Simulation Reduction as Constraint

R. Gentilini, C. Piazza, and A. Policriti

University of Udine

Workshop SAVE 2001

Plan

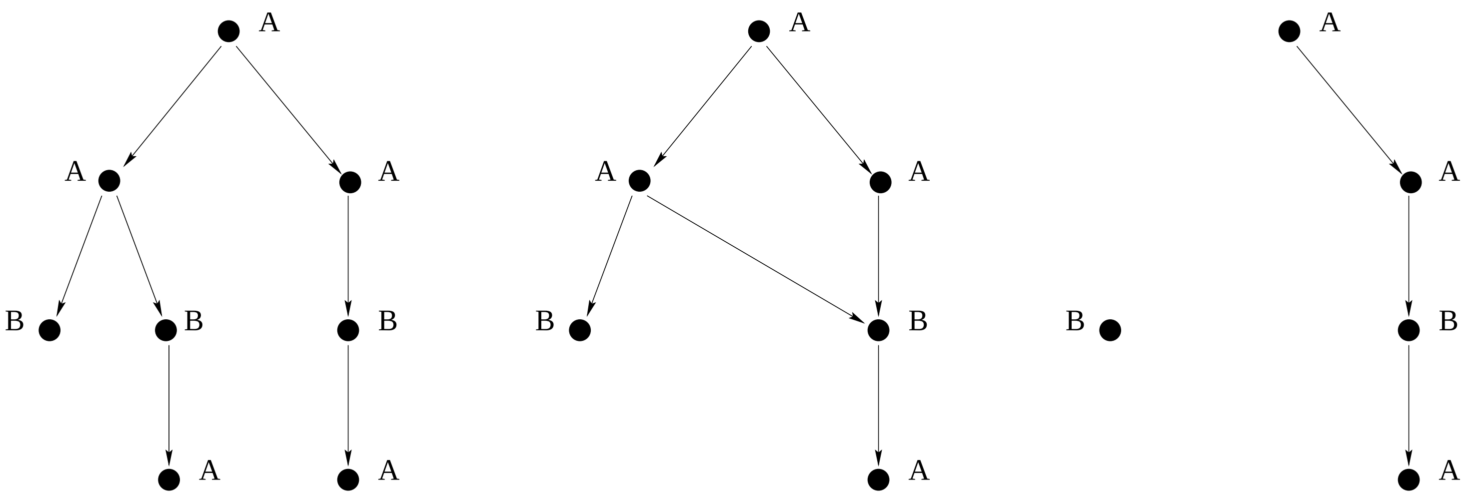
1. Bisimulation and Simulation
2. Encodings
3. Incremental Computation
4. OBDD Representation
5. Complexity Issues
6. Conclusions

What are (bi)simulations?

Equivalence relations on the nodes of a graph.

Which are their applications?

Reduce the state-spaces, compare systems, test properties, ...

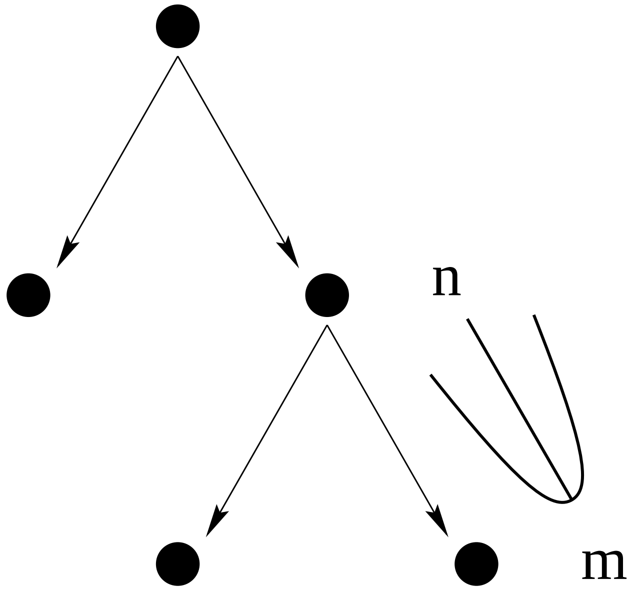


Labelled Graph Bisimulation Simulation

Colored Hypersets

$G = \langle N, E, \ell \rangle$ Colored Hyperset

$u' \in u \quad \text{iff} \quad u' \leftarrow u$



$$(E) \quad \forall u, v(u = v \leftrightarrow \ell(u) = \ell(v) \land \forall z(z \in u \leftrightarrow z \in v))$$

Bisimulation

Equality $\Leftrightarrow$ Bisimulation

$B \subseteq N \times N$ Bisimulation

$$\begin{array}{lllll}
\text{label} & uBv & \Rightarrow & \ell(u) = \ell(v) \\
\text{forw} & uBv \wedge u' \leftarrow u & \Rightarrow & \exists v'(v' \leftarrow v \wedge u'Bv') \\
\text{back} & uBv \wedge v' \leftarrow v & \Rightarrow & \exists u'(u' \leftarrow u \wedge u'Bv')
\end{array}$$

$\equiv$ Maximum Bisimulation

$$G \Rightarrow G/\equiv$$

Simulation

$S \subseteq N \times N$ Simulation

label $uSv \Rightarrow \ell(u) = \ell(v)$

forw $uSv \wedge u' \leftarrow u \Rightarrow \exists v'(v' \leftarrow v \wedge u'Sv')$

$\leq_s$ Maximum Simulation Quasi Order

$u \equiv_s v$ iff $u \leq_s v \wedge v \leq_s u$

$\equiv_s$ Maximum Simulation

$$G \Rightarrow G/\equiv_s$$

It preserves ACTL and it reduces the models more than Bisimulation

History

1971 Milner. An algebraic definition of simulation between programs

1989 Bloom. Time $O(|E|^6|N|)$.

1992 Cleaveland and Steffen. Time $O(|E|^2)$.

1993 Cleaveland, Parrow, and Steffen. Time $O(|E||N|^4)$.

1995 Bloom and Paige - Henzinger, Henzinger, and Kopke.
Time $O(|E||N|)$, Space $O(|N|^2 \log |N|)$.

2000 Bustan and Grumberg.
Time $O(|M|^2|N|^2(|M|^2 + |E|)$, Space $O(|M|^2 + |N| \log(|M|)$.

2001 Gentilini, Piazza, and Policriti.
Time $O(|M|^2|E|)$, Space $O(|M|^2 + |N| \log |M|)$.

Encodings

In the case of Bisimulation $\quad u \to u' \quad \Leftrightarrow \quad \mathit{Ack}(u) \ni \mathit{Ack}(u')$

It follows: $\quad u \equiv v \quad \Leftrightarrow \quad \mathit{Ack}(u) = \mathit{Ack}(v)$

Ackermann (1937) acyclic case.

Piazza Policriti (2001) cyclic case.

Advantages: by *rank* computation, OBDD representation.

In the case of Simulation $\quad u \leq_s v \quad \not\Leftrightarrow \quad \mathit{Ack}(u) \subseteq \mathit{Ack}(v)$

A sort of "recursive inclusion" would be necessary.

Idea: using *two* encodings

For each $u \in N$ use the codes $a(u)$ and $A(u)$.

Define the a's and the A's a way such that

$$u \leq_s v \quad \Leftrightarrow \quad a(u) \in A(v).$$

The easy ("expensive") way:

assign a different a-encoding to each node u, e.g.

$$a(u) = \{\emptyset\}^u = \underline{u}$$

determine the A-encoding using the a's.

The Constraint Defining the A's

From the Definition of Simulation

$$\begin{aligned} C(v) \quad = \quad & \{a(v)\} \subseteq \mathtt{A}_{\mathtt{v}} \subseteq \{a(u) \mid u \in N \wedge \ell(u) = \ell(v)\} \wedge \\ & \bigwedge_{\ell(u)=\ell(v)} (a(u) \in \mathtt{A}_{\mathtt{v}} \leftrightarrow \{a(u') \mid u \to u'\} \subseteq \bigcup_{v \to v'} \mathtt{A}'_{\mathtt{v}}) \end{aligned}$$

$$Sys_G \quad = \quad \bigwedge_{u \in N} C(u)$$

Max Simulation as Max Solution

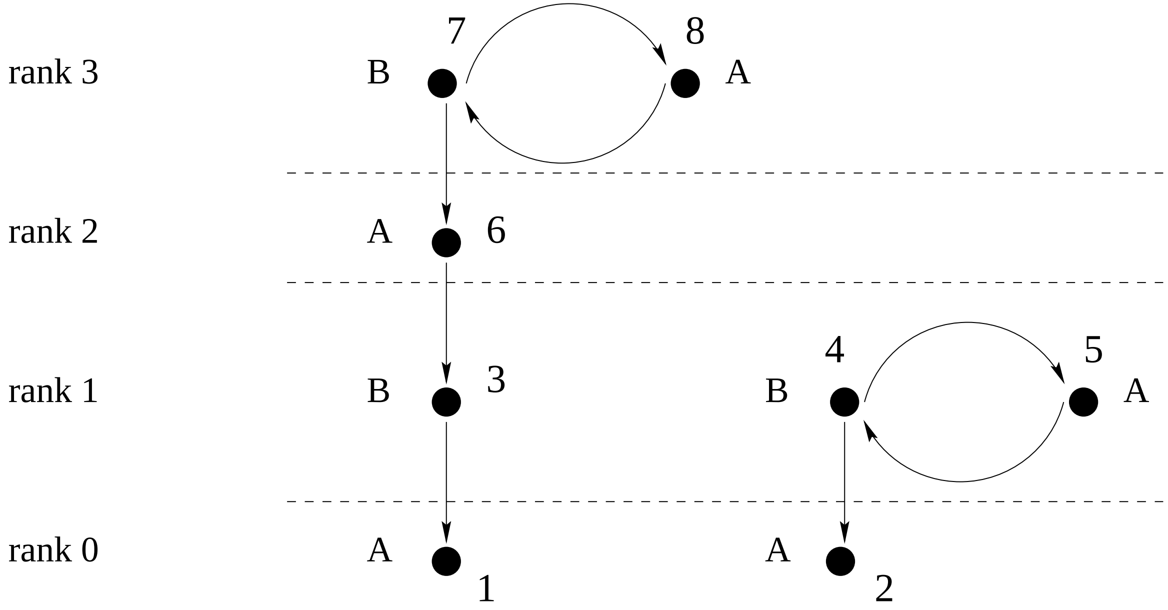
A solution Sol_1 of Sys_G $\quad\Rightarrow\quad$ A simulation $\leq_{Sol_1}$ on G

$$u \leq_{Sol_1} v \qquad \text{iff} \qquad a(u) \in A_1(v).$$

Let $Sol_1 \sqsubseteq Sol_2$ iff $\forall u \in N(A_1(u) \subseteq A_2(u))$

$$SM \text{ the } \sqsubseteq\text{-max sol. of } Sys_G \Rightarrow \begin{cases} a(u) \in A_{SM}(v) & \Leftrightarrow \quad u \leq_s v \\ \\ A_{SM}(u) = A_{SM}(v) & \Leftrightarrow \quad u \equiv_s v \end{cases}$$

The notion of *rank*



$rank(u)$ = length of the longest path from u to a leaf in the scc-graph.

Incremental Computation using *rank* - Basic Ideas (I)

Unfortunately

$$u \leq_s v \quad \not\Rightarrow \quad \mathit{rank}(u) \leq \mathit{rank}(v)$$

$$u \equiv_s v \quad \not\Rightarrow \quad \mathit{rank}(u) = \mathit{rank}(v)$$

At step i consider only the nodes up to *rank* $i-1$.

If at the end of step i $A(u) = A(v)$, then re-assign $a(u) := a(v)$.

At step $i+1$:

- compute the A-encoding of *rank* i;
- upgrade the old A-encoding with the nodes at *rank* i.

Incremental Computation using *rank* - Basic Ideas (II)

Let $rank(u), rank(v) < i$.

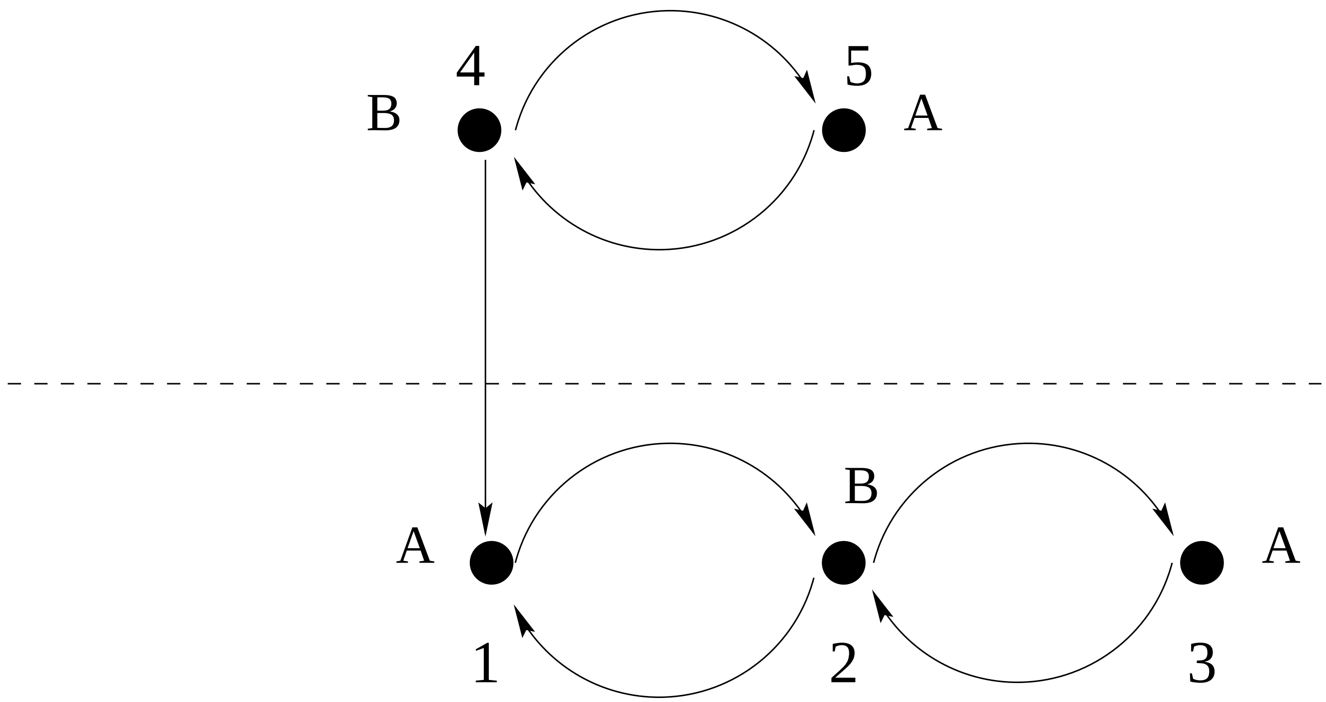
It is never necessary to remove $a(u)$ from $A(v)$.

If at step i $a(u) \notin A(v)$, this is definitely true and $u \not\leq_s v$.

If at step i $A(u) = A(v)$, this is definitely true and $u \equiv_s v$.

If at step i $A(u) \neq A(v)$, this is definitely true and $u \not\equiv_s v$.

Example



At the end of step 1 $A(1) = A(3) = \{1, 3\}, A(2) = \{2\}$,
hence $a(3) = a(1) = 1$.

At the end of step 1 $A(1) = A(5) = \{1, 5\}, A(2) = A(4) = \{2, 4\}$,
hence $a(5) = a(1) = 1, a(4) = a(2) = 2$.

Incremental Computation using *rank* (I)

At step i using the nodes at ranks $0, 1, 2, \ldots, i-1$
we have the partial solution

$$a(1) = \underline{1} \quad \ldots \quad a(k) = \underline{k'}$$

$$A(1) = \{a(1), \ldots\} \quad \ldots \quad A(k) = \{a(k), \ldots, \},$$

$$\text{with } a(h) = a(j) \Leftarrow A(h) = A(j)$$

$$\text{and } max\{a(1), \ldots, a(k)\} = \underline{m},$$

Incremental Computation using *rank* (II)

At step $i+1$ we have to:

consider all the nodes at rank $0, 1, 2, \ldots, i$;

assign to the s nodes at rank i an a-encoding

$$a(k+1) = \underline{m+1} \quad \ldots \quad a(k+s) = \underline{m+s};$$

if h is a node at rank $0, 1, 2, \ldots, i-1$, consider

$$A(h) \subseteq \mathtt{A_h} \subseteq A(h) \cup \{a(j) \mid rank(j) = i+1 \wedge \ell(j) = \ell(h)\} \wedge$$

$$\bigwedge_{rank(j)=i+1 \wedge \ell(j)=\ell(h)} (a(j) \in \mathtt{A_h} \leftrightarrow \{a(t) \mid j \to t\} \subseteq \bigcup_{h \to v} \mathtt{A_v});$$

Incremental Computation using *rank* (III)

if h is a node at rank i, consider

$$\{a(h)\} \subseteq \mathtt{A_h} \subseteq \{a(j) \mid rank(j) \leq i+1 \wedge \ell(j) = \ell(h)\} \wedge$$

$$\bigwedge_{rank(j) \leq i+1 \wedge \ell(j) = \ell(h)} (a(j) \in \mathtt{A_h} \leftrightarrow \{a(t) \mid j \to t\} \subseteq \bigcup_{h \to v} \mathtt{A_v});$$

if $A(1) = \{a(1), \dots\} \quad \dots \quad A(k+s) = \{a(k+s), \dots\}$ is the $\sqsubseteq$-max. solution and $A(h) = A(j) \wedge a(h) < a(j)$, then re-assign to j the a-encoding $a(h)$.

Fully Incremental OBDD representation

At step i if $rank(u) = i - 1$ consider the OBDD

$$E(u) \quad = \quad \bigvee_{u' \in Big(u)} a(u')$$

where

$$Big(u) \quad = \quad \{u' \mid u \to u' \wedge a(u') \notin \bigcup_{u \to u'' \wedge u'' \neq u'} A(u'')\}.$$

It is never necessary to upgrade $E(u)$!

If at step $j \geq i$ $A(u) = A(v)$, do not compute $E(v)$!

The collection of the OBDD's is a symbolic representation of $G/\equiv_s$.

Complexity Issues

Exploit existing constraint solvers, such as $CLP(\mathcal{SET})$.

Trade-off between the expressive power and the time complexity.

Easy to implement a constraint solver running in $O(|N|^3|E|)$.

Conclusions

We propose an alternative way to:

- efficiently compute $G/\equiv_s$
- produce simultaneously $\equiv_s$ and an OBDD representation

Under investigation:

- algorithmic complexity aspects and testing
- operations on the OBDD representation