Example: Model Checking and Repair

Barbara Jobstmann
Verimag/CNRS (Grenoble, France)

November 13, 2009

Homework

1. Given $G = (S, S_0, E)$ and $F \subseteq Q$, give an algorithm that computes the 0-Attractor(F) in time $O(|E|)$.

Solution:

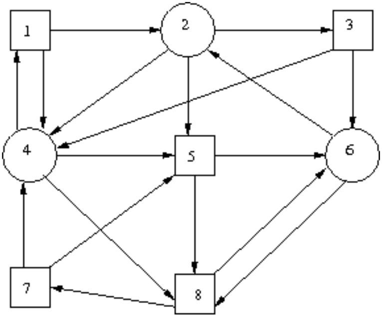
1. Preprocessing: Compute for every state $s \in S_1$ outdegree $\text{out}(s)$
2. Set $n(s) := \text{out}(s)$ for each $s \in S_1$
3. To breadth-first search backwards from F with the following conventions:
   - mark all $s \in F$
   - mark $s \in S_0$ if reached from marked state
   - mark $s \in S_1$ if $n(s) = 0$, other set $n(s) := n(s) - 1$.

The marked vertices are the ones of $\text{Attr}_0(F)$.

Homework (cont.)

2. Consider the game graph shown in below and the following winning conditions:
   (a) $\text{Occ}(\rho) \cap \{1\} \neq \emptyset$ and
   (b) $\text{Occ}(\rho) \subseteq \{1, 2, 3, 4, 5, 6\}$ and
   (c) $\text{Inf}(\rho) \cap \{4, 5\} \neq \emptyset$.

Compute the winning regions and corresponding winning strategies showing the intermediate steps (i.e., the Attractor and Recurrence sets) of the computation.



Homework (cont.)

3. Given a game graph $G = (S, S_0, T)$ and a set $F \subseteq S$. Let W_0 and W_1 be the winning regions of Player 0 and Player 1, respectively, in the Buchi game (G, F). Prove or disprove:
   (a) The winning set of Player 0 in the safety game for (G, W_0) is W_0,
   (b) If f_0 is a winning strategy for Player 0 in the safety game for (G, W_0), then f_0 is also a winning strategy for Player 0 in the Buchi game for (G, F),
   (c) the winning set of Player 1 in the guaranty game for (G, W_0) is W_1, and
   (d) if f_1 is a winning strategy for Player 1 in the guaranty game for (G, W_0), then f_1 is also a winning strategy for Player 1 in the Buchi game (G, F).

MC and Repair Example

Lock Example

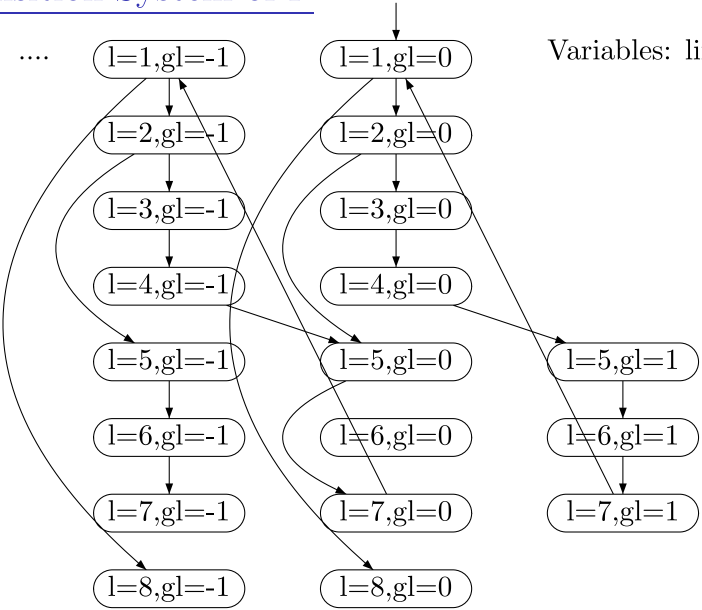
```
     ...
1    while(...) {
2      if (...) {
3        lock();
4        gotlock++;
       }
       ...
       ...
5      if (gotlock!=0)
6        unlock();
7      gotlock--;
     }
8    ...
```

Properties

1. P1: do not aquire a lock twice
2. P2: do not call unlock without holding the lock

Transition System of P

Variables: line, gotlock



Recall LTL

Boolean Operators: $\neg$, $\wedge$, $\vee$, $\rightarrow$,...

Temporal Operators:

- **next**: $\bigcirc\varphi$ ... in the next step φ holds
- **until**: $\varphi_1 \mathsf{U} \varphi_2$ ... at some point in the future φ_2 holds and until then φ_1 holds

Useful abbreviations:

- **eventually**: $\Diamond\varphi = \mathsf{true} \mathsf{U} \varphi$
- **always**: $\Box\varphi = \neg\Diamond\neg\varphi$
- **weakuntil**: $\varphi_1 \mathsf{W} \varphi_2 = (\varphi_1 \mathsf{U} \varphi_2) \vee \Box\varphi_1$

Note that

$\neg(\varphi_1 \mathsf{U} \varphi_2) = (\neg\varphi_2 \mathsf{U} \neg\varphi_1 \wedge \neg\varphi_2) \vee \Box\neg\varphi_2 = \neg\varphi_2 \mathsf{W}(\neg\varphi_1 \wedge \neg\varphi_2)$.

Our properties in LTL

1. P1: do not aquire a lock twice

   Whenever we have called lock, we are not allowed to call it again before calling unlock.

Our properties in LTL

1. P1: do not aquire a lock twice

   Whenever we have called lock, we are not allowed to call it again before calling unlock. $\Box((l = 3) \rightarrow \bigcirc(\neg(l = 3)\,\mathsf{W}(l = 6)))$

Our properties in LTL

1. P1: do not aquire a lock twice

   Whenever we have called lock, we are not allowed to call it again before calling unlock. $\Box((l = 3) \rightarrow \bigcirc(\neg(l = 3)\,\mathsf{W}(l = 6)))$

2. P2: do not call unlock without holding the lock

Our properties in LTL

1. P1: do not aquire a lock twice

   Whenever we have called lock, we are not allowed to call it again before calling unlock. $\Box((l = 3) \rightarrow \bigcirc(\neg(l = 3)\,\mathsf{W}(l = 6)))$

2. P2: do not call unlock without holding the lock

   $(\neg(l = 6)\,\mathsf{W}(l = 3)) \wedge (l = 6 \rightarrow \bigcirc(\neg(l = 6)\,\mathsf{W}(l = 3)))$

From LTL to Automata: Expansion rules

- $\Box\varphi = \varphi \wedge \bigcirc\Box\varphi$
- $\Diamond\varphi = \varphi \vee \bigcirc\Diamond\varphi$
- $\varphi_1 \,\mathsf{U}\, \varphi_2 = \varphi_2 \vee (\varphi_1 \wedge \bigcirc\varphi_1 \,\mathsf{U}\, \varphi_2)$
- $\varphi_1 \,\mathsf{W}\, \varphi_2 = \varphi_2 \vee (\varphi_1 \wedge \bigcirc\varphi_1 \,\mathsf{W}\, \varphi_2)$

Example: $\Box((l = 3) \rightarrow \bigcirc(\neg(l = 3) \,\mathsf{W}(l = 6)))$

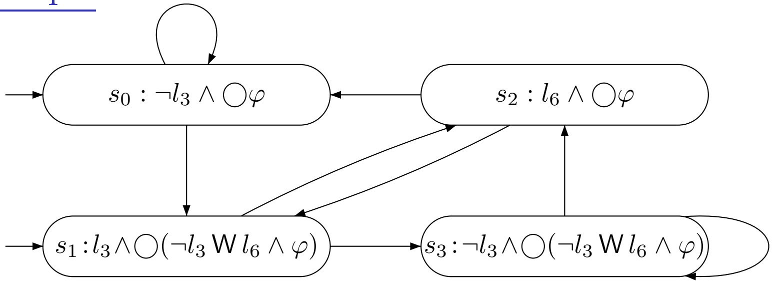
Shortcuts: l_3 for $(l = 3)$ and l_6 for $(l = 6)$

$$\varphi = \Box(\neg l_3 \vee (l_3 \wedge \bigcirc(\neg l_3 \,\mathsf{W}\, l_6)))$$

Expand: $(\neg l_3 \vee (l_3 \wedge \bigcirc(\neg l_3 \,\mathsf{W}\, l_6))) \wedge \bigcirc\varphi$

DNF: $s_0 \vee s_1$ with $s_0 = \neg l_3 \wedge \bigcirc\varphi$ and $s_1 = l_3 \wedge \bigcirc(\neg l_3 \,\mathsf{W}\, l_6 \wedge \varphi)$

Example



Expand: $\neg l_3 \, \mathsf{W} \, l_6 \wedge \varphi$

$(l_6 \vee (\neg l_3 \wedge \bigcirc(\neg l_3 \, \mathsf{W} \, l_6))) \wedge ((\neg l_3 \wedge \bigcirc\varphi) \vee (l_3 \wedge \bigcirc(\neg l_3 \, \mathsf{W} \, l_6 \wedge \varphi)))$

(1) $l_6 \wedge \neg l_3 \wedge \bigcirc\varphi \; : \; s_2$

(2) $l_6 \wedge l_3 \cdots = \mathsf{false}$

(3) $(\neg l_3 \wedge \bigcirc(\neg l_3 \, \mathsf{W} \, l_6)) \wedge (\neg l_3 \wedge \bigcirc\varphi) \; : \; s_3$

(4) $(\neg l_3 \wedge \cdots \wedge l_3 \cdots = \mathsf{false}$

Model Checking

$$L(\text{Program}) \subseteq L(P1)$$

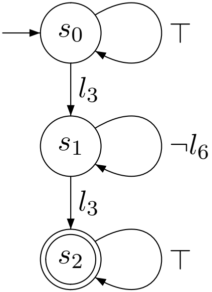
$$L(\text{Program}) \cap L(\neg P1) = \emptyset$$

Automaton for ¬ P1

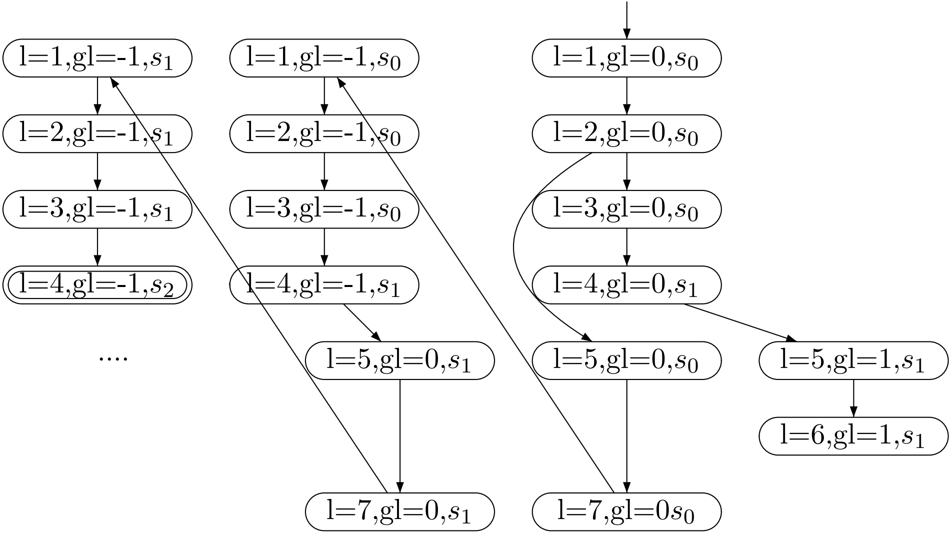
$\neg P1 = \neg\Box(l_3 \rightarrow \bigcirc(\neg l_3 \mathsf{W}\, l_6))$

$\neg P1 = \Diamond(l_3 \wedge \bigcirc(\neg l_6 \mathsf{U}\, l_3))$

Simplified version:



Product of Program and Property



Counterexample

1. Line 1: enter while loop
2. Line 2: skip over if
3. ...
4. Line 1: enter while loop
5. Line 2: enter if (call lock)
6. ...
7. Line 1: enter while loop
8. Line 2: enter if (call lock again)

```
      ...
1     while(...) {
2       if (...) {
3         lock();
4         gotlock++;
        }
        ...
        ...
5       if (gotlock!=0)
6         unlock();
7       gotlock--;
      }
8     ...
```

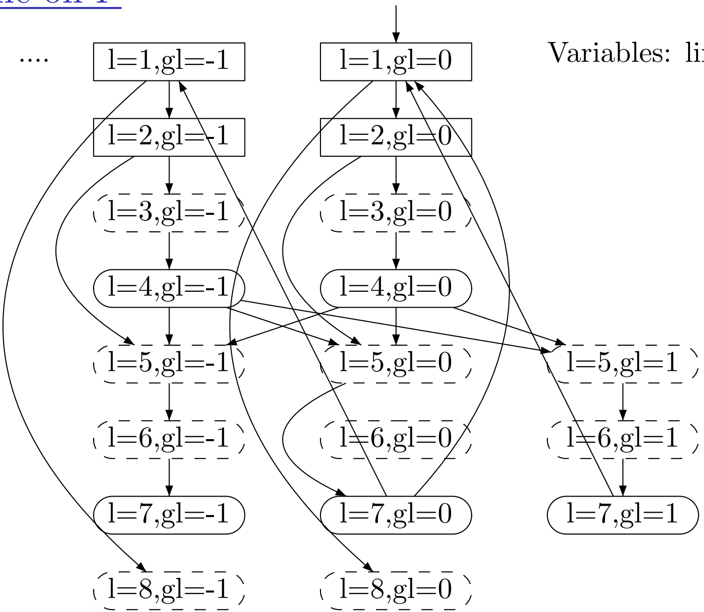
Repair

Repair: Step 1 - Free variables

```
while(...) {
  if (...) {
    lock();
    gotlock=?;
  }
  ...
  ...
  if (gotlock!=0)
    unlock();
  gotlock=?;
}
...
```

Game on P

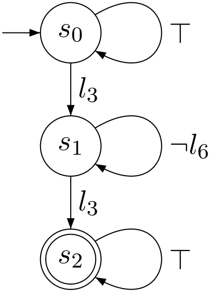
Variables: line, gotlock



Repair: Winning Condition

Note in MC: non-determinism due to input and due to automaton are treated the same way!

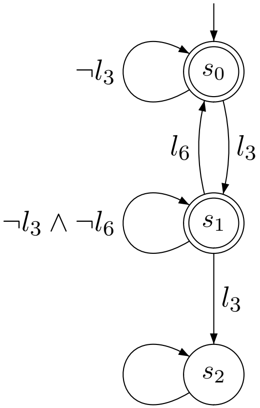
In Game: non-determinism may cause troubles.



Deterministic Automata/Observer

Recall,

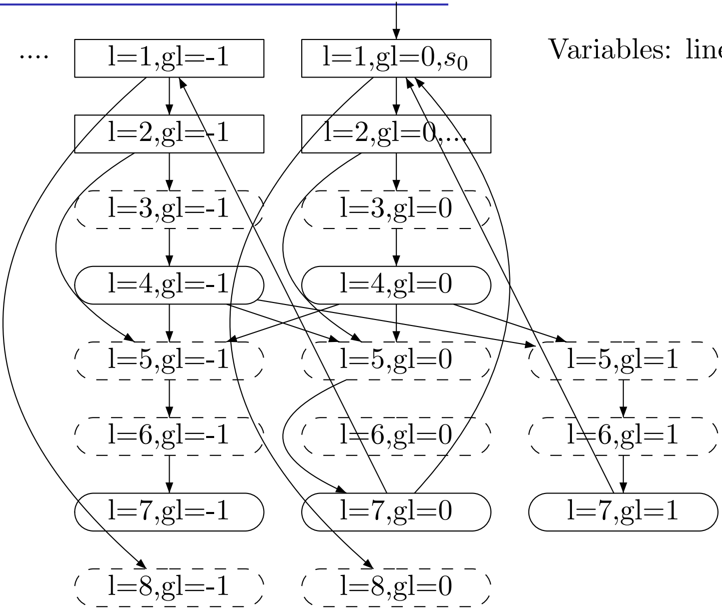
$$\varphi = \Box(\neg l_3 \vee (l_3 \wedge \bigcirc(\neg l_3 \, \mathsf{W} \, l_6)))$$



Note: this is a safety automaton.

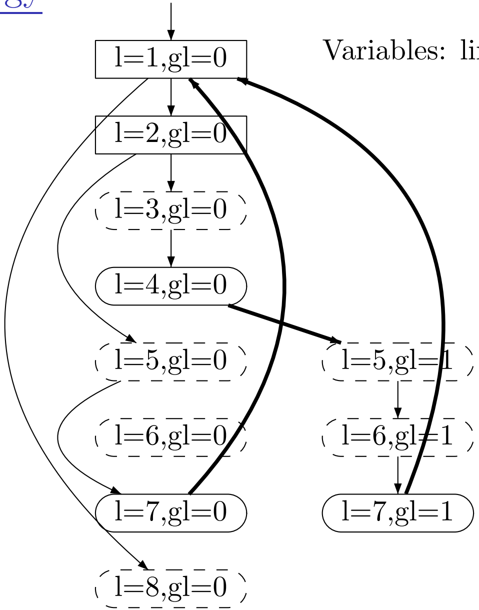
Add Automaton to Game on P

Variables: line, gotlock



A Winning Strategy

Variables: line, gotlock



A Correct Program

```
1   while(...) {
2     if (...) {
3       lock();
4       gotlock=1;
      }
      ...
      ...
5     if (gotlock!=0)
6       unlock();
7     gotlock=0;
    }
8   ...
```