Automated Reasoning

Natural Deduction in First-Order Logic

Jacques Fleuriot

Problem

Consider the following problem:

Every person has a heart.

George Bush is a person.

Does George Bush have a heart?

Is Propositional logic rich enough to formally represent and reason about this problem?

The finer logical structure of this problem would not be captured by the constructs we have so far encountered.

We need a richer language!

A Richer Language

First order logic (FOL) extends propositional logic:

- Reasons about “individuals in a universe of discourse” and their “properties”
- Have predicates and functions to denote properties
- A variable stands for an element of the universe
- Variables range over individuals but not over functions and predicates
- Propositional connectives used to build up statements
- Quantifiers $\forall$ (for all) and $\exists$ (there exists) used
- FOL also known as Predicate logic

FOL

- First order language is characterized by giving a finite collection of functions $\mathcal{F}$ and predicates $\mathcal{P}$ as well as a set of variables.
  - Often call ($\mathcal{F}$, $\mathcal{P}$) *a signature*
- *2 syntactic categories: terms and formulae*
  - *terms stand for individuals while formulae stand for truth values*

Terms of FOL

Terms of a first-order language are defined as:

- Any variable is a term
- If $c \in \mathcal{F}$ *is a nullary function (i.e. a constant), then* c *is a term*
- If $t_1, \ldots, t_n$ are terms and function $f \in \mathcal{F}$ *has arity* n > 0, then $f(t_1, \ldots, t_n)$ is a term
- Nothing else is a term

Formulae of FOL

A well-formed formula in FOL is defined as:

- If $P \in \mathcal{P}$ is a predicate symbol of arity $n \geqslant 0$, and if $t_1, \ldots, t_n$ are terms over $\mathcal{F}$, then $P(t_1, \ldots, t_n)$ is a formula.
- If ϕ is a formula, then so is $(\neg\phi)$.
- If ϕ and ψ are formulas, then so are $(\phi\wedge\psi)$, $(\phi\vee\psi)$, $(\phi\rightarrow\psi)$ and $(\phi\leftrightarrow\psi)$.
- If ϕ is a formula and x is a variable, then $(\exists x.\ \phi)$ and $(\forall x.\ \phi)$ are formulas.
- Nothing else is a formula.

Example: Problem Revisited

We can now formally represent our problem in FOL:

Every person has a heart: $\forall x.\ person(x) \rightarrow hasHeart(x)$

George Bush is a person: $person(bush)$

To answer the question

Does George Bush have a heart?

we need to prove:

$$((\forall x.\ person(x) \rightarrow hasHeart(x)) \wedge person(bush)) \rightarrow hasHeart(bush)$$

How do we prove if this is a valid statement?

- more on this later

Variables

- In FOL, variables can be in one of two states:
  - bound: $\forall x.\ x=x$ or $\exists x.\ x=x$, *etc* ...
  - free: $x=x$

- For example, in the proposition:

  $\forall x.\ \exists y.\ x * y = z$

  x and y are bound variables and z is a free variable.

Substitution Rule

If P is a formula, s is a term, and x is a free variable, then

$$P[s/x]$$

is the formula obtained by substituting s for x throughout P. Such a substitution rule can be defined as:

$$\frac{s=t \quad P[s/x]}{P[t/x]}\ subst$$

Example: $\exists x.\ P(x,y)\ [3/y] = \exists x.\ P(x,3)$

$\exists x.\ P(x,y)\ [2/x] = \exists x.\ P(x,y)$

Semantics of FOL Formulae

Informal view:

An interpretation of a formula maps its function symbols, including constants, to actual functions, and its predicate symbols to actual relations.

The interpretation also specifies some domain $\mathcal{D}$ (a non-empty set or universe) on which the functions and relations are defined.

Definition of Interpretation

An interpretation for a wff consists of a nonempty set $\mathcal{D}$, called the domain of the interpretation, together with an assignment of meanings to the symbols of the wff.

1. Each predicate symbol is assigned to a relation over $\mathcal{D}$.

   A nullary predicate is assigned a truth value.

2. Each function symbol is assigned to a function over $\mathcal{D}$.

   Each nullary function (constant) is assigned to a value in $\mathcal{D}$.

3. Each free variable is assigned to a value in $\mathcal{D}$.

   All free occurrences of a free variable x are assigned to the same value in $\mathcal{D}$.

Example of Interpretation

Consider the formula

$$P(a) \land \exists x.\ Q(a,x) \quad (*)$$

formula does not mean anything on its own

A possible interpretation is:

- Domain is the set of natural numbers (*e.g.* 0, 1, 2, 3 ,…)
- Assign 2 to a, assign the property of being even to P, and the relation of being greater than to Q, i.e $Q(x,y)$ means x is greater than y
- Under this interpretation: (*) affirms that 2 is even and there exists a natural number that 2 is greater than. Is (*) satisfied under this interpretation? -Yes
- Such a satisfying interpretation is known as a model

Semantics of FOL Formulae

The semantics (meaning) of a **wff** in FOL with respect to an interpretation with domain $\mathcal{D}$ is the truth value obtained by applying the following rules:

1. If the **wff** has no quantifiers then its meaning is the truth value of the proposition obtained by applying the interpretation to the **wff**.

2. If the **wff** contains $\forall x.\ W$ then $\forall x.$ W is true if W $[d/x]$ is true for every $d \in \mathcal{D}$. Otherwise, $\forall x.$ W is false.

3. If the **wff** contains $\exists x.$ W then $\exists x.$ W is true if W $[d/x]$ is true for some $d \in \mathcal{D}$. Otherwise, $\exists x.$ W is false.

More Introduction Rules

Our natural deduction rules for Propositional logic need to be extended to deal with FOL.

Quantifiers $\forall$, $\exists$ need substitution and notion of arbitrary variable:

x_0 is an arbitrary free variable i.e. we make no assumptions about it

$$\frac{P\ x_0}{\forall x. P\ x}\ allI$$

provided x_0 is fresh

$$\frac{P\ a}{\exists x. P\ x}\ exI$$

Existential Elimination

The proviso is part of the rule definition and cannot be omitted

$$\frac{\exists u. P\ u \qquad \begin{matrix}[P\ x]\\ \vdots \\ Q\end{matrix}}{Q}\ exE$$

Provided x does not occur in $P\ u$ or Q or any other premise other than $P\ x$ on which derivation of Q from $P\ x$ depends

Universal Elimination

"*specialization*" rule

$$\frac{\forall u.\, P\ u}{P\ \ x}\ spec$$

An alternative universal elimination rule is `allE`:

$$\frac{\forall u.\, P\ u \qquad \begin{array}{c}[P\ x]\\ \vdots \\ R\end{array}}{R}\ allE$$

Note: This rule is mostly useful when doing a mechanical proof

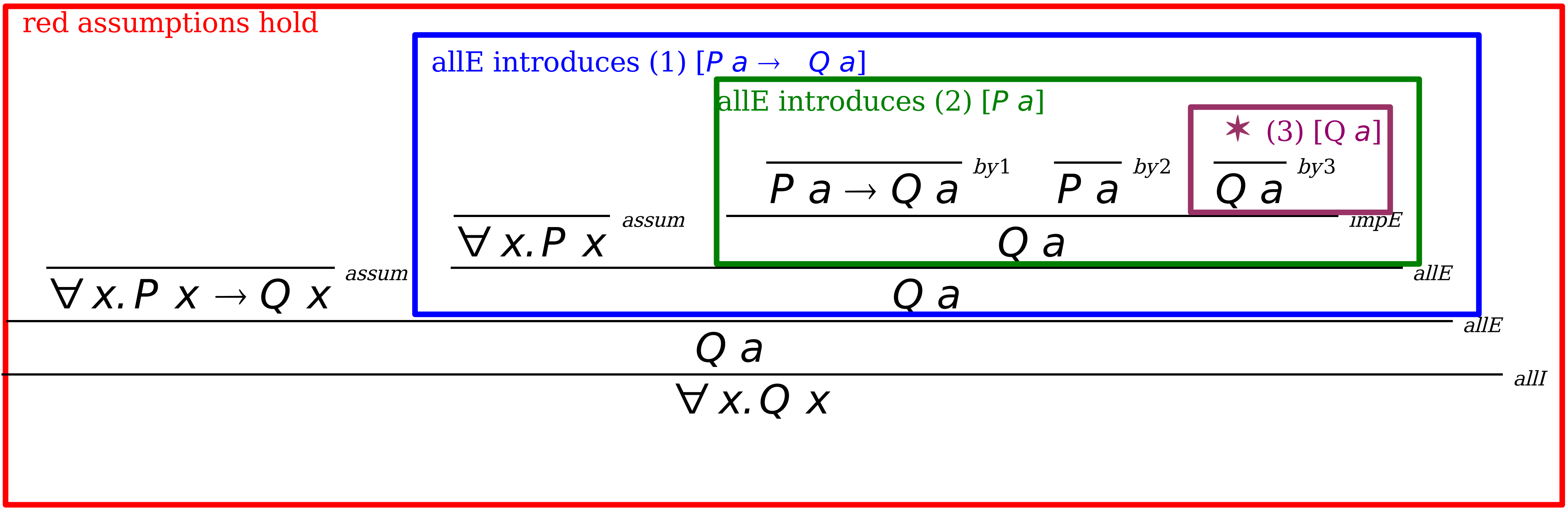
Example proof

Prove that $\exists y.\ P\ y$ is true, given that $\forall x.\ P\ x$ holds.

$$\cfrac{\cfrac{\overline{\forall x. P\ x}\ \textit{assum}}{P\ a}\ \textit{spec}}{\exists y. P\ y}\ \textit{exI}$$

Example proof (II)

Prove that $\forall x.\ Q\ x$ is true, given that $\forall x.\ P\ x$ and $(\forall x.\ P\ x \to Q\ x)$ both hold.



✶ impE introduces (3) [$Q\ a$]

Exercise: Redo this proof using “spec” instead of allE

Problem (III)

Prove that *hasHeart(bush)* given that $\forall x.\ person(x) \rightarrow hasHeart(x)$ *and* *person(bush)* hold.

red assumptions hold

allE introduces assumption (1) $[per(b) \rightarrow heart(b)]$

impE introduces (2)$[heart(b)]$

$$\dfrac{\dfrac{}{\forall x.\, per(x) \rightarrow heart(x)}\,assum \qquad \dfrac{\dfrac{}{per(b) \rightarrow heart(b)}\,by1 \quad \dfrac{}{per(b)}\,assum \quad \dfrac{}{heart(b)}\,by2}{heart(b)}\,impE}{heart(b)}\,allE$$

abbrevs: *heart*(*x*) for *hasHeart*(*x*) and *per*(*x*) for *person*(*x*)

Exercise: Redo this proof using “spec” instead of allE

FOL in Coq

In Coq, FOL is a typed logic with

- types such as `nat` (for natural numbers), `bool` (for boolean values) and `list` (for lists)
- type constructors such as `O` and `S` for constructing `nat` terms: e.g. `O` represents “zero”, `S O` represents “one” and `S (S O))` represents “two”.
- function types written using →, e.g. `nat → nat → nat` *is the type of a function that takes two* `nat` *term arguments and returns a* `nat` *term.*
- parameterized types that allow us to define types parameterized by other types e.g. `nat list` for lists of `nat` terms and `bool list` for lists of `bool` terms.

FOL in Coq (II)

- Consider the mathematical predicate *mod*. In Coq, we could formalize this as:

  ```
  Definition mod (a:nat) (b:nat) (c:nat) : Prop :=
    exists k, a = b * k + c.
  ```

  We can use this definition to write propositions like:

  ```
  forall (a b c d:nat), a = d -> mod d b c = mod a b c.
  ```

- Coq performs type inference. The definition above could have been written as:

  ```
  Definition mod a b c := exists k, a = b * k + c.
  ```

  The proposition could have been written as:

  ```
  forall a b c d, a = d -> mod d b c = mod a b c.
  ```

Coq Demo

Can be found on course webpage ...

Summary

- Introduction to FOL
  - Syntax and Semantics
  - Substitution
  - Intro and elim rules for quantifiers
- Coq
  - Declaring predicates
  - Brief look at types
- Next time: matters of representation