Program Verification: Lecture 23

José Meseguer

University of Illinois
at Urbana-Champaign

Deductive Verification of Distributed Systems

Deductive Verification of Distributed Systems

Model checking of invariants and LTL properties is very useful.
But it has some limitations:

Deductive Verification of Distributed Systems

Model checking of invariants and LTL properties is very useful.
But it has some limitations:

1. Explicit-state model checking algorithms can only deal with *finite* sets of reachable states.

Deductive Verification of Distributed Systems

Model checking of invariants and LTL properties is very useful.
But it has some limitations:

1. Explicit-state model checking algorithms can only deal with *finite* sets of reachable states.
2. Even if an *equational abstraction* can be used to make the set of reachable states finite,

Deductive Verification of Distributed Systems

Model checking of invariants and LTL properties is very useful.
But it has some limitations:

1. Explicit-state model checking algorithms can only deal with *finite* sets of reachable states.
2. Even if an *equational abstraction* can be used to make the set of reachable states finite, the *set of abstracted initial states* of interest may be *infinite*.

Deductive Verification of Distributed Systems

Model checking of invariants and LTL properties is very useful.
But it has some limitations:

1. Explicit-state model checking algorithms can only deal with *finite* sets of reachable states.
2. Even if an *equational abstraction* can be used to make the set of reachable states finite, the *set of abstracted initial states* of interest may be *infinite*.
3. More generally, *state infinity* can block the use of explicit-state model checking in two different ways:

Deductive Verification of Distributed Systems

Model checking of invariants and LTL properties is very useful.
But it has some limitations:

1. Explicit-state model checking algorithms can only deal with *finite* sets of reachable states.
2. Even if an *equational abstraction* can be used to make the set of reachable states finite, the *set of abstracted initial states* of interest may be *infinite*.
3. More generally, *state infinity* can block the use of explicit-state model checking in two different ways:
   - The number of states reachable from a given states is *infinite*.

Deductive Verification of Distributed Systems

Model checking of invariants and LTL properties is very useful. But it has some limitations:

1. Explicit-state model checking algorithms can only deal with *finite* sets of reachable states.
2. Even if an *equational abstraction* can be used to make the set of reachable states finite, the *set of abstracted initial states* of interest may be *infinite*.
3. More generally, *state infinity* can block the use of explicit-state model checking in two different ways:
   - The number of states reachable from a given states is *infinite*.
   - The number of initial states is *infinite*.

Deductive Verification of Distributed Systems

Model checking of invariants and LTL properties is very useful.
But it has some limitations:

1. Explicit-state model checking algorithms can only deal with *finite* sets of reachable states.
2. Even if an *equational abstraction* can be used to make the set of reachable states finite, the *set of abstracted initial states* of interest may be *infinite*.
3. More generally, *state infinity* can block the use of explicit-state model checking in two different ways:
    - The number of states reachable from a given states is *infinite*.
    - The number of initial states is *infinite*.

This suggests two other options: (1) *symbolic model checking* (automatic)

Deductive Verification of Distributed Systems

Model checking of invariants and LTL properties is very useful. But it has some limitations:

1. Explicit-state model checking algorithms can only deal with *finite* sets of reachable states.
2. Even if an *equational abstraction* can be used to make the set of reachable states finite, the *set of abstracted initial states* of interest may be *infinite*.
3. More generally, *state infinity* can block the use of explicit-state model checking in two different ways:
   - The number of states reachable from a given states is *infinite*.
   - The number of initial states is *infinite*.

This suggests two other options: (1) *symbolic model checking* (automatic) and (2) *deductive methods* based on theorem proving (more general).

Deductive Verification of Distributed Systems

Model checking of invariants and LTL properties is very useful.
But it has some limitations:

1. Explicit-state model checking algorithms can only deal with *finite* sets of reachable states.
2. Even if an *equational abstraction* can be used to make the set of reachable states finite, the *set of abstracted initial states* of interest may be *infinite*.
3. More generally, *state infinity* can block the use of explicit-state model checking in two different ways:
   - The number of states reachable from a given states is *infinite*.
   - The number of initial states is *infinite*.

This suggests two other options: (1) *symbolic model checking* (automatic) and (2) *deductive methods* based on theorem proving (more general). We will explore *logics* for option (2) in this lecture.

Hoare Logic

Suppose that a concurrent system has been specified by a rewrite theory $\mathcal{R}$,

Hoare Logic

Suppose that a concurrent system has been specified by a rewrite theory $\mathcal{R}$, a top sort $State$ of states has been chosen, and

Hoare Logic

Suppose that a concurrent system has been specified by a rewrite theory $\mathcal{R}$, a top sort $State$ of states has been chosen, and some *state predicates* Π have also been defined.

Hoare Logic

Suppose that a concurrent system has been specified by a rewrite theory $\mathcal{R}$, a top sort $State$ of states has been chosen, and some *state predicates* Π have also been defined.

We can then define properties of $\mathcal{R}$ in *Hoare Logic* by means of so-called *Hoare Triples* of the form:

Hoare Logic

Suppose that a concurrent system has been specified by a rewrite theory $\mathcal{R}$, a top sort $State$ of states has been chosen, and some *state predicates* Π have also been defined.

We can then define properties of $\mathcal{R}$ in *Hoare Logic* by means of so-called *Hoare Triples* of the form: $\{A\}\ \mathcal{R}\ \{B\}$,

Hoare Logic

Suppose that a concurrent system has been specified by a rewrite theory $\mathcal{R}$, a top sort $State$ of states has been chosen, and some *state predicates* Π have also been defined.

We can then define properties of $\mathcal{R}$ in *Hoare Logic* by means of so-called *Hoare Triples* of the form: $\{A\}\ \mathcal{R}\ \{B\}$, where A and B are *formulas* on predicates Π defining *sets of states* $[\![A]\!]$ and $[\![B]\!]$.

Hoare Logic

Suppose that a concurrent system has been specified by a rewrite theory $\mathcal{R}$, a top sort $State$ of states has been chosen, and some *state predicates* Π have also been defined.

We can then define properties of $\mathcal{R}$ in *Hoare Logic* by means of so-called *Hoare Triples* of the form: $\{A\}\ \mathcal{R}\ \{B\}$, where A and B are *formulas* on predicates Π defining *sets of states* $[\![A]\!]$ and $[\![B]\!]$.

A is called the *precondition* and B the *postcondition*.

Hoare Logic

Suppose that a concurrent system has been specified by a rewrite theory $\mathcal{R}$, a top sort $State$ of states has been chosen, and some *state predicates* Π have also been defined.

We can then define properties of $\mathcal{R}$ in *Hoare Logic* by means of so-called *Hoare Triples* of the form: $\{A\}\ \mathcal{R}\ \{B\}$, where A and B are *formulas* on predicates Π defining *sets of states* $[\![A]\!]$ and $[\![B]\!]$.

A is called the *precondition* and B the *postcondition*. The Hoare triple $\{A\}\ \mathcal{R}\ \{B\}$ is a *partial correctness assertion* about $\mathcal{R}$:

Hoare Logic

Suppose that a concurrent system has been specified by a rewrite theory $\mathcal{R}$, a top sort $State$ of states has been chosen, and some *state predicates* Π have also been defined.

We can then define properties of $\mathcal{R}$ in *Hoare Logic* by means of so-called *Hoare Triples* of the form: $\{A\}\ \mathcal{R}\ \{B\}$, where A and B are *formulas* on predicates Π defining *sets of states* $[\![A]\!]$ and $[\![B]\!]$.

A is called the *precondition* and B the *postcondition*. The Hoare triple $\{A\}\ \mathcal{R}\ \{B\}$ is a *partial correctness assertion* about $\mathcal{R}$:

- for each state $[u_0] \in \mathcal{T}_{\mathcal{R},State}$, if $[u_0]$ satisfies the precondition A, then,

Hoare Logic

Suppose that a concurrent system has been specified by a rewrite theory $\mathcal{R}$, a top sort $State$ of states has been chosen, and some *state predicates* Π have also been defined.

We can then define properties of $\mathcal{R}$ in *Hoare Logic* by means of so-called *Hoare Triples* of the form: $\{A\}\ \mathcal{R}\ \{B\}$, where A and B are *formulas* on predicates Π defining *sets of states* $[\![A]\!]$ and $[\![B]\!]$.

A is called the *precondition* and B the *postcondition*. The Hoare triple $\{A\}\ \mathcal{R}\ \{B\}$ is a *partial correctness assertion* about $\mathcal{R}$:

- for each state $[u_0] \in \mathcal{T}_{\mathcal{R},State}$, if $[u_0]$ satisfies the precondition A, then,
- for each *terminating* sequence of transitions

$$[u_0] \rightarrow_{\mathcal{R}} [u_1] \ldots [u_{n-1}] \rightarrow_{\mathcal{R}} [u_n]$$

the terminating state $[u_n]$ satisfies postcondition B.

Pattern Predicates and Parameters

What formulas A and B shall we use in a Hoare triple
$\{A\}\ \mathcal{R}\ \{B\}$?

Pattern Predicates and Parameters

What formulas A and B shall we use in a Hoare triple $\{A\}\,\mathcal{R}\,\{B\}$? Assuming $\mathcal{R} = (\Sigma, B, R)$ has constructors Ω, we can use *pattern predicates* of the form $u \mid \varphi$ where u is an Ω-term of sort $State$ and φ is a Σ-condition.

Pattern Predicates and Parameters

What formulas A and B shall we use in a Hoare triple $\{A\}\,\mathcal{R}\,\{B\}$? Assuming $\mathcal{R} = (\Sigma, B, R)$ has constructors Ω, we can use *pattern predicates* of the form $u \mid \varphi$ where u is an Ω-term of sort $State$ and φ is a Σ-condition. Then $u \mid \varphi$ denotes the set of its ground instance states:

Pattern Predicates and Parameters

What formulas A and B shall we use in a Hoare triple $\{A\}\,\mathcal{R}\,\{B\}$? Assuming $\mathcal{R} = (\Sigma, B, R)$ has constructors Ω, we can use *pattern predicates* of the form $u \mid \varphi$ where u is an Ω-term of sort $State$ and φ is a Σ-condition. Then $u \mid \varphi$ denotes the set of its ground instance states:

$$[\![u \mid \varphi]\!] = \{[u\rho]_B \mid \rho \in [X \to T_\Omega] \ \wedge \ E \cup B \models \varphi\rho\}.$$

Pattern Predicates and Parameters

What formulas A and B shall we use in a Hoare triple $\{A\}\ \mathcal{R}\ \{B\}$? Assuming $\mathcal{R} = (\Sigma, B, R)$ has constructors Ω, we can use *pattern predicates* of the form $u \mid \varphi$ where u is an Ω-term of sort $State$ and φ is a Σ-condition. Then $u \mid \varphi$ denotes the set of its ground instance states:

$$[\![u \mid \varphi]\!] = \{[u\rho]_B \mid \rho \in [X \to T_\Omega] \ \wedge \ E \cup B \models \varphi\rho\}.$$

Let $Y = vars(A) \cap vars(B)$. Then we call Y the *parameters* of the Hoare triple $\{A\}\ \mathcal{R}\ \{B\}$.

Pattern Predicates and Parameters

What formulas A and B shall we use in a Hoare triple $\{A\}\ \mathcal{R}\ \{B\}$? Assuming $\mathcal{R} = (\Sigma, B, R)$ has constructors Ω, we can use *pattern predicates* of the form $u \mid \varphi$ where u is an Ω-term of sort $State$ and φ is a Σ-condition. Then $u \mid \varphi$ denotes the set of its ground instance states:

$$[\![u \mid \varphi]\!] = \{[u\rho]_B \mid \rho \in [X \to T_\Omega] \ \wedge \ E \cup B \models \varphi\rho\}.$$

Let $Y = vars(A) \cap vars(B)$. Then we call Y the *parameters* of the Hoare triple $\{A\}\ \mathcal{R}\ \{B\}$. Such a triple is in fact *universally quantified* on its parameters. That is, $\{A\}\ \mathcal{R}\ \{B\}$ implicitly means: $(\forall Y)\ \{A\}\ \mathcal{R}\ \{B\}$.

Pattern Predicates and Parameters

What formulas A and B shall we use in a Hoare triple $\{A\}\ \mathcal{R}\ \{B\}$? Assuming $\mathcal{R} = (\Sigma, B, R)$ has constructors Ω, we can use *pattern predicates* of the form $u \mid \varphi$ where u is an Ω-term of sort $State$ and φ is a Σ-condition. Then $u \mid \varphi$ denotes the set of its ground instance states:

$$[\![u \mid \varphi]\!] = \{[u\rho]_B \mid \rho \in [X \to T_\Omega] \ \wedge \ E \cup B \models \varphi\rho\}.$$

Let $Y = vars(A) \cap vars(B)$. Then we call Y the *parameters* of the Hoare triple $\{A\}\ \mathcal{R}\ \{B\}$. Such a triple is in fact *universally quantified* on its parameters. That is, $\{A\}\ \mathcal{R}\ \{B\}$ implicitly means: $(\forall Y)\ \{A\}\ \mathcal{R}\ \{B\}$.

Let us see an example of a parametric Hoare triple involving a slight modification of the `CHOICE` module in Lecture 16.

A Hoare Triple for the CHOICE Module

```
mod CHOICE is
protecting NAT .
sorts MSet State Pred .
subsorts Nat < MSet .
op __ : MSet MSet -> MSet [ctor assoc comm] .
op {_} : MSet -> State .
op tt : -> Pred [ctor] .
op _=C_ : MSet MSet -> Pred [ctor] .  *** MSet containment
vars U V : MSet . var N : Nat .
eq U =C U = tt .
eq U =C U V = tt .
rl [choice] : {U V} => {U} .
endm
```

A Hoare Triple for the CHOICE Module

```
mod CHOICE is
protecting NAT .
sorts MSet State Pred .
subsorts Nat < MSet .
op __ : MSet MSet -> MSet [ctor assoc comm] .
op {_} : MSet -> State .
op tt : -> Pred [ctor] .
op _=C_ : MSet MSet -> Pred [ctor] .  *** MSet containment
vars U V : MSet . var N : Nat .
eq U =C U = tt .
eq U =C U V = tt .
rl [choice] : {U V} => {U} .
endm
```

The Hoare triple: $\{\{M\} \mid \top\}$ CHOICE $\{\{N\} \mid N \subseteq M = tt\}$ is *parametric* on M.

A Hoare Triple for the CHOICE Module

```
mod CHOICE is
protecting NAT .
sorts MSet State Pred .
subsorts Nat < MSet .
op __ : MSet MSet -> MSet [ctor assoc comm] .
op {_} : MSet -> State .
op tt : -> Pred [ctor] .
op _=C_ : MSet MSet -> Pred [ctor] .  *** MSet containment
vars U V : MSet . var N : Nat .
eq U =C U = tt .
eq U =C U V = tt .
rl [choice] : {U V} => {U} .
endm
```

The Hoare triple: $\{\{M\} \mid \top\}$ CHOICE $\{\{N\} \mid N \subseteq M = tt\}$ is *parametric* on M. It states that for each M every final state reachable from $\{M\}$ is a singleton set $\{N\}$ with N in M.

From Hoare Logic to Reachability Logic

Hoare logic is widely used to state and verify properties of an *imperative program* p in, say, Java or C.

From Hoare Logic to Reachability Logic

Hoare logic is widely used to state and verify properties of an *imperative program* p in, say, Java or C. it is then written in the form $\{A\}\ p\ \{B\}$.

From Hoare Logic to Reachability Logic

Hoare logic is widely used to state and verify properties of an *imperative program* p in, say, Java or C. it is then written in the form $\{A\}\ p\ \{B\}$. We shall see later in the course that this is just a special case of a Hoare triple of the form $\{A'(p)\}\ \mathcal{R}_{\mathcal{L}}\ \{B'\}$, where $\mathcal{L}$ is the imperative programming language, and $\mathcal{R}_{\mathcal{L}}$ is the *rewriting logic semantics* of $\mathcal{L}$.

From Hoare Logic to Reachability Logic

Hoare logic is widely used to state and verify properties of an *imperative program* p in, say, Java or C. it is then written in the form $\{A\}\ p\ \{B\}$. We shall see later in the course that this is just a special case of a Hoare triple of the form $\{A'(p)\}\ \mathcal{R}_{\mathcal{L}}\ \{B'\}$, where $\mathcal{L}$ is the imperative programming language, and $\mathcal{R}_{\mathcal{L}}$ is the *rewriting logic semantics* of $\mathcal{L}$.

A serious difficulty with traditional Hoare logic is that it is *language-dependent*. There is a Hoare logic for Java, another for C, and so on.

From Hoare Logic to Reachability Logic

Hoare logic is widely used to state and verify properties of an *imperative program* p in, say, Java or C. it is then written in the form $\{A\}\ p\ \{B\}$. We shall see later in the course that this is just a special case of a Hoare triple of the form $\{A'(p)\}\ \mathcal{R}_{\mathcal{L}}\ \{B'\}$, where $\mathcal{L}$ is the imperative programming language, and $\mathcal{R}_{\mathcal{L}}$ is the *rewriting logic semantics* of $\mathcal{L}$.

A serious difficulty with traditional Hoare logic is that it is *language-dependent*. There is a Hoare logic for Java, another for C, and so on. Furthermore for each language *different inference rules* must be defined and proved correct.

From Hoare Logic to Reachability Logic

Hoare logic is widely used to state and verify properties of an *imperative program* p in, say, Java or C. it is then written in the form $\{A\}\ p\ \{B\}$. We shall see later in the course that this is just a special case of a Hoare triple of the form $\{A'(p)\}\ \mathcal{R}_{\mathcal{L}}\ \{B'\}$, where $\mathcal{L}$ is the imperative programming language, and $\mathcal{R}_{\mathcal{L}}$ is the *rewriting logic semantics* of $\mathcal{L}$.

A serious difficulty with traditional Hoare logic is that it is *language-dependent*. There is a Hoare logic for Java, another for C, and so on. Furthermore for each language *different inference rules* must be defined and proved correct.

Roșu, Stefanescu et al. at UIUC have made Hoare logic *programming-language-independent* by generalizing it to *reachability logic*.

From Hoare Logic to Reachability Logic

Hoare logic is widely used to state and verify properties of an *imperative program* p in, say, Java or C. it is then written in the form $\{A\}\ p\ \{B\}$. We shall see later in the course that this is just a special case of a Hoare triple of the form $\{A'(p)\}\ \mathcal{R}_{\mathcal{L}}\ \{B'\}$, where $\mathcal{L}$ is the imperative programming language, and $\mathcal{R}_{\mathcal{L}}$ is the *rewriting logic semantics* of $\mathcal{L}$.

A serious difficulty with traditional Hoare logic is that it is *language-dependent*. There is a Hoare logic for Java, another for C, and so on. Furthermore for each language *different inference rules* must be defined and proved correct.

Roșu, Stefanescu et al. at UIUC have made Hoare logic *programming-language-independent* by generalizing it to *reachability logic*.

Skeirik, Stefanescu and Meseguer at UIUC have in turn made reachability logic *rewrite-theory-independent* by defining it for rewrite theories $\mathcal{R}$.

Reachability Logic (RL) is:

Reachability Logic (RL) is:

- parameterized over an underlying *rewrite theory* $\mathcal{R}$

Reachability Logic (RL) is:

- parameterized over an underlying *rewrite theory* $\mathcal{R}$
- considers formulas $A \longrightarrow^{\circledast} B$ where A is a *pattern predicate*, and B a *disjunction of pattern predicates*.

$$u \mid \varphi \longrightarrow^{\circledast} \bigvee_i v_i \mid \psi_i$$

Reachability Logic (RL) is:

- parameterized over an underlying *rewrite theory* $\mathcal{R}$
- considers formulas $A \longrightarrow^{\circledast} B$ where A is a *pattern predicate*, and B a *disjunction of pattern predicates*.

$$u \mid \varphi \longrightarrow^{\circledast} \bigvee_i v_i \mid \psi_i$$

- a generalization of Hoare Logic *partial correctness*, i.e., $A \longrightarrow^{\circledast} B$ generalizes $\{A\}\mathcal{R}\{B\}$

Reachability Logic (RL) is:

- parameterized over an underlying *rewrite theory* $\mathcal{R}$
- considers formulas $A \longrightarrow^{\circledast} B$ where A is a *pattern predicate*, and B a *disjunction of pattern predicates*.

$$u \mid \varphi \longrightarrow^{\circledast} \bigvee_i v_i \mid \psi_i$$

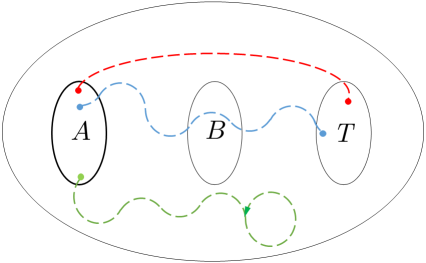
- a generalization of Hoare Logic *partial correctness*, i.e., $A \longrightarrow^{\circledast} B$ generalizes $\{A\}\mathcal{R}\{B\}$
- directly captures *inductive reasoning* in *any* theory $\mathcal{R}$, unlike Hoare Logic, special rules for loops, etc, *unnecessary*

Q: What does the relation $A \longrightarrow^{\circledast} B$ mean?

A: Suppose we have:

(1) a rewrite theory $\mathcal{R}$
(2) pattern fomulas A, B
(3) and terminating states T

Then $A \longrightarrow^{\circledast} B$ means:
for each state $[t] \in [\![A]\!]$
and rewrite path p from $[t]$,
either: (1) p crosses $[\![B]\!]$ or
(2) p is infinite



- - - indicates counterex.
- - - satisfies $A \to^{\circledast} B$
- - - *vacuously* satisfies

Reachability Logic

Precise Definition

Let $\mathcal{R} = (\Sigma, E \cup B, R)$ be a rewrite theory with good executability conditions, and having a subsignature Ω of constructors and a chosen top sort $State$ of states.

Reachability Logic

Precise Definition

Let $\mathcal{R} = (\Sigma, E \cup B, R)$ be a rewrite theory with good executability conditions, and having a subsignature Ω of constructors and a chosen top sort $State$ of states. Let $\mathcal{C}_{\mathcal{R}}$ denote the canonical reachability model.

Reachability Logic

Precise Definition

Let $\mathcal{R} = (\Sigma, E \cup B, R)$ be a rewrite theory with good executability conditions, and having a subsignature Ω of constructors and a chosen top sort $State$ of states. Let $\mathcal{C}_{\mathcal{R}}$ denote the canonical reachability model. For a reachability formula $A \longrightarrow^{\circledast} B$ call $Y = vars(A) \cap vars(B)$ its *parameters*.

Reachability Logic

Precise Definition

Let $\mathcal{R} = (\Sigma, E \cup B, R)$ be a rewrite theory with good executability conditions, and having a subsignature Ω of constructors and a chosen top sort $State$ of states. Let $\mathcal{C}_{\mathcal{R}}$ denote the canonical reachability model. For a reachability formula $A \longrightarrow^{\circledast} B$ call $Y = vars(A) \cap vars(B)$ its *parameters*.

If $Y = \emptyset$, then we write $\mathcal{R} \models A \longrightarrow^{\circledast} B$ iff

Reachability Logic

Precise Definition

Let $\mathcal{R} = (\Sigma, E \cup B, R)$ be a rewrite theory with good executability conditions, and having a subsignature Ω of constructors and a chosen top sort $State$ of states. Let $\mathcal{C}_{\mathcal{R}}$ denote the canonical reachability model. For a reachability formula $A \longrightarrow^{\circledast} B$ call $Y = vars(A) \cap vars(B)$ its *parameters*.

If $Y = \emptyset$, then we write $\mathcal{R} \models A \longrightarrow^{\circledast} B$ iff for each $[u_0] \in \mathcal{C}_{\mathcal{R},State}$ such that $[u_0] \in [\![A]\!]$

Reachability Logic

Precise Definition

Let $\mathcal{R} = (\Sigma, E \cup B, R)$ be a rewrite theory with good executability conditions, and having a subsignature Ω of constructors and a chosen top sort $State$ of states. Let $\mathcal{C}_{\mathcal{R}}$ denote the canonical reachability model. For a reachability formula $A \longrightarrow^{\circledast} B$ call $Y = vars(A) \cap vars(B)$ its *parameters*.

If $Y = \emptyset$, then we write $\mathcal{R} \models A \longrightarrow^{\circledast} B$ iff for each $[u_0] \in \mathcal{C}_{\mathcal{R},State}$ such that $[u_0] \in [\![A]\!]$ and each *terminating* sequence:

Reachability Logic

Precise Definition

Let $\mathcal{R} = (\Sigma, E \cup B, R)$ be a rewrite theory with good executability conditions, and having a subsignature Ω of constructors and a chosen top sort $State$ of states. Let $\mathcal{C}_{\mathcal{R}}$ denote the canonical reachability model. For a reachability formula $A \longrightarrow^{\circledast} B$ call $Y = vars(A) \cap vars(B)$ its *parameters*.

If $Y = \emptyset$, then we write $\mathcal{R} \models A \longrightarrow^{\circledast} B$ iff for each $[u_0] \in \mathcal{C}_{\mathcal{R},State}$ such that $[u_0] \in [\![A]\!]$ and each *terminating* sequence:

$$[u_0] \rightarrow_{\mathcal{C}_{\mathcal{R}}} [u_1] \ldots [u_{n-1}] \rightarrow_{\mathcal{C}_{\mathcal{R}}} [u_n]$$

Reachability Logic

Precise Definition

Let $\mathcal{R} = (\Sigma, E \cup B, R)$ be a rewrite theory with good executability conditions, and having a subsignature Ω of constructors and a chosen top sort $State$ of states. Let $\mathcal{C}_{\mathcal{R}}$ denote the canonical reachability model. For a reachability formula $A \longrightarrow^{\circledast} B$ call $Y = vars(A) \cap vars(B)$ its *parameters*.

If $Y = \emptyset$, then we write $\mathcal{R} \models A \longrightarrow^{\circledast} B$ iff for each $[u_0] \in \mathcal{C}_{\mathcal{R},State}$ such that $[u_0] \in [\![A]\!]$ and each *terminating* sequence:

$$[u_0] \rightarrow_{\mathcal{C}_{\mathcal{R}}} [u_1] \ldots [u_{n-1}] \rightarrow_{\mathcal{C}_{\mathcal{R}}} [u_n]$$

there exist j, $0 \leq j \leq n$ such that $[u_j] \in [\![B]\!]$.

Reachability Logic

Precise Definition

Let $\mathcal{R} = (\Sigma, E \cup B, R)$ be a rewrite theory with good executability conditions, and having a subsignature Ω of constructors and a chosen top sort $State$ of states. Let $\mathcal{C}_\mathcal{R}$ denote the canonical reachability model. For a reachability formula $A \longrightarrow^{\circledast} B$ call $Y = vars(A) \cap vars(B)$ its *parameters*.

If $Y = \emptyset$, then we write $\mathcal{R} \models A \longrightarrow^{\circledast} B$ iff for each $[u_0] \in \mathcal{C}_{\mathcal{R},State}$ such that $[u_0] \in [\![A]\!]$ and each *terminating* sequence:

$$[u_0] \to_{\mathcal{C}_\mathcal{R}} [u_1] \ldots [u_{n-1}] \to_{\mathcal{C}_\mathcal{R}} [u_n]$$

there exist j, $0 \leq j \leq n$ such that $[u_j] \in [\![B]\!]$.

If $Y \neq \emptyset$, then we write $\mathcal{R} \models A \longrightarrow^{\circledast} B$ iff

Reachability Logic

Precise Definition

Let $\mathcal{R} = (\Sigma, E \cup B, R)$ be a rewrite theory with good executability conditions, and having a subsignature Ω of constructors and a chosen top sort $State$ of states. Let $\mathcal{C}_{\mathcal{R}}$ denote the canonical reachability model. For a reachability formula $A \longrightarrow^{\circledast} B$ call $Y = vars(A) \cap vars(B)$ its *parameters*.

If $Y = \emptyset$, then we write $\mathcal{R} \models A \longrightarrow^{\circledast} B$ iff for each $[u_0] \in \mathcal{C}_{\mathcal{R},State}$ such that $[u_0] \in [\![A]\!]$ and each *terminating* sequence:

$$[u_0] \rightarrow_{\mathcal{C}_{\mathcal{R}}} [u_1] \ldots [u_{n-1}] \rightarrow_{\mathcal{C}_{\mathcal{R}}} [u_n]$$

there exist j, $0 \leq j \leq n$ such that $[u_j] \in [\![B]\!]$.

If $Y \neq \emptyset$, then we write $\mathcal{R} \models A \longrightarrow^{\circledast} B$ iff for each $\rho \in [Y \to T_{\Omega}]$ we have $\mathcal{R} \models A\rho \longrightarrow^{\circledast} B\rho$.

Let $\mathcal{R} = (\Sigma, E \cup B, R)$ be a rewrite theory with good executability conditions, and having a subsignature Ω of constructors and a chosen top sort $State$ of states. Let $\mathcal{C}_{\mathcal{R}}$ denote the canonical reachability model. For a reachability formula $A \longrightarrow^{\circledast} B$ call $Y = vars(A) \cap vars(B)$ its *parameters*.

If $Y = \emptyset$, then we write $\mathcal{R} \models A \longrightarrow^{\circledast} B$ iff for each $[u_0] \in \mathcal{C}_{\mathcal{R},State}$ such that $[u_0] \in [\![A]\!]$ and each *terminating* sequence:

$$[u_0] \rightarrow_{\mathcal{C}_{\mathcal{R}}} [u_1] \ldots [u_{n-1}] \rightarrow_{\mathcal{C}_{\mathcal{R}}} [u_n]$$

there exist j, $0 \leq j \leq n$ such that $[u_j] \in [\![B]\!]$.

If $Y \neq \emptyset$, then we write $\mathcal{R} \models A \longrightarrow^{\circledast} B$ iff for each $\rho \in [Y \to T_\Omega]$ we have $\mathcal{R} \models A\rho \longrightarrow^{\circledast} B\rho$.

That is, the parameters Y in $A \longrightarrow^{\circledast} B$ are *universally quantified*,

Reachability Logic

Precise Definition

Let $\mathcal{R} = (\Sigma, E \cup B, R)$ be a rewrite theory with good executability conditions, and having a subsignature Ω of constructors and a chosen top sort $State$ of states. Let $\mathcal{C}_{\mathcal{R}}$ denote the canonical reachability model. For a reachability formula $A \longrightarrow^{\circledast} B$ call $Y = vars(A) \cap vars(B)$ its *parameters*.

If $Y = \emptyset$, then we write $\mathcal{R} \models A \longrightarrow^{\circledast} B$ iff for each $[u_0] \in \mathcal{C}_{\mathcal{R},State}$ such that $[u_0] \in [\![A]\!]$ and each *terminating* sequence:

$$[u_0] \to_{\mathcal{C}_{\mathcal{R}}} [u_1] \ldots [u_{n-1}] \to_{\mathcal{C}_{\mathcal{R}}} [u_n]$$

there exist j, $0 \leq j \leq n$ such that $[u_j] \in [\![B]\!]$.

If $Y \neq \emptyset$, then we write $\mathcal{R} \models A \longrightarrow^{\circledast} B$ iff for each $\rho \in [Y \to T_\Omega]$ we have $\mathcal{R} \models A\rho \longrightarrow^{\circledast} B\rho$.

That is, the parameters Y in $A \longrightarrow^{\circledast} B$ are *universally quantified*, so that $A \longrightarrow^{\circledast} B$ implicitly means: $(\forall Y)\ A \longrightarrow^{\circledast} B$.

Relationships with Hoare Logic and Temporal Logic

Q: How is a Hoare triple $\{A\}\mathcal{R}\{B\}$ expressed in reachability logic?

Relationships with Hoare Logic and Temporal Logic

Q: How is a Hoare triple $\{A\}\mathcal{R}\{B\}$ expressed in reachability logic?

A: as the formula $A \longrightarrow^{\circledast} (B \wedge T)$, with $[\![T]\!]$ the terminating states.

Relationships with Hoare Logic and Temporal Logic

Q: How is a Hoare triple $\{A\}\mathcal{R}\{B\}$ expressed in reachability logic?

A: as the formula $A \longrightarrow^{\circledast} (B \wedge T)$, with $[\![T]\!]$ the terminating states.

Q: How is a reachability logic sequent $A \longrightarrow^{\circledast} B$ expressed in linear temporal logic?

Relationships with Hoare Logic and Temporal Logic

Q: How is a Hoare triple $\{A\}\mathcal{R}\{B\}$ expressed in reachability logic?

A: as the formula $A \longrightarrow^{\circledast} (B \wedge T)$, with $[\![T]\!]$ the terminating states.

Q: How is a reachability logic sequent $A \longrightarrow^{\circledast} B$ expressed in linear temporal logic?

A: as the LTL formula $A \rightarrow (\Box enabled) \vee \Diamond B$.

Relationships with Hoare Logic and Temporal Logic

Q: How is a Hoare triple $\{A\}\mathcal{R}\{B\}$ expressed in reachability logic?

A: as the formula $A \longrightarrow^{\circledast} (B \wedge T)$, with $[\![T]\!]$ the terminating states.

Q: How is a reachability logic sequent $A \longrightarrow^{\circledast} B$ expressed in linear temporal logic?

A: as the LTL formula $A \rightarrow (\Box enabled) \vee \Diamond B$.

Example. For `CHOICE`, the formula

$$\{M\} \mid \top \longrightarrow^{\circledast} \{M'\} \mid M' \subseteq M = tt$$

Relationships with Hoare Logic and Temporal Logic

Q: How is a Hoare triple $\{A\}\mathcal{R}\{B\}$ expressed in reachability logic?

A: as the formula $A \longrightarrow^{\circledast} (B \wedge T)$, with $[\![T]\!]$ the terminating states.

Q: How is a reachability logic sequent $A \longrightarrow^{\circledast} B$ expressed in linear temporal logic?

A: as the LTL formula $A \to (\Box enabled) \vee \Diamond B$.

Example. For `CHOICE`, the formula

$$\{M\} \mid \top \longrightarrow^{\circledast} \{M'\} \mid M' \subseteq M = tt$$

is *parametric* on M.

Relationships with Hoare Logic and Temporal Logic

Q: How is a Hoare triple $\{A\}\mathcal{R}\{B\}$ expressed in reachability logic?

A: as the formula $A \longrightarrow^{\circledast} (B \wedge T)$, with $[\![T]\!]$ the terminating states.

Q: How is a reachability logic sequent $A \longrightarrow^{\circledast} B$ expressed in linear temporal logic?

A: as the LTL formula $A \rightarrow (\Box \mathit{enabled}) \vee \Diamond B$.

Example. For `CHOICE`, the formula

$$\{M\} \mid \top \longrightarrow^{\circledast} \{M'\} \mid M' \subseteq M = tt$$

is *parametric* on M. It states that for each M every state reachable from $\{M\}$ is a submultiset M' of M.

Relationships with Hoare Logic and Temporal Logic

Q: How is a Hoare triple $\{A\}\mathcal{R}\{B\}$ expressed in reachability logic?

A: as the formula $A \longrightarrow^{\circledast} (B \wedge T)$, with $[\![T]\!]$ the terminating states.

Q: How is a reachability logic sequent $A \longrightarrow^{\circledast} B$ expressed in linear temporal logic?

A: as the LTL formula $A \rightarrow (\Box \mathit{enabled}) \vee \Diamond B$.

Example. For `CHOICE`, the formula

$$\{M\} \mid \top \longrightarrow^{\circledast} \{M'\} \mid M' \subseteq M = \mathit{tt}$$

is *parametric* on M. It states that for each M every state reachable from $\{M\}$ is a submultiset M' of M. Note that this reachability property *cannot* be expressed by a Hoare triple.

The Invariant Paradox

Consider the readers and writers example (Lecture 18):

The Invariant Paradox

Consider the readers and writers example (Lecture 18):

```
mod READERS-WRITERS is
protecting NAT .
sort State .
op <_,_> : Nat Nat -> State [ctor] . --- readers/writers
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
```

The Invariant Paradox

Consider the readers and writers example (Lecture 18):

```
mod READERS-WRITERS is
protecting NAT .
sort State .
op <_,_> : Nat Nat -> State [ctor] . --- readers/writers
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
```

Q: How can we express its *mutual exclusion* invariant as a reachability formula $A \longrightarrow^{\circledast} B$?

The Invariant Paradox

Consider the readers and writers example (Lecture 18):

```
mod READERS-WRITERS is
protecting NAT .
sort State .
op <_,_> : Nat Nat -> State [ctor] . --- readers/writers
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
```

Q: How can we express its *mutual exclusion* invariant as a reachability formula $A \longrightarrow^{\circledast} B$?

A: Since:

The Invariant Paradox

Consider the readers and writers example (Lecture 18):

```
mod READERS-WRITERS is
protecting NAT .
sort State .
op <_,_> : Nat Nat -> State [ctor] . --- readers/writers
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
```

Q: How can we express its *mutual exclusion* invariant as a reachability formula $A \longrightarrow^{\circledast} B$?

A: Since: (i) $A \longrightarrow^{\circledast} B$ just means $A \rightarrow (\Box \mathit{enabled}) \vee \Diamond B$, and

The Invariant Paradox

Consider the readers and writers example (Lecture 18):

```
mod READERS-WRITERS is
protecting NAT .
sort State .
op <_,_> : Nat Nat -> State [ctor] . --- readers/writers
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
```

Q: How can we express its *mutual exclusion* invariant as a reachability formula $A \longrightarrow^{\circledast} B$?

A: Since: (i) $A \longrightarrow^{\circledast} B$ just means $A \rightarrow (\Box enabled) \vee \Diamond B$, and (ii) `READERS-WRITERS` is a *never terminating* rewrite theory,

The Invariant Paradox

Consider the readers and writers example (Lecture 18):

```
mod READERS-WRITERS is
protecting NAT .
sort State .
op <_,_> : Nat Nat -> State [ctor] . --- readers/writers
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
```

Q: How can we express its *mutual exclusion* invariant as a reachability formula $A \longrightarrow^{\circledast} B$?

A: Since: (i) $A \longrightarrow^{\circledast} B$ just means $A \to (\Box enabled) \vee \Diamond B$, and (ii) `READERS-WRITERS` is a *never terminating* rewrite theory, *all* formulas $A \longrightarrow^{\circledast} B$ are satisfied!!

The Invariant Paradox

Consider the readers and writers example (Lecture 18):

```
mod READERS-WRITERS is
protecting NAT .
sort State .
op <_,_> : Nat Nat -> State [ctor] . --- readers/writers
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
```

Q: How can we express its *mutual exclusion* invariant as a reachability formula $A \longrightarrow^{\circledast} B$?

A: Since: (i) $A \longrightarrow^{\circledast} B$ just means $A \to (\Box enabled) \vee \Diamond B$, and (ii) `READERS-WRITERS` is a *never terminating* rewrite theory, *all* formulas $A \longrightarrow^{\circledast} B$ are satisfied!! So we *cannot*!!

The Invariant Paradox

Consider the readers and writers example (Lecture 18):

```
mod READERS-WRITERS is
protecting NAT .
sort State .
op <_,_> : Nat Nat -> State [ctor] . --- readers/writers
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
```

Q: How can we express its *mutual exclusion* invariant as a reachability formula $A \longrightarrow^{\circledast} B$?

A: Since: (i) $A \longrightarrow^{\circledast} B$ just means $A \to (\Box enabled) \vee \Diamond B$, and (ii) READERS-WRITERS is a *never terminating* rewrite theory, *all* formulas $A \longrightarrow^{\circledast} B$ are satisfied!! So we *cannot*!! (Paradox!!).

Solving the Invariant Paradox

Let us add a *stopwatch* to `READERS-WRITERS` as follows:

Solving the Invariant Paradox

Let us add a *stopwatch* to READERS-WRITERS as follows:

```
mod READERS-WRITERS-stop is
protecting NAT .
sort State .
op <_,_> : Nat Nat -> State [ctor] .
op [_,_] : Nat Nat -> State [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 > .
rl < R, W > => [R,W] .
endm
```

Solving the Invariant Paradox

Let us add a *stopwatch* to `READERS-WRITERS` as follows:

```
mod READERS-WRITERS-stop is
protecting NAT .
sort State .
op <_,_> : Nat Nat -> State [ctor] .
op [_,_] : Nat Nat -> State [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 > .
rl < R, W > => [R,W] .
endm
```

The rule `< R, W > => [R,W]` can now *stop* any state and make it terminating.

Solving the Invariant Paradox

Let us add a *stopwatch* to `READERS-WRITERS` as follows:

```
mod READERS-WRITERS-stop is
protecting NAT .
sort State .
op <_,_> : Nat Nat -> State [ctor] .
op [_,_] : Nat Nat -> State [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 > .
rl < R, W > => [R,W] .
endm
```

The rule `< R, W > => [R,W]` can now *stop* any state and make it terminating. For any pattern predicate $B = \langle u, v \rangle \mid \varphi$ let $[B]$ denote the pattern predicate $[B] = [u, v] \mid \varphi$.

Solving the Invariant Paradox

Let us add a *stopwatch* to READERS-WRITERS as follows:

```
mod READERS-WRITERS-stop is
protecting NAT .
sort State .
op <_,_> : Nat Nat -> State [ctor] .
op [_,_] : Nat Nat -> State [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 > .
rl < R, W > => [R,W] .
endm
```

The rule `< R, W > => [R,W]` can now *stop* any state and make it terminating. For any pattern predicate $B = \langle u, v \rangle \mid \varphi$ let $[B]$ denote the pattern predicate $[B] = [u, v] \mid \varphi$.

Fact. B is an *invariant* from initial states S_0 in READERS-WRITERS iff

Solving the Invariant Paradox

Let us add a *stopwatch* to READERS-WRITERS as follows:

```
mod READERS-WRITERS-stop is
protecting NAT .
sort State .
op <_,_> : Nat Nat -> State [ctor] .
op [_,_] : Nat Nat -> State [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 > .
rl < R, W > => [R,W] .
endm
```

The rule `< R, W > => [R,W]` can now *stop* any state and make it terminating. For any pattern predicate $B = \langle u, v \rangle \mid \varphi$ let $[B]$ denote the pattern predicate $[B] = [u, v] \mid \varphi$.

Fact. B is an *invariant* from initial states S_0 in READERS-WRITERS iff $S_0 \longrightarrow^{\circledast} [B]$ holds in READERS-WRITERS-stop.

Solving the Invariant Paradox (General Case)

Suppose $\mathcal{R}$ is *never terminating* (has no terminating states),

Solving the Invariant Paradox (General Case)

Suppose $\mathcal{R}$ is *never terminating* (has no terminating states), $State$ has a single constructor $\langle \_, \ldots, \_ \rangle : s_1 \ldots s_n \to State$,

Solving the Invariant Paradox (General Case)

Suppose $\mathcal{R}$ is *never terminating* (has no terminating states), $State$ has a single constructor $\langle \_, \ldots, \_ \rangle : s_1 \ldots s_n \to State$, and all rules are between terms of sort $State$.

Solving the Invariant Paradox (General Case)

Suppose $\mathcal{R}$ is *never terminating* (has no terminating states), $State$ has a single constructor $\langle \_, \ldots, \_ \rangle : s_1 \ldots s_n \to State$, and all rules are between terms of sort $State$. Call $\mathcal{R}_{stop}$ the rewrite theory extending $\mathcal{R}$ by adding:

Solving the Invariant Paradox (General Case)

Suppose $\mathcal{R}$ is *never terminating* (has no terminating states), $State$ has a single constructor $\langle \_, \ldots, \_ \rangle : s_1 \ldots s_n \to State$, and all rules are between terms of sort $State$. Call $\mathcal{R}_{stop}$ the rewrite theory extending $\mathcal{R}$ by adding: (i) $[\_, \ldots, \_] : s_1 \ldots s_n \to State$, and

Solving the Invariant Paradox (General Case)

Suppose $\mathcal{R}$ is *never terminating* (has no terminating states), $State$ has a single constructor $\langle \_, \ldots, \_ \rangle : s_1 \ldots s_n \to State$, and all rules are between terms of sort $State$. Call $\mathcal{R}_{stop}$ the rewrite theory extending $\mathcal{R}$ by adding: (i) $[\_, \ldots, \_] : s_1 \ldots s_n \to State$, and (ii) a *stop rule* $\langle x_1, \ldots, x_n \rangle \to [x_1, \ldots, x_n]$.

Solving the Invariant Paradox (General Case)

Suppose $\mathcal{R}$ is *never terminating* (has no terminating states), $State$ has a single constructor $\langle \_, \ldots, \_ \rangle : s_1 \ldots s_n \to State$, and all rules are between terms of sort $State$. Call $\mathcal{R}_{stop}$ the rewrite theory extending $\mathcal{R}$ by adding: (i) $[\_, \ldots, \_] : s_1 \ldots s_n \to State$, and (ii) a *stop rule* $\langle x_1, \ldots, x_n \rangle \to [x_1, \ldots, x_n]$. Then:

Solving the Invariant Paradox (General Case)

Suppose $\mathcal{R}$ is *never terminating* (has no terminating states), $State$ has a single constructor $\langle \_, \ldots, \_ \rangle : s_1 \ldots s_n \to State$, and all rules are between terms of sort $State$. Call $\mathcal{R}_{stop}$ the rewrite theory extending $\mathcal{R}$ by adding: (i) $[\_, \ldots, \_] : s_1 \ldots s_n \to State$, and (ii) a *stop rule* $\langle x_1, \ldots, x_n \rangle \to [x_1, \ldots, x_n]$. Then:

Theorem

B is an invariant for $\mathcal{R}$ from initial sates S_0 iff $S_0 \longrightarrow^{\circledast} [B]$ holds in $\mathcal{R}_{stop}$.

Solving the Invariant Paradox (General Case)

Suppose $\mathcal{R}$ is *never terminating* (has no terminating states), $State$ has a single constructor $\langle \_, \ldots, \_ \rangle : s_1 \ldots s_n \to State$, and all rules are between terms of sort $State$. Call $\mathcal{R}_{stop}$ the rewrite theory extending $\mathcal{R}$ by adding: (i) $[\_, \ldots, \_] : s_1 \ldots s_n \to State$, and (ii) a *stop rule* $\langle x_1, \ldots, x_n \rangle \to [x_1, \ldots, x_n]$. Then:

Theorem

B is an invariant for $\mathcal{R}$ from initial sates S_0 iff $S_0 \longrightarrow^{\circledast} [B]$ holds in $\mathcal{R}_{stop}$.

Corollary

If $[\![S_0]\!] \subseteq [\![B]\!]$ and $B \longrightarrow^{\circledast} [B]$ holds in $\mathcal{R}_{stop}$, then B is an invariant for $\mathcal{R}$ from initial sates S_0.

Solving the Invariant Paradox (General Case)

Suppose $\mathcal{R}$ is *never terminating* (has no terminating states), $State$ has a single constructor $\langle \_, \ldots, \_ \rangle : s_1 \ldots s_n \to State$, and all rules are between terms of sort $State$. Call $\mathcal{R}_{stop}$ the rewrite theory extending $\mathcal{R}$ by adding: (i) $[\_, \ldots, \_] : s_1 \ldots s_n \to State$, and (ii) a *stop rule* $\langle x_1, \ldots, x_n \rangle \to [x_1, \ldots, x_n]$. Then:

Theorem

B is an invariant for $\mathcal{R}$ from initial sates S_0 iff $S_0 \longrightarrow^{\circledast} [B]$ holds in $\mathcal{R}_{stop}$.

Corollary

If $[\![S_0]\!] \subseteq [\![B]\!]$ and $B \longrightarrow^{\circledast} [B]$ holds in $\mathcal{R}_{stop}$, then B is an invariant for $\mathcal{R}$ from initial sates S_0.

Example. Mutual exclusion from $\langle 0, 0 \rangle$ in `READERS-WRITERS` is the predicate:

Solving the Invariant Paradox (General Case)

Suppose $\mathcal{R}$ is *never terminating* (has no terminating states), $State$ has a single constructor $\langle \_, \ldots, \_ \rangle : s_1 \ldots s_n \to State$, and all rules are between terms of sort $State$. Call $\mathcal{R}_{stop}$ the rewrite theory extending $\mathcal{R}$ by adding: (i) $[\_, \ldots, \_] : s_1 \ldots s_n \to State$, and (ii) a *stop rule* $\langle x_1, \ldots, x_n \rangle \to [x_1, \ldots, x_n]$. Then:

Theorem

B is an invariant for $\mathcal{R}$ from initial sates S_0 iff $S_0 \longrightarrow^{\circledast} [B]$ holds in $\mathcal{R}_{stop}$.

Corollary

If $[\![S_0]\!] \subseteq [\![B]\!]$ and $B \longrightarrow^{\circledast} [B]$ holds in $\mathcal{R}_{stop}$, then B is an invariant for $\mathcal{R}$ from initial sates S_0.

Example. Mutual exclusion from $\langle 0, 0 \rangle$ in `READERS-WRITERS` is the predicate: $Mutex = \langle R, W \rangle \mid W = 0 \vee (W = 1 \wedge R = 0)$.

Solving the Invariant Paradox (General Case)

Suppose $\mathcal{R}$ is *never terminating* (has no terminating states), $State$ has a single constructor $\langle\_,\ldots,\_\rangle : s_1 \ldots s_n \to State$, and all rules are between terms of sort $State$. Call $\mathcal{R}_{stop}$ the rewrite theory extending $\mathcal{R}$ by adding: (i) $[\_,\ldots,\_] : s_1 \ldots s_n \to State$, and (ii) a *stop rule* $\langle x_1,\ldots,x_n\rangle \to [x_1,\ldots,x_n]$. Then:

Theorem

B is an invariant for $\mathcal{R}$ from initial sates S_0 iff $S_0 \longrightarrow^{\circledast} [B]$ holds in $\mathcal{R}_{stop}$.

Corollary

If $[\![S_0]\!] \subseteq [\![B]\!]$ and $B \longrightarrow^{\circledast} [B]$ holds in $\mathcal{R}_{stop}$, then B is an invariant for $\mathcal{R}$ from initial sates S_0.

Example. Mutual exclusion from $\langle 0,0\rangle$ in `READERS-WRITERS` is the predicate: $Mutex = \langle R,W\rangle \mid W = 0 \vee (W = 1 \wedge R = 0)$. We can prove it by showing:

Solving the Invariant Paradox (General Case)

Suppose $\mathcal{R}$ is *never terminating* (has no terminating states), $State$ has a single constructor $\langle \_, \ldots, \_ \rangle : s_1 \ldots s_n \to State$, and all rules are between terms of sort $State$. Call $\mathcal{R}_{stop}$ the rewrite theory extending $\mathcal{R}$ by adding: (i) $[\_, \ldots, \_] : s_1 \ldots s_n \to State$, and (ii) a *stop rule* $\langle x_1, \ldots, x_n \rangle \to [x_1, \ldots, x_n]$. Then:

Theorem

B is an invariant for $\mathcal{R}$ from initial sates S_0 iff $S_0 \longrightarrow^{\circledast} [B]$ holds in $\mathcal{R}_{stop}$.

Corollary

If $[\![S_0]\!] \subseteq [\![B]\!]$ and $B \longrightarrow^{\circledast} [B]$ holds in $\mathcal{R}_{stop}$, then B is an invariant for $\mathcal{R}$ from initial sates S_0.

Example. Mutual exclusion from $\langle 0, 0 \rangle$ in `READERS-WRITERS` is the predicate: $Mutex = \langle R, W \rangle \mid W = 0 \vee (W = 1 \wedge R = 0)$. We can prove it by showing: (i) $\langle 0, 0 \rangle \in Mutex$ (easy), and

Solving the Invariant Paradox (General Case)

Suppose $\mathcal{R}$ is *never terminating* (has no terminating states), $State$ has a single constructor $\langle\_, \ldots, \_\rangle : s_1 \ldots s_n \to State$, and all rules are between terms of sort $State$. Call $\mathcal{R}_{stop}$ the rewrite theory extending $\mathcal{R}$ by adding: (i) $[\_, \ldots, \_] : s_1 \ldots s_n \to State$, and (ii) a *stop rule* $\langle x_1, \ldots, x_n\rangle \to [x_1, \ldots, x_n]$. Then:

Theorem

B is an invariant for $\mathcal{R}$ from initial sates S_0 iff $S_0 \longrightarrow^{\circledast} [B]$ holds in $\mathcal{R}_{stop}$.

Corollary

If $[\![S_0]\!] \subseteq [\![B]\!]$ and $B \longrightarrow^{\circledast} [B]$ holds in $\mathcal{R}_{stop}$, then B is an invariant for $\mathcal{R}$ from initial sates S_0.

Example. Mutual exclusion from $\langle 0, 0\rangle$ in `READERS-WRITERS` is the predicate: $Mutex = \langle R, W\rangle \mid W = 0 \vee (W = 1 \wedge R = 0)$. We can prove it by showing: (i) $\langle 0, 0\rangle \in Mutex$ (easy), and (ii) $Mutex \longrightarrow^{\circledast} [Mutex]$ in `READERS-WRITERS-stop`.

Q: Then given RWL theory $\mathcal{R}$, how do we prove $A \longrightarrow^{\circledast} B$?

Reachability Logic

Proof Rules

Q: Then given RWL theory $\mathcal{R}$, how do we prove $A \longrightarrow^{\circledast} B$?

A: Perhaps surprisingly, two proof rules are enough

Reachability Logic

Proof Rules

Q: Then given RWL theory $\mathcal{R}$, how do we prove $A \longrightarrow^{\circledast} B$?

A: Perhaps surprisingly, two proof rules are enough

- A rule that traces *rewrite steps* of *symbolic* states in $\mathcal{R}$

Reachability Logic

Proof Rules

Q: Then given RWL theory $\mathcal{R}$, how do we prove $A \longrightarrow^{\circledast} B$?

A: Perhaps surprisingly, two proof rules are enough

- A rule that traces *rewrite steps* of *symbolic* states in $\mathcal{R}$
- A rule that captures *circular behavior* of $\mathcal{R}$

Q: Then given RWL theory $\mathcal{R}$, how do we prove $A \longrightarrow^{\circledast} B$?

A: Perhaps surprisingly, two proof rules are enough

- A rule that traces *rewrite steps* of *symbolic* states in $\mathcal{R}$
- A rule that captures *circular behavior* of $\mathcal{R}$

We call these two rules *Step+Subsumption* and *Axiom* resp.

Q: Then given RWL theory $\mathcal{R}$, how do we prove $A \longrightarrow^{\circledast} B$?

A: Perhaps surprisingly, two proof rules are enough

- A rule that traces *rewrite steps* of *symbolic* states in $\mathcal{R}$
- A rule that captures *circular behavior* of $\mathcal{R}$

We call these two rules *Step+Subsumption* and *Axiom* resp.

The key ideas are:

Reachability Logic

Proof Rules

Q: Then given RWL theory $\mathcal{R}$, how do we prove $A \longrightarrow^{\circledast} B$?

A: Perhaps surprisingly, two proof rules are enough

- A rule that traces *rewrite steps* of *symbolic* states in $\mathcal{R}$
- A rule that captures *circular behavior* of $\mathcal{R}$

We call these two rules *Step+Subsumption* and *Axiom* resp.

The key ideas are: (i) to prove $A \longrightarrow^{\circledast} B$ we may need some *auxiliary lemmas*.

Q: Then given RWL theory $\mathcal{R}$, how do we prove $A \longrightarrow^{\circledast} B$?

A: Perhaps surprisingly, two proof rules are enough

- A rule that traces *rewrite steps* of *symbolic* states in $\mathcal{R}$
- A rule that captures *circular behavior* of $\mathcal{R}$

We call these two rules *Step+Subsumption* and *Axiom* resp.

The key ideas are: (i) to prove $A \longrightarrow^{\circledast} B$ we may need some *auxiliary lemmas*. Call $\mathcal{C}$ the formulas $A \longrightarrow^{\circledast} B$ plus these lemmas;

Reachability Logic

Proof Rules

Q: Then given RWL theory $\mathcal{R}$, how do we prove $A \longrightarrow^{\circledast} B$?

A: Perhaps surprisingly, two proof rules are enough

- A rule that traces *rewrite steps* of *symbolic* states in $\mathcal{R}$
- A rule that captures *circular behavior* of $\mathcal{R}$

We call these two rules *Step+Subsumption* and *Axiom* resp.

The key ideas are: (i) to prove $A \longrightarrow^{\circledast} B$ we may need some *auxiliary lemmas*. Call $\mathcal{C}$ the formulas $A \longrightarrow^{\circledast} B$ plus these lemmas; (ii) we start with labeled sequents of the form $[\emptyset,\ \mathcal{C}]\ \vdash_T\ u \mid \varphi \longrightarrow^{\circledast} \bigvee_i v_i \mid \psi_i$ for all formulas in $\mathcal{C}$;

Reachability Logic

Proof Rules

Q: Then given RWL theory $\mathcal{R}$, how do we prove $A \longrightarrow^{\circledast} B$?

A: Perhaps surprisingly, two proof rules are enough

- A rule that traces *rewrite steps* of *symbolic* states in $\mathcal{R}$
- A rule that captures *circular behavior* of $\mathcal{R}$

We call these two rules *Step+Subsumption* and *Axiom* resp.

The key ideas are: (i) to prove $A \longrightarrow^{\circledast} B$ we may need some *auxiliary lemmas*. Call $\mathcal{C}$ the formulas $A \longrightarrow^{\circledast} B$ plus these lemmas; (ii) we start with labeled sequents of the form $[\emptyset,\ \mathcal{C}]\ \vdash_T\ u \mid \varphi \longrightarrow^{\circledast} \bigvee_i v_i \mid \psi_i$ for all formulas in $\mathcal{C}$; (iii) the first component ($\emptyset$) are the formulas we can assume as *axioms* (none);

Reachability Logic

Proof Rules

Q: Then given RWL theory $\mathcal{R}$, how do we prove $A \longrightarrow^{\circledast} B$?

A: Perhaps surprisingly, two proof rules are enough

- A rule that traces *rewrite steps* of *symbolic* states in $\mathcal{R}$
- A rule that captures *circular behavior* of $\mathcal{R}$

We call these two rules *Step+Subsumption* and *Axiom* resp.

The key ideas are: (i) to prove $A \longrightarrow^{\circledast} B$ we may need some *auxiliary lemmas*. Call $\mathcal{C}$ the formulas $A \longrightarrow^{\circledast} B$ plus these lemmas; (ii) we start with labeled sequents of the form $[\emptyset,\ \mathcal{C}]\ \vdash_T\ u \mid \varphi \longrightarrow^{\circledast} \bigvee_i v_i \mid \psi_i$ for all formulas in $\mathcal{C}$; (iii) the first component ($\emptyset$) are the formulas we can assume as *axioms* (none); (iv) the second ($\mathcal{C}$) the formulas we need to prove and *cannot yet assume*;

Reachability Logic
Proof Rules

Q: Then given RWL theory $\mathcal{R}$, how do we prove $A \longrightarrow^{\circledast} B$?

A: Perhaps surprisingly, two proof rules are enough

- A rule that traces *rewrite steps* of *symbolic* states in $\mathcal{R}$
- A rule that captures *circular behavior* of $\mathcal{R}$

We call these two rules *Step+Subsumption* and *Axiom* resp.

The key ideas are: (i) to prove $A \longrightarrow^{\circledast} B$ we may need some *auxiliary lemmas*. Call $\mathcal{C}$ the formulas $A \longrightarrow^{\circledast} B$ plus these lemmas; (ii) we start with labeled sequents of the form $[\emptyset,\ \mathcal{C}]\ \vdash_T\ u \mid \varphi \longrightarrow^{\circledast} \bigvee_i v_i \mid \psi_i$ for all formulas in $\mathcal{C}$; (iii) the first component ($\emptyset$) are the formulas we can assume as *axioms* (none); (iv) the second ($\mathcal{C}$) the formulas we need to prove and *cannot yet assume*; (v) the *Step+Subsumption* rule allows us to *inductively assume* $\mathcal{C}$ after a rewrite step with rules $R = \{l_j \to r_j\ if\, \phi_j\}$.

$$\frac{\displaystyle\bigwedge_{(j,\alpha)\in\textsc{unify}(u|\varphi',\ R)} [\mathcal{A}\cup\mathcal{C},\ \emptyset]\ \vdash_T\ (r_j \mid \varphi' \wedge \phi_j)\alpha \longrightarrow^{\circledast} \bigvee_i (v_i \mid \psi_i)\alpha}{[\mathcal{A},\ \mathcal{C}]\ \vdash_T\ u \mid \varphi \longrightarrow^{\circledast} \bigvee_i v_i \mid \psi_i}$$

$$\frac{\displaystyle\bigwedge_j [\{u' \mid \varphi' \longrightarrow^{\circledast} \bigvee_j v'_j \mid \psi'_j\} \cup \mathcal{A},\ \emptyset]\ \vdash_T\ v'_j\alpha \mid \varphi \wedge \psi'_j\alpha \longrightarrow^{\circledast} \bigvee_i v_i \mid \psi_i}{[\{u' \mid \varphi' \longrightarrow^{\circledast} \bigvee_j v'_j \mid \psi'_j\} \cup \mathcal{A},\ \emptyset]\ \vdash_T\ u \mid \varphi \longrightarrow^{\circledast} \bigvee_i v_i \mid \psi_i}$$

The *Step+Subsumption* and *Axiom* Rules

Current Progress

Q: So what work has been done already?

Current Progress

Q: So what work has been done already?

A: A substantial RL framework is already in place with:

Current Progress

Q: So what work has been done already?

A: A substantial RL framework is already in place with:

- full semantics for RL developed in terms of RWL

Current Progress

Q: So what work has been done already?

A: A substantial RL framework is already in place with:

- full semantics for RL developed in terms of RWL
- soundness proof for proof system and semantics

Current Progress

Q: So what work has been done already?

A: A substantial RL framework is already in place with:

- full semantics for RL developed in terms of RWL
- soundness proof for proof system and semantics
- Maude tool semi-automating the proof system

Current Progress

Q: So what work has been done already?

A: A substantial RL framework is already in place with:

- full semantics for RL developed in terms of RWL
- soundness proof for proof system and semantics
- Maude tool semi-automating the proof system
- a collection of case studies.

Current Progress

Q: So what work has been done already?

A: A substantial RL framework is already in place with:

- full semantics for RL developed in terms of RWL
- soundness proof for proof system and semantics
- Maude tool semi-automating the proof system
- a collection of case studies.

Next lecture will illustrate the use of the Maude Reachability Logic tool by means of examples.