Theorem proving with equality

master thesis in computer science

by

Alexander Maringele

submitted to the Faculty of Mathematics, Computer Science and Physics of the University of Innsbruck

in partial fulfillment of the requirements for the degree of Master of Science

supervisor: Assoc. Prof. Dr. Georg Moser, Institute of Computer Science

Innsbruck, 17 July 2017

Master Thesis

Yet another instantiation based first order theorem prover with equality

Alexander Maringele (8517725)
alexander.maringele@uibk.ac.at

17 July 2017

Supervisor: Assoc. Prof. Dr. Georg Moser

Eidesstattliche Erklärung

Ich erkläre hiermit an Eides statt durch meine eigenhändige Unterschrift, dass ich die vorliegende Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe. Alle Stellen, die wörtlich oder inhaltlich den angegebenen Quellen entnommen wurden, sind als solche kenntlich gemacht.
Die vorliegende Arbeit wurde bisher in gleicher oder ähnlicher Form noch nicht als Magister-/Master-/Diplomarbeit/Dissertation eingereicht.

Datum	Unterschrift

Abstract

Instantiation-based ...

Acknowledgments

Thanks.

Contents

1 Introduction

2 Preliminaries

In this thesis we assume the reader's familiarity with propositional and predicate logic [8], term rewriting [1], decision procedures [9], and satisfiability checking modulo theories [2]. Nevertheless – for clarity – we state basic notions and definitions of first order logic with equality in section 2.1, introduce basic concepts of first order semantics in section 2.2, and describe basic term rewriting terminology in section 2.3. These notions and notations largely follow those in lecture notes [10] and [12].

2.1 Syntax

In this section we introduce the syntax of arbitrary first order formualae (FOF) and clausal normal form (CNF).

Definition 2.1. A first order *signature* with equality $\mathcal{F} = \mathcal{F}_\mathsf{f} \uplus \mathcal{F}_\mathsf{P} \uplus \{\approx\}$ is the disjoint union of a set of *function symbols* $\mathcal{F}_\mathsf{f}$, a set of *predicate symbols* $\mathcal{F}_\mathsf{P}$, and an equality symbol. The *arity* of a symbol determines the number of its arguments in a first order expression. With $\mathcal{F}^{(n)} = \{f \in \mathcal{F} \mid \text{arity}(f) = n\}$ we denote symbols with arity n.

Definition 2.2. We build the set of (first order) *terms* $\mathcal{T}_\mathsf{f} = \mathcal{T}(\mathcal{F}_\mathsf{f}, \mathcal{V})$ from function symbols and a countable set of *variables* $\mathcal{V}$ disjoint from $\mathcal{F}$. Every variable $x \in \mathcal{V}$ is a term, every *constant* $\mathsf{c} \in \mathcal{F}_\mathsf{f}^{(0)}$ is a term, and every expression $\mathsf{f}(t_1, \ldots, t_n)$ is a term for $n > 0$, function symbol $\mathsf{f} \in \mathcal{F}_\mathsf{f}^{(n)}$, and arbitrary terms $t_1, \ldots, t_n$.

Definition 2.3. For an unary function symbol $\mathsf{g} \in \mathcal{F}_\mathsf{f}^{(1)}$, a natural number $i \in \mathbb{N}$, and an arbitrary term $t \in \mathcal{T}_\mathsf{f}$ we introduce the notation $\mathsf{g}^i(t)$ defined as follows:

$$\mathsf{g}^0(t) := t \qquad \mathsf{g}^{i+1}(t) := \mathsf{g}(\mathsf{g}^i(t))$$

Definition 2.4. We build the set of (first order) *predicates* $\mathcal{P}(\mathcal{F}_\mathsf{P}, \mathcal{T}_\mathsf{f})$ from predicate symbols and terms. Every proposition $\mathsf{p} \in \mathcal{F}_\mathsf{P}^{(0)}$ is a predicate, and every expression $\mathsf{P}(t_1, \ldots, t_n)$ is a predicate for $n > 0$, predicate symbol $\mathsf{P} \in \mathcal{F}_\mathsf{P}^{(n)}$ and arbitrary terms $t_1, \ldots, t_n$. We build the set of (first order) *equations* $\mathcal{E}(\approx, \mathcal{T}_\mathsf{f})$ from the equality symbol and terms. Every expression $s \approx t$ is an equation for arbitrary terms s and t. The set of atomic formulas (or *atoms* for short) is the union of predicates and equations.

Definition 2.5 (FOF)**.** Atoms from the previous definition are *first order formulae*. The universal quantification $(\forall x F)$ and the existential quantification $(\exists x F)$ of a first order formula are (quantified) first order formulae. The negation $(\neg F)$ of a first order formula is a (composite) first order formula. Further, the disjunction $(F \vee F')$, the conjunction

$(F \wedge F')$, and the implication $(F \to F')$ of two first order formulae are (composite) first order formulae.

Definition 2.6. We define the set of variables of a first order term t and the set of *free* variables of a first order formula F as follows:

$$\mathit{Vars}(t) = \begin{cases} \{x\} & \text{if } t = x \in \mathcal{V} \\ \bigcup_{i=1}^{n} \mathit{Vars}(t_i) & \text{if } t = \mathsf{f}(t_1, \ldots t_n), \mathsf{f} \in \mathcal{F}_{\mathsf{f}}^{(n)} \end{cases}$$

$$\mathit{Fvars}(F) = \begin{cases} \bigcup_{i=1}^{n} \mathit{Vars}(t_i) & \text{if } F = P(t_1, \ldots, t_n) \text{ or } t_1 \approx t_2 \\ \mathit{Fvars}(G) \setminus \{x\} & \text{if } F \in \{\, \forall x\, G, \exists x\, G \,\} \\ \mathit{Fvars}(G) \cup \mathit{Fvars}(H) & \text{if } F \in \{\, G \wedge H, G \vee H, G \to H \,\} \\ \mathit{Fvars}(G) & \text{if } F = \neg G \end{cases}$$

A first order *sentence* is a closed formula, i.e. without free variables, over a signature.

Definition 2.7 (`CNF`)**.** A (first order) *literal* L is either an atom A or the negation $(\neg A)$ of an atom. We usually abbreviate $\neg(s \approx t)$ with $s \not\approx t$. The *complement* L^c of an atom (positive literal) is the negation of the atom. The complement of a negated atom (negative literal) is the atom itself. A (first order) *clause* $\mathcal{C} = L_1 \vee \ldots \vee L_n$ is a possible empty multiset of literals. A finite *set of clauses* $S = \{\mathcal{C}_1, \ldots, \mathcal{C}_n\}$ is in *clausal normal form*.

Remark. First order terms, atoms, literals, clauses, and formulae are first order expressions. We call a first order expression without variables *ground*, i.e. we build our ground expressions over an empty set of variables.

Definition 2.8. A *substitution* σ is a mapping from variables $x \in \mathcal{V}$ to terms in $\mathcal{T}(\mathcal{F}_{\mathsf{f}}, \mathcal{V})$ where the *domain* $\mathrm{dom}(\sigma) = \{x \in \mathcal{V} \mid \sigma(x) \neq x\}$ is finite. We write substitutions as bindings, e.g. $\sigma = \{x_1 \mapsto s_1, \ldots, x_n \mapsto s_n\}$ where $\mathrm{dom}(\sigma) = \{x_1, \ldots, x_n\}$ and $\sigma(x_i) = s_i$. A *variable substitution* is a mapping from $\mathcal{V}$ to $\mathcal{V} \subseteq \mathcal{T}(\mathcal{F}_{\mathsf{f}}, \mathcal{V})$. A *renaming* is a bijective variable substitution. A *proper instantiator* is a substitution that is not a variable substitution. We define the instance $t\sigma$ respectively the application of a substitution σ to a literal or term t as follows

$$t\sigma = \begin{cases} s_i & \text{if } t = x_i \in \mathrm{dom}(\sigma) \\ y & \text{if } t = y \in \mathcal{V} \setminus \mathrm{dom}(\sigma) \\ f(t_1\sigma, \ldots, t_n\sigma) & \text{if } t = f(t_1, \ldots, t_n) \text{ where } f \in \mathcal{F}^{(n)} \\ \neg(A\sigma) & \text{if } t = \neg A, \text{ where } A \text{ is an atom} \end{cases}$$

We define the instance of a clause as the multiset of the instances of its literals.

Remark. We write $F(x)$ for a clause or formula to express that $\mathit{Fvars}(F) = \{x\}$ and we abbreviate $F(x)\{x \mapsto t\}$ with $F(t)$, i.e. all occurrences of the first and only free variable in F is substituted with term t.

Definition 2.9. A clause $\mathcal{C}$ *subsumes* a clause $\mathcal{D}$ if their exists a substitution θ such that $\mathcal{C}\theta \subseteq \mathcal{D}$, i.e. clause $\mathcal{D} = \mathcal{C}\theta \vee \mathcal{D}'$ is a weakened instance of clause $\mathcal{C}$.

Definition 2.10. We define the *composition* of two substitutions σ and τ as follows

$$\sigma\tau = \{x_i \mapsto s_i\tau \mid x_i \in \mathrm{dom}(\sigma)\} \cup \{y_i \mapsto t_i \mid y_i \in \mathrm{dom}(\tau) \setminus \mathrm{dom}(\sigma)\}.$$

Lemma 2.11. *With the definitions in 2.8 and 2.10 the equation $(t\sigma)\tau = t(\sigma\tau)$ holds for arbitrary first order expressions and substitutions.*

Proof. Assume σ and τ are substitutions. Then we use induction on the structure of an arbitrary expression t that the equation $(t\sigma)\tau = t(\sigma\tau)$ holds in all possible cases.

- (base case) Let $t = x_i \in \mathrm{dom}(\sigma)$ then $((x_i)\sigma)\tau \overset{\text{def}}{=} s_i\tau \overset{\text{def}}{=} x_i(\sigma\tau)$ holds.
- (base case) Let $t = y \notin \mathrm{dom}(\sigma)$ then $(y\sigma)\tau \overset{\text{def}}{=} y\tau \overset{\text{def}}{=} y(\sigma\tau)$ holds.
- (step case) Let $t = f(t_1, \ldots, t_n)$ then $((f(t_1, \ldots, t_n))\sigma)\tau \overset{\text{def}}{=} (f(t_1\sigma, \ldots, t_n\sigma))\tau \overset{\text{def}}{=} f((t_1\sigma)\tau, \ldots, (t_n\sigma)\tau) \overset{\text{IH}}{=} f(t_1(\sigma\tau), \ldots, t_n(\sigma\tau)) \overset{\text{def}}{=} (f(t_1, \ldots, t_n))(\sigma\tau)$ holds.
- (step case) Let $t = \neg A$ then $((\neg A)\sigma)\tau \overset{\text{def}}{=} (\neg(A\sigma))\tau \overset{\text{def}}{=} \neg((A\sigma)\tau) \overset{\text{IH}}{=} \neg(A(\sigma\tau)) \overset{\text{def}}{=} (\neg A)(\sigma\tau)$ holds.

□

Definition 2.12. Two first order expressions t, u are *unifiable* if there exists a *unifier*, i.e. a substitution σ such that $t\sigma = u\sigma$. The *most general unifier* $\mathrm{mgu}(t, u)$ is a unifier such that for every other unifier σ' there exists a substitution τ where $\sigma' = \sigma\tau$. Two literals are variants if their most general unifier is a renaming. Two literals are *clashing* when the first literal and the complement of the second literal are unifiable, i.e. literals L' and L are clashing if $\mathrm{mgu}(L', L^c)$ exists.

Example 2.13. Literals $\mathsf{P}(x)$ and $\neg\mathsf{P}(\mathsf{f}(\mathsf{a}, y))$ are clashing by unifier $\{x \mapsto \mathsf{f}(\mathsf{a}, y)\}$.

Definition 2.14. A *position* is a finite sequence of positive integers. The root position is the empty sequence ϵ. The position pq is obtained by concatenation of positions p and q. A position p is *above* a position q if p is a prefix of q, i.e. there exists a unique position r such that $pr = q$, we write $p \leq q$ and $q\backslash r = p$. We write $p < q$ if p is a proper prefix of q, i.e. $p \leq q$ but $p \neq q$. We define $\mathrm{head}(iq) = i$ and $\mathrm{tail}(iq) = q$) for $i \in \mathbb{N}$, $q \in \mathbb{N}^*$, further $\mathrm{length}(\epsilon) = 0$, $\mathrm{length}(iq) = 1 + \mathrm{length}(q)$. Two positions $p \parallel q$ are parallel if none is above the other, i.e. for any common prefix r both remaining tails $p\backslash r$ and $q\backslash r$ are different and not root positions. A position p is left of position q if $\mathrm{head}(p\backslash r) < \mathrm{head}(p\backslash r)$ for maximal common prefix r.

Definition 2.15. We define the set of *term positions* in an atom or a term,

$$t\text{-}\mathcal{P}os(t) = \begin{cases} \{\epsilon\} & \text{if } t = x \in \mathcal{V} \\ \{\epsilon\} \cup \bigcup_{i=1}^{n} \{iq \mid q \in t\text{-}\mathcal{P}os(t_i)\} & \text{if } t = \mathsf{f}(t_1, \ldots, t_n), \mathsf{f} \in \mathcal{F}_{\mathsf{f}}^{(n)} \\ \cancel{\{\epsilon\} \cup} \bigcup_{i=1}^{n} \{iq \mid q \in t\text{-}\mathcal{P}os(t_i)\} & \text{if } t = \mathsf{P}(t_1, \ldots, t_n), \mathsf{P} \in \mathcal{F}_{\mathsf{P}}^{(n)} \\ \cancel{\{\epsilon\} \cup} \{1q \mid q \in t\text{-}\mathcal{P}os(t_1)\} \cup \{2q \mid q \in t\text{-}\mathcal{P}os(t_2)\} & \text{if } t = t_1 \approx t_2 \end{cases}$$

the extraction of a subterm at a term position from an atom or a term,

$$t|_p = \begin{cases} t & \text{if } p = \epsilon \\ t_i|_q & \text{if } t = f(t_1, \ldots, t_n), p = iq, f \in \mathcal{F}^{(n)} \end{cases}$$

and the insertion of a term s at a term position into an atom or a term by replacing the subterm at that term position.

$$t[s]_p = \begin{cases} s & \text{if } p = \epsilon, (t \text{ is a term}) \\ f(t_1, \ldots, t_i[s]_q, \ldots, t_n) & \text{if } t = f(t_1, \ldots, t_n), p = iq, f \in \mathcal{F}^{(n)}, 0 < i \leq n \end{cases}$$

2.2 Semantics

In this section we recall some basic aspects and definitions of semantics in first order logic. We state satisfiability of clauses. See the appendix for more details and mathematics.

2.2.1 Models

Definition 2.16. An *interpretation* $\mathcal{I}$ over a signature $\mathcal{F}$ consists of a non-empty set A – the *universe* or *domain*, definitions of mappings $\mathsf{f}_\mathcal{I} : A^n \to A$ for every function symbol $\mathsf{f} \in \mathcal{F}_\mathsf{f}$, and definitions of (possibly empty) n-ary relations $\mathsf{P}_\mathcal{I} \subseteq A^n$ for every predicate symbol $\mathsf{P} \in \mathcal{F}_\mathsf{P}$ and the definition of a binary relation $\approx_\mathcal{I} \subseteq A^2$ for the equality symbol. A (variable) *assignment* is a mapping from variables to elements of the domain. We define the *evaluation* $\alpha_\mathcal{I}$ of a term t for assingment α and interpretation $\mathcal{I}$:

$$\alpha_\mathcal{I}(t) = \begin{cases} \alpha_\mathcal{I}(x) & \text{if } t = x \in \mathcal{V} \\ \mathsf{c}_\mathcal{I} & \text{if } t = \mathsf{c} \in \mathcal{F}_\mathsf{f}^{(0)} \\ \mathsf{f}_\mathcal{I}(\alpha_\mathcal{I}(t_1), \ldots, \alpha_\mathcal{I}(t_n)) & \text{if } t = \mathsf{f}(t_1, \ldots, t_n), \mathsf{f} \in \mathcal{F}_\mathsf{f}^{(n>0)}, t_i \in \mathcal{T}_\mathsf{f} \end{cases}$$

Remark. The evaluation of ground terms does not depend on variable assignments.

Definition 2.17. A predicate $\mathsf{P}(t_1, \ldots, t_n)$ *holds* for an assignment $\alpha_\mathcal{I}$ if and only if the evaluation of its n-tuple $(\alpha_\mathcal{I}(t_1), \ldots, \alpha_\mathcal{I}(t_n))$ is an element of the relation $\mathsf{P}_\mathcal{I} \subseteq A^n$. Similar an equation $s \approx t$ holds if $\alpha_\mathcal{I}(s) \approx_\mathcal{I} \alpha_\mathcal{I}(t)$.

Definition 2.18 (Semantics of `FOF`)**.** A universally quantified formula $\forall x F$ holds in an interpretation if its subformula F holds for all assignments for x. An existential quantified formula $\exists x F$ holds if its subformula F holds for at least one assignment for x. A negation $\neg F$ holds if its subformula F does not hold, a disjunction $F \vee G$ holds if one or both of its subformulae F or G hold, a conjunction $F \wedge G$ holds, if both of its subformualae F and G hold, an implication $F \to G$ holds if its first subformula F does not hold or its second subformula F' holds (or both).

Remark. Usually we use precedences on connectives to omit parentheses and some heuristics to structure the formulae for readability without introducing semantic ambiguity.

Definition 2.19 (Semantics of CNF)**.** An atom holds in an interpretation if and only if it holds with all possible assignments. A literal holds if and only if its complement does not hold. A clause holds if at least one of its literals holds, hence the empty clause $\square$ does not hold in any interpretation. A set of clauses holds if and only if every clause in the set holds.

Definition 2.20. A *model* $\mathcal{M}$ for a set of clauses (a formula) is an interpretation that *satisfies* the set of clauses (the formula), i.e. the set of clauses (the formula) holds in this interpretation. A set of clauses (a formula) is *satisfiable* if and only if there is at least one model for it. A set of clauses (a formula) is *valid* if and only if every interpretation is a model.

Example 2.21. The set of clauses $\{ \neg\mathsf{E}(\mathsf{z}, \mathsf{f}(x)) \}$ is satisfiable if and only if the quantified first order formula $\exists x \forall y \, (\neg\mathsf{E}(x, y))$ without free variables is satisfiable.

Definition 2.22. The *Herbrand universe* for a first order signature $\mathcal{F}$ is the smallest set of terms that contains all $H_{i\geq 0}$ defined inductively as

$$H_0 = \begin{cases} \{ \mathsf{c} \mid \mathsf{c} \in \mathcal{F}_{\mathsf{f}}^{(0)} \} & \text{if } \mathcal{F}_{\mathsf{f}}^{(0)} \neq \emptyset \\ \{ \mathsf{c}_0 \} & \text{if } \mathcal{F}_{\mathsf{f}}^{(0)} = \emptyset, \mathsf{c}_0 \notin \mathcal{F} \end{cases} \qquad H'_0 = H_0$$

$$H_{k+1} = \bigcup_{n>0} \{ \mathsf{f}(t_1, \ldots, t_n) \mid \mathsf{f} \in \mathcal{F}_{\mathsf{f}}^{(n)}, t_1, \ldots, t_n \in H'_k \} \qquad H'_{k+1} = H_k \cup H'_{k+1}$$

Definition 2.23. An *Herbrand interpretation* $\mathcal{H}$ is an interpretation where the domain is an Hebrand universe and the interpretation of each ground term $t_{\mathcal{H}} := t$ is the term itself.

2.2.2 Equality

In Definition 2.17 we have interpreted the equality symbol as binary relation without restrictions. This of course allows undesirable models as in the following Example 2.24. Hence we introduce useful definitions to deal with this situation and demonstrate their usage in an example.

Example 2.24. Any interpretation $\mathcal{I}$ with $\approx_{\mathcal{I}} = \emptyset$ satisfies $\mathsf{a} \not\approx \mathsf{a}$.

Definition 2.25. An *normal* interpretation defines $\approx_{\mathcal{I}}$ as identity on its domain, e.g. the equation of terms $s \approx_{\mathcal{I}} t$ holds if and only if any evaluation of its terms are equal $\alpha_{\mathcal{I}}(s) = \alpha_{\mathcal{I}}(t)$ for all assignments α. In other words a normal interpretation yields different elements for ground terms s' and t' if and only if $s' \not\approx_{\mathcal{I}} t'$.

Definition 2.26. A *term interpretation* $\mathcal{I}_t$ is an interpretation where the elements of its domain $A = \mathcal{T}(\mathcal{F}_{\mathsf{f}}, \emptyset)/_{\sim}$ are equivalence classes of ground terms and the interpretation of each ground term $t^{\mathcal{I}_t} := [t]_{\sim}$ is its equivalence class. A ground predicate $\mathsf{P}(t_1, \ldots, t_n)$ holds if $([t_1]_{\sim}, \ldots, [t_n]_{\sim}) \in \mathsf{P}^{\mathcal{I}_t} \subseteq A^n$.

Example 2.27. Consider the satisfiable set of clauses $S = \{\mathsf{f}(x) \approx x\}$. We easily find a Herbrand model $\mathcal{H}$ with predicate definition $\approx_{\mathcal{H}} = \{(\mathsf{f}^{i+1}(\mathsf{a})), \mathsf{f}^i(\mathsf{a})) \mid i \geq 0\}$. However $\mathcal{H}$ is not a normal model because obviously $\mathsf{f}(\mathsf{a}) \neq \mathsf{a}$ in its domain. Further on we easily find an normal model $\mathcal{M}$ with domain $\{\mathsf{c}\}$, function definition $\mathsf{f}_{\mathcal{M}}(\mathsf{c}) \mapsto \mathsf{c}$, and the relation $\approx_{\mathcal{M}} = \{(\mathsf{c}, \mathsf{c})\}$ coincides with identity in its domain. Certainly this model $\mathcal{M}$ is not an Hebrand model because the interpretation of ground term $\mathsf{f}(\mathsf{c})_{\mathcal{M}} = \mathsf{c}$ is not the ground term $\mathsf{f}(\mathsf{c})$ itself. On the other hand we easily construct a normal term model $\mathcal{M}_t$ with domain $\{[\mathsf{a}]_\sim\}$, a plain function definition $\mathsf{f}_{\mathcal{M}_t}([\mathsf{a}]_\sim) \mapsto [\mathsf{a}]_\sim$ with equivalence relation $\mathsf{a} \sim \mathsf{f}(\mathsf{a})$. Hence $\approx_{\mathcal{M}_t}$ agrees to equality in its domain of equivalence classes of ground terms.

2.3 Term Rewriting

Definition 2.28. A *rewrite rule* is an equation of terms where the left-hand side is not a variable and the variables occuring in the right-hand side occur also in the left-hand side. A rewrite rule $\ell' \to r'$ is a *variant* of $\ell \to r$ if there is a renaming ϱ such that $(\ell \to r)\varrho := \ell\varrho \to r\varrho = l' \to r'$. A *term rewrite system* is a set of rewrite rules without variants.

Definition 2.29. We say $s \to_{\mathcal{R}} t$ is a *rewrite step* with respect to TRS $\mathcal{R}$ when there is a position $p \in \text{t-}\mathcal{P}os(s)$, a rewrite rule $\ell \to r \in \mathcal{R}$, and a substitution σ such that $s|_p = \ell\sigma$ and $s[r\sigma]_p = t$. The subterm $\ell\sigma$ is called *redex* and s rewrites to t by *contracting* $\ell\sigma$ to *contractum* $r\sigma$. We say a term s is *irreducible* or in *normal form* with respect to TRS $\mathcal{R}$ if there is no rewrite step $s \to_{\mathcal{R}} t$ for any term t. The set of normal forms $\mathsf{NF}(\mathcal{R})$ contains all irreducible terms of the TRS $\mathcal{R}$.

Definition 2.30. A term s can be rewritten to term t with notion $s \to^*_{\mathcal{R}} t$ if there exists at least one *rewrite sequence* $(a_1, \ldots, a_n)$ such that $s = a_1$, $a_n = t$, and $a_i \to_{\mathcal{R}} a_{i+1}$ are rewrite steps for $1 \leq i < n$. A TRS is *terminating* if there is no infinite rewrite sequence of terms. Two Terms s and t are *joinable* with notion $s \downarrow t$ if both can be rewritten to some term c, i.e. $s \to^* c \ {}^*\!\leftarrow t$. Two Terms s and t are *meetable* with notion $s \uparrow t$ if both can be rewritten from some common ancestor term a, i.e. $s \leftarrow^* a \ {}^*\!\to t$. A TRS is *confluent* if s and t are joinable whenever $s \ {}^*\!\leftarrow a \to^* t$ holds for some term a. Terms s and t are *convertible* with notion $s \leftrightarrow^* t$ if there exists a sequence $(a_1, \ldots, a_n)$ such that $s = a_1$, $a_n = t$, and $a_i \leftrightarrow a_{i+1}$, i.e. $a_i \to a_{i+1}$ or $a_i \leftarrow a_{i+1}$ are rewrite steps for $1 \leq i < n$.

Definition 2.31. A *rewrite relation* is a binary relation $\circledast$ on arbitrary terms s and t, which additionally is *closed under contexts* (whenever $s \circledast t$ then $u[s]_p \circledast u[t]_p$ for an arbitrary term u and any position $p \in \text{t-}\mathcal{P}os(u)$) and *closed under substitutions* (whenever $s \circledast t$ then $s\sigma \circledast t\sigma$ for an arbitrary substitution σ).

Lemma 2.32. *The relations $\to^*_{\mathcal{R}}$, $\to^+_{\mathcal{R}}$, $\downarrow_{\mathcal{R}}$, $\uparrow_{\mathcal{R}}$ are rewrite relations on every TRS $\mathcal{R}$.*

Definition 2.33. A proper (i.e. irreflexive and transitive) order on terms is called *rewrite order* if it is a rewrite relation. A *reduction order* is a well-founded rewrite order, i.e. there

is no infinite sequence $(a_i)_{i\in\mathbb{N}}$ where $a_i \succ a_{i+1}$ for all i. A *simplification order* is a rewrite order with the *subterm property*, i.e. $u[t]_p \succ t$ for all terms u, t and positions $p \neq \epsilon$.

Lemma 2.34. *Every simplification order is well-founded, hence it is a reduction order.*

3 Undecidabitlity

We call a logical calculus, i.e. a formal proof system for an underlying logic, *complete* if every true formula is valid, i.e. provable or derivable in the calculus. In other words, any expression that is true, i.e. satisfied in all possible interpretations, is deducible or justifiable by applying rules of the formal system only. We expect a useful calculus to be *sound*, that is, every valid formula is true, i.e. it holds in any interpretation. We call a logical system *decidable* if and only if a purely syntactical algorithm exists that determines the validity or non-validity for any arbitrary formula in the given logical system.

We first look at a sound and complete calculus for first-order logic in 3.1. After that we recall undecidability of first-order logic in 3.2 together with other useful theorems about first-order logic. Then we enumerate decidable fragments of first-order logic which can be described purely syntactically in 3.3 . After that we look at decidable first-order theories in 3.4, which are not necessarily expressible in one of the syntactically describable and decidable fragments of first-order logic.

3.1 Natural Deduction

In Natural Deduction [8] we start with a (possible empty) set of formulae – the premises – and infer other formulae – the conclusions – by using syntactic proof rules from Tables 3.1−3.3. The letters F and G in these rules represent arbitrary first-order formulae as defined in Definition 2.5 on page 2. We open a box when we assume something, e.g. that F holds. We close the box when we discard the assumption. With $F_1, \ldots, F_n \vdash H$ we claim that formula H is in the transitive closure of inferable formulae from $\{F_1, \ldots, F_n\}$. (you find examples to the natural deduction proofs to Lemma 3.23 on page 13).

Theorem 3.1 (Soundness)**.** *The inference rules of natural deduction are sound.*

3.2 Theorems of First Order Logic

The purpose of this section is to state useful theorems about first-order logic.

Theorem 3.2 (Completeness, Gödel 1929)**.** *Natural deduction (see above), a sound formal system, is complete.*

Theorem 3.3 (Undecidability, Church 1936, Turing 1937)**.** *The satisfiability problem for first-order logic is undecidable.*

$$\frac{F \quad G}{F \wedge G}\,(\wedge i) \qquad \frac{F \wedge G}{G}\,(\wedge e_1) \qquad \frac{F \wedge G}{F}\,(\wedge e_2) \qquad \frac{F}{\neg\neg F}\,(\neg\neg i) \qquad \frac{\neg\neg F}{F}\,(\neg\neg e)$$

$$\frac{\bot}{F}\,(\bot e) \qquad \frac{F \quad \neg F}{\bot}\,(\neg e) \qquad \frac{}{F \vee \neg F}\ \text{LEM} \qquad \frac{F}{F \vee G}\,(\vee i_1) \qquad \frac{G}{F \vee G}\,(\vee i_2)$$

$$\frac{\boxed{\begin{array}{c} F \\ \vdots \\ \bot \end{array}}}{\neg F}\,(\neg i) \qquad \frac{\boxed{\begin{array}{c} \neg F \\ \vdots \\ \bot \end{array}}}{F}\ \text{PBC} \qquad \frac{\boxed{\begin{array}{c} F \\ \vdots \\ G \end{array}}}{F \to G}\,(\to i) \qquad \frac{F \vee G \qquad \boxed{\begin{array}{c} F \\ \vdots \\ H \end{array}} \qquad \boxed{\begin{array}{c} G \\ \vdots \\ H \end{array}}}{H}\,(\vee e)$$

$$\frac{F \quad F \to G}{G}\ \begin{array}{l}\text{modus}\\ \text{ponens}\end{array} \qquad \frac{F \to G \quad \neg G}{\neg F}\ \begin{array}{l}\text{modus}\\ \text{tollens}\end{array}$$

Table 3.1: Natural Deduction Rules for Connectives

$$\frac{}{t = t}\,(=i) \qquad \frac{s = t \quad F\{x \mapsto s\}}{F\{x \mapsto t\}}\,(=e)$$

Table 3.2: Natural Deduction Rules for Equality

Theorem 3.4 (Trakhtenbort 1950, Craig 1950)**.** *The satisfiability problem for first-order logic on* finite *structures (domains) is undecidable.*

Definition 3.5 (Finite model property)**.** A logic has the finite model property if each non-theorem is falsified by some finite model.

Lemma 3.6 (Refutation)**.** *By definition of the semantics of negation a formula is valid if and only if its negation is not satisfiable.*

Theorem 3.7 (Compactness, Gödel 1930, Maltsev 1936)**.** *If every finite subset of a set of formulas S has a model then S has a model.*

Theorem 3.8 (Löwenheim Skolem, 1915, 1920)**.** *If a set of formulas S has a model then S has a countable model.*

Theorem 3.9 (Herbrand, 1930)**.** *Let S be a set of clauses without equality. Then the following statements are equivalent.*

- *S is satisfiable.*
- *S has a Herbrand model.*
- *Every finite subset of all ground instances of S has a Herbrand model.*

$$\frac{\forall x F}{F\{x \to t\}}\ (\forall e) \qquad \frac{F\{x \mapsto x_0\}}{\exists x F}\ (\exists i)$$

$$\frac{\begin{array}{|l} x_0 \\ \quad \vdots \\ \quad F\{x \mapsto x_0\} \end{array}}{\forall x F}\ (\forall i) \qquad \frac{\exists x F \quad \begin{array}{|ll} x_0 & F\{x \mapsto x_0\} \\ & \vdots \\ & H \end{array}}{H}\ (\exists e)$$

Table 3.3: Natural Deduction Rules for Quantifiers

Corollary 3.10. *Let S be a set of clauses without equality. Then S is unsatisfiable if and only if there exists an unsatisfiable finite set of ground instances of S.*

Lemma 3.11. *With Skolemization and Tseitin transformation we can effectively transform a arbitrary first-order formula into an equisatisfiable set of clauses.*

3.3 Decidable Fragments of First Order Logic

This section presents purely syntactical defined fragments of first-order logic where satisfiability is decidable.[1]

Definition 3.12 (PNF)**.** A first-order formula $F = \exists\!\!\!/_1 x_1 \ldots \exists\!\!\!/_n x_n\, G$ where $\exists\!\!\!/_i \in \{\exists, \forall\}$ are quantifiers for $1 < i \leq n$, the subformula G is quantifier free, and the set of free variables $\mathcal{F}\mathit{vars}(G) = \{x_1, \ldots, x_n\}$ matches the quantified variables is in *prenex normal form*.

Definition 3.13 ([3])**.** We describe classes of first-order formulae in PNF with triples

$$[\,\Pi, (p_1, p_2, \ldots), (f_1, f_2, \ldots)\,]_{(\approx)} \subseteq [\,all, all, all\,]_{\approx}$$

where $\Pi = \exists\!\!\!/_1 \ldots \exists\!\!\!/_n$, $\exists\!\!\!/_i \in \{\forall, \exists\}$ describes the structure of the quantifier prefix (without variables) of the formulae, the value p_i is the maximal number of predicate symbols with arity i, and the value f_i the maximal number of function symbols with arity i in the signature. The equality symbol is not counted as binary predicate symbol. Instead, the absence or presence of subscript $(\approx)$ indicates whether the equality symbol is in the signature.

Example 3.14. The monadic predicate calculus includes formulae with arbitrary quantifier prefixes, arbitrary many unary predicate symbols, the equality symbol, but no function symbols.

$$[\,all, (\omega), (1)\,]_{=} \quad \supsetneq \quad [\,all, (\omega), (0)\,]_{=} \qquad \text{(Löwenheim 1925, Kalmár 1929)}$$

[1] Definitions and compact overviews follow the presentation "Decidable fragments of first-order and fixed-point logic" by E. Grädel (`http://logic.rwth-aachen.de/~graedel/`).

Example 3.15. The Ackermann prefix class contains formulae with arbitrary many existential quantifiers, but just one universal quantifier. It contains arbitrary many predicate symbols with arbitrary arities, the equality symbol, but no function symbols.

$$[\,\exists^*\forall\exists^*, all, (1)\,]_= \quad \supsetneq \quad [\,\exists^*\forall\exists^*, all, (0)\,]_= \qquad \text{(Ackermann 1928)}$$

Remark. One unary function symbol can be added to these fragments of first order logic above without loosing decidability (see Table 3.5).

$[\,\exists^*\forall^*, all, (0)\,]_=$	(Bernays, Schönfinkel 1928, Ramsey 1932)
$[\,\exists^*\forall^2\exists^*, all, (0)\,]$	(Gödel 1932, Kalmár 1933, Schütte 1934)
$[\,all, (\omega), (\omega)\,]$	(Löb 1967, Gurevich 1969)
$[\,\exists^*\forall\exists^*, all, all\,]$	(Gurevich 1973)
$[\,\exists^*, all, all\,]_=$	(Gurevich 1976)

Table 3.4: Decidable prefix classes with finite model property

$[\,all, (\omega), (1)\,]_=$	(Rabin 1969)
$[\,\exists^*\forall\exists^*, all, (1)\,]_=$	(Shelah 1977)

Table 3.5: Decidable prefix classes with infinity axioms.

Lemma 3.16. *Satisfiability is decidable [3] in all prefix classes from Tables 3.4 and 3.5. Each of theses classes is closed under conjunction with respect to satisfiability.*

3.4 Theories in First Order Logic

We follow definitions and examples in [11].

Definition 3.17 (Theory)**.** A *first-order theory* is a pair of a first-order signature and the possible infinite conjunction $\bigwedge_i A_i$ of first-order formulae, i.e. the axioms, over the theory's signature. A theory is *consistent* if the contradiction is not derivable. A theory is satisfiable if there exists a model for its axioms. A *theorem* is a sentence over the theory's signature, i.e. a closed formula, that holds in any model for the theory's axioms.

$$\bigwedge_i A_i \models \mathsf{theorem} \quad \text{or} \quad \bigwedge_i A_i \to \mathsf{theorem}$$

A theory is decidable if it is decidable whether an arbitrary sentence holds in the theory.

Example 3.18. A theory with axioms $\forall x\, \mathsf{P}(x)$ and $\exists x\, \neg \mathsf{P}(x)$ is neither consistent nor satisfiable.

Lemma 3.19. *A first order theory is consistent if and only if it is satisfiable.*

Remark. In refuational theorem proving we show the unsatisfiabilty of a negated sentence, i.e. a *conjecture*, in conjunction with the axioms to conclude that the conjecture is indeed a theorem.

$$\neg \left(\bigwedge_i A_i \to \mathsf{conj} \right) \equiv \neg \left(\neg \bigwedge_i A_i \vee \mathsf{conj} \right) \equiv \bigwedge_i A_i \wedge \neg \mathsf{conj}$$

3.4.1 Theory of equality

The following equivalence and congruence axioms form the theory of equality over a first order signature.

Definition 3.20 (Equivalence)**.** A binary relation $\approx$ over a domain is an equivalence relation if and only if the following axioms hold over the given domain.

$$\begin{aligned} \forall x\ (x \approx x) && \text{reflexivity} \\ \forall x \forall y\ (x \approx y \to y \approx x) && \text{symmetry} \\ \forall x \forall y \forall z\ (x \approx y \wedge y \approx z \to x \approx z) && \text{transitiviy} \end{aligned}$$

Definition 3.21 ($\vec{x}$-Notation)**.** Occasionally we may abbreviate a sequence of n variables by $\vec{x}$. Then we write $f(\vec{x})$ for first-order expression $f(x_1, \ldots, x_n)$ with n-ary function or predicate symbol f, a single equation $\vec{x} \approx \vec{y}$ for the conjunction of n equations $x_1 \approx y_1 \wedge \ldots \wedge x_n \approx y_n$, and $\forall \vec{x}$ for the sequence of quantified variables $\forall x_1 \ldots \forall x_n$.

Definition 3.22 (Congruence schemata)**.** An equivalence relation $\approx$ is a congruence relation if and only if the following formulae hold

$$\begin{aligned} \forall \vec{x} \forall \vec{y}\ (\vec{x} \approx \vec{y} \to \mathsf{f}(\vec{x}) \approx \mathsf{f}(\vec{y})) && \text{for all } \mathsf{f} \in \mathcal{F}_{\mathsf{f}}^{(n)} \\ \forall \vec{x} \forall \vec{y}\ (\vec{x} \approx \vec{y} \to (\mathsf{P}(\vec{x}) \to \mathsf{P}(\vec{y}))) && \text{for all } \mathsf{P} \in \mathcal{F}_{\mathsf{P}}^{(n)} \end{aligned}$$

Lemma 3.23. *Equivalence and congruence of equality are provable by Natural Deduction (see Section 3.1 on page 9).*

Proof. For brevity we skip the quantifier introductions for symmetry, transitivity, and congruence. Additionally we just show congruence for a unary function and a unary

predicate symbol.

1	$\boxed{x_0 \quad x_0 = x_0}$	$=i$
2	$\forall x\,(x = x)$	$\forall i, 1$

1	$y = y$	$=i$
2	$\boxed{x = y}$	assume
3	$\boxed{y \neq x}$	assume
4	$y \neq y$	$=e, 2, 3$
5	$\bot$	$\neg e, 1, 4$
6	$y = x$	PBC, 3–5
7	$x = y \to y = x$	$\to i, 1$–5

1	$\boxed{x = y \wedge y = z}$	assume
2	$y = z$	$\wedge e_2$
3	$x = y$	$\wedge e_1, 2$
4	$x = z$	$=e, 2, 3$
5	$x = y \wedge y = z \to x = z$	$\to i, 2$–4

1	$\mathsf{f}(y) = \mathsf{f}(y)$	$=i$
2	$\boxed{x = y}$	assume
3	$\boxed{\mathsf{f}(x) \neq \mathsf{f}(y)}$	assume
4	$\mathsf{f}(y) \neq \mathsf{f}(y)$	$=e, 2, 3$
5	$\bot$	$\neg e, 1, 4$
6	$\mathsf{f}(x) = \mathsf{f}(y)$	PBC, 3–5
7	$x = y \to \mathsf{f}(x) = \mathsf{f}(y)$	$\to i, 1$–5

1	$\boxed{x = y}$	assume
2	$\boxed{\mathsf{P}(x)}$	assume
3	$\boxed{\mathsf{P}(y)}$	$=e, 1, 2$
4	$\mathsf{P}(x) \to \mathsf{P}(y)$	$\to i, 2$–3
5	$x = y \to (\mathsf{P}(x) \to \mathsf{P}(y))$	$\to i, 1$–4

□

3.4.2 Natural numbers

The following axioms characterize natural numbers, addition, and multiplication.

Definition 3.24 (Natural Numbers)**.** We introduce a fresh constant $0 \in \mathcal{F}^{(0)}$, a unary successor symbol $\mathsf{s} \in \mathcal{F}^{(1)}$ and restrict their models with two axioms.

$$\forall x\,(\mathsf{s}(x) \not\approx 0) \qquad \text{zero is smallest}$$
$$\forall x \forall y\,(\mathsf{s}(x) \approx \mathsf{s}(y) \to x \approx y) \qquad \text{injectivity of } \mathsf{s}$$
$$\forall x \forall y\,(x \approx y \to \mathsf{s}(x) \approx \mathsf{s}(y)) \qquad \text{congruence of } \mathsf{s}$$
$$\underbrace{G(0)}_{\text{base}} \wedge \forall x'\ \underbrace{(G(x') \to G(\mathsf{s}(x')))}_{\text{step case}} \to \forall x\, G(x) \qquad \text{induction schema}$$

Example 3.25. We may prove $\forall x\,(\mathsf{s}(x) \not\approx x)$ with $G(x) = \mathsf{s}(x) \not\approx x$ by induction.

$$\underbrace{\mathsf{s}(0) \not\approx 0}_{\text{base}} \wedge \forall x'\ \underbrace{(\mathsf{s}(x') \not\approx x' \to \mathsf{s}(\mathsf{s}(x')) \not\approx \mathsf{s}(x'))}_{\text{step case}} \to \forall x\, \mathsf{s}(x) \not\approx x$$

Definition 3.26 (Addition)**.** We introduce the binary addition symbol $+ \in \mathcal{F}^{(2)}_{\mathsf{f}}$ with two axioms about defining equalities of sums.

$$\forall x\,(x + 0 \approx x) \qquad \text{addition of zero}$$
$$\forall x \forall y\,(x + \mathsf{s}(y)) \approx \mathsf{s}(x + y) \qquad \text{addition of non-zero}$$
$$\forall x_1 \forall x_2 \forall y_1 \forall y_2\,(x_1 \approx y_1 \wedge x_2 \approx y_2 \to x_1 + y_1 \approx x_2 + y_2) \qquad \text{congruence of } +$$

Example 3.27.

$$\mathsf{s}(\mathsf{s}(\mathsf{s}(0))) + \mathsf{s}(\mathsf{s}(0)) \approx \mathsf{s}(\mathsf{s}(\mathsf{s}(\mathsf{s}(0))) + \mathsf{s}(0)) \approx \mathsf{s}(\mathsf{s}(\mathsf{s}(\mathsf{s}(\mathsf{s}(0))) + 0)) \approx \mathsf{s}(\mathsf{s}(\mathsf{s}(\mathsf{s}(\mathsf{s}(0)))))$$

Theorem 3.28. *Presburger Artihmetic (Mojżesz Presburger, 1929), i.e. the first-order theory that includes the axioms for equality, natural numbers, induction schemata, and addition, is consistent, complete and decidable. The computational complexity of the decision problem is at least doubly exponential $2^{2^{cn}}$ (Fischer and Rabin, 1974), but less than triple exponential (Oppen, 1978. Berman, 1980).*

Definition 3.29 (Multiplikation)**.** We introduce the binary multiplication symbol $\times \in \mathcal{F}_{\mathsf{f}}^{(2)}$ with two axioms about defining equalities of products.

$$\begin{aligned}
\forall x\,(x \times 0 \approx 0) &\qquad \text{multiplication by zero}\\
\forall x \forall y\,(x \times \mathsf{s}(y) \approx (x \times y) + x) &\qquad \text{multiplication by non-zero}\\
\forall x_1 \forall x_2 \forall y_1 \forall y_2\,(x_1 \approx y_1 \wedge x_2 \approx y_2 \to x_1 \times y_1 \approx x_2 \times y_2) &\qquad \text{congruence of } \times
\end{aligned}$$

Theorem 3.30. *Peano Artihmetic (Guiseppe Peano, 1889), i.e. the first-order theory that extends Presburger Arithmetic with multiplication, is incomplete (Gödel) and undecidable.*

Theorem 3.31. *The axioms of Peano Artihmetic appear consistent (Gentzen, 1936).*

Lemma 3.32 (ACN)**.** *Addition and Multiplication on natural numbers are associative, commutative, and determine neutral elements.*

$$\begin{aligned}
\forall x \forall y \forall z\,(x \circ (y \circ z) \approx (x \circ y) \circ z) &\qquad \text{associativity of } \circ \in \{+, \times\}\\
\forall x \forall y\,(x \circ y \approx y \circ x) &\qquad \text{commutativity of } \circ \in \{+, \times\}\\
\forall x\,(x + 0 \approx x \wedge 0 + x \approx x) &\qquad \text{neutral element for } +\\
\forall x\,(x \times \mathsf{s}(0) \approx x \wedge \mathsf{s}(0) \times x \approx x) &\qquad \text{neutral element for } \times
\end{aligned}$$

4 Automation

Tous les hommes sont mortels
Fosca est un homme

Fosca est mortel

In this chapter, we demonstrate refutation complete proving procedures. It seems natural to expect decision procedures for decidable fragments of first order logic (Section 3.3) and decidable first order theories (Section 3.4), but we will see in simple examples that decision procedures automatically fall out from refutational completeness.

We know by Herbrand's theorem that each satisfiable set of (non-ground) clauses and each finite set of ground instances of a satisfiable set of (non-ground) clauses has a Hebrand model. And we know by compactness that if every finite subset of a set of clauses is satisfiable then this set is satisfiable. The satisfiability of ground instances is decidable as we have seen in the previous chapter. So the central idea of instantiation-based first order theorem proving is to find an unsatisfiable and finite set of ground instances for a given set of clauses. In general a failure in this search does not show satisfiability of the given set of clauses. In practice we make many detours in the search and we experience very finite resources of space and time, while in general there is no bound on the size of a smallest set of unsatisfiable ground instances.

To actually prove a theorem we first make use of the fact that a first-order formula is valid if and only if its negation is unsatisfiable. Second we efficiently (Tseitin's transformation [14], ...) transform the negated formula into an *equisatisfiable* set of clauses, i.e. the formula is satisfiable if and only if the set of clauses is satisfiable.

It would sufficient to just luckily guess an unsatisfiable set of ground instances. But usually instantiation based automated provers generate a sequence of growing sets ground instances such that an unsatisfiable one will be found for an arbitrary unsatisfiable set of clauses eventually.

First we translate axioms and lemmata into clausal normal form in Sections 4.1, then we look at Gilmore's Prover from 1960 in Section 4.2. After that we look at more modern calculi for refutation based first order theorem proving without equality in Section 4.3 and with equality in Section 4.4.

4.1 Theory axioms in `CNF`

In the previous chapter we expressed axioms and lemmas of first order theories in `FOF` syntax. Since Gilmore's prover, resolution and `Inst-Gen` work on sets of clauses we transform those axioms into (at least) equi-satisfiable representations in `CNF` syntax as

summarized for equivalence, congruence, natural numbers, and induction in Table 4.1, for addition and multiplication in Table 4.2.

$x \approx x,\ x \not\approx y \vee y \approx x,\ x \not\approx y \vee y \not\approx z \vee x \approx z$	equivalence
$x \not\approx y \vee \mathsf{s}(x) \approx \mathsf{s}(y)$	congruence of s
$\mathsf{s}(x) \not\approx 0,\ \mathsf{s}(x) \not\approx \mathsf{s}(y) \vee x \approx y$	natural numbers
$\neg G(0) \vee \boxed{G(\mathsf{c}_G)} \vee G(x),\ \neg G(0) \vee \boxed{\neg G(\mathsf{s}(\mathsf{c}_G))} \vee G(x)$	induction schema

Table 4.1: The theory of natural numbers in CNF

$x_1 \not\approx y_1 \vee x_2 \not\approx y_2 \vee x_1 + y_1 \approx x_2 + y_2$	congruence of $+$
$x + 0 \approx x,\ x + \mathsf{s}(x) \approx \mathsf{s}(x + y)$	addition
$x_1 \not\approx y_1 \vee x_2 \not\approx y_2 \vee x_1 \times y_1 \approx x_2 \times y_2$	congruence of $\times$
$x \times 0 \approx 0,\ x \times \mathsf{s}(y) \approx (x \times y) + x$	multiplication

Table 4.2: Addition and multiplication in CNF

Example 4.1. For the formula $G(x) = \mathsf{s}(x) \not\approx x$ we state the induction axioms in CNF in the theory of natural numbers. We had to introduce a fresh constant c_s in this equivalence transformation process.

$$\mathsf{s}(0) \approx 0 \vee \boxed{\mathsf{s}(\mathsf{c}_\mathsf{s}) \not\approx \mathsf{c}_\mathsf{s}} \vee \mathsf{s}(x) \not\approx x$$
$$\mathsf{s}(0) \approx 0 \vee \boxed{\mathsf{s}(\mathsf{s}(\mathsf{c}_\mathsf{s})) \approx \mathsf{s}(\mathsf{c}_\mathsf{s})} \vee \mathsf{s}(x) \not\approx x$$

4.2 Gilmore's Prover

In 1960 Paul Gilmore presented a first *implementation* of an automated theorem prover [6] for first order logic (without equality), which happened to use an instantiation-based approach. The procedure is complete, i.e. for every valid formula a refutation proof can be found eventually.

In practice this prover ran into memory issues or time outs more often than not. We will discuss reasons for this inefficiency after we have described and demonstrated the procedure.

First the negation of a sentence F has to be transformed into an equisatisfiable set of clauses. Then the prover's procedure creates a sequence of finite sets of ground instances S_k for the set of clauses $S \approx \neg F$ to prove the validity of a formula F. Each set S_k contains all possible ground instances of S where all variables are substituted by elements

of H_k from definition 2.22 of the Hebrand universe. Each S_k is then transformed into a disjunctive normal form where satisfiability is obvious. The procedure is aborted when an unsatisfiable S_k is encountered.

Procedure 1 (Gilmore's Prover)**.** We translate the negation of our formula F into an equisatisfiable set of clauses $\neg F \approx S = \bigcup_{i=1}^{n} \mathcal{C}_i$ with an efficient algorithm [14], [13]. Then we start our first iteration with $k = 0$.

1. We create the set of all ground terms up to term depth k, i.e. the partial Herbrand universe H_k according to Definition 2.22. We use H_k to create the set of clause instances S_k by substituting all variables in each clause by terms from H_k in any possible permutation.

$$S_k = \bigcup_{i=1}^{n} \{\, \mathcal{C}_i\sigma \mid \mathcal{C}_i \in S,\, \sigma : \mathcal{V} \to H_k \,\}$$

2. We translate S_k into an equivalent disjunctive normal form (i.e. a disjunction of conjunctions of literals) where satisfiability is easily checked.

3. When every conjunction contains a pair of complementary literals then we exit the procedure and report unsatisfiability of S, hence validity of F.

   Otherwise we increase k by one and continue with step 1.

Gilmore's procedure will eventually terminate for an unsatisfiable set of clauses. It enumerates all possible sets of ground instances iteratively and one of them must be unsatisfiable for an unsatisfiable set of clauses. However the number of iterations has no general upper bound. Otherwise it would be a decision procedure for satisfiability in first order logic which does not exist because of undecidability of satisfiability in first order logic.

Lemma 4.2. *Gilmore's procedure is a decison procedure for monadic first order logic (Examples 3.14 and 4.3) and the Schönfinkel-Bernays fragment (see Table 3.4) of First Order Logic.*

Proof. In the absence of non-constant function symbols the set $H'_{i+1} = \emptyset$ is empty. The procedure can stop after the first iteration because $H_i = H_0$ and $S_i = S_0$ for all $i \geq 0$, i.e. after the first iteration no new terms are added to the Hebrand model and no new ground instances can be generated. □

Following Gilmore's prover we can easily prove the syllogism from above.

Example 4.3. First we translate the syllogism into a formula F in first order logic.

$$F = A \to (B \to C) \equiv \neg(A \wedge B) \vee C \equiv (A \wedge B) \to C \qquad \text{formula}$$

$$A = \forall x\, (\mathsf{human}(x) \to \mathsf{mortal}(x) \qquad \text{theory}$$

$$B = \mathsf{human}(\mathsf{fosca}) \qquad \text{fact}$$

$$C = \mathsf{mortal}(\mathsf{fosca}) \qquad \text{conjecture}$$

Then we negate the formula to clausal normal form $S_{(4.3)} = A \wedge B \wedge \neg C \equiv \neg F$. Since there is exactly one constant we get $H_0 = \{\mathsf{fosca}\}$ and $S_0 = \{(\neg\mathsf{human}(\mathsf{fosca}) \vee \mathsf{mortal}(\mathsf{fosca})) \wedge \mathsf{human}(\mathsf{fosca}) \wedge \neg\mathsf{mortal}(\mathsf{fosca})\}$ in our first iteration. As last step we transform the single formula in the set of ground instances S_0 into a disjunctive normal form for easy satisfiability checking.

$$\begin{gathered}(\neg\mathsf{human}(\mathsf{fosca}) \wedge \mathsf{human}(\mathsf{fosca}) \wedge \neg\mathsf{mortal}(\mathsf{fosca})) \\ \vee \\ (\mathsf{mortal}(\mathsf{fosca}) \wedge \mathsf{human}(\mathsf{fosca}) \wedge \neg\mathsf{mortal}(\mathsf{fosca}))\end{gathered}$$

Both conjunctions contain complementary literals and we conclude the negated formula is unsatisfiable and the given syllogism holds.

Example 4.4. Let $k \in \mathbb{N}$ be an arbitrary but fixed number. Consider the unsatisfiable set of clauses $S_{(4.4)} = \{\, \neg\mathcal{L}_1,\ \mathcal{L}_2 \,\} = \{\, \neg\mathsf{E}(x, \mathsf{s}(x))),\ \mathsf{E}(\mathsf{s}^k(y), \mathsf{s}(\mathsf{s}^k(y))) \,\}$. The sets of instances S_{i+1} are satisfiable for all $i + 1 < k$. The set of instances S_k is clearly unsatisfiable.

$$\begin{aligned} H'_0 &= \{\, \mathsf{z} \,\} & S_0 &= \{\, \neg\mathsf{E}(\mathsf{z}, \mathsf{s}(\mathsf{z})),\ \mathsf{E}(\mathsf{s}^k(\mathsf{z}),\ \mathsf{s}(\mathsf{s}^k(\mathsf{z}))) \,\} \subsetneq S_k \\ H'_{i+1} &= \{\, \mathsf{s}(\mathsf{s}^i(\mathsf{z})) \,\} & S_{i+1} &\supsetneq \{\, \neg\mathsf{E}(\mathsf{s}^{i+1}(\mathsf{z}), \mathsf{s}(\mathsf{s}^{i+1}(\mathsf{z}))),\ \mathsf{E}(\mathsf{s}^k(\mathsf{s}^{i+1}(\mathsf{z})), \mathsf{s}(\mathsf{s}^k(\mathsf{s}^{i+1}(\mathsf{z})))) \,\} \\ H'_k &= \{\, \mathsf{s}^k(\mathsf{z}) \,\} & S_k &\supsetneq \{\, \neg\mathsf{E}(\mathsf{s}^k(\mathsf{z}), \mathsf{s}(\mathsf{s}^k(\mathsf{z}))),\ \mathsf{E}(\mathsf{s}^k(\mathsf{s}^k(\mathsf{z})), \mathsf{s}(\mathsf{s}^k(\mathsf{s}^k(\mathsf{z})))) \,\} \end{aligned}$$

We've produced $2 \cdot k$ *irrelevant* instances, i.e. these clauses did not cause any conflict in propositional satisfiability. In this example the guess for a finite unsatisfiable set of ground instances appears feasible and yields a smaller set.

$$\{\, \neg\mathcal{L}_1\sigma,\ \mathcal{L}_2\sigma \,\} \qquad \sigma = \{x \mapsto \mathsf{s}^k(\mathsf{z}),\ y \mapsto \mathsf{z}\}$$

Example 4.5. Consider the satisfiable set of clauses $S_{4.5} = \{\neg\mathsf{E}(\mathsf{z}, \mathsf{s}(x)\}$. This set is clearly in the decidable Ackermann fragment of first order logic. But the procedure yields an infinite sequence of distinct and satisfiable sets $S_{k\geq 0}$:

$$\begin{aligned} H'_0 &:= \{\mathsf{z}\} & S_0 &:= \{\neg\mathsf{E}(\mathsf{z}, \mathsf{s}(\mathsf{z}))\} && \text{satisfiable} \\ H'_{i+1} &:= \{\mathsf{s}(\mathsf{s}^i(\mathsf{z}))\} & S_{i+1} &:= S_i \uplus \{\neg\mathsf{E}(\mathsf{z}, \mathsf{s}(\mathsf{s}^i(\mathsf{z})))\} && \text{satisfiable} \end{aligned}$$

So Gilmore's prover wouldn't terminate on this simple and decidable problem.

We have observed three main disadvantages in Gilmore's procedure.

1. The generation of instances is unguided. With each iteration exponentially many (mostly useless) instances are created – depending on the number and the arities of used symbols.

$$\begin{aligned} |S_i| &= \sum_n \left(|\mathcal{F}_{\mathsf{P}}^{(n)}| \cdot |H_i|^n\right) & |H_0| &\geq 1 \\ |S_{i+1}| &= \sum_n \left(|\mathcal{F}_{\mathsf{P}}^{(n)}| \cdot |H_{i+1}|^n\right) & |H_{i+1}| &\geq \sum_{n>0} \left(|\mathcal{F}_{\mathsf{f}}^{(n)}| \cdot |H_i|^n\right) \end{aligned}$$

   This makes disadvantage No. 2 which is invoked at every iteration even worse.

2. The check for unsatisfiability is far from efficient. The transformation from a set of clauses to a formula in disjunctive normal form[1] usually introduces an exponential[2] blow up in the size of the formula – depending on the number of clauses n in the set and the number of literals c_i per clause $\mathcal{C}_i$ we get the disjunction of $\prod_1^n c_i$ conjunctions of n literals.

$$\bigwedge_{i=1}^{n}\left(\bigvee_{j_i=1}^{c_i} p_{(i,j_i)}\right) \equiv \bigvee_{(j_1,\ldots,j_n)}\left(\bigwedge_{i=1}^{n} p_{(i,j_i)}\right) \quad \text{with } (j_1,\ldots,j_n) \in \prod_{i=1}^{n}\{1,\ldots,c_i\}$$

In total the number of literals in the set of clauses is $n \cdot \bar{c}_{arith}$, while the equivalent disjunctive normal form contains $(\bar{c}_{geom})^n \cdot n$ literals[3].

$$\{1\} \times \{1,2\} \times \{1,2,3\} = \{(1,1,1),(1,1,2),(1,1,3),(1,2,1),(1,2,2),(1,2,3)\}$$

3. The procedure will not terminate for satisfiable sets when at least one non-constant predicate symbol is used in the set of clauses and one non-constant function symbol is available, e.g. for $S = \{\, \mathsf{P}(\mathsf{f}(x)) \,\}$ we get

$$\begin{aligned} H_0 &= \{\, \mathsf{c} \,\} & S_0 &= \{\, \mathsf{P}(\mathsf{f}(\mathsf{c})) \,\} \\ H_{i+1} &= \bigcup_{k=0}^{i+1} \{\, \mathsf{f}^k(\mathsf{c})) \,\} & S_{i+1} &= \{\, \mathsf{P}(\mathsf{f}(t)) \mid t \in H_{i+1} \,\} \\ \mathsf{f}^{i+1}(\mathsf{c}) &\in H_{i+1} \setminus H_i & \mathsf{P}(\mathsf{f}(\mathsf{f}^{i+1}(\mathsf{c}))) &\in S_{i+1} \setminus S_i \end{aligned}$$

Issue 2 was already implicitly addressed in 1960 [5] (which also incorporated the basic idea of resolution – on ground instances of terms) and refined in 1962 [4] by Davis, Putnam, Longeman, and Loveland, which was the starting point for the development of efficient propositionally satisfiability checkers, i.e. efficient modern SAT solvers.

4.3 Proving without Equality

In this section we introduce the equality symbol $\approx$ into our formulae, but we treat it not different from an arbitrary binary predicate symbol with infix notation.

4.3.1 Resolution

Definition 4.6 (Resolution calculus)**.** Let A, B be atoms and $\mathcal{C}, \mathcal{D}$ be clauses.

$$\frac{A \vee \mathcal{C} \quad \neg B \vee \mathcal{D}}{(\mathcal{C} \vee \mathcal{D})\sigma}\ \text{Resolution} \qquad \frac{A \vee B \vee \mathcal{C}}{(A \vee \mathcal{C})\sigma}\ \text{Factoring}$$

[1] In contrast the linear Tseitin transformation yields an equisatisfiable conjunctive normal form.

[2] The existence of a polynomial algorithm for the transformation of an arbitrary propositional formula into an equisatisfiable formula in *disjunctive normal form* (where satisfiability is a linear check) would show that SAT in $\mathcal{P}$ and $\mathcal{P} = \mathcal{NP}$, which is still unknown.

[3] Geometric mean $\bar{c}_{geom} := \left(\prod_1^n c_i\right)^{\frac{1}{n}}$, arithmetic mean $\bar{c}_{arith} := \left(\sum_1^n c_i\right) \cdot \frac{1}{n}$, and $\bar{c}_{geom} \leq \bar{c}_{arith}$.

$$\text{where } \sigma = \text{mgu}(A, B)$$

Example 4.7. Modus tollens is a special case of resolution.

$$\frac{F \to G \quad \neg G}{\neg F}\ \text{modus tollens} \qquad \frac{\neg F \vee G \quad \neg G}{\neg F}$$

Example 4.8. We easily infer the empty clause and conclude unsatisfiability of the set of clauses $S_{4.8} = \{\, \mathsf{s}(x) \not\approx x,\ \mathsf{s}(\mathsf{s}^k(y)) \approx \mathsf{s}^k(y) \,\}$.

$$\frac{\mathsf{s}(x) \not\approx x \quad \mathsf{s}(\mathsf{s}^k(y)) \approx \mathsf{s}^k(y)}{\square}\ \{x \mapsto \mathsf{s}^k(y)\}$$

Example 4.9. With observe an infinite sequence of resolution steps from satisfiable set $S_{4.9} = \{\, \mathsf{s}(x) \not\approx x,\ \mathsf{s}(y) \not\approx \mathsf{s}(z) \vee y \approx z \,\}$.

$$\frac{\mathsf{s}^{i+1}(x) \not\approx \mathsf{s}^i(x) \quad \mathsf{s}(x') \not\approx \mathsf{s}(y') \vee x' \approx y'}{\mathsf{s}^{i+2}(x) \not\approx \mathsf{s}^{i+1}(x)}\ \{x' \mapsto \mathsf{s}^{i+1}(x), y' \mapsto \mathsf{s}^i(x)\} \qquad (i \geq 0)$$

We can deduce or observe disadvantages in resolution.

1. If clauses contain more than two literals the resolution inference rule yields clauses with more literals than the sources.

2. We have to check all pairings of all positive literals with all negative literals for clashing. $n_{\mathcal{C}\vee\mathcal{D}} = n_{(A\vee\mathcal{C})} - 1 + n_{(\neg B\vee\mathcal{D})} - 1$, which was addressed in Ordered Resolution.

4.3.2 Ordered resolution

Lemma 4.10. *A well-founded and total order on general ground terms always exists.*

Definition 4.11 (Order on literals)**.** We extend a well-founded and total order $\succ$ on general ground terms, i.e general atoms to a well-founded proper order $\succ_{\mathsf{L}}$ on literals such that for all atoms A and B with $A \succ B$ the relations $A \succ_{\mathsf{L}} B$, $\neg A \succ_{\mathsf{L}} \neg B$ and $\neg A \succ_{\mathsf{L}} A$ hold. A (non-ground) literal L is *(strictly) maximal* if there exists a ground substitution τ such for no other literal L' the relation $L'\tau \succ L\tau$ (strictly: $\succcurlyeq$) holds. We write $\succ_{gr}$ to suggest the existence of such a ground substitution τ.

Definition 4.12 (Ordered Resolution)**.** Let A, B be atoms and $\mathcal{C}, \mathcal{D}$ be clauses.

$$\frac{A \vee \mathcal{C} \quad \neg B \vee \mathcal{D}}{(\mathcal{C} \vee \mathcal{D})\sigma}\ \begin{matrix}\text{Ordered}\\ \text{Resolution}\end{matrix} \qquad \frac{A \vee B \vee \mathcal{C}}{(A \vee \mathcal{C})\sigma}\ \begin{matrix}\text{Ordered}\\ \text{Factoring}\end{matrix}$$

where $\sigma = \text{mgu}(A, B)$, $A\sigma$ is strictly maximal in $\mathcal{C}\sigma$, $\neg B\sigma$ is maximal in $\mathcal{D}\sigma$.

Example 4.13. With an ordering such that $\mathsf{s}(y) \approx \mathsf{s}(z)) \succ y \approx z$ on atoms and $\neg A \succ A$ the satisfiable set $S_{4.9}$ saturates with ordered resolution, because the strictly maximal literals $\mathsf{s}(x) \not\approx x$ and $\mathsf{s}(y) \not\approx \mathsf{s}(z)$ do not clash and the ordered resolution rule is not applicable.

Definition 4.14 (Order on clauses)**.** multiset order

Lemma 4.15. *Ordered resolution is refutation complete.*

Proof. If no new resolvant can be derived and the empty clause is not in the set of clauses we have a model.

The resolvant $(\mathcal{C} \vee \mathcal{D})\sigma$ is smaller than $(A \vee \mathcal{C})\sigma$ and $(\neg B \vee \mathcal{D})\sigma$

$\neg B \succ_{\mathsf{L}} A$ because $A\sigma = B\sigma = C$ and $\neg X \succ_{\mathsf{L}} X$. □

4.3.3 InstGen

Gilmore's prover blindly constructs new ground instances.

Definition 4.16 (`Inst-Gen`)**.** Let A, B be atoms and $\mathcal{C}, \mathcal{D}$ be clauses.

$$\frac{A \vee \mathcal{C} \qquad \neg B \vee \mathcal{D}}{(A \vee \mathcal{C})\sigma \qquad (\neg B \vee \mathcal{D})\sigma}\ \texttt{Inst-Gen}$$

$$\sigma = \mathrm{mgu}(A, B)$$

Example 4.17. With `Inst-Gen` we immediately can derive a helpful clause from set $S_{(4.4)} = \{\, \neg\mathsf{E}(x, \mathsf{s}(x))),\ \mathsf{E}(\mathsf{s}^k(y), \mathsf{s}(\mathsf{s}^k(y)))\,\}$ introduced in Example 4.4.

$$\frac{\neg\mathsf{E}(x, \mathsf{s}(x))) \quad \mathsf{E}(\mathsf{s}^k(y), \mathsf{s}(\mathsf{s}^k(y)))}{\neg\mathsf{E}(\mathsf{s}^k(y), \mathsf{s}(\mathsf{s}^k(y)))) \qquad \mathsf{E}(\mathsf{s}^k(y), \mathsf{s}(\mathsf{s}^k(y)))}\ \{x \mapsto \mathsf{s}^k(y)\}$$

and we conclude unsatisfiability because of propositional unsatisfiability of

$$\{\, \mathsf{E}(\mathsf{s}^k(\mathsf{c}_\bot), \mathsf{s}(\mathsf{s}^k(\mathsf{c}_\bot))), \neg\mathsf{E}(\mathsf{s}^k(\mathsf{c}_\bot), \mathsf{s}(\mathsf{s}^k(\mathsf{c}_\bot)))) \,\}$$

Example 4.18. With `Inst-Gen` we cannot derive any new clause from set $S_{(4.5)} = \{\, \mathsf{E}(\mathsf{z}, y)\,\}$ introduced in Example 4.5 and we conclude satisfiability of the `Inst-Gen`-saturated set $S_{(4.5)}$ because of the propositional satisfiability of $S_{(4.5)}\sigma_\bot$.

Lemma 4.19. *The set of clauses $S_0 = S \cup \{A \vee \mathcal{C}, \neg B \vee \mathcal{D}\}$ is satisfiable if and only if the derived set of clauses $S_1 = S_0 \cup \{(A \vee \mathcal{C})\sigma, (\neg B \vee \mathcal{D})\sigma\}$ with $\sigma = \mathrm{mgu}(A, B)$ is satisfiable.*

Proof. If S_1 is satisfiable then there exists an interpretation that satisfies all clauses in S_1. The same interpretation models all clauses in S_0 because $S_0 \subseteq S_1$. In reverse S_1 cannot be satisfiable if S_0 is not.

□

Procedure 2 (Inst-Gen-Loop)**.** As in Gilmore's prover (Procedure 1) we translate the negation of our formula F into an equisatisfiable set of clauses S_0. Then we introduce a distinct constant symbol $\mathsf{c}_\bot \notin \mathcal{F}(S_0)$ even when there are constant symbols in the signature. We start our first iteration with $k = 0$.

1. We construct a set $S_k\,\sigma_\bot$ of ground instances from S_k where instantiator $\sigma_\bot := \{x \mapsto \mathsf{c}_\bot \mid x \in \mathit{Vars}(S_k)\}$ substitutes all occurring variables with constant symbol $\mathsf{c}_\bot$.

2. We check the decidable satisfiability of $S_k\,\sigma_\bot$ with a modern `SAT` or `SMT`-solver.

   If $S_k\,\sigma_\bot$ is unsatisfiable then we exit the procedure and report *usatisfiability* of S, i.e the original formula F is valid.

3. The set $S_k\,\sigma_\bot$ is satisfiable, hence we can retrieve a model $\mathcal{M}_k \models S_k\sigma_\bot$. We select one literal $L_i = \mathrm{sel}(\mathcal{C}_i)$ per clause $\mathcal{C}_i \in S_k$ such that the each grounded selected literal holds in model $\mathcal{M}_k \models L_i\sigma_\bot$ for all $i \leq |S_k|$.

4. We search for pairs of selected literals $(A, \neg B) = (L_i, L_j^c)$ such that the most general unifier $\tau = \mathrm{mgu}(A, B)$ exists.

5. We set $S_{k+1} ::= S_k$ and for each pair of clashing literals (L_i, L_j^c) we apply `Inst-Gen` to the originating clauses $\{\,\mathcal{C}_i, \mathcal{C}_j\,\} = \{\,L_i \vee \mathcal{C}, L_j \vee \mathcal{D}\,\}$ to add new (not necessarily ground) instances to S_{k+1}.

   If no new clauses were added, i.e. $S_{k+1} = S_k$ after all pairs were processed we exit the procedure and report *satisfiability* of S, i.e. the original formula F is not valid.

6. We increase k by 1 and continue with step 1.

Example 4.20. The selected literals of the first and the second clause change between iterations.

$$
\begin{aligned}
S_0 &= \{\,\mathsf{P}(\mathsf{a}) \vee \mathsf{Q}(\mathsf{a}), \mathsf{P}(\mathsf{a}) \vee \neg\mathsf{Q}(y), \neg\mathsf{P}(x)\,\} \\
S_0\sigma_\bot &= \{\,\mathsf{P}(\mathsf{a}) \vee \mathsf{Q}(\mathsf{a}), \mathsf{P}(\mathsf{a}) \vee \neg\mathsf{Q}(\mathsf{c}_\bot), \neg\mathsf{P}(\mathsf{c}_\bot)\,\} && \text{(satisfiable)} \\
&\qquad \frac{\mathsf{P}(\mathsf{a}) \vee \mathsf{Q}(*) \quad \neg\mathsf{P}(x)}{\mathsf{P}(\mathsf{a}) \vee \mathsf{Q}(*) \quad \neg\mathsf{P}(\mathsf{a})}\; x \mapsto \mathsf{a} && * \in \{\mathsf{a}, y\} \\
S_1 &= \{\,\mathsf{P}(\mathsf{a}) \vee \mathsf{Q}(\mathsf{a}), \mathsf{P}(\mathsf{a}) \vee \neg\mathsf{Q}(y), \neg\mathsf{P}(x), \neg\mathsf{P}(\mathsf{a})\,\} \\
S_1\sigma_\bot &= \{\,\mathsf{P}(\mathsf{a}) \vee \mathsf{Q}(\mathsf{a}), \mathsf{P}(\mathsf{a}) \vee \neg\mathsf{Q}(\mathsf{c}_\bot), \neg\mathsf{P}(\mathsf{c}_\bot), \neg\mathsf{P}(\mathsf{a})\,\} && \text{(satisfiable)} \\
&\qquad \frac{\mathsf{P}(\mathsf{a}) \vee \mathsf{Q}(\mathsf{a}) \quad \mathsf{P}(\mathsf{a}) \vee \neg\mathsf{Q}(y)}{\mathsf{P}(\mathsf{a}) \vee \mathsf{Q}(\mathsf{a}) \quad \mathsf{P}(\mathsf{a}) \vee \neg\mathsf{Q}(\mathsf{a})}\; y \mapsto \mathsf{a} \\
S_2 &= \{\,\mathsf{P}(\mathsf{a}) \vee \mathsf{Q}(\mathsf{a}), \mathsf{P}(\mathsf{a}) \vee \neg\mathsf{Q}(y), \neg\mathsf{P}(x), \neg\mathsf{P}(\mathsf{a}), \mathsf{P}(\mathsf{a}) \vee \neg\mathsf{Q}(\mathsf{a})\,\} \\
S_2\sigma_\bot &= \{\,\mathsf{P}(\mathsf{a}) \vee \mathsf{Q}(\mathsf{a}), \mathsf{P}(\mathsf{a}) \vee \neg\mathsf{Q}(\mathsf{c}_\bot), \neg\mathsf{P}(\mathsf{c}_\bot), \neg\mathsf{P}(\mathsf{a}), \mathsf{P}(\mathsf{a}) \vee \neg\mathsf{Q}(\mathsf{a})\,\} && \text{(unsatisfiable)}
\end{aligned}
$$

Lemma 4.21. *The $\tau = \mathrm{mgu}(A, B)$ in Procedure 2, step 4 is a proper instantiator, i.e. it is not a variable renaming.*

Proof. Assume τ in Procedure 2 is a renaming, then we have $A\tau\sigma_\perp = A\sigma_\perp$, $B\tau\sigma_\perp = B\sigma_\perp$, and by definition of the most general unifier $A\tau = B\tau$. Hence $A\sigma_\perp = B\sigma_\perp$ which contradicts that $M_k \models A\sigma_\perp, \neg B\sigma_\perp$ by definition of step 3. Hence the assumption is false and τ must be a proper instantiator. □

Example 4.22. Let $S_0 = S_{(4.4)}$ be the set of unsatisfiable clauses from Example 4.4. Then the initial set of ground instances $S_0\sigma_\perp = \{\neg\mathsf{E}(\mathsf{c}_\perp, \mathsf{s}(\mathsf{c}_\perp))), \mathsf{E}(\mathsf{s}^k(\mathsf{c}_\perp), \mathsf{s}(\mathsf{s}^k(\mathsf{c}_\perp)))\}$ is satisfiable with domain $A = \{\mathsf{c}_\perp, \mathsf{s}(\mathsf{c}_\perp), \mathsf{s}^k(\mathsf{c}_\perp), \mathsf{s}(\mathsf{s}^k(\mathsf{c}_\perp))\}$ and predicate interpretation $\mathsf{E}^{\mathcal{I}} = \{(\mathsf{s}^k(\mathsf{c}_\perp), \mathsf{s}(\mathsf{s}^k(\mathsf{c}_\perp)))\} \subseteq A^2$. With just two unit clauses we easily find the only pair of clashing literals and compute the unifier $\tau = \{x \mapsto \mathsf{s}(s^k(y))\}$. By application of `Inst-Gen` we construct our next set of clauses $S_1 = S_0 \uplus \{\neg\mathsf{E}(\mathsf{s}^k(y), \mathsf{s}(\mathsf{s}^k(y)))\}$ and get an unsatisfiable set of ground instances $S_1\sigma_\perp$.

4.4 Proving with Equality

So far we have treated the equality symbol like any other binary predicate symbol, which can yield models where $\mathsf{a} \not\approx \mathsf{a}$ holds. Understandably, we are only interested in normal models or at least in models that implies the existence of a normal model. We have allready seen that a normal Herbrand model might not exist, but we can ensure that we find only desired models.

4.4.1 Adding equality axioms

Theorem 4.23. *[7] Any set of clauses (a formula) has a* normal *model if and only if it has a model that satisfies the* equality axioms*, i.e. reflexivity, symmetry, transitivity, and congruence for all function symbols* $\mathsf{f} \in \mathcal{F}_\mathsf{f}$ *and all predicate symbols* $\mathsf{P} \in \mathcal{F}_\mathsf{P}$*.*

Example 4.24 (Ordered resolution with equality axioms)**.** We add the equality axioms to a small set of clauses $S = \{\mathsf{s}(x) \not\approx 0,\ \mathsf{s}(x) \not\approx \mathsf{s}(y) \vee x \approx y\}$ and mark maximal literals.

$$
\begin{aligned}
x \approx x,\ x \approx y \vee y \not\approx x,\ x \approx z \vee x \not\approx y \vee y \not\approx z \quad & \approx\text{-equivalence} \\
\mathsf{s}(x) \approx \mathsf{s}(y) \vee x \not\approx y \quad & \mathsf{s}\text{-congruence} \\
\mathsf{s}(x) \not\approx 0,\ \mathsf{s}(x) \not\approx \mathsf{s}(y) \vee x \approx y \quad & S
\end{aligned}
$$

With ordered resolution we cannot infer new clauses from clauses in S. But we can apply

rules of ordered resolution to pairs of equality axioms and clauses.

$$\frac{x \approx x \quad x' \approx y' \vee y' \not\approx x'}{x \approx x} \{x' \mapsto x, y' \mapsto x\} \qquad R, S \vdash R$$

$$\frac{x \approx x \quad x' \approx z' \vee x' \not\approx y' \vee y' \not\approx z'}{x \approx z' \vee x \not\approx z'} \{x' \mapsto x, y' \mapsto x\} \qquad R, T \vdash \mathsf{true}$$

$$\frac{x \approx x \quad x' \approx z' \vee x' \not\approx y' \vee y' \not\approx z'}{x' \approx x \vee x' \not\approx x} \{y' \mapsto x, z' \mapsto x\} \qquad R, T \vdash \mathsf{true}$$

$$\frac{x \approx y \vee y \not\approx x \quad \mathsf{s}(x') \approx \mathsf{s}(y') \vee x' \not\approx y'}{\mathsf{s}(y') \approx \mathsf{s}(x') \vee x' \not\approx y'} \{y \mapsto \mathsf{s}(x'), x \mapsto \mathsf{s}(y')\} \qquad S, C \vdash C_S$$

$$\frac{x \approx z \vee x \not\approx y \vee y \not\approx z \quad \mathsf{s}(x') \approx \mathsf{s}(y') \vee x' \not\approx y'}{\mathsf{s}(x') \approx z \vee \mathsf{s}(y') \not\approx z \vee x' \not\approx y'} \{x \mapsto \mathsf{s}(x'), y \mapsto \mathsf{s}(y')\} \qquad T, C \vdash$$

$$\frac{x \approx z \vee x \not\approx y \vee y \not\approx z \quad \mathsf{s}(x') \approx \mathsf{s}(y') \vee x' \not\approx y'}{x \approx \mathsf{s}(y') \vee x \not\approx \mathsf{s}(x') \vee x' \not\approx y'} \{y \mapsto \mathsf{s}(x'), z \mapsto \mathsf{s}(y')\} \qquad T, C \vdash$$

$$\frac{\mathsf{s}(x) \approx \mathsf{s}(y) \vee x \not\approx y \quad x' \approx y' \vee \mathsf{s}(x') \not\approx \mathsf{s}(y')}{x \not\approx y \vee x \approx y} \{x' \mapsto x, y' \mapsto y\} \qquad C, I \vdash \mathsf{true}$$

Example 4.25. We extend our set with clause $\mathsf{s}(\mathsf{s}(x)) \approx \mathsf{s}(0)$ that clashes with injectivity of s.

$$\frac{s(x') \not\approx 0 \qquad \dfrac{x' \approx y' \vee \mathsf{s}(x') \not\approx \mathsf{s}(y') \quad \mathsf{s}(\mathsf{s}(x)) \approx \mathsf{s}(0)}{\mathsf{s}(x) \approx 0} \{x' \mapsto x\}}{\square} \{x' \mapsto \mathsf{s}(x), y' \mapsto 0\}$$

Example 4.26 (`Inst-Gen` with equality axioms)**.** The default grounding for `Inst-Gen` substitutes *all* variables with one constant function symbol. We notice that the selection process is unfortunate, because the selected *positive* literals of each axiom but congruence clash with $\mathsf{s}(x) \not\approx 0$.

$$\mathsf{c}_\perp \approx \mathsf{c}_\perp \qquad \text{reflexivity}$$

$$\mathsf{c}_\perp \approx \mathsf{c}_\perp \vee \mathsf{c}_\perp \not\approx \mathsf{c}_\perp \qquad \text{symmetry}$$

$$\mathsf{c}_\perp \approx \mathsf{c}_\perp \vee \mathsf{c}_\perp \not\approx \mathsf{c}_\perp \vee \mathsf{c}_\perp \not\approx \mathsf{c}_\perp \qquad \text{transitivity}$$

$$\mathsf{s}(\mathsf{c}_\perp) \approx \mathsf{s}(\mathsf{c}_\perp) \vee \mathsf{c}_\perp \not\approx \mathsf{c}_\perp \qquad \text{congruence}$$

$$\mathsf{c}_\perp \approx \mathsf{c}_\perp \vee \mathsf{s}(\mathsf{c}_\perp) \not\approx \mathsf{s}(\mathsf{c}_\perp) \qquad \text{injectivity}$$

$$\mathsf{s}(\mathsf{c}_\perp) \not\approx 0$$

$$\frac{0 \not\approx \mathsf{s}(x') \quad x \approx y \vee y \not\approx x}{0 \approx \mathsf{s}(x') \vee \mathsf{s}(x') \approx 0} x \mapsto 0, y \mapsto \mathsf{s}(x')$$

$$\frac{0 \not\approx \mathsf{s}(x') \quad x \approx y \vee \mathsf{s}(x) \not\approx \mathsf{s}(y)}{0 \approx \mathsf{s}(x') \vee \mathsf{s}(x') \approx 0} x \mapsto 0, y \mapsto \mathsf{s}(x')$$

$$\frac{\mathsf{s}(x') \not\approx \mathsf{s}(y') \vee x' \approx y' \quad \mathsf{s}(x) \not\approx x}{\boxed{\mathsf{s}(\mathsf{s}(x)) \not\approx \mathsf{s}(x)} \vee \mathsf{s}(x) \approx x} \{x' \mapsto \mathsf{s}(x), y' \mapsto x\}$$

$$\frac{\mathsf{s}(x') \not\approx \mathsf{s}(y') \vee x' \approx y' \quad \boxed{\mathsf{s}^{i+2}(x) \not\approx \mathsf{s}^{i+1}(x)} \vee \mathsf{s}^{i+1}(x) \approx \mathsf{s}^{i}(x)}{\mathsf{s}(\mathsf{s}^{i+2}(x)) \not\approx \mathsf{s}(\mathsf{s}^{i+1}(x)) \vee \mathsf{s}^{i+2}(x) \approx \mathsf{s}^{i+1}(x)} \{x' \mapsto \mathsf{s}^{i+2}(x), y' \mapsto \mathsf{s}^{i+1}(x)\}$$

for all $i \geq 0$.

$$\begin{aligned}
\mathsf{c}_{x_0} &\approx \mathsf{c}_{x_0} && \approx\text{-reflexivity}\\
\mathsf{c}_{x_1} \approx \mathsf{c}_{y_1} &\vee \mathsf{c}_{y_1} \not\approx \mathsf{c}_{x_1} && \approx\text{-symmetry}\\
\mathsf{c}_{x_2} \approx \mathsf{c}_{z_2} &\vee \mathsf{c}_{x_2} \not\approx \mathsf{c}_{y_2} \vee \mathsf{c}_{y_2} \not\approx \mathsf{c}_{z_2} && \approx\text{-transitivity}\\
\mathsf{s}(\mathsf{c}_{x_3}) \approx \mathsf{s}(\mathsf{c}_{y_3}) &\vee \mathsf{c}_{x_3} \not\approx \mathsf{c}_{y_3} && \mathsf{s}\text{-congruence}\\
\mathsf{s}(\mathsf{c}_{x_4}) &\not\approx 0 && 0 \notin \mathrm{img}(\mathsf{s})\\
\mathsf{s}(\mathsf{c}_{x_5}) \not\approx \mathsf{s}(\mathsf{c}_{y_5}) &\vee \mathsf{c}_{x_5} \approx \mathsf{c}_{y_5} && \mathsf{s}\text{-injectivity}
\end{aligned}$$

4.4.2 Applying equality rules

Instead of adding equality axioms, complete calculi with additional derivation rules for the equality symbol were developed.

Definition 4.27. The *superposition calculus* includes

- ordered resolution (R), ordered factoring (F)

$$\frac{A \vee \mathcal{C} \quad \neg B \vee \mathcal{D}}{(\mathcal{C} \vee \mathcal{D})\sigma}\ (R) \qquad \frac{A \vee B' \vee \mathcal{C}}{(A \vee \mathcal{C})\sigma}\ (F)$$

  where $\sigma = \mathrm{mgu}(A, B)$ is defined, $A\sigma$ is strictly maximal in $\mathcal{C}\sigma$, $\neg B\sigma$ is maximal in $\mathcal{D}\sigma$, $A\sigma$ is maximal in $\mathcal{D}\sigma$;

- ordered paramodulation (P_-, P_+)

$$\frac{s \approx t \vee \mathcal{C} \quad \neg A[s'] \vee \mathcal{D}}{(\mathcal{C} \vee \neg A[t] \vee \mathcal{D})\,\sigma}\ (P_-) \qquad \frac{s \approx t \vee \mathcal{C} \quad A[s'] \vee \mathcal{D}}{(\mathcal{C} \vee A[t] \vee \mathcal{D})\,\sigma}\ (P_+)$$

  where $\sigma = \mathrm{mgu}(s, s')$ is defined, $(s \approx t)\sigma$ is strictly maximal in $\mathcal{C}\sigma$, $\neg A[s']$ is maximal in $\mathcal{D}\sigma$, $A[s']$ is strictly maximal in $\mathcal{D}\sigma$;

- superposition (S_-, S_+)

$$\frac{s \approx t \vee \mathcal{C} \quad u[s'] \not\approx v \vee \mathcal{D}}{(\mathcal{C} \vee u[t] \not\approx v \vee \mathcal{D})\,\sigma}\ (S_-) \qquad \frac{s \approx t \vee \mathcal{C} \quad u[s'] \approx v \vee \mathcal{D}}{(\mathcal{C} \vee u[t] \approx v \vee \mathcal{D})\,\sigma}\ (S_+)$$

  where $\sigma = \mathrm{mgu}(s, s')$ is defined, $s' \notin \mathcal{V}$, $t\sigma \not\succcurlyeq s\sigma$, $v\sigma \not\succcurlyeq u[s']\sigma$;

- equality resolution ($R_{\approx}$), and equality factoring ($F_{\approx}$)

$$\frac{s \not\approx s' \vee \mathcal{C}}{\mathcal{C}\sigma}\ (R_{\approx}) \qquad \frac{s \approx s' \vee u \approx v \vee \mathcal{C}}{(v \not\approx s' \vee u \approx s' \vee \mathcal{C})\sigma}\ (F_{\approx})$$

  where $\sigma = \mathrm{mgu}(s, s')$ is defined, $(s = s')\sigma \not\succ (u \approx v)$.

Example 4.28. With Superposition calculus no derivation rule is applicable to clauses of the set $S = \{\mathsf{s}(x) \not\approx 0, \mathsf{s}x \not\approx \mathsf{s}y \vee x \approx y\}$ because the maximal literals are both negations.

$$\mathsf{s}(x_1) \not\approx 0 \qquad \mathsf{s}(x_2) \not\approx \mathsf{s}(y_2) \vee x_2 \approx y_2$$

The saturated set does not contain the empty clause, hence we conclude it's satisfiability.

4.4.3 Inst-Gen-Eq

Definition 4.29. The *unit superposition calculus* includes

- unit paramodulation (UP)

$$\frac{s \approx t \quad L[s']}{(L[t])\,\sigma}\ (UP)$$

  where $\sigma = \mathrm{mgu}(s, s')$ is defined, $s' \notin \mathcal{V}$, $t\sigma \not\succ s\sigma$;

- unit superposition (US_-, US_+)

$$\frac{s \approx t \quad u[s'] \not\approx v}{(u[t] \not\approx v)\,\sigma}\ (US_-) \qquad \frac{s \approx t \quad u[s'] \approx v}{(u[t] \approx v)\,\sigma}\ (US_+)$$

  where $\sigma = \mathrm{mgu}(s, s')$ is defined, $s' \notin \mathcal{V}$, $t\sigma \not\succ s\sigma$, $v\sigma \not\succ u[s']\sigma$;

- unit equality resolution ($UR_{\approx}$), and unit resolution (UR)

$$\frac{s \not\approx t}{\square}\ (UR_{\approx}) \qquad \frac{A \quad \neg B}{\square}\ (UR)$$

  where s and t (A and B respectively) are unifiable.

At a first glance `Inst-Gen-Eq` is expected to behave similar to the application of Superposition. But actually it shares a disadvantage with `Inst-Gen` (see Example 4.26) as we can see in the following example.

Example 4.30. Let $S = \{\, \mathsf{s}(x_1) \not\approx \mathsf{s}(y_1) \vee x_1 \approx y_1, \mathsf{s}(x_2) \not\approx x_2 \,\}$. We start with $S_0 = S$, construct the `SMT`-encoding for $S_{0_\perp}$ and select one literal per clause from S_0 into L_1. The selection is unambiguous by any model. We easily derive the empty clause from the set of

selected literals L_0 by first applying unit superposition first and unit equality resolution afterwards.

$$
\begin{aligned}
S_0 &= \{\, \mathsf{s}(x_1) \not\approx \mathsf{s}(y_1) \vee x_1 \approx y_1,\, \mathsf{s}(x_2) \not\approx x_2 \,\} \\
S_{0_\bot} &= \{\, \mathsf{s}(\mathsf{c}_\bot) \not\approx \mathsf{s}(\mathsf{c}_\bot) \vee \mathsf{c}_\bot \approx \mathsf{c}_\bot,\, \mathsf{s}(\mathsf{c}_\bot) \not\approx \mathsf{c}_\bot \,\} \\
L_0 &= \{x_1 \approx y_1, \mathsf{s}(x_2) \not\approx x_2\}
\end{aligned}
$$

$$
\dfrac{\dfrac{x_1 \approx y_1 \quad [\mathsf{s}(x_2)] \not\approx x_2}{y_1 \not\approx x_2}\ \{y_1 \mapsto x_2\}}{\square}\ \sigma_1 = \{x_1 \mapsto \mathsf{s}(x_2)\}
$$

Since the clauses just contributed to the first step we instantiate $\mathcal{C}_3' = \mathcal{C}_1 \cdot \sigma_1$. For convenience we rename the variables $\mathcal{C}_3 = \mathcal{C}_3' \cdot \rho$. We ignore $\mathcal{C}_2$ which would just yield a variant of itself.

$$
\begin{aligned}
\mathcal{C}_3 &= \mathsf{s}(\mathsf{s}(x_3)) \not\approx \mathsf{s}(y_3) \vee \mathsf{s}(x_3) \approx y_3 \\
\mathcal{C}_{i+3} &= \mathsf{s}^{i+2}(x_{i+3}) \approx \mathsf{s}(y_{i+3}) \vee \mathsf{s}^{i+1}(x_{i+3}) \approx y_{i+3} && i = 0 \text{ (base case)}
\end{aligned}
$$

We now show by induction that `Inst-Gen-Eq` yields an infinite sequence of distinct clauses $\mathcal{C}_{i+3}$ for $i \in \mathbb{N}$. The base case $i = 0$ is already covered. We assume for simplicity and without loss of generality that the literal $\mathsf{s}^{i+1}(x_{i+3}) \approx y_{i+3}$ will never be selected.[4] We then can derive the contradiction from the selected literals of the first and the newest clause and instantiate the first clause with the new unifier σ_{i+2}.

$$
\begin{aligned}
S_{i+1} &= S_i \uplus \{\, \mathsf{s}^{i+2}(x_{i+3}) \not\approx \mathsf{s}(y_{i+3}) \vee \mathsf{s}^{i+1}(x_{i+3}) \approx y_{i+3} \,\} && i \geq 0 \text{ (IH)} \\
S_{(i+1)_\bot} &= S_{0_\bot} \uplus \{\, \mathsf{s}^{i+2}(\mathsf{c}_\bot) \not\approx \mathsf{s}(\mathsf{c}_\bot) \vee \mathsf{s}^{i+1}(\mathsf{c}_\bot) \approx \mathsf{c}_\bot \,\} \\
L_{i+1} &= L_i \uplus \{\, \mathsf{s}^{i+2}(x_{i+3})) \not\approx \mathsf{s}(y_{i+3}) \,\}
\end{aligned}
$$

$$
\dfrac{\dfrac{x_1 \approx y_1 \quad [\mathsf{s}^{i+2}(x_{i+3})] \not\approx \mathsf{s}(y_{i+3})}{y_1 \not\approx \mathsf{s}(y_{i+3})}\ \{y_1 \mapsto \mathsf{s}(y_{i+3})\}}{\square}\ \sigma_{i+2} = \{x_1 \mapsto \mathsf{s}^{i+2}(x_{i+3})\}
$$

$$
\begin{aligned}
\mathcal{C}'_{(i+1)+3} &= \mathcal{C}_1 \cdot \sigma_{i+2} = \mathsf{s}(\mathsf{s}^{i+2}(x_{i+3})) \not\approx \mathsf{s}(y_{i+3}) \vee \mathsf{s}^{i+2}(x_{i+4}) \approx y_{i+3} \\
\mathcal{C}_{(i+1)+3} &= \mathsf{s}^{(i+1)+2}(x_{i+4}) \not\approx \mathsf{s}(y_{i+4}) \vee \mathsf{s}^{(i+1)+1}(x_{i+4}) \approx y_{i+4} && \text{(step case)}
\end{aligned}
$$

[4] Otherwise we quickly derive the unit clause $\mathsf{s}^{i+1}(x_{i+3}) \not\approx x_2$ that prohibits the selection.

$$S_0 = \{\, \boxed{\mathsf{s}(x') \not\approx \mathsf{s}(y') \vee x' \approx y'},\ \mathsf{s}(x) \not\approx 0 \,\}$$

$$S_{0_\perp} = \{\, \mathsf{s}(\mathsf{c}_\perp) \not\approx \mathsf{s}(\mathsf{c}_\perp) \vee \mathsf{c}_\perp \approx \mathsf{c}_\perp,\ \mathsf{s}(\mathsf{c}_\perp) \not\approx 0 \,\}$$

$$\dfrac{\dfrac{x' \approx y' \quad \mathsf{s}(x) \not\approx x}{y' \not\approx x}\ y' \mapsto 0}{\square}\ x' \mapsto \mathsf{s}(x)$$

$$S_1 = S_0 \uplus \{\, \boxed{\mathsf{s}(\mathsf{s}(x)) \not\approx \mathsf{s}(0) \vee \mathsf{s}(x) \approx 0} \,\}$$

$$S_{i+1} = S_i \uplus \{\, \mathsf{s}^{i+2}(x)) \not\approx \mathsf{s}^{i+1}(0) \vee \mathsf{s}^{i+1}(x) \approx \mathsf{s}^{i}(0) \,\}$$

$$S_{(i+1)_\perp} = S_i \uplus \{\, \mathsf{s}^{i+2}(\mathsf{c}_\perp)) \not\approx \mathsf{s}^{i+1}(0) \vee \mathsf{s}^{i+1}(\mathsf{c}_\perp) \approx \mathsf{s}^{i}(0) \,\}$$

$$\dfrac{\dfrac{x' \approx y' \quad [\mathsf{s}^{i+2}(x)] \not\approx \mathsf{s}^{i+1}(0)}{y' \not\approx \mathsf{s}^{i+1}(0)}\ y' \mapsto \mathsf{s}^{i+1}(0)}{\square}\ x' \mapsto \mathsf{s}^{i+2}(x)$$

$$S_{i+2} = S_{i+1} \uplus \{\, \boxed{\mathsf{s}^{i+3}(x) \not\approx \mathsf{s}^{i+2}(0) \vee \mathsf{s}^{i+2}(x) \approx \mathsf{s}^{i+1}(0)} \,\}$$

4.5 Roundup

Calculus	Equality	Exit condition	Implementations
Gilmore	axioms	DNF	Gilmore
Resolution	axioms	$\square \in S$	
`Inst-Gen`	axioms	$\neg\mathtt{SAT}(S\perp)$	iProver
Superposition	rules	$\square \in S$	Vampire
`Inst-Gen-Eq`	rules	$\neg\mathtt{SMT}(S\perp)$	iProverEq

List of Figures

List of Tables

Bibliography

[1] F. Baader and T. Nipkow. *Term Rewriting and All That*. Cambridge University Press, New York, NY, USA, 1998.

[2] A. Biere, A. Biere, M. Heule, H. van Maaren, and T. Walsh. *Handbook of Satisfiability: Volume 185 Frontiers in Artificial Intelligence and Applications*. IOS Press, Amsterdam, The Netherlands, The Netherlands, 2009.

[3] E. Börger, E. Grädel, and Y. Gurevich. *The classical decision problem*. Perspectives in Mathematical Logic. Springer-Verlag, Berlin, 1997. With an appendix by Cyril Allauzen and Bruno Durand.

[4] M. Davis, G. Logemann, and D. Loveland. A machine program for theorem-proving. *Commun. ACM*, 5(7):394–397, July 1962.

[5] M. Davis and H. Putnam. A computing procedure for quantification theory. *J. ACM*, 7(3):201–215, July 1960.

[6] P. C. Gilmore. A proof method for quantification theory: Its justification and realization. *IBM Journal of Research and Development*, 4(1):28–35, Jan 1960.

[7] J. Harrison. *Handbook of Practical Logic and Automated Reasoning*. Cambridge University Press, New York, NY, USA, 1st edition, 2009.

[8] M. Huth and M. Ryan. *Logic in Computer Science: Modelling and Reasoning About Systems*. Cambridge University Press, New York, NY, USA, 2004.

[9] D. Kroening and O. Strichman. *Decision Procedures: An Algorithmic Point of View*. Springer Publishing Company, Incorporated, 1 edition, 2008.

[10] A. Middeldorp. Lecture Notes – Term Rewriting, 2015.

[11] A. Middeldorp. Lecture Notes – Logic, 2016.

[12] G. Moser. Lecture Notes – Module Automated Reasoning, 2013.

[13] D. A. Plaisted and S. Greenbaum. A structure-preserving clause form translation. *Journal of Symbolic Computation*, 2(3):293 – 304, 1986.

[14] G. S. Tseitin. On the complexity of derivations in the propositional calculus. *Studies in Constructive Mathematics and Mathematical Logic*, Part II:115–125, 1970.