

# Semantics and Verifications

*Based on the lectures of Colin RIBA*  
*Notes written by Hugo SALOU*



*September 19, 2025*

# Contents

<b>1</b>	<b>Introduction.</b>	<b>3</b>
<b>2</b>	<b>Transition systems.</b>	<b>5</b>
2.1	Transition systems. . . . .	5
2.2	Program graphs. . . . .	6
2.3	Transition system of a program graph. . . . .	8
<b>3</b>	<b>Linear Time Properties.</b>	<b>11</b>
3.1	Linear-time properties. . . . .	12
3.2	Decomposition of a linear-time property. . . . .	14
3.2.1	Safety properties. . . . .	15
3.2.2	Safety properties and trace equivalences. . . . .	15

# 1 Introduction.

Let us give precisions on the terms in the name of the course, and in the broader space of semantics and verifications.

**Verification.** Formal techniques to ensure the correctness software or hardware of systems.

**Model Checking.** “Automatic” checking of the correctness by means of exhaustive exploration.

**Example 1.1.** Consider a program that is 10 lines long, contains 3 booleans variables and 5 integers variables in the range  $\{0, \dots, 9\}$ . The number of states for this program is:

$$10 \times 2^3 \times 10^5 = 8\,000\,000.$$

The real issue with the state exploration problem is the factor  $10^5$ , coming from the use of 5 integers.

**Example 1.2.** Consider a server and  $n$  clients. Clients can make requests to the server and the server can answer a client. The specification of this server should include the following:

- ▷ Each client which makes a request is eventually answered.
- ▷ We *abstract* away from precise quantitative constraints.

We will sometimes reason about an infinite amount of executions. For example, if some client makes infinitely-many requests (then it'll have infinitely-many answers). Infinite sequences are represented by  $\omega$ -words, *i.e.* infinite words indexed by  $\mathbb{N}$ . Thus,  $\omega$ -words on some

alphabet  $\Sigma$  are functions  $\mathbb{N} \rightarrow \Sigma$ . We will denote  $\Sigma^\omega$  the set of those infinite words on the alphabet  $\Sigma$ .

If  $|\Sigma| \geq 2$ , then the set  $\Sigma^\omega$  is uncountable.

This course will cover the following:

- ▷ Transition systems;
- ▷ Linear-time properties;
- ▷ Topology;
- ▷ Orders and Lattices;
- ▷ Linear Temporal Logic (LTL);
- ▷ Büchi automata;
- ▷ **Stone duality** (mostly in homework);
- ▷ Bisimilarity/bisimulation;
- ▷ Modal Logic.

Ressources from this course include:

- ▷ the [course notes](#) (available online, non-exhaustive);
- ▷ Baier, C. and Katoen, J.-P., Principles of Model Checking, MIT Press, 2008.

Prerequisites for this course include:

- ▷ First-order logic (*see my [course notes](#) for the “Logique” L3 course, in french*);
- ▷ Finite automata (“FDI” L3 course).

Evaluation for this course will be in two parts: the final exam (50 %) and the homework, in two parts (25 % each).

The tutorials will be done by Lison Blondeau-Patissier.

# 2 Transition systems.

## 2.1 Transition systems.

**Definition 2.1.** A transition system is a tuple

$$TS = (S, \text{Act}, \rightarrow, I, \text{AP}, L)$$

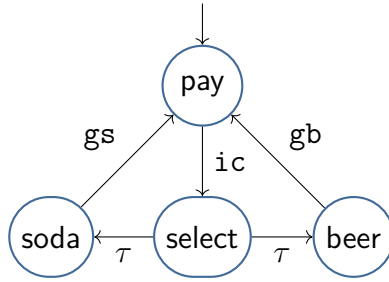
where

- ▷  $S$  is the set of *states*;
- ▷  $\text{Act}$  is the set of *actions*;
- ▷  $\rightarrow \subseteq S \times \text{Act} \times S$  the *transition relation*;
- ▷  $I \subseteq S$  the set of *initial states*;
- ▷  $\text{AP}$  is the set of *atomic propositions*;
- ▷  $L : S \rightarrow \wp(\text{AP}) \cong 2^{\text{AP}}$  is the *state labelling function*.

We will write  $s \xrightarrow{\alpha} s'$  when  $(s, \alpha, s') \in \rightarrow$ .

**Example 2.1 (Beverage Vending Machine, BVM).** We can model a beverage vending machine using a diagram like in figure 2.1. Here we have that:

- ▷  $S = \{\text{pay}, \text{select}, \text{soda}, \text{beer}\},$
- ▷  $I = \{\text{pay}\},$
- ▷  $\text{Act} = \{\text{ic}, \tau, \text{gb}, \text{gs}\}.$ <sup>1</sup>



**Figure 2.1** | Transition system for the BVM

We can define the labels:

$L(\text{pay}) = \emptyset$   $L(\text{soda}) = L(\text{beer}) = \{\text{paid, drink}\}$   $L(\text{select}) = \{\text{paid}\}$ ,  
with  $\text{AP} = \{\text{paid, drink}\}$ .

## 2.2 Program graphs.

The goal is to represent the evaluation of a program.

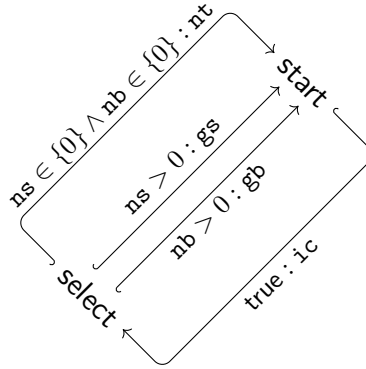
**Definition 2.2 (Typed variables).**  $\triangleright$  A set  $\text{Var}$  of *variables*.

- $\triangleright$  For each variable  $x \in \text{Var}$ , consider a set  $\text{Dom}(x)$ .
- $\triangleright$  Given  $TV = (\text{Var}, (\text{Dom}(x))_{x \in \text{Var}})$ , we define

$$\text{Eval}(TV) = \prod_{x \in \text{Var}} \text{Dom}(x),$$

the set of valuations of the form  $\eta : x \in \text{Var} \mapsto \eta(x) \in \text{Dom}(x)$  (in the sense of a dependent function type).

<sup>1</sup>The meaning of the actions are the following: **ic** means *insert coin*, **gb** means *get beer* and **gs** for *get soda*.



**Figure 2.2** | BVM as a program graph

**Definition 2.3 (Program graph).** A *program graph* is a tuple

$$PG = (\text{Loc}, \text{Act}, \text{Effect}, \hookrightarrow, \text{Loc}_0, g_0),$$

where

- ▷ Loc is the set of *locations* (lines of codes);
- ▷ Act is the set of *actions*;
- ▷ Effect : Act  $\times$  Eval(TV)  $\rightarrow$  Eval(TV);
- ▷  $\hookrightarrow \subseteq \text{Loc} \times \text{Conditions} \times \text{Act} \times \text{Loc}$  where conditions are propositional formula built from atoms of the forms “ $x \in D$ ” for some variable  $x$  and some set  $D \subseteq \text{Dom}(x)$ ;
- ▷  $\text{Loc}_0 \subseteq \text{Loc}$  the set of *initial locations*;
- ▷  $g_0$  is the *initial condition*.

We will write  $\ell \xrightarrow{g:\alpha} \ell'$  for  $(\ell, g, \alpha, \ell') \in \hookrightarrow$ .

**Example 2.2 (BVM as a program graph).** In figure 2.2, we use

- ▷ Loc = {start, select};

- ▷  $\text{Var} = \{\text{ns}, \text{nb}\};$
- ▷  $\text{Act} = \{\text{ic}, \text{nt}, \text{gs}, \text{gb}, \text{refill}\};$
- ▷  $\text{Loc}_0 = \{\text{start}\};$
- ▷  $g_0 = \text{ns} \in \{\text{max}\} \wedge \text{nb} \in \{\text{max}\}$
- ▷

$$\begin{aligned}
 \text{Effect} : \text{Act} \times \text{Eval}(TV) &\longrightarrow \text{Eval}(TV) \\
 (\text{refill}, \eta) &\longmapsto [\text{ns} \mapsto \text{max}, \text{nb} \mapsto \text{max}] \\
 (\text{gs}, \eta) &\longmapsto \eta[\text{ns} \mapsto \eta(\text{ns}) - 1] \\
 (\text{gb}, \eta) &\longmapsto \eta[\text{nb} \mapsto \eta(\text{nb}) - 1]
 \end{aligned}$$

## 2.3 Transition system of a program graph.

**Definition 2.4.** Given  $TV$  and  $PG$  a program graph, we define

$$TS(PG) := (S, \text{Act}, \rightarrow, I, \text{AP}, L)$$

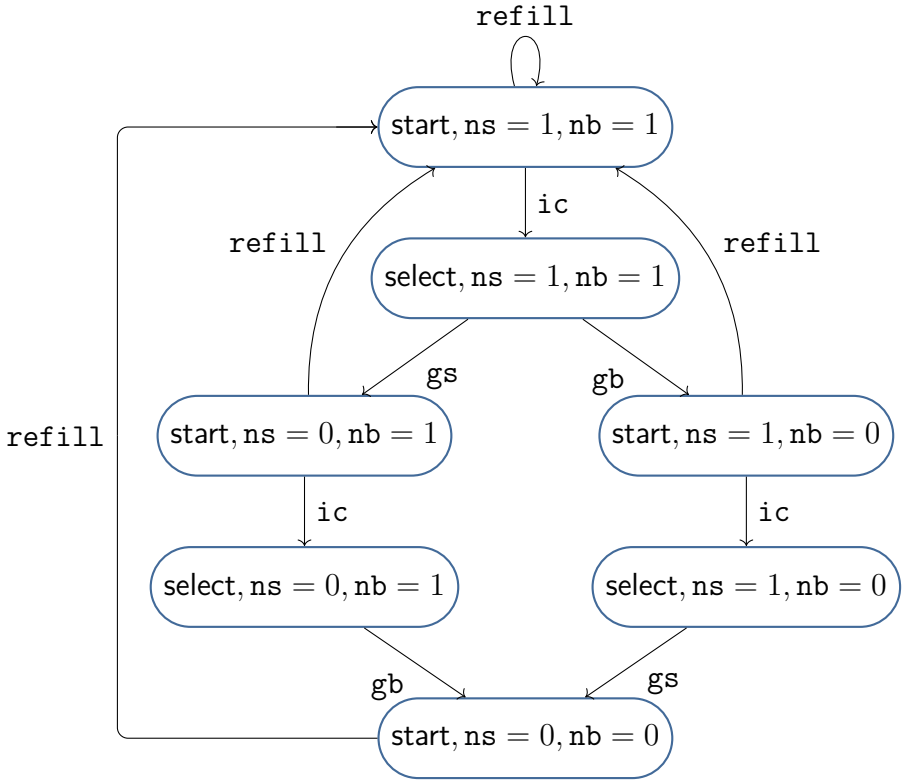
where

- ▷  $S = \text{Loc} \times \text{Eval}(TV);$
- ▷  $\text{AP} = \text{Loc} \cup \text{Conditions} ;$
- ▷  $I = \{(\ell_0, \eta) \mid \ell_0 \in \text{Loc}_0, \eta \models g_0\};$
- ▷  $\rightarrow$  is defined by:

$$\frac{\ell \xrightarrow{g:\alpha} \ell' \quad \eta \models g}{(\ell, \eta) \xrightarrow{\alpha} (\ell', \text{Effect}(\alpha, \eta))},$$

- ▷ and  $L(\ell, \eta) = \{\ell\} \cup \{g \mid \eta \models g\}.$





**Figure 2.3** | *Transition system of the BVM program graph*

**Example 2.3.** The BVM program graph example seen in the previous example can be transformed as a transition system thanks to the previous definition; it is shown in figure 2.3. To simplify, we assume  $\text{max} = 1$ .

# 3 Linear Time Properties.

**Definition 3.1.** Let  $\Sigma$  be an alphabet (*i.e.* a set).

1. A  $\omega$ -word on  $\Sigma$  is a function  $\sigma : \mathbb{N} \rightarrow \Sigma$ . We denote  $\Sigma^\omega$  for the set of  $\omega$ -words on  $\Sigma$ .
2. We define  $\Sigma^\infty := \Sigma^\omega \cup \Sigma^*$  the set of finite or infinite words.
3. Given  $\hat{\sigma} \in \Sigma^*$  and  $\sigma \in \Sigma^\infty$ , we say that  $\hat{\sigma}$  is a prefix of  $\sigma$ , written  $\hat{\sigma} \subseteq \sigma$ , whenever

$$\forall i < \text{length}(\hat{\sigma}), \quad \hat{\sigma}(i) = \sigma(i).$$

4. Given  $\sigma \in \Sigma^\infty$ , we define

$$\text{Pref}(\sigma) := \{ \hat{\sigma} \in \Sigma^* \mid \hat{\sigma} \subseteq \sigma \},$$

which we extend to sets of words: for  $E \subseteq \Sigma^\infty$ ,

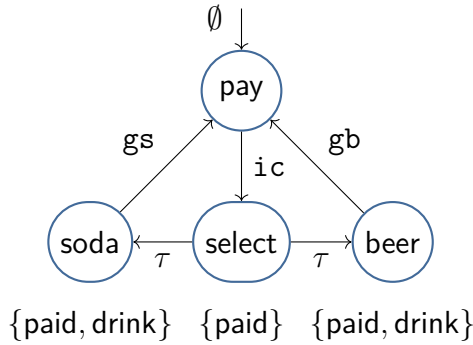
$$\text{Pref}(E) := \bigcup_{\sigma \in E} \text{Pref}(\sigma).$$

**Remark 3.1.**  $\triangleright$  The prefix order  $\subseteq$  on  $\Sigma^*$  is generally<sup>1</sup> a partial order: there are  $u, v \in \Sigma^*$  such that  $u \not\subseteq v$  and  $v \not\subseteq u$ .

- $\triangleright$  Given  $\sigma \in \Sigma^\infty$ , the prefix order  $\subseteq$  on  $\text{Pref}(\sigma)$  is a linear (or total order).

---

<sup>1</sup>As long as the alphabet has at least two letters.



**Figure 3.1** | Transition system for the BVM with labels

### 3.1 Linear-time properties.

Let  $\text{AP}$  be a set of *atomic propositions*.

**Definition 3.2.** A *linear-time property* (sometimes written LT property) on  $\text{AP}$  is a set  $P \subseteq (\mathbf{2}^{\text{AP}})^\omega$ .

The idea is that a linear-time property  $A : \mathbb{N} \rightarrow \mathbf{2}^{\text{AP}}$  specifies, for each  $i \in \mathbb{N}$ , a set  $\sigma(i) \subseteq \text{AP}$  of all atomic propositions are assumed at time  $i$ .

**Example 3.1.** For the Beverage vending machine (shown in figure 3.1), we can have the following linear-time properties:

- ▷  $\{\sigma \in (\mathbf{2}^{\text{AP}})^\omega \mid \forall n \in \mathbb{N}, \text{drink} \in \sigma(n) \implies \exists k < n, \text{paid} \in \sigma(k)\},$
- ▷  $\{\sigma \in (\mathbf{2}^{\text{AP}})^\omega \mid \forall n \in \mathbb{N}, \#\{k \leq n \mid \text{drink} \in \sigma(k)\} \leq \#\{k \leq n \mid \text{paid} \in \sigma(k)\}\},$
- ▷  $\{\sigma \in (\mathbf{2}^{\text{AP}})^\omega \mid (\exists^\infty t, \text{paid} \in \sigma(t)) \implies (\exists^\infty t, \text{drink} \in \sigma(t))\},$
- ▷  $\{\sigma \in (\mathbf{2}^{\text{AP}})^\omega \mid (\forall^\infty t, \text{paid} \notin \sigma(t)) \implies (\forall^\infty t, \text{drink} \notin \sigma(t))\}.$

**Remark 3.2.** The notations  $\exists^\infty$  and  $\forall^\infty$  are “infinitely many” and “ultimately all” quantifiers:

- ▷  $\exists^\infty t, P(t)$  is, by definition,  $\forall N \in \mathbb{N}, \exists t \geq N, P(t);$

▷  $\forall^{\infty} t, P(t)$  is, by definition,  $\exists N \in \mathbb{N}, \forall t \geq N, P(t)$ .

**Definition 3.3.** A (finite or infinite) *path* in  $TS$  is a finite or infinite sequence  $\pi = (s_i)_i \in S^{\infty}$  which respects transitions: for all  $i$ , we have  $s_i \xrightarrow{a} s_{i+1}$  for some  $a \in \text{Act}$ .

A path  $\pi = (s_i)_i$  is *initial* if  $s_0 \in I$ .

**Definition 3.4 (Trace).** 1. The *trace* of a path  $\pi = (s_i)_i$  is the (finite or infinite) word

$$L(\pi) := (L(s_i))_i \in L^{\infty}.$$

2. We define

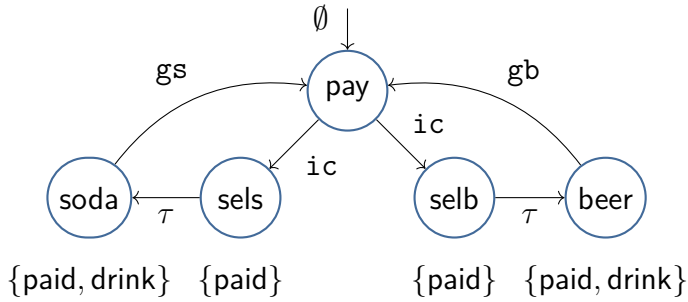
- ▷  $\text{Tr}(TS) := \{L(\pi) \mid \pi \text{ is a finite or infinite path in } TS\};$
- ▷  $\text{Tr}^{\omega}(TS) := \{L(\pi) \mid \pi \text{ is a infinite path in } TS\};$
- ▷  $\text{Tr}_{\text{fin}}(TS) := \{L(\pi) \mid \pi \text{ is a finite path in } TS\}.$

**Definition 3.5 (Satisfaction of a LT property).** We say that a transition system  $TS$  over AP *satisfies* a LT property  $P$  on AP, written  $TS \approx P$ , when  $\text{Tr}^{\omega}(TS) \subseteq P$ .

**Example 3.2.** The BVM satisfies all the properties from example 3.1.

**Example 3.3.** We use a different transition system  $\text{BVM}'$  to model the beverage vending machine, as seen in figure 3.2. The two transition systems are equivalent in the sense that:

$$\text{Tr}^{\omega}(\text{BVM}') = \text{Tr}^{\omega}(\text{BVM}),$$



**Figure 3.2** | Transition system for the alternative BVM

so, for any LT Property  $P \subseteq (\mathbf{2}^{\text{AP}})^\omega$ ,

$$\text{BVM}' \models P \quad \text{iff} \quad \text{BVM} \models P.$$

We have a very simple result, which we will (probably) prove in the tutorials.

**Proposition 3.1.** Given two transition systems  $TS_1$  and  $TS_2$  over AP, then the following are equivalent:

- ▷  $\text{Tr}^\omega(TS_1) \subseteq \text{Tr}^\omega(TS_2)$ ,
- ▷  $\forall P \subseteq (\mathbf{2}^{\text{AP}})^\omega, TS_2 \models P \implies TS_1 \models P$ .

## 3.2 Decomposition of a linear-time property.

In this section, we introduce the notions of a “safety property” and a “liveness property” such that, for any LT property  $P$ ,

1. there exists a safety property  $P_{\text{safe}}$  and a liveness property  $P_{\text{liveness}}$  such that

$$P = P_{\text{safe}} \cap P_{\text{liveness}};$$

2.  $P$  is a liveness and a safety property if and only if  $P = (\mathbf{2}^{\text{AP}})^\omega$ .

### 3.2.1 Safety properties.

The idea of a safety property is to ensure that “nothing bad is going to happen.”

**Definition 3.6.** We say that  $P \subseteq (2^{\text{AP}})^\omega$  is a *safety property* if there exists a set  $P_{\text{bad}} \subseteq (2^{\text{AP}})^*$  such that

$$\sigma \in P \iff \text{Pref}(\sigma) \cap P_{\text{bad}} = \emptyset.$$

**Example 3.4.** Considering the examples of LT-properties from example 3.1,

- ▷ Property (1) is a safety property: we can consider

$$P_{\text{bad}}^{(1)} = \{\hat{\sigma} \in \Sigma^* \mid \text{drink} \in \hat{\sigma}(n) \wedge \forall i < n, \text{paid} \notin \hat{\sigma}(i)\},$$

where  $n$  is the length of  $\hat{\sigma}$ .

- ▷ Property (2) is a safety property: we can consider

$$P_{\text{bad}}^{(2)} = \{\hat{\sigma} \in \Sigma^* \mid \#\{t \mid \text{paid} \in \hat{\sigma}(t)\} < \#\{t \mid \text{drink} \in \hat{\sigma}(t)\}\}.$$

- ▷ Properties (3) and (4) are not safety properties: for any finite word  $\hat{\sigma} \in (2^{\text{AP}})^*$ , there exists  $\sigma \in (2^{\text{AP}})^\omega$  such that  $\hat{\sigma} \subseteq \sigma$  and  $\sigma \in P$ .

**Example 3.5 (Traffic Light).** We consider a traffic light as a transition system over  $\text{AP} = \{\text{G}, \text{Y}, \text{R}\}$ , as shown in figure 3.3. An example of a safety property is

$$\forall n, \text{R} \in \sigma(n) \implies n > 0 \text{ and } \text{Y} \in \sigma(n-1).$$

### 3.2.2 Safety properties and trace equivalences.

**Example 3.6.** Consider the transition system shown in figure 3.4, a safety property  $P$  with  $P_{\text{bad}} = \{\mathbf{a}\}^* \{\mathbf{b}\}$  is satisfied:  $TS \approx P$ . This is true since  $\text{Tr}^\omega(TS) = \{\mathbf{a}\}^\omega$ . However, when we consider *finite* (instead of infinite) traces, we have that  $\text{Tr}_{\text{fin}}(TS) \cap P_{\text{bad}} \neq \emptyset$ .

**Definition 3.7 (Terminal state).** A state  $s \in S$  of a transition system  $TS$  is *terminal* if

$$\forall s' \in S, \quad \forall \alpha \in \text{Act}, \quad s \not\stackrel{\alpha}{\rightarrow} s'.$$

**Proposition 3.2.** Let  $TS$  be a transition system without terminal states, and a safety property  $P$  with  $P_{\text{bad}}$  the set of “bad behaviours”. Then,

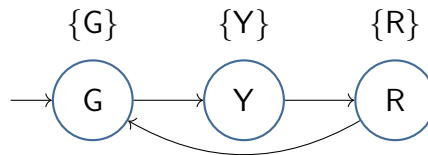
$$TS \approx P \quad \text{if and only if} \quad \text{Tr}_{\text{fin}}(TS) \cap P_{\text{bad}} = \emptyset.$$

**Proof.** See the course notes in §3.2.3. □

**Lemma 3.1.** Let  $TS$  and  $TS'$  be two transition systems over AP without terminal states. Then, the following are equivalent:

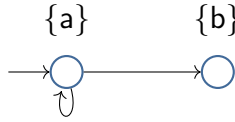
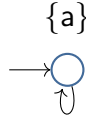
- ▷  $\text{Tr}_{\text{fin}}(TS) \subseteq \text{Tr}_{\text{fin}}(TS')$ ;
- ▷ for any safety property  $P$ ,  $TS' \approx P$  implies  $TS \approx P$ .

**Proof.** ▷ “ $\implies$ ”. This is true by the last proposition.



**Figure 3.3** | Transition system for the traffic light



**Figure 3.4** | *Another transition system***Figure 3.5** | *One-state transition system*

▷ “ $\Leftarrow$ ”. Let  $P$  be a safety property with

$$P_{\text{bad}} = (2^{\text{AP}})^* \setminus \text{Tr}_{\text{fin}}(TS').$$

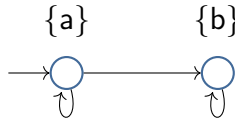
So,  $TS' \models P$  hence  $TS \models P$  by assumption. Therefore,  $\text{Tr}_{\text{fin}}(TS) \subseteq \text{Tr}_{\text{fin}}(TS')$  by the last proposition.

□

**Example 3.7.** Consider the transition system  $TS$  from figure 3.4 (example 3.6, which has a terminal state), and the transition system  $TS'$  from figure 3.5. Safety properties satisfied in  $TS'$  are satisfied in  $TS$ . However,  $\text{Tr}_{\text{fin}}(TS) \not\subseteq \text{Tr}_{\text{fin}}(TS')$  (even though the sets of infinite traces are equal).

**Example 3.8.** Consider the transition system  $TS'$  shown in figure 3.7 (page 19) and  $TS$  the transition system shown in figure 3.6. The transition system  $TS'$  has terminal states and  $TS$  does not. However, we have that

$$\text{Tr}_{\text{fin}}(TS) = \text{Tr}_{\text{fin}}(TS').$$

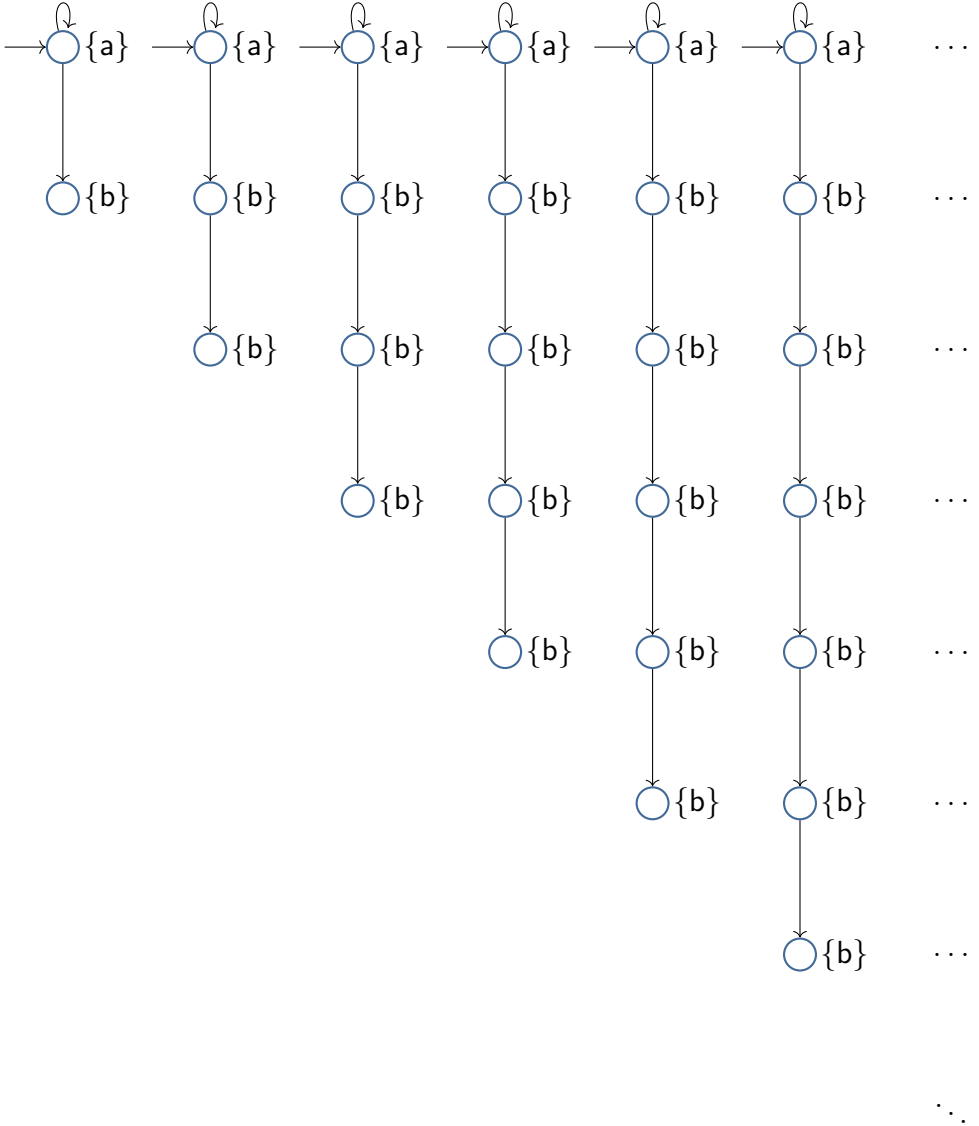


**Figure 3.6** | A transition system with no terminal state

We also have that

$$\text{Tr}^\omega(TS') = \{a\}^\omega \quad \text{and} \quad \text{Tr}^\omega(TS) = \{a\}^\omega \cup \{a\}^+ \cdot \{b\}^\omega.$$

Thus giving a counter example to the previous lemma if one of the transition system has terminal states: consider a linear-time property  $P$  with the set of “bad behaviors” as  $P_{\text{bad}} = \{a\}^+ \cdot \{b\}$ , then  $TS' \approx P$  but  $TS \not\approx P$ .



**Figure 3.7** | *An infinite transition system*