

Automated Reasoning

Georg Moser

Institute of Computer Science @ UIBK

Winter 2013



Summary

Correction

Definition (splitting rule)

suppose the clause set C' can be written as

$C' = \{A_1, \dots, A_n, B_1, \dots, B_m\} \cup \mathcal{D}$ where

- 1 \exists literal L , such that neither L nor $\neg L$ occurs in \mathcal{D}
- 2 L occurs in any A_i (but in no B_j); A'_i is the result of removing L
- 3 $\neg L$ occurs in any B_j (but in no A_i); B'_j is the result of removing $\neg L$
- 4 rule consists in splitting C' into $C'_1 := \{A'_1, \dots, A'_n\} \cup \mathcal{D}$ and $C'_2 := \{B'_1, \dots, B'_m\} \cup \mathcal{D}$

Summary

Summary Last Lecture

Gilmore's Prover in Pseudo-Code

```
begin {
  contr := false;
  n := 0;
  while (not contr) do {
    D' := DNF(C'_n);
    contr := all constituents of D'
             contain complementary literals;
    n := n + 1;
  }
}
```

GM (Institute of Computer Science @ UIBK)

Automated Reasoning

212/1

Summary

Method of Davis and Putnam in Pseudo-Code

```
if C does not contain function symbols
then apply DPLL(a)-DPLL(c) on C'_0
else {
  n := 0;
  contr := false;
  while (not contr) do {
    apply DPLL(a)-DPLL(c) on C'_n;
    if the decision tree proves unsatisfiability,
    then contr := true
    else contr := false;
    n := n + 1;
  }
}
```

Definition

$$\text{resolution} \quad \frac{C \vee A \quad D \vee \neg B}{(C \vee D)\sigma}$$

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

σ is a mgu of the **atomic** formulas A and B

let \mathcal{C} be a set of clauses; define **resolution operator** $\text{Res}(\mathcal{C})$

- $\text{Res}(\mathcal{C}) = \{D \mid D \text{ is resolvent or factor with premises in } \mathcal{C}\}$
- $\text{Res}^0(\mathcal{C}) = \mathcal{C}$; $\text{Res}^{n+1}(\mathcal{C}) := \text{Res}^n(\mathcal{C}) \cup \text{Res}(\text{Res}^n(\mathcal{C}))$
- $\text{Res}^*(\mathcal{C}) := \bigcup_{n \geq 0} \text{Res}^n(\mathcal{C})$

Theorem

resolution is sound and complete

Outline of the Lecture

Early Approaches in Automated Reasoning

short recollection of Herbrand's theorem, Gilmore's prover, method of Davis and Putnam

Starting Points

resolution, **tableau provers**, structural Skolemisation, redundancy and deletion

Automated Reasoning with Equality

ordered resolution, paramodulation, ordered completion and proof orders, superposition

Applications of Automated Reasoning

Neuman-Stubblebinde Key Exchange Protocol, group theory, resolution and paramodulation as decision procedure, ...

Tableau Expansion Rules

Definition (uniform notation)

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

Definition (tableau expansion rules)

$$\frac{\neg\neg A}{A} \qquad \frac{\alpha}{\alpha_1 \quad \alpha_2} \qquad \frac{\beta}{\beta_1 | \beta_2}$$

Propositional Semantic Tableaux

Definition

let $\{A_1, \dots, A_n\}$ be propositional formulas

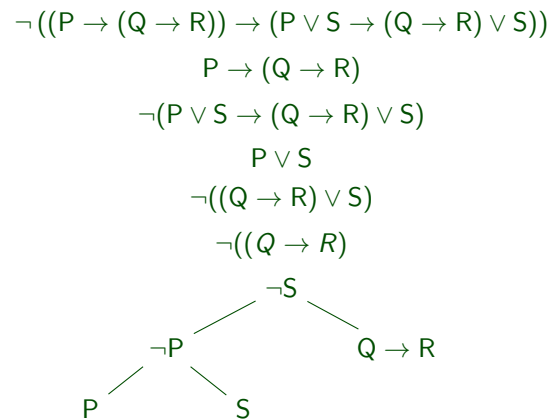
- the following tree T is a **tableau** for $\{A_1, \dots, A_n\}$:

$$\begin{array}{c} A_1 \\ A_2 \\ \vdots \\ A_n \end{array}$$

- suppose T is a tableau for $\{A_1, \dots, A_n\}$ and T^* is obtained by applying a tableau expansion rule to T , then T^* is a **tableau** for $\{A_1, \dots, A_n\}$

Example

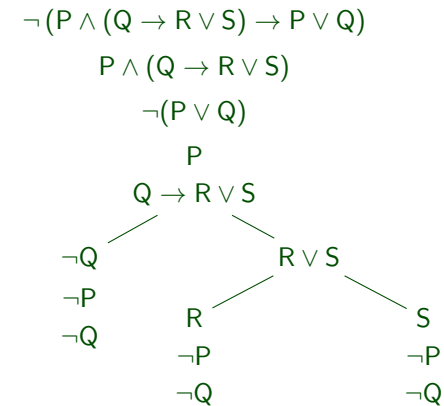
consider the tableau proof of $(P \rightarrow (Q \rightarrow R)) \rightarrow (P \vee S \rightarrow (Q \rightarrow R) \vee S)$



Heuristics Matters

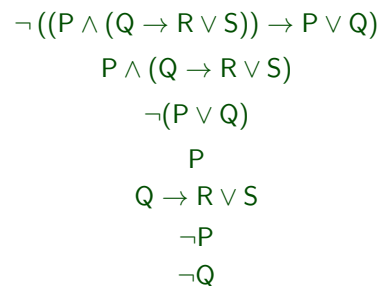
Example

consider $P \wedge (Q \rightarrow R \vee S) \rightarrow P \vee Q$ and the following tableau proof



Example (cont'd)

now consider the following tableau proof



Soundness and Completeness

Definitions

- a branch is **closed** if the formulas F and $\neg F$ occur on it
- if F is atomic, then the branch is said to be **atomically closed**
- a tableau is **closed** if every branch is closed
- a **tableau proof** of F is a closed tableau for $\neg F$
- in a **strict** tableau no formula is expanded twice on the same branch

Theorem

the tableau procedure is **sound** and **complete**:

$$F \text{ is a tautology} \iff F \text{ has a tableau proof}$$

Lemma

any application of a tableau expansion rule to a satisfiable tableau yields another satisfiable tableau

Strong Propositional Completeness

Lemma

suppose F is a valid; a strict tableau construction for $\neg F$ that is continued as long as possible must terminate in an atomically closed tableau

Proof.

- 1 any strict tableau construction for $\neg F$ has to terminate
- 2 suppose T is a strict tableau for $\neg F$ that is not atomically closed
- 3 \exists a branch in T which does not contain conflicting literals
- 4 \forall non-literals, all possible expansion rules have been applied
- 5 an assignment for $\neg F$ can be read off from the branch
- 6 contradiction to validity of F



Implementation of Semantic Tableaux

Naive Approach

```
tableau_prover(X) :-
    expand([[neg X]], Y),
    closed(Y).
```

Slightly More Efficient

```
tableau_prover2(X) :-
    expand([[neg X]], Y),
    !,
    closed(Y).
```

A Bit More Efficient

```
tableau_prover3(X) :-
    expand_and_close([[neg X]]).
```

First-Order Semantic Tableaux

Definition (uniform notation)

universal		existential	
γ	$\gamma(t)$	δ	$\delta(t)$
$\forall x A(x)$	$A(t)$	$\exists x A(x)$	$A(t)$
$\neg \exists x A(x)$	$\neg A(t)$	$\neg \forall x A(x)$	$\neg A(t)$

Definition (tableau expansion rules)

$\frac{\gamma}{\gamma(t)}$ t term in \mathcal{L}^+ $\frac{\delta}{\delta(k)}$ k fresh constant in \mathcal{L}^+

- 1 \mathcal{L}^+ denotes extension of base language \mathcal{L}
- 2 new individual constants are introduced in δ rules
- 3 **fresh** means new to the branch of the tableau

Example

consider $\forall x(P(x) \vee Q(x)) \rightarrow \exists x P(x) \vee \forall x Q(x)$
we give a tableau proof

$$\begin{array}{c}
 \neg (\forall x(P(x) \vee Q(x)) \rightarrow \exists x P(x) \vee \forall x Q(x)) \\
 \forall x(P(x) \vee Q(x)) \\
 \neg (\exists x P(x) \vee \forall x Q(x)) \\
 \neg \exists x P(x) \\
 \neg \forall x Q(x) \\
 \neg Q(c) \\
 \neg P(c) \\
 P(c) \vee Q(c) \\
 \swarrow \quad \searrow \\
 P(c) \quad Q(c)
 \end{array}$$

First-Order Soundness

Definitions

- a **tableau proof** of F is a closed tableau for $\neg F$
- a tableau branch is **satisfiable** if the set \mathcal{G} of sentences on it is satisfiable, i.e., there exists a model of \mathcal{G}
- a tableau is **satisfiable** if some branch is satisfiable

Theorem

if F has a tableau proof, then F is valid

Proof.

if any tableau expansion rule is applied to a satisfiable tableau, the result is satisfiable

First-Order Completeness

Theorem

if a sentence F is valid, then F has a tableau proof

Proof.

- 1 a set \mathcal{G} is **tableau consistent** if there is no closed tableau for \mathcal{G}
- 2 the collection of all tableau consistent sets fulfils the satisfaction properties

Free-Variable Semantic Tableaux

Definition (expansion rules)

$$\frac{\gamma}{\gamma(x)} \quad x \text{ a free variable} \qquad \frac{\delta}{\delta(f(x_1, \dots, x_n))} \quad f \text{ a Skolem function}$$

- x_1, \dots, x_n denote all free variables of the formula δ
- Skolem function f must be new on the branch

Remark

- δ -rule leaves a lot of room for improvement
- requirement that f must be new on the branch forces the introduction of inefficiently many new Skolem functions
- prevented with cleverer notions of the δ -rule

Definition (atomic closure rule)

- 1 \exists branch in tableau T that contains two literals A and $\neg B$
- 2 \exists mgu σ of A and B
- 3 then $T\sigma$ is also a tableau

Definition

consider the following **tableau substitution rule**:

- 1 T is a tableau for \mathcal{G}
- 2 σ is free for any sentence in \mathcal{G}
- 3 then $T\sigma$ is also a tableau

Remark

completeness of **free-variable tableaux** follows again via model existence

Soundness of Free-Variable Tableaux

Definition

- a branch in a free-variable tableau is called **satisfiable**, if \exists structure \mathcal{A} and \forall environment ℓ : $(\mathcal{A}, \ell) \models \mathcal{G}$
- a free-variable tableau is **satisfiable**, if there exists a satisfiable branch

Lemma

- T be a satisfiable (free-variable) tableau
- propositional or (free-variable) first-order expansion rule is applied
- then the result is satisfiable

Proof

the lemma follows by case-distinction on the applied expansion rule, it suffices to consider the δ -rule all other cases are similar

Proof (cont'd).

- suppose B is a satisfiable branch in T such that δ occurs on B
- extend B with $\delta(f(x_1, \dots, x_n))$ and call the result B' ; T' denotes the corresponding tableau
- \mathcal{G} collects all formulas on B and assume $(\mathcal{A}, \ell) \models \mathcal{G}$
- let x denote the existentially bound variable x replaced by the term $f(x_1, \dots, x_n)$
- \exists witness $a \in \mathcal{A}$ for x such that $(\mathcal{A}, \ell\{x \mapsto a\}) \models \delta(x)$
- construct \mathcal{A}' such that

$$f^{\mathcal{A}'}(\ell(x_1), \dots, \ell(x_n)) := a$$

- extendable to a total definition of $f^{\mathcal{A}'}$
- we conclude

$$(\mathcal{A}, \ell) \models \delta \implies (\mathcal{A}', \ell) \models \delta(f(x_1, \dots, x_n))$$

Lemma

if the atomic closure rule is applicable to a tableau T and T is satisfiable, then the result is also satisfiable

Proof.

- we show a more general statement:
if the substitution rule is applied to a satisfiable tableau T , then its result is satisfiable
- \forall environments ℓ , \exists environment ℓ' such that $t^{(\mathcal{A}, \ell')} = t_{\sigma}^{(\mathcal{A}, \ell)}$
- we have to show that $T\sigma$ is satisfiable
- this follows from the observation and definition of satisfiability

Theorem

if the sentence F has a free-variable tableau proof, then F is valid

Strong Completeness of Free-Variable Tableaux

NB: may consider a sequence of atomic closure rules that leads to an (atomically closed) tableau as one block

Definition

- T be a tableau with branches B_1, \dots, B_n
- $\forall i$ A_i and $\neg B_i$ are literals on B_i
- if σ is a mgu of $A_1 = B_1, \dots, A_n = B_n$
- then σ is called **most general atomic closure substitution**

Lemma (Lifting Lemma)

- τ a substitution free for tableau T such that each branch in $T\tau$ is atomically closed
- then \exists a most general atomic closure substitution σ and
- $T\sigma$ is closed by n applications of the atomic closure rule

Definition

a **strategy** S details:

- 1 which expansion rule is supposed to be applied
- 2 or that no expansion rule can be applied

a strategy may use extra information which is updated

Definition

a strategy S is **fair** if for sequence of tableaux T_1, T_2, \dots following S :

- 1 any non-literal formula in T_i is eventually expanded, and
- 2 any γ -formula occurrence in T_i has the γ -rule applied to it **arbitrarily often**

Example

strategy employed in the implementation of free-variable tableaux is fair

Example

- for each tableau the extra information includes
 - 1 which formula occurrences have been used on which branch
 - 2 priority order for formula occurrences on each branch
 - 3 priority order for branches
- extra information for initial tableau
 - 1 $\neg F$ is not used
 - 2 $\neg F$ has top priority
 - 3 single branch has top priority
- select branch of highest priority with unused formula
- select formula occurrence on this branch of highest priority
- apply expansion rule; give formula occurrence and branch lowest priority
- if every non-literal formula has been used on any branch no continuation is possible

this strategy is **not** fair

Theorem (Strong Completeness)

- 1 S be a fair strategy
- 2 F be a valid sentence
- 3 F has a tableau proof with the following properties:
 - all tableau expansion rules are considered first and follow strategy S
 - a block of atomic closure rules closes the tableau

Proof Sketch.

- 1 we argue indirectly and suppose that a given formula F does not admit a tableau proof
- 2 \exists open branch starting with $\neg F$
- 3 set of formulas \mathcal{G} on this branch admit the closure properties
- 4 hence $\neg F$ is satisfiable

Implementation of γ -Rule

γ -rule (simplified)

```
singlestep([OldBranch | Rest], NewTree) :-
    member(NotatedGamma, OldBranch),
    notation(NotatedGamma, Free),
    fmla(NotatedGamma, Gamma),
    is_universal(Gamma),
    remove(NotatedGamma, OldBranch, TempBranch),
    NewFree = [V | Free],
    instance(Gamma, V, GammaInstance),
    notation(NotatedGammaInstance, NewFree),
    fmla(NotatedGammaInstance, GammaInstance),
    append([NotatedGammaInstance | TempBranch],
           [NotatedGamma], NewBranch),
    append(Rest, [NewBranch], NewTree).
```