Automated Theorem Proving 2/4: First-Order Theorem Proving

A.L. Lamprecht

Course Program Semantics and Verfication 2020, Utrecht University

September 23, 2020

Lecture Notes

"Automated Reasoning" by Gerard A.W. Vreeswijk.
Available for download on the course website.
My slides are largely based on them.

In This Course

- Propositional theorem proving (last Monday),
  Chapter 2 of the lecture notes
- First-order theorem proving (today),
  Chapter 3 of the lecture notes
- Clause sets and resolution (next Monday),
  Chapters 4 and 5 of the lecture notes
- Satisfiability checkers, SAT/SMT (next Wednesday),
  Chapter 6 of the lecture notes, additional material

Recap: Propositional Theorem Proving

- The nature of theorem proving.
- Searching for counterexamples, refutation trees, semantic tableaux.
- Turning refutation trees into proofs.
- NP-completeness of propositional theorem proving.

Recap FOL

This lecture assumes familiarity with the syntax and semantics of first-order logics. In particular:

- well-formed formulas,
- scope of a quantifier,
- free and bound variables,
- closed well-formed formulas (sentences),
- fair substitutions,
- first-order domains,
- interpretation of constants, function symbols and predicate symbols,
- interpretation of well-formed formulas,
- variable assignments,
- first-order models,
- first-order countermodels

(Recap was homework.)

Recap FOL

- *Predicate logic*: first-order logic without further restrictions on the semantics of the of the formulas
- *Completeness of predicate logic* has been proven.
- *Church's thesis* on computability, connecting algorithms and symbol-manipulating mechanisms.
- *Undecidability of predicate logic* has been proven.
- *Semi-decidability of predicate logic*: There exist algorithms that prove precisely all formulas that are valid in the predicate logic. (Basis of ATP!)
- *Incompleteness of arithmetic*: Any first-order logic that is expressive as arithmetic cannot be axiomatized.

Recap FOL

- "first order": quantifiers range over variables
- "second order": quantifiers can also range over predicate variables
- ...

- Higher-order logics are more difficult to manage.
- Most higher-order theories can be translated into first-order theories.
- FOL suffices for the expression of most mathematical theories.
- Dealing with FOL is difficult enough.
- Most efforts of ATP research in this area.

Reduction Rules for FOL

(Additional) refutation rules that describe how quantified formulas can be made true or false:

Reduction rules for building a refutation tree

If $(\forall x)\phi$, then $\phi[t/x]$ for all terms t.	If $(\forall x)\phi$ is false, then $\phi[c/x]$ is false for some constant c.
If $(\exists x)\phi$, then $\phi[c/x]$ for some constant c.	If $(\exists x)\phi$ is false, then $\phi[t/x]$ is false for all terms t.

FOL Proofs: Idea

The idea behind making a universal statement $(\forall x)(P)$ true is that we make all instances true, one at a time:

$$
\begin{aligned}
(\forall x)(P) &\equiv (\forall x)(P) \wedge P[a/x] \quad (\text{spawn formula with } t = a) \\
&\equiv (\forall x)(P) \wedge P[a/x] \wedge P[b/x] \quad (\text{spawn with } t = b) \\
&\equiv (\forall x)(P) \wedge P[a/x] \wedge P[b/x] \wedge P[f(a)/x] \\
&\qquad\qquad (\text{spawn with } t = f(a)) \\
&\equiv (\forall x)(P) \wedge P[a/x] \wedge P[b/x] \wedge P[f(a)/x] \wedge P[f(b)/x] \\
&\qquad\qquad (\text{spawn with } t = f(b)) \\
&\vdots
\end{aligned}
$$

Herbrand Domain

- $(\forall x)p$ on the LHS may generate a potentially infinite number of *different* terms.
- A *ground term* is a term without variables.
- The *Herbrand domain* of a formula is the set of all possible ground terms that can be made with constants and function symbols that occur in the formula.
- If a term has no constants, then a fresh constant c_0 is used to prevent the Herbrand domain from being empty.
- Generalizable to sets of formulas and terms.

Herbrand Domain (Examples)

Set of formulas and terms	*Constants and function symbols*	*Herbrand domain*
$\{(\forall x)(p\,x, a)\}$	$\{a\}$	$\{a\}$
$\{(\forall x)(pf(x), a)\}$	$\{a, f\}$	$\{a, f(a), f(f(a)), \ldots\}$
$\{(\forall x)(pf(x))\}$	$\{f\}$	$\{c_0, f(c_0), f(f(c_0)), \ldots\}$
$\{(\forall x)(pg(x, y))\}$	$\{g\}$	$\{c_0, g(c_0, c_0),$ $g(g(c_0, c_0), c_0), \ldots\}$
$\{(\forall x)(pg(x, y)),\ qf(c_3)\}$	$\{f, g, c_3\}$	$\{c_3,\ f(c_3),\ g(c_3, c_3),$ $g(f(c_3), c_3),$ $g(c_3, f(c_3)),$ $f(g(c_3, c_3)),$ $g(g(c_3, c_3), c_3),\ \ldots\}$

Exercise

Determine the Herbrand domain of the following formulas.

1. Px, a
2. $(\forall x)(Px \supset Qf(x), a)$
3. $Rg(f(x), y), z$

Solution

1. $D(Px, a) = \{a\}$
2. $D((\forall x)Px \supset Qf(x), a) = \{f^n(a) | n \geq 0\} = \{a, f(a), f(f(a)), \ldots\}$
3. $D(Rg(f(x), y), z)$ is the set H such that $c_0 \in H$, $f(t) \in H$ if $t \in H$, and $g(t_1, t_2) \in H$ if $\{t_1, t_2\} \subseteq H$. I.e., $H = \{c_0, f(c_0), g(c_0, c_0), f(c_0), f^2(c_0), f(g(c_0, c_0)), \ldots$.

Analytic Refutation Rules for FOL

Previous rules, plus:

$$\begin{array}{l} (\forall x)\phi \circ \\ \quad | \\ \phi[t/x] \circ \end{array} \qquad \text{left-}\forall \text{ Term } t \text{ free for } x \text{ in } \phi$$

$$\begin{array}{l} \circ\ (\forall x)\phi \\ \quad | \\ \circ\ \phi[y/x] \end{array} \qquad \text{right-}\forall \ y \text{ does not occur in the current tree}$$

$$\begin{array}{l} (\exists x)\phi \circ \\ \quad | \\ \phi[y/x] \circ \end{array} \qquad \text{left-}\exists \ y \text{ does not occur in the current tree}$$

$$\begin{array}{l} \circ\ (\exists x)\phi \\ \quad | \\ \circ\ \phi[t/x] \end{array} \qquad \text{right-}\exists \text{ Term } t \text{ free for } x \text{ in } \phi$$

Gentzen System for FOL

Previous rules, plus:

$$\text{left-}\forall: \quad \frac{\ldots, (\forall x)\phi, \phi[t/x], \ldots \vdash \ldots}{\ldots, (\forall x)\phi, \ldots \vdash \ldots} \qquad \text{right-}\forall: \quad \frac{\ldots \vdash \ldots, \phi[c/x], \ldots}{\ldots \vdash \ldots, (\forall x)\phi, \ldots}$$

$$\text{left-}\exists: \quad \frac{\ldots, \phi[c/x], \ldots \vdash \ldots}{\ldots, (\exists x)\phi, \ldots \vdash \ldots} \qquad \text{right-}\exists: \quad \frac{\ldots \vdash \ldots, (\exists x)\phi, \phi[t/x], \ldots}{\ldots \vdash \ldots, (\exists x)\phi, \ldots}$$

FOL Reduction and Complexity

- Reduction rules of propositional logic reduce the complexity of the formula (sub-formula property).
- "left-$\forall$" and "right-$\exists$" lack this property, they do not reduce the complexity of the formula they operate on.
- In fact, reductions may go on forever and branches may grow indefinitely, which is inherent to the undecidability of predicate logic.
- However, never-ending scenarios are a worst-case scenario.
- In many cases, it is possible to guess with substitutions must be made to steer the refutation to an end.

Cases

In the following we will look at FOL theorem proving with:

- No functions and no equality
- Functions and no equality
- Functions and equality

No Functions and No Equality

- If a sentence contains no function and no equality symbols, its Herbrand domain is a finite but non-empty set of constants.
- Set may grow, but does not do so excessively.
- It is no problem to substitute all variables and constants that have been encountered in the refutation so far.

Example: $(\forall x)(p\,x) \vdash pa$

- Only applicable rule: left-$\forall$
- Herbrand domain: $\{a\}$, thus $t = a$

$$\begin{array}{c} (\forall x)(p\,x) \circ pa \\ \text{left-}\forall \\ | \\ (\forall x)(p\,x); \mathbf{pa} \circ \mathbf{pa} \\ \times \end{array}$$

Example: $p \vdash (\forall x)(qx)$

Only applicable rule: right-$\forall$, with a fresh constant c_1:

$$\begin{array}{c} p \circ (\forall x)(qx) \\ \text{right-}\forall \\ | \\ p \circ qc_1 \\ \lightning \end{array}$$

$\mathrm{LHS} \cap \mathrm{RHS} = \emptyset$, so that we have found a counterexample model M with domain $D = \{1\}$, such that c_1 and all other constants are mapped to 1, and

Predicate	*Extension*
p	*true*
q	$\emptyset$

Example: $(\forall x)(p\,x) \vdash (\exists x)(p\,x)$

- Two reductions possible: "left-$\forall$," and "right-$\exists$".
- For both, need to choose a term from the Herbrand domain.
- No such term, since the Herbrand domain of is empty.
- Use an arbitrary constant c_0 to kick off the refutation:

$$(\forall x)(p\,x) \circ (\exists x)(p\,x)$$
$$\text{left-}\forall$$
$$\Big|$$
$$(\forall x)(p\,x); pc_0 \circ (\exists x)(p\,x)$$
$$\text{right-}\exists$$
$$\Big|$$
$$(\forall x)(p\,x); \mathbf{pc_0} \circ (\exists x)(p\,x); \mathbf{pc_0}$$
$$\times$$

Example: $(\exists x)(p\,x) \vdash (\forall x)(p\,x)$

Refutation of the converse direction:

$$(\exists x)(p\,x) \circ (\forall x)(p\,x)$$
$$\text{left-}\exists$$
$$\Big|$$
$$pc_1 \circ (\forall x)(p\,x)$$
$$\text{right-}\forall$$
$$\Big|$$
$$pc_1 \circ pc_2$$
$$\lightning$$

Counterexample model M with:

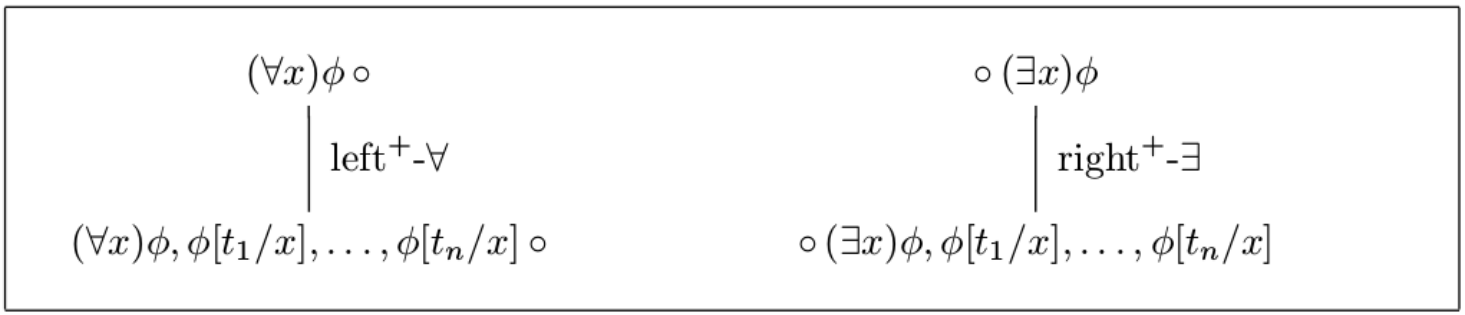
Predicate	*Extension*
p	$\{c_1\}$

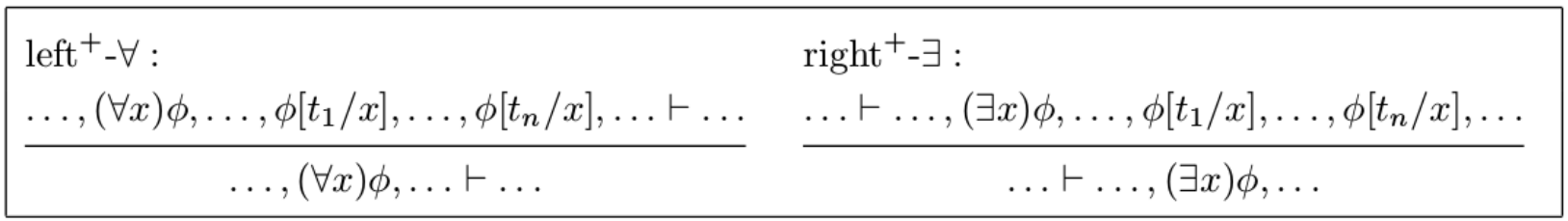
and domain $D = \{1, 2\}$, such that c_1 and c_2 are interpreted as 1 and 2. Then $M \models p(c_1)$ but $M \not\models p(c_2)$.

Many-on-One Variants of left-$\forall$ and right-$\exists$

Instead of using "left-$\forall$" and "right-$\exists$," use:

- "left^+-$\forall$", meaning one or more applications of "left-$\forall$",
- "right^+-$\exists$", meaning one or more applications of "right-$\exists$".





Do not enable reductions that would otherwise be impossible, but can reduce the size of the refutation trees.

Example

$$(\exists x)((\forall y)(p\,x, y)) \circ (\forall y)((\exists x)(p\,x, y))$$
left-$\exists$

$$(\forall y)(pc_1, y) \circ (\forall y)((\exists x)(p\,x, y))$$
left$^+$-$\forall$

$$(\forall y)(pc_1, y); pc_1, c_1 \circ (\forall y)((\exists x)(p\,x, y))$$
right-$\forall$

$$(\forall y)(pc_1, y); pc_1, c_1 \circ (\exists x)(p\,x, c_2)$$
left$^+$-$\forall$

$$(\forall y)(pc_1, y); pc_1, c_2; pc_1, c_1 \circ (\exists x)(p\,x, c_2)$$
right$^+$-$\exists$

$$(\forall y)(pc_1, y); \mathbf{pc_1}, \mathbf{c_2}; pc_1, c_1 \circ (\exists x)(p\,x, c_2); \mathbf{pc_1}, \mathbf{c_2}; pc_2, c_2$$
$\times$

Sound- and Completeness

- Soundness: if a sequent is falsifiable, then all refutation trees for that sequent have at least one branch that cannot be closed.
- Completeness: if a sequent is valid (i.e., not falsifiable), then every refutation tree closes.
- Sound- and completeness: a sequent is valid if and only if all refutations close.
- Proof sketch in the lecture notes.

Functions (No Equality)

- Function symbols complicate theorem proving, because it is possible to produce many terms with the help of only a few function symbols:

$$\mathrm{HerbrandDomain}(pf(a)) = \{a, f(a), f(f(a)), f(f(f(a))), \ldots\}$$

- All terms thus generated could, in principle, be used by left-$\forall$ or right-$\exists$ as long as at least one branch remains open.

Example

- Situation: two constants a and b, and a one-place function symbol f.
- The formula $(\forall x)(p\,x)$ may be "unfolded" as follows:

$$
\begin{aligned}
(\forall x)(p\,x) &\equiv (\forall x)(p\,x) \\
&\equiv (\forall x)(p\,x) \wedge pa \\
&\equiv (\forall x)(p\,x) \wedge pb \wedge pa \\
&\equiv (\forall x)(p\,x) \wedge pf(a) \wedge pb \wedge pa \\
&\equiv (\forall x)(p\,x) \wedge pf(b) \wedge pf(a) \wedge pb \wedge pa \\
&\equiv (\forall x)(p\,x) \wedge pf(f(a)) \wedge pf(b) \wedge pf(a) \wedge pb \wedge pa \\
&\equiv (\forall x)(p\,x) \wedge \ldots \wedge pf(f(a)) \wedge pf(b) \wedge pf(a) \wedge pb \wedge pa
\end{aligned}
$$

Number of Generated Formulas

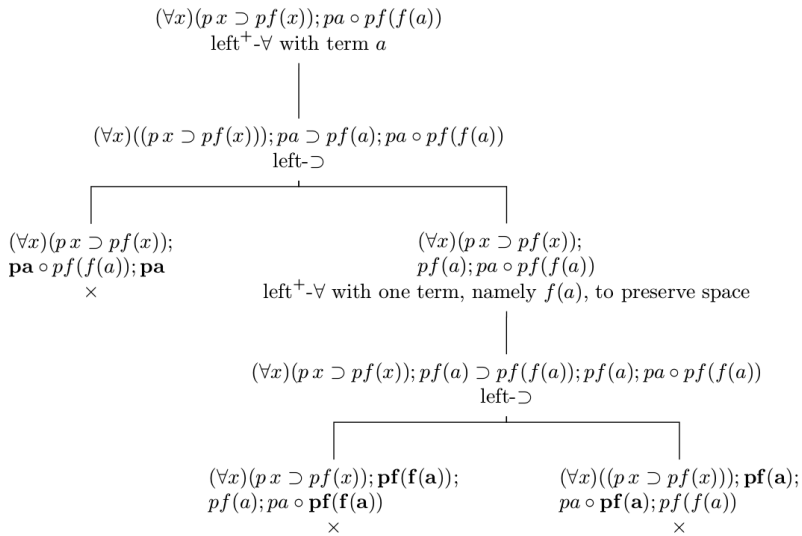
- Problem: left^+-$\forall$ or right^+-$\exists$ do not reduce the formula they operate on.
- Candidate terms in a general first-order language:

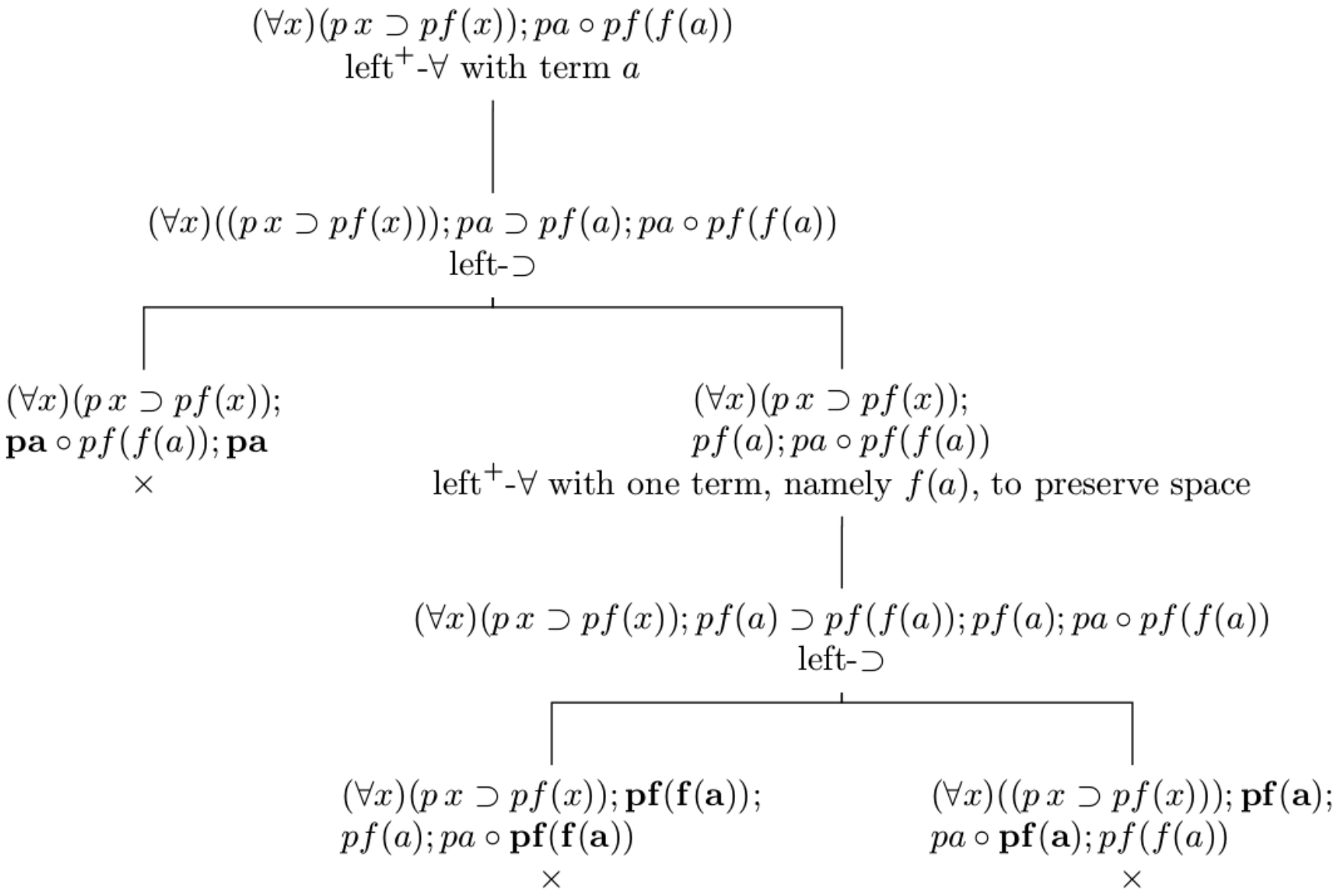
$$c_1, x_1, f_1^1(c_1), f_1^1(x_1), c_2, x_2, f_1^1(c_2), f_1^1(x_2), f_2^1(c_1), f_2^1(x_1), \ldots$$

- Countably infinite, but it will take a while to encounter the right terms to close a refutation tree (if at all possible).
- But: counterexamples need only be constructed from the Herbrand domain of the formula!

Example: $(\forall x)(p\,x \supset pf(x)); pa \vdash pf(f(a))$

- Herbrand domain (infinite): $\{a, f(a), f(f(a)), \ldots\}$
- Impossible to substitute all terms at once.
- Take care when applying left^+-$\forall$ or right^+-$\exists$




$(\forall x)(p\,x \supset pf(x)); pa \circ pf(f(a))$
left$^+$-$\forall$ with term a
$(\forall x)((p\,x \supset pf(x))); pa \supset pf(a); pa \circ pf(f(a))$
left-$\supset$
$(\forall x)(p\,x \supset pf(x)); \mathbf{pa} \circ pf(f(a)); \mathbf{pa}$
$\times$
$(\forall x)(p\,x \supset pf(x)); pf(a); pa \circ pf(f(a))$
left$^+$-$\forall$ with one term, namely $f(a)$, to preserve space
$(\forall x)(p\,x \supset pf(x)); pf(a) \supset pf(f(a)); pf(a); pa \circ pf(f(a))$
left-$\supset$
$(\forall x)(p\,x \supset pf(x)); \mathbf{pf(f(a))}; pf(a); pa \circ \mathbf{pf(f(a))}$
$\times$
$(\forall x)((p\,x \supset pf(x))); \mathbf{pf(a)}; pa \circ \mathbf{pf(a)}; pf(f(a))$
$\times$

Naive Refutation

- For us (humans) it is often obvious which terms to pick.
- A naive algorithm would proceed differently (example):

$$
\begin{array}{c}
(\forall x)(Pf(x)) \circ Pa \\
\text{left-}\forall \\
| \\
(\forall x)(Pf(x)), Pf(a) \circ Pa \\
\text{left-}\forall \\
| \\
(\forall x)(Pf(x)), Pf(a), Pf(f(a)) \circ Pa \\
\text{left-}\forall \\
| \\
\vdots \\
\text{left-}\forall \\
| \\
(\forall x)(Pf(x)), Pf(a), Pf(f(a)), \ldots, Pf^n(a) \circ Pa \\
\vdots
\end{array}
$$

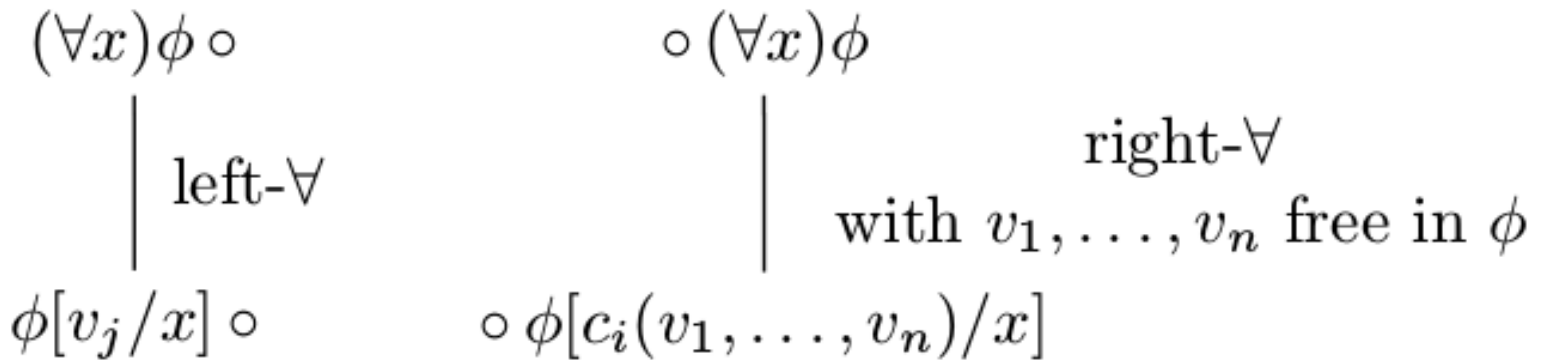
Free-Variable Substitutions

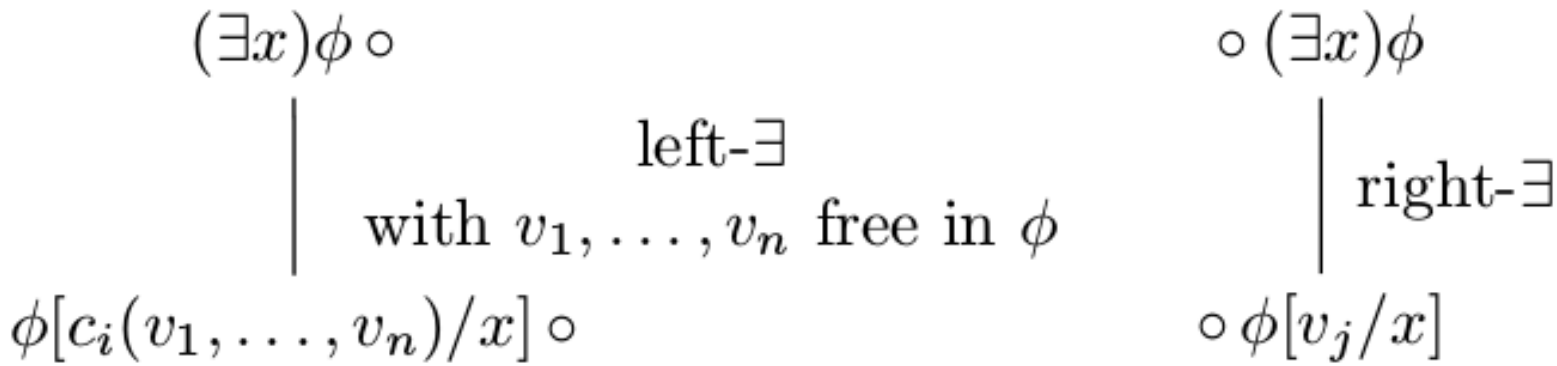
- (Human) strategy: postpone term substitutions until we see an opportunity to close a branch.
- When encountering a left-$\forall$, for instance, do not substitute a specific term, but instead mark it with a *place-holder*, i.e., a fresh variable v_i (and replace later).
- Advantage: rules out left^+-$\forall$ and right^+-$\exists$.
- Problem 1: Unclear if this works when branches split.
- Problem 2: left-$\exists$ and right-$\forall$ become problematic, since they need fresh constants.

Skolem Functions

- Solution to Problem 2: Indicate that constants on a branch with postponed substitutions $v_1, \ldots, v_n$ depend on the value of $v_1, \ldots, v_n$.
- For example, if a branch contains the constants a, d, e and pending variables $v_1, \ldots, v_n$, do not introduce a fresh constant c, but a fresh *function* ς, and indicate that ς depends on $v_1, \ldots, v_n$ by writing $\varsigma(v_1, \ldots, v_n)$.
- I.e. run proofs with fresh variables v_i and fresh function symbols ς_i.

Analytic Refutation Rules for FOL with Postponed Substitutions





Example: $(\exists w, \forall x)(Rx, w, f(x, w)) \vdash (\exists w, \forall x, \exists y)(Rx, w, y)$

$$(\exists w, \forall x)(Rx, w, f(x, w)) \circ (\exists w, \forall x, \exists y)(Rx, w, y)$$

left-$\exists$, introducing the constant ς_1 (no pending variables)

$$(\forall x)(Rx, \varsigma_1, f(x, \varsigma_1)) \circ (\exists w, \forall x, \exists y)(Rx, w, y)$$

right-$\exists$, substituting v_1 for w

$$(\forall x)(Rx, \varsigma_1, f(x, \varsigma_1)) \circ (\forall x, \exists y)(Rx, v_1, y)$$

right-$\forall$, introducing the function ς_2 with argument v_1

$$(\forall x)(Rx, \varsigma_1, f(x, \varsigma_1)) \circ (\exists y)(R\varsigma_2(v_1), v_1, y)$$

left-$\forall$, substituting v_2 for x

$$Rv_2, \varsigma_1, f(v_2, \varsigma_1) \circ (\exists y)(R\varsigma_2(v_1), v_1, y)$$

right-$\exists$, substituting v_3 for y

$$Rv_2, \varsigma_1, f(v_2, \varsigma_1) \circ R\varsigma_2(v_1), v_1, v_3$$

Example (cont'd)

$$\vdots$$
$$Rv_2, \varsigma_1, f(v_2, \varsigma_1) \circ R\varsigma_2(v_1), v_1, v_3$$

The last node of the refutation tree has the predicate R on the left and on the right, and the branch would close if these predicates were equal, i.e, if $v_2 = \varsigma_2(v_1)$, $\varsigma_1 = v_1$, and $f(v_2, \varsigma_1) = v_3$.
This can be realized with the substitution ς_1/v_1, $\varsigma_2(\varsigma_1)/v_2$, $f(\varsigma_2(\varsigma_1), \varsigma_1)/v_3$:

$$\vdots$$
$$Rv_2, \varsigma_1, f(v_2, \varsigma_1) \circ R\varsigma_2(v_1), v_1, v_3 \;/v_3$$
$$\times, \text{ with } \varsigma_1/v_1, \varsigma_2(\varsigma_1)/v_2, f(\varsigma_2(\varsigma_1)), \varsigma_1$$

Exercise

Prove the following sequents with semantic tableaux with postponed substitutions.

1. $(\forall x)(p\,x) \vdash (\forall x)(p\,x)$
2. $(\exists x)[p\,x \supset (\forall x)(p\,x)]$

Solution (1)

Two possible solutions:

$$\begin{array}{c}(\forall x)(p\,x) \circ (\forall x)(p\,x)\\ \text{right-}\forall\ \text{(no free variables)}\\ \Big|\\ (\forall x)(p\,x) \circ p\varsigma_1\\ \text{left-}\forall\ \text{(introducing new variable } v_1)\\ \Big|\\ \phi, pv_1 \circ p\varsigma_1\\ \times \text{ with } \sigma = \{\varsigma_1/v_1\}\end{array}$$

and

$$\begin{array}{c}(\forall x)(p\,x) \circ (\forall x)(p\,x)\\ \text{left-}\forall\ \text{(introducing new variable } v_1)\\ \Big|\\ \phi, pv_1 \circ (\forall x)(p\,x)\\ \text{right-}\forall\ \text{(no free variables in the RHS)}\\ \Big|\\ \phi, pv_1 \circ p\varsigma_1\\ \times \text{ with } \sigma = \{\varsigma_1/v_1\}\end{array}$$

in both refutations, $\phi = (\forall x)(p\,x)$.

Solution (2)

$$\circ\ (\exists x)[p\,x \supset (\forall x)(p\,x)]$$
right-$\exists$ (introducing new variable v_1)
$$\Big|$$
$$\circ\ \phi, pv_1 \supset (\forall x)(p\,x)$$
right-$\supset$
$$\Big|$$
$$pv_1 \circ \phi, (\forall x)(p\,x)$$
right-$\forall$ (no free variables in the RHS)
$$\Big|$$
$$pv_1 \circ \phi, p\varsigma_1$$
$\times$ with $\sigma = \{\varsigma_1/v_1\}$

where $\phi = (\exists x)[p\,x \supset (\forall x)(p\,x)]$.

Unification

- With free-variable substitutions, we need an algorithm that computes for us if two terms can be made equal, and if so, how.
- The process of trying to make two terms equal is called *unification*.
- Unification is an important process in ATP because it also plays an important role in resolution.

Definition (Unification)

A unification of terms t_1 and t_2 is a substitution σ that makes $t_1\sigma$ and $t_2\sigma$ syntactically equal.

Difference between Terms

The *differences* between two terms t_1 and t_2 can be expressed by a set

$$D = \{(s_1^1, s_2^1), \ldots, (s_1^n, s_2^n)\}$$

of pairs of terms, such that

- $s_1^i \neq s_2^i$
- Every s_1^i is a sub-term of t_1 and every s_2^i is a sub-term of t_2.
- The terms t_1' and t_2' that are created when all s_j^i's are replaced by a similar token are syntactically equal.
- All the s_j^i's are as large as possible.

Unification Algorithm

Input: two terms t_1, t_2

1: - let $\sigma = \varnothing$
2: **while** $t_1\sigma \neq t_2\sigma$ **do**
3: - choose pair of terms (s_1, s_2) that are different in $(t_1\sigma, t_2\sigma)$
4: - **return undef if** neither s_1 nor s_2 are variables
5: - swap s_1 and s_2 **if** s_1 is not a variable
6: - **return undef if** variable s_1 occurs in s_2
7: - $\sigma = \sigma \cup \{s_2/s_1\}$
8: - **return** σ

Unification Example

Let

$$t_1 = g(f(h(b), y), h(c)),$$
$$t_2 = g(f(z, d), h(h(x)))$$

The differences between t_1 and t_2 are

$$s_1^1, s_2^1 = h(b), z$$
$$s_1^2, s_2^2 = y, d$$
$$s_1^3, s_2^3 = c, h(x)$$

If we would like to unify t_1 and t_2, then $h(b)$ should be equal to z, y should be equal to d, and c should be equal to $h(x)$.
The first two requirements can be fulfilled by the (partial) substitution $h(b)/z$ and d/y.
The third requirement cannot be fulfilled, however,because s_1^3 is a constant while s_2^3 starts with a function symbol.

Exercise

Explain why the unification algorithm always halts. (Hint: consider the total number of variables in t_1 and t_2.)

Solution

- The total number of variables is finite.
- The algorithm chooses a pair of terms in each iteration.
- It will either halt the procedure (because not reasonable substitution is possible), or substitute (eliminate) one of the variables to make the terms equal, possibly up until the point where the input terms are unified.
- Thus, if it never halts within the iteration, the loop condition will eventually be false and the algorithm will thus terminate.

Functions and Equality

- Almost all realistic problems involve equalities.
- Unfortunately, equality makes things rather complicated...
- Additional rules for dealing with equality:

$$\dfrac{}{s = s \circ}\ \text{left-=}$$

$$\dfrac{}{f(t_1, \ldots, t_n) = f(t_1, \ldots, t_n) \circ}\ \text{left-=, for functions}$$

$$\dfrac{s = t, P(\ldots, s, \ldots) \circ}{s = t, P(\ldots, s, \ldots), P(\ldots, t, \ldots) \circ}\ \text{left replacement}$$

$$\dfrac{s = t \circ P(\ldots, s, \ldots)}{s = t \circ P(\ldots, s, \ldots), P(\ldots, t, \ldots)}\ \text{right replacement}$$

Functions and Equality

$$\dfrac{}{s = s \circ}\ \text{left-=}$$

$$\dfrac{}{f(t_1, \ldots, t_n) = f(t_1, \ldots, t_n) \circ}\ \text{left-=, for functions}$$

$$\dfrac{s = t, P(\ldots, s, \ldots) \circ}{s = t, P(\ldots, s, \ldots), P(\ldots, t, \ldots) \circ}\ \text{left replacement}$$

$$\dfrac{s = t \circ P(\ldots, s, \ldots)}{s = t \circ P(\ldots, s, \ldots), P(\ldots, t, \ldots)}\ \text{right replacement}$$

- Important: The left and right replacement rules are directional, permit only the left-to-right use of equalities.
- Further, the predicate P can be the equality predicate itself.
- Sound and complete (proof out of scope).

Example (proof of transitivity of $=$)

$$\circ(\forall x)((\forall y)((\forall z)(x = y \wedge y = z \supset x = z)))$$
right-$\forall$
$$\circ(\forall y)((\forall z)(\varsigma_1 = y \wedge y = z \supset \varsigma_1 = z))$$
right-$\forall$
$$\circ(\forall z)(\varsigma_1 = \varsigma_2 \wedge \varsigma_2 = z \supset \varsigma_1 = z)$$
right-$\forall$
$$\circ\varsigma_1 = \varsigma_2 \wedge \varsigma_2 = \varsigma_3 \supset \varsigma_1 = \varsigma_3$$
right-$\supset$
$$\varsigma_1 = \varsigma_2 \wedge \varsigma_2 = \varsigma_3 \circ \varsigma_1 = \varsigma_3$$
left-$\wedge$
$$\varsigma_1 = \varsigma_2, \varsigma_2 = \varsigma_3 \circ \varsigma_1 = \varsigma_3$$
using $\varsigma_1 = \varsigma_2$ on the left to rewrite $\varsigma_1 = \varsigma_3$ on the right into $\varsigma_2 = \varsigma_3$
$$\varsigma_1 = \varsigma_2, \varsigma_2 = \varsigma_3 \circ \varsigma_1 = \varsigma_3, \varsigma_2 = \varsigma_3$$
$\times$

Exercise

Prove the following formulas by refutation. (Function and predicate substitution are not needed.)

1. Symmetry: $(\forall x)((\forall y)(x = y \supset y = x))$
2. Existential reflexivity: $(\forall x)((\exists y)(x = y))$

Solution (1)

$\circ\ (\forall x)((\forall y)(x = y\ \supset y = x))$

(replace quantified x by a new constant, ς_1)

|

$\circ\ (\forall y)(\varsigma_1 = y\ \supset y = \varsigma_1)$

(replace quantified y by a new constant, ς_2)

|

$\circ\ \varsigma_1 = \varsigma_2\ \supset \varsigma_2 = \varsigma_1$

(right-$\supset$)

|

$\varsigma_1 = \varsigma_2\ \circ \varsigma_2 = \varsigma_1$

(right replacement of ς_1 with $\varsigma_1 = \varsigma_2$)

|

$\varsigma_1 = \varsigma_2\ \circ \varsigma_2 = \varsigma_2$

(left replacement of ς_1 with $\varsigma_1 = \varsigma_2$)

|

$\varsigma_2 = \varsigma_2\ \circ \varsigma_2 = \varsigma_2$

$\times$

Solution (2)

$\circ\ (\forall x)((\exists y)(x = y))$

(replace quantified x by a new constant, ς_1)

|

$\circ\ (\exists y)(\varsigma_1 = y)$

(replace quantified y by the same constant, ς_1.

[We work with conventional tableaus.])

|

$\circ\ \varsigma_1 = \varsigma_1$

(left-$=$ with ς_1)

|

$\varsigma_1 = \varsigma_1 \circ \varsigma_1 = \varsigma_1$

$\times$

Heuristics

- Moving from propositional to first order logic: gain of expressive power at the price of proof complexity.
- Equalities make it even worse.
- Heuristics needed to prioritize the schedule of logic operations.
- "Guidelines" for the ATP algorithm to decide which branches to explore first, which sequents to analyze first, and which term-substitutions to make first.

In This Course

- Propositional theorem proving (last Monday),
  Chapter 2 of the lecture notes
- First-order theorem proving (today),
  Chapter 3 of the lecture notes
- Clause sets and resolution (next Monday),
  Chapters 4 and 5 of the lecture notes
- Satisfiability checkers, SAT/SMT (next Wednesday),
  Chapter 6 of the lecture notes, additional material

Homework

The following homework exercises are useful to review today's content in preparation for the next lecture:

- Sec. 3.6 Problem 2 (c)-(d) (page 72)
- Sec. 3.8 Problems 2–3 (pages 75/76)