Branching Time Model Checking and Abstraction

Helmut Veith
Branching Time Logic
Kripke Structures

Kripke structures
K = (States, Transition Relation, Initial States, Labelling) = (S,R,I,L)

Specifications
Temporal logic, e.g. CTL (branching time) and LTL (linear time)

Model Relation
K |= f Specification f holds true in model K
Branching Time Logic

„unwinding“
CTL - Computation Tree Logic

$\text{EF } g$  \quad \text{“}g\text{ will possibly become true}\text{”}$
CTL - Computation Tree Logic

\[ \text{AF } g \quad \text{“g will necessarily become true”} \]
CTL - Computation Tree Logic

AG g  “g is an invariant”
CTL - Computation Tree Logic

EG g  "g is a potential invariant"
Computation Tree Logic

Family of Temporal Logics

<table>
<thead>
<tr>
<th>ACTL</th>
<th>AX, AG, AF, AU</th>
</tr>
</thead>
<tbody>
<tr>
<td>ECTL</td>
<td>EX, EG, EF, EU</td>
</tr>
<tr>
<td>CTL</td>
<td>ACTL &amp; ECTL</td>
</tr>
<tr>
<td>CTL*</td>
<td>AXX, AGX, EXF, ...</td>
</tr>
</tbody>
</table>
Simulation and Bisimulation
Simulation Game

Combinatorial two player game between Spoiler and Duplicator.

Spoiler wins if Duplicator gets stuck.
Duplicator wins if game continues forever.

Example of a Combinatorial Game.

→ Ehrenfeucht-Fraïssé Games, Pebble Games, Parity Games etc.
Simulation

I can be simulated by S step by step.

“S simulates I”: \( I \preceq S \)
The simulation preorder [Milner]

Given two models $M_1 = (S_1, I_1, R_1, L_1)$, $M_2 = (S_2, I_2, R_2, L_2)$

$H \subseteq S_1 \times S_2$ is a simulation iff
for every $(s_1, s_2) \in H$:

- $s_1$ and $s_2$ satisfy the same propositions
- For every successor $t_1$ of $s_1$ there is a successor $t_2$ of $s_2$ such that $(t_1, t_2) \in H$

Notation: $s_1 \leq s_2$
The simulation preorder [Milner]

Given two models \( M_1 = (S_1, I_1, R_1, L_1) \), \( M_2 = (S_2, I_2, R_2, L_2) \)

\( H \subseteq S_1 \times S_2 \) is a simulation iff

for every \((s_1, s_2) \in H\) :

• \( \forall p \in \text{AP} : \; s_2 \models p \Rightarrow s_1 \models p \)
  \[ s_2 \models \neg p \Rightarrow s_1 \models \neg p \]

• \( \forall t_1 [ (s_1, t_1) \in R_1 \Rightarrow \exists t_2 [ (s_2, t_2) \in R_2 \land (t_1, t_2) \in H ] ] \)

Notation: \( s_1 \leq s_2 \)
Simulation preorder (cont.)

\[ H \subseteq S_1 \times S_2 \text{ is a simulation from } M_1 \text{ to } M_2 \text{ iff } \]
\[ H \text{ is a simulation and for every } s_1 \in I_1 \text{ there is } s_2 \in I_2 \text{ s.t. } (s_1, s_2) \in H \]

Notation: \( M_1 \preceq M_2 \)
Bisimulation relation [Park]

For models $M_1$ and $M_2$, $H \subseteq S_1 \times S_2$ is a bisimulation iff for every $(s_1, s_2) \in H$:

- $\forall p \in AP : p \in L(s_2) \iff p \in L(s_1)$
- $\forall t_1 [ (s_1, t_1) \in R_1 \Rightarrow \exists t_2 [ (s_2, t_2) \in R_2 \land (t_1, t_2) \in H ] ]$
- $\forall t_2 [ (s_2, t_2) \in R_2 \Rightarrow \exists t_1 [ (s_1, t_1) \in R_1 \land (t_1, t_2) \in H ] ]$

Notation: $s_1 \equiv s_2$
Bisimulation relation (cont.)

$H \subseteq S_1 \times S_2$ is a **Bisimulation** between $M_1$ and $M_2$

iff $H$ is a bisimulation and

for every $s_1 \in I_1$ there is $s_2 \in I_2$ s.t. $(s_1, s_2) \in H$

and

for every $s_2 \in I_2$ there is $s_1 \in I_1$ s.t. $(s_1, s_2) \in H$

Notation: $M_1 \equiv M_2$
Bisimulation equivalence

\[ \mathcal{M}_1 \equiv \mathcal{M}_2 \]

\[ H = \{ (1,1'), (2,4'), (4,2'), (3,5'), (3,6'), (5,3'), (6,3') \} \]
Simulation preorder

$M_1 \leq M_2$

$M_1$

$M_2$

wait

coin

coke

coin

pepsi

wait

coin

coke

pepsi
$M_1 \leq M_2$
\[ M_1 \leq M_2 \quad \text{and} \quad M_1 \geq M_2 \quad \text{but not} \quad M_1 \equiv M_2 \]
(bi)simulation and logic preservation

Theorem

If $M_1 \equiv M_2$ then for every $\text{CTL}^*$ formula $\varphi$, $M_1 \models \varphi \iff M_2 \models \varphi$

If $M_2 \succeq M_1$ then for every $\text{ACTL}^*$ formula $\varphi$, $M_2 \models \varphi \implies M_1 \models \varphi$
Simulation Relation

If M has partial behavior of N, we say that

“N simulates M”: \( M \leq N \)

Let \( f \) be an ACTL specification. If \( M \leq N \) and \( N \models f \) then \( M \models f \).
Simulation and Abstraction

If M has partial behavior of N, we say that "N simulates M": \( M \leq N \)

M

\[
\begin{array}{ccc}
  a & \rightarrow & b \\
  b & \rightarrow & c \\
  c & \rightarrow & d \\
\end{array}
\]

M |= f

N

\[
\begin{array}{ccc}
  a & \rightarrow & b \\
  b & \rightarrow & c \\
  c & \rightarrow & d \\
\end{array}
\]

N |= f
Abstraction
Abstraction
Data Abstraction

Given a program $P$ with variables $x_1, \ldots, x_n$, each over domain $D$, the \textbf{concrete model} of $P$ is defined over states $(d_1, \ldots, d_n) \in D \times \ldots \times D$

Choosing

- abstract domain $A$
- Abstraction mapping (surjection) $h: D \rightarrow A$

we get an \textbf{abstract model} over abstract states $(a_1, \ldots, a_n) \in A \times \ldots \times A$
Example

Given a program $P$ with variable $x$ over the integers

**Abstraction 1:**

$A_1 = \{ a_-, a_0, a_+ \}$

$h_1(d) = \begin{cases} 
a_+ & \text{if } d>0 \\
a_0 & \text{if } d=0 \\
a_- & \text{if } d<0 
\end{cases}$

**Abstraction 2:**

$A_2 = \{ a_{\text{even}}, a_{\text{odd}} \}$

$h_2(d) = \begin{cases} 
\text{if even}(d) & \text{then } a_{\text{even}} \text{ else } a_{\text{odd}} 
\end{cases}$
Reduced abstract model
Existential abstraction

Given $M, A, h : D \rightarrow A$
the reduced model $M_r = (S_r, I_r, R_r, L_r)$ is

$S_r = A \times \ldots \times A$
$s_r \in I_r \iff \exists s \in I : h(s) = s_r$
$(s_r, t_r) \in R_r \iff$

$\exists s, t [h(s) = s_r \land h(t) = t_r \land (s, t) \in R]$ 

For $s_r = (a_1, \ldots, a_n)$, $L_r(s_r) = \{ (x_i^A = a_i) \mid i = 1, \ldots, n \}$
Existential Abstraction
Preservation

Theorem:
\( M_r \geq M \) by the simulation preorder

Corollary:
For every ACTL* formula \( \varphi \):
If \( M_r \models \varphi \) then \( M \models \varphi \)
Traffic Light Example

Property:
\[ \varphi = \text{AG AF} \neg \text{(state=red)} \]

Abstraction function \( h \) maps green, yellow to go.

\[ M \models \varphi \iff M_h \models \varphi \]
If the abstract model invalidates a specification, the actual model may still satisfy the specification.

Property: \( \varphi = \text{AG AF (state=red)} \)

- \( M \models \varphi \) but \( M_h \not\models \varphi \)

Spurious Counterexample: \( \langle \text{red,go,go, ...} \rangle \)
CEGAR Methodology

- **M and \( \varphi \)**
- **generate initial abstraction**
- **model check**
  - \( M_h \models \varphi \)
  - \( M_h \not\models \varphi \)
- **refinement**
- **generate counterexample** \( T_h \)
- **check spurious counterexample**
  - \( T_h \) is spurious
  - \( T_h \) is not spurious

stop
CEGAR (Counterexample-Guided Abstraction Refinement)

Adaptive Strategy

Counterexample-Guided Abstraction Refinement
Clarke, Grumberg, Jha, Lu, Veith’00
CEGAR (Counterexample-Guided Abstraction Refinement)

Adaptive Strategy

Counterexample-Guided Abstraction Refinement
Clarke, Grumberg, Jha, Lu, Veith’00
Counterexample-Guided Abstraction Refinement
Clarke, Grumberg, Jha, Lu, Veith’00
Software Model Checking

CEGAR + Predicate Abstraction
Integration of Theorem Proving / Decision Procedures / SMT
SIGSOFT Distinguished Paper Award (ICSE 2003)