

## CS246 - Answers / Grading Notes for Exercise Set 2

### Exercise 5.1 (5 points)

Pretty simple, just make sure all of your objects have a primary class.

### Exercise 5.2 (5 points)

(3 points) Rules for inheritance of predicates, for example:

```
(<= (person.instance ?x) (student.instance ?x))
(<= (person.instance ?x) (advisor.instance ?x))
```

(2 points) Adding person attributes (firstname and lastname) to advisor and student. This can be done manually or with rules

### Exercise 5.3 (5 points)

The following rules do the trick:

```
(<= (advisor.student ?a ?s) (student.advisor ?s ?a))
```

```
(=> (advisor.student ?a ?s) (student.advisor ?s ?a))
```

```
(=> (not (advisor.student ?a ?s)) (not (student.advisor ?s ?a)))
```

### Exercise 5.4 (7 points)

Conditions:

```
(person.name ?person ?name)
```

```
(stringposition " " ?name ?spacepos)
```

```
(- ?spacepos 1 ?firstname_end)
```

```
(+ ?spacepos 1 ?lastname_begin)
```

```
(stringlength ?name ?lastname_end)
```

```
(stringsubseq ?name 1 ?firstname_end ?firstname)
```

```
(stringsubseq ?name ?lastname_begin ?lastname_end ?lastname)
```

Conclusions:

(person.firstname ?person ?firstname)

(person.lastname ?person ?lastname)

(not (person.name ?person ?name))

*Exercise 5.5 (5 points)*

Not much to say about this.

*Exercise 5.6 (5 points)*

The trick here was that the order of the antecedents in the recursive rule mattered, or else you could get an infinite recursion.

(<= (ancestor ?x ?y) (student.advisor ?x ?y))

(<= (ancestor ?x ?y) (student.advisor ?x ?z) (ancestor ?z ?y))

*Exercise 5.7 (5 points)*

The easiest way to do this was to notice the symmetry between ancestor and descendant:

(<= (descendant ?x ?y) (ancestor ?y ?x))

*Exercise 5.8 (5 points)*

There are many ways to do this. The simplest is the following:

(<= (familymember ?x ?y)  
    (student.ancestor ?x ?z)  
    (student.ancestor ?y ?z))

*Exercise 5.9 (20 points)*

(3 points) You should have run some variant of the rules from 5.8 in XForm to populate the founder's familymembers. For example (together with the above query rules in the system):

Conditions:

(familymember ?x ?y)  
(unprovable (student.instance ?x))

Conclusions:

(familymember ?x ?y)

(5 points) Then you should have deleted the current query rules and replaced them with something that reads off the stored family members from the founder like:

(<= (familymember ?x ?z)  
    (ancestor ?x ?y)  
    (unprovable (student.instance ?y))  
    (familymember ?x ?z))

(4 points) You should deal with the case of a new family being created with the following rule:

(=> (advisor.instance ?x)  
    (familymember ?x ?x))

(5 points) You should deal with the case where a student is added to an existing advisor.

(=> (student.advisor ?x ?y)  
    (ancestor ?x ?z)  
    (unprovable (student.instance ?z))  
    (familymember ?z ?x))

(3 points) A rule to deal with the case when a student is added as an advisor of another student:

(=> (student.advisor ?x ?y)  
    (advisor.instance ?y))

### *Exercise 5.10 (10 bonus points)*

Note that the only way to merge two trees is to assign the founder of one of the trees an advisor.

(4 points) Removing old stored facts:

(=> (student.advisor ?x ?y)  
    (familymember ?y ?z)  
    (not (familymember ?y ?z)))

(6 points) Adding new facts:

(=> (student.advisor ?x ?y)  
(ancestor ?y ?z)  
(uprovable (student.instance ?z))  
(descendant ?z ?w)  
(familymember ?z ?w))