Labelled transition systems

A labelled transition system on a set of actions $A$ is a pair $T = (S, \delta)$, where

- $S$ is a set of states,
- $\delta \subseteq S \times A \times S$ is the transition relation

A labelled transition system is finite if $S$ is finite.
Labelled transition systems

- A labelled transition system on a set of actions $A$ is a pair $T = (S, \delta)$, where
  - $S$ is a set of states,
  - $\delta \subseteq S \times A \times S$ is the transition relation
- A labelled transition system is finite if $S$ is finite
- For process $E$, we denote by $T_E = (S_E, \delta_E)$ the labelled transition system associated to $E$, inductively defined as follows:
  1. $E \in S_E$, and
  2. if $F \in S_E$, and $F \xrightarrow{a} G$, then $G \in S_E$ and $(F, a, G) \in \delta_E$.

Model checking CTL

- Given: a process $E$, a formula $\phi$ of CTL\(^{-}\).

- Decide: does $E$ satisfies $\phi$?
- Restrict to finite $T_E$ and for convenience we add negation to CTL\(^{-}\)
- For formula $\psi$, $[ \neg \psi ]$ is the set of states of $T_E$ satisfying $\psi$.
- Sketch of the algorithm:
  1. Compute the subformulas of $\phi$
  2. Compute $[ \neg \psi ]$ for each subformula $\psi$ of $\phi$, starting with the smallest subformulas and moving up to larger and larger subformulas
  3. Answer: "$E$ satisfies $\phi$" iff $E \in [ \neg \phi ]$.
Given: a process $E$, a formula $\phi$ of $\text{CTL}^-$.  
Decide: does $E$ satisfies $\phi$?  
Restrict to finite $T_E$ and for convenience we add negation to $\text{CTL}^-$.  
For formula $\psi$, $[[\psi]]$ is the set of states of $T_E$ satisfying $\psi$.  
Sketch of the algorithm:
Given: a process $E$, a formula $\phi$ of CTL$^\neg$.

Decide: does $E$ satisfies $\phi$?

Restrict to finite $T_E$ and for convenience we add negation to CTL$^\neg$.

For formula $\psi$, $[\psi]$ is the set of states of $T_E$ satisfying $\psi$.

Sketch of the algorithm:
1. Compute the subformulas of $\phi$
2. Compute $[\psi]$ for each subformula $\psi$ of $\phi$, starting with the smallest subformulas and moving up to larger and larger subformulas
3. Answer: “$E$ satisfies $\phi$” iff $E \in [\phi]$

Computing $[\psi]$: the easy cases

Because of the equivalences

- $[K]\psi \equiv \neg\langle K \rangle \neg \psi$
- $AG \psi \equiv \neg EF \neg \psi$
- $AF \psi \equiv \neg EG \neg \psi$

we can assume that $\phi$ does not contain $[K]$, $AG$ or $AF$ operators.

$[tt] = S_E$

$[ff] = \emptyset$

$[\psi_1 \land \psi_2] = [\psi_1] \cap [\psi_2]$

$[\psi_1 \lor \psi_2] = [\psi_1] \cup [\psi_2]$

$[\neg \psi] = S_E \setminus [\psi]$

$[\langle K \rangle \psi_1] = pre_K([\psi_1])$
Computing $[\psi]$: the easy cases

- Because of the equivalences
  \[ [K]\psi \equiv \neg\langle K \rangle \neg\psi \]
  \[ AG\psi \equiv \neg EF \neg\psi \]
  \[ AF\psi \equiv \neg EG \neg\psi \]
  we can assume that $\phi$ does not contain $[K]$, AG or AF operators.

\[
\begin{align*}
[tt] &= S_E \\
[ff] &= \emptyset \\
[\psi_1 \land \psi_2] &= [\psi_1] \cap [\psi_2] \\
[\psi_1 \lor \psi_2] &= [\psi_1] \cup [\psi_2] \\
[\neg\psi] &= S_E \setminus [\psi] \\
\langle K \rangle\psi_1 &= pre_K([\psi_1])
\end{align*}
\]

- $pre_K([\psi_1]) \overset{\text{def}}{=} \text{states from which some state in } [\psi_1] \text{ can be reached through some action in } K$.

Computing $[EF\psi_1]$

Let $pre(\cdot)$ denote $pre_A(\cdot)$.

\[
\begin{algorithm}
\text{Input: } T_E, [\psi_1] \\
\text{Output: } [EF\psi_1] \\
\text{Initialize } C := \emptyset; \\
\text{Iterate } C := [\psi_1] \cup pre(C) \\
\text{until a fixpoint is reached; return } C \\
\end{algorithm}
\]

Complexity: $O(|S_E| \cdot (|S_E| + |\delta_E|))$

Better algorithm: explore each state only once using depth-first or breadth-first search. (Complexity: $O(|S_E| + |\delta_E|)$)

Computing $[EG\psi_1]$

\[
\begin{algorithm}
\text{Input: } T_E, [\psi_1] \\
\text{Output: } [EG\psi_1] \\
\text{Initialize } C := S_E; \\
\text{Initialize } D := \text{states of } S_E \text{ without successors;}
\text{Iterate } C := [\psi_1] \cap (pre(C) \cup D) \\
\text{until a fixpoint is reached; return } C \\
\end{algorithm}
\]

Complexity: $O(|S_E| \cdot (|S_E| + |\delta_E|))$
Computing $[\mathbf{EG} \; \psi_1]$ II

An algorithm with $O(|S_E| + |\delta_E|)$ complexity:

- Compute $D' := \text{states of } S_E \text{ without successors in } [\psi_1]$
- Compute the labelled transition system $T'_E = ([\psi_1], \delta_E \cap ([\psi_1] \times A \times [\psi_1])$
- Compute the set of states $C$ that belong to some strongly connected component of $T'_E$.
- Using the algorithm for $[\mathbf{EF} \; \psi_1]$ case, compute the states from which some state in $C \cup D'$ can be reached (using transitions of $T'_E$ only).

Complexity of the complete model-checking algorithm

- A formula $\phi$ has at most $|\phi|$ subformulas (where $|\phi|$ is the length of $\phi$).
- So the algorithms for the easy cases, for $\mathbf{EF} \; \psi$, and for $\mathbf{EG} \; \psi$ have to be executed altogether at most $|\phi|$ times.
- Each execution of one of the algorithms takes at most $O(|S_E| + |\delta_E|)$ time (using the fast algorithms).
Complexity of the complete model-checking algorithm

- A formula $\phi$ has at most $|\phi|$ subformulas (where $|\phi|$ is the length of $\phi$).
- So the algorithms for the easy cases, for $EF \psi$, and for $EG \psi$ have to be executed altogether at most $|\phi|$ times.
- Each execution of one of the algorithms takes at most $O(|SE| + |\delta_E|)$ time (using the fast algorithms).
- So the overall complexity is $O(|\phi| \cdot (|SE| + |\delta_E|))$.

Fixpoint view of the algorithms

- The following equivalences hold:
  
  $EF \phi \equiv \phi \lor (\neg)EF \phi$
  $EG \phi \equiv \phi \land (\neg)EG \phi \lor [-]ff$

- So we have
  
  $\llbracket EF \phi \rrbracket = \llbracket \phi \rrbracket \cup pre(\llbracket EF \phi \rrbracket)$
  $\llbracket EG \phi \rrbracket = \llbracket \phi \rrbracket \cap (pre(\llbracket EG \phi \rrbracket) \cup \llbracket [-]ff \rrbracket)$

- And so $\llbracket EF \phi \rrbracket$ and $\llbracket EG \phi \rrbracket$ are solutions of the equations
  
  $X = \llbracket \phi \rrbracket \cup pre(X) \overset{def}{=} ef(X)$
  $X = \llbracket \phi \rrbracket \cap (pre(X) \cup \llbracket [-]ff \rrbracket) \overset{def}{=} eg(X)$
Fixpoint view of the algorithms

The following equivalences hold:

\[
\begin{align*}
\text{EF} \phi & \equiv \phi \lor (\Box)\text{EF} \phi \\
\text{EG} \phi & \equiv \phi \land ((\Box)\text{EG} \phi \lor \Box \bot)
\end{align*}
\]

So we have

\[
\begin{align*}
[\text{EF} \phi] & = [\phi] \cup \text{pre}([\text{EF} \phi]) \\
[\text{EG} \phi] & = [\phi] \cap (\text{pre}([\text{EG} \phi]) \cup [\Box \bot])
\end{align*}
\]

and so \([\text{EF} \phi]\) and \([\text{EG} \phi]\) are solutions of the equations

\[
\begin{align*}
X & = [\phi] \cup \text{pre}(X) & \text{def} & \text{ef}(X) \\
X & = [\phi] \cap (\text{pre}(X) \cup [\Box \bot]) & \text{def} & \text{eg}(X)
\end{align*}
\]

These solutions are fixpoints of the mappings \(\text{ef}\) and \(\text{eg}\).

Proposition: \([\text{EF} \phi]\) is the smallest solution (least fixpoint) of \(X = \text{ef}(X)\).

Proof: Notice that \([\text{EF} \phi]\) is the smallest solution (least fixpoint) of \(X = \text{ef}(X)\).

Let \(X_0\) be an arbitrary solution.

We prove \(\text{pre}^i([\phi]) \subseteq X_0\) for every \(i \geq 0\) by induction on \(i\).

Base: \(i = 0\). Obvious from \(X_0 = [\phi] \cup \text{"something"}\).

Step: Assume \(\text{pre}^i([\phi]) \subseteq X_0\). Then:

\[
\begin{align*}
\text{pre}^{i+1}([\phi]) & = \text{pre}(\text{pre}^i([\phi])) & \text{(definition of pre)} \\
\subseteq & \text{pre}(X_0) & \text{(induction hypothesis)} \\
\subseteq & X_0 & (X_0 = \text{pre}(X_0) \cup \text{"something"})
\end{align*}
\]
Which solutions?

Proposition: \([\text{EF } \phi]\) is the smallest solution (least fixpoint) of \(X = \text{ef}(X)\).

Proof: Notice that \([\text{EF } \phi]\) = \(\bigcup_{i \geq 0} \text{pre}^i(\{\phi\})\), where \(\text{pre}^0(\{\phi\}) \overset{\text{def}}{=} \{\phi\}\).
Let \(X_0\) be an arbitrary solution.
We prove \(\text{pre}^i(\{\phi\}) \subseteq X_0 \) for every \(i \geq 0\) by induction on \(i\).
Base: \(i = 0\). Obvious from \(X_0 = \{\phi\} \cup \text{"something"}\).
Step: Assume \(\text{pre}^i(\{\phi\}) \subseteq X_0\). Then:

\[
\text{pre}^{i+1}(\{\phi\}) \\
= \text{pre}(\text{pre}^i(\{\phi\})) \\
\subseteq \text{pre}(X_0) \\
\subseteq X_0 \\
\]

Proposition: \([\text{EG } \phi]\) is the largest solution (greatest fixpoint) of \(X = \text{eg}(X)\).
Proof: Exercise

Fixpoint algorithms

- The mappings \(\text{ef}(X)\) and \(\text{eg}(X)\) are monotonic, i.e., if \(X \subseteq Y\), then \(\text{ef}(X) \subseteq \text{ef}(Y)\) and \(\text{eg}(X) \subseteq \text{eg}(Y)\).
- Given a finite set \(S\) and a monotonic mapping \(m: 2^S \rightarrow 2^S\), the least fixpoint of \(m\) exists, is unique, and can be calculated by iteratively computing \(\emptyset, m(\emptyset), m^2(\emptyset), \ldots\) until \(m^i(\emptyset) = m^{i+1}(\emptyset)\). The least fixpoint is \(m^i(\emptyset)\).
Fixpoint algorithms

- The mappings $ef(X)$ and $eg(X)$ are monotonic, i.e., if $X \subseteq Y$, then $ef(X) \subseteq ef(Y)$ and $eg(X) \subseteq eg(Y)$.
- Given a finite set $S$ and a monotonic mapping $m : 2^S \rightarrow 2^S$,
- the least fixpoint of $m$ exists, is unique, and can be calculated by iteratively computing $\emptyset, m(\emptyset), m^2(\emptyset), \ldots$ until $m^i(\emptyset) = m^{i+1}(\emptyset)$. The least fixpoint is $m^i(\emptyset)$;
- the greatest fixpoint of $m$ exists, is unique, and can be calculated by iteratively computing $S, m(S), m^2(S), \ldots$ until $m^i(S) = m^{i+1}(S)$. The greatest fixpoint is $m^i(S)$.

Applications

- Fixpoint theory allows to easily derive algorithms for other temporal operators.

$$E_0 \models \phi \text{ EU } \psi \text{ iff } \text{ for some run } E_0 \xrightarrow{a_1} E_1 \xrightarrow{a_2} \ldots, \text{ for some } i \geq 0, E_i \models \psi, \text{ and for all } j < i, E_j \models \phi$$
- Equivalence: $\phi \text{ EU } \psi \equiv \psi \lor (\phi \land \langle - \rangle \phi \text{ EU } \psi)$
- So: $[\phi \text{ EU } \psi] = [\psi] \cup ([\phi] \cap \text{pre}([\phi \text{ EU } \psi]))$
Applications

Fixpoint theory allows to easily derive algorithms for other temporal operators.

\[ E_0 \models \phi \text{ EU } \psi \iff \text{ for some run } E_0 \xrightarrow{a_1} E_1 \xrightarrow{a_2} \ldots \]

for some \( i \geq 0 \), \( E_i \models \psi \), and for all \( j < i \), \( E_j \models \phi \).

Equivalence: \( \phi \text{ EU } \psi \equiv \psi \lor (\phi \land \langle \neg \rangle \phi \text{ EU } \psi) \)

So: \( \llbracket \phi \text{ EU } \psi \rrbracket = \llbracket \psi \rrbracket \cup (\llbracket \phi \rrbracket \cap \text{pre}(\llbracket \phi \text{ EU } \psi \rrbracket)) \)

\( \llbracket \phi \text{ EU } \psi \rrbracket \) is solution of the equation

\[ X = \llbracket \psi \rrbracket \cup (\llbracket \phi \rrbracket \cap \text{pre}(X)) \overset{\text{def}}{=} eu(X) \]

The logic of the Workbench

In order to encode CTL\(^-\) in the Workbench’s logic, we write

\[
\begin{align*}
\text{prop } AG(P) &= \max(Z.P \land [\neg]Z) \\
\text{prop } EF(P) &= \min(X.P \lor \langle\neg\rangle X) \\
\text{prop } AF(P) &= \min(X.P \lor \langle\leftrightarrow\rangle T \land [\neg]X) \\
\text{prop } EG(P) &= \max(X.P \land [\neg]F \lor \langle\neg\rangle X)
\end{align*}
\]

These definitions correspond to recursive equations.

The logic of the Workbench

Fixpoint theory allows to easily derive algorithms for other temporal operators.

\[ E_0 \models \phi \text{ EU } \psi \iff \text{ for some run } E_0 \xrightarrow{a_1} E_1 \xrightarrow{a_2} \ldots \]

for some \( i \geq 0 \), \( E_i \models \psi \), and for all \( j < i \), \( E_j \models \phi \).

Equivalence: \( \phi \text{ EU } \psi \equiv \psi \lor (\phi \land \langle \neg \rangle \phi \text{ EU } \psi) \)

So: \( \llbracket \phi \text{ EU } \psi \rrbracket = \llbracket \psi \rrbracket \cup (\llbracket \phi \rrbracket \cap \text{pre}(\llbracket \phi \text{ EU } \psi \rrbracket)) \)

\( \llbracket \phi \text{ EU } \psi \rrbracket \) is solution of the equation

\[ X = \llbracket \psi \rrbracket \cup (\llbracket \phi \rrbracket \cap \text{pre}(X)) \overset{\text{def}}{=} eu(X) \]

It is the smallest solution, and so, since \( eu \) is monotonic, we can compute \( \llbracket \phi \text{ EU } \psi \rrbracket \) as the stabilizing point of \( \emptyset, e(u(\emptyset)), e(u^2(\emptyset)), \ldots \)
In order to encode CTL $\Box$ in the Workbench’s logic, we write

- $\text{prop AG}(P) = \max(Z.P \land [-]Z)$;
- $\text{prop EF}(P) = \min(X.P \lor <-X)$;
- $\text{prop AF}(P) = \min(X.P \lor (\langle\rangle T \land [-]X))$;
- $\text{prop EG}(P) = \max(X.P \land (-]F \lor <-X))$.

These definitions correspond to recursive equations.

E.g., the definition of $\text{EF} \phi$ states that $\llbracket \text{EF} \phi \rrbracket$ is the smallest solution ($\min$) of the equation

$$X = \llbracket \phi \rrbracket \lor \text{pre}(X)$$

In other words, in the Workbench a property is defined through a (possibly recursive) equation.