Equivalence, Simulation, and Abstraction

(Based on [Clarke et al. 1999])

Yih-Kuen Tsay
(with help from Yu-Fang Chen)

Dept. of Information Management
National Taiwan University

Introduction: The Need to Abstract

- Abstraction is probably the most important technique for alleviating the state-explosion problem.
- Traditionally, finite-state verification (in particular, model checking) methods are geared towards control-oriented systems.
- When nontrivial data manipulations are involved, the complexity of verification is often very high.
- Fortunately, many verification tasks do not require complete information about the system (e.g., one may concern only about whether the value of a variable is odd or even).
- The main idea is to map the set of actual data values to a small set of abstract values.
- An abstract version of the actual system thus obtained is smaller and easier to verify.

Outline

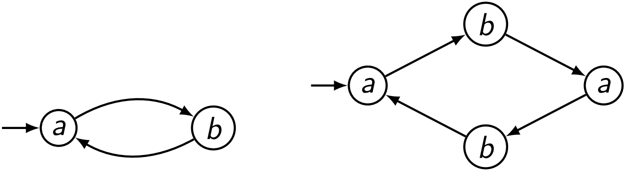
- Bisimulation Equivalence
- Simulation Relation (Preorder)
- Cone of Influence Reduction
- Data Abstraction
  - Approximation
  - Exact Approximation

Bisimulation Equivalence

- Let $M = \langle AP, S, S_0, R, L\rangle$ and $M' = \langle AP, S', S_0', R', L'\rangle$ be two Kripke structures with the same set AP of atomic propositions.
- A relation $B \subseteq S \times S'$ is a bisimulation relation between M and M' iff, for all s and s', $B(s, s')$ implies the following:
  - $L(s) = L'(s')$.
  - For every state s_1 satisfying $R(s, s_1)$, there is s_1' such that $R'(s', s_1')$ and $B(s_1, s_1')$.
  - For every state s_1' satisfying $R'(s', s_1')$, there is s_1 such that $R(s, s_1)$ and $B(s_1, s_1')$.
- Two structures M and M' are bisimulation equivalent, denoted $M \equiv M'$, if there exists a bisimulation relation B between M and M' such that:
  - for every $s_0 \in S_0$ there is an $s_0' \in S_0'$ such that $B(s_0, s_0')$, and
  - for every $s_0' \in S_0'$ there is an $s_0 \in S_0$ such that $B(s_0, s_0')$.

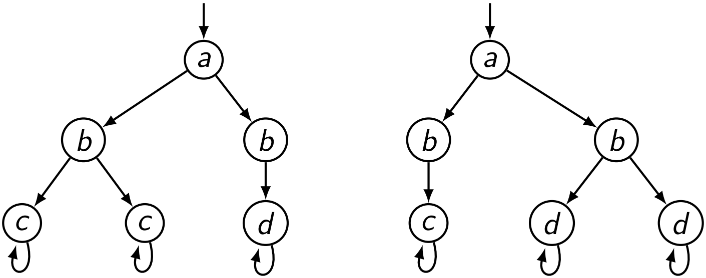
Bisimulation Equivalence (cont.)

- Unwinding preserves bisimulation.



Bisimulation Equivalence (cont.)

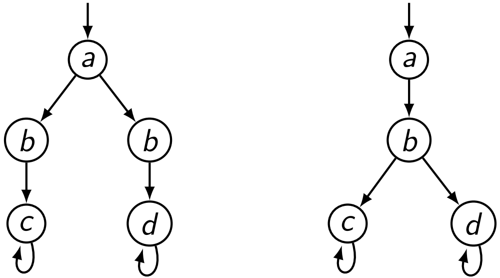
- Duplication preserves bisimulation.



- Two states related by a bisimulation relation is said to be bisimular.

Bisimulation Equivalence (cont.)

- These two structures are not bisimulation equivalent:



Relating CTL* and Bisimulation

Theorem

If $M \equiv M'$ then, for every CTL formula f, $M \models f \Leftrightarrow M' \models f$.*

- This can be proven with the following two lemmas.
- We say that two paths $\pi = s_0 s_1 \ldots$ in M and $\pi' = s'_0 s'_1 \ldots$ in M' correspond iff, for every $i \geq 0$, $B(s_i, s'_i)$.

Lemma

Let s and s' be two states such that $B(s, s')$. Then for every path starting from s there is a corresponding path starting from s' and vice versa.

Relating CTL* and Bisimulation (cont.)

Lemma

Let f be either a state or a path formula. Assume that s and s' are bisimilar states and that π and π' are corresponding paths. Then,

- *if f is a state formula, then $s \models f \Leftrightarrow s' \models f$, and*
- *if f is a path formula, then $\pi \models f \Leftrightarrow \pi' \models f$.*

- Base: $f = p \in AP$. Since $B(s, s')$, $L(s) = L'(s')$. Thus, $s \models p \Leftrightarrow s' \models p$.
- Induction (partial): $f = \mathbf{E} f_1$, a state formula.
  - If $s \models \mathbf{E} f_1$ then there is a path π from s s.t. $\pi \models f_1$.
  - From the previous lemma, there is a corresponding path π' starting from s'.
  - From the induction hypothesis, $\pi \models f_1 \Leftrightarrow \pi' \models f_1$.
  - Therefore, $s' \models \mathbf{E} f_1$.

Simulation Relation (Preorder)

- Let $M = \langle AP, S, S_0, R, L \rangle$ and $M' = \langle AP', S', S_0', R', L' \rangle$ be two structures with $AP \supseteq AP'$.
- A relation $H \subseteq S \times S'$ is a simulation relation between M and M' iff, for all s and s', if $H(s, s')$ then the following conditions hold:
  - $L(s) \cap AP' = L'(s')$.
  - For every state s_1 satisfying $R(s, s_1)$ there is s_1' such that $R'(s', s_1')$ and $H(s_1, s_1')$.
- We say that M' simulates M or M is simulated by M', denoted $M \preceq M'$, if there exists a simulation relation H such that for every $s_0 \in S$ there is an $s_0' \in S_0'$ for which $H(s_0, s_0')$ holds.
- The simulation relation can be shown to be a preorder (i.e., reflexive and transitive).

Relating ACTL* and Simulation

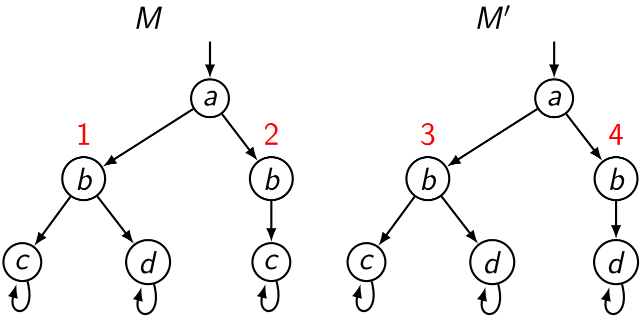
Theorem

Suppose $M \preceq M'$. Then for every ACTL formula f (with atomic propositions in AP'), $M' \models f \Rightarrow M \models f$.*

- Formulae in ACTL* describe properties that are quantified over all possible behaviors of a structure.
- Because every behavior of M is a behavior of M', every formula of ACTL* that is true in M' must also be true in M.
- The theorem does not hold for CTL* formulae.
- In the example on the next slide, M' simulates M; however, $\mathbf{AG}(b \rightarrow \mathbf{EX}\ d)$ is true in M' but false in M.

Compare Bisimulation and Simulation

- Consider these two structures:



- M and M' are not bisimulation equivalent, but each simulates the other.
- $\mathbf{AG}(b \rightarrow \mathbf{EX}\ d)$ is true in M', but false in M.

Cone of Influence Reduction

- The cone of influence reduction attempts to decrease the size of a state transition graph by focusing on the variables of the system that are referred to in the desired property specification.
- The reduction is obtained by eliminating variables that do not influence the variables in the specification.
- In this way, the checked properties are preserved, but the size of the model that needs to be verified is smaller.

Cone of Influence Reduction (cont.)

- Let $V = \{v_1, \ldots, v_n\}$ be the set of Boolean variables of a given structure $M = (S, R, S_0, L)$.
- The transition relation R is specified by $\bigwedge_{i=1}^{n}[v_i' = f_i(V)]$.
- Suppose we are given a set of variables $V' \subseteq V$ that are of interest w.r.t. the property specification.
- The cone of influence C of V' is the minimal set of variables such that
  - $V' \subseteq C$
  - if for some $v_l \in C$ its f_l depends on v_j, then $v_j \in C$.
- We construct a new (reduced) structure by removing all the clauses in R whose left hand side variables do not appear in C and using C to construct states.

An Example

- Let $V = \{v_0, v_1, v_2\}$ and $M = (S, R, S_0, L)$ a structure over V, where $R = (v_0' = \neg v_0) \wedge (v_1' = v_0 \oplus v_1) \wedge (v_2' = v_1 \oplus v_2)$.
  - If $V' = \{v_0\}$ then $C = \{v_0\}$, since $f_0 = \neg v_0$ does not depend on any variable other than v_0.
  - If $V' = \{v_1\}$ then $C = \{v_0, v_1\}$, since $f_1 = v_0 \oplus v_1$ depends on both variables.
  - If $V' = \{v_2\}$ then $C = \{v_0, v_1, v_2\}$, since $f_2 = v_1 \oplus v_2$ depends on v_1, v_2 and $f_1 = v_0 \oplus v_1$ depends on v_0, v_1 (because v_1 is in C).

The Reduced Model

- Let $V = \{v_1, \ldots, v_n\}$.
- $M = (S, R, S_0, L)$ is a structure over V:
  - $S = \{0,1\}^n$ is the set of all valuations of V.
  - $R = \bigwedge_{i=1}^{n} [v_i' = f_i(V)]$.
  - $L(s) = \{v_i \mid s(v_i) = 1 \text{ for } 1 \leq i \leq n\}$.
  - $S_0 \subseteq S$.
- The reduced model $\widehat{M} = (\widehat{S}, \widehat{R}, \widehat{S_0}, \widehat{L})$ w.r.t. $C = \{v_1, \ldots, v_k\}$ for some $k \leq n$:
  - $\widehat{S} = \{0,1\}^k$ is the set of all valuations of C.
  - $\widehat{R} = \bigwedge_{i=1}^{k} [v_i' = f_i(V)]$.
  - $\widehat{L}(\widehat{s}) = \{v_i \mid \widehat{s}(v_i) = 1 \text{ for } 1 \leq i \leq k\}$.
  - $\widehat{S_0} = \{(\widehat{d_1}, \ldots, \widehat{d_k}) \mid$ there is a state $(d_1, \ldots, d_n) \in S_0$ s.t. $\widehat{d_1} = d_1 \wedge \cdots \wedge \widehat{d_k} = d_k\}$.

Bisimulation Equivalence between Models

- Let $B \subseteq S \times \widehat{S}$ be the relation defined as follows: $((d_1, \ldots, d_n), (\widehat{d_1}, \ldots, \widehat{d_k})) \in B \Leftrightarrow d_i = \widehat{d_i}$ for all $1 \leq i \leq k$.
- We show that B is a bisimulation relation between M and $\widehat{M}$ ($M \equiv \widehat{M}$).
  - For every $s_0 \in S$ there is a corresponding $\widehat{s_0} \in \widehat{S}$ and *vice versa*.
  - Let $s = (d_1, \ldots, d_n)$ and $\widehat{s} = (\widehat{d_1}, \ldots, \widehat{d_k})$ s.t. $(s, \widehat{s}) \in B$.
  - $L(s) \cap C = \widehat{L}(\widehat{s})$.
  - If $s \rightarrow t$ is a transition in M, then there is a transition $\widehat{s} \rightarrow \widehat{t}$ in $\widehat{M}$ s.t. $(t, \widehat{t}) \in B$.
  - If $\widehat{s} \rightarrow \widehat{t}$ is a transition in $\widehat{M}$, then there is a transition $s \rightarrow t$ in M s.t. $(t, \widehat{t}) \in B$.

Bisimulation Equiv. between Models (cont.)

- Let $s \longrightarrow t$ be a transition in M.
- There is a transition $\widehat{s} \longrightarrow \widehat{t}$ in $\widehat{M}$ s.t. $(t, \widehat{t}) \in B$.
  1. For $1 \leq i \leq n, v_i' = f_i(V)$. (Transition relation)
  2. For $1 \leq i \leq k$, v_i depends only on variables in C, hence $v_i' = f_i(C)$. (Definition of C)
  3. $(s, \widehat{s}) \in B$ implies $\bigwedge_{i=1}^{k}(d_i = \widehat{d_i})$. (Bisimilar states)
  4. Let $t = (e_1, \ldots, e_k)$. For every $1 \leq i \leq k$, $e_i = f_i(d_1, \ldots, d_k) = f_i(\widehat{d_1}, \ldots, \widehat{d_k})$. (From 2,3)
  5. If we choose $\widehat{t} = (e_1, \ldots, e_k)$, then $\widehat{s} \longrightarrow \widehat{t}$ and $(t, \widehat{t}) \in B$ as required.

Theorem

Let f be a CTL formula with atomic propositions in C. Then $M \models f \Leftrightarrow \widehat{M} \models f$.*

Data Abstraction

- Data abstraction involves finding a mapping between the actual data values in the system and a small set of abstract data values.
- By extending this mapping to states and transitions, it is possible to obtain an abstract system that simulates the original system and is usually much smaller.
- **Example:** Assume we are interested in expressing a property involving the sign of x. We create a domain A_x of abstract values for x, with $\{a_0, a_+, a_-\}$, and define a mapping h_x from D_x to A_x as follows:

$$h_x(d) = \begin{cases} a_0 & \text{if } d = 0 \\ a_+ & \text{if } d > 0 \\ a_- & \text{if } d < 0 \end{cases}$$

Data Abstraction (cont.)

IM NTU

- The abstract value of x can be expressed by three APs: "$\widehat{x} = a_0$", "$\widehat{x} = a_+$", and "$\widehat{x} = a_-$".
- All states labelled with "$\widehat{x} = a_+$" will be collapsed into one state; that is, all states where $x > 0$ are merged into one.
- If there is a transition between, e.g., states corresponding to $x = 0$ and $x = 5$, there must be a transition between states labelled $\widehat{x} = a_0$ and $\widehat{x} = a_+$.

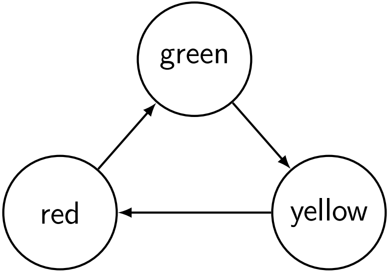
The Reduced Model by Abstraction

- Let h be a mapping form D to an abstract domain A.
- The mapping determines a set of abstract atomic propositions AP.
- We now obtain a new structure $M = (S, R, S_0, L)$ that is identical to the original one expect that L labels each state with a subset of AP.
- The structure M can be collapsed into a reduced structure M_r over AP defined as follows:
  - $S_r = \{L(s) \mid s \in S\}$.
  - $R_r(s_r, t_r)$ iff there exist s and t s.t. $s_r = L(s)$, $t_r = L(t)$, and $R(s, t)$.
  - $s_r \in S_0^r$ iff there exists an s s.t. $s_r = L(s)$ and $s \in S_0$.
  - $L_r(s_r) = s_r$ (each s_r is a set of atomic propositions).

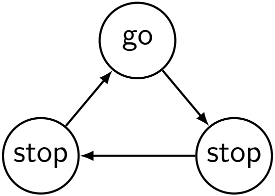
The Reduced Model by Abstraction (cont.)

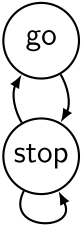
- M_r simulates the structure M.
- Every path that can be generated by M can also be generated by M_r.
- Whatever ACTL* properties we can prove about M_r will be also hold in M.
- Note that using this technique it is only possible to determine whether formulae over AP are true in M.

The Reduced Model by Abstraction (cont.)



$h(red) = stop$; $h(yellow) = stop$; $h(green) = go$.





Approximation

- The construction of M_r, as described, requires the construction of M.
- When M is too large, we use an implicit representation in terms of $\mathcal{S}_0$ and $\mathcal{R}$.
- In many cases, M_r may still be too large to construct exactly.
- To further reduce the state space, an approximation M_a that simulates M_r is constructed.
- The goal here is to have M_a sufficiently close to M_r so that it is still possible to verify interesting properties.

The Model in FOL

- We use the first order formulae $\mathcal{S}_0$ and $\mathcal{R}$ to define the Kripke structure $M = (S, R, S_0, L)$ with state set $S = D \times \cdots \times D$.
- S_0 is the set of valuations satisfying $\mathcal{S}_0$.
- Similarly, R is derived from $\mathcal{R}$.
- L is defined over abstract atomic propositions, e.g., $\{\text{“}\widehat{x_1} = a_1\text{”}, \text{“}\widehat{x_2} = a_2\text{”}, \ldots, \text{“}\widehat{x_n} = a_n\text{”}\}$.

The Reduced Model in FOL

- To produce M_r over the abstract state set $A \times \cdots \times A$, we construct formulae over $\widehat{x_1}, \ldots, \widehat{x_n}$ and $\widehat{x_1}', \ldots, \widehat{x_n}'$ that will represent the initial states and transition relation of M_r.
- $\widehat{\mathcal{S}_0} = \exists x_1 \cdots \exists x_n (h(x_1) = \widehat{x_1} \wedge \cdots \wedge h(x_n) = \widehat{x_n} \wedge \mathcal{S}_0(x_1, \ldots, x_n))$.
- $\widehat{\mathcal{R}} = \exists x_1 \cdots \exists x_n \exists x_1' \cdots \exists x_n' (h(x_1) = \widehat{x_1} \wedge \cdots \wedge h(x_n) = \widehat{x_n} \wedge$ $h(x_1') = \widehat{x_1}' \wedge \cdots \wedge h(x_n') = \widehat{x_n}' \wedge \mathcal{R}(x_1, \ldots, x_n, x_1', \ldots, x_n'))$.
- For conciseness, this existential abstraction operation is denoted by $[\cdot]$.
- If ϕ depends on the free variables $x_1, \ldots, x_m$, then define $[\phi](\widehat{x_1}, \ldots, \widehat{x_m}) =$ $\exists x_1 \cdots \exists x_m (h(x_1) = \widehat{x_1} \wedge \cdots \wedge h(x_m) = \widehat{x_m} \wedge \phi(x_1, \ldots, x_m))$
- So, $\widehat{\mathcal{S}_0} = [\mathcal{S}_0]$ and $\widehat{\mathcal{R}} = [\mathcal{R}]$.

Computing Approximation

- Ideally, we would like to extract S_0^r and R_r from $[\mathcal{S}_0]$ and $[\mathcal{R}]$. However, this is often computationally expensive.
- To circumvent this difficulty, we define a transformation $\mathcal{A}$ on formula ϕ.
- The idea is to simplify the formulae to which $[\cdot]$ is applied ("pushing the abstractions inward").
- This will make it easier to extract the Kripke structure from the formulae.

Computing Approximation (cont.)

- Assume ϕ is given in the negation normal form.
- The approximation $\mathcal{A}(\phi)$ of $[\phi]$ is computed as follows.
  - $\mathcal{A}(P(x_1, \ldots, x_m)) = [P](\widehat{x_1}, \ldots, \widehat{x_m})$ if P is a primitive relation.
  - Similarly, $\mathcal{A}(\neg P(x_1, \ldots, x_m)) = [\neg P](\widehat{x_1}, \ldots, \widehat{x_m})$.
  - $\mathcal{A}(\phi_1 \wedge \phi_2) = \mathcal{A}(\phi_1) \wedge \mathcal{A}(\phi_2)$.
  - $\mathcal{A}(\phi_1 \vee \phi_2) = \mathcal{A}(\phi_1) \vee \mathcal{A}(\phi_2)$.
  - $\mathcal{A}(\exists x \phi) = \exists \widehat{x} \mathcal{A}(\phi)$.
  - $\mathcal{A}(\forall x \phi) = \forall \widehat{x} \mathcal{A}(\phi)$.

Computing Approximation (cont.)

- The approximation Kripke structure $M_a = (S_a, s_0^a, R_a, L_a)$ can be derived from $\mathcal{A}(\mathcal{S}_0)$ and $\mathcal{A}(\mathcal{R})$.
- Let $s_a = (a_1, \ldots, a_n) \in S_a$. Then $L_a(s_a) = \{\text{“}\widehat{x_1} = a_1\text{”}, \text{“}\widehat{x_2} = a_2\text{”}, \ldots, \text{“}\widehat{x_n} = a_n\text{”}\}$.
- Note that $s = (d_1, \ldots, d_n) \in S$ and s_a will be labeled identically if for all i, $h(d_i) = a_i$.

Computing Approximation (cont.)

- The price for the approximation is that it may be necessary to add extra initial states and transitions to the corresponding structure.
- This is because $[\phi]$ implies $\mathcal{A}(\phi)$, but the converse may not be true.
- In particular, $[\mathcal{S}_0] \rightarrow \mathcal{A}(\mathcal{S}_0)$ and $[\mathcal{R}] \rightarrow \mathcal{A}(\mathcal{R})$.

Theorem

$[\phi]$ *implies* $\mathcal{A}(\phi)$.

Computing Approximation (cont.)

IM NTU

- The proof is by induction on the structure of ϕ.
- We show the case $\phi(x_1, \ldots, x_m) = \forall x \phi_1$ only.

$$
\begin{array}{ll}
 & [\forall x \phi_1] \\
= & \exists x_1 \cdots \exists x_m (\bigwedge h(x_i) = \widehat{x_i} \wedge \forall x \phi_1(x, x_1, \ldots, x_m)) \\
= & \exists x_1 \cdots \exists x_m \forall x (\bigwedge h(x_i) = \widehat{x_i} \wedge \phi_1(x, x_1, \ldots, x_m)) \\
\rightarrow & \forall x \exists x_1 \cdots \exists x_m (\bigwedge h(x_i) = \widehat{x_i} \wedge \phi_1(x, x_1, \ldots, x_m)) \\
\rightarrow & \forall \widehat{x} \exists x [\exists x_1 \cdots \exists x_m (h(x) = \widehat{x} \wedge \bigwedge h(x_i) = \widehat{x_i} \wedge \phi_1(x, x_1, \ldots, x_m)) \\
= & \forall \widehat{x} [\phi_1] \\
\rightarrow & \forall \widehat{x} \mathcal{A}(\phi_1) \\
= & \mathcal{A}(\forall x \phi_1)
\end{array}
$$

Computing Approximation (cont.)

IM NTU

Theorem

$M \preceq M_a$.

Proof.

1. Because the approximation M_a only adds extra initial states and transitions to the reduced model M_r, all paths in the M_r are reserved. So, $M_r \preceq M_a$.
2. Since $M \preceq M_r$ and $\preceq$ is transitive, $M \preceq M_a$.

□

Corollary

Every ACTL formula that holds in M_a also holds in M.*

Exact Approximation

IM NTU

- We consider some additional conditions that allow us to show that M is bisimulation equivalent to M_a.
- Each abstraction mapping h_x for variable x induces an equivalence relation $\sim_x$:
  - Let d_1 and d_2 be in D_x.
  - $d_1 \sim_x d_2$ iff $h_x(d_1) = h_x(d_2)$.
- The equivalence relation $\sim_{x_i}$ is a congruence with respect to a primitive relation P iff

$$\forall d_1 \cdots \forall d_m \forall e_1 \cdots \forall e_m$$
$$(\bigwedge_{i=1}^{m} d_i \sim_{x_i} e_i \rightarrow (P(d_1, \ldots, d_m) \Leftrightarrow P(e_1, \ldots, e_m)))$$

Exact Approximation (cont.)

Theorem

If the $\sim_{x_i}$ are congruences with respect to the primitive relations and ϕ is a formula defined over these relations, then $[\phi] \Leftrightarrow \mathcal{A}(\phi)$, i.e., $M_a \equiv M_r$.

Theorem

If $\sim_{x_i}$ are congruences with respect to the primitive relations, then $M \equiv M_a$.