Dan Scarafoni

Prolog Week 3/4

Goal- The two main goals were as follows-

1. Create a parser than can parse a limited range (at least four) English sentences or return false otherwise
2. Create a translator based on the parser that could translte an English sentence into first order predicate logic with quantifiers

Methods- The following grammar was used for the project

```
S --> NP VP

NP --> Det Noun
--> Det Noun RELCL
--> Noun
VP --> Verb
--> Verb NP
--> BeVerb Adj
RELCL --> Rel VP
--> Rel NP Verb
```

WHERE-

S- the entire sentence

Np- the noun phrase

Det- determiner (a,an,the)

Noun- The noun in the sentence

RELCL- relative clause

VP- verb phrase

BeVerb- a conjugation of the verb "to be")

Verb- any other verb

The following vocabulary was used, note that if a verb/noun is listed than the program will be able to handle both plural and singular forms (this is denoted by either 's' or 'p' respectively as a parameter, and that the program can correctly assign “a” or “an” to appropriate nouns (this is denoted respectively, 'v' and 'c').

```
noun(apple,s,v).
noun(apples,p,v).
noun(boy,s,c).
noun(boys,p,c).
noun(girl,s,c).
noun(girls,p,c).
noun(government,s,c).
noun(governments,p,c).
noun(watermelon,s,c).
noun(watermelons,p,c).
noun(person,s,c).
noun(people,p,c).
noun(flavor,s,c).
noun(flavors,p,c).
det(a,c).
det(an,v).
det(the,c).
det(the,v).
verb(conscripts,s).
verb(conscript,p).

verb(contains,s).

verb(contain,p).

verb(likes,s).

verb(like,p).

verb(runs,s).

verb(run,p).

beVerb(iss,s).

beVerb(are,p).

adj(evil).

adj(divine).

adj(pacifist).

rel(that).

rel(whom).

rel(who).

rel(which).
```

The program S (which was used for parsing English sentences) required that all sentences be entered as a list of words separated by commas, for instance, to enter the sentence, “the boys run,” the program required is to be entered as s([the,boys,run]).

This parser ran through each term parsing them one at a time. Correctly parsed words were placed in an accumulator list, which at the end was converted into a sentence and printed out. If the parser failed (i.e. The sentence was not grammatically correct) it would return false. NOTE- Certain sentences, such as “the government conscripts people,” are grammatically correct before the end of the sentence (i.e. “the government conscripts,” is acceptable, and the program will return it as an early answer). The ';' key must be hit when running s([the,government,conscripts,people]) At least once until the full sentence is parsed.

The translator (s2) was implemented in a similar way. This parser took in an English sentence and would parse it into first order predicate logic. Because this translator implemented the parser above, it had the same limits (see above). The main difference between the parsers for the first and second project was the determiners. S used determiners, whereas S2, in their place, took quantifiers (all, some). It is also worth noting that this translator/parser required quantifiers for every noun. In other words, s2([all,boys,like,all,girls]) is acceptable, but s2([all,boys,like,girls]) is not.

Results- the first parser was able to correctly parse sentences and check their grammatical validity. The following examples illustrate the programs ability to parse

```
s([the,boy,runs]).
s(NP(det(the),noun(boy),)VP(verb(runs))

s([girls,like,an,apple]).
s(NP(noun(girls)),VP(verb(like),NP(det(an),noun(apple)))

s([a,government,that,conscripts,people,iss,evil]).
s(NP(det(a),noun(government),RELCL(rel(that),verb(conscripts),NP(noun(people)))),VP(beVerb(is),A
dj(evil)))

s([the,boy,whom,the,girl,likes,likes,a,watermelon]).

s(NP(det(the),noun(boy),RELCL(rel(whom),NP(det(the),noun(girl),)VP(verb(likes)),VP(verb(likes),N
P(det(a),noun(watermelon))))))
```

The following examples demonstrate the effectiveness of the translator.

```
s2([all,boys,run]).
        all(X1 boys(X1) => run(X1)))
All boys like all watermelons that contain some divine flavor.
        all(X1 boys(X1) => like(X1,all(X2 (watermelons(X2) & & contain(X2,exists(X3 divine(X3) &
        flavor(X3))))))
s2([some,boy,eats,some,apple]).
        exists(X1 boy(X1) & eats(X1,exists(X2 apple(X2))))
s2([some,governments,conscript,some,pacifist,people])
        exists(X1 governments(X1) & conscript(X1,exists,(X2 pacifist(X2) & people(X2))))
s2([all,governments,that,conscript,some,pacifist,people,are,evil]).
        exists(X1 (governments(X1) & conscript(X1,exists(X2 pacifist(X2) & people(X2))) &
evil(X1))
```

Discussion- The project has shown the powerful and elegant grammars, parsers, and translators that can be made in prolog. As opposed to the other language that this class has had a parser project, c, Prolog does not require multiple files and copious amounts of helper functions and files to function. An effective parser and translator can be written in one file, and can receive input directly and easily from the console.

Referenes- The only resources used were the assignment description posted on the CSC173 website and Programming in Prolog usin gthe ISO Standard By Clocksin and Mellish.