Preliminary Notes on Model Checking

CS511

November 26, 2024[*]

These notes introduce basic notions on state-based model-checking. They are based on [BK08]. For full details and further examples and material, the reader is referred to *op.cit.*.

1 Transition Systems

Definition 1 (TS)**.** A Transition System (TS) $\mathcal{T}$ is a tuple $\mathcal{T} = (S, Act, \to, I, AP, L)$ where

- S is a set of states,
- Act is a set of actions,
- $\to \subseteq S \times Act \times S$ is a transition relation,
- $I \subseteq S$ is a set of initial states,
- AP is a set of atomic propositions[1], and
- $L : S \to 2^{AP}$ is a labeling function.

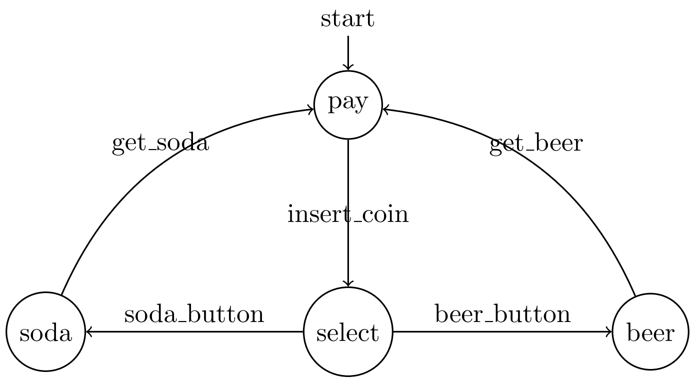
A TS is *finite* if S,Act and AP are finite.

We typically write $s \overset{\alpha}{\to} s'$ for $(s, \alpha, s') \in \to$. Also, $L(s)$ are the set of atomic propositions in AP that are satisfied at state s.

Example 2 (Beverage vending machine)**.** A beverage vending machine may be modeled as the TS $(S, Act, \to, I, AP, L)$ depicted below (AP and L are described further down):

[*]V0.023

[1]A "proposition" is a sentence that can be true or false.



From the diagram above we can see that the set of states is $S = \{pay, select, soda, beer\}$ and the set of actions is $Act = \{insert\_coin, soda\_button, beer\_button, get\_soda, get\_beer\}$.

The atomic propositions AP are not illustrated in the diagram above. We can fix AP to be any set of atomic propositions that allows us to formulate properties that we are interested in, about the executions of the beverage vending machine. For example, if we wanted to be able to state that one can only get a drink if it was payed for, then we might define the following set of atomic propositions:

$$AP = \{paid, got\_drink\}$$

and the following assignments of atomic propositions to states (labelling):

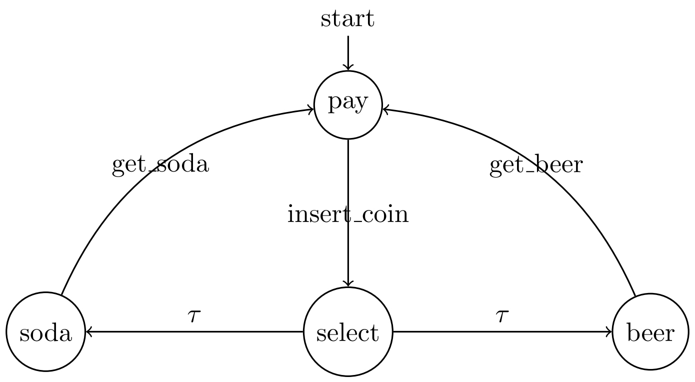
$$\begin{aligned} L(pay) &= \emptyset \\ L(select) &= \{paid\} \\ L(soda) = L(beer) &= \{paid, got\_drink\} \end{aligned}$$

Remark 3. Actions are used to model communication between a transition system and its environment. For example, the *insert_coin* action allows the software controlling the operation of the vending machine to interact with the software controlling the sensor that detects that a coin has been inserted. Interaction is modeled sharing this action in their corresponding transitions, those of the software controller and those of the machine itself, and then synchronizing on them.

We will mostly be working with transition systems that result from all possible interleavings of a set of sequential threads. As a consequence, these transition systems are "closed" in the sense that they are self-contained (they do not communicate with their environment). For such systems the set of actions is taken to be $Act = \{\tau\}$. The tau action thus models some "internal action".

Example 4 (Beverage vending machine II)**.** Same as above except this one non-deterministically chooses the drink for the user rather than detecting buttons being pushed.

$Act := \{get\_soda, get\_beer, insert\_coin, \tau\}$

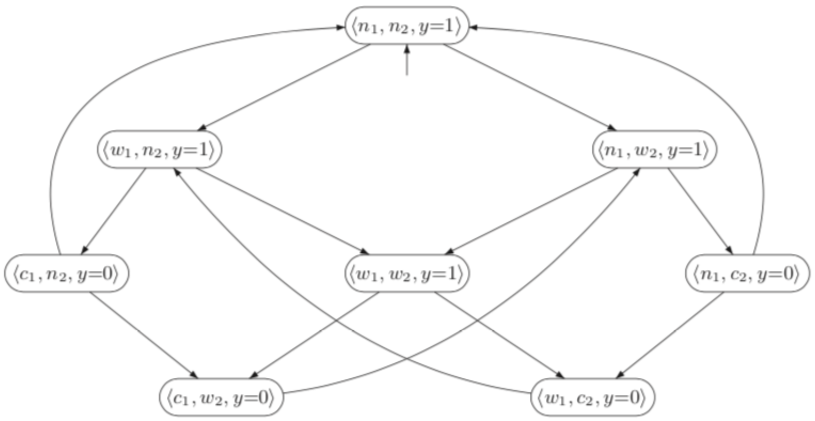


Example 5 (Semaphore based solution to the MEP)**.** Simple example of a semaphore based solution to the MEP problem. The semaphore is actually simulated by a global variable `y` holding the number of permits. The square brackets indicate atomic actions and have the same meaning as the `atomic` blocks in Promela. In particular, if any statement within the atomic sequence blocks, atomicity is lost, and other processes are then allowed to start executing statements. When the blocked statement becomes executable again, the execution of the atomic sequence can be resumed at any time, but not necessarily immediately. The labels (such as `n1` and `w1`) will allow us to refer to specificl instructions in each of the two threads.

```
int y=1;
```

```
Thread.start { //P       Thread.start { //Q
  while (true) {           while (true) {
      n1: // non-CS            n2: // non-CS
      w1: [y>0;                w2: [y>0;
           y--;]                    y--;]
      c1: [y++;]               c3: [y++;]
  }                        }
}                        }
```

The TS for the system resulting from the concurrent execution of both threads is depicted below, where all interleavings have been taken into account.


$\langle n_1, n_2, y{=}1\rangle$
$\langle w_1, n_2, y{=}1\rangle$
$\langle n_1, w_2, y{=}1\rangle$
$\langle c_1, n_2, y{=}0\rangle$
$\langle w_1, w_2, y{=}1\rangle$
$\langle n_1, c_2, y{=}0\rangle$
$\langle c_1, w_2, y{=}0\rangle$
$\langle w_1, c_2, y{=}0\rangle$

Note that the set of actions is $Act = \{\tau\}$ and the label τ has been omitted from the transitions in the picture.

As for AP, we might consider $AP = \{wait1, crit1, wait2, crit2\}$ and the labeling where if a thread is at wi then $waiti$ is true and if it is at ci then $criti$ is true. For example, $L(\langle w1, c2, y = 0\rangle) = \{wait1, crit2\}$ indicating that "it is true that P is waiting to enter the CS" and "it is true that Q is inside the CS".

Definition 6 (Predecessors, Successors)**.** Let $\mathcal{T} = (S, Act, \rightarrow, I, AP, L)$ be a TS. For $s \in S$ and $\alpha \in Act$, we define:

$$\begin{array}{lcl} \mathsf{Post}(s,\alpha) & := & \{s' \in S \mid s \xrightarrow{\alpha} s'\} \\ \mathsf{Post}(s) & := & \bigcup_{\alpha \in Act} \mathsf{Post}(s,\alpha) \\ \mathsf{Pre}(s,\alpha) & := & \{s' \in S \mid s' \xrightarrow{\alpha} s\} \\ \mathsf{Pre}(s) & := & \bigcup_{\alpha \in Act} \mathsf{Pre}(s,\alpha) \end{array}$$

These notions are extended to sets of states $C \subseteq S$, pointwise.

Definition 7 (Terminal State)**.** Let $\mathcal{T} = (S, Act, \rightarrow, I, AP, L)$ be a TS. A state $s \in S$ is *terminal* iff $\mathsf{Post}(s) = \emptyset$.

Definition 8 (Path fragment)**.** Let $\mathcal{T} = (S, Act, \rightarrow, I, AP, L)$ be a TS. A *finite path fragment* $\hat{\pi}$ of $\mathcal{T}$ is a sequence of states $s_0 s_1 \dots s_n$ such that $s_i \in \mathsf{Post}(s_{i-1})$ for all $0 < i \leq n$.

An *infinite path fragment* π of $\mathcal{T}$ is a sequence of states $s_0 s_1 \dots$ such that $s_i \in \mathsf{Post}(s_{i-1})$ for all $0 < i$.

Notation 9. Let π be the path fragment $s_0 s_1 \dots$. We define:

$$\begin{array}{lcl} \mathsf{first}(\pi) & := & s_0 \\ \pi[j] & := & s_j \\ \pi[..j] & := & s_0 s_1 \dots s_j \\ \pi[j..] & := & s_j s_{j+1} \dots \end{array}$$

Remark 10. Corresponds to Def.3.4 of [BK08]. Paths correspond to pieces of "executions" of a system where the actions are ignored and only the states are taken into account. This is called the *state-based* approach to analyzing a computer system.

Definition 11 (Maximal and initial path fragment)**.** A *maximal path fragment* is either a finite path that ends in a terminal state, or an infinite path fragment. A path fragment $s_0 s_1 \dots$ is *initial* if $s_0 \in I$.

Definition 12 (Path)**.** A *path* of a transition system $\mathcal{T}$ is an initial, maximal path fragment.

A path is an execution in the system: it starts at a start state and runs to completion, where completion means either reaching a terminal state or else running infinitely.

Example 13 (Beverage Vending Machine (cont.))**.** Consider Example 2.

$$\begin{array}{lcl} \hat{\pi} & = & pay\ select\ soda\ pay\ select\ soda \\ \pi_1 & = & pay\ select\ soda\ pay\ select\ soda \dots \\ \pi_2 & = & select\ soda\ pay\ select\ soda \dots \end{array}$$

Example 14 (Semaphore based solution to the MEP (cont.))**.** Consider Example 5.

Let $\mathsf{paths}(s)$ denote the set of maximal path fragments π with $\mathsf{first}(\pi) = s$.

Definition 15 (Trace)**.** Let $\mathcal{T} = (S, Act, \to, I, AP, L)$ be a transition system without terminal states. The trace of the infinite path fragment $\pi = s_0 s_1 ...$ is defined as

$$\mathsf{trace}(\pi) := L(s_0)L(s_1)...$$

The trace of the finite path fragment $\hat{\pi} = s_0 s_1 ... s_n$ is defined as

$$\mathsf{trace}(\pi) := L(s_0)L(s_1)...L(s_n)$$

The set of traces of a set of path fragments Π is

$$\mathsf{trace}(\Pi) := \bigcup_{\pi \in \Pi} \mathsf{trace}(\pi)$$

The trace of a state s is

$$\mathsf{traces}(s) := \mathsf{trace}(\mathsf{paths}(s))$$

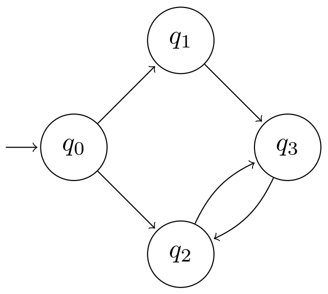
The set of traces of $\mathcal{T}$ is

$$\mathsf{traces}(\mathcal{T}) := \bigcup_{s \in I} \mathsf{traces}(s)$$

Note that the trace of a path fragment is just a (possibly infinite) word over the alphabet 2^{AP}.

Example 16 (Semaphore based solution to the MEP (cont.))**.** Consider Example 14.

Exercise 17. Consider the following transition system:

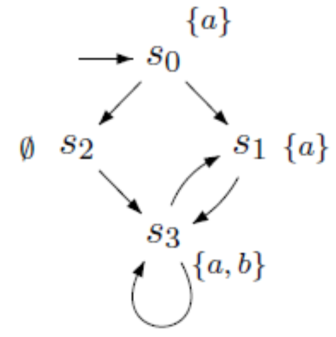

$$\begin{tikzcd}
 & q_1 \arrow[dr] & \\
\arrow[r] & q_0 \arrow[ur] \arrow[dr] & & q_3 \arrow[dl, bend left] \\
 & & q_2 \arrow[ur, bend left] &
\end{tikzcd}$$

where S, I, and $\to$ are described above, $AP = \{a, b\}$, $Act = \{\tau\}$ (not drawn), $L(q_0) = \{a\}$, $L(q_1) = \emptyset$, $L(q_2) = \{a\}$ and $L(q_3) = \{a, b\}$. Give an example of

1. A finite path fragment
2. An infinite path fragment
3. An infinite path
4. An infinite path fragment that is not a path
5. Are there any finite paths? Justify your answer

6. A trace for a path

Exercise 18. Provide an expression denoting all the traces on the set of $AP = \{a, b\}$ of the following transition system:



2 Linear Time Properties

A Linear Time Property specifies the traces that a transition system should exhibit in order to be considered well-behaved.

Below we assume a given, fixed set AP of atomic propositions.

Definition 19 (LT Property)**.** A linear-time property (LT property) over the set of atomic propositions AP is a subset of $(2^{AP})^\omega$.

As mentioned, such a property denotes the set of admissible behaviors.

Definition 20 (Satisfaction for LT Properties)**.** Let $\mathcal{T}$ be a TS without terminal states and P be an LT property over AP. The relations "$\mathcal{T}$ satisfies P" (written $\mathcal{T} \models P$) and "s satisfies P" (written $s \models P$), are defined as follows:

$$\begin{array}{lll} \mathcal{T} \models P & \text{iff} & \mathsf{traces}(\mathcal{T}) \subseteq P \\ s \models P & \text{iff} & \mathsf{traces}(s) \subseteq P \end{array}$$

Note that we are only interested in TS without terminal states.

Example 21 (Traffic lights)**.** Consider the following set of atomic propositions $AP := \{red_1, green_1, red_2, green_2\}$. It aims to provide the necessary propositions to be able to state properties about a pair of traffic lights located at a crossroads. We'll look at some example of such properties.

- *"The first traffic light is infinitely often green".*

  This LT property corresponds to the set of infinite words of the form $A_0A_1A_2...$ over 2^{AP}, such that $green_1 \in A_i$ holds for infinitely many i. For example, P contains the infinite words

  $$\begin{array}{l} \{red1, green2\}\ \{green1, red2\}\ \{red1, green2\}\ \{green1, red2\}..., \\ \emptyset\ \{green1\}\ \emptyset\ \{green1\}\ \emptyset\ \{green1\}\ \emptyset\ \{green1\}\ \emptyset... \\ \{red1, green1\}\{red1, green1\}\{red1, green1\}\{red1, green1\}...\ \text{and} \\ \{green1, green2\}\{green1, green2\}\{green1, green2\}\{green1, green2\}... \end{array}$$

  An alternative formulation is:

$$P = \{A_0A_1A_2... \in (2^{AP})^\omega \mid \forall i \geq 0. \exists j > i. green_1 \in A_j\}$$

The infinite word $\{red1, green1\}\{red1, green1\}\emptyset\emptyset\emptyset\emptyset...$ is not in P as it contains only finitely many occurrences of $green_1$.

- *"The traffic lights are never both green simultaneously".*

  This property is formalized by the set of infinite words of the form $A_0A_1A_2...$ such that either $green_1 \notin A_i$ or $green_2 \notin A_i$, for all $i > 0$. For example, the following infinite words are in P:

  $$\begin{array}{l} \{red1, green2\}\{green1, red2\}\{red1, green2\}\{green1, red2\}..., \\ \emptyset\ \{green1\}\ \emptyset\ \{green1\}\ \emptyset\ \{green1\}\ \emptyset\ \{green1\}\ \emptyset ... and \\ \{red1, green1\}\{red1, green1\}\{red1, green1\}\{red1, green1\}..., \end{array}$$

  whereas the infinite word $\{red1, green2\}\ \{green1, green2\}, ...$ is not in P.

  Alternative formulations are:

  $$P = \{A_0A_1A_2... \in (2^{AP})^\omega \mid \forall i \geq 0. \{green_1, green_2\} \not\subseteq A_i\}$$
  $$P = \{A_0A_1A_2... \in (2^{AP})^\omega \mid \forall i \geq 0. green_1 \notin A_i \vee green_2 \notin A_i\}$$
  $$P = \{A_0A_1A_2... \in (2^{AP})^\omega \mid \forall i \geq 0. \neg(green_1 \in A_i \wedge green_2 \in A_i)\}$$

Example 22 (Mutual Exclusion Property)**.** The formalization of the mutual exclusion property is given by the LT property

$$P_{mutex} := \{A_0A_1A_2 \ldots \in (2^{AP})^\omega \mid \forall i \geq 0. \{crit1, crit2\} \not\subseteq A_i\}$$

For example, the infinite words

$$\begin{array}{ll} \{crit1\}\{crit2\}\{crit1\}\{crit2\}\{crit1\}\{crit2\}..., & and \\ \{crit1\}\{crit1\}\{crit1\}\{crit1\}\{crit1\}\{crit1\}..., & and \\ \emptyset\ \emptyset\ \emptyset\ \emptyset\ \emptyset\ \emptyset\ \emptyset ... & \end{array}$$

are all contained in P_{mutex} . However, this does not apply to words of the form $\{crit1\}\ \emptyset\ \{crit1, crit2\}...$

Example 23 (Finite wait)**.**

$$P_{finwait} := \{A_0A_1A_2... \in (2^{AP})^\omega \mid \forall j. waiti \in A_j \implies \exists k > j. criti \in A_k, \forall i \in \{1, 2\}\}$$

Here, we assumed the set of propositions to be: $AP = \{wait1, crit1, wait2, crit2\}$. Property $P_{finwait}$ expresses that each of the two processes enters its critical section eventually if they are waiting. This property is stronger than freedom from starvation (next example). A process should be allowed to "back-off" and decide not to retry later; only after waiting often should it enter often its critical section.

Example 24 (Freedom from starvation)**.**

$$P_{nostarve} := \{A_0A_1A_2... \in (2^{AP})^\omega \mid (\forall k \geq 0. \exists j \geq k. waiti \in A_j) \implies (\forall k \geq 0. \exists j \geq k. criti \in A_j), \forall i \in \{1, 2\}\}$$

In abbreviated form we write:

$$\overset{\infty}{\exists} j.waiti \in A_j \implies \overset{\infty}{\exists} j.criti \in A_j, \forall i \in \{1,2\}$$

where $\overset{\infty}{\exists} j$ stands for "there are infinitely many $j \in \mathbb{N}$". Likewise, $\overset{\infty}{\forall} j.F$ is defined as $\exists i \geq 0.\forall j \geq i.F$ and stands for "for almost all $j \in \mathbb{N}$". Property $P_{nostarve}$ expresses that each of the two processes enters its critical section infinitely often if they are waiting infinitely often. This natural requirement is, however, not satisfied for the semaphore-based solution, since

$$\emptyset(\{wait2\}\{wait1, wait2\}\{crit1, wait2\})^{\omega}$$

is a possible trace of the transition system but does not belong to $P_{nostarve}$. This trace represents an execution in which only the first process enters its critical section infinitely often. In fact, the second process waits infinitely long to enter its critical section.

Exercise 25. ($\Diamond$) Consider the set AP of atomic propositions defined by $AP = \{x = 0, x > 1\}$ and consider a nonterminating sequential computer program P that manipulates the variable x. Formulate the following informally stated properties as LT properties:

1. false
2. initially x is equal to zero
3. initially x differs from zero
4. initially x is equal to zero, but at some point x exceeds one
5. x exceeds one only finitely many times
6. x exceeds one infinitely often
7. the value of x alternates between zero and one (you may assume that x ranges over positive integers only)
8. true

For each of the above, indicate whether they are safety or liveness properties.

Exercise 26. ($\Diamond$) Let $AP = \{a, b, c\}$ be the set of atomic propositions. Consider the following linear time properties informally stated:

1. initially a holds and b does not hold
2. c holds only finitely many times
3. from some point on the truth value of a alternates between true and false
4. whenever c holds, then a holds sometime afterwards
5. b holds infinitely many times and whenever b holds then c holds afterwards

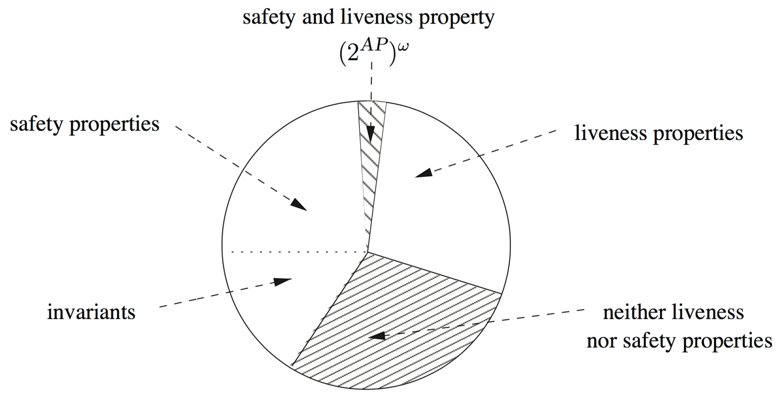


Figure 1: Classification of LT properties

6. whenever c holds, then a or b must also hold
7. a holds only finitely many times and c holds infinitely many times
8. whenever a holds then b and c holds after one step
9. never a and b hold at the same time and eventually c holds
10. at any point the number of times a held in the past is always greater than or equal to the number of times b held in the past.

For each property above, (a) formally write it as a set of infinite traces on 2^{AP} and (b) determine whether it is a safety, liveness or mixed (safety and liveness) linear time property. Justify your answers!

2.1 Safety Properties

Properties such that any infinite run that violates the requirement has a *finite* bad prefix. Before introducing the precise definition of safety properties, we present a particular subset of safety properties called *invariants*.

Definition 27 (Invariant). An LT property P_{inv} over AP is an *invariant* if there is a propositional logic formula Φ over AP s.t.:

$$P_{inv} = \{A_0 A_1 A_2 \ldots \in (2^{AP})^\omega \mid \forall j \geq 0. A_j \models \Phi\}$$

The formula Φ is called an *invariant condition* of P_{inv}.

An example of an invariant is P_{mutex} modeling mutual exclusion, where:

$$\Phi = \neg crit1 \vee \neg crit2$$

Checking that a transition system satisfies an invariant amounts to checking the validity of Φ for every reachable state from some initial state. Assuming the TS is finite, this can be achieved by a standard DFS or BFS traversal.

Not all safety properties are invariant. Validity of some safety properties cannot be determined by inspecting a state in isolation. For example, the property that a yellow light should precede a red light is a safety property but not an invariant property.

Informally, a safety property is such that "nothing bad should happen". Formally, a safety property P is defined as an LT property over AP s.t. any infinite work σ where P does not hold contains a bad prefix. The latter means a finite prefix $\hat{\sigma}$ where the bad thing has happened, and thus no infinite word that starts with $\hat{\sigma}$ fulfills P.

Definition 28 (Safety Property)**.** An LT property P_{safe} over AP is called a *safety property* if for all words $\sigma \in (2^{AP})^\omega \setminus P_{safe}$ there exists a finite prefix $\hat{\sigma}$ of σ such that

$$P_{safe} \cap \{\sigma' \in (2^{AP})^\omega \mid \hat{\sigma} \text{ is a finite prefix of } \sigma'\} = \emptyset$$

Any such finite word $\hat{\sigma}$ is called a bad prefix for P_{safe}. A shortest such bad prefix is called a *minimal bad prefix* for P_{safe}.

Note that all invariants are safety properties.

Example 29 (Traffic lights)**.** $AP = \{red, yellow, green\}$.

- *"Always at least one of the lights is on".* This property may be expressed as:

  $$\{A_0A_1A_2... \in (2^{AP})^\omega \mid \forall i \geq 0. A_i \neq \emptyset\}$$

  The bad prefixes are finite words that contain $\emptyset$.

- *"It is never the case that two lights are switched on at the same time".*

  $$\{A_0A_1A_2... \in (2^{AP})^\omega \mid \forall i \geq 0. |A_i| \leq 1\}$$

  The bad prefixes are finite words that contain sets such as $\{red, green\}$ or $\{green, yellow\}$.

Example 30 (Traffic lights)**.** $AP = \{red, yellow\}$.

- *"A red phase must be preceded immediately by a yellow phase"*

  $$\{A_0A_1A_2... \in (2^{AP})^\omega \mid \forall i > 0. red \in A_i \implies yellow \in A_{i-1}\}$$

  Examples of bad prefixes are $\emptyset\emptyset\{red\}$ and $\emptyset\{red\}$. The prefix $\emptyset\{red\}\{red\}$ is also a bad prefix, but not a minimal one.

- A non-example: *"A red light eventually turns on"*:

  $$P = \{A_0A_1A_2... \in (2^{AP})^\omega \mid \exists i \geq 0. red \in A_i\}$$

  We'll show that it is not a safety property. Let σ be the word $\emptyset^\omega$. Notice that $\sigma \in (2^{AP})^\omega \setminus P$. Then any prefix of σ, say $\hat{\sigma}$, will be of the form $\hat{\sigma} = \emptyset^n$ for some $n \geq 0$. But we can always "fix" $\hat{\sigma}$ by"extending" it so as to obtain a word in P. In other words:

  $$P \cap \{\sigma' \in (2^{AP})^\omega \mid \hat{\sigma} \text{ is a finite prefix of } \sigma'\} \neq \emptyset$$

Example 31 (Vending Machine). $AP = \{pay, drink\}$

"The number of inserted coins is always at least the number of dispensed drinks"

$$P = \{A_0A_1A_2... \in (2^{AP})^\omega \mid \forall i \geq 0.|\{0 \leq j \leq i \mid pay \in A_j\}| \geq |\{0 \leq j \leq i \mid drink \in \ A_j\}|\}$$

Examples of bad prefixes are $\emptyset\{pay\}\{drink\}\{drink\}$ or $\emptyset\{pay\}\{drink\}\emptyset\{pay\}\{drink\}\{drink\}$.

Remark 32. Show that a LT property P is a safety property iff $\mathsf{closure}(P) = P$, where:

$$\mathsf{closure}(P) := \{\sigma \in (2^{AP})^\omega \mid \mathsf{pref}(\sigma) \subseteq \mathsf{pref}(P)\}$$

2.2 Liveness Properties

Informally, a "liveness" or "progress" property states that "something good"will happen in the future. Whereas safety properties are violated in finite time, *i.e.* by finite system runs, liveness properties are violated in infinite time *i.e.* by infinite system runs.

The set of finite traces are therefore of no use at all to decide whether a liveness property holds or not. Intuitively, any such finite trace could always be extended such that the resulting infinite trace satisfies the liveness property.

Definition 33 (Liveness Property). LT property P_{live} over AP is a liveness property whenever $\mathsf{pref}(P_{live}) = (2^{AP})^*$.

Stated differently, P is a liveness property if for all $w \in (2^{AP})^*$ there is a $\sigma \in (2^{AP})^\omega$ s.t. $w\sigma \in P$.

Example 34. $AP = \{wait1, crit1, wait2, crit2\}$

- Eventually

  "each process will eventually enter its critical section"

$$\{A_0A_1A_2... \in (2^{AP})^\omega \mid (\exists i \geq 0.crit1 \in A_i) \wedge (\exists i \geq 0.crit2 \in A_i)\}$$

- Repeated eventually

  "each process will enter its critical section infinitely often"

$$\{A_0A_1A_2... \in (2^{AP})^\omega \mid (\forall i \geq 0.\exists j \geq i.crit1 \in A_i) \wedge (\forall i \geq 0.\exists j \geq i.crit2 \in A_i)\}$$

- Starvation freedom

  "each waiting process will eventually enter its critical section"

$$\{A_0A_1A_2... \in (2^{AP})^\omega \mid [(\forall i \geq 0.\exists j \geq i.wait1 \in A_i) \implies (\forall i \geq 0.\exists j \geq i.crit1 \in A_i)] \wedge [(\forall i \geq 0.\exists j \geq i.wait2 \in A_i) \implies$$

2.3 Safety vs Liveness Properties

Apart from $P = (2^{AP})^\omega$, safety and liveness properties are disjoint:

Lemma 35 ([BK08] L. 3.35)**.** The only LT property over AP that is both a safety and liveness property is $(2^{AP})^\omega$.

Not every LT property is either a safety or a liveness property. An example is, *"The beverage machine provides beer infinitely often after initially providing soda three times in a row".* However, one can prove that every LT property is a combination of a safety and a liveness property.

Theorem 36 ([BK08] Thm. 3.37)**.** For any LT property P over AP there exists a safety property P_{safe} and a liveness property P_{live} (both over AP) s.t.

$$P = P_{safe} \cap P_{live}$$

2.4 Fairness

In general, fairness assumptions are needed in order to prove liveness or other properties stating that the system makes some progress ("something good will eventually happen"). Fairness constraints are used to rule out computations that are considered to be unreasonable for the system under consideration.

3 Automata and Infinite Words

ω-regular properties (4.3.1)

3.1 Nondeterministic Buüchi Automata

Definition 37 (NBAs (4.3.2))**.** A non-deterministic Buchi automaton (NBA) $\mathcal{A}$ is a tuple $\mathcal{A} = (Q, \Sigma, \delta, Q_0, F)$ where

- Q is a finite set of states,
- Σ is an alphabet,
- $\delta : Q \times \Sigma \rightarrow 2^Q$ is a transition function,
- Q_0 is a set of initial states, and
- $F \subseteq Q$ is a set of final states.

A run for $\sigma = A_0A_1A_2 \ldots \in \Sigma^\omega$ denotes an infinite sequence $q_0q_1q_2 \ldots$ of states in $\mathcal{A}$ s.t. $q_0 \in Q_0$ and $q_i \overset{\rightarrow}{A_i} q_{i+1}$. The language accepted by $\mathcal{A}$

$$\mathcal{L}_\omega(\mathcal{A}) = \{\}$$

Example 38 (4.28)**.**

Remark 39. Deterministic Büchi Automata do not have the same expressive power as FA. Deterministic Buüchi Automaton (DBA, Definition 4.48) are defined as follows. Let $\mathcal{A} = (Q, \Sigma, \delta, Q_0, F)$ be an NBA. $\mathcal{A}$ is called deterministic, if $|Q_0| \leq 1$ and $|\delta(q, A)| \leq 1$ for all $q \in Q$ and $A \in \Sigma$.

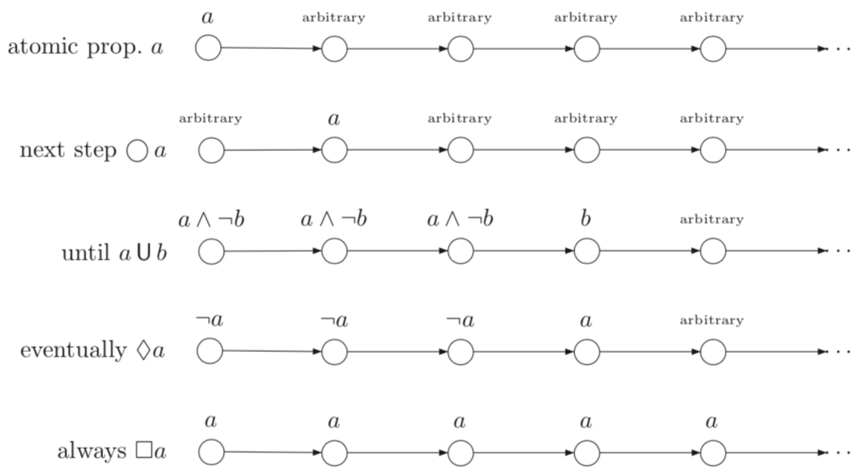


Figure 2: Intuitive semantics of LTL connectives

3.2 Model-checking ω-regular properties (4.4)

4 Linear Temporal Logic

4.1 Syntax

Definition 40 (Syntax of LTL)**.** LTL formulae over the set AP of atomic proposition are formed according to the following grammar:

$$\phi \quad ::= \quad \mathit{true} \,|\, a \,|\, \neg\phi \,|\, \phi_1 \wedge \phi_2 \,|\, \bigcirc \phi \,|\, \phi_1 \cup \phi_2$$

where $a \in AP$. Here $\neg$ and $\wedge$ are the standard connectives for negation and conjunction. The connectives $\bigcirc$ and $\cup$ are called *temporal connectives*. We read $\bigcirc$ as “next” and $\cup$ as “until”. A formula such as $\bigcirc\phi$ states that ϕ holds in the next state and a formula $\phi_1 \cup \phi_2$ states that ϕ_2 will hold sometime in the future and, until then, ϕ_1 holds. Further intuition is provided in Fig. 2.

Two derived (meaning they can be defined in terms of the others) connectives that are used often are:

$$\begin{array}{rcll} \Diamond\phi & := & \mathit{true} \cup \phi & \text{”eventually”} \\ \Box\phi & := & \neg\Diamond\neg\phi & \text{”always”} \end{array}$$

For example, $\Box\neg(\mathit{crit}1 \wedge \mathit{crit}2)$ states mutual exclusion.

4.2 Semantics

We next define when a trace satisfies an LTL-formula. Before doing so, however, we recall the semantics of Propositional Logic.

Definition 41 (Satisfiability for Propositional Logic)**.** Let $A \subseteq AP$ be an assignment of truth values to atomic propositions[2]. Let ϕ a propositional formula over the same set AP of atomic propositions. We say A satisfies ϕ if $A \models \phi$, where the binary relation $\models$ is defined by induction on ϕ as follows:

$$\begin{array}{lll} A \models \mathit{true} & & \\ A \models a & \text{iff} & a \in A \\ A \models \neg\phi & \text{iff} & A \not\models \phi \\ A \models \phi_1 \wedge \phi_2 & \text{iff} & A \models \phi_1 \text{ and } A \models \phi_2 \end{array}$$

For example, if $AP = \{\mathit{wait}, \mathit{crit}\}$ and $A = \{\mathit{wait}\}$, then $A \models \mathit{wait} \wedge \neg\mathit{crit}$ but $A \not\models \mathit{wait} \wedge \mathit{crit}$.

Next we consider the case for LTL. We assume given a TS $\mathcal{T} = (S, Act, \to, I, AP, L)$.

Definition 42 (Satisfiability for LTL)**.** Let σ be a trace of the form $A_0A_1A_2\ldots$ and ϕ an LTL formula, both over the same set AP of atomic propositions. We say σ satisfies ϕ if $\sigma \models \phi$, where the binary relation $\models$ is defined by induction on ϕ as follows:

$$\begin{array}{lll} \sigma \models \mathit{true} & & \\ \sigma \models a & \text{iff} & a \in A_0 \\ \sigma \models \neg\phi & \text{iff} & \sigma \not\models \phi \\ \sigma \models \phi_1 \wedge \phi_2 & \text{iff} & \sigma \models \phi_1 \text{ and } \sigma \models \phi_2 \\ \sigma \models \bigcirc\phi & \text{iff} & \sigma[1..] \models \phi \\ \sigma \models \phi_1 \cup \phi_2 & \text{iff} & \exists j \geq 0.\sigma[j..] \models \phi_2 \text{ and } \forall i.0 \leq i < j.\sigma[i..] \models \phi_1 \end{array}$$

This notion may be extended to states. A state s satisfies a formula ϕ, written $s \models \phi$, if $\mathsf{traces}(s) \models \phi$. A set T of traces satisfies ϕ if each trace in T satisfies ϕ.

The next definition defines the notion of the property induced by an LTL formula.

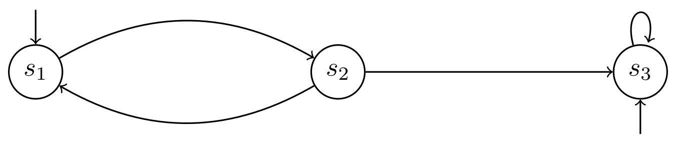
Definition 43 (Property induced by an LTL formula)**.** Let ϕ be an LTL formula. The *LT property induced by* ϕ is:

$$\mathsf{words}(\phi) := \{\sigma \in (2^{AP})^\omega \mid \sigma \models \phi\}$$

Definition 44 (Satisfiability for TSs)**.** Let $\mathcal{T} = (S, Act, \to, I, AP, L)$ be a TS without terminal states and let ϕ be a formula over AP.

- Let π be a path fragment: $\pi \models \phi$ iff $\mathsf{trace}(\pi) \models \phi$
- Let $s \in S$: $s \models \phi$ iff $(\forall \pi \in \mathsf{paths}(s).\pi \models \phi)$
- For the TS $\mathcal{T}$: $\mathcal{T} \models \phi$ iff $\mathsf{traces}(\mathcal{T}) \models \mathsf{words}(\phi)$ (see Def. 15 for the definition of $\mathsf{traces}(\mathcal{T})$).

Example 45. Consider the following transition system $\mathcal{T}$:



[2]Those propositions in A are considered to be "true" and those that are not are considered to be "false".

where $L(s_1) = \{a, b\}$, $L(s_2) = \{a, b\}$ and $L(s_3) = \{a\}$. Then $s_1 \models \bigcirc(a \wedge b)$ but $s_2 \not\models \bigcirc(a \wedge b)$ and $s_3 \not\models \bigcirc(a \wedge b)$. In particular, since $s_3 \not\models \bigcirc(a \wedge b)$ and s_3 is a start state, then $\mathcal{T} \not\models \bigcirc(a \wedge b)$.

Example 46 (Traffic Lights)**.** How are these properties, from Example 21, expressed as LTL formulas? Recall from above that $AP := \{red_1, green_1, red_2, green_2\}$.

- *"The first traffic light is infinitely often green".*

$$\Box\Diamond green_1$$

- *"The traffic lights are never both green simultaneously".*

$$\Box(\neg green_1 \vee \neg green_2)$$

Example 47 (Semaphore-based solution to the MEP)**.** Recall that $AP = \{wait1, crit1, wait2, crit2\}$.

1. $\Box(crit_1 \vee crit_2)$. "Its always the case that one of the two threads is in the critical section"
2. $\Box\neg(crit_1 \wedge crit_2)$. "Mutual exclusion".
3. $\Box(wait1 \implies \Diamond crit1) \wedge \Box(wait2 \implies \Diamond crit2)$. Finite wait.
4. $\Box(y = 0 \supset (crit_1 \vee crit_2))$. "If there are no permits available, then one of the two threads must be in the CS".

There are two combinations of temporal connectives that appear often:

$$\begin{array}{ll} \Box\Diamond\phi & \text{"Infinitely often } \phi\text{"} \\ \Diamond\Box\phi & \text{"Eventually forever } \phi\text{"} \end{array}$$

Satisfiability for these is defined as follows:

$$\begin{array}{lll} \sigma \models \Box\Diamond\phi & \text{iff} & \forall i \geq 0. \exists j \geq i. \sigma[j..] \models \phi \\ \sigma \models \Diamond\Box\phi & \text{iff} & \exists i \geq 0. \forall j \geq i. \sigma[j..] \models \phi \end{array}$$

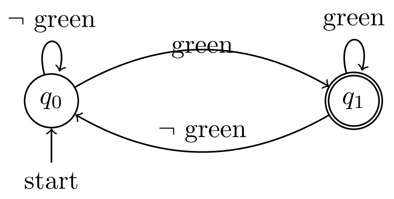
Example 48 (Semaphore-based solution to the MEP)**.** Here are two examples of formulae in which these combinations of connectives occur:

1. $\Box\Diamond crit1 \wedge \Box\Diamond crit2$ states that each process is infinitely often in its critical section. This is a liveness property.
2. $(\Box\Diamond wait1 \implies \Box\Diamond crit1) \wedge (\Box\Diamond wait2 \implies \Box\Diamond crit2)$ states freedom from starvation, also a liveness property.

4.3 LTL and Büchi Automata

Theorem 49. For any LTL formula ϕ over AP there exists a NBA $\mathcal{A}_\phi$ such that $\mathsf{words}(\phi) = \mathcal{L}_\omega(\mathcal{A}_\phi)$.

Example 50. $AP = \{green, red\}$. Let $\phi = \Box\Diamond green$. The NBA $\mathcal{A}_\phi$ is:



Example 51. $AP = \{a, b\}$. Let $\phi = \Box(a \implies \Diamond b)$.

5 Automata-based LTL model-checking (5.2)

Given a finite TS without terminal states $\mathcal{T}$ and an LTL formula ϕ, we are interested in determining whether: $\mathcal{T} \models \phi$. Note the following:

$$\begin{aligned} & \mathcal{T} \models \phi \\ \iff \quad & \mathsf{traces}(\mathcal{T}) \subseteq \mathsf{words}(\phi) \\ \iff \quad & \mathsf{traces}(\mathcal{T}) \cap ((2^{AP})^\omega \setminus \mathsf{words}(\phi)) = \emptyset \\ \iff \quad & \mathsf{traces}(\mathcal{T}) \cap \mathsf{words}(\neg\phi) = \emptyset \end{aligned}$$

6 CTL

7 Solutions to Selected Exercises

Answer to exercise 25

1. false: $P := \emptyset$
2. initially x is equal to zero: $P = \{A_0, A_1, A2... \in (2^{AP})^\omega \mid x = 0 \in A_0\}$
3. initially x differs from zero: $P = \{A_0, A_1, A2... \in (2^{AP})^\omega \mid x = 0 \notin A_0\}$
4. initially x is equal to zero, but at some point x exceeds one: $P = \{A_0, A_1, A2... \in (2^{AP})^\omega \mid x = 0 \in A_0 \wedge \exists i > 0.(x > 0) \in A_i\}$
5. x exceeds one only finitely many times: $P = \{A_0, A_1, A_2... \in (2^{AP})^\omega | \exists i \geq 0.\forall j \geq i.(x > 1) \notin A_j\}$
6. x exceeds one infinitely often: $P = \{A_0, A_1, A_2... \in (2^{AP})^\omega \mid \forall i \geq 0.\exists j \geq i.(x > 1) \in A_j\}$
7. The value of x alternates between zero and one:
$$\begin{array}{l} P = \{A_0, A_1, A_2... \in (2^{AP})^\omega \mid [\forall i.((x = 0) \in A_i \to A_{i+1} = \emptyset)] \\ \wedge[\forall i.(A_i = \emptyset \to (x = 0) \in A_{i+1})] \\ \wedge[\forall i.A_i \subseteq \{x = 0\}\}] \end{array}$$
8. true: $P = (2^{AP})^\omega$

Answer to exercise ??

1. Three solutions:
$P = \{A_0A_1... \in (2^{AP})^\omega | (\forall i \geq 0.\{a, b\} \nsubseteq A_i)\}$
$P = \{A_0A_1... \in (2^{AP})^\omega | (\forall i \geq 0.(a \in A_i \implies b \notin A_i) \wedge (b \in A_i \implies a \notin A_i)\}$
$P = \{A_0A_1... \in (2^{AP})^\omega | (\forall i \geq 0.(a \in A_i \wedge b \notin A_i) \vee (a \notin A_i \wedge b \in A_i) \vee (a \notin A_i \wedge b \notin A_i)\}$
2. The word $\emptyset^\omega$ is not in P (and should be)

Answer to exercise 26

1. initially a holds and b does not hold
$P = \{A_0A_1... \in (2^{AP})^\omega \mid a \in A_0 \wedge b \notin A_0\}$
This property is a SAFETY PROPERTY. A bad prefix can be any word in $(2^{AP})^\star$ starting with $\{\}$ or $\{b\}$ or $\{c\}$ or $\{a, b\}$ or $\{b, c\}$ or $\{a, b, c\}$.
2. c holds only finitely many times
$P = \{A_0A_1... \in (2^{AP})^\omega | \exists i \geq 0.\forall j \geq i.c \notin A_i\}$
This is a LIVENESS PROPERTY because no prefix can be classified as bad because the information on the occurrences of "c" in the tail of the word is missing.
3. from some point on the truth value of a alternates between true and false
$P = \{A_0A_1... \in (2^{AP})^\omega | \exists i \geq 0.\forall j \geq i.a \in A_j \leftrightarrow a \notin A_{j+i}\}$
LIVENESS: no prefix can be classified as bad without the information on the tail of the word.

4. whenever c holds, then also a or b must hold

   $P = \{A_0A_1... \in (2^{AP})^\omega | \forall i \geq 0.(c \in A_i \rightarrow a \in A_j \vee b \in A_j)\}$

   SAFETY: as above

5. whenever c holds, then a or be must hold sometime afterwards

   $P = \{A_0A_1... \in (2^{AP})^\omega | \forall i \geq 0.(c \in A_i \rightarrow \exists j \geq i.a \in A_j \vee b \in A_j)\}$

   LIVENESS: as above.

6. b holds infinitely many times and whenever b holds then c holds afterwards

   $P = \{A_0A_1... \in (2^{AP})^\omega | (\forall i \geq 0.\exists j \geq i.b \in A_i) \wedge (\forall i \geq 0.(b \in A_i \rightarrow \exists j \geq i : c \in A_j))\}$

   or,

   $P = \{A_0A_1... \in (2^{AP})^\omega | (\forall i.\exists j \geq i.b \in A_j) \wedge (\forall i \geq 0.(b \in A_i \rightarrow \exists j \geq i : c \in A_j))\}$

   LIVENESS.

7. whenever c holds then also a or b holds

   $P = \{A_0A_1... \in (2^{AP})^\omega | \forall i \geq 0.(c \in A_i \rightarrow (a \in A_i \vee b \in A_i))\}$

   SAFETY: a bad prefix is, for instance, $\{c\}\{\}\{\}\{\}$

8. a holds only finitely many times and c holds infinitely many times

   $P = \{A_0A_1... \in (2^{AP})^\omega | (\exists i \geq 0.\forall j \geq i.a \notin A_j) \wedge (\forall i.\exists j \geq i.c \in A_j)\}$

   LIVENESS

9. whenever a holds then b and c holds after one step

   $P = \{A_0A_1... \in (2^{AP})^\omega | \forall i \geq 0.a \in A_i \rightarrow (b \in A_{i+1} \wedge c \in A_{i+1})\}$

   SAFETY: a bad prefix is for instance $\{a\}\{a\}\{a\}$

10. never a and b hold at the same time and eventually c holds

    $P = \{A_0A_1... \in (2^{AP})^\omega | (\forall i \geq 0.\{a, b\} \not\subseteq A_i) \wedge (\exists i \geq 0.c \in A_i)\}$

    MIXED: a bad prefix for the first part is $\{a, b\}\{\}\{\}...$ The part on "eventually" c cannot have a bad prefix, so it is liveness property.

11. at any point the number of times a held in the past is always greater than or equal to the number of times b held in the past.

    $P = \{A_0A_1... \in (2^{AP})^\omega \mid \forall i \geq 0.|\{0 \leq j \ \leq i : a \in A_j\}| \geq |\{0 \leq j \leq i : b \in A_j\}|\}$

    where $|\{...\}|$ is set cardinality.

    SAFETY: a bad prefix for example $\{b\}\{\}\{\}$

Answer to exercise 26

1. initially a holds and b does not hold

   $P = \{A_0A_1... \in (2^{AP})^\omega \mid a \in A_0 \wedge b \notin A_0\}$

   This property is a SAFETY PROPERTY. A bad prefix can be any word in $(2^{AP})^\star$ starting with $\{\}$ or $\{b\}$ or $\{c\}$ or $\{a, b\}$ or $\{b, c\}$ or $\{a, b, c\}$.

2. c holds only finitely many times

   $P = \{A_0 A_1 ... \in (2^{AP})^\omega | \exists i \geq 0. \forall j \geq i. c \notin A_j\}$

   This is a LIVENESS PROPERTY because no prefix can be classified as bad because the information on the occurrences of "c" in the tail of the word is missing.

3. from some point on the truth value of a alternates between true and false

   $P = \{A_0 A_1 ... \in (2^{AP})^\omega | \exists i \geq 0. \forall j \geq i. a \in A_j \leftrightarrow a \notin A_{j+1}\}$

   LIVENESS: no prefix can be classified as bad without the information on the tail of the word.

4. whenever c holds, then a holds sometime afterwards

   $P = \{A_0 A_1 ... \in (2^{AP})^\omega | \forall i \geq 0. (c \in A_i \rightarrow \exists j \geq i. a \in A_j)\}$

   SAFETY: as above

5. b holds infinitely many times and whenever b holds then c holds afterwards

   $P = \{A_0 A_1 ... \in (2^{AP})^\omega | (\forall i \geq 0. \exists j \geq i. b \in A_i) \wedge (\forall i \geq 0. (b \in A_i \rightarrow \exists j \geq i. c \in A_j))\}$

   or,

   $P = \{A_0 A_1 ... \in (2^{AP})^\omega | (\forall i. \exists j \geq i. b \in A_j) \wedge (\forall i \geq 0. (b \in A_i \rightarrow \exists j \geq i. c \in A_j))\}$

   LIVENESS.

6. whenever c holds, then also a or b holds

   $P = \{A_0 A_1 ... \in (2^{AP})^\omega | \forall i \geq 0. (c \in A_i \rightarrow (a \in A_i \vee b \in A_i))\}$

   SAFETY: a bad prefix is, for instance, $\{c\}\{\}\{\}\{\}$

7. a holds only finitely many times and c holds infinitely many times

   $P = \{A_0 A_1 ... \in (2^{AP})^\omega | (\exists i \geq 0. \forall j \geq i. a \notin A_j) \wedge (\forall i. \exists j \geq i. c \in A_j)\}$

   LIVENESS

8. whenever a holds then b and c holds after one step

   $P = \{A_0 A_1 ... \in (2^{AP})^\omega | \forall i \geq 0. a \in A_i \rightarrow (b \in A_{i+1} \wedge c \in A_{i+1})\}$

   SAFETY: a bad prefix is for instance $\{a\}\{a\}\{a\}$

9. never a and b hold at the same time and eventually c holds

   $P = \{A_0 A_1 ... \in (2^{AP})^\omega | (\forall i \geq 0. \{a, b\} \not\subseteq A_i) \wedge (\exists i \geq 0. c \in A_i)\}$

   MIXED: a bad prefix for the first part is $\{a, b\}\{\}\{\}$ The part on "eventually" c cannot have a bad prefix, so it is liveness property.

10. at any point the number of times a held in the past is always greater than or equal to the number of times b held in the past.

    $P = \{A_0 A_1 ... \in (2^{AP})^\omega \mid \forall i \geq 0. \parallel \{\forall j. 0 \leq j \wedge j \leq i. a \in A_j\} \parallel \geq \parallel \{\forall j. 0 \leq j \wedge j \leq i \rightarrow b \in A_j\} \parallel\}$

    where $|\{...\}|$ is set cardinality.

    SAFETY: a bad prefix for example $\{b\}\{\}\{\}$

References

[BK08] Christel Baier and Joost-Pieter Katoen. *Principles of Model Checking (Representation and Mind Series)*. The MIT Press, 2008.