Refining Abstract Interpretation-based Approximations with Constraint Solvers

Olivier Ponsini
Claude Michel, Michel Rueher

University of Nice-Sophia Antipolis / I3S – CNRS
France

Numerical Software Verification July 2011

Outline

Programs with floating-point computations

Context

- Embedded Systems (transportation, nuclear energy...) rely more and more on floating-point computations
- C language is widely used for such applications
- Floats are an additional possible source of errors (floating-point arithmetic pitfalls)

Real numbers versus floating-point numbers semantics

Programs are run over the floats but:

- Specification (properties of programs) is written with the semantics of reals "in mind"
- Programs are sometimes written with the semantics of reals "in mind"
- Differences between semantics may reveal problems with floats

$\rightarrow$ Approximations are over the floats and over the reals

Objective & Approach

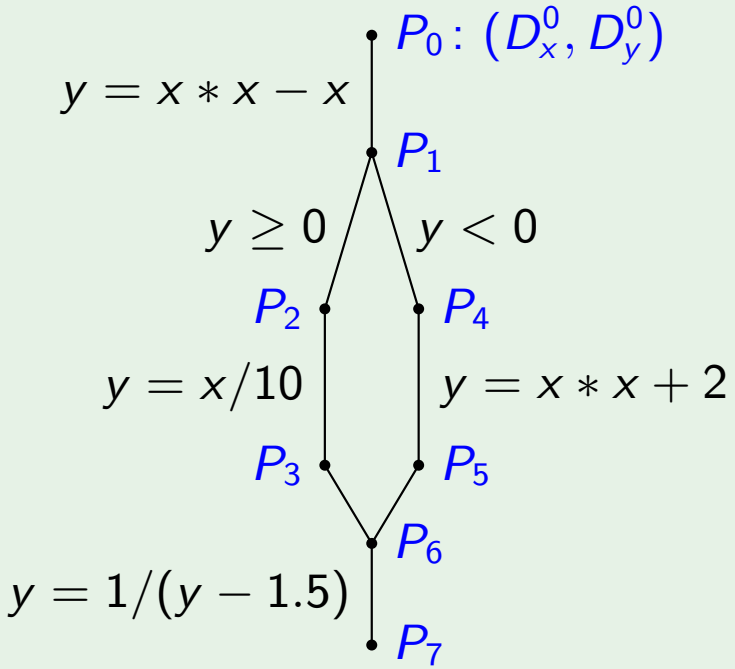
- Objective: Refine the approximations of the domains of the program variables computed by abstract interpretation

- Approach: Use local consistencies to "shave" the domains
  - More accurate "collecting semantics"
  - Complementary "abstract domains"

Example (Abstract Interpretation)

Abstract domain of intervals

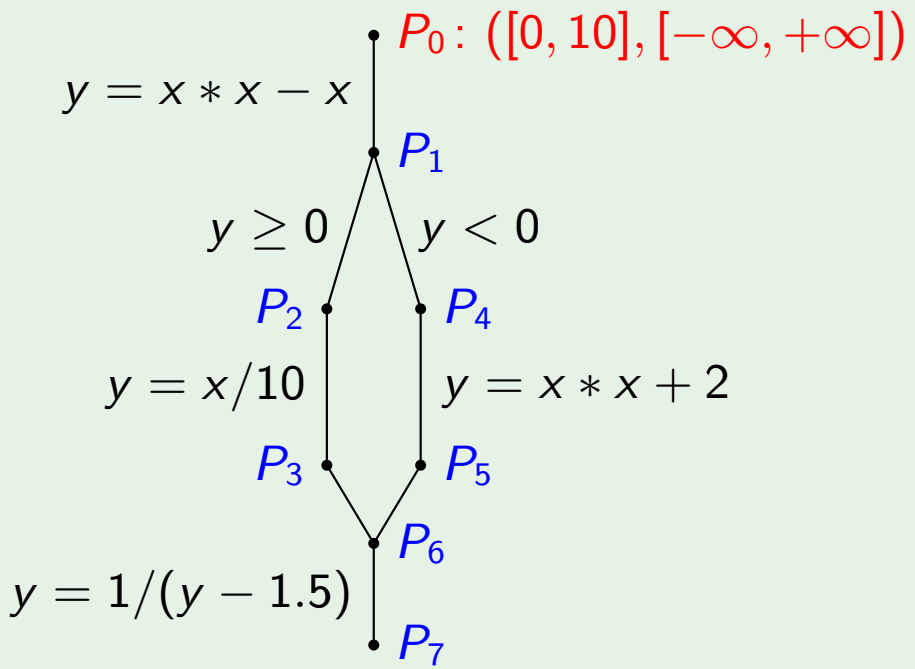
```
float x = [0,10];
float y = x*x - x;
if (y >= 0)
  y = x/10;
else
  y = x*x + 2;
y = 1 / (y-1.5);
```



Example (Abstract Interpretation)

Abstract domain of intervals

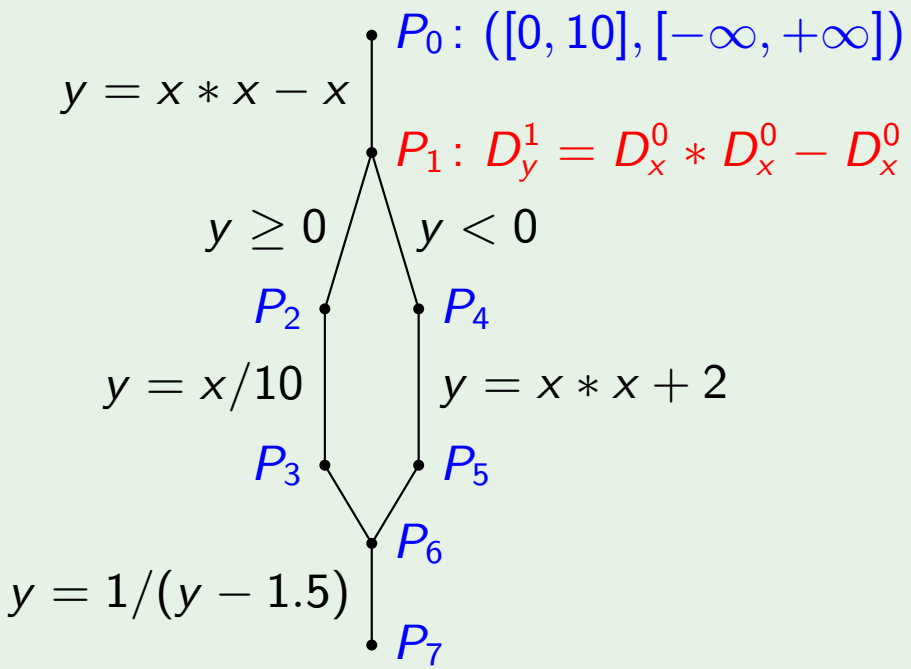
```
float x = [0,10];
float y = x*x - x;
if (y >= 0)
  y = x/10;
else
  y = x*x + 2;
y = 1 / (y-1.5);
```



Example (Abstract Interpretation)

Abstract domain of intervals

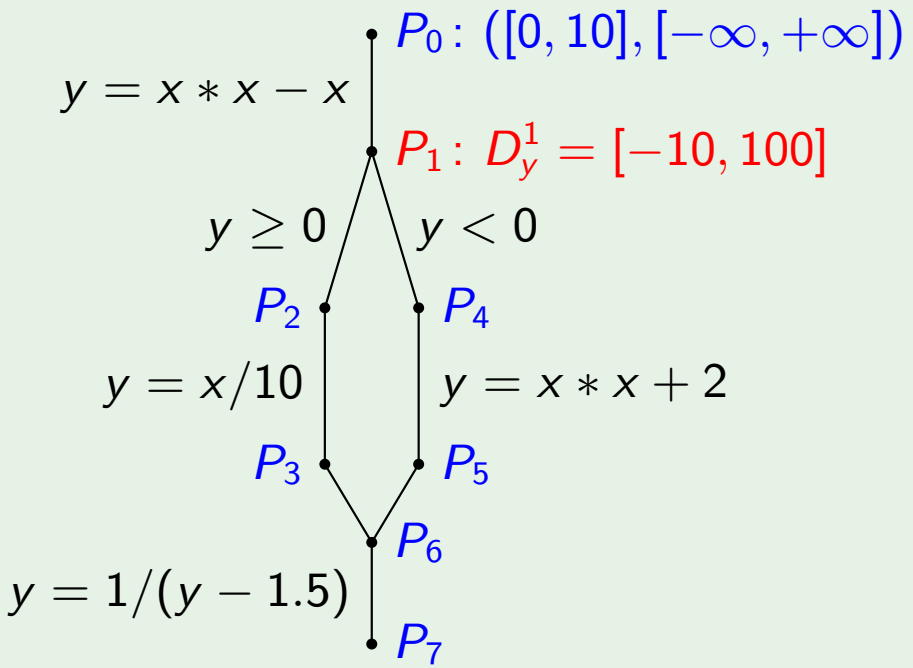
```
float x = [0,10];
float y = x*x - x;
if (y >= 0)
  y = x/10;
else
  y = x*x + 2;
y = 1 / (y-1.5);
```



Example (Abstract Interpretation)

Abstract domain of intervals

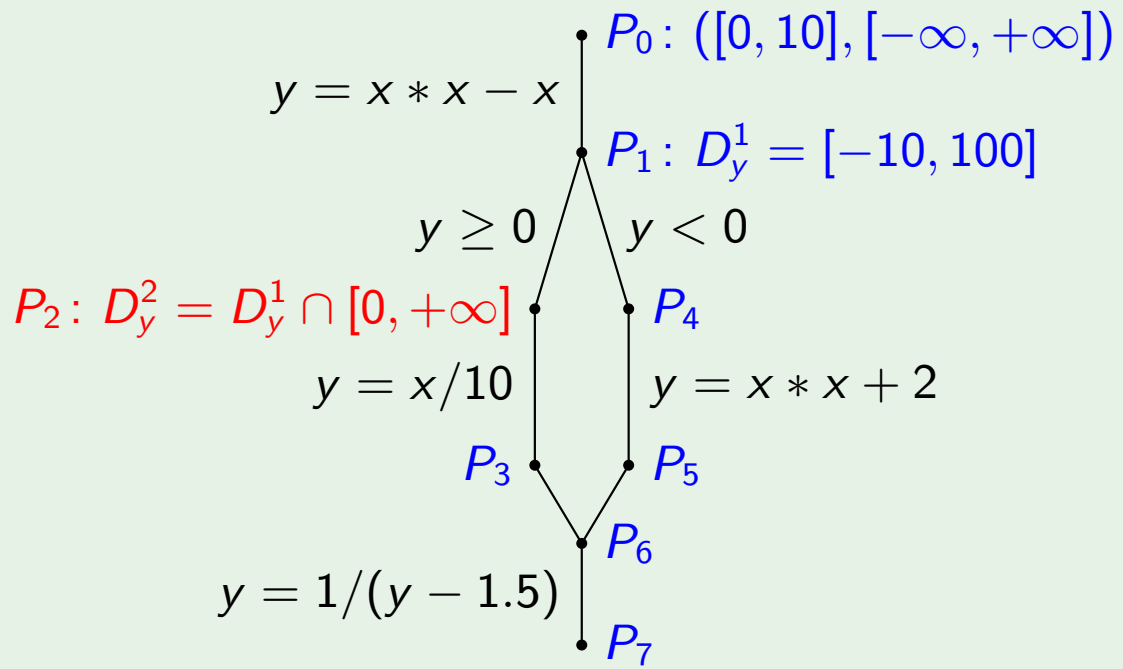
```
float x = [0,10];
float y = x*x - x;
if (y >= 0)
  y = x/10;
else
  y = x*x + 2;
y = 1 / (y-1.5);
```



Example (Abstract Interpretation)

Abstract domain of intervals

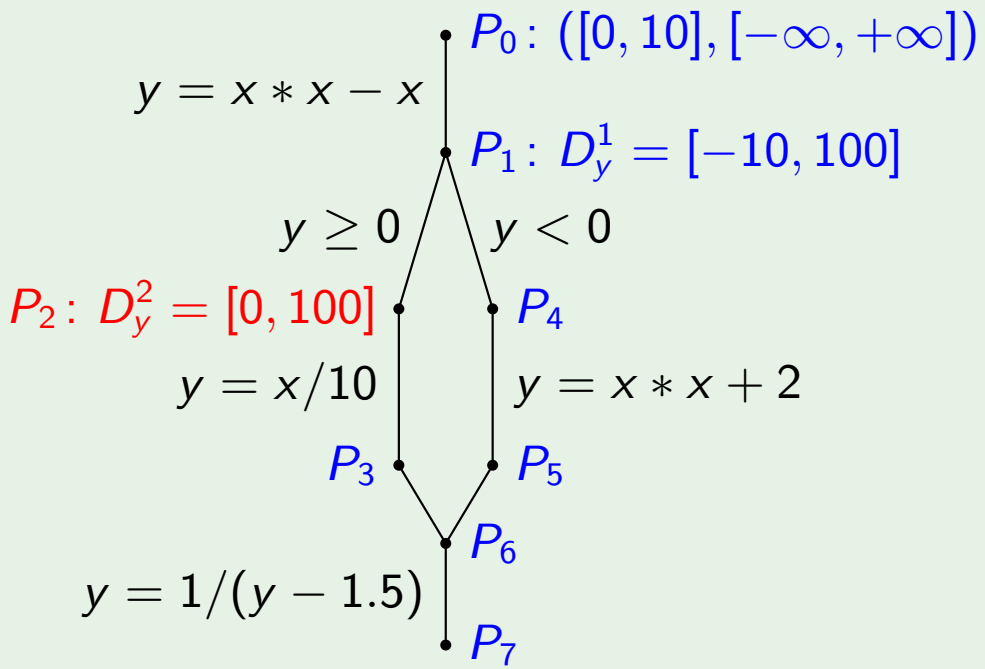
```
float x = [0,10];
float y = x*x - x;
if (y >= 0)
  y = x/10;
else
  y = x*x + 2;
y = 1 / (y-1.5);
```



Example (Abstract Interpretation)

Abstract domain of intervals

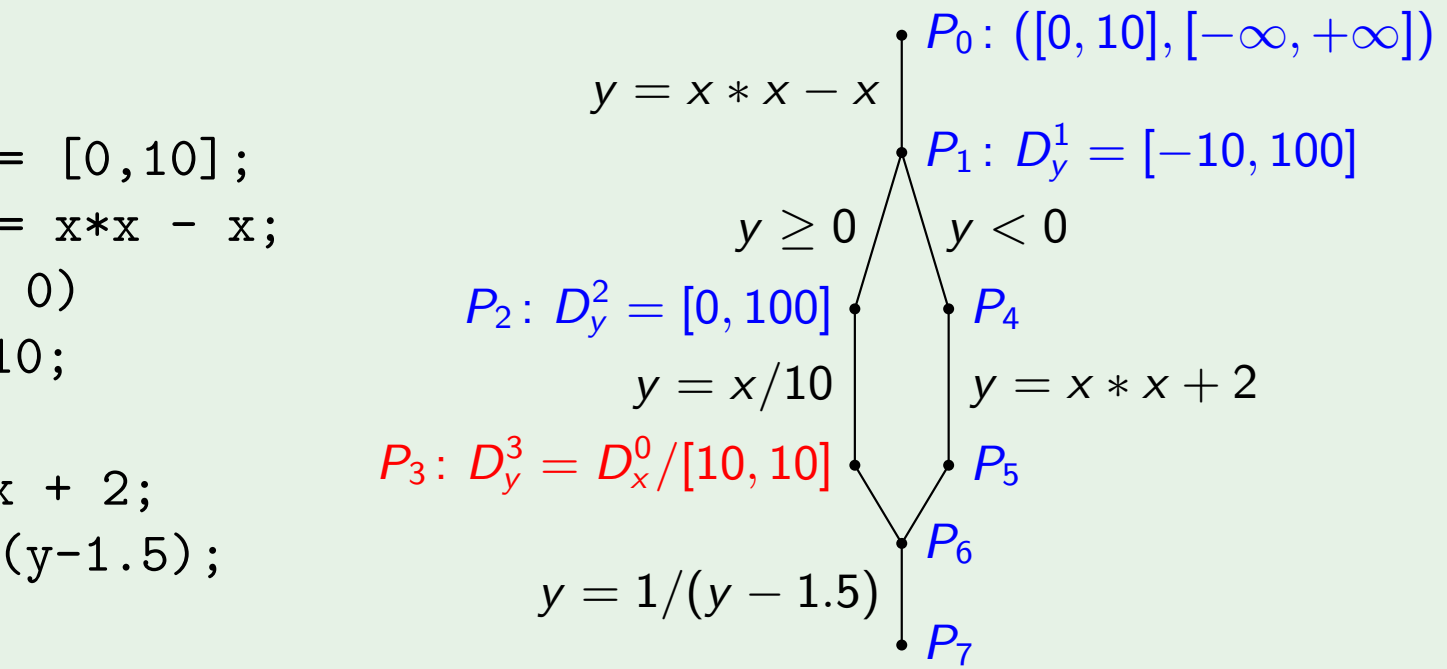
```
float x = [0,10];
float y = x*x - x;
if (y >= 0)
  y = x/10;
else
  y = x*x + 2;
y = 1 / (y-1.5);
```



Example (Abstract Interpretation)

Abstract domain of intervals

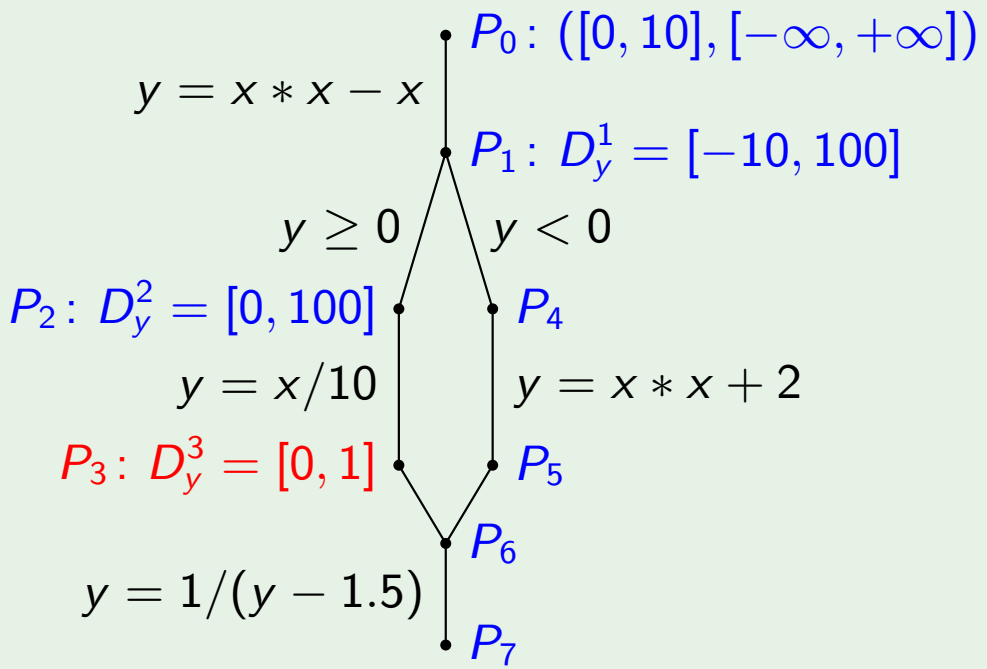
```
float x = [0,10];
float y = x*x - x;
if (y >= 0)
  y = x/10;
else
  y = x*x + 2;
y = 1 / (y-1.5);
```



Example (Abstract Interpretation)

Abstract domain of intervals

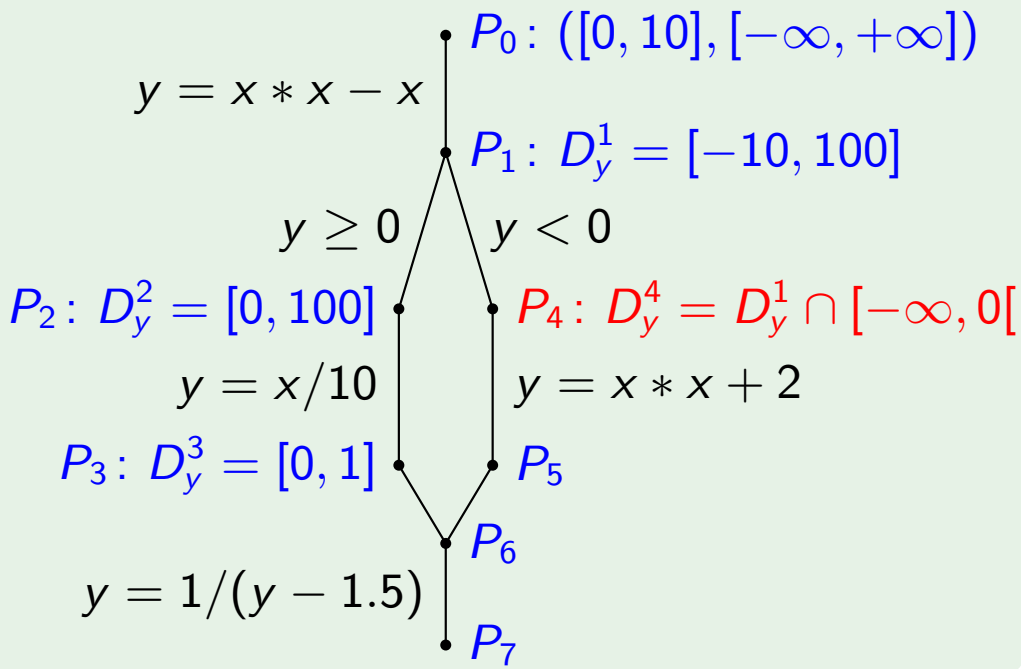
```
float x = [0,10];
float y = x*x - x;
if (y >= 0)
  y = x/10;
else
  y = x*x + 2;
y = 1 / (y-1.5);
```



Example (Abstract Interpretation)

Abstract domain of intervals

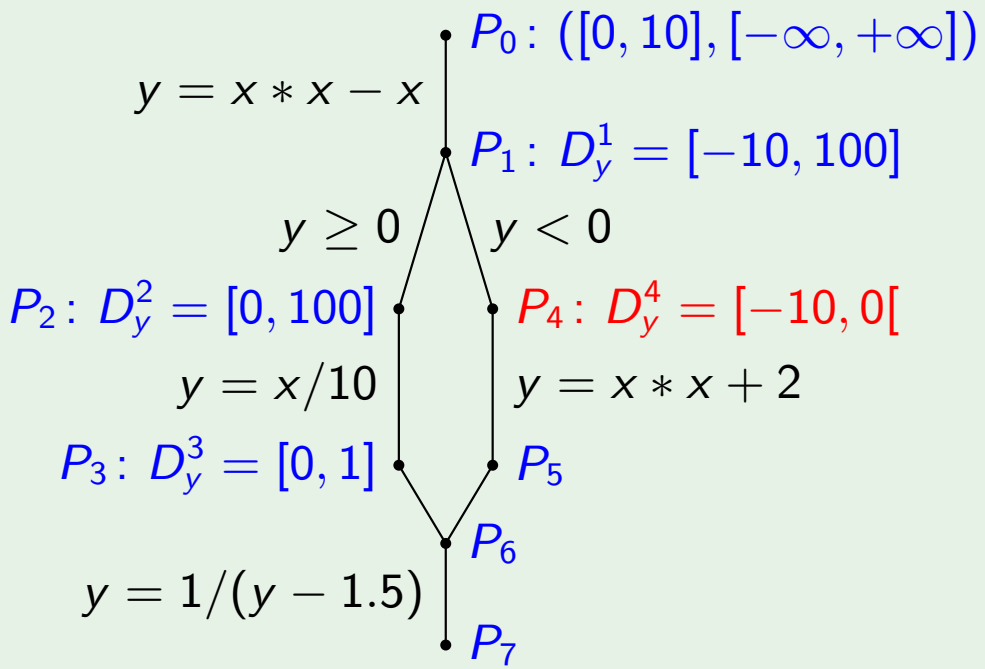
```
float x = [0,10];
float y = x*x - x;
if (y >= 0)
  y = x/10;
else
  y = x*x + 2;
y = 1 / (y-1.5);
```



Example (Abstract Interpretation)

Abstract domain of intervals

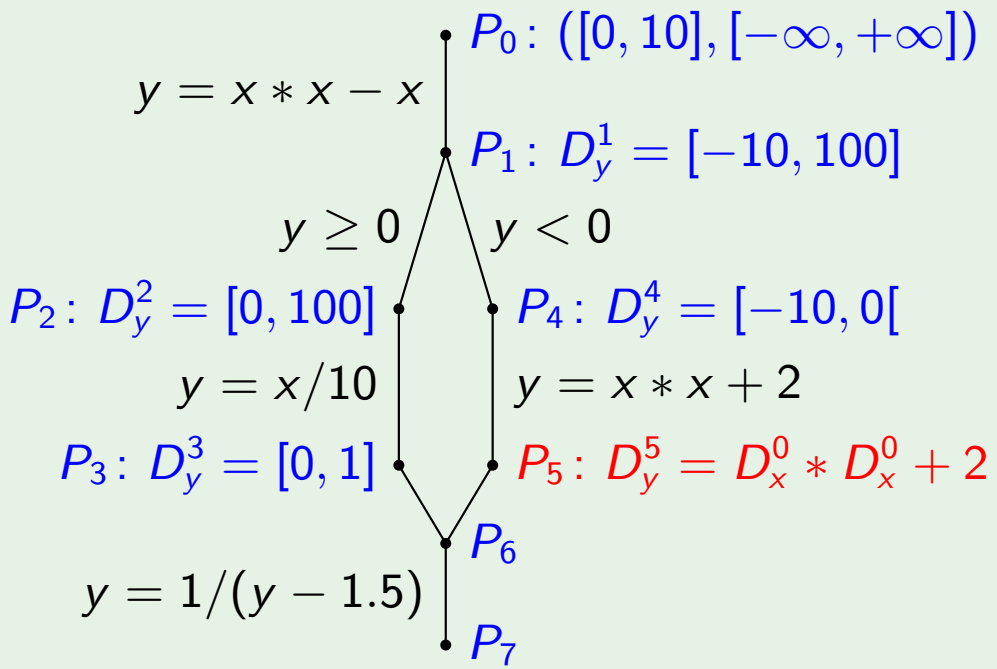
```
float x = [0,10];
float y = x*x - x;
if (y >= 0)
  y = x/10;
else
  y = x*x + 2;
y = 1 / (y-1.5);
```



Example (Abstract Interpretation)

Abstract domain of intervals

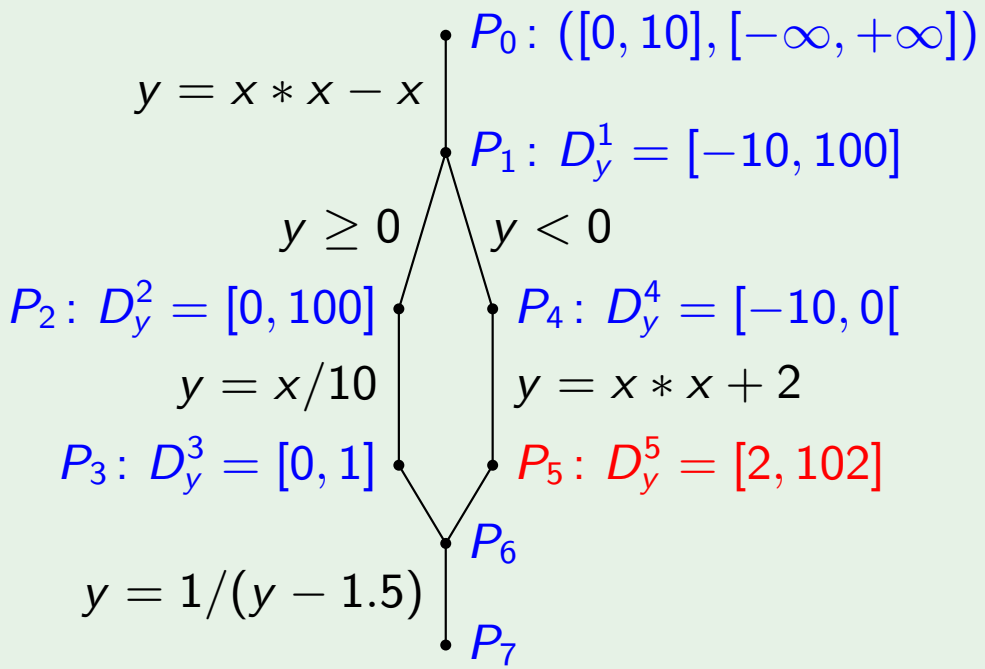
```
float x = [0,10];
float y = x*x - x;
if (y >= 0)
  y = x/10;
else
  y = x*x + 2;
y = 1 / (y-1.5);
```



Example (Abstract Interpretation)

Abstract domain of intervals

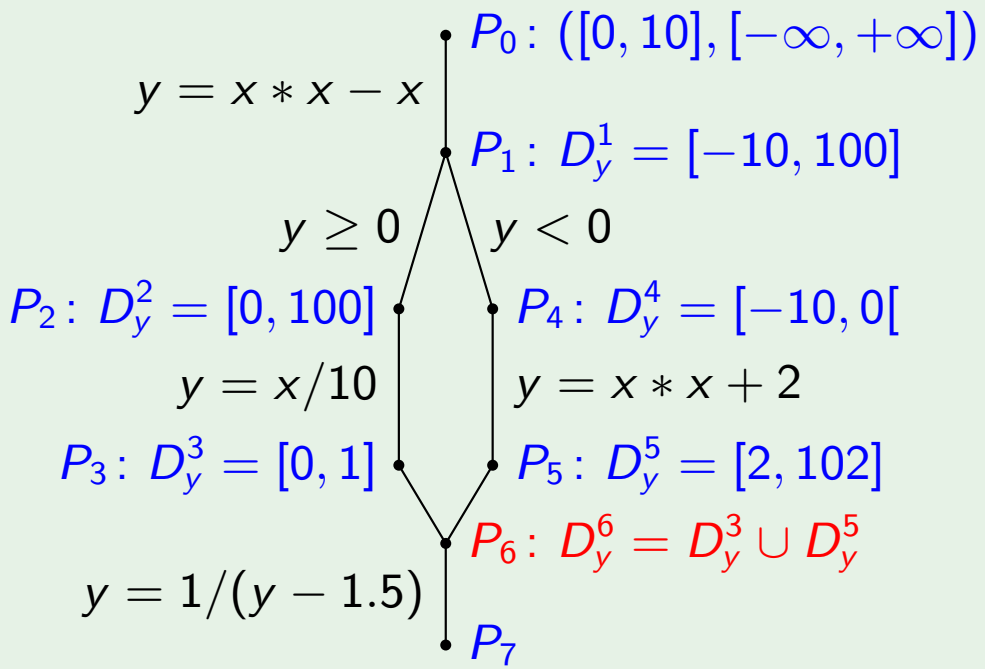
```
float x = [0,10];
float y = x*x - x;
if (y >= 0)
  y = x/10;
else
  y = x*x + 2;
y = 1 / (y-1.5);
```



Example (Abstract Interpretation)

Abstract domain of intervals

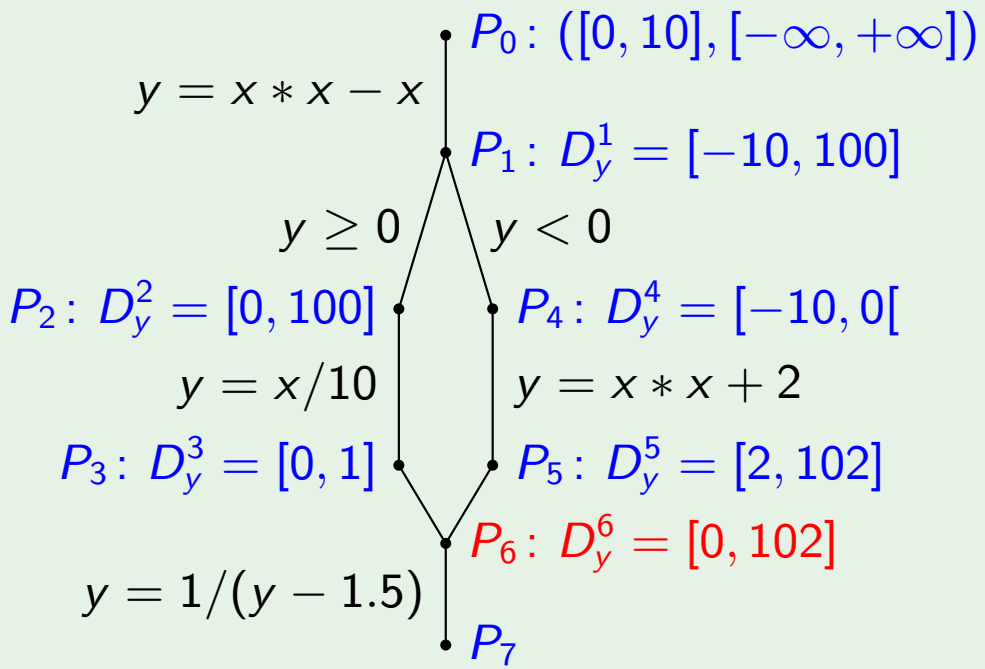
```
float x = [0,10];
float y = x*x - x;
if (y >= 0)
  y = x/10;
else
  y = x*x + 2;
y = 1 / (y-1.5);
```



Example (Abstract Interpretation)

Abstract domain of intervals

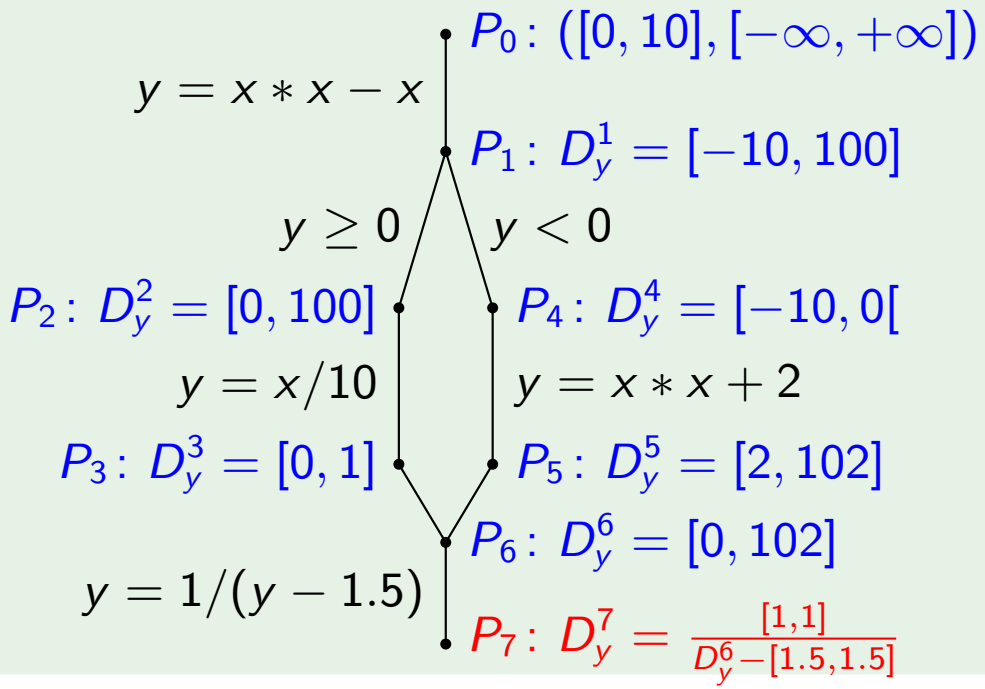
```
float x = [0,10];
float y = x*x - x;
if (y >= 0)
  y = x/10;
else
  y = x*x + 2;
y = 1 / (y-1.5);
```



Example (Abstract Interpretation)

Abstract domain of intervals

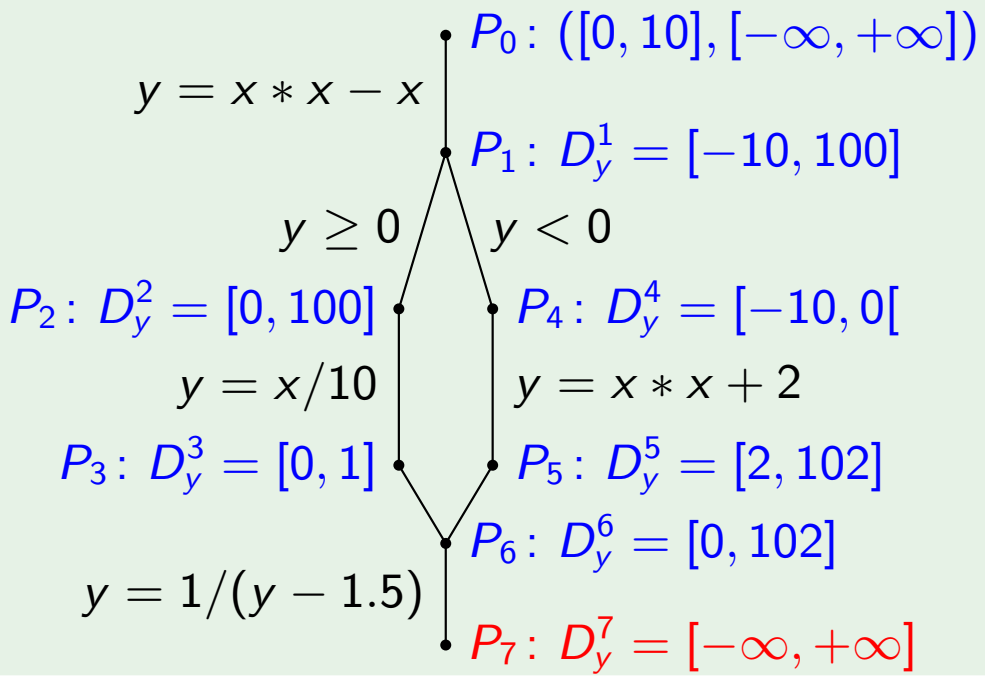
```
float x = [0,10];
float y = x*x - x;
if (y >= 0)
  y = x/10;
else
  y = x*x + 2;
y = 1 / (y-1.5);
```



Example (Abstract Interpretation)

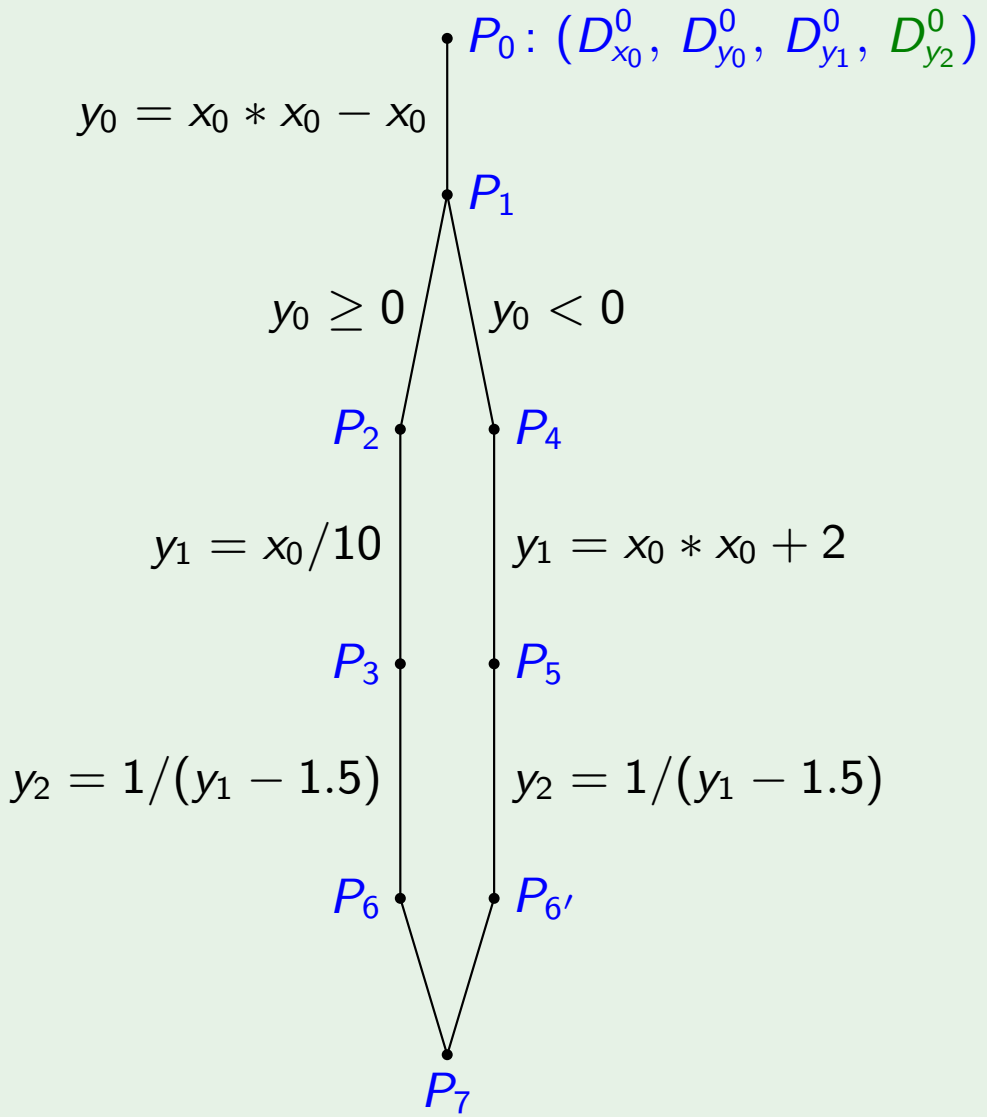
Abstract domain of intervals

```
float x = [0,10];
float y = x*x - x;
if (y >= 0)
  y = x/10;
else
  y = x*x + 2;
y = 1 / (y-1.5);
```



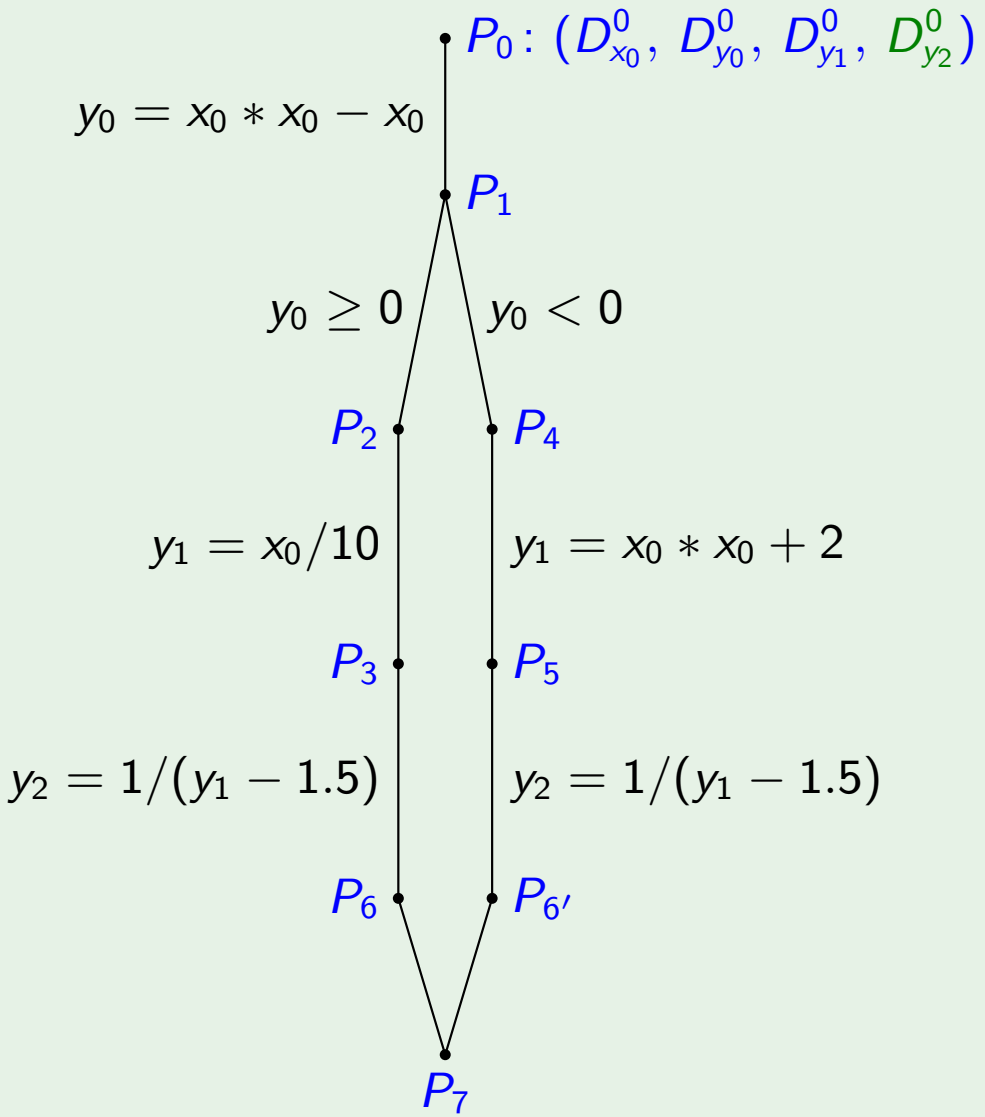
Example (constraint-based local consistencies)

```
float x  = [0,10];
float y  = x *x  - x ;
if (y  >= 0)
  y  = x /10;
else
  y  = x *x  + 2;
y  = 1 / (y -1.5);
```



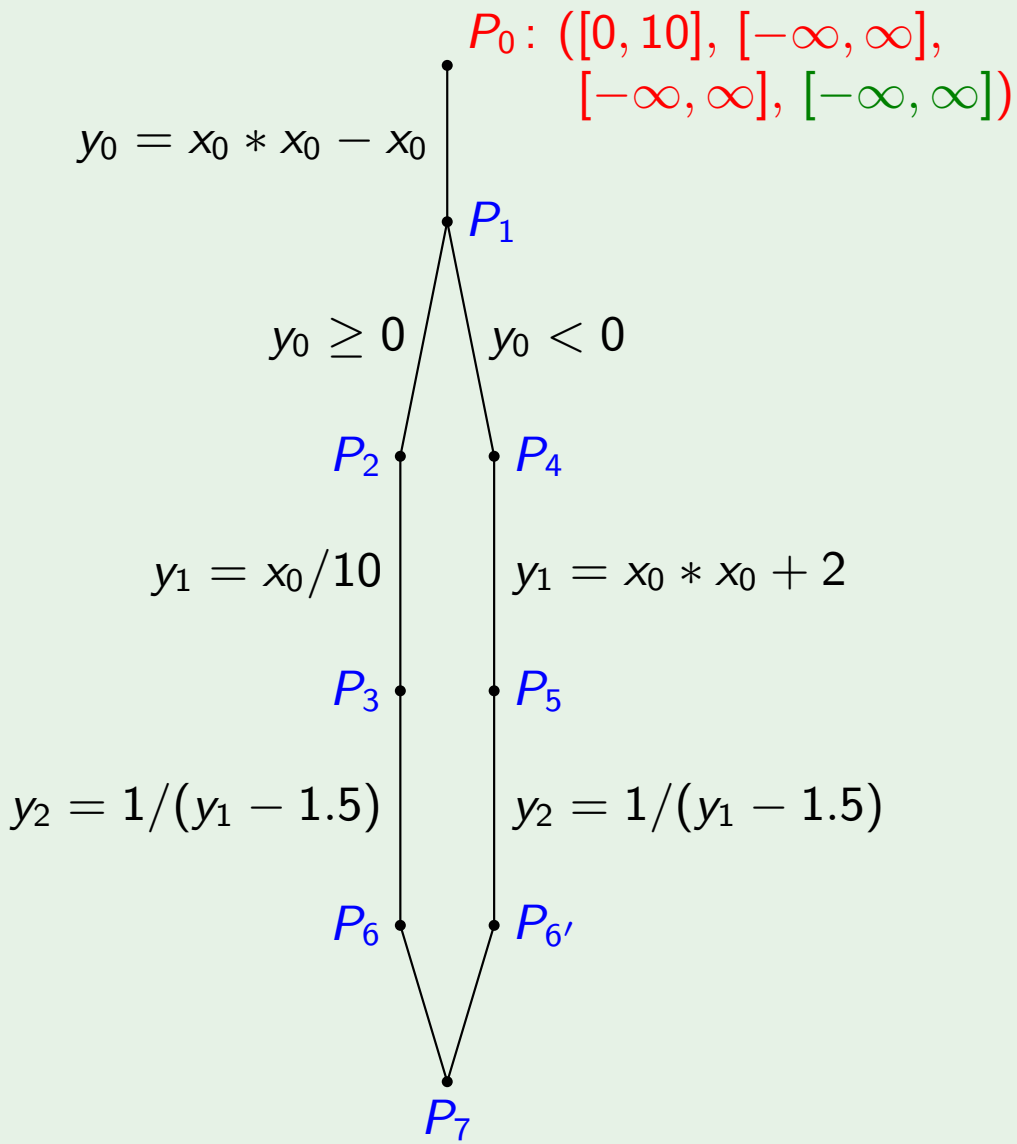
Example (constraint-based local consistencies)

```
float x0 = [0,10];
float y0 = x0*x0 - x0;
if (y0 >= 0)
  y1 = x0/10;
else
  y1 = x0*x0 + 2;
y2 = 1 / (y1-1.5);
```



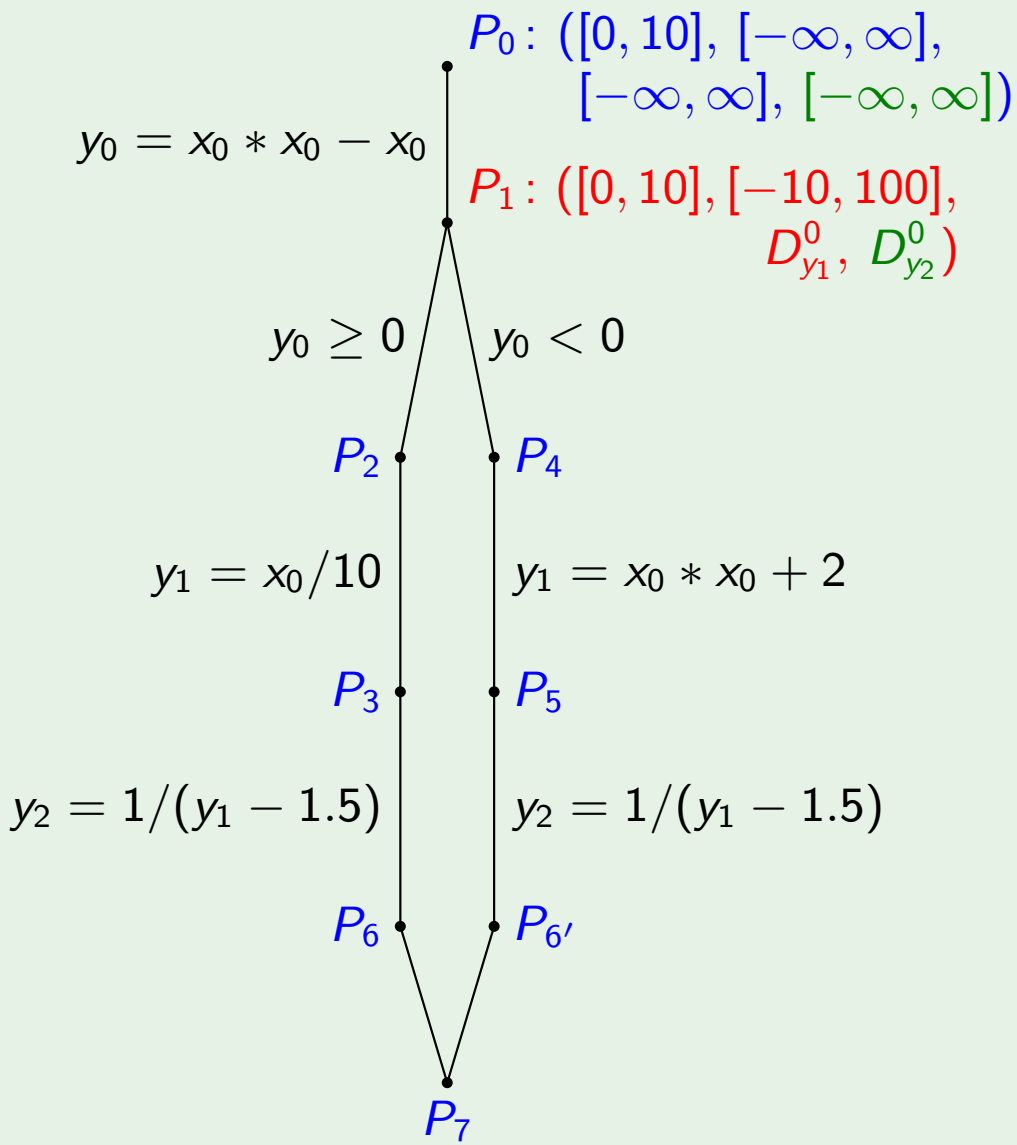
Example (constraint-based local consistencies)

```
float x0 = [0,10];
float y0 = x0*x0 - x0;
if (y0 >= 0)
  y1 = x0/10;
else
  y1 = x0*x0 + 2;
y2 = 1 / (y1-1.5);
```



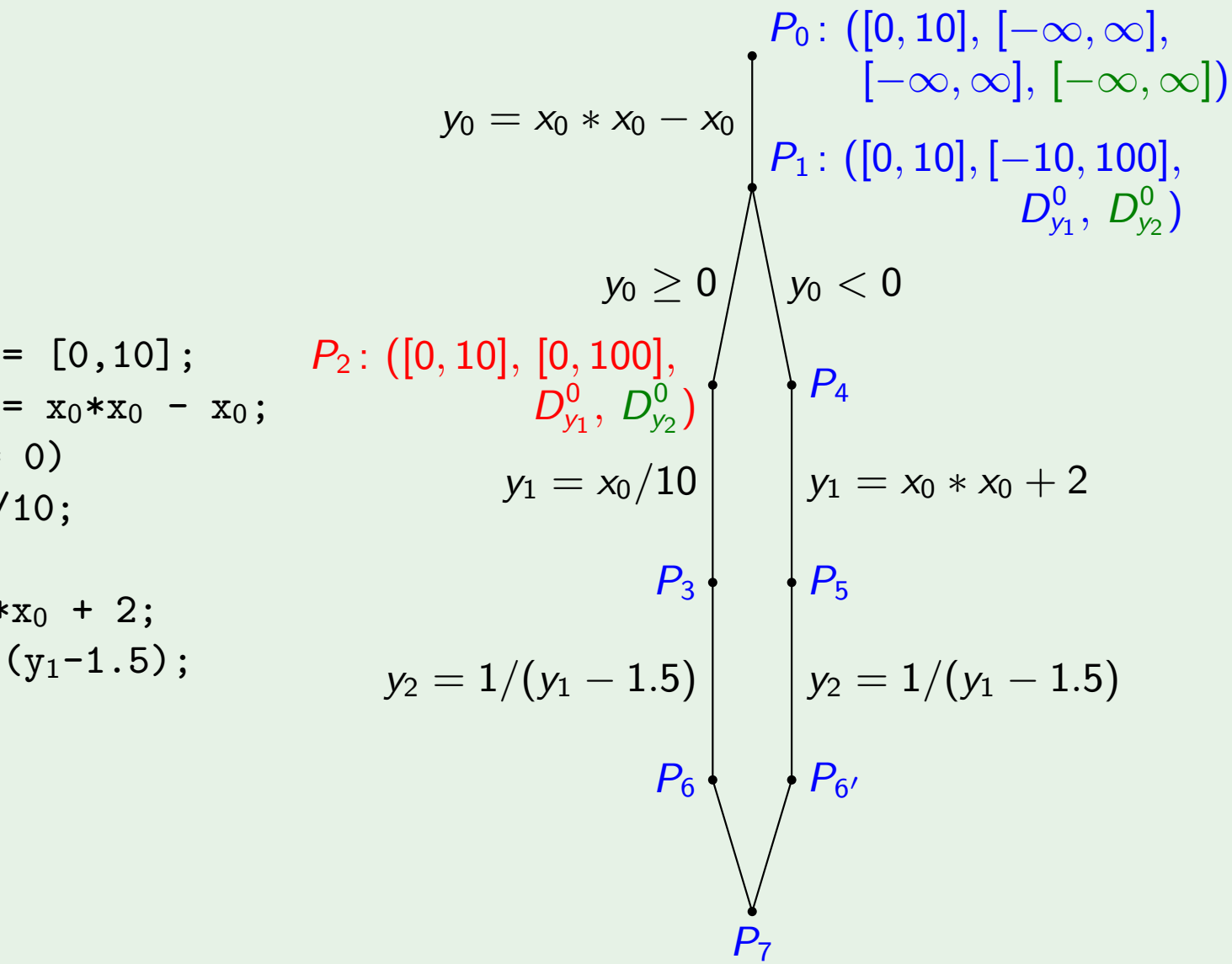
Example (constraint-based local consistencies)

```
float x0 = [0,10];
float y0 = x0*x0 - x0;
if (y0 >= 0)
  y1 = x0/10;
else
  y1 = x0*x0 + 2;
y2 = 1 / (y1-1.5);
```



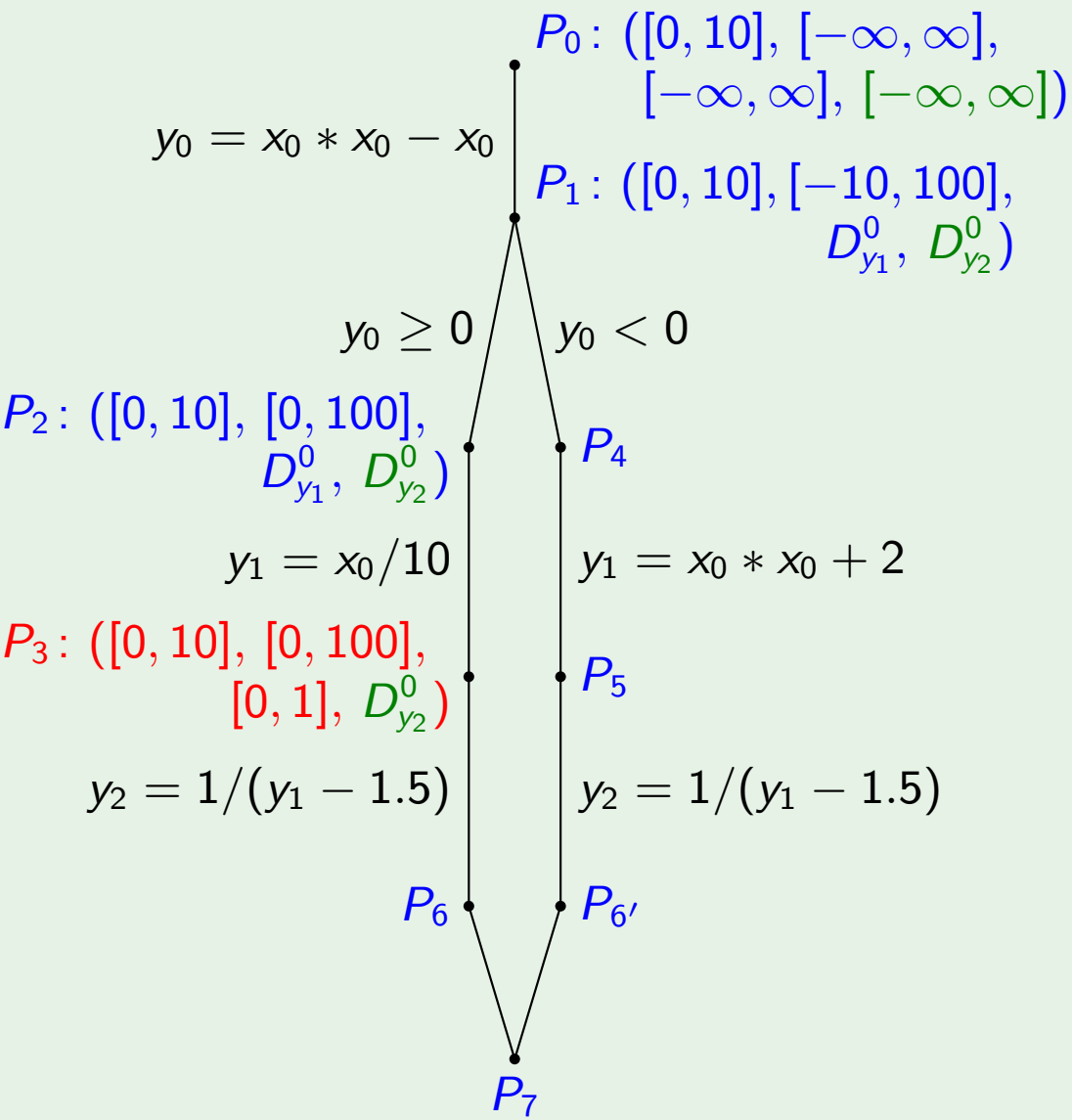
Example (constraint-based local consistencies)

```
float x0 = [0,10];
float y0 = x0*x0 - x0;
if (y0 >= 0)
  y1 = x0/10;
else
  y1 = x0*x0 + 2;
y2 = 1 / (y1-1.5);
```



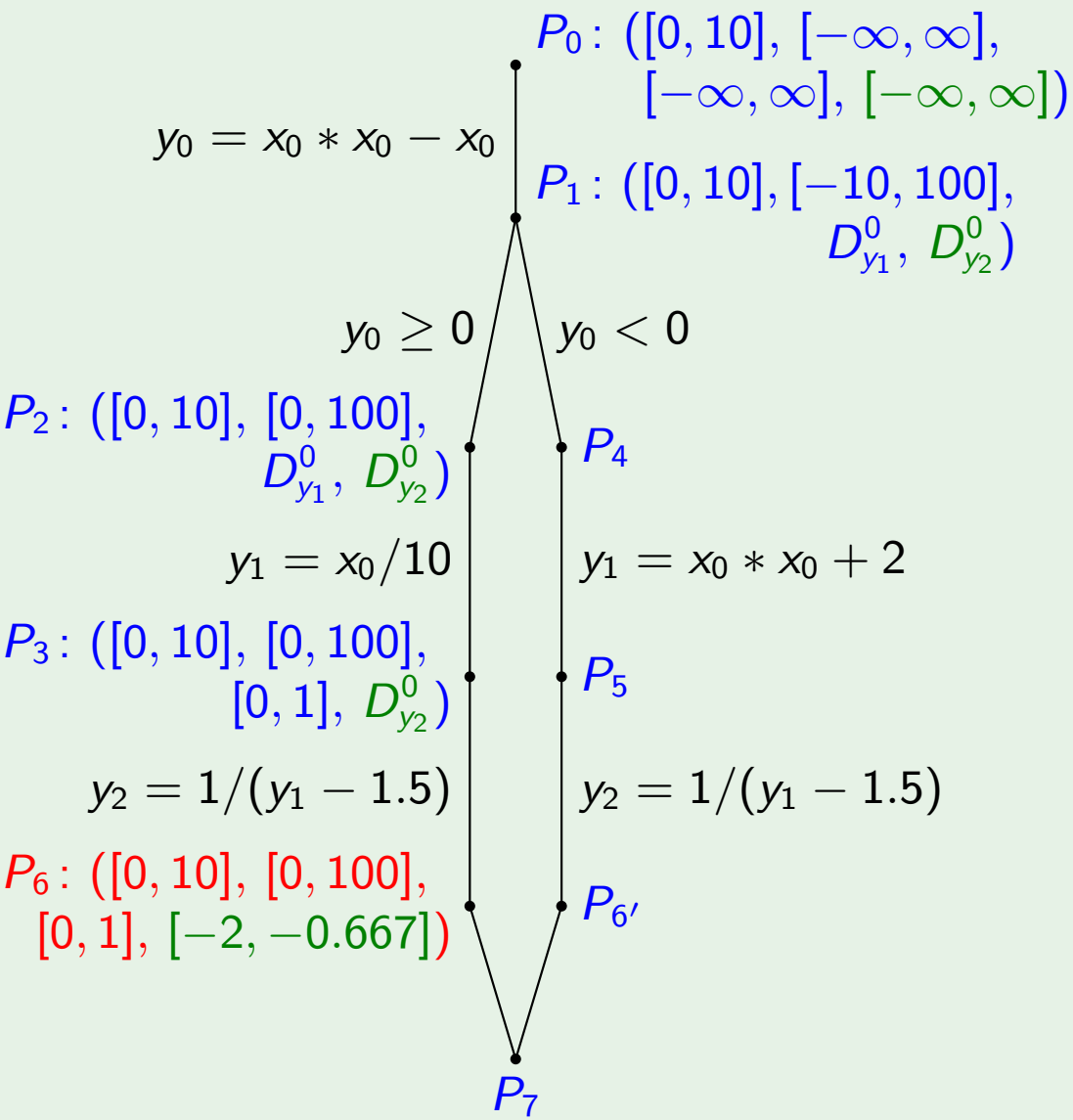
Example (constraint-based local consistencies)

```
float x0 = [0,10];
float y0 = x0*x0 - x0;
if (y0 >= 0)
  y1 = x0/10;
else
  y1 = x0*x0 + 2;
y2 = 1 / (y1-1.5);
```



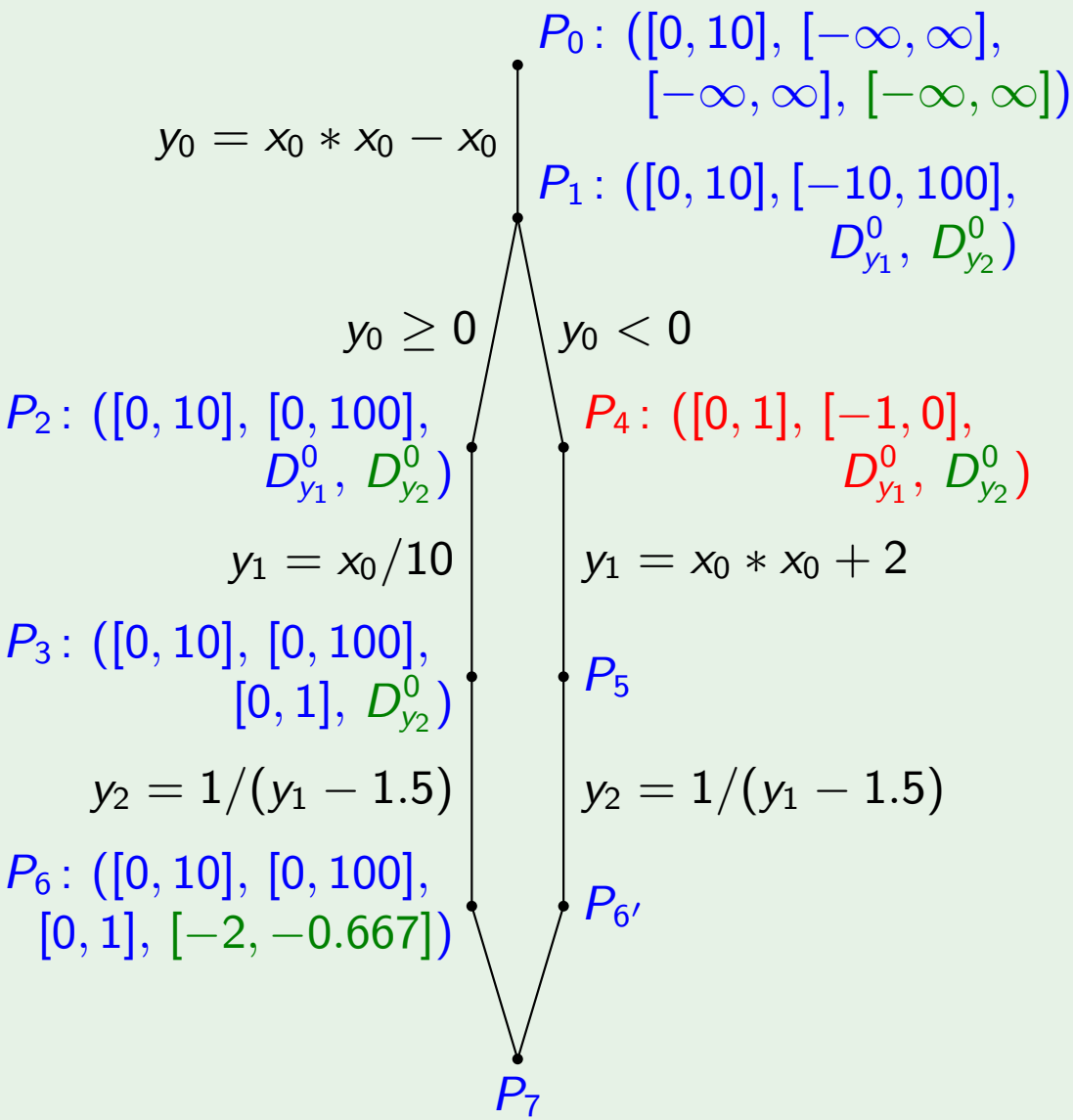
Example (constraint-based local consistencies)

```
float x0 = [0,10];
float y0 = x0*x0 - x0;
if (y0 >= 0)
  y1 = x0/10;
else
  y1 = x0*x0 + 2;
y2 = 1 / (y1-1.5);
```



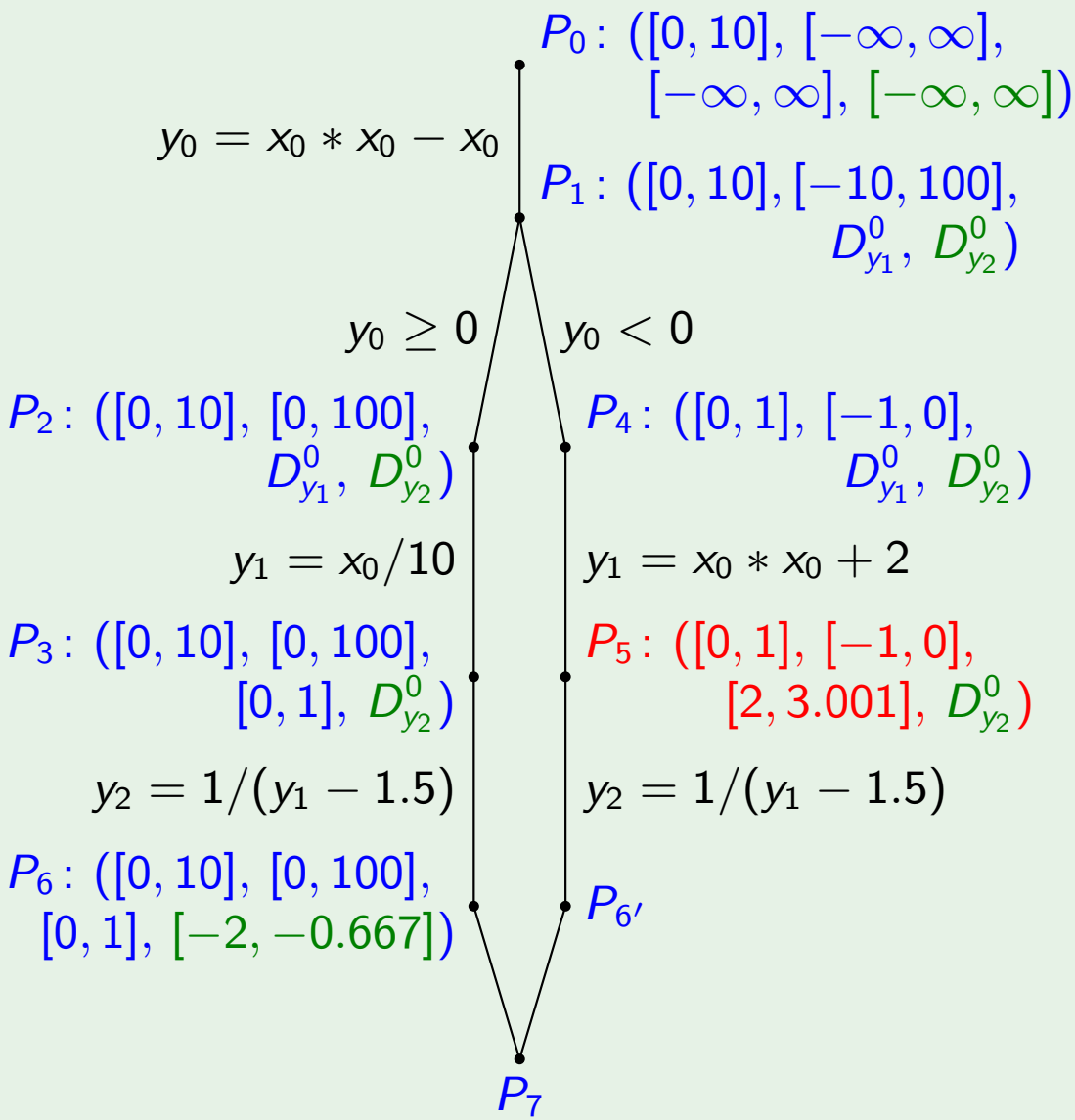
Example (constraint-based local consistencies)

```
float x0 = [0,10];
float y0 = x0*x0 - x0;
if (y0 >= 0)
  y1 = x0/10;
else
  y1 = x0*x0 + 2;
y2 = 1 / (y1-1.5);
```



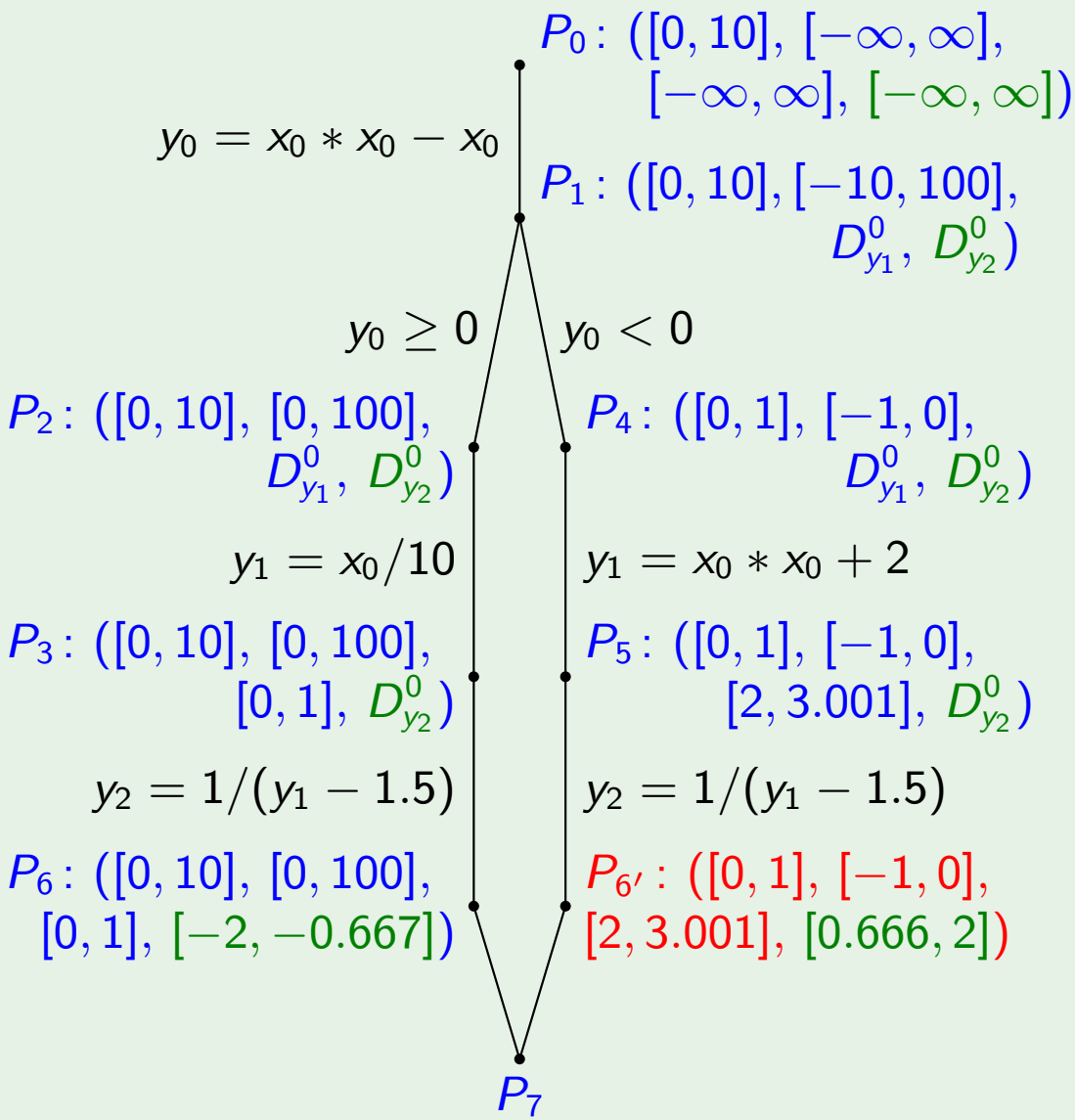
Example (constraint-based local consistencies)

```
float x0 = [0,10];
float y0 = x0*x0 - x0;
if (y0 >= 0)
  y1 = x0/10;
else
  y1 = x0*x0 + 2;
y2 = 1 / (y1-1.5);
```



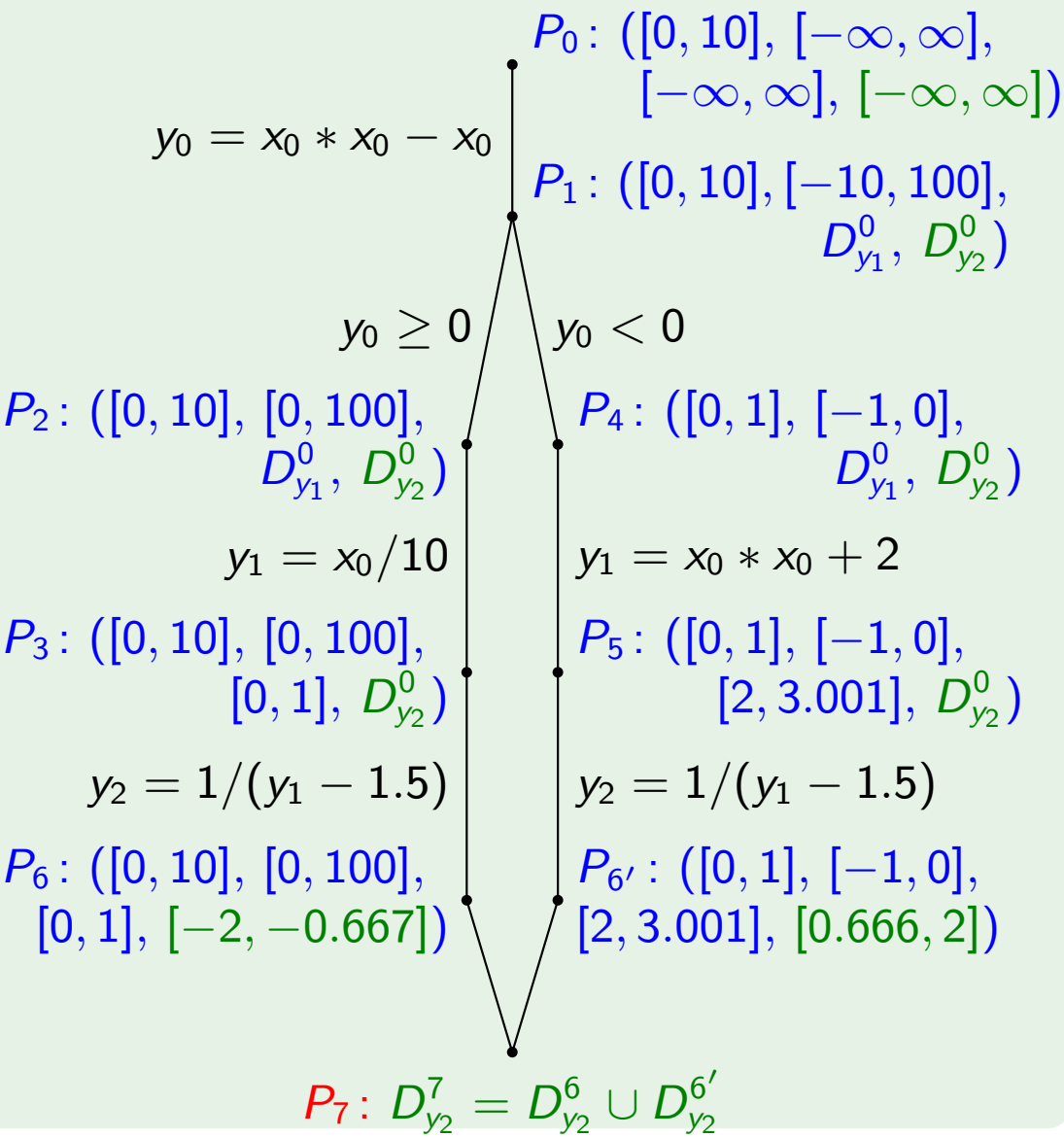
Example (constraint-based local consistencies)

```
float x0 = [0,10];
float y0 = x0*x0 - x0;
if (y0 >= 0)
  y1 = x0/10;
else
  y1 = x0*x0 + 2;
y2 = 1 / (y1-1.5);
```



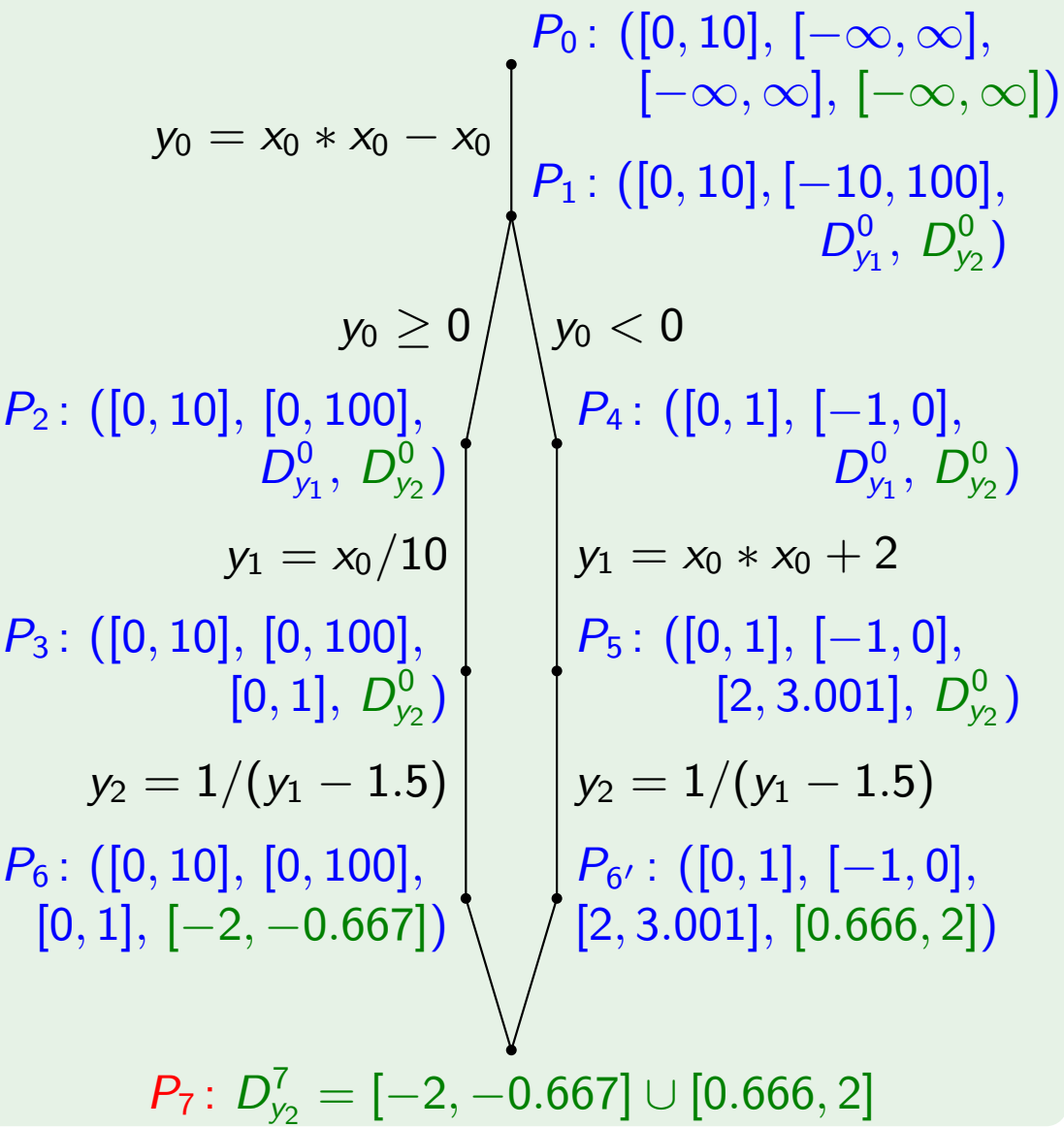
Example (constraint-based local consistencies)

```
float x0 = [0,10];
float y0 = x0*x0 - x0;
if (y0 >= 0)
  y1 = x0/10;
else
  y1 = x0*x0 + 2;
y2 = 1 / (y1-1.5);
```



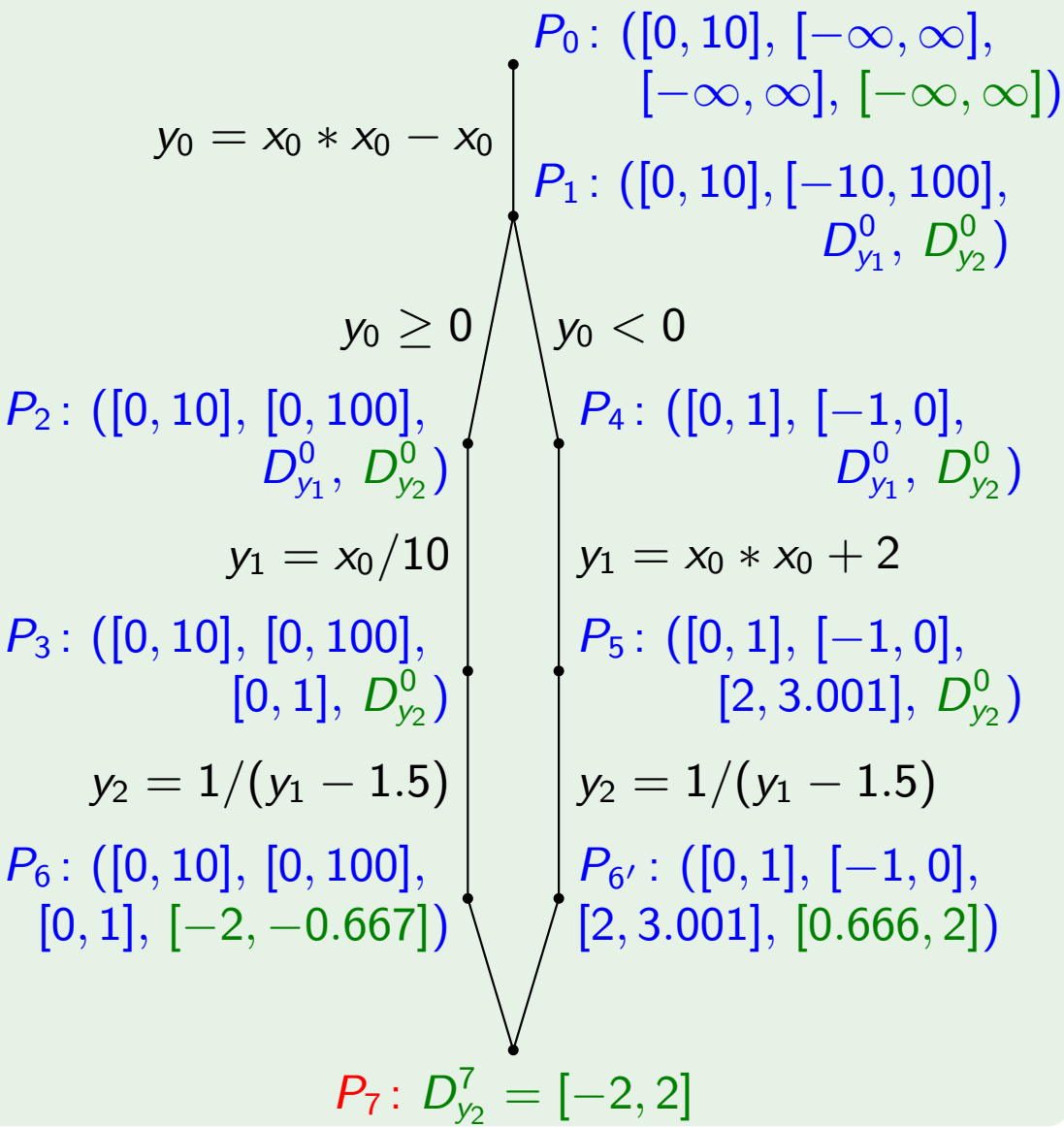
Example (constraint-based local consistencies)

```
float x0 = [0,10];
float y0 = x0*x0 - x0;
if (y0 >= 0)
  y1 = x0/10;
else
  y1 = x0*x0 + 2;
y2 = 1 / (y1-1.5);
```



Example (constraint-based local consistencies)

```
float x0 = [0,10];
float y0 = x0*x0 - x0;
if (y0 >= 0)
  y1 = x0/10;
else
  y1 = x0*x0 + 2;
y2 = 1 / (y1-1.5);
```



Fluctuat

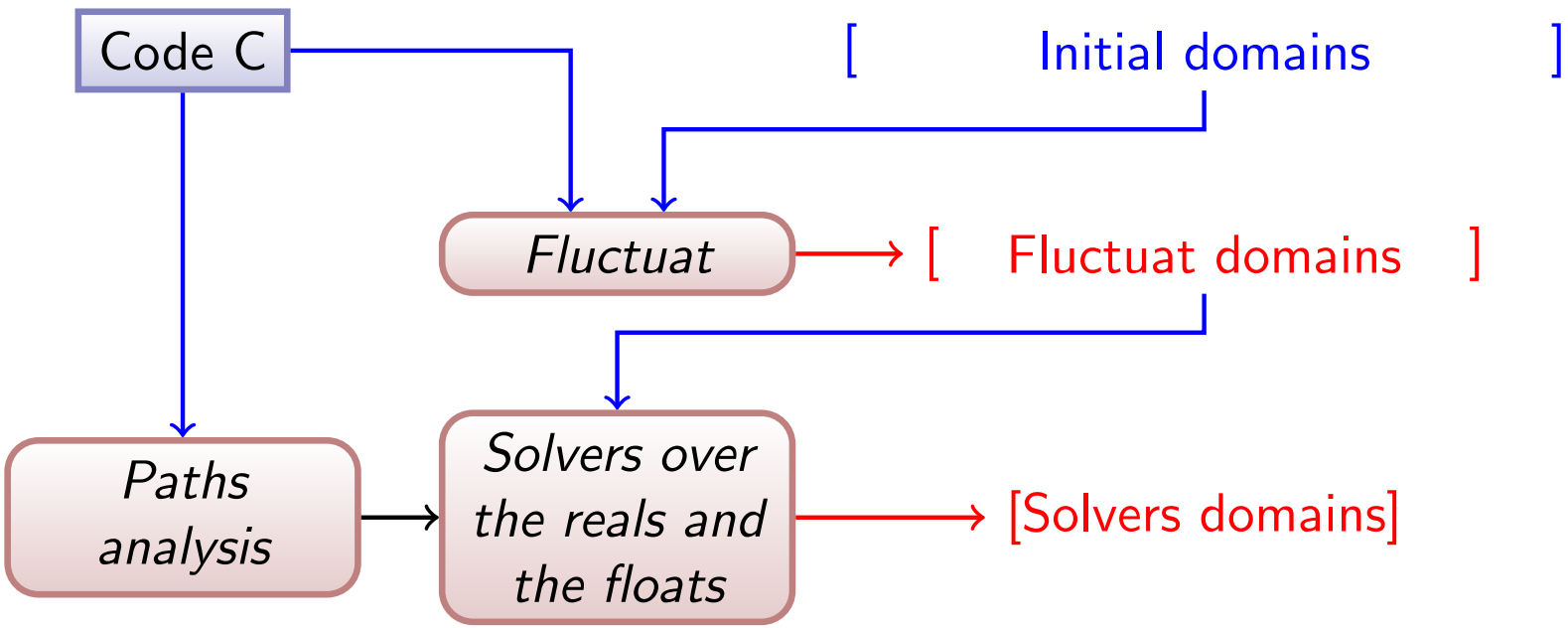
Static analyzer of C programs based on Abstract Interpretation: estimate rounding errors and their propagation

- Programs are considered both:
  - As a specification over the reals
  - As an implementation over the floats
- Abstract domain: zonotopes
  - Intuition: convex polytopes with a central symmetry
  + Advantages:
    - Linear correlations between variables are preserved
    - Good trade-off between performance and precision
  - Limits:
    - Weakly relational: better than the intervals, not as good as polyhedra
    - Not very accurate for nonlinear operations
    - Not accurate on very common program constructs such as if

Proposed approach

Overview

We use constraint-based local consistencies to reduce the domains of variables computed by Fluctuat



Proposed approach

Details

- A set of constraint systems is generated for a C program
  - A constraint system is built on-the-fly while exploring a path
  - Exploration of a path stops if the constraint system becomes inconsistent
  - Loops are unfolded a finite number of times
- Local consistencies (filtering)
  - Reals: Hull & Box-consistency (RealPaver)
  - Floats: 3B-consistency (FPCS)
- Reduced domain of a variable: union of the intervals generated for this variable from the filtering of all the consistent systems of the program

Local consistencies

- Constraint Programming relies on filtering
  - Remove inconsistent values from domains
  - Shaving: remove inconsistent values at interval bounds

- Local consistency: consistency of a relaxation of the system
  - Property that holds on a subset of variables or constraints
  - Associated with a filtering algorithm

Local consistencies

- Constraint Programming relies on filtering
  - Remove inconsistent values from domains
  - Shaving: remove inconsistent values at interval bounds

- Local consistency: consistency of a relaxation of the system
  - Property that holds on a subset of variables or constraints
  - Associated with a filtering algorithm

Local consistencies

- Constraint Programming relies on filtering
  - Remove inconsistent values from domains
  - Shaving: remove inconsistent values at interval bounds

- Local consistency: consistency of a relaxation of the system
  - Property that holds on a subset of variables or constraints
  - Associated with a filtering algorithm

Local consistencies

Box-consistency

- Intuition: enforce that bounds of domains are solutions of the interval extension of a given constraint

Consider CSP: $\{x + y - x = 0, x \in [-1, 1], y \in [0, 1]\}$

- Check variable x is box-consistent
  - $([-1, -1^+] \oplus [0, 1] \ominus [-1, -1^+]) \cap [0, 0] \neq \emptyset$
  - $([1^-, 1] \oplus [0, 1] \ominus [1^-, 1]) \cap [0, 0] \neq \emptyset$
- Check variable y is box-consistent
  - $([-1, 1] \oplus [0, 0^+] \ominus [-1, 1]) \cap [0, 0] \neq \emptyset$
  - $([-1, 1] \oplus [1^-, 1] \ominus [-1, 1]) \cap [0, 0] \neq \emptyset$

- Filtering algorithm combines dichotomy and interval Newton method

Experiments

- Programs
  - `quadratic`: computing the roots of a quadratic equation (GSL library) – conditionals
  - `sinus7`: expression of the 7th-order Taylor series of function sinus – nonlinearity
  - `rump`: polynomial of Rump – nonlinearity
  - `sqrt`: square root computation (Babylonian method) – iterative program
- Expected loss of accuracy of Fluctuat
  - Union at the earliest of program states (join operator): `quadratic`, `sqrt`;
  - Domain intersection due to conditional statements (meet operator): `quadratic`, `sqrt`;
  - Interpolation by expanding approximations due to loops (widening operator): `sqrt`;
  - Nonlinear expression approximation: `quadratic`, `sinus7`, `rump`, `sqrt`.

Results over the reals

	Fluctuat (AI)		RealPaver (CP)	
	Domain	Time	Domain	Time
$\text{quadratic}_1\ x_0$	$[-\infty, \infty]$	0.1 s	$[-\infty, 0]$	1.5 s
$\text{quadratic}_1\ x_1$	$[-\infty, \infty]$	0.1 s	$[-8.011, \infty]$	1.5 s
$\text{quadratic}_2\ x_0$	$[-2e6, 0]$	0.1 s	$[-1e6, 0]$	0.5 s
$\text{quadratic}_2\ x_1$	$[-1e6, 0]$	0.1 s	$[-5.186e5, 0]$	0.5 s
sinus7	$[-1.009, 1.009]$	0.1 s	$[-0.842, 0.843]$	0.3 s
rump	$[-1e37, 2e37]$	0.1 s	$[-1e36, 1.7e37]$	1.2 s
sqrt_1	$[2.116, 2.354]$	0.1 s	$[2.121, 2.346]$	0.3 s
sqrt_2	$[2.098, 3.435]$	0.1 s	$[2.232, 3.165]$	0.5 s

Results over the floats

	Fluctuat (AI)		FPCS (CP)	
	Domain	Time	Domain	Time
$\text{quadratic}_1\ x_0$	$[-\infty, \infty]$	0.1 s	$[-\infty, 0]$	0.3 s
$\text{quadratic}_1\ x_1$	$[-\infty, \infty]$	0.1 s	$[-8.064, \infty]$	0.3 s
$\text{quadratic}_2\ x_0$	$[-2e6, 0]$	0.1 s	$[-2e6, 0]$	0.3 s
$\text{quadratic}_2\ x_1$	$[-1e6, 0]$	0.1 s	$[-2503.8, 0]$	0.3 s
sinus7	$[-1.009, 1.009]$	0.1 s	$[-0.853, 0.852]$	0.2 s
rump	$[-1e37, 2e37]$	0.1 s	$[-1e37, 2e37]$	0.2 s
sqrt_1	$[2.116, 2.354]$	0.1 s	$[2.120, 2.347]$	1 s
sqrt_2	$[-\infty, \infty]$	0.1 s	$[2.232, 3.168]$	1.6 s

Conclusion

- Constraint solvers
  - \+ Advantages:
    - Good refutation capabilities
    - Handling of nonlinear expressions
  - \- Limits:
    - Distinct exploration of each executable path is a critical issue
- Abstract Interpretation
  - \+ Advantages:
    - Good scaling capabilities
    - Handling of linear expressions
  - \- Limits:
    - Loss of accuracy may occur

→ Complementary techniques

- Greater domain reduction achieved when combining both approaches
- Automated and tight cooperation is promising

Conclusion

- Constraint solvers
  - \+ Advantages:
    - Good refutation capabilities
    - Handling of nonlinear expressions
  - \- Limits:
    - Distinct exploration of each executable path is a critical issue
- Abstract Interpretation
  - \+ Advantages:
    - Good scaling capabilities
    - Handling of linear expressions
  - \- Limits:
    - Loss of accuracy may occur

→ Complementary techniques

- Greater domain reduction achieved when combining both approaches
- Automated and tight cooperation is promising