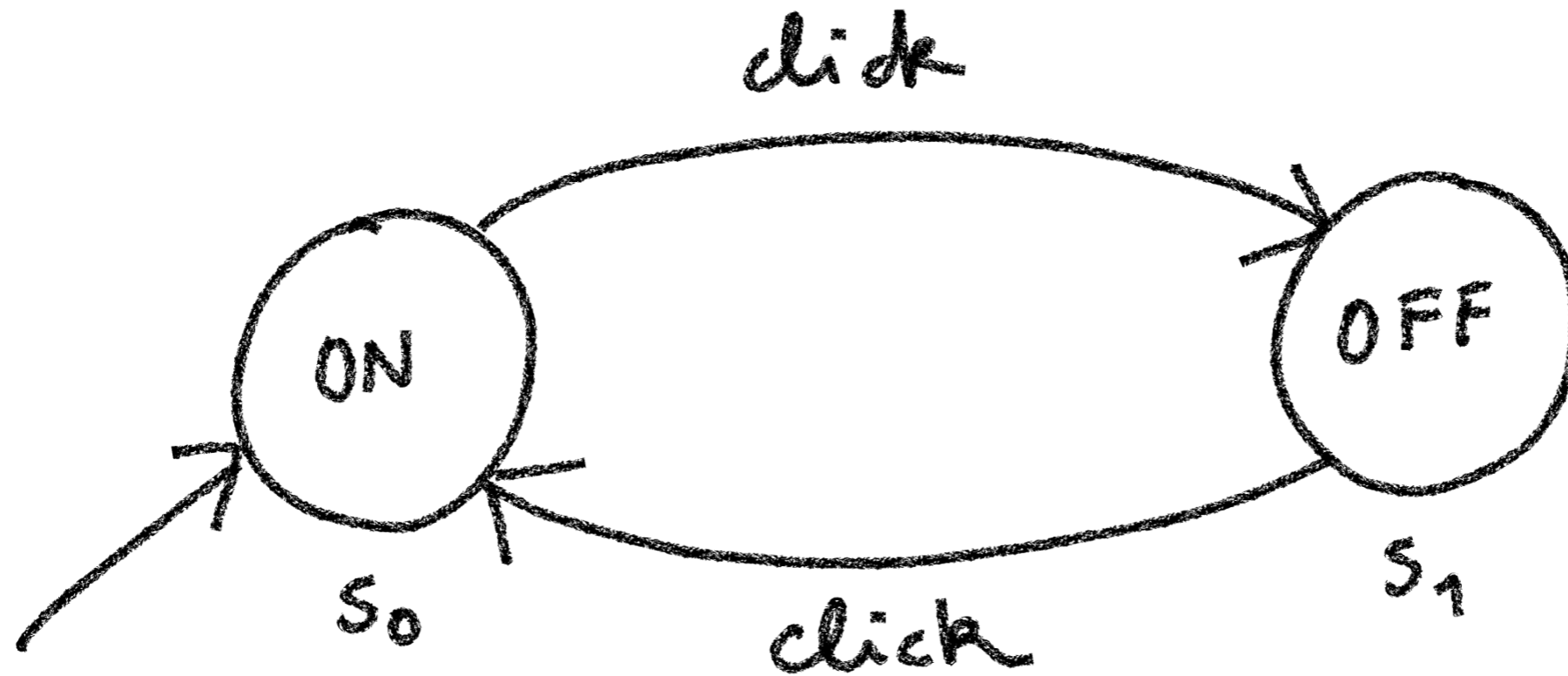


Lecture 01 - Transitions Systems

- What are Transition Systems?
- Modelling with Transition Systems
- Semantics of Transition Systems
- Composing Transition Systems

Transition systems, graphically



Transition systems, formally

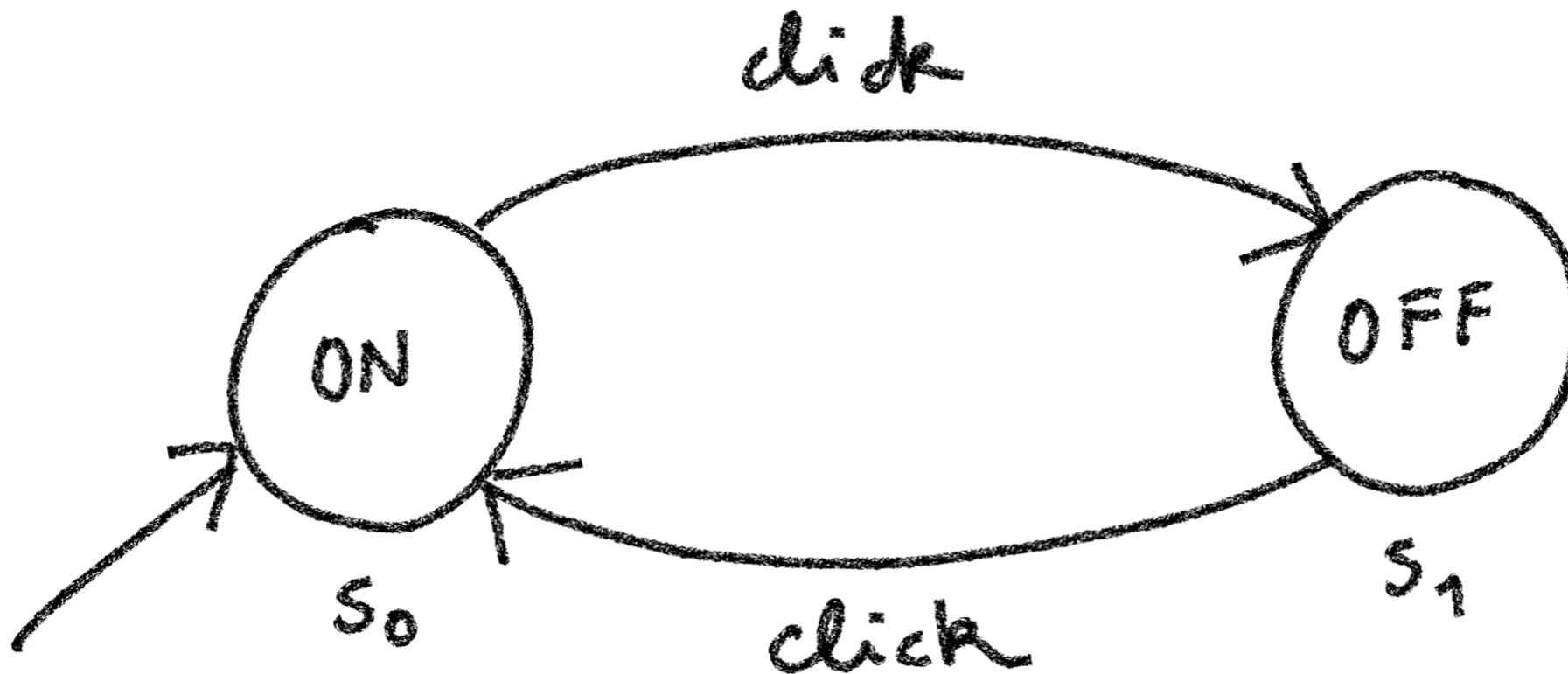
Def. A transition system is a tuple

$$\langle S, A, \rightarrow, L, AP, I \rangle$$

such that

- S is a set of states
- A is a set of "Actions"
- $\rightarrow \subseteq S \times A \times S$ is a set of transitions
- $L : S \rightarrow 2^{AP}$ is a labelling function
- AP is a finite set of "Atomic Propositions"
- $I \subseteq S$ is the set of initial states

Our example, formally



$$S = \{s_0, s_1\}$$

$$\rightarrow = \{(s_0, \text{click}, s_1), (s_1, \text{click}, s_0)\}$$

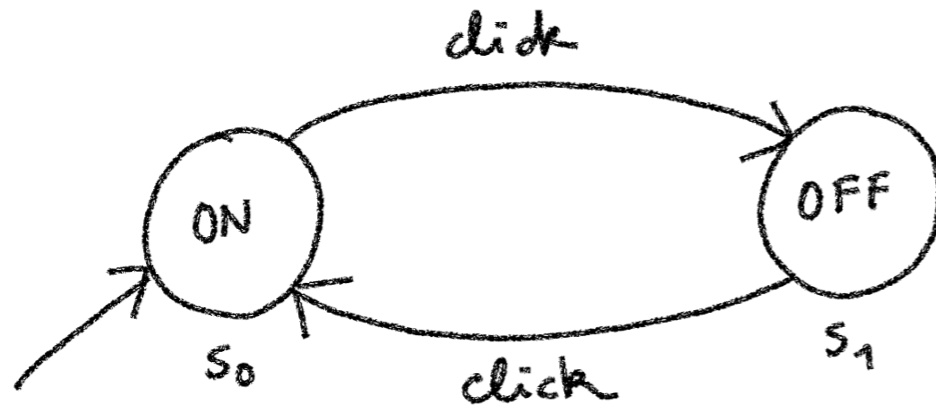
$$A = \{\text{click}\}$$

$$L = \left\{ \begin{array}{l} s_0 \mapsto \{\text{ON}\} \\ s_1 \mapsto \{\text{OFF}\} \end{array} \right\}$$

$$AP = \{\text{ON}, \text{OFF}\}$$

$$S_0 = \{s_0\}$$

Our example, in PRISM



$$S = \{s_0, s_1\}$$

$$T = \{(s_0, \text{click}, s_1), (s_1, \text{click}, s_0)\}$$

$$A = \{\text{click}\}$$

$$L = \left\{ \begin{array}{l} s_0 \mapsto \{\text{ON}\} \\ s_1 \mapsto \{\text{OFF}\} \end{array} \right\}$$

$$AP = \{\text{ON}, \text{OFF}\}$$

$$S_0 = \{s_0\}$$

```
PRISM 4.3.1
File Edit Model Properties Simulator Log Options

PRISM Model File: switch.prism*

1 mdp
2
3 module SWITCH
4
5     state : [0..1] init 0;
6
7     [click] state=0 -> (state'=1);
8     [click] state=1 -> (state'=0);
9
10 endmodule
11
12 label "ON" = (state=0);
13 label "OFF" = (state=1);
14

Model Properties Simulator Log
Parsing model... done.
```

Forward and backwards reachability

We sometimes need to talk about how states can reach each other.

The set of **direct successors** of a state s is defined as

$$Post(s) = \{s' \mid \exists s. s \xrightarrow{\alpha} s'\}$$

Similarly, we can define the **direct predecessors** of a state s as

$$Pre(s) = \{s' \mid \exists s. s' \xrightarrow{\alpha} s\}$$

The above definitions can be lifted to sets of states S .

$$Post(S) = \dots \qquad Pre(S) = \dots$$

Successors/predecessors **in any number of transitions** can be defined likewise.

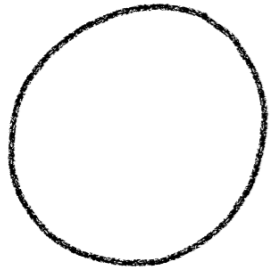
On a related note, one is typically interested in the part of a transition system that is reachable from the initial state(s).

Lecture 01 - Transitions Systems

- What are Transition Systems?
- Modelling with Transition Systems
- Semantics of Transition Systems
- Composing Transition Systems

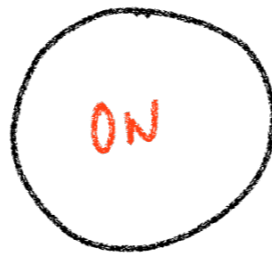
Modelling with transition systems

states

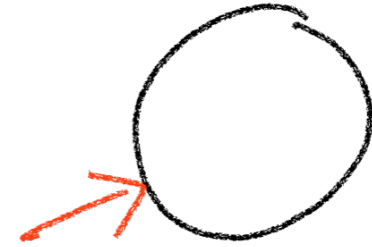


↪

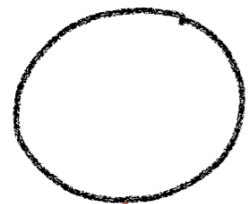
atomic prop.



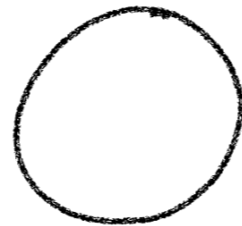
initial states



transition



actions



click



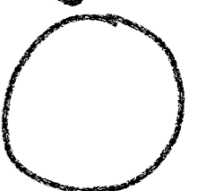
non-determinism



click



click

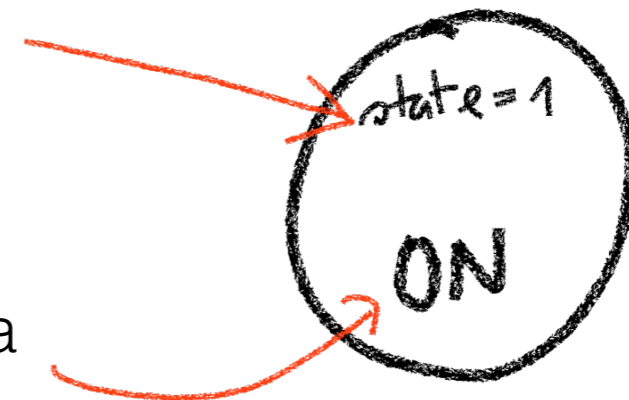


State names, state descriptions and state labels

State names: a unique name to distinguish each individual state



State descriptions: whatever actually defines a state (e.g. the value of each variable, the content of each communication channel, etc.).



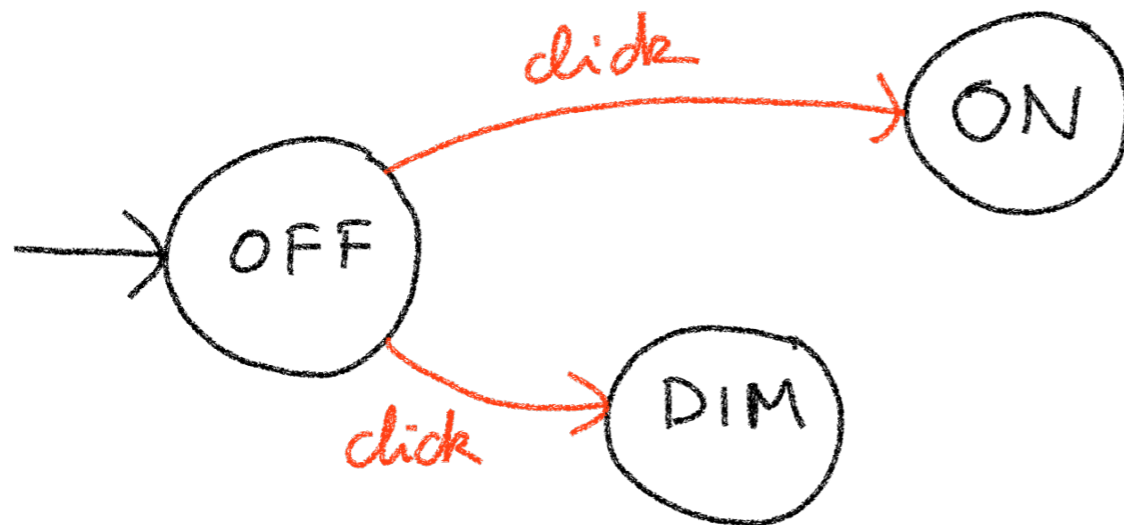
State labels: atomic properties (facts) that are true in a specific state. This is the abstract way of looking at state by focusing on its properties of interest.

Sometimes you may see these concepts mixed up due to (often unwritten) conventions... pay attention!

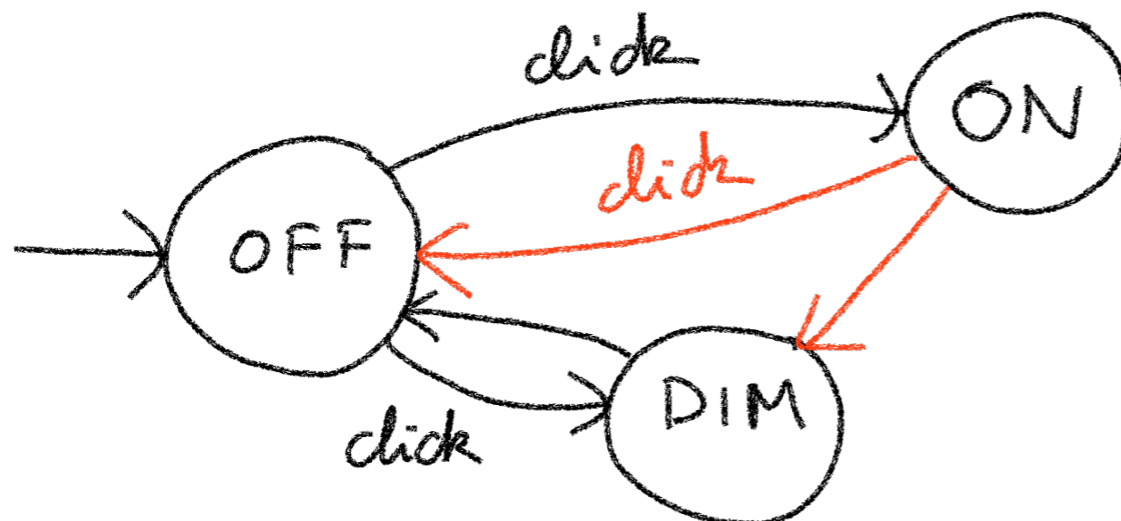
Non-determinism

Non-determinism is a fundamental abstraction.

It may represent internal decisions of the system that are intentionally underspecified (e.g. an “if-then-else” which logic will be decided later).



Or it may represent the uncertainty of the environment where the system will operate (users, schedulers, network, etc.).



Deterministic transition system

A transition system is **action-deterministic** iff

(1) There is no more than 1 initial state

$$|I| \leq 1$$

(2) For every state s and every action a , there is only one outgoing transition from s labelled with a .

$$|\{s' \mid s \xrightarrow{a} s'\}| \leq 1$$

Terminal states

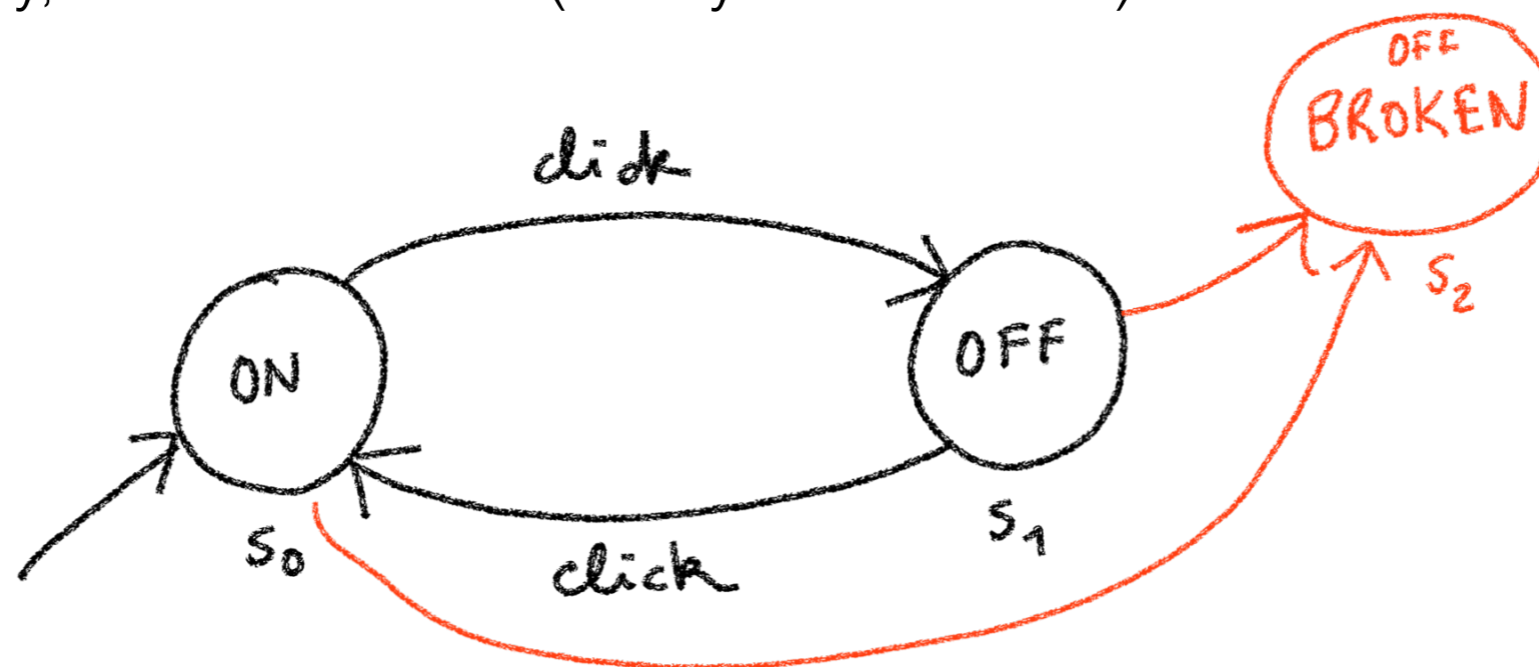
A state is a **terminal state** if it has no outgoing transition.

We can write this formally in several ways: a state s is terminal iff

$$\neg \exists s' . s \rightarrow s'$$

$$Post(s) = \emptyset$$

Terminal states usually represent **final states** (the system finished doing its job) or, more typically, **deadlock states** (the system is stuck).

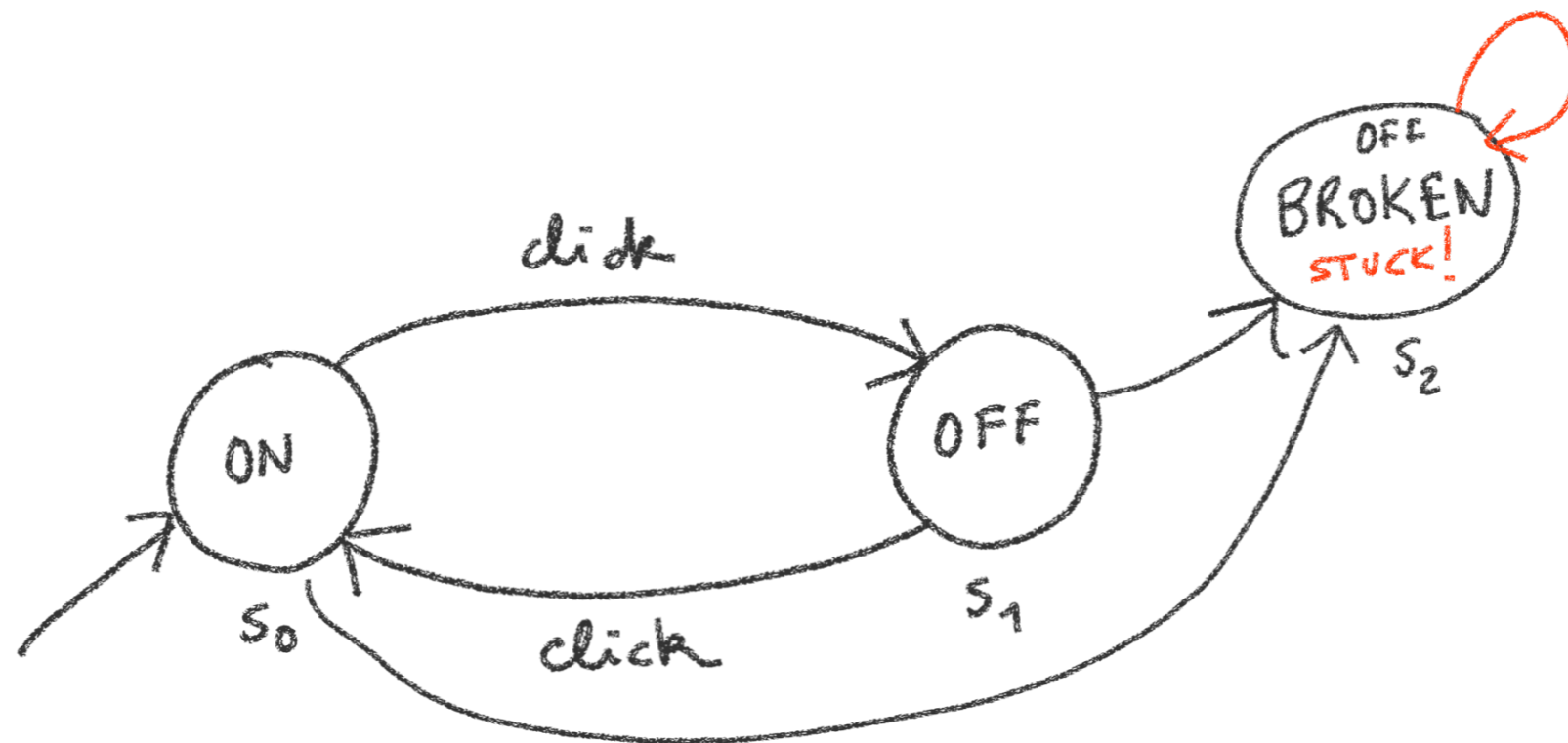


Transition systems with no terminals

From now on we consider w.o.l.g. transition systems with no terminal states.

If you have a terminal state, just add a self-loop.

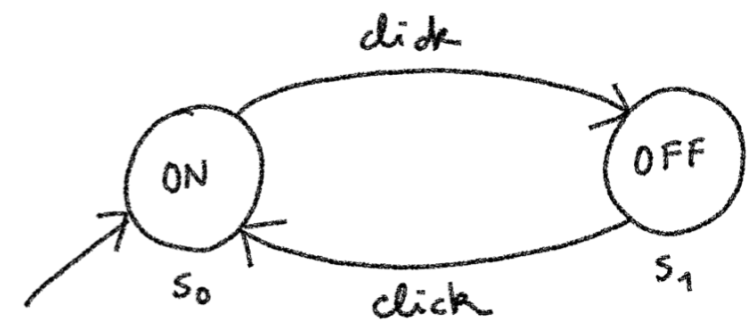
If you are interested in marking the state as “terminal” and distinguish it from proper self-loops, just add a special label to it.



Lecture 01 - Transitions Systems

- What are Transition Systems?
- Modelling with Transition Systems
- Semantics of Transition Systems
- Composing Transition Systems

Executions



An execution fragment is a sequence of transitions.

$$s_0 \xrightarrow{\text{click}} s_1 \xrightarrow{\text{click}} s_0 \xrightarrow{\text{click}} \dots$$

An execution is **finite/infinite** if the sequence is **finite/infinite**.

$$s_0 \xrightarrow{\text{click}} s_1 \xrightarrow{\text{click}} s_0 \quad (s_0 \xrightarrow{\text{click}} s_1 \xrightarrow{\text{click}})^\omega$$

An execution is **initial** if the first state of the sequence is in I .

$$s_0 \xrightarrow{\text{click}} s_1 \xrightarrow{\text{click}} s_0 \rightarrow \dots \quad s_1 \xrightarrow{\text{click}} s_0 \xrightarrow{\text{click}} s_1 \xrightarrow{\text{click}} \dots$$

An execution is **maximal** if it cannot be extended: either it is finite and the last state is a terminal state, or it is infinite.

NOTE: by our assumption of “no terminal state”, only the second case applies.

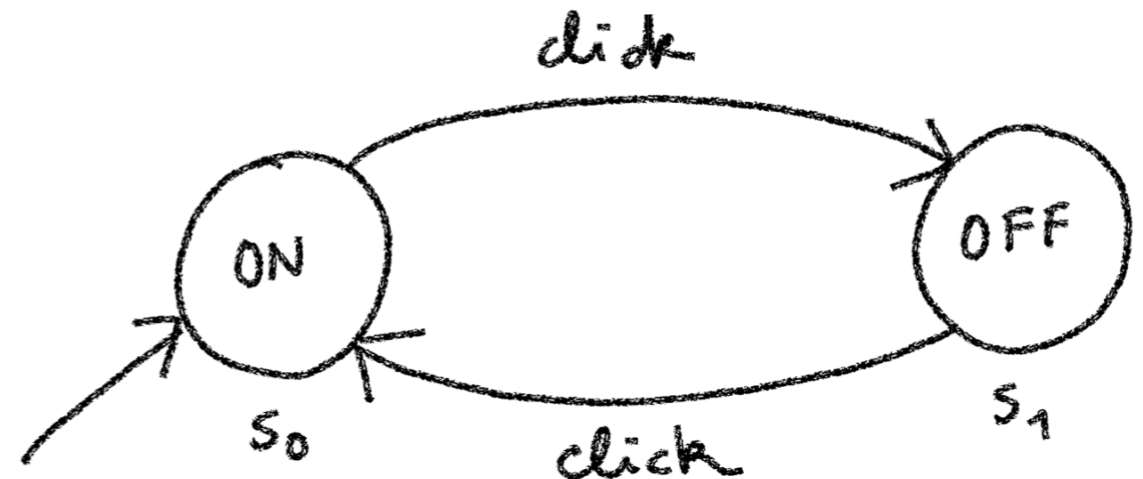
NOTE: often, we drop the arrows.

Traces

Executions may contain too much information, i.e. actual states.

Often, one is interested in the **atomic properties of states**.

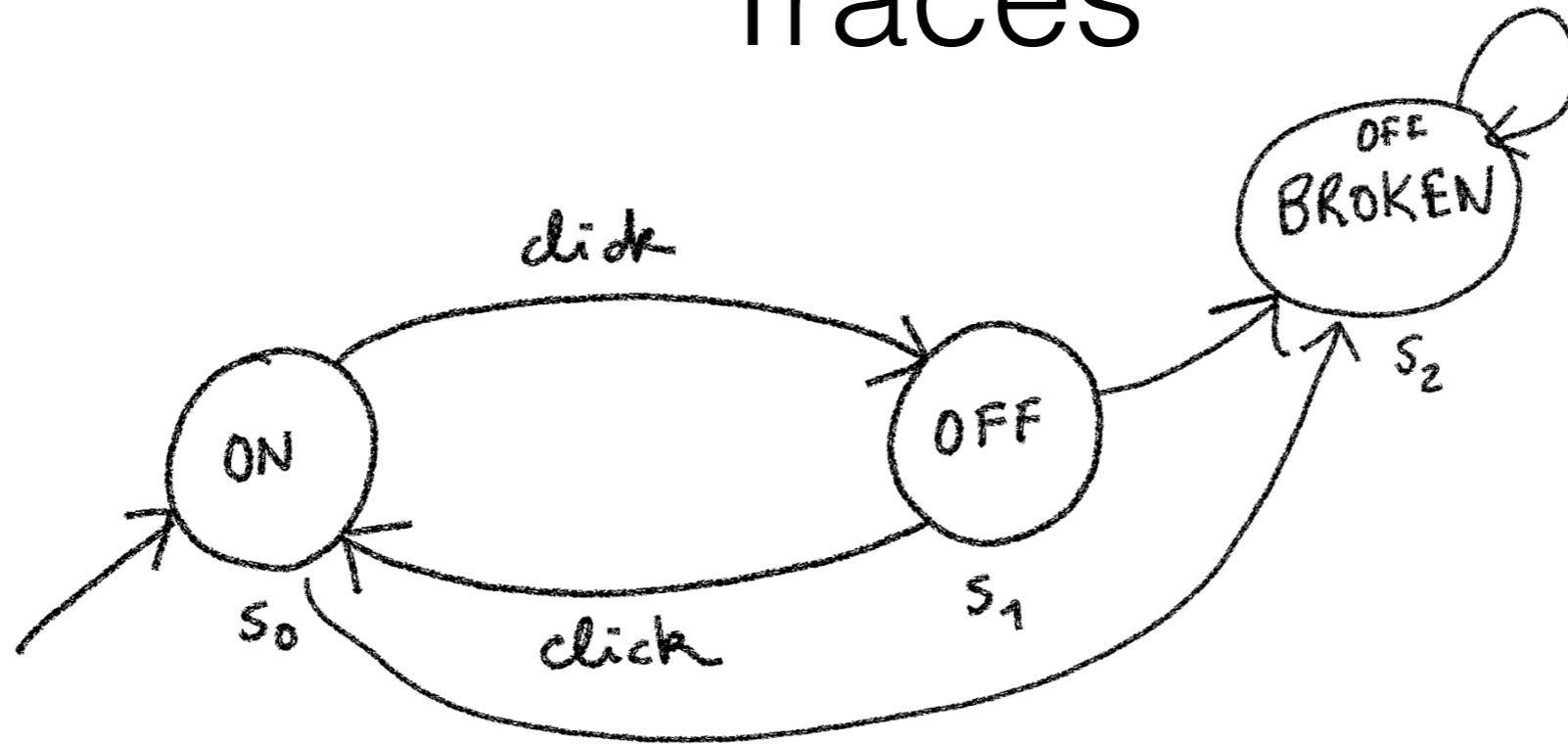
Replacing every state in an execution by its atomic properties yields a **trace**.

$$s_0 \xrightarrow{\text{click}} s_1 \xrightarrow{\text{click}} s_0 \rightarrow \dots$$
$$ON \xrightarrow{\text{click}} OFF \xrightarrow{\text{click}} ON \xrightarrow{\text{click}} \dots$$


We may even drop the transitions and their actions

$$ON, OFF, ON, \dots$$

Traces



Computation trees

Executions and traces can be seen as the system running with a fixed scheduler.

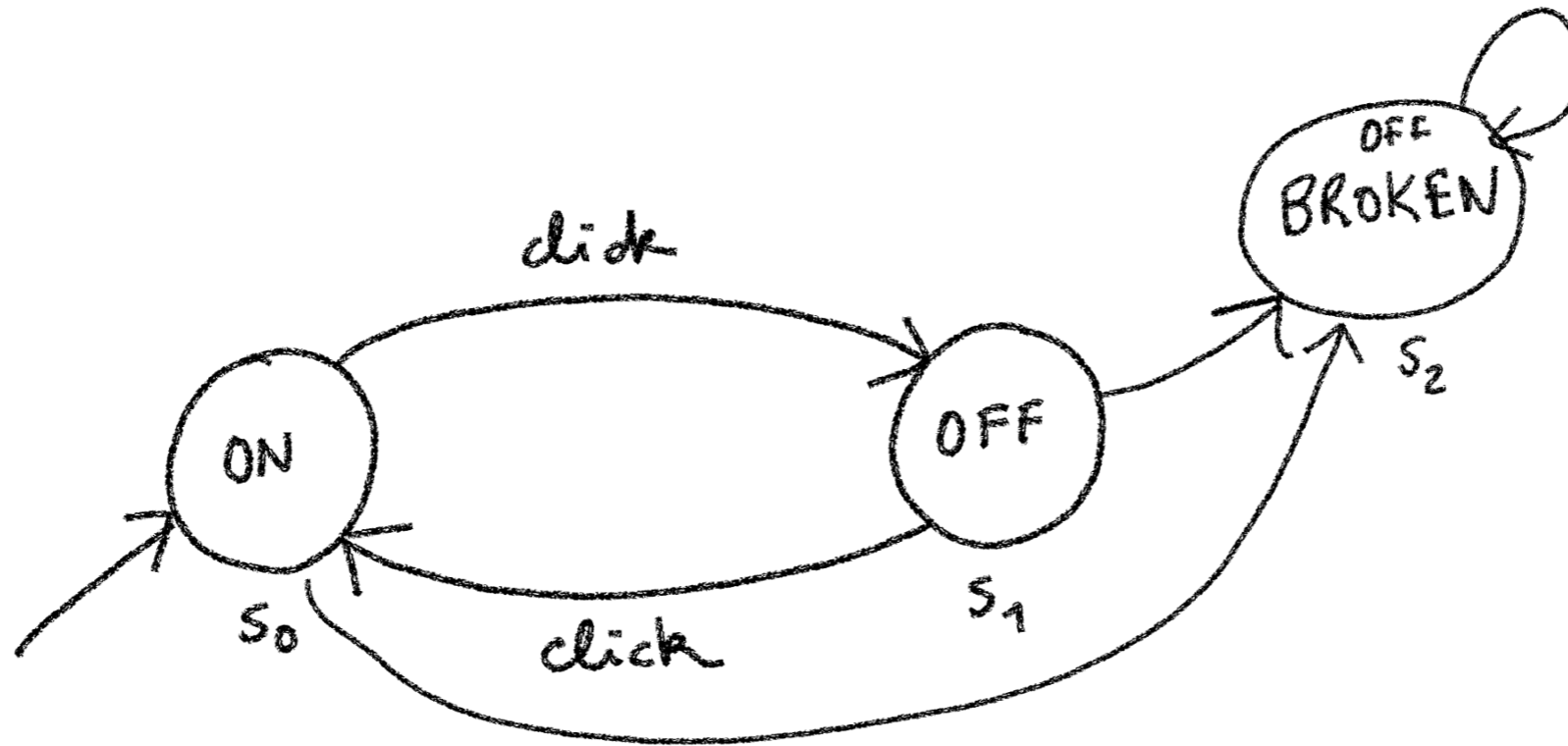
They are not appropriate if we want to see how the system makes choices.

Computation trees can save the day!

A computation tree is a tree whose nodes are states (or their atomic propositions). Each state has each immediate successor as a successor.

A computation tree is the unfolding of the transition system.

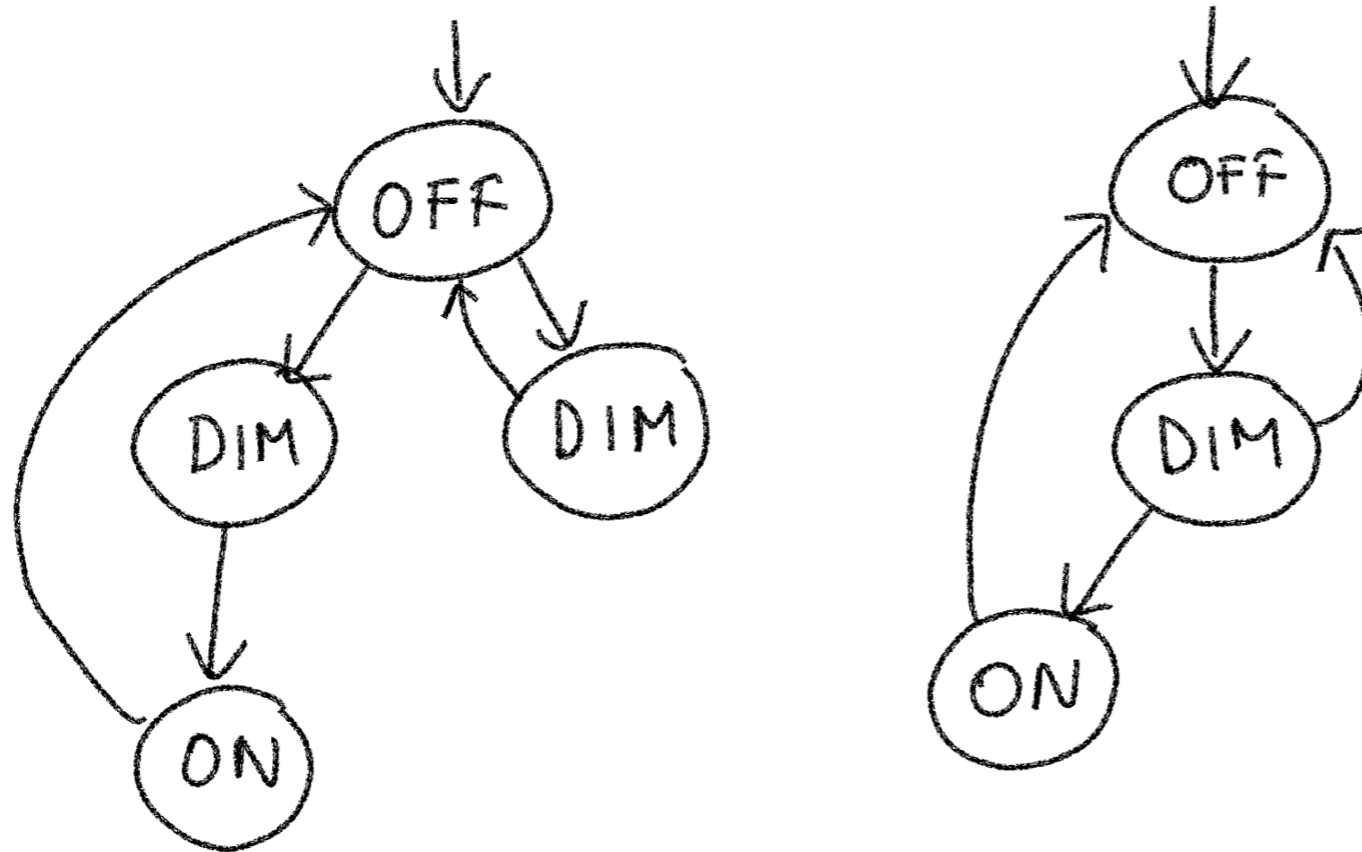
Computation trees



Traces vs computation trees

Computation trees contain more information than traces.

Indeed, all (initial) traces of the system are encoded in its (initial) computation tree(s).



Lecture 01 - Transitions Systems

- What are Transition Systems?
- Modelling with Transition Systems
- Semantics of Transition Systems
- Composing Transition Systems

System interactions

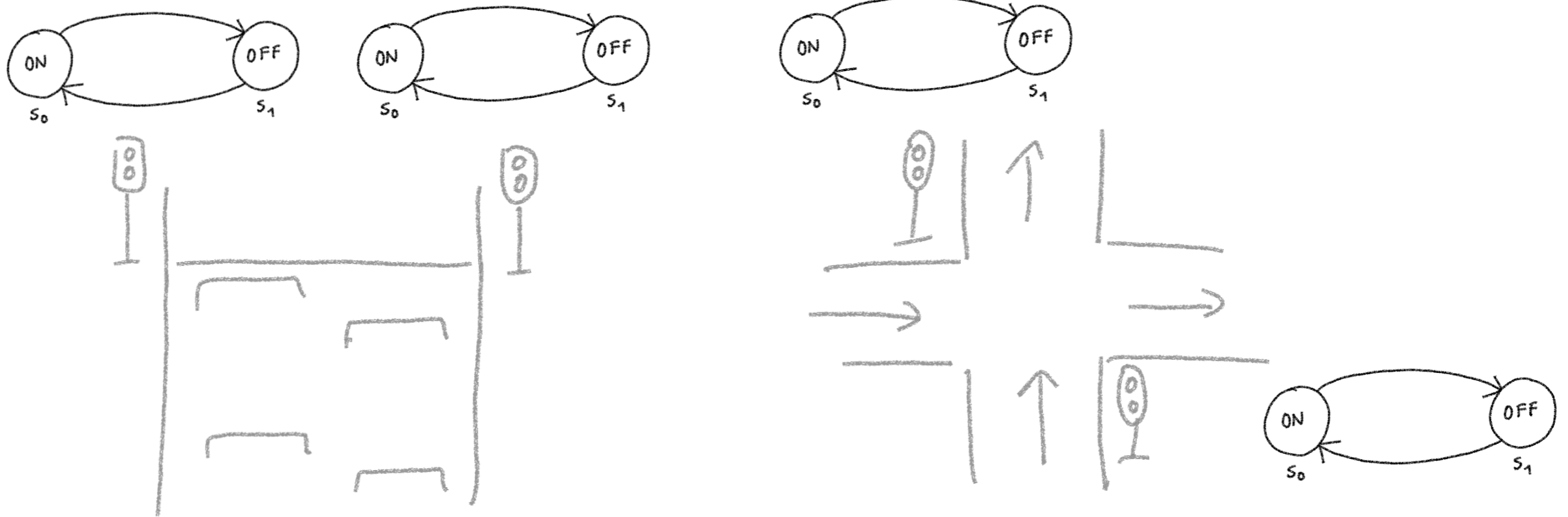
Often, systems (and their models) are made of several components, which interact with each other.

There is a plethora of interaction mechanisms:

- Shared memory / objects / storage (consistent, weak, safe, etc.)
- Networks (synchronous/asynchronous, binary/multi-party, value-passing/rendezvous, etc.)

System interactions, an example

How would you synchronise the actions these traffic light controllers?

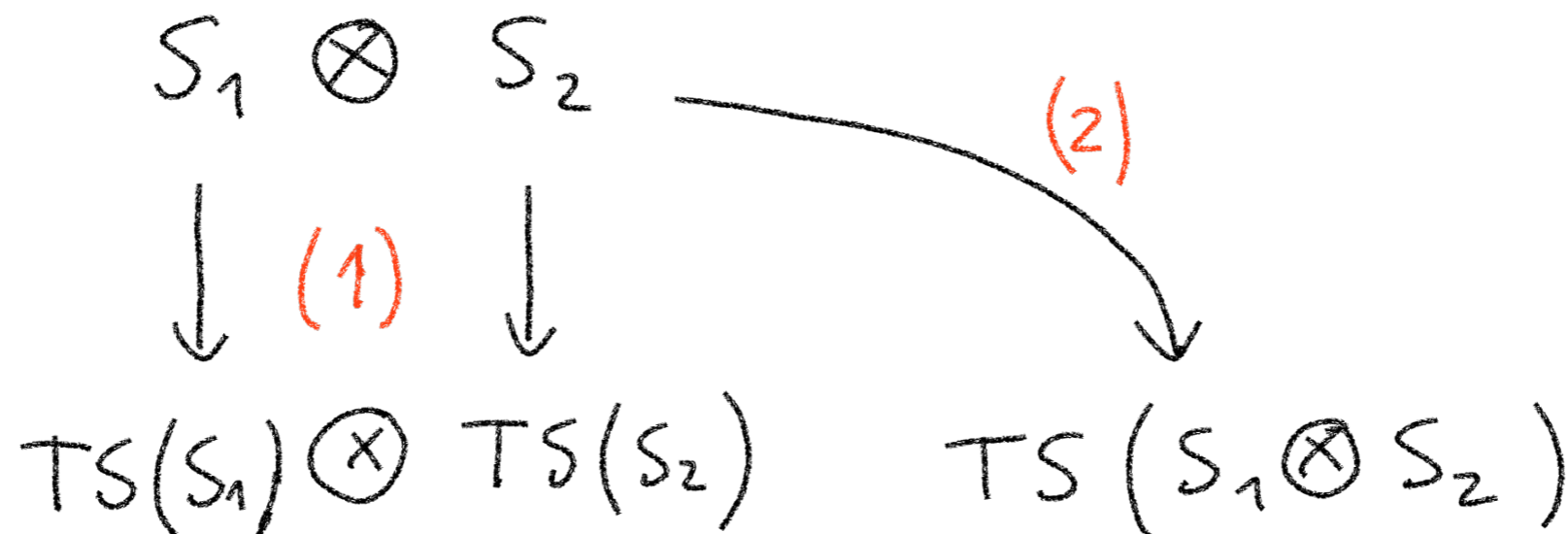


Composing systems or transition systems?

The transition system of the composition $S_1 \otimes S_2$ of two systems S_1, S_2 can be obtained in two ways.

(1) Build $TS(S_1)$ and $TS(S_2)$, then compose

(2) Build the composed TS directly from S_1, S_2



We will see

2 examples of (1): pure interleaving and action-synchronisation
and 1 example of (2): concurrent threads with shared memory

Interleaving composition of transition systems

Interleaving composition of T_1 and T_2 is defined as follows

$$T_1 = \langle S_1, A_1, \rightarrow_1, L_1, AP_1, I_1 \rangle$$

$$T_2 = \langle S_2, A_2, \rightarrow_2, L_2, AP_2, I_2 \rangle$$

$$T_1 \parallel T_2 = \langle S_1 \times S_2, A_1 \cup A_2, \rightarrow, L, AP_1 \cup AP_2, I_1 \times I_2 \rangle$$

where the key point is the new transition function

$$\frac{S_1 \xrightarrow{\alpha} S_1'}{\langle S_1, S_2 \rangle \xrightarrow{\alpha} \langle S_1', S_2 \rangle} \quad \frac{S_2 \xrightarrow{\alpha} S_2'}{\langle S_1, S_2 \rangle \xrightarrow{\alpha} \langle S_1, S_2' \rangle}$$

$$L(\langle S_1, S_2 \rangle) = L_1(S_1) \cup L_2(S_2)$$

alternatively

$$\begin{aligned} \rightarrow &= \left\{ \langle S_1, S_2 \rangle \xrightarrow{\alpha} \langle S_1', S_2 \rangle \mid S_1 \xrightarrow{\alpha} S_1' \right\} \\ &\cup \left\{ \langle S_1, S_2 \rangle \xrightarrow{\alpha} \langle S_1, S_2' \rangle \mid S_2 \xrightarrow{\alpha} S_2' \right\} \end{aligned}$$

Synchronised composition of transition systems

Synchronised composition of T_1 and T_2 over actions B

$$T_1 = \langle S_1, A_1, \rightarrow_1, L_1, AP_1, I_1 \rangle$$

$$T_2 = \langle S_2, A_2, \rightarrow_2, L_2, AP_2, I_2 \rangle$$

$$T_1|_B T_2 = \langle S_1 \times S_2, A_1 \cup A_2, \rightarrow, L, AP_1 \cup AP_2, I_1 \times I_2 \rangle$$

$$L(\langle s_1, s_2 \rangle) = L_1(s_1) \cup L_2(s_2)$$

again, the key point is the new transition function

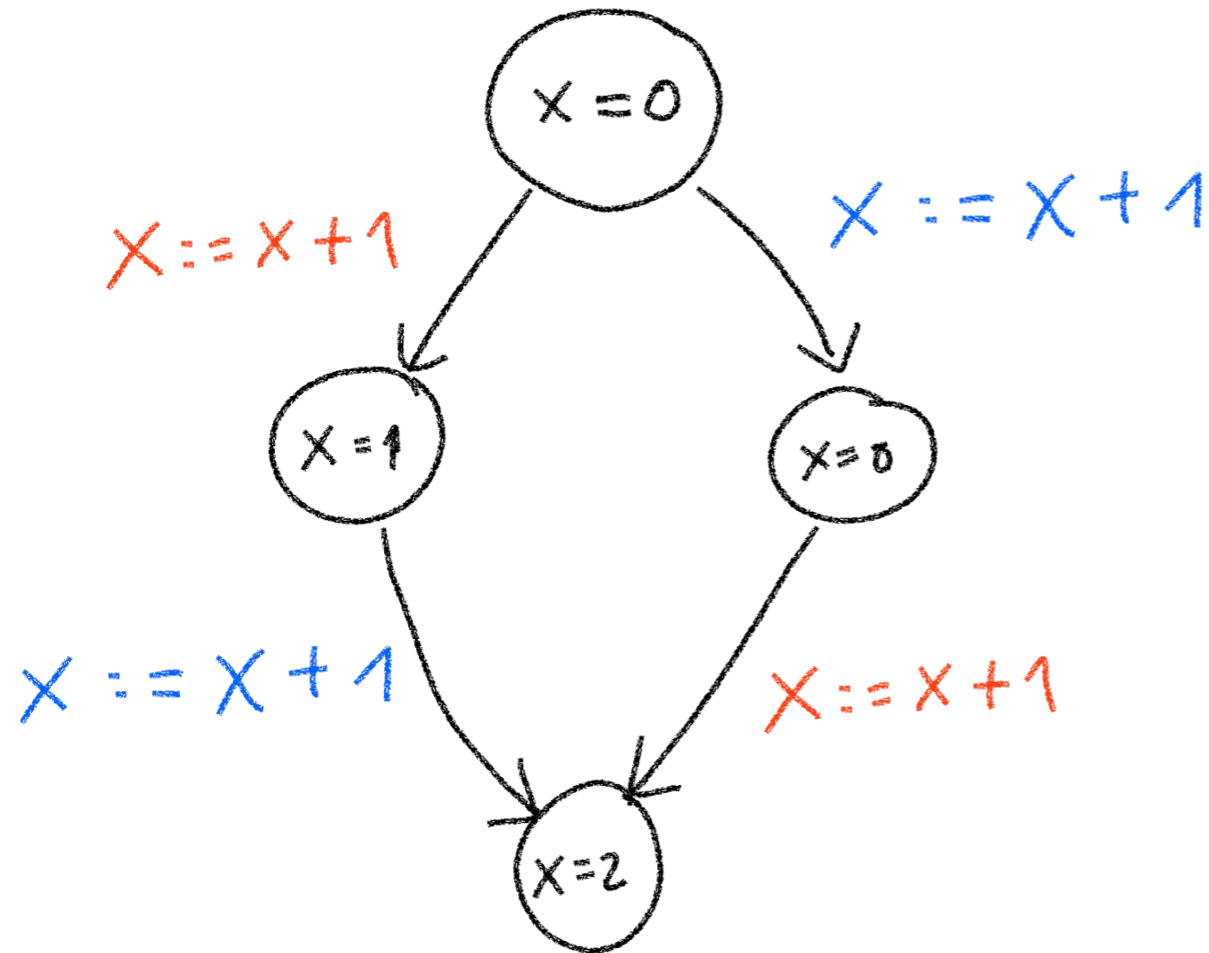
$$\frac{s_1 \xrightarrow{\alpha} s_1' \quad \alpha \notin B}{\langle s_1, s_2 \rangle \xrightarrow{\alpha} \langle s_1', s_2 \rangle} \quad \frac{s_2 \xrightarrow{\alpha} s_2' \quad \alpha \notin B}{\langle s_1, s_2 \rangle \xrightarrow{\alpha} \langle s_1, s_2' \rangle}$$

$$\frac{s_1 \xrightarrow{\alpha} s_1' \quad s_2 \xrightarrow{\alpha} s_2' \quad \alpha \in B}{\langle s_1, s_2 \rangle \xrightarrow{\alpha} \langle s_1', s_2' \rangle}$$

Interleaving composition of concurrent threads with shared memory

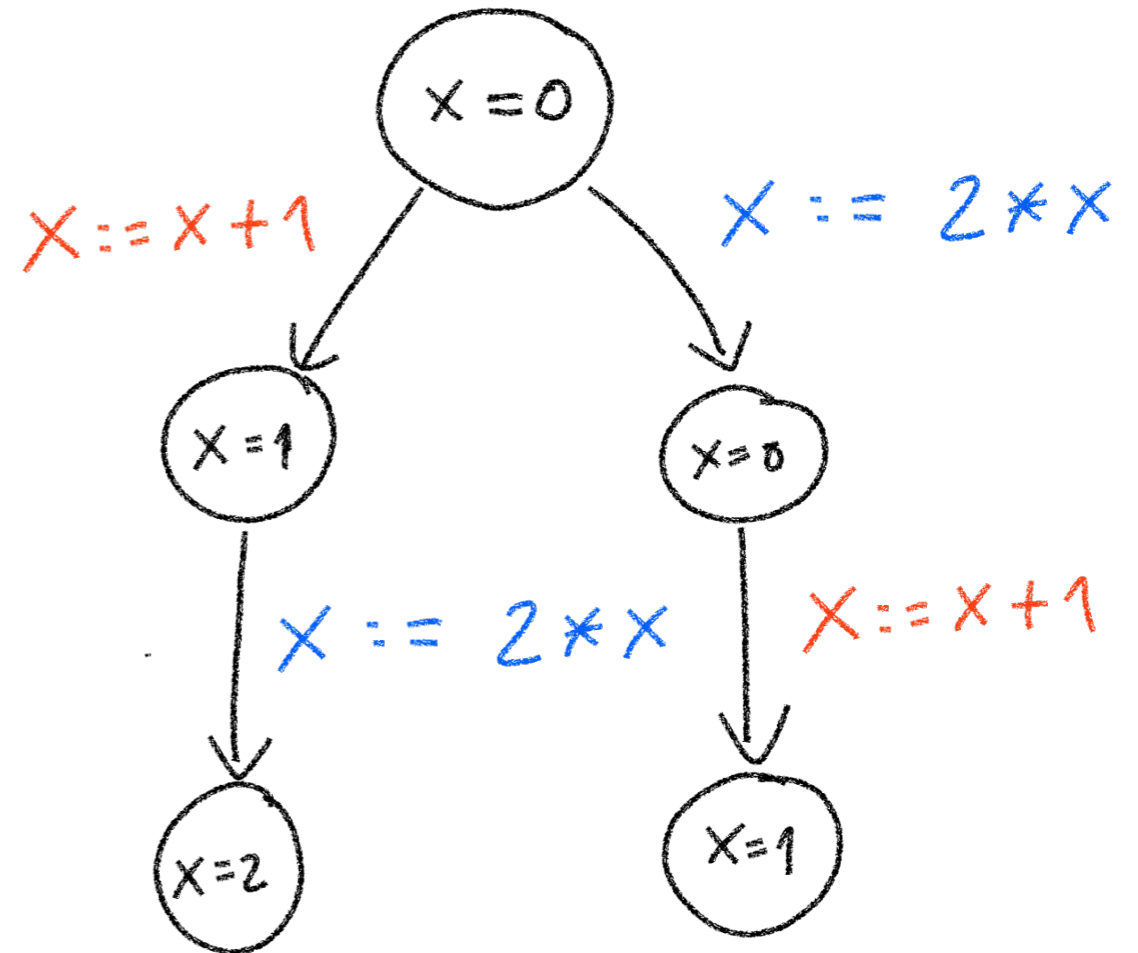
Thread 1
 $x := x + 1$

Thread 2
 $x := x + 1$



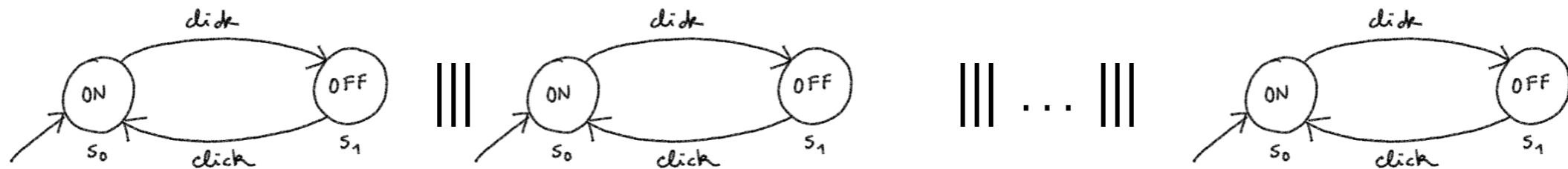
Thread 1
 $x := x + 1$

Thread 2
 $x := 2 * x$



State space explosion

How many states does this transition system have?



In general, the size of the interleaving of n transition systems of m states each is...

Synchronisations may reduce the size composition, but the worst-case is still exponential in the number of components.

Lecture 01 - Transitions Systems

- What are Transition Systems?
- Modelling with Transition Systems
- Semantics of Transition Systems
- Composing Transition Systems

Key points so far

Definition of **transition systems** and related concepts (successors, predecessors, etc.).

Definition of **executions, traces and computation trees**, and related concepts (finite, maximal, etc.).

Concept of **non-determinism** as powerful modelling abstraction and its formal definition.

Interaction and composition of systems and transition systems and the state space explosion problem.