Automated Theorem Proving

Lecture 7: Resolution Continued

Prof. Dr. Jasmin Blanchette
based on slides by Dr. Uwe Waldmann

Winter Term 2024/25

3.13 Ordered Resolution with Selection

Motivation: Search space for *Res* *very* large.

Ideas for improvement:

1. In the completeness proof (Model Existence Theorem 3.9.5) one only needs to resolve and factor maximal atoms
   $\Rightarrow$ if the calculus is restricted to inferences involving maximal atoms, the proof remains correct
   $\Rightarrow$ *ordering restrictions*
2. In the proof, it does not really matter with which negative literal an inference is performed
   $\Rightarrow$ choose a negative literal don't-care-nondeterministically
   $\Rightarrow$ *selection*

Ordering Restrictions

In the completeness proof one only needs to resolve and factor maximal atoms. Therefore the proof remains correct if we impose ordering restrictions on ground inferences.

(Ground) Ordered Resolution:

$$\frac{D \vee A \qquad C \vee \neg A}{D \vee C}$$

if $A \succ L$ for all L in D and $\neg A \succeq L$ for all L in C.

(Ground) Ordered Factorization:

$$\frac{C \vee A \vee A}{C \vee A}$$

if $A \succeq L$ for all L in C.

Ordering Restrictions

Problem: How to extend this to nonground inferences?

In the completeness proof, we talk about (strictly) maximal literals of *ground* clauses.

In the nonground calculus, we have to consider those literals that correspond to (strictly) maximal literals of ground instances.

Ordering Restrictions

An ordering $\succ$ on atoms (or terms) is called stable under substitutions if $A \succ B$ implies $A\sigma \succ B\sigma$.

Note:

- We can not require that $A \succ B$ if and only if $A\sigma \succ B\sigma$ for all σ, because this is not computable.
- We can not require that $\succ$ is total on nonground atoms, because this would be incompatible with stability under substitution.

Consequence:
In the ordering restrictions for nonground inferences, we have to replace $\succ$ by $\not\preceq$ and $\succeq$ by $\not\prec$.

Ordering Restrictions

Ordered Resolution:

$$\frac{D \vee B \qquad C \vee \neg A}{(D \vee C)\sigma}$$

if $\sigma = \mathrm{mgu}(A, B)$ and $B\sigma \not\preceq L\sigma$ for all L in D and $\neg A\sigma \not\prec L\sigma$ for all L in C.

Ordered Factorization:

$$\frac{C \vee A \vee B}{(C \vee A)\sigma}$$

if $\sigma = \mathrm{mgu}(A, B)$ and $A\sigma \not\prec L\sigma$ for all L in C.

Selection Functions

Selection functions can be used to override ordering restrictions for individual clauses.

A selection function is a mapping

$$\mathrm{sel} : C \quad \mapsto \quad \text{set of occurrences of \textit{negative} literals in } C$$

Example of selection with selected literals indicated as $\boxed{X}$:

$$\boxed{\neg A} \lor \neg A \lor B$$

$$\boxed{\neg B_0} \lor \boxed{\neg B_1} \lor A$$

Selection Functions

Intuition:

- If a clause has at least one selected literal, compute only inferences that involve a selected literal.
- If a clause has no selected literals, compute only inferences that involve a maximal literal.

Resolution Calculus $Res^{\succ}_{\mathsf{sel}}$

The resolution calculus $Res^{\succ}_{\mathsf{sel}}$ is parameterized by

- a selection function sel
- and a well-founded ordering $\succ$ on atoms that is total on ground atoms and stable under substitutions.

Resolution Calculus $Res^{\succ}_{\mathrm{sel}}$

(Ground) Ordered Resolution with Selection:

$$\frac{D \vee A \qquad C \vee \neg A}{D \vee C}$$

if the following conditions are satisfied:

(i) $A \succ L$ for all L in D;

(ii) nothing is selected in $D \vee A$ by sel;

(iii) $\neg A$ is selected in $C \vee \neg A$,
or nothing is selected in $C \vee \neg A$ and $\neg A \succeq L$ for all L in C.

Resolution Calculus $Res^{\succ}_{\text{sel}}$

(Ground) Ordered Factorization with Selection:

$$\frac{C \vee A \vee A}{C \vee A}$$

if the following conditions are satisfied:

(i) $A \succeq L$ for all L in C;

(ii) nothing is selected in $C \vee A \vee A$ by sel.

Resolution Calculus $Res^{\succ}_{\mathrm{sel}}$

The extension from ground inferences to nonground inferences is analogous to ordered resolution (replace $\succ$ by $\not\preceq$ and $\succeq$ by $\not\prec$). Again we assume that $\succ$ is stable under substitutions.

Resolution Calculus $\mathit{Res}^{\succ}_{\mathrm{sel}}$

Ordered Resolution with Selection:

$$\frac{D \vee B \qquad C \vee \neg A}{(D \vee C)\sigma}$$

if the following conditions are satisfied:

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

(ii) $B\sigma \not\preceq L\sigma$ for all L in D;

(iii) nothing is selected in $D \vee B$ by sel;

(iv) $\neg A$ is selected in $C \vee \neg A$,
or nothing is selected in $C \vee \neg A$ and $\neg A\sigma \not\prec L\sigma$ for all L in C.

Resolution Calculus $Res^{\succ}_{\text{sel}}$

Ordered Factorization with Selection:

$$\frac{C \vee A \vee B}{(C \vee A)\sigma}$$

if the following conditions are satisfied:

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

(ii) $A\sigma \not\prec L\sigma$ for all L in C;

(iii) nothing is selected in $C \vee A \vee B$ by sel.

Lifting Lemma for $Res^{\succ}_{\text{sel}}$

Lemma 3.13.1:

Let C and D be variable-disjoint clauses. If

$$\begin{tikzcd}
D \arrow[d, "\theta_1"] & C \arrow[d, "\theta_2"] \\
D\theta_1 & C\theta_2
\end{tikzcd}$$

$$\frac{D\theta_1 \qquad C\theta_2}{C'} \qquad \text{[ground inference in } Res^{\succ}_{\text{sel}}\text{]}$$

and if $\text{sel}(D\theta_1) \simeq \text{sel}(D)$, $\text{sel}(C\theta_2) \simeq \text{sel}(C)$ (that is, “corresponding” literals are selected), then there exists a substitution ρ such that

$$\frac{D \qquad C}{C''} \qquad \text{[inference in } Res^{\succ}_{\text{sel}}\text{]}$$

$$\begin{tikzcd}
C'' \arrow[d, "\rho"] \\
C' = C''\rho
\end{tikzcd}$$

Lifting Lemma for $Res^{\succ}_{\mathsf{sel}}$

An analogous lifting lemma holds for factorization.

Saturation of Sets of General Clauses

Corollary 3.13.2:

Let N be a set of general clauses saturated under $\mathit{Res}^{\succ}_{\mathrm{sel}}$, i.e., $\mathit{Res}^{\succ}_{\mathrm{sel}}(N) \subseteq N$. Then there exists a selection function sel' such that $\mathrm{sel}|_N = \mathrm{sel}'|_N$ and $G_\Sigma(N)$ is also saturated, i.e.,

$$\mathit{Res}^{\succ}_{\mathrm{sel}'}(G_\Sigma(N)) \subseteq G_\Sigma(N).$$

Soundness and Refutational Completeness

Theorem 3.13.3:

Let $\succ$ be an atom ordering and sel a selection function such that $Res^{\succ}_{\text{sel}}(N) \subseteq N$. Then $N \models \bot \Leftrightarrow \bot \in N$

Proof:

($\Leftarrow$): trivial.

($\Rightarrow$): Consider first the propositional level:

Construct a candidate interpretation I_N as for unrestricted resolution, except that clauses C in N that have selected literals are never productive (even if they are false in I_C

and if their maximal atom occurs only once and is positive).

The result for general clauses follows using Corollary 3.13.2. $\square$

What Do We Gain?

Search spaces become smaller:

1	$P \vee Q$	
2	$P \vee \boxed{\neg Q}$	
3	$\neg P \vee Q$	
4	$\neg P \vee \boxed{\neg Q}$	
5	$Q \vee Q$	Res 1, 3
6	Q	Fact 5
7	$\neg P$	Res 6, 4
8	P	Res 6, 2
9	$\bot$	Res 8, 7

We assume $P \succ Q$ and sel as indicated by $\boxed{X}$. The maximal literal in a clause is depicted in red.

What Do We Gain?

Rotation redundancy can be avoided:

From

$$\dfrac{\dfrac{C_1 \vee A \quad C_2 \vee \neg A \vee B}{C_1 \vee C_2 \vee B} \qquad C_3 \vee \neg B}{C_1 \vee C_2 \vee C_3}$$

we can obtain by rotation

$$\dfrac{C_1 \vee A \qquad \dfrac{C_2 \vee \neg A \vee B \quad C_3 \vee \neg B}{C_2 \vee \neg A \vee C_3}}{C_1 \vee C_2 \vee C_3}$$

another proof of the same clause. In large proofs many rotations are possible. However, if $A \succ B$, then the second proof does not fulfill the ordering restrictions.

3.14 Redundancy

So far: local restrictions of the resolution inference rules using orderings and selection functions.

Is it also possible to delete clauses altogether?
Under which circumstances are clauses unnecessary
(e.g., if they are tautologies)?

Intuition: If a clause is guaranteed to be neither a minimal counterexample nor productive, then we do not need it.

A Formal Notion of Redundancy

Let N be a set of ground clauses and C a ground clause (not necessarily in N). C is called redundant w.r.t. N if there exist $C_1, \dots, C_n \in N$, $n \geq 0$, such that $C_i \prec C$ and $C_1, \dots, C_n \models C$.

Redundancy for general clauses:
C is called redundant w.r.t. N if all ground instances $C\sigma$ of C are redundant w.r.t. $G_\Sigma(N)$.

Intuition: If a ground clause C is redundant and all clauses smaller than C hold in I_C, then C holds in I_C
(so C is neither a minimal counterexample nor productive).

A Formal Notion of Redundancy

Notation: The set of all clauses that are redundant w.r.t. N is denoted by $Red(N)$.

Note: The same ordering $\succ$ is used for ordering restrictions and for redundancy (and for the completeness proof).

Examples of Redundancy

In general, redundancy is undecidable. Decidable approximations are sufficient for us, however.

Proposition 3.14.1:
Some redundancy criteria:

- C tautology (i.e., $\models C$) $\;\Rightarrow\;$ C redundant w.r.t. any set N.
- $C\sigma \subset D \;\Rightarrow\; D$ redundant w.r.t. $N \cup \{C\}$.

(Under certain conditions one may also use nonstrict subsumption, but this requires a slightly more complicated definition of redundancy.)

Saturation up to Redundancy

N is called saturated up to redundancy (w.r.t. $\mathit{Res}^{\succ}_{\mathrm{sel}}$) if

$$\mathit{Res}^{\succ}_{\mathrm{sel}}(N \setminus \mathit{Red}(N)) \subseteq N \cup \mathit{Red}(N)$$

Theorem 3.14.2:

Let N be saturated up to redundancy. Then

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

Monotonicity Properties of Redundancy

When we want to delete redundant clauses during a derivation, we have to ensure that redundant clauses *remain redundant* in the rest of the derivation.

Theorem 3.14.3:

(i) $N \subseteq M \Rightarrow \mathit{Red}(N) \subseteq \mathit{Red}(M)$

(ii) $M \subseteq \mathit{Red}(N) \Rightarrow \mathit{Red}(N) \subseteq \mathit{Red}(N \setminus M)$

Recall that $\mathit{Red}(N)$ may include clauses that are not in N.

Computing Saturated Sets

Redundancy is preserved when, during a theorem proving derivation one adds new clauses or deletes redundant clauses. This motivates the following definitions:

A run of the resolution calculus is a sequence
$N_0 \vdash N_1 \vdash N_2 \vdash \cdots$, such that
(i) $N_i \models N_{i+1}$, and
(ii) all clauses in $N_i \setminus N_{i+1}$ are redundant w.r.t. N_{i+1}.

In other words, during a run we may add a new clause if it follows from the old ones, and we may delete a clause if it is redundant w.r.t. the remaining ones.

Computing Saturated Sets

For a run, we define $N_\infty = \bigcup_{i \geq 0} \bigcap_{j \geq i} N_j$.

The set N_∞ of all persistent clauses is called the limit of the run.

Computing Saturated Sets

Lemma 3.14.4:

Let $N_0 \vdash N_1 \vdash N_2 \vdash \cdots$ be a run.

Then $\mathit{Red}(N_i) \subseteq \mathit{Red}(\bigcup_{i \geq 0} N_i)$ and $\mathit{Red}(N_i) \subseteq \mathit{Red}(N_\infty)$ for every i.

Proof:

Omitted. $\square$

Computing Saturated Sets

Corollary 3.14.5:
$N_i \subseteq N_\infty \cup Red(N_\infty)$ for every i.

Proof:
If $C \in N_i \setminus N_\infty$, then there is a $k \geq i$ such that $C \in N_k \setminus N_{k+1}$, so C must be redundant w.r.t. N_{k+1}.
Consequently, C is redundant w.r.t. N_∞. $\square$

Computing Saturated Sets

Even if a set N is inconsistent, it could happen that $\bot$ is never derived, because some required inference is never computed.

The following definition rules out such runs:

A run is called fair if the conclusion of every inference from clauses in $N_\infty \setminus \mathit{Red}(N_\infty)$ is contained in some $N_i \cup \mathit{Red}(N_i)$.

Lemma 3.14.6:
If a run is fair, then its limit is saturated up to redundancy.

Computing Saturated Sets

Theorem 3.14.7 (Refutational Completeness: Dynamic View):
Let $N_0 \vdash N_1 \vdash N_2 \vdash \cdots$ be a fair run, let N_∞ be its limit.
Then N_0 has a model if and only if $\bot \notin N_\infty$.

Proof:
($\Leftarrow$): By fairness, N_∞ is saturated up to redundancy.
If $\bot \notin N_\infty$, then it has an Herbrand model.
Since every clause in N_0 is contained in N_∞ or redundant w.r.t. N_∞, this model is also a model of $G_\Sigma(N_0)$
and therefore a model of N_0.

($\Rightarrow$): Obvious, since $N_0 \models N_\infty$. $\square$

Simplifications

In theory, the definition of a run permits to add arbitrary clauses that are entailed by the current ones.

Simplifications

In practice, we restrict to two cases:

- We add conclusions of $\mathit{Res}^{\succ}_{\mathrm{sel}}$-inferences from nonredundant premises.
  $\rightsquigarrow$ necessary to guarantee fairness

- We add clauses that are entailed by the current ones if this *makes* other clauses redundant:

$$N \cup \{C\} \vdash N \cup \{C, D\} \vdash N \cup \{D\}$$

$$\text{if } N \cup \{C\} \models D \text{ and } C \in \mathit{Red}(N \cup \{D\}).$$

Net effect: C is *simplified* to D.
$\rightsquigarrow$ useful to get easier/smaller clause sets

Simplifications

Notation for simplification rules:

$$\frac{C_1 \quad \ldots \quad C_n}{\overline{D_1 \quad \ldots \quad D_m}}$$

means

$$N \cup \{C_1, \ldots, C_n\} \vdash N \cup \{D_1, \ldots, D_m\}$$

Simplifications

Examples of simplification techniques:

- Deletion of duplicated literals:

$$\begin{array}{c} C \vee L \vee L \\ \hline\hline C \vee L \end{array}$$

- Subsumption resolution:

$$\begin{array}{cc} D \vee L & C \vee D\sigma \vee \overline{L}\sigma \\ \hline\hline D \vee L & C \vee D\sigma \end{array}$$

3.15 Hyperresolution

There are *many* variants of resolution.

One well-known example is hyperresolution (Robinson 1965):

Assume that several negative literals are selected in a clause C.
If we perform an inference with C, then one of the selected literals is eliminated.

Suppose that the remaining selected literals of C are again selected in the conclusion. Then we must eliminate the remaining selected literals one by one by further resolution steps.

Hyperresolution

Hyperresolution replaces these successive steps by a single inference.

As for $Res^{\succ}_{\mathsf{sel}}$, the calculus is parameterized by an atom ordering $\succ$ and a selection function sel.

Hyperresolution

$$\frac{D_1 \vee B_1 \quad \ldots \quad D_n \vee B_n \qquad C \vee \neg A_1 \vee \ldots \vee \neg A_n}{(D_1 \vee \ldots \vee D_n \vee C)\sigma}$$

with $\sigma = \mathrm{mgu}(A_1 \doteq B_1, \ldots, A_n \doteq B_n)$ if

(i) $B_i\sigma$ strictly maximal in $D_i\sigma$, $1 \leq i \leq n$;

(ii) nothing is selected in D_i;

(iii) the indicated occurrences of the $\neg A_i$ are exactly the ones selected by sel, or nothing is selected in the right premise and $n = 1$ and $\neg A_1\sigma$ is maximal in $C\sigma$.

Similarly to resolution, hyperresolution has to be complemented by a factorization inference.

Hyperresolution

As we have seen, hyperresolution can be simulated by iterated binary resolution.

However this yields intermediate clauses which HR might not derive, and many of them might not be extendable into a full HR inference.

3.16 Implementing Resolution: The Main Loop

Standard approach:

Select one clause (“Given clause”).

Find many partner clauses that can be used in inferences together with the “given clause” using an appropriate index data structure.

Compute the conclusions of these inferences; add them to the set of clauses.

Implementing Resolution: The Main Loop

The set of clauses is split into two subsets:

- WO = "Worked-off" (or "active") clauses:
  Have already been selected as "given clause."
- U = "Usable" (or "passive") clauses:
  Have not yet been selected as "given clause."

Implementing Resolution: The Main Loop

During each iteration of the main loop:

```
Select a new given clause $C$ from $U$;
$U := U \setminus \{C\}$.

Find partner clauses $D_i$ from $WO$;
$New :=$ Conclusions of inferences from $\{D_i \mid i \in I\} \cup C$
          where one premise is $C$;
$U := U \cup New$;
$WO := WO \cup \{C\}$
```

⇒ At any time, all inferences between clauses in WO have been computed.

⇒ The procedure is fair if no clause remains in U forever.

Implementing Resolution: The Main Loop

Additionally:

Try to simplify C using WO.
(Skip the remainder of the iteration if C can be eliminated.)

Try to simplify (or even eliminate) clauses from WO using C.

Implementing Resolution: The Main Loop

Design decision: should one also simplify U using C?

Yes $\leadsto$ "Otter loop":
Advantage: simplifications of U may be useful to derive the empty clause.

No $\leadsto$ "DISCOUNT loop":
Advantage: clauses in U are really passive;
only clauses in WO have to be kept in index data structure.
(Hence: can use index data structure for which retrieval is faster, even if update is slower and space consumption is higher.)

3.17 Summary: Resolution Theorem Proving

- Resolution is a machine-oriented calculus.
- Using unification, the enumeration of instances becomes a by-product of inference computation.
- Parameters: atom ordering $\succ$ and selection function sel.
  On the nonground level, ordering constraints can (only) be solved approximatively.
- Completeness proof by constructing candidate interpretations from productive clauses $C \vee A$, $A \succ C$.

Summary: Resolution Theorem Proving

- *Local* restrictions of inferences via $\succ$ and sel
  $\Rightarrow$ fewer proof variants.
- *Global* restrictions of the search space via redundancy
  $\Rightarrow$ computing with "smaller" / "easier" clause sets.
  (In practice: simplification and detection of redundant clauses uses 90% of the prover runtime.)
- Termination on many decidable fragments.
- However, not good enough for dealing with orderings, equality, and more specific algebraic theories (lattices, abelian groups, rings, fields)
  $\Rightarrow$ further specialization of inference systems required.