universität innsbruck

Interactive Theorem Proving and *Automation*

Lecture & Exercises Week 12

Jan Jakubův (and Cezary Kaliszyk)

Summary

Previous Lecture

- Higher-order logics and ITPs

Today

- Automated reasoning in First-order logic
- TPTP World: Language and problem library
- Clausification and Unification
- Resolution Calculus
- Applications in proof assistants

Language of First-order Logic

- **Variables and signature:**

$$x ::= x \mid y \mid z \mid \ \dots \qquad \text{Variables (typically countable)}$$

$$f ::= f \mid g \mid h \mid \ \dots \qquad \text{Function symbols (typically finite)}$$

$$P ::= P \mid Q \mid R \mid \ \dots \qquad \text{Predicate symbols (typically finite)}$$

- Each symbol from f, P has a fixed arity: $f/2$ (binary), $P/3$ (ternary), ...
- **Syntax of terms and formulae:**

$$t ::= x \mid f(t_1, \dots, t_n) \qquad \text{Terms (type } \iota\text{)}$$

$$\alpha ::= P(t_1, \dots, t_n) \qquad \text{Atoms (type } o\text{)}$$

$$A, B, C ::= \alpha \mid \neg(A) \mid (A) \to (B) \mid \forall x\,(A) \qquad \text{Formulae (type } o\text{)}$$

- Convention: Drop unnecessary parenthesis, e.g., $\neg\neg A$ instead of $\neg(\neg(A))$.

First-order Logic Abbreviations

- **Abbreviations** introduce other symbols:

$$\begin{aligned} A \vee B &\equiv \neg A \rightarrow B \\ A \wedge B &\equiv \neg(A \rightarrow \neg B) \\ A \Leftrightarrow B &\equiv (A \rightarrow B) \wedge (B \rightarrow A) \\ \exists x\,(A) &\equiv \neg \forall x\,(\neg A) \end{aligned}$$

- **Propositional logic** is a special case:
  - without function symbols and variables
  - with only nulary predicate symbols P/0 (*propositional constants*)
- **Alternative** complete set of base connectives instead of $\{\neg, \rightarrow\}$:
  - $\{\wedge, \vee, \neg\}$
  - $\{\uparrow\}$ where $A \uparrow B \equiv \neg(A \Leftrightarrow B)$ (Sheffer stroke, NAND)

Axiomatization of First-order Logic

- **Axiom schemes:**

$A \to (B \to A)$	(A_1)	
$(A \to (B \to C)) \to ((A \to B) \to (A \to C))$	(A_2)	
$\neg\neg A \to A$	(A_3)	
$\forall x(A[x]) \to A[t]$	(A_4)	if substitutable
$\forall x(A \to B) \to (A \to \forall x B)$	(A_5)	if $x \notin \mathrm{vars}(A)$

substitutable: No $z \in \mathrm{vars}(t)$ is $\exists$-bound in A (simplification).

Axiomatization of First-order Logic

- **Axiom schemes:**

$$A \to (B \to A) \qquad (A_1)$$

$$(A \to (B \to C)) \to ((A \to B) \to (A \to C)) \qquad (A_2)$$

$$\neg\neg A \to A \qquad (A_3)$$

$$\forall x(A[x]) \to A[t] \qquad (A_4) \quad \text{if substitutable}$$

$$\forall x(A \to B) \to (A \to \forall x B) \qquad (A_5) \quad \text{if } x \notin \mathrm{vars}(A)$$

- Every instance of $A_1, \ldots, A_5$ is an axiom and is valid.

Axiomatization of First-order Logic

- **Axiom schemes:**

$$
\begin{array}{lll}
A \to (B \to A) & (A_1) & \\
(A \to (B \to C)) \to ((A \to B) \to (A \to C)) & (A_2) & \\
\neg\neg A \to A & (A_3) & \\
\forall x(A[x]) \to A[t] & (A_4) & \text{if substitutable} \\
\forall x(A \to B) \to (A \to \forall x B) & (A_5) & \text{if } x \notin \text{vars}(A)
\end{array}
$$

- Every instance of $A_1, \ldots, A_5$ is an axiom and is valid.
- **Inference rules:**

$$
\frac{A \qquad A \to B}{B} \quad \text{(Modus Ponens)} \qquad\qquad \frac{A}{\forall x(A)} \quad \text{(Generalization)}
$$

- Infer a valid formula from valid formulae.

Proofs in First-order Logic

- **Proof of A** in FOL is a sequence of formulae ending with A, where every formula is either
  - an axiom, or
  - is derived from formula(s) coming before in the proof.
- **A is probable** (written $\vdash A$) if there exists some proof of A.
- Axiom schemes $A_1, \ldots, A_3$ with MP, is a
  - **correct**: only tautologies can be proved, and
  - **complete**: all tautologies can be proved,

  axiomatization of **propositional logic**.
- Schemes $A_1, \ldots, A_5$ with MP and Gen, is a **correct and complete** axiomatization of **First-order logic**: only logically valid formulae are proved.

Exercise: Propositional Logic

- **Axiom schemes:**

$$A \to (B \to A) \qquad (A_1)$$

$$(A \to (B \to C)) \to ((A \to B) \to (A \to C)) \qquad (A_2)$$

$$\neg\neg A \to A \qquad (A_3)$$

- **Inference rules:**

$$\frac{A \qquad A \to B}{B} \quad \text{(Modus Ponens)}$$

- **Exercise:** Prove $\vdash A \to A$ using $A_1, \ldots, A_3$ with MP.
  Proof can be represented by a tree/dag (the derivation of $A \to A$).

Exercise: Solution

Claim: $\vdash A \rightarrow A$

Proof:

(1)	$A \rightarrow ((B \rightarrow A) \rightarrow A)$	(instance of A_1)
(2)	$(A \rightarrow ((B \rightarrow A) \rightarrow A)) \rightarrow ((A \rightarrow (B \rightarrow A)) \rightarrow (A \rightarrow A))$	(instance of A_2)
(3)	$(A \rightarrow (B \rightarrow A)) \rightarrow (A \rightarrow A)$	(from (1) and (2))
(4)	$A \rightarrow (B \rightarrow A)$	(instance of A_1)
(6)	$A \rightarrow A$	(from (4) and (3))

Theories in First-order Logic

- **Theory T** is an additional (countable) set of axioms.
- **A is provable in T**, written $T \vdash A$, when there exists a proof of A ($\vdash \left(\wedge T\right) \rightarrow A$).
- **Equality axioms** can be added:

$$\begin{aligned} & t = t && \text{(reflexivity)} \\ & t = s \rightarrow s = t && \text{(symmetry)} \\ & (t = s) \wedge (s = r) \rightarrow (t = r) && \text{(transitivity)} \end{aligned}$$

- with **congruence** axioms:

$$\begin{aligned} & t = s \rightarrow f(t) = f(s) && \text{for every function } f \\ & (t = s \wedge P(t)) \rightarrow P(s) && \text{for every predicate } P \end{aligned}$$

- for every term t, s, r.
- No new inference rule is necessary (but can be added).

TPTP World of Automated Theorem Provers

Thousands of Problems for Theorem Provers (TPTP)

- Library of first-order problems from various fields.
- Language to represent logic formulae as text for computers.
- Online interface to run many ATP provers (SystemOnTPTP).

TPTP syntax for terms (ASCII):

object	*syntax*	*comment*
variables	`X`	capital letter first
other symbols	`f`	lower case first
application	`f(X,a)`	prefix notation, comma-separated

TPTP Language

- **Connectives**:

FOL symbol	$\wedge$	$\vee$	$\rightarrow$	$\neg$	$\equiv$
TPTP syntax	`&`	`|`	`=>`	`~`	`<=>`

- **Formula:**

composed	`p(a) & p(b)`	infix syntax for connectives
forall	`![X]:( p(X) )`	don't forget parenthesis
exists	`?[X]:( p(X) )`	here as well

- **TPTP file** is a sequence of TPTP statements:

```
fof(name, role, (formula)).     # this is a comment
```

where `name` is a user-defined text, and `role` is either `axiom` or `conjecture`.

TPTP Language

- **Connectives**:

FOL symbol	$\wedge$	$\vee$	$\rightarrow$	$\neg$	$\equiv$
TPTP syntax	`&`	`|`	`=>`	`~`	`<=>`

- **Formula:**

composed	`p(a) & p(b)`	infix syntax for connectives
forall	`![X]:( p(X) )`	don't forget parenthesis
exists	`?[X]:( p(X) )`	here as well

- **TPTP file** is a sequence of TPTP statements:

```
fof(name, role, (formula)).     # this is a comment
```

where `name` is a user-defined text, and `role` is either `axiom` or `conjecture`.

- **Exercise:** Go to `tptp.org` and locate and investigate problem `PUZ001+1`.

System on TPTP

- SystemOnTPTP provides a web interface to experiment with provers.
- **Exercise 1:** Use SystemOnTPTP to prove problem `PUZ001+1` by E or Vampire.
- Hint: Search for the text “`SZS status`” in the output.
- **Exercise 2:** Prove or disprove the *Drinker’s paradox* using E:

$$\exists x\,(P(x) \rightarrow \forall y\, P(y))$$

- **Exercise 3:** Compare with the results for:

$$\exists x\,(P(x)) \rightarrow \forall y\, P(y)$$

Core of Automated Theorem Proving (ATP)

Clauses

- Use simpler clauses instead of general formulae.
- Clause is a disjuction of literals (atom α or $\neg\alpha$), e.g., $P(x) \vee \neg Q(x) \vee R(x, f(y))$
- No quantifiers in clauses.
- All (free) variables are implicitelly $\forall$-qualified.
- Every formula can be translated to a logically equivalent set of clauses.

Proof by contradiction

- To prove $T \vdash A$, show that $T \cup \{\neg A\}$ is contradictory (unsatisfiable).
- Proof is a sequence deriving the empty clause ($\Box$).
- We show: $T \cup \{\neg A\} \vdash \Box$
- The empty clause represents the contradiction.

Clausal Normal Form

Clausification

Translation of first-order formula A to a set of clauses $\{C_1, \ldots, C_n\}$ such that

$$A \qquad \text{and} \qquad \forall x_1\,(C_1) \wedge \cdots \wedge \forall x_n\,(C_n)$$

are equisatisfiable, where x_i stands for all (free) variables in C_i.

Consists of

- **Skolemization** to eliminate existential quantifiers ($\exists$).
- **CNF transformation** to construct a conjunction of disjunctions (of literals).

Clausification

Skolemization

1. Eliminate all but $\{\wedge, \vee, \neg\}$. $A \to B \equiv \neg A \vee B$ $A \Leftrightarrow B \equiv (A \wedge B) \vee (\neg A \vee \neg B)$

Clausification

Skolemization

1. Eliminate all but $\{\wedge, \vee, \neg\}$. $A \rightarrow B \equiv \neg A \vee B$ $A \Leftrightarrow B \equiv (A \wedge B) \vee (\neg A \vee \neg B)$
2. Translate to the prenex form with all quantifiers $(\forall, \exists)$ at the top.

$$\neg\forall\, x(A) \equiv \exists x\,(\neg A) \qquad A \wedge \forall x\,(B) \equiv \forall x\,(A \wedge B) \quad (\text{if } x \notin \text{vars}(A))$$

Clausification

Skolemization

1. Eliminate all but $\{\wedge, \vee, \neg\}$. $A \rightarrow B \equiv \neg A \vee B$ $A \Leftrightarrow B \equiv (A \wedge B) \vee (\neg A \vee \neg B)$
2. Translate to the prenex form with all quantifiers ($\forall, \exists$) at the top.

$$\neg \forall x(A) \equiv \exists x\,(\neg A) \qquad A \wedge \forall x\,(B) \equiv \forall x\,(A \wedge B) \quad (\text{if } x \notin \text{vars}(A))$$

3. Translate $\exists x(A)$ to $A[x \mapsto c]$ where c is new Skolem constant (witness).

Clausification

Skolemization

1. Eliminate all but $\{\wedge, \vee, \neg\}$. $A \rightarrow B \equiv \neg A \vee B$ $A \Leftrightarrow B \equiv (A \wedge B) \vee (\neg A \vee \neg B)$
2. Translate to the prenex form with all quantifiers $(\forall, \exists)$ at the top.

$$\neg\forall x(A) \equiv \exists x\,(\neg A) \qquad A \wedge \forall x\,(B) \equiv \forall x\,(A \wedge B) \quad (\text{if } x \notin \text{vars}(A))$$

3. Translate $\exists x(A)$ to $A[x \mapsto c]$ where c is new Skolem constant (witness).
4. Translate $\forall y \exists x(A)$ to $\forall y\,(A[x \mapsto f(y)])$ where f is new Skolem function.

Clausification

Skolemization

1. Eliminate all but $\{\wedge, \vee, \neg\}$. $A \rightarrow B \equiv \neg A \vee B$ $A \Leftrightarrow B \equiv (A \wedge B) \vee (\neg A \vee \neg B)$
2. Translate to the prenex form with all quantifiers $(\forall, \exists)$ at the top.

$$\neg\forall x(A) \equiv \exists x\,(\neg A) \qquad A \wedge \forall x\,(B) \equiv \forall x\,(A \wedge B) \quad (\text{if } x \notin \mathrm{vars}(A))$$

3. Translate $\exists x(A)$ to $A[x \mapsto c]$ where c is new Skolem constant (witness).
4. Translate $\forall y \exists x(A)$ to $\forall y\,(A[x \mapsto f(y)])$ where f is new Skolem function.

CNF transformation is done using

- **de Morgan laws** $\neg(A \wedge B) \equiv (\neg A) \vee (\neg B)$ $\neg(A \vee B) \equiv (\neg A) \wedge (\neg B)$
- **distributivity** $A \wedge (B \vee C) \equiv (A \wedge B) \vee (A \wedge C)$ (etc.)
- **double negation** elimination

Exercises

- Translate the following to the clausal normal form.
- **Exercise 1:** $\exists x\, (P(x) \rightarrow \forall y\, P(y))$
- **Exercise 2:** $\exists x\, (P(x)) \rightarrow \forall y\, P(y)$
- **Exercise 3:** $\forall x\, (P(x)) \rightarrow \exists y\, P(y)$
- By hand or with the help of SystemOnTptp.

Unification in First-order Logic

Most ATPs rely on unification.

Unificator σ of terms t and s

- Is a substitution (a mapping from variables to terms) such that $\sigma(t) \equiv \sigma(s)$.
- Typically written postfix as $t\sigma$.
- Substitution can be applied to formulas: $A\sigma$ (modify free variables only!)

Most general unificator

- By example: Both $\sigma_1 = \{x \mapsto y\}$ and $\sigma_2 = \{x \mapsto a, y \mapsto a\}$ unify $f(x, y)$ and $f(y, x)$.
- The first is more general w.r.t. composition: σ_2 is σ_1 composed with $\{y \mapsto a\}$.
- But σ_1 can not be written as composition of σ_2 with something.
- All unifiable terms have a most general unifier (mgu).

Martelli-Montanari Unification Algorithm

- Work with set of equations of the shape $t = s$ for terms t, s.
- To unify t and s start with a singleton set $\{t = s\}$.

Keep applying the following rules (nondeterministically)

- **Delete** all equations of shape $t = t$
- **Eliminate** equations of shape $x = t$ if $x \notin \text{vars}(t)$:
  Apply $\{x \mapsto t\}$ to all other equations (and remember the binding).
- **Decompose** equation $f(t_1, \ldots, t_n) = f(s_1, \ldots, s_n)$ into $t_1 = s_1, \ldots, t_n = s_n$.

- **Terminate** with success if empty set reached, fail otherwise.
- The algorithm returns the mgu of s and t if exists.

Martelli-Montanari Unification Algorithm

- Work with set of equations of the shape $t = s$ for terms t, s.
- To unify t and s start with a singleton set $\{t = s\}$.

Keep applying the following rules (nondeterministically)

- **Delete** all equations of shape $t = t$
- **Eliminate** equations of shape $x = t$ if $x \notin \mathrm{vars}(t)$:
  Apply $\{x \mapsto t\}$ to all other equations (and remember the binding).
- **Decompose** equation $f(t_1, \ldots, t_n) = f(s_1, \ldots, s_n)$ into $t_1 = s_1$,...,$t_n = s_n$.

- **Terminate** with success if empty set reached, fail otherwise.
- The algorithm returns the mgu of s and t if exists.
- Exercise: Find the mgu of $Q(a, g(x, a), f(y))$ and $Q(a, g(f(b), a), x)$.

Resolution Calculus: Inference Rules

Binary resolution

$$\frac{L_1 \vee \mathcal{C} \qquad \neg L_2 \vee \mathcal{D}}{(\mathcal{C} \vee \mathcal{D})\sigma} \quad \sigma = \mathrm{mgu}(L_1, L_2)$$

- $\mathcal{C}, \mathcal{D}$ are disjunctions of literals, and premises do not share variables.

Factorization

$$\frac{L_1 \vee L_2 \vee \mathcal{C}}{(L_1 \vee \mathcal{C})\sigma} \quad \sigma = \mathrm{mgu}(L_1, L_2)$$

- Resolution with factorization are **refutationally complete**:

  If $\quad T \vdash \Box$ (in FOL) $\quad$ then $\quad \Box$ can be derived by resolution from axioms T.

Resolution Calculus: Inference Rules

Binary resolution

$$\frac{L_1 \vee \mathcal{C} \qquad \neg L_2 \vee \mathcal{D}}{(\mathcal{C} \vee \mathcal{D})\sigma} \quad \sigma = \mathrm{mgu}(L_1, L_2)$$

- $\mathcal{C}, \mathcal{D}$ are disjunctions of literals, and premises do not share variables.

Factorization

$$\frac{L_1 \vee L_2 \vee \mathcal{C}}{(L_1 \vee \mathcal{C})\sigma} \quad \sigma = \mathrm{mgu}(L_1, L_2)$$

- Resolution with factorization are **refutationally complete**:

  If $T \vdash \Box$ (in FOL) then $\Box$ can be derived by resolution from axioms T.

- Exercise 1: Prove $\vdash A \to A$ by resolution.

Resolution Calculus: Inference Rules

Binary resolution

$$\frac{L_1 \vee \mathcal{C} \qquad \neg L_2 \vee \mathcal{D}}{(\mathcal{C} \vee \mathcal{D})\sigma} \quad \sigma = \mathrm{mgu}(L_1, L_2)$$

- $\mathcal{C}, \mathcal{D}$ are disjunctions of literals, and premises do not share variables.

Factorization

$$\frac{L_1 \vee L_2 \vee \mathcal{C}}{(L_1 \vee \mathcal{C})\sigma} \quad \sigma = \mathrm{mgu}(L_1, L_2)$$

- Resolution with factorization are **refutationally complete**:

  If $\quad T \vdash \Box$ (in FOL) $\quad$ then $\quad \Box$ can be derived by resolution from axioms T.

- Exercise 2: Prove $\vdash \exists x\,(P(x) \rightarrow \forall y\, P(y))$.

Application: ATPs for ITPs

ATPs are used by ITP Hammers

- Translate ITP problem to FOL.
- Select appropriate definition and lemmas as axioms.
- Call the ATP prover(s).
- Translate ATP proof back to ITP.