Better State Pictures Facilitating State Machine Characteristic Conjecture

Dang Duy Bui, Kazuhiro Ogata
School of Information Science, JAIST
1-1 Asahidai, Nomi, Ishikawa 923-1292, Japan
Email: {bddang,ogata}@jaist.ac.jp

Abstract—MCS shared-memory mutual exclusion protocol is used as an example to demonstrate that state picture designs should be better visualized. A case study has been conducted in which we demonstrate that better visualized state pictures make it possible to conjecture more non-trivial characteristics of state machines than the old state pictures. The lessons learned acquired through the case study are summarized as new tips on how to make state picture designs.

Keywords—graphical animation; MCS protocol; SMGA; state machine; state picture design

I. INTRODUCTION

Our group has been developing a state machine graphical animation (SMGA) tool [1], [2]. Given a state picture design for a state machine, SMGA basically takes a sequence of states in text and plays a graphical animation for the state machine by regarding the sequence as a movie film based on the state picture design. SMGA has been used to make graphical animations of state machines formalizing Alternating Bit Protocol (ABP) [1], MCS shared-memory mutual exclusion protocol [3] and Suzuki-Kasami distributed mutual exclusion protocol [2]. One of the main goals to be achieved with SMGA is for human users to conjecture state machine characteristics or properties by visually/graphically observing graphical animations. This is because humans are good at visual perception [4].

We summarize some tips on how to design state pictures in our paper [2] published in 2019. The tips have been accumulated through the case studies for ABP [1], MCS protocol [3] and Suzuki-Kasami protocol [2]. They are still useful but we realized that there is some room to improve some state picture designs used for SMGA. Some part of each state of a state machine formalizing MCS protocol was not visualized sufficiently enough in that the part is almost the same as text representation. We have visualized the part, conducting a case study in which we have conjectured several non-trivial characteristics of (the state machine formalizing) MCS protocol. In this paper, we propose some more tips on how to make better state picture designs.

ShiViz [5] is a tool to visualize logs generated by distributed systems. The visualization used by ShiViz (still visualization) and the one (graphical animation) used by

SMGA can be complementary. Magee, et al. [6] have proposed a way to visualize the behavior of a Labeled Transition System (LTS) described in FSP and developed a tool to support their proposed technique. One novelty of their approach to graphical animation of the LTS behavior is to use Timed Automata as formal semantics of animations. Their tool has been implemented with SceneBeans a library of JavaBeans. Their visualization is graphical animation like ours. SceneBeans could be used to implement a future version of SMGA.

We suppose that readers are familiar with state machines and Maude [7] to some extent.

MCS protocol [8] is a shared-memory mutual exclusion protocol invented by Mellor-Crummey and Scott:

```
rs : "Remainder Section"
l1 : nextp := nop;
l2 : predp := fetch&store(glock, p);
l3 : if predp ≠ nop {
l4 :   lockp := true;
l5 :   nextpredp := p;
l6 :   repeat while lockp; }
cs : "Critical Section"
l7 : if nextp = nop {
l8 :   if comp&swap(glock, p, nop)
l9 :   goto rs;
l10 : repeat while nextp = nop; }
l11 : locknextp := false;
l12 : goto rs;
```

It uses one global variable *glock* and three local variables *next_p*, *pred_p* and *lock_p* for each process *p*. The protocol uses two atomic operations (or instructions): *fetch&store* and *comp&swap*.

II. SPECIFICATION OF MCS PROTOCOL IN MAUDE

We suppose that there are three processes that participate in MCS protocol. Let $\mathcal{M}_{\text{MCS}}$ formalize MCS protocol. Each state in $\mathcal{S}_{\text{MCS}}$ is expressed as follows:

```
{ (glock : bp)
  (pc[p1] : l1) (pred[p1] : pp1)
  (lock[p1] : b1) (next[p1] : np1)
  (pc[p2] : l2) (pred[p2] : pp2)
  (lock[p2] : b2) (next[p2] : np2)
  (pc[p3] : l3) (pred[p3] : pp3)
  (lock[p3] : b3) (next[p3] : np3) }
```

where *bp*, *pp_i* and *np_i* for *i* = 1, 2, 3 are process IDs, *l_i* for *i* = 1, 2, 3 is a label, such as rs, l1 and cs, and *b_i* for

Figure 1. A state picture design for MCS protocol (1)

Figure 2. A state picture for MCS protocol (1)

$i = 1, 2, 3$ is a Boolean value. Initially, bp , pp_i and np_i are nop , l_i is rs and b_i is false . $\mathcal{I}_{\text{MCS}}$ consists of one state. Let init equal the initial state.

$\mathcal{T}_{\text{MCS}}$ is described in terms of Maude rewrite rules. Among them are as follows:

```

rl [stprd] : {(glock: Q) (pc[P]: 12)
(pred[P]: Q1) OCs}
=> {(glock: P) (pc[P]: 13)
(pred[P]: Q) OCs} .
rl [chprd] : {(pc[P]: 13) (pred[P]: Q) OCs}
=> {(pc[P]: (if Q == nop then cs else l4 fi))
(pred[P]: Q) OCs} .
rl [stlok] : {(pc[P]: 14) (lock[P]: B) OCs}
=> {(pc[P]: 15) (lock[P]: true) OCs} .

```

where OCs is a Maude variable of observable component soups, P , Q and $Q1$ are Maude variables of process IDs, and B is a Maude variable of Boolean values. $\text{if } b \text{ then } x \text{ else } y \text{ fi}$ equals x if b equals true and y if b equals false .

III. STATE PICTURE DESIGN

States are expressed as braced soups of observable components, where observable components are name-value pairs. Therefore, it would be possible to automatically produce a straightforward state picture design. However, state pictures generated from the state picture design are almost the same as states in text. We summarize some tips on how to design state pictures for mutual exclusion protocols in our paper [2] published in 2019. Among the tips are as follows:

Figure 3. A state picture design for MCS protocol (2)

Figure 4. A state picture for MCS protocol (2)

- To recognize what sections there are at which each process or node is located, allocate the pane (or place) for each section such that the relations among the sections are visually perceived and display some diagram, such as a circle on which a process or node ID is written, on the designated pane;
- To recognize what pieces of information, such as the network for the Suzuki-Kasami protocol and variable locked for the test&set protocol, are shared by all processes or nodes, allocate the pane (or place) for each such piece of information such that we can visually perceive they are shared by all processes and nodes and display them on the designated panes adequately;

Nguyen and Ogata [3] made the state picture design shown in Fig. 1 for MCS protocol, which follows the tips. A state picture generated from the state picture design is shown in Fig. 2. The state picture allows us to immediately realize that processes $p1$, $p2$ and $p3$ are located at $l5$, $l6$ and $l8$, respectively. Nguyen and Ogata [3] conducted a case study in which several characteristics of MCS protocol can be discovered by observing graphical animations of MCS protocol such that each state picture used in the graphical animation is generated from the state picture design. For example, one of the characteristics found is as follows:

No state such that a process is at cs , $l7$, $l8$, $l10$, or $l11$ and another process is at cs , $l7$, $l8$, $l10$, or $l11$.

Let the characteristic be called Characteristic 0 in this paper.

Although the state picture shown in Fig. 2 also allows us to

notice the values stored in the global variable *glock* and the three local variables *next_p*, *pred_p* and *lock_p* for each process *p*, their representations on the state picture are almost the same as the text representation. Since SMGA requires and/or permits human users to make state picture designs, the representations must be able to be visually/graphically perceivable. To this end, we came up with the state picture design shown in Fig. 3. Fig. 4 shows a state picture generated from the state picture design.

The design of the *glock* representation used in Fig. 1 is as follows:

The value of *glock* is nop, p1, p2 or p3. Regardless of the value, the value is displayed on the same place. For example, when the value is p1, it is displayed as follows:

The design of the *glock* representation used in Fig. 3 is as follows:

If the value is nop, nothing is displayed on the rectangle or pane for *glock*. If the value is p1, p1 is displayed at the left-most place of the rectangle for *glock*. If the value is p2, p2 is displayed at the middle place of the rectangle for *glock*. If the value is p3, p3 is displayed at the right-most place of the rectangle for *glock*. For example, when the value is p1, it is displayed as follows:

We can say that the *glock* representation used in Fig. 3 and Fig. 4 helps human users more visually/graphically perceive its value than the one used in Fig. 1 and Fig. 2.

The design of the *next_p*, *pred_p* and *lock_p* representations for each process *p* used in Fig. 1 is as follows:

<i>next</i> [p1] : nop	<i>next</i> [p2] : nop	<i>next</i> [p3] : nop
<i>pred</i> [p1] : nop	<i>pred</i> [p2] : nop	<i>pred</i> [p3] : nop
<i>lock</i> [p1] : false	<i>lock</i> [p2] : false	<i>lock</i> [p3] : false

Regardless of the values of *next_p*, *pred_p* and *lock_p*, their values are displayed on the same places. For example, when *next_{p1}* is nop, *pred_{p1}* is p2, *lock_{p1}* is true, *next_{p2}* is nop, *pred_{p2}* is p3, *lock_{p2}* is true, *next_{p3}* is p2, *pred_{p3}* is nop and *lock_{p3}* is false, those values are displayed as follows:

<i>next</i> [p1] : nop	<i>next</i> [p2] : nop	<i>next</i> [p3] : p2
<i>pred</i> [p1] : p2	<i>pred</i> [p2] : p3	<i>pred</i> [p3] : nop
<i>lock</i> [p1] : true	<i>lock</i> [p2] : true	<i>lock</i> [p3] : false

The design of the *next_p*, *pred_p* and *lock_p* representations for each process *p* used in Fig. 3 is as follows:

The *next_p* representation appears at the right-most place, where there are three rectangles, the first, second and third ones of which from top are used for p1, p2 and p3, respectively. For example, if the value of *next_{p1}* is nop, nothing is shown on the first rectangle; if the value is *pi*, the circle on which *pi* is written is shown at the designated place on the first rectangle. The *lock_p* representation appears at the middle place, which also indicates that the first, second and third rows are used for p1, p2 and p3, respectively. When *lock_{pi}* is true, the background color of *pi* is red; otherwise the color is non-red (or light blue). The *pred_p* representation appears at the left-most place, where there are three rectangles, the first, second and third ones of which from top are used for p1, p2 and p3, respectively. For example, if the value of *pred_{p1}* is nop, nothing is shown on the first rectangle; if the value is *pi*, the circle on which *pi* is written is shown at the designated place on the first rectangle. For instance, when *next_{p1}* is nop, *pred_{p1}* is p2, *lock_{p1}* is true, *next_{p2}* is nop, *pred_{p2}* is p3, *lock_{p2}* is true, *next_{p3}* is p2, *pred_{p3}* is nop and *lock_{p3}* is false, those values are displayed as follows:

We can say that the *next_p*, *pred_p* and *lock_p* representations for each process *p* used in Fig. 3 and Fig. 4 help human users more visually/graphically perceive their values than the ones used in Fig. 1 and Fig. 2.

We advocate that state picture designs should be visualized as much as possible so that state pictures generated from the designs can help human users visually/graphically perceive the value of each observable component and some relations among multiple observable component values. In what follows, we demonstrate what characteristics of MCS protocol can be perceived when we use the state picture design shown in Fig. 3.

IV. CHARACTERISTIC DISCOVERY BASED ON NEW STATE PICTURES

Carefully observing graphical animations for MCS protocol in which new state pictures, such as Fig. 2, were used, we realized that there is always at most one process at cs, l7, l8, l10 and l11. This is exactly the same as Characteristic 0 discovered based on old state pictures, such as Fig. 4. We call these sections (cs, l7, l8, l10 and l11) CS region. We also noticed that there exists at most one process *p* such that *p* is located at l3 and *pred_p* is nop and there exists at most

Figure 5. Some pictures for extended CS region

one process p such that p is located at $l6$ and $lock_p$ is false. Moreover, if p is located at $l3$ and $pred_p$ is nop, there is no process in CS region and there is no process q at $l6$ such that $lock_q$ is false, and if p is located at $l6$ and $lock_p$ is false, there is no process in CS region and there is no q at $l3$ such that $pred_q$ is nop. $l3$ and $l6$ are only the sections from which processes enter CS region. We call CS region plus $l3$ and $l6$ extended CS region. The first characteristic that can be conjectured by carefully observing graphical animations for MCS protocol in which new state pictures, such as Fig. 2, are used is as follows:

- Characteristic 1: There exists at most one process except for processes p such that (1) p is located at $l3$ and $pred_p$ is not nop and (2) p is located at $l6$ and $lock_p$ is not false in extended CS region.

Extended CS region is one key concept that captures one important aspect of MCS protocol and Characteristic 1 is very crucial in that several other characteristics can be discovered based on the characteristic. Fig. 5 shows some state pictures that capture Characteristic 1. On the top left state picture, there are two processes $p1$ and $p2$ in extended CS region such that both are located at $l6$, $lock_{p1}$ is true and $lock_{p2}$ is false and therefore $p2$ is only the process in extended CS region in the sense of Characteristic 1 because

$p1$ satisfies condition (2) in Characteristic 1. On the top right state picture, there are two processes $p1$ and $p3$ in extended CS region such that both are located at $l3$, $pred_{p1}$ is $p2$ and $pred_{p3}$ is nop and therefore $p3$ is only the process in extended CS region in the sense of Characteristic 1 because $p1$ satisfies condition (1) in Characteristic 1. Each of the other five state pictures shows that there exists one process at one section of CS region and for all processes q located at $l3$ and $l6$ if any, $pred_q$ is not nop and $lock_q$ is not false, respectively.

Focusing on a process in CS region or extended CS region, we can recognize the following characteristics:

- Characteristic 2.1: Whenever there is a process at $l10$, there is at least one process at $l3$, $l4$, $l5$ or $l6$;
- Characteristic 2.2: Whenever there is a process at $l11$, there is at least one process at $l6$.

The bottom left state picture of Fig. 5 is an example of Characteristic 2.1 and the bottom right state picture of Fig. 5 is an example of Characteristic 2.2.

Carefully observing the $next_p$ and $pred_p$ representations for each process p , nothing may be displayed and the circle on which q that is different from p is written may be displayed but the circle on which p is written is never displayed. Thus, we can realize the following characteristics:

- Characteristic 3.1: The value of $next_p$ for each process p is never p .
- Characteristic 3.2: The value of $pred_p$ for each process p is never p .

For example, taking a look at the $next_{p1}$ representation or rectangle, it is visually/graphically observable that the circle on which $p1$ is written never comes into sight on the designated position. This is because $next_{p1}$ is visually/graphically represented in the new state picture design.

Some relations between two observable components can be discovered by carefully observing graphical animations, from which some characteristics can be conjectured. A relation between the $glock$ observable component and the $pc[p]$ observable component can be perceived by graphical animations and allows us to conjecture the following characteristics:

- Characteristic 4.1: If $glock$ is nop, then there is no process at l3, l4, l5 or l6 or in CS region;
- Characteristic 4.2: If $glock$ is a process p , then p is located at l3, l4, l5 or l6 or in CS region;
- Characteristic 4.3: If $glock$ is not nop (or equivalently a process), then there exists a process in CS region.

A relation between the $pred[p]$ observable component and the $pc[p]$ observable component allows us to conjecture the following characteristic:

- Characteristic 5: If $pred_p$ for each process p is nop, then p is never located at l4, l5, l6 or l12.

A relation between the $pc[p]$ observable component and the $next[p]$ observable component allows us to conjecture the following characteristics:

- Characteristic 6.1: If each process p is located at l2 or l9, then $next_p$ is nop;
- Characteristic 6.2: If $next_p$ for each process p is nop, then p is not located at l11 or l12.

A relation between the $lock[p]$ observable component and the $pc[p]$ observable component allows us to conjecture the following characteristics:

- Characteristic 7.1: If $lock_p$ for each process p is true, then p is located at l5 or l6;
- Characteristic 7.2: If each process p is located at l5, then $lock_p$ is true.

Note that when a process p is located at l6, another process may set $lock_p$ to false.

It is necessary to fix the values of some observable components so as to discover some similarities of multiple states and/or some relations among observable components. It is not straightforward to do so by observing graphical animations because we need to remember states in which the former observable components have the fixed values. For example, it must not be reasonable to remember all states in a graphical animations such that $pred_{p1}$ is $p2$ and $p2$ is located at rs, l1, l2, l9 or l12. One possible remedy for it is

Figure 6. Three state pictures discovered by Find Pattern functionality.

to use the Find Pattern functionality of SMGA [1], [3], [2]. Given a regular expression written by human users, the Find Pattern functionality finds all states in an input sequence of states such that they match the regular expression. When we would like to find all states in an input sequence of states such that $pred_{p1}$ is $p2$ and $p2$ is located at rs, l1, l2, l9 or l12, it suffices to write the following regular expression:

```
state['pred[p1]'] == 'p2' &&
(state['pc[p2]'] == 'l1' ||
 state['pc[p2]'] == 'l2' ||
 state['pc[p2]'] == 'rs' ||
 state['pc[p2]'] == 'l9' ||
 state['pc[p2]'] == 'l12')
```

`state['pred[p1]'] == 'p2'` says that $pred_{p1}$ equals $p2$. `&&` and `||` are logical conjunction and disjunction, respectively. Fig.6 shows three state pictures among the states discovered by the Find Pattern functionality for the regular expression.

It does not suffice, however, to use the regular expression abovementioned so as to conjecture non-trivial characteristics. This is because the pair $(p1, p2)$ is one possible combination and there are five more combinations to consider: $(p1, p3)$, $(p2, p1)$, $(p2, p3)$, $(p3, p1)$ and $(p3, p2)$. Note that we do not need to take the three combinations $(p1, p1)$, $(p2, p2)$ and $(p3, p3)$ into account because of

Characteristic 3.1. Let (p, q) be each of the six combinations to consider. Carefully observing all states discovered by the Find Pattern functionality for the six combinations to consider, we can conjecture some non-trivial characteristics:

- Characteristic 8.1: If $pred_p$ is q and q is located at rs, l1, l2, l9, l11 or l12, then p is not located at l3, l4 or l5;
- Characteristic 8.2: If $pred_p$ is q and $glock$ is q , then p is not located at l3, l4 or l5;
- Characteristic 8.3: If $pred_p$ is q , $next_q$ is not nop and p is located at l3, l4, l5, l6, cs, l7, l8 or l10, then q is not located at l3, l4 or l5.

We noticed that there are both states in which $next_q$ is nop and those in which $next_q$ is not nop among the states found by the Find Pattern functionality for the regular expression of the condition of Characteristic 8.1, while there are only states in which $next_q$ is nop among the states found by the Find Pattern functionality for the regular expression of the condition of Characteristic 8.2. Then, finding the states with the Find Pattern functionality for the regular expression of the second sub-condition only of Characteristic 8.2 because the first sub-condition is shared by both characteristics, we realized that $next_q$ is nop in all of them. Therefore, we came up with the following characteristic:

- Characteristic 9: If $glock$ is a process p (or equivalently non-nop), $next_p$ is nop.

We have used the Maude reachability analyzer (or the search command) to model check that all characteristics conjectured in this section are invariant properties with respect to the state machine formalizing MCS protocol and have not found any counterexamples for each characteristic.

V. SOME LESSONS LEARNED

Let us summarize some lessons learned through the case study with SMGA and MCS protocol. We have already described one important lesson learned in Sect. III. Let us repeat it:

- State picture designs should be visualized as much as possible so that state pictures generated from the designs can help human users visually/graphically perceive the value of each observable component and some relations among multiple observable component values.

We have demonstrated that the new state picture design (more visualized) allows us to more visually/graphically perceive the values of $glock$ and $next_p$, $pred_p$ & $lock_p$ for each process p than the old state picture design (less visualized) and then conjecture more non-trivial characteristics of MCS protocol than the old ones in Sect. IV.

To make the lesson more practical, we describe some more concrete lessons or guides.

- When an observable component can have two different values, such as $lock_p$ for each process p , it should be visually/graphically represented as a light bulb.

For example, if an observable component has one value, such as on, we should use a fancy or light color; if it is the other value, such as off, we should use a plain or dark color.

- When an observable component can have three or more (but moderate) different values, such as $glock$ and $next_p$ & $pred_p$ for each process p , we should prepare some designated area, such as a rectangle, and a specific position in the area for each value where some visual object, such as a circle on which the value is written, is displayed; if the observable component has a value, only the visual object for it should be displayed and the other visual objects for the other values should disappear; there may be some special value, such as nop, and if the observable component has such a value, nothing should be displayed.
- If there are some local variables, such as $next_p$, $pred_p$ and $lock_p$ for each process p , to processes or nodes, then we should design the layout of the visual representations for them so that we can visually/graphically identify what variables or observable components are local to what processes or nodes; for example, all local variables for each process should be aligned horizontally.

VI. CONCLUSION

We have used MCS protocol to demonstrate two things. One is that the old state picture design for the protocol can be more visualized. The other is that the new state picture design makes it possible for human users to more visually/graphically perceive interesting phenomena in graphical animations and then conjecture more non-trivial state machine characteristics. We have summarized some new tips on how to make state picture designs.

REFERENCES

- [1] T. T. T. Nguyen and K. Ogata, "Graphical animations of state machines," in *15th DASC*, 2017, pp. 604–611.
- [2] D. D. Bui and K. Ogata, "Graphical animations of the Suzuki-Kasami distributed mutual exclusion protocol," *JVLC*, vol. 2019, no. 2, pp. 105–115, 2019.
- [3] T. T. T. Nguyen and K. Ogata, "Graphically perceiving characteristics of the MCS lock and model checking them," in *7th SOFL+MSVL*, 2017, pp. 3–23.
- [4] K. W. Brodlie, et al., Ed., *Scientific Visualization: Techniques and Applications*. Springer, 1992.
- [5] I. Beschastnikh, et al., "Visualizing distributed system executions," *ACM TOSEM*, vol. 29, no. 2, p. 38 pages, 2020.
- [6] J. Magee, et al., "Graphical animation of behavior models," in *22nd ICSE*, 2000, pp. 499–508.
- [7] M. Clavel, et al., Ed., *All About Maude*, ser. LNCS. Springer, 2007, vol. 4350.
- [8] J. M. Mellor-Crummey and M. L. Scott, "Algorithms for scalable synchronization on shared-memory multiprocessors," *ACM TOCS*, vol. 9, no. 1, pp. 21–65, 1991.