Knowledge Representation and Reasoning – Assignment I Model-based Diagnosis

1 Introduction

Model-based reasoning is one of the central topics of knowledge representation and reasoning in articial intelligence. This first assignment of the course "Knowledge Representation and Reasoning" has the aim of increasing your understanding of Prolog and model-based reasoning through modelling and the implementation of algorithms. The first part of this assignment consists of implementing the hitting-set algorithm (Task I), the second part of this assignment consists of modelling and diagnosing a *smart home* (Task II).

You need to carry out the tasks on your own or in pairs.

2 Warm-up exercise: implementation of trees

In contrast to imperative programming languages, Prolog does not contain constructs for "real" data structures in the language. However, using *terms*, it is possible to *represent* any data structure by a compound term that constructs such a data structure. While you have seen some basic data structures before (e.g. lists), the following exercise illustrates this further using tree structures in Prolog. It is highly recommended to use this exercise as a source of inspiration for Task I. You do not need to submit the warm-up exercises and you may ask the instructors for help, if necessary.

(a) Consider for example a binary tree which we can build up using the following two terms:

- a constant `leaf` which represents a leaf node;
- a function `node` with arity 2, which, given 2 nodes (its children), returns a tree.

Define a predicate `isBinaryTree(Term)`, which is true if and only if `Term` represents a tree. Test this on compound terms such as:

- `leaf` (true)
- `node(leaf)` (false)
- `node(leaf,leaf)` (true)
- `node(leaf,node(leaf,leaf))` (true)

(b) Define a predicate `nnodes(Tree, N)`, which computes the number of nodes N of a given `Tree`, e.g.

```
?- nnodes(leaf,N).
N = 1.

?- nnodes(node(leaf,node(leaf,leaf)),N).
N = 5.
```

(c) Extend the representation of the tree so that each node (and leaf) is labelled with a number. Adapt your definition of `isBinaryTree` and `nnodes` to reflect this representation.

(d) Define a predicate `makeBinary(N, Tree)`, which gets some number $N \geq 0$ and returns a tree where the root node is labelled by N. Furthermore, if a node is labelled by $K > 0$, then it has children that are labelled by $K - 1$. If a node is labelled by 0, then it does not have children.

(e) Now extend the representation of your tree so that each node can have an *arbitrary* number of children. Also define a `nnodes` predicate for these kind of trees.

3 Task I: Implementation of the hitting-set algorithm

The *hitting-set algorithm* acts as the core of consistency-based diagnosis, and has been discussed during the course. In this task, you will implement this algorithm in Prolog.

(0)
- Download `tp.pl` and `diagnosis.pl` from the webpage or from blackboard.
- In `tp.pl`, scroll down to the bottom and inspect the definition of `tp/5`.
- In `diagnosis.pl`, inspect the definitions of the diagnostic problems in the file. Formulas are represented by Prolog terms where constants (and functions) are interpreted as predicates, with additional operators `~` (*not*), `,` (*and*), `;` (*or*), `=>` (*implies*), `<=>` (*iff*), and quantification $\{$`all`, `or`$\}$ `X:`f, where `X` is a (Prolog) variable and f is a term which contains `X`. Since `,` and `;` are also Prolog operators, it is often required to put brackets around these terms. For example, the formula $\forall_x(P(x) \vee Q(x))$ can be represented by the term `(all X:(p(X) ; q(X)))`.
  Experiment with `tp/5` and determine some conflict sets for the diagnostic problems.

(1) Define a Prolog representation for hitting set trees. Program a corresponding predicate `isHittingSetTree(Tree)` and convince yourself that this predicate is true if and only if `Tree` is a hittingset tree.

(2) Use this representation to develop a Prolog program of the hitting-set algorithm, preferably using refinements to prune the search space as described in [3] (see blackboard or the library for the paper). The input to the program is a diagnostic problem; the output is the set of all *minimal* hitting sets (i.e. diagnoses).

(3) Evaluate your program using the given diagnostic problems.

4 Task II: Modelling a smart home

There is an increasing interest to use information technology and artificial intelligence in the home environment. This is also sometimes called home automation or *domotics*. You can think of topics such as centralized control of lighting, health care systems at home (eHealth), but also remote interfaces to home appliances.

In this task, you are expected to model a 'smart' lighting system for a room in a house. This systems should at least have the following properties:

- there should be at least two lamps in the room (l_1 and l_2);
- there is a motion sensor (s) in the room;
- l_1 is turned on by a remote control;
- l_2 should be turned on if s senses motion in the room;
- there is a (central) switch that turns off all the lighting in the room;
- of course, the lamp can only be lit if there is a live wire coming from the service panel to the room, and is somehow connected to the lamp (through switches, and possibly other components).

You need to carry out the following tasks:

(1) Design a system (as a circuit diagram) for such a lighting system controlled by sensors and switches. This representation should be at the level of abstraction as the description. In particular, it should be specific enough to reason about the relevant components and observations, but it should not be *too specific*. For example, you could represent the actual voltages and currents in such an electrical system, but you may abstract from these details when reasoning about whether a certain component behaves normally.

(2) Formalize this system in first-order logic so that it can be used as input for a consistency-based diagnosis tool.

(3) Consider the following scenario: John enters the room that you designed, but unfortunately there is no light. John observes that the central switch is on. Moreover, if he tries to turn on l_1 using his remote, l_1 produces light. Obviously, John is puzzled and needs technical assistance.

   - Model this scenario as a set of observations using terms of the system description.
   - Define the system description and these observations as a diagnostic problem.
   - Apply your program (developed in Task I) to find the diagnoses.

(4) Evaluate your solution. In particular, consider the question why or why not the central switch should be a diagnosis.

5 What to submit?

To summarise, you have to submit a report before **Friday, 7th December, 2012** containing the following sections:

(1) A brief introduction to your report.

(2) A description of the most important ideas behind the implementation of Task I.

(3) The lighting system model and the most important ideas of your system.

(4) Examples that show that your Prolog program works like expected and that it can be used to diagnose the lighting system.

(5) A reflection on your code (such as: what are the limitations? how can it be improved? what are the problems encountered?)

Please submit the report and your Prolog code on blackboard!

References

[1] R. Greiber, B.A. Smith and R.W. Wilkerson (1989). A correction to the algorithm in Reiter's theory of diagnosis. *Artificial Intelligence*, **41**, 79–88.

[2] P.J.F. Lucas (1997). Symbolic diagnosis and its formalisation. *The Knowledge Engineering Review*, **12**(2), 109–146.

[3] R. Reiter (1987). A theory of diagnosis from first principles. *Artificial Intelligence*, **32**, 57–95.