Program Verification: Lecture 20

José Meseguer

Computer Science Department
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.

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 = Narrowing

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 method is called narrowing. It generalizes rewriting, where $l \to r$ rewrites t if there is a position p and a substitution θ such that $t|_p = l\theta$, and then $t \to t[r\theta]_p$, by replacing the matching condition $t|_p = l\theta$ by a unification condition $\theta \in \mathit{Unif}(t|_p = l)$. Then we get a symbolic evaluation step, called narrowing, and denoted:

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

In our example we get $x + y \overset{\theta_1}{\rightsquigarrow} x$ and $x + y \overset{\theta_2}{\rightsquigarrow} s(x + y')$.

More on Narrowing

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

$$t \overset{\theta_1}{\rightsquigarrow}_R t_1 \dots t_n \overset{\theta_{n+1}}{\rightsquigarrow}_R t_{n+1}$$

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

Symbolic computations in such sequences $t \overset{\theta}{\rightsquigarrow}^{*}_{R} v$ from a common t are the paths in the so-called narrowing tree of t (see Figure 1).

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 $v\delta = u$.

Note that, since each narrowing step in the Lifting Lemma preserves the invariant that the substitution θ for t, resp. γ for v, is R-irreducible, this lemma extends in a straightforward manner to narrowing sequences $t \overset{\theta_1}{\leadsto}_R t_1 \dots t_n \overset{\theta_{n+1}}{\leadsto}_R t_{n+1}$, which do indeed cover all R-rewriting computations $t\theta \to_R^* w$ 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):

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 $\theta \in \mathit{Unif}_B(t|_p = l)$ 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: (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 $\{\_\} : \mathit{Configuration} \to \mathit{State}$ and convert, for example, a rule $\mathit{credit}(O, M)\ \langle O : \mathit{Accnt} | \mathit{bal} : N \rangle \to \langle O : \mathit{Accnt} | \mathit{bal} : N + M \rangle$ into the semantically equivalent rule:
$\{\mathit{credit}(O, M)\ \langle O : \mathit{Accnt} | \mathit{bal} : N \rangle\ C\} \to \langle O : \mathit{Accnt} | \mathit{bal} : N{+}M \rangle\ C\}$, 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:

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:

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

is satisfied in $\mathbb{C}_{\mathcal{R}}$ iff there is an R, B-narrowing sequence $t \overset{\theta}{\rightsquigarrow}{}^{*}_{R,B}\ v$ such that there is 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

The same way that 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, 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 (a term t of sort *State* 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 can be expressed in the cool way, as a term u with variables, or, more generally, as a finite set $\{u_1, \ldots, u_n\}$ of terms with variables.

Symbolic Verification of Invariants by Narrowing (II)

Just as for the `search` command, the narrowing search may not terminate. However, Maude supports a `{fold} vu-narrow` narrowing search command that tries to fold the infinite narrowing search tree into a hopefully finite narrowing search graph, by not exploring tree nodes that are substitution instances modulo B of previously explored nodes. In practice this makes the search finite, allowing full verification of the invariant, in significant examples.

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
```

The states of `BAKERY` have the form “`m | x? | PS`” with `m` the ticket-dispensing counter, `x?` the (possibly locked) counter to access the critical section, and `PS` a multiset of processes either waiting or in the critical section. `BAKERY` is infinite-state: `[new]` creates new processes, and the counters can grow unboundedly. When a waiting process enters the critical section with `[enter]`, the second counter `n` is locked as `[n]`; and it is unlocked and incremented when it leaves it with `[leave]`. The key invariant is mutual exclusion. Note that the term “`i | x? | < crit, j > < crit, k > PS`” describes all states in the complement of the invariant. of mutual exclusion states.

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

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

No solution.
rewrites: 4 in 1ms cpu (1ms real) (2677 rewrites/second)
```

We can visualize the dramatic state space reduction from an infinite tree of symbolic states to a finite graph with only four states in Figure 2.

A somewhat counterintuitive lesson that we can learn from this example and the very general initial state `m | n | WPS` is that for symbolic 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. But such quite specific states will often lack the capacity to generalize other symbolic states by folding.

In particular, if we had started with a ground state like `0 | 0 | mt`, since for ground terms narrowing coincides with rewriting, we would in fact be performing Maude's standard `search` command, and would have lost all chances of obtaining a finite graph by folding.