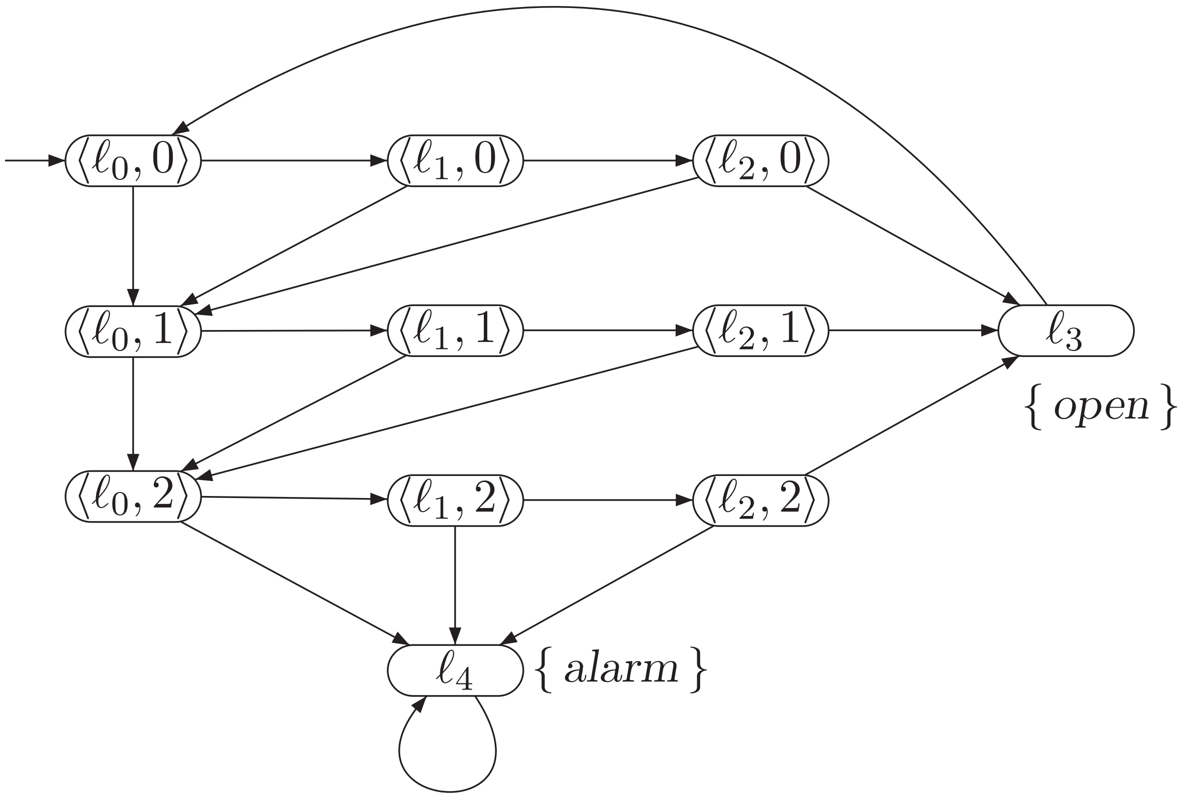
Abstraction functions: example

Example: a door opener

The door opener requires a three-digit code d1 d2 d3 as input with di ∈ { 0, . . . , 9 }. It allows an erroneous digit to be entered, but this may happen at most twice.

The variable *error* keeps track of the number of wrong digits that have been provided, and is initially zero. In case *error* exceeds two, the door opener issues an alarm signal. On a successful input of the door code, the door is opened. Once locked again, it returns to its initial state

Transition System for the door opener



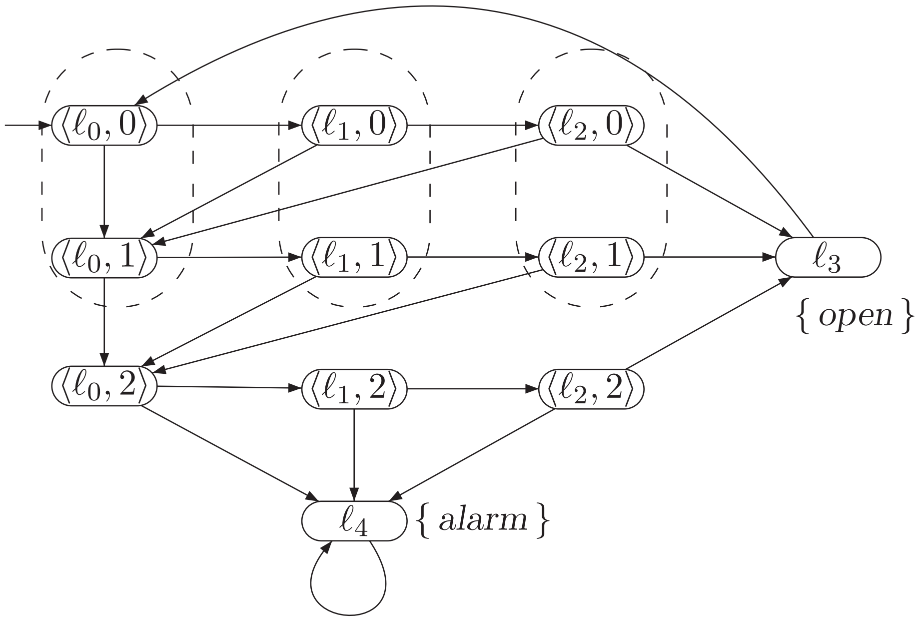
AP = { alarm, open }.

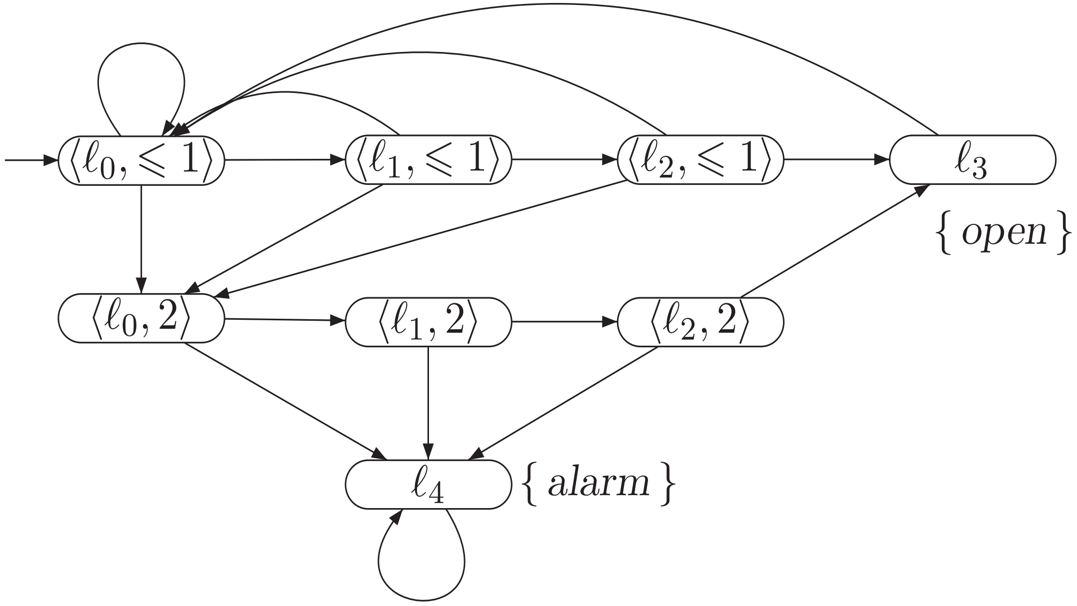
Location ℓ_i (for i = 0, 1, 2) indicates that the first i digits of the code have been correctly entered;
the second component of a state indicates the value of the variable *error* (if applicable).

A first abstraction function

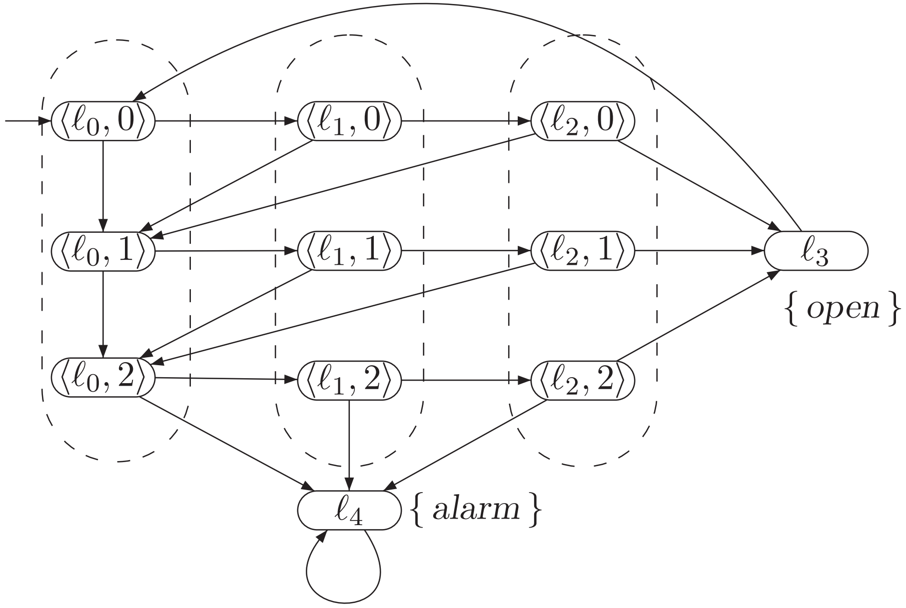
A data abstraction: domain of the variable error is restricted to { <=1, 2 }, i.e., the values 0 and 1 are not distinguished in the abstract transition system.

$$f(\langle \ell, error = k \rangle) = \begin{cases} \langle \ell, error \leqslant 1 \rangle & \text{if } k \in \{0, 1\} \\ \langle \ell, error = 2 \rangle & \text{if } k = 2 \end{cases}$$





Another abstraction function



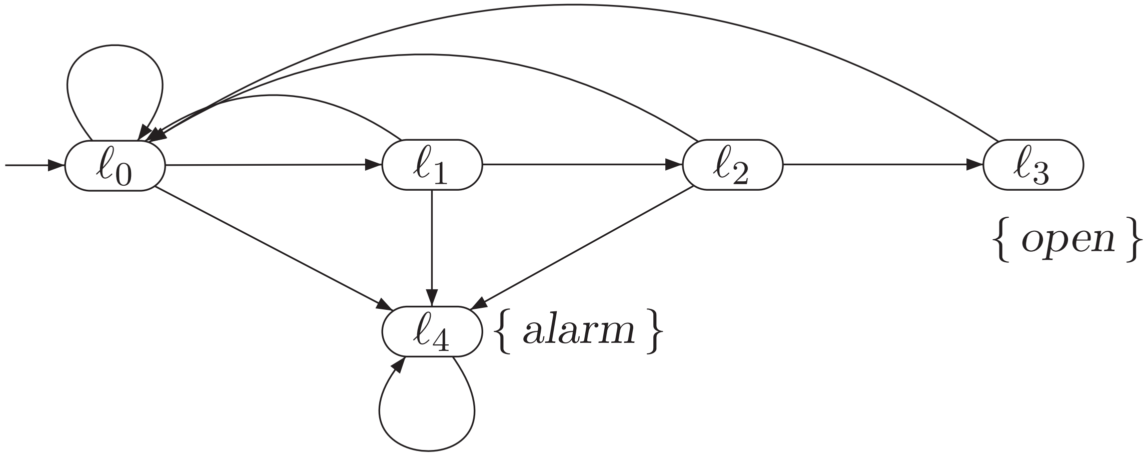


Figure 7.20: Alternative aggregation of states for the door opener.

$$g(\langle \ell, error = k\rangle) = \ell,$$

Stutter Bisimulation

Some brief remarks

Stuttering

- Bisimulation R requires for R-equivalent states s1 and s2 that each transition s1 → t1 is matched by some transition s2 → t2 (and vice versa)
- *Stutter bisimulation R’* allows s1 → t1 to be matched by a *path fragment*:
- s2 u1 u2 ...un t2 (for n >= 0) such that:
- t1 and t2 are R’-equivalent, and each ui is R’equivalent to s2.
- That is, single transitions may be matched by (suitable) path fragments.

Formal definition of Stutter Bisimulation

Let $TS = (S, Act, \rightarrow, I, AP, L)$ be a transition system. A *stutter bisimulation* for TS is a binary relation $\mathcal{R}$ on S such that for all $(s_1, s_2) \in \mathcal{R}$:

1. $L(s_1) = L(s_2)$.
2. If $s_1' \in Post(s_1)$ with $(s_1', s_2) \notin \mathcal{R}$, then there exists a finite path fragment $s_2\, u_1 \, \ldots \, u_n\, s_2'$ with $n \geqslant 0$ and $(s_1, u_i) \in \mathcal{R}$, $i = 1, \ldots, n$ and $(s_1', s_2') \in \mathcal{R}$.
3. If $s_2' \in Post(s_2)$ with $(s_1, s_2') \notin \mathcal{R}$, then there exists a finite path fragment $s_1\, v_1 \, \ldots \, v_n\, s_1'$ with $n \geqslant 0$ and $(v_i, s_2) \in \mathcal{R}$, $i = 1, \ldots, n$ and $(s_1', s_2') \in \mathcal{R}$.

s_1, s_2 are *stutter bisimulation equivalent* (stutter-bisimilar, for short), denoted $s_1 \approx_{TS} s_2$, if there exists a stutter bisimulation $\mathcal{R}$ for TS with $(s_1, s_2) \in \mathcal{R}$. ■

Condition 2 explained

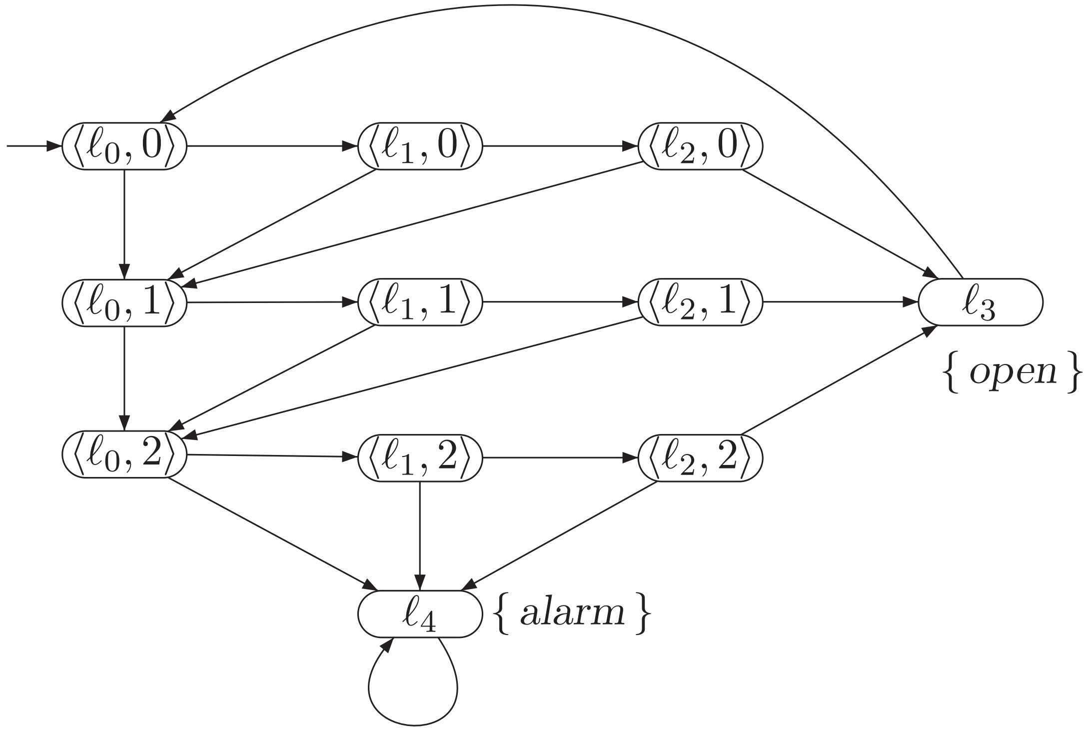
$$\begin{array}{ccc} s_1 & \approx_{TS} & s_2 \\ \downarrow & & \\ s_1' & & \end{array}$$

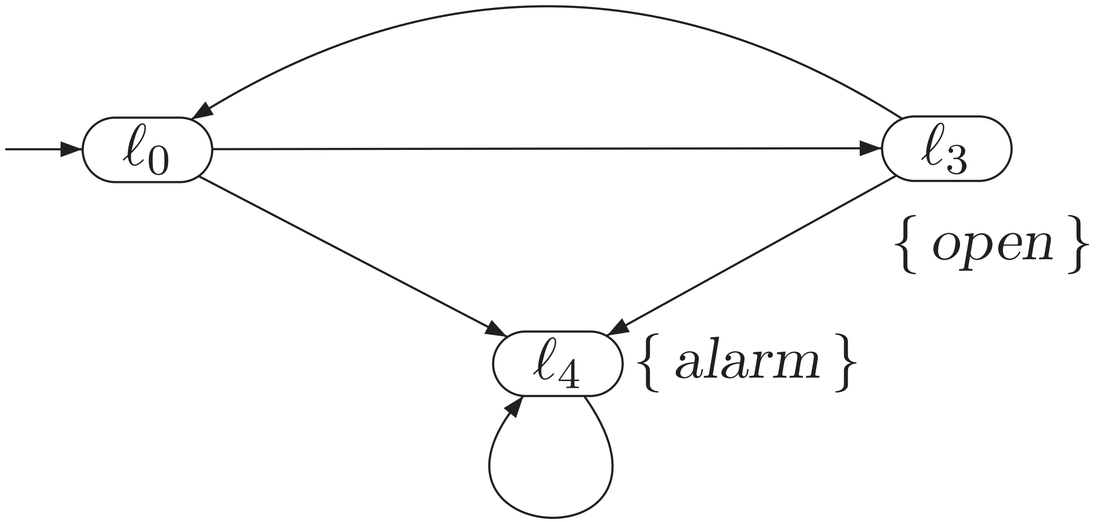
(with $s_1 \not\approx_{TS} s_1'$)

can be completed to

$$\begin{array}{ccc} s_1 & \approx_{TS} & s_2 \\ & & \downarrow \\ s_1 & \approx_{TS} & u_1 \\ & & \downarrow \\ s_1 & \approx_{TS} & u_2 \\ & & \downarrow \\ & & \vdots \\ & & \downarrow \\ s_1 & \approx_{TS} & u_n \\ \downarrow & & \downarrow \\ s_1' & \approx_{TS} & s_2' \end{array}$$

Ex: a stutter bisimilar TS of the door opener





It is an equivalence relation

s_1, s_2 are *stutter bisimulation equivalent* (stutter-bisimilar, for short), denoted $s_1 \approx_{TS} s_2$, if there exists a stutter bisimulation $\mathcal{R}$ for TS with $(s_1, s_2) \in \mathcal{R}$. ■

For transition system TS with state space S:

1. $\approx_{TS}$ *is an equivalence relation on* S.
2. $\approx_{TS}$ *is a stutter bisimulation for TS.*
3. $\approx_{TS}$ *is the coarsest stutter bisimulation for TS and coincides with the union of all stutter bisimulations for TS.*

Stutter-trace equivalence

Transition $s \to s'$ in transition system $TS = (S, Act, \to, I, AP, L)$ is a *stutter step* if $L(s) = L(s')$. ■

Intuitively, a stutter step operates on program or control variables that are either not visible from the outside or viewed to be irrelevant at a certain abstraction level.

The notion of stuttering is lifted to paths as follows. Two paths are called stutter-equivalent if their traces only differ in their stutter steps, i.e., if there is a sequence $A_0 A_1 A_2 \ldots$ of sets of atomic propositions $A_i \subseteq AP$ such that the traces of both paths have the form $A_0^+ A_1^+ A_2^+ \ldots$.

traces σ_1 and σ_2 over 2^{AP} are stutter-equivalent, denoted $\sigma_1 \triangleq \sigma_2$, if they are both of the form $A_0^+ A_1^+ A_2^+ \ldots$ for $A_0, A_1, A_2, \ldots \subseteq AP$. ■

Example: stutter equivalent traces of two paths π_1, π_2

$$
\begin{aligned}
trace(\pi_1) &= \underbrace{A_0 \ldots A_0}_{n_0\text{-times}} \underbrace{A_1 \ldots A_1}_{n_1\text{-times}} \underbrace{A_2 \ldots A_2}_{n_2\text{-times}} \ldots \\
trace(\pi_2) &= \underbrace{A_0 \ldots A_0}_{m_0\text{-times}} \underbrace{A_1 \ldots A_1}_{m_1\text{-times}} \underbrace{A_2 \ldots A_2}_{m_2\text{-times}} \ldots
\end{aligned}
$$

Stutter implementation relations

stutter trace inclusion:

$$TS_1 \trianglelefteq TS_2 \quad \text{iff} \quad \forall \sigma_1 \in \mathit{Traces}(TS_1)\ \exists \sigma_2 \in \mathit{Traces}(TS_2).\ \ \sigma_1 \triangleq \sigma_2$$

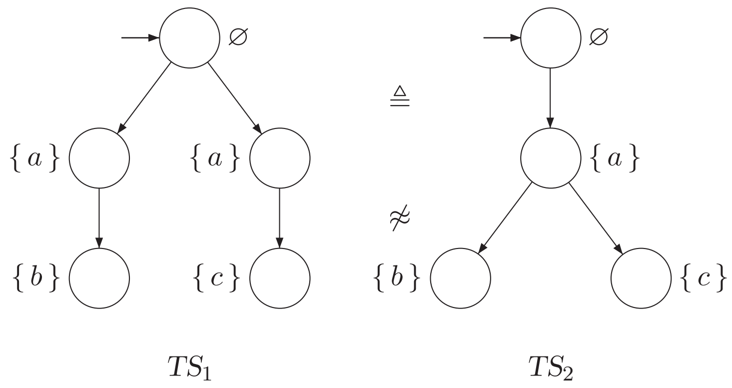
stutter trace equivalence:

$$TS_1 \ \triangleq\ TS_2 \quad \text{iff} \quad TS_1 \trianglelefteq TS_2 \ \text{ and } \ TS_2 \trianglelefteq TS_1$$

stutter bisimulation equivalence:

$$TS_1 \ \approx\ TS_2 \quad \text{iff} \quad \text{there exists a stutter bisimulation for } (TS_1, TS_2)$$

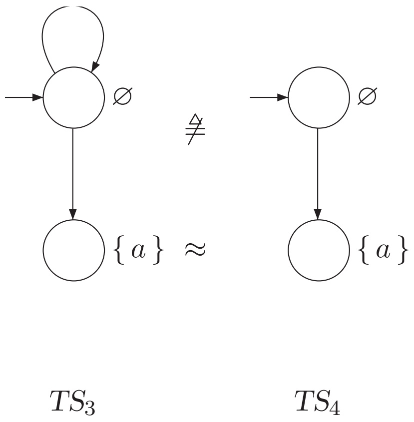
Stuttering: trace and bisimulation are incomparable



$$TS_1 \triangleq TS_2, \text{ but } TS_1 \not\approx TS_2.$$

$$Traces(TS_1) = Traces(TS_2) \text{ and } TS_1 \not\sim TS_2.$$

TS1 and TS2 don't make stutter steps, stuttering is irrelevant.



$$TS_3 \approx TS_4, \text{ while } TS_3 \not\triangleq TS_4.$$

TS_3 exhibits the trace $\emptyset^\omega$, whereas TS_4 cannot generate this trace.
NB: the trace $\emptyset^\omega$ just consists of stutter steps.