UNIVERSITÄT WIEN
CSLEARN - EDUCATIONAL TECHNOLOGIES

Natural Language Processing

Exercise Sheet 10

Analyzing the Meaning of Sentences

Exercise 1

Translate the following sentences into propositional logic and verify that they can be processed with `Expression.fromstring()`. Provide a key which shows how the propositional variables in your translation correspond to expressions of English.

1. If Angus sings, it is not the case that Bertie sulks.
2. Cyril runs and barks.
3. It will snow if it doesn't rain.
4. It's not the case that Irene will be happy if Olive or Tofu comes.
5. Pat didn't cough or sneeze.
6. If you don't come if I call, I won't come if you call.

Exercise 2

Translate the following sentences into predicate-argument formula of first order logic.

1. Angus likes Cyril and Irene hates Cyril.
2. Tofu is taller than Bertie.
3. Bruce loves himself and Pat does too.
4. Cyril saw Bertie, but Angus didn't.
5. Cyril is a fourlegged friend.
6. Tofu and Olive are near each other.

Exercise 3

Translate the following sentences into quantified formulas of first order logic.

1. Angus likes someone and someone likes Julia.
2. Angus loves a dog who loves him.

3. Nobody smiles at Pat.
4. Somebody coughs and sneezes.
5. Nobody coughed or sneezed.
6. Bruce loves somebody other than Bruce.
7. Nobody other than Matthew loves somebody Pat.
8. Cyril likes everyone except for Irene.
9. Exactly one person is asleep.

Exercise 4

Translate the following verb phrases using λ-abstracts quantified formulas of first order logic.

1. feed Cyril and give a capuccino to Angus
2. be given 'War and Peace' by Pat
3. be loved by everyone
4. be loved or detested by everyone
5. be loved by everyone and detested by no-one

Exercise 5

Consider the following statements:

```
>>> read_expr = nltk.sem.Expression.fromstring
>>> e2 = read_expr('pat')
>>> e3 = nltk.sem.ApplicationExpression(e1, e2)
>>> print(e3.simplify())
exists y.love(pat, y)
```

Clearly something is missing here, namely a declaration of the value of `e1`. In order for `ApplicationExpression(e1, e2)` to be β-convertible to `exists y.love(pat, y)`, `e1` must be a λ-abstract which can take `pat` as an argument. Your task is to construct such an abstract, bind it to `e1`, and satisfy yourself that the statements above are all satisfied (up to alphabetic variance). In addition, provide an informal English translation of `e3.simplify()`.

Now carry on doing this same task for the further cases of `e3.simplify()` shown below.

```
>>> print(e3.simplify())
exists y.(love(pat,y) | love(y,pat))

>>> print(e3.simplify())
walk(pat)
```

Exercise 6

As in the preceding exercise, find a λ-abstract `e1` that yields results equivalent to those shown below.

```
>>> e2 = read_expr('chase')
>>> e3 = nltk.sem.ApplicationExpression(e1, e2)
>>> print(e3.simplify())
\x.all y.(dog(y) -> chase(x,pat))

>>> e2 = read_expr('chase')
>>> e3 = nltk.sem.ApplicationExpression(e1, e2)
>>> print(e3.simplify())
\x.exists y.(dog(y) & chase(pat,x))

>>> e2 = read_expr('give')
>>> e3 = nltk.sem.ApplicationExpression(e1, e2)
>>> print(e3.simplify())
\x0 x1.exists y.(present(y) & give(x1,y,x0))
```

Exercise 7

As in the preceding exercise, find a λ-abstract `e1` that yields results equivalent to those shown below.

```
>>> e2 = read_expr('bark')
>>> e3 = nltk.sem.ApplicationExpression(e1, e2)
>>> print(e3.simplify())
exists y.(dog(x) & bark(x))

>>> e2 = read_expr('bark')
>>> e3 = nltk.sem.ApplicationExpression(e1, e2)
>>> print(e3.simplify())
bark(fido)

>>> e2 = read_expr('\\P. all x. (dog(x) -> P(x))')
>>> e3 = nltk.sem.ApplicationExpression(e1, e2)
>>> print(e3.simplify())
all x.(dog(x) -> bark(x))
```