

Program Verification: Lecture 19

José Meseguer

Computer Science Department
University of Illinois at Urbana-Champaign

More on Reachability Homomorphisms

Recall that given two (Σ, ϕ) -reachability models $\mathcal{A}_{\rightarrow} = (\mathcal{A}, \rightarrow_{\mathcal{A}})$, and $\mathcal{B}_{\rightarrow} = (\mathcal{B}, \rightarrow_{\mathcal{B}})$, a (Σ, ϕ) -**reachability homomorphism** $h : \mathcal{A}_{\rightarrow} \rightarrow \mathcal{B}_{\rightarrow}$ is a Σ -homomorphism $h : \mathcal{A} \rightarrow \mathcal{B}$ such that “preserves reachability,” that is, for each $k \in K$, $a \rightarrow_{\mathcal{A},k} a'$ implies $h_k(a) \rightarrow_{\mathcal{B},k} h_k(a')$.

We call a reachability homomorphism $h : \mathcal{A}_{\rightarrow} \rightarrow \mathcal{B}_{\rightarrow}$ a (stuttering) **bisimulation** if, in addition, for each $k \in K$, and each $a \in A_k$, $h_k(a) \rightarrow_{\mathcal{B},k} b'$ implies that there exists $a' \in A_k$ such that: (i) $h_k(a') = b'$, and (ii) $a \rightarrow_{\mathcal{A},k} a'$. That is, a bisimulation preserves reachability “in both directions.”

More on Reachability Homomorphisms (II)

Recall that, given any (Σ, ϕ) -reachability model $\mathcal{A}_{\rightarrow} = (\mathcal{A}, \rightarrow_{\mathcal{A}})$ a kind k and an element $a \in A_k$, we defined $Reach_{\mathcal{A}_{\rightarrow}}(a) = \{x \in A_k \mid a \rightarrow_{\mathcal{A}} x\}$.

The following lemma is left as an easy exercise.

Lemma 1. If $h : \mathcal{A}_{\rightarrow} \longrightarrow \mathcal{B}_{\rightarrow}$ is a reachability homomorphism (resp. a bisimulation) then for each kind k and an element $a \in A_k$ we have $h_k[Reach_{\mathcal{A}_{\rightarrow}}(a)] \subseteq Reach_{\mathcal{B}_{\rightarrow}}(h(a))$ (resp. $h_k[Reach_{\mathcal{A}_{\rightarrow}}(a)] = Reach_{\mathcal{B}_{\rightarrow}}(h(a))$).

More on Reachability Homomorphisms (III)

Recall that, given a (Σ, ϕ) -reachability model $\mathcal{A}_{\rightarrow} = (\mathcal{A}, \rightarrow_{\mathcal{A}})$ having a Boolean predicate I with arguments of kind k we defined $\llbracket I \rrbracket_{\mathcal{A}_{\rightarrow}} = \{x \in A_k \mid I_{\mathcal{A}}(x) = \text{true}_{\mathcal{A}}\}$.

We say that a (Σ, ϕ) -reachability model $\mathcal{A}_{\rightarrow} = (\mathcal{A}, \rightarrow_{\mathcal{A}})$, where Σ contains the sort *Bool* with constants *true* and *false* **protects** *Bool* if $A_{Bool} = \{\text{true}_{\mathcal{A}}, \text{false}_{\mathcal{A}}\}$, and $\text{true}_{\mathcal{A}} \neq \text{false}_{\mathcal{A}}$.

Lemma 2. Suppose now that we have (Σ, ϕ) -reachability models $\mathcal{A}_{\rightarrow} = (\mathcal{A}, \rightarrow_{\mathcal{A}})$, and $\mathcal{B}_{\rightarrow} = (\mathcal{B}, \rightarrow_{\mathcal{B}})$ that both protect *Bool*. Then, any (Σ, ϕ) -reachability homomorphism $h : \mathcal{A}_{\rightarrow} \longrightarrow \mathcal{B}_{\rightarrow}$ must satisfy: (i) $h_k[\llbracket I \rrbracket_{\mathcal{A}_{\rightarrow}}] \subseteq \llbracket I \rrbracket_{\mathcal{B}_{\rightarrow}}$, and (ii) $h_k^{-1}[\llbracket I \rrbracket_{\mathcal{B}_{\rightarrow}}] = \llbracket I \rrbracket_{\mathcal{A}_{\rightarrow}}$.

More on Reachability Homomorphisms (IV)

Proof: By assumption I is a Boolean predicate in both $\mathcal{A}_{\rightarrow}$ and $\mathcal{B}_{\rightarrow}$, and since both protect $Bool$, for each $a \in A_k$ either $I_{\mathcal{A}}(a) = true_{\mathcal{A}}$, or $I_{\mathcal{A}}(a) = false_{\mathcal{A}}$. Likewise, for each $b \in B_k$ either $I_{\mathcal{B}}(b) = true_{\mathcal{B}}$, or $I_{\mathcal{B}}(b) = false_{\mathcal{B}}$. Of course, since h is a homomorphism, we have $h(true_{\mathcal{A}}) = true_{\mathcal{B}}$, and $h(false_{\mathcal{A}}) = false_{\mathcal{B}}$, which trivially implies (i). Applying h_k^{-1} to both sides of (i) we get the containment:

$h_k^{-1}[h_k[[[I]]_{\mathcal{A}_{\rightarrow}}]] \subseteq h_k^{-1}[[[I]]_{\mathcal{B}_{\rightarrow}}]$, which trivially gives us the containment $[[I]]_{\mathcal{A}_{\rightarrow}} \subseteq h_k^{-1}[[[I]]_{\mathcal{B}_{\rightarrow}}]$. So, to prove (ii) we just need to show $h_k^{-1}[[[I]]_{\mathcal{B}_{\rightarrow}}] \subseteq [[I]]_{\mathcal{A}_{\rightarrow}}$. To see this containment, suppose that $a \in A_k$ is such that $I_{\mathcal{B}}(h_k(a)) = true_{\mathcal{B}}$. We reason by contradiction. Suppose $I_{\mathcal{A}}(a) = false_{\mathcal{A}}$. Since h is a homomorphism we must have $h(I_{\mathcal{A}}(a)) = I_{\mathcal{B}}(h_k(a)) = false_{\mathcal{B}}$, a contradiction. q.e.d.

Proving Invariants with Reachability Homomorphisms

The lemmas we have been assembling can be put to good use to prove invariants. The point is that reachability homomorphisms allow us to **shift our ground** in the verification process. Using a reachability homomorphism $h : \mathcal{A}_{\rightarrow} \longrightarrow \mathcal{B}_{\rightarrow}$ we can reduce proving an invariant for $\mathcal{A}_{\rightarrow}$ (which may be infinite-state or too big) to proving an invariant for $\mathcal{B}_{\rightarrow}$ (which may be finite-state or smaller).

Theorem. Suppose now that we have a (Σ, ϕ) -reachability homomorphism $h : \mathcal{A}_{\rightarrow} \longrightarrow \mathcal{B}_{\rightarrow}$ and that both $\mathcal{A}_{\rightarrow} = (\mathcal{A}, \rightarrow_{\mathcal{A}})$, and $\mathcal{B}_{\rightarrow} = (\mathcal{B}, \rightarrow_{\mathcal{B}})$ protect *Bool*. Then, for any Boolean predicate in Σ we have the implication

$$\mathcal{B}_{\rightarrow}, h(a) \models \Box I \quad \Rightarrow \quad \mathcal{A}_{\rightarrow}, a \models \Box I$$

Furthermore, if h is a bisimulation this is an equivalence.

Proving Invariants with Reachability Homomorphisms (II)

Proof: By definition, we have to prove the implication:

$$Reach_{\mathcal{B} \rightarrow}(h(a)) \subseteq \llbracket I \rrbracket_{\mathcal{B} \rightarrow} \quad \Rightarrow \quad Reach_{\mathcal{A} \rightarrow}(a) \subseteq \llbracket I \rrbracket_{\mathcal{A} \rightarrow}$$

But by Lemma 2, applying h^{-1} to both sides of the hypothesis we get, $h^{-1}[Reach_{\mathcal{B} \rightarrow}(h(a))] \subseteq \llbracket I \rrbracket_{\mathcal{A} \rightarrow}$. And by Lemma 1 we have, $h^{-1}[h(Reach_{\mathcal{A} \rightarrow}(a))] \subseteq h^{-1}[Reach_{\mathcal{B} \rightarrow}(h(a))]$, which together with the trivial containment

$Reach_{\mathcal{A} \rightarrow}(a) \subseteq h^{-1}[h(Reach_{\mathcal{A} \rightarrow}(a))]$, give us

$Reach_{\mathcal{A} \rightarrow}(a) \subseteq \llbracket I \rrbracket_{\mathcal{A} \rightarrow}$, as desired. If, in addition, h is a bisimulation and we have $Reach_{\mathcal{A} \rightarrow}(a) \subseteq \llbracket I \rrbracket_{\mathcal{A} \rightarrow}$, then by Lemmas 1 and 2 we have,

$Reach_{\mathcal{B} \rightarrow}(h(a)) = h_k[Reach_{\mathcal{A} \rightarrow}(a)] \subseteq h[\llbracket I \rrbracket_{\mathcal{A} \rightarrow}] \subseteq \llbracket I \rrbracket_{\mathcal{B} \rightarrow}$, as desired. q.e.d.

Equational Abstractions

A very simple method to exploit the above theorem is to use an **equational abstraction**. The idea is enormously simple. Suppose that our system has been specified by means of a rewrite theory $\mathcal{R} = (\Sigma, E, \phi, R)$ which we assume satisfies all the executability conditions and protects the sort `Bool`. We can then **add new equations**, say G to \mathcal{R} to obtain a rewrite theory $\mathcal{R}/G = (\Sigma, E \cup G, \phi, R)$. Now consider the initial model $\mathcal{T}_{\mathcal{R}/G}$. By construction $\mathcal{T}_{\mathcal{R}/G}$ satisfies $E \cup G$ (in particular E), and R . Therefore, by the initiality theorem for $\mathcal{T}_{\mathcal{R}}$ we have a unique reachability homomorphism

$$eval_{\mathcal{T}_{\mathcal{R}/G}}^{\mathcal{R}} : \mathcal{T}_{\mathcal{R}} \longrightarrow \mathcal{T}_{\mathcal{R}/G}$$

mapping each $[t]_E$ to $[t]_{E \cup G}$. We call $\mathcal{T}_{\mathcal{R}/G}$ the **equational abstraction** by equations G of $\mathcal{T}_{\mathcal{R}}$.

Equational Abstractions (II)

Suppose that \mathcal{R}/G protects Bool. Then, for any invariant I of interest, our previous theorem immediately applies to give as an implication:

$$\mathcal{T}_{\mathcal{R}/G}, [t]_{E \cup G} \models \Box I \quad \Rightarrow \quad \mathcal{T}_{\mathcal{R}}, [t]_E \models \Box I$$

Therefore, we can use the equational abstraction $\mathcal{T}_{\mathcal{R}/G}$, which typically is much smaller and can even be finitely reachable when $\mathcal{T}_{\mathcal{R}}$ is infinitely reachable, to verify the invariant $\mathcal{T}_{\mathcal{R}}, [t]_E \models \Box I$. But to do this in Maude we need, besides checking the requirement that \mathcal{R}/G protects Bool, to also check that \mathcal{R}/G satisfies the usual executability requirements, namely, that it is ground confluent, sort decreasing, and terminating, and also that it is ground coherent.

A Readers&Writers Example

We can illustrate the power of equational abstractions with our readers and writers example, for which we already performed bounded model checking of invariants up to depth 10^6 in Lecture 22. Since the Maude Church-Rosser Checker and Coherence Checker tools do not currently allow built-in submodules like NAT or BOOL, we will consider the following slight variants of our original specifications that do not use any built-ins.

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

```

    rl < 0, 0 > => < 0, s(0) > .
    rl < R, s(W) > => < R, W > .
    rl < R, 0 > => < s(R), 0 > .
    rl < s(R), W > => < R, W > .
endm

mod R&W-PREDS is
  protecting R&W .
  sort NewBool .
  ops tt ff : -> NewBool [ctor] .
  ops mutex one-writer : Config -> NewBool [frozen] .
  eq mutex(< s(N:Nat),s(M:Nat) >) = ff .
  eq mutex(< 0,N:Nat >) = tt .
  eq mutex(< N:Nat,0 >) = tt .
  eq one-writer(< N:Nat,s(s(M:Nat)) >) = ff .
  eq one-writer(< N:Nat,0 >) = tt .
  eq one-writer(< N:Nat,s(0) >) = tt .
endm

```

A Readers&Writers Example (II)

We can then drastically collapse the set of states by defining the following equational abstraction, which we can immediately use to verify our two invariants:

```
mod R&W-ABS is
  including R&W-PREDS .
  eq < s(s(N:Nat)),0 > = < s(0),0 > .
endm
```

```
=====
search in R&W-ABS : < 0,0 > =>* C:Config such that
                                     mutex(C:Config) = ff .
```

No solution.

```
=====
search in R&W-ABS : < 0,0 > =>* C:Config such that
                                     one-writer(C:Config) = ff .
```

No solution.

A Readers&Writers Example (III)

Since Maude computes really in the canonical reachability model $\mathcal{C}_{\mathcal{R}}$ of the given rewrite theory \mathcal{R} which is only isomorphic to the initial model $\mathcal{T}_{\mathcal{R}}$ under the required executability assumptions, in reality we are not yet finished verifying that our original readers and writers system satisfies the two invariants using the above equational abstraction. We furthermore need to show that:

1. both R&W-PREDS and R&W-ABS protect NewBool;
2. R&W-ABS is ground confluent, sort-decreasing, and terminating; and
3. R&W-ABS is ground coherent.

A Readers&Writers Example (IV)

Note that we can easily show (1): that both `R&W-PREDS` and `R&W-ABS` protect `NewBool`, by showing that: (i) both are ground confluent, sort-decreasing, and terminating; and (ii) both are sufficiently complete. Indeed, by (i), the canonical term algebra by the equations is isomorphic to the initial algebra; and by (ii), the only canonical terms of sort `NewBool` must be the constructors `tt` and `ff`, which are both in canonical form, and therefore different. Note also that (i) above shows (2) as well.

A Readers&Writers Example (V)

We can check local confluence and sort-decreasingness of R&W-PREDS and R&W-ABS with the CRC tool:

```
Maude> (check Church-Rosser R&W-PREDS .)
```

Checking solution:

All critical pairs have been joined. The specification is
locally-confluent.

The specification is sort-decreasing.

```
Maude> (check Church-Rosser R&W-ABS .)
```

Checking solution:

All critical pairs have been joined. The specification is
locally-confluent.

The specification is sort-decreasing.

A Readers&Writers Example (VI)

Since the termination of the equations in R&W-ABS implies that of the (fewer) equations in R&W-PREDS, it is enough to check the first module. After extracting a functional module with the equations from R&W-ABS and with a slight change of syntax and using AProVe through the MTT tool (with no sort information) we get:

(fmod RWABS is

```
sort Nat Config .
op cfg : Nat Nat -> Config [ctor] .
op zero : -> Nat [ctor] .
op s : Nat -> Nat [ctor] .
vars R W : Nat .

sort NewBool .
```



```

op tt : -> NewBool [ctor] .
op ff : -> NewBool [ctor] .
op mutex  : Config -> NewBool .
op one-writer : Config -> NewBool .

```

```

vars M N : Nat .

```

```

eq mutex(cfg(s(N), s(M))) = ff .
eq mutex(cfg(zero, N)) = tt .
eq mutex(cfg(N, zero)) = tt .
eq one-writer(cfg(N, s(s(M)))) = ff .
eq one-writer(cfg(N, zero)) = tt .
eq one-writer(cfg(N, s(zero))) = tt .
eq cfg(s(s(N)), zero) = cfg(s(zero), zero) .

```

```

endfm)

```

*** AProVe output

Termination of R to be shown.

R ->Removing Redundant Rules

Removing the following rules from R which fullfill a polynomial ordering:

```
and(tt, X) -> X
mutex(cfg(s(N), s(M))) -> ff
one - writer(cfg(N, s(zero))) -> tt
one - writer(cfg(N, s(s(M)))) -> ff
```

where the Polynomial interpretation:

$POL(\text{and}(xo1, xo2)) = xo1 + xo2$

$POL(ff) = 0$

$POL(\text{mutex}(xo1)) = xo1$

$POL(tt) = 1$

$POL(s(xo1)) = 1 + xo1$

```
POL(cfg(xo1, xo2)) = xo1 + xo2
POL(xo1 - xo2) = xo1 + xo2
POL(one) = 0
POL(zero) = 1
POL(writer(xo1)) = xo1
was used.
```

Not all Rules of R can be deleted, so we still have to regard a part of R.

```
R      ->RRRPol0
      ->TRS2
      ->Removing Redundant Rules
```

Removing the following rules from R which fullfill a polynomial ordering:

`cfg(s(s(N)), zero) -> cfg(s(zero), zero)`

where the Polynomial interpretation:

`POL(tt) = 0`

`POL(mutex(xo1)) = xo1`

`POL(s(xo1)) = 1 + xo1`

`POL(cfg(xo1, xo2)) = xo1 + xo2`

`POL(xo1 - xo2) = xo1 + xo2`

`POL(one) = 0`

`POL(zero) = 0`

`POL(writer(xo1)) = xo1`

was used.

Not all Rules of R can be deleted, so we still have to regard a part of R.

`R -> RRRPol0`

->TRS2

->RRRPol0

->TRS3

->Removing Redundant Rules

Removing the following rules from R which fullfill a polynomial ordering:

one - writer(cfg(N, zero)) -> tt

where the Polynomial interpretation:

POL(mutex(xo1)) = xo1

POL(tt) = 0

POL(cfg(xo1, xo2)) = xo1 + xo2

POL(xo1 - xo2) = xo1 + xo2

POL(one) = 1

POL(zero) = 0

POL(writer(xo1)) = xo1

was used.

Not all Rules of R can be deleted, so we still have to regard a part of R.

R ->RRRPol0

 ->TRS2

 ->RRRPol0

 ->TRS3

 ->RRRPol0

 ...

 ->TRS4

 ->Removing Redundant Rules

Removing the following rules from R which fullfill a polynomial ordering:

```
mutex(cfg(zero, N)) -> tt
mutex(cfg(N, zero)) -> tt
```

where the Polynomial interpretation:

$POL(mutex(xo1)) = xo1$

$POL(tt) = 0$

$POL(cfg(xo1, xo2)) = 1 + xo1 + xo2$

$POL(zero) = 0$

was used.

All Rules of R can be deleted.

R \rightarrow RRRPololo

->TRS2

->RRRPol0

->TRS3

->RRRPol0

...

->TRS5

->Overlay and local confluence Check

The TRS is overlay and locally confluent (all critical pairs are trivially joinable)

R ->RRRPol0

->TRS2

->RRRPol0

->TRS3

->RRRPol0

...

->TRS6

->Dependency Pair Analysis

R contains no Dependency Pairs and therefore no SCCs.

Termination of R successfully shown.

Duration:

0:00 minutes

A Readers&Writers Example (VII)

Next we can use the SCC Tool to check the sufficient completeness of R&W-PREDS and R&W-ABS which, together with the confluence and termination checks already performed ensures that R&W-PREDS and R&W-ABS protect BOOL.

```
Maude> load scc
Maude> (scc R&W-PREDS .)
Checking sufficient completeness of R&W-PREDS ...
Success: R&W-PREDS is sufficiently complete under the assumption that it is
        weakly-normalizing, confluent, and sort-decreasing.
Maude> (scc R&W-ABS .)
Checking sufficient completeness of R&W-ABS ...
Success: R&W-ABS is sufficiently complete under the assumption that it is
        weakly-normalizing, confluent, and sort-decreasing.
```

A Readers&Writers Example (VIII)

The only remaining check is the ground coherence of R&W-ABS. Using the ChC tool get get:

```
Maude> (check coherence R&W-ABS .)
```

```
The following critical pairs cannot be rewritten:
```

```
cp < s(0),0 >
```

```
=> < s(N*0:Nat),0 > .
```

Ground coherence requires that all **ground** instances of such a pair can indeed be rewritten in one step by the rules. We can reason by cases and consider the canonical forms by our equation of the following instances of the righthand side:

- $\text{can}(\langle s(0), 0 \rangle) = \langle s(0), 0 \rangle$
- $\text{can}(\langle s(s(N:\text{Nat})), 0 \rangle) = \langle s(0), 0 \rangle$

A Readers&Writers Example (IX)

So, all boils down to checking whether we can rewrite the term $\langle s(0), 0 \rangle$ to **itself** in one step with the rules of the module, up to canonical form. This is indeed the case, since using rule

$$r1 \langle R, 0 \rangle \Rightarrow \langle s(R), 0 \rangle .$$

we can rewrite $\langle s(0), 0 \rangle$ to $\langle s(s(0)), 0 \rangle$, whose canonical form is $\langle s(0), 0 \rangle$.