2 Preliminaries

In this thesis we assume the reader's familiarity with propositional and first order logic [12], term rewriting (TRW) [2], decision procedures (DP) [14], and satisfiability checking (SAT) modulo theories (SMT) [4]. 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 [15, 17].

2.1 Syntax

In this section we introduce the syntax of arbitrary first order formulae (FOF), formulae in prenex normal form (PNF), and sentences in 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 $\mathcal{F}^{(n)} = \{\mathsf{¤} \in \mathcal{F} \mid \text{arity}(\mathsf{¤}) = n\}$ 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 a 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 First Order 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. F and F' are *subformulae* of the quantified or composite formulae above.

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:

$$\mathcal{F}vars(F) = \begin{cases} \bigcup_{i=1}^{n} \mathcal{V}ars(t_i) & \text{if } F = \mathsf{P}(t_1, \ldots, t_n), \mathsf{P} \in \mathcal{F}_\mathsf{P}^{(n)} \\ \mathcal{V}ars(t_1) \cup \mathcal{V}ars(t_2) & \text{if } F = t_1 \approx t_2 \\ \mathcal{F}vars(G) & \text{if } F = \neg G \\ \mathcal{F}vars(G) \cup \mathcal{F}vars(H) & \text{if } F \in \{ G \wedge H, G \vee H, G \to H \} \\ \mathcal{F}vars(G) \setminus \{x\} & \text{if } F \in \{ \forall x\, G, \exists x\, G \} \end{cases}$$

$$\mathcal{B}vars(F) = \begin{cases} \emptyset & \text{if } F = \mathsf{P}(t_1, \ldots, t_n), \mathsf{P} \in \mathcal{F}_\mathsf{P}^{(n)} \\ \emptyset & \text{if } F = t_1 \approx t_2 \\ \mathcal{B}vars(G) & \text{if } F = \neg G \\ \mathcal{B}vars(G) \cup \mathcal{B}vars(H) & \text{if } F \in \{ G \wedge H, G \vee H, G \to H \} \\ \{x\} \cup \mathcal{B}vars(G) & \text{if } F \in \{ \forall x\, G, \exists x\, G \} \end{cases}$$

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. In Section 2.2 on page 9 we will see that formula $F' = (\forall x(x \approx y')) \vee (\exists y \mathsf{P}(x', y))$ is equivalent to our first formula F and we get $\mathcal{F}vars(F') \cap \mathcal{B}vars(F') = \emptyset$.

We often will use 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 P, Q and lowercase function symbols $\mathsf{f}, \mathsf{g}, \mathsf{h}$ with indices. 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. For practical reasons we assume that the variables of a sentence are bound exactly once and occur as free variables in the subformulae of the quantified formulae where they were bound. Thus the following will be assured.

$$\begin{aligned} \mathcal{F}vars(\phi) &= \emptyset \text{ iff } \phi \text{ is a sentence} \\ \mathcal{B}vars(G * H) &= \mathcal{B}vars(G) \dot\cup \mathcal{B}vars(H) && * \in \{\vee, \wedge, \to\} \\ \mathcal{B}vars(\exists\!\!\!/ x F) &= \{x\} \dot\cup \mathcal{B}vars(F) && \exists\!\!\!/ \in \{\forall, \exists\} \\ \exists\!\!\!/ x F \Rightarrow x &\in \mathcal{F}vars(F), x \notin \mathcal{B}vars(F) && \exists\!\!\!/ \in \{\forall, \exists\} \end{aligned}$$

Example 2.10. The sentences on the left do not match the assumptions in Definition 2.9 but their equivalent (Definition 2.36 on page 10) counterparts on the right do.

$$\begin{array}{lrcll} \text{✗} & \forall x\,(\mathsf{P}(x) \vee \forall x\,\mathsf{Q}(x)) & \equiv & \forall x\,(\mathsf{P}(x) \vee \forall y\,\mathsf{Q}(y)) & \text{✓} \\ \text{✗} & (\forall x\,\mathsf{P}(x)) \vee (\forall x\,\mathsf{Q}(x)) & \equiv & (\forall x\,\mathsf{P}(x)) \vee (\forall y\,\mathsf{Q}(y)) & \text{✓} \\ \text{✗} & \forall x\,\mathsf{P}(\mathsf{a}) & \equiv & \mathsf{P}(\mathsf{a}) & \text{✓} \end{array}$$

Definition 2.11 (PNF)**.** A first order sentence $F = \exists\!\!\!/_1 x_1 \ldots \exists\!\!\!/_n x_n\, G$ with n quantifiers $\exists\!\!\!/_i \in \{\exists, \forall\}$, 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 $s \not\approx t$. Further we call atoms *positive* literals and negated atoms *negative* literals. We define the *complement* L^c of a literal as follows:

$$L^c = \begin{cases} \neg A & \text{if } L = A \quad \text{where } A \text{ is an atom} \\ A & \text{if } L = \neg A \quad \text{where } A \text{ is an atom} \end{cases}$$

A (first order) *clause* $\mathcal{C} = L_1 \vee \ldots \vee L_n$ is a possibly 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, formulae, sets of clauses are first order expressions. We call a first order expression without variables *ground*, i.e. we build ground first order expressions over an empty set of variables. Obviously, ground first order formulae cannot contain quantifiers that depend on variables.

2.1.2 Substitutions and Positions

We now introduce notions for the manipulation of clauses, literals and terms.

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 application of a substitution σ to term t, a literal L, and clauses C as follows

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

$$L\sigma = \begin{cases} \mathsf{P}(t_1\sigma, \ldots, t_n\sigma) & \text{if } L = \mathsf{P}(t_1, \ldots, t_n) \text{ where } \mathsf{P} \in \mathcal{F}_{\mathsf{P}}^{(n)} \\ t_1\sigma \approx t_2\sigma & \text{if } L = t_1 \approx t_2 \\ \neg(A\sigma) & \text{if } L = \neg A \text{ where } A \text{ is an atom} \end{cases}$$

$$C\sigma = \bigvee_{L \in C} L\sigma$$

Definition 2.15. We define the application of the special "grounding substitution" $\sigma_\perp$ to a term t with distinct constant symbol $\mathsf{c}_\perp \notin \mathcal{F}$ as follows

$$t\sigma = \begin{cases} \mathsf{c}_\perp & \text{if } t = x \in \mathcal{V} \\ \mathsf{f}(t_1\mathsf{c}_\perp, \ldots, t_n\mathsf{c}_\perp) & \text{if } t = \mathsf{f}(t_1, \ldots, t_n) \text{ where } \mathsf{f} \in \mathcal{F}_{\mathsf{f}}^{(n)} \end{cases}$$

We define the application of $\sigma_\perp$ to literals and clauses as in Definition 2.14.

Definition 2.16. We can easily extend the application of substitutions to composite first order formulae. The cases of quantified formulae need more consideration — we must not substitute bound variables.

$$F\sigma = \begin{cases} \neg(G\sigma) & \text{if } F = \neg G \\ (G\sigma) * (H\sigma) & \text{if } F = G * H, * \in \{\vee, \wedge, \rightarrow\} \\ \exists\!\!\!\forall x(G\sigma') & \text{if } \begin{cases} F = \exists\!\!\!\forall x\, G, \exists\!\!\!\forall \in \{\forall, \exists\} \text{ and} \\ \sigma'(x) = x, \sigma'(y) = \sigma(y) \text{ for all } y \neq x. \end{cases} \end{cases}$$

Definition 2.17. A formula G' is an *instance* of formula $F = \exists\!\!\!\forall x(G)$ if there exists a term t such that $G' = G\{x \mapsto t\}$. If F is closed (i.e. $\mathcal{F}vars(G) = \{x\}$) and t is a ground term (i.e. $\mathcal{V}ars(t) = \emptyset$) then G' is a ground instance of F.

Definition 2.18. A clause $\mathcal{C}$ *strictly subsumes* a clause $\mathcal{D}$ if there 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 2.19. 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 2.20. *With the Definitions in 2.13 and 2.19 the resulting expressions for* $(t\sigma)\tau$ *and* $t(\sigma\tau)$ *coincide for any term or literal* t*, i.e.* $(t\sigma)\tau = t(\sigma\tau)$.

Proof. Assume σ and τ are substitutions. We use induction on the structure of t to show that $(t\sigma)\tau = t(\tau\sigma)$ in all possible cases.

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

□

Definition 2.21. Two quantifier-free 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.

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

Example 2.22. 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 2.23. 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 $q \backslash p = r$. 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 $\mathrm{head}(p\backslash r) <_{\mathbb{N}} \mathrm{head}(p\backslash r)$ for maximal common prefix r.

Definition 2.24. We define the set of *positions* in an atom or a term recursively,

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

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

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

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

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

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

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

We may write $t[s]$ if s is a subterm of t (at some term position $p \in t\text{-}\mathcal{P}os(t)$, 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 $t[s']_p$.

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.

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

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

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

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

Table 2.1: Natural Deduction Rules for Connectives

Definition 2.25. We say a term t is *free* for variable x in a formula F if for every variable $y \in \mathcal{V}ars(t)$ and every subformula $G = \forall y H$ of F, the variable $x \notin \mathcal{F}vars(H)$.

Definition 2.26 ([12])**.** We recall the rules of *natural deduction* for connectives in Table 2.1 where F, G, and H are arbitrary first order formulae, equality in Table 2.2 where first order terms s and t are free for x in F', and quantifiers in Table 2.3 where x_0 is a *fresh* variable symbol, i.e. x_0 did not occur in any formula so far.

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

Table 2.2: Natural Deduction Rules for Equality

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

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

Table 2.3: Natural Deduction Rules for Quantifiers

Definition 2.27. 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 outside of any box and has been inferred (via a finite set of intermediate formulae) from $\{F_1, \ldots, F_n\}$.

Example 2.28. 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 a fresh variable x_0(2), we create an instance of our premise with quantifier elimination and the variable x_0 (3), we extract the literals with both variants of conjunction elimination (4, 5), we introduce a conjunction of the literals (6), and close the box to introduce the universally quantified conjunction

(7).

1	$\forall x(\mathsf{P}(x) \wedge \neg\mathsf{Q}(x))$	`premise`
2	x_0	
3	$\mathsf{P}(x_0) \wedge \neg\mathsf{Q}(x_0)$	$1 : \forall e$
4	$\neg\mathsf{Q}(x_0)$	$3 : \wedge e_1$
5	$\mathsf{P}(x_0)$	$3 : \wedge e_2$
6	$\neg\mathsf{Q}(x_0) \wedge \mathsf{P}(x_0)$	$4 + 5 : \wedge i$
7	$\forall x(\neg\mathsf{Q}(x) \wedge \mathsf{P}(x))$	$2 - 6 : \forall i$

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 2.29. An *interpretation* $\mathcal{I}$ over a first order 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 (possibly empty) 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 an assignment α and an interpretation $\mathcal{I}$:

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

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

Definition 2.30. A predicate $\mathsf{P}(t_1, \ldots, t_n)$ *holds* for assignment α and interpretation $\mathcal{I}$ if and only if the evaluation of its n-tuple $\alpha_\mathcal{I}(t_1, \ldots, t_n) = (\alpha_\mathcal{I}(t_1), \ldots, \alpha_\mathcal{I}(t_n)) \in \mathsf{P}_\mathcal{I}$ is an element of the relation $\mathsf{P}_\mathcal{I} \subseteq A^n$. Similarly an equation $s \approx t$ holds if $\alpha_\mathcal{I}(s) \approx_\mathcal{I} \alpha_\mathcal{I}(t)$.

Definition 2.31 (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 existentially 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 $F \to G$ holds if its first subformula F does not hold or its second subformula F' holds (or both).

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

Definition 2.32 (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 2.33. 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 2.34. 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 2.35. 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 Equisatisfiability

Definition 2.36. 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 say that Γ entails G. Two sentences are *equivalent*, denoted $F \equiv G$, 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 2.37. For an arbitrary first order sentence F we can easily see the composite sentence $F \wedge \neg F$ is unsatisfiable and entails every sentence, that the composite sentence $F \vee \neg F$ is valid and entailed by every sequence of sentences, and further that

$$\begin{array}{rcl} \Delta, F \vDash G & \text{if and only if} & \Delta \vDash \neg F \vee G \\ F_1, \ldots, F_n \vDash G & \text{if and only if} & F_1 \wedge \ldots \wedge F_n \vDash G \\ F \wedge G \equiv G \wedge F & F \vee G \equiv G \vee F & F \to G \equiv \neg F \vee G \\ \forall x(G) \equiv \forall x'(G\{x \mapsto x'\}) & \text{if } x' \notin \mathcal{F}vars(G) & \text{renaming of one bound variable} \end{array}$$

by the definitions for semantics, entailment, and equivalence. $\square$

Lemma 2.38 (Renaming of bound variables)**.** *Let $F[G]_p$ be a first order sentence with quantified subformula $G = (\forall\!\!\!\forall x\, H)$ at position p. We assume an injective position index function $i(p)$, i.e. $\forall p \forall q (p \approx q \vee i(p) \not\approx i(q))$, from positions to natural numbers and $x'_{i(p)} \notin \mathcal{Vars}(F)$. Then $G' = \forall\!\!\!\forall x'_{i(p)}(H\{x \mapsto x'_{i(p)}\})$ is well-formed with $G \equiv G'$ and $F[\,G\,]_p \equiv F[\,G'\,]_p$.*

We use Lemma 2.38 to justify our assumption about first order sentences in Definition 2.9 by renaming each quantified variable by its position index.

Definition 2.39. Two sentences $F \approx\!\!\!\!\approx G$ are *equisatisfiable* if F is satisfiable whenever G is satisfiable and the other way round. A *satisfiability transformation* morphs an arbitrary sentence F to a sentence F' such that $F \approx\!\!\!\!\approx F'$. Satisfiability transformations respect the satisfiability of sentences, but the validity may differ between F and F'.

Remark. By itself equisatisfiability of arbitrary formulae usually gives no insights since any unsatisfiable formulae is equisatisfiable to *any* unsatisfiable formula and any satisfiable formula is equisatisfiable to *any* satisfiable formula by definition, e.g. $\mathsf{P}(\mathsf{a}) \approx\!\!\!\!\approx \mathsf{b} \approx \mathsf{c}$.

Example 2.40. Let F be a sentence. Then $F \approx\!\!\!\!\approx F \wedge \exists x \mathsf{P}(x)$. But $F \not\models F \wedge \exists x \mathsf{P}(x)$ as we can see for the tautology $F = \mathsf{P}(\mathsf{a}) \vee \neg\mathsf{P}(\mathsf{a})$ and interpretation $\mathsf{P}_\mathcal{I} = \emptyset$.

Example 2.41. Let F be an arbitrary formula with $\mathcal{Fvars}(F) = \{x\}$. It is easy to see that in general $F\{x \mapsto \mathsf{a}\} \not\equiv F\{x \mapsto \mathsf{b}\}$ but $F\{x \mapsto \mathsf{a}\} \approx\!\!\!\!\approx F\{x \mapsto \mathsf{b}\}$, 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 2.42. $\exists x\, \mathsf{P}(x) \approx\!\!\!\!\approx \mathsf{P}(\mathsf{a})$, but only $\mathsf{P}(\mathsf{a}) \vDash \exists x\, \mathsf{P}(x)$ holds.

2.2.3 Equality

Example 2.43. Any interpretation $\mathcal{I}$ with $\approx_\mathcal{I} = \emptyset$ satisfies $\mathsf{a} \not\approx \mathsf{a}$.

In Definition 2.30 on page 9 we allowed to interpret the equality symbol without restrictions. This may yield unwelcome models as in Example 2.43. Hence we state useful definitions to deal with this situation and demonstrate their usage in an example.

Definition 2.44. A *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 2.45. 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 2.46. 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 a normal model $\mathcal{M}$ with domain $\{\mathsf{c}\}$, function definition $\mathsf{f}_{\mathcal{M}}(\mathsf{c}) = \mathsf{c}$, 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 the ground term $\mathsf{f}(\mathsf{c})_{\mathcal{M}} = \mathsf{f}_{\mathcal{M}}(\mathsf{c}) = \mathsf{c}$ 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 Term Orderings

Term rewriting provides theoretical foundations for practical procedures when dealing with equations of terms. This section follows basic definitions and notions as in [16].

Definition 2.47. 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 2.48. A *rewrite rule* $\ell \to r$ 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* (TRS) 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.

Definition 2.49. 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 $\mathcal{R}$.

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, because the implication is used between first order formulae and the rewrite symbol is used between (function) terms.

Definition 2.50. 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 2.51. 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).

Definition 2.52. Let $\mathcal{R}$ be a TRS, s and t be terms. The relation $s \to_{\mathcal{R}}^* t$ holds if s can be rewritten to t. $s \to_{\mathcal{R}}^+ t$ holds s can be rewritten to t and the length of the rewrite sequence is at least one, relation $s \downarrow_{\mathcal{R}} t$ holds if there is a term r such that $r \to_{\mathcal{R}}^* s$ and $r \to_{\mathcal{R}}^* t$, and relation $\uparrow_{\mathcal{R}}$ holds if there is a term u such that $s \to_{\mathcal{R}}^* u$ and $t \to_{\mathcal{R}}^* u$.

Lemma 2.53. *The relations $\to_{\mathcal{R}}^*$, $\to_{\mathcal{R}}^+$, $\downarrow_{\mathcal{R}}$, $\uparrow_{\mathcal{R}}$ are rewrite relations on every TRS $\mathcal{R}$.*

Definition 2.54. 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 $p \neq \epsilon$.

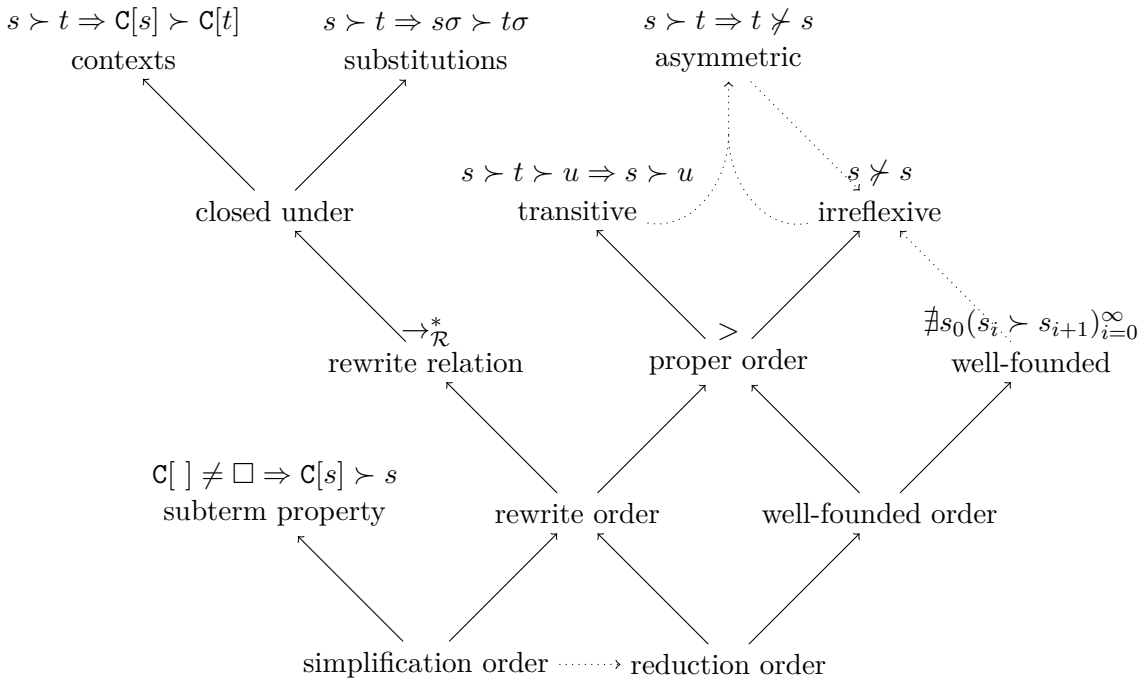

Figure 2.1: Properties of relations on terms

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 2.51); a simplification order is a rewrite order that respects the subterm property (Definition 2.54). The dotted arrows mark derived properties, e.g. every simplification order is a reduction order (Lemma 2.55); transitive and irreflexive relations are always asymmetric, etc.

Lemma 2.55. *Every simplification order is well-founded, hence it is a reduction order.*

Theorem 2.56. *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.*

Lemma 2.57. *A total simplification order over ground terms always exists [18].*

Lemma 2.58. *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.

Definition 2.59 (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.

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] D. W. Bennett. An elementary completeness proof for a system of natural deduction. *Notre Dame J. Formal Logic*, 14(3):430–432, 07 1973.

[4] 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.

[5] 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.

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

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

[8] 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.

[9] 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.

[10] P. Graf and D. Fehrer. *Term Indexing*, pages 125–147. Springer Netherlands, Dordrecht, 1998.

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

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

[13] K. Korovin. Inst-Gen – a Modular Approch. In *IJCAR 2008. Proceedings*, pages 292–298.

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

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

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

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

[18] 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.

[19] N. Olivetti and A. Tiwari, editors. *Automated Reasoning - 8th International Joint Conference, IJCAR 2016, Coimbra, Portugal, June 27 - July 2, 2016, Proceedings*, volume 9706 of *Lecture Notes in Computer Science*. Springer, 2016.

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

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

[22] S. Schulz and M. Möhrmann. Performance of clause selection heuristics for saturation-based theorem proving. In N. Olivetti and A. Tiwari, editors, *Automated Reasoning - 8th International Joint Conference, IJCAR 2016, Coimbra, Portugal, June 27 - July 2, 2016, Proceedings*, volume 9706 of *Lecture Notes in Computer Science*, pages 330–345. Springer, 2016.

[23] C. Sticksel. *Efficient Equational Reasoning for the Inst-Gen framework*. PhD thesis, School of Computer Science, University of Manchester, 2011.

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

[25] L. Wos, G. A. Robinson, and D. F. Carson. Efficiency and completeness of the set of support strategy in theorem proving. *J. ACM*, 12(4):536–541, Oct. 1965.