safeAI | checking logical models

ANNELINE DAGGELINCKX, MATTHIJS KEMP, and OTTO MÄTTAS, Utrecht University, The Netherlands

ACM Reference Format:
Anneline Daggelinckx, Matthijs Kemp, and Otto Mättas. 2021. safeAI | checking logical models. 1, 1 (May 2021), 3 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 WEEK 1 ASSIGNMENTS

1.1 Defining reward functions

Below is a specification for a reward function in LTL for the following office world task: the agent is required to get to a state where coffee is true, and then back to the office, and after that maintain stop forever, all the while without stepping on decorations. Use coffee, office, stop and decs for propositions.

$$G(\neg decs) \wedge F(coffee \wedge F(office)) \wedge G(stop \leftrightarrow (coffee \wedge office)) \tag{1}$$

As to further the discussion, the following has been noticed. The premise dictates that the function will only run once until a state where coffee is true is achieved. On the other hand, personal intuition conflicts with the premise in the sense that this function could also be considered not to be terminated. This way, as soon as the state where coffee is true is lost, the function seeks actively to regain the state. In short, when coffee is not true, the function will activate. Right now, the premise does not allow for this.

1.2 Describing paths

Below is a path that satisfies $Gp \rightarrow Fq$ and does not satisfy $G(p \rightarrow Fq)$. Let λ be a path such that

$$\lambda[i...\infty] = s1s0^{\omega} = s1s0s0... \tag{2}$$

, where *s0 : p=True, q=False* and *s1 : p=False, q=True.*

1.3 Tracing formulas

Below is an example formula that is true on some finite trace and false on all infinite traces.

$$F(\neg X \top) \tag{3}$$

In English, this translates to there is some future where negation of a next turn exists or more plainly - next turn does not exist.

Authors' address: Anneline Daggelinckx, a.daggelinckx@students.uu.nl; Matthijs Kemp, m.g.r.kemp@students.uu.nl; Otto Mättas, o.mattas@students.uu.nl, Utrecht University, P.O. Box 80125, Utrecht, Utrecht, The Netherlands, 3508 TC.

XXXX-XXXX/2021/5-ART $15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

2 WEEK 2 ASSIGNMENTS

Following section holds assignments which have been carried out using the MCMAS software, version 1.3.0 [VAS Group 2021a]. MCMAS is an open-source, OBDD-based symbolic model checker tailored to the verification of Multi-Agent Systems (MAS). MAS descriptions are given by means of ISPL (Interpreted Systems Programming Language) programs. ISPL is an agent-based, modular language inspired by interpreted systems, a popular semantics in MAS.

2.1 Designing agents in ISPL

Below is a description for a vacuum cleaner agent in Interpreted Systems Programming Language (ISPL). The vacuum world domain (variables, actions, initial state, formulae) is given in the Coursework 1 document, assignment W2-1.

```
-- INFOMLSAI 2021 Vacuum Robot designed by
-- ANNELINE DAGGELINCKX, MATTHIJS KEMP and OTTO MATTAS

Agent Robot1
    Vars:
        inA: boolean;
        inB: boolean;
        cleanA: boolean;
        cleanB: boolean;
    end Vars
    Actions = {nil,move,suck};
    Protocol:
        inA=true or inB=true: {nil,move,suck};
    end Protocol
    Evolution:
        cleanA=true if inA=true and Action = suck;
        cleanA=true if cleanA=true and Action = nil;
        cleanB=true if inB=true and Action = suck;
        cleanB=true if cleanB=true and Action = nil;
        inA=true if inB=true and Action = move;
        inA=true if inA=true and Action = nil;
        inB=true if inA=true and Action = move;
        inB=true if inB=true and Action = nil;
    end Evolution
end Agent

Evaluation
        inA if Robot1.inA = true;
        inB if Robot1.inB = true;
        cleanA if Robot1.cleanA = true;
        cleanB if Robot1.cleanB = true;
end Evaluation

InitStates
        Robot1.inA = true and
        Robot1.inB = false and
        Robot1.cleanA = false and
        Robot1.cleanB = false;
end InitStates
```

```
Formulae
       (inA and !inB) and (!cleanA and !cleanB);
       inA -> EF inB;
       inA -> EF cleanA;
       EF (cleanA and cleanB);
       -- cleanB;
       -- AG inA;
       -- AF inB;
end Formulae
```

2.1.1 Proving formulae. The team has extracted a witness for the formula

$$EF(cleanA \wedge cleanB) \tag{4}$$

which indicates both rooms A and B being clean in some future state. Below, you can see the proof which has been generated via command-line interface (CLI) for MCMAS with the help of a **-c** flag as described by [VAS Group 2021b].

```
  Formula number 4: (EF (cleanA && cleanB)), is TRUE in the
       model
  The following is a witness for the formula:
   < 0 1 2 3 >
  States description:
------------- State: 0 -----------------
Agent Environment
Agent Cleaner
  cleanA = false
  cleanB = false
  location = inA
---------------------------------------
------------- State: 1 -----------------
Agent Environment
Agent Cleaner
  cleanA = true
  cleanB = false
  location = inA
---------------------------------------
------------- State: 2 -----------------
Agent Environment
Agent Cleaner
  cleanA = true
  cleanB = false
  location = inB
---------------------------------------
------------- State: 3 -----------------
Agent Environment
Agent Cleaner
  cleanA = true
  cleanB = true
  location = inB
---------------------------------------
```

2.1.2 Disproving the formulae. The team has extracted a counterexample for the formula

$$AGinA \tag{5}$$

which indicates all possible future paths lead to room A. Below, you can see the proof which has been generated via command-line interface (CLI) for MCMAS with the help of a **-c** flag as described by [VAS Group 2021b].

```
  Formula number 5: (AG inA), is FALSE in the model
  The following is a counterexample for the formula:
   < 0 1 >
  States description:
------------- State: 0 -----------------
Agent Environment
Agent Cleaner
  cleanA = false
  cleanB = false
  location = inA
---------------------------------------
------------- State: 1 -----------------
Agent Environment
Agent Cleaner
  cleanA = false
  cleanB = false
  location = inB
---------------------------------------
```

2.2 Modifying ISPL definitions

This section focuses on modifying the agent definitions in order to notice the evolution and changes in how the agent behaves. The updated robot carriage world domain details are given in the Coursework 1 document, assignment W2-2.

2.2.1 Translation from English to CTL. Firstly, a translation to Computation Tree Logic (CTL) is needed in order to describe the rules in ISPL.

English:

(1) it is possible to avoid pos2 forever
(2) it is possible to be in position pos2 in the next step
(3) it is possible in the future to be in position pos1 and in the next step after that in pos2
(4) it is possible to be in pos0 until reaching pos2
(5) it is possible to be in pos0 or pos1 until reaching pos2
(6) it is possible to always have pos1 reachable in at most 2 steps

CTL:

(1) EG !pos2
(2) EX pos2
(3) EX (pos1 and EX pos2)
(4) E (pos0 U pos2)
(5) E ((pos0 or pos1) U pos2)
(6) EG (EX (pos1) or EX (EX pos1))

2.2.2 Proofs. Secondly, the proofs are given through witnesses and counterexamples which have been generated via command-line interface (CLI) for MCMAS with the help of a **-c** flag as described by [VAS Group 2021b].

(1) EG !pos2

```
Formula number 4: (EG (! pos2)), is TRUE in the model
The following is a witness for the formula:
 < 0 0 >
```

```
  States description:
------------- State: 0 -----------------
Agent Environment
Agent Robot1
  pos = pos0
```

(2) EX pos2

```
  Formula number 5: (EX pos2), is FALSE in the model
  The following is a counterexample for the formula:
   < 0 >
  States description:
------------- State: 0 -----------------
Agent Environment
Agent Robot1
  pos = pos0
```

(3) EX (pos1 and EX pos2)

```
  Formula number 6: (EX (pos1 && (EX pos2))), is TRUE
       in the model
  The following is a witness for the formula:
   < 0 1 >
   < 1 2 >
  States description:
------------- State: 0 -----------------
Agent Environment
Agent Robot1
  pos = pos0
----------------------------------------
------------- State: 1 -----------------
Agent Environment
Agent Robot1
  pos = pos1
----------------------------------------
------------- State: 2 -----------------
Agent Environment
Agent Robot1
  pos = pos2
```

(4) E (pos0 U pos2)

```
  Formula number 7: E(pos0 U pos2), is FALSE in the
       model
  The following is a counterexample for the formula:
   < 0 >
  States description:
------------- State: 0 -----------------
Agent Environment
Agent Robot1
  pos = pos0
```

(5) E ((pos0 or pos1) U pos2)

```
  Formula number 8: E((pos0 || pos1) U pos2), is TRUE
       in the model
  The following is a witness for the formula:
   < 0 1 2 >
  States description:
------------- State: 0 -----------------
Agent Environment
Agent Robot1
  pos = pos0
----------------------------------------
------------- State: 1 -----------------
Agent Environment
Agent Robot1
  pos = pos1
----------------------------------------
------------- State: 2 -----------------
Agent Environment
Agent Robot1
  pos = pos2
```

(6) EG (EX (pos1) or EX (EX pos1))

```
  Formula number 9: (EG ((EX pos1) || (EX (EX
       pos1)))), is TRUE in the model
  The following is a witness for the formula:
   < 0 0 >
   < 0 1 >
   < 1 2 >
  States description:
------------- State: 0 -----------------
Agent Environment
Agent Robot1
  pos = pos0
----------------------------------------
------------- State: 1 -----------------
Agent Environment
Agent Robot1
  pos = pos0
----------------------------------------
------------- State: 2 -----------------
Agent Environment
Agent Robot1
  pos = pos1
```

2.3 Describing cases for algorithms

Below is a case for a model checking algorithm for CTL. Details about the modality are given in the Coursework 1 document, assignment W2-3.

$$E((\phi U \psi) and \neg \psi) \tag{6}$$

In English, this translates to there is an individual where ϕ until ψ holds and ψ does not hold as to enable ϕ.

REFERENCES

Imperial College London VAS Group. 2021a. *MCMAS.* Retrieved May 6, 2021 from https://vas.doc.ic.ac.uk/software/mcmas/

Imperial College London VAS Group. 2021b. *MCMAS v1.2.2: User Manual.* Retrieved May 6, 2021 from https://vas.doc.ic.ac.uk/software/mcmas/manual.pdf