

Inference in first-order logic

- Topics today:
- ① Substitutions
 - ② Propositionalization
 - ③ Completeness and decidability of FOL^{9.2.1} arithmetic
- not based on textbook, but still required for this course
- } based closely on textbook 9.1,

① Substitutions

Notation for substitutions:

- $\{x/A, y/B, z/C\}$

means replace x with A , y with B , z with C .

- $\text{SUBST}(\{x/A\}, \alpha)$

means replace x with A in sentence α

examples:

- $\text{SUBST}(\{x/\text{John}, y/\text{Mother}(\text{John})\}, \text{Knows}(x, y))$

becomes do as exercise

- $\text{SUBST}(\{x/z, y/z\}, \text{Knows}(x, y))$

becomes do as exercise

② Propositionalization

Recall that a major motivation for introducing FOL was that PL required too many facts

e.g.

PL: $\text{CSMajor}(\text{Joanna})$
 \vdots
 $\text{CSMajor}(\text{Tim})$ } one fact for every student

FOL: $\forall x \text{CSMajor}(x)$

But, we can go back the other way if desired.

ie. transform from FOL to PL by eliminating quantifiers and variables.

This requires application of two obvious rules:

UI (Universal Instantiation)

For any variable v and ground term g and sentence α :

a term with no variables

$$\frac{\forall v \alpha}{\text{SUBST}(\{v/g\}, \alpha)}$$

Example: Objects are students in this class.
 Relation $CSMajor(x)$, function $ProjectPartner(x)$
 Knowledge base is:

$$\forall x \ CSMajor(x)$$

Then we can use UI to add any or all of the following to the KB:

$CSMajor(Sasha)$
 $CSMajor(Ashir)$
 $CSMajor(ProjectPartner(y))$
 $CSMajor(ProjectPartner(ProjectPartner(Jon)))$

Existential Instantiation (EI)

For any sentence α , variable v , and constant symbol k that does not appear elsewhere in the knowledge base

$$\frac{\exists v \ \alpha}{\text{subst}(\{v/k\}, \alpha)}$$

Example: Similar scenario to last example,
 Objects are students in this class.
 Relation $CSMajor(x)$, Relation $IsProjectPartner(x, y)$

not the same as the function in the earlier example.

But now suppose entire KB is:

CSMajor (Tucker)

CSMajor (Lam)

$\exists x$ AstronomyMinor(x)

$\exists x$ IsProjectPartner (Lam, x)

— \star

— $*$

Applying GI to \star we can add to the KB:

AstronomyMinor (Someone) — (++)

or even:

AstronomyMinor (Tim)

Tim is just
an arbitrary
label here
(it might not
be the 'real'
Tim)

but not:

AstronomyMinor (Tucker)

but what if Tucker were an Astronomy
minor? Statement (++) covers that!
'Tucker' and 'Someone' can refer to
the same object.

Applying GI to $*$ we can add to the KB:

IsProjectPartner (Lam, Someone2)

or

IsProjectPartner (Lam, Tim2)

← feels wrong.
why is it OK?
see comments above

Remark The above rules can convert any KB from FOL to PL. Then we can apply our PL resolution technique to infer entailment. But there are two problems:

- ① KB could become very big. It's better to do inference within FOL rather than converting to PL. The textbook explains how to do this (e.g. there is an FOL version of resolution in Section 9.5) but we don't study it.
- ② Functions lead to infinitely many possible substitutions e.g. $\text{Father}(x)$, $\text{Father}(\text{Father}(x))$, $\text{Father}(\text{Father}(\text{Father}(x)))$ etc.

In practice this means that

- any entailed sentence can be proved
- we cannot, in general, prove that a sentence is not entailed.

Quoting from textbook p325:

"The question of entailment for first-order logic is semidecidable - that is algorithms exist that say yes to every entailed sentence, but no algorithm exists that also says no to every nonentailed sentence."

③ Summary of completeness and decidability for
PL, FOL, arithmetic

- See next page for summary
- definition of complete : every valid statement has a proof
- definition of decidable : there exists an algorithm that decides whether a given statement is valid

Gödel's completeness theorem (1929)

we resolution

	complete?	decidable?
propositional logic	✓	✓
first-order logic	✓	✗
arithmetic	✗	✗

FOL + an axiom providing induction

Gödel's first incompleteness theorem (1931)

Church (1936)
Turing (1937)

proof: Encode "you can't prove this sentence is true" as a statement in plain arithmetic

proof: If alg. to decide FOL existed, we could write the following program:

```
done = false
while not done:
  if this program halts, done = false
  else done = true
```

but this program is impossible, hence QED.