

3.14 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_{sel}^\succ , the calculus is parameterized by an atom ordering \succ and a selection function sel .

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

with $\sigma = \text{mgu}(A_1 \doteq B_1, \dots, A_n \doteq B_n)$, 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.

3.15 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 ;
 $U := U \cup New$;
 $WO := WO \cup \{C\}$

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

\Rightarrow 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 \rightsquigarrow “Otter loop”:

Advantage: simplifications of U may be useful to derive the empty clause.

no \rightsquigarrow “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.)

3.16 Summary: Resolution Theorem Proving

- Resolution is a machine calculus.
- Subtle interleaving of enumerating instances and proving inconsistency through the use of unification.
- 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 elimination of redundancy
 \Rightarrow computing with “smaller”/“easier” clause sets;
 \Rightarrow 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.

3.17 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.

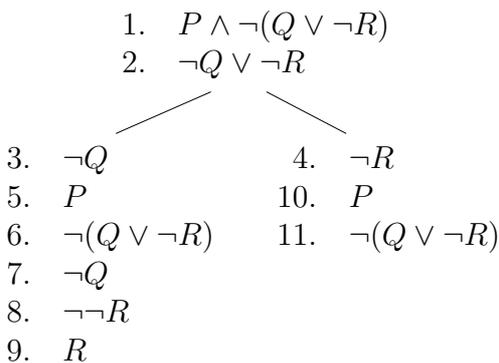
A set $\{F \vee G\} \cup N$ of formulas has a model if and only if $\{F \vee G, F\} \cup N$ or $\{F \vee G, G\} \cup N$ has a model.

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



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.)

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 (unlike, e. g., ordered resolution).

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

$$\frac{\neg\neg F}{F} \quad \frac{\neg\top}{\perp} \quad \frac{\neg\perp}{\top}$$

α -Expansion

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

$$\frac{\alpha}{\alpha_1 \alpha_2}$$

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* 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.

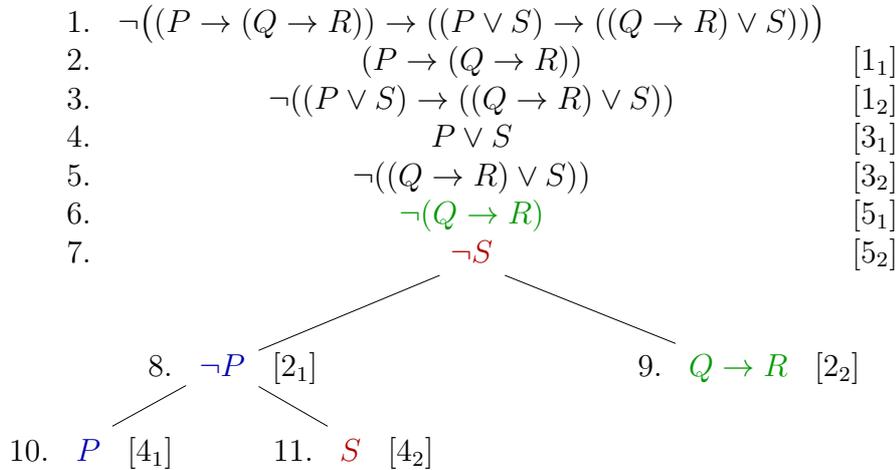
A tableau is called *maximal*, if each path is closed or maximal.

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.



There are three paths, each of them closed.

Properties of Propositional Tableaux

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

Theorem 3.49 $\{F_1, \dots, F_n\}$ 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, \dots, F_n .

(\Rightarrow) By induction over the structure of T . □

Corollary 3.50 T closed $\Rightarrow \{F_1, \dots, F_n\}$ unsatisfiable

Theorem 3.51 Every strict propositional tableau expansion is finite.

Proof. New formulas resulting from expansion are either \perp , \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 1.8. □

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) $\perp \notin \mathcal{H}$, $\neg\top \notin \mathcal{H}$;
- (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 3.52 (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. \square

Theorem 3.53 *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. \square

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

Theorem 3.54 $\{F_1, \dots, F_n\}$ *satisfiable* \Leftrightarrow *there exists no closed strict tableau for $\{F_1, \dots, F_n\}$.*

Proof. (\Rightarrow) Clear by Cor. 3.50.

(\Leftarrow) Let T be a strict maximal tableau for $\{F_1, \dots, F_n\}$ 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 3.49 the set $\{F_1, \dots, F_n\}$, is satisfiable. \square

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 3.53).
- 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 $F \vee G \models F \vee (G \wedge \neg F)$, the β expansion rule

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

can be replaced by the following variant:

$$\frac{\beta}{\beta_1 \mid \begin{array}{l} \beta_2 \\ \neg\beta_1 \end{array}}$$

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.

3.18 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	$\delta(t)$
$\forall xF$	$F\{x \mapsto t\}$	$\exists xF$	$F\{x \mapsto t\}$
$\neg\exists xF$	$\neg F\{x \mapsto t\}$	$\neg\forall xF$	$\neg F\{x \mapsto t\}$

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

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

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 $\{F_1, \dots, F_n\}$ and if σ is a substitution, then $T\sigma$ is also a tableau for $\{F_1, \dots, F_n\}$.

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 $\sigma = \text{mgu}(A, B)$. Such tableaux are called *AMGU-Tableaux*.

Example

- | | |
|--|-----------------------------|
| 1. $\neg(\exists w \forall x p(x, w, f(x, w)) \rightarrow \exists w \forall x \exists y p(x, w, y))$ | |
| 2. $\exists w \forall x p(x, w, f(x, w))$ | 1 ₁ [α] |
| 3. $\neg \exists w \forall x \exists y p(x, w, y)$ | 1 ₂ [α] |
| 4. $\forall x p(x, c, f(x, c))$ | 2(c) [δ] |
| 5. $\neg \forall x \exists y p(x, v_1, y)$ | 3(v_1) [γ] |
| 6. $\neg \exists y p(b(v_1), v_1, y)$ | 5($b(v_1)$) [δ] |
| 7. $p(v_2, c, f(v_2, c))$ | 4(v_2) [γ] |
| 8. $\neg p(b(v_1), v_1, v_3)$ | 6(v_3) [γ] |

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.