2 Preliminaries

In this thesis we assume the reader's familiarity with propositional and first order logic [10], term rewriting (TRW [2]), decision procedures (DP [11]), and satisfiability checking modulo theories (SMT [3]). 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 the lecture notes to term rewriting and automated reasoning [12, 14].

2.1 Syntax

In this section we introduce the syntax of arbitrary first order formulae (`FOF`), prenex normal form (`PNF`), and clausal normal form (`CNF`).

Definition 2.1. A first order *signature* with equality $\mathcal{F} = \mathcal{F}_{\mathsf{f}} \,\dot{\cup}\, \mathcal{F}_{\mathsf{P}} \,\dot{\cup}\, \{\approx\}$ is the disjoint union of a set of *function symbols* $\mathcal{F}_{\mathsf{f}}$, a set of *predicate symbols* $\mathcal{F}_{\mathsf{P}}$, and one distinct 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.

Remark. We use $\approx$ as equality symbol in our signatures to emphasize that at this point it is just a highlighted symbol without "meaning". On the other hand we use $=$ to express "identity" of objects like formulae or sets without actually defining how this identity can be determined.

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. We define the set of variables of a first order term t as follows:

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

A *ground* term t' does not contain any variables, i.e. $\mathcal{V}ars(t') = \emptyset$.

Definition 2.4. 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 2.5. 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 pair $s \approx t$ is an equation for arbitrary terms s and t. The set of atomic formulas (or *atoms* for short) is the (distinct) union of predicates and equations.

2.1.1 Formulae and Normal forms

Definition 2.6 (`FOF`)**.** The atoms in Definition 2.5 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 with *bound* variable $x \in \mathcal{V}$. 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.

Remark. The Symbol $\to$ for implication is not to be confused with the equational symbol for rewrite rules in Section 2.3.

Definition 2.7. We define the set of *free* variables and the set of *bound* variables of a first order formula F as follows:

$$\mathcal{F}vars(F) = \begin{cases} \bigcup_{i=1}^{n} \mathcal{V}ars(t_i) & \text{if } F = \mathsf{P}(t_1, \ldots, t_n) \text{ or } t_1 \approx t_{n=2} \\ \mathcal{F}vars(G) & \text{if } F = \neg G \\ \mathcal{F}vars(G) \cup \mathcal{F}vars(H) & \text{if } F \in \{\, G \wedge H, G \vee H, G \to H \,\} \\ \mathcal{F}vars(G) \setminus \{x\} & \text{if } F \in \{\, \forall x\, G, \exists x\, G \,\} \end{cases}$$

$$\mathcal{B}vars(F) = \begin{cases} \emptyset & \text{if } F = \mathsf{P}(t_1, \ldots, t_n) \text{ or } t_1 \approx t_{n=2} \\ \mathcal{B}vars(G) & \text{if } F = \neg G \\ \mathcal{B}vars(G) \cup \mathcal{B}vars(H) & \text{if } F \in \{\, G \wedge H, G \vee H, G \to H \,\} \\ \{x\} \cup \mathcal{B}vars(G) & \text{if } F \in \{\, \forall x\, G, \exists x\, G \,\} \end{cases}$$

Example 2.8. With formula $F = (\forall x(x \approx y)) \vee (\exists y \mathsf{P}(x, y))$ we have $\mathcal{F}vars(F) = \mathcal{B}vars(F)$, which we would like to avoid: Formulae $F' = (\forall x(x \approx y')) \vee (\exists y \mathsf{P}(x', y))$ is equivalent to F (see Section 2.2 on page 8) and we get $\mathcal{F}vars(F') \cap \mathcal{B}vars(F') = \emptyset$.

We often will write formulae or sentences without stating the signature. The reader can easily deduce the underlying *implicit* signature with arities and the set of variables by applying the definitions of the syntax for first order formulae. We follow the convention to use x, y, z for variables and $\mathsf{a}, \mathsf{b}, \mathsf{c}$ for constant function symbols (which avoids ambiguity in the presence of free variables). For easier readability we will use uppercase predicate symbols and lowercase function symbols. We may denote $\mathcal{F}(F)$ for the implicit signature of an formula F.

Definition 2.9. A first order formula is closed, i.e. a first order *sentence*, if it does not contain free variables — all occurring variables are bound. Additionally we assume for any sentence that each variable is bound exactly once. Additionally the variable occurs as free variable in the subformula of the quantified subformula where it was bound:

$$\begin{aligned} \mathcal{B}vars(G * H) &= \mathcal{B}vars(G) \mathbin{\dot\cup} \mathcal{B}vars(H) & * \in \{\land, \lor, \to\} \\ \mathcal{B}vars(\text{Ɐ∃} x F) &= \{x\} \mathbin{\dot\cup} \mathcal{B}vars(F),\ x \in \mathcal{F}vars(F) & \text{Ɐ∃} \in \{\forall, \exists\} \end{aligned}$$

Example 2.10.

$$\begin{array}{llll} \forall x(\mathsf{P}(x) \lor \forall x \mathsf{Q}(x)) & ✗ & \forall x(\mathsf{P}(x) \lor \forall y \mathsf{Q}(y)) & ✓ \\ (\forall x \mathsf{P}(x)) \lor (\forall x \mathsf{Q}(x)) & ✗ & (\forall x \mathsf{P}(x)) \lor (\forall y \mathsf{Q}(y)) & ✓ \end{array}$$

Definition 2.11 (PNF). A first order sentence $F = \text{Ɐ∃}_1 x_1 \ldots \text{Ɐ∃}_n x_n\, G$ with n quantifiers $\text{Ɐ∃}_i \in \{\exists, \forall\}$, n bound and distinct variables x_i, and quantifier free subformula G with a matching set of free variables, i.e. $\mathcal{F}vars(G) = \mathcal{B}vars(F)$, is in *prenex normal form*.

Definition 2.12 (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* $\mathcal{C} = L_1 \lor \ldots \lor L_n$ 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. Implicitly ground first order formulae do not contain quantifiers.

2.1.2 Substitution

Definition 2.13. A *substitution* σ is a mapping from variables $x \in \mathcal{V}$ to terms in $\mathcal{T}(\mathcal{F}_\mathsf{f}, \mathcal{V})$ where *domain* $\mathrm{dom}(\sigma) = \{x \in \mathcal{V} \mid \sigma(x) \neq x\}$ and image $\mathrm{img}(\sigma) = \{\sigma(x) \mid x \in V, \sigma(x) \neq x\}$ are 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 (at least one variable is mapped to a non-variable term).

Definition 2.14. We define the instance $t\sigma$ respectively the application of a substitution σ to a literal or term t as follows

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

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

Definition 2.15. We can easily extend our definition to composite first order formulae, but the cases of quantified formulae need more consideration. So we only partially define $F\sigma$ for first order formulae F and substitution σ as follows (if $G\sigma$ and $H\sigma$ are defined in the respective cases).

$$F\sigma = \begin{cases} \neg(G\sigma) & \text{if } F = \neg G \\ (G\sigma) * (H\sigma) & \text{if } F = G * H, * \in \{\wedge, \vee, \rightarrow\} \\ \exists\!\!\!\forall x(G\sigma) & \text{if } F = \exists\!\!\!\forall xG, \exists\!\!\!\forall \in \{\forall, \exists\}, x \notin \operatorname{dom}(\sigma) \end{cases}$$

Definition 2.16. A clause $\mathcal{C}$ *strictly subsumes* a clause $\mathcal{D}$ if their exists a substitution θ such that $\mathcal{C}\theta \subsetneq \mathcal{D}$, e.g. when clause $\mathcal{D} = \mathcal{C}\theta \vee \mathcal{D}'$ is a weakened instance of clause $\mathcal{C}$.

Definition 2.17. We define the *composition* of two substitutions σ and τ as follows

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

Lemma 2.18. *With the definitions in 2.13 and 2.17 the equation $(t\sigma)\tau = t(\sigma\tau)$ holds for term, atoms, and literals.*

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

- (base case) Let $t = x_i \in \operatorname{dom}(\sigma)$ then $((x_i)\sigma)\tau \overset{\text{def}}{=} s_i\tau \overset{\text{def}}{=} x_i(\sigma\tau)$ holds.
- (base case) Let $t = y \notin \operatorname{dom}(\sigma)$ 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 2.19. 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* $\operatorname{mgu}(t, u)$ 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 $\operatorname{mgu}(L', L^c)$ exists.

Remark. The unification of quantified formulae remains undefined in this thesis.

Example 2.20. 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 2.21. 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 2.22. We define the set of *positions* in an atom or a term recursively,

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

the set of *term positions* in an atom or a term,

$$t\text{-}\mathcal{Pos}(t) = \begin{cases} \mathcal{Pos}(t) & \text{if } t \text{ is a term} \\ \mathcal{Pos}(t) \setminus \{\epsilon\} & \text{if } t \text{ is an atom} \end{cases}$$

the extraction of a subterm at a term position $p \in t\text{-}\mathcal{Pos}(t)$ from an atom or a term,

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

and the insertion of a term s at a term position $p \in t\text{-}\mathcal{Pos}(t)$ into an atom or a term by replacing the subterm at that term position.

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

We may write $t[s]$ if s is a subterm of t (at some term position $p \in t\text{-}\mathcal{Pos}(t)$, such that $t|_p = s$). With a follow up statement $t[s']$ in the same scope we express the replacement of subterm s with term s' in t, i.e. the application of $t[s']_p$.

2.1.3 Provability

In general a proof may be a finite sequence of proof steps from none or some premises via intermediate statements to a final, the then proven statement. A formal proof system or logical calculus describes admissible basic proof steps in the underlying logic of the statements, in our case first order logic. A formal proof comprises only proof steps confirmed by rules of the applied logical calculus.

Definition 2.23 ([10])**.** We recally the rules of *natural deduction* for connectives in Table 2.1, for equality in Table 2.2, and for quantifiers in Table 2.3. Natural deduction provides a logical calculus, i.e. a formal proof system for first order logic. The formulae F and G in these rules are sentences, the bound variable in $\forall x F'$ occurs free in F', and terms s and t are ground.

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

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

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

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

Table 2.1: Natural Deduction Rules for Connectives

$$\frac{s = t \quad F'\{x \mapsto s\}}{F'\{x \mapsto t\}}\,(=e) \qquad \frac{}{t = t}\,(=i)$$

Table 2.2: Natural Deduction Rules for Equality

Definition 2.24. A sentence in first order logic is provable if their exists a proof in a formal proof system for first order logic, e.g. natural deduction. We write $F_1, \ldots, F_n \vdash G$ when we can prove G from premises $F_1, \ldots, F_n$.

A natural deduction proof starts with a (possible empty) set of sentences — the premises — and infer other sentences — the conclusions — by applying the syntactic proof inference rules. A box must be opened for each assumption, e.g. a term or a sentence. Closing the box discards the assumption and all its conclusions within the box(es), but may introduce a derived sentence outside the box(es). Then $F_1, \ldots, F_n \vdash H$ claims that H is in the transitive closure of inferable formulae from $\{F_1, \ldots, F_n\}$ outside of any box.

Example 2.25. We show $\forall x(\mathsf{P}(x) \wedge \neg\mathsf{Q}(x)) \vdash \forall x(\neg\mathsf{Q}(x) \wedge \mathsf{P}(x))$ with natural deduction. We note our premise (1), we open a box and assume an arbitrary constant (2), we create a ground instance of our premise with quantifier elimination and the constant (3), we extract the literals with both variants of conjunction elimination (4, 5), we introduce a conjunction of the ground literals (6), and close the box to introduce the universal

$$\frac{\forall x F'}{F'\{x \to t\}}\ (\forall e) \qquad\qquad \frac{F'\{x \mapsto t\}}{\exists x F'}\ (\exists i)$$

$$\frac{\boxed{\begin{array}{l} t \\ \qquad \vdots \\ \quad F'\{x \mapsto t\} \end{array}}}{\forall x F'}\ (\forall i) \qquad\qquad \frac{\exists x F' \quad \boxed{\begin{array}{c} t \quad F'\{x \mapsto t\} \\ \vdots \\ H \end{array}}}{H}\ (\exists e)$$

Table 2.3: Natural Deduction Rules for Quantifiers

quantified conjunction (7).

1	$\forall x(\mathsf{P}(x) \land \neg\mathsf{Q}(x))$	`premise`
2	c	
3	$\mathsf{P}(\mathsf{c}) \land \neg\mathsf{Q}(\mathsf{c})$	$1 : \forall e$
4	$\neg\mathsf{Q}(\mathsf{c})$	$3 : \land e_1$
5	$\mathsf{P}(\mathsf{c})$	$3 : \land e_2$
6	$\neg\mathsf{Q}(\mathsf{c}) \land \mathsf{P}(\mathsf{c})$	$4 + 5 : \land i$
7	$\forall x(\neg\mathsf{Q}(x) \land \mathsf{P}(x))$	$2 - 6 : \forall i$

2.2 Semantics

In this section we recall some basic aspects and definitions of semantics in first order logic. We state satisfiability and validity of arbitrary first order formulae or sets of clauses.

2.2.1 Models

Definition 2.26. An *interpretation* $\mathcal{I}$ over a signature $\mathcal{F}$ consists of a non-empty set A (i.e. 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 assignment 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 2.27. 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 2.28 (Semantics of `FOF`)**.** A universally quantified sentence $\forall x F$ holds in an interpretation if its subformula F holds for all assignments for x. An existential quantified sentence $\exists x F$ holds if its subformula F holds for at least one assignment for x. For a given interpretation and predefined assignments for all occurring free variables 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 subformulae F and G hold, an implication $F \rightarrow G$ 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 2.29 (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 2.30. A *model* $\mathcal{M}$ for a set of clauses S (for a sentence F) is an interpretation that *satisfies* the set of clauses (the sentence), i.e. the set of clauses (the sentence) holds in that interpretation $\mathcal{M}$. We write $\mathcal{M} \models S$ or $\mathcal{M} \models F$.

A set of clauses (a sentence) is *satisfiable* if there exists at least one model for it. A set of clauses (a sentence) is *valid* if and only if every interpretation is a model.

Definition 2.31. 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 2.32. An *Herbrand interpretation* $\mathcal{H}$ is an interpretation where the domain is an Herbrand universe and the interpretation of each ground term $t_{\mathcal{H}} := t$ is the term itself.

2.2.2 Equivalence and Equisatisfiability

Definition 2.33. A (first order) sentence G is a *semantic consequence* of a set of sentences $\Gamma = \{F_1, \ldots, F_n\}$ if G holds in all models for $F_1, \ldots, F_n$. We write $\Gamma \vDash G$ and also say that Γ entails G. Two sentences $F \equiv G$ are *equivalent* if an only if the first sentence entails the second and vice versa. An *equivalence transformation* morphs an arbitrary sentence F to another sentence F' such that $F \equiv F'$. Equivalence transformations preserve validity *and* satisfiability of sentences.

Example 2.34. We can easily see that not satisfiable $F \land \neg F$ entails every formula, that valid $F \lor \neg F$ is entailed by every sequence, further that

$$\begin{array}{ccc} \Delta, F \models G & \text{if and only if} & \Delta \models \neg F \lor G \\ F_1, \ldots, F_n \models G & \text{if and only if} & F_1 \land \ldots \land F_n \models G \\ F \land G \equiv G \land F & F \lor G \equiv G \lor F & F \to G \equiv \neg F \lor \neg G \end{array}$$

by the definitions for semantics, entailment, and equivalence. □

Definition 2.35. Two sentences $F \approxeq G$ are *equisatisfiable* if F is satisfiable whenever G is satisfiable and the other way round. An *satisfiable transformation* morphs an arbitrary sentence F to a sentence F' such that $F \approxeq F'$. Equisatisfiable transformations respect the satisfiability of sentences.

Example 2.36. Let F be an arbitrary formula with $\mathcal{F}vars(F) = \{x\}$. It is easy to see that in general $F\{x \mapsto \mathsf{a}\} \not\equiv F\{x \mapsto \mathsf{b}\}$ but $F\{x \mapsto \mathsf{a}\} \approxeq F\{x \mapsto \mathsf{b}\}$, e.g. we can construct a model such that $\mathsf{P}(\mathsf{a})$ holds but $\mathsf{P}(\mathsf{b})$ does not. But undoubtedly $\mathsf{P}(\mathsf{a})$ is as satisfiable as $\mathsf{P}(\mathsf{b})$.

Example 2.37. $\exists x\, \mathsf{P}(x) \approxeq \mathsf{P}(\mathsf{a})$, but only $\mathsf{P}(\mathsf{a}) \models \exists x\, \mathsf{P}(x)$ holds.

Example 2.38.

$$\{\, \mathsf{P}(x, \mathsf{f}(x)),\ \mathsf{Q}(y, \mathsf{a}) \,\} \approxeq \forall x \exists y (\mathsf{P}(x, y)) \land \exists a \forall y (\mathsf{Q}(y, a)).$$

2.2.3 Equality

In Definition 2.27 we have interpreted the equality symbol as binary relation without restrictions. This allows unwelcome models as in Example 2.39. Hence we state useful definitions to deal with this situation and demonstrate their usage in an example.

Example 2.39. Any interpretation $\mathcal{I}$ with $\approx_\mathcal{I} = \emptyset$ satisfies $\mathsf{a} \not\approx \mathsf{a}$.

Definition 2.40. 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 2.41. 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 2.42. 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 Herbrand 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 and Orderings

Definition 2.43. A term rewrite signature $\mathcal{F}_\mathsf{f}$ is a set of function symbols with associated arities as in Definition 2.1. Terms, term variables, ground terms and unary function symbol notations are defined as in Definitions 2.2 to 2.4.

Definition 2.44. 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 variable 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. In a *ground* term rewrite system every term on every side in every rule is a ground term.

Although we use the the same symbol for implications $F \to G$ between first order formulae and rewrite rules $s \to t$ or rewrite steps $s' \to_{\mathcal{R}} t$ between first order terms, there will not arise any ambiguity for the reader about the role of the symbol.

Definition 2.45. We say $s \to_{\mathcal{R}} t$ is a *rewrite step* with respect to TRS $\mathcal{R}$ when there is a position $p \in \mathcal{P}os(s)$, 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 2.46. 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.

Definition 2.47. 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 $p \in \mathcal{P}os(u)$) and *closed under substitutions* (whenever $s \circledast t$ then $s\sigma \circledast t\sigma$ for an arbitrary substitution σ).

Lemma 2.48. *The relations $\to^*_{\mathcal{R}}$, $\to^+_{\mathcal{R}}$, $\downarrow_{\mathcal{R}}$, $\uparrow_{\mathcal{R}}$ are rewrite relations on every TRS $\mathcal{R}$.*

Definition 2.49. 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 $p \neq \epsilon$.

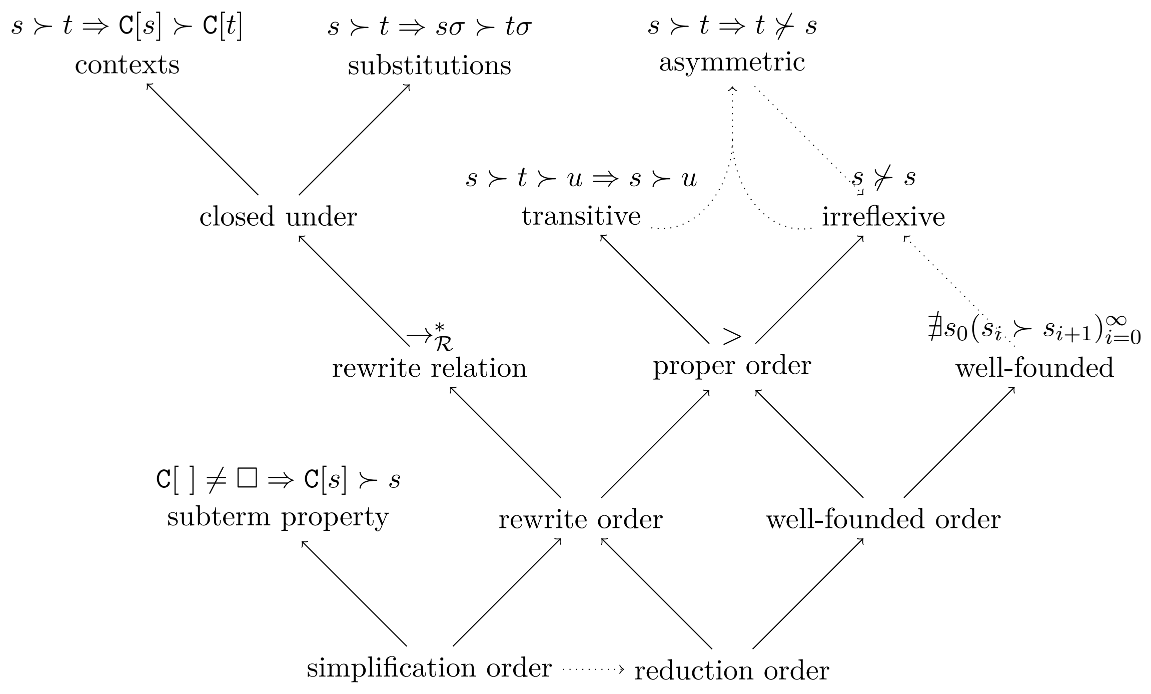
$s \succ t \Rightarrow \mathtt{C}[s] \succ \mathtt{C}[t]$
contexts
$s \succ t \Rightarrow s\sigma \succ t\sigma$
substitutions
$s \succ t \Rightarrow t \not\succ s$
asymmetric
closed under
$s \succ t \succ u \Rightarrow s \succ u$
transitive
$s \not\succ s$
irreflexive
$\rightarrow^*_{\mathcal{R}}$
rewrite relation
$>$
proper order
$\nexists s_0(s_i \succ s_{i+1})_{i=0}^{\infty}$
well-founded
$\mathtt{C}[\,] \neq \square \Rightarrow \mathtt{C}[s] \succ s$
subterm property
rewrite order
well-founded order
simplification order
reduction order

Figure 2.1: Properties of relations on terms

Figure 2.3 summarizes the properties of relations on terms. The solid arrows mark definitions, e.g. a rewrite order is closed under contexts and substitutions (Definition 2.47); a simplification order is a rewrite order that respects the subterm property (Definition 2.49). The dotted arrows mark derived properties, e.g. every simplification order is a reduction order (Lemma 2.50); transitive and irreflexive relations are always asymmetric, etc.

Lemma 2.50. *Every simplification order is well-founded, hence it is a reduction order.*

Theorem 2.51. *A TRS $\mathcal{R}$ is terminating if and only if there exists a reduction order $\succ$ such that $l \succ r$ for every rewrite rule $l \to r \in \mathcal{R}$. We call $\mathcal{R}$ simply terminating if $\succ$ is a simplification order.*

2.3.1 Clause and literal orderings

Lemma 2.52. *Any ordering $\succ$ on a set C can be extended to an ordering on multisets over C as follows $N \succ M$ if $N \neq M$ and whenever there is $x \in C$ with $N(x) < M(x)$ then there is $y \succ x$ with $N(y) > M(y)$.*

An ordering $\succ$ on terms can be extended to orderings on literals and clauses.

For the following definition we assume $\succ_{\mathbf{gr}}$ as a a total, well-founded and monotone extension from a total simplification ordering on ground terms to ground clauses [15].

Definition 2.53. We define an order $\succ_{\mathtt{L}}$ on ground closures of literals as an arbitrary total well-founded extension of $\succ_{\mathtt{gr}}$ such that $L \cdot \sigma \succ_{\mathtt{L}} L' \cdot \sigma'$ whenever $L\sigma \succ_{\mathtt{gr}} L'\sigma'$.

We define an order $\succ_{\mathtt{C}}$ on ground closures as an arbitrary total well-founded extension of $\succ'_{\mathtt{C}}$ — an inherently well-founded order defined as extension of $\succ_{\mathtt{gr}}$ such that $C \cdot \tau \succ'_{\mathtt{C}} D \cdot \rho$ whenever $C\tau \succ_{\mathtt{gr}} D\rho$ or $C\theta = D$ for an proper instantiator θ.

Lemma 2.54. *A well-founded and total order on general ground terms always exists.*

Definition 2.55 (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 $\succ_{\mathtt{L}}$ on literals such that for all atoms A and B with $A \succ B$ the relations $A \succ_{\mathtt{L}} B$, $\neg A \succ_{\mathtt{L}} \neg B$ and $\neg A \succ_{\mathtt{L}} A$ 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: $\succeq$) holds. We write $\succ_{gr}$ to suggest the existence of such a ground substitution τ.

3 Undecidability

A logical calculus, i.e. a formal (purely syntactical) proof system for an underlying logic, is *complete* if every (semantically) valid formula is (syntactically) provable from its premises by the calculus. In other words, every sentence that holds in all possible models for its premises is derivable from its premises by applying rules of the formal system only. Additional we expect a useful calculus to be *sound*, that is, every (syntactically) provable formula (semantically) holds in any model for its premises. Without premises a sentence has to be provable if and only if it holds in any interpretation.

We first state some completeness, undecidability, and other fundamental theorems about first order logic in Section 3.1. Then we enumerate decidable fragments of first order logic which can be described purely syntactically in Section 3.2 on the facing page. We conclude this chapter with a look at decidable first-order theories in Section 3.3, which are not necessarily contained in one of the syntactically describable and decidable fragments of first-order logic.

3.1 Theorems of First Order Logic

The most fundamental theorem about first order logic were introduced and proven in the first half of the 20th century.

Theorem 3.1 (Soundness)**.** *The inference rules of natural deduction (Definition 2.23 on page 6) are sound.*

Proof. We prove the soundness of each inference rule by case distinction and the use of the semantic definition of validity. □

Theorem 3.2 (Gödels Vollständigkeitssatz)**.** *Es gibt einen Kalkül der Prädikatenlogik erster Stufe derart, dass für jede Formelmenge Γ und für jede Formel φ gilt: φ folgt genau dann aus Γ, wenn φ im Kalkül aus Γ hergeleitet werden kann.*

Theorem 3.3. *Natural deduction is a complete calculus, i.e. a proof $\Gamma \vdash G$ exists, whenever $\Gamma \models G$.*

Theorem 3.4 (Completeness, Gödel 1929)**.** *Natural deduction as a sound formal proof system for first order logic is complete.*

Theorem 3.5 (Undecidability, Church 1936, Turing 1937)**.** *The satisfiability problem for first-order logic is undecidable.*

Theorem 3.6 (Trakhtenbort 1950, Craig 1950)**.** *The satisfiability problem for first-order logic on finite structures (domains) is undecidable.*

Definition 3.7 (Finite model property)**.** A logic has the finite model property if each non-theorem is falsified by some finite model.

Lemma 3.8 (Refutation)**.** *By definition of the semantics of negation a formula is valid if and only if its negation is not satisfiable.*

Theorem 3.9 (Compactness, Gödel 1930, Maltsev 1936)**.** *If every finite subset of a set of formulas S has a model then S has a model.*

Theorem 3.10 (Löwenheim Skolem, 1915, 1920)**.** *If a set of formulas S has a model then S has a countable model.*

Theorem 3.11 (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.*

Corollary 3.12. *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 3.13. *Skolemization preserves satisfiability.*
Tseytin's transformation preserves satisfiability.

Lemma 3.14. *Herbrandization preserves validity.*

Lemma 3.15. *With Skolemization and Tseytin transformation we can effectively transform a arbitrary first-order formula into an equisatisfiable set of clauses.*

3.2 Decidable Fragments of First Order Logic

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

Definition 3.16 ([4])**.** 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 $\Pi = \exists\!\!\!\forall_1 \ldots \exists\!\!\!\forall_n$, $\exists\!\!\!\forall_i \in \{\forall, \exists\}$ 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 equality in the formulae is indicated by the absence or presence of a subscript $\approx$.

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

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

Example 3.18. 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.

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

Remark. One unary function symbol can be added to these fragments of first order logic above without loosing decidability (see Table 3.2).

$[\,\exists^*\forall^*, all, (0)\,]_{=}$	(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 3.1: Decidable prefix classes with finite model property

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

Table 3.2: Decidable prefix classes with infinity axioms.

Lemma 3.19. *Satisfiability is decidable [4] in all prefix classes from Tables 3.1 and 3.2. Each of theses classes is closed under conjunction with respect to satisfiability.*

3.3 Theories in First Order Logic

We follow definitions and examples in [13].

Definition 3.20 (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 3.21. A theory with axioms $\forall x\, \mathsf{P}(x)$ and $\exists x\, \neg\mathsf{P}(x)$ is neither consistent nor satisfiable.

Lemma 3.22. *A first order theory is consistent if and only if it is satisfiable.*

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

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

3.3.1 Theory of equality

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

Definition 3.23 (Equivalence)**.** A binary relation $\approx$ over a domain is an equivalence relation if and only if the following axioms hold over the given domain.

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

Definition 3.24 ($\vec{x}$-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 3.25 (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 3.26. *The equivalence and congruence axioms of equality are provable with natural deduction (Definition 2.23 on page 6, Table 2.1 on page 7, Table 2.2 on page 7, and Table 2.3 on page 8).*

Proof. For brevity we skip the quantifier introductions (and handle variables like constants) for symmetry, transitivity, and congruence. Additionally we just show congruence for a unary function and a unary predicate symbol.

1	$\mathsf{c}_0 \quad \mathsf{c}_0 = \mathsf{c}_0$	$=i$
2	$\forall x\,(x = x)$	$\forall i, 1, \{x \mapsto \mathsf{c}_0\}$

1	$y = y$	$=i$
2	$x = y$	`assume`
3	$y \neq x$	`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	$\mathsf{f}(y) = \mathsf{f}(y)$	$=i$
2	$x = y$	`assume`
3	$\mathsf{f}(x) \neq \mathsf{f}(y)$	`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	$x = y \land y = z$	`assume`
2	$y = z$	$\land e_2$
3	$x = y$	$\land e_1, 2$
4	$x = z$	$=e, 2, 3$
5	$x = y \land y = z \to x = z$	$\to i, 2–4$

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

□

3.3.2 Natural numbers

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

Definition 3.27 (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.

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

Example 3.28. We may prove $\forall x\,(\mathsf{s}(x) \not\approx x)$ with $G(x) = \mathsf{s}(x) \not\approx x$ by induction.

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

Definition 3.29 (Addition). We introduce the binary addition symbol $+ \in \mathcal{F}_{\mathsf{f}}^{(2)}$ with two axioms about defining equalities of sums.

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

Example 3.30.

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

Theorem 3.31. *Presburger arithmetic (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 3.32 (Multiplication)**.** We introduce the binary multiplication symbol $\times \in \mathcal{F}_{\mathsf{f}}^{(2)}$ with two axioms about defining equalities of products.

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

Theorem 3.33. *Peano Arithmetic (Giuseppe Peano, 1889), i.e. a first-order theory that extends Presburger Arithmetic with multiplication, is incomplete (Gödel‘s second incompleteness theorem in 1932) and undecidable.*

Theorem 3.34. *The axioms of Peano Arithmetic appear consistent (Gentzen, 1936).*

Lemma 3.35 (ACN)**.** *Addition and Multiplication on natural numbers are associative, commutative, and determine neutral elements.*

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

List of Figures

List of Tables

Bibliography

[1] L. Albert, R. Casas, and F. Fages. Average-case analysis of unification algorithms. *Theoretical Computer Science*, 113(1):3 – 34, 1993.

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

[3] 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.

[4] 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.

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

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

[7] H. Ganzinger and K. Korovin. Integrating Equational Reasoning into Instantiation-Based Theorem Proving. In *18th CSL 2004. Proceedings*, volume 3210 of *LNCS*, pages 71–84, 2004.

[8] 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.

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

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

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

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

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

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

[15] R. Nieuwenhuis and A. Rubio. Paramodulation-based theorem proving. In A. Robinson and A. Voronkov, editors, *Handbook of Automated Reasoning*, volume I, pages 371–443. Elsevier, 2001.

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

[17] J. A. Robinson. A machine-oriented logic based on the resolution principle. *J. ACM*, 12(1):23–41, Jan. 1965.

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