Algorithms for Model Checking (2IW55)

Lecture 9

Data Abstraction

Chapter 13

Tim Willemse
(timw@win.tue.nl)
http://www.win.tue.nl/∼timw
HG 6.81

Outline

1. Recap
2. Data Abstraction
3. Conclusion

Recap

- We have seen:
  - If $M_1 \equiv M_2$ (M_1 and M_2 are bisimilar), then M_1 and M_2 satisfy the same CTL* formulae
  - If $M_1 \sqsubseteq M_2$ (M_2 simulates M_1), then all ACTL* formulae that are valid for M_2 hold for M_1 as well
- Consider a specification S with a very large state space M_1
- The question is how to compute an M_2 such that:
  - $M_1 \sqsubseteq M_2$ (i.e., useful for ACTL* formulae)
  - M_2 is smaller than M_1
  - generation of M_1 is not required
- Method: data abstraction

Recap

Recall that the system specification is given by formulae from first-order predicate logic.

A specification is a tuple $\langle V, D, \mathcal{S}, \mathcal{R} \rangle$, where:

- V is a set of variables $v_1, \ldots, v_n$
- D is the domain of these variables
- The formula $\mathcal{S}_0(V)$ represents the initial states
- Let V' be a copy of the variables: $v'_1, \ldots, v'_n$
- The formula $\mathcal{R}(V, V')$ represents the transition relation:
    - V denotes the value of variables before the transition
    - V' denotes the value of variables after the transition

Conventions:

- $\phi(V)$ denotes a first-order formula with all free variables among V.
- Given a valuation $\alpha : V \to D$, the semantics of ϕ under α is written as $[\![\phi]\!]\alpha \in \{\mathsf{true}, \mathsf{false}\}$.

Recap

Semantics of a first-order predicate logic specification.

Given a specification $\langle V, D, \mathcal{S}_0, \mathcal{R} \rangle$, we get the underlying Kripke Structure (alias state space) $\langle AP, S, S_0, R, L \rangle$ as follows:

- The set of states S of the Kripke Structure will be the set of valuations $\alpha : V \to D$
- The atomic propositions *AP* will be of the form $v = d$, where $v \in V$ and $d \in D$
- The set of initial states S_0 will be the valuations α, such that $[\![\mathcal{S}_0]\!]\alpha$ is true
- Let $\alpha, \beta : V \to D$ be valuations. Define $\beta'(v_i') = \beta(v_i)$ for all $v_i \in V$. There is a transition $R(\alpha, \beta)$ if $[\![\mathcal{R}]\!](\alpha \cup \beta')$ is true
- Finally, the set of labels of state α is $L(\alpha) := \{v_1 = \alpha(v_1), \ldots, v_n = \alpha(v_n)\}$

Outline

1. Recap
2. Data Abstraction
3. Conclusion

Data Abstraction

Idea of Data Abstraction:

- State variables range over some domain D
- D is now called the concrete domain
- A data abstraction consists of:
  - An abstract domain A
  - A surjective mapping $h : D \to A$
- Example:
  - Concrete domain: natural numbers $\mathcal{N}$
  - Abstract domain: $\{even, odd\}$
  - $h(2n) = even$, $h(2n+1) = odd$
- Abstract interpretation: evaluating an expression directly in the abstract domain
- For this, all concrete operations must be mimicked by abstract operations
- In the example: $\widehat{m+n}$ can be computed directly as $\widehat{m}\widehat{+}\widehat{n}$ if we set:

$even \widehat{+} even = even$	$odd \widehat{+} odd = even$
$even \widehat{+} odd = odd$	$odd \widehat{+} even = odd$

Data Abstraction

Let Kripke Structure $\langle AP, S, S_0, R, L\rangle$ be given, where $S = [V \to D]$ (i.e., the set of functions $V \to D$).

Let abstract domain A with $h : D \to A$ be given:

- The idea will be:
  - replace concrete data by abstract values
  - collapse states with the same abstract value
- The price to be paid is that we can now only express properties on abstract values.
  - Let $\widehat{v}$ denote the abstract version of variable v, ranging over A
  - Change the set of atomic propositions: $\widehat{AP} := \{\widehat{v} = a \mid v \in V, a \in A\}$
  - Change the labelling: $L'(s) := \{\dots, \widehat{v_i} = h(s), \dots\}$
- We now compare the concrete $M' = \langle \widehat{AP}, S, S_0, R, L'\rangle$ with the abstract $M_r = \langle \widehat{AP}, S_r, S_0^r, R_r, L_r\rangle$ defined next.

Data Abstraction

Given a concrete $M' = \langle \widehat{AP}, S, S_0, R, L' \rangle$, we define abstract $M_r = \langle \widehat{AP}, S_r, S_0^r, R_r, L_r \rangle$:

- We want to collapse (identify) states with the same labels, so we define:

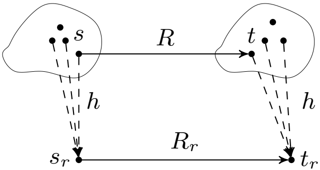
$$S_r := \{L'(s) \mid s \in S\} \ldots\ldots\ldots\ldots\ldots\ldots (= [\widehat{V} \to A])$$

- Of course, $L_r(s_r) := s_r$
- M_r must simulate M', so it should reflect all initial states:

$$S_0^r(s_r) \text{ iff } \exists s \in S_0 : \; L'(s) = s_r$$

- M_r must simulate M', so it should reflect all transitions. So define $R_r(s_r, t_r)$ iff

$$\exists s, t \in S : \; s_r = L'(s) \wedge t_r = L'(t) \wedge R(s, t)$$



Data Abstraction

Problem: constructing M_r requires M'.
Given specification $\langle V, D, \mathcal{S}_0, \mathcal{R} \rangle$, and data abstraction (A, h), we want a specification for M_r.

- Notation: write $[\phi](\widehat{V})$ to abbreviate the ideal abstraction of $\phi(V)$:

$$
\begin{aligned}
&[\phi](\widehat{v_1}, \ldots, \widehat{v_n}) := \\
&\qquad \exists v_1, \ldots, v_n : \; h(v_1) = \widehat{v_1} \wedge \cdots \wedge h(v_n) = \widehat{v_n} \wedge \phi(v_1, \ldots, v_n)
\end{aligned}
$$

- Then the ideal abstract specification is defined as $\langle \widehat{V}, A, [\mathcal{S}_0](\widehat{V}), [\mathcal{R}](\widehat{V}, \widehat{V'}) \rangle$
- Crucial property achieved by this construction:

$$
\begin{array}{ll}
 & \phi(s,t) \\
\Leftrightarrow & h(s) = h(s) \wedge h(t) = h(t) \wedge \phi(s,t) \\
\Rightarrow & \exists v_1, v_2 : \; h(v_1) = h(s) \wedge h(v_2) = h(t) \wedge \phi(v_1, v_2) \\
\Leftrightarrow & [\phi](h(s), h(t))
\end{array}
$$

- However, due to all the existential quantifications, this abstraction is costly to generate.

Data Abstraction

- Idea: generate a cheaper approximation by pushing the quantifications inside
- Assume that $\mathcal{S}_0$ and $\mathcal{R}$ are in positive normal form. We define $\mathcal{A}(\phi)$, the abstract interpretation of ϕ as follows:
  - $\mathcal{A}(P(x_1,\ldots,x_n)) = [P](\widehat{x_1},\ldots,\widehat{x_n})$
  - $\mathcal{A}(\neg P(x_1,\ldots,x_n)) = [\neg P](\widehat{x_1},\ldots,\widehat{x_n})$
  - $\mathcal{A}(\phi_1 \wedge \phi_2) = \mathcal{A}(\phi_1) \wedge \mathcal{A}(\phi_2)$
  - $\mathcal{A}(\phi_1 \vee \phi_2) = \mathcal{A}(\phi_1) \vee \mathcal{A}(\phi_2)$
  - $\mathcal{A}(\exists x.\ \phi) = \exists \widehat{x}.\mathcal{A}(\phi)$
  - $\mathcal{A}(\forall x.\ \phi) = \forall \widehat{x}.\mathcal{A}(\phi)$
- By induction on ϕ, one can show that $[\phi] \implies \mathcal{A}(\phi)$ (Here we use that h is surjective!)
- Note: not necessarily $\mathcal{A}(\phi) \implies [\phi]$
- So, given the specification $\langle V, D, \mathcal{S}_0, \mathcal{R}\rangle$, we define its abstract version to be $\langle \widehat{V}, A, \mathcal{A}(\mathcal{S}_0), \mathcal{A}(\mathcal{R})\rangle$

Data Abstraction

Correctness of abstract interpretation

- Let M' be the Kripke Structure over $\widehat{AP}$, induced by specification $\langle V, D, \mathcal{S}_0, \mathcal{R}\rangle$
- Let M_α be the Kripke Structure from the abstract specification $\langle \widehat{V}, A, \mathcal{A}(\mathcal{S}_0), \mathcal{A}(\mathcal{R})\rangle$
- Claim: $M' \sqsubseteq M_\alpha$
- Proof: Let $s = \{v_1 = d_1, \ldots, v_n = d_n\}$ and $s_\alpha = \{\widehat{v_1} = a_1, \ldots, \widehat{v_n} = a_n\}$
  The following H is a simulation relation:

  $$H(s, s_\alpha) = \forall 1 \leq i \leq n : \ h(d_i) = a_i$$

  - By definition, the (abstract) labels coincide
  - Let $s \rightarrow t$ in M'. Then $\mathcal{R}(s,t)$ holds. Hence, $[\mathcal{R}](h(s), h(t))$ holds, and therefore $\mathcal{A}(\mathcal{R}(h(s), h(t)))$ holds. So, $h(s) \rightarrow h(t)$ is a transition in M_α.
  - Note that $H(s, h(s))$ holds for all s.
  - Similarly, if $s \in \mathcal{S}_0$, then $s \in \mathcal{A}(\mathcal{S}_0)$.
- Hence, abstract interpretation is sound for ACTL*.

Outline

1. Recap
2. Data Abstraction
3. Conclusion

Conclusion

- Data abstraction transforms a specification S (with KS M) to a more abstract specification S' (with KS M')
- By a careful construction, we know that $M \sqsubseteq M'$
- For ACTL* properties, $M' \models \phi$ implies $M \models \phi$, so generation of M can be avoided
- This technique is important to reduce big state spaces, and essential when state spaces become infinite
- Hence for applying model checking to software systems, data abstraction is a crucial technique
- However, finding the right abstraction is a creative step, as hard as finding e.g., correct invariants. So this approach to verification is not completely mechanical
- In practice, a bunch of standard abstractions are used. Their power is quite limited on complicated programs