Optional material (unassessed)

Overview ...

These slides cover additional topics which couldn’t be covered in lectures. In particular:

- Soundness and refutational completeness of Res for first-order clause logic
- Lexicographic orderings, reduction orderings

Generalising Resolution to Non-Ground Clauses

- Propositional/ground resolution:
  - refutationally complete,
  - in its most naive version:
    - not guaranteed to terminate for satisfiable sets of clauses,
      (improved versions do terminate, however)
    - clearly inferior to the DPLL procedure
      (even with various improvements).

- But: in contrast to the DPLL procedure, resolution can be easily extended to non-ground clauses.
General Resolution through Instantiation

Idea: instantiate clauses appropriately:

\[ P(z', z) \lor \neg Q(z) \quad \neg P(a, y) \quad P(x', b) \lor Q(f(x', x)) \]

\[ z'/a, \, z/f(a, b) \]

\[ P(a, a) \lor \neg Q(f(a, b)) \quad \neg P(a, a) \quad \neg P(a, b) \quad P(a, b) \lor Q(f(a, b)) \]

\[ \neg Q(f(a, b)) \quad Q(f(a, b)) \]

\[ \bot \]

General Resolution through Instantiation: Problems

- Problems:
  - More than one instance of a clause can participate in a proof.
  - Even worse: There are infinitely many possible instances.

- Observation:
  - Instantiation must produce complementary literals (so that inferences become possible).

- Idea:
  - Do not instantiate more than necessary to get complementary literals.

General Resolution through Lazy Instantiation

Idea: do not instantiate more than necessary:

\[ P(z', z') \lor \neg Q(z) \quad \neg P(a, y) \quad P(x', b) \lor Q(f(x', x)) \]

\[ z'/a \]

\[ P(a, a) \lor \neg Q(z) \quad \neg P(a, a) \quad \neg P(a, b) \quad P(a, b) \lor Q(f(a, x)) \]

\[ \neg Q(z) \quad Q(f(a, x)) \]

\[ z/f(a, x) \]

\[ \neg Q(f(a, x)) \quad Q(f(a, x)) \]

\[ \bot \]

Lifting Principle

- Problem:
  Make saturation of infinite sets of clauses as they arise from taking the (ground) instances of finitely many general clauses (with variables) effective and efficient.

- Idea (Robinson 1965):
  - Resolution for general clauses:
  - Equality of ground atoms (matching) is generalised to unifiability of general atoms;
  - Only compute most general (minimal) unifiers.
Lifting Principle (cont’d)

- Significance:
  - The advantage of the method in Robinson (1965) compared with Gilmore (1960) 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.

Lifting Lemma

Lemma 1
Let $C$ and $D$ be variable-disjoint clauses. If

\[
\begin{align*}
C & \quad D \\
\sigma & \quad \rho
\end{align*}
\]

then there exist $C''$ and a substitution $\tau$ such that

\[
\begin{align*}
C'' & \quad D \\
\tau
\end{align*}
\]

Resolution for General Clauses (cont’d)

- For inferences with more than one premise, we assume that the variables in the premises are (bijectively) renamed such that they become different to any variable in the other premises.
- We do not formalize this. Which names one uses for variables is otherwise irrelevant.

Resolution for General Clauses

- General binary resolution calculus $Res$:
  \[
  \frac{C \lor A \quad D \lor \neg B}{(C \lor D)\sigma} \quad \text{if } \sigma = \text{mgu}(A, B) \quad \text{(resolution)}
  \]
  \[
  \frac{C \lor A \lor B}{(C \lor A)\sigma} \quad \text{if } \sigma = \text{mgu}(A, B) \quad \text{(positive factoring)}
  \]

- General resolution calculus $RIF$ with implicit factoring:
  \[
  \frac{C \lor A_1 \lor \ldots \lor A_n \quad D \lor \neg B}{(C \lor D)\sigma} \quad \text{if } \sigma = \text{mgu}(A_1, \ldots, A_n, B) \quad \text{(RIF)}
  \]

- General resolution calculus $DIF$ with implicit factoring:
  \[
  \frac{C \lor A_1 \lor \ldots \lor A_n \quad D \lor \neg B}{(C \lor D)\sigma} \quad \text{if } \sigma = \text{mgu}(A_1, \ldots, A_n, B) \quad \text{(DIF)}
  \]
Saturation of Sets of General Clauses (cont’d)

- An analogous lifting lemma holds for factoring.

Proof:
W.l.o.g. we may assume that clauses in $N$ are pairwise variable-disjoint. (Otherwise make them disjoint, and this renaming process changes neither $\text{Res}(N)$ nor $G_\Sigma(N)$.)

Let $C' \in \text{Res}(G_\Sigma(N))$, meaning

(i) there exist resolvable ground instances $C\sigma$ and $D\rho$ of $C$ and $D$ belonging to $N$ and $C'$ is their resolvent, or else

(ii) $C'$ is a factor of a ground instance $C\sigma$ of $C \in N$.

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

Case (ii): Similar (exercise).

Saturation of Sets of General Clauses

Recall that $G_\Sigma(N)$ denotes the set of ground instances of $N$ over the signature $\Sigma$.

Corollary 2
Let $N$ be a set of general clauses saturated under $\text{Res}$, i.e. $\text{Res}(N) \subseteq N$. Then also $G_\Sigma(N)$ is saturated, that is,

$$\text{Res}(G_\Sigma(N)) \subseteq G_\Sigma(N).$$

Herbrand’s Theorem

Lemma 3
Let $N$ be a set of $\Sigma$-clauses, let $\mathcal{M}$ be an interpretation. Then $\mathcal{M} \models N$ implies $\mathcal{M} \models G_\Sigma(N)$.

Lemma 4
Let $N$ be a set of $\Sigma$-clauses, let $I$ be a Herbrand interpretation. Then $I \models G_\Sigma(N)$ implies $I \models N$. 

Herbrand’s Theorem (cont’d)

Property 5 (Herbrand Theorem)
A set $N$ of $\Sigma$-clauses is satisfiable iff it has a Herbrand model over $\Sigma$.

Proof:
The “$\Leftarrow$” part is trivial. For the “$\Rightarrow$” part let $N \not\models \bot$.

$N \not\models \bot \Rightarrow \bot \not\in Res^*(N)$ (resolution is sound)
$\Rightarrow \bot \not\in G_\Sigma(Res^*(N))$
$\Rightarrow I_{G_\Sigma(Res^*(N))} \models G_\Sigma(Res^*(N))$ (Prt. 12 (BG90); Cor. 2)
$\Rightarrow I_{G_\Sigma(Res^*(N))} \models Res^*(N)$ (Lemma 8)
$\Rightarrow I_{G_\Sigma(Res^*(N))} \models N \ (N \subseteq Res^*(N))$

The Theorem of Löwenheim-Skolem

Property 6 (Löwenheim-Skolem Theorem)
Let $\Sigma$ be a countable signature and let $S$ be a set of closed $\Sigma$-formulae. Then $S$ is satisfiable iff $S$ has a model over a countable universe.

Proof:
If both $X$ and $\Sigma$ are countable, then $S$ can be at most countably infinite. Now generate, maintaining satisfiability, a set $N$ of clauses from $S$. This extends $\Sigma$ by at most countably many new Skolem functions to $\Sigma'$. As $\Sigma'$ is countable, so is $T_{\Sigma'}$, the universe of Herbrand-interpretations over $\Sigma'$. Now apply Herbrand’s Theorem (Property 5).

Refutational Completeness of General Resolution

Property 7
Let $N$ be a set of general clauses where $Res(N) \subseteq N$. Then

$N \models \bot \iff \bot \in N$.

Proof:
Let $Res(N) \subseteq N$.
By Corollary 2 $Res(G_\Sigma(N)) \subseteq G_\Sigma(N)$
$N \models \bot \iff G_\Sigma(N) \models \bot \ (\text{Lemmas } 3 \& 4, \text{ Property 5})$
$\iff \bot \in G_\Sigma(N) \ (\text{prop. resol. is sound and complete})$
$\iff \bot \in N$

Compactness of First-Order Logic

Property 8 (Compactness Theorem for First-Order Logic)
Let $\Phi$ be a set of first-order formulae. $\Phi$ is unsatisfiable iff some finite subset $\Psi \subseteq \Phi$ is unsatisfiable.

Proof:
The “$\Leftarrow$” part is trivial. For the “$\Rightarrow$” part let $\Phi$ be unsatisfiable and let $N$ be the set of clauses obtained by Skolemisation and CNF transformation of the formulae in $\Phi$. Clearly $Res^*(N)$ is unsatisfiable. By Property 7 $\bot \in Res^*(N)$, and therefore $\bot \in Res^n(N)$ for some $n \in N$. Consequently, $\bot$ has a finite resolution proof $\Pi$ of depth $\leq n$. Choose $\Psi$ as the subset of formulae in $\Phi$ such that the corresponding clauses contain the assumptions (leaves) of $\Pi$. 
**Lifting Lemma for Res**

**Lemma 9**
Let $C$ and $D$ be variable-disjoint clauses. If
\[
\frac{C \quad D}{\sigma \quad \rho}
\]
(propositional inference in $Res^\prec$)
and if $S(C\sigma) \simeq S(C)$, $S(D\rho) \simeq S(D)$ (that is, “corresponding” literals are selected), then there exist $C''$ and a substitution $\tau$ s.t.
\[
\frac{C \quad D}{C''}
\]
(inference in $Res^\prec$)
\[
\frac{\tau}{C' = C'' \tau}
\]

**Lifting Lemma for Res** (cont’d)

- An analogous lifting lemma holds for factoring.

---

**Saturation of General Clause Sets**

**Corollary 10**
Let $N$ be a set of general clauses saturated under $Res^\prec$, i.e. $Res^\prec(N) \subseteq N$. Then there exists a selection function $S'$ such that $S'|_N = S'|_N$ and $G_{\Sigma}(N)$ is also saturated, i.e.,
\[
Res^\prec_{S'}(G_{\Sigma}(N)) \subseteq G_{\Sigma}(N).
\]

**Proof:**
We first define the selection function $S'$ such that $S'(C) = S(C)$ for all clauses $C \in G_{\Sigma}(N) \cap N$. For $C \in G_{\Sigma}(N) \setminus N$ we choose a fixed but arbitrary clause $D \in N$ with $C \in G_{\Sigma}(D)$ and define $S'(C)$ to be those occurrences of literals that are ground instances of the occurrences selected by $S$ in $D$. Then proceed as in the proof of Corollary 2 using the above lifting lemma.

---

**Soundness and Refutational Completeness**

**Property 11**
Let $\succ$ be an atom ordering and $S$ a selection function such that $Res^\prec_{S}(N) \subseteq N$. Then
\[
N \models \bot \text{ iff } \bot \in N
\]

**Proof:**
The “$\Leftarrow$” part is trivial. For the “$\Rightarrow$” part consider the propositional level: Construct a candidate model $I^\prec_N$ as for unrestricted resolution, except that clauses $C$ in $N$ that have selected literals are not productive, even when they are false in $I_C$ and when their maximal atom occurs only once and positively. The result for general clauses follows using Corollary 10.
Lexicographic Orderings

Let \((X_1, \succ_1), (X_2, \succ_2)\) be well-founded orderings. Define their lexicographic combination \(\succ = (\succ_1, \succ_2)_{\text{lex}}\) as an ordering on \(X_1 \times X_2\) such that

\[(x_1, x_2) \succ (y_1, y_2) \text{ iff (i) } x_1 \succ_1 y_1, \text{ or else (ii) } x_1 = y_1 \text{ and } x_2 \succ_2 y_2\]

(Analogously for more than two orderings.) This again yields a well-founded ordering (proof below).

Notation: \(\succ_{\text{lex}}\) for the lexicographic combination of \((X, \succ)\) twice (in general \(n\) times). I.e. \(\succ_{\text{lex}} = (\succ, \succ)_{\text{lex}}\).

Lexicographic Orderings: Examples

Length-based ordering on words: For alphabets \(\Sigma\) with a well-founded ordering \(\succ_{\Sigma}\), the relation \(\succ\), defined as

\[w \succ w' \text{ iff (i) } |w| > |w'| \text{ or (ii) } |w| = |w'| \text{ and } w \succ_{\Sigma, \text{lex}} w',\]

is a well-founded ordering on \(\Sigma^*\).

Notation: \(\succ_{\Sigma, \text{lex}} = (\succ_{\Sigma})_{\text{lex}}\).
Lexicographic Path Orderings

- Let $\Sigma$ be a finite signature and let $X$ be a countably infinite set of variables.
- Let $\succ$ be a strict ordering (precedence) on the set of predicate and functions symbols in $\Sigma$.
- The lexicographic path ordering $\succ_{\text{lpo}}$ on the set of terms (and atoms) over $\Sigma$ and $X$ is an ordering induced by $\succ$, satisfying:
  1. $s \succ_{\text{lpo}} t$ iff $t \in \text{var}(s)$ and $t \neq s$, or
  2. $s \equiv f(s_1, \ldots, s_m)$, $t \equiv g(t_1, \ldots, t_n)$, and
     (a) $s_i \succ_{\text{lpo}} t_i$ for some $i$, or
     (b) $f \succ g$ and $s \succ_{\text{lpo}} t_j$ for all $j$, or
     (c) $f = g$, $s \succ_{\text{lpo}} t_j$ for all $j$, and
        $(s_1, \ldots, s_m) \succ_{\text{lex}} (t_1, \ldots, t_n)$.

Lexicographic Path Orderings (cont’d)

- Definition: $s \geq_{\text{lpo}} t$ iff $s \succ_{\text{lpo}} t$ or $s = t$
- Examples:
  - $f(x) \succ_{\text{lpo}} x$
  - if $t$ is a subterm of $s$ then $s \succ_{\text{lpo}} t$
  - $f(a, b, g(c), a) \succ_{\text{lpo}} f(a, b, c, g(b))$
  - If $t$ can be homomorphically embedded into $s$ and $s \neq t$ then $s \succ_{\text{lpo}} t$
    E.g.
    $$h(f(g(a), f(b, y))) \succ_{\text{lpo}} f(g(a), y)$$

Lexicographic Combinations of Well-Founded Orderings

Lemma 12
$(X_i, \succ_i)$ is well-founded for $i \in \{1, 2\}$ iff $(X_1 \times X_2, \succ)$ with
$\succ = (\succ_1, \succ_2)_{\text{lex}}$ is well-founded.

Proof:
(i) “⇒”: Suppose $(X_1 \times X_2, \succ)$ is not well-founded. Then there is an infinite sequence $(a_0, b_0) \succ (a_1, b_1) \succ (a_2, b_2) \succ \ldots$.
Let $A = \{a_i \mid i \geq 0\} \subseteq X_1$. Since $(X_1, \succ_1)$ is well-founded, $A$ has a minimal element $a_n$. But then $B = \{b_i \mid i \geq n\} \subseteq X_2$ cannot have a minimal element, contradicting the well-foundedness of $(X_2, \succ_2)$.
(ii) “⇐”: obvious (exercise).

Reduction orderings

- A strict ordering $\succ$ is a reduction ordering iff
  (i) $\succ$ is well-founded
  (ii) $\succ$ is stable under substitutions, i.e.
       $s \succ t$ implies $s\sigma \succ t\sigma$
       for all terms $s, t$ and substitutions $\sigma$
  (iii) $\succ$ is compatible with contexts, i.e.
       $s \succ t$ implies $u[s] \succ u[t]$
       for all terms $s, t$ and contexts $u$

- Examples:
  - For (ii): $f(x) \succ g(x)$ implies $f(a) \succ g(a)$.
  - For (iii): $a \succ b$ implies $f(a) \succ f(a)$.
Properties of LPOs

Lemma 13
$s \succ_{\text{lpo}} t$ implies $\text{var}(s) \supseteq \text{var}(t)$.

Proof:
By induction on $|s| + |t|$ and case analysis.

Properties of LPOs (cont’d)

Property 14
$\succ_{\text{lpo}}$ is a reduction ordering on the set of terms (and atoms) over $\Sigma$ and $X$.

Proof:
Show transitivity, stability under substitutions, compatibility contexts, and irreflexivity, usually by induction on the sum of the term sizes and case analysis.
Details: Baader and Nipkow, page 119–120.

Properties of LPOs (cont’d)

Property 15
If the precedence $\succ$ is total, then the lexicographic path ordering $\succ_{\text{lpo}}$ is total on ground terms (and ground atoms), i.e. for all ground terms (or atoms) $s, t$ of the following is true: $s \succ_{\text{lpo}} t$ or $t \succ_{\text{lpo}} s$ or $s = t$.

Proof:
By induction on $|s| + |t|$ and case analysis.

Variations of the LPO

There are several possibilities to compare subterms in 2.(c):
- compare list of subterms lexicographically left-to-right ("lexicographic path ordering (lpo)", Kamin and Lvy)
- compare list of subterms lexicographically right-to-left (or according to some permutation $\pi$)
- compare multiset of subterms using the multiset extension ("multiset path ordering (mpo)", Dershowitz)
- to each function symbol $f/n$ associate a status $\in \{\text{mul}\} \cup \{\text{lex}_\pi | \pi : \{1, \ldots, n\} \rightarrow \{1, \ldots, n\}\}$ and compare according to that status ("recursive path ordering (rpo) with status")