Program Verification: Lecture 23

José Meseguer

University of Illinois at Urbana-Champaign

Symbolic Evaluation

Consider the equations [1] $n + 0 = n$, [2] $n + s(m) = s(n + m)$ defining natural number addition.

Symbolic Evaluation

Consider the equations [1] $n + 0 = n$, [2] $n + s(m) = s(n + m)$ defining natural number addition.

Q1: Can we evaluate $x + y$?

Symbolic Evaluation

Consider the equations [1] $n + 0 = n$, [2] $n + s(m) = s(n + m)$ defining natural number addition.

Q1: Can we evaluate $x + y$?

A1: No, since $x + y$ is not an instance of either $n + 0$ or $n + s(m)$.

Symbolic Evaluation

Consider the equations [1] $n + 0 = n$, [2] $n + s(m) = s(n + m)$ defining natural number addition.

Q1: Can we evaluate $x + y$?

A1: No, since $x + y$ is not an instance of either $n + 0$ or $n + s(m)$.

Q2: Can we symbolically evaluate $x + y$?

Symbolic Evaluation

Consider the equations [1] $n + 0 = n$, [2] $n + s(m) = s(n + m)$ defining natural number addition.

Q1: Can we evaluate $x + y$?

A1: No, since $x + y$ is not an instance of either $n + 0$ or $n + s(m)$.

Q2: Can we symbolically evaluate $x + y$?

A2: We could, if we could find most general instances of $x + y$ that can be evaluated in the standard sense.

Symbolic Evaluation

Consider the equations [1] $n + 0 = n$, [2] $n + s(m) = s(n + m)$ defining natural number addition.

Q1: Can we evaluate $x + y$?

A1: No, since $x + y$ is not an instance of either $n + 0$ or $n + s(m)$.

Q2: Can we symbolically evaluate $x + y$?

A2: We could, if we could find most general instances of $x + y$ that can be evaluated in the standard sense.

Q3: How do we find those most general instances of $x + y$?

Symbolic Evaluation

Consider the equations [1] $n + 0 = n$, [2] $n + s(m) = s(n + m)$ defining natural number addition.

Q1: Can we evaluate $x + y$?

A1: No, since $x + y$ is not an instance of either $n + 0$ or $n + s(m)$.

Q2: Can we symbolically evaluate $x + y$?

A2: We could, if we could find most general instances of $x + y$ that can be evaluated in the standard sense.

Q3: How do we find those most general instances of $x + y$?

A3: By unifying $x + y$ with the lefthand sides $n + 0$ and $n + s(m)$ equations [1], [2].

Symbolic Evaluation

Consider the equations [1] $n + 0 = n$, [2] $n + s(m) = s(n + m)$ defining natural number addition.

Q1: Can we evaluate $x + y$?

A1: No, since $x + y$ is not an instance of either $n + 0$ or $n + s(m)$.

Q2: Can we symbolically evaluate $x + y$?

A2: We could, if we could find most general instances of $x + y$ that can be evaluated in the standard sense.

Q3: How do we find those most general instances of $x + y$?

A3: By unifying $x + y$ with the lefthand sides $n + 0$ and $n + s(m)$ equations [1], [2]. This gives unifiers $\theta_1 = \{n \mapsto x, y \mapsto 0\}$, which evaluates to y with rule [1], and $\theta_2 = \{n \mapsto x, y \mapsto s(y'), m \mapsto y'\}$, which evaluates to $s(x + y')$ with rule [2].

Symbolic Evaluation

Consider the equations [1] $n + 0 = n$, [2] $n + s(m) = s(n + m)$ defining natural number addition.

Q1: Can we evaluate $x + y$?

A1: No, since $x + y$ is not an instance of either $n + 0$ or $n + s(m)$.

Q2: Can we symbolically evaluate $x + y$?

A2: We could, if we could find most general instances of $x + y$ that can be evaluated in the standard sense.

Q3: How do we find those most general instances of $x + y$?

A3: By unifying $x + y$ with the lefthand sides $n + 0$ and $n + s(m)$ equations [1], [2]. This gives unifiers $\theta_1 = \{n \mapsto x, y \mapsto 0\}$, which evaluates to y with rule [1], and $\theta_2 = \{n \mapsto x, y \mapsto s(y'), m \mapsto y'\}$, which evaluates to $s(x + y')$ with rule [2]. This is called narrowing.

Symbolic Evaluation = Narrowing

Narrowing generalizes rewriting:

Symbolic Evaluation = Narrowing

Narrowing generalizes rewriting:

- $(l \to r) \in R$ rewrites t to $t[r\theta]_p$, denoted $t \to_R t[r\theta]_p$, iff

Symbolic Evaluation = Narrowing

Narrowing generalizes rewriting:

- $(l \to r) \in R$ rewrites t to $t[r\theta]_p$, denoted $t \to_R t[r\theta]_p$, iff there is a position p and a matching substitution θ such that $t|_p = l\theta$.

Symbolic Evaluation = Narrowing

Narrowing generalizes rewriting:

- $(l \to r) \in R$ rewrites t to $t[r\theta]_p$, denoted $t \to_R t[r\theta]_p$, iff there is a position p and a matching substitution θ such that $t|_p = l\theta$.
- Rewriting generalizes to narrowing by replacing the matching substitution θ s.t. $t|_p = l\theta$ by a unifying substitution θ s.t. $\theta \in \mathit{Unif}(t|_p = l)$.

Symbolic Evaluation = Narrowing

Narrowing generalizes rewriting:

- $(l \to r) \in R$ rewrites t to $t[r\theta]_p$, denoted $t \to_R t[r\theta]_p$, iff there is a position p and a matching substitution θ such that $t|_p = l\theta$.
- Rewriting generalizes to narrowing by replacing the matching substitution θ s.t. $t|_p = l\theta$ by a unifying substitution θ s.t. $\theta \in \mathit{Unif}(t|_p = l)$. Here is the precise definition:

Symbolic Evaluation = Narrowing

Narrowing generalizes rewriting:

- $(l \to r) \in R$ rewrites t to $t[r\theta]_p$, denoted $t \to_R t[r\theta]_p$, iff there is a position p and a matching substitution θ such that $t|_p = l\theta$.
- Rewriting generalizes to narrowing by replacing the matching substitution θ s.t. $t|_p = l\theta$ by a unifying substitution θ s.t. $\theta \in \mathit{Unif}(t|_p = l)$. Here is the precise definition:

Definition. $(l \to r) \in R$ narrows t to $t[r]_p\theta$ iff

Symbolic Evaluation = Narrowing

Narrowing generalizes rewriting:

- $(l \to r) \in R$ rewrites t to $t[r\theta]_p$, denoted $t \to_R t[r\theta]_p$, iff there is a position p and a matching substitution θ such that $t|_p = l\theta$.
- Rewriting generalizes to narrowing by replacing the matching substitution θ s.t. $t|_p = l\theta$ by a unifying substitution θ s.t. $\theta \in \mathit{Unif}(t|_p = l)$. Here is the precise definition:

Definition. $(l \to r) \in R$ narrows t to $t[r]_p\theta$ iff there is a non-variable position p of t (i.e., $t|_p \neq \mathit{vars}(t|_p)$), and a unifier $\theta \in \mathit{Unif}(t|_p = l)$.

Symbolic Evaluation = Narrowing

Narrowing generalizes rewriting:

- $(l \to r) \in R$ rewrites t to $t[r\theta]_p$, denoted $t \to_R t[r\theta]_p$, iff there is a position p and a matching substitution θ such that $t|_p = l\theta$.
- Rewriting generalizes to narrowing by replacing the matching substitution θ s.t. $t|_p = l\theta$ by a unifying substitution θ s.t. $\theta \in Unif(t|_p = l)$. Here is the precise definition:

Definition. $(l \to r) \in R$ narrows t to $t[r]_p\theta$ iff there is a non-variable position p of t (i.e., $t|_p \neq vars(t|_p)$), and a unifier $\theta \in Unif(t|_p = l)$. This defines the narrowing relation, denoted:

$$t \overset{\theta}{\rightsquigarrow}_R t[r]_p\theta$$

Symbolic Evaluation = Narrowing

Narrowing generalizes rewriting:

- $(l \to r) \in R$ rewrites t to $t[r\theta]_p$, denoted $t \to_R t[r\theta]_p$, iff there is a position p and a matching substitution θ such that $t|_p = l\theta$.
- Rewriting generalizes to narrowing by replacing the matching substitution θ s.t. $t|_p = l\theta$ by a unifying substitution θ s.t. $\theta \in Unif(t|_p = l)$. Here is the precise definition:

Definition. $(l \to r) \in R$ narrows t to $t[r]_p\theta$ iff there is a non-variable position p of t (i.e., $t|_p \neq vars(t|_p)$), and a unifier $\theta \in Unif(t|_p = l)$. This defines the narrowing relation, denoted:

$$t \overset{\theta}{\rightsquigarrow}_R t[r]_p\theta$$

Note that if $t \in T_\Sigma$, $t \to_R t[r\theta]_p$ iff $t \overset{\theta}{\rightsquigarrow}_R t[r]_p\theta$. I.e., narrowing and rewriting coincide for ground terms.

Symbolic Evaluation = Narrowing (II)

A narrowing step is a symbolic evaluation step. That is,

Symbolic Evaluation = Narrowing (II)

A narrowing step is a symbolic evaluation step. That is,

Symbolic Evaluation = Narrowing

Symbolic Evaluation = Narrowing (II)

A narrowing step is a symbolic evaluation step. That is,

Symbolic Evaluation = Narrowing

In our example two such narrowing steps symbolically evaluate $x + y$ with rules [1] and [2]:

Symbolic Evaluation = Narrowing (II)

A narrowing step is a symbolic evaluation step. That is,

Symbolic Evaluation = Narrowing

In our example two such narrowing steps symbolically evaluate $x + y$ with rules [1] and [2]: $x + y \overset{\theta_1}{\rightsquigarrow} x$

Symbolic Evaluation = Narrowing (II)

A narrowing step is a symbolic evaluation step. That is,

Symbolic Evaluation = Narrowing

In our example two such narrowing steps symbolically evaluate $x + y$ with rules [1] and [2]: $x + y \overset{\theta_1}{\rightsquigarrow} x$ and $x + y \overset{\theta_2}{\rightsquigarrow} s(x + y')$.

Symbolic Evaluation = Narrowing (II)

A narrowing step is a symbolic evaluation step. That is,

Symbolic Evaluation = Narrowing

In our example two such narrowing steps symbolically evaluate $x + y$ with rules [1] and [2]: $x + y \overset{\theta_1}{\leadsto} x$ and $x + y \overset{\theta_2}{\leadsto} s(x + y')$.

As for rewriting, we have the reflexive-transitive closure $t \overset{\theta}{\leadsto}{}^{*}_{R} v$, where for 0 steps we get $\theta = id$ and $v = t$, and for $n + 1$ steps we get a sequence:

$$t \overset{\theta_1}{\leadsto}_R t_1 \ldots t_n \overset{\theta_{n+1}}{\leadsto}_R t_{n+1}$$

with $v = t_{n+1}$ and θ the composed substitution $\theta = \theta_1 \ldots \theta_{n+1}$.

Symbolic Evaluation = Narrowing (II)

A narrowing step is a symbolic evaluation step. That is,

Symbolic Evaluation = Narrowing

In our example two such narrowing steps symbolically evaluate $x + y$ with rules [1] and [2]: $x + y \overset{\theta_1}{\leadsto} x$ and $x + y \overset{\theta_2}{\leadsto} s(x + y')$.

As for rewriting, we have the reflexive-transitive closure $t \overset{\theta}{\leadsto}{}^{*}_{R} v$, where for 0 steps we get $\theta = id$ and $v = t$, and for $n + 1$ steps we get a sequence:

$$t \overset{\theta_1}{\leadsto}_R t_1 \ldots t_n \overset{\theta_{n+1}}{\leadsto}_R t_{n+1}$$

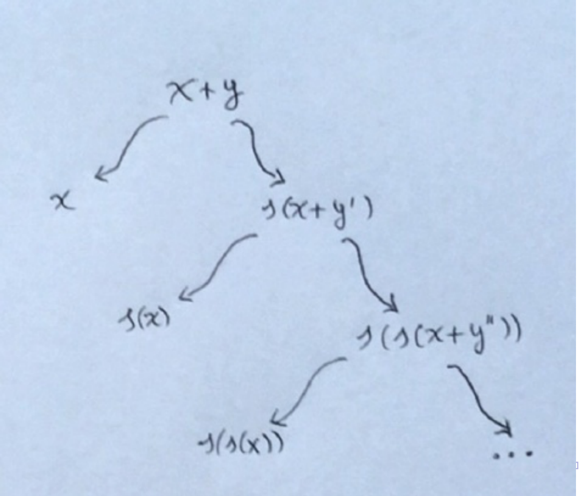
with $v = t_{n+1}$ and θ the composed substitution $\theta = \theta_1 \ldots \theta_{n+1}$. To avoid variable capture, we always assume that rules in R are variable renamed so that they do not share any variables with any of the terms t_i; and that for each unifier θ_i, $1 \leq i \leq n + 1$, the variables in $rng(\theta_i) = \{y \in X \mid \exists x \in dom(\theta_i)\ s.t.\ y \in vars(\theta_i(x)\}$, are fresh (i.e., never used before).

The Narrowing Tree

Symbolic computations in such sequences $t \overset{\theta}{\rightsquigarrow}^{*}_{R} v$ from an initial term t are the paths in the so-called narrowing tree of t. E.g.,

The Narrowing Tree

Symbolic computations in such sequences $t \overset{\theta}{\leadsto}{}^{*}_{R} v$ from an initial term t are the paths in the so-called narrowing tree of t. E.g.,



The Lifting Lemma

Symbolic computation by narrowing covers all rewriting computations as instances as shown below (proof in Appendix):

Theorem (Lifting Lemma). Let (Σ, R) be a term rewriting system, $t \in T_{\Sigma}(X)$, and θ an R-irreducible substitution (i.e., if $x \in dom(\theta)$, then $\theta(x)$ cannot be rewritten with R).

The Lifting Lemma

Symbolic computation by narrowing covers all rewriting computations as instances as shown below (proof in Appendix):

Theorem (Lifting Lemma). Let (Σ, R) be a term rewriting system, $t \in T_{\Sigma}(X)$, and θ an R-irreducible substitution (i.e., if $x \in dom(\theta)$, then $\theta(x)$ cannot be rewritten with R). Then for each rewrite step $t\theta \rightarrow_R u$ there is a narrowing step $t \overset{\alpha}{\rightsquigarrow}_R v$ and an R-irreducible substitution δ such that $v\delta = u$.

The Lifting Lemma

Symbolic computation by narrowing covers all rewriting computations as instances as shown below (proof in Appendix):

Theorem (Lifting Lemma). Let (Σ, R) be a term rewriting system, $t \in T_{\Sigma}(X)$, and θ an R-irreducible substitution (i.e., if $x \in dom(\theta)$, then $\theta(x)$ cannot be rewritten with R). Then for each rewrite step $t\theta \rightarrow_R u$ there is a narrowing step $t \overset{\alpha}{\leadsto}_R v$ and an R-irreducible substitution δ such that $v\delta = u$.

Since each narrowing step in the Lifting Lemma preserves the invariant that the substitution θ for t, resp. γ for v, is R-irreducible,

The Lifting Lemma

Symbolic computation by narrowing covers all rewriting computations as instances as shown below (proof in Appendix):

Theorem (Lifting Lemma). Let (Σ, R) be a term rewriting system, $t \in T_\Sigma(X)$, and θ an R-irreducible substitution (i.e., if $x \in dom(\theta)$, then $\theta(x)$ cannot be rewritten with R). Then for each rewrite step $t\theta \to_R u$ there is a narrowing step $t \overset{\alpha}{\leadsto}_R v$ and an R-irreducible substitution δ such that $v\delta = u$.

Since each narrowing step in the Lifting Lemma preserves the invariant that the substitution θ for t, resp. γ for v, is R-irreducible, the Lifting Lemma extends in a straightforward manner to R-rewriting sequences of the form $t\theta \to_R^* w$ with θ an R-irreducible, which are indeed all covered as instances by narrowing sequences $t \overset{\theta_1}{\leadsto}_R t_1 \dots t_n \overset{\theta_{n+1}}{\leadsto}_R t_{n+1}$, with $w = t_{n+1}\delta$.

Narrowing Modulo B

The same way that rewriting with R extends to rewriting modulo axioms B, narrowing extends in a completely smilar way. Here is the precise definition (including the case $B = \emptyset$ as a special case):

Narrowing Modulo B

The same way that rewriting with R extends to rewriting modulo axioms B, narrowing extends in a completely smilar way. Here is the precise definition (including the case $B = \emptyset$ as a special case):

Definition. Given a rewrite theory (Σ, B, R), and a term $t \in T_\Sigma(X)$, an R-narrowing step modulo B, denoted $t \leadsto^{\theta}_{R,B} v$ holds iff there exists a non-variable position p in t, a rule $l \to r$ in R, and a B-unifier $\theta \in Unif_B(t|_p = l)$ such that $v = t[r]_p\theta$.

Narrowing Modulo B

The same way that rewriting with R extends to rewriting modulo axioms B, narrowing extends in a completely smilar way. Here is the precise definition (including the case $B = \emptyset$ as a special case):

Definition. Given a rewrite theory (Σ, B, R), and a term $t \in T_{\Sigma}(X)$, an R-narrowing step modulo B, denoted $t \leadsto^{\theta}_{R,B} v$ holds iff there exists a non-variable position p in t, a rule $l \to r$ in R, and a B-unifier $\theta \in Unif_B(t|_p = l)$ such that $v = t[r]_p\theta$.

In particular, the Lifting Lemma extends in a natural way to narrowing steps and narrowing sequences modulo B, so that all R/B-rewriting computations $t\theta \to^{*}_{R/B} w$ are covered as instances.

Narrowing Modulo B

The same way that rewriting with R extends to rewriting modulo axioms B, narrowing extends in a completely smilar way. Here is the precise definition (including the case $B = \emptyset$ as a special case):

Definition. Given a rewrite theory (Σ, B, R), and a term $t \in T_{\Sigma}(X)$, an R-narrowing step modulo B, denoted $t \leadsto^{\theta}_{R,B} v$ holds iff there exists a non-variable position p in t, a rule $l \to r$ in R, and a B-unifier $\theta \in Unif_B(t|_p = l)$ such that $v = t[r]_p\theta$.

In particular, the Lifting Lemma extends in a natural way to narrowing steps and narrowing sequences modulo B, so that all R/B-rewriting computations $t\theta \to^{*}_{R/B} w$ are covered as instances.

A small technicality is that we should narrow t not just with R, but with all its B-extensions, which for R/B-rewriting is done automatically by Maude (see §4.8 in "All About Maude").

Topmost Rewrite Theories

Call a rewrite theory $\mathcal{R} = (\Sigma, E \cup B, R)$ topmost if it has a sort *State*, which is the top sort of one of its connected components, such that:

Topmost Rewrite Theories

Call a rewrite theory $\mathcal{R} = (\Sigma, E \cup B, R)$ topmost if it has a sort *State*, which is the top sort of one of its connected components, such that: (i) no Σ-term $f(u_1, \ldots, u_n)$ can have a proper subterm of sort *State*; and

Topmost Rewrite Theories

Call a rewrite theory $\mathcal{R} = (\Sigma, E \cup B, R)$ topmost if it has a sort *State*, which is the top sort of one of its connected components, such that: (i) no Σ-term $f(u_1, \ldots, u_n)$ can have a proper subterm of sort *State*; and (ii) for all rules $l \to r$ in R, l (and therefore r) has sort *State*.

Topmost Rewrite Theories

Call a rewrite theory $\mathcal{R} = (\Sigma, E \cup B, R)$ topmost if it has a sort *State*, which is the top sort of one of its connected components, such that: (i) no Σ-term $f(u_1, \ldots, u_n)$ can have a proper subterm of sort *State*; and (ii) for all rules $l \to r$ in R, l (and therefore r) has sort *State*. As we shall see shortly, topmost rewrite theories are very useful for narrowing-based symbolic model checking.

Topmost Rewrite Theories

Call a rewrite theory $\mathcal{R} = (\Sigma, E \cup B, R)$ topmost if it has a sort *State*, which is the top sort of one of its connected components, such that: (i) no Σ-term $f(u_1, \ldots, u_n)$ can have a proper subterm of sort *State*; and (ii) for all rules $l \to r$ in R, l (and therefore r) has sort *State*. As we shall see shortly, topmost rewrite theories are very useful for narrowing-based symbolic model checking.

Many rewrite theories can be easily transformed into semantically equivalent topmost ones.

Topmost Rewrite Theories

Call a rewrite theory $\mathcal{R} = (\Sigma, E \cup B, R)$ topmost if it has a sort *State*, which is the top sort of one of its connected components, such that: (i) no Σ-term $f(u_1, \ldots, u_n)$ can have a proper subterm of sort *State*; and (ii) for all rules $l \to r$ in R, l (and therefore r) has sort *State*. As we shall see shortly, topmost rewrite theories are very useful for narrowing-based symbolic model checking.

Many rewrite theories can be easily transformed into semantically equivalent topmost ones. For example, if $\mathcal{R}$ specifies a concurrent object system, we can just add a new sort *State* and a constructor $\{\_\} : Configuration \to State$ and convert, for example, a rule $credit(O, M)\ \langle O : Accnt \mid bal : N \rangle \to \langle O : Accnt \mid bal : N + M \rangle$ into the semantically equivalent rule:

Topmost Rewrite Theories

Call a rewrite theory $\mathcal{R} = (\Sigma, E \cup B, R)$ topmost if it has a sort *State*, which is the top sort of one of its connected components, such that: (i) no Σ-term $f(u_1, \ldots, u_n)$ can have a proper subterm of sort *State*; and (ii) for all rules $l \to r$ in R, l (and therefore r) has sort *State*. As we shall see shortly, topmost rewrite theories are very useful for narrowing-based symbolic model checking.

Many rewrite theories can be easily transformed into semantically equivalent topmost ones. For example, if $\mathcal{R}$ specifies a concurrent object system, we can just add a new sort *State* and a constructor $\{\_\} : Configuration \to State$ and convert, for example, a rule $credit(O, M)\ \langle O : Accnt | bal : N \rangle \to \langle O : Accnt | bal : N + M \rangle$ into the semantically equivalent rule: $\{credit(O, M)\ \langle O : Accnt | bal : N \rangle\ C\} \to \langle O : Accnt | bal : N + M \rangle\ C\}$, with C of sort *Configuration*.

Symbolic Model Checking of Topmost Rewrite Theories

Given a topmost rewrite theory $\mathcal{R} = (\Sigma, B, R)$, where the number of reachable states from a given initial state may be infinite, narrowing with R modulo axioms B supports the following symbolic reachability analysis result:

Symbolic Model Checking of Topmost Rewrite Theories

Given a topmost rewrite theory $\mathcal{R} = (\Sigma, B, R)$, where the number of reachable states from a given initial state may be infinite, narrowing with R modulo axioms B supports the following symbolic reachability analysis result:

Theorem (Completeness of Narrowing Search). For $\mathcal{R} = (\Sigma, B, R)$ topmost, t a non-variable term of sort *State* with variables $\vec{x}$, and u a term of sort *State* with variables $\vec{y}$, the FOL existential formula:

Symbolic Model Checking of Topmost Rewrite Theories

Given a topmost rewrite theory $\mathcal{R} = (\Sigma, B, R)$, where the number of reachable states from a given initial state may be infinite, narrowing with R modulo axioms B supports the following symbolic reachability analysis result:

Theorem (Completeness of Narrowing Search). For $\mathcal{R} = (\Sigma, B, R)$ topmost, t a non-variable term of sort *State* with variables $\vec{x}$, and u a term of sort *State* with variables $\vec{y}$, the FOL existential formula:

$$\exists \vec{x}, \vec{y},\ t \rightarrow^* u$$

with $\vec{x} \cap \vec{y} = \emptyset$ is satisfied in $\mathbb{C}_{\mathcal{R}}$ iff

Symbolic Model Checking of Topmost Rewrite Theories

Given a topmost rewrite theory $\mathcal{R} = (\Sigma, B, R)$, where the number of reachable states from a given initial state may be infinite, narrowing with R modulo axioms B supports the following symbolic reachability analysis result:

Theorem (Completeness of Narrowing Search). For $\mathcal{R} = (\Sigma, B, R)$ topmost, t a non-variable term of sort *State* with variables $\vec{x}$, and u a term of sort *State* with variables $\vec{y}$, the FOL existential formula:

$$\exists \vec{x}, \vec{y},\ t \to^{*} u$$

with $\vec{x} \cap \vec{y} = \emptyset$ is satisfied in $\mathbb{C}_{\mathcal{R}}$ iff there is an R, B-narrowing sequence $t \overset{\theta}{\leadsto}{}^{*}_{R,B}\ v$ and a B-unifier $\gamma \in \mathit{Unif}_B(u = v)$.

Symbolic Model Checking of Topmost Rewrite Theories

Given a topmost rewrite theory $\mathcal{R} = (\Sigma, B, R)$, where the number of reachable states from a given initial state may be infinite, narrowing with R modulo axioms B supports the following symbolic reachability analysis result:

Theorem (Completeness of Narrowing Search). For $\mathcal{R} = (\Sigma, B, R)$ topmost, t a non-variable term of sort *State* with variables $\vec{x}$, and u a term of sort *State* with variables $\vec{y}$, the FOL existential formula:

$$\exists \vec{x}, \vec{y},\ t \rightarrow^* u$$

with $\vec{x} \cap \vec{y} = \emptyset$ is satisfied in $\mathbb{C}_{\mathcal{R}}$ iff there is an R, B-narrowing sequence $t \overset{\theta}{\rightsquigarrow}^*_{R,B} v$ and a B-unifier $\gamma \in \mathit{Unif}_B(u = v)$.

The proof is a simple application of the Lifting Lemma and is left as an exercise.

Symbolic Verification of Invariants by Narrowing

Breadth-first search with the rewriting relation $\to_{R/B}$ gives us a semi-decision procedure for verifying invariant failure from a concrete initial state by searching for a counterexample.

Symbolic Verification of Invariants by Narrowing

Breadth-first search with the rewriting relation $\rightarrow_{R/B}$ gives us a semi-decision procedure for verifying invariant failure from a concrete initial state by searching for a counterexample.

Likewise, thanks to the Completeness of Narrowing Theorem, breadth-first search with the narrowing relation $\leadsto_{R,B}$ gives us a semi-decision procedure for verifying invariant failure from a symbolic initial state (i.e., a term t of sort *State* with variables or, more generally, a finite set $\{t_1, \ldots, t_n\}$ of terms with variables) by searching for a symbolic counterexample, provided $\mathcal{R} = (\Sigma, B, R)$ is topmost.

Symbolic Verification of Invariants by Narrowing

Breadth-first search with the rewriting relation $\rightarrow_{R/B}$ gives us a semi-decision procedure for verifying invariant failure from a concrete initial state by searching for a counterexample.

Likewise, thanks to the Completeness of Narrowing Theorem, breadth-first search with the narrowing relation $\rightsquigarrow_{R,B}$ gives us a semi-decision procedure for verifying invariant failure from a symbolic initial state (i.e., a term t of sort *State* with variables or, more generally, a finite set $\{t_1, \ldots, t_n\}$ of terms with variables) by searching for a symbolic counterexample, provided $\mathcal{R} = (\Sigma, B, R)$ is topmost.

The only requirement is that the negation of the invariant (i.e., its complement) can be expressed as a term u with variables, or, more generally, as a finite set $\{u_1, \ldots, u_m\}$ of terms with variables.

Narrowing with Folding May Terminate

Just as for the `search` command, a narrowing search may not terminate.

Narrowing with Folding May Terminate

Just as for the `search` command, a narrowing search may not terminate. However, Maude supports a `{fold} vu-narrow` narrowing search command option that tries to fold the usually infinite narrowing search tree into a hopefully finite narrowing search graph, by not exploring tree nodes that are substitution instances modulo B of more general, previously explored nodes.

Narrowing with Folding May Terminate

Just as for the `search` command, a narrowing search may not terminate. However, Maude supports a `{fold} vu-narrow` narrowing search command option that tries to fold the usually infinite narrowing search tree into a hopefully finite narrowing search graph, by not exploring tree nodes that are substitution instances modulo B of more general, previously explored nodes.

Definition. Term u, reached by narrowing at depth d, is folded into another term v reached by narrowing at depth $\leq d$ (i.e., u is no further explored) if there is a substitution α s.t. $u =_B v\alpha$.

Narrowing with Folding May Terminate

Just as for the `search` command, a narrowing search may not terminate. However, Maude supports a `{fold} vu-narrow` narrowing search command option that tries to fold the usually infinite narrowing search tree into a hopefully finite narrowing search graph, by not exploring tree nodes that are substitution instances modulo B of more general, previously explored nodes.

Definition. Term u, reached by narrowing at depth d, is folded into another term v reached by narrowing at depth $\leq d$ (i.e., u is no further explored) if there is a substitution α s.t. $u =_B v\alpha$.

Folding can make the search graph finite, allowing termination of narrowing seach and making verification of an invariant decidable.

Narrowing with Folding May Terminate

Just as for the `search` command, a narrowing search may not terminate. However, Maude supports a `{fold} vu-narrow` narrowing search command option that tries to fold the usually infinite narrowing search tree into a hopefully finite narrowing search graph, by not exploring tree nodes that are substitution instances modulo B of more general, previously explored nodes.

Definition. Term u, reached by narrowing at depth d, is folded into another term v reached by narrowing at depth $\leq d$ (i.e., u is no further explored) if there is a substitution α s.t. $u =_B v\alpha$.

Folding can make the search graph finite, allowing termination of narrowing seach and making verification of an invariant decidable.

Lets us see an example. Consider the following Maude specification of Lamport's bakery protocol:

Lamport's Bakery Protocol

```
mod BAKERY is sorts Nat LNat Nat? State WProcs Procs .
  subsorts Nat LNat < Nat? .  subsort WProcs < Procs .
  op 0 : -> Nat .                    op s : Nat -> Nat .
  op [_] : Nat -> LNat .                     *** number-locking operator
  op < wait,_> : Nat -> WProcs .    op < crit,_> : Nat -> Procs .
  op mt : -> WProcs .                 *** empty multiset
  op __ : Procs Procs -> Procs [assoc comm id: mt] .    *** union
  op __ : WProcs WProcs -> WProcs [assoc comm id: mt] . *** union
  op _|_|_ : Nat Nat? Procs -> State .
  vars n m i j k : Nat . var x? : Nat? . var PS : Procs . var WPS : WProcs .
  rl [new]:  m | n | PS => s(m) | n | < wait,m > PS [narrowing] .
  rl [enter]: m | n | < wait,n > PS => m | [n] | < crit,n > PS [narrowing] .
  rl [leave]: m | [n] | < crit,n > PS => m | s(n) | PS [narrowing] .
endm
```

States have the form "`m | x? | PS`" with `m` the ticket counter, `x?` the counter to access the critical section, and `PS` a multiset of processes. `BAKERY` is infinite-state because of `[new]`. When a waiting process `n` enters the critical section, the second counter `n` is locked as `[n]`; and it is unlocked and incremented when `n` leaves.

Lamport's Bakery Protocol (II)

The key invariant is mutual exclusion. Its complement is specified by the term `i | x? | < crit, j > < crit, k > PS`.

Lamport's Bakery Protocol (II)

The key invariant is mutual exclusion. Its complement is specified by the term `i | x? | < crit, j > < crit, k > PS`.

Without the `fold` option, narrowing search does not terminate. But with the following folding search command we can verify that `BAKERY` satisfies mutual exclusion, not just for the initial state `0 | 0 | mt`, but for the more general infinite set of initial states, having only waiting processes, `m | n | WPS`.

Lamport's Bakery Protocol (II)

The key invariant is mutual exclusion. Its complement is specified by the term `i | x? | < crit, j > < crit, k > PS`.

Without the `fold` option, narrowing search does not terminate. But with the following folding search command we can verify that `BAKERY` satisfies mutual exclusion, not just for the initial state `0 | 0 | mt`, but for the more general infinite set of initial states, having only waiting processes, `m | n | WPS`.

```
Maude> {fold} vu-narrow
        m | n | WPS =>* i | x? | < crit, j > < crit, k > PS .

No solution.
```

Lamport's Bakery Protocol (II)

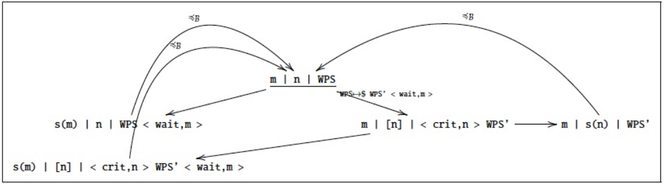
The key invariant is mutual exclusion. Its complement is specified by the term `i | x? | < crit, j > < crit, k > PS`.

Without the `fold` option, narrowing search does not terminate. But with the following folding search command we can verify that `BAKERY` satisfies mutual exclusion, not just for the initial state `0 | 0 | mt`, but for the more general infinite set of initial states, having only waiting processes, `m | n | WPS`.

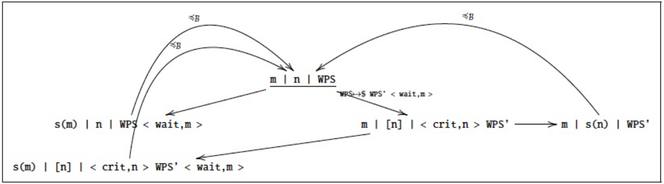
```
Maude> {fold} vu-narrow
        m | n | WPS =>* i | x? | < crit, j > < crit, k > PS .

No solution.
```

We can visualize the dramatic state space reduction obtained by folding an infinite tree of symbolic states into a finite graph with only four states in the figure below.

Lamport's Bakery Protocol (III)

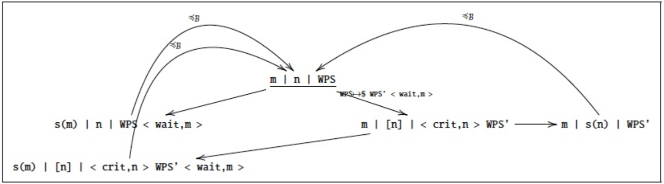


Lamport's Bakery Protocol (III)



A somewhat counterintuitive lesson that can be learned from this example and its initial state `m | n | WPS` is that, for narrowing model checking, the more general the initial state, the better.

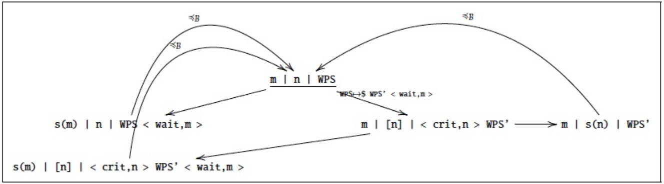
Lamport's Bakery Protocol (III)



A somewhat counterintuitive lesson that can be learned from this example and its initial state `m | n | WPS` is that, for narrowing model checking, the more general the initial state, the better.

The reason is that, if we start with a quite specific initial state, the subsequent symbolic states will be even more specific.

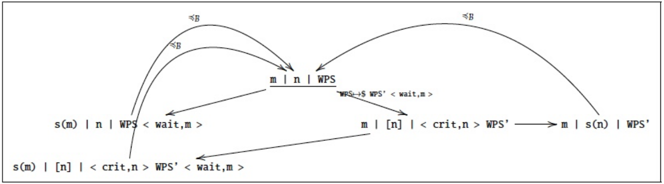
Lamport's Bakery Protocol (III)



A somewhat counterintuitive lesson that can be learned from this example and its initial state `m | n | WPS` is that, for narrowing model checking, the more general the initial state, the better.

The reason is that, if we start with a quite specific initial state, the subsequent symbolic states will be even more specific. This is what the word "narrowing" means.

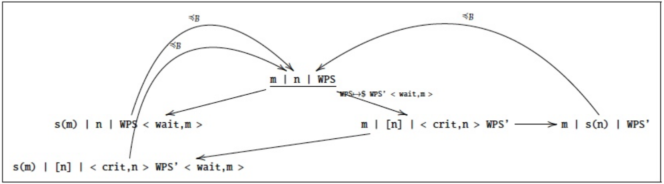
Lamport's Bakery Protocol (III)



A somewhat counterintuitive lesson that can be learned from this example and its initial state `m | n | WPS` is that, for narrowing model checking, the more general the initial state, the better.

The reason is that, if we start with a quite specific initial state, the subsequent symbolic states will be even more specific. This is what the word "narrowing" means. The more specific a state is, the less it can generalize other symbolic states by folding them.

Lamport's Bakery Protocol (III)



A somewhat counterintuitive lesson that can be learned from this example and its initial state `m | n | WPS` is that, for narrowing model checking, the more general the initial state, the better.

The reason is that, if we start with a quite specific initial state, the subsequent symbolic states will be even more specific. This is what the word "narrowing" means. The more specific a state is, the less it can generalize other symbolic states by folding them. Worse of all are ground initial states like `0 | 0 | mt`, which turn narrowing search into rewriting search and make generalization impossible.

Initial States as Disjunctions of Patterns

Since the more general the initial state, the better, an important way of achieving such generality, so as to increase the chances that variant narrowing search terminates, is to allow initial states to be a disjunction of patterns. Indeed, Maude 3.5 allows narrowing search commands of the form:

Initial States as Disjunctions of Patterns

Since the more general the initial state, the better, an important way of achieving such generality, so as to increase the chances that variant narrowing search terminates, is to allow initial states to be a disjunction of patterns. Indeed, Maude 3.5 allows narrowing search commands of the form:

{fold} vu-narrow $u_1 \vee \ldots \vee u_n$ =>* v

Initial States as Disjunctions of Patterns

Since the more general the initial state, the better, an important way of achieving such generality, so as to increase the chances that variant narrowing search terminates, is to allow initial states to be a disjunction of patterns. Indeed, Maude 3.5 allows narrowing search commands of the form:

`{fold} vu-narrow` $u_1 \vee \ldots \vee u_n$ `=>*` v

This means that we can try to solve existential reachability formulas of the form $\exists \vec{x}, \vec{y},\ u_1 \vee \ldots \vee u_n \rightarrow^* v_1 \vee \ldots \vee v_m$ by means of m `{fold} vu-narrow` commands of the above form.

Initial States as Disjunctions of Patterns

Since the more general the initial state, the better, an important way of achieving such generality, so as to increase the chances that variant narrowing search terminates, is to allow initial states to be a disjunction of patterns. Indeed, Maude 3.5 allows narrowing search commands of the form:

`{fold} vu-narrow` $u_1 \vee \ldots \vee u_n$ `=>*` v

This means that we can try to solve existential reachability formulas of the form $\exists \vec{x}, \vec{y},\ u_1 \vee \ldots \vee u_n \rightarrow^* v_1 \vee \ldots \vee v_m$ by means of m `{fold} vu-narrow` commands of the above form.

We shall further explore the advantages of this greater generality for specifying initial states in Lecture 24.