Protokollsicherheit

Automated Verification – Reachability Secrecy

Logic Programming (Prolog)

Logic Programming (Prolog)

The main motto is **to described declaratively the solution of a problem and not how it should be solved**.

Logic Programming (Prolog)

The main motto is **to described declaratively the solution of a problem and not how it should be solved**.

Example

Imperative solution:

```
int sum(int[] list) =
    result = 0;
    for(int i = 0; i ¡ list.length; ++i)
        result += list[i]
    return result
```

Logic Programming (Prolog)

The main motto is **to described declaratively the solution of a problem and not how it should be solved**.

Example

Imperative solution:

```
int sum(int[] list) =
    result = 0;
    for(int i = 0; i ¡ list.length; ++i)
        result += list[i]
    return result
```

We define how to compute!

Logic Programming (Prolog)

The main motto is **to described declaratively the solution of a problem and not how it should be solved**.

Example

Functional solution:

```
fun sum ([]) = 0
    | sum(h :: lst) = h + sum(lst)
```

We define how to compute!

Logic Programming (Prolog)

The main motto is **to described declaratively the solution of a problem and not how it should be solved**.

Example

Prolog solution:

$\text{sum}([], 0) \wedge$
$\forall H, T, N, M.[\text{sum}(T, M) \wedge \text{eq}(N, M + H) \supset \text{sum}([H|T], N)]$

Logic Programming (Prolog)

The main motto is **to described declaratively the solution of a problem and not how it should be solved**.

Example

Prolog solution:

$\text{sum}([], 0) \wedge$
$\forall H, T, N, M.[\text{sum}(T, M) \wedge \text{eq}(N, M + H) \supset \text{sum}([H|T], N)]$

Programs are logical theories!

We do not specify how to compute it, but just state what is true about it.

Logic Programming (Prolog)

Logic Programming (Prolog)

Prolog Program

$\text{sum}([], 0) \wedge$
$\forall H, T, N, M.[\text{sum}(T, M) \wedge \text{eq}(N, M + H) \supset \text{sum}([H|T], N)]$

Logic Programming (Prolog)

Prolog Program

$\text{sum}([], 0) \wedge$
$\forall H, T, N, M.[\text{sum}(T, M) \wedge \text{eq}(N, M + H) \supset \text{sum}([H|T], N)]$

Query

$\text{sum}([4, 2, 3, 4], X).$

Logic Programming (Prolog)

Prolog Program

$\text{sum}([], 0) \wedge$
$\forall H, T, N, M.[\text{sum}(T, M) \wedge \text{eq}(N, M + H) \supset \text{sum}([H|T], N)]$

Query

$\text{sum}([4, 2, 3, 4], X).$

Answer

$X = 13$

Logic Programming (Prolog)

Prolog Program

$\text{sum}([], 0) \wedge$
$\forall H, T, N, M.[\text{sum}(T, M) \wedge \text{eq}(N, M + H) \supset \text{sum}([H|T], N)]$

Query

$\text{sum}([4, 2, 3, 4], X).$

Answer

$X = 13$

What the prolog engine is doing is searching for a proof demonstrating that the query **logically** follows from the logic program.

Proof Theory Basics – Sequent Calculus

Proof Theory Basics – Sequent Calculus

Proof theory is the field of mathematics that studies the properties and structure of **formal** proofs.

Proof Theory Basics – Sequent Calculus

Proof theory is the field of mathematics that studies the properties and structure of **formal** proofs.

Sequents

$$F_1, \ldots, F_n \longrightarrow G_1, \ldots, G_m$$

$$F_1 \wedge \cdots \wedge F_n \supset G_1 \vee \cdots \vee G_m$$

Proof Theory Basics – Inference Rules

Proof Theory Basics – Inference Rules

Identity Rules

$$\frac{}{A \longrightarrow A}\ \text{Init} \qquad \frac{\Gamma_1 \longrightarrow A, \Delta_1 \quad \Gamma_2, A \longrightarrow \Delta_2}{\Gamma_1, \Gamma_2 \longrightarrow \Delta_1, \Delta_2}\ \text{Cut}$$

Proof Theory Basics – Inference Rules

Identity Rules

$$\frac{}{A \longrightarrow A}\ \text{Init} \qquad \frac{\Gamma_1 \longrightarrow A, \Delta_1 \quad \Gamma_2, A \longrightarrow \Delta_2}{\Gamma_1, \Gamma_2 \longrightarrow \Delta_1, \Delta_2}\ \text{Cut}$$

Logical Rules

$$\frac{\Gamma_1, A \longrightarrow \Delta_1 \quad \Gamma_2, B \longrightarrow \Delta_2}{\Gamma_1, \Gamma_2, A \vee B \longrightarrow \Delta_1, \Delta_2}\ \vee_l \qquad \frac{\Gamma \longrightarrow A_1, A_2, \Delta}{\Gamma \longrightarrow A_1 \vee A_2, \Delta}\ \vee_r$$

Proof Theory Basics – Inference Rules

Identity Rules

$$\frac{}{A \longrightarrow A}\ \text{Init} \qquad \frac{\Gamma_1 \longrightarrow A, \Delta_1 \quad \Gamma_2, A \longrightarrow \Delta_2}{\Gamma_1, \Gamma_2 \longrightarrow \Delta_1, \Delta_2}\ \text{Cut}$$

Logical Rules

$$\frac{\Gamma_1, A \longrightarrow \Delta_1 \quad \Gamma_2, B \longrightarrow \Delta_2}{\Gamma_1, \Gamma_2, A \vee B \longrightarrow \Delta_1, \Delta_2}\ \vee_l \qquad \frac{\Gamma \longrightarrow A_1, A_2, \Delta}{\Gamma \longrightarrow A_1 \vee A_2, \Delta}\ \vee_r$$

$$\frac{\Gamma, A, B \longrightarrow \Delta}{\Gamma, A \wedge B \longrightarrow \Delta}\ \wedge_l \qquad \frac{\Gamma_1 \longrightarrow \Delta_1, G_1 \quad \Gamma_2 \longrightarrow \Delta_2, G_2}{\Gamma_1, \Gamma_2 \longrightarrow \Delta_1, \Delta_2, G_1 \wedge G_2}\ \wedge_r$$

Proof Theory Basics – Inference Rules

Identity Rules

$$\frac{}{A \longrightarrow A}\ \text{Init} \qquad \frac{\Gamma_1 \longrightarrow A, \Delta_1 \quad \Gamma_2, A \longrightarrow \Delta_2}{\Gamma_1, \Gamma_2 \longrightarrow \Delta_1, \Delta_2}\ \text{Cut}$$

Logical Rules

$$\frac{\Gamma_1, A \longrightarrow \Delta_1 \quad \Gamma_2, B \longrightarrow \Delta_2}{\Gamma_1, \Gamma_2, A \vee B \longrightarrow \Delta_1, \Delta_2}\ \vee_l \qquad \frac{\Gamma \longrightarrow A_1, A_2, \Delta}{\Gamma \longrightarrow A_1 \vee A_2, \Delta}\ \vee_r$$

$$\frac{\Gamma, A, B \longrightarrow \Delta}{\Gamma, A \wedge B \longrightarrow \Delta}\ \wedge_l \qquad \frac{\Gamma_1 \longrightarrow \Delta_1, G_1 \quad \Gamma_2 \longrightarrow \Delta_2, G_2}{\Gamma_1, \Gamma_2 \longrightarrow \Delta_1, \Delta_2, G_1 \wedge G_2}\ \wedge_r$$

$$\frac{\Gamma_1 \longrightarrow A, \Delta_1 \qquad \Gamma_2, B \longrightarrow \Delta_2}{\Gamma_1, \Gamma_2, A \supset B \longrightarrow \Delta_1, \Delta_2}\ \supset_l \qquad \frac{\Gamma, A \longrightarrow B, \Delta}{\Gamma \longrightarrow A \supset B, \Delta}\ \supset_r$$

Proof Theory Basics – Inference Rules

Identity Rules

$$\frac{}{A \longrightarrow A}\ \text{Init} \qquad \frac{\Gamma_1 \longrightarrow A, \Delta_1 \quad \Gamma_2, A \longrightarrow \Delta_2}{\Gamma_1, \Gamma_2 \longrightarrow \Delta_1, \Delta_2}\ \text{Cut}$$

Logical Rules

$$\frac{\Gamma_1, A \longrightarrow \Delta_1 \quad \Gamma_2, B \longrightarrow \Delta_2}{\Gamma_1, \Gamma_2, A \vee B \longrightarrow \Delta_1, \Delta_2}\ \vee_l \qquad \frac{\Gamma \longrightarrow A_1, A_2, \Delta}{\Gamma \longrightarrow A_1 \vee A_2, \Delta}\ \vee_r$$

$$\frac{\Gamma, A, B \longrightarrow \Delta}{\Gamma, A \wedge B \longrightarrow \Delta}\ \wedge_l \qquad \frac{\Gamma_1 \longrightarrow \Delta_1, G_1 \quad \Gamma_2 \longrightarrow \Delta_2, G_2}{\Gamma_1, \Gamma_2 \longrightarrow \Delta_1, \Delta_2, G_1 \wedge G_2}\ \wedge_r$$

$$\frac{\Gamma_1 \longrightarrow A, \Delta_1 \qquad \Gamma_2, B \longrightarrow \Delta_2}{\Gamma_1, \Gamma_2, A \supset B \longrightarrow \Delta_1, \Delta_2}\ \supset_l \qquad \frac{\Gamma, A \longrightarrow B, \Delta}{\Gamma \longrightarrow A \supset B, \Delta}\ \supset_r$$

$$\frac{\Gamma, A\{t/x\} \longrightarrow \Delta}{\Gamma, \forall A \longrightarrow \Delta}\ \forall_l \qquad \frac{\Gamma \longrightarrow A\{c/x\}, \Delta}{\Gamma \longrightarrow \forall x\, A, \Delta}\ \forall_r$$

Proof Theory Basics – Inference Rules

Identity Rules

$$\frac{}{A \longrightarrow A}\ \text{Init} \qquad \frac{\Gamma_1 \longrightarrow A, \Delta_1 \quad \Gamma_2, A \longrightarrow \Delta_2}{\Gamma_1, \Gamma_2 \longrightarrow \Delta_1, \Delta_2}\ \text{Cut}$$

Logical Rules

$$\frac{\Gamma_1, A \longrightarrow \Delta_1 \quad \Gamma_2, B \longrightarrow \Delta_2}{\Gamma_1, \Gamma_2, A \vee B \longrightarrow \Delta_1, \Delta_2}\ \vee_l \qquad \frac{\Gamma \longrightarrow A_1, A_2, \Delta}{\Gamma \longrightarrow A_1 \vee A_2, \Delta}\ \vee_r$$

$$\frac{\Gamma, A, B \longrightarrow \Delta}{\Gamma, A \wedge B \longrightarrow \Delta}\ \wedge_l \qquad \frac{\Gamma_1 \longrightarrow \Delta_1, G_1 \quad \Gamma_2 \longrightarrow \Delta_2, G_2}{\Gamma_1, \Gamma_2 \longrightarrow \Delta_1, \Delta_2, G_1 \wedge G_2}\ \wedge_r$$

$$\frac{\Gamma_1 \longrightarrow A, \Delta_1 \qquad \Gamma_2, B \longrightarrow \Delta_2}{\Gamma_1, \Gamma_2, A \supset B \longrightarrow \Delta_1, \Delta_2}\ \supset_l \qquad \frac{\Gamma, A \longrightarrow B, \Delta}{\Gamma \longrightarrow A \supset B, \Delta}\ \supset_r$$

$$\frac{\Gamma, A\{t/x\} \longrightarrow \Delta}{\Gamma, \forall A \longrightarrow \Delta}\ \forall_l \qquad \frac{\Gamma \longrightarrow A\{c/x\}, \Delta}{\Gamma \longrightarrow \forall x\, A, \Delta}\ \forall_r$$

$$\frac{}{\Gamma, \bot \longrightarrow \Delta}\ \bot_L$$

Proof Theory Basics – Inference Rules

Structural Rules

Proof Theory Basics – Inference Rules

Structural Rules

$$\frac{\Gamma \longrightarrow \Delta}{\Gamma, A \longrightarrow \Delta}\ W_l \qquad \frac{\Gamma \longrightarrow \Delta}{\Gamma \longrightarrow A, \Delta}\ W_r$$

Proof Theory Basics – Inference Rules

Structural Rules

$$\frac{\Gamma \longrightarrow \Delta}{\Gamma, A \longrightarrow \Delta}\ W_l \qquad \frac{\Gamma \longrightarrow \Delta}{\Gamma \longrightarrow A, \Delta}\ W_r$$

$$\frac{\Gamma, A, A \longrightarrow \Delta}{\Gamma, A \longrightarrow \Delta}\ C_l \qquad \frac{\Gamma \longrightarrow A, A, \Delta}{\Gamma \longrightarrow A, \Delta}\ C_r$$

Proof Theory Basics – Inference Rules

Structural Rules

$$\frac{\Gamma \longrightarrow \Delta}{\Gamma, A \longrightarrow \Delta}\ W_l \qquad \frac{\Gamma \longrightarrow \Delta}{\Gamma \longrightarrow A, \Delta}\ W_r$$

$$\frac{\Gamma, A, A \longrightarrow \Delta}{\Gamma, A \longrightarrow \Delta}\ C_l \qquad \frac{\Gamma \longrightarrow A, A, \Delta}{\Gamma \longrightarrow A, \Delta}\ C_r$$

$$\frac{\Gamma, B, A, \Gamma' \longrightarrow \Delta}{\Gamma, A, B, \Gamma' \longrightarrow \Delta}\ X_l \qquad \frac{\Gamma \longrightarrow \Delta, B, A, \Delta'}{\Gamma \longrightarrow \Delta, A, B, \Delta'}\ X_r$$

Proof Theory Basics – Inference Rules

Structural Rules

$$\frac{\Gamma \longrightarrow \Delta}{\Gamma, A \longrightarrow \Delta}\ W_l \qquad \frac{\Gamma \longrightarrow \Delta}{\Gamma \longrightarrow A, \Delta}\ W_r$$

$$\frac{\Gamma, A, A \longrightarrow \Delta}{\Gamma, A \longrightarrow \Delta}\ C_l \qquad \frac{\Gamma \longrightarrow A, A, \Delta}{\Gamma \longrightarrow A, \Delta}\ C_r$$

$$\frac{\Gamma, B, A, \Gamma' \longrightarrow \Delta}{\Gamma, A, B, \Gamma' \longrightarrow \Delta}\ X_l \qquad \frac{\Gamma \longrightarrow \Delta, B, A, \Delta'}{\Gamma \longrightarrow \Delta, A, B, \Delta'}\ X_r$$

$$\underbrace{F_1, \ldots, F_n} \longrightarrow \underbrace{G_1, \ldots, G_m}$$

These contexts can be treated as sets of formulas.

We assume that the structural rules are applied implicitly.

Proof Theory Basics – Proofs

Proof Theory Basics – Proofs

Proofs are tree-like structures constructed using inference rules and whose leaves are instances of the **initial rule** and/or of the **false** rule.

Proof Theory Basics – Proofs

Proofs are tree-like structures constructed using inference rules and whose leaves are instances of the **initial rule** and/or of the **false** rule.

$$\dfrac{\dfrac{\dfrac{}{P \longrightarrow P}\,\text{Init} \quad \dfrac{}{Q \longrightarrow Q}\,\text{Init}}{P, Q \longrightarrow P \wedge Q}\,\wedge_r}{P \wedge Q \longrightarrow P \wedge Q}\,\wedge_l}{\cdot \longrightarrow (P \wedge Q) \supset (P \wedge Q)}\,\supset_r$$

Proof Theory Basics – Proofs

Proofs are tree-like structures constructed using inference rules and whose leaves are instances of the **initial rule** and/or of the **false** rule.

$$\dfrac{\dfrac{\dfrac{}{P \longrightarrow P}\,\text{Init} \quad \dfrac{}{Q \longrightarrow Q}\,\text{Init}}{P, Q \longrightarrow P \wedge Q}\,\wedge_r}{P \wedge Q \longrightarrow P \wedge Q}\,\wedge_l}{\cdot \longrightarrow (P \wedge Q) \supset (P \wedge Q)}\,\supset_r$$

Theorem: (Soudness and Completeness) A formula F is a tautology if and only if there is proof of $\longrightarrow F$.

Proof Theory Basics – Cut Elimination

Proof Theory Basics – Cut Elimination

$$\frac{\Gamma_1 \longrightarrow A, \Delta_1 \quad \Gamma_2, A \longrightarrow \Delta_2}{\Gamma_1, \Gamma_2 \longrightarrow \Delta_1, \Delta_2}\ \text{Cut}$$

Proof Theory Basics – Cut Elimination

This a new formula in the proof. Use of a **Lemma**.

$$\frac{\Gamma_1 \longrightarrow A, \Delta_1 \quad \Gamma_2, A \longrightarrow \Delta_2}{\Gamma_1, \Gamma_2 \longrightarrow \Delta_1, \Delta_2}\ \text{Cut}$$

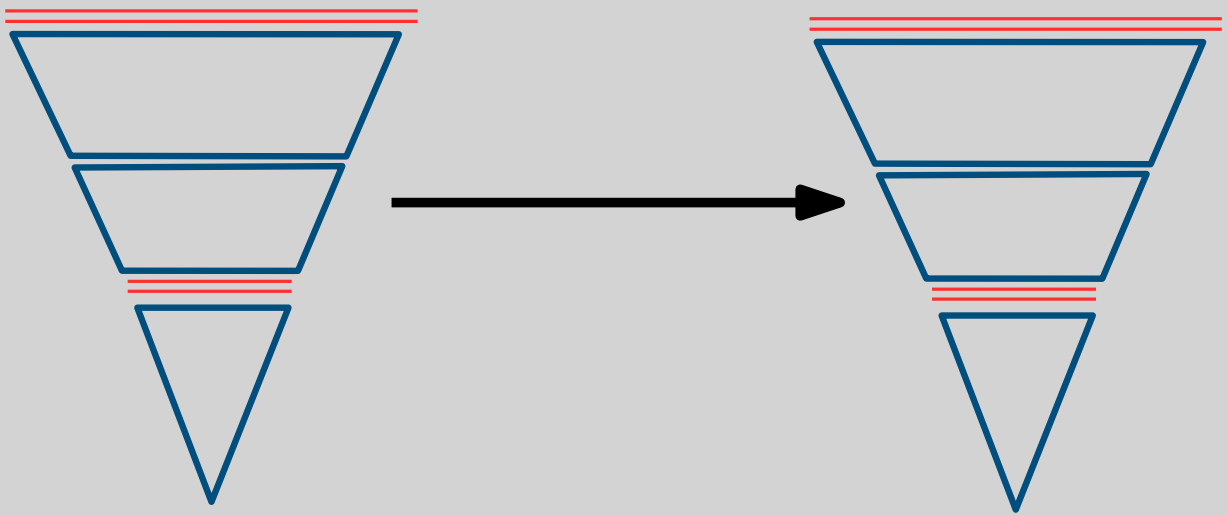
Proof Theory Basics – Cut Elimination

This a new formula in the proof. Use of a **Lemma**.

$$\frac{\Gamma_1 \longrightarrow A, \Delta_1 \quad \Gamma_2, A \longrightarrow \Delta_2}{\Gamma_1, \Gamma_2 \longrightarrow \Delta_1, \Delta_2}\ \text{Cut}$$

Proof with Cuts

Proof without Cuts

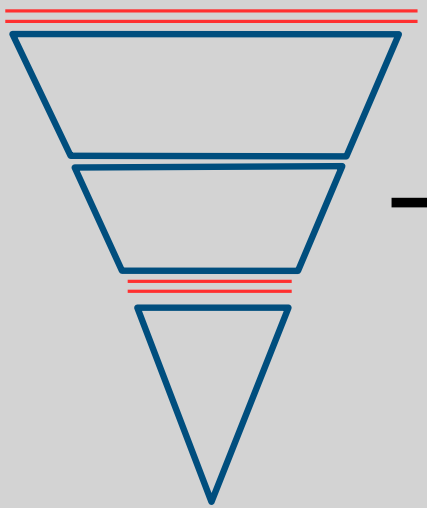


Proof Theory Basics – Cut Elimination

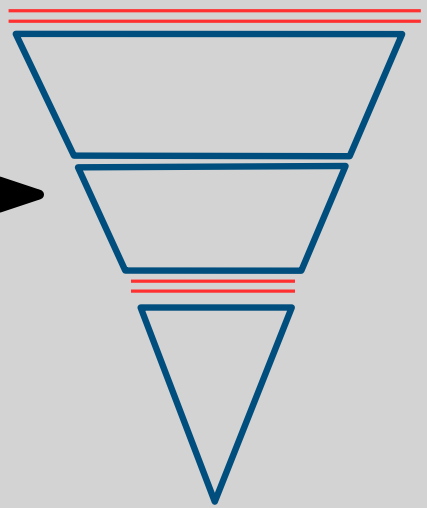
This a new formula in the proof. Use of a **Lemma**.

$$\frac{\Gamma_1 \longrightarrow A, \Delta_1 \quad \Gamma_2, A \longrightarrow \Delta_2}{\Gamma_1, \Gamma_2 \longrightarrow \Delta_1, \Delta_2}\ \text{Cut}$$

Proof with Cuts



Proof without Cuts



Consistency

$\longrightarrow A \qquad A \longrightarrow$

Sub-formula

No needs for **Lemmas**.

Logic Programming (Prolog)

Horn Clauses

Logic Programming (Prolog)

Horn Clauses

Goals (or queries)

$$G ::= true \mid A \mid G \wedge G \mid \exists x.G$$

Logic Programming (Prolog)

Horn Clauses

Goals (or queries)

$$G ::= true \mid A \mid G \wedge G \mid \exists x.G$$

Programs

$$P ::= A \mid G \supset A \mid P_1 \wedge P_2 \mid \forall x.P$$

Logic Programming (Prolog)

Horn Clauses

Goals (or queries)

$$G ::= true \mid A \mid G \wedge G \mid \exists x.G$$

Programs

$$P ::= A \mid G \supset A \mid P_1 \wedge P_2 \mid \forall x.P$$

Prove the sequent: $P \longrightarrow G$

$$\text{sum}([], 0) \wedge$$
$$\forall H, T, N, M.[\text{sum}(T, M) \wedge \text{eq}(N, M + H) \supset \text{sum}([H|T], N)]$$

$$\longrightarrow$$

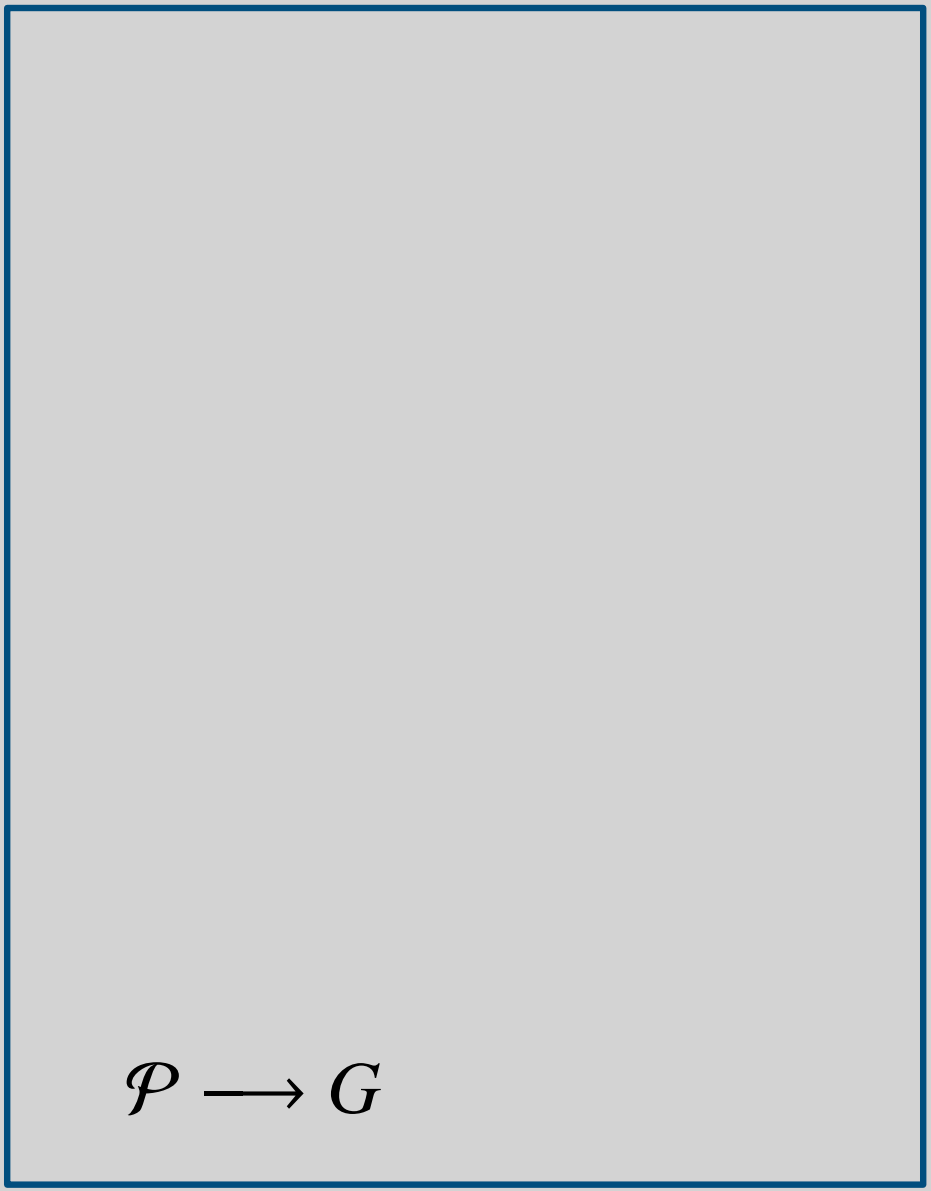
$$\exists X.\text{sum}([4, 2, 3, 4], X)$$

Logic Programming (Prolog)

Computation is Cut-free Proof Search

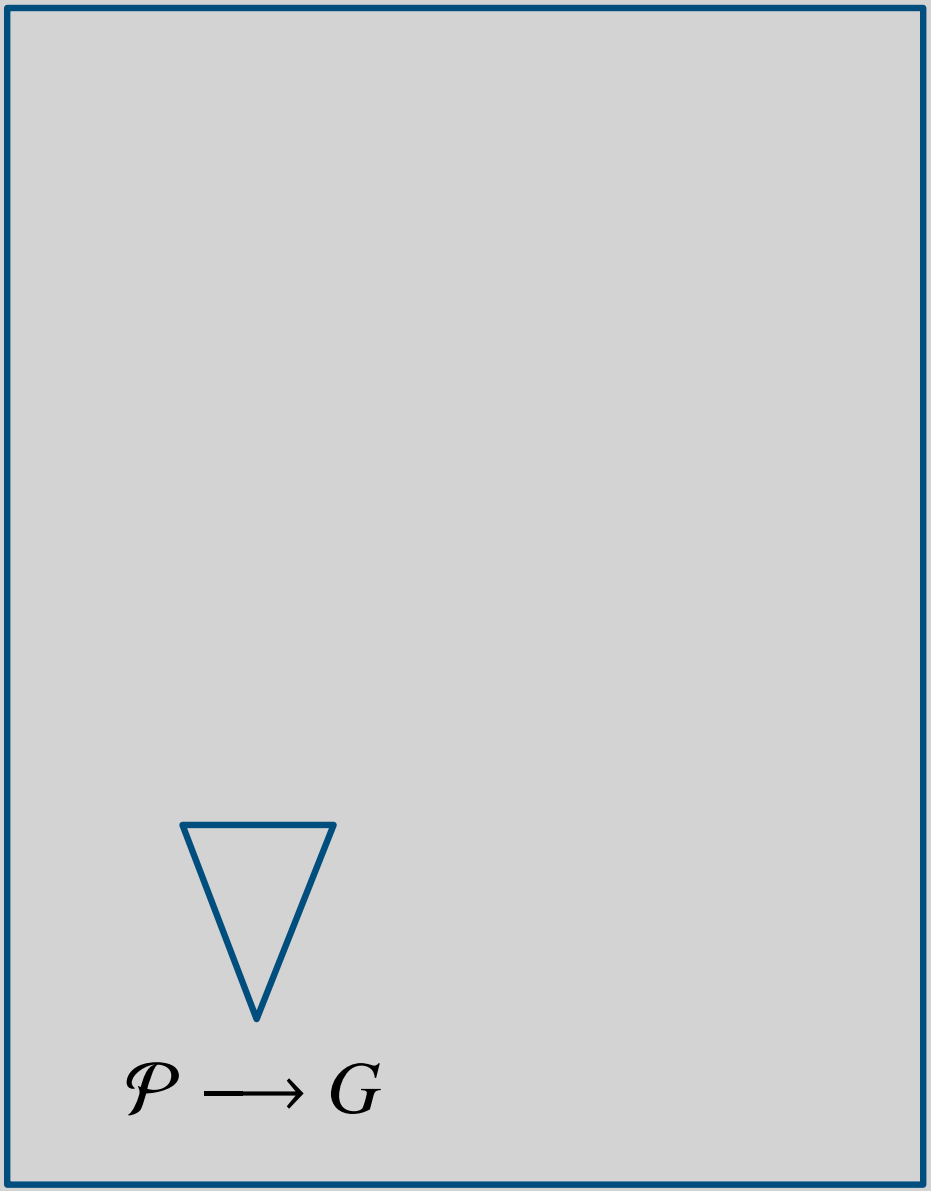
Logic Programming (Prolog)

Computation is Cut-free Proof Search



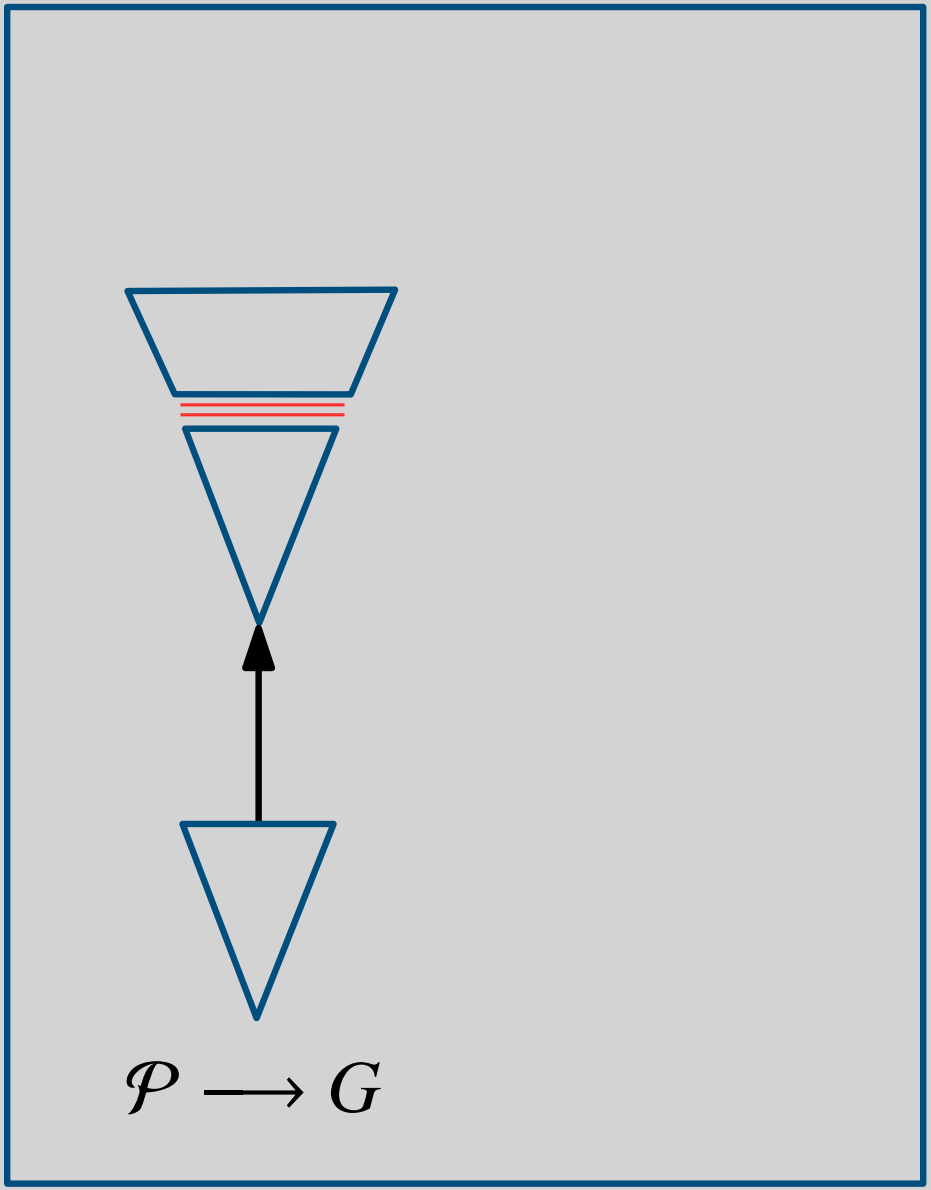
Logic Programming (Prolog)

Computation is Cut-free Proof Search



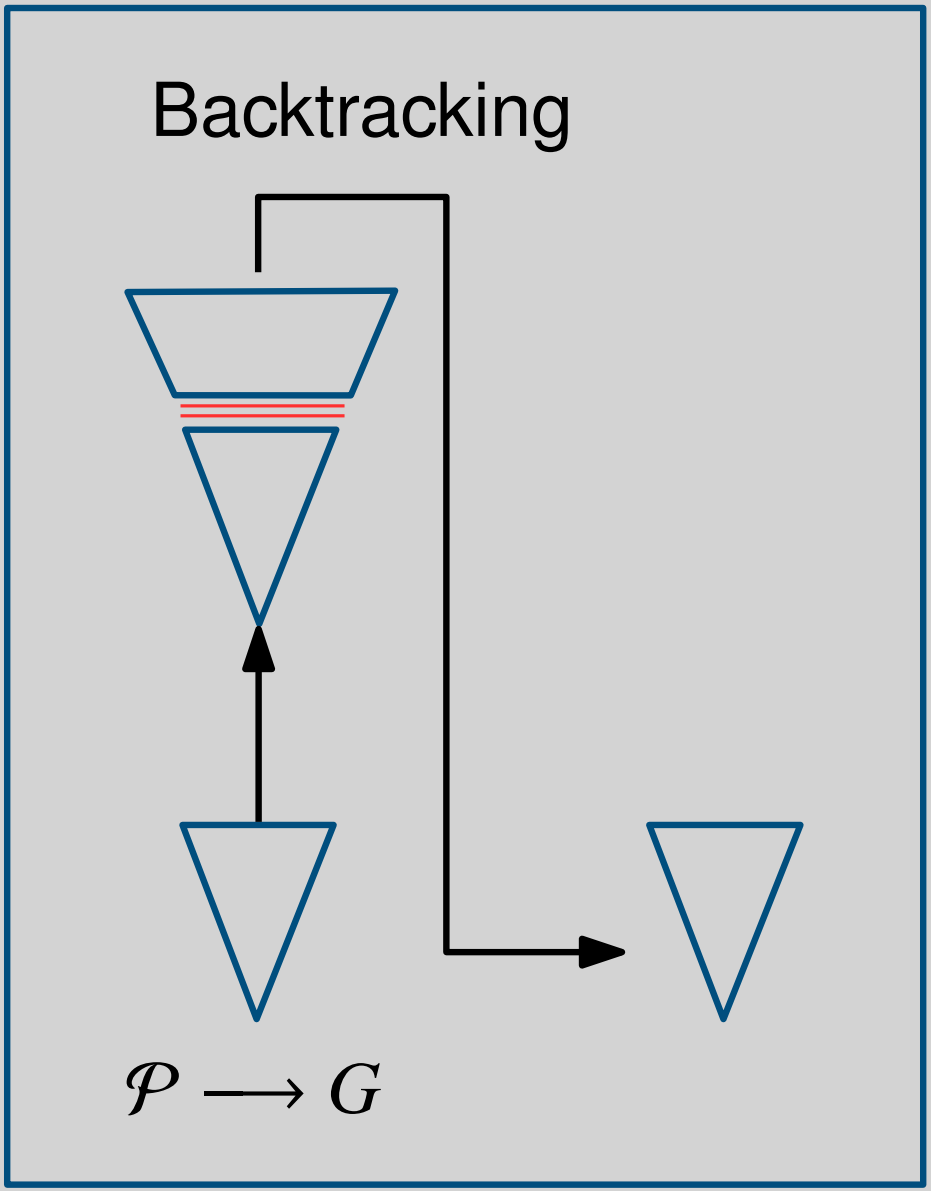
Logic Programming (Prolog)

Computation is Cut-free Proof Search



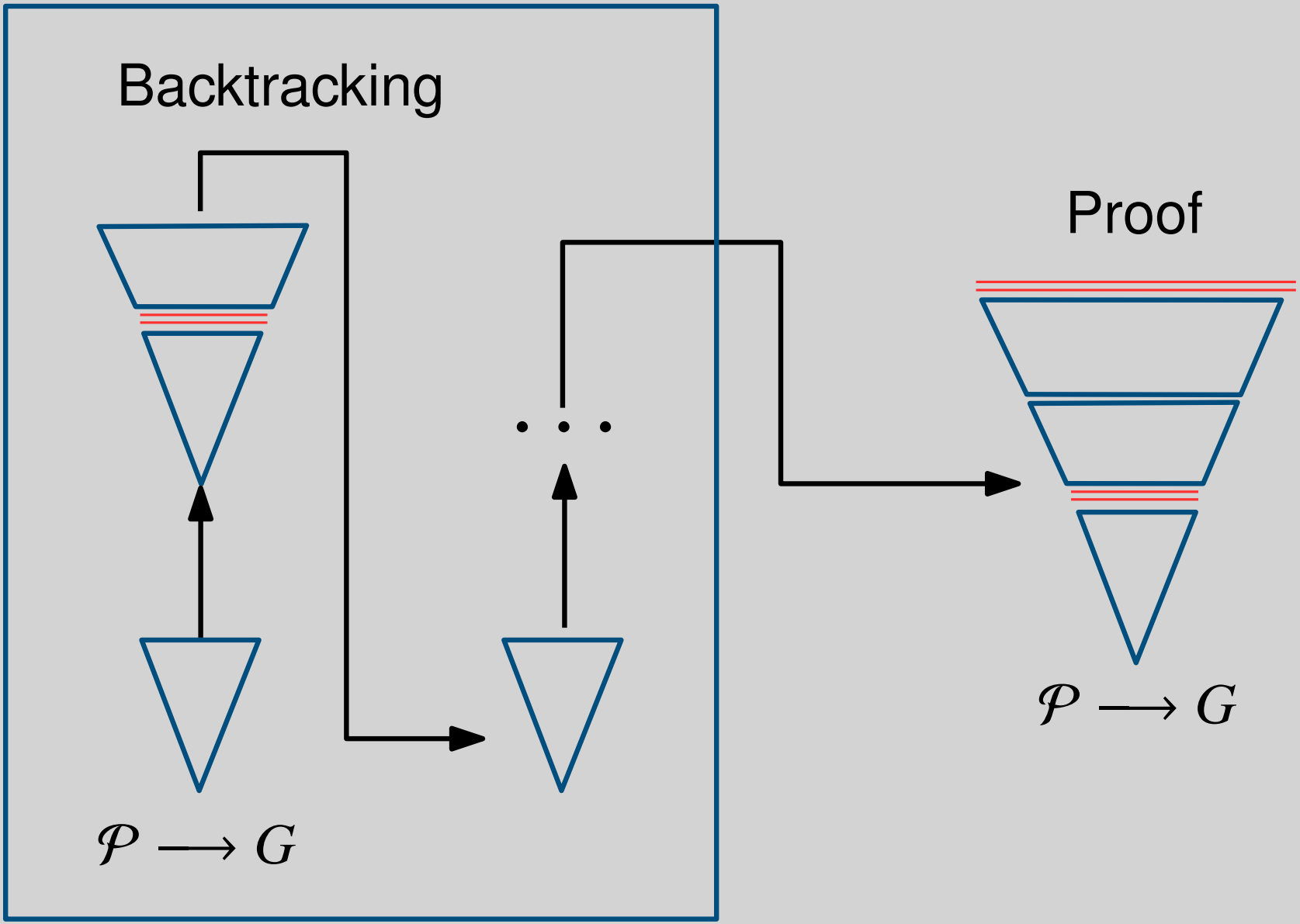
Logic Programming (Prolog)

Computation is Cut-free Proof Search



Logic Programming (Prolog)

Computation is Cut-free Proof Search



Logic Programming (Prolog)

$$G ::= true \mid A \mid G \wedge G \mid \exists x.G$$
$$P ::= A \mid G \supset A \mid P_1 \wedge P_2 \mid \forall x.P$$

No disjunction on the right of sequents.

Logic Programming (Prolog)

$G ::= true \mid A \mid G \wedge G \mid \exists x.G$

$P ::= A \mid G \supset A \mid P_1 \wedge P_2 \mid \forall x.P$

No disjunction on the right of sequents.

Single formula to the right-hand-side: $F_1, \ldots, F_n \longrightarrow G$

Logic Programming (Prolog)

$G ::= true \mid A \mid G \wedge G \mid \exists x.G$

$P ::= A \mid G \supset A \mid P_1 \wedge P_2 \mid \forall x.P$

No disjunction on the right of sequents.

Single formula to the right-hand-side: $F_1, \ldots, F_n \longrightarrow G$

Proof Theory – Sequent Calculus LJ

Logic Programming (Prolog)

$G ::= true \mid A \mid G \wedge G \mid \exists x.G$

$P ::= A \mid G \supset A \mid P_1 \wedge P_2 \mid \forall x.P$

No disjunction on the right of sequents.

Single formula to the right-hand-side: $F_1, \ldots, F_n \longrightarrow G$

Proof Theory – Sequent Calculus LJ

$$\frac{\Gamma, P, Q \longrightarrow G}{\Gamma, P \wedge Q \longrightarrow G} \wedge_l \qquad \frac{\Gamma \longrightarrow G_1 \quad \Gamma \longrightarrow G_2}{\Gamma \longrightarrow G_1 \wedge G_2} \wedge_r$$

$$\frac{\Gamma \longrightarrow P \qquad \Gamma, Q \longrightarrow G}{\Gamma, P \supset Q \longrightarrow G} \supset_l \qquad \frac{\Gamma \longrightarrow G\{t/x\}}{\Gamma \longrightarrow \exists x.G} \exists_r$$

$$\frac{\Gamma, P\{t/x\} \longrightarrow G}{\Gamma, \forall P \longrightarrow G} \forall_l \qquad \frac{}{A \longrightarrow A} \text{ Init}$$

Logic Programming (Prolog)

$G ::= true \mid A \mid G \wedge G \mid \exists x.G$

$P ::= A \mid G \supset A \mid P_1 \wedge P_2 \mid \forall x.P$

No disjunction on the right of sequents.

Single formula to the right-hand-side: $F_1, \ldots, F_n \longrightarrow G$

Proof Theory – Sequent Calculus LJ

$$\frac{\Gamma, P, Q \longrightarrow G}{\Gamma, P \wedge Q \longrightarrow G} \wedge_l \qquad \frac{\Gamma \longrightarrow G_1 \quad \Gamma \longrightarrow G_2}{\Gamma \longrightarrow G_1 \wedge G_2} \wedge_r$$

$$\frac{\Gamma \longrightarrow P \qquad \Gamma, Q \longrightarrow G}{\Gamma, P \supset Q \longrightarrow G} \supset_l \qquad \frac{\Gamma \longrightarrow G\{t/x\}}{\Gamma \longrightarrow \exists x.G} \exists_r$$

$$\frac{\Gamma, P\{t/x\} \longrightarrow G}{\Gamma, \forall P \longrightarrow G} \forall_l \qquad \frac{}{A \longrightarrow A} \text{ Init}$$

Searching a proof with the rules above is too inefficient! The search space is huge!

Logic Programming (Prolog)

$G ::= true \mid A \mid G \wedge G \mid \exists x.G$

$P ::= A \mid G \supset A \mid P_1 \wedge P_2 \mid \forall x.P$

Backchaining Rule

Logic Programming (Prolog)

$$G ::= true \mid A \mid G \wedge G \mid \exists x.G$$
$$P ::= A \mid G \supset A \mid P_1 \wedge P_2 \mid \forall x.P$$

Backchaining Rule

$$\frac{\Gamma, \forall \vec{x}.[G_1 \wedge \cdots \wedge G_n \supset A] \longrightarrow G_1\theta \quad \cdots \quad \Gamma, \forall \vec{x}.[G_1 \wedge \cdots \wedge G_n \supset A] \longrightarrow G_n\theta}{\Gamma, \forall \vec{x}.[G_1 \wedge \cdots \wedge G_n \supset A] \longrightarrow A_g}\ bc$$

where θ is the **most general unifier** of A and A_g.

Logic Programming (Prolog)

$$G ::= true \mid A \mid G \wedge G \mid \exists x.G$$
$$P ::= A \mid G \supset A \mid P_1 \wedge P_2 \mid \forall x.P$$

Backchaining Rule

$$\frac{\Gamma, \forall \vec{x}.[G_1 \wedge \cdots \wedge G_n \supset A] \longrightarrow G_1\theta \quad \cdots \quad \Gamma, \forall \vec{x}.[G_1 \wedge \cdots \wedge G_n \supset A] \longrightarrow G_n\theta}{\Gamma, \forall \vec{x}.[G_1 \wedge \cdots \wedge G_n \supset A] \longrightarrow A_g}\ bc$$

where θ is the **most general unifier** of A and A_g.

The backchaining rule yields a goal directed search.

$$\forall H, T, N, M.[\text{sum}(T, M) \wedge \text{eq}(N, M + H) \supset \text{sum}([H|T], N)]$$

$$\frac{\Gamma \longrightarrow \text{sum}(l, m) \qquad \Gamma \longrightarrow \text{eq}(m + 2, m + 2)}{\Gamma \longrightarrow \text{sum}([2 \mid l], m + 2)}\ bc$$

Logic Programming (Prolog)

$\mathsf{sum}([], 0) \wedge$

$\forall H, T, N, M.[\mathsf{sum}(T, M) \wedge \mathsf{eq}(N, M + H) \supset \mathsf{sum}([H|T], M)]$

$$\frac{\Gamma \longrightarrow \mathsf{sum}(l, m) \qquad \Gamma \longrightarrow \mathsf{eq}(m + 2, m + 2)}{\Gamma \longrightarrow \mathsf{sum}([2 \mid l], m + 2)}\ bc$$

Logic Programming (Prolog)

$\mathsf{sum}([], 0) \wedge$

$\forall H, T, N, M.[\mathsf{sum}(T, M) \wedge \mathsf{eq}(N, M + H) \supset \mathsf{sum}([H|T], M)]$

$$\frac{\Gamma \longrightarrow \mathsf{sum}(l, m) \qquad \Gamma \longrightarrow \mathsf{eq}(m + 2, m + 2)}{\Gamma \longrightarrow \mathsf{sum}([2 \mid l], m + 2)}\ bc$$

The backchaining rule can be interpreted as a collection of atomic rules.

$$\dfrac{\dfrac{}{\Gamma, \mathsf{sum}([2 \mid l], m + 2) \longrightarrow \mathsf{sum}([2 \mid l], m + 2)} \quad \dfrac{\Gamma \longrightarrow \mathsf{sum}(l, m) \quad \Gamma \longrightarrow \mathsf{eq}(m + 2, m + 2)}{\Gamma \longrightarrow \mathsf{sum}(l, m) \wedge \mathsf{eq}(m + 2, m + 2)}}{\dfrac{\Gamma, [\mathsf{sum}(l, m) \wedge \mathsf{eq}(m + 2, m + 2) \supset \mathsf{sum}([2 \mid l], m + 2)] \longrightarrow \mathsf{sum}([2 \mid l], m + 2)}{\Gamma \longrightarrow \mathsf{sum}([2 \mid l], m + 2)}}\ 4 \times \forall$$

Logic Programming (Prolog)

Question: Clearly proof search using the backchaining rule is sound, **but is it complete?**

Logic Programming (Prolog)

Question: Clearly proof search using the backchaining rule is sound, **but is it complete?**

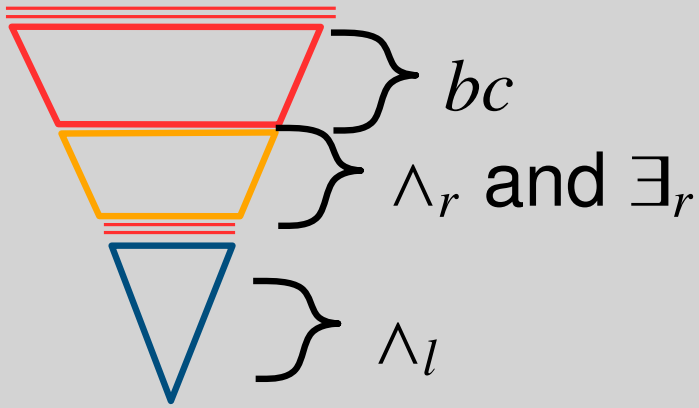
Theorem: Any LJ proof of a sequent of the form $P \longrightarrow G$ can be transformed into a proof that contains only occurrences of $\wedge_l$, bc, and $\exists_r$.

Logic Programming (Prolog)

Question: Clearly proof search using the backchaining rule is sound, **but is it complete?**

Theorem: Any LJ proof of a sequent of the form $P \longrightarrow G$ can be transformed into a proof that contains only occurrences of $\wedge_l$, bc, and $\exists_r$.

In fact, the proof of the theorem above tells a bit more. The set of proofs of the following shape is complete.

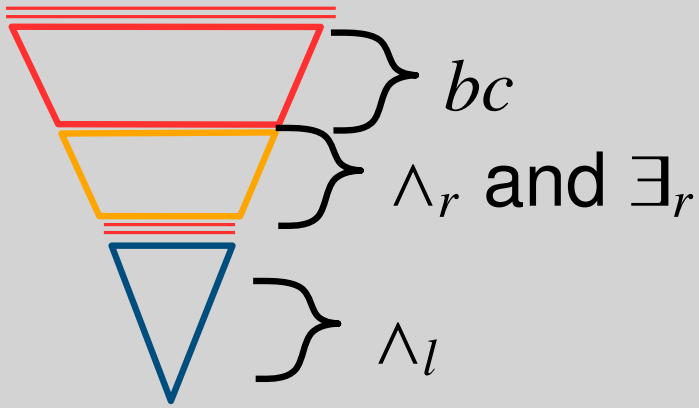


Logic Programming (Prolog)

Question: Clearly proof search using the backchaining rule is sound, **but is it complete?**

Theorem: Any LJ proof of a sequent of the form $P \longrightarrow G$ can be transformed into a proof that contains only occurrences of $\wedge_l$, bc, and $\exists_r$.

In fact, the proof of the theorem above tells a bit more. The set of proofs of the following shape is complete.



In literature, these proofs are called **uniform proofs**.

Logic Programming (Prolog)

Theorem: Any LJ proof of a sequent of the form $P \longrightarrow G$ can be transformed into a proof that contains only occurrences of $\wedge_l$, bc, and $\exists_r$.

Proof Sketch:

Logic Programming (Prolog)

Theorem: Any LJ proof of a sequent of the form $P \longrightarrow G$ can be transformed into a proof that contains only occurrences of $\wedge_l$, bc, and $\exists_r$.

Proof Sketch:

The proof relies in permutation lemmas:

Lemma 1: All rules permute over $\wedge_l$.

Logic Programming (Prolog)

Theorem: Any LJ proof of a sequent of the form $P \longrightarrow G$ can be transformed into a proof that contains only occurrences of $\wedge_l$, bc, and $\exists_r$.

Proof Sketch:

The proof relies in permutation lemmas:

Lemma 1: All rules permute over $\wedge_l$.

$$\dfrac{\dfrac{\Gamma, A, B \longrightarrow F}{\Gamma, A \wedge B \longrightarrow F}\,\wedge_l \quad \dfrac{\Gamma, G, A, B \longrightarrow R}{\Gamma, G, A \wedge B \longrightarrow R}\,\wedge_l}{\Gamma, F \supset G, A \wedge B \longrightarrow R}\,\supset_l$$

$$\dfrac{\Gamma, A, B \longrightarrow F \qquad \Gamma, G, A, B \longrightarrow R}{\dfrac{\Gamma, F \supset G, A, B \longrightarrow R}{\Gamma, F \supset G, A \wedge B \longrightarrow R}\,\wedge_l}\,\supset_l$$

Logic Programming (Prolog)

Theorem: Any LJ proof of a sequent of the form $P \longrightarrow G$ can be transformed into a proof that contains only occurrences of $\wedge_l$, bc, and $\exists_r$.

Proof Sketch:

The proof relies in permutation lemmas:

Lemma 1: All rules permute over $\wedge_l$.

Lemma 2: All other rules permute over each other.

Logic Programming (Prolog)

Theorem: Any LJ proof of a sequent of the form $P \longrightarrow G$ can be transformed into a proof that contains only occurrences of $\wedge_l$, bc, and $\exists_r$.

Proof Sketch:

The proof relies in permutation lemmas:

Lemma 1: All rules permute over $\wedge_l$.

Lemma 2: All other rules permute over each other.

$$\dfrac{\Gamma \longrightarrow F \qquad \dfrac{\Gamma, G \longrightarrow A \quad \Gamma, G \longrightarrow B}{\Gamma, G \longrightarrow A \wedge B}\ \wedge_r}{\Gamma, F \supset G \longrightarrow A \wedge B}\ \supset_l$$

$$\dfrac{\dfrac{\Gamma \longrightarrow F \quad \Gamma, G \longrightarrow B}{\Gamma, F \supset G \longrightarrow B}\ \supset_l \qquad \dfrac{\Gamma \longrightarrow F \quad \Gamma, G \longrightarrow A}{\Gamma, F \supset G \longrightarrow A}\ \supset_l}{\Gamma, F \supset G \longrightarrow A \wedge B}\ \wedge_r$$

Logic Programming (Prolog)

Proof Sketch:

Lemma 1: All rules permute over $\wedge_l$.

Lemma 2: All other rules permute over each other.

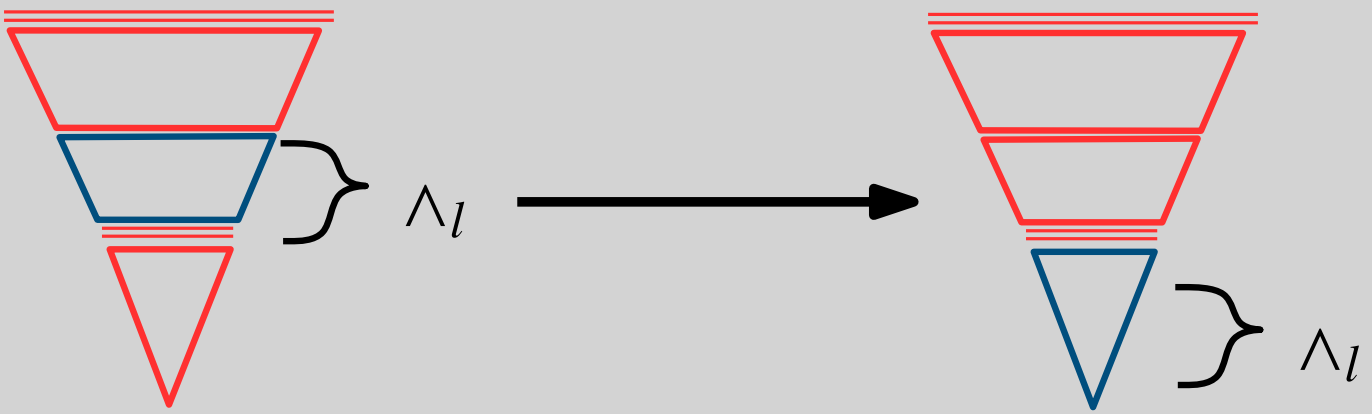
Logic Programming (Prolog)

Proof Sketch:

Lemma 1: All rules permute over $\wedge_l$.

Lemma 2: All other rules permute over each other.

Using Lemma 1, we can move all instances of $\wedge_l$ **downwards** in the proof.



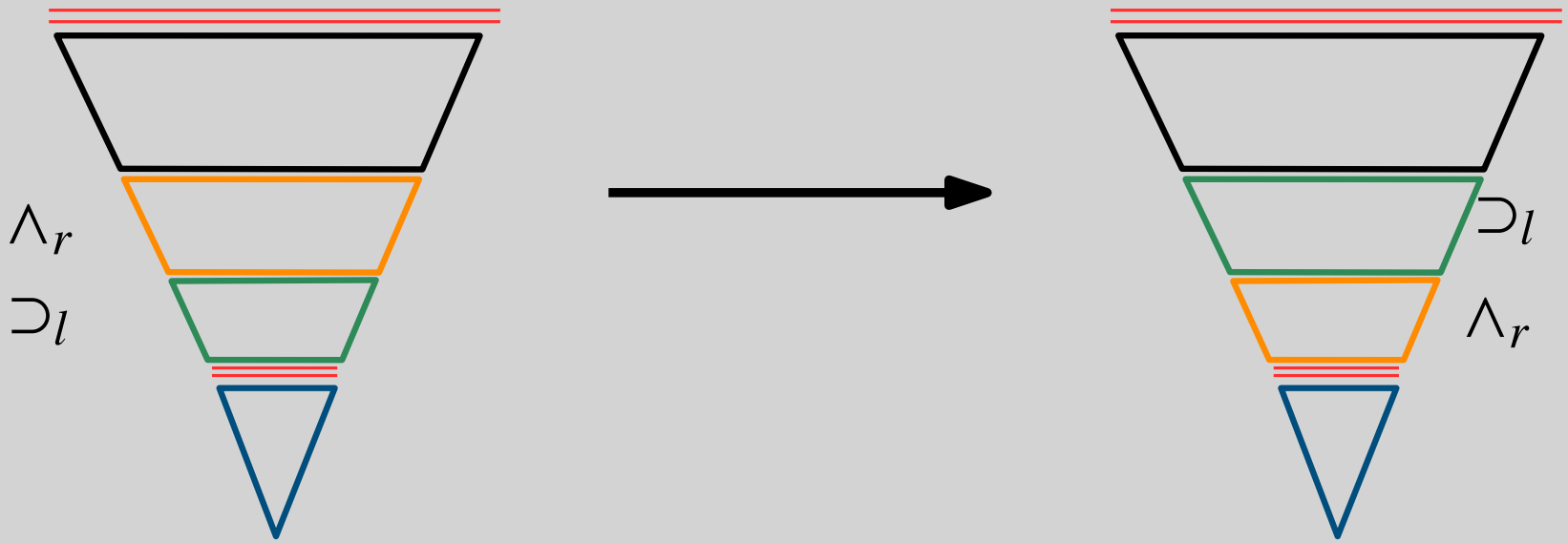
Logic Programming (Prolog)

Proof Sketch:

Lemma 1: All rules permute over $\wedge_l$.

Lemma 2: All other rules permute over each other.

Using Lemma 2, we can **organize** the remaining rules to form a backchaining rule instance.



Logic Programming (Prolog)

How about the witnesses for existentials?

Logic Programming (Prolog)

How about the witnesses for existentials?

$\text{sum}([], 0) \wedge$

$\forall H, T, N, M.[\text{sum}(T, M) \wedge \text{eq}(N, M + H) \supset \text{sum}([H|T], M)]$

$\longrightarrow$

$\exists X.\text{sum}([4, 2, 3, 4], X)$

Logic Programming (Prolog)

How about the witnesses for existentials?

$\text{sum}([], 0) \wedge$
$\forall H, T, N, M.[\text{sum}(T, M) \wedge \text{eq}(N, M + H) \supset \text{sum}([H|T], M)]$

$$\longrightarrow$$

$$\exists X.\text{sum}([4, 2, 3, 4], X)$$

Unification **delays the search for witnesses** for the $\exists_r$ rule.

$\text{sum}([], 0) \wedge$
$\forall H, T, N, M.[\text{sum}(T, M) \wedge \text{eq}(N, M + H) \supset \text{sum}([H|T], M)]$

$$\longrightarrow$$

$$\text{sum}([4, 2, 3, 4], \_\mathbf{X})$$

where **_X** is a **logical variable** created by the logic interpreter (Prolog engine). That is, this is a strictly implementation technique. There is no correspondence to logic.

Logic Programming (Prolog)

There is still one decision to make: which clause to backchain?

Logic Programming (Prolog)

There is still one decision to make: which clause to backchain?

$$\forall X.p(f(X)) \supset p(X)$$

$$\frac{\Gamma, \forall X.[p(f(X) \supset p(X)] \longrightarrow p(f(a))}{\Gamma, \forall X.[p(f(X) \supset p(X)] \longrightarrow p(a)}\ bc$$

Logic Programming (Prolog)

There is still one decision to make: which clause to backchain?

$$\forall X.p(f(X)) \supset p(X)$$

$$\frac{\Gamma, \forall X.[p(f(X) \supset p(X)] \longrightarrow p(f(a))}{\Gamma, \forall X.[p(f(X) \supset p(X)] \longrightarrow p(a)}\ bc$$

If the interpreter always backchains on the clause above, then it will **not terminate**.

Verifying Protocols with Prolog

We model the intruder as well as the protocol by using Horn clauses.

As before, we assume that there are constructors and destructors.

For each **constructor** f, we add a rule of the form:

$$\text{att}(M_1) \wedge \cdots \wedge \text{att}(M_n) \supset \text{att}(f(M_1, \ldots, M_n))$$

Examples

$\text{att}(m) \wedge \text{att}(pk) \supset \text{att}(\text{aenc}(m, pk))$

$\text{att}(sk) \supset \text{att}(\text{pk}(sk))$

$\text{att}(m) \wedge \text{att}(sk) \supset \text{att}(\text{sign}(m, sk))$

Verifying Protocols with Prolog

For each **destructor** defined by $g(M_1, \ldots, M_n) = M$, we add a rule of the form:

$$\text{att}(M_1) \wedge \cdots \wedge \text{att}(M_n) \supset \text{att}(M)$$

$\text{att}(\text{aenc}(m, \text{pk}(sk))) \wedge \text{att}(sk) \supset \text{att}(m)$

$\text{att}(\text{sign}(m, sk)) \supset \text{att}(m)$

Destructors never appear in the rules. They are modelled using **pattern-matching**.

Verifying Protocols with Prolog

Specification of a Protocol (by example)

$A \rightarrow B : \{\text{sign}(k, sk_A)\}_{pk_B}$

The intruder starts protocols with anyone he wants, or alternatively it can intercept A's message faking to be B.

$\text{att}(\text{pk}(k)) \supset \text{att}(\text{aenc}(\text{sign}(k, sk_A[]), \text{pk}(k))$

However, keys are fresh!

$\text{att}(\text{pk}(k)) \supset \text{att}(\text{aenc}(\text{sign}(k[\text{pk(x)}], sk_A[]), \text{pk}(k))$

This is the first abstraction that Proverif uses. Fresh values are considered as **functions** of the messages previously received by the protocol.

Verifying Protocols with Prolog

Specification of a Protocol (by example)

Once B receives the message of the form $\{\text{sign}(k', sk)\}_{pk_B}$ she cheks whether the signature is correct, that is, $sk = sk_A[]$:

$$B \rightarrow A : \{s[]\}_{k'}$$

In our model, the attacker does all this:

$$\text{att}(\text{enc}(\text{sign}(k', sk_A[])), \text{pk}(sk_B)) \supset \text{att}(\text{aenc}(s[], k'))$$

A symmetric rule is added for when B starts the protocol.

In general, a protocol that contains n messages encoded by n sets of rules, each set containing the same corresponding rule for each principal.

Verifying Protocols with Prolog

Reachability Secrecy Problem

Given a logic program P encoding a protocol containing a secret s, is the fact att(s[]) derivable from P.

Notice that since we are in **classical logic**, the number of times a message is used is not evident.

$$\text{att}(\text{enc}(s[], k))$$

In a derivation, this message can be derived as many times needed. This is because formulas can be contracted. Hence, messages meant to appear only once in a future protocol may **mix** up with messages in the past. This may lead to **false attacks**.

Verifying Protocols with Prolog

Problem with Prolog Proof Search

Using the backchaining rule with the rule below would lead to **non-termination**.

$$\text{att}(\text{enc}(m, \text{pk}(sk))) \wedge \text{att}(sk) \supset \text{att}(m)$$

We need another solving algorithm!

Verifying Protocols with Prolog

Main Idea

The solving algorithm works in two phases:

In the first phase, we **pre-process** the logic program by constructing rewriting rules so that the body of all rules of the form $\mathrm{att}(x)$.

The second phase is to use the **same Prolog search strategy** but using the pre-processed program.

Verifying Protocols with Prolog

Phase One

We want that the body of all clauses are of the form att(x).

Selection function: sel($F_1 \wedge \cdots \wedge F_n \supset F$) $\subseteq \{F_1, \ldots, F_n\}$

sel($F_1 \wedge \cdots \wedge F_n \supset F$) $= \emptyset$ if $F_i =$ att(x) for all $1 \leq i \leq n$.

sel($F_1 \wedge \cdots \wedge F_n \supset F$) $= \{F_i\}$ otherwise, where F_i has the greatest size.

Example

sel(att(enc(m, pk(sk))) $\wedge$ att(sk) $\supset$ att(m)) = att(enc(m, pk(sk)))

Verifying Protocols with Prolog

Phase One – Resolution

$R = F_1 \wedge \cdots \wedge F_n \supset F$ $\quad$ $R' = F'_1 \wedge \cdots \wedge F'_{n'} \supset F'$

$\text{sel}(R) = \emptyset$ $\quad$ $F'_i \in \text{sel}(R')$

Add to the program the following clause, denoted as $R \circ_{F'_i} R'$

$$F_1\theta \wedge \cdots \wedge F_n\theta \wedge F'_1\theta \wedge \cdots F'_{i-1}\theta \wedge F'_{i+1}\theta \wedge \cdots \wedge F'_{n'}\theta \supset F'\theta$$

This operation can be seen as a **forward-chaining step**.

Example

$\text{att}(\text{aenc}(\text{sign}(z, sk_A[]), pk_B[])) \supset \text{att}(\text{enc}(s, z))$

$\text{att}(m) \wedge \text{att}(pk) \supset \text{att}(\text{enc}(m, pk))$

$\text{att}(\text{sign}(z, sk_A[])) \wedge \text{att}(pk_B[]) \supset \text{att}(\text{enc}(s, z))$