Examples & board definitions for part 2 of the “Model Checking and Games” lecture series

Ruediger Ehlers

September 2, 2019

1 On notation

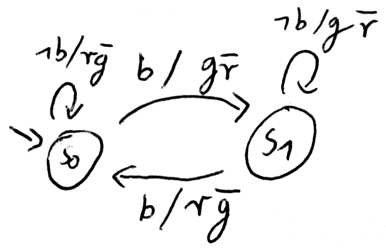
Henceforth, we use sets and their *characteristic functions* interchangeably. A *characteristic function* of a set $S' \subseteq S$ is a function $f : S \rightarrow \mathbb{B}$ such that for all $s \in S$, $f_{S'}(s) = \textbf{true}$ if $s \in S'$, and $f(s) = \textbf{false}$ otherwise.

For instance, $\{a, b\} \subseteq S$ for the base set $S = \{a, b, c\}$ is seen as equivalent to the function $\{a \mapsto \textbf{true}, b \mapsto \textbf{true}, c \mapsto \textbf{false}\}$.

Henceforth, we will also use 0 and 1 and **ff** and **tt** as synonyms for **false** and **true**, which are the elements of the set of Booleans $\mathbb{B}$. Many examples otherwise become too lengthy.

An example for a trace

Let us consider the following Mealy machine (in graphical notation):



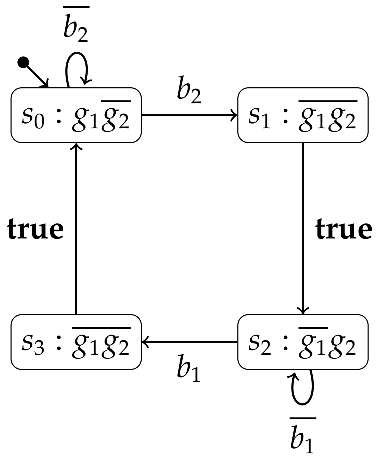
The Mealy machine can also be represented using a *transition table*, which explains for every computation of state and next input what the next state and the next output will be:

Predecessor state	Next input	Successor state	Next output
s_0	$\emptyset$	s_0	$\{r\}$
s_0	$\{b\}$	s_1	$\{g\}$
s_1	$\emptyset$	s_1	$\{g\}$
s_1	$\{b\}$	s_0	$\{r\}$

A combination of example trace and an example run:

$\rho =$	$\{b \mapsto \textbf{false}, r \mapsto \textbf{true}, g \mapsto \textbf{false}\}$	$\{b \mapsto \textbf{true}, r \mapsto \textbf{false}, r \mapsto \textbf{true}\}$	$\{b \mapsto \textbf{true}, r \mapsto \textbf{true}, g \mapsto \textbf{false}\}$	$\ldots$		$\in (2^{\mathsf{AP}^I \cup \mathsf{AP}^O})^\omega$
$\pi =$	s_0	s_0	s_1	s_0	$\ldots$	$\in S^\omega$

2 A Moore machine

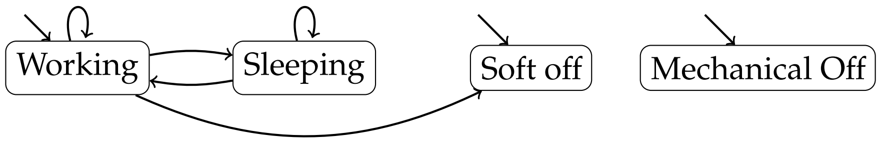


Example trace/run combination:

$\rho =$	$\{g_1, b_1\}$	$\{b_1, b_2, g_1\}$	$\emptyset$	g_2	$\ldots$	
$\pi =$	s_0	s_0	s_1	s_2	s_2	$\ldots$

3 Transition systems

Example: Processor ACPI States

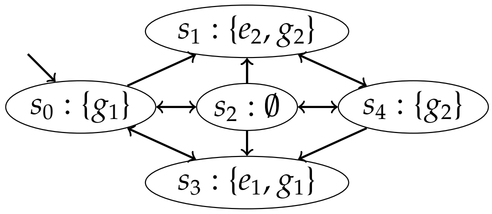


4 Labeled Transition Systems

Example: Processor ACPI States - Without state names, but with "computing" labels

5 Traces of Kripke structures

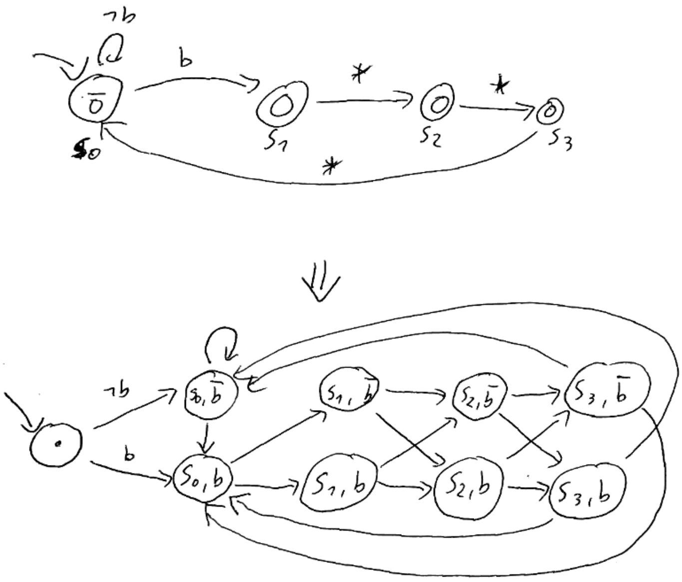
A traffic light with an ambulance override



Example trace: $\{g_1\}\ \{e_1, g_1\}\ \{g_1\}\ \{\emptyset\}\ \{e_1, g_1\} \ldots$

6 Translation from Mealy/Moore machines to Kripke structures

Example: A light switch that switches itself off after a while. Button presses with the light on are ignored.



Proposition 6.1. *Let $\mathcal{M} = (S, \Sigma^I, \Sigma^O, \delta, L, s_0)$ be a Moore machine with $\Sigma^I = 2^{\mathsf{AP}^I}$ and $\Sigma^O = 2^{\mathsf{AP}^O}$ and $\mathcal{K} = (S', S'_0, T', \mathsf{AP}, L')$ be the corresponding Kripke structure.*

There is a bijection between traces $\rho = (\rho^O_0, \rho^I_0)(\rho^O_1, \rho^I_1) \ldots \in (\Sigma^I \times \Sigma^O)^\omega$ of $\mathcal{M}$ and traces $\rho' = \rho_0\rho_1 \ldots \in (2^{\mathsf{AP}})^\omega$ of $\mathcal{K}$.

Proof. We prove a bijection between traces $\rho = (\rho^O_0, \rho^I_0)(\rho^O_1, \rho^I_1) \ldots \in (\Sigma^I \times \Sigma^O)^\omega$ of $\mathcal{M}$ and traces $\rho' = \rho_0\rho_1 \ldots \in (2^{\mathsf{AP}})^\omega$ with $\rho'_0 = \emptyset$ and $\rho'_{j+1} = \rho^O_j \cup \rho^I_j$ for all $j \in \mathbb{N}$.

<u>$\Rightarrow$:</u> If ρ is a trace of $\mathcal{M}$ and $\pi = \pi_0\pi_1 \ldots$ is the corresponding run of $\mathcal{M}$, then by the definition of traces and runs, we have that for all $j \in \mathbb{N}$, we have $(\pi_j, \rho^I_j, \pi_{j+1}) \in \delta$. By the definition of $\mathcal{K}$, this implies that for all $j \in \mathbb{N}$, we have $((\pi_j, \rho^I_j), (\pi_{j+1}, \rho^I_{j+1})) \in T$. By the definition of L', we furthermore have that $L'((\pi_j, \rho^I_j)) = \rho^I_j \cup \rho^O_j$. Hence, we have $\rho' = \emptyset(\rho^O_0, \rho^I_0)(\rho^O_1, \rho^I_1) \ldots$.

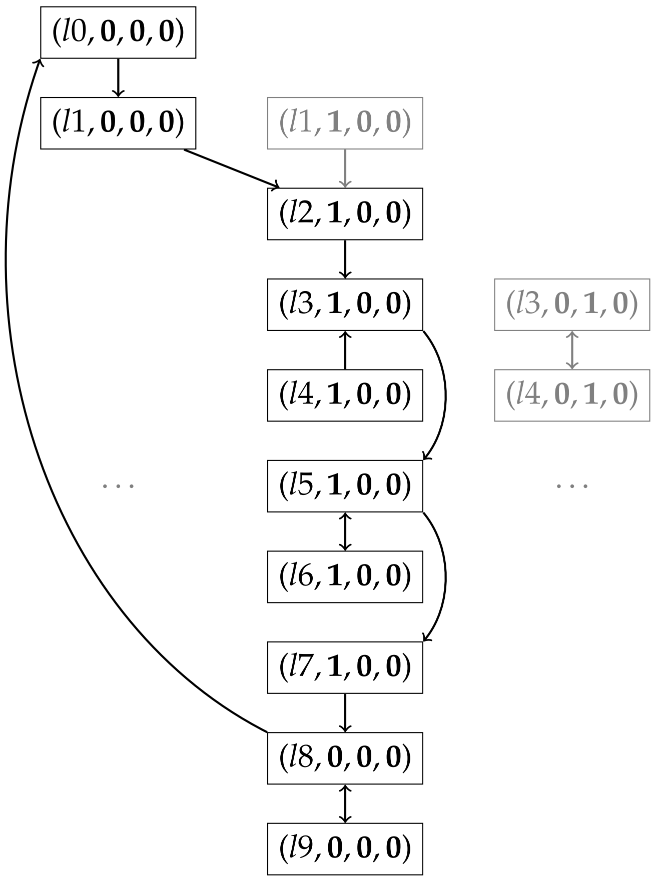
$\underline{\Leftarrow}$: Let $\rho = \emptyset(\rho^I_1, \rho^O_1)(\rho^I_2, \rho^O_2)\ldots \in (2^{\mathsf{AP}})^\omega$ be a trace of $\mathcal{K}$.[1] Be the definition of a trace, there exists a corresponding run $\pi' = \pi'_0(\pi'_1, i_1)(\pi'_2, i_2)\ldots \in S'^\omega$ for which we furthermore have $\pi'_0 = \cdot$ (by the fact that $\cdot$ is the only initial state in $\mathcal{K}$), and for all $j \in \{1, 2, \ldots\}$, we have $\rho^O_j = L(\pi_j)$ and $\rho^I_j = i_j$ (by the definition of T'). We build a run $\pi = \pi_0\pi_1$ for $\mathcal{M}$ for ρ by setting $\pi_j = \pi'_{j+1}$ for all $j \in \mathbb{N}$. By the fact that by the construction of T', we have $\pi_{j+1} = \delta(\pi_j, \rho^I_{j-1})$ for all $j \in \mathbb{N}$, it follows that π is a run of $\mathcal{M}$. Since furthermore by $L(\pi_j) = \rho^O_{j-1}$ for all $j \in \mathbb{N}$, it follows that $(\rho^I_1, \rho^O_1)(\rho^I_2, \rho^O_2)\ldots$ is also a trace of $\mathcal{M}$.

□

7 Programs as Kripke structures – Single process

States are *named* by:

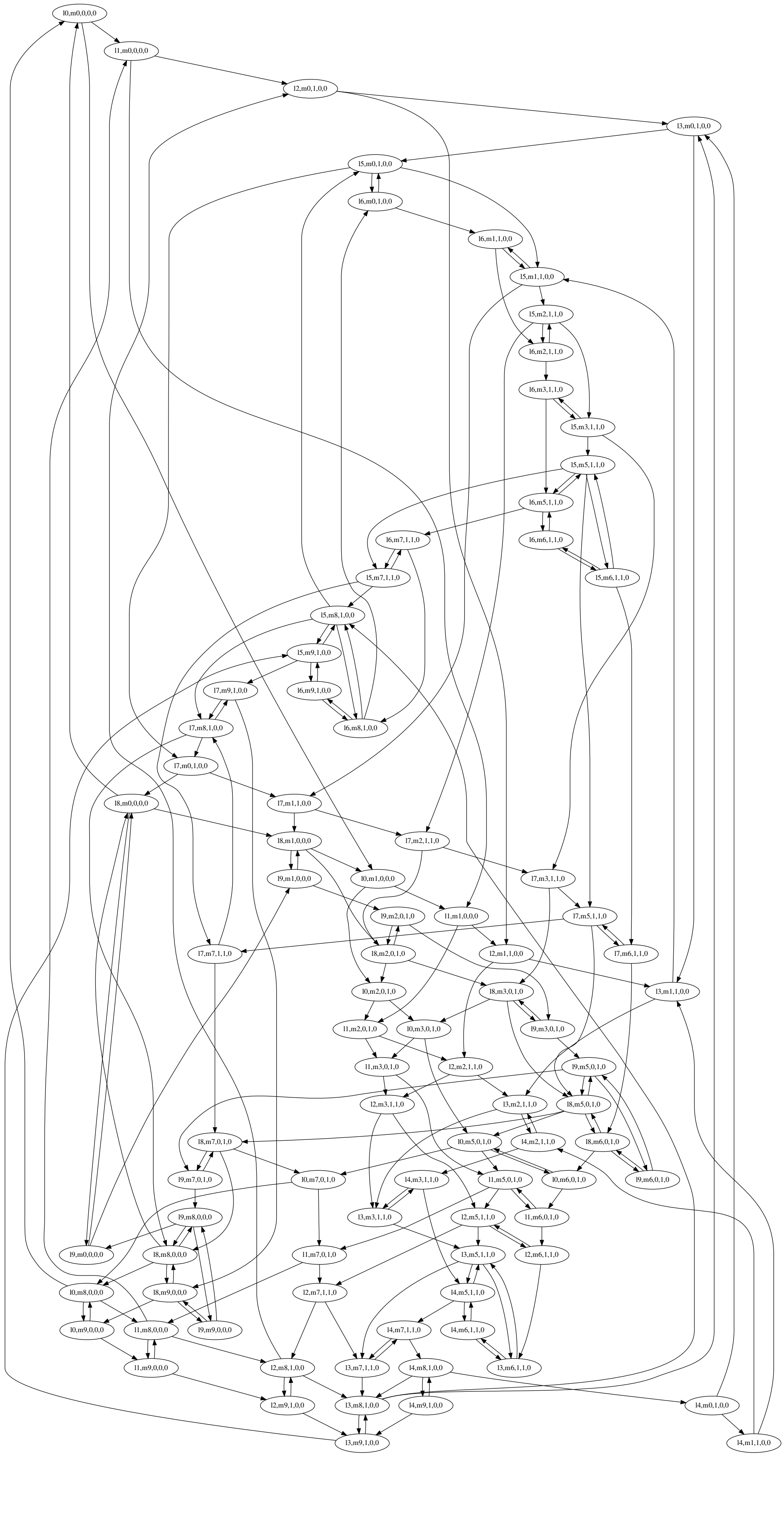
- their program location (also called the *program counter*),
- the value of `flag0`,
- the value of `flag1`, and
- the value of `turn`.



Note that the Kripke structure contains states for all combinations of program location and variable values. This becomes important for the composition of different processes.

[1]We can assume, w.l.o.g., that the first element is $\emptyset$ as the only initial state of ρ is labeled by $\emptyset$.

8 Programs as Kripke structures – Both processes



9 SPIN – First model (warning – with bug)

```
int turn = 0;
int flag0 = 0;
int flag1 = 0;
active proctype A() {
    do
        :: {
            flag0 = 1;
            turn = 0;
            do
                :: flag1==1 && turn==0 -> skip;
                :: else -> break;
            od
            do
                :: skip;
                :: break;
            od
            flag0 = 0;
            do
                :: skip;
                :: break;
            od
        }
    od
}

active proctype B() {
    do
        :: {
            flag1 = 1;
            turn = 1;
            do
                :: flag0==1 && turn==0 -> skip;
                :: else -> break;
            od
            do
                :: skip;
                :: break;
            od
            flag1 = 0;
            do
                :: skip;
                :: break;
            od
        }
    od
}
```

10 SPIN – Model with verification condition (still with bug)

```
int turn = 0;
int flag0 = 0;
int flag1 = 0;
int critical1 = 0;
active proctype A() {
    do
        :: {
            flag0 = 1;
            turn = 0;
            do
                :: flag1==1 && turn==0 -> skip;
                :: else -> break;
            od
            do
                :: assert(critical1==0);
                :: break;
            od
            flag0 = 0;
            do
                :: skip -> skip;
                :: break;
            od
        }
    od
}

active proctype B() {
    do
        :: {
            flag1 = 1;
            turn = 1;
            do
                :: flag0==1 && turn==0 -> skip;
                :: else -> break;
            od
            critical1 = 1;
            do
                :: skip -> skip;
                :: break;
            od
            critical1 = 0;
            flag1 = 0;
            do
                :: skip -> skip;
                :: break;
            od
```

```
        }
    od
}
```

The bug is that the second condition "`turn==0`" should be "`turn==1`".