
DOCTORAL DISSERTATION

# On formula equations and invariant generation

Completed at the

Institute of Discrete Mathematics and Geometry
of the Technical University of Vienna

under the supervision of

Associate Prof. Dr. techn. Stefan Hetzl

by

Dipl.-Ing. Sebastian Zivota
Schönbrunner Straße 87
1050 Wien

Date

Signature

**Abstract**

We present formula equations—first-order formulas with unknowns standing for predicates—as a general formalism for treating certain questions in logic and computer science, like the Auflösungsproblem and loop invariant generation. We investigate the relationship between problems of loop invariant generation and inductive theorem proving. Furthermore, we obtain decidability and undecidability results for formula equations in certain languages, most notably that of affine terms over $\mathbb{Q}$.

# ACKNOWLEDGEMENTS

First and foremost, I want to thank my advisor Professor Stefan Hetzl for his patience, understanding, and support throughout the entirety of my PhD studies. I could not have wished for a better advisor.

I also want to thank Professor Helmut Seidl for graciously hosting me for three months at the Technical University of Munich.

Furthermore, I want to thank my friends Astrid Berg and Lukas Daniel Klausner for their invaluable assistance with the process of actually writing a thesis.

Finally, I want to thank my girlfriend Elisabeth for brightening my life and giving me the strength to get on with it.

# CONTENTS

# INTRODUCTION

This thesis has its origin in automated inductive theorem proving. Induction is an important proof method that can be applied to many different structures, but the case most relevant to us is that of the natural numbers. A simplified version of the problem looks like this: given a set of axioms $\Gamma$ and a formula $\varphi$, both in the language of the natural numbers, find a proof by induction of $\varphi$ from $\Gamma$. The difficulty lies in the fact that it is sometimes not sufficient to apply induction to the goal formula directly, but to a different formula that we do not know, which we call an induction formula. Finding such induction formulas is the central task of inductive theorem proving.

Our work is specifically based on an approach based on tree grammars that was first presented in [EH15]. The idea is to compute a certain number of instance proofs, i.e., proofs of $\varphi(0), \varphi(1), ..., \varphi(k)$ for some $k$, and analyze the terms and instances of axioms occurring in them. Next, we attempt to generalize these terms by computing a parametric grammar. If we succeed, this grammar contains all information about how to instantiate the induction formula in the hypothetical induction proof, but it tells us nothing about the induction formula itself.

From the grammar, we can extract an induction proof schema

1. $\Gamma_1$ implies $X(0, \bar{t}_1)$,
2. $\Gamma_2$ and $X(i, \bar{t}_2)$ implies $X(i+1, \bar{t}_3)$,

3. $\Gamma_3$ and $\forall n\, X(n, \bar{t}_4)$ implies $\varphi(n)$.

where $X$ stands for the unknown induction formula and $\Gamma_{1,2,3}, \bar{t}_{1,2,3,4}$ stand for the axioms and instance terms computed by the grammar. Thus, the problem has been reduced to that of finding a quantifier-free formula satisfying three constraints.

These three constraints bear a striking similarity to another situation where an unknown formula is sought, namely loop invariant generation in software verification. Consider, for example, the following program $p$:

---

**for** $i := 0, ..., n$ **do**
  $p'$
**end for**

---

Now suppose that we want to show that under some axioms $\Gamma$, a formula $\varphi$ holds after running $p$. This can be accomplished by finding an invariant of the loop, that is, a formula that holds before the first iteration of the loop and is preserved by the loop body $p'$. As in the case of induction, we cannot generally assume that $\varphi$ itself can serve as the invariant. Consequently, we have to find a formula $I(i)$ such that

1. $\Gamma$ implies $I(0)$,
2. $\Gamma$, $i < n$ and $I(i)$ imply that $I(i+1)$ holds after $p'$,
3. $\Gamma$ and $I(n)$ imply $\varphi$.

The similarity between the induction and invariant generation problems suggested two questions:

**Question 1.** Can we make the analogy between the problems of finding induction formulas and finding loop invariants precise by constructing a program corresponding to each induction problem such that the invariants of the program are exactly the induction formulas of the original problem?

**Question 2.** If the answer to Question 1 is "yes", can we use this fact to reduce induction problems to loop invariant problems and tackle them with the wealth of methods that have been developed for loop invariant generation?

The short answer to both questions turns out to be "no, but …". We treat Question 1 in Chapter 3. There we shall see that expressing any non-trivial induction

problem as a program requires us to take nondeterminism into account, and that is where the simple analogy we have presented earlier breaks down. We turn to dynamic logic, a form of modal logic that generalizes the Hoare calculus, to clarify the difference between the treatments of nondeterminism in inductive theorem proving and formal verification.

Question 2 is, strictly speaking, rendered moot by the negative answer to Question 1. This precipitated a change in perspective on our part: instead of trying to reduce one kind of problem to the other, we decided to focus on the fact that both kinds of problems are easily expressible in a common formalism, namely that of formula equations.

What we call a formula equation is nothing other than a second-order formula $\exists X_1 ... X_n \, \varphi(X_1, ..., X_n)$ where the $X_i$ are predicate variables and $\varphi$ contains no other second-order variables. This immediately leads to the definition of a solution problem: the question of whether formula equations of a given form have solutions—that is, instantiations of the second-order variables with formulas such that the resulting formula is valid—in a given class modulo a given theory. Clearly, then, both the problems we discussed previously can be viewed as solution problems.

The question of how to solve formula equations has a history going back to the 19th century (see [Wer17a, Wer17b]). It was considered by Schröder in the context of propositional logic under the name *Auflösungsproblem* [Sch95]. The *Auflösungsproblem* is related to the notion of Boolean unification: given propositional formulas $\varphi, \psi$ with Boolean variables, find a substitution $\sigma$ such that $\varphi\sigma = \psi\sigma$ modulo the theory of Boolean algebra. We can view this as an instance of the *Auflösungsproblem* because every substitution $\sigma$ that makes $(\varphi \leftrightarrow \psi)\sigma$ true modulo Boolean algebra serves as a unifier for $\varphi$ and $\psi$. On the other hand, solving a formula $\varphi$ is tantamount to unifying $\varphi$ with $\top$. Boolean unification has been studied extensively, see [MN89] and [MN88, Baa97]. It plays a role in several application areas such as database systems [KKR90, KKR95] and logic programming [BS87].

There is a parallel between Boolean unification problems and equations over e.g. the rational numbers: solving the equation $t(x_1, ..., x_n) = 0$ amounts to finding terms $g_1, ..., g_n$ such that $t(g_1, ..., g_n)$ evaluates to 0 modulo the theory of $\mathbb{Q}$; sim-

ilarly, solving the Boolean unification problem $\varphi(X_1, ..., X_n)$ amounts to finding formulas $G_1, ..., G_n$ such that $\varphi(G_1, ..., G_n)$ is equivalent to $\top$ modulo equality. In other words, the problem is to solve the "equation" $\varphi(X_1, ..., X_n) \leftrightarrow \top$. This parallel is the origin of the term "formula equation".

Boolean unification has been extended to first-order logic under the name "Boolean unification with predicates" (BUP) in [EHW17]. There, the authors consider what we would now call the solution problem of first-order logic with equality with quantifier-free solutions. They show that the problem is undecidable if $\varphi$ is of the form $\forall \bar{x}\, \varphi'$ or $\exists \bar{x}\, \varphi'$ with $\varphi'$ quantifier-free by reduction from the Post correspondence problem and the validity problem of first-order logic, respectively. On the other hand, they prove that for quantifier-free $\varphi$, the problem is $\Pi^p_2$-complete. We present an updated version of the PCP reduction in Chapter 2.

The problem of solving formula equations is related to that of second-order quantifier elimination: we say that a theory $\mathcal{T}$ has quantifier elimination if every formula is equivalent to a quantifier-free formula modulo $\mathcal{T}$. Consider the second-order formula $\psi \equiv \exists X\, \varphi(X)$. If we can find a solution $G$ to the formula equation $\varphi(X)$ modulo $\mathcal{T}$, then $\psi$ is equivalent to the quantifier-free formula $\varphi(G)$. On the other hand, a theory having second-order quantifier elimination does not necessarily allow us to solve formula equations, as it does not guarantee the existence of a witness. Ackermann investigated second-order quantifier elimination in [Ack35].

The authors of [BGMR15] advocate the use of sets of constrained Horn clauses as a target language for problems of program verification. They give a procedure for extracting verification conditions from a program formalism with assertions and subroutine calls that results in nonlinear constrained Horn clauses, i.e., those with more than one formula variable in the antecedent. Formula equations are more general in not restricting the number of positive formula variables that can occur in a clause, resulting in a structure that has no obvious correlate in programs. We follow the line of reasoning of [BGMR15] in considering a logical formalism useful for the representation and solution of problems of program verification and we extend it from solving sets of constrained Horn clauses to solving formula equations. Thus, we arrive at

**Question 3.** Can methods for invariant generation, viewed as procedures for

solving classes of Horn formula equations, be generalized to solving general formula equations?

In Chapter 4, we consider this question specifically in the context of affine spaces over $\mathbb{Q}$. The corresponding invariant generation problem is that of finding affine invariants—i.e., those that can be written as systems of linear equations—for programs that only contain affine operations in their assignments. We show that the formula equations describing invariants form a nontrivial subclass of all affine formula equations and extend a particular invariant generation procedure to cover all affine formula equations. As a result, we obtain a decision procedure for what we call affine solution problems.

Chapter 5 builds on the work on Chapter 4 and widens the scope to take *inequalities* into account. We consider solution problems both over the integers and the rational numbers. Crucially, by adapting a proof from [Mon19], we obtain an undecidability result for solution problems in the presence of inequalities and multiplication, thereby establishing an upper bound on decidable solution problems.

# CHAPTER 1

# LOGICAL PRELIMINARIES

## 1.1 Many-sorted first-order logic

While the basic constructs of first-order logic are well-known, we will present a *many-sorted* version of first-order logic that may be less familiar. The treatment is very similar to that found in [End02]. Note that many-sorted first-order logic possesses no greater expressive power than ordinary (single-sorted) first-order logic.

Let $\mathcal{S}$ be a finite nonempty set whose elements we will call *sorts*. Then a *first-order language* $\mathcal{L}$ over $\mathcal{S}$ is a collection of

- *constant symbols*, each belonging to a sort. We write $c\colon \sigma$ for a symbol $c$ belonging to sort $\sigma$.
- *function symbols*. Associated with each function symbol $f$ is a finite nonempty list of sorts $\sigma_1, ..., \sigma_n \in \mathcal{S}$, called the *domain* of $f$, and a sort $\tau \in \mathcal{S}$, called the *codomain* of $f$. As a slight abuse of notation, we say that $f$ is of the type $\sigma_1 \times ... \times \sigma_n \to \tau$ (written as $f\colon \sigma_1 \times ... \times \sigma_n \to \tau$).
- *predicate symbols*. Associated with each predicate symbol $P$ is a finite (possibly empty) list $\sigma_1, ..., \sigma_n$ of sorts in $\mathcal{S}$. Accordingly we call $P$ a predicate symbol of type $\sigma_1 \times ... \times \sigma_n$ (written as $P\colon \sigma_1 \times ... \times \sigma_n$). We assume

that $\mathcal{L}$ always includes, for each sort $\sigma$, a predicate symbol $=_\sigma : \sigma \times \sigma$ and subsequently omit the subscript.

In addition to $\mathcal{L}$, we also assume that for each $\sigma \in \mathcal{S}$, there is an infinite set $\mathrm{Var}_\sigma$ containing *variables* of sort $\sigma$. We write Var for the set $\bigcup_{\sigma\in\mathcal{S}} \mathrm{Var}_\sigma$.

Now *terms over* $\mathcal{L}$ (or $\mathcal{L}$*-terms*) are constructed inductively. Each term unambiguously belongs to a sort. We write $\mathrm{Terms}_\sigma(\mathcal{L})$ for the set of $\mathcal{L}$-terms of sort $\sigma$ and $\mathrm{Terms}(\mathcal{L})$ for $\bigcup_{\sigma\in\mathcal{S}} \mathrm{Terms}_\sigma(\mathcal{L})$.

- If $x \in \mathrm{Var}_\sigma$, then $x \in \mathrm{Terms}_\sigma(\mathcal{L})$.
- If $c : \sigma$ is a constant symbol of $\mathcal{L}$, then $c \in Terms_\sigma(\mathcal{L})$.
- If $f : \sigma_1 \times ... \times \sigma_n \to \tau$ is a function symbol of $\mathcal{L}$ and $t_1 \in \mathrm{Terms}_{\sigma_1}(\mathcal{L}), ..., t_n \in \mathrm{Terms}_{\sigma_n}(\mathcal{L})$, then $f(t_1, ..., t_n) \in \mathrm{Terms}_\tau(\mathcal{L})$.

Similarly, *formulas over* $\mathcal{L}$ (or $\mathcal{L}$*-formulas*) are constructed inductively. We write $\mathrm{Forms}(\mathcal{L})$ for the set of $\mathcal{L}$-formulas.

- If $P$ is a predicate symbol of $\mathcal{L}$ of sort $\sigma_1 \times ... \times \sigma_n$ and $t_1 : \sigma_1, ..., t_n : \sigma_n$ are $\mathcal{L}$-terms, then $P(t_1, ..., t_n) \in \mathrm{Forms}(\mathcal{L})$. Formulas of this kind are called *atomic formulas*.
- If $\varphi, \psi \in \mathrm{Forms}(\mathcal{L})$, then $\neg\varphi$, $\varphi \wedge \psi$, $\varphi \vee \psi$, and $\varphi \to \psi$ are $\mathcal{L}$-formulas. The connectives $\neg, \wedge, \vee, \to$ are called *propositional connectives*.
- If $\varphi$ is an $\mathcal{L}$-formula and $x \in \mathrm{Var}$, then $\forall x\, \varphi$ and $\exists x\, \varphi$ are $\mathcal{L}$-formulas. The connectives $\forall x$ and $\exists x$ are called *quantifiers*.

We use the symbol $\equiv$ to denote syntactic equality of formulas and the symbols $\Rightarrow, \Leftrightarrow$ as abbreviations of the meta-expressions "if … then …" and "… if and only if …".

The symbols $\wedge$ and $\vee$ bind more tightly than $\to$, so $\varphi \wedge \psi \to \varphi \vee \psi$ is read as $(\varphi \wedge \psi) \to (\varphi \vee \psi)$. The quantifiers $\forall x$ and $\exists x$ bind more tightly than the binary propositional connectives, so $\forall x\, \varphi \wedge \psi$ is read as $(\forall x\, \varphi) \wedge \psi$. If a quantifier is intended to apply as far to the right as syntactically possible, we denote this by placing a period after the quantifier. Thus, $\forall x.\ \varphi \wedge \psi$ stands for $\forall x\, (\varphi \wedge \psi)$.

We can define what it means for variables to occur *freely* in formulas. We write $FV(\varphi)$ for the free variables of $\varphi$.

- If $P(t_1, ..., t_n)$ is an atomic formula, then

$$FV(P(t_1, ..., t_n)) = \{\, v \in \mathrm{Var} \mid v \text{ occurs in some } t_i \,\}.$$

- If $\varphi, \psi$ are formulas, then

$$\begin{aligned} FV(\neg\varphi) &= FV(\varphi) \\ FV(\varphi \wedge \psi) = FV(\varphi \vee \psi) = FV(\varphi \to \psi) &= FV(\varphi) \cup FV(\psi), \\ FV(\forall x\, \varphi) = FV(\exists x\, \varphi) &= FV(\varphi) \setminus \{x\}. \end{aligned}$$

If $FV(\varphi) = \emptyset$, then $\varphi$ is called *closed*.

Let $x_1, ..., x_n$ and $u_1, ..., u_n$ be lists of variables and terms, respectively, such that the $x_i$ are distinct and $x_i$ and $u_i$ are of the same sort for all $i \in \{1, ..., n\}$. The *substitution* $[x_1 \backslash u_1, ..., x_n \backslash u_n]$ (abbreviated as $[\bar{x} \backslash \bar{u}]$) is a function both from terms to terms and from formulas to formulas. On terms, it is defined inductively by

$$\begin{aligned} x_i[\bar{x} \backslash \bar{u}] &= u_i \text{ for } i \in \{1, ..., n\}, \\ y[\bar{x} \backslash \bar{u}] &= y \text{ for all variables } y \notin \{x_1, ..., x_n\}, \\ f(t_1, ..., t_m)[\bar{x} \backslash \bar{u}] &= f(t_1[\bar{x} \backslash \bar{u}], ..., t_n[\bar{x} \backslash \bar{u}]). \end{aligned}$$

Thus, $[x_1 \backslash u_1, ..., x_n \backslash u_n]$ replaces every occurrence of $x_i$ by the corresponding $u_i$. This is also the behavior we expect on formulas, but we have to define substitution a little more carefully there. On the one hand, we do not want to substitute bound occurrences of variables. On the other hand, we need to take care not to "capture" variables under a quantifier by substitution. As an example, consider $\varphi \equiv \forall x\, x = y$ and the substitution $[y \backslash x]$. Applied naively, $\varphi[y \backslash x]$ would result in $\forall x\, x = x$, so the term on the right-hand side is now bound by the quantifier when it was not before. The solution is to rename bound variables such that no capturing occurs. In the example, we can for instance replace the $x$ in $\varphi$ by $z$, whence $\varphi[y \backslash x]$ results in $\forall z\, z = x$.

We will often write terms and formulas as $t(x_1, ..., x_n)$ or $\varphi(x_1, ..., x_n)$, respectively. This does not indicate that all the variables $x_1, ..., x_n$ occur in $t$ (resp. $\varphi$), nor does it mean that no other variables occur in $t$ (resp. $\varphi$). Rather, we use this notation mostly to allow for easier substitutions: if $u_1, ..., u_n$ are terms of appropriate sorts, then $t(u_1, ..., u_n)$ abbreviates $t[x_1 \backslash u_1, ..., x_n \backslash u_n]$, and analogously for formulas. If

we intend to say that there are no free variables in a term or formula other than those indicated, we say that the formula is *fully indicated*.

We can now define what a structure for an $\mathcal{S}$-sorted language $\mathcal{L}$ is. The definition is largely analogous to the single-sorted case. Let $(M_\sigma)_{\sigma\in\mathcal{S}}$ be a family of nonempty, pairwise disjoint sets indexed by $\mathcal{S}$. Furthermore, let $\cdot^{\mathcal{M}}$ be a function that interprets the elements of $\mathcal{L}$ in a "well-sorted" manner, that is:

- If $c\colon \sigma$ is a constant symbol, then $c^{\mathcal{M}} \in M_\sigma$.
- If $f\colon \sigma_1 \times ... \times \sigma_n \to \tau$ is a function symbol, then $f^{\mathcal{M}}$ is a function from $M_{\sigma_1} \times ... \times M_{\sigma_n}$ to $M_\tau$.
- If $P\colon \sigma_1 \times ... \times \sigma_n$ is a predicate symbol, then $P^{\mathcal{M}} \subseteq M_{\sigma_1} \times ... \times M_{\sigma_n}$. Recall our stipulation that every language $\mathcal{L}$ contain a predicate symbol $=_\sigma$ for each sort $\sigma$. We further stipulate that $=_\sigma^{\mathcal{M}}$ always be the identity relation on $M_\sigma$.

Then the pair $\mathcal{M} = ((M_\sigma)_{\sigma\in\mathcal{S}}, \cdot^{\mathcal{M}})$ is called an $\mathcal{L}$-structure.

The function $\cdot^{\mathcal{M}}$ on its own is not enough to assign a value in $\mathcal{M}$ to every term. In order to do so, we also need to assign values to the variables. If $\mathcal{M} = ((M_\sigma)_{\sigma\in\mathcal{S}}, \cdot^{\mathcal{M}})$ is an $\mathcal{L}$-structure, then a *valuation* on $\mathcal{M}$ is a function $v\colon \mathrm{Var} \to \bigcup_{\sigma\in\mathcal{S}} M_\sigma$ such that for each $\sigma$, $v|_{\mathrm{Var}_\sigma}\colon \mathrm{Var}_\sigma \to M_\sigma$. In other words, a valuation on $\mathcal{M}$ associates an element of $\mathcal{M}$ with each variable while preserving sorts. We write $\mathrm{Val}(\mathcal{M})$ for the set of valuations on $\mathcal{M}$.

Valuations can easily be extended from functions on variables to functions on terms over $\mathcal{L}$. We do not distinguish syntactically between these different uses of valuations.

- If $c$ is a constant symbol, then $v(c) = c^{\mathcal{M}}$.
- If $f(t_1, ..., t_n)$ is an $\mathcal{L}$-term, then $v(f(t_1, ..., t_n)) = f^{\mathcal{M}}(v(t_1), ..., v(t_n))$.

It is easy to see that $v(t)$ only depends on the values of $v$ on variables that actually occur in $t$.

This leads us to the definition of the relation "$\varphi$ is true for valuation $v$ in the structure $\mathcal{M}$", written as $(\mathcal{M}, v) \vDash \varphi$:

- Let $\varphi = P(t_1, ..., t_n)$ be an atomic formula. Then

$$(\mathcal{M}, v) \vDash \varphi \text{ iff } (v(t_1), ..., v(t_n)) \in P^{\mathcal{M}}.$$

- Let $\varphi, \psi$ be formulas. Then

$$\begin{aligned}(\mathcal{M}, v) \vDash \neg\varphi &\text{ iff } (\mathcal{M}, v) \nvDash \varphi,\\ (\mathcal{M}, v) \vDash \varphi \wedge \psi &\text{ iff } (\mathcal{M}, v) \vDash \varphi \text{ and } (\mathcal{M}, v) \vDash \psi,\\ (\mathcal{M}, v) \vDash \varphi \vee \psi &\text{ iff } (\mathcal{M}, v) \vDash \varphi \text{ or } (\mathcal{M}, v) \vDash \psi,\\ (\mathcal{M}, v) \vDash \varphi \to \psi &\text{ iff } (\mathcal{M}, v) \nvDash \varphi \text{ or } (\mathcal{M}, v) \vDash \psi.\end{aligned}$$

- Let $\varphi$ be a formula. For two valuations $v, w$ and a variable $x$, we write $v =_x w$ to mean that $v$ and $w$ agree on all variables except possibly $x$. Then

$$\begin{aligned}(\mathcal{M}, v) \vDash \forall x\, \varphi &\text{ iff } (\mathcal{M}, w) \vDash \varphi \text{ for all } w \in \mathrm{Val}(\mathcal{M}) \text{ with } v =_x w,\\ (\mathcal{M}, v) \vDash \exists x\, \varphi &\text{ iff } (\mathcal{M}, w) \vDash \varphi \text{ for some } w \in \mathrm{Val}(\mathcal{M}) \text{ with } v =_x w.\end{aligned}$$

As for terms, the relation $(\mathcal{M}, v) \vDash \varphi$ only depends on the values of $v$ on the free variables of $\varphi$. As a consequence, the truth of a closed formula in a structure does not depend on a valuation at all. Thus, for a closed formula $\varphi$, we write $\mathcal{M} \vDash \varphi$ iff $(\mathcal{M}, v) \vDash \varphi$ for any $v \in \mathrm{Val}(\mathcal{M})$.

It is typical, given an $\mathcal{L}$-structure $\mathcal{M}$, to use elements of $\mathcal{M}$ in terms and formulas as if they were constant symbols of $\mathcal{L}$.

We can generalize the interpretation function $\cdot^{\mathcal{M}}$ to all terms and formulas.

- If $x_1 : \sigma_1, ..., x_n : \sigma_n \in \mathrm{Var}$ and $t(x_1, ..., x_n) \in \mathrm{Terms}_\tau(\mathcal{L})$ is a fully indicated term, then

$$t^{\mathcal{M}} : \prod_{i=1}^{n} M_{\sigma_i} \to \tau,\ (x_1, ..., x_n) \mapsto t(x_1, ..., x_n)^{\mathcal{M}}.$$

- If $x_1 : \sigma_1, ..., x_n : \sigma_n \in \mathrm{Var}$ and $\varphi(x_1, ..., x_n) \in \mathrm{Forms}(\mathcal{L})$ is a fully indicated formula, then

$$\varphi^{\mathcal{M}} = \left\{ (a_1, ..., a_n) \in \prod_{i=1}^{n} M_{\sigma_i} \;\middle|\; \mathcal{M} \vDash \varphi(a_1, ..., a_n) \right\}.$$

  Note that $\varphi(a_1, ..., a_n)$ is closed because $\varphi$ has no free variables other than $x_1, ..., x_n$, so $\mathcal{M} \vDash \varphi(a_1, ..., a_n)$ is meaningful.

The following proposition about the interpretations of terms and formulas is easily proved.

**Proposition 1.1.** *Let $\mathcal{S}$ be a set of sorts, $\mathcal{L}$ an $\mathcal{S}$-sorted language, and $\mathcal{M} = ((M_\sigma)_{\sigma\in\mathcal{S}}, \cdot^{\mathcal{M}})$ an $\mathcal{L}$-structure. Let $M = \bigcup_{\sigma\in S} M_\sigma$. Furthermore, let*

- $t(\bar{x}), t_1(\bar{x}), ..., t_n(\bar{x}) \in \mathrm{Terms}(\mathcal{L})$,
- $\varphi(\bar{x}), \psi(\bar{x}), \varphi_1(\bar{x}), ..., \varphi_k(\bar{x}) \in \mathrm{Forms}(\mathcal{L})$.

*Then:*

1. $(\bigvee_{i=1}^m \varphi_i)^{\mathcal{M}} = \bigcup_{i=1}^m \varphi_i^{\mathcal{M}}$
2. $(\bigwedge_{i=1}^m \varphi_i)^{\mathcal{M}} = \bigcap_{i=1}^m \varphi_i^{\mathcal{M}}$
3. $\varphi[\bar{x}\backslash\bar{t}]^{\mathcal{M}} = (\bar{t}^{\mathcal{M}})^{-1}(\varphi^{\mathcal{M}})$
4. $\mathcal{M} \vDash \forall\bar{x}.\ \varphi(\bar{x}) \to \psi(\bar{x})$ *iff* $\varphi^{\mathcal{M}} \subseteq \psi^{\mathcal{M}}$

A set $\mathcal{T}$ of closed formulas is true in a structure $\mathcal{M}$, written as $\mathcal{M} \vDash \mathcal{T}$, if all formulas in $\mathcal{T}$ are true in $\mathcal{M}$. In this case, $\mathcal{M}$ is called a *model* of $\mathcal{T}$. If $\mathcal{M}$ is an $\mathcal{L}$-structure, we call the set $\mathrm{Th}(\mathcal{M})$ of all closed $\mathcal{L}$-formulas that are true in $\mathcal{M}$ the theory of $\mathcal{M}$.

Let $\mathcal{L}$ be a first-order language, $\mathcal{T}$ a set of closed formulas, and $\varphi$ an $\mathcal{L}$-formula. We say that $\mathcal{T}$ *entails* $\varphi$, written as $\mathcal{T} \vDash \varphi$, if $\mathcal{M} \vDash \varphi$ for every model $\mathcal{M}$ of $\mathcal{T}$.

A set of closed formulas $\mathcal{T}$ is called a *theory* if it is *closed under entailment*, i.e., for all formulas $\varphi$, $\mathcal{T} \vDash \varphi$ implies $\varphi \in \mathcal{T}$.

Finally, let us mention *sequent notation*, which we will use at several points in this thesis. Fundamentally, a sequent is a pair $(\Gamma, \Delta)$ of sets of formulas, typically written as $\Gamma \vdash \Delta$. $\Gamma$ is called the *antecedent* of the sequent and $\Delta$ the *succedent*. If $\Gamma = \{A_1, ..., A_m\}$ and $\Delta = \{B_1, ..., B_n\}$, then $\Gamma \vdash \Delta$ can also be written as $A_1, ..., A_m \vdash B_1, ..., B_n$. We may even mix sets and individual formulas when writing sequents, as in $\Gamma, \{\varphi(u_i)\}_{i=1}^m, x = y \vdash \Delta$.

The sequent $\Gamma \vdash \Delta$ abbreviates the formula $\bigwedge\Gamma \to \bigvee\Delta$. In other words, the commas in the antecedent of $A_1, ..., A_m \vdash B_1, ..., B_n$ are interpreted conjunctively and those in the succedent are interpreted disjunctively, with the turnstile symbol $\vdash$ denoting an implication. For this reason, sequents typically do not contain top-level conjunctions in the antecedent or top-level disjunctions in the succedent. The empty sequent $\vdash$ is equivalent to $\bot$.

## 1.2 Rewriting and normal forms

We will briefly discuss the concept of *rewriting*. We follow the terminology of [BN98]. An *abstract reduction system* is just a binary relation $\rightsquigarrow$ on some set $M$. We write the reflexive, transitive closure of such a relation $\rightsquigarrow$ as $\rightsquigarrow^*$. If $x \in M$ and there is a $y \in M$ such that $x \rightsquigarrow y$, we call $x$ *reducible*, otherwise we call it *irreducible* or a *normal form*. If $y$ is a normal form and $x \rightsquigarrow^* y$, we say that $y$ is a normal form of $x$. We say that $\rightsquigarrow$ is *normalizing* if every $x \in M$ has a normal form and *terminating* if there are no infinite sequences $x_0 \rightsquigarrow x_1 \rightsquigarrow x_2 \rightsquigarrow \ldots$ (i.e., every sequence of reductions eventually leads to a normal form). In particular, termination implies that $\rightsquigarrow$ is irreflexive. We say that $\rightsquigarrow$ is *confluent* if, whenever $x \rightsquigarrow^* y_1$ and $x \rightsquigarrow^* y_2$, there is a $z \in M$ such that $y_1 \rightsquigarrow^* z$ and $y_2 \rightsquigarrow^* z$. It is easy to see that if $\rightsquigarrow$ is confluent then every element of $M$ has at most one normal form. This immediately implies that if $\rightsquigarrow$ is both normalizing and confluent, every element of $M$ has a unique normal form.

Now let $\mathcal{L}$ be a logical language. A *rewrite rule* is an equation $s = t$ where $s, t$ are $\mathcal{L}$-terms such that $s$ is not a variable and $t$ contains no variables other than those in $s$. Rewrite rules are intended to be read left to right, that is, the left-hand term can be replaced by the right-hand term, but not the other way around. A *term rewriting system* is a set of rewrite rules. A term rewriting system $R$ induces a *rewriting relation* as follows: let $\rightsquigarrow_R$ be the least binary relation on Terms($\mathcal{L}$) that subsumes all the rewrite rules of $R$ (i.e., if $s = t \in R$, then $s \rightsquigarrow_R t$) and is compatible with term construction. By identifying $R$ with $\rightsquigarrow_R$, we can regard $R$ as an abstract reduction system and consequently apply concepts like normalization, confluence, etc., to it.

The case that is particularly relevant for us is that of rewriting formulas. There are several important syntactical classes of formulas that can be described as normal forms with respect to certain sets of rewriting rules. Strictly speaking, this does not fit the definition of a term rewriting system, since we are not dealing with terms over a logical language. However, the same principle applies: we define sets of rewriting rules and obtain rewriting relations by extending them to be compatible with formula construction.

We list the rules in question in the following definition.

**Definition 1.2** (Logical rewrite rules)**.** Let $\varphi, \psi, \vartheta$ be formulas.

$$\varphi \to \psi \rightsquigarrow \neg\varphi \lor \psi \qquad (\to\text{-el})$$
$$\neg\neg\varphi \rightsquigarrow \varphi \qquad (\neg\neg)$$
$$\neg(\varphi \land \psi) \rightsquigarrow \neg\varphi \lor \neg\psi \qquad (\neg\land)$$
$$\neg(\varphi \lor \psi) \rightsquigarrow \neg\varphi \land \neg\psi \qquad (\neg\lor)$$
$$\neg\forall x\, \varphi \rightsquigarrow \exists x\, \neg\varphi \qquad (\neg\forall)$$
$$\neg\exists x\, \varphi \rightsquigarrow \forall x\, \neg\varphi \qquad (\neg\exists)$$

$$\varphi \lor (\psi \land \vartheta) \rightsquigarrow (\varphi \lor \psi) \land (\varphi \lor \vartheta), \quad (\varphi \land \psi) \lor \vartheta \rightsquigarrow (\varphi \lor \vartheta) \land (\psi \lor \vartheta) \qquad (\lor\land)$$
$$\varphi \land (\psi \lor \vartheta) \rightsquigarrow (\varphi \land \psi) \lor (\varphi \land \vartheta), \quad (\varphi \lor \psi) \land \vartheta \rightsquigarrow (\varphi \land \vartheta) \lor (\psi \land \vartheta) \qquad (\land\lor)$$
$$\forall x\, \varphi \land \psi \rightsquigarrow \forall x\, (\varphi \land \psi), \quad \varphi \land \forall x\, \psi \rightsquigarrow \forall x\, (\varphi \land \psi) \qquad (\forall\land)$$
$$\forall x\, \varphi \lor \psi \rightsquigarrow \forall x\, (\varphi \lor \psi), \quad \varphi \lor \forall x\, \psi \rightsquigarrow \forall x\, (\varphi \lor \psi) \qquad (\forall\lor)$$
$$\forall x\, \varphi \to \psi \rightsquigarrow \exists x\, (\varphi \to \psi) \qquad (\forall \to\colon \ell)$$
$$\varphi \to \forall x\, \psi \rightsquigarrow \forall x\, (\varphi \to \psi) \qquad (\forall \to\colon r)$$
$$\exists x\, \varphi \land \psi \rightsquigarrow \exists x\, (\varphi \land \psi), \quad \varphi \land \exists x\, \psi \rightsquigarrow \exists x\, (\varphi \land \psi) \qquad (\exists\land)$$
$$\exists x\, \varphi \lor \psi \rightsquigarrow \exists x\, (\varphi \lor \psi), \quad \varphi \lor \exists x\, \psi \rightsquigarrow \exists x\, (\varphi \lor \psi) \qquad (\exists\lor)$$
$$\exists x\, \varphi \to \psi \rightsquigarrow \forall x\, (\varphi \to \psi) \qquad (\exists \to\colon \ell)$$
$$\varphi \to \exists x\, \psi \rightsquigarrow \exists x\, (\varphi \to \psi) \qquad (\exists \to\colon r)$$

The rules $\to$-el, $\neg\neg$, $\neg\land$, $\neg\lor$, $\neg\exists$, $\neg\forall$, $\lor\land$, $\land\lor$, $\forall\land$, $\exists\lor$, $\forall \to\colon \ell$, and $\exists \to\colon r$ preserve equivalence. The rules $\forall\lor$, $\exists\land$, $\forall \to\colon r$, and $\exists \to\colon \ell$ only preserve equivalence if the quantified variable $x$ does not occur freely in the unquantified subformula. We can always ensure that these conditions hold for a formula if we stipulate that no two quantifiers quantify over the same variable and that no variable occurs both freely and bound in the formula.

Before we proceed to the various normal forms on formulas, we need to define one more concept.

**Definition 1.3** (Literal)**.** A *literal* is an atomic formula or negated atomic formula.

**Definition 1.4** (Negation normal form)**.** *Negation normal form* (NNF) is the normal form with respect to the set $\{\to\text{-el}, \neg\neg, \neg\land, \neg\lor, \neg\forall, \neg\exists\}$; that is to say,

a formula is in NNF if none of the aforementioned rewrite rules can be applied to it.

It is easy to see that a formula is in negation normal form if it is composed only of literals, $\wedge$, $\vee$, $\forall$, and $\exists$. Moreover, if $\varphi$ is a formula, then any NNF of $\varphi$ is equivalent to $\varphi$ because all relevant rewrite rules preserve equivalence.

**Definition 1.5** (Prenex normal form)**.** *Prenex normal form* is the normal form with respect to the set $\{\neg\forall,\ \neg\exists,\ \forall\wedge,\ \forall\vee,\ \forall\ \to\colon \ell,\ \forall\ \to\colon r,\ \forall\wedge,\ \forall\vee,\ \forall\ \to\colon \ell,\ \forall\ \to\colon r\}$. Formulas in prenex normal form are also said to be *prenex*.

We have remarked before that not all of these rules preserve equivalence. This means that naively applying them to a formula will not necessarily result in an equivalent prenex formula. However, we can always obtain an equivalent prenex formula by renaming bound quantifiers before applying the rewriting rules.

Intuitively, a formula is in prenex normal form if it starts with a (possibly empty) block of quantifiers followed by a quantifier-free formula, which is called its matrix.

**Definition 1.6** (Quantifier complexity)**.** We inductively define the classes $\Pi_n$ and $\Sigma_n$ of prenex formulas for $n \in \mathbb{N}$:

1. If $\varphi$ is quantifier-free, then $\varphi$ is both $\Pi_0$ and $\Sigma_0$.
2. If $\varphi$ is of the form $\forall x_1 ... \forall x_n\, \psi$ where $\psi$ is $\Sigma_n$, then $\varphi$ is $\Pi_{n+1}$.
3. If $\varphi$ is of the form $\exists x_1 ... \exists x_n\, \psi$ where $\psi$ is $\Pi_n$, then $\varphi$ is $\Sigma_{n+1}$.

We have already established that every first-order formula is equivalent to a formula in prenex normal form. Therefore, every formula is equivalent to a $\Pi_n$ or $\Sigma_n$ formula for some $n$. Moreover, since adding vacuous quantifiers to a formula preserves equivalence, every $\Pi_n$ formula is also equivalent to a $\Pi_{n+1}$ and a $\Sigma_{n+1}$ formula.

**Definition 1.7** (Polarity)**.** Let $\varphi$ be a first-order formula. We assign to each subformula $\psi$ of $\varphi$ a value $\mathrm{pol}(\psi) \in \{-1, +1\}$, called its *polarity*: let $n$ be the number of negations and antecedents of implications above $\psi$. Then $\mathrm{pol}(\psi) = (-1)^n$. In particular, $\varphi$ itself has polarity $+1$.

**Definition 1.8** (Strong and weak quantifiers)**.** Let $\varphi$ be a first-order formula and $\psi \equiv Qx\,\psi'$ with $Q \in \{\forall, \exists\}$ a subformula of $\varphi$. Then $Qx$ is called a *strong quantifier* if $Q = \forall$ and $\mathrm{pol}(\psi) = +1$ or $Q = \exists$ and $\mathrm{pol}(\psi) = -1$. Otherwise, $Qx$ is called a *weak quantifier*.

Simply put, strong quantifiers are negated existential and non-negated universal ones. Conversely, weak quantifiers are negated universal and non-negated existential ones.

Since prenex formulas never contain negated quantifiers, the characterization of strong and weak quantifiers is very simple in that case: "strong" and "weak" just coincide with "universal" and "existential".

**Definition 1.9** (Clause, dual clause)**.**

1. A *clause* is a disjunction of literals.
2. A *dual clause* is the dual form of a clause, that is, a conjunction of literals.

**Definition 1.10** (Conjunctive normal form)**.** *Conjunctive normal form* (CNF) is the normal form on quantifier-free formulas with respect to the rules {→-el,¬¬, ¬∧, ¬∨, ∨∧ }.

**Remark.** In other words, a formula $\varphi$ is in CNF if it is a conjunction of clauses, i.e.,

$$\varphi \equiv \bigwedge_{i=1}^{m} \bigvee_{j=1}^{n_i} L_i,$$

where the $L_i$ are literals.

Clearly, if a formula is in CNF, it is also in NNF.

The clause $\neg A_1 \vee ... \vee \neg A_m \vee B_1 \vee ... \vee B_n$ is logically equivalent to the formula $A_1 \wedge ... \wedge A_m \to B_1 \vee ... \vee B_n$. For this reason, we typically write it in sequent form as $A_1, ..., A_m \vdash B_1, ..., B_n$.

We can slightly generalize conjunctive normal form to $\Pi_1$ formulas. Moreover, it is common to regard the CNF of a formula as a set of clauses.

**Definition 1.11** (Cl($\varphi$))**.** Let $\varphi$ be a $\Pi_1$ formula and $\psi \equiv \forall \bar{x}\, \psi'$ a prenex normal form of $\varphi$. Then the CNF of $\psi'$ is unique modulo associativity and idempotence. If the CNF is

$$\bigwedge_{i=1}^{m} \bigvee_{j=1}^{n_i} L_{i,j},$$

where $L_{i,j}$ are literals, then Cl($\varphi$) is the set

$$\left\{ \left\{ L_{i,j} \mid j \in \{1, ..., n_i\} \right\} \mid i \in \{1, ..., m\} \right\}.$$

It follows that

$$\models \varphi \leftrightarrow \forall \bar{x} \bigwedge \left\{ \left\{ \bigvee c \mid c \in \mathrm{Cl}(\varphi) \right\} \right\}.$$

The dual of the conjunctive normal form is the *disjunctive normal form*. Its definition and the procedure for obtaining it are very much analogous.

**Definition 1.12** (Disjunctive normal form)**.** *Disjunctive normal form* (DNF) is the normal form on quantifier-free formulas with respect to {$\to$-el,$\neg\neg$, $\neg\wedge$, $\neg\vee$, $\wedge\vee$ }.

A formula $\varphi$ is in DNF if it is a disjunction of dual clauses.

To close out this section, let us briefly touch on *skolemization*. There are two procedures, duals of one another, referred to by that name. The first is a map $\mathrm{sk}^-$ from first-order formulas over a language $\mathcal{L}$ to formulas over a language $\mathcal{L}' \supseteq \mathcal{L}$ that only contain strong quantifiers. It does in general not preserve equivalence, but it does preserve satisfiability, i.e., $\varphi$ is true in some $\mathcal{L}$-structure iff $\mathrm{sk}^-(\varphi)$ is true in some $\mathcal{L}'$-structure. As a consequence, this form of skolemization is integral for automated theorem proving with refutational systems like resolution: $\varphi$ is valid iff $\neg\varphi$ is unsatisfiable iff $\mathrm{sk}^-(\neg\varphi)$ is unsatisfiable.

Let $\varphi$ be a formula. $\mathrm{sk}^-$ works as follows: for each weak quantifier $Qx$ in $\varphi$, let $y_1, ..., y_n$ be the variables in $\varphi$ that are strongly quantified above $Qx$. We choose a fresh $n$-ary function symbol $f_x$ and replace all occurrences of $x$ that are bound by $Qx$ by $f_x(y_1, ..., y_n)$. Finally we delete the quantifier $Qx$.

**Example 1.13.** Consider the formula $\varphi \equiv \forall x\, P(x) \vee \exists y \forall z \exists w\, Q(y, z, w)$. We have $\mathrm{sk}^-(\varphi) \equiv \forall x\, P(x) \vee \forall z\, Q(f_y, z, f_w(z))$.

The dual operation $\mathrm{sk}^+$ maps formulas over $\mathcal{L}$ to formulas over some $\mathcal{L}' \supseteq \mathcal{L}$ that only contain weak quantifiers. It works in exactly the opposite way as $\mathrm{sk}^-$ and consequently preserves not satisfiability, but validity, so $\varphi$ is true in all $\mathcal{L}$-structures iff $\mathrm{sk}^+(\varphi)$ is true in all $\mathcal{L}'$-structures. This makes $\mathrm{sk}^+$ relevant to theorem proving in direct systems like **LK** since provability in such calculi is equivalent to validity.

## 1.3 Proofs

We will mostly use formal proofs in our investigation of inductive theorem proving in Section 2.3. We use Gentzen's proof calculus **LK**. This calculus operates on sequents instead of arbitrary first-order formulas. For an exhaustive discussion of **LK**, see [Tak87].

**Definition 1.14** (Inference rules of **LK**)**.** An $n$-ary *inference rule* is an expression of the form $\dfrac{\Gamma_1 \vdash \Delta_1 \quad \ldots \quad \Gamma_n \vdash \Delta_n}{\Gamma \vdash \Delta}\,\mathrm{r}$ . We list the inference rules of **LK** below. In all of these rules, the sequents above the line are called *premises* and the sequent below the line is called the *conclusion*. Moreover, in all rules except *cut*, the formulas that are emphasized in the conclusion are called the *main formulas*, while those that are emphasized in the premises are called *auxiliary formulas*.

1. Axioms:

$$A \vdash A \qquad \vdash t = t$$

   Here, $A$ is an atomic formula and $t$ is a term.

2. Contraction:

$$\frac{\varphi, \varphi, \Gamma \vdash \Delta}{\varphi, \Gamma \vdash \Delta}\,\mathrm{c{:}l} \qquad \frac{\Gamma \vdash \Delta, \varphi, \varphi}{\Gamma \vdash \Delta, \varphi}\,\mathrm{c{:}r}$$

3. Weakening:

$$\frac{\Gamma \vdash \Delta}{\varphi, \Gamma \vdash \Delta}\,\mathrm{w{:}l} \qquad \frac{\Gamma \vdash \Delta}{\Gamma \vdash \Delta, \varphi}\,\mathrm{w{:}r}$$

4. Propositional rules:

$$\frac{\Gamma \vdash \Delta, \varphi}{\neg\varphi, \Gamma \vdash \Delta}\,\neg\mathrm{:l} \qquad \frac{\varphi, \Gamma \vdash \Delta}{\Gamma \vdash \Delta, \neg\varphi}\,\neg\mathrm{:r}$$

$$\frac{\varphi, \psi, \Gamma \vdash \Delta}{\varphi \wedge \psi, \Gamma \vdash \Delta} \wedge\text{:l} \qquad \frac{\Gamma \vdash \Delta, \varphi \qquad \Pi \vdash \Lambda, \psi}{\Gamma, \Pi \vdash \Delta, \Lambda, \varphi \wedge \psi} \wedge\text{:r}$$

$$\frac{\varphi, \Gamma \vdash \Delta \qquad \psi, \Pi \vdash \Lambda}{\varphi \vee \psi, \Gamma, \Pi \vdash \Delta, \Lambda} \vee\text{:l} \qquad \frac{\Gamma \vdash \Delta, \varphi, \psi}{\Gamma \vdash \Delta, \varphi \vee \psi} \vee\text{:r}$$

$$\frac{\Gamma \vdash \Delta, \varphi \qquad \psi, \Pi \vdash \Lambda}{\varphi \to \psi, \Gamma, \Pi \vdash \Delta, \Lambda} \to\text{:l} \qquad \frac{\varphi, \Gamma \vdash \Delta, \psi}{\Gamma \vdash \Delta, \varphi \to \psi} \vee\text{:r}$$

5. Quantifier rules:

$$\frac{\varphi[x\backslash t], \Gamma \vdash \Delta}{\forall x\, \varphi, \Gamma \vdash \Delta} \forall\text{:l} \qquad \frac{\Gamma \vdash \Delta, \varphi[x\backslash \alpha]}{\Gamma \vdash \Delta, \forall x\, \varphi} \forall\text{:r}$$

$$\frac{\varphi[x\backslash \alpha], \Gamma \vdash \Delta}{\exists x\, \varphi, \Gamma \vdash \Delta} \exists\text{:l} \qquad \frac{\Gamma \vdash \Delta, \varphi[x\backslash t]}{\Gamma \vdash \Delta, \exists x\, \varphi} \exists\text{:r}$$

   Here, $t$ is any term, while $\alpha$ is a variable that does not occur in $\Gamma$, $\Delta$ or $A$, called an *eigenvariable*. The inference rules that use eigenvariables are called *strong quantifier rules*, the others *weak quantifier rules*.

6. The cut rule:

$$\frac{\Gamma \vdash \Delta, \varphi \qquad \varphi, \Pi \vdash \Lambda}{\Gamma, \Pi, \vdash \Delta, \Lambda} \text{cut}$$

   The formula $\varphi$ is called the *cut formula* of the inference.

7. The equality rule:

$$\frac{s = t, \Gamma \vdash \Delta, \varphi(s)}{s = t, \Gamma \vdash \Delta, \varphi(t)} \text{eq}$$

**LK** proofs are constructed inductively from instances of the inference rules.

## 1.4 Arithmetic

**Definition 1.15** (Arithmetical language)**.** A first-order language $\mathcal{L}$ is called *arithmetical* if it contains the sort **Nat** and

- the constant symbol $0 \colon \mathbf{Nat}$,
- the function symbols

$$s \colon \mathbf{Nat} \to \mathbf{Nat}, \qquad p \colon \mathbf{Nat} \to \mathbf{Nat},$$
$$+ \colon \mathbf{Nat} \times \mathbf{Nat} \to \mathbf{Nat}, \qquad - \colon \mathbf{Nat} \to \mathbf{Nat},$$
$$\cdot \colon \mathbf{Nat} \times \mathbf{Nat} \to \mathbf{Nat},$$

- and the relation symbol $\leq\colon \mathbf{Nat} \times \mathbf{Nat}$.

The *language of arithmetic* $\mathcal{L}_A$ is the smallest arithmetical language, i.e., $\mathcal{L}_A$ contains only the sort **Nat** and the symbols described above.

We will use $x < y$ to abbreviate $x \leq y \wedge x \neq y$ and occasionally write $x \geq y$ and $x > y$ for $y \leq x$ and $y < x$, respectively.

The intended interpretations of $p$ and $-$ are the *truncated predecessor* and *truncated subtraction* functions, respectively:

$$\begin{aligned} p(0) &= 0, & x - 0 &= x, \\ p(sx) &= x, & x - sy &= p(x - y). \end{aligned}$$

If $\mathcal{L}$ is an arithmetical language and $n \in \mathbb{N}$, then we use $\bar{n}$ to denote the $\mathcal{L}$-term $\underbrace{s \ldots s}_{n} 0\colon \mathbf{Nat}$, the *numeral* corresponding to $n$.

We define a collection of arithmetical axioms that we will refer to later.

**Definition 1.16** (Arithmetical axioms)**.**

- Successor:

$$\forall x\, sx \neq 0 \tag{S1}$$
$$\forall x, y.\ sx = sy \to x = y \tag{S2}$$
$$\forall x.\ x = 0 \vee \exists y\, x = sy \tag{S3}$$

- Predecessor:

$$p0 = 0 \tag{P1}$$
$$\forall x\, psx = x, \tag{P2}$$

- Addition:

$$\forall x\, x + 0 = x \tag{A1}$$
$$\forall x\, x + sy = s(x + y) \tag{A2}$$

- Subtraction:

$$\forall x\, x - 0 = 0 \tag{Mi1}$$
$$\forall x, y\, x - sy = p(x - y) \tag{Mi2}$$

- Multiplication:

$$\forall x\, x \cdot 0 = 0 \tag{M1}$$
$$\forall x, y\, x \cdot sy = x \cdot y + x \tag{M2}$$

- Order:

$$\forall x\, 0 \leq x, \tag{O1}$$
$$\forall x\, x \leq x, \tag{O2}$$
$$\forall x, y.\ x \leq y \wedge y \leq x \to x = y \tag{O3}$$
$$\forall x, y, z.\ x \leq y \wedge y \leq z \to x \leq z \tag{O4}$$
$$\forall x, y.\ x \leq y \vee x \geq y \tag{O5}$$
$$\forall x, y.\ x \leq y \leftrightarrow \exists z.\ z + x = y \tag{OA}$$
$$\forall x, y.\ x \leq y \leftrightarrow x < sy \tag{OS1}$$
$$\forall x, y.\ x < y \leftrightarrow sx \leq y \tag{OS2}$$
$$\forall x, y.\ x > 0 \leftrightarrow spx = x \tag{OSP}$$
$$\forall x, y.\ x \leq y \to x - y = 0 \tag{OMi}$$
$$\forall x, y.\ x > y \to x - sy < x - y \tag{OSMi}$$

**Definition 1.17** (Robinson arithmetic)**.** *Robinson arithmetic* (**Q**) is the theory axiomatized by the arithmetical axioms (S1)–(S3), (A1), (A2), (M1), (M2), and (OA).

**Definition 1.18** (Peano arithmetic)**.** Let $\mathcal{L}$ be an arithmetical language and $\varphi(n)$ an $\mathcal{L}$-formula, where $n$ is a variable of sort **Nat**. Then the induction axiom for $\varphi(n)$ is the formula

$$\mathrm{IND}_{\varphi(n)} \equiv \varphi(0) \wedge \forall i\, (\varphi(i) \to \varphi(si)) \to \forall n\, \varphi(n) \tag{IND}$$

*Peano arithmetic* (**PA**) is the $\mathcal{L}_A$-theory that is axiomatized by the axioms of **Q** plus the induction axioms for all $\mathcal{L}_A$-formulas.

Later, we will also use a slightly different induction principle. This is justified because the two principles are equivalent under very weak assumptions.

**Lemma 1.19.** *Let $\mathcal{L}$ be an arithmetical language and for $\varphi(n) \in \mathrm{Forms}(\mathcal{L})$ let $\mathrm{IND}'_{\varphi(n)}$ be the $\mathcal{L}$-formula*

$$\mathrm{IND}'_{\varphi(n)} \equiv \forall n.\ \varphi(0) \wedge \forall i\,(\varphi(i) \wedge i < n \to \varphi(si)) \to \varphi(n).$$

*Furthermore, let*

$$\mathcal{T} = \{(\mathrm{O2}), (\mathrm{OS2})\} \cup \left\{ \mathrm{IND}_{\varphi(n)} \;\middle|\; \varphi(n) \in \mathrm{Forms}(\mathcal{L}) \right\},$$
$$\mathcal{T}' = \{(\mathrm{O2}), (\mathrm{OS2})\} \cup \left\{ \mathrm{IND}'_{\varphi(n)} \;\middle|\; \varphi(n) \in \mathrm{Forms}(\mathcal{L}) \right\}.$$

*Then $\mathcal{T}' = \mathcal{T}$.*

*Proof.* We show $\mathcal{T} \vDash \mathrm{IND}'_{\varphi(n)}$ and $\mathcal{T}' \vDash \mathrm{IND}_{\varphi(n)}$ for all formulas $\varphi(n)$. Note that $\mathrm{IND}_{\varphi(n)}$ is equivalent to $\forall n.\ \varphi(0) \wedge \forall i\,(\varphi(i) \to \varphi(si)) \to \varphi(n)$.

We deal with the easier case first. Let $\mathcal{M}$ be a model of $\mathcal{T}'$, $v \in \mathrm{Val}(\mathcal{M})$, and assume $(\mathcal{M}, v) \vDash \varphi(0) \wedge \forall i\,(\varphi(i) \to \varphi(si))$. Clearly, we also have $(\mathcal{M}, v) \vDash \forall i.\ \varphi(i) \wedge i < n \to \varphi(si)$ and because $\mathcal{M} \vDash \mathrm{IND}'_{\varphi(n)}$ we obtain $(\mathcal{M}, v) \vDash \varphi(n)$. Therefore, $\mathcal{M} \vDash \mathrm{IND}_{\varphi(n)}$.

For the converse case, let $\mathcal{M}$ be a model of $\mathcal{T}$, $v \in \mathrm{Val}(\mathcal{M})$, and assume $(\mathcal{M}, v) \vDash \varphi(0) \wedge \forall i\,(\varphi(i) \wedge i < n \to \varphi(si))$. Now let $\varphi'(j) \equiv j \leq n \to \varphi(j)$ and consider $\mathrm{IND}_{\varphi'(j)}$:

$$\begin{aligned}
\mathrm{IND}_{\varphi'(j)} &\equiv \forall j. \quad \varphi'(0) \wedge \forall i\,(\varphi'(i) \to \varphi'(si)) \to \varphi'(j) \\
&\equiv \forall j. \quad (0 \leq n \to \varphi(0)) \\
&\qquad \wedge \forall i\,((i \leq n \to \varphi(i)) \to (si \leq n \to \varphi(si))) \\
&\qquad \to j \leq n \to \varphi(j)
\end{aligned}$$

Because

$$\begin{aligned}
(i \leq n \to \varphi(i)) \to (si \leq n \to \varphi(si)) &\Leftrightarrow (i \leq n \to \varphi(i)) \wedge si \leq n \to \varphi(si) \\
&\overset{(\mathrm{OS2})}{\Leftrightarrow} (i \leq n \to \varphi(i)) \wedge i < n \to \varphi(si) \\
&\Leftrightarrow \varphi(i) \wedge i < n \to \varphi(si),
\end{aligned}$$

$\mathrm{IND}_{\varphi'(j)}$ is equivalent to $\forall j.\ (0 \leq n \to \varphi(0)) \wedge \forall i\,(\varphi(i) \wedge i < n \to \varphi(si)) \to j \leq n \to \varphi(j)$ in all models of $\mathcal{S}$. Since $\mathcal{M} \vDash 0 \leq n \to \varphi(0)$ and $\mathcal{M} \vDash \forall i\,(\varphi(i) \wedge i < n \to \varphi(si)) \to j \leq n \to \varphi(j)$ by assumption, we thus obtain $(\mathcal{M}, v) \vDash n \leq n \to \varphi(n)$. Because (O2) is satisfied in $\mathcal{M}$, $(\mathcal{M}, v) \vDash \varphi(n)$. □

## 1.5 Lattices

**Definition 1.20** (Lattice)**.** A *lattice* is a partially ordered set $\langle L, \leq \rangle$ in which any two elements have supremum and infimum, i.e., for $x, y \in P$ there are $s, i \in L$ such that

$$s \geq x \wedge s \geq y \wedge \forall s'.\ s' \geq x \wedge s' \geq y \to s' \geq s,$$
$$i \leq x \wedge i \leq y \wedge \forall i'.\ i' \leq x \wedge i' \leq y \to i' \leq i.$$

The supremum and infimum of $x$ and $y$ are typically written as $x \vee y$ and $x \wedge y$, respectively. It is easy to see that $\vee$ and $\wedge$ are associative, commutative, and idempotent.

A lattice $\langle L, \leq \rangle$ is called *complete* if every subset of $L$ has supremum and infimum. It is immediately clear that every complete lattice has both a least and a greatest element.

**Proposition 1.21.** *Let $\langle L, \leq \rangle$ be a partially ordered set in which all subsets have infima. Then $\langle L, \leq \rangle$ is a complete lattice.*

*Proof.* Let $M \subseteq L$. We only need to show that the supremum of $M$ exists. Let $S = \{\, x \in L \mid \forall y \in M\ y \leq x \}$ be the set of upper bounds of $M$ and $s = \inf S$. We claim that $s = \sup M$. Since every $m \in M$ is a lower bound of $S$ and $s$ is the greatest lower bound of $S$, we have $\forall m \in M\ m \leq s$, which means that $s$ is an upper bound of $M$. That it is the least upper bound is immediate from its definition. □

**Definition 1.22** (Ascending chain condition)**.** A partially ordered set $P$ satisfies the *ascending chain condition* (ACC) if there is no infinite strictly ascending sequence $x_0 < x_1 < \ldots$ in $P$.

An equivalent formulation of the ACC is that every infinite weakly ascending sequence $x_0 \leq x_1 \leq \ldots$ must be eventually constant, i.e., there must be some $i \in \mathbb{N}$ such that $x_k = x_i$ for all $k \geq i$.

**Example 1.23.** Let $V$ be a vector space (over any field). Consider the partially ordered set $\langle \operatorname{Lin} V, \subseteq \rangle$ of linear subspaces of $V$. Since $A \subseteq V$ is a linear subspace

of $V$ iff it contains 0 and is closed under linear combinations, it follows that for nonempty $M \subseteq \operatorname{Lin} V$, $\bigwedge M = \bigcap M \in \operatorname{Lin} V$. Moreover, $\bigwedge \emptyset = V \in \operatorname{Lin} V$. By 1.21, $\operatorname{Lin} V$ is therefore a complete lattice.

If $\dim V = n \in \mathbb{N}$, then $\operatorname{Lin} V$ satisfies ACC. To see this, observe that dim is a strictly monotone function from $\operatorname{Lin} V$ to $[0, n] \subseteq \mathbb{N}$ and $[0, n]$, as a finite set, satisfies ACC. Since a strictly monotone function preserves strictly ascending sequences, $\operatorname{Lin} V$ must satisfy ACC as well.

We will later make use of the following famous theorem. A proof can be found in [Tar55].

**Theorem 1.24** (Knaster-Tarski)**.** *Let $\langle P, \leq \rangle$ be a complete lattice and let $F\colon P \to P$ be a monotone function. Then the set $\{ x \in P \mid F(x) = x \}$ of fixed points of $F$ is a complete lattice.*

# CHAPTER 2

# FORMULA EQUATIONS, INDUCTION PROOFS, AND GRAMMARS

## 2.1 Formula equations

We now introduce the central concepts of this thesis: formula equations and solution problems. For the definition, we briefly recall second-order logic. In second-order logic, one may quantify over functions and predicates. Consequently, second-order formulas can contain function and predicate variables. These are typed in exactly the same way as the constant function and predicate symbols of first-order logic; for instance, we can have a predicate variable $X$ of type $\sigma_1 \times ... \times \sigma_n$. Function variables work analogously, but we will not be needing them.

**Definition 2.1** (Formula equation)**.** Let $\mathcal{L}$ be a first-order language over the set of sorts $\mathcal{S}$.

1. A *formula variable* of type $\sigma_1 \times ... \times \sigma_n$ is a second-order predicate variable $X$ of type $\sigma_1 \times ... \times \sigma_n$.
2. If $X : \sigma_1 \times ... \times \sigma_n$ is a formula variable, then we call atomic formulas of the form $X(t_1, ..., t_n)$ *variable atoms*. By contrast, we may refer to conventional

first-order atoms as *constant atoms*.

3. Let $\varphi$ be a formula such that
   - $\varphi$ is first-order closed, i.e., it contains no free first-order variables, and
   - $\varphi$ may contain variable atoms in addition to constant $\mathcal{L}$-atoms.

   Let $X_1, ..., X_n$ be the formula variables occurring in $\varphi$. Then the second-order formula $\Phi \equiv \exists X_1, ..., X_n\, \varphi$ is called an *$\mathcal{L}$-formula equation*. The formula variables $X_1, ..., X_n$ are called the *unknowns* of $\Phi$ and $\varphi$ is called the *matrix* of $\Phi$.

We say that a formula equation $\Phi$ is quantifier-free, $\Pi_n$, $\Sigma_n$ if its matrix has the respective property.

**In the sequel, when we write a formula equation as $\exists \bar{X}\, \varphi$, we mean that $\varphi$ is the matrix of the formula equation.**

**Definition 2.2** (Solution of a formula equation)**.** Let $\Phi \equiv \exists \bar{X}\, \varphi$ be an $\mathcal{L}$-formula equation with unknowns $X_1, ..., X_n$ and $\mathcal{T}$ an $\mathcal{L}$-theory. A second-order substitution $\tau = [X_1 \backslash \psi_1, ..., X_n \backslash \psi_n]$, where $\psi_1, ..., \psi_n$ are $\mathcal{L}$-formulas of appropriate types, is a *$\mathcal{T}$-solution* of $\Phi$ if $\mathcal{T} \models \varphi\tau$.

We refer to these second-order formulas as "equations" because there is a parallel with algebraic equations and their solutions. Note that $\tau$ is a $\mathcal{T}$-solution of $\exists \bar{X}\, \varphi$ iff $\mathcal{T} \models \varphi\tau \leftrightarrow \top$. Thus, $\tau$ assigns "values" to the variables of $\varphi$ in such a way that $\varphi$ becomes "equal" to $\top$ (modulo $\mathcal{T}$). This is analogous to how a solution of an ordinary equation $t(x_1, ..., x_n) = 0$ is a substitution of the variables in $t$ by values such that $t$ becomes equal to 0.

**Example 2.3.** $\Phi \equiv \exists X.\ \forall n.\ X(2 \cdot n) \wedge (X(n) \rightarrow \neg X(sn))$ is a $\mathcal{L}_A$-formula equation with one unary unknown. $[X(n) \backslash \psi(n) \equiv \exists k\ n = 2 \cdot k]$ is a **PA**-solution of $\Phi$ because

$$\mathbf{PA} \models \forall n.\ \exists k\ 2 \cdot n = 2 \cdot k \wedge (\exists k\ n = 2 \cdot k \rightarrow \nexists k\ sn = 2 \cdot k).$$

**Definition 2.4** (Solution problem)**.** Let $\mathcal{L}$ be a first-order language. A solution problem over $\mathcal{L}$ has three components:

1. An $\mathcal{L}$-theory $\mathcal{T}$, called the background theory;

2. A class $\mathcal{F}$ of $\mathcal{L}$-formula equations;
3. A class $\mathcal{C}$ of $\mathcal{L}$-formulas (the candidate solutions).

Given $\mathcal{T}, \mathcal{F}, \mathcal{C}$ as above, the *solution problem* $\langle \mathcal{T}, \mathcal{F}, \mathcal{C} \rangle$ is the set of formula equations in $\mathcal{F}$ that have $\mathcal{T}$-solutions in $\mathcal{C}$, i.e., the set

$$\left\{ \exists \bar{X}\, \varphi \in \mathcal{F} \;\middle|\; \exists \bar{\psi} \in \mathcal{C} \text{ s.t. } \mathcal{T} \models \varphi[X_1 \backslash \psi_1, ..., X_n \backslash \psi_n] \right\}.$$

If $\mathcal{P} = \langle \mathcal{T}, \mathcal{F}, \mathcal{C} \rangle$ is a solution problem, we call the formula equations in $\mathcal{F}$ *instances* of $\mathcal{P}$ and those that are in $\mathcal{P}$ *positive instances* of $\mathcal{P}$.

When the class of candidate solutions that we have in mind is clear, for instance in the context of discussing a particular solution problem, we will leave it implicit. Thus, in such a context, by the question "does $\Phi$ have a solution?" we actually mean "does $\Phi$ have a solution in the class $\mathcal{C}$?".

**Example 2.5.** Consider the solution problem $\mathcal{P} = \langle \mathbf{PA}, \mathcal{F}, \mathcal{C} \rangle$, where $\mathcal{C}$ is the set of purely existential formulas of $\mathcal{L}_A$ and $\mathcal{F}$ is the set of all $\mathcal{L}_A$-formula equations. Then $\Phi$ from Example 2.3 is a positive instance of $\mathcal{P}$, since $\Phi \in \mathcal{F}$, $\psi(n) \in \mathcal{C}$, and $[X(n) \backslash \psi(n)]$ is a $\mathbf{PA}$-solution of $\Phi$.

Let us give some more general examples of the expressive power of formula equations.

**Example 2.6.**

1. Let $\mathcal{L}$ be any first-order language and $\mathcal{F}$ the set of formula equations without unknowns—in other words, closed first-order formulas—over $\mathcal{L}$. The solution problem $\langle \emptyset, \mathcal{F}, \emptyset \rangle$ is the problem of first-order validity.
2. Let $\mathcal{C} = \{\top, \bot\}$ and $\mathcal{F}$ the set of formula equations that contain only nullary formula variables and propositional connectives. Then $\langle \emptyset, \mathcal{F}, \mathcal{C} \rangle$ is SAT.
3. Provability in Peano arithmetic can be obtained as a solution problem as follows. A formula $\varphi$ is provable in $\mathbf{PA}$ iff $\varphi$ is provable from a single induction axiom and the axioms of a finitely axiomatizable base theory such as Robinson arithmetic $\mathbf{Q}$ in first-order logic, see e.g. [HW18, Corollary 2.1]. Let $\mathcal{C}$ be the class of all first-order formulas over $\mathcal{L}_A$. Furthermore, let $X$ be a unary formula variable, $\mathrm{IND}_X$ the induction axiom for $X$ (cf. Definition

1.18), and $\mathcal{F}$ the set of all formula equations of the form $\exists X.\, Q \wedge \mathrm{IND}_X \to \varphi$, where $Q$ is the conjunction of the axioms of Robinson arithmetic and $\varphi$ is a closed $\mathcal{L}_A$-formula. Then $\langle \emptyset, \mathcal{F}, \mathcal{C} \rangle$ is the problem of provability in Peano arithmetic.

These examples show that a large variety of questions can be formulated as solution problems, among them some that are undecidable or whose algorithmic solution poses considerable difficulties in practice. We are particularly interested in the decidability of loop invariant generation.

## 2.2 Quantified solution problems

Here we will show that $\Pi_1$ and $\Sigma_1$ formula equations are generally unsolvable. The proof originally appeared in [EHW17]; we present an updated version that is based on many-sorted logic and contains more details.

First, we need to define a specific language.

**Definition 2.7** ($\mathcal{L}_{PCP}$)**.** $\mathcal{L}_{PCP}$ is the language containing

- the sorts **Nat**, **BS**, and **IS**;
- the constant symbols $0 \colon \mathbf{Nat}$, $\varepsilon_B \colon \mathbf{BS}$, and $\varepsilon_I \colon \mathbf{IS}$;
- the function symbols

$$\begin{aligned} s &\colon \mathbf{Nat} \to \mathbf{Nat}, & s_0, s_1 &\colon \mathbf{BS} \to \mathbf{BS}, \\ \mathbf{lw}, \mathbf{rw} &\colon \mathbf{IS} \to \mathbf{BS}, & \mathbf{len} &\colon \mathbf{IS} \to \mathbf{Nat}, \\ p_i &\colon \mathbf{IS} \to \mathbf{IS} \text{ for every } i \geq 1; \end{aligned}$$

- and the predicate symbol $P \colon \mathbf{Nat}$.

The intended interpretation of $\mathcal{L}_{PCP}$ is as follows: The sort **Nat** represents the natural numbers, **BS** represents finite binary sequences, and **IS** represents finite sequences of natural numbers. The symbols **lw**, **rw**, **len**, and $P$ will be explained later.

We extend the usual notation $\bar{\cdot}$ for numerals to other terms of $\mathcal{L}_{\mathcal{P}}$:

$$\begin{aligned} \bar{k} &= s^k 0 \colon \mathbf{Nat} && \text{for } k \in \mathbb{N}, \\ \overline{a_1 \cdots a_k} &= s_{a_1} \cdots s_{a_k} \varepsilon_B \colon \mathbf{BS} && \text{for } a_1, ..., a_k \in \{0,1\}, \\ \overline{(q_1, ..., q_k)} &= p_{q_1} \cdots p_{q_k} \varepsilon_I \colon \mathbf{IS} && \text{for } q_1, ..., q_k \in \{1, ..., n\}. \end{aligned}$$

Thus, we can encode natural numbers, binary strings, and sequences of numbers in $\{1, ..., n\}$. Note that in all cases, equality of the encodings implies equality of the original objects. Moreover, it is easy to see that the three encodings cover the entirety of Terms($\mathcal{L}_{PCP}$) apart from those containing **lw**, **rw**, or **len**.

If $s = a_1 \cdots a_k$ is a binary string and $t \colon \mathbf{BS}$ is an $\mathcal{L}_{PCP}$-term, then we write $s * t$ for the $\mathcal{L}_{PCP}$-term $s_{a_1} \cdots s_{a_k} t \colon \mathbf{BS}$.

**Definition 2.8** (The existential and universal solution problems)**.** Let $\mathcal{C}$ be the class of quantifier-free formulas over $\mathcal{L}_{PCP}$, $\mathcal{E}$ the set of $\Sigma_1$ formula equations and $\mathcal{U}$ the set of $\Pi_1$ formula equations over $\mathcal{L}_{PCP}$. Then the solution problem $\langle \emptyset, \mathcal{E}, \mathcal{C} \rangle$ is called the *existential solution problem* and $\langle \emptyset, \mathcal{U}, \mathcal{C} \rangle$ is called the *universal solution problem.*

Both the existential and the universal solution problem are undecidable, but the proofs are very different. The existential solution problem is quickly dealt with.

**Theorem 2.9.** *The existential solution problem is undecidable.*

*Proof.* By reduction from the validity problem of first-order logic.

By application of validity-preserving skolemization $\mathrm{sk}^+$ and prenexification, the validity problem of first-order logic reduces to the validity problem of formulas of the form $\exists \bar{x}\, \varphi$ with $\varphi$ quantifier-free. This problem, in turn, obviously reduces to the existential solution problem. Since the first-order validity problem is undecidable, the existential solution problem must be undecidable as well. □

We will prove the undecidability of the universal solution problem by a reduction from the *Post Correspondence Problem* (PCP). Let us first define PCP.

**Definition 2.10** (Modified Post Correspondence Problem)**.** An instance of the Post Correspondence Problem is a sequence $\mathcal{P} = (v_1, w_1), ...(v_n, w_n)$, where $n \geq 2$,$v_i, w_i \in \{0,1\}^*$, $v_1 \neq w_1$ and $v_1, w_1 \neq \varepsilon$.

A solution of an instance $\mathcal{P}$ of the PCP is a sequence $q_1, ..., q_m$ such that

$$v_{q_1}...v_{q_m}v_1 = w_{q_1}...w_{q_m}w_1 \tag{2.1}$$

**Remark.** The "modification" is the requirement that both sides in (2.1) must end at index 1.

PCP is a famous example of an undecidable problem.

**Theorem 2.11.** *The modified PCP is undecidable.*

A proof can be found in [Sip12].

Our aim is now to construct, for a given instance $\mathcal{P}$ of the modified PCP, an instance $\exists X \forall \bar{y}\, \varphi_{\mathcal{P}}(X, y)$ of the universal solution problem that has a solution iff $\mathcal{P}$ does.

Every instance of the modified PCP induces a theory.

**Definition 2.12** ($\mathcal{T}_{\mathcal{P}}$)**.** Let $\mathcal{P} = (v_1, w_1), ...(v_m, w_m)$ be an instance of the modified PCP. Then $\mathcal{T}_{\mathcal{P}}$ is the set containing the following $\mathcal{L}_{PCP}$-formulas:

$$\begin{aligned}
\mathrm{LW}_0 &\equiv \mathbf{lw}\varepsilon_I = \overline{v_1}, & \mathrm{RW}_0 &\equiv \mathbf{rw}\varepsilon_I = \overline{w_1},\\
\mathrm{LW}_1(x) &\equiv \mathbf{lw}(p_1 x) = v_1 * \mathbf{lw}x, & \mathrm{RW}_1(x) &\equiv \mathbf{rw}(p_1 x) = w_1 * \mathbf{rw}x,\\
&\vdots & &\vdots\\
\mathrm{LW}_m(x) &\equiv \mathbf{lw}(p_m x) = v_m * \mathbf{lw}x, & \mathrm{RW}_m(x) &\equiv \mathbf{rw}(p_m x) = w_m * \mathbf{rw}x
\end{aligned}$$

$$\begin{aligned}
\mathrm{LEN}_0 &\equiv \mathbf{len}\,\varepsilon_I = 0, & \mathrm{NS}(x) &\equiv \mathbf{lw}x \neq \mathbf{rw}x,\\
\mathrm{LEN}_1(x) &\equiv \mathbf{len}(p_1 x) = s(\mathbf{len}x),\\
&\vdots\\
\mathrm{LEN}_m(x) &\equiv \mathbf{len}(p_m x) = s(\mathbf{len}x).
\end{aligned}$$

If $t\colon \mathbf{IS}$ is a term, we write $\mathcal{T}_{\mathcal{P}}(t)$ for the set $\{\, \psi(t) \mid \psi \in \mathcal{T}_{\mathcal{P}} \}$.

Clearly, **lw** and **rw** represent the functions that map a sequence $(p_{i_1}, ..., p_{i_q})$ to $v_{i_1} \cdots v_{i_q} v_1$ and $w_{i_1} \cdots w_{i_q} v_1$, respectively, and **len** represents the function that maps such a sequence to its length $q$.

We can view the formulas $\mathrm{LW}_0$ – $\mathrm{LW}_m$, $\mathrm{RW}_0$ – $\mathrm{RW}_m$, and $\mathrm{LEN}_0$ – $\mathrm{LEN}_m$ as rewrite rules in the sense of Section 1.2. by reading them from left to right. Consequently, they induce a rewriting relation $\rightsquigarrow_{\mathcal{P}}$.

**Lemma 2.13.** *Let $\mathcal{P}$ be an instance of the modified PCP and $\rightsquigarrow_{\mathcal{P}}$ the rewriting relation induced by $\mathcal{T}_{\mathcal{P}}$. Then $\rightsquigarrow_{\mathcal{P}}$ is terminating and confluent.*

*Proof.* All reducible terms are of the form $\mathbf{lw}(p_{i_k} \cdots p_{i_1} \varepsilon_I)$, $\mathbf{rw}(p_{i_k} \cdots p_{i_1} \varepsilon_I)$, or $\mathbf{len}(p_{i_k} \cdots p_{i_1} \varepsilon_I)$. We only consider the first case, the others are proved analogously. For $k = 0$, we have $\mathbf{lw}\varepsilon_I \rightsquigarrow_{\mathcal{P}} \overline{v_1}$ by the only applicable rewrite rule $\mathrm{LW}_0$. Now assume that every term of the form $\mathbf{lw}(p_{i_k} \cdots p_{i_1} \varepsilon_I)$ has a unique normal form $t$. Then $\mathbf{lw}(p_{i_{k+1}} p_{i_k} \cdots p_{i_1} \varepsilon_I) \rightsquigarrow_{\mathcal{P}} v_{i_{k+1}} * \mathbf{lw}(p_{i_k} \cdots p_{i_1} \varepsilon_I) \rightsquigarrow^*_{\mathcal{P}} v_{i_{k+1}} * t$. Again, there is only one applicable rule, so the normal form is unique. □

Let $\sim_{\mathcal{P}}$ be the relation on $\mathcal{L}_{PCP}$-terms defined by $s \sim_{\mathcal{P}} t$ iff $s$ and $t$ share a $\rightsquigarrow_{\mathcal{P}}$-normal form. Since $\rightsquigarrow_{\mathcal{P}}$ is strongly normalizing and confluent, $\sim_{\mathcal{P}}$ is an equivalence relation. We denote the equivalence class of $t$ modulo this relation by $[t]_{\mathcal{P}}$.

Note that all of the rewrite rules operate on terms starting with **lw**, **rw**, or **len**. This implies that all terms that do not contain any of these symbols are already in normal form. In particular, this includes all terms of the sort **IS**.

**Definition 2.14** (Standard model)**.** Let $\mathcal{P}$ be an instance of the modified PCP. A *$\mathcal{P}$-standard model* is an $\mathcal{L}_{PCP}$-structure $\mathcal{M}$ such that $M = \{\, [t]_{\mathcal{P}} \mid t \in \mathrm{Terms}(\mathcal{L}_{PCP}) \}$ and $t^{\mathcal{M}} = [t]_{\mathcal{P}}$ for all $\mathcal{L}_{PCP}$-terms $t$.

We call an $\mathcal{L}_{PCP}$-formula *$\mathcal{P}$-standard valid* if it holds in all $\mathcal{P}$-standard models. Conversely, a formula is called *$\mathcal{P}$-standard unsatisfiable* if it holds in no $\mathcal{P}$-standard model. As usual, we will drop the $\mathcal{P}$ if it is unambiguous.

For a given $\mathcal{P}$, how many $\mathcal{P}$-standard models are there, and what do they look like? The carrier sets of the three sorts are uniquely determined by the relation $\sim_{\mathcal{P}}$, which is itself uniquely determined by $\mathcal{P}$. Moreover, each of the function symbols $\mathbf{lw}, \mathbf{rw}, \mathbf{len}$ admits only one interpretation. The only remaining degree of freedom is in the interpretation of the predicate symbol $P$. It is easy to see that for any set $S \subseteq \mathbb{N}$, there is a $\mathcal{P}$-standard model $\mathcal{M}$ such that $P^{\mathcal{M}} = \left\{ \overline{i} \;\middle|\; i \in S \right\}$. Consequently, any formula not containing $P$ must either be standard valid or

standard unsatisfiable. Note also that each element of the sort **Nat** has exactly one representation—that is, member of the equivalence class—of the form $\bar{n}$ for some $n \in \mathbb{N}$. Analogously, each element of the sort **IS** has a unique representation of the form $\overline{a_1 \cdots a_q}$, where $a_i \in \{0, 1\}$. Meanwhile, the elements of the sort **IS** are equivalence classes with only a single member, each of which has the form $\overline{(p_{i_1}, ..., p_{i_q})}$ with $i_j \leq m$.

The next lemma shows why we are justified in using the name "standard model".

**Lemma 2.15.** *Let $\mathcal{P}$ be an instance of the modified PCP. Then the axioms* $\mathrm{LW}_0$ – $\mathrm{LW}_m$, $\mathrm{RW}_0$ – $\mathrm{RW}_m$ *of $\mathcal{T}_\mathcal{P}$ are $\mathcal{P}$-standard valid.* NS *is $\mathcal{P}$-standard valid iff $\mathcal{P}$ has no solution.*

*Proof.* The first part is obvious from the definition of a $\mathcal{P}$-standard model. For the second part, we first show that

$$[\mathbf{lw}\overline{(q_k, ..., q_1)}]_\mathcal{P} = \overline{v_{q_k} \cdots v_{q_1} v_1}$$

and

$$[\mathbf{rw}\overline{(q_k, ..., q_1)}]_\mathcal{P} = \overline{w_{q_k} \cdots w_{q_1} w_1}.$$

Again, we only prove the first statement. For $k = 0$ we obtain

$$[\mathbf{lw}\overline{()}]_\mathcal{P} = [\mathbf{lw}\varepsilon_I]_\mathcal{P} = \overline{v_1}.$$

Now assume $[\mathbf{lw}\overline{(q_k, ..., q_1)}]_\mathcal{P} = \overline{v_{q_k} \cdots v_{q_1} v_1}$. Then

$$\begin{aligned}
[\mathbf{lw}\overline{(q_{k+1} q_k, ..., q_1)}]_\mathcal{P} &= v_{q_{k+1}} * [\mathbf{lw}\overline{(q_k, ..., q_1)}]_\mathcal{P} \\
&= v_{q_{k+1}} * \overline{v_{q_k} \cdots v_{q_1} v_1} \\
&= \overline{v_{q_{k+1}} \cdots v_{q_1} v_1}.
\end{aligned}$$

Now let $\mathcal{M}$ be a $\mathcal{P}$-standard model. It follows that

$$\mathcal{M} \models \mathbf{lw}\overline{(q_k, ..., q_1)} = \overline{v_{q_k} \cdots v_{q_1} v_1} \wedge \mathbf{rw}\overline{(q_k, ..., q_1)} = \overline{w_{q_k} \cdots w_{q_1} w_1}.$$

Assume that $(q_k, ..., q_1)$ is a solution of $\mathcal{P}$. Then $\overline{v_{q_k} \cdots v_{q_1} v_1} = \overline{w_{q_k} \cdots w_{q_1} w_1}$ and consequently

$$\begin{aligned}
\mathcal{M} \models \mathbf{lw}\overline{(q_k, ..., q_1)} &= \overline{v_{q_k} \cdots v_{q_1} v_1} \\
&= \overline{w_{q_k} \cdots w_{q_1} w_1} \\
&= \mathbf{rw}\overline{(q_k, ..., q_1)},
\end{aligned}$$

so $\mathcal{M} \not\models \mathrm{NS}$.

For the other direction, assume that $\mathcal{M} \not\models \mathrm{NS}$. Then there is $\tilde{t}\colon \mathbf{IS} \in \mathcal{M}$ such that $\mathcal{M} \vDash \mathbf{lw}\tilde{t} = \mathbf{rw}\tilde{t}$. By the definition of a standard model, $\tilde{t}$ is an equivalence class of $\sim_{\mathcal{P}}$, and since $\tilde{t}$ belongs to the sort **IS**, there is only one element $t$ in $\tilde{t}$. Clearly, $t$ must be of the form $\overline{(q_k \cdots q_1)}$ for some $q_1, ..., q_k \in \{1, ..., n\}$, so in fact

$$\begin{aligned} \mathcal{M} \vDash \overline{v_{q_k} \cdots v_{q_1} v_1} &= \mathbf{lw}\overline{(q_k, ..., q_1)} \\ &= \mathbf{rw}\overline{(q_k, ..., q_1)} \\ &= \overline{w_{q_k} \cdots w_{q_1} w_1}. \end{aligned}$$

It follows that $(q_k, ..., q_1)$ is a solution of $\mathcal{P}$. □

**Definition 2.16** ($\varphi_{\mathcal{P}}$)**.**

1. Let $t\colon \mathbf{IS}$ be an $\mathcal{L}_{PCP}$-term and $m, n \in \mathbb{N}$. Then

$$\mathcal{C}_n^m = \left\{ p_{i_1} \cdots p_{i_q} t \;\middle|\; q \in \{0, ..., n\}, i_q \in \{1, ..., m\} \right\}.$$

2. Let $\mathcal{P}$ be an instance of the modified PCP, $m$ the length of $\mathcal{P}$, and $\mathcal{T}_{\mathcal{P}}$ the theory induced by $\mathcal{P}$. Moreover, let $X\colon \mathbf{Nat} \times \mathbf{IS}$ be a formula variable. We define three sequents:

$$\begin{aligned} S_0^{\mathcal{P}}(X, \gamma) &\equiv \mathcal{T}_{\mathcal{P}}(\gamma), P(0) \vdash X(0, \gamma), \\ S_1^{\mathcal{P}}(X, \nu, \gamma) &\equiv \left\{ \begin{array}{c} \mathcal{T}_{\mathcal{P}}(\gamma), \{\, X(\nu, t) \mid t \in \mathcal{C}_1^m(\gamma) \}, \\ P(\mathbf{len}\,\gamma) \to P(s(\mathbf{len}\,\gamma)) \end{array} \right\} \vdash X(s\nu, \gamma), \\ S_2^{\mathcal{P}}(X, \alpha) &\equiv X(\alpha, \varepsilon_I) \vdash P(\alpha). \end{aligned}$$

   Then $\varphi_{\mathcal{P}}(X) \equiv \forall \alpha, \gamma, \nu.\ S_0^{\mathcal{P}}(X, \gamma) \wedge S_1^{\mathcal{P}}(X, \nu, \gamma) \wedge S_2^{\mathcal{P}}(X, \alpha)$.

The intended meaning of $X(\bar{n}, \bar{t})$ is "$t$ cannot be extended by $n$ terms such that the result is a solution of $\mathcal{P}$" and the intended meaning of $P(\bar{n})$ is "there is no solution of $\mathcal{P}$ of length $n$".

Before we perform the reduction, we define some auxiliary formulas.

**Definition 2.17.** Let $m, n \in \mathbb{N}$. For any $\mathcal{L}_{PCP}$-formula $\psi(y\colon \mathbf{Nat}, z\colon \mathbf{IS})$ and

variables $\nu, \gamma$, we define the sequents

$$\Theta_n^m(\psi,\nu,\gamma) \equiv \left\{ \begin{array}{c} \{\, P(s^i(\mathbf{len}\,\gamma)) \to P(s^{i+1}(\mathbf{len}\,\gamma)) \;\mid\; i < n\,\}, \\ \{\, \psi(\nu,t) \mid t \in \mathcal{C}_n^m(\gamma)\} \end{array} \right\} \vdash \psi(s^n\nu,\gamma),$$

$$\Sigma_n^m(\psi,\gamma) \equiv P(0), \{\, P(s^i(\mathbf{len}\,\gamma)) \to P(s^{i+1}(\mathbf{len}\,\gamma)) \;\mid\; i < n\} \vdash \psi(s^n 0,\gamma).$$

**Lemma 2.18.** *Let $\mathcal{P}$ be a negative instance of the modified PCP and let $m$ be the length of $\mathcal{P}$. If $\psi$ is a solution of $\exists X.\ \varphi_{\mathcal{P}}(X)$, then the sequents $\Theta_n^m(\psi,\nu,\gamma)$ and $\Sigma_n^m(\psi,\gamma)$ are $\mathcal{P}$-standard valid for all $n \in \mathbb{N}$.*

*Proof.* We first show the standard validity of $\Theta_n^m$. $\mathcal{S}_1(\psi,\nu,\gamma)$ is valid by assumption. By Lemma 2.15, this means that the sequent

$$S_1'(\psi,\nu,\gamma) \equiv P(\mathbf{len}\,\gamma) \to P(s(\mathbf{len}\,\gamma)), \{\, \psi(\nu,t) \mid t \in \mathcal{C}_1^m(\gamma)\} \vdash \psi(s\nu,\gamma)$$

is standard valid as well. In particular, for $k \in \mathbb{N}$ all sequents in

$$\mathcal{S}_k = \{\, S_1'(\psi, s^{n-k-1}\nu, u) \;\mid\; u \in \mathcal{C}_k(\gamma)\}$$

and the sequent $T \equiv S_1'(\psi, s^{n-1}\nu, \gamma)$ are standard valid. Consider $\mathcal{S}_1$:

$$\left\{ \begin{array}{c} P(\mathbf{len}\,\gamma) \to P(s(\mathbf{len}\,\gamma)), \\ \{\, \psi(s^{n-2}\nu,t) \mid t \in \mathcal{C}_1^m(\gamma)\} \end{array} \right\} \vdash \psi(s^{n-1}\nu,\gamma),$$

$$\left\{ \begin{array}{c} P(\mathbf{len}(p_1\gamma)) \to P(s(\mathbf{len}(p_1\gamma))), \\ \{\, \psi(s^{n-2}\nu,t) \mid t \in \mathcal{C}_1^m(p_1\gamma)\} \end{array} \right\} \vdash \psi(s^{n-1}\nu,p_1\gamma),$$

$$\vdots$$

$$\left\{ \begin{array}{c} P(\mathbf{len}(p_m\gamma)) \to P(s(\mathbf{len}(p_m\gamma))), \\ \{\, \psi(s^{n-2}\nu,t) \mid t \in \mathcal{C}_1^m(p_m\gamma)\} \end{array} \right\} \vdash \psi(s^{n-1}\nu,p_m\gamma)$$

Observe that all the formulas in the succedents of these sequents occur in the antecedent of $T$. This means that using the cut rule, we can show that the sequent

$$\{\, P(\mathbf{len}\,t) \to P(s(\mathbf{len}\,t)) \mid t \in \mathcal{C}_1^m\}, \{\, \psi(s^{n-2}\nu,t) \;\mid\; t \in \mathcal{C}_2^m(\gamma)\} \vdash \psi(s^n\nu,\gamma)$$

is standard valid. By repeating this process with the sets $\mathcal{S}_2, ..., \mathcal{S}_{n-1}$, we prove that

$$\Theta_n^m(\psi,\nu,\gamma) \equiv \left\{ \begin{array}{c} \{\, P(s^i(\mathbf{len}\,\gamma)) \to P(s^{i+1}(\mathbf{len}\,\gamma)) \;\mid\; i < n\,\}, \\ \{\, \psi(\nu,t) \mid t \in \mathcal{C}_n^m(\gamma)\} \end{array} \right\} \vdash \psi(s^n\nu,\gamma)$$

is standard valid.

For the standard validity of $\Sigma_n^m$, observe that by assumption and Lemma 2.15, the sequent

$$S_0'(\gamma) \equiv P(0) \vdash \psi(0, \gamma)$$

is standard valid. Now consider

$$\{\, S_0'(t) \mid t \in \mathcal{C}_1(\gamma)\} = \left\{ \begin{array}{lll} P(0) & \vdash & \psi(0,\gamma), \\ P(0) & \vdash & \psi(0,p_1\gamma), \\ & \vdots & \\ P(0) & \vdash & \psi(0,p_m\gamma) \end{array} \right\}.$$

By applying the cut rule to these sequents and $S_1'(\psi, 0, \gamma)$, we obtain the standard validity of $P(0), P(\mathbf{len}\,\gamma) \to P(s(\mathbf{len}\,\gamma)) \vdash \psi(s0, \gamma)$. By repeating the same process with $S_1'(\psi, s0, \gamma), ..., S_1'(\psi, s^{n-1}0, \gamma)$, we eventually arrive at the standard validity of

$$\Sigma_n^m(\psi, \gamma) \equiv P(0), \{\, P(s^i(\mathbf{len}\,\gamma)) \to P(s^{i+1}(\mathbf{len}\,\gamma)) \mid i < n\} \vdash \psi(s^n 0, \gamma).$$

□

**Lemma 2.19.** *Let $\mathcal{P}$ be an instance of the modified PCP and $\varphi$ a ground quantifier-free $\mathcal{L}_{PCP}$-formula.*

1. *Let $\varphi'$ be the formula obtained by replacing all standard-valid $=$-atoms in $\varphi$ by $\top$ and all standard-unsatisfiable $=$-atoms by $\bot$. Then $\varphi \leftrightarrow \varphi'$ is standard valid and $\varphi'$ only contains the predicate symbol $P$.*
2. *Assume that $\varphi$ contains only the predicate symbol $P$. Let $t, u \colon \mathbf{Nat}$ and let $\varphi'$ be the formula obtained by replacing all terms $t'$ in $\varphi$ such that $t' \sim_{\mathcal{P}} t$ by $u$. If $\varphi$ is standard valid, so is $\varphi'$.*

*Proof.* For 1, observe that every ground $=$-atom $t = u$ is either standard valid (if $t \sim_{\mathcal{P}} u$) or standard unsatisfiable (if $t \nsim_{\mathcal{P}} u$). The claim follows immediately.

For 2, we argue by contraposition. Let $\mathcal{M} \not\models \varphi'$. If the replacement turns a term $w$ into $w'$, there must be unique terms $r_w(x), v_w$ such that

$$\begin{aligned} v_w &\sim_{\mathcal{P}} t, \\ w &= r_w(v_w), \\ w' &= r_w(u). \end{aligned}$$

Now let $S = \{ [r_w(v_w)]_{\mathcal{P}} \mid [r_w(u)]_{\mathcal{P}} \in P^{\mathcal{M}} \}$ and let $\mathcal{N}$ be the standard model with $P^{\mathcal{N}} = S$. Then $\mathcal{N} \not\models \varphi$. □

**Lemma 2.20.** *Let $\mathcal{P}$ be an instance of the modified PCP, $t(x), u(x)\colon$ **Nat** terms, and $m, n, d \in \mathbb{N}$. Then $t(\bar{m}) = u(\bar{n}) \leftrightarrow t(\overline{m+d}) = u(\overline{n+d})$ is standard valid.*

*Proof.* Trivial. □

**Lemma 2.21.** *Let $\mathcal{P}$ be an instance of the modified PCP. If $\exists X\, \varphi_{\mathcal{P}}(X)$ is a positive instance of the universal solution problem, then $\mathcal{P}$ has a solution.*

*Proof.* By contradiction. Assume that $\psi(y, z)$ is a solution of $\exists X\, \varphi_{\mathcal{P}}(X)$ and $\mathcal{P}$ has no solution. Let $m$ be the size of $\mathcal{P}$, $k$ an upper bound on the sizes of terms in $\psi$ and $n, q_1, q_2 \in \mathbb{N}$ such that $n > k$ and $q_2 > q_1 > 2n$. By Lemma 2.18,

$$\begin{aligned}\Theta_n^m(\psi, \overline{q_1 - n}, \varepsilon_I) \equiv \{\, & P(s^i(\mathbf{len}\, \varepsilon_I)) \to P(s^{i+1}(\mathbf{len}\, \varepsilon_I)) \mid i < n \},\\ & \{\, \psi(\overline{q_1 - n}, t) \mid t \in \mathcal{C}_n^m(\varepsilon_I) \}\\ & \vdash \psi(\overline{q_1}, \varepsilon_I)\end{aligned}$$

is standard valid. Since $\mathbf{len}\, \varepsilon_I = 0$ holds in every standard model, this simplifies to

$$\{\, P(\bar{i}) \to P(\overline{i+1}) \mid i < n \}, \{\, \psi(\overline{q_1 - n}, t) \mid t \in \mathcal{C}_n^m(\varepsilon_I) \} \vdash \psi(\overline{q_1}, \varepsilon_I)$$

By Lemma 2.19, for each ground instance $\psi(u, t)$ of $\psi$ there is a formula $\kappa_{u,t}$ containing only the predicate symbol $P$ such that $\psi(u, t) \leftrightarrow \kappa_{u,t}$ is standard valid. Note that since we only replace $=$-atoms by $\top$ or $\bot$, $\kappa_{u,t}$ contains no terms that $\psi(u, t)$ does not already contain. Consequently, the sequent

$$\{\, P(\bar{i}) \to P(\overline{i+1}) \mid i < n \}, \{\, \kappa_{\overline{q_1 - n}, t} \mid t \in \mathcal{C}_n^m(\varepsilon_I) \} \vdash \kappa_{\overline{q_1}, \varepsilon_I}$$

is standard valid. Now consider what happens when we replace all occurrences of $\overline{q_1}$ in this sequent by $\overline{q_2}$. The antecedent cannot contain $\overline{q_1}$ due to our assumptions about $q_1, m$, and $n$, which only leaves $\kappa_{\overline{q_1}, \varepsilon_I}$. Here, the replacement yields $\kappa_{\overline{q_2}, \varepsilon_I}$, because due to Lemma 2.20, each atom $t(\overline{q_1}) = u(\overline{q_1})$ in $\psi(\overline{q_1}, \varepsilon_I)$ is standard valid (or standard unsatisfiable) if $t(\overline{q_2}) = u(\overline{q_2})$ in $\psi(\overline{q_2}, \varepsilon_I)$ is standard valid (or standard unsatisfiable). Therefore, we obtain the standard validity of

$$T \equiv \{\, P(\bar{i}) \to P(\overline{i+1}) \mid i < n \}, \{\, \kappa_{\overline{q_1 - n}, t} \mid t \in \mathcal{C}_n^m(\varepsilon_I) \} \vdash \kappa_{\overline{q_2}, \varepsilon_I}$$

again by Lemma 2.19. We also know that for each $t \in \mathcal{C}_n^m(\varepsilon_I)$, the sequent

$$\begin{aligned}\Sigma_{q_1-n}^m(\psi, t) \equiv & P(0), \{\, P(s^i(\mathbf{len}\, t)) \to P(s^{i+1}(\mathbf{len}\, t)) \mid i < q_1 - n\} \\ & \vdash \psi(\overline{q_1 - n}, t)\end{aligned}$$

is standard valid by Lemma 2.15. Like before, we can replace $\psi(\overline{q_1 - n}, t)$ by $\kappa_{\overline{q_1-n},t}$ without compromising standard validity. This allows us to cut these sequents with $T$, from which we obtain the standard valid sequent

$$P(0), \{\, P(\bar{i}) \to P(\overline{i+1}) \mid i < q_1\} \vdash \kappa_{\overline{q_2},\varepsilon_I}.$$

To see this, observe that for $t \in \mathcal{C}_n^m(\varepsilon_I)$, $\mathbf{len}\, t = \bar{k}$ for some $k \leq n$. Since the sequent $S_3(\psi, \overline{q_2}) \equiv \psi(\overline{q_2}, \varepsilon_I) \vdash P(\overline{q_2})$ is standard valid by assumption, it also follows that $P(0), \{\, P(\bar{i}) \to P(\overline{i+1}) \mid i < q_1\} \vdash P(\overline{q_2})$ is standard valid. But there is a standard model $\mathcal{M}$ such that $P^{\mathcal{M}} = \{\, [s^i 0]_{\mathcal{P}} \mid i < q_2\}$, which is a contradiction. □

**Definition 2.22** $(\mathbf{L}_n, \mathbf{R}_n, \mathbf{N}_n, \sigma_n)$**.** Let $\mathcal{P}$ be an instance of the modified PCP, $m$ the size of $\mathcal{P}$, and $n \in \mathbb{N}$. Then we define the sets

$$\begin{aligned}\mathbf{L}_n^{\mathcal{P}} &= \bigcup_{t \in \mathcal{C}_n^m(\varepsilon_I)} \{\mathrm{LW}_0, \mathrm{LW}_1(t), ..., \mathrm{LW}_m(t)\}, \\ \mathbf{R}_n^{\mathcal{P}} &= \bigcup_{t \in \mathcal{C}_n^m(\varepsilon_I)} \{\mathrm{RW}_0, \mathrm{RW}_1(t), ..., \mathrm{RW}_m(t)\}, \\ \mathbf{N}_n^{\mathcal{P}} &= \{\, \mathrm{NS}(t) \mid t \in \mathcal{C}_n^m(\varepsilon_I)\}\,.\end{aligned}$$

Furthermore, we define the formulas

$$\sigma_n^{\mathcal{P}}(z) \equiv P(0) \wedge \bigwedge_{i<n} (P(s^i(\mathbf{len}\, z)) \to P(s^{i+1}(\mathbf{len}\, z))) \wedge \bigwedge_{t \in \mathcal{C}_n^m(z)} \mathcal{T}_{\mathcal{P}}(t)$$

The following properties of $\sigma_n^{\mathcal{P}}$ are easy to prove.

**Lemma 2.23.** *Let $\mathcal{P}$ be an instance of the modified PCP.*

1. *If $k > n$, then $\sigma_k^{\mathcal{P}}(z) \to \sigma_n^{\mathcal{P}}(z)$ is valid.*
2. *$\sigma_n^{\mathcal{P}}(\varepsilon_I) \to P(\bar{n})$ is valid for all $n \in \mathbb{N}$.*
3. *$\sigma_n^{\mathcal{P}}(\varepsilon_I) \to \psi$ is valid for all $n \in \mathbb{N}$ and $\psi \in \mathbf{L}_n^{\mathcal{P}} \cup \mathbf{R}_n^{\mathcal{P}} \cup \mathbf{N}_n^{\mathcal{P}}$.*

**Lemma 2.24.** *Let $\mathcal{P}$ be an instance of the modified PCP. If $q_1, ..., q_n$ is a solution of $\mathcal{P}$, then $\neg\sigma_n^{\mathcal{P}}(\varepsilon_I)$ is valid.*

*Proof.* By contraposition. Assume that $\sigma_n^{\mathcal{P}}(\varepsilon_I)$ is valid. By Lemma 2.23, this implies the validity of $\mathbf{L}_n^{\mathcal{P}} \cup \mathbf{R}_n^{\mathcal{P}} \cup \mathbf{N}_n^{\mathcal{P}}$. It is easy to see that a solution of length $n$ would contradict the validity of this set. □

**Lemma 2.25.** *Let $\mathcal{P}$ be a positive instance of the modified PCP. Then $\exists X\, \varphi_{\mathcal{P}}(X)$ is a positive instance of the universal solution problem.*

*Proof.* Let $q_1, ..., q_n$ be a solution of $\mathcal{P}$ and $m$ the length of $\mathcal{P}$. We define a formula $\psi(y, z)$ as follows:

$$\begin{aligned} &(y = 0 \to \sigma_0^{\mathcal{P}}(z)) \\ \wedge\ &(y \neq 0 \wedge y = \bar{1} \to \sigma_1^{\mathcal{P}}(z)) \\ &\vdots \\ \wedge\ &(y \neq 0 \wedge ... \wedge y = \overline{n-1} \to \sigma_{n-1}^{\mathcal{P}}(z)) \\ \wedge\ &(y \neq 0 \wedge ... \wedge y \neq \overline{n-1} \to \sigma_n^{\mathcal{P}}(z)) \end{aligned}$$

We claim that $\psi(y, z)$ is a solution of $\exists X\, \varphi_{\mathcal{P}}(X)$, i.e., that the sequents

$$\begin{aligned} \forall z\, S_0(\psi, z) &\equiv \forall z.\ \mathcal{T}_{\mathcal{P}}(z), P(0) \vdash \psi(0, z), \\ \forall y, z\, S_1(\psi, y, z) &\equiv \forall y, z. \left\{ \begin{array}{c} \mathcal{T}_{\mathcal{P}}(z), \{\, \psi(y, t) \mid t \in \mathcal{C}_1^m(z) \}, \\ P(\mathbf{len}\, z) \to P(s(\mathbf{len}\, z)) \end{array} \right\} \vdash \psi(sy, z), \\ \forall y\, S_2(\psi, y) &\equiv \forall y.\ \psi(y, \varepsilon_I) \vdash P(y). \end{aligned}$$

are valid.

For $\forall z\, S_0(\psi, z)$, observe that $\psi(0, z)$ is equivalent to $P(0) \wedge \mathcal{T}_{\mathcal{P}}(z)$, so the validity of the sequent follows immediately.

For $\forall y, z\, S_1(\psi, y, z)$, we have to prove $\psi(sy, z)$ from the assumptions

$$\begin{gathered} \mathcal{T}_{\mathcal{P}}(z), \\ P(\mathbf{len}\, z) \to P(s(\mathbf{len}\, z)), \\ \{\, \psi(y, t) \mid t \in \mathcal{C}_1^m(z) \}. \end{gathered}$$

There are three cases to distinguish.

- $sy = 0$: $\psi(sy, z)$ is equivalent to $\sigma_0^{\mathcal{P}}(z)$. We further distinguish cases according to the value of $y$. If $y \neq 0 \wedge ... \wedge y \neq \overline{k-1}, y = \overline{k}$ for some $k < n$, then clearly $\psi(y, z) \Leftrightarrow \sigma_k^{\mathcal{P}}(z)$, and hence $\psi(sy, z)$ follows from $\psi(y, z)$ by Lemma 2.23. If $y \neq 0 \wedge ... \wedge y \neq \overline{n-1}$, then $\psi(y, z) \Leftrightarrow \sigma_n^{\mathcal{P}}(z)$ and the claim follows in the same way.
- $sy \neq 0 \wedge ... \wedge sy \neq \overline{k-1} \wedge sy = \overline{k}$ for some $1 \leq k < n$: This immediately implies $y \neq 0 \wedge ... \wedge y \neq \overline{k-2}, y = \overline{k-1}$. Now let $t \in \mathcal{C}_1^m(z)$. By assumption, we know $\psi(y, t) \equiv \psi(\overline{k-1}, t)$. In particular, this implies $\mathbf{len}\, t = \mathbf{len}\, z \vee \mathbf{len}\, t = s(\mathbf{len}\, z)$. Thus, we know

  $$\sigma_{k-1}^{\mathcal{P}}(t) \equiv P(0) \wedge \bigwedge_{i<k-1} (P(\mathbf{len}\, t) \to P(s(\mathbf{len}\, t))) \wedge \bigwedge_{u \in \mathcal{C}_{k-1}^m(t)} \mathcal{T}_{\mathcal{P}}(u)$$

  for all $t \in \mathcal{C}_1^m(z)$. It is easy to see that this implies

  $$P(0) \wedge \bigwedge_{i<k} (P(\mathbf{len}\, z) \to P(s(\mathbf{len}\, z))) \wedge \bigwedge_{u \in \mathcal{C}_k^m(z)} \mathcal{T}_{\mathcal{P}}(u)$$

  by the definition of $\mathcal{C}_n^m$. We have thus established $\sigma_k^{\mathcal{P}}(z)$, which implies $\psi(sy, z)$ because $sy = \overline{k}$.
- $sy \neq 0 \wedge ... \wedge sy \neq \overline{n-1}$: By a similar argument to the previous case, we can deduce $\sigma_{n-1}^{\mathcal{P}}(t) \vee \sigma_n^{\mathcal{P}}(t)$ for all $t \in \mathcal{C}_1^m(z)$. This formula implies $\sigma_{n-1}^{\mathcal{P}}(t)$ by Lemma 2.23. We deduce $\sigma_n^{\mathcal{P}}(z)$ and hence $\psi(sy, z)$ in the same way as in the previous case.

Finally, for $\forall y\, S_3(\psi, y)$, we have to prove $P(y)$ from $\psi(y, \varepsilon_I)$. If $y = \overline{i}$ for $i < n$, then $\psi(y, \varepsilon_I) \Rightarrow \sigma_i^{\mathcal{P}}(\varepsilon_I)$, which implies $P(\overline{i})$ and hence $P(y)$. On the other hand, if $y \neq 0 \wedge ... \wedge y \neq \overline{n-1}$, then $\psi(y, \varepsilon_i)$ implies $\sigma_n^{\mathcal{P}}(\varepsilon_I)$. But by Lemma 2.24, $\neg\sigma_n^{\mathcal{P}}(\varepsilon_I)$ is valid, so we can deduce $P(y)$ by ex falso quodlibet. □

## 2.3 Inductive theorem proving

Inductive theorem proving, or the task of automatically finding proofs containing induction inferences, is a prime example of a problem that can be modeled with formula equations. In this section, we will define the problem and show how it can be represented as a solution problem with quantifier-free candidate solutions.

Tree grammars, which we briefly discuss in Section 2.4, play an integral role in the translation.

For any formula $\varphi \equiv Qx_1 ... Qx_n \psi$, where $Q_i \in \{\forall, \exists\}$ and $\psi$ is quantifier-free, and terms $t_1, ..., t_i$, we call $\psi[\bar{x} \backslash \bar{t}]$ an *instance* of $\varphi$. We call a sequent $\Gamma \vdash \Delta$ such that all formulas in $\Gamma$ are $\Pi_1$ and all formulas in $\Delta$ are $\Sigma_1$ a $\Sigma_1$ *sequent*.

**Definition 2.26** (Herbrand sequent)**.** Let $\Gamma \vdash \Delta$ be a $\Sigma_1$ sequent. Then an *Herbrand sequent* of $\Gamma \vdash \Delta$ is a sequent $\Gamma' \vdash \Delta'$ such that

- $\Gamma'$ comprises instances of formulas in $\Gamma$ and $\Delta'$ comprises instances of formulas in $\Delta$;
- $\models \bigwedge \Gamma' \to \bigvee \Delta'$.

We present the following version of *Herbrand's Theorem* without proof.

**Theorem 2.27.** *Let $\Gamma \vdash \Delta$ be a $\Sigma_1$ sequent. Then $\models \Gamma \vdash \Delta$ iff $\Gamma \vdash \Delta$ has an Herbrand sequent.*

Later, we will need to consider Herbrand sequents as sets of terms. Therefore, we introduce new function symbols for all formulas in the original $\Sigma_1$ sequent, which allows us to translate from the formula to the term level. To this end, we assume that $\mathbf{o}$ is a fresh sort symbol.

**Definition 2.28** (Function symbols for formulas)**.** Let

$$\forall x_1 : \sigma_1, ..., x_n : \sigma_n \, \varphi(\bar{x})$$

with $\varphi$ quantifier-free be a closed $\Pi_1$-formula. Then $[\forall \bar{x} \, \varphi(\bar{x})]$ is a fresh function symbol of the sort $\sigma_1 \times ... \times \sigma_n \to \mathbf{o}$. The symbol $[\exists \bar{x} \, \varphi(\bar{x})]$ is defined analogously.

The symbol $[\forall \bar{x} \, \varphi(\bar{x})]$ allows us to represent instances of $\forall \bar{x} \, \varphi(\bar{x})$ as terms: the instance $\varphi(t_1, ..., t_n)$ is represented by the term $[\forall \bar{x} \, \varphi(\bar{x})](t_1, ..., t_n)$.

**Definition 2.29** (Term set of a Herbrand sequent)**.** Let

$$\Gamma \vdash \Delta \equiv \forall \bar{x}_1 \, \gamma_1(\bar{x}_1), ..., \forall \bar{x}_m \, \gamma_m(\bar{x}_m) \vdash \exists \bar{x}_1 \, \delta_1(\bar{x}_1), ..., \exists \bar{x}_n \, \delta_n(\bar{x}_n)$$

be a $\Sigma_1$ sequent and $H \equiv \Gamma' \vdash \Delta'$ an Herbrand sequent of $\Gamma \vdash \Delta$. Then the *term set* of $H$ is the set

$$T(H) = \left\{ [\forall \bar{x}_i \, \gamma_i(\bar{x}_i)](\bar{t}) \;\middle|\; \gamma_i(\bar{t}) \in \Gamma' \right\} \cup \left\{ [\exists \bar{x}_j \, \delta_i(\bar{x}_j)](\bar{t}) \;\middle|\; \delta_j(\bar{t}) \in \Delta' \right\}.$$

In practice, we will only consider $\Sigma_1$ sequents $\Gamma \vdash \Delta$ where $\Delta$ only contains a single quantifier-free formula $\varphi$. Consequently, any Herbrand sequent $H$ of $\Gamma \vdash \Delta$ will have the form $\Gamma' \vdash \varphi$ and the term set of $H$ reduces to

$$T(H) = \{ [\forall \bar{x}\, \gamma(\bar{x})](\bar{t}) \mid \gamma(\bar{t}) \in \Gamma' \} \cup \{[\varphi]\}.$$

We define provability using induction via an **LK** rule. A conventional induction rule looks like the following:

$$\frac{\Gamma \vdash \Delta, \varphi(0) \qquad \varphi(\nu), \Pi \vdash \Lambda, \varphi(s\nu)}{\Gamma, \Pi \vdash \Lambda, \varphi(\alpha)}$$

However, for technical reasons, we will choose a slightly different rule in the spirit of Lemma 1.19.

**Definition 2.30** (Induction rule)**.** Let $\mathcal{L}$ be an arithmetical language and $\varphi(\alpha)$ an $\mathcal{L}$-formula. The *induction rule* is the inference rule

$$\frac{\Gamma \vdash \Delta, \varphi(0) \qquad \varphi(\nu), \nu < \alpha, \Pi \vdash \Lambda, \varphi(s\nu)}{\Gamma, \Pi \vdash \Lambda, \varphi(\alpha)}\ \text{ind}$$

When we say that a sequent is "provable in **LK**+ induction", we mean that it is provable using the **LK** rules given in Definition 1.14 plus the rule ind. We have shown in Lemma 1.19 that under fairly weak conditions (the axioms (O2) and (OS2)), the ind rule is equivalent to the conventional induction rule.

We will specifically be interested in proofs of the form

$$\dfrac{\dfrac{\begin{matrix}\pi_0 \\ \Gamma \vdash \varphi(\alpha, 0)\end{matrix} \quad \begin{matrix}\pi_1 \\ \varphi(\alpha,\nu), \nu < \alpha, \Pi \vdash \varphi(\alpha, s\nu)\end{matrix}}{\Gamma, \Pi \vdash \varphi(\alpha, \alpha)}\ \text{ind} \quad \begin{matrix}\pi_2 \\ \varphi(\alpha,\alpha), \Lambda \vdash \psi(\alpha)\end{matrix}}{\Gamma, \Pi, \Lambda \vdash \psi(\alpha)}\ \text{cut} \tag{2.2}$$

where $\psi$ is quantifier-free and $\pi_0, \pi_1, \pi_2$ contain no further induction inferences. To simplify things in the sequel, we codify this combination of induction and cut as an inference rule of its own.

**Definition 2.31** (Non-analytic induction rule)**.** The *non-analytic induction rule* is the inference rule

$$\frac{\Gamma \vdash \varphi(0) \qquad \varphi(\nu), \nu < \alpha, \Pi \vdash \varphi(s\nu) \qquad \varphi(\alpha), \Lambda \vdash \Delta}{\Gamma, \Pi, \Lambda \vdash \Delta}\,\text{ind}*$$

This rule is called "non-analytic" because the premises contain a formula that is not present in the end-sequent. It is obvious that in the presence of the cut rule, **LK** with ind is equivalent to **LK** with ind$*$.

Using the non-analytic induction rule, the proof in (2.2) can be written as

$$\frac{\begin{matrix}\pi_0 & \pi_1 & \pi_2\\ \Gamma \vdash \varphi(\alpha,0) & \varphi(\alpha,\nu), \nu < \alpha, \Pi \vdash \varphi(\alpha, s\nu) & \varphi(\alpha,\alpha), \Lambda \vdash \psi(\alpha)\end{matrix}}{\Gamma, \Pi, \Lambda \vdash \psi(\alpha)}\,\text{ind}* \tag{2.3}$$

Let us consider two special cases in more detail. First, let $\forall\Gamma, \forall\Pi, \forall\Lambda$ be sets of closed $\Pi_1$ formulas, $\Gamma, \Pi, \Lambda$ the sets of their matrices, and $\Gamma', \Pi', \Lambda'$ sets of instances of $\forall\Gamma, \forall\Pi, \forall\Lambda$.

Second, assume that $\varphi(\alpha, \nu) \equiv \forall y\, \varphi'(\alpha, \nu, y)$ with $\varphi'$ quantifier-free. In this case, we may assume that $\pi_0, \pi_1, \pi_2$ have the forms

$$\pi_0 = \dfrac{\dfrac{\begin{matrix}\pi'_0\\ \Gamma' \vdash \varphi'(\alpha, 0, \gamma)\end{matrix}}{\forall\Gamma \vdash \varphi'(\alpha, 0, \gamma)}\,\forall\text{:l, c:l}}{\forall\Gamma \vdash \forall y\, \varphi'(\alpha, 0, y)}\,\forall\text{:r}$$

$$\pi_1 = \dfrac{\dfrac{\begin{matrix}\pi'_1\\ \{\varphi'(\alpha, \nu, t_i)\}_{i=1}^m, \nu < \alpha, \Pi' \vdash \varphi'(\alpha, s\nu, \gamma)\end{matrix}}{\forall y\, \varphi'(\alpha, \nu, y), \nu < \alpha, \forall\Pi \vdash \varphi'(\alpha, s\nu, \gamma)}\,\forall\text{:l, c:l}}{\forall y\, \varphi'(\alpha, \nu, y), \nu < \alpha, \forall\Pi \vdash \forall y\, \varphi'(\alpha, s\nu, y)}\,\forall\text{:r}$$

$$\pi_2 = \dfrac{\begin{matrix}\pi'_2\\ \{\varphi'(\alpha, \alpha, u_i)\}_{i=1}^m, \Lambda' \vdash \psi(\alpha)\end{matrix}}{\forall y\, \varphi'(\alpha, \alpha, y), \forall\Lambda \vdash \psi(\alpha)}\,\forall\text{:l, c:l}$$

where a doubled inference line represents a block of several applications of the indicated rules. Let $\mathcal{S}_i$ denote the conclusion of $\pi'_i$ and $\forall\mathcal{S}_i$ the conclusion of $\pi_i$. Then by substituting into (2.3), we obtain

$$\frac{\begin{matrix}\pi'_0 & \pi'_1 & \pi'_2\\ \dfrac{\mathcal{S}_0}{\forall\mathcal{S}_0}* & \dfrac{\mathcal{S}_1}{\forall\mathcal{S}_1}* & \dfrac{\mathcal{S}_2}{\forall\mathcal{S}_2}*\end{matrix}}{\forall\Gamma, \forall\Pi, \forall\Lambda \vdash \psi(\alpha)}\,\text{ind}* \tag{2.4}$$

By an analogous argument, if the conditions on $\Gamma, \Pi, \Lambda$ remain the same, but $\varphi(\alpha, \nu) \equiv \exists y\, \varphi'(\alpha, \nu, y)$, we obtain a proof of the same form with the sequents

$$\begin{aligned}
\mathcal{S}_0 &\equiv \Gamma' \vdash \{\varphi'(\alpha, 0, u_i)\}_{i=1}^{m}, a \\
\forall\mathcal{S}_0 &\equiv \forall\Gamma \vdash \exists y\, \varphi'(\alpha, 0, y), \\
\mathcal{S}_1 &\equiv \left\{ \begin{array}{c} \varphi'(\alpha, \nu, \gamma), \\ \nu < \alpha, \Pi' \end{array} \right\} \vdash \{\varphi'(\alpha, s\nu, t_i)\}_{i=1}^{n}, \\
\forall\mathcal{S}_1 &\equiv \left\{ \begin{array}{c} \exists y\, \varphi'(\alpha, \nu, y), \\ \nu < \alpha, \forall\Pi \end{array} \right\} \vdash \exists y\, \varphi'(\alpha, s\nu, y), \\
\mathcal{S}_2 &\equiv \varphi'(\alpha, \alpha, \gamma), \Lambda' \vdash \psi(\alpha), \\
\forall\mathcal{S}_2 &\equiv \exists y\, \varphi'(\alpha, \alpha, y), \forall\Lambda \vdash \psi(\alpha).
\end{aligned}$$

Proofs of these two forms are obviously completely determined by the sets $\Gamma', \Pi', \Lambda'$, the formulas $\varphi'$ and $\psi'$, and the terms $t_1, ..., t_n$ and $u_1, ..., u_m$. This leads us to the following definition.

**Definition 2.32** (Simple induction proof)**.** Let $\mathcal{L}$ be an arithmetical language and $\mathcal{T}$ a theory over $\mathcal{L}$. Furthermore, let $\forall\Gamma$ be a set of closed $\Pi_1$-formulas. A *universal simple induction proof of* $\forall\Gamma \vdash \psi(\alpha)$ *over* $\mathcal{T}$ (universal $\mathcal{T}$-sip) is a triple of sequents

$$\begin{aligned}
\mathcal{S}_0 &\equiv \Gamma'(\alpha, \gamma) \vdash \varphi(\alpha, 0, \gamma), \\
\mathcal{S}_1 \equiv \Pi'(\alpha, \nu, \gamma), \nu < \alpha, \{\varphi(\alpha, \nu, t_i(\alpha, \nu, \gamma))\}_{i=1}^{n} &\vdash \varphi(\alpha, s\nu, \gamma), \\
\mathcal{S}_2 &\equiv \Lambda'(\alpha), \{\varphi(\alpha, \alpha, u_i(\alpha))\}_{i=1}^{m} \vdash \psi(\alpha),
\end{aligned}$$

where

- all formulas, terms, and sets are fully indicated,
- $\alpha, \nu$: **Nat**,
- $\Gamma', \Pi', \Lambda'$ comprise instances of $\forall\Sigma$,

such that $\mathcal{T} \vdash_{\mathbf{LK}} \mathcal{S}_i$ for $i \in \{0, 1, 2\}$.

Dually, an *existential simple induction proof of* $\forall\Gamma \vdash \psi(\alpha)$ *over* $\mathcal{T}$ (existential

$\mathcal{T}$-sip) is a triple of sequents

$$\begin{aligned} \mathcal{S}_0 &\equiv \Gamma'(\alpha) \vdash \{\varphi(\alpha, 0, u_i(\alpha))\}_{i=1}^m, \\ \mathcal{S}_1 &\equiv \Pi'(\alpha, \nu, \gamma), \nu < \alpha, \varphi(\alpha, \nu, \gamma) \vdash \{\varphi(\alpha, s\nu, t_i(\alpha, \nu, \gamma))\}_{i=1}^n, \\ \mathcal{S}_2 &\equiv \Lambda'(\alpha, \gamma), \varphi(\alpha, \alpha, \gamma) \vdash \psi(\alpha), \end{aligned}$$

with the same conditions as in the universal case.

In either case, the terms $t_1, ..., t_n$ are called the *step terms*, $u_1, ..., u_m$ are called the *base terms*, and $\varphi$ is called the *induction formula* of the sip.

The fact that we allow sips to depend on a background theory will prove useful in later in Chapter 3. It is clear that universal and existential simple induction proofs (over the empty background theory) can be used to reconstruct proofs of the form (2.4).

**Example 2.33.** Consider the language of arithmetic $\mathcal{L}_A$ augmented with the function symbols $f\colon \mathbf{Nat} \to \mathbf{Nat}$ and $g\colon \mathbf{Nat} \times \mathbf{Nat} \to \mathbf{Nat}$. Let

$$\begin{aligned} \forall\Gamma = \{&\forall x\, x \cdot s0 = x, \forall x\, s0 \cdot x = x, \\ &f(0) = s0, \forall x\, f(sx) = sx \cdot f(x), \\ &\forall x\, g(x, 0) = x, \forall x, y\, g(x, sy) = g(x \cdot sy, y), \\ &\forall x, y, z\, (x \cdot y) \cdot z = x \cdot (y \cdot z)\}, \end{aligned}$$

i.e., $\forall\Gamma$ contains some theorems of arithmetic and an axiomatization of two definitions $f\colon \mathbb{N} \to \mathbb{N}$ and $g\colon \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ of the factorial function given by

$$\begin{aligned} f(0) &= 1, & g(m, 0) &= m, \\ f(n+1) &= (n+1) \cdot f(n), & g(m, n+1) &= g(m \cdot (n+1), n). \end{aligned}$$

Furthermore, let $\psi \equiv \forall\Gamma \vdash g(1, \alpha) = f(\alpha)$ be the claim that the two definitions are equivalent. In order to prove this claim from the assumptions $\forall\Gamma$, we need the induction formula

$$\forall y\, \varphi(\nu, y) \equiv \forall y\, g(y, \nu) = y \cdot f(\nu),$$

of which $\psi$ is merely the special case with $y = 1$. Now let $\pi$ be the triple of sequents

$$\begin{aligned} \Gamma' &\vdash g(\gamma, 0) = \gamma \cdot f(0), \\ \Pi', \nu < \alpha, g(\gamma \cdot s\nu, \nu) = (\gamma \cdot s\nu) \cdot f(\nu) &\vdash g(\gamma, s\nu) = \gamma \cdot f(s\nu), \\ \Lambda', g(s0, \alpha) = s0 \cdot f(\alpha) &\vdash g(s0, \alpha) = f(\alpha) \end{aligned}$$

where

$$\begin{aligned} \Gamma' &= \{f(0) = s0, g(\gamma, 0) = \gamma, \gamma \cdot s0 = \gamma\}, \\ \Pi' &= \left\{ \begin{array}{c} f(s\nu) = s\nu \cdot f(\nu), \\ g(\gamma, s\nu) = g(\gamma \cdot s\nu, \nu), \\ (\gamma \cdot s\nu) \cdot f(\nu) = \gamma \cdot (s\nu \cdot f(\nu)) \end{array} \right\}, \\ \Lambda' &= \{s0 \cdot f(\alpha) = f(\alpha)\}. \end{aligned}$$

It is easy to see that $\Gamma', \Pi', \Lambda'$ are in fact valid and $\pi$ therefore is a simple induction proof of $\forall\Gamma \vdash \psi$ with the induction formula $\forall y\, \varphi$. Its sole base term is $\alpha$ and its sole step term is $\gamma \cdot s\nu$.

Given a set $\forall\Gamma$ of universally quantified axioms and a quantifier-free formula $\psi(\alpha)$, a question immediately presents itself: is there a universal (or existential) simple induction proof $\pi$ of $\psi$ with some induction formula $\forall y\, \varphi(\alpha, \nu, y)$ (or $\exists y\, \varphi(\alpha, \nu, y)$)? As Example 2.33 shows, $\varphi$ can be a nontrivial generalization of $\psi$. The approach taken towards answering this question in [EH15] is based on tree grammars, which we will consider now.

## 2.4 Parametric grammars

Let us restate the question at the end of the previous section: given $\forall\Gamma \vdash \psi(\alpha)$, is there a quantifier-free formula $\varphi$ such that there is a universal (respectively existential) simple induction proof $\pi$ with induction formula $\forall y\, \varphi$ (respectively $\exists y\, \varphi$)?

We give a high-level overview of the approach to the universal version of this question taken in [EH15]. First, we fix a $k \in \mathbb{N}$ and consider the instances $\forall\Gamma \vdash \varphi(0), ..., \forall\Gamma \vdash \varphi(k)$ of the goal sequent. If one of these instances is not valid,

then there is certainly no proof of the general sequent. Otherwise, these instance sequents have Herbrand sequents $H_0, ..., H_k$. The next step is to find a *parametric grammar* (see Definition 2.39) $G$ such that $L(G_i^{\forall}) \supseteq T(H_i)$. Intuitively, this means that there is some regularity in the term sets and $G$ generalizes this regularity. From $G$, in turn, we obtain a formula equation called a *simple induction proof schema* (see Definition 2.43).

Before we can introduce parametric grammars, we need some preliminary definitions.

**Definition 2.34** (Regular tree grammar)**.** A *regular tree grammar* is a tuple $\langle \tau, N, \Sigma, P \rangle$, where

- $N$ is a set of constant symbols (the set of *nonterminals*);
- $\Sigma$ is a set of constant and function symbols, disjoint from $N$ (the set of *terminals*);
- $\tau \in N$ is the *starting symbol*;
- $P$ is a set of *productions* over $N$ and $\Sigma$, that is, expressions of the form $\alpha \to t$ where $\alpha \in N$ and $t \in \text{Terms}(N \cup \Sigma)$.

Multiple productions $\alpha \to t_1, ..., \alpha \to t_n$ starting from the same nonterminal are typically written concisely as $\alpha \to t_1 | ... | t_n$.

The language of a regular tree grammar is defined via a *(derivability) relation* $\rightsquigarrow$. Let $G = \langle \tau, N, \Sigma, P \rangle$ be a regular tree grammar. Let $s \in \text{Terms}(N \cup \Sigma)$ and $\alpha \to t \in P$. Then $s \rightsquigarrow s[\alpha \backslash t]$. We denote the reflexive, transitive closure of $\rightsquigarrow$ by $\rightsquigarrow^*$.

**Definition 2.35** (Language of a regular tree grammar)**.** Let $G = \langle \tau, N, \Sigma, P \rangle$ be a regular tree grammar. The *language* of $G$, written as $L(G)$, is defined by

$$L(G) = \{\, t \in \text{Terms}(\Sigma) \mid \tau \rightsquigarrow^* t \,\}.$$

**Example 2.36.** Consider the regular tree grammar $G = \langle \varphi, N, \Sigma, P \rangle$ with

$$\begin{aligned} N &= \{\varphi, x, y\} \\ \Sigma &= \{a/0, b/0, g/1, f/2\} \\ P &= \{\varphi \to f(x, y), x \to a | g(y), y \to a | b\}. \end{aligned}$$

Then

$$L(G) = \{f(a,a), f(a,b), f(g(a),a), f(g(a),b), f(g(b),a), f(g(b),b)\}.$$

In particular, we will be considering *totally rigid acyclic* grammars. Total rigidity is defined via a restriction on the derivation relation, acyclicity by a condition on productions.

**Definition 2.37** (Totally rigid acyclic tree grammar)**.** We obtain *totally rigid acyclic tree grammars* by imposing the following additional restrictions on regular tree grammars:

- Total rigidity: for each nonterminal $\alpha$, at most one production of the form $\alpha \to t$ can occur in a derivation.
- Acyclicity: the nonterminals can be linearly ordered such that for each production $\alpha \to t$, all nonterminals in $t$ are greater than $\alpha$.

Note the difference between these conditions: acyclicity is a property that a given tree grammar either does or does not satisfy. By contrast, we can always view any tree grammar as a totally rigid one by restricting the notion of derivability.

**Example 2.38.** Let $G$ be the grammar from Example 2.36. It is easy to see that $G$ is acyclic if we put the nonterminals in the order $\varphi < x < y$. If we view it as a totally rigid grammar, its language is reduced to $\{f(a,a), f(a,b), f(g(a),a), f(g(b),b)\}$.

**Definition 2.39** (Parametric grammar)**.** Let $\mathcal{L}$ be an arithmetical language and $G$ a regular tree grammar with exactly the nonterminals $\tau, \alpha, \nu, \gamma, \gamma^*$. $G$ is a *parametric grammar* if each of its productions has one of these forms:

- $\tau \to t(\alpha, \nu, \gamma)$,
- $\gamma \to t(\alpha, \nu, \gamma)$,
- $\gamma^* \to t(\alpha)$,

where $t$ is an $\mathcal{L}$-term.

Out of context, these production forms seem arbitrary. In fact, they closely mimic the structure of simple induction proofs. Consequently, we can extract parametric grammars from both universal and existential sips.

**Definition 2.40** (Extracting parametric grammars from simple induction proofs)**.** Let $\pi$ be the universal simple induction proof

$$\begin{aligned} \mathcal{S}_0 &\equiv \Gamma'(\alpha,\gamma) \vdash \varphi(\alpha,0,\gamma),\\ \mathcal{S}_1 &\equiv \Pi'(\alpha,\nu,\gamma), \nu<\alpha, \{\varphi(\alpha,\nu,t_i(\alpha,\nu,\gamma))\}_{i=1}^n \vdash \varphi(\alpha,s\nu,\gamma),\\ \mathcal{S}_2 &\equiv \Lambda'(\alpha), \{\varphi(\alpha,\alpha,u_i(\alpha))\}_{i=1}^m \vdash \psi(\alpha) \end{aligned}$$

of $\forall\Gamma \vdash \psi(\alpha)$. Then $G(\pi)$ is the parametric grammar with the productions

$$\begin{aligned} \{\,\tau \to\ [\forall\bar{x}\,\vartheta(\bar{x})](s_1,...,s_k) \mid \forall\bar{x}\,\vartheta(\bar{x}) \in\ \forall\Gamma, \vartheta(s_1,...,s_k) \in\ \Gamma'\cup\Pi'\cup\Lambda'\}\\ \cup\{\,\gamma\to t_i(\alpha,\nu,\gamma) \mid i\in\{1,...,n\}\}\\ \cup\{\,\gamma^*\to\ u_i(\alpha) \mid i\in\ \{1,...,m\}\}\,. \end{aligned}$$

If $\pi$ is the existential sip

$$\begin{aligned} \mathcal{S}_0 &\equiv \Gamma'(\alpha) \vdash \{\varphi(\alpha,0,u_i(\alpha))\}_{i=1}^m\,,\\ \mathcal{S}_1 &\equiv \Pi'(\alpha,\nu,\gamma), \nu<\alpha, \varphi(\alpha,\nu,\gamma) \vdash \{\varphi(\alpha,s\nu,t_i(\alpha,\nu,\gamma))\}_{i=1}^n\,,\\ \mathcal{S}_2 &\equiv \Lambda'(\alpha,\gamma), \varphi(\alpha,\alpha,\gamma) \vdash \psi(\alpha), \end{aligned}$$

of $\forall\Gamma \vdash \psi(\alpha)$, then $G(\pi)$ is defined in exactly the same way.

Parametric grammars are not interesting as grammars in themselves. Rather, they can be instantiated with natural numbers, resulting in totally rigid acyclic tree grammars. Thus, a parametric grammar can be seen as a scheme generalizing a sequence of tree grammars.

In fact, we can instantiate a parametric grammar in two different ways ("universal" and "existential") to obtain proper tree grammars.

**Definition 2.41** (Universal instance grammars of a parametric grammar)**.** Let $G=\langle\tau,N,\mathcal{L},P\rangle$ be a parametric grammar and $n\in\mathbb{N}$. Then the $n$-th universal instance grammar of $G$ is the totally rigid acyclic tree grammar $G_n^\forall=\langle\tau,N_n,\mathcal{L},P_n\rangle$, where $N_n=\{\tau,\gamma_0,...,\gamma_n\}$ and

$$\begin{aligned} P_n =& \{\,\tau\to t(\bar{n},\gamma_0) \mid \tau\to t(\alpha,\gamma)\in P\}\\ &\cup\{\,\tau\to t(\bar{n},\bar{i},\gamma_{i+1}) \mid \tau\to t(\alpha,\nu,\gamma)\in P, 0\le i<n\}\\ &\cup\{\,\gamma_i\to t(\bar{n},\bar{i},\gamma_{i+1}) \mid \gamma\to t(\alpha,\nu,\gamma)\in P, 0\le i<n\}\\ &\cup\{\,\gamma_n\to t(\bar{n}) \mid \gamma^*\to t(\alpha)\in P\} \end{aligned}$$

The nonterminals of $G_n^\forall$ are clearly ordered by $\tau < \gamma_0 < \gamma_1 < ... < \gamma_n$.

The language of $G_n^\forall$ can be described as follows: Let

$$\begin{aligned} L_n(G_n^\forall) &= \{\, t(\bar{n}) \mid \gamma^* \to t(\alpha) \in P \}\,, \\ L_k(G_n^\forall) &= \left\{\, t(\bar{n}, \bar{k}, t') \;\middle|\; \gamma \to t(\alpha, \nu, \gamma) \in P, t' \in L_{k+1}(G_n^\forall) \right\}, \qquad 0 \le k < n. \end{aligned}$$

Then

$$\begin{aligned} L(G_n^\forall) = &\left\{\, t(\bar{n}, \bar{k}, t') \;\middle|\; \tau \to t(\alpha, \nu, \gamma) \in P, 0 < k \le n, t' \in L_k(G_n^\forall) \right\} \\ &\cup \left\{\, t(\bar{n}, t') \;\middle|\; \tau \to t(\alpha, \beta) \in P, t' \in L_0(G_n^\forall) \right\}. \end{aligned}$$

The definition of existential instance grammars is dual to that of universal instance grammars:

**Definition 2.42** (Existential instance grammars of a parametric grammar)**.** Let $G = \langle \tau, N, \mathcal{L}, P \rangle$ be a parametric grammar and $n \in \mathbb{N}$. Then the *n-th existential instance grammar of* $G$ is the totally rigid acyclic tree grammar $G_n^\exists = \langle \tau, N_n, \mathcal{L}, P_n \rangle$, where $N_n = \{\tau, \gamma_0, ..., \gamma_n\}$ and

$$\begin{aligned} P_n = &\{\, \tau \to t(\bar{n}, \gamma_n) \mid \tau \to t(\alpha, \gamma) \in P \} \\ &\cup \left\{\, \tau \to t(\bar{n}, \bar{i}, \gamma_i) \;\middle|\; \tau \to t(\alpha, \nu, \gamma) \in P, 0 \le i < n \right\} \\ &\cup \left\{\, \gamma_{i+1} \to t(\bar{n}, \bar{i}, \gamma_i) \;\middle|\; \gamma \to t(\alpha, \nu, \gamma) \in P, 0 \le i < n \right\} \\ &\cup \{\, \gamma_0 \to t(\bar{n}) \mid \gamma^* \to t(\alpha) \in P \}\,. \end{aligned}$$

The order on the nonterminals of $G_n^\exists$ is easily seen to be $\tau < \gamma_n < \gamma_{n-1} < ... < \gamma_0$.

We can also give a description of the language of $G_n^\exists$: Let

$$\begin{aligned} L_0(G_n^\exists) &= \{\, t(\bar{n}) \mid \gamma^* \to t(\alpha) \in P \}\,, \\ L_{k+1}(G_n^\exists) &= \left\{\, t(\bar{n}, \bar{k}, t') \;\middle|\; \gamma \to t(\alpha, \nu, \gamma) \in P, t' \in L_k(G_n^\exists) \right\}, \qquad 0 \le k < n. \end{aligned}$$

Then

$$\begin{aligned} L(G_n^\exists) = &\left\{\, t(\bar{n}, \bar{k}, t') \;\middle|\; 0 \le k < n, \tau \to t(\alpha, \nu, \gamma) \in P, t' \in L_k(G_n^\exists) \right\} \\ &\cup \left\{\, t(\bar{n}, t') \;\middle|\; \tau \to t(\alpha, \beta) \in P, t' \in L_n(G_n^\exists) \right\} \end{aligned}$$

Imagine that we have a (universal or existential) sip of a sequent $\forall\Gamma \vdash \psi(\alpha)$. If we compute the parametric grammar $G(\pi)$, we cannot completely reconstruct $\pi$ from it, since the induction formula $\varphi$ has been lost in the translation. The only option is to introduce a formula variable for the unknown induction formula. This leads us to the following definition.

**Definition 2.43** (Simple induction proof schema)**.** Let $\mathcal{L}$ be an arithmetical language. A *universal simple induction proof schema* is a formula equation

$$\exists X.\ \forall \alpha, \nu, \gamma.\ \mathcal{S}_0 \wedge \mathcal{S}_1 \wedge \mathcal{S}_2,$$

where

$$\begin{aligned} \mathcal{S}_0 &\equiv \Gamma'(\alpha,\gamma) \vdash X(\alpha, 0, \gamma), \\ \mathcal{S}_1 &\equiv \Pi'(\alpha,\nu,\gamma), \nu < \alpha, \{X(\alpha,\nu,t_i(\alpha,\nu,\gamma))\}_{i=1}^{n} \vdash X(\alpha, s\nu, \gamma), \\ \mathcal{S}_2 &\equiv \Lambda'(\alpha), \{X(\alpha,\alpha,u_i(\alpha))\}_{i=1}^{m} \vdash \psi(\alpha), \end{aligned}$$

and

- All formulas, terms, and sets are fully indicated,
- $X$ is a formula variable of sort $\mathbf{Nat} \times \mathbf{Nat} \times \tau$ for some sort $\tau$,
- $\psi$ is a quantifier-free formula of sort $\mathbf{Nat}$,
- $\Gamma', \Pi', \Lambda'$ are sets of quantifier-free formulas,

such that $\vdash_{\mathbf{LK}} \mathcal{S}_i$ for $i \in \{0, 1, 2\}$.

Dually, an *existential simple induction proof schema* is a formula equation

$$\exists X.\ \forall \alpha, \nu, \gamma.\ \mathcal{S}_0 \wedge \mathcal{S}_1 \wedge \mathcal{S}_2,$$

where

$$\begin{aligned} \mathcal{S}_0 &\equiv \Gamma'(\alpha) \vdash \{X(\alpha, 0, u_i(\alpha))\}_{i=1}^{m} \\ \mathcal{S}_1 &\equiv \Pi'(\alpha,\nu,\gamma), \nu < \alpha, X(\alpha,\nu,\gamma) \vdash \{X(\alpha, s\nu, t_i(\alpha,\nu,\gamma))\}_{i=1}^{n} \\ \mathcal{S}_2 &\equiv \Lambda'(\alpha,\gamma), X(\alpha,\alpha,\gamma) \vdash \psi(\alpha) \end{aligned}$$

with the same conditions as in the universal case.

A simple induction proof schema is completely determined by the instance sets $\Gamma', \Pi', \Lambda'$, the formula $\psi$, and the sets of step terms and base terms.

Simply put, a simple induction proof schema is like a simple induction proof with an undetermined induction formula of which we only know that it is $\Pi_1$ or $\Sigma_1$ and the terms it is instantiated with.

Simple induction schemas directly induce two solution problems.

**Definition 2.44** (Induction problems)**.** Let $\mathcal{L}$ be an arithmetical language, $\mathcal{T}$ an $\mathcal{L}$-theory, $\mathcal{C}$ the class of quantifier-free $\mathcal{L}$-formulas of the form $\varphi(\alpha, \nu, \gamma)$, where $\alpha, \nu$: **Nat**, and $\Phi_\forall$ and $\Phi_\exists$ the classes of universal and existential simple induction proof schemas over $\mathcal{L}$, respectively. Then the *universal induction problem* over $\mathcal{T}$ is the solution problem $\langle \mathcal{T}, \Phi_\forall, \mathcal{C} \rangle$ and the *existential induction problem* over $\mathcal{T}$ is the solution problem $\langle \mathcal{T}, \Phi_\exists, \mathcal{C} \rangle$.

As we have hinted earlier, simple induction proof schemas can be extracted from parametric grammars.

**Definition 2.45** (Extracting universal simple induction proof schemas from parametric grammars)**.** Let $G$ be a parametric grammar for the sequent $\forall\Gamma, \forall\Pi, \forall\Lambda \vdash \psi(\alpha)$. Then $G$ induces a universal simple induction proof schema with the sequents

$$\begin{aligned} \Gamma' &\vdash X(\alpha, 0, \gamma) \\ \Pi', \bigwedge_{t \in \mathcal{S}} X(\alpha, \nu, t(\alpha, \nu, \gamma)) &\vdash X(\alpha, s\nu, \gamma) \\ \Lambda', \bigwedge_{t \in \mathcal{C}} X(\alpha, \alpha, t(\alpha)) &\vdash \psi(\alpha), \end{aligned}$$

where

$$\begin{aligned} \Gamma' &= \{\, \rho(\bar{t}(\alpha, \gamma)) \mid \tau \to [\forall \bar{x}\, \rho(\bar{x})](\bar{t}(\alpha, \gamma)) \in G \} \\ &\quad \cup \{\, \rho(\bar{t}(\alpha)) \mid \tau \to [\forall \bar{x}\, \rho(\bar{x})](\bar{t}(\alpha)) \in G \} \\ \Pi' &= \{\, \rho(\bar{t}(\alpha, \nu, \gamma)) \mid \tau \to [\forall \bar{x}\, \rho(\bar{x})](\bar{t}(\alpha, \nu, \gamma)) \} \\ &\quad \cup \{\, \rho(\bar{t}(\alpha)) \mid \tau \to [\forall \bar{x}\, \rho(\bar{x})](\bar{t}(\alpha)) \in G \} \\ \Lambda' &= \{\, \rho(\bar{t}(\alpha)) \mid \tau \to [\forall \bar{x}\, \rho(\bar{x})](\bar{t}(\alpha)) \in G \} \\ \mathcal{S} &= \{\, t(\alpha, \nu, \gamma) \mid \gamma \to t(\alpha, \nu, \gamma) \in G \} \\ \mathcal{C} &= \{\, t(\alpha) \mid \gamma_{end} \to t(\alpha) \in G \} \end{aligned}$$

Similarly, $G$ also induces an existential simple induction proof schema:

$$\begin{aligned} \Gamma' &\vdash \bigvee_{t \in \mathcal{C}} X(\alpha, 0, t(\alpha)) \\ \Pi', X(\alpha, \nu, \gamma) &\vdash \bigvee_{t \in \mathcal{S}} X(\alpha, s\nu, t(\alpha, \nu, \gamma)) \\ \Lambda', X(\alpha, \alpha, \gamma) &\vdash \psi(\alpha), \end{aligned}$$

with $\Gamma', \Pi', \Lambda', \mathcal{S}, \mathcal{C}$ defined in exactly the same way as in the universal case.

We have introduced formula equations and solution problems and shown how the problem of inductive theorem proving can be expressed in these terms by using parametric grammars as a translation mechanism.

# CHAPTER 3

# NONDETERMINISTIC PROGRAMS AND DYNAMIC LOGIC

In this chapter, we will investigate formula equations in the context of the deductive verification of nondeterministic imperative programs. Concretely, we define a simple programming language called **IMP** that supports nondeterministic assignments.

The motivation of this investigation comes from the program of inductive theorem proving via tree grammars as described in Section 2.3. The original idea was this: let $\pi$ be an existential simple induction proof schema and $G_\pi$ its associated parametric grammar. Recall that $\pi$ is a formula equation whose solutions are induction formulas. The terms used for instantiating a solution $\iota$ are computed by $G_\pi$, but the propositional structure of $\iota$ is unknown. It is clear that a parametric grammar can be viewed as a nondeterministic program $p$ that takes $n \in \mathbb{N}$ as an input and outputs exactly the elements of the $n$th instance language of $G$. We hoped that the solutions of $\pi$ would then coincide with invariants of $p$, but this is not the case—at least not for the conventional meaning of "invariant". While e.g. Hoare calculus can be adapted for nondeterministic programs, the result is

a deductive system for invariants of *all* executions of nondeterministic programs. However, what we need are "invariants" that are preserved by *some* execution of a program. This dichotomy between "universal" and "existential" invariants suggests a kind of modal logic in which each program induces two modalities,resulting in the modal logic known as *dynamic logic*. Thus, the first step towards realizing the analogy between induction formulas and loop invariants is an investigation based on dynamic logic.

## 3.1 Dynamic Logic

Our treatment of dynamic logic is based on [HTK00]. The modalities of dynamic logic are induced by programs. Therefore, we first describe a general framework for programs that we will build our definition of dynamic logic on.

**Definition 3.1** (Regular program)**.** Let $\mathcal{L}$ be a first-order language. The *regular programs* over $\mathcal{L}$ are inductively defined as follows:

1. If $x$ is a variable and $t$ is an $\mathcal{L}$-term, then $x := t$ is a regular program.
2. If $\varphi$ is a quantifier-free $\mathcal{L}$-formula, then $\varphi?$ is a regular program.
3. If $p, q$ are regular programs, then $p; q$ is a regular program.
4. If $p, q$ are regular programs, then $p \cup q$ is a regular program.
5. If $p$ is a regular program, then $p^*$ is a regular program.

The semantics of regular programs will be described in Definition 3.2. If the language $\mathcal{L}$ under discussion is unambiguous, we will just refer to regular programs.

The condition of quantifier-freeness in 2 makes this a definition of regular programs with *poor tests*. If we allowed all first-order or even dynamic logic formulas, we would obtain regular programs with *rich tests*. Allowing dynamic logic formulas would, of course, require us to define regular programs and dynamic logic by mutual induction. Fortunately, poor tests easily suffice for our purposes.

Intuitively, these constructs have the following meanings:

1. $x := t$ is the *assignment* of the term $t$ to the variable $x$.
2. $\varphi?$ is the *test* whether $\varphi$ is true in the current program state.
3. $p; q$ is the *sequential composition* of $p$ and $q$. It first executes $p$ and then $q$.

4. $p \cup q$ is the *nondeterministic choice* between $p$ and $q$. It executes either $p$ or $q$.
5. $p^*$ is the *nondeterministic iteration* of $p$. It executes $p$ zero or more times.

Let us make these intuitions precise.

**Definition 3.2** (Input/output relation)**.** Let $\mathcal{L}$ be a logical language and $\mathcal{M}$ an $\mathcal{L}$-structure. Every regular program $p$ over $\mathcal{L}$ induces a binary relation $p^{\mathcal{M}}$ on the valuations of $\mathcal{M}$, the *input/output relation*:

1. $(x := t)^{\mathcal{M}} = \{ (v, v[x \backslash v(t)]) \mid v \in \mathrm{Val}(\mathcal{M}) \}$.
2. $(\varphi?)^{\mathcal{M}} = \{ (v, v) \mid v \in \mathrm{Val}(\mathcal{M}), (\mathcal{M}, v) \vDash \varphi \}$.
3. $(p; q)^{\mathcal{M}} = p^{\mathcal{M}} \circ q^{\mathcal{M}}$.
4. $(p \cup q)^{\mathcal{M}} = p^{\mathcal{M}} \cup q^{\mathcal{M}}$.
5. $(p^*)^{\mathcal{M}} = \bigcup_{n \in \mathbb{N}} (p^{\mathcal{M}})^n$

We write $p^{\mathcal{M}}(v)$ for the set $\{ u \in \mathrm{Val}(\mathcal{M}) \mid (v, u) \in p^{\mathcal{M}} \}$.

**Observe that a regular program's meaning is completely determined by which input states it associates to which output states, not by its "implementation details".** For instance, consider the programs $\top? \cup \bot?$ and $\top?$. The first one nondeterministically chooses between succeeding and failing, the second always succeeds. Nevertheless, they are semantically equivalent, since they both induce the identity relation (on every structure).

Moreover, since the relational product $\circ$ is associative, the programs $(p; q); r$ and $p; (q; r)$ are semantically equivalent, and so we can consider the composition operator ; associative as well. Likewise, we can regard the choice operator $\cup$ as associative and commutative.

We are now ready to introduce dynamic logic.

**Definition 3.3** (First-order dynamic logic)**.** Let $\mathcal{L}$ be a first-order language and $\mathcal{P}$ some class of regular programs over $\mathcal{L}$. The formulas of *dynamic logic over $\mathcal{L}$ and $\mathcal{P}$* ($(\mathcal{L}, \mathcal{P})$-DL) are constructed the same way as first-order $\mathcal{L}$-formulas, with the addition of *modalities*:

If $\varphi$ is a $(\mathcal{L}, \mathcal{P})$-DL formula and $p \in \mathcal{P}$, then $[p]\varphi$ and $\langle p \rangle \varphi$ are also $(\mathcal{L}, \mathcal{P})$-DL formulas.

As usual, we will just refer to DL if there is no danger of confusion about what $\mathcal{L}$ and $\mathcal{P}$ are.

The semantics of DL formulas is an extension of the semantics of regular programs.

**Definition 3.4** (Semantics of DL)**.** Let $\mathcal{L}$ be a first-order language, $\mathcal{P}$ a class of regular programs over $\mathcal{L}$, and $\mathcal{M}$ an $\mathcal{L}$-structure. Recall that for $p \in \mathcal{P}$, $p^{\mathcal{M}}$ is the input-output relation of $p$ on $\mathcal{M}$.

We extend the definition of the relation $(\mathcal{M}, v) \vDash \varphi$ for $v \in \mathrm{Val}(\mathcal{M})$ from first-order logic to the connectives $[\cdot]$ and $\langle\cdot\rangle$: If $\varphi$ is a DL formula and $p \in \mathcal{P}$, then $(\mathcal{M}, v) \vDash_{DL} [p]\varphi$ iff for all $u \in p^{\mathcal{M}}(v)$, $(\mathcal{M}, u) \vDash_{DL} \varphi$. Likewise, $(\mathcal{M}, v) \vDash_{DL} \langle p\rangle\varphi$ iff there is a $u \in p^{\mathcal{M}}(v)$ such that $(\mathcal{M}, u) \vDash_{DL} \varphi$.

As in first-order logic, we say that a DL formula $\varphi$ holds in $\mathcal{M}$, written as $\mathcal{M} \vDash_{DL} \varphi$, if $(\mathcal{M}, v) \vDash_{DL} \varphi$ for all $v \in \mathrm{Val}(\mathcal{M})$. Likewise, the relations $\vDash_{DL} \varphi$ and $\mathcal{T} \vDash_{DL} \varphi$ are defined analogously to first-order logic.

To reduce notational clutter, we will sometimes write $v \vDash_{DL} \varphi$ instead of $(\mathcal{M}, v) \vDash_{DL} \varphi$ in cases where the structure can safely be left implicit.

The semantics of the modalities can be explained as follows. If $p$ is a regular program and $\mathcal{M}$ an $\mathcal{L}$-structure, each valuation $v \in \mathrm{Val}(\mathcal{M})$ can be interpreted as a state of all the variables occurring in $p$ (and also of every other variable, but their values are irrelevant). Then $p^{\mathcal{M}}(v)$ is the set of all states that can be the result of running $p$ to completion, starting from $v$. It follows that $[p]\varphi$ holds in the state $v$ if $\varphi$ holds in every possible end-state of $p$ and $\langle p\rangle\varphi$ holds if $\varphi$ holds in some end-state of $p$. Clearly, if $p^{\mathcal{M}}(v) = \emptyset$, then $(\mathcal{M}, v) \vDash [p]\varphi \leftrightarrow \top$ and $(\mathcal{M}, v) \vDash \langle p\rangle\varphi \leftrightarrow \bot$.

Let us give some simple valid formulas of DL.

**Proposition 3.5.** *The following formulas of DL are valid:*

1. $[p]\varphi \leftrightarrow \neg\langle p\rangle\neg\varphi$.
2. $[p; q]\varphi \leftrightarrow [p][q]\varphi$.
3. $\langle p; q\rangle\varphi \leftrightarrow \langle p\rangle\langle q\rangle\varphi$.
4. $[\varphi?]\psi \leftrightarrow \varphi \to \psi$.
5. $\langle\varphi?\rangle\psi \leftrightarrow \varphi \wedge \psi$.
6. $[p \cup q]\varphi \leftrightarrow [p]\varphi \wedge [q]\varphi$.

7. $\langle p \cup q\rangle\varphi \leftrightarrow \langle p\rangle\varphi \vee \langle q\rangle\varphi$.
8. $[x := t]\varphi \leftrightarrow \varphi[x\backslash t] \leftrightarrow \langle x := t\rangle\varphi$.
9. $[p^*]\varphi \leftrightarrow \varphi \wedge [p^*](\varphi \to [p]\varphi)$ *(the induction principle of dynamic logic)*

*Proof.* We only prove the induction principle. Let $\mathcal{M}$ be a structure and $v \in \mathrm{Val}(\mathcal{M})$, and assume $v \vDash_{DL} [p^*]\varphi$. Since $v \in (p^*)^{\mathcal{M}}(v)$, this implies $v \vDash_{DL} \varphi$. Now assume $u \in (p^*)^{\mathcal{M}}(v)$. We need to show $u \vDash_{DL} \varphi \to [p]\varphi$. By assumption, $v \vDash_{DL} [p^*]\varphi$, so $u \vDash_{DL} \varphi$. To see that $u \vDash_{DL} [p]\varphi$, observe that if $w \in p^{\mathcal{M}}(u)$, then $w \in (p^*)^{\mathcal{M}}(v)$ and hence $w \vDash_{DL} \varphi$, again by assumption. This concludes the proof of the forward direction.

For the other direction, assume $v \vDash_{DL} \varphi \wedge [p^*](\varphi \to [p]\varphi)$. We need to show $u \vDash \varphi$ for all $u \in (p^*)^{\mathcal{M}}(v)$. The latter is equivalent to the existence of a sequence $v = w_0\ p^{\mathcal{M}}\ w_1\ p^{\mathcal{M}}\ \ldots\ p^{\mathcal{M}}\ w_{n-1}\ p^{\mathcal{M}}\ w_n = u$. We proceed by induction on $n$. If $n = 0$, then $v = u$ and we know $v \vDash_{DL} \varphi$ by assumption. For the step case, suppose we already know $w_n \vDash_{DL} \varphi$. Since $w_n \in (p^*)^{\mathcal{M}}(v)$, we also have $w_n \vDash_{DL} \varphi \to [p]\varphi$. From these two facts and $w_n\ p^{\mathcal{M}}\ u$, we conclude $u \vDash_{DL} \varphi$. □

We also easily obtain a version of the generalization rule familiar from other modal logics.

**Lemma 3.6.** *Let $\mathcal{M}$ be a structure, $\varphi$ a DL formula, and $p$ a regular program. If $\mathcal{M} \vDash_{DL} \varphi$, then $\mathcal{M} \vDash [p]\varphi$. In other words, the inference rule*

$$\frac{\varphi}{[p]\varphi}$$

*is sound.*

Formulas of the forms $\varphi \to [p]\psi$ and $\varphi \to \langle p\rangle\psi$ are of particular interest to us. By the explanation above, the former expresses "if $\varphi$ holds in the current state, then running $p$ will always result in a state in which $\psi$ holds". Likewise, the latter can be interpreted as "if $\varphi$ holds in the current state, then there is some execution of $p$ that results in a state satisfying $\psi$". This means that $\varphi \to [p]\psi$ is equivalent to the Hoare triple $\{\varphi\}p\{\psi\}$, while $\varphi \to \langle p\rangle\psi$ is not expressible in Hoare logic.

Regular programs permit nondeterministic choice between any two programs. We will restrict ourselves to a much less general class of programs that only contain nondeterministic assignments.

**Definition 3.7** (**IMP**)**.** Let $\mathcal{L}$ be a first-order language. We define $\mathbf{IMP}_{\mathcal{L}}$ *programs* as abbreviations of certain regular programs:

$$\begin{aligned}
\mathbf{skip} &\equiv \top? \\
x := t_1 \mid \cdots \mid t_n &\equiv x := t_1 \cup \ldots \cup x := t_n \\
\mathbf{if}\ \varphi\ \mathbf{then}\ p\ \mathbf{else}\ q &\equiv (\varphi?; p) \cup (\neg\varphi?; q) \\
\mathbf{while}\ \varphi\ \mathbf{do}\ p &\equiv (\varphi?; p)^*; \neg\varphi?
\end{aligned}$$

The sequential composition operator ; of **IMP** is interpreted as that of regular programs.

Let us briefly discuss the semantics of **IMP**. The trivial program **skip** leaves every state unchanged (i.e. $\mathbf{skip}^{\mathcal{M}} = \mathrm{id}_M$); $x := t_1 \mid \cdots \mid t_n$ assigns one of the terms $t_1, \ldots, t_n$ to the variable $x$; **if-then-else**, **while**, and composition have their classical meanings. It is important to note that even though the definitions of **if-then-else** and **while** include nondeterministic operators, they are themselves deterministic, due to the way in which those operators are used. The tests $\varphi?$ and $\neg\varphi?$ in **if-then-else** ensure that only one of the two subprograms can actually execute in any given state, and in **while**, these same tests ensure that the iteration cannot terminate while $\varphi$ still holds and must terminate when $\varphi$ ceases to hold. Thus, assignments are the only source of nondeterminism in **IMP**.

Observe also that **skip** is the neutral element with respect to sequential composition, as far as semantics is concerned. This means that, semantically speaking, we can assume that every program is either **skip** or of the form $p; q$ with $q \neq \mathbf{skip}$.

**In the sequel, "DL" refers to $(\mathcal{L}, \mathbf{IMP}_{\mathcal{L}})$-DL for some language $\mathcal{L}$.**

As for regular programs, we give a few simple valid formulas for DL with **IMP** programs.

**Proposition 3.8.** *The following DL formulas are valid:*

1. $[\mathbf{skip}]\varphi \leftrightarrow \varphi \leftrightarrow \langle\mathbf{skip}\rangle\varphi$.
2. $[x := t_1 | \ldots | t_n]\varphi \leftrightarrow \bigwedge_{i=1}^{n} \varphi[x \backslash t_i]$.
3. $\langle x := t_1 | \ldots | t_n\rangle\varphi \leftrightarrow \bigvee_{i=1}^{n} \varphi[x \backslash t_i]$.
4. $[\mathbf{if}\ \gamma\ \mathbf{then}\ p\ \mathbf{else}\ q]\varphi \leftrightarrow (\gamma \wedge [p]\varphi) \vee (\neg\gamma \wedge [q]\varphi)$.

5. $\langle \textbf{if } \gamma \textbf{ then } p \textbf{ else } q\rangle\varphi \leftrightarrow (\gamma \wedge \langle p\rangle\varphi) \vee (\neg\gamma \wedge \langle q\rangle\varphi)$.

*Proof.* All items follow straightforwardly from Proposition 3.5: 1 from 4 and 5; 2 and 3 from 6, 7, 8; 4 and 5 from 2, 4, 6 and 3, 5, 7, respectively. □

## 3.2 Verification conditions

In this section we will only deal with arithmetical languages. In fact, we will restrict our attention to structures in which the sort **Nat** has a certain inductive property. This allows us to formulate strong assumptions about the termination of loops.

In order to be able to prove termination of loops and some invariants, we will require some basic facts about the $\leq$ relation and the operations $p$ and $-$ (see Definition 1.15). Therefore, we define a small arithmetical theory that will be sufficient for these purposes.

**Definition 3.9** ($\mathbf{Q}_{\text{min}}$)**.** $\mathbf{Q}_{\text{min}}$ is the theory generated by the arithmetical axioms (O1) – (O5), (OS1), (OS2), (OSP), (OMi), and (OSMi) (cf. Definition 1.16 in Chapter 1).

**Definition 3.10** (Inductive structure, convergence rule)**.** Let $\mathcal{L}$ be an arithmetical language and $\mathcal{M}$ an $\mathcal{L}$-structure. We call $\mathcal{M}$ inductive if $\mathcal{M} \models \mathbf{Q}_{\text{min}}$ and, for all first-order formulas $\tau$ and regular programs $p$ not containing $\nu$, $\mathcal{M}$ satisfies the *convergence property*:

$$\mathcal{M} \models_{DL} \tau(s\nu) \rightarrow \langle p\rangle\tau(\nu)$$

implies

$$\mathcal{M} \models_{DL} \tau(\alpha) \rightarrow \langle p^*\rangle\tau(0).$$

In other words, the *convergence rule*

$$\frac{\tau(s\nu) \rightarrow \langle p\rangle\tau(\nu)}{\tau(\alpha) \rightarrow \langle p^*\rangle\tau(0)}$$

(with the aforementioned conditions on $\tau$ and $p$) is sound for inductive structures.

In the sequel, we write $\Gamma \models_{DL}^{ind} \psi$ if $\mathcal{M} \models_{DL} \Gamma$ implies $\mathcal{M} \models_{DL} \psi$ for all inductive structures $\mathcal{M}$.

Our aim is to define a set of sufficient first-order conditions, called verification conditions, for a formula of the form $\varphi \to [p]\psi$ or $\varphi \to \langle p\rangle\psi$ to be valid in inductive structures. Unfortunately, these conditions cannot simply be extracted from the triple of $\varphi, p, \psi$. Rather, it is necessary to *annotate* certain instructions of $p$ with formulas.

**Definition 3.11** (Annotated **IMP** program)**.** Let $\mathcal{L}$ be an arithmetical language. An *annotated* $\mathbf{IMP}_{\mathcal{L}}$ *program* is an $\mathbf{IMP}_{\mathcal{L}}$ program $p$ in which each conditional and loop instruction carries an *annotation*:

- $\{\alpha\}$ **if** $\varphi$ **then** $p$ **else** $q$, where $\alpha$ is an $\mathcal{L}$-formula;
- **while** $\varphi$ $\{\iota; t\}$ **do** $p$, where $\iota$ is an $\mathcal{L}$-formula (the *invariant* of the loop) and $t\colon$ **Nat** is an $\mathcal{L}$-term (the *termination witness* of the loop).

We extend the semantics of **IMP** programs to annotated **IMP** programs by simply erasing all annotations. If erasing all annotations from an annotated program $p'$ yields $p$, we call $p'$ an annotation of $p$.

We immediately obtain the following lemma.

**Lemma 3.12.** *Let $p$ be an* **IMP** *program, $p'$ an annotation of $p$, and $\varphi$ a DL formula. Then $\models_{DL} [p]\varphi \leftrightarrow [p']\varphi$ and $\models_{DL} \langle p\rangle\varphi \leftrightarrow \langle p'\rangle\varphi$.*

We can now define the aforementioned verification conditions.

**Definition 3.13** (Verification conditions)**.** Let $\mathcal{L}$ be an arithmetical language. Let $\varphi, \psi$ be formulas and $p$ an annotated **IMP** program. We define the sets $\mathrm{VC}^{\Box}(\{\varphi\}\, p\, \{\psi\})$ of *box verification conditions* and $\mathrm{VC}^{\Diamond}(\{\varphi\}\, p\, \{\psi\})$ of *diamond*

*verification conditions* by induction.

$$
\begin{aligned}
\mathrm{VC}^{\Box}\left(\{\varphi\}\,\mathbf{skip}\,\{\psi\}\right) &= \mathrm{VC}^{\Diamond}\left(\{\varphi\}\,\mathbf{skip}\,\{\psi\}\right) \\
&= \{\varphi \to \psi\}, \\
\mathrm{VC}^{\Box}\left(\{\varphi\}\,p; x := t_1|...|t_n\,\{\psi\}\right) &= \mathrm{VC}^{\Box}\left(\{\varphi\}\,p\left\{\bigwedge_{i=1}^{n} \psi[x\backslash t_i]\right\}\right), \\
\mathrm{VC}^{\Diamond}\left(\{\varphi\}\,x := t_1|...|t_n\,\{\psi\}\right) &= \mathrm{VC}^{\Diamond}\left(\{\varphi\}\,p\left\{\bigvee_{i=1}^{n} \psi[x\backslash t_i]\right\}\right), \\
\mathrm{VC}^{\Box}\left(\{\varphi\}\,p; \{\alpha\}\,\mathbf{if}\ \gamma\ \mathbf{then}\ q\ \mathbf{else}\ r\,\{\psi\}\right) &= \mathrm{VC}^{\Box}\left(\{\varphi\}\,p\,\{\alpha\}\right) \\
&\quad \cup \mathrm{VC}^{\Box}\left(\{\alpha \wedge \gamma\}\,q\,\{\psi\}\right) \\
&\quad \cup \mathrm{VC}^{\Box}\left(\{\alpha \wedge \neg\gamma\}\,r\,\{\psi\}\right), \\
\mathrm{VC}^{\Diamond}\left(\{\varphi\}\,p; \{\alpha\}\,\mathbf{if}\ \gamma\ \mathbf{then}\ q\ \mathbf{else}\ r\,\{\psi\}\right) &= \mathrm{VC}^{\Diamond}\left(\{\varphi\}\,p\,\{\alpha\}\right) \\
&\quad \cup \mathrm{VC}^{\Diamond}\left(\{\alpha \wedge \gamma\}\,q\,\{\psi\}\right) \\
&\quad \cup \mathrm{VC}^{\Diamond}\left(\{\alpha \wedge \neg\gamma\}\,r\,\{\psi\}\right), \\
\mathrm{VC}^{\Box}\left(\{\varphi\}\,p; \mathbf{while}\ \gamma\ \{\iota; t\}\ \mathbf{do}\ q\,\{\psi\}\right) &= \mathrm{VC}^{\Box}\left(\{\varphi\}\,p\,\{\iota\}\right) \\
&\quad \cup \mathrm{VC}^{\Box}\left(\{\iota \wedge \gamma\}\,q\,\{\iota\}\right) \\
&\quad \cup \{\iota \wedge \neg\gamma \to \psi\}, \\
\mathrm{VC}^{\Diamond}\left(\{\varphi\}\,p; \mathbf{while}\ \gamma\ \{\iota; t\}\ \mathbf{do}\ q\,\{\psi\}\right) &= \mathrm{VC}^{\Diamond}\left(\{\varphi\}\,p\,\{\iota\}\right) \\
&\quad \cup \mathrm{VC}^{\Diamond}\left(\{\iota \wedge \gamma \wedge t = z\}\,q\,\{\iota \wedge t < z\}\right) \\
&\quad \cup \{\iota \wedge \neg\gamma \to \psi\}.
\end{aligned}
$$

In the conditions for the while loop, $z$ is a fresh variable that does not occur in $\gamma$, $t$, $q$, or $\iota$.

Recall the defining property we expect of these verification conditions: $\mathrm{VC}^{\Box}\left(\{\varphi\}\,p\,\{\psi\}\right)$ should imply $\varphi \to [p]\psi$ and $\mathrm{VC}^{\Diamond}\left(\{\varphi\}\,p\,\{\psi\}\right)$ should imply $\varphi \to \langle p\rangle\psi$. We will show that this is actually the case in Theorem 3.15.

We can now see the reason for the annotations on conditionals and loops. Without them, we could not straightforwardly define the verification conditions for $p; \mathbf{if}\ \gamma\ \mathbf{then}\ q\ \mathbf{else}\ r$ and $p; \mathbf{while}\ \gamma\ \{\iota; t\}\ \mathbf{do}\ q$ based on those of $p$ and the subsequent conditional or loop, respectively.

Note that most of the definition is symmetrical in $\Box$ and $\Diamond$. The one exception is

the fact that the diamond conditions for the while loop involve the loop's termination witness. This is the case because $\langle \textbf{while } \gamma \ \{\iota; t\} \textbf{ do } q\rangle\psi$ asserts that there is at least one execution of $\textbf{while } \gamma \textbf{ do } q$ from the current state that leads to a state satisfying $\psi$. In particular, it must be possible to execute $\textbf{while } \gamma \textbf{ do } q$ in the current state $v$ and get to any state at all. We enforce this by ensuring that the termination witness always decreases for at least one of the possible executions of the loop body. Because its sort is **Nat**, this implies that $(\textbf{while } \gamma \textbf{ do } q)^{\mathcal{M}}(v) \neq \emptyset$ for an inductive structure $\mathcal{M}$. Moreover, since we need the invariant $\iota$ to hold after the loop, there must be some execution of the body preserving $\iota$, and in fact these two conditions must be simultaneously fulfilled by one execution. This stipulation is captured by the set $\mathrm{VC}^{\Diamond}(\{\iota \wedge \gamma \wedge t = z\}\, q\, \{\iota \wedge t < z\})$.

The termination condition on while loops may seem extremely restrictive: there needs to be a type of the sort **Nat** that bounds the possible number of iterations. This means that while we can express very general programs, our capacity for verification, at least in the diamond case, is effectively restricted to for loops. However, for our purposes this restriction is immaterial—we will see that the only loops we require iterate from 0 to $\alpha$ or the other way around.

The following lemma will be useful in the proof of Theorem 3.15.

**Lemma 3.14.** *Let $\varphi, \psi, \rho$ be DL formulas and $p, q$ regular programs. Then*

$$\varphi \to [p]\psi, \psi \to [q]\rho \vDash_{DL} \varphi \to [p;q]\rho,$$
$$\varphi \to \langle p\rangle\psi, \psi \to \langle q\rangle\rho \vDash_{DL} \varphi \to \langle p;q\rangle\rho.$$

*Proof.* Let $\mathcal{M}$ be a structure such that $\mathcal{M} \vDash_{DL} \varphi \to [p]\psi$ and $\mathcal{M} \vDash_{DL} \psi \to [q]\rho$. Let $v \in \operatorname{Val}\mathcal{M}$ and assume $(\mathcal{M}, v) \vDash_{DL} \varphi$. We need to show $(\mathcal{M}, v) \vDash_{DL} [p;q]\rho$, or in other words, $(\mathcal{M}, u) \vDash_{DL} \rho$ for all $u \in (p;q)^{\mathcal{M}}(v)$. Therefore, let $u \in (p;q)^{\mathcal{M}}(v)$. This means that there is some $w \in \operatorname{Val}\mathcal{M}$ such that $v\ p^{\mathcal{M}}\ w$ and $w\ q^{\mathcal{M}}\ u$. By assumption, we have

$$(\mathcal{M}, v) \vDash_{DL} \varphi \text{ and} \tag{3.1}$$
$$(\mathcal{M}, v) \vDash_{DL} \varphi \to [p]\psi, \tag{3.2}$$

so we easily obtain $(\mathcal{M}, v) \vDash_{DL} [p]\psi$ and hence $(\mathcal{M}, w) \vDash_{DL} \psi$. By repeating the same argument with $w, \psi, q, \rho$ instead of $v, \varphi, p, \psi$, we deduce $(\mathcal{M}, u) \vDash_{DL} \rho$.

The diamond case is handled analogously. □

We now prove that the verification conditions indeed have the desired property.

**Theorem 3.15.** *Let $p$ be an annotated* **IMP** *program and $\varphi, \psi$ DL formulas. Then*

$$\mathrm{VC}^{\Box}(\{\varphi\}\, p\, \{\psi\}) \vDash^{ind}_{DL} \varphi \to [p]\psi,$$
$$\mathrm{VC}^{\Diamond}(\{\varphi\}\, p\, \{\psi\}) \vDash^{ind}_{DL} \varphi \to \langle p\rangle\psi.$$

*Proof.* By induction on $p$. Let $\mathcal{M}$ be an inductive structure. We need to show

$$\mathcal{M} \vDash_{DL} \bigwedge \mathrm{VC}^{\Box}(\{\varphi\}\, p\, \{\psi\}) \Rightarrow \mathcal{M} \vDash_{DL} \varphi \to [p]\psi,$$
$$\mathcal{M} \vDash_{DL} \bigwedge \mathrm{VC}^{\Diamond}(\{\varphi\}\, p\, \{\psi\}) \Rightarrow \mathcal{M} \vDash_{DL} \varphi \to \langle p\rangle\psi.$$

The base case of **skip** follows immediately because

$$\mathrm{VC}^{\Box}(\{\varphi\}\, \mathbf{skip}\, \{\psi\}) = \{\varphi \to \psi\} = \mathrm{VC}^{\Diamond}(\{\varphi\}\, \mathbf{skip}\, \{\psi\})$$

by definition and

$$\mathcal{M} \vDash_{DL} \varphi \to [\mathbf{skip}]\psi \Leftrightarrow \mathcal{M} \vDash_{DL} \varphi \to \psi \Leftrightarrow \mathcal{M} \vDash_{DL} \varphi \to \langle p\rangle\psi$$

by Proposition 3.8.

For the case of $p; x := t_1|...|t_n$ we have

$$\mathrm{VC}^{\Box}(\{\varphi\}\, p; x := t_1|...|t_n\, \{\psi\}) = \mathrm{VC}^{\Box}\left(\{\varphi\}\, p \left\{\bigwedge_{i=1}^{n} \psi[x\backslash a_i\right\}\right)$$

by definition. By the induction hypothesis, we can deduce

$$\mathcal{M} \vDash_{DL} \varphi \to [p] \bigwedge_{i=1}^{n} \psi[x\backslash t_i].$$

From there, we obtain

$$\mathcal{M} \vDash_{DL} \varphi \to [p] \bigwedge_{i=1}^{n} \psi[x\backslash t_i] \Leftrightarrow \mathcal{M} \vDash_{DL} \varphi \to [p][x := t_1|...|t_n]\psi$$
$$\Leftrightarrow \mathcal{M} \vDash_{DL} \varphi \to [p; x := t_1|...|t_n]\psi$$

by Propositions 3.5 and 3.8. The diamond conditions are handled analogously.

For $p; \{\alpha\}$ **if** $\gamma$ **then** $q$ **else** $r$, let us first consider the box case. Assume

$$\mathcal{M} \vDash_{DL} \bigwedge \mathrm{VC}^{\Box} (\{\varphi\}\, p; \{\alpha\}\, \textbf{if } \gamma \textbf{ then } q \textbf{ else } r\, \{\psi\})\,.$$

By the induction hypothesis, we obtain

$$\mathcal{M} \vDash_{DL} \varphi \to [p]\alpha, \tag{3.3}$$
$$\mathcal{M} \vDash_{DL} \alpha \wedge \gamma \to [q]\psi, \tag{3.4}$$
$$\mathcal{M} \vDash_{DL} \alpha \wedge \neg\gamma \to [r]\psi. \tag{3.5}$$

Our goal is

$$\mathcal{M} \vDash_{DL} \varphi \to [p; \{\alpha\}\textbf{if } \gamma \textbf{ then } q \textbf{ else } r]\psi.$$

By Lemma 3.14 and (3.3), it suffices to show

$$\mathcal{M} \vDash_{DL} \alpha \to [\textbf{if } \gamma \textbf{ then } q \textbf{ else } r]\psi.$$

We can transform this formula according to Propositions 3.5 and 3.8:

$$\begin{aligned}
\alpha \to [\textbf{if } \gamma \textbf{ then } q \textbf{ else } r]\psi &\Leftrightarrow_{DL} \alpha \to [(\gamma?; q) \cup (\neg\gamma?; r)]\psi \\
&\Leftrightarrow_{DL} \alpha \to ([\gamma?; q]\psi \wedge [\neg\gamma?; r]\psi) \\
&\Leftrightarrow_{DL} \alpha \to ((\gamma \to [q]\psi) \wedge (\neg\gamma \to [r]\psi)) \\
&\Leftrightarrow_{DL} (\alpha \wedge \gamma \to [q]\psi) \wedge (\alpha \wedge \neg\gamma[r]\psi).
\end{aligned}$$

$\mathcal{M} \vDash_{DL} (\alpha \wedge \gamma \to [q]\psi) \wedge (\alpha \wedge \neg\gamma[r]\psi)$ follows immediately from (3.4) and (3.5). The diamond case is completely analogous.

Now let us consider $\varphi \to [p; \textbf{while } \gamma\ \{\iota; t\} \textbf{ do } q]$. Assume

$$\mathcal{M} \vDash_{DL} \mathrm{VC}^{\Box} (\{\varphi\}\, p; \textbf{while } \gamma\ \{\iota; t\} \textbf{ do } q\, \{\psi\})\,.$$

In the same way as in the **if-then-else** case, we use the induction hypothesis to immediately dispense with $\varphi \to [p]\iota$ and obtain

$$\mathcal{M} \vDash_{DL} \iota \wedge \gamma \to [q]\iota, \tag{3.6}$$
$$\mathcal{M} \vDash_{DL} \iota \wedge \neg\gamma \to \psi. \tag{3.7}$$

What we need to show is $\mathcal{M} \vDash_{DL} \iota \to [\textbf{while } \gamma \, \{\iota; t\} \textbf{ do } q]\psi$, which we can rewrite:

$$\begin{aligned} \iota \to [\textbf{while } \gamma \, \{\iota; t\} \textbf{ do } q]\psi &\Leftrightarrow \iota \to [(\gamma?; q)^*; \neg\gamma?]\psi \\ &\Leftrightarrow \iota \to [(\gamma?; q)^*][\neg\gamma?]\psi \\ &\Leftrightarrow \iota \to [(\gamma?; q)^*](\neg\gamma \to \psi) \end{aligned}$$

Because of (3.7), it is sufficient to show

$$\mathcal{M} \vDash_{DL} \iota \to [(\gamma?; q)^*]\iota,$$

which is equivalent to

$$\mathcal{M} \vDash_{DL} \iota \to \iota \wedge [(\gamma?; q)^*](\iota \to [\gamma?; q]\iota))$$

by the induction principle (9 in 3.5). We obtain $\mathcal{M} \vDash_{DL} (\iota \to [\gamma?; q]\iota)$ from (3.6), so $\mathcal{M} \vDash_{DL} [(\gamma?; q)^*](\iota \to [\gamma?; q]\iota)$ follows by generalization and the rest by simple propositional manipulations.

Now for the diamond case. Assume

$$\mathcal{M} \vDash_{DL} \bigwedge \mathrm{VC}^{\Diamond} (\{\varphi\} \, p; \textbf{while } \gamma \, \{\iota; t\} \textbf{ do } q \, \{\psi\}) \, .$$

By a similar argument as before, we obtain

$$\mathcal{M} \vDash_{DL} \iota \wedge \gamma \wedge t = z \to \langle q \rangle (\iota \wedge t < z), \tag{3.8}$$

$$\mathcal{M} \vDash_{DL} \iota \wedge \neg\gamma \to \psi. \tag{3.9}$$

and only need to prove

$$\begin{aligned} &\mathcal{M} \vDash_{DL} \iota \to \langle \textbf{while } \gamma \textbf{ do } q \rangle \iota \\ \Leftrightarrow \; &\mathcal{M} \vDash_{DL} \iota \to \langle (\gamma?; q)^* \rangle (\iota \wedge \neg\gamma). \end{aligned}$$

For technical reasons, we will instead prove

$$\mathcal{M} \vDash_{DL} \iota \to \langle ((\gamma?; q) \cup \neg\gamma?)^* \rangle (\iota \wedge \neg\gamma). \tag{3.10}$$

This substitution is justified because the programs $(\gamma?; q)^*$ and $((\gamma?; q) \cup \neg\gamma?)^*$ are semantically equivalent; we show this in Lemma 3.16. To prove (3.10), we will

apply the DL convergence rule (see Definition 3.10) to the program $((\gamma?;q)\cup\neg\gamma?)^*$ and the formula

$$\tau(n) \equiv \iota \wedge (\neg\gamma \vee t \leq n),$$

which intuitively says that there are at most $n$ iterations left in the loop.

Consider the premise of the convergence inference:

$$\begin{aligned}\sigma(n) \equiv \tau(sn) &\to \langle(\gamma?;q)\cup\neg\gamma?\rangle\tau(n)\\ &\Leftrightarrow_{DL} \tau(sn) \to \langle\gamma?;q\rangle\tau(n) \vee \langle\neg\gamma?\rangle\tau(n)\\ &\Leftrightarrow_{DL} \tau(sn) \to (\gamma \wedge \langle q\rangle\tau(n)) \vee (\neg\gamma \wedge \tau(n))\\ &\Leftrightarrow_{DL} \iota \wedge (\neg\gamma \vee t \leq sn) \to (\gamma \wedge \langle q\rangle(\iota \wedge (\neg\gamma \vee t \leq n)))\\ &\qquad\qquad \vee (\neg\gamma \wedge \iota \wedge (\neg\gamma \vee t \leq n)).\end{aligned}$$

In order to use the convergence rule, we must prove $\mathcal{M} \vDash_{DL} \sigma$. For each valuation $v$, there are two cases to consider.

- $v \vDash_{DL} \gamma$: In this case, (3.8) implies

  $$v \vDash_{DL} \iota \wedge t = z \to \langle q\rangle(\iota \wedge t < z) \tag{3.11}$$

  and $v \vDash \sigma(n)$ is equivalent to

  $$v \vDash_{DL} \iota \wedge t \leq sn \to \langle q\rangle(\iota \wedge t \leq n),$$

  which is easily derived from (3.11) and the axioms of $\mathbf{Q}_{\min}$.
- $v \vDash_{DL} \neg\gamma$: $\sigma$ reduces to the tautology $\iota \to \iota$ at $v$.

We have now shown $\mathcal{M} \vDash_{DL} \sigma(n)$. By the convergence rule, we may conclude

$$\begin{aligned}&\mathcal{M} \vDash_{DL} \tau(t) \to \langle((\gamma?;q)\cup\neg\gamma?)^*\rangle\tau(0)\\ \Rightarrow\ &\mathcal{M} \vDash_{DL} \iota \wedge (\neg\gamma \vee t \leq t) \to \langle((\gamma?;q)\cup\neg\gamma?)^*\rangle\tau(0)\\ \Rightarrow\ &\mathcal{M} \vDash_{DL} \iota \to \langle((\gamma?;q)\cup\neg\gamma?)^*\rangle\tau(0).\end{aligned}$$

To conclude the proof of (3.10), we only need to show $\mathcal{M} \vDash_{DL} \tau(0) \to \neg\gamma$. Towards a contradiction, assume $v \vDash \gamma \wedge t = 0$. Together with

$$v \vDash_{DL} \gamma \wedge \iota \wedge t = z \to \langle q\rangle(\iota \wedge t < z),$$

which we obtain from (3.8), we can deduce $v \vDash_{DL} \langle q \rangle(\iota \wedge t < 0)$, which contradicts the assumption that $\mathcal{M} \vDash \mathbf{Q}_{\min}$. Therefore $v \vDash_{DL} \tau(0) \to \neg\gamma$ and, since $v$ was arbitrary, $\mathcal{M} \vDash_{DL} \tau(0) \to \neg\gamma$. □

**Lemma 3.16.** *Let $\gamma$ be a formula and $p$ a program. Let*

$$r = \gamma?; p,$$
$$s = (\gamma?; p) \cup \neg\gamma?.$$

*Then $r^*$ and $s^*$ are equivalent, i.e., $(r^*)^{\mathcal{M}} = (s^*)^{\mathcal{M}}$ for all structures $\mathcal{M}$.*

*Proof.* Let $\mathcal{M}$ be a structure. Then $(r^*)^{\mathcal{M}} \subseteq (s^*)^{\mathcal{M}}$ is clear. For the other direction, we need to show that $(r^*)^{\mathcal{M}}$ is reflexive and closed under composition with $s$. The former is trivial. For the latter, assume that $(u, v) \in (r^*)^{\mathcal{M}}$ and $(v, w) \in s^{\mathcal{M}}$. This means that either $(v, w) \in (\gamma?; p)^{\mathcal{M}} = r^{\mathcal{M}}$ or $(v, w) \in (\neg\gamma?)^{\mathcal{M}} \subseteq \mathrm{id}_{\mathcal{M}}$. In both cases $(u, w) \in (r^*)^{\mathcal{M}}$ follows immediately. □

**Corollary 3.17.** *Let $p$ be an* **IMP** *program, $p'$ an annotation of $p$, and $\varphi, \psi$ formulas. Then*

$$\mathrm{VC}^{\Box}(\{\varphi\}\, p'\, \{\psi\}) \vDash^{ind}_{DL} \varphi \to [p]\psi,$$
$$\mathrm{VC}^{\Diamond}(\{\varphi\}\, p'\, \{\psi\}) \vDash^{ind}_{DL} \varphi \to \langle p \rangle\psi.$$

*Proof.* From Theorem 3.15 and the semantic equivalence of $p$ and $p'$. □

It follows that each formula of the form $\varphi \to [p]\psi$ or $\varphi \to \langle p \rangle\psi$ induces a formula equation. Consider the case $\varphi \to [p]\psi$ and let $p'$ be an annotation of $p$ in which each **while** and **if** instruction is annotated with a fresh formula variable. Then $\exists \bar{X}.\ \bigwedge \mathrm{VC}^{\Box}(\{\varphi\}\, p'\, \{\psi\})$ is a formula equation whose solvability implies $\vDash^{ind}_{DL} \varphi \to [p]\psi$.

**Definition 3.18** (Verification equations)**.** Let $p$ be an **IMP** program, $\varphi, \psi$ first-order formulas, $n$ the number of loops in $p$, $t_1 : \mathbf{Nat}, ..., t_n : \mathbf{Nat}$ terms, and $p'$ an annotation of $p$ in which each conditional and loop is annotated with a fresh formula variable and the termination witness of the $i$-th loop is $t_i$. Then the formula equations

$$\mathrm{VE}^{\Box}(\{\varphi\}\, p\, \{\psi\}\, ; \bar{t}) \equiv \exists \bar{X}.\ \bigwedge \mathrm{VC}^{\Box}(\{\varphi\}\, p'\, \{\psi\}),$$
$$\mathrm{VE}^{\Diamond}(\{\varphi\}\, p\, \{\psi\}\, ; \bar{t}) \equiv \exists \bar{X}.\ \bigwedge \mathrm{VC}^{\Diamond}(\{\varphi\}\, p'\, \{\psi\})$$

are, respectively, the *box verification equation* and the *diamond verification equation* of $\langle \varphi, p, \psi, \bar{t} \rangle$.

Given $\varphi, p, \psi, \bar{t}$, we are justified in speaking of *the* box (or diamond) verification equation determined by $\langle \varphi, p, \psi, \bar{t} \rangle$ because, while there may be many ways to annotate $p$ in accordance with Definition 3.18, such annotations only differ in the names of the fresh formula variables and are thus equivalent.

## 3.3 From simple induction proofs to programs

We will now investigate the relationship between induction proofs and **IMP** programs. The translation is somewhat more straightforward in the existential case. This is because in an existential sip, terms are introduced "with" the implication, so to speak. By contrast, in the universal case, they are introduced "against" the implication.

### 3.3.1 The existential case

Let $\pi$ be the existential simple induction proof

$$\begin{aligned} \mathcal{S}_0 &\equiv \Gamma(\alpha) \vdash \{\varphi(\alpha, 0, u_i(\alpha))\}_{i=1}^{m} \\ \mathcal{S}_1 &\equiv \Pi(\alpha, \nu, \gamma), \nu < \alpha, \varphi(\alpha, \nu, \gamma) \vdash \{\varphi(\alpha, s\nu, t_i(\alpha, \nu, \gamma))\}_{i=1}^{n} \\ \mathcal{S}_2 &\equiv \Lambda(\alpha, \gamma), \varphi(\alpha, \alpha, \gamma) \vdash \psi(\alpha). \end{aligned}$$

Observe that the induction formula $\varphi$ plays a role similar to that of a loop invariant: $\mathcal{S}_0$ asserts that $\varphi$ holds "at the beginning", $\mathcal{S}_1$ asserts that the step from $\nu$ to $s\nu$ preserves $\varphi$, and $\mathcal{S}_2$ asserts that "at the end", $\varphi$ implies a certain goal formula. In this section, we will make this analogy precise. Since, in general, there are multiple step and base terms, it is intuitively clear that extracting a program from $\pi$ must result in a nondeterministic program. This raises a problem, however: usually, an invariant of a nondeterministic program is a formula that holds across *all* possible executions of the program. By contrast, $\mathcal{S}_1$ only requires that if $\varphi(\alpha, \nu, \gamma)$ holds, then so does $\varphi(\alpha, s\nu, t_i)$ for some $i \in \{1, ..., n\}$. This corresponds

to the concept of an invariant that holds for some execution of a nondeterministic program. Dynamic logic has allowed us to formalize this notion.

**Definition 3.19** (Program of an existential simple induction proof)**.** Let $\pi$ be the existential simple induction proof

$$\begin{aligned} \mathcal{S}_0 \equiv \Gamma(\alpha) &\vdash \{\varphi(\alpha, 0, u_i(\alpha))\}_{i=1}^{m} \\ \mathcal{S}_1 \equiv \Pi(\alpha, \nu, \gamma), \nu < \alpha, \varphi(\alpha, \nu, \gamma) &\vdash \{\varphi(\alpha, s\nu, t_i(\alpha, \nu, \gamma))\}_{i=1}^{r} \\ \mathcal{S}_2 \equiv \Lambda(\alpha, \gamma), \varphi(\alpha, \alpha, \gamma) &\vdash \psi(\alpha). \end{aligned}$$

Then $p(\pi)$ is the annotated **IMP** program

---

$\nu := 0;$
$\gamma := u_1(\alpha)|...|u_m(\alpha);$
**while** $\nu < \alpha \wedge \bigwedge \Pi(\alpha, \nu, \gamma)$ **do**
  $\{\varphi(\alpha, \nu, \gamma) \wedge \nu \leq \alpha; \alpha - \nu\}$
  $\gamma := t_1(\alpha, \nu, \gamma) \mid \cdots \mid t_r(\alpha, \nu, \gamma);$
  $\nu := \nu + 1;$
**end while**

---

We immediately obtain

**Lemma 3.20.** *Let $\pi$ be the existential $\mathbf{Q}_{\min}$-sip*

$$\begin{aligned} \Gamma(\alpha) &\vdash \{\varphi(\alpha, 0, u_i(\alpha))\}_{i=1}^{m} \\ \Pi(\alpha, \nu, \gamma), \nu < \alpha, \varphi(\alpha, \nu, \gamma) &\vdash \{\varphi(\alpha, s\nu, t_i(\alpha, \nu, \gamma))\}_{i=1}^{r} \\ \Lambda(\alpha, \gamma), \varphi(\alpha, \alpha, \gamma) &\vdash \psi(\alpha). \end{aligned}$$

*Then*

$$\mathbf{Q}_{\min} \models \mathrm{VC}^{\Diamond}\left(\left\{\bigwedge \Gamma(\alpha)\right\} \langle p(\pi)\rangle \left\{\bigwedge \Lambda(\alpha, \gamma) \wedge \nu = \alpha \to \psi(\alpha)\right\}\right)$$

*and consequently*

$$\models_{DL}^{ind} \bigwedge \Gamma(\alpha) \to \langle p(\pi)\rangle(\bigwedge \Lambda(\alpha, \gamma) \wedge \nu = \alpha \to \psi(\alpha)).$$

*Proof.* Consider $\mathrm{VC}^{\Diamond}\left(\{\bigwedge \Gamma(\alpha)\}\, p(\pi)\, \{\bigwedge \Lambda(\alpha,\gamma) \wedge \nu = \alpha \to \psi(\alpha)\}\right)$:

$$\begin{aligned}
\Gamma(\alpha) &\vdash \{\varphi(\alpha, 0, u_i)\}_{i=1}^{m} \\
\Gamma(\alpha) &\vdash 0 \leq \alpha \\
\Pi(\alpha,\nu,\gamma), \nu < \alpha, \varphi(\alpha,\nu,\gamma), \nu \leq \alpha, \alpha - \nu = z &\vdash \{\varphi(\alpha, s\nu, t_i)\}_{i=1}^{r} \\
\Pi(\alpha,\nu,\gamma), \nu < \alpha, \varphi(\alpha,\nu,\gamma), \nu \leq \alpha, \alpha - \nu = z &\vdash s\nu \leq \alpha \\
\Pi(\alpha,\nu,\gamma), \nu < \alpha, \varphi(\alpha,\nu,\gamma), \nu \leq \alpha, \alpha - \nu = z &\vdash \alpha - s\nu < z \\
\varphi(\alpha,\nu,\gamma), \nu \leq \alpha, (\neg\nu < \alpha \vee \neg \bigwedge \Pi(\alpha,\nu,\gamma)) &\vdash \bigwedge \Lambda(\alpha,\gamma) \wedge \nu = \alpha \to \psi(\alpha).
\end{aligned}$$

The last sequent can be rearranged to

$$\Lambda(\alpha,\gamma), \nu = \alpha, \varphi(\alpha,\nu,\gamma), \nu \leq \alpha, (\neg\nu < \alpha \vee \neg \bigwedge \Pi(\alpha,\nu,\gamma)) \vdash \psi(\alpha).$$

Since the formulas

$$\begin{gathered}
0 \leq \alpha, \\
\nu < \alpha \wedge \alpha - \nu = z \to \nu \leq \alpha \wedge s\nu \leq \alpha \wedge \alpha - s\nu < z, \\
\nu = \alpha \to \nu \leq \alpha \wedge \neg\nu < \alpha
\end{gathered}$$

are valid modulo $\mathbf{Q}_{\min}$, we can reduce these to

$$\begin{aligned}
\Gamma(\alpha) &\vdash \{\varphi(\alpha, 0, u_i)\}_{i=1}^{m} \\
\Pi(\alpha,\nu,\gamma), \nu < \alpha, \varphi(\alpha,\nu,\gamma) &\vdash \{\varphi(\alpha, s\nu, t_i)\}_{i=1}^{r} \\
\Lambda(\alpha,\gamma), \varphi(\alpha,\alpha,\gamma) &\vdash \psi(\alpha)
\end{aligned}$$

which is just $\pi$. The sequents of $\pi$ are valid modulo $\mathbf{Q}_{\min}$ by definition, so it follows that $\mathbf{Q}_{\min} \models \mathrm{VC}^{\Diamond}\left(\{\bigwedge \Gamma'(\alpha)\}\, p(\pi)\, \{\bigwedge \Lambda'(\alpha,\gamma) \wedge \nu = \alpha \to \psi(\alpha)\}\right)$. The second part of the claim follows by Theorem 3.15. □

**Corollary 3.21.** *Let $\pi$ be an existential simple induction proof as in Lemma 3.20. Let $\forall\Gamma, \forall\Pi, \forall\Lambda$ be sets of closed $\Pi_1$ formulas such that $\Gamma, \Pi, \Lambda$ are instances of $\forall\Gamma, \forall\Pi, \forall\Lambda$, respectively. Then $\forall\Gamma, \forall\Pi, \forall\Lambda \models^{ind}_{DL} \psi(\alpha)$.*

*Proof.* Let $\mathcal{M}$ be an inductive model of $\forall\Gamma \cup \forall\Pi \cup \forall\Lambda$. We first prove $\mathcal{M} \models_{DL} \langle p(\pi)\rangle\psi(\alpha)$. By Lemma 3.20, we have

$$\mathcal{M} \models_{DL} \Gamma(\alpha) \to \langle p(\pi)\rangle(\Lambda(\alpha,\gamma) \wedge \nu = \alpha \to \psi(\alpha))$$

and hence

$$\mathcal{M} \models_{DL} \langle p(\pi)\rangle(\nu = \alpha \to \psi(\alpha)).$$

Observe that in $\mathcal{M}$, the loop condition $\nu < \alpha \wedge \bigwedge \Pi(\alpha, \nu, \gamma)$ is equivalent to $\nu < \alpha$, so we have

$$\begin{aligned}\mathcal{M} \models_{DL} &[p(\pi)](\nu \leq \alpha \wedge \nu \geq \alpha)\\ \Leftrightarrow &[p(\pi)](\nu = \alpha).\end{aligned}$$

It is easy to see that $\langle p(\pi)\rangle(\nu = \alpha \to \psi(\alpha))$ and $[p(\pi)](\nu = \alpha)$ jointly imply $\langle p(\pi)\rangle\psi(\alpha)$, and $\mathcal{M} \models_{DL} \langle p(\pi)\rangle\psi(\alpha)$ follows immediately.

By the definition of a simple induction proof, $\psi(\alpha)$ is fully indicated, so $\alpha$ is its only free variable. Now let $v \in \mathrm{Val}(\mathcal{M})$. As we have just shown, $(\mathcal{M}, v) \models_{DL} \langle p(\pi)\rangle\psi(\alpha)$, so there is a $w \in \mathrm{Val}(\mathcal{M})$ such that $(v, w) \in p(\pi)^{\mathcal{M}}$ and $(\mathcal{M}, w) \models_{DL} \psi(\alpha)$. Since $p(\pi)$ never modifies $\alpha$, $v$ and $w$ must agree on $\alpha$, and hence $(\mathcal{M}, v) \models_{DL} \psi(\alpha)$. Since $v$ was arbitrary, we obtain $\mathcal{M} \models_{DL} \psi(\alpha)$. □

It is worthwhile to look at this corollary and its proof in a little more detail. In a sense, the statement $\forall\Gamma, \forall\Pi, \forall\Lambda \models^{ind}_{DL} \psi(\alpha)$ is unremarkable: we already know that $\pi$ being a sip implies that $\forall\Gamma, \forall\Pi, \forall\Lambda \vdash \psi(\alpha)$ is provable in **LK** with induction. What is interesting about this corollary is that we prove it purely by reasoning about programs, although we obviously presuppose some notion of induction in the definition of inductive structures.

It is straightforward to obtain results analogous to Lemma 3.20 and Corollary 3.21 for simple induction proof schemas.

**Definition 3.22** (Program of an existential simple induction proof schema)**.** Let $\pi$ be the existential simple induction proof schema with the sequents

$$\begin{aligned}\Gamma(\alpha) &\vdash \{X(\alpha, 0, u_i(\alpha))\}_{i=1}^{m}\\ \Pi(\alpha, \nu, \gamma), \nu < \alpha, X(\alpha, \nu, \gamma) &\vdash \{X(\alpha, s\nu, t_i(\alpha, \nu, \gamma))\}_{i=1}^{r}\\ \Lambda(\alpha, \gamma), X(\alpha, \alpha, \gamma) &\vdash \psi(\alpha).\end{aligned}$$

Then $p(\pi)$ is the annotated **IMP** program

$\nu := 0;$
$\gamma := u_1(\alpha)|...|u_m(\alpha);$
**while** $\nu < \alpha \wedge \bigwedge \Pi(\alpha, \nu, \gamma)$ **do**
  $\{X(\alpha, \nu, \gamma); \alpha - \nu\}$
  $\gamma := t_1(\alpha, \nu, \gamma)|...|t_r(\alpha, \nu, \gamma);$
  $\nu := s\nu;$
**end while**

This leads us to another solution problem.

**Definition 3.23** (Existential verification problem)**.** Let $\mathcal{L}$ be an arithmetical language, $\mathcal{C}$ the class of quantifier-free $\mathcal{L}$-formulas of the form $\varphi(\alpha, \nu, \gamma) \wedge \nu \leq \alpha$, where $\nu, \alpha \colon \mathbf{Nat}$, and $\Phi$ the set of all verification equations

$$\mathrm{VE}^{\Diamond}\left(\{\Gamma\}\, p(\pi)\, \{\Lambda \wedge \nu = \alpha \to \psi\}\, ; \alpha - \nu\right)$$

where $\pi$ is an existential sip-schema. Then $\langle \mathbf{Q}_{\min}, \Phi, \mathcal{C} \rangle$ is the *existential verification problem* over $\mathcal{L}$.

We can now state this chapter's main theorem in its existential version.

**Theorem 3.24.** *Let $\mathcal{L}$ be an arithmetical language. Then an $\mathcal{L}$-formula $\varphi$ is a solution of the instance $\pi$ of the existential induction problem over $\mathbf{Q}_{\min}$ iff $\varphi \wedge \nu \leq \alpha$ is a solution of the instance*

$$\mathrm{VE}^{\Diamond}\left(\{\Gamma\}\, p(\pi)\, \{\Lambda \wedge \nu = \alpha \to \psi\}\, ; \alpha - \nu\right)$$

*of the existential verification problem.*

*Proof.* Let $\pi$ be the existential simple induction proof schema

$$\begin{aligned} \mathcal{S}_0 &\equiv \Gamma(\alpha) \vdash \{X(\alpha, 0, u_i(\alpha))\}_{i=1}^{m} \\ \mathcal{S}_1 &\equiv \Pi(\alpha, \nu, \gamma), \nu < \alpha, X(\alpha, \nu, \gamma) \vdash \{X(\alpha, s\nu, t_i(\alpha, \nu, \gamma))\}_{i=1}^{r} \\ \mathcal{S}_2 &\equiv \Lambda(\alpha, \gamma), X(\alpha, \alpha, \gamma) \vdash \psi(\alpha). \end{aligned}$$

Then the set of verification conditions

$$\mathrm{VC}^{\Diamond}\left(\left\{\bigwedge \Gamma(\alpha)\right\} p(\pi) \left\{\bigwedge \Lambda(\alpha, \beta) \wedge \nu = \alpha \to \psi(\alpha)\right\}\right)$$

contains the formulas (written in sequent form)

$$\begin{aligned}
&\mathcal{S}'_0 \equiv \Gamma(\alpha) \vdash \{X(\alpha, 0, u_i(\alpha))\}_{i=1}^m \\
&\mathcal{S}'_1 \equiv \Gamma(\alpha) \vdash 0 \leq \alpha \\
&\mathcal{S}'_2 \equiv \Pi(\alpha, \nu, \gamma), X(\alpha, \nu, \gamma), \nu < \alpha, \alpha - \nu = z \vdash \{X(\alpha, s\nu, t_i(\alpha, \nu, \gamma))\}_{i=1}^r \\
&\mathcal{S}'_3 \equiv \Pi(\alpha, \nu, \gamma), X(\alpha, \nu, \gamma), \nu < \alpha, \alpha - \nu = z \vdash s\nu \leq \alpha \\
&\mathcal{S}'_4 \equiv \Pi(\alpha, \nu, \gamma), X(\alpha, \nu, \gamma), \nu < \alpha, \alpha - \nu = z \vdash \alpha - s\nu < z \\
&\mathcal{S}'_5 \equiv \Lambda(\alpha, \gamma), \nu = \alpha, (\neg\nu < \alpha \vee \neg \bigwedge \Pi(\alpha, \nu, \gamma)), X(\alpha, \nu, \gamma) \vdash \psi(\alpha).
\end{aligned}$$

Since we are only interested in solutions of the form $\rho(\alpha, \nu, \gamma) \wedge \nu \leq \alpha$, we can replace these sequents by

$$\begin{aligned}
&\mathcal{S}'_0 \equiv \Gamma(\alpha) \vdash \{X(\alpha, 0, u_i(\alpha))\}_{i=1}^m \\
&\mathcal{S}'_1 \equiv \Gamma(\alpha) \vdash 0 \leq \alpha \\
&\mathcal{S}'_2 \equiv \Pi(\alpha, \nu, \gamma), X(\alpha, \nu, \gamma), \nu < \alpha, \alpha - \nu = z \vdash \{X(\alpha, s\nu, t_i(\alpha, \nu, \gamma))\}_{i=1}^r \\
&\mathcal{S}'_3 \equiv \Pi(\alpha, \nu, \gamma), X(\alpha, \nu, \gamma), \nu < \alpha, \alpha - \nu = z \vdash s\nu \leq \alpha \\
&\mathcal{S}'_4 \equiv \Pi(\alpha, \nu, \gamma), X(\alpha, \nu, \gamma), \nu < \alpha, \alpha - \nu = z \vdash \alpha - s\nu < z \\
&\mathcal{S}'_5 \equiv \Lambda(\alpha, \gamma), \nu = \alpha, (\neg\nu < \alpha \vee \neg \bigwedge \Pi(\alpha, \nu, \gamma)), X(\alpha, \nu, \gamma), \nu \leq \alpha \vdash \psi(\alpha).
\end{aligned}$$

and ignore the condition on solutions.

The theorem then easily follows by observing that for any formula $\varphi(\alpha, \nu, \gamma)$,

$$\mathbf{Q}_{\min} \vDash \bigwedge_{i=0}^{2} \mathcal{S}_i[X\backslash\varphi] \leftrightarrow \bigwedge_{i=0}^{5} \mathcal{S}'_i[X\backslash\varphi]. \qquad \square$$

### 3.3.2 The universal case

Consider the universal simple induction proof $\pi$:

$$\begin{aligned}
\Gamma(\alpha, \gamma) &\vdash \varphi(\alpha, 0, \gamma), \\
\Pi(\alpha, \nu, \gamma), \nu < \alpha, \{\varphi(\alpha, \nu, t_i(\alpha, \nu, \gamma))\}_{i=1}^r &\vdash \varphi(\alpha, s\nu, \gamma) \\
\Lambda(\alpha), \{\varphi(\alpha, \alpha, u_i(\alpha))\}_{i=1}^m &\vdash \psi(\alpha).
\end{aligned}$$

We cannot translate this into a program quite as straightforwardly as in the existential case. In the second sequent, the step terms occur in the antecedent, but

the natural direction is from the antecedent to the succedent. The same holds for the terms in the third sequent. Consequently, we will have to rewrite $\pi$ a bit before we can proceed. By simple propositional manipulation, we can obtain the equivalent form

$$\begin{aligned} \neg\varphi(\alpha, 0, \beta) &\vdash \neg\Gamma(\alpha, \beta), \\ \Pi(\alpha, \nu, \gamma), \nu < \alpha, \neg\varphi(\alpha, s\nu, \gamma) &\vdash \{\neg\varphi(\alpha, \nu, t_i(\alpha, \nu, \gamma))\}_{i=1}^{r} \\ \Lambda(\alpha), \neg\psi(\alpha) &\vdash \{\neg\varphi(\alpha, \alpha, u_i(\alpha))\}_{i=1}^{m}. \end{aligned}$$

In the same way as an existential simple induction proof suggests a loop counting from 0 to $\alpha$, here we obtain a loop counting down from $\alpha$ to 0. We capture this intuition in the following definition.

**Definition 3.25** (Program of a universal simple induction proof)**.** Let $\pi$ be the universal simple induction proof

$$\begin{aligned} \Gamma(\alpha, \beta) &\vdash \varphi(\alpha, 0, \beta), \\ \Pi(\alpha, \nu, \gamma), \nu < \alpha, \{\varphi(\alpha, \nu, t_i(\alpha, \nu, \gamma))\}_{i=1}^{r} &\vdash \varphi(\alpha, s\nu, \gamma) \\ \Lambda(\alpha), \{\varphi(\alpha, \alpha, u_i(\alpha))\}_{i=1}^{m} &\vdash \psi(\alpha). \end{aligned}$$

Then $p(\pi)$ is the annotated **IMP** program

---

$\nu := \alpha$;
$\gamma := u_1(\alpha)|...|u_m(\alpha)$;
**while** $\nu > 0 \wedge \bigwedge \Pi(\alpha, p\nu, \gamma)$ **do**
  $\{\neg\varphi(\alpha, \nu, \gamma) \wedge \nu \leq \alpha; \nu\}$
  $\nu := p\nu$;
  $\gamma := t_1(\alpha, \nu, \gamma)|...|t_r(\alpha, \nu, \gamma)$;
**end while**

---

The following lemma, echoing Lemma 3.20, follows immediately.

**Lemma 3.26.** *Let $\pi$ be the universal simple induction proof*

$$\begin{aligned} \Gamma(\alpha, \gamma) &\vdash \varphi(\alpha, 0, \gamma), \\ \Pi(\alpha, \nu, \gamma), \nu < \alpha, \{\varphi(\alpha, \nu, t_i(\alpha, \nu, \gamma))\}_{i=1}^{r} &\vdash \varphi(\alpha, s\nu, \gamma) \\ \Lambda(\alpha), \{\varphi(\alpha, \alpha, u_i(\alpha))\}_{i=1}^{m} &\vdash \psi(\alpha). \end{aligned}$$

*Then*

$$\mathbf{Q}_{\min} \models \mathrm{VC}^{\Diamond}\left(\left\{\bigwedge \Lambda(\alpha) \wedge \neg\psi(\alpha)\right\} \langle p(\pi)\rangle \left\{\bigwedge \Gamma(\alpha,\gamma) \wedge \nu = 0 \to \bot\right\}\right)$$

*and consequently*

$$\models^{ind}_{DL} \bigwedge \Lambda(\alpha) \wedge \neg\psi(\alpha) \to \langle p(\pi)\rangle(\bigwedge \Gamma(\alpha,\gamma) \wedge \nu = 0 \to \bot).$$

*Proof.* Consider $\mathrm{VC}^{\Diamond}\left(\{\bigwedge \Lambda(\alpha) \wedge \neg\psi(\alpha)\} \langle p(\pi)\rangle \{\bigwedge \Gamma(\alpha,\gamma) \wedge \nu = 0 \to \bot\}\right)$:

$$\begin{aligned}
\Lambda(\alpha), \neg\psi(\alpha) &\vdash \{\neg\varphi(\alpha,\alpha,u_i(\alpha))\}_{i=1}^{m}\\
\Pi(\alpha,p\nu,\gamma), \nu > 0, \neg\varphi(\alpha,\nu,\gamma), \nu \leq \alpha, \nu = z &\vdash \{\neg\varphi(\alpha,p\nu,t_i(\alpha,p\nu,\gamma))\}_{i=1}^{r}\\
\Pi(\alpha,p\nu,\gamma), \nu > 0, \neg\varphi(\alpha,\nu,\gamma), \nu \leq \alpha, \nu = z &\vdash p\nu \leq \alpha\\
\Pi(\alpha,p\nu,\gamma), \nu > 0, \neg\varphi(\alpha,\nu,\gamma), \nu \leq \alpha, \nu = z &\vdash p\nu < z\\
\left\{\begin{array}{c}(\bigvee \neg\Pi'(\alpha,\nu,\gamma) \vee \neg\nu > 0),\\ \neg\varphi(\alpha,\nu,\gamma),\\ \nu = 0, \Gamma(\alpha,\gamma)\end{array}\right\} &\vdash \bigwedge \Gamma(\alpha,\gamma) \wedge \nu = 0 \to \bot
\end{aligned}$$

These sequents are propositionally equivalent to

$$\begin{aligned}
\Lambda(\alpha), \{\varphi(\alpha,\alpha,u_i(\alpha))\}_{i=1}^{m} &\vdash \psi(\alpha)\\
\left\{\begin{array}{c}\Pi(\alpha,p\nu,\gamma), \nu > 0,\\ \nu \leq \alpha, \nu = z,\\ \{\varphi(\alpha,p\nu,t_i(\alpha,p\nu,\gamma))\}_{i=1}^{r}\end{array}\right\} &\vdash \varphi(\alpha,\nu,\gamma)\\
\Pi(\alpha,p\nu,\gamma), \nu > 0, \nu \leq \alpha, \nu = z &\vdash p\nu \leq \alpha, \varphi(\alpha,\nu,\gamma)\\
\Pi(\alpha,p\nu,\gamma), \nu > 0, \nu \leq \alpha, \nu = z &\vdash p\nu < z, \varphi(\alpha,\nu,\gamma)\\
\Gamma(\alpha,\gamma), (\bigvee \neg\Pi'(\alpha,\nu,\gamma) \vee \neg\nu > 0), \nu = 0 &\vdash \varphi(\alpha,\nu,\gamma).
\end{aligned}$$

The third and fourth sequent are easily seen to be valid modulo $\mathbf{Q}_{\min}$. Moreover, because

$$\begin{aligned}
&\mathbf{Q}_{\min} \models s\nu > 0,\\
&\mathbf{Q}_{\min} \models s\nu \leq \alpha \to \nu < \alpha,\\
&\mathbf{Q}_{\min} \models \nu > 0 \wedge \nu \leq \alpha \to \nu = sp\nu \wedge p\nu < z \wedge p\nu \leq \alpha,\\
&\mathbf{Q}_{\min} \models \neg 0 > 0,
\end{aligned}$$

we can replace $\nu$ and $p\nu$ in sequent 2 by $s\nu$ and $\nu$, respectively, and simplify the sequents to

$$\begin{aligned} \Lambda(\alpha), \{\varphi(\alpha, \alpha, u_i(\alpha))\}_{i=1}^{m} &\vdash \psi(\alpha) \\ \Pi(\alpha, \nu, \gamma), \nu < \alpha, \{\varphi(\alpha, p\nu, t_i(\alpha, \nu, \gamma))\}_{i=1}^{r} &\vdash \varphi(\alpha, s\nu, \gamma), \\ \Gamma(\alpha, \gamma) &\vdash \varphi(\alpha, 0, \gamma) \end{aligned}$$

Again, we have reduced the verification conditions to $\pi$. The claim follows just as in Lemma 3.20. □

**Corollary 3.27.** *Let $\pi$ be a universal simple induction proof as in Lemma 3.26. Let $\forall\Gamma, \forall\Pi, \forall\Lambda$ be sets of closed $\Pi_1$ formulas such that $\Gamma, \Pi, \Lambda$ are instances of $\forall\Gamma, \forall\Pi, \forall\Lambda$, respectively. Then $\forall\Gamma, \forall\Pi, \forall\Lambda \models^{ind}_{DL} \psi(\alpha)$.*

*Proof.* We proceed as in the proof of 3.21. Let $\mathcal{M}$ be inductive with $\mathcal{M} \models \forall\Gamma \cup \forall\Pi \cup \forall\Lambda$. Then by Lemma 3.26,

$$\mathcal{M} \models_{DL} \bigwedge \Lambda(\alpha) \wedge \neg\psi(\alpha) \to \langle p(\pi)\rangle(\bigwedge \Gamma(\alpha, \gamma) \wedge \nu = 0 \to \bot)$$

and consequently

$$\mathcal{M} \models_{DL} \neg\psi(\alpha) \to \langle p(\pi)\rangle(\nu = 0 \to \bot).$$

We can further simplify this formula to $\mathcal{M} \models_{DL} \neg\psi(\alpha) \to \langle p(\pi)\rangle\bot$ because $\mathcal{M} \models \Pi(\alpha, \nu, \gamma)$. The claim follows immediately. □

Again, it is worth considering this proof in a little more detail. We start from the assumption that $\psi(\alpha)$ is false in some state of an inductive model $\mathcal{M}$ that satisfies the axioms in $\forall\Gamma, \forall\Pi, \forall\Lambda$. This immediately implies that $\varphi(\alpha, \alpha, u_i)$ is also false for some term $u_i$. From there, we construct ever smaller counterexamples to $\varphi$ until we arrive at 0, but $\forall\Gamma$ implies $\forall y.\ \varphi(\alpha, 0, y)$, a contradiction. Thus, $\psi(\alpha)$ must be true. This proof corresponds to the formulation of induction as Fermat's "principle of infinite descent": if for every $\nu$ with $\neg\varphi(\nu)$ there is a $\nu' < \nu$ with $\neg\varphi(\nu')$, then $\varphi$ must hold everywhere.

We define the programs corresponding to universal sip-schemas and their solution problems.

**Definition 3.28** (Program of a universal simple induction proof schema)**.** Let $\pi$ be the universal simple induction proof schema with the sequents

$$\begin{aligned} \Gamma(\alpha,\gamma) &\vdash X(\alpha,0,\gamma), \\ \Pi(\alpha,\nu,\gamma), \nu < \alpha, \{X(\alpha,\nu,t_i(\alpha,\nu,\gamma))\}_{i=1}^{r} &\vdash X(\alpha,s\nu,\gamma) \\ \Lambda(\alpha), \{X(\alpha,\alpha,u_i(\alpha))\}_{i=1}^{m} &\vdash \psi(\alpha). \end{aligned}$$

Then $p(\pi)$ is the annotated **IMP** program

---

$\nu := \alpha$;
$\gamma := u_1(\alpha)|...|u_m(\alpha)$;
**while** $\nu > 0 \wedge \bigwedge \Pi(\alpha, p\nu, \gamma)$ **do**
  $\{X(\alpha,\nu,\gamma);\nu\}$
  $\nu := p\nu$;
  $\gamma := t_1(\alpha,\nu,\gamma)|...|t_r(\alpha,\nu,\gamma)$;
**end while**

---

**Definition 3.29** (Universal verification problem)**.** Let $\mathcal{L}$ be an arithmetical language, $\mathcal{C}$ the class of quantifier-free $\mathcal{L}$-formulas of the form $\varphi(\alpha,\nu,\gamma) \wedge \nu \leq \alpha$, where $\nu, \alpha \colon \mathbf{Nat}$, and $\Phi$ the set of all verification equations

$$\mathrm{VE}^{\Diamond}\left(\{\Lambda'(\alpha) \wedge \neg\psi(\alpha)\}\, p(\pi)\, \{\Gamma'(\alpha,\gamma) \wedge \nu = 0 \to \bot\}\,;\nu\right)$$

where $\pi$ is a universal sip-schema. Then $\langle \mathbf{Q}_{\min}, \Phi, \mathcal{C}\rangle$ is the *universal verification problem* over $\mathcal{L}$.

We can now complete this chapter's main result by proving the universal case.

**Theorem 3.30.** *Let $\mathcal{L}$ be an arithmetical language. Then an $\mathcal{L}$-formula $\varphi$ is a solution of the instance $\pi$ of the universal induction problem over $\mathbf{Q}_{\min}$ iff $\neg\varphi \wedge \nu \leq \alpha$ is a solution of the instance*

$$\mathrm{VE}^{\Diamond}\left(\{\Lambda(\alpha) \wedge \neg\psi(\alpha)\}\, p(\pi)\, \{\Gamma(\alpha,\gamma) \wedge \nu = 0 \to \bot\}\,;\nu\right)$$

*of the universal verification problem.*

*Proof.* Let $\pi$ be the universal simple induction proof schema with the sequents

$$\begin{aligned}\Gamma(\alpha,\beta) &\vdash X(\alpha,0,\beta),\\ \Pi(\alpha,\nu,\gamma),\nu<\alpha,\{X(\alpha,\nu,t_i(\alpha,\nu,\gamma))\}_{i=1}^{r} &\vdash X(\alpha,s\nu,\gamma)\\ \Lambda(\alpha),\{X(\alpha,\alpha,u_i(\alpha))\}_{i=1}^{m} &\vdash \psi(\alpha).\end{aligned}$$

Now consider $\mathrm{VC}^{\Diamond}\left(\{\bigwedge\Lambda(\alpha)\wedge\neg\psi(\alpha)\}\,p(\pi)\,\{\bigwedge\Gamma(\alpha,\gamma)\wedge\nu=0\to\bot\}\right)$:

$$\begin{aligned}\Lambda(\alpha),\neg\psi(\alpha) &\vdash \{X(\alpha,\alpha,u_i(\alpha))\}_{i=1}^{m}\\ \Pi(\alpha,p\nu,\gamma),\nu>0,X(\alpha,\nu,\gamma),\nu=z &\vdash \{X(\alpha,p\nu,t_i(\alpha,p\nu,\gamma))\}_{i=1}^{r}\\ \Pi(\alpha,p\nu,\gamma),\nu>0,X(\alpha,\nu,\gamma),\nu=z &\vdash p\nu<\alpha\\ \Pi(\alpha,p\nu,\gamma),\nu>0,X(\alpha,\nu,\gamma),\nu=z &\vdash p\nu<z\\ (\bigvee\neg\Pi(\alpha,\nu,\gamma)\vee\neg\nu>0),X(\alpha,\nu,\gamma) &\vdash \Gamma(\alpha,\gamma)\wedge\nu=0\to\bot.\end{aligned}$$

Now let $\varphi(\alpha,\nu,\gamma)$ be a formula and substitute $\neg\varphi(\alpha,\nu,\gamma)\wedge\nu\leq\alpha$ for $X$:

$$\begin{aligned}\Lambda(\alpha),\neg\psi(\alpha) &\vdash \{\neg\varphi(\alpha,\alpha,u_m(\alpha))\}_{i=1}^{m}\\ \Lambda(\alpha),\neg\psi(\alpha) &\vdash \alpha\leq\alpha\\ \Pi(\alpha,p\nu,\gamma),\nu>0,\neg\varphi(\alpha,\nu,\gamma),\nu\leq\alpha,\nu=z &\vdash \{\neg\varphi(\alpha,p\nu,t_i(\alpha,p\nu,\gamma))\}_{i=1}^{r}\\ \Pi(\alpha,p\nu,\gamma),\nu>0,\neg\varphi(\alpha,\nu,\gamma),\nu\leq\alpha,\nu=z &\vdash p\nu\leq\alpha\\ \Pi(\alpha,p\nu,\gamma),\nu>0,\neg\varphi(\alpha,\nu,\gamma),\nu\leq\alpha,\nu=z &\vdash p\nu<z\\ \left(\bigvee\neg\Pi(\alpha,\nu,\gamma)\vee\neg\nu>0\right),\neg\varphi(\alpha,\nu,\gamma),\nu\leq\alpha &\vdash \Gamma(\alpha,\gamma)\wedge\nu=0\to\bot.\end{aligned}$$

Some rearranging yields

$$\begin{aligned}\Lambda(\alpha),\{\varphi(\alpha,\alpha,u_m(\alpha))\}_{i=1}^{m} &\vdash \psi(\alpha)\\ \Lambda(\alpha),\alpha>\alpha &\vdash \psi(\alpha)\\ \left\{\begin{array}{c}\Pi(\alpha,p\nu,\gamma),\\ \{\varphi(\alpha,p\nu,t_i(\alpha,p\nu,\gamma))\}_{i=1}^{r},\\ \nu>0,\nu\leq\alpha,\nu=z\end{array}\right\} &\vdash \varphi(\alpha,\nu,\gamma)\\ \Pi(\alpha,p\nu,\gamma),\nu>0,\nu\leq\alpha,\nu=z &\vdash p\nu\leq\alpha,\varphi(\alpha,\nu,\gamma)\\ \Pi(\alpha,p\nu,\gamma),\nu>0,\nu\leq\alpha,\nu=z &\vdash p\nu<z,\varphi(\alpha,\nu,\gamma)\\ \left\{\begin{array}{c}\Gamma(\alpha,\gamma),\nu=0,\nu\leq\alpha,\\ (\bigvee\neg\Pi(\alpha,\nu,\gamma)\vee\neg\nu>0)\end{array}\right\} &\vdash \varphi(\alpha,\nu,\gamma).\end{aligned}$$

Modulo $\mathbf{Q}_{\min}$, we can simplify as in the proof of Lemma 3.26:

$$\begin{aligned}\Lambda(\alpha)\left\{\varphi(\alpha,\alpha,u_m(\alpha))\right\}_{i=1}^{m} &\vdash \psi(\alpha)\\ \Pi(\alpha,\nu,\gamma)\left\{\varphi(\alpha,\nu,t_i(\alpha,\nu,\gamma))\right\}_{i=1}^{r},\nu<\alpha &\vdash \varphi(\alpha,s\nu,\gamma)\\ \Gamma(\alpha,\gamma) &\vdash \varphi(\alpha,0,\gamma).\end{aligned}$$

This is exactly

$$\begin{aligned}&\mathcal{S}_0[X\backslash\varphi(\alpha,\nu,\gamma)],\\ &\mathcal{S}_1[X\backslash\varphi(\alpha,\nu,\gamma)],\\ &\mathcal{S}_2[X\backslash\varphi(\alpha,\nu,\gamma)],\end{aligned}$$

which completes the proof. □

## 3.4 Converting parametric grammars to programs

The main result of this chapter was proved in the previous section. In this section, we will show that just as there is a connection between simple induction proofs and **IMP** programs, there is also one between parametric grammars and **IMP** programs: For any parametric grammar, we can define a program that computes exactly the grammar's language.

We will restrict ourselves to the existential case, the universal case works analogously.

**Definition 3.31** (Existential program of a parametric grammar)**.** Let $G$ be a parametric grammar with productions

$$\begin{aligned}\tau &\to w_1^{end}(\alpha,\gamma) \mid \cdots \mid w_\ell^{end}(\alpha,\gamma),\\ \tau &\to w_1(\alpha,\nu,\gamma) \mid \cdots \mid w_k(\alpha,\nu,\gamma),\\ \gamma &\to t_1(\alpha,\nu,\gamma) \mid \cdots \mid t_r(\alpha,\nu,\gamma),\\ \gamma^* &\to u_1(\alpha) \mid \cdots \mid u_m(\alpha).\end{aligned}$$

Then $p^{\exists}(G)$, the existential program of $G$, is the **IMP** program

$\nu := 0;$
$\delta := 0;$
**while** $\nu < \alpha$ **do**
    $\delta := \delta \mid s\delta;$
    $\nu := s\nu;$
**end while**
$\nu := 0;$
$\gamma := u_1(\alpha) \mid \cdots \mid u_m(\alpha);$
**while** $\nu < \delta$ **do**
    $\gamma := t_1(\alpha, \nu, \gamma) \mid \cdots \mid t_r(\alpha, \nu, \gamma);$
    $\nu := s\nu;$
**end while**
**if** $\delta = \alpha$ **then**
    $\tau := w_1^{end}(\alpha, \gamma) \mid \cdots \mid w_\ell^{end}(\alpha, \gamma);$
**else**
    $\tau := w_1(\alpha, \nu, \gamma) \mid \cdots \mid w_k(\alpha, \nu, \gamma);$
**end if**

The first loop in $p(G)$ nondeterministically sets $\delta$ to a value between 0 and $\alpha$. Note that we could not accomplish this in a single nondeterministic assignment because it is not clear a priori how many possible terms there are. Since $\delta$ is the upper limit of the second loop, the second loop will run somewhere between 0 and $\alpha$ times.

For the remainder of this section, we will write $G_n$ for $G_n^\exists$ and $p(G)$ for $p^\exists(G)$.

We will show that $p(G)$ computes the existential instance languages of $G$ in two lemmas. Lemma 3.32 shows that the instance languages of $G$ form an upper bound for the output of $p(G)$; Lemma 3.33 shows that each element of $L(G_n)$ is actually realized.

**Lemma 3.32.** *Recall the definition of* $\mathbf{Q}_{\min}$ *(Definition 3.9). Let* $G$ *be a parametric grammar,* $p(G)$ *its existential program, and* $n \in \mathbb{N}$*. Then*

$$\mathbf{Q}_{\min} \vDash_{DL} \alpha = \bar{n} \to [p(G)]\, \tau \in L(G_n)$$

*where* $\tau \in S$ *abbreviates* $\bigvee_{w \in S} \tau = w$*.*

*Proof.* Note, first of all, that since $\alpha$ never changes throughout $p(G)$, the precondition $\alpha = \bar{n}$ stays valid throughout the entire program. We may thus assume w.l.o.g. that $\alpha = \bar{n}$ is part of all invariants and intermediate formulas.

Let $p'$ be the annotation

---

$\nu := 0;$
$\delta := 0;$
**while** $\nu < \alpha$ **do**
  $\{J; \alpha - \nu\}$
  $\delta := \delta \mid s\delta;$
  $\nu := s\nu;$
**end while**
$\nu := 0;$
$\gamma := u_1(\alpha) \mid \cdots \mid u_m(\alpha);$
**while** $\nu < \delta$ **do**
  $\{\nu \leq \delta \leq \alpha \wedge I; \delta - \nu\}$
  $\gamma := t_1(\alpha, \nu, \gamma) \mid \cdots \mid t_r(\alpha, \nu, \gamma);$
  $\nu := s\nu;$
**end while**
$\{\nu = \delta \leq \alpha \wedge I\}$
**if** $\delta = \alpha$ **then**
  $\tau := w_1^{end}(\alpha, \gamma) \mid \cdots \mid w_\ell^{end}(\alpha, \gamma);$
**else**
  $\tau := w_1(\alpha, \nu, \gamma) \mid \cdots \mid w_k(\alpha, \nu, \gamma);$
**end if**

---

of $P(G)$, where

$$I(\nu, \gamma) \equiv \bigvee_{i=0}^{n} (\nu = \bar{i} \wedge \gamma \in L_i(G_n)),$$
$$J(\alpha, \nu, \delta) \equiv \delta \leq \nu \leq \alpha.$$

Now consider $\mathrm{VC}^{\square}\left(\{\alpha=\bar{n}\}\, P' \left\{\bigvee_{w\in L(G_n)} \tau = w\right\}\right)$:

$$\begin{aligned}
\alpha=\bar{n} &\vdash J(\alpha,0,0)\\
J(\alpha,\nu,\delta),\nu<\alpha &\vdash J(\alpha,s\nu,\delta), J(\alpha,s\nu,s\delta)\\
J(\alpha,\nu,\delta),\nu\geq\alpha &\vdash 0\leq\delta\leq\alpha\wedge\bigwedge_{j=1}^{m} I(0,u_j(\alpha))\\
\nu\leq\delta\leq\alpha, I(\nu,\gamma),\nu<\delta &\vdash s\nu\leq\delta\leq\alpha\wedge\bigwedge_{j=1}^{r} I(s\nu,t_j(\alpha,\nu,\gamma))\\
\nu\leq\delta\leq\alpha, I(\nu,\gamma),\nu\geq\delta &\vdash \nu=\delta\leq\alpha\wedge I(\nu,\gamma)\\
\nu=\delta\leq\alpha, I(\nu,\gamma),\delta=\alpha &\vdash \bigwedge_{j=1}^{\ell} w_j^{end}(\alpha,\gamma)\in L(G_n)\\
\nu=\delta\leq\alpha, I(\nu,\gamma),\delta\neq\alpha) &\vdash \bigwedge_{j=1}^{k} w_j(\alpha,\nu,\gamma)\in L(G_n).
\end{aligned}$$

After substituting $I$ and $J$ in the verification conditions and simplifying, we are left with:

$$\begin{aligned}
\alpha=\bar{n} &\vdash 0\leq\alpha\\
\delta\leq\nu<\alpha &\vdash \delta\leq s\nu\leq\alpha\wedge s\delta\leq s\nu\leq\alpha\\
\delta\leq\nu=\alpha &\vdash 0\leq\delta\leq\alpha\wedge\bigwedge_{j=1}^{m} u_j(\alpha)\in L_0(G_n) \qquad (3.12)\\
\left\{\begin{array}{c}\nu<\delta\leq\alpha,\\ \bigvee_{i=0}^{n}(\nu=\bar{i}\wedge\gamma\in L_i(G_n))\end{array}\right\} &\vdash s\nu\leq\delta\leq\alpha\\
\left\{\begin{array}{c}\nu<\delta\leq\alpha,\\ \bigvee_{i=0}^{n}(\nu=\bar{i}\wedge\gamma\in L_i(G_n))\end{array}\right\} &\vdash \left\{(s\nu=\bar{i}\wedge t_j(\alpha,\nu,\gamma)\in L_i(G_n))\right\}_{i=0}^{n}\\
& \qquad j\in\{1,...,r\}, \qquad (3.13)\\
\left\{\begin{array}{c}\nu=\delta\leq\alpha,\\ \bigvee_{i=0}^{n}(\nu=\bar{i}\wedge\gamma\in L_i(G_n))\end{array}\right\} &\vdash \left\{\nu=\bar{i}\wedge\gamma\in L_i(G_n)\right\}_{i=0}^{n}\\
\left\{\begin{array}{c}\nu=\delta=\alpha,\\ \bigvee_{i=0}^{n}(\nu=\bar{i}\wedge\gamma\in L_i(G_n))\end{array}\right\} &\vdash w_j^{end}(\alpha,\gamma)\in L(G_n), j\in\{1,...,\ell\} \qquad (3.14)
\end{aligned}$$

$$\left\{ \begin{array}{c} \nu = \delta < \alpha, \\ \bigvee_{i=0}^{n} (\nu = \bar{i} \wedge \gamma \in L_i(G_n)) \end{array} \right\} \vdash w_j(\alpha, \nu, \gamma) \in L(G_n), j \in \{1, ..., k\}. \quad (3.15)$$

Most of these are easily seen to be valid modulo $\mathbf{Q}_{\min}$, but we comment on four specific ones:

- (3.12): Due to the implicit assumption $\alpha = \bar{n}$, $u_j(\alpha) \in L_0(G_n)$ holds by definition.
- (3.13): Follows by $\alpha = \bar{n}$ and the definition of $L_i(G_n)$.
- (3.14): The antecedent implies $\nu = \bar{n}$ and $\gamma \in L_n(G_n)$ and consequently $w_j^{end}(\alpha, \gamma) \in L(G_n)$ for any $j$.
- (3.15): The antecedent implies $\nu = \bar{i}$ and $\gamma \in L_i(G_n)$ for some $i < n$ and hence $w_j(\alpha, \nu, \gamma) \in L(G_n)$ for any $j$.

Thus, we have shown $\mathbf{Q}_{\min} \models_{DL} \bigwedge \mathrm{VC}^{\square}(\{\alpha = \bar{n}\}\, p'\, \{\tau \in L(G_n)\})$. $\mathbf{Q}_{\min} \models_{DL} \alpha = \bar{n} \to [p(G)]\tau \in L(G_n)$ follows by Corollary 3.17. □

**Lemma 3.33.** *Let $G$ be a parametric grammar, $p(G)$ its existential program, $n \in \mathbb{N}$ and $w \in L(G_n)$. Then $\mathbf{Q}_{\min} \models_{DL} \alpha = \bar{n} \to \langle p(G) \rangle \tau = w$.*

*Proof.* As in the proof of Lemma 3.32, we will tacitly assume $\alpha = \bar{n}$.

Clearly, $w$ is either of the form $w_i(\bar{n}, \bar{k}, w')$ with $k < n$ and $w' \in L_k(G_n)$ or $w_i^{end}(\bar{n}, w')$ with $w' \in L_n(G_n)$. We will only treat the former case, the latter works analogously. Therefore, let $k < n$ be arbitrary and $w' \in L_k(G_n)$. This implies that there are $i_0, ..., i_{k+1}$ and $w'_0, ..., w'_k$ such that

$$\begin{array}{c} w'_0 = u_{i_0}(\bar{n}) \in L_0(G_n), \\ w'_1 = t_{i_1}(\bar{n}, 0, w'_0) \in L_1(G_n), \\ \vdots \\ w'_k = t_{i_k}(\bar{n}, \overline{k-1}, w'_{k-1}) \in L_k(G_n), \\ w = w_{i_{k+1}}(\bar{n}, \bar{k}, w'_k) \in L(G_n). \end{array}$$

Let $p'$ be the annotation

$\nu := 0;$
$\delta := 0;$
**while** $\nu < \alpha$ **do**
  $\{J; \alpha - \nu\}$
  $\delta := \delta \mid s\delta;$
  $\nu := s\nu;$
**end while**
$\nu := 0;$
$\gamma := u_1(\alpha) \mid \cdots \mid u_m(\alpha);$
**while** $\nu < \delta$ **do**
  $\{\nu \leq \delta \leq \alpha \wedge I; \delta - \nu\}$
  $\gamma := t_1(\alpha, \nu, \gamma) \mid \cdots \mid t_r(\alpha, \nu, \gamma);$
  $\nu := s\nu;$
**end while**
$\{\nu = \delta \leq \alpha \wedge I\}$
**if** $\delta = \alpha$ **then**
  $\tau := w_1^{end}(\alpha, \gamma) \mid \cdots \mid w_\ell^{end}(\alpha, \gamma);$
**else**
  $\tau := w_1(\alpha, \nu, \gamma) \mid \cdots \mid w_k(\alpha, \nu, \gamma);$
**end if**

where

$$I(\nu, \gamma) \equiv \nu \leq \delta = \bar{k} \wedge \bigvee_{j=1}^{k} (\nu = \bar{j} \wedge \gamma = w'_j),$$
$$J(\alpha, \nu, \delta) \equiv \delta \leq \bar{k} \wedge \nu \leq \alpha \wedge \bar{k} - \delta \leq \alpha - \nu.$$

Since $\nu = \alpha$ at the end of the first loop, $J$ forces $\delta$ to become equal to $\bar{k}$, while $I$ encodes the step-by-step construction of $w$.

Now consider $\mathrm{VC}^{\Diamond}(\{\alpha = \bar{n}\}\, P' \,\{\tau = w\})$:

$$\alpha = \bar{n} \vdash J(\alpha, 0, 0)$$
$$\left\{ \begin{array}{c} J(\alpha, \nu, \delta), \\ \nu < \alpha, \\ \alpha - \nu = z \end{array} \right\} \vdash J(\alpha, s\nu, \delta), J(\alpha, s\nu, s\delta) \tag{3.16}$$

$$\left\{\begin{array}{c} J(\alpha,\nu,\delta), \\ \nu<\alpha, \\ \alpha-\nu=z \end{array}\right\} \vdash \alpha - s\nu < z$$

$$J(\alpha,\nu,\delta), \nu \geq \alpha \vdash \{I(0,u_i(\alpha))\}_{i=1}^{m} \tag{3.17}$$

$$\left\{\begin{array}{c} \nu<\delta\leq\alpha, \\ I(\nu,\gamma), \\ \delta-\nu=z \end{array}\right\} \vdash s\nu \leq \delta \leq \alpha$$

$$\left\{\begin{array}{c} \nu<\delta\leq\alpha, \\ I(\nu,\gamma), \\ \delta-\nu=z \end{array}\right\} \vdash \delta - s\nu < z$$

$$\left\{\begin{array}{c} \nu<\delta\leq\alpha, \\ I(\nu,\gamma), \\ \delta-\nu=z \end{array}\right\} \vdash \{I(s\nu,t_i(\alpha,\nu,\gamma))\}_{i=1}^{r} \tag{3.18}$$

$$I(\nu,\gamma), \nu \geq \delta \vdash \nu = \delta \leq \alpha \wedge I(\nu,\gamma)$$

$$\nu=\delta\leq\alpha, I(\nu,\gamma), \delta=\alpha \vdash \{w_i^{end}(\alpha,\gamma)=w\}_{i=1}^{\ell} \tag{3.19}$$

$$\nu=\delta\leq\alpha, I(\nu,\gamma), \delta\neq\alpha \vdash \{w_k(\alpha,\nu,\gamma)=w\}_{i=1}^{k}. \tag{3.20}$$

We once again discuss some of these formulas:

- (3.16): Assume the antecedent. If $\delta = \bar{k}$, the first disjunct of the consequent holds; otherwise—which is to say, if $\delta < \bar{k}$—the second disjunct holds.
- (3.17): In order to show the disjunction in $I$, we have to show one disjunct. We obtain $u_{i_0}(\alpha) = w'_0$ from $w'_0 = u_{i_0}(\bar{n})$ and $\alpha = \bar{n}$.
- (3.18): The antecedent implies $\nu = \bar{j}$ and $\gamma = w'_j$ for some $j \in \{0,...,k-1\}$. Clearly, $s\nu \leq \bar{k}$. Moreover, $w'_{j+1} = t_{i_{j+1}}(\bar{n},\bar{j},w')$, so the rest of the consequent follows.
- (3.19): This formula is trivially true because $I$ and $\delta = \alpha$ jointly imply $\bar{k} = \bar{n}$, in contradiction to our assumption.
- (3.20): The consequent follows because $w = w_{i_{k+1}}(\bar{n},\bar{k},w'_k)$.

This concludes the proof of the case $k < n$. The proof of the case $k = n$ proceeds analogously. □

**Theorem 3.34.** *Let $G$ be a parametric grammar, $p(G)$ its existential program, $n \in \mathbb{N}$ and $w$ a term. Then $\mathbf{Q}_{\min} \models_{DL} \alpha = \bar{n} \to \langle P(G)\rangle\tau = w$ iff $w \in L(G_n)$.*

*Proof.* Assume $\alpha = \bar{n} \to \langle P(G)\rangle\tau = w$. By Lemma 3.32, $\alpha = \bar{n} \to [P(G)]\tau \in L(G_n)$. Together these imply $\alpha = \bar{n} \to \langle P(G)\rangle(\tau = w \wedge \tau \in L(G_n))$, whence follows $w \in L(G_n)$.

The other direction was proved in Lemma 3.33. □

The above translation from grammars to programs, together with the results of the previous section, suggests a joint relationship between simple induction proofs, parametric grammars, and **IMP** programs. However, there is something of a mismatch here: recall that the language of a parametric grammar consists of terms representing formulas. It follows that the program of a parametric grammar also computes a language of (terms representing) formulas. By contrast, the program we extract from a simple induction proof produces "actual" terms; the formulas in the sip only occur as pre-, post-, and loop conditions. We believe that this difference is merely technical, not essential; unifying the two notions of program is a topic for future research.

# CHAPTER 4

# THE AFFINE SOLUTION PROBLEM

Algorithms for computing affine invariants of affine programs have been given in [MOS04] and its precursor [Kar76]. When translated to formula equations, their methods provide solutions for formula equations which are conjunctions of *affine Horn clauses*. We significantly generalize this result to arbitrary quantifier-free formula equations. The gist is that an affine formula equation $\varphi$ can be clausified and the clauses translated into statements about affine spaces over $\mathbb{Q}$ called *affine conditions*; during this process, the unknown formulas in $\varphi$ (the predicate variables) become unknown affine subspaces of $\mathbb{Q}^k$ for some $k$. At this point, we use the fact that affine subspaces of $\mathbb{Q}^k$ have a certain disjunction property: if a subspace $\mathcal{Y}$ is covered by $\bigcup_{i=1}^{n} \mathcal{X}_i$, then it is already covered by some $\mathcal{X}_{i_0}$. This allows us to break the affine conditions of $\varphi$ apart into *affine Horn conditions*. If a solution to some set of affine Horn conditions exists, we find it with an iteration procedure that is guaranteed to terminate. In fact, the procedure returns the smallest subspaces satisfying all affine Horn conditions.

The results of this chapter have been published in [HZ19].

## 4.1 Affine formula equations

**Definition 4.1** ($\mathcal{L}_{\text{aff}}$)**.** Let $\mathcal{L}_{\text{aff}}$ be the first-order language consisting of

- the constant symbols 0 and 1;
- the binary function symbol $+$;
- countably many unary function symbols $\{\, c \mid c \in \mathbb{Q}\}$;
- the binary predicate symbol $=$.

In the sequel, let $\text{Th}(\mathbb{Q})$ be the theory of $\mathbb{Q}$ over $\mathcal{L}_{\text{aff}}$, where 0 and 1 are interpreted as the rational numbers 0 and 1, $+$ as actual addition, and each unary function symbol $c$ as the function $x \mapsto cx$. We will abbreviate the term $c1$ as $c$ and the term $-1t$ as $-t$.

**For the rest of this chapter, we will tacitly interpret terms and formulas over $\mathcal{L}_{\text{aff}}$ modulo $\text{Th}(\mathbb{Q})$.** Since $\text{Th}(\mathbb{Q})$ entails associativity and commutativity of addition and distributivity of scalar multiplication over addition, we can assume without loss of generality that every term $t(x_1, ..., x_n)$ is of the form $c_0 + \sum_{i=1}^{n} c_i x_i$ and every atomic formula $A(x_1, ..., x_n)$ is of the form $t(x_1, ..., x_n) = 0$. We call such atomic formulas *linear equations* and conjunctions of linear equations *linear equation systems*.

If $\varphi$ is a formula over $\mathcal{L}_{\text{aff}}$ containing exactly the individual variables $x_1, ..., x_n$, we may assume without loss of generality that each term in $\varphi$ actually contains all the individual variables $x_1, ..., x_n$. This is because $\text{Th}(\mathbb{Q}) \vDash \sum_{i=1}^{n} c_i x_i = \sum_{i=1}^{n} c_i x_i + 0y$, so we can always add variables to terms without affecting their meanings. The same holds for atomic formulas, for the same reason. Note that we do not make the same stipulation for formula variables, i.e., formula variables in $\varphi$ may contain any subset of individual variables of $\varphi$.

**Definition 4.2** (Affine formula equation)**.** An *affine formula equation* is a $\Pi_1$ $\mathcal{L}_{\text{aff}}$-formula equation $\Phi$.

Let $x_1, ..., x_n$ be the individual variables in $\Phi$ and $X_1, ..., X_m$ the formula variables in $\Phi$ with respective arities $k_1, ..., k_m$. Then the tuple $\langle n; k_1, ..., k_m \rangle$ is called the *dimensions* of $\Phi$.

We can now define the affine solution problem.

**Definition 4.3** (Affine solution problem)**.** The *affine solution problem* is the solution problem $\langle \mathrm{Th}(\mathbb{Q}), \mathcal{F}, \mathcal{C} \rangle$ where $\mathcal{C}$ is the class of linear equation systems and $\mathcal{F}$ is the class of affine formula equations.

In other words, the affine solution problem captures the following question: given a $\Pi_1$ formula equation $\exists \bar{X}\, \varphi$ over $\mathcal{L}_{\mathrm{aff}}$ with unknowns $X_1, ..., X_n$, are there linear equation systems $\psi_1, ..., \psi_n$ such that $\varphi[X_1 \backslash \psi_1, ..., X_n \backslash \psi_n]$ is valid modulo $\mathrm{Th}(\mathbb{Q})$?

The main result of this chapter is

**Theorem 4.4.** *The affine solution problem is decidable.*

In Section 4.3, we will develop the results necessary to prove Theorem 4.4. Throughout the rest of this chapter, we will illustrate the decision procedure with a running example.

**Example 4.5.** Let $\Phi \equiv \exists X.\ \varphi$ be the affine formula equation

$$\begin{aligned} \Phi &\equiv \exists X.\ \varphi \\ &\equiv \exists X.\ \forall x, y.\ \begin{pmatrix} X(1,0) \\ \wedge\, (X(x,y) \to X(-y,x) \vee X(x,-y)) \\ \wedge\, (X(x,y) \to x = y \vee y = 0) \end{pmatrix}. \end{aligned}$$

The dimensions of $\Phi$ are $\langle 2; 2 \rangle$.

## 4.2 Invariant generation in affine programs

In [MOS04] the authors consider a formalism for the verification of affine programs based on state transition diagrams: a program consists of points connected by directed edges, each of which is labelled with either an affine variable assignment or a subroutine call. There are no guards of any kind; loops continue iterating or terminate nondeterministically. Operations other than affine assignments are abstracted as *nondeterministic assignments* $x := ?$ that may set a variable to an arbitrary value. The authors present an algorithm that calculates, for each program point, the set of all linear equations that hold at that point whenever execution reaches it. This algorithm is based on iteratively computing (an abstract

representation of) the set of runs that reach each program point. This idea has no obvious analogue in formula equations, since there is no sense of "flow" from one location to the next. Instead, we take the position that an affine formula equation describes a relationship between certain affine spaces and their images and preimages under some affine transformations. It is then straightforward to iteratively calculate minimal affine spaces that can possibly satisfy this relationship. If they do, they are a (maximally precise) solution; if they do not, then no solution exists.

Let us present the transformation from the graph formalism of [MOS04] to formula equations in a bit more detail. Each procedure $p$ in the program has an entry point $e_p$ and a return point $r_p$. Moreover, there is a special procedure **Main** whose entry and return points serve as the entry and return points of the whole program. For each point $s$, let $X_s$ be a formula variable over all individual variables of the program. We give the resulting formula equation as a set of constrained Horn clauses:

- $\vdash X_{e_{\mathbf{Main}}}(\bar{x})$ asserts that at the start of the program, no nontrivial linear equations are valid;
- If $s \xrightarrow{x_j := t(\bar{x})} s'$ is an edge in the program, then we add the clause $X_s(\bar{x}) \vdash X_{s'}[x_j \backslash t(\bar{x})]$.
- If $s \xrightarrow{x_j := ?} s'$ is an edge in the program, then we add the clauses $X_s(\bar{x}) \vdash X_{s'}[x_j \backslash 0]$ and $X_s(\bar{x}) \vdash X_{s'}[x_j \backslash 1]$. The idea here is that if $(x_1, ..., 0, ..., x_n)$ and $(x_1, ..., 1, ..., x_n)$ both solve the linear equation system $X_{s'}$, then so does $(x_1, ..., y, ..., x_n)$ for any $y \in \mathbb{Q}$.
- If $s \xrightarrow{p} s'$ is an edge in the program and $p$ is a procedure, we add the clauses $X_s(\bar{x}) \vdash X_{e_p}(\bar{x})$ and $X_{r_p}(\bar{x}) \vdash X_{s'}(\bar{x})$.

These formula equations are even simpler than in the case of Hoare calculus: there is only at most one negative formula variable in each clause; constrained Horn clauses with this property are called *linear constrained Horn clauses*. Moreover, due to the absence of guards, there are no constant formulas in these clauses.

Note that [MOS04] is not concerned with whether a solution exists; in fact, the existence of a solution of $\varphi$ is not in question, since we can let $X_p \equiv \top$ for all program points $p$ of $P$. Rather, they compute a *strongest* solution for each unknown in $\varphi$. In this sense, our approach diverges from theirs. Moreover, it is

easily seen that in general, a formula equation need not have a unique strongest solution. It turns out, however, that our algorithm MinimalSolution produces a *maximally strong* solution. We will elaborate on this in Sections 4.3.4 and 4.4.

## 4.3 Deciding affine solution problems

This section is devoted to proving Theorem 4.4. We proceed as follows: Let $\Phi \equiv \exists \bar{X}\, \varphi$ be an affine formula equation. We first translate $\varphi$ into a set of clauses $\mathrm{Cl}(\varphi)$. Each clause $C$ induces a statement $\mathcal{C}$ that asserts the inclusion of an intersection of affine spaces in a union of affine spaces. The statements we obtain in this way from clauses in $\mathrm{Cl}(\varphi)$ form the set $\mathrm{AC}(\varphi)$ of affine conditions of $\varphi$. Notably, solutions of $\varphi$ and $\mathrm{AC}(\varphi)$ correspond to each other. We decompose the affine conditions into projections that are amenable to algorithmic solving, obtaining the set $\mathrm{Pr}(\varphi)$. Then we apply Algorithm 4.3.4, which iteratively computes a solution of $\mathrm{Pr}(\varphi)$ if a solution exists and reports failure otherwise. The resulting tuple of affine subspaces can be translated back into a tuple of linear equation systems.

### 4.3.1 Clausification

Let $\Phi \equiv \exists \bar{X}\, \varphi$ be an affine formula equation. By definition, $\varphi$ is $\Pi_1$, so it is of the form $\forall \bar{x}\, \varphi'$ with $\varphi'$ quantifier-free. It follows that $\varphi'$ is a Boolean combination of linear equations and formula variables. However, it is more convenient for us to regard linear equation systems as the atoms of $\varphi'$. Therefore, we use the word "clause" to refer to a disjunction of possibly negated linear equation systems (and formula variables, which range over linear equation systems).

As per Definition 1.11, we thus obtain the set $\mathrm{Cl}(\varphi)$ containing clauses of the form

$$\begin{gathered} A(\bar{x}), X_1(\bar{s}_1(\bar{x})), ..., X_m(\bar{s}_m(\bar{x})) \\ \vdash \\ B_1(\bar{x}), ..., B_r(\bar{x}), Y_1(\bar{t}_1(\bar{x})), ..., Y_n(\bar{t}_n(\bar{x})), \end{gathered}$$

where $s_i, t_i$ are affine terms; $A, B_i$ are linear equation systems, and $X_i, Y_i$ are formula variables ranging over linear equation systems. As per the usual convention,

such a clause is implicitly universally closed over its individual variables. Note that for $i \neq j$, $X_i$ and $X_j$ are not necessarily distinct formula variables; for instance, a clause might contain the same formula variable multiple times with different terms. Note that there is only a single constant literal on the left-hand side. This is the case because a conjunction of linear equation systems is just a single linear equation system.

In principle, any kind of clause form transformation that preserves satisfiability will serve here if we are only interested in the existence of solutions, but for transformations that do not preserve logical equivalence, it may be unclear how the solutions of the original formula equation and the clause form relate to each other. Thus, for the sake of simplicity, we choose the naive method of computing a clause form by distributivity.

**Example 4.6.** Let $\Phi \equiv \exists \bar{X}.\ \varphi$ be the affine formula equation defined in Example 4.5. As it happens, $\varphi$ is already in clause form; writing the clauses of $\varphi$ as sequents yields

$$\begin{aligned} &\vdash X(1,0) \\ X(-x,y) &\vdash X(-y,x), X(x,-y) \\ X(x,y) &\vdash x = y, y = 0. \end{aligned}$$

### 4.3.2 Affine geometry

By first-order semantics, linear equation systems map directly to affine subspaces of $\mathbb{Q}^n$ for some $n$. Having decomposed an affine formula equation into clauses, in this section we will reinterpret these clauses as descriptions of relationships between certain affine spaces and affine transformations. In other words, we will move from syntax to semantics and consider the problem from the perspective of linear algebra. We will need some elementary results of affine geometry. Let us start by reviewing some important concepts. For a more rigorous treatment, see [ST71].

**We will exclusively deal with finite-dimensional vector spaces over $\mathbb{Q}$. Consequently, we may assume that all vector spaces we consider are of the form $\mathbb{Q}^n$ for $n \in \mathbb{N}$.**

Let us review the definition of linear and affine combinations. A *linear combination* of vectors $a_1, ..., a_n \in \mathbb{Q}^m$ is a sum $\sum_{i=1}^n c_i a_i$ with $c_i \in \mathbb{Q}$. An *affine combination* is a linear combination with the additional condition that $\sum_{i=1}^n c_i = 1$. The set of all linear combinations of elements of $M \subseteq \mathbb{Q}^m$ is called the *linear hull* of $M$ and written as $[M]$.

An *affine subspace* $\mathcal{A}$ of $\mathbb{Q}^n$ is a subset of $\mathbb{Q}^n$ that can be written as $a + \vec{\mathcal{A}}$ for a vector $a \in \mathbb{Q}^n$ and a linear subspace $\vec{\mathcal{A}} \subseteq \mathbb{Q}^n$. The *dimension* of $\mathcal{A}$ is the dimension of $\vec{\mathcal{A}}$.

A *coordinate system* $\langle a; B\rangle$ of an affine subspace $\mathcal{A}$ consists of an element $a$ of $\mathcal{A}$ and a basis $B$ of $\vec{\mathcal{A}}$. For convenience, we also say that $\emptyset$ is a coordinate system of itself.

**Definition 4.7** (Aff)**.** $\operatorname{Aff}\mathbb{Q}^n$ is the set containing the empty set and all affine subspaces of $\mathbb{Q}^n$.

**Proposition 4.8.** *Let $n \in \mathbb{N}$.*

1. $\operatorname{Aff}\mathbb{Q}^n$ *contains precisely those subsets of* $\mathbb{Q}^n$ *that are closed under affine combinations.*
2. $\operatorname{Aff}\mathbb{Q}^n$ *is a complete lattice.*
3. $\operatorname{Aff}\mathbb{Q}^n$ *satisfies ACC.*

*Proof.*

1. The empty set is trivially closed under affine combinations. For an affine subspace $\mathcal{A} = a + \vec{\mathcal{A}}$ of $\mathbb{Q}^n$, let $b, c \in \mathcal{A}$ and $t \in [0,1] \cap \mathbb{Q}$. Then $b = a + b'$ and $c = a + c'$ with $b', c' \in \vec{\mathcal{A}}$. It follows that $tb + (1-t)c = a + \underbrace{tb' + (1-t)c'}_{\in \vec{\mathcal{A}}} \in \mathcal{A}$, so $\mathcal{A}$ is closed under affine combinations.
   Now let $\mathcal{A} \subseteq \mathbb{Q}^n$ be nonempty and closed under affine combinations. We need to show that $\mathcal{A}$ can be written as $a + A$ for some $a \in \mathcal{A}$ and some linear subspace $A$ of $\mathbb{Q}^n$. Let $a \in \mathcal{A}$ be arbitrary and $A = \{\, x - a \mid x \in \mathcal{A}\}$. Clearly, $\vec{0} \in A$. If $b, c \in A$ and $\lambda \in \mathbb{Q}$, then $b = b' - a, c = c' - a$ with $b', c' \in \mathcal{A}$ and $a + b + \lambda c = -\lambda a + 1b' - \lambda c' \in \mathcal{A}$ because $\mathcal{A}$ is closed under affine combinations. Consequently, $b + \lambda c \in A$, so $A$ is closed under linear combinations. This shows that $\mathcal{A}$ is an affine subspace of $\mathbb{Q}^n$.

2. It is sufficient to show that $\operatorname{Aff}\mathbb{Q}^n$ is closed under arbitrary intersections. $\bigcap\emptyset = \mathbb{Q}^n$, so let $S \subseteq \operatorname{Aff}\mathbb{Q}^n$ be nonempty. By 1, every element of $S$ is closed under affine combinations and hence $\bigcap S$ is as well. It follows that $\bigcap S \in \operatorname{Aff}\mathbb{Q}^n$.
3. The argument is analogous to the one in Example 1.23. □

An *affine transformation* $\mathcal{T}$ from $\mathbb{Q}^m$ to $\mathbb{Q}^n$ is a map that can be written as a linear map followed by a translation, i.e., $\mathcal{T}$ is affine if there is a matrix $A \in \mathbb{Q}^{n\times m}$ and a vector $b \in \mathbb{Q}^n$ such that $\mathcal{T} : x \mapsto Ax + b$. The images and preimages of affine subspaces under affine transformations are themselves affine subspaces or else empty.

The following proposition shows the connection between formulas and terms of $\mathcal{L}_{\text{aff}}$ and affine spaces and transformations, respectively.

**Proposition 4.9.**

1. *Let $m, n \in \mathbb{N}$, $\mathcal{T} : \mathbb{Q}^m \to \mathbb{Q}^n$ affine, $\mathcal{B} \in \operatorname{Aff}\mathbb{Q}^m$, and $\mathcal{C} \in \operatorname{Aff}\mathbb{Q}^n$. Then $\mathcal{T}(\mathcal{B}) \in \operatorname{Aff}\mathbb{Q}^n$ and $\mathcal{T}^{-1}(\mathcal{C}) \in \operatorname{Aff}\mathbb{Q}^m$.*
2. *Let $\mathcal{A} \subseteq \mathbb{Q}^n$. Then $\mathcal{A} \in \operatorname{Aff}\mathbb{Q}^n$ iff there is a linear equation system $A(x_1, ..., x_n)$ such that $\mathcal{A} = A^{\mathbb{Q}}$.*
3. *Let $\mathcal{T} : \mathbb{Q}^m \to \mathbb{Q}^n$. Then $\mathcal{T}$ is affine iff there are $\mathcal{L}_{\text{aff}}$-terms $t_1(\bar{x}), ..., t_n(\bar{x})$ such that $\mathcal{T} = (t_1, ..., t_n)^{\mathbb{Q}}$.*

*Proof.* For 1., it is immediately clear that the image and preimage of an affine subspace (or the empty set) under a translation are again an affine subspace (or empty). Since $\mathcal{T}$ is the composition of a linear transformation and a translation, we can assume without loss of generality that $\mathcal{T}$ is linear. Let $a, b \in \mathcal{T}(\mathcal{B})$ and $t \in [0, 1] \cap \mathbb{Q}$. Then $a = \mathcal{T}(a')$ and $b = \mathcal{T}(b')$ for some $a', b' \in \mathcal{B}$. It follows that

$$ta + (1-t)b = t(\mathcal{T}(a')) + (1-t)(\mathcal{T}(b')) = \mathcal{T}(ta' + (1-t)b') \in \mathcal{T}(\mathcal{B}).$$

Now let $a, b \in \mathcal{T}^{-1}(\mathcal{C})$ and $t \in [0, 1] \cap \mathbb{Q}$ Then we have

$$\mathcal{T}(ta + (1-t)b) = t\mathcal{T}(a) + (1-t)\mathcal{T}(b) \in \mathcal{C},$$

so $t(a) + (1-t)b \in \mathcal{T}^{-1}(\mathcal{C})$.

For 2., let $\mathcal{A} = A^{\mathbb{Q}}$ for a linear equation system $A(x_1, ..., x_n)$. We may assume that $A$ consists of a single equation $\sum_{i=1}^{n} c_i x_i + c_0 = 0$; the case of multiple equations follows immediately. Let $a, b \in \mathcal{A}$ and $t \in [0,1] \cap \mathbb{Q}$. Then

$$\mathbb{Q} \vDash \sum_{i=1}^{n} c_i(ta + (1-t)b) + c_0 = t(\sum_{i=1}^{n} c_i a_i + c_0) + (1-t)(\sum_{i=1}^{n} c_i b_i + c_0) = 0.$$

Conversely, let $\mathcal{A} \in \operatorname{Aff} \mathbb{Q}^n$. If $\mathcal{A} = \emptyset$, then $\mathcal{A} = \bot^{\mathbb{Q}}$. Otherwise, $\mathcal{A}$ is an affine subspace of $\mathbb{Q}^n$, so $\mathcal{A} = a + \vec{\mathcal{A}}$ for a linear subspace $\vec{\mathcal{A}}$. Let $b_1, ..., b_k$ be a basis of $\vec{\mathcal{A}}$. We obtain the desired linear equation system by solving

$$\begin{pmatrix} 1 & b_0^T \\ 0 & b_1^T \\ \vdots & \vdots \\ 0 & b_k^T \end{pmatrix} \begin{pmatrix} c_0 \\ \vdots \\ c_n \end{pmatrix} = \bar{0}.$$

for $\bar{c}$.

For 3., let $\mathcal{T} = \bar{t}^{\mathbb{Q}}$ for $\mathcal{L}_{\text{aff}}$-terms $t_1(\bar{x}), ..., t_n(\bar{x})$, where $t_j(\bar{x}) = \sum_{i=1}^{m} a_{i,j} x_j + b_i$. This implies

$$\mathcal{T}(\bar{x}) = \begin{pmatrix} \sum_{i=1}^{m} a_{1,i} x_i + b_1 \\ \vdots \\ \sum_{i=1}^{m} a_{n,i} x_i + b_n \end{pmatrix} = (a_{i,j})\bar{x} + \begin{pmatrix} b_1 \\ \vdots \\ b_n \end{pmatrix},$$

so $\mathcal{T}$ is affine.

On the other hand, if $\mathcal{T}$ is affine, it can be written as

$$\mathcal{T}(\bar{x}) = A\bar{x} + b.$$

From this we obtain terms $t_j(\bar{x}) = \sum_{i=1}^{m} a_{i,j} x_i + b_i$ for $j = 1, ..., n$ such that $\mathcal{T} = \bar{t}^{\mathbb{Q}}$. □

We can now perform the switch from syntax to semantics. To this end, we define a certain form of linear algebraic statement that corresponds to a logical clause in the language $\mathcal{L}_{\text{aff}}$. These are the affine conditions.

**Definition 4.10** (Affine condition)**.** Let $m, n, \ell, r \in \mathbb{N}$. An *affine condition* $\mathcal{C}$ is a statement of the form

$$\mathcal{A} \cap \bigcap_{i=1}^{\ell} \mathcal{T}_i^{-1}(\mathcal{X}_i) \subseteq \bigcup_{i=1}^{m} \mathcal{B}_i \cup \bigcup_{j=\ell+1}^{\ell+r} \mathcal{T}_j^{-1}(\mathcal{X}_j),$$

where $\mathcal{A}, \mathcal{B}_1, ..., \mathcal{B}_m \in \operatorname{Aff}\mathbb{Q}^n$ and for $i = 1, ..., \ell + r$, $\mathcal{T}_i \colon \mathbb{Q}^n \to \mathbb{Q}^{k_i}$ is an affine transformation and $\mathcal{X}_i$ is a variable ranging over $\operatorname{Aff}\mathbb{Q}^{k_i}$. The tuple $\langle n; k_1, ..., k_{\ell+r}\rangle$ is called the *dimensions* of $\mathcal{C}$.

A tuple $\bar{\mathcal{F}} \in \operatorname{Aff}\mathbb{Q}^{k_1} \times ... \times \operatorname{Aff}\mathbb{Q}^{k_{l+r}}$ is a *solution* of $\mathcal{C}$ if $\mathcal{C}[\bar{\mathcal{X}} \backslash \bar{\mathcal{F}}]$ is true.

We obtain the affine condition corresponding to a clause, and thence the affine conditions corresponding to an affine formula equation, by semantic interpretation of terms and linear equation systems.

**Definition 4.11** (Affine conditions of an affine formula equation)**.** Let $\Phi \equiv \exists \bar{X}\, \varphi$ be an affine formula equation and

$$\begin{gathered} C \equiv A(\bar{x}), X_1(\bar{t}_1(\bar{x})), ..., X_\ell(\bar{t}_\ell(\bar{x})) \\ \vdash \\ B_1(\bar{x}), ..., B_m(\bar{x}), X_{\ell+1}(\bar{t}_{\ell+1}(\bar{x})), ..., X_{\ell+r}(\bar{t}_{\ell+r}(\bar{x})), \end{gathered}$$

a clause with dimensions $\langle n; k_1, ..., k_{\ell+r}\rangle$ in $\mathrm{Cl}(\varphi)$. Then we call

$$\mathcal{C} :\equiv A^{\mathbb{Q}} \cap \bigcap_{i=1}^{\ell} (\bar{t}_i^{\mathbb{Q}})^{-1}(\mathcal{X}_i) \subseteq \bigcup_{i=1}^{m} B_i^{\mathbb{Q}} \cup \bigcup_{j=\ell+1}^{\ell+r} (\bar{t}_j^{\mathbb{Q}})^{-1}(\mathcal{X}_j)$$

the *affine condition* corresponding to $C$. The dimensions of $\mathcal{C}$ are equal to those of $C$.

We say that $\mathcal{C}$ is an affine condition of $\varphi$ iff it corresponds to some clause in $\mathrm{Cl}(\varphi)$. We write $\mathrm{AC}(\varphi)$ for the set of affine conditions of $\varphi$.

**Example 4.12.** Consider the affine formula equation $\Phi \equiv \exists \bar{X}\, \varphi$ from Example 4.6. We obtain the affine conditions of $\varphi$ by translating each of the clauses:

$$\begin{aligned} \mathbb{Q}^2 &\subseteq \mathcal{T}_1^{-1}(\mathcal{X}), \\ \mathcal{S}^{-1}(\mathcal{X}) &\subseteq \mathcal{T}_2^{-1}(\mathcal{X}) \cup \mathcal{T}_3^{-1}(\mathcal{X}), \\ \mathcal{X} &\subseteq \mathcal{B}_1 \cup \mathcal{B}_2, \end{aligned}$$

where $\mathcal{X}$ is a variable ranging over $\operatorname{Aff}\mathbb{Q}^2$ and

$$\mathcal{T}_1\colon \begin{pmatrix} x \\ y \end{pmatrix} \mapsto \begin{pmatrix} 1 \\ 0 \end{pmatrix} \qquad \mathcal{B}_1 = \begin{pmatrix} 0 \\ 0 \end{pmatrix} + \left[ \begin{pmatrix} 1 \\ 1 \end{pmatrix} \right]$$
$$\mathcal{T}_2\colon \begin{pmatrix} x \\ y \end{pmatrix} \mapsto \begin{pmatrix} -y \\ x \end{pmatrix} \qquad \mathcal{B}_2 = \begin{pmatrix} 0 \\ 0 \end{pmatrix} + \left[ \begin{pmatrix} 1 \\ 0 \end{pmatrix} \right]$$
$$\mathcal{T}_3\colon \begin{pmatrix} x \\ y \end{pmatrix} \mapsto \begin{pmatrix} x \\ -y \end{pmatrix} \qquad \mathcal{S}\colon \begin{pmatrix} x \\ y \end{pmatrix} \mapsto \begin{pmatrix} -x \\ y \end{pmatrix}.$$

$\mathcal{B}_1$ and $\mathcal{B}_2$ are straightforwardly obtained as solution spaces of the linear equations $x = y$ and $y = 0$, respectively.

The following theorem shows that the translation from clauses to affine conditions is fit for our purpose because solutions of one translate to solutions of the other.

**Theorem 4.13.** *Let $\exists\bar{X}\,\varphi$ be an affine formula equation, $C \in \mathrm{Cl}(\varphi)$, $\mathcal{C}$ the corresponding affine condition, and $\bar{F}$ a tuple of linear equation systems of appropriate arities. Then $\mathbb{Q} \models C[\bar{X}\backslash\bar{F}]$ iff $\bar{F}^{\mathbb{Q}}$ is a solution of $\mathcal{C}$.*

*Proof.* Let $W, Z$ be such that $C[\bar{X}\backslash\bar{F}]$ is syntactically equal to $W \vdash Z$ and $\mathcal{W}, \mathcal{Z}$ such that $\mathcal{C}[\bar{\mathcal{X}}\backslash\bar{F}^{\mathbb{Q}}]$ is syntactically equal to $\mathcal{W} \subseteq \mathcal{Z}$.

Then $\mathcal{W} = W^{\mathbb{Q}}$ and $\mathcal{Z} = Z^{\mathbb{Q}}$ by Proposition 1.1 1–3. Moreover, by Proposition 1.1 4, $\mathcal{W} \subseteq \mathcal{Z}$ iff $\mathbb{Q} \models W \to Z$, from which the claim follows immediately. □

**Corollary 4.14.** *Let $\Phi \equiv \exists\bar{X}\,\varphi$ be an affine formula equation. Then $\Phi$ has a solution iff $\mathrm{AC}(\varphi)$ does.*

*Proof.* If $\Phi$ has a solution $\bar{F}$, then $\bar{F}^{\mathbb{Q}}$ is a solution of $\mathrm{AC}(\varphi)$ by Theorem 4.13. Conversely, if $\mathrm{AC}(\varphi)$ has a solution $\mathcal{F}_1, ..., \mathcal{F}_n$, then by Proposition 4.9, there are linear equation systems $F_1, ..., F_n$ such that $F_i^{\mathbb{Q}} = \mathcal{F}_i$. $\bar{F}$ is then a solution of $\Phi$ (again by Theorem 4.13). □

Corollary 4.14 shows that to solving an affine formula equation $\Phi$—an object of predicate logic—reduces to solving the set of affine conditions of $\Phi$, which are objects of linear algebra. Thus, we will leave logic behind for the moment and consider the problem from the perspective of linear algebra.

### 4.3.3 Projections

In the previous section, we transformed an affine formula equation into a set of affine conditions, which are statements about affine subspaces of $\mathbb{Q}^n$. When developing our solution algorithm, we want to work with a more restricted form of affine conditions that only allows a single set on the right-hand side instead of a union. We will see in Lemma 4.18 and Corollary 4.19 why restricting our attention to these simpler conditions does not result in a loss of generality.

**Definition 4.15** (Affine Horn condition)**.** We call affine conditions of the form

$$\mathcal{A} \cap \bigcap_{i=1}^{\ell} \mathcal{S}_i^{-1}(\mathcal{X}_i) \subseteq \mathcal{B} \quad \text{or} \quad \mathcal{A} \cap \bigcap_{i=1}^{\ell} \mathcal{S}_i^{-1}(\mathcal{X}_i) \subseteq \mathcal{S}^{-1}(\mathcal{Y})$$

*affine Horn conditions*.

**Remark.** The name "affine Horn condition" is due to the similarity in form between these conditions and *Horn clauses*, which are clauses containing at most one positive literal. Written in sequent form, this means that a Horn clause has the form $A_1, ..., A_n \vdash B$ for atomic formulas $A_1, ..., A_n, B$.

**Definition 4.16** (Projections)**.**

1. Let

$$\mathcal{C} : \mathcal{A} \cap \bigcap_{i=1}^{\ell} \mathcal{T}_i^{-1}(\mathcal{X}_i) \subseteq \bigcup_{i=1}^{m} \mathcal{B}_i \cup \bigcup_{j=\ell+1}^{\ell+r} \mathcal{T}_j^{-1}(\mathcal{X}_j)$$

   be an affine condition. We call the affine Horn conditions

$$\mathcal{A} \cap \bigcap_{i=1}^{\ell} \mathcal{T}_i^{-1}(\mathcal{X}_i) \subseteq \mathcal{B}_{i_0}, i_0 \in \{1, ..., m\},$$
$$\mathcal{A} \cap \bigcap_{i=1}^{\ell} \mathcal{T}_i^{-1}(\mathcal{X}_i) \subseteq \mathcal{T}_{j_0}^{-1}(\mathcal{X}_{j_0}), j_0 \in \{\ell+1, ..., \ell+r\}$$

   *projections* of $\mathcal{C}$. The latter form can equivalently be written as

$$\mathcal{T}_{j_0}(\mathcal{A} \cap \bigcap_{i=1}^{\ell} \mathcal{T}_i^{-1}(\mathcal{X}_i)) \subseteq \mathcal{X}_{j_0}.$$

2. Let $S$ be a set of affine conditions. We call a set of affine Horn conditions a projection of $S$ if it consists of exactly one projection of each element of $S$. We write $\Pr(S)$ for the set of projections of $S$.

3. Let $\exists \bar{X}\, \varphi$ be an affine formula equation. We abbreviate $\mathrm{Pr}(\mathrm{AC}(\varphi))$ as $\mathrm{Pr}(\varphi)$ and call its elements *affine projections* of $\varphi$.

**Example 4.17.** Let $\mathrm{AC}(\varphi)$ be the affine conditions of $\varphi$ from Example 4.12. The first affine condition is already an affine Horn condition and hence has only itself as a projection. Since the latter two affine conditions each have 2 projections, there are four affine projections of $\varphi$:

$$\begin{aligned} \mathcal{T}_1(\mathbb{Q}^2) &\subseteq \mathcal{X} \\ \mathcal{T}_2(\mathcal{S}^{-1}(\mathcal{X})) &\subseteq \mathcal{X} \\ \mathcal{X} &\subseteq \mathcal{B}_1 \end{aligned} \qquad \text{(P1)} \qquad \begin{aligned} \mathcal{T}_1(\mathbb{Q}^2) &\subseteq \mathcal{X} \\ \mathcal{T}_2(\mathcal{S}^{-1}(\mathcal{X})) &\subseteq \mathcal{X} \\ \mathcal{X} &\subseteq \mathcal{B}_2 \end{aligned} \qquad \text{(P2)}$$

$$\begin{aligned} \mathcal{T}_1(\mathbb{Q}^2) &\subseteq \mathcal{X} \\ \mathcal{T}_3(\mathcal{S}^{-1}(\mathcal{X})) &\subseteq \mathcal{X} \\ \mathcal{X} &\subseteq \mathcal{B}_1 \end{aligned} \qquad \text{(P3)} \qquad \begin{aligned} \mathcal{T}_1(\mathbb{Q}^2) &\subseteq \mathcal{X} \\ \mathcal{T}_3(\mathcal{S}^{-1}(\mathcal{X})) &\subseteq \mathcal{X} \\ \mathcal{X} &\subseteq \mathcal{B}_2 \end{aligned} \qquad \text{(P4)}$$

The motivation behind projections is the following. A single affine condition $\mathcal{C}$, considered abstractly, has the form $\bigcap_{i\in I} \mathcal{M}_i \subseteq \bigcup_{j\in J} \mathcal{N}_j$, where the $\mathcal{M}_i, \mathcal{N}_j$ are elements of $\mathrm{Aff}\,\mathbb{Q}^n$. Suppose we replace $\mathcal{C}$ with the disjunction over its projections $\mathcal{C}' \equiv \bigvee_{j\in J} \left(\bigcap_{i\in I} \mathcal{M}_i \subseteq \mathcal{N}_j\right)$. If the $\mathcal{M}_i, \mathcal{N}_j$ were general sets, $\mathcal{C}'$ would be strictly stronger than $\mathcal{C}$, since $\mathcal{C}'$ asserts that $\mathcal{M} = \bigcap_{i\in I} \mathcal{M}_i$ is contained in one of the $\mathcal{N}_j$, whereas $\mathcal{C}$ only asserts that $\mathcal{M}$ is contained in the union of the $\mathcal{N}_j$. However, a family $(\mathcal{A}_i)_{i\in I}$ of affine subspaces of $\mathbb{Q}^n$ cannot cover another subspace $\mathcal{B}$ unless $\mathcal{B}$ is already contained in one of the $\mathcal{A}_i$. Consequently, $\mathcal{C}$ and $\mathcal{C}'$ are actually equivalent and we can restrict our attention to affine Horn conditions. We will spend the rest of this subsection making this reasoning precise.

**Lemma 4.18.** *Let $\mathcal{A}_1, ..., \mathcal{A}_n$ be proper affine subspaces of $\mathbb{Q}^m$. Then $\bigcup_{i=1}^n \mathcal{A}_i \subsetneq \mathbb{Q}^m$.*

*Proof.* By induction on $n$. The case of $n = 1$ is immediate.

Now assume the lemma holds for $n$ and let $\mathcal{A}_1, ..., \mathcal{A}_{n+1}$ be proper subspaces of $\mathbb{Q}^m$. We may assume without loss of generality that $\mathcal{A}_i \nsubseteq \bigcup_{\substack{j=1 \\ j\neq i}}^{n+1} \mathcal{A}_j$ for $i = 1, ..., n+1$, because otherwise this case reduces to that of $n$ subspaces and we are done. It

follows that there are vectors

$$a \in \mathcal{A}_1 \setminus \bigcup_{i=2}^{n+1} \mathcal{A}_i,$$
$$b \in \mathcal{A}_{n+1} \setminus \bigcup_{i=1}^{n} \mathcal{A}_i.$$

Consider the family of vectors $\{\, c(\lambda) := a + \lambda(b-a) \mid \lambda \in \mathbb{Q}\}$.

- Assume $c(\lambda) \in \mathcal{A}_1$ for some $\lambda \neq 0$. Since $a \in \mathcal{A}_1$, it follows that $b = (1 - \frac{1}{\lambda})a + \frac{1}{\lambda}c(\lambda) \in \mathcal{A}_1$, which is a contradiction. Therefore, $c(\lambda) \in \mathcal{A}_1$ only for $\lambda = 0$.
- $c(\lambda) \in \mathcal{A}_{n+1}$ only for $\lambda = 1$. The argument is analogous to the previous case.
- For any $2 \leq i \leq n$, assume $c(\lambda), c(\mu) \in \mathcal{A}_i$ and $\lambda \neq \mu$. Let $\nu = \frac{\lambda}{\lambda - \mu}$. Then $a = (1-\nu)c(\lambda) + \nu c(\mu) \in \mathcal{A}_i$, but $a \notin \mathcal{A}_i$ by assumption. Therefore, there is at most one $\lambda \in \mathbb{Q}$ such that $c(\lambda) \in \mathcal{A}_i$.

This means that for each $i = 1, ..., n+1$, there is at most one $\lambda$ such that $c(\lambda) \in \mathcal{A}_i$. It is therefore possible to choose $\lambda_0 \in \mathbb{Q}$ such that $c(\lambda_0)$ is in none of the $\mathcal{A}_i$. From this, we obtain $\bigcup_{i=1}^{n+1} \mathcal{A}_i \subsetneq \mathbb{Q}^m$. □

**Corollary 4.19.** *Let $\mathcal{A}_1, ..., \mathcal{A}_n, \mathcal{B}$ be affine subspaces of $\mathbb{Q}^m$. If $\mathcal{B} \subseteq \bigcup_{i=1}^{n} \mathcal{A}_i$, then $\mathcal{B} \subseteq \mathcal{A}_{i_0}$ for some $i_0$.*

*Proof.* By contraposition. Assume that $\mathcal{B} \not\subseteq \mathcal{A}_i$ for all $i$. This means that all of the sets $\mathcal{A}_i \cap \mathcal{B}$ are either empty or proper affine subspaces of $\mathcal{B}$. By applying Lemma 4.18 to $\mathcal{B}$ and the $\mathcal{A}_i \cap \mathcal{B}$ (ignoring the empty sets), we obtain $\bigcup_{i=1}^{n} (\mathcal{A}_i \cap \mathcal{B}) \subsetneq \mathcal{B}$. It follows that $\mathcal{B} \not\subseteq \bigcup_{i=1}^{n} \mathcal{A}_i$. □

**Theorem 4.20.** *Let $\mathcal{C}$ be an affine condition with dimensions $\langle n; k_1, ..., k_{\ell+r} \rangle$ and $\bar{\mathcal{F}} \in \operatorname{Aff}\mathbb{Q}^{k_1} \times ... \times \operatorname{Aff}\mathbb{Q}^{k_{\ell+r}}$. Then $\bar{\mathcal{F}}$ is a solution of $\mathcal{C}$ iff it is a solution of some projection of $\mathcal{C}$.*

*Proof.* Let $\mathcal{W}, \mathcal{Z}_1, ..., \mathcal{Z}_m$ be such that $\mathcal{C}[\bar{\mathcal{X}} \backslash \bar{\mathcal{F}}]$ is syntactically equal to $\mathcal{W} \subseteq \bigcup_{i=1}^{m} \mathcal{Z}_i$. If $\mathcal{F}$ satisfies $\mathcal{C}$, then $\mathcal{W} \subseteq \bigcup_{i=1}^{m} \mathcal{Z}_i$. By Corollary 4.19, this implies that $\mathcal{W} \subseteq \mathcal{Z}_{i_0}$ for some $i_0$, hence $\bar{\mathcal{F}}$ satisfies a projection of $\mathcal{C}$. The other direction is obvious. □

**Corollary 4.21.** *Let $S$ be a set of affine conditions with dimensions $\langle n; k_1, ..., k_{\ell+r}\rangle$ and $\bar{\mathcal{F}} \in \operatorname{Aff}\mathbb{Q}^{k_1} \times ... \times \operatorname{Aff}\mathbb{Q}^{k_{\ell+r}}$. Then $\bar{\mathcal{F}}$ is a solution of $S$ iff it is a solution of some element of* $\Pr(S)$.

Theorem 4.20 shows that the solvability of affine formula equations reduces to the solvability of sets of affine Horn conditions. In the next subsections, we will show that the latter question is decidable.

### 4.3.4 Finding a fixed point by iteration

In this section, we will present an iterative procedure for solving sets of affine Horn conditions. To this end, we will interpret these affine conditions in two ways according to their form: as instructions for constructing affine subspaces from the bottom up or as checks whether the constructed spaces have grown too large.

**Definition 4.22** (Upper and lower bound conditions)**.** Let $\psi$ be a set of affine Horn conditions. As per Definition 4.15, $\psi$ contains elements of the forms

$$\mathcal{A} \cap \bigcap_{i=1}^{m} \mathcal{S}_i^{-1}(\mathcal{X}_i) \subseteq \mathcal{B}$$

and

$$\mathcal{A} \cap \bigcap_{i=1}^{m} \mathcal{S}_i^{-1}(\mathcal{X}_i) \subseteq \mathcal{T}^{-1}(\mathcal{Y})$$
$$\Leftrightarrow \mathcal{T}(\mathcal{A} \cap \bigcap_{i=1}^{m} \mathcal{S}_i^{-1}(\mathcal{X}_i)) \subseteq \mathcal{Y},$$

where $\mathcal{A}, \mathcal{B}$ are affine subspaces, $\mathcal{S}_i, \mathcal{T}$ are affine transformations, and $\mathcal{X}_i, \mathcal{Y}$ are variables ranging over affine spaces. We call the former kind of affine Horn condition *upper bound* conditions and the latter *lower bound* conditions of $\psi$.

Let $\psi$ be a set of affine Horn conditions with dimensions $\langle k_0; k_1, ..., k_n\rangle$. Recall that this means that the unknown $\mathcal{X}_i$ ranges over $\operatorname{Aff}\mathbb{Q}^{k_i}$. It follows that the candidate solutions of $\psi$ are elements of $\operatorname{Aff}\mathbb{Q}^{k_1} \times ... \times \operatorname{Aff}\mathbb{Q}^{k_n}$.

**Definition 4.23** (CS($\psi$))**.** Let $\psi$ be a set of affine Horn conditions with dimensions $\langle k_0; k_1, ..., k_n\rangle$. We write CS($\psi$) for the lattice $\operatorname{Aff}\mathbb{Q}^{k_1} \times ... \times \operatorname{Aff}\mathbb{Q}^{k_n}$ of candidate solutions of $\psi$.

The order on $\mathrm{CS}(\psi)$ is defined by

$$(\mathcal{U}_1, ..., \mathcal{U}_n) \leq (\mathcal{V}_1, ..., \mathcal{V}_n) \text{ if } \mathcal{U}_1 \subseteq \mathcal{V}_1 \wedge ... \wedge \mathcal{U}_n \subseteq \mathcal{V}_n.$$

Its least element is $0 := (\emptyset, ..., \emptyset)$. As the finite product of complete lattices of finite height, $\mathrm{CS}(\psi)$ is complete and has finite height as well.

Let $\mathcal{C}$ be the lower bound condition $\mathcal{T}(\mathcal{A} \cap \bigcap_{i=1}^{m} \mathcal{S}_i^{-1}(\mathcal{X}_i)) \subseteq \mathcal{Y}$. Also, let $S = (\mathcal{F}_1, ..., \mathcal{F}_m, \mathcal{G}) \in \mathrm{CS}(\mathcal{C})$ be a candidate solution of $\mathcal{C}$. $S$ solves $\mathcal{C}$ if $\mathcal{C}[\bar{\mathcal{X}} \backslash \bar{\mathcal{F}}, \mathcal{Y} \backslash \mathcal{G}]$ is true, i.e., if $\mathcal{G}$ contains the set $\mathcal{F} = \mathcal{T}(\mathcal{A} \cap \bigcap_{i=1}^{m} \mathcal{S}_i^{-1}(\mathcal{F}_i))$. Now consider the set $\mathcal{G}' = \mathcal{G} \vee \mathcal{F}$. By definition, it is the least element of $\mathrm{Aff}\,\mathbb{Q}^n$ containing both $\mathcal{G}$ and $\mathcal{F}$. If $S$ is a solution of $\mathcal{C}$, then $\mathcal{G}' = \mathcal{G}$, otherwise $\mathcal{G}' > \mathcal{G}$. In either case, $S' = (\mathcal{F}_1, ..., \mathcal{F}_m, \mathcal{G}')$ solves $\mathcal{C}$ and is in fact the least solution of $\mathcal{C}$ greater or equal to $S$. This shows that we can always increase a non-solution of a lower bound condition to a solution by enlarging the space on the right-hand side and that we do not skip over any solutions in this process. The following definition formalizes this intuition and generalizes it to sets of lower bound conditions. As a consequence, we will not generally reach a solution in just one step, but will need to iterate the process.

**Definition 4.24** ($\Phi$)**.** Let $\psi$ be a set of affine Horn conditions. Let $\mathcal{X}_1, ..., \mathcal{X}_n$ be the variables in $\psi$ and $k_1, ..., k_n$ their respective dimensions. We define an operator $\Phi_\psi \colon \mathrm{CS}(\psi) \to \mathrm{CS}(\psi)$ in the following manner: Let $1 \leq \alpha \leq n$ and

$$\mathcal{T}_1 \left( \mathcal{A}_1 \cap \bigcap_{i=1}^{m_1} \mathcal{S}_{1,i}^{-1}(\mathcal{X}_{j_{1,i}}) \right) \subseteq \mathcal{X}_\alpha$$
$$\vdots$$
$$\mathcal{T}_\ell \left( \mathcal{A}_\ell \cap \bigcap_{i=1}^{m_\ell} \mathcal{S}_{\ell,i}^{-1}(\mathcal{X}_{j_{\ell,i}}) \right) \subseteq \mathcal{X}_\alpha$$

be the lower bound conditions in $\psi$ whose right-hand side is $\mathcal{X}_\alpha$. Then

$$\Phi_\psi(\mathcal{F})_\alpha := \mathcal{F}_\alpha \vee \bigvee_{i'=1}^{\ell} \mathcal{T}_{i'} \left( \mathcal{A}_{i'} \cap \bigcap_{i=1}^{m_{i'}} \mathcal{S}_{i',i}^{-1}(\mathcal{F}_{j_{i',i}}) \right).$$

Since each component of $\Phi_\psi$ is a composition of monotone operations (intersection, supremum, image, preimage), $\Phi_\psi$ is clearly monotone. As previously indicated, $\Phi_\psi$ can be viewed as a procedure for improving an approximate solution of the lower bound conditions in $\psi$. Indeed, the fixed points of $\Phi_\psi$ are precisely the solutions of the lower bound conditions.

**Lemma 4.25.** *Let $\psi$ be a set of lower bound conditions. Then a tuple $\bar{\mathcal{F}} \in \mathrm{CS}(\psi)$ solves $\psi$ iff $\Phi_\psi(\bar{\mathcal{F}}) = \bar{\mathcal{F}}$.*

*Proof.* Assume $\bar{\mathcal{F}}$ solves all lower bound conditions in $\psi$. Let

$$\begin{gathered}\mathcal{T}_1\left(\mathcal{A}_1 \cap \bigcap_{i=1}^{m_1} \mathcal{S}_{1,i}^{-1}(\mathcal{X}_{j_{1,i}})\right) \subseteq \mathcal{X}_\alpha \\ \vdots \\ \mathcal{T}_\ell\left(\mathcal{A}_\ell \cap \bigcap_{i=1}^{m_\ell} \mathcal{S}_{\ell,i}^{-1}(\mathcal{X}_{j_{\ell,i}})\right) \subseteq \mathcal{X}_\alpha\end{gathered}$$

be the lower bound conditions in $\psi$ whose right-hand side is $\mathcal{X}_\alpha$. Then by assumption,

$$\begin{gathered}\mathcal{T}_1\left(\mathcal{A}_1 \cap \bigcap_{i=1}^{m_1} \mathcal{S}_{1,i}^{-1}(\mathcal{F}_{j_{1,i}})\right) \subseteq \mathcal{F}_\alpha \\ \vdots \\ \mathcal{T}_\ell\left(\mathcal{A}_\ell \cap \bigcap_{i=1}^{m_\ell} \mathcal{S}_{\ell,i}^{-1}(\mathcal{F}_{j_{\ell,i}})\right) \subseteq \mathcal{F}_\alpha\end{gathered}$$

are all true. This implies that

$$\Phi_\psi(\mathcal{F})_\alpha = \mathcal{F}_\alpha \vee \underbrace{\bigvee_{i'=1}^{\ell} \mathcal{T}_{i'}(\mathcal{A}_{i'} \cap \bigcap_{i=1}^{m_{i'}} \mathcal{S}_{i',i}^{-1}(\mathcal{F}_{j_{i',i}}))}_{\subseteq \mathcal{F}_\alpha} = \mathcal{F}_\alpha,$$

so $\bar{\mathcal{F}}$ is a fixed point of $\Phi_\psi$.

Conversely, assume that $\bar{\mathcal{F}}$ is a fixed point of $\Phi_\psi$, i.e., for every $\alpha$,

$$\mathcal{F}_\alpha \vee \bigvee_{i'=1}^{\ell} \mathcal{T}_{i'}\left(\mathcal{A}_{i'} \cap \bigcap_{i=1}^{m_{i'}} \mathcal{S}_{i',i}^{-1}(\mathcal{F}_{j_{i',i}})\right) = \Phi_\psi(\mathcal{F})_\alpha = \mathcal{F}_\alpha.$$

This implies that $\bigcup_{i'=1}^{\ell} \mathcal{T}_{i'} \left( \mathcal{A}_{i'} \cap \bigcap_{i=1}^{m_{i'}} \mathcal{S}_{i',i}^{-1}(\mathcal{F}_{j_{i',i}}) \right) \subseteq \mathcal{F}_\alpha$, so $\bar{\mathcal{F}}$ solves all lower bound conditions whose right-hand side is $\mathcal{X}_\alpha$. Since this works for all $\alpha$, $\bar{\mathcal{F}}$ solves all lower bound conditions. □

Lemma 4.25 implies that the solutions of a set $\psi$ of affine Horn conditions are precisely those fixed points of $\Phi_\psi$ that solve the upper bound conditions of $\psi$. It is easy to see that if $\bar{\mathcal{G}} \in \mathrm{CS}(\psi)$ solves the upper bound conditions, so does any $\bar{\mathcal{F}} \leq \bar{\mathcal{G}}$. If we assume that $\Phi_\psi$ has a least fixed point $\mu$, the solvability of $\psi$ thus reduces to the question of whether $\mu$ is a solution. Accordingly, we need to prove the following two things:

1. The least fixed point of $\Phi_\psi$ exists for every $\psi$ and can be computed.
2. It is decidable whether the least fixed point of $\Phi_\psi$ solves $\psi$.

We tackle the first point with the help of a famous theorem of Kleene. The version given here is slightly less general than usual, but sufficient for our purposes.

**Theorem 4.26** (Kleene's fixed point theorem)**.** *Let $L$ be a lattice with least element $0$ in which all chains have suprema. Let $f\colon L \to L$ be continuous on chains, i.e., for any chain $M$, $f(\sup M) = \sup f(M)$. Then $m = \sup \{ f^i(0) \mid i \in \mathbb{N} \}$ is the least fixed point of $f$.*

*Proof.* The continuity of $f$ on chains implies that $f$ is monotone: if $a, b \in L$ with $a \leq b$, then $\sup\{f(a), f(b)\} = f(\sup\{a, b\}) = f(b)$, so $f(a) \leq f(b)$. Now consider $\{ f^i(0) \mid i \in \mathbb{N} \}$. Because $f$ is monotone, this set is a chain and so has a supremum $m$ in $L$. We have $f(m) = f(\sup \{ f^i(0) \mid i \in \mathbb{N} \}) = \sup \{ f^{i+1}(0) \mid i \in \mathbb{N} \} = m$, so $m$ is a fixed point of $f$. Moreover, if $n$ is any fixed point of $f$, then $f^i(0) \leq n$ for all $i \in \mathbb{N}$, as is easily seen by an inductive argument, so $m = \sup \{ f^i(0) \mid i \in \mathbb{N} \} \leq n$. This means that $m$ is the least fixed point of $f$. □

**Definition 4.27** ($\bar{\mathcal{F}}^\psi$)**.** Let $\psi$ be a set of affine Horn conditions. Then $\bar{\mathcal{F}}^\psi \in \mathrm{CS}(\psi)$ denotes the least fixed point of $\Phi_\psi$.

To prove that this definition is justified, we show that we can apply Theorem 4.26 to $\mathrm{CS}(\psi)$ and $\Phi_\psi$. Every chain in $\mathrm{CS}(\psi)$ has a supremum—a maximum, in fact—because $\Phi_\psi$ satisfies ACC. Together with the monotonicity of $\Phi_\psi$ this also implies

its continuity: if $C$ is a chain, then $\Phi_\psi(\sup C) = \Phi_\psi(\max C) = \max \Phi_\psi(C) = \sup \Phi_\psi(C)$.

Since the chain $\left\{ \Phi_\psi^i(0) \;\middle|\; i \in \mathbb{N} \right\}$ must be finite, we also obtain the fact that there is some $k \in \mathbb{N}$ such that $\bar{\mathcal{F}}^\psi = \Phi_\psi^k(0)$.

**Theorem 4.28.** *Let $\psi$ be a set of affine Horn conditions. If there is a solution of $\psi$, then $\bar{\mathcal{F}}^\psi$ is its least solution. Conversely, if $\bar{\mathcal{F}}^\psi$ is not a solution of $\psi$, then $\psi$ has no solution.*

*Proof.* Let $\bar{\mathcal{G}}$ be a solution of $\psi$. In particular, $\bar{\mathcal{G}}$ must satisfy all lower bound conditions of $\psi$ and hence be a fixed point of $\Phi_\psi$ by Lemma 4.25. Since $\bar{\mathcal{F}}^\psi$ is the least fixed point of $\Phi_\psi$, we have $\bar{\mathcal{F}}^\psi \leq \bar{\mathcal{G}}$. As a solution, $\bar{\mathcal{G}}$ must also satisfy all upper bound conditions, and it follows that $\bar{\mathcal{F}}^\psi$ does as well. The second claim follows by contraposition. □

**Theorem 4.29.** *The function $\psi \mapsto \bar{\mathcal{F}}^\psi$ is computable. Moreover, it is decidable whether $\bar{\mathcal{F}}^\psi$ solves $\psi$.*

*Proof.* Computability of $\psi \mapsto \bar{\mathcal{F}}^\psi$: Let $\psi$ be a set of affine Horn conditions. As remarked previously, repeated application of $\Phi_\psi$ arrives at $\bar{\mathcal{F}}^\psi$ after finitely many steps, so we only need to establish that we can compute $\Phi_\psi$. For the purposes of the computation, we need a finite representation of each element of $\operatorname{Aff} \mathbb{Q}^k$ we encounter. For some computation steps, it is more convenient to assume that the operands are represented by coordinate systems; for others, a representation by a linear equation system is preferable. Since we can always computably recover a coordinate system from a linear equation system and vice versa, we may assume that we have either one available in each of the computation steps.

The computation of $\Phi_\psi$ involves the operations intersection, supremum, and image as well as preimage under affine transformations. Some cases involving the empty set are trivial and will not be treated explicitly.

- $\emptyset$ is represented by the coordinate system $\emptyset$ or the equation system $\bot$.
- If $\mathcal{A}, \mathcal{B}$ are represented by the equation systems $\sigma, \tau$, then $\sigma \wedge \tau$ represents $\mathcal{A} \cap \mathcal{B}$.
- If $\mathcal{A}, \mathcal{B}$ are affine subspaces with coordinate systems $\langle a, A \rangle$ and $\langle b, B \rangle$, then $\langle a, A \cup B \cup \{b - a\} \rangle$ is a coordinate system of $\mathcal{A} \vee \mathcal{B}$.

- If $\mathcal{A} \in \operatorname{Aff}\mathbb{Q}^m$ is represented by the coordinate system $\langle a, A\rangle$ and $\mathcal{T} : \mathbb{Q}^m \to \mathbb{Q}^n, x \mapsto Tx + b$ is an affine transformation, then $\langle Ta + b, T(A)\rangle$ is a coordinate system of $\mathcal{T}(\mathcal{A})$.
- If $\mathcal{A} \in \operatorname{Aff}\mathbb{Q}^n$ is represented by the linear equation system $\sigma(\bar{x})$ and $\mathcal{T} : \mathbb{Q}^m \to \mathbb{Q}^n, x \mapsto Tx + b$ is an affine transformation, then the equation system $\sigma(T\bar{x} + b)$ represents $\mathcal{T}^{-1}(\mathcal{A})$.

Decidability: $\bar{\mathcal{F}}^\psi$ solves all lower bound conditions of $\psi$ by Lemma 4.25. Consequently, it is sufficient to check whether $\bar{\mathcal{F}}^\psi$ is also a solution of all upper bound conditions in $\psi$. To do this, we need the same operations as before plus the ability to decide whether an affine subspace is contained within another. Given a coordinate system $\langle a, A\rangle$ of an affine subspace $\mathcal{A}$ and a linear equation system $\tau$ of $\mathcal{B}$, $\mathcal{A} \subseteq \mathcal{B}$ holds iff each element of $\{a\} \cup \{\, a + b \mid b \in A\}$ solves $\tau$. □

**Remark.** Obviously, we need representations not only of the sets we compute, but also of the constant sets occurring in the affine conditions. This is not a problem in practice: we obtained these sets in the first place as solution sets of linear equation systems, so these same systems can be used as the representations.

We are now able to define the algorithm MinimalSolution and prove it correct.

**Theorem 4.30.** *Let $\Psi$ be a set of affine conditions. If $\Psi$ is solvable, then* MinimalSolution$(\psi)$ *is a minimal solution of $\Psi$. If $\Psi$ is unsolvable, then* MinimalSolution$(\Psi) =$ **failure**.

*Proof.* By Theorems 4.28 and 4.29, after the end of the **for** loop in line 7, Sol contains exactly the least solutions of the solvable $\psi \in \Pr(\Psi)$. By Corollary 4.21, Sol is nonempty iff $\Psi$ is solvable, and if Sol $\neq \emptyset$, then each of its elements is a solution of $\Psi$. It immediately follows that MinimalSolution outputs a solution of $\Psi$ if $\Psi$ is solvable and **failure** otherwise. It only remains to show that in the positive case, the returned solution is minimal.

Assume Sol $\neq \emptyset$ and let $\bar{\mathcal{F}} \in$ Sol be minimal. If $\bar{\mathcal{G}}$ is any solution of $\Psi$, then $\bar{\mathcal{G}}$ is a solution of some projection of $\Psi$ (again by Corollary 4.21) and hence $\bar{\mathcal{G}}$ is above some element of Sol. Since $\bar{\mathcal{F}}$ is minimal in Sol, this shows that $\bar{\mathcal{G}} \not< \bar{\mathcal{F}}$. So $\bar{\mathcal{F}}$ is in fact a minimal solution of $\Psi$. □

**Procedure 1** MinimalSolution
**Input:** A set $\Psi$ of affine conditions
**Output:** An element of $\mathrm{CS}(\Psi)$ or **failure**

```
Sol := ∅
for ψ ∈ Pr(Ψ) do
    compute F̄ψ
    if F̄ψ satisfies the upper bound conditions in ψ then
        add F̄ψ to Sol
    end if
end for
if Sol ≠ ∅ then
    return a minimal element of Sol
else
    return failure
end if
```

We are now able to prove:

**Theorem 4.4.** *The affine solution problem is decidable.*

*Proof.* Let $\exists \bar{X}\, \varphi$ be an affine formula equation. By Corollary 4.14, we have that $\exists \bar{X}\, \varphi$ is solvable iff $\mathrm{AC}(\varphi)$ is solvable. The latter is decidable by Theorem 4.30. □

**Example 4.31.** Let P1, P2, P3, P4 be the affine projections defined in Example 4.17. Each projection P$i$ induces an operator $\Phi_i$ on $\operatorname{Aff}\mathbb{Q}^3$:

$$\Phi_{1,2} \colon \mathcal{F} \mapsto \mathcal{F} \vee \mathcal{T}_1(\mathbb{Q}^2) \vee \mathcal{T}_2(\mathcal{S}^{-1}(\mathcal{F}))$$
$$\Phi_{3,4} \colon \mathcal{F} \mapsto \mathcal{F} \vee \mathcal{T}_1(\mathbb{Q}^2) \vee \mathcal{T}_3(\mathcal{S}^{-1}(\mathcal{F}))$$

P1 and P2 induce the same operator because they contain the same lower bound conditions, and so do P3 and P4.

Iterating $\Phi_{1,2}$ on $\emptyset \in \operatorname{Aff}\mathbb{Q}^2$ results in the sequence

$$\emptyset, \left\{ \begin{pmatrix} 1 \\ 0 \end{pmatrix} \right\}, \begin{pmatrix} 1 \\ 0 \end{pmatrix} + \left[ \begin{pmatrix} 1 \\ 1 \end{pmatrix} \right] =: \mathcal{F}_1^*.$$

Since $\begin{pmatrix} 1 \\ 0 \end{pmatrix} \notin \mathcal{B}_1$ and $\begin{pmatrix} 1 \\ 1 \end{pmatrix} \notin \vec{\mathcal{B}}_2$, $\mathcal{F}_1^*$ solves neither upper bound condition, so P1 and P2 are unsolvable.

On the other hand, $\Phi_{3,4}$ results in the iteration sequence

$$\emptyset, \left\{ \begin{pmatrix} 1 \\ 0 \end{pmatrix} \right\}, \begin{pmatrix} 1 \\ 0 \end{pmatrix} + \left[ \begin{pmatrix} 1 \\ 0 \end{pmatrix} \right] =: \mathcal{F}_2^*.$$

$\mathcal{F}_2^* \subseteq \mathcal{B}_1$ is false and $\mathcal{F}_2^* \subseteq \mathcal{B}_2$ is true, so P3 is unsolvable and P4 has the solution $\mathcal{F}_2^*$.

**Remark.** As Example 4.31 suggests, it is possible that several projections of an affine formula equation contain the same lower bound conditions and consequently induce the same operator. In such cases it is sufficient to only compute the fixed point once.

## 4.4 Backwards translation

Let $\exists \bar{X}\, \varphi$ be an affine formula equation. In the previous sections, we have shown that it is decidable whether $\exists \bar{X}\, \varphi$ has a solution in linear equation systems. In fact, we can effectively compute a maximally strong solution $F_1, ..., F_n$ of $\exists \bar{X}\, \varphi$, in the following sense: If $G_1, ..., G_n$ is another solution of $\exists \bar{X}\, \varphi$ such that for all $i = 1, ..., n$, $G_i \to F_i$, then for all $i = 1, ..., n$, $G_i \leftrightarrow F_i$. Let $\mathcal{F}_1, ..., \mathcal{F}_n$ be a solution of $\mathrm{AC}(\varphi)$. By Proposition 4.9, there are linear equation systems $F_1, ..., F_n$ such that $\mathcal{F}_i = F_i^{\mathbb{Q}}$. By Corollary 4.14, $\bar{F}$ is a solution of $\varphi$.

Note that if $\mathcal{F} \subseteq \mathcal{G}$, then $F \to G$. In other words, smaller spaces translate to stronger formulas. Since we can compute a minimal solution of $\mathrm{AC}(\varphi)$ by Theorem 4.30, we also obtain a maximally strong solution of $\exists \bar{X}\, \varphi$.

**Example 4.32.** Let $\mathcal{F}_2^*$ be the solution of the projection P4 computed in Example 4.31. We can translate $\mathcal{F}_2^*$ back to the linear equation system $F(x, y) \equiv y = 0$. This gives us the solved formula equation $\varphi[X \backslash F]$:

$$0 = 0 \wedge (y = 0 \to x = 0 \vee -y = 0) \wedge (y = 0 \to x = y \vee y = 0).$$

Putting all the results of this chapter together, we obtain the following algorithm for solving affine formula equations (or proving their unsolvability):

**Procedure 2** SolveASP

**Input:** An instance $\exists \bar{X}\, \varphi$ of the affine solution problem

**Output:** A solution of $\exists \bar{X}\, \varphi$ or **failure**

1: Compute $\mathrm{Cl}(\varphi)$
2: Compute $\mathrm{AC}(\varphi)$ from $\mathrm{Cl}(\varphi)$
3: **if** MinimalSolution$(\mathrm{AC}(\varphi)) =$ **failure then**
4: **return failure**
5: **else**
6: $\bar{\mathcal{F}} :=$ MinimalSolution$(\mathrm{AC}(\varphi))$
7: **end if**
8: Compute linear equation systems $\bar{F}$ from $\bar{\mathcal{F}}$
9: **return** $\bar{F}$

# CHAPTER 5

# THE INTERVAL AND POLYHEDRAL SOLUTION PROBLEMS

In this chapter, we will discuss attempts to generalize the results of Chapter 4. Specifically, we will move from affine equalities to affine inequalities by adding the symbol $\leq$ to the language. Semantically, this means that we will be generalizing from affine spaces to *convex polyhedra*.

It would seem natural to take the same approach as in the affine solution problem discussed in Chapter 4. There are two principal reasons this is not straightforwardly possible. First, we used a covering property of affine subspaces in Lemma 4.18 to reduce the general case to that of a single subspace covering an intersection of subspaces. This allowed us to read the resulting affine Horn conditions as an approximation procedure. It is easy to see, however, that convex polyhedra do not have the same property: there can be convex polyhedra $A, B, C \subseteq \mathbb{Q}^n$ such that $A \subseteq B \cup C$, but $A \nsubseteq B$ and $A \nsubseteq C$.

In principle, we could restrict our attention to formula equations which already have Horn form, but even then, a more serious problem remains: recall that we showed the termination of Procedure 4.3.4 by demonstrating that the least fixed

point of a certain operator always exists and is reached after finitely many iterations, which is the case because the lattice $\operatorname{Aff}\mathbb{Q}^n$ is complete and satisfies the ACC. By contrast, the lattice $\operatorname{Pol}\mathbb{Q}^n$ of convex polyhedra of $\mathbb{Q}^n$ has neither of these properties. To see that $\operatorname{Pol}\mathbb{Q}^n$ is not complete, consider the set $M = \{\, P \in \operatorname{Pol}\mathbb{Q}^n \mid B_1^n(0) \subseteq P\}$, where $B_1^n(0)$ is the unit ball around the origin. Then $\bigcap M = B_1^n(0) \notin \operatorname{Pol}\mathbb{Q}^n$. To see that the ACC does not hold in $\operatorname{Pol}\mathbb{Q}^n$, consider the infinite strictly increasing sequence $([-i,i]^n)_{i\in\mathbb{N}}$ of $n$-dimensional cubes.

In software verification, *widening* is used to remedy this problem. Given a partially ordered set $P$, a *widening operator* on $P$ is a partial function $\nabla\colon P \times P \to P$ such that

- for $x, y \in P$, if $x \nabla y$ exists, then $x \nabla y \geq x$ and $x \nabla y \geq y$;
- for every weakly ascending sequence $(x_i)_{i\in\mathbb{N}}$, the weakly ascending sequence $(y_i)_{i\in\mathbb{N}}$ defined by $y_0 = x_0$ and $y_{n+1} = y_n \nabla x_{n+1}$ is eventually constant.

For instance, the original widening operator on convex polyhedra, introduced by Cousot and Halbwachs in [CH78] and called the *standard widening* in [BHRZ03], is intuitively defined as follows: Let $P_1, P_2$ be convex polyhedra in $\mathbb{Q}^n$. If $P_1 = \emptyset$, then $P_1 \nabla P_2 = P_2$, otherwise $P_1 \nabla P_2$ is the convex polyhedron defined by those inequalities of $P_1$ that every point of $P_2$ satisfies. It is straightforward to show that this operator fits the definition of a widening.

Obviously, overapproximations such as the standard widening are not suitable for finding the least fixed point of an operator in general. In software verification, this induces a tradeoff between precision and speed: coarser widenings may converge to a fixed point faster, but overestimate the set of reachable program states to a greater degree, and vice versa for finer widenings. For our purpose—deciding whether a formula equation has a solution—such a tradeoff is not acceptable, since it compromises completeness: the minimality of $\mathcal{F}^\psi$ is essential in Theorem 4.28. Thus, we will have to approach the problem of convex polyhedra from another angle.

Widening, and the wider topic of abstract interpretation, is discussed in some detail in Chapter 12 of [BM07]. The framework of abstract interpretation was developed by Cousot and Cousot in [CC77] and applied to convex polyhedra by Cousot and Halbwachs in [CH78]. [BHRZ03] features an investigation of procedures for

refining existing widening operators.

**Definition 5.1** ($\mathcal{L}_{\leq}^{\mathbb{Q}}$)**.** Let $\mathcal{L}_{\leq}^{\mathbb{Q}}$ be the language $\mathcal{L}_{\text{aff}}$ (see 4.1) extended by the binary relation symbol $\leq$.

Let $\text{Th}(\mathbb{Q})$ be the theory of $\mathbb{Q}$ over $\mathcal{L}_{\leq}^{\mathbb{Q}}$, where the symbols of $\mathcal{L}_{\text{aff}}$ are interpreted as before and $\leq$ is interpreted as the actual "less than or equal" relation. As in Chapter 4, we will tacitly interpret $\mathcal{L}_{\leq}^{\mathbb{Q}}$-formulas over $\text{Th}(\mathbb{Q})$. As a consequence, we could equivalently remove the symbol $=$ from $\mathcal{L}_{\leq}^{\mathbb{Q}}$ and define $a = b$ as an abbreviation of $a \leq b \wedge b \leq a$. We also define $a \geq b$ to mean $b \leq a$, $a < b$ to mean $\neg(b \leq a)$, and $a_1 \leq a_2 \leq ... \leq a_{n-1} \leq a_n$ to mean $a_1 \leq a_2 \wedge a_2 \leq a_3 \wedge ... \wedge a_{n-1} \leq a_n$.

**Definition 5.2** (Polyhedral formula, convex polyhedron)**.** A polyhedral formula is a conjunction of atoms of the form $\sum_{i=1}^{n} c_i x_i \leq c$ with $c, c_1, ..., c_n \in \mathbb{Q}$.

A *convex polyhedron* is a set $M \subseteq \mathbb{Q}^n$ such that there is a polyhedral formula $\varphi(x_1, ..., x_n)$ with $\varphi^{\mathbb{Q}} = M$.

Before we proceed, let us define conic and convex combinations. A *conic combination* of vectors $a_1, ..., a_n \in \mathbb{Q}^m$ is a linear combination $\sum_{i=1}^{m} c_i a_i$ such that $c_i \geq 0$ for all $i \in \{1, ..., m\}$. A *convex combination* is a linear combination with the additional conditions $0 \leq c_i \leq 1$ for all $i \in \{1, ..., m\}$ and $\sum_{i=1}^{n} c_i = 1$. Consequently, every convex combination is also a conic and affine combination. For $M \subseteq \mathbb{Q}^n$ we write $\text{cone}\, M$ for the set of conic combinations over $M$, called the *conic hull* of $M$, and $\text{conv}\, M$ for the set of convex combinations over $M$, the *convex hull* of $M$. A set $M$ satisfying $\text{conv}\, M = M$ is said to be *convex*. Convex polyhedra are in fact convex in this sense, as is easily seen. Note that the convex hull of a finite set is always bounded.

The following well-known theorem shows that convex polyhedra can equivalently be defined "from without" (as an intersection of half-spaces) and "from within" (generated by a set of points and a set of rays). It is a generalization of the observation that affine subspaces can be represented as intersections of affine hyperplanes or as affine hulls of sets of points.

**Theorem 5.3** (Minkowski-Weyl)**.** *Let $P \subseteq \mathbb{Q}^n$. The following are equivalent:*

1. *$P$ is a convex polyhedron.*
2. *There are finite sets $V, R \subseteq \mathbb{Q}^n$ such that $P = \operatorname{conv} V + \operatorname{cone} R$.*

A proof of Theorem 5.3 can be found in [Zie12], where it is called the *main theorem for polyhedra*.

If $P$ is a convex polyhedron, we call any polyhedral formula that defines it an *H-representation* of $P$ and a tuple $\langle V, R\rangle$ such that $P = \operatorname{conv} V + \operatorname{cone} R$ a *V-representation* of $P$.

**Proposition 5.4.** *Let $P \subseteq \mathbb{Q}^n$ be a convex polyhedron and $\langle V, R\rangle$ a V-representation of $P$.*

1. *$P = \emptyset$ iff $V = \emptyset$.*
2. *$P$ is bounded iff $V = \emptyset$ or $R = \emptyset$.*

*Proof.*
1. If $V = \emptyset$, then $P = \emptyset + \operatorname{cone} R = \emptyset$. If $a \in V$, then $a = a + 0 \in \operatorname{conv} V + \operatorname{cone} R = P$.
2. By 1, $V = \emptyset$ implies $P = \emptyset$ and therefore $P$ is bounded. If $R = \emptyset$, then $P = \operatorname{conv} V$ and hence bounded. □

We can now define the solution problem that is the subject of this chapter.

**Definition 5.5** (Polyhedral solution problem)**.**

1. A polyhedral formula equation is a $\Pi_1$ formula equation over $\mathcal{L}_{\leq}^{\mathbb{Q}}$.
2. The rational polyhedral solution problem is the solution problem $\langle \mathrm{Th}(\mathbb{Q}), \mathcal{F}, \mathcal{C}\rangle$ where $\mathcal{C}$ is the set of polyhedral formulas and $\mathcal{F}$ is the set of polyhedral formula equations.

**Open problem.** *Is the rational polyhedral solution problem decidable?*

We will examine this problem from two different perspectives. We will show that two specific subproblems of the polyhedral decision problem are decidable, and we will adapt an argument originally presented in [Mon19] to demonstrate that even a slight extension of the problem results in undecidability.

## 5.1 The polyhedral solution problem in the rational plane

First, we will quickly treat a very simple case of the rational polyhedral solution problem only involving a single formula variable and rotations in $\mathbb{Q}^2$.

**Theorem 5.6.** *Let $\mathcal{F}$ be the class of formula equations $\exists X\,\varphi$ where $X$ is a binary formula variable and $\varphi$ is a conjunction of linear Horn clauses of the forms*

$$\begin{aligned} &\vdash X(a,b),\\ X(x,y) &\vdash X(s(x,y),t(x,y)),\\ X(c,d) &\vdash, \end{aligned}$$

*with $a,b,c,d \in \mathbb{Q}$ and $\rho\colon \mathbb{Q}^2 \to \mathbb{Q}^2, \begin{pmatrix} x \\ y \end{pmatrix} \to \begin{pmatrix} s(x,y) \\ t(x,y) \end{pmatrix}$ a rotation. Moreover, let $\mathcal{C}$ be the class of polyhedral formulas. Then the solution problem $\langle \mathrm{Th}(\mathbb{Q}), \mathcal{F}, \mathcal{C} \rangle$ is decidable.*

*Proof.* Let $\exists X\,\varphi \in \mathcal{F}$ and let

$$p_1 = \begin{pmatrix} a_1 \\ b_1 \end{pmatrix}, ..., p_m = \begin{pmatrix} a_m \\ b_m \end{pmatrix}$$

be all the points that occur in clauses of the first form,

$$\rho_1\colon \begin{pmatrix} x \\ y \end{pmatrix} \to \begin{pmatrix} s_1(x,y) \\ t_1(x,y) \end{pmatrix}, ..., \rho_n\colon \begin{pmatrix} x \\ y \end{pmatrix} \to \begin{pmatrix} s_n(x,y) \\ t_n(x,y) \end{pmatrix}$$

all the rotations that occur in clauses of the second form, and

$$q_1 = \begin{pmatrix} c_1 \\ d_1 \end{pmatrix}, ..., q_\ell = \begin{pmatrix} c_\ell \\ d_\ell \end{pmatrix}$$

all the points that occur in clauses of the third form. $X$ describes a convex polyhedron that must include all points $p_1, ..., p_m$, be closed under the rotations $\rho_1, ..., \rho_n$, and exclude the points $q_1, ..., q_\ell$. We construct the least convex polyhedron $P$ satisfying the former two conditions. It is then straightforward to check whether $P$

satisfies the latter condition, and $\varphi$ has a solution iff this is the case, due to the minimality of $P$. The construction proceeds as follows. If $m = 0$, then $P = \emptyset$. Likewise, if $m > 0$ and $n = 0$, then $P = \mathrm{conv}(p_1, ..., p_m)$. Otherwise, consider the rotations $\rho_1, ..., \rho_n$. Clearly, $P$ must at least contain the orbits of the $p_i$ under these rotations. Such a rotation is periodic iff its angle is $\pm\pi/2$ or $\pi$ (for a proof, see Corollary 1 in [Tan96]), in which case its period is at most 4. Thus, if the orbit of a point $p$ under a rotation $\rho$ has more than 4 elements, we can conclude that it is infinite. There are now two cases to consider. If all the $\rho_i$ are periodic—that is, their angles are all $\pm\pi/2$ or $\pi$—then we may assume without loss of generality that there is only one rotation $\rho$ and its angle is either $\pi/2$ or $\pi$. In either case, each $p_i$ has a finite orbit $O_i$ and $O = \bigcup_{i=1}^{n} O_i$ is the least set that contains all $p_i$ and is closed under $\rho$. It follows that $P = \mathrm{conv}(O)$ is the least convex set (and thus certainly the least convex polyhedron) with these properties.

Now assume that one rotation $\rho_j$ is aperiodic. This implies that $P$ is unbounded, for if it were bounded, then it would be the convex hull of finitely many points and thus could not be closed under $\rho_j$. Moreover, whenever $P$ contains a point $p \neq \mathbf{0}$, $P$ must also contain an entire circle centered at the origin and passing through $p$. This rules out $P$ being contained in any halfspace, so $P = \mathbb{Q}^2$. □

**Example 5.7.** Let

$$\begin{aligned}\Phi \equiv \exists X.\ \forall x, y.\quad & X(2,3) \\ & \wedge \left(X(x,y) \to X\left(\frac{3}{5}x + \frac{4}{5}y, -\frac{4}{5}x + \frac{3}{5}y\right)\right) \\ & \wedge \left(X(x,y) \to X(y,-x)\right) \\ & \wedge \neg X(4,6)\end{aligned}$$

Then, following the terminology from the proof of Theorem 5.6, we have

$$p = \begin{pmatrix} 2 \\ 3 \end{pmatrix}, \qquad \rho_1 : \begin{pmatrix} x \\ y \end{pmatrix} \mapsto \frac{1}{5}\begin{pmatrix} 3x + 4y \\ -4x + 3y \end{pmatrix},$$

$$q = \begin{pmatrix} 4 \\ 6 \end{pmatrix}, \qquad \rho_1 : \begin{pmatrix} x \\ y \end{pmatrix} \mapsto \begin{pmatrix} y \\ -x \end{pmatrix}.$$

Assume $P \subseteq \mathbb{Q}^2$ is a convex polyhedron that solves $\Phi$. Then $P$ must contain $p$ and be closed under $\rho_1$ and $\rho_2$. Since $\rho_1$ is not periodic, this is only possible if $P = \mathbb{Q}^2$, but on the other hand, $q \notin P$, which is a contradiction. Thus, $\Phi$ has no solution.

Extending the above solution problem to more than two dimensions already introduces a massive complication in the fact that rotations can have different axes.

In the next section, we will show the decidability of a more complex subproblem of the polyhedral decision problem.

## 5.2 The interval solution problem

The *interval solution problem* we will be discussing in this section features two major differences from the general polyhedral solution problem. First, the sets we consider are not general convex polyhedra, but rather products of (bounded or unbouded) intervals, or in other words, convex polyhedra aligned along the coordinate axes. Second, we will be working over the integers instead of the rational numbers. The main reason for this is that it allows us to ignore negations in formulas, which will simplify technical matters significantly.

**Definition 5.8** ($\mathcal{L}^{\mathbb{Z}}_{\leq}$)**.** Let $\mathcal{L}^{\mathbb{Z}}_{\leq}$ be the first-order language consisting of

- the constants 0 and 1;
- the binary function symbol $+$;
- the unary function symbols $\{\, c \mid c \in \mathbb{Z}\}$;
- the binary predicate symbols $=$ and $\leq$.

In the sequel, let $\mathrm{Th}(\mathbb{Z})$ be the theory of $\mathbb{Z}$ over $\mathcal{L}^{\mathbb{Z}}_{\leq}$, where 0 and 1 are interpreted as the integers 0 and 1, $+$ as actual addition, each unary function symbol $c$ as the function $x \mapsto cx$, and $\leq$ as the actual "less than or equal" relation.

We will interpret terms and formulas of $\mathcal{L}^{\mathbb{Z}}_{\leq}$ modulo $\mathrm{Th}(\mathbb{Z})$, as usual. We also use $a \geq b$, $a < b$, and $a_1 \leq ... \leq a_n$ in the same sense as in the case of $\mathbb{Q}$. There are two observations about $\mathrm{Th}(\mathbb{Z})$ that simplify the structure of the formulas we have to consider.

1. $s(\bar{x}) = t(\bar{y})$ is equivalent to $t(\bar{y}) \leq s(\bar{x}) \wedge t(\bar{y}) \geq s(\bar{x})$, so we can assume without loss of generality that all atomic formulas are inequalities.
2. $\neg s(\bar{x}) \leq t(\bar{y})$ is equivalent to $s(\bar{x}) \geq t(\bar{y}) + 1$, and we will shortly see that this allows us to disregard negation entirely.

**Definition 5.9** (Interval formula)**.** An *interval formula* over the variables $x_1, ..., x_n$ is a conjunction of formulas of the form $x_i \leq c$ or $x_i \geq c$ for $c \in \mathbb{Z}$.

Every interval formula can be written as a conjunction of at most 2 linear inequalities per variable it contains. The interpretation of an interval formula $I(x_1, ..., x_n)$ in $\mathbb{Z}$ is a box in $\mathbb{Z}^n$, i.e., a product of intervals, each of which may be bounded, bounded in one direction, or unbounded. Note that since $ax + b \leq c$ is equivalent to $x \leq \lfloor \frac{c-b}{a} \rfloor$, allowing general terms in interval formulas would provide no extra generality.

**Definition 5.10** (Interval formula equation)**.** An *interval formula equation* is a $\Pi_1$ formula equation over $\mathcal{L}^{\mathbb{Z}}_{\leq}$ in which no term or constant atom contains more than one individual variable.

Note that formula variables may contain any number of individual variables. Let us describe the structure of an interval formula equation $\exists \bar{I}\, \varphi$ in a little more detail. By definition, $\varphi$ is $\Pi_1$, and hence in prenex form. Therefore, $\varphi \equiv \forall \bar{x}\, \varphi'$ with $\varphi'$ quantifier-free. Since equivalence-preserving transformations do not affect the existence of solutions of a formula equation, we can transform $\varphi'$ into NNF. A formula in NNF is a $\wedge\vee$-combination of (possibly negated) atoms, and by observations 1 and 2, we may disregard both equality atoms and negations. The result is that we can assume that $\varphi'$ is an $\wedge\vee$-combination of formulas of the form $I(a_1 x_{i_1} + b_1, ..., a_m x_{i_m} + b_m)$ (a formula variable instantiated with terms) or $s(x) \leq t$ or $s(x) \geq t$ (constant atoms). Modulo $\mathrm{Th}(\mathbb{Z})$, each such atom is equivalent to an atom of the form $x \leq c$ or $x \geq c$, so the latter are actually the only atoms we need.

**Definition 5.11** (Interval solution problem)**.** The *interval solution problem* is the solution problem $\langle \mathrm{Th}(\mathbb{Z}), \Phi, \mathcal{C} \rangle$ where $\mathcal{C}$ is the class of interval formulas and $\Phi$ is the class of interval formula equations.

As in the case of the affine solution problem, we have:

**Theorem 5.12.** *The interval solution problem is decidable.*

The first step to solving the interval solution problem is the observation that the solution of each formula variable in an interval formula equation can assume one

of only finitely many forms, according to the shape of the box it describes. For each dimension of the box, there can be a lower bound, an upper bound, both, or neither. Moreover, the box might be empty, in which case it is described by the formula $\bot$. This notion is captured in the following definition.

**Definition 5.13** (Configuration)**.**

1. Let $\exists \bar{I}\, \varphi$ be an interval formula equation and $I(x_1, ..., x_n)$ a formula variable in $\varphi$. Let $L, R \subseteq \{1, ..., n\}$ and $(\ell_i)_{i \in L}$ and $(r_i)_{i \in R}$ be fresh individual variables. Then the formula

$$\bigwedge_{i \in L} x_i \geq \ell_i \wedge \bigwedge_{j \in R} x_j \ \leq r_j$$

   is called a *configuration* of $I$. We call the variables $(\ell_i)_{i \in L}$ and $(r_i)_{i \in R}$ its interval variables.
   Moreover, $\bot$ is a configuration of every formula variable.
2. Let $\exists \bar{I}\, \varphi$ be an interval formula equation over the formula variables $I_1, ..., I_m$. Furthermore, let $C_1, ..., C_m$ be configurations of $I_1, ..., I_m$ with distinct interval variables. Then $[I_1 \backslash C_1, ..., I_m \backslash C_m]$ is called a configuration of $\varphi$. The interval variables of a configuration of $\varphi$ are those of the $C_1, ..., C_m$.
   We write $\mathrm{Conf}(\varphi)$ for the set of configurations of $\varphi$. Configurations that differ only in the names of interval variables are considered equal.

**Theorem 5.14.** *Let $\exists \bar{I}\, \varphi$ be an instance of the interval solution problem. The following are equivalent:*

1. *$\exists \bar{I}\, \varphi$ has a solution.*
2. *There is $\tau \in \mathrm{Conf}(\varphi)$ with interval variables $y_1, ..., y_n$ such that $\mathrm{Th}(\mathbb{Z}) \models \exists \bar{y}\, \varphi\tau$.*

*Proof.* By definition, $\varphi$ is prenex, so $\varphi \equiv \forall \bar{x}\, \varphi'$ with $\varphi'$ quantifier-free.

Assume that $\varphi$ has a solution $\sigma = [I_1 \backslash F_1, ..., I_n \backslash F_n]$. Consider $I_k(x_1, ..., x_{m_k})$ and $F_k(x_1, ..., x_{m_k})$ for $1 \leq k \leq n$. As an interval formula, $F_k$ is equivalent to either $\bot$ or a formula of the form

$$\bigwedge_{i \in L(k)} x_i \geq c_{k,i} \wedge \bigwedge_{j \in R(k)} x_j \ \leq d_{k,j}$$

with $L(k), R(k) \subseteq \{1, ..., m_k\}$ and $c_{k,i}, d_{k,j} \in \mathbb{Z}$. It follows that for each $k$,

$$C_k \equiv \bigwedge_{i \in L(k)} x_i \geq \ell_{k,i} \wedge \bigwedge_{j \in R(k)} x_j \ \leq r_{k,j}$$

is a configuration of $I_k$ and hence $\tau = [I_1 \backslash C_1, ..., I_m \backslash C_m]$ is a configuration of $\varphi$. Note, moreover, that

$$\varphi\tau[\bar{\ell}\backslash\bar{c}, \bar{r}\backslash\bar{d}] \equiv \varphi[I_1 \backslash C_1, ..., I_m \backslash C_m][\bar{\ell}\backslash\bar{c}, \bar{r}\backslash\bar{d}] \equiv \varphi\sigma.$$

Since $\mathrm{Th}(\mathbb{Z}) \models \varphi\sigma$ by assumption, it follows that $\mathrm{Th}(\mathbb{Z}) \models \varphi\tau[\bar{\ell}\backslash\bar{c}, \bar{r}\backslash\bar{d}]$ and hence $\mathrm{Th}(\mathbb{Z}) \models \exists\bar{\ell}\exists\bar{r}\,\varphi\tau$.

Conversely, assume that $\tau$ is a configuration of $\varphi$ with interval variables $y_1, ..., y_n$ such that $\mathrm{Th}(\mathbb{Z}) \models \exists\bar{y}\,\varphi\tau$. It follows that there are $c_1, ..., c_m \in \mathbb{Z}$ such that $\mathrm{Th}(\mathbb{Z}) \models \psi \equiv \varphi\tau[y_1 \backslash c_1, ..., y_m \backslash c_m]$. Now $\sigma = \tau[y_1 \backslash c_1, ..., y_m \backslash c_m]$ is a solution of $\exists\bar{I}\,\varphi$. □

Note that with Theorem 5.14, we have reduced the interval solution problem from a second-order to a first-order problem: in order to check whether an interval formula equation is solvable, we only need to compute its finitely many configurations and check the resulting first-order formulas. If $\exists\bar{I}\,\varphi$ is an interval formula equation and $\tau$ is a configuration of $\varphi$ with interval variables $y_1, ..., y_n$, then $\exists\bar{y}\,\varphi\tau$ is of the form $\exists\bar{y}\forall\bar{x}\,\varphi'(\bar{x}, \bar{y})$. In $\varphi'$, all formula variables have been replaced with conjunctions of atoms of the forms $ax + b \leq y$ or $ax + b \geq y$. Consequently, $\varphi'$ is an $\wedge\vee$-combination of inequalities of the forms $ax + b \leq y$, $ax + b \geq y$, $x \leq c$, and $x \geq c$. The next step will be reducing the quantifier complexity of $\exists\bar{y}\forall\bar{x}\,\varphi'(\bar{x}, \bar{y})$ to $\Sigma_1$ by eliminating the $\forall\bar{x}$ quantifier. Before we can do this, we will need some lemmas. The first two concern the definability of division with rounding, minimum, and maximum in $\mathrm{Th}(\mathbb{Z})$.

**Lemma 5.15.** *Let $a \in \mathbb{Z}, a > 0$. Then the functions $x \mapsto \lfloor \frac{x}{a} \rfloor$ and $x \mapsto \lceil \frac{x}{a} \rceil$ are definable in* $\mathrm{Th}(\mathbb{Z})$.

*Proof.* Consider the formula

$$F_{\lfloor \frac{\cdot}{a} \rfloor}(x, y) \equiv x - a + 1 \leq ay \leq x.$$

$\mathrm{Th}(\mathbb{Z}) \vDash F_{\lfloor \frac{\cdot}{a} \rfloor}(b,c)$ iff $c \in [\frac{b}{a} - \frac{a-1}{a}, \frac{b}{a}]$. This interval contains exactly one integer, namely $\lfloor \frac{b}{a} \rfloor$. Therefore, $F_{\lfloor \frac{\cdot}{a} \rfloor}(x,y)$ defines $x \mapsto \lfloor \frac{x}{a} \rfloor$. By an analogous argument, we obtain

$$F_{\lceil \frac{\cdot}{a} \rceil}(x,y) \equiv x \leq ay \leq x + a - 1$$

as the definition of $x \mapsto \lceil \frac{x}{a} \rceil$. □

**Lemma 5.16.** *Let $n \geq 1$. Then the $n$-ary maximum and minimum functions are definable in* $\mathrm{Th}(\mathbb{Z})$.

*Proof.* The $n$-ary minimum and maximum are defined by

$$F^n_{\min}(x_1, ..., x_n, y) \equiv \bigwedge_{i=1}^{n} y \leq x_i \wedge \bigvee_{i=1}^{n} y = x_i$$

and

$$F^n_{\max}(x_1, ..., x_n, y) \equiv \bigwedge_{i=1}^{n} y \geq x_i \wedge \bigvee_{i=1}^{n} y = x_i,$$

respectively. □

Because of Lemmas 5.15 and 5.16 we are justified in making the following definitons.

**Definition 5.17.** Let $\varphi(x, \bar{y})$ be an $\mathcal{L}^{\mathbb{Z}}_{\leq}$-formula. We define the formulas $\varphi(\lfloor \frac{x}{a} \rfloor, \bar{y})$, $\varphi(\lceil \frac{x}{a} \rceil, \bar{y})$, $\varphi(\min(x_1, ..., x_n), \bar{y})$, and $\varphi(\max(x_1, ..., x_n), \bar{y})$ to be abbreviations of $\mathcal{L}^{\mathbb{Z}}_{\leq}$-formulas:

$$\begin{aligned}
\varphi(\left\lfloor \frac{x}{a} \right\rfloor, \bar{y}) &\equiv \exists z.\ x - a + 1 \leq az \leq x \wedge \varphi(z, \bar{y}),\\
\varphi(\left\lceil \frac{x}{a} \right\rceil, \bar{y}) &\equiv \exists z.\ x \leq az \leq x + a - 1 \wedge \varphi(z, \bar{y}),\\
\varphi(\min(x_1, ..., x_n), \bar{y}) &\equiv \bigvee_{i=1}^{n} \bigwedge_{j=1}^{n} x_i \leq x_j \wedge \varphi(x_i, \bar{y}),\\
\varphi(\max(x_1, ..., x_n), \bar{y}) &\equiv \bigvee_{i=1}^{n} \bigwedge_{j=1}^{n} x_i \geq x_j \wedge \varphi(x_i, \bar{y}).
\end{aligned}$$

There is one more technical lemma we will require.

**Lemma 5.18.** *For any first-order formulas $\varphi(\bar{x}, y), \psi(y)$, we have*

$$\forall \bar{x} \exists y.\ \varphi(\bar{x}, y) \wedge \psi(y), \exists^{\leq 1} y\, \psi(y) \vDash \exists y \forall \bar{x}.\ \varphi(\bar{x}, y) \wedge \psi(y).$$

*Proof.* Let $\mathcal{M}$ be any model of $\forall \bar{x} \exists y.\ \varphi(\bar{x}, y) \wedge \psi(y)$ and $\exists^{\leq 1} y\, \psi(y)$. In particular, $\psi^{\mathcal{M}} \subseteq \mathcal{M}$ contains exactly one element, say $c$. Therefore, $\mathcal{M} \vDash \forall \bar{x}.\ \varphi(\bar{x}, c) \wedge \psi(c)$ and hence $\mathcal{M} \vDash \exists y \forall \bar{x}.\ \varphi(\bar{x}, y) \wedge \psi(y)$. □

We can now prove that modulo $\mathrm{Th}(\mathbb{Z})$, every universal formula of a certain simple form is equivalent to an existential formula.

**Theorem 5.19.** *Let $\varphi(y_1, ..., y_n) \equiv \forall \bar{x}\, \varphi'(x_1, ..., x_m, y_1, ..., y_n)$ be a fully indicated formula over $\mathcal{L}^{\mathbb{Z}}_{\leq}$ such that $\varphi'$ is quantifier-free and each atomic formula in $\varphi'$ contains at most one of the $x_i$. Then there is a formula $\psi(y_1, ..., y_n) \equiv \exists \bar{z}\, \psi'(y_1, ..., y_n, z_1, ..., z_k)$ with $\psi'$ quantifier-free such that $\mathrm{Th}(\mathbb{Z}) \vDash \varphi \leftrightarrow \psi$.*

*Proof.* As we mentioned before, we may assume without loss of generality that $\varphi$ contains no negations. We may also assume that every inequality in $\varphi$ that contains $x_i$ is of the form $ax_i \leq t(\bar{y})$ or $ax_i \geq t(\bar{y})$ with $a > 0$, because if $a < 0$, we can multiply the inequality by $-1$. We first transform $\varphi'$ into a formula in which the variables $x_i$ only occur with coefficient 1. This is possible because of the definability of division with rounding.

Let $ax_i \leq t(\bar{y})$ be an atom in $\varphi$. If $a = 1$, we are done. If $a > 1$, $ax_i \leq t(\bar{y})$ is equivalent to $x_i \leq \lfloor \frac{t(\bar{y})}{a} \rfloor$ in $\mathrm{Th}(\mathbb{Z})$, and the latter is defined as an abbreviation of $\exists z.\ t(\bar{y}) - a + 1 \leq az \leq t(\bar{y}) \wedge x \leq z$. Likewise, $ax_i \geq t(\bar{y})$ with $a > 1$ is equivalent to $\exists z.\ t(\bar{y}) \leq az \leq t(\bar{y}) + a - 1 \wedge x \geq z$ modulo $\mathrm{Th}(\mathbb{Z})$. By rewriting all atomic formulas in $\varphi$ containing one of the $x_i$ in this manner, we can ensure that the $x_i$ only occur in atoms of the form $x_i \leq t(\bar{y}, \bar{z})$ or $x_i \geq t(\bar{y}, \bar{z})$, at the cost of introducing new existentially quantified variables below the universal quantifier block in $\varphi$. But since $\mathrm{Th}(\mathbb{Z}) \vDash \exists^{\leq 1} z\, F_{\lfloor \frac{\cdot}{a} \rfloor}(t(\bar{y}), z)$ —because $F_{\lfloor \frac{\cdot}{a} \rfloor}$ defines a function—, we can apply Lemma 5.18 to move the new existential quantifiers outside the original universal ones. Thus, we obtain a formula $\sigma(y_1, ..., y_n) \equiv \exists \bar{z} \forall \bar{x}\, \sigma'(x_1, ..., x_m, y_1, ..., y_n, z_1, ..., z_k)$ that is equivalent to $\varphi$ modulo $\mathrm{Th}(\mathbb{Z})$ and satisfies the condition that the $x_i$ only occur with trivial coefficients. We are done

if we can show that $\forall \bar{x}\, \sigma'(\bar{x}, \bar{y}, \bar{z})$ is equivalent to a quantifier-free formula with the free variables $\bar{y}, \bar{z}$.

The formula $\forall \bar{x}\, \sigma'(\bar{x}, \bar{y}, \bar{z})$ is equivalent to a formula of the form $\bigwedge_{k=1}^{N} \forall \bar{x}\, \sigma_k(\bar{x}, \bar{y}, \bar{z}) \vee \tau_k(\bar{y}, \bar{z})$, where the $\sigma_k$ are disjunctions of atoms containing the $x_i$ and $\tau_k$ are disjunctions of atoms that do not contain the $x_i$. The $\tau_k$ already have the required form, so we only need to show that each of the $\forall \bar{x}\, \sigma_k(\bar{x}, \bar{y}, \bar{z})$ is equivalent to a formula that only contains $\bar{y}$ and $\bar{z}$.

$\sigma_k$ is of the form

$$\begin{aligned} &\bigvee_{i=1}^{\ell_1} x_1 \geq s_{1,i}(\bar{y}, \bar{z}) \vee \bigvee_{i=1}^{r_1} x_1 \leq t_{1,i}(\bar{y}, \bar{z}) \vee \\ &\qquad \vdots \\ \vee &\bigvee_{i=1}^{\ell_m} x_m \geq s_{m,i}(\bar{y}, \bar{z}) \vee \bigvee_{i=1}^{r_m} x_m \leq t_{m,i}(\bar{y}, \bar{z}). \end{aligned}$$

Since each line involves only one of the universally quantified variables, $\forall \bar{x}\, \sigma_k$ can equivalently be written as

$$\begin{aligned} &\forall x_1.\ \bigvee_{i=1}^{\ell_1} x_1 \geq s_{1,i}(\bar{y}, \bar{z}) \vee \bigvee_{i=1}^{r_1} x_1 \leq t_{1,i}(\bar{y}, \bar{z}) \vee \\ &\qquad \vdots \\ \vee \forall x_m.\ &\bigvee_{i=1}^{\ell_m} x_m \geq s_{m,i}(\bar{y}, \bar{z}) \vee \bigvee_{i=1}^{r_m} x_m \leq t_{m,i}(\bar{y}, \bar{z}), \end{aligned}$$

so we can restrict our attention to the case of $m = 1$:

$$\forall x\, \sigma_k \equiv \forall x.\ \bigvee_{i=1}^{\ell} x \geq s_i(\bar{y}, \bar{z}) \vee \bigvee_{i=1}^{r} x \leq t_i(\bar{y}, \bar{z}).$$

This formula can be read as a statement about integer intervals: it asserts that the intervals $[s_i(\bar{y}, \bar{z}), \infty), i = 1, ..., \ell$, and $(-\infty, t_i(\bar{y}, \bar{z})], i = 1, ..., r$, cover $\mathbb{Z}$. Obviously, the truth of this statement depends entirely on the values of the terms $s_i(\bar{x}, \bar{y}), t_j(\bar{x}, \bar{y})$. Accordingly, we will reduce it to a quantifier-free formula over the same free variables. If $\ell = 0$ or $r = 0$, then $\forall x \sigma_j \leftrightarrow \bot$ modulo $\mathrm{Th}(\mathbb{Z})$ and we are finished, so assume $\ell, r \geq 1$. Observe that $\bigcup_{i=1}^{\ell} [s_i(\bar{y}, \bar{z}), \infty) = [\min_{i=1}^{\ell} s_i(\bar{y}, \bar{z}), \infty)$.

Consequently, we have

$$\begin{aligned}\mathrm{Th}(\mathbb{Z}) \models \forall x\, \sigma_k &\equiv \forall x.\ \bigvee_{i=1}^{\ell} x \geq s_i(\bar{y},\bar{z}) \vee \bigvee_{i=1}^{r} x \leq t_i(\bar{y},\bar{z}) \\ &\leftrightarrow \forall x.\ x \geq \min_{i=1}^{\ell} s_i(\bar{y},\bar{z}) \vee x \leq \max_{j=1}^{r} t_j(\bar{y},\bar{z}) \\ &\leftrightarrow \min_{i=1}^{\ell} s_i(\bar{y},\bar{z}) \leq \max_{j=1}^{r} t_j(\bar{y},\bar{z}) + 1.\end{aligned}$$

The last formula is the abbreviation of a quantifier-free $\mathcal{L}_{\leq}^{\mathbb{Z}}$-formula. This completes the proof. □

**Corollary 5.20.** *Let $\exists \bar{I} \forall \bar{x}\, \varphi(\bar{x})$ be an instance of the interval solution problem. There are quantifier-free $\mathcal{L}_{\leq}^{\mathbb{Z}}$-formulas $\psi_1(\bar{y},\bar{z}),...,\psi_n(\bar{y},\bar{z})$ such that $\exists \bar{I} \forall \bar{x}\, \varphi(\bar{x})$ has a solution iff $\mathrm{Th}(\mathbb{Z}) \models \exists \bar{y} \exists \bar{z}\, \psi_i(\bar{y},\bar{z})$ for some $i$.*

*Proof.* Let $\exists \bar{I}\, \varphi$ be an interval formula equation. By Theorem 5.14, $\exists \bar{I}\, \varphi$ has a solution if $\mathrm{Th}(\mathbb{Z}) \models \exists \bar{y}\, \varphi\tau$ for some configuration $\tau$ of $\varphi$ with interval variables $\bar{y}$. Let $\tau_1,...,\tau_n$ be the finitely many configurations of $\varphi$. Each $\varphi\tau_i$ is of the form $\forall \bar{x}\, \theta_i(\bar{x},\bar{y}_i)$ with $\theta_i$ quantifier-free. Because of the forms of $\varphi$ and $\tau_i$, the $\varphi\tau_i$ satisfy the prerequisites of Theorem 5.19 and hence we can compute quantifier-free formulas $\psi_1(\bar{y}_1,\bar{z}_1),...,\psi_n(\bar{y}_n,\bar{z}_n)$ such that $\mathrm{Th}(\mathbb{Z}) \models \varphi\tau_i \leftrightarrow \exists \bar{z}_i\, \psi_i$. We obtain

$$\begin{aligned}\exists \bar{I}\, \varphi \text{ has a solution} &\Leftrightarrow \mathrm{Th}(\mathbb{Z}) \models \exists \bar{y}_i\, \varphi\tau_i \text{ for some } 1 \leq i \leq n \\ &\Leftrightarrow \mathrm{Th}(\mathbb{Z}) \models \exists \bar{y}_i \exists \bar{z}_i\, \psi_i \text{ for some } 1 \leq i \leq n.\end{aligned}$$ □

It follows from Corollary 5.20 that in order to decide instances of the interval decision problem, it is sufficient to decide the validity of $\Sigma_1$ $\mathcal{L}_{\leq}^{\mathbb{Z}}$-formulas in $\mathrm{Th}(\mathbb{Z})$. In the next section, we briefly describe how that can be accomplished, which will allow us to complete the proof of Theorem 5.12.

Let $\varphi$ be a prenex $\Sigma_1$ $\mathcal{L}_{\leq}^{\mathbb{Z}}$-formula, i.e. $\varphi \equiv \exists \bar{y}\, \varphi'$ with $\varphi'$ quantifier-free. We can transform $\varphi'$ into disjunctive normal form (see Definition 1.12). Due to the algebraic properties of $\mathrm{Th}(\mathbb{Z})$, we can assume that this DNF contains only positive literals and each atom has the form $t(\bar{y}) \leq 0$ for some $\mathcal{L}_{\leq}^{\mathbb{Z}}$-term $t$. We therefore obtain

**Proposition 5.21.** *Let $\varphi \equiv \exists \bar{y}\, \varphi'(\bar{y})$ be a closed $\Sigma_1$-formula over $\mathcal{L}^{\mathbb{Z}}_{\leq}$. Then there are $M, N_1, ..., N_M \in \mathbb{N}$ and $\mathcal{L}^{\mathbb{Z}}_{\leq}$-terms $t_{i,j}$ for $1 \leq i \leq M, 1 \leq j \leq N_M$ such that*

$$\mathrm{Th}(\mathbb{Z}) \models \exists \bar{y}\, \varphi'(\bar{y}) \leftrightarrow \bigvee_{i=1}^{M} \exists \bar{y} \bigwedge_{j=1}^{N_i} t_{i,j}(\bar{y}) \leq 0.$$

Deciding formulas of the form $\exists \bar{y} \bigwedge_{j=1}^{N} t_{i,j}(\bar{y}) \leq 0$ is a problem of *integer linear programming*. This subject is treated in more detail in [KS08] and [WN99]. We call an $\mathcal{L}^{\mathbb{Z}}_{\leq}$-formula of the form $\exists \bar{y} \bigwedge_{i=1}^{n} t(\bar{y}) \leq 0$ a *system of linear constraints*.

**Theorem 5.22.** *The satisfiability of systems of linear constraints in* $\mathrm{Th}(\mathbb{Z})$ *is decidable.*

*Sketch of proof.* We sketch a proof that is presented in detail in [KS08]. Considered over $\mathbb{Q}$, the satisfiability of a system of linear constraints $S$ can be decided by a variant of the *simplex algorithm*. Of course a solution obtained in this way may not be integral. Therefore, we use a *branch and bound* procedure: if $S$ has a solution in which the variable $x$ is assigned a value $c \in \mathbb{Q} \setminus \mathbb{Z}$, we branch by defining two new systems of linear constraints $S \cup \{x \leq \lfloor c \rfloor\}$ and $S \cup \{x \geq \lceil c \rceil\}$ and applying the simplex algorithm again. It is not obvious that this branching procedure terminates, but it can be shown that if there is a solution, there is also a solution within a computable bound. □

**Corollary 5.23.** *The validity of closed $\Sigma_1$ formulas modulo* $\mathrm{Th}(\mathbb{Z})$ *is decidable.*

*Proof.* Immediately by Proposition 5.21 and Theorem 5.22. □

We can now prove

**Theorem 5.12.** *The interval solution problem is decidable.*

*Proof.* Let $\exists \bar{I}\, \varphi \equiv \exists \bar{I} \forall \bar{x}\, \varphi'(\bar{x})$ be an interval formula equation. By Theorem 5.14, $\exists \bar{I}\, \varphi$ has a solution modulo $\mathrm{Th}(\mathbb{Z})$ iff there is a configuration $\tau$ of $\varphi$ with interval variables $\bar{y}$ such that $\mathrm{Th}(\mathbb{Z}) \models \exists \bar{y} \forall \bar{x}\, \varphi' \tau(\bar{x}, \bar{y})$. By the definition of an interval formula equation, no atom or term in $\varphi'$ contains more than one of the $x_i$, and this property is preserved by the substitution $\tau$. Therefore, we can apply Theorem

5.19 to $\forall \bar{x}\, \varphi' \tau$ and obtain a $\mathrm{Th}(\mathbb{Z})$-equivalent formula $\psi_\tau(\bar{y}) \equiv \exists \bar{z}\, \psi'_\tau(\bar{y}, \bar{z})$ with $\psi'_\tau$ quantifier-free. It follows that $\varphi$ has a solution iff for some configuration $\tau$ of $\varphi$, $\mathrm{Th}(\mathbb{Z}) \models \exists \bar{y} \exists \bar{z}\, \psi'_\tau(\bar{y}, \bar{z})$. Since there are only finitely many configurations of $\varphi$, deciding whether a solution of $\varphi$ exists amounts to deciding whether $\mathrm{Th}(\mathbb{Z})$ entails one of finitely many $\Sigma_1$ formulas. This is possible by Corollary 5.23. □

## 5.3 An upper bound on polyhedral solution problems

Finally, we will present a generalization of the polyhedral solution problem that can be shown to be undecidable. This section is based on David Monniaux's paper [Mon19]. There, the author reduces the halting problem for register machines to the problem of deciding whether a register machine has a separating invariant. We adapt his proof into a reduction of the halting problem to a solution problem and make the geometric considerations in the original proof more explicit. For this reason, we will once again need some geometric and topological notions. Since the only topological space we will be dealing with is $\mathbb{Q}^n$, we do not need to state them in their most general forms. Let $M \subset \mathbb{Q}^n$. $M$ is called *open* if it contains a ball around each of its elements. The *interior* $M^\circ$ of $M$ is the greatest open subset of $M$. It always exists because the union of open sets is open. The *boundary* $\partial M$ of $M$ is the set of all $x \in \mathbb{Q}^n$ such that every ball around $x$ intersects both $M$ and $M^c$. Clearly, $M^\circ \cap \partial M = \emptyset$.

Equipped with these definitions, we can define what it means for a set to be *strictly convex*: $M$ is strictly convex if it is convex and moreover, for any two distinct points $p_1, p_2 \in \partial M$ and any $t \in (0,1) \cap \mathbb{Q}$ we have $tp_1 + (1-t)p_2 \in M^\circ$. In other words, the line segment connecting $p_1$ and $p_2$ does not intersect $\partial M$ except at its endpoints.

As an example, consider the filled parabola

$$M = \left\{ (x, y) \in \mathbb{Q}^2 \;\middle|\; y \geq ax^2 + b + c \right\}$$

where $a > 0$. Then

$$M^\circ = \left\{ (x, y) \in \mathbb{Q}^2 \;\middle|\; y > ax^2 + b + c \right\},$$
$$\partial M = \left\{ (x, y) \in \mathbb{Q}^2 \;\middle|\; y = ax^2 + b + c \right\}.$$

It is easy to check that $M$ is in fact strictly convex.

The generalization of affine and polyhedral solution problems that we investigate in this section amounts to allowing multiplication of variables. In fact, we will need only a single term containing a squared variable.

**Definition 5.24** ($\mathcal{L}^{\mathbb{Q}}_{\leq,\times}, \mathrm{Th}(\mathbb{Q}^\times)$)**.** $\mathcal{L}^{\mathbb{Q}}_{\leq,\times}$ is the language $\mathcal{L}^{\mathbb{Q}}_{\leq}$ extended by the binary function symbol $\cdot$. $\mathrm{Th}(\mathbb{Q}^\times)$ is the theory of $\mathbb{Q}$ over $\mathcal{L}^{\mathbb{Q}}_{\leq,\times}$ with the symbols of $\mathcal{L}^{\mathbb{Q}}_{\leq}$ interpreted as before and $\cdot$ interpreted as multiplication.

**Definition 5.25** (Generalized polyhedral solution problem)**.**

1. A generalized polyhedral formula equation is a $\Pi_1$ formula equation over $\mathrm{Th}(\mathbb{Q}^\times)$.
2. The *generalized polyhedral solution problem* is the solution problem $\langle \mathrm{Th}(\mathbb{Q}^\times), \Phi, \mathcal{C} \rangle$, where $\mathcal{C}$ is the class of generalized polyhedral formula equations and $\Phi$ is the class of integral polyhedral formulas.

**Theorem 5.26.** *The generalized polyhedral solution problem is undecidable.*

We will prove Theorem 5.26 by reducing the halting problem for a suitable kind of program to the generalized polyhedral solution problem.

**Definition 5.27** (Rational program)**.** A *rational program* is a tuple $P = \langle (x_1, ..., x_n), S, \sigma_0, K, \Lambda \rangle$ where

1. $x_1, ..., x_n$ are the *program variables*;
2. $S$ is a finite set of *states*;
3. $\sigma_0 \in S$ is the *starting state*;
4. $K = \left\{ \kappa^{\sigma'}_{\sigma} \;\middle|\; \sigma, \sigma' \in S \right\}$ is a set of quantifier-free formulas such that $\mathrm{Th}(\mathbb{Q}^\times) \vDash \forall \bar{x}\, \neg(\kappa^{\sigma'}_{\sigma}(\bar{x}) \wedge \kappa^{\sigma''}_{\sigma}(\bar{x}))$ for $\sigma' \neq \sigma''$;
5. $\Lambda = \left\{ \lambda^{\sigma'}_{\sigma} \;\middle|\; \sigma, \sigma' \in S \right\}$ is a set of affine transformations on $\mathbb{Q}^n$.

$K$ and $\Lambda$ induce a *partial transition function* $\tau\colon S \times \mathbb{Q}^n \hookrightarrow S \times \mathbb{Q}^n$:

$$\tau(\sigma, \bar{x}) = \begin{cases} (\sigma', \lambda_\sigma^{\sigma'}(\bar{x})), & \text{if } \mathrm{Th}(\mathbb{Q}^\times) \vDash \kappa_\sigma^{\sigma'}(\bar{x}), \\ \text{undefined}, & \text{if } \mathrm{Th}(\mathbb{Q}^\times) \nvDash \kappa_\sigma^{\sigma'} \text{ for all } \sigma' \in S. \end{cases}$$

$\tau$ is well-defined because of the condition on $K$.

$P$ terminates on an input $\bar{x}_0$ if $\tau^i(\sigma_0, \bar{x}_0)$ is undefined for some $i \in \mathbb{N}$.

**Example 5.28.** Let $P = \langle (x, y, p), \{\sigma, \rho\}, \sigma, \{\kappa_\sigma^\sigma, \kappa_\sigma^\rho, \kappa_\rho^\sigma, \kappa_\rho^\rho\}, \{\lambda_\sigma^\sigma, \lambda_\sigma^\rho, \lambda_\rho^\sigma, \lambda_\rho^\rho\} \rangle$, where

$$\begin{aligned} \kappa_\sigma^\sigma(x, y, p) &\equiv \bot, & \kappa_\sigma^\rho(x, y, p) &\equiv \top, \\ \kappa_\rho^\sigma(x, y, p) &\equiv \bot, & \kappa_\rho^\rho(x, y, p) &\equiv x \geq 1, \\ \lambda_\sigma^\sigma &= \mathrm{id}, & \lambda_\sigma^\rho(x, y, p) &= (x, y, 0), \\ \lambda_\rho^\sigma &= \mathrm{id}, & \lambda_\rho^\rho(x, y, p) &= (x - 1, y, p + y). \end{aligned}$$

It might be written in pseudocode as

**Input:** Rational numbers $x$, $y$, $p$ ▷ State $\sigma$ is here.
1: $p := 0$
2: **while** $x \geq 1$ **do** ▷ State $\rho$ is here.
3: $\quad x := x - 1$ ▷ These two assignments form $\lambda_\rho^\rho$.
4: $\quad p := p + y$
5: **end while**

It is easy to verify that $P$ terminates for any input $(x_0, y_0, p_0)$ and the last value of $p$ is $\max(0, \lfloor x_0 \rfloor) \cdot y_0$.

The halting problem for rational programs is clearly undecidable since register machines can be straightforwardly encoded as rational programs.

The proof of Theorem 5.26 also involves a geometric object that we will describe in some detail.

**Definition 5.29** ($\mathcal{P}_n$)**.** Let $n \in \mathbb{N}$. Then $\mathcal{P}_n \subseteq \mathbb{Q}^{n+2}$ is the set

$$\left\{ (x_1, ..., x_n, y, z) \in \mathbb{Q}^{n+2} \;\middle|\; z \geq \frac{y^2 + y}{2} \right\}.$$

Geometrically, $\mathcal{P}_n$ is a parabolic cylinder in $\mathbb{Q}^{n+2}$. We will make use of the following properties of $\mathcal{P}$:

1. $\mathcal{P}_n$ is convex.
2. The boundary and interior of $\mathcal{P}_n$ are

$$\partial\mathcal{P}_n = \left\{ (\bar{x}, y, z) \in \mathbb{Q}^{n+2} \;\middle|\; 2z = y^2 + y \right\},$$
$$\mathcal{P}_n^\circ = \left\{ (\bar{x}, y, z) \in \mathbb{Q}^{n+2} \;\middle|\; 2z > y^2 + y \right\}.$$

3. $\mathcal{P}_n$ is not strictly convex, but it does have a weaker version of the property that is sufficient for our purposes. Let $p_1 = (\bar{x}_1, y_1, z_1), p_2 = (\bar{x}_2, y_2, z_2) \in \partial\mathcal{P}_n$ with $y_1 \neq y_2$ and $t \in (0,1)$. Moreover, let $\pi = (\bar{x}, y, z) \mapsto (y, z)$ be the projection onto the last two coordinates. Because the parabola $\pi(\mathcal{P}_n)$ is strictly convex and because $\pi$ maps open sets to open sets, we have

$$\pi(tp_1 + (1-t)p_2) = t\pi(p_1) + (1-t)\pi(p_2) \in \pi(\mathcal{P}_n)^\circ = \pi(\mathcal{P}_n^\circ),$$

   and since $\pi^{-1}(\pi(\mathcal{P}_n^\circ)) = \mathcal{P}_n^\circ$, it follows that $tp_1 + (1-t)p_2 \in \mathcal{P}_n^\circ$. In other words, if $p_1$ and $p_2$ are distinct points on the boundary of $\mathcal{P}$ with different $n+1$-th coordinates, then their connecting line does not intersect the boundary except at its endpoints.
4. As an immediate consequence of the previous point, if $M \subseteq \partial\mathcal{P}_n$ such that the $n+1$-th coordinates of all elements of $M$ are pairwise distinct, then $\operatorname{conv} M \cap \partial\mathcal{P}_n = M$.

*Proof of Theorem 5.26.* The proof proceeds by reduction from the halting problem for rational programs. Let $P = \langle (x_1, ..., x_n), S, \sigma_0, K, \Lambda \rangle$ be a rational program and $\tau$ its transition function. Let $\bar{x}_0 \in \mathbb{Q}^n$. We will construct an instance $\exists \bar{I}\, \varphi$ of the generalized polyhedral solution problem that has a solution iff $P$ halts on $\bar{x}_0$.

Let $\sigma_b$ be a new state not in $S$ and $S' = S \dot{\cup} \{\sigma_b\}$. Moreover, let there be a formula variable $I_\sigma(x_1, ..., x_n, y, z)$ for all $\sigma \in S'$. For every $\sigma, \sigma' \in S$, let

$$\begin{aligned}\varphi_\sigma^{\sigma'}(\bar{x}, y, z) \equiv\ & I_\sigma(\bar{x}, y, z) \\ & \wedge \kappa_\sigma^{\sigma'}(\bar{x}) \\ & \wedge z = (y^2 + y)/2 \\ & \to I_{\sigma'}(\lambda_\sigma^{\sigma'}(\bar{x}), y+1, z+y+1)\end{aligned}$$

and

$$\varphi_\sigma^{\sigma_b}(\bar{x}, y, z) \equiv I_\sigma(\bar{x}, y, z) \wedge z < (y^2 + y)/2 \to I_{\sigma_b}(\bar{x}, i, y).$$

Now let

$$\begin{aligned} \varphi \equiv \forall \bar{x} \forall y \forall z. \quad & I_{\sigma_0}(\bar{x}_0, 0, 0) \\ & \wedge \neg I_{\sigma_b}(\bar{x}, y, z) \\ & \wedge \bigwedge_{\substack{\sigma \in S \\ \sigma' \in S'}} \varphi_\sigma^{\sigma'}(\bar{x}, y, z). \end{aligned}$$

Furthermore, we introduce the abbreviation $(\sigma_i, \bar{x}_i)$ for $\tau^i(\sigma_0, \bar{x}_0)$.

Now assume that $P$ terminates on $\bar{x}_0$. This means that there is some $k \in \mathbb{N}$ such that $\tau(\sigma_k, \bar{x}_k)$ is undefined. For each $\sigma \in S$, we define

$$\tilde{J}_\sigma = \left\{ (\bar{x}, i, (i^2 + i)/2) \in \mathbb{Q}^{n+2} \;\middle|\; \sigma = \sigma_i \wedge \bar{x} = \bar{x}_i \right\}.$$

A point $(\bar{x}, i, z)$ is in $\tilde{J}_\sigma$ iff after $i$ execution steps $P$ is in state $\sigma$, its program variables have the values $\bar{x}$, and $z = (i^2 + i)/2$. Because of the latter equation, $\tilde{J}_\sigma \subseteq \partial\mathcal{P}_n$, and due to the convexity of $\mathcal{P}_n$, $\operatorname{conv} \tilde{J}_\sigma \subseteq \mathcal{P}_n$. Clearly, the $n+1$-th coordinates of all elements of $\tilde{J}_\sigma$ are pairwise distinct. We also show that $(\bar{x}, i, z) \in \operatorname{conv} \tilde{J}_\sigma \cap \partial\mathcal{P}_n$ iff $z = (i^2 + i)/2$ and $(\sigma, \bar{x}) = (\sigma_i, \bar{x}_i)$. For the forward direction, assume $(\bar{x}, i, z) \in \operatorname{conv} \tilde{J}_\sigma \cap \partial\mathcal{P}_n$. Then $z = (i^2 + i)/2$ is obvious. Because of 4, $(\bar{x}, i, z) \in \tilde{J}_\sigma$ and it follows immediately from the definition of $\tilde{J}_\sigma$ that $(\sigma, \bar{x}) = (\sigma_i, \bar{x}_i)$. Conversely, assume $z = (i^2 + i)/2$ and $(\sigma, \bar{x}) = (\sigma_i, \bar{x}_i)$. The first equation implies $(\bar{x}, i, z) \in \partial\mathcal{P}_n$ and the second implies $(\bar{x}, i, z) \in \tilde{J}_\sigma \subseteq \operatorname{conv} \tilde{J}_\sigma$.

Now, for each $\sigma \in S$, let $J_\sigma(\bar{x}, y, z)$ be a H-representation of $\operatorname{conv} \tilde{J}_\sigma$. Then by the previous argument, we have

$$\operatorname{Th}(\mathbb{Q}^\times) \models \forall \bar{x} \forall i \forall z.\ J_\sigma(\bar{x}, i, z) \to z \geq (i^2 + i)/2. \tag{5.1}$$

Moreover, let $J_{\sigma_b} \equiv \bot$. We claim that $\vartheta = [(I_\sigma \backslash J_\sigma)_{\sigma \in S'}]$ is a solution of $\varphi$. We need to show

$$\begin{aligned} &\operatorname{Th}(\mathbb{Q}^\times) \models J_{\sigma_0}(x_0, 0, 0), \\ &\operatorname{Th}(\mathbb{Q}^\times) \models \forall \bar{x} \forall i \forall z\ \neg J_{\sigma_b}(\bar{x}, i, z), \\ &\operatorname{Th}(\mathbb{Q}^\times) \models \forall \bar{x} \forall i \forall z\ \varphi_\sigma^{\sigma'} \vartheta(\bar{x}, i, z) \text{ for all } \sigma \in S, \sigma' \in S'. \end{aligned}$$

The first line holds because $(\bar{x}_0, 0, 0) \in \tilde{J}_{\sigma_0}$ and the second line is immediate. For the third line, let $\sigma \in S$ and $\sigma' \in S'$. If $\sigma' = \sigma_b$, then consider

$$\forall \bar{x} \forall i \forall z \, \varphi_\sigma^{\sigma_b} \vartheta(\bar{x}, i, z) \equiv \forall \bar{x} \forall i \forall z. \; J_\sigma(\bar{x}, i, z) \wedge z < (i^2 + i)/2 \rightarrow J_{\sigma_b}(\bar{x}, i, z).$$

From (5.1) we deduce $J_\sigma(\bar{x}, i, z) \wedge z < (i^2+i)/2 \rightarrow \bot$, so $\mathrm{Th}(\mathbb{Q}^\times) \models \forall \bar{x} \forall i \forall y \, \varphi_\sigma^{\sigma_b} \vartheta(\bar{x}, i, y)$. On the other hand, if $\sigma' \neq \sigma_b$, then $\sigma' \in S$ and we consider

$$\begin{aligned} \forall \bar{x} \forall i \forall z \, \varphi_\sigma^{\sigma'} \vartheta(\bar{x}, i, z) \equiv \forall \bar{x} \forall i \forall z. \; & J_\sigma(\bar{x}, i, z) \\ & \wedge \kappa_\sigma^{\sigma'}(\bar{x}) \\ & \wedge z = (i^2 + i)/2 \\ & \rightarrow J_{\sigma'}(\lambda_\sigma(\bar{x}), i + 1, y + i + 1). \end{aligned}$$

Assume $J_\sigma(\bar{x}, i, z)$, $\kappa_\sigma^{\sigma'}(\bar{x})$, and $z = (i^2+i)/2$. As we have remarked previously, this means that $(\sigma, \bar{x}) = (\sigma_i, \bar{x}_i)$. Recall that $\tau(\sigma, \bar{x}) = (\sigma', \lambda_\sigma(\bar{x}))$ iff $\mathrm{Th}(\mathbb{Q}^\times) \models \kappa_\sigma^{\sigma'}(\bar{x})$. It follows that

$$(\sigma_{i+1}, x_{i+1}) = \tau(\sigma, \bar{x}) = (\sigma', \lambda_\sigma(\bar{x})),$$

so $(\lambda_\sigma(\bar{x}), i + 1, z + i + 1) \in \tilde{J}_{\sigma'}$ and hence $J_{\sigma'}(\lambda_\sigma(\bar{x}), i + 1, z + i + 1)$.

For the other direction, suppose that $\varphi$ has a solution $\vartheta = [(I_\sigma \backslash J_\sigma)_{\sigma \in S'}]$. From $\mathrm{Th}(\mathbb{Q}^\times) \models \varphi\vartheta$ we immediately obtain $\mathrm{Th}(\mathbb{Q}^\times) \models \forall \bar{x} \forall i \forall z \neg J_{\sigma_b}(\bar{x}, i, z)$. Observe that for $\sigma \in S$, $\mathrm{Th}(\mathbb{Q}^\times) \models \forall \bar{x} \forall i \forall z. \; J_\sigma(\bar{x}, i, z) \rightarrow z \geq (i^2 + i)/2$. This is the case because otherwise, $\mathrm{Th}(\mathbb{Q}^\times) \models \exists \bar{x} \exists i \exists z. \; J_\sigma(\bar{x}, i, z) \wedge z < (i^2 + i)/2$, and we could deduce $\mathrm{Th}(\mathbb{Q}^\times) \models \exists \bar{x} \exists i \exists z \, J_{\sigma_b}(\bar{x}, i, z)$, a contradiction. Now assume that $P$ does not terminate. Then there is a state $\sigma \in S$ that is reached infinitely often during the execution of $P$, that is, there is an infinite sequence $i_0 < i_1 < \ldots$ such that $\sigma_{i_j} = \sigma$. It is easy to see by an inductive argument that for each $i_j$, $\mathrm{Th}(\mathbb{Q}^\times) \models J_\sigma(\bar{x}_{i_j}, i_j, (i_j^2 + i_j)/2)$. This means that the interpretation $\tilde{J}_\sigma$ of $J_\sigma$ contains the infinitely many points $(\bar{x}_{i_j}, i_j, (i_j^2 + i_j)/2) \in \partial\mathcal{P}$. Such a point cannot be the convex combination of other points of $\tilde{J}_\sigma$ due to Property 4 and must therefore be a vertex of $\tilde{J}_\sigma$. This implies that $\tilde{J}_\sigma$ is a convex polyhedron with infinitely many vertices, which is a contradiction. Hence, $P$ must terminate. □

# CONCLUSION

Let us take a look back at what we have accomplished. Our goal in this thesis was to demonstrate the expressive power of formula equations and make the case for using them to express problems of inductive theorem proving and invariant generation.

The original hypothesis was that instances of these two problems, when formulated as formula equations, would be structured similarly enough for it to be possible to use algorithms for invariant generation as black boxes to solve induction problems. We saw in Chapter 3 that this is not the case—not because of any technical challenges, but fundamentally. Let us recapitulate the argument. Induction problems will, in general, involve multiple terms in the base case and the step. This obviously suggests using a nondeterministic program formalism. While notion such as precondition, postcondition, and invariant are well-defined for such nondeterministic programs, the Hoare-triple $\{\varphi\}p\{\psi\}$ in this case expresses the fact that if $\varphi$ holds before $p$, then $\psi$ holds after *every* possible execution of $p$ (a universal property). The key insight is that the nondeterminism in induction problems is of a different nature; the property we require in this case is that if $\varphi$ holds before $p$, there is some execution of $p$ after which $\psi$ holds (an existential property). Unlike the universal property above, this is not expressible in Hoare logic. Therefore, we turn to dynamic logic, a modal logic that generalizes Hoare logic and is powerful enough for our purposes. In dynamic logic, the universal property is expressed as

$\varphi \to [p]\psi$ and the existential one as $\varphi \to \langle p\rangle\psi$. Thus, we obtain the result that for every simple induction proof schema $\pi$ there is a program $p$ such that induction formulas of $\pi$ correspond precisely to "invariants" needed to prove a formula of the form $\varphi \to [p]\psi$.

In Chapter 4, we investigated a specific kind of formula equation, namely affine formula equations. Since using loop invariant procedures without modification does not work, we approached the problem from a different angle: taking a loop invariant generation procedure and adapting it to our needs. The algorithm in question finds the strongest affine invariants in programs with only affine assignments and no control flow structures. We first transformed a given affine formula equation step by step into a set of affine conditions, which jointly form a statement about affine subspaces of some $\mathbb{Q}^n$ and their behavior under affine transformations. This translation required the insight that any covering of an affine space by countably many affine spaces is trivial in the sense that it has a subcovering consisting of just one space. Because of the finite height of the lattice of affine subspaces of $\mathbb{Q}^n$, the question of whether subspaces satisfying a set of affine conditions exist is amenable to a solution by fixed point iteration.

While one may hope that the approach of Chapter 4 can be applied to other kinds of formula equations, Chapter 5 provides an example of a case where that is not possible. There, we considered inequalities over the integers and the rational numbers. This means that on the geometric level, the sets under consideration were convex polyhedra. Since convex polyhedra do not have the covering property of affine spaces, formula equations involving inequalities cannot be decomposed in the same way as affine formula equations, and because convex polyhedra admit infinite ascending chains, an iteration procedure is not guaranteed to terminate. For these reasons, we investigated several more specific problems. First, we obtained a decidability result for the simple case of rotations in the rational plane and formula equations containing only a single formula variable. Next, we treated the case of products of intervals over the integers, also showing it to be decidable. Finally, we showed, following [Mon19], that polyhedral solution problems over the rational numbers are undecidable if multiplication is admitted.

A number of questions remain open for future research. Chief among them, in our opinion, are the problem of finding classes of formula equations amenable to

a fixed point treatment like the one we used for affine formula equations, and the question of the decidability of the polyhedral solution problem.

# BIBLIOGRAPHY


[Ack35] Wilhelm Ackermann, *Untersuchungen über das Eliminationsproblem der mathematischen Logik*, Mathematische Annalen **110** (1935), no. 1, 390–413.

[Baa97] Franz Baader, *On the complexity of Boolean unification*, LTCS-Report LTCS-97-03, LuFg Theoretical Computer Science, RWTH Aachen, Germany, 1997.

[BGMR15] Nikolaj Bjørner, Arie Gurfinkel, Ken McMillan, and Andrey Rybalchenko, *Horn clause solvers for program verification*, pp. 24–51, Springer International Publishing, Cham, 2015.

[BHRZ03] Roberto Bagnara, Patricia Hill, Elisa Ricci, and Enea Zaffanella, *Precise widening operators for convex polyhedra*, Science of Computer Programming - SCP, vol. 58, 01 2003, pp. 337–354.

[BM07] Aaron R. Bradley and Zohar Manna, *The Calculus of Computation*, Springer, 2007.

[BN98] Franz Baader and Tobias Nipkow, *Term rewriting and all that*, Cambridge University Press, 1998.

[BS87] Wolfram Büttner and Helmut Simonis, *Embedding Boolean expressions into logic programming*, Journal of Symbolic Computation **4** (1987), no. 2, 191–205.

[CC77] Patrick Cousot and Radhia Cousot, *Abstract interpretation: A unified lattice model for static analysis of programs by construction or approximation of fixpoints*, Conference Record of the Fourth Annual ACM Symposium on Principles of Programming Languages, ACM Press, 1977, pp. 238–252.

[CH78] Patrick Cousot and Nicolas Halbwachs, *Automatic discovery of linear restraints among variables of a program*, Conference Record of the Fifth Annual ACM Symposium on Principles of Programming Languages, ACM Press, 1978, pp. 84–96.

[EH15] Sebastian Eberhard and Stefan Hetzl, *Inductive theorem proving based on tree grammars*, Annals of Pure and Applied Logic **166** (2015), no. 6, 665–700.

[EHW17] Sebastian Eberhard, Stefan Hetzl, and Daniel Weller, *Boolean unification with predicates*, Journal of Logic and Computation **27** (2017), 109–128.

[End02] Herbert B. Enderton, *A mathematical introduction to logic*, second ed., Harcourt/Academic Press, 2002.

[HTK00] David Harel, Jerzy Tiuryn, and Dexter Kozen, *Dynamic logic*, MIT Press, Cambridge, MA, USA, 2000.

[HW18] Stefan Hetzl and Tin Lok Wong, *Some observations on the logical foundations of inductive theorem proving*, Logical Methods in Computer Science **13** (2018), no. 4.

[HZ19] Stefan Hetzl and Sebastian Zivota, *Decidability of affine solution problems*, Journal of Logic and Computation **30** (2019), no. 3, 697–714.

[Kar76] Michael Karr, *Affine relationships among variables of a program*, Acta Inf. **6** (1976), 133–151.

[KKR90] Paris C. Kanellakis, Gabriel M. Kuper, and Peter Z. Revesz, *Constraint query languages*, Proceedings of the Ninth Symposium on Principles of Database Systems (PODS) (Daniel J. Rosenkrantz and Yehoshua Sagiv, eds.), ACM Press, 1990, pp. 299–313.

[KKR95] ———, *Constraint query languages*, Journal of Computer and System Sciences **51** (1995), no. 1, 26–52.

[KS08] Daniel Kroening and Ofer Strichman, *Decision procedures*, Springer, 2008.

[MN88] Ursula Martin and Tobias Nipkow, *Unification in Boolean rings*, Journal of Automated Reasoning **4** (1988), no. 4, 381–396.

[MN89] Ursula Martin and Tobias Nipkow, *Boolean Unification – The Story So Far*, Journal of Symbolic Computation **7** (1989), no. 3-4, 275–293.

[Mon19] David Monniaux, *On the decidability of the existence of polyhedral invariants in transition systems*, Acta Informatica **56** (2019), no. 4, 385–389.

[MOS04] Markus Müller-Olm and Helmut Seidl, *Precise interprocedural analysis through linear algebra*, Proceedings of the 31st ACM SIGPLAN-SIGACT symposium on Principles of programming languages (New York, NY, USA), POPL '04, ACM, 2004, pp. 330–341.

[Sch95] Ernst Schröder, *Vorlesungen über die algebra der logik*, Teubner, vol. 1, 1890; vol. 2, pt. 1, 1891; vol.2, pt. 2, 1905; vol. 3, 1895.

[Sip12] Michael Sipser, *Introduction to the theory of computation*, third ed., Cengage Learning, 2012.

[ST71] Ernst Snapper and Robert J. Troyer, *Metric affine geometry*, Academic Press, 1971.

[Tak87] Gaisi Takeuti, *Proof theory*, second ed., Studies in Logic and the Foundations of Mathematics, vol. 81, North-Holland Publishing Co., Amsterdam, 1987, With an appendix containing contributions by Georg

Kreisel, Wolfram Pohlers, Stephen G. Simpson and Solomon Feferman. MR 882549 (89a:03115)

[Tan96] Lin Tan, *The group of rational points on the unit circle*, Mathematics Magazine **69** (1996), no. 3, 163–171.

[Tar55] Alfred Tarski, *A lattice-theoretical fixpoint theorem and its applications*, Pacific Journal of Mathematics **5** (1955), no. 2, 285–309.

[Wer17a] Christoph Wernhard, *The Boolean solution problem from the perspective of predicate logic*, 11th International Symposium on Frontiers of Combining Systems, FroCoS 2017 (Clare Dixon and Marcelo Finger, eds.), LNCS (LNAI), vol. 10483, Springer, 2017, pp. 333–350.

[Wer17b] ———, *The Boolean solution problem from the perspective of predicate logic - extended version*, CoRR **abs/1706.08329** (2017).

[WN99] Laurence Wolsey and George Nemhauser, *Integer and combinatorial optimization*, Wiley-Interscience, 1999.

[Zie12] Günter M. Ziegler, *Lectures on polytopes*, Graduate Texts in Mathematics, Springer New York, 2012.