossible to prove a contradiction classically, but Δ is not specifying the color of the block B , this means that both $\Delta \models_W \neg red(B)$ and $\Delta \models_W \neg green(B)$ should be true (where \models is minimum model entailment) but this contradicts the axiom $\forall x:Block \neg green(x) \leftrightarrow red(x)$ in the theory, so we should be able to prove, under minimum model entailment, any sentence from this Δ in Υ^M . This case poses an interesting problem in Isabelle just at the start of the proof. The proof goes by proving a contradiction by using the not elimination rule: `[| P; ~P |] ==> A` where A is a free variable standing for any expression.

```
> lemma "[| @ (on A (Inl B)); @ (on B (Inr T)); @ (on C (Inl D));
```

```

      @ (red A); @ (green C) []
      ==> R"
> apply(insert ax1 ax2 ax3)
> apply(rule notE)

goal (2 subgoals):
  1. [| @ on A (Inl B) ; @ on B (Inr T) ; @ on C (Inl D) ;
      @ red A ; @ green C ;
      ALL x y. @ on x y --> @ above x y ;
      ALL x y z. @ above x (Inl y) & @ above y z
      --> @ above x z ;
      ALL x. (~ @ green x ) = @ red x  []
      ==> ~ ?P3
  2. [| @ on A (Inl B) ; @ on B (Inr T) ; @ on C (Inl D) ;
      @ red A ; @ green C ;
      ALL x y. @ on x y --> @ above x y ;
      ALL x y z. @ above x (Inl y) & @ above y z
      --> @ above x z ;
      ALL x. (~ @ green x ) = @ red x  []
      ==> ?P3

> apply(rule min_rule)

```

```

goal (3 subgoals):
  1. [| @ on A (Inl B) ; @ on B (Inr T) ; @ on C (Inl D) ;
      @ red A ; @ green C ;
      ALL x y. @ on x y --> @ above x y ;
      ALL x y z. @ above x (Inl y) & @ above y z
      --> @ above x z ;
      ALL x. (~ @ green x ) = @ red x  []
      ==> ?P6 : Obs
  2. [| @ on A (Inl B) ; @ on B (Inr T) ; @ on C (Inl D) ;
      @ red A ; @ green C ;
      ALL x y. @ on x y --> @ above x y ;
      ALL x y z. @ above x (Inl y) & @ above y z

```

```

--> @ above x z ;
ALL x. (~ @ green x ) = @ red x  []
==> ?P6 ~: Delta
3. [| @ on A (Inl B) ; @ on B (Inr T) ; @ on C (Inl D) ;
   @ red A ; @ green C ;
   ALL x y. @ on x y --> @ above x y ;
   ALL x y z. @ above x (Inl y) & @ above y z
   --> @ above x z ;
   ALL x. (~ @ green x ) = @ red x  []
==> @ ?P6

```

The problem is that the first subgoal ?P6 : Obs is too general for applying `blast`, as it involves an unknown formula! We will need to help Isabelle by selecting manually the formula in the set of observations that will produce the contradiction (we are interested in the predicate `red`). The following sequence of tactics will do exactly that.

```

> apply(unfold Obs_def)
> apply(rule CollectI)
> apply(rule exI)
> apply(rule exI)
> apply(rule exI)
> apply(rule exI)
> apply(rule exI)
> apply(rule disjI2)
> apply(rule disjI1)
> apply(rule refl)

```

goal (2 subgoals):

```

1. [| @ on A (Inl B) ; @ on B (Inr T) ; @ on C (Inl D) ;
   @ red A ; @ green C ;
   ALL x y. @ on x y --> @ above x y ;
   ALL x y z. @ above x (Inl y) & @ above y z
   --> @ above x z ;
   ALL x. (~ @ green x ) = @ red x  []
==> red ?y15 ~: Delta
2. [| @ on A (Inl B) ; @ on B (Inr T) ; @ on C (Inl D) ;

```

```

@ red A ; @ green C ;
ALL x y. @ on x y --> @ above x y ;
ALL x y z. @ above x (Inl y) & @ above y z
           --> @ above x z ;
ALL x. (~ @ green x ) = @ red x  []
==> @ red ?y15

```

```

> apply(unfold Delta_def)
> apply(blast)

```

goal (1 subgoal):

```

1. [| @ on A (Inl B) ; @ on B (Inr T) ; @ on C (Inl D) ;
    @ red A ; @ green C ;
    ALL x y. @ on x y --> @ above x y ;
    ALL x y z. @ above x (Inl y) & @ above y z
              --> @ above x z ;
    ALL x. (~ @ green x ) = @ red x  []
==> @ red D

```

So that `blast` selected D instead of B, but the result will be the same. The discharged goal proved $\sim @ \text{red D}$. Now we need to prove that $@ \text{red D}$ by showing that $\sim @ \text{green D}$. The rest is just classical reasoning and one last application of the `min_rule` to prove $\sim @ \text{green D}$.

```

> apply(rule allE)
> apply(assumption)
> back
> back
> apply(rule iffE)
> apply(assumption)
> apply(rule mp)
> apply(assumption)

```

goal (1 subgoal):

```

1. [| @ on A (Inl B) ; @ on B (Inr T) ; @ on C (Inl D) ;
    @ red A ; @ green C ;

```

```

ALL x y. @ on x y --> @ above x y ;
ALL x y z. @ above x (Inl y) & @ above y z
                --> @ above x z ;
ALL x. (~ @ green x ) = @ red x ;
(~ @ green D ) = @ red D ;
~ @ green D --> @ red D ;
@ red D --> ~ @ green D []
==> ~ @ green D

```

```

> apply(rule min_rule)
> apply(unfold Obs_def)
> apply(blast)
> apply(unfold Delta_def)
> apply(blast)

```

goal:

No subgoals!

> done

4.3 Implementing $\Upsilon^{M2}(\mathcal{I}, \gamma)$ using an example theory

In this section we will take the Blocks World theory of Example 3.3 of Section 3.4 to exemplify how the proof system $\Upsilon^{M2}(\mathcal{I}, \gamma)$ of Section 3.4 could be implemented in Isabelle and how we could carry out some proofs in $\Upsilon^{M2}(\mathcal{I}, \gamma)$. We will extend the example to four blocks instead of two.

As in Section 4.2, we will not represent the sequents $\mathcal{O}, \Delta, \Gamma \mid_W^{M2} \psi$ of $\Upsilon^{M2}(\mathcal{I}, \gamma)$ as such in Isabelle, for the same reasons as explained in Section 4.2. We will follow the same conventions as in Section 4.2 regarding the representation of the sequents, with the exception that we will not represent Δ as an external set, but we will assume that Δ is at all times in the working set. Here, we will add an extra convention regarding $\gamma(W \upharpoonright \mathcal{I})$, i.e. the computation of the decidable part of the theory. Even though the process of selecting the part of the theory that

lies inside the decidable fragment can be programmed in principle (as the decision is completely based on the form of the sentences in the theory), here we will take the approach of giving manually that part of the theory to the Minimum Axiom Schema. The reason is that we want to exemplify what can be achieved by the approach in terms of reasoning in the proof system instead of worrying on technical matters on how to get the part of the theory. Nevertheless, once we are given the part of the theory, we will not evade the technicalities involved in the representation of the decision procedure in Isabelle as this is the heart of the approach.

Also, there is a technical difference between the theory of Example 3.3 and the theory we will present in here. The difference lies in the representation of the type `Object`, as Isabelle uses a disjoint union to represent it, and the disjoint union representation introduces two unary functions `Inl` and `Inr` to map blocks to objects and tables to objects, respectively. So, technically speaking, the representation in Isabelle of the theory of Example 3.3 is not strictly in the decidable fragment described in the example, as the Isabelle representation is using function symbols. But, I want to point out that as the type domain of the functions are enumeration types, this means that any interpretation of the function symbols (i.e. a function) under any model of the theory will be a function of finite image, also as will be shown below in the theory file, every use of the function symbols in the theory always appear in closed terms (i.e. terms without variables), so that we can safely replace the expressions `Inl A`, `Inl B`, `Inl C`, `Inl D`, `Inr T` by new constants and specify that these new constants enumerate the type `Object`. Therefore, the theory is still decidable.

Having established the conventions we will follow, we start the definitions in Isabelle:

```
theory BWUpsilonM2
imports Main

begin

datatype Block = A | B | C | D
datatype Table = T

types
```

```

Object = "Block + Table"

datatype Formula = on Block Object | green Block | red Block |
                 above Block Object | under Object Block |
                 And Formula Formula

fun coercion :: "Formula => bool"   ("@ _ ") where
"@ (And x y)  = ((@ x) & (@ y))"

axioms

ax1: "ALL x::Block. ALL y::Object.
      (@ (on x y)) --> (@ (above x y))"
ax2: "ALL x::Block. ALL y::Block. ALL z::Object.
      ((@ (above x (Inl y))) & (@ (above y z)))
      --> (@ (above x z))"
ax3: "ALL x::Block. (~ (@ (green x))) <-> (@ (red x))"
ax4: "ALL x::Block. (@ (under (Inr T) x)) --> (@ (green x))"
ax5: "ALL x::Block. ALL y::Object.
      (@ (on x y)) <-> (@ (under y x))"

```

The definitions are almost identical to those in Section 4.2, but now the datatype `Formula` was extended with the predicate `under`. Also two extra axioms were added that establish interactions between observations. Now, we will add the definitions of the constants and all the constructions required for the Minimum Axiom Schema.

```

datatype Answer = yes | no

consts

provable :: "Formula => Answer"
Obs      :: "Formula set"

```

```

defs
Obs_def :
"Obs == {t. EX w::Block. EX x::Object. EX y::Block. EX z::Block.
        EX r::Object. EX s::Block.
        (t = (on w x) | t = (red y) |
         t = (green z) | t = (under r s)
        )}"

axioms

min_rule: "[| P : Obs; (provable P) = no |] ==> ~(@ P)"
pos_rule: "(provable P) = yes ==> @ P"

```

The declarations require a little introduction to properly understand them. Remember that the Minimum Axiom Schema (see axiom schema $(*)$ of Section 3.4) requires to test the condition $\Delta \not\vdash_{\gamma(W \cap \mathcal{I})} \psi$. To test the condition, we will provide a decision procedure. How can we represent a decision procedure in Isabelle? Isabelle allows the definition of general computable functions as long as we provide a proof in Isabelle which shows that the function terminates under any input. Providing a proof for termination of a function can be quite complex, specially for the kind of function we need to build. Nevertheless, Isabelle provides another way of defining a decision procedure: build your decision procedure externally, using some programming language, and then connect your decision procedure with the constructs provided by Isabelle. The advantage of this last approach is that it is not required to provide a proof for termination. The disadvantage is that if the external program is badly programmed, Isabelle will no longer guarantee soundness. This last approach is the one we will follow as we will not construct from scratch a decision procedure, but we will “wrap” one of the automated search tactics of Isabelle. This decision is justified due the form of the sentences in the part of the theory we will use to decide observations, as they are mainly implications and iff’s between atomic formulae and implications involving disjunctions between constants for the enumeration type axioms (see construction of class \mathcal{I} in Example 3.3 and see the sentences in the equivalent untyped form of the theory). So that the form of the theory is simple enough such that if the Isabelle tactic fails, that means that the atomic sentence we were testing is not provable.

Now, returning to the Isabelle file, the first datatype is just declaring a type with two values, this is the type that will be used for the output of the decision procedure. Next comes the constants, the decision procedure will be represented by the function symbol `provable` that maps expressions of type `Formula` to an answer `yes` or `no`, i.e. the formula is provable or not. If we want to be faithful to the Minimum Axiom Schema in $\Upsilon^{M^2}(\mathcal{I}, \gamma)$ (see axiom schema $(*)$ of Section 3.4) the `provable` function symbol should receive five arguments: the sentence to test ψ , the state description Δ , the theory W , the class \mathcal{I} and the choice function γ , but by the conventions at the start of the section, Δ will always be in the working set, and we will provide $\gamma(W \upharpoonright \mathcal{I})$ manually, but not to the function `provable` but to the tactic that invokes the decision procedure (this will become clearer later).

Regarding the constant for representing the set of observations, it is fairly the same as in Section 4.2 but now we added the predicate `under` to its definition set. Observe that `Delta` is no longer defined in the constants section, as by convention, Δ will always be in the working set.

Next comes the Minimum Axiom Schema (the `min_rule`), after all the reductions that were made to it by convention. One extra rule of inference was added: the `pos_rule`. The `pos_rule` is stating that if a `Formula` is classically provable from $\gamma(W \upharpoonright \mathcal{I})$, then it can be concluded. Obviously, everything that can be proved using the decision procedure can be proved from classical rules inside Isabelle, so that the extra rule is superfluous, but it will provide means to test the tactics that we will build later on.

What the decision procedure will do? Assuming the current goal in the proof state is `(provable P) = no` where `P` is a `Formula` that previously has been proved to be in `Obs` and given a list of axioms representing $\gamma(W \upharpoonright \mathcal{I})$, the decision procedure will do roughly the following:

- Select the first subgoal in the current proof state O .
- Extract `P` from the expression `(provable P) = no`.
- Start a new proof state N with one subgoal, such that the working set of the new proof state contains: the list of axioms provided and the working set of the first subgoal in the proof state O . The goal of the new proof state will be the expression: `@ P`.

- Execute the tactic that implements the search procedure, on the proof state N . If the tactic fails, discharge the first subgoal in the proof state O . If the tactic succeeds, state failure.

As stated before, in this example we will use a tactic already provided by Isabelle for doing the search, as the theory is simple enough to allow such a thing and also, because we are only interested in deciding atomic sentences. But any other algorithm for doing the search, for example, a resolution-based search, can be executed in the last step, as a resolution-based search also works for this simple Blocks World Theory. Why the “wrapping” of the tactic specified in the procedure above is needed and not just, for example, use the tactic in the last step of the procedure directly in proof mode without all those preparations? The reason is that failure of a tactic does not imply unprovability in general, which is why Isabelle will not allow discharging of a goal by normal means when a tactic fails. Therefore, all those preparations are required to discharge the goal at the end. This will become clearer as I present the required code to do exactly that.

The procedure described above will be programmed in ML. The most important reason is that Isabelle is written in ML, and therefore, this approach makes available all the low-level libraries in Isabelle. To write ML code inside an Isabelle file, the code needs to be put inside the construct `ML {* ... ML code goes here ... *}`. We will present the top level functions first and then proceed to explain the auxiliary functions, as this approach is better for explaining the functions. Observe that in practice the functions need to be declared in reverse order as to the order they are being declared in here: first auxiliary functions and then top level functions, as the Isabelle file is interpreted in the order in which it is declared.

```
ML {*
fun not_provable_tac thmList =
  fn st =>
    let
      val goal = nth (prems_of st) 0
      val fm = extract_observation goal
      val new_st = prepare_state goal fm
    in
      if execute_subproof new_st thmList
```

```

        then Seq.empty
        else resolve_tac [provability_oracle (fm, false)] 1 st
    end
end
*}

ML {*
fun provable_tac thmList =
  fn st =>
    let
      val goal = nth (prems_of st) 0
      val fm = extract_observation goal
      val new_st = prepare_state goal fm
    in
      if execute_subproof new_st thmList
        then resolve_tac [provability_oracle (fm, true)] 1 st
        else Seq.empty
    end
end
*}

```

As the name of both functions implies, the decision procedure is being built as a tactic, such that the decision procedure can be invoked at any moment in proof mode in Isabelle. Tactics in Isabelle are functions that return an expression of type `Thm.thm -> Thm.thm Seq.seq`. This is the reason both functions are returning a function as their return value. The type `Seq.seq` is an implementation of a lazy list. The reason of using lazy lists is because tactics transform a proof state into a possibly infinite sequence of proof states. The type `Thm.thm` represents theorems (i.e. expressions of the form $P \implies Q \implies \dots \implies R \implies S$). Proof states are represented in Isabelle as theorems, by nesting the sub-goals of the proof state. For example, imagine you are trying to prove the lemma $[| P; Q; R |] \implies Y$ and imagine your current proof state has two sub-goals:

```

goal (2 subgoals):
1. [| P; Q; R |] ==> S
2. [| P; Q; R; T |] ==> W

```

The entire proof state will be represented in Isabelle as the expression of type `Thm.thm`:

```
[| ([| P; Q; R |] ==> S); ([| P; Q; R; T |] ==> W);
  P; Q; R |] ==> Y
```

Having established the proof state representation in Isabelle, the explanation of what the functions are doing is clearer: Given a list `thmList` of expressions of type `Thm.thm` as argument (i.e. this list will represent $\gamma(W \cap \mathcal{I})$), both functions will return an anonymous function that receives as argument a proof state `st`. The anonymous function will extract the first subgoal from the proof state (using function `prems_of` that returns the list of premises in an expression of the form `[| P; Q; ...; R |] ==> S`). Then, assuming the current sub-goal has the form `[| P; Q; ...; R |] ==> provable fm = g`) where `fm` is a `Formula` and `g` is an `Answer`, it will extract `fm` using function `extract_observation`. Then, using the function `prepare_state`, it will prepare a new proof state of the form `[| ([| P; Q; ...; R |] ==> @ fm); P; Q; ...; R |] ==> @ fm` where `@` is the coercion function defined before in Isabelle (see declarations above). Finally, it will run the decision procedure on the new proof state using the function `execute_subproof`. Depending on the success of the decision procedure, `not_provable_tac` will report failure if the decision procedure succeeds (by giving as output an empty sequence: This is the standard way of reporting tactics failure in Isabelle) and if the decision procedure failed it will discharge the subgoal in the old proof state by using the oracle `provability_oracle` combined with the tactic `resolve_tac`. `provable_tac` will do exactly the opposite.

What does `resolve_tac [provability_oracle ...]` is doing? Oracles allow Isabelle to create arbitrary theorems based on the results of some external reasoner. The usual thing to do is to program the decision procedure inside the oracle, such that when the decision procedure outputs yes or no, the oracle generates the corresponding theorem. Here, I programmed the decision procedure outside the oracle, because I wanted to reuse the code of all the auxiliary functions in both tactics `not_provable_tac` and `provable_tac` but this is not really substantial for the concept. The oracle receives as parameters, a `Thm.term` and a boolean value. If the boolean value is `true`, the oracle will respond with the theorem `(provable fm) = yes`, if the boolean value is `false`, the oracle will respond with the theorem `(provable fm) = no`, where `fm` is the `Thm.term` that is being given to the oracle as a parameter. Observe that both tactics, `not_provable_tac` and `provable_tac`, decide which boolean value to send to the oracle based on what the decision procedure outputs when they evaluate the

function `execute_subproof`. Once the oracle responds with a theorem, the tactic `resolve_tac` is used to unify the conclusion of the first sub-goal in the proof state with the answer given by the oracle. In this way, the sub-goal will be discharged.

Therefore, we will use tactic `not_provable_tac` when we want to prove a sub-goal of the form `[| P; Q; ...; R |] ==> (provable fm) = no` for some Formula `fm`. And we will use tactic `provable_tac` when we want to prove a sub-goal of the form `[| P; Q; ...; R |] ==> (provable fm) = yes` for some Formula `fm`.

Now, I will explain the auxiliary functions and the oracle.

```
ML {*
fun extract_observation goal =
  let
    val (lhs, _) = HOLogic.dest_eq(
      HOLogic.dest_Trueprop(
        Logic.strip_imp_concl goal
      )
    )
  in dest_prov(lhs)
  end
*}

ML {*
fun dest_prov t =
  case t of
    @{term "provable"} $ fm => fm
  | anything => raise TERM ("dest_prov", [anything])
*}
```

`extract_observation` is decomposing the sub-goal given as parameter. The function `Logic.strip_imp_concl` will return the conclusion of the sub-goal given as parameter. The function `HOLogic.dest_Trueprop` will remove the coercion function `Trueprop` from the term. What is this function `Trueprop`? The theory `HOL` is built on top the theory `Pure`, so that `HOL` can use all the constructions available in `Pure`. In order for `HOL` to use the operators in `Pure`, `HOL` is doing something similar to what we are doing: it is coercing its expressions of type

`bool` into the type `prop` which is the type of expressions in `Pure`. `Trueprop` is not explicitly stated when an expression is built in Isabelle due to the fact that `Trueprop` has an abbreviation with no symbols (for example, the `coercion` function in our Blocks World Theory is being abbreviated with the symbol `@` and we cannot remove the symbol because that will create an ambiguity: Is it `Trueprop` or `coercion` the one that should be used?). Next, `HOLogic.dest_eq` extracts the expressions in the left-hand side and right-hand side of an expression with equality symbol. Finally, the left-hand side term is sent to the function `dest_prov` which will remove the function symbol `provable` from the term and return the argument that was given to `provable`.

`dest_prov` is extracting the argument to the function `provable` by using pattern matching over low-level term construction in Isabelle. But instead of building from scratch the term `provable fm`, where `fm` is a `Formula`, we use *antiquotations* [22]. Antiquotations have the form `@{id parameters}` (for a list of antiquotations, see [22]). Antiquotations allow ML code to refer to objects declared in the Isabelle theory file without the need to build the objects using low-level constructors. For example, if we do not use antiquotations in `dest_prov`, the expression `@{term "provable"}` will have to be written as `Const("BWUpsilonM2.provable", "BWUpsilonM2.Formula => BWUpsilonM2.Answer")` which is how the constant `provable` is constructed in Isabelle at low-level code. The operator `$` is used to concatenate terms, for example `red A` can be represented as `@{term "red"} $ @{term "A"}`.

```
ML {*
fun prepare_state goal fm =
  let
    val prems = Logic.strip_imp_premis goal
    val exp = Thm.ctrm_of @{theory} (
      Logic.list_implies(premis,
        HOLogic.mk_Trueprop (
          @{term "coercion"} $ fm
        )
      )
    )
  in
    trivial exp
```

```

end
*}

```

Given the sub-goal and the `Formula` we are interested in proving or disproving, `prepare_state` is building the proof state in which the decision procedure will act. First, it extracts the working set of the sub-goal using `Logic.strip_imp_premis`. Then, it builds the expression `[| prems |] ==> Trueprop coercion fm` (or equivalently `[| prems |] ==> @ fm`) using `Logic.list_implies` where `[| prems |]` is the working set of the original sub-goal given as input. Then, it verifies that the expression is well-formed using `Thm.cterm_of` where `@{theory}` is an antiquotation referring to the theory file we are declaring (note: the function `Thm.cterm_of` requires a reference to a theory in order to check that the term given as parameter will match the declarations in the theory given as parameter). If the term is well-formed, `Thm.cterm_of` will output an expression of type `Thm.cterm` which is the type of well-formed terms. Finally, it transform `[| prems |] ==> @ fm` into its nested form `[| ([| prems |] ==> @ fm); prems |] ==> @ fm` using function `trivial`, to make it a proof state.

```

ML {*
fun execute_subproof new_st thmList =
  let
    val tac = cut_facts_tac thmList 1 THEN
              slow_tac @{claset} 1
  in
    case Seq.pull(tac new_st) of
      NONE      => false
    | seqcell => true
  end
*}

```

First, `execute_subproof` builds a tactic using the *tactical* `THEN`. A tactical is a function that receives tactics as arguments and outputs a tactic. The tactical `THEN` receives two tactics as arguments, `THEN` will build a tactic that does the following: given a proof state, execute the first tactic on the proof state, then for each element on the resulting sequence of proof states, execute the second tactic and concatenate the results of the application of the second tactic.

The tactic `cut_facts_tac thmList 1` will add each element in the list of axioms `thmList` to the working set of sub-goal 1 (remember that this is the only sub-goal, as the new prepared proof state has only one sub-goal). Only rules with no premises are inserted [14]. The tactic `slow_tac @{claset} 1` uses depth-first search to prove sub-goal 1 [14] where `@{claset}` is an antiquotation that gives a reference to a set of axioms generated automatically by Isabelle. This auto-generated set of axioms contains axioms for the datatype declarations and simplification axioms declared previously in Isabelle. Observe that due to the argument `@{claset}`, the axioms we will provide manually in the argument list need to include only axioms in the Blocks World Theory as Isabelle will take care automatically of datatype axioms.

After the tactic `tac` is built, it executes the tactic on the proof state given as argument (remember that tactics are functions from proof states to sequences of proof states). If the resulting sequence is empty (the `NONE` case), that means that the tactic failed, so it outputs `false`. It will output `true` otherwise.

Observe that instead of using the tactic `slow_tac` (which uses a depth-first search) we could use any other search procedure instead, for example, a resolution-based search will do fine.

Now, we declare the oracle. The syntax for declaring an oracle in Isabelle is `oracle name = text`, which will produce the effect of turning the given ML expression `text` of type `'a -> Thm.ctrm` into an ML function of type `'a -> Thm.thm`, which is bound to the global identifier `name` [21]. So, in this sense, an oracle is used to generate expressions of type `Thm.thm` which can be used to unify with sub-goals, in order to discharge them.

```
oracle provability_oracle = {*
fn (fm, mode) =>
  let
    val posThm = HOLogic.mk_Trueprop (
      HOLogic.mk_eq(
        @{term "provable"} $ fm,
        @{term "yes"}
      )
    )
    val negThm = HOLogic.mk_Trueprop (
      HOLogic.mk_eq(
```

```

        @term "provable" $ fm,
        @term "no"
      )
    )
  in
    if mode
    then Thm.ctrm_of @theory posThm
    else Thm.ctrm_of @theory negThm
  end
*}

```

The oracle is giving as output an anonymous function. This anonymous function receives two arguments, one expression of type `Thm.term` and a boolean value (the `mode`) and outputs a `Thm.ctrm` expression as is required by the definition of an oracle (see above paragraph).

The anonymous function first builds the terms `(provable fm) = yes` and `(provable fm) = no` using `HOLogic.mk_eq` and `HOLogic.mk_Trueprop`, where `fm` is the `Thm.term` that is being given as a parameter. Then, if the boolean value of `mode` is `true`, the anonymous function will verify that `posThm` is well-formed using the function `Thm.ctrm_of` and if it is, it will output the `Thm.ctrm` expression `(provable fm) = yes`, where `fm` is the `Thm.term` that is being given as a parameter. If the boolean value of `mode` is `false`, the anonymous function will verify that `negThm` is well-formed using the function `Thm.ctrm_of` and if it is, it will output the `Thm.ctrm` expression `(provable fm) = no`.

Whenever the oracle `provability_oracle` is invoked, it will transform automatically the `Thm.ctrm` expression of the output of the anonymous function into an expression of type `Thm.thm`.

Now, it is time to do some proofs in this setting. Let us prove the sentence $\neg red(B) \wedge \neg on(B, B)$ using $\Delta = \{on(A, B), on(B, T), on(D, T), red(A), red(C)\}$.

```

> lemma "[| @ (on A (Inl B)); @ (on B (Inr T)); @ (on D (Inr T));
          @ (red A); @ (red C) |]
          ==> ~ (@ (red B)) & ~ (@ (on B (Inl B)))"
> apply(rule conjI)

goal (2 subgoals):

```

```

1. [| @ on A (Inl B) ; @ on B (Inr T) ; @ on D (Inr T) ;
    @ red A ; @ red C  |]
   ==> ~ @ red B
2. [| @ on A (Inl B) ; @ on B (Inr T) ; @ on D (Inr T) ;
    @ red A ; @ red C  |]
   ==> ~ @ on B (Inl B)

```

```
> apply(min_rule)
```

```
goal (3 subgoals):
```

```

1. [| @ on A (Inl B) ; @ on B (Inr T) ; @ on D (Inr T) ;
    @ red A ; @ red C  |]
   ==> red B : Obs
2. [| @ on A (Inl B) ; @ on B (Inr T) ; @ on D (Inr T) ;
    @ red A ; @ red C  |]
   ==> provable (red B) = no
3. [| @ on A (Inl B) ; @ on B (Inr T) ; @ on D (Inr T) ;
    @ red A ; @ red C  |]
   ==> ~ @ on B (Inl B)

```

```
> apply(unfold Obs_def)
```

```
> apply(blast)
```

```
goal (2 subgoals):
```

```

1. [| @ on A (Inl B) ; @ on B (Inr T) ; @ on D (Inr T) ;
    @ red A ; @ red C  |]
   ==> provable (red B) = no
2. [| @ on A (Inl B) ; @ on B (Inr T) ; @ on D (Inr T) ;
    @ red A ; @ red C  |]
   ==> ~ @ on B (Inl B)

```

At this point, we apply the `not_provable_tac`. What is the list of axioms we need to provide? As I discussed previously, Isabelle will take care of the datatype axioms, so we only need to provide axioms `ax4` and `ax5` (see declaration of the axioms of the theory at the start of this section), as these are the only axioms in the class \mathcal{I} (see Example 3.3 of Section 3.4) that are not datatype or enumeration

axioms. How can the tactic `not_provable_tac` be invoked? We use the tactic `tactic` which receives as a parameter a string with ML code. The ML code MUST evaluate to an expression of type `Thm.thm -> Thm.thm Seq.seq`, i.e. a tactic. Also, as the string has ML code, in order to reference the axioms `ax4` and `ax5` declared in the theory, we will need to use the antiquotations `@{thm axiom_id}`. So, we will invoke it like this: `tactic "not_provable_tac [@{thm ax4}, @{thm ax5}]"`.

```
> apply(tactic "not_provable_tac [@{thm ax4}, @{thm ax5}]")
```

```
goal (1 subgoal):
```

1. [| @ on A (Inl B) ; @ on B (Inr T) ; @ on D (Inr T) ;
 @ red A ; @ red C |]
 ==> ~ @ on B (Inl B)

```
> apply(rule min_rule)
```

```
> apply(unfold Obs_def)
```

```
> apply(blast)
```

```
> apply(tactic "not_provable_tac [@{thm ax4}, @{thm ax5}]")
```

```
goal:
```

```
No subgoals!
```

```
> done
```

What happens if we apply the `not_provable_tac` to a sub-goal that actually is provable? Suppose we want to prove $\neg green(B)$ which does NOT follow from Δ and the theory W under minimum model semantics.

```
> lemma "[| @ (on A (Inl B)); @ (on B (Inr T)); @ (on D (Inr T));  

          @ (red A); @ (red C) |]  

          ==> ~ (@ (green B))
```

```
> apply(rule min_rule)
```

```
> apply(unfold Obs_def)
```

```
> apply(blast)
```

```
goal (1 subgoal):
```

```

1. [| @ on A (Inl B) ; @ on B (Inr T) ; @ on D (Inr T) ;
    @ red A ; @ red C  |]
    ==> provable (green B) = no

> apply(tactic "not_provable_tac [@{thm ax4}, @{thm ax5}]")

*** empty result sequence -- proof command failed
*** At command "apply".

```

And if we try to use `provable_tac` instead, it will also fail, as `provable_tac` requires that the sub-goal be of the form `provable (green B) = yes`. So, we rectify and instead of trying to prove $\neg green(B)$ we will try to prove $green(B)$ which obviously follows:

```

> lemma "[| @ (on A (Inl B)); @ (on B (Inr T)); @ (on D (Inr T));
          @ (red A); @ (red C)  |]
          ==> @ (green B)"
> apply(rule pos_rule)

goal (1 subgoal):
1. [| @ on A (Inl B) ; @ on B (Inr T) ; @ on D (Inr T) ;
    @ red A ; @ red C  |]
    ==> provable (green B) = yes

> apply(tactic "provable_tac [@{thm ax4}, @{thm ax5}]")

goal:
No subgoals!

> done

```

The implementation I am providing still has a lot of areas for improvement. One example occurs when we try to prove a sentence with a quantifier, say $\exists x:Block \sim (\@ \text{ on } x \text{ (Inl B)})$ which follows from the current Δ (as, for example, $on(C, B)$ is not provable and therefore, by the Minimum Axiom Schema $\neg on(C, B)$ and so $\exists x:Block \neg on(x, B)$). After applying the existential introduction rule, the new goal is $\sim (\@ \text{ on } ?x \text{ (Inl B)})$ where $?x$ is an unknown.

Then, after applying the `min_rule` and `blast` to discharge the observation testing, we get the goal `provable (on ?x (In1 B)) = no`. But if we apply the tactic `not_provable_tac`, it will fail, because the statement `@ (on A (In1 B))` is in the working set and it will be unifiable with the goal when doing the sub-proof. So, whenever the tactic `not_provable_tac` finds unknowns in the goal, it needs to iterate over the possible values of the unknowns until it finds one substitution in which the decision procedure says `no`, or if the possible values are exhausted without getting a `no` from the decision procedure, then `not_provable_tac` can fail.

Although the implementation requires improvements, and still we require to add the code for selecting the part of the theory that lies in the class (which certainly can be done by using the ML libraries of Isabelle), we can see that the implementation captures the key idea of the proof system of Section 3.4: use a decision procedure to decide the observation. In contrast, Section 4.2 showed that the implementation of the proof system of Section 3.2 is almost identical to its formal definition. Nevertheless, this was a first exploration on the implementation of the proof systems Υ^M and $\Upsilon^{M2}(\mathcal{I}, \gamma)$. Also, the implementation of the ideas on the class of proof systems $\Upsilon^{M3}(\text{III}\vdash)$ of Section 3.5 need to be explored as future work.

Chapter 5

Conclusions

This report explained in Chapter 2 the concept of minimum models as presented in [2] and showed, by means of examples (Section 2.3), that the concept is useful to capture the “absence as negation” interpretation usually imposed on the states of systems that can be described by positive facts, in the presence of a theory that relates observational facts (i.e. the theory represents the “rules” of the system). The report presented some proof systems to deal with minimum model reasoning (Chapter 3). The first one was a simple proof system that works for “ideal” states (i.e. a state where all positive observations that hold in the system are included in the state description). Soundness and completeness results over minimum model entailment as well as effectiveness results were shown for the simple proof system (Section 3.2). Then, two kind of proof systems for dealing with non-ideal states were presented:

- The first type of proof system selects a part of the theory to decide classical non-entailment of observations on that part (where the selection is based completely on the form of the sentences in the theory) and assumes that the observational deductive closure of the state description, with respect to the selected part of the theory, is equal to the observational deductive closure of the state description with respect to the entire theory, in other words, that the selected part of the theory is expressive enough to entail all positive observations that can be entailed from the entire theory. Soundness and completeness results over minimum model entailment as well as effectiveness results were shown for this proof system (Section 3.4).

- The second type of proof system (which is actually a class of proof systems) drops any assumption on the state description or its observational deductive closure and concentrates on preserving soundness with respect to minimum model entailment and effectiveness of the proof system, but no longer worries about *full* completeness with respect to minimum model entailment (Section 3.5).

The report also presented a computational representation and semi-automation in the proof assistant Isabelle of some of the proof systems (Chapter 4). Technical details for implementing in Isabelle the proof systems of Sections 3.2 and 3.4 were explained in Sections 4.2 and 4.3, respectively, as well as some example proofs in the respective proof systems were carried out in Isabelle using toy versions of the Blocks World Theory.

All the proof systems on this report are based on the result given by Corollary 2.2 of Section 2.2.3. The corollary specifies that minimum model entailment is equivalent to classical entailment when we add as assumptions the negations of the observations that are not classically entailed from the state description and the theory (i.e. the “absence as negation” interpretation in the presence of a theory that relates observational facts). So, if there is a procedure or a set of rules of inference to decide when an observation is not classically entailed from the state description and the theory, we could add that procedure or that set of rules of inference to a classical proof system to get a proof system for minimum model reasoning. This leads to the following issues to solve, when constructing a proof system:

- How to build a set of rules of inference to represent classical non-entailment? Non-entailment or non-provability, when looked at the level of proof theory, seems to be a non-natural concept to capture by a set of rules of inference. Non-provability, unlike provability, seems to be a global property of the system, in the sense that we can only be sure that something is not provable unless we have exhausted *all* possible derivations in the system that could lead to what we want to prove. Provability, on the other hand, is a local property of the system, in the sense that it is enough to find only *one* derivation to be able to show that something is provable. So, it seems to me that if there is a set of rules of inference to represent non-provability, these inference rules will have to emulate a *search tree* as the rules of inference

will have to “remember” what has been explored so far in order to conclude that something is not provable. So, instead of representing non-provability by rules of inference, I took the approach of representing non-provability as an algorithmic procedure or something that is *external* to the proof system. This is the reason of why all the proof systems in this report are using a *Minimum Axiom Schema*. It is an *axiom schema* and not a rule of inference because we are connecting something non-sequent-like (an algorithmic entity in the conditions) to a sequent in the proof system (the conclusion of the axiom schema). This approach of connecting a non-sequent-like entity to a sequent in the proof system produces some issues during the automation of the proof system: we have to transform statements in the logic of the proof system into statement in the logic of the non-sequent-like entity, see for example how we had to introduce a coercion function \mathbb{C} in the Isabelle implementation of Sections 4.2 and 4.3 to talk about formulae being true in the HOL logic and then to remove the coercion function when we want to talk about formulae being an element of some set in the `Set` logic or formulae being not provable using some decision procedure, which is how the condition of the Minimum Axiom Schema is stated on those sections. It is unknown to me at this point if there is a way to state non-provability in a nice way in the context of proof theory, but for now, it seems to me that the best approaches for testing non-provability are the semantic ones, like searching for models that falsify the sentence and similar approaches (see for example the database-like theory discussion at the end of Section 3.5.1).

- When classical non-entailment is decidable? It is known that for general theories, non-entailment is undecidable and not even recursively enumerable. Each proof system on this report solves that question by imposing either a restriction on the kind of state descriptions or by giving up some nice property of the proof system, like completeness, as it seems that trying to come up with a proof system for minimum model reasoning that is sound, complete and effective for general theories and general state descriptions, is very difficult, if not impossible (see Section 3.3 for a discussion on the existence of such proof system). So, many of the ideas presented on this report were the result of trying to produce an effective proof system, as we are interested in the automation of minimum model reasoning. The techniques

used were: assuming that the state description is “ideal” i.e. a state where all positive observations that hold in the system are included in the state description (Section 3.2); splitting the theory by selecting the part of the theory that lies in some decidable fragment of first-order logic based on the form of the sentences in the theory (Section 3.4); giving up completeness, but trying to have some “degree of completeness” for the proof system to be useful (Section 3.5) and among those approaches that give up completeness: identify which properties of a theory produce a decidable theory at least when we want to decide non-entailment over quantifier-free sentences. The database-like theory approach of the last part of Section 3.5.1 was an attempt to identify that kind of properties.

Observe that the semantic definition of minimum models seems to be very natural: by Theorem 2.1 of Section 2.2.3 a minimum model is the one that satisfies the state description together with the negations of the observations that are not implied from the state description and the theory, which is just the “absence as negation” interpretation in the presence of a theory that relates observational facts. But, when we try to move from the semantic definition to an effective representation in proof theory of the semantic idea, the issues described above appear. Maybe the naive approach I took by taking the result in Corollary 2.2 of Section 2.2.3 as a way to patch a classical proof system by adding an Axiom Schema for minimum model reasoning was not a very good idea. Maybe there is a more elegant way to map the seemingly natural semantic idea to its proof theory equivalent, although it will have to cope, in some unknown way to me, with the inherent properties of classical non-entailment or non-provability described above.

The proof systems presented on this report are in principle programmable in a computer, but I want to point out that this report did not explore the computational complexity of the implementation of the proof systems presented in here. So, this report is not claiming anything on how much computationally intensive the proof systems will turn out. Also, there are a lot of areas to explore that were pointed out along the report, so it would be premature to arrive to a conclusion regarding how practical could be the automation of minimum model reasoning. I summarize some of the observations that were made along the report together with some comments regarding the usefulness of what was achieved on this report and possible areas of improvement as future work:

- At the end of Section 3.3 it was suggested the idea of having a different syntactical universe for the observations. Some implications were discussed: the proof system has to be “unidirectional” in order to sever the interaction between observations and theory and therefore, the proof system will be inherently incomplete with respect to minimum model entailment, also, the definition of satisfaction of observations by models needs to be changed as observations are no longer sentences in a typed, first-order signature \mathcal{L} . It could be worth exploring how expressive that kind of proof systems could be.
- The decidable fragments of first-order logic presented in the proof system of Section 3.4 turned out to be too restrictive, as many practical theories, usually used in the context of “evolvable systems” [1], cannot be expressed completely inside any of the fragments, and even if they are not expressible inside the fragment, the part of the theory that was left out could not guarantee the \mathcal{O} -closedness condition for all state descriptions that have minimum models under those theories or it could be very hard to test if the \mathcal{O} -closedness condition holds for those state descriptions when the theory is too complex.
- The above limitation motivated the class of proof systems of Section 3.5. The section presented two approaches to generate recursive admissible relations (see Section 3.5 for the definition of this concept), one based on a syntactic approach and another based on a semantic approach.

The syntactic approach does not look very promising, and sincerely, to me it looks like it is useful only for theories whose form is very simple (like Horn clauses and maybe other simple generalizations of it). Again, the difficulty resides in that we are trying to decide non-provability, and a recursive test for non-provability is highly dependent on the form of the theory. Nevertheless, it could be worth to look at how syntactic approaches for testing non-provability are being done in the literature.

On the other hand, the semantic approach (searching for finite models) looks more promising as it allows a broader form of sentences in the theory. The report presented a property called \mathcal{C} -frp (i.e. finite representation property for the set of sentences \mathcal{C}) which is a generalization of the finite model property, restricted to the set \mathcal{C} . The report focused when \mathcal{C} has sentences

of simple form, like atomic sentences or quantifier-free sentences, due to the fact that observations are usually of that form. The importance of the property resides in that a theory with the \mathcal{C} -frp is decidable when querying sentences in \mathcal{C} . The report then, went to try to identify what properties of a theory imply the \mathcal{C} -frp. The database-like properties were an attempt to make such identification, but still it seems that the database-like properties are still too restrictive for practical applications (although, they allow more theories than the decidable fragments of first-order logic presented in Section 3.4).

- Regarding the database-like properties, I am making the conjecture that the condition on the functions can be dropped and the condition on the existentials can be generalized to closed terms instead of just constants, and still the theory will have the $QFSen(\mathcal{L})$ -frp. The reason is that the proof of Theorem 3.13 in the last part of Section 3.5.1 requires only to build a finite structure that “looks like” a substructure of a general model of the theory and such that this constructed structure will preserve the quantifier-free sentences and the property in the claim regarding sentences with quantifiers. This could be accomplished by imposing a limit on how deep we are allowed to compose function symbols when building a term (for example, by adding to the signature \mathcal{L} a natural number N that specifies the allowed deepness, to get signature \mathcal{L}^N) and then, if we can show that for every natural number n , the finite theory has the $QFSen(\mathcal{L}^n)$ -frp, then the finite theory will have the $QFSen(\mathcal{L})$ -frp (in the standard signature \mathcal{L} without limits in the deepness of terms). But, of course, this conjecture requires testing. If the conjecture is true, this will allow a huge class of theories (almost any conceivable theory that appears in practical applications).
- Sections 4.2 and 4.3 presented an implementation of the proof systems Υ^M (Section 3.2) and $\Upsilon^{M2}(\mathcal{I}, \gamma)$ (Section 3.4).

The proof system Υ^M is so simple, that the Isabelle implementation of the proof system is almost identical to its formal definition. Isabelle allowed to represent the proof system Υ^M in a very elegant way, and its full automation

by building suitable tactics in Isabelle is certainly within practical range. Unfortunately, this proof system is only useful for “ideal” state descriptions. The proof system $\Upsilon^{M2}(\mathcal{I}, \gamma)$, on the other hand, is a lot more complex as it involves a lot of technical details while implementing it in Isabelle. The heart of the difficulty on the implementation of this proof system is that it uses a non-natural concept in the world of Isabelle: non-provability. Therefore, we had to rely on non-standard ways to represent non-provability by providing an external decision procedure programmed in ML and then connect it with Isabelle using oracles. In order to show the proof system in action, the report stated simplifications to the formal definition that were not fundamental to the key idea in the proof system (like giving manually the part of the theory that was in the class \mathcal{I}). The provided implementation still has a lot of areas of improvement, as it is unable to prove formulae with unknowns (see last paragraphs at the end of Section 4.3). Although it still needs lots of polishing, the full automation in Isabelle of this proof system is certainly within practical reach.

Just as a comparison, observe that the proof system Υ^M evades all this problems because of the “ideal” state description assumption. This assumption allows the system to replace the non-provability condition in the Axiom Schema by a non-element testing over a set, which can be represented naturally in Isabelle.

- Regarding the implementation in Isabelle of the class of proof systems in Section 3.5, their implementation will also have lots of technical details due to their algorithmic nature. It is unclear to me at this point if Isabelle would be a natural tool for representing those systems, unless we want to use Isabelle as a programming library for building reasoners, which is a more sensible thing to do for these kind of proof systems in which non-provability is the key concept.

Overall, as stated at the start of this chapter, the concept of minimum models seems to be a natural way to model systems whose states are described by positive observations and where we impose, to the state descriptions, an “absence as negation” interpretation in the presence of a theory relating observational facts. But, as we try to map the semantic concept into its proof theory equivalent, a lot of issues appear, mainly due to the presence of classical non-entailment at the

core of the concept. This report tried to explore solvable cases of the problem and exemplify some of those cases by providing computer implementations. Still, a lot of work needs to be done to exactly characterise the solvable cases.

Bibliography

- [1] H. Barringer, D. Rydeheard, and D. Gabbay. Modelling evolvable component systems: Part I - a logical framework. Full journal paper, to appear in the Journal of IGPL, see <http://www.cs.manchester.ac.uk/evolve>, 2009.
- [2] Howard Barringer and David Rydeheard. A note on logical description, observational orders and minimum models. <http://www.cs.man.ac.uk/~david/evolution/logic.pdf>, 2008.
- [3] Egon Börger, Erich Grädel, and Yuri Gurevich. *The classical decision problem*. Perspectives in Mathematical Logic. Springer-Verlag, 1997.
- [4] Peter Chapman. An automatic prover for sequent calculus in Isabelle. Research report, University of St Andrews, 2007. See <http://www.cs.st-andrews.ac.uk/~pc/AutomaticProver.pdf>.
- [5] D. W. Etherington, R. E. Mercer, and R. Reiter. On the adequacy of predicate circumscription for closed-world reasoning. *Computational Intelligence*, 1:11–15, 1985.
- [6] R. E. Fikes and N. J. Nilsson. Strips: A new approach to the application of theorem proving to problem solving. *Artificial Intelligence*, 2(3–4):189–208, 1971.
- [7] David Griffioen and Marieke Huisman. A comparison of PVS and Isabelle/HOL. In *Theorem Proving in Higher Order Logics*, number 1479, pages 123–142. Springer-Verlag, 1998.
- [8] J. McCarthy. Circumscription—a form of non-monotonic reasoning. *Artificial Intelligence*, 13:27–39, 1980.

- [9] J. McCarthy and P. J. Hayes. Some philosophical problems from the standpoint of artificial intelligence. In B. Meltzer and D. Michie, editors, *Machine Intelligence 4*, pages 463–502. Edinburgh University Press, 1969.
- [10] William McCune. Prover9 and Mace4. <http://www.cs.unm.edu/~mccune/prover9>.
- [11] Tobias Nipkow and Lawrence C. Paulson. Isabelle-91. In *Proceedings of the 11th International Conference on Automated Deduction: Automated Deduction*, volume 607 of *Lecture Notes In Computer Science*, pages 673–676. Springer, 1992.
- [12] Tobias Nipkow, Lawrence C. Paulson, and Markus Wenzel. *Isabelle/HOL — A Proof Assistant for Higher-Order Logic*, volume 2283 of *LNCS*. Springer, 2002.
- [13] Lawrence C. Paulson. *Isabelle - A Generic Theorem Prover (with a contribution by T. Nipkow)*, volume 828 of *Lecture Notes in Computer Science*. Springer, 1994.
- [14] Lawrence C. Paulson. Old isabelle reference manual. <http://www.cl.cam.ac.uk/research/hvg/Isabelle/dist/Isabelle/doc/ref.pdf>, 2009.
- [15] D. Perlis and J. Minker. Completeness results for circumscription. *Artificial Intelligence*, 28(1):29–42, 1986.
- [16] A. Riazanov and A. Voronkov. The design and implementation of VAMPIRE. *AI Communications*, 15(2):91–110, 2002.
- [17] P. Rondogiannis and W. W. Wadge. Minimum model semantics for logic programs with negation-as-failure. *ACM Trans. Comput. Logic*, 6(2):441–467, 2005.
- [18] S. J. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall Series in Artificial Intelligence, 2003.
- [19] Natarajan Shankar. PVS: Combining specification, proof checking, and model checking. In *Formal Methods in Computer-Aided Design*, pages 257–264, 1996.

- [20] J. C. Shepherdson. A sound and complete semantics for a version of negation as failure. *Theor. Comput. Sci.*, 65(3):343–371, 1989.
- [21] Makarius Wenzel. The Isabelle/Isar reference manual. <http://www.cl.cam.ac.uk/research/hvg/Isabelle/dist/Isabelle/doc/isar-ref.pdf>, 2009.
- [22] Makarius Wenzel and Amine Chaieb. SML with antiquotations embedded into Isabelle/Isar. <http://www4.in.tum.de/~wenzelm/papers/Isar-SML.pdf>. Programming Languages for Mechanized Mathematics Workshop (CALCULEMUS 2007).
- [23] Larry Wos, Ross Overbeek, Ewing Lusk, and Jim Boyle. *Automated reasoning: Introduction and Applications*. McGraw Hill, 1992.