# 3. Inductive Reachability Analysis

**Camilo Rocha**
Escuela Colombiana de Ingeniería
Joint work with J.Meseguer (UIUC)

7th International School on Rewriting (ISR2014)
Universidad Técnica Federico Santa María
August 25-29, 2014

# Introduction

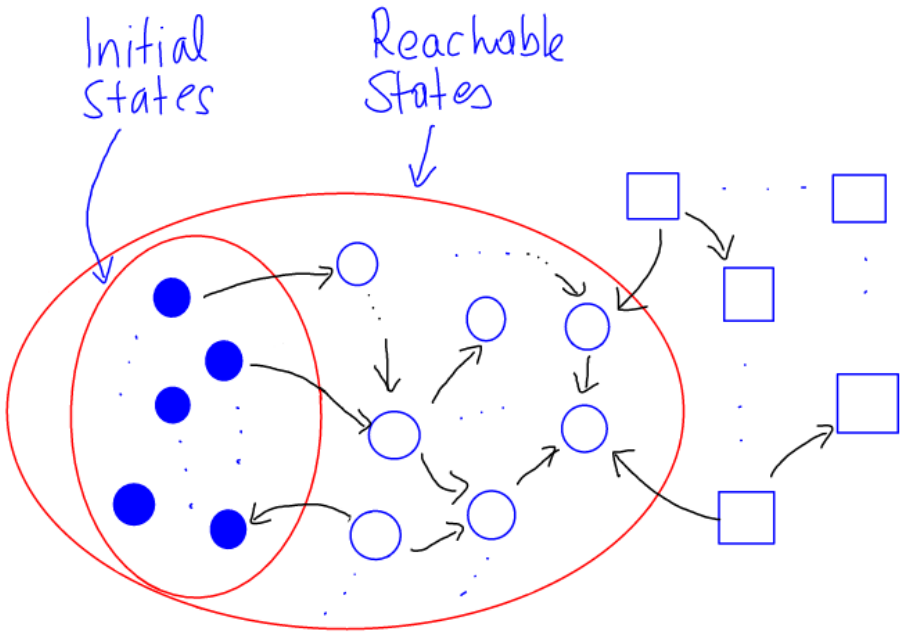
- Verification is important in the software disciplines
  - concurrency, state explosion, ...

- A **safety property** is a property stating that "something bad does never happen"
  - "a critical resource is never shared by more than one thread"
  - "the power plant will never blow up"

# The Problem

How do we verify a safety property for a transition system having

- an arbitrary number of initial states?
- an arbitrary number of reachable states?



# Overview

We will focus on **stability** and **invariance** properties for the initial reachability model $\mathcal{T}_{\mathcal{R}}$ of a topmost rewrite theory $\mathcal{R}$

- $\mathcal{T}_{\mathcal{R}}$ may be **infinite-state**, so that standard model checking algorithms may not be usable
- we employ **deductive proof methods** that can be applied to infinite-state systems and can assume infinite sets of initial states

# Outline


1 Preliminaries

2 Safety Properties

3 Proving Stability and Invariance Properties

4 Maude's Invariant Analyzer Tool (InvA)

# Rewrite Theories

In what follows, we assume $\mathcal{R} = (\Sigma, E \cup A, R)$ is such that:

- the equations $E$ are ground confluent, ground strongly normalizing, and ground coherent w.r.t. $R$ modulo $A$
- it is topmost, i.e., $\Sigma$ has a topmost sort $\mathfrak{s}$ and each rule $l \to r$ if $C \in R$ has $l, r \in T_\Sigma(X)_\mathfrak{s}$
- $E$ and $R$ can be conditional

and let:

- $\mathcal{E}_\mathcal{R} = (\Sigma, E \cup A)$
- $\mathcal{T}_{\Sigma/E \cup A}$ be the initial algebra of $\mathcal{E}_\mathcal{R}$
- $\mathcal{T}_\mathcal{R} = (\mathcal{T}_{\Sigma/E \cup A}, \overset{*}{\to}_\mathcal{R})$ be the initial reachability model of $\mathcal{R}$

# The Running Example

## The Transition System $\mathcal{T}_{\texttt{2BAK}}$

Consider the infinite-state protocol 2BAK, a simplified version of Lamport's bakery protocol, that achieves **mutual exclusion** between two processes:

```
fmod 2BAK-STATE is protecting NAT .
  sorts Mode State .
  ops sleep wait crit : -> Mode [ctor] .
  op <_,_,_,_> : Mode Nat Mode Nat -> State [ctor] .
endfm

mod 2BAK is protecting 2BAK-STATE .
  vars M N : Nat .  vars X Y : Mode .
  rl [p1_s] : < sleep, M, Y, N > => < wait, s N, Y, N > .
 crl [p1_w] : < wait, M, Y, N >  => < crit, M, Y, N > if M <= N = true .
  rl [p1_c] : < crit, M, Y, N >  => < sleep, 0, Y, N > .
  rl [p2_s] : < X, M, sleep, N > => < X, M, wait, s M > .
 crl [p2_w] : < X, M, wait, N >  => < X, M, crit, N > if N < M = true .
  rl [p2_c] : < X, M, crit, N >  => < X, M, sleep, 0 > .
endm
```

# State Predicates

We let $\Pi$ be a set of **state predicates** for $\mathcal{R} = (\Sigma, E \cup A, R)$, defined in an equational theory $\mathcal{E}_\Pi = (\Sigma_\Pi, E \cup A \cup E_\Pi)$ such that:

# State Predicates

We let $\Pi$ be a set of **state predicates** for $\mathcal{R} = (\Sigma, E \cup A, R)$, defined in an equational theory $\mathcal{E}_\Pi = (\Sigma_\Pi, E \cup A \cup E_\Pi)$ such that:

- $\Sigma_\Pi$ contains $\Sigma$, two sorts $\mathbb{B} \leqslant [\mathbb{B}]$ with constants $\top$ and $\bot$ of sort $\mathbb{B}$, predicate symbols $P : \mathfrak{s} \longrightarrow [\mathbb{B}]$ for each $P \in \Pi$, and optionally some auxiliary function symbols

# State Predicates

We let $\Pi$ be a set of **state predicates** for $\mathcal{R} = (\Sigma, E \cup A, R)$, defined in an equational theory $\mathcal{E}_\Pi = (\Sigma_\Pi, E \cup A \cup E_\Pi)$ such that:

- $\Sigma_\Pi$ contains $\Sigma$, two sorts $\mathbb{B} \leqslant [\mathbb{B}]$ with constants $\top$ and $\bot$ of sort $\mathbb{B}$, predicate symbols $P : \mathfrak{s} \longrightarrow [\mathbb{B}]$ for each $P \in \Pi$, and optionally some auxiliary function symbols
- the equations $E_\Pi$ define the predicate symbols in $\Sigma_\Pi$ and the auxiliary function symbols, and they protect both $\mathcal{E}_\mathcal{R}$ and the sort $\mathbb{B}$

# State Predicates

We let $\Pi$ be a set of **state predicates** for $\mathcal{R} = (\Sigma, E \cup A, R)$, defined in an equational theory $\mathcal{E}_\Pi = (\Sigma_\Pi, E \cup A \cup E_\Pi)$ such that:

- $\Sigma_\Pi$ contains $\Sigma$, two sorts $\mathbb{B} \leqslant [\mathbb{B}]$ with constants $\top$ and $\bot$ of sort $\mathbb{B}$, predicate symbols $P : \mathfrak{s} \longrightarrow [\mathbb{B}]$ for each $P \in \Pi$, and optionally some auxiliary function symbols
- the equations $E_\Pi$ define the predicate symbols in $\Sigma_\Pi$ and the auxiliary function symbols, and they protect both $\mathcal{E}_\mathcal{R}$ and the sort $\mathbb{B}$
- the equations $E \cup E_\Pi$ are ground confluent, ground strongly normalizing, and ground coherent w.r.t. $R$ modulo $A$

# The Running Example

## State Predicates for $\mathcal{T}_{\text{2BAK}}$

```
fmod 2BAK-PROPS is protecting 2BAK-STATE .
  vars M N : Nat .  vars X Y : Mode .
  ops init mutex : State -> [Bool] .
  eq [init] : init(< sleep, 0, sleep, 0 >) = true .
  eq [p1_s-p2_s] : mutex(< sleep, M, sleep, N >) = true .
  eq [p1_w-p2_s] : mutex(< wait, M, sleep, N >) = true .
  eq [p1_c-p2_s] : mutex(< crit, M, sleep, N >) = true .
  eq [p1_s-p2_w] : mutex(< sleep, M, wait, N >) = true .
  eq [p1_w-p2_w] : mutex(< wait, M, wait, N >) = true .
  eq [p1_c-p2_w] : mutex(< crit, M, wait, N >) = true .
  eq [p1_s-p2_c] : mutex(< sleep, M, crit, N >) = true .
  eq [p1_w-p2_c] : mutex(< wait, M, crit, N >) = true .
  eq [p1_c-p2_c] : mutex(< crit, M, crit, N >) = false .
endfm
```

### Definition (Next)

Let $P$ be a state predicate defined on the set of states of $\mathcal{T}_{\mathcal{R}}$:

- for $t \in T_{\Sigma,\mathfrak{s}}$, we say that

$$\mathcal{T}_{\mathcal{R}}, t \models \bigcirc P \iff (\forall u \in T_{\Sigma,\mathfrak{s}})\, \mathcal{R} \vdash t \xrightarrow{1} u \implies \mathcal{E}_{\Pi} \vdash P(u) = \top$$

- we define

$$\mathcal{T}_{\mathcal{R}} \models \bigcirc P \iff (\forall t \in T_{\Sigma,\mathfrak{s}})\, \mathcal{T}_{\mathcal{R}}, t \models \bigcirc P$$

### Definition (Always)

Let $P$ be a state predicate defined on the set of states of $\mathcal{T}_{\mathcal{R}}$:

- for $t \in T_{\Sigma,\mathfrak{s}}$, we say that

$$\mathcal{T}_{\mathcal{R}}, t \models \Box P \iff (\forall u \in T_{\Sigma,\mathfrak{s}})\, \mathcal{R} \vdash t \xrightarrow{*} u \implies \mathcal{E}_{\Pi} \vdash P(u) = \top$$

- we define

$$\mathcal{T}_{\mathcal{R}} \models \Box P \iff (\forall t \in T_{\Sigma,\mathfrak{s}})\, \mathcal{T}_{\mathcal{R}}, t \models \Box P$$

### Safety Properties

We are interested in verifying **safety properties** of the form

$$\mathcal{T}_{\mathcal{R}} \models I \Rightarrow \Box P$$

with $I$ and $P$ state predicates in $\Pi$. $I$ denotes the set of **initial states** and $P$ is the **invariant**.

### Safety Properties

We are interested in verifying **safety properties** of the form

$$\mathcal{T}_{\mathcal{R}} \models I \Rightarrow \Box P$$

with $I$ and $P$ state predicates in $\Pi$. $I$ denotes the set of **initial states** and $P$ is the **invariant**.

### Our Approach

We use a proof system that reduces the verification task of proving **stability** and **invariance** properties of $\mathcal{T}_{\mathcal{R}}$ to a first-order equational (inductive) theorem proving task in $\mathcal{E}_{\Pi}$

# Stability

For $P$ a state predicate defined on the set of states of $\mathcal{T}_{\mathcal{R}}$, $P$-stability is the safety property

$$\mathcal{T}_{\mathcal{R}} \models P \Rightarrow \Box P$$

intuitively expressing that once $P$ becomes true, it remains true forever.

## Definition (Stability)

Let $P \in \Pi$. We define **$P$-stability** for $\mathcal{T}_{\mathcal{R}}$ as follows:

$$\mathcal{T}_{\mathcal{R}} \models P \Rightarrow \Box P \quad \Longleftrightarrow \quad (\forall t \in T_{\Sigma,\mathfrak{s}})\ \mathcal{E}_{\Pi} \vdash P(t) = \top \Longrightarrow \mathcal{T}_{\mathcal{R}}, t \models \Box P$$

# Invariance

For $I$ and $P$ state predicates defined on the set of states of $\mathcal{T}_\mathcal{R}$, $P$-invariance from a set $I$ of initial states is the safety property

$$\mathcal{T}_\mathcal{R} \models I \Rightarrow \Box P$$

intuitively expressing that once $I$ becomes true, $P$ is true forever.

## Definition (Invariance)

Let $I, P \in \Pi$. We define **$P$-invariance from $I$** for $\mathcal{T}_\mathcal{R}$ as follows:

$$\mathcal{T}_\mathcal{R} \models I \Rightarrow \Box P \quad \Longleftrightarrow \quad (\forall t \in T_{\Sigma,\mathfrak{s}})\ \mathcal{E}_\Pi \vdash I(t) = \top \Longrightarrow \mathcal{T}_\mathcal{R}, t \models \Box P$$

# The Running Example

## State Predicates for $\mathcal{T}_{\texttt{2BAK}}$

Mutual exclusion for 2BAK can be expressed by the invariant property:

$$\mathcal{T}_{\texttt{2BAK}} \models \texttt{init} \Rightarrow \Box\texttt{mutex}$$

# The Proof System

The proof system comprises three groups of inference rules:

- **reduction inference rules** used to reduce stability and invariance properties for $\mathcal{T}_{\mathcal{R}}$ to properties of the form $P \Rightarrow Q$ and $P \Rightarrow \bigcirc P$
- **descent inference rules** used to reduce properties of the form $P \Rightarrow Q$ and $P \Rightarrow \bigcirc P$ for $\mathcal{T}_{\mathcal{R}}$ to equational inductive reasoning in $\mathcal{E}_{\Pi}$
- **strenghtening inference rules** used to strenghten invariants

# Reduction Inference Rules

## Inductive Stability and Inductive Invariance

**Theorem**

*For $I, P \in \Pi$ state predicates defined on the set of states of $\mathcal{T}_{\mathcal{R}}$, the following reduction inference rules are sound:*

$$\frac{\mathcal{T}_{\mathcal{R}} \models P \Rightarrow \bigcirc P}{\mathcal{T}_{\mathcal{R}} \models P \Rightarrow \Box P}\ \textsc{St}$$

$$\frac{\mathcal{T}_{\mathcal{R}} \models I \Rightarrow P \qquad \mathcal{T}_{\mathcal{R}} \models P \Rightarrow \Box P}{\mathcal{T}_{\mathcal{R}} \models I \Rightarrow \Box P}\ \textsc{Inv}$$

# The Running Example

Composing ST and INV for the mutual exclusion of 2BAK, we get:

$$\dfrac{\dfrac{}{\mathcal{T}_{\texttt{2BAK}} \models \texttt{init} \Rightarrow \texttt{mutex}} \qquad \dfrac{\dfrac{}{\mathcal{T}_{\texttt{2BAK}} \models \texttt{mutex} \Rightarrow \bigcirc\texttt{mutex}}}{\mathcal{T}_{\texttt{2BAK}} \models \texttt{mutex} \Rightarrow \Box\texttt{mutex}}}{\mathcal{T}_{\texttt{2BAK}} \models \texttt{init} \Rightarrow \Box\texttt{mutex}}$$

# The Running Example

Composing ST and INV for the mutual exclusion of 2BAK, we get:

$$\dfrac{\dfrac{?}{\mathcal{T}_{\texttt{2BAK}} \models \texttt{init} \Rightarrow \texttt{mutex}} \qquad \dfrac{\dfrac{?}{\mathcal{T}_{\texttt{2BAK}} \models \texttt{mutex} \Rightarrow \bigcirc\texttt{mutex}}}{\mathcal{T}_{\texttt{2BAK}} \models \texttt{mutex} \Rightarrow \Box\texttt{mutex}}}{\mathcal{T}_{\texttt{2BAK}} \models \texttt{init} \Rightarrow \Box\texttt{mutex}}$$

# Descent Inference Rules

**Theorem**

*For $P, Q \in \Pi$ state predicates defined on the set of states of $\mathcal{T}_{\mathcal{R}}$, the following descent inference rules are sound:*

$$\frac{\mathcal{E}_\Pi \vdash_{\text{ind}} (\forall x : \mathfrak{s})\ P(x) = \top \ \Rightarrow\ Q(x) = \top}{\mathcal{T}_{\mathcal{R}} \models P \Rightarrow Q}\ \textsc{Eq-D}$$

$$\frac{\{\mathcal{E}_\Pi \vdash_{\text{ind}} (\forall \vec{y} : \vec{s})\ (P(l) = \top \wedge C) \ \Rightarrow\ P(r) = \top\}_{(l \to r \text{ if } C) \in R}}{\mathcal{T}_{\mathcal{R}} \models P \Rightarrow \bigcirc P}\ \textsc{Rew-D}$$

# Simplifying Proof Obligations

To prove $P \Rightarrow \bigcirc P$, with $P \in \Pi$, the inference system generates for each rewrite rule $(\forall X)\ l \rightarrow r$ if $C$ inductive proof obligations of the form:

$$\mathcal{E}_\Pi \vdash_{\text{ind}} (\forall X)\ P(r) = \top \text{ if } P(l) = \top \wedge C$$

# Simplifying Proof Obligations

To prove $P \Rightarrow \bigcirc P$, with $P \in \Pi$, the inference system generates for each rewrite rule $(\forall X)\ l \to r$ if $C$ inductive proof obligations of the form:

$$\mathcal{E}_\Pi \vdash_{\text{ind}} (\forall X)\ P(r) = \top \text{ if } P(l) = \top \wedge C$$

**Key idea**: 1-step narrowing with the predicate $P$ in the antecedent!!!

# Simplifying Proof Obligations

To prove $P \Rightarrow \bigcirc P$, with $P \in \Pi$, the inference system generates for each rewrite rule $(\forall X)\ l \to r$ if $C$ inductive proof obligations of the form:

$$\mathcal{E}_\Pi \vdash_{\text{ind}} (\forall X)\ P(r) = \top \text{ if } P(l) = \top \wedge C$$

**Key idea**: 1-step narrowing with the predicate $P$ in the antecedent!!!
For a ground substitution $\alpha$, $\mathcal{E}_\Pi \vdash P(l\alpha) = \top$ holds iff

- $\exists\ P(v) = w$ if $D \in E_\Pi$
  - assume $v$ (also $l$) is a constructor-based term
  - assume free constructors modulo the axioms $A$
  - let $\text{CSU}_A(l = v)$ be the set of most general $A$-unifiers of $l = v$

# Simplifying Proof Obligations

To prove $P \Rightarrow \bigcirc P$, with $P \in \Pi$, the inference system generates for each rewrite rule $(\forall X)\ l \to r$ if $C$ inductive proof obligations of the form:

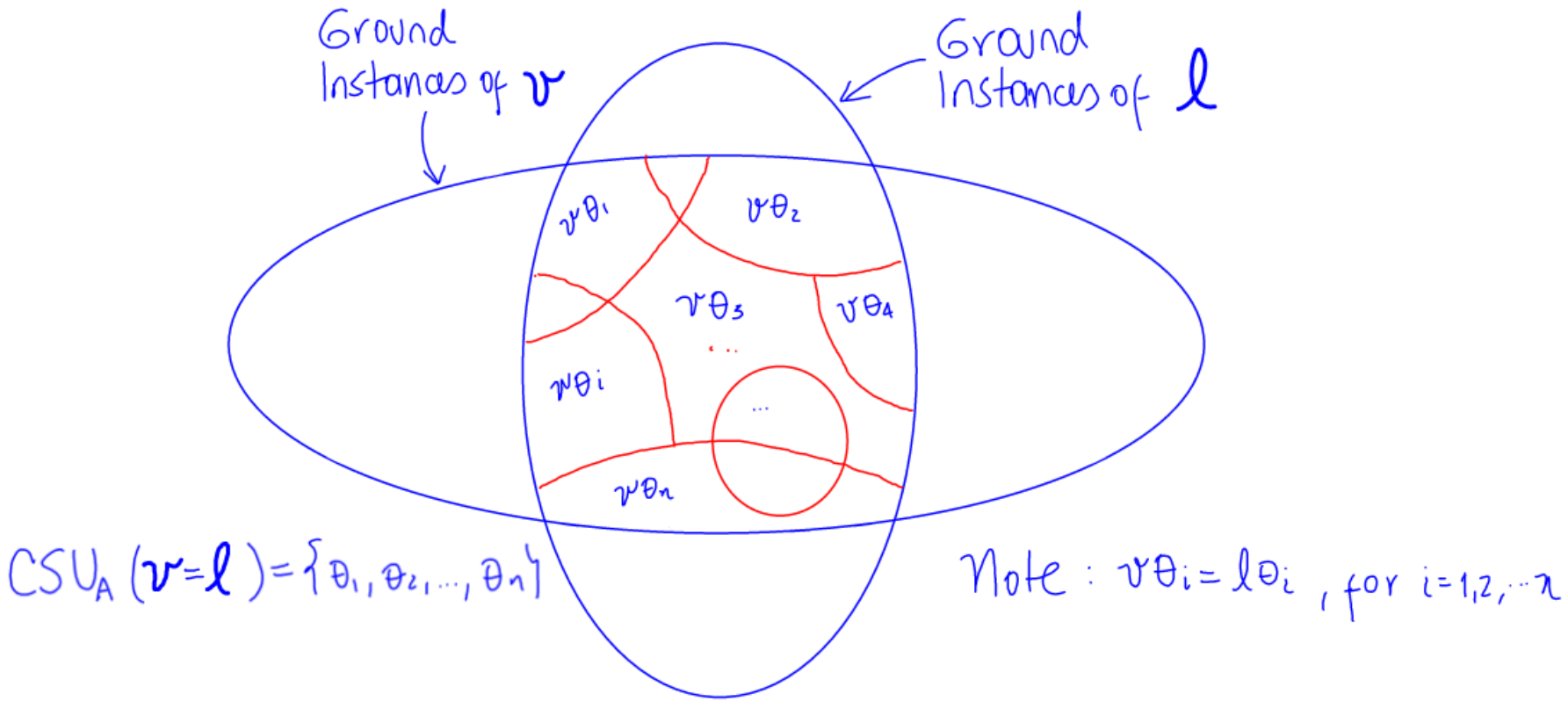
$$\mathcal{E}_\Pi \vdash_{\text{ind}} (\forall X)\ P(r) = \top \text{ if } P(l) = \top \wedge C$$

**Key idea**: 1-step narrowing with the predicate $P$ in the antecedent!!!
For a ground substitution $\alpha$, $\mathcal{E}_\Pi \vdash P(l\alpha) = \top$ holds iff

- $\exists\ P(v) = w$ if $D \in E_\Pi$
  - assume $v$ (also $l$) is a constructor-based term
  - assume free constructors modulo the axioms $A$
  - let $\text{CSU}_A(l = v)$ be the set of most general $A$-unifiers of $l = v$
- $\exists\ \theta \in \text{CSU}_A(l = v)$ and ground substitution $\gamma$ such that
  - $l\theta =_A v\theta$
  - $\alpha =_{E \cup A} \theta\gamma$
  - $\mathcal{E}_\Pi \vdash D\theta\gamma \wedge w\theta\gamma = \top$

# Simplifying Proof Obligations

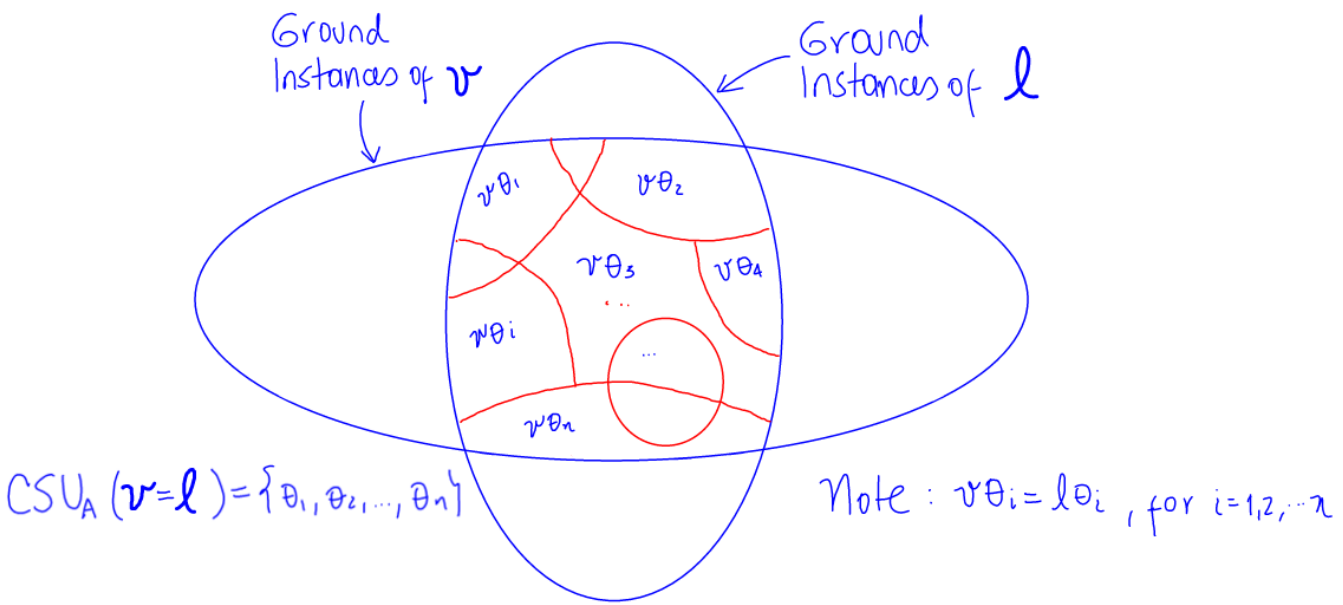
## Graphical Argument



# Simplifying Proof Obligations

Then REW-D can be streamlined as follows:

$$\frac{\{\mathcal{E}_\Pi \vdash_{\text{ind}} (\forall \text{ran}(\theta))\ P(r\theta) = \top \text{ if } C\theta \wedge D\theta \wedge w\theta = \top\}^{\substack{((\forall Y)\ P(v)=w \text{ if } D)\in E_\Pi \\ ((\forall X)\ l\to r \text{ if } C)\in R, \theta\in \text{CSU}_A(l=v)}}}{\mathcal{T}_\mathcal{R} \models P \Rightarrow \bigcirc P}$$



# Strengthening Rules

**Theorem**

*For $I, J, P, Q \in \Pi$ state predicates defined on the set of states of $\mathcal{T}_\mathcal{R}$ , the following strengthening inference rules are sound:*

$$\frac{\mathcal{T}_\mathcal{R} \models I \Rightarrow J \qquad \mathcal{T}_\mathcal{R} \models J \Rightarrow \Box Q \qquad \mathcal{T}_\mathcal{R} \models Q \Rightarrow P}{\mathcal{T}_\mathcal{R} \models I \Rightarrow \Box P} \textsc{Str1}$$

$$\frac{\mathcal{T}_\mathcal{R} \models I \Rightarrow P \qquad \mathcal{T}_\mathcal{R} \models I \Rightarrow \Box Q \qquad \mathcal{T}_\mathcal{R} \models Q \wedge P \Rightarrow \bigcirc P}{\mathcal{T}_\mathcal{R} \models I \Rightarrow \Box P} \textsc{Str2}$$

# Main Theorem

**Theorem**

*Let $I, P \in \Pi$ and let $\Psi$ be the set of equational Horn clauses generated by the inference system on input $I \Rightarrow \Box P$ for $\mathcal{R} = (\Sigma, E \cup A, R)$. If*

$$(\forall \psi \in \Psi)\ \ \mathcal{E}_\Pi \vdash_{\text{ind}} \psi$$

*then*

$$\mathcal{T}_\mathcal{R} \models I \Rightarrow \Box P$$

# The InvA Tool

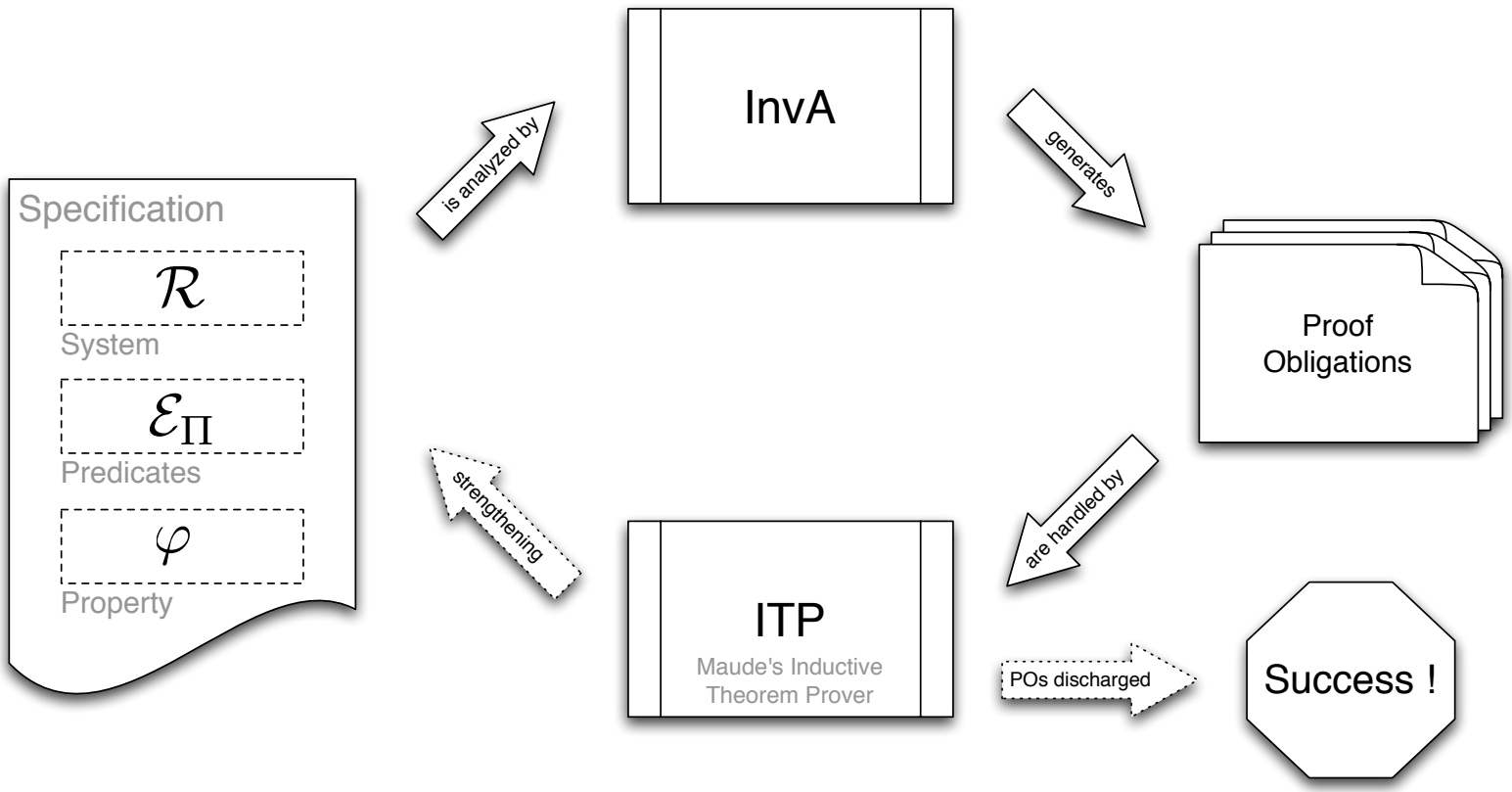
## An Implementation

The Maude Invariant Analyzer Tool (**InvA**) is an implementation in Maude of the inference system

- it can automatically discharge many proof obligations by using
  - rewriting and narrowing-based reasoning
  - SMT decision procedures (experimental)
- it runs on Maude 2.6
  - SMT solving is available from an extension of Maude with the CVC3 theorem prover

# InvA

## Methodology



## Example (Running InvA)

If `maude` is your Maude executable and the InvA specification is in `inva.maude`, InvA can be started as follows:

```
> maude inva.maude
                     ||||||||||||||||||/
                   --- Welcome to Maude ---
                    /||||||||||||||||||
            Maude 2.6 built: Dec 10 2010 11:12:39
            Copyright 1997-2010 SRI International
            Mon Apr 16 22:59:21 2012

            Full Maude 2.6 - February 3rd 2011

            Invariant Analyzer 1.2 - April 17th 2012
            (with Church-Rosser Checker 3l and CVC3)

Maude> _
```

### Example (Loading Specifications)

If `spec.maude` is the name of the file containing the specification of $\mathcal{R}$ and $\mathcal{E}_{\Pi}$, they can be loaded as follows:

```
Maude> load spec.maude
Maude> _
```

### Example (Activate InvA)

InvA is made active as follows:

```
Maude> select INVA .
Maude> loop init .
          Invariant Analyzer 1.2 - April 17th 2012
          (with Church-Rosser Checker 3l and CVC3)
Maude> _
```

# InvA

## Help Menu

InvA's help menu can be shown by typing the command

```
(help .)
```

## Example (Showing the Help Menu)

```
Maude> (help .)

Commands available from the

            Invariant Analyzer 1.2 - April 17th 2012

 (analyze-stable <pred> in <fmodule> <module> .)
   generates the proof obligations for the ground
   <pred> -stability of the topmost rewrite theory
 ...
 (help .)
   shows this help menu
Maude> _
```

```
(analyze <pred> implies <pred> in <fmodule> .)
```

generates the proof obligations for proving the given implication in the given module, according to inference $\mathrm{EQ}$-$\mathrm{D}$. This command tries to eagerly discharge the proof obligations; those that cannot be discharged are shown to the user.

**Example (Analyzing $\mathcal{T}_{\mathtt{2BAK}} \models \mathtt{init} \Rightarrow \mathtt{mutex}$)**

```
Maude> (analyze init(S:State) implies mutex(S:State) in 2BAK-PROPS .)
Checking 2BAK-PROPS-EXT ||- init => mutex ...
Proof obligations generated:  1
Proof obligations discharged: 1
Success!
Maude> _
```

```
(analyze-stable <pred> in <fmodule> <module> .)
```

generates the proof obligations for proving the premise of inference ST with the streamlined inference REW-D, for the given predicate and the given modules. The first module equationally specifies the state predicate and the second one the top most rewrite theory. This command tries to eagerly discharge the proof obligations; those that cannot be discharged are shown to the user.

# InvA

## Analyzing Stability (cont.)

```
(analyze-stable <pred> in <fmodule> <module> .)
```

**Example (Analyzing $\mathcal{T}_{\texttt{2BAK}} \models \texttt{mutex} \Rightarrow \Box\texttt{mutex}$)**

```
Maude> (analyze-stable mutex(S:State) in 2BAK-PROPS 2BAK .)
rewrites: 37678 in 184ms cpu (189ms real) (203997 rewrites/second)
Checking 2BAK ||- mutex => O mutex ...
Proof obligations generated:  18
Proof obligations discharged: 16
The following proof obligations need to be discharged:
 cpo for rule p1_w and equation p1_w-p2_c :
  false = true
  if #1:Nat <= #3:Nat = true .
 cpo for rule p2_w and equation p1_c-p2_w :
  false = true
  if #3:Nat < #2:Nat = true .
Maude> _
```

```
(analyze-stable <pred> in <fmodule> <module> .)
```

## Example (Analyzing $\mathcal{T}_{\mathtt{2BAK}} \models \mathtt{mutex} \Rightarrow \Box\mathtt{mutex}$)

```
Maude> (analyze-stable mutex(S:State) in 2BAK-PROPS 2BAK .)
rewrites: 37678 in 184ms cpu (189ms real) (203997 rewrites/second)
Checking 2BAK ||- mutex => O mutex ...
Proof obligations generated:  18
Proof obligations discharged: 16
The following proof obligations need to be discharged:
 cpo for rule p1_w and equation p1_w-p2_c :
  false = true
  if #1:Nat <= #3:Nat = true .
 cpo for rule p2_w and equation p1_c-p2_w :
  false = true
  if #3:Nat < #2:Nat = true .
Maude> _
```

## Does $\mathcal{T}_{\mathtt{2BAK}} \models \mathtt{init} \Rightarrow \Box\mathtt{mutex}$?

Since $\mathcal{E}_\Pi$ protects `Bool` and both conditions above are satisfiable, we can safely conclude $\mathcal{T}_{\mathtt{2BAK}} \not\models \mathtt{mutex} \Rightarrow \Box\mathtt{mutex}$, i.e., $\mathcal{T}_{\mathtt{2BAK}}$ is NOT `mutex`-stable.

# InvA

## Strengthening for $\mathcal{T}_{\texttt{2BAK}} \models \texttt{init} \Rightarrow \Box\texttt{mutex}$

We can strengthen `mutex` to the stronger state predicate `aux` defined as follows:

```
fmod 2BAK-PROPS-EXT is
  protecting 2BAK-PROPS .
  vars M N : Nat .
  op aux : State -> [Bool] .
  eq [p1_s-p2_s] : aux(< sleep, M, sleep, N >) = true .
  eq [p1_w-p2_s] : aux(< wait, M, sleep, N >) = true .
  eq [p1_c-p2_s] : aux(< crit, M, sleep, N >) = true .
  eq [p1_s-p2_w] : aux(< sleep, M, wait, N >) = true .
  eq [p1_w-p2_w] : aux(< wait, M, wait, N >) = true .
  eq [p1_c-p2_w] : aux(< crit, M, wait, N >) = M <= N .  --- strengthening
  eq [p1_s-p2_c] : aux(< sleep, M, crit, N >) = true .
  eq [p1_c-p2_c] : aux(< crit, M, crit, N >) = false .
  eq [p1_w-p2_c] : aux(< wait, M, crit, N >) = N < M .   --- strengthening
endfm
```

# InvA

## Strengthening for $\mathcal{T}_{\mathtt{2BAK}} \models \mathtt{init} \Rightarrow \Box\mathtt{mutex}$

We can strengthen `mutex` to the stronger state predicate `aux` defined as follows:

```
fmod 2BAK-PROPS-EXT is
  protecting 2BAK-PROPS .
  vars M N : Nat .
  op aux : State -> [Bool] .
  eq [p1_s-p2_s] : aux(< sleep, M, sleep, N >) = true .
  eq [p1_w-p2_s] : aux(< wait, M, sleep, N >) = true .
  eq [p1_c-p2_s] : aux(< crit, M, sleep, N >) = true .
  eq [p1_s-p2_w] : aux(< sleep, M, wait, N >) = true .
  eq [p1_w-p2_w] : aux(< wait, M, wait, N >) = true .
  eq [p1_c-p2_w] : aux(< crit, M, wait, N >) = M <= N .  --- strengthening
  eq [p1_s-p2_c] : aux(< sleep, M, crit, N >) = true .
  eq [p1_c-p2_c] : aux(< crit, M, crit, N >) = false .
  eq [p1_w-p2_c] : aux(< wait, M, crit, N >) = N < M .   --- strengthening
endfm
```

We can use this strengthening as follows:

$$\frac{\mathcal{T}_{\mathtt{2BAK}} \models \mathtt{init} \Rightarrow \mathtt{aux} \qquad \dfrac{\mathcal{T}_{\mathtt{2BAK}} \models \mathtt{aux} \Rightarrow \bigcirc\mathtt{aux}}{\mathcal{T}_{\mathtt{2BAK}} \models \mathtt{aux} \Rightarrow \Box\mathtt{aux}} \qquad \mathcal{T}_{\mathtt{2BAK}} \models \mathtt{aux} \Rightarrow \mathtt{mutex}}{\mathcal{T}_{\mathtt{2BAK}} \models \mathtt{init} \Rightarrow \Box\mathtt{mutex}}$$

# InvA

## Strengthening for $\mathcal{T}_{2BAK} \models \texttt{init} \Rightarrow \Box\texttt{mutex}$

## Example ($\mathcal{T}_{2BAK} \models \texttt{init} \Rightarrow \texttt{aux}$)

```
Maude> (analyze init(S:State) implies aux(S:State) in 2BAK-PROPS-EXT .)
rewrites: 4993 in 29ms cpu (31ms real) (166716 rewrites/second)
Checking 2BAK-PROPS-EXT |- init => aux ...
Proof obligations generated:  1
Proof obligations discharged: 1
Success!
```

## Example ($\mathcal{T}_{2BAK} \models \texttt{aux} \Rightarrow \texttt{mutex}$)

```
Maude> (analyze aux(S:State) implies mutex(S:State) in 2BAK-PROPS-EXT .)
rewrites: 10883 in 39ms cpu (41ms real) (276106 rewrites/second)
Checking 2BAK-PROPS-EXT |- aux => mutex ...
Proof obligations generated:  9
Proof obligations discharged: 9
Success!
```

$$\dfrac{\dfrac{\checkmark}{\mathcal{T}_{2BAK} \models \texttt{init} \Rightarrow \texttt{aux}} \qquad \dfrac{\mathcal{T}_{2BAK} \models \texttt{aux} \Rightarrow \bigcirc\texttt{aux}}{\mathcal{T}_{2BAK} \models \texttt{aux} \Rightarrow \Box\texttt{aux}} \qquad \dfrac{\checkmark}{\mathcal{T}_{2BAK} \models \texttt{aux} \Rightarrow \texttt{mutex}}}{\mathcal{T}_{2BAK} \models \texttt{init} \Rightarrow \Box\texttt{mutex}}$$

# InvA

## Strengthening for $\mathcal{T}_{\texttt{2BAK}} \models \texttt{init} \Rightarrow \Box\texttt{mutex}$

## Example ($\mathcal{T}_{\texttt{2BAK}} \models \texttt{aux} \Rightarrow \bigcirc\texttt{aux}$)

```
Maude> (analyze-stable aux(S:State) in 2BAK-PROPS-EXT 2BAK .)
rewrites: 39839 in 171ms cpu (173ms real) (232067 rewrites/second)
Checking 2BAK ||- aux => O aux ...
Proof obligations generated:  18
Proof obligations discharged: 16
The following proof obligations need to be discharged:
 po for rule p1_s and equation p1_s-p2_c :
  #3:Nat < s #3:Nat = true .
 po for rule p2_s and equation p1_c-p2_s :
  #2:Nat <= s #2:Nat = true .
```

$$\dfrac{\dfrac{\checkmark}{\mathcal{T}_{\texttt{2BAK}} \models \texttt{init} \Rightarrow \texttt{aux}} \qquad \dfrac{\dfrac{}{\mathcal{T}_{\texttt{2BAK}} \models \texttt{aux} \Rightarrow \bigcirc\texttt{aux}}}{\mathcal{T}_{\texttt{2BAK}} \models \texttt{aux} \Rightarrow \Box\texttt{aux}} \qquad \dfrac{\checkmark}{\mathcal{T}_{\texttt{2BAK}} \models \texttt{aux} \Rightarrow \texttt{mutex}}}{\mathcal{T}_{\texttt{2BAK}} \models \texttt{init} \Rightarrow \Box\texttt{mutex}}$$

# InvA

## Strengthening for $\mathcal{T}_{\texttt{2BAK}} \models \texttt{init} \Rightarrow \Box\texttt{mutex}$

## Example ($\mathcal{T}_{\texttt{2BAK}} \models \texttt{aux} \Rightarrow \bigcirc\texttt{aux}$)

```
Maude> (analyze-stable aux(S:State) in 2BAK-PROPS-EXT 2BAK .)
rewrites: 39839 in 171ms cpu (173ms real) (232067 rewrites/second)
Checking 2BAK |- aux => O aux ...
Proof obligations generated:  18
Proof obligations discharged: 16
The following proof obligations need to be discharged:
 po for rule p1_s and equation p1_s-p2_c :
  #3:Nat < s #3:Nat = true .
 po for rule p2_s and equation p1_c-p2_s :
  #2:Nat <= s #2:Nat = true .
```

$$\dfrac{\dfrac{\checkmark}{\mathcal{T}_{\texttt{2BAK}} \models \texttt{init} \Rightarrow \texttt{aux}} \qquad \dfrac{\dfrac{\checkmark\text{(with the ITP)}}{\mathcal{T}_{\texttt{2BAK}} \models \texttt{aux} \Rightarrow \bigcirc\texttt{aux}}}{\mathcal{T}_{\texttt{2BAK}} \models \texttt{aux} \Rightarrow \Box\texttt{aux}} \qquad \dfrac{\checkmark}{\mathcal{T}_{\texttt{2BAK}} \models \texttt{aux} \Rightarrow \texttt{mutex}}}{\mathcal{T}_{\texttt{2BAK}} \models \texttt{init} \Rightarrow \Box\texttt{mutex}}$$