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 state 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.
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 state 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.
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.
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.
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:
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.
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 \textit{finite} sets of reachable states.

2. Even if an \textit{equational abstraction} can be used to make the set of reachable states finite, the \textit{set of abstracted initial states} of interest may be \textit{infinite}.

3. More generally, \textit{state infinity} can block the use of explicit-state model checking in two different ways:
   - The number of states reachable from a given state is \textit{infinite}.
   - The number of initial states is \textit{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 \textit{finite} sets of reachable states.

2. Even if an \textit{equational abstraction} can be used to make the set of reachable states finite, the \textit{set of abstracted initial states} of interest may be \textit{infinite}.

3. More generally, \textit{state infinity} can block the use of explicit-state model checking in two different ways:
   - The number of states reachable from a given state is \textit{infinite}.
   - The number of initial states is \textit{infinite}.

This suggests two other options: (1) \textit{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 \textit{finite} sets of reachable states.

2. Even if an \textit{equational abstraction} can be used to make the set of reachable states finite, the \textit{set of abstracted initial states} of interest may be \textit{infinite}.

3. More generally, \textit{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 \textit{infinite}.
   - The number of initial states is \textit{infinite}.

This suggests two other options: (1) \textit{symbolic model checking} (automatic) and (2) \textit{deductive methods} based on theorem proving (more general). We will explore \textit{logics} for option (2) in this lecture.
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
Suppose that a concurrent system has been specified by a rewrite theory \( R \), a top sort \( State \) of states has been chosen, and some \textit{state predicates} \( \Pi \) have also been defined.
Suppose that a concurrent system has been specified by a rewrite theory $\mathcal{R}$, a top sort $\textit{State}$ of states has been chosen, and some \textit{state predicates} $\Pi$ have also been defined.

We can then define properties of $\mathcal{R}$ in \textit{Hoare Logic} by means of so-called \textit{Hoare Triples} of the form:
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 $\Pi$ 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 $\Pi$ 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 $\Pi$ 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 $\textit{State}$ of states has been chosen, and some \textit{state predicates} $\Pi$ have also been defined.

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

$A$ is called the \textit{precondition} and $B$ the \textit{postcondition}.
Suppose that a concurrent system has been specified by a rewrite theory \( \mathcal{R} \), a top sort \( \text{State} \) of states has been chosen, and some \textit{state predicates} \( \Pi \) have also been defined.

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

\( A \) is called the \textit{precondition} and \( B \) the \textit{postcondition}. The Hoare triple \( \{A\} \mathcal{R} \{B\} \) is a \textit{partial correctness assertion} about \( \mathcal{R} \):
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 $\Pi$ 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 $\Pi$ 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,
Suppose that a concurrent system has been specified by a rewrite theory $\mathcal{R}$, a top sort $\text{State}$ of states has been chosen, and some state predicates $\Pi$ 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 $\Pi$ 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 $[u_0] \in \mathcal{T}_{\mathcal{R}, \text{State}}$, if $[u_0]$ satisfies the precondition $A$, then,
- for each terminating sequence of transitions

$$[u_0] \rightarrow_{\mathcal{R}} [u_1] \ldots [u_{n-1}] \rightarrow_{\mathcal{R}} [u_n]$$
Suppose that a concurrent system has been specified by a rewrite theory \( \mathcal{R} \), a top sort \( \textit{State} \) of states has been chosen, and some \textit{state predicates} \( \Pi \) have also been defined.

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

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

- for each state \([u_0]\) \( \in \mathcal{T}_{\mathcal{R},\text{State}} \), if \([u_0]\) satisfies the precondition \( A \), then,
- for each \textit{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 \( B \).
What formulas $A$ and $B$ shall we use in a Hoare triple $\{A\} \mathcal{R} \{B\}$?
What formulas $A$ and $B$ shall we use in a Hoare triple $\{A\} \mathcal{R} \{B\}$? Assuming $\mathcal{R} = (\Sigma, B, R)$ has constructors $\Omega$, we can use *pattern predicates* of the form $u \models \phi$ where $u$ is an $\Omega$-term of sort $State$ and $\phi$ is a $\Sigma$-condition.
What formulas \( A \) and \( B \) shall we use in a Hoare triple \( \{ A \} R \{ B \} \)? Assuming \( R = (\Sigma, B, R) \) has constructors \( \Omega \), we can use \textit{pattern predicates} of the form \( u \mid \varphi \) where \( u \) is an \( \Omega \)-term of sort \textit{State} and \( \varphi \) is a \( \Sigma \)-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
$\{A\} R \{B\}$? Assuming $R = (\Sigma, B, R)$ has constructors $\Omega$, we can use pattern predicates of the form $u \mid \varphi$ where $u$ is an $\Omega$-term of sort $State$ and $\varphi$ is a $\Sigma$-condition. Then $u \mid \varphi$ denotes the set of its ground instance states:

$$\llbracket u \mid \varphi \rrbracket = \{[u\rho]_B \mid \rho \in [X \to T_\Omega] \land E \cup B \models \varphi \rho\}.$$
What formulas $A$ and $B$ shall we use in a Hoare triple $\{A\} \mathcal{R} \{B\}$? Assuming $\mathcal{R} = (\Sigma, B, R)$ has constructors $\Omega$, we can use *pattern predicates* of the form $u \mid \varphi$ where $u$ is an $\Omega$-term of sort $\text{State}$ and $\varphi$ is a $\Sigma$-condition. Then $u \mid \varphi$ denotes the set of its ground instance states:

$$\llbracket u \mid \varphi \rrbracket = \{[u\rho]_B \mid \rho \in [X \to T\Omega] \land E \cup B = \varphi\rho\}.$$ 

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


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

$$\llbracket u | \varphi \rrbracket = \{[u\rho]_B | \rho \in [X \rightarrow T\Omega] \land E \cup B \models \varphi\rho\}.$$ 

Let $Y = \text{vars}(A) \cap \text{vars}(B)$. 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 $\{A\} \ R \ {B}$? Assuming $R = (\Sigma, B, R)$ has constructors $\Omega$, we can use pattern predicates of the form $u \mid \varphi$ where $u$ is an $\Omega$-term of sort $\text{State}$ and $\varphi$ is a $\Sigma$-condition. Then $u \mid \varphi$ denotes the set of its ground instance states:

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

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

$$(\forall Y) \ \{A\} \ R \ {B}.$$ 

Let us see an example of a parametric Hoare triple involving a slight modification of the CHOICE module in Lecture 16.
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
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: \(\{\{U\} \mid \top\} \text{ CHOICE } \{\{N\} \mid N \subseteq U = tt\}\) is
\textit{parametric} on \(U\).
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: \{\{U\} | \top\} CHOICE \{\{N\} | N \subseteq U = tt\} is
parametric on \(U\). It states that for each \(U\) every final state
reachable from \(\{U\}\) is a singleton set \(\{N\}\) with \(N\) in \(U\).
Hoare logic is widely used to state and verify properties of an *imperative program* $p$ in, say, Java or C.
Hoare logic is widely used to state and verify properties of an \textit{imperative program} $p$ in, say, Java or C. It is then written in the form $\{A\} p \{B\}$.

Roșu, Stefanescu et al. at UIUC have made Hoare logic programming-language-independent by generalizing it to \textit{reachability logic}. Skeirik, Stefanescu and Meseguer at UIUC have in turn made reachability logic rewrite-theory-independent by defining it for rewrite theories $R$. 
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 $\{A'(p)\} \mathcal{R}_L \{B'\}$, where $\mathcal{L}$ is the imperative programming language, and $\mathcal{R}_L$ 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 $\{A'(p)\} \mathcal{R}_L \{B'\}$, where 
$L$ is the imperative programming language, and $\mathcal{R}_L$ is the 
*rewriting logic semantics* of $L$.

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.
Hoare logic is widely used to state and verify properties of an \textit{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 $\{A'(p)\} \mathcal{R}_L \{B'\}$, where $\mathcal{L}$ is the imperative programming language, and $\mathcal{R}_L$ is the \textit{rewriting logic semantics} of $\mathcal{L}$.

A serious difficulty with traditional Hoare logic is that it is \textit{language-dependent}. There is a Hoare logic for Java, another for C, and so on. Furthermore for each language \textit{different inference rules} must be defined and proved correct.
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 $\{A'(p)\} \mathcal{R}_L \{B'\}$, where $\mathcal{L}$ is the imperative programming language, and $\mathcal{R}_L$ 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.

Roşu, Stefanescu et al. at UIUC have made Hoare logic programming-language-independent by generalizing it 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 $\{A'(p)\} \mathcal{R}_L \{B'\}$, where $L$ is the imperative programming language, and $\mathcal{R}_L$ is the rewriting logic semantics of $L$.

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 $\mathcal{R}$. 

From Hoare Logic to Reachability Logic
Reachability Logic (RL) is:

Reachability Logic (RL) is:

- Parameterized over an underlying rewrite theory $R$
- Considers formulas $A \rightarrow \ast B$ where $A$ is a pattern predicate, and $B$ a disjunction of pattern predicates.
- Generalizes Hoare Logic partial correctness, i.e., $A \rightarrow \ast B$ generalizes $\{A\}R\{B\}$.
- Directly captures inductive reasoning in any theory $R$, unlike Hoare Logic, special rules for loops, etc, unnecessary.
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 $A \longrightarrow^\star B$ where $A$ is a pattern predicate, and $B$ a disjunction of pattern predicates.

$$u \mid \phi \longrightarrow^\star \bigvee_i v_i \mid \psi_i$$
Reachability Logic (RL) is:

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

$$u \mid \varphi \rightarrow^\ast \bigvee_{i} v_i \mid \psi_i$$

- a generalization of Hoare Logic *partial correctness*, i.e., $A \rightarrow^\ast B$ generalizes $\{A\} \mathcal{R} \{B\}$
Reachability Logic (RL) is:

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

\[
    u \mid \varphi \rightarrow^* \bigvee_{i} v_i \mid \psi_i
\]

- a generalization of Hoare Logic *partial correctness*, i.e., $A \rightarrow^* B$ 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 $A \rightarrow^{\ast} B$ mean?

A: Suppose we have:

1. a rewrite theory $R$
2. pattern formulas $A, B$
3. $\text{vars}(A) \cap \text{vars}(B) = \emptyset$

Then $A \rightarrow^{\ast} B$ means:

- For each state $[t] \in J_A$ and rewrite path $p$ from $[t]$, either:
  - $p$ crosses $J_B$ or
  - $p$ is infinite (counterex. satisfies $A \rightarrow^{\ast} B$, vacuously satisfies $A \rightarrow^{\ast} B$).
Q: What does the relation $A \rightarrow^\ast B$ mean?

A: Suppose we have:
Q: What does the relation $A \rightarrow^{\ast} B$ mean?

A: Suppose we have:
   (1) a rewrite theory $\mathcal{R}$
Q: What does the relation $A \rightarrow^* B$ mean?

A: Suppose we have:

(1) a rewrite theory $\mathcal{R}$
(2) pattern formulas $A, B$, $\text{vars}(A) \cap \text{vars}(B) = \emptyset$
Q: What does the relation $A \rightarrow^{\ast} B$ mean?

A: Suppose we have:
   (1) a rewrite theory $\mathcal{R}$
   (2) pattern formulas $A, B$,
   $\text{vars}(A) \cap \text{vars}(B) = \emptyset$
   (3) and terminating states $T$
Q: What does the relation $A \rightarrow^\ast B$ mean?

A: Suppose we have:
(1) a rewrite theory $\mathcal{R}$
(2) pattern formulas $A, B$,
\[ \text{vars}(A) \cap \text{vars}(B) = \emptyset \]
(3) and terminating states $T$

Then $A \rightarrow^\ast B$ means:
Q: What does the relation $A \xrightarrow{\diamondsuit} B$ mean?

A: Suppose we have:

1. a rewrite theory $\mathcal{R}$
2. pattern formulas $A, B$,
   
   $\text{vars}(A) \cap \text{vars}(B) = \emptyset$
3. and terminating states $T$

Then $A \xrightarrow{\diamondsuit} B$ means:

for each state $[t] \in \llbracket A \rrbracket$
Q: What does the relation $A \rightarrow^* B$ mean?

A: Suppose we have:
1. a rewrite theory $\mathcal{R}$
2. pattern formulas $A, B$, 
   $\text{vars}(A) \cap \text{vars}(B) = \emptyset$
3. and terminating states $T$

Then $A \rightarrow^* B$ means:
for each state $[t] \in [A]$
and rewrite path $p$ from $[t]$,
Q: What does the relation $A \xrightarrow{\ast} B$ mean?

A: Suppose we have:
   (1) a rewrite theory $\mathcal{R}$
   (2) pattern formulas $A, B$
       $\text{vars}(A) \cap \text{vars}(B) = \emptyset$
   (3) and terminating states $T$

Then $A \xrightarrow{\ast} B$ means:
for each state $[t] \in [A]$ and rewrite path $p$ from $[t]$, either:
Q: What does the relation $A \xrightarrow{\star} B$ mean?

A: Suppose we have:

1. a rewrite theory $\mathcal{R}$
2. pattern formulas $A, B$,
   
   $\text{vars}(A) \cap \text{vars}(B) = \emptyset$
3. and terminating states $T$

Then $A \xrightarrow{\star} B$ means:

for each state $[t] \in \mathcal{J}A$

and rewrite path $p$ from $[t]$, either:

1. $p$ crosses $[B]$ or
Q: What does the relation $A \rightarrow^{\ast} B$ mean?

A: Suppose we have:
(1) a rewrite theory $\mathcal{R}$
(2) pattern formulas $A, B$, $\text{vars}(A) \cap \text{vars}(B) = \emptyset$
(3) and terminating states $T$

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

- - - indicates counterex.
- - - satisfies $A \rightarrow^{\ast} B$
- - - vacuously satisfies
Let $\mathcal{R} = (\Sigma, E \cup B, R)$ be a rewrite theory with good executability conditions, and having a subsignature $\Omega$ 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 $\Omega$ of constructors and a chosen top sort $State$ of states. Let $C_\mathcal{R}$ denote the canonical reachability model.
Let \( \mathcal{R} = (\Sigma, E \cup B, R) \) be a rewrite theory with good executability conditions, and having a subsignature \( \Omega \) of constructors and a chosen top sort \( \text{State} \) of states. Let \( C_{\mathcal{R}} \) denote the canonical reachability model. For a reachability formula \( A \rightarrow ^\ast B \) call \( Y = \text{vars}(A) \cap \text{vars}(B) \) 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 $\Omega$ of constructors and a chosen top sort $State$ of states. Let $C_\mathcal{R}$ denote the canonical reachability model. For a reachability formula $A \rightarrow^{\star} B$ call $Y = \text{vars}(A) \cap \text{vars}(B)$ its parameters.

If $Y = \emptyset$, then we write $\mathcal{R} \models A \rightarrow^{\star} B$ iff
Let $R = (\Sigma, E \cup B, R)$ be a rewrite theory with good executability conditions, and having a subsignature $\Omega$ of constructors and a chosen top sort $State$ of states. Let $C_R$ denote the canonical reachability model. For a reachability formula $A \rightarrow^\ast B$ call $Y = \text{vars}(A) \cap \text{vars}(B)$ its \textit{parameters}.

If $Y = \emptyset$, then we write $R \models A \rightarrow^\ast B$ iff for each $[u_0] \in C_{R,State}$ such that $[u_0] \in [[A]]$
Let $\mathcal{R} = (\Sigma, E \cup B, R)$ be a rewrite theory with good executability conditions, and having a subsignature $\Omega$ of constructors and a chosen top sort $\text{State}$ of states. Let $\mathcal{C}_R$ denote the canonical reachability model. For a reachability formula $A \rightarrow^\star B$ call $Y = \text{vars}(A) \cap \text{vars}(B)$ its parameters.

If $Y = \emptyset$, then we write $\mathcal{R} \models A \rightarrow^\star B$ iff for each $[u_0] \in \mathcal{C}_{R, \text{State}}$ such that $[u_0] \in \llbracket A \rrbracket$ and each terminating sequence:
Let $\mathcal{R} = (\Sigma, E \cup B, R)$ be a rewrite theory with good executability conditions, and having a subsignature $\Omega$ of constructors and a chosen top sort $\text{State}$ of states. Let $\mathcal{C}_R$ denote the canonical reachability model. For a reachability formula $A \rightarrow^\ast B$ call $Y = \text{vars}(A) \cap \text{vars}(B)$ its parameters.

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

$$[u_0] \rightarrow_{\mathcal{C}_R} [u_1] \ldots [u_{n-1}] \rightarrow_{\mathcal{C}_R} [u_n]$$
Let $\mathcal{R} = (\Sigma, E \cup B, R)$ be a rewrite theory with good executability conditions, and having a subsignature $\Omega$ of constructors and a chosen top sort $\text{State}$ of states. Let $\mathcal{C}_R$ denote the canonical reachability model. For a reachability formula $A \rightarrow^{\ast} B$ call $Y = \text{vars}(A) \cap \text{vars}(B)$ its parameters.

If $Y = \emptyset$, then we write $\mathcal{R} \models A \rightarrow^{\ast} B$ iff for each $[u_0] \in \mathcal{C}_R, \text{State}$ such that $[u_0] \in [A]$ and each terminating sequence:

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

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 $\Omega$ of constructors and a chosen top sort $\textit{State}$ of states. Let $\mathcal{C}_\mathcal{R}$ denote the canonical reachability model. For a reachability formula $A \rightarrow^\oplus B$ call $Y = \text{vars}(A) \cap \text{vars}(B)$ its parameters.

If $Y = \emptyset$, then we write $\mathcal{R} \models A \rightarrow^\oplus B$ 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 $\mathcal{R} \models A \rightarrow^\oplus B$ 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 $\Omega$ of constructors and a chosen top sort $State$ of states. Let $C_\mathcal{R}$ denote the canonical reachability model. For a reachability formula $A \rightarrow^* B$ call $Y = \text{vars}(A) \cap \text{vars}(B)$ its parameters.

If $Y = \emptyset$, then we write $\mathcal{R} \models A \rightarrow^* B$ iff for each $[u_0] \in C_\mathcal{R,State}$ such that $[u_0] \in \llbracket A \rrbracket$ and each terminating sequence:

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

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

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

Precise Definition

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

If $Y = \emptyset$, then we write $\mathcal{R} \models A \rightarrow^\ast B$ iff for each $[u_0] \in C_\mathcal{R},\text{State}$ such that $[u_0] \in \llbracket A \rrbracket$ and each terminating sequence:

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

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

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

That is, the parameters $Y$ in $A \rightarrow^\ast B$ are universally quantified,
Let $\mathcal{R} = (\Sigma, E \cup B, R)$ be a rewrite theory with good executability conditions, and having a subsignature $\Omega$ of constructors and a chosen top sort $\text{State}$ of states. Let $\mathcal{C}_\mathcal{R}$ denote the canonical reachability model. For a reachability formula $A \rightarrow^\ast B$ call $Y = \text{vars}(A) \cap \text{vars}(B)$ its parameters.

If $Y = \emptyset$, then we write $\mathcal{R} \models A \rightarrow^\ast B$ iff for each $[u_0] \in \mathcal{C}_\mathcal{R},\text{State}$ such that $[u_0] \in \llbracket A \rrbracket$ 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 $[u_j] \in \llbracket B \rrbracket$.

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

That is, the parameters $Y$ in $A \rightarrow^\ast B$ are universally quantified, so that $A \rightarrow^\ast B$ implicitly means: $(\forall Y) A \rightarrow^\ast B$. 
Q: How is a Hoare triple $\{A\}R\{B\}$ expressed in reachability logic?

A: as the formula $A \rightarrow \star (B \land T)$, with $J T K$ the terminating states.

Q: How is a reachability logic sequent $A \rightarrow \star B$ expressed in linear temporal logic?

A: as the LTL formula $A \rightarrow (\square \text{enabled}) \lor \diamond B$.

Example. For CHOICE, the formula

$$\{\text{UN}\} | N \subseteq U \neq \text{tt} \rightarrow \star \{V\} | V \subseteq \text{UN} = \text{tt} \land (N \subseteq V \neq \text{tt} \lor V = N)$$

is parametric on $U$ and $N$. It states that for each $\text{UN}$ where $U$ does not contain $N$ we must reach a submultiset $V$ of $\text{UN}$ that either does not containing $N$ or $V = N$.

Note that this reachability property cannot be expressed by a Hoare triple.
Q: How is a Hoare triple \( \{A\} \mathcal{R} \{B\} \) expressed in reachability logic?
A: as the formula \( A \xrightarrow{\star} (B \land T) \), with \([T]\) the terminating states.

Example. For \(\text{CHOICE}\), the formula
\[
\{\text{UN}\} | N \subseteq U \neq \text{tt} \xrightarrow{\star} \{V\} | V \subseteq U \neq \text{tt} \land (N \subseteq V \neq \text{tt} \lor V = N)
\]
is parametric on \(U\) and \(N\).

It states that for each \(\text{UN}\) where \(U\) does not contain \(N\) we must reach a submultiset \(V\) of \(\text{UN}\) that either does not containing \(N\) or \(V = N\).

Note that this reachability property cannot be expressed by a Hoare triple.
Q: How is a Hoare triple \( \{ A \} \mathcal{R} \{ B \} \) expressed in reachability logic?

A: as the formula \( A \xrightarrow{\ast} (B \land T) \), with \( [T] \) the terminating states.

Q: How is a reachability logic sequent \( A \xrightarrow{\ast} B \) expressed in linear temporal logic?

Example. For CHOICE, the formula
\[
\{ \text{UN} \mid N \subseteq U \neq \tt \} \xrightarrow{\ast} \{ V \mid V \subseteq \text{UN} = \tt \land (N \subseteq V \neq \tt \lor V = N) \}
\]
is parametric on \( U \) and \( N \). It states that for each \( \text{UN} \) where \( U \) does not contain \( N \) we must reach a submultiset \( V \) of \( \text{UN} \) that either does not containing \( N \) or \( V = N \). Note that this reachability property cannot be expressed by a Hoare triple.
Q: How is a Hoare triple $\{A\} \mathcal{R} \{B\}$ expressed in reachability logic?

A: as the formula $A \longrightarrow \ast (B \land \mathbb{[}T\mathbb{]})$, with $\mathbb{[}T\mathbb{]}$ the terminating states.

Q: How is a reachability logic sequent $A \longrightarrow \ast B$ expressed in linear temporal logic?

A: as the LTL formula $A \rightarrow (\square enabled) \lor \diamond B$. 

Example. For CHOICE, the formula $\{UN\} | N \subseteq U \neq tt \longrightarrow \ast \{V\} | V \subseteq UN = tt \land (N \subseteq V \neq tt \lor V = N)$ is parametric on $U$ and $N$. It states that for each $UN$ where $U$ does not contain $N$ we must reach a submultiset $V$ of $UN$ that either does not containing $N$ or $V = N$. Note that this reachability property cannot be expressed by a Hoare triple.
Q: How is a Hoare triple $\{A\} \mathcal{R} \{B\}$ expressed in reachability logic?

A: as the formula $A \xrightarrow{\star} (B \land T)$, with $[T]$ the terminating states.

Q: How is a reachability logic sequent $A \xrightarrow{\star} B$ expressed in linear temporal logic?

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

Example. For CHOICE, the formula

$$\{U N\} \mid N \subseteq U \neq \text{tt} \xrightarrow{\star} \{V\} \mid V \subseteq U \text{ } N = \text{tt} \land (N \subseteq V \neq \text{tt} \lor V = N)$$
Q: How is a Hoare triple \( \{ A \} R \{ B \} \) expressed in reachability logic?

**A:** as the formula \( A \rightarrow^\star (B \land T) \), with \([T]\) the terminating states.

Q: How is a reachability logic sequent \( A \rightarrow^\star B \) expressed in linear temporal logic?

**A:** as the LTL formula \( A \rightarrow (\square \text{enabled}) \lor \diamond B \).

**Example.** For CHOICE, the formula

\[
\{ U \mid N \subseteq U \neq tt \} \rightarrow^\star \{ V \mid V \subseteq U \mid N = tt \land (N \subseteq V \neq tt \lor V = N) \}
\]

is *parametric* on \( U \) and \( N \).
Q: How is a Hoare triple {A}R{B} expressed in reachability logic?
A: as the formula $A \xrightarrow{\ominus} (B \land T)$, with $[T]$ the terminating states.

Q: How is a reachability logic sequent $A \xrightarrow{\ominus} B$ expressed in linear temporal logic?
A: as the LTL formula $A \rightarrow (\Box \text{enabled}) \lor \Diamond B$.

Example. For CHOICE, the formula

$$\{U N\} \mid N \subseteq U \neq tt \xrightarrow{\ominus} \{V\} \mid V \subseteq U \land (N \subseteq V \neq tt \lor V = N)$$

is parametric on $U$ and $N$. It states that for each $U N$ where $U$ does not contain $N$ we must reach a submultiset $V$ of $U N$ that either does not contain $N$ or $V = N$. 

Note that this reachability property cannot be expressed by a Hoare triple.
Q: How is a Hoare triple \( \{A\} \mathcal{R} \{B\} \) expressed in reachability logic?

A: as the formula \( A \rightarrow^\otimes (B \land T) \), with \([T]\) the terminating states.

Q: How is a reachability logic sequent \( A \rightarrow^\otimes B \) expressed in linear temporal logic?

A: as the LTL formula \( A \rightarrow (\square \text{enabled}) \lor \diamond B \).

Example. For CHOICE, the formula

\[
\{U \; N\} \mid N \subseteq U \neq tt \rightarrow^\otimes \{V\} \mid V \subseteq U \; N = tt \land (N \subseteq V \neq tt \lor V = N)
\]

is parametric on \( U \) and \( N \). It states that for each \( U \; N \) where \( U \) does not contain \( N \) we must reach a submultiset \( V \) of \( U \; N \) that either does not containing \( N \) or \( V = N \). Note that this reachability property cannot be expressed by a Hoare triple.
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 $A \rightarrow \Rightarrow B$?

A: Since:
   (i) $A \rightarrow \Rightarrow B$ just means $A \rightarrow (\Box \text{enabled}) \lor \Diamond B$, and
   (ii) READERS-WRITERS is a never terminating rewrite theory, all formulas $A \rightarrow \Rightarrow B$ are satisfied!!

So we cannot!! (Paradox!!).
Consider the readers and writers example (Lecture 18):

```plaintext
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 $A \rightarrow \square B$?

A: Since:

(i) $A \rightarrow \square B$ just means $A \rightarrow (\square \text{enabled}) \lor \diamond B$,

(ii) READERS-WRITERS is a never terminating rewrite theory, all formulas $A \rightarrow \square B$ are satisfied!!

So we cannot (Paradox!!).

The Invariant Paradox
Consider the readers and writers example (Lecture 18):

```plaintext
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 \( A \rightarrow^\ast B \)?
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 $A \rightarrow^\ast B$?

A: Since:
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 $A \rightarrow^\ast B$?

A: Since: (i) $A \rightarrow^\ast B$ just means $A \rightarrow (\Box \text{enabled}) \lor \Diamond B$, and
The Invariant Paradox

Consider the readers and writers example (Lecture 18):

\[
\text{mod READERS-WRITERS is}
\]
\[
\text{protecting NAT .}
\]
\[
\text{sort State .}
\]
\[
\text{op } <_,_> : \text{Nat Nat } \to \text{State [ctor] . --- readers/writers}
\]
\[
\text{vars } R \ W : \text{Nat .}
\]
\[
\text{rl } < 0, 0 > \Rightarrow < 0, s(0) > .
\]
\[
\text{rl } < R, s(W) > \Rightarrow < R, W > .
\]
\[
\text{rl } < R, 0 > \Rightarrow < s(R), 0 > .
\]
\[
\text{rl } < s(R), W > \Rightarrow < R, W > .
\]
\[
\text{endm}
\]

Q: How can we express its mutual exclusion invariant as a reachability formula \( A \longrightarrow^{\star} B \)?

A: Since: (i) \( A \longrightarrow^{\star} B \) just means \( A \to (\square \text{enabled}) \lor \Diamond B \), and (ii) READERS-WRITERS is a never terminating rewrite theory,
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 \( A \rightarrow^{\ast} B \)?

A: Since: (i) \( A \rightarrow^{\ast} B \) just means \( A \rightarrow (\square enabled) \lor \Diamond B \), and (ii) READERS–WRITERS is a never terminating rewrite theory, all formulas \( A \rightarrow^{\ast} B \) are satisfied!!
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 \(A \longrightarrow^\ast B\)?

A: Since: (i) \(A \longrightarrow^\ast B\) just means \(A \rightarrow (\Box \text{enabled}) \lor \Diamond B\), and 
(ii) READERS–WRITERS is a *never terminating* rewrite theory, *all* 
formulas \(A \longrightarrow^\ast B\) are satisfied!! So we *cannot*!!
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 $A \rightarrow^{\circ} B$?

A: Since: (i) $A \rightarrow^{\circ} B$ just means $A \rightarrow (\square \text{enabled}) \lor \lozenge B$, and (ii) READERS-WRITERS is a *never terminating* rewrite theory, all formulas $A \rightarrow^{\circ} B$ 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:

```plaintext
mod READERS–WRITERS–stop is
   protecting NAT .
   sort State .
   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 rule `< R, W > => [R,W]` can now stop any state and make it terminating.

For any pattern predicate `B = ⟨u,v⟩|ϕ` let `[B]` denote the pattern predicate `[u,v]|ϕ`.

Fact.

`B` is an invariant from states `S₀` in READERS–WRITERS iff `S₀ → ⊛[B]` holds in READERS–WRITERS–stop.
Solving the Invariant Paradox

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

mod READERS–WRITERS–stop is
  protecting NAT .
sort State .
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 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:

```plaintext
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 $[B] = [u, v] \mid \varphi$. 
Let us add a *stopwatch* to READERS–WRITERS as follows:

```plaintext
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 \( B = \langle u, v \rangle | \varphi \) let \( [B] \) denote the pattern predicate \( [B] = [u, v] | \varphi \).

**Fact.**  \( B \) is an *invariant* from 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 .
    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 rule \(< R, W > => [R,W] \) can now *stop* any state and make it terminating. For any pattern predicate \(B = \langle u, v \rangle | \varphi\) let \([B]\) denote the pattern predicate \([B] = \langle u, v \rangle | \varphi\).

**Fact.** \(B\) is an *invariant* from states \(S_0\) in READERS–WRITERS iff
Let us add a *stopwatch* to READERS–WRITERS as follows:

mod READERS–WRITERS–stop is

protectioning NAT .

sort State .


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

**Fact.** $B$ is an *invariant* from states $S_0$ in READERS–WRITERS iff $S_0 \xrightarrow{\circ} [B]$ holds in in READERS–WRITERS–stop.
Suppose $\mathcal{R}$ is *never terminating* (has no terminating states),
Suppose $\mathcal{R}$ is *never terminating* (has no terminating states), $State$ has a single constructor $\langle _, \ldots, _ \rangle : s_1 \ldots s_n \rightarrow State$, 

Theorem B is an invariant for $\mathcal{R}$ from initial states $S_0$ iff $S_0 \xrightarrow{\ast} \Box [B]$ holds in $\mathcal{R}_{stop}$.

Corollary If $J S_0 K \subseteq J B K$ and $B \xrightarrow{\ast} \Box [B\sigma]$ (\(\sigma\) variable renaming) holds in $\mathcal{R}_{stop}$, then $B$ is an invariant for $\mathcal{R}$ from initial states $S_0$.

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

(i) $\langle 0, 0 \rangle \in Mutex$ (easy), and

(ii) $Mutex \xrightarrow{\ast} \Box [Mutex\sigma]$ in READERS-WRITERS-stop.
Solving the Invariant Paradox (General Case)

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

\begin{itemize}
  \item [(i)] $\langle\_,\ldots,\_\rangle : s_1 \ldots s_n \rightarrow State$,
  \item [(ii)] a \emph{stop rule} $\langle x_1,\ldots,x_n \rangle \rightarrow \langle x_1,\ldots,x_n \rangle$.
\end{itemize}

Then:

\begin{itemize}
  \item Theorem $B$ is an invariant for $\mathcal{R}$ from initial states $S_0$ iff $S_0 \rightarrow \bigstar B$ holds in $\mathcal{R}_{stop}$.
  \item Corollary If $J \subseteq J_B$ and $B \rightarrow \bigstar B\sigma$ (\(\sigma\) variable renaming) holds in $\mathcal{R}_{stop}$, then $B$ is an invariant for $\mathcal{R}$ from initial states $S_0$.
\end{itemize}

\begin{example}
Mutual exclusion from $\langle 0,0 \rangle$ in READERS-WRITERS is the predicate:

$Mutex = \langle R,W \rangle | W = 0 \lor (W = 1 \land R = 0)$.

We can prove it by showing:

\begin{itemize}
  \item [(i)] $\langle 0,0 \rangle \in Mutex$ (easy), and
  \item [(ii)] $Mutex \rightarrow \bigstar Mutex\sigma$ in READERS-WRITERS-stop.
\end{itemize}
\end{example}
Suppose \( \mathcal{R} \) is *never terminating* (has no terminating states), \( \text{State} \) has a single constructor \( \langle \_, \ldots, \_ \rangle : s_1 \ldots s_n \rightarrow \text{State} \), and all rules are between terms of sort \( \text{State} \). Call \( \mathcal{R}_{\text{stop}} \) the rewrite theory extending \( \mathcal{R} \) by adding: (i) \( \langle \_, \ldots, \_ \rangle : s_1 \ldots s_n \rightarrow \text{State} \), and
Suppose $\mathcal{R}$ is never terminating (has no terminating states), $State$ has a single constructor $\langle -, \ldots, - \rangle : s_1 \ldots s_n \rightarrow State$, and all rules are between terms of sort $State$. Call $\mathcal{R}_{stop}$ the rewrite theory extending $\mathcal{R}$ by adding: (i) $[-, \ldots, -] : s_1 \ldots s_n \rightarrow State$, and (ii) a stop rule $\langle x_1, \ldots, x_n \rangle \rightarrow [x_1, \ldots, x_n]$. 

Theorem $B$ is an invariant for $\mathcal{R}$ from initial states $S_0$ iff $S_0 \rightarrow \ast \backslash [B]$ holds in $\mathcal{R}_{stop}$.

Corollary If $J \subseteq J_B$ and $B \rightarrow \ast \backslash [B\sigma]$ (σ variable renaming) holds in $\mathcal{R}_{stop}$, then $B$ is an invariant for $\mathcal{R}$ from initial states $S_0$.

Example. Mutual exclusion from $\langle 0, 0 \rangle$ in READERS-WRITERS is the predicate: $\text{Mutex} = \langle R, W \rangle | W = 0 \lor (W = 1 \land R = 0)$. We can prove it by showing: (i) $\langle 0, 0 \rangle \in \text{Mutex}$ (easy), and (ii) $\text{Mutex} \rightarrow \ast \backslash [\text{Mutex}\sigma]$ in READERS-WRITERS-stop.
Suppose $\mathcal{R}$ is \textit{never terminating} (has no terminating states), $\text{State}$ has a single constructor $\langle \_, \ldots, \_ \rangle : s_1 \ldots s_n \rightarrow \text{State}$, and all rules are between terms of sort $\text{State}$. Call $\mathcal{R}_{\text{stop}}$ the rewrite theory extending $\mathcal{R}$ by adding: (i) $\langle \_, \ldots, \_ \rangle : s_1 \ldots s_n \rightarrow \text{State}$, and (ii) a \textit{stop rule} $\langle x_1, \ldots, x_n \rangle \rightarrow \lbrack x_1, \ldots, x_n \rbrack$. Then:
Solving the Invariant Paradox (General Case)

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

\textbf{Theorem}

\textit{B} is an \textit{invariant} for \( R \) from initial states \( S_0 \) iff \( S_0 \xrightarrow{\ast} [B] \) holds in \( R_{\text{stop}} \).
Suppose $R$ is *never terminating* (has no terminating states), $State$ has a single constructor $\langle \_, \ldots, \_ \rangle : s_1 \ldots s_n \rightarrow State$, and all rules are between terms of sort $State$. Call $R_{stop}$ the rewrite theory extending $R$ by adding: (i) $\langle \_, \ldots, \_ \rangle : s_1 \ldots s_n \rightarrow State$, and (ii) a stop rule $\langle x_1, \ldots, x_n \rangle \rightarrow [x_1, \ldots, x_n]$. Then:

**Theorem**

$B$ is an invariant for $R$ from initial states $S_0$ iff $S_0 \rightarrow^{\ast} [B]$ holds in $R_{stop}$.

**Corollary**

If $[S_0] \subseteq [B]$ and $B \rightarrow^{\ast} [B\sigma]$ ($\sigma$ variable renaming) holds in $R_{stop}$, then $B$ is an invariant for $R$ from initial states $S_0$. 

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

$ Mutex = \langle \text{R, W} \rangle | W = 0 \lor (W = 1 \land R = 0) $.

We can prove it by showing:

(i) $\langle 0, 0 \rangle \in Mutex$ (easy), and

(ii) $Mutex \rightarrow^{\ast} \emptyset [Mutex\sigma]$ in READERS-WRITERS-stop.
Solving the Invariant Paradox (General Case)

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

**Theorem**
\( B \) is an invariant for \( \mathcal{R} \) from initial states \( S_0 \) iff \( S_0 \rightarrow^{\ast} [B] \) holds in \( \mathcal{R}_{\text{stop}} \).

**Corollary**
If \( [S_0] \subseteq [B] \) and \( B \rightarrow^{\ast} [B\sigma] \) (\( \sigma \) variable renaming) holds in \( \mathcal{R}_{\text{stop}} \), then \( B \) is an invariant for \( \mathcal{R} \) from initial states \( S_0 \).

**Example.** Mutual exclusion from \( \langle 0, 0 \rangle \) in READERS-WRITERS is the predicate:
Suppose $R$ is never terminating (has no terminating states), $State$ has a single constructor $⟨_,...,_⟩ : s_1 \ldots s_n \rightarrow State$, and all rules are between terms of sort $State$. Call $R_{stop}$ the rewrite theory extending $R$ by adding: (i) $[_,...,_] : s_1 \ldots s_n \rightarrow State$, and (ii) a stop rule $⟨x_1,\ldots,x_n⟩ \rightarrow [x_1,\ldots,x_n]$. Then:

**Theorem**

$B$ is an invariant for $R$ from initial states $S_0$ iff $S_0 \rightarrow^\ast [B]$ holds in $R_{stop}$.

**Corollary**

If $[S_0] \subseteq [B]$ and $B \rightarrow^\ast [B\sigma]$ ($\sigma$ variable renaming) holds in $R_{stop}$, then $B$ is an invariant for $R$ from initial states $S_0$.

**Example.** Mutual exclusion from $⟨0,0⟩$ in READERS–WRITERS is the predicate: $Mutex = ⟨R,W⟩ | \ W = 0 \lor (W = 1 \land R = 0)$. 


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

\textbf{Theorem}

\( B \) is an \textit{invariant} for \( \mathcal{R} \) from initial states \( S_0 \) iff \( S_0 \xrightarrow{\ast} [B] \) holds in \( \mathcal{R}_{\text{stop}} \).

\textbf{Corollary}

If \( [S_0] \subseteq [B] \) and \( B \xrightarrow{\ast} [B\sigma] \) (\( \sigma \) variable renaming) holds in \( \mathcal{R}_{\text{stop}} \), then \( B \) is an \textit{invariant} for \( \mathcal{R} \) from initial states \( S_0 \).

\textbf{Example}. Mutual exclusion from \( \langle 0, 0 \rangle \) in READERS–WRITERS is the predicate: \( \text{Mutex} = \langle R, W \rangle \mid W = 0 \lor (W = 1 \land R = 0) \). We can prove it by showing:
Suppose $\mathcal{R}$ is *never terminating* (has no terminating states), $\textit{State}$ has a single constructor $\langle -,\ldots, - \rangle : s_1 \ldots s_n \rightarrow \textit{State}$, and all rules are between terms of sort $\textit{State}$. Call $\mathcal{R}_{\text{stop}}$ the rewrite theory extending $\mathcal{R}$ by adding: (i) $\langle -,\ldots, - \rangle : s_1 \ldots s_n \rightarrow \textit{State}$, and (ii) a *stop rule* $\langle x_1,\ldots, x_n \rangle \rightarrow \langle x_1,\ldots, x_n \rangle$. Then:

**Theorem**

$B$ is an invariant for $\mathcal{R}$ from initial states $S_0$ iff $S_0 \rightarrow^{\ast} [B]$ holds in $\mathcal{R}_{\text{stop}}$.

**Corollary**

If $[S_0] \subseteq [B]$ and $B \rightarrow^{\ast} [B\sigma]$ ($\sigma$ variable renaming) holds in $\mathcal{R}_{\text{stop}}$, then $B$ is an invariant for $\mathcal{R}$ from initial states $S_0$.

**Example.** Mutual exclusion from $\langle 0, 0 \rangle$ in READERS-WRITERS is the predicate: $\text{Mutex} = \langle R, W \rangle \mid W = 0 \lor (W = 1 \land R = 0)$. We can prove it by showing: (i) $\langle 0, 0 \rangle \in \text{Mutex}$ (easy), and
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 $\mathcal{R}_{\text{stop}}$ the rewrite theory extending $\mathcal{R}$ by adding: (i) $\langle _, \ldots, _ \rangle : s_1 \ldots s_n \to State$, 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 states $S_0$ iff $S_0 \to^{\ast} [B]$ holds in $\mathcal{R}_{\text{stop}}$.

**Corollary**

If $[S_0] \subseteq [B]$ and $B \to^{\ast} [B\sigma]$ ($\sigma$ variable renaming) holds in $\mathcal{R}_{\text{stop}}$, then $B$ is an invariant for $\mathcal{R}$ from initial states $S_0$.

**Example.** Mutual exclusion from $\langle 0, 0 \rangle$ in READERS–WRITERS is the predicate: $\text{Mutex} = \langle R, W \rangle \mid W = 0 \lor (W = 1 \land R = 0)$. We can prove it by showing: (i) $\langle 0, 0 \rangle \in \text{Mutex}$ (easy), and (ii) $\text{Mutex} \to^{\ast} [\text{Mutex}\sigma]$ in READERS–WRITERS–stop.
Q: Then given RWL theory $\mathcal{R}$, how do we prove $A \rightarrow^\star B$?
Q: Then given RWL theory $\mathcal{R}$, how do we prove $A \xrightarrow{\otimes} B$?

A: Three proof rules are enough:
Q: Then given RWL theory $\mathcal{R}$, how do we prove $A \xrightarrow{\circledast} B$?

A: Three proof rules are enough:

- A rule that traces rewrite steps of symbolic states in $\mathcal{R}$
Q: Then given RWL theory \( \mathcal{R} \), how do we prove \( A \rightarrow^\circ B \)?

A: Three proof rules are enough:

- A rule that traces *rewrite steps* of *symbolic* states in \( \mathcal{R} \)
- A rule that performs *inductive reasoning* in \( \mathcal{R} \)
Q: Then given RWL theory $\mathcal{R}$, how do we prove $A \longrightarrow^\mathcal{R} B$?

A: Three proof rules are enough:

- A rule that traces *rewrite steps* of *symbolic* states in $\mathcal{R}$
- A rule that performs *inductive reasoning* in $\mathcal{R}$
- A rule that checks if we have “arrived,” i.e. *discharges* goals that trivially hold.
Q: Then given RWL theory \( \mathcal{R} \), how do we prove \( A \xrightarrow{\circ} B \)?

A: Three proof rules are enough:

- A rule that traces *rewrite steps* of *symbolic* states in \( \mathcal{R} \)
- A rule that performs *inductive reasoning* in \( \mathcal{R} \)
- A rule that checks if we have “arrived,” i.e. *discharges* goals that trivially hold.

We call these rules **Step**, **Axiom**, and **Subsumption** resp.
Q: Then given RWL theory $\mathcal{R}$, how do we prove $A \rightarrow^{\text{\#}} B$?

A: Three proof rules are enough:

- A rule that traces *rewrite steps* of *symbolic* states in $\mathcal{R}$
- A rule that performs *inductive reasoning* in $\mathcal{R}$
- A rule that checks if we have “arrived,” i.e. *discharges* goals that trivially hold.

We call these rules **Step**, **Axiom**, and **Subsumption** resp.
Q: Then given RWL theory $\mathcal{R}$, how do we prove $A \rightarrow^\star B$?
A: (Continued) The key ideas are:
Q: Then given RWL theory $\mathcal{R}$, how do we prove $A \xrightarrow{\ast} B$?

A: (Continued) The key ideas are:

1. to prove $A \xrightarrow{\ast} B$ we may need some *auxiliary lemmas*;
   let $C$ be the formula $A \xrightarrow{\ast} B$ plus these lemmas;
Q: Then given RWL theory $\mathcal{R}$, how do we prove $A \rightarrow^\ast B$?

A: (Continued) The key ideas are:

1. to prove $A \rightarrow^\ast B$ we may need some auxiliary lemmas; let $C$ be the formula $A \rightarrow^\ast B$ plus these lemmas;

2. we start with labeled sequents of the form $[\emptyset, C] \triangleright^T u | \varphi \rightarrow^\ast \bigvee_i v_i | \psi_i$ for all formulas in $C$;
Q: Then given RWL theory $\mathcal{R}$, how do we prove $A \xrightarrow{\star} B$?

A: (Continued) The key ideas are:

1. to prove $A \xrightarrow{\star} B$ we may need some auxiliary lemmas; let $C$ be the formula $A \xrightarrow{\star} B$ plus these lemmas;

2. we start with labeled sequents of the form $[\emptyset, C] \vdash T u \mid \varphi \xrightarrow{\star} \bigvee_i v_i \mid \psi_i$ for all formulas in $C$;

3. the first component ($\emptyset$) are the formulas we can assume as axioms via Axiom (initially none);
Q: Then given RWL theory $\mathcal{R}$, how do we prove $A \rightarrow^{\circ} B$?

A: (Continued) The key ideas are:

1. to prove $A \rightarrow^{\circ} B$ we may need some auxiliary lemmas; let $C$ be the formula $A \rightarrow^{\circ} B$ plus these lemmas;

2. we start with labeled sequents of the form $[\emptyset, C] \vdash_T u \mid \varphi \rightarrow^{\circ} \bigvee_i v_i \mid \psi_i$ for all formulas in $C$;

3. the first component ($\emptyset$) are the formulas we can assume as axioms via Axiom (initially none);

4. the second ($C$) are the formulas we need to prove and cannot yet assume;
Q: Then given RWL theory $\mathcal{R}$, how do we prove $A \rightarrow^* B$?

A: (Continued) The key ideas are:

1. to prove $A \rightarrow^* B$ we may need some auxiliary lemmas; let $C$ be the formula $A \rightarrow^* B$ plus these lemmas;

2. we start with labeled sequents of the form $[\emptyset, C] \vdash T u \mid \varphi \rightarrow^* \bigvee_i v_i \mid \psi_i$ for all formulas in $C$;

3. the first component ($\emptyset$) are the formulas we can assume as axioms via Axiom (initially none);

4. the second ($C$) are the formulas we need to prove and cannot yet assume;

5. Step lets us inductively assume $C$ after rewriting with rules $R = \{ l_j \rightarrow r_j \text{ if } \phi_j \}$.
Reachability Logic
Proof Rules

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

A: (Continued) The key ideas are:

1. to prove $A \xrightarrow{\ast} B$ we may need some auxiliary lemmas; let $C$ be the formula $A \xrightarrow{\ast} B$ plus these lemmas;

2. we start with labeled sequents of the form $[\emptyset, C] \vdash_T u | \varphi \xrightarrow{\ast} \bigvee_i v_i | \psi_i$ for all formulas in $C$;

3. the first component ($\emptyset$) are the formulas we can assume as axioms via Axiom (initially none);

4. the second ($C$) are the formulas we need to prove and cannot yet assume;

5. Step lets us inductively assume $C$ after rewriting with rules $R = \{ l_j \rightarrow r_j \text{ if } \phi_j \}$.

Reachability Logic
Proof Rules

\[ \bigwedge_{j \in J, \alpha \in \text{UNIFY}_B(u,l_j)} [A \cup C, \emptyset] \vdash_T (r_j \mid \varphi' \land \phi_j)_\alpha \rightarrow B \alpha \]

\[ [A, C] \vdash_T u \mid \varphi \rightarrow B \]

where:

1. \( R = \{ l_j \rightarrow r_j \text{ if } \phi_j \}_{j \in J} \)
2. \([u \mid \varphi'] \supseteq [u \mid \varphi] \setminus \llbracket B \rrbracket \)

The **Step** Rule
Reachability Logic
Proof Rules

\[ \bigwedge_j \{ u' | \varphi' \xrightarrow{\ast} \bigvee_j v'_j | \psi'_j \} \cup A, \emptyset \models_T v'_j \alpha | \varphi \land \psi'_j \alpha \xrightarrow{\ast} B \]

\[ \{ u' | \varphi' \xrightarrow{\ast} \bigvee_j v'_j | \psi'_j \} \cup A, \emptyset \models_T u | \varphi \xrightarrow{\ast} B \]

where:

1. \( u' \alpha =_B u \)
2. \( [u | \varphi] \subseteq [(u' | \varphi') \alpha] \)

The Axiom Rule
Reachability Logic
Proof Rules

\[ [A, C] \vdash_T A \rightarrow^* B \]

where: \([A] \subseteq [B]\)

The Subsumption Rule
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.

The paper on Reachability Logic in the course web page is recommended as extra reading.

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

The paper on Reachability Logic in the course web page is recommended as extra reading.

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

The paper on Reachability Logic in the course web page is recommended as extra reading.

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

The paper on Reachability Logic in the course web page is recommended as extra reading. Next lecture will illustrate the use of the Maude Reachability Logic tool by means of examples.
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.

The paper on Reachability Logic in the course web page is recommended as extra reading.
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.

The paper on Reachability Logic in the course web page is recommended as extra reading.

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