4. Resolution can be combined with more powerful redundancy elimination methods; because of its global nature this is more difficult for the tableau method.

5. Resolution can be refined to work well with equality; for tableau this seems to be impossible.

6. On the other hand tableau calculi can be easily extended to other logics; in particular tableau provers are very successful in modal and description logics.

3.9 First-Order CNF Transformation

Similar to the propositional case, first-order resolution and superposition operate on clauses. In this section I show how any first-order sentence can be efficiently transformed into a CNF, preserving satisfiability. To this end all existentially quantified variables are replaced with so called Skolem functions. Similar to the renaming of subformulas this replacement preserves satisfiability only. Eventually, all variables in clauses are implicitly universally quantified.

More concretely, the acnf CNF transformation is algorithm from Section 2.5.3 is generalized to first-order logic with equality. The adiitional complications are: (i) additional rules for the quantifiers, (ii) the formula renaming technique is extended to cope with variables and (iii) removal of existential quantifiers through the introduction of *Skolem* functions. Basically, all rules known from the propositional case apply.

The first two extra rules eliminate $\top$ and $\bot$ from first-order formula starting with a quantifier.

ElimTB13 $\chi[\{\forall,\exists\}x.\top]_p \Rightarrow_{\text{ACNF}} \chi[\top]_p$

ElimTB14 $\chi[\{\forall,\exists\}x.\bot]_p \Rightarrow_{\text{ACNF}} \chi[\bot]_p$

Next, in order to obtain a negation normal form with negation symbols in front of atoms only, the respective rules for pushing negations over the quantifiers are needed as well.

PushNeg4 $\chi[\neg\forall x.\phi]_p \Rightarrow_{\text{ACNF}} \chi[\exists x.\neg\phi]_p$

PushNeg5 $\chi[\neg\exists x.\phi]_p \Rightarrow_{\text{ACNF}} \chi[\forall x.\neg\phi]_p$

where the expression $\{\forall, \exists\}x.\phi$ covers both cases $\forall x.\phi$ and $\exists x.\phi$. The next step is to rename all variables such that different quantifiers bind different variables. This step is necessary to prevent a later on confusion of variables, once the quantifiers are dropped.

RenVar $\phi \Rightarrow_{\text{ACNF}} \phi\sigma$
for $\sigma = \{\}$

In first-order logic, the renaming of subformulas has to take care of variables as well. The notion of an obvious position remains unchanged. Therefore, the basic mechanism of renaming and the concept of a beneficial subformula is exactly the same as in propositional logic. The only difference is that renaming does introduce an atom in the free variables of the respective subformula. When some formula ψ is renamed at position p an atom $P(\vec{x}_n)$, $\vec{x}_n = x_1, \ldots, x_n$ replaces $\psi|_p$ where $\text{fvars}(\psi|_p) = \{x_1 \ldots, x_n\}$. The respective definition of $P(\vec{x}_n)$ becomes

$$\text{def}(\psi, p, P(\vec{x}_n)) := \begin{cases} \forall \vec{x}_n.(P(\vec{x}_n) \rightarrow \psi|_p) & \text{if } \text{pol}(\psi, p) = 1 \\ \forall \vec{x}_n.(\psi|_p \rightarrow P(\vec{x}_n)) & \text{if } \text{pol}(\psi, p) = -1 \\ \forall \vec{x}_n.(P(\vec{x}_n) \leftrightarrow \psi|_p) & \text{if } \text{pol}(\psi, p) = 0 \end{cases}$$

and the rule SimpleRenaming is changed accordingly.

SimpleRenaming $\phi \Rightarrow_{\text{ACNF}} \phi[P_1(\vec{x_1}_{,j_1})]_{p_1}[P_2(\vec{x_2}_{,j_2})]_{p_2} \ldots [P_n(\vec{x_n}_{,j_n})]_{p_n} \wedge$
$\text{def}(\phi, p_1, P_1(\vec{x_1}_{,j_1})) \wedge \ldots \wedge \text{def}(\phi[P_1(\vec{x_1}_{,j_1})]_{p_1}[P_2(\vec{x_2}_{,j_2})]_{p_2} \ldots [P_{n-1}(\vec{x_{n-1}}_{,j_{n-1}})]_{p_{n-1}}, p_n, P_n(\vec{x_n}_{,j_n}))$
provided $\{p_1, \ldots, p_n\} \subset \text{pos}(\phi)$ and for all $i, i+j$ either $p_i \parallel p_{i+j}$ or $p_i > p_{i+j}$
and where $\text{fvars}(\phi|_{p_i}) = \{x_{i,1}, \ldots, x_{i,j_i}\}$ and all P_i are different and new to ϕ

SimpleRenaming shares the variables of ϕ with the variables used for the definitions of the new predicates. This does not cause any confusion, because there will never be a clause consisting of literals from the remaning ϕ after renaming and literals from a definition. In propositional logic after subformula renaming, removal of equivalences and implications, and pushing negations down in front of atoms, the CNF can be generated using distributivity. In first-order logic the existential quantifiers are eliminated first by the introduction of Skolem functions. In order to receive Skolem functions with few arguments, the quantifiers are first moved inwards as far as passible. This step is called *mini-scoping*.

MiniScope1 $\chi[\forall x.(\psi_1 \circ \psi_2)]_p \Rightarrow_{\text{ACNF}} \chi[(\forall x.\psi_1) \circ \psi_2]_p$
provided $\circ \in \{\wedge, \vee\}$, $x \notin \text{fvars}(\psi_2)$

MiniScope2 $\chi[\exists x.(\psi_1 \circ \psi_2)]_p \Rightarrow_{\text{ACNF}} \chi[(\exists x.\psi_1) \circ \psi_2]_p$
provided $\circ \in \{\wedge, \vee\}$, $x \notin \text{fvars}(\psi_2)$

MiniScope3 $\chi[\forall x.(\psi_1 \wedge \psi_2)]_p \Rightarrow_{\text{ACNF}} \chi[(\forall x.\psi_1) \wedge (\forall x.\psi_2)\sigma]_p$

where $\sigma = \{\}$, $x \in (\text{fvars}(\psi_1) \cap \text{fvars}(\psi_2))$

MiniScope4 $\chi[\exists x.(\psi_1 \vee \psi_2)]_p \Rightarrow_{\text{ACNF}} \chi[(\exists x.\psi_1) \vee (\exists x.\psi_2)\sigma]_p$
where $\sigma = \{\}, x \in (\text{fvars}(\psi_1) \cap \text{fvars}(\psi_2))$

The rules MiniScope1, MiniScope2 are applied modulo the commutativity of $\wedge$, $\vee$. Once the quantifiers are moved inwards Skolemization can take place. Skolemization replaces all existentially quantified variables by shallow Skolem function terms.

Skolemization $\chi[\exists x.\phi]_p \Rightarrow_{\text{ACNF}} \chi[\phi\{x \mapsto f(y_1, \ldots, y_n)\}]_p$
provided there is no q, $q < p$ with $\phi|_q = \exists x'.\psi'$, $\text{fvars}(\exists x.\psi) = \{y_1, \ldots, y_n\}$, $f : \text{sort}(y_1) \times \ldots \times \text{sort}(y_n) \to \text{sort}(x)$ is a new function symbol

Theorem 3.9.1 (Skolemization Preserves Satisfiability)**.** A formula $\chi[\exists x.\phi]_p$ is satisfiable iff the formula $\chi[\phi\{x \mapsto f(y_1, \ldots, y_n)\}]_p$ is, where χ is in negation normal form, p the maximal position of an existential quantifier, $\text{fvars}(\exists x.\psi) = \{y_1, \ldots, y_n\}$, and $\text{arity}(f) = n$ is a new function symbol to ϕ, $f : \text{sort}(y_1) \times \ldots \times \text{sort}(y_n) \to \text{sort}(x)$.

Proof. Both directions of the proof are done by induction on the length of p and then by a case analysis on the structure of the formula. I only show the relevant cases.

$\Rightarrow$: If $\mathcal{A}, \beta \models \exists x.\phi$ then there exists an $a \in (\text{sort}(x))^{\mathcal{A}}$ such that $\mathcal{A}, \beta[x \mapsto a] \models \phi$. Now define $f^{\mathcal{A}}(\beta(y_1), \ldots, \beta(y_n)) := a$. Then obviously $\mathcal{A}, \beta \models \chi[\phi\{x \mapsto f(y_1, \ldots, y_n)\}]_p$. The function $f^{\mathcal{A}}$ is well-defined, because the truth value of $\exists x.\phi$ under $\mathcal{A}, \beta$ depends only on the values β assigns to the free variables of $\exists x.\phi$, i.e., the free variables $\text{fvars}(\exists x.\psi) = \{y_1, \ldots, y_n\}$ the function f depends on.

$\Leftarrow$: If $\mathcal{A}, \beta \models \chi[\phi\{x \mapsto f(y_1, \ldots, y_n)\}]_p$ then $\mathcal{A}, \beta[x \mapsto f^{\mathcal{A}}(\beta(y_1), \ldots, \beta(y_n))] \models \phi$ and therefore $\mathcal{A}, \beta \models \exists x.\phi$. □

Example 3.9.2 (Mini-Scoping and Skolemization)**.** Consider the simple formula $\forall x.\exists y.(R(x, x) \wedge P(y))$. Applying Skolemization directly to this formula, without mini-scoping results in

$$\forall x.\exists y.(R(x, x) \wedge P(y)) \Rightarrow^{\text{Skolem}}_{\text{ACNF}} \forall x.(R(x, x) \wedge P(g(x))$$

for a unary Skolem function g because $\text{fvars}(\exists y.(R(x, x) \wedge P(y))) = \{x\}$. Applying mini-scoping and then Skolemization generates

$$\begin{array}{lll} \forall x.\exists y.(R(x, x) \wedge P(y)) & \Rightarrow^{\text{MiniScope},*}_{\text{ACNF}} & \forall x.R(x, x) \wedge \exists y.P(y) \\ & \Rightarrow^{\text{Skolem}}_{\text{ACNF}} & \forall x.R(x, x) \wedge P(a) \end{array}$$

for some Skolem constant $a :\to \text{sort}(y)$ because $\text{fvars}(\exists y.P(y)) = \emptyset$. Now the former formula after Skolemization is seriously more complex than the latter. The former belongs to an undecidable fragment of first-order logic while the latter belongs to a decidable one (see Section 3.15).

Finally, the universal quantifiers are removed. In a first-order logic CNF any variable is universally quantified by default. Furthermore, the variables of two different clauses are always assumed to be different.

RemForall $\chi[\forall x.\psi]_p \Rightarrow_{\text{ACNF}} \chi[\psi]_p$

The actual CNF is then done by distributivity, exactly as it is done in propositional logic.

Algorithm 11: acnf(ϕ)

Input : A first-order formula ϕ.
Output: A formula ψ in CNF satisfiability preserving to ϕ.

```
1  whilerule (ElimTB1(φ),...,ElimTB14(φ)) do ;
2  RenVar(φ);
3  SimpleRenaming(φ) on obvious positions;
4  whilerule (ElimEquiv1(φ),ElimEquiv2(φ)) do ;
5  whilerule (ElimImp(φ)) do ;
6  whilerule (PushNeg1(φ),...,PushNeg5(φ)) do ;
7  whilerule (MiniScope1(φ),...,MiniScope4(φ)) do ;
8  whilerule (Skolemization(φ)) do ;
9  whilerule (RemForall(φ)) do ;
10 whilerule (PushDisj(φ)) do ;
11 return φ;
```

Theorem 3.9.3 (Properties of the ACNF Transformation)**.** Let ϕ be a first-order sentence, then

1. acnf(ϕ) terminates
2. ϕ is satisfiable iff acnf(ϕ) is satisfiable

Proof. (Idea) 1. is a straightforward extension of the propositional case. It is easy to define a measure for any line of Algorithm 11.
2. can also be established separately for all rule applications. The rules SimpleRenaming and Skolemization need separate proofs, the rest is straightforward or copied from the propositional case. □

In addition to the consideration of repeated subformulas, discussed in Section 2.5, for first-order renaming another technique can pay off: generalization. Consider the formula $[\phi_1 \vee (Q_1(a_1) \wedge Q_2(a_1))] \wedge [\phi_2 \vee (Q_1(a_2) \wedge Q_2(a_2))] \wedge \ldots \wedge [\phi_n \vee (Q_1(a_n) \wedge Q_2(a_n)]$. SimpleRenaming on obvious renamings applied to this formula will independently rename any occurrences of a formula $(Q_1(a_i) \wedge Q_2(a_i))$. However generalization pays off here. By adding

the definition $\forall x, y\,(R(x,y) \rightarrow (Q_1(x) \wedge Q_2(y)))$ and replacing the i^{th} occurrence of the conjunct by $R(x,y)\{x \mapsto a_i, y \mapsto a_i\}$ one definition for all subformula occurrences suffices.

3.10 First-Order Resolution

As already mentioned, I still consider first-order logic without equality. First-order resolution on ground clauses corresponds to propositional resolution. Each ground atom becomes a propositional variable. However, since there are up to infinitely many ground instances for a first-oder clause set with variables and it is not a priori known which ground instances are needed in a proof, the first-order resolution calculus operates on clauses with variables. Roughly, the relationship between ground resolution and first-order resolution corresponds to the relationship between standard tableau and free-variable tableau. However, the variables in free-variable tablea can only be instantiated once, thereas in resolution they can be instantiated arbitrarily often.

Propositional (or first-order ground) resolution is refutationally complete, without reduction rules it is not guaranteed to terminate for satisfiable sets of clauses, and inferior to the CDCL calculus. However, in contrast to the CDCL calculus, resolution can be easily extended to non-ground clauses via unification. The problem to lift the CDCL calculus lies in the lifting of the model representation of the trail. I'll discuss this in more detail in Section 3.15.

Lemma 3.10.1. Let $\mathcal{A}$ be a $\Sigma - algebra$ and let ϕ be a $\Sigma - formula$ with free variables $x_1, \ldots, x_n$. Then $\mathcal{A} \models \forall x_1, \ldots, x_n \phi$ iff $\mathcal{A} \models \phi$

Lemma 3.10.2. Let ϕ be a $\Sigma - formula$ with free variables $x_1, \ldots, x_n$, let σ be a substitution and let $y_1, \ldots, y_m$ be free variables of $\phi\sigma$. Then $\mathcal{A} \models \forall x_1, \ldots, x_n \phi$ implies $\mathcal{A} \models \forall y_1, \ldots, y_m \phi\sigma$.

In particular, if $\mathcal{A}$ is a model of an (implicitly universally quantified) clause C then it is also a model of all (implicitly universally quantified) instances $C\sigma$ of C. Consequently, if it is shown that some instances of clauses in a set N are unsatisfiable then it is also shown that N itself is unsatisfiable.

General Resolution through Instantiation

The approach is to instantiate clauses appropriately. An example is shown in Figure 3.3. However, this may lead to several problems. First of all, more than one instance of a clause can participate in a proof and secondly, which is even worse, there are infinitely many possible instances. Due to the fact that instantiation must produce complementary literals so that inferences become possible, the idea is to not instantiate more than necessary to get complementary literals. An instantiation of the clause set from Figure 3.3 is again shown in Figure 3.4 with the difference that the latter instantiates only as much as necessary, inevitably reducing the number of substitutions.

Lifting Principle

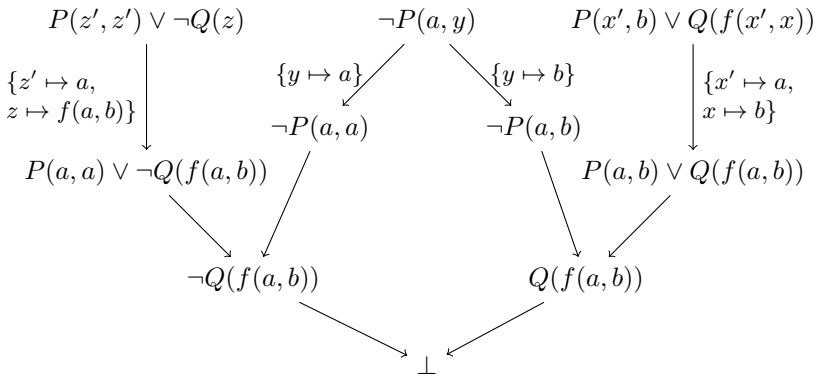

Figure 3.3: Instantiation of the clause set $N = P(z', z') \vee \neg Q(z), \neg P(a, y), P(x', b) \vee Q(f(x', x))$

In order to overcome the problem of effectively and efficiently saturating infinite sets of clauses as they arise from taking the (ground) instances of finitely many *general* clauses (with variables), the general idea is to lift the resolution principle as proposed by Robinson [41]. The lifting is as follows: For the resolution of general clauses, *equality* of ground atoms is generalized to *unifiability* of general atoms and only the *most general* (minimal) unifiers (mgu) are computed.

The advantage of the method in Robinson [41] compared with Gilmore [21] is that unification enumerates only those instances of clauses that participate in an inference. Moreover, clauses are not right away instantiated into ground clauses. Rather they are instantiated only as far as required for an inference. Inferences with non-ground clauses in general represent infinite sets of ground inferences which are computed simultaneously in a single step.

The *first-order resolution calculus* consists of the inference rules *Resolution* and *Factoring* and generalizes the propositional resolution calculus (Section 2.6). Variables in clauses are implicitely universally quantified, so they can be instantiated in an arbitrary way. For the application of any inference or reduction rule, I can therefore assume that the involved clauses don't share any variables, i.e., variables are a priori renamed. Furthermore, clauses are assumed to be unique with respect to renaming in a set.

Resolution $(N \uplus \{D \vee A, \neg B \vee C\}) \Rightarrow_{\mathrm{RES}} (N \cup \{D \vee A, \neg B \vee C\} \cup \{(D \vee C)\sigma\})$
if $\sigma = mgu(A, B)$ for atoms A, B

Factoring $(N \uplus \{C \vee L \vee K\}) \Rightarrow_{\mathrm{RES}} (N \cup \{C \vee L \vee K\} \cup \{(C \vee L)\sigma\})$
if $\sigma = mgu(L, K)$ for literals L, K

The reduction rules are

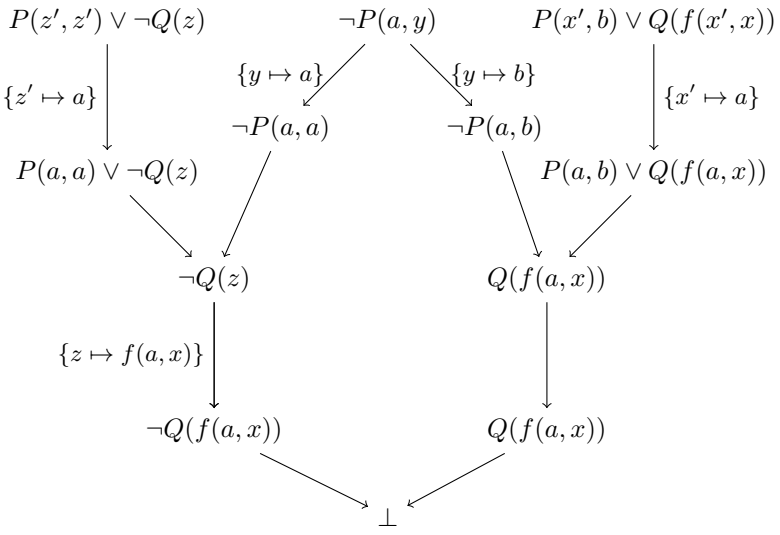

Figure 3.4: Instantiation of the clause set $N = P(z', z') \vee \neg Q(z), \neg P(a, y), P(x', b) \vee Q(f(x', x))$ with a reduced number of instantiations.

Subsumption $(N \uplus \{C_1, C_2\}) \Rightarrow_{\text{RES}} (N \cup \{C_1\})$
provided $C_1\sigma \subset C_2$ for some matcher σ

Tautology Deletion $(N \uplus \{C \vee A \vee \neg A\}) \Rightarrow_{\text{RES}} (N)$

Condensation $(N \uplus \{C\}) \Rightarrow_{\text{RES}} (N \cup \{C'\})$
where C' is the result of removing duplicate literals from $C\sigma$ for some matcher σ and C' subsumes C

Subsumption Resolution $(N \uplus \{C_1 \vee L, C_2 \vee K\}) \Rightarrow_{\text{RES}} (N \cup \{C_1 \vee L, C_2\})$
where $L\sigma = \text{comp}(K)$ and $C_1\sigma \subseteq C_2$

Lifting Lemma

Lemma 3.10.3. Let C and D be variable-disjoint clauses. If

Propositional Resolution $(N \uplus \{D\sigma, C\rho\}) \Rightarrow (N \cup \{D\sigma, C\rho\} \cup \{C'\})$
where σ and ρ are substitutions then there exists a substitution τ so that

General Resolution $(N \uplus \{D, C\}) \Rightarrow (N \cup \{D, C\} \cup \{C''\tau = C'\})$
An analogous lifting lemma holds for factorization.

Saturation of Sets of General Clauses

Definition 3.10.4 (Resolution Saturation)**.** A set of clauses N is saturated up to redundancy

Corollary 3.10.5. Let N be a set of general clauses saturated under Res, i.e., $\mathrm{Res}(N) \subseteq N$. Then also $G_\Sigma(N)$ is saturated, that is, $\mathrm{Res}(G_\Sigma(N)) \subseteq G_\Sigma(N)$.

Proof. W.l.o.g. assume that clauses in N are pairwise variable-disjoint. (Otherwise they have to be made disjoint and this renaming process changes neither $Res(N)$ nor $G_\Sigma(N)$.) Let $C' \in Res(G_\Sigma(N))$, meaning (i) there exist resolvable ground instances $D\sigma$ and $C\rho$ of N with resolvent C', or else (ii) C' is a factor of a ground instance $C\sigma$ of C.

Case (i): By the Lifting Lemma, D and C are resolvable with a resolvent C'' with $C''\tau = C'$, for a suitable substitution τ. As $C'' \in N$ by assumption, $C' \in G_\Sigma(N)$ is obtained.

Case (ii): Similar. □

Herbrand's Theorem

Lemma 3.10.6. Let N be a set of Σ-clauses, let $\mathcal{A}$ be an interpretation. Then $\mathcal{A} \models N$ implies $\mathcal{A} \models G_\Sigma(N)$.

Lemma 3.10.7. Let N be a set of Σ-clauses, let $\mathcal{A}$ be a *Herbrand* interpretation. Then $\mathcal{A} \models G_\Sigma(N)$ implies $\mathcal{A} \models N$.

Theorem 3.10.8 (Herbrand)**.** A set N of Σ-clauses is satisfiable if and only if it has a Herbrand model over Σ.

Proof. ($\Leftarrow$) Assume N has a Herbrand model I over Σ, i.e., $I \models N$. Then N is satisfiable.

($\Rightarrow$) Let $N \not\models \bot$.

$$
\begin{array}{lcll}
N \not\models \bot & \Rightarrow & \bot \notin Res^*(N) & \text{(resolution is sound)} \\
 & \Rightarrow & \bot \notin G_\Sigma(Res^*(N)) & \\
 & \Rightarrow & I_{G_\Sigma(Res^*(N))} \models G_\Sigma(Res^*(N)) & \text{(Theorem ; Corollary 3.10.5)} \\
 & \Rightarrow & I_{G_\Sigma(Res^*(N))} \models Res^*(N) & \text{(Lemma 3.10.7)} \\
 & \Rightarrow & I_{G_\Sigma(Res^*(N))} \models N & (N \subseteq Res^*(N))
\end{array}
$$

□

The Theorem of Löwenheim-Skolem

Theorem 3.10.9 (Löwenheim–Skolem)**.** Let Σ be a countable signature and let S be a set of closed Σ-formulas. Then S is satisfiable iff S has a model over a countable universe.

Proof. If both X and Σ are countable, then S can be at most countably infinite. Now generate, maintaining satisfiability, a set N of clauses from S. This extends Σ by at most countably many new Skolem functions to Σ'. As Σ' is countable, so is $T_{\Sigma'}$, the universe of Herbrand-interpretations over Σ'. Now apply Theorem 3.10.8. □

Refutational Completeness of General Resolution

Theorem 3.10.10 (Soundness and Completenss of Resolution)**.** The resolution calculus is sound and complete:

$$N \text{ is unsatisfiable iff } N \Rightarrow^*_{\text{RES}} N' \text{ and } \bot \in N' \text{ for some } N'$$

Theorem 3.10.11 (Soundness and Completenss of Resolution)**.** Let N be a set of first-clauses where $Res(N) \subseteq N$. Then

$$N \models \bot \Leftrightarrow \bot \in N.$$

Proof. Let $Res(N) \subseteq N$. By Corollary 3.10.5: $Res(G_\Sigma(N)) \subseteq G_\Sigma(N)$

$$\begin{array}{lll} N \models \bot & \Leftrightarrow G_\Sigma(N) \models \bot & \text{(Lemma 3.10.6/3.10.7; Theorem 3.10.8)} \\ & \Leftrightarrow \bot \in G_\Sigma(N) & \text{(propositional resolution sound and complete)} \\ & \Leftrightarrow \bot \in N & \end{array}$$

□

Compactness of First-Order Logic

Theorem 3.10.12 (Compactness Theorem for First-Order Logic)**.** Let S be a set of first-order formulas. S is unsatisfiable if and only if some finite subset $S' \subseteq S$ is unsatisfiable.

Proof. ($\Leftarrow$) Assume that S' is unsatisfiable. Then there exists at least one unsatisfiable formula $\phi \in S'$. Since $S' \subseteq S$, S is also unsatisfiable.

($\Rightarrow$) Let S be unsatisfiable and let N be the set of clauses obtained by Skolemization and CNF transformation of the formulas in S. Clearly $Res^*(N)$ is unsatisfiable. By Theorem 3.10.11, $\bot \in Res^*(N)$, and therefore $\bot \in Res^n(N)$ for some $n \in \mathbb{N}$. Consequently, $\bot$ has a finite resolution proof B of depth$\leq n$. Choose S' as the subset of formulas in S so that the corresponding clauses contain the assumptions (leaves) of B. □

3.11 Orderings

Propositional superposition is based on an ordering on the propositional variables, Section 2.7. The ordering is total and well-founded. Basically, propositional variables correspond to ground atoms in first-order logic. This section generalizes the ideas of the propositional superposition ordering to first-order logic. In first-order logic the ordering has to also consider terms and variables and operations on terms like the application of a substitution.

Definition 3.11.1 (Σ-Operation Compatible Relation)**.** A binary relation $\sqsupset$ over $T(\Sigma, \mathcal{X})$ is called *compatible with Σ-operations,* if $s \sqsupset s'$ implies $f(t_1, \ldots, s, \ldots, t_n) \sqsupset f(t_1, \ldots, s', \ldots, t_n)$ for all $f \in \Omega$ and $s, s', t_i \in T(\Sigma, \mathcal{X})$.

Lemma 3.11.2 (Σ-Operation Compatible Relation)**.** A relation $\sqsupset$ is compatible with Σ-operations iff $s \sqsupset s'$ implies $t[s]_p \sqsupset t[s']_p$ for all $s, s', t \in T(\Sigma, \mathcal{X})$ and $p \in pos(t)$.

In the literature *compatible with Σ-operations* is sometimes also called *compatible with contexts*.

Definition 3.11.3 (Substitution Stable Relation, Rewrite Relation)**.** A binary relation $\sqsupset$ over $T(\Sigma, \mathcal{X})$ is called *stable under substitutions*, if $s \sqsupset s'$ implies $s\sigma \sqsupset s'\sigma$ for all $s, s' \in T(\Sigma, \mathcal{X})$ and substitutions σ. A binary relation $\sqsupset$ is called a *rewrite relation*, if it is compatible with Σ-operations and stable under substitutions.

A *rewrite ordering* is then an ordering that is a rewrite relation.

Definition 3.11.4 (Subterm Ordering)**.** The *proper subterm ordering* $s > t$ is defined by $s > t$ iff $s|_p = t$ for some position $p \neq \epsilon$ of s.

Definition 3.11.5 (Simplification Ordering)**.** A rewrite ordering $\succ$ over $T(\Sigma, \mathcal{X})$ is called *simplification ordering*, if it enjoys the *subterm property* $s \succ t$ implies $s > t$ for all $s, t \in T(\Sigma, \mathcal{X})$ of the same sort.

Definition 3.11.6 (Lexicographical Path Ordering (LPO))**.** Let $\Sigma = (\mathcal{S}, \Omega, \Pi)$ be a signature and let $\succ$ be a strict partial ordering on operator symbols in Ω, called *precedence*. The *lexicographical path ordering* $\succ_{lpo}$ on $T(\Sigma, \mathcal{X})$ is defined as follows: if s, t are terms in $T_S(\Sigma, \mathcal{X})$ then $s \succ_{lpo} t$ iff

1. $t = x \in \mathcal{X}$, $x \in vars(s)$ and $s \neq t$ or
2. $s = f(s_1, \ldots, s_n)$, $t = g(t_1, \ldots, t_m)$ and
   (a) $s_i \succeq_{lpo} t$ for some $i \in \{1, \ldots, n\}$ or
   (b) $f \succ g$ and $s \succ_{lpo} t_j$ for every $j \in \{1, \ldots, m\}$ or
   (c) $f = g$, $s \succ_{lpo} t_j$ for every $j \in \{1, \ldots, m\}$ and $(s_1, \ldots, s_n)(\succ_{lpo})_{lex}(t_1, \ldots, t_m)$.

Theorem 3.11.7 (LPO Properties)**.**

1. The LPO is a rewrite ordering.
2. LPO enjoys the subterm property, hence is a simplification ordering.
3. If the precedence $\succ$ is total on Ω then $\succ_{lpo}$ is total on the set of ground terms $T(\Sigma)$.
4. If Ω is finite then $\succ_{lpo}$ is well-founded.

Example 3.11.8. Consider the terms $g(x)$, $g(y)$, $g(g(a))$, $g(b)$, $g(a)$, b, a. With respect to the precedence $g \succ b \succ a$ the ordering on the ground terms is $g(g(a)) \succ_{lpo} g(b) \succ_{lpo} g(a) \succ_{lpo} b \succ_{lpo} a$. The terms $g(x)$ and $g(y)$ are not comparable. Note that the terms $g(g(a))$, $g(b)$, $g(a)$ are all instances of both $g(x)$ and $g(y)$.

With respect to the precedence $b \succ a \succ g$ the ordering on the ground terms is $g(b) \succ_{lpo} b \succ_{lpo} g(g(a)) \succ_{lpo} g(a) \succ_{lpo} a$.

Definition 3.11.9 (The Knuth-Bendix Ordering)**.** Let $\Sigma = (\mathcal{S}, \Omega, \Pi)$ be a finite signature, let $\succ$ be a strict partial ordering *("precedence")* on Ω, let $w : \Omega \cup \mathcal{X} \to \mathbb{R}^+$ be a *weight function*, so that the following admissibility condition is satisfied:
$w(x) = w_0 \in \mathbb{R}^+$ for all variables $x \in \mathcal{X}$; $w(c) \geq w_0$ for all constants $c \in \Omega$.

Then, the weight function w can be extended to terms recursively:

$$w(f(t_1, \ldots, t_n)) = w(f) + \sum_{1 \leq i \leq n} w(t_i)$$

or alternatively

$$\sum w(t) = \sum_{x \in vars(t)} w(x) \cdot \#(x, t) + \sum_{f \in \Omega} w(f) \cdot \#(f, t)$$

where $\#(a, t)$ is the number of occurrences of a in t.

The *Knuth-Bendix ordering* $\succ_{kbo}$ on $T(\Sigma, \mathcal{X})$ induced by $\succ$ and admissible w is defined by: $s \succ_{kbo} t$ iff

1. $\#(x, s) \geq \#(x, t)$ for all variables x and $w(s) > w(t)$, or
2. $\#(x, s) \geq \#(x, t)$ for all variables x, $w(s) = w(t)$, and
   (a) $s = f(s_1, \ldots, s_m)$, $t = g(t_1, \ldots, t_n)$, and $f \succ g$, or
   (b) $s = f(s_1, \ldots, s_m)$, $t = f(t_1, \ldots, t_m)$, and $(s_1, \ldots, s_m)(\succ_{kbo})_{lex}(t_1, \ldots, t_m)$.

Theorem 3.11.10 (KBO Properties)**.**

1. The KBO is a rewrite ordering.
2. KBO enjoys the subterm property, hence is a simplification ordering.
3. If the precedence $\succ$ is total on Ω then $\succ_{kbo}$ is total on the set of ground terms $T(\Sigma)$.
4. If Ω is finite then $\succ_{kbo}$ is well-founded.

The KBO ordering can be extended to contain unary function symbols with weight zero. This was motivated by completion of the group axioms, see Chapter 4.

Definition 3.11.11 (The Knuth-Bendix Ordering Extended)**.** The additional requirements added to Definition 3.11.9 are

1. Extend w to $w : \Omega \cup \mathcal{X} \to \mathbb{R}_0^+$

2. If $w(f) = 0$ for some $f \in \Omega$ with arity$(f) = 1$, then $f \succeq g$ for all $g \in \Omega$.

3. As a first case to the disjunction of 3.11.9-2.
   (a') $t = x$, $s = f^n(x)$ for some $n \geq 1$

The LPO ordering as well as the KBO ordering can be extended to atoms in a straightforward way. The precedence $\succ$ is extended to Π. For LPO atoms are then compared according to Definition 3.11.6-2. For KBO the weight function w is also extended to atoms by giving predicates a non-zero positive weight and then atoms are compared according to terms.

Actually, since atoms are never substituted for variables in first-order logic, an alternative to the above would be to first compare the predicate symbols and let $\succ$ decide the ordering. Only if the atoms share the same predicate symbol, the argument terms are considered, e.g., in a lexicographic way and are then compared with respect to KBO or LPO, respectively.

3.12 First-Order Ground Superposition

Propositional clauses and ground clauses are essentially the same, as long as equational atoms are not considered. This section deals only with ground clauses and recalls mostly the material from Section 2.7 for first-order ground clauses. The main difference is that the atom ordering is more complicated, see Section 3.11. Let N be a possibly infinite set of ground clauses.

Definition 3.12.1 (Ground Clause Ordering)**.** Let $\prec$ be a strict rewrite ordering total on ground terms and ground atoms. Then $\prec$ can be lifted to a total ordering $\prec_L$ on literals by its multiset extension $\prec_{\text{mul}}$ where a positive literal $P(t_1, \ldots, t_n)$ is mapped to the multiset $\{P(t_1, \ldots, t_n)\}$ and a negative literal $\neg P(t_1, \ldots, t_n)$ to the multiset $\{P(t_1, \ldots, t_n), P(t_1, \ldots, t_n)\}$. The ordering $\prec_L$ is further lifted to a total ordering on clauses $\prec_C$ by considering the multiset extension of $\prec_L$ for clauses.

Proposition 3.12.2 (Properties of the Ground Clause Ordering)**.**

1. The orderings on literals and clauses are total and well-founded.

2. Let C and D be clauses with $P(t_1, \ldots, t_n) = \text{atom}(\max(C))$, $Q(s_1, \ldots, s_m) = \text{atom}(\max(D))$, where $\max(C)$ denotes the maximal literal in C.

   (a) If $Q(s_1, \ldots, s_m) \prec_L P(t_1, \ldots, t_n)$ then $D \prec_C C$.

   (b) If $P(t_1, \ldots, t_n) = Q(s_1, \ldots, s_m)$, $P(t_1, \ldots, t_n)$ occurs negatively in C but only positively in D, then $D \prec_C C$.

Eventually, as I did for propositional logic, I overload $\prec$ with $\prec_L$ and $\prec_C$. So if $\prec$ is applied to literals it denotes $\prec_L$, if it is applied to clauses, it denotes $\prec_C$. Note that $\prec$ is a total ordering on literals and clauses as well. For superposition, inferences are restricted to maximal literals with respect to $\prec$. For a clause set N, I define $N^{\prec C} = \{D \in N \mid D \prec C\}$.

Definition 3.12.3 (Abstract Redundancy)**.** A ground clause C is *redundant* with respect to a set of ground clauses N if $N^{\prec C} \models C$.

Tautologies are redundant. Subsumed clauses are redundant if $\subseteq$ is strict. Duplicate clauses are anyway eliminated quietly because the calculus operates on sets of clauses.

Note that for finite N, and any $C \in N$ redundancy $N^{\prec C} \models C$ can be decided but is as hard as testing unsatisfiability for a clause set N. So the goal is to invent redundancy notions that can be efficiently decided and that are useful.

Definition 3.12.4 (Selection Function)**.** The selection function sel maps clauses to one of its negative literals or $\bot$. If $\mathrm{sel}(C) = \neg P(t_1, \ldots, t_n)$ then $\neg P(t_1, \ldots, t_n)$ is called *selected* in C. If $\mathrm{sel}(C) = \bot$ then no literal in C is *selected*.

The selection function is, in addition to the ordering, a further means to restrict superposition inferences. If a negative literal is selected in a clause, any superposition inference must be on the selected literal.

Definition 3.12.5 (Partial Model Construction)**.** Given a clause set N and an ordering $\prec$ we can construct a (partial) model $N_{\mathcal{I}}$ for N inductively as follows:

$$
\begin{array}{rcl}
N_C & := & \bigcup_{D \prec C} \delta_D \\
\delta_D & := & \begin{cases} \{P(t_1, \ldots, t_n)\} & \text{if } D = D' \vee P(t_1, \ldots, t_n), P(t_1, \ldots, t_n) \text{ strictly maximal, no literal} \\ & & \text{selected in } D \text{ and } N_D \not\models D \\ \emptyset \qquad \text{otherwise} & & \end{cases} \\
N_{\mathcal{I}} & := & \bigcup_{C \in N} \delta_C
\end{array}
$$

Clauses C with $\delta_C \neq \emptyset$ are called *productive*.

Proposition 3.12.6 (Propertied of the Model Operator)**.** Some properties of the partial model construction.

1. For every D with $(C \vee \neg P(t_1, \ldots, t_n)) \prec D$ we have $\delta_D \neq \{P(t_1, \ldots, t_n)\}$.
2. If $\delta_C = \{P(t_1, \ldots, t_n)\}$ then $N_C \cup \delta_C \models C$.
3. If $N_C \models D$ and $D \prec C$ then for all C' with $C \prec C'$ we have $N_{C'} \models D$ and in particular $N_{\mathcal{I}} \models D$.
4. There is no clause C with $P(t_1, \ldots, t_n) \vee P(t_1, \ldots, t_n) \prec C$ such that $\delta_C = \{P(t_1, \ldots, t_n)\}$.

T Please properly distinguish: N is a set of clauses interpreted as the conjunction of all clauses. $N^{\prec C}$ is of set of clauses from N strictly smaller than C with respect to $\prec$. $N_{\mathcal{I}}$, N_C are Herbrand interpretations (see Proposition 3.5.3). $N_{\mathcal{I}}$ is the overall (partial) model for N, whereas N_C is generated from all clauses from N strictly smaller than C.

Superposition Left $(N \uplus \{C_1 \vee P(t_1, \ldots, t_n), C_2 \vee \neg P(t_1, \ldots, t_n)\}) \Rightarrow_{\text{SUP}} (N \cup \{C_1 \vee P(t_1, \ldots, t_n), C_2 \vee \neg P(t_1, \ldots, t_n)\} \cup \{C_1 \vee C_2\})$

where (i) $P(t_1, \ldots, t_n)$ is strictly maximal in $C_1 \vee P(t_1, \ldots, t_n)$ (ii) no literal in $C_1 \vee P(t_1, \ldots, t_n)$ is selected (iii) $\neg P(t_1, \ldots, t_n)$ is maximal and no literal selected in $C_2 \vee \neg P(t_1, \ldots, t_n)$, or $\neg P(t_1, \ldots, t_n)$ is selected in $C_2 \vee \neg P(t_1, \ldots, t_n)$

Factoring $(N \uplus \{C \vee P(t_1, \ldots, t_n) \vee P(t_1, \ldots, t_n)\}) \Rightarrow_{\text{SUP}} (N \cup \{C \vee P(t_1, \ldots, t_n) \vee P(t_1, \ldots, t_n)\} \cup \{C \vee P(t_1, \ldots, t_n)\})$

where (i) $P(t_1, \ldots, t_n)$ is maximal in $C \vee P(t_1, \ldots, t_n) \vee P(t_1, \ldots, t_n)$ (ii) no literal is selected in $C \vee P(t_1, \ldots, t_n) \vee P(t_1, \ldots, t_n)$

Note that the superposition factoring rule differs from the resolution factoring rule in that it only applies to positive literals.

Definition 3.12.7 (Saturation)**.** A set N of clauses is called *saturated up to redundancy*, if any inference from non-redundant clauses in N yields a redundant clause with respect to N or is contained in N.

Examples for specific redundancy rules that can be efficiently decided are

Subsumption $(N \uplus \{C_1, C_2\}) \Rightarrow_{\text{SUP}} (N \cup \{C_1\})$

provided $C_1 \subset C_2$

Tautology Deletion $(N \uplus \{C \vee P(t_1, \ldots, t_n) \vee \neg P(t_1, \ldots, t_n)\}) \Rightarrow_{\text{SUP}} (N)$

Condensation $(N \uplus \{C_1 \vee L \vee L\}) \Rightarrow_{\text{SUP}} (N \cup \{C_1 \vee L\})$

Subsumption Resolution $(N \uplus \{C_1 \vee L, C_2 \vee \neg L\}) \Rightarrow_{\text{SUP}} (N \cup \{C_1 \vee L, C_2\})$

where $C_1 \subseteq C_2$

Proposition 3.12.8 (Completeness of the Reduction Rules)**.** All clauses removed by Subsumption, Tautology Deletion, Condensation and Subsumption Resolution are redundant with respect to the kept or added clauses.

Theorem 3.12.9 (Completeness)**.** Let N be a, possibly countably infinite, set of ground clauses. If N is saturated up to redundancy and $\bot \notin N$ then N is satisfiable and $N_{\mathcal{I}} \models N$.

Proof. The proof is by contradiction. So I assume: (i) for any clause D derived by Superposition Left or Factoring from N that D is redundant, i.e., $N^{\prec D} \models D$, (ii) $\bot \notin N$ and (iii) $N_{\mathcal{I}} \not\models N$. Then there is a minimal, with respect to $\prec$, clause $C \vee L \in N$ such that $N_{\mathcal{I}} \not\models C \vee L$ and L is a selected literal in $C \vee L$ or no literal in $C \vee L$ is selected and L is maximal. This clause must exist because $\bot \notin N$.

The clause $C \vee L$ is not redundant. For otherwise, $N^{\prec C \vee L} \models C \vee L$ and hence $N_{\mathcal{I}} \models C \vee L$, because $N_{\mathcal{I}} \models N^{\prec C \vee L}$, a contradiction.

I distinguish the case L is a positive and no literal selected in $C \vee L$ or L is a negative literal. Firstly, assume L is positive, i.e., $L = P(t_1, \ldots, t_n)$ for some ground atom $P(t_1, \ldots, t_n)$. Now if $P(t_1, \ldots, t_n)$ is strictly maximal in $C \vee P(t_1, \ldots, t_n)$ then actually $\delta_{C \vee P} = \{P(t_1, \ldots, t_n)\}$ and hence $N_{\mathcal{I}} \models C \vee P$, a contradiction. So $P(t_1, \ldots, t_n)$ is not strictly maximal. But then actually $C \vee P(t_1, \ldots, t_n)$ has the form $C'_1 \vee P(t_1, \ldots, t_n) \vee P(t_1, \ldots, t_n)$ and Factoring derives $C'_1 \vee P(t_1, \ldots, t_n)$ where $(C'_1 \vee P(t_1, \ldots, t_n)) \prec (C'_1 \vee P(t_1, \ldots, t_n) \vee P(t_1, \ldots, t_n))$. Now $C'_1 \vee P(t_1, \ldots, t_n)$ is not redundant, strictly smaller than $C \vee L$, we have $C'_1 \vee P(t_1, \ldots, t_n) \in N$ and $N_{\mathcal{I}} \not\models C'_1 \vee P(t_1, \ldots, t_n)$, a contradiction against the choice that $C \vee L$ is minimal.

Secondly, let us assume L is negative, i.e., $L = \neg P(t_1, \ldots, t_n)$ for some ground atom $P(t_1, \ldots, t_n)$. Then, since $N_{\mathcal{I}} \not\models C \vee \neg P(t_1, \ldots, t_n)$ we know $P(t_1, \ldots, t_n) \in N_{\mathcal{I}}$. So there is a clause $D \vee P(t_1, \ldots, t_n) \in N$ where $\delta_{D \vee P(t_1, \ldots, t_n)} = \{P(t_1, \ldots, t_n)\}$ and $P(t_1, \ldots, t_n)$ is strictly maximal in $D \vee P(t_1, \ldots, t_n)$ and $(D \vee P(t_1, \ldots, t_n)) \prec (C \vee \neg P(t_1, \ldots, t_n))$. So Superposition Left derives $C \vee D$ where $(C \vee D) \prec (C \vee \neg P(t_1, \ldots, t_n))$. The derived clause $C \vee D$ cannot be redundant, because for otherwise either $N^{\prec D \vee P(t_1, \ldots, t_n)} \models D \vee P(t_1, \ldots, t_n)$ or $N^{\prec C \vee \neg P(t_1, \ldots, t_n)} \models C \vee \neg P(t_1, \ldots, t_n)$. So $C \vee D \in N$ and $N_{\mathcal{I}} \not\models C \vee D$, a contradiction against the choice that $C \vee L$ is the minimal false clause. □

So the proof actually tells us that at any point in time we need only to consider either a superposition left inference between a minimal false clause and a productive clause or a factoring inference on a minimal false clause.

Theorem 3.12.10 (Compactness of First-Order Logic)**.** Let N be a, possibly countably infinite, set of first-order logic ground clauses. Then N is unsatisfiable iff there is a finite subset $N' \subseteq N$ such that N' is unsatisfiable.

Proof. If N is unsatisfiable, saturation via superposition generates $\bot$. So there is an i such that $N \Rightarrow^{i}_{\text{SUP}} N'$ and $\bot \in N'$. The clause $\bot$ is the result of at most i-many superposition inferences, reductions on clauses $\{C_1, \ldots, C_n\} \subseteq N$. Superposition is sound, so $\{C_1, \ldots, C_n\}$ is a finite, unsatisfiable subset of N. □

Corollary 3.12.11 (Compactness of First-Order Logic: Classical)**.** A set N of clauses is satisfiable iff all finite subsets of N are satisfiable.

Theorem 3.12.12 (Soundness and Completeness of Ground Superposition)**.** A first-order Σ-sentence ϕ is valid iff there exists a ground superposition refutation for $\text{ground}(\Sigma, \text{cnf}(\neg\phi))$.

Proof. A first-order sentence ϕ is valid iff $\neg\phi$ is unsatisfiable iff $\text{acnf}(\neg\phi)$ is unsatisfiable iff $\text{ground}(\Sigma, \text{cnf}(\neg\phi))$ is unsatisfiable iff superposition provides a refutation of $\text{ground}(\Sigma, \text{cnf}(\neg\phi))$. □

Theorem 3.12.13 (Semi-Decidability of First-Order Logic by Ground Superposition)**.** If a first-order Σ-sentence ϕ is valid then a ground superposition refutation can be computed.

Proof. In a fair way enumerate $\text{ground}(\Sigma, \text{acnf}(\neg\phi))$ and perform superposition inference steps. The enumeration can, e.g., be done by considering Herbrand terms of increasing size. □

Example 3.12.14 (Ground Superposition)**.** Consider the below clauses 1-4 and superposition refutation with respect a KBO with precedence $P \succ Q \succ g \succ f \succ c \succ b \succ a$ where the weight function w returns 1 for all signature symbols. Maximal literals are marked with a *.

1.	$\neg P(f(c))^* \vee \neg P(f(c))^* \vee Q(b)$	(Input)
2.	$P(f(c))^* \vee Q(b)$	(Input)
3.	$\neg P(g(b,c))^* \vee \neg Q(b)$	(Input)
4.	$P(g(b,c))^*$	(Input)
5.	$\neg P(f(c))^* \vee Q(b)$	(Cond(1))
6.	$Q(b)^* \vee Q(b)^*$	(Sup(5, 2)))
7.	$Q(b)^*$	(Fact(6))
8.	$\neg Q(b)^*$	(Sup(3, 4))
10.	$\bot$	(Sup(8, 7))

Note that clause 5 cannot be derived by Factoring whereas clause 7 can also be derived by Condensation. Clause 8 is also the result of a Subsumption Resolution application to clauses 3, 4.

Theorem 3.12.15 (Craig Theorem [15])**.** Let ϕ and ψ be two propositional (first-order ground) formulas so that $\phi \models \psi$. Then there exists a formula χ (called the *interpolant* for $\phi \models \psi$), so that χ contains only propositional variables (first-order signature symbols) occurring both in ϕ and in ψ so that $\phi \models \chi$ and $\chi \models \psi$.

Proof. Translate ϕ and $\neg\psi$ into CNF. let N and M, respectively, denote the resulting clause set. Choose an atom ordering $\succ$ for which the propositional variables that occur in ϕ but not in ψ are maximal. Saturate N into N^* w.r.t. $Sup^{\succ}_{sel}$ with an empty selection function sel. Then saturate $N^* \cup M$ w.r.t. $Sup^{\succ}_{sel}$ to derive $\bot$. As N^* is already saturated, due to the ordering restrictions only inferences need to be considered where premises, if they are from N^*, only contain symbols that also occur in ψ. The conjunction of these premises is an interpolant χ. The theorem also holds for first-order formulas. For universal formulas the above proof can be easily extended. In the general case, a proof based on superposition technology is more complicated because of Skolemization. □

3.13 First-Order Superposition

Now the result for ground superposition are lifted to superposition on first-order clauses with variables, still without equality. The completeness proof of ground superposition above talks about (strictly) maximal literals of *ground* clauses. The non-ground calculus considers those literals that correspond to (strictly) maximal literals of ground instances.

The used ordering is exactly the ordering of Definition 3.12.1 where clauses with variables are projected to their ground instances for ordering computations.

Definition 3.13.1 (Maximal Literal)**.** A literal L is called *maximal* in a clause C if and only if there exists a grounding substitution σ so that $L\sigma$ is maximal in $C\sigma$, i.e., there is no different $L' \in C$: $L\sigma \prec L'\sigma$. The literal L is called *strictly maximal* if there is no different $L' \in C$ such that $L\sigma \preceq L'\sigma$.

Note that the orderings KBO and LPO cannot be total on atoms with variables, because they are stable under substitutions. Therefore, maximality can also be defined on the basis of absence of greater literals. A literal L is called *maximal* in a clause C if $L \not\prec L'$ for all other literals $L' \in C$. It is called *strictly maximal* in a clause C if $L \not\preceq L'$ for all other literals $L' \in C$.

Superposition Left $(N \uplus \{C_1 \vee P(t_1, \ldots, t_n), C_2 \vee \neg P(s_1, \ldots, s_n)\}) \Rightarrow_{\text{SUP}}$
$(N \cup \{C_1 \vee P(t_1, \ldots, t_n), C_2 \vee \neg P(s_1, \ldots, s_n)\} \cup \{(C_1 \vee C_2)\sigma\})$
where (i) $P(t_1, \ldots, t_n)\sigma$ is strictly maximal in $(C_1 \vee P(t_1, \ldots, t_n))\sigma$ (ii) no literal in $C_1 \vee P(t_1, \ldots, t_n)$ is selected (iii) $\neg P(s_1, \ldots, s_n)\sigma$ is maximal and no literal selected in $(C_2 \vee \neg P(s_1, \ldots, s_n))\sigma$, or $\neg P(s_1, \ldots, s_n)$ is selected in $(C_2 \vee \neg P(s_1, \ldots, s_n))\sigma$ (iv) σ is the mgu of $P(t_1, \ldots, t_n)$ and $P(s_1, \ldots, s_n)$

Factoring $(N \uplus \{C \vee P(t_1, \ldots, t_n) \vee P(s_1, \ldots, s_n)\}) \Rightarrow_{\text{SUP}}$
$(N \cup \{C \vee P(t_1, \ldots, t_n) \vee P(s_1, \ldots, s_n)\} \cup \{(C \vee P(t_1, \ldots, t_n))\sigma\})$
where (i) $P(t_1, \ldots, t_n)\sigma$ is maximal in $(C \vee P(t_1, \ldots, t_n) \vee P(s_1, \ldots, s_n))\sigma$ (ii) no literal is selected in $C \vee P(t_1, \ldots, t_n) \vee P(s_1, \ldots, s_n)$ (iii) σ is the mgu of $P(t_1, \ldots, t_n)$ and $P(s_1, \ldots, s_n)$

Note that the above inference rules Superposition Left and Factoring are generalizations of their respective counterparts from the ground superposition calculus above. Therefore, on ground clauses they coincide. Therefore, we can safely overload them in the sequel.

Definition 3.13.2 (Abstract Redundancy)**.** A clause C is *redundant* with respect to a clause set N if for all ground instances $C\sigma$ there are clauses $\{C_1, \ldots, C_n\} \subseteq N$ with ground instances $C_1\tau_1, \ldots, C_n\tau_n$ such that $C_i\tau_i \prec C\sigma$ for all i and $C_1\tau_1, \ldots, C_n\tau_n \models C\sigma$.

Definition 3.13.3 (Saturation)**.** A set N of clauses is called *saturated up to redundancy*, if any inference from non-redundant clauses in N yields a redundant clause with respect to N or is contained in N.

In contrast to the ground case, the above abstract notion of redundancy is not effective, i.e., it is undecidable for some clause C whether it is redundant, in general. Nevertheless, the concrete ground redundancy notions carry over to the non-ground case. Note also that a clause C is contained in N modulo renaming of variables.

Let rdup be a function from clauses to clauses that removes duplicate literals, i.e., $\mathrm{rdup}(C) = C'$ where $C' \subseteq C$, C' does not contain any duplicate literals, and for each $L \in C$ also $L \in C'$.

Subsumption $(N \uplus \{C_1, C_2\}) \Rightarrow_{\mathrm{SUP}} (N \cup \{C_1\})$
provided $C_1\sigma \subset C_2$ for some σ

Tautology Deletion $(N \uplus \{C \vee P(t_1, \ldots, t_n) \vee \neg P(t_1, \ldots, t_n)\}) \Rightarrow_{\mathrm{SUP}} (N)$

Condensation $(N \uplus \{C_1 \vee L \vee L'\}) \Rightarrow_{\mathrm{SUP}} (N \cup \{\mathrm{rdup}((C_1 \vee L \vee L')\sigma)\})$
provided $L\sigma = L'$ and $\mathrm{rdup}((C_1 \vee L \vee L')\sigma)$ subsumes $C_1 \vee L \vee L'$ for some σ

Subsumption Resolution $(N \uplus \{C_1 \vee L, C_2 \vee L'\}) \Rightarrow_{\mathrm{SUP}} (N \cup \{C_1 \vee L, C_2\})$
where $L\sigma = \neg L'$ and $C_1\sigma \subseteq C_2$ for some σ

Lemma 3.13.4. All reduction rules are instances of the abstract redundancy criterion.

Proof. Do it □

Lemma 3.13.5 (Subsumption is NP-complete)**.** Subsumption is NP-complete.

Proof. Let C_1 subsume C_2 with substitution σ Subsumption is in NP because the size of σ is bounded by the size of C_2 and the subset relation can be checked in time at most quadratic in the size of C_1 and C_2.

Propositional SAT can be reduced as follows. Assume a 3-SAT clause set N. Consider a 3-place predicate R and a unary function g and a mapping from propositional variables P to first order variables x_P. □

Lemma 3.13.6 (Lifting)**.** Let $D \vee L$ and $C \vee L'$ be variable-disjoint clauses and σ a grounding substitution for $C \vee L$ and $D \vee L'$. If there is a superposition left inference

$$(N \uplus \{(D \vee L)\sigma, (C \vee L')\sigma\}) \Rightarrow_{\mathrm{SUP}} (N \cup \{(D \vee L)\sigma, (C \vee L')\sigma\} \cup \{D\sigma \vee C\sigma\})$$

and if $\mathrm{sel}((D \vee L)\sigma) = \mathrm{sel}((D \vee L))\sigma$, $\mathrm{sel}((C \vee L')\sigma) = \mathrm{sel}((C \vee L'))\sigma$, then there exists a mgu τ such that

$$(N \uplus \{D \vee L, C \vee L'\}) \Rightarrow_{\mathrm{SUP}} (N \cup \{D \vee L, C \vee L'\} \cup \{(D \vee C)\tau\}).$$

Let $C \vee L \vee L'$ be a clause and σ a grounding substitution for $C \vee L \vee L'$. If there is a factoring inference

$$(N \uplus \{(C \vee L \vee L')\sigma\}) \Rightarrow_{\mathrm{SUP}} (N \cup \{(C \vee L \vee L')\sigma\} \cup \{(C \vee L)\sigma\})$$

and if $\mathrm{sel}((C \vee L \vee L')\sigma) = \mathrm{sel}((C \vee L \vee L'))\sigma$, then there exists a mgu τ such that

$$(N \uplus \{C \vee L \vee L'\}) \Rightarrow_{\mathrm{SUP}} (N \cup \{C \vee L \vee L'\} \cup \{(C \vee L)\tau\})$$

Note that in the above lemma the clause $D\sigma \vee C\sigma$ is an instance of the clause $(D \vee C)\tau$. The reduction rules cannot be lifted in the same way as the following example shows.

Example 3.13.7 (First-Order Reductions are not Liftable)**.** Consider the two clauses $P(x) \vee Q(x)$, $P(g(y))$ and grounding substitution $\{x \mapsto g(a), y \mapsto a\}$. Then $P(g(y))\sigma$ subsumes $(P(x) \vee Q(x))\sigma$ but $P(g(y))$ does not subsume $P(x) \vee Q(x)$. For all other reduction rules similar examples can be constructed.

Lemma 3.13.8 (Soundness and Completeness)**.** First-Order Superposition is sound and complete.

Proof. Soundness is obvious. For completeness, Theorem 3.12.12 proves the ground case. Now by applying Lemma 3.13.6 to this proof it can be lifted to the first-order level, as argued in the following.

Let N be a an unsatisfiable set of first-order clauses. By Theorem 3.5.5 and Lemma 3.6.10 there exist a finite unsatisfiable set N' of ground instances from clauses from N such that for each clause $C\sigma \in N'$ there is a clause $C \in N$. Now ground superposition is complete, Theorem 3.12.12, so there exists a derivation of the empty clause by ground superposition from N': $N' = N'_0 \Rightarrow_{\mathrm{SUP}} \ldots \Rightarrow_{\mathrm{SUP}} N'_k$ and $\bot \in N'_k$. Now by an inductive argument on the length of the derivation k this derivation can be lifted to the first-order level. The invariant is: for any ground clause $C\sigma \in N'_i$ used in the ground proof, there is a clause $C \in N_i$ on the first-order level. The induction base holds for N and N' by construction. For the induction step Lemma 3.13.6 delivers the result. □

There are questions left open by Lemma 3.13.8. It just says that a ground refutation can be lifted to a first-order refutation. But what about abstract redundancy, Definition 3.13.2? Can first-order redundant clauses be deleted without harming completeness? And what about the ground model operator with respect to clause sets N saturated on the first-order level. Is in this case $\mathrm{ground}(\Sigma, N)_{\mathcal{I}}$ a model? The next two lemmas answer these questions positively.

Lemma 3.13.9 (Redundant Clauses are Obsolete)**.** If a clause set N is unsatisfiable, then there is a derivation $N \Rightarrow^*_{\mathrm{SUP}} N'$ such that $\bot \in N'$ and no clause in the derivation of $\bot$ is redundant.

Proof. If N is unsatisfiable then there is a ground superposition refutation of $\mathrm{ground}(\Sigma, N)$ such that no ground clause in the refutation is redundant. Now according to Lemma 3.13.8 this proof can be lifted to the first-order level. Now

assume some clause C in the first-order proof is redundant that is the lifting of some clause $C\sigma$ from the ground proof with respect to a grounding substitution σ. The clause C is redundant by Definition 3.13.2 if all its ground instances are, in particular, $C\sigma$. But this contradicts the fact that the lifted ground proof does not contain redundant clauses. □

Lemma 3.13.10 (Model Property)**.** If N is a saturated clause set and $\bot \notin N$ then $\text{ground}(\Sigma, N)_{\mathcal{I}} \models N$.

Proof. As usual we assume that selection on the ground and respective non-ground clauses is identical. Assume $\text{ground}(\Sigma, N)_{\mathcal{I}} \not\models N$. Then there is a minimal ground clause $C\sigma$, $C \neq \bot$, $C \in N$ such that $\text{ground}(\Sigma, N)_{\mathcal{I}} \not\models C\sigma$. Note that $C\sigma$ is not redundant as for otherwise $\text{ground}(\Sigma, N)_{\mathcal{I}} \models C\sigma$. So $\text{ground}(\Sigma, N)$ is not saturated. If $C\sigma$ is productive, i.e., $C\sigma = (C' \vee L)\sigma$ such that L is positive, $L\sigma$ strictly maximal in $(C' \vee L)\sigma$ then $L\sigma \in \text{ground}(\Sigma, N)_{\mathcal{I}}$ and hence $\text{ground}(\Sigma, N)_{\mathcal{I}} \models C\sigma$ contradicting $\text{ground}(\Sigma, N)_{\mathcal{I}} \not\models C\sigma$.

If $C\sigma = (C' \vee L \vee L')\sigma$ such that L is positive, $L\sigma$ maximal in $(C' \vee L \vee L')\sigma$ then, because N is saturated, there is a clause $(C' \vee L)\tau \in N$ such that $(C' \vee L)\tau\sigma = (C' \vee L)\sigma$. Now $(C' \vee L)\tau$ is not redundant, $\text{ground}(\Sigma, N)_{\mathcal{I}} \not\models (C' \vee L)\tau$, contradicting the minimal choice of $C\sigma$.

If $C\sigma = (C' \vee L)\sigma$ such that L is selected, or negative and maximal then there is a clause $(D' \vee L') \in N$ and grounding substitution ρ, such that $L'\rho$ is a strictly maximal positive literal in $(D' \vee L')\rho$, $L'\rho \in \text{ground}(\Sigma, N)_{\mathcal{I}}$ and $L'\rho = \neg L\sigma$. Again, since N is saturated, there is variable disjoint clause $(C' \vee D')\tau \in N$ for some unifier τ, $(C' \vee D')\tau\sigma\rho \prec C\sigma$, and $\text{ground}(\Sigma, N)_{\mathcal{I}} \not\models (C' \vee D')\tau\sigma\rho$ contradicting the minimal choice of $C\sigma$. □

Dynamic stuff: a clause C is called *persistent* in a derivation $N \to^*_{\text{SUP}} N'$ if there is some i such that $C \in N_i$ for $N \to^i_{\text{SUP}} N_i$ and for all $j > i$, $N \to^j_{\text{SUP}} N_j$ then $C \in N_j$. A derivation $N \to^*_{\text{SUP}} N'$ is called *fair* if any two persistent clauses C, D and any superposition inference C' out of the two clauses, there is an index j such with $N \to^j_{\text{SUP}} N_j \to^*_{\text{SUP}} N'$ such that $C' \in N_j$.

Definition 3.13.11 (Persistent Clause)**.** Let $N_0 \Rightarrow_{\text{SUP}} N_1 \Rightarrow_{\text{SUP}} \ldots$ be a, possibly infinite, superposition derivation. A clause C is called *persistent* in this derivation if $C \in N_i$ for some i and for all $j > i$ also $C \in N_j$.

Definition 3.13.12 (Fair Derivation)**.** A derivation $N_0 \Rightarrow_{\text{SUP}} N_1 \Rightarrow_{\text{SUP}} \ldots$ is called *fair* if for any persistent clause $C \in N_i$ where factoring is applicable to C, there is a j such that the factor of $C' \in N_j$ or $\bot \in N_j$. If $\{C, D\} \subseteq N_i$ are persistent clauses such that superposition left is applicable to C, D then the superposition left result is also in N_j for some j or $\bot \in N_j$.

Theorem 3.13.13 (Dynamic Superposition Completeness)**.** If N is unsatisfiable and $N = N_0 \Rightarrow_{\text{SUP}} N_1 \Rightarrow_{\text{SUP}} \ldots$ is a fair derivation, then there is $\bot \in N_j$ for some j.