Formal Methods in Software Development

Ivano Salvo
and Igor Melatti

Computer Science Department
Sapienza Università di Roma

Lesson **9**, November 19^{th}, 2019

Equivalences, Reloaded

Basic idea: Having to check $\mathcal{M} \models \varphi$, find a (hopefully) **smaller system $\mathcal{M}'$**, such that $\mathcal{M} \models \varphi$ if and only if $\mathcal{M}' \models \varphi$.

This idea is related to the definition of **some equivalence $\cong$** among transition systems or Kripke structures, so that $\mathcal{M} \cong \mathcal{M}'$.

The equivalence $\cong$ should be invariant for the logic at hand.

As a matter of fact, depending also on the property φ (and the temporal logic at hand), **many behaviours of $\mathcal{M}$ can be irrelevant** to the satisfaction of $\mathcal{M} \models \varphi$.

For example, **stuttering equivalence is invariant for LTL_{-X}**

Ideally:

- $\mathcal{M}'$ should be much smaller than $\mathcal{M}$.
- The computation of $\mathcal{M}'$ should be much faster than checking $\mathcal{M} \models \varphi$.

Lesson 9a:

Simulation and Bisimulation

Bisimulation

Bisimulation plays a central role in the Theory of Concurrency (usually in an action-oriented version).

Bisimulation usually is defined as the **maximum equivalence** satisfying certain properties (see Definition in the next slide), so it is usually defined as a **maximum fixpoint**.

It has been introduced in the framework of Process Algebras (and of course, Labeled Transition Systems).

Bisimulation

Definition: Let $\mathcal{M}$=(S, R, L, S_0, AP) and $\mathcal{M}'$=(S', R', L', S'_0, AP) be two Kripke structures with the **same set of atomic propositions**.

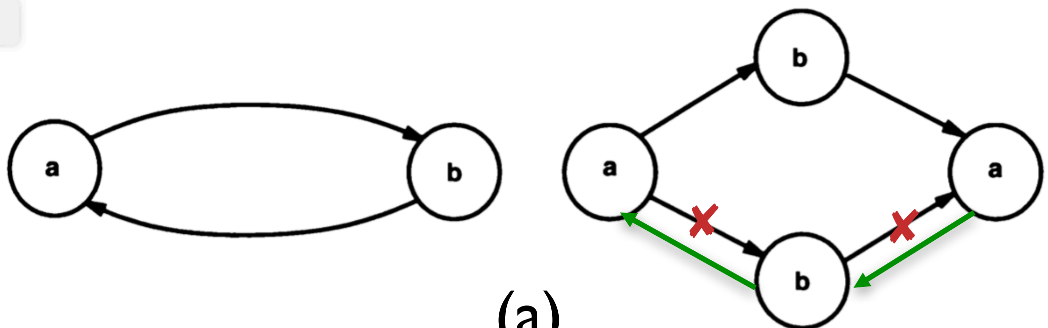
A relation $B \subseteq S \times S'$ is a **bisimulation** relation iff for all $(s, s') \in B$ we have:

1. $L(s) = L(s')$
2. For all t such that $R(s, t)$ there exists t' such that $R(s', t')$ and $B(t, t')$
3. For all t' such that $R(s', t')$ there exists t such that $R(s, t)$ and $B(t, t')$

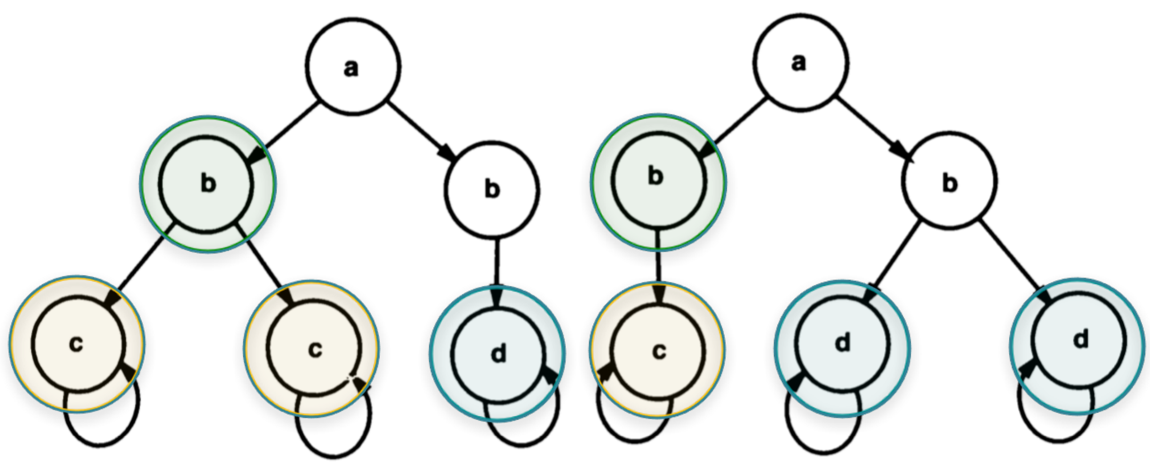
Two Kripke structures are **bisimulation equivalent** if for each initial state $s \in S_0$ in $\mathcal{M}$ there exists an initial state $s' \in S'_0$ in $\mathcal{M}'$ such that $B(s, s')$ and for each initial state $s' \in S'_0$ in $\mathcal{M}'$ there exists an initial state $s \in S_0$ in $\mathcal{M}$ such that $B(s, s')$.

Bisimulation: Examples

Bisimulation preserves some operations like **unwinding** (a) and **duplication** (b) of sub-structures.



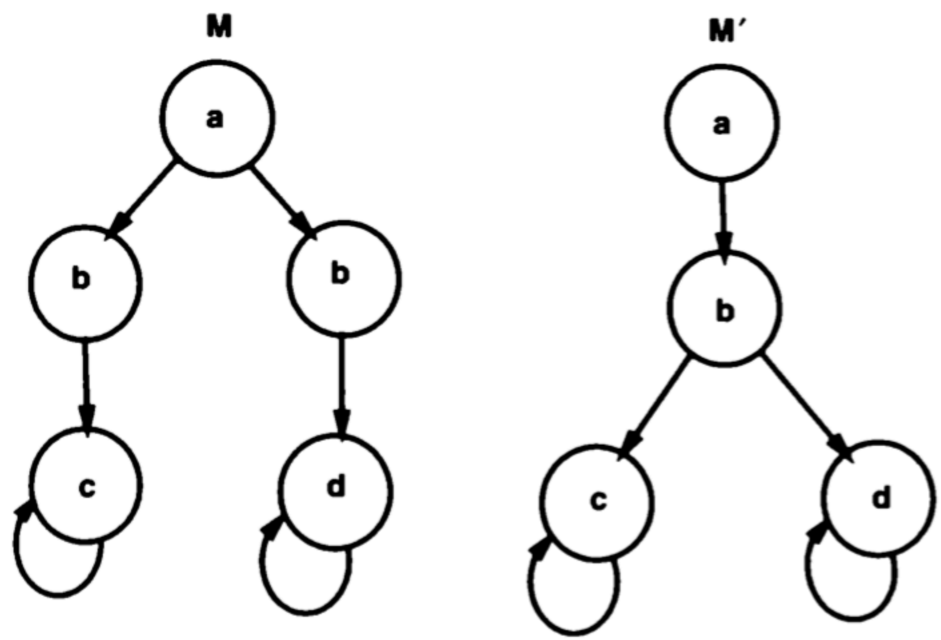
(a)



(b)

Bisimulation: Examples

Bisimulation takes a **branching-time** perspective: these systems are not bisimilar, because $\mathcal{M}'$ defers a decision to go in c or d. $\mathcal{M}'$ is "stronger" than $\mathcal{M}$: $\mathcal{M}'$ **simulates** $\mathcal{M}$ in the sense it can reply to any action of $\mathcal{M}$ (not viceversa) **bisimulation game**)



Corresponding paths

Definition: Two paths $\pi = s_0 s_1 \ldots s_i \ldots$ in $\mathcal{M}$ and $\pi' = s_0' s_1' \ldots s_i' \ldots$ in $\mathcal{M}'$ correspond if for all i, we have $B(s_i, s_i')$.

Lemma: Let s, s' be such that $B(s, s')$. Then, for every path starting from s, there exists a corresponding path starting from s' and viceversa.

Proof: Let $\pi = s_0 s_1 \ldots s_i \ldots$ in $\mathcal{M}$ with $s = s_0$. We prove the statement by induction on i. Clearly we put $s'_0 = s'$. Let us now assume that $B(s_i, s_i')$ holds. Because $B(s_i, s_i')$ and $R(s_i, s_{i+1})$ there exists a state s_i'' such that $R(s_i', s_i'')$ and $B(s_{i+1}, s_i'')$ and we clearly choose s_i'' as s'_{i+1}. Symmetrically, given a path π' in $\mathcal{M}'$ we can construct a corresponding path in $\mathcal{M}$. □

CTL and bisimulation*

Lemma: Let f be a CTL* formula and let s, s' be such that $B(s, s')$ and π, π' be corresponding path starting from (resp.) s, s'. Then:

- If f is a state formula, $\mathcal{M}, s \vDash f \Leftrightarrow \mathcal{M}', s' \vDash f$
- If f is a path formula, $\mathcal{M}, \pi \vDash f \Leftrightarrow \mathcal{M}', \pi' \vDash f$

Proof: (Easy induction on the structure of f).

[**BASE**] Let f be an atomic proposition p. We know that if $B(s, s')$ then $L(s)=L'(s')$ and hence $\mathcal{M}, s \vDash p \Leftrightarrow \mathcal{M}', s' \vDash p$.

[**IND.**] Let $f \equiv \neg g$ be a state formula. $\mathcal{M}, s \vDash \neg g \Leftrightarrow_{\text{def of } \neg} \mathcal{M}, s \nvDash g \Leftrightarrow$ (IND. HYP) $\mathcal{M}, s' \nvDash g \Leftrightarrow_{\text{def}} \mathcal{M}, s' \vDash \neg g$. The same if f is a path form.

Let $f \equiv g \vee h$ be a state formula. $\mathcal{M}, s \vDash g \vee h \Leftrightarrow_{\text{def}} \mathcal{M}, s \vDash g$ or $\mathcal{M}, s \vDash h \Leftrightarrow$ (IND. HYP) $\mathcal{M}, s' \vDash g$ or $\mathcal{M}, s' \vDash h \Leftrightarrow_{\text{def of sat of } \vee} \mathcal{M}, s' \vDash g \vee h \equiv f$. The same if f is a path formula.

Let $f \equiv \mathbf{E}\, g$. If $\mathcal{M}, s \vDash \mathbf{E}\, g$ then there exists a path π starting in s such that $\pi \vDash g$. Then there exists a corresponding path π' in $\mathcal{M}'$ starting in s', and by (IND. HYP) $\pi' \vDash g \Leftrightarrow \pi \vDash g$. By def. This implies that $\mathcal{M}, s' \vDash \mathbf{E}\, g$. The converse is the same. (to be cntd →

CTL and bisimulation*

Let $f \equiv \mathbf{X}\, g$ be a path formula. $\mathcal{M}, \pi \vDash \mathbf{X}\, g \Leftrightarrow_{\text{def of } \mathbf{X}} \mathcal{M}, \pi^1 \vDash g$. Since by hypothesis we have a corresponding path π', we have also that π^1 corresponds to π'^1 and hence, by IND. HYP., $\mathcal{M}, \pi'^1 \vDash g \Leftrightarrow_{\text{def of } \mathbf{X}} \mathcal{M}, \pi' \vDash \mathbf{X}\, g$. The same argument works for the conv.

Let $f \equiv g\, \mathbf{U}\, h$ be a path formula. By definition of **U**, there exists k such that $\mathcal{M}, \pi^k \vDash h$ and $\mathcal{M}, \pi^j \vDash g$ for all $0 \leq j < k$. Since π' corresponds to π, we have, by using IND. HYP. $\mathcal{M}, \pi'^k \vDash h$ and $\mathcal{M}, \pi'^j \vDash g$ for all $0 \leq j < k$, that is (by def. of **U**) $\mathcal{M}, \pi' \vDash g\, \mathbf{U}\, h$. The same argument works for the converse.

The case $f \equiv g\, \mathbf{R}\, h$ is similar to $f \equiv g\, \mathbf{U}\, h$, the case $f \equiv \mathbf{A}\, g$ is similar to $f \equiv \mathbf{E}\, g$, and the case $f \equiv g \wedge h$ is similar to $f \equiv g \vee h$.$\square$

Theorem: Let f be a CTL* formula and $B(s, s')$. Then $\mathcal{M}, s \vDash f \Leftrightarrow \mathcal{M}', s \vDash f$.

Theorem: Let f be a CTL* formula and $B(\mathcal{M}, \mathcal{M}')$. Then $\mathcal{M} \vDash f \Leftrightarrow \mathcal{M}' \vDash f$.

Bisimulation: CTL versus CTL*

Theorem: Let f be a CTL* formula and $B(\mathcal{M}, \mathcal{M}')$.
Then $\mathcal{M} \models f \Leftrightarrow \mathcal{M}' \models f$.

Interestingly, **this theorem holds also for CTL**! Therefore, if two structures can be distinguished by a CTL* formula, they can be distinguished also by a CTL formula.

This **does not mean** that CTL and CTL* **have the same expressive power**.

CTL* and CTL would be equivalent if for each CTL* formula it would exist a CTL formula with the same set of models (Kripke structures). But this is known to be false!

Here, we are just saying that, for each model there exists a CTL formula that is true in that model but false in any inequivalent model (with respect to **bisimulation** – remember that LTL is sensible to **stuttering equivalence**).

Abstraction: simulation

Often, it is interesting to consider an abstraction $\mathcal{A}$ of a system $\mathcal{M}$ with the property that all behaviors of $\mathcal{M}$ is also a behaviour of $\mathcal{A}$ (but not necessarily the converse). Abstraction $\mathcal{A}$ may have some **spurious behaviour**.

Definition: Let $\mathcal{M}=(S, R, L, I, AP)$ and $\mathcal{M}'=(S', R', L', I', AP')$ be two Kripke structures with $AP' \subseteq AP$.

A relation $H \subseteq S \times S'$ is a **simulation** iff for all $(s, s') \in H$:

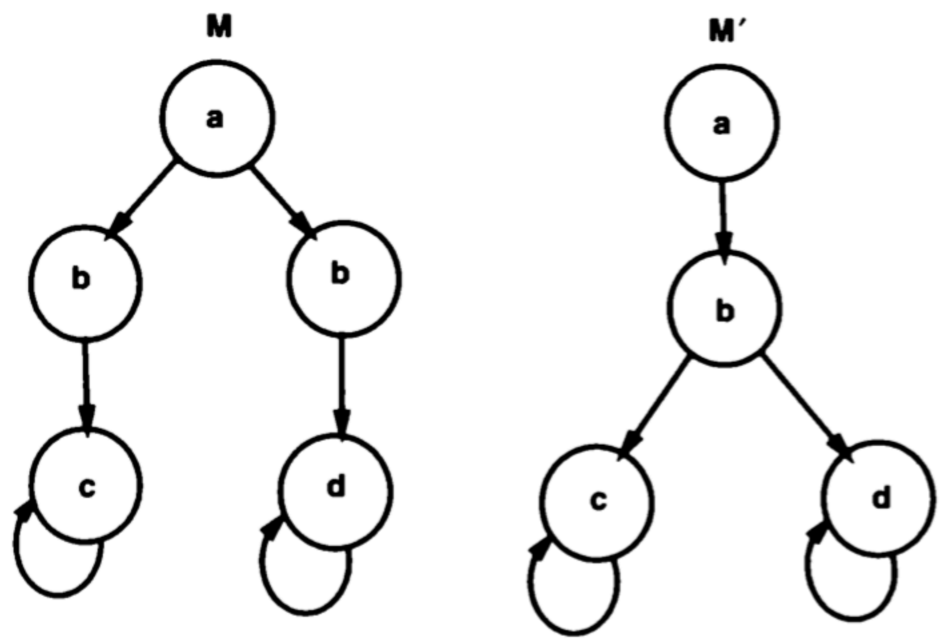
1. $L(s) \cap AP' = L(s')$
2. For all t such that $R(s, t)$ there exists t' such that $R(s', t')$ and $H(t, t')$

We say that $\mathcal{M}'$ **simulates** $\mathcal{M}$, notation $\mathcal{M} \leqslant \mathcal{M}'$ if for each state $s \in S_0$ in $\mathcal{M}$ there exists an initial state $s' \in I'$ in $\mathcal{M}'$ such that $H(s, s')$.

Proposition: $\leqslant$ is a preorder on the set of Kripke structures.

Simulation: Examples

If we consider the relation $H=\{ (s, s') \mid L(s) = L(s') \}$ it is easy to see that $\mathcal{M} \preccurlyeq \mathcal{M}'$. As a simulation game, $\mathcal{M}'$ can always `reply' to any move of $\mathcal{M}$.



The logic ACTL* and simulation

ACTL(*) is the restriction of CTL(*) that considers only the universal path quantifier **A** and negations only on atomic proposition (otherwise, **implicit existentials** would be present).

Lemma: Let s, s' be such that $H(s, s')$. Then, for every path starting from s, there exists a corresponding (with respect to H) path starting from s'.

Theorem: If $\mathcal{M} \preccurlyeq \mathcal{M}'$ then for each ACTL* formula f, $\mathcal{M}' \models f$ **implies** $\mathcal{M} \models f$.

This theorem holds (intuitively) because ACTL* formulas quantify over all behaviours of a Kripke structures $\mathcal{M}$ and if a formula holds for all behaviour of $\mathcal{M}'$ then it holds for all behaviour of $\mathcal{M}$.

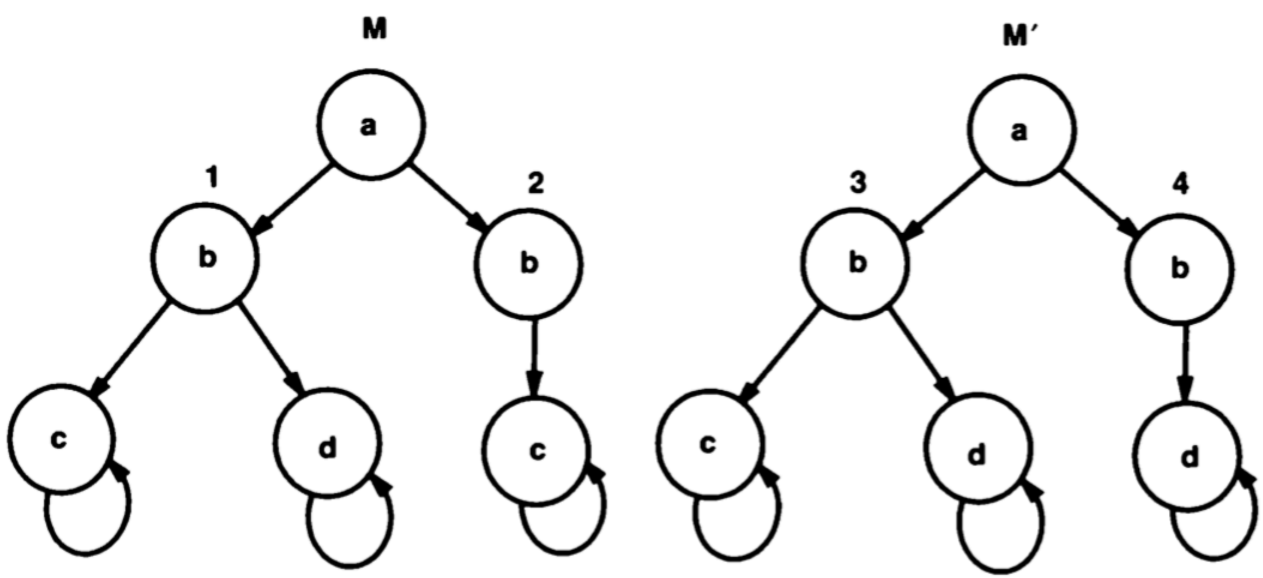
On the other hand, if $\mathcal{M}' \not\models f$, nothing can be deduced for $\mathcal{M}$. We have to check if the counterexample is a counterexample also for $\mathcal{M}$. The counterexample may drive the consideration of another structure $\mathcal{M}''$ with $\mathcal{M} \preccurlyeq \mathcal{M}'' \preccurlyeq \mathcal{M}'$ and try $\mathcal{M}'' \models f$ (**counterexample guided refinement**)

Simulation: Examples

In this example, $\mathcal{M} \preccurlyeq \mathcal{M}'$ and $\mathcal{M}' \preccurlyeq \mathcal{M}$ but $\mathcal{M}$ and $\mathcal{M}'$ are not bisimilar. State 1 of $\mathcal{M}$ simulates both states 3 and 4 of $\mathcal{M}'$. Similarly, state 3 of $\mathcal{M}'$ simulates both states 1 and 2 of $\mathcal{M}$.

They are not bisimilar because no state in $\mathcal{M}$ can be associated to state 4 in $\mathcal{M}'$. No state in $\mathcal{M}'$ to state 2 in $\mathcal{M}'$.

Using logic characterisation of bisimulation, $\mathcal{M} \vDash \mathbf{AG}\ (b \rightarrow \mathbf{EX}\ c)$ but $\mathcal{M}' \nvDash \mathbf{AG}\ (b \rightarrow \mathbf{EX}\ c)$



Checking (bi)simulation

We compute a sequence of relations $B_0\, B_1 \ldots$ in $S \times S'$ as follows:

$B_0(s, s')$ iff $L(s) = L(s')$

$B_{n+1}(s, s')$ iff
$B_n(s, s')$ and
$\forall t\, [R(s, t) \Rightarrow [\exists t'\, R(s', t') \wedge B_n(t, t')]]$ and
$\forall t'\, [R(s', t') \Rightarrow [\exists t\, R(s, t) \wedge B_n(t, t')]]$

Note that $B_n \supseteq B_{n+1}$ for all n. Therefore, **we are computing a greatest fixpoint**! We know that **there exists n such that $B_{n+1} = B_n$**. We can define $B^* = \cap_n B_n$.

Proposition: B^* is the largest bisimulation between $\mathcal{M}$ and $\mathcal{M}'$.

Proof: We show that for any bisimulation B, $B \subseteq B^*$. Induction on n. Clearly, $B \subseteq B_0$ (cond. 1 in def. of bisim.). Assume $B \subseteq B_n$ and $B(s, s')$. If $R(s, t)$ then $R'(s', t')$ and $B(t, t')$ and the symmetric case. This implies $B_{n+1}(s, s')$ and hence $B \subseteq B_{n+1}$. □

Lesson 9b:

Yet Another Tableau Construction

Checking ACTL formulas

We present here a tableau construction for the logic ACTL.

ACTL considers only the universal path quantifier **A**.

To avoid implicit existential path quantifier, negation are allowed only on atomic propositions.

To maintain expressive power, both $\wedge$ and $\vee$ are in the logic, as well as both **U** and **R** (**F** and **G** can be derived from **U** and **R**).

For any ACTL formula *f*, the tableau $\mathcal{T}_f$ is a maximal model for *f* with respect to $\leqslant_F$ (we use this property in abstractions, next topic). That is $\mathcal{M} \models f$ iff $\mathcal{M} \leqslant_F \mathcal{T}_f$.

Eventualities (formula of the shape **A**[*g* **U** *h*]) are satisfied by means of fair paths. States that are not at the beginning of fair paths will be characterized by formula of the shape **AX** false.

Elementary Formulas

The Kripke structure $\mathcal{T}_f$ is on the set of atomic proposition AP_f of atomic propositions occurring as sub-formulas of *f*.

Each state $s \in S_T = \mathcal{P}(el(f))$ is a set of **elementary propositions**.

1. $el(p) = el(\neg p) = \{p\}$ if $p \in AP_f$.
2. $el(g_1 \vee g_2) = el(g_1 \wedge g_2) = el(g_1) \cup el(g_2)$.
3. $el(\mathbf{AX}\, g_1) = \{\mathbf{AX}\, g_1\} \cup el(g_1)$.
4. $el(\mathbf{A}[g_1\, \mathbf{U}\, g_2]) = \{\mathbf{AX}\, \mathit{False}, \mathbf{AX}(\mathbf{A}[g_1\, \mathbf{U}\, g_2])\} \cup el(g_1) \cup el(g_2)$.
5. $el(\mathbf{A}[g_1\, \mathbf{R}\, g_2]) = \{\mathbf{AX}\, \mathit{False}, \mathbf{AX}(\mathbf{A}[g_1\, \mathbf{R}\, g_2])\} \cup el(g_1) \cup el(g_2)$.

The labeling $L_T(s)$ is defined so each state is labeled with the set of atomic propositions contained in the state.

Building the transition relation

To define the transition relation, we need to define the set of states that satisfies a given formula in $el(f)$ as follows (observe why we don't need to add negations in $el(f)$):

1. $sat(True) = S_T$ and $sat(False) = \emptyset$.
2. $sat(g) = \{s \mid g \in s\}$ where $g \in el(f)$.
3. $sat(\neg g) = \{s \mid g \notin s\}$ where g is an atomic proposition. Recall that only atomic propositions can be negated in ACTL.
4. $sat(g \vee h) = sat(g) \cup sat(h)$.
5. $sat(g \wedge h) = sat(g) \cap sat(h)$.
6. $sat(\mathbf{A}[g\ \mathbf{U}\ h]) = \big(sat(h) \cup \big(sat(g) \cap sat(\mathbf{AX}(\mathbf{A}[g\ \mathbf{U}\ h]))\big)\big) \cup sat(\mathbf{AX}\ False)$.
7. $sat(\mathbf{A}[g\ \mathbf{R}\ h]) = \big(sat(h) \cap \big(sat(g) \cup sat(\mathbf{AX}(\mathbf{A}[g\ \mathbf{R}\ h]))\big)\big) \cup sat(\mathbf{AX}\ False)$.

Differently from LTL, we want to define R_T in such a way that $\mathcal{T}_f$ has all behaviours that satisfies f. As usual, **AX** is the key.

$$R_T(s_1, s_2) = \bigwedge_{\mathbf{AX}g \in el(f)} s_1 \in sat(\mathbf{AX}\ g) \Rightarrow s_2 \in sat(g).$$

Pay attention!

Fairness constraints

Similarly to LTL, **eventually properties** are fullfilled along fair paths. A state can be in sat(**AX A**[g **U** h]) without satisfying **AX A**[g **U** h] only if there exists a path starting from s in sat(**AX A**[g **U** h]) $\cap$ ($S_T \setminus$ sat(h)).

Therefore, we impose fairness constraints containing the complement of these sets:

$$F_T = \{S_T \setminus \text{sat}(\mathbf{AX\ A}[g\ \mathbf{U}\ h]) \cup \text{sat}(h) \mid \mathbf{AX\ A}[g\ \mathbf{U}\ h] \in \text{el}(f)\}$$

Lemma: For all sub-formulas g of f, if $s \in$ sat(g) then $s \vDash g$.

By putting the set of initial states $S_0{}^T$ = sat(f), we have that $\mathcal{T}_f \vDash f$. Let $\mathcal{M} \vDash f$, we define: $H=\{(s, s') \mid s = \{g \in \text{el}(f) \mid s' \vDash g\}\}$. Then:

Lemma: $H(s, s')$ then $s \vDash g$ implies $s' \vDash g$.

Lemma: H is a fair simulation between $\mathcal{M}$ and $\mathcal{T}_f$.

All this implies that if $\mathcal{M} \vDash f$ then $\mathcal{M} \leqslant_F \mathcal{T}_f$.

Lesson 9c:

Compositional Reasoning

Assume-Guarantee paradigm

Many complex systems consist of many sub-systems.

Remember that the parallel composition of two systems result in a combinatorial explosion of the number of states with respect to sub-components.

It would be desirable to deduce **global properties** from **local properties** of sub-systems (**compositionality**).

Let us consider a system $\mathcal{M}=\mathcal{M}_1 \mid \mathcal{M}_2$: the behavior of $\mathcal{M}_1$ depends on $\mathcal{M}_2$: one can specify assumptions that must be satisfied by $\mathcal{M}_2$ in order to guarantee the correctness of $\mathcal{M}_1$.

At the same time, the behavior of $\mathcal{M}_2$ depends on $\mathcal{M}_1$: one can specify assumptions that must be satisfied by $\mathcal{M}_1$ in order to guarantee the correctness of $\mathcal{M}_2$.

Idea: By combining the set of assumed and guaranteed properties by $\mathcal{M}_1$ and $\mathcal{M}_2$ it is possible establish correctness of the whole system $\mathcal{M}_1 \mid \mathcal{M}_2$.

Formulas and Inference Rules

A formula is a triple of the shape $\langle f \rangle\ \mathcal{M}\ \langle g \rangle$ where f and g are temporal logic formulas and $\mathcal{M}$: the intended meaning is that whenever $\mathcal{M}$ is part of a system satisfying an assumption g, then the system must also guarantee the porperty f.

We can express system properties as inference rules:

$$\frac{\langle true \rangle \mathcal{M}_1 \langle g \rangle \qquad \langle g \rangle \mathcal{M}_2 \langle f \rangle}{\langle true \rangle \mathcal{M}_1 | \mathcal{M}_2 \langle f \rangle}$$

Be careful to avoid circularity in inference rules. Some deductions that seems reasonable are wrong! For example, the following inference rule is **unsound**:

$$\frac{\langle g \rangle \mathcal{M}_1 \langle f \rangle \qquad \langle f \rangle \mathcal{M}_2 \langle g \rangle}{\mathcal{M}_1 | \mathcal{M}_2 \vDash f \wedge g}$$

For example, let $\mathcal{M}_1$=**wait**(y=1); x=1; and $\mathcal{M}_2$=**wait**(x=1); y=1; and f =**AF** (y=1) and g =**AF** (x=1): the premises of the rule holds, but not the conclusions!

Composition of structures

Definition: Let $\mathcal{M}_1$=(S_1, I_1, AP_1, L_1, R_1, F_1) and $\mathcal{M}_2$=(S_2, I_2, AP_2, L_2, R_2, F_2) be two fair Kripke structures. We define the parallel composition $\mathcal{M}_1 \mid \mathcal{M}_2$ = (S, I, AP, L, R, F) of $\mathcal{M}_1$ and $\mathcal{M}_2$ as:

- $S = \{ (s_1, s_2) \mid L(s_1) \cap AP_2 = L(s_2) \cap AP_1 \}$
- $I = (I_1 \times I_2) \cap S$
- $AP = AP_1 \cup AP_2$
- $L(s_1, s_2) = L(s_1) \cup L(s_2)$
- $R((s_1, s_2), (t_1, t_2))$ iff $R_1(s_1, t_1)$ and $R_2(s_2, t_2)$
- $F = \{ (P \times S_2) \mid P \in F_1 \} \cup \{ (S_1 \times P) \mid P \in F_2 \}$

Observation: The definition of F is such that a path in $\mathcal{M}_1 \mid \mathcal{M}_2$ is fair if and only if both its restrictions to states of $\mathcal{M}_1$ and $\mathcal{M}_2$ are fair too.

Some (technical) theorems

Proposition: Parallel composition is associative and commutative (up to isomorphism).

Proof: Easy, but tedious. □

Lemma: For all $\mathcal{M}_1$ and $\mathcal{M}_2$, $\mathcal{M}_1 | \mathcal{M}_2 \preccurlyeq_F \mathcal{M}_1$, and $\mathcal{M}_1 | \mathcal{M}_2 \preccurlyeq_F \mathcal{M}_2$.

Proof: Just define H as $\{((s_1, s_2), s_1) \mid (s_1, s_2) \in S(\mathcal{M}_1 | \mathcal{M}_2)\}$. If $(s_1, s_2) \in I(\mathcal{M}_1 | \mathcal{M}_2)$ then $s_1 \in I_1$. $L(s_1, s_2) = L(s_1) \cup L(s_2)$ with $L(s_1) \cap AP_1 = L(s_2) \cap AP_1 = L(s_1)$. Properties of fair paths end the proof. □

Lemma: If $\mathcal{M}_1 \preccurlyeq_F \mathcal{M}_2$ then for all $\mathcal{M}$, we have $\mathcal{M} | \mathcal{M}_1 \preccurlyeq_F \mathcal{M} | \mathcal{M}_2$.

Proof: Having $H_{1,2}$ simulation of $\mathcal{M}_1$ with $\mathcal{M}_2$ we can define H' as the set $\{((s, s_1), (s, s_2)) \mid H_{1,2}(s_1, s_2)\}$. □

Lemma: For all $\mathcal{M}$, we have $\mathcal{M} \preccurlyeq_F \mathcal{M} | \mathcal{M}$.

Proof: For each state s of $\mathcal{M}$, (s, s) is a state of $\mathcal{M} | \mathcal{M}$. It is easy to show that H defined by $\{(s, (s, s)) \mid s \in S\}$. □

Justifiying Assume-Guarantee Proofs

Example: Proof of soundness of the rule:

$$\frac{\langle true \rangle \mathcal{M}_1 \langle A \rangle \qquad \langle A \rangle \mathcal{M}_2 \langle g \rangle \qquad \langle g \rangle \mathcal{M}_1 \langle f \rangle}{\langle true \rangle \mathcal{M}_1 | \mathcal{M}_2 \langle f \rangle}$$

That is equivalent (using ACTL* satisfiability) to:

$$\frac{\mathcal{M}_1 \preccurlyeq A \qquad A \,|\, \mathcal{M}_2 \vDash g \qquad \mathcal{T}_g \,|\, \mathcal{M}_1 \vDash f}{\mathcal{M}_1 | \mathcal{M}_2 \vDash f}$$

1. $\mathcal{M}_1 | \mathcal{M}_2 \preccurlyeq A \,|\, \mathcal{M}_2$ (hypoth. $\mathcal{M}_1 \preccurlyeq A$ + Theorem)
2. $A \,|\, \mathcal{M}_2 \preccurlyeq \mathcal{T}_g$ (hypoth. $A \,|\, \mathcal{M}_2 \vDash g$ + Theorem)
3. $\mathcal{M}_1 | \mathcal{M}_2 \preccurlyeq \mathcal{T}_g$ (line 1, 2 and transitivity of $\preccurlyeq$)
4. $\mathcal{M}_1 | \mathcal{M}_1 | \mathcal{M}_2 \preccurlyeq \mathcal{T}_g | \mathcal{M}_1$ (line 3 + Theorem)
5. $\mathcal{M}_1 | \mathcal{M}_1 | \mathcal{M}_2 \vDash f$ (lines 4 + hypoth. $\mathcal{T}_g \,|\, \mathcal{M}_1 \vDash f$ + Theor.)
6. $\mathcal{M}_1 \preccurlyeq \mathcal{M}_1 | \mathcal{M}_1 |$ (Theorem)
7. $\mathcal{M}_1 | \mathcal{M}_2 \preccurlyeq \mathcal{M}_1 | \mathcal{M}_1 | \mathcal{M}_2$ (line 6 + Theorem)
8. $\mathcal{M}_1 | \mathcal{M}_2 \vDash f$ (line 5, 7 + Theorem) $\square$

Lesson 9d:

Cone of Influence Reduction

Checking circuits

We consider the problem of checking synchrounous circuits, that can be described by (V is the set of variables):

$$v'_i = f_i(V) \qquad \text{for each } v_i \in V$$

where f_i are boolean functions.

Let us assume that the property of interest depends on a set of variables $V' \subseteq V$. Obviously, variables in V' can depend on the value of variables in V.

Definition: The **cone of influence** of V' is the minimal set of variables $C \subseteq V$ such that:

- $V' \subseteq C$
- if for some $v_i \in C$ its f_i depends on v_j, then $v_j \in C$.

Idea: remove all equations whose left-hand side are variables that do not belong to C.

Checking circuits

Example: Let us consider a counter modulo 8:

$v'_0 = \neg v_0$ $\quad$ $v'_1 = v_0 \oplus v_1$ $\quad$ $v'_2 = (v_0 \wedge v_1) \oplus v_2$

If $V'=\{v_0\}$, then $C=\{v_0\}$ since f_0 depends on v_0 only.

If $V'=\{v_1\}$, then $C=\{v_0, v_1\}$ since f_1 depends on boht v_0 and v_1.

If $V'=\{v_2\}$, then $C=\{v_0, v_1, v_2\}$ since f_2 depends on all variables.

Reduced Model

Let $V = \{v_1, \ldots, v_n\}$ be a set of variables and let $\mathcal{M}=(S, I, R, L)$ be the model of a synchronous circuit.

- $S = \{0, 1\}^n$, the set of valuations of variables in V and $I \subseteq S$.
- $R = \bigwedge_{i \leq n} v'_i = f_i(V)$
- $L(s) = \{v_i \mid s(v_i)=1 \text{ for } 1 \leq i \leq n \}$

Let $C = \{v_1, \ldots, v_k\}$ be the cone of influence of $\mathcal{M}$. The reduced model $\underline{\mathcal{M}} = (\underline{S}, \underline{I}, \underline{R}, \underline{L})$ is defined by:

- $\underline{S} = \{0, 1\}^k$, the set of valuations of variables in C and $I \subseteq S$.
- $\underline{R} = \bigwedge_{i \leq k} v'_i = f_i(V)$
- $\underline{L}(s) = \{v_i \mid \underline{s}(v_i)=1 \text{ for } 1 \leq i \leq k \}$
- $\underline{I}(s) = \{(\underline{d}_1, \ldots, \underline{d}_k) \mid \exists\, (d_1, \ldots, d_n) \in I, d_1= \underline{d}_1, \ldots, d_k= \underline{d}_k \}$

Properties of the Reduced Model

Let $B \subseteq S \times S'$ defined by:

$$((d_1, \dots, d_n), (\underline{d}_1, \dots, \underline{d}_k)) \in B \;\Leftrightarrow\; d_i = \underline{d}_i \text{ for all } 1 \le i \le k$$

Theorem: B is a bisimulation between $\mathcal{M}$ and $\underline{\mathcal{M}}$.

Proof: First, we notice that for each initial state of $\mathcal{M}$, there is a corresponding initial state of $\underline{\mathcal{M}}$.

Let us now consider $(s, \underline{s}) \in B$. Then $d_i = \underline{d}_i$ for all $1 \le i \le k$. Their labelings restricted to C agree and hence $L(s) \cap C = \underline{L}(\underline{s})$.

Let $R(s, t)$ and let $t=(e_1, \dots, e_n)$. The definition of R is such that $v'_i = f_i(V)$, $1 \le i \le n$. By def. of COI, $v'_i = f_i(C)$, $1 \le i \le k$, that is variables in C depends only on C. $B(s, \underline{s})$ implies $\bigwedge_{1 \le i \le k} d_i = \underline{d}_i$ and hence $e_i=f_i(d_1, \dots, d_n)=f_i(\underline{d}_1, \dots, \underline{d}_k)$. If we choose $\underline{t} = (e_1, \dots, e_k)$, then $\underline{R}(\underline{s}, \underline{t})$ and $B(t, \underline{t})$.

The converse is similar, starting from a $\underline{t}$ such that $R(\underline{s}, \underline{t})$ □

Lesson 9

That's all Folks…

…Questions?