L12.X.FOL.Exercises

Adolfo Villafiorita

<2020-10-24 Sat>

Outline

Existential with $\wedge$, Universal with $\supset$

Oh my! Delta rules!

Informal to Formal and Tableaux

Validity, Satisfiability, Unsatisfiability

Informal to Formal

Esiste uno studente intelligente

Approach with $\supset$ [wrong!]

$\exists x.(Student(x) \supset Smart(x))$
(issue when premiss is false)

Approach with $\wedge$ [correct!]

$\exists x.(Student(x) \wedge Smart(x))$

Explanation

World

	not Smart	Smart
Student	Sam	Stephan
not Student	Peter	Pamela

Sastifiability wrt an Assignment

$\exists x.(Student(x) \supset Smart(x))$

$\exists x.(Student(x) \wedge Smart(x))$

a[x/. . .]	Student(x)	Smart(x)	$\supset$	$\wedge$	Comment
Sam	T	F	F	F	Equivalent, here
Stephan	T	T	T	T	Equivalent, here
Peter	F	F	T	F	$\supset$ is “wrongly” true
Pamela	F	T	T	F	$\supset$ is “wrongly” true

“wrongly” true: it does not capture our sentence in English

Universal with Implication

Chi studia è intelligente

Approach with $\supset$ [correct!]

$\forall x.(Student(x) \supset Smart(x))$

Approach with $\wedge$ [wrong!]

$\forall x.(Student(x) \wedge Smart(x))$
Issue when we have an interpretation in which some people are not students.

Explanation

World

	not Smart	Smart
Student		Stephan, Sam
not Student	Peter	Pamela

The “cell” student-smart should be empty, because it is not the case that someone is a student and not smart.

Interpretation

$\forall x.(Student(x) \supset Smart(x))$
$\forall x.(Student(x) \wedge Smart(x))$

$\forall$ a[x/...]	Student(x)	Smart(x)	$\supset$	$\wedge$	Comment
Sam	T	T	T	T	Equivalent, here
Stephan	T	T	T	T	Equivalent, here
Peter	F	F	T	F	$\wedge$ makes it “wrongly” false
Pamela	F	T	T	F	$\wedge$ makes it “wrongly” false

Gamma and Delta Rules

1. I can reuse a term with $\forall x.P(x)$ and $\neg\exists x.P(x)$
2. Why do I need to pick a fresh variable with $\exists x.P(x)$ and $\neg\forall x.P(x)$?

Answer:

- The first set of formulas **predicates over the whole domain** and, hence, **I can** pick whatever term I like
- The second set of formulas, instead, asserts the **existence of (at least) one element** in the domain. I don't know which one it is and, hence, I cannot assume it is exactly the one I already picked (I would be arbitrarily restricting models)

Remark:

- See: L11 at the Existential Quantification Rule slide.

Informal to Formal

Gli scienziati leggono i libri. Fred è uno scienziato. Nessun uomo primitivo leggeva libri. Fred legge libri? Fred è un uomo primitivo?

- three sentences in our theory
- two formulas to prove
- problem type: $\Gamma \models \alpha$

Language

General:

- the standard syntactic elements of FOL (logical connectors, variables)

Domain Specific:

- one constant: $\{f\}$
- predicates: S, LL, and P of arity 1

Formalization in First Order Logic

Formalization of formulas in Γ

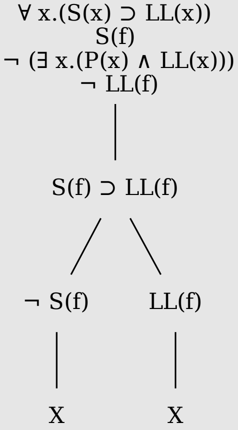
$\forall x.(S(x) \supset LL(x))$
$S(f)$
$\neg\exists x.(P(x) \wedge LL(x))$

Formalization of formulas to prove

$LL(fred)$?
$P(fred)$?
Remark: finite domain, we reason about Fred in PL.

Proving: $LL(f)$

Tableau

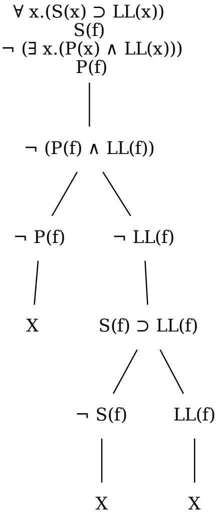


Remarks

- All branches closed, the formula is unsatisfiable
- Since we assume the premiss to hold, it is $-LL(f)$ causing the “troubles”, hence $LL(f)$ must be satisfiable (in fact, if you think about it, using $LL(f)$ would leave the right branch open).
- Some formulas are irrelevant for the proof at hand

Proving: $P(f)$

Different approach: we build the Tableau with $P(f)$.



Definitions

A formula is:

- **Valid** if satisfied by every model
- **Satisfiable** if there is at least one model
- **Unsatisfiable** if there are no models

f	$\neg f$	Comment
valid	unsatisfiable	all for f, nothing for $\neg f$
satisfiable	not valid	some for f, $\neg f$ can't have them all
not valid	satisfiable	
unsatisfiable	valid	

Validity, Satisfiability, Unsatisfiability

How do I check for validity, satisfiability, unsatisfiability?

Preliminary Considerations:

- Valid formulas are such for structural properties (e.g., $A \lor \neg A$)
- Same for unsatisfiable (e.g., $A \land \neg A$)
- For satisfiable formulas, which are not valid, there are models satisfying A and models satisfying $\neg A$

Ho do I check for Validity, Satisfiability, Unsatisfiability?

1. **Meta reasoning**: I reason about the structure of formulas, I use my deduction capabilities to argument
2. **"Semantic" reasoning**: I build the models I need to prove my assertion (however, reasoning about validity/unsatisfiability falls back to case 1, because you need to describe the way in which models are built)
3. **Deductive reasoning**: I use Hilbert or another calculus to prove a property (good for validity and unsatisfiability)
4. **Tableaux**: using the formula in its positive or negative form, to test different properties.

Nice discussion and four exercises on: Checking the validity of a few FOL formulas.

Example 1: $\forall x P(x)$

- $\forall x P(x)$
- Intuitively: satisfiable, since we have a predicate P and I am pretty sure I can find some models satisfying P and some other not satisfying P
- Solution:
  - build two models, one satisfying $\forall x P(x)$ and the other satisfying $\neg \forall x P(x)$
  - use a Tableau, if you are really lost

Example 2: $\forall x.P(x) \supset \exists y P(y)$

- $\forall x.P(x) \supset \exists y P(y)$
- Intuitively: valid, since if a P is true for every element of the domain it will also be true for a specific element and if does not hold for some elements, the premiss if false and the formula still true.
- Solution:
  - building models does not help here: we would need to formalize the intuition above.
  - use a Tableau with the negated formula, which must be unsatisfiable.

Example 2: Tableau

$\neg (\forall x.\ P(x) \supset \exists y.\ P(y))$

|

$\forall x.\ P(x)$
$\neg \exists y.\ P(y)$

|

$P(a)$

|

$\neg P(a)$

|

X

What now?

- The Materials page on the website has been updated with various references and exercises
- A bit of “scavenging” and might be necessary, but there are many examples you can work on
- LogicTools on Datascientia local instance of the Logic Tools, where you can have PL and FOL problems solved. The tools are more relevant for PL than for FOL
- Tree Proof Generator builds Tableaux for PL and FOL