Model Checking

Juri Kolčák

Friday 9th January, 2026

Automated Verification

Model checking is a general framework for computationally verifying linear-time properties (specified in temporal logic), on a transition system.

Recall that for a Boolean network of dimension n, the transition system is of size 2^n. Automated verification is therefore paramount.

Model checking is typically "the last line of defence", and one would attempt to reduce the transition system beforehand.

- Static analysis;
- Model reduction (trap spaces, merging/removing variables, . . .);
- Decomposition;
- Symbolic representation;
- . . .

Kripke Structure

A **Kripke structure** is a tuple $\mathcal{T} = (S, \to, I, P, \alpha)$ consisting of:

- A transition system $(S, \to)$;
- A set of initial states $I \subseteq S$;
- A set of atomic propositions P;
- A function $\alpha\colon S \to 2^P$, mapping each configuration to a set of atomic propositions.

Kripke Structure

A **Kripke structure** is a tuple $\mathcal{T} = (S, \to, I, P, \alpha)$ consisting of:

- A transition system $(S, \to)$;
- A set of initial states $I \subseteq S$;
- A set of atomic propositions P;
- A function $\alpha \colon S \to 2^P$, mapping each configuration to a set of atomic propositions.

For each trace $\sigma = (\mathbf{x}^0, \mathbf{x}^1, \mathbf{x}^2, \dots)$ of the transition system $(S, \to)$, the corresponding trace of the Kripke structure $\mathcal{T}$ is $\pi = (\alpha(\mathbf{x}^0), \alpha(\mathbf{x}^1), \alpha(\mathbf{x}^2), \dots)$.

A trace σ is **initial**, if $\mathbf{x}^0 \in I$.

$\mathcal{S}(\mathcal{T})$ is the set of all the initial and infinite (maximal) traces of the Kripke structure.

Linear-time Properties

A linear-time (LT) property is a language $L \subseteq (2^P)^\omega$ of infinite words over the alphabet $\Sigma = 2^P$.

L is the language of all the "good" behaviours, that satisfy the desired property.

A transition system (Kripke structure) $\mathcal{T}$ satisfies the LT property L, $\mathcal{T} \models L$, if and only if $\mathcal{S}(\mathcal{T}) \subseteq L$.

Invariant Properties

A property L_{inv} is an **invariant** if there exists a propositional logic formula Φ over the atomic propositions P, the invariant condition, such that:

$$L_{inv} = \{P_0 P_1 P_2 \cdots \in \left(2^P\right)^\omega \mid \forall i \in \mathbb{N}_0, P_i \models \Phi\}$$

Let $\mathcal{R}(\mathcal{T}) = \{\mathbf{x} \in S \mid \exists \mathbf{y} \in I, \mathbf{y} \to^* \mathbf{x}\}$ be the set of all configurations reachable from at least one of the initial configurations.

A transition system $\mathcal{T}$ satisfies the invariant property L_{inv}, $\mathcal{T} \models L_{inv}$, if an only if for all $\mathbf{x} \in \mathcal{R}(\mathcal{T})$, $\mathbf{x} \models \Phi$.

Invariant Properties

A property L_{inv} is an **invariant** if there exists a propositional logic formula Φ over the atomic propositions P, the invariant condition, such that:

$$L_{inv} = \{P_0 P_1 P_2 \cdots \in \left(2^P\right)^\omega \mid \forall i \in \mathbb{N}_0, P_i \models \Phi\}$$

Let $\mathcal{R}(\mathcal{T}) = \{\mathbf{x} \in S \mid \exists \mathbf{y} \in I, \mathbf{y} \rightarrow^* \mathbf{x}\}$ be the set of all configurations reachable from at least one of the initial configurations.

A transition system $\mathcal{T}$ satisfies the invariant property L_{inv}, $\mathcal{T} \models L_{inv}$, if an only if for all $\mathbf{x} \in \mathcal{R}(\mathcal{T})$, $\mathbf{x} \models \Phi$.

EXAMPLE:

$\mathbf{G}(\mathbf{x}_1 \vee \mathbf{x}_2)$

Safety Properties

In simple terms, safety properties say: “something bad never happens”.

An LT property L_{safe} is a **safety** property if for all words $\sigma \in (2^P)^\omega \setminus L_{safe}$, there exists a finite prefix $\hat{\sigma}$ of σ which is not prefix of any word in L_{safe}, $L_{safe} \cap \{\sigma' \in (2^P)^\omega \mid \hat{\sigma} \text{ is a prefix of } \sigma'\} = \emptyset$.

Any such finite word $\hat{\sigma}$ is a **bad prefix** for the safety property L_{safe}, describing a part of behaviour in which the safety property is violated, or that guarantees its future violation.

A bad prefix $\hat{\sigma}$ is **minimal** if there is no proper prefix of $\hat{\sigma}$ which is also bad for L_{safe}.

We use $\texttt{Bad}(L_{safe})$ to denote the set of all bad prefixes of L_{safe}, and $\texttt{MinBad}(L_{safe})$ to denote the set of all minimal bad prefixes of L_{safe}.

Safety Properties – Example

Every invariant property is a safety property.

Consider a Boolean network of dimension at least 2. The following LTL formula defines a safety property:

$$\mathbf{G}((\neg \mathsf{x}_2 \wedge \mathbf{X}(\mathsf{x}_2)) \Rightarrow \mathsf{x}_1)$$

Safety Properties – Example

Every invariant property is a safety property.

Consider a Boolean network of dimension at least 2. The following LTL formula defines a safety property:

$$\mathbf{G}((\neg \mathbf{x}_2 \wedge \mathbf{X}(\mathbf{x}_2)) \Rightarrow \mathbf{x}_1)$$

$$\pi_1 = (\{\mathbf{x}_1\}, \{\mathbf{x}_1, \mathbf{x}_2\}, \{\mathbf{x}_1, \mathbf{x}_2\}, \ldots);$$

Safety Properties – Example

Every invariant property is a safety property.

Consider a Boolean network of dimension at least 2. The following LTL formula defines a safety property:

$$\mathbf{G}((\neg \mathsf{x}_2 \wedge \mathbf{X}(\mathsf{x}_2)) \Rightarrow \mathsf{x}_1)$$

$$\pi_1 = (\{\mathsf{x}_1\}, \{\mathsf{x}_1, \mathsf{x}_2\}, \{\mathsf{x}_1, \mathsf{x}_2\}, \ldots);$$

$$\pi_2 = (\{\mathsf{x}_1\}, \{\mathsf{x}_2\}, \{\mathsf{x}_2\}, \ldots);$$

Safety Properties – Example

Every invariant property is a safety property.

Consider a Boolean network of dimension at least 2. The following LTL formula defines a safety property:

$$\mathbf{G}((\neg \mathsf{x}_2 \wedge \mathbf{X}(\mathsf{x}_2)) \Rightarrow \mathsf{x}_1)$$

$$\pi_1 = (\{\mathsf{x}_1\}, \{\mathsf{x}_1, \mathsf{x}_2\}, \{\mathsf{x}_1, \mathsf{x}_2\}, \dots);$$

$$\pi_2 = (\{\mathsf{x}_1\}, \{\mathsf{x}_2\}, \{\mathsf{x}_2\}, \dots);$$

$$\pi_3 = (\{\mathsf{x}_1\}, \emptyset, \{\mathsf{x}_2\}, \{\mathsf{x}_2\}, \dots);$$

Safety Properties – Example

Every invariant property is a safety property.

Consider a Boolean network of dimension at least 2. The following LTL formula defines a safety property:

$$\mathbf{G}((\neg \mathsf{x}_2 \wedge \mathbf{X}(\mathsf{x}_2)) \Rightarrow \mathsf{x}_1)$$

$$\pi_1 = (\{\mathsf{x}_1\}, \{\mathsf{x}_1, \mathsf{x}_2\}, \{\mathsf{x}_1, \mathsf{x}_2\}, \ldots);$$

$$\pi_2 = (\{\mathsf{x}_1\}, \{\mathsf{x}_2\}, \{\mathsf{x}_2\}, \ldots);$$

$$\pi_3 = (\{\mathsf{x}_1\}, \emptyset, \{\mathsf{x}_2\}, \{\mathsf{x}_2\}, \ldots);$$

Safety Properties – Example

Every invariant property is a safety property.

Consider a Boolean network of dimension at least 2. The following LTL formula defines a safety property:

$$\mathbf{G}((\neg \mathbf{x}_2 \wedge \mathbf{X}(\mathbf{x}_2)) \Rightarrow \mathbf{x}_1)$$

$$\pi_1 = (\{\mathbf{x}_1\}, \{\mathbf{x}_1, \mathbf{x}_2\}, \{\mathbf{x}_1, \mathbf{x}_2\}, \dots);$$

$$\pi_2 = (\{\mathbf{x}_1\}, \{\mathbf{x}_2\}, \{\mathbf{x}_2\}, \dots);$$

$$\pi_3 = (\{\mathbf{x}_1\}, \emptyset, \{\mathbf{x}_2\}, \{\mathbf{x}_2\}, \dots);$$

$$\pi_4 = (\{\mathbf{x}_1\}, \{\mathbf{x}_2\}, \emptyset, \{\mathbf{x}_1, \mathbf{x}_2\}, \{\mathbf{x}_1, \mathbf{x}_2\}, \dots);$$

Safety Properties – Example

Every invariant property is a safety property.

Consider a Boolean network of dimension at least 2. The following LTL formula defines a safety property:

$$\mathbf{G}((\neg \mathsf{x}_2 \wedge \mathbf{X}(\mathsf{x}_2)) \Rightarrow \mathsf{x}_1)$$

$$\pi_1 = (\{\mathsf{x}_1\}, \{\mathsf{x}_1, \mathsf{x}_2\}, \{\mathsf{x}_1, \mathsf{x}_2\}, \ldots);$$

$$\pi_2 = (\{\mathsf{x}_1\}, \{\mathsf{x}_2\}, \{\mathsf{x}_2\}, \ldots);$$

$$\pi_3 = (\{\mathsf{x}_1\}, \emptyset, \{\mathsf{x}_2\}, \{\mathsf{x}_2\}, \ldots);$$

$$\pi_4 = (\{\mathsf{x}_1\}, \{\mathsf{x}_2\}, \emptyset, \{\mathsf{x}_1, \mathsf{x}_2\}, \{\mathsf{x}_1, \mathsf{x}_2\}, \ldots);$$

Safety Properties – Alternative Definition

Let $\sigma \in \left(2^P\right)^\omega$ be an arbitrary word, we formally define the set of all finite prefixes of σ as follows:

$$\texttt{pref}(\sigma) \triangleq \left\{(\sigma^0, \sigma^1, \ldots, \sigma^{k-1}, \sigma^k) \mid k \in \mathbb{N}_0\right\}$$

The definition of finite prefixes extends naturally to languages:

$$\texttt{pref}(L) \triangleq \bigcup_{\sigma \in L} \texttt{pref}(\sigma)$$

Finally, we define the **closure** of a language to contain all the words that share finite prefixes with the language:

$$\texttt{closure}(L) \triangleq \left\{\sigma \in \left(2^P\right)^\omega \mid \texttt{pref}(\sigma) \subseteq \texttt{pref}(L)\right\}$$

$L \subseteq closure(L)$

A property L_{safe} is a safety property if and only if $\texttt{closure}(L_{safe}) = L_{safe}$.

Liveness Properties

In simple terms, liveness properties say: "something good eventually happens".

An LT property L_{live} is a **liveness** property, if and only if $\texttt{pref}(L_{live}) = \left(2^P\right)^*$. closure $(L_{live}) = (2^P)^\omega$

Liveness Properties

In simple terms, liveness properties say: "something good eventually happens".

An LT property L_{live} is a **liveness** property, if and only if $\mathtt{pref}(L_{live}) = \left(2^P\right)^*$.

Example:

$\mathbf{F}(\neg \mathbf{x}_1 \wedge \mathbf{x}_2)$

Liveness and Safety

- Are there LT properties that are both safety and liveness properties?

- Is every LT property either a safety property or a liveness property?

Liveness and Safety

- Are there LT properties that are both safety and liveness properties?
  Yes! But only one, $\left(2^P\right)^\omega$.
- Is every LT property either a safety property or a liveness property?

Liveness and Safety

- Are there LT properties that are both safety and liveness properties?

  Yes! But only one, $\left(2^P\right)^\omega$.

- Is every LT property either a safety property or a liveness property?

  No, but every LT property is an intersection of a safety and a liveness property.

Liveness and Safety

- Are there LT properties that are both safety and liveness properties?

  Yes! But only one, $\left(2^P\right)^\omega$.

- Is every LT property either a safety property or a liveness property?

  No, but every LT property is an intersection of a safety and a liveness property.

EXAMPLE:

$\mathbf{x}_1 \, \mathbf{U} \, \mathbf{x}_2$

- $\mathbf{x}_1$ stays active until $\mathbf{x}_2$ activates;

- $\mathbf{x}_2$ eventually activates;

Liveness and Safety

- Are there LT properties that are both safety and liveness properties?

  Yes! But only one, $\left(2^{P}\right)^{\omega}$.

- Is every LT property either a safety property or a liveness property?

  No, but every LT property is an intersection of a safety and a liveness property.

EXAMPLE:

$\mathbf{x}_1 \,\mathbf{U}\, \mathbf{x}_2$

- $\mathbf{x}_1$ stays active until $\mathbf{x}_2$ activates;

  L_{safe} such that
  $\texttt{MinBad}(L_{safe}) = \left\{([\{\mathbf{x}_1, \neg\mathbf{x}_2\}]^k, \{\neg\mathbf{x}_1, \neg\mathbf{x}_2\}) \mid k \in \mathbb{N}_0\right\}$;

- $\mathbf{x}_2$ eventually activates;

Liveness and Safety

- Are there LT properties that are both safety and liveness properties?

  Yes! But only one, $\left(2^P\right)^\omega$.

- Is every LT property either a safety property or a liveness property?

  No, but every LT property is an intersection of a safety and a liveness property.

EXAMPLE:

$\mathbf{x}_1 \, \mathbf{U} \, \mathbf{x}_2$

- $\mathbf{x}_1$ stays active until $\mathbf{x}_2$ activates;

  L_{safe} such that
  $\texttt{MinBad}(L_{safe}) = \left\{ ([\{\mathbf{x}_1, \neg\mathbf{x}_2\}]^k, \{\neg\mathbf{x}_1, \neg\mathbf{x}_2\}) \mid k \in \mathbb{N}_0 \right\}$;

- $\mathbf{x}_2$ eventually activates;

  L_{live} given by $\mathbf{F}(\mathbf{x}_2)$;

Decomposition Theorem

For any LT property L, there exists a safety property L_{safe} and a liveness property L_{live} such that $L = L_{safe} \cap L_{live}$.

First, distributivity of union over closure:

$$\texttt{closure}(L) \cup \texttt{closure}(L') = \texttt{closure}(L \cup L')$$

$closure(L) \cup closure(L') \subseteq closure(L \cup L')$ $\qquad Z \subseteq Z' \Rightarrow closure(Z) \subseteq closure(Z')$

$pref(Z) \subseteq pref(Z')$

$L \subseteq L \cup L'$

$closure(L) \subseteq closure(L \cup L')$

$L' \subseteq L \cup L'$

$closure(L') \subseteq closure(L \cup L')$

$closure(L \cup L') \subseteq closure(L) \cup closure(L')$

we need: $pref(\sigma) \subseteq pref(L \cup L') \Rightarrow pref(\sigma) \subseteq pref(L) \vee pref(\sigma) \subseteq pref(L')$

$pref(L \cup L') = pref(L) \cup pref(L')$

$pref(\sigma)$ is an infinite set

either $pref(\sigma) \cap pref(L)$ is infinite or $pref(\sigma) \cap pref(L')$ is infinite or both

w.l.o.g. $pref(\sigma) \cap pref(L)$ is infinite

Now by contradiction: $pref(\sigma) \not\subseteq pref(L)$

$\exists k \in \mathbb{N}_0$ s.t. $\hat{\sigma} = (\sigma_0, \ldots, \sigma_k)$ is s.t. $\hat{\sigma} \notin pref(L)$

However $pref(\sigma) \cap pref(L)$ is infinite $\Rightarrow$ there has to infinitely many prefixes of σ longer than $\hat{\sigma}$ that are in $pref(L)$

Since $\hat{\sigma}$ is a prefix of any such longer prefix of σ, $\hat{\sigma}$ must also be in $pref(L)$ which is a contradiction with $pref(\sigma) \not\subseteq pref(L)$

Decomposition Theorem

For any LT property L, there exists a safety property L_{safe} and a liveness property L_{live} such that $L = L_{safe} \cap L_{live}$.

First, distributivity of union over closure:

$$\texttt{closure}(L) \cup \texttt{closure}(L') = \texttt{closure}(L \cup L')$$

Decomposition Theorem proof by construction:

$$L = \underbrace{\texttt{closure}(L)}_{L_{safe}} \cap \underbrace{\left(L \cup \left(\left(2^P\right)^\omega \setminus \texttt{closure}(L)\right)\right)}_{L_{live}}$$

$$closure(closure(L)) = closure(L)$$

$$closure(L \cup ((2^P)^\omega \setminus closure(L))) =$$

$$= closure(L) \cup closure((2^P)^\omega \setminus closure(L)) \supseteq$$

$$\supseteq closure(L) \cup ((2^P)^\omega \setminus closure(L)) = (2^P)^\omega$$

Finite Automata

A **nondeterministic finite automaton** (NFA) is a tuple $\mathcal{A} = (Q, \Sigma, \delta, Q_0, F)$ where:

- Q is a finite set of states;
- Σ is an alphabet (for us $\Sigma = 2^P$);
- $\delta \colon Q \times \Sigma \to 2^Q$ is a transition function;
- $Q_0 \subseteq Q$ is a set of initial states;
- $F \subseteq Q$ is the set of accepting/final states;

Given a finite word $w = w_1 w_2 \dots w_n \in \Sigma^*$, a **run** of $\mathcal{A}$ for w is a sequence of states $(q_0, q_1, \dots, q_n)$ such that:

- $q_0 \in Q_0$;
- For all $0 \leq i < n$, $q_{i+1} \in \delta(q_i, w_{i+1})$ (we write $q_i \xrightarrow{w_{i+1}} q_{i+1}$);

A run is **accepting** if it ends in a final state ($q_n \in F$).

Regular Languages

An NFA $\mathcal{A} = (Q, \Sigma, \delta, Q_0, F)$ defines a language of accepted words $L(\mathcal{A}) = \{w \in \Sigma^* |$ there exists an accepting run of $\mathcal{A}$ for $w\}$.

We say the language $L(\mathcal{A})$ is **accepted** by the automaton $\mathcal{A}$.

A language L which is accepted by some NFA $\mathcal{A}$, $L = L(\mathcal{A})$, is called **regular**.

Regular Languages

An NFA $\mathcal{A} = (Q, \Sigma, \delta, Q_0, F)$ defines a language of accepted words $L(\mathcal{A}) = \{w \in \Sigma^* |$ there exists an accepting run of $\mathcal{A}$ for $w\}$.

We say the language $L(\mathcal{A})$ is **accepted** by the automaton $\mathcal{A}$.

A language L which is accepted by some NFA $\mathcal{A}$, $L = L(\mathcal{A})$, is called **regular**.

EXAMPLE:

$L\left((a \mid b)^* b(a \mid b)\right)$ for $\Sigma = \{a, b\}$

Regular Languages

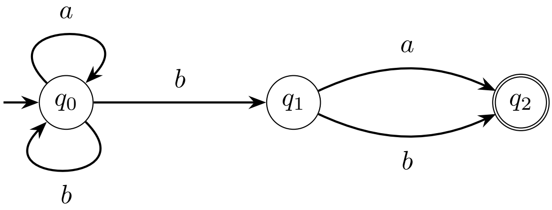
An NFA $\mathcal{A} = (Q, \Sigma, \delta, Q_0, F)$ defines a language of accepted words $L(\mathcal{A}) = \{w \in \Sigma^* |$ there exists an accepting run of $\mathcal{A}$ for $w\}$.

We say the language $L(\mathcal{A})$ is **accepted** by the automaton $\mathcal{A}$.

A language L which is accepted by some NFA $\mathcal{A}$, $L = L(\mathcal{A})$, is called **regular**.

EXAMPLE:

$L((a \mid b)^* b(a \mid b))$ for $\Sigma = \{a, b\}$



Regular Safety Properties

A safety property L_{safe} is regular, if the language $\texttt{Bad}(L_{safe})$ is regular, that is, if there exists an NFA $\mathcal{A}$ such that $L(\mathcal{A}) = \texttt{Bad}(L_{safe})$.

$\texttt{Bad}(L_{safe})$ is regular if and only if $\texttt{MinBad}(L_{safe})$ is regular.

Regular Safety Properties

A safety property L_{safe} is regular, if the language $\texttt{Bad}(L_{safe})$ is regular, that is, if there exists an NFA $\mathcal{A}$ such that $L(\mathcal{A}) = \texttt{Bad}(L_{safe})$.

$\texttt{Bad}(L_{safe})$ is regular if and only if $\texttt{MinBad}(L_{safe})$ is regular.

EXAMPLE:

$\mathbf{G}\,((\neg \mathbf{x}_2 \wedge \mathbf{X}(\mathbf{x}_2)) \Rightarrow \mathbf{x}_1)$

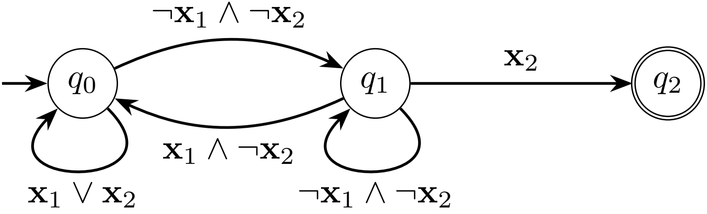
Regular Safety Properties

A safety property L_{safe} is regular, if the language $\mathtt{Bad}(L_{safe})$ is regular, that is, if there exists an NFA $\mathcal{A}$ such that $L(\mathcal{A}) = \mathtt{Bad}(L_{safe})$.

$\mathtt{Bad}(L_{safe})$ is regular if and only if $\mathtt{MinBad}(L_{safe})$ is regular.

EXAMPLE:

$\mathbf{G}\,((\neg \mathsf{x}_2 \wedge \mathbf{X}(\mathsf{x}_2)) \Rightarrow \mathsf{x}_1)$



Model Checking Regular Safety Properties

Let $\mathcal{T} = (S, \rightarrow, I, P, \alpha)$ be a transition system and let $\mathcal{A} = (Q, 2^P, \delta, Q_0, F)$ an NFA such that $Q_0 \cap F = \emptyset$.

Then their product is a transition system $\mathcal{T} \otimes \mathcal{A} = (S', \rightarrow', I', P', \alpha')$ where:

- $S' = S \times Q$;
- $\rightarrow' \subseteq (S \times Q) \times (S \times Q)$ such that $(\mathbf{x}, q) \rightarrow (\mathbf{x}', q') \Leftrightarrow \mathbf{x} \rightarrow \mathbf{x}' \wedge q \xrightarrow{\alpha(\mathbf{x}')} q'$;
- $I' = \left\{ (\mathbf{x}, q) \mid \mathbf{x} \in I \wedge \exists q_0 \in Q_0, q_0 \xrightarrow{\alpha(\mathbf{x})} q \right\}$;
- $P' = Q$;
- $\alpha' \colon S \times Q \rightarrow 2^Q$ such that $\alpha' \colon (\mathbf{x}, q) \mapsto \{q\}$;

Model Checking Regular Safety Properties

Let $\mathcal{T} = (S, \rightarrow, I, P, \alpha)$ be a transition system and let $\mathcal{A} = (Q, 2^P, \delta, Q_0, F)$ an NFA such that $Q_0 \cap F = \emptyset$.
Then their product is a transition system $\mathcal{T} \otimes \mathcal{A} = (S', \rightarrow', I', P', \alpha')$ where:

- $S' = S \times Q$;
- $\rightarrow' \subseteq (S \times Q) \times (S \times Q)$ such that $(\mathbf{x}, q) \rightarrow (\mathbf{x}', q') \Leftrightarrow \mathbf{x} \rightarrow \mathbf{x}' \wedge q \xrightarrow{\alpha(\mathbf{x}')} q'$;
- $I' = \left\{ (\mathbf{x}, q) \mid \mathbf{x} \in I \wedge \exists q_0 \in Q_0, q_0 \xrightarrow{\alpha(\mathbf{x})} q \right\}$;
- $P' = Q$;
- $\alpha' \colon S \times Q \rightarrow 2^Q$ such that $\alpha' \colon (\mathbf{x}, q) \mapsto \{q\}$;

Given a regular safety property L_{safe} and an NFA $\mathcal{A}$ such that $L(\mathcal{A}) = \mathtt{MinBad}(L_{safe})$:

$$\mathcal{T} \models L_{safe} \Longleftrightarrow \mathcal{T} \otimes \mathcal{A} \models L_{inv} = \{P_0, P_1, P_2 \dots \in \left(2^P\right)^\omega \mid \forall i \in \mathbb{N}_0, P_i \models \neg F\}$$

Büchi Automata

A **nondeterministic Büchi automaton** (NBA) is a tuple $\mathcal{A} = (Q, \Sigma, \delta, Q_0, F)$ where:

- Q is a finite set of states;
- Σ is an alphabet (for us $\Sigma = 2^P$);
- $\delta \colon Q \times \Sigma \to 2^Q$ is a transition function;
- $Q_0 \subseteq Q$ is a set of initial states;
- $F \subseteq Q$ is the set of accepting/final states;

Given an infinite word $\sigma = \sigma_0\sigma_1\sigma_2 \cdots \in \Sigma^\omega$, a **run** of $\mathcal{A}$ for σ is an infinite sequence of states $(q_0, q_1, q_2, \dots)$ such that:

- $q_0 \in Q_0$;
- For all $0 \leq i$, $q_i \xrightarrow{\sigma_i} q_{i+1}$;

A run is **accepting** if it visits an accepting state infinitely often.

ω-Regular Languages

A Büchi automaton $\mathcal{A} = (Q, \Sigma, \delta, Q_0, F)$ defines a language of accepted words $L(\mathcal{A}) = \{\sigma \in \Sigma^\omega |$ there exists an accepting run of $\mathcal{A}$ for $\sigma\}$.

We say the language $L(\mathcal{A})$ is **accepted** by the automaton $\mathcal{A}$.

A language L which is accepted by some Büchi automaton $\mathcal{A}$, $L = L(\mathcal{A})$, is called ω-**regular**.

ω-Regular Languages

A Büchi automaton $\mathcal{A} = (Q, \Sigma, \delta, Q_0, F)$ defines a language of accepted words $L(\mathcal{A}) = \{\sigma \in \Sigma^\omega |$ there exists an accepting run of $\mathcal{A}$ for $\sigma\}$.

We say the language $L(\mathcal{A})$ is **accepted** by the automaton $\mathcal{A}$.

A language L which is accepted by some Büchi automaton $\mathcal{A}$, $L = L(\mathcal{A})$, is called ω-**regular**.

EXAMPLE:

$L\left(a^+ b(b^+ \mid ba^+ b)^\omega\right)$ for $\Sigma = \{a, b\}$

ω-Regular Languages

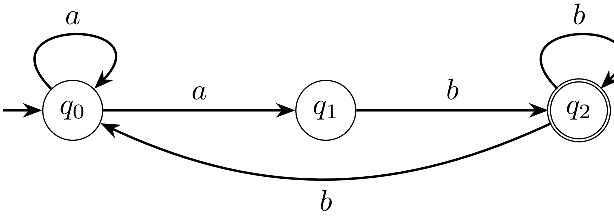
A Büchi automaton $\mathcal{A} = (Q, \Sigma, \delta, Q_0, F)$ defines a language of accepted words $L(\mathcal{A}) = \{\sigma \in \Sigma^\omega |$ there exists an accepting run of $\mathcal{A}$ for $\sigma\}$.

We say the language $L(\mathcal{A})$ is **accepted** by the automaton $\mathcal{A}$.

A language L which is accepted by some Büchi automaton $\mathcal{A}$, $L = L(\mathcal{A})$, is called ω-**regular**.

EXAMPLE:

$L\left(a^+b(b^+ \mid ba^+b)^\omega\right)$ for $\Sigma = \{a, b\}$



Persistence Properties

An LT property L_{pers} is a **persistence** property if there exists a propositional logic formula Φ over the atomic propositions P, such that:

$$L_{pers} = \left\{ P_0 P_1 P_2 \cdots \in \left(2^P\right)^{\omega} \mid \exists j \in \mathbb{N}_0, \forall i \geq j, P_i \models \Phi \right\}$$

The persistence property L_{pers} is given by the LTL formula $\mathbf{F}(\mathbf{G}(\Phi))$.

$\mathcal{T} \models L_{pers}$ can be verified by searching for "lasso", a reachable state $\mathbf{x}$ such that $\mathbf{x} \not\models \Phi$ which lies on a cycle.

Persistence Properties

An LT property L_{pers} is a **persistence** property if there exists a propositional logic formula Φ over the atomic propositions P, such that:

$$L_{pers} = \left\{ P_0 P_1 P_2 \dots \in \left(2^P\right)^\omega \mid \exists j \in \mathbb{N}_0, \forall i \geq j, P_i \models \Phi \right\}$$

The persistence property L_{pers} is given by the LTL formula $\mathbf{F}(\mathbf{G}(\Phi))$.

$\mathcal{T} \models L_{pers}$ can be verified by searching for "lasso", a reachable state $\mathbf{x}$ such that $\mathbf{x} \not\models \Phi$ which lies on a cycle.

EXAMPLE:

$\mathbf{F}(\mathbf{G}(\mathbf{x}_1 \vee \mathbf{x}_2))$

Model Checking ω-Regular Properties

Let $\mathcal{T} = (S, \rightarrow, I, P, \alpha)$ be a transition system and let $\mathcal{A} = (Q, 2^P, \delta, Q_0, F)$ a non-blocking Büchi automaton.

Then their product is a transition system $\mathcal{T} \otimes \mathcal{A} = (S \times Q, \rightarrow', I', Q, \alpha')$ where:

- $\rightarrow' \subseteq (S \times Q) \times (S \times Q)$ such that $(\mathbf{x}, q) \rightarrow (\mathbf{x}', q') \Leftrightarrow \mathbf{x} \rightarrow \mathbf{x}' \wedge q \xrightarrow{\alpha(\mathbf{x}')} q'$;
- $I' = \left\{(\mathbf{x}, q) \mid \mathbf{x} \in I \wedge \exists q_0 \in Q_0, q_0 \xrightarrow{\alpha(\mathbf{x})} q\right\}$;
- $\alpha' \colon S \times Q \rightarrow 2^Q$ such that $\alpha' \colon (\mathbf{x}, q) \mapsto \{q\}$;

Model Checking ω-Regular Properties

Let $\mathcal{T} = (S, \to, I, P, \alpha)$ be a transition system and let $\mathcal{A} = (Q, 2^P, \delta, Q_0, F)$ a non-blocking Büchi automaton.

Then their product is a transition system $\mathcal{T} \otimes \mathcal{A} = (S \times Q, \to', I', Q, \alpha')$ where:

- $\to' \subseteq (S \times Q) \times (S \times Q)$ such that $(\mathbf{x}, q) \to (\mathbf{x}', q') \Leftrightarrow \mathbf{x} \to \mathbf{x}' \wedge q \xrightarrow{\alpha(\mathbf{x}')} q'$;
- $I' = \left\{ (\mathbf{x}, q) \mid \mathbf{x} \in I \wedge \exists q_0 \in Q_0, q_0 \xrightarrow{\alpha(\mathbf{x})} q \right\}$;
- $\alpha' \colon S \times Q \to 2^Q$ such that $\alpha' \colon (\mathbf{x}, q) \mapsto \{q\}$;

Given an ω-regular property L and a Büchi automaton $\mathcal{A}$ such that $L(\mathcal{A}) = \left(2^P\right)^\omega \setminus L$:

$$\mathcal{T} \models L \iff \mathcal{T} \otimes \mathcal{A} \models L_{pers} = \{P_0 P_1 P_2 \dots \in \left(2^P\right)^\omega \mid \exists j \in \mathbb{N}_0, \forall i \geq j, P_i \models \neg F\}$$