COMP6012: Automated Reasoning, Part II
Advanced Topics

Optional Material

Renate Schmidt

School of Computer Science
University of Manchester

`schmidt@cs.man.ac.uk`

`http://www.cs.man.ac.uk/~schmidt/COMP60121/`

COMP6012: Automated Reasoning II

Optional material (unassessed)

Overview . . .

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

- Soundness and refutational completeness of *Res* for first-order clause logic
- Lexicographic orderings, reduction orderings
- Semantic tableau for propositional logic
- Free-varibale tableau for first-order logic

Generalising Resolution to Non-Ground Clauses

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

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

General Resolution through Instantiation

Idea: instantiate clauses appropriately:

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

General Resolution through Instantiation: Problems

- Problems:
  - More than one instance of a clause can participate in a proof.
  - Even worse: There are infinitely many possible instances.
- Observation:
  - Instantiation must produce complementary literals (so that inferences become possible).
- Idea:
  - Do not instantiate more than necessary to get complementary literals.

General Resolution through Lazy Instantiation

Idea: do not instantiate more than necessary:

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

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 1965):**
  - Resolution for general clauses:
  - Equality of ground atoms (matching) is generalised to unifiability of general atoms;
  - Only compute most general (minimal) unifiers.

Lifting Principle (cont'd)

- **Significance:**
  - The advantage of the method in Robinson (1965) compared with Gilmore (1960) is that unification enumerates only those instances of clauses that participate in an inference.
  - Moreover, clauses are not right away instantiated into ground clauses. Rather they are instantiated only as far as required for an inference.
  - Inferences with non-ground clauses in general represent infinite sets of ground inferences which are computed simultaneously in a single step.

Resolution for General Clauses

- **General binary resolution calculus *Res*:**

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

$$\frac{C \vee A \vee B}{(C \vee A)\sigma} \quad \text{if } \sigma = \mathrm{mgu}(A, B) \quad (\text{positive factoring})$$

- **General resolution calculus *RIF* with implicit factoring:**

$$\frac{C \vee A_1 \vee \ldots \vee A_n \qquad D \vee \neg B}{(C \vee D)\sigma} \quad (\text{RIF})$$

$$\text{if } \sigma = \mathrm{mgu}(A_1, \ldots, A_n, B)$$

Resolution for General Clauses (cont'd)

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

Lifting Lemma

Lemma 1

Let C and D be variable-disjoint clauses. If

$$\begin{array}{cc} C & D \\ \downarrow \sigma & \downarrow \rho \\ \hline \multicolumn{2}{c}{} \end{array}$$

$$\frac{C\sigma \qquad D\rho}{C'} \quad (\text{propositional resolution})$$

then there exist C'' and a substitution τ such that

$$\frac{C \qquad D}{C''} \quad (\text{general resolution})$$

$$\downarrow \tau$$

$$C' = C''\tau$$

Lifting Lemma (cont'd)

- An analogous lifting lemma holds for factoring.

Saturation of Sets of General Clauses

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

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

$$Res(G_\Sigma(N)) \subseteq G_\Sigma(N).$$

Saturation of Sets of General Clauses (cont'd)

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

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

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

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

Herbrand's Theorem

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

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

Herbrand's Theorem (cont'd)

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

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

$$
\begin{aligned}
N \not\models \bot \;&\Rightarrow\; \bot \notin \mathit{Res}^*(N) && \text{(resolution is sound)}\\
&\Rightarrow\; \bot \notin G_\Sigma(\mathit{Res}^*(N))\\
&\Rightarrow\; I_{G_\Sigma(\mathit{Res}^*(N))} \models G_\Sigma(\mathit{Res}^*(N)) && \text{(Prt. 12 (BG90); Cor.}\\
&\Rightarrow\; I_{G_\Sigma(\mathit{Res}^*(N))} \models \mathit{Res}^*(N) && \text{(Lemma 4)}\\
&\Rightarrow\; I_{G_\Sigma(\mathit{Res}^*(N))} \models N && (N \subseteq \mathit{Res}^*(N))
\end{aligned}
$$

The Theorem of Löwenheim-Skolem

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

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

Refutational Completeness of General Resolution

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

$$N \models \bot \;\text{ iff }\; \bot \in N.$$

Proof:
Let $\mathit{Res}(N) \subseteq N$.
By Corollary 2: $\mathit{Res}(G_\Sigma(N)) \subseteq G_\Sigma(N)$

$$
\begin{aligned}
N \models \bot \;&\Leftrightarrow\; G_\Sigma(N) \models \bot && \text{(Lemmas 3 \& 4; Property 5)}\\
&\Leftrightarrow\; \bot \in G_\Sigma(N) && \text{(prop. resol. is sound and complete)}\\
&\Leftrightarrow\; \bot \in N
\end{aligned}
$$

Compactness of First-Order Logic

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

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

Lifting Lemma for $Res_S^{\succ}$

Lemma 9

Let C and D be variable-disjoint clauses. If

$$\begin{array}{ccc} C & & D \\ \downarrow \sigma & & \downarrow \rho \\ \dfrac{C\sigma \qquad\qquad D\rho}{C'} & & \end{array} \qquad \text{(propositional inference in } Res_S^{\succ}\text{)}$$

and if $S(C\sigma) \simeq S(C)$, $S(D\rho) \simeq S(D)$ (that is, "corresponding" literals are selected), then there exist C'' and a substitution τ s.t.

$$\begin{array}{c} \dfrac{C \qquad D}{C''} \\ \downarrow \tau \\ C' = C''\tau \end{array} \qquad \text{(inference in } Res_S^{\succ}\text{)}$$

Lifting Lemma for $Res_S^{\succ}$ (cont'd)

- An analogous lifting lemma holds for factoring.

Saturation of General Clause Sets

Corollary 10

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

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

Proof:

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

Soundness and Refutational Completeness

Property 11

Let $\succ$ be an atom ordering and S a selection function such that $Res_S^{\succ}(N) \subseteq N$. Then

$$N \models \bot \ \text{ iff } \ \bot \in N$$

Proof:

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

Summary

- Resolution for general, first-order clauses
- Refutational completeness, consequence of:
  1. refutational completeness of ground case
  2. lifting lemmas
  3. Herbrand's Theorem
- Löwenheim–Skolem Theorem
- Compactness of FOL
- lifting lemmas for ordered resolution with selection

COMP6012: Automated Reasoning II

Optional material (unassessed)

Lexicographic Orderings

- **Lexicographic orderings:** Let $(X_1, \succ_1)$, $(X_2, \succ_2)$ be well-founded orderings. Define their lexicographic combination

$$\succ \; = (\succ_1, \succ_2)_{\text{lex}}$$

as an ordering on $X_1 \times X_2$ such that

$$(x_1, x_2) \succ (y_1, y_2) \quad \text{iff} \quad \begin{array}{l} \text{(i) } x_1 \succ_1 y_1, \text{ or else} \\ \text{(ii) } x_1 = y_1 \text{ and } x_2 \succ_2 y_2 \end{array}$$

(Analogously for more than two orderings.)

This again yields a well-founded ordering (proof below).

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

Lexicographic Orderings: Examples

- **Length-based ordering on words:** For alphabets Σ with a well-founded ordering $>_\Sigma$, the relation $\succ$, defined as

$$w \succ w' \quad \text{iff} \quad \begin{array}{l} \text{(i) } |w| > |w'| \text{ or} \\ \text{(ii) } |w| = |w'| \text{ and } w >_{\Sigma,\text{lex}} w', \end{array}$$

is a well-founded ordering on Σ^*.

- **Notation:** $>_{\Sigma,\text{lex}} = (>_\Sigma)_{\text{lex}}$.

Lexicographic Combinations of Well-Founded Orderings

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

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

Reduction orderings

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

Lexicographic Path Orderings

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

Lexicographic Path Orderings (cont'd)

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

Properties of LPOs

Lemma 13

$s \succ_{\mathsf{lpo}} t$ implies $\mathrm{var}(s) \supseteq \mathrm{var}(t)$.

Proof:

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

Properties of LPOs (cont'd)

Property 14

$\succ_{\mathsf{lpo}}$ is a reduction ordering on the set of terms (and atoms) over Σ and X.

Proof:

Show transitivity, stability under substitutions, compatibility contexts, and irreflexivity, usually by induction on the sum of the term sizes and case analysis.

Details: Baader and Nipkow, page 119–120.

Properties of LPOs (cont'd)

Property 15

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

Proof:

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

Variations of the LPO

There are several possibilities to compare subterms in 2.(c):

- compare list of subterms lexicographically left-to-right ("lexicographic path ordering (lpo)", Kamin and Lvy)
- compare list of subterms lexicographically right-to-left (or according to some permutation π)
- compare multiset of subterms using the multiset extension ("multiset path ordering (mpo)", Dershowitz)
- to each function symbol f/n associate a status $\in \{mul\} \cup \{\, lex_\pi \mid \pi : \{1, \ldots, n\} \to \{1, \ldots, n\} \,\}$ and compare according to that status ("recursive path ordering (rpo) with status")

COMP6012: Automated Reasoning II

Lecture 1

Previously ...

- Resolution theorem proving
- Construction of candidate models

Semantic Tableaux

- Literature:
  - M. Fitting (1996), First-Order Logic and Automated Theorem Proving. Springer, Chapters 3, 6, 7.
  - R. M. Smullyan (1968), First-Order Logic. Dover Publ., New York, revised 1995.
- Like resolution, semantic tableaux were developed in the sixties, by R. M. Smullyan on the basis of work by Gentzen in the 1930s and of Beth in the 1950s.
- (According to Fitting, semantic tableaux were first proposed by the Polish scientist Z. Lis in (1960) (Studia Logica 10), that was only recently rediscovered.)

Idea for the propositional case

- A set $N \cup \{F \wedge G\}$ of formulae has a model iff $N \cup \{F \wedge G, F, G\}$ has a model.
- A set $N \cup \{F \vee G\}$ of formulae has a model iff $N \cup \{F \vee G, F\}$ or $N \cup \{F \vee G, G\}$ has a model.
- Similarly for other connectives.
- To avoid duplication, represent sets as paths of a tree.
- Continue splitting until two complementary formulae 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. $\neg R$
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	

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 of tableau calculi

- Tableau calculi are:
  - 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 formulae), as we will see later.

Application of Expansion Rules

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

Propositional Tableau Expansion Rules

- **Negation Elimination**

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

- **α-Expansion**
  For formulae that are essentially conjunctions:
  append subformulae α_1 and α_2 one on top of the other

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

Propositional Tableau Expansion Rules (cont'd)

- **β-Expansion**
  For formulae 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 Formulae

- Definition of α- and β-formulae:

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.

Definition of Semantic Tableau

- A semantic tableau is a marked (by formulae), finite, unordered tree and inductively defined as follows.
- Let $\{F_1, \ldots, F_n\}$ be a set of formulae.
  (i) The tree consisting of a single branch

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

  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 $\{F_1, \ldots, F_n\}$.

Definition of Semantic Tableau (cont'd)

- A branch (i.e. a path from the root to a leaf) in a tableau is called closed, if it either contains $\perp$, or else it contains both F and $\neg F$ for some formula F.
  If F is atomic we speak of atomic closure.
  Otherwise the branch is called open.
- A tableau is called closed, if all branches are closed.
- A tableau proof for F is a closed tableau for $\{\neg F\}$.

Further Notions

- A branch $\mathcal{B}$ in a tableau is called maximal (or complete), if for each non-atomic formula F on $\mathcal{B}$ there exists a node in $\mathcal{B}$ at which the expansion rule for F has been applied.
- In that case, if F is a formula on $\mathcal{B}$, $\mathcal{B}$ also contains:
  (i) F_1 and F_2, if F is an α-formula,
  (ii) F_1 or F_2, if F is a β-formula, and
  (iii) F', if F is a negated formula, where F' the conclusion of the corresponding elimination rule.
- A tableau is called maximal (or complete), if each branch is closed or maximal.

Further Notions (cont'd)

- A tableau is called strict, if for each formula the corresponding expansion rule has been applied at most once on each branch containing that formula.
- A tableau is called clausal, if each of its formulae is a clause.

A Sample Proof

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

1. $\neg\Big[\big(P \rightarrow (Q \rightarrow R)\big) \rightarrow \big((P \vee S) \rightarrow ((Q \rightarrow R) \vee S)\big)\Big]$
2. $P \rightarrow (Q \rightarrow R)$ $[\alpha, 1_1]$
3. $\neg\big((P \vee S) \rightarrow ((Q \rightarrow R) \vee S)\big)$ $[\alpha, 1_2]$
4. $P \vee S$ $[\alpha, 3_1]$
5. $\neg((Q \rightarrow R) \vee S))$ $[\alpha, 3_2]$
6. $\neg(Q \rightarrow R)$ $[\alpha, 5_1]$
7. $\neg S$ $[\alpha, 5_2]$

Branching after 7:
- 8. $\neg P$ $[\beta, 2_1]$
  - 10. P $[\beta, 4_1]$
  - 11. S $[\beta, 4_2]$
- 9. $Q \rightarrow R$ $[\beta, 2_2]$

There are three branches, each of them closed.

Properties of Propositional Tableaux

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

Property 16
$\{F_1, \ldots, F_n\}$ satisfiable iff some branch (i.e. the set of its formulae) in T is satisfiable.

(Proof by induction over the structure of T.)

Corollary 17
T closed $\Rightarrow \{F_1, \ldots, F_n\}$ unsatisfiable

Properties of Propositional Tableaux (cont'd)

Property 18
Let T be a strict propositional tableau. Then T is finitely bounded.

Proof:
New formulae resulting from expansion are either $\bot$, $\top$ or subformulae of the expanded formula.
By strictness, on each branch in T a formula was expanded at most once.
Therefore, each branch is finitely bounded, and a finitely branching tree with finite branches is finite (König's Lemma).

- Conclusion: For propositional logic strict and maximal tableaux can be effectively constructed
  $\Rightarrow$ provides decision procedure for propositional logic

Refutational Completeness

Property 19
Let $\mathcal{B}$ be a maximal, open branch in a tableau. Then the set of formulae on $\mathcal{B}$ is satisfiable.

Proof (we consider only the case of a clausal tableau):
Let N be the set of formulae on $\mathcal{B}$. As $\mathcal{B}$ is open, $\bot$ is not in N.
Let $C \vee A$ and $D \vee \neg A$ be two resolvable clauses in N. One of the two subclauses C or D, C say, is not empty, as otherwise $\mathcal{B}$ would be closed.
Since $\mathcal{B}$ is maximal, the β-rule was applied on $C \vee A$. Therefore, $\mathcal{B}$ (and N) contains a proper subclause of $C \vee A$, and hence $C \vee A$ is redundant w. r. t. N.
By the same reasoning, if N contains a clause that can be factored, that clause must be redundant w. r. t. N.
In other words, N is saturated up to redundancy wrt. Res(olution). Now apply the Model Existence Theorem to conclude that N is satisfiable.

Refutational Completeness (cont'd)

Property 20
$\{F_1, \ldots, F_n\}$ satisfiable iff there exists no closed strict tableau for $\{F_1, \ldots, F_n\}$.

Proof:
One direction is clear by Property 16. For the reverse direction, let T be a strict, maximal tableau for $\{F_1, \ldots, F_n\}$ and let $\mathcal{B}$ be an open branch in T. By the previous theorem, the set of formulae on $\mathcal{B}$, and hence by Property 16 the set $\{F_1, \ldots, F_n\}$, is satisfiable.

Tableau Algorithm for Propositional Logic

- The validity of a propositional formula F can be established by constructing a strict, maximal tableau for $\{\neg F\}$:
  - T closed $\Leftrightarrow$ F valid.

- Algorithm for checking validity of F, or (un)satisfiability of $\neg F$:
  1. Start with $\neg F$.
  2. Repeatedly apply the expansion rules to the branches.
  3. Stop expanding a branch when either:
     (a) the branch is closed, or
     (b) the branch is maximal, open, i.e. no more non-redundant inferences are possible.
  4. Unless a maximal, open branch was found, or all branches are closed continue with step 2.

Consequences

- It suffices to test complementarity of branches wrt. atomic formulae (cf. reasoning in the proof of Property 19).
- 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 tableau size.
- Since it is sufficient to saturate branches wrt. ordered resolution (up to redundancy), tableau expansion rules can be even more restricted, in particular by certain ordering constraints.

Summary

- Semantic tableau
  - negation elimination expansion rules
  - α-expansion rules
  - β-expansion rules
- closed branch/tableau, open branch/tableau, maximal branch/tableau, strict tableau
- soundness & refutational completeness
- tableau decidability of propositional logic
- contruction of tableau derivations

COMP6012: Automated Reasoning II

Lecture 2

Previously ...

- Propositional semantic tableau
  - negation elimination expansion rules
  - α-expansion rules
  - β-expansion rules
- closedness, openness, maximality, strictness

Semantic Tableaux for First-Order Logic

- Additional classification of quantified formulae:

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

- Moreover we assume that the set of variables X is partitioned into 2 disjoint, infinite subsets X_b and X_f, so that bound [free] variables can be chosen from X_b [X_f].
(This avoids the variable capturing problem.)
- We will denote variables in X_f by $v, v_1, v_2, \ldots$.

Additional Tableau Expansion Rules

- **γ-expansion**

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

- **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

Notes

- Skolemisation becomes part of the calculus and needs not necessarily be applied in a preprocessing step.
Of course, one could do Skolemisation 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 branch to any formula on the branch.
- In this form the rules go back to Hähnle and Schmitt (1994), The Liberalized δ-Rule in Free Variable Semantic Tableaux. In J. Automated Reasoning 13(2), 211–221.

Definition: Free-Variable Tableau

Let $\{F_1, \ldots, F_n\}$ be a set of closed formulae.

(i) The tree consisting of a single branch

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

is a tableau for $\{F_1, \ldots, F_n\}$.

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

(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 $\{F_1, \ldots, F_n\}$.

Notes on Free-Variable Tableau

- The substitution rule (iii) may, potentially, modify all the formulae of a tableau.
- This feature is what 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 formulae.
- That would be a major drawback compared with resolution. Fortunately, we can improve on this.

Example

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

- 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 $\sigma = \{v_1/a,\;\; v_2/b(a),\;\; v_3/f(b(a), a)\}$.
- Hence, $T\sigma$ is a closed (linear) tableau for the formula in 1.

AMGU-Tableaux

- First-order free-variable tableau: MGU substitution rule.
  - Restriction of the substitution rule to unifiers of complementary formulae.
- AMGU-Tableau:
  - Further restriction: the following A(tomic)MGU (substitution) rule is applied.

    The substitution rule is only applied for substitutions σ for which there is a branch in T containing two literals $\neg A$ and B such that $\sigma = \mathrm{mgu}(A, B)$.

Soundness (optional)

- Given a signature Σ, by $\Sigma^{\mathbf{sko}}$ we denote the result of adding infinitely many new Skolem function symbols which we may use in the δ-rule.
- Let $\mathcal{M}$ be a $\Sigma^{\mathbf{sko}}$-interpretation, T a tableau, and s a variable assignment over $\mathcal{M}$.
- T is called $(\mathcal{M}, s)$-satisfiable, if there is a branch $\mathcal{B}_s$ in T such that $\mathcal{M}, s \models F$, for each formula F on $\mathcal{B}_s$.
- T is called satisfiable if there exists a structure $\mathcal{M}$ such that for each assignment s the tableau T is $(\mathcal{M}, s)$-satisfiable. (This implies that we may choose $\mathcal{B}_s$ depending on s.)

Soundness (cont'd)

Property 21

Let T be a tableau for $\{F_1, \ldots, F_n\}$, where the F_i are closed Σ-formulae. Then

$$\{F_1, \ldots, F_n\} \text{ is satisfiable iff } T \text{ is satisfiable.}$$

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

Meaning of Strictness?

- Forbidding re-use of γ-formulae is incomplete:

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

$$\begin{array}{lll|lll} 5. & \neg p(a) & 3_1 & 6. & \neg p(b) & 3_2 \end{array}$$

- If we placed a once-only restriction on applications to γ formulae, the tableau is now only expandable by the substitution rule.
- However, there is no substitution (for v_1) that can close both branches simultaneously.

Multiple Application of γ Solves the Problem

- Apply γ-rule again with 2:

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

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

- 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 $\{v_1/a,\ v_2/b\}$ which closes the tableau.

Strictness in AMGU-Tableau

- 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 branch.

Refutational Completeness

Property 22

$\{F_1, \ldots, F_n\}$ satisfiable iff there exists no closed, strict AMGU-tableau for $\{F_1, \ldots, F_n\}$.

Proof outline:

One defines a fair tableau expansion process converging against an infinite tableau where on each branch each γ-formula is expanded into all its variants (modulo the choice of the free variable).
One may then again show that each branch 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?

Property 23

There is no recursive function $f : For_\Sigma \times For_\Sigma \longrightarrow \mathbb{N}$ such that, if the closed formula F is unsatisfiable, then there exists a closed tableau for F, where to all formulae $\forall x\, G$ appearing in T the γ-rule is applied at most $f(F, \forall x G)$ times on each branch 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.
$\forall$ is treated like an infinite conjunction: By repeatedly applying γ, together with the substitution rule, one can enumerate all instances $F[x/t]$ vertically, that is, conjunctively, in each branch containing $\forall x F$.

Summary

- Free-variable tableau
  - rules of propositional tableau
  - γ-expansion rules
  - δ-expansion rules
  - substitution rule
- AMGU tableau
  - AMGU substitution rule
- strictness in first-order tableau
  - re-use of γ formulae
- soundness & refutational completeness

Semantic Tableaux vs. Resolution

- Both methods are machine methods upon which present day provers are based.
- Tableaux: global, goal-oriented, “backward” approach.
- Resolution: local, “forward” approach.
- Goal-orientation is a clear advantage if only a small subset of a large set of formulae is necessary for a proof. In general, resolution provers saturate also those parts of the clause set that are irrelevant for proving the goal.
- Resolution is very versatile; good for:
  - saturation of theories, yielding simplified theories
  - developing decision procedures
  - can be used for model generation, hierarchical theorem proving, ...

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 (beyond the scope of this course) and algebraic structures; for tableaux this seems to be harder, if not impossible.