3. Modelling Sequential and Parallel Systems

Computer-Aided Verification

Dave Parker

University of Birmingham

2017/18

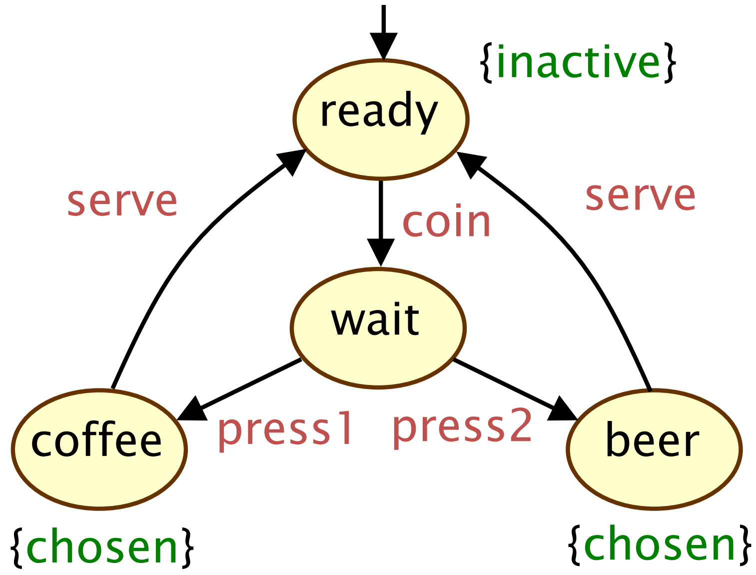
Recap

- Formal verification & model checking
  - need precise models of system behaviour over time
  - results of verification only as good as the models used
  - need to be wary of the state space explosion problem

- Labelled transition systems (LTSs)
  - states, transitions & labels
  - states capture all information needed…
    - to determine what happens next
    - and to specify properties to be verified

Labelled transition systems (LTSs)

- Transitions
  - Post(wait,press1) = {coffee}
  - Post(wait) = {coffee,beer}
  - Pre(ready) = {coffee,beer}

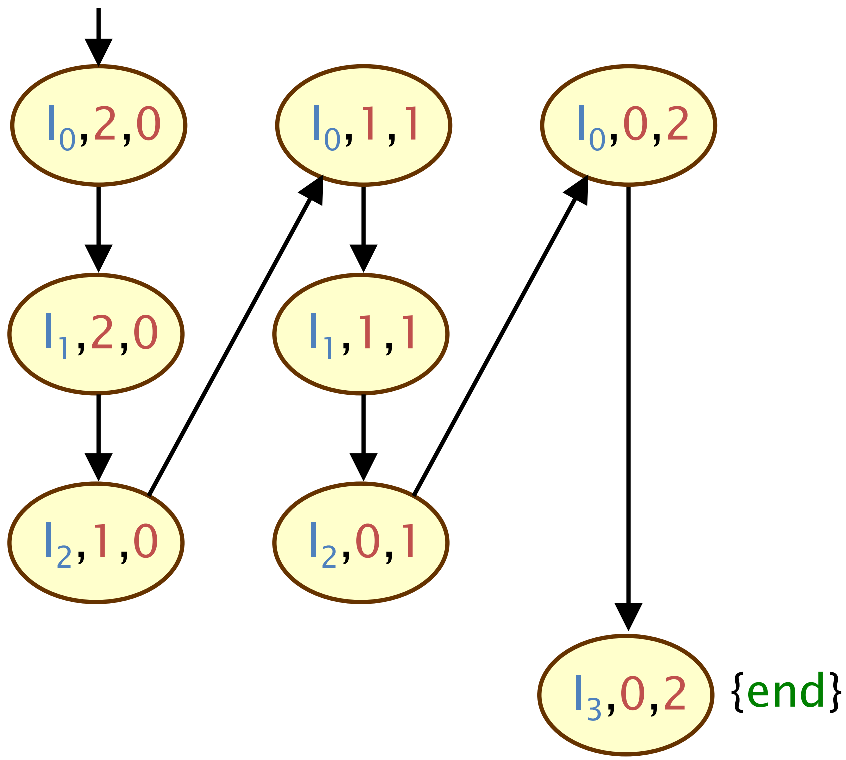


- A finite path
  - ready coin wait press1 coffee
- An infinite path/execution
  - ready coin wait press1 coffee serve ready (coin wait press2 beer serve ready)$^\omega$
- All states are reachable and non-terminal

Programs as LTSs

- How to model a (sequential) program as an LTS?
  - states are tuples (l_i,x,y) of location & variable values

```
     x := 2; y:= 0
l0 → while x > 0 do
l1 →    x := x – 1
l2 →    y := y + 1
     od
l3 →
```



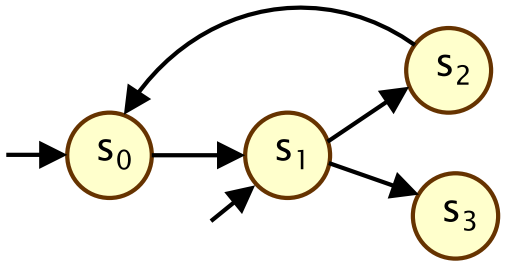
Overview

- Nondeterminism
- Parallelism and concurrency
  - interleaving, shared variables, handshaking
  - SOS-style semantics
  - see [BK08] chapter 2 (specifically: 2.1–2.1.1, 2.2–2.2.3)
- Linear-time properties of LTSs
  - see Chapter 3 of [BK08]

Nondeterminism

- Nondeterminism
  - the outcome of the event is not known in advance

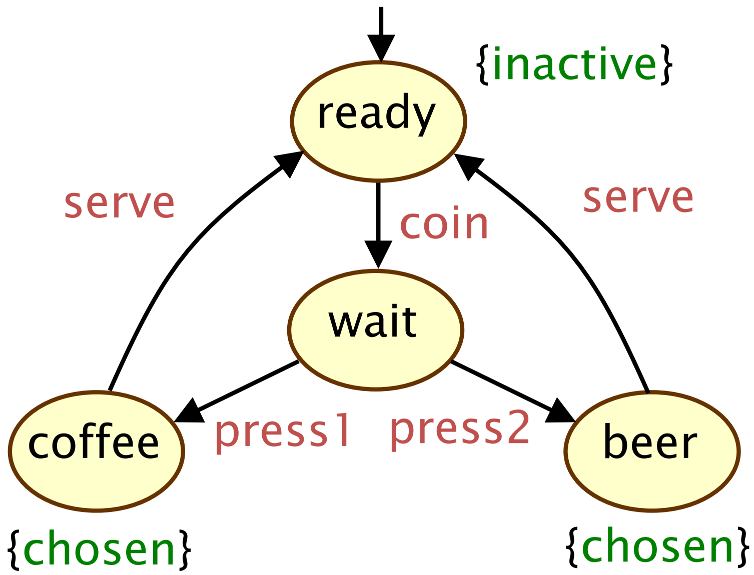
- Nondeterminism in labelled transition systems
  1. choice of action/transition in each state
  2. choice of initial state



- Uses of nondeterminism in system modelling
  1. unknown system environment (e.g. reactive systems, user input)
  2. abstraction (omitting detail), underspecification
  3. concurrency (parallelism)

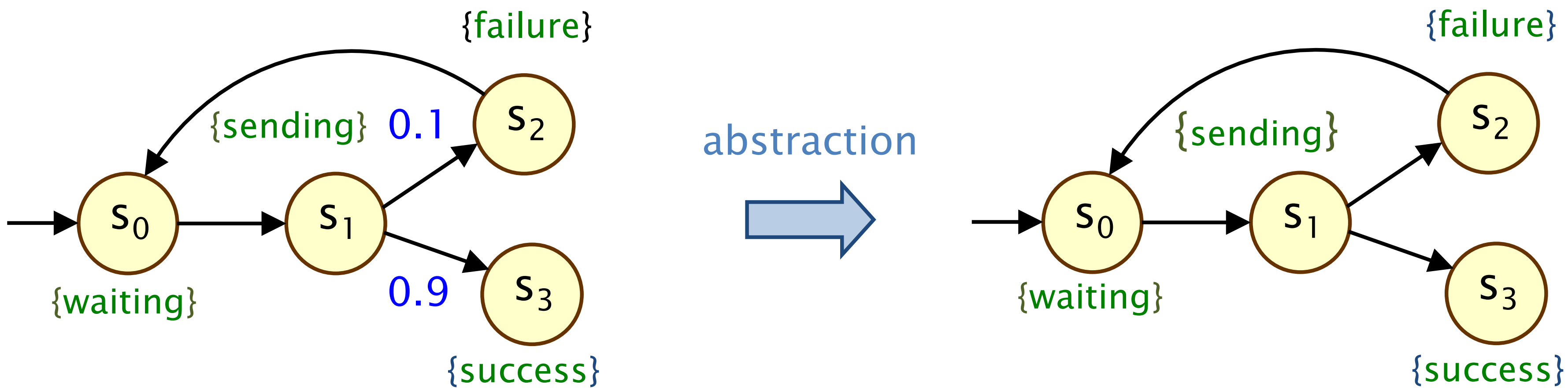
Unknown system environment

- E.g., interaction with human, controller, …



Abstraction, underspecification

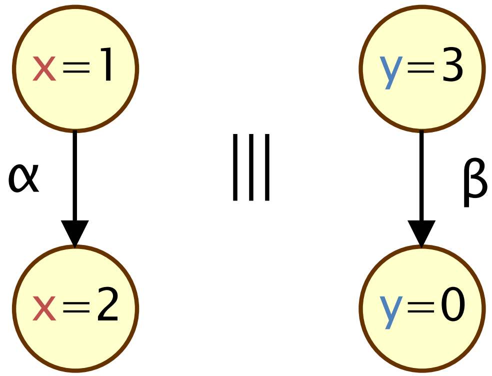
- A simple model of message transmission
  - model abstracted – probability of success is unknown/ignored



Concurrency: Parallel composition

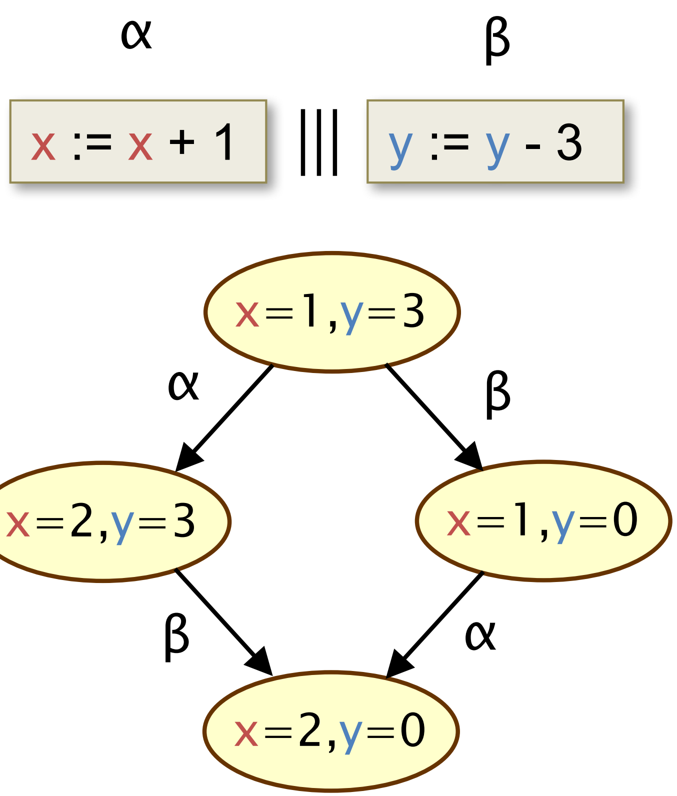
- Consider two asynchronous components in parallel
  - parallel execution of independent actions
  - nondeterminism models interleaving (e.g. of scheduler)

$$\underset{\texttt{x := x + 1}}{\alpha} \quad ||| \quad \underset{\texttt{y := y - 3}}{\beta}$$



Concurrency: Parallel composition

- Consider two asynchronous components in parallel
  - parallel execution of independent actions
  - nondeterminism models interleaving (e.g. of scheduler)



Composition (interleaving) of LTSs

- A formal definition… Let M_1 and M_2 be two LTSs:
  - $M_i = (S_i, Act_i, \rightarrow_i, I_i, AP_i, L_i)$ for $i=1,2$

- Then we define their interleaving $M_1 \mid\mid\mid M_2$ as the LTS:
  - $M_1 \mid\mid\mid M_2 = (S_1 \times S_2, Act_1 \cup Act_2, \rightarrow, I_1 \times I_2, AP_1 \cup AP_2, L)$
- where:
  - $L((s_1,s_2)) = L(s_1) \cup L(s_2)$ for any $s_1 \in S_1$, $s_2 \in S_2$
- and $\rightarrow$ is defined as follows:

$$\frac{s_1 \ -\alpha\rightarrow_1 \ s_1'}{(s_1,s_2) \ -\alpha\rightarrow \ (s_1',s_2)} \qquad \frac{s_2 \ -\alpha\rightarrow_2 \ s_2'}{(s_1,s_2) \ -\alpha\rightarrow \ (s_1,s_2')}$$

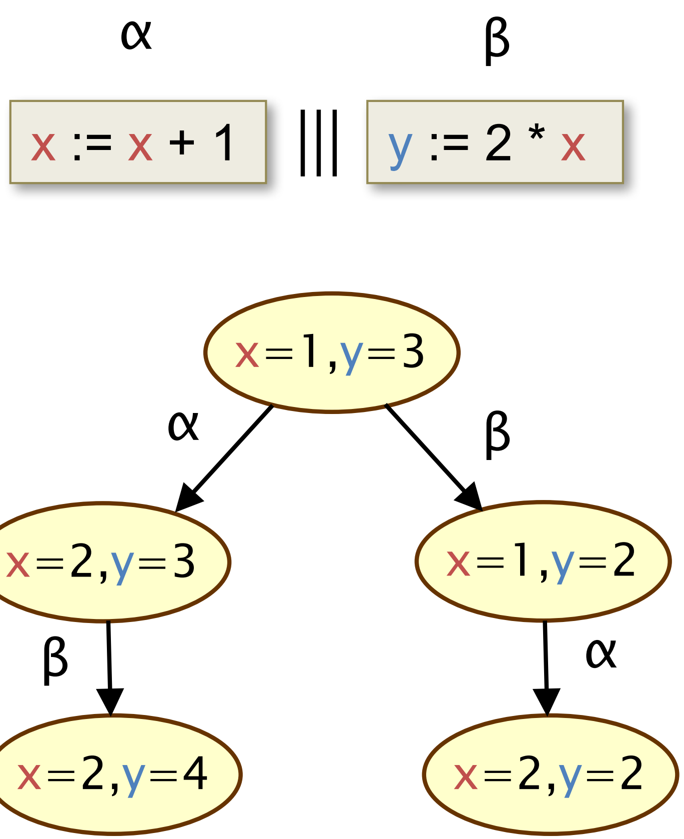
premise

conclusion

SOS rules (structured operational semantics)

Concurrency: Shared variables

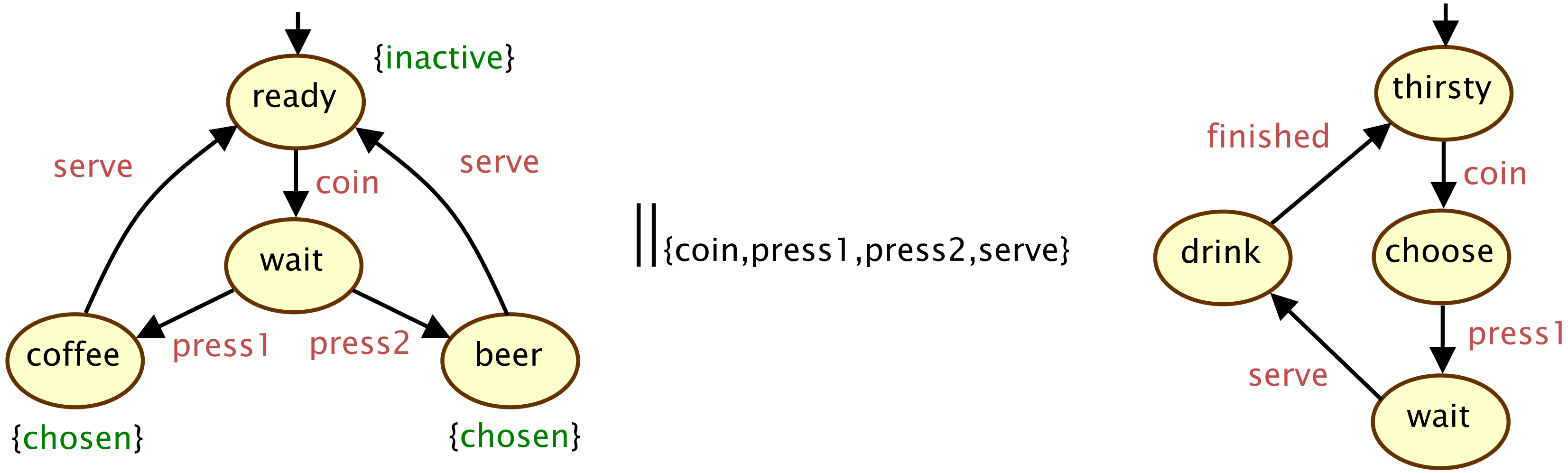
- Consider again two components in parallel
  - but with shared access to some variable
  - nondeterminism models competition between components



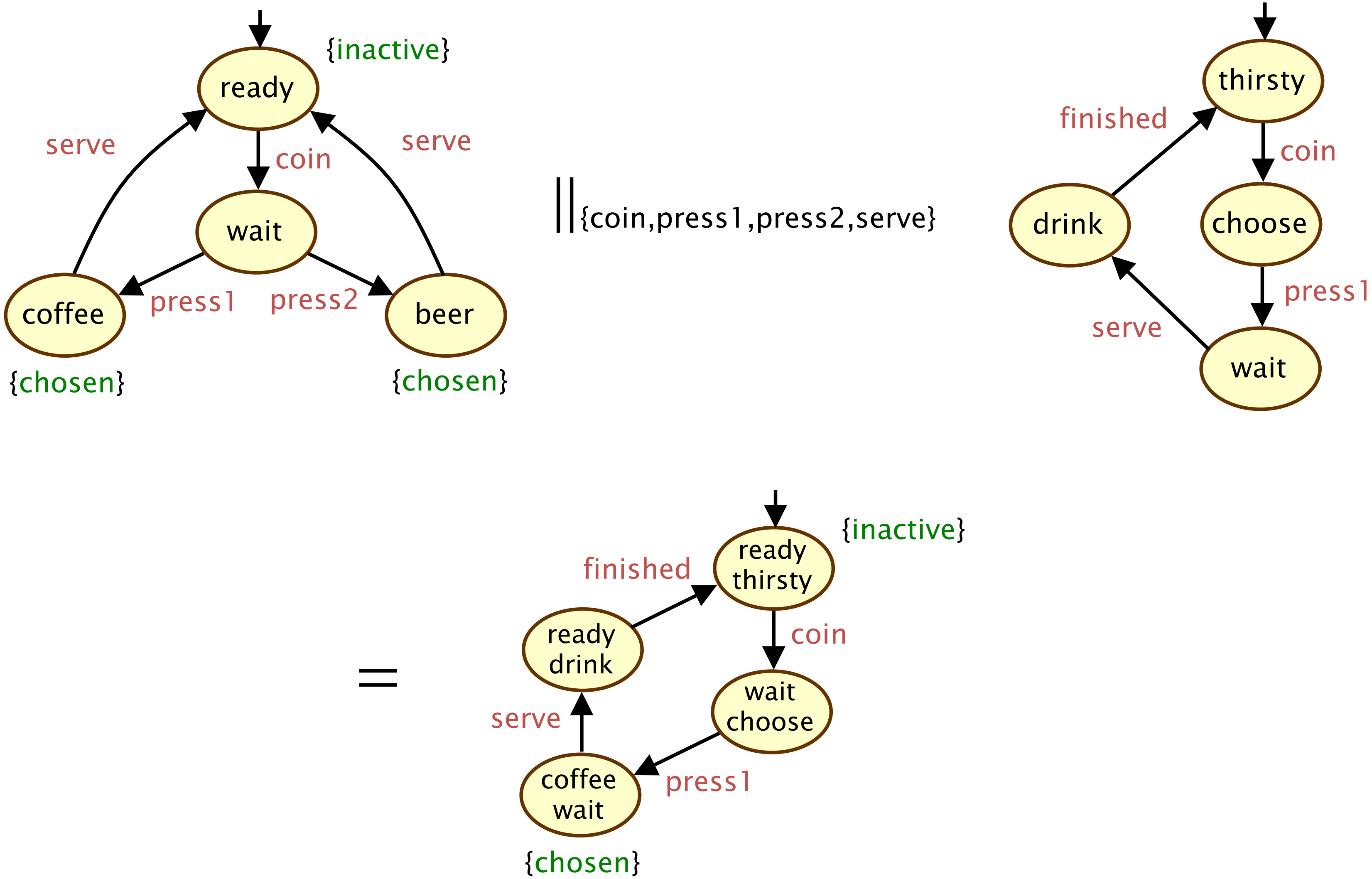
Concurrency: Synchronisation

- Synchronisation between parallel components: handshaking
  - parallel composition $M_1 \,||_H\, M_2$ of LTSs M_1 and M_2
  - for a set $H \subseteq Act$ of handshake actions
  - synchronise (only) on actions in H
  - (like synchronous message passing but message itself is ignored)

- Example



Example



{inactive}
ready
serve
coin
serve
wait
coffee
press1
press2
beer
{chosen}
{chosen}
||{coin,press1,press2,serve}
thirsty
finished
coin
drink
choose
press1
serve
wait
=
{inactive}
ready
thirsty
finished
ready
drink
coin
wait
choose
serve
coffee
wait
press1
{chosen}

Concurrency: Synchronisation

- A formal definition of handshaking:

- $M_1 \;||_H\; M_2$ is defined as for $M_1 \;|||\; M_2$
  - $M_1 \;||_H\; M_2 = (S_1 \times S_2,\ Act_1 \cup Act_2,\ \rightarrow,\ I_1 \times I_2,\ AP_1 \cup AP_2,\ L)$
- except that $\rightarrow$ is defined as follows:

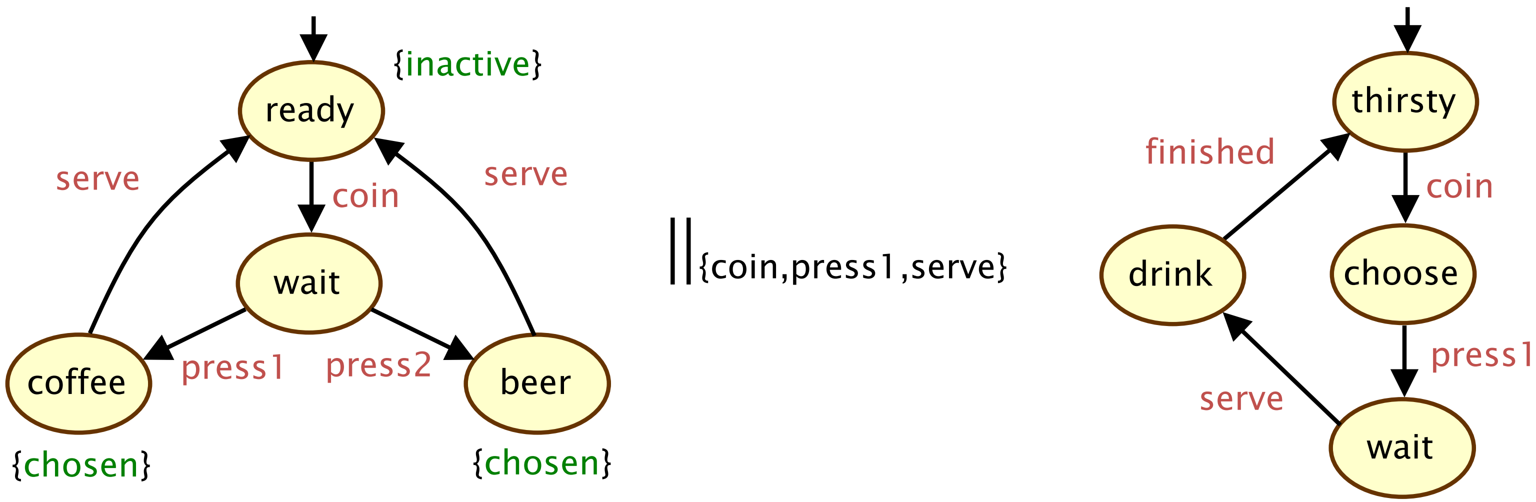
$$\frac{s_1 \xrightarrow{\alpha}_1 s_1' \wedge s_2 \xrightarrow{\alpha}_2 s_2'}{(s_1,s_2) \xrightarrow{\alpha} (s_1',s_2')}\ \alpha \in H$$

$$\frac{s_1 \xrightarrow{\alpha}_1 s_1'}{(s_1,s_2) \xrightarrow{\alpha} (s_1',s_2)}\ \alpha \notin H$$

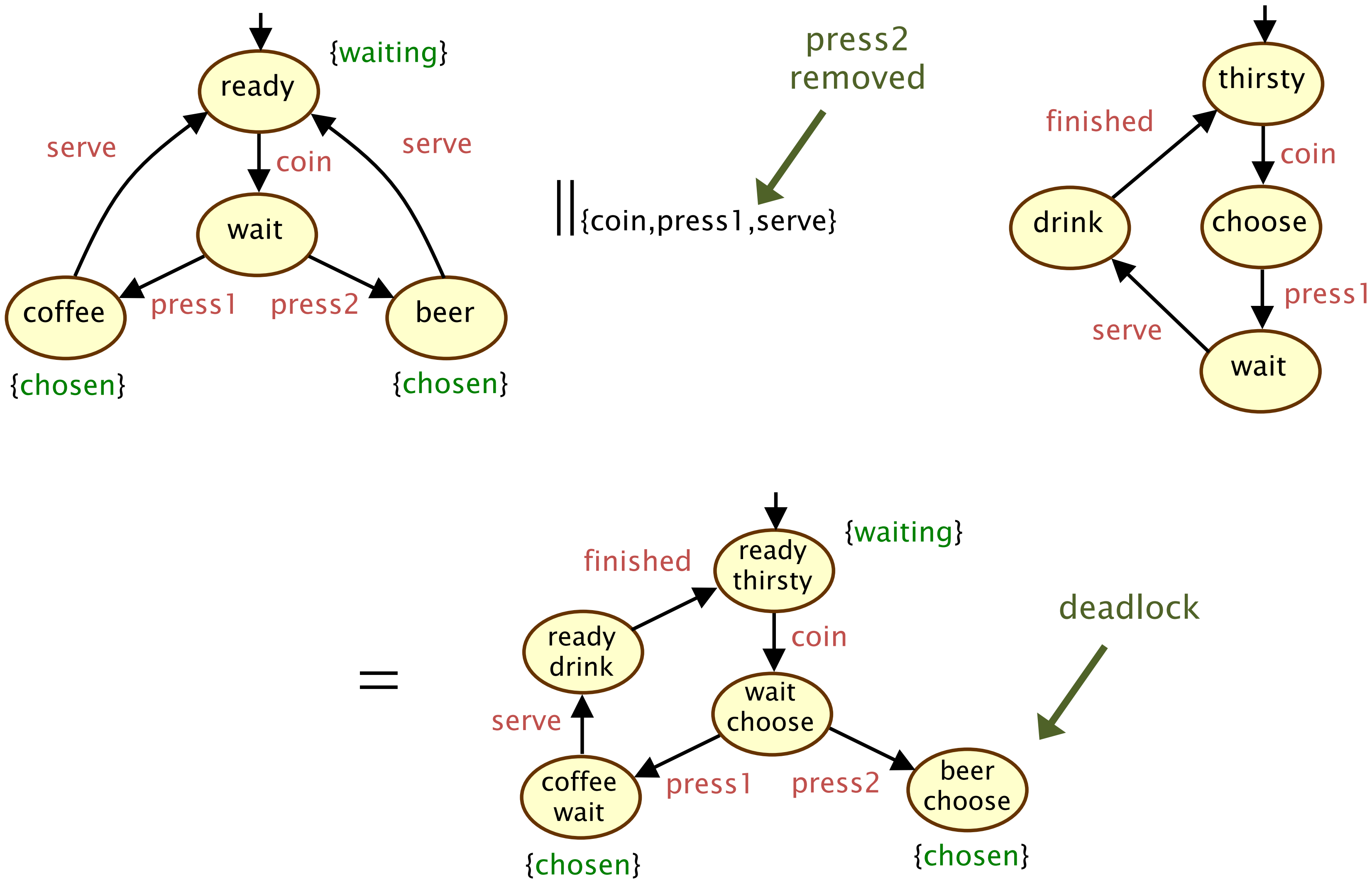
$$\frac{s_2 \xrightarrow{\alpha}_2 s_2'}{(s_1,s_2) \xrightarrow{\alpha} (s_1,s_2')}\ \alpha \notin H$$

Example

- What about this one?
  - (only the handshake set changed)



Another example



Recall the earlier example (Lec 1)

- Does variable x always remain in the range {0,1,...,200}?

process Inc = **while** true **do** **if** x < 200 **then** x := x + 1 **od**

process Dec = **while** true **do** **if** x > 0 **then** x := x − 1 **od**

process Reset = **while** true **do** **if** x = 200 **then** x := 0 **od**

```
....................
pan: assertion violated ((x >= 0) && (x <= 200)) (at depth 1802)
pan: wrote pan_in.trail
....................
State-vector 32 byte, depth reached 3598, errors: 1
      12609 states, stored
```

Parallel composition – Key ideas

- Nondeterminism models interleaving of parallel components
  - i.e. unknown execution order (or unknown scheduling)

- Parallel composition of two LTSs
  - resulting LTS has product state space $S_1 \times S_2$
  - i.e., each state has the state for both (or all) components