Proving Safety Properties of Rewrite Theories

Camilo Rocha   José Meseguer

Department of Computer Science
University of Illinois at Urbana-Champaign

4th Conference on Algebra and Coalgebra in Computer Science
August 31, 2011
Winchester, UK

Main Contribution

A deductive method and an infrastructure for proving **safety properties** of rewrite theories

Main Contribution

A deductive method and an infrastructure for proving **safety properties** of rewrite theories

- a proof system that reduces the verification task of proving **stability** and **invariance** properties of concurrent rewrites to a first-order equational inductive theorem proving task

Main Contribution

A deductive method and an infrastructure for proving **safety properties** of rewrite theories

- a proof system that reduces the verification task of proving **stability** and **invariance** properties of concurrent rewrites to a first-order equational inductive theorem proving task
  - it can be applied to infinite-state systems and can assume infinite sets of initial states

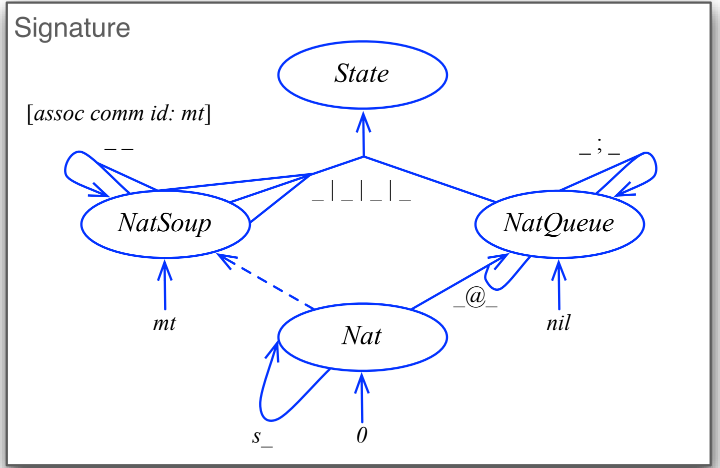
Main Contribution

A deductive method and an infrastructure for proving **safety properties** of rewrite theories

- a proof system that reduces the verification task of proving **stability** and **invariance** properties of concurrent rewrites to a first-order equational inductive theorem proving task
  - it can be applied to infinite-state systems and can assume infinite sets of initial states
- the Maude Invariant Analyzer Tool (**InvA**) is an implementation in Maude of the inference system above that can automatically discharge many proof obligations without user intervention

Motivation

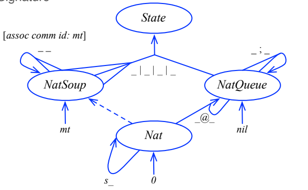
The QLOCK Protocol



Motivation

The QLOCK Protocol

Signature



Transitions

$n, n' : Nat \qquad Si, Sw, Sc : NatSoup \qquad Q : NatQueue$

$$n\ Si \mid Sw \mid Sc \mid Q \longrightarrow Si \mid n\ Sw \mid Sc \mid Q ; (n @ nil)$$

$$Si \mid n\ Sw \mid Sc \mid n @ Q \longrightarrow Si \mid Sw \mid n\ Sc \mid n @ Q$$

$$Si \mid Sw \mid n\ Sc \mid n' @ Q \longrightarrow n\ Si \mid Sw \mid Sc \mid Q$$

Motivation

Mutual Exclusion in the QLOCK Protocol

Consider an initial state in which all processes are in state `inactive`. How do we check that there is at most one process in the critical section at any point of execution, i.e., that QLOCK satisfies the mutual exclusion property?

Transitions

$n, n' : Nat \qquad Si, Sw, Sc : NatSoup \qquad Q : NatQueue$

$$n\ Si \mid Sw \mid Sc \mid Q \longrightarrow Si \mid n\ Sw \mid Sc \mid Q\ ;\ (n\ @\ nil)$$

$$Si \mid n\ Sw \mid Sc \mid n\ @\ Q \longrightarrow Si \mid Sw \mid n\ Sc \mid n\ @\ Q$$

$$Si \mid Sw \mid n\ Sc \mid n'\ @\ Q \longrightarrow n\ Si \mid Sw \mid Sc \mid Q$$

Outline

Rewrite Theories

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

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

Rewrite Theories

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

- the equations E are ground Church-Rosser, ground strongly normalizing, and ground coherent w.r.t. R modulo A
- it is topmost, i.e., Σ has a topmost sort $\mathfrak{s}$ and each rule $l \to r$ if $cond \in R$ 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 $\mathcal{R}$

State Predicates

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

State Predicates

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

- Σ_Π contains Σ, two sorts $Bool \leq [Bool]$ with constants $\top$ and $\perp$ of sort $Bool$, predicate symbols $P : \mathfrak{s} \longrightarrow [Bool]$ for each $P \in \Pi$, and optionally some auxiliary function symbols

State Predicates

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

- Σ_Π contains Σ, two sorts $Bool \leq [Bool]$ with constants $\top$ and $\bot$ of sort $Bool$, predicate symbols $P : \mathfrak{s} \longrightarrow [Bool]$ for each $P \in \Pi$, and optionally some auxiliary function symbols
- the equations E_Π define the predicate symbols in Σ_Π and the auxiliary function symbols, and they protect both $\mathcal{E}_\mathcal{R}$ and the theory BOOL specifying the sort $Bool$, $\top$, $\bot$, and the Boolean operations

State Predicates

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

- Σ_Π contains Σ, two sorts $Bool \leq [Bool]$ with constants $\top$ and $\bot$ of sort $Bool$, predicate symbols $P : \mathfrak{s} \longrightarrow [Bool]$ for each $P \in \Pi$, and optionally some auxiliary function symbols
- the equations E_Π define the predicate symbols in Σ_Π and the auxiliary function symbols, and they protect both $\mathcal{E}_\mathcal{R}$ and the theory BOOL specifying the sort $Bool$, $\top$, $\bot$, and the Boolean operations
- the equations $E \cup E_\Pi$ are ground confluent, ground strongly normalizing, and ground coherent w.r.t. R modulo A

Temporal Operators

Next ($\bigcirc$)

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

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

- we define

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

Temporal Operators

Always ($\Box$)

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

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

- we define

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

Outline

Safety Properties

Main Idea

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 Π. I denotes the set of **initial states** and P is the **invariant**.

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:

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

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:

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

Mutual Exclusion for QLOCK

Predicates

$n\ n' : Nat \qquad Si, Sw, Sc : NatSoup \qquad Q : NatQueue$

$init : State \rightarrow [Bool]$

$$init(\ Si \mid mt \mid mt \mid nil\ ) = set?(Si)$$

$mutex : State \rightarrow [Bool]$

$$mutex(\ Si \mid Sw \mid mt \mid Q\ ) = \top$$
$$mutex(\ Si \mid Sw \mid n \mid Q\ ) = \top$$
$$mutex(\ Si \mid Sw \mid n\ n'\ Sc \mid Q\ ) = \bot$$

Mutual Exclusion for QLOCK

Predicates

$n\ n' : Nat \qquad Si, Sw, Sc : NatSoup \qquad Q : NatQueue$

$init : State \to [Bool]$

$$init(\ Si \mid mt \mid mt \mid nil\ ) = set?(Si)$$

$mutex : State \to [Bool]$

$$mutex(\ Si \mid Sw \mid mt \mid Q\ ) = \top$$
$$mutex(\ Si \mid Sw \mid n \mid Q\ ) = \top$$
$$mutex(\ Si \mid Sw \mid n\ n'\ Sc \mid Q\ ) = \bot$$

Mutual exclusion for QLOCK can be expressed by the invariant property

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

Outline

The Proof System

The proof system comprises two 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 $\mathcal{E}_{\Pi}$

Reduction Inference Rules

For Stability and Invariance Properties

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:

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

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

Proving the Mutual Exclusion for QLOCK

Composing $\textsc{St}$ and $\textsc{Inv}$ for the mutual exclusion of QLOCK, we get:

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

Proving the Mutual Exclusion for QLOCK

Composing $\text{S}\textsc{t}$ and $\text{I}\textsc{nv}$ for the mutual exclusion of QLOCK, we get:

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

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:

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

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

Streamlining $\textsc{Rew-D}$

In the inference rule $\textsc{Rew-D}$, how can we prove for $l \to r$ if $C \in R$ that

$$\mathcal{E}_{\Pi} \vdash_{\text{ind}} (\forall \mathbf{y} : \mathbf{s})\, (P(l) = \top \wedge C) \;\Rightarrow\; P(r) = \top\,?$$

Streamlining $\textsc{Rew-D}$

In the inference rule $\textsc{Rew-D}$, how can we prove for $l \to r$ if $C \in R$ that

$$\mathcal{E}_{\Pi} \vdash_{\text{ind}} (\forall \mathbf{y} : \mathbf{s})\,(P(l) = \top \wedge C) \;\Rightarrow\; P(r) = \top\,?$$

Key idea: 1-step narrowing with the predicate P in the condition!!!

Streamlining $\textsc{Rew-D}$

In the inference rule $\textsc{Rew-D}$, how can we prove for $l \to r$ if $C \in R$ that

$$\mathcal{E}_\Pi \vdash_{\mathrm{ind}} (\forall \mathbf{y} : \mathbf{s})\, (P(l) = \top \wedge C) \;\Rightarrow\; P(r) = \top \,?$$

Key idea: 1-step narrowing with the predicate P in the condition!!!

- assume l and each v in $P(v) = w$ if $D \in E_\Pi$ are free constructor terms modulo A

Streamlining $\textsc{Rew-D}$

In the inference rule $\textsc{Rew-D}$, how can we prove for $l \to r$ if $C \in R$ that

$$\mathcal{E}_\Pi \vdash_{\text{ind}} (\forall \mathbf{y} : \mathbf{s})\, (P(l) = \top \wedge C) \;\Rightarrow\; P(r) = \top\,?$$

Key idea: 1-step narrowing with the predicate P in the condition!!!

- assume l and each v in $P(v) = w$ if $D \in E_\Pi$ are free constructor terms modulo A
- then, for each ground substitution α, $\mathcal{E}_\Pi \vdash P(l\alpha) = \top$ if and only if there is $P(v) = w$ if $D \in E_\Pi$ and substitutions θ and γ, such that $l\theta =_A v\theta$, $\alpha =_{E \cup A} \theta\gamma$, and $\mathcal{E}_\Pi \vdash C\theta\gamma \wedge D\theta\gamma \wedge w\theta\gamma = \top$

Streamlining $\textsc{Rew-D}$

In the inference rule $\textsc{Rew-D}$, how can we prove for $l \to r$ if $C \in R$ that

$$\mathcal{E}_\Pi \vdash_{\mathrm{ind}} (\forall \mathbf{y} : \mathbf{s})\,(P(l) = \top \wedge C) \;\Rightarrow\; P(r) = \top\;?$$

Key idea: 1-step narrowing with the predicate P in the condition!!!

- assume l and each v in $P(v) = w$ if $D \in E_\Pi$ are free constructor terms modulo A
- then, for each ground substitution α, $\mathcal{E}_\Pi \vdash P(l\alpha) = \top$ if and only if there is $P(v) = w$ if $D \in E_\Pi$ and substitutions θ and γ, such that $l\theta =_A v\theta$, $\alpha =_{E \cup A} \theta\gamma$, and $\mathcal{E}_\Pi \vdash C\theta\gamma \wedge D\theta\gamma \wedge w\theta\gamma = \top$
- let $\mathrm{CSU}_A(l = v)$ be the set of most general A-unifiers of the Σ-equation $l = v$

Streamlining $\textsc{Rew-D}$

In the inference rule $\textsc{Rew-D}$, how can we prove for $l \to r \text{ if } C \in R$ that

$$\mathcal{E}_\Pi \vdash_{\mathrm{ind}} (\forall \mathbf{y} : \mathbf{s})\, (P(l) = \top \wedge C) \;\Rightarrow\; P(r) = \top\ ?$$

Key idea: 1-step narrowing with the predicate P in the condition!!!

- assume l and each v in $P(v) = w \text{ if } D \in E_\Pi$ are free constructor terms modulo A
- then, for each ground substitution α, $\mathcal{E}_\Pi \vdash P(l\alpha) = \top$ if and only if there is $P(v) = w \text{ if } D \in E_\Pi$ and substitutions θ and γ, such that $l\theta =_A v\theta$, $\alpha =_{E \cup A} \theta\gamma$, and $\mathcal{E}_\Pi \vdash C\theta\gamma \wedge D\theta\gamma \wedge w\theta\gamma = \top$
- let $\mathrm{CSU}_A(l = v)$ be the set of most general A-unifiers of the Σ-equation $l = v$

Then, we can streamline $\textsc{Rew-D}$ as follows:

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

Outline

Strengthening of Invariants

Strengthening of invariants is an important technique for verifying safety properties.

- a **strengthening** for $\mathcal{T}_\mathcal{R} \models I \Rightarrow \Box P$ is a state predicate $Q \in \Pi$ such that $\mathcal{T}_\mathcal{R} \models I \Rightarrow \Box Q$ and Q can be used to prove $\mathcal{T}_\mathcal{R} \models I \Rightarrow \Box P$
- state predicate Q is the result of a gradual refinement of a too-weakly defined P for which $\mathcal{T}_\mathcal{R} \models I \Rightarrow \Box P$ cannot be proven directly using the inference rules mentioned so far

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:

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

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

Strengthening for the Mutual Exclusion of QLOCK

Strengthening

$n\ n' : Nat \qquad Si, Sw, Sc : NatSoup \qquad Q : NatQueue$

$aux : State \rightarrow [Bool]$

$$aux(\,Si \mid Sw \mid mt \mid Q\,) = set?(Si\ Sw)$$

$$aux(\,Si \mid Sw \mid n \mid n\ @\ Q\,) = set?(Si\ Sw\ n)$$

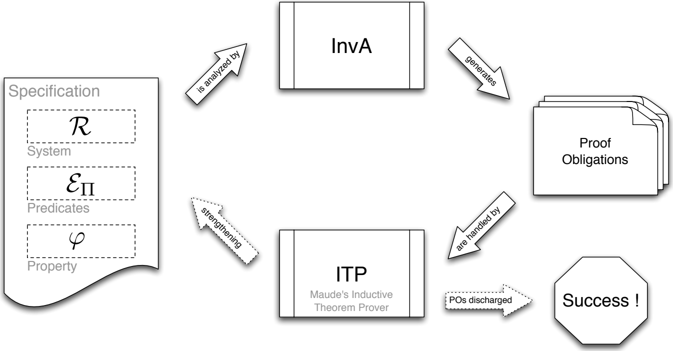
$$aux(\,Si \mid Sw \mid n\ n'\ Sc \mid Q\,) = \bot$$

Outline

InvA

Methodology

The InvA Tool is an **interactive environment**, implemented in Full Maude, that assists in the task of proving stability and invariance properties for $\mathcal{T}_{\mathcal{R}}$ by generating equational subgoals



InvA

Proving $\mathcal{T}_{\texttt{QLOCK}} \models init \Rightarrow \Box mutex$

Proof

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

InvA

Proving $\mathcal{T}_{\texttt{QLOCK}} \models init \Rightarrow \Box mutex$

Proof

```
(analyze init(S:State) implies aux(S:State)
  in QLOCK-PROPS .)
rewrites: 4265 in 10ms cpu (11ms real)
Checking QLOCK-PROPS ||- init => aux ...
Proof obligations generated:  1
Proof obligations discharged: 1
Success!
```

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

InvA

Proving $\mathcal{T}_{\texttt{QLOCK}} \models init \Rightarrow \Box mutex$

Proof

```
(analyze init(S:State) implies aux(S:State)
  in QLOCK-PROPS .)
rewrites: 4265 in 10ms cpu (11ms real)
Checking QLOCK-PROPS ||- init => aux ...
Proof obligations generated:  1
Proof obligations discharged: 1
Success!
```

```
(analyze-stable aux(S:State)
  in QLOCK-PROPS QLOCK .)
rewrites: 42333 in 95ms cpu (100ms real)
Checking QLOCK ||- aux => O aux ...
Proof obligations generated:  12
Proof obligations discharged: 12
Success!
```

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

InvA

Proving $\mathcal{T}_{\texttt{QLOCK}} \models init \Rightarrow \Box mutex$

Proof

```
(analyze init(S:State) implies aux(S:State)
  in QLOCK-PROPS .)
rewrites: 4265 in 10ms cpu (11ms real)
Checking QLOCK-PROPS ||- init => aux ...
Proof obligations generated:  1
Proof obligations discharged: 1
Success!
```

$$\mathcal{T}_{\texttt{QLOCK}} \models init \Rightarrow aux$$

```
(analyze-stable aux(S:State)
  in QLOCK-PROPS QLOCK .)
rewrites: 42333 in 95ms cpu (100ms real)
Checking QLOCK ||- aux => O aux ...
Proof obligations generated:  12
Proof obligations discharged: 12
Success!
```

$$\frac{\mathcal{T}_{\texttt{QLOCK}} \models aux \Rightarrow \bigcirc aux}{\mathcal{T}_{\texttt{QLOCK}} \models aux \Rightarrow \Box aux}$$

```
(analyze aux(S:State) implies mutex(S:State)
  in QLOCK-PROPS .)
rewrites: 10007 in 21ms cpu (22ms real)
Checking QLOCK-PROPS ||- aux => mutex ...
Proof obligations generated:  3
Proof obligations discharged: 3
Success!
```

$$\mathcal{T}_{\texttt{QLOCK}} \models aux \Rightarrow mutex$$

$$\frac{\mathcal{T}_{\texttt{QLOCK}} \models init \Rightarrow aux \qquad \mathcal{T}_{\texttt{QLOCK}} \models aux \Rightarrow \Box aux \qquad \mathcal{T}_{\texttt{QLOCK}} \models aux \Rightarrow mutex}{\mathcal{T}_{\texttt{QLOCK}} \models init \Rightarrow \Box mutex}$$

InvA

Proving $\mathcal{T}_{\mathtt{QLOCK}} \models init \Rightarrow \Box mutex$

Proof

```
(analyze init(S:State) implies aux(S:State)
  in QLOCK-PROPS .)
rewrites: 4265 in 10ms cpu (11ms real)
Checking QLOCK-PROPS ||- init => aux ...
Proof obligations generated:  1
Proof obligations discharged: 1
Success!
```

```
(analyze-stable aux(S:State)
  in QLOCK-PROPS QLOCK .)
rewrites: 42333 in 95ms cpu (100ms real)
Checking QLOCK ||- aux => O aux ...
Proof obligations generated:  12
Proof obligations discharged: 12
Success!
```

```
(analyze aux(S:State) implies mutex(S:State)
  in QLOCK-PROPS .)
rewrites: 10007 in 21ms cpu (22ms real)
Checking QLOCK-PROPS ||- aux => mutex ...
Proof obligations generated:  3
Proof obligations discharged: 3
Success!
```

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

A technical report, the InvA, and more examples are available at

http://camilorocha.info

Thank you!