PDF Download
3779416.pdf
25 March 2026
Total Citations: 0
Total Downloads: 329

Published: 20 March 2026
Online AM: 15 December 2025
Accepted: 29 October 2025
Revised: 16 March 2026
Received: 15 February 2022

Citation in BibTeX format

RESEARCH-ARTICLE

An Abstract Fixed-Point Theorem for Horn Formula Equations

STEFAN HETZL, Vienna University of Technology, Vienna, Vienna, Austria

JOHANNES KLOIBHOFER, University of Amsterdam, Amsterdam, Noord-Holland, Netherlands

Open Access Support provided by:

University of Amsterdam

Vienna University of Technology

An Abstract Fixed-Point Theorem for Horn Formula Equations

STEFAN HETZL, Institute of Discrete Mathematics and Geometry, TU Wien, Wien, Austria
JOHANNES KLOIBHOFER, Institute for Logic, Language and Computation, University of Amsterdam, Amsterdam, The Netherlands

We consider a class of formula equations in first-order logic, Horn formula equations, which are defined by a syntactic restriction on the occurrences of predicate variables. Horn formula equations play an important role in many applications in computer science. We state and prove a fixed-point theorem for Horn formula equations in first-order logic with a least fixed-point operator. Our fixed-point theorem is abstract in the sense that it applies to an abstract semantics which generalises standard semantics. We describe several corollaries of this fixed-point theorem in various areas of computational logic, ranging from the logical foundations of program verification to inductive theorem proving.

CCS Concepts: • **Theory of computation** → **Logic and verification**; *Constraint and logic programming*; Automated reasoning; Hoare logic;

Additional Key Words and Phrases: Horn formulas, fixed-point logic, program verification

ACM Reference format:
Stefan Hetzl and Johannes Kloibhofer. 2026. An Abstract Fixed-Point Theorem for Horn Formula Equations. *ACM Trans. Comput. Logic* 27, 2, Article 9 (March 2026), 35 pages.
https://doi.org/10.1145/3779416

1 Introduction

Solving Boolean equations is one of the oldest and one of the most central problems of logic. It goes back to the 19th century and was already thoroughly investigated in [56]. It has connections and applications throughout computational logic: It can be understood as a variant of unification, see, e.g., [47]. It is strongly related to second-order quantifier elimination and thus has many applications in areas such as modal logic, knowledge representation, and common-sense reasoning, see, e.g., [22]. It is suitable as a logical foundation for software verification via constrained Horn clause solving, see, e.g., [8].

We are interested in formula equations in first-order logic. A formula equation is simply an existential second-order formula, such as:

$$\exists X\ (X(2) \land \forall u\ (X(u) \to X(u+u)) \land \neg X(3)). \tag{*}$$

Solving this equation means finding a first-order formula $\varphi(v)$ such that, when $X(v)$ is replaced by $\varphi(v)$, a valid first-order formula is obtained. In this example, the formula $\exists w\ w + w = v$, expressing

Authors' Contact Information: Stefan Hetzl (corresponding author), Institute of Discrete Mathematics and Geometry, TU Wien, Wien, Austria; e-mail: stefan.hetzl@tuwien.ac.at; Johannes Kloibhofer, Institute for Logic, Language and Computation, University of Amsterdam, Amsterdam, The Netherlands; e-mail: j.kloibhofer@uva.nl.

ACM 1557-945X/2026/3-ART9
https://doi.org/10.1145/3779416

that v is even, would be a solution (assuming some basic background theory of natural number arithmetic).

In this paper, we consider a certain class of formula equations, Horn formula equations. Horn formula equations are characterised by the syntactic restriction of being a clause set where every clause contains at most one positive occurrence of a predicate variable (just as the above example (*) does). In the context of verification, Horn formula equations are known as constrained Horn clause sets. It is known that solving a Horn formula equation is strongly related to the computation of a least fixed point. In our example (*), the definite clauses, i.e., those with at least one positive occurrence of the predicate variable X, define the mapping $F : \mathcal{P}(\mathbb{N}) \to \mathcal{P}(\mathbb{N})$, $S \mapsto \{2\} \cup \{2 \cdot n \mid n \in S\}$. The least fixed point of F is $P = \{2^m \mid m \geq 1\}$. By inserting P for X, the definite clauses are satisfied trivially. Thus, (*) is seen to be equivalent to the conjunction of the remaining clauses, i.e., those that do not contain a positive occurrence of X, satisfying P. In our case, this amounts to the statement $3 \notin P$. This is true, so (*) is true.

This solution P, the set of powers of two, is known to be undefinable in the original language of the problem: that of linear arithmetic [18]. Another solution of (*) is $E = \{n \geq 2 \mid n \text{ is even}\}$ which is definable in linear arithmetic (by the formula $\exists y\, y + y = x$). The question of the expressivity of solutions in a restricted language is central for solving formula equations, and, in a wider context, for inductive theorem proving and software verification where the solutions of formula equations correspond to induction and loop invariants, respectively. The technique of abstract interpretation, introduced in [12], is a flexible and powerful approach to dealing with such situations in software verification. In this paper, we introduce model abstractions, a notion of abstract Tarski model which corresponds to abstract interpretation and thus allows to deal with this phenomenon in the context of formula equations.

In this paper, we formulate and prove a fixed point theorem for Horn formula equations which, in its simplest form, states the following: Every Horn formula equations $\exists X\, \varphi(X)$ has a solution of the form $\mathrm{lfp}_X\, \Phi(X)$ and, moreover, this is the least solution. Here, $\Phi(X)$ is a formula that defines an operator and is induced by $\varphi(X)$. The mere fact that this is true is well known in constrained Horn clause solving and in logic programming. Our main contributions are: (1) We formulate and prove this result in a logic with an explicit fixed point operator in a general way that encompasses both, simultaneous least fixed points and abstract interpretation. (2) We show that this formulation with an explicit least fixed point operator is useful for a wide range of different applications. In fact, about half of this paper is devoted to discussing these different applications: as a simple corollary, one can obtain the expressibility of the weakest precondition and the strongest postcondition and thus the partial correctness of an imperative program in FO[LFP] without expressivity hypothesis. It permits to considerably simplify the proof of the decidability of affine formula equations [30]. As another corollary, it allows a generalisation of a result by Ackermann [1] on second-order quantifier-elimination in a direction different from the recent generalisation [63]. A result from a recently introduced approach to inductive theorem proving with tree grammars described in [16] can also be obtained from our fixed-point theorem as another straightforward corollary.

This paper is structured as follows: in Section 2, we introduce basic technical notions and results. In Section 3, we introduce and discuss formula equations and, in particular, Horn formula equations. The abstract fixed-point theorem is stated and proved in detail in Section 4. In Section 5, we obtain the decidability of the affine solution problem as corollary of the abstract fixed-point theorem. Section 6 describes the applications of our fixed-point theorem to the logical foundations of program verification, in particular the definability of the weakest precondition and the strongest postcondition in FO[LFP] without expressivity hypothesis. In Section 7, we use it to approximate certain second-order formulas, and in Section 8, we obtain a result about a method for inductive theorem proving based on tree grammars [16] as corollary of the fixed-point theorem. In Section 9, we discuss the related work.

Many of the results of this paper originate from the second author's master's thesis [41]. A concise presentation of the main results of the master thesis, in particular the fixed-point theorem for standard semantics, can be found in [28]. The abstract version of the fixed-point theorem and its corollaries have been obtained later and have been announced in the abstract [29]. In this paper, we give a full proof of the abstract fixed-point theorem and describe its applications in detail.

2 Preliminaries

2.1 Notations

A *language* $\mathcal{L}$ consists of constant, predicate and function symbols. *Terms* over $\mathcal{L}$ are built from individual variables, constant and function symbols of $\mathcal{L}$. *First-Order (FO) formulas* over $\mathcal{L}$ are built from predicate symbols, terms, the logical connectives $\neg, \vee, \wedge$ and the quantifiers $\exists, \forall$ over individual variables. Moreover, we use the symbols $\rightarrow$ and $\leftrightarrow$, where $A \rightarrow B$ is defined to be an abbreviation for $\neg A \vee B$ and $A \leftrightarrow B$ to be an abbreviation for $A \rightarrow B \wedge B \rightarrow A$. In a *Second-Order (SO) formula*, in addition, there may occur predicate variables, which may be bound by universal or existential quantifiers. Predicate variables stand for predicates of a certain arity. To distinguish them from individual variables, we denote them with upper-case letters. Unless otherwise noted, we always talk about logic *with equality*, which means that we have a specified binary predicate symbol '=', which is interpreted as equality.

A subformula ψ of a formula φ occurs positively in φ if ψ occurs in the scope of an even number of negations in φ and occurs negatively if it occurs in the scope of an odd number of negations. For example, in the formula $\varphi := A \rightarrow B$, the subformula A occurs negatively and B occurs positively in φ, as φ is an abbreviation for $\neg A \vee B$. A predicate variable X *occurs only positively* in φ if every occurrence of X as a subformula in φ occurs positively. Conversely, X *occurs only negatively* if every occurrence of X as a subformula in φ occurs negatively.

For a formula φ, variables $x_1, ..., x_n$ and terms $t_1, ..., t_n$ we define $\varphi[x_1\backslash t_1, ..., x_n\backslash t_n]$ to be the formula φ, where every occurrence of x_j is replaced by t_j for every $j \in \{1, ..., n\}$ simultaneously. If a free variable in $t_1, ..., t_n$ also occurs as a bound variable in φ, then we consider the variant φ', where every bound variable which occurs in $t_1, ..., t_n$ is renamed. Thus, the substitution $\varphi[x_1\backslash t_1, ..., x_n\backslash t_n]$ is always defined, with possible renaming of bound variables. Similarly, for predicate variables $X_1, ..., X_n$ and formulas $\alpha_1, ..., \alpha_n$, we define $\varphi[X_1\backslash\alpha_1, ..., X_n\backslash\alpha_n]$ to be the formula φ, where every occurrence of X_j is replaced by α_j for every $j \in \{1, ..., n\}$. We use the usual vector notation, i.e., we write $\overline{X}$ for $X_1, ..., X_n$ if n is clear from the context or unimportant. Consistently, we write $\varphi[\overline{X}\backslash\overline{\alpha}]$ for $\varphi[X_1\backslash\alpha_1, ..., X_n\backslash\alpha_n]$.

An $\mathcal{L}$*-structure* is a pair $\mathcal{M} = (M, I)$, where M is a set and I is an interpretation of $\mathcal{L}$, i.e., $I(P) \subseteq M^k$ for a k-ary predicate symbol $P \in \mathcal{L}$ and $I(f) : M^k \rightarrow M$ for a k-ary function symbol $f \in \mathcal{L}$. An *environment* is an interpretation of free variables by elements of the structure. For an environment θ, a variable x and $m \in M$ we define $\theta[x := m]$ by $\theta[x := m](x) = m$ and $\theta[x := m](y) = \theta(y)$ for $x \neq y$. $\theta[X := S]$ is defined analogously for a k-ary predicate variable X and $S \subseteq M^k$. For a structure $\mathcal{M}$, an environment θ and a formula φ we define $\mathcal{M}, \theta \models \varphi$ as usual. In particular $\mathcal{M}, \theta \models \exists X\varphi(X)$ for a k-ary predicate variable X if there exists an $S \subseteq M^k$ such that $\mathcal{M}, \theta[X := S] \models \varphi(X)$. We also write $\mathcal{M}, \theta \models \varphi(S)$ as an abbreviation for $\mathcal{M}, \theta[X := S] \models \varphi(X)$, where X is a fresh predicate variable. Similarly, if $a \in M$, we write $\mathcal{M}, \theta \models \psi(a)$ as abbreviation for $\mathcal{M}, \theta[x := a] \models \psi(x)$. We define $\mathcal{M} \models \varphi$ if $\mathcal{M}, \theta \models \varphi$ for all environments θ. A formula is *valid*, written as $\models \varphi$, if $\mathcal{M} \models \varphi$ for all structures $\mathcal{M}$. We write $\varphi \equiv \psi$, if φ and ψ are logically equivalent, i.e. if $\models \varphi \leftrightarrow \psi$.

We also consider atomic *Least Fixed-Point (LFP)* formulas of the form:

$$[\mathrm{lfp}_R\ \varphi(R,\bar{x})](\bar{t}),$$

where $\varphi(R,\bar{x})$ is a first-order formula in $\mathcal{L} \cup \{R\}$, such that R occurs only positively in φ, R is k-ary and $\bar{t}$ is a k-tuple of terms. The semantics of an atomic LFP formula will be defined in the next subsection.

In this paper, we talk about the following formula classes:

(1) FO: Classical first-order logic.
(2) SO: Second-order logic.
(3) FO[LFP]: *Least fixed-point logic* is an extension of first-order logic, which in addition to the usual formation rules allows atomic LFP-formulas.
(4) SO[LFP]: *Second-order least fixed-point logic* is a syntactic extension of second-order logic, which additionally allows atomic LFP formulas.

Example 1. Let $\mathcal{L} = \{E, s\}$ be the language of graphs with edge relation E and one specified vertex s. Then:

(1) $\varphi(R,x) := E(s,x) \lor \exists y(R(s,y) \land E(y,x))$ is an FO-formula, where R is a binary predicate variable.
(2) $\forall R(\varphi(R,x) \to E(s,x))$ is an SO-formula.
(3) $\psi(y) := [\mathrm{lfp}_R\ \varphi(R,x)](y)$ is an FO[LFP]-formula.
(4) $\exists P(\psi(y) \leftrightarrow \neg P(y))$ is an SO[LFP]-formula.

Monotonicity is a key property for defining least fixed-points. Before we define the semantics of atomic least fixed-point formulas, we state a simple monotonicity lemma that will be useful at several occasion later on.

Lemma 2. *Let $\mathcal{L}$ be a language and $\mathcal{M}$ be an $\mathcal{L}$-structure. Let R be a predicate variable and let S and S' be relations in $\mathcal{M}$ with the same arity as R. Let φ be a first-order formula in $\mathcal{L} \cup \{R\}$, such that R occurs only positively in φ. Then:*

$$\mathcal{M} \models S(\bar{x}) \to S'(\bar{x}) \quad \Rightarrow \quad \mathcal{M} \models \varphi[R\backslash S] \to \varphi[R\backslash S'].$$

Conversely, if R occurs only negatively in φ, then:

$$\mathcal{M} \models S(\bar{x}) \to S'(\bar{x}) \quad \Rightarrow \quad \mathcal{M} \models \varphi[R\backslash S'] \to \varphi[R\backslash S].$$

2.2 Fixed-Point Logics

2.2.1 Semantics of Least Fixed-Point Formulas. For the semantics of least fixed-point logic, we need some background on fixed points, which is best presented in the context of complete lattices.

Definition 3. Let $(E, \leq)$ be a complete lattice.

(1) A function $f : E \to E$ is called an *operator* on E.
(2) f is called *monotone* if for all $x \leq y$ it holds that $f(x) \leq f(y)$.
(3) An element x is a *fixed point* of f if $f(x) = x$.
(4) An element x is the *least fixed point* of f, if x is a fixed point of f and for any fixed point y of f it holds that $x \leq y$. This is denoted as $x = \mathrm{lfp}(f)$.

We are particularly interested in the complete lattice $(M^k, \subseteq)$, where M is the domain of a structure $\mathcal{M}$. Towards the definition of the semantics of least fixed-point atoms, we start with defining the operator induced by a formula.

Definition 4. Let $\mathcal{L}$ be a language and R a predicate variable of arity k. Let $\varphi(R, x_1, \dots, x_k)$ be a first-order formula in $\mathcal{L} \cup \{R\}$. Note that there could also occur free variables in φ, which are not stated explicitly. For an $\mathcal{L}$-structure, $\mathcal{M}$ define the operator F_φ on M^k by:

$$F_\varphi : \quad X \mapsto \{\overline{a} \in M^k \mid \mathcal{M} \models \varphi(X, \overline{a})\}.$$

For the definition of F_φ in the case that φ has free variables, recall that we defined $\mathcal{M} \models \varphi$ if $\mathcal{M}, \theta \models \varphi$ for all environments θ.

Lemma 5. *If R occurs only positively in φ, then F_φ is monotone.*

Proof. Let $A \subseteq B \subseteq M^k$. We have $\mathcal{M} \models A(\overline{x}) \to B(\overline{x})$. Then, Lemma 2 yields $\mathcal{M} \models \varphi(A, \overline{a}) \to \varphi(B, \overline{a})$. Thus, $F_\varphi(A) \subseteq F_\varphi(B)$. □

The following theorem is an important result in the study of complete lattices, a proof can be found in [2].

Theorem 6 (Knaster–Tarski). *Let $(E, \leq)$ be a complete lattice and f be a monotone operator on E. Then, f has a least fixed point and*:

$$\mathrm{lfp}(f) = \bigwedge \{x \mid f(x) \leq x\}.$$

Definition 7 (LFP). The semantics of an atomic least fixed-point formula is defined as follows:

$$\mathcal{M} \models [\mathrm{lfp}_R\ \varphi(R, \overline{x})](\overline{a}) :\Leftrightarrow\ \overline{a} \in \mathrm{lfp}(F_\varphi).$$

As R occurs only positively, F_φ is a monotone operator according to Lemma 5. Using Theorem 6, the least fixed point exists, and thus, the semantics of least fixed-point atoms is well defined.

Example 8. Let $\mathcal{L} = \{E\}$ be the language of graphs, where E is a binary relation symbol representing the edge relation, and let R be a binary predicate variable. Define:

$$\varphi(R, u, v) \equiv E(u, v) \vee \exists w (R(u, w) \wedge E(w, v)).$$

As R occurs only positively in φ, we can define $[\mathrm{lfp}_R\ \varphi(R, u, v)](x, y)$, which holds iff there is a path from x to y.

2.2.2 *Simultaneous Fixed-Point Logics.*

The previously introduced concepts may be generalised for product lattices. This leads to the notion of simultaneous fixed points.

Definition 9. Let $(E_1, \leq), \dots, (E_n, \leq)$ be complete lattices.

(1) A function $f : E_1 \times \cdots \times E_n \to E_1 \times \cdots \times E_n$ is called an *n-ary operator* on $E_1 \times \cdots \times E_n$.
(2) For two tuples of elements $\overline{x} := (x_1, \dots, x_n)$ and $\overline{y} := (y_1, \dots, y_n)$ in $E_1 \times \cdots \times E_n$, we write $\overline{x} \leq \overline{y}$ if $x_i \leq y_i$ for all $i \in \{1, \dots, n\}$.
(3) f is *monotone* if for all $\overline{x} \leq \overline{y}$ it holds that $f(\overline{x}) \leq f(\overline{y})$.
(4) $\overline{x}$ is a *fixed point* of f if $f(\overline{x}) = \overline{x}$.
(5) If $\overline{x}$ is a fixed point of f and for every fixed point $\overline{y}$ we have $\overline{x} \leq \overline{y}$, then $\overline{x}$ is called the *least fixed point* of f, written $\overline{x} = \mathrm{lfp}(f)$.

Similarly to Theorem 6 we have (cf. [2]):

Theorem 10. *Let $(E_1, \leq), \dots, (E_n, \leq)$ be complete lattices and f be a monotone operator on $E_1 \times \cdots \times E_n$. Then, f has a least fixed point.*

Definition 11. Let $\mathcal{L}$ be a language and $R_1, \dots, R_n$ be predicate variables, with R_i being of arity k_i. Consider an n-tuple of first-order formulas $\Phi = (\varphi_i(R_1, \dots, R_n, \overline{x_i}))_{i=1}^n$ in $\mathcal{L} \cup \{R_1, \dots, R_n\}$, where $|\overline{x_i}| = k_i$.

For a structure $\mathcal{M}$ define:

$$\begin{aligned} F_i^{\mathcal{M}}: \quad & M^{k_1} \times \cdots \times M^{k_n} \to M^{k_i}, \\ & (X_1, ..., X_n) \mapsto \{\overline{x} \in M^{k_i} \mid \mathcal{M} \models \varphi_i(X_1, ..., X_n, \overline{x})\}. \end{aligned}$$

Now define $F_\Phi^{\mathcal{M}} = (F_1^{\mathcal{M}}, ..., F_n^{\mathcal{M}})$. If $\mathcal{M}$ is clear from the context, we write F_Φ. Moreover, we write $(F_\Phi)_i$ for $F_i^{\mathcal{M}}$.

Lemma 12. *If $R_1, ..., R_n$ occur only positively in Φ, then F_Φ is monotone.*

Proof. This proof is analogous to the proof of Lemma 5. □

Definition 13 (LFP^{SIM}). Let $\mathcal{L}$ be a language and $R_1, ..., R_n$ be predicate variables with R_i being of arity k_i. Let $\Phi = (\varphi_i(R_1, ..., R_n, \overline{x_i}))_{i=1}^n$ be an n-tuple of first-order formulas in $\mathcal{L} \cup \{R_1, ..., R_n\}$, where $|\overline{x_i}| = k_i$ and $R_1, ..., R_n$ occur only positively in Φ. An atomic *simultaneous least fixed-point(LFP^{SIM})* formula is of the form:

$$[\text{lfp}_{R_i}\ \Phi](\overline{t}),$$

where $\overline{t}$ is a k_i-tuple of terms in $\mathcal{L}$. The semantics is defined as follows:

$$\mathcal{M} \models [\text{lfp}_{R_i}\ \Phi](\overline{a}) :\Leftrightarrow\ \overline{a} \in \text{lfp}(F_\Phi)_i.$$

Example 14. Again consider the language of graphs $\mathcal{L} = \{E\}$, where E is a binary relation symbol. Let R and S be two binary predicate variables. Define Φ as:

$$\begin{aligned} \varphi_1(R, S, u, v) &\equiv E(u, v) \vee \exists w(S(u, w) \wedge E(w, v)), \\ \varphi_2(R, S, u, v) &\equiv \exists w(R(u, w) \wedge E(w, v)). \end{aligned}$$

As R and S occur only positively in Φ, we can define the atomic LFP^{SIM} formulas $[\text{lfp}_R\ \Phi](x, y)$ and $[\text{lfp}_S\ \Phi](x, y)$. The former holds iff there is a path of odd length and the latter iff there is a path of even length from x to y.

Define FO[LFP^{SIM}] to be an extension of first-order logic, which also allows atomic LFP^{SIM} formulas. Analogously SO[LFP^{SIM}] is an extension of second-order logic, which additionally allows atomic LFP^{SIM} formulas.

Note that FO[LFP^{SIM}], while being more convenient, is not more expressive than FO[LFP] in the following sense: For every formula φ in FO[LFP^{SIM}], there exists a formula ψ in FO[LFP], such that $\varphi \equiv \psi$. A proof of this result can be found in [45]. Note that the converse of this result is also true, as FO[LFP^{SIM}] is a generalisation of FO[LFP]. Thus, we can use LFP and LFP^{SIM} interchangeably. It is also well-known that the LFP-operator can be expressed in second-order logic so that we have: For every FO[LFP] formula φ, there exists an SO formula ψ such that $\varphi \equiv \psi$.

2.2.3 Fixed-Point Approximation. We are particularly interested in least fixed-point formulas defined from existential first-order formulas. We call a first-order formula φ existential, if there exists a formula $\tilde{\varphi}$ in prenex normal form without universal quantifiers, that is logically equivalent to φ. For many applications, in particular in software verification, it suffices to consider least fixed-points of existential formulas. This point has been made prominently in [9]. Those least fixed-point formulas may be approximated by first-order logic formulas. In order to achieve this, we first define infinite conjunctions and disjunctions:

Definition 15. Let $\mathcal{L}$ be a language and $\mathcal{M}$ be an $\mathcal{L}$-structure. Let Ψ be a set of first-order formulas in $\mathcal{L}$. Define:

$$\mathcal{M} \models \bigvee_{\varphi \in \Psi} \varphi :\Leftrightarrow \exists \varphi \in \Psi : \mathcal{M} \models \varphi,$$
$$\mathcal{M} \models \bigwedge_{\varphi \in \Psi} \varphi :\Leftrightarrow \forall \varphi \in \Psi : \mathcal{M} \models \varphi.$$

Lemma 16. *Let $\mathcal{L}$ be a language and let Ψ_1, Ψ_2 be sets of first-order formulas in $\mathcal{L}$. Then:*

(1) $\neg \bigvee_{\varphi \in \Psi_1} \varphi \equiv \bigwedge_{\varphi \in \Psi_1} \neg\varphi$,
(2) $\bigwedge_{\varphi_1 \in \Psi_1} \varphi_1 \vee \bigwedge_{\varphi_2 \in \Psi_2} \varphi_2 \equiv \bigwedge_{(\varphi_1, \varphi_2) \in \Psi_1 \times \Psi_2} \varphi_1 \vee \varphi_2$,
(3) $\bigwedge_{\varphi_1 \in \Psi_1} \varphi_1 \wedge \bigwedge_{\varphi_2 \in \Psi_2} \varphi_2 \equiv \bigwedge_{\varphi \in \Psi_1 \cup \Psi_2} \varphi$,
(4) $\forall y \bigwedge_{\varphi \in \Psi_1} \varphi(y) \equiv \bigwedge_{\varphi \in \Psi_1} \forall y\, \varphi(y)$.

For existential first-order formulas, we can define a sequence of sets that converges to the least fixed point. These sets may be described by first-order formulas.

Definition 17. Let $(E_1, \leq), ..., (E_n, \leq)$ be complete lattices and f be an n-ary operator on $E_1 \times \cdots \times E_n$. By induction define the following elements in $E_1 \times \cdots \times E_n$:

$$\begin{aligned} S_f^0 &= (\bot, ..., \bot), \\ S_f^{n+1} &= f(S_f^n), \\ S_f^\omega &= \bigcup_{n \in \omega} S_f^n. \end{aligned} \tag{1}$$

If it is obvious which operator we talk of, we sometimes omit the subscript and write S^n and S^ω.

We are interested in the operator F_Φ of Definition 11. If F_Φ is defined from existential formulas, then F_Φ is continuous. In this case, Kleene's fixed point theorem states that S^ω coincides with the least fixed point of F_Φ. The lemma also follows from [9, Theorem 9].

Lemma 18. *Let $\mathcal{L}$ be a language and $R_1, ..., R_n$ be predicate variables. Let $\Phi = (\varphi_i(R_1, ..., R_n, \overline{x_i}))_{i=1}^n$ be an n-tuple of existential first-order formulas, such that $R_1, ..., R_n$ occur only positively in Φ. Let $\mathcal{M}$ be an $\mathcal{L}$-structure and $f = F_\Phi^\mathcal{M}$. Then, $S_f^\omega = \mathrm{lfp}(f)$.*

The next step is to describe the sets S^n and S^ω with first-order formulas.

Definition 19. Let $\mathcal{L}$ be a language and $R_1, ..., R_n$ be predicate variables. Let $\Phi = (\varphi_i(R_1, ..., R_n, \overline{x_i}))_{i=1}^n$ be an n-tuple of first-order formulas. For $j \in \{1, ..., n\}$ define:

$$\begin{aligned} \sigma_{j,\Phi}^0(\overline{x_j}) &\equiv \bot \\ \sigma_{j,\Phi}^{l+1}(\overline{x_j}) &\equiv \varphi_j(\sigma_{1,\Phi}^l, ..., \sigma_{n,\Phi}^l, \overline{x_j}) \\ \sigma_{j,\Phi}^\omega(\overline{x_j}) &\equiv \bigvee_{l \in \omega} \sigma_{j,\Phi}^l(\overline{x_j}). \end{aligned}$$

Lemma 20. *Let $\mathcal{L}$ be a language and let $R_1, ..., R_n$ be predicate variables. Let $\Phi = (\varphi_i(R_1, ..., R_n, \overline{x_i}))_{i=1}^n$ be an n-tuple of first-order formulas, such that $R_1, ..., R_n$ occur only positively in Φ. Let $\mathcal{M}$ be an $\mathcal{L}$-structure. For every $j \in \{1, ..., n\}$ and $\overline{a} \in M^{k_j}$, where k_j equals the arity of R_j, we have:*

$$\mathcal{M} \models \sigma_{j,\Phi}^l(\overline{a}) \quad \Leftrightarrow \quad \overline{a} \in (S_{F_\Phi}^l)_j,$$

for every $l \in \omega \cup \{\omega\}$.

Proof. This follows by induction on l from the definition of $\sigma^l_{j,\Phi}$ and $S^l_{F_\Phi}$. □

Theorem 21. *Let $\mathcal{L}$ be a language and let $R_1, \ldots, R_n$ be predicate variables. Let $\Phi = (\varphi_i(R_1, \ldots, R_n, \overline{x_i}))_{i=1}^n$ be an n-tuple of existential first-order formulas, such that $R_1, \ldots, R_n$ occur only positively in Φ. Then, for all $j \in \{1, \ldots, n\}$:*

$$[\mathrm{lfp}_{R_j}\ \Phi] \equiv \sigma^\omega_{j,\Phi}.$$

Proof. Let $\mathcal{M}$ be an $\mathcal{L}$-structure and $\overline{a} \in M^{k_j}$. Lemma 18 states that $\mathcal{M} \models [\mathrm{lfp}_{R_j}\ \Phi](\overline{a})$ iff $\overline{a} \in (S^\omega_{F_\Phi})_j$ for $j \in \{1, \ldots, n\}$. Now Lemma 20 concludes $\mathcal{M} \models [\mathrm{lfp}_{R_j}\ \Phi](\overline{a}) \leftrightarrow \sigma^\omega_{j,\Phi}(\overline{a})$ for $j \in \{1, \ldots, n\}$.□

3 Formula Equations

3.1 Introduction to Formula Equations

Equations are pervasive in mathematics. Solving a given equation means to find objects for the unknowns such that, after substitution, the two sides of the equation are equal. What kinds of objects are permissible as solutions and what ‘equal’ means depends on the type of equations under consideration. Occasionally, equations can be brought into normal forms in which one of the two sides becomes trivial. This is, for example, the case for systems of linear equations.

In this paper, we work with formula equations in first-order logic. As a formula equation we want to consider

$$\text{an } \mathcal{L}\text{-formula } \psi_1 \leftrightarrow \psi_2 \text{ containing predicate variables } \overline{X} = X_1, \ldots, X_n. \tag{2}$$

The kind of objects we want to allow as solutions are first-order formulas. The notion of equality we want to satisfy after inserting a solution for the predicate variables is logical equivalence. Consequently, a solution of Equation (2) is a first-order substitution $[\overline{X} \backslash \overline{\chi}]$ such that $\models \psi_1[\overline{X} \backslash \overline{\chi}] \leftrightarrow \psi_2[\overline{X} \backslash \overline{\chi}]$. Formula equations can easily be brought into a form in which one of the two sides is trivial: we can simplify formula equations by instead considering:

$$\text{an } \mathcal{L}\text{-formula } \psi \text{ containing predicate variables } \overline{X} = X_1, \ldots, X_n. \tag{3}$$

Then, as a solution, we ask for a first-order substitution $[\overline{X} \backslash \overline{\chi}]$ such that $\models \psi[\overline{X} \backslash \overline{\chi}]$. Note that every instance of Equation (2) is an instance of Equation (3) by letting ψ be $\psi_1 \leftrightarrow \psi_2$ and every instance of Equation (3) is an instance of Equation (2) by letting ψ_1 be ψ and ψ_2 be $\top$. Moreover, it will be notationally useful to explicitly indicate the predicate variables by existential quantifiers. Consequently, we define:

Definition 22. A *formula equation* is a closed $\mathcal{L}$-formula $\exists \overline{X}\, \psi$ where ψ contains only first-order quantifiers. A *solution* of $\exists \overline{X}\, \psi$ is a first-order substitution $[\overline{X} \backslash \overline{\chi}]$ such that $\models \psi[\overline{X} \backslash \overline{\chi}]$. We call a formula equation *solvable* if there exists a solution.

Note that a formula equation is simply a closed existential second-order formula. We prefer to work with the terminology ‘formula equation’ because (1) it is shorter and (2) it emphasises the aspect of finding a solution which is central for this entire work.

Example 23. We will now start a more precise treatment of the formula equation:

$$\exists X\ (X(2) \land \forall u\ (X(u) \to X(u+u)) \land \neg X(3)),$$

mentioned in the Introduction: we work in the language $\mathcal{L}_A = \{0, s, +, <\}$ of linear arithmetic. 2 is an abbreviation for the term $s(s(0))$, 3 is an abbreviation of $s(s(s(0)))$, and so on. We need a simple background theory to establish basic facts, such as, $\neg\exists x\ 3 = x + x$. To that aim let T be the theory of open induction for $\mathcal{L}_A = \{0, s, +, <\}$. Then, we define the formula equation:

$$\exists X \left(T \rightarrow (X(2) \land \forall u\, (X(u) \rightarrow X(u+u)) \land \neg X(3))\right).$$

Let $\chi_1(v) \equiv \exists w\, (v = w + w)$ and $\chi_2(v) \equiv v \neq 3$. Then, $[X\backslash\chi_1]$ and $[X\backslash\chi_2]$ are solutions.

The problem of computing a solution of a formula equation given as input will be denoted as FEQ in the sequel. A formula equation $\exists\overline{X}\,\psi$ is called *valid* if it is a valid second-order formula and *satisfiable* if it is a satisfiable second-order formula. Every solvable formula equation is valid, and every valid formula equation is satisfiable but neither of the converse implications are true as the following example shows.

Example 24. If ψ is a first-order formula which is satisfiable but not valid and does not contain X then, trivially, $\exists X\,\psi$ is a formula equation which is satisfiable but not valid.

Towards an example for a valid but unsolvable formula equation, we work in the first-order language $\mathcal{L} = \{0, s\}$, where 0 is a constant symbol and s is a unary function symbol. Let A_1 be $\forall x\, s(x) \neq 0$, let A_2 be $\forall x \forall y\, (s(x) = s(y) \rightarrow x = y)$ and consider the formula:

$$A_1 \land A_2 \rightarrow \exists X \exists Y \forall u \Big(X(0) \land Y(s(0)) \land (X(u) \rightarrow Y(s(u))) \land (Y(u) \rightarrow X(s(u))) \land \neg(X(u) \land Y(u))\Big),$$

which, up to some simple logical equivalence transformations, is a formula equation Φ. Now Φ is valid since, in a model $\mathcal{M}$ of $A_1 \land A_2$, interpreting X by $\{s^{2n}(0)^{\mathcal{M}} \mid n \in \mathbb{N}\}$ and Y by $\{s^{2n+1}(0)^{\mathcal{M}} \mid n \in \mathbb{N}\}$ makes the remaining formula true.

For unsolvability suppose that Φ has a solution $[X\backslash\chi(u), Y\backslash\psi(u)]$. Then, since the standard model $\mathbb{N}$ in the language $\mathcal{L}$ satisfies $A_1 \land A_2$, we would have:

$$\mathbb{N} \models \chi(0) \land \psi(s(0)) \land \forall u\, (\chi(u) \rightarrow \psi(s(u))) \land \forall u\, (\psi(u) \rightarrow \chi(s(u))) \land \forall u\, \neg(\chi(u) \land \psi(u)),$$

in particular, χ would be a definition of the even numbers and ψ a definition of the odd numbers. However, the theory of $\mathbb{N}$ in $\mathcal{L}$ admits quantifier elimination [18, Theorem 31G], which has the consequence that the L-definable sets in $\mathbb{N}$ are the finite and co-finite subsets of $\mathbb{N}$ [18, Section 3.1, Exercise 4]. Thus, we obtain a contradiction to χ being a definition of the even numbers (which is neither finite nor co-finite).

Solving formula equations (FEQ) is closely related to the problem of second-order quantifier elimination: given a formula $\exists\overline{X}\,\psi$ where ψ contains only first-order quantifiers, find a first-order formula φ such that $\models \exists\overline{X}\,\psi \leftrightarrow \varphi$. The relationship between FEQ and SOQE is often based on a third problem: second-order quantifier elimination by a witness (WSOQE): given a formula $\exists\overline{X}\,\psi$ where ψ contains only first-order quantifiers, find a first-order substitution $[\overline{X}\backslash\overline{\chi}]$ such that $\models \exists\overline{X}\,\psi \leftrightarrow \psi[\overline{X}\backslash\overline{\chi}]$. Such formulas $\overline{\chi}$ are called ELIM-witnesses in [64]. Such a tuple $\overline{\chi}$ of formulas is a canonical witness for $\exists\overline{X}\,\psi$ in the sense that, whenever there is a tuple of sets $\overline{X}$ satisfying ψ then $\overline{\chi}$ does. The existence of (computable) canonical witnesses has two useful corollaries: (1) SOQE can be solved in a straightforward way and (2) FEQ boils down to a validity check. This has been exploited, for example, in the proof of decidability of the QFBUP problem [17]. Our fixed point theorem can be seen as precisely such a result on the existence of a canonical witness in FO[LFP].

The complex of these three problems, FEQ, SOQE and WSOQE, has a long history in logic and a wealth of applications in computer science, see the textbook [22] on second-order quantifier elimination. A number of algorithms for second-order quantifier elimination have been developed, for example: The SCAN algorithm introduced in [21] (tries to) compute(s) a first-order formula equivalent to $\exists X\,\psi$ for a conjunctive normal form ψ by forming the closure of ψ under constraint resolution which only resolves on X-literals. The DLS algorithm has been introduced in [14] and consists essentially of formula rewriting steps tailored to allow application of Ackermann's lemma (which instantiates a predicate variable provided some conditions on the polarity of its occurrences are met). This approach has been generalised by Nonnengart and Szałas in [50] by allowing FO[LFP], which resulted in the DLS* algorithm.

3.2 Horn Formula Equations

Towards the definition of Horn formula equations, we first introduce the notion of constrained clause.

Definition 25. Let $\mathcal{L}$ be a first-order language. A *constrained clause* is a formula C of the form:

$$\gamma \vee \bigvee_{i=1}^{m} \neg X_i(\overline{t_i}) \vee \bigvee_{j=1}^{n} Y_j(\overline{s_j}),$$

where γ is a first-order formula in $\mathcal{L}$, X_i, Y_j are predicate variables and $\overline{t_i}, \overline{s_j}$ are tuples of terms in $\mathcal{L}$ for $i \in \{1, ...m\}$, $j \in \{1, ..., n\}$. C is called

(1) *Horn*, if $n \leq 1$,
(2) *dual-Horn*, if $m \leq 1$, and
(3) *linear-Horn*, if $m, n \leq 1$.

Note that a constrained clause is allowed to (and typically does) contain free individual variables which, as usual in clause logic, are treated as universally quantified. A finite set S of constrained clauses is considered the conjunction of these clauses and is thus logically equivalent to a formula of the form $\forall^* \bigwedge_{C \in S} C$ where $\forall^*$ denotes the universal closure w.r.t. individual variables.

Definition 26. A formula equation $\exists \overline{X} \forall^* \bigwedge_{C \in S} C$ is called a

(1) *Horn formula equation*, if S is a set of constrained Horn clauses,
(2) *dual-Horn formula equation*, if S is a set of constrained dual-Horn clauses and
(3) *linear-Horn formula equation*, if S is a set of constrained linear-Horn clauses.

Thus, Horn formula equations correspond to existential second-order Horn logic, which also plays a significant role in finite model theory, see [24].

Example 27. The formula equation:

$$\exists X \left(T \to (X(2) \wedge \forall u\, (X(u) \to X(u+u)) \wedge \neg X(3)\right),$$

from Example 23 is logically equivalent to the Horn formula equation:

$$\exists X\, \psi \equiv \exists X \left((T \to X(2)) \wedge \forall u\, (T \wedge X(u) \to X(u+u)) \wedge (T \to \neg X(3))\right).$$

We will use this Horn formula equation as running example throughout most of this paper.

There are different notions of solvability for constrained Horn clauses in the literature: *satisfiability* of [26] is satisfiability of a Horn formula equation, *semantic solvability* of [55] is validity of a Horn formula equation, and *syntactic solvability* of [55] is solvability of a Horn formula equation. In this paper, we will primarily be interested in this last notion: solvability of a Horn formula equation. Note that the formula equation considered in Example 24 is Horn so that, also when restricted to the class of Horn formula equations, solvability implies validity and validity implies satisfiability, but both inverse implications are not true.

Let $\exists \overline{X}\, \psi$ be a Horn formula equation. We distinguish three different types of clauses in ψ:

$$\begin{array}{lrl} \text{(B)} & \gamma & \to Y(\overline{s}), \\ \text{(I)} & \gamma \wedge X_1(\overline{t_1}) \wedge \cdots \wedge X_m(\overline{t_m}) & \to Y(\overline{s}), \\ \text{(E)} & \gamma \wedge X_1(\overline{t_1}) \wedge \cdots \wedge X_m(\overline{t_m}) & \to \bot, \end{array}$$

where the constraint γ is a formula in $\mathcal{L}$ not containing predicate variables, $m \geq 1$, $\overline{t_1}, ..., \overline{t_m}, \overline{s}$ are tuples of first-order terms in $\mathcal{L}$ of appropriate arity and $Y, X_1, ..., X_m$ are predicate variables. Note that free variables $\overline{y}$ may occur in the formulas γ and the terms $\overline{s}, \overline{t_1}, ..., \overline{t_m}$. We call the first *base clauses*, the second *induction clauses*, and the third *end clauses*.

For simplicity, let us first consider a Horn formula equation $\exists X\, \psi$ containing only one predicate variable X. For finding a solution R for X in a given model $\mathcal{M}$, it is natural to proceed as follows: for any base clause $\gamma \to X(\overline{s})$, if $\mathcal{M} \models \gamma$, then add $\overline{s}$ to R. Inductively, for any induction clause $\gamma \wedge X(\overline{t_1}) \wedge \cdots \wedge X(\overline{t_m}) \to X(\overline{s})$, if $\mathcal{M} \models \gamma$ and $\overline{t_1}, ..., \overline{t_m}$ are already in R, then add $\overline{s}$ to R. In the rest of this section, we lay the groundwork for pushing this procedure on the object level. We translate the clauses of the form (B) and (I) to formulas, where all predicate variables occur only positively: Let C be an induction clause of the form $\gamma \wedge X_1(\overline{t_1}) \wedge \cdots \wedge X_m(\overline{t_m}) \to Y(\overline{s})$. We define the translated clause $T_C(\overline{x})$ to be $\gamma \wedge X_1(\overline{t_1}) \wedge \cdots \wedge X_m(\overline{t_m}) \wedge \overline{x} = \overline{s}$. Similarly, for a base clause C of the form $\gamma \to Y(\overline{s})$, the translated clause $T_C(\overline{x})$ is $\gamma \wedge \overline{x} = \overline{s}$.

Example 28. Continuing the running example, Example 27, let C_1 be the base clause $T \to X(2)$ and C_2 be the induction clause $T \wedge X(u) \to X(u+u)$. Then, $T_{C_1}(x) \equiv T \wedge x = 2$ and $T_{C_2}(x) \equiv T \wedge X(u) \wedge x = u + u$.

Next, we gather all clauses having the same positive predicate variable. For a base or induction clause C, let P_C be the unique predicate variable that occurs positively. Let $\exists X_1 \cdots \exists X_n\, \psi$ be a Horn formula equation with predicate variables $X_1, ..., X_n$. We define B_j and I_j to be the set of base and induction clauses C in ψ, such that $P_C = X_j$, for $j = 1, ..., n$.

Definition 29. Let $\exists X_1 \cdots \exists X_n\, \psi$ be a Horn formula equation. Define Φ_ψ to be the n-tuple of first-order formulas $(\varphi_1, ..., \varphi_n)$ defined as:

$$\varphi_j(X_1, ..., X_n, \overline{x_j}) := \exists \overline{y} \bigvee_{C \in B_j \cup I_j} T_C(\overline{x_j}),$$

where $\overline{y}$ are the free variables occurring in the clauses in $B_j \cup I_j$ and $\overline{x_j}$ is a tuple of variables such that $|\overline{x_j}|$ equals the arity of X_j. If the context is clear, we sometimes omit the subscript and write Φ instead of Φ_ψ to achieve clearer notation.

Example 30. Continuing the running example, Example 27, Φ_ψ is a 1-tuple consisting of:

$$\varphi(X, x) \equiv \exists u\, ((T \wedge x = 2) \vee (T \wedge X(u) \wedge x = u + u)),$$

which is logically equivalent to:

$$T \wedge \big((x = 2) \vee \exists u\, (X(u) \wedge x = u + u)\big).$$

In Φ_ψ all predicate variables occur only positively. Hence, it defines a monotone operator which has a least fixed point. From the point of view of (constraint) logic programming, the above tuple of formulas is a first-order definition of the operator T_P induced by $\exists \overline{X}\, \psi$ when considered as a constraint logic program P, see, e.g., [37].

Lemma 31. *Let $\exists X_1 \cdots \exists X_n\, \psi$ be a Horn formula equation. Let E be the set of end clauses in ψ and $\Phi_\psi = (\varphi_1, ..., \varphi_n)$. Then:*

$$\psi \equiv \bigwedge_{j=1}^{n} \forall \overline{x_j}\, (\varphi_j(X_1, ..., X_n, \overline{x_j}) \to X_j(\overline{x_j})) \wedge \forall \overline{y} \bigwedge_{C \in E} C.$$

Proof. It suffices to show that, for all $j \in \{1, ..., n\}$, for all models $\mathcal{M}$, and for all environments θ interpreting $X_1, ..., X_n$:

$$\mathcal{M}, \theta \models \forall \overline{x_j}\,(\varphi_j(X_1, ..., X_n, \overline{x_j}) \to X_j(\overline{x_j})) \text{ iff}$$
$$\mathcal{M}, \theta \models C \text{ for all induction clauses } C \text{ in } \psi \text{ with head symbol } X_j.$$

First assume that $\mathcal{M}, \theta \not\models C$ for some induction clause C with head symbol X_j, i.e., $\mathcal{M}, \theta \models \exists \overline{y}(\gamma \wedge \bigwedge_{k=1}^{m} Z_k(\overline{t_k}) \wedge \neg X_j(\overline{s}))$. Then, $T_C(\overline{x}) \equiv \gamma \wedge \bigwedge_{k=1}^{m} Z_k(\overline{t_k}) \wedge \overline{x} = \overline{s}$ and therefore $\mathcal{M}, \theta \models \exists \overline{y}\, T_C(\overline{s})$. Thus, it holds $\mathcal{M}, \theta \models (\varphi_j(X_1, ..., X_n, \overline{s}) \wedge \neg X_j(\overline{s}))$ and hence $\mathcal{M}, \theta \not\models \forall \overline{x_j}(\varphi_j(X_1, ..., X_n, \overline{x_j}) \to X_j(\overline{x_j}))$. The analogous argument holds for base clauses.

For the other direction assume that $\mathcal{M}, \theta \not\models \bigwedge_{j=1}^{n} \forall \overline{x_j}(\varphi_j(X_1, ..., X_n, \overline{x_j}) \to X_j(\overline{x_j}))$, i.e., $\mathcal{M}, \theta \models \exists \overline{x_j}(\varphi_j(X_1, ..., X_n, \overline{x_j}) \wedge \neg X_j(\overline{x_j}))$ for some $j \in \{1, ..., n\}$. Thus, there is a clause $C \in B_j \cup I_j$ such that $\mathcal{M}, \theta \models \exists \overline{x_j}(\exists \overline{y}\, T_C(\overline{x_j}) \wedge \neg X_j(\overline{x_j}))$. Let C be an induction clause, the same argument also holds for base clauses. Then, $T_C(\overline{x}) \equiv \gamma \wedge \bigwedge_{k=1}^{m} Z_k(\overline{t_k}) \wedge \overline{x} = \overline{s}$. It follows that $\overline{x_j} = \overline{s}$ and $\mathcal{M}, \theta \models \exists \overline{y}(\gamma \wedge \bigwedge_{k=1}^{m} Z_k(\overline{t_k}) \wedge \neg X_j(\overline{s}))$. Hence, $\mathcal{M}, \theta \not\models C$. □

The idea of the above transformation is to split the positive and negative occurrences of $X_1, ..., X_n$ in ψ, note that $X_1, ..., X_n$ occur only positively in $\varphi_1, ..., \varphi_n$ and only negatively in the end clauses $C \in E$. This will be taken advantage of later.

4 The Abstract Fixed-Point Theorem

The aim of this section is to prove the main result of this paper, the following abstract fixed-point theorem. It states that Horn formula equations have a least solution.

Theorem 32 (Abstract Horn Fixed-Point Theorem). *Let $\exists \overline{X} \psi$ be a Horn formula equation and $\mu_j := [\mathrm{lfp}_{X_j}\, \Phi_\psi]$ for $j \in \{1, ..., n\}$, then:*

(1) $\models_{\mathrm{a}} \exists \overline{X}\, \psi \leftrightarrow \psi[\overline{X} \backslash \overline{\mu}]$ *and*
(2) *if* $(\mathcal{M}, G) \models_{\mathrm{a}} \psi[\overline{X} \backslash \overline{R}]$ *for some model abstraction* $(\mathcal{M}, G)$ *and abstract relations* $R_1, ..., R_n$, *then* $(\mathcal{M}, G) \models_{\mathrm{a}} \bigwedge_{j=1}^{n}(\mu_j \to R_j)$.

Here, Φ_ψ is the tuple of first-order formulas induced by $\exists \overline{X}\, \psi$ as in Definition 29. The theorem is formulated for abstract semantics $\models_{\mathrm{a}}$ which is a generalisation of standard semantics $\models$ that corresponds to abstract interpretation. It will be defined in detail in the upcoming Section 4.1. As a special case, the fixed-point theorem also applies to standard semantics: Just replace $\models_{\mathrm{a}}$ by $\models$, 'model abstraction $(\mathcal{M}, G)$' by 'model $\mathcal{M}$' and 'abstract relation' by 'relation'.

Example 33. Continuing the running example, Theorem 32 states that, for $\mu(y) := [\mathrm{lfp}_X\, \Phi_\psi](y)$:

$$\exists X\, \psi \equiv (T \to \mu(2)) \wedge \forall u(T \wedge \mu(u) \to \mu(u + u)) \wedge (T \to \neg\mu(3)).$$

The first two clauses are true by the definition of μ, hence $\exists X\, \psi$ is further equivalent to $T \to \neg\mu(3)$.

4.1 Model Abstractions

In this section, we define the semantics of *model abstractions*. This is a generalisation of the usual semantics of SOL[LFP] obtained by restricting the scope of second-order quantifiers to range only over certain relations and the domain in which the least fixed point of the lfp operator is computed. The primary motivation for this semantics is that in many applications one is interested in speaking about the relations definable by certain formulas, see, e.g., Sectiond 5 and 6. For example, we want to prove that a formula equation has a solution in a certain class of formulas. We will cover this situation with the notion of *definable model abstractions*. Since we also want to interpret the lfp operators, the relations need to form a complete lattice.

Model abstractions are defined via a Galois connection to the powerset lattice, and thus, they are the analogue of abstract interpretation for logical formulas. Abstract interpretation has been introduced in [12] and is nowadays one of the most important techniques in static analysis and software verification. For more background on Galois connections, see [13]. The standard semantics is the special case where the Galois connection is the identity.

The idea of restricting the range of second-order quantifiers is reminiscent of Henkin semantics of second-order logic [27]. Our abstract semantics differs from Henkin semantics in two respects: on the one hand we do not require, as Henkin semantics does, closure under definability which has the effect that there are model abstractions which are not Henkin structures. On the other hand, we require the lattice of sets to be complete which is necessary for interpreting lfp formulas in a suitable way.

Definition 34. Let $\mathcal{A} = (A, \subseteq)$ and $\mathcal{B} = (B, \sqsubseteq)$ be two partially ordered sets. A *Galois connection* between $\mathcal{A}$ and $\mathcal{B}$ consists of two functions $\alpha : A \to B$ and $\gamma : B \to A$, such that for all $X \in A$ and $Y \in B$:

$$X \subseteq \gamma(Y) \quad \Leftrightarrow \quad \alpha(X) \sqsubseteq Y.$$

From this condition, it follows that α and γ are monotone.

Definition 35. Let $\mathcal{L}$ be a language. A *model abstraction* is a pair $(\mathcal{M}, G)$, where $\mathcal{M}$ is an $\mathcal{L}$-structure and $G = (\mathcal{V}_k, \alpha_k, \gamma_k)_{k\in\mathbb{N}}$ is a sequence of triples, such that for all $k \in \mathbb{N} : \mathcal{V}_k = (V_k, \sqsubseteq)$ is a lattice and $\alpha_k : \mathcal{P}(M^k) \to V_k$ and $\gamma_k : V_k \to \mathcal{P}(M^k)$ form a Galois-connection between $(\mathcal{P}(M^k), \subseteq)$ and $\mathcal{V}_k$. We call G the *set domain* of $(\mathcal{M}, G)$ and sets $R \sqsubseteq V_k$ *abstract relations* of arity k.

Remark 36. Note that the completeness of $(M^k, \subseteq)$ and the Galois-connection imply the completeness of the lattices $(V_k, \sqsubseteq)$.

Model abstractions which are defined by a set of first-order formulas are of particular interest for applications in program verification.

Definition 37. Let $\mathcal{L}$ be a language and $\mathcal{M}$ be a model. Let C be a set of first-order formulas such that the sets definable by formulas from C, i.e., $V_k = \{R \subseteq M^k \mid \exists\varphi \in C : \mathcal{M} \models R(\overline{x}) \leftrightarrow \varphi(\overline{x})\}$, form a complete lattice $\mathcal{V}_k = (V_k, \subseteq)$ for all $k \in \mathbb{N}$. Then, there is a unique α_k such that $\alpha_k : \mathcal{P}(M^k) \to V_k$ and $\mathrm{id}_k : V_k \to \mathcal{P}(M^k)$ form a Galois-connection. We define $G_C = (\mathcal{V}_k, \alpha_k, \mathrm{id}_k)_{k\in\mathbb{N}}$ and call $(\mathcal{M}, G_C)$ a *definable model abstraction.*

Example 38. Consider the language $\mathcal{L}_{\mathrm{aff}} = (0, 1, +, (c)_{c\in\mathbb{Q}})$ where the intended interpretation of the unary function symbol c for $c \in \mathbb{Q}$ is multiplication with c. Define the set domain $G_{\mathrm{aff}} = ((\mathrm{Aff}\ \mathbb{Q}^k, \subseteq), \mathrm{aff}_k, \mathrm{id}_k)_{k\in\mathbb{N}}$, where Aff $\mathbb{Q}^k$ is the set of all affine subspaces of $\mathbb{Q}^k$, aff_k maps every subset of $\mathbb{Q}^k$ to its affine hull and id_k is the embedding of Aff $\mathbb{Q}^k$ in $\mathcal{P}(\mathbb{Q}^k)$. Then, $(\mathbb{Q}, G_{\mathrm{aff}})$ is a model abstraction. Moreover, letting C be the set of conjunctions of equations in $\mathcal{L}_{\mathrm{aff}}$, $(\mathbb{Q}, G_{\mathrm{aff}})$ turns out to be the definable model abstraction $(\mathbb{Q}, G_C)$.

The intention of the notion of definable model abstraction is seen most clearly in Theorem 72, which shows that the truth of the verification condition of a partial correctness assertion in $(\mathcal{M}, G_C)$ is equivalent to its provability in the Hoare calculus with invariants restricted to formulas from C.

We will now introduce the abstract semantics of SO[LFP] formulas by defining a satisfaction relation $(\mathcal{M}, G) \models_{\mathrm{a}} \varphi$. The crucial difference between $\models_{\mathrm{a}}$ and standard Tarski semantics $\models$ will be that second-order quantifiers and the least fixed-point operator will not be interpreted in the power set of the domain but in G instead (for the appropriate arity).

Definition 39. The defining clauses for first-order atoms, propositional connectives, and first-order quantifiers for $\models_{\mathrm{a}}$ are identical to those for $\models$. For formulas of the form $\exists X\, \psi$, where X is a k-ary predicate variable, we define:

$$(\mathcal{M}, G), \theta \models_{\mathrm{a}} \exists X \psi \quad \Leftrightarrow \quad \exists R \in V_k : (\mathcal{M}, G), \theta[X := \gamma_k(R)] \models_{\mathrm{a}} \psi,$$

and analogously for formulas of the form $\forall X \psi$. The semantics of the lfp-operator is defined as follows: Let $(\mathcal{M}, G)$ be a model abstraction and $X_1, \ldots, X_n$ be predicate variables with X_i having arity k_i. Let $\Phi = (\varphi_i(X_1, \ldots, X_n, \overline{u_i}))_{i=1}^n$ be an n-tuple of first-order formulas such that $|\overline{u_i}| = k_i$ and $X_1, \ldots, X_n$ occur only positively in Φ. Define:

$$\begin{aligned} F_i^\# : \ & V_{k_1} \times \cdots \times V_{k_n} \to V_{k_i} \\ & (Y_1, \ldots, Y_n) \mapsto \alpha_{k_i} \circ (F_\Phi^{\mathcal{M}})_i(\gamma_{k_1}(Y_1), \ldots, \gamma_{k_n}(Y_n)), \end{aligned}$$

and $F_\Phi^\# = (F_1^\#, \ldots, F_n^\#)$. Let $\overline{a} \in \mathcal{M}$, then:

$$(\mathcal{M}, G) \models_{\mathrm{a}} [\mathrm{lfp}_{X_i}\ \Phi](\overline{a}) \quad \Leftrightarrow \quad \overline{a} \in \gamma_{k_i}(\mathrm{lfp}(F_\Phi^\#)_i).$$

Let R be an abstract relation of arity k. Then, we write $(\mathcal{M}, G), \theta \models_{\mathrm{a}} \varphi(R)$ as abbreviation for $(\mathcal{M}, G), \theta[X := \gamma_k(R)] \models_{\mathrm{a}} \varphi(X)$.

Note that the semantics of the least fixed-point operator is well defined: We already know that F_Φ is a monotone n-ary operator and α_{k_i} and γ_{k_i} are monotone as they form a Galois connection for $i \in \{1, \ldots, n\}$. Thus, $F_i^\#$ is monotone for all $i \in \{1, \ldots, n\}$, and therefore, $F_\Phi^\#$ is monotone as well. As $\mathcal{V}_k$ is a complete lattice for all $k \in \mathbb{N}$, we can use the Knaster–Tarski theorem to obtain the least fixed point of $F_\Phi^\#$.

Note that every classical model defines, by choosing the identity for α_k and γ_k, a model abstraction, that satisfies the same formulas. Hence, $\models_{\mathrm{a}} \varphi \implies \ \models \varphi$ for every SO[LFP] formula φ, and for a formula equation $\exists \overline{X}\, \varphi$, we even have $(\mathcal{M}, G) \models_{\mathrm{a}} \exists \overline{X}\, \varphi \implies \mathcal{M} \models \exists \overline{X}\, \varphi$. The converse is not true as the following example shows, yet for first-order formulas φ we have $\models_{\mathrm{a}} \varphi \Leftrightarrow \ \models \varphi$. In particular, if φ_1 and φ_2 are logically equivalent first-order formulas, we also have $\models_{\mathrm{a}} \varphi_1 \leftrightarrow \varphi_2$.

Example 40. We have observed in Example 24 that the formula equation Φ defined there is valid, i.e., $\models \Phi$. However, we do not have $\models_{\mathrm{a}} \Phi$. To see this, consider the model abstraction $(\mathbb{N}, G)$ where $\mathbb{N}$ is considered as structure in the language $\mathcal{L} = \{0, s\}$, $G = (\mathcal{V}_k, \alpha_k, \gamma_k)_{k \in \mathbb{N}}$ with $\mathcal{V}_k = (V_k, \sqsubseteq)$ where $V_k = \{\emptyset, \mathbb{N}^k\}$, $\alpha_k = \mathrm{id} \restriction V_k$ and $\gamma_k = \mathrm{id}$. Then, $(\mathbb{N}, G) \not\models_{\mathrm{a}} \Phi$ because neither $\emptyset$ nor $\mathbb{N}$ is a solution for the formula equation Φ.

4.2 Proof of the Fixed-Point Theorem

The proof of the abstract fixed-point theorem relies on a generalisation of a result by Nonnengart and Szałas [50], which is in turn a generalisation of Ackermann's Lemma [1]. Ackermann's Lemma states that for every formula φ of the form $\exists X(\forall \overline{x}(\alpha(\overline{x}) \to X(\overline{x})) \land \beta(X))$, where α and β are first order and X occurs only negatively in β, the formula $\beta(\alpha(\overline{x}))$ is equivalent to φ. Nonnengart and Szałas' generalisation allows positive occurrences of X in $\alpha(X, \overline{x})$. In order to deal with this more general case, they use a fixed-point operator. The equivalent formula becomes the FO[LFP] formula $\beta([\mathrm{lfp}_X\, \alpha(X, \overline{x})])$, i.e., they prove:

$$\models \exists X(\forall \overline{x}(\alpha(X, \overline{x}) \to X(\overline{x})) \land \beta(X)) \leftrightarrow \beta([\mathrm{lfp}_X\, \alpha(X, \overline{x})]).$$

We further generalise that allowing an arbitrary number of predicate variables by using a simultaneous least fixed-point, extending the scope of the result from standard semantics to abstract semantics, and adding property (2) below on the minimality of the solution. This lemma will be the main tool in the proof of the abstract fixed-point theorems.

Lemma 41. *Let $X_1, \dots, X_n$ be predicate variables. Let $\beta(X_1, \dots, X_n)$ be a first-order formula and $\Phi = (\alpha_i(X_1, \dots, X_n, \overline{x_i},))_{i=1}^n$ be an n-tuple of first-order formulas such that $|\overline{x_i}|$ equals the arity of X_i for $i \in \{1, \dots, n\}$*[1]. *If $X_1, \dots, X_n$ occur only positively in $\alpha_1, \dots, \alpha_n$ and only negatively in β, then:*

(1) $\models_{\mathrm{a}} \exists \overline{X} \left(\bigwedge_{i=1}^n \forall \overline{x_i} \left(\alpha_i(X_1, \dots, X_n, \overline{x_i}) \to X_i(\overline{x_i}) \right) \wedge \beta(X_1, \dots, X_n) \right) \leftrightarrow \beta([\mathrm{lfp}_{X_1}\ \Phi], \dots, [\mathrm{lfp}_{X_n}\ \Phi])$,
(2) *If* $(\mathcal{M}, G) \models_{\mathrm{a}} \bigwedge_{i=1}^n \forall \overline{x_i} \left(\alpha_i(R_1, \dots, R_n, \overline{x_i}) \to R_i(\overline{x_i}) \right)$ *for some model abstraction* $(\mathcal{M}, G)$ *and abstract relations* $R_1, \dots, R_n$, *then* $(\mathcal{M}, G) \models_{\mathrm{a}} \bigwedge_{i=1}^n \forall \overline{x_i} ([\mathrm{lfp}_{X_i}\ \Phi](\overline{x_i}) \to R_i(\overline{x_i}))$.

Proof. (1) '$\to$': Let $(\mathcal{M}, G)$ be a model abstraction and θ an environment such that:

$$(\mathcal{M}, G), \theta \models_{\mathrm{a}} \exists \overline{X} \bigwedge_{i=1}^n \forall \overline{x_i} \left(\alpha_i(X_1, \dots, X_n, \overline{x_i}) \to X_i(\overline{x_i}) \right) \wedge \beta(X_1, \dots, X_n).$$

Hence, there exist abstract relations $S_1, \dots, S_n$ and an environment $\theta' := \theta[X_1 := \gamma_{k_1}(S_1), \dots, X_n := \gamma_{k_n}(S_n)]$ such that:

$$(\mathcal{M}, G), \theta' \models_{\mathrm{a}} \bigwedge_{i=1}^n \forall \overline{x_i} \left(\alpha_i(X_1, \dots, X_n, \overline{x_i}) \to X_i(\overline{x_i}) \right) \wedge \beta(X_1, \dots, X_n). \tag{4}$$

Then, $(F_\Phi)_i(\gamma_{k_1}(S_1), \dots, \gamma_{k_n}(S_n)) \subseteq \gamma_{k_i}(S_i)$ for all $i \in \{1, \dots, n\}$ and as α_k, γ_k form a Galois connection for every k we have $\alpha_{k_i} \circ (F_\Phi)_i(\gamma_{k_1}(S_1), \dots, \gamma_{k_n}(S_n)) \sqsubseteq S_i$ for all $i \in \{1, \dots, n\}$. Thus, $(S_1, \dots, S_n)$ is a fixed point of $F^{\#}_{\Phi}$. Using the monotonicity of γ_k for every k, this implies:

$$(\mathcal{M}, G), \theta' \models_{\mathrm{a}} \bigwedge_{i=1}^n \forall \overline{x_i} ([\mathrm{lfp}_{X_i}\ \Phi](\overline{x_i}) \to X_i(\overline{x_i})). \tag{5}$$

Note that the FO[LFP]-formulas are well defined, as $X_1, \dots, X_n$ occur only positively in $\alpha_1, \dots, \alpha_n$. From Equation (4), we also obtain $(\mathcal{M}, G), \theta' \models_{\mathrm{a}} \beta(X_1, \dots, X_n)$, and as $X_1, \dots, X_n$ occur only negatively in β, we may use Lemma 2 to obtain:

$$(\mathcal{M}, G), \theta' \models_{\mathrm{a}} \beta([\mathrm{lfp}_{X_1}\ \Phi], \dots, [\mathrm{lfp}_{X_n}\ \Phi]).$$

This formula does not depend on $X_1, \dots, X_n$, hence $(\mathcal{M}, G), \theta \models_{\mathrm{a}} \beta([\mathrm{lfp}_{X_1}\ \Phi], \dots, [\mathrm{lfp}_{X_n}\ \Phi])$.

'$\leftarrow$': For the other direction let $(\mathcal{M}, G)$ be a model abstraction and θ an environment such that:

$$(\mathcal{M}, G), \theta \models_{\mathrm{a}} \beta([\mathrm{lfp}_{X_1}\ \Phi], \dots, [\mathrm{lfp}_{X_n}\ \Phi]).$$

Let $S_i \in V_{k_i}$ be the abstract relations defined by $[\mathrm{lfp}_{X_i}\ \Phi]$ for $i \in \{1, \dots, n\}$. $(S_1, \dots, S_n)$ is a fixed point of $F^{\#}_{\Phi}$, therefore by the definition of $F^{\#}_{\Phi}$, we have:

$$\alpha_{k_i} \circ (F_\Phi)_i(\gamma_{k_1}(S_1), \dots, \gamma_{k_n}(S_n)) \sqsubseteq S_i, \quad \text{for every } i \in \{1, \dots, n\},$$

and as α_k, γ_k form a Galois connection for all k we get:

$$(F_\Phi)_i(\gamma_{k_1}(S_1), \dots, \gamma_{k_n}(S_n)) \subseteq \gamma_{k_i}(S_i), \quad \text{for every } i \in \{1, \dots, n\}.$$

[1]Note that there could also be free variables in $\alpha_1, \dots, \alpha_n, \beta$, that are not stated explicitly.

Thus, $(\gamma_{k_1}(S_1), \dots, \gamma_{k_n}(S_n))$ is a fixed point of F_Φ and by defining $\theta' := \theta[X_1 := \gamma_{k_1}(S_1), \dots, X_n := \gamma_{k_n}(S_n)]$, we obtain:

$$(\mathcal{M}, G), \theta' \models_a \bigwedge_{i=1}^{n} \forall \overline{x_i}\, (\alpha_i(X_1, \dots, X_n, \overline{x_i}) \to X_i(\overline{x_i}))\,.$$

By assumption $(\mathcal{M}, G), \theta' \models_a \beta(X_1, \dots, X_n)$ and therefore we conclude:

$$(\mathcal{M}, G), \theta \models_a \exists \overline{X} \bigwedge_{i=1}^{n} \forall \overline{x_i}\, (\alpha_i(X_1, \dots, X_n, \overline{x_i}) \to X_i(\overline{x_i})) \land \beta(X_1, \dots, X_n).$$

We get (2) directly from Equation (5), if we choose $S_1, \dots, S_n$ to be the abstract relations $R_1, \dots, R_n$. □

We are now able to prove the abstract fixed-point theorem, the main result of this paper. For clarity, we restate the theorem.

Theorem 32 (Abstract Horn Fixed-Point Theorem). *Let $\exists \overline{X} \psi$ be a Horn formula equation and $\mu_j := [\mathrm{lfp}_{X_j}\, \Phi_\psi]$ for $j \in \{1, \dots, n\}$, then:*

(1) $\models_a \exists \overline{X}\, \psi \leftrightarrow \psi[\overline{X} \backslash \overline{\mu}]$ *and*
(2) *if $(\mathcal{M}, G) \models_a \psi[\overline{X} \backslash \overline{R}]$ for some model abstraction $(\mathcal{M}, G)$ and abstract relations $R_1, \dots, R_n$, then $(\mathcal{M}, G) \models_a \bigwedge_{j=1}^{n} (\mu_j \to R_j)$.*

Proof. '$\to$': Assume that $(\mathcal{M}, G) \models_a \exists \overline{X} \psi$. We first note that in the transformation of Lemma 31 there does not occur any second-order quantifiers nor fixed point operators. Thus, it also holds for abstract models and we obtain:

$$(\mathcal{M}, G) \models_a \exists \overline{X} \bigwedge_{j=1}^{n} \forall \overline{x_j} (\varphi_j(X_1, \dots, X_n, \overline{x_j}) \to X_j(\overline{x_j})) \land \forall \overline{y} \bigwedge_{C \in E} C(X_1, \dots, X_n, \overline{y}),$$

where $X_1, \dots, X_n$ occur only positively in $\varphi_1, \dots, \varphi_n$ and only negatively in the clauses C. Hence, we can apply Lemma 41/1 to obtain:

$$(\mathcal{M}, G) \models_a \forall \overline{y} \bigwedge_{C \in E} C(\mu_1, \dots, \mu_n, \overline{y}).$$

By construction, the FO[LFP] formulas $\mu_1, \dots, \mu_n$ also satisfy the base and induction clauses, i.e., for any base or induction clause $C(X_1, \dots, X_n, \overline{y})$ it holds $(\mathcal{M}, G) \models_a \forall \overline{y}\, C(\mu_1, \dots, \mu_n, \overline{y})$. Therefore, it follows that:

$$(\mathcal{M}, G) \models_a \psi[X_1 \backslash \mu_1, \dots, X_n \backslash \mu_n].$$

The other direction '$\leftarrow$' is immediate. The second part of the theorem follows directly from Lemma 41/2. □

We also present the fixed-point theorem for classical semantics. This has been proven directly in [41], here it is a corollary of the abstract fixed-point theorem as every classical model is also an abstract model if the abstraction and concretisation functions are chosen as the identity functions.

Corollary 42. *Let $\exists \overline{X} \psi$ be a Horn formula equation and $\mu_j := [\mathrm{lfp}_{X_j}\, \Phi_\psi]$ for $j \in \{1, \dots, n\}$, then:*

(1) $\models \exists \overline{X}\, \psi \leftrightarrow \psi[\overline{X} \backslash \overline{\mu}]$ *and*
(2) *if $\mathcal{M} \models \psi[\overline{X} \backslash \overline{R}]$ for some structure $\mathcal{M}$ and relations $R_1, \dots, R_n$ in $\mathcal{M}$, then $\mathcal{M} \models \bigwedge_{j=1}^{n} (\mu_j \to R_j)$.*

Remark 43. Note that Corollary 42 applies equally to constraints being FO[LFP]-formulas. It therefore shows that FO[LFP], in contrast to first-order logic, has the property of being closed under solving Horn formula equations. It thus shows that in FO[LFP] validity and solvability of Horn formula equations coincide. This is in contrast to formula equations in first-order logic, cf. Example 24.

Remark 44. One of the most prominent approaches to solving an SOQE problem $\exists \overline{X}\, \varphi$ is to solve the corresponding WSOQE problem to obtain formulas $\overline{\chi}$ such that $\models \exists \overline{X}\, \varphi \leftrightarrow \varphi[\overline{X} \backslash \overline{\chi}]$. For example, it underlies the family of DLS algorithms: the original DLS [14], DLS* [15, 50] and DLS$'$ [17]. In this context part (1) of Corollary 42 can be understood as yielding a solution to the WSOQE- and hence the SOQE-problem for Horn formula equations in FO[LFP].

4.3 The Dual and Linear-Horn Fixed-Point Theorems

We now turn our interest to dual-Horn and linear-Horn formula equations. Recall that a constrained dual-Horn clause consists of at most one positive predicate variable, in contrast to constrained Horn clauses, where at most one negative predicate variables occurs. Thus, dual-Horn formula equations are dual to Horn formula equations.

For a formula ψ, we define ψ^D as $\psi[X_1 \backslash \neg X_1, \dots, X_n \backslash \neg X_n]$, where $X_1, \dots, X_n$ are all predicate variables occurring in ψ. Note that $\psi \equiv (\psi^D)^D$ for all formulas ψ. Moreover, note that $\models \exists \overline{X}\, \psi \leftrightarrow \exists \overline{X}\, \psi^D$. If $\exists \overline{X}\, \psi$ is a Horn formula equation, then $\exists \overline{X}\, \psi^D$ is logically equivalent to a dual-Horn formula equation and if $\exists \overline{X}\, \varphi$ is a dual-Horn formula equation, then $\exists \overline{X}\, \varphi^D$ is logically equivalent to a Horn formula equation. Note that dualisation of a (dual) Horn formula equation interchanges (B)- and (E)-clauses.

Example 45. Consider the constrained dual-Horn clauses in the language of arithmetic:

$$\psi \quad \equiv \quad X(1) \wedge (X(n) \rightarrow X(n+1) \vee X(n+2)) \wedge \neg X(4).$$

The dualisation of ψ is:

$$\psi^D \quad \equiv \quad \neg X(1) \wedge (\neg X(n) \rightarrow \neg X(n+1) \vee \neg X(n+2)) \wedge X(4),$$

which is logically equivalent to the constrained Horn clauses:

$$\neg X(1) \wedge (X(n+1) \wedge X(n+2) \rightarrow X(n)) \wedge X(4).$$

If $\exists \overline{X} \psi$ is a dual-Horn formula equation, then $\exists \overline{X} \psi^D$ is equivalent to a Horn formula equation. Now let $\overline{\mu}$ be the least solution of $\exists \overline{X} \psi^D$, then $\overline{\neg \mu}$ is the greatest solution of $\exists \overline{X} \psi$. Thus, the dual-Horn fixed-point theorem follows from the Horn case in Theorem 32.

Theorem 46 (Abstract Dual-Horn Fixed-Point Theorem). *Let $\exists \overline{X} \psi$ be a dual-Horn formula equation and $\nu_j := \neg[\mathrm{lfp}_{X_j}\, \Phi_{\psi^D}]$ for $j \in \{1, \dots, n\}$, then:*

1. $\models_{\mathrm{a}} \exists \overline{X}\, \psi \leftrightarrow \psi[\overline{X} \backslash \overline{\nu}]$ *and*
2. *if $(\mathcal{M}, G) \models_{\mathrm{a}} \psi[\overline{X} \backslash \overline{R}]$ for some model abstraction $(\mathcal{M}, G)$ and abstract relations $R_1, \dots, R_n$, then $(\mathcal{M}, G) \models_{\mathrm{a}} \bigwedge_{j=1}^{n} (R_j \rightarrow \nu_j)$.*

Proof. Let $\mu_j := [\mathrm{lfp}_{X_j}\, \Phi_{\psi^D}]$ for $j = 1, \dots, n$. For (1) note that, since $\exists \overline{X}\, \psi$ is a dual Horn formula equation, $\exists \overline{X}\, \psi^D$ is logically equivalent to a Horn formula equation. An application of Theorem 32.1. yields $\models_{\mathrm{a}} \exists \overline{X}\, \psi^D \leftrightarrow \psi^D[\overline{X} \backslash \overline{\mu}]$. Since $\psi^D[\overline{X} \backslash \overline{\mu}]$ is syntactically equal to $\psi[\overline{X} \backslash \overline{\nu}]$, we obtain $\models_{\mathrm{a}} \exists \overline{X}\, \psi \leftrightarrow \exists \overline{X} \psi^D \leftrightarrow \psi^D[\overline{X} \backslash \overline{\mu}] \leftrightarrow \psi[\overline{X} \backslash \overline{\nu}]$.

For (2) assume that $(\mathcal{M}, G) \models_{\mathrm{a}} \psi[X_1\backslash R_1, ..., X_n\backslash R_n]$ for some model abstraction $(\mathcal{M}, G)$ and abstract relations $R_1, ..., R_n$. Then, $(\mathcal{M}, G) \models_{\mathrm{a}} \psi^D[X_1\backslash\neg R_1, ..., X_n\backslash\neg R_n]$, so, by Theorem 32/2., $\mathcal{M} \models \bigwedge_{j=1}^n(\mu_j \to \neg R_j)$ which yields $(\mathcal{M}, G) \models_{\mathrm{a}} \bigwedge_{j=1}^n(R_j \to \nu_j)$ by contraposition. □

Note that the operator induced by Φ_{ψ^D} is not the dual operator of the one induced by Φ_ψ in the sense of [20] because Φ_{ψ^D} is not the (pointwise) negation of Φ_ψ. Therefore, ν is not the greatest fixed point of Φ_ψ.

As a constrained linear-Horn clause is both a constrained Horn and a constrained dual-Horn clause, we can combine the two fixed-point theorems and we obtain for the case of linear-Horn formula equations:

Theorem 47 (Abstract linear-Horn fixed-point theorem). *Let $\exists\overline{X}\psi$ be a linear-Horn formula equation, $\mu_j := [\mathrm{lfp}_{X_j}\ \Phi_\psi]$ and $\nu_j := \neg[\mathrm{lfp}_{X_j}\ \Phi_{\psi^D}]$ for $j \in \{1, ..., n\}$, then:*

(1) *$\models_{\mathrm{a}} \exists\overline{X}\ \psi \leftrightarrow \psi[\overline{X}\backslash\overline{\mu}]$ and $\models_{\mathrm{a}} \exists\overline{X}\ \psi \leftrightarrow \psi[\overline{X}\backslash\overline{\nu}]$ and*
(2) *if $(\mathcal{M}, G) \models_{\mathrm{a}} \psi[\overline{X}\backslash\overline{R}]$ for some model abstraction $(\mathcal{M}, G)$ and abstract relations $R_1, ..., R_n$, then $(\mathcal{M}, G) \models_{\mathrm{a}} \bigwedge_{j=1}^n(\mu_j \to R_j \wedge R_j \to \nu_j)$.*

An *interpolant* of two variable-free tuples of FO[LFP] formulas $\overline{\mu}$ and $\overline{\nu}$ in the same first-order language is a tuple of first-order formulas $\overline{\chi}$ such that $\models \bigwedge_{j=1}^n(\mu_j \to \chi_j \wedge \chi_j \to \nu_j)$. Theorem 47/2 states that every solution of the linear-Horn formula equation is an interpolant of $\overline{\mu}$ and $\overline{\nu}$. The converse is only true if we add another assumption.

Corollary 48. *Let $\exists\overline{X}\psi$ be a linear-Horn formula equation, $\mu_j := [\mathrm{lfp}_{X_j}\ \Phi_\psi]$ and $\nu_j := \neg[\mathrm{lfp}_{X_j}\ \Phi_{\psi^D}]$ for $j \in \{1, ..., n\}$, then:*

(1) *Every solution of $\exists\overline{X}\psi$ is an interpolant of $\overline{\mu}$ and $\overline{\nu}$,*
(2) *an interpolant $\overline{\chi}$ is a solution of $\exists\overline{X}\psi$ iff $\overline{\chi}$ satisfies all induction clauses.*

Proof. (1) is an immediate consequence of Theorem 47/2. For (2), we first note that $\overline{X}$ occur only positively in the base clauses and only negatively in the end clauses. Thus, we can use Lemma 2 to obtain that the interpolant $\overline{\chi}$ satisfies all base and end clauses, hence $\overline{\chi}$ is a solution iff it also satisfies all induction clauses. □

Interpolation is an important technique for solving constrained Horn clauses, see, e.g., [48]. The above result provides a theoretical connection between interpolation and verification. The relationship between interpolation and Horn clauses has also been studied by encoding interpolation problems with a language condition on the constant symbols as Horn clause sets [25, 55].

5 Decidability of the Affine Solution Problem

As an application of the abstract fixed-point theorem, we take a look at the affine solution problem, which is shown to be decidable in [30]. This result is based on a generalisation of Karr's algorithm [40] to non-Horn formula equations. It is proved by computing a fixed point similarly to how we did in Section 4. The main difference is that the fixed point is not computed on the logical level, but in the lattice of affine subspaces of $\mathbb{Q}^n$. The abstract fixed point theorem shown in Section 4 applies to this problem and drastically shortens the proof of decidability.

The setting of the affine solution problem is the language $\mathcal{L}_{\mathrm{aff}} = (0, 1, +, (c)_{c\in\mathbb{Q}})$ as discussed in Example 38. As we are only working in $\mathbb{Q}$, we can assume without loss of generality that every term $t(x_1, ..., x_n)$ is of the form $c_0 + \sum_{i=1}^m c_i x_i$ and every atomic formula $A(x_1, ..., x_n)$ is of the form $c_0 + \sum_{i=1}^m c_i x_i = 0$. We call such atomic formulas *linear equations* and conjunctions of linear equations *linear equation systems*. It is well known that linear equation systems define affine subspaces of $\mathbb{Q}^n$.

Definition 49. An *affine formula equation* is a formula equation of the form $\exists \overline{X}\ \psi$, where ψ is a quantifier-free first-order formula in $\mathcal{L}_{\text{aff}} \cup \{X_1, \ldots, X_n\}$.

Definition 50. The affine solution problem is the set of affine formula equations that have solutions in the class of linear equation systems in $\mathbb{Q}$, i.e. the set:

$$\{\exists \overline{X}\psi \mid \text{There exist linear equation systems } F_1, \ldots, F_n \text{ such that } \mathbb{Q} \models \psi[X_1 \backslash F_1, \ldots, X_n \backslash F_n]\}.$$

Recall from Example 38 the set domain $G_{\text{aff}} = ((\text{Aff}\ \mathbb{Q}^k, \subseteq), \text{aff}_k, \text{id}_k)_{k \in \mathbb{N}}$, where $\text{Aff}\ \mathbb{Q}^k$ is the set of all affine subspaces of $\mathbb{Q}^k$, aff_k maps every subset of $\mathbb{Q}^k$ to its affine hull and id_k is the embedding of $\text{Aff}\ \mathbb{Q}^k$ in $\mathcal{P}(\mathbb{Q}^k)$. Then, $(\mathbb{Q}, G_{\text{aff}})$ is a model abstraction. G_{aff} is defined such that an affine formula equation $\exists \overline{X}\ \psi$ is in the affine solution problem iff $(\mathbb{Q}, G_{\text{aff}}) \models_{\text{a}} \exists \overline{X}\ \psi$. Thus, we can use the abstract fixed-point theorem.

Theorem 51. *The affine solution problem is decidable.*

Proof. As in [30], we reduce the solvability of a formula equation $\exists \overline{X}\psi$ to the solvability of one of its finitely many projections, which are Horn formula equations. Let $\exists \overline{X}\varphi$ be one of them, then we may apply Theorem 32. This yields a tuple of FO[LFP]-formulas $\overline{\mu}$ such that $(\mathbb{Q}, G_{\text{aff}}) \models_{\text{a}} \exists \overline{X}\ \varphi \leftrightarrow \varphi[\overline{X} \backslash \overline{\mu}]$. Since all lattices $\text{Aff}\ \mathbb{Q}^k$ have finite height, we can compute fixed-point free formulas $\overline{\chi}$ equivalent to $\overline{\mu}$ from $\overline{\mu}$.

For notational simplicity, we describe how this is achieved in the case of one predicate variable X. Let $\mu = [\text{lfp}_X\ \Phi_\varphi]$ be an FO[LFP]-formula. As in Section 2.2.3, μ can be approximated by the sequence of sets in V_k:

$$S^0 := \emptyset,$$
$$S^{l+1} := F_\Phi^\#(S^l).$$

These sets describe increasing affine subspaces of M^k, thus, as the lattice $\text{Aff}\ \mathbb{Q}^k$ has height k, it holds $F_\Phi^\#(S^k) = S^k$ and therefore $\overline{a} \in S^k \Leftrightarrow (\mathbb{Q}, G_{\text{aff}}) \models_{\text{a}} \mu(\overline{a})$. In the actual computation, the affine subspace S^i is represented by a linear equation system, i.e., a formula σ_i. The computation of $F_\Phi^\#$ is based on the fact that, given finite representations of affine spaces $\mathcal{A}$ and $\mathcal{B}$ and of an affine transformation $\mathcal{T}$, it is possible to compute finite representations of $\mathcal{A} \cap \mathcal{B}$, $\mathcal{A} \cup \mathcal{B}$, $\mathcal{T}(\mathcal{A})$ and $\mathcal{T}^{-1}(\mathcal{A})$. This is an elementary result of affine geometry and linear algebra, see [57]. Hence, σ_k is the desired first-order formula such that $(\mathbb{Q}, G_{\text{aff}}) \models_{\text{a}} \mu(\overline{a}) \Leftrightarrow \mathbb{Q} \models \sigma_k(\overline{x})$.

Therefore, $(\mathbb{Q}, G_{\text{aff}}) \models_{\text{a}} \exists \overline{X}\ \varphi \leftrightarrow \varphi[\overline{X} \backslash \overline{\chi}]$. Now $\varphi[\overline{X} \backslash \overline{\chi}]$ is a first-order formula and hence $(\mathbb{Q}, G_{\text{aff}}) \models_{\text{a}} \varphi[\overline{X} \backslash \overline{\chi}]$ iff $\mathbb{Q} \models \varphi[\overline{X} \backslash \overline{\chi}]$. The latter statement can be checked by a decision procedure for linear arithmetic. □

Karr's algorithm was extended to the computation of polynomial invariants in [52, 53], see also [34, 35]. The authors do not know whether the analogue of Theorem 51 for the case of polynomials is true. If it is, a different proof strategy will be necessary because the reduction to projections is not possible in the case of polynomials.

6 Applications to Program Verification

In this section, we will describe some direct applications of our fixed-point theorems to the foundations of program verification. As an exemplary framework, we will consider the Hoare calculus for a simple imperative programming language as in [65]. The two main goals of this section are:

(1) We show that proving a partial correctness assertion amounts to solving a linear Horn formula equation in Theorem 65.

   - (a) This allows to express the partial correctness of a program as an FO[LFP] formula, see Corollary 66.
   - (b) We show that the canonical solutions of this linear Horn formula equation correspond to the least precondition and strongest postcondition in Theorems 67 and 68.
- (2) We give an alternative proof of completeness of Hoare logic based on our abstract fixed-point theorem in Theorem 72 which has the following advantages:
   - (a) It does not need the expressivity hypothesis of the standard proof for $\mathbb{Z}$.
   - (b) It accommodates abstract interpretation in the form of a restriction on the syntactic form of the loop invariants.

6.1 Hoare Triples

In the study of program verification, it is quite common to only work in the language of arithmetic and the structure $\mathbb{Z}$. This may disguise the importance of the expressivity of $\mathbb{Z}$. As we are aiming for a framework where the usage of FO[LFP]-formulas replaces the encoding of finite sequences, we work in a more general setting.

Definition 52. Let $\mathcal{L}$ be a language. The set of programs in $\mathcal{L}$ is defined by:

$$p ::= \textbf{skip} \mid x := t \mid p_0; p_1 \mid \textbf{if } B \textbf{ then } p_0 \textbf{ else } p_1 \mid \textbf{while } B \textbf{ do } p_0,$$

where t is an $\mathcal{L}$-term, B a quantifier-free first-order formula in $\mathcal{L}$ and x is a program variable.

Definition 53. A *Hoare triple* is a triple (φ, p, ψ) consisting of a program p and two first-order formulas φ and ψ. This is traditionally denoted as $\{\varphi\}p\{\psi\}$.

Definition 54. Let $\mathcal{L}$ be a language and $\mathcal{M}$ be a structure with domain M. Define $\mathrm{Var} = \{x_0, x_1, \ldots\}$ to be the set of variables which may occur in a program. A function $\sigma : \mathrm{Var} \to M$ is called a *state.*[2] We denote the set of all states with Σ. For $m \in M$, we write $\sigma[x_j \to m]$ for the unique state σ' such that $\sigma'(x_j) = m$ and $\sigma'(x_i) = \sigma(x_i)$ for $i \neq j$.

Definition 55 (Denotational Semantics). For every program p, we define a relation $C(p)$ on $\Sigma \times \Sigma$ by structural induction:

$$\begin{aligned}
C(\textbf{skip}) &= \{(\sigma, \sigma) \mid \sigma \in \Sigma\} \\
C(x_j := t) &= \{(\sigma, \sigma[x_j \to m]) \mid \sigma \in \Sigma,\ m \in M \text{ and } \mathcal{M}, \sigma \models t = m\} \\
C(p_0; p_1) &= C(p_1) \circ C(p_0) \\
C(\textbf{if } B \textbf{ then } p_0 \textbf{ else } p_1) &= \{(\sigma, \sigma') \mid \mathcal{M}, \sigma \models B \text{ and } (\sigma, \sigma') \in C(p_0)\} \cup \\
&\quad\ \{(\sigma, \sigma') \mid \mathcal{M}, \sigma \models \neg B \text{ and } (\sigma, \sigma') \in C(p_1)\} \\
C(\textbf{while } B \textbf{ do } p_0) &= \mathrm{lfp}(\Gamma),
\end{aligned}$$

where Γ is an operator on $\Sigma \times \Sigma$ defined as:

$$\begin{aligned}
\Gamma(X) = \{&(\sigma, \sigma') \mid \mathcal{M}, \sigma \models B \text{ and } (\sigma, \sigma') \in X \circ C(p_0)\} \cup \\
&\{(\sigma, \sigma) \mid \mathcal{M}, \sigma \models \neg B\}.
\end{aligned}$$

We see that Γ is a monotone operator and thus $\mathrm{lfp}(\Gamma)$ is well defined. Moreover, we can convince ourselves that $C(p)$ is actually a partial function from $\Sigma \to \Sigma$. If $C(p)(\sigma)$ is not defined, it means that a while-loop is not terminating. To make it a total function, we extend the set of states Σ with the state $\perp$, which is associated with a non-terminating computation, i.e., we define $\Sigma_\perp := \Sigma \cup \{\perp\}$.

[2]Later, we will talk of formulas, in which the variables of a program occur. In this sense, a state may also be seen as an environment.

For every σ such that $C(p)(\sigma)$ is not defined yet, we define $C(p)(\sigma) := \bot$ and in that way $C(p)$ is a total function from $\Sigma \to \Sigma_\bot$.

Now, we define the meaning of partial correctness of a Hoare triple:

Definition 56 (Semantics of Hoare Triples). Let $\mathcal{L}$ be a language and $\mathcal{M}$ be a structure. Let σ be a state, for a Hoare triple $\{\varphi\}p\{\psi\}$ define:

$$\mathcal{M}, \sigma \models \{\varphi\}p\{\psi\} \quad \text{if} \quad \mathcal{M}, \sigma \models \varphi \;\Rightarrow\; \mathcal{M}, C(p)(\sigma) \models \psi.$$

Here, $\mathcal{M}, \bot \models \psi$ is defined to be true. In doing so, we only check validity for states, in which the computation terminates. Now define:

$$\mathcal{M} \models \{\varphi\}p\{\psi\} \quad \text{if} \quad \forall \sigma \in \Sigma: \; \mathcal{M}, \sigma \models \{\varphi\}p\{\psi\}.$$

Note that we can define $\mathcal{M} \models \{R\}p\{S\}$ analogously for relations R and S. This will be of use in some proofs.

6.2 Verification Condition

We define a different semantics of Hoare triples using the verification condition, which turns out to be equivalent to the usual semantics. As the verification condition is a linear-Horn formula equation, we are able to apply our results from Section 4. In particular, we are able to express partial correctness of a Hoare triple as an FO[LFP]-formula.

Definition 57. The *verification condition* of a Hoare triple $\{\varphi\}p\{\psi\}$, written $\mathrm{vc}(\{\varphi\}p\{\psi\})$, is a formula equation $\exists \bar{I} \forall \bar{x}\ \tilde{\mathrm{vc}}(\{\varphi\}p\{\psi\})$, where $\tilde{\mathrm{vc}}(\{\varphi\}p\{\psi\})$ is defined by structural induction on p. Here, I is a fresh new formula variable, which does not appear in φ nor ψ:

$$\begin{aligned}
\tilde{\mathrm{vc}}(\{\varphi\}\mathbf{skip}\{\psi\}) &= (\varphi \to \psi)\\
\tilde{\mathrm{vc}}(\{\varphi\}x_j := t\{\psi\}) &= (\varphi \to \psi[x_j \backslash t])\\
\tilde{\mathrm{vc}}(\{\varphi\}p_0; p_1\{\psi\}) &= \tilde{\mathrm{vc}}(\{\varphi\}p_0\{I\}) \land \tilde{\mathrm{vc}}(\{I\}p_1\{\psi\}),\\
\tilde{\mathrm{vc}}(\{\varphi\}\mathbf{if}\ B\ \mathbf{then}\ p_0\ \mathbf{else}\ p_1\{\psi\}) &= \tilde{\mathrm{vc}}(\{\varphi \land B\}p_0\{\psi\}) \land \tilde{\mathrm{vc}}(\{\varphi \land \neg B\}p_1\{\psi\})\\
\tilde{\mathrm{vc}}(\{\varphi\}\mathbf{while}\ B\ \mathbf{do}\ p_0\{\psi\}) &= \tilde{\mathrm{vc}}(\{I \land B\}p_0\{I\}) \land (\varphi \to I) \land (I \land \neg B \to \psi).
\end{aligned}$$

Then, $\mathrm{vc}(\{\varphi\}p\{\psi\}) = \exists \bar{I} \forall \bar{x}\ \tilde{\mathrm{vc}}(\{\varphi\}p\{\psi\})$ is defined by universal quantification of every individual variable occurring in $\tilde{\mathrm{vc}}(\{\varphi\}p\{\psi\})$ and existential quantification of every predicate variable in $\tilde{\mathrm{vc}}(\{\varphi\}p\{\psi\})$.

Note that there are translations which are more efficient in practice, for example, in dealing with composition. However, since we aim for theoretical results in this paper, we have opted for this simple translation. Also note that this is a purely syntactic definition, thus we can define $\mathrm{vc}(\{R\}p\{S\})$ analogously for a program p and predicate variables R, S.

Lemma 58. *Let $\{\varphi\}p\{\psi\}$ be Hoare triple. Then, $\mathrm{vc}(\{\varphi\}p\{\psi\})$ is a linear-Horn formula equation.*

Proof. The verification condition is a formula equation $\exists \bar{I} \forall \bar{x}\ \tilde{\mathrm{vc}}(\{\varphi\}p\{\psi\})$, where $\tilde{\mathrm{vc}}(\{\varphi\}p\{\psi\})$ is a conjunction of clauses. Each clause has the form $\gamma \lor \neg C \lor D$, where γ is a first-order formula (it is a disjunction of the first-order formulas B or $\neg B$) and C and D are either predicate variables or the first-order formulas φ or ψ. □

Lemma 59. *Let $\{\varphi\}p\{\psi\}$ be a Hoare triple. Then, φ occurs only negatively and ψ occurs only positively in $\mathrm{vc}(\{\varphi\}p\{\psi\})$, respectively.*

Theorem 60. *Let $\mathcal{L}$ be a language and $\mathcal{M}$ be a structure. Let $\{\varphi\}p\{\psi\}$ be a Hoare triple. Then:*

$$\mathcal{M} \models \mathrm{vc}(\{\varphi\}p\{\psi\}) \quad \Rightarrow \quad \mathcal{M} \models \{\varphi\}p\{\psi\}.$$

Proof. This theorem is shown by structural induction on p. For a detailed proof see [41]. □

The next aim is to show the converse direction of Theorem 60, i.e., that $\mathcal{M} \models \{\varphi\}p\{\psi\}$ implies $\mathcal{M} \models \mathrm{vc}(\{\varphi\}p\{\psi\})$. To prove this, we need the concept of the weakest precondition.

Definition 61. Let $\mathcal{L}$ be a language and $\mathcal{M}$ be a structure. Let p be a program and φ, ψ be first-order formulas in $\mathcal{L}$. The *weakest precondition*[3] of p and ψ, written $\mathrm{wp}(p, \psi)$, is defined as:

$$\mathrm{wp}(p, \psi) = \{\sigma \in \Sigma \mid \mathcal{M}, C(p)(\sigma) \models \psi\}.$$

For technical reasons, we also define the relation $R_{\mathrm{wp}}(p, \psi)$ defined by the weakest precondition, i.e.,

$$\mathcal{M}, \sigma \models R_{\mathrm{wp}}(p, \psi) \;\Leftrightarrow\; \mathcal{M}, C(p)(\sigma) \models \psi.$$

The *strongest postcondition* of p and φ, written $\mathrm{sp}(p, \varphi)$, is defined as:

$$\mathrm{sp}(p, \varphi) = \{\sigma \in \Sigma \mid \exists \sigma' \in \Sigma : \mathcal{M}, \sigma' \models \varphi \text{ and } C(p)(\sigma') = \sigma\}.$$

Definition 62. Let $\mathcal{L}$ be a language and $\mathcal{M}$ be a structure. For every formula φ in $\mathcal{L}$ define a subset of Σ:

$$[\varphi] := \{\sigma \in \Sigma \mid \mathcal{M}, \sigma \models \varphi\}.$$

Similarly, we define for a relation R in $\mathcal{M}$:

$$[R] := \{\sigma \in \Sigma \mid \mathcal{M}, \sigma \models R\}.$$

In particular $[R_{\mathrm{wp}}(p, \psi)] = \mathrm{wp}(p, \psi)$.

The following properties of the weakest precondition and the strongest postcondition justify the terminology and are of fundamental importance.

Lemma 63. *Let $\mathcal{L}$ be a language and $\mathcal{M}$ be a structure. Let $\{\varphi\}p\{\psi\}$ be a Hoare triple. Then*:

(a) $\mathcal{M} \models \{\varphi\}p\{\psi\}$ *iff* $[\varphi] \subseteq \mathrm{wp}(p, \psi)$,
(b) $\mathcal{M} \models \{\varphi\}p\{\psi\}$ *iff* $[\psi] \supseteq \mathrm{sp}(p, \varphi)$.

Lemma 64. *Let $\mathcal{L}$ be a language and $\mathcal{M}$ be a structure. Let p be a program and ψ be a first-order formula. Then, $\mathcal{M} \models \mathrm{vc}(\{R_{\mathrm{wp}}(p, \psi)\}p\{\psi\})$.*

Proof. This is proven by structural induction on the program p. We prove the cases $p \equiv p_0; p_1$ and $p \equiv$ **while** B **do** p_0, the others are routine.

Let $p \equiv p_0; p_1$. We ought to show that $\mathcal{M} \models \exists I\ \mathrm{vc}(\{R_{\mathrm{wp}}(p, \psi)\}p_0\{I\}) \land \mathrm{vc}(\{I\}p_1\{\psi\})$. By the induction hypothesis it holds:

$$\begin{aligned} &\mathcal{M} \models \mathrm{vc}(\{R_{\mathrm{wp}}(p_1, \psi)\}p_1\{\psi\}) \quad \text{and} \\ &\mathcal{M} \models \mathrm{vc}(\{R_{\mathrm{wp}}(p_0, R_{\mathrm{wp}}(p_1, \psi))\}p_0\{R_{\mathrm{wp}}(p_1, \psi)\}). \end{aligned}$$

[3]In the literature, this is mostly called weakest liberal precondition, and the term weakest precondition is reserved for the context of total correctness. As we only talk about relative correctness of programs, there is no need for us to do so.

We are finished if we can show that $R_{\mathrm{wp}}(p,\psi) = R_{\mathrm{wp}}(p_0, R_{\mathrm{wp}}(p_1,\psi))$, yet this follows from:

$$\begin{aligned}\mathcal{M},\sigma \models R_{\mathrm{wp}}(p,\psi) &\Leftrightarrow \mathcal{M}, C(p_0;p_1)(\sigma) \models \psi\\ &\Leftrightarrow \mathcal{M}, C(p_1)\circ C(p_0)(\sigma) \models \psi\\ &\Leftrightarrow \mathcal{M}, C(p_0)(\sigma) \models R_{\mathrm{wp}}(p_1,\psi)\\ &\Leftrightarrow \mathcal{M},\sigma \models R_{\mathrm{wp}}(p_0, R_{\mathrm{wp}}(p_1,\psi)).\end{aligned}$$

Now consider the case $p \equiv \textbf{while } B \textbf{ do } p_0$. For shorter notation, we define $R = R_{\mathrm{wp}}(p,\psi)$. We have to show that $\mathcal{M} \models \exists I\, \mathrm{vc}(\{I \land B\}p_0\{I\}) \land (R \to I) \land (I \land \neg B \to \psi)$. It suffices to show that:

(1) $\mathcal{M} \models \mathrm{vc}(\{R \land B\}p_0\{R\})$ and
(2) $\mathcal{M} \models (R \land \neg B) \to \psi$.

To prove these two claims, we use the following fact, cf. [65, Proposition 5.1], for a program of the form $p \equiv \textbf{while } B \textbf{ do } p_0$:

$$C(p) \equiv C(\textbf{if } B \textbf{ then } p_0; p \textbf{ else skip}). \tag{6}$$

We now prove (1). The induction hypothesis states $\mathcal{M} \models \mathrm{vc}(\{R_{\mathrm{wp}}(p_0,R)\}p_0\{R\})$, hence it remains to show that $\mathcal{M} \models (R \land B) \to R_{\mathrm{wp}}(p_0,R)$. This follows as:

$$\begin{aligned}\mathcal{M},\sigma \models R_{\mathrm{wp}}(p,\psi) \land B &\Leftrightarrow \mathcal{M}, C(p)(\sigma) \models \psi \land \mathcal{M},\sigma \models B\\ &\overset{(6)}{\Rightarrow} \mathcal{M}, C(p_0,p)(\sigma) \models \psi\\ &\Leftrightarrow \mathcal{M}, C(p)\circ C(p_0)(\sigma) \models \psi\\ &\Leftrightarrow \mathcal{M}, C(p_0)(\sigma) \models R_{\mathrm{wp}}(p,\psi)\\ &\Leftrightarrow \mathcal{M},\sigma \models R_{\mathrm{wp}}(p_0, R_{\mathrm{wp}}(p,\psi)).\end{aligned}$$

For the second claim, let σ be a state such that $\mathcal{M},\sigma \models R \land \neg B$. By the definition of R, we have $\mathcal{M}, C(p)(\sigma) \models \psi$, thus Equation (6) yields $\mathcal{M}, \textbf{skip}(\sigma) \models \psi$, which translates to $\mathcal{M},\sigma \models \psi$. □

Theorem 65. *Let $\mathcal{L}$ be a language and $\mathcal{M}$ be a structure. Let $\{\varphi\}p\{\psi\}$ be a Hoare triple. Then:*

$$\mathcal{M} \models \{\varphi\}p\{\psi\} \quad\Rightarrow\quad \mathcal{M} \models \mathrm{vc}(\{\varphi\}p\{\psi\}).$$

Proof. By Lemma 63 the assumption $\mathcal{M} \models \{\varphi\}p\{\psi\}$ is equivalent to $\mathcal{M} \models \varphi \to R_{\mathrm{wp}}(p,\psi)$. Lemma 64 states that $\mathcal{M} \models \mathrm{vc}(\{R_{\mathrm{wp}}(p,\psi)\}p\{\psi\})$ and, as $R_{\mathrm{wp}}(p,\psi)$ occurs only negatively in $\mathrm{vc}(\{R_{\mathrm{wp}}(p,\psi)\}p\{\psi\})$, Lemma 2 concludes $\mathcal{M} \models \mathrm{vc}(\{\varphi\}p\{\psi\})$. □

As $\mathrm{vc}(\{\varphi\}p\{\psi\})$ is a linear-Horn formula equation, we can now obtain the statement that a partial correctness assertion of an imperative program is expressible as a formula in FO[LFP], a point also made in [9] for existential least fixed-point logic.

Corollary 66. *Let $\mathcal{L}$ be a language and $\mathcal{M}$ be a structure. Let $\{\varphi\}p\{\psi\}$ be a Hoare triple. Consider the linear-Horn formula equation $\exists \bar{I}\ \pi \ \equiv\ \mathrm{vc}(\{\varphi\}p\{\psi\})$ with predicate variables $I_1,\dots,I_n$. Define $\mu_j := [\mathrm{lfp}_{I_j}\ \Phi_\pi]$ and $\nu_j := \neg[\mathrm{lfp}_{I_j}\ \Phi_{\pi^D}]$ for $j \in \{1,\dots,n\}$. Then:*

(1) $\mathcal{M} \models \mathrm{vc}(\{\varphi\}p\{\psi\})$ *iff* $\mathcal{M} \models \tilde{\mathrm{vc}}(\{\varphi\}p\{\psi\})[\bar{I}\backslash\bar{\mu}]$ *iff* $\mathcal{M} \models \tilde{\mathrm{vc}}(\{\varphi\}p\{\psi\})[\bar{I}\backslash\bar{\nu}]$.
(2) *If* $\mathcal{M} \models \tilde{\mathrm{vc}}(\{\varphi\}p\{\psi\})[\bar{I}\backslash\bar{R}]$ *for relations* $R_1,\dots,R_n$*, then* $\mathcal{M} \models \bigwedge_{j=1}^{n} \mu_j \to R_j \land R_j \to \nu_j$.

6.3 Weakest Precondition and Strongest Postcondition

By defining the equivalent semantics based on the verification condition, which is a linear-Horn formula equation, we now get some corollaries of our linear-Horn fixed-point theorem. We will see that the canonical solutions correspond to the weakest precondition and strongest postcondition.

In the integers, it is well known that for any program p and any formula ψ there is a first-order formula φ_{wp} which defines the weakest precondition, i.e., $[\varphi_{\mathrm{wp}}] = \mathrm{wp}(p, \psi)$ and, symmetrically, for any program p and any formula φ, there is a first-order formula ψ_{sp} which defines the strongest postcondition, i.e., $[\psi_{\mathrm{sp}}] = \mathrm{sp}(p, \varphi)$. Note that these formulas rely on the expressivity of the assertion language, i.e., in this setting, on an encoding of finite sequences in $\mathbb{Z}$. By allowing FO[LFP] formulas, we are not dependent on the expressivity of the structure, and moreover, the FO[LFP] formulas more clearly describe the runs of the program. This point was made prominently in [9].

Consider the formula $\exists X\ \mathrm{vc}(\{\varphi\}p\{X\})$, which asks for a relation X such that all states satisfying φ fulfil X after running the program p. Similarly, we are interested in solutions of the formula $\exists Y\ \mathrm{vc}(\{Y\}p\{\psi\})$. Note that these are linear-Horn formula equations, and therefore, we can apply the results from Section 4. In general, there also occur predicate variables in $\mathrm{vc}(\{\varphi\}p\{\psi\})$, yet here we are only interested in the predicate variables that are stated specifically.

Theorem 67. *Let $\mathcal{L}$ be a language and $\mathcal{M}$ be a structure. Let p be a program and ψ be a first-order formula. Consider the linear-Horn formula equation $\exists Y\ \pi \equiv \exists Y\ \mathrm{vc}(\{Y\}p\{\psi\})$. Let $\nu := \neg[\mathrm{lfp}_Y\ \Phi_{\pi^D}]$, then*:

$$[\nu] = \mathrm{wp}(p, \psi).$$

Proof. As $\mathcal{M} \models \mathrm{vc}(\{\bot\}p\{\psi\})$ we know that $\mathcal{M} \models \exists Y\ \mathrm{vc}(\{Y\}p\{\psi\})$. Hence with Theorem 46/1, we obtain $\mathcal{M} \models \mathrm{vc}(\{\nu\}p\{\psi\})$, which implies $\mathcal{M} \models \{\nu\}p\{\psi\}$ due to Theorem 60. Now Lemma 63 states $[\nu] \subseteq \mathrm{wp}(p, \psi)$.

For the other inclusion, let $R_{\mathrm{wp}}(p, \psi)$ be the relation defined by the weakest precondition, it holds $[R_{\mathrm{wp}}(p, \psi)] = \mathrm{wp}(p, \psi)$. From Lemma 64, we obtain $\mathcal{M} \models \mathrm{vc}(\{R_{\mathrm{wp}}(p, \psi)\}p\{\psi\})$, and we can apply Theorem 46/2. This yields $\mathcal{M} \models \forall \overline{x}\ (R_{\mathrm{wp}}(p, \psi)(\overline{x}) \rightarrow \nu(\overline{x}))$. In particular, we got $\mathcal{M}, \sigma \models R_{\mathrm{wp}}(p, \psi) \rightarrow \nu$ for all $\sigma \in \Sigma$. Hence, $[\nu] \supseteq [R_{\mathrm{wp}}(p, \psi)] = \mathrm{wp}(p, \psi)$. □

Theorem 68. *Let $\mathcal{L}$ be a language and $\mathcal{M}$ be a structure. Let p be a program and φ be a first-order formula. Consider the linear-Horn formula equation $\exists X\ \pi \equiv \exists X\ \mathrm{vc}(\{\varphi\}p\{X\})$. Let $\mu := [\mathrm{lfp}_X\ \Phi_{\pi}]$, then*:

$$[\mu] = \mathrm{sp}(p, \varphi).$$

Proof. Dual to the proof of Theorem 67. □

Note that the formulas μ and ν thus obtained do not rely on an expressivity hypothesis anymore. The encoding of sequences is replaced by the least fixed-point operator.

6.4 Hoare Logic

It is well known that Hoare logic, as introduced in [31], is a sound and relative complete proof system for Hoare triples. Hoare logic is usually formulated in the language of arithmetic $\mathcal{L}_A$ and for the structure $\mathbb{Z}$. We generalise this definition for an arbitrary structure and show its equivalence to the abstract semantics of the verification condition of a Hoare triple. In the classical setting of $\mathbb{Z}$, we obtain the usual soundness and completeness result, where the use of the expressivity hypothesis is pointed out explicitly.

Since many practical algorithms for verification generate invariants of a particular form, we introduce the Hoare calculus parameterised by a set of first-order formulas C which denotes the formulas allowed as invariants. In Theorem 72, we show that provability in the Hoare calculus restricted to C is equivalent to the truth of the verification condition in the definable model abstraction of C.

Definition 69 (Hoare rules). Let $\mathcal{L}$ be a language, $\mathcal{M}$ be a model and C be a set of first-order formulas in $\mathcal{L}$. Let $\{\varphi\}p\{\psi\}$ be a Hoare triple. We define the following rules, called Hoare rules. We write $\mathcal{M} \vdash_C \{\varphi\}p\{\psi\}$ if $\{\varphi\}p\{\psi\}$ is provable by the Hoare rules in $\mathcal{M}$ and C. If C is the set of all first-order formulas, we often omit the subscript and write $\mathcal{M} \vdash \{\varphi\}p\{\psi\}$. Here, B is a first-order formula in $\mathcal{L}$, p, p_0 and p_1 are programs, t is an $\mathcal{L}$-term, x_j is a variable and, most importantly, I is in C:

$$\frac{}{\{\varphi\}\mathbf{skip}\{\varphi\}}(skip)$$

$$\frac{}{\{\varphi[x_j\backslash t]\}x_j := t\{\varphi\}}(assign)$$

$$\frac{\{\varphi\}p_0\{I\} \quad \{I\}p_1\{\psi\}}{\{\varphi\}p_0; p_1\{\psi\}}(sequence)$$

$$\frac{\{\varphi \land B\}p_0\{\psi\} \quad \{\varphi \land \neg B\}p_1\{\psi\}}{\{\varphi\}\mathbf{if}\ B\ \mathbf{then}\ p_0\ \mathbf{else}\ p_1\{\psi\}}(conditional)$$

$$\frac{\mathcal{M} \models \varphi \to I \quad \{I \land B\}p\{I\} \quad \mathcal{M} \models I \land \neg B \to \psi}{\{\varphi\}\ \mathbf{while}\ B\ \mathbf{do}\ p\{\psi\}}(while)$$

$$\frac{\mathcal{M} \models \varphi \to \varphi' \quad \{\varphi'\}p\{\psi'\} \quad \mathcal{M} \models \psi' \to \psi}{\{\varphi\}p\{\psi\}}(consequence).$$

Remark 70. In most of the literature the (while)-rule is replaced by:

$$\frac{\{\varphi \land B\}p\{\varphi\}}{\{\varphi\}\mathbf{while}\ B\ \mathbf{do}\ p\{\varphi \land \neg B\}}(while').$$

If C is the set of all first-order formulas, this leads to an equivalent proof calculus, as (while') is a special case of (while) for $I = \varphi$ and $\psi = \varphi \land \neg B$. In the other direction (while) can be obtained by combining the (while') and (consequence) rule. Observe that this is not true in general as (while') is not a special case of (while) if $\varphi \notin C$.

In the next theorem, we relate Hoare logic with the validity of the verification condition of a Hoare triple. As the provability relation above, the validity relation will be parameterised with a set of first-order formulas C. This leads to the relation $\mathcal{M} \models_C \psi$, where the second-order quantifiers in ψ are interpreted over the formulas in C. Note that if $(\mathcal{M}, G_C)$ is a definable model abstraction, then $(\mathcal{M}, G_C) \models_a \psi$ iff $\mathcal{M} \models_C \psi$ for all SO-formulas ψ. The difference is that for the relation $\models_C$ we do not demand that the sets defined by formulas in C form a complete lattice, and thus, fixed-point operators cannot be interpreted.

Definition 71. Let C be a set of first-order formulas. Given a model $\mathcal{M}$ and environment θ, we define the relation $\mathcal{M}, \theta \models_C \varphi$ inductively on SO-formulas φ. If φ is first-order, then $\models_C$ is identical to $\models$. For formulas of the form $\exists X\psi$, we define:

$$\mathcal{M}, \theta \models_C \exists\psi \quad \Leftrightarrow \quad \exists\chi \in C : \ \mathcal{M}, \theta[X := \chi^{\mathcal{M}}] \models_C \psi,$$

and analogously for formulas of the form $\forall X\psi$. As usual, we define $\mathcal{M} \models_C \psi$ if $\mathcal{M}, \theta \models_C \psi$ for all environments θ.

Theorem 72. *Let $\mathcal{L}$ be a language and $\mathcal{M}$ be a model. Let $\{\varphi\}p\{\psi\}$ be a Hoare triple and let C be a set of first-order formulas. Then:*

$$\mathcal{M} \models_C \mathrm{vc}(\{\varphi\}p\{\psi\}) \quad \Leftrightarrow \quad \mathcal{M} \vdash_C \{\varphi\}p\{\psi\}.$$

Proof. '$\Rightarrow$': The proof is by induction on the program p. We only show the cases $p \equiv \mathbf{skip}$ and $p \equiv \mathbf{while}\ B\ \mathbf{do}\ p_0$, the others are similar.

- $p \equiv \mathbf{skip}$: The assumption is $\mathcal{M} \models \varphi \to \psi$ and the (skip)-rule yields $\mathcal{M} \vdash \{\varphi\}\mathbf{skip}\{\varphi\}$. Hence, we obtain $\mathcal{M} \vdash \{\varphi\}\mathbf{skip}\{\psi\}$ with the (consequence)-rule. Note that in the (consequence)-rule φ' and ψ' may be any first-order formulas and are not restricted to C, which is of importance here.
- $p \equiv \mathbf{while}\ B\ \mathbf{do}\ p_0$: The assumption is $\mathcal{M} \models_C \exists I\ \mathrm{vc}(\{I \land B\}p_0\{I\}) \land (\varphi \to I) \land (I \land \neg B \to \psi)$, which means that there exists $\chi \in C$ such that $\mathcal{M} \models_C \mathrm{vc}(\{\chi \land B\}p_0\{\chi\})$ and $\mathcal{M} \models (\varphi \to \chi) \land (\chi \land \neg B \to \psi)$. By the induction hypothesis, we have $\mathcal{M} \vdash_C \{\chi \land B\}p_0\{\chi\}$, now we can use the (while)-rule to obtain $\mathcal{M} \vdash_C \{\varphi\}p\{\psi\}$.

'$\Leftarrow$': This direction is shown by proving inductively, that every Hoare rule yields Hoare triples such that $\mathcal{M} \models_C \mathrm{vc}(\{\varphi\}p\{\psi\})$. In other words, we show that the Hoare rules are sound with respect to the semantics $\mathcal{M} \models_C \mathrm{vc}(\{\varphi\}p\{\psi\})$. We concentrate on the exemplary (sequence) and (consequence)-rule.

- (sequence): By assumption, there exists $I \in C$ such that $\mathcal{M} \vdash_C \{\varphi\}p_0\{I\}$ and $\mathcal{M} \vdash_C \{I\}p_1\{\psi\}$. Using the induction hypothesis, we obtain that there exists $I \in C$ such that $\mathcal{M} \models_C \mathrm{vc}(\{\varphi\}p_0\{I\}) \land \mathrm{vc}(\{I\}p_1\{\psi\})$, which is the definition of $\mathcal{M} \models_C \mathrm{vc}(\{\varphi\}p_0; p_1\{\psi\})$.
- (consequence): By assumption $\mathcal{M} \models \varphi \to \varphi'$, $\mathcal{M} \models \psi' \to \psi$ and $\mathcal{M} \vdash_C \mathrm{vc}(\{\varphi'\}p\{\psi'\})$. Using the induction hypothesis, we have $\mathcal{M} \models_C \mathrm{vc}(\{\varphi'\}p\{\psi'\})$. Now Lemma 2 yields $\mathcal{M} \models_C \mathrm{vc}(\{\varphi\}p\{\psi\})$. Note that we stated Lemma 2 only for classical semantics, yet it straightforwardly generalises to the relation $\models_C$. □

Lemma 73. *Let $\mathcal{L}$ be a language and $\mathcal{M}$ be a model. Let $\{\varphi\}p\{\psi\}$ be a Hoare triple and let C be a set of first-order formulas in $\mathcal{L}$. Then:*

$$\mathcal{M} \models_C \mathrm{vc}(\{\varphi\}p\{\psi\}) \quad \Rightarrow \quad \mathcal{M} \models \mathrm{vc}(\{\varphi\}p\{\psi\}).$$

Proof. This follows as $\mathrm{vc}(\{\varphi\}p\{\psi\})$ is an existential second-order formula. □

Notably in the traditional case of the integers and the set of all first-order formulas C the converse of Lemma 73 holds as well. Here, the expressivity of $\mathbb{Z}$ is crucial.

Lemma 74. *Let $\mathcal{L}_A = \{0, 1, +, -, \cdot, \leq\}$ be the language of arithmetic and C be the set of all first-order formulas. Then:*

$$\mathbb{Z} \models \mathrm{vc}(\{\varphi\}p\{\psi\}) \quad \Rightarrow \quad \mathbb{Z} \models_C \mathrm{vc}(\{\varphi\}p\{\psi\}).$$

Proof. The fixed-point theorem Corollary 42 states that $\mathbb{Z} \models \mathrm{vc}(\{\varphi\}p\{\psi\})$ is equivalent to $\mathbb{Z} \models \tilde{\mathrm{vc}}(\{\varphi\}p\{\psi\})[\overline{I}\backslash\overline{\mu}]$, where $\overline{\mu}$ is a tuple of LFP-atoms. Using Gödel's β-function, we can encode those LFP-atoms and find first-order formulas $\lambda_1, \dots, \lambda_n$ such that $\mathbb{Z} \models \mu_j \leftrightarrow \lambda_j$ for $j = 1, \dots, n$. We explain how this can be done for an LFP-atom $\mu \equiv [\mathrm{lfp}_R\ \varphi]$, where φ is an existential first-order formula containing R. For $l \in \omega$, let σ^l_φ be defined as in Definition 19. Theorem 21 states that:

$$[\mathrm{lfp}_R\ \varphi] \equiv \bigvee_{l \in \omega} \sigma^l_\varphi.$$

Now define $\lambda(\overline{x})$ using Gödel's β-function as follows: 'There exists a sequence of formulas $\sigma^0_\varphi, \ldots, \sigma^n_\varphi$ defined as above such that $\sigma^n_\varphi(\overline{x})$ holds'. Then, $\mathbb{Z} \models \mu \leftrightarrow \lambda$. As $\overline{\mu}$ is defined from existential first-order formulas, this applies here. Hence, $\mathbb{Z} \models \tilde{\mathrm{vc}}(\{\varphi\}p\{\psi\})[\overline{I}\backslash\overline{\lambda}]$, which implies $\mathbb{Z} \models_C \mathrm{vc}(\{\varphi\}p\{\psi\})$. □

Remark 75. Theorem 72 together with Lemmas 73 and 74 yields:

$$\mathbb{Z} \models \mathrm{vc}(\{\varphi\}p\{\psi\}) \quad \Leftrightarrow \quad \mathbb{Z} \vdash \{\varphi\}p\{\psi\}.$$

This has also been shown in [41], yet here we pointed out explicitly where the expressivity of $\mathbb{Z}$ is used and which parts of the proof can be generalised to different structures. Combining this equality with Theorems 60 and 65, we obtain the classical statement:

$$\mathbb{Z} \models \{\varphi\}p\{\psi\} \quad \Leftrightarrow \quad \mathbb{Z} \vdash \{\varphi\}p\{\psi\}.$$

7 Fixed-Point Approximation

The problem of finding first-order formulas which approximate a second-order formula is an intensively studied topic in the history of logic. For second-order formulas of the form $\exists\overline{X}\forall\overline{y}\ \psi$, where ψ is quantifier-free, it has already been investigated by Ackermann in 1935 [1]. Under the assumptions that the language $\mathcal{L}$ is relational, i.e., $\mathcal{L}$ contains no function symbols, and there is only one unary predicate variable X, Ackermann [1] shows in detail how to compute an infinite conjunction of first-order formulas that is equivalent to the second-order formula $\exists X\forall\overline{y}\ \psi$. This is achieved with a method similar to modern resolution. This result has also been extended to any number of predicate variables of arbitrary arity [1, 63], yet the assumption of a relational language remains. In this section, we show how to obtain a similar result for any language $\mathcal{L}$, but with another assumption: we only consider Horn formula equations. Moreover, this is attained with a completely different method as a straightforward corollary of our Horn fixed-point theorem.

Let $\exists\overline{X}\psi$ be a Horn formula equation. In the last section, we found least fixed-point logic formulas $\mu_1, \ldots, \mu_n$ such that $\models \exists\overline{X}\ \psi \leftrightarrow \psi[X_1\backslash\mu_1, \ldots, X_n\backslash\mu_n]$. We can now use the first-order formulas defined in Section 2.2.3, which approximate $\mu_1, \ldots, \mu_n$ and therefore lead to an approximation of the second-order formula $\exists\overline{X}\psi$.

Theorem 76. *Let $\exists\overline{X}\psi$ be a Horn formula equation. Then, there exists a, possibly infinite, set of first-order formulas Ψ such that*:

$$\exists\overline{X}\psi \equiv \bigwedge_{\varphi \in \Psi} \varphi.$$

Proof. Let $\mu_j := [\mathrm{lfp}_{X_j}\ \Phi_\psi]$ for $j \in \{1, \ldots, n\}$. Applying Corollary 42, we have:

$$\exists\overline{X}\ \psi \equiv \psi[X_1\backslash\mu_1, \ldots, X_n\backslash\mu_n].$$

By construction $\overline{\mu}$ satisfies all base and induction clauses, thus:

$$\exists\overline{X}\psi \equiv \forall\overline{y} \bigwedge_{C \in E} C[X_1\backslash\mu_1, \ldots, X_n\backslash\mu_n](\overline{y}),$$

where E is the set of all end clauses in ψ. Note that the formulas $\varphi_1, \ldots, \varphi_n$ in the n-tuple Φ are existential first-order formulas, hence we can use Theorem 21 to obtain:

$$\exists \overline{X} \psi \equiv \forall \overline{y} \bigwedge_{C \in E} C[X_1 \backslash \sigma^{\omega}_{1,\Phi}, \ldots, X_n \backslash \sigma^{\omega}_{n,\Phi}](\overline{y}),$$

where $\sigma^{\omega}_{j,\Phi} \equiv \bigvee_{l \in \omega} \sigma^{l}_{j,\Phi}$ is an infinite disjunction of first-order formulas for $j \in \{1, \ldots, n\}$. The clauses in E have the form $\neg\gamma \vee \neg X_{i_1}(\overline{t_1}) \vee \cdots \vee \neg X_{i_m}(\overline{t_m})$. We denote a clause in E by the determining tuple (γ, ι, τ), where $\iota := \{i_1, \ldots, i_m\}$ and $\tau := \{\overline{t_1}, \ldots, \overline{t_m}\}$. Then:

$$\exists \overline{X} \psi \equiv \forall \overline{y} \bigwedge_{(\gamma,\iota,\tau) \in E} \left(\neg\gamma \vee \neg\sigma^{\omega}_{i_1,\Phi}(\overline{t_1}) \vee \cdots \vee \neg\sigma^{\omega}_{i_m,\Phi}(\overline{t_m}) \right).$$

Per definition, there is a set of first-order formulas Ψ_j such that $\sigma^{\omega}_{j,\Phi} \equiv \bigvee_{\varphi \in \Psi_j} \varphi$ for every $j \in \{1, \ldots, n\}$. Thus, with repeated application of Lemma 16, there exists a set of first-order formulas Ψ such that:

$$\forall \overline{y} \bigwedge_{(\gamma,\iota,\tau) \in E} \left(\neg\gamma \vee \neg \left(\bigvee_{\varphi \in \Psi_{i_1}} \varphi(\overline{t_1}) \right) \vee \cdots \vee \neg \left(\bigvee_{\varphi \in \Psi_{i_m}} \varphi(\overline{t_m}) \right) \right) \equiv \bigwedge_{\varphi \in \Psi} \varphi.$$

□

Note that the set Ψ in Theorem 76 is countable and given constructively, let us say $\Psi = \{\psi_1, \psi_2, \ldots\}$. Then, the first-order formulas $\psi_1, \psi_1 \wedge \psi_2, \ldots$ approximate the second-order formula $\exists \overline{X} \psi$.

8 Inductive Theorem Proving

In this section, we consider an approach to inductive theorem proving based on tree grammars [16] and show how to obtain a central result of [16] from the Horn fixed-point theorem shown in Section 4. The aim of this approach is to generate a proof of a universal statement in two phases. In the first phase, proofs of small instances are computed, from which a second-order unification problem is deduced. Each solution of the unification problem is an induction invariant. We will not go into depth about phase one as we are more interested in phase two, in which solutions of the second-order unification problem are computed. We will see that the second-order unification problem is in fact a Horn formula equation, where we use the Horn fixed-point theorem to get solutions. In [16], as a running example, the approach is demonstrated on a proof that the head-recursive and the tail-recursive definitions of the factorial function compute the same function. For space-reasons, we do not repeat this example here. The interested reader is referred to [16], and in particular to Section 7.1 which contains the computation of a solution of the Horn formula equation induced by this running example. This approach has been developed for the natural numbers in [16]. However, this is not a fundamental restriction; the approach could easily be extended to induction on recursive data types, such as lists, trees, and so on.

In this chapter, we work in a language $\mathcal{L}$, which contains the constant symbol 0 and the unary function symbol s. We define $\overline{n} = s^n(0)$.

Definition 77. Let $F_1, \ldots, F_n, G_1, \ldots, G_m$ be formulas. A formula of the form $F_1 \wedge \cdots \wedge F_n \rightarrow G_1 \vee \cdots \vee G_m$ is called a *sequent* and is written as $\Gamma \Rightarrow \Delta$, where $\Gamma = \{F_1, \ldots, F_n\}$ and $\Delta = \{G_1, \ldots, G_m\}$.

Definition 78 *([16], Definition 6.1).* Let $\alpha, \beta, \nu, \gamma$ be variables only occurring where indicated. Let $\Gamma_0(\alpha, \beta), \Gamma_1(\alpha, \nu, \gamma), \Gamma_2(\alpha)$ be multisets of quantifier-free first-order formulas and let $B(\alpha)$ be a quantifier-free formula. Let $t_i(\alpha, \nu, \gamma)$ and $u_j(\alpha)$ be terms for $i \in \{1, \ldots, n\}$ and $j \in \{1, \ldots, m\}$, where $n, m \geq 1$. Let X be a ternary predicate variable. Then, the list of the following three sequents is a *schematic simple induction proof (schematic s.i.p.)*:

(1) $\Gamma_0(\alpha,\beta) \Rightarrow X(\alpha,0,\beta)$.
(2) $\Gamma_1(\alpha,\nu,\gamma), \bigwedge_{1\le i\le n} X(\alpha,\nu,t_i(\alpha,\nu,\gamma)) \Rightarrow X(\alpha,s(\nu),\gamma)$.
(3) $\Gamma_2(\alpha), \bigwedge_{1\le j\le m} X(\alpha,\alpha,u_j(\alpha)) \Rightarrow B(\alpha)$.

Note that every schematic s.i.p. defines a Horn formula equation $\exists X\forall\alpha,\beta,\nu,\gamma\,\psi$, where ψ is the conjunction of the three sequents. A solution of a schematic s.i.p. S is defined to be a quantifier-free formula $F(x,y,z)$, such that the three sequents of S with X replaced by F are quasi-tautological, which means that it is valid in first-order logic with equality. This means exactly $\models \forall\alpha,\beta,\nu,\gamma\,\psi[X\backslash F]$ as we always talk about logic with equality. For solving a schematic s.i.p., $\Gamma_0,\Gamma_1,\Gamma_2$ may be arbitrary multisets of first-order formulas. In [16], the aim is to prove a universal statement $\forall\alpha B(\alpha)$. A solution of an appropriate schematic s.i.p. yields a proof of $\forall\alpha B(\alpha)$ as follows: Of particular interest is the case, when $\Gamma_0,\Gamma_1,\Gamma_2$ are instances of a theory Γ. In applications, this is an arithmetic theory, e.g., Robinson arithmetic. Assume we have a solution F of a schematic s.i.p. S in that specific case. Then, we can obtain a proof of $\forall\Gamma \Rightarrow \forall\beta\,F(\alpha,0,\beta)$ from the first sequent of S, where $\forall\Gamma$ is the universal closure of Γ. Similarly, we obtain $\forall\Gamma,\forall\beta F(\alpha,\nu,\beta) \Rightarrow \forall\beta F(\alpha,s(\nu),\beta)$ from the second sequent. As we are interested in proofs with an induction rule, we then are able to deduce $\forall\Gamma \Rightarrow \forall\nu,\beta F(\alpha,\nu,\beta)$. Thus, using the third sequent of S yields a proof of $\forall\Gamma \Rightarrow B(\alpha)$ and therefore of $\forall\Gamma \Rightarrow \forall\alpha B(\alpha)$.

Next, we examine how to get a solution of a schematic s.i.p.

Definition 79 ([16] , *Definition 6.10).* Let S be a schematic s.i.p. with premises given as in Definition 78. By recursion define the following sequence of formulas:

$$C_{S,0}(x,z) := \bigwedge \Gamma_0(x,z)$$
$$C_{S,q+1}(x,z) := \bigwedge \Gamma_1(x,\overline{q},z) \wedge \bigwedge_{1\le i\le n} C_{S,q}(x,t_i(x,\overline{q},z)).$$

$C_{S,q}(x,z)$ is called the qth canonical solution of S.

Let $(\sigma^l_{\Phi_\psi})_{l\in\omega}$ be the sequence of first-order formulas, describing the unfolding of the fixed-point operator, defined in Definition 19 from the 1-tuple Φ_ψ of first-order formulas (φ) defined in Definition 29 from the Horn formula equation $\exists X\psi$.

Lemma 80. *Let S be a schematic s.i.p. and let $\exists X\psi$ be the formula equation defined from S. Then, for all $q\in\mathbb{N}$ it holds*:

$$C_{S,q}(x,z) \equiv \sigma^{q+1}_{\Phi_\psi}(x,\overline{q},z).$$

Proof. We show the equivalence by induction on q. The definition of $(\sigma^l_\Phi)_{l\in\omega}$ is $\sigma^0_\Phi \equiv \bot$ and for $l\in\omega$:

$$\sigma^{l+1}_\Phi(x,y,z) \equiv\ \varphi(\sigma^l_\Phi,x,y,z) \equiv \exists\alpha,\beta,\nu,\gamma\Big(\Big(\bigwedge \Gamma_0(\alpha,\beta) \wedge x=\alpha \wedge y=0 \wedge z=\beta\Big)$$
$$\vee\Big(\bigwedge \Gamma_1(\alpha,\nu,\gamma) \wedge \bigwedge_{1\le i\le n} \sigma^l_\Phi(\alpha,\nu,t_i(\alpha,\nu,\gamma)) \wedge x=\alpha \wedge y=s(\nu) \wedge z=\gamma\Big)\Big).$$

In particular, it holds that $\sigma^1_\Phi(x,0,z) \equiv \bigwedge \Gamma_0(x,z) \equiv C_{S,0}(x,z)$. For $q \geq 0$, we have:

$$\begin{aligned}\sigma^{q+2}_\Phi(x,\overline{q+1},z) &\equiv \bigwedge \Gamma_1(x,\overline{q},z) \wedge \bigwedge_{1\leq i\leq n} \sigma^{l+1}_\Phi(x,\overline{q},t_i(x,\overline{q},z)) \\ &\equiv \bigwedge \Gamma_1(x,\overline{q},z) \wedge \bigwedge_{1\leq i\leq n} C_{S,q}(x,t_i(x,\overline{q},z)) \equiv C_{S,q+1}(x,z).\end{aligned}$$

□

Now, we want to present Lemma 6.12 from [16] as a direct corollary of Corollary 42.

Lemma 81 ([16], Lemma 6.12). *Let S be a schematic s.i.p. Then, for any solution F of S and any $q \in \mathbb{N}$, the qth canonical solution $C_{S,q}(x,z)$ logically implies $F(x,\overline{q},z)$.*

Proof. From the definition of σ^ω_Φ, it follows that $\models \sigma^q_\Phi \rightarrow \sigma^\omega_\Phi$ for all $q \in \omega$. Theorem 21 states that $\sigma^\omega_\Phi \equiv [\mathrm{lfp}_X\ \Phi_\psi]$, and thus, Lemma 80 yields $\models C_{S,q}(x,z) \rightarrow [\mathrm{lfp}_X\ \Phi_\psi]$ for all $q \in \omega$. Now, Corollary 42 concludes $\models C_{S,q}(x,z) \rightarrow F(x,\overline{q},z)$ for all $q \in \omega$. □

Thus, we see that using the results from Section 4 shortens the proofs of a result from [16] and explains it as a corollary of our fixed-point theorem.

9 Related Work

Much of the motivation for studying the solutions of formula equations rests on the many connections this problem has to a wide variety of other topics in computational logic. In addition to the connections described in the previous sections, in this section, we discuss work on various related topics.

Boolean Equations and Unification. Under the name of Boolean equations, solving formula equations is one of the oldest and one of the most central problems of logic. It goes back to the 19th century and was already thoroughly investigated in [56]. See [54] for a comprehensive textbook. Solving Boolean equations is closely related to Boolean unification, a subject of thorough study in computer science, see, e.g., [3, 11, 46] and [47] for a survey. The generalisation of this problem from propositional to first-order logic has been made explicit as early as [6, 7], see [64] for a recent survey.

Second-Order Quantifier Elimination. Solving a formula equation in first-order logic is closely related to second-order quantifier elimination, a problem with applications in a variety of areas in computer science, e.g., modal logic, databases and common-sense reasoning [22]. A seminal work on second-order quantifier elimination and basis of many modern algorithms is Ackermann's [1]. The two main modern algorithms for SOQE are the SCAN algorithm and the DLS algorithm. SCAN has been introduced in [21] and is closely related to hierarchic superposition [4]. Implementations of SCAN can be found in [19, 51]. It has been used for a range of applications in various areas of computational logic [22, 23]. The DLS algorithm has been introduced in [14] and extended to the DLS* algorithm that works with fixed-points in [15, 50]. From the perspective of second-order quantifier elimination, our fixed-point theorem can be seen as establishing that Horn formula equations have canonical witnesses, called ELIM-witnesses in [64], in FO[LFP].

Fixed-Point Theorems. Fixed-point theorems play a central role for solving equations in many areas of mathematics. They abound in logic with one of the most famous examples being the recursion theorem in computability theory which guarantees the existence of a solution of a system of recursion equations by computing a fixed-point. But also in areas quite remote from logic such constructions can be found, as, for example, in the use of Banach's fixed-point theorem in the proof of the Picard–Lindelöf theorem on the unique solvability of ordinary differential equations. Our use of the fixed-point theorem for solving Horn formula equations in FO[LFP] follows exactly the same scheme.

Least Fixed-Point Logics. First-order logic with a least fixed-point operator, FO[LFP], has been studied at least since [49]. It is well known in finite model theory and computational complexity, most notably because of the Immerman–Vardi theorem [36, 62]. FO[LFP] has already been used for second-order quantifier elimination in the literature, for example, by Nonnengart and Szałas in [50]. In fact, the key lemma for our fixed-point theorem is a generalisation of a result of [50], which, in turn, is a generalisation of a result of Ackermann's [1].

Logic Programming. In logic programming, it is well known that a set of first-order Horn clauses has a unique minimal model, i.e., a satisfying set of ground atoms, and that this set can be obtained as fixed-point of an operator defined by the clause set [37]. From that point of view, our fixed-point theorem can be understood as, essentially, expressing this computation within the logic itself.

Verification. An entire workshop series is devoted to applications of constrained Horn clauses in verification and synthesis, see, e.g., [33]. A set of constrained Horn clauses is simply a Horn formula equation in the terminology of this paper. Also, in this community, the relationship between Horn formula equations and least fixed points is well known and has been exploited for practical purposes, e.g., in [61], see also [59]. In contrast to this line of work, our contribution is the formulation of a general, theoretical, result based on an explicit fixed point operator that encompasses both, simultaneous least fixed points and abstract interpretation. Abstract interpretation has been used in tools for solving Horn clauses [32, 39].

The work [9] advocates for the use of existential fixed-point logic as a logical foundation of verification. In particular, it was already shown in [9] that the weakest precondition and the strongest postcondition can be defined without expressivity hypothesis in existential fixed-point logic. We consider this paper as a continuation of this approach which adds to it by using the more abstract concept of formula equation and showing how this is useful for a wide range of different application in- and outside of verification.

Extensions. Various extensions of Horn formula equations have been considered, mostly motivated by their applications in verification. In particular, Horn formula equations have been extended to coinduction in [5]. In [42, 60], Horn formula equations have been investigated in the context of first-order logic with least and greatest fixed point operators, thus generalizing coinduction. Extensions to higher-order logic have been investigated in [10, 38, 43, 44]. In addition to higher-order logic, Tsukada [58] also discusses various other extensions and applications to program verification from the point of view of computational complexity. Our work, being restricted to first-order Horn formula equations and least fixed points, does not directly apply to these results, and extending our work to these settings is left as future work by this paper.

10 Conclusion

We have shown a fixed-point theorem for Horn, dual-Horn and linear-Horn formula equations. These fixed-point theorems apply to an abstract semantics which generalises standard semantics by allowing to interpret the least fixed-point operator and the second-order quantifier in a complete lattice. The central lemma for proving these results is a generalisation of a result from [50], which, in turn, is a generalisation of a result from [1]. From the point of view of logic programming, our proof can be understood as expressing the construction of a minimal model of a set of Horn clauses on the object level as a formula in first-order logic with least fixed points.

These fixed-point theorems contribute to our theoretical understanding of the logical foundations of constrained Horn clause solving and software verification. As corollary to our fixed point theorem, we have obtained the expressibility of the weakest precondition and the strongest postcondition and thus the partial correctness of an imperative program in FO[LFP]. We believe that it is fruitful to consider constrained Horn clause solving from the more general point of view of solving formula equations. On the theoretical level, this perspective uncovers connections to a number of topics such as second-order quantifier elimination and results such as Ackermann's [1]. On the practical level, it suggests to study the applicability of algorithms such as DLS and SCAN for constrained Horn clauses and vice versa, that of algorithms for constrained Horn clause solving for applications of second-order quantifier elimination.

Moreover, we have shown that this fixed-point theorem has a number of applications throughout computational logic: as described in Section 5, it allows to considerably simplify the proof of the decidability of affine formula equations given in [30]. As shown in Section 7, it allows a generalisation of a result by Ackermann [1] on second-order quantifier elimination in a direction different from the recent generalisation [63] of that result. Furthermore, as shown in Section 8, it allows to obtain a result on the generation of a proof with induction based on partial information about that proof shown in [16] as straightforward corollary. Due to the naturalness of both, the class of formula equations and the result, we expect many further applications of this fixed-point theorem.

In conclusion, we believe that Horn formula equations have a central role to play in computational logic and that the fixed-point theorem is one of their most important theoretical properties. The proof of the fixed-point theorem integrates neatly with the existing literature on second-order quantifier elimination. The expressivity of Horn formula equations enables their applicability in a wide range of topics from software verification to inductive theorem proving.

Acknowledgements

We would like to thank the anonymous reviewers for their thorough reading of our submission and for their many comments which have led to considerable improvements of this paper. We would also like to thank everyone who has provided feedback on the workshop papers [28] and [29] which were instrumental in developing the full results presented in this paper.

References

[1] Wilhelm Ackermann. 1935. Untersuchungen über das eliminationsproblem der mathematischen logik. *Mathematische Annalen* 110, 1 (1935), 390–413. DOI: https://doi.org/10.1007/BF01448035

[2] A. Arnold and D. Niwinski. 2001. *Rudiments of Calculus*. Elsevier Science.

[3] Franz Baader. 1998. On the complexity of boolean unification. *Information Processing Letters* 67 (1998), 215–220.

[4] Leo Bachmair, Harald Ganzinger, and Uwe Waldmann. 1994. Refutational theorem proving for hierarchic first-order theories. *Applicable Algebra in Engineering, Communication and Computing* 5 (1994), 193–212. DOI: https://doi.org/10.1007/BF01190829

[5] Henning Basold, Ekaterina Komendantskaya, and Yue Li. 2019. Coinduction in uniform: Foundations for corecursive proof search with horn clauses. In *Proceedings of the 28th European Symposium on Programming—Programming Languages and Systems (ESOP '19), Held as Part of the European Joint Conferences on Theory and Practice of Software (ETAPS '19).* Luís Caires (Ed.), Lecture Notes in Computer Science, Vol. 11423, Springer, 783–813. DOI: https://doi.org/10.1007/978-3-030-17184-1_28

[6] Heinrich Behmann. 1950. Das auflösungsproblem in der klassenlogik. *Archiv Für Mathematische Logik Und Grundlagenforschung* 1, 1 (1950), 17–29.

[7] Heinrich Behmann. 1951. Das auflösungsproblem in der klassenlogik. *Archiv Für Mathematische Logik Und Grundlagenforschung* 1, 2 (1951), 33–51.

[8] Nikolaj Bjørner, Arie Gurfinkel, Kenneth L. McMillan, and Andrey Rybalchenko. 2015. Horn clause solvers for program verification. In *Fields of Logic and Computation II—Essays Dedicated to Yuri Gurevich on the Occasion of His*

75th Birthday. Lev D. Beklemishev, Andreas Blass, Nachum Dershowitz, Bernd Finkbeiner, and Wolfram Schulte (Eds.), Lecture Notes in Computer Science, Vol. 9300, Springer, 24–51. DOI: https://doi.org/10.1007/978-3-319-23534-9_2

[9] Andreas Blass and Yuri Gurevich. 1987. Existential fixed-point logic. In *Computation Theory and Logic, In Memory of Dieter Rödding*. Egon Börger (Ed.), Lecture Notes in Computer Science, Vol. 270, Springer, 20–36. DOI: https://doi.org/10.1007/3-540-18170-9_151

[10] Toby Cathcart Burn, C.-H. Luke Ong, and Steven J. Ramsay. 2018. Higher-order constrained horn clauses for verification. *Proceedings of the ACM on Programming Languages* 2, POPL (2018), Article 11, 1–28. DOI: https://doi.org/10.1145/3158099

[11] Wolfram Büttner and Helmut Simonis. 1987. Embedding Boolean expressions into logic programming. *Journal of Symbolic Computation* 4, 2 (1987), 191–205.

[12] Patrick Cousot and Radhia Cousot. 1977. Abstract interpretation: A unified lattice model for static analysis of programs by construction or approximation of fixpoints. In *4th ACM Symposium on Principles of Programming Languages*. ACM, 238–252. DOI: https://doi.org/10.1145/512950.512973

[13] B. A. Davey and H. A. Priestley. 2002. *Introduction to Lattices and Order* (2nd ed.). Cambridge University Press, Cambridge. DOI: https://doi.org/10.1017/CBO9780511809088

[14] Patrick Doherty, Witold Lukaszewicz, and Andrzej Szalas. 1997. Computing circumscription revisited: A reduction algorithm. *Journal of Automated Reasoning* 18, 3 (1997), 297–336.

[15] Patrick Doherty, Witold Łukaszewicz, and Andrzej Szałas. 1998. General domain circumscription and its effective reductions. *Fundamenta Informaticae* 36, 1 (1998), 23–55. DOI: https://doi.org/10.3233/FI-1998-3612

[16] Sebastian Eberhard and Stefan Hetzl. 2015. Inductive theorem proving based on tree grammars. *Annals of Pure and Applied Logic* 166, 6 (2015), 665–700. DOI: https://doi.org/10.1016/j.apal.2015.01.002

[17] Sebastian Eberhard, Stefan Hetzl, and Daniel Weller. 2017. Boolean unification with predicates. *Journal of Logic and Computation* 27, 1 (2017), 109–128. DOI: https://doi.org/10.1093/logcom/exv059

[18] Herbert B. Enderton. 2001. *A Mathematical Introduction to Logic* (2nd ed.). Academic Press.

[19] T. Engel. 1996. *Quantifier Elimination in Second-Order Predicate Logic*. Diplomarbeit. Fachbereich Informatik, Univ. des Saarlandes, Saarbrücken, Germany.

[20] Carsten Fritz. 2001. Some fixed point basics. In *Automata, Logics, and Infinite Games: A Guide to Current Research*. Erich Grädel, Wolfgang Thomas, and Thomas Wilke (Eds.), Lecture Notes in Computer Science, Vol. 2500, Springer, 359–364. DOI: https://doi.org/10.1007/3-540-36387-4_20

[21] Dov Gabbay and Hans Jürgen Ohlbach. 1992. Quantifier elimination in second order predicate logic. *South African Computer Journal* 7 (1992), 35–43.

[22] Dov M. Gabbay, Renate A. Schmidt, and Andrzej Szałas. 2008. *Second-Order Quantifier Elimination*. College Publications.

[23] Valentin Goranko, Ullrich Hustadt, Renate A. Schmidt, and Dimiter Vakarelov. 2003. SCAN is complete for all Sahlqvist formulae. In *7th International Seminar on Relational Methods in Computer Science and 2nd International Workshop on Applications of Kleene Algebra*. Rudolf Berghammer, Bernhard Möller, and Georg Struth (Eds.), Lecture Notes in Computer Science, Vol. 3051, Springer, 149–162. DOI: https://doi.org/10.1007/978-3-540-24771-5_13

[24] Erich Grädel. 1991. The expressive power of second order horn logic. In *8th Annual Symposium on Theoretical Aspects of Computer Science (STACS)*. Christian Choffrut and Matthias Jantzen (Eds.), Lecture Notes in Computer Science, Vol. 480, Springer, 466–477. DOI: https://doi.org/10.1007/BFb0020821

[25] Ashutosh Gupta, Corneliu Popeea, and Andrey Rybalchenko. 2014. Generalised interpolation by solving recursion-free horn clauses. In *Proceedings of the 1st Workshop on Horn Clauses for Verification and Synthesis (HCVS)*, 31–38. DOI: https://doi.org/10.4204/EPTCS.169.5

[26] Arie Gurfinkel and Nikolaj Bjørner. 2019. The science, art, and magic of constrained horn clauses. In *21st International Symposium on Symbolic and Numeric Algorithms for Scientific Computing SYNASC*. IEEE, 6–10. DOI: https://doi.org/10.1109/SYNASC49474.2019.00010

[27] Leon Henkin. 1950. Completeness in the theory of types. *Journal of Symbolic Logic* 15, 2 (1950), 81–91. DOI: https://doi.org/10.2307/2266967

[28] Stefan Hetzl and Johannes Kloibhofer. 2021. A fixed point theorem for horn formula equations. In *Proceedings of the 8th Workshop on Horn Clauses for Verification and Synthesis (HCVS)*. Retrieved from https://www.sci.unich.it/hcvs21/papers/HCVS_2021_paper_1.pdf

[29] Stefan Hetzl and Johannes Kloibhofer. 2021. An abstract fixed-point theorem for Horn formula equations (abstract). In *Proceedings of the 2nd Workshop on Second-Order Quantifier Elimination and Related Topics (SOQE)*. CEUR-WS.org, 59–60.

[30] Stefan Hetzl and Sebastian Zivota. 2020. Decidability of affine solution problems. *Journal of Logic and Computation* 30, 3 (2020), 697–714. DOI: https://doi.org/10.1093/logcom/exz033

[31] C. A. R. Hoare. 1969. An axiomatic basis for computer programming. *Communications of the ACM* 12, 10 (Oct. 1969), 576–580. DOI: https://doi.org/10.1145/363235.363259

[32] Krystof Hoder, Nikolaj Bjørner, and Leonardo Mendonça de Moura. 2011. μZ—An efficient engine for fixed points with constraints. In *23rd International Conference on Computer-Aided Verification (CAV)*. Ganesh Gopalakrishnan and Shaz Qadeer (Eds.), Lecture Notes in Computer Science, Vol. 6806, Springer, 457–462. DOI: https://doi.org/10.1007/978-3-642-22110-1_36

[33] Hossein Hojjat and Bishoksan Kafle. 2021. Proceedings of the 8th Workshop on Horn Clauses for Verification and Synthesis (HCVS@ETAPS '21). arXiv: 2109.03988. Retrieved from https://doi.org/10.48550/arXiv.2109.03988

[34] Ehud Hrushovski, Joël Ouaknine, Amaury Pouly, and James Worrell. 2018. Polynomial invariants for affine programs. In *33rd Annual Symposium on Logic in Computer Science (LICS)*. ACM, 530–539.

[35] Ehud Hrushovski, Joël Ouaknine, Amaury Pouly, and James Worrell. 2023. On strongest algebraic program invariants. *Journal of the ACM* 70, 5 (2023), Article 29, 1–22. DOI: https://doi.org/10.1145/3614319

[36] Neil Immerman. 1986. Relational queries computable in polynomial time. *Information and Control* 68, 1–3 (1986), 86–104. DOI: https://doi.org/10.1016/S0019-9958(86)80029-8

[37] Joxan Jaffar and Michael J. Maher. 1994. Constraint logic programming: A survey. *The Journal of Logic Programming* 19/20 (1994), 503–581. DOI: https://doi.org/10.1016/0743-1066(94)90033-7

[38] Jerome Jochems, Eddie Jones, and Steven J. Ramsay. 2023. Higher-order MSL horn constraints. *Proceedings of the ACM on Programming Languages* 7, POPL (2023), 2017–2047. DOI: https://doi.org/10.1145/3571262

[39] Bishoksan Kafle, John P. Gallagher, and José F. Morales. 2016. Rahft: A tool for verifying horn clauses using abstract interpretation and finite tree automata. In *28th International Conference on Computer Aided Verification*. Swarat Chaudhuri and Azadeh Farzan (Eds.), Lecture Notes in Computer Science, Vol. 9779, Springer, 261–268. DOI: https://doi.org/10.1007/978-3-319-41528-4_14

[40] Michael Karr. 1976. Affine relationships among variables of a program. *Acta Informatica* 6 (1976), 133–151. DOI: https://doi.org/10.1007/BF00268497

[41] Johannes Kloibhofer. 2020. A Fixed-Point Theorem for Horn Formula Equations. Master's Thesis. TU Wien, Austria.

[42] Naoki Kobayashi, Takeshi Nishikawa, Atsushi Igarashi, and Hiroshi Unno. 2019. Temporal verification of programs via first-order fixpoint logic. In *26th International Symposium on Static Analysis*. Bor-Yuh Evan Chang (Ed.), Lecture Notes in Computer Science, Vol. 11822, Springer, 413–436. DOI: https://doi.org/10.1007/978-3-030-32304-2_20

[43] Naoki Kobayashi, Kento Tanahashi, Ryosuke Sato, and Takeshi Tsukada. 2023. HFL(Z) validity checking for automated program verification. *Proceedings of the ACM on Programming Languages* 7, POPL (2023), 154–184. DOI: https://doi.org/10.1145/3571199

[44] Naoki Kobayashi, Takeshi Tsukada, and Keiichi Watanabe. 2018. Higher-order program verification via HFL model checking. In *Proceedings of the 27th European Symposium on Programming—Programming Languages and Systems (ESOP '18), Held as Part of the European Joint Conferences on Theory and Practice of Software (ETAPS '18)*. Amal Ahmed (Ed.), Lecture Notes in Computer Science, Vol. 10801, Springer, 711–738. DOI: https://doi.org/10.1007/978-3-319-89884-1_25

[45] Leonid Libkin. 2004. *Elements Of Finite Model Theory (Texts in Theoretical Computer Science. An Eatcs Series)*. Springer-Verlag.

[46] Ursula Martin and Tobias Nipkow. 1988. Unification in Boolean rings. *Journal of Automated Reasoning* 4, 4 (1988), 381–396.

[47] Ursula Martin and Tobias Nipkow. 1989. Boolean unification—The story so far. *Journal of Symbolic Computation* 7, 3–4 (1989), 275–293. DOI: https://doi.org/10.1016/S0747-7171(89)80013-6

[48] Kenneth L. McMillan and Andrey Rybalchenko. 2012. *Solving Constrained Horn Clauses Using Interpolation*. Technical Report, Microsoft Research, MSR-TR-2013-06.

[49] Yiannis N. Moschovakis. 1974. *Elementary Induction on Abstract Structures*. North Holland.

[50] Andreas Nonnengart and Andrzej Szałas. 1998. *A Fixpoint Approach to Second-Order Quantifier Elimination with Applications to Correspondence Theory. Studies in Fuzziness and Soft Computing*, Vol. 24. Springer, 307–328.

[51] H. J. Ohlbach. 1996. SCAN—Elimination of predicate quantifiers. In *Automated Deduction: CADE-13*. Springer, 161–165.

[52] Enric Rodríguez-Carbonell and Deepak Kapur. 2007. Automatic generation of polynomial invariants of bounded degree using abstract interpretation. *Science of Computer Programming* 64, 1 (2007), 54–75. DOI: https://doi.org/10.1016/J.SCICO.2006.03.003

[53] Enric Rodríguez-Carbonell and Deepak Kapur. 2007. Generating all polynomial invariants in simple loops. *Journal of Symbolic Computation* 42, 4 (2007), 443–476. DOI: https://doi.org/10.1016/J.JSC.2007.01.002

[54] Sergiu Rudeanu. 1974. *Boolean Functions and Equations*. North-Holland.

[55] Philipp Rümmer, Hossein Hojjat, and Viktor Kuncak. 2013. Classifying and solving horn clauses for verification. In *5th International Conference on Verified Software: Theories, Tools, Experiments (VSTTE)*. Ernie Cohen and Andrey Rybalchenko (Eds.), Lecture Notes in Computer Science, Vol. 8164, Springer, 1–21. DOI: https://doi.org/10.1007/978-3-642-54108-7_1

[56] Ernst Schröder. 1890. *Vorlesungen Über Die Algebra Der Logik*. Vol. 1. Teubner.

[57] Ernst Snapper and Robert J. Troyer. 1971. *Metric Affine Geometry*. Academic Press.

[58] Takeshi Tsukada. 2020. On computability of logical approaches to branching-time property verification of programs. In *35th Annual ACM/IEEE Symposium on Logic in Computer Science LICS*. ACM, 886–899. DOI: https://doi.org/10.1145/3373718.3394766

[59] Takeshi Tsukada and Hiroshi Unno. 2022. Software model-checking as cyclic-proof search. *Proceedings of the ACM on Programming Languages* 6, POPL (2022), 1–29. DOI: https://doi.org/10.1145/3498725

[60] Hiroshi Unno, Tachio Terauchi, Yu Gu, and Eric Koskinen. 2023. Modular primal-dual fixpoint logic solving for temporal verification. *Proceedings of the ACM on Programming Languages* 7, POPL (2023), 2111–2140. DOI: https://doi.org/10.1145/3571265

[61] Hiroshi Unno, Sho Torii, and Hiroki Sakamoto. 2017. Automating induction for solving horn clauses. In *29th International Conference on Computer Aided Verification*. Rupak Majumdar and Viktor Kuncak (Eds.), Lecture Notes in Computer Science, Vol. 10427, Springer, 571–591. DOI: https://doi.org/10.1007/978-3-319-63390-9_30

[62] Moshe Y. Vardi. 1982. The complexity of relational query languages (extended abstract). In *14th ACM Symposium on Theory of Computing*. ACM, 137–146. DOI: https://doi.org/10.1145/800070.802186

[63] Christoph Wernhard. 2017. Approximating resultants of existential second-order quantifier elimination upon universal relational first-order formulas. In *Proceedings of the Workshop on Second-Order Quantifier Elimination and Related Topics (SOQE '17)*, 82–98.

[64] Christoph Wernhard. 2017. The Boolean solution problem from the perspective of predicate logic. In *11th International Symposium on Frontiers of Combining Systems (FroCoS)*. Clare Dixon and Marcelo Finger (Eds.), Lecture Notes in Computer Science, Vol. 10483, Springer, 333–350. DOI: https://doi.org/10.1007/978-3-319-66167-4_19

[65] Glynn Winskel. 1993. *The Formal Semantics of Programming Languages—An Introduction*. MIT Press.

Received 15 February 2022; revised 16 March 2026; accepted 29 October 2025