

Relative Safety

Joxan Jaffar, Andrew Santosa, Răzvan Voicu

{joxan, andrews, razvan}@comp.nus.edu.sg

School of Computing, National University of Singapore

Relative Safety

- A new kind of assertion:
Specifies **non-behavioral/structural properties** of programs
... in addition to traditional safety
- Purpose: Reduction
- Driving application: Symmetry
- A **PROOF METHOD** for relative safety using **COINDUCTION**
... reduction using relative safety

Relative Safety

Program m : 2-process mutex, $\langle 2 \rangle = \text{CRITICAL SECTION}$

```

while (true) do
   $\langle 0 \rangle$     await ( $x2 \neq 1$ )  $x1 := 1$ 
   $\langle 1 \rangle$     await ( $x2 \neq 2$ )  $x1 := 2$  ||
   $\langle 2 \rangle$      $x1 := 0$ 
end

while (true) do
   $\langle 0 \rangle$      $x2 := 1$ 
   $\langle 1 \rangle$     await ( $x1 = 0$ )  $x2 := 2$ 
   $\langle 2 \rangle$      $x2 := 0$ 
end
```

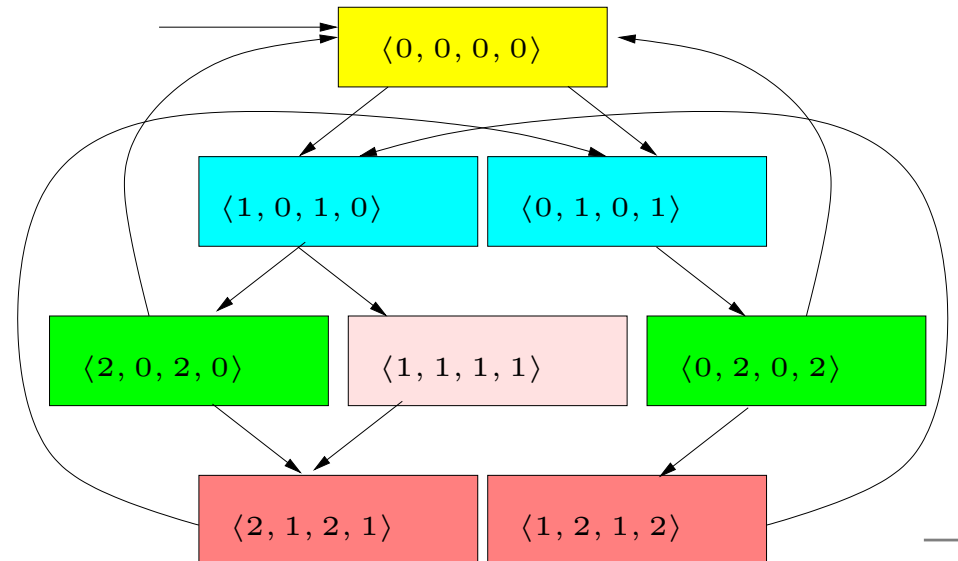
Relative Safety

Program m : 2-process mutex, $\langle 2 \rangle = \text{CRITICAL SECTION}$

```
while (true) do
   $\langle 0 \rangle$    await (x2  $\neq$  1) x1 := 1
   $\langle 1 \rangle$    await (x2  $\neq$  2) x1 := 2
   $\langle 2 \rangle$    x1 := 0
end
```

```
while (true) do
   $\langle 0 \rangle$    x2 := 1
   $\langle 1 \rangle$    await (x1 = 0) x2 := 2
   $\langle 2 \rangle$    x2 := 0
end
```

A state = $\langle P_1, P_2, X_1, X_2 \rangle$
 P_1, P_2 = program points of process 1, 2
 X_1, X_2 = values of x_1, x_2



Relative Safety

In program m :

State $\langle P_1, P_2, X_1, X_2 \rangle$ is **REACHABLE ONLY IF** state $\langle P_2, P_1, X_2, X_1 \rangle$ is

$\langle 1, 0, 1, 0 \rangle$ is **REACHABLE ONLY IF** $\langle 0, 1, 0, 1 \rangle$ is

$\langle 2, 2, 0, 1 \rangle$ is **REACHABLE ONLY IF** $\langle 2, 2, 1, 0 \rangle$ is

(although $\langle 2, 2, 0, 1 \rangle$ is not reachable)

Relative Safety

In program m :

State $\langle P_1, P_2, X_1, X_2 \rangle$ is **REACHABLE ONLY IF** state $\langle P_2, P_1, X_2, X_1 \rangle$ is

$\langle 1, 0, 1, 0 \rangle$ is **REACHABLE ONLY IF** $\langle 0, 1, 0, 1 \rangle$ is

$\langle 2, 2, 0, 1 \rangle$ is **REACHABLE ONLY IF** $\langle 2, 2, 1, 0 \rangle$ is

(although $\langle 2, 2, 0, 1 \rangle$ is not reachable)

SYMMETRY

Relative Safety

Program c :

$$\begin{array}{l} \vdots \\ \langle 5 \rangle \quad x := x+3 \\ \vdots \end{array} \quad || \quad \begin{array}{l} \vdots \\ \langle 11 \rangle \quad x := x-5 \\ \vdots \end{array}$$
$$\begin{array}{l} \langle 6 \rangle \quad \vdots \\ \vdots \end{array} \quad \begin{array}{l} \langle 12 \rangle \quad \vdots \\ \vdots \end{array}$$

State $\langle 6, 12, X \rangle$ is **REACHABLE ONLY IF** state $\langle 5, 11, X + 2 \rangle$ is
(and vice versa)

Representing Program in CLP

```
Program m:      while (true) do
                  <0>      await (x2≠1) x1 := 1
                  <1>      await (x2≠2) x1 := 2
                  <2>      x1 := 0
                  end
```

CLP RULES:

TRANSITIONS: $m(P_1, P_2, X_1, X_2) \leftarrow m(P'_1, P_2, X'_1, X_2),$
 $P_1 = 1, X_1 = 1, P'_1 = 0, X_2 \neq 1.$

INITIAL STATE: $m(P_1, P_2, X_1, X_2) \leftarrow P_1 = P_2 = 0, X_1 = X_2 = 0.$

Collecting semantics is characterized by all valuations σ of P_1, P_2, X_1, X_2 such that $m(P_1\sigma, P_2\sigma, X_1\sigma, X_2\sigma)$ is true

Relative Safety in CLP

● **SAFETY:** $p(\tilde{X}), \Psi_1(\tilde{X}) \models \Psi_2(\tilde{X})$

EXAMPLE: $p(X, Y) \models X + Y < 10$

Relative Safety in CLP

● **SAFETY:** $p(\tilde{X}), \Psi_1(\tilde{X}) \models \Psi_2(\tilde{X})$

EXAMPLE: $p(X, Y) \models X + Y < 10$

● **RELATIVE SAFETY:** $p(\tilde{X}), \Psi_1(\tilde{X}) \models p(\tilde{X}'), \Psi_2(\tilde{X}, \tilde{X}')$

● **SYMMETRY:** $m(P_1, P_2, X_1, X_2) \models m(P_2, P_1, X_2, X_1)$

● **COMMUTATIVITY:** $c(6, 12, X) \models c(5, 11, X + 2)$

Another Example

SERIALIZABILITY

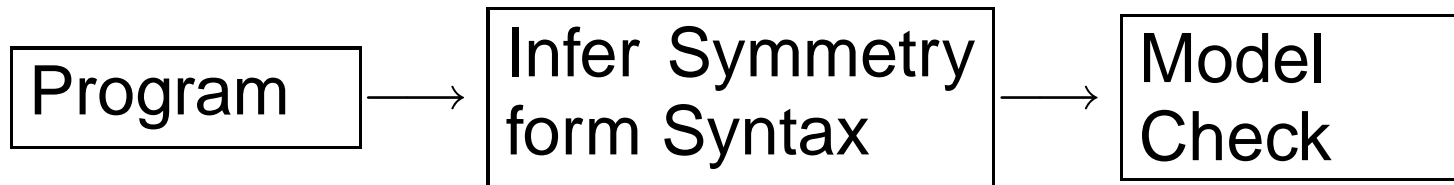
```
while (true) do
  <0>   ...
  <1>   x := f(x)      ||
  <2>   x := g(x) <3>
end

while (true) do
  ... // No x
end
```

$s(3, P_2, g(f(X))) \models s(1, P_2, X)$.

Related Work

Previous approaches are **SYNTACTIC**:



● Scalarset [Ip and Dill 1993]

```
Pid: scalarset [5]
i : Pid
...
j := 0
while (j < i) do
    await (x[j] < 2) j := j+1
end
```

Scalarset variables may only be compared using =

Related Work

- Identical processes, e.g., SMC [Sistla et al. 2000], RED [Wang 1997]

Our program m is syntactically asymmetric

- PSMC [Sistla and Godefroid 2004]

Attach $\text{Priority}(X_1; \dots; X_n)$ to transition schema, where X_1, \dots, X_n disjoint subsets of process ids

await $(\forall j < i : x[j] < 2)$

Processes j prioritized only when $x[j] \geq 2$

Our Approach

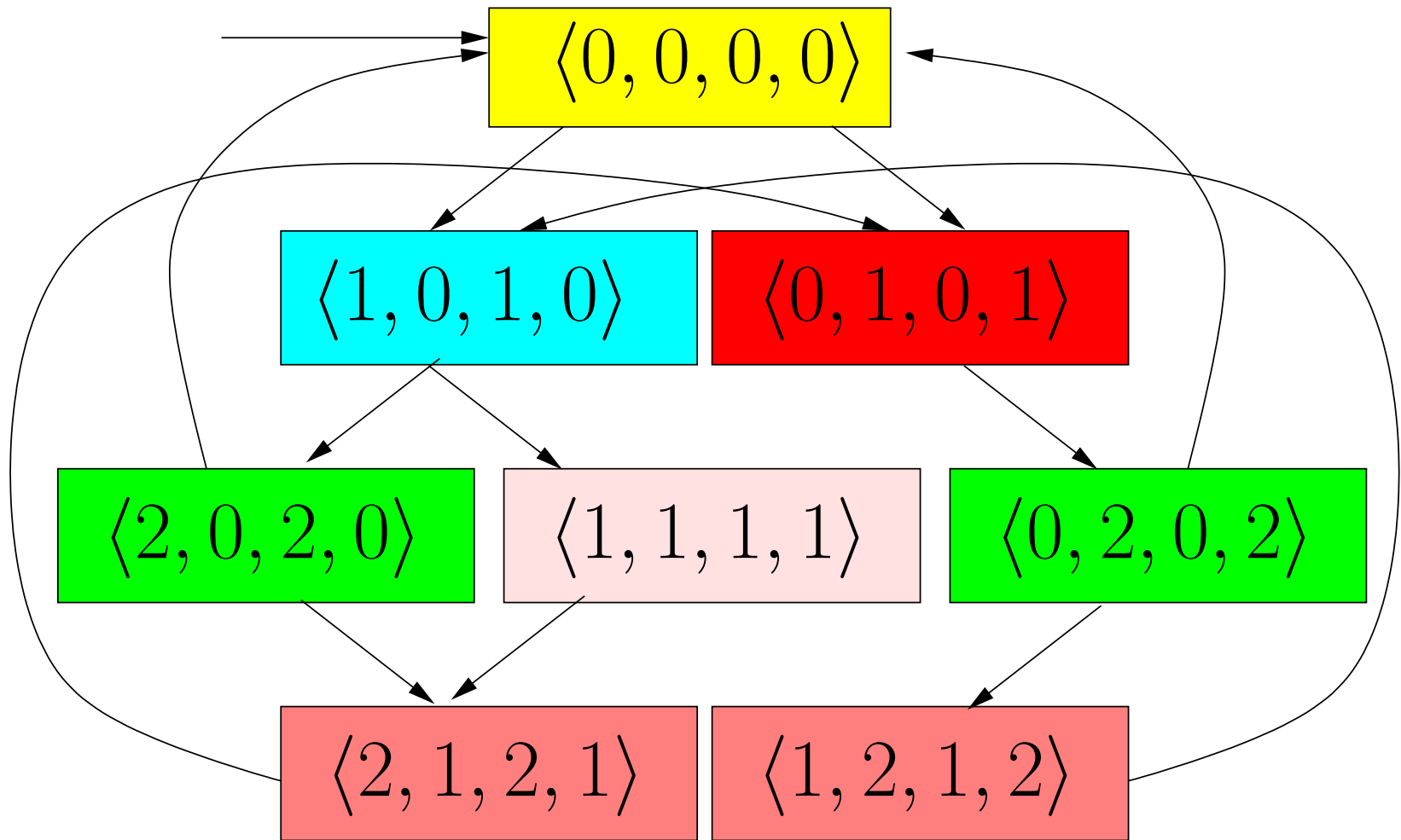
SEMANTIC:



Advantage: More flexible (e.g., Szymanski's algorithm).

PLUS: Can handle infinite-state systems

Automorphism of State Graph



[Emerson and Sistla 1993] [Ip and Dill 1993], [Sistla and Godefroid 2004], [Emerson et al. 2000] **preserves TL**

Automorphism on Collecting Semantics

$\langle 0, 0, 0, 0 \rangle$

$\langle 1, 0, 1, 0 \rangle$

$\langle 0, 1, 0, 1 \rangle$

$\langle 2, 0, 2, 0 \rangle$

$\langle 1, 1, 1, 1 \rangle$

$\langle 0, 2, 0, 2 \rangle$

$\langle 2, 1, 2, 1 \rangle$

$\langle 1, 2, 1, 2 \rangle$

$$m(P_1, P_2, X_1, X_2) \models m(P_2, P_1, X_2, X_1)$$

*An automorphism group on the collecting semantics.
preserves safety*

Proof Method

- Proof tree generation using **UNFOLD** operation
- Termination of proof process by **COINDUCTION** and **DIRECT PROOF**
- **RIGHT UNFOLD** for proving relative safety

Coinduction

Use original assertion to prove other assertions
Proof holds as long as no contradiction

$$p(X) \leftarrow X = 0.$$

$$p(X) \leftarrow p(X'), X = X' + 2.$$

$$p(X) \models \text{even}(X)$$

Unfold (LU+C)

$$X = 0 \models \text{even}(X)$$

Coinduction

Use original assertion to prove other assertions
Proof holds as long as no contradiction

$$p(X) \leftarrow X = 0.$$

$$p(X) \leftarrow p(X'), X = X' + 2.$$

$$p(X) \models \text{even}(X)$$

Unfold (LU+C)

$$X = 0 \models \text{even}(X)$$

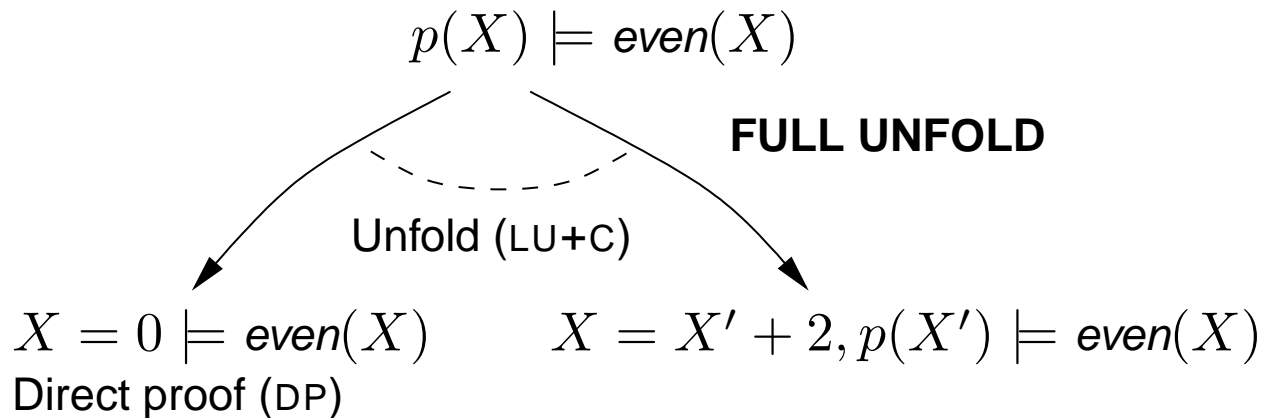
Direct proof (DP)

Coinduction

Use original assertion to prove other assertions
Proof holds as long as no contradiction

$$p(X) \leftarrow X = 0.$$

$$p(X) \leftarrow p(X'), X = X' + 2.$$

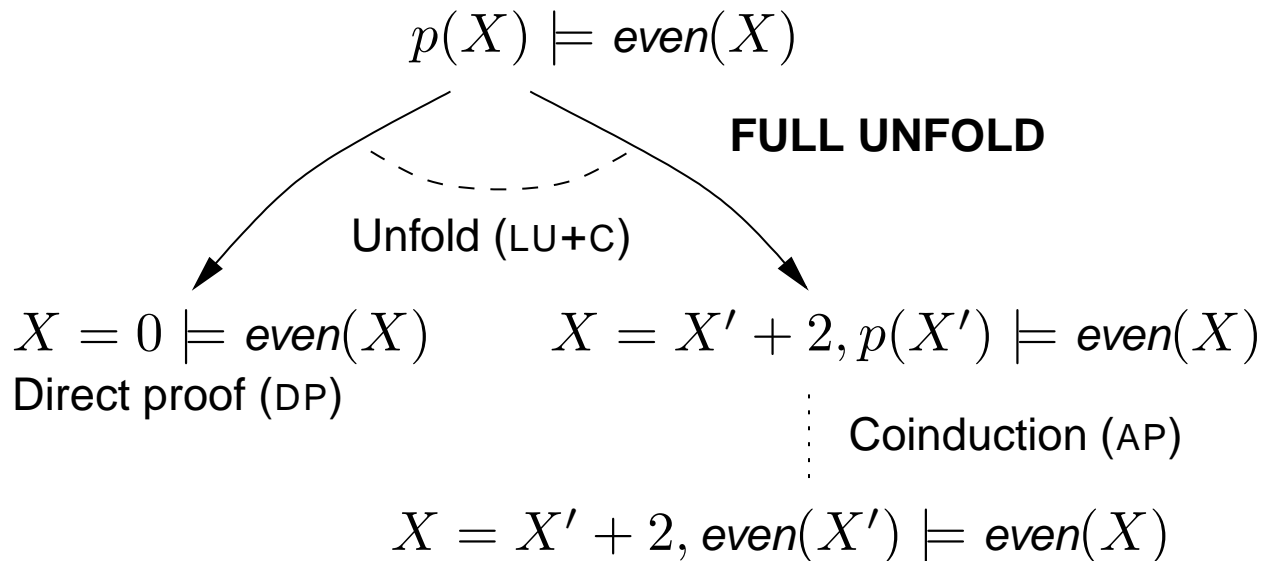


Coinduction

Use original assertion to prove other assertions
Proof holds as long as no contradiction

$$p(X) \leftarrow X = 0.$$

$$p(X) \leftarrow p(X'), X = X' + 2.$$

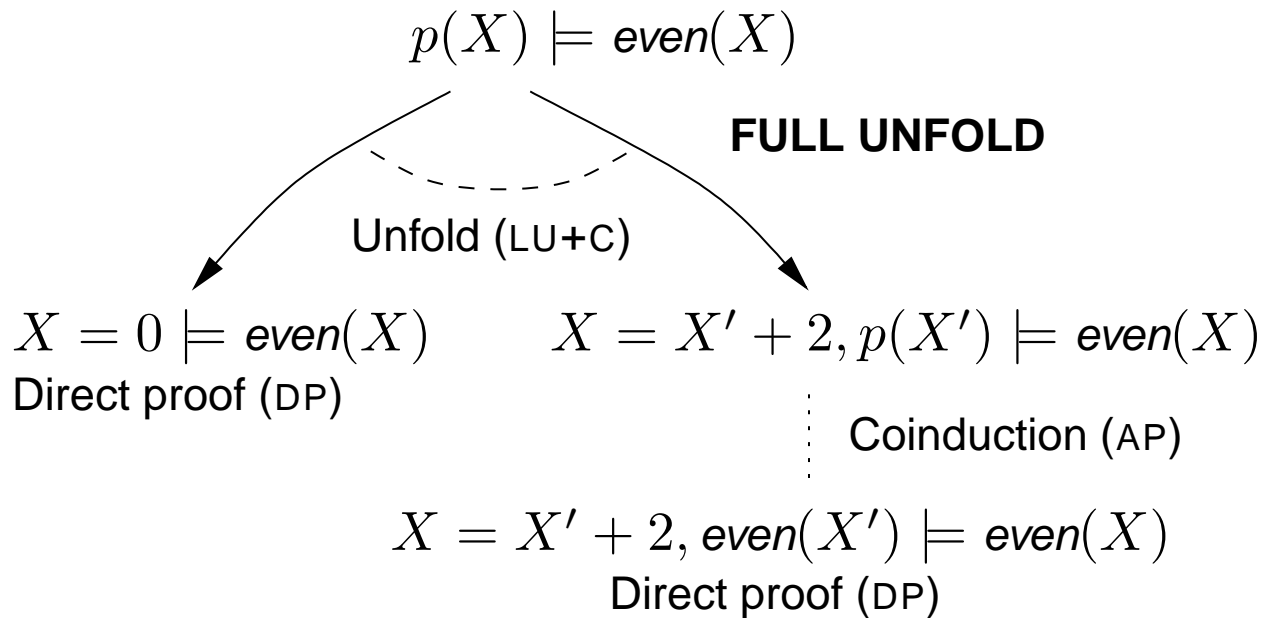


Coinduction

Use original assertion to prove other assertions
Proof holds as long as no contradiction

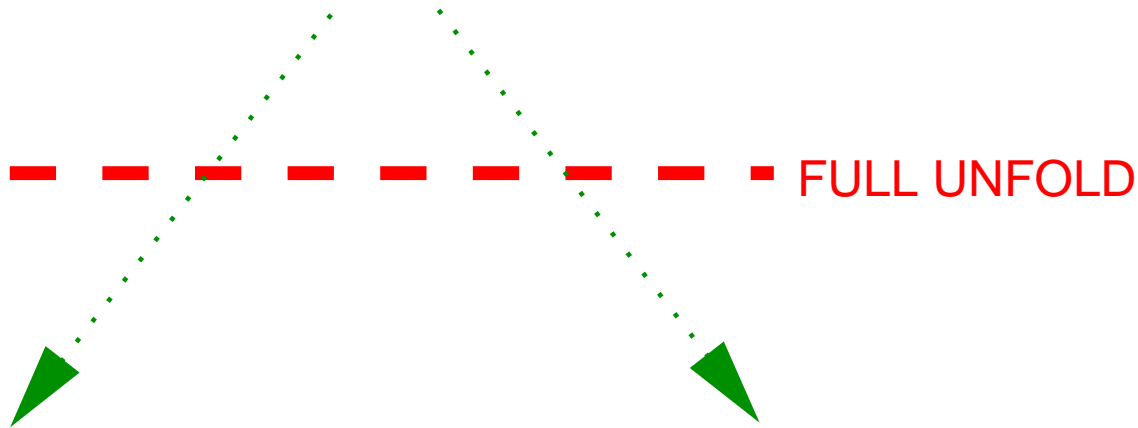
$$p(X) \leftarrow X = 0.$$

$$p(X) \leftarrow p(X'), X = X' + 2.$$



Using Relative Safety

$bak(P_1, P_2, T_1, T_2), P_1 = P_2 = 2 \models \mathbf{false}$



(A)

$bak(P_1'', P_2', T_1, T_2),$
 $P_1'' = 0, P_2' = 1, T_2 = 0 \models \mathbf{false}$

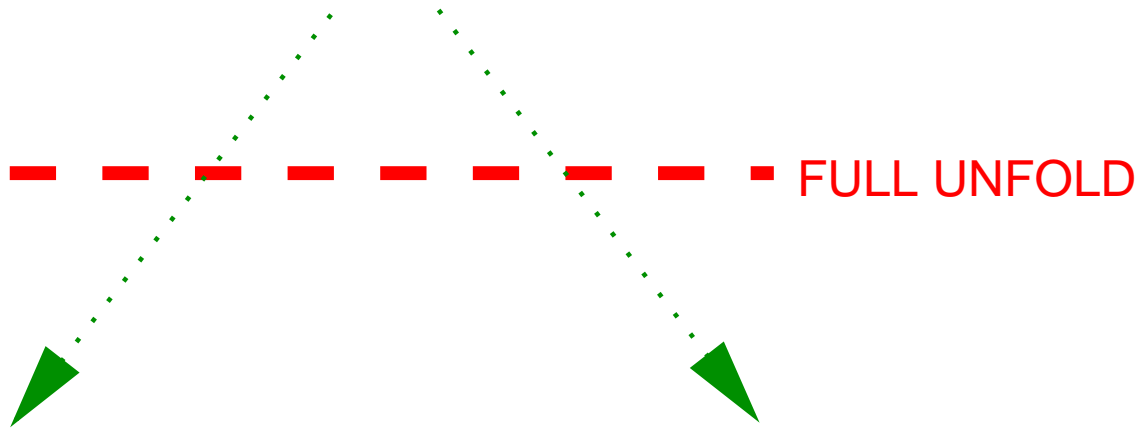
**BIG
SUBTREE**

$bak(P_1', P_2''', T_1, T_2'),$
 $P_1' = 1, P_2''' = 0, T_1 = 0 \models \mathbf{false}$

**BIG
SUBTREE**

Using Relative Safety

$bak(P_1, P_2, T_1, T_2), P_1 = P_2 = 2 \models \mathbf{false}$



(A)

$bak(P_1'', P_2', T_1, T_2),$
 $P_1'' = 0, P_2' = 1, T_2 = 0 \models \mathbf{false}$

**BIG
SUBTREE**

$bak(P_1', P_2''', T_1, T_2'),$
 $P_1' = 1, P_2''' = 0, T_1 = 0 \models \mathbf{false}$

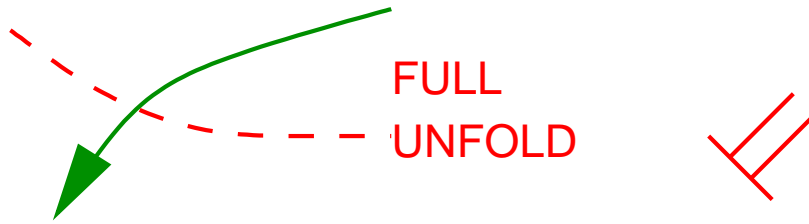
Using RS: $bak(P_1, P_2, T_1, T_2) \models$
 $bak(P_2, P_1, T_2, T_1)$

$bak(P_2''', P_1', T_2', T_1),$
 $P_1' = 1, P_2''' = 0, T_1 = 0 \models \mathbf{false}$

Redundant to **(B)**

Relative Safety Proof

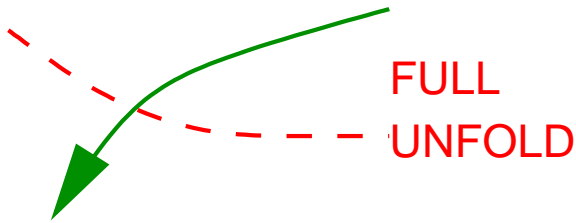
$$m(P_1, P_2, X_1, X_2) \models m(P_2, P_1, X_2, X_1)$$



$$m(P'_1, P_2, X'_1, X_2),$$
$$P'_1 = 0, X_1 = 1, P_1 = 1, X_2 \neq 1$$

Relative Safety Proof

$$m(P_1, P_2, X_1, X_2) \models m(P_2, P_1, X_2, X_1)$$

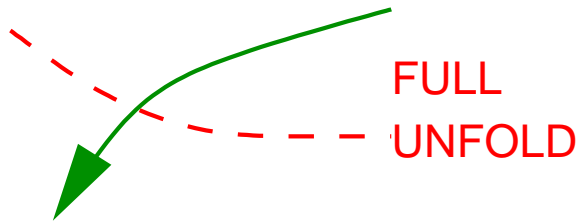


$$m(P'_1, P_2, X'_1, X_2), \\ P'_1 = 0, X_1 = 1, P_1 = 1, X_2 \neq 1 \quad \models$$

$$m(P_2, P''_1, X_2, X''_1), \\ P''_1 = 0, X_1 = 1, P_1 = 1$$

Relative Safety Proof

$$m(P_1, P_2, X_1, X_2) \models m(P_2, P_1, X_2, X_1)$$



$$m(P'_1, P_2, X'_1, X_2), \\ P'_1 = 0, X_1 = 1, P_1 = 1, X_2 \neq 1$$

\models

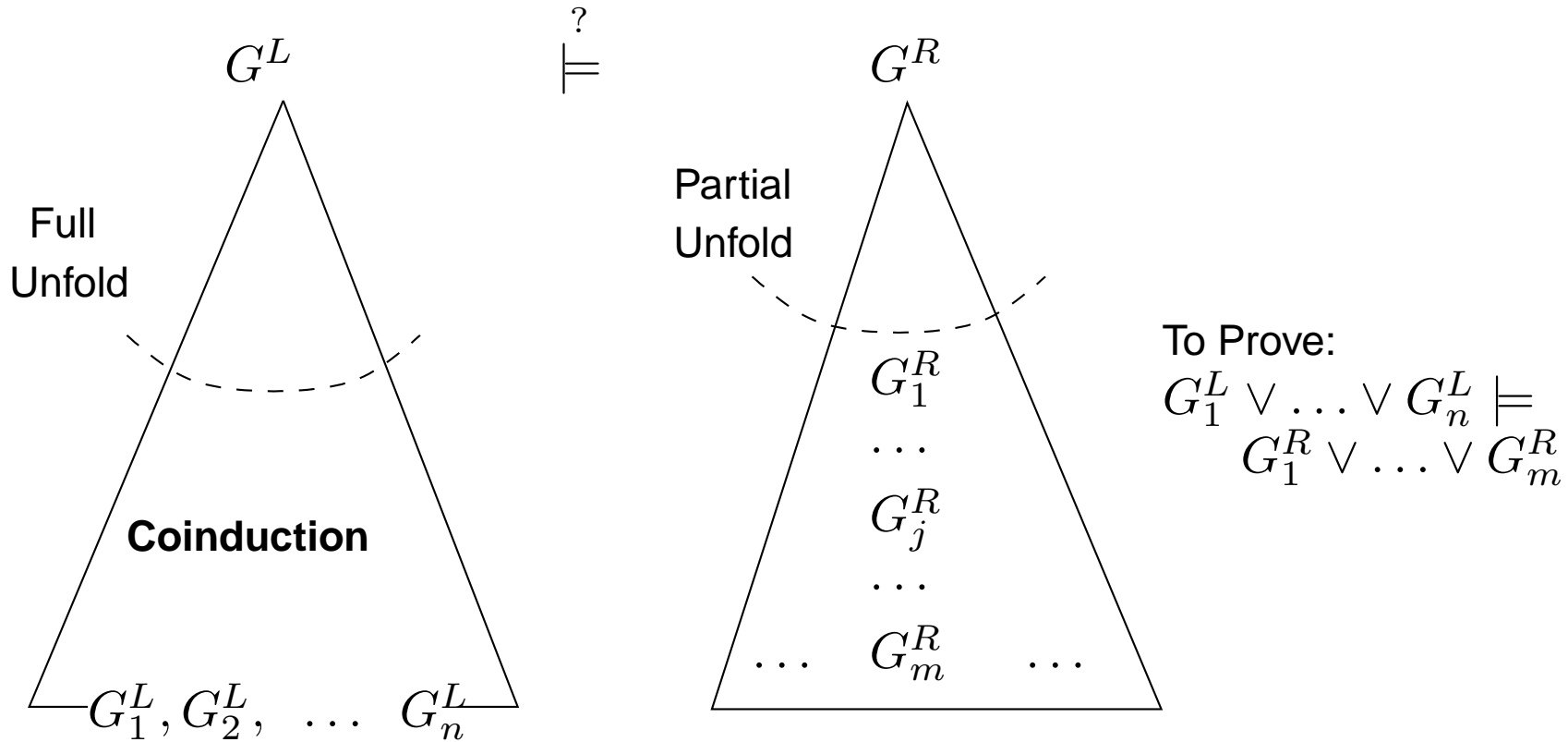
$$m(P_2, P''_1, X_2, X''_1), \\ P''_1 = 0, X_1 = 1, P_1 = 1$$



$$m(P_2, P'_1, X_2, X'_1), \\ P'_1 = 0, X_1 = 1, P_1 = 1, X_2 \neq 1$$



Summary of Proof Technique



Proof holds if $\forall 1 \leq i \leq n, \exists j : 1 \leq j \leq m : G_i^L \models G_j^R$

Proceeds by manipulation of a set of **proof obligations**.

$$\tilde{A} \vdash G^L \models G^R$$

Proof Rules

$$(LU+C) \quad \frac{\Pi \cup \{\tilde{A} \vdash G^L \models G^R\}}{\Pi \cup \bigcup_{i=1}^n \{\tilde{A} \cup \{G^L \models G^R\} \vdash G_i^L \models G^R\}} \quad \text{unfold}(G^L) = \{G_1^L, \dots, G_n^L\}$$

$$(RU) \quad \frac{\Pi \cup \{\tilde{A} \vdash G^L \models G^R\}}{\Pi \cup \{\tilde{A} \vdash G^L \models G_i^R\}} \quad G_i^R \in \text{unfold}(G^R)$$

$$(AP) \quad \frac{\Pi \cup \{\tilde{A} \vdash G^L, \phi \models G^R\}}{\Pi \cup \{\tilde{A} \vdash G_1^R \theta, \phi \models G^R\}} \quad \begin{array}{l} G_1^L \models G_1^R \in \tilde{A} \text{ and there} \\ \text{exists a renaming } \theta \text{ s.t.} \\ G^L \models G_1^L \theta \end{array}$$

$$(DP) \quad \frac{\Pi \cup \{\tilde{A} \vdash G^L, \phi \models G^R\}}{\Pi} \quad \text{There exists a renaming} \\ \theta \text{ s.t. } G^L \models G^R \theta$$

$$(SPL) \quad \frac{\Pi \cup \{\tilde{A} \vdash G^L \models G^R\}}{\Pi \cup \bigcup_{i=1}^k \{\tilde{A} \vdash G^L, \phi_i \models G^R\}} \quad \phi_1 \vee \dots \vee \phi_k \text{ is true.}$$

Main Theorem

A relative safety assertion $G^L \models G^R$ holds if, starting with the proof obligation $\Pi = \{\emptyset \vdash G^L \models G^R\}$, there exists a sequence of applications of proof rules that results in $\Pi = \emptyset$.

The relative safety assertion holds conditionally on \tilde{A} if we start with $\Pi = \{\tilde{A} \vdash G^L \models G^R\}$, where $\tilde{A} \neq \emptyset$.

Proving Relative Safety

Implementation in CLP(\mathcal{R})

Problem	Assertions #	Left	Right	Time
<i>Bakery-2</i>	1	9	27	0.00
<i>Bakery-3</i>	2	44	254	11.28
<i>Bakery-4</i>	3	147	1557	11.28
<i>Bakery-5</i>	4	424	7804	2320.3
<i>Priority</i>	1	43	220	0.04
<i>Szymanski-2</i>	8	362	28419	59.11
<i>Philosopher-3</i>	1	19	124	0.01
<i>Philosopher-4</i>	1	24	232	0.02
<i>Prod/Cons-10</i>	2	0	170	0.19
<i>Prod/Cons-20</i>	2	0	530	1.88

Using Relative Safety (Proving Safety)

Problem	No Rel. Safety		W/ Rel. Safety	
	Nodes	Time	Nodes	Time
<i>Bakery-4</i>	4624	6.60	191	0.20
<i>Bakery-5</i>	∞	∞	677	2.88
<i>Bakery-6</i>	∞	∞	2569	49.08
<i>Bakery-7</i>	∞	∞	11865	1052.32
<i>Szymanski-2</i>	240	0.08	84	0.02
<i>Szymanski-3</i>	10883	35.43	3176	2.91
<i>Philosopher-3</i>	882	0.51	553	0.30
<i>Philosopher-4</i>	4293	27.77	2783	9.67
<i>Prod/Cons-10</i>	664	0.10	171	0.02
<i>Prod/Cons-20</i>	2314	1.90	331	0.04

Conclusion

- Introduced **RELATIVE SAFETY** to specify **non-behavioral/structural properties** of programs
- Driving application: Symmetry
- A **PROOF METHOD** for relative safety using **COINDUCTION**
- Future works:
 - Application to more non-behavioral/structural properties.
 - Improvements on implementation.

What is Constraint $\models p(\dots)$?

Answer: Reachability

$$X - Y = 2 \models p(X, Y)$$

Proof by right partial unfold until initial state is found.