Artificial Intelligence

Lab 4

Marco Piastra

The world of lists

- Lists of items $[a, b, c, \ldots]$

  $cons/2$
  it's a function that associates items (e.g. a) to a list (e.g. $[b, c]$)
  $cons(a,cons(b,cons(c,nil)))$ is the list $[a, b, c]$

  $Append/3$
  it's a predicate: each pair of lists x and y is associated to their *concatenation* z

  nil
  it's a constant, the *empty list*.

  Shorthand notation (Prolog):
  $[] \Leftrightarrow nil$
  $[a] \Leftrightarrow cons(a,nil)$
  $[a,b] \Leftrightarrow cons(a,cons(b,nil))$
  $[a|[b,c]] \Leftrightarrow cons(a,[b,c])$

Axioms ($\mathbf{AL}$)

$\forall x\ Append(nil,x,x)$
$\forall x \forall y \forall z\ (Append(x,y,z) \rightarrow \forall s\ Append([s,x],y,[s,z]))$

Examples of entailment

$\mathbf{AL} + \exists z\ Append([a],[b,c],z)$	$\models Append([a],[b,c],[a,b,c])$	$= [z/[a,b,c]]$
$\mathbf{AL} + \exists x \exists y\ Append(x,y,[a,b])$	$\models Append([a],[b],[a,b])$	$= [x/[a], x/[b]]$
	$\models Append(nil,[a,b],[a,b])$	$= [x/nil, y/[a,b]]$
	$\models Append([a,b],nil,[a,b])$	$= [x/[a,b], y/nil]$

The world of lists

Problem: $\forall x\ Append(nil, x, x) \models \exists y\ \forall x\ Append(nil, cons(y, x), cons(a, x))$

1: $\forall x\ Append(nil, x, x),\ \neg\exists y\ \forall x\ Append(nil, cons(y, x), cons(a, x))$ (refutation)

2: $\forall x\ Append(nil, x, x),\ \forall y\ \exists x\ \neg Append(nil, cons(y, x), cons(a, x))$ (prenex normal form)

3: $\{Append(nil, x, x)\}, \{\neg Append(nil, cons(y, k(y)), cons(a, k(y)))\}$

($k/1$ is a Skolem function, clausal form)

(N.B. there is no *skolemization* in Prolog : the programmer does it)

The pair of **literals**

$Append(nil, x, x),\ \neg Append(nil, cons(y, k(y)), cons(a, k(y))))$

… contains the same predicate $Append/3$ but the arguments are **<u>different</u>**

There is however an MGU $\sigma = [x/cons(a, k(a)), y/a]$ that yields

$\{Append(nil, cons(a,k(a)), cons(a,k(a)))\}, \{\neg Append(nil, cons(a, k(a)), cons(a, k(a)))\}$

From this, the resolvent is the empty clause.

The world of lists in Prolog

```
% Identical to built-in predicate append/3, although it uses "cons"
% as a defined predicate, thus allowing trace-ability.

append(cons(S,X),Y,cons(S,Z)) :- append(X,Y,Z).
append(nil,X,X).

% WARNING: express your queries with cons. Examples:
% ?- append(cons(a,nil), cons(b,cons(c, nil)),cons(a,cons(b,cons(c, nil)))).
% ?- append(X,Y,cons(a,cons(b,cons(c, nil)))).
```