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.

Restrict to finite transition systems: where $A$ and $S$ (and $\delta$) are finite.
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
- Restrict to finite transition systems: where $A$ and $S$ (and $\delta$) are 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 $(F, a, G) \in \delta_E$. 

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
- Restrict to finite transition systems: where $A$ and $S$ (and $\delta$) are 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⁻.

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
- Restrict to finite transition systems: where $A$ and $S$ (and $\delta$) are 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 $\phi$?
- **For convenience** we add negation to CTL$^-$
- **For formula** $\psi$, $\llbracket \psi \rrbracket$ is the set of states of $T_E$ satisfying $\psi$.
- **Sketch of the algorithm:**
  1. Compute the subformulas of $\phi$
  2. Compute $\llbracket \psi \rrbracket$ for each subformula $\psi$ of $\phi$, starting with the smallest subformulas and then with larger and larger subformulas
  3. Answer: "$E$ satisfies $\phi$" iff $E \in \llbracket \phi \rrbracket$
Model checking $\text{CTL}^-$

- **Given**: a process $E$, a formula $\phi$ of $\text{CTL}^-$.
- **Decide**: does $E$ satisfies $\phi$?
- **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:**
  1. Compute the subformulas of $\phi$
  2. Compute $[\psi]$ for each subformula $\psi$ of $\phi$, starting with the smallest subformulas and then with larger and larger subformulas
  3. Answer: "$E$ satisfies $\phi$" iff $E \in [\phi]$

Computing $[\psi]$: the easy cases

- **Because of the equivalences**
  
  $[K]\psi \equiv \neg (K)\neg \psi$
  
  $AG \psi \equiv \neg EF \neg \psi$
  
  $AF \psi \equiv \neg EG \neg \psi$

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

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

Model checking $\text{CTL}^-$

- **Given**: a process $E$, a formula $\phi$ of $\text{CTL}^-$.
- **Decide**: does $E$ satisfies $\phi$?
- **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:**
  1. Compute the subformulas of $\phi$
  2. Compute $[\psi]$ for each subformula $\psi$ of $\phi$, starting with the smallest subformulas and then with larger and larger subformulas
Computing $\llbracket \psi \rrbracket$: the easy cases

- Because of the equivalences
  
  $[K]\psi \equiv \neg (K) \neg \psi$
  
  $AG \psi \equiv \neg EF \neg \psi$
  
  $AF \psi \equiv \neg EG \neg \psi$

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

  
  $[tt] = S_E$
  
  $[ff] = \emptyset$
  
  $[\psi_1 \land \psi_2] = [\psi_1] \cap [\psi_2]$
  
  $[\psi_1 \lor \psi_2] = [\psi_1] \cup [\psi_2]$
  
  $[\neg \psi] = S_E \setminus [\psi]$
  
  $[(K)\psi_1] = pre_K([\psi_1])$

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

Computing $\llbracket EF \psi_1 \rrbracket$

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

\begin{itemize}
  \item Input: $T_E$, $[\psi_1]$
  \item Output $\llbracket EF \psi_1 \rrbracket$
  \item Initialize $C := [\psi_1]$;
  \item Iterate $C := C \cup pre(C)$ until a fixpoint is reached;
  \item return $C$
\end{itemize}

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 $\llbracket EG \psi_1 \rrbracket$

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

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 $D' \coloneqq$ states of $S_E$ 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 $O(|S_E| + |\delta_E|)$ time (using the fast algorithms).
- So the overall complexity is

$$O(|\phi| \cdot (|S_E| + |\delta_E|))$$

Fixpoint view of the algorithms

- The following equivalences hold:
  $$EF \phi \equiv \phi \lor \langle - \rangle EF \phi$$
  $$EG \phi \equiv \phi \land (\langle - \rangle EG \phi \lor [-]ff)$$
- So we have
  $$\begin{align*}
  [EF \phi] & = [\phi] \cup \text{pre}([EF \phi]) \\
  [EG \phi] & = [\phi] \cap (\text{pre}([EG \phi]) \cup [[-]ff])
  \end{align*}$$
- and so $[EF \phi]$ and $[EG \phi]$ are solutions of the equations
  $$X = [\phi] \cup \text{pre}(X) \overset{\text{def}}{=} ef(X)$$
  $$X = [\phi] \cap (\text{pre}(X) \cup [[-]ff]) \overset{\text{def}}{=} eg(X)$$
Fixpoint view of the algorithms

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

- So we have
  
  \[ \llbracket \text{EF} \phi \rrbracket = \llbracket \phi \rrbracket \cup \text{pre}(\llbracket \text{EF} \phi \rrbracket) \]
  
  \[ \llbracket \text{EG} \phi \rrbracket = \llbracket \phi \rrbracket \cap (\text{pre}(\llbracket \text{EG} \phi \rrbracket) \cup \llbracket \neg \text{ff} \rrbracket) \]

- and so \( \llbracket \text{EF} \phi \rrbracket \) and \( \llbracket \text{EG} \phi \rrbracket \) are solutions of the equations

  \[ X = \llbracket \phi \rrbracket \cup \text{pre}(X) \quad \text{def} \quad \text{ef}(X) \]
  
  \[ X = \llbracket \phi \rrbracket \cap (\text{pre}(X) \cup \llbracket \neg \text{ff} \rrbracket) \quad \text{def} \quad \text{eg}(X) \]

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

Which solutions?

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

**Proof:** Notice that \( \llbracket \text{EF} \phi \rrbracket = \bigcup_{i \geq 0} \text{pre}^i(\llbracket \phi \rrbracket) \), where \( \text{pre}^0(\llbracket \phi \rrbracket) \triangleq \llbracket \phi \rrbracket \).

Let \( X_0 \) be an arbitrary solution.

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

**Base:** \( i = 0 \). Obvious from \( X_0 = \llbracket \phi \rrbracket \cup \text{“something”} \).

**Step:** Assume \( \text{pre}^i(\llbracket \phi \rrbracket) \subseteq X_0 \). Then:

\[
\text{pre}^{i+1}(\llbracket \phi \rrbracket) \\
= \text{pre}(\text{pre}^i(\llbracket \phi \rrbracket)) \quad \text{(definition of pre)} \\
\subseteq \text{pre}(X_0) \quad \text{(induction hypothesis)} \\
\subseteq X_0 \quad \text{(} X_0 = \text{pre}(X_0) \cup \text{“something”)}
\]
Which solutions?

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

**Proof:** Notice that $[EF \phi] = \bigcup_{i \geq 0} pre^i([\phi])$, where $pre^0([\phi]) = [\phi]$.

Let $X_0$ be an arbitrary solution.
We prove $pre^i([\phi]) \subseteq X_0$ for every $i \geq 0$ by induction on $i$.

**Base:** $i = 0$. Obvious from $X_0 = [\phi] \cup “something”$.

**Step:** Assume $pre^i([\phi]) \subseteq X_0$. Then:

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

**Proposition:** $[EG \phi]$ is the largest solution (greatest fixpoint) of $X = eg(X)$.

**Proof:** Exercise

---

**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 $m'(\emptyset) = m^{i+1}(\emptyset)$. The least fixpoint is $m'(\emptyset)$.\]
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 $m^*(\emptyset)$.
- 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 $m^*(S)$.

Applications

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

\[
E_0 \models \phi \mathsf{EU} \psi \quad \text{iff} \quad \text{for some run } E_0 \xrightarrow{a_1} E_1 \xrightarrow{a_2} \cdots, \\
\text{for some } i \geq 0, E_i \models \psi, \text{ and} \\
\text{for all } j < i, E_j \models \phi
\]

- Equivalence: $\phi \mathsf{EU} \psi \equiv \psi \lor (\phi \land (\neg)\phi \mathsf{EU} \psi)$

- Equivalence: $\phi \mathsf{EU} \psi \equiv \psi \lor (\phi \land (\neg)\phi \mathsf{EU} \psi)$

- So: $[\phi \mathsf{EU} \psi] = [\psi] \cup ([\phi] \cap \text{pre}([\phi \mathsf{EU} \psi]))$
The logic of the Workbench

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

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

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

▶ Equivalence: \( \phi \text{ EU } \psi \equiv \psi \lor (\phi \land \lnot \phi \text{ EU } \psi) \)

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

▶ [\phi \text{ EU } \psi] is solution of the equation

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

▶ It is the smallest solution, and so, since \( \text{eu} \) is monotonic, we can compute \( [\phi \text{ EU } \psi] \) as the stabilizing point of \( \emptyset, \text{eu}(\emptyset), \text{eu}^2(\emptyset), \ldots \).

 Applications

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

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

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

▶ Equivalence: \( \phi \text{ EU } \psi \equiv \psi \lor (\phi \land \lnot \phi \text{ EU } \psi) \)

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

▶ [\phi \text{ EU } \psi] is solution of the equation

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

▶ These definitions correspond to recursive equations.

The logic of the Workbench

▶ In order to encode CTL− in the Workbench’s logic, we write

\[
\text{prop } \text{AG}(P) = \max(Z.P \land \lnot Z);
\]

\[
\text{prop } \text{EF}(P) = \min(X.P \land \lnot X);
\]

\[
\text{prop } \text{AF}(P) = \min(X.P \land (\lnot X \land \lnot T));
\]

\[
\text{prop } \text{EG}(P) = \max(X.P \land \lnot F);
\]

▶ In order to encode CTL− in the Workbench’s logic, we write

\[
\text{prop } \text{AG}(P) = \max(Z.P \land \lnot Z);
\]

\[
\text{prop } \text{EF}(P) = \min(X.P \land \lnot X);
\]

\[
\text{prop } \text{AF}(P) = \min(X.P \land (\lnot X \land \lnot T));
\]

\[
\text{prop } \text{EG}(P) = \max(X.P \land \lnot F);
\]

▶ These definitions correspond to recursive equations.
The logic of the Workbench

In order to encode CTL$^{-}$ in the Workbench’s logic, we write

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

These definitions correspond to recursive equations.

E.g., the definition of $\mathsf{EF} \phi$ states that $\llbracket \mathsf{EF} \phi \rrbracket$ 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.