3.13 Ordered Resolution with Selection

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

Ideas for improvement:

1. In the completeness proof (Model Existence Theorem 3.19) 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.

Problem: How to extend this to non-ground inferences?

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

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

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 $A\sigma \succ B\sigma$.
- We can not require that $\succ$ is total on non-ground atoms.

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

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

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

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

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

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

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

The resolution calculus $Res^{\succ}_{\mathrm{sel}}$ 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.

(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.

(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.

The extension from ground inferences to non-ground 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.

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.

Ordered Factorization with Selection:

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

if the following conditions are satisfied:

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

(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 $Res_{\text{sel}}^{\succ}$

Lemma 3.39 *Let C and D be variable-disjoint clauses. If*

$$\begin{array}{cc} D & C \\ \downarrow \theta_1 & \downarrow \theta_2 \\ D\theta_1 & C\theta_2 \end{array}$$
$$\frac{D\theta_1 \qquad C\theta_2}{C'} \qquad \textit{[ground inference in } Res_{\text{sel}}^{\succ}\textit{]}$$

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*

$$\frac{D \qquad C}{C''} \qquad \textit{[inference in } Res_{\text{sel}}^{\succ}\textit{]}$$
$$\begin{array}{c} C'' \\ \downarrow \rho \\ C' = C''\rho \end{array}$$

An analogous lifting lemma holds for factorization.

Saturation of Sets of General Clauses

Corollary 3.40 *Let N be a set of general clauses saturated under $Res_{\text{sel}}^{\succ}$, i. e., $Res_{\text{sel}}^{\succ}(N) \subseteq N$. Then there exists a selection function* sel' *such that* $\text{sel}|_N = \text{sel}'|_N$ *and $G_\Sigma(N)$ is also saturated, i. e.,*

$$Res_{\text{sel}'}^{\succ}(G_\Sigma(N)) \subseteq G_\Sigma(N).$$

Proof. We first define the selection function sel' such that $\text{sel}'(C) = \text{sel}(C)$ 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 $\text{sel}'(C)$ to be those occurrences of literals that are ground instances of the occurrences selected by sel in D. Then proceed as in the proof of Cor. 3.30 using the lifting lemma above. $\square$

Soundness and Refutational Completeness

Theorem 3.41 *Let $\succ$ be an atom ordering and* sel *a selection function such that* $Res_{\text{sel}}^{\succ}(N) \subseteq N$. *Then*

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

Proof. The "$\Leftarrow$" part is trivial. For the "$\Rightarrow$" part 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 3.40. $\square$

What Do We Gain?

Search spaces become smaller:

1	$P \lor Q$	
2	$P \lor \boxed{\neg Q}$	
3	$\neg P \lor Q$	
4	$\neg P \lor \boxed{\neg Q}$	
5	$Q \lor Q$	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.

In this example, the ordering and selection function even ensure that the refutation proceeds strictly deterministically.

Rotation redundancy can be avoided:

From

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

we can obtain by *rotation*

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

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.

Craig-Interpolation

Theorem 3.42 (Craig 1957) *Let F and G be two propositional formulas such that $F \models G$. Then there exists a formula H (called the* interpolant *for $F \models G$), such that H contains only propositional variables occurring both in F and in G, and such that $F \models H$ and $H \models G$.*

Proof. Let Π_F, Π_G, and Π_{FG} be the sets of propositional variables that occur only in F, only in G, or both in F and G. Translate F and $\neg G$ into CNF; let N and M, respectively, denote the resulting clause set. Choose an atom ordering $\succ$ for which the propositional variables in Π_F are larger than those in $\Pi_{FG} \cup \Pi_G$. Saturate N into N' w. r. t. $Res^{\succ}_{\text{sel}}$ with an empty selection function sel. Then saturate $N' \cup M$ w. r. t. $Res^{\succ}_{\text{sel}}$ 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 from Π_{FG}. The conjunction of these premises is an interpolant H. □

The theorem also holds for first-order formulas, but in the general case, a proof based on resolution technology is complicated because of Skolemization.

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[4]

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).

[4] The notations in this subsection differ significantly from the 2021/2022 lecture. Keep that in mind when you use online lecture recordings or read exercises or exam questions from previous years.

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 3.43 *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 non-strict 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 3.44 *Let N be saturated up to redundancy. Then*

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

Proof (Sketch).
(i) Ground case: Consider the construction of the candidate interpretation $I^{\succ}_N$ for $\mathit{Res}^{\succ}_{\mathrm{sel}}$.

If a clause $C \in N$ is redundant, then there exist $C_1, \dots, C_n \in N$, $n \geq 0$, such that $C_i \prec C$ and $C_1, \dots, C_n \models C$.

If $I_C \models C_i$ by minimality, then $I_C \models C$.

In particular, C is not productive.

$\Rightarrow$ Redundant clauses are not used as premises for "essential" inferences.

By saturation, the conclusion $D' \vee C'$ of a resolution inference is contained in N (as before) or in $\mathit{Red}(N)$. In the first case, minimality of C ensures that $D' \vee C'$ is productive or $I_{D' \vee C'} \models D' \vee C'$; in the second case, it ensures that $I_{D' \vee C'} \models D' \vee C'$. So in both cases we get a contradiction (analogously for factorization). The rest of the proof works as before.

(ii) Lifting: no additional problems over the proof of Theorem 3.41. $\square$

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 3.45

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

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

Proof. (i) Obvious.

(ii) For ground clause sets N, the well-foundedness of the multiset extension of the clause ordering implies that every clause in $\mathit{Red}(N)$ is entailed by smaller clauses in N that are themselves not in $\mathit{Red}(N)$.

For general clause sets N, the result follows from the fact that every clause in $G_\Sigma(N) \setminus \mathit{Red}(G_\Sigma(N))$ is an instance of a clause in $N \setminus \mathit{Red}(N)$. □

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 $N_0 \vdash N_1 \vdash N_2 \vdash \ldots$, 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.

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.

Lemma 3.46 *Let $N_0 \vdash N_1 \vdash N_2 \vdash \ldots$ 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. Exercise. □

Corollary 3.47 *$N_i \subseteq N_\infty \cup \mathit{Red}(N_\infty)$ 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_∞. □

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 3.48 *If a run is fair, then its limit is saturated up to redundancy.*

Proof. If the run is fair, then the conclusion of every inference from non-redundant clauses in N_∞ is contained in some $N_i \cup \mathit{Red}(N_i)$, and therefore contained in $N_\infty \cup \mathit{Red}(N_\infty)$. Hence N_∞ is saturated up to redundancy. □

Theorem 3.49 (Refutational Completeness: Dynamic View) *Let $N_0 \vdash N_1 \vdash N_2 \vdash \ldots$ 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 $N_0 \models N_\infty$. □

Simplifications

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

In practice, we restrict to two cases:

- We add conclusions of $\mathit{Res}^{\succ}_{\mathrm{sel}}$-inferences from non-redundant 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

Notation for simplification rules:

$$\frac{C_1 \;\ldots\; C_n}{D_1 \;\ldots\; D_m}$$

means

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

Examples of simplification techniques:

- Deletion of duplicated literals:

$$\frac{C \vee L \vee L}{C \vee L}$$

- Subsumption resolution:

$$\frac{D \vee L \qquad C \vee D\sigma \vee \overline{L}\sigma}{D \vee L \qquad C \vee D\sigma}$$

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 replaces these successive steps by a single inference. As for $Res^{\succ}_{\mathrm{sel}}$, the calculus is parameterized by an atom ordering $\succ$ and a selection function sel.

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

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.

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.

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”.

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.

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.

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 non-ground level, ordering constraints can (only) be solved approximatively.
- Completeness proof by constructing candidate interpretations from productive clauses $C \vee A$, $A \succ C$.
- *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.

3.18 Semantic Tableaux

Literature:

M. Fitting: First-Order Logic and Automated Theorem Proving, Springer-Verlag, New York, 1996, chapters 3, 6, 7.

R. M. Smullyan: First-Order Logic, Dover Publ., New York, 1968, revised 1995.

Like resolution, semantic tableaux were developed in the sixties, independently by Zbigniew Lis and Raymond Smullyan on the basis of work by Gentzen in the 30s and of Beth in the 50s.

Idea

Idea (for the propositional case):

A set $\{F \wedge G\} \cup N$ of formulas has a model if and only if $\{F \wedge G,\ F,\ G\} \cup N$ has a model.

A set $\{F \vee G\} \cup N$ of formulas has a model if and only if $\{F \vee G,\ F\} \cup N$ or $\{F \vee G,\ G\} \cup N$ has a model.

(and similarly for other connectives).

To avoid duplication, represent sets as paths of a tree.

Continue splitting until two complementary formulas are found $\Rightarrow$ inconsistency detected.

A Tableau for $\{P \wedge \neg(Q \vee \neg R),\ \neg Q \vee \neg R\}$

1. $P \wedge \neg(Q \vee \neg R)$
2. $\neg Q \vee \neg R$

3. $\neg Q$	4. $\neg R$	This tableau is not "maximal", however the first "path" is. This path is not "closed", hence the set $\{1, 2\}$ is satisfiable. (These notions will all be defined below.)
5. P	10. P	
6. $\neg(Q \vee \neg R)$	11. $\neg(Q \vee \neg R)$	
7. $\neg Q$		
8. $\neg\neg R$		
9. R		

Properties

Properties of tableau calculi:

analytic: inferences correspond closely to the logical meaning of the symbols.

goal oriented: inferences operate directly on the goal to be proved.

global: some inferences affect the entire proof state (set of formulas), as we will see later.

Propositional Expansion Rules

Expansion rules are applied to the formulas in a tableau and expand the tableau at a leaf. We append the conclusions of a rule (horizontally or vertically) at a *leaf*, whenever the premise of the expansion rule matches a formula appearing *anywhere* on the path from the root to that leaf.

Negation Elimination

$$\frac{\neg\neg F}{F} \qquad \frac{\neg\top}{\bot} \qquad \frac{\neg\bot}{\top}$$

α-Expansion

(for formulas that are essentially conjunctions: append subformulas α_1 and α_2 one on top of the other)

$$\frac{\alpha}{\begin{array}{l}\alpha_1\\ \alpha_2\end{array}}$$

β-Expansion

(for formulas that are essentially disjunctions:
append β_1 and β_2 horizontally, i. e., branch into β_1 and β_2)

$$\frac{\beta}{\beta_1 \mid \beta_2}$$

Classification of Formulas

conjunctive			disjunctive		
α	α_1	α_2	β	β_1	β_2
$F \wedge G$	F	G	$\neg(F \wedge G)$	$\neg F$	$\neg G$
$\neg(F \vee G)$	$\neg F$	$\neg G$	$F \vee G$	F	G
$\neg(F \rightarrow G)$	F	$\neg G$	$F \rightarrow G$	$\neg F$	G

We assume that the binary connective $\leftrightarrow$ has been eliminated in advance.

Tableaux: Notions

A *semantic tableau* is a marked (by formulas), finite, unordered tree and inductively defined as follows: Let $\{F_1, \dots, F_n\}$ be a set of formulas.

(i) The tree consisting of a single path

$$\begin{array}{c} F_1 \\ \vdots \\ F_n \end{array}$$

is a tableau for $\{F_1, \dots, F_n\}$. (We do not draw edges if nodes have only one successor.)

(ii) If T is a tableau for $\{F_1, \dots, F_n\}$ and if T' results from T by applying an expansion rule then T' is also a tableau for $\{F_1, \dots, F_n\}$.

Note: We may also consider the *limit tableau* of a tableau expansion; this can be an *infinite* tree.

A *path* (from the root to a leaf) in a tableau is called *closed*, if it either contains $\perp$, or else it contains both some formula F and its negation $\neg F$. Otherwise the path is called *open*.

A tableau is called *closed*, if all paths are closed.

A *tableau proof* for F is a closed tableau for $\{\neg F\}$.

A path π in a tableau is called *maximal*, if for each formula F on π that is neither a literal nor $\perp$ nor $\top$ there exists a node in π at which the expansion rule for F has been applied.

In that case, if F is a formula on π, π also contains:

(i) α_1 and α_2, if F is a α-formula,

(ii) β_1 or β_2, if F is a β-formula, and

(iii) F', if F is a negation formula, and F' the conclusion of the corresponding elimination rule.

A tableau is called *maximal*, if each path is closed or maximal.

A tableau is called *strict*, if for each formula the corresponding expansion rule has been applied at most once on each path containing that formula.

A tableau is called *clausal*, if each of its formulas is a clause.

A Sample Proof

One starts out from the negation of the formula to be proved.

$$
\begin{array}{lcr}
1. & \neg\big((P \to (Q \to R)) \to ((P \vee S) \to ((Q \to R) \vee S))\big) & \\
2. & (P \to (Q \to R)) & [1_1] \\
3. & \neg((P \vee S) \to ((Q \to R) \vee S)) & [1_2] \\
4. & P \vee S & [3_1] \\
5. & \neg((Q \to R) \vee S)) & [3_2] \\
6. & \neg(Q \to R) & [5_1] \\
7. & \neg S & [5_2]
\end{array}
$$

$$
8.\ \neg P\ [2_1] \qquad\qquad 9.\ Q \to R\ [2_2]
$$

$$
10.\ P\ [4_1] \qquad 11.\ S\ [4_2]
$$

There are three paths, each of them closed.

Properties of Propositional Tableaux

We assume that T is a tableau for $\{F_1, \ldots, F_n\}$.

Theorem 3.50 *$\{F_1, \ldots, F_n\}$ satisfiable $\Leftrightarrow$ some path (i. e., the set of its formulas) in T is satisfiable.*

Proof. ($\Leftarrow$) Trivial, since every path contains in particular $F_1, \ldots, F_n$.
($\Rightarrow$) By induction over the structure of T. $\square$

Corollary 3.51 *T closed $\Rightarrow$ $\{F_1, \ldots, F_n\}$ unsatisfiable*

Theorem 3.52 *Every strict propositional tableau expansion is finite.*

Proof. New formulas resulting from expansion are either $\bot$, $\top$ or subformulas of the expanded formula (modulo de Morgan's law), so the number of formulas that can occur is finite. By strictness, on each path a formula can be expanded at most once. Therefore, each path is finite, and a finitely branching tree with finite paths is finite by Lemma 1.9. $\square$

Conclusion: Strict and maximal tableaux can be effectively constructed.

Refutational Completeness

A set $\mathcal{H}$ of propositional formulas is called a Hintikka set, if

(1) there is no $P \in \Pi$ with $P \in \mathcal{H}$ and $\neg P \in \mathcal{H}$;

(2) $\bot \notin \mathcal{H}$, $\neg\top \notin \mathcal{H}$;

(3) if $\neg\neg F \in \mathcal{H}$, then $F \in \mathcal{H}$;

(4) if $\alpha \in \mathcal{H}$, then $\alpha_1 \in \mathcal{H}$ and $\alpha_2 \in \mathcal{H}$;

(5) if $\beta \in \mathcal{H}$, then $\beta_1 \in \mathcal{H}$ or $\beta_2 \in \mathcal{H}$.

Lemma 3.53 (Hintikka's Lemma) *Every Hintikka set is satisfiable.*

Proof. Let $\mathcal{H}$ be a Hintikka set. Define a valuation $\mathcal{A}$ by $\mathcal{A}(P) = 1$ if $P \in \mathcal{H}$ and $\mathcal{A}(P) = 0$ otherwise. Then show that $\mathcal{A}(F) = 1$ for all $F \in \mathcal{H}$ by induction over the size of formulas. □

Theorem 3.54 *Let π be a maximal open path in a tableau. Then the set of formulas on π is satisfiable.*

Proof. We show that set of formulas on π is a Hintikka set: Conditions (3), (4), (5) follow from the fact that π is maximal; conditions (1) and (2) follow from the fact that π is open and from maximality for the second negation elimination rule. □

Note: The theorem holds also for infinite trees that are obtained as the limit of a tableau expansion.

Theorem 3.55 $\{F_1, \ldots, F_n\}$ *satisfiable* $\Leftrightarrow$ *there exists no closed strict tableau for* $\{F_1, \ldots, F_n\}$.

Proof. ($\Rightarrow$) Clear by Cor. 3.51.
($\Leftarrow$) Let T be a strict maximal tableau for $\{F_1, \ldots, F_n\}$ and let π be an open path in T. By the previous theorem, the set of formulas on π is satisfiable, and hence by Theorem 3.50 the set $\{F_1, \ldots, F_n\}$, is satisfiable. □

Consequences

The validity of a propositional formula F can be established by constructing a strict maximal tableau for $\{\neg F\}$:

- T closed $\Leftrightarrow$ F valid.
- It suffices to test complementarity of paths w. r. t. atomic formulas (cf. reasoning in the proof of Theorem 3.54).
- Which of the potentially many strict maximal tableaux one computes does not matter. In other words, tableau expansion rules can be applied don't-care non-deterministically ("*proof confluence*").
- The expansion strategy, however, can have a dramatic impact on the tableau size.

A Variant of the β-Rule

Since $F \vee G \models\mathrel{\mkern-3mu}\dashv F \vee (G \wedge \neg F)$, the β expansion rule

$$\frac{\beta}{\beta_1 \mid \beta_2}$$

can be replaced by the following variant:

$$\frac{\beta}{\beta_1 \;\left|\; \begin{array}{l}\beta_2\\ \neg\beta_1\end{array}\right.}$$

The variant β-rule can lead to much shorter proofs, but it is not always beneficial.

In general, it is most helpful if $\neg\beta_1$ can be at most (iteratively) α-expanded.

3.19 Semantic Tableaux for First-Order Logic

There are two ways to extend the tableau calculus to quantified formulas:

- using ground instantiation,
- using free variables.

Tableaux with Ground Instantiation

Classification of quantified formulas:

universal		existential	
γ	$\gamma(t)$	δ	$\delta(t)$
$\forall x F$	$F\{x \mapsto t\}$	$\exists x F$	$F\{x \mapsto t\}$
$\neg\exists x F$	$\neg F\{x \mapsto t\}$	$\neg\forall x F$	$\neg F\{x \mapsto t\}$

Idea:

Replace universally quantified formulas by appropriate ground instances.

γ-expansion

$$\frac{\gamma}{\gamma(t)} \quad \text{where } t \text{ is some ground term}$$

δ-expansion

$$\frac{\delta}{\delta(c)} \quad \text{where } c \text{ is a new Skolem constant}$$

Skolemization becomes part of the calculus and needs not necessarily be applied in a preprocessing step. Of course, one could do Skolemization beforehand, and then the δ-rule would not be needed.

Note:

Skolem *constants* are sufficient:
In a δ-formula $\exists x\, F$, $\exists$ is the outermost quantifier and x is the only free variable in F.

Problems:

Having to guess ground terms is impractical.

Even worse, we may have to guess *several* ground instances, as strictness for γ is incomplete. For instance, constructing a closed tableau for

$$\{\forall x\, (P(x) \rightarrow P(f(x))),\ P(b),\ \neg P(f(f(b)))\}$$

is impossible without applying γ-expansion twice on one path.

Free-Variable Tableaux

An alternative approach:

Delay the instantiation of universally quantified variables.

Replace universally quantified variables by new free variables.

Intuitively, the free variables are universally quantified *outside* of the entire tableau.

γ-expansion

$$\frac{\gamma}{\gamma(x)} \quad \text{where } x \text{ is a new free variable}$$

δ-expansion

$$\frac{\delta}{\delta(f(x_1, \ldots, x_n))}$$

where f is a new Skolem function, and the x_i are the free variables in δ

Application of expansion rules has to be supplemented by a *substitution rule:*

(iii) If T is a tableau for $\{F_1, \ldots, F_n\}$ and if σ is a substitution, then $T\sigma$ is also a tableau for $\{F_1, \ldots, F_n\}$.

The substitution rule may, potentially, modify all the formulas of a tableau. This feature is what makes the tableau method a *global proof method*. (Resolution, by comparison, is a local method.)

One can show that it is sufficient to consider substitutions σ for which there is a path in T containing two *literals* $\neg A$ and B such that $\sigma = \mathrm{mgu}(A, B)$. Such tableaux are called *AMGU-Tableaux*.

Example

1.	$\neg\big(\exists w \forall x\ P(x,w,f(x,w)) \rightarrow \exists w \forall x \exists y\ P(x,w,y)\big)$	
2.	$\exists w \forall x\ P(x,w,f(x,w))$	1_1 $[\alpha]$
3.	$\neg\exists w \forall x \exists y\ P(x,w,y)$	1_2 $[\alpha]$
4.	$\forall x\ P(x,c,f(x,c))$	$2(c)$ $[\delta]$
5.	$\neg\forall x \exists y\ P(x,v_1,y)$	$3(v_1)$ $[\gamma]$
6.	$\neg\exists y\ P(b(v_1),v_1,y)$	$5(b(v_1))$ $[\delta]$
7.	$P(v_2,c,f(v_2,c))$	$4(v_2)$ $[\gamma]$
8.	$\neg P(b(v_1),v_1,v_3)$	$6(v_3)$ $[\gamma]$

7. and 8. are complementary (modulo unification):

$$\{v_2 \doteq b(v_1),\ c \doteq v_1,\ f(v_2,c) \doteq v_3\}$$

is solvable with an mgu $\sigma = \{v_1 \mapsto c,\ v_2 \mapsto b(c),\ v_3 \mapsto f(b(c),c)\}$, and hence, $T\sigma$ is a closed (linear) tableau for the formula in 1.

Problem:

Strictness for γ is still incomplete. For instance, constructing a closed tableau for

$$\{\forall x\,(P(x) \rightarrow P(f(x))),\ P(b),\ \neg P(f(f(b)))\}$$

is impossible without applying γ-expansion twice on one path.

Semantic Tableaux vs. Resolution

- Tableaux: global, goal-oriented, “backward”.
- Resolution: local, “forward”.
- Goal-orientation is a clear advantage if only a small subset of a large set of formulas is necessary for a proof. (Note that resolution provers saturate also those parts of the clause set that are irrelevant for proving the goal.)
- Resolution can be combined with more powerful redundancy elimination methods; because of its global nature this is more difficult for the tableau method.
- Resolution can be refined to work well with equality; for tableaux this seems to be impossible.
- 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.

3.20 Other Deductive Systems

- Instantiation-based methods
  Resolution-based instance generation
  Disconnection calculus
  ...
- Natural deduction
- Sequent calculus/Gentzen calculus
- Hilbert calculus

Instantiation-Based Methods for FOL

Idea:

Overlaps of complementary literals produce instantiations (as in resolution);

However, contrary to resolution, clauses are not recombined.

Instead: treat remaining variables as constant and use efficient propositional proof methods, such as CDCL.

There are both saturation-based variants, such as partial instantiation (Hooker et al. 2002) or resolution-based instance generation (Inst-Gen) (Ganzinger and Korovin 2003), and tableau-style variants, such as the disconnection calculus (Billon 1996; Letz and Stenz 2001).

Successful in practice for problems that are "almost propositional" (i. e., no non-constant function symbols, no equality).

Natural Deduction

Idea:

Model the concept of proofs from assumptions as humans do it.

To prove $F \rightarrow G$, assume F and try to derive G.

Initial ideas: Jaśkowski (1934), Gentzen (1934); extended by Prawitz (1965).

Popular in interactive proof systems.

Sequent Calculus

Idea:

Assumptions internalized into the data structure of sequents

$$F_1, \ldots, F_m \vdash G_1, \ldots, G_k$$

meaning

$$F_1 \wedge \cdots \wedge F_m \rightarrow G_1 \vee \cdots \vee G_k$$

Inferences rules, e. g.:

$$\frac{\Gamma \vdash \Delta}{\Gamma, F \vdash \Delta}\ (WL) \qquad \frac{\Gamma, F \vdash \Delta \qquad \Sigma, G \vdash \Pi}{\Gamma, \Sigma, F \vee G \vdash \Delta, \Pi}\ (\vee L)$$

$$\frac{\Gamma \vdash \Delta}{\Gamma \vdash F, \Delta}\ (WR) \qquad \frac{\Gamma \vdash F, \Delta \qquad \Sigma \vdash G, \Pi}{\Gamma, \Sigma \vdash F \wedge G, \Delta, \Pi}\ (\wedge R)$$

Initial idea: Gentzen 1934.

Perfect symmetry between the handling of assumptions and their consequences; interesting for proof theory.

Can be used both backwards and forwards.

Allows to simulate both natural deduction and semantic tableaux.

Hilbert Calculus

Idea:

Direct proof method (proves a theorem from axioms, rather than refuting its negation)

Axiom schemes, e. g.,

$$F \rightarrow (G \rightarrow F)$$
$$(F \rightarrow (G \rightarrow H)) \rightarrow ((F \rightarrow G) \rightarrow (F \rightarrow H))$$

plus Modus ponens:

$$\frac{F \qquad F \rightarrow G}{G}$$

Unsuitable for finding or reading proofs, but sometimes used for *specifying* (e. g., modal) logics.