Introduction to Logic in Computer Science: Autumn 2006

Ulle Endriss
Institute for Logic, Language and Computation
University of Amsterdam

Plan for Today

Today's class will be an introduction to analytic tableaux for classical first-order logic:

- Quick review of syntax and semantics of first-order logic
- Quantifier rules for Smullyan-style and KE-style tableaux
- Soundness and completeness proofs
- Discussion of efficiency issues, undecidability
- Countermodel generation

Syntax of FOL

The *syntax* of a language defines the way in which basic elements of the language may be put together to form clauses of that language. In the case of FOL, the basic ingredients are (besides the logic operators): *variables*, *function symbols*, and *predicate symbols*. Each function and predicate symbol is associated with an *arity* $n \geq 0$.

Definition 1 (Terms) *We inductively define the set of terms as the smallest set such that:*

(1) every variable is a term;

(2) if f is a function symbol of arity k and $t_1, \ldots, t_k$ are terms, then $f(t_1, \ldots, t_k)$ is also a term.

Function symbols of arity 0 are better known as *constants*.

Syntax of FOL (2)

Definition 2 (Formulas) *We inductively define the set of formulas as the smallest set such that:*

(1) if P is a predicate symbol of arity k and $t_1 \ldots, t_k$ are terms, then $P(t_1, \ldots, t_k)$ is a formula;

(2) if φ and ψ are formulas, so are $\neg\varphi$, $\varphi \wedge \psi$, $\varphi \vee \psi$, and $\varphi \rightarrow \psi$;

(3) if x is a variable and φ is a formula, then $(\forall x)\varphi$ and $(\exists x)\varphi$ are also formulas.

Syntactic sugar: $\varphi \leftrightarrow \psi \equiv (\varphi \rightarrow \psi) \wedge (\psi \rightarrow \varphi)$; $\top \equiv P \vee \neg P$ (for an arbitrary 0-place predicate symbol P); $\bot \equiv \neg\top$.

Also recall: *atoms*, *literals*, *ground terms*, *bound* and *free variables*, *closed formulas* (aka *sentences*), ...

Semantics of FOL

The *semantics* of a language defines the meaning of clauses in that language. In the case of FOL, we do this through the notion of models (and variable assignments).

Definition 3 (Models) *A model is a pair $\mathcal{M} = (\mathcal{D}, \mathcal{I})$, where $\mathcal{D}$ (the domain) is a non-empty set of objects and $\mathcal{I}$ (the interpretation function) is mapping each n-place function symbol f to some n-ary function $f^{\mathcal{I}} : \mathcal{D}^n \to \mathcal{D}$ and each n-place predicate symbol P to some n-ary relation $P^{\mathcal{I}} : \mathcal{D}^n \to \{true, false\}$.*

Note that this definition also covers the cases of 0-place function symbols (constants) and predicate symbols.

Semantics of FOL (2)

Definition 4 (Assignments) *A variable assignment over a domain $\mathcal{D}$ is a function g from the set of variables to $\mathcal{D}$.*

Definition 5 (Valuation of terms) *We define a valuation function $val_{\mathcal{I},g}$ over terms as follows:*

$$\begin{aligned} val_{\mathcal{I},g}(x) &= g(x) \quad \text{for variables } x \\ val_{\mathcal{I},g}(f(t_1,\ldots,t_n)) &= f^{\mathcal{I}}(val_{\mathcal{I},g}(t_1),\ldots,val_{\mathcal{I},g}(t_n)) \end{aligned}$$

Definition 6 (Assignment variants) *Let g and g' be assignments over $\mathcal{D}$ and let x be a variable, Then g' is called an x-variant of g iff $g(y) = g'(y)$ for all variables $y \neq x$.*

Semantics of FOL (3)

Definition 7 (Satisfaction relation) *We write $\mathcal{M}, g \models \varphi$ to say that the formula φ is satisfied in the model $\mathcal{M} = (\mathcal{I}, \mathcal{D})$ under the assignment g. The relation $\models$ is defined inductively as follows:*

(1) $\mathcal{M}, g \models P(t_1,\ldots,t_n)$ *iff* $P^{\mathcal{I}}(val_{\mathcal{I},g}(t_1),\ldots,val_{\mathcal{I},g}(t_n)) = true$;

(2) $\mathcal{M}, g \models \neg\varphi$ *iff not* $\mathcal{M}, g \models \varphi$;

(3) $\mathcal{M}, g \models \varphi \wedge \psi$ *iff* $\mathcal{M}, g \models \varphi$ *and* $\mathcal{M}, g \models \psi$;

(4) $\mathcal{M}, g \models \varphi \vee \psi$ *iff* $\mathcal{M}, g \models \varphi$ *or* $\mathcal{M}, g \models \psi$;

(5) $\mathcal{M}, g \models \varphi \to \psi$ *iff not* $\mathcal{M}, g \models \varphi$ *or* $\mathcal{M}, g \models \psi$;

(6) $\mathcal{M}, g \models (\forall x)\varphi$ *iff* $\mathcal{M}, g' \models \varphi$ *for all x-variants g' of g; and*

(7) $\mathcal{M}, g \models (\exists x)\varphi$ *iff* $\mathcal{M}, g' \models \varphi$ *for some x-variant g' of g.*

Semantics of FOL (4)

Observe that in the case of *closed* formulas φ the variable assignment g does not matter (we just write $\mathcal{M} \models \varphi$).

Satisfiability. A closed formula φ is called *satisfiable* iff it has a model, i.e. there exists a model $\mathcal{M}$ with $\mathcal{M} \models \varphi$.

Validity. A closed formula φ is called *valid* iff for every model $\mathcal{M}$ we have $\mathcal{M} \models \varphi$. We write $\models \varphi$.

Consequence relation. Let φ be a closed formula and let Δ be a set of closed formulas. We write $\Delta \models \varphi$ iff whenever $\mathcal{M} \models \psi$ holds for all $\psi \in \Delta$ then also $\mathcal{M} \models \varphi$ holds.

Quantifier Rules

Both the KE-style and the Smullyan-style tableau method for propositional logic can be extended with the following rules.

Gamma Rules: $\dfrac{(\forall x)A}{A[x/t]}$ $\dfrac{\neg(\exists x)A}{\neg A[x/t]}$ **Delta Rules:** $\dfrac{(\exists x)A}{A[x/c]}$ $\dfrac{\neg(\forall x)A}{\neg A[x/c]}$

Here, t is an *arbitrary ground term* and c is a *constant symbol* that is *new* to the branch.

Unlike all other rules, the gamma rule may have to be applied more than once to the same formula on the same branch.

Substitution. $\varphi[x/t]$ denotes the formula we get by replacing each free occurrence of the variable x in the formula φ by the term t.

Smullyan's Uniform Notation

Formulas of universal (γ) and existential (δ) type:

γ	$\gamma_1(u)$
$(\forall x)A$	$A[x/u]$
$\neg(\exists x)A$	$\neg A[x/u]$

δ	$\delta_1(u)$
$(\exists x)A$	$A[x/u]$
$\neg(\forall x)A$	$\neg A[x/u]$

We can now state gamma and delta rules as follows:

$\dfrac{\gamma}{\gamma_1(t)}$ $\dfrac{\delta}{\delta_1(c)}$

where:

- t is an arbitrary ground term
- c is a constant symbol new to the branch

Exercises

Give Smullyan-style or KE-style tableau proofs for the following arguments:

- $(\forall x)P(x) \vee (\forall x)Q(x) \models \neg(\exists x)(\neg P(x) \wedge \neg Q(x))$
- $\models (\exists x)(P(x) \vee Q(x)) \leftrightarrow (\exists x)P(x) \vee (\exists x)Q(x)$

Soundness and Completeness

Let φ be a first-order formula and Δ a set of such formulas. We write $\Delta \vdash \varphi$ to say that there exists a closed tableau for $\Delta \cup \{\neg\varphi\}$.

Theorem 1 (Soundness) *If* $\Delta \vdash \varphi$ *then* $\Delta \models \varphi$.

Theorem 2 (Completeness) *If* $\Delta \models \varphi$ *then* $\Delta \vdash \varphi$.

We shall prove soundness and completeness only for Smullyan-style tableaux (but it's almost the same for KE-style tableaux).

Important note: The mere *existence* of a closed tableau does not mean that we have an effective method of *finding* it! Concretely: we don't know how often we need to apply the gamma rule and what terms to use for the substitutions.

Proof of Soundness

This works exactly as in the propositional case ($\leadsto$ last week).

The central step is to show that each of the expansion rules preserves satisfiability:

- If a non-branching rule is applied to a satisfiable branch, the result is another satisfiable branch.
- If a branching rule is applied to a satisfiable branch, at least one of the resulting branches is also satisfiable.

Proof of Soundness (cont.)

Gamma rule: If γ appears on a branch, you may add $\gamma_1(t)$ for any ground term t to the same branch.

Proof: suppose branch B with $\gamma \equiv (\forall x)\gamma_1(x) \in B$ is satisfiable
$\Rightarrow$ there exists $\mathcal{M} = (\mathcal{D}, \mathcal{I})$ s.t. $\mathcal{M} \models B$ and hence $\mathcal{M} \models (\forall x)\gamma_1(x)$
$\Rightarrow$ for all var. assignments g: $\mathcal{M}, g \models \gamma_1(x)$; choose g' s.t. $g'(x) = t^{\mathcal{I}}$
$\Rightarrow \mathcal{M}, g' \models \gamma_1(x) \;\Rightarrow\; \mathcal{M} \models \gamma_1(t) \;\Rightarrow\; \mathcal{M} \models B \cup \{\gamma_1(t)\}$ ✓

Delta rule: If δ appears on a branch, you may add $\delta_1(c)$ for any new constant symbol c to the same branch.

Proof: suppose branch B with $\delta \equiv (\exists x)\delta_1(x) \in B$ is satisfiable
$\Rightarrow$ there exists $\mathcal{M} = (\mathcal{D}, \mathcal{I})$ s.t. $\mathcal{M} \models B$ and hence $\mathcal{M} \models (\exists x)\delta_1(x)$
$\Rightarrow$ there exists a variable assignment g s.t. $\mathcal{M}, g \models \delta_1(x)$
now suppose $g(x) = d \in \mathcal{D}$; define new model $\mathcal{M}' = (\mathcal{D}, \mathcal{I}')$ with $\mathcal{I}'$ like $\mathcal{I}$ but additionally $c^{\mathcal{I}'} = d$ (this is possible, because c is *new*)
$\Rightarrow \mathcal{M}' \models \delta_1(c)$ and $\mathcal{M}' \models B \;\Rightarrow\; \mathcal{M}' \models B \cup \{\delta_1(c)\}$ ✓

Hintikka's Lemma

Definition 8 (Hintikka set) *A set of first-order formulas H is called a Hintikka set provided the following hold:*

(1) not both $P \in H$ and $\neg P \in H$ for propositional atoms P;

(2) if $\neg\neg\varphi \in H$ then $\varphi \in H$ for all formulas φ;

(3) if $\alpha \in H$ then $\alpha_1 \in H$ and $\alpha_2 \in H$ for alpha formulas α;

(4) if $\beta \in H$ then $\beta_1 \in H$ or $\beta_2 \in H$ for beta formulas β.

(5) for all terms t built from function symbols in H (at least one constant symbol): if $\gamma \in H$ then $\gamma_1(t)$ for gamma formulas γ;

(6) if $\delta \in H$ then $\delta_1(t) \in H$ for some term t, for delta formulas δ.

Lemma 1 (Hintikka) *Every Hintikka set is satisfiable.*

Proof of Hintikka's Lemma

Construct a model $\mathcal{M} = (\mathcal{D}, \mathcal{I})$ from a given Hintikka set H:

- $\mathcal{D}$: set of terms constructible from function symbols appearing in H (add one constant symbol in case there are none)
- $\mathcal{I}$: (1) function symbols are being interpreted "as themselves": $f^{\mathcal{I}}(d_1, \ldots, d_n) = f(d_1, \ldots, d_n)$; (2) predicate symbols: $P^{\mathcal{I}}(d_1, \ldots, d_n) = \mathit{true}$ iff $P(d_1, \ldots, d_n) \in H$

Claim: $\varphi \in H$ entails $\mathcal{M} \models \varphi$.

Proof: By structural induction. [...] ✓

Proof of Completeness

Fairness. We call a tableau proof *fair* iff every non-literal gets *eventually* analysed on every branch and, additionally, every gamma formula gets *eventually* instantiated with every term constructible from the function symbols appearing on a branch.

Proof sketch. We will show the contrapositive: assume $\Delta \nvdash \varphi$ and try to conclude $\Delta \not\models \varphi$.

If there is no proof for $\Delta \cup \{\neg\varphi\}$ (assumption), then there can also be no *fair* proof. Observe that any fairly constructed non-closable branch enumerates the elements of a Hintikka set H.

H is satisfiable (Hintikka's Lemma) and we have $\Delta \cup \{\neg\varphi\} \subseteq H$.

So there is a model for $\Delta \cup \{\neg\varphi\}$, *i.e.* we get $\Delta \not\models \varphi$. ✓

Efficiency Issues

Due to the undecidability of first-order logic there can be no general method for finding a closed tableau for a given theorem (although its existence is guaranteed by completeness).

Nevertheless, there are some heuristics:

- As in the propositional case, use "deterministic" rules first: propositional rules except PB and the delta rule.
- As in the propositional case, use *beta simplification*.
- Use the gamma rule a "reasonable" number of times (with "promising" substitutions) before attempting to use PB.

  > *Example:* for the automated theorem prover implemented in WinKE you can choose n, the maximum number of applications of the gamma rule on a given branch before PB will be used.

- Use analytic PB only.

Saturated Branches

An open branch is called *saturated* iff every non-literal has been analysed at least once and, additionally, every gamma formula has been instantiated with every term we can construct using the function symbols on the branch.

Failing proofs. A tableau with an open saturated branch can never be closed, *i.e.* we can stop an declare the proof a failure.

The solution? This only helps us in special cases though. (A single 1-place function symbol together with a constant is already enough to construct infinitely many terms ...)

Propositional logic. In propositional logic (where we have no gamma formulas), after a limited number of steps, every branch will be either closed or saturated. This gives us a decision procedure.

Countermodels

If a KE proof fails with a *saturated open branch*, you can use it to help you define a model $\mathcal{M}$ for all the formulas on that branch:

- domain: set of all terms we can construct using the function symbols appearing on the branch (so-called *Herbrand universe*)
- terms are interpreted as themselves
- interpretation of predicate symbols: see literals on branch

In particular, $\mathcal{M}$ will be a model for the premises Δ and the negated conclusion $\neg\varphi$, *i.e.* a *counterexample* for $\Delta \models \varphi$.

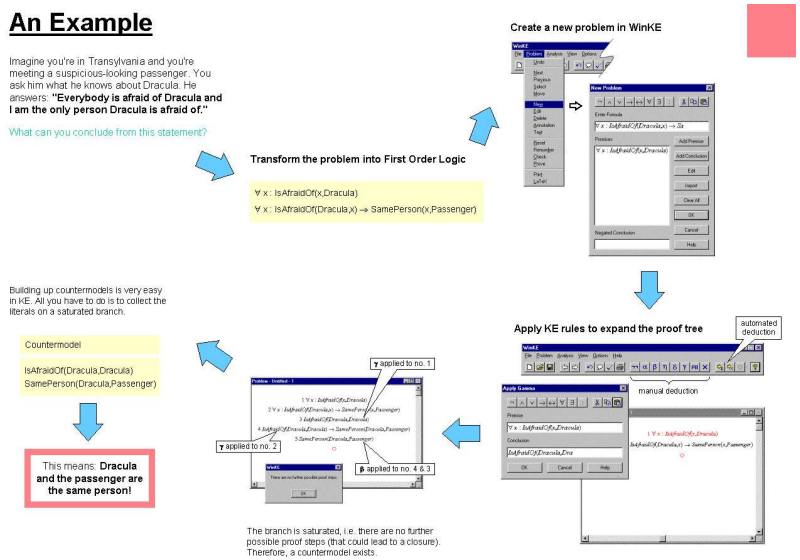
You can do the same with Smullyan-style tableaux, but for KE distinct open branches always generate distinct models.

Careful: There's a bug in WinKE: sometimes, what is presented as a countermodel is in fact only *part* of a countermodel (but it can always be extended to an actual model).

Exercise

Construct a counterexample for the following argument:

- $(\forall x)(P(x) \vee Q(x)) \models^{?} (\forall x)P(x) \vee (\forall x)Q(x)$



Summary

- Two tableau methods for first-order logic: Smullyan-style (syntactic branching) and KE-style (semantic branching)
- Soundness and completeness
- Undecidability: gamma rule is the culprit
- Countermodels: sometimes we get termination for failed proofs and can extract a counterexample (particularly nice for KE)