The Dissertation Committee for Amelia J. Harrison

certifies that this is the approved version of the following dissertation:

Formal Methods for Answer Set Programming

Committee:

Vladimir Lifschitz, Supervisor

Robert S. Boyer

Isil Dillig

Warren A. Hunt, Jr.

Torsten Schaub

Formal Methods for Answer Set Programming

by

Amelia J. Harrison

Dissertation

Presented to the Faculty of the Graduate School of

The University of Texas at Austin

in Partial Fulfillment

of the Requirements

for the Degree of

Doctor of Philosophy

The University of Texas at Austin

December 2017

To Grandmother Betsy

Acknowledgments

It's not an exaggeration to say that were it not for many people other than myself this document would not exist. This is by no means an exhaustive list.

I've had the opportunity to collaborate with many wonderful and talented colleagues. Among them are Daniel Bailey, Martin Gebser, Yuliya Lierler, Roland Kaminski, Julian Michael, David Pearce, Dhananjay Raju, Mirek Truszczynski, Agustin Valverde, and Fangkai Yang.

Without friends to help maintain balance, it's entirely unclear if I would have been able to see this through. Backyard barbecues and Game of Thrones viewing parties have been essential. I feel especially lucky to have met early on, and shared much of this journey with Hannah Alpert-Abrams and Kate Mesh.

My family has been a constant source of reassurance and has helped me navigate many of the inevitable bouts of self-doubt that arise in the process of a Ph.D. This includes my parents, Jack and Betsy Harrison, who have always unequivocally encouraged me to find my own path. It includes my brother, Will Harrison, and my cousins, Beth Stevens, Sarah Harrison, Marshall Harrison, Joshua Harrison, and Reece Harrison, who have often provided welcome reminders that the world does not revolve around logic programming.

Family also now includes Karl Pichotta. Over the past five years he has become family, and in the past five months he became my husband (and incidentally, also became a Ph.D.). I have received support from him in so many forms. He

provided technical sanity checks when I thought the impossible was possible, and a clear-headed, pragmatic view of the state of my work when I thought the possible (e.g. finishing a dissertation) was impossible. Of all that I've gained in the past five years, his love and affection are the most treasured.

I owe many thanks to my committee members: Robert S. Boyer, Isil Dillig, Warren A. Hunt, Jr., and Torsten Schaub. Each in various ways has contributed substantially to this document. Were in not for the PHIL313K class taught by Bob, once upon a time, I may never have found my way to studying logic programming. In any case, it definitely would have taken even longer to get there. Warren planted the seed in my head to consider pursuing a doctorate, when he tutored me in an independent study course many years ago. Isil, in a class that I audited, inspired with her mastery of the art of commanding a classroom. Torsten has been an invaluable collaborator. Moreover, without him, there would be no GRINGO input language, which has played a key role in advancing answer set programming as a paradigm, and also plays a key role in the research presented here.

Finally, thanks upon thanks to Vladimir Lifschitz. Five years ago, I decided to study with Vladimir because I thought that if anyone could teach me how to be a mathematician it would be him. I hope that he's had some success in that, but much of what I've learned from Vladimir has little to do with mathematics. I have learned, for example, the value of doing things slowly and deliberately and of taking the time to consider the audience's point of view and feelings (whether for a presentation, an article, an email, or a live conversation). I've enjoyed tremendously sharing time, discoveries, and ideological perspectives. I couldn't have asked for a better advisor.

Formal Methods for Answer Set Programming

Amelia J. Harrison, Ph.D.
The University of Texas at Austin, 2017

Supervisor: Vladimir Lifschitz

Answer set programming (ASP) is a declarative programming paradigm for the design and implementation of knowledge-intensive applications, particularly useful for modeling problems involving combinatorial search. The input languages of the first ASP software systems had the attractive property of a simple, fully specified declarative semantics, making it possible to use formal methods to analyze ASP programs – to verify correctness, for example, or to show that two programs were equivalent. Since that time, many useful new constructs have been added to input languages. The increase in usability, however, has come at the expense of a fully specified semantics, as the semantics of newer constructs has not quite kept pace with the most general syntax that solvers can handle.

In this thesis, we will describe one approach to bridging the gap between mathematical formulations of the semantics of ASP languages and the current state of the languages themselves. Our approach is to view ASP programs as corresponding to infinitary formulas (formulas with infinitely long conjunctions and disjunctions).

In addition to a semantics for ASP programs, we would like to have general methods for analyzing programs based on that semantics. For example, we would like to know when we are licensed to replace one rule with another within a program, without changing the meaning of the program. Answers to questions of this nature are based on the notion of “strong equivalence”, well-known in the ASP community. Intuitively, two rules are strongly equivalent if they are interchangeable in the context of any program. In this thesis, we will describe how this notion can be extended to infinitary formulas. We show that strong equivalence between such formulas can be established using an infinitary system of natural deduction, and in many cases in a finite deductive system. We give general methods for simplifying the translations of some common ASP constructs, and provide examples showing how our semantics and the formal methods we develop can be used to prove program correctness.

Table of Contents

Chapter 1

Introduction

Answer set programming (ASP) is a declarative programming paradigm for the design and implementation of knowledge-intensive applications (Lifschitz, 2008; Brewka et al., 2011). ASP is particularly useful for modelling problems involving combinatorial search. It has been used, for instance, to design a space shuttle decision support system (Balduccini et al., 2001); a tool that automates the specification of configurations for configurable products in an e-commerce setting (Tiihonen et al., 2003); a system for inferring phylogenetic trees (Brooks et al., 2007); a search system for e-tourism (Ricca et al., 2010); and a workforce management tool for scheduling employees for the Gioia-Tauro port authority (Ricca et al., 2012). The common thread in all of these industrial applications is that they involve combinatorial search in some way.

To solve a problem using ASP, a programmer needs to encode the problem using a logic program. The program should describe what it means to be a solution to the problem. Then an ASP system can be used to find the answer sets (also called stable models) of the program.[1] Most ASP systems operate in two phases: first grounding a program by replacing all variables with ground terms that do not include arithmetic expressions, and then using methods similar to those used by SAT solvers to find the answer sets of the program. The first phase is called

[1]In the literature, a distinction is sometimes made between stable models and answer sets. In this document we use the terms interchangeably.

```
% place queens on the chess board
{ q(1..n,1..n) }.

% exactly 1 queen per row/column
:- X = 1..n, not #count{ Y : q(X,Y) } = 1.
:- Y = 1..n, not #count{ X : q(X,Y) } = 1.

% at most one queen per diagonal
:- D = 1..n*2-1, not #count { X,Y : q(X,Y), D=X-Y+n } <= 1.
:- D = 1..n*2-1, not #count { X,Y : q(X,Y), D=X+Y-1 } <= 1.
```

Table 1.1: A GRINGO encoding of the n-queens problem.

grounding and software that performs it is called a grounder; the second is called solving and software that performs it is called a solver.

Table 1.1 shows an example of an ASP program. This program encodes a solution to the n-queens problem: How can we place n queens on an $n \times n$ chessboard such that no queen attacks another? The program is written in the input language of the popular ASP grounder GRINGO. This grounder is the front-end component of the ASP system CLINGO. If we use CLINGO to run this program with $n = 8$ it produces 92 answer sets, each corresponding to a different solution to the 8-queens problem. In an answer set, each atom of the form `q(i,j)` tells us where to place a queen. If we identify the upper-left hand corner of the chessboard with the square (1,1), then the answer set

$$\mathtt{q}(1,6)\ \ \mathtt{q}(2,2)\ \ \mathtt{q}(3,7)\ \ \mathtt{q}(4,1)\ \ \mathtt{q}(5,3)\ \ \mathtt{q}(6,5)\ \ \mathtt{q}(7,8)\ \ \mathtt{q}(8,4),$$

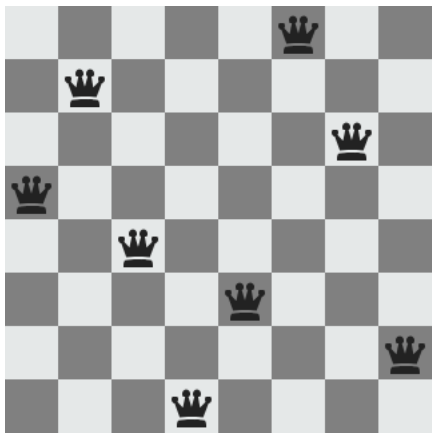
Figure 1.1: A solution to the 8-queens problem.

for example, corresponds to the solution shown in Figure 1.1.

When the first software systems for ASP were invented, one of their attractive features was that their input language had a simple, fully specified semantics, based on the concept of a stable model (Gelfond and Lifschitz, 1988). On the basis of this semantics, formal methods could be used to analyze ASP programs – to verify the correctness of a program, for example, or to show that two programs were equivalent. As the number of ASP users increased, useful new constructs were added to input languages. This increase in usability, however, came at the expense of a fully specified semantics, as proposals for the semantics of newer constructs often did not quite keep pace with the most general syntax that solvers could handle.

Here, we describe one approach to bridging the gap between mathematical formulations of the semantics of ASP languages and the current state of the languages themselves. This dissertation follows and builds upon previous work in this direction.

Ferraris (2005), for example, defines the semantics of ASP input languages using (possibly infinite) sets of finite propositional formulas (see Section 3.3). A shortcoming of that proposal is that it applies only to programs that do not contain variables. Eliminating global variables (like `X` in the second rule of Table 1.1) is straightforward enough: simply replace the rule by the set of rules formed by substituting the variable in question with elements of the domain. Eliminating local variables (like `Y` in the second rule of Table 1.1), however, is more complicated, and requires reasoning about the extents of predicates as defined by the program. For this reason, eliminating variables from a program sometimes requires reasoning about the entire program rather than one rule at a time. Our approach, on the other hand, is entirely modular in the sense that the translation of a rule does not depend on its context within a program.

Another translational approach to the semantics of ASP input languages was given by Ferraris et al. (2009). In that paper, the authors propose the use of first-order formulas to capture the meaning of ASP rules. That approach is modular: even when local variables are present it is always sufficient to look only at the rule in question to determine its first-order representation. For example, the variable `X` is local in the rule

$$\mathtt{q :-\ \#count\{X : p(X)\} = 0.} \tag{1.1}$$

Intuitively, this rule says that we can conclude `q` if the set `p` is empty. Its meaning can be captured by the first-order formula

$$\forall x\, \neg p(x) \ \rightarrow\ q. \tag{1.2}$$

However, the proposal to use first-order formulas as a target language falls short in another way. The rule

$$\mathtt{q(N) :-\ \#count\{X : p(X)\} = N.}$$

says, intuitively, that we may conclude $\mathtt{q(N)}$ for any numeral $\mathtt{N}$, if we've established that the cardinality of set $\mathtt{p}$ is exactly $\mathtt{N}$. The meaning of this rule cannot be captured using a first-order formula without significantly extending the language. We choose instead to use infinitary formulas. In Chapter 5, we show how to capture the meaning of this rule using an infinitary formula.

There are other translational approaches to the semantics of ASP input language that can capture the meaning of all of the constructs mentioned. Lee and Meng (2012) propose using formulas with generalized quantifiers as a target language, and Ferraris and Lifschitz (2010) propose using a first-order language extended by aggregates. Both of these approaches provide broad coverage of ASP constructs. On the other hand, we do not know of sound and complete deductive systems for reasoning about such formulas, and for our purposes a deductive system is an attractive feature.

Our proposal is to view ASP programs as corresponding to sets of infinitary formulas (formulas with infinite conjunctions and disjunctions). Using such formulas we can capture the meaning of all of the rules discussed above. Stable models for infinitary formulas were defined by Truszczynski (2012), generalizing Ferraris's (2005) definition of stable models for finite propositional formulas.

Our definition of the semantics focuses, in particular, on the input language of GRINGO, the front-end to the CLINGO ASP system. We have worked in collaboration with the system designers to define a precise syntax and semantics for a large subset of that input language, which we call Abstract Gringo, or AG. Perhaps one of the most valuable aspects of this line of work was that it helped us to clarify and in some cases refine the behavior of GRINGO in a number of ways. In this sense, the semantics we define has evolved in parrallel with the CLINGO system. In response to work on the semantics, for example, GRINGO changed the way partial functions such as division are treated. (This is just one small example, which easily admits description without reference to many technical details, but there have been a number of other such changes to the implementation of GRINGO prompted by this work.)

On the other hand, the focus on GRINGO was somewhat arbitrary in the sense that most ASP systems have very similar input languages. Indeed, there have been efforts to formalize a common central "core" of ASP input languages with a universal syntax and semantics (Calimeri et al., 2012). The language AG covers most of the constructs available in this core language (exceptions are extra-logical constructs like weak constraints and queries), as well as many not available there. It is believed that the semantics of AG also agrees with that of the core language in most regards (an exception is in the case of aggregates used recursively in the scope of negation), but that claim remains to be formally verified (Calimeri et al., 2012, Section 6.3).

After providing some history and background in Chapters 2 and 3, in Chapters 4 and 5, we describe the syntax and semantics of a subset of the AG language that we believe covers most naturally occurring programs. We present a subset, rather than the full language described by Gebser et al. (2015), because narrowing the focus in this way eases exposition, but moreover because we believe that a judicious selection of syntactic constructs can improve the quality of code. In addition to a semantics for ASP programs, we would like to have general methods for analyzing programs based on that semantics. In the remaining chapters we discuss methods for doing just that.

For example, we would like to be able to say when we are licensed to replace one rule with another within a program, without changing the meaning of the program. Answers to questions of this nature are based on the notion of "strong equivalence" (see Section 3.4). Intuitively, two rules are strongly equivalent if they are interchangeable in the context of any program. In Chapter 6, we describe how the notion of strong equivalence can be extended to infinitary formulas and show that strong equivalence between such formulas can be established using an infinitary system of natural deduction. Because the semantics of AG is in terms of infinitary formulas, this provides us with a method for characterizing strong equivalence for AG rules and programs.

In general, proofs in the infinitary deductive system used to reason about strong equivalence are infinite objects. This is somewhat unsatisfying in the sense that part of the appeal of a formalized deductive system is that finite proof objects can establish the validity of assertions about infinite domains. It turns out that in many cases, we can indeed establish the validity of infinitary formulas using proofs in a finite deductive system. Methods for doing so are described in Chapter 7.

In our proposed semantics, the translation of an aggregate atom is often an unwieldy formula. However, using strongly equivalent transformations it is often possible to simplify this formula. In Chapter 8 we present some theorems that show how to simplify the translations of some aggregate atoms based on their syntactic form.

In Chapter 9, we introduce the notion of stable models for infinitary formulas with “extensional” atoms. Intuitively, these are “input” atoms whose truth value is not determined by the program itself, but rather defined externally. The concept of a stable model for infinitary formulas with extensional atoms enables us to establish that a result known for first-order formulas holds in the infinitary case as well: that under certain conditions, the stable models of a set of formulas (without extensional atoms) can be expressed in terms of the stable models of its parts.

In Chapter 10, we demonstrate how the proposed semantics, as well as the formal methods developed in previous chapters, can be used to prove the correctness of the program shown in Table 1.1. In Chapter 11, we do the same for a more optimized ASP solution to the n-queens program that includes auxilliary predicates. For the case of the optimized program, the approach to proving correctness uses the results on splitting from Chapter 9.

In Chapter 12 we discuss the alternative semantics proposed by Calimeri et al. (2012). We describe an oversight in that semantics which prevents it from being broadly applicable. It does not apply to programs with local variables. We offer a correction to that proposal and discuss its relation to the AG semantics.

Finally, in Chapter 13 we conclude, and discuss directions for future work.

Chapter 2

A Short History of ASP

Some of the early work on the theory of logic programming was motivated by the desire to give a declarative semantics explaining the behavior of PROLOG (Sterling and Shapiro, 1986).

2.1 Positive Programs

The following is an example of a positive (logic) program:

$$\begin{array}{c} \mathtt{p(a).\ \ p(b).\ \ q(c).} \\ \mathtt{q(X) :- p(X).} \end{array} \tag{2.1}$$

This program is called "positive" because none of the rules contain negation. The first three rules of this program are called "facts". They establish that property `p` holds of the objects `a`, `b`, and `c`. The second rule of the program says that if we can establish that property `p` holds of an object, we may conclude that property `q` holds of that object as well.

Very early attempts to define the semantics of logic programs were limited to programs of this kind. In 1976, van Emden and Kowalski proposed that the meaning of a positive program be its minimal Herbrand model (where minimality is with respect to set inclusion). Indeed, such programs always have a unique minimal model. The minimal Herbrand model of (2.1), for example, is the set

$\{\mathtt{p(a), p(b), q(a), q(b), q(c)}\}$, which can be obtained by replacing the last rule with the following ground rules and finding the minimal model of the resulting program:

$$\begin{aligned}&\mathtt{q(a) :- p(a).}\\&\mathtt{q(b) :- p(b).}\\&\mathtt{q(c) :- p(c).}\end{aligned} \tag{2.2}$$

This definition of the meaning of a program aligns with our intuitive understanding: it consists of all the atoms stated as facts, and the atoms that can be generated from facts or from previously generated atoms.

In the rest of this chapter and in Chapter 3, for simplicity, we focus on programs without variables.

2.2 Programs with Negation

When we include negation in logic programs, things become a bit more complicated. Consider, for instance, a program containing the following rule with negation:

$$\mathtt{q :- r,\ not\ p.} \tag{2.3}$$

How can we intuitively describe the meaning of this rule? A first attempt might be to say that we can generate `q` if we can establish `r` and all attempts to establish `p` fail.[1] But this is a circular definition – an attempt to establish `r` might only succeed if we are able to establish `q`, which in turns depends on the failure of all attempts to

[1]This intuition explains why negation in logic programs is called "negation-as-failure."

establish r.

Clark (1978) proposed program completion as a way of understanding the meaning of programs with negation-as-failure. The definition of a "stable model" due to Gelfond and Lifschitz (1988) answered the question of how to understand negation-as-failure in a way that turned out to be closely related to default logic (Reiter, 1980). The definition of a stable model is based on a program transformation called the "reduct", which given a program (possibly containing negation-as-failure) and a candidate model produces a positive program. The candidate model is said to be "stable" if it is the minimal model of the positive program that results from applying the reduct operation. The meaning of a program is given by its stable models. (Details regarding the reduct operation are given in Section 3.1.)

As we saw in the previous section, positive programs always have a unique minimal model. Programs with negation-as-failure, on the other hand, may have more than one stable model, as demonstrated by the program

$$\begin{aligned} \mathtt{p} :- \mathtt{not}\ \mathtt{q}. \\ \mathtt{q} :- \mathtt{not}\ \mathtt{p}. \\ \mathtt{r} :- \mathtt{p}. \end{aligned} \tag{2.4}$$

This program has two stable models: $\{\mathtt{p}, \mathtt{r}\}$ and $\{\mathtt{q}\}$.

2.3 Disjunctive Programs

Gelfond and Lifschitz (1991) extended the definition of stable models to programs involving disjunction. For example, consider the following program:

$$\begin{aligned} &\mathtt{t.} \\ &\mathtt{r :- not\ q.} \\ &\mathtt{p\ ;\ s :- q.} \\ &\mathtt{q :- not\ r, t.} \end{aligned} \tag{2.5}$$

In this program, the symbol ; is understood to represent disjunction. The stable models of this program are $\{\mathtt{p}, \mathtt{q}, \mathtt{t}\}$, $\{\mathtt{q}, \mathtt{s}, \mathtt{t}\}$, and $\{\mathtt{r}, \mathtt{t}\}$.

A rule with nothing to the left of the : − symbol is called a *constraint*. For example, the last four rules of 1.1 are constraints. The effect of adding such a rule to a program is to eliminate those stable models that satisfy the body of the constraint. For example, consider the rule

$$\mathtt{:- q, p.}$$

If we add this rule to program (2.5) the set $\{\mathtt{p}, \mathtt{q}, \mathtt{t}\}$ is eliminated as a stable model.

Another extension proposed by Gelfond and Lifschitz (1991) is a second kind of negation, called “classical” or “strong” negation. For the sake of brevity, we omit a discussion of that construct in this document.

2.4 Answer Set Solvers

Answer set programming systems typically consist of two parts: a grounder, which takes a program and eliminates variables from it by replacing them with constants in the program domain, and a solver, which computes the stable models of the resulting variable-free program. Most ASP solvers use search methods similar to those used by fast satisfiability solvers.

The first answer set programming system was introduced by Niemelä and Simons in 1997. Its grounder was called PARSE, and its solver was called SMODELS. This system does not handle disjunctive programs. It originally was applicable only to programs like those discussed in Sections 2.1 and 2.2. The grounder for the system was later replaced by an improved grounder called LPARSE (Niemelä, 1999). The system GNT extended SMODELS to handle disjunctive programs (Janhunen et al., 2006).

Leone et al. (2006) introduced an answer set programming system that applied to disjunctive logic programs called DLV. Many efficient answer set solvers and systems have followed. Some, like CMODELS (Lierler, 2005), only handle solving, and use LPARSE as the grounder. Others, like CLINGO, include both grounder (in this case GRINGO) and solver (CLASP).

2.5 Choice Rules, Weight Constraints, and Aggregate Expressions

Niemelä et al. (1999) introduced choice rules to ASP input languages as a way to concisely represent choices among the elements of a set. For example, consider the rule

$$\{\mathtt{p}\} :- \ \mathtt{q}.$$

This rule says that if we have established `q` we may choose whether or not to include `p`. A program consisting of this rule and the fact `q` has two stable models: {`q, p`} and {`q`}. Prior to the introduction of choice rules all ASP programs had the "anti-chain property": no stable model of a program was a subset of another. Of course, this property may not hold for programs containing choice rules.

Niemelä et al. (1999) also allowed numerals to be placed on either side of the curly braces. In this case, the construct is called a "weight constraint" and one use is to concisely represent constraints on the cardinality of a set. For example, the expression

$$\mathtt{0}\ \{\mathtt{p},\mathtt{q}\}\ \mathtt{1} \tag{2.6}$$

is a weight constraint that expresses that either zero elements or exactly one element from the set $\{\mathtt{p},\mathtt{q}\}$ may be true, but not both. For a slightly more complicated example, consider

$$\mathtt{1}\ \{\mathtt{p}=\mathtt{1},\mathtt{q}=\mathtt{1},\mathtt{r}=\mathtt{2}\}\ \mathtt{2}. \tag{2.7}$$

To the left of each = symbol is an atom, and to the right is the corresponding weight. (In the case of (2.6) all atoms are implicitly assigned a weight of 1.) Weight constraint (2.7) says that the sum of the weights of the true elements among `p`,`q`, and `r` is at least 1, and at most 2. The sets that satisfy this expression are {`p`}`,` {`q`}`,` {`p,q`}`,` and {`r`}.

Aggregate expressions were introduced to ASP in the DLV system. Like weight constraints, they allow users to identify sets of atoms that satisfy certain conditions, but they are more general than weight constraints. A weight constraint expresses that the sum of the weights of the elements between the curly braces lies in some range. In an aggregate, an arbitrary function can be applied to those weights — not just summation. For example, the aggregate expression

$$\#\mathtt{max}\{\mathtt{X} : \mathtt{p(X)}\} > \mathtt{2}$$

is satisfied just in case the maximum value of X for which p(X) holds is strictly greater than 2. We saw other examples of rules containing aggregate expressions in Table 1.1.

Chapter 3

Background

In this chapter we review prior technical work including definitions and theorems that are useful for our own work.

3.1 Stable Models for Disjunctive Programs

As previously mentioned, ASP is based on the concept of a stable model. In this section, we give a formal definition of that concept. The definition we present here was given by Gelfond and Lifschitz (1991) and applies to disjunctive programs (see Section 2.3). A *(disjunctive logic) program* is a set of *(disjunctive) rules*, strings of the form

$$\mathtt{h}_1; \ldots ; \mathtt{h}_\mathtt{k} \ :- \ \mathtt{b}_0, \ldots , \mathtt{b}_\mathtt{m}, \mathtt{not}\ \mathtt{b}_{\mathtt{m}+1}, \ldots , \mathtt{not}\ \mathtt{b}_\mathtt{n}, \tag{3.1}$$

where $\mathtt{h}_1, \ldots , \mathtt{h}_\mathtt{k}, \mathtt{b}_0, \ldots , \mathtt{b}_\mathtt{n}$ are propositional atoms. A *literal* is an atom or an atom preceded by `not`. (The latter is called a *negative literal*.) In a rule (3.1), the literals $\mathtt{b}_0, \ldots , \mathtt{not}\ \mathtt{b}_\mathtt{n}$ are called the *body*; the atoms $\mathtt{h}_1, \ldots , \mathtt{h}_\mathtt{k}$ are called the *head*. A rule in which the body is empty and there is only one atom in the head is called a *fact*. In any rule in which the body is empty the symbol $:-$ may be dropped. A rule or program that does not contain any negative literals is called *positive*.

By a propositional interpretation we mean a function from the set of propositional atoms to truth values. Throughout this document, we identify an interpretation with the set of atoms satisfied by it. A *positive program* is one in which all rules are positive. An interpretation I is a *model* of a positive program Π (alternatively, interpretation I *satisfies* program Π) if for any rule R in Π such that all atoms in the body of R are in I, at least one atom from the head is also in I. The set I is a *minimal model of* Π if no proper subset of it is also a model.

If Π is a program and I is a set of atoms then the *reduct of* Π *with respect to* I, denoted Π^I is the positive program formed by dropping

- each rule in Π containing a negative literal whose atomic part occurs in I, and
- all negative literals from all remaining rules.

For example, the reduct of program (2.4) with respect to $\{\mathtt{p}, \mathtt{r}\}$ is

$$\begin{array}{r} \mathtt{p.} \\ \mathtt{r\ :-\ p.} \end{array} \tag{3.2}$$

Since p is an element of $\{\mathtt{p}, \mathtt{r}\}$ but also occurs in a negative literal in the body of the second rule of (2.4), that rule is dropped. The negative literal `not q` is also dropped from the body of the first rule. The last rule of the program, which contains no negative literals, remains unchanged.

An interpretation I is a *stable model* of a program Π if it is a minimal model of the reduct Π^I. For example, since $\{\mathtt{p}, \mathtt{r}\}$ is a model of (3.2) and none of its subsets are models of (3.2), we may conclude that it is a stable model of (2.4).

3.2 Equilibrium Logic

Equilibrium logic (Pearce, 1997) provides an alternative characterization of the stable models of a logic program that coincides with the definition for disjunctive logic programs from Gelfond and Lifschitz (1991). It is based on an intermediate propositional logic called the logic of here-and-there.[1] In equilibrium logic, logic programs are viewed as sets of propositional formulas and stable models are models of these formulas in the logic of here-and-there that meet an additional selection criteria. The definition of stable models based on equilibrium logic is more general than the original definition of stable models in that it applies to arbitrary propositional theories (not just those corresponding to logic programs).[2] A rule of the form (3.1) is rewritten as a propositional formula simply by writing it as an implication from left-to-right, with a conjunction over all literals in the body of the rule in the antecedent, all instances of `not` replaced by $\neg$, and a disjunction over the atoms in the head of the rule in the consequent. For example, program (2.5) is

[1]"Intermediate" because the set of valid formulas in the logic of here-and-there is a subset of the set of valid formulas under the classical semantics, but a superset of the set of valid formulas in intuitionistic logic (Heyting, 1930).

[2]In fact, the original paper on equilibrium logic gives a definition that is even more general than what we review here. It covers also programs with "strong negation" (see (Gelfond and Lifschitz, 1991)).

identified with the set containing

$$\begin{aligned}t,\\ \neg q \to r,\\ q \to p \vee s,\\ \neg r \wedge t \to q.\end{aligned} \tag{3.3}$$

Program (3.2) is identified with the set containing

$$\begin{aligned}p,\\ p \to r.\end{aligned} \tag{3.4}$$

Here, we refer to the logic of here-and-there as HT. The syntax of HT is the same as that of propositional logic. Formulas are propositional combinations of atoms or the symbol $\bot$. The connectives $\wedge$, $\vee$, and $\to$ viewed as primary. The expressions $\neg F$ and $F \leftrightarrow G$ are abbreviations for the formulas $F \to \bot$ and $(F \to G) \wedge (G \to F)$, respectively.

An *HT-interpretation* is an ordered pair $\langle I, J\rangle$ of sets of atoms such that $I \subseteq J$. Intuitively, the atoms in I are true "here" ("in the world H"), and the atoms in J are true "there" ("in the world T").[3]

The satisfaction relation between an HT-interpretation and a formula is defined recursively, as follows:

- For every atom p, $\langle I, J\rangle \models p$ if $p \in I$.

[3]Thus HT-interpretations can be thought of as linearly ordered Kripke models on two worlds.

- $\langle I, J\rangle \models F \wedge G$ if $\langle I, J\rangle \models F$ and $\langle I, J\rangle \models G$.
- $\langle I, J\rangle \models F \vee G$ if $\langle I, J\rangle \models F$ or $\langle I, J\rangle \models G$.
- $\langle I, J\rangle \models F \rightarrow G$ if

  (i) $\langle I, J\rangle \not\models F$ or $\langle I, J\rangle \models G$, and

  (ii) $J \models F \rightarrow G$.

An *HT-model* of a set $\mathcal{H}$ of formulas is an HT-interpretation that satisfies all formulas in $\mathcal{H}$.

Let us check, for instance, that the HT-interpretation $\langle \emptyset, \{p\}\rangle$ satisfies the formula $\neg\neg p$. According to the clause for implication, it is sufficient to check that $\langle \emptyset, \{p\}\rangle \not\models \neg p$ and $\{p\} \models \neg\neg p$. The second condition is obvious; to check the first, observe that $\{p\} \not\models \neg p$.

An HT-interpretation $\langle I, J\rangle$ is *total* if $I = J$. It is clear that a total HT-interpretation $\langle J, J\rangle$ satisfies F iff J satisfies F. An *equilibrium model* of a set $\mathcal{H}$ of formulas is a total HT-model $\langle J, J\rangle$ of $\mathcal{H}$ such that for every proper subset I of J, $\langle I, J\rangle$ is not an HT-model of $\mathcal{H}$.

The following theorem (a corollary to Proposition 2 from Pearce (1997)) characterizes the relationship between stable models and equilibrium models.

Theorem 1 (Pearce, 1997) *An interpretation J is a stable model of a program Π iff $\langle J, J\rangle$ is an equilibrium model of Π.*

This fact justifies the claim that equilibrium logic provides a more general definition of stable models.

3.3 Reducts for Arbitrary Propositional Theories

The definition of the reduct was generalized to arbitrary propositional theories by Ferraris (2005), providing another general definition of stable models.

Let F be a propositional formula formed from propositional atoms, and the connectives $\bot, \vee, \wedge$, and $\rightarrow$. The *reduct* F^I of formula F with respect to an interpretation I is defined recursively, as follows:

- $\bot^I$ is $\bot$.
- For every atom p, p^I is p if $p \in I$, and $\bot$ otherwise.
- $(F \wedge G)^I$ is $F^I \wedge G^I$.
- $(F \vee G)^I$ is $F^I \vee G^I$.
- $(G \rightarrow H)^I$ is $G^I \rightarrow H^I$ if I satisfies $G \rightarrow H$, and $\bot$ otherwise.

If $\mathcal{H}$ is a set of propositional formulas then the *reduct* $\mathcal{H}^I$ is the set $\{F^I : F \in \mathcal{H}\}$. An interpretation I is a *stable model* of a set $\mathcal{H}$ of formulas if it is minimal with respect to set inclusion among the interpretations satisfying the reduct $\mathcal{H}^I$.

For example, according to this definition, the reduct of (3.3) with respect to interpretation $\{p, q, t\}$ is

$$\begin{aligned} t&,\\ \bot \rightarrow \bot&,\\ q \rightarrow p \vee \bot&,\\ \neg\bot \wedge t \rightarrow q&. \end{aligned}$$

In general, the reduct of a program according to the definition presented in

Section 3.1 may look very different than the reduct according to the definition in this section. However, it turns out that the stable models are the same no matter which definition we use (Ferraris, 2005, Corollary 1). Furthermore, the definition of stable models according to Ferraris (2005) coincides also with the definition according to equilibrium logic:

Theorem 2 (Ferraris, 2005) *An interpretation I is a stable model of a set of propositional formulas $\mathcal{H}$ iff $\langle I, I\rangle$ is an equilibrium model of $\mathcal{H}$.*

3.4 Strong Equivalence

When dealing with ASP programs, a natural question that arises is "Can we simplify this rule without changing the meaning of the program?" This is a question that is of interest not only to programmers, but to designers of ASP systems as well. This question is relevant to system designers interested in automatic program optimizations similar to those performed by optimizing compilers. At first, it may seem like the question should have a simple answer. If we identify ground rules with propositional formulas, we might expect that we can replace any formula with an equivalent one without changing the meaning of a program. It turns out that if we understand equivalence in the sense of classical propositional logic, this is not the case. Consider, for example, the formulas

$$\neg p \rightarrow q \tag{3.5}$$

and

$$\neg q \rightarrow p. \tag{3.6}$$

These formulas are classically equivalent, however, even in isolation, they have different stable models: the unique stable model of (3.5) is $\{q\}$, while the unique stable model of (3.6) is $\{p\}$.

On the other hand, consider the formulas

$$p \rightarrow q \tag{3.7}$$

and

$$p \rightarrow r. \tag{3.8}$$

These formulas, taken in isolation, do have the same stable model (namely $\emptyset$). But consider what happens to each of these rules in the context of the fact p. A program formed from formula (3.7) and fact p has $\{p, q\}$ as its unique stable model, while a program formed from formula (3.8) and fact p has $\{p, r\}$ as its unique stable model. In order to justify swapping rules in logic programs, we need a notion of equivalence that is stronger than both classical equivalence and equivalence of stable models. In this section, we review prior work formalizing and characterizing such a notion of equivalence.

Let $\mathcal{H}_1$ and $\mathcal{H}_2$ be sets of propositional formulas. We say these sets are *strongly equivalent* if for every other set of formulas $\mathcal{H}$ the sets $\mathcal{H}_1 \cup \mathcal{H}$ and $\mathcal{H}_2 \cup \mathcal{H}$ have the same stable models. We say formulas F and G are strongly equivalent if the singleton sets $\{F\}$ and $\{G\}$ are strongly equivalent.

It turns out that strong equivalence is completely characterized by equivalence in the logic HT.

Theorem 3 (Lifschitz et al., 2001) *Sets of formulas $\mathcal{H}_1$ and $\mathcal{H}_2$ are strongly equivalent iff they have the same HT-models.*

3.5 Natural Deduction for the Logic of Here-and-There

In systems of natural deduction for classical propositional logic, theorems that can be proved without the use of the law of excluded middle are called *intuitionistically provable.*

The first axiomatization of the logic of here-and-there was given without proof by Jan Łukasiewicz (1941): add the axiom schema

$$(\neg F \to G) \to (((G \to F) \to G) \to G) \tag{3.9}$$

to propositional intuitionistic logic. This axiomatization was rediscovered and proved complete by Ivo Thomas (1962). (In the notation of that paper, schema (3.9) is $3''_2$.) The simpler axiom schema

$$F \vee (F \to G) \vee \neg G, \tag{3.10}$$

was proposed by Toshio Umezawa (1959), and the completeness of the system obtained by adding that schema to intuitionistic logic was proved by Tsutomu Hosoi (1966).

Here, we describe in full a system of natural deduction, which we call HT. It is almost identical to the classical system of natural deduction described by Lifschitz et al. (2008), except that it uses axiom schema (3.10) instead of the excluded middle axiom schema, $F \vee \neg F$, and does not include inference rules related to the unary connective $\neg$, which we view here as an abbreviation.

Derivable objects in HT are *sequents*—expressions of the form $\Gamma \;\Rightarrow\; F$, where F is a formula, and Γ is a finite set of formulas. (Such an expression can be read "F under assumptions Γ".) To simplify notation, we write Γ as a list. We identify a sequent of the form $\Rightarrow F$ with the formula F.

The axiom schemas of this system are

$$F \Rightarrow F$$

and (3.10).

The inference rules include introduction and elimination rules for the propositional connectives

$$(\wedge I)\ \frac{\Gamma \Rightarrow F \quad \Delta \Rightarrow G}{\Gamma, \Delta \Rightarrow F \wedge G} \qquad (\wedge E)\ \frac{\Gamma \Rightarrow F \wedge G}{\Gamma \Rightarrow F} \ \ \frac{\Gamma \Rightarrow F \wedge G}{\Gamma \Rightarrow G}$$

$$(\vee I)\ \frac{\Gamma \Rightarrow F}{\Gamma \Rightarrow F \vee G} \ \ \frac{\Gamma \Rightarrow G}{\Gamma \Rightarrow F \vee G} \qquad (\vee E)\ \frac{\Gamma \Rightarrow F \vee G \qquad \Delta_1, F \Rightarrow \Sigma \qquad \Delta_2, G \Rightarrow \Sigma}{\Gamma, \Delta_1, \Delta_2 \Rightarrow \Sigma}$$

$$(\rightarrow I)\ \frac{\Gamma, F \Rightarrow G}{\Gamma \Rightarrow F \rightarrow G} \qquad (\rightarrow E)\ \frac{\Gamma \Rightarrow F \quad \Delta \Rightarrow F \rightarrow G}{\Gamma, \Delta \Rightarrow G},$$

and the contradiction and weakening rules

$$(C)\ \frac{\Gamma \Rightarrow \bot}{\Gamma \Rightarrow F},$$

and

$$(W)\ \frac{\Gamma \Rightarrow F}{\Gamma, \Delta \Rightarrow F}.$$

A *proof* in this system is a finite list of sequents $S_1, \dots, S_n$, such that each sequent in the list is either an axiom or can be derived from sequents occurring earlier in the list using one of the inference rules. A sequent S will be called a *theorem of HT* if there exists a proof with S as the last sequent.

3.6 Infinitary Formulas and their Stable Models

Our proposal for the semantics of ASP programs uses infinitary propositional formulas. Truszczynski (2012) introduced definitions for such formulas and their stable models, albeit for a very different purpose: Infinitary stable models were used in that paper as a tool for relating first-order stable models to the semantics of first-order logic with inductive definitions, or FO(ID) (Denecker, 2000). The definitions presented in this section are equivalent to those introduced in that paper.

Let σ be a propositional signature, that is, a set of propositional atoms. For every nonnegative integer r, *(infinitary propositional) formulas (over σ) of rank r* are defined recursively, as follows:

- every atom from σ is a formula of rank 0,

- if $\mathcal{H}$ is a set of formulas, and r is the smallest nonnegative integer that is greater than the ranks of all elements of $\mathcal{H}$, then $\mathcal{H}^{\wedge}$ and $\mathcal{H}^{\vee}$ are formulas of rank r,
- if F and G are formulas, and r is the smallest nonnegative integer that is greater than the ranks of F and G, then $F \rightarrow G$ is a formula of rank r.

We write $\{F, G\}^{\wedge}$ as $F \wedge G$, and $\{F, G\}^{\vee}$ as $F \vee G$. The symbols $\top$ and $\bot$ are understood as abbreviations for $\emptyset^{\wedge}$ and $\emptyset^{\vee}$ respectively; $\neg F$ stands for $F \rightarrow \bot$, and $F \leftrightarrow G$ stands for $(F \rightarrow G) \wedge (G \rightarrow F)$. These conventions allow us to view finite propositional formulas over σ as a special case of infinitary formulas.

A set or family of formulas is *bounded* if the ranks of its members are bounded from above. For any bounded family $(F_\alpha)_{\alpha \in A}$ of formulas, we denote the formula $\{F_\alpha : \alpha \in A\}^{\wedge}$ by $\bigwedge_{\alpha \in A} F_\alpha$, and similarly for disjunctions.

Subsets of a signature σ are will also be called *interpretations*, as in the finite case. The satisfaction relation between an interpretation and a formula is defined recursively, as follows:

- For every atom p from σ, $I \models p$ if $p \in I$.
- $I \models \mathcal{H}^{\wedge}$ if for every formula F in $\mathcal{H}$, $I \models F$.
- $I \models \mathcal{H}^{\vee}$ if there is a formula F in $\mathcal{H}$ such that $I \models F$.
- $I \models F \rightarrow G$ if $I \not\models F$ or $I \models G$.

An infinitary formula that is satisfied by all interpretations will be called *tautological*.

The *reduct* F^I of an infinitary formula F with respect to an interpretation I

is defined recursively, as follows:

- For every atom p from σ, p^I is p if $p \in I$, and $\bot$ otherwise.
- $(\mathcal{H}^\wedge)^I$ is $\{G^I \mid G \in \mathcal{H}\}^\wedge$.
- $(\mathcal{H}^\vee)^I$ is $\{G^I \mid G \in \mathcal{H}\}^\vee$.
- $(G \to H)^I$ is $G^I \to H^I$ if $I \models G \to H$, and $\bot$ otherwise.

If $\mathcal{H}$ is a set of infinitary formulas then the *reduct* $\mathcal{H}^I$ is the set $\{F^I : F \in \mathcal{H}\}$. An interpretation I is a *stable model* of a set $\mathcal{H}$ of formulas if it is minimal with respect to set inclusion among the interpretations satisfying the reduct $\mathcal{H}^I$.

For example, if for every term t from an infinite set T, $p(t)$ is an atom

$$\bigwedge_{t \in T} \neg p(t) \;\to\; q \tag{3.11}$$

is an infinitary formula. It is one way to capture the meaning of rule (1.1) from the Introduction. The only stable model of the set containing this formula in isolation is $\{q\}$. To verify that $\{q\}$ is a stable model, we can show that the reduct of (3.11) with respect to $\{q\}$ is

$$\neg\bot \to q, \tag{3.12}$$

and that the minimal interpretation satisfing this formula is $\{q\}$.

Chapter 4

Abstract Gringo: Syntax

In this chapter, we describe the syntax of Abstract Gringo (AG for short), an ASP language that corresponds to a large subset of the GRINGO input language.[1] AG is abstract in that it does away with some of the complexities involved in encoding programs using ASCII characters, by using a richer character set. For example, in the GRINGO input language the semicolon plays many roles: for example, it is used to represent disjunction in the head of a rule, and to represent conjunction in the body of a rule. In AG, we use a different symbol for both of these functions. Table 4.1 shows the AG encoding of the n-queens program from Chapter 1 for comparison. (The lines beginning with $\%$ are comments, and are not valid AG expressions. We show those lines only to highlight the parallel with Table 1.1. Similarly, the right-hand column in the table displays labels for each of the rules; these are not part of the rules themselves.)

[1]Much of the material in this chapter was originally published in the following publications: Amelia Harrison, Vladimir Lifschitz, and Fangkai Yang. The semantics of Gringo and infinitary propositional formulas. In Proceedings of International Conference on Principles of Knowledge Representation and Reasoning (KR), 2014.

and

Martin Gebser, Amelia Harrison, Roland Kaminski, Vladimir Lifschitz, and Torsten Schaub. Abstract Gringo. Theory and Practice of Logic Programming, 15:449 463, 2015.

The definition of AG given here is more limited that that presented in that most recent paper. We offer this more limited definition for ease of exposition. My own contributions to those papers involved formalizing the definition of the syntax.

% place queens on the chess board

$\{q(\overline{1}..\overline{n}, 1..\overline{n})\}$ R_1

% exactly 1 queen per row/column

$\leftarrow X = \overline{1}..\overline{n} \wedge \textit{not count}\{Y : q(X,Y)\} = \overline{1}$ R_2
$\leftarrow Y = \overline{1}..\overline{n} \wedge \textit{not count}\{X : q(X,Y)\} = \overline{1}$ R_3

% at most one queen per diagonal

$\leftarrow D = \overline{1}..\ \overline{n}\times\overline{2}-\overline{1} \wedge \textit{not count}\{X,Y : q(X,Y), D = X - Y + \overline{n}\} \leq \overline{1}$ R_4
$\leftarrow D = \overline{1}..\ \overline{n}\times\overline{2}-\overline{1} \wedge \textit{not count}\{X,Y : q(X,Y), D = X + Y - \overline{1}\} \leq \overline{1}$ R_5

Table 4.1: Program K: An AG encoding of the n-queens problem.

4.1 Symbols and Terms

We assume that four sets of symbols are selected: *numerals*, *symbolic constants*, *variables*, and *aggregate names*. Moreover, we assume that these sets do not contain the symbols

$$+ \qquad - \qquad \times \qquad / \qquad .. \tag{4.1}$$

$$\textit{inf} \qquad \textit{sup} \tag{4.2}$$

$$= \qquad \neq \qquad < \qquad > \qquad \leq \qquad \geq \tag{4.3}$$

$$\bot \qquad \textit{not} \qquad \wedge \qquad \vee \qquad \leftarrow \tag{4.4}$$

$$, \qquad ; \qquad : \qquad (\qquad) \qquad \{ \qquad \} \tag{4.5}$$

and that they are pairwise disjoint. All these symbols together form the alphabet of AG, and AG rules are defined as strings over this alphabet.

When a symbol is represented in ASCII, its type is determined by its first two characters. For instance, a numeral starts with a digit or - followed by a digit. A symbolic constant starts with a lower-case letter. A variable starts with an upper-case letter, and an aggregate name starts with `#`. (The strings `#false`, `#inf`, and `#sup`, which represent $\bot$, *inf*, and *sup*, also start with `#`.) Each of the symbols (4.1)–(4.5) except for $\wedge$ and $\vee$ has a unique ASCII representation; the symbols $\wedge$ and $\vee$ are represented either by semicolons or by commas, depending on the context.

We assume that a 1–1 correspondence between the set of numerals and the set $\mathbf{Z}$ of integers is chosen. For every integer n, the corresponding numeral is denoted by $\overline{n}$.

Terms are defined recursively, as follows:

- all numerals, symbolic constants, and variables as well as the symbols *inf* and *sup* are terms;
- if f is a symbolic constant and $\mathbf{t}$ is a non-empty tuple of terms (separated by commas) then $f(\mathbf{t})$ is a term;

- if t_1 and t_2 are terms and $\star$ is one of the symbols (4.1) then $(t_1 \star t_2)$ is a term.

For example, the expression $\overline{1}..\overline{n}$ in R_1 is a term, as is the expression $X - Y + \overline{n}$ in R_4.

In a term of the form $(t_1 \star t_2)$ we drop the parentheses when it should not lead to confusion. A term of the form $(\overline{0} - t)$ can be abbreviated as $-t$.

A term, or a tuple of terms, is *precomputed* if it contains neither variables nor symbols (4.1). (So, for example, $g(a)$ and $\overline{3}$ are both precomputed terms, while $\overline{5} + \overline{2}$ and $\overline{1}..\overline{n}$ are not.) We assume a total order on precomputed terms such that *inf* is its least element, *sup* is its greatest element, and, for any integers m and n, $\overline{m} \leq \overline{n}$ iff $m \leq n$.

We assume that for each aggregate name α a function $\widehat{\alpha}$ is chosen that maps every set of non-empty tuples of precomputed terms to a precomputed term. For instance, we use *count* as an aggregate name; for any set T of tuples of precomputed terms, $\widehat{count}(T)$ is the numeral corresponding to the cardinality of T if T is finite, and *sup* otherwise.

4.2 Atoms, Literals, and Choice Expressions

An *atom* is a symbolic constant or a string of the form $p(\mathbf{t})$ where p is a symbolic constant and $\mathbf{t}$ is a non-empty tuple of terms. For instance, the expression $q(\overline{1}..\overline{n}, \overline{1}..\overline{n})$ in R_1 is an atom.

For any atom A, the strings

$$A \qquad \text{and} \qquad \textit{not } A$$

are *symbolic literals.* An *arithmetic literal* is a string of the form $t_1 \prec t_2$ where t_1, t_2 are terms and $\prec$ is one of the symbols (4.3). For instance, the expression $D = \overline{1}..\overline{n} \times \overline{2} - \overline{1}$ from R_4 is an arithmetic literal, while the expression $q(\overline{1}..\overline{n}, \overline{1}..\overline{n})$ is a symbolic literal.

An *aggregate element* is a string of the form

$$\mathbf{t} : \mathbf{L}, \tag{4.6}$$

where $\mathbf{t}$ is a tuple of terms, and $\mathbf{L}$ is a tuple of literals (if $\mathbf{L}$ is empty then the preceding colon may be dropped). An *aggregate atom* is a string of the form

$$\alpha\{E\} \prec s \tag{4.7}$$

where

- α is an aggregate name,
- E is an aggregate element,
- $\prec$ is one of the symbols (4.3),
- and s is a term.

For any aggregate atom A, the strings

$$A \qquad \text{and} \qquad \textit{not } A \tag{4.8}$$

are *aggregate literals.* Each of the rules R_2–R_5 in Table 4.1 contains an aggregate literal.

A *literal* is a symbolic, arithmetic, or aggregate literal.

A *choice expression* is a string of the form $\{A\}$ where A is an atom. Rule R_1 shows an example of a choice expression.

4.3 Rules and Programs

A *rule* is a string of the form

$$H_1 \lor \cdots \lor H_k \leftarrow B_1 \land \cdots \land B_n \tag{4.9}$$

or of the form

$$C \leftarrow B_1 \land \cdots \land B_n \tag{4.10}$$

($k, n \geq 0$), where each H_i is a symbolic literal, C is a choice expression, and each B_j is a literal. The expression $B_1 \land \cdots \land B_n$ is the *body* of the rule; $H_1 \lor \cdots \lor H_k$ is the *head* of (4.9); C is the *head* of (4.10). If the body of a rule is empty and the head is not then the arrow can be dropped. Rule R_1 is an example of a rule of form (4.10) with an empty body. Rules R_2–R_5 are of the form (4.9) with empty heads.

A *program* is a finite set of rules.

4.4 Abbreviations

An expression of the form

$$p(\mathbf{t}_1; \ldots; \mathbf{t}_k) \leftarrow B_1 \land \cdots \land B_n \tag{4.11}$$

where $k > 1$, each $\mathbf{t}_i$ is a non-empty tuple of terms, and each B_j is a literal, is an abbreviation for the set of k rules

$$p(\mathbf{t}_i) \leftarrow B_1 \wedge \cdots \wedge B_n \tag{4.12}$$

where $i = 1, \ldots, k$.

An expression of the form

$$l\ \{p(\mathbf{t}) : \mathbf{L}\}\ u \leftarrow B_1 \wedge \cdots \wedge B_n \tag{4.13}$$

where

- l and u are terms such that any variables occurring in them occur also in $B_1 \wedge \cdots \wedge B_n$,
- $p(\mathbf{t})$ is an atom,
- $\mathbf{L}$ is a tuple of arithmetic or symbolic literals (if $\mathbf{L}$ is empty then the preceding colon may be dropped),
- and each B_j is a literal

stands for the following set of rules:

$$\{p(\mathbf{t})\} \leftarrow B_1 \wedge \cdots \wedge B_n \wedge C, \tag{4.14}$$

$$\leftarrow B_1 \wedge \cdots \wedge B_n \wedge \mathit{count}\{\mathbf{x} : p(\mathbf{x}) \wedge \mathbf{x} = \mathbf{t} \wedge C\} < l, \tag{4.15}$$

$$\leftarrow B_1 \wedge \cdots \wedge B_n \wedge \textit{count}\{\mathbf{x} : p(\mathbf{x}) \wedge \mathbf{x} = \mathbf{t} \wedge C\} > u, \tag{4.16}$$

where C is the conjunction over all literals from $\mathbf{L}$; $\mathbf{x}$ is a tuple of distinct variables not occurring in (4.13); and $\mathbf{x} = \mathbf{t}$ is the conjunction of $x_i = t_i$ for all variables from $\mathbf{x}$ and terms from $\mathbf{t}$. The terms l and u are optional in (4.13). If l is absent then (4.15) should be dropped and if u is absent then (4.16) should be dropped.

Chapter 5

Abstract Gringo: Semantics

The semantics of AG is defined using a syntactic transformation τ that converts rules into infinitary formulas (see Section 3.6) formed from atoms of the form p or $p(\mathbf{t})$, where p is a symbolic constant, and $\mathbf{t}$ is a tuple of precomputed terms. Stable models of an AG program are stable models of the result of applying τ in the sense of Truszczynski (2012) (also reviewed in Section 3.6). Prior to defining the semantics of rules, we define the semantics of terms, literals, and choice expressions.[1]

5.1 Semantics of Terms

A term is *ground* if it does not contain variables. The definition of "ground" for arithmetic literals and symbolic literals is the same.

Semantically, every ground term t represents a finite set of precomputed terms $[t]$, which is defined recursively:

[1]Much of the material in this chapter was originally published in the following publications:

Amelia Harrison, Vladimir Lifschitz, and Fangkai Yang. The semantics of Gringo and infinitary propositional formulas. In Proceedings of International Conference on Principles of Knowledge Representation and Reasoning (KR), 2014.

and

Martin Gebser, Amelia Harrison, Roland Kaminski, Vladimir Lifschitz, and Torsten Schaub. Abstract Gringo. Theory and Practice of Logic Programming, 15:449 463, 2015.

My own contributions to those papers involved formalizing the semantics of many constructs, including intervals, choice expressions, and aggregate expressions.

- if t is a numeral, symbolic constant, or one of the symbols *inf*, *sup* then $[t]$ is $\{t\}$;
- if t is $f(t_1, \dots, t_n)$ then $[t]$ is the set of terms $f(r_1, \dots, r_n)$ for all tuples $r_1, \dots, r_n$ such that $r_1 \in [t_1], \dots, r_n \in [t_n]$;
- if t is $(t_1 + t_2)$ then $[t]$ is the set of numerals $\overline{n_1 + n_2}$ for all integers n_1, n_2 such that $\overline{n_1} \in [t_1]$ and $\overline{n_2} \in [t_2]$; similarly when t is $(t_1 - t_2)$ or $(t_1 \times t_2)$;
- if t is (t_1/t_2) then $[t]$ is the set of numerals $\overline{\lfloor n_1/n_2 \rfloor}$ for all integers n_1, n_2 such that $\overline{n_1} \in [t_1]$, $\overline{n_2} \in [t_2]$, and $n_2 \neq 0$;
- if t is $(t_1 \mathinner{..} t_2)$ then $[t]$ is the set of numerals $\overline{m}$ for all integers m such that, for some integers n_1, n_2,

$$\overline{n_1} \in [t_1], \qquad \overline{n_2} \in [t_2], \qquad n_1 \leq m \leq n_2.$$

For example, $[\overline{2} \times \overline{3}]$ is $\{\overline{6}\}$, $[\overline{1}..\overline{3}]$ is $\{\overline{1}, \overline{2}, \overline{3}\}$, and $[(\overline{1}..\overline{2}) + \overline{2}]$ is $\{\overline{3}, \overline{4}\}$.

If $\mathbf{t}$ is a tuple $t_1, \dots, t_n$ of terms $(n \neq 1)$ then $[\mathbf{t}]$ is the set of tuples $r_1, \dots, r_n$ for all $r_1 \in [t_1], \dots, r_n \in [t_n]$. For example, $[\overline{1}..\overline{n}, \overline{1}..\overline{n}]$ is the set of tuples $\overline{i},\ \overline{j}$ such that $1 \leq i, j \leq n$.

It is clear that if a ground term t does not contain symbolic constants then every element of $[t]$ is a numeral. If a tuple $\mathbf{t}$ of ground terms is precomputed then $[\mathbf{t}]$ is $\{\mathbf{t}\}$. The set $[t]$ can be empty. For example,

$$[\overline{1}..\overline{0}] = [\overline{1}/\overline{0}] = [\overline{1} + a] = \emptyset.$$

About a tuple of terms that does not contain .. we say that it is *interval-free.*

It is clear that if a tuple $\mathbf{t}$ of ground terms is interval-free then the cardinality of the set $[\mathbf{t}]$ is at most 1.

5.2 Semantics of Arithmetic Literals, Symbolic Literals, and Choice Expressions

In this section, we define τ for arithmetic and symbolic literals, as well as choice expressions.

For any ground arithmetic literal $\tau(t_1 \prec t_2)$ is $\top$ if the relation $\prec$ holds between some terms r_1 and r_2 such that $r_1 \in [t_1]$ and $r_2 \in [t_2]$, and $\bot$ otherwise. For example, $\tau(r = \overline{1}..\overline{n})$ is $\top$ if r is one of the numerals $\overline{1}, \ldots, \overline{n}$ and $\bot$ otherwise.

For any ground atom $p(\mathbf{t})$,

- by $[p(\mathbf{t})]$ we denote the set of atoms $p(\mathbf{r})$ for all tuples $\mathbf{r}$ in $[\mathbf{t}]$, and $\tau p(\mathbf{t})$ stands for the disjunction of these atoms;
- by $[$*not* $p(\mathbf{t})]$ we denote the set of formulas $\neg p(\mathbf{r})$ for all tuples $\mathbf{r}$ in $[\mathbf{t}]$, and $\tau($*not* $p(\mathbf{t}))$ stands for the disjunction of these formulas.

For example, $\tau p(\overline{2}..\overline{4})$ is $p(\overline{2}) \vee p(\overline{3}) \vee p(\overline{4})$.

For any tuple $\mathbf{L}$ of ground literals, $\tau\mathbf{L}$ stands for the conjunction of the formulas τL for all members L of $\mathbf{L}$. The expression $\tau\bot$ stands for $\bot$.

It is clear that if A has the form $p(\mathbf{t})$, where $\mathbf{t}$ is a tuple of precomputed terms, then τA is A.

The result of applying τ to a choice expression $\{p(\mathbf{t})\}$ is the conjunction of the formulas

$$p(\mathbf{r}) \vee \neg p(\mathbf{r})$$

over all tuples $\mathbf{r}$ in $[\mathbf{t}]$.

For instance, the result of applying τ to rule R_1 from Table 1.1 is

$$\bigwedge_{1 \le i,j \le n} \left(q(\bar{i}, \bar{j}) \vee \neg q(\bar{i}, \bar{j}) \right). \tag{5.1}$$

5.3 Global Variables

About a variable we say that it is *global*

- in a symbolic or arithmetic literal L, if it occurs in L;
- in an aggregate atom (4.7) if it occurs in s;
- in an aggregate literal (4.8) if it is global in A;
- in a rule (4.9), if it is global in at least one of the expressions H_i, B_j;
- in a rule (4.10), if it occurs in C or is global in at least one of the expressions B_j.

A variable will be called *local* in a rule R if it is not global in R. An *instance* of a rule R is any rule that can be obtained from R by substituting precomputed terms for all global variables. For example, X is global in R_2, so that the instances of R_2 are rules of the form

$$\leftarrow\ r = \overline{1} \,..\, \overline{n} \ \wedge\ \textit{not count}\{Y : q(r, Y)\} = \overline{1}$$

for all precomputed terms r. The variable D is global in R_4; instances of R_4 are rules of the form

$$\leftarrow r = \overline{1}..\overline{n} \times \overline{2} - \overline{1} \wedge \textit{not count}\{X, Y : q(X,Y), r = X - Y + \overline{n}\} \leq \overline{1}$$

for all precomputed terms r.

A literal or a rule is *closed* if it has no global variables. It is clear that any instance of a rule is closed.

5.4 Semantics of Aggregate Literals

In this section, the semantics of ground aggregates proposed by Ferraris (2005, Section 4.1) is adapted to closed aggregate literals.

If t is a term, $\mathbf{x}$ is a tuple of distinct variables, and $\mathbf{r}$ is a tuple of terms of the same length as $\mathbf{x}$, then the term obtained from t by substituting $\mathbf{r}$ for $\mathbf{x}$ is denoted by $t^{\mathbf{x}}_{\mathbf{r}}$.[2]

Let E be a closed aggregate atom of the form (4.7), and let $\mathbf{x}$ be the list of variables occurring in aggregate element $\mathbf{t} : \mathbf{L}$. By A we denote the set of tuples $\mathbf{r}$ of precomputed terms of the same length as $\mathbf{x}$. Let Δ be a subset of A. Then by $[\Delta]$ we denote the union of the sets $[\mathbf{t}^{\mathbf{x}}_{\mathbf{r}}]$ for all $\mathbf{r} \in \Delta$. We say that Δ *justifies* E with respect to a precomputed term[3] t if the relation $\prec$ holds between $\widehat{\alpha}[\Delta]$ and t. If t is

[2]Later, we use similar notation for the result of substituting $\mathbf{r}$ for $\mathbf{x}$ in expressions of other kinds, such as literals and tuples of literals, and also for the result of substituting terms for free occurrences of variables in formulas, and function and predicate names for free occurrences of function and predicate variables in second-order formulas.

[3]This definition of the semantics of aggregates is more complicated than that from (Gebser et al., 2015). There we say that a set Δ either justifies an aggregate atom or not, without reference to a

a precomputed term, we define $\tau_t E$ as the conjunction of the implications

$$\bigwedge_{\mathbf{r}\in\Delta} \tau(\mathbf{L}^{\mathbf{x}}_{\mathbf{r}}) \to \bigvee_{\mathbf{r}\in A\setminus\Delta} \tau(\mathbf{L}^{\mathbf{x}}_{\mathbf{r}}) \tag{5.2}$$

over all sets Δ that do not justify E with respect to t.

For any closed aggregate atom E,

- by τE we denote the disjunction of formulas $\tau_t E$ over all terms t in $[s]$, and
- by $\tau(\textit{not}\ E)$ we denote the disjunction of formulas $\neg\tau_t E$ over all terms t in $[s]$.

For example, let E be the closed aggregate atom occurring in the instance of R_2 resulting from substituting the precomputed term r for X:

$$\textit{count}\{Y : q(r,Y)\} = \overline{1}. \tag{5.3}$$

For any set S, by $|S|$ we denote the cardinality of S if S is finite, and ∞ otherwise. Let P be the set of all precomputed terms. Then τE is the same as[4] $\tau_{\overline{1}} E$ which is

$$\bigwedge_{\substack{\Delta\subseteq P\\ |[\Delta]|\neq 1}} \left(\bigwedge_{s\in\Delta} q(r,s) \to \bigvee_{s\in P\setminus\Delta} q(r,s) \right). \tag{5.4}$$

The result of applying τ to *not* E is the negation of this conjunction.

It is easy to see that in the case when E is a closed aggregate atom (4.7) such

particular precomputed term t. The version here corrects a discrepancy between the semantics and the behavior of GRINGO in the case when s represents a non-singleton set.

[4]Technically speaking, τE is the disjunction over the singleton set containing $\tau_{\overline{1}} E$.

that $[s] = \emptyset$, $\tau E = \tau(\textit{not } E) = \bot$. In Section 8.3 we give some results regarding how expressions such as τE can be simplified.

5.5 Semantics of Rules and Programs

Recall that for any bounded set S of formulas, by $S^\wedge$ we denote the conjunction over all elements of S (Truszczynski, 2012). For any rule R of form (4.9), τR stands for the set of the formulas

$$\tau B_1 \wedge \cdots \wedge \tau B_n \rightarrow [H_1]^\wedge \vee \cdots \vee [H_k]^\wedge \tag{5.5}$$

for all instances (4.9) of R. For a rule of form (4.10), τR stands for the set of the formulas

$$\tau B_1 \wedge \cdots \wedge \tau B_n \rightarrow \tau C$$

for all instances (4.10) of R.

For example, τR_2 is the set of formulas

$$\neg \big(\tau(r = \overline{1} \mathrel{..} \overline{n}) \wedge \tau(\textit{not count}\{Y : q(r, Y)\} = \overline{1})\big)$$

for all precomputed terms r. The expression $\tau(r = \overline{1}..\overline{n})$ was calculated in Section 5.2. The expression $\tau(\textit{not count}\{Y : q(X, Y)\} = \overline{1})$ was calculated in the previous section.

For any program Π, $\tau\Pi$ stands for the union of the sets τR for all rules R of Π. A *stable model* of a program Π is any stable model of $\tau\Pi$.

The last definition can be viewed as a specification for the answer set system CLINGO and other systems with the same input language. If such a system terminates given the ASCII representation of an AG program Π as input, and produces neither error messages nor warnings, then its output is expected to represent the stable models of Π.

Chapter 6

Strong Equivalence for Infinitary Formulas

In light of the semantics proposed in the previous section, in order to justify replacing one AG rule with another within the context of an arbitrary program, we need a theory of strong equivalence for infinitary propositional formulas. It turns out that much of the work on strong equivalence for finite propositional formulas (see Section 3.4) extends straightforwardly to the infinitary case.[1] Proofs of the theorems are postponed until the end of the chapter.

6.1 Infinitary Equilibrium Logic

In this section, we define equilibrium logic for infinitary propositional formulas. Most definitions are trivial extensions from the finite case (see Section 3.2). Just as in the finite case, an *HT-interpretation* of a propositional signature σ is an ordered pair $\langle I, J\rangle$ of interpretations of σ such that $I \subseteq J$.

The satisfaction relation between an HT-interpretation and a formula is defined recursively, as follows:

[1]Much of the material in this chapter was originally published in the following publications:
Amelia Harrison, Vladimir Lifschitz, and Miroslaw Truszczynski. On equivalence of infinitary formulas under the stable model semantics. Theory and Practice of Logic Programming, 15(1):1834, 2015.

and

Amelia Harrison, Vladimir Lifschitz, David Pearce, and Agustin Valverde. Infinitary equilibrium logic and strong equivalence. In Proceedings of International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR), pages 398410, 2015.

My own contributions to those papers included statements and proofs of some theorems.

- For every atom p from σ, $\langle I, J\rangle \models p$ if $p \in I$.
- $\langle I, J\rangle \models \mathcal{H}^\wedge$ if for every formula F in $\mathcal{H}$, $\langle I, J\rangle \models F$.
- $\langle I, J\rangle \models \mathcal{H}^\vee$ if there is a formula F in $\mathcal{H}$ such that $\langle I, J\rangle \models F$.
- $\langle I, J\rangle \models F \to G$ if

  (i) $\langle I, J\rangle \not\models F$ or $\langle I, J\rangle \models G$, and

  (ii) $J \models F \to G$.

An *HT-model* of a set $\mathcal{H}$ of infinitary formulas is an HT-interpretation that satisfies all formulas in $\mathcal{H}$.

The semantics of infinitary formulas defined above can be viewed as a 3-valued semantics as follows. About a formula F we say that it is *forced in the world* H of an HT-interpretation $\langle I, J\rangle$ if it is satisfied by $\langle I, J\rangle$; we say that it is *forced in the world* T if it is satisfied by J. The set of worlds in which F is forced will be called the *truth value* of F with respect to $\langle I, J\rangle$. It is easy to check by induction on the rank that every formula that is forced in H is forced in T as well. Consequently, the only possible truth values of a formula are $\emptyset$, $\{T\}$, and $\{H, T\}$.

As in the finite case, an HT-interpretation $\langle I, J\rangle$ is called *total* if $I = J$. A total HT-interpretation $\langle J, J\rangle$ satisfies F iff J satisfies F. An *equilibrium model* of a set $\mathcal{H}$ of infinitary formulas is a total HT-model $\langle J, J\rangle$ of $\mathcal{H}$ such that for every proper subset I of J, $\langle I, J\rangle$ is not an HT-model of $\mathcal{H}$.

The following theorem is similar to Theorem 2.

Theorem 4 (Harrison et al., 2015a) *An interpretation J is a stable model of a set $\mathcal{H}$ of infinitary formulas iff $\langle J, J\rangle$ is an equilibrium model of $\mathcal{H}$.*

6.2 Strong Equivalence and HT-models

About sets $\mathcal{H}_1, \mathcal{H}_2$ of infinitary formulas we say that they are *strongly equivalent* to each other if, for every set $\mathcal{H}$ of infinitary formulas, the sets $\mathcal{H}_1 \cup \mathcal{H}$ and $\mathcal{H}_2 \cup \mathcal{H}$ have the same stable models. About formulas F and G we say that they are *strongly equivalent* if the singleton sets $\{F\}$ and $\{G\}$ are strongly equivalent.

A *unary formula* is an atom or a formula of the form $p \to q$, where p and q are atoms. The following theorem is similar to the main theorem from Lifschitz et al. (2001). A part of that theorem was reproduced in Section 3.4.

Theorem 5 (Harrison et al., 2015a) *For any sets $\mathcal{H}_1$, $\mathcal{H}_2$ of infinitary formulas, the following conditions are equivalent:*

(i) $\mathcal{H}_1$ is strongly equivalent to $\mathcal{H}_2$,

(ii) for every set $\mathcal{H}$ of unary formulas, sets $\mathcal{H}_1 \cup \mathcal{H}$ and $\mathcal{H}_2 \cup \mathcal{H}$ have the same stable models;

(iii) sets $\mathcal{H}_1$ and $\mathcal{H}_2$ have the same HT-models.

For instance, the formulas $F \vee \neg F$ and $\neg\neg F \to F$, where F is an arbitrary infinitary formula, are strongly equivalent to each other. Indeed, an HT-interpretation does *not* satisfy $F \vee \neg F$ iff the truth value of F with respect to that interpretation is $\{T\}$, and the same holds for $\neg\neg F \to F$.

In Chapter 8 the relationship between strong equivalence and HT-models is used to simplify translations of aggregate atoms. In view of this relationship, we would like to have a deductive system for proving that infinitary formulas have the same HT-models. Indeed, such a system exists, but first we review C^∞, a deductive system for establishing the validity of infinitary formulas in classical logic.

6.3 Review: Natural Deduction for Classical Infinitary Logic

In this section, we describe a system of natural deduction for classical infinitary propositional logic, which we call C^∞. Although the precise formulation given here is our own, the soundness and completeness result discussed below is not new (Scott and Tarski, 1958).

As in the system of natural deduction for the finite logic of here-and-there, discussed in Section 3.5, derivable objects in the deductive system C^∞ are *sequents* (in this case infinitary)—expressions of the form $\Gamma \Rightarrow F$, where F is an infinitary formula, and Γ is a finite set of infinitary formulas. Again, we simplify notation by writing Γ as a list, and identify a sequent of the form $\Rightarrow F$ with the formula F.

The inference rules are the introduction and elimination rules for the propositional connectives

$$(\wedge I)\ \frac{\Gamma \Rightarrow H \quad \text{for all } H \in \mathcal{H}}{\Gamma \Rightarrow \mathcal{H}^\wedge} \qquad (\wedge E)\ \frac{\Gamma \Rightarrow \mathcal{H}^\wedge}{\Gamma \Rightarrow H} \quad (H \in \mathcal{H})$$

$$(\vee I)\ \frac{\Gamma \Rightarrow H}{\Gamma \Rightarrow \mathcal{H}^\vee} \quad (H \in \mathcal{H}) \qquad (\vee E)\ \frac{\Gamma \Rightarrow \mathcal{H}^\vee \qquad \Delta, H \Rightarrow F \quad \text{for all } H \in \mathcal{H}}{\Gamma, \Delta \Rightarrow F}$$

$$(\to I)\ \frac{\Gamma, F \Rightarrow G}{\Gamma \Rightarrow F \to G} \qquad (\to E)\ \frac{\Gamma \Rightarrow F \quad \Delta \Rightarrow F \to G}{\Gamma, \Delta \Rightarrow G},$$

where $\mathcal{H}$ is a bounded set of formulas, and the weakening rule

$$(W)\ \frac{\Gamma \Rightarrow F}{\Gamma, \Delta \Rightarrow F}.$$

The axiom schemas of C$^\infty$ are

$$F \Rightarrow F$$

and

$$\bigvee_{B \subseteq A} \left(\bigwedge_{\alpha \in B} F_\alpha \wedge \bigwedge_{\alpha \in A \setminus B} \neg F_\alpha \right) \tag{6.1}$$

where $\{F_\alpha\}_{\alpha \in A}$ is a non-empty bounded family of formulas. The latter generalizes the law of the excluded middle: if $A = \{1\}$ then (6.1) becomes

$$(F_1 \wedge \top) \vee (\top \wedge \neg F_1).$$

The set of *theorems of C^∞* is the smallest set of sequents that includes the axioms of the system and is closed under the application of its inference rules.

Notice that, in contrast with the finite system of natural deduction described in Section 3.5, theorems of C$^\infty$ are defined using closure properties rather than using proofs. We do not define the notion of a proof in the system C$^\infty$ at all. In fact, this is a stylistic choice. We could have defined proofs for the infinitary system, but they would, in general, have to be infinite objects.

System C$^\infty$ is sound and complete: An infinitary formula is a theorem of C$^\infty$ iff it is tautological. Proving soundness is straightforward. The proof of completeness is analogous to the proof of completeness for finite classical propositional logic due to Kalmár (1936). It can be presented using the following notation: for any interpretation I, M_I stands for the set

$$I \cup \{\neg p \mid p \in \sigma \setminus I\}.$$

It is easy to check by induction that for any formula F, $M_I^{\wedge} \to F$ is a theorem of C^{∞} if I satisfies F, and $M_I^{\wedge} \to \neg F$ is a theorem of C^{∞} otherwise. In particular, if F is tautological then $M_I^{\wedge} \to F$ is a theorem of C^{∞} for any interpretation I. On the other hand, the disjunction of the formulas $M_I^{\wedge}$ over all interpretations I is an instance of axiom schema (6.1): take $\{F_\alpha\}_{\alpha \in A}$ to be the family of all atoms. It follows that every tautological formula is a theorem.

6.4 Axiomatizing the Infinitary Logic of Here-and-There

The infinitary deductive system HT^{∞} is obtained from C^{∞} by replacing the generalized law of the excluded middle (6.1) with two axiom schemas:

$$F \vee (F \to G) \vee \neg G \tag{6.2}$$

and

$$\bigwedge_{\alpha \in A} \bigvee_{F \in \mathcal{H}_\alpha} F \to \bigvee_{(F_\alpha)_{\alpha \in A}} \bigwedge_{\alpha \in A} F_\alpha \tag{6.3}$$

for every non-empty family $(\mathcal{H}_\alpha)_{\alpha \in A}$ of sets of formulas such that its union is bounded; the disjunction in the consequent of (6.3) extends over all elements $(F_\alpha)_{\alpha \in A}$ of the Cartesian product of the family $(\mathcal{H}_\alpha)_{\alpha \in A}$. Axiom schema (6.3) generalizes one direction of the distributivity of conjunction over disjunction to infinitary formulas: if $A = \{1, 2\}$, $\mathcal{H}_1 = \{F_1, G_1\}$, and $\mathcal{H}_2 = \{F_2, G_2\}$, then (6.3) turns into

$$(F_1 \vee G_1) \wedge (F_2 \vee G_2) \to (F_1 \wedge F_2) \vee (F_1 \wedge G_2) \vee (G_1 \wedge F_2) \vee (G_1 \wedge G_2).$$

As in the classical case, the set of *theorems of HT$^\infty$* is the smallest set of sequents that includes the axioms and is closed under the inference rules. We say that formulas F and G are *equivalent in HT$^\infty$* if $F \leftrightarrow G$ is a theorem of HT$^\infty$.

The following theorem expresses the soundness and completeness of HT$^\infty$.

Theorem 6 (Harrison et al., 2015a) *An infinitary formula is a theorem of HT$^\infty$ iff it is satisfied by all HT-interpretations.*

It turns out that HT$^\infty$ remains complete if we require that formulas F and G in axiom schema (6.2) be literals and that the sets $\mathcal{H}_i$ in axiom schema (6.3) be finite. This is clear from the proof of the theorem (see Section 6.7.3).

From Theorems 5 and 6 we conclude:

Corollary 1 *Bounded sets $\mathcal{H}_1$, $\mathcal{H}_2$ of infinitary formulas are strongly equivalent iff $\mathcal{H}_1^\wedge$ is equivalent to $\mathcal{H}_2^\wedge$ in HT$^\infty$.*

6.5 Substitutions and Instances

The work presented in this section combines results originally proved in (Harrison et al., 2015b) with results from (Harrison et al., 2015a).

A part of any formula can be replaced with a strongly equivalent formula without changing the set of stable models. For instance, it is easy to check that the formulas $p \wedge \neg p$ and $\bot$ are strongly equivalent to each other; it follows that for any formula F, the formulas

$$F \wedge (q \rightarrow (p \wedge \neg p)) \qquad \text{and} \qquad F \wedge \neg q \tag{6.4}$$

have the same stable models. Proposition 1 below expresses a more general fact: several parts (even infinitely many) can be simultaneously replaced by strongly equivalent formulas.

Let σ and σ' be propositional signatures. A *substitution (from σ' to σ)* is a function ϕ that maps each atom from σ' to an infinitary formula over signature σ, such that the range of ϕ is bounded. A substitution is extended from propositional atoms to arbitrary infinitary formulas as follows:

- If F is $\mathcal{H}^{\wedge}$ then $\phi F = \{\phi G \mid G \in \mathcal{H}\}^{\wedge}$.
- If F is $\mathcal{H}^{\vee}$ then $\phi F = \{\phi G \mid G \in \mathcal{H}\}^{\vee}$.
- If F is $G \rightarrow H$ then $\phi F = \phi G \rightarrow \phi H$.

Formulas of the form ϕF will be called *instances* of F.

Consider, for instance, a pair of formulas (6.4) of signature σ. Let σ' be obtained from σ by adding the atom r, where r is a new atom. If ϕ is the function that maps r to $p \wedge \neg p$ and all other atoms to themselves, and ψ is the function that maps r to $\bot$ and all other atoms to themselves, then $\phi(F \wedge (q \rightarrow r))$ and $\psi(F \wedge (q \rightarrow r))$ are formulas (6.4).

Proposition 1 *Let ϕ and ψ be substitutions such that for all $p \in \sigma$, ϕp is strongly equivalent to ψp. Then for any formula F, ϕF is strongly equivalent to ψF, so that ϕF and ψF have the same stable models.*

By Theorem 5, the assertion of the proposition can be stated as follows: if for all $p \in \sigma$, ϕp and ψp have the same HT-models, then for any formula F, ϕF and ψF have the same HT-models. This is easy to check by induction on the rank of F.

We will refer to Proposition 1 as the *replacement property* of HT$^\infty$. To illustrate the replacement property, consider the formula

$$\bigwedge_{k\geq 1}(p_k \to \neg p_k) \to p_0. \tag{6.5}$$

It is strongly equivalent to

$$\bigwedge_{k\geq 1}\neg p_k \ \to\ p_0, \tag{6.6}$$

because (6.6) can be obtained from (6.5) by replacing $p_k \to \neg p_k$ with the strongly equivalent $\neg p_k$. More formally, let F be $\bigwedge_{k\geq 1} q_k \ \to\ p_0$. Let ϕ be the substitution that maps p_0 to p_0 and maps q_k to $p_k \to \neg p_k$ for all $k \geq 1$. Let ψ be the substitution that maps p_0 to p_0 and maps q_k to $\neg p_k$ for all $k \geq 1$. Then ϕF is (6.5), and ψF is (6.6). By the replacement property, (6.5) is strongly equivalent to (6.6).

The replacement property allows us to reason about the strong equivalence of AG programs in terms of particular constructs. For example, we can verify that the result of applying τ to the AG literal

$$\mathit{count}\{X : p(X)\} = 0 \tag{6.7}$$

is strongly equivalent to the result of applying τ to *count*$\{X : p(X)\} \leq 0$ conjoined with the result of applying τ to *count*$\{X : p(X)\} \geq 0$. The replacement property then justifies replacing (6.7) by this pair of aggregate literals in the body of any AG rule.

Proposition 2 *If F is a theorem of HT$^\infty$ then every instance of F is a theorem of HT$^\infty$ as well.*

The notation ϕF extends to sequents in a natural way. The proposition above is justified by the observation that the property "ϕS is a theorem of HT$^\infty$" holds for every axiom S of HT^∞, and it is preserved by all inference rules. We will refer to this as the *substitution property* of HT$^\infty$. To illustrate a use of this property, we will show that for any formulas F, G, the formula $\neg(F \vee G)$ is equivalent to $\neg F \wedge \neg G$ in HT$^\infty$. Note first that the formula

$$\neg(p \vee q) \leftrightarrow \neg p \wedge \neg q \tag{6.8}$$

is intuitionistically provable. It follows that it is a theorem of HT$^\infty$. The equivalence

$$\neg(F \vee G) \leftrightarrow \neg F \wedge \neg G$$

is an instance of (6.8): take $\phi p = F$, $\phi q = G$. By the substitution property, it follows that it is a theorem of HT$^\infty$.

6.6 Deriving Theorems in HT$^{\infty}$

In this section, we provide a couple of examples of how to establish that an infinitary formula is a theorem in HT$^{\infty}$ without using the substitution or replacement properties described in the previous section. For these examples, it is unclear how those properties might be useful.

It is easy to verify that the infinitary counterparts of the intuitionistically provable De Morgan's laws

$$\bigvee_{F\in\mathcal{H}} \neg F \rightarrow \neg \bigwedge_{F\in\mathcal{H}} F \tag{6.9}$$

and

$$\bigwedge_{F\in\mathcal{H}} \neg F \leftrightarrow \neg \bigvee_{F\in\mathcal{H}} F, \tag{6.10}$$

where $\mathcal{H}$ is a bounded set of formulas, are theorems in HT$^{\infty}$.

The remaining infinitary De Morgan's law,

$$\neg \bigwedge_{F\in\mathcal{H}} F \rightarrow \bigvee_{F\in\mathcal{H}} \neg F, \tag{6.11}$$

is also a theorem. We will show directly, without a reference to the general completeness theorem, how to prove (6.11) in HT$^{\infty}$. Let Q stand for the set of disjunctions including instances of the Hosoi axiom schema (3.10) for all formulas F, G from $\mathcal{H}$. Let $(\mathcal{H}_D)_{D\in Q}$ be the following family of sets:

$$\mathcal{H}_D = \{F, F \rightarrow G, \neg G\}.$$

Then the formula

$$\bigwedge_{D\in Q}\bigvee_{S\in\mathcal{H}_D} S \to \bigvee_{(S_D)_{D\in Q}}\bigwedge_{D\in Q} S_D, \tag{6.12}$$

(where the disjunction in the consequent extends over all elements $(S_D)_{D\in Q}$ of the Cartesian product of the family $(\mathcal{H}_D)_{D\in Q}$) is an instance of axiom schema (6.3). Since the antecedent of this implication is the conjunction of all formulas in Q, it is a theorem of HT$^\infty$. It follows that the consequent is a theorem as well. To complete the proof it is sufficient to show that from the antecedent of (6.11) and any disjunctive term

$$\bigwedge_{D\in Q} S_D \tag{6.13}$$

of the consequent of (6.12), we can derive the consequent of (6.11). Consider one of the conjunctions (6.13), and let C be set of its conjunctive terms. The elements of C are formulas of the forms

$$F,\ F\to G,\ \neg G.$$

If C contains $\neg F$ for some formula F then the consequent of (6.11) follows immediately. Otherwise, we will show that assuming $C^\wedge$ and any element F of $\mathcal{H}$ we can derive

$$\bigwedge_{F\in\mathcal{H}} F, \tag{6.14}$$

contradicting the antecedent of (6.11), and allowing us to derive $\neg F$ from $C^\wedge$ and the antecedent of (6.11). If C contains every formula F in $\mathcal{H}$ then (6.14) is immediate. Otherwise, there is some G from $\mathcal{H}$ that is not in C. Assume G. Since G is not in C and C does not contain the negation of any formula, we may conclude that C contains $G \to F$ for all formulas F from $\mathcal{H}$. It follows that from G and $C^\wedge$ we can derive (6.14).

The following formula is another example of a theorem in HT$^\infty$:

$$\bigvee_{J \subseteq I} \left(\neg \bigvee_{j \in I \setminus J} F_j \wedge \neg\neg \bigwedge_{j \in J} F_j \right) \tag{6.15}$$

for any non-empty bounded family $\{F_i\}_{i \in I}$ of formulas. (Just as axiom schema (6.1) generalizes the law of excluded middle to infinite sets of formulas, (6.15) generalizes the "weak law of the excluded middle"

$$\neg F \vee \neg\neg F. \tag{6.16}$$

Formula (6.16) corresponds to the special case of (6.15) where the family $\{F_i\}_{i \in I}$ has only a single element. While it is easy to verify that 6.16 is a theorem of HT$^\infty$, it is more difficult for the general case.) Indeed

$$\bigwedge_{i \in I} (\neg F_i \vee \neg\neg F_i)$$

is a theorem of HT$^\infty$ because $\neg F_i \wedge \neg\neg F_i$ can be intuitionistically derived from (3.10) with F_i as F and $\neg F_i$ as G. Using (6.3) we obtain

$$\bigvee_{J \subseteq I} \left(\bigwedge_{j \in I \setminus J} \neg F_j \wedge \bigwedge_{j \in J} \neg\neg F_j \right),$$

and (6.15) follows by De Morgan's laws.

6.7 Proofs

6.7.1 Proof of Theorem 4

The proof of the theorem relies on the following lemma.

Lemma 1 *For any infinitary formula F and any HT-interpretation $\langle I, J\rangle$,*

$$I \models F^J \text{ iff } \langle I, J\rangle \models F.$$

The lemma can be easily proved by strong induction on the rank of F.

Theorem 4 *An interpretation J is a stable model of a set $\mathcal{H}$ of infinitary formulas iff $\langle J, J\rangle$ is an equilibrium model of $\mathcal{H}$.*

Proof. It follows from the lemma that a total HT-interpretation $\langle J, J\rangle$ is an equilibrium model of $\mathcal{H}$ iff

- J satisfies all formulas from $\mathcal{H}$, and
- there is no proper subset I of J such that I satisfies the reducts F^J of all formulas F from $\mathcal{H}$.

These conditions express that J is a stable model of $\mathcal{H}$. □

6.7.2 Proof of Theorem 5

Theorem 5 *For any sets $\mathcal{H}_1$, $\mathcal{H}_2$ of infinitary formulas, the following conditions are equivalent:*

(i) $\mathcal{H}_1$ is strongly equivalent to $\mathcal{H}_2$,

(ii) for every set $\mathcal{H}$ of unary formulas, sets $\mathcal{H}_1 \cup \mathcal{H}$ and $\mathcal{H}_2 \cup \mathcal{H}$ have the same stable models;

(iii) sets $\mathcal{H}_1$ and $\mathcal{H}_2$ have the same HT-models.

Proof. Clearly, (i) implies (ii). To see that (iii) implies (i), observe that if sets $\mathcal{H}_1$ and $\mathcal{H}_2$ have the same HT-models then $\mathcal{H}_1 \cup \mathcal{H}$ and $\mathcal{H}_2 \cup \mathcal{H}$ have the same HT-models, and consequently have the same equilibrium models. It follows by Theorem 4 that $\mathcal{H}_1 \cup \mathcal{H}$ and $\mathcal{H}_2 \cup \mathcal{H}$ have the same stable models.

It remains to check that (ii) implies (iii). Suppose $\langle I, J\rangle$ is an HT-model of $\mathcal{H}_1$ but not an HT-model of $\mathcal{H}_2$. We will show how to find a set $\mathcal{H}$ of unary formulas such that $\langle J, J\rangle$ is an equilibrium model of one of the sets $\mathcal{H}_1 \cup \mathcal{H}, \mathcal{H}_2 \cup \mathcal{H}$ but not the other. It will follow that the interpretation J is a stable model of one but not the other.

Case 1: $\langle J, J\rangle$ is not an HT-model of $\mathcal{H}_2$. Since $\langle I, J\rangle$ is an HT-model of $\mathcal{H}_1$, it is easy to see that $\langle J, J\rangle$ must be an HT-model of $\mathcal{H}_1$ as well. Then we can take $\mathcal{H} = J$. Indeed, it is clear that $\langle J, J\rangle$ is an HT-model of $\mathcal{H}_1 \cup J$. Furthermore, for any I that is a proper subset of J, $\langle I, J\rangle$ cannot be an HT-model of $\mathcal{H}_1 \cup J$, so that $\langle J, J\rangle$ is an equilibrium model of $\mathcal{H}_1 \cup J$. On the other hand, since $\langle J, J\rangle$ is not a HT-model of $\mathcal{H}_2$, it cannot be an HT-model of $\mathcal{H}_2 \cup J$.

Case 2: $\langle J, J\rangle$ is an HT-model of $\mathcal{H}_2$. Let $\mathcal{H}$ be the set

$$I \cup \{p \to q \mid p, q \in J \setminus I\}.$$

Since $\langle J, J\rangle$ satisfies every formula in $\mathcal{H}$, it is an HT-model of $\mathcal{H}_2 \cup \mathcal{H}$. To see that it is an equilibrium model, consider any HT-model $\langle K, J\rangle$ of $\mathcal{H}_2 \cup \mathcal{H}$. Clearly, K must contain I. But it cannot be equal to I, since $\langle I, J\rangle$ is not an HT-model of $\mathcal{H}_2$. Thus $I \subset K \subset J$. Consider an atom p in $K \setminus I$ and an atom q in $J \setminus K$. For these atoms, $p \to q$ belongs to $\mathcal{H}$. But $\langle K, J\rangle$ does not satisfy this implication, contrary to the assumption that it is an HT-model of $\mathcal{H}_2 \cup \mathcal{H}$. We may conclude that $\langle J, J\rangle$ is an equilibrium model of $\mathcal{H}_2 \cup \mathcal{H}$. Finally, we check that $\langle J, J\rangle$ is not an equilibrium model of $\mathcal{H}_1 \cup \mathcal{H}$. Consider the HT-model $\langle I, J\rangle$ of $\mathcal{H}_1$. Clearly, it is an HT-model of I. Moreover, it satisfies each implication $p \to q$ in $\mathcal{H}$: $\langle I, J\rangle$ does not satisfy p because $p \notin I$, and J satisfies q because $q \in J$. We see that $\langle I, J\rangle$ satisfies all formulas in $\mathcal{H}$, so that it is an HT-model of $\mathcal{H}_1 \cup \mathcal{H}$. Furthermore, I is different from J since $\langle J, J\rangle$ is an HT-model of $\mathcal{H}_2$ and $\langle I, J\rangle$ is not. Consequently, I is a proper subset of J, and we may conclude that $\langle J, J\rangle$ is not an equilibrium model of $\mathcal{H}_1 \cup \mathcal{H}$. □

6.7.3 Proof of Theorem 6

The proof of completeness applies the idea of the proof of completeness for C^∞ from Section 6.3 to the 3-valued case. It uses the following construction, due to Cabalar and Ferraris (2007, Section 5): for any HT-interpretation $\langle I, J\rangle$, M_{IJ}

stands for

$$I \cup \{\neg\neg p \mid p \in J\} \cup \{\neg p \mid p \in \sigma \setminus J\} \cup \{p \to q \mid p, q \in J \setminus I\},$$

where σ is the signature. By $v_{IJ}(F)$ we denote the truth value of formula F with respect to $\langle I, J\rangle$ (see Section 6.1). We omit the subscripts I, J in M_{IJ} and $v_{IJ}(F)$ when it is clear which HT-interpretation we refer to.

Lemma 2 *For any infinitary formula F and HT-interpretation $\langle I, J\rangle$,*

(i) if $v(F) = \emptyset$ then $M^\wedge \Rightarrow \neg F$ is a theorem of HT$^\infty$;

(ii) if $v(F) = \{T\}$ then for every atom q in $J \setminus I$, $M^\wedge \Rightarrow F \leftrightarrow q$ is a theorem of HT$^\infty$;

(iii) if $v(F) = \{H, T\}$ then $M^\wedge \Rightarrow F$ is a theorem of HT$^\infty$.

Proof. We prove the lemma by strong induction on the rank of F. We assume the lemma holds for all formulas with rank less than n and show that it holds for a formula F of rank n. We consider cases corresponding to the different possible forms of F and truth values $v(F)$. Note that if $v(F)$ is $\{T\}$ then the set $J \setminus I$ is non-empty. Indeed, if $I = J$ then the truth value of any formula is either $\emptyset$ or $\{H, T\}$.

Case 1: F is an atom.

Case 1.1: $v(F) = \emptyset$. Then $F \in \sigma \setminus J$, and $\neg F \in M$.

Case 1.2: $v(F) = \{T\}$. Then $F \in J \setminus I$, and for every atom q in $J \setminus I$, the implications $F \to q$ and $q \to F$ are in M.

Case 1.3: $v(F) = \{H, T\}$. Then $F \in M$.

Case 2: F is of the form $\mathcal{H}^\wedge$. The induction hypothesis is then applicable to all formulas in $\mathcal{H}$.

Case 2.1: $v(F) = \emptyset$. Then there exists a formula G in $\mathcal{H}$ such that $v(G)$ is $\emptyset$. By the induction hypothesis, $M^\wedge \Rightarrow \neg G$ is a theorem of HT$^\infty$. From this we can derive $M^\wedge \Rightarrow \neg(\mathcal{H}^\wedge)$.

Case 2.2: $v(F) = \{T\}$. Let $\mathcal{H}_1$ be the set of all formulas in $\mathcal{H}$ with truth value $\{T\}$, and $\mathcal{H}_2$ be the set of all formulas in $\mathcal{H}$ with truth value $\{H, T\}$. It is clear that $\mathcal{H}_1 \cup \mathcal{H}_2 = \mathcal{H}$ and that $\mathcal{H}_1$ is non-empty. Consider an arbitrary element q of $J \setminus I$. By the induction hypothesis, $M^\wedge \Rightarrow G \leftrightarrow q$ is a theorem for every G in $\mathcal{H}_1$, and $M^\wedge \Rightarrow G$ is a theorem for every G in $\mathcal{H}_2$. From these we can derive $M^\wedge \Rightarrow \mathcal{H}_1^\wedge \leftrightarrow q$ and $M^\wedge \Rightarrow \mathcal{H}_2^\wedge$. Then we can derive $M^\wedge \Rightarrow \mathcal{H}^\wedge \leftrightarrow q$.

Case 2.3: $v(F) = \{H, T\}$. Then for each element G in $\mathcal{H}$, $v(G) = \{H, T\}$, and by the induction hypothesis $M^\wedge \Rightarrow G$ is a theorem. From these sequents we can derive $M^\wedge \Rightarrow \mathcal{H}^\wedge$.

Case 3: F is of the form $\mathcal{H}^\vee$. The induction hypothesis is then applicable to all formulas in $\mathcal{H}$.

Case 3.1: $v(F) = \emptyset$. Then for each element G in $\mathcal{H}$, $v(G) = \emptyset$, and by the induction hypothesis $M^\wedge \Rightarrow \neg G$ is a theorem. From these sequents we can derive $M^\wedge \Rightarrow \neg(\mathcal{H}^\vee)$.

Case 3.2: $v(F) = \{T\}$. Let $\mathcal{H}_1$ be the set of all formulas in $\mathcal{H}$ with truth value $\{T\}$, and $\mathcal{H}_2$ be the set of all formulas in $\mathcal{H}$ with truth value $\emptyset$. It is clear that

$\mathcal{H}_1 \cup \mathcal{H}_2 = \mathcal{H}$ and that $\mathcal{H}_1$ is non-empty. Consider an arbitrary element q of $J \setminus I$. By the induction hypothesis, $M^\wedge \Rightarrow G \leftrightarrow q$ is a theorem for every G in $\mathcal{H}_1$, and $M^\wedge \Rightarrow \neg G$ is a theorem for every G in $\mathcal{H}_2$. From these we can derive $M^\wedge \Rightarrow \mathcal{H}_1^\vee \leftrightarrow q$ and $M^\wedge \Rightarrow \neg(\mathcal{H}_2^\vee)$. Then we can derive $M^\wedge \Rightarrow \mathcal{H}^\vee \leftrightarrow q$.

Case 3.3: $v(F) = \{H, T\}$. Then there exists a formula G in $\mathcal{H}$ such that $v(G)$ is $\{H, T\}$. By the induction hypothesis, $M^\wedge \Rightarrow G$ is a theorem. From this we can derive $M^\wedge \Rightarrow \mathcal{H}^\vee$.

Case 4: F is of the form $F_1 \rightarrow F_2$. The induction hypothesis is then applicable to F_1 and F_2.

Case 4.1: $v(F) = \emptyset$. Then $v(F_1)$ is non-empty and $v(F_2)$ is empty.

Case 4.1.1: $v(F_1) = \{T\}$. By the induction hypothesis, $M^\wedge \Rightarrow \neg F_2$ is a theorem, as is $M^\wedge \Rightarrow F_1 \leftrightarrow q$ for any q in $J \setminus I$. Consider an atom q in $J \setminus I$. By the construction of M, we know that $\neg\neg q$ is an element of M. From the sequents $M^\wedge \Rightarrow F_1 \leftrightarrow q$, $M^\wedge \Rightarrow \neg F_2$, and $M^\wedge \Rightarrow \neg\neg q$, we can derive $M^\wedge \Rightarrow \neg(F_1 \rightarrow F_2)$.

Case 4.1.2: $v(F_1) = \{H, T\}$. By the induction hypothesis, both $M^\wedge \Rightarrow F_1$ and $M^\wedge \Rightarrow \neg F_2$ are theorems. From these sequents we can derive $M^\wedge \Rightarrow \neg(F_1 \rightarrow F_2)$.

Case 4.2: $v(F) = \{T\}$. Then $v(F_1) = \{H, T\}$ and $v(F_2) = \{T\}$. By the induction hypothesis, $M^\wedge \Rightarrow F_2 \leftrightarrow q$ is a theorem for any $q \in J \setminus I$, and $M^\wedge \Rightarrow F_1$ is a theorem as well. From these two sequents we can derive $M^\wedge \Rightarrow (F_1 \rightarrow F_2) \leftrightarrow q$.

Case 4.3: $v(F) = \{H, T\}$.

Case 4.3.1: $v(F_1) = \emptyset$. Then by the induction hypothesis $M^\wedge \Rightarrow \neg F_1$ is a theorem. From this we can derive $M^\wedge \Rightarrow F_1 \rightarrow F_2$.

Case 4.3.2: $v(F_2) = \{H, T\}$. Then by the induction hypothesis $M^{\wedge} \Rightarrow F_2$ is a theorem. From this we can derive $M^{\wedge} \Rightarrow F_1 \to F_2$.

Case 4.3.3: $v(F_1) \neq \emptyset$ and $v(F_2) \neq \{H, T\}$. Since $v(F)$ is $\{H, T\}$, $v(F_1)$ is different from $\{H, T\}$ and therefore must be equal to $\{T\}$. It follows that $v(F_2)$ is different from $\emptyset$, and therefore must be $\{T\}$ also. Consider an element q in $J \setminus I$. By the induction hypothesis, both $M^{\wedge} \Rightarrow F_1 \leftrightarrow q$ and $M^{\wedge} \Rightarrow F_2 \leftrightarrow q$ are theorems. From these two sequents we can derive $M^{\wedge} \Rightarrow F_1 \to F_2$. $\square$

Note that in the proof of the lemma we did not refer to axiom schemas (6.2) and (6.3); the assertion of the lemma would hold even if those axioms were removed from HT$^{\infty}$.

Lemma 3 *The disjunction of the formulas $M^{\wedge}_{IJ}$ over all HT-interpretations $\langle I, J\rangle$ is a theorem of HT$^{\infty}$.*

Proof. Let Q stand for the set of disjunctions including instances of the Hosoi axiom (3.10)

$$p \vee (p \to q) \vee \neg q, \tag{6.17}$$

for all p, q from σ, and instances of the weak law of excluded the middle law (see Section 6.6)

$$\neg p \vee \neg\neg p \tag{6.18}$$

for all p from σ. Let $(\mathcal{H}_D)_{D \in Q}$ be the following family of sets:

$$\mathcal{H}_D = \{p, p \to q, \neg q\} \qquad \text{if } D = p \vee (p \to q) \vee \neg q;$$
$$\mathcal{H}_D = \{\neg p, \neg\neg p\} \qquad \text{if } D = \neg p \vee \neg\neg p.$$

Then the formula

$$\bigwedge_{D\in Q}\bigvee_{S\in\mathcal{H}_D} S \to \bigvee_{(S_D)_{D\in Q}}\bigwedge_{D\in Q} S_D,$$

(where the disjunction in the consequent extends over all elements $(S_D)_{D\in Q}$ of the Cartesian product of the family $(\mathcal{H}_D)_{D\in Q}$) is an instance of axiom schema (6.3). Since the antecedent of this implication is the conjunction of all formulas in Q, it is a theorem of HT$^\infty$. It follows that the consequent is a theorem as well. To complete the proof it is sufficient to show that for every disjunctive term

$$\bigwedge_{D\in Q} S_D \tag{6.19}$$

of the consequent there exists an HT-interpretation $\langle I, J\rangle$ such that the sequent

$$\bigwedge_{D\in Q} S_D \Rightarrow M^{\wedge}_{IJ} \tag{6.20}$$

is a theorem.

Consider one of the conjunctions (6.19), and let C be set of its conjunctive terms. The elements of C are formulas of the forms

$$p,\ \neg p,\ \neg\neg p,\ p \to q.$$

If C contains both a formula and its negation then (6.20) is a theorem for every $\langle I, J\rangle$. Otherwise, let I denote the set of all atoms in C, and J denote the set of all atoms p such that $\neg\neg p$ is in C. Let us check that $I \subseteq J$. Assume $p \in I$ so that $p \in C$. Since C is consistent, it does not contain $\neg p$, and since it contains a term from each disjunction (6.18), it contains $\neg\neg p$. So $\langle I, J\rangle$ is an HT-interpretation.

We will show that every formula from M_{IJ} belongs to C. By the choice of I, $I \subseteq C$. By the choice of J, $\{\neg\neg p \mid p \in J\} \subseteq C$. Consequently,

$$\{\neg p \mid p \in \sigma \setminus J\} \subseteq C,$$

because C contains one term from each disjunction (6.18). Finally, we need to check that $\{p \rightarrow q \mid p, q \in J \setminus I\} \subseteq C$. Consider a pair of atoms p, q that occur in J but not in I. By the choice of I, p is not in C, and by the choice of J, $\neg q$ is not in C. Since C contains one term from each of the disjunctions (6.17) and contains neither p nor $\neg q$, C must contain $p \rightarrow q$. $\square$

Theorem 6 *An infinitary formula is a theorem of HT$^\infty$ iff it is satisfied by all HT-interpretations.*

Proof. Proving soundness is straightforward. To prove completeness, let F be an infinitary formula over signature σ that is satisfied by all HT-interpretations of σ. By Lemma 2(iii), $M_{IJ} \Rightarrow F$ is a theorem of HT$^\infty$ for all HT-interpretations $\langle I, J\rangle$. By Lemma 3, it follows that F is a theorem also. $\square$

Chapter 7

Finite Proofs for Infinitary Formulas

In Chapter 6 we avoided a discussion of proofs in the infinitary deductive systems C^∞ and HT^∞, and defined a theorem as an element of the smallest set of formulas satisfying some closure conditions. We chose to avoid the notion of a proof because, in general, proofs in those systems are infinite objects. However, part of the appeal of a formalized deductive system is that finite proof objects can be used to establish the validity of assertions about infinite domains. In this chapter, we show how it is often possible to establish the validity of an infinitary formula using a proof in a finite deductive system.[1] Our method is based on the definition of an instance, given in Section 6.5. Proposition 2 in that section (in conjunction with the soundness of HT^∞) shows that substituting infinitary formulas for atoms in an intuitionistically provable formula results in a formula that is satisfied by all HT-interpretations. For example, the formula

$$(p \vee q) \wedge r \leftrightarrow (p \wedge r) \vee (q \wedge r) \tag{7.1}$$

[1]Much of the material in this chapter was originally published in the following publication: Amelia Harrison, Vladimir Lifschitz, and Julian Michael. Proving infinitary formulas. Theory and Practice of Logic Programming, 16(5-6):787799, 2016.

My own contributions to that paper included collaboration on efinitions as well as statements and proofs of theorems.

is intuitionistically provable; it follows that for any infinitary formulas F, G, H, the infinitary formula

$$(F \lor G) \land H \leftrightarrow (F \land H) \lor (G \land H) \tag{7.2}$$

is satisfied by all HT-interpretations. We can think of a proof of (7.1) as a proof of (7.2) with respect to the substitution that maps p to F, q to G, and r to H. In a similar way, we can talk about proofs of the formula

$$\left(\bigvee_{\alpha \in A} F_\alpha\right) \land G \leftrightarrow \bigvee_{\alpha \in A} (F_\alpha \land G) \tag{7.3}$$

for any non-empty finite family $(F_\alpha)_{\alpha \in A}$ of infinitary formulas and any infinitary formula G.

In this chapter, we show how the idea of an infinitary instance of a finite formula can be used in a different setting. We define instances for first-order formulas, and that allows us, for example, to talk about finite proofs of (7.3) even when A is infinite. Consider the signature that has (symbols for) the elements of A as object constants, the unary predicate constant P, and the propositional constant Q. We will see that (7.3) is the instance of the first-order formula

$$\exists x P(x) \land Q \leftrightarrow \exists x (P(x) \land Q) \tag{7.4}$$

corresponding to the substitution that maps $P(\alpha)$ to F_α, and Q to G. This formula is intuitionistically provable[2], and we will see that it follows that (7.3) is satisfied by all HT-interpretations.

7.1 Substitutions and Instances

By Σ we denote an arbitrary signature in the sense of first-order logic that contains at least one object constant. The signature may include propositional constants (viewed as predicate constants of arity 0). Object constants will be viewed as function constants of arity 0. In first-order formulas over Σ, we treat the binary connectives $\wedge$, $\vee$, and $\rightarrow$ and the 0-place connective $\bot$ as primitive; $\top$, $\neg$, and $\leftrightarrow$ are the usual abbreviations from propositional logic.

A *substitution* is a function ψ that maps each closed atomic formula over Σ to an infinitary formula over σ, such that the range of ψ is bounded. A substitution ψ is extended from closed atomic formulas to arbitrary closed first-order formulas over Σ as follows:

- $\psi\bot$ is $\bot$;
- $\psi(\alpha_1 = \alpha_2)$, where α_1, α_2 are ground terms, is $\top$ if α_1 is α_2, and $\bot$ otherwise;
- $\psi(F \odot G)$, where $\odot$ is a binary connective, is $\psi F \odot \psi G$;
- $\psi\forall vF$ is $\bigwedge_\alpha \psi F^v_\alpha$, where α ranges over the ground terms of Σ;
- $\psi\exists vF$ is $\bigvee_\alpha \psi F^v_\alpha$, where α ranges over the ground terms of Σ.

[2]Formalizations of first-order intuitionistic logic can be found in Mints's book (2000, Chapters 13 and 15).

The formula ψF will be called the *instance of F with respect to ψ*.

This notion of a substitution is similar to the notion of a substitution presented in Section 6.5, but in that section substitutions mapped infinitary formulas to infinitary formulas, while here they map first-order formulas to infinitary formulas.

For example, if Σ includes the elements of A as object constants, but no other function constants, then (7.3) is the instance of (7.4) with respect to the substitution ψ defined as follows:

$$\psi P(\alpha) = F_\alpha,$$
$$\psi Q = G.$$

If the function constants of Σ are the object constant a and the unary function constant s, then any infinite conjunction of the form

$$\bigwedge_{i\geq 0}(F_i \rightarrow G_i),$$

where F_i, G_i are infinitary formulas, is the instance of the first-order formula

$$\forall x(P(x) \rightarrow Q(x))$$

with respect to the substitution ψ defined as follows:

$$\psi(P(s^i(a))) = F_i,$$

$$\psi(Q(s^i(a))) = G_i.$$

7.2 Herbrand Logic of Here-and-There

The theorem stated below shows that if a closed first-order formula is intuitionistically provable then all its instances are satisfied by all HT-interpretations. The theorem is actually more general because it refers to a deductive system that includes, in addition to the axioms and inference rules of first-order intuitionistic logic with equality, some additional axioms. We can add, first of all, the axiom schema (3.10); the axiom schema

$$\exists x(F \to \forall x F) \tag{7.5}$$

(Lifschitz et al., 2007); and the "decidable equality" axiom

$$x = y \lor x \neq y. \tag{7.6}$$

We include also the axioms of the Clark Equality Theory (Clark, 1978):

$$f(x_1, \dots, x_n) \neq g(y_1, \dots, y_m) \tag{7.7}$$

for all pairs of distinct function constants f, g from Σ;

$$f(x_1, \dots, x_n) = f(y_1, \dots, y_n) \to (x_1 = y_1 \land \cdots \land x_n = y_n) \tag{7.8}$$

for all function constants f from Σ of arity greater than 0; and

$$t(x) \neq x \tag{7.9}$$

for all terms $t(x)$ that contain x but are different from x.

The deductive system obtained from first-order intuitionistic logic with equality by adding axioms (7.5)–(7.9) and (3.10) is denoted by $\mathbf{HHT}$ ("Herbrand logic of here-and-there").

Theorem 7 (Harrison et al., 2016) *If a closed first-order formula F is provable in* $\mathbf{HHT}$ *then any instance of F is satisfied by all HT-interpretations.*

As an example application of this theorem, recall formula (6.11), the infinitary counterpart of the non-intuitionistically provable De Morgan's law. In Section 6.6, we showed how this formula might be proved in HT$^\infty$. However, our explanation of that derivation required more than a page of text and the formal derivation itself is an infinite object. Using the theorem above (in conjunction with the completeness of HT$^\infty$), we can justify the provability of (6.11) with non-empty $\mathcal{H}$ on the basis of the fact that it is an instance of the first-order formula

$$\neg\forall x P(x) \to \exists x \neg P(x),$$

and this formula is provable in $\mathbf{HHT}$. (Use (7.5) with $P(x)$ as F.)

As discussed above, the fact that formula (7.3) is satisfied by all HT-interpretations follows from the provability of (7.4) in first-order intuitionistic logic. Consider the formula dual to (7.3):

$$\left(\bigwedge_{\alpha\in A} F_\alpha\right) \vee G \leftrightarrow \bigwedge_{\alpha\in A}(F_\alpha \vee G).$$

(As before, $(F_\alpha)_{\alpha\in A}$ is a non-empty family of infinitary formulas, and G is an infinitary formula.) The fact that this formula is satisfied by all HT-interpretations can be derived from Theorem 7 in a similar way, with the corresponding first-order formula

$$\forall x P(x) \vee Q \leftrightarrow \forall x(P(x) \vee Q).$$

The proof of the right-to-left direction uses (7.5), again with $P(x)$ as F.

Any formula of the form

$$\left(\left(\bigvee_{\alpha\in A} F_\alpha\right) \to G\right) \leftrightarrow \bigwedge_{\alpha\in A}(F_\alpha \to G)$$

with non-empty A is satisfied by all HT-interpretations because it is an instance of the intuitionistically provable formula

$$(\exists x P(x) \to Q) \leftrightarrow \forall x(P(x) \to Q).$$

Any formula of the form

$$\bigvee_{\alpha\in A}\left(F_\alpha \to \bigwedge_{\beta\in A} F_\beta\right),$$

where A is non-empty, is satisfied by all HT-interpretations because it is an instance of the axiom schema (7.5).

7.3 Including Restrictors

In this section we give a more general definition of an instance, which may allow our method to be applicable to more formulas.

We assume here that some unary predicate symbols of the signature Σ may be designated as *restrictors*. The role of restrictors is somewhat similar to the role of sorts in a many-sorted signature. A *generalized variable* is defined as either a variable or an expression of the form

$$(x_1\!:\!R_1, \ldots, x_n\!:\!R_n) \tag{7.10}$$

where $x_1, \ldots, x_n$ ($n \geq 1$) are distinct variables, and $R_1, \ldots, R_n$ are restrictors. *Formulas with restrictors* are defined recursively in the same way as first-order formulas over Σ except that a quantifier may be followed by a generalized variable. For instance, if Σ includes the unary predicate constants P and R, and the latter is a restrictor, then

$$\forall x P(x) \rightarrow \forall (x\!:\!R) P(x) \tag{7.11}$$

is a formula with restrictors.

Generalized variables (7.10) can be eliminated from a formula with restrictors by replacing subformulas of the form

$$\forall (x_1\!:\!R_1, \ldots, x_n\!:\!R_n) F$$

with

$$\forall x_1 \dots x_n (R_1(x_1) \wedge \dots \wedge R_n(x_n) \to F),$$

and subformulas of the form

$$\exists (x_1 \!:\! R_1, \dots, x_n \!:\! R_n) F$$

with

$$\exists x_1 \dots x_n (R_1(x_1) \wedge \dots \wedge R_n(x_n) \wedge F).$$

To prove a formula with restrictors in a deductive system means to prove the first-order formula obtained by this transformation. For instance, we can say that formula (7.11) is provable in the intuitionistic predicate calculus because the formula

$$\forall x P(x) \to \forall x (R(x) \to P(x))$$

is provable in that deductive system. Satisfaction of closed formulas with restrictors is defined in a similar way.

In the presence of restrictors, a *substitution* is defined as a function ψ that maps each closed atomic formula F over Σ to one of the formulas $\top$, $\bot$, if F begins with a restrictor, and to an infinitary formula over σ otherwise, such that the range of ψ is bounded. A substitution ψ is extended to closed first-order formulas over Σ with restrictors in the same way as for first-order formulas as in Section 7.1, with the additional clauses:

- $\psi\,\forall(x_1{:}R_1,\dots,x_n{:}R_n)F$ is

$$\bigwedge_{\alpha_1,\dots,\alpha_n\;:\;\psi R_1(\alpha_1)=\cdots=\psi R_n(\alpha_n)=\top} \psi F^{x_1\cdots x_n}_{\alpha_1\cdots\alpha_n},$$

- $\psi\,\exists(x_1{:}R_1,\dots,x_n{:}R_n)F$ is

$$\bigvee_{\alpha_1,\dots,\alpha_n\;:\;\psi R_1(\alpha_1)=\cdots=\psi R_n(\alpha_n)=\top} \psi F^{x_1\cdots x_n}_{\alpha_i\cdots\alpha_n}.$$

Theorem 8 (Harrison et al., 2016) *If a closed first-order formula F with restrictors is provable in* $\mathbf{HHT}$ *then any instance of F is satisfied by all HT-interpretations.*

For example, consider a formula of the form

$$\bigwedge_{\alpha\in A} F_\alpha \to \bigwedge_{\alpha\in B} F_\alpha, \tag{7.12}$$

where B is a proper subset of A. It is an instance of (7.11): take the elements of A to be the only function constants of Σ, and define the substitution ψ by the conditions

$$\begin{aligned}&\psi R(\alpha) = \top \text{ iff } \alpha \in B,\\ &\psi P(\alpha) = F_\alpha.\end{aligned}$$

Since (7.11) is intuitionistically provable, (7.12) is satisfied by all HT-interpretations.

Similarly, any formula of the form

$$\bigvee_{\alpha\in A} F_\alpha \wedge \bigvee_{\beta\in B} G_\beta \leftrightarrow \bigvee_{(\alpha,\beta)\in A\times B} (F_\alpha \wedge G_\beta) \tag{7.13}$$

is an instance of the formula

$$\exists(x\!:\!R_1)P(x) \wedge \exists(y\!:\!R_2)Q(y) \leftrightarrow \exists(x\!:\!R_1, y\!:\!R_2)(P(x) \wedge Q(y)). \tag{7.14}$$

Indeed, we can include the elements of $A \cup B$ among the object constants of σ and choose ψ so that

$$\begin{aligned}
&\psi R_1(\alpha) = \top \text{ iff } \alpha \in A,\\
&\psi R_2(\alpha) = \top \text{ iff } \alpha \in B,\\
&\psi P(\alpha) = F_\alpha \text{ for all } \alpha \in A,\\
&\psi Q(\alpha) = G_\alpha \text{ for all } \alpha \in B.
\end{aligned}$$

Since (7.14) is intuitionistically provable, (7.13) is satisfied by all HT-interpretations.

References to Theorem 8 can be replaced in some cases by references to Theorem 7 from Section 7.2 at the cost of using more complicated substitutions. For instance, the claim that formula (7.12) is satisfied by all HT-interpretations, under the additional assumption that B is non-empty, can be justified as follows. Take Σ to be the signature consisting of the elements of A as object constants, the unary function constant f, and the unary predicate constant P. Choose an element α_0 of B. Then (7.12) is the instance of the formula

$$\forall x P(x) \rightarrow \forall x P(f(x))$$

with respect to the substitution ψ defined by the condition: for all object constants α,

$$\begin{aligned}
&\psi P(\alpha) = F_\alpha, \\
&\psi P(f^i(\alpha)) = F_\alpha && \text{if } i \geq 1 \text{ and } \alpha \in B, \\
&\psi P(f^i(\alpha)) = F_{\alpha_0} && \text{if } i \geq 1 \text{ and } \alpha \notin B.
\end{aligned}$$

7.4 Including Second-Order Axioms

We define now an extension $\mathbf{HHT}^2$ of $\mathbf{HHT}$ where predicate and function variables of arbitrary arity are included in the language, as in Section 1.2.3 of the handbook chapter by Lifschitz et al. 2008. The set of axioms and inference rules of $\mathbf{HHT}$ is extended by adding the usual postulates for second-order quantifiers, the axiom schema of comprehension

$$\exists p \forall x_1 \dots x_n (p(x_1, \dots, x_n) \leftrightarrow F) \tag{7.15}$$

$(n \geq 0)$, where the predicate variable p is not free in F, and the axiom of choice

$$\begin{aligned}
&\forall x_1 \dots x_n \exists x_{n+1}\, p(x_1, \dots, x_{n+1}) \rightarrow \\
&\qquad \exists f \forall x_1 \dots x_n (p(x_1, \dots, x_n, f(x_1, \dots, x_n)))
\end{aligned} \tag{7.16}$$

$(n > 0)$. The main theorem can be extended as follows.

Theorem 9 (Harrison et al., 2016) *If a closed first-order formula F (possibly with restrictors) is provable in* $\mathbf{HHT^2}$ *then any instance of F is satisfied by all HT-*

interpretations.

In the special case when the signature Σ contains finitely many function constants, by DCA we denote the domain closure axiom:

$$\forall p \left(\bigwedge C_f(p) \; \rightarrow \; \forall x \, p(x) \right)$$

where the conjunction extends over all function constants f from Σ, and $C_f(p)$ ("set p is closed under f") stands for the formula

$$\forall x_1 \dots x_n (p(x_1) \wedge \dots \wedge p(x_n) \rightarrow p(f(x_1, \dots, x_n)).$$

(In the presence of DCA, axioms (7.6) and (7.9) become redundant.) For instance, if Σ contains an object constant a and unary function constant s and no other function constants, then DCA turns into the second-order axiom of induction

$$\forall p \, (p(a) \wedge \forall x \, (p(x) \rightarrow p \, (s(x))) \rightarrow \forall x \, p(x)) \,, \tag{7.17}$$

and $\mathbf{HHT}^2+$ DCA becomes an extension of second-order intuitionistic arithmetic.

In the following theorem, the signature Σ is assumed to contain finitely many function constants.

Theorem 10 (Harrison et al., 2016) *If a closed first-order formula F (possibly with restrictors) is provable in $\mathbf{HHT}^2+$ DCA then any instance of F is satisfied by all HT-interpretations.*

Note that both Theorems 9 and 10 refer to first-order formulas provable using second-order axioms. The notion of a substitution is not defined here for second-order formulas.

As an example, we show that any equivalence of the form

$$\left(F_0 \wedge \bigwedge_{i\geq 0}(F_i \rightarrow F_{i+1})\right) \leftrightarrow \bigwedge_{i\geq 0} F_i$$

is satisfied by all HT-interpretations. Indeed, with the appropriate choice of the signature Σ, it is an instance of the formula

$$P(a) \wedge \forall x(P(x) \rightarrow P(s(x))) \leftrightarrow \forall x P(x).$$

This formula is provable in $\mathbf{HHT}^2$+ DCA. (The implication left-to-right is given by axiom (7.17).)

7.5 Proofs

7.5.1 Proof of Theorem 9

Theorem 9 *If a closed first-order formula F (possibly with restrictors) is provable in $\mathbf{HHT^2}$ then any instance of F is satisfied by all HT-interpretations.*

The proof of the theorem makes use of "Herbrand HT-interpretations"—Kripke models with two worlds and with the universe consisting of all ground terms of the signature Σ. We will see that all theorems of $\mathbf{HHT}$ (and its extensions discussed in the previous section) are satisfied by all Herbrand HT-interpretations.

On the other hand, for any substitution ψ and any HT-interpretation I of σ, we can find an Herbrand HT-interpretation J such that J satisfies a closed first-order formula F if and only if I satisfies ψF.[3] Theorem 9 directly follows from these two facts. In fact, each of the theorems in this chapter can be established in the same way.

An *Herbrand HT-interpretation* of a first-order signature Σ is a pair $\langle J^h, J^t \rangle$ of subsets of the Herbrand base of Σ (that is, the set of all ground atomic formulas over Σ that do not include equality) such that $J^h \subseteq J^t$. By $\mathcal{U}$ we denote the Herbrand universe of Σ, that is, the set of all ground terms over Σ.

For each function $\mathfrak{f}$ of arity $n > 0$ that maps from $\mathcal{U}^n$ to $\mathcal{U}$ we introduce a function constant $\mathfrak{f}^*$ of arity n, called the *function name* of $\mathfrak{f}$. For each pair $\mathfrak{p} = (\mathfrak{p}_h, \mathfrak{p}_t)$ of subsets of $\mathcal{U}^n$ such that $\mathfrak{p}_h \subseteq \mathfrak{p}_t$, we introduce an n-ary predicate constant $\mathfrak{p}^*$, called the *predicate name* of $(\mathfrak{p}_h, \mathfrak{p}_t)$. By Σ^* we denote the signature obtained by adding all function and predicate names to Σ, and by $\mathcal{U}^*$ we denote the Herbrand universe of Σ^*. Then for each term $\alpha \in \mathcal{U}^*$, we define the term $\widehat{\alpha} \in \mathcal{U}$ recursively as follows:

- if α is an object constant from $\mathcal{U}$ then $\widehat{\alpha}$ is α;
- if α is of the form $f(\alpha_1, \dots, \alpha_n)$ where f is a function constant from Σ, then $\widehat{\alpha}$ is $f(\widehat{\alpha_1}, \dots, \widehat{\alpha_n})$;
- if α is of the form $\mathfrak{f}^*(\alpha_1, \dots, \alpha_n)$ where $\mathfrak{f}^*$ is a function name, then $\widehat{\alpha}$ is the element of $\mathcal{U}$ obtained by applying $\mathfrak{f}$ to $\langle \widehat{\alpha_1}, \dots, \widehat{\alpha_n} \rangle$.

[3]Here, we use the single letter I to refer to an HT-interpretation rather than pair notation because we have no need to refer to the individual elements of the pair.

The satisfaction relation between an Herbrand HT-interpretation $J = \langle J^h, J^t \rangle$, a world w, and a closed second-order formula F over Σ is defined recursively, as follows:

1. $J, w \not\models \bot$.
2. $J, w \models \alpha_1 = \alpha_2$ if $\widehat{\alpha_1}$ is $\widehat{\alpha_2}$.
3. $J, w \models P(\alpha_1, \dots, \alpha_n)$ if $P(\widehat{\alpha_1}, \dots, \widehat{\alpha_n}) \in J^w$.
4. $J, w \models \mathfrak{p}^*(\alpha_1, \dots, \alpha_n)$ if $\langle\widehat{\alpha_1}, \dots, \widehat{\alpha_n}\rangle \in \mathfrak{p}_w$.
5. $J, w \models F \wedge G$ if $J, w \models F$ and $J, w \models G$; similarly for $\vee$.
6. $J, w \models F \rightarrow G$ if for every world w' such that $w \leq w'$, $J, w' \not\models F$ or $J, w' \models G$.
7. $J, w \models \forall v F$, where v is an object variable, if for each ground term α over Σ, $J, w \models F^v_\alpha$; similarly for $\exists$.
8. $J, w \models \forall v F$, where v is a function variable, if for each function name $\mathfrak{f}^*$ of the same arity as v, $J, w \models F^v_{\mathfrak{f}^*}$; similarly for $\exists$.
9. $J, w \models \forall v F$, where v is a predicate variable, if for each predicate name $\mathfrak{p}^*$ of the same arity as v, $J, w \models F^v_{\mathfrak{p}^*}$; similarly for $\exists$.

A closed second-order formula F over Σ^* is *HHT-valid* if $J, h \models F$ for every Herbrand HT-interpretation J.

The following lemma establishes the soundness of system $\mathbf{HHT}^2$ as well as $\mathbf{HHT}^2$+ DCA.

Lemma 4 *(a) If a second-order formula F over Σ^* is provable in* $\mathbf{HHT}^2$ *then the universal closure of F is HHT-valid.*

(b) For any first-order signature Σ containing finitely many function constants, if a second-order formula F over Σ^* is provable in $\mathbf{HHT}^2$+ DCA then the universal closure of F is HHT-valid.

The lemma is proved by induction on the derivation of F.

Lemma 5 *Let I be an HT-interpretation of a propositional signature σ, ψ be a substitution from a first-order signature Σ (possibly containing restrictors) to σ, and J be the Herbrand HT-interpretation defined by the condition: for every world w*

$$J, w \models P(\alpha_1, \ldots, \alpha_n) \text{ iff } I, w \models \psi P(\alpha_1, \ldots \alpha_n).$$

Then for any closed first-order formula F (possibly with restrictors)

$$J, w \models F \text{ iff } I, w \models \psi F.$$

Proof. This lemma is proved by strong induction on the total number of connectives and quantifiers in F. If F is atomic, then the assertion of the lemma is immediate from the definition of J. Here are two of the other cases.

Case $\forall vF$:

$J, w \models \forall vF$
iff for each ground term α, $J, w \models F^v_\alpha$
iff for each ground term α, $I, w \models \psi F^v_\alpha$
iff $I, w \models \bigwedge_\alpha \psi F^v_\alpha$
iff $I, w \models \psi \left(\bigwedge_\alpha F^v_\alpha\right)$.

Case $\forall(x_1\!:\!R_1,\dots,x_n\!:\!R_n)F$: We need to show that

$$J, w \models \forall(x_1\!:\!R_1,\dots,x_n\!:\!R_n)F$$

iff

$$I, w \models \bigwedge_{\alpha_1,\dots,\alpha_n:\ \psi R_1(\alpha_1)=\cdots=\psi R_n(\alpha_n)=\top} \psi F^{x_1,\cdots,x_n}_{\alpha_1,\cdots,\alpha_n}. \tag{7.18}$$

Indeed,

$J, w \models \forall(x_1\!:\!R_1,\dots,x_n\!:\!R_n)F$

iff $J, w \models \forall x_1,\dots,x_n(R_1(x_1)\wedge\cdots\wedge R_n(x_n)\rightarrow F)$

iff $J, w' \models F^{x_1,\cdots,x_n}_{\alpha_1,\cdots,\alpha_n}$ in every world $w' \geq w$ and for each tuple of ground terms $\alpha_1,\dots,\alpha_n$ such that $J, w' \models R_1(\alpha_1)\wedge\cdots\wedge R_n(\alpha_n)$

iff $I, w' \models \psi F^{x_1,\cdots,x_n}_{\alpha_1,\cdots,\alpha_n}$ in every world $w' \geq w$ and for each tuple of ground terms $\alpha_1,\dots,\alpha_n$ such that $I, w' \models \psi R_1(\alpha_1)\wedge\cdots\wedge \psi R_n(\alpha_n)$

iff $I, w' \models \psi F^{x_1,\cdots,x_n}_{\alpha_1,\cdots,\alpha_n}$ in every world $w' \geq w$ and for each tuple of ground terms $\alpha_1,\dots,\alpha_n$ such that $\psi R_1(\alpha_1)=\cdots=\psi R_n(\alpha_n)=\top$

iff in every world $w' \geq w$,
$I, w' \models \bigwedge_{\alpha_1,\dots,\alpha_n:\ \psi R_1(\alpha_1)=\cdots=\psi R_n(\alpha_n)=\top} \psi F^{x_1,\cdots,x_n}_{\alpha_1,\cdots,\alpha_n}$.

The condition above is equivalent to (7.18) by the monotonicity property of the satisfaction relation in the logic of here-and-there. □

Theorem 9 is immediate from the two lemmas stated above.

Chapter 8

Simplifying Translations of Aggregates

The result of applying τ to an aggregate literal is often an unweildy formula. However, depending on the particular syntactic form of the literal we may be able to simplify its translation.[1] As we show in Chapter 10, these simplifications are useful for reasoning about the aggregate literals occurring in Table 1.1.

8.1 Monotone and Anti-Monotone Aggregate Expressions

The translations of "monotone" and "antimonotone" aggregate literals (Ferraris, 2005) can be simplified using strongly equivalent transformations.

Recall that a set Δ of precomputed terms justifies a closed aggregate atom E with respect to a precomputed term t if the relation $\prec$ holds between $\widehat{\alpha}[\Delta]$ and t (see Section 5.4). A closed aggregate atom E is *monotone* if for any set Δ and term t such that Δ justifies E with respect to t, all supersets of Δ also justify E with respect to t. Similarly, E is *anti-monotone* if for any set Δ and term t such that Δ justifies E with respect to t, all subsets of Δ also justify E with respect to t. The monotonicity or anti-monotonicity of an aggregate atom (4.7) can be sometimes established simply on the basis of its aggregate name α and its relation symbol $\prec$.

[1]Some of the material in this chapter was originally published in the following publication: Martin Gebser, Amelia Harrison, Roland Kaminski, Vladimir Lifschitz, and Torsten Schaub. Abstract Gringo. Theory and Practice of Logic Programming, 15:449 463, 2015.

My own contributions to that paper included collaboration on the statements and proofs presented here.

If α is one of the symbols *count*, *sum+*, *max*, then (4.7) is monotone when $\prec$ is $>$ or $\geq$, and anti-monotone when $\prec$ is $<$ or $\leq$. It is the other way around if α is *min*.

Recall that $\tau_t E$ for a closed aggregate atom E is defined as the conjunction of the implications (5.2) over all sets Δ that do not justify E with respect to t. The two theorems stated below show that the antecedent in (5.2) can be dropped if E is monotone, and the consequent can be dropped if E is anti-monotone. The theorems are similar to the properties of monotone and anti-monotone ground aggregates stated in (Ferraris, 2005).

Theorem 11 *If a closed aggregate atom E is monotone, then for any term t, $\tau_t E$ is strongly equivalent to*

$$\bigwedge_{\Delta} \bigvee_{\mathbf{r} \in A \setminus \Delta} \tau(\mathbf{L}^{\mathbf{x}}_{\mathbf{r}}), \tag{8.1}$$

where A is the set of all precomputed terms, and the conjunction extends over all subsets Δ of A that do not justify E with respect to t.

Theorem 12 *If a closed aggregate atom E is anti-monotone, then for any term t, $\tau_t E$ is strongly equivalent to*

$$\bigwedge_{\Delta} \neg \bigwedge_{\mathbf{r} \in \Delta} \tau(\mathbf{L}^{\mathbf{x}}_{\mathbf{r}}), \tag{8.2}$$

where A is the set of all precomputed terms, and the conjunction extends over all subsets Δ of A that do not justify E with respect to t.

The complexity of the definition of τ_t applied to an aggregate atom (see Section 5.4) is the price we pay for a fully general definition that covers aggregate atoms that are neither monotone nor antimonotone. (Aggregate atoms involving $=$ and $\neq$, for example, are neither monotone nor antimonotone.) If we were willing to forgo this general coverage, we could reformulate the complex definition of τ_t using Theorems 11 and 12.

8.2 Eliminating Equality from Aggregate Atoms

If the relation $\prec$ in an aggregate atom (4.7) is $=$ then the following theorem can be useful, especially in combination with the facts reviewed in the previous section.

Theorem 13 *Let E be a closed aggregate atom of the form*

$$\alpha\{\mathbf{t} : \mathbf{L}\} = s.$$

Let $E_{\leq}$ be

$$\alpha\{\mathbf{t} : \mathbf{L}\} \leq s$$

and $E_{\geq}$ be

$$\alpha\{\mathbf{t} : \mathbf{L}\} \geq s.$$

Then for any precomputed term t, $\tau_t E$ is strongly equivalent to $\tau_t E_{\leq} \wedge \tau_t E_{\geq}$.

As an example application of Theorem 13, recall (5.3), the closed aggregate atom occurring in the instance of R_2 resulting from substituting the precomputed term r for X. Let E be this aggregate atom. Theorem 13 tells us that $\tau_{\bar{1}} E$ can be replaced by

$$\tau_{\overline{1}}(\textit{count}\{Y : q(r, Y)\} \leq \overline{1}) \wedge \tau_{\overline{1}}(\textit{count}\{Y : q(r, Y)\} \geq \overline{1}).$$

Since $[\overline{1}]$ is a singleton it is clear that this formula is the same as

$$\tau(\textit{count}\{Y : q(r, Y)\} \leq \overline{1}) \wedge \tau(\textit{count}\{Y : q(r, Y)\} \geq \overline{1}). \tag{8.3}$$

As observed in Section 5.4, $\tau_{\overline{1}}E$ is τE. So τE is strongly equivalent to (8.3).

It is worth noting that there is no result analogous to Theorem 13 for a closed aggregate atom E of the form $\alpha\{\mathbf{t} : \mathbf{L}\} \neq s$. We may be tempted to replace $\tau_t E$ in this case by the disjunction of the result of applying τ_t to aggregate atoms with strict inequalities, but this is not, in general, a strongly equivalent transformation.

8.3 Properties of Counting

The theorems in Section 8.1 show that in the case of monotone and anti-monotone aggregate atoms the result of applying τ_t is strongly equivalent to an infinite conjunction of a particular form. In this section, we show that for many aggregate atoms involving *count* some of the conjunctive terms are redundant.

Theorem 14 (Gebser et al., 2015) *For any closed aggregate atom E of the form*

$$\textit{count}\{\mathbf{t} : \mathbf{L}\} \leq \overline{m}$$

where m is an integer and $\mathbf{t}$ is interval-free, τE is strongly equivalent to

$$\bigwedge_{\substack{\Delta\subseteq A\\ |[\Delta]|=m+1}} \neg \bigwedge_{\mathbf{r}\in\Delta} \tau(\mathbf{L}^{\mathbf{x}}_{\mathbf{r}}). \tag{8.4}$$

Theorem 15 (Gebser et al., 2015) *For any closed aggregate atom E of the form*

$$\textit{count}\{\mathbf{t} : \mathbf{L}\} \geq \overline{m}$$

where m is an integer and $\mathbf{t}$ is interval-free, τE is strongly equivalent to

$$\bigvee_{\substack{\Delta\subseteq A\\ |[\Delta]|=m}} \bigwedge_{\mathbf{r}\in\Delta} \tau(\mathbf{L}^{\mathbf{x}}_{\mathbf{r}}). \tag{8.5}$$

Without the assumption that $\mathbf{t}$ is interval-free the assertions of the theorems would be incorrect. For instance, if E is *count*$\{\overline{1}..\overline{2} : p\} \geq \overline{1}$ then τE is $\top \to p$, and (8.5) is $\bot$.

In the special (but common) case when $\mathbf{t}$ is a tuple of distinct variables and each variable in $\mathbf{L}$ occurs in $\mathbf{t}$, the condition $|[\Delta]| = m+1$ in (8.4) can be replaced by $|\Delta| = m+1$. Indeed, in this case Δ and $[\Delta]$ have the same cardinality because $[\Delta]$ is the set of tuples $\mathbf{r}$ of terms such that $\mathbf{r} \in \Delta$. Similarly, if $\mathbf{t}$ is a tuple of distinct variables and each variable in $\mathbf{L}$ occurs in $\mathbf{t}$, the condition $|[\Delta]| = m$ in (8.5) can be replaced by $|\Delta| = m$.

Using these observations in conjunction with Theorems 14 and 15, we see that (8.3) is strongly equivalent to

$$\bigwedge_{\substack{\Delta\subseteq A\\|\Delta|=2}} \neg \bigwedge_{s\in\Delta} q(r,s) \quad\wedge\quad \bigvee_{\substack{\Delta\subseteq A\\|\Delta|=1}} \bigwedge_{s\in\Delta} q(r,s). \tag{8.6}$$

It follows that (5.4) can be replaced with (8.6) within the translation of any program.

8.4 Proofs

8.4.1 Proof of Theorem 11

Theorem 11 *If a closed aggregate atom E is monotone, then for any term t, $\tau_t E$ is strongly equivalent to* (8.1) *where A is the set of all precomputed terms, and the conjunction extends over all subsets Δ of A that do not justify E with respect to t.*

Proof. In view of Corollary 1, in order to establish the strong equivalence of these formulas, we need only to establish that they are equivalent in HT$^\infty$.

The implication from (8.1) to $\tau_t E$ is obvious. Now, assume $\tau_t E$, and consider the conjunctive term of (8.1) corresponding to a set Δ_0 that does not justify E with respect to t. Since E is monotone, from $\tau_t E$ we can derive the conjunction of all terms (5.2) where $\Delta \subseteq \Delta_0$. Any formula of the form

$$\left(F \rightarrow \bigvee_{i\in I} G_i\right) \rightarrow \left(\bigvee_{i\in I} (F \rightarrow G_i)\right)$$

is provable in HT$^\infty$. Consequently, we can derive the conjunction of the disjunctions

$$\bigvee_{\mathbf{r}\in A\setminus\Delta}\left(\bigwedge_{\mathbf{s}\in\Delta}\tau(\mathbf{L}^{\mathbf{x}}_{\mathbf{s}})\;\rightarrow\;\tau(\mathbf{L}^{\mathbf{x}}_{\mathbf{r}})\right)$$

over all $\Delta \subseteq \Delta_0$. By distributivity, this is equivalent in HT$^\infty$ to the disjunction of the conjunctions

$$\bigwedge_{\Delta\subseteq\Delta_0}\left(\bigwedge_{\mathbf{s}\in\Delta}\tau(\mathbf{L}^{\mathbf{x}}_{\mathbf{s}})\;\rightarrow\;\tau\left(\mathbf{L}^{\mathbf{x}}_{f(\Delta)}\right)\right) \tag{8.7}$$

over all functions f defined on all subsets Δ of Δ_0 such that $f(\Delta)$ is an element of $A \setminus \Delta$. Now we reason by cases, with one case corresponding to each disjunctive term (8.7). For a particular disjunctive term (8.7), we wish to show that we can derive from it $\tau(\mathbf{L}^{\mathbf{x}}_{\mathbf{r}})$ for some $\mathbf{r}$ in $A \setminus \Delta_0$. Consider the set Δ^* of all pairs $\mathbf{r}$ such that $\tau(\mathbf{L}^{\mathbf{x}}_{\mathbf{r}})$ is derivable from (8.7) in HT$^\infty$. We will show that Δ^* is not a subset of Δ_0. Assume $\Delta^* \subseteq \Delta_0$, so that

$$f(\Delta^*) \in A \setminus \Delta^*, \tag{8.8}$$

and one of the conjunctive terms of (8.7) is

$$\bigwedge_{\mathbf{s}\in\Delta^*}\tau(\mathbf{L}^{\mathbf{x}}_{\mathbf{s}})\;\rightarrow\;\tau\left(\mathbf{L}^{\mathbf{x}}_{f(\Delta^*)}\right).$$

By the definition of Δ^*, every conjunctive term in the antecedent of this implication is derivable from (8.7). It follows that the consequent is derivable as well, so that $f(\Delta^*)$ belongs to Δ^*, which contradicts (8.8). Therefore, Δ^* is not a subset of Δ_0, so that at least one disjunctive term of (8.1) is derivable from (8.7). $\square$

8.4.2 Proof of Theorem 12

Theorem 12 *If a closed aggregate atom E is anti-monotone, then for any term t, $\tau_t E$ is strongly equivalent to* (8.2) *where A is the set of all precomputed terms, and the conjunction extends over all subsets Δ of A that do not justify E with respect to t.*

Proof. The implication from (8.2) to $\tau_t E$ is obvious. We will derive (8.2) from $\tau_t E$ using the disjunction

$$\bigvee_{\Delta \subseteq A} \left(\neg \bigvee_{\mathbf{r} \in A \setminus \Delta} \tau(\mathbf{L}^{\mathbf{x}}_{\mathbf{r}}) \wedge \neg\neg \bigwedge_{\mathbf{r} \in \Delta} \tau(\mathbf{L}^{\mathbf{x}}_{\mathbf{r}}) \right). \tag{8.9}$$

This is a special case of the generalized law of weak excluded middle (6.15). Each disjunctive term (8.9) corresponds to a subset Δ of A. We will reason by cases with one case corresponding to each disjunctive term D_Δ of (8.9). Each D_Δ is equivalent to

$$\neg \left(\bigwedge_{\mathbf{r} \in \Delta} \tau(\mathbf{L}^{\mathbf{x}}_{\mathbf{r}}) \rightarrow \bigvee_{\mathbf{r} \in A \setminus \Delta} \tau(\mathbf{L}^{\mathbf{x}}_{\mathbf{r}}) \right) \tag{8.10}$$

in HT$^\infty$. If Δ does not justify E with respect to t, then (8.10) contradicts one of the conjunctive terms of (5.2) and (8.2) follows. Consider now the case when Δ justifies E with respect to t. Assume

$$\bigwedge_{\mathbf{r} \in \Delta_0} \tau(\mathbf{L}^{\mathbf{x}}_{\mathbf{r}}) \tag{8.11}$$

for some Δ_0 that does not justify E with respect to t. Since E is anti-monotone

Δ_0 is not a subset of Δ. We conclude that Δ_0 and $A \setminus \Delta$ overlap on at least one element. It follows that

$$\bigvee_{\mathbf{r} \in A \setminus \Delta} \tau(\mathbf{L}^{\mathbf{x}}_{\mathbf{r}})$$

can be derived from (8.11) since at least one of its disjunctive terms can be derived. This disjunction contradicts (8.10), and so we may conclude the negation of (8.11). $\square$

8.4.3 Proof of Theorem 13

Theorem 13 *Let E be a closed aggregate atom of the form*

$$\alpha\{\mathbf{t} : \mathbf{L}\} = s.$$

Let $E_{\leq}$ be

$$\alpha\{\mathbf{t} : \mathbf{L}\} \leq s$$

and $E_{\geq}$ be

$$\alpha\{\mathbf{t} : \mathbf{L}\} \geq s,$$

then for any precomputed term t, $\tau_t E$ is strongly equivalent to $\tau_t E_{\leq} \wedge \tau_t E_{\geq}$.

Proof. We will show that for any term t, the set of conjunctive terms of $\tau_t E$ is the union of the sets of conjunctive terms of $\tau_t E_{\leq}$ and $\tau_t E_{\geq}$. For any subset Δ of A, (5.2) is a conjunctive term of $\tau_t E$

iff Δ does not justify E with respect to t

iff $\widehat{\alpha}[\Delta] \neq t$

iff $\widehat{\alpha}[\Delta] < t$ or $\widehat{\alpha}[\Delta] > t$

iff Δ does not justify $E_{\geq}$ with respect to t or Δ does not justify $E_{\leq}$ with respect to t

iff (5.2) is a conjunctive term of $\tau_t E_{\geq}$ or of $\tau_t E_{\leq}$

iff (5.2) is a conjunctive term of $\tau_t E_{\leq} \wedge \tau_t E_{\geq}$. $\square$

8.4.4 Proof of Theorem 14

Theorem 14 *For any closed aggregate atom E of the form*

$$\textit{count}\{\mathbf{t} : \mathbf{L}\} \leq \overline{m}$$

where m is an integer and $\mathbf{t}$ *is interval-free, τE is strongly equivalent to* (8.4).

Proof. Since $[\overline{m}]$ is the singleton set $\{\overline{m}\}$, τE is $\tau_{\overline{m}} E$. By Theorem 12, $\tau_{\overline{m}} E$ is strongly equivalent to

$$\bigwedge_{\substack{\Delta \subseteq A \\ |[\Delta]| > m}} \neg \bigwedge_{\mathbf{r} \in \Delta} \tau(\mathbf{L}^{\mathbf{x}}_{\mathbf{r}}). \tag{8.12}$$

Every conjunctive term of (8.4) is a conjunctive term of (8.12). To derive (8.12) from (8.4), consider a set Δ such that $|[\Delta]| > m$. Let $f(\mathbf{r})$ stand for the set $[\mathbf{t}^{\mathbf{x}}_{\mathbf{r}}]$. Since $\mathbf{t}$ is interval-free, this set is either empty or a singleton. Let $\mathbf{s}_1, \dots, \mathbf{s}_{m+1}$ be $m+1$ distinct elements of $[\Delta]$. Choose elements $\mathbf{r}, \dots, \mathbf{r}_{m+1}$ of Δ such that each s_k belongs to $f(\mathbf{r}_k)$, and let Δ' be $\{\mathbf{r}, \dots, \mathbf{r}_{m+1}\}$. The cardinality of $[\Delta']$ is at least $m+1$, because this set includes $\mathbf{s}_1, \dots, \mathbf{s}_{m+1}$. On the other hand, it is at most $m + 1$, because this set is the union of $m + 1$ sets of cardinality at most 1. Consequently, $|[\Delta']| = m + 1$. From (8.4) we can conclude in HT$^\infty$ that

$$\neg \bigwedge_{\mathbf{r} \in \Delta'} \tau(\mathbf{L}^{\mathbf{x}}_{\mathbf{r}}). \tag{8.13}$$

Then the conjunctive term

$$\neg \bigwedge_{\mathbf{r} \in \Delta} \tau(\mathbf{L}^{\mathbf{x}}_{\mathbf{r}})$$

of (8.12) follows, because $\Delta' \subseteq \Delta$. □

8.4.5 Proof of Theorem 15

Theorem 15 *For any closed aggregate atom E of the form*

$$\textit{count}\{\mathbf{t} : \mathbf{L}\} \geq \overline{m}$$

where m is an integer and $\mathbf{t}$ *is interval-free, τE is strongly equivalent to* (8.5).

Proof. Since $[\overline{m}]$ is the singleton set $\{\overline{m}\}$, τE is $\tau_{\overline{m}} E$. By Theorem 11, the antecedent of (5.2) can be dropped (Section 8.1), so that $\tau_{\overline{m}} E$ is strongly equivalent to

$$\bigwedge_{\substack{\Delta \subseteq A \\ |[\Delta]| < m}} \bigvee_{\mathbf{r} \in A \setminus \Delta} \tau(\mathbf{L}^{\mathbf{x}}_{\mathbf{r}}). \tag{8.14}$$

To derive (8.14) from (8.5) in HT$^\infty$, assume (8.5). We reason by cases, with one case corresponding to each disjunctive term

$$\bigwedge_{\mathbf{r}\in\Delta} \tau(\mathbf{L}^{\mathbf{x}}_{\mathbf{r}}) \tag{8.15}$$

of (8.5). Let Δ' be a subset of A such that $|[\Delta']| < m$. We will show that the conjunctive term of (8.14) corresponding to Δ' can be derived from (8.15). Since

$$|[\Delta']| < m = |[\Delta]|, \tag{8.16}$$

there exists a tuple $\mathbf{r}$ that is an element of Δ but not an element of Δ'. Indeed, if $\Delta \subseteq \Delta'$ then $[\Delta] \subseteq [\Delta']$, which contradicts (8.16). Since $\mathbf{r} \in \Delta$, from (8.15) we can derive $\tau(\mathbf{L}^{\mathbf{x}}_{\mathbf{r}})$. Since $\mathbf{r} \in A \setminus \Delta'$, we can further derive

$$\bigvee_{\mathbf{r}\in A\setminus\Delta'} \tau(\mathbf{L}^{\mathbf{x}}_{\mathbf{r}}).$$

It follows that each conjunctive term of (8.14) can be derived from (8.15).

We prove by induction on m that (8.5) can be derived from (8.14) in HT$^\infty$. Base case: when $m = 0$ the disjunctive term of (8.5) corresponding to the empty Δ is$\top$. Inductive step: assume that (8.5) can be derived from (8.14), and assume

$$\bigwedge_{\substack{\Delta\subseteq A\\ |[\Delta]|<m+1}} \bigvee_{\mathbf{r}\in A\setminus\Delta} \tau(\mathbf{L}^{\mathbf{x}}_{\mathbf{r}}). \tag{8.17}$$

From (8.17) we can derive (8.14), and consequently (8.5). Now we reason by cases, with one case corresponding to each disjunctive term of (8.5). Assume

$$\bigwedge_{\mathbf{r}\in\Sigma} \tau(\mathbf{L}^{\mathbf{x}}_{\mathbf{r}}) \tag{8.18}$$

where Σ is a subset of A such that $|[\Sigma]| = m$. Consider the set

$$\Sigma' = \{\mathbf{r} : [\mathbf{t}^{\mathbf{x}}_{\mathbf{r}}] \subseteq [\Sigma]\}.$$

By the definition of $[\Sigma]$, for any $\mathbf{r} \in \Sigma$, $[\mathbf{t}^{\mathbf{x}}_{\mathbf{r}}] \subseteq [\Sigma]$. So $\Sigma \subseteq \Sigma'$. It follows that $[\Sigma] \subseteq [\Sigma']$. On the other hand,

$$[\Sigma'] = \bigcup_{\mathbf{r}\in\Sigma'} [\mathbf{t}^{\mathbf{x}}_{\mathbf{r}}] = \bigcup_{\mathbf{r}\,:\,[\mathbf{t}^{\mathbf{x}}_{\mathbf{r}}]\subseteq[\Sigma]} [\mathbf{t}^{\mathbf{x}}_{\mathbf{r}}] \subseteq [\Sigma].$$

Consequently, $[\Sigma] = [\Sigma']$, and $|[\Sigma']| = |[\Sigma]| = m$. From (8.17),

$$\bigvee_{\mathbf{r}\in A\setminus\Sigma'} \tau(\mathbf{L}^{\mathbf{x}}_{\mathbf{r}}). \tag{8.19}$$

Again, we reason by cases, with one case corresponding to each disjunctive term of (8.19). Assume $\tau(\mathbf{L})^{\mathbf{x}}_{\mathbf{s}})$, where $\mathbf{s} \in A \setminus \Sigma'$. Combining assumption (8.18) and $\tau(\mathbf{L})^{\mathbf{x}}_{\mathbf{s}})$, we derive

$$\bigwedge_{\mathbf{r})\in\Sigma\cup\{\mathbf{s}\}} \tau(\mathbf{L}^{\mathbf{x}}_{\mathbf{r}}). \tag{8.20}$$

Consider the set $[\Sigma \cup \{\mathbf{s}\}]$, that is,

$$[\Sigma] \cup [\mathbf{t}^{\mathbf{x}}_{\mathbf{s}}]. \tag{8.21}$$

Recall that the cardinality of $[\Sigma]$ is m. Since $\mathbf{t}$ is interval-free, the cardinality of $[\mathbf{t}_{\mathbf{s}}^{\mathbf{x}}]$ is at most 1. Furthermore, since $\mathbf{s} \notin \Sigma'$ it follows that

$$[\mathbf{t}_{\mathbf{s}}^{\mathbf{x}}] \not\subseteq [\Sigma],$$

so that $[\mathbf{t}_{\mathbf{s}}^{\mathbf{x}}]$ is nonempty. Consequently, the set is a singleton, and therefore $[\Sigma]$ is disjoint from it. It follows that the cardinality of (8.21) is $m + 1$. So from (8.20) we can derive

$$\bigvee_{\substack{\Delta \subseteq A \\ |[\Delta]| = m+1}} \bigwedge_{\mathbf{r} \in \Delta} \tau(\mathbf{L}_{\mathbf{r}}^{\mathbf{x}}).$$

□

Chapter 9

Extensional Atoms and Symmetric Splitting for Infinitary Formulas

In this chapter, we augment the definition of a stable model for infinitary formulas by introducing a distinction between "intensional" and "extensional" atoms.[1] Intuitively, an intensional atom is one that is included in a stable model if it is justified by the program, while extensional atoms are defined externally. Our definition generalizes the notion of intensional and extensional predicates from Ferraris et al. (2011). Similar distinctions have been proposed many times: Gelfond and Przymusinska (1996) distinguish between input and output predicates in their "lp-functions", Oikarinen and Janhunen (2008) distinguish between input and output atoms, and Lierler and Truszczynski (2011) between input and non-input atoms. These distinctions are useful because they allow for a modular view of logic programs. For example, in the splitting theorem from Ferraris et al. (2009), the authors showed that stable models for a program can sometimes be computed by breaking the program into parts and computing the stable models of each part separately using different sets of intensional predicates. We use the definition of stable models for infinitary formulas with intensional atoms to generalize that result. In particular,

[1]The material in this chapter was originally published in the following publication:

Amelia Harrison and Vladimir Lifschitz. Stable models for infinitary formulas with extensional atoms. Theory and Practice of Logic Programming, 16(5-6):771 786, 2016.

My own contributions to that paper included definitions, theorem statements, and proofs.

we look at the splitting lemma from Ferraris et al. (2009), which showed that under certain conditions the stable models of a formula can be computed by computing the stable models of the same formula with respect to smaller sets of intensional predicates. We find that a straightforward infinitary counterpart to the splitting lemma does not hold, and show how the lemma needs to be modified for the infinitary case. The situation is similar for the splitting theorem discussed above. The infinitary splitting theorem is used to generalize the lemma on explicit definitions due to Ferraris (2005), which describes how adding explicit definitions to a program affects its stable models.

9.1 $\mathcal{A}$-stable Models

Following Ferraris et al. (2011), we will assume that some atoms in a program are designated "intensional" while all others are regarded as "extensional".

Recall that σ denotes a propositional signature. Let $\mathcal{A} \subseteq \sigma$ be a (possibly infinite) set of atoms. The partial order $\leq_{\mathcal{A}}$ is defined as follows: for any sets $I, J \subseteq \sigma$, we say that $I \leq_{\mathcal{A}} J$ if $I \subseteq J$ and $J \setminus I \subseteq \mathcal{A}$. (Intuitively, if the atoms in $\mathcal{A}$ are treated as intensional and all other atoms from σ are treated as extensional, the relation holds if $I \subseteq J$ and I, J agree on all extensional atoms.) An interpretation I is called an (infinitary) $\mathcal{A}$-*stable model* of a formula F if it is a minimal model of F^I w.r.t. $\leq_{\mathcal{A}}$.

Observe that if $\mathcal{A} = \sigma$ then $\mathcal{A}$-stable models of a formula F are the same as stable models. If $\mathcal{A} = \emptyset$ then $\mathcal{A}$-stable models are all models of F. Truszczynski observed that an interpretation I satisfies F iff I satisfies F^I (2012, Proposition 1).

It follows that all $\mathcal{A}$-stable models of F also satisfy F.

To illustrate the definition of $\mathcal{A}$-stability, let's find all $\{q\}$-stable models[2] of the infinitary formula (3.11). This formula captures the meaning of the AG rule

$$q \leftarrow \textit{count}\{X : p(X)\} = 0. \tag{9.1}$$

The stable model $\{q\}$ of (3.11) is $\{q\}$-stable as well, because it is a minimal model of (3.12) w.r.t. $\leq_{\{q\}}$. On the other hand, any non-empty set $\mathcal{P}$ of atoms of the form $p(t)$ is $\{q\}$-stable too. Indeed, the reduct of (3.11) w.r.t. such a set is an implication whose antecedent has $\bot$ as one of its conjunctive terms. Such a formula is tautological so that it is satisfied by $\mathcal{P}$. Furthermore, $\mathcal{P}$ is a minimal model w.r.t. $\leq_{\{q\}}$ since any subset of $\mathcal{P}$ will disagree with it on extensional atoms.

The fact that all stable models of (3.11) are also $\{q\}$-stable is an instance of a more general fact: If I is an $\mathcal{A}$-stable model of F and $\mathcal{B}$ is a subset of $\mathcal{A}$ then I is also a $\mathcal{B}$-stable model of F. This follows directly from the definition of $\mathcal{A}$-stability.

The following proposition provides two alternative definitions for $\mathcal{A}$-stability.

Proposition 3 (Harrison and Lifschitz, 2016) *The following three conditions are equivalent:*

(i) I is an $\mathcal{A}$-stable model of F;

[2]Here, we understand σ as implicitly defined to be the set containing q and all atoms of the form $p(t)$ where t is a ground term.

(ii) I is a minimal model (w.r.t. set inclusion) of

$$F^I \wedge \bigwedge_{p \in I \setminus \mathcal{A}} p; \tag{9.2}$$

(iii) I is a stable model of

$$F \;\; \wedge \bigwedge_{p \in \sigma \setminus \mathcal{A}} (p \vee \neg p). \tag{9.3}$$

9.2 Relating Infinitary and First-Order $\mathcal{A}$-Stable Models

Truszczynski (2012) showed that infinitary stable models can be viewed as a generalization of first-order stable models in the sense of Ferraris et al. (2011). In this section, we show that the corresponding result holds for **p**-stable models as well.[3] First, we review Truszczynski's results.

Let Σ be a first-order signature, and I be an interpretation of Σ with non-empty domain $|I|$. For each element u of $|I|$, by u^* we denote a new object constant, called the *name of* u. By $\Sigma^{|I|}$ we denote the signature obtained by adding the names of all elements of $|I|$ to Σ. An interpretation I is identified with its extension I' to $\Sigma^{|I|}$ in which for each u in $|I|$, $I'(u^*) = u$. By $A_{\Sigma,I}$ we denote the set of all atomic sentences over $\Sigma^{|I|}$ built with relation symbols from Σ and names of elements in $|I|$, and by I^r we denote the subset of $A_{\Sigma,I}$ that describes in the obvious way the extents of the relations in I. Let F be a formula over signature $\Sigma^{|I|}$. Then the *grounding*

[3]The definition of **p**-stable models, where **p** is a list of distinct predicate symbols, can be found in (Ferraris et al., 2011, Section 2.3).

of F w.r.t. I, $gr_I(F)$ is defined recursively, as follows:

- $\mathrm{gr}_I(\bot)$ is $\bot$;
- $\mathrm{gr}_I(p(t_1,\dots,t_k))$ is $p((t_1^I)^*,\dots,(t_k^I)^*)$;
- $\mathrm{gr}_I(t_1 = t_2)$ is $\top$ if $t_1^I = t_2^I$ and $\bot$ otherwise;
- $\mathrm{gr}_I(F \odot G)$ is $\mathrm{gr}_I(F)\odot$ $\mathrm{gr}_I(G)$, where $\odot \in \{\wedge,\vee,\rightarrow\}$;
- $\mathrm{gr}_I(\forall x F(x))$ is $\{\mathrm{gr}_I(F^x_{u^*}) | u \in |I|\}^\wedge$;
- $\mathrm{gr}_I(\exists x F(x))$ is $\{\mathrm{gr}_I(F^x_{u^*}) | u \in |I|\}^\vee$.

(By $F^x_{u^*}$ we denote the result of substituting u^* for all free occurrences of x in F.) It is clear that for any first-order sentence F over signature Σ, $\mathrm{gr}_I(F)$ is an infinitary formula over the signature $A_{\Sigma,I}$.

For example, the first-order formula

$$\forall x\ \neg p(x) \rightarrow q \tag{9.4}$$

can also be used to capture the meaning of rule (9.1). If Σ consists of the unary predicate p and the propositional symbol q, and I is an interpretation of Σ such that the domain $|I|$ is the set of all ground terms t, then the grounding of the first-order formula w.r.t. I is (3.11). (To simplify notation we identify the name of each term t with t.)

According to Theorem 5 from (Truszczynski, 2012), if F is a first-order sentence and I is an interpretation, then I is a first-order stable model of F iff I^r is an infinitary stable model of $\mathrm{gr}_I(F)$. The proposition below generalizes this result to the case of **p**-stable models. By $\mathbf{p}^I$ we denote the atomic formulas in $A_{\Sigma,I}$ built with predicates from **p**.

For example, if **p** is p then $\mathbf{p}^I$ is the set of all atoms of the form $p(t)$.

Proposition 4 (Harrison and Lifschitz, 2016) *For any first-order sentence F over Σ and any tuple **p** of distinct predicate symbols from Σ, an interpretation I is a **p**-stable model of F iff I^r is a $\mathbf{p}^I$-stable model of $gr_I(F)$.*

For example, let I be the interpretation that interprets p as identically false and assigns the value $\top$ to q. Then I^r is $\{q\}$. Let J be an interpretation that satisfies at least one atomic formula $p(t)$ and assigns the value $\bot$ to q. Then J^r is $\{p(t) \mid J \models p(t)\}$ (the same as $\mathcal{P}$ from the previous section). We saw in the previous section that $\{q\}$-stable models of (3.11) are $\{q\}$ and any non-empty set of atoms of the form $p(t)$. In accordance with the proposition above, I and J are $\{q\}$-stable models of (9.4).

9.3 Review: First-Order Splitting Lemma

The lemma presented in the next section is a generalization of the splitting lemma from Ferraris et al. (2009). In order to state that lemma, we first review the definition of the predicate dependency graph given in that paper. We say that an occurrence of a predicate symbol or a subformula in a first-order formula F is *positive* if it occurs in the antecedent of an even number of implications and *strictly positive* if it occurs in the antecedent of no implication. An occurrence of a predicate constant is said to be *negated* if it belongs to a subformula of the form $\neg F$, and *nonnegated* otherwise. A *rule* of a first-order formula F is a strictly positive occurrence of an implication in F. The *(positive) predicate dependency graph* of a first-order formula F w.r.t. a list $\mathbf{p}$ of distinct predicates, denoted $\mathrm{DG}_{\mathbf{p}}[F]$ is the directed graph that

- has all predicate symbols in $\mathbf{p}$ as its vertices, and
- has an edge from p to q if, for some rule $G \to H$ of F,
  - p has a strictly positive occurrence in H, and
  - q has a positive nonnegated occurrence in G.

We say that a partition[4] $\{\mathbf{p}_1, \mathbf{p}_2\}$ of the vertices in a graph G is *separable (on G)* if every strongly connected component of G is a subset of either $\mathbf{p}_1$ or $\mathbf{p}_2$. (Here, we identify the list $\mathbf{p}$ with the set of its members.)

The following assertion is a reformulation of Version 1 of the splitting lemma from Ferraris et al. (2009).

Splitting Lemma *If F is a first-order sentence and $\boldsymbol{p}_1, \boldsymbol{p}_2$ are lists of distinct predicate symbols such that the partition $\{\boldsymbol{p}_1, \boldsymbol{p}_2\}$ is separable on $DG_{\boldsymbol{p}_1\boldsymbol{p}_2}[F]$ then I is a $\boldsymbol{p}_1\boldsymbol{p}_2$-stable model of F iff it is both a $\boldsymbol{p}_1$-stable model and a $\boldsymbol{p}_2$-stable model of F.*

9.4 Infinitary Splitting Lemma

The statement of the infinitary splitting lemma refers to the positive dependency graph of an infinitary formula. As we will see, the vertices of this graph correspond to intensional atoms. This definition is similar to the definition of a predicate dependency graph from Ferraris et al. (2009) reviewed in the previous

[4]We understand a partition of X to be a set of disjoint subsets (possibly empty) that cover X.

section. The concepts necessary to define the dependency graph of an infinitary formula are all straightforward extensions of the concepts used in the previous section to define the predicate dependency graph in the first-order case. However, because infinitary formulas are not syntactic structures, we have to define these concepts recursively.

We define the set of *strictly positive atoms* of an infinitary formula F, denoted P(F), recursively, as follows:

- For every atom $p \in \sigma$, P(p) is $\{p\}$;
- P($\mathcal{H}^\wedge$) is $\bigcup_{H \in \mathcal{H}}$ P(H), and so is P($\mathcal{H}^\vee$);
- P($G \to H$) is P(H).

The set of *positive nonnegated atoms* and the set of *negative nonnegated atoms* of an infinitary formula F, denoted Pnn(F) and Nnn(F) respectively, were introduced in (Lifschitz and Yang, 2012). These sets are defined recursively as well:

- For every atom $p \in \sigma$, Pnn(p) is $\{p\}$;
- Pnn($\mathcal{H}^\wedge$) is $\bigcup_{H \in \mathcal{H}}$ Pnn(H), and so is Pnn($\mathcal{H}^\vee$);
- Pnn($G \to H$) is $\emptyset$ if H is $\bot$ and Nnn(G) $\cup$ Pnn(H) otherwise.

and

- For every atom $p \in \sigma$, Nnn(p) is $\emptyset$;
- Nnn($\mathcal{H}^\wedge$) is $\bigcup_{H \in \mathcal{H}}$ Nnn(H), and so is Nnn($\mathcal{H}^\vee$);
- Nnn($G \to H$) is $\emptyset$ if H is $\bot$ and Pnn(G) $\cup$ Nnn(H) otherwise.

The set of *rules* of an infinitary formula is defined as follows:

- The rules of $G \to H$ are $G \to H$ and all rules of H;
- The rules of $\mathcal{H}^\wedge$ and $\mathcal{H}^\vee$ are the rules of all formulas in $\mathcal{H}$.

For example, the set of positive nonnegated atoms in formula (3.11) is the same as the set of strictly positive atoms: $\{q\}$. The only rule of formula (3.11) is the formula itself.

For any infinitary formula F the *(positive) dependency graph* of F (relative to a set of atoms $\mathcal{A}$), denoted $\mathrm{DG}_{\mathcal{A}}[F]$, is the directed graph, that

- has all atoms in $\mathcal{A}$ as its vertices, and
- has an edge from p to q if, for some rule $G \to H$ of F,
  - p is an element of P(H), and
  - q is an element of Pnn(G).

The following statement appears to be a plausible counterpart to the splitting lemma reproduced in Section 9.3 for infinitary formulas:

If F is an infinitary formula and $\mathcal{P}_1, \mathcal{P}_2$ are sets of atoms such that the partition $\{\mathcal{P}_1, \mathcal{P}_2\}$ is separable on $\mathrm{DG}_{\mathcal{P}_1 \cup \mathcal{P}_2}[F]$ then I is a $\mathcal{P}_1 \cup \mathcal{P}_2$-stable model of F iff it is both a $\mathcal{P}_1$-stable model and a $\mathcal{P}_2$-stable model of F. (∗)

But this statement does not hold; in the case of infinitary formulas separability is not a sufficient condition to ensure splittability. Let F be the infinitary

$$\begin{tikzcd} \cdots \arrow[r] & p_{-1} \arrow[r] & p_0 \arrow[r] & p_1 \arrow[r] & \cdots \end{tikzcd}$$

Figure 9.1: Any partition of the vertices in this graph is separable.

conjunction

$$\bigwedge_n (p_{n+1} \to p_n),$$

where the conjunction extends over all integers n. Let $\mathcal{P}$ be the set of all atoms p_n. Let $\mathcal{P}_1$ be the set $\{p_n \mid n \text{ is even}\}$, and $\mathcal{P}_2$ be the set $\{p_n \mid n \text{ is odd}\}$. Then the partition $\{\mathcal{P}_1, \mathcal{P}_2\}$ is separable on $\mathrm{DG}_{\mathcal{P}}[F]$ (shown in Figure 9.1). Indeed, the strongly connected components of this graph are singletons. If I is the set of all atoms p_n then the reduct of F w.r.t. I is F itself. It is easy to check that I is a $\mathcal{P}_1$-stable model as well as a $\mathcal{P}_2$-stable model of F, but is not $\mathcal{P}$-stable. This counterexample shows that $(*)$ is incorrect.

In order to extend the splitting lemma to infinitary formulas, we will need a stronger notion of separability. An *infinite walk* W of a directed graph G is an infinite sequence $(v_1, v_2, \dots)$ of vertices occurring in G, such that each pair v_i, v_{i+1} in W corresponds to an edge in G. A partition $\{\mathcal{P}_1, \mathcal{P}_2\}$ of the vertices in G will be called *infinitely separable (on G)* if every infinite walk $(v_1, v_2, \dots)$ of G visits either $\mathcal{P}_1$ or $\mathcal{P}_2$ finitely many times, that is either $\{i : v_i \in \mathcal{P}_1\}$ or $\{i : v_i \in \mathcal{P}_2\}$ is finite.

Proposition 5 (Harrison and Lifschitz, 2016) *For any graph G,*

(i) every infinitely separable partition of G is separable, and

(ii) if G has finitely many strongly connected components and partition $\{\mathcal{P}_1, \mathcal{P}_2\}$

is separable on G then it is infinitely separable on G.

Claim $(*)$ will become correct if we require the partition $\{\mathcal{P}_1, \mathcal{P}_2\}$ to be infinitely separable:

Lemma 6 (Infinitary Splitting Lemma, Harrison and Lifschitz, 2016) *If F is an infinitary formula and $\mathcal{P}_1, \mathcal{P}_2$ are sets of atoms such that the partition $\{\mathcal{P}_1, \mathcal{P}_2\}$ is infinitely separable on $DG_{\mathcal{P}_1 \cup \mathcal{P}_2}[F]$ then I is a $\mathcal{P}_1 \cup \mathcal{P}_2$-stable model of F iff it is both a $\mathcal{P}_1$-stable model and a $\mathcal{P}_2$-stable model of F.*

The splitting lemma reproduced in Section 9.3 is a consequence of the infinitary splitting lemma in view of Proposition 4 and the following fact:

Proposition 6 (Harrison and Lifschitz, 2016) *For any first-order sentence F and tuple $\boldsymbol{p}$ of distinct predicate symbols, if $\{\boldsymbol{p}_1, \boldsymbol{p}_2\}$ is a partition of $\boldsymbol{p}$ that is separable on $DG_{\boldsymbol{p}}[F]$, then for any interpretation I, $\{\boldsymbol{p}_1^I, \boldsymbol{p}_2^I\}$ is infinitely separable on $DG_{\boldsymbol{p}^I}[gr_I(F)]$.*

9.5 Infinitary Splitting Theorem

The infinitary splitting lemma can be used to prove the following theorem, which is similar to the splitting theorem from (Ferraris et al., 2009).

Theorem 16 (Infinitary Splitting Theorem, Harrison and Lifschitz, 2016) *Let F, G be infinitary formulas, and $\mathcal{A}_1, \mathcal{A}_2$ be disjoint sets of atoms such that the partition $\{\mathcal{A}_1, \mathcal{A}_2\}$ is infinitely separable on $DG_{\mathcal{A}_1 \cup \mathcal{A}_2}[F \wedge G]$. If $\mathcal{A}_2$ is disjoint from*

$P(F)$, and $\mathcal{A}_1$ is disjoint from $P(G)$, then for any interpretation I, I is an $\mathcal{A}_1 \cup \mathcal{A}_2$-stable model of $F \wedge G$ iff it is both an $\mathcal{A}_1$-stable model of F and an $\mathcal{A}_2$-stable model of G.

For example, consider the conjunction of (3.11) with the formula $\mathcal{P}^\wedge$ where $\mathcal{P}$ is as before some non-empty set of atoms of the form $p(t)$. We saw previously that $\{q\}$ and all non-empty sets of atoms of the form $p(t)$ are $\{q\}$-stable models of (3.11). It is easy to check that $\sigma \backslash \{q\}$-stable models of $\mathcal{P}^\wedge$ are $\mathcal{P}$ and $\mathcal{P} \cup \{q\}$. In accordance with the splitting theorem, $\mathcal{P}$ is the only stable model of this formula.

The infinitary splitting theorem can be used, for example, to generalize teh lemma on explicit definitions due to (Ferraris, 2005).

About a formula G and a set $\mathcal{Q}$ of atoms we will say that G is a *definition for $\mathcal{Q}$* if it is a conjunction of a set of formulas of the form $H \wedge \mathcal{C}^\wedge \rightarrow q$, where q is an atom in $\mathcal{Q}$, C is a subset of $\mathcal{Q}$ (possibly empty), and no atoms from $\mathcal{Q}$ occur in H.[5]

A simple special case is "explicit definitions": conjunctions of formulas $H \rightarrow q$ such that atoms from $\mathcal{Q}$ don't occur in any H. For example, (3.11) is an explicit definition of $\{q\}$. The conjunction of the formulas

$$p_{\alpha\beta} \rightarrow q_{\alpha\beta} \quad \text{and} \quad q_{\alpha\beta} \wedge q_{\beta\gamma} \rightarrow q_{\alpha\gamma}$$

for all α, β, γ from some set of indices, which represents the usual recursive definition of transitive closure, is a definition in our sense as well. On the other hand, the formula $\neg q \rightarrow q$ is not a definition.

[5]The relation p *occurs in* F is defined recursively in a straightforward way.

The following theorem shows that all definitions are "conservative".

Theorem 17 (Harrison and Lifschitz, 2016) *For any infinitary formula F, any set $\mathcal{Q}$ of atoms that do not occur in F, and any definition G for $\mathcal{Q}$, the map $I \mapsto I \setminus \mathcal{Q}$ is a 1-1 correspondence between the stable models of $F \wedge G$ and the stable models of F.*

This theorem generalizes Ferraris's 2005 lemma in two ways: it applies to infinitary formulas, and it allows definitions to be recursive.

9.6 Proofs

9.6.1 Proof of Proposition 3

Proposition 3 *The following three conditions are equivalent:*

(i) I is an $\mathcal{A}$-stable model of F;

(ii) I is a minimal model (w.r.t. set inclusion) of (9.2)*;*

(iii) I is a stable model of (9.3).

Proof. We first establish that conditions (i) and (ii) are equivalent: I is an $\mathcal{A}$-stable model of F

iff I is a minimal model of F^I w.r.t. $\leq_{\mathcal{A}}$

iff $I \models F^I$ and there is no $J \subset I$ such that $J \models F^I$ and $I \setminus J \subseteq \mathcal{A}$

iff $I \models F^I$ and there is no $J \subset I$ such that $J \models F^I$ and $\forall p(p \in I \wedge p \notin J \rightarrow p \in \mathcal{A})$

iff $I \models F^I$ and there is no $J \subset I$ such that $J \models F^I$ and $\forall p(p \in I \wedge p \notin \mathcal{A} \rightarrow p \in J)$

iff $I \models F^I$ and there is no $J \subset I$ such that $J \models F^I$ and $I \setminus \mathcal{A} \subseteq J$

iff $I \models F^I \wedge \bigwedge_{p \in I \setminus \mathcal{A}} p$ and there is no $J \subset I$ such that $J \models F^I \wedge \bigwedge_{p \in I \setminus \mathcal{A}} p$

iff I is an minimal model of (9.2).

Finally, we will establish that conditions (ii) and (iii) are equivalent. It is easy to see that the reduct of (9.3) is equivalent to (9.2):

$$\begin{array}{lll}
 & F^I & \wedge \quad \left(\bigwedge_{p \in \sigma \setminus \mathcal{A}} (p \vee \neg p) \right)^I \\
\leftrightarrow & F^I & \wedge \quad \bigwedge_{p \in \sigma \setminus \mathcal{A}} (p^I \vee (\neg p)^I) \\
\leftrightarrow & F^I & \wedge \quad \bigwedge_{p \in I \setminus \mathcal{A}} (p^I \vee (\neg p)^I) \quad \wedge \quad \bigwedge_{p \in \sigma \setminus (I \cup \mathcal{A})} (p^I \vee (\neg p)^I) \\
\leftrightarrow & F^I & \wedge \quad \bigwedge_{p \in I \setminus \mathcal{A}} (p \vee \bot) \quad \wedge \quad \bigwedge_{p \in \sigma \setminus (I \cup \mathcal{A})} (\bot \vee \top) \\
\leftrightarrow & F^I & \wedge \quad \bigwedge_{p \in I \setminus \mathcal{A}} p.
\end{array}$$

So I is a minimal model of (9.2) iff it is a stable model of (9.3). $\square$

9.6.2 Proof of Propostion 4

Proposition 4 *For any first-order sentence F over Σ and any tuple **p** of distinct predicate symbols from Σ, an interpretation I is a **p**-stable model of F iff I^r is a $\mathbf{p}^I$-stable model of $gr_I(F)$.*

Proof. Consider a first-order sentence F and list of distinct predicate symbols **p**. Let $\mathcal{Q}$ be the set of all predicates occurring in F but not in **p**. Consider an interpretation I of the signature of F. By Theorem 2 from (Ferraris et al., 2011), I is a **p**-stable model of F iff it is a stable model of

$$F \wedge \bigwedge_{q \in \mathcal{Q}} \forall \mathbf{x}(q(\mathbf{x}) \vee \neg q(\mathbf{x})),$$

where $\mathbf{x}$ is a list of distinct object variables the same length as the arity of q. By Theorem 5 from (Truszczynski, 2012), I is a stable model of the formula above iff I^r is a stable model of the grounding of this formula w.r.t. I. The grounding of the formula above w.r.t. I is

$$\mathrm{g}r_I(F) \wedge \bigwedge_{\substack{q \in \mathcal{Q} \\ A \in q^I}} (A \vee \neg A)\,. \tag{9.5}$$

By Proposition 3, I^r is a stable model of (9.5) iff it is a $\mathbf{p}^I$-stable model of $\mathrm{gr}_I(F)$.
$\square$

9.6.3 Proof of Proposition 5

Proposition 5 *For any graph G,*

(i) every infinitely separable partition of G is separable, and

(ii) if G has finitely many strongly connected components and partition $\{\mathcal{P}_1, \mathcal{P}_2\}$ is separable on G then it is infinitely separable on G.

Proof. (i) We will prove the contrapositive: if $\{\mathcal{P}_1, \mathcal{P}_2\}$ is a partition that is not separable on G, then there is some strongly connected component of G that contains at least one vertex from $\mathcal{P}_1$ and at least one vertex from $\mathcal{P}_2$. Let's call these vertices v and w, respectively. Since v and w are in the same strongly connected component, each vertex is reachable from the other. Then there is an infinite walk that visits each of these vertices (and therefore both $\mathcal{P}_1$ and $\mathcal{P}_2$) infinitely many times, so that the partition is not infinitely separable on G.

(ii) Again we prove the contrapositive: if $\{\mathcal{P}_1, \mathcal{P}_2\}$ is a partition that is not infinitely separable on G, then there is some infinite walk $(v_1, v_2, \dots)$ of G that visits both $\mathcal{P}_1$ and $\mathcal{P}_2$ infinitely many times. Since there are only finitely many strongly connected components in G, at least one strongly connected component of $\mathcal{P}_1$ and at least one strongly connected component of $\mathcal{P}_2$ must be visited infinitely many times. Call these strongly connected components C_1 and C_2 respectively; then C_1 must be reachable from C_2 and vice versa. Then $C_1 = C_2$ so that the partition is not separable on G. $\square$

9.6.4 Proof of Proposition 6

Proposition 6 *For any first-order sentence F and tuple $\boldsymbol{p}$ of distinct predicate symbols, if $\{\boldsymbol{p}_1,\boldsymbol{p}_2\}$ is a partition of $\boldsymbol{p}$ that is separable on $DG_{\boldsymbol{p}}[F]$, then for any interpretation I, $\{\boldsymbol{p}_1^I,\boldsymbol{p}_2^I\}$ is infinitely separable on $DG_{\boldsymbol{p}^I}[gr_I(F)]$.*

Proof. If $\{\mathbf{p}_1, \mathbf{p}_2\}$ is a partition of $\mathbf{p}$ that is separable on $\mathrm{DG}_{\mathbf{p}}[F]$, then for any interpretation I, the partition $\{\mathbf{p}_1^I, \mathbf{p}_2^I\}$ is separable on the atomic dependency graph of $\mathrm{gr}_I(F)$ with respect to $\mathbf{p}^I$. Furthermore, it is easy to see that $\mathrm{DG}_{\mathbf{p}^I}[\mathrm{gr}_I(F)]$ must have finitely many strongly connected components, so that $\{\mathbf{p}_1^I, \mathbf{p}_2^I\}$ must be infinitely separable on it. $\square$

9.6.5 Proof of Lemma 6

The following proposition can be easily proved by induction on the rank of F.

Proposition 7 *If the set $\mathcal{A}$ is disjoint from $P(F)$ and I satisfies F, then $I \setminus \mathcal{A}$ satisfies F^I.*

In particular, if I satisfies F then I satisfies F^I. (This is the direction left-to-right of Proposition 1 from Truszczynski (2012).)

Propositions 8–10 are similar to Lemmas 3–5 from Ferraris et al. (2009).

Proposition 8 *For any disjoint sets of atoms $\mathcal{B}_1, \mathcal{B}_2$, interpretation I, and formula F,* *(i) If $\mathcal{B}_2$ is disjoint from Pnn(F) and $I \setminus \mathcal{B}_1$ satisfies F^I then $I \setminus (\mathcal{B}_1 \cup \mathcal{B}_2)$ satisfies F^I.*

(ii) If $\mathcal{B}_2$ is disjoint from Nnn(F) and $I \setminus (\mathcal{B}_1 \cup \mathcal{B}_2)$ satisfies F^I then $I \setminus \mathcal{B}_1$ satisfies F^I.

Proof. Both parts of the lemma are proved simultaneously by induction on the rank of F. Here, we show only the most interesting case when F is of the form $G \rightarrow H$. (i) If I does not satisfy F the reduct is equivalent to $\perp$ so that the proposition is trivially true. Assume that $I \setminus \mathcal{B}_1$ satisfies $G^I \rightarrow H^I$ and that $\mathcal{B}_2$ is disjoint from Pnn$(G \rightarrow H)$. Then either H is $\perp$ or $\mathcal{B}_2$ is disjoint from both Nnn(G) and Pnn(H). If H is $\perp$ then the set P(F) is empty, so that $(\mathcal{B}_1 \cup \mathcal{B}_2)$ is disjoint from it. Then by Proposition 7, if I satisfies F then $I \setminus (\mathcal{B}_1 \cup \mathcal{B}_2)$ satisfies F^I. If, on the other hand, $\mathcal{B}_2$ is disjoint from both Nnn(G) and Pnn(H), then by part (i) of the induction hypothesis we may conclude that

$$\text{if } I \setminus \mathcal{B}_1 \text{ satisfies } H^I \text{ then so does } I \setminus (\mathcal{B}_1 \cup \mathcal{B}_2), \tag{9.6}$$

and by part (ii) of the induction hypothesis we may conclude that

$$\text{if } I \setminus (\mathcal{B}_1 \cup \mathcal{B}_2) \text{ satisfies } G^I \text{ then so does } I \setminus \mathcal{B}_1. \tag{9.7}$$

Assume that $I \setminus (\mathcal{B}_1 \cup \mathcal{B}_2)$ satisfies G^I. Then by (9.7), $I \setminus \mathcal{B}_1$ satisfies G^I. Then, since $I \setminus \mathcal{B}_1$ satisfies $G^I \rightarrow H^I$, that interpretation must satisfy H^I. Then by (9.6) we can conclude that $I \setminus (\mathcal{B}_1 \cup \mathcal{B}_2)$ satisfies H^I. It follows that that $I \setminus (\mathcal{B}_1 \cup \mathcal{B}_2)$ satisfies $G^I \rightarrow H^I$. (ii) Similar to Part (i). $\square$

Proposition 9 *Let $\mathcal{B}, \mathcal{C}$ be disjoint sets of atoms and let F be an infinitary formula such that there are no edges from $\mathcal{B}$ to $\mathcal{C}$ in DG$_{\mathcal{B}\cup\mathcal{C}}[F]$. If $I \setminus (\mathcal{B} \cup \mathcal{C})$ satisfies F^I then so does $I \setminus \mathcal{B}$.*

Proof. The proof is by induction on the rank of F. Again we show only the most interesting case when F is of the form $G \rightarrow H$. Assume that $I \setminus (\mathcal{B} \cup \mathcal{C})$ satisfies $(G \rightarrow H)^I = G^I \rightarrow H^I$. We need to show that $I \setminus \mathcal{B}$ also satisfies $G^I \rightarrow H^I$. If $\mathcal{B}$ is disjoint from P(H), then by Proposition 7, $I \setminus \mathcal{B}$ satisfies H^I, and therefore satisfies $G^I \rightarrow H^I$. If, on the other hand, $\mathcal{B}$ is not disjoint from P(H) then $\mathcal{C}$ must be disjoint from Pnn(G), because there are no edges from $\mathcal{B}$ to $\mathcal{C}$ in DG$_{\mathcal{B}\cup\mathcal{C}}[G \rightarrow H]$. Then by Proposition 8(i), $I \setminus (\mathcal{B} \cup \mathcal{C})$ satisfies G^I. Since we assumed that $I \setminus (\mathcal{B} \cup \mathcal{C})$ satisfies $G^I \rightarrow H^I$, it follows that $I \setminus (\mathcal{B} \cup \mathcal{C})$ satisfies H^I. Since every edge in DG$_{\mathcal{B}\cup\mathcal{C}}[H]$ occurs in DG$_{\mathcal{B}\cup\mathcal{C}}[G \rightarrow H]$ there is no edge from $\mathcal{B}$ to $\mathcal{C}$ in DG$_{\mathcal{B}\cup\mathcal{C}}[H]$. Then by the induction hypothesis, $I \setminus \mathcal{B}$ satisfies H^I and therefore satisfies $G^I \rightarrow H^I$. $\square$

Proposition 10 *For any non-empty graph G and any infinitely separable partition $\{\mathcal{A}_1, \mathcal{A}_2\}$ on G, there exists a non-empty subset $\mathcal{B}$ of the vertices in G such that*

(i) $\mathcal{B}$ is either a subset of $\mathcal{A}_1$ or a subset of $\mathcal{A}_2$, and

(ii) there are no edges from $\mathcal{B}$ to vertices not in $\mathcal{B}$.

Proof. Since $\{\mathcal{A}_1, \mathcal{A}_2\}$ is infinitely separable on G, there is some vertex b such that the set of vertices reachable from b is either a subset of $\mathcal{A}_1$ or a subset of $\mathcal{A}_2$. (If no such b existed then $\mathcal{A}_1$ would be reachable from every vertex in $\mathcal{A}_2$ and vice versa, and we could construct an infinite walk visiting both elements of the partition infinitely many times.) It is easy to see that the set of all vertices reachable from b satisfies both (i) and (ii). $\Box$

Lemma 6 *If F is an infinitary formula and $\mathcal{P}_1, \mathcal{P}_2$ are sets of atoms such that the partition $\{\mathcal{P}_1, \mathcal{P}_2\}$ is infinitely separable on DG$_{\mathcal{P}_1 \cup \mathcal{P}_2}[F]$ then I is a $\mathcal{P}_1 \cup \mathcal{P}_2$-stable model of F iff it is both a $\mathcal{P}_1$-stable model and a $\mathcal{P}_2$-stable model of F.*

Proof. Let F be an infinitary formula such that the partition $\{\mathcal{A}_1, \mathcal{A}_2\}$ is infinitely separable on DG$_{\mathcal{A}_1 \cup \mathcal{A}_2}[F]$. We need to show that I is an $\mathcal{A}_1 \cup \mathcal{A}_2$-stable model of F iff it is an $\mathcal{A}_1$-stable model and an $\mathcal{A}_2$-stable model of F. The direction left-to-right is obvious. To establish the direction right-to-left, assume that I is both an $\mathcal{A}_1$-stable model and an $\mathcal{A}_2$-stable model of F. By Proposition 3 it is sufficient to show that I is a minimal model of

$$F^I \wedge \bigwedge_{p \in I \setminus (\mathcal{A}_1 \cup \mathcal{A}_2)} p. \tag{9.8}$$

Clearly, I satisfies this formula. It remains to show that I is minimal. Assume there is some non-empty subset X of I such that $I \setminus X$ satisfies (9.8). Then $I \setminus X$ satisfies the second conjunctive term of (9.8), so $I \setminus (\mathcal{A}_1 \cup \mathcal{A}_2) \subseteq I \setminus X$. Consequently, $X \subseteq \mathcal{A}_1 \cup \mathcal{A}_2$. Consider the sets $X \cap \mathcal{A}_1$ and $X \cap \mathcal{A}_2$. Since $\mathcal{A}_1$ and $\mathcal{A}_2$ are infinitely separable on DG$_{\mathcal{A}_1 \cup \mathcal{A}_2}[F]$, the sets $X \cap \mathcal{A}_1$ and $X \cap \mathcal{A}_2$ must be infinitely separable on DG$_X[F]$. Then by Proposition 10, there is some non-empty set $\mathcal{B}$ that is either a subset of $X \cap \mathcal{A}_1$ or a subset of $X \cap \mathcal{A}_2$ and such that there are no edges from $\mathcal{B}$ to $X \setminus \mathcal{B}$. We will show that $I \setminus \mathcal{B}$ satisfies

$$F^I \wedge \bigwedge_{p \in I \setminus \mathcal{A}_1} p, \tag{9.9}$$

which contradicts the assumption that I is an $\mathcal{A}_1$-stable model of F. Since $I \setminus X$ satisfies the first conjunctive term of (9.9), by Proposition 9 so does $I \setminus \mathcal{B}$. Assume, for instance, that $\mathcal{B}$ is a subset of $X \cap \mathcal{A}_1$. Then $\mathcal{B}$ is a subset of $\mathcal{A}_1$, so that $I \setminus \mathcal{A}_1$ is a subset of $I \setminus \mathcal{B}$. We may conclude that $I \setminus B$ satisfies the second conjunction term of (9.9) as well. $\square$

9.6.6 Proof of Theorem 16

The following lemma, analogous to Theorem 3 from Ferraris et al. (2011), is used to prove the infinitary splitting theorem.

Lemma 7 *For any infinitary formulas F, G, if $\mathcal{A}$ is disjoint from $P(G)$ then I is an $\mathcal{A}$-stable model of $F \wedge G$ iff it is an $\mathcal{A}$-stable model of F and satisfies G.*

Proof. $\Leftarrow$: Assume I is an $\mathcal{A}$-stable model of F and I satisfies G. Since I satisfies G it satisfies G^I. Since I is an $\mathcal{A}$-stable model of F, it is a minimal w.r.t. $\leq_{\mathcal{A}}$ among the models of F, and consequently among the models of $F \wedge G$.

$\Rightarrow$: Assume I is an $\mathcal{A}$-stable model of $F \wedge G$. Then I is a minimal model of $(F \wedge G)^I$ w.r.t. $\leq_{\mathcal{A}}$. So I satisfies $F \wedge G$ and therefore satisfies G. It remains to show that there is no proper subset J of I such that $I \setminus J \subseteq \mathcal{A}$ and J satisfies F^I. Assume that there is some such J. Then J must not satisfy G^I. (If it did, then I would not be minimal with respect to $\leq_{\mathcal{A}}$ among the models of $(F \wedge G)^I$.) Let $\mathcal{A}'$ denote $I \setminus J$.

Since $\mathcal{A}$ is disjoint from $\mathrm{P}(G)$, so is $\mathcal{A}'$. So by Proposition 7, $I \setminus \mathcal{A}' = J$ must satisfy G^I. Contradiction. $\square$

Theorem 16 *Let F, G be infinitary formulas, and $\mathcal{A}_1, \mathcal{A}_2$ be disjoint sets of atoms such that the partition $\{\mathcal{A}_1, \mathcal{A}_2\}$ is infinitely separable on $DG_{\mathcal{A}_1 \cup \mathcal{A}_2}[F \wedge G]$. If $\mathcal{A}_2$ is disjoint from $P(F)$, and $\mathcal{A}_1$ is disjoint from $P(G)$, then for any interpretation I, I is an $\mathcal{A}_1 \cup \mathcal{A}_2$-stable model of $F \wedge G$ iff it is both an $\mathcal{A}_1$-stable model of F and an $\mathcal{A}_2$-stable model of G.*

Proof. Let F, G be infinitary formulas and $\mathcal{A}_1, \mathcal{A}_2$ be disjoint sets of atoms such that the partition $\{\mathcal{A}_1, \mathcal{A}_2\}$ is infinitely separable on $\mathrm{DG}_{\mathcal{A}_1 \cup \mathcal{A}_2}[F \wedge G]$ and the other conditions of the infinitary splitting theorem hold. By the infinitary splitting lemma, I is an $\mathcal{A}_1 \cup \mathcal{A}_2$-stable model of $F \wedge G$ iff it is both an $\mathcal{A}_1$-stable model and an $\mathcal{A}_2$-stable model of $F \wedge G$. Since $\mathcal{A}_2$ is disjoint from $\mathrm{P}(F)$, by Lemma 7, I is an $\mathcal{A}_2$-stable model of $F \wedge G$ iff it is an $\mathcal{A}_2$-stable model of G and it satisfies F. Similarly, I is an $\mathcal{A}_1$-stable model of $F \wedge G$ iff it is an $\mathcal{A}_1$-stable model of F and it satisfies G. It remains to observe that if I is an $\mathcal{A}_2$-stable model of F then it satisfies F, and similarly if I is an $\mathcal{A}_1$-stable model of G. $\square$

9.6.7 Proof of Theorem 17

Lemma 8 *If all atoms that occur in F belong to $\mathcal{A}$ then, for any interpretation I, I is an $\mathcal{A}$-stable model of F iff $I \cap \mathcal{A}$ is a stable model of F.*

Proof. If all atoms that occur in F belong to $\mathcal{A}$ then

$$F^{I\cap\mathcal{A}} \wedge \bigwedge_{p\in I\setminus(I\cap\mathcal{A})} p$$

is identical to (9.2).

Lemma 9 *Let G be a definition for a set $\mathcal{Q}$ of atoms, and let I be a model of G. For any subset K of I such that $K \setminus \mathcal{Q} = I \setminus \mathcal{Q}$, K satisfies G^I iff K satisfies G.*

Proof. We can show that K satisfies a conjunctive term $H \wedge \mathcal{C}^\wedge \rightarrow q$ of G iff K satisfies its reduct $H^I \wedge (\mathcal{C}^\wedge)^I \rightarrow q^I$ as follows:

$K \not\models H^I \wedge (C^\wedge)^I \rightarrow q^I$

iff $K \models H^I,\ K \models (C^\wedge)^I,$ and $K \not\models q^I$

iff $K \models H^I,\ K \models (C^\wedge)^I,$ and $q \notin K$ (because $K \subseteq I$)

iff $I \models H^I,\ K \models (C^\wedge)^I,$ and $q \notin K$ (K and I agree on atoms occurring in H)

iff $I \models H,\ K \models (C^\wedge)^I,$ and $q \notin K$

iff $K \models H,\ K \models (C^\wedge)^I,$ and $q \notin K$ (K and I agree on atoms occurring in H)

iff $K \models H,\ C \subseteq K,$ and $q \notin K$ ($K \subseteq I$)

iff $K \not\models H \wedge C^\wedge \rightarrow q.$ □

Lemma 10 *Let G be a definition for a set $\mathcal{Q}$ of atoms. For any set J of atoms disjoint from $\mathcal{Q}$ there exists a unique $\mathcal{Q}$-stable model I of G such that $I \setminus \mathcal{Q} = J$.*

Proof. Let I be the intersection of all models K of G such that $K \setminus \mathcal{Q} = J$. We will show first that I satisfies G. Assume otherwise, and take a conjunctive term $H \wedge C^\wedge \to q$ of G that is not satisfied by I. Then I satisfies H, $C \subseteq I$, and $q \notin I$. By the choice of I, it follows that there is a model K of G such that $K \setminus Q = J$ and $q \notin K$. On the other hand, since I satisfies H and does not differ from K on atoms occurring in H, K satisfies H. Since $C \subseteq I \subseteq K$, K satisfies $C^\wedge$. Hence K does not satisfy one of the conjunctive terms of G, which is a contradiction. Thus I is a model of G, and consequently a model of G^I. To prove that it is $\mathcal{Q}$-stable, consider any model K of G^I such that $K \leq_{\mathcal{Q}} I$. By Lemma 9, K is also a model G. By the choice of I, it follows that $I \subseteq K$. Consequently $K = I$. It remains to show that I is unique. Let K be a $\mathcal{Q}$-stable model of G such that $K \setminus Q = J$. It is easy to see that $I \subseteq K$. Furthermore, K satisfies G^K and I satisfies G, so by Lemma 9, I satisfies G^K. Since $I \leq_{\mathcal{Q}} K$, it follows that $I = K$.

Theorem 17 *For any infinitary formula F, any set $\mathcal{Q}$ of atoms that do not occur in F, and any definition G for $\mathcal{Q}$, the map $I \mapsto I \setminus \mathcal{Q}$ is a 1-1 correspondence between the stable models of $F \wedge G$ and the stable models of F.*

Proof. Let σ denote the set of all atoms occurring in $F \wedge G$. Since atoms from $\mathcal{Q}$ do not occur in F and $\mathrm{P}(G) \subseteq \mathcal{Q}$, there are no edges from $\sigma \setminus \mathcal{Q}$ to $\mathcal{Q}$ in $\mathrm{DG}_\sigma[F \wedge G]$. Consequently the partition $\{\sigma \setminus \mathcal{Q}, \mathcal{Q}\}$ is infinitely separable on this graph. By the splitting theorem for infinitary formulas, an interpretation I is a stable model of $F \wedge G$ iff it is a $(\sigma \setminus \mathcal{Q})$-stable model of F and a $\mathcal{Q}$-stable model of G. Consider a stable model I of $F \wedge G$. We have seen that I is a $(\sigma \setminus \mathcal{Q})$-stable model of F. By Lemma 8, it follows that $I \setminus \mathcal{Q}$ is a stable model of F. Consider now a stable model J of F, and let S be the set of all interpretations I such that $J = I \setminus \mathcal{Q}$. We will show that S contains exactly one stable model of $F \wedge G$, or equivalently, that there is exactly one interpretation that is a $(\sigma \setminus \mathcal{Q})$-stable model of F and a $\mathcal{Q}$-stable model of G in S. By Lemma 8, any interpretation in S is a $(\sigma \setminus \mathcal{Q})$-stable model of F. By Lemma 10, S contains exactly one $\mathcal{Q}$-stable model of G. $\square$

Chapter 10

An Example of Proving Program Correctness: n-Queens

In this chapter, we prove the correctness of program K shown in Table 4.1. We show how the machinery and methods developed in the previous chapters can be used in application to this particular program. Although here we give here only a single concrete example, we believe that these methods should generalize and be useful for reasoning about the correctness of many ASP programs.

10.1 Formalizing Correctness for K

As described in the introduction, the n-queens problem involves placing n queens on an $n \times n$ chess board such that no two queens threaten each other. We represent squares by pairs of integers (i, j) where $1 \leq i, j \leq n$. Two squares (i_1, j_1) and (i_2, j_2) are said to be in the same row if $i_1 = i_2$; in the same column if $j_1 = j_2$; and in the same diagonal if $|i_1 - i_2| = |j_1 - j_2|$. A set Q of n squares is a *solution* to the n-queens problem if no two elements of Q are in the same row, in the same column, or in the same diagonal.

In the following we identify an atom of the form $q(\overline{i}, \overline{j})$ with the square (i, j).

Theorem 18 *A set of squares is a solution to the n-queens problem iff it is a stable model of K.*

10.2 Applying τ to Program K

Table 10.1 shows the result of applying τ to program K, after applying strongly equivalent transformations justified by the theorems in Chapter 8. The singleton set (10.1) is the literal result of applying τ to rule R_1 from Table 4.1. The sets of formulas in the subsequent lines of Table 10.1 have been simplified. (In each of these formulas A denotes the set of all precomputed terms.)

$$\left\{ \bigwedge_{1\le i,j\le n} \big(q(\bar{i},\bar{j}) \vee \neg q(\bar{i},\bar{j})\big) \right\} \tag{10.1}$$

$$\left\{ \neg\neg \left(\left(\bigwedge_{\substack{\Delta\subseteq A\\ |\Delta|=2}} \neg \bigwedge_{s\in\Delta} q(\bar{i}, s) \right) \wedge \left(\bigvee_{\substack{\Delta\subseteq A\\ |\Delta|=1}} \bigwedge_{s\in\Delta} q(\bar{i}, s) \right) \right) : 1 \le i \le n \right\} \tag{10.2}$$

$$\left\{ \neg\neg \left(\left(\bigwedge_{\substack{\Delta\subseteq A\\ |\Delta|=2}} \neg \bigwedge_{s\in\Delta} q(s, \bar{j}) \right) \wedge \left(\bigvee_{\substack{\Delta\subseteq A\\ |\Delta|=1}} \bigwedge_{s\in\Delta} q(s, \bar{j}) \right) \right) : 1 \le j \le n \right\} \tag{10.3}$$

$$\left\{ \neg\neg \left(\bigwedge_{\substack{\Delta\subseteq A\times A\\ |\Delta|=2}} \neg \bigwedge_{\substack{i,j\in\Delta\\ i-j+n=k}} q(\bar{i},\bar{j}) \right) : 1 \le k \le 2n-1 \right\} \tag{10.4}$$

$$\left\{ \neg\neg \left(\bigwedge_{\substack{\Delta\subseteq A\times A\\ |\Delta|=2}} \neg \bigwedge_{\substack{i,j\in\Delta\\ i+j-1=k}} q(\bar{i},\bar{j}) \right) : 1 \le k \le 2n-1 \right\} \tag{10.5}$$

Table 10.1: The result of applying τ to program K.

For example, the result of applying τ to rule R_2 is a set of formulas corresponding to the instances of that rule, which were computed in Section 5.3. The result of applying τ to an instance (5.3) can be simplified, first by applying Theorem 13 to get (8.3); then by applying Theorems 14 and 15 to the first and second conjunctive terms of (8.3), respectively, and applying the observation at the end of Section 8.3 to get (8.6). The set of formulas (10.2) from Table 10.1 is the set of instances of R_2, simplified in this way.

10.3 Proof of Theorem 18

A *constraint* is an infinitary formula of the form $\neg F$ (shorthand for $F \rightarrow \bot$).[1] The following proposition, which is used to prove Theorem 18 is a straightforward generalization of Proposition 4 from (Ferraris, 2005).

Proposition 11 *Let $\mathcal{H}_1$ be an infinitary formula $\mathcal{H}_2$ be a set of constraints. A set I of atoms is a stable model of $\mathcal{H}_1 \cup \mathcal{H}_2$ iff I is a stable model of $\mathcal{H}_1$ and satisfies all formulas in $\mathcal{H}_2$.*

Proof. *Case 1:* Every formula in $\mathcal{H}_1 \cup \mathcal{H}_2$ is satisfied by I. For each formula $\neg F$ in $\mathcal{H}_2$, I does not satisfy F. So the reduct of each formula in $\mathcal{H}_2$ w.r.t. I is $\neg\bot$. It follows that the set of reducts of all formulas in $\mathcal{H}_1 \cup \mathcal{H}_2$ is satisfied by the same interpretations as the set of reducts of all formulas in $\mathcal{H}_1$. Consequently, I is minimal among the sets satisfying the reducts of all formulas from $\mathcal{H}_1 \cup \mathcal{H}_2$ iff it

[1]This terminology is also used for an ASP rule with an empty head (see Section 2.3), which is consistent with the use here in view of the fact that applying τ to such a rule yields a constraint in our sense.

is minimal among the sets satisfying the reducts of all formulas from $\mathcal{H}_1$. *Case 2:* Some formula F in $\mathcal{H}_1 \cup \mathcal{H}_2$ is not satisfied by I. Then I is not a stable model of $\mathcal{H}_1 \cup \mathcal{H}_2$. If $F \in \mathcal{H}_1$ then I is not a stable model of $\mathcal{H}_1$. Otherwise, it is not true that I satisfies all formulas in $\mathcal{H}_2$. $\square$

Lemma 11 *Interpretation I satisfies* (10.2) *iff for all $i \in \{1, \dots, n\}$, I contains exactly one atom of the form $q(\bar{i}, \bar{j})$.*

Proof. Consider the conjunctive terms of a formula from (10.2). Note that I satisfies

$$\bigwedge_{\substack{\Delta \subseteq A \\ |\Delta|=2}} \neg \bigwedge_{s \in \Delta} q(\bar{i}, s)$$

iff it contains at most one atom of the form $q(\bar{i}, s)$. On the other hand, I satisfies

$$\bigvee_{\substack{\Delta \subseteq A \\ |\Delta|=1}} \bigwedge_{s \in \Delta} q(\bar{i}, s)$$

iff it contains at least one atom of the form $q(\bar{i}, s)$. Since I is a set of squares, s in this atom is one of $\overline{1}, \dots, \overline{n}$. $\square$

The proof of the following lemma is similar to that of the previous one.

Lemma 12 *Interpretation I satisfies* (10.3) *iff for all $j \in \{1, \dots, n\}$, I contains exactly one atom of the form $q(\bar{i}, \bar{j})$.*

Lemma 13 *Interpretation I satisfies* (10.4) *and* (10.5) *iff no two squares in I are in the same diagonal.*

Proof. First note that two squares $(\overline{i_1}, \overline{j_1}), (\overline{i_2}, \overline{j_2})$ are in the same diagonal iff there exists a $k \in \{1, \dots, 2n-1\}$ such that

$$i_1 - j_1 + n = k \text{ and } i_2 - j_2 + n = k, \tag{10.6}$$

or

$$i_1 + j_1 - 1 = k \text{ and } i_2 + j_2 - 1 = k. \tag{10.7}$$

It is easy to see that a set of squares I does not satisfy (10.4) iff there exists a k such that (10.6) holds for two distinct elements $q(\overline{i_1}, \overline{j_1}), q(\overline{i_2}, \overline{j_2}) \in I$, and that it does not satisfy (10.5) iff there exists a k such that (10.7) holds for two such elements. □

Theorem 18. *A set of squares is a solution to the n-queens problem iff it is a stable model of K.*

Proof. It is easy to see that a set of atoms is a stable model of (10.1) iff it is a set of squares. This can be established directly by reasoning about the reduct of (10.1). Let $\mathcal{H}_2$ be the union of sets (10.2)–(10.5). All formulas in $\mathcal{H}_2$ are constraints. So by Proposition 11, I is a stable model of τK iff it is a stable model of (10.1) and satisfies all formulas in $\mathcal{H}_2$. By Lemmas 11–13, a set I satisfies all formulas in $\mathcal{H}_2$ iff it is a solution to the n-queens problem. □

Chapter 11

An Example of Proving Program Correctness for a Program with Auxiliary Predicates: Optimized n-Queens

The proof of correctness for program K in Chapter 10 is made particularly simple by the fact that a single predicate (other than equality) appears in that program, namely the binary predicate q. Often, ASP programs include many predicates, and moreover many predicates are commonly auxiliary in the sense that their extents are not relevant to the solution. Table 11.1 shows an optimized ASP solution to the n-queens problem encoded in AG. It is similar to an optimized version solution to that problem presented in (Gebser et al., 2011). In that paper, the authors show that using an optimized encoding with precalculated diagonals improves efficiency so much that the problem can be solved for values of n as large as 1000. More naive encodings, like that in Table 1.1, can only be used to solve the problem for values of n up to about 100 on the same machine.

The program in Table 11.1 contains three distinct predicate symbols: the binary predicate q, as well as the ternary predicates $d1$ and $d2$, which precompute labels corresponding the diagonals for each square. The atom $d1(\overline{1}, \overline{1}, \overline{8})$ for example expresses that the square $(1, 1)$ occurs in the diagonal labelled with integer 8. In order to prove the correctness of this, more complicated program, in addition to the

```
% place queens on the chess board
{ q(1..n,1..n) }.

% exactly 1 queen per row/column
:- X = 1..n, not #count{ Y : q(X,Y) } = 1.
:- Y = 1..n, not #count{ X : q(X,Y) } = 1.

% pre-calculate the diagonals
d1(X,Y,X-Y+n) :- X = 1..n, Y = 1..n.
d2(X,Y,X+Y-1) :- X = 1..n, Y = 1..n.

% at most one queen per diagonal
:- D = 1..n*2-1, #count{ X,Y : q(X,Y), d1(X,Y,D) } >= 2.
:- D = 1..n*2-1, #count{ X,Y : q(X,Y), d2(X,Y,D) } >= 2.
```

Table 11.1: An optimized GRINGO encoding of the n-queens problem.

methods used in the previous chapter, we also use the results on symmetric splitting from Chapter 9.

We show how we can prove the correctness of the program in Table 11.1. Table 11.2 shows the result of representing the program in Table 11.1 using AG syntax. We will call this program K'.

11.1 Formalizing Correctness for K'

As in Section 10.1, we represent squares by pairs of integers (i, j) where $1 \leq i, j \leq n$, and identify an atom of the form $q(\bar{i}, \bar{j})$ with the square (i, j). For any stable model I of K', by Q_I we denote the set of pairs (i, j) such that $q(\bar{i}, \bar{j}) \in I$. Because K' contains auxiliary predicates, the statement of the program's correct-

$\{q(\overline{1}..\overline{n}, 1..\overline{n})\}$	R'_1
$\leftarrow X = \overline{1}..\overline{n} \wedge \textit{not count}\{Y : q(X,Y)\} = \overline{1}$	R'_2
$\leftarrow Y = \overline{1}..\overline{n} \wedge \textit{not count}\{X : q(X,Y)\} = \overline{1}$	R'_3
$\textit{d1}(X,Y,X-Y+\overline{n}) \;\leftarrow\; X = \overline{1}\,..\,\overline{n} \;\wedge\; Y = \overline{1}\,..\,\overline{n},$	R'_4
$\textit{d2}(X,Y,X+Y-\overline{1}) \;\leftarrow\; X = \overline{1}\,..\,\overline{n} \;\wedge\; Y = \overline{1}\,..\,\overline{n}$	R'_5
$\leftarrow\; D = \overline{1}\,..\,\overline{n} \times \overline{2} - \overline{1} \;\;\wedge\;\; \textit{count}\{X,Y : q(X,Y), \textit{d1}(X,Y,D)\} \geq \overline{2}$	R'_6
$\leftarrow\; D = \overline{1}\,..\,\overline{n} \times \overline{2} - \overline{1} \;\;\wedge\;\; \textit{count}\{X,Y : q(X,Y), \textit{d2}(X,Y,D)\} \geq \overline{2}$	R'_7

Table 11.2: Program K': An optimized AG encoding of the n-queens problem.

ness posits the existence of a 1-1 correspondence between stable models and solutions to the n-queens problem.

Theorem 19 (Gebser et al., 2015) *For each stable model I of K', Q_I is a solution to the n-queens problem; and for each solution Q to the n-queens problem there is exactly one stable model I of K' such that $Q_I = Q$.*

11.2 Proof of Theorem 19

By D_n we denote the set of all atoms of the forms $\textit{d1}(\overline{i}, \overline{j}, \overline{i-j+n})$ and $\textit{d2}(\overline{i}, \overline{j}, \overline{i+j-1})$ for all i, j from $\{1, \ldots, n\}$. Recall that the rules of the program K' are denoted by $R'_1, \ldots, R'_7$.

Lemma 14 *A set of atoms is a stable model of*

$$\tau R'_1 \cup \tau R'_4 \cup \tau R'_5 \tag{11.1}$$

iff it is of the form $Q \cup D_n$ where Q is a set of squares.

Proof. The result of applying τ to R'_1 is (10.1). Let F be the conjunction of these formulas. The set $\tau R'_4$ is strongly equivalent to the set of formulas

$$\top \to d1(\bar{i}, \bar{j}, \overline{i-j+n}) \tag{11.2}$$

$(1 \leq i, j \leq n)$. (We take into account that $\tau(\bar{i} = \bar{1}..\bar{n})$ is equivalent to $\top$ if $1 \leq i \leq n$ and to $\bot$ otherwise, and similarly for j.) Similarly, $\tau R'_5$ is strongly equivalent to the set of formulas

$$\top \to d2(\bar{i}, \bar{j}, \overline{i+j-1}) \tag{11.3}$$

$(1 \leq i, j \leq n)$. Let G be the conjunction of formulas (11.2), and (11.3). Then (11.1) is strongly equivalent to $F \wedge G$. Let Q_n be the set of all squares. If all atoms are considered intensional, the dependency graph of (11.1) contains all atoms from $Q_n \cup D_n$ and no edges. So the partition $\{Q_n, D_n\}$ is infinitely separable on this graph. It is easy to check that P$(F) = Q_n$ and P$(G) = D_n$ so that Q_n is disjoint from P(G), and D_n is disjoint from P(F). Then by Theorem 16, I is a stable model of (11.1) iff it is a Q_n-stable model of F, and a D_n-stable model of G. By Proposition 3(ii), I is a Q_n-stable model of F iff it is a minimal model of

$$F^I \wedge \bigwedge_{p \in I \setminus Q_n} p. \tag{11.4}$$

It is easy to check, by reasoning directly about the reduct F^I that I is a minimal model of (11.4) iff it is of the form

$$Q \cup D, \tag{11.5}$$

where Q is a set of squares and $D \subseteq D_n$, is a Q_n-stable model of F. Also by Proposition 3(ii), I is a D_n-stable model of G iff it is a minimal model of

$$G^I \wedge \bigwedge_{p \in I \setminus D_n} p. \tag{11.6}$$

Again, we can reason directly about the reduct G^I to see that I is a minimal model of (11.6) iff it is of the form

$$D_n \cup Q, \tag{11.7}$$

where Q is a set of squares, is a D_n-stable model of G. The intersection of (11.5) and (11.7) is the set of all interpretations $Q \cup D_n$ where Q is a set of squares. $\square$

Lemma 15 *A set I of atoms is a stable model of $\tau K'$ iff it has the form $Q \cup D_n$, where Q is a solution to the n-queens problem.*

Proof. Let $\mathcal{H}_1$ be (11.1) and $\mathcal{H}_2$ be

$$\tau R_2' \cup \tau R_3' \cup \tau R_6' \cup \tau R_7'.$$

All formulas in $\mathcal{H}_2$ are constraints. Consequently, by Proposition 11, I is a stable model of $\tau K'$ iff it is a stable model of $\mathcal{H}_1$ and satisfies all formulas in $\mathcal{H}_2$. By

Lemma 14, I is a stable model of $\mathcal{H}_1$ iff it is of the form $Q \cup D_n$, where Q is a set of squares. It remains to show that a set I of the form $Q \cup D_n$ satisfies all formulas in $\mathcal{H}_2$ iff Q is a solution to the n-queens problem. Specifically, we will show that for any set I of the form $Q \cup D_n$

(i) I satisfies τR_2 iff for all $i \in \{1, \dots, n\}$, I contains exactly one atom of the form $q(\bar{i}, \bar{j})$;

(ii) I satisfies τR_3 iff for all $j \in \{1, \dots, n\}$, I contains exactly one atom of the form $q(\bar{i}, \bar{j})$;

(iii) I satisfies $\tau R_6 \cup \tau R_7$ iff no two squares in I are in the same diagonal.

Since $R_2' = R_2$ and $R_3' = R_3$, (i) and (ii) are established by Lemmas 11 and 12. To prove (iii), note that two squares $(\overline{i_1}, \overline{j_1}), (\overline{i_2}, \overline{j_2})$ are in the same diagonal iff there exists a $k \in \{1, \dots, 2n-1\}$ such that

$$d1(\bar{i}_1, \bar{j}_1, \bar{k}), d1(\bar{i}_2, \bar{j}_2, \bar{k}) \in D_n \tag{11.8}$$

or

$$d2(\bar{i}_1, \bar{j}_1, \bar{k}), d2(\bar{i}_2, \bar{j}_2, \bar{k}) \in D_n. \tag{11.9}$$

We will show that a set I of the form $Q \cup D_n$ does not satisfy $\tau R_6'$ iff there exists a k such that (11.8) holds for two distinct elements $q(\overline{i_1}, \overline{j_1}), q(\overline{i_2}, \overline{j_2}) \in Q$, and that it does not satisfy $\tau R_7'$ iff there exists a k such that (11.9) holds for such two elements.

The result of applying τ to R_6' is strongly equivalent to the set of formulas

$$\neg\tau(\mathit{count}\{X,Y : q(X,Y), \mathit{d1}(X,Y,\overline{k})\} \geq \overline{2}) \tag{11.10}$$

$(1 \leq k \leq 2n-1)$. In view of Theorem 11 and the fact that $[\overline{2}]$ is a singleton, it follows that it is strongly equivalent to

$$\neg \bigvee_{\substack{\Delta\subseteq A\\ |\Delta|=2}} \bigwedge_{(1,(r,s))\in\Delta} (q(r,s) \wedge d1(r,s,\overline{k}))$$

$(1 \leq k \leq 2n-1)$. This formula can be written as

$$\neg \bigvee_{\substack{\Sigma\subseteq P\times P\\ |\Sigma|=2}} \bigwedge_{(r,s)\in\Sigma} (q(r,s) \wedge d1(r,s,\overline{k})). \tag{11.11}$$

For any set Q of squares,

$Q \cup D_n$ does not satisfy (11.11)

iff there exist two distinct pairs $(r_1,s_1),(r_2,s_2)$ from $P \times P$ such that $q(r_1,s_1), q(r_2,s_2) \in Q$ and $d1(r_1,s_1,\overline{k}), d1(r_2,s_2,\overline{k}) \in D_n$

iff there exist two distinct squares $(\overline{i}_1,\overline{j}_1),(\overline{i}_2,\overline{j}_2) \in Q$ such that (11.8) holds.

The claim about (11.9) is proved in a similar way. □

Theorem 19 *For each stable model I of K', Q_I is a solution to the n-queens problem; and for each solution Q to the n-queens problem there is exactly one stable model I of K' such that $Q_I = Q$.*

Theorem 19 is immediate from the lemmas.

Chapter 12

Correcting the ASPCORE2 Semantics

The ASPCORE2 document (Calimeri et al., 2012), which describes a syntax and semantics for ASP languages, was intended as a specification for the behavior of answer set programming (ASP) systems. As we have argued to justify the AG semantics, the existence of such a specification is important because it allows us to reason about the correctness of programs. The ASPCORE2 authors were further interested in a formal specification because it enables system comparisons and competitions to evaluate different ASP systems. The definition of the semantics given in that document is based on the FLP semantics for logic programs (Faber et al., 2004), but contains an oversight which limits its application to programs without local variables. We propose a correction to that oversight. Our proposal applies to programs in the input language of ASPCORE2 that do no contain classical negation, weak constraints, or queries. We suggest that local variables can be accommodated by employing the same translation τ from programs to sets of infinitary formulas. We show that for ASPCORE2 programs without local variables, the stable models according to our proposed semantics are the same as the stable models prescribed by the ASPCORE2 document.

12.1 ASPCORE2 and Local Variables

The definition from Calimeri et al. (2012) was meant to apply to arbitrary programs, but it contained an oversight. According to that semantics, local variables are eliminated in the process of forming instances by substituting tuples of precomputed terms. The following passage from that paper explains this process in detail.

> Given a collection $\{e_1; \ldots; e_n\}$ of aggregate elements, the *instantiation* of $\{e_1; \ldots; e_n\}$ is the following set of aggregate elements:
>
> $$\textit{inst}(\{e_1; \ldots; e_n\}) = \cup_{1 \leq i \leq n} \{e_i\sigma \mid \sigma \text{ is a well-formed substitution for } e_i\}$$
>
> A ground instance of a rule, weak constraint or query r is obtained in two steps: (1) a well-formed global substitution σ for r is applied to r; (2) for every aggregate atom #aggr$\{e_1; \ldots; e_n\} \prec u$ appearing in $r\sigma$, $e_1; \ldots; e_n$ is replaced by *inst*$(e_1; \ldots; e_n)$ (where aggregate elements are syntactically separated by ";").

Stable models are then defined for ground programs. The issue is that, typically, there are infinitely many well-formed[1] substitutions for an aggregate element, because arbitrary numerals can be substituted for a variable. So that *inst*$(\{e_1; \ldots; e_n\})$ is usually an infinite set, and cannot be part of a rule, which is a finite syntactic object.

[1]In this context, a substitution for a particular expression is well-formed if it results in an expression for which all arithmetic subterms–those involving symbols (4.1)–are well-defined.

For example, consider the rule (1.1). This is rule in the ASPCORE2 input language as well the GRINGO input language. The variable X is local in this rule. Any substitution of a precomputed term r for X is well-formed, so that the set *inst*$(\{X : p(X)\})$ contains the aggregate element $r : p(r)$ for every precomputed term r. Then the result of replacing `X : p(X)` in (1.1) by *inst*$(\{X : p(X)\})$ is not a valid rule. Indeed, this problem arises for any rule that contains local variables.

12.2 Syntax of Programs in ASPCORE2

We restrict our attention to a subset of the ASPCORE2 input language, which we call AC2. It is a proper subset of the input language from Calimeri et al. (2012): classical negation, weak constraints, and queries are all constructs available in input language defined by Calimeri et al. (2012), but not in AC2.

It is convenient to define the syntax of AC2 as a subset of the syntax of AG (see Chapter 4). By an AC2 program we mean an AG program that satisfies each of the following conditions:

- the program contains neither intervals nor choice expressions;
- the only aggregate names that are used are $\{$*count*, *sum*, *max*, or *min*$\}$;
- in every rule (4.9) in the program, each element H_i in the head of the rule is an atom.

12.3 Semantics for AC2 Programs with Local Variables

In this section, we propose a definition for the semantics of AC2 programs that is applicable to AC2 programs with local variables. Our proposal is to use τ as defined in Chapter 5 to translate from closed AC2 programs to sets of infinitary formulas of a particular form. We can then define the stable models of an AC2 program with local variables using a generalization of the FLP-stable models (Faber et al., 2004) that applies to such sets.

The definition of the FLP-reduct given in this section applies only to sets of infinitary formulas of the form $G \to H$, where H is a disjunction of atoms. Since any AC2 program contains only rules (4.9) and only atoms in the head, it is easy to check that all formulas in $\tau\Pi$ are of this form.

The *FLP-reduct* of an infinitary formula $G \to H$ w.r.t. an interpretation I is $G \to H$ if I satisfies G, and $\top$ otherwise. An interpretation I is an *FLP-stable model* of a set $\mathcal{H}$ of formulas of the form $G \to H$ if it is minimal w.r.t. set inclusion among the interpretations satisfying the FLP-reducts of all formulas from $\mathcal{H}$.

Since the FLP-reduct of a formula $G \to H$ with respect to I depends only on whether or not I satisfies G classically, we can simplify the definition of τ when applied to an aggregate literal (see Section 5.4) without affecting the FLP-stable models of a program. This observation allows us to introduce an alternative, more intuitive translation function for translating aggregates in the context of the FLP semantics. We will call this translation function τ^*. If E is a closed aggregate atom and t is a precomputed term, let $\tau^*_t E$ be the disjunction of formulas

$$\bigwedge_{\mathbf{r}\in\Delta} \tau(\mathbf{L}^{\mathbf{x}}_{\mathbf{r}}) \wedge \bigwedge_{\mathbf{r}\in A\setminus\Delta} \neg\tau(\mathbf{L}^{\mathbf{x}}_{\mathbf{r}}) \tag{12.1}$$

over the subsets Δ of A that *do* justify E with respect to t, where A is defined as in Section 5.4. It is easy to check that for any aggregate literal E and precomputed term t, $\tau^*_t E$ is classically equivalent to $\tau_t E$. Indeed, if D is the disjunction of formulas (12.1) over all subsets Δ of A that *do not* justify E with respect to t, the equivalence between D and $\neg\tau^*_t E$ is clear. It is also clear that D is classically equivalent to $\tau_t E$. The fact that $\tau^*_t E$ is classically equivalent to $\tau_t E$ justifies the use of τ^*_t in place of τ_t to translate aggregate literals in the context of the FLP semantics. In application to other literals, rules, and programs, the definition of τ^* is the same as τ.

12.4 Review: AC2-Stable Models

This section contains a review of the definition of a stable model for programs given by Calimeri et al. (2012). The definition of the semantics that we review here applies to AC2 programs without local variables.

A ground term t will be called *ill-defined* if $[t]$ is $\emptyset$ (For example, $\overline{1} + s$, and $\overline{5}\ /\ \overline{0}$ are both ill-defined.) We will define the semantics for any AC2 program that does not contain local variables.

Interpretations in the sense of AC2 are the same as in AG: subsets of the set of atoms of the form $p(\mathbf{t})$ where p is a symbolic constant and $\mathbf{t}$ is a tuple of precomputed terms.[2] *Instances* in the sense of the AC2 semantics are all rules that can be formed by substituting precomputed terms for global variables in such a way that no term outside of aggregate elements in the resulting rule is ill-defined. Clearly, each instance of a AC2 rule in this sense is also an instance in the sense of AG.

The semantics of terms in AC2 is the same as in Section 5.1. Because intervals are not allowed and AC2 instances do not contain ill-defined terms outside of aggregate elements, for any occurrence of a term t outside of aggregate elements in an instance of a rule, $[t]$ is a singleton. Similarly, for any occurrence of a tuple of terms $\mathbf{t}$ or atom $p(\mathbf{t})$ outside of aggregate elements, $[\mathbf{t}]$ is a singleton and $[p(\mathbf{t})]$ is $p(\mathbf{t})$. For this reason, in this chapter we will identify $[t]$ with its unique element, and similarly for $[\mathbf{t}]$ and $[p(\mathbf{t})]$, when appropriate.

A ground arithmetic or symbolic literal L is satisfied by an interpretation I if τL is satisfied by I; (see Section 5.2). If I is an interpretation and $\mathbf{t} : \mathbf{L}$ is a ground aggregate element, by $(\mathbf{t} : \mathbf{L})^I$ we denote the set $[\mathbf{t}]$ if all literals in $\mathbf{L}$ are satisfied by I, and the empty set otherwise.

An interpretation I satisfies

- an aggregate atom (4.7) if the relation $\prec$ holds between s and $\widehat{\alpha}(\mathbf{t} : \mathbf{L})^I$;
- an aggregate literal *not* A if it does not satisfy A; and
- a ground rule (4.9) if it satisfies some atom in the head or does not satisfy some literal in the body.

[2]Here we understand the set of precomputed terms as in Section 4.1. The understanding in (Calimeri et al., 2012) is slightly different. In that document, there is a distinction between symbolic constants, and predicate constants, while in AG (and AC2) no such distinction exists.

The *AC2-reduct* of a program Π with respect to an interpretation I is the program consisting of all instances R of rules from Π such that I satisfies the body of R. An interpretation I is an $AC2$*-stable model* of Π if I is a subset minimal model of the AC2-reduct.

Proposition 12 *Let Π be an AC2 program without local variables. Then I is an AC2-stable model of Π iff it is an FLP-stable model of $\tau^*\Pi$.*

This proposition supports the proposal to use FLP-stable models and τ^* to define the semantics of ASPCORE2.

12.5 Proof of Proposition 12

The following proposition clarifies the correspondence between AC2 and AG instances.

Lemma 16 *Let R be an AC2 rule without local variables and let R' be an instance of R in the sense of AG. If R' contains a term t that is ill-defined appearing outside of aggregate elements, then τ^*R' is strongly equivalent to $\top$.*

Proof. Since R is an AC2 rule, recall that it is of the form (4.9), where the head is a disjunction of atoms. Consider the ill-defined term t that occurs in R'. First, we consider the case when s is ill-defined in an aggregate atom E of the form (4.7) in the body of a rule. In this case, the result of applying τ^* to the aggregate literal containing E is $\bot$, independent of whether or not E is in the scope of negation-as-failure, so $\bot$ is one of the conjunctive terms in the antecedent of τ^*R'. Next, we consider the case when t occurs either in the head of R' or in a symbolic or arithmetic literal in the body. If t occurs in an atom $p(\mathbf{t})$ in the head of R', then $[p(\mathbf{t})]^\wedge$ is $\emptyset^\wedge$ which is $\top$, so that $\top$ is one of the disjunctive terms in the consequent of τ^*R'. If t occurs in an arithmetic literal $t_1 \prec t_2$ in the body of R', then $\tau^*(t_1 \prec t_2)$ is $\bot$, so $\bot$ is one of the conjunctive terms in the antecedent of τ^*R'. If t occurs in a symbolic literal $p(\mathbf{t})$ (or *not* $p(\mathbf{t})$) in the body of R', then $[p(\mathbf{t})]$ (respectively, $[$*not* $p(\mathbf{t})]$) is $\emptyset$, so τ^* applied to the symbolic literal is $\emptyset^\vee$ which is $\bot$, so $\bot$ is one of the conjunctive terms in the antecedent of τ^*R'. In each of these cases it is easy to see that τ^*R' is strongly equivalent to $\top$. □

Proposition 12 *Let* Π *be an AC2 program without local variables. Then* I *is an AC2-stable model of* Π *iff it is an FLP-stable model of* $\tau^*\Pi$.

Proof. Let Π be an AC2 program without local variables, and let Π' be the set of all AC2 instances of rules in Π. Each AC2 instance in Π' is also an AG instance of a rule in Π. Furthermore, by Lemma 16, each nontautological formula in $\tau^*\Pi$ is an AC2 instance of a rule from Π. Then it suffices to show that for a ground AC2 rule R that does not contain ill-defined terms outside of aggregate elements,

- I satisfies the body of R iff it satisfies the antecedent of the τ^*R, and
- I satisfies the head of R iff it satisfies the consequent of τ^*R.

Consider such a rule R. For any atom $p(\mathbf{t})$ in the head of R, since $[\mathbf{t}]$ is a singleton, $[p(\mathbf{t})]^\vee = p(\mathbf{t})$. So it is clear that I satisfies an atom $p(\mathbf{t})$ in the head of R iff it satisfies that atom in the consequent of τ^*R. To show that I satisfies the body of R iff it satisfies the antecedent of τ^*R, we need to show that I satisfies a literal L in the body of R iff it satisfies τ^*L. For the cases of arithmetic and symbolic literals, this is easy to check. Consider a ground aggregate atom of the form (4.7) occurring in an aggregate literal L in R. Since R does not contain ill-defined terms outside of aggregate literals, $[s] = \{s\}$, and τ^*L has exactly one disjunctive term: either $\tau^*_s E$ or its negation (depending on whether L is a positive or negative aggregate literal). It is sufficient to show that I satisfies the result of applying τ^*_s to (4.7) iff I satisfies (4.7) in the sense of the AC2 semantics. In other words, we need to verify that the relation $\prec$ holds between $\widehat{\alpha}(\mathbf{t} : \mathbf{L})^I$ and s iff I satisfies the disjunction of formulas (12.1) over all sets Δ that justify (4.7) with respect to s. Since the aggregate element $\mathbf{t} : \mathbf{L}$ is ground, the set A of tuples of precomputed terms the same length as the list of variables in $\mathbf{t} : \mathbf{L}$ is the singleton set $\{\epsilon\}$. Then the possible values for Δ are $\{\epsilon\}$ and $\emptyset$, and formula (12.1) becomes

$$\tau^*\mathbf{L} \wedge \top$$

if Δ is $\{\epsilon\}$, and

$$\top \wedge \neg\tau^*\mathbf{L}$$

if Δ is $\emptyset$. So I satisfies the result of applying τ^*_s to (4.7) iff either

$$\{\epsilon\} \text{ justifies (4.7) with respect to } s \text{ and } I \text{ satisfies } \tau^*\mathbf{L} \tag{12.2}$$

or

$$\emptyset \text{ justifies (4.7) with respect to } s \text{ and } I \text{ does not satisfy } \tau^*\mathbf{L}. \tag{12.3}$$

Condition (12.2) holds iff relation $\prec$ holds between $\widehat{\alpha}[\mathbf{t}]$ and s and $(\mathbf{t} : \mathbf{L})^I$ is $[\mathbf{t}]$.

Condition (12.3) holds iff relation $\prec$ holds between $\widehat{\alpha}\emptyset$ and s and $(\mathbf{t} : \mathbf{L})^I$ is $\emptyset$. $\square$

Chapter 13

Conclusions and Future Work

In this document, we have proposed a semantics for ASP input languages, and argued that our proposal can be viewed as a specification for the behavior of ASP systems. Such a specification is useful because it allows us to evaluate whether or not a system produced "correct" output. Moreover, the process of producing this specification has already lead to changes in the behavior of the CLINGO system. In response to our joint work on this project, the designers of that system have made a number of modifications to it. They have changed aspects of the way partial functions are treated by the system, for example, and changed the treatment of atoms involving "pools" in the body of rules from conjunctive to disjunctive.[1] In this way, the semantics has already been useful in clarifying the behavior of that system in particular.

The existence of a specification for GRINGO also opens the door for work in other directions. A current project investigates how to define program completion (Clark, 1978) for a large class of GRINGO programs (Harrison et al., 2017). We show that for some GRINGO programs there is a precise correspondence between models of the completion and stable models in the sense of AG. We hope that this line of work may serve as the theoretical basis for a software tool to reconstruct program descriptions is a conventional mathematical language from GRINGO

[1]A pool is a syntactic construct that is not covered in this document, but is covered by Gebser et al. (2015).

programs.

We have also provided a number of methods that may be useful for reasoning about programs on the basis of our semantics. This work involved extending many aspects of the existing theory developed for a finite propositional representation of ASP programs to our own infinitary representation. We have extended, for example, the theory of strong equivalence to infinitary formulas and found a deductive system that can be used to establish whether or not two infinitary formulas strongly equivalent. We have also developed methods for using proofs in finite deductive systems to establish the validity of infinitary formulas. We have proved a number of theorems that allow us to simplify the translations of some aggregate atoms based on their syntactic form, and extended the theory of splitting to infinitary formulas as well. Finally, we've given examples illustrating how our proposed semantics, in combination with these methods can be used to prove program correctness.

The idea to use infinitary formulas as a representational formalism for ASP programs has applications outside of our own semantics as well. We have used the same translation as the basis for a semantics that corrects the proposed ASPCORE2 semantics (Calimeri et al., 2012).

In future work, we hope to achieve a better understanding of the relationship between our AG semantics and that of Calimeri et al. (2012). There have been informal claims made about this relationship. For example, the authors of the ASPCORE2 document state that the lack of recursive aggregates in a program ensures "an uncontroversial semantics." More recently, Cabalar et al. (2017) state that while the approach to the semantics of aggregates given by Ferraris (2009) may differ from that of Faber et al. (2010) “when aggregates are in the scope of default negation, they coincide for the rest of the cases.” Our own semantics of aggregates in AG is based on that of Ferraris (2009), while the semantics of aggregates given by Calimeri et al. (2012) is based on that of Faber et al. (2010). We hope to use the AG semantics and our proposed correction to the semantics from Calimeri et al. (2012) to clarify and generalize these claims.

References

Marcello Balduccini, Michael Gelfond, Monica Nogueira, Richard Watson, and Matthew Barry. An A-Prolog decision support system for the Space Shuttle. In *Working Notes of the AAAI Spring Symposium on Answer Set Programming*, 2001.

Gerhard Brewka, Thomas Eiter, and Miroslaw Truszczynski. Answer set programming at a glance. *Communications of the ACM*, 54(12):92–103, 2011.

Daniel R. Brooks, Esra Erdem, Selim T. Erdoğan, James W. Minett, and Donald Ringe. Inferring phylogenetic trees using answer set programming. *Journal of Automated Reasoning*, 39:471–511, 2007.

Pedro Cabalar and Paolo Ferraris. Propositional theories are strongly equivalent to logic programs. *Theory and Practice of Logic Programming*, 7, 2007.

Pedro Cabalar, Jorge Fandinno, Torsten Schaub, and Sebastian Schellhorn. Gelfond-zhang aggregates as propositional formulas. In *Proceedings of International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR)*, 2017.

Francesco Calimeri, Wolfgang Faber, Martin Gebser, Giovambattista Ianni, Roland Kaminski, Thomas Krennwallner, Nicola Leone, Francesco Ricca, and Torsten Schaub. ASP-Core-2: Input language format. Available at `https://www.mat.unical.it/aspcomp2013/files/ASP-CORE-2.0.pdf`, 2012.

Keith Clark. Negation as failure. In Herve Gallaire and Jack Minker, editors, *Logic and Data Bases*, pages 293–322. Plenum Press, New York, 1978.

Marc Denecker. Extending classical logic with inductive definitions. In *LNAI*, pages 703–717. Springer, 2000.

Wolfgang Faber, Nicola Leone, and Gerald Pfeifer. Recursive aggregates in disjunctive logic programs: Semantics and complexity. In *Proceedings of European Conference on Logics in Artificial Intelligence (JELIA)*, 2004.

Wolfgang Faber, Nicola Leone, and Gerald Pfeifer. Semantics and complexity of recursive aggregates in answer set programming. *Artificial Intelligence*, 2010.

Paolo Ferraris and Vladimir Lifschitz. The stable model semantics for first-order formulas with aggregates[2]. In *Proceedings of International Workshop on Nonmonotonic Reasoning (NMR)*, 2010.

Paolo Ferraris, Joohyung Lee, Vladimir Lifschitz, and Ravi Palla. Symmetric splitting in the general theory of stable models. In *Proceedings of International Joint Conference on Artificial Intelligence (IJCAI)*, pages 797–803, 2009.

Paolo Ferraris, Joohyung Lee, and Vladimir Lifschitz. Stable models and circumscription. *Artificial Intelligence*, 175:236–263, 2011.

Paolo Ferraris. Answer sets for propositional theories. In *Proceedings of International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR)*, pages 119–131, 2005.

Paolo Ferraris. Logic programs with propositional connectives and aggregates[3] Unpublished draft, 2009.

Martin Gebser, Roland Kaminski, Benjamin Kaufmann, and Torsten Schaub. Challenges in answer set solving. In *Logic programming, knowledge representation, and nonmonotonic reasoning*, pages 74–90. Springer, 2011.

Martin Gebser, Amelia Harrison, Roland Kaminski, Vladimir Lifschitz, and Torsten Schaub. Abstract Gringo. *Theory and Practice of Logic Programming*, 15:449–463, 2015.

Michael Gelfond and Vladimir Lifschitz. The stable model semantics for logic programming. In Robert Kowalski and Kenneth Bowen, editors, *Proceedings of International Logic Programming Conference and Symposium*, pages 1070–1080. MIT Press, 1988.

Michael Gelfond and Vladimir Lifschitz. Classical negation in logic programs and disjunctive databases. *New Generation Computing*, 9:365–385, 1991.

Michael Gelfond and Halina Przymusinska. Towards a theory of elaboration tolerance: Logic programming approach. *International Journal of Software Engineering and Knowledge Engineering*, 6(1):89–112, 1996.

[2]`http://userweb.cs.utexas.edu/users/vl/papers/smaf.pdf`

[3]`http://arxiv.org/abs/0812.1462.`

Amelia Harrison and Vladimir Lifschitz. Stable models for infinitary formulas with extensional atoms. *Theory and Practice of Logic Programming*, 16(5-6):771–786, 2016.

Amelia Harrison, Vladimir Lifschitz, David Pearce, and Agustin Valverde. Infinitary equilibrium logic and strong equivalence. In *Proceedings of International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR)*, pages 398–410, 2015.

Amelia Harrison, Vladimir Lifschitz, and Miroslaw Truszczynski. On equivalence of infinitary formulas under the stable model semantics. *Theory and Practice of Logic Programming*, 15(1):18–34, 2015.

Amelia Harrison, Vladimir Lifschitz, and Julian Michael. Proving infinitary formulas. *Theory and Practice of Logic Programming*, 16(5-6):787–799, 2016.

Amelia Harrison, Vladimir Lifschitz, and Dhananjay Raju. Program completion in the input language of gringo. *Theory and Practice of Logic Programming*, 2017. To Appear.

Arend Heyting. Die formalen Regeln der intuitionistischen Logik. *Sitzungsberichte der Preussischen Akademie von Wissenschaften. Physikalisch-mathematische Klasse*, pages 42–56, 1930.

Tsutomu Hosoi. The axiomatization of the intermediate propositional systems S_n of Gödel. *Journal of the Faculty of Science of the University of Tokyo*, 13:183–187, 1966.

Tomi Janhunen, Ilkka Niemelä, Dietmar Seipel, Patrik Simons, and Jia-Huai You. Unfolding partiality and disjunctions in stable model semantics. *ACM Trans. Comput. Logic*, 7(1):1–37, 2006.

László Kalmár. Zurückführung des Entscheidungsproblems auf den Fall von Formeln mit einer einzigen, bindren, Funktionsvariablen. *Compositio Mathematica*, 4:137–144, 1936.

Joohyung Lee and Yunsong Meng. Stable models of formulas with generalized quantifiers. In *Working Notes of the 14th International Workshop on Non-Monotonic Reasoning (NMR)*, 2012.

Nicola Leone, Gerald Pfeifer, Wolfgang Faber, Thomas Eiter, Georg Gottlob, Simona Perri, and Francesco Scarcello. The DLV system for knowledge representation and reasoning. *ACM Transactions on Computational Logic*, 7(3):499–562, 2006.

Yuliya Lierler and Miroslaw Truszczynski. Transition systems for model generators — a unifying approach. *Theory and Practice of Logic Programming, 27th International Conference on Logic Programming (ICLP'11) Special Issue*, 11, issue 4-5, 2011.

Yuliya Lierler. Cmodels: SAT-based disjunctive answer set solver. In *Proceedings of International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR)*, pages 447–452, 2005.

Vladimir Lifschitz and Fangkai Yang. Lloyd-Topor completion and general stable models. In *Working Notes of the 5th Workshop of Answer Set Programming and Other Computing Paradigms (ASPOCP 2012)*, 2012.

Vladimir Lifschitz, David Pearce, and Agustin Valverde. Strongly equivalent logic programs. *ACM Transactions on Computational Logic*, 2:526–541, 2001.

Vladimir Lifschitz, David Pearce, and Agustin Valverde. A characterization of strong equivalence for logic programs with variables. In *Procedings of International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR)*, pages 188–200, 2007.

Vladimir Lifschitz, Leora Morgenstern, and David Plaisted. Knowledge representation and classical logic. In Frank van Harmelen, Vladimir Lifschitz, and Bruce Porter, editors, *Handbook of Knowledge Representation*, pages 3–88. Elsevier, 2008.

Vladimir Lifschitz. What is answer set programming? In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 1594–1597. MIT Press, 2008.

Jan Łukasiewicz. Die Logik und das Grundlagenproblem. In *Les Entretiens de Zürich sue les Fondements et la méthode des sciences mathématiques 1938*, pages 82–100. Leemann, Zürich, 1941.

Grigori Mints. *A Short Introduction to Intuitionistic Logic*. Kluwer, 2000.

Ilkka Niemelä and Patrik Simons. Smodels—an implementation of the stable model and well-founded semantics for normal logic programs. In *Proceedings 4th International Conference on Logic Programming and Nonmonotonic Reasoning (Lecture Notes in Artificial Intelligence 1265)*, pages 420–429. Springer, 1997.

Ilkka Niemelä, Patrik Simons, and Timo Soininen. Stable model semantics for weight constraint rules. In *Procedings of International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR)*, pages 317–331, 1999.

Ilkka Niemelä. Logic programs with stable model semantics as a constraint programming paradigm. *Annals of Mathematics and Artificial Intelligence*, 25:241–273, 1999.

Emilia Oikarinen and Tomi Janhunen. Achieving compositionality of the stable model semantics for Smodels programs. *Theory and Practice of Logic Programming*, 5–6:717–761, 2008.

David Pearce. A new logical characterization of stable models and answer sets. In Jürgen Dix, Luis Pereira, and Teodor Przymusinski, editors, *Non-Monotonic Extensions of Logic Programming (Lecture Notes in Artificial Intelligence 1216)*, pages 57–70. Springer, 1997.

Raymond Reiter. A logic for default reasoning. *Artificial Intelligence*, 13:81–132, 1980.

Francesco Ricca, Antonella Dimasi, Giovanni Grasso, Salvatore Maria Ielpa, Salvatore Iiritano, Marco Manna, and Nicola Leone. A logic-based system for e-tourism. *Fundam. Inf.*, 105(1-2):35–55, January 2010.

Francesco Ricca, Giovanni Grasso, Mario Alviano, Vincenzino Lio, Salvatore Iiritano, and Nicola Leone. Team-building with asnwer set programming in gioiatauro seaport. *Theory and Practice of Logic Programming*, 12:361–381, 2012.

Dana Scott and Alfred Tarski. The sentential calculus with infinitely long expressions. In *Colloquium Mathematicae*, volume 6, pages 165–170, 1958.

Leon Sterling and Ehus Shapiro. *The Art of Prolog: Advanced Programming Techniques*. MIT Press, 1986.

Ivo Thomas. Finite limitations on Dummett's LC. *Notre Dame Journal of Formal Logic*, III(3):170–174, July 1962.

Juha Tiihonen, Timo Soininen, Ilkka Niemelä, and Reijo Sulonen. A practical tool for mass-customising configurable products. In *Proceedings of the 14th International Conference on Engineering Design*, pages 1290–1299, 2003.

Miroslaw Truszczynski. Connecting first-order ASP and the logic FO(ID) through reducts. In Esra Erdem, Joohyung Lee, Yuliya Lierler, and David Pearce, editors, *Correct Reasoning: Essays on Logic-Based AI in Honor of Vladimir Lifschitz*, pages 543–559. Springer, 2012.

Toshio Umezawa. On intermediate many-valued logics. *Journal of the Mathematical Society of Japan*, 11(2):116–128, 1959.

Vita

Amelia grew up in Wimberley, Texas with her younger brother Will. They were cared for by her parents, Jack and Betsy. She attended Wimberley High School where she participated in theatre and debate. After high school, she took a cicuitious route that eventually saw her graduate from UT with B.S./M.S in computer science in 2010. She then worked at Sandia National Laboratories for one year before enrolling in the Ph.D. program at UT.

Permanent Address: amelia.j.harrison@gmail.com

This dissertation was typeset with LaTeX 2_ε[4] by the author.

[4]LaTeX 2_ε is an extension of LaTeX. LaTeX is a collection of macros for TeX. TeX is a trademark of the American Mathematical Society. The macros used in formatting this dissertation were written by Dinesh Das, Department of Computer Sciences, The University of Texas at Austin, and extended by Bert Kay, James A. Bednar, and Ayman El-Khashab.