Semantics and Verifications

Based on the lectures of Colin Riba
Notes written by Hugo Salou

September 17, 2025

Contents

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 ω-*words*, *i.e.* infinite words indexed by $\mathbb{N}$. Thus, ω-words on some

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

If $|\Sigma| \geq 2$, then the set Σ^ω 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*;
- 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*;
- AP is the set of *atomic propositions*;
- $L : S \to \wp(\text{AP}) \cong \mathbf{2}^{\text{AP}}$ 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:

- $S = \{\mathsf{pay}, \mathsf{select}, \mathsf{soda}, \mathsf{beer}\}$,
- $I = \{\mathsf{pay}\}$,
- $\text{Act} = \{\mathtt{ic}, \tau, \mathtt{gb}, \mathtt{gs}\}$.[1]

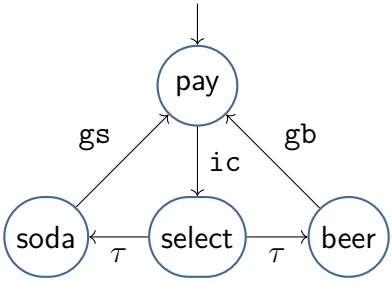

Figure 2.1 | *Transition system for the BVM*

We can define the labels:

$L(\mathsf{pay}) = \emptyset \quad L(\mathsf{soda}) = L(\mathsf{beer}) = \{\mathsf{paid}, \mathsf{drink}\} \quad L(\mathsf{select}) = \{\mathsf{paid}\},$

with $\mathrm{AP} = \{\mathsf{paid}, \mathsf{drink}\}$.

2.2 Program graphs.

The goal is to represent the evaluation of a program.

Definition 2.2 (Typed variables). ▷ A set Var of *variables*.

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

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

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

[1] The meaning of the actions are the following: `ic` means *insert coin*, `gb` means *get beer* and `gs` for *get soda*.

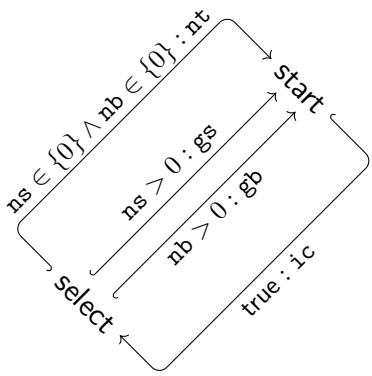

Figure 2.2 | *BVM as a program graph*

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*;
- ▷ $\text{Effect} : \text{Act} \times \text{Eval}(TV) \to \text{Eval}(TV)$;
- ▷ $\hookrightarrow \subseteq \text{Loc} \times \text{Conditions} \times \text{Act} \times \text{Loc}$ 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 $\ell \xhookrightarrow{g:\alpha} \ell'$ for $(\ell, g, \alpha, \ell') \in \hookrightarrow$.

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

- ▷ $\text{Loc} = \{\mathsf{start}, \mathsf{select}\}$;

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

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

2.3 Transition system of a program graph.

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

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

where

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

$$\frac{\ell \overset{g:\alpha}{\hookrightarrow} \ell' \qquad \eta \models g}{(\ell, \eta) \overset{\alpha}{\to} (\ell', \text{Effect}(\alpha, \eta))},$$

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

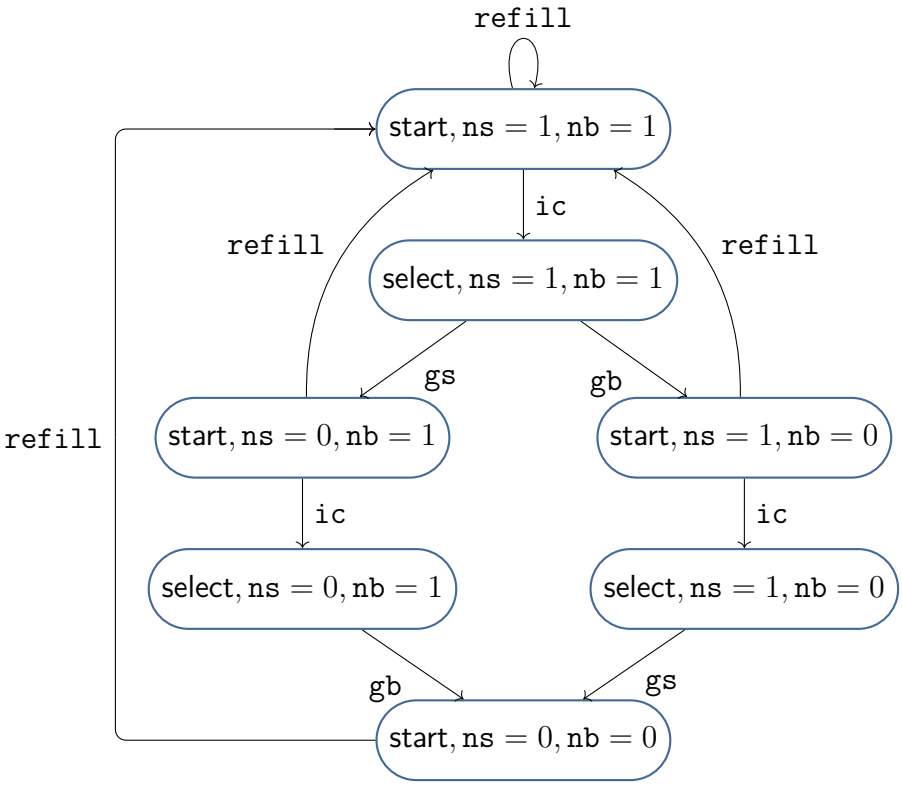

Figure 2.3 | *Transition system of the BVM program graph*

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 max = 1.

3 Linear Time Properties.

Definition 3.1. Let Σ be an alphabet (*i.e.* a set).

1. A *ω-word* on Σ is a function $\sigma : \mathbb{N} \to \Sigma$. We denote Σ^ω for the set of ω-words on Σ.
2. We define $\Sigma^\infty := \Sigma^\omega \cup \Sigma^\star$ the set of finite or infinite words.
3. Given $\hat{\sigma} \in \Sigma^\star$ and $\sigma \in \Sigma^\infty$, we say that $\hat{\sigma}$ is a prefix of σ, written $\hat{\sigma} \subseteq \sigma$, whenever
$$\forall i < \mathsf{length}(\hat{\sigma}), \quad \hat{\sigma}(i) = \sigma(i).$$
4. Given $\sigma \in \Sigma^\infty$, we define
$$\mathrm{Pref}(\sigma) := \{\, \hat{\sigma} \in \Sigma^\star \mid \hat{\sigma} \subseteq \sigma \,\},$$
which we extend to sets of words: for $E \subseteq \Sigma^\infty$,
$$\mathrm{Pref}(E) := \bigcup_{\sigma \in E} \mathrm{Pref}(\sigma).$$

Remark 3.1.
- ▷ The prefix order $\subseteq$ on $\Sigma^\star$ is generally a partial order: there are $u, v \in \Sigma^\star$ such that $u \not\subseteq v$ and $v \not\subseteq u$.
- ▷ Given $\sigma \in \Sigma^\infty$, the prefix order $\subseteq$ on $\mathrm{Prefix}(\sigma)$ is a linear (or total order).

3.1 Linear-time properties.

Let AP be a set of *atomic propositions*.

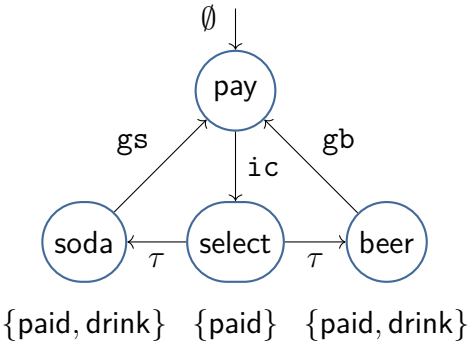

Figure 3.1 | *Transition system for the BVM with labels*

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

The idea is that a linear-time property $A : \mathbb{N} \to \mathbf{2}^{\mathrm{AP}}$ specifies, for each $i \in \mathbb{N}$, a set $\sigma(i) \subseteq \mathrm{AP}$ 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:

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

Remark 3.2. The notations $\exists^{\infty}$ and $\forall^{\infty}$ are "infinitely many" and "ultimately all" quantifiers:

- $\forall^{\infty} t, P(t)$ is, by definition, $\forall N \in \mathbb{N}, \exists t \geq N, P(t)$;
- $\exists^{\infty} t, P(t)$ 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 $s_i \xrightarrow{\mathtt{a}} s_{i+1}$ for some $\mathtt{a} \in \mathrm{Act}$.

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
$$L(\pi) := \big(L(s_i)\big)_i \in L^\infty.$$
2. We define
   - ▷ $\mathrm{Tr}(TS) := \{L(\pi) \mid \pi \text{ is a finite or infinite path in } TS\}$;
   - ▷ $\mathrm{Tr}^\omega(TS) := \{L(\pi) \mid \pi \text{ is a infinite path in } TS\}$;
   - ▷ $\mathrm{Tr}_{\mathrm{fin}}(TS) := \{L(\pi) \mid \pi \text{ is a finite path in } TS\}$.

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 $TS \models P$, when $\mathrm{Tr}^\omega(TS) \subseteq P$.

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

Example 3.3. We use a different transition system BVM′ to model the beverage vending machine, as seen in figure 3.2. The two transition systems are equivalent in the sense that:
$$\mathrm{Tr}^\omega(\mathrm{BVM}') = \mathrm{Tr}^\omega(\mathrm{BVM}),$$
so, for any LT Property $P \subseteq (\mathbf{2}^{\mathrm{AP}})^\omega$,
$$\mathrm{BVM}' \models P \qquad \text{iff} \qquad \mathrm{BVM} \models P.$$

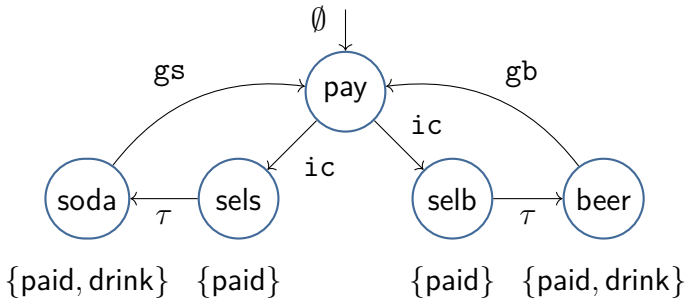

Figure 3.2 | *Transition system for the alternative BVM*

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:

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

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_{safe} and a liveness property P_{liveness} such that
$$P = P_{\text{safe}} \cap P_{\text{liveness}};$$
2. P is a liveness and a safety property if and only if $P = (\mathbf{2}^{\mathrm{AP}})^\omega$.

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 $P \subseteq (\mathbf{2}^{\mathrm{AP}})^\omega$ is a *safety property* if there exists a set $P_{\mathrm{bad}} \subseteq (\mathbf{2}^{\mathrm{AP}})^\star$ such that

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

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

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

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

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

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

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

- ▷ Properties (3) and (4) are not safety properties: for any finite word $\hat{\sigma} \in (\mathbf{2}^{\mathrm{AP}})^\omega$, there exists $\sigma \in (\mathbf{2}^{\mathrm{AP}})^\omega$ 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 $\mathrm{AP} = \{\mathsf{G}, \mathsf{Y}, \mathsf{R}\}$, as shown in figure 3.3. An example of a safety property is

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

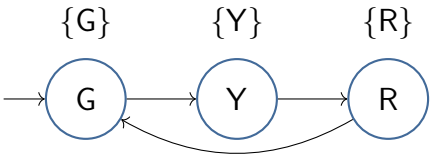

Figure 3.3 | *Transition system for the traffic light*

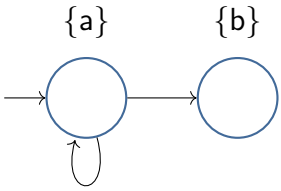

Figure 3.4 | *Transition system for the traffic light*

Example 3.6. Consider the transition system shown in figure 3.4, a safety property P with $P_{\text{bad}} = \{\mathsf{a}\}^\star\{\mathsf{b}\}$ is satisfied: $TS \not\approx P$. This is true since $\text{Tr}^\omega(TS) = \{\mathsf{a}\}^\omega$. However, when we consider *finite* (instead of infinites) 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

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

Proposition 3.2. Let TS be a transition system without terminal states, and a safety property P with the set of "bad behaviours" is written P_{bad}. Then,

$$TS \not\approx P \qquad \text{if and only if } \text{Tr}_{\text{fin}}(TS) \cap P_{\text{bad}} = \emptyset.$$

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

3.2.2 Safety properties and trace equivalences.

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, $TS' \not\approx P$ implies $TS \not\approx P$.

Proof. ▷ " $\Longrightarrow$ ". This is true by the last proposition.

▷ " $\Longleftarrow$ ". Let P be a safety property with

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

So, $TS' \not\approx P$ hence $TS \not\approx P$ by assumption. Therefore, $\text{Tr}_{\text{fin}}(TS) \subseteq \text{Tr}_{\text{fin}}(TS')$ by the last proposition.

□