Tutorium to Introduction to AI, 5th week - Nicolas Höning

Nicolas Höning

May 5, 2006

natural language 2 Pl1
- motivation
- examples

Pl1 2 Prolog
- PL1 vs Prolog
- the algorithm

motivation

- It's all about two things: translating natural language to Pl1 and Pl1 to Prolog (we'll see about that later).
- Pl1 stands for "predicate Logic 1", which is first order logic. Prolog can handle everything that can be expressed in Pl1. If we would translate a problem, given in human language, to Prolog, we first would translate it to Pl1.
- So this is all about translation, or communication between real life and AI.
  But I know, it seems to be a lot of dry theory and work.

example 1

- I'll also translate the examples to the style you need for the homework
- "if someone has blond hair, he/she is british"
- $\forall x : \mathit{has\_blond\_hair}(x) \rightarrow \mathit{british}(x)$
- all x: has_blond_hair(x) $->$ british(x)

example 2

- "no elephant is blue"
- $\forall x : elephant(x) \rightarrow \neg blue(x)$
- all x: elephant(x) $- >$ not blue(x)

example 3

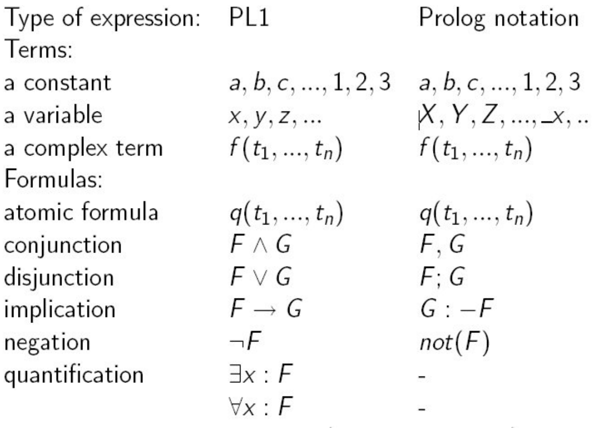
- "there is exactly one Rudi Voeller"
- $\exists x : \mathit{rudivoeller}(x) \wedge \forall y : \mathit{rudivoeller}(y) \rightarrow x = y$
- ex x: rudivoeller(x) and all y: rudivoeller(y) $->$ x=y

what is clark completion?

- the homework uses the term "clark completion". I never heard it myself.
- it basically means: "a iff b" translates to $a \leftrightarrow b$ which is
  $a \longrightarrow b, a \longleftarrow b$
  that leads to an infinite loop. you can leave away one direction in this homework and say that the other direction is done by clark completion...

Pl1 vs Prolog

- Let's first compare them again. We need to get rid of what we can't say and transform what we can say:



Type of expression:	PL1	Prolog notation
Terms:		
a constant	$a, b, c, ..., 1, 2, 3$	$a, b, c, ..., 1, 2, 3$
a variable	$x, y, z, ...$	$X, Y, Z, ..., \_x, ..$
a complex term	$f(t_1, ..., t_n)$	$f(t_1, ..., t_n)$
Formulas:		
atomic formula	$q(t_1, ..., t_n)$	$q(t_1, ..., t_n)$
conjunction	$F \wedge G$	F, G
disjunction	$F \vee G$	$F; G$
implication	$F \rightarrow G$	$G : -F$
negation	$\neg F$	$not(F)$
quantification	$\exists x : F$	-
	$\forall x : F$	-

-

- Here is my example: "In every soccer season, there is at least one really amazing game so that no fan is unhappy." Pl1, anyone?
- $\forall x : \mathit{soccerseason}(x) \wedge$
  $\exists y : ((\mathit{soccergame}(y) \wedge \mathit{amazing}(y)) \rightarrow$
  $\neg \exists x : (\mathit{fan}(x) \wedge \neg \mathit{happy}(x)))$
- Basically, we want to do two things: get rid of quantifiers that we do not have in Prolog (see last slide) and then transform it into Conjunctive Normal Form, which easily can be translated into Prolog.
  we'll go through these steps:
  - variable renaming
  - moving quantifiers out (Prenex form)
  - Eliminating existential quantifiers (skolemization)
  - Conjunctive Normal Form
  - Prolog

variable renaming

- In logic, you can -by using it in the scope of another quantifier- introduce a variable that has the same name as some other that already occured. Of course, we don't want that in our Prolog program (we won't have quantifiers later on)!
- $\forall x : soccerseason(x) \wedge$
  $\exists y : ((soccergame(y) \wedge amazing(y)) \rightarrow$
  $\neg \exists z : (fan(z) \wedge \neg happy(z))$

moving quantifiers out (Prenex form)

- We want all quantifiers to be on the left. We can move them all to the left, now that all variable names are unique. But first, negated quantifiers need to be made into positive ones. This is pretty intuitive if you think about it:
- $\neg \forall a : f(a) \equiv \exists a : \neg f(a)$
- $\neg \exists a : f(a) \equiv \forall a : \neg f(a)$
- $\forall x : soccerseason(x) \wedge$
  $\exists y : ((soccergame(y) \wedge amazing(y)) \rightarrow$
  $\forall z : \neg(fan(z) \wedge \neg happy(z))$

moving quantifiers out (Prenex form)

- now they all can go to the left (keeping the order):
- $\forall x : \exists y : \forall z : soccerseason(x) \wedge$
  $((soccergame(y) \wedge amazing(y)) \rightarrow$
  $\neg(fan(z) \wedge \neg happy(z))$

Eliminating existential quantifiers (skolemization)

- In Prolog, it's a convention that every variable is universally quantified. That's how we get rid of universal quantifiers - ignore them.
  But what about existential quantifiers? They get "skolemized". We say that all the possibly occuring instances are a function of the universally quantified variables that have greater scope (their quantifier is left to the existential).
  Let's call that function sk(). For all x, there are some y that fit them (soccergames y in that season x). sk() computes them.
- $\forall x : \forall z : soccerseason(x) \wedge$
  $((soccergame(sk(x)) \wedge amazing(sk(x))) \rightarrow$
  $\neg(fan(z) \wedge \neg happy(z))$

Eliminating existential quantifiers (skolemization)

- Now we can ignore all quantifiers - finally
- $soccerseason(x) \wedge$
  $((soccergame(sk(x)) \wedge amazing(sk(x))) \rightarrow$
  $\neg(fan(z) \wedge \neg happy(z))$

Conjunctive Normal Form

- The CNF is a conjunction of disjunctions, for example $(a \vee b) \wedge (c \vee d)$
- we can use the rules logic gives us:
  $F \leftrightarrow G \equiv (F \rightarrow G) \wedge (G \rightarrow F)$
  $F \rightarrow G \equiv \neg F \vee G$
  $\neg(F \wedge G) \equiv (\neg F \vee \neg G)$
  $\neg(F \vee G) \equiv (\neg F \wedge \neg G)$
  $F \vee (G \wedge H) \equiv (F \vee G) \wedge (F \vee H)$

Conjunctive Normal Form

- first, let's eliminate the implication
- $soccerseason(x) \wedge$
  $(\neg(soccergame(sk(x)) \wedge amazing(sk(x))) \vee$
  $\neg(fan(z) \wedge \neg happy(z))$

Conjunctive Normal Form

- now we need to move negation inside (applying De Morgan's laws)
- $soccerseason(x) \wedge$ $((\neg soccergame(sk(x)) \vee \neg amazing(sk(x))) \vee$ $(\neg fan(z) \vee happy(z))$

Conjunctive Normal Form

- for machine readability, we can transform it into this form:
- $\{\{soccerseason(x)\}, \{\neg soccergame(sk(x)), \neg amazing(sk(x)), \neg fan(z), happy(z)\}\}$
- the embedding level determines the logical operator:
  first level: $\wedge$, second level: $\vee$

Prolog

- We can turn this into Prolog, if every first-level (conjuncted) term contains at least one positive literal.
  Luckily, we can do it here. If not, the problem might be reformulated. For example, if we formulated $\neg$ happy(z) as unhappy(z), this example could not be translated to Prolog.
- Terms with only one literal become facts. The others can be formulated as implications:
- $soccerseason(x).$
  $happy(z) : - soccergame(sk(x)), amazing(sk(x)), fan(z).$