LIN 177 Homework 2

Hardy Jones
999397426
Professor Ojeda
Winter 2015

1. Given the query

   `?- spanish([eva, mira, a, adan], SM, ES, EM).`

   Prolog attempts to match against the first rule with a variable as none of the facts match. It finds this rule:

```
spanish(A,[verbphrase],B,[property]):-
        spanish(A,[verb,intransitive],B,[property]).
```

   It attempts to unify `A = [eva, mira, a, adan]`. To unify it needs to prove the body. So prolog searches for

   `spanish([eva, mira, a, adan], [verb, intransitive], ES, [property])`

   There are no facts or rules for this, so it backtracks and tries the next rule.

```
spanish(A,[nounphrase,accusative],B,[entity]):-
        spanish(C,[nounphrase],B,[entity]),
        append([a],C,A).
```

   Again, it attempts to unify `A = [eva, mira, a, adan]`. To unify it needs to prove the body. So prolog searches for

   `spanish(C, [nounphrase], ES, [entity]), append([a], C, [eva, mira, a, adan])`

   There are no facts or rules for this, so it backtracks and tries the next rule.

```
spanish(A,[verbphrase],B,[property]):-
        spanish(C,[verb,transitive],D,[relation]),
        spanish(E,[nounphrase,accusative],F,[entity]),
        append(C,E,A),
        append(D,F,B).
```

   Again, it attempts to unify `A = [eva, mira, a, adan]`. To unify it needs to prove the body. The important searches are

   `spanish(C, [verb, transitive], D, [relation]), spanish(E, [nounphrase, accusative], F, [entity])`

   It is also constrained by

   `append(C, E, A)`

   There are no facts or rules that satisfy all constraints, so it backtracks and tries the next rule.

```
spanish(A,[sentence],B,[proposition]):-
        spanish(C,[nounphrase],D,[entity]),
        spanish(E,[verbphrase],F,[property]),
        append(C,E,A),
        append(D,F,B).
```

Again, it attempts to unify `A = [eva, mira, a, adan]`. To unify it needs to prove the body.

So prolog searches for

`spanish(C, [nounphrase], D, [entity]), spanish(E, [verbphrase], F, [property])`

These are constrained also by

`append(C, E, A)`

The fact

```
spanish([eva  ],[nounphrase],[eve],[entity]).
```

matches the first part of the body, so prolog unifys `C = [eva], D = [eve]`.

This means prolog is trying to solve the constraint

`append([eva], E, [eva, mira, a, adan])`

So it attempts to unify `E = [mira, a, adan]`. Now prolog just needs to find a fact or rule satisfying

`spanish([mira, a, adan], [verbphrase], F, [property])`

It finds this rule:

```
spanish(A,[verbphrase],B,[property]):-
        spanish(A,[verb,intransitive],B,[property]).
```

It attempts to unify `A = [mira, a, adan]`. To unify it needs to prove the body. So prolog searches for

`spanish([mira, a, adan], [verb, intransitive], F, [property])`

There are no facts or rules for this, so it backtracks and tries the next rule that matches.

```
spanish(A,[verbphrase],B,[property]):-
        spanish(C,[verb,transitive],D,[relation]),
        spanish(E,[nounphrase,accusative],F,[entity]),
        append(C,E,A),
        append(D,F,B).
```

Again, it attempts to unify `A = [mira, a, adan]`. To unify it needs to prove the body. The important searches are

`spanish(C, [verb, transitive], D, [relation]), spanish(E, [nounphrase, accusative], F, [entity])`

It is also constrained by

`append(C, E, A)`

The fact

```
spanish([mira ],[verb,transitive],[watches],[relation]).
```

matches the first part of the body, so prolog unifies `C = [mira], D = [watches]`.

Given the append constraint, prolog then needs to solve

`append([mira], E, [mira, a, adan])`

So it attempts to unify `E = [a, adan]` which means it needs to solve

`spanish([a, adan], [nounphrase, accusative], F, [entity])`

The only matching rule is

```
spanish(A,[nounphrase,accusative],B,[entity]):-
        spanish(C,[nounphrase],B,[entity]),
        append([a],C,A).
```

So it attempts to unify `A = [a, adan]`. To unify it needs to prove the body. So prolog searches for

`spanish(C, [nounphrase], F, [entity]), append([a], C, [a, adan])`

Prolog can instantiate `C = [adan]` assuming it can solve

`spanish([adan], [nounphrase], F, [entity])`

This is a fact

```
spanish([adan ],[nounphrase],[adam],[entity]).
```

So prolog unifies `F = [adam]`

And we can start filling in the blanks.

When prolog makes to back to the toplevel, we have unified the query with the rule

```
spanish(A,[sentence],B,[proposition]):-
        spanish(C,[nounphrase],D,[entity]),
        spanish(E,[verbphrase],F,[property]),
        append(C,E,A),
        append(D,F,B).
```

And the unifications are

```
spanish([eva, mira, a, adan], [sentence], [eve, watches, adam], [proposition]) :-
    spanish([eva], [nounphrase], [eve], [entity]),
    spanish([mira, a, adan], [verbphrase], [watches, adam], [property]),
    append([eva], [mira, a, adan], [eva, mira, a, adan]),
    append([eve], [watches, adam], [eve, watches, adam]).
```

2.
```
speaks(anne,    english).
speaks(anne,    french).
speaks(anne,    german).
speaks(carol,   english).
speaks(david,   english).
speaks(david,   spanish).
speaks(jacques, english).
speaks(jacques, french).
speaks(nguyen,  english).
speaks(nguyen,  french).
speaks(nguyen,  spanish).
speaks(nguyen,  vietnamese).

trilingual(Person, [Lang1, Lang2, Lang3]) :-
    speaks(Person, Lang1),
    speaks(Person, Lang2),
    speaks(Person, Lang3),
    Lang1 @< Lang2,
    Lang2 @< Lang3.
```

3. (a) This fact is ill-formed as the second argument is two atoms separated by a space.
   (b) This fact is ill-formed as the functor name starts with an uppercase letter.
   (c) This fact is ill-formed as the fact lacks a period.
   (d) This fact is ill-formed as the functor name starts with a number.
   (e) This fact is ill-formed as the functor name starts with a number.

4.
```
english(['Chomsky'],    noun).
english([ate],          verb).
english([balloon],      adjective).
english([balloon],      noun).
english([balloon],      verb).
english([large],        adjective).
english([large],        adverb).
english([large],        noun).
english([red],          adjective).
english([red],          noun).
english([some],         adjective).
english([some],         adverb).
english([some],         pronoun).
english([space],        noun).
english([space],        verb).
english([surprisingly], adverb).
english([too],          adverb).
english([us],           pronoun).
english([will],         noun).
english([will],         verb).
```

5.
```
spanish([nino],  [noun],      [boy],   [masculine, singular]).
spanish([nina],  [noun],      [girl],  [feminine, singular]).
spanish([ninos], [noun],      [boys],  [masculine, singular]).
spanish([ninas], [noun],      [girls], [feminine, singular]).
spanish([alto],  [adjective], [tall],  [masculine, singular]).
spanish([alta],  [adjective], [tall],  [feminine, singular]).
spanish([altos], [adjective], [tall],  [masculine, singular]).
spanish([altas], [adjective], [tall],  [feminine, singular]).
```