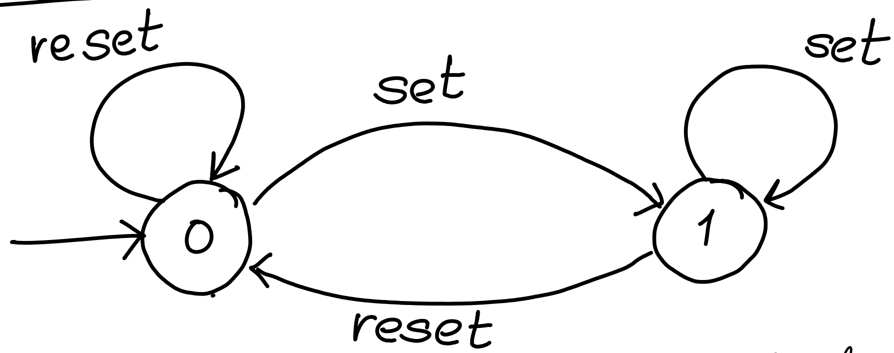
FOAV Lecture 4 14 Jan 2010

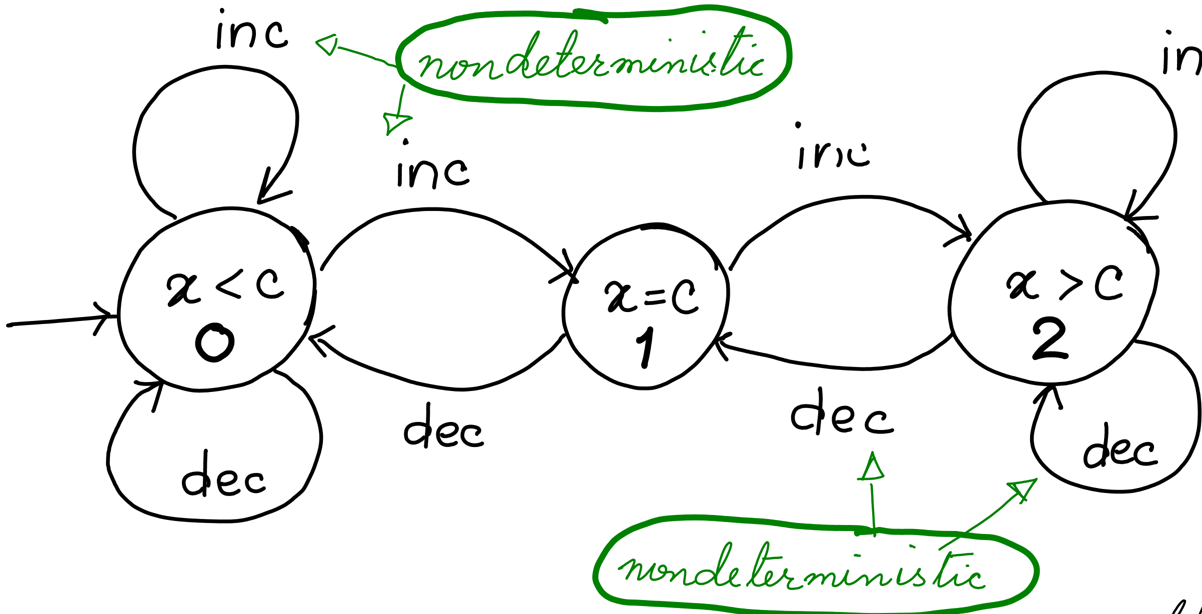
Examples of Transition Systems

1. A boolean variable



The above LTS models a boolean variable.
A similar modelling of an integer variable on the other hand would require an infinite number of states and further setting the integer variable to a particular value would require an infinite number of actions too.

However given a finite number of integer constants or a finite number of integer intervals we could provide a nondeterministic model of an integer variable with a finite number of actions such as inc and dec.



Note the nondeterminism in this modelling of the integer variable. There are several reasons for the use of nondeterminism.

Nondeterminism in Modelling

1. It may be used as a form of abstraction where the full modelling with deterministic mechanisms may be too detailed, intricate unimportant or irrelevant
2. As a form of underspecification when the exact deterministic mechanism may be unknown
3. The interleaving of concurrent systems is often achieved through a non-deterministic modelling of the system

However in most cases of verifying control mechanisms we require that the mechanism of interest be deterministic.

The above example of integer variables also has important consequences for model-checking.

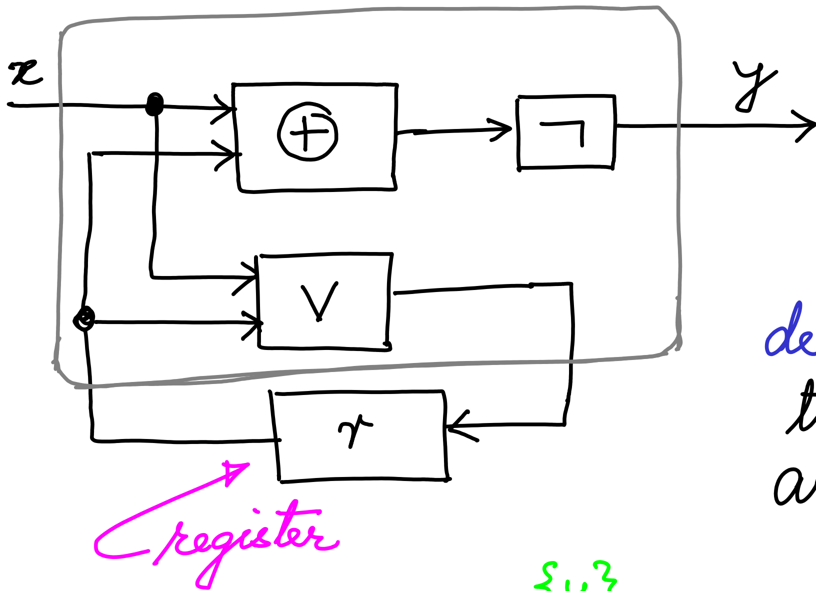
$$\boxed{\text{Model-checking is far more suitable for verifying the correctness of control-intensive applications rather than data-intensive ones.}}$$

In the above example we have modelled the integer variable using a DLTS where

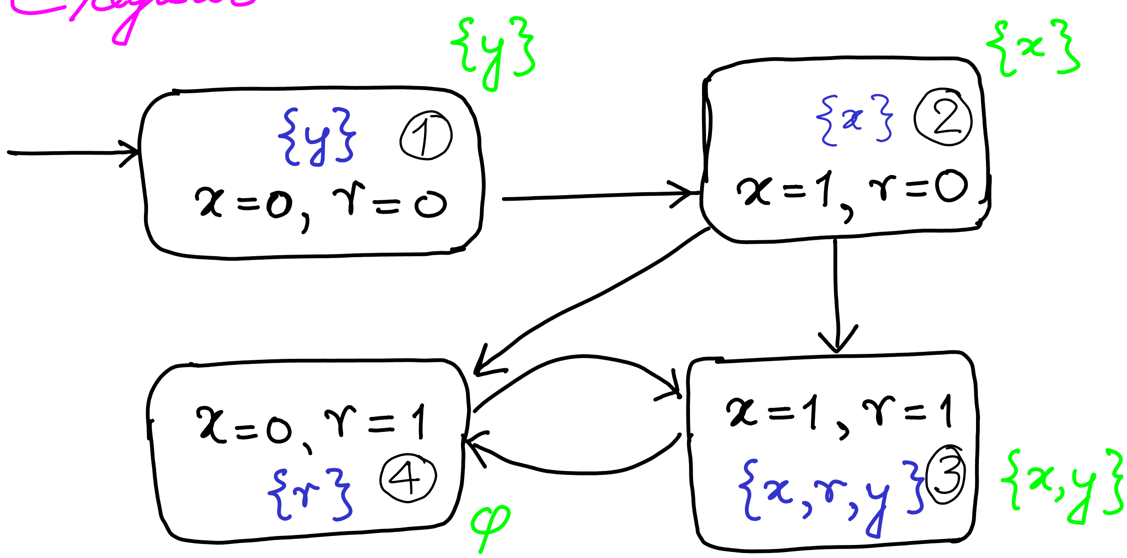
$$D(1) = \{x < c\} \quad D(2) = \{x = c\} \quad D(3) = \{x > c\}$$

and $A = \{inc, dec\}$.

Sequential Hardware Circuits



A DTS with D decorations specifying the quantities that are 1.



In this case the labels (on the transitions) are omitted as they are considered irrelevant or unimportant.

If we were to consider the value of the register also irrelevant then we get the decorations D′

Interleaving of Transition Systems

$$\mathcal{L} = \langle S, A, \rightarrow_L, I_L, AP_L, D \rangle$$
$$\mathcal{M} = \langle T, B, \rightarrow_M, I_M, AP_M, E \rangle$$
$$\mathcal{N} = \langle U, C, \rightarrow_N, I_N, AP_N, F \rangle$$

The $\mathcal{N} = \mathcal{L} \,|||\, \mathcal{M}$ if

$U = S \times T$

$C = A \cup B$

$I_N = I_L \times I_M$

$F = D \cup E$

actually $F(s,t) = D(s) \cup E(t)$

$\rightarrow_N$ is the smallest relation such that

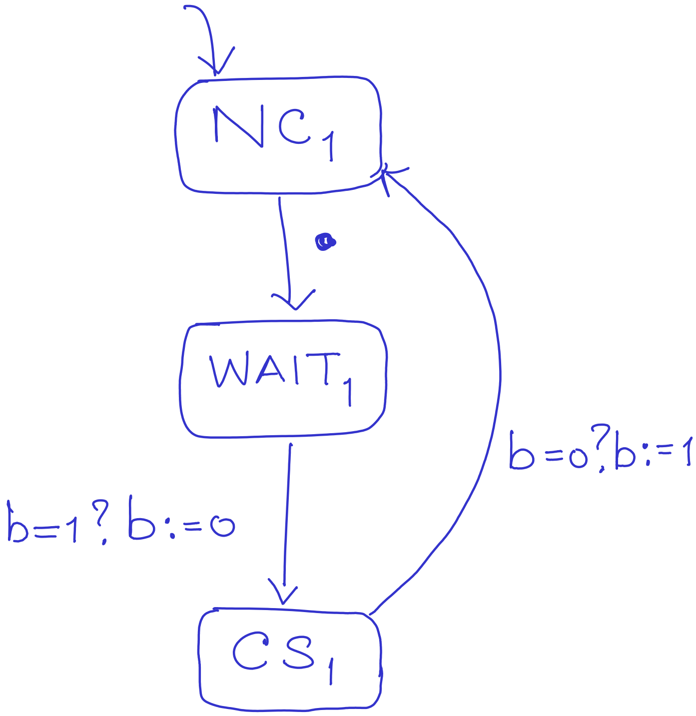
$$\frac{s \xrightarrow{a}_L s'}{(s,t) \xrightarrow{a}_N (s',t)} \qquad \frac{t \xrightarrow{b}_M t'}{(s,t) \xrightarrow{b}_N (s,t')}$$

Example. Consider two processes trying to access a shared resource through a binary semaphore which allows both testing & setting as a single atomic action.

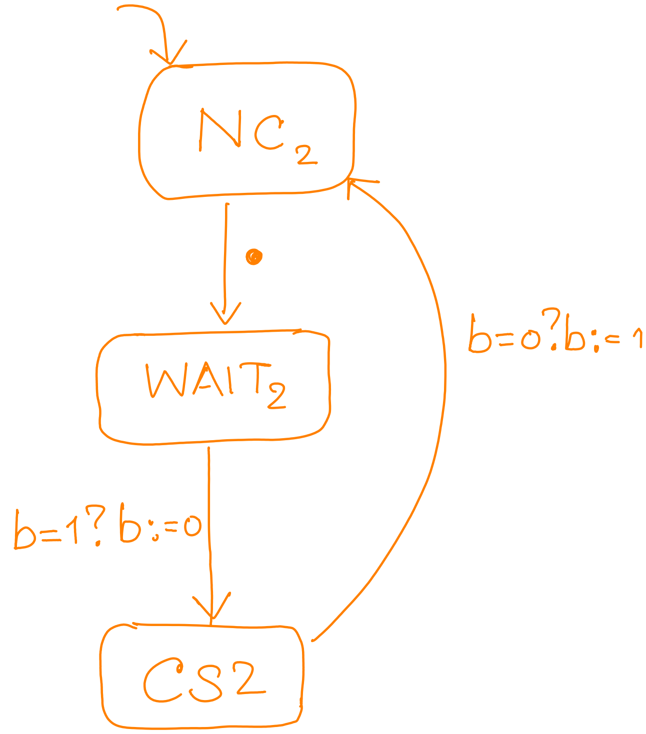
Let b be the semaphore and the action set on b be the following

$\boxed{b=1 \,?\, b:=0}$ $\qquad$ $\boxed{b=0 \,?\, b:=1}$

Each of these a single atomic action where the setting of the variable b cannot be done until it passes the test.

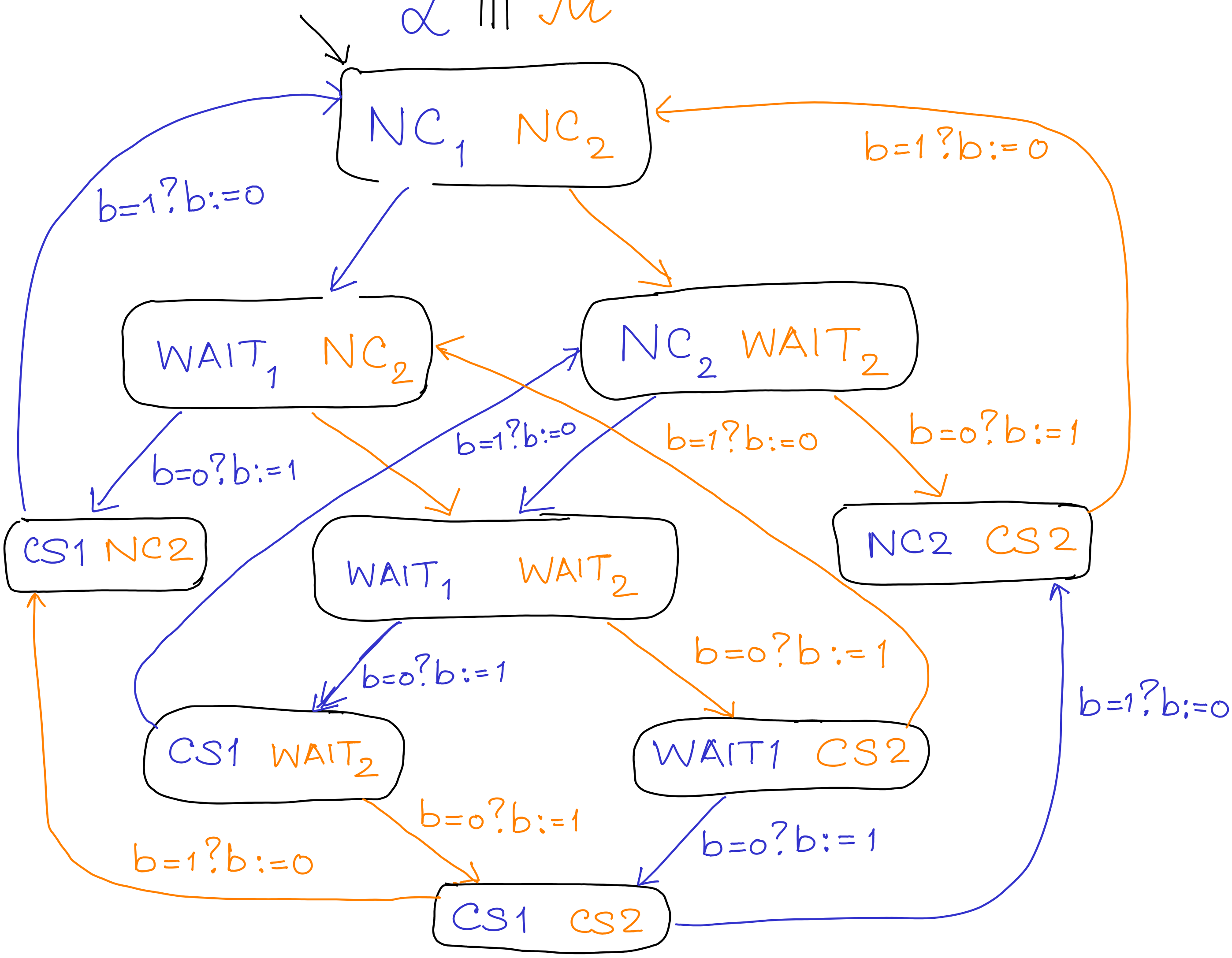


$\mathcal{L}$



$\mathcal{M}$

$\mathcal{L} \,|||\, \mathcal{M}$



It is easy to see that the state $\boxed{CS1 \quad CS2}$ will never be reached from the initial state. But this is under the assumption the the actions are all atomic and guaranteeing atomicity is often non-trivial