

An Interpolation Method for CLP Traversal

Joxan Jaffar, Andrew E. Santosa, Răzvan Voicu

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

School of Computing
National University of Singapore

Problem and Approach

CLP can be used for tree traversal:

Proving safety in program verification

= Proving desired answers satisfy a constraint

Requires memoing (tabling) s.t. not to repeat a subtree already traversed

Problem and Approach

CLP can be used for tree traversal:

Proving safety in program verification

= Proving desired answers satisfy a constraint

Requires memoing (tabling) s.t. not to repeat a subtree already traversed

Unfortunately: *State-space blowup*

Problem and Approach

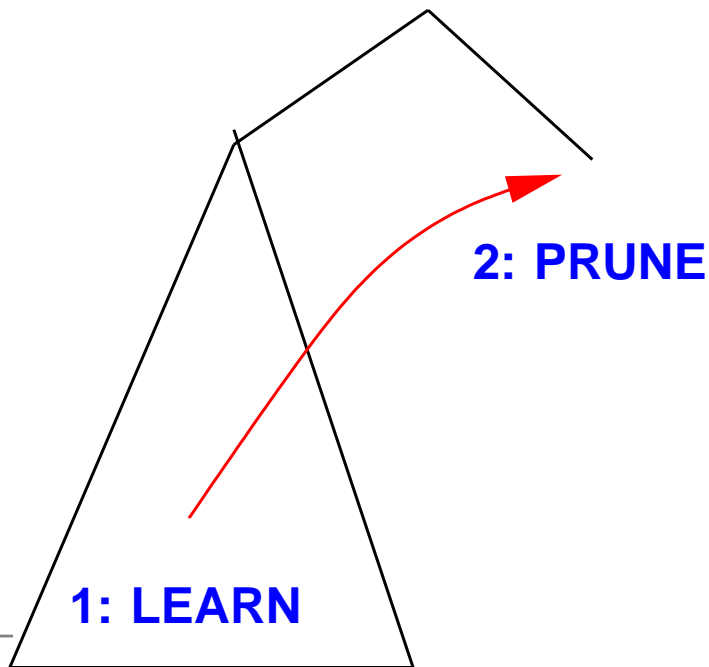
CLP can be used for tree traversal:

Proving safety in program verification

= Proving desired answers satisfy a constraint

Requires memoing (tabling) s.t. not to repeat a subtree already traversed

Unfortunately: *State-space blowup*



Mitigation: Use *learning* based on *interpolation*

* Enhances information stored in the table.

Outline

1. CLP for Modeling Programs
2. Interpolated Search
3. Algorithm
4. Computing Interpolants
5. Experimental Results
6. Related Work and Conclusion

Outline

1. CLP for Modeling Programs
2. Interpolated Search
3. Algorithm
4. Computing Interpolants
5. Experimental Results
6. Related Work and Conclusion

Constraint Logic Programs

A conjunction of implications (*Horn clauses*)

$$\text{even}(X) \text{ :- } X = 0.$$
$$\text{even}(X) \text{ :- } \text{even}(Y), X = Y + 2.$$

A clause's antecedent is called *body*,
its consequent is called *head*.

Head contains *single atom*.

Body contains *atoms* and *constraints*.

Constraint Logic Programs

A conjunction of implications (*Horn clauses*)

$$\text{even}(X) \text{ :- } X = 0.$$
$$\text{even}(X) \text{ :- } \text{even}(Y), X = Y + 2.$$

A clause's antecedent is called *body*,
its consequent is called *head*.

Head contains *single atom*.

Body contains *atoms* and *constraints*.

Executable via *resolution* constructing derivation *tree*:

→ can implement verifier:

derivation tree matches execution tree

CLP for Modeling Programs

⟨0⟩ if (*) then $x := x + 1$

⟨1⟩ if $(y \geq 1)$ then $x := x + 2$

⟨2⟩ if $(y < 1)$ then $x := x + 4$ ⟨3⟩

$p_0(X, Y) :- p_1(X, Y).$

$p_0(X, Y) :- p_1(X', Y), X' = X + 1.$

$p_1(X, Y) :- p_2(X, Y), Y < 1.$

$p_1(X, Y) :- p_2(X', Y), X' = X + 2, Y \geq 1.$

$p_2(X, Y) :- p_3(X, Y), Y \geq 1.$

$p_2(X, Y) :- p_3(X', Y), X' = X + 4, Y < 1.$

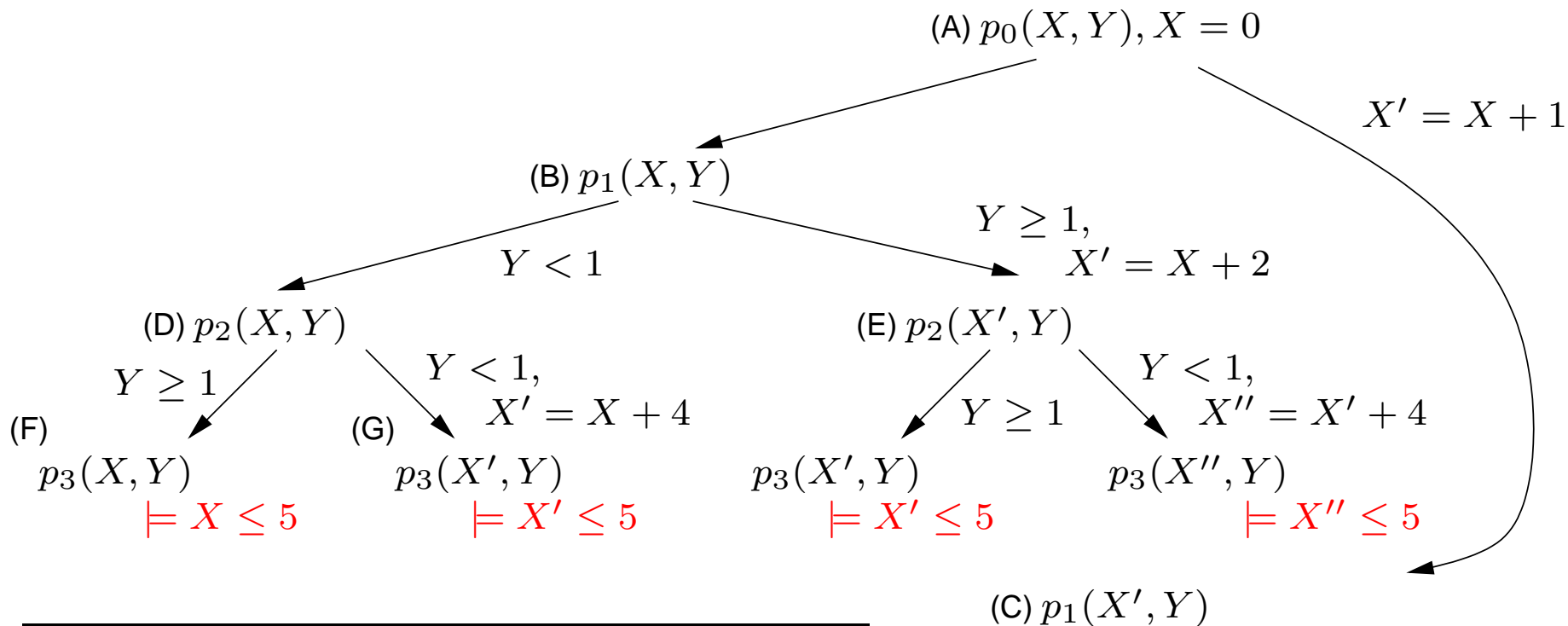
Outline

1. CLP for Modeling Programs
- 2. Interpolated Search**
3. Algorithm
4. Computing Interpolants
5. Experimental Results
6. Related Work and Conclusion

Interpolant

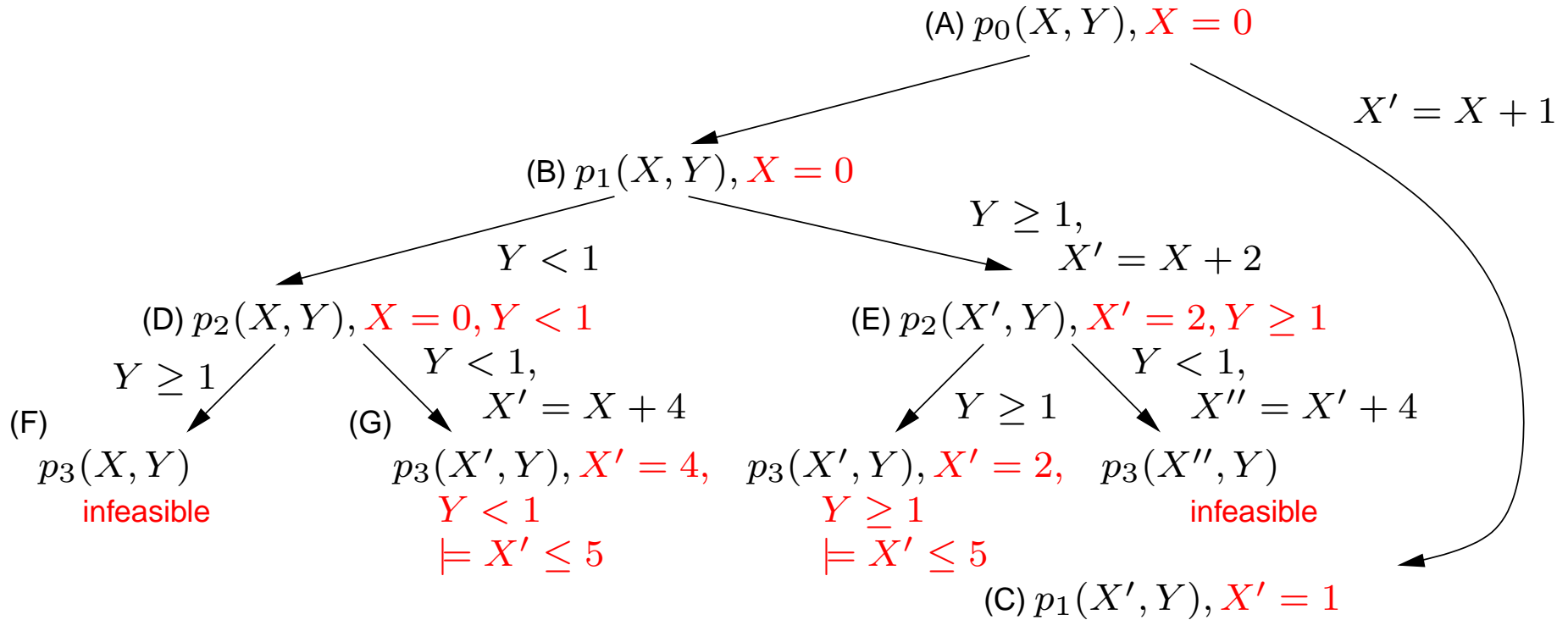
If F and G are formulas such that F entails G , then there exists an *interpolant* H which is a formula such that F entails H and H entails G , and each parameter of H is a parameter of both F and G . [Craig 1955]

Interpolated Search

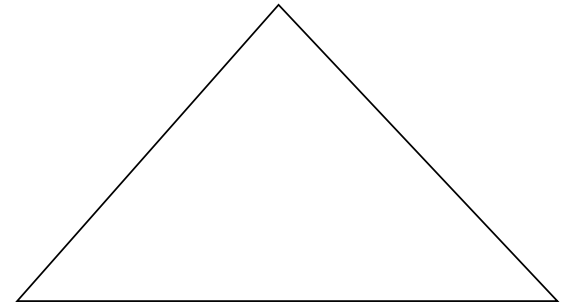


$p_0(X, Y) :- p_1(X, Y).$
 $p_0(X, Y) :- p_1(X', Y), X' = X + 1.$
 $p_1(X, Y) :- p_2(X, Y), Y < 1.$
 $p_1(X, Y) :- p_2(X', Y), X' = X + 2, Y \geq 1.$
 $p_2(X, Y) :- p_3(X, Y), Y \geq 1.$
 $p_2(X, Y) :- p_3(X', Y), X' = X + 4, Y < 1.$

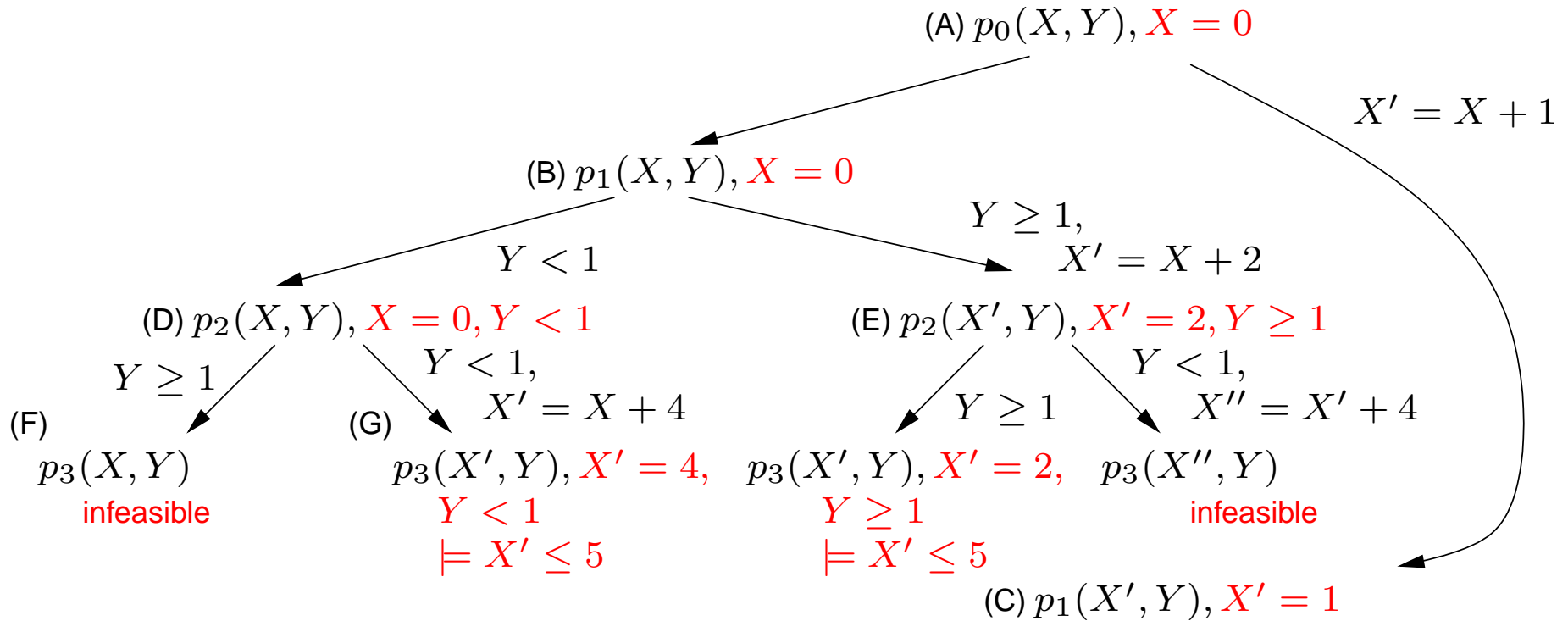
Interpolated Search



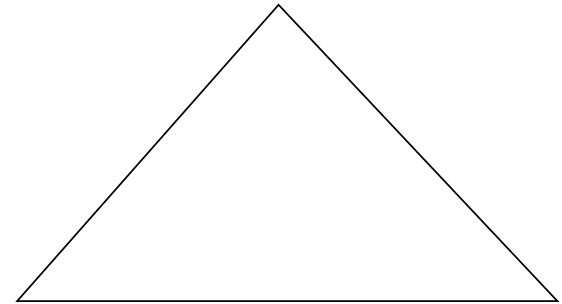
$X = 0, Y < 1 \models$
Interpolant \models
 $(Y \geq 1 \models \text{false})$



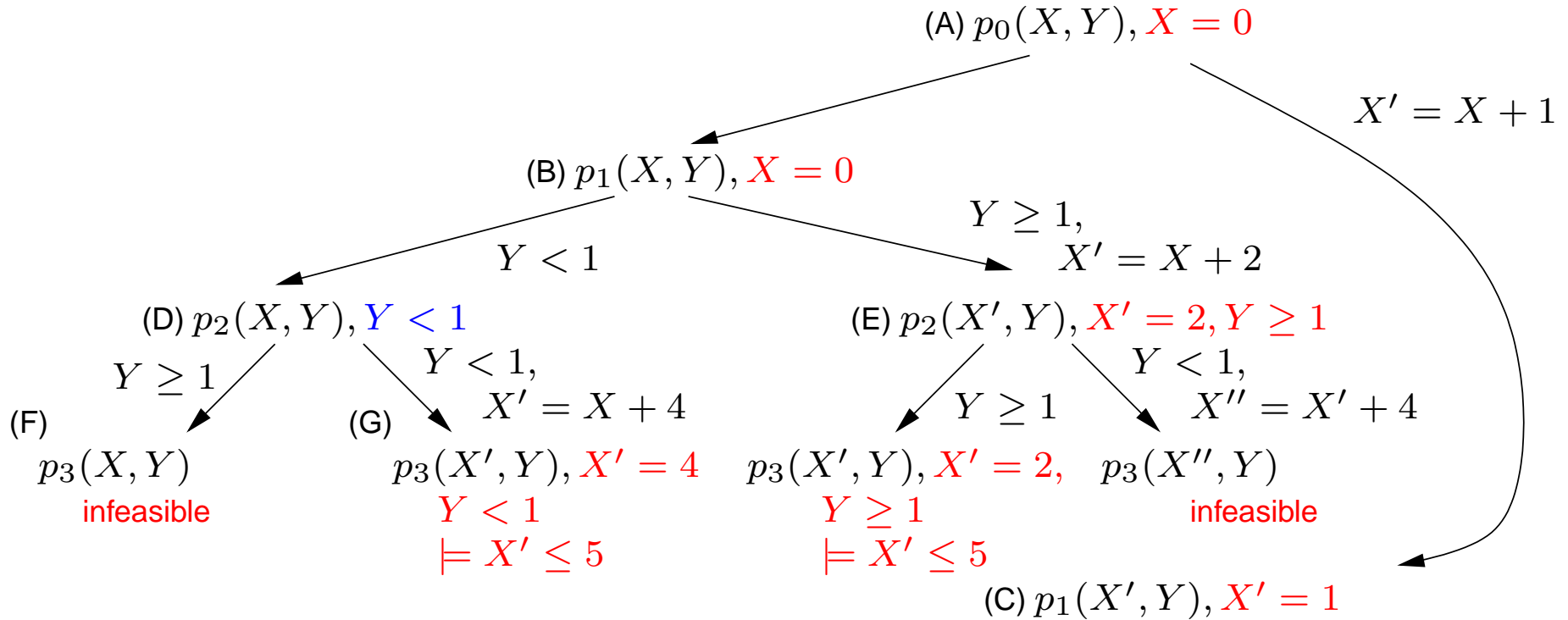
Interpolated Search



$X = 0, Y < 1 \models$
 $Y < 1 \models$
 $(Y \geq 1 \models \text{false})$

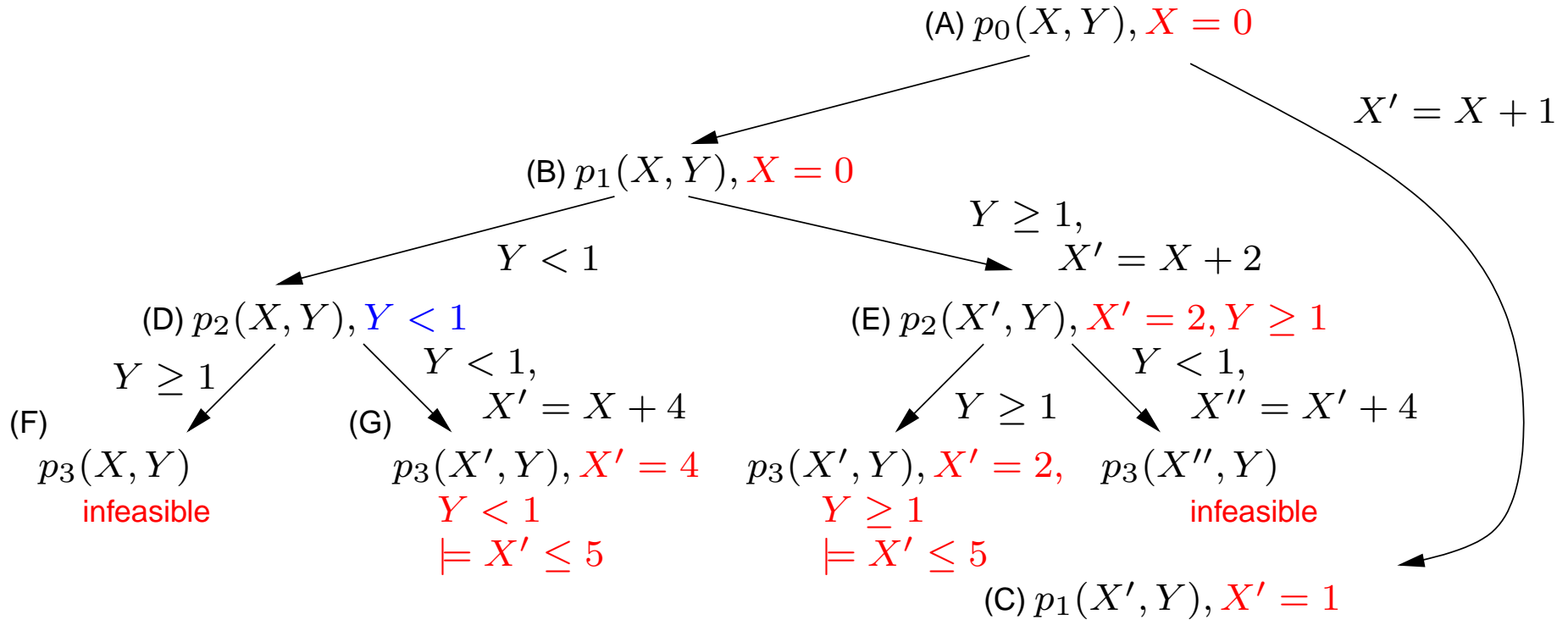


Interpolated Search



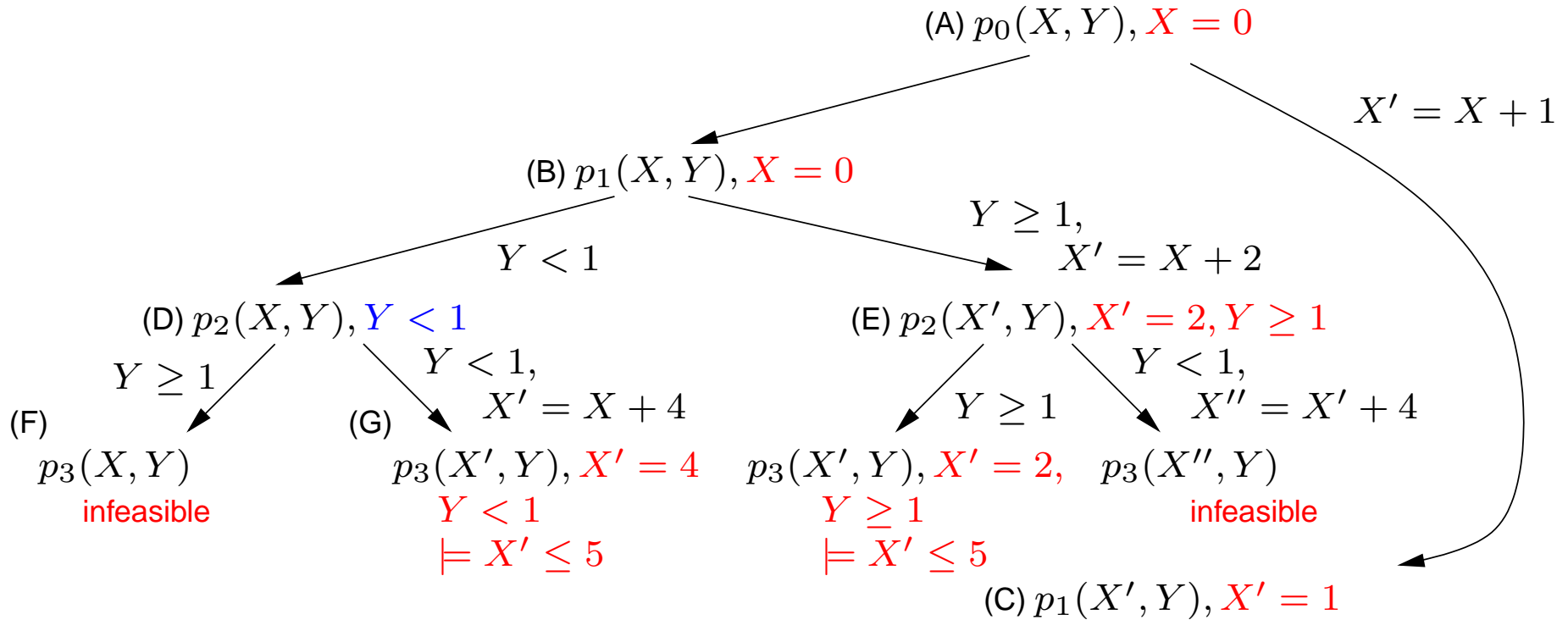
$X = 0, Y < 1 \models$
 $Y < 1 \models$
 $(Y \geq 1 \models \text{false})$

Interpolated Search

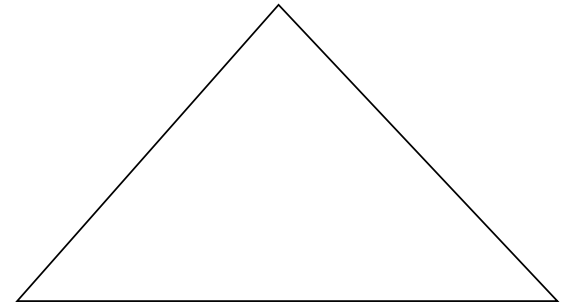


$X = 0, Y < 1, X' = X + 4 \models$
Interpolant \models
 $X' \leq 5$

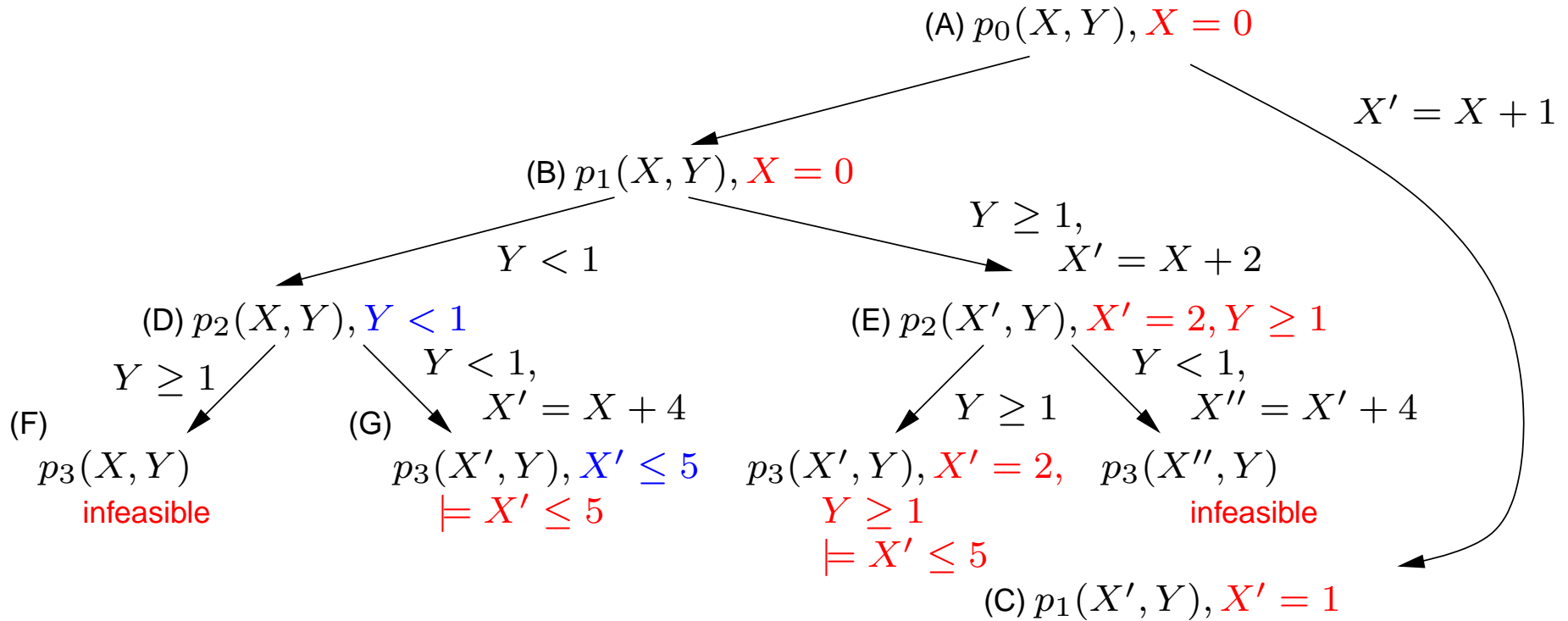
Interpolated Search



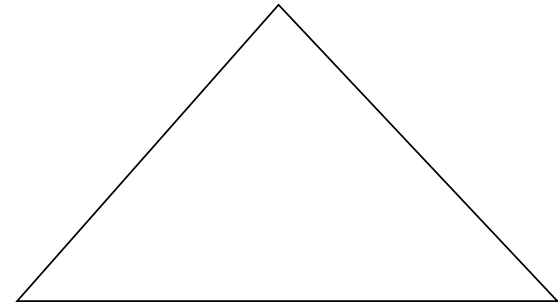
$$\begin{aligned}
 &X = 0, Y < 1, X' = X + 4 \models \\
 &X' \leq 5 \models \\
 &X' \leq 5
 \end{aligned}$$



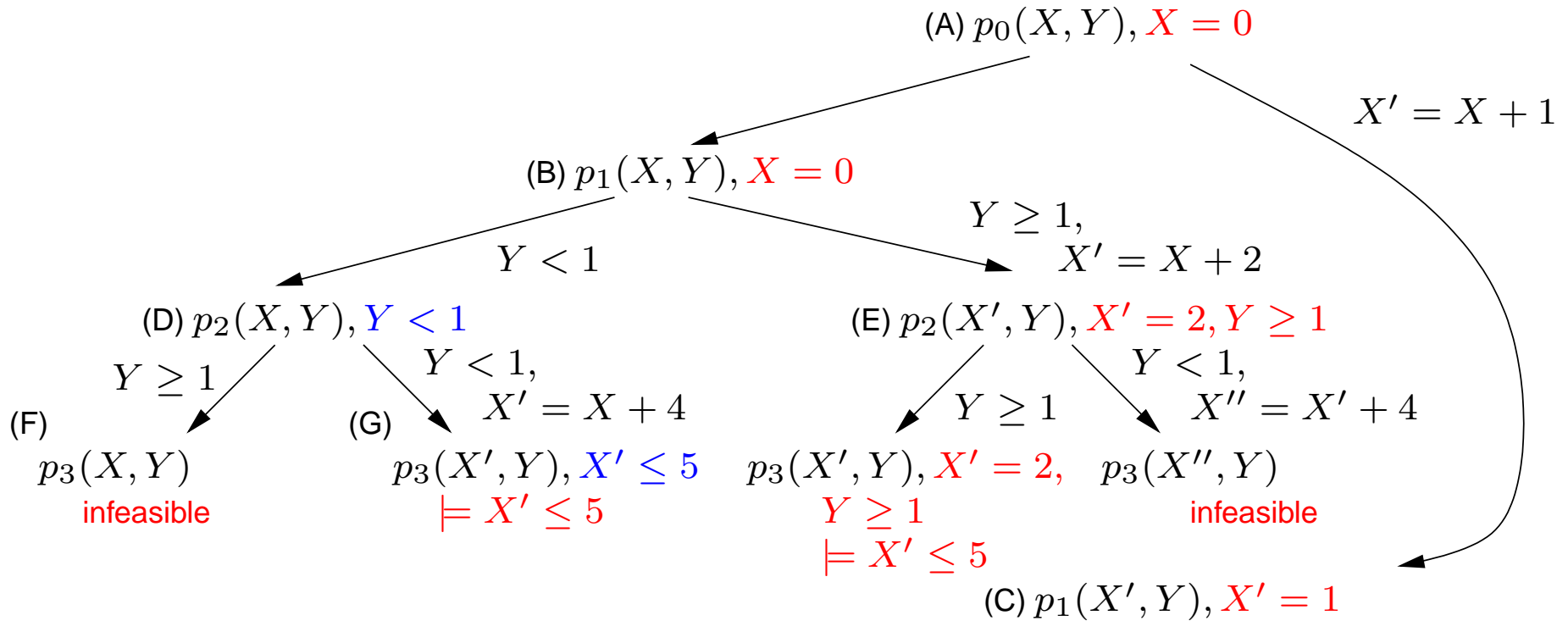
Interpolated Search



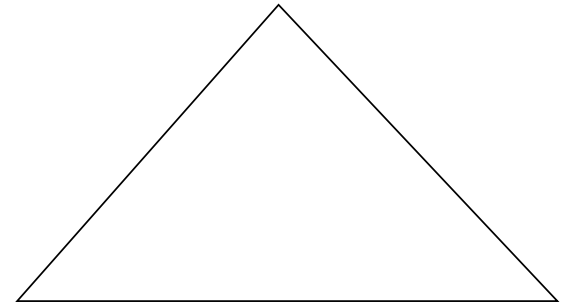
$$\begin{aligned}
 &X = 0, Y < 1, X' = X + 4 \models \\
 &X' \leq 5 \models \\
 &X' \leq 5
 \end{aligned}$$



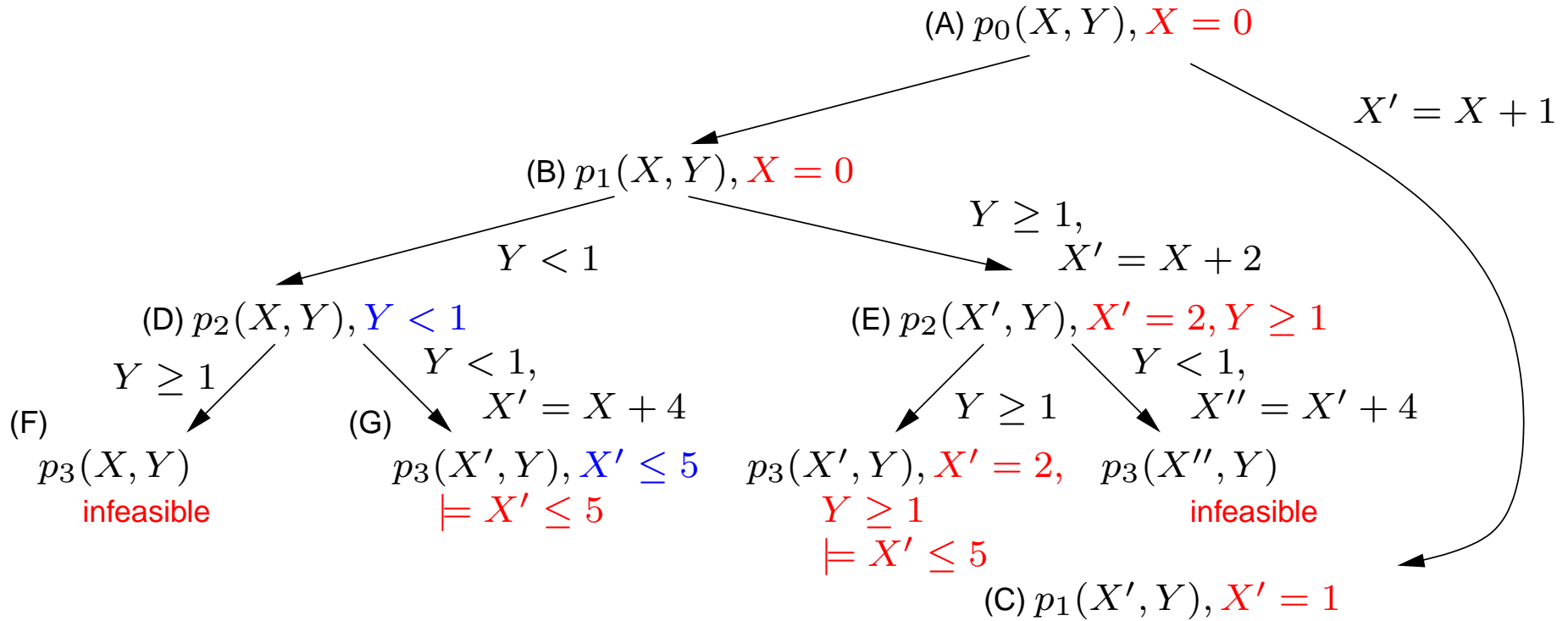
Interpolated Search



$X = 0, Y < 1 \models$
Interpolant \models
 $(Y < 1, X' = X + 4 \models X' \leq 5)$



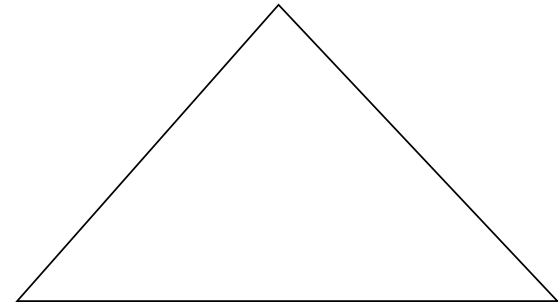
Interpolated Search



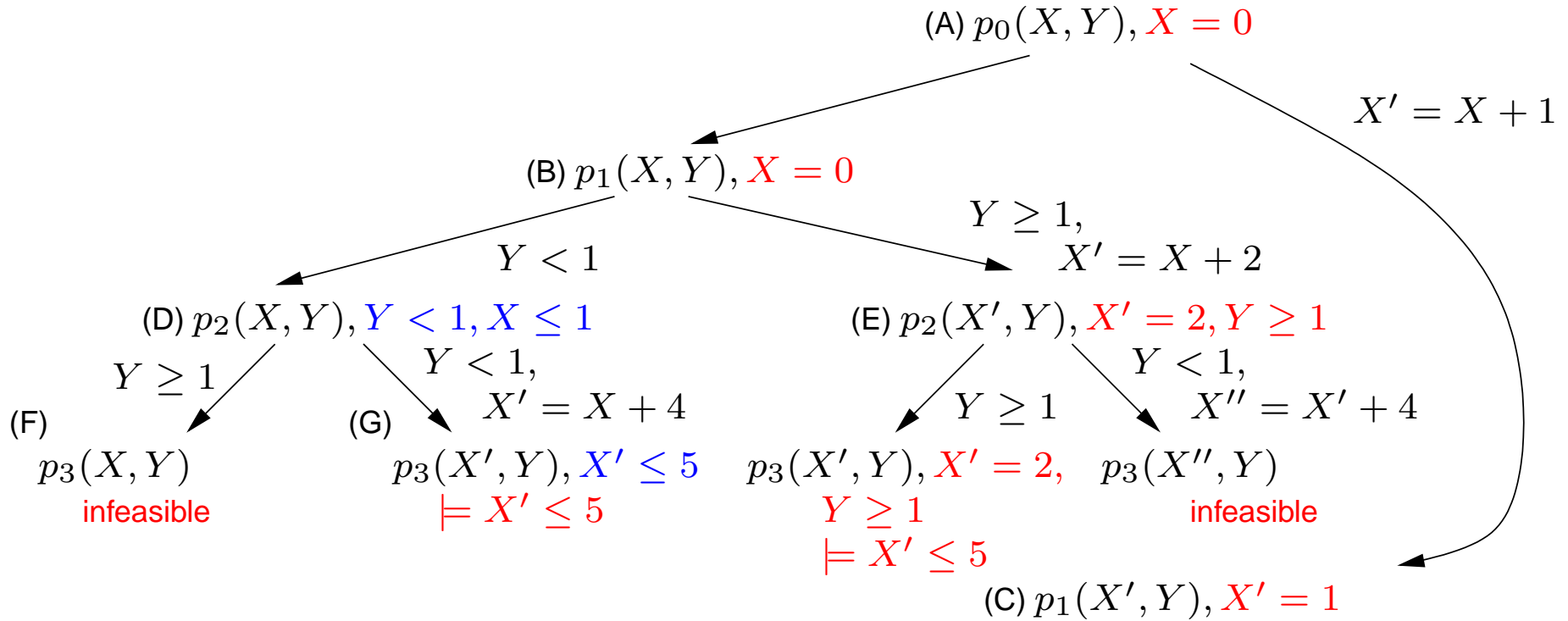
$$X = 0, Y < 1 \models$$

$$X \leq 1 \models$$

$$(Y < 1, X' = X + 4 \models X' \leq 5)$$



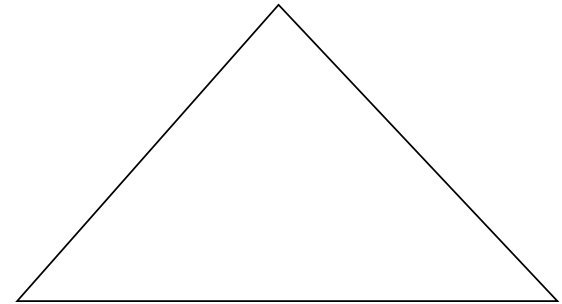
Interpolated Search



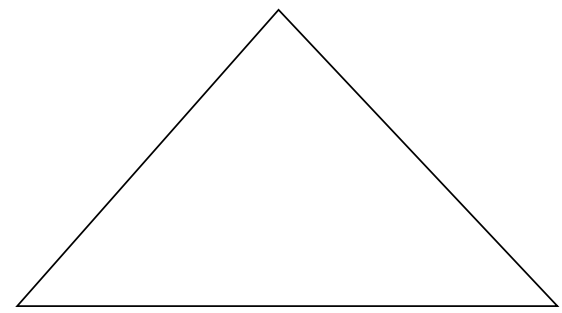
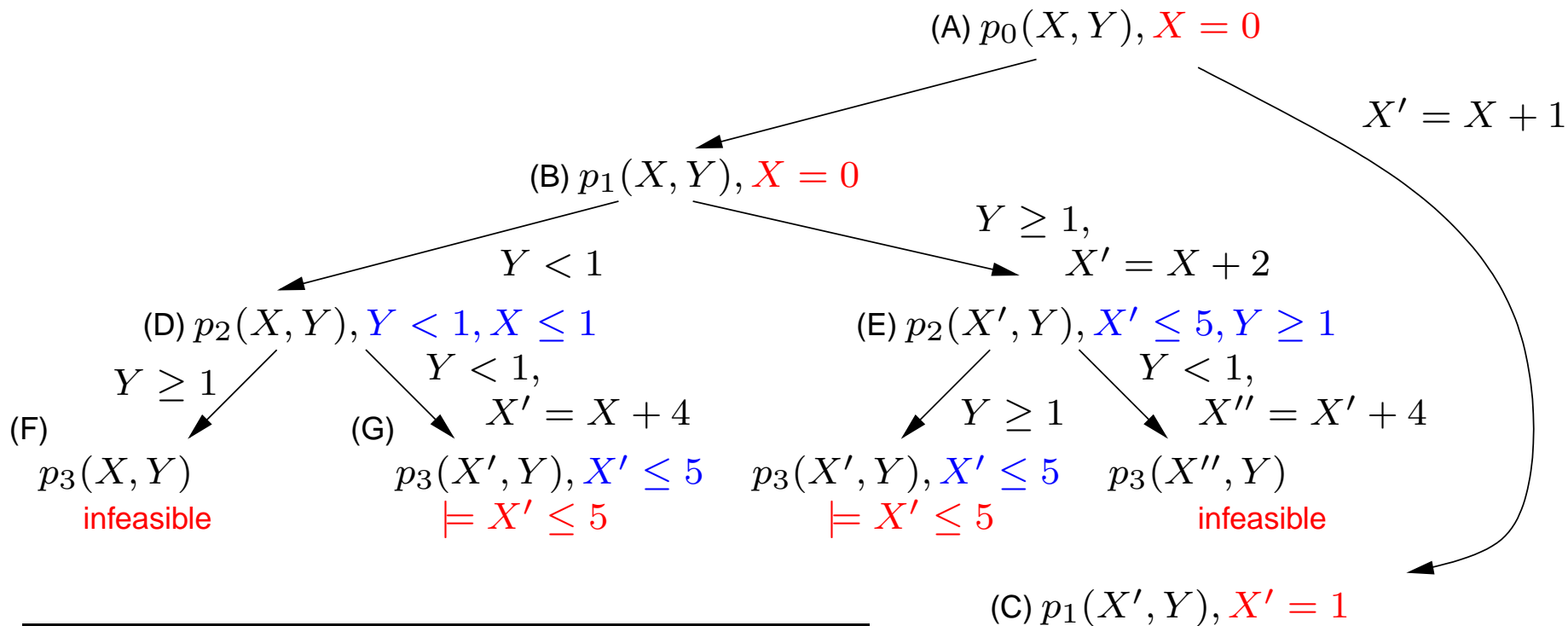
$$X = 0, Y < 1 \models$$

$$X \leq 1 \models$$

$$(Y < 1, X' = X + 4 \models X' \leq 5)$$

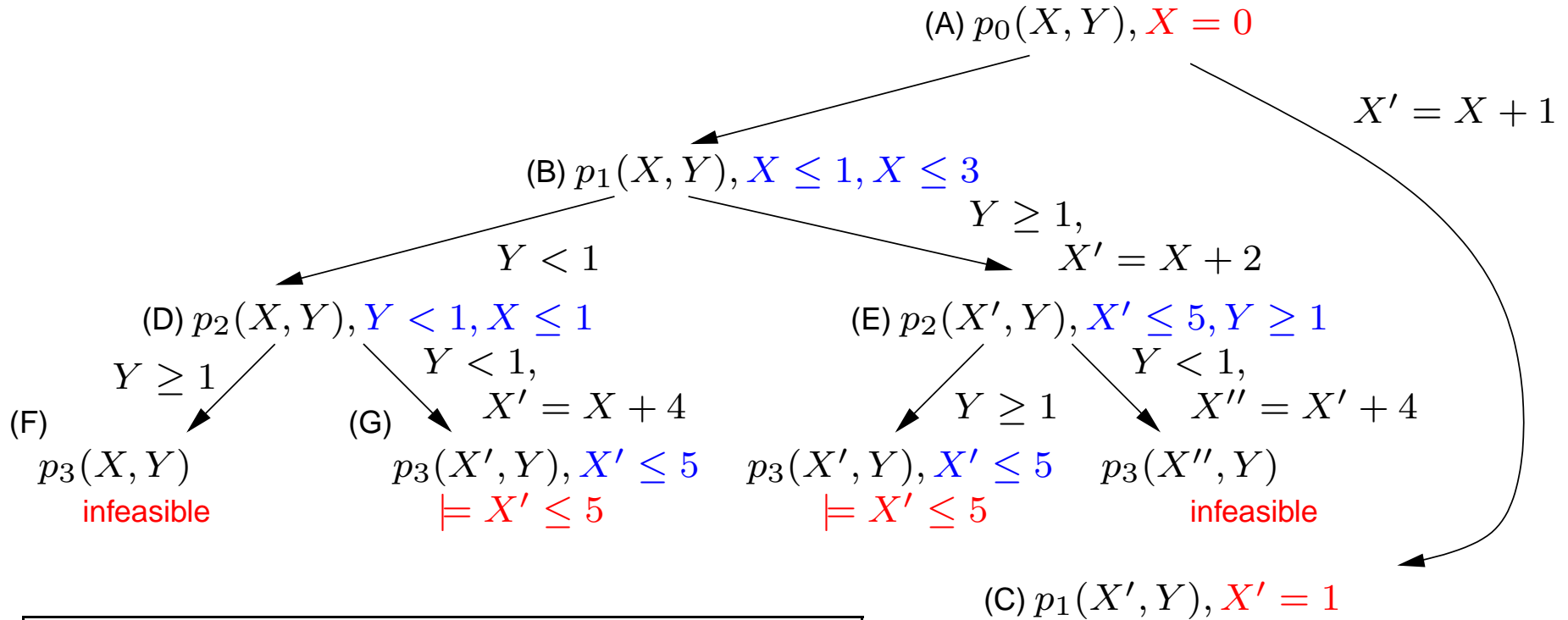


Interpolated Search

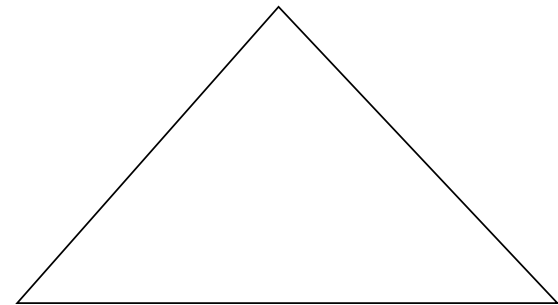


$p_0(X, Y) :- p_1(X, Y).$
 $p_0(X, Y) :- p_1(X', Y), X' = X + 1.$
 $p_1(X, Y) :- p_2(X, Y), Y < 1.$
 $p_1(X, Y) :- p_2(X', Y), X' = X + 2, Y \geq 1.$
 $p_2(X, Y) :- p_3(X, Y), Y \geq 1.$
 $p_2(X, Y) :- p_3(X', Y), X' = X + 4, Y < 1.$

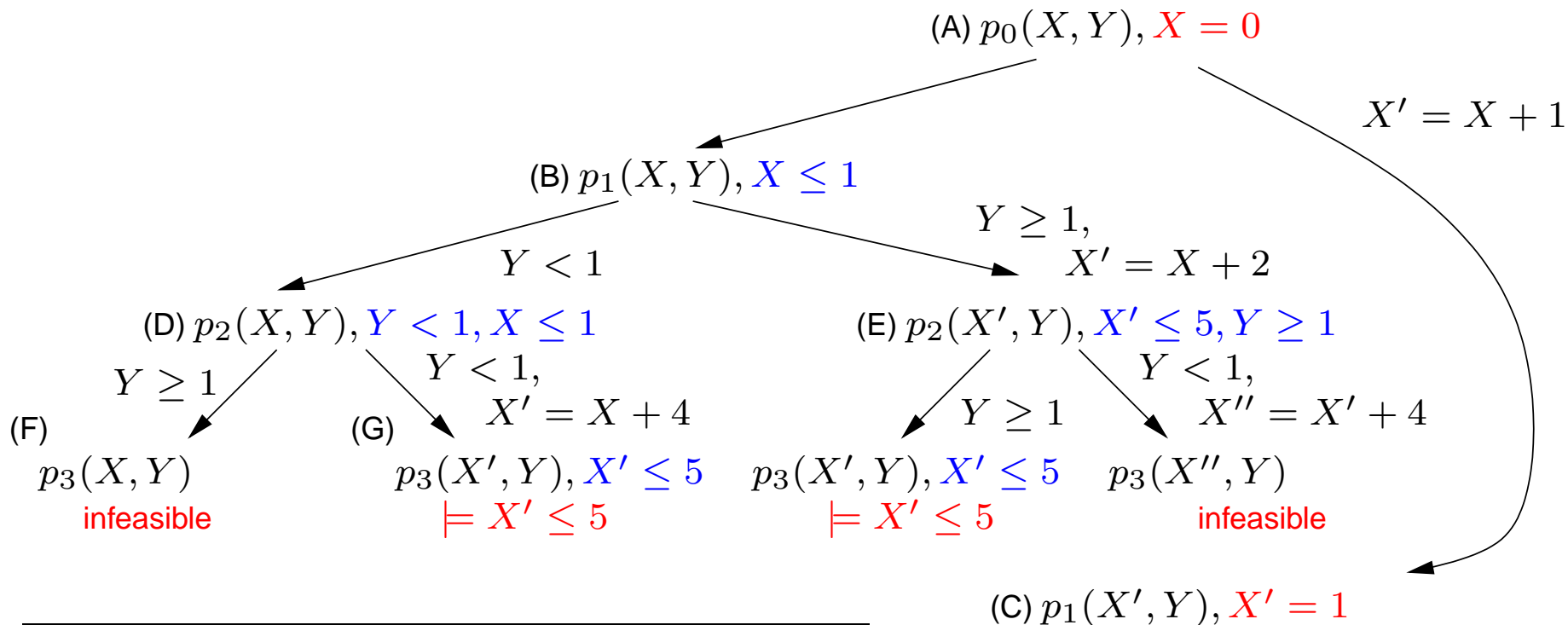
Interpolated Search



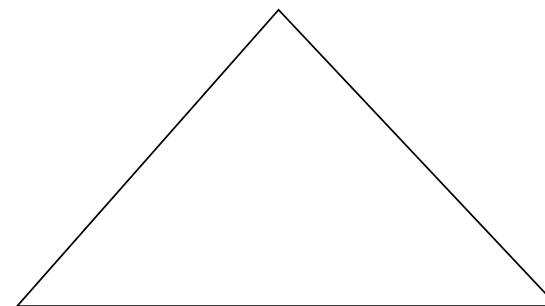
$p_0(X, Y) :- p_1(X, Y).$
 $p_0(X, Y) :- p_1(X', Y), X' = X + 1.$
 $p_1(X, Y) :- p_2(X, Y), Y < 1.$
 $p_1(X, Y) :- p_2(X', Y), X' = X + 2, Y \geq 1.$
 $p_2(X, Y) :- p_3(X, Y), Y \geq 1.$
 $p_2(X, Y) :- p_3(X', Y), X' = X + 4, Y < 1.$



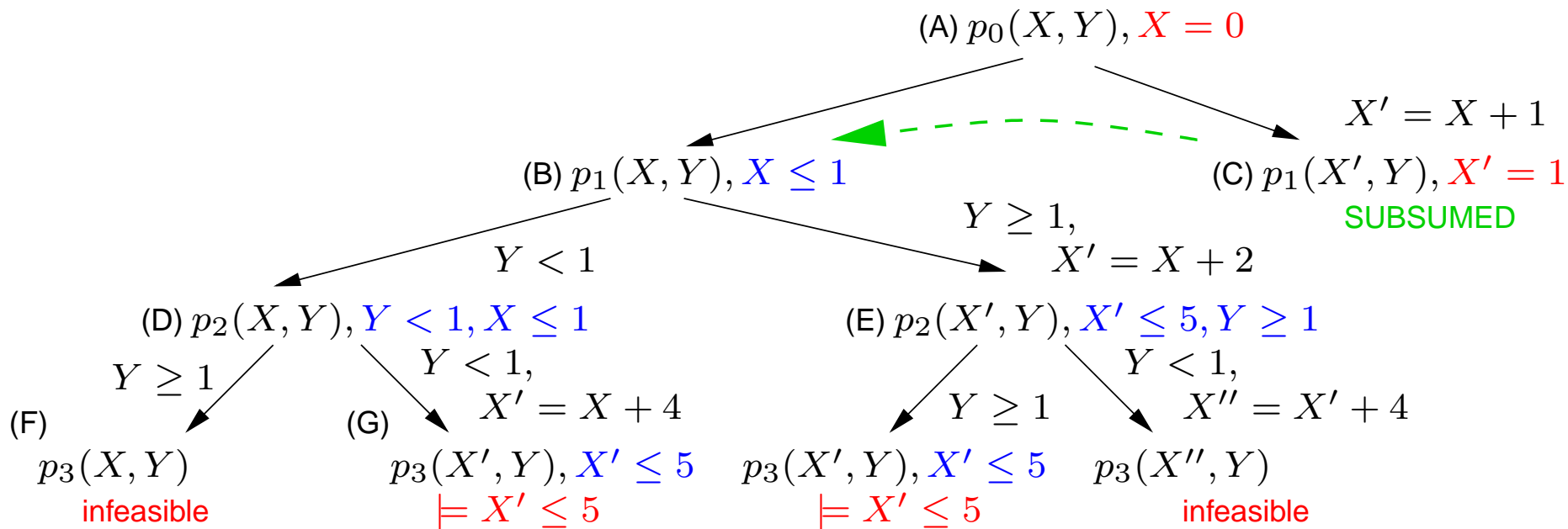
Interpolated Search



$p_0(X, Y) :- p_1(X, Y).$
 $p_0(X, Y) :- p_1(X', Y), X' = X + 1.$
 $p_1(X, Y) :- p_2(X, Y), Y < 1.$
 $p_1(X, Y) :- p_2(X', Y), X' = X + 2, Y \geq 1.$
 $p_2(X, Y) :- p_3(X, Y), Y \geq 1.$
 $p_2(X, Y) :- p_3(X', Y), X' = X + 4, Y < 1.$



Interpolated Search



$p_0(X, Y) :- p_1(X, Y).$

$p_0(X, Y) :- p_1(X', Y), X' = X + 1.$

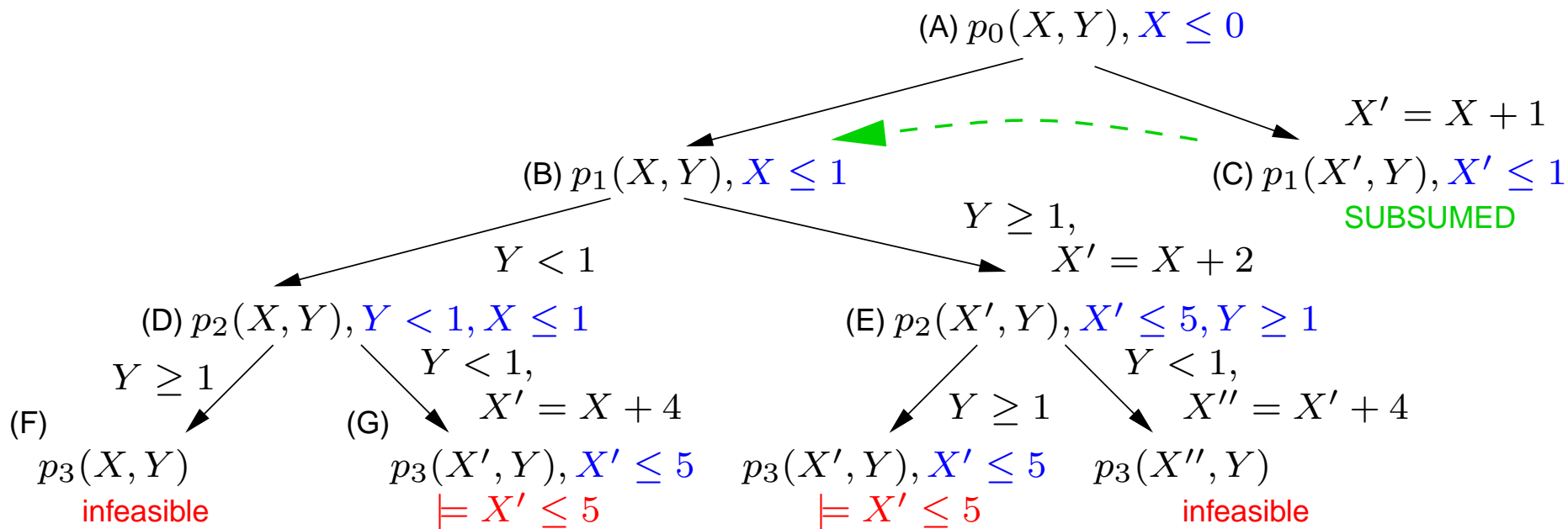
$p_1(X, Y) :- p_2(X, Y), Y < 1.$

$p_1(X, Y) :- p_2(X', Y), X' = X + 2, Y \geq 1.$

$p_2(X, Y) :- p_3(X, Y), Y \geq 1.$

$p_2(X, Y) :- p_3(X', Y), X' = X + 4, Y < 1.$

Interpolated Search



$p_0(X, Y) :- p_1(X, Y).$

$p_0(X, Y) :- p_1(X', Y), X' = X + 1.$

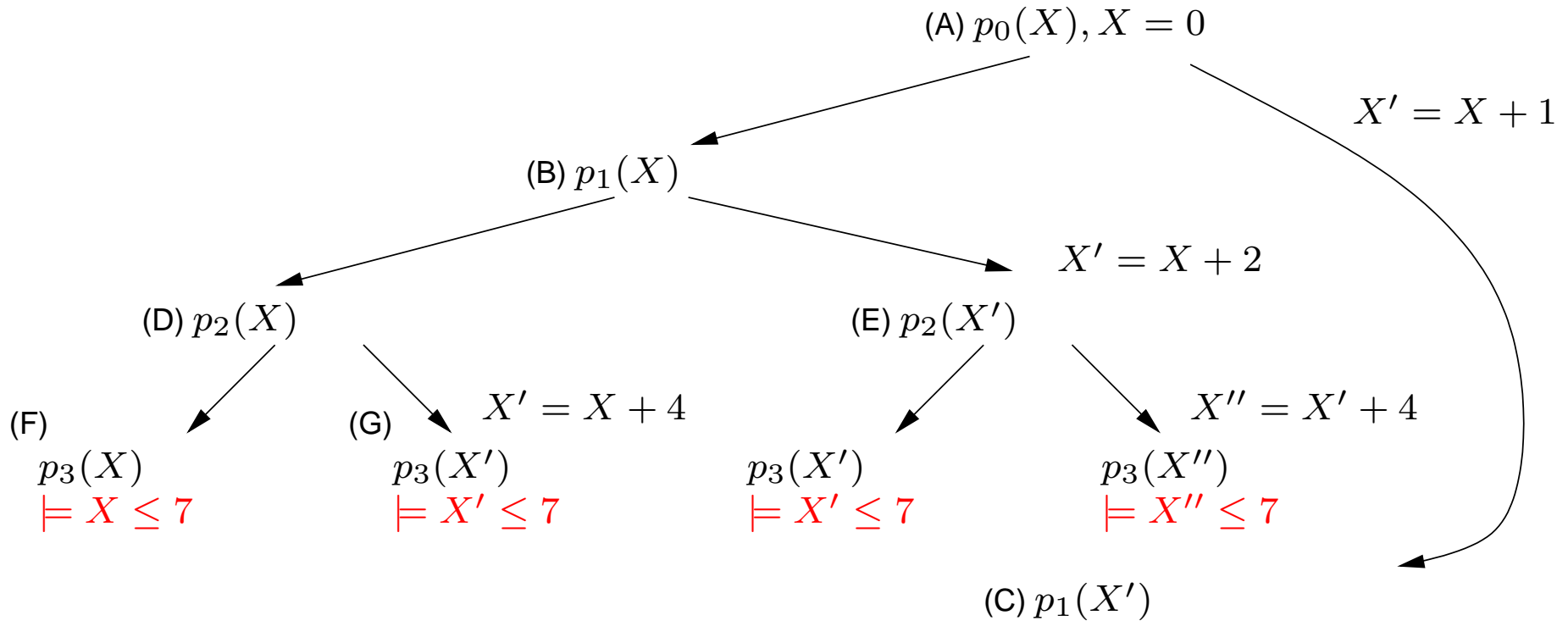
$p_1(X, Y) :- p_2(X, Y), Y < 1.$

$p_1(X, Y) :- p_2(X', Y), X' = X + 2, Y \geq 1.$

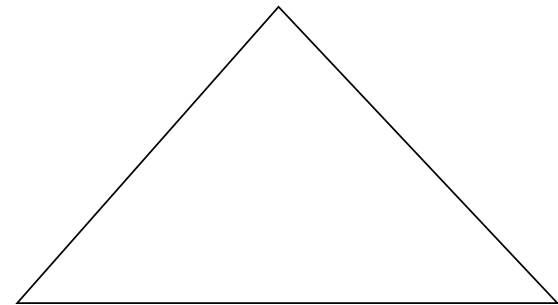
$p_2(X, Y) :- p_3(X, Y), Y \geq 1.$

$p_2(X, Y) :- p_3(X', Y), X' = X + 4, Y < 1.$

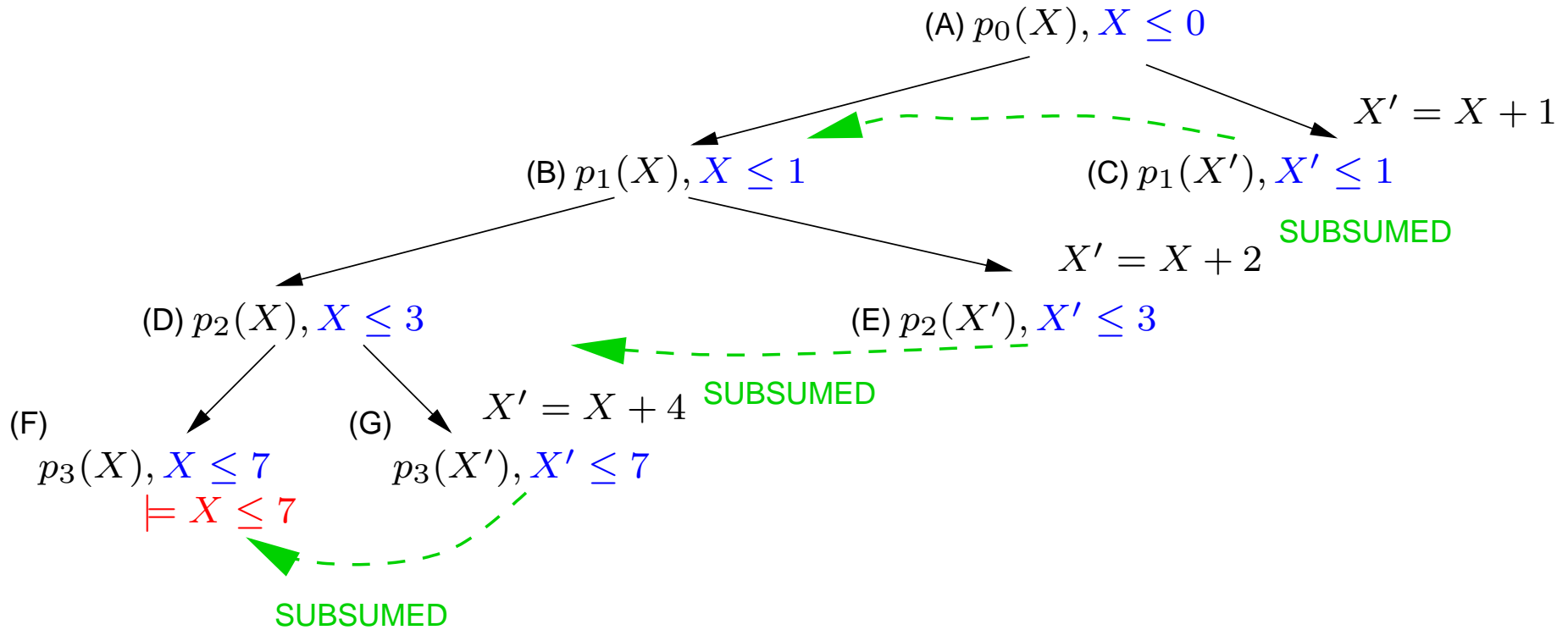
With No Infeasible Path



$p_0(X) :- p_1(X).$
 $p_0(X) :- p_1(X'), X' = X + 1.$
 $p_1(X) :- p_2(X).$
 $p_1(X) :- p_2(X'), X' = X + 2.$
 $p_2(X) :- p_3(X).$
 $p_2(X) :- p_3(X'), X' = X + 4.$



With No Infeasible Path



$p_0(X) :- p_1(X).$
 $p_0(X) :- p_1(X'), X' = X + 1.$
 $p_1(X) :- p_2(X).$
 $p_1(X) :- p_2(X'), X' = X + 2.$
 $p_2(X) :- p_3(X).$
 $p_2(X) :- p_3(X'), X' = X + 4.$

We made exponential size
linear (to the depth)

Interpolating Loop

$p_0(I, N, X, Y) :- p_1(I, N, X, Y), \varphi(X).$

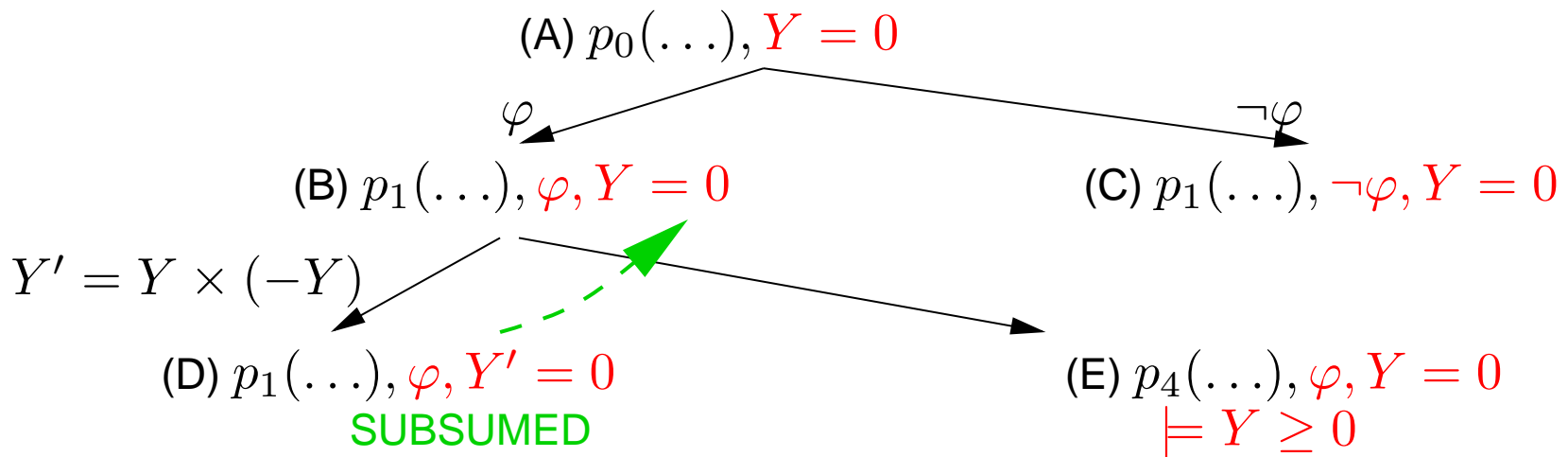
$p_0(I, N, X, Y) :- p_1(I, N, X, Y), \neg\varphi(X).$

$p_1(I, N, X, Y) :- p_2(I, N, X, Y), I \leq N.$

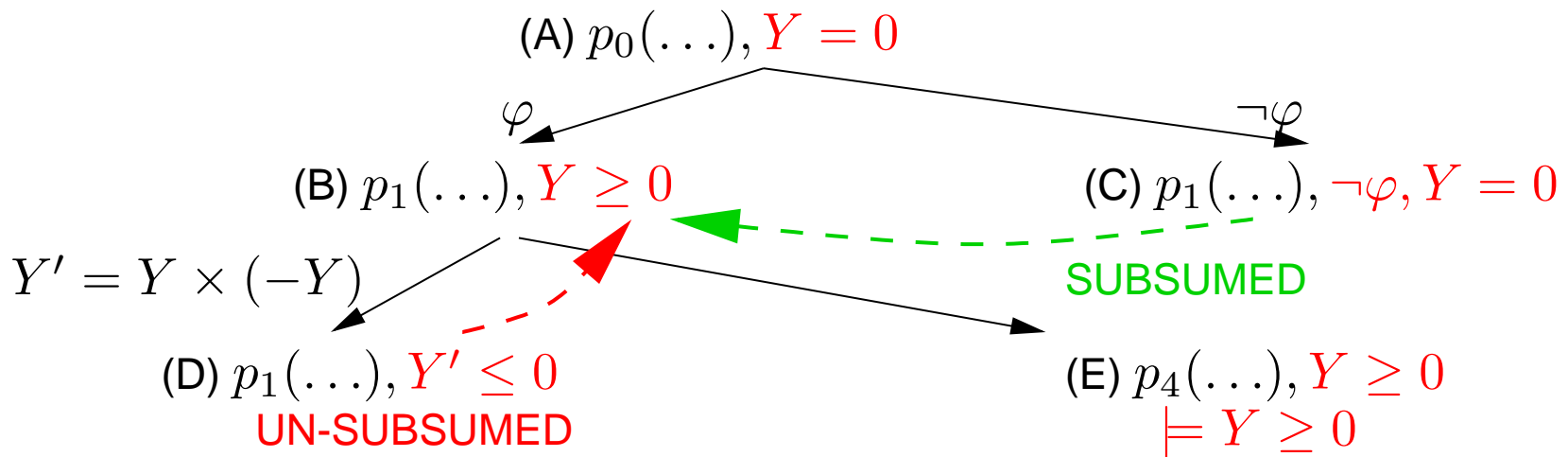
$p_1(I, N, X, Y) :- p_4(I, N, X, Y), I > N.$

$p_2(I, N, X, Y) :- p_3(I, N, X, Y'), Y' = Y \times (-Y).$

$p_3(I, N, X, Y) :- p_1(I', N, X, Y), I' = I + 1.$



Interpolating Loop

$$p_0(I, N, X, Y) :- p_1(I, N, X, Y), \varphi(X).$$
$$p_0(I, N, X, Y) :- p_1(I, N, X, Y), \neg\varphi(X).$$
$$p_1(I, N, X, Y) :- p_2(I, N, X, Y), I \leq N.$$
$$p_1(I, N, X, Y) :- p_4(I, N, X, Y), I > N.$$
$$p_2(I, N, X, Y) :- p_3(I, N, X, Y'), Y' = Y \times (-Y).$$
$$p_3(I, N, X, Y) :- p_1(I', N, X, Y), I' = I + 1.$$


Interpolating Loop

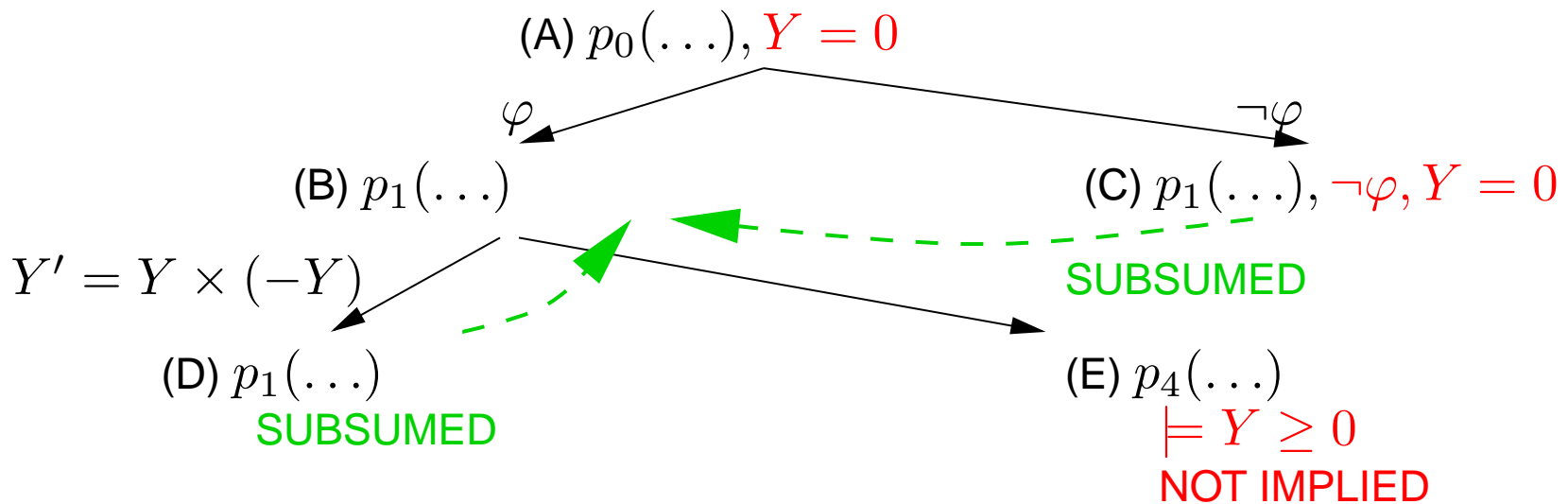
$$p_0(I, N, X, Y) :- p_1(I, N, X, Y), \varphi(X).$$

$$p_0(I, N, X, Y) :- p_1(I, N, X, Y), \neg\varphi(X).$$

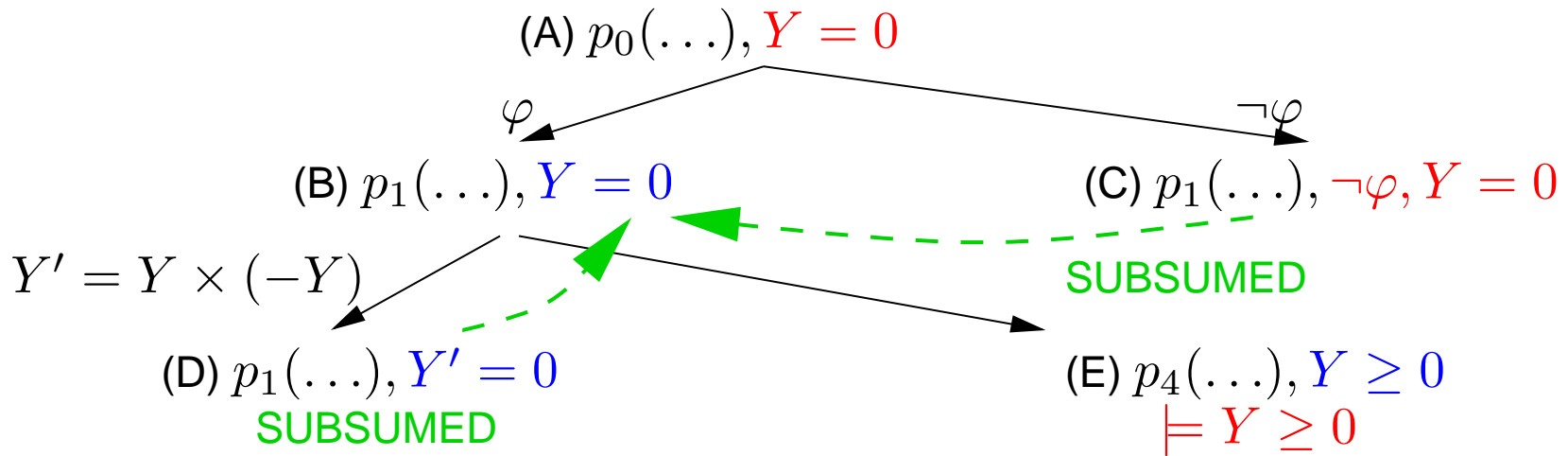
$$p_1(I, N, X, Y) :- p_2(I, N, X, Y), I \leq N.$$

$$p_1(I, N, X, Y) :- p_4(I, N, X, Y), I > N.$$

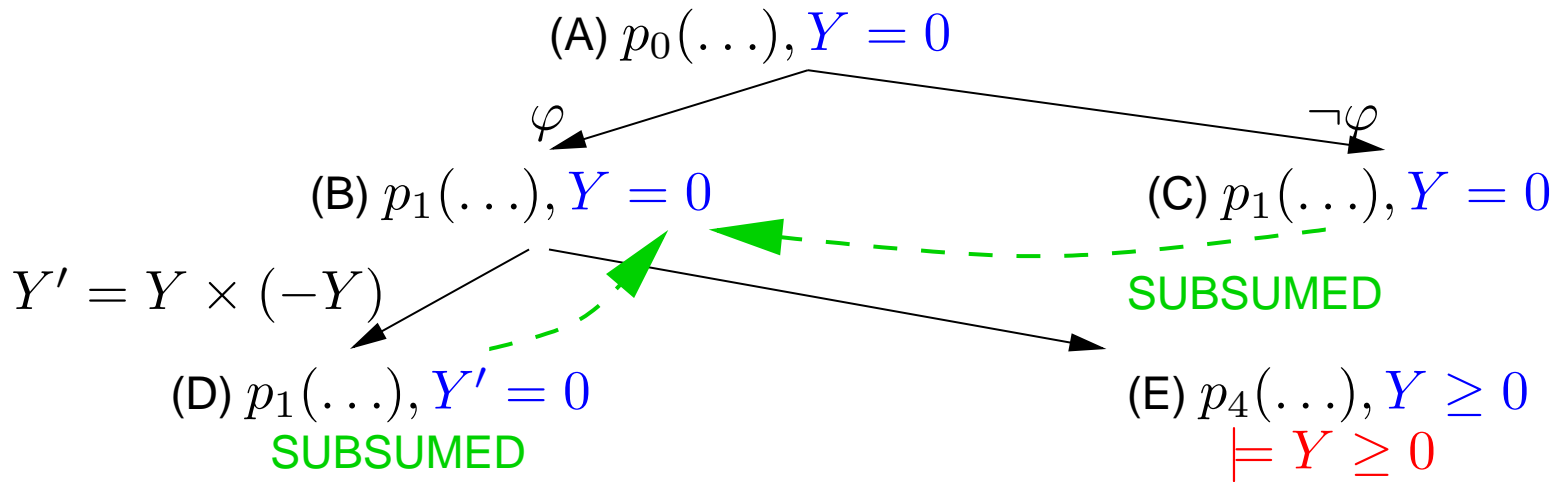
$$p_2(I, N, X, Y) :- p_3(I, N, X, Y'), Y' = Y \times (-Y).$$

$$p_3(I, N, X, Y) :- p_1(I', N, X, Y), I' = I + 1.$$


Interpolating Loop

$$p_0(I, N, X, Y) :- p_1(I, N, X, Y), \varphi(X).$$
$$p_0(I, N, X, Y) :- p_1(I, N, X, Y), \neg\varphi(X).$$
$$p_1(I, N, X, Y) :- p_2(I, N, X, Y), I \leq N.$$
$$p_1(I, N, X, Y) :- p_4(I, N, X, Y), I > N.$$
$$p_2(I, N, X, Y) :- p_3(I, N, X, Y'), Y' = Y \times (-Y).$$
$$p_3(I, N, X, Y) :- p_1(I', N, X, Y), I' = I + 1.$$


Interpolating Loop

$$p_0(I, N, X, Y) :- p_1(I, N, X, Y), \varphi(X).$$
$$p_0(I, N, X, Y) :- p_1(I, N, X, Y), \neg\varphi(X).$$
$$p_1(I, N, X, Y) :- p_2(I, N, X, Y), I \leq N.$$
$$p_1(I, N, X, Y) :- p_4(I, N, X, Y), I > N.$$
$$p_2(I, N, X, Y) :- p_3(I, N, X, Y'), Y' = Y \times (-Y).$$
$$p_3(I, N, X, Y) :- p_1(I', N, X, Y), I' = I + 1.$$


Outline

1. CLP for Modeling Programs
2. Interpolated Search
- 3. Algorithm**
4. Computing Interpolants
5. Experimental Results
6. Related Work and Conclusion

The Algorithm

```
solve( $\mathcal{G} \equiv p_k(\tilde{X}), \Psi$ ) returns a set of interpolants
  case ( $\mathcal{I} = \text{memoed}(\mathcal{G})$ ): return  $\mathcal{I}$ 
  case  $\mathcal{G}$  is false ( $\Psi \equiv \text{false}$ ): return  $\{p_k(\tilde{X}), \text{false}\}$ 
  case  $\mathcal{G}$  is target ( $k = k_f$ ):
    if ( $\Psi[\tilde{X}_f/\tilde{X}] \not\models \Psi_f$ ) abort
    else return  $\{\bar{\mathcal{G}} : \llbracket \bar{\mathcal{G}} \rrbracket \subseteq \llbracket p_{k_f}(\tilde{X}_f), \Psi_f \rrbracket\}$ 
  case  $\mathcal{G}$  is cyclic:
    let cyclic ancestor of  $\mathcal{G}$  be  $p_k(\tilde{X}'), \Psi'$  and  $\Psi \equiv \Psi' \wedge \Phi$ 
    return  $\{p_k(\tilde{X}), \bar{\Psi}'[\tilde{X}/\tilde{X}'] : \bar{\Psi}' \wedge \Phi \models \bar{\Psi}'[\tilde{X}/\tilde{X}']\}$ 
  case otherwise:
    foreach rule  $p_k(\tilde{X}) \quad :- \quad p_{k'}(\tilde{X}'), \rho(\tilde{X}, \tilde{X}')$ :
       $\mathcal{I} := \mathcal{I} \cap \{p_k(\tilde{X}), \text{WP}(\bar{\mathcal{G}}', \rho) : \bar{\mathcal{G}}' \in \text{solve}(p_{k'}(\tilde{X}'), \Psi \wedge \rho)\}$ 
    endfor
  memo( $\mathcal{I}$ ) and return  $\mathcal{I}$ 
```

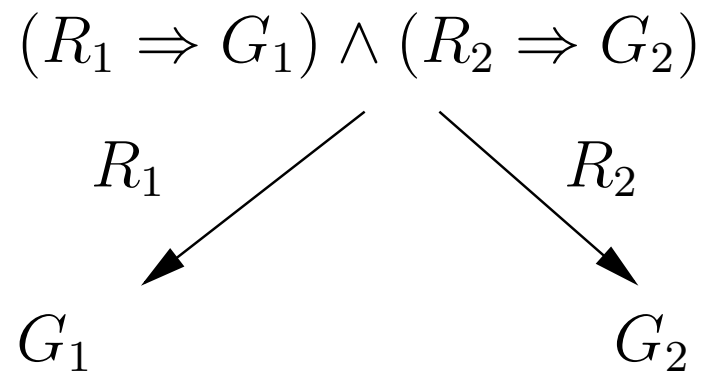
Outline

1. CLP for Modeling Programs
2. Interpolated Search
3. Algorithm
- 4. Computing Interpolants**
5. Experimental Results
6. Related Work and Conclusion

Serial Constraint Replacement

We wish we can compute *weakest precondition*
= most general interpolant

**** Complex Formula ****



Serial Constraint Replacement

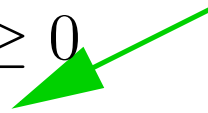
$$p_0(\dots), X = 0, Y = 2$$



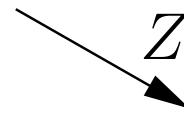
$$X' = X + 1$$

$$p_1(\dots), X' = 1, Y = 2$$

$$Z \geq 0$$



$$Z < 0$$



$$p_2(\dots), X' = 1, Y = 2, Z \geq 0$$

$$p_3(\dots), X' = 1, Y = 2, Z < 0$$



$$Y' = X'$$

$$\models Y \geq 1$$

$$p_3(\dots), X' = 1, Y' = 1, Z \geq 0$$

$$\models Y' \geq 1$$

Serial Constraint Replacement

$$p_0(\dots), \underline{X = 0}, Y = 2$$



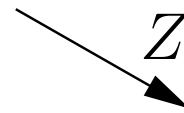
$$\underline{X' = X + 1}$$

$$p_1(\dots), X' = 1, Y = 2$$

$$Z \geq 0$$

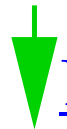


$$Z < 0$$



$$p_2(\dots), X' = 1, Y = 2, Z \geq 0$$

$$p_3(\dots), X' = 1, Y = 2, Z < 0$$



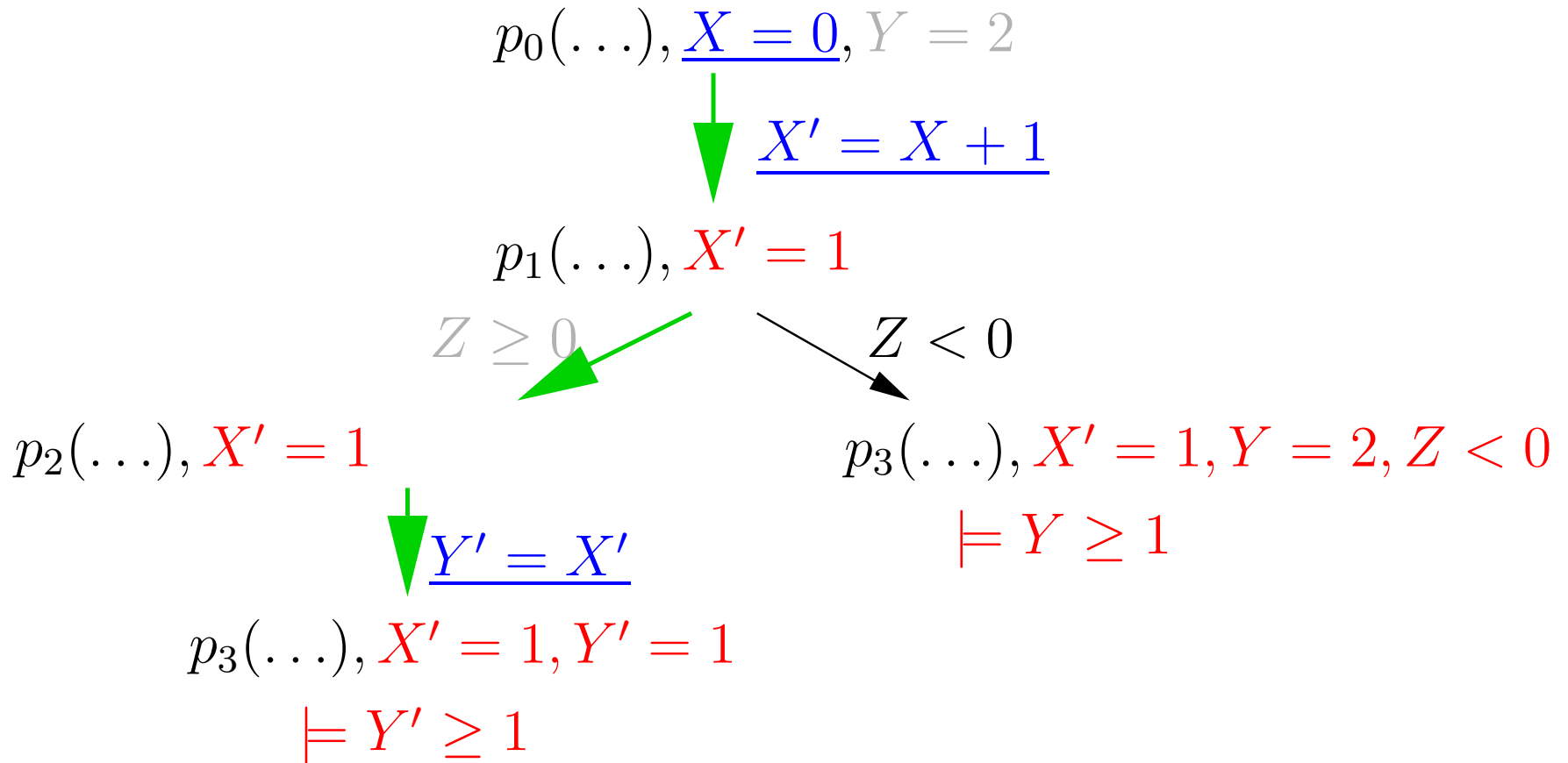
$$\underline{Y' = X'}$$

$$\models Y \geq 1$$

$$p_3(\dots), X' = 1, Y' = 1, Z \geq 0$$

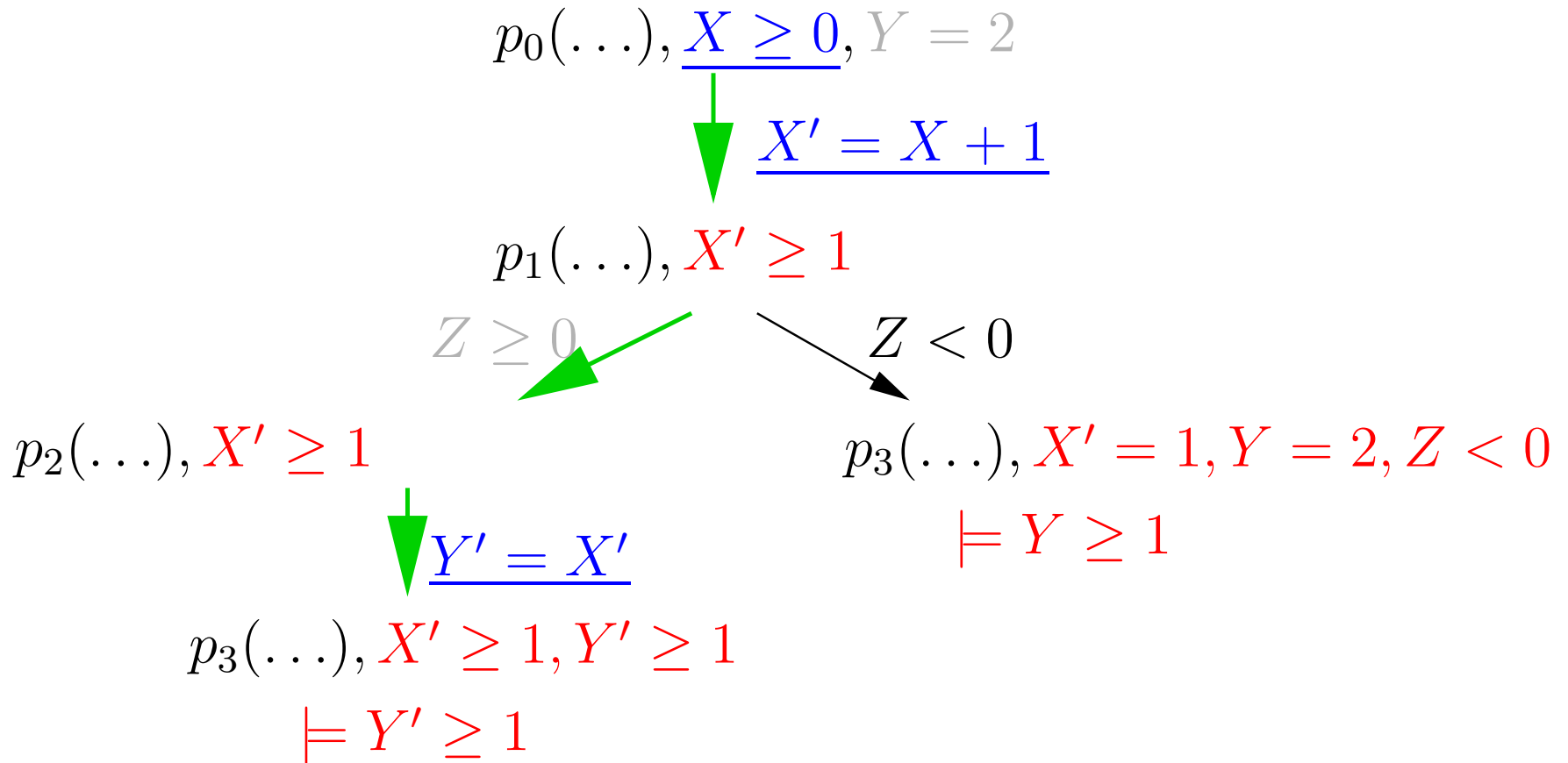
$$\models Y' \geq 1$$

Serial Constraint Replacement



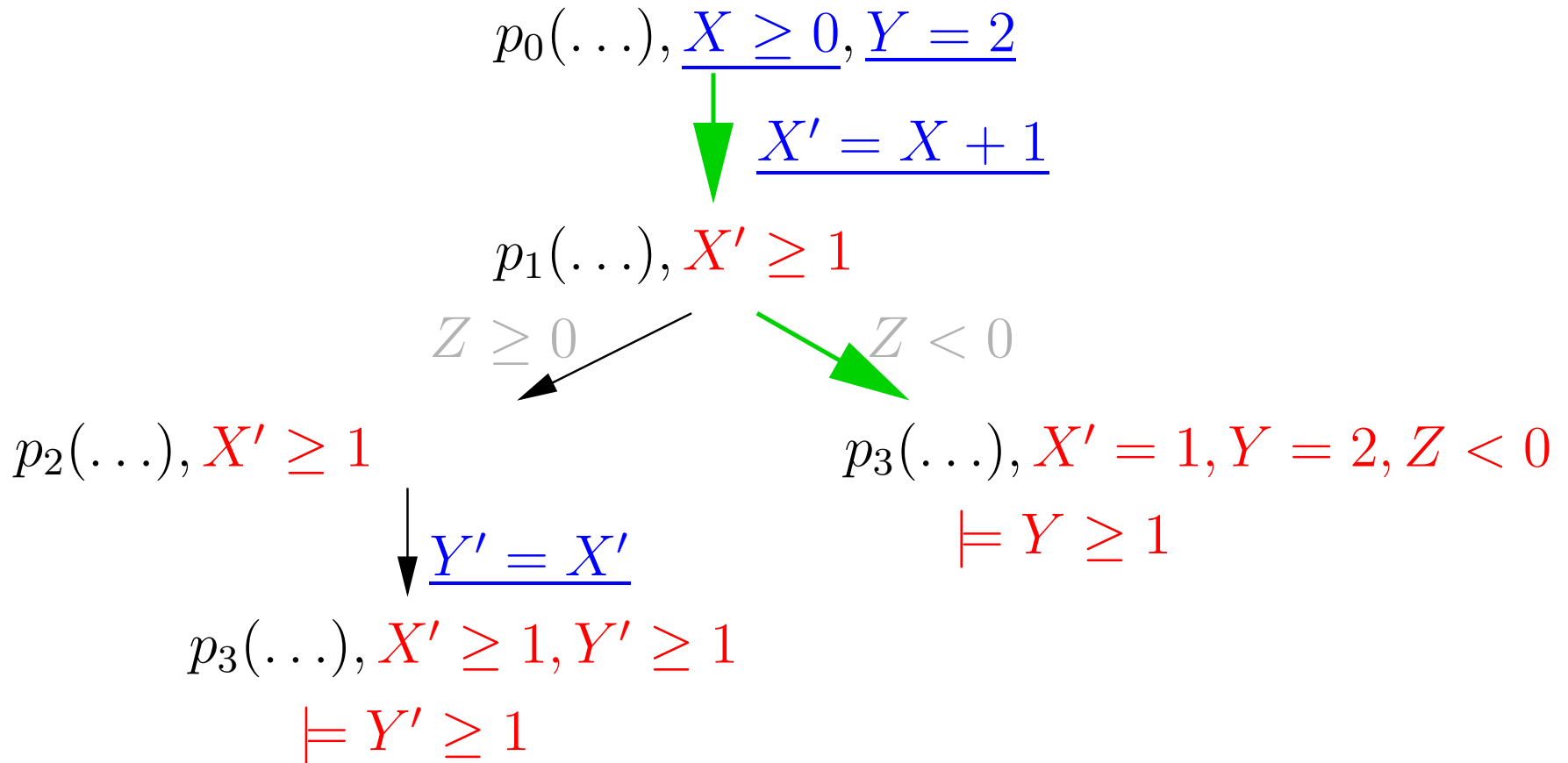
”Constraint Deletion”

Serial Constraint Replacement

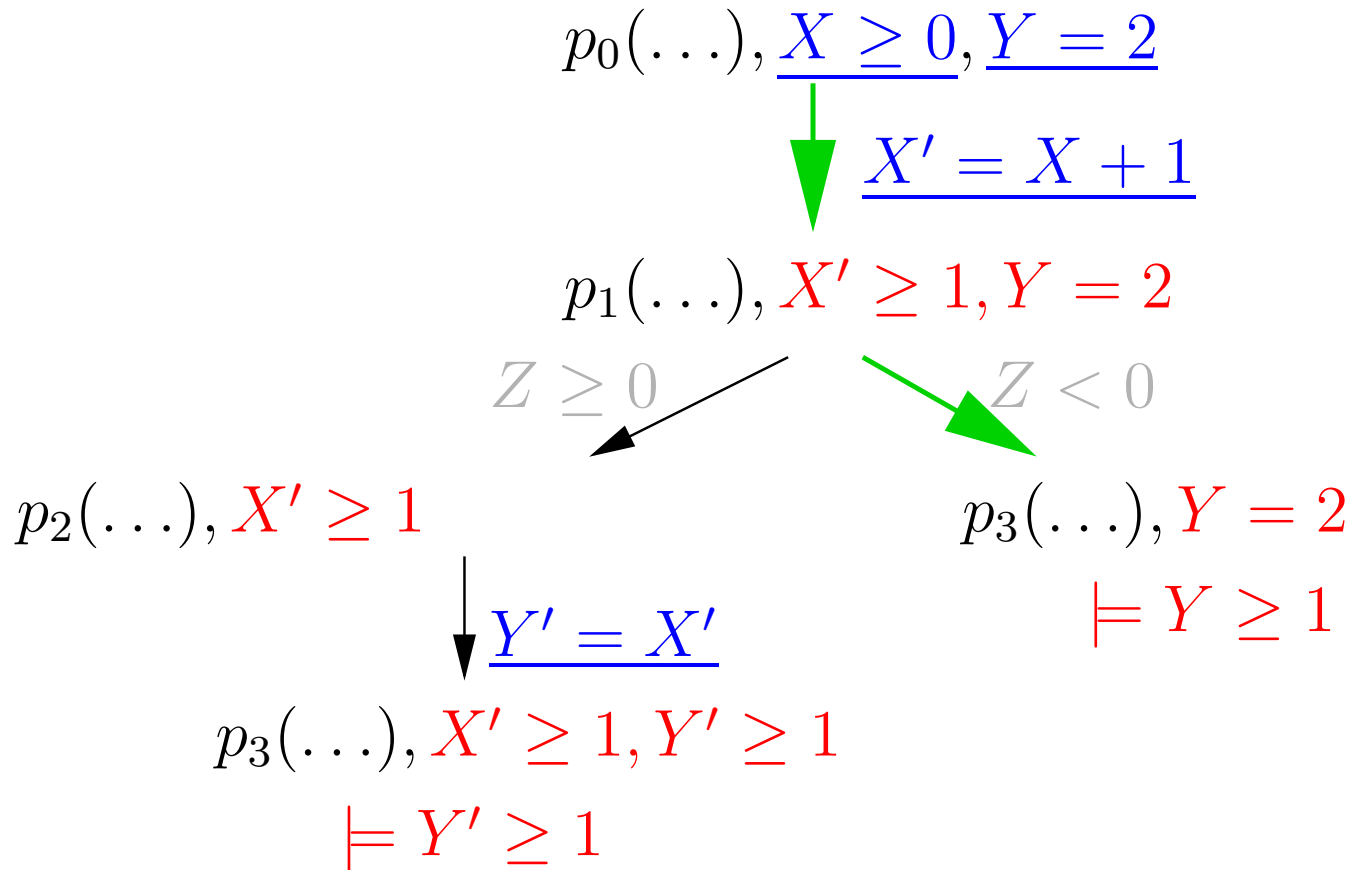


”Slackening”

Serial Constraint Replacement

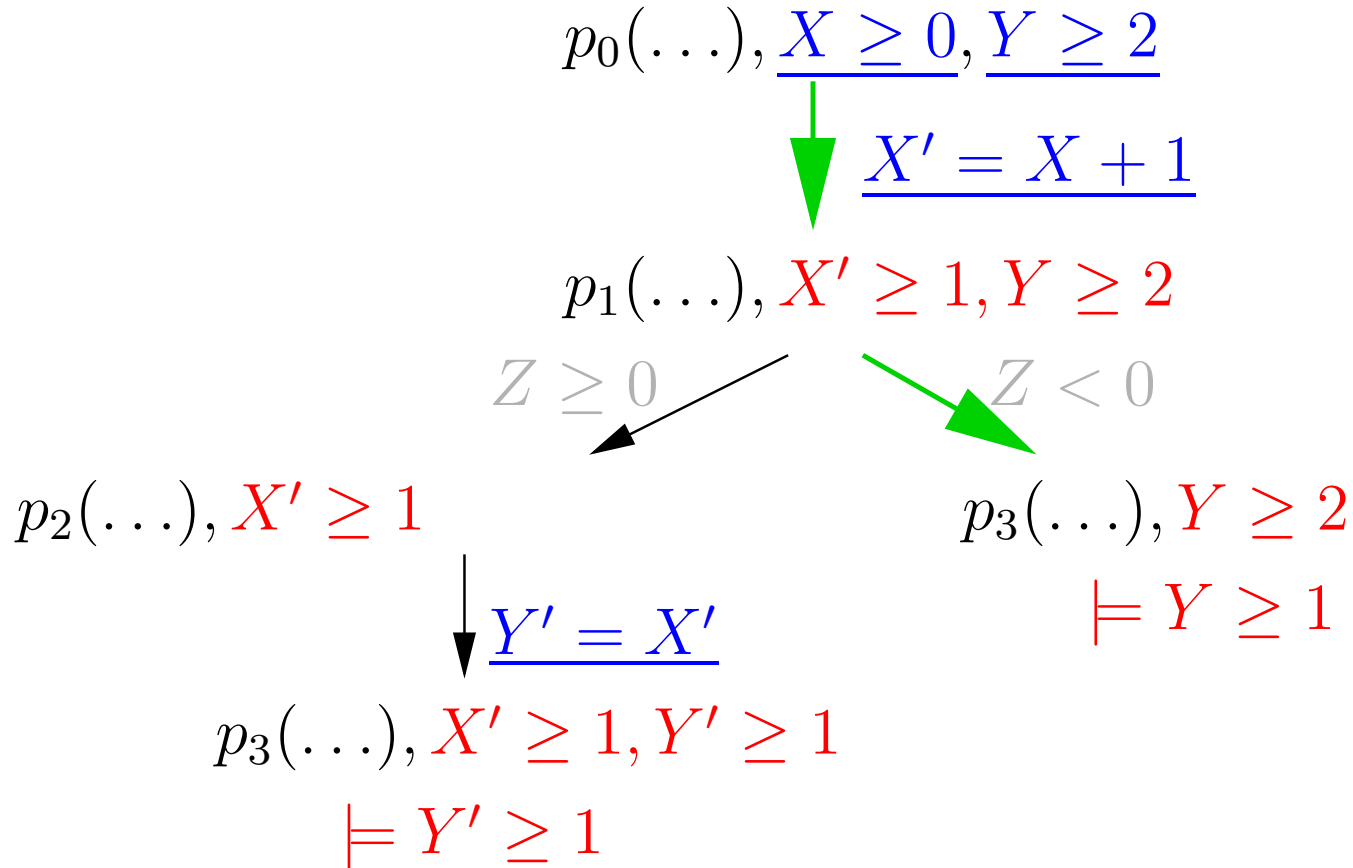


Serial Constraint Replacement



”Constraint Deletion”

Serial Constraint Replacement



”Slackening”

Outline

1. CLP for Modeling Programs
2. Interpolated Search
3. Algorithm
4. Computing Interpolants
- 5. Experimental Results**
6. Related Work and Conclusion

Array Bounds Verification

Problem	LOC	W/o Interpolation		Interpolation	
		States	Time (s)	States	Time (s)
FFT	165	2755	10.62	120	0.10
LU	102	298	0.39	138	0.17
linpack ²⁰⁰	907	6385	19.70	332	0.53
linpack ⁴⁰⁰	907	8995	151.65	867	2.47
linpack ⁶⁰⁰	907	16126	773.63	867	2.47
linpack	907	∞	∞	867	2.47

Resource Bound Verification

Problem	LOC	W/o Interpolation		Interpolation	
		States	Time (s)	States	Time (s)
decoder	27	344	1.42	160	0.49
sqrt	16	923	27.13	253	7.46
qurt	40	1104	38.65	290	11.39
janne_complex	15	1410	48.64	439	7.87
statemate ²⁰	1298	21	0.05	21	0.08
statemate ³⁰	“	1581	2.93	48	0.16
statemate ⁴⁰	“	∞	∞	71	0.24
statemate	“	∞	∞	1240	17.09

Outline

1. CLP for Modeling Programs
2. Interpolated Search
3. Algorithm
4. Computing Interpolants
5. Experimental Results
6. Related Work and Conclusion

Related Work

- Tabled resolution [Chen & Warren 1996, etc.]
- Solving *Resource-Constrained Shortest Path (RCSP)* using interpolation [Jaffar et al., AAI 2008]
- Nogood/conflict-driven/clause learning in CSP/SAT/SMT [Frost & Dechter 1994, Bayardo & Schrag 1997, Biere et al. 1999, Moskewicz et al. 2001, Silva & Sakallah 1996, etc.]
- (Bounded) model checking [McMillan 2003] and abstract interpretation (esp. CEGAR) [Clarke et al. 2003, Henzinger et al. 2004, McMillan 2006, etc.]

Related Work

- Tabled resolution [Chen & Warren 1996, etc.]
- Solving *Resource-Constrained Shortest Path (RCSP)* using interpolation [Jaffar et al., AAI 2008]
- Nogood/conflict-driven/clause learning in CSP/SAT/SMT [Frost & Dechter 1994, Bayardo & Schrag 1997, Biere et al. 1999, Moskewicz et al. 2001, Silva & Sakallah 1996, etc.]
- (Bounded) model checking [McMillan 2003] and abstract interpretation (esp. CEGAR) [Clarke et al. 2003, Henzinger et al. 2004, McMillan 2006, etc.]

More details ...

On CEGAR

- Starts abstract: accelerates optimistically but traverses spurious paths

On CEGAR

- Starts abstract: accelerates optimistically but traverses spurious paths

Our algorithm starts concrete: accelerates conservatively but avoiding spurious paths

On CEGAR

- Starts abstract: accelerates optimistically but traverses spurious paths
Our algorithm starts concrete: accelerates conservatively but avoiding spurious paths
- Employs interpolation for abstraction refinement on *spurious counterexample*

On CEGAR

- Starts abstract: accelerates optimistically but traverses spurious paths
Our algorithm starts concrete: accelerates conservatively but avoiding spurious paths
- Employs interpolation for abstraction refinement on *spurious counterexample*
We interpolate on any path

On CEGAR

- Starts abstract: accelerates optimistically but traverses spurious paths
Our algorithm starts concrete: accelerates conservatively but avoiding spurious paths
- Employs interpolation for abstraction refinement on *spurious counterexample*
We interpolate on any path
- Starting concrete allows us to collect information (e.g., bounds), in *dynamic programming* manner [Jaffar et al. 2008], not just proving safety
- On one problem, we traverse only 10% of state space

Conclusion

- Presented a general method for reducing goal space exploration using interpolation.
- Introduced serial constraint replacement for computing these interpolants, achieving efficiency while still obtaining good interpolants.
- Experimentally good space reduction.

Future Work

- Engine for *Discovery*:
 - Dependency (slicing) (draft available)
 - Lower/upper bounds
 - Various variable relationships (functional, etc.)
- Not suitable for decision algorithms (e.g., model checker)