Harry Langford
hjel2@cam.ac.uk

Part I
Sean's exercise sheet part 3

1 Knowledge representation and reasoning

1. There were in fact *two* queries suggested in the notes for obtaining a sequence of actions. The details for

$$\exists a \exists s.\text{Sequence}(a, s_0, s) \wedge \text{Goal}(s)$$

were provided, but earlier in the notes the format

$$\exists \text{actionList}.\text{Goal}(\dots \text{actionList} \dots)$$

was suggested. Explain how this alternative form of query might be made to work.

Informally, we could hardcode the initial state s_0 into the Goal predicate. We then use it as a wrapper to the initial query. Using prolog notation, we would encode this as:

```
goal(ActionList) :- sequence(ActionList, s0, S), isGoal(S).
```

This has the disadvantage of requiring the query to hardcode the start state – and not exposing the final state to the caller – perhaps for the purposes of further queries or actions.

2. Making correct use of the situation calculus, write the sentences in FOL required to implement the `Shoot` action in Wumpus World. Write further sentences in FOL to allow movement and change of orientations.

$$\begin{gathered} Poss(a, s) \implies (Poss(\text{Shoot}, result(a, s)) \iff (\\ (a = \text{grab} \wedge At(\ell, s) \wedge Available(\text{Arrow}, \ell, s)) \\ \vee \\ (Poss(\text{Shoot}, s) \wedge \neg(a = \text{Shoot} \vee a = release(Arrow))) \\ )) \end{gathered}$$

$$\begin{gathered} Poss(a, s) \implies (At(\ell, result(a, s)) \iff (\\ At(\ell', s) \wedge a = go(\ell', \ell) \\ \vee \\ At(\ell', s) \wedge a \neq go \wedge \ell = \ell' \\ )) \end{gathered}$$

$$\begin{gathered} Poss(a, s) \implies (Direction(north, result(a, s)) \iff (\\ Direction(north, s) \wedge \neg(a = turnLeft \vee a = turnRight) \\ \vee \\ Direction(east, s) \wedge a = turnLeft \\ \vee \\ Direction(west, s) \wedge a = turnRight \\ )) \end{gathered}$$

$$
\begin{gathered}
Poss(a,s) \implies (Direction(east, result(a,s)) \iff (\\
Direction(east,s) \land \neg(a = turnLeft \lor a = turnRight) \\
\lor \\
Direction(south,s) \land a = turnLeft \\
\lor \\
Direction(north,s) \land a = turnRight \\
))
\end{gathered}
$$

$$
\begin{gathered}
Poss(a,s) \implies (Direction(south, result(a,s)) \iff (\\
Direction(south,s) \land \neg(a = turnLeft \lor a = turnRight) \\
\lor \\
Direction(west,s) \land a = turnLeft \\
\lor \\
Direction(east,s) \land a = turnRight \\
))
\end{gathered}
$$

$$
\begin{gathered}
Poss(a,s) \implies (Direction(west, result(a,s)) \iff (\\
Direction(west,s) \land \neg(a = turnLeft \lor a = turnRight) \\
\lor \\
Direction(north,s) \land a = turnLeft \\
\lor \\
Direction(south,s) \land a = turnRight \\
))
\end{gathered}
$$

2 2003 Paper 9 Question 8

https://www.cl.cam.ac.uk/teaching/exams/pastpapers/y2003p9q8.pdf

(a) Explain what the terms *ontological commitment* and *epistemological commitment* mean in the context of a language for knowledge representation and reasoning. what are the ontological and epistemological commitments made by propositional and by first order logic?

Ontological Commitment is something which is true in the world. In a language for knowledge representation and reasoning, Ontological commitments represent the set of things which are actually true.

Epistemological Commitment is something which an agent believes to be true. In a language for knowledge representation and reasoning, this is the set of things which the agent can infer to be true.

(b) You wish to construct a robotic pet cat for the purposes of entertainment. One purpose of the cat is to scratch valuable objects when the owner is not present. Give a brief general description of *situation calculus* and describe how it might be used for knowledge representation by the robot. Include in your answer one example each of a *frame axiom*, an *effect axiom* and a *successor-state axiom*, along with example definitions of suitable predicates and functions.

The situation calculus is a restricted form of first order logic which operates on "situations". In the calculus, we use a knowledge base `KB` to reason about predicates that are true. This knowledge base defines the initial situation s_0 and a set of rules for subsequent situations. The "user" can then input the sequence of actions which they have taken and use the situation calculus to determine whether this is legal and if so what the resulting situation will be.

A frame axiom is used to model the ways in which the world stays the same. For example, if a predicate is true and we do not make it false, then it is still true!

$$Have(\text{Arrow}, s) \land a \neq \text{shoot} \land a \neq \text{drop}(\text{Arrow}) \implies Have(\text{Arrow}, result(a, s))$$

An effect axiom determines the predicates which have been made true as a result of performing an action:

$$Poss(\text{grab}, o, s) \implies Have(o, result(\text{grab}, s))$$

A successor-state axiom combines a frame axiom and an effect axiom into a single axiom which determines under what conditions a predicate will hold. For example:

$$\begin{gathered} Poss(a, s) \implies (Have(\text{gold}, result(a, s)) \iff ((\\ a = \text{grab} \land Available(\text{gold}, s)) \\ \lor \\ (Have(\text{gold}, s) \land \neg(a = drop)) \\ )) \end{gathered}$$

(c) Give a brief description of the *representational frame problem*, the *inferential frame problem*, the *qualification problem* and the *ramification problem.*

In most environments, there are many fluents and each action only changes a very small proportion of them. This is the frame problem.

- Representational Frame Problem

  The Representational Frame Problem is a subset of the frame problem which focuses on using a minimal number of frame axioms – proportional to the amount of variables which an action actually changes.

- Inferential Frame Problem

  The Inferential Frame Problem is a subset of the frame problem which focuses on efficiently carrying all the unchanged fluents through many timesteps.

- Qualification problem

  Often, the number of prerequisites which an action requires is immense and cannot be efficiently modelled. The qualification problem refers to the problem of efficiently listing the requirements for every action.

- Ramification problem

  Actions tend to have a wide range of implicit consequences at a very high level of detail. The Ramification problem is the problem of carrying through all the implicit effects of actions.

  For example, if I pick up a pen then I have not directly encoded the property that I am also picking up the ink inside the pen – this could lead to strange plans involving "pick up pen" and "pick up ink" as separate steps.