Exam Modelling of Complex Systems: feedback

Exam of 14/01/2019

1 Theory: small questions

1. In the course, it is stated that the CTL* state formula $\mathrm{E}[\mathrm{G}\,\mathrm{F}\,p]$ cannot be expressed in CTL nor in LTL. What does it express? Prove that it cannot be expressed in LTL.

   $\mathrm{E}[\mathrm{G}\,\mathrm{F}\,p]$ expresses that there exists a path (in the implicit state) such that p is infinitely often true.

   To prove that it cannot be expressed in LTL, take a transition structure in which it is true, and such that it is false in a substructure:

   ```
   Node = {a,b}
   Edge = { (a,a); (b,b); (a,b)  }
   L = { a -> {~p} ;  b -> {p} }
   ```

   In state a, the formula is true due to the path to b.

   In the substructure where only a is present, the formula is false.

2. Consider the following FO(.) definition of unary predicate A in terms of parameter $G/2$, a graph predicate.

   $$\left\{ \ \forall x(A(x) \leftarrow \forall y(G(x,y) \Rightarrow A(y))) \ \right\}$$

   What is the set A defined here? Draw a small structure in which A is not the complete domain.

   This is the same as Exercise 3.3.20 in the course.

   The definition is an inductive definition that defines A as the set of nodes that have only finite outgoing paths. We prove this by an inductive argument.

   What is the base case of this definition? The base case is any x that has no outgoing edgesx. For such a node x, the body of the rule is trivially satisfied, since $\forall y(\neg G(x,y))$ holds which trivially entails the body of the unique rule. So these nodes form the first layer of elements of A.

   What is the inductive case? Suppose that we obtained some intermediate set A. The induction hypothesis is that all elements of A have only finite outgoing paths. The inductive rule now adds all nodes x such that for any outgoing edge $G(x,y)$, y belongs to A. Since any such y has only finite outgoing paths, so has x.

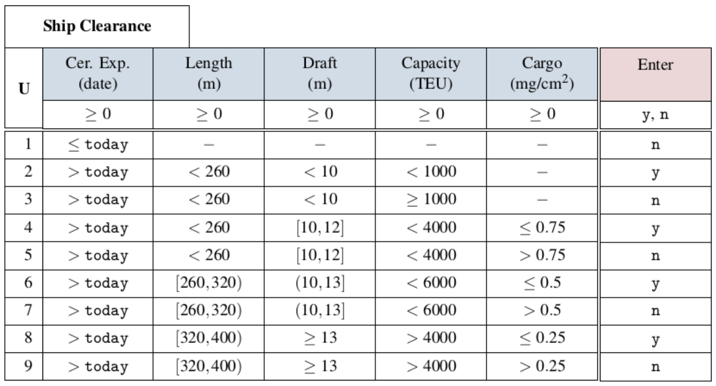
   Thus, we have shown by induction that any element in A has only finite outgoing paths.

   A small structure where A is not all nodes is

```
Node = {a,b,c}
G = { (a,a); (c,b) }
A = { b, c }
```

3. Two watched literal technique : it is a technique to optimize unit clause propagation. See course.

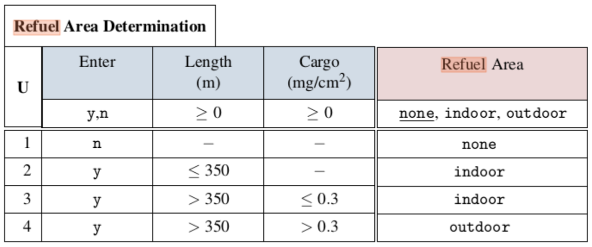
2 Modeling DMN in FO(.)



Ship Clearance

U	Cer. Exp. (date)	Length (m)	Draft (m)	Capacity (TEU)	Cargo (mg/cm^2)	Enter
	≥ 0	≥ 0	≥ 0	≥ 0	≥ 0	y, n
1	$\leq$ today	–	–	–	–	n
2	$>$ today	< 260	< 10	< 1000	–	y
3	$>$ today	< 260	< 10	≥ 1000	–	n
4	$>$ today	< 260	$[10, 12]$	< 4000	≤ 0.75	y
5	$>$ today	< 260	$[10, 12]$	< 4000	> 0.75	n
6	$>$ today	$[260, 320)$	$(10, 13]$	< 6000	≤ 0.5	y
7	$>$ today	$[260, 320)$	$(10, 13]$	< 6000	> 0.5	n
8	$>$ today	$[320, 400)$	≥ 13	> 4000	≤ 0.25	y
9	$>$ today	$[320, 400)$	≥ 13	> 4000	> 0.25	n

Table 1: DMN representation of the *ship clearance* decision of Figure 1b



Refuel Area Determination

U	Enter	Length (m)	Cargo (mg/cm^2)	Refuel Area
	y,n	≥ 0	≥ 0	none, indoor, outdoor
1	n	–	–	none
2	y	≤ 350	–	indoor
3	y	> 350	≤ 0.3	indoor
4	y	> 350	> 0.3	outdoor

Table 2: DMN representation of the *refuel area determination* decision of Figure 1b

Decision Modeling Notation (DMN) is a well-known tabular knowledge representation format used in business applications. The DMN tables 1 and 2 represent knowledge for a ship to enter or refuel at an unnamed Dutch harbour.

- The first row in each table specifies the names of the attributes and the possible values.
- The first column in each table is just a numbering of rows.
- The table “Ship Clearance” specifies whether a ship may enter the harbour for loading or unloading as expressed by the attribute “Enter”. It is the “output” attribute of this table. As for the other attributes, “Cer. Exp.” stands for Certificate Expiration Date; the attribute “Cargo” is a measure of the residual load of the boat; “Length”, “Draft” (in Dutch: diepgang), “Capacity” have the obvious meaning.

- E.g., the second row specifies that a ship with valid certificate and length less than $260m$, draft less than $10m$ and capacity less than 100 *TEU* may enter.
- The table "Refuel Area Determination" specifies whether and where a ship may refuel: not, or in the indoor fuel station or in the outdoor fuel station. Note that "Enter" defined in the first table, is an input attribute of this one.

The questions are to express (part of) these tables in FO(.) and use the theories to solve questions. **A model should represent a correct decision regarding entry and fuel station of the ship.** Solve the following questions:

1. Introduce your vocabulary: you may use types such as rational numbers and dates; for the attribute *Enter* use a constant symbol; for the attribute *RefuelArea*, use a unary predicate; for the other attributes, you may choose. Indicate for each table whether it is a definition or a set of constraints. If it is a definition, express it as a definition. For the first table express line 2 and 8. For the second table, express lines 2 and 3.
2. Now assume that line 3 of the second table is extended with a second possible value 'outdoor', so that in this case, it has value 'indoor' or 'outdoor'. Both choices are acceptable. What changes? Adapt your representation.
3. Assume that for a given ship we know its expiration date and cargo, and furthermore, only that it belongs to the Panamax class of ships for which length, draft and capacity is known to range over given intervals $[l_1, l_2]$, $[d_1, d_2]$ and $[c_1, c_2]$. E.g., $l_1 = 190; l_2 = 250$. How to represent this information and use your theory and these data to decide whether and where to enter and refuel? Describe input and output of inference tasks in detail.

Solution

2.1 The vocabulary

The first step is to choose a vocabulary for modelling the knowledge and solving the various problems.

The general idea of KR is : every symbol should match a natural concept of the world that appears to be of relevance for the class of problems to be solved. So that every structure will represent a possible state of the world.

In the question two informations are given regarding how to choose a vocabulary. (1) The vocabulary and the theory should be such that a model of it represents a correct decision. As such, a model will specify the parameters of a ship, whether it may enter, and the location where to refuel. (2) It is said that *Enter* is a constant symbol; the attribute *RefuelArea* should be a unary predicatte.

```
type date
type RefArea constructed from {Indoors, Outdoors}
type YesNo constructed from {yes, no}

ExpDate:date
Length:Integer
Draft:Integer
Capacity:Integer
Cargo:Rational
Enter:YesNo
RefuelArea(RefuelArea)
today:date
```

Each structure of this vocabulary corresponds to a world restricted to one ship and expresses a decision for it.

The identity of the ship is abstracted away; there is no type `ship`. This is because all problems we consider are about one ship.

The value "none" for refuel area was not created. The reason is that none is a null value which in logic is more naturally and more safely handled by ... no value. Thus, if RefuelArea has empty value, it correspond to "none".

2.2 The theory

The first question is: do the tables specify constraints or definitions?

Recall that a definition of some concept in terms of parameter concepts specifies a unique value for the defined concept, given a values for the parameter concepts. Is this the case with these DMN tables?

Yes! The first one defines Enter in terms of the ship attributes. The second one defines RefuelArea in terms of Enter and two ship attributes.

To express definitions, we may use FO(.)-definitions.

```
{
Enter=no <- CerExp =< today.
Enter=yes <- CerExp > today & Length < 260 & Draft <  10 & Cpacity < 1000.
...
Enter=yes <- CerExp > today &  Length >= 320 & ..
...
}

{
RefuelArea(indoor) <-  Enter=yes &  Lenght =< 350.
RefuelArea(indoor) <-  Enter=yes &  Lenght  > 350 & Cargo =< 0.3.
RefuelArea(outdoor) <-  Enter=yes &  Lenght > 350 & Cargo > 0.3.
}
```

This is a standard way to express such definitions in FO(.).

Notice that the rules for RefuelArea correspond to the rows 2-4. There is NO rule corresponding to row 1. Why is that? Because a situation satisfying the first row, does not satisfy the conditions of the three other rules, hence, in that situation, RefuelArea is defined as empty. This corresponds to "none".

2.3 Offering another option for refueling

Suppose we extend row 3 with an additional possible value, outdoors. Note that in any decision, a unique refuel area should be assigned. We see that the assignment has become non-deterministic in this case. Hence, it is not a definition anymore.

One way to solve it is to make it deterministic by adding an auxilary predicate e.g., `ChooseIndoors`. Now we can write:

```
{
...
RefuelArea(indoor) <-  Enter=yes &  Lenght  > 350 & Cargo =< 0.3 & ChooseIndoors.
RefuelArea(outdoor) <-  Enter=yes &  Lenght  > 350 & Cargo =< 0.3 & ~ChooseIndoors.
..
}
```

Another way, and why not to choose it Another way that many students implemented was to add the following rule to the definiiton.

```
...
RefuelArea(outdoor) <- Enter=yes &  Lenght > 350 & Cargo =< 0.3.
..
```

Now, models for ships matching the third line are assigned two refuel areas. That is, in models where the ship parameters match the third line, the value of RefuelArea contains two refuel ares. What to think about this?

Well, in the first place, it violates the requirements of the question. The requirement was that each model should represent a correct decision. To assign two refuel areas is not a correct decision. A ship can be at only one place at the same time.

We could take another interpretation of the predicate. Rather than interpreting it as the place where a ship is refueled in a possible state of affairs, we coud interpret it as the set of all possible refueling places. Is there something wrong with that? Certainly!

We changed the meaning of the symbol RefuelArea/1. Where it used to be the case that RefuelArea(x) meant "the ship should refuel at x", it now means "the ship may refuel at x". That is not the same! In fact, no decision has been made: providing a range of options is not the same as making a decision, right?

In a larger application, changing the meaning of a predicate is dangerous. Existing formulas or queries or constraints or definitions elsewhere in the application that contain the symbol, are very likely wrong under the changed meaning. E.g., we have expressed that in a decision, one or no refuel location is assigned. $\#\{x : refuelArea(x)\} \leq 1$. This constraint does not hold if $RefuelArea$ represents the possible refuel areas.

2.4 Decision problem

To decide about the Panamax ship, we extend the theory T with all data we know about the ship including the constraint of Panamax ships:

```
Date=...
Cargo= ...
Lenth >= l1.
Lenght =< l2.
...
```

(We might add these data also to an IDP structure.)

Now we check the possible values for Enter and RefuelArea. If in all cases, the decision is the same, then that is our decision. If in different models the values are different, no decision can be taken, and more information about the ship is needed.

What kind of inference is needed:

- Run model expansion on $T' = T \cup \{Enter = yes\}$
  - Input: T' , $\mathfrak{A}_i$ (only types)
  - Output: unsat or a model $\mathfrak{A}$.
- Run model expansion on $T' = T \cup \{Enter = no\}$
- If both succeed, there is no decision.

  If one succeeds, that is the decision.

  If both fail, there is something wrong.
- Idem for $T \cup \{RefuelArea(indoor)\}$ and $T \cup \{\neg RefuelArea(indoor)\}$
- Idem for outdoor.

Alternatively (but perhaps less efficiently):

- Perform optimal propagation inference:
  - Input: $T', \mathfrak{A}$
  - Output: three-valued structure approximating all models.
- If Enter is propagated to “yes” or to “no”, then a decision is possible. Otherwise, if Enter remains unknown, then information is lacking to make a decision.
- Idem for RefuelArea.

3 Discussion

Many students represented each table by a predicate with an argument for each attribute of the table (ShipClearance/6, RefuelArea/4). Their aim was to build a theory with one model, in which the value of ShipClearance contains all assignments to attributes that satisfy the DMN table and likewise for RefuelArea.

This is a bad idea. First, it does not satisfy what was asked for: that models should represent a decision, that Enter should be a constant (with possible values yes or no), RefuelArea a unary predicate. Second, it is much more difficult to use this theory to solve the above problems.

E.g., write up the procedure to use logical inference in this representation to compute a decision for a ship with all attributes given. This is very difficult to do with this representation. This is something you may want to verify yourself.

Alternative theories The definition of `Enter` may be expressed as wel by a set of conditional equations.

```
CerExp =< today => Enter=no.
CerExp > today & Length < 260 & .... => Enter=yes.
..
CerExp > today & ... => Enter=yes.
...
```

This definition is "almost" equivalent to FO(.) definition, but it differs in worlds that do not match with any row of the array. In such cases, the definitions says `Enter` is undefined, the equational theory says nothing, and there will be different models where Enter had different values. This is something you may want to verify yourself.

In those cases that do not match any row (where Enter is unspecified), the safe approach is to have no value. It is unsafe to have an arbitrary value.

What if we use implications to express `RefuelArea`?

```
Enter=yes &  Lenght =< 350 => RefuelArea(indoor).
Enter=yes &  Lenght =< 350 => RefuelArea(indoor).
Enter=yes &  Lenght  > 350 & Cargo =< 0.3 => RefuelArea(indoor).
...
```

This is far too weak since it does not forbid that, e.g., ships matching the first case are refueled both indoors and outdoors. The following constraint has to be added:

```
~(RefuelArea(indoor) & RefuelArea(outdoor)).
```

Like for Enter, this representation is not equivalent with the definition in those cases that are not covered by a row of the array: with the definition, there is no refuel area; with the implications, the refuel area is different in different models. This is unsafe.

In case of multiple options for refuelng, it is possible to adapt the representation with constraints as follows:

```
...
Enter=yes &  Lenght  > 350 & Cargo =< 0.3 =>
                             RefuelArea(indoor) | RefuelArea(outdoor)
...
~(RefuelArea(indoor) & RefuelArea(outdoor)).
```

Here, the last constraint is again crucial.