Computational Semantics, Ling 334
Wintersemester 2012
University of Konstanz
Miriam Butt and Maribel Romero

Homework 1

1 Knowledge Bases

1.1 First-Order Models

Create a model M for the following situation/world. Remember to specify a Domain D and a function F.

> Augustus, Butch and Cain are criminals. Augustus is from Manhattan and Butch and Cain from Long Island. Either Augustus or Butch stole the Red Ruby. Dr. Volt suspects Augustus and Butch. Mr. Grinch suspects Augustus, Butch, Cain and Dr. Volt. If Augustus stole the Red Ruby, the Red Ruby is in Augustus' house. If Butch stole the Red Ruby, it is (also) in Ausgustus house.

1.2 Prolog

Show how this Model could be realized as a Prolog *knowledge base.*

2 First Order Predicate Logic

- Formulate the following statements in first-order predicate logic
- and in Prolog.

(1) a. Dr. Volt suspects everybody Grinch suspects.
b. Dr. Volt suspects every criminal except Cain.
c. The Red Ruby is in Augustus' house.
d. If a thief from the Manhattan stole the Red Ruby, the Red Ruby is in his house.

- Which of the statements are true and which are false?

3 Recursion and Lists in Prolog

3.1 Correspondences

Assume you have a *knowledge base* that contains the names of linguists paired with something they are famous for.

```
linguist(chomsky, generativeSyntax).
linguist(kaplan, computationalLFG).
linguist(bresnan, lfg).
linguist(kamp, drt).
linguist(bos, udrt).
linguist(karttunen, finiteState).
linguist(kiparsky, everything).
linguist(clark, languageAcquisition).
linguist(partee, semantics).
linguist(asher, sdrt).
```

Write a predicate `name/2` that gives a list of areas for the relevant name.

For example:
`name([chomsky, kaplan, bos], V).`
should give:
`V = [generativeSyntax, computationalLFG, udrt]`

Conversely, it should of course be possible to specify the area and get the corresponding last names.

How to proceed: first define the base case (list is empty), then start with the head of the first list and keep working through it recursivly.

3.2 List Comparison

Write a predicate `swap/3` that compares lists with one another and that checks if the lists are the same, except for the second and the third element on the list. These should be swapped. The predicate should also tell you which two elements are the ones that are swapped (e.g., tell you what in particular they are).

So `swap([1,2,3],[1,3,2],X).`
should give you:
`X = [2,3]`