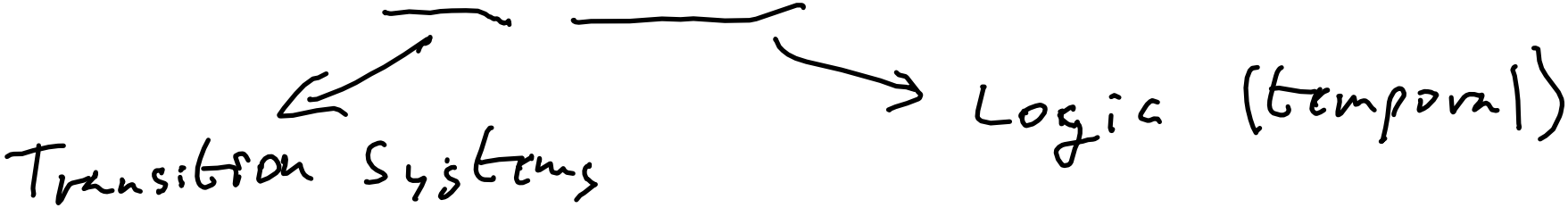
Model Checking

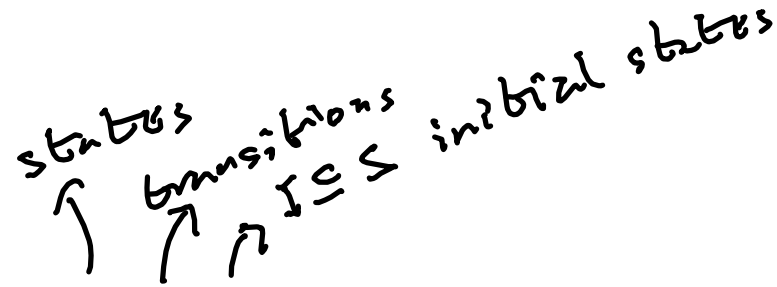
Juri Kolčák

Overview

Verification of **model properties**.



Kripke Structure



A transition system $TS = (S, \rightarrow, I, AP, L)$ enriched with a labelling function $L\colon S \rightarrow 2^{AP}$ defining a set of atomic propositions valid in each state.

Path fragment is a sequence of states $s_0, s_1, \ldots$ such that for all $i \in \mathbb{N}$, $(s_{i-1}, s_i) \in \rightarrow$.

A path fragment is **maximal** if it is infinite.

A path fragment is **initial** if $s_0 \in I$.

A **path** is an initial and maximal path fragment.

Given a path fragment $\pi = s_0, s_1, \ldots$, the **trace** of π is a word $L(s_0)L(s_1)\ldots$ over the alphabet 2^{AP}.

Linear-Time (LT) Properties

A language $P \subseteq (2^{AP})^{\omega}$.

A transition system TS satisfies a property P, $TS \models P \Leftrightarrow trace(TS) \subseteq P$

P language of all the "good" / allowed behaviours

Invariant Properties

An invariant condition given by a propositional formula Φ holds in every state.

$$P_{inv} = \{A_0 A_1 A_2 \cdots \in (2^{AP})^\omega \mid \forall j \in \mathbb{N}_0, A_j \models \Phi\}$$

$TS \models P_{inv} \Leftrightarrow \forall s \in Reach(TS), L(s) \models \Phi$

↪ We can use DFS

polynomial in both $|TS|$ and $|\Phi|$

Safety Properties

"Something bad never happens."

P is a safety property if for all words $\sigma \in (2^{AP})^\omega \setminus P$, there exists a finite prefix $\hat{\sigma}$ of σ such that $P \cap \{\sigma' \in (2^{AP})^\omega \mid \hat{\sigma} \text{ is a prefix of } \sigma'\} = \emptyset$

$\Rightarrow BadPref(P_{safe})$ .. set of all finite words which are not a prefix of any $\sigma \in P$

$(Min\,BadPref(P_{safe}))$

$TS \models P_{safe} \iff trace_{fin}(TS) \cap BadPref(P_{safe}) = \emptyset$

$$pref(\sigma) = \{\hat{\sigma} \mid \hat{\sigma} \text{ is a finite prefix of } \sigma\}$$

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

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

(Progress)

Liveness Properties

"Eventually something good happens."

P is a liveness property if and only if $pref(P) = (2^{AP})^*$.

$P = (2^{AP})^\omega$ the only property which is both safety and liveness

Any LT prop. can be expressed as an intersection of a liveness and a safety property

$$P = \underbrace{closure(P)}_{\text{safety } P_{safe}} \cap \underbrace{\left(P \cup \left((2^{AP})^\omega \setminus closure(P)\right)\right)}_{\text{liveness } P_{live}}$$

$closure(P_{live}) = (2^{AP})^\omega$

$closure(P_{live}) \supseteq closure(P) \cup \left((2^{AP})^\omega \setminus closure(P)\right) = (2^{AP})^\omega$

Fairness

"Everyone gets their turn."

Different fairness constraints are considered.

Unconditional Fairness Every process gets its turn infinitely often.

Strong Fairness Every process that is enabled infinitely often gets its turn infinitely often.

Weak Fairness Every process that is continuously enabled from some point onwards gets its turn infinitely often.

Regular Properties

As a property P defines a language, we say it is regular whenever it defines a regular language.

NFA $(Q, \Sigma, \delta : Q \times \Sigma \to 2^Q, Q_0 \subseteq Q, F \subseteq Q)$

regular safety properties, we can build NFA recognising $BadPref(P_{safe})$

$BadPref(P_{safe})$ is regular $\iff$ $MinBadPref(P_{safe})$ is reg.

$TS \otimes A = (S', \to', I', AP', L')$

$S' = S \times Q$

$\to'$ is the smallest relation satisfying

$$\frac{s \to t \ \wedge \ q \xrightarrow{L(t)} p}{\langle s, q \rangle \to' \langle t, p \rangle}$$

$I' = \{ \langle s_0, q \rangle \mid s_0 \in I \wedge \exists q_0 \in Q_0, \ q_0 \xrightarrow{L(s_0)} q \}$

$AP' = O$

$L' : S \times Q \to 2^Q \qquad L' : \langle s, q \rangle \mapsto \{q\}$

$TS \models P_{safe} \iff TS \otimes A \models P_{inv(A)}$

invariant is $\neg F$

ω-Regular Properties

A language (property) is ω-regular if there exists an ω-regular expression producing it.

$$G = E_1 . F_1^\omega + \dots + E_n F_n^\omega \qquad n \in \mathbb{N}$$

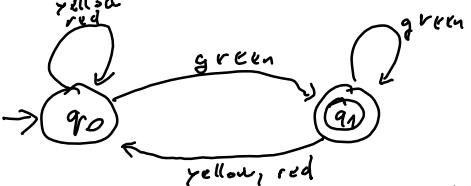
each E_i and F_i are regular expressions

$$\varepsilon \notin L(F_i) \qquad L_\omega(G) = L(E_1) L(F_1)^\omega \cup \dots \cup L(E_n) . L(F_n)^\omega$$

ω-regular languages are recognised by nondeterministic Büchi automata.

$$(Q, \Sigma, \delta : Q \times \Sigma \to 2^Q, Q_0 \subseteq Q, F \subseteq Q)$$

Büchi automata accept words that visit an accepting state infinitely often



$$TS \otimes A = (S', \to', I', AP', L')$$

$$S' = S \times Q$$

$\to'$ smallest relation satisfying

$$\frac{s \to t \wedge q \xrightarrow{L(t)} p}{\langle s, q \rangle \to' \langle t, p \rangle}$$

$$I' = \{\langle s_0, q \rangle \mid s_0 \in I \wedge \exists\, q_0 \in Q_0, q_0 \xrightarrow{L(s_0)} q\}$$

$$AP' = Q$$

$$L' : S \times Q \to 2^Q \qquad L' : \langle s, q \rangle \mapsto \{q\}$$

$$P_{pers} = \{A_0 A_1 A_2 \dots \mid \exists i \in \mathbb{N}_0, \forall j \geq i, A_j \vdash \Phi\}$$

$$TS \vdash P \iff TS \otimes A \models P_{pers(A)}$$

A recognise $(2^{AP})^\omega \setminus P$

$$\Phi = \neg F$$

Temporal Logic

Temporal modalities on top of predicate logic:

- $\Diamond$ "Eventually";
- $\Box$ "Always";

Linear Temporal Logic (LTL)

↪ reasoning about paths

next $\bigcirc\varphi$

$\Diamond\varphi \triangleq \top \mathbf{U} \varphi$

$\Box\varphi \triangleq \neg\Diamond\neg\varphi$

Syntax:

$$\varphi ::= \top \mid a \mid \varphi_1 \wedge \varphi_2 \mid \neg\varphi \mid \bigcirc\varphi \mid \varphi_1 \mathbf{U} \varphi_2$$

Semantics over words: = traces

$$\begin{array}{lcl}
\sigma \models \top & & \\
\sigma \models a & \Leftrightarrow & a \in A_0 \\
\sigma \models \varphi_1 \wedge \varphi_2 & \Leftrightarrow & \sigma \models \varphi_1 \text{ and } \sigma \models \varphi_2 \\
\sigma \models \neg\varphi & \Leftrightarrow & \sigma \not\models \varphi \\
\sigma \models \bigcirc\varphi & \Leftrightarrow & \sigma[1:] \models \varphi \\
\sigma \models \varphi_1 \mathbf{U} \varphi_2 & \Leftrightarrow & \exists j \in \mathbb{N}_0, \sigma[j:] \models \varphi_2 \text{ and } \forall i \leq j, \sigma[i:] \models \varphi_1
\end{array}$$

$\Box(\text{green} \rightarrow (\text{green } \mathbf{U} \text{ yellow})) \wedge \Box(\text{yellow} \rightarrow (\text{yellow } \mathbf{U} \text{ red})) \ldots$

LTL formulae can be translated into corresponding Büchi automata automatically.

In the worst case, this transformation is exponential

$$\mathcal{O}(|TS| \cdot 2^{|\varphi|})$$

Fairness constraints can be expressed directly in LTL

→ but explicit fairness MC is often more efficient

Computational Tree Logic (CTL)

Syntax:

state formula $\Phi ::= \top \mid a \mid \Phi_1 \wedge \Phi_2 \mid \neg\Phi \mid \exists\varphi \mid \forall\varphi$

path formula $\varphi ::= \bigcirc\Phi \mid \Phi_1 \,\mathbf{U}\, \Phi_2$

Semantics over states and paths:

$s \models \top$

$s \models a \quad \Leftrightarrow \quad a \in L(s)$

$s \models \Phi_1 \wedge \Phi_2 \quad \Leftrightarrow \quad s \models \Phi_1$ and $s \models \Phi_2$

$s \models \neg\Phi \quad \Leftrightarrow \quad s \not\models \Phi$

$s \models \exists\varphi \quad \Leftrightarrow \quad \exists\pi \in Paths(s), \pi \models \varphi$

$s \models \forall\varphi \quad \Leftrightarrow \quad \forall\pi \in Paths(s), \pi \models \varphi$

$\pi \models \bigcirc\Phi \quad \Leftrightarrow \quad \pi[1] \models \Phi$

$\pi \models \Phi_1 \,\mathbf{U}\, \Phi_2 \quad \Leftrightarrow \quad$ ~~$\pi \in Paths(s), \pi \models \varphi$~~

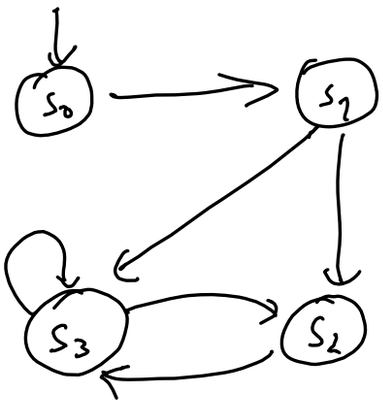
$\exists j \in \mathbb{N}\ \pi[j] \models \Phi_2 \ \wedge \ \forall i \leq j\ \pi[i] \models \Phi$

Compute a set of states $Sat(\Phi)$ which satisfy Φ

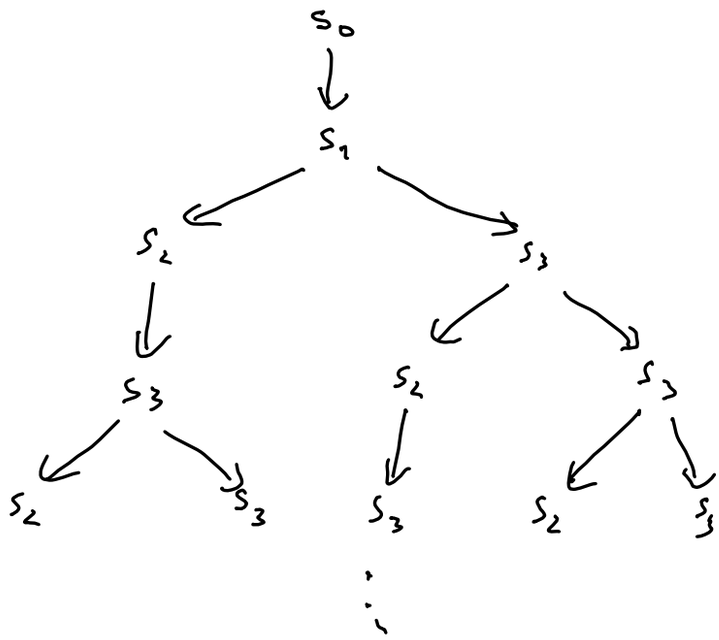
↳ recursively → parse tree of Φ

and go "bottom-up"

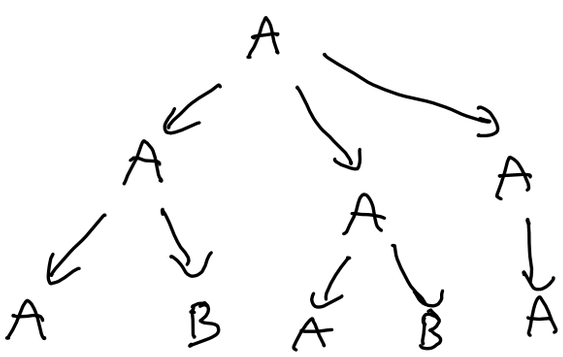
$\mathcal{O}(|TS| \cdot |\Phi|)$



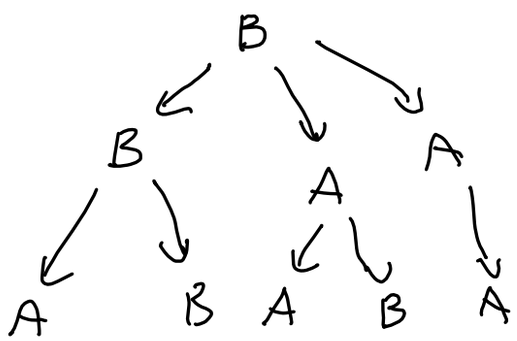
"Unfolding"



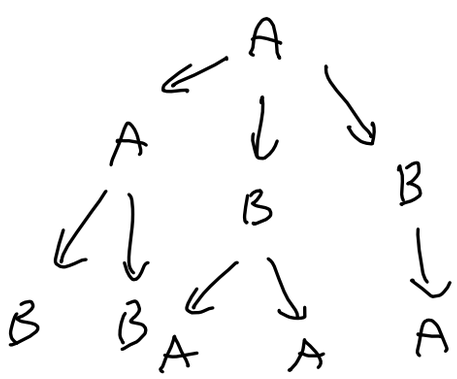
$\exists \Diamond B$



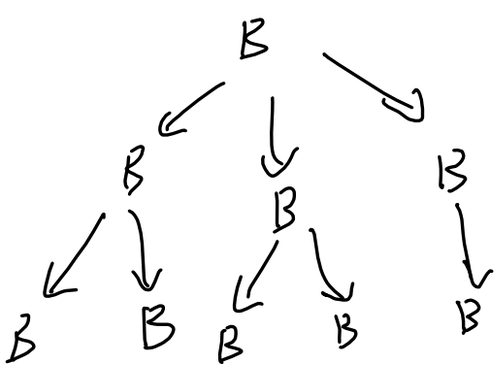
$\exists \Box B$



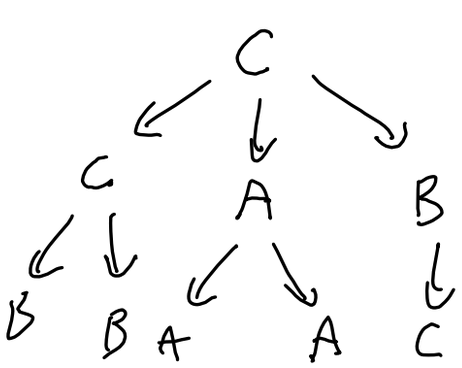
$\forall \Diamond B$



$\forall \Box B$



$\exists C \cup B$



$\forall C \cup B$