Program Verification: Lecture 27

José Meseguer
University of Illinois at Urbana-Champaign

Simulation and Bisimulation Maps of Transition Systems

Given two transition systems $\mathcal{A} = (A, \rightarrow_{\mathcal{A}})$ and $\mathcal{B} = (B, \rightarrow_{\mathcal{B}})$, a simulation map f from $\mathcal{A}$ to $\mathcal{B}$, denoted $f : \mathcal{A} \rightarrow \mathcal{B}$, is a function $f : A \rightarrow B$ that is "transition preserving" in the sense that any transition $a \rightarrow_{\mathcal{A}} a'$ in $\mathcal{A}$ is mapped by f to a corresponding transition $f(a) \rightarrow_{\mathcal{B}} f(a')$ in $\mathcal{B}$.

Simulation and Bisimulation Maps of Transition Systems

Given two transition systems $\mathcal{A} = (A, \rightarrow_{\mathcal{A}})$ and $\mathcal{B} = (B, \rightarrow_{\mathcal{B}})$, a simulation map f from $\mathcal{A}$ to $\mathcal{B}$, denoted $f : \mathcal{A} \rightarrow \mathcal{B}$, is a function $f : A \rightarrow B$ that is "transition preserving" in the sense that any transition $a \rightarrow_{\mathcal{A}} a'$ in $\mathcal{A}$ is mapped by f to a corresponding transition $f(a) \rightarrow_{\mathcal{B}} f(a')$ in $\mathcal{B}$.

A simulation map $f : \mathcal{A} \rightarrow \mathcal{B}$ is called a bisimulation iff, in addition, for any state of the form $f(a) \in B$ and any transition $f(a) \rightarrow_{\mathcal{B}} b$ there exists and $a' \in A$ and transition $a \rightarrow_{\mathcal{A}} a'$ such that $f(a') = b$.

Simulation and Bisimulation Maps of Transition Systems

Given two transition systems $\mathcal{A} = (A, \to_{\mathcal{A}})$ and $\mathcal{B} = (B, \to_{\mathcal{B}})$, a simulation map f from $\mathcal{A}$ to $\mathcal{B}$, denoted $f : \mathcal{A} \to \mathcal{B}$, is a function $f : A \to B$ that is "transition preserving" in the sense that any transition $a \to_{\mathcal{A}} a'$ in $\mathcal{A}$ is mapped by f to a corresponding transition $f(a) \to_{\mathcal{B}} f(a')$ in $\mathcal{B}$.

A simulation map $f : \mathcal{A} \to \mathcal{B}$ is called a bisimulation iff, in addition, for any state of the form $f(a) \in B$ and any transition $f(a) \to_{\mathcal{B}} b$ there exists and $a' \in A$ and transition $a \to_{\mathcal{A}} a'$ such that $f(a') = b$.

Given a transition system $\mathcal{A} = (A, \to_{\mathcal{A}})$ and subsets $U, V \subseteq A$, we are interested in the reachability property:

$$\exists x \in U,\ \exists y \in V,\ x \to_{\mathcal{A}}^{*} y$$

which we abbreviate to $\exists U \to^{*} V$. If this property holds for specific $U, V \subseteq A$ we write: $\mathcal{A} \models \exists U \to^{*} V$.

Simulation and Bisimulation Maps of Transition Systems

Given two transition systems $\mathcal{A} = (A, \to_{\mathcal{A}})$ and $\mathcal{B} = (B, \to_{\mathcal{B}})$, a simulation map f from $\mathcal{A}$ to $\mathcal{B}$, denoted $f : \mathcal{A} \to \mathcal{B}$, is a function $f : A \to B$ that is "transition preserving" in the sense that any transition $a \to_{\mathcal{A}} a'$ in $\mathcal{A}$ is mapped by f to a corresponding transition $f(a) \to_{\mathcal{B}} f(a')$ in $\mathcal{B}$.

A simulation map $f : \mathcal{A} \to \mathcal{B}$ is called a bisimulation iff, in addition, for any state of the form $f(a) \in B$ and any transition $f(a) \to_{\mathcal{B}} b$ there exists and $a' \in A$ and transition $a \to_{\mathcal{A}} a'$ such that $f(a') = b$.

Given a transition system $\mathcal{A} = (A, \to_{\mathcal{A}})$ and subsets $U, V \subseteq A$, we are interested in the reachability property:

$$\exists x \in U,\ \exists y \in V,\ x \to^*_{\mathcal{A}} y$$

which we abbreviate to $\exists U \to^* V$. If this property holds for specific $U, V \subseteq A$ we write: $\mathcal{A} \models \exists U \to^* V$. Note that $\mathcal{A} \not\models \exists U \to^* V$ iff $\forall x \in U,\ \forall y \in V,\ x \not\to^*_{\mathcal{A}} y$ holds in $\mathcal{A}$, abbreviated $\mathcal{A} \models \forall U \not\to^* V$.

Preservation of Reachability Properties by (Bi)Simulations

The proofs of these two theorems are given in the Appendix.

Preservation of Reachability Properties by (Bi)Simulations

The proofs of these two theorems are given in the Appendix.

Theorem

Let $f : \mathcal{A} \to \mathcal{B}$ be a simulation map, then for any $U, V \subseteq A$, $\mathcal{A} \models \exists U \to^* V$ implies $\mathcal{B} \models \exists f(U) \to^* f(V)$. Equivalently, $\mathcal{B} \models \forall f(U) \not\to^* f(V)$ implies $\mathcal{A} \models \forall U \not\to^* V$.

Preservation of Reachability Properties by (Bi)Simulations

The proofs of these two theorems are given in the Appendix.

Theorem

Let $f : \mathcal{A} \to \mathcal{B}$ *be a simulation map, then for any* $U, V \subseteq A$, $\mathcal{A} \models \exists U \to^* V$ *implies* $\mathcal{B} \models \exists f(U) \to^* f(V)$. *Equivalently,* $\mathcal{B} \models \forall f(U) \not\to^* f(V)$ *implies* $\mathcal{A} \models \forall U \not\to^* V$.

Theorem

Let $f : \mathcal{A} \to \mathcal{B}$ *be a bisimulation map, then for any* $U, V \subseteq A$, $\mathcal{A} \models \exists U \to^* V$ *iff* $\mathcal{B} \models \exists f(U) \to^* f(V)$. *Equivalently,* $\mathcal{A} \models \forall U \not\to^* V$ *iff* $\mathcal{B} \models \forall f(U) \not\to^* f(V)$.

Preservation of Reachability Properties by (Bi)Simulations

The proofs of these two theorems are given in the Appendix.

Theorem

Let $f : \mathcal{A} \to \mathcal{B}$ be a simulation map, then for any $U, V \subseteq A$, $\mathcal{A} \models \exists U \to^* V$ implies $\mathcal{B} \models \exists f(U) \to^* f(V)$. Equivalently, $\mathcal{B} \models \forall f(U) \not\to^* f(V)$ implies $\mathcal{A} \models \forall U \not\to^* V$.

Theorem

Let $f : \mathcal{A} \to \mathcal{B}$ be a bisimulation map, then for any $U, V \subseteq A$, $\mathcal{A} \models \exists U \to^* V$ iff $\mathcal{B} \models \exists f(U) \to^* f(V)$. Equivalently, $\mathcal{A} \models \forall U \not\to^* V$ iff $\mathcal{B} \models \forall f(U) \not\to^* f(V)$.

Note the for $U, I \subseteq A$, I is an invariant from U iff $\mathcal{A} \models \forall U \not\to^* A \setminus I$. Thus, we can verify the invariant by proving $\mathcal{B} \models \forall f(U) \not\to^* f(A \setminus I)$.

Preservation of Reachability Properties by (Bi)Simulations

The proofs of these two theorems are given in the Appendix.

Theorem

Let $f : \mathcal{A} \to \mathcal{B}$ be a simulation map, then for any $U, V \subseteq A$, $\mathcal{A} \models \exists U \to^* V$ implies $\mathcal{B} \models \exists f(U) \to^* f(V)$. Equivalently, $\mathcal{B} \models \forall f(U) \not\to^* f(V)$ implies $\mathcal{A} \models \forall U \not\to^* V$.

Theorem

Let $f : \mathcal{A} \to \mathcal{B}$ be a bisimulation map, then for any $U, V \subseteq A$, $\mathcal{A} \models \exists U \to^* V$ iff $\mathcal{B} \models \exists f(U) \to^* f(V)$. Equivalently, $\mathcal{A} \models \forall U \not\to^* V$ iff $\mathcal{B} \models \forall f(U) \not\to^* f(V)$.

Note the for $U, I \subseteq A$, I is an invariant from U iff $\mathcal{A} \models \forall U \not\to^* A \setminus I$. Thus, we can verify the invariant by proving $\mathcal{B} \models \forall f(U) \not\to^* f(A \setminus I)$. But we could have $\mathcal{B} \models \exists f(U) \to^* f(A \setminus I)$, while $\mathcal{A} \models \forall U \not\to^* A \setminus I$ (spurious counterexample).

Preservation of Reachability Properties by (Bi)Simulations

The proofs of these two theorems are given in the Appendix.

Theorem

Let $f : \mathcal{A} \to \mathcal{B}$ be a simulation map, then for any $U, V \subseteq A$, $\mathcal{A} \models \exists U \to^* V$ implies $\mathcal{B} \models \exists f(U) \to^* f(V)$. Equivalently, $\mathcal{B} \models \forall f(U) \not\to^* f(V)$ implies $\mathcal{A} \models \forall U \not\to^* V$.

Theorem

Let $f : \mathcal{A} \to \mathcal{B}$ be a bisimulation map, then for any $U, V \subseteq A$, $\mathcal{A} \models \exists U \to^* V$ iff $\mathcal{B} \models \exists f(U) \to^* f(V)$. Equivalently, $\mathcal{A} \models \forall U \not\to^* V$ iff $\mathcal{B} \models \forall f(U) \not\to^* f(V)$.

Note the for $U, I \subseteq A$, I is an invariant from U iff $\mathcal{A} \models \forall U \not\to^* A \setminus I$. Thus, we can verify the invariant by proving $\mathcal{B} \models \forall f(U) \not\to^* f(A \setminus I)$. But we could have $\mathcal{B} \models \exists f(U) \to^* f(A \setminus I)$, while $\mathcal{A} \models \forall U \not\to^* A \setminus I$ (spurious counterexample). However, if f is a bisimulation no spurious counterexamples can exist.

Equational Abstractions

Simulation and bisimulation maps can be very useful to verify properties of concurrent systems specified as (not necessarily topmost) rewrite theories $\mathcal{R} = (\Sigma, E \cup B, R)$, not by reasoning directly on $\mathcal{R}$, but by shifting our ground and reasoning on a quotient of $\mathcal{R}$.

Equational Abstractions

Simulation and bisimulation maps can be very useful to verify properties of concurrent systems specified as (not necessarily topmost) rewrite theories $\mathcal{R} = (\Sigma, E \cup B, R)$, not by reasoning directly on $\mathcal{R}$, but by shifting our ground and reasoning on a quotient of $\mathcal{R}$. Specifically, we can add more equations, say $G = E' \cup B'$ to $\mathcal{R}$ to identify more states.

Equational Abstractions

Simulation and bisimulation maps can be very useful to verify properties of concurrent systems specified as (not necessarily topmost) rewrite theories $\mathcal{R} = (\Sigma, E \cup B, R)$, not by reasoning directly on $\mathcal{R}$, but by shifting our ground and reasoning on a quotient of $\mathcal{R}$. Specifically, we can add more equations, say $G = E' \cup B'$ to $\mathcal{R}$ to identify more states. We then call the resulting rewrite theory $(\Sigma, E \cup B \cup G, R)$ an equational abstraction of $\mathcal{R}$, denoted $\mathcal{R}/G$.

Equational Abstractions

Simulation and bisimulation maps can be very useful to verify properties of concurrent systems specified as (not necessarily topmost) rewrite theories $\mathcal{R} = (\Sigma, E \cup B, R)$, not by reasoning directly on $\mathcal{R}$, but by shifting our ground and reasoning on a quotient of $\mathcal{R}$. Specifically, we can add more equations, say $G = E' \cup B'$ to $\mathcal{R}$ to identify more states. We then call the resulting rewrite theory $(\Sigma, E \cup B \cup G, R)$ an equational abstraction of $\mathcal{R}$, denoted $\mathcal{R}/G$.

The following theorem follows trivially from the fact that if $t \to_{R/E \cup B} t'$, then, a fortiori, $t \to_{R/E \cup B \cup G} t'$.

Equational Abstractions

Simulation and bisimulation maps can be very useful to verify properties of concurrent systems specified as (not necessarily topmost) rewrite theories $\mathcal{R} = (\Sigma, E \cup B, R)$, not by reasoning directly on $\mathcal{R}$, but by shifting our ground and reasoning on a quotient of $\mathcal{R}$. Specifically, we can add more equations, say $G = E' \cup B'$ to $\mathcal{R}$ to identify more states. We then call the resulting rewrite theory $(\Sigma, E \cup B \cup G, R)$ an equational abstraction of $\mathcal{R}$, denoted $\mathcal{R}/G$.

The following theorem follows trivially from the fact that if $t \rightarrow_{R/E \cup B} t'$, then, a fortiori, $t \rightarrow_{R/E \cup B \cup G} t'$.

Theorem

Given a rewrite theory $\mathcal{R} = (\Sigma, E \cup B, R)$, Σ-equations $G = E' \cup B'$, and a top sort $State$, the unique surjective Σ-homomorphism $[\_]_{E \cup B \cup G} : \mathbb{T}_{\Sigma/E \cup B} \rightarrow \mathbb{T}_{\Sigma/E \cup B \cup G}$ induces a simulation map $[\_]_{E \cup B \cup G} : (T_{\Sigma/E \cup B, State}, \rightarrow_{R/E \cup B}) \rightarrow (T_{\Sigma/E \cup B \cup G, State}, \rightarrow_{R/E \cup B \cup G})$.

Equational Abstractions (II)

Equational abstractions can make the set of reachable states from an initial state *init* finite. In this way, invariants and, more generally, LTL properties that cannot be verified by explicit-state model checking can be verified using an equational abstraction $\mathcal{R}/G$.

Equational Abstractions (II)

Equational abstractions can make the set of reachable states from an initial state *init* finite. In this way, invariants and, more generally, LTL properties that cannot be verified by explicit-state model checking can be verified using an equational abstraction $\mathcal{R}/G$.

I refer to §12.4 and §13.4 of *All About Maude* for further details on the use of equational abstraction for explicit-state model checking of (respectively) invariants and LTL properties.

Equational Abstractions (II)

Equational abstractions can make the set of reachable states from an initial state *init* finite. In this way, invariants and, more generally, LTL properties that cannot be verified by explicit-state model checking can be verified using an equational abstraction $\mathcal{R}/G$.

I refer to §12.4 and §13.4 of *All About Maude* for further details on the use of equational abstraction for explicit-state model checking of (respectively) invariants and LTL properties.

In what follows I shall focus on the use of equational abstractions for symbolic model checking. Therefore, I will assume a topmost rewrite theory $\mathcal{R} = (\Sigma, E \cup B, R)$ such that $E \cup B$ is FVP. We shall then be interested in equational abstractions of the form $\mathcal{R}/G$, where $G = E' \cup B'$ is such that $E \cup E' \cup B \cup B'$ is also FVP modulo $B \cup B'$.

Equational Abstractions (II)

Equational abstractions can make the set of reachable states from an initial state *init* finite. In this way, invariants and, more generally, LTL properties that cannot be verified by explicit-state model checking can be verified using an equational abstraction $\mathcal{R}/G$.

I refer to §12.4 and §13.4 of *All About Maude* for further details on the use of equational abstraction for explicit-state model checking of (respectively) invariants and LTL properties.

In what follows I shall focus on the use of equational abstractions for symbolic model checking. Therefore, I will assume a topmost rewrite theory $\mathcal{R} = (\Sigma, E \cup B, R)$ such that $E \cup B$ is FVP. We shall then be interested in equational abstractions of the form $\mathcal{R}/G$, where $G = E' \cup B'$ is such that $E \cup E' \cup B \cup B'$ is also FVP modulo $B \cup B'$.

Since for each pattern term with variables p, the quotient homomorphism $[\_]_{E \cup B \cup G} : \mathbb{T}_{\Sigma/E \cup B}(X) \rightarrow \mathbb{T}_{\Sigma/E \cup B \cup G}(X)$ maps each $[p]_{E \cup B}$ to $[p]_{E \cup B \cup G}$, p in $\mathcal{R}/G$ just describes the image under $[\_]_{E \cup B \cup G}$ of p in $\mathcal{R}$ as the symbolic description of the set $[\![p]\!]_{\mathcal{R}}$ of all $E \cup B$-equivalence classes of ground instances of p, which is just $[\![p]\!]_{\mathcal{R}/G}$.

Equational Abstractions (III)

In particular, if the complement of an invariant I in $\mathcal{R}$ is symbolically described by a finite set of pattern terms $p_1, \ldots, p_k$, in case the symbolic state space to reach an instance of some p_i from a symbolic initial state u is infinite, we can use a topmost equational abstraction $\mathcal{R}/G$ whose equations are FVP to try to make the symbolic search space finite.

Equational Abstractions (III)

In particular, if the complement of an invariant I in $\mathcal{R}$ is symbolically described by a finite set of pattern terms $p_1, \ldots, p_k$, in case the symbolic state space to reach an instance of some p_i from a symbolic initial state u is infinite, we can use a topmost equational abstraction $\mathcal{R}/G$ whose equations are FVP to try to make the symbolic search space finite.

Then, by the first Theorem in pg. 3 of this 3 of this lecture, we can use symbolic model checking from a symbolic initial state u to show in $\mathcal{R}/G$ that $\forall u \not\rightarrow^* p_i$, $1 \leq i \leq k$. However, in some cases we might get some spurious counterexample.

Equational Abstractions (III)

In particular, if the complement of an invariant I in $\mathcal{R}$ is symbolically described by a finite set of pattern terms $p_1, \ldots, p_k$, in case the symbolic state space to reach an instance of some p_i from a symbolic initial state u is infinite, we can use a topmost equational abstraction $\mathcal{R}/G$ whose equations are FVP to try to make the symbolic search space finite.

Then, by the first Theorem in pg. 3 of this 3 of this lecture, we can use symbolic model checking from a symbolic initial state u to show in $\mathcal{R}/G$ that $\forall u \not\to^* p_i$, $1 \leq i \leq k$. However, in some cases we might get some spurious counterexample.

But by the second Theorem in page 3 of this lecture, no spurious counterexamples will exist if the homomorphism $[\_]_{E \cup B \cup G} : \mathbb{T}_{\Sigma/E \cup B} \to \mathbb{T}_{\Sigma/E \cup B \cup G}$ actually defines a bisimulation. I shall focus on bisimulations in what follows.

An Equational Abstraction for R&W

Recall that it was impossible to verify the mutual exclusion and one-writer invariants for BAKERY from $< \mathtt{0},\ \mathtt{0} >$ by narrowing in a forwards direction: one had to narrow backwards.

An Equational Abstraction for R&W

Recall that it was impossible to verify the mutual exclusion and one-writer invariants for BAKERY from < 0, 0 > by narrowing in a forwards direction: one had to narrow backwards. But we can symbolicallt verify both invariants by forwards narrowing in an equational abstraction of R&W from < 0, 0 >.

An Equational Abstraction for R&W

Recall that it was impossible to verify the mutual exclusion and one-writer invariants for BAKERY from $<$ `0, 0` $>$ by narrowing in a forwards direction: one had to narrow backwards. But we can symbolicallt verify both invariants by forwards narrowing in an equational abstraction of R&W from $<$ `0, 0` $>$. Can you guess the G?

An Equational Abstraction for R&W

Recall that it was impossible to verify the mutual exclusion and one-writer invariants for BAKERY from < 0, 0 > by narrowing in a forwards direction: one had to narrow backwards. But we can symbolicallt verify both invariants by forwards narrowing in an equational abstraction of R&W from < 0, 0 >. Can you guess the G?

```
mod R&W is
  sorts Nat Config .
  op <_,_> : Nat Nat -> Config [ctor] .
  op 0 : -> Nat [ctor] .
  op s : Nat -> Nat [ctor] .
  vars R W : Nat .

  rl < 0, 0 > => < 0, s(0) > [narrowing] .
  rl < R, s(W) > => < R, W > [narrowing] .
  rl < R, 0 > => < s(R), 0 > [narrowing] .
  rl < s(R), W > => < R, W > [narrowing] .
endm
```

An Equational Abstraction for R&W

Recall that it was impossible to verify the mutual exclusion and one-writer invariants for BAKERY from < 0, 0 > by narrowing in a forwards direction: one had to narrow backwards. But we can symbolicallt verify both invariants by forwards narrowing in an equational abstraction of R&W from < 0, 0 >. Can you guess the G?

```
mod R&W is
  sorts Nat Config .
  op <_,_> : Nat Nat -> Config [ctor] .
  op 0 : -> Nat [ctor] .
  op s : Nat -> Nat [ctor] .
  vars R W : Nat .

  rl < 0, 0 > => < 0, s(0) > [narrowing] .
  rl < R, s(W) > => < R, W > [narrowing] .
  rl < R, 0 > => < s(R), 0 > [narrowing] .
  rl < s(R), W > => < R, W > [narrowing] .
endm
```

The equation < s(s(N)),0 > = < s(0),0 > is confluent, terminating and FVP and provides the desired abstraction:

An Equational Abstraction for R&W (II)

```
mod R&W-ABS is
    including R&W .
    vars N M R W : Nat .
    eq < s(s(N)),0 > = < s(0),0 > [variant] .
endm


Maude> {fold} vu-narrow < 0, 0 > =>* < s(N:Nat), s(M:Nat) > .

No solution.

Maude> {fold} vu-narrow < 0 , 0 > =>* < N:Nat , s(s(M:Nat)) > .
fvu-narrow in R&W-ABS : < 0,0 > =>* < N,s(s(M)) > .

No solution.
```

An Equational Abstraction for R&W (II)

```
mod R&W-ABS is
    including R&W .
    vars N M R W : Nat .
    eq < s(s(N)),0 > = < s(0),0 > [variant] .
endm
```

```
Maude> {fold} vu-narrow < 0, 0 > =>* < s(N:Nat), s(M:Nat) > .

No solution.

Maude> {fold} vu-narrow < 0 , 0 > =>* < N:Nat , s(s(M:Nat)) > .
fvu-narrow in R&W-ABS : < 0,0 > =>* < N,s(s(M)) > .

No solution.
```

Of course, in this example the equational abstraction was not needed: we could symbolically verify these properties from the more general pattern < R,0 >.

An Equational Abstraction for R&W (II)

```
mod R&W-ABS is
    including R&W .
    vars N M R W : Nat .
    eq < s(s(N)),0 > = < s(0),0 > [variant] .
endm
```

```
Maude> {fold} vu-narrow < 0, 0 > =>* < s(N:Nat), s(M:Nat) > .

No solution.

Maude> {fold} vu-narrow < 0 , 0 > =>* < N:Nat , s(s(M:Nat)) > .
fvu-narrow in R&W-ABS : < 0,0 > =>* < N,s(s(M)) > .

No solution.
```

Of course, in this example the equational abstraction was not needed: we could symbolically verify these properties from the more general pattern `< R,0 >`. However, folding variant narrowing may loop in other examples, yet may reach a fixpoint using an equational abstraction; sometimes (like above) even from a ground initial state.

Bakery Protocol: Transition System

Token to give **;** Token serving **;** Set of Processes
Nat Nat [{ idle, wait(Nat), crit(Nat) }]

$$
\begin{array}{llll}
\texttt{rl}\ N ; M ; [\mathrm{idle}]\ PS & \Rightarrow (s\,N) ; & M & ; [\mathrm{wait}(N)]\ PS\ . \\
\texttt{rl}\ N ; M ; [\mathrm{wait}(M)]\ PS & \Rightarrow \quad N & ; \quad M & ; [\mathrm{crit}(M)]\ PS\ . \\
\texttt{rl}\ N ; M ; [\mathrm{crit}(M)]\ PS & \Rightarrow \quad N & ; (s\,M) & ; [\mathrm{idle}]\ PS\ .
\end{array}
$$

Bakery Protocol: Transition System

Token to give **;** Token serving **;** Set of Processes

Nat Nat [{ idle, wait(Nat), crit(Nat) }]

$$
\begin{array}{l}
\text{rl } N \;;\; M \;;\; [\text{idle}]\ PS \Rightarrow (s\,N) \;;\; M \;;\; [\text{wait}(N)]\ PS\ . \\
\text{rl } N \;;\; M \;;\; [\text{wait}(M)]\ PS \Rightarrow N \;;\; M \;;\; [\text{crit}(M)]\ PS\ . \\
\text{rl } N \;;\; M \;;\; [\text{crit}(M)]\ PS \Rightarrow N \;;\; (s\,M) \;;\; [\text{idle}]\ PS\ .
\end{array}
$$

$$
\begin{tikzcd}
\mathtt{0\ ;\ 0\ ;\ [idle]} \arrow[d] & \mathtt{s\ ;\ s\ ;\ [idle]} \arrow[d] & \mathtt{s\,s\ ;\ s\,s\ ;\ [idle]} \arrow[d] \\
\mathtt{s\ ;\ 0\ ;\ [wait(0)]} \arrow[d] & \mathtt{s\ ;\ s\ ;\ [wait(s)]} \arrow[d] & \mathtt{s\,s\,s\ ;\ s\,s\ ;\ [wait(s\,s)]} \arrow[d] \\
\mathtt{s\ ;\ 0\ ;\ [crit(s)]} \arrow[uur] & \mathtt{s\,s\ ;\ s\ ;\ [crit(s)]} \arrow[uur] & \infty
\end{tikzcd}
$$

(Transition System: one initial state - infinite space)

Bakery Protocol: Symbolic Transition System

$$\begin{tikzcd}
\mathtt{0 ; 0 ; [idle]} \arrow[d] & \mathtt{s ; s ; [idle]} \arrow[d] & \mathtt{s\,s ; s\,s ; [idle]} \arrow[d] \\
\mathtt{s ; 0 ; [wait(0)]} \arrow[d] & \mathtt{s ; s ; [wait(s)]} \arrow[d] & \mathtt{s\,s\,s ; s\,s ; [wait(s\,s)]} \arrow[d] \\
\mathtt{s ; 0 ; [crit(s)]} \arrow[uur] & \mathtt{s\,s ; s ; [crit(s)]} \arrow[uur] & \infty
\end{tikzcd}$$

(Transition System: one initial state - infinite state space)

Bakery Protocol: Symbolic Transition System

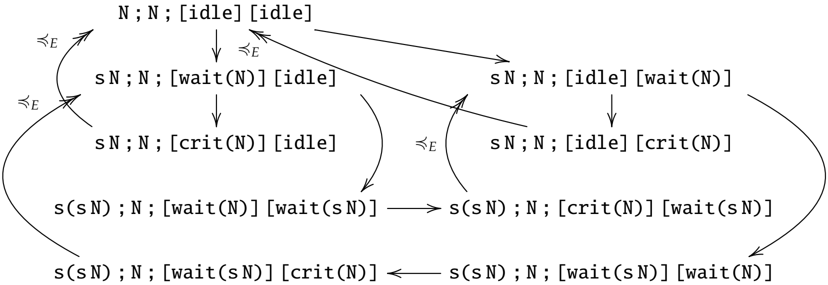
$$
\begin{tikzcd}
\mathtt{0\ ;\ 0\ ;\ [idle]} \arrow[d] & \mathtt{s\ ;\ s\ ;\ [idle]} \arrow[d] & \mathtt{s\,s\ ;\ s\,s\ ;\ [idle]} \arrow[d] \\
\mathtt{s\ ;\ 0\ ;\ [wait(0)]} \arrow[d] & \mathtt{s\ ;\ s\ ;\ [wait(s)]} \arrow[d] & \mathtt{s\,s\,s\ ;\ s\,s\ ;\ [wait(s\,s)]} \arrow[d] \\
\mathtt{s\ ;\ 0\ ;\ [crit(s)]} \arrow[uur] & \mathtt{s\,s\ ;\ s\ ;\ [crit(s)]} \arrow[uur] & \infty
\end{tikzcd}
$$

(Transition System: one initial state - infinite state space)

$$
\begin{tikzcd}
\mathtt{N\,;\,N\,;\,[idle]} \arrow[d] & \mathtt{s\,N\,;\,s\,N\,;\,[idle]} \arrow[d] & \mathtt{s\,s\,N\,;\,s\,s\,N\,;\,[idle]} \arrow[d] \\
\mathtt{s\,N\,;\,N\,;\,[wait(N)]} \arrow[d] & \mathtt{s\,s\,N\,;\,s\,N\,;\,[wait(s\,N)]} \arrow[d] & \mathtt{s\,s\,s\,N\,;\,s\,s\,N\,;\,[wait(s\,s\,N)]} \arrow[d] \\
\mathtt{s\,N\,;\,N\,;\,[crit(N)]} \arrow[uur] & \mathtt{s\,s\,N\,;\,s\,N\,;\,[crit(s\,N)]} \arrow[uur] & \cdots
\end{tikzcd}
$$

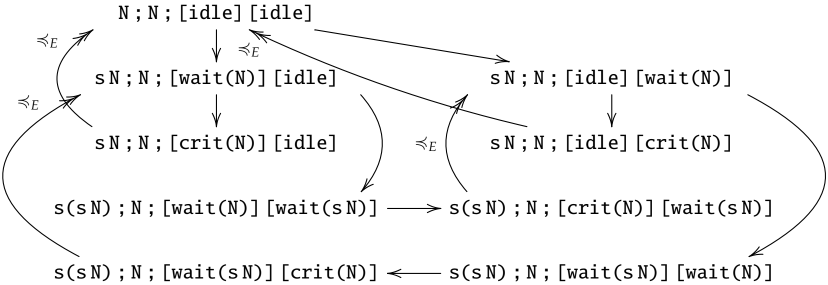
(Symbolic Transition System: infinite initial state set - infinite state space)

Bakery Protocol: Folding the Symbolic Transition System



(Folding Symbolic Transition System : infinite initial state set - finite state space)

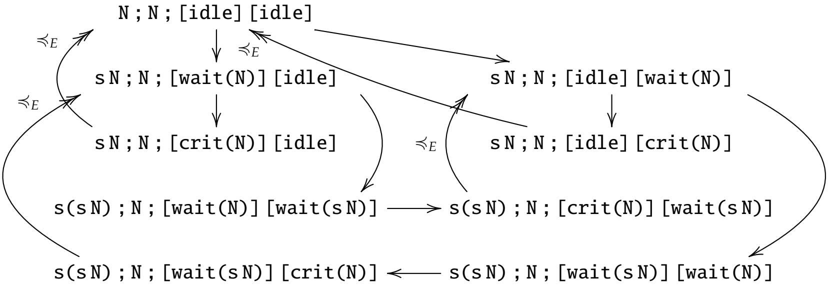
Bakery Protocol: Folding the Symbolic Transition System



(Folding Symbolic Transition System : infinite initial state set - finite state space)

However, folding variant narrowing loops if we start with a more general symbolic initial state of the form: `N ; N  IS`, where `IS` is a variable of sort `IdleProcesses`.

Bakery Protocol: Folding the Symbolic Transition System



(Folding Symbolic Transition System : infinite initial state set - finite state space)

However, folding variant narrowing loops if we start with a more general symbolic initial state of the form: `N ; N  IS`, where `IS` is a variable of sort `IdleProcesses`. Let us explore the notion of bisimilar equational abstractions.

Bisimilar Equational Abstractions

We say that an equational abstraction $\mathcal{R}/G$ defines an bisimilar equational abstraction of $\mathcal{R}$ iff the simulation map

$$[\_]_{E \cup B \cup G} : (T_{\Sigma/E \cup B, State}, \rightarrow_{R/E \cup B}) \rightarrow (T_{\Sigma/E \cup B \cup G, State}, \rightarrow_{R/E \cup B \cup G})$$

is actually a bisimulation.

Bisimilar Equational Abstractions

We say that an equational abstraction $\mathcal{R}/G$ defines an bisimilar equational abstraction of $\mathcal{R}$ iff the simulation map

$$[\_]_{E \cup B \cup G} : (T_{\Sigma/E \cup B, State}, \to_{R/E \cup B}) \to (T_{\Sigma/E \cup B \cup G, State}, \to_{R/E \cup B \cup G})$$

is actually a bisimulation. We are interested in finding checkable conditions ensuring that G defines a bisimilar equational abstraction. See the Appendix for a proof of the following theorem:

Bisimilar Equational Abstractions

We say that an equational abstraction $\mathcal{R}/G$ defines an bisimilar equational abstraction of $\mathcal{R}$ iff the simulation map

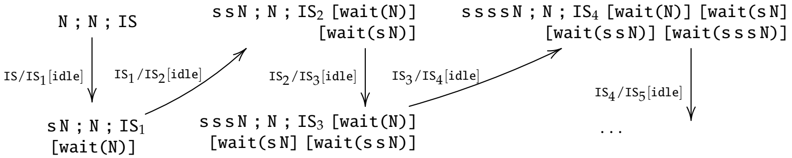
$$[\_]_{E \cup B \cup G} : (T_{\Sigma/E \cup B, State}, \to_{R/E \cup B}) \to (T_{\Sigma/E \cup B \cup G, State}, \to_{R/E \cup B \cup G})$$

is actually a bisimulation. We are interested in finding checkable conditions ensuring that G defines a bisimilar equational abstraction. See the Appendix for a proof of the following theorem:

Theorem

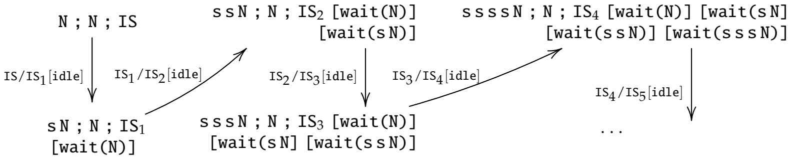
Let $\mathcal{R} = (\Sigma, E \cup B, R)$ be a topmost rewrite theory such that $G = E \cup B$ is FVP, and $G' = E' \cup B'$ is such that $E \cup E' \cup B \cup B'$ is FVP modulo $B \cup B'$. $\mathcal{R}/G'$ defines a bisimilar equational abstraction of $\mathcal{R}$ if for each $(u_0^i = u_1^i) \in G'$, $1 \leq i \leq p$, and $(t_0^j \to t_1^j) \in R$, $1 \leq j \leq q$, and each $\sigma \in Unif_G(t_{b'}^j = u_b^i)$, $0 \leq b \leq 1$, $0 \leq b' \leq 1$, there exists a θ such that $u_{b' \oplus 1}^i \sigma =_G t_b^j \theta \ \wedge \ t_{b \oplus 1}^j \theta =_G t_{b \oplus 1}^j \sigma$, where $\oplus$ denotes exclusive or.

Bakery Protocol: Infinite-State for some Initial States



(Infinite Folding Logical Transition System : infinite initial state - infinite state space)

Bakery Protocol: Infinite-State for some Initial States



(Infinite Folding Logical Transition System : infinite initial state - infinite state space)

- Many verification problems for infinite-state systems are due to unbounded number of processes
- All approaches use a symbolic finite representation of an infinite number of processes
- Bisimulation proofs written by hand or hard to reuse

An Equational Abstraction for the Bakery Protocol

- For our bakery protocol we can obtain a bisimilar equational abstraction by restricting the abstraction only to the following equation G', which intuitively collapses extra waiting processes that do not introduce any new behaviors:

An Equational Abstraction for the Bakery Protocol

- For our bakery protocol we can obtain a bisimilar equational abstraction by restricting the abstraction only to the following equation G', which intuitively collapses extra waiting processes that do not introduce any new behaviors:
- G':

```
eq (s s s L M) ; M ; PS0 [wait(s L M)] [wait(s s L M)]
= (s s L M) ; M ; PS0 [wait(s L M)] .
```

An Equational Abstraction for the Bakery Protocol

- For our bakery protocol we can obtain a bisimilar equational abstraction by restricting the abstraction only to the following equation G', which intuitively collapses extra waiting processes that do not introduce any new behaviors:
- G':

```
eq (s s s L M) ; M ; PS_0 [wait(s L M)] [wait(s s L M)]
= (s s L M) ; M ; PS_0 [wait(s L M)] .
```

$$
\begin{tikzcd}
\underline{\mathtt{N ; N ; IS}} \arrow[r, "\mathtt{IS/IS_1[idle]}"'] & \mathtt{s\,N; N ; IS_1\ [wait(N)]} \arrow[r] \arrow[d, "\mathtt{IS_1/IS_2[idle]}"] & \mathtt{s\,N; N ; IS_1\ [crit(N)]} \arrow[ll, bend right] \\
& \mathtt{s\,s\,N; N ; IS_2\ [wait(N)]\ [wait(s\,N)]} \arrow[loop left, "\mathtt{IS_2/IS_3[idle]}"] \arrow[r] & \mathtt{s\,s\,N; N ; IS_2\ [crit(N)]\ [wait(s\,N)]} \arrow[ul]
\end{tikzcd}
$$

(Abstract Bisimilar Folding Logical Transition System)