Program Verification: Lecture 28

José Meseguer

University of Illinois at Urbana-Champaign

The Broader Picture: Narrowing with Constraints

Maude's `fvu-narrow` command allows us to symbolically model check invariants of infinite-state systems.

The Broader Picture: Narrowing with Constraints

Maude's `fvu-narrow` command allows us to symbolically model check invariants of infinite-state systems. But it has three limitations: (1) sets of states must be describable as disjunctions of unconstrained patterns;

The Broader Picture: Narrowing with Constraints

Maude's `fvu-narrow` command allows us to symbolically model check invariants of infinite-state systems. But it has three limitations: (1) sets of states must be describable as disjunctions of unconstrained patterns; (2) the equations $E \cup B$ must be FVP; and

The Broader Picture: Narrowing with Constraints

Maude's `fvu-narrow` command allows us to symbolically model check invariants of infinite-state systems. But it has three limitations: (1) sets of states must be describable as disjunctions of unconstrained patterns; (2) the equations $E \cup B$ must be FVP; and (3) the rules R must be unconditional.

The Broader Picture: Narrowing with Constraints

Maude's `fvu-narrow` command allows us to symbolically model check invariants of infinite-state systems. But it has three limitations: (1) sets of states must be describable as disjunctions of unconstrained patterns; (2) the equations $E \cup B$ must be FVP; and (3) the rules R must be unconditional.

Restrictions (1)–(3) evaporate when we perform constrained narrowing.

The Broader Picture: Narrowing with Constraints

Maude's `fvu-narrow` command allows us to symbolically model check invariants of infinite-state systems. But it has three limitations: (1) sets of states must be describable as disjunctions of unconstrained patterns; (2) the equations $E \cup B$ must be FVP; and (3) the rules R must be unconditional.

Restrictions (1)–(3) evaporate when we perform constrained narrowing. Given a constrained pattern $p \mid \varphi$ and a set R of conditional rewrite rules which, W.L.O.G., have disjoint variables from $p \mid \varphi$, the topmost constrained narrowing relation

The Broader Picture: Narrowing with Constraints

Maude's `fvu-narrow` command allows us to symbolically model check invariants of infinite-state systems. But it has three limitations: (1) sets of states must be describable as disjunctions of unconstrained patterns; (2) the equations $E \cup B$ must be FVP; and (3) the rules R must be unconditional.

Restrictions (1)–(3) evaporate when we perform constrained narrowing. Given a constrained pattern $p \mid \varphi$ and a set R of conditional rewrite rules which, W.L.O.G., have disjoint variables from $p \mid \varphi$, the topmost constrained narrowing relation

$$p \mid \varphi \overset{\theta}{\leadsto}_{R/E_1 \cup B_1} q \mid \psi$$

The Broader Picture: Narrowing with Constraints

Maude's `fvu-narrow` command allows us to symbolically model check invariants of infinite-state systems. But it has three limitations: (1) sets of states must be describable as disjunctions of unconstrained patterns; (2) the equations $E \cup B$ must be FVP; and (3) the rules R must be unconditional.

Restrictions (1)–(3) evaporate when we perform constrained narrowing. Given a constrained pattern $p \mid \varphi$ and a set R of conditional rewrite rules which, W.L.O.G., have disjoint variables from $p \mid \varphi$, the topmost constrained narrowing relation

$$p \mid \varphi \overset{\theta}{\leadsto}_{R/E_1 \cup B_1} q \mid \psi$$

modulo FVP $E_1 \cup B_1 \subseteq E \cup B$ holds iff

The Broader Picture: Narrowing with Constraints

Maude's `fvu-narrow` command allows us to symbolically model check invariants of infinite-state systems. But it has three limitations: (1) sets of states must be describable as disjunctions of unconstrained patterns; (2) the equations $E \cup B$ must be FVP; and (3) the rules R must be unconditional.

Restrictions (1)–(3) evaporate when we perform constrained narrowing. Given a constrained pattern $p \mid \varphi$ and a set R of conditional rewrite rules which, W.L.O.G., have disjoint variables from $p \mid \varphi$, the topmost constrained narrowing relation

$$p \mid \varphi \overset{\theta}{\rightsquigarrow}_{R/E_1 \cup B_1} q \mid \psi$$

modulo FVP $E_1 \cup B_1 \subseteq E \cup B$ holds iff there is a rule $l \to r \;\; \mathit{if} \;\; \phi$ in R

The Broader Picture: Narrowing with Constraints

Maude's `fvu-narrow` command allows us to symbolically model check invariants of infinite-state systems. But it has three limitations: (1) sets of states must be describable as disjunctions of unconstrained patterns; (2) the equations $E \cup B$ must be FVP; and (3) the rules R must be unconditional.

Restrictions (1)–(3) evaporate when we perform constrained narrowing. Given a constrained pattern $p \mid \varphi$ and a set R of conditional rewrite rules which, W.L.O.G., have disjoint variables from $p \mid \varphi$, the topmost constrained narrowing relation

$$p \mid \varphi \overset{\theta}{\rightsquigarrow}_{R/E_1 \cup B_1} q \mid \psi$$

modulo FVP $E_1 \cup B_1 \subseteq E \cup B$ holds iff there is a rule $l \to r \ \ if \ \ \phi$ in R and a $E_1 \cup B_1$-unifier θ of $l = p$ such that:

The Broader Picture: Narrowing with Constraints

Maude's `fvu-narrow` command allows us to symbolically model check invariants of infinite-state systems. But it has three limitations: (1) sets of states must be describable as disjunctions of unconstrained patterns; (2) the equations $E \cup B$ must be FVP; and (3) the rules R must be unconditional.

Restrictions (1)–(3) evaporate when we perform constrained narrowing. Given a constrained pattern $p \mid \varphi$ and a set R of conditional rewrite rules which, W.L.O.G., have disjoint variables from $p \mid \varphi$, the topmost constrained narrowing relation

$$p \mid \varphi \leadsto^{\theta}_{R/E_1 \cup B_1} q \mid \psi$$

modulo FVP $E_1 \cup B_1 \subseteq E \cup B$ holds iff there is a rule $l \to r \;\; \mathit{if} \;\; \phi$ in R and a $E_1 \cup B_1$-unifier θ of $l = p$ such that: (i) $q = (r\theta)$, and

The Broader Picture: Narrowing with Constraints

Maude's `fvu-narrow` command allows us to symbolically model check invariants of infinite-state systems. But it has three limitations: (1) sets of states must be describable as disjunctions of unconstrained patterns; (2) the equations $E \cup B$ must be FVP; and (3) the rules R must be unconditional.

Restrictions (1)–(3) evaporate when we perform constrained narrowing. Given a constrained pattern $p \mid \varphi$ and a set R of conditional rewrite rules which, W.L.O.G., have disjoint variables from $p \mid \varphi$, the topmost constrained narrowing relation

$$p \mid \varphi \overset{\theta}{\rightsquigarrow}_{R/E_1 \cup B_1} q \mid \psi$$

modulo FVP $E_1 \cup B_1 \subseteq E \cup B$ holds iff there is a rule $l \to r \;\; if \;\; \phi$ in R and a $E_1 \cup B_1$-unifier θ of $l = p$ such that: (i) $q = (r\theta)$, and (ii) $\psi = (\varphi \wedge \phi)\theta$.

The Broader Picture: Narrowing with Constraints (II)

With topmost constrained narrowing, restrictions (1)–(3) evaporate as follows:

The Broader Picture: Narrowing with Constraints (II)

With topmost constrained narrowing, restrictions (1)–(3) evaporate as follows:

1. Sets of initial states and complements of invariants can both often be expressed as disjunctions of constrained patterns.

The Broader Picture: Narrowing with Constraints (II)

With topmost constrained narrowing, restrictions (1)–(3) evaporate as follows:

1. Sets of initial states and complements of invariants can both often be expressed as disjunctions of constrained patterns.
2. In the topmost rewrite theory $\mathcal{R} = (\Sigma, E \cup B, R)$ satisfying executability requirements (1)–(4), the equations $E \cup B$ need not be FVP, but we assume an *FVP* subtheory inclusion $(\Sigma_1, E_1 \cup B_1) \subseteq (\Sigma, E \cup B)$ such that $\mathbb{C}_{\Sigma/E,B}|_{\Sigma_1} = \mathbb{C}_{\Sigma/E_1,B_1}$.

The Broader Picture: Narrowing with Constraints (II)

With topmost constrained narrowing, restrictions (1)–(3) evaporate as follows:

1. Sets of initial states and complements of invariants can both often be expressed as disjunctions of constrained patterns.
2. In the topmost rewrite theory $\mathcal{R} = (\Sigma, E \cup B, R)$ satisfying executability requirements (1)–(4), the equations $E \cup B$ need not be FVP, but we assume an *FVP* subtheory inclusion $(\Sigma_1, E_1 \cup B_1) \subseteq (\Sigma, E \cup B)$ such that $\mathbb{C}_{\Sigma/E,B}|_{\Sigma_1} = \mathbb{C}_{\Sigma/E_1,B_1}$.
3. The rules $l \rightarrow r \;\; \mathit{if} \;\; \psi$ in R may be conditional, but we assume that l, r are Σ_1-terms.

The Broader Picture: Narrowing with Constraints (II)

With topmost constrained narrowing, restrictions (1)–(3) evaporate as follows:

1. Sets of initial states and complements of invariants can both often be expressed as disjunctions of constrained patterns.
2. In the topmost rewrite theory $\mathcal{R} = (\Sigma, E \cup B, R)$ satisfying executability requirements (1)–(4), the equations $E \cup B$ need not be FVP, but we assume an *FVP* subtheory inclusion $(\Sigma_1, E_1 \cup B_1) \subseteq (\Sigma, E \cup B)$ such that $\mathbb{C}_{\Sigma/E,B}|_{\Sigma_1} = \mathbb{C}_{\Sigma/E_1,B_1}$.
3. The rules $l \to r \;\; if \;\; \psi$ in R may be conditional, but we assume that l, r are Σ_1-terms.

The key result is that the **Lifting Lemma** generalizes to the constrained narrowing case.

The Broader Picture: Narrowing with Constraints (II)

With topmost constrained narrowing, restrictions (1)–(3) evaporate as follows:

1. Sets of initial states and complements of invariants can both often be expressed as disjunctions of constrained patterns.
2. In the topmost rewrite theory $\mathcal{R} = (\Sigma, E \cup B, R)$ satisfying executability requirements (1)–(4), the equations $E \cup B$ need not be FVP, but we assume an *FVP* subtheory inclusion $(\Sigma_1, E_1 \cup B_1) \subseteq (\Sigma, E \cup B)$ such that $\mathbb{C}_{\Sigma/E,B}|_{\Sigma_1} = \mathbb{C}_{\Sigma/E_1,B_1}$.
3. The rules $l \to r \;\; if \;\; \psi$ in R may be conditional, but we assume that l, r are Σ_1-terms.

The key result is that the **Lifting Lemma** generalizes to the constrained narrowing case. This supports symbolic model checking with constraints verification of invariants,

The Broader Picture: Narrowing with Constraints (II)

With topmost constrained narrowing, restrictions (1)–(3) evaporate as follows:

1. Sets of initial states and complements of invariants can both often be expressed as disjunctions of constrained patterns.
2. In the topmost rewrite theory $\mathcal{R} = (\Sigma, E \cup B, R)$ satisfying executability requirements (1)–(4), the equations $E \cup B$ need not be FVP, but we assume an *FVP* subtheory inclusion $(\Sigma_1, E_1 \cup B_1) \subseteq (\Sigma, E \cup B)$ such that $\mathbb{C}_{\Sigma/E,B}|_{\Sigma_1} = \mathbb{C}_{\Sigma/E_1,B_1}$.
3. The rules $l \to r$ *if* ψ in R may be conditional, but we assume that l, r are Σ_1-terms.

The key result is that the **Lifting Lemma** generalizes to the constrained narrowing case. This supports symbolic model checking with constraints verification of invariants, including folding.

Symbolic Model Checking vs. Inductive Theorem Proving

Folding is a powerful state space reduction technique to make the symbolic search space finite.

Symbolic Model Checking vs. Inductive Theorem Proving

Folding is a powerful state space reduction technique to make the symbolic search space finite. But in the presence of constraints it needs inductive theorem proving.

Symbolic Model Checking vs. Inductive Theorem Proving

Folding is a powerful state space reduction technique to make the symbolic search space finite. But in the presence of constraints it needs inductive theorem proving. We say that $p \mid \varphi$ folds into the (more general) $q \mid \psi$ iff

Symbolic Model Checking vs. Inductive Theorem Proving

Folding is a powerful state space reduction technique to make the symbolic search space finite. But in the presence of constraints it needs inductive theorem proving. We say that $p \mid \varphi$ folds into the (more general) $q \mid \psi$ iff there is a substitution θ such that:

Symbolic Model Checking vs. Inductive Theorem Proving

Folding is a powerful state space reduction technique to make the symbolic search space finite. But in the presence of constraints it needs inductive theorem proving. We say that $p \mid \varphi$ folds into the (more general) $q \mid \psi$ iff there is a substitution θ such that: (i) $q\theta =_B p$, and (ii)

Symbolic Model Checking vs. Inductive Theorem Proving

Folding is a powerful state space reduction technique to make the symbolic search space finite. But in the presence of constraints it needs inductive theorem proving. We say that $p \mid \varphi$ folds into the (more general) $q \mid \psi$ iff there is a substitution θ such that: (i) $q\theta =_B p$, and (ii) $\mathbb{C}_{\Sigma/E,B} \models \varphi \Rightarrow (\psi\theta)$.

Symbolic Model Checking vs. Inductive Theorem Proving

Folding is a powerful state space reduction technique to make the symbolic search space finite. But in the presence of constraints it needs inductive theorem proving. We say that $p \mid \varphi$ folds into the (more general) $q \mid \psi$ iff there is a substitution θ such that: (i) $q\theta =_B p$, and (ii) $\mathbb{C}_{\Sigma/E,B} \models \varphi \Rightarrow (\psi\theta)$. But (ii) means that $\varphi \Rightarrow (\psi\theta)$ as an inductive theorem,

Symbolic Model Checking vs. Inductive Theorem Proving

Folding is a powerful state space reduction technique to make the symbolic search space finite. But in the presence of constraints it needs inductive theorem proving. We say that $p \mid \varphi$ folds into the (more general) $q \mid \psi$ iff there is a substitution θ such that: (i) $q\theta =_B p$, and (ii) $\mathbb{C}_{\Sigma/E,B} \models \varphi \Rightarrow (\psi\theta)$. But (ii) means that $\varphi \Rightarrow (\psi\theta)$ as an inductive theorem, which has to be proved.

Symbolic Model Checking vs. Inductive Theorem Proving

Folding is a powerful state space reduction technique to make the symbolic search space finite. But in the presence of constraints it needs inductive theorem proving. We say that $p \mid \varphi$ folds into the (more general) $q \mid \psi$ iff there is a substitution θ such that: (i) $q\theta =_B p$, and (ii) $\mathbb{C}_{\Sigma/E,B} \models \varphi \Rightarrow (\psi\theta)$. But (ii) means that $\varphi \Rightarrow (\psi\theta)$ as an inductive theorem, which has to be proved.

Likewise, checking that not bad states $q \mid \psi$ in the target are shared with those in $p \mid \varphi$ cannot be settled by failure of unification:

Symbolic Model Checking vs. Inductive Theorem Proving

Folding is a powerful state space reduction technique to make the symbolic search space finite. But in the presence of constraints it needs inductive theorem proving. We say that $p \mid \varphi$ folds into the (more general) $q \mid \psi$ iff there is a substitution θ such that: (i) $q\theta =_B p$, and (ii) $\mathbb{C}_{\Sigma/E,B} \models \varphi \Rightarrow (\psi\theta)$. But (ii) means that $\varphi \Rightarrow (\psi\theta)$ as an inductive theorem, which has to be proved.

Likewise, checking that not bad states $q \mid \psi$ in the target are shared with those in $p \mid \varphi$ cannot be settled by failure of unification: a $E_1 \cup B_1$-unifier θ of $p = q$ will yield an empty intersection $(p \mid \varphi \wedge \psi)\theta$ iff

Symbolic Model Checking vs. Inductive Theorem Proving

Folding is a powerful state space reduction technique to make the symbolic search space finite. But in the presence of constraints it needs inductive theorem proving. We say that $p \mid \varphi$ folds into the (more general) $q \mid \psi$ iff there is a substitution θ such that: (i) $q\theta =_B p$, and (ii) $\mathbb{C}_{\Sigma/E,B} \models \varphi \Rightarrow (\psi\theta)$. But (ii) means that $\varphi \Rightarrow (\psi\theta)$ as an inductive theorem, which has to be proved.

Likewise, checking that not bad states $q \mid \psi$ in the target are shared with those in $p \mid \varphi$ cannot be settled by failure of unification: a $E_1 \cup B_1$-unifier θ of $p = q$ will yield an empty intersection $(p \mid \varphi \wedge \psi)\theta$ iff $\mathbb{C}_{\Sigma/E,B} \models \neg(\varphi\theta) \vee \neg(\psi\theta)$,

Symbolic Model Checking vs. Inductive Theorem Proving

Folding is a powerful state space reduction technique to make the symbolic search space finite. But in the presence of constraints it needs inductive theorem proving. We say that $p \mid \varphi$ folds into the (more general) $q \mid \psi$ iff there is a substitution θ such that: (i) $q\theta =_B p$, and (ii) $\mathbb{C}_{\Sigma/E,B} \models \varphi \Rightarrow (\psi\theta)$. But (ii) means that $\varphi \Rightarrow (\psi\theta)$ as an inductive theorem, which has to be proved.

Likewise, checking that not bad states $q \mid \psi$ in the target are shared with those in $p \mid \varphi$ cannot be settled by failure of unification: a $E_1 \cup B_1$-unifier θ of $p = q$ will yield an empty intersection $(p \mid \varphi \wedge \psi)\theta$ iff $\mathbb{C}_{\Sigma/E,B} \models \neg(\varphi\theta) \vee \neg(\psi\theta)$, which, again, has to be proved as an inductive theorem.

Symbolic Model Checking vs. Inductive Theorem Proving

Folding is a powerful state space reduction technique to make the symbolic search space finite. But in the presence of constraints it needs inductive theorem proving. We say that $p \mid \varphi$ folds into the (more general) $q \mid \psi$ iff there is a substitution θ such that: (i) $q\theta =_B p$, and (ii) $\mathbb{C}_{\Sigma/E,B} \models \varphi \Rightarrow (\psi\theta)$. But (ii) means that $\varphi \Rightarrow (\psi\theta)$ as an inductive theorem, which has to be proved.

Likewise, checking that not bad states $q \mid \psi$ in the target are shared with those in $p \mid \varphi$ cannot be settled by failure of unification: a $E_1 \cup B_1$-unifier θ of $p = q$ will yield an empty intersection $(p \mid \varphi \wedge \psi)\theta$ iff $\mathbb{C}_{\Sigma/E,B} \models \neg(\varphi\theta) \vee \neg(\psi\theta)$, which, again, has to be proved as an inductive theorem.

Is this model checking?

Symbolic Model Checking vs. Inductive Theorem Proving

Folding is a powerful state space reduction technique to make the symbolic search space finite. But in the presence of constraints it needs inductive theorem proving. We say that $p \mid \varphi$ folds into the (more general) $q \mid \psi$ iff there is a substitution θ such that: (i) $q\theta =_B p$, and (ii) $\mathbb{C}_{\Sigma/E,B} \models \varphi \Rightarrow (\psi\theta)$. But (ii) means that $\varphi \Rightarrow (\psi\theta)$ as an inductive theorem, which has to be proved.

Likewise, checking that not bad states $q \mid \psi$ in the target are shared with those in $p \mid \varphi$ cannot be settled by failure of unification: a $E_1 \cup B_1$-unifier θ of $p = q$ will yield an empty intersection $(p \mid \varphi \wedge \psi)\theta$ iff $\mathbb{C}_{\Sigma/E,B} \models \neg(\varphi\theta) \vee \neg(\psi\theta)$, which, again, has to be proved as an inductive theorem.

Is this model checking? Is it theorem proving?

Symbolic Model Checking vs. Inductive Theorem Proving

Folding is a powerful state space reduction technique to make the symbolic search space finite. But in the presence of constraints it needs inductive theorem proving. We say that $p \mid \varphi$ folds into the (more general) $q \mid \psi$ iff there is a substitution θ such that: (i) $q\theta =_B p$, and (ii) $\mathbb{C}_{\Sigma/E,B} \models \varphi \Rightarrow (\psi\theta)$. But (ii) means that $\varphi \Rightarrow (\psi\theta)$ as an inductive theorem, which has to be proved.

Likewise, checking that not bad states $q \mid \psi$ in the target are shared with those in $p \mid \varphi$ cannot be settled by failure of unification: a $E_1 \cup B_1$-unifier θ of $p = q$ will yield an empty intersection $(p \mid \varphi \wedge \psi)\theta$ iff $\mathbb{C}_{\Sigma/E,B} \models \neg(\varphi\theta) \vee \neg(\psi\theta)$, which, again, has to be proved as an inductive theorem.

Is this model checking? Is it theorem proving? It is both!

A Two-Way Street

The synergy between symbolic model checking and inductive theorem proving is a two-way street:

A Two-Way Street

The synergy between symbolic model checking and inductive theorem proving is a two-way street: Not only does model checking become more powerful;

A Two-Way Street

The synergy between symbolic model checking and inductive theorem proving is a two-way street: Not only does model checking become more powerful; theorem proving does so too.

A Two-Way Street

The synergy between symbolic model checking and inductive theorem proving is a two-way street: Not only does model checking become more powerful; theorem proving does so too. For example, finding and proving inductive invariants can require significant theorem proving effort.

A Two-Way Street

The synergy between symbolic model checking and inductive theorem proving is a two-way street: Not only does model checking become more powerful; theorem proving does so too. For example, finding and proving inductive invariants can require significant theorem proving effort. I will show in what follows how finding and proving them with the combined power of symbolic model checking and inductive theorem proving can make it easier.

A Two-Way Street

The synergy between symbolic model checking and inductive theorem proving is a two-way street: Not only does model checking become more powerful; theorem proving does so too. For example, finding and proving inductive invariants can require significant theorem proving effort. I will show in what follows how finding and proving them with the combined power of symbolic model checking and inductive theorem proving can make it easier.

An invariant $Q \subseteq \mathbb{C}_{\Sigma/\vec{E},B,St}$ is called inductive iff it is transition closed.

A Two-Way Street

The synergy between symbolic model checking and inductive theorem proving is a two-way street: Not only does model checking become more powerful; theorem proving does so too. For example, finding and proving inductive invariants can require significant theorem proving effort. I will show in what follows how finding and proving them with the combined power of symbolic model checking and inductive theorem proving can make it easier.

An invariant $Q \subseteq \mathbb{C}_{\Sigma/\vec{E},B,St}$ is called inductive iff it is transition closed. I.e., for each $[u] \in Q$ and each transition $[u] \rightarrow_{\mathcal{C}_{\mathcal{R}}} [v]$ we must have $[v] \in Q$.

A Two-Way Street

The synergy between symbolic model checking and inductive theorem proving is a two-way street: Not only does model checking become more powerful; theorem proving does so too. For example, finding and proving inductive invariants can require significant theorem proving effort. I will show in what follows how finding and proving them with the combined power of symbolic model checking and inductive theorem proving can make it easier.

An invariant $Q \subseteq \mathbb{C}_{\Sigma/\vec{E},B,St}$ is called inductive iff it is transition closed. I.e., for each $[u] \in Q$ and each transition $[u] \rightarrow_{\mathcal{C}_{\mathcal{R}}} [v]$ we must have $[v] \in Q$. This has two useful consequences:

A Two-Way Street

The synergy between symbolic model checking and inductive theorem proving is a two-way street: Not only does model checking become more powerful; theorem proving does so too. For example, finding and proving inductive invariants can require significant theorem proving effort. I will show in what follows how finding and proving them with the combined power of symbolic model checking and inductive theorem proving can make it easier.

An invariant $Q \subseteq \mathbb{C}_{\Sigma/\vec{E},B,St}$ is called inductive iff it is transition closed. I.e., for each $[u] \in Q$ and each transition $[u] \rightarrow_{\mathcal{C}_{\mathcal{R}}} [v]$ we must have $[v] \in Q$. This has two useful consequences: (1) If for an invariant guess Q_0 describable as a disjunction of constrained patterns we obtain a finite folding graph disjoint from its negation, then we have found and proved that the disjunction of patterns in such a graph is an inductive invariant.

A Two-Way Street

The synergy between symbolic model checking and inductive theorem proving is a two-way street: Not only does model checking become more powerful; theorem proving does so too. For example, finding and proving inductive invariants can require significant theorem proving effort. I will show in what follows how finding and proving them with the combined power of symbolic model checking and inductive theorem proving can make it easier.

An invariant $Q \subseteq \mathbb{C}_{\Sigma/\vec{E},B,St}$ is called inductive iff it is transition closed. I.e., for each $[u] \in Q$ and each transition $[u] \rightarrow_{\mathcal{C}_{\mathcal{R}}} [v]$ we must have $[v] \in Q$. This has two useful consequences: (1) If for an invariant guess Q_0 describable as a disjunction of constrained patterns we obtain a finite folding graph disjoint from its negation, then we have found and proved that the disjunction of patterns in such a graph is an inductive invariant. (2) If we can also show that Q_0 folds into itself, then Q_0 is also an inductive invariant.

Proving Invariants and Inductive Invariants

We can summarize the following methods to prove inductive invariants, and, after doing so, proving also other invariants.

Proving Invariants and Inductive Invariants

We can summarize the following methods to prove inductive invariants, and, after doing so, proving also other invariants.

(1). **Initial States Contained**. Suppose that we want to prove that $\bigvee_{j \in J} v_j \mid \psi_j$ is an inductive invariant from initial states $\bigvee_{i \in I} u_i \mid \varphi_i$.

Proving Invariants and Inductive Invariants

We can summarize the following methods to prove inductive invariants, and, after doing so, proving also other invariants.

(1). **Initial States Contained**. Suppose that we want to prove that $\bigvee_{j\in J} v_j \mid \psi_j$ is an inductive invariant from initial states $\bigvee_{i\in I} u_i \mid \varphi_i$. We first need to show the set containment $[\![\bigvee_{i\in I} u_i \mid \varphi_i]\!]_{\vec{E}/B} \subseteq [\![\bigvee_{j\in J} v_j \mid \psi_j]\!]_{\vec{E}/B}$.

Proving Invariants and Inductive Invariants

We can summarize the following methods to prove inductive invariants, and, after doing so, proving also other invariants.

(1). **Initial States Contained**. Suppose that we want to prove that $\bigvee_{j \in J} v_j \mid \psi_j$ is an inductive invariant from initial states $\bigvee_{i \in I} u_i \mid \varphi_i$. We first need to show the set containment $[\![\bigvee_{i \in I} u_i \mid \varphi_i]\!]_{\vec{E}/B} \subseteq [\![\bigvee_{j \in J} v_j \mid \psi_j]\!]_{\vec{E}/B}$. A sufficient condition for this contaiment is to show that for each $i \in I$ there is a $j \in J$ such that $(\subseteq_{i,j})$ $[\![u_i \mid \varphi_i]\!]_{\vec{E}/B} \subseteq [\![v_j \mid \psi_j]\!]_{\vec{E}/B}$.

Proving Invariants and Inductive Invariants

We can summarize the following methods to prove inductive invariants, and, after doing so, proving also other invariants.

(1). **Initial States Contained**. Suppose that we want to prove that $\bigvee_{j \in J} v_j \mid \psi_j$ is an inductive invariant from initial states $\bigvee_{i \in I} u_i \mid \varphi_i$. We first need to show the set containment $[\![\bigvee_{i \in I} u_i \mid \varphi_i]\!]_{\vec{E}/B} \subseteq [\![\bigvee_{j \in J} v_j \mid \psi_j]\!]_{\vec{E}/B}$. A sufficient condition for this contaiment is to show that for each $i \in I$ there is a $j \in J$ such that $(\subseteq_{i,j})$ $[\![u_i \mid \varphi_i]\!]_{\vec{E}/B} \subseteq [\![v_j \mid \psi_j]\!]_{\vec{E}/B}$. To prove $(\subseteq_{i,j})$ it is in turn enough to show that $u_i \mid \varphi_i \sqsubseteq_{B_1} v_j \mid \psi_j$,

Proving Invariants and Inductive Invariants

We can summarize the following methods to prove inductive invariants, and, after doing so, proving also other invariants.

(1). **Initial States Contained**. Suppose that we want to prove that $\bigvee_{j\in J} v_j \mid \psi_j$ is an inductive invariant from initial states $\bigvee_{i\in I} u_i \mid \varphi_i$. We first need to show the set containment $[\![\bigvee_{i\in I} u_i \mid \varphi_i]\!]_{\vec{E}/B} \subseteq [\![\bigvee_{j\in J} v_j \mid \psi_j]\!]_{\vec{E}/B}$. A sufficient condition for this contaiment is to show that for each $i \in I$ there is a $j \in J$ such that $(\subseteq_{i,j})$ $[\![u_i \mid \varphi_i]\!]_{\vec{E}/B} \subseteq [\![v_j \mid \psi_j]\!]_{\vec{E}/B}$. To prove $(\subseteq_{i,j})$ it is in turn enough to show that $u_i \mid \varphi_i \sqsubseteq_{B_1} v_j \mid \psi_j$, which by definition means:

Proving Invariants and Inductive Invariants

We can summarize the following methods to prove inductive invariants, and, after doing so, proving also other invariants.

(1). **Initial States Contained**. Suppose that we want to prove that $\bigvee_{j \in J} v_j \mid \psi_j$ is an inductive invariant from initial states $\bigvee_{i \in I} u_i \mid \varphi_i$. We first need to show the set containment $[\![\bigvee_{i \in I} u_i \mid \varphi_i]\!]_{\vec{E}/B} \subseteq [\![\bigvee_{j \in J} v_j \mid \psi_j]\!]_{\vec{E}/B}$. A sufficient condition for this containment is to show that for each $i \in I$ there is a $j \in J$ such that $(\subseteq_{i,j})$ $[\![u_i \mid \varphi_i]\!]_{\vec{E}/B} \subseteq [\![v_j \mid \psi_j]\!]_{\vec{E}/B}$. To prove $(\subseteq_{i,j})$ it is in turn enough to show that $u_i \mid \varphi_i \sqsubseteq_{B_1} v_j \mid \psi_j$, which by definition means:

$$\exists \alpha \;\; s.t. \;\; u_i =_{B_1} v_j\alpha \;\; \wedge \;\; \mathbb{C}_{\Sigma/\vec{E},B} \models \varphi_i \Rightarrow (\psi_j\alpha)$$

Proving Invariants and Inductive Invariants

We can summarize the following methods to prove inductive invariants, and, after doing so, proving also other invariants.

(1). **Initial States Contained**. Suppose that we want to prove that $\bigvee_{j \in J} v_j \mid \psi_j$ is an inductive invariant from initial states $\bigvee_{i \in I} u_i \mid \varphi_i$. We first need to show the set containment $[\![\bigvee_{i \in I} u_i \mid \varphi_i]\!]_{\vec{E}/B} \subseteq [\![\bigvee_{j \in J} v_j \mid \psi_j]\!]_{\vec{E}/B}$. A sufficient condition for this contaiment is to show that for each $i \in I$ there is a $j \in J$ such that $(\subseteq_{i,j})$ $[\![u_i \mid \varphi_i]\!]_{\vec{E}/B} \subseteq [\![v_j \mid \psi_j]\!]_{\vec{E}/B}$. To prove $(\subseteq_{i,j})$ it is in turn enough to show that $u_i \mid \varphi_i \sqsubseteq_{B_1} v_j \mid \psi_j$, which by definition means:

$$\exists \alpha \ \ s.t. \ \ u_i =_{B_1} v_j\alpha \ \ \wedge \ \ \mathbb{C}_{\Sigma/\vec{E},B} \models \varphi_i \Rightarrow (\psi_j\alpha)$$

If $\psi_j \equiv \top$, this is decidable by finding a B_1-matching substitution α. Otherwise, $\varphi_i \Rightarrow (\psi_j\alpha)$ is an inductive theorem that has to be proved.

Proving Invariants and Inductive Invariants

We can summarize the following methods to prove inductive invariants, and, after doing so, proving also other invariants.

(1). **Initial States Contained**. Suppose that we want to prove that $\bigvee_{j \in J} v_j \mid \psi_j$ is an inductive invariant from initial states $\bigvee_{i \in I} u_i \mid \varphi_i$. We first need to show the set containment $[\![\bigvee_{i \in I} u_i \mid \varphi_i]\!]_{\vec{E}/B} \subseteq [\![\bigvee_{j \in J} v_j \mid \psi_j]\!]_{\vec{E}/B}$. A sufficient condition for this containment is to show that for each $i \in I$ there is a $j \in J$ such that $(\subseteq_{i,j})$ $[\![u_i \mid \varphi_i]\!]_{\vec{E}/B} \subseteq [\![v_j \mid \psi_j]\!]_{\vec{E}/B}$. To prove $(\subseteq_{i,j})$ it is in turn enough to show that $u_i \mid \varphi_i \sqsubseteq_{B_1} v_j \mid \psi_j$, which by definition means:

$$\exists \alpha \;\; s.t. \;\; u_i =_{B_1} v_j\alpha \;\; \wedge \;\; \mathbb{C}_{\Sigma/\vec{E},B} \models \varphi_i \Rightarrow (\psi_j\alpha)$$

If $\psi_j \equiv \top$, this is decidable by finding a B_1-matching substitution α. Otherwise, $\varphi_i \Rightarrow (\psi_j\alpha)$ is an inductive theorem that has to be proved. **Note**. Could replace $\sqsubseteq_{B_1}$ by the more general $\sqsubseteq_{E_1 \cup B_1}$.

Proving Invariants and Inductive Invariants (II)

(2). **Proving the Inductive Invariant**. To prove that $\bigvee_{j \in J} v_j \mid \psi_j$ is an inductive invariant, we need to prove that the set of ground states $[\![\bigvee_{j \in J} v_j \mid \psi_j]\!]_{\vec{E}/B}$ is transition closed.

Proving Invariants and Inductive Invariants (II)

(2). **Proving the Inductive Invariant**. To prove that $\bigvee_{j \in J} v_j \mid \psi_j$ is an inductive invariant, we need to prove that the set of ground states $[\![\bigvee_{j \in J} v_j \mid \psi_j]\!]_{\vec{E}/B}$ is transition closed. But by the **Lifting Lemma** this is equivalent to showing that:

Proving Invariants and Inductive Invariants (II)

(2). **Proving the Inductive Invariant**. To prove that $\bigvee_{j \in J} v_j \mid \psi_j$ is an inductive invariant, we need to prove that the set of ground states $[\![\bigvee_{j \in J} v_j \mid \psi_j]\!]_{\vec{E}/B}$ is transition closed. But by the **Lifting Lemma** this is equivalent to showing that:

$$\forall j \in J,\ \forall (l \to r \ \mathit{if}\ \phi) \in R,\ \forall \gamma \in \mathit{Unif}_{E_1 \cup B_1}(v_j, l)\ [\![(r \mid \psi_j \wedge \pi)\gamma]\!]_{\vec{E}/B} \subseteq [\![\bigvee_{j \in J} v_j \mid \psi_j]\!]_{\vec{E}/B}.$$

Proving Invariants and Inductive Invariants (II)

(2). **Proving the Inductive Invariant**. To prove that $\bigvee_{j \in J} v_j \mid \psi_j$ is an inductive invariant, we need to prove that the set of ground states $[\![\bigvee_{j \in J} v_j \mid \psi_j]\!]_{\vec{E}/B}$ is transition closed. But by the **Lifting Lemma** this is equivalent to showing that:

$$\forall j \in J,\ \forall (l \to r \ if\ \phi) \in R,\ \forall \gamma \in Unif_{E_1 \cup B_1}(v_j, l)\ [\![(r \mid \psi_j \wedge \pi)\gamma]\!]_{\vec{E}/B} \subseteq [\![\bigvee_{j \in J} v_j \mid \psi_j]\!]_{\vec{E}/B}.$$

That is, we need to show that the ground instances of each child by a narrowing step $v_j \mid \psi_j \rightsquigarrow_{R/E_1 \cup B_1} (r \mid \psi_j \wedge \phi)\gamma$ are contained in (are folded into) the conjectured invariant.

Proving Invariants and Inductive Invariants (II)

(2). **Proving the Inductive Invariant**. To prove that $\bigvee_{j \in J} v_j \mid \psi_j$ is an inductive invariant, we need to prove that the set of ground states $[\![\bigvee_{j \in J} v_j \mid \psi_j]\!]_{\vec{E}/B}$ is transition closed. But by the **Lifting Lemma** this is equivalent to showing that:

$$\forall j \in J,\ \forall (l \to r \textit{ if } \phi) \in R,\ \forall \gamma \in \mathit{Unif}_{E_1 \cup B_1}(v_j, l)\ [\![(r \mid \psi_j \wedge \pi)\gamma]\!]_{\vec{E}/B} \subseteq [\![\bigvee_{j \in J} v_j \mid \psi_j]\!]_{\vec{E}/B}.$$

That is, we need to show that the ground instances of each child by a narrowing step $v_j \mid \psi_j \rightsquigarrow_{R/E_1 \cup B_1} (r \mid \psi_j \wedge \phi)\gamma$ are contained in (are folded into) the conjectured invariant. For this it is again a sufficient condition to prove that there exists a $j' \in J$ s.t. $(r \mid \psi_j \wedge \phi)\gamma \sqsubseteq_{B_1} v_{j'} \mid \psi_{j'}$.

Proving Invariants and Inductive Invariants (II)

(2). **Proving the Inductive Invariant**. To prove that $\bigvee_{j\in J} v_j \mid \psi_j$ is an inductive invariant, we need to prove that the set of ground states $[\![\bigvee_{j\in J} v_j \mid \psi_j]\!]_{\vec{E}/B}$ is transition closed. But by the **Lifting Lemma** this is equivalent to showing that:

$$\forall j \in J,\ \forall (l \to r \text{ if } \phi) \in R,\ \forall \gamma \in Unif_{E_1 \cup B_1}(v_j, l)\ [\![(r \mid \psi_j \wedge \pi)\gamma]\!]_{\vec{E}/B} \subseteq [\![\bigvee_{j\in J} v_j \mid \psi_j]\!]_{\vec{E}/B}.$$

That is, we need to show that the ground instances of each child by a narrowing step $v_j \mid \psi_j \rightsquigarrow_{R/E_1 \cup B_1} (r \mid \psi_j \wedge \phi)\gamma$ are contained in (are folded into) the conjectured invariant. For this it is again a sufficient condition to prove that there exists a $j' \in J$ s.t. $(r \mid \psi_j \wedge \phi)\gamma \sqsubseteq_{B_1} v_{j'} \mid \psi_{j'}$. **Note**. Could replace $\sqsubseteq_{B_1}$ by $\sqsubseteq_{E_1 \cup B_1}$.

Proving Invariants and Inductive Invariants (II)

(2). **Proving the Inductive Invariant**. To prove that $\bigvee_{j \in J} v_j \mid \psi_j$ is an inductive invariant, we need to prove that the set of ground states $[\![\bigvee_{j \in J} v_j \mid \psi_j]\!]_{\vec{E}/B}$ is transition closed. But by the **Lifting Lemma** this is equivalent to showing that:

$$\forall j \in J,\ \forall (l \to r \ \mathit{if}\ \phi) \in R,\ \forall \gamma \in \mathit{Unif}_{E_1 \cup B_1}(v_j, l)\ [\![(r \mid \psi_j \wedge \pi)\gamma]\!]_{\vec{E}/B} \subseteq [\![\bigvee_{j \in J} v_j \mid \psi_j]\!]_{\vec{E}/B}.$$

That is, we need to show that the ground instances of each child by a narrowing step $v_j \mid \psi_j \leadsto_{R/E_1 \cup B_1} (r \mid \psi_j \wedge \phi)\gamma$ are contained in (are folded into) the conjectured invariant. For this it is again a sufficient condition to prove that there exists a $j' \in J$ s.t. $(r \mid \psi_j \wedge \phi)\gamma \sqsubseteq_{B_1} v_{j'} \mid \psi_{j'}$. **Note**. Could replace $\sqsubseteq_{B_1}$ by $\sqsubseteq_{E_1 \cup B_1}$.

Again, proving that for a B_1-matching substitution α $\mathbb{C}_{\Sigma/\vec{E},B} \models (\psi_j \wedge \phi)\gamma \Rightarrow (\psi_{j'}\alpha)$ requires inductive theorem proving.

Proving Invariants and Inductive Invariants (III)

(3). **Proving Other Invariant**. Once we have proved that $\bigvee_{j \in J} v_j \mid \psi_j$ is an inductive invariant from $\bigvee_{i \in I} u_i \mid \varphi_i$, we can prove another invariant Q from $\bigvee_{i \in I} u_i \mid \varphi_i$ in one of two ways:

Proving Invariants and Inductive Invariants (III)

(3). **Proving Other Invariant**. Once we have proved that $\bigvee_{j \in J} v_j \mid \psi_j$ is an inductive invariant from $\bigvee_{i \in I} u_i \mid \varphi_i$, we can prove another invariant Q from $\bigvee_{i \in I} u_i \mid \varphi_i$ in one of two ways:

1. **Positively**: If $Q = [\![\bigvee_{k \in K} w_k \mid \phi_k]\!]_{\vec{E}/B}$ it is enough to show that

Proving Invariants and Inductive Invariants (III)

(3). **Proving Other Invariant**. Once we have proved that $\bigvee_{j \in J} v_j \mid \psi_j$ is an inductive invariant from $\bigvee_{i \in I} u_i \mid \varphi_i$, we can prove another invariant Q from $\bigvee_{i \in I} u_i \mid \varphi_i$ in one of two ways:

1. **Positively**: If $Q = [\![\bigvee_{k \in K} w_k \mid \phi_k]\!]_{\vec{E}/B}$ it is enough to show that $\forall j \in J\, \exists k \in K$ s.t. $[\![\bigvee_{j \in J} v_j \mid \psi_j]\!]_{\vec{E}/B} \subseteq [\![\bigvee_{k \in K} w_k \mid \phi_k]\!]_{\vec{E}/B}$,

Proving Invariants and Inductive Invariants (III)

(3). **Proving Other Invariant**. Once we have proved that $\bigvee_{j \in J} v_j \mid \psi_j$ is an inductive invariant from $\bigvee_{i \in I} u_i \mid \varphi_i$, we can prove another invariant Q from $\bigvee_{i \in I} u_i \mid \varphi_i$ in one of two ways:

1. **Positively**: If $Q = [\![\bigvee_{k \in K} w_k \mid \phi_k]\!]_{\vec{E}/B}$ it is enough to show that $\forall j \in J\, \exists k \in K$ s.t. $[\![\bigvee_{j \in J} v_j \mid \psi_j]\!]_{\vec{E}/B} \subseteq [\![\bigvee_{k \in K} w_k \mid \phi_k]\!]_{\vec{E}/B}$, which holds if we can prove

Proving Invariants and Inductive Invariants (III)

(3). **Proving Other Invariant**. Once we have proved that $\bigvee_{j \in J} v_j \mid \psi_j$ is an inductive invariant from $\bigvee_{i \in I} u_i \mid \varphi_i$, we can prove another invariant Q from $\bigvee_{i \in I} u_i \mid \varphi_i$ in one of two ways:

1. **Positively**: If $Q = [\![\bigvee_{k \in K} w_k \mid \phi_k]\!]_{\vec{E}/B}$ it is enough to show that $\forall j \in J \, \exists k \in K$ s.t. $[\![\bigvee_{j \in J} v_j \mid \psi_j]\!]_{\vec{E}/B} \subseteq [\![\bigvee_{k \in K} w_k \mid \phi_k]\!]_{\vec{E}/B}$, which holds if we can prove $v_i \mid \psi_j \sqsubseteq_{B_1} w_k \mid \phi_k$.

Proving Invariants and Inductive Invariants (III)

(3). **Proving Other Invariant**. Once we have proved that $\bigvee_{j\in J} v_j \mid \psi_j$ is an inductive invariant from $\bigvee_{i\in I} u_i \mid \varphi_i$, we can prove another invariant Q from $\bigvee_{i\in I} u_i \mid \varphi_i$ in one of two ways:

1. **Positively**: If $Q = [\![\bigvee_{k\in K} w_k \mid \phi_k]\!]_{\vec{E}/B}$ it is enough to show that $\forall j \in J\, \exists k \in K$ s.t. $[\![\bigvee_{j\in J} v_j \mid \psi_j]\!]_{\vec{E}/B} \subseteq [\![\bigvee_{k\in K} w_k \mid \phi_k]\!]_{\vec{E}/B}$, which holds if we can prove $v_i \mid \psi_j \sqsubseteq_{B_1} w_k \mid \phi_k$.
2. **Negatively**: If $Q^c = [\![\bigvee_{k\in K} w_k \mid \phi_k]\!]_{\vec{E}/B}$,

Proving Invariants and Inductive Invariants (III)

(3). **Proving Other Invariant**. Once we have proved that $\bigvee_{j \in J} v_j \mid \psi_j$ is an inductive invariant from $\bigvee_{i \in I} u_i \mid \varphi_i$, we can prove another invariant Q from $\bigvee_{i \in I} u_i \mid \varphi_i$ in one of two ways:

1. **Positively**: If $Q = [\![\bigvee_{k \in K} w_k \mid \phi_k]\!]_{\vec{E}/B}$ it is enough to show that $\forall j \in J \, \exists k \in K$ s.t. $[\![\bigvee_{j \in J} v_j \mid \psi_j]\!]_{\vec{E}/B} \subseteq [\![\bigvee_{k \in K} w_k \mid \phi_k]\!]_{\vec{E}/B}$, which holds if we can prove $v_i \mid \psi_j \sqsubseteq_{B_1} w_k \mid \phi_k$.
2. **Negatively**: If $Q^c = [\![\bigvee_{k \in K} w_k \mid \phi_k]\!]_{\vec{E}/B}$, then if $[\![\bigvee_{j \in J} v_j \mid \psi_j]\!]_{\vec{E}/B} \cap \subseteq [\![\bigvee_{k \in K} w_k \mid \phi_k]\!]_{\vec{E}/B} = \emptyset$

Proving Invariants and Inductive Invariants (III)

(3). **Proving Other Invariant**. Once we have proved that $\bigvee_{j \in J} v_j \mid \psi_j$ is an inductive invariant from $\bigvee_{i \in I} u_i \mid \varphi_i$, we can prove another invariant Q from $\bigvee_{i \in I} u_i \mid \varphi_i$ in one of two ways:

1. **Positively**: If $Q = [\![\bigvee_{k \in K} w_k \mid \phi_k]\!]_{\vec{E}/B}$ it is enough to show that $\forall j \in J \, \exists k \in K$ s.t. $[\![\bigvee_{j \in J} v_j \mid \psi_j]\!]_{\vec{E}/B} \subseteq [\![\bigvee_{k \in K} w_k \mid \phi_k]\!]_{\vec{E}/B}$, which holds if we can prove $v_i \mid \psi_j \sqsubseteq_{B_1} w_k \mid \phi_k$.
2. **Negatively**: If $Q^c = [\![\bigvee_{k \in K} w_k \mid \phi_k]\!]_{\vec{E}/B}$, then if $[\![\bigvee_{j \in J} v_j \mid \psi_j]\!]_{\vec{E}/B} \cap \subseteq [\![\bigvee_{k \in K} w_k \mid \phi_k]\!]_{\vec{E}/B} = \emptyset$ we have proved $[\![v_j \mid \psi_j]\!]_{\vec{E}/B} \subseteq Q$,

Proving Invariants and Inductive Invariants (III)

(3). **Proving Other Invariant**. Once we have proved that $\bigvee_{j\in J} v_j \mid \psi_j$ is an inductive invariant from $\bigvee_{i\in I} u_i \mid \varphi_i$, we can prove another invariant Q from $\bigvee_{i\in I} u_i \mid \varphi_i$ in one of two ways:

1. **Positively**: If $Q = [\![\bigvee_{k\in K} w_k \mid \phi_k]\!]_{\vec{E}/B}$ it is enough to show that $\forall j \in J\, \exists k \in K$ s.t. $[\![\bigvee_{j\in J} v_j \mid \psi_j]\!]_{\vec{E}/B} \subseteq [\![\bigvee_{k\in K} w_k \mid \phi_k]\!]_{\vec{E}/B}$, which holds if we can prove $v_j \mid \psi_j \sqsubseteq_{B_1} w_k \mid \phi_k$.
2. **Negatively**: If $Q^c = [\![\bigvee_{k\in K} w_k \mid \phi_k]\!]_{\vec{E}/B}$, then if $[\![\bigvee_{j\in J} v_j \mid \psi_j]\!]_{\vec{E}/B} \cap \subseteq [\![\bigvee_{k\in K} w_k \mid \phi_k]\!]_{\vec{E}/B} = \emptyset$ we have proved $[\![v_j \mid \psi_j]\!]_{\vec{E}/B} \subseteq Q$, i.e., Q is an invariant from $\bigvee_{i\in I} u_i \mid \varphi_i$.

Proving Invariants and Inductive Invariants (III)

(3). **Proving Other Invariant**. Once we have proved that $\bigvee_{j \in J} v_j \mid \psi_j$ is an inductive invariant from $\bigvee_{i \in I} u_i \mid \varphi_i$, we can prove another invariant Q from $\bigvee_{i \in I} u_i \mid \varphi_i$ in one of two ways:

1. **Positively**: If $Q = [\![\bigvee_{k \in K} w_k \mid \phi_k]\!]_{\vec{E}/B}$ it is enough to show that $\forall j \in J \, \exists k \in K$ s.t. $[\![\bigvee_{j \in J} v_j \mid \psi_j]\!]_{\vec{E}/B} \subseteq [\![\bigvee_{k \in K} w_k \mid \phi_k]\!]_{\vec{E}/B}$, which holds if we can prove $v_j \mid \psi_j \sqsubseteq_{B_1} w_k \mid \phi_k$.
2. **Negatively**: If $Q^c = [\![\bigvee_{k \in K} w_k \mid \phi_k]\!]_{\vec{E}/B}$, then if $[\![\bigvee_{j \in J} v_j \mid \psi_j]\!]_{\vec{E}/B} \cap \subseteq [\![\bigvee_{k \in K} w_k \mid \phi_k]\!]_{\vec{E}/B} = \emptyset$ we have proved $[\![v_j \mid \psi_j]\!]_{\vec{E}/B} \subseteq Q$, i.e., Q is an invariant from $\bigvee_{i \in I} u_i \mid \varphi_i$.

Positively, it is enough to show that $\forall j \in J \, \exists k \in K$ s.t. $v_j \mid \psi_j \sqsubseteq_{B_1} w_k \mid \phi_k$.

Proving Invariants and Inductive Invariants (III)

(3). **Proving Other Invariant**. Once we have proved that $\bigvee_{j \in J} v_j \mid \psi_j$ is an inductive invariant from $\bigvee_{i \in I} u_i \mid \varphi_i$, we can prove another invariant Q from $\bigvee_{i \in I} u_i \mid \varphi_i$ in one of two ways:

1. **Positively**: If $Q = [\![\bigvee_{k \in K} w_k \mid \phi_k]\!]_{\vec{E}/B}$ it is enough to show that $\forall j \in J\ \exists k \in K$ s.t. $[\![\bigvee_{j \in J} v_j \mid \psi_j]\!]_{\vec{E}/B} \subseteq [\![\bigvee_{k \in K} w_k \mid \phi_k]\!]_{\vec{E}/B}$, which holds if we can prove $v_i \mid \psi_j \sqsubseteq_{B_1} w_k \mid \phi_k$.
2. **Negatively**: If $Q^c = [\![\bigvee_{k \in K} w_k \mid \phi_k]\!]_{\vec{E}/B}$, then if $[\![\bigvee_{j \in J} v_j \mid \psi_j]\!]_{\vec{E}/B} \cap \subseteq [\![\bigvee_{k \in K} w_k \mid \phi_k]\!]_{\vec{E}/B} = \emptyset$ we have proved $[\![v_j \mid \psi_j]\!]_{\vec{E}/B} \subseteq Q$, i.e., Q is an invariant from $\bigvee_{i \in I} u_i \mid \varphi_i$.

Positively, it is enough to show that $\forall j \in J\ \exists k \in K$ s.t. $v_j \mid \psi_j \sqsubseteq_{B_1} w_k \mid \phi_k$. **Negatively**, it is enough to show that $\forall j \in J\ \forall k \in K\ \forall \theta \in \mathit{DisjUnif}_{B_1}(v_j = w_k)$ (the disjoint B_1-unifiers of $v_j = w_k$),

Proving Invariants and Inductive Invariants (III)

(3). **Proving Other Invariant**. Once we have proved that $\bigvee_{j \in J} v_j \mid \psi_j$ is an inductive invariant from $\bigvee_{i \in I} u_i \mid \varphi_i$, we can prove another invariant Q from $\bigvee_{i \in I} u_i \mid \varphi_i$ in one of two ways:

1. **Positively**: If $Q = [\![\bigvee_{k \in K} w_k \mid \phi_k]\!]_{\vec{E}/B}$ it is enough to show that $\forall j \in J\, \exists k \in K$ s.t. $[\![\bigvee_{j \in J} v_j \mid \psi_j]\!]_{\vec{E}/B} \subseteq [\![\bigvee_{k \in K} w_k \mid \phi_k]\!]_{\vec{E}/B}$, which holds if we can prove $v_i \mid \psi_j \sqsubseteq_{B_1} w_k \mid \phi_k$.
2. **Negatively**: If $Q^c = [\![\bigvee_{k \in K} w_k \mid \phi_k]\!]_{\vec{E}/B}$, then if $[\![\bigvee_{j \in J} v_j \mid \psi_j]\!]_{\vec{E}/B} \cap \subseteq [\![\bigvee_{k \in K} w_k \mid \phi_k]\!]_{\vec{E}/B} = \emptyset$ we have proved $[\![v_j \mid \psi_j]\!]_{\vec{E}/B} \subseteq Q$, i.e., Q is an invariant from $\bigvee_{i \in I} u_i \mid \varphi_i$.

Positively, it is enough to show that $\forall j \in J\, \exists k \in K$ s.t. $v_j \mid \psi_j \sqsubseteq_{B_1} w_k \mid \phi_k$. **Negatively**, it is enough to show that $\forall j \in J\, \forall k \in K\, \forall \theta \in \mathit{DisjUnif}_{B_1}(v_j = w_k)$ (the disjoint B_1-unifiers of $v_j = w_k$), $[\![(v_j \mid \psi_j \wedge \phi_k)\theta]\!]_{\vec{E}/B} = \emptyset$,

Proving Invariants and Inductive Invariants (III)

(3). **Proving Other Invariant**. Once we have proved that $\bigvee_{j\in J} v_j \mid \psi_j$ is an inductive invariant from $\bigvee_{i\in I} u_i \mid \varphi_i$, we can prove another invariant Q from $\bigvee_{i\in I} u_i \mid \varphi_i$ in one of two ways:

1. **Positively**: If $Q = [\![\bigvee_{k\in K} w_k \mid \phi_k]\!]_{\vec{E}/B}$ it is enough to show that $\forall j \in J\, \exists k \in K$ s.t. $[\![\bigvee_{j\in J} v_j \mid \psi_j]\!]_{\vec{E}/B} \subseteq [\![\bigvee_{k\in K} w_k \mid \phi_k]\!]_{\vec{E}/B}$, which holds if we can prove $v_i \mid \psi_j \sqsubseteq_{B_1} w_k \mid \phi_k$.
2. **Negatively**: If $Q^c = [\![\bigvee_{k\in K} w_k \mid \phi_k]\!]_{\vec{E}/B}$, then if $[\![\bigvee_{j\in J} v_j \mid \psi_j]\!]_{\vec{E}/B} \cap \subseteq [\![\bigvee_{k\in K} w_k \mid \phi_k]\!]_{\vec{E}/B} = \emptyset$ we have proved $[\![v_j \mid \psi_j]\!]_{\vec{E}/B} \subseteq Q$, i.e., Q is an invariant from $\bigvee_{i\in I} u_i \mid \varphi_i$.

Positively, it is enough to show that $\forall j \in J\, \exists k \in K$ s.t. $v_j \mid \psi_j \sqsubseteq_{B_1} w_k \mid \phi_k$. **Negatively**, it is enough to show that $\forall j \in J\, \forall k \in K\, \forall \theta \in \mathit{DisjUnif}_{B_1}(v_j = w_k)$ (the disjoint B_1-unifiers of $v_j = w_k$), $[\![(v_j \mid \psi_j \wedge \phi_k)\theta]\!]_{\vec{E}/B} = \emptyset$, i.e., $\mathbb{C}_{\Sigma/\vec{E},B} \models \neg(\psi_j \wedge \phi_k)\theta$.

The **DM-Check** Tool

Maude's Deductive Model Checker (**DM-Check**) is a tool under development by a team of researchers at the Technical University of Valencia, Spain, (S.Escobar, R. López and J. Sapiña), Postech University, South Korea (K. Bae), and UIUC (J. Meseguer).

The **DM-Check** Tool

Maude's Deductive Model Checker (**DM-Check**) is a tool under development by a team of researchers at the Technical University of Valencia, Spain, (S.Escobar, R. López and J. Sapiña), Postech University, South Korea (K. Bae), and UIUC (J. Meseguer). The tool (not yet relased) uses the NuITP and has been made available for CS 476 students thanks to the efforts of the **DM-Check** Team.

The **DM-Check** Tool

Maude's Deductive Model Checker (**DM-Check**) is a tool under development by a team of researchers at the Technical University of Valencia, Spain, (S.Escobar, R. López and J. Sapiña), Postech University, South Korea (K. Bae), and UIUC (J. Meseguer). The tool (not yet relased) uses the NuITP and has been made available for CS 476 students thanks to the efforts of the **DM-Check** Team.

For an overview, user instructions and the implementation see:

The **DM-Check** Tool

Maude's Deductive Model Checker (**DM-Check**) is a tool under development by a team of researchers at the Technical University of Valencia, Spain, (S.Escobar, R. López and J. Sapiña), Postech University, South Korea (K. Bae), and UIUC (J. Meseguer). The tool (not yet relased) uses the NuITP and has been made available for CS 476 students thanks to the efforts of the **DM-Check** Team.

For an overview, user instructions and the implementation see:

`https://safe-tools.dsic.upv.es/dmc/`

The **DM-Check** Tool

Maude's Deductive Model Checker (**DM-Check**) is a tool under development by a team of researchers at the Technical University of Valencia, Spain, (S.Escobar, R. López and J. Sapiña), Postech University, South Korea (K. Bae), and UIUC (J. Meseguer). The tool (not yet relased) uses the NuITP and has been made available for CS 476 students thanks to the efforts of the **DM-Check** Team.

For an overview, user instructions and the implementation see:

`https://safe-tools.dsic.upv.es/dmc/`

The current functionality supports verification of inductive and other invariants according the above methods (1)–(3).

The **DM-Check** Tool

Maude's Deductive Model Checker (**DM-Check**) is a tool under development by a team of researchers at the Technical University of Valencia, Spain, (S.Escobar, R. López and J. Sapiña), Postech University, South Korea (K. Bae), and UIUC (J. Meseguer). The tool (not yet relased) uses the NuITP and has been made available for CS 476 students thanks to the efforts of the **DM-Check** Team.

For an overview, user instructions and the implementation see:

`https://safe-tools.dsic.upv.es/dmc/`

The current functionality supports verification of inductive and other invariants according the above methods (1)–(3). Methods (1) and (3)-**Positive** are supported by the command, `check in M : ` $\bigvee_{i \in I} u_i \mid \varphi_i$ `subsumed-by` $\bigvee_{j \in J} v_j \mid \psi_j$ `.`

The **DM-Check** Tool

Maude's Deductive Model Checker (**DM-Check**) is a tool under development by a team of researchers at the Technical University of Valencia, Spain, (S.Escobar, R. López and J. Sapiña), Postech University, South Korea (K. Bae), and UIUC (J. Meseguer). The tool (not yet relased) uses the NuITP and has been made available for CS 476 students thanks to the efforts of the **DM-Check** Team.

For an overview, user instructions and the implementation see:

`https://safe-tools.dsic.upv.es/dmc/`

The current functionality supports verification of inductive and other invariants according the above methods (1)–(3). Methods (1) and (3)-**Positive** are supported by the command, `check in M` `:` $\bigvee_{i \in I} u_i \mid \varphi_i$ `subsumed-by` $\bigvee_{j \in J} v_j \mid \psi_j$ `.` And Method (2) by the command, `check-inv in M :` $\bigvee_{j \in J} v_j \mid \psi_j$ `.`

The **DM-Check** Tool

Maude's Deductive Model Checker (**DM-Check**) is a tool under development by a team of researchers at the Technical University of Valencia, Spain, (S.Escobar, R. López and J. Sapiña), Postech University, South Korea (K. Bae), and UIUC (J. Meseguer). The tool (not yet relased) uses the NuITP and has been made available for CS 476 students thanks to the efforts of the **DM-Check** Team.

For an overview, user instructions and the implementation see:

`https://safe-tools.dsic.upv.es/dmc/`

The current functionality supports verification of inductive and other invariants according the above methods (1)–(3). Methods (1) and (3)-**Positive** are supported by the command, `check in M` `:` $\bigvee_{i\in I} u_i \mid \varphi_i$ `subsumed-by` $\bigvee_{j\in J} v_j \mid \psi_j$ `.` And Method (2) by the command, `check-inv in M :` $\bigvee_{j\in J} v_j \mid \psi_j$ `.`

Let us prove inductive and other invariants with **DM-Check**.

An Inductive Invariant Case Study: `R&W`

A simple example like `R&W` can illustrate all the ideas.

An Inductive Invariant Case Study: R&W

A simple example like R&W can illustrate all the ideas.

```
mod R&W is
  sort Natural .
  op 0 : -> Natural [ctor] .
  op s : Natural -> Natural [ctor] .
  sort Config .
  op <_,_> : Natural Natural -> Config [ctor] .

  vars R W : Natural .

  rl [enter-w] : < 0, 0 > => < 0, s(0) > [narrowing] .
  rl [leave-w] : < R, s(W) > => < R, W > [narrowing] .
  rl [enter-r] : < R, 0 > => < s(R), 0 > [narrowing] .
  rl [leave-r] : < s(R), W > => < R, W > [narrowing] .
endm
```

An Inductive Invariant Case Study: R&W

A simple example like R&W can illustrate all the ideas.

```
mod R&W is
  sort Natural .
  op 0 : -> Natural [ctor] .
  op s : Natural -> Natural [ctor] .
  sort Config .
  op <_,_> : Natural Natural -> Config [ctor] .

  vars R W : Natural .

  rl [enter-w] : < 0, 0 > => < 0, s(0) > [narrowing] .
  rl [leave-w] : < R, s(W) > => < R, W > [narrowing] .
  rl [enter-r] : < R, 0 > => < s(R), 0 > [narrowing] .
  rl [leave-r] : < s(R), W > => < R, W > [narrowing] .
endm
```

We first enter it into Maude.

An Inductive Invariant Case Study: R&W

A simple example like R&W can illustrate all the ideas.

```
mod R&W is
  sort Natural .
  op 0 : -> Natural [ctor] .
  op s : Natural -> Natural [ctor] .
  sort Config .
  op <_,_> : Natural Natural -> Config [ctor] .

  vars R W : Natural .

  rl [enter-w] : < 0, 0 > => < 0, s(0) > [narrowing] .
  rl [leave-w] : < R, s(W) > => < R, W > [narrowing] .
  rl [enter-r] : < R, 0 > => < s(R), 0 > [narrowing] .
  rl [leave-r] : < s(R), W > => < R, W > [narrowing] .
endm
```

We first enter it into Maude. Then we load **DM-Check**:

```
Maude> load dm-check-ui.maude
```

An Inductive Invariant Case Study: R&W

A simple example like R&W can illustrate all the ideas.

```
mod R&W is
  sort Natural .
  op 0 : -> Natural [ctor] .
  op s : Natural -> Natural [ctor] .
  sort Config .
  op <_,_> : Natural Natural -> Config [ctor] .

  vars R W : Natural .

  rl [enter-w] : < 0, 0 > => < 0, s(0) > [narrowing] .
  rl [leave-w] : < R, s(W) > => < R, W > [narrowing] .
  rl [enter-r] : < R, 0 > => < s(R), 0 > [narrowing] .
  rl [leave-r] : < s(R), W > => < R, W > [narrowing] .
endm
```

We first enter it into Maude. Then we load **DM-Check**:

```
Maude> load dm-check-ui.maude
```

We are now ready to give commands to **DM-Check**.

An Inductive Invariant Case Study: R&W

A simple example like R&W can illustrate all the ideas.

```
mod R&W is
  sort Natural .
  op 0 : -> Natural [ctor] .
  op s : Natural -> Natural [ctor] .
  sort Config .
  op <_,_> : Natural Natural -> Config [ctor] .

  vars R W : Natural .

  rl [enter-w] : < 0, 0 > => < 0, s(0) > [narrowing] .
  rl [leave-w] : < R, s(W) > => < R, W > [narrowing] .
  rl [enter-r] : < R, 0 > => < s(R), 0 > [narrowing] .
  rl [leave-r] : < s(R), W > => < R, W > [narrowing] .
endm
```

We first enter it into Maude. Then we load **DM-Check**:

```
Maude> load dm-check-ui.maude
```

We are now ready to give commands to **DM-Check**. A natural guess for an inductive invariant for R&W is:

```
< N:Natural , 0 > | true \/ < 0 , s(0) > | true
```

An Inductive Invariant Case Study: `R&W` (II)

Can show containment of initial state $<$ `0,0` $>$ with the command:

An Inductive Invariant Case Study: R&W (II)

Can show containment of initial state < 0,0 > with the command:

```
DM-Check> check in R&W : (((< 0,0 >) | (true))) subsumed-by
(((< N:Natural , 0 >) | (true)) \/ ((< 0 , s(0) >) | (true))) .

  Subsumption satisfied.
```

Can prove that it is an inductive invariant with the command:

```
DM-Check> check-inv in R&W : (((< N:Natural , 0 >) | (true)) \/
((< 0 , s(0) >) | (true))) .

  Invariant satisfied.
```

Now we can show **Positively** that R&W satisfies the deadlock-freedom invariant from < 0,0 > with the command:

```
DM-Check> check in R&W : (((< N:Natural , 0 >) | (true)) \/
((< 0 , s(0) >) | (true))) subsumed-by (((< 0, 0 >) | (true)) \/
((< R:Natural, s(W:Natural) >) | (true)) \/ ((< R:Natural, 0 >) | (true))
\/ ((< s(R:Natural), W:Natural >) | (true))) .

  Subsumption satisfied.
```

An Inductive Invariant Case Study: `R&W` (III)

DM-Check does not yet support the **Negative** method to prove other invariants.

An Inductive Invariant Case Study: `R&W` (III)

DM-Check does not yet support the **Negative** method to prove other invariants. But, once we know that `< N:Natural , 0 > | true \/ < 0 , s(0) > | true` is inductive we can carry out such proofs in Maude itself.

An Inductive Invariant Case Study: R&W (III)

DM-Check does not yet support the **Negative** method to prove other invariants. But, once we know that `< N:Natural , 0 > | true \/ < 0 , s(0) > | true` is inductive we can carry out such proofs in Maude itself.

For example, the mutex invariant from `< 0, 0 >` is proved by the commands (unification is by construction disjoint because the two sides share no variables):

An Inductive Invariant Case Study: `R&W` (III)

DM-Check does not yet support the **Negative** method to prove other invariants. But, once we know that `< N:Natural , 0 > | true \/ < 0 , s(0) > | true` is inductive we can carry out such proofs in Maude itself.

For example, the mutex invariant from `< 0, 0 >` is proved by the commands (unification is by construction disjoint because the two sides share no variables):

```
Maude> unify < N:Natural,0 > =? < s(M:Natural),s(K:Natural) > .

No unifier.

Maude> unify < 0,s(0) > =? < s(M:Natural),s(K:Natural) > .

No unifier.
```

An Inductive Invariant Case Study: `R&W` (IV)

Likewise, we can prove the the one-writer invariant from `< 0, 0 >` by the commands:

An Inductive Invariant Case Study: R&W (IV)

Likewise, we can prove the the one-writer invariant from < 0, 0 > by the commands:

```
Maude> unify < N:Natural,0 > =? < M:Natural,s(s(K:Natural)) > .

No unifier.

Maude> unify < 0,s(0) > =? < M:Natural,s(s(K:Natural)) > .

No unifier.
```

Second Inductive Invariant Case Study: `R&W-FAIR`

`R&W` is unfair. Non-starvation for readers and writers is achieved by the following `R&W-FAIR` protocol, which is parametric on the maximum number n of readers that are allowed:

Second Inductive Invariant Case Study: R&W-FAIR

R&W is unfair. Non-starvation for readers and writers is achieved by the following R&W-FAIR protocol, which is parametric on the maximum number n of readers that are allowed:

```
mod R&W-FAIR is
  sorts NzNat Nat Conf .  subsorts NzNat < Nat .
  op 0 : -> Nat [ctor] .
  op 1 : -> NzNat [ctor] .
  op _+_ : Nat Nat -> Nat [assoc comm id: 0] .
  op _+_ : NzNat Nat -> NzNat [ctor assoc comm id: 0] .
  op [_]<_,_>[_|_] : Nat Nat Nat Nat Nat -> Conf .
  op init : NzNat -> Conf .
  vars N M K I J L : Nat .  var N' M' : NzNat .

  eq init(N') = [N']< 0,0 >[ 0 | N'] .

  rl [w-in] : [N]< 0,0 >[ 0 | N] => [N]< 0,1 >[0 | N] [narrowing] .
  rl [w-out] : [N]< 0,1 >[ 0 | N] => [N]< 0,0 >[N | 0] [narrowing] .
  rl [r-in] : [K + N + M + 1]< N,0 >[M + 1 | K] =>
                   [K + N + M + 1]< N + 1,0 >[M | K] [narrowing] .
  rl [r-out] : [K + N + M + 1]< N + 1,0 >[M | K] =>
                   [K + N + M + 1]< N,0 >[M | K + 1] [narrowing] .
endm
```

Second Inductive Invariant Case Study: R&W-FAIR

R&W is unfair. Non-starvation for readers and writers is achieved by the following R&W-FAIR protocol, which is parametric on the maximum number n of readers that are allowed:

```
mod R&W-FAIR is
  sorts NzNat Nat Conf .  subsorts NzNat < Nat .
  op 0 : -> Nat [ctor] .
  op 1 : -> NzNat [ctor] .
  op _+_ : Nat Nat -> Nat [assoc comm id: 0] .
  op _+_ : NzNat Nat -> NzNat [ctor assoc comm id: 0] .
  op [_]<_,_>[_|_] : Nat Nat Nat Nat Nat -> Conf .
  op init : NzNat -> Conf .
  vars N M K I J L : Nat .  var N' M' : NzNat .

  eq init(N') = [N']< 0,0 >[ 0 | N'] .

  rl [w-in] : [N]< 0,0 >[ 0 | N] => [N]< 0,1 >[0 | N] [narrowing] .
  rl [w-out] : [N]< 0,1 >[ 0 | N] => [N]< 0,0 >[N | 0] [narrowing] .
  rl [r-in] : [K + N + M + 1]< N,0 >[M + 1 | K] =>
                    [K + N + M + 1]< N + 1,0 >[M | K] [narrowing] .
  rl [r-out] : [K + N + M + 1]< N + 1,0 >[M | K] =>
                    [K + N + M + 1]< N,0 >[M | K + 1] [narrowing] .
endm
```

Second Inductive Invariant Case Study: R&W-FAIR (II)

A natural guess for an inductive invariant is:

```
[N']< 0,0 >[ 0 | N'] | true \/ [N']< 0,1 >[0 | N'] | true \/
[N' + K + M]< M,0 >[N' | K] | true \/ [N' + K + M]< M,0 >[K | N'] | true
\/ [N' + K + M]< N',0 >[M | K] | true
```

Second Inductive Invariant Case Study: R&W-FAIR (II)

A natural guess for an inductive invariant is:

```
[N']< 0,0 >[ 0 | N'] | true \/ [N']< 0,1 >[0 | N'] | true \/
[N' + K + M]< M,0 >[N' | K] | true \/ [N' + K + M]< M,0 >[K | N'] | true
\/ [N' + K + M]< N',0 >[M | K] | true
```

We can check that it contains the parametric initial state thus:

Second Inductive Invariant Case Study: R&W-FAIR (II)

A natural guess for an inductive invariant is:

```
[N']< 0,0 >[ 0 | N'] | true \/ [N']< 0,1 >[0 | N'] | true \/
[N' + K + M]< M,0 >[N' | K] | true \/ [N' + K + M]< M,0 >[K | N'] | true
\/ [N' + K + M]< N',0 >[M | K] | true
```

We can check that it contains the parametric initial state thus:

```
DM-Check> check  in R&W-FAIR : ((([N':NzNat]< 0,0 >[ 0 | N':NzNat]) | (true)))
subsumed-by  ((([N':NzNat]< 0,0 >[ 0 | N':NzNat]) | (true)) \/
(([N':NzNat]< 0,1 >[ 0 | N':NzNat]) | (true)) \/
(([N':NzNat + K:Nat + M:Nat]< M:Nat,0 >[N':NzNat | K:Nat]) | (true)) \/
(([N':NzNat + K:Nat + M:Nat]< N':NzNat,0 >[M:Nat | K:Nat]) | (true)) \/
(([N':NzNat + K:Nat + M:Nat]< M:Nat,0 >[K:Nat | N':NzNat]) | (true))) .

  Subsumption satisfied.
```

We can also check that our guess invariant is inductive by giving the command:

Second Inductive Invariant Case Study: R&W-FAIR (III)

```
DM-Check> check-inv in R&W-FAIR : ([N':NzNat]< 0,0 >[ 0 | N':NzNat]) | true \/
([N':NzNat]< 0,1 >[ 0 | N':NzNat]) | true \/
([N':NzNat + K:Nat + M:Nat]< M:Nat,0 >[N':NzNat | K:Nat]) | true \/
([N':NzNat + K:Nat + M:Nat]< N':NzNat,0 >[M:Nat | K:Nat]) | true \/
([N':NzNat + K:Nat + M:Nat]< M:Nat,0 >[K:Nat | N':NzNat]) | true .

  Invariant satisfied.
```

Second Inductive Invariant Case Study: R&W-FAIR (III)

```
DM-Check> check-inv in R&W-FAIR : ([N':NzNat]< 0,0 >[ 0 | N':NzNat]) | true \/
([N':NzNat]< 0,1 >[ 0 | N':NzNat]) | true \/
([N':NzNat + K:Nat + M:Nat]< M:Nat,0 >[N':NzNat | K:Nat]) | true \/
([N':NzNat + K:Nat + M:Nat]< N':NzNat,0 >[M:Nat | K:Nat]) | true \/
([N':NzNat + K:Nat + M:Nat]< M:Nat,0 >[K:Nat | N':NzNat]) | true .

  Invariant satisfied.
```

We can verify mutex **Negatively** thus:

Second Inductive Invariant Case Study: R&W-FAIR (III)

```
DM-Check> check-inv in R&W-FAIR : ([N':NzNat]< 0,0 >[ 0 | N':NzNat]) | true \/
([N':NzNat]< 0,1 >[ 0 | N':NzNat]) | true \/
([N':NzNat + K:Nat + M:Nat]< M:Nat,0 >[N':NzNat | K:Nat]) | true \/
([N':NzNat + K:Nat + M:Nat]< N':NzNat,0 >[M:Nat | K:Nat]) | true \/
([N':NzNat + K:Nat + M:Nat]< M:Nat,0 >[K:Nat | N':NzNat]) | true .

  Invariant satisfied.
```

We can verify mutex **Negatively** thus:

```
Maude> unify [N']< 0,0 >[ 0 | N'] =?
                             [N + 1 + I + 1 + J + L]< N + 1,I + 1 >[L | J] .
No unifier.
Maude> unify [N']< 0,1 >[0 | N'] =?
                            [N + 1 + I + 1 + J + L]< N + 1,I + 1 >[L | J] .
No unifier.
Maude> unify [N' + K + M]< M,0 >[N' | K] =?
                           [N + 1 + I + 1 + J + L]< N + 1,I + 1 >[L | J] .
No unifier.
Maude> unify [N' + K + M]< N',0 >[M | K] =?
                            [N + 1 + I + 1 + J + L]< N + 1,I + 1 >[L | J] .
No unifier.
Maude> unify [N' + K + M]< M,0 >[K | N'] =?
             [N + 1 + I + 1 + J + L]< N + 1,I + 1 >[L | J] .      No unifier.
```

Second Inductive Invariant Case Study: R&W-FAIR (III)

```
DM-Check> check-inv in R&W-FAIR : ([N':NzNat]< 0,0 >[ 0 | N':NzNat]) | true \/
([N':NzNat]< 0,1 >[ 0 | N':NzNat]) | true \/
([N':NzNat + K:Nat + M:Nat]< M:Nat,0 >[N':NzNat | K:Nat]) | true \/
([N':NzNat + K:Nat + M:Nat]< N':NzNat,0 >[M:Nat | K:Nat]) | true \/
([N':NzNat + K:Nat + M:Nat]< M:Nat,0 >[K:Nat | N':NzNat]) | true .

  Invariant satisfied.
```

We can verify mutex **Negatively** thus:

```
Maude> unify [N']< 0,0 >[ 0 | N'] =?
                             [N + 1 + I + 1 + J + L]< N + 1,I + 1 >[L | J] .
No unifier.
Maude> unify [N']< 0,1 >[0 | N'] =?
                            [N + 1 + I + 1 + J + L]< N + 1,I + 1 >[L | J] .
No unifier.
Maude> unify [N' + K + M]< M,0 >[N' | K] =?
                           [N + 1 + I + 1 + J + L]< N + 1,I + 1 >[L | J] .
No unifier.
Maude> unify [N' + K + M]< N',0 >[M | K] =?
                            [N + 1 + I + 1 + J + L]< N + 1,I + 1 >[L | J] .
No unifier.
Maude> unify [N' + K + M]< M,0 >[K | N'] =?
            [N + 1 + I + 1 + J + L]< N + 1,I + 1 >[L | J] .      No unifier.
```

Second Inductive Invariant Case Study: R&W-FAIR (III)

```
DM-Check> check-inv in R&W-FAIR : ([N':NzNat]< 0,0 >[ 0 | N':NzNat]) | true \/
([N':NzNat]< 0,1 >[ 0 | N':NzNat]) | true \/
([N':NzNat + K:Nat + M:Nat]< M:Nat,0 >[N':NzNat | K:Nat]) | true \/
([N':NzNat + K:Nat + M:Nat]< N':NzNat,0 >[M:Nat | K:Nat]) | true \/
([N':NzNat + K:Nat + M:Nat]< M:Nat,0 >[K:Nat | N':NzNat]) | true .

  Invariant satisfied.
```

We can verify mutex **Negatively** thus:

```
Maude> unify [N']< 0,0 >[ 0 | N'] =?
                             [N + 1 + I + 1 + J + L]< N + 1,I + 1 >[L | J] .
No unifier.
Maude> unify [N']< 0,1 >[0 | N'] =?
                            [N + 1 + I + 1 + J + L]< N + 1,I + 1 >[L | J] .
No unifier.
Maude> unify [N' + K + M]< M,0 >[N' | K] =?
                           [N + 1 + I + 1 + J + L]< N + 1,I + 1 >[L | J] .
No unifier.
Maude> unify [N' + K + M]< N',0 >[M | K] =?
                            [N + 1 + I + 1 + J + L]< N + 1,I + 1 >[L | J] .
No unifier.
Maude> unify [N' + K + M]< M,0 >[K | N'] =?
            [N + 1 + I + 1 + J + L]< N + 1,I + 1 >[L | J] .     No unifier.
```

Second Inductive Invariant Case Study: R&W-FAIR (III)

```
DM-Check> check-inv in R&W-FAIR : ([N':NzNat]< 0,0 >[ 0 | N':NzNat]) | true \/
([N':NzNat]< 0,1 >[ 0 | N':NzNat]) | true \/
([N':NzNat + K:Nat + M:Nat]< M:Nat,0 >[N':NzNat | K:Nat]) | true \/
([N':NzNat + K:Nat + M:Nat]< N':NzNat,0 >[M:Nat | K:Nat]) | true \/
([N':NzNat + K:Nat + M:Nat]< M:Nat,0 >[K:Nat | N':NzNat]) | true .

  Invariant satisfied.
```

We can verify mutex **Negatively** thus:

```
Maude> unify [N']< 0,0 >[ 0 | N'] =?
                             [N + 1 + I + 1 + J + L]< N + 1,I + 1 >[L | J] .
No unifier.
Maude> unify [N']< 0,1 >[0 | N'] =?
                            [N + 1 + I + 1 + J + L]< N + 1,I + 1 >[L | J] .
No unifier.
Maude> unify [N' + K + M]< M,0 >[N' | K] =?
                           [N + 1 + I + 1 + J + L]< N + 1,I + 1 >[L | J] .
No unifier.
Maude> unify [N' + K + M]< N',0 >[M | K] =?
                            [N + 1 + I + 1 + J + L]< N + 1,I + 1 >[L | J] .
No unifier.
Maude> unify [N' + K + M]< M,0 >[K | N'] =?
             [N + 1 + I + 1 + J + L]< N + 1,I + 1 >[L | J] .   No unifier.
```

Second Inductive Invariant Case Study: R&W-FAIR (IV)

And we can verify one-writer **Negatively** thus:

```
Maude> unify [N']< 0,0 >[ 0 | N'] =?
                         [N + 1 + I + 1 + J + L]< N,I + 1 + 1 >[L | J] .

No unifier.
Maude> unify [N']< 0,1 >[0 | N'] =?
                         [N + 1 + I + 1 + J + L]< N,I + 1 + 1 >[L | J] .

No unifier.
Maude> unify [N' + K + M]< M,0 >[N' | K] =?
                         [N + 1 + I + 1 + J + L]< N,I + 1 + 1 >[L | J] .

No unifier.
Maude> unify [N' + K + M]< N',0 >[M | K] =?
                         [N + 1 + I + 1 + J + L]< N,I + 1 + 1 >[L | J] .

No unifier.

Maude> unify [N' + K + M]< M,0 >[K | N'] =?
                         [N + 1 + I + 1 + J + L]< N,I + 1 + 1 >[L | J] .

No unifier.
```

Second Inductive Invariant Case Study: R&W-FAIR (IV)

And we can verify one-writer **Negatively** thus:

```
Maude> unify [N']< 0,0 >[ 0 | N'] =?
                          [N + 1 + I + 1 + J + L]< N,I + 1 + 1 >[L | J] .

No unifier.
Maude> unify [N']< 0,1 >[0 | N'] =?
                          [N + 1 + I + 1 + J + L]< N,I + 1 + 1 >[L | J] .

No unifier.
Maude> unify [N' + K + M]< M,0 >[N' | K] =?
                          [N + 1 + I + 1 + J + L]< N,I + 1 + 1 >[L | J] .

No unifier.
Maude> unify [N' + K + M]< N',0 >[M | K] =?
                          [N + 1 + I + 1 + J + L]< N,I + 1 + 1 >[L | J] .

No unifier.

Maude> unify [N' + K + M]< M,0 >[K | N'] =?
                          [N + 1 + I + 1 + J + L]< N,I + 1 + 1 >[L | J] .

No unifier.
```

Second Inductive Invariant Case Study: R&W-FAIR (V)

We can try to verify **Positively** deadlock-freedom thus:

```
DM-Check> check  in R&W-FAIR : (([N':NzNat]< 0,0 >[ 0 | N':NzNat]) | true) \/
(([N':NzNat]< 0,1 >[ 0 | N':NzNat]) | true) \/
(([N':NzNat + K:Nat + M:Nat]< M:Nat,0 >[N':NzNat | K:Nat]) | true) \/
(([N':NzNat + K:Nat + M:Nat]< N':NzNat,0 >[M:Nat | K:Nat]) | true) \/
(([N':NzNat + K:Nat + M:Nat]< M:Nat,0 >[K:Nat | N':NzNat]) | true) subsumed-by
((([N:Nat]< 0,0 >[ 0 | N:Nat]) | true) \/ (([N:Nat]< 0,1 >[ 0 | N:Nat]) | true)
\/ (([K:Nat + N:Nat + M:Nat + 1]< N:Nat,0 >[M:Nat + 1 | K:Nat]) | true) \/
(([K:Nat + N:Nat + M:Nat + 1]< (N:Nat + 1), 0 >[M:Nat | K:Nat]) | true)) .

  Constrained terms on the left that could not be subsumed:

Term 7: [N':NzNat + K:Nat + M:Nat] < M:Nat, 0 >[N':NzNat | K:Nat]
Constraint 7: true

Term 8: [N':NzNat + K:Nat + M:Nat] < N':NzNat, 0 >[M:Nat | K:Nat]
Constraint 8: true

Term 9: [N':NzNat + K:Nat + M:Nat] < M:Nat, 0 >[K:Nat | N':NzNat]
Constraint 9: true
```

Second Inductive Invariant Case Study: R&W-FAIR (V)

We can try to verify **Positively** deadlock-freedom thus:

```
DM-Check> check  in R&W-FAIR : (([N':NzNat]< 0,0 >[ 0 | N':NzNat]) | true) \/
(([N':NzNat]< 0,1 >[ 0 | N':NzNat]) | true) \/
(([N':NzNat + K:Nat + M:Nat]< M:Nat,0 >[N':NzNat | K:Nat]) | true) \/
(([N':NzNat + K:Nat + M:Nat]< N':NzNat,0 >[M:Nat | K:Nat]) | true) \/
(([N':NzNat + K:Nat + M:Nat]< M:Nat,0 >[K:Nat | N':NzNat]) | true) subsumed-by
((([N:Nat]< 0,0 >[ 0 | N:Nat]) | true) \/ (([N:Nat]< 0,1 >[ 0 | N:Nat]) | true)
\/ (([K:Nat + N:Nat + M:Nat + 1]< N:Nat,0 >[M:Nat + 1 | K:Nat]) | true) \/
(([K:Nat + N:Nat + M:Nat + 1]< (N:Nat + 1), 0 >[M:Nat | K:Nat]) | true)) .

  Constrained terms on the left that could not be subsumed:

Term 7: [N':NzNat + K:Nat + M:Nat] < M:Nat, 0 >[N':NzNat | K:Nat]
Constraint 7: true

Term 8: [N':NzNat + K:Nat + M:Nat] < N':NzNat, 0 >[M:Nat | K:Nat]
Constraint 8: true

Term 9: [N':NzNat + K:Nat + M:Nat] < M:Nat, 0 >[K:Nat | N':NzNat]
Constraint 9: true
```

This just means that further reasoning is needed.

Sufficient but not Necessary Conditions

The methods (1)–(3) and their **DM-Check** commands are based on sufficient conditions that need not hold in general because:

Sufficient but not Necessary Conditions

The methods (1)–(3) and their **DM-Check** commands are based on sufficient conditions that need not hold in general because:

1. We need not have subsumptions $u_i \mid \varphi_i \sqsubseteq_{B_1} v_j \mid \psi_j$.

Sufficient but not Necessary Conditions

The methods (1)–(3) and their **DM-Check** commands are based on sufficient conditions that need not hold in general because:

1. We need not have subsumptions $u_i \mid \varphi_i \sqsubseteq_{B_1} v_j \mid \psi_j$.
2. Even if we do, the inductive validity of formulas like $\varphi_i \Rightarrow (\psi_j \alpha)$ and $(\psi_j \wedge \phi)\gamma \Rightarrow (\psi_{j'} \alpha)$ may require non-trivial inductive proofs.

Sufficient but not Necessary Conditions

The methods (1)–(3) and their **DM-Check** commands are based on sufficient conditions that need not hold in general because:

1. We need not have subsumptions $u_i \mid \varphi_i \sqsubseteq_{B_1} v_j \mid \psi_j$.
2. Even if we do, the inductive validity of formulas like $\varphi_i \Rightarrow (\psi_j \alpha)$ and $(\psi_j \wedge \phi)\gamma \Rightarrow (\psi_{j'} \alpha)$ may require non-trivial inductive proofs.
3. Likewise, in the **Negative** method of proving invariants, unifications may yield constrained terms $(v_j \mid \psi_j \wedge \phi_k)\theta$ whose constraint $(\psi_j \wedge \phi_k)\theta$ is inductively unsatisfiable; but showing this may require non-trivial inductive proofs.

Sufficient but not Necessary Conditions

The methods (1)–(3) and their **DM-Check** commands are based on sufficient conditions that need not hold in general because:

1. We need not have subsumptions $u_i \mid \varphi_i \sqsubseteq_{B_1} v_j \mid \psi_j$.
2. Even if we do, the inductive validity of formulas like $\varphi_i \Rightarrow (\psi_j \alpha)$ and $(\psi_j \wedge \phi)\gamma \Rightarrow (\psi_{j'} \alpha)$ may require non-trivial inductive proofs.
3. Likewise, in the **Negative** method of proving invariants, unifications may yield constrained terms $(v_j \mid \psi_j \wedge \phi_k)\theta$ whose constraint $(\psi_j \wedge \phi_k)\theta$ is inductively unsatisfiable; but showing this may require non-trivial inductive proofs.

For all these reasons deductive methods are needed to handle the cases where **DM-Check** cannot achieve an automatic proof.

Sufficient but not Necessary Conditions

The methods (1)–(3) and their **DM-Check** commands are based on sufficient conditions that need not hold in general because:

1. We need not have subsumptions $u_i \mid \varphi_i \sqsubseteq_{B_1} v_j \mid \psi_j$.
2. Even if we do, the inductive validity of formulas like $\varphi_i \Rightarrow (\psi_j \alpha)$ and $(\psi_j \wedge \phi)\gamma \Rightarrow (\psi_{j'} \alpha)$ may require non-trivial inductive proofs.
3. Likewise, in the **Negative** method of proving invariants, unifications may yield constrained terms $(v_j \mid \psi_j \wedge \phi_k)\theta$ whose constraint $(\psi_j \wedge \phi_k)\theta$ is inductively unsatisfiable; but showing this may require non-trivial inductive proofs.

For all these reasons deductive methods are needed to handle the cases where **DM-Check** cannot achieve an automatic proof. Some of these methods will be discussed in Lecture 29.