Non-classical logics

Lecture 4: Classical logic, Part 4

19.11.2014

Viorica Sofronie-Stokkermans

sofronie@uni-koblenz.de

Winter Semester 2014/2015

Last time

- Propositional logic

  Syntax, Semantics

  Decision procedures for satisfiability (resolution, tableaux)

- First-order logic

  Syntax, Semantics

  Bound/free Variables; Substitutions

Structures

A Σ-structure (also called Σ-interpretation or sometimes Σ-algebra) is a triple

$$\mathcal{A} = (U,\ (f_{\mathcal{A}} : U^n \to U)_{f/n \in \Omega},\ (p_{\mathcal{A}} \subseteq U^m)_{p/m \in \Pi})$$

where $U \neq \emptyset$ is a set, called the universe of $\mathcal{A}$.

Remark: Instead of writing $p_{\mathcal{A}} \subseteq U^m$ we can also us the characteristic function and write:

$$p_{\mathcal{A}} : U^m \to \{0, 1\}.$$

Normally, by abuse of notation, we will have $\mathcal{A}$ denote both the structure and its universe.

By Σ-Str we denote the class of all Σ-structures.

Assignments

A variable has no intrinsic meaning. The meaning of a variable has to be defined externally (explicitly or implicitly in a given context) by an assignment.

A (variable) assignment, also called a valuation (over a given Σ-structure $\mathcal{A}$), is a map $\beta : X \to \mathcal{A}$.

Value of a term in $\mathcal{A}$ with respect to β

By structural induction we define

$$\mathcal{A}(\beta) : \mathsf{T}_\Sigma(X) \to \mathcal{A}$$

as follows:

$$\begin{aligned} \mathcal{A}(\beta)(x) &= \beta(x), & x \in X \\ \mathcal{A}(\beta)(f(s_1, \ldots, s_n)) &= f_{\mathcal{A}}(\mathcal{A}(\beta)(s_1), \ldots, \mathcal{A}(\beta)(s_n)), & f/n \in \Omega \end{aligned}$$

Value of a term in $\mathcal{A}$ with respect to β

In the scope of a quantifier we need to evaluate terms with respect to modified assignments. To that end, let $\beta[x \mapsto a] : X \to \mathcal{A}$, for $x \in X$ and $a \in \mathcal{A}$, denote the assignment

$$\beta[x \mapsto a](y) := \begin{cases} a & \text{if } x = y \\ \beta(y) & \text{otherwise} \end{cases}$$

Truth value of a formula in $\mathcal{A}$ with respect to β

$\mathcal{A}(\beta) : \mathrm{F}_\Sigma(X) \to \{0, 1\}$ is defined inductively as follows:

$$
\begin{aligned}
\mathcal{A}(\beta)(\bot) &= 0 \\
\mathcal{A}(\beta)(\top) &= 1 \\
\mathcal{A}(\beta)(p(s_1, \dots, s_n)) = 1 &\quad\Leftrightarrow\quad (\mathcal{A}(\beta)(s_1), \dots, \mathcal{A}(\beta)(s_n)) \in p_{\mathcal{A}} \\
\mathcal{A}(\beta)(s \approx t) = 1 &\quad\Leftrightarrow\quad \mathcal{A}(\beta)(s) = \mathcal{A}(\beta)(t) \\
\mathcal{A}(\beta)(\neg F) = 1 &\quad\Leftrightarrow\quad \mathcal{A}(\beta)(F) = 0 \\
\mathcal{A}(\beta)(F \rho G) &= \mathrm{B}_\rho(\mathcal{A}(\beta)(F), \mathcal{A}(\beta)(G)) \\
&\qquad \text{with } \mathrm{B}_\rho \text{ the Boolean function associated with } \rho \\
\mathcal{A}(\beta)(\forall x F) &= \min_{a \in U} \{\mathcal{A}(\beta[x \mapsto a])(F)\} \\
\mathcal{A}(\beta)(\exists x F) &= \max_{a \in U} \{\mathcal{A}(\beta[x \mapsto a])(F)\}
\end{aligned}
$$

Example

The "Standard" Interpretation for Peano Arithmetic:

$$
\begin{array}{rcl}
U_{\mathbb{N}} & = & \{0, 1, 2, \ldots\} \\
0_{\mathbb{N}} & = & 0 \\
s_{\mathbb{N}} & : & n \mapsto n + 1 \\
+_{\mathbb{N}} & : & (n, m) \mapsto n + m \\
*_{\mathbb{N}} & : & (n, m) \mapsto n * m \\
\leq_{\mathbb{N}} & = & \{(n, m) \mid n \text{ less than or equal to } m\} \\
<_{\mathbb{N}} & = & \{(n, m) \mid n \text{ less than } m\}
\end{array}
$$

Note that $\mathbb{N}$ is just one out of many possible Σ_{PA}-interpretations.

Example

Values over $\mathbb{N}$ for Sample Terms and Formulas:

Under the assignment $\beta : x \mapsto 1, y \mapsto 3$ we obtain

$$
\begin{array}{lcl}
\mathbb{N}(\beta)(s(x) + s(0)) & = & 3 \\
\mathbb{N}(\beta)(x + y \approx s(y)) & = & 1 \\
\mathbb{N}(\beta)(\forall x, y (x + y \approx y + x)) & = & 1 \\
\mathbb{N}(\beta)(\forall z\ z \leq y) & = & 0 \\
\mathbb{N}(\beta)(\forall x \exists y\ x < y) & = & 1
\end{array}
$$

2.3 Models, Validity, and Satisfiability

F is valid in $\mathcal{A}$ under assignment β:

$$\mathcal{A}, \beta \models F \quad :\Leftrightarrow \quad \mathcal{A}(\beta)(F) = 1$$

F is valid in $\mathcal{A}$ ($\mathcal{A}$ is a model of F):

$$\mathcal{A} \models F \quad :\Leftrightarrow \quad \mathcal{A}, \beta \models F, \text{ for all } \beta \in X \to U_{\mathcal{A}}$$

F is valid (or is a tautology):

$$\models F \quad :\Leftrightarrow \quad \mathcal{A} \models F, \text{ for all } \mathcal{A} \in \Sigma\text{-Str}$$

F is called satisfiable iff there exist $\mathcal{A}$ and β such that $\mathcal{A}, \beta \models F$. Otherwise F is called unsatisfiable.

Entailment and Equivalence

F entails (implies) G (or G is a consequence of F), written $F \models G$

$:\Leftrightarrow$ for all $\mathcal{A} \in \Sigma$-Str and $\beta \in X \to U_{\mathcal{A}}$,
whenever $\mathcal{A}, \beta \models F$ then $\mathcal{A}, \beta \models G$.

F and G are called equivalent

$:\Leftrightarrow$ for all $\mathcal{A} \in \Sigma$-Str und $\beta \in X \to U_{\mathcal{A}}$ we have
$\mathcal{A}, \beta \models F \quad \Leftrightarrow \quad \mathcal{A}, \beta \models G$.

Entailment and Equivalence

Proposition 2.6:
F entails G iff $(F \rightarrow G)$ is valid

Proposition 2.7:
F and G are equivalent iff $(F \leftrightarrow G)$ is valid.

Extension to sets of formulas N in the "natural way", e.g.,
$N \models F$

$:\Leftrightarrow$ for all $\mathcal{A} \in \Sigma$-Str and $\beta \in X \rightarrow U_{\mathcal{A}}$:
if $\mathcal{A}, \beta \models G$, for all $G \in N$, then $\mathcal{A}, \beta \models F$.

Validity vs. Unsatisfiability

Validity and unsatisfiability are just two sides of the same medal as explained by the following proposition.

Proposition 2.8:

$$F \text{ valid} \quad \Leftrightarrow \quad \neg F \text{ unsatisfiable}$$

$$N \models F \quad \Leftrightarrow \quad N \cup \{\neg F\} \text{ unsatisfiable}$$

Hence in order to design a theorem prover (validity checker) it is sufficient to design a checker for unsatisfiability.

2.4 Algorithmic Problems

Validity(F): $\models F$?

Satisfiability(F): F satisfiable?

Entailment(F,G): does F entail G?

Model($\mathcal{A}$,F): $\mathcal{A} \models F$?

Solve($\mathcal{A}$,F): find an assignment β such that $\mathcal{A}, \beta \models F$

Solve(F): find a substitution σ such that $\models F\sigma$

Abduce(F): find G with "certain properties" such that G entails F

Gödel's Famous Theorems

1. For most signatures Σ, validity is undecidable for Σ-formulas. (One can easily encode Turing machines in most signatures.)

2. For each signature Σ, the set of valid Σ-formulas is recursively enumerable.
   (We will prove this by giving complete deduction systems.)

3. For $\Sigma = \Sigma_{PA}$ and $\mathbb{N}_* = (\mathbb{N}, 0, s, +, *)$, the theory $Th(\mathbb{N}_*)$ is not recursively enumerable.

2.5 Normal Forms and Skolemization

Study of normal forms motivated by

- reduction of logical concepts,
- efficient data structures for theorem proving.

The main problem in first-order logic is the treatment of quantifiers. The subsequent normal form transformations are intended to eliminate many of them.

Prenex Normal Form

Prenex formulas have the form

$$Q_1 x_1 \ldots Q_n x_n \; F,$$

where F is quantifier-free and $Q_i \in \{\forall, \exists\}$;
we call $Q_1 x_1 \ldots Q_n x_n$ the quantifier prefix and F the matrix of the formula.

Prenex Normal Form

Computing prenex normal form by the rewrite relation $\Rightarrow_P$:

$$\begin{array}{rcll}
(F \leftrightarrow G) & \Rightarrow_P & (F \rightarrow G) \wedge (G \rightarrow F) & \\
\neg QxF & \Rightarrow_P & \overline{Q}x\neg F & (\neg Q) \\
(QxF \ \rho \ G) & \Rightarrow_P & Qy(F[y/x] \ \rho \ G), \ y \text{ fresh}, \ \rho \in \{\wedge, \vee\} & \\
(QxF \rightarrow G) & \Rightarrow_P & \overline{Q}y(F[y/x] \rightarrow G), \ y \text{ fresh} & \\
(F \ \rho \ QxG) & \Rightarrow_P & Qy(F \ \rho \ G[y/x]), \ y \text{ fresh}, \ \rho \in \{\wedge, \vee, \rightarrow\} &
\end{array}$$

Here $\overline{Q}$ denotes the quantifier dual to Q, i.e., $\overline{\forall} = \exists$ and $\overline{\exists} = \forall$.

Example

$$F := \Big(\forall x((p(x) \vee q(x,y)) \wedge \exists z\, r(x,y,z))\Big) \rightarrow \Big((p(z) \wedge q(x,z)) \wedge \forall z\, r(z,x,y)\Big)$$

Example

$$F := \Big(\forall x((p(x) \vee q(x,y)) \wedge \exists z\, r(x,y,z))\Big) \rightarrow \Big((p(z) \wedge q(x,z)) \wedge \forall z\, r(z,x,y)\Big)$$

$$\Rightarrow_P \;\; \exists x' \,[((p(x') \vee q(x',y)) \wedge \exists z\, r(x',y,z)) \rightarrow ((p(z) \wedge q(x,z)) \wedge \forall z\, r(z,x,y))]$$

Example

$$F := \left(\forall x((p(x) \vee q(x,y)) \wedge \exists z\, r(x,y,z))\right) \rightarrow \left((p(z) \wedge q(x,z)) \wedge \forall z\, r(z,x,y)\right)$$

$$\Rightarrow_P \ \exists x'[((p(x') \vee q(x',y)) \wedge \exists z\, r(x',y,z)) \rightarrow ((p(z) \wedge q(x,z)) \wedge \forall z\, r(z,x,y))]$$

$$\Rightarrow_P \ \exists x'[(\exists z'((p(x') \vee q(x',y)) \wedge r(x',y,z'))) \rightarrow ((p(z) \wedge q(x,z)) \wedge \forall z\, r(z,x,y))]$$

Example

$$F := (\forall x\Big((p(x) \vee q(x,y)) \wedge \exists z\, r(x,y,z))\Big) \rightarrow \Big((p(z) \wedge q(x,z)) \wedge \forall z\, r(z,x,y)\Big)$$

$$\begin{aligned}
&\Rightarrow_P \ \exists x'[((p(x') \vee q(x',y)) \wedge \exists z\, r(x',y,z))) \rightarrow ((p(z) \wedge q(x,z)) \wedge \forall z\, r(z,x,y))] \\
&\Rightarrow_P \exists x'[(\exists z'((p(x') \vee q(x',y)) \wedge r(x',y,z'))) \rightarrow ((p(z) \wedge q(x,z)) \wedge \forall z\, r(z,x,y))] \\
&\Rightarrow_P \exists x'\forall z'\,[(((p(x') \vee q(x',y)) \wedge r(x',y,z'))) \rightarrow ((p(z) \wedge q(x,z)) \wedge \forall z\, r(z,x,y))]
\end{aligned}$$

Example

$$F := \Big(\forall x((p(x) \vee q(x,y)) \wedge \exists z\, r(x,y,z))\Big) \rightarrow \Big((p(z) \wedge q(x,z)) \wedge \forall z\, r(z,x,y)\Big)$$

$$\begin{aligned}
&\Rightarrow_P\ \exists x'[((p(x') \vee q(x',y)) \wedge \exists z\, r(x',y,z))) \rightarrow ((p(z) \wedge q(x,z)) \wedge \forall z\, r(z,x,y))]\\
&\Rightarrow_P \exists x'[(\exists z'((p(x') \vee q(x',y)) \wedge r(x',y,z'))) \rightarrow ((p(z) \wedge q(x,z)) \wedge \forall z\, r(z,x,y))]\\
&\Rightarrow_P \exists x'\forall z'\,[((p(x') \vee q(x',y)) \wedge r(x',y,z')) \rightarrow ((p(z) \wedge q(x,z)) \wedge \forall z\, r(z,x,y))]\\
&\Rightarrow_P \exists x'\forall z'\,[((p(x') \vee q(x',y)) \wedge r(x',y,z')) \rightarrow \forall z''((p(z) \wedge q(x,z)) \wedge r(z'',x,y))]
\end{aligned}$$

Example

$$F := \Big(\forall x((p(x) \vee q(x,y)) \wedge \exists z\, r(x,y,z))\Big) \rightarrow \Big((p(z) \wedge q(x,z)) \wedge \forall z\, r(z,x,y)\Big)$$

$$\begin{aligned}
&\Rightarrow_P \; \exists x'[((p(x') \vee q(x',y)) \wedge \exists z\, r(x',y,z))) \rightarrow ((p(z) \wedge q(x,z)) \wedge \forall z\, r(z,x,y))] \\
&\Rightarrow_P \; \exists x'[(\exists z'((p(x') \vee q(x',y)) \wedge r(x',y,z'))) \rightarrow ((p(z) \wedge q(x,z)) \wedge \forall z\, r(z,x,y))] \\
&\Rightarrow_P \; \exists x' \forall z' \, [((p(x') \vee q(x',y)) \wedge r(x',y,z')) \rightarrow ((p(z) \wedge q(x,z)) \wedge \forall z\, r(z,x,y))] \\
&\Rightarrow_P \; \exists x' \forall z' \, [((p(x') \vee q(x',y)) \wedge r(x',y,z')) \rightarrow \forall z''((p(z) \wedge q(x,z)) \wedge r(z'',x,y))] \\
&\Rightarrow_P \; \exists x' \forall z' \forall z''[(((p(x') \vee q(x',y)) \wedge r(x',y,z')) \rightarrow ((p(z) \wedge q(x,z)) \wedge r(z'',x,y)))]
\end{aligned}$$

Skolemization

Intuition: replacement of $\exists y$ by a concrete choice function computing y from all the arguments y depends on.

Transformation $\Rightarrow_S$ (to be applied outermost, *not* in subformulas):

$$\forall x_1, \ldots, x_n \exists y F \quad \Rightarrow_S \quad \forall x_1, \ldots, x_n F[f(x_1, \ldots, x_n)/y]$$

where f/n is a new function symbol (Skolem function).

Goal: check satisfiability

All free variables in F are replaced by new Skolem constants.

Skolemization

Together: $F \overset{*}{\Rightarrow}_P \underbrace{G}_{\text{prenex}} \overset{*}{\Rightarrow}_S \underbrace{H}_{\text{prenex, no } \exists}$

Theorem 2.9:

Let F, G, and H as defined above and closed. Then

(i) F and G are equivalent.

(ii) $H \models G$ but the converse is not true in general.

(iii) G satisfiable (wrt. Σ-Str) $\Leftrightarrow$ H satisfiable (wrt. Σ'-Str) where $\Sigma' = (\Omega \cup SKF, \Pi)$, if $\Sigma = (\Omega, \Pi)$.

Example

Formula in Prenex form:

$$F \quad : \quad \exists z \exists x \forall y \exists x' (\neg p(z,x) \vee (q(z,y) \wedge r(y,x')))$$

Skolemization: $z \mapsto sk_1; x \mapsto sk_2; x' \mapsto sk_3(y)$

$$\Rightarrow_S^* \; \forall y(\neg p(sk_1, sk_2) \vee (q(sk_1, y) \wedge r(y, sk_3(y))))$$

Clausal Normal Form (Conjunctive Normal Form)

$$
\begin{aligned}
(F \leftrightarrow G) \quad &\Rightarrow_K \quad (F \rightarrow G) \wedge (G \rightarrow F) \\
(F \rightarrow G) \quad &\Rightarrow_K \quad (\neg F \vee G) \\
\neg(F \vee G) \quad &\Rightarrow_K \quad (\neg F \wedge \neg G) \\
\neg(F \wedge G) \quad &\Rightarrow_K \quad (\neg F \vee \neg G) \\
\neg\neg F \quad &\Rightarrow_K \quad F \\
(F \wedge G) \vee H \quad &\Rightarrow_K \quad (F \vee H) \wedge (G \vee H) \\
(F \wedge \top) \quad &\Rightarrow_K \quad F \\
(F \wedge \bot) \quad &\Rightarrow_K \quad \bot \\
(F \vee \top) \quad &\Rightarrow_K \quad \top \\
(F \vee \bot) \quad &\Rightarrow_K \quad F
\end{aligned}
$$

These rules are to be applied modulo associativity and commutativity of $\wedge$ and $\vee$. The first five rules, plus the rule $(\neg Q)$, compute the negation normal form (NNF) of a formula.

Example

Formula in Prenex form:

$$F \quad : \quad \exists z \exists x \forall y \exists x' (\neg p(z, x) \vee (q(z, y) \wedge r(y, x')))$$

Skolemization: $z \mapsto sk_1; x \mapsto sk_2; x' \mapsto sk_3(y)$

$$\Rightarrow_S^* \ \forall y (\neg p(sk_1, sk_2) \vee (q(sk_1, y) \wedge r(y, sk_3(y))))$$

Clause normal form:

$$\Rightarrow_K^* \ \forall y ((\neg p(sk_1, sk_2) \vee q(sk_1, y)) \wedge (\neg p(sk_1, sk_2) \vee r(y, sk_3(y))))$$

The Complete Picture

$$
\begin{aligned}
F \quad &\Rightarrow^*_P \quad Q_1 y_1 \ldots Q_n y_n \; G && (G \text{ quantifier-free}) \\
&\Rightarrow^*_S \quad \forall x_1, \ldots, x_m \; H && (m \leq n,\ H \text{ quantifier-free}) \\
&\Rightarrow^*_K \quad \underbrace{\underbrace{\forall x_1, \ldots, x_m}_{\text{leave out}} \; \bigwedge_{i=1}^{k} \underbrace{\bigvee_{j=1}^{n_i} L_{ij}}_{\text{clauses } C_i}}_{F'}
\end{aligned}
$$

$N = \{C_1, \ldots, C_k\}$ is called the clausal (normal) form (CNF) of F.

Note: the variables in the clauses are implicitly universally quantified.

The Complete Picture

Theorem 2.10:

Let F be closed. Then $F' \models F$.

(The converse is not true in general.)

Theorem 2.11:

Let F be closed. Then F is satisfiable iff F' is satisfiable iff N is satisfiable

Optimization

Here is lots of room for optimization since we only can preserve satisfiability anyway:

- size of the CNF exponential when done naively;
- want to preserve the original formula structure;
- want small arity of Skolem functions.

Part 3: Automated reasoning

3.1: Resolution

3.2: Tableaux

3.1 Resolution

Propositional resolution:

Resolution inference rule:

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

Terminology: $C \vee D$: resolvent; A: resolved atom

(Positive) factorisation inference rule:

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

Refinements of resolution

1. We assume that $\succ$ is any fixed ordering on propositional variables that is *total* and well-founded.

2. Extend $\succ$ to an ordering $\succ_L$ on literals:

$$\begin{array}{lcll} [\neg]P & \succ_L & [\neg]Q & \text{, if } P \succ Q \\ \neg P & \succ_L & P & \end{array}$$

3. Extend $\succ_L$ to an ordering $\succ_C$ on clauses:
$\succ_C = (\succ_L)_{\mathrm{mul}}$, the multi-set extension of $\succ_L$.

   Notation: $\succ$ also for $\succ_L$ and $\succ_C$.

(well-founded)

Selection Functions

A selection function is a mapping

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

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

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

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

Resolution Calculus $Res_S^{\succ}$

$$\frac{C \vee A \qquad D \vee \neg A}{C \vee D}$$ [ordered resolution with selection]

if

(i) $A \succ C$;

(ii) nothing is selected in C by S;

(iii) $\neg A$ is selected in $D \vee \neg A$,
or else nothing is selected in $D \vee \neg A$ and $\neg A \succeq \max(D)$.

Note: For positive literals, $A \succ C$ is the same as $A \succ \max(C)$.

Resolution Calculus $Res_S^{\succ}$

$$\frac{C \vee A \vee A}{(C \vee A)} \qquad \text{[ordered factoring]}$$

if A is maximal in C and nothing is selected in C.

Resolution for ground clauses

- Exactly the same as for propositional clauses

  Ground atoms $\mapsto$ propositional variables

Theorem

- Res is sound and refutationally complete
  (for all sets of ground clauses)
- $\text{Res}_S^{\succ}$ is sound and refutationally complete
  (for all sets of ground clauses)

Sample Refutation

1.	$\neg P(f(a)) \vee \neg P(f(a)) \vee Q(b)$	(given)
2.	$P(f(a)) \vee Q(b)$	(given)
3.	$\neg P(g(b, a)) \vee \neg Q(b)$	(given)
4.	$P(g(b, a))$	(given)
5.	$\neg P(f(a)) \vee Q(b) \vee Q(b)$	(Res. 2. into 1.)
6.	$\neg P(f(a)) \vee Q(b)$	(Fact. 5.)
7.	$Q(b) \vee Q(b)$	(Res. 2. into 6.)
8.	$Q(b)$	(Fact. 7.)
9.	$\neg P(g(b, a))$	(Res. 8. into 3.)
10.	$\perp$	(Res. 4. into 9.)

General Resolution through Instantiation

Idea: instantiate clauses appropriately:

$$
\begin{tikzcd}
P(z',z') \vee \neg Q(z) \arrow[d, "{[a/z', f(a,b)/z]}"'] & & & \neg P(a,y) \arrow[dl, "{[a/y]}"'] \arrow[dr, "{[b/y]}"] & & & P(x',b) \vee Q(f(x',x)) \arrow[d, "{[a/x', b/x]}"] \\
P(a,a) \vee \neg Q(f(a,b)) \arrow[dr] & & \neg P(a,a) \arrow[dl] & & \neg P(a,b) \arrow[dr] & & P(a,b) \vee Q(f(a,b)) \arrow[dl] \\
& \neg Q(f(a,b)) \arrow[drr] & & & & Q(f(a,b)) \arrow[dll] & \\
& & & \perp & & &
\end{tikzcd}
$$

General Resolution through Instantiation

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 Instantiation

Idea: do not instantiate more than necessary:

$$
\begin{tikzcd}
P(z',z') \lor \neg Q(z) \arrow[d, "{[a/z']}"'] & & & \neg P(a,y) \arrow[dl, "{[a/y]}"'] \arrow[dr, "{[b/y]}"] & & & P(x',b) \lor Q(f(x',x)) \arrow[d, "{[a/x']}"] \\
P(a,a) \lor \neg Q(z) \arrow[dr] & & \neg P(a,a) \arrow[dl] & & \neg P(a,b) \arrow[dr] & & P(a,b) \lor Q(f(a,x)) \arrow[dl] \\
& \neg Q(z) \arrow[d, "{[f(a,x)/z]}"'] & & & & Q(f(a,x)) \arrow[d] & \\
& \neg Q(f(a,x)) \arrow[drr] & & & & Q(f(a,x)) \arrow[dll] & \\
& & & \bot & & &
\end{tikzcd}
$$

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 65):

- Resolution for general clauses:
- *Equality* of ground atoms is generalized to *unifiability* of general atoms;
- Only compute *most general* (minimal) unifiers.

Lifting Principle

Significance: The advantage of the method in (Robinson 65) compared with (Gilmore 60) 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.

Resolution for General Clauses

General binary resolution ***Res*****:**

$$\frac{C \vee A \qquad D \vee \neg B}{(C \vee D)\sigma} \quad \text{if } \sigma = \mathrm{mgu}(A, B) \qquad [\text{resolution}]$$

$$\frac{C \vee A \vee B}{(C \vee A)\sigma} \quad \text{if } \sigma = \mathrm{mgu}(A, B) \quad [\text{factorization}]$$

Resolution for General Clauses

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.

Refutational Completeness of General Resolution

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

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

Unification

Let $E = \{s_1 \doteq t_1, \ldots, s_n \doteq t_n\}$ (s_i, t_i terms or atoms) a multi-set of equality problems. A substitution σ is called a unifier of E if $s_i\sigma = t_i\sigma$ for all $1 \leq i \leq n$.

If a unifier of E exists, then E is called unifiable.

Unification

A substitution σ is called more general than a substitution τ, denoted by $\sigma \leq \tau$, if there exists a substitution ρ such that $\rho \circ \sigma = \tau$, where $(\rho \circ \sigma)(x) := (x\sigma)\rho$ is the composition of σ and ρ as mappings.
(Note that $\rho \circ \sigma$ has a finite domain as required for a substitution.)

If a unifier of E is more general than any other unifier of E, then we speak of a most general unifier of E, denoted by $\mathrm{mgu}(E)$.

Unification after Martelli/Montanari

$$
\begin{array}{rll}
t \doteq t, E & \Rightarrow_{MM} & E \\[1em]
f(s_1, \ldots, s_n) \doteq f(t_1, \ldots, t_n), E & \Rightarrow_{MM} & s_1 \doteq t_1, \ldots, s_n \doteq t_n, E \\[1em]
f(\ldots) \doteq g(\ldots), E & \Rightarrow_{MM} & \perp \\[1em]
x \doteq t, E & \Rightarrow_{MM} & x \doteq t, E[t/x] \\
& \multicolumn{2}{l}{\text{if } x \in var(E), x \notin var(t)} \\[1em]
x \doteq t, E & \Rightarrow_{MM} & \perp \\
& \multicolumn{2}{l}{\quad \text{if } x \neq t, x \in var(t)} \\[1em]
t \doteq x, E & \Rightarrow_{MM} & x \doteq t, E \\
& \multicolumn{2}{l}{\quad \text{if } t \notin X}
\end{array}
$$

Example 1

$$\{f(g(a,x),g(y,b)) \doteq f(x,g(v,w)), f(x,g(v,w)) \doteq f(g(x,a),g(v,b))\}$$

$$\Rightarrow^{(2)}_{MM} \quad \{g(a,x) \doteq x, g(y,b) \doteq g(v,w), x \doteq g(x,a), g(v,w) \doteq g(v,b)\}$$

$$\Rightarrow^{(5)}_{MM} \quad \perp$$

Example 2

$$\{f(g(a,x),g(y,b)) \doteq g(x,g(v,w)), f(x,g(v,w)) \doteq f(g(x,a),g(v,b))\}$$

$$\overset{(3)}{\Rightarrow}_{MM} \quad \perp$$

Example 3

$$\{f(g(a,x),g(y,b)) \doteq f(z,g(v,w)), f(z,g(v,w)) \doteq f(g(x,a),g(v,b)\}$$

$$\overset{(2)}{\Rightarrow}_{MM} \quad \{g(a,x) \doteq z, g(y,b) \doteq g(v,w), z \doteq g(x,a), g(v,w) \doteq g(v,b)\}$$

$$\overset{(4)}{\Rightarrow}_{MM} \quad \{z \doteq g(a,x), g(y,b) \doteq g(v,w), g(a,x) \doteq g(x,a), g(v,w) \doteq g(v,b)$$

Example 3

$$\{f(g(a,x),g(y,b)) \doteq f(z,g(v,w)), f(z,g(v,w)) \doteq f(g(x,a),g(v,b)\}$$

$$\overset{(2)}{\Rightarrow}_{MM} \quad \{g(a,x) \doteq z, g(y,b) \doteq g(v,w), z \doteq g(x,a), g(v,w) \doteq g(v,b)\}$$

$$\overset{(4)}{\Rightarrow}_{MM} \quad \{z \doteq g(a,x), g(y,b) \doteq g(v,w), g(a,x) \doteq g(x,a), g(v,w) \doteq g(v,b)$$

$$\Rightarrow^{*}_{MM} \quad \{z \doteq g(a,x), y \doteq v, b \doteq w, a \doteq x, x \doteq a, v \doteq v, w \doteq b\}$$

Example 3

$$\{f(g(a,x),g(y,b)) \doteq f(z,g(v,w)), f(z,g(v,w)) \doteq f(g(x,a),g(v,b))\}$$

$$\begin{aligned}
&\overset{(2)}{\Rightarrow}_{MM} && \{g(a,x) \doteq z, g(y,b) \doteq g(v,w), z \doteq g(x,a), g(v,w) \doteq g(v,b)\} \\
&\overset{(4)}{\Rightarrow}_{MM} && \{z \doteq g(a,x), g(y,b) \doteq g(v,w), g(a,x) \doteq g(x,a), g(v,w) \doteq g(v,b) \\
&\Rightarrow^{*}_{MM} && \{z \doteq g(a,x), y \doteq v, b \doteq w, a \doteq x, x \doteq a, v \doteq v, w \doteq b\} \\
&\Rightarrow^{*}_{MM} && \{z \doteq g(a,x), y \doteq v, b \doteq w, a \doteq x, x \doteq a, w \doteq b\}
\end{aligned}$$

Example 3

$$\{f(g(a,x),g(y,b)) \doteq f(z,g(v,w)), f(z,g(v,w)) \doteq f(g(x,a),g(v,b))\}$$

$$\overset{(2)}{\Rightarrow}_{MM} \quad \{g(a,x) \doteq z, g(y,b) \doteq g(v,w), z \doteq g(x,a), g(v,w) \doteq g(v,b)\}$$

$$\overset{(4)}{\Rightarrow}_{MM} \quad \{z \doteq g(a,x), g(y,b) \doteq g(v,w), g(a,x) \doteq g(x,a), g(v,w) \doteq g(v,b)$$

$$\Rightarrow^{*}_{MM} \quad \{z \doteq g(a,x), y \doteq v, b \doteq w, a \doteq x, x \doteq a, v \doteq v, w \doteq b\}$$

$$\Rightarrow^{*}_{MM} \quad \{z \doteq g(a,x), y \doteq v, b \doteq w, a \doteq x, x \doteq a, w \doteq b\}$$

$$\Rightarrow^{*}_{MM} \quad \{z \doteq g(a,a), y \doteq v, b \doteq b, a \doteq a, x \doteq a, w \doteq b\}$$

$$\Rightarrow^{*}_{MM} \quad \{z \doteq g(a,a), y \doteq v, x \doteq a, w \doteq b\}$$

Example 3

$$\{f(g(a,x),g(y,b)) \doteq f(z,g(v,w)), f(z,g(v,w)) \doteq f(g(x,a),g(v,b))\}$$

$$\begin{aligned}
&\overset{(2)}{\Rightarrow}_{MM} && \{g(a,x) \doteq z, g(y,b) \doteq g(v,w), z \doteq g(x,a), g(v,w) \doteq g(v,b)\} \\
&\overset{(4)}{\Rightarrow}_{MM} && \{z \doteq g(a,x), g(y,b) \doteq g(v,w), g(a,x) \doteq g(x,a), g(v,w) \doteq g(v,b) \\
&\Rightarrow^{*}_{MM} && \{z \doteq g(a,x), y \doteq v, b \doteq w, a \doteq x, x \doteq a, v \doteq v, w \doteq b\} \\
&\Rightarrow^{*}_{MM} && \{z \doteq g(a,x), y \doteq v, b \doteq w, a \doteq x, \color{red}{x \doteq a}, \color{red}{w \doteq b}\} \\
&\Rightarrow^{*}_{MM} && \{z \doteq g(a,\color{red}{a}), y \doteq v, b \doteq \color{red}{b}, a \doteq \color{red}{a}, \color{red}{x \doteq a}, \color{red}{w \doteq b}\} \\
&\Rightarrow^{*}_{MM} && \{z \doteq g(a,a), y \doteq v, x \doteq a, w \doteq b\}
\end{aligned}$$

Most general unifier (m.g.u):

$$[g(a,a)/z, v/y, a/x, b/w]$$

MM: Main Properties

If $E = x_1 \doteq u_1, \ldots, x_k \doteq u_k$, with x_i pairwise distinct, $x_i \notin var(u_j)$, then E is called an (equational problem in) solved form representing the solution $\sigma_E = [u_1/x_1, \ldots, u_k/x_k]$.

Proposition 2.28:
If E is a solved form then σ_E is am mgu of E.

MM: Main Properties

Theorem 2.29:

1. If $E \Rightarrow_{MM} E'$ then σ is a unifier of E iff σ is a unifier of E'
2. If $E \Rightarrow^*_{MM} \bot$ then E is not unifiable.
3. If $E \Rightarrow^*_{MM} E'$ with E' in solved form, then $\sigma_{E'}$ is an mgu of E.

Theorem 2.30:

E is unifiable if and only if there is a most general unifier σ of E, such that σ is idempotent and $dom(\sigma) \cup codom(\sigma) \subseteq var(E)$.

Problem: *exponential growth* of terms possible

Example of resolution step

$$C_1 \lor L = p(x,x) \lor \underbrace{p(a,x)}_{L}$$

$$D_1 \lor \neg L' = \underbrace{\neg p(y,y)}_{\neg L'}$$

Most general unifier of L, L':

$$\begin{aligned} \{p(a,x) \doteq p(y,y)\} \quad & \Rightarrow_{MM} \quad \{a \doteq y, x \doteq y\} \\ & \Rightarrow_{MM} \quad \{y \doteq a, x \doteq a\} \end{aligned}$$

$\sigma = mgu(L, L') = [a/y, a/x]$

$$\frac{C_1 \lor L \qquad C_2 \lor \neg L'}{C_1\sigma \cup C_2\sigma}$$

Resolvent: $p(x,x)\sigma = p(a,a)$

Example

1. $P(x) \vee P(f(x)) \vee \neg Q(x)$
2. $\neg P(y)$
3. $P(g(x', x)) \vee Q(x)$

Example

1.	$P(x) \vee P(f(x)) \vee \neg Q(x)$	[Given]
2.	$\neg P(y)$	[Given]
3.	$P(g(x', x'')) \vee Q(x'')$	[Given; Rename variables]
4.	$P(f(x)) \vee \neg Q(x)$	[Res. 1, 2]; $\sigma_1 = [x/y]$
5.	$\neg Q(x)$	[Res. 4, 2]; $\sigma_2 = [f(x)/y]$
6.	$Q(x'')$	[Res. 3, 2]; $\sigma_3 = [g(x', x'')/y]$
7.	$\perp$	[res. 5, 6]; $\sigma_4 = [x/x'']$

Ordered Resolution with Selection

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

- Ordering on literals
- Selection function

Resolution Calculus $Res_S^{\succ}$

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:

Let $\succ$ be a total and well-founded ordering on ground atoms. A literal L is called [strictly] maximal in a clause C if and only if there exists a ground substitution σ such that for all L' in C: $L\sigma \succeq L'\sigma$ [$L\sigma \succ L'\sigma$].

Resolution Calculus $Res_S^{\succ}$

Let $\succ$ be an atom ordering and S a selection function.

$$\frac{C \vee A \qquad \neg B \vee D}{(C \vee D)\sigma} \qquad \text{[ordered resolution with selection]}$$

if $\sigma = \text{mgu}(A, B)$ and

(i) $A\sigma$ strictly maximal wrt. $C\sigma$;

(ii) nothing is selected in C by S;

(iii) either $\neg B$ is selected,
or else nothing is selected in $\neg B \vee D$ and $\neg B\sigma$ is maximal in $D\sigma$.

Resolution Calculus $Res_S^{\succ}$

$$\frac{C \vee A \vee B}{(C \vee A)\sigma} \qquad \text{[ordered factoring]}$$

if $\sigma = \mathrm{mgu}(A, B)$ and $A\sigma$ is maximal in $C\sigma$ and nothing is selected in C.

2.3 Semantic Tableaux

Properties of tableau calculi:

analytic: inferences according to the logical content 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

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

Propositional Expansion Rules

α-Expansion

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

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

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
$X \wedge Y$	X	Y	$\neg(X \wedge Y)$	$\neg X$	$\neg Y$
$\neg(X \vee Y)$	$\neg X$	$\neg Y$	$X \vee Y$	X	Y
$\neg(X \rightarrow Y)$	X	$\neg Y$	$X \rightarrow Y$	$\neg X$	Y

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 $\{F_1, \ldots, F_n\}$ 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 $\{F_1, \ldots, F_n\}$.
(We do not draw edges if nodes have only one successor.)

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

A Sample Proof

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

1. $\neg[(P \to (Q \to R)) \to ((P \vee S) \to ((Q \to R) \vee S))]$
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. $\neg S$ $[5_2]$

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

10. P $[4_1]$ 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. $\{F_1, \ldots, F_n\}$ satisfiable $\Leftrightarrow$ some path (i.e., the set of its formulas) in T is satisfiable.

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

Theorem. Let T be a strict propositional tableau. Then T is finite.

Conclusion: Strict and maximal tableaux can be effectively constructed.

Refutational Completeness

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

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 wrt. atomic formulas.
- 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 can have a dramatic impact on tableau size.
- Since it is sufficient to saturate paths wrt. ordered resolution (up to redundancy), tableau expansion rules can be even more restricted, in particular by certain ordering constraints.

Semantic Tableaux for First-Order Logic

Additional classification of quantified formulas:

universal		existential	
γ	$\gamma(t)$	δ	$\delta(t)$
$\forall x F$	$F[t/x]$	$\exists x F$	$F[t/x]$
$\neg \exists x F$	$\neg F[t/x]$	$\neg \forall x F$	$\neg F[t/x]$

Moreover we assume that the set of variables X is partitioned into 2 disjoint infinite subsets X_g and X_f, so that bound [free] variables variables can be chosen from X_g [X_f]. (This avoids the variable capturing problem.)

Additional Expansion Rules

γ-expansion

$$\frac{\gamma}{\gamma(x)} \quad \text{where } x \text{ is a variable in } X_f$$

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

Additional Expansion Rules

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 that the rules are parametric, instantiated by the choices for x and f, respectively. Strictness here means that only one instance of the rule is applied on each path to any formula on the path.

In this form the rules go back to Hähnle and Schmitt: The liberalized δ-rule in free variable semantic tableaux, J. Automated Reasoning 13,2, 1994, 211–221.

Definition: Free-Variable Tableau

Let $\{F_1, \dots, F_n\}$ be a set of *closed formulas*.

(i) The tree consisting of a single path: $\begin{array}{c} F_1 \\ \vdots \\ F_n \end{array}$ is a tableau for $\{F_1, \dots, F_n\}$.

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

(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 (iii) may, potentially, modify all the formulas of a tableau. This feature makes the tableau method a *global proof method*. (Resolution, by comparison, is a local method.) If one took (iii) literally, by repeated application of γ-rule one could enumerate all substitution instances of the universally quantified formulas (major drawback compared with resolution). Fortunately, we can improve on this.

Example

1.	$\neg[\exists w \forall x\ p(x, w, f(x, w)) \to \exists w \forall x \exists y\ p(x, w, y)]$	
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.	$\forall x\ p(x, a, f(x, a))$	$2(a)$ $[\delta]$
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.	$p(v_2, a, f(v_2, a))$	$4(v_2)$ $[\gamma]$
8.	$\neg p(b(v_1), v_1, v_3)$	$6(v_3)$ $[\gamma]$

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

$$v_2 \doteq b(v_1),\ \ a \doteq v_1,\ \ f(v_2, a) \doteq v_3$$

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

AMGU-Tableaux

Idea: Restrict the substitution rule to unifiers of complementary formulas.

We speak of an *A*MGU-Tableau, whenever the substitution rule is only applied for substitutions σ for which there is a path in T containing two *literals* $\neg A$ and B such that $\sigma = \text{mgu}(A, B)$.

Correctness

Given an signature Σ, by Σ^{sko} we denote the result of adding infinitely many new Skolem function symbols which we may use in the δ-rule.

Let $\mathcal{A}$ be a Σ^{sko}-interpretation, T a tableau, and β a variable assignment over $\mathcal{A}$.

T is called $(\mathcal{A}, \beta)$-valid, if there is a path P_β in T such that $\mathcal{A}, \beta \models F$, for each formula F on P_β.

T is called satisfiable if there exists a structure $\mathcal{A}$ such that for each assignment β the tableau T is $(\mathcal{A}, \beta)$-valid.
(This implies that we may choose P_β depending on β.)

Correctness

Theorem 2.52:
Let T be a tableau for $\{F_1, \ldots, F_n\}$, where the F_i are closed Σ-formulas. Then $\{F_1, \ldots, F_n\}$ is satisfiable $\Leftrightarrow$ T is satisfiable.

(Proof of "$\Rightarrow$" by induction over the depth of T. For δ one needs to reuse the ideas for proving that Skolemization preserves [un-]satisfiability.)

Incompleteness of Strictness

Strictness for γ is incomplete:

$$
\begin{array}{lcr}
1. & \neg[\forall x\ p(x) \rightarrow (p(a) \wedge p(b))] & \\
2. & \forall x\ p(x) & 1_1 \\
3. & \neg(p(a) \wedge p(b)) & 1_2 \\
4. & p(v_1) & 2(v_1)
\end{array}
$$

$$
5.\quad \neg p(a) \quad 3_1 \qquad\qquad 6.\quad \neg p(b) \quad 3_2
$$

If we placed a strictness requirement also on applications of γ, the tableau would only be expandable by the substitution rule. However, there is no substitution (for v_1) that can close both paths simultaneously.

Multiple Application of γ Solves the Problem

$$
\begin{array}{lcr}
1. & \neg[\forall x\ p(x) \rightarrow (p(a) \wedge p(b))] & \\
2. & \forall x\ p(x) & 1_1 \\
3. & \neg(p(a) \wedge p(b)) & 1_2 \\
4. & p(v_1) & 2_{v_1}
\end{array}
$$

$$
\begin{array}{lll}
5.\quad \neg p(a) \quad 3_1 & \qquad & 6.\quad \neg p(b) \quad 3_2 \\
 & & 7.\quad p(v_2) \quad 2_{v_2}
\end{array}
$$

The point is that different applications of γ to $\forall x\ p(x)$ may employ different free variables for x.

Now, by two applications of the AMGU-rule, we obtain the substitution $[a/v_1, b/v_2]$ closing the tableau.

Multiple Application of γ Solves the Problem

Therefore strictness for γ should from now on mean that each *instance* of γ (depending on the choice of the free variable) is applied at most once to each γ-formula on any path.

Refutational Completeness

Theorem 2.53:
$\{F_1, \ldots, F_n\}$ satisfiable $\Leftrightarrow$ there exists no closed, strict AMGU-Tableau for $\{F_1, \ldots, F_n\}$.

For the proof one defines a fair tableau expansion process converging against an infinite tableau where on each path each γ-formula is expanded into all its variants (modulo the choice of the free variable).

One may then again show that each path in that tableau is saturated (up to redundancy) by resolution. This requires to apply the lifting lemma for resolution in order to show completeness of the AMGU-restriction.

How Often Do we Have to Apply γ?

Theorem 2.54:

There is no recursive function $f : F_\Sigma \times F_\Sigma \to \mathbb{N}$ such that, if the closed formula F is unsatisfiable, then there exists a closed tableau for F where to all formulas $\forall x G$ appearing in T the γ-rule is applied at most $f(F, \forall x G)$ times on each path containing $\forall x G$.

Otherwise unsatisfiability or, respectively, validity for first-order logic would be decidable. In fact, one would be able to enumerate in finite time all tableaux bounded in depth as indicated by f. In other words, free-variable tableaux are not recursively bounded in their depth.

Again $\forall$ is treated like an infinite conjunction. By repeatedly applying γ, together with the substitution rule, one can enumerate all instances $F[t/x]$ vertically, that is, conjunctively, in each path containing $\forall x F$.

Semantic Tableaux vs. Resolution

- Both methods are machine methods on which today's provers are based upon.
- 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.)

Semantic Tableaux vs. Resolution

- Like resolution, the tableau method, in order to be useful in practice, must be accompanied by refinements: lemma generation, ordering restrictions, efficient term and proof data structures.
- Resolution can be combined with more powerful redundancy elimination methods.
- Because of its global nature redundancy elimination is more difficult for the tableau method.
- Resolution can be refined to work well with equality and algebraic structures; for tableaux this is more problematic.