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), 4 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 WEEK 3 ASSIGNMENTS

1.1 Describing Kripke models

Below, a description for a distributed system is given in a Kripke model. It is a structure consisting of a certain set of ordinary models for classical logic, ordered by a certain relation, and serving for the interpretation of various non-classical logics (intuitionistic, modal, etc.)

1.1.1 States. First, to define model M_{abc}, all the possible states need to be described. Let the states be St_{abc} such that $St_{abc} = \{s0, s1, ..., s5\}$, where

- $s0 = \{a1, b2, c3\}$;
- $s1 = \{a1, b3, c2\}$;
- $s2 = \{a2, b1, c3\}$;
- $s3 = \{a2, b3, c1\}$;
- $s4 = \{a3, b1, c2\}$;
- $s5 = \{a3, b2, c1\}$.

1.1.2 Indistinguishability relations. The states with indistinguishable knowledge for each agent $A = \{a, b, c\}$ have been described

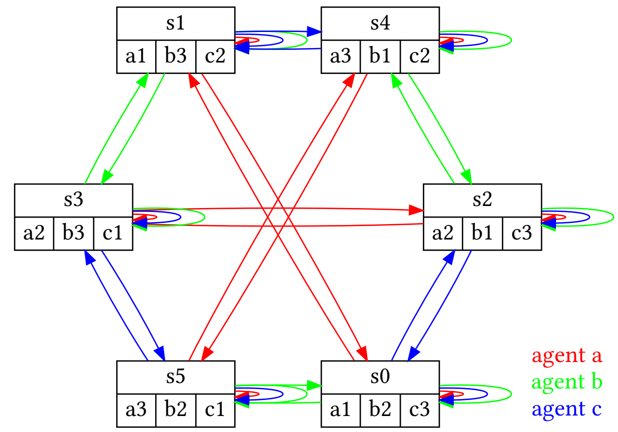
- $a = \{s0, s1\}, \{s2, s3\}, \{s4, s5\}$;
- $b = \{s2, s4\}, \{s0, s1\}, \{s1, s5\}$;
- $c = \{s3, s5\}, \{s1, s4\}, \{s0, s2\}$.

1.1.3 Valuation. Following the example given in course, the valuations can be described as follows. Let the valuations be the form of $x_y \mapsto \{S_a\}$, where

- $a1 \mapsto \{s0, s1\}$;
- $a2 \mapsto \{s2, s3\}$;
- $a3 \mapsto \{s4, s5\}$;
- $b1 \mapsto \{s2, s4\}$;
- $b2 \mapsto \{s0, s5\}$;
- $b3 \mapsto \{s1, s3\}$;
- $c1 \mapsto \{s3, s5\}$;
- $c2 \mapsto \{s1, s4\}$;
- $c3 \mapsto \{s0, s2\}$.

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



1.2 Proving Kripke models using ELCD

Below, there are three statements to help prove whether the model is properly constructed:

(1) it is distributed knowledge between a, b and c that $a1 \wedge b2 \wedge c3$;
(2) it is common knowledge between a, b and c that $a1 \vee b2 \vee c3$;
(3) it is common knowledge between a, b and c that $a1 \vee a2 \vee a3$.

1.2.1 Translation. First, the statements need to be translated from English to the notation form of epistemic logic with common and distributed knowledge (ELCD):

(1) $D_{a,b,c}(a1 \wedge b2 \wedge c3)$;
(2) $C_{a,b,c}(a1 \vee b2 \vee c3)$;
(3) $C_{a,b,c}(a1 \vee a2 \vee a3)$.

1.2.2 Argumentation. Then, it is possible to determine whether the statement holds (is true) or not:

(1) This statement is true. Each agent knows it's own card. Agent a knows $a1$, agent b knows $b2$ and agent c knows $c3$. Together, the agents would know which agent holds which card. Formally: $\sim^D_{a,b,c} = \bigcap_{i \in \{a,b,c\}} \sim_i$ which only keeps the equivalence arrows from a state to itself. Therefore, $M, s_0 \models D_{a,b,c}(a1 \wedge b2 \wedge c3)$
(2) This statement is false. Agent c knows that b holds card 1 or 2. It can reason that if b has card 2, agent b knows that either situation $a2b1c3$ or $a3b1c2$ holds. In the latter, $(a1 \vee b2 \vee c3)$ would be false. Therefore, agent c doesn't know that agent b knows $(a1 \vee b2 \vee c3)$ and it is not common knowledge. Formally: $\sim^E_{a,b,c} = \bigcup_{i \in \{a,b,c\}} \sim_i$. For this relation, every state is in the same equivalence class. The transitive closure and therefore $\sim^C_{a,b,c}$, contains the same relations (all states "connected"). There are states where $(a1 \vee b2 \vee c3)$ is not true, for example s3. Therefore $C_{a,b,c}(a1 \vee b2 \vee c3)$ is not true in the model.

(3) This statement is true. The agents know that there are only three cards and that every agent has one card. Therefore, they know that agent a has to have a card with a single value $\{1; 2; 3\}$. Since this is defined by the games rules, the agents know that the others also know this. Therefore it is common knowledge.
Formally: Just like above, for the relation $\sim^{C}_{a,b,c}$ every state is in relation with every other state. $a1 \vee a2 \vee a3$ is true in every state, thus also in every state that is in relation with s_0. Therefore, it is indeed common knowledge.

1.3 Investigating truthfulness in Kripke models

As multiple agents are trying to interact with each other and the environment, there is a way to introduce novel information to all agents through public announcements.

In our assignment, there is a public announcement $\varphi = a1$, that allows agents b and c to know all the cards of all the agents while allowing agent a to only know about itself and the card it is holding.

1.3.1 States. The model $M^{a1}_{a,b,c}$ is then reduced to states as follows $St = \{q_{123}, q_{132}\}$.

1.3.2 Indistinguishability relations. The states with indistinguishable knowledge for each agent $A = \{a, b, c\}$ have been described as

- $a = \{s0, s1\}$;
- $b = \{s0\}, \{s1\}$;
- $c = \{s0\}, \{s1\}$.

As to extend - agents b and c can distinguish between all states, but cannot distinguish states from themselves.

1.3.3 Valuation. In the updated model, we also only need a subset of valuations. Namely those corresponding with impossible worlds can be discarded. All initial worlds can be seen in 1.1.3. Valuations that are left out, map to empty sets and are always false. Let the valuations be the form of $x_y \mapsto \{S_a\}$, where

- $a1 \mapsto \{s0, s1\}$;
- $b2 \mapsto \{s0, s5\}$;
- $b3 \mapsto \{s1, s3\}$;
- $c2 \mapsto \{s1, s4\}$;
- $c3 \mapsto \{s0, s2\}$.

1.4 Common knowledge

Does φ also guarantee common knowledge $C_{b,c}(a1 \wedge b2 \wedge c3)$? Yes, b knows that c also got the message and the other way around. Therefore, they know that the other knows the card of a, and their own card. The card of the third agent can be deducted. Therefore it is common knowledge between b and c.

Formally, $\sim^{E}_{\{b,c\}}$ has equivalence classes $\{q_{123}\}$ and $\{q_{132}\}$. $\sim^{C}_{\{b,c\}}$ is the transitive closure, which stays the same. Therefore we have that $M^{a1}_{a,b,c}, q_{123} \models C_{\{b,c\}} a_1 \wedge b2 \wedge c3$.

2 WEEK 4 ASSIGNMENTS

2.1 Designing interpreted systems in ISPL

Following section holds assignments which have been carried out using the MCMAS software, version 1.3.0 [VAS Group 2021]. 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.

A description for an interpreted system with three agents in Interpreted Systems Programming Language (ISPL) has been added as an appendix A. The world domain (variables, actions, initial state, formulae) is given in the Coursework 2 document, assignment W4-1.

2.1.1 Evaluating formulae. The team has checked given formulae and verified the properties by doing so. Below is the summary output from MCMAS.

```
% ./mcmas push_carriage.ispl
************************************************************************
                    MCMAS v1.3.0

 This software comes with ABSOLUTELY NO WARRANTY, to the
      extent
 permitted by applicable law.

 Please check http://vas.doc.ic.ac.uk/tools/mcmas/ for the
      latest release.
 Please send any feedback to <mcmas@imperial.ac.uk>
************************************************************************

Command line: ./mcmas push_carriage.ispl

push_carriage.ispl has been parsed successfully.
Global syntax checking...
1
1
1
Done
Encoding BDD parameters...
Building partial transition relation...
Building BDD for initial states...
Building reachable state space...
Checking formulae...
Verifying properties...
  Formula number 1: (pos2 -> (! K(Robot1, pos2))), is TRUE
      in the model
  Formula number 2: (pos2 -> K(Robot1, (! pos1))), is TRUE
      in the model
  Formula number 3: (pos2 -> K(Robot2, K(Robot1, (!
      pos1)))), is TRUE in the model
  Formula number 4: (pos2 -> DK(g, pos2)), is TRUE in the
      model
  Formula number 5: (pos2 -> (! GK(g, pos2))), is TRUE in
      the model
  Formula number 6: (pos2 -> GK(g, (! pos1))), is TRUE in
      the model
  Formula number 7: (pos2 -> (! GCK(g, pos2))), is TRUE in
      the model
  Formula number 8: (pos2 -> (! GCK(g, (! pos1)))), is TRUE
      in the model
  Formula number 9: (AG GCK(g, ((pos0 || pos1) || pos2))),
      is TRUE in the model
```

```
  Formula number 10: (EF GK(g, (! pos1))), is TRUE in the
      model
done, 10 formulae successfully read and checked
execution time = 0.029
number of reachable states = 3
BDD memory in use = 8981056
```

2.2 Designing model checking algorithms

We rewrite the universal properties in terms of properties that can be checked using the existential pre-image. This way, we don't need the universal pre-image anymore.

$$AX\psi = \neg EX\neg\psi \tag{1}$$

$$AG\psi = \neg EF\neg\psi = \neg E(\top U(\neg\psi)) \tag{2}$$

$$\begin{aligned} A\psi_1 U\psi_2 &= \neg E\big((\neg\psi_2)U(\neg\psi_1 \wedge \neg\psi_2)\big) \wedge \neg EG(\neg\psi_2) \\ &= \neg\Big(E\Big((\neg\psi_2)U\neg(\psi_1 \vee \psi_2)\Big) \vee EG(\neg\psi_2)\Big) \end{aligned} \tag{3}$$

REFERENCES

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

A INTERPRETED SYSTEM IN ISPL

```
-- INFOMLSAI 2021 an interpreted system designed by
-- ANNELINE DAGGELINCKX, MATTHIJS KEMP and OTTO MTTAS

Agent Environment
   Vars:
      pos: {pos0,pos1,pos2};
      colour: {blue,white};
      texture: {smooth,rough};
   end Vars
   Actions = {ND};
   Protocol:
      Other : {ND};
    end Protocol
   Evolution:
      -- No action
      pos=pos0 and colour=blue and texture=rough if pos=pos0
           and Robot1.Action=nil and Robot2.Action=nil;
      pos=pos1 and colour=white and texture=rough if
           pos=pos1 and Robot1.Action=nil and
           Robot2.Action=nil;
      pos=pos2 and colour=blue and texture=smooth if
           pos=pos2 and Robot1.Action=nil and
           Robot2.Action=nil;
      -- Robot1 push
        pos=pos0 and colour=blue and texture=rough if
             pos=pos2 and Robot1.Action=push1 and
             Robot2.Action=nil;
        pos=pos1 and colour=white and texture=rough if
             pos=pos0 and Robot1.Action=push1 and
             Robot2.Action=nil;
        pos=pos2 and colour=blue and texture=smooth if
             pos=pos1 and Robot1.Action=push1 and
             Robot2.Action=nil;
        -- Robot2 push
        pos=pos0 and colour=blue and texture=rough if
             pos=pos1 and Robot1.Action=nil and
             Robot2.Action=push2;
        pos=pos1 and colour=white and texture=rough if
             pos=pos2 and Robot1.Action=nil and
             Robot2.Action=push2;
        pos=pos2 and colour=blue and texture=smooth if
             pos=pos0 and Robot1.Action=nil and
             Robot2.Action=push2;
        -- Robot1 and Robot2 push
        pos=pos0 and colour=blue and texture=rough if
             pos=pos0 and Robot1.Action=push1 and
             Robot2.Action=push2;
      pos=pos1 and colour=white and texture=rough if
           pos=pos1 and Robot1.Action=push1 and
           Robot2.Action=push2;
      pos=pos2 and colour=blue and texture=smooth if
           pos=pos2 and Robot1.Action=push1 and
           Robot2.Action=push2;
   end Evolution
end Agent

Agent Robot1
   Lobsvars = {colour};
    -- vars required by ISPL, give empty placeholder
   Vars:
      placeholder : {ND};
   end Vars
    Actions = {push1,nil};
    Protocol:
        Other : {push1,nil};
    end Protocol
    -- evolution required by ISPL, give empty placeholder
    Evolution:
        placeholder=ND if placeholder=ND;
    end Evolution
end Agent

Agent Robot2
   Lobsvars = {texture};
    -- vars required by ISPL, give empty placeholder
   Vars:
      placeholder : {ND};
   end Vars
    Actions = {push2, nil};
    Protocol:
        Other : {push2, nil};
    end Protocol
    -- evolution required by ISPL, give empty placeholder
    Evolution:
        placeholder=ND if placeholder=ND;
    end Evolution
end Agent

Evaluation
    pos0 if Environment.pos=pos0;
    pos1 if Environment.pos=pos1;
    pos2 if Environment.pos=pos2;
    blue if Environment.colour=blue;
    white if Environment.colour=white;
    rough if Environment.texture=rough;
    smooth if Environment.texture=smooth;
end Evaluation

InitStates
   Environment.pos=pos0 and
   Environment.colour=blue and
   Environment.texture=rough;
end InitStates

Groups
   g = {Robot1,Robot2};
end Groups

Formulae
   pos2 -> !K(Robot1,pos2);
   pos2 -> K(Robot1,!pos1);
   pos2 -> K(Robot2,K(Robot1,!pos1));
   pos2 -> DK(g,pos2);
   pos2 -> !GK(g,pos2);
   pos2 -> GK(g,!pos1);
   pos2 -> !GCK(g,pos2);
   pos2 -> !GCK(g,!pos1);
   AG GCK(g,(pos0 or pos1 or pos2));
   EF GK(g,!pos1);
end Formulae
```