INF5390-2014 – Kunstig intelligens

Exercise 2 Solution

Roar Fjellheim

Exercise 2.1: First-Order Logic (INF5390-05)

Represent the following sentences in first-order logic, using a consistent vocabulary (which you must define):

a. Every person who dislikes all vegetarians is smart

b. No person likes a smart vegetarian

Exercise 2.1: First-Order Logic (INF5390-05)

c. There is a woman who likes all men who are not vegetarians

$$\exists x Woman(x) \wedge \forall y Man(y) \wedge$$
$$\neg Vegetarian(y) \Rightarrow Likes(x, y)$$

d. There is a barber who shaves all men in town who do not shave themselves

$$\exists x Barber(x) \wedge \forall y Man(y) \wedge$$
$$\neg Shaves(y, y) \Rightarrow Shaves(x, y)$$

Exercise 2.1: First-Order Logic (INF5390-05)

e. Politicians can fool some of the people all of the time, and they can fool all of the people some of the time, but they can't fool all of the people all of the time

$$\forall x Politician(x) \Rightarrow$$
$$(\exists y \forall t Person(y) \wedge Fools(x, y, t))$$
$$\wedge (\exists t \forall y Person(y) \Rightarrow Fools(x, y, t))$$
$$\wedge \neg(\forall t \forall y Person(y) \Rightarrow Fools(x, y, t))$$

Exercise 2.2: Knowledge Engineering in FOL (INF5390-06)

- Express the following statement in first order logic

  At least one sock of each pair of socks will eventually get lost

- Use the following vocabulary:
  - √ *Sock(x)* – *x* is a sock
  - √ *Pair(x, y)* – *x* and *y* are a pair
  - √ *Now* – current time
  - √ *Before(t1, t2)* – the time *t1* is before the time *t2*
  - √ *Lost(x, t)* - *x* is lost at time *t*

Exercise 2.2: Knowledge Engineering in FOL (INF5390-06)

- Answer:

$$\forall s1, s2\, Sock(s1) \wedge Sock(s2) \wedge Pair(s1, s2) \Rightarrow$$
$$(\exists t1\, Before(Now, t1) \wedge \forall t\, Before(t1, t) \Rightarrow Lost(s1, t)) \vee$$
$$(\exists t2\, Before(Now, t2) \wedge \forall t\, Before(t2, t) \Rightarrow Lost(s2, t))$$

Exercise 2.3: Classical Planning (INF5390-08)

There are many ways to characterize planners. For each of the following dichotomies, explain what they mean:

a. **State space** vs. **plan space**
State space planner tries to find path through problem space. Plan space planner searches space of possible plans.

b. **Progressive** vs. **regressive**
Progressive means starting at start and move towards goal. Regressive means starting in goal and moving backwards.

Exercise 2.3: Classical Planning (INF5390-08)

c. Least commitment vs. **more commitment**
Least commitment means making abstract plans with no more choices than are strictly necessary. More commitment makes more choices.

d. Total order vs. **partial order**
Total order represents plans as a strict sequence of steps. Partial order impose constraints on sequence, but may not totally order the steps.

Exercise 2.3: Classical Planning (INF5390-08)

Given the action schemas and initial state shown on slide no. 10 of INF5390-08, list all concrete instances of the schema *Fly(p, from, to)* that are applicable in the state described by:

$$\boldsymbol{At(P1, JFK) \wedge At(P2, SFO) \wedge Plane(P1) \wedge Plane(P2) \wedge Airport(JFK) \wedge Airport(SFO)}$$

Example - Air cargo planning in PDDL

- ***Init***$(At(C1, SFO) \wedge At(C2, JFK) \wedge At(P1, SFO) \wedge At(P2, JFK) \wedge Cargo(C1) \wedge Cargo(C2) \wedge Plane(P1) \wedge Plane(P2) \wedge Airport(JFK) \wedge Airport(SFO))$
- ***Goal***$(At(C1, JFK) \wedge At(C2, SFO))$
- ***Action***$(Load(c, p, a),$
  PRECOND: $At(c, a) \wedge At(p, a) \wedge Cargo(c) \wedge Plane(p) \wedge Airport(a),$
  EFFECT: $\neg At(c, a) \wedge In(c, p))$
- ***Action***$(Unload(c, p, a),$
  PRECOND: $In(c, p) \wedge At(p, a) \wedge Cargo(c) \wedge Plane(p) \wedge Airport(a),$
  EFFECT: $At(c, a) \wedge \neg In(c, p))$
- ***Action***$(Fly(p, from, to),$
  PRECOND: $At(p, from) \wedge Plane(p) \wedge Airport(from) \wedge Airport(to),$
  EFFECT: $\neg At(p, from) \wedge At(p, to))$

Exercise 2.3: Classical Planning (INF5390-08)

Applicable instances of *Fly(p, from to)*:

- *Fly(P1, JFK, SFO)* – p=P1, from=JFK, to=SFO
- *Fly(P1, JFK, JFK)* – p=P1, from=JFK, to=JFK
- *Fly(P2, SFO, JFK)* – p=P2, from=SFO, to=JFK
- *Fly(P2, SFO, SFO)* – p=P2, from=SFO, to=SFO

Applicable: Preconditions satisfied

Instance: All variables replaced by constants