Automated Reasoning

Jacques Fleuriot

September 30, 2013

Natural Deduction in First-order Logic¹ Jacques Fleuriot

¹With contributions by Paul Jackson

Consider the following problem:

- 1. If someone cheats then everyone loses the game.
- 2. If everyone who cheats also loses, then I lose the game.
- 3. Did I lose the game?

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 predicate logic (FOL) extends propositional logic:

Atomic formulas are now assertions about the *properties* of an *individual*.

e.g. an individual might have the property of being a cheat.

- We can use *variables* to denote arbitrary individuals.
 e.g. x is a cheater.
- We can *bind* variables with quantifiers ∀ (for all) and ∃ (there exists).

e.g. for all x, x is a cheater.

- We can use connectives to compose formulas:
 e.g. for all x, if x is a cheater then x loses.
- We can use quantifiers on subformulas.
 e.g. we can formally distinguish between: "if anyone cheats we lose the game" and "if *everyone* cheats, we lose the game".

Given a countably infinite set of (individual) variables $\mathcal{V} = \{x, y, z, ...\}$ and a finite or countably infinite set of function letters \mathcal{F} each assigned a unique arity (possibly 0), then the set of terms is the smallest set such that

• if $f \in \mathcal{F}$ has arity n, and t_1, \ldots, t_n are terms, so is $f(t_1, \ldots, t_n)$.

Remark

If f has arity 0, we usually write f rather than f(), and call f a constant

Formulas of FOL

Given a countably infinite set of predicates \mathcal{P} , each assigned a unique arity (possibly 0), the set of wffs is the smallest set such that

- if $P \in \mathcal{P}$ has arity n, and $t_1, \ldots t_n$ are terms, then $P(t_1 \ldots t_n)$ is a wff;
- ▶ if ϕ and ψ are wffs, so are $\neg \phi$, $\phi \lor \psi$, $\phi \land \psi$, $\phi \longrightarrow \psi$, $\phi \longleftrightarrow \psi$,
- if ϕ are wffs, so are $\exists x. \phi$ and $\forall x. \phi$ for any $x \in \mathcal{V}$;
- if ϕ is a wff, then (ϕ) is a wff.

Remarks

- If P has arity 0, we usually write P rather than P(), and call P a propositional variable
- We assume ∃x and ∀x bind more weakly than any of the propositional connectives.

 $\exists x.\phi \land \psi$ is $\exists x.(\phi \land \psi)$, not $(\exists x.\phi) \land \psi$. (NB: H&R assume $\exists x$ and $\forall x$ bind like \neg .)

6/26

We can now formally represent our problem in FOL:

Assumption 1 If someone cheats then everyone loses the game: $(\exists x. Cheats(x)) \longrightarrow \forall x. Loses(x).$

Assumption 2 If everyone who cheats also loses, then I lose the game : $(\forall x. Cheats(x) \longrightarrow Loses(x)) \longrightarrow Loses(me).$

To answer the question *Did I lose the game?* we need to prove either Loses(me) or $\neg Loses(me)$ from these assumptions.

More on this later.

- An occurrence of a variable x in a formula φ is **bound** if it is in the scope of a ∀x or ∃x quantifier.
- A variable occurrence x is in the scope of a quantifier occurrence ∀x or ∃x if the quantifier occurrence is the first occurrence of a quantifier over x in a traversal from the variable occurrence position to the root of the formula tree.
- If a variable occurrence is not bound, it is free

Example

In

$$P(x) \land \forall x. P(y) \longrightarrow P(x)$$

The first occurrence of x and the occurrence of y are free, while the second occurrence of x is bound.

Substitution Rules

If ϕ is a formula, \boldsymbol{s} is a term and \boldsymbol{x} is a variable, then

 $\phi[s/x]$

is the formula obtained by substituting s for all free occurrences of x throughout ϕ .

Example

$$(\exists x. P(x, y)) [3/y] = \exists x. P(x, 3). (\exists x. P(x, y)) [2/x] = \exists x. P(x, y).$$

If necessary, bound variables in ϕ must be renamed to avoid capture of free variables in s.

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

Informally, an **interpretation** of a formula maps its function letters to actual functions, and its predicate symbols to actual predicates. The interpretation also **specifies some domain** \mathcal{D} (a non-empty set or universe) on which the functions and relations are defined. A formal definition requires some work!

Definition (Interpretation)

An **interpretation** consists of a **non-empty set** \mathcal{D} , called the domain of the intepretation, together with the following assignments

- 1. each **predicate letter** of arity n > 0 is assigned to a subset of $\mathcal{D} \times \cdots \times \mathcal{D}$. Each **nullary predicate** is assigned either **T** or **F**.
- Each function letter of arity n > 0 is assigned to a function (D×···×D) → D. Each nullary function (constant) is assigned to a value in D.

Consider the formula:

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

In one possible interpretation:

- the domain is the set of natural numbers $\mathbb{N} = \{0, 1, 2, 3, \ldots\}$;
- 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 sometimes known as a model.

NB: In H&R, a model is *any* interpretation.

Definition (Assignment)

Given an interpretation M, an assignment s assigns a value from the domain \mathcal{D} to each variable in \mathcal{V} .

We extend this assignment to all terms inductively by saying that

1. if M maps the *n*-ary function letter f to the function F, and

2. if terms t_1, \ldots, t_n have been assigned values $a_1, \ldots, a_n \in D$

then we can assign value $F(a_1, \ldots, a_n) \in \mathcal{D}$ to the term $f(t_1, \ldots, t_n)$.

An assignment s of values to variables is also commonly known as an **environment** and we denote by $s[x \mapsto a]$ the environment that maps $x \in D$ to a (and any other variable $y \in D$ to s(y)).

Definition (Satisfaction)

Given an interpretation M and an assignment s from ${\mathcal V}$ to ${\mathcal D}$

- any wff which is a nullary predicate letter P is satisfied if and only if the interpretation in M of P is T;
- 2. suppose we have a wff ϕ of the form $P(t_1 \dots t_n)$, where P is interpreted as relation R and t_1, \dots, t_n have been assigned values a_1, \dots, a_n by s. Then ϕ is satisfied if and only if $(a_1, \dots, a_n) \in R$;
- any wff of the form ∀x.φ is satisfied if and only if φ is satisfied with respect to assignment s[x → a] for all a ∈ D;
- any wff of the form ∃x.φ is satisfied if and only if φ is satisfied with respect to assignment s[x → a] for some a ∈ D;
- any wffs of the form φ ∨ ψ, φ ∧ ψ, φ → ψ, φ ↔ ψ, ¬φ are satisfied according to the truth-tables for each connective (e.g. φ ∨ ψ is satisfied if and only if φ is satisfied or ψ is satisfied.

Definition (Entailment)

We write $M \models_{s} \phi$ to mean that wff ϕ is satisfied by interpretation M and assignment s.

We say that the wffs $\phi_1, \phi_2, \ldots, \phi_n$ entail wff ψ and write

 $\phi_1,\phi_2,\ldots,\phi_n\models\psi$

if, for any interpretation M and assignment s for which $M \models_s \phi_i$ for all i, we also have $M \models_s \psi$ As with propositional logic, we must ensure that our inference rule

As with propositional logic, we must ensure that our inference rules are *valid*. That is, if

$$\frac{\phi_1 \quad \phi_2 \quad \dots \quad \phi_n}{\psi}$$

then we must have $\phi_1, \phi_2, \ldots, \phi_n \models \psi$.

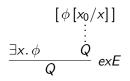
We now consider the additional natural deduction rules we need for FOL.

$$rac{\phi \left[x_{0} / x
ight]}{orall x. \phi}$$
 all

Provided that x_0 is not free in the assumptions.

16/26

$$\frac{\phi\left[t/x\right]}{\exists x.\,\phi} \, exl$$



Provided x_0 does not occur in Q or any assumption other than $\phi [x_0/x]$ on which the derivation of Q from $\phi [x_0/x]$ depends.

Specialisation rule:

$$rac{orall x. \phi}{\phi \left[t/x
ight]}$$
 spec

An alternative universal elimination rule is allE:

$$\frac{ \begin{bmatrix} \phi [t/x] \end{bmatrix} }{ \begin{matrix} \vdots \\ Q \end{matrix}} \frac{\forall x. \phi \qquad Q}{Q} all E$$

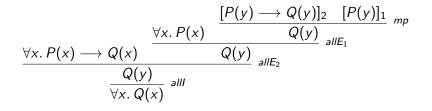
▲□▶ ▲□▶ ▲臣▶ ▲臣▶ 三臣 - のへで

18/26

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

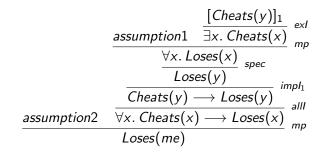
$$\frac{\forall x. P(x)}{P(a)} \sup_{a \neq x} ext$$

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



Prove that Loses(me) given that

1.
$$(\exists x. Cheats(x)) \longrightarrow \forall x. Loses(x)$$
.
2. $(\forall x. Cheats(x) \longrightarrow Loses(x)) \longrightarrow Loses(me)$.



FOL in Isabelle-HOL

Isabelle's HOL object logic is richer than the FOL so far presented. All variables, terms and formulas have **types**.

The type language is built using

base types such as *bool* (the type of truth values) and *nat* (the type of natural numbers).

type constructors such as *list* and *set* which are written postfix, i.e. *nat list*.

function types written using \Rightarrow ; e.g.

 $nat \times nat \Rightarrow nat$

which is a function taking two arguments of type *nat* and returning an object of type *nat*.

type variables such as 'a, 'b etc. These give rise to polymorphic types such as ' $a \Rightarrow 'a$.

Consider the mathematical predicate a = b mod n. We could formalise this operator as: constdefs mod :: "int × int × int ⇒ bool"

"mod (a,b,n)
$$\equiv \exists k. a = k * n + b$$
"

Isabelle performs type inference, allowing us to write:

$$\forall x \ y \ n. \ mod(x, y, n) \longrightarrow mod(y, x, n)$$

instead of

 \forall (x :: int) (y :: int) (n :: int). mod(x, n, y) \longrightarrow mod(y, n, x)

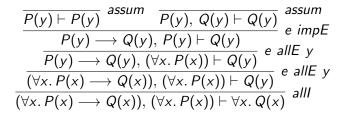
Addendum: FOL L-System Sequent Rules

$$\forall \quad \frac{\Gamma \vdash \phi[x_0/x]}{\Gamma \vdash \forall x. \phi} \text{ alll} \qquad \frac{\Gamma, \phi[t/x] \vdash \psi}{\Gamma, \forall x. \phi \vdash \psi} \text{ e allE } t \qquad \frac{\Gamma, \forall x. \phi, \phi[t/x] \vdash \psi}{\Gamma, \forall x. \phi \vdash \psi} \text{ f spec } t$$

$$\exists \quad \frac{\Gamma \vdash \phi[t/x]}{\Gamma \vdash \exists x. \phi} exl \qquad \frac{\Gamma, \phi[x_0/x] \vdash \psi}{\Gamma, \exists x. \phi \vdash \psi} e exE t \qquad \frac{\Gamma, \forall x. \neg \phi \vdash \bot}{\Gamma \vdash \exists x. \phi} exCIF$$

- Rule prefixes: e = erule, f = frule
- x₀ is some variable not free in hypotheses or conclusion of rule conclusion. With Isabelle, name automatically chosen.
- When t suffix is used above (e.g. as in e allE t), then the term t can be explicitly specified in Isabelle method using a variant of the existing method. e.g. apply (erule_tac x="t" in allE).
- Rule exCIF is a variation on standard Isabelle rule exCI, introduced in the 3rd set of self-help exercises.

Addendum: Example II as FOL Sequent Proof



Introduction to FOL

- syntax and semantics;
- substitution;
- intro and elimination rules for quantifiers.
- Isabelle
 - declaring predicates;
 - a brief look at types.
- Next time: matters of representation.