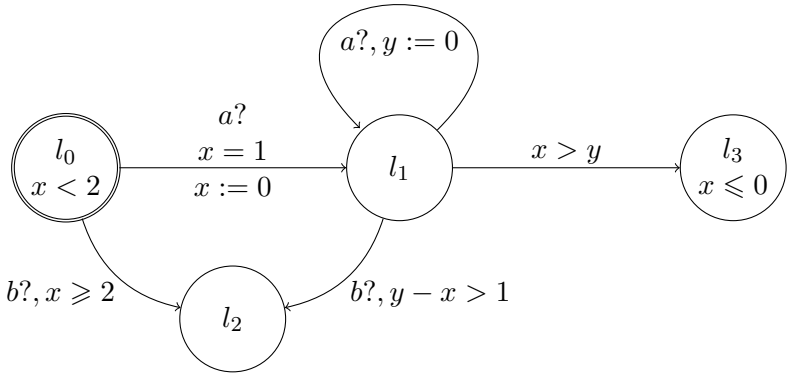
Exercícios para Arquitectura e Cálculo

Dep. Informática, Universidade do Minho
Renato Neves e José Proença

2019-2020

Modelação e análise de sistemas ciber-físicos

Exercício 1. Considere o automato de tempo real abaixo.



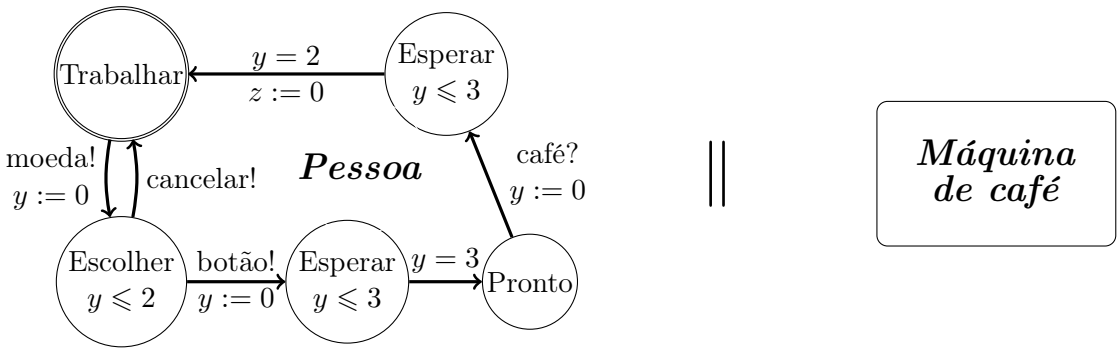
1.1. Defina o automato formalmente como um tuplo $(L, L_0, Act, C, Tr, Inv)$.

1.2. O automato tem algum caminho (*trace*) com comportamento Zeno? Justifique.

1.3. O automato tem algum caminho (*trace*) com um *timelock*? Justifique.

1.4. As localizações l_1, l_2, e l_3 são alcançáveis? Se sim, indique um caminho que o mostre.

Exercício 2. Considere o sistema abaixo composto por um autómato a representar um utilizador, e uma máquina de café. A pessoa tenta repetidamente introduzir uma moeda, carregar num botão, e retirar o café para que ele(a) posso voltar ao trabalho. Às vezes arrepende-se e cancela a operação depois de inserir a moeda e volta a trabalhar. Esta pessoa demora algum tempo a reagir entre ações.



2.1. Modele o autómato da máquina de café, em paralelo à pessoa, sabendo que a máquina de café demora T tempo a produzir café, e disponibiliza-o por um período de tempo máximo MAX (ao fim do qual o utilizador é obrigado a retirá-lo para não perder o café).

2.2. Diga se algum dos 2 autómatos tem algum comportamento Zeno ou Timelock quando isolado. Justifique a sua resposta, e sugira formas de evitar estes comportamentos caso encontre algum.

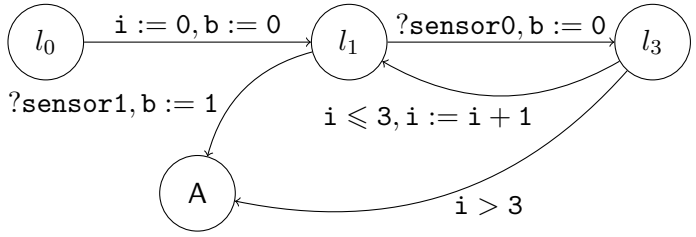
2.3. Formalize as seguintes propriedades usando lógica temporal (como em UPPAAL).

- ϕ_1 – A pessoa nunca passa mais de 500 unidades de tempo a trabalhar sem ir à máquina de café.
- ϕ_2 – Sempre que a pessoa inserir uma moeda ela terá um café.

Exercício 3. Neste exercício iremos estudar dois programas ciber-físicos via teoria de autómatos híbridos. Um dos programas em questão é um controlador digital que testa se pode entrar num procedimento A com base em informação providenciada por um sensor. O programa é descrito pelo seguinte código.

```
int i = 0;
bool b = 0;
b := ?sensor;
while (¬b ∧ i ⩽ 3) {
  i := i + 1;
  b := ?sensor
}
A
```

Note que logo que a variável `b` toma o valor `true` o programa avança para o trecho de código A. A expressão `?sensor` representa uma chamada ao sensor. O programa acima é representado pelo seguinte autómato.



No autómato acima, a expressão ?`sensor0` representa o evento "*sensor envia o valor* 0" e a expressão ?`sensor1` representa o evento análogo.

3.1. Qual o número máximo de vezes que o sensor (?`sensor`) é chamado por este programa? Para esse número apresente o caminho (*trace*) correspondente.

Vamos agora analisar o segundo programa do sistema: este vai ser o sensor propriamente dito e tem como objectivo ler os diferentes valores que um processo físico `x` toma. O sensor e o processo físico em questão são dados pelo seguinte código.

```
real x = 0;
while true do {
  ẋ = 2 for 3;
  if x ⩾ 4 then send(!sensor0) else send(!sensor1)
  }
```

A expressão $\dot{\texttt{x}} = \texttt{2}$ `for` `3` corresponde à evolução do processo físico `x` – por exemplo, nível da água a subir – ao longo do tempo, em que o valor `3` corresponde ao tempo de espera que o sensor toma antes de enviar informação para o exterior (`!sensor0` ou `!sensor1`).

3.2. Modele este último programa através de um autómato híbrido.

Agora temos os dois programas em questão modelados como autómatos híbridos, e portanto podemos estudar a sua *composição paralela*.

3.3. Assuma que os programas estão a correr em paralelo. Será que é possível chegarmos a um estado do programa em que `i = 3`? Justifique.