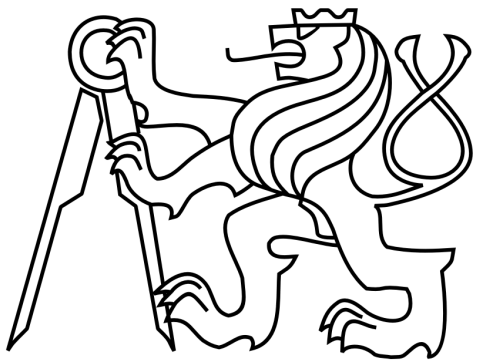
Interacting System Components

Stefan Ratschan

Katedra číslicového návrhu
Fakulta informačních technologií
České vysoké učení technické v Praze



Evropský sociální fond Praha & EU: Investujeme do vaší budoucnosti

Big Question

How can we describe and computationally analyze complex systems?

Previous lecture

Systems consist of communicating sub-systems

Ways of describing such sub-systems (discrete-time system, automaton)

Today:

How can we describe interaction between sub-systems?

Black-box Description of Systems

A *(discrete time) system* with input set I and output set O is
a relation between signals over I and signals over O, that is
a subset of $\Sigma_I \times \Sigma_O$

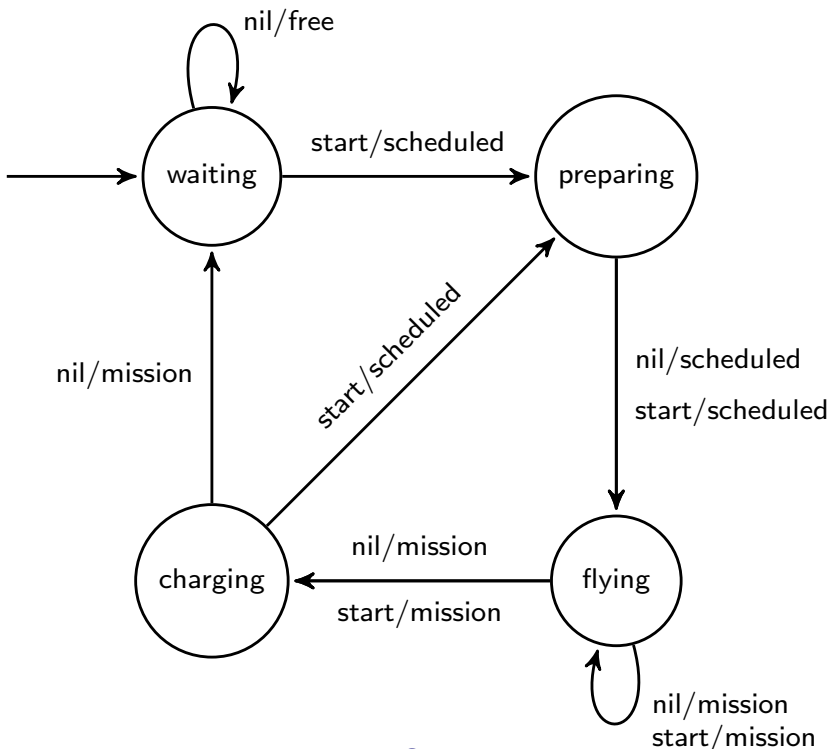
Quadrocopter:

$$\{(i, o) \in \Sigma_{\{nil, start\}} \times \Sigma_{\{free, scheduled, mission\}} \mid \\ \forall t \in \mathbb{N}_0 \,.\, i(t) = start \Rightarrow \exists d \in \{0, \ldots, 10\} \,.\, o(t + d) = mission\}$$

Discrete Time Automata

A *(discrete time) automaton* is a quintuple (S, S_0, I, O, R), where

- S is a set (*state space*) whose elements we call *states*
- $S_0 \subseteq S$ (set of *initial states*)
- I is a set whose elements we call *inputs*
- O is a set whose element we call *outputs*
- $R \subseteq I \times S \times S \times O$ (*transition relation*) s.t.
  for all $i \in I$, $s \in S$, there is $s' \in S$, $o \in O$ s.t. $(i, s, s', o) \in R$



Combination of Systems

One component: quadrocopter

Further component: switching on/off light

$$\left\{ (i, o) \mid \forall t \,. \begin{array}{l} i(t) = \text{switch-off} \Rightarrow o(t) = \text{off} \wedge \\ i(t) = \text{switch-on} \Rightarrow o(t) = \text{on} \wedge \\ [i(t) = \text{stay} \wedge t > 0] \Rightarrow o(t) = o(t-1) \end{array} \right\}$$

System that behaves like two components running in parallel

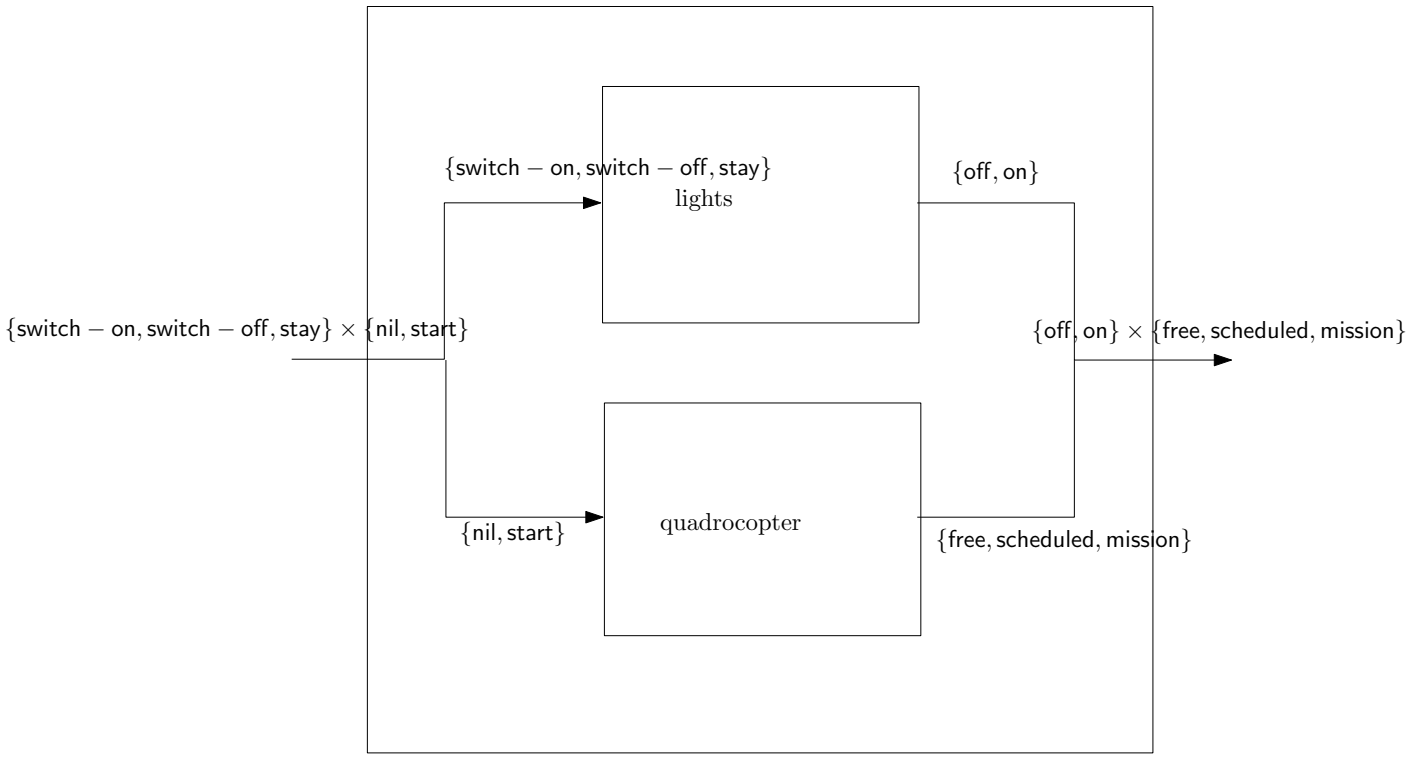
Synchronous Parallel Composition

Synchronous:

Systems always execute one step together.

Synchronous Parallel Composition of Systems

First: composition of black-boxes



For example,

for input ((stay, nil), (switch-on, start), (stay, nil), (stay, nil), . . .),
output ((off, free), (on, scheduled), (on, scheduled), (on, mission), . . .)

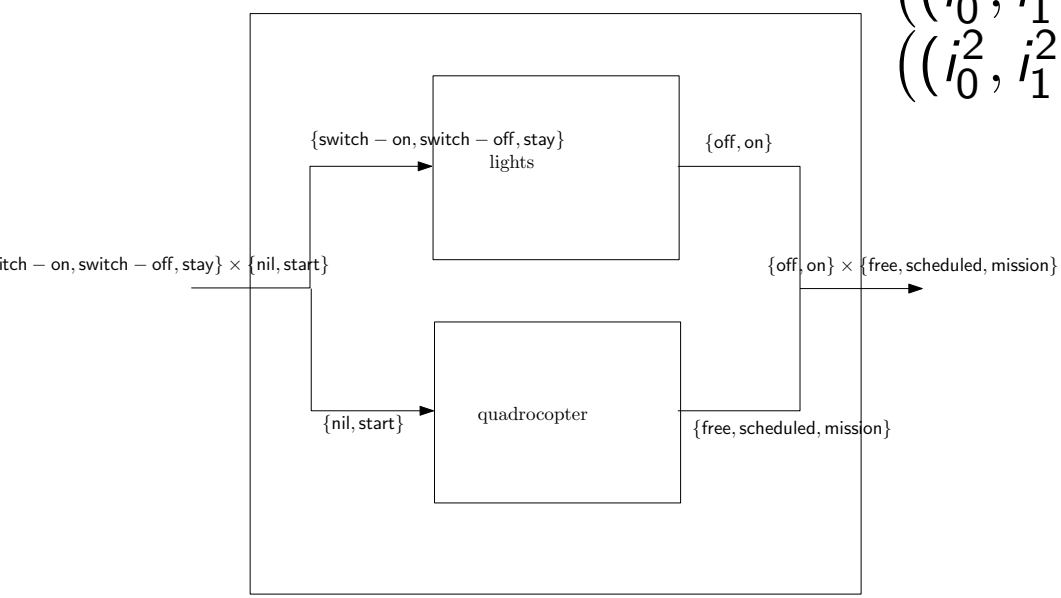
Synchronous Parallel Composition of Systems

Given: discrete-time systems $\mathcal{S}^1, \mathcal{S}^2$ with
input set I^1, I^2, and output set O^1, O^2, respectively.

Result: System $\mathcal{S}^1 \otimes \mathcal{S}^2$, input set $I^1 \times I^2$, output set $O^1 \times O^2$ where

$$\mathcal{S}^1 \otimes \mathcal{S}^2 :=$$

$$\left\{ \left(\left((i_0^1, i_0^2), (i_1^1, i_1^2), (i_2^1, i_2^2), \dots \right), \left((o_0^1, o_0^2), (o_1^1, o_1^2), (o_2^1, o_2^2) \dots \right) \right) \mid \begin{array}{l} \left((i_0^1, i_1^1, i_2^1 \dots), (o_0^1, o_1^1, o_2^1 \dots) \right) \in \mathcal{S}^1 \\ \left((i_0^2, i_1^2, i_2^2 \dots), (o_0^2, o_1^2, o_2^2 \dots) \right) \in \mathcal{S}^2 \end{array} \right\}$$



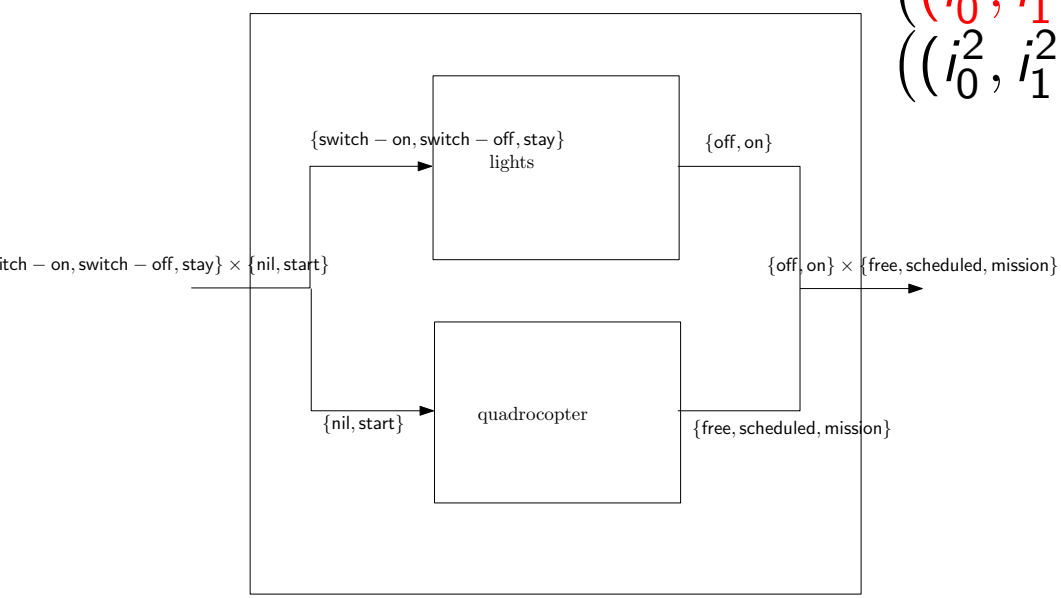
Synchronous Parallel Composition of Systems

Given: discrete-time systems $\mathcal{S}^1, \mathcal{S}^2$ with
input set I^1, I^2, and output set O^1, O^2, respectively.

Result: System $\mathcal{S}^1 \otimes \mathcal{S}^2$, input set $I^1 \times I^2$, output set $O^1 \times O^2$ where

$$\mathcal{S}^1 \otimes \mathcal{S}^2 :=$$

$$\left\{ \left(\left((i_0^1, i_0^2), (i_1^1, i_1^2), (i_2^1, i_2^2), \dots \right), \left((o_0^1, o_0^2), (o_1^1, o_1^2), (o_2^1, o_2^2) \dots \right) \right) \mid \begin{array}{l} \left((i_0^1, i_1^1, i_2^1 \dots), (o_0^1, o_1^1, o_2^1 \dots) \right) \in \mathcal{S}^1 \\ \left((i_0^2, i_1^2, i_2^2 \dots), (o_0^2, o_1^2, o_2^2 \dots) \right) \in \mathcal{S}^2 \end{array} \right\}$$



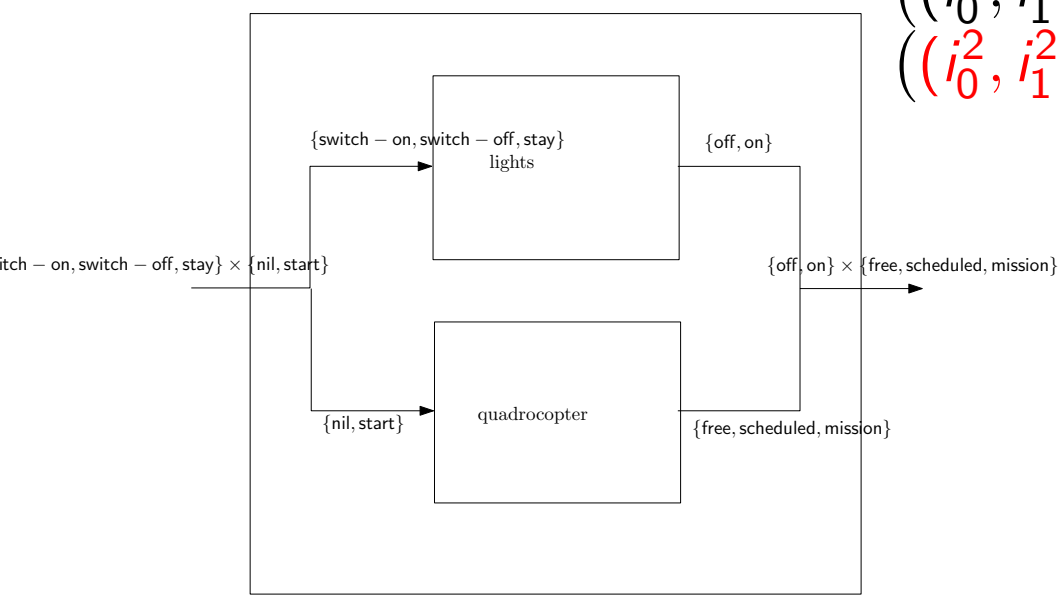
Synchronous Parallel Composition of Systems

Given: discrete-time systems $\mathcal{S}^1, \mathcal{S}^2$ with
input set I^1, I^2, and output set O^1, O^2, respectively.

Result: System $\mathcal{S}^1 \otimes \mathcal{S}^2$, input set $I^1 \times I^2$, output set $O^1 \times O^2$ where

$$\mathcal{S}^1 \otimes \mathcal{S}^2 :=$$

$$\left\{ \left(\left((i_0^1, i_0^2), (i_1^1, i_1^2), (i_2^1, i_2^2), \dots \right), \left((o_0^1, o_0^2), (o_1^1, o_1^2), (o_2^1, o_2^2) \dots \right) \right) \mid \begin{array}{l} \left((i_0^1, i_1^1, i_2^1 \dots), (o_0^1, o_1^1, o_2^1 \dots) \right) \in \mathcal{S}^1 \\ \left((i_0^2, i_1^2, i_2^2 \dots), (o_0^2, o_1^2, o_2^2 \dots) \right) \in \mathcal{S}^2 \end{array} \right\}$$

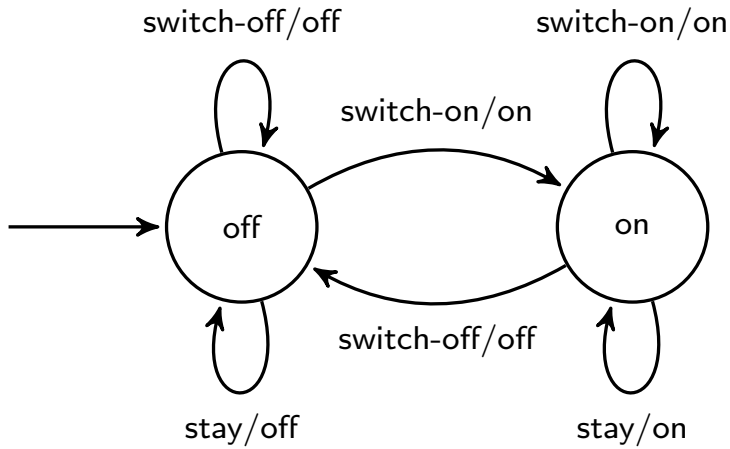


Combination of Automata

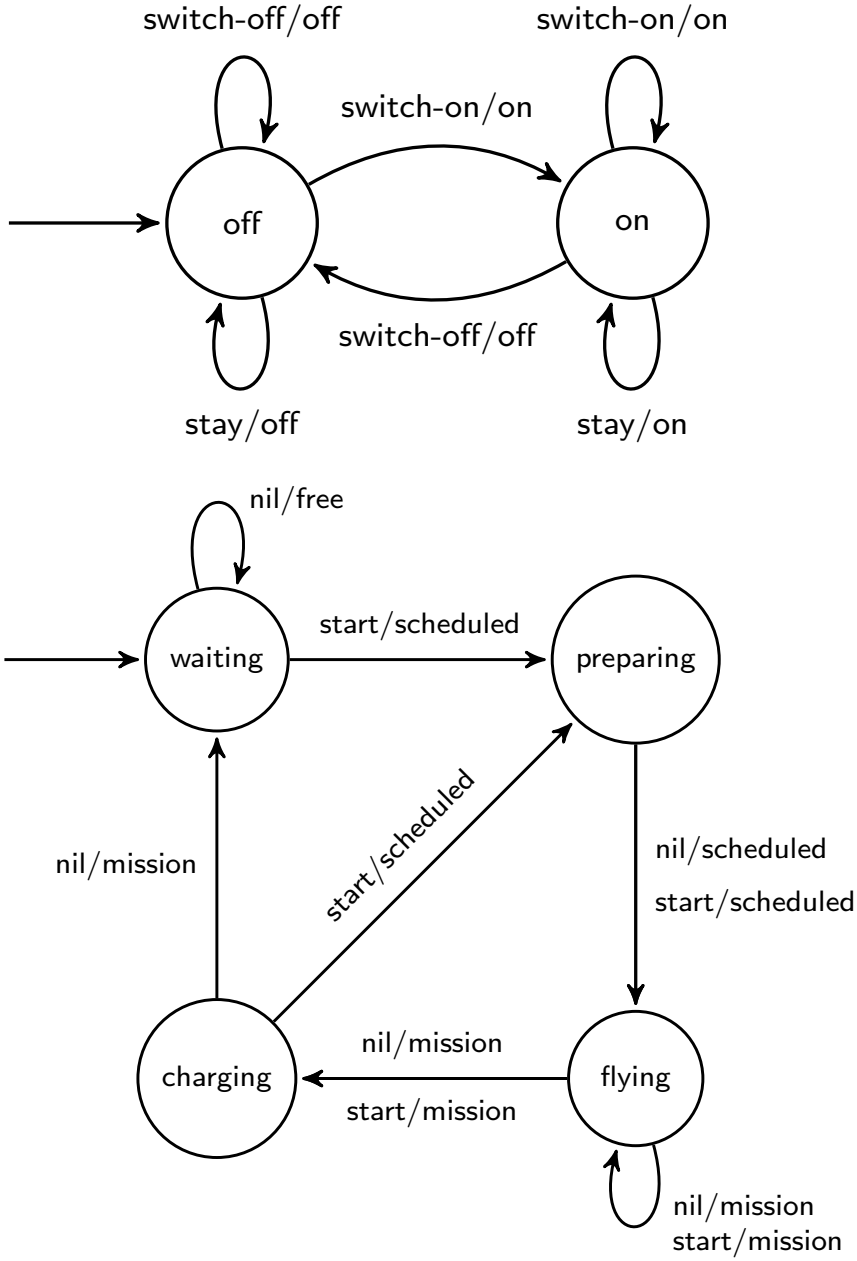
Light as a discrete-time system:

$$\left\{ (i, o) \mid \forall t \; . \; \begin{array}{r} i(t) = \text{switch-off} \Rightarrow o(t) = \text{off} \wedge \\ i(t) = \text{switch-on} \Rightarrow o(t) = \text{on} \wedge \\ [i(t) = \text{stay} \wedge t > 0] \Rightarrow o(t) = o(t-1) \end{array} \right\}$$

Implementation of light

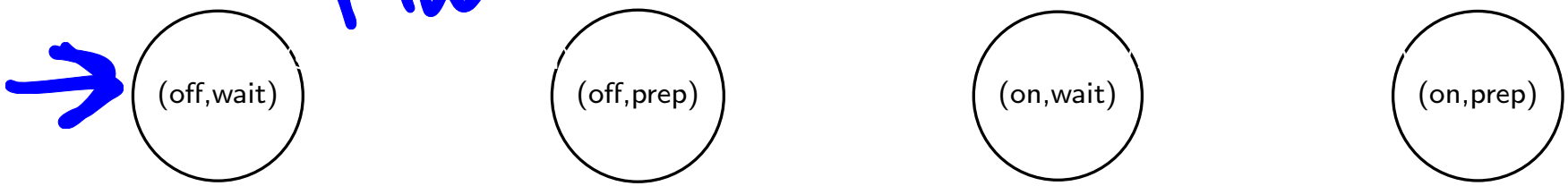


Synchronous Parallel Composition of Automata



Synchronous Parallel Composition of Automata: Example

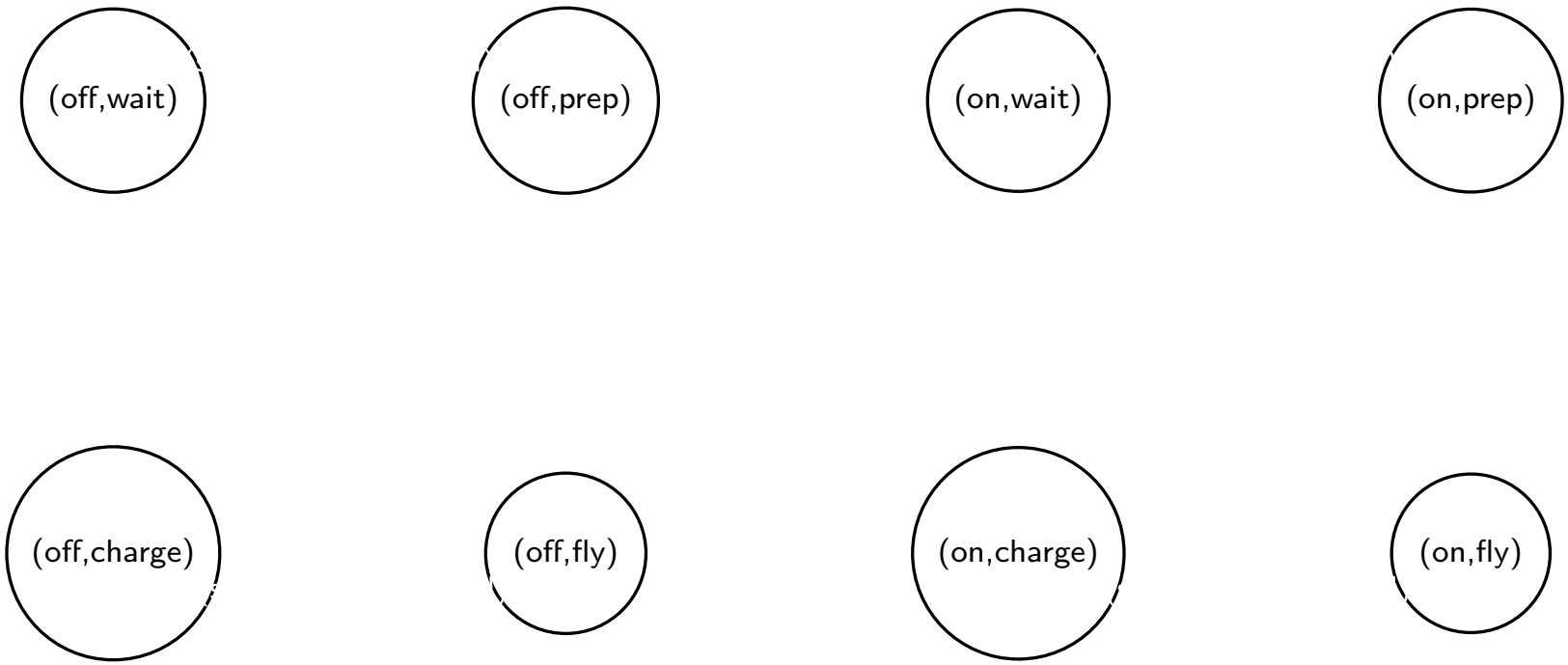
I have to combine all possible states from all systems



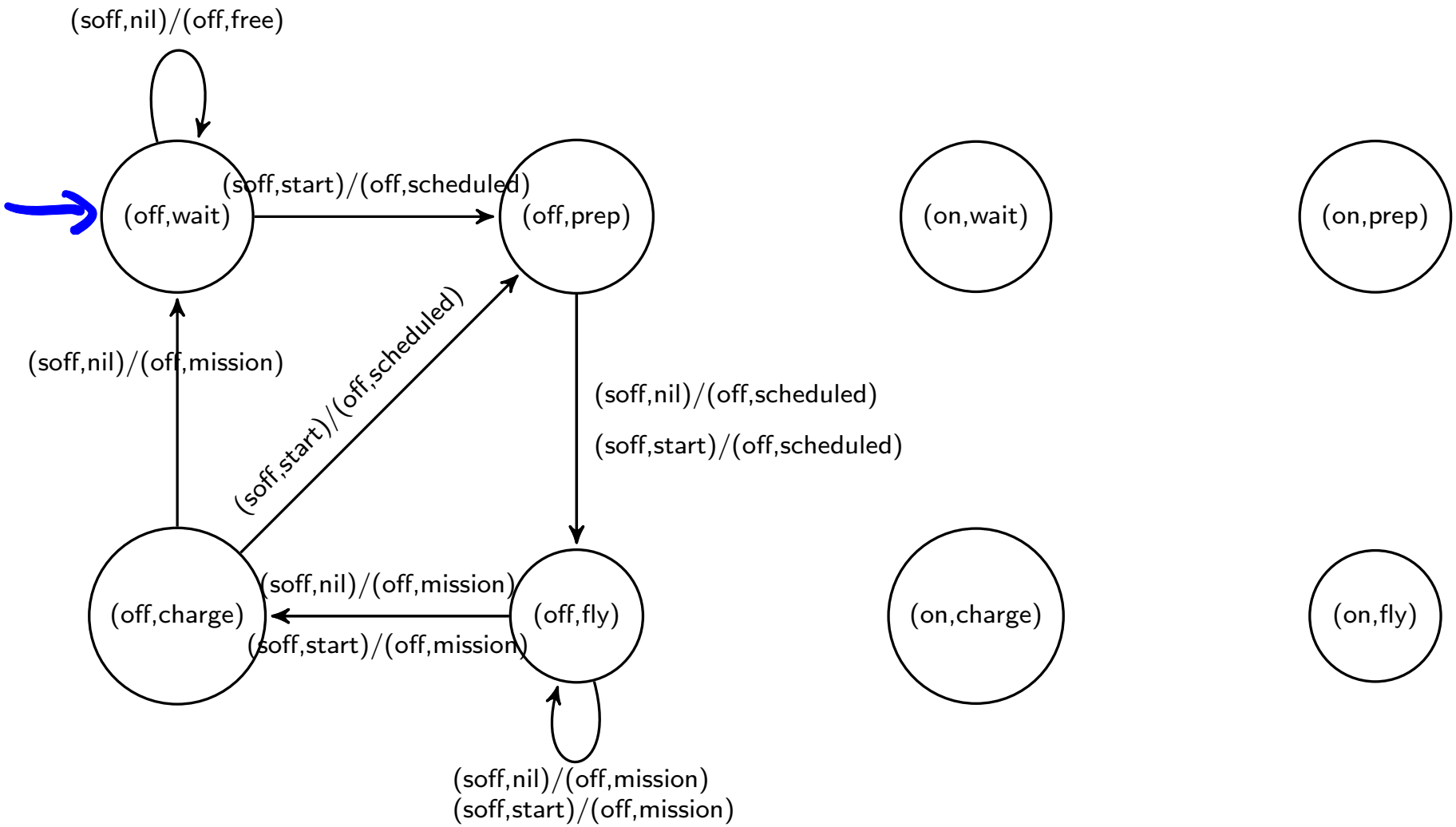
I have to combine all transitions with all possible states

abbreviated names!

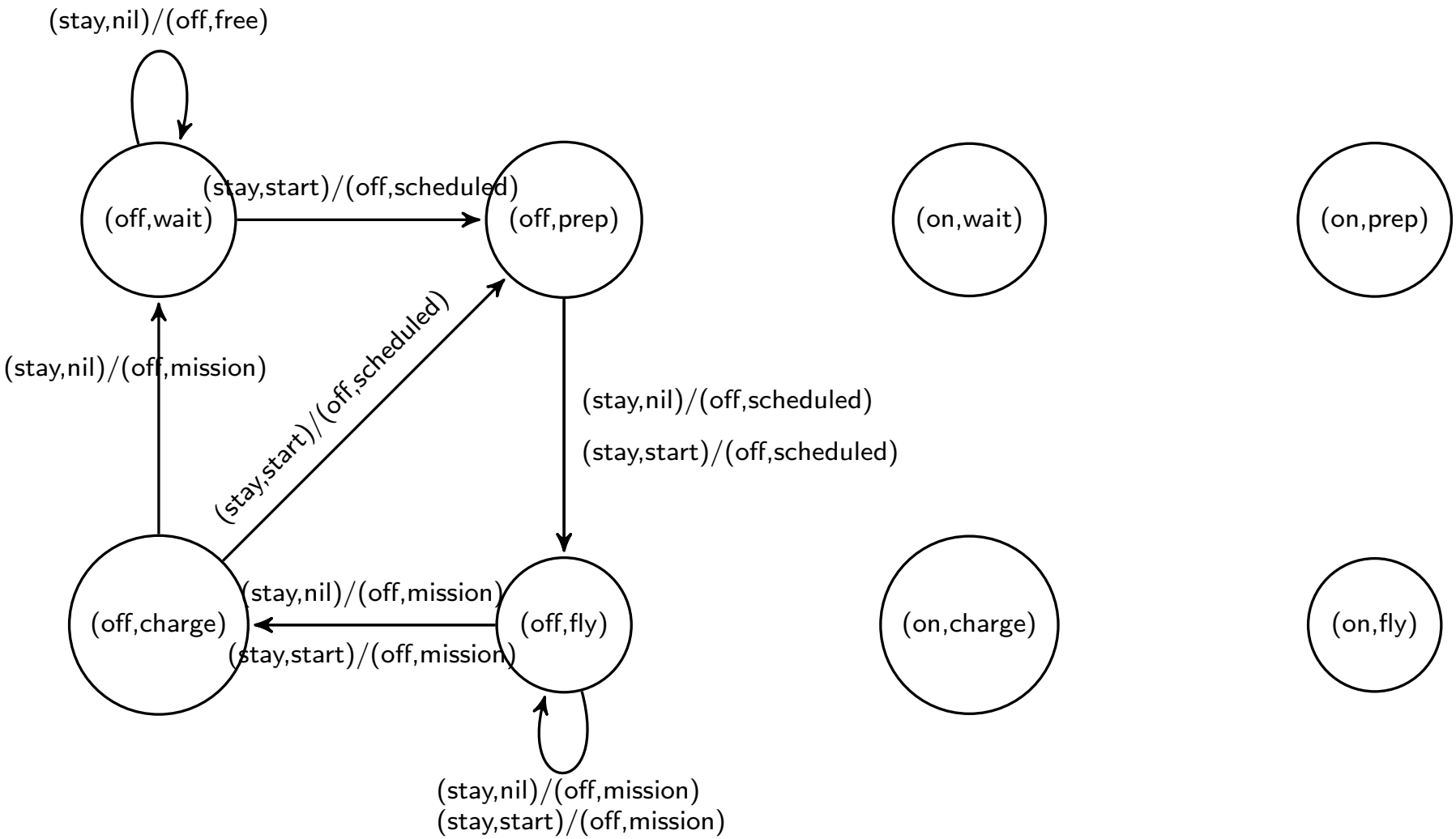
Composition of Automata: Initial State



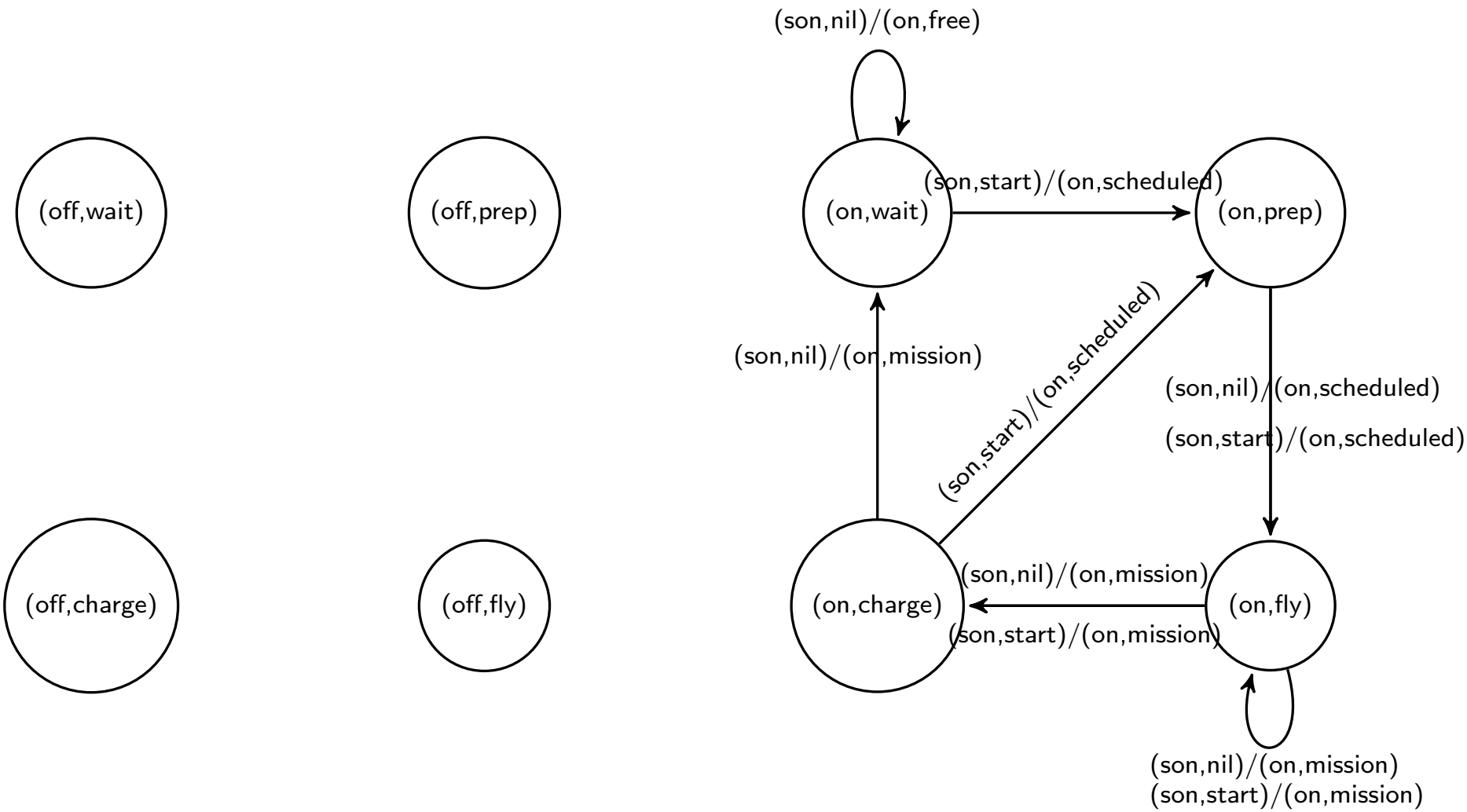
Composition of Automata: Transitions, Part 1



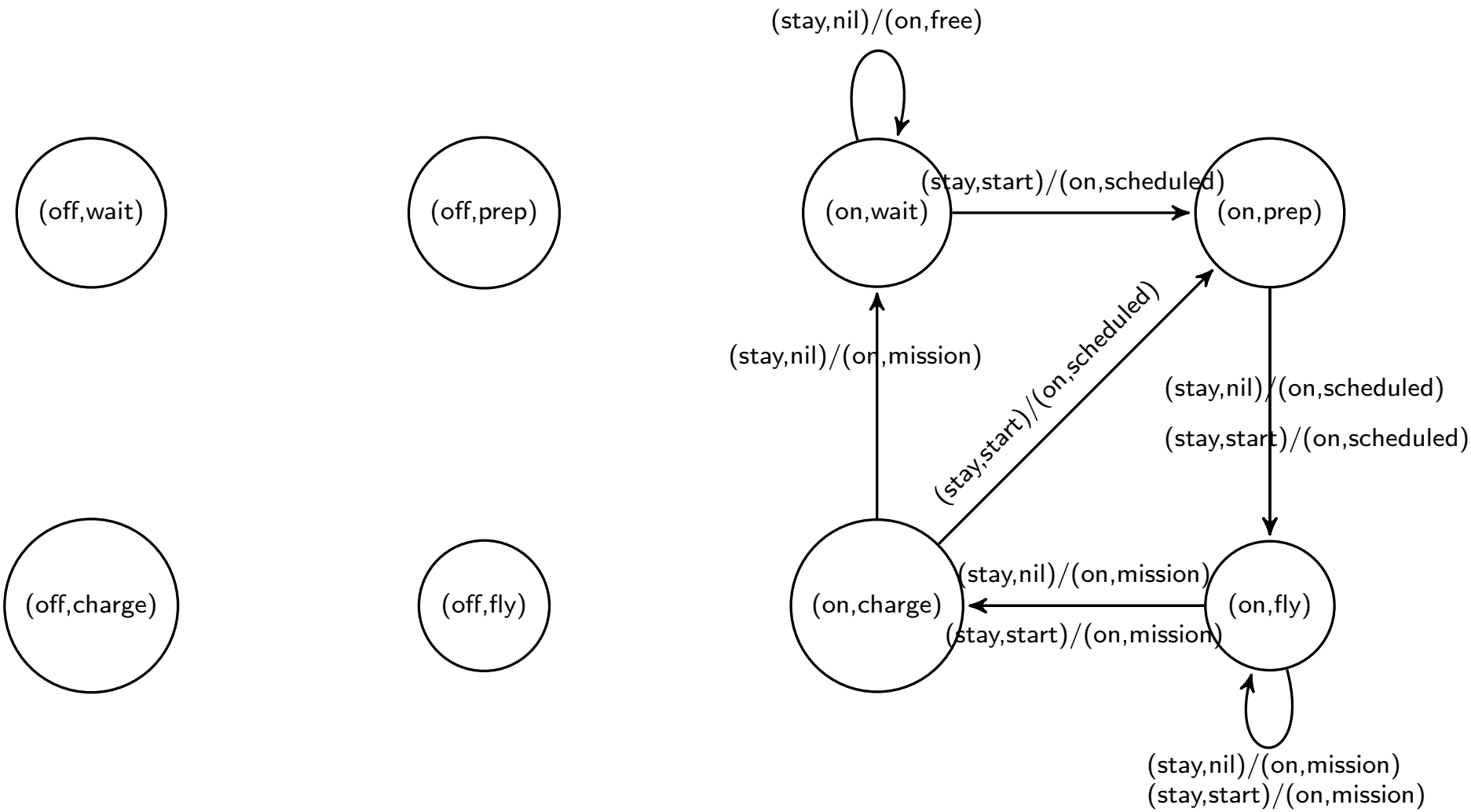
Composition of Automata: Transitions, Part 2



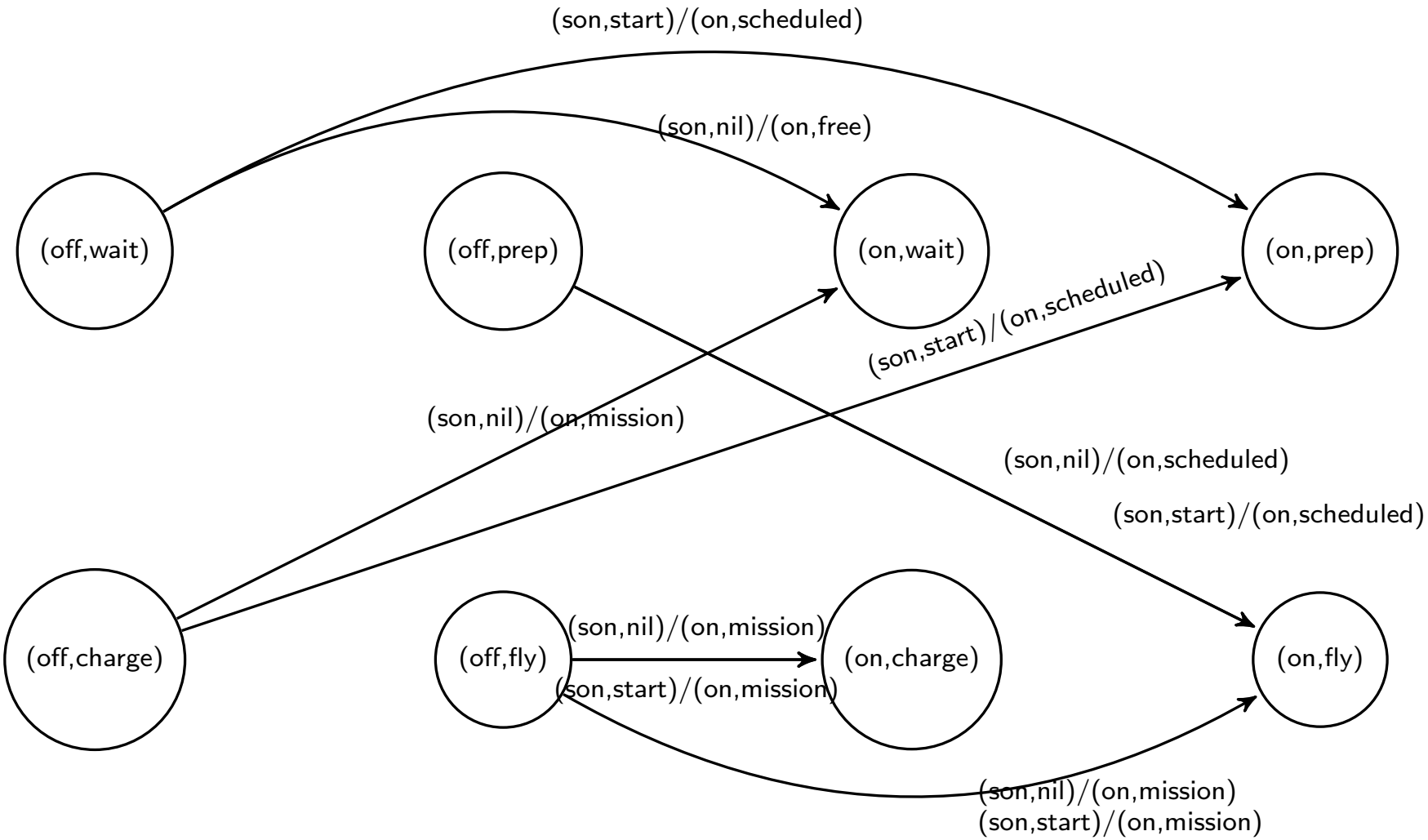
Composition of Automata: Transitions, Part 3



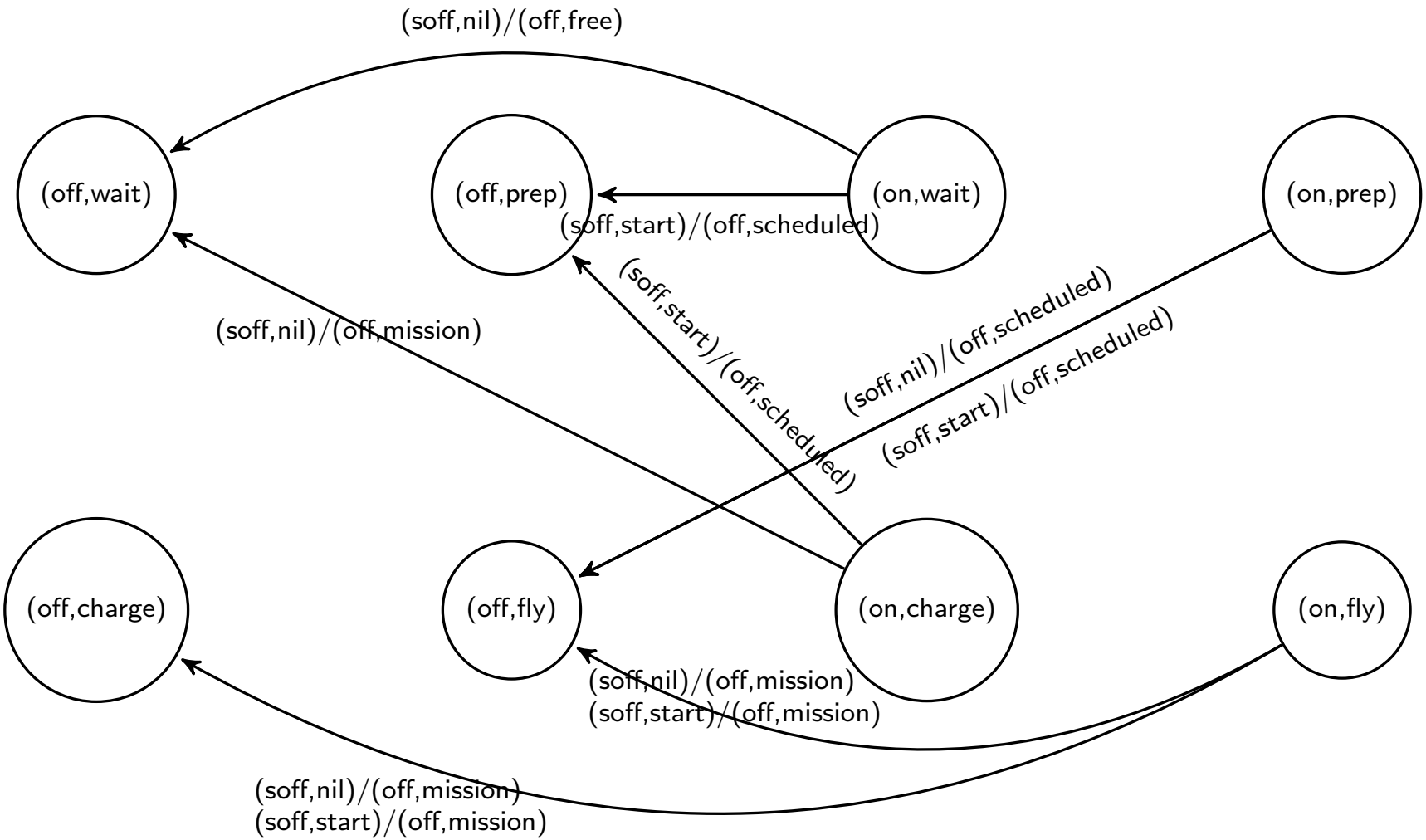
Composition of Automata: Transitions, Part 4



Composition of Automata: Transitions, Part 5



Composition of Automata: Transitions, Part 6



Synchronous Parallel Composition of Automata: Formalization

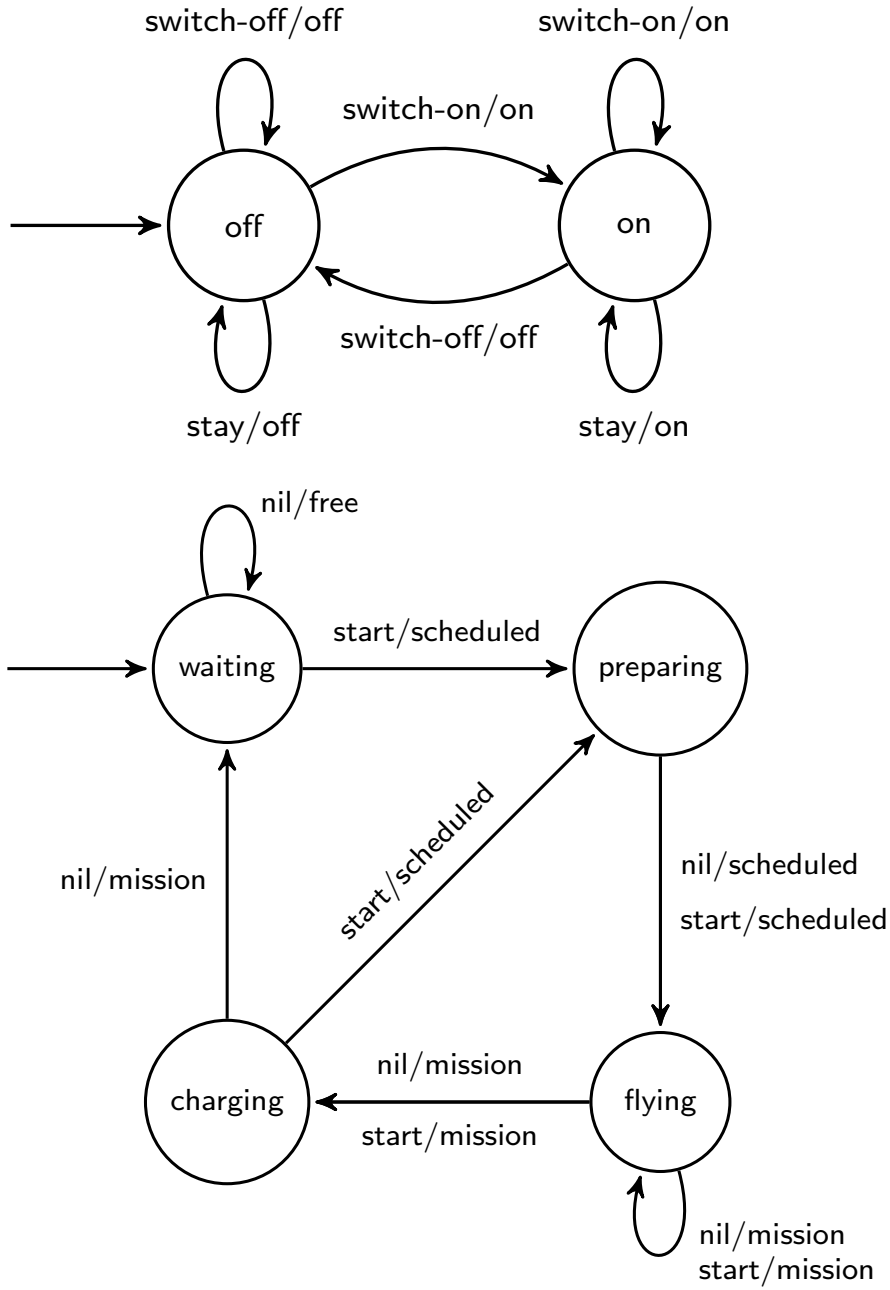
$$(S^1, S_0^1, I^1, O^1, R^1) \otimes (S^2, S_0^2, I^2, O^2, R^2) \doteq$$
$$(S^1 \times S^2, S_0^1 \times S_0^2, I^1 \times I^2, O^1 \times O^2, R^{\otimes}),$$

where

$$R^{\otimes} := \{((i^1, i^2), (s^1, s^2), (s'^1, s'^2), (o^1, o^2)) \mid$$
$$(i^1, s^1, s'^1, o^1) \in R^1, (i^2, s^2, s'^2, o^2) \in R^2\}$$

Can be easily generalized to more than two systems/automata.

Example


switch-off/off
switch-on/on
switch-on/on
off
on
switch-off/off
stay/off
stay/on
nil/free
waiting
start/scheduled
preparing
nil/mission
start/scheduled
nil/scheduled
start/scheduled
charging
nil/mission
start/mission
flying
nil/mission
start/mission

Compatibility of Composition with Represented System

Remember: For an automaton $\mathcal{A}$,
$[\![\mathcal{A}]\!]$ is the system represented by $\mathcal{A}$.

We have: For automata $\mathcal{A}_1$ a $\mathcal{A}_2$,

$$[\![\mathcal{A}_1 \otimes \mathcal{A}_2]\!] = [\![\mathcal{A}_1]\!] \otimes [\![\mathcal{A}_2]\!]$$

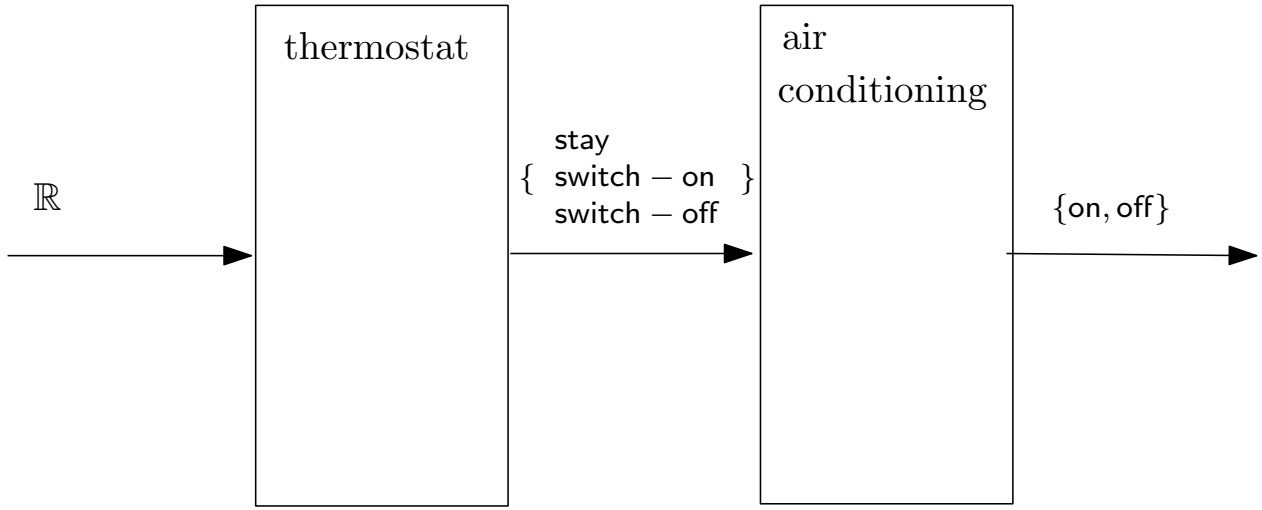
Cascade Composition of Systems

Certain behavior of a component may result in
reaction of other components

Output of the first system: input of second system.

Example: unix pipes: `tail -0f logfile | grep login`

Cascade Composition of Systems



Systems $\mathcal{S}^1, \mathcal{S}^2$ with input set I^1, I^2 and output set O^1, O^2, respectively

Condition: $O^1 \subseteq I^2$

Result: $\mathcal{S}^1 \rightsquigarrow \mathcal{S}^2$ with input set I^1 and output set O^2.

$$\mathcal{S}^1 \rightsquigarrow \mathcal{S}^2 := \{(i_1, o_2) \mid \exists io \,.\, (i_1, io) \in \mathcal{S}^1, (io, o_2) \in \mathcal{S}^2\}$$

Cascade Composition of Automata

Corresponding operation on automata?

Ptolemy demo

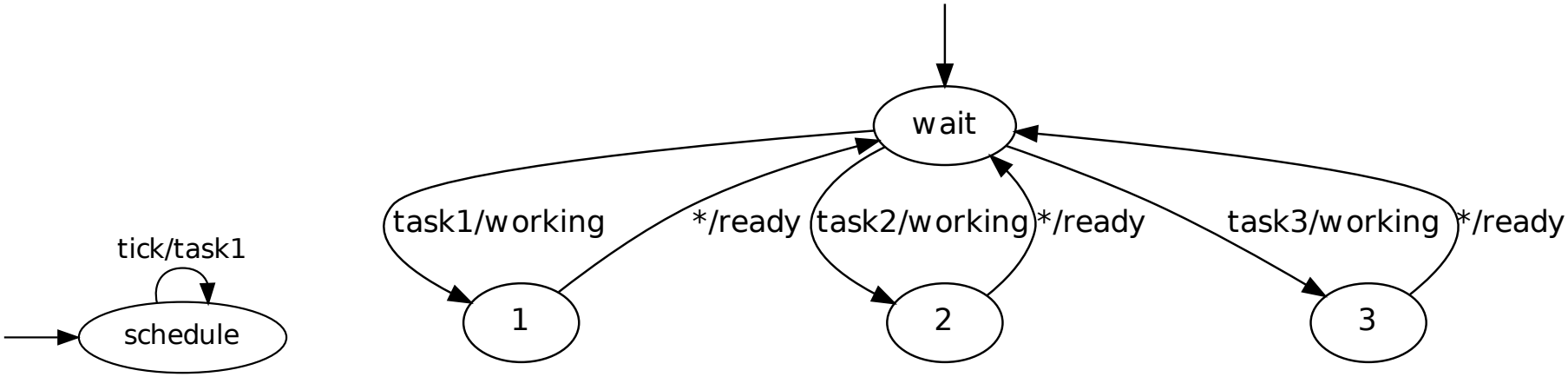
$$(S^1, S_0^1, I^1, O^1, R^1) \rightsquigarrow (S^2, S_0^2, I^2, O^2, R^2) \doteq (S^1 \times S^2, S_0^1 \times S_0^2, I^1, O^2, R^{\rightsquigarrow}),$$

where

$$R^{\rightsquigarrow} := \{(i^1, (s^1, s^2), (s'^1, s'^2), o^2) \mid \exists x \,.\, (i^1, s^1, s'^1, x) \in R^1, (x, s^2, s'^2, o^2) \in R^2\}$$

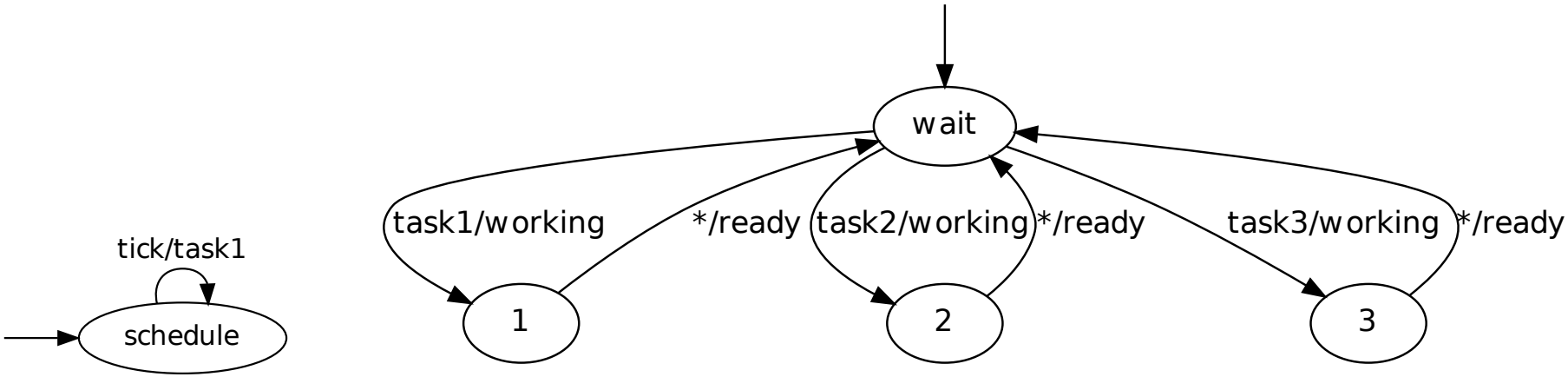
Condition: $O^1 \subseteq I^2$ (so x also in I^2)

Example:



- State space:
  $\{(schedule, wait), (schedule, 1), (schedule, 2), (schedule, 3)\}$
- Initial states: $\{(schedule, wait)\}$
- Input states: $\{tick\}$
- Output states: $\{working, ready\}$
- Transition relation: ...

Example Continued



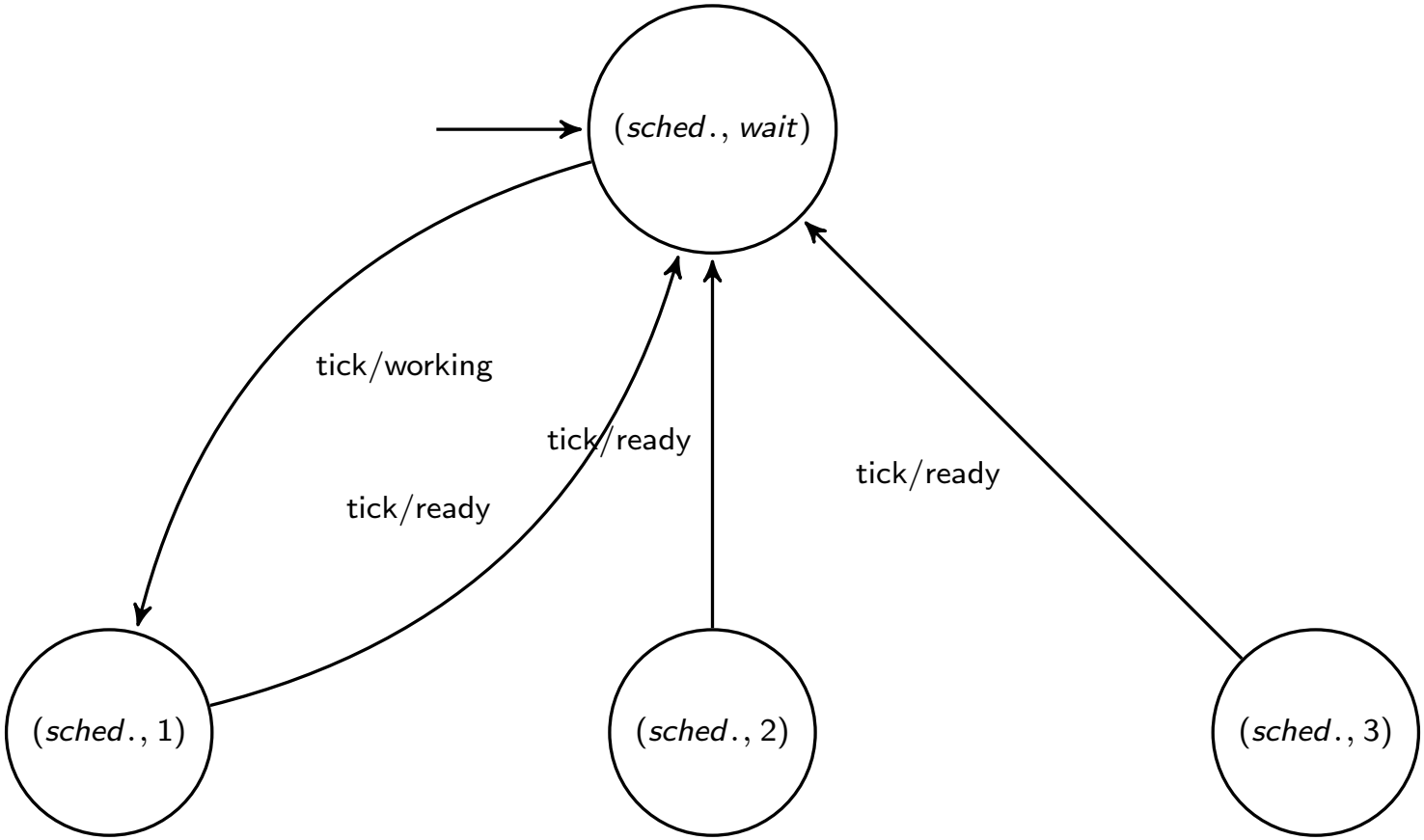
$$R^{\rightsquigarrow} := \{(i^1, (s^1, s^2), (s'^1, s'^2), o^2) \mid \exists x \,.\, (i^1, s^1, s'^1, x) \in R^1, (x, s^2, s'^2, o^2) \in R^2\}$$

$$R_1 = \{(tick, schedule, schedule, task1)\}$$

$$R_2 = \left\{ \begin{array}{l} (task1, wait, 1, working), (*, 1, wait, ready), \\ (task2, wait, 2, working), (*, 2, wait, ready), \\ (task3, wait, 3, working), (*, 3, wait, ready) \end{array} \right\}$$

$$\{(tick, (schedule, wait), (schedule, 1), working),$$

$$(tick, (schedule, 1), (schedule, wait), ready), \ldots$$



Unreachable states, simplification

Composition vs Represented Systems

Again compatible:

$$[\![\mathcal{A}_1 \rightsquigarrow \mathcal{A}_2]\!] = [\![\mathcal{A}_1]\!] \rightsquigarrow [\![\mathcal{A}_2]\!]$$

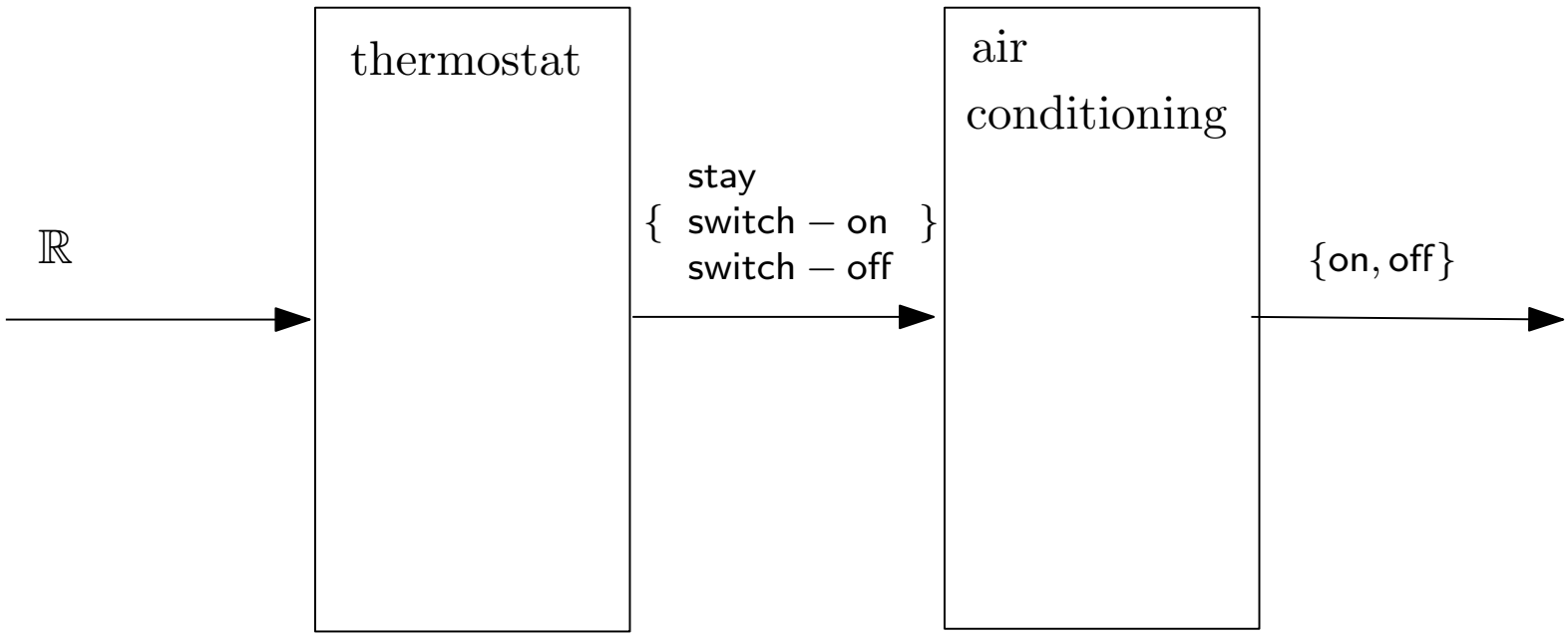
Preliminary Summary

	systems	automata
Synchronous Parallel Composition		
Cascade Composition		

Why both systems and automata?

- Systems: more general
- Automata: easy to compute with

General Composition—Synchronous Reactive Models

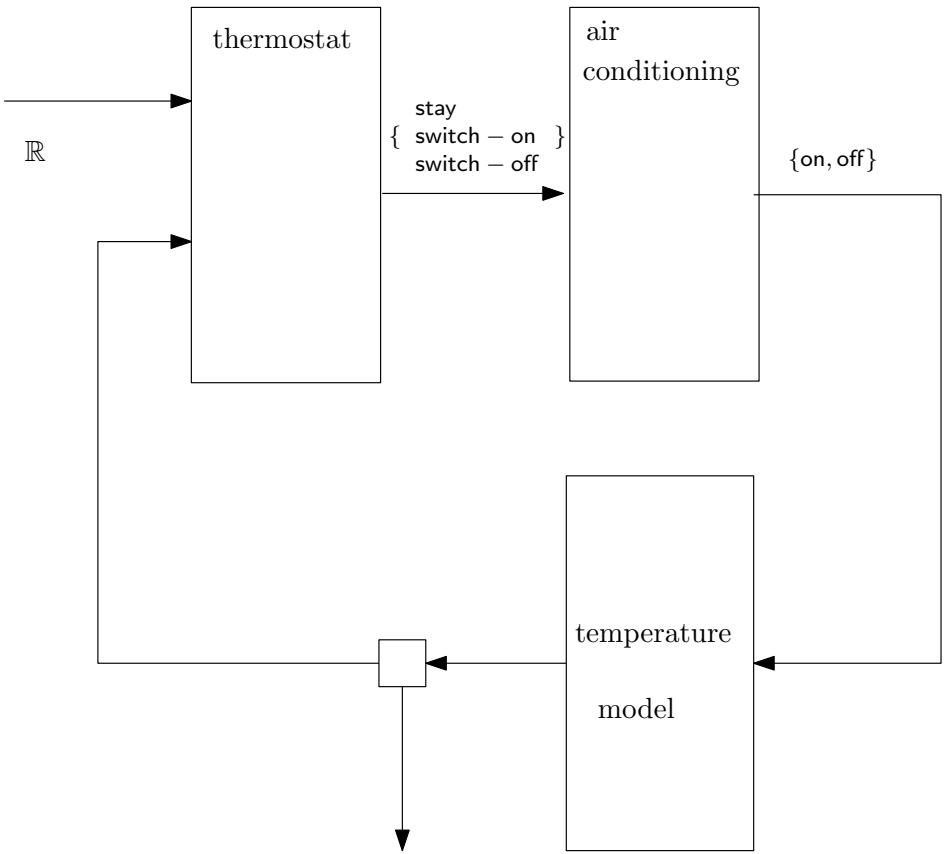


airconditioning again influences room temperature!

feedback loop?

temperature model needed

General Composition—Synchronous Reactive Models



Arbitrary connection of inputs and outputs, especially:

- loops
- several inputs and outputs
- flow control

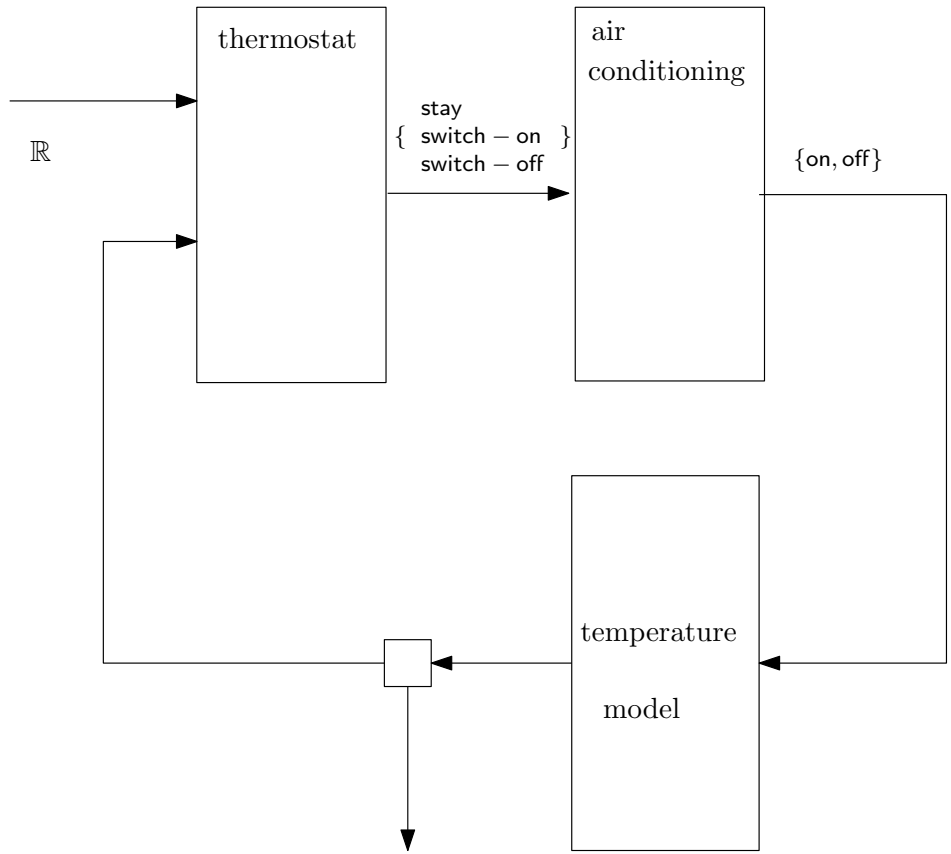
Systems with several inputs/outputs: $S \subseteq \Sigma_{I_1} \times \cdots \times \Sigma_{I_r} \times \Sigma_{O_1} \times \cdots \times \Sigma_{O_s}$

Tee: $\{(i, o_1, o_2) \mid o_1 = i, o_2 = i\}$

Semantics of General Composition: Example

Network with

- one input and one output
- four components
- four connections between components



The network represents a system $N \subseteq \Sigma_{\mathbb{R}} \times \Sigma_{\mathbb{R}}$, where $(i, o) \in N$ iff there are signals (s_1, s_2, s_3, s_4) s.t. $(i, s_4, s_1) \in$ thermostat, $(s_1, s_2) \in$ air conditioning, $(s_2, s_3) \in$ temperature model, $(s_3, s_4, o) \in$ tee

Semantics of General Composition

Given: network of components N, network has r inputs, and s outputs.

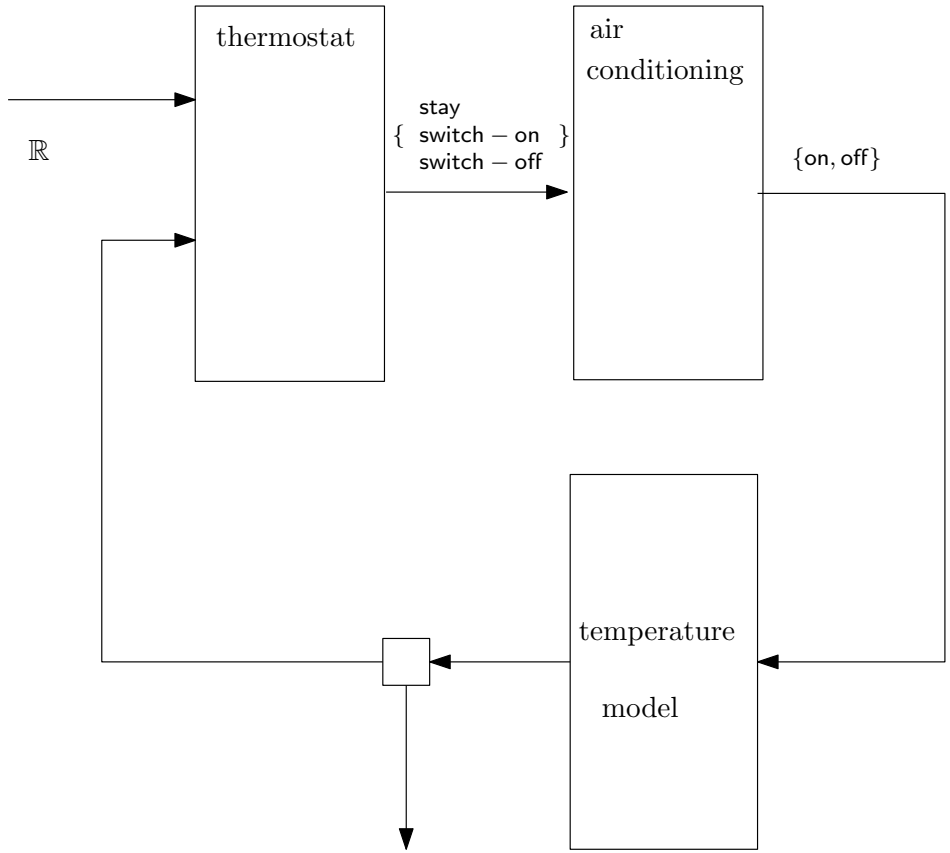
Then: $(i_1, \ldots, i_r, o_1, \ldots, o_s) \in N$ iff
there exist a corresponding signal for
every connection between two components of the network s.t.

for every component S with corresponding
input signals $(i_1^C, \ldots, i_{rC}^C)$ and output signals $(o_1^C, \ldots, o_{rC}^C)$,
$(i_1^C, \ldots, i_{rC}^C, o_1^C, \ldots, o_{sC}^C) \in S$

Here:

- each of the signals $(i_1^C, \ldots, i_{rC}^C)$ may either be
  a signal corresponding to a connection of N or
  an input signal of N.
- each of the signals $(o_1^C, \ldots, o_{rC}^C)$ may either be
  a signal corresponding to a connection of N or
  an output signal of N.

Modeling and Simulation



Ptolemy demo cannot simulate loop

physical explanation: components cannot react immediately

Breaking Loops: Delay

Intuition: $\mathcal{D}_{S_0}$, first output from S_0, then previous input

Examples:

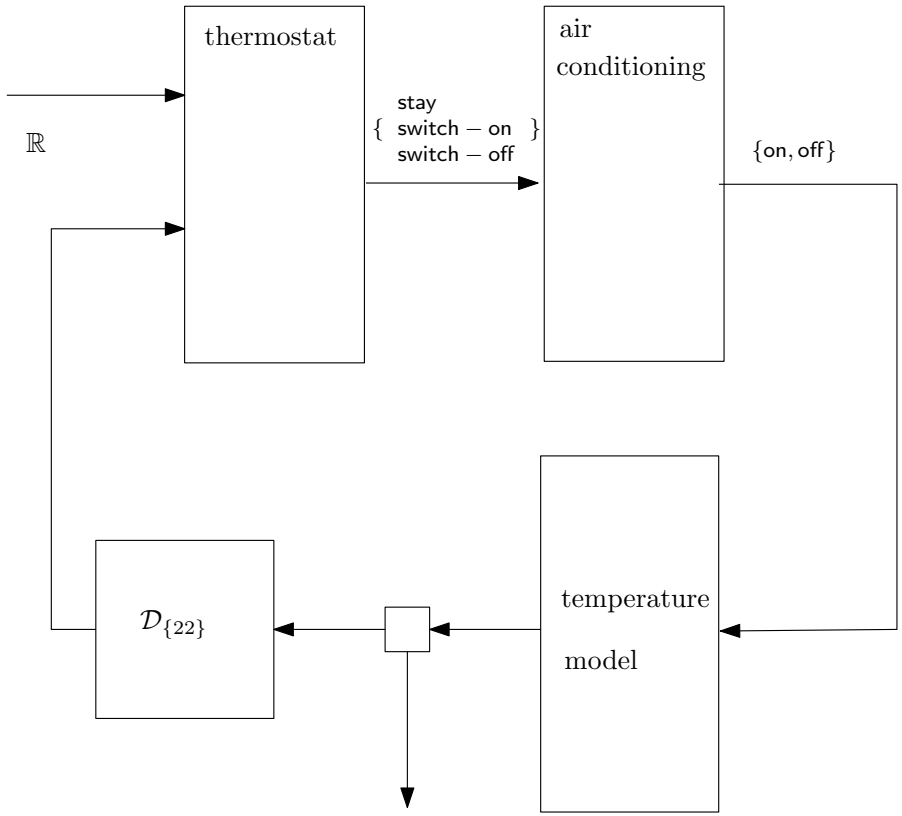
- $((18, 22, 18, 22, \dots), (0, 18, 22, 18, 22, \dots)) \in \mathcal{D}_{\{0,1\}}$
- $((18, 22, 18, 22, \dots), (1, 18, 22, 18, 22, \dots)) \in \mathcal{D}_{\{0,1\}}$
- $((18, 22, 18, 22, \dots), (18, 22, 18, 22, \dots)) \notin \mathcal{D}_{\{0,1\}}$
- $((18, 22, 18, 22, \dots), (0, 22, 18, 22, 18, \dots)) \notin \mathcal{D}_{\{0,1\}}$

For sets S_0, I and O s.t. $I \subseteq O$, $S_0 \subseteq O$,
the *delay* with set of initial states S_0, inputs I, and outputs O is

$$\mathcal{D}_{S_0,I,O} := \{(i, o) \mid o(0) \in S_0, \forall k \in \mathbb{N}, o(k) = i(k-1)\}.$$

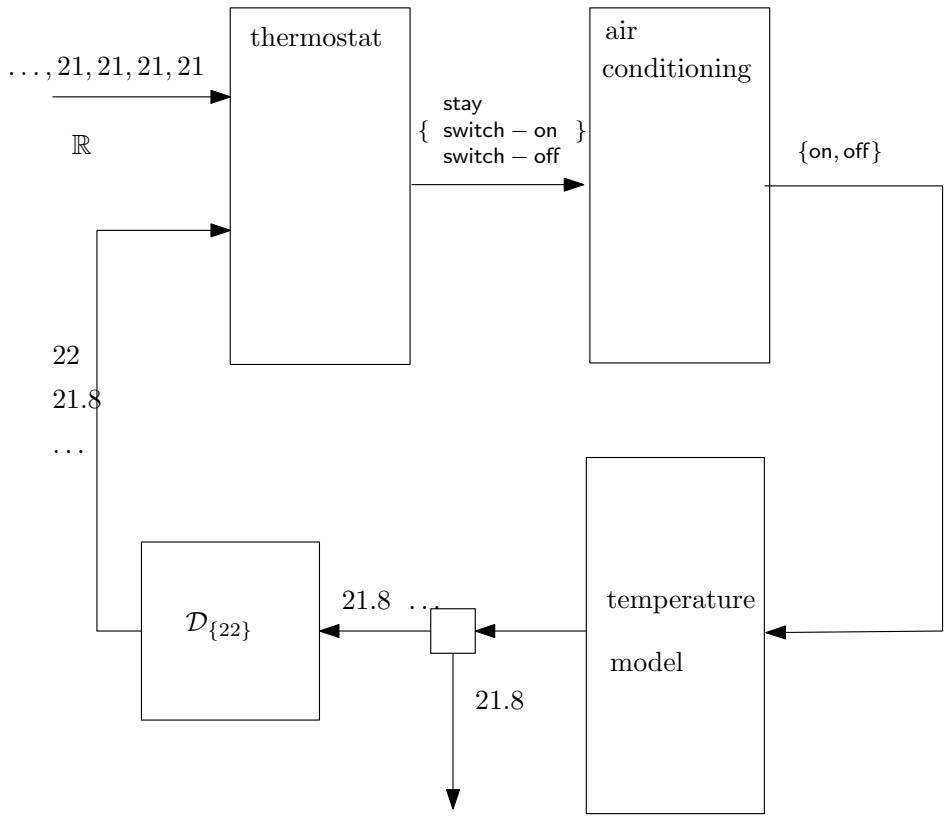
If I and O are clear from the context, then also denoted by $\mathcal{D}_{S_0}$.

Feedback Loop with Delay



Simulation

- Given: input signal
- Find: a corresponding output signal



From definition: $(i_1, \ldots, i_r, o_1, \ldots, o_s) \in N$ iff

there exist a corresponding signal for

every connection between two components of the network s.t. . . .

Simulation

For a given network N,
for given inputs $i_1, \ldots, i_r$ we want to compute corresponding outputs.

for $t \leftarrow 0 \ldots$ **do**

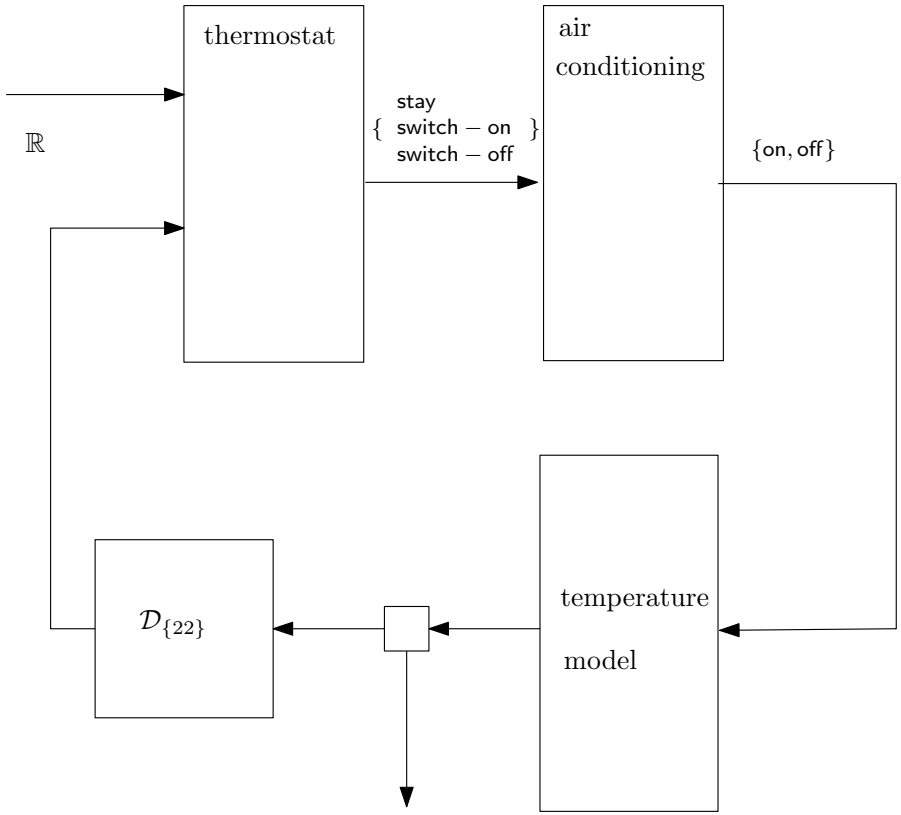
for each delay $\mathcal{D}_{S_0}$ in N and corresponding input signal $i_{\mathcal{D}_{S_0}}$ and output signal $o_{\mathcal{D}_{S_0}}$

let $o_{\mathcal{D}_{S_0}}(t)$ be
if $t = 0$ **then** s_0, for an arbitrary $s_0 \in S_0$,
else $i_{\mathcal{D}_{S_0}}(t-1)$

while there is a network element S with known $i_s(t)$ and unknown $o_s(t)$

compute $o_s(t)$ from $i_s(t)$

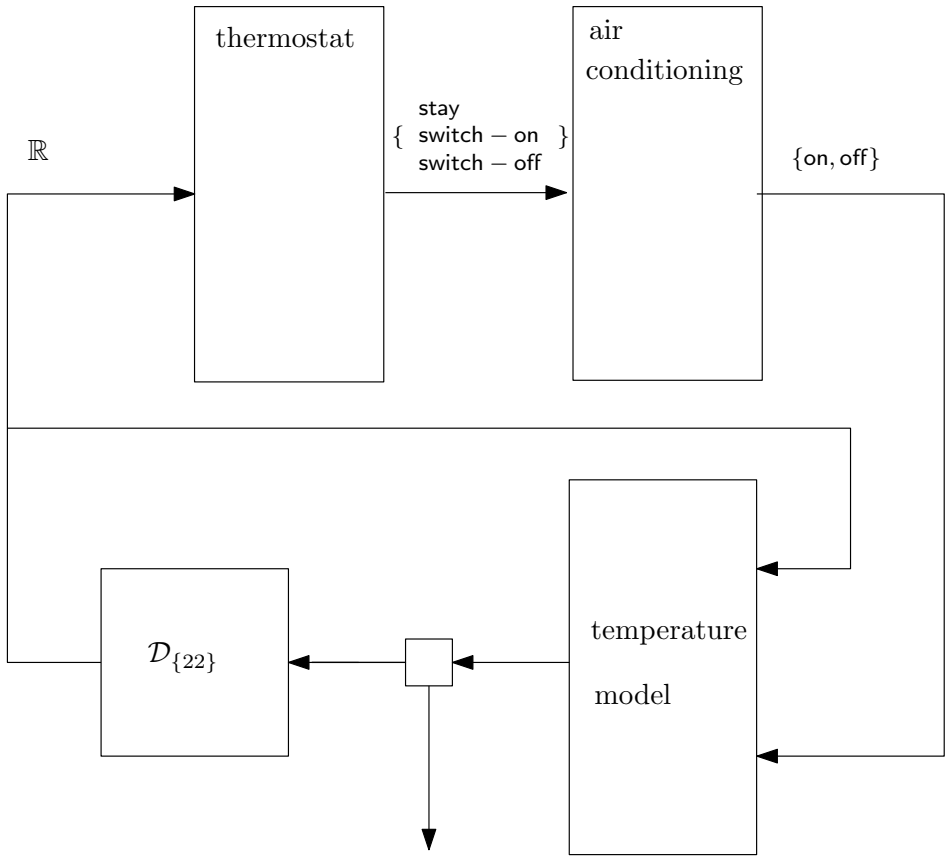
Control Loop with Delay



Problem: temperature jumps within one step!?

Solution: We can use delay as memory element. Ptolemy demo.

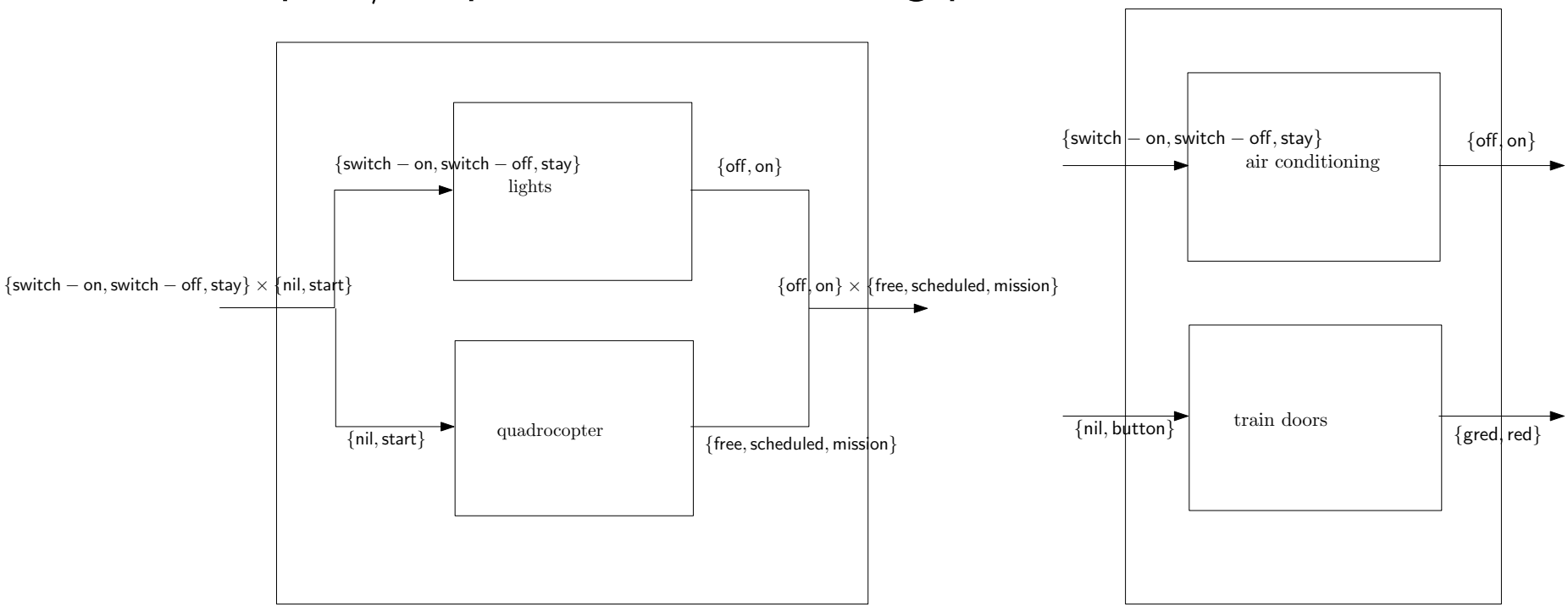
Control Loop with Delay as Memory Element



Ptolemy demo.

General Composition: Comparison

General composition vs. sync. parallel composition:
several inputs/outputs instead of taking pairs



General composition vs. cascade composition

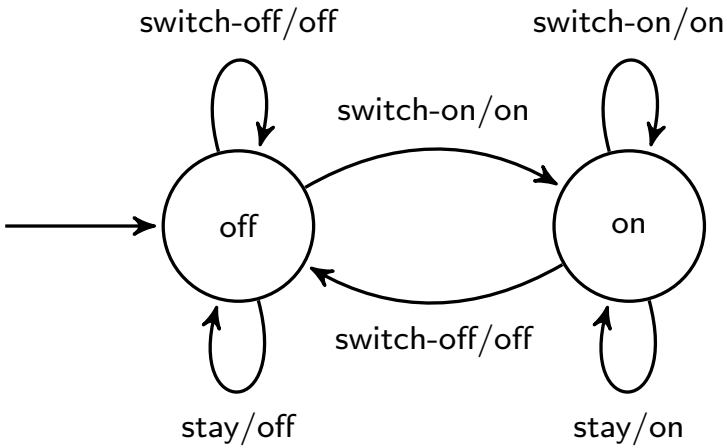
Synchronous Reactive Models and Automata

	systems	automata
Synchronous Parallel Composition	$\surd$	$\surd$
Cascade Composition	$\surd$	$\surd$
General Composition	$\surd$	???

We will not discuss general composition of automata.

Instead: Can we construct automata from even simpler elements?

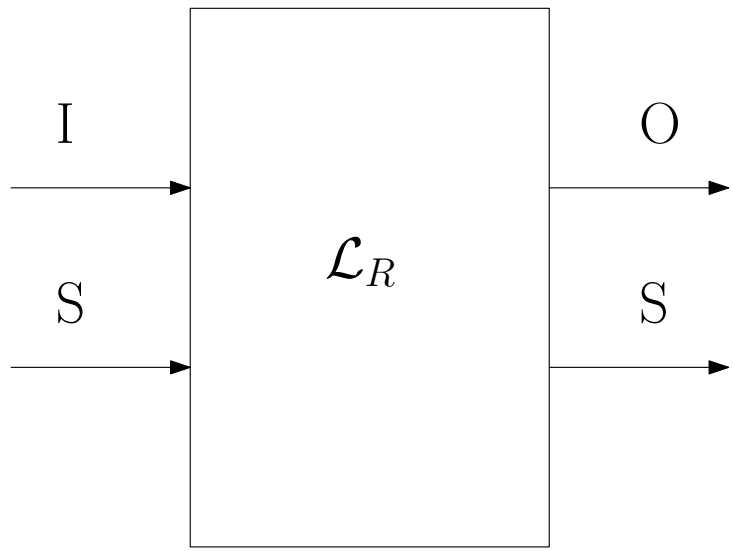
Deconstructing Automata



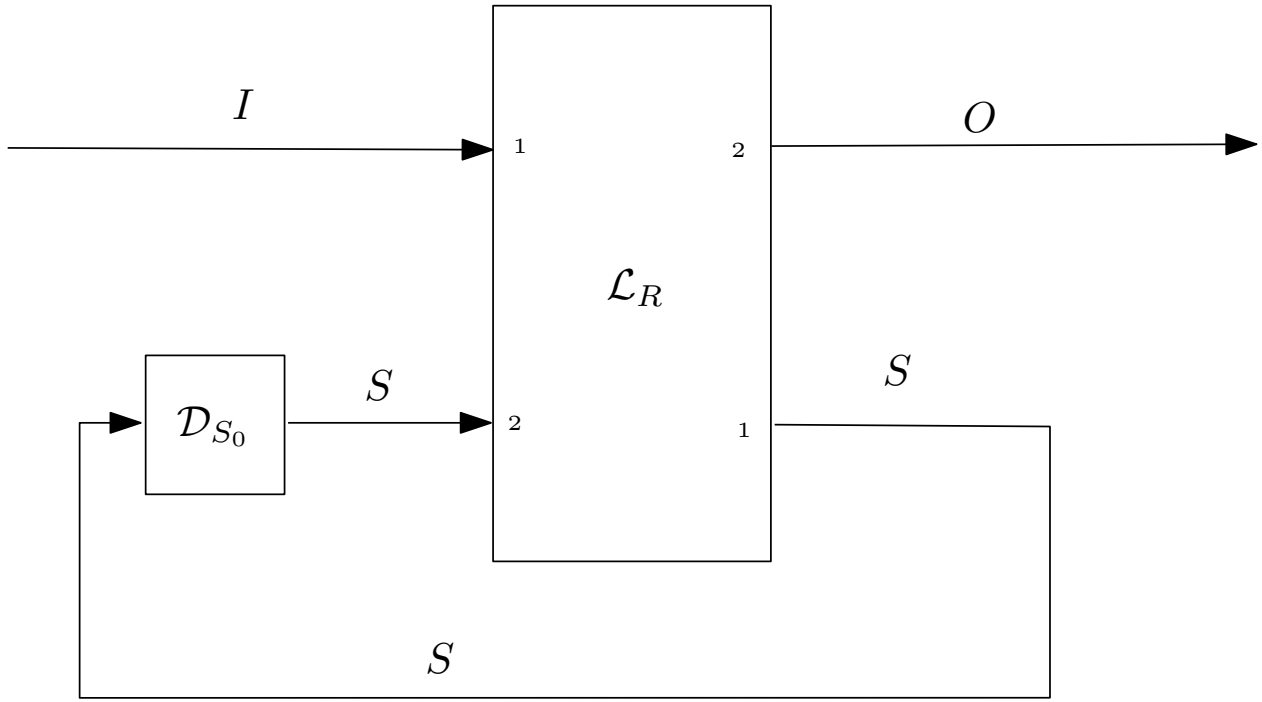
switch-off	off	off	off
stay	off	off	off
switch-on	off	on	on
switch-off	on	off	off
switch-on	on	on	on
stay	on	on	on

Given: input signal, how to compute output signal?

From current input and state, compute output and next state



Automata as Synchronous Reactive Models



Feedback loop

Using delay, table lookup, and general composition with loops, we can build arbitrary discrete time automata

Table Lookup

For a certain relation $R \subseteq I_1 \times \cdots \times I_r \times O_1 \times \cdots \times O_s$,

table lookup in R (which we will often denote by $\mathcal{L}_R$) is

a system with input sets $I_1, \ldots, I_r$

and output sets $O_1, \ldots, O_s$ s.t.

$(i_1, \ldots, i_r, o_1, \ldots, o_s) \in \mathcal{L}_R$ iff for all $k \in \mathbb{N}_0$,

$$(i_1(k), \ldots, i_r(k), o_1(k), \ldots, o_s(k)) \in R$$

Examples:

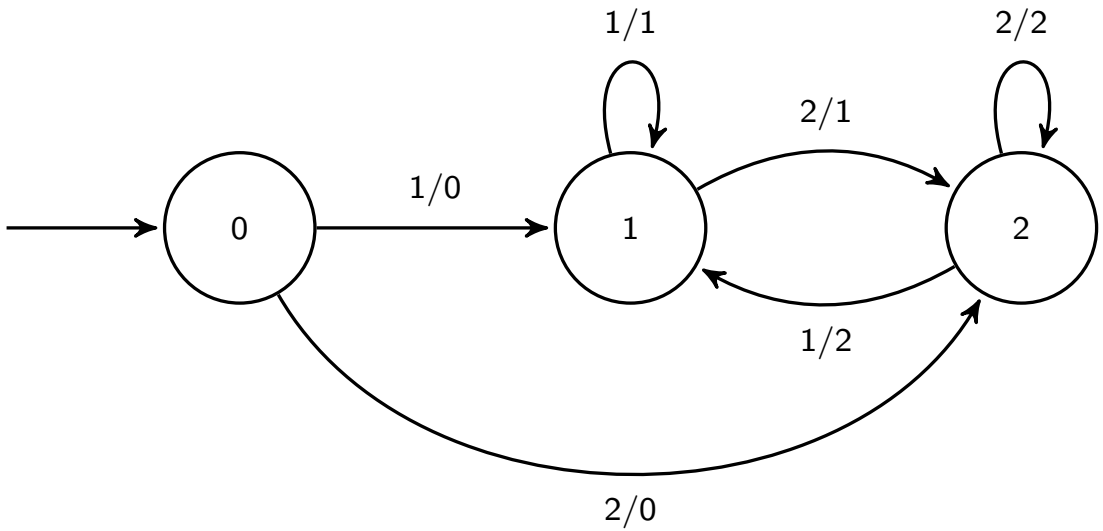
- $R = \{(x, 2x) \mid x \in \mathbb{Z}\}$.
  Then $((1, 2, 3, 4, \ldots), (2, 4, 6, 8, \ldots)) \in \mathcal{L}_R$
- $R = \{(1, a), (1, b)\}$.
  Then $((1, 1, 1, \ldots), (a, b, a, b, \ldots)) \in \mathcal{L}_R$
- $R = \{(x, y, \max\{x, y\}) \mid x, y \in \mathbb{R}\}$
  Then $((5, 5, 5, \ldots), (2, 7, 3, 5, 8, \ldots), (5, 7, 5, 5, 8, \ldots)) \in \mathcal{L}_R$

memory-less

Result can be non-deterministic, non-receptive.

Opposite Direction: Delay as Automaton

$\mathcal{D}_{\{0\},\{1,2\},\{0,1,2\}}$:



Delay $\mathcal{D}_{S_0}$, for a set S_0, $(i, o) \in \mathcal{D}_{S_0}$ iff

- $o(0) \in S_0$,
- for all $k \in \mathbb{N}$, $o(k) = i(k-1)$.

where $I \subseteq O$, $S_0 \subseteq O$.

Automaton representing this system
$(I \cup S_0, S_0, I, O, \{(i, o, i, o) \mid i \in I, o \in O\})$

Table Lookup as Automaton

$\mathcal{L}_{\{(1,a),(1,b),(2,c)\}}$:

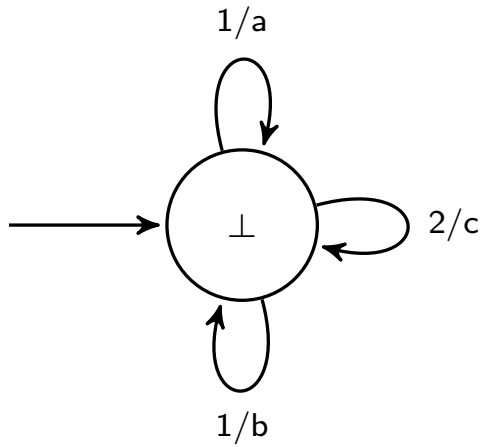


Table lookup $\mathcal{L}_R$:

For a certain relation $R \subseteq I \times O$,

$(i, o) \in \mathcal{L}_R$ iff for all $k \in \mathbb{N}_0$, $(i(k), o(k)) \in R$

Automaton representing this system

$(\{\bot\}, \{\bot\}, I, O, \{(i, \bot, \bot, o) \mid (i, o) \in R)\})$

Comparison with Compiler Construction

In compiler construction, automata represent languages
(i.e., sets of strings)

Operations on automata implement language operations
(union, intersection etc.)

Here: Operations should model how systems interact in the real-world

In Practice

Various modeling tools based on those principles

- Ptolemy
- Matlab/Simulink/Stateflow
- Scilab/Xcos
- UML/SysML based tools
- ...

Especially: Synchronuous reactive programming languages extend this to full programming languages (e.g., Scade/Lustre)

IBM stream computing/InfoSphere Streams:

- Original motivation: gathering and analyzing security information from all across US after 9/11
- Nowadays: online analysis for companies, transportation information, banking security, intelligent energy networks

Conclusion

Synchronous reactive models:
arbitrary connection of system inputs and outputs.

Synchronous reactive models with certain elements
can implement automata

Delay, table lookup: in practice, many further elements

Further models of communication (often asynchronous):

- actors
- dataflow
- process networks, process algebras
- functional reactive programming

Literature

Edward A. Lee and Sanjit A. Seshia. *Introduction to Embedded Systems, A Cyber-Physical Systems Approach*. `http://LeeSeshia.org`, 2011.

Edward A. Lee and Pravin Varaiya. *Structure and Interpretation of Signals and Systems*. `http://LeeVaraiya.org`, 2011.

Claudius Ptolemaeus, editor. *System Design, Modeling, and Simulation using Ptolemy II*. Ptolemy.org, 2014. URL `http://ptolemy.org/books/Systems`.