Software Verification

Grégoire Sutre

LaBRI, University of Bordeaux, CNRS, France

Summer School on Verification Technology, Systems & Applications

September 2008

Part I

Introduction

Outline — Introduction

Outline — Introduction

Ubiquity of Software in Modern Life

Once upon a time, lecturers used hand-written transparencies with an overhead projector.

- pens
- transparencies
- scissors
- sticky tape
- lamp
- lenses
- mirror
- screen

Nowadays softwares are used to design the slides and to project them

Similar evolution in many, many areas

Ubiquity of Software in Modern Life

Once upon a time, lecturers used hand-written transparencies with an overhead projector.

- pens
- transparencies
- scissors
- sticky tape
- lamp
- lenses
- mirror
- screen

Nowadays softwares are used to design the slides and to project them

Similar evolution in many, many areas

Why?

Some advantages of software over dedicated hardware components

- Reduce time to market
  - Less time to write the slides (really?)
  - Ability to re-organize the presentation
- Reduce costs
  - No pen, no transparencies
  - Re-usability of slides, ability to make minor modifications for free
- Increase functionality
  - Automatic generation of some slides (table of contents)
  - Nicer overlays (sticky tape is not required anymore!)
  - Ability to display videos

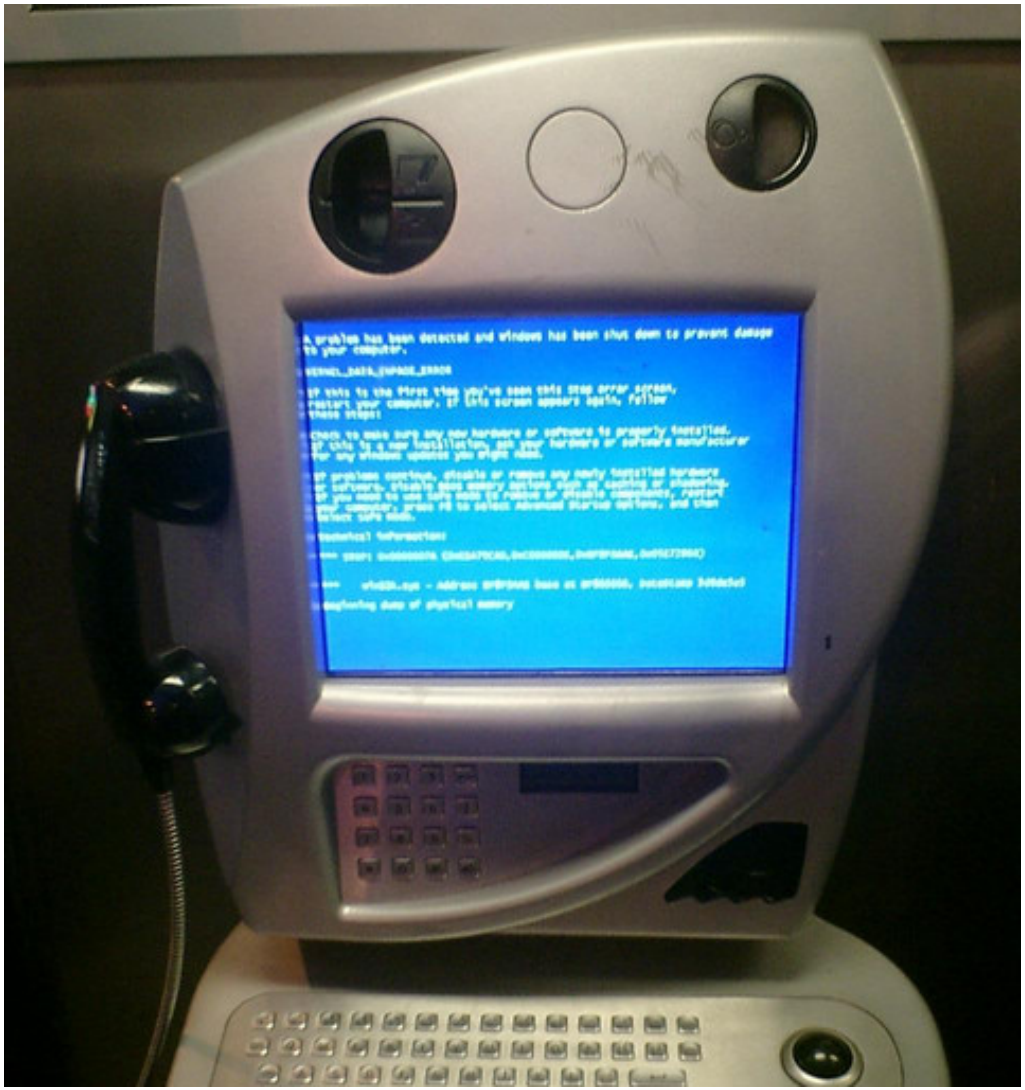
But software is not without risk...

Bugs are Frequent in Software

Bugs are Frequent in Software

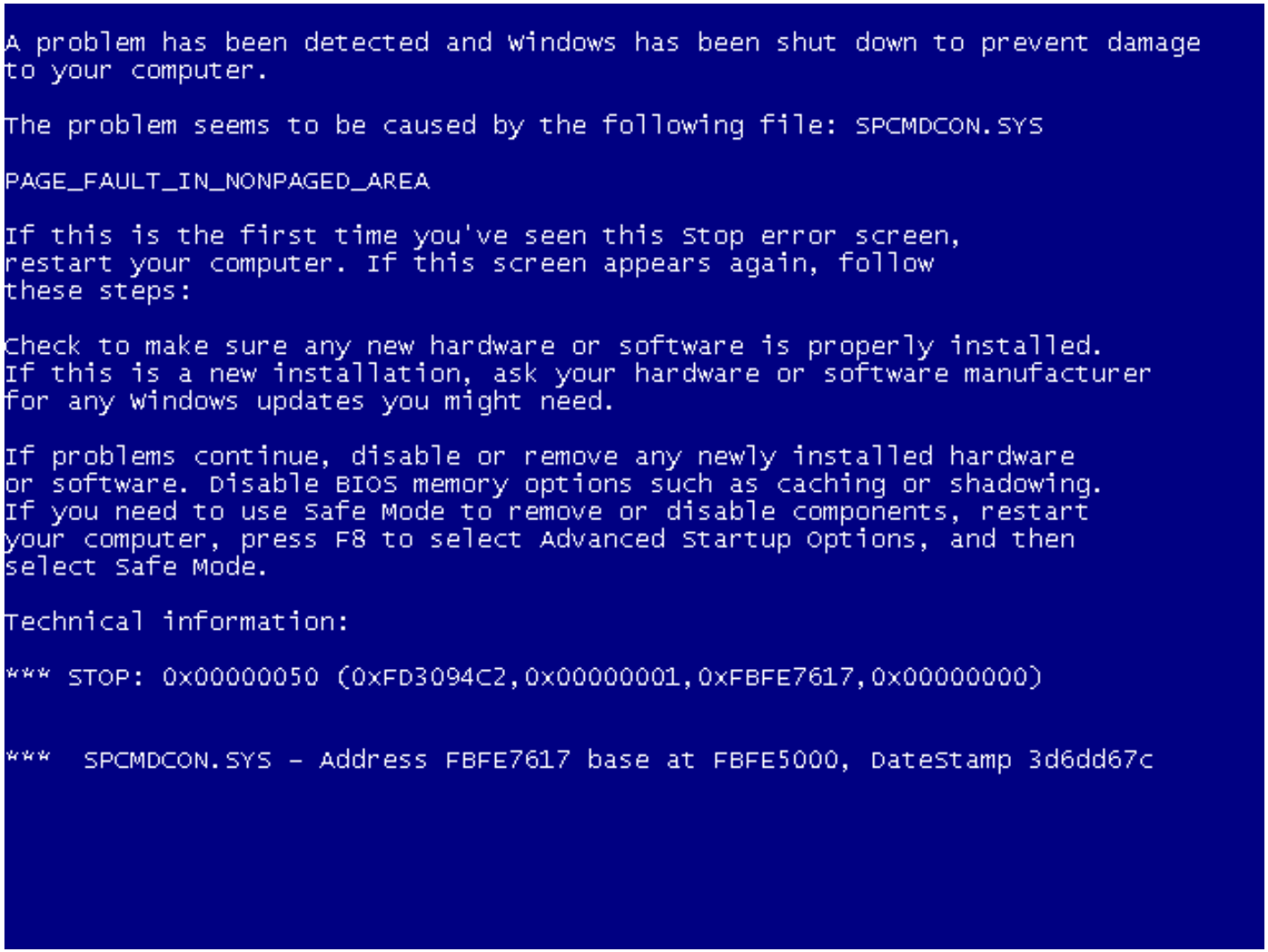
Bugs are Frequent in Software

Bugs are Frequent in Software



Bugs are Frequent in Software

Bugs are Frequent in Software



A Critical Software Bug: Ariane 5.01

« On 4 June 1996, the maiden flight of the Ariane 5 launcher ended in a failure. Only about 40 seconds after initiation of the flight sequence, at an altitude of about 3700 m, the launcher veered off its flight path, broke up and exploded. »

« The failure of the Ariane 5.01 was caused by the complete loss of guidance and attitude information 37 seconds after start of the main engine ignition sequence (30 seconds after lift-off). This loss of information was due to specification and design errors in the software of the inertial reference system. »

A Critical Software Bug: Ariane 5.01

« On 4 June 1996, the maiden flight of the Ariane 5 launcher ended in a failure. Only about 40 seconds after initiation of the flight sequence, at an altitude of about 3700 m, the launcher veered off its flight path, broke up and exploded. »

« The failure of the Ariane 5.01 was caused by the complete loss of guidance and attitude information 37 seconds after start of the main engine ignition sequence (30 seconds after lift-off). This loss of information was due to specification and design errors in the software of the inertial reference system. »

Software in Embedded Systems

Embedded systems in: cell phones, satellites, airplanes, cars, wireless routers, MP3 players, refrigerators, ...

Examples of Critical Systems

- attitude and orbit control systems in satellites
- *X-by-wire* control systems in airplanes and in cars (soon)

Increasing importance of software in embedded systems

- custom hardware replaced by processor + custom software
- software is a dominant factor in design time and cost (70 %)

Critical embedded systems require “exhaustive” validation

Software Complexity Grows Exponentially

As computational power grows . . .

Moore's law: *« the number of transistors on a chip doubles every two years »*

. . . software complexity grows . . .

Wirth's Law: *« software gets slower faster than hardware gets faster »*

. . . and so does the number of bugs!

Watts S. Humphrey: *« 5 – 10 bugs per 1000 lines of code after product test »*

Growing need for automatic validation techniques

Software Complexity Grows Exponentially

As computational power grows . . .

Moore's law: *« the number of transistors on a chip doubles every two years »*

. . . software complexity grows . . .

Wirth's Law: *« software gets slower faster than hardware gets faster »*

. . . and so does the number of bugs!

Watts S. Humphrey: *« 5 – 10 bugs per 1000 lines of code after product test »*

Growing need for automatic validation techniques

Software Complexity Grows Exponentially

As computational power grows . . .

Moore's law: *« the number of transistors on a chip doubles every two years »*

. . . software complexity grows . . .

Wirth's Law: *« software gets slower faster than hardware gets faster »*

. . . and so does the number of bugs!

Watts S. Humphrey: *« 5 – 10 bugs per 1000 lines of code after product test »*

Growing need for automatic validation techniques

Software Complexity Grows Exponentially

As computational power grows . . .

Moore's law: *« the number of transistors on a chip doubles every two years »*

. . . software complexity grows . . .

Wirth's Law: *« software gets slower faster than hardware gets faster »*

. . . and so does the number of bugs!

Watts S. Humphrey: *« 5 – 10 bugs per 1000 lines of code after product test »*

Growing need for automatic validation techniques

Outline — Introduction

Software Testing

Running the executable (obtained by compilation)

- on multiple inputs
- usually on the target platform

Testing is a widespread validation approach in the software industry

- can be (partially) automated
- can detect a lot of bugs

But

Costly and time-consuming

Not exhaustive

Software Testing

Running the executable (obtained by compilation)

- on multiple inputs
- usually on the target platform

Testing is a widespread validation approach in the software industry

- can be (partially) automated
- can detect a lot of bugs

But

Costly and time-consuming

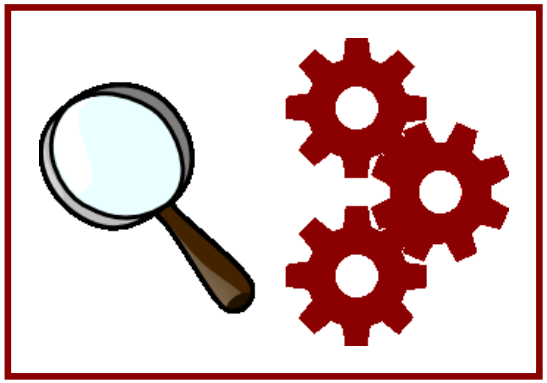
Not exhaustive

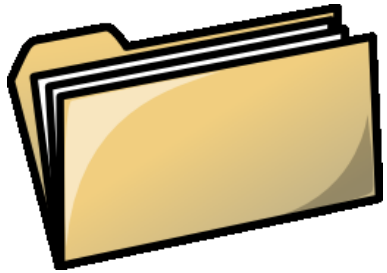
Dream of Software Model-Checking

```
x = 1;
if (y <= 10) {
   y = 10;
}
else {
   while (x < y) {
      x = 2 * x;
      y = y - 1;
   }
}
x = y + 1;
```

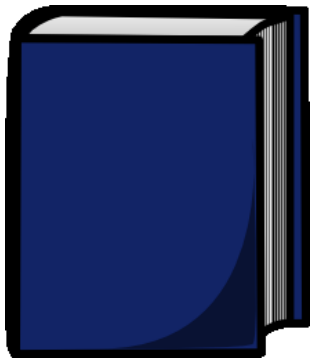
Program

Model Checker





Results



Requirements

Fundamental Limit: Undecidability

Rice's Theorem

Any non-trivial semantic property of programs is undecidable.

Classical Example: Termination

There exists no algorithm which can solve the halting problem:

- given a description of a program as input,
- decide whether the program terminates or loops forever.

Practical Limit: Combinatorial Explosion

Implicit in Rice's Theorem is an idealized program model, where programs have access to *unbounded memory*.

In reality programs are run on a computer with *bounded memory*.

Model-checking becomes decidable for finite-state systems.

But even with bounded memory, complexity in practice is too high for finite-state model-checking:

- 1 megabyte (1 000 000 bytes) of memory $\approx 10^{2\,400\,000}$ states
- 1000 variables $\times$ 64 bits $\approx 10^{19\,200}$ states
- optimistic limit for finite-state model checkers: 10^{100} states

More Realistic Objectives for Software Verification

Incomplete Methods

Approximate Algorithms

- ☺ Always terminate
- ☹ Indefinite answer (yes / no / ?)

Exact Semi-Algorithms

- ☺ Definite answer (yes / no)
- ☹ May not terminate

Topics of the lecture

Static Analysis

Abstraction Refinement

More Realistic Objectives for Software Verification

Incomplete Methods

Approximate Algorithms

- ☺ Always terminate
- ☹ Indefinite answer (yes / no / ?)

Exact Semi-Algorithms

- ☺ Definite answer (yes / no)
- ☹ May not terminate

Topics of the lecture

Static Analysis

Abstraction Refinement

Static Analysis

Tentative Definition

Compile-time techniques to gather run-time information about programs without actually running them

Example

Detection of variables that are used before initialization

- ☺ Always terminates
- ☺ Applies to large programs
- ☹ Simple analyses (original goal was compilation)
- ☹ Indefinite answer (yes / no / ?)

In the Lecture

Data Flow Analysis Abstract Interpretation

Static Analysis

Tentative Definition

Compile-time techniques to gather run-time information about programs without actually running them

Example

Detection of variables that are used before initialization

- ☺ Always terminates
- ☺ Applies to large programs
- ☹ Simple analyses (original goal was compilation)
- ☹ Indefinite answer (yes / no / ?)

In the Lecture

Data Flow Analysis Abstract Interpretation

Abstraction Refinement

Tentative Definition

Analysis-time techniques to verify programs by model-checking and refinement of finite-state approximate models

Example

Verification of safety and fairness of a mutual exclusion algorithm

- ☺ Complex analyses (properties expressed in temporal logics)
- ☺ Definite answer (yes / no)
- ☹ May not terminate
- ☹ Modeling of the program into a finite-state transition system

In the Lecture

Abstract Model Refinement for Safety Properties

Abstraction Refinement

Tentative Definition

Analysis-time techniques to verify programs by model-checking and refinement of finite-state approximate models

Example

Verification of safety and fairness of a mutual exclusion algorithm

- ☺ Complex analyses (properties expressed in temporal logics)
- ☺ Definite answer (yes / no)
- ☹ May not terminate
- ☹ Modeling of the program into a finite-state transition system

In the Lecture

Abstract Model Refinement for Safety Properties

Common Ingredient: Property-Preserving Abstraction

Abstraction Process

Interpret programs according to a simplified, “abstract” semantics.

Property-Preserving Abstraction

Formally relate the “abstract” semantics with the “standard” semantics, so as to preserve relevant properties.

Preservation of Properties

Program interpretation with this abstract semantics therefore gives “correct” information about properties of real runs.

Abstract Interpretation Example: Sign Analysis

Objective of Sign Analysis

Discover for each program point the sign of possible run-time values that numerical variables can have at that point.

The abstract semantics "tracks" the following information, for each variable x:

$$\begin{array}{lll} x & < & 0 \\ x & \leq & 0 \\ x & = & 0 \\ x & \geq & 0 \\ x & > & 0 \end{array}$$

Abstract Interpretation Example: Sign Analysis

```
1  x = 1;
2  if (y ≤ 10) {
3     y = 10;
4  }
5  else {
6     while (x < y) {
7        x = 2 * x;
8        y = y - 1;
9     }
10 }
11 x = y + 1;
12 assert(x > 0);
```

Abstract Interpretation Example: Sign Analysis

```
1  x = 1;
2  if (y ≤ 10) {
3     y = 10;
4  }
5  else {
6     while (x < y) {
7        x = 2 * x;
8        y = y - 1;
9     }
10 }
11 x = y + 1;
12 assert(x > 0);
```

$x > 0$

Abstract Interpretation Example: Sign Analysis

```
1  x = 1;
2  if (y ≤ 10) {
3     y = 10;
4  }
5  else {
6     while (x < y) {
7        x = 2 * x;
8        y = y - 1;
9     }
10 }
11 x = y + 1;
12 assert(x > 0);
```

$x > 0$

$x > 0$

Abstract Interpretation Example: Sign Analysis

```
1  x = 1;
2  if (y ≤ 10) {
3     y = 10;
4  }
5  else {
6     while (x < y) {
7        x = 2 * x;
8        y = y - 1;
9     }
10 }
11 x = y + 1;
12 assert(x > 0);
```

$x > 0$

$x > 0$

$x > 0 \wedge y > 0$

Abstract Interpretation Example: Sign Analysis

```
1  x = 1;
2  if (y ≤ 10) {
3     y = 10;
4  }
5  else {
6     while (x < y) {
7        x = 2 * x;
8        y = y - 1;
9     }
10 }
11 x = y + 1;
12 assert(x > 0);
```

$x > 0$

$x > 0$

$x > 0 \wedge y > 0$

$x > 0 \wedge y > 0$

Abstract Interpretation Example: Sign Analysis

```
1  x = 1;
2  if (y ≤ 10) {
3     y = 10;
4  }
5  else {
6     while (x < y) {
7        x = 2 * x;
8        y = y - 1;
9     }
10 }
11 x = y + 1;
12 assert(x > 0);
```

$x > 0$

$x > 0$

$x > 0 \wedge y > 0$

$x > 0 \wedge y > 0$

$x > 0 \wedge y > 0$

Abstract Interpretation Example: Sign Analysis

```
1  x = 1;                          x > 0
2  if (y ≤ 10) {                   x > 0
3     y = 10;                      x > 0 ∧ y > 0
4  }
5  else {                          x > 0 ∧ y > 0
6     while (x < y) {              x > 0 ∧ y > 0
7        x = 2 * x;                x > 0 ∧ y > 0
8        y = y - 1;
9     }
10 }
11 x = y + 1;
12 assert(x > 0);
```

Abstract Interpretation Example: Sign Analysis

```
1  x = 1;
2  if (y ≤ 10) {
3     y = 10;
4  }
5  else {
6     while (x < y) {
7        x = 2 * x;
8        y = y - 1;
9     }
10 }
11 x = y + 1;
12 assert(x > 0);
```

$x > 0$

$x > 0$

$x > 0 \wedge y > 0$

$x > 0 \wedge y > 0$

$x > 0 \wedge y > 0$

$x > 0 \wedge y > 0$

$x > 0 \wedge y \geq 0$

Abstract Interpretation Example: Sign Analysis

```
1  x = 1;
2  if (y ≤ 10) {
3     y = 10;
4  }
5  else {
6     while (x < y) {
7        x = 2 * x;
8        y = y - 1;
9     }
10 }
11 x = y + 1;
12 assert(x > 0);
```

$x > 0$

$x > 0$

$x > 0 \wedge y > 0$

$x > 0 \wedge y > 0$

$x > 0 \wedge y > 0$ $\bigvee$ $x > 0 \wedge y \geq 0 \wedge x < y$

$x > 0 \wedge y > 0$

$x > 0 \wedge y \geq 0$

Abstract Interpretation Example: Sign Analysis

```
1  x = 1;
2  if (y ≤ 10) {
3     y = 10;
4  }
5  else {
6     while (x < y) {
7        x = 2 * x;
8        y = y - 1;
9     }
10 }
11 x = y + 1;
12 assert(x > 0);
```

$x > 0$

$x > 0$

$x > 0 \wedge y > 0$

$x > 0 \wedge y > 0$

$x > 0 \wedge y > 0$ $\bigvee$ $x > 0 \wedge y \geq 0 \wedge x < y$

$x > 0 \wedge y > 0$

$x > 0 \wedge y \geq 0$

$x > 0 \wedge y \geq 0$

Abstract Interpretation Example: Sign Analysis

```
1  x = 1;
2  if (y ≤ 10) {
3     y = 10;
4  }
5  else {
6     while (x < y) {
7        x = 2 * x;
8        y = y - 1;
9     }
10 }
11 x = y + 1;
12 assert(x > 0);
```

$x > 0$

$x > 0$

$x > 0 \wedge y > 0$

$x > 0 \wedge y > 0$

$x > 0 \wedge y > 0 \quad \bigvee \quad x > 0 \wedge y \geq 0 \wedge x < y$

$x > 0 \wedge y > 0$

$x > 0 \wedge y \geq 0$

$x > 0 \wedge y \geq 0$

$x > 0 \wedge y \geq 0 \quad (x > 0 \wedge y > 0) \bigvee (x > 0 \wedge y \geq 0)$

Abstract Interpretation Example: Sign Analysis

```
1  x = 1;
2  if (y ≤ 10) {
3     y = 10;
4  }
5  else {
6     while (x < y) {
7        x = 2 * x;
8        y = y - 1;
9     }
10 }
11 x = y + 1;
12 assert(x > 0);
```

$x > 0$

$x > 0$

$x > 0 \wedge y > 0$

$x > 0 \wedge y > 0$

$x > 0 \wedge y > 0 \quad \bigvee \quad x > 0 \wedge y \geq 0 \wedge x < y$

$x > 0 \wedge y > 0$

$x > 0 \wedge y \geq 0$

$x > 0 \wedge y \geq 0$

$x > 0 \wedge y \geq 0 \quad (x > 0 \wedge y > 0) \bigvee (x > 0 \wedge y \geq 0)$

$x > 0 \wedge y \geq 0$

Abstract Interpretation Example: Sign Analysis

```
1  x = 1;
2  if (y ≤ 10) {
3     y = 10;
4  }
5  else {
6     while (x < y) {
7        x = 2 * x;
8        y = y - 1;
9     }
10 }
11 x = y + 1;
12 assert(x > 0);
```

$x > 0$

$x > 0$

$x > 0 \wedge y > 0$

$x > 0 \wedge y > 0$

$x > 0 \wedge y > 0$ $\bigvee$ $x > 0 \wedge y \geq 0 \wedge x < y$

$x > 0 \wedge y > 0$

$x > 0 \wedge y \geq 0$

$x > 0 \wedge y \geq 0$

$x > 0 \wedge y \geq 0$ $(x > 0 \wedge y > 0) \bigvee (x > 0 \wedge y \geq 0)$

$x > 0 \wedge y \geq 0$

Credits: Pioneers (1970's)

Iterative Data Flow Analysis

Gary Kildall

John Kam & Jeffrey Ullman

Michael Karr

...

Abstract Interpretation

Patrick Cousot & Radhia Cousot

Nicolas Halbwachs

...

And many, many more...

Apologies!

Outline of the Lecture

- Control Flow Automata
- Data Flow Analysis
- Abstract Interpretation
- Abstract Model Refinement

Outline of the Lecture

- Control Flow Automata
- Data Flow Analysis
- Abstract Interpretation
- Abstract Model Refinement

Static Analysis

Outline of the Lecture

- Control Flow Automata
- Data Flow Analysis
- Abstract Interpretation
- Abstract Model Refinement

Static Analysis

Abstraction Refinement

Part II

Control Flow Automata

Outline — Control Flow Automata

Outline — Control Flow Automata

Short Introduction to Control Flow Automata

Requirement for verification: formal semantics of programs

Formal Semantics

Formalization as a mathematical model of the meaning of programs

Denotational | Operational | Axiomatic

Operational Semantics

Labeled transition system describing the possible computational steps

First Step Towards an Operational Semantics

Program text $\longrightarrow$ Graph-based representation

Short Introduction to Control Flow Automata

Requirement for verification: formal semantics of programs

Formal Semantics

Formalization as a mathematical model of the meaning of programs

Denotational | Operational | Axiomatic

Operational Semantics

Labeled transition system describing the possible computational steps

First Step Towards an Operational Semantics

Program text $\longrightarrow$ Graph-based representation

Short Introduction to Control Flow Automata

Requirement for verification: formal semantics of programs

Formal Semantics

Formalization as a mathematical model of the meaning of programs

Denotational Operational Axiomatic

Operational Semantics

Labeled transition system describing the possible computational steps

First Step Towards an Operational Semantics

Program text $\longrightarrow$ Graph-based representation

Short Introduction to Control Flow Automata

Requirement for verification: formal semantics of programs

Formal Semantics

Formalization as a mathematical model of the meaning of programs

Denotational | Operational | Axiomatic

Operational Semantics

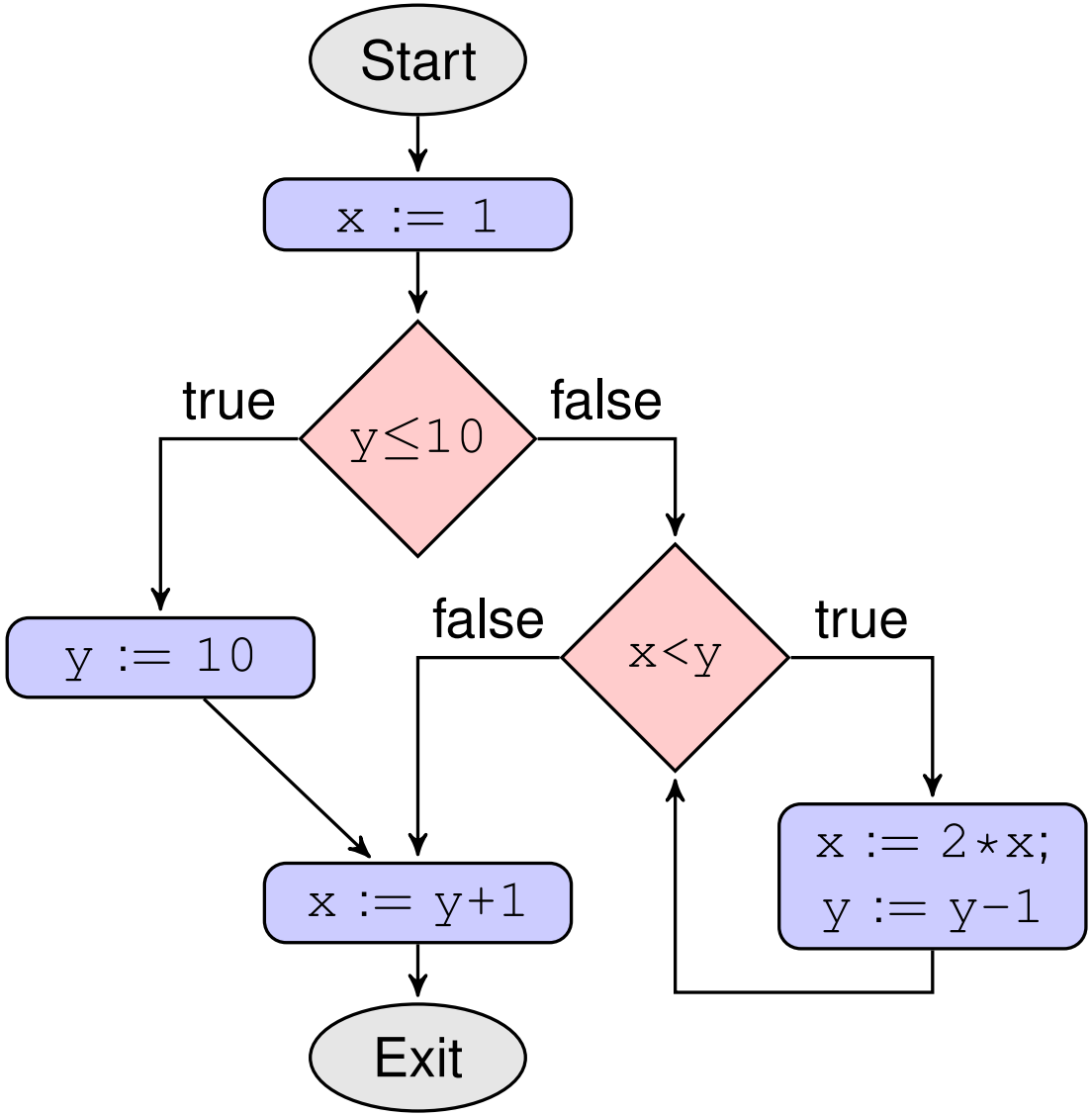
Labeled transition system describing the possible computational steps

First Step Towards an Operational Semantics

Program text $\longrightarrow$ Graph-based representation
Control flow automaton

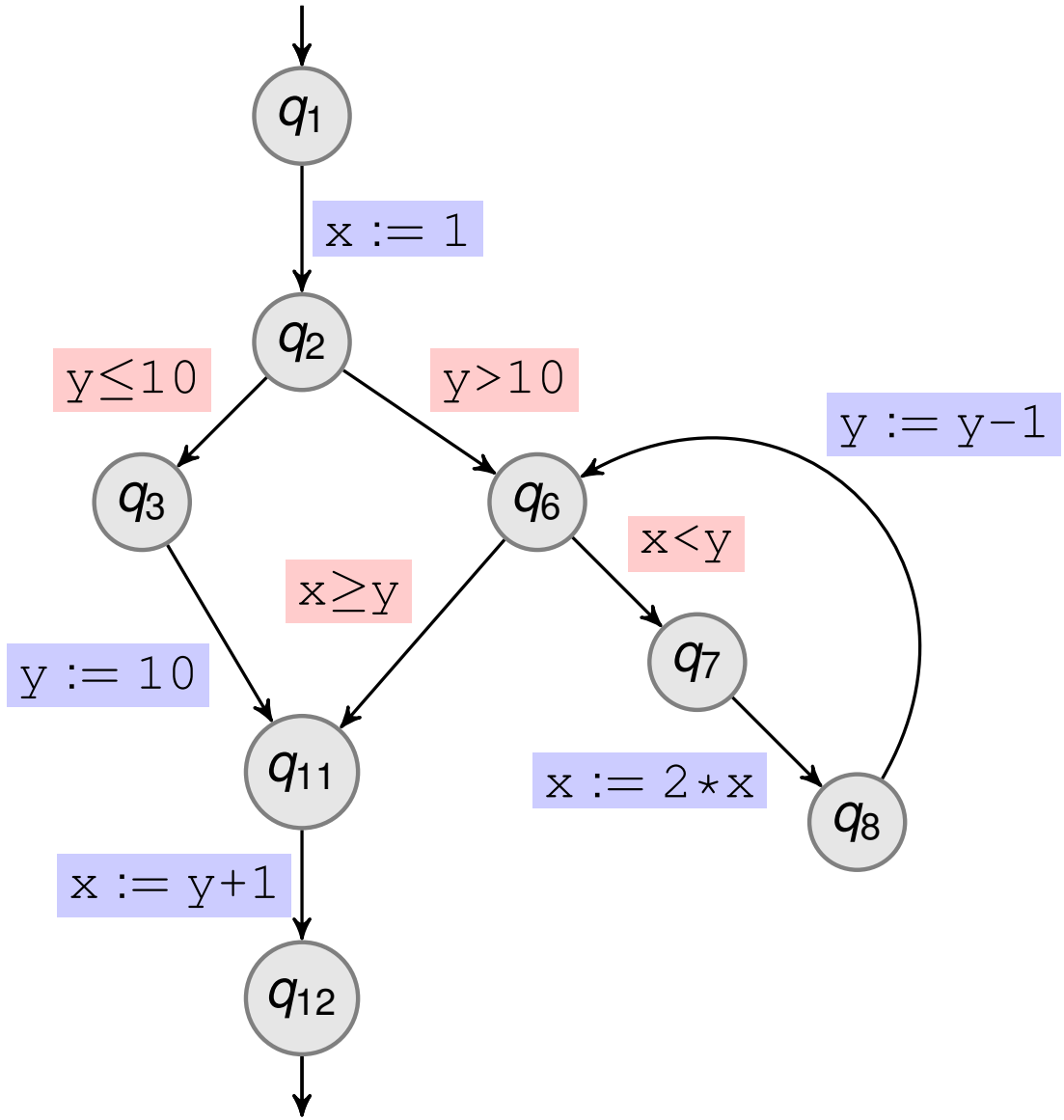
Control Flow Graph

```
1  x = 1;
2  if (y ≤ 10) {
3     y = 10;
4  }
5  else {
6     while (x < y) {
7        x = 2 * x;
8        y = y - 1;
9     }
10 }
11 x = y + 1;
12
```



Control Flow Automaton

```
1  x = 1;
2  if (y ≤ 10) {
3     y = 10;
4  }
5  else {
6     while (x < y) {
7        x = 2 * x;
8        y = y - 1;
9     }
10 }
11 x = y + 1;
12
```



Labeled Directed Graphs

Definition

A labeled directed graph is a triple $G = \langle V, \Sigma, \rightarrow \rangle$ where:

- V is a finite set of *vertices*,
- Σ is a finite set of *labels*,
- $\rightarrow \subseteq V \times \Sigma \times V$ is a finite set of *edges*.

Notation for edges: $v \xrightarrow{\sigma} v'$ instead of $(v, \sigma, v') \in \rightarrow$

A path in G is a finite sequence $v_0 \xrightarrow{\sigma_0} v'_0, \ldots, v_k \xrightarrow{\sigma_k} v'_k$ of edges such that $v'_i = v_{i+1}$ for each $0 \leq i < k$.

Notation for paths: $v_0 \xrightarrow{\sigma_0} v_1 \cdots v_k \xrightarrow{\sigma_k} v'_k$

Labeled Directed Graphs

Definition

A labeled directed graph is a triple $G = \langle V, \Sigma, \rightarrow \rangle$ where:

- V is a finite set of *vertices*,
- Σ is a finite set of *labels*,
- $\rightarrow \subseteq V \times \Sigma \times V$ is a finite set of *edges*.

Notation for edges: $v \xrightarrow{\sigma} v'$ instead of $(v, \sigma, v') \in \rightarrow$

A path in G is a finite sequence $v_0 \xrightarrow{\sigma_0} v'_0, \ldots, v_k \xrightarrow{\sigma_k} v'_k$ of edges such that $v'_i = v_{i+1}$ for each $0 \leq i < k$.

Notation for paths: $v_0 \xrightarrow{\sigma_0} v_1 \cdots v_k \xrightarrow{\sigma_k} v'_k$

Labeled Directed Graphs

Definition

A labeled directed graph is a triple $G = \langle V, \Sigma, \rightarrow \rangle$ where:

- V is a finite set of *vertices*,
- Σ is a finite set of *labels*,
- $\rightarrow \subseteq V \times \Sigma \times V$ is a finite set of *edges*.

Notation for edges: $v \xrightarrow{\sigma} v'$ instead of $(v, \sigma, v') \in \rightarrow$

A path in G is a finite sequence $v_0 \xrightarrow{\sigma_0} v'_0, \ldots, v_k \xrightarrow{\sigma_k} v'_k$ of edges such that $v'_i = v_{i+1}$ for each $0 \leq i < k$.

Notation for paths: $v_0 \xrightarrow{\sigma_0} v_1 \cdots v_k \xrightarrow{\sigma_k} v'_k$

Control Flow Automata: Syntax

Definition

A control flow automaton is a quintuple $\langle Q, q_{in}, q_{out}, \mathtt{X}, \rightarrow \rangle$ where:

- Q is a finite set of *locations*,
- $q_{in} \in Q$ is an *initial location* and $q_{out} \in Q$ is an *exit location*,
- $\mathtt{X}$ is a finite set of *variables*,
- $\rightarrow \subseteq Q \times \mathtt{Op} \times Q$ is a finite set of *transitions*.

$\mathtt{Op}$ is the set of operations defined by:

$$
\begin{array}{rcl}
\mathit{cst} & ::= & c \in \mathbb{Q} \\
\mathit{var} & ::= & x \in \mathtt{X} \\
\mathit{expr} & ::= & \mathit{cst} \mid \mathit{var} \mid \mathit{expr} \bullet \mathit{expr}, \text{ with } \bullet \in \{+, -, *\} \\
\mathit{guard} & ::= & \mathit{expr} \blacktriangleleft \mathit{expr}, \text{ with } \blacktriangleleft \in \{<, \leq, =, \neq, \geq, >\} \\
\mathtt{Op} & ::= & \mathit{guard} \mid \mathit{var} := \mathit{expr}
\end{array}
$$

Control Flow Automata: Syntax

Definition

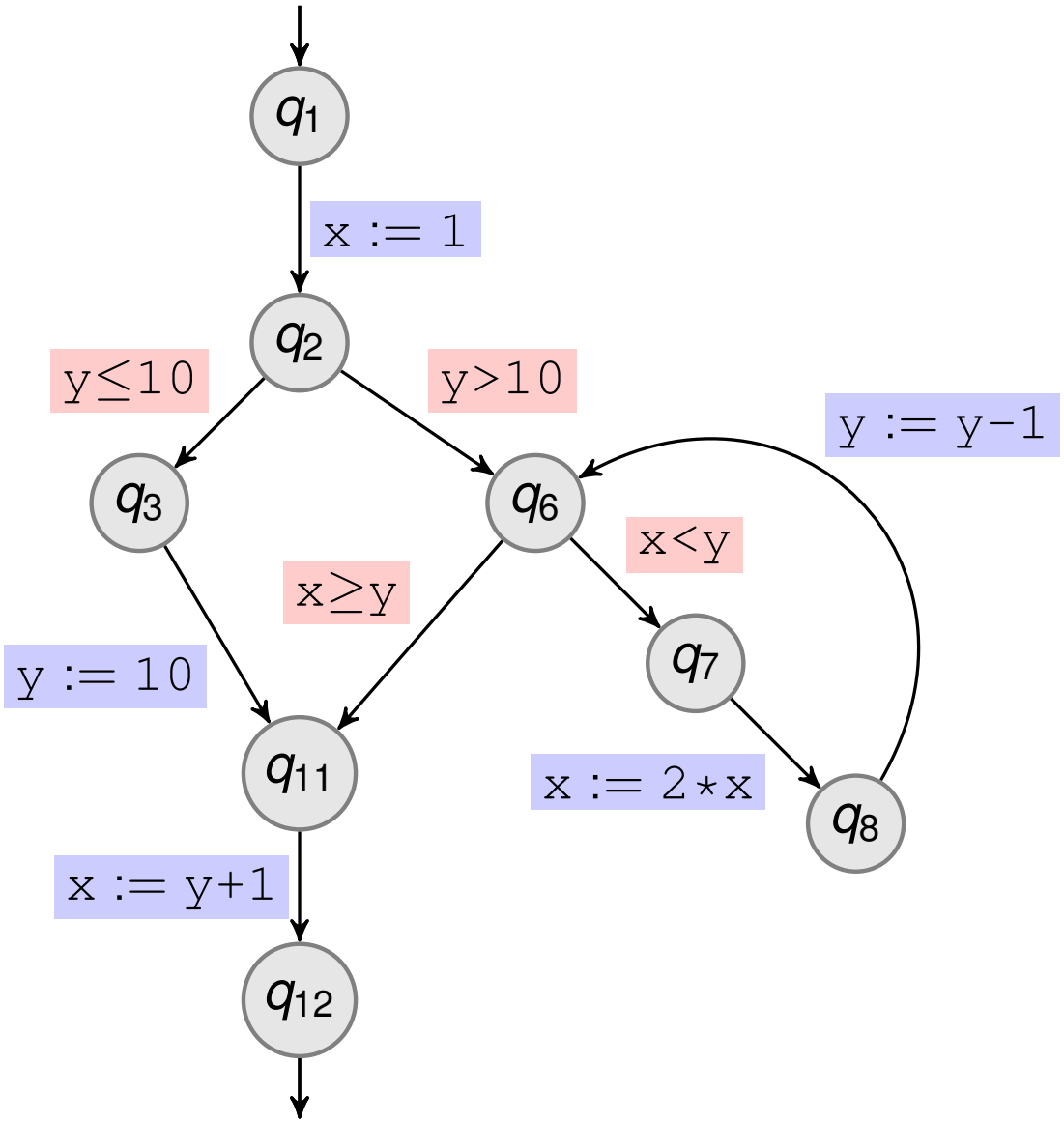
A control flow automaton is a quintuple $\langle Q, q_{in}, q_{out}, \mathtt{X}, \rightarrow \rangle$ where:

- Q is a finite set of *locations*,
- $q_{in} \in Q$ is an *initial location* and $q_{out} \in Q$ is an *exit location*,
- $\mathtt{X}$ is a finite set of *variables*,
- $\rightarrow \subseteq Q \times \mathtt{Op} \times Q$ is a finite set of *transitions*.

$\mathtt{Op}$ is the set of operations defined by:

$$
\begin{array}{rcl}
\mathit{cst} & ::= & c \in \mathbb{Q} \\
\mathit{var} & ::= & x \in \mathtt{X} \\
\mathit{expr} & ::= & \mathit{cst} \mid \mathit{var} \mid \mathit{expr} \bullet \mathit{expr}, \text{ with } \bullet \in \{+, -, *\} \\
\mathit{guard} & ::= & \mathit{expr} \blacktriangleleft \mathit{expr}, \text{ with } \blacktriangleleft \in \{<, \leq, =, \neq, \geq, >\} \\
\mathtt{Op} & ::= & \mathit{guard} \mid \mathit{var} := \mathit{expr}
\end{array}
$$

Control Flow Automata: Syntax



$$Q = \left\{ \begin{matrix} q_1, q_2, q_3, q_6, \\ q_7, q_8, q_{11}, q_{12} \end{matrix} \right\}$$

$$q_{in} = q_1$$

$$q_{out} = q_{12}$$

$$\mathtt{X} = \{\mathtt{x}, \mathtt{y}\}$$

$$\rightarrow = \left\{ \begin{matrix} (q_1, \mathtt{x := 1}, q_2), \\ (q_2, \mathtt{y \leq 10}, q_3), \\ (q_2, \mathtt{y > 10}, q_6), \\ (q_3, \mathtt{y := 10}, q_{11}), \\ \ldots \end{matrix} \right\}$$

Programs as Control Flow Automata

Control flow automata can model:

- ☺ flow of control (program points),
- ☺ numerical variables and numerical operations,
- ☺ non-determinism (uninitialized variables, boolean inputs).

Control flow automata cannot model:

- ☹ pointers
- ☹ recursion
- ☹ threads
- ☹ . . .

But they are complex enough for verification. . . . . . and for learning!

Programs as Control Flow Automata

Control flow automata can model:

- ☺ flow of control (program points),
- ☺ numerical variables and numerical operations,
- ☺ non-determinism (uninitialized variables, boolean inputs).

Control flow automata cannot model:

- ☹ pointers
- ☹ recursion
- ☹ threads
- ☹ . . .

But they are complex enough for verification. . . . . . and for learning!

Programs as Control Flow Automata

Control flow automata can model:

- ☺ flow of control (program points),
- ☺ numerical variables and numerical operations,
- ☺ non-determinism (uninitialized variables, boolean inputs).

Control flow automata cannot model:

- ☹ pointers
- ☹ recursion
- ☹ threads
- ☹ ...

Forget about these...

But they are complex enough for verification... ... and for learning!

Verification of Safety Properties

Goal

Check that “nothing bad can happen”.

Bad behaviors specified e.g. as assertion violations in the original program

An assertion violation can be modeled as a location:

```
assert(x > 0)   ⟹   if (x > 0) then { BAD: }
```

Goal (refined)

Check that there is no “run” that visits a location q contained in a given set $Q_{BAD} \subseteq Q$ of bad locations.

Verification of Safety Properties

Goal

Check that “nothing bad can happen”.

Bad behaviors specified e.g. as assertion violations in the original program

An assertion violation can be modeled as a location:

```
assert(x > 0)   ⟹   if (x > 0) then { BAD: }
```

Goal (refined)

Check that there is no “run” that visits a location q contained in a given set $Q_{BAD} \subseteq Q$ of bad locations.

Verification of Safety Properties

Goal

Check that “nothing bad can happen”.

Bad behaviors specified e.g. as assertion violations in the original program

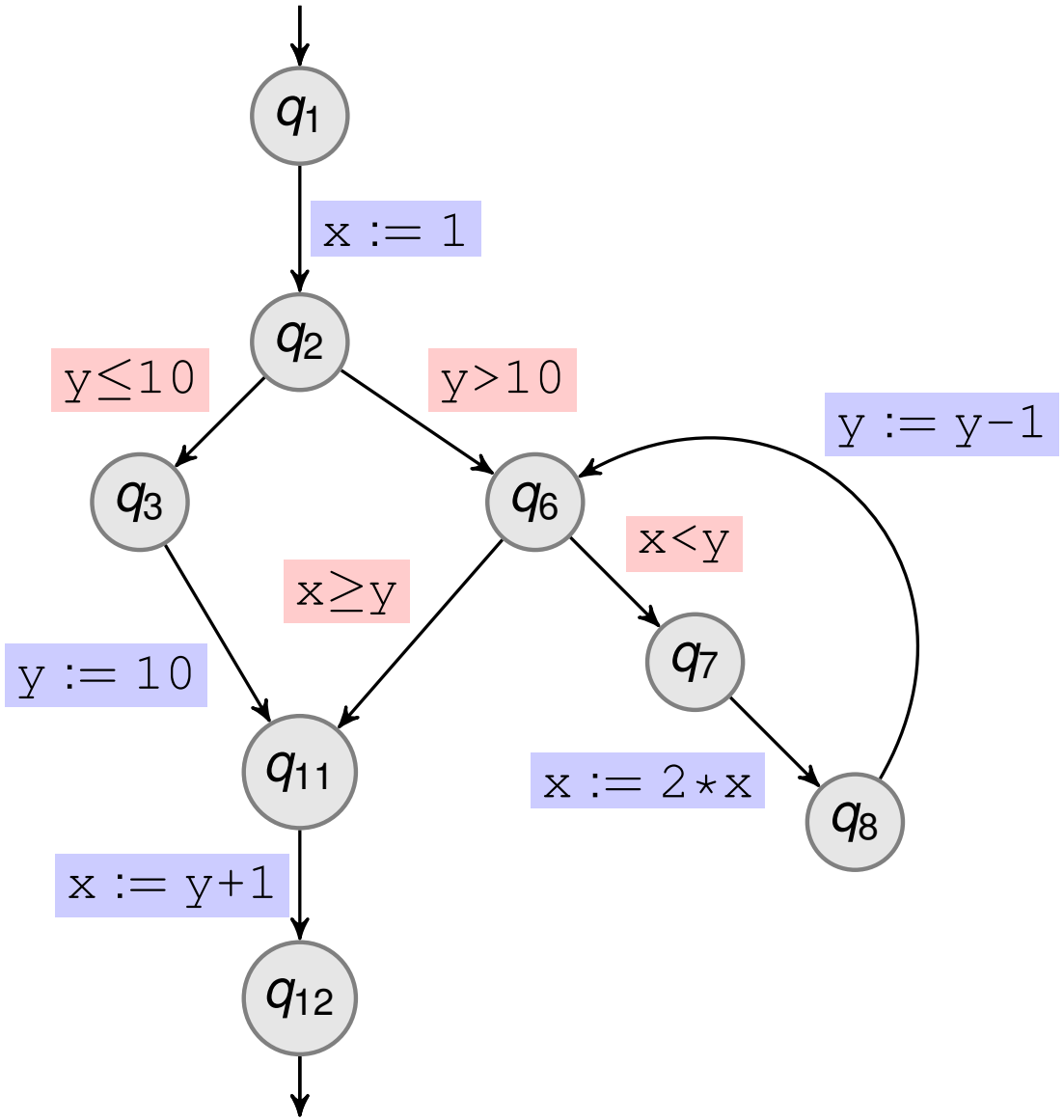
An assertion violation can be modeled as a location:

```
assert(x > 0)   ⟹   if (x > 0) then { BAD: }
```

Goal (refined)

Check that there is no “run” that visits a location q contained in a given set $Q_{BAD} \subseteq Q$ of bad locations.

Runs: Examples



$(q_1, 0, 0)$

$\downarrow$ x := 1

$(q_2, 1, 0)$

$\downarrow$ y≤10

$(q_3, 1, 0)$

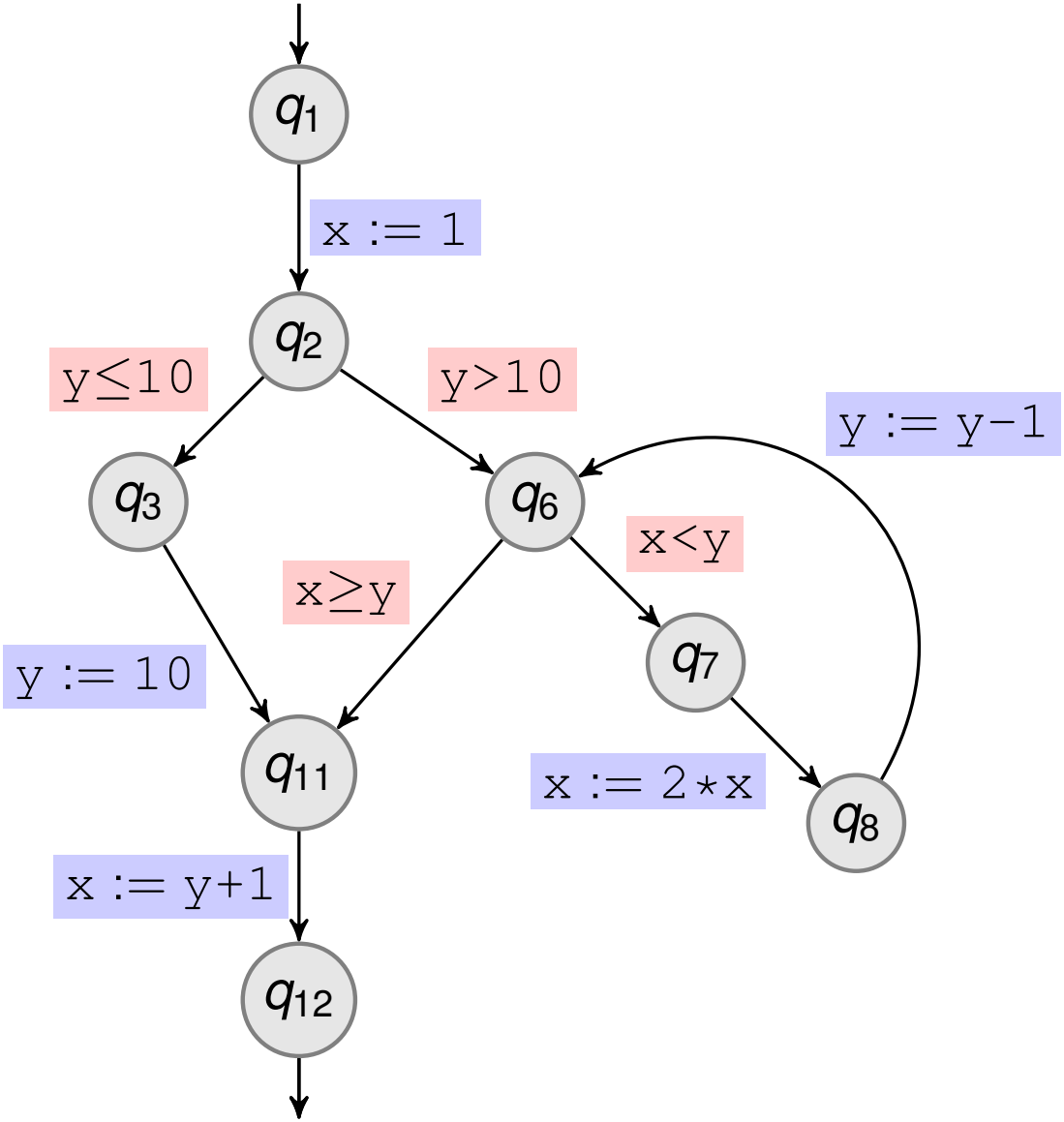
$\downarrow$ y := 10

$(q_{11}, 1, 10)$

$\downarrow$ x := y+1

$(q_{12}, 11, 10)$

Runs: Examples



$(q_1, -159, 27)$

$\downarrow$ x := 1

$(q_2, 1, 27)$

$\downarrow$ y>10

$(q_6, 1, 27)$

$\downarrow$ x<y

$(q_7, 1, 27)$

$\downarrow$ x := 2∗x

$(q_8, 2, 27)$

$\downarrow$ y := y−1

$(q_6, 2, 26)$

$\vdots$

Labeled Transition Systems

Definition

A labeled transition system is a quintuple $\langle C, \mathit{Init}, \mathit{Out}, \Sigma, \rightarrow \rangle$ where :

- C is a set of *configurations*
- $\mathit{Init} \subseteq C$ and $\mathit{Out} \subseteq C$ are sets of *initial* and *exit configurations*
- Σ is a finite set of *actions*
- $\rightarrow \subseteq C \times \Sigma \times C$ is a set of *transitions*

$$\mathrm{Post}(c, \sigma) = \left\{ c' \in C \;\middle|\; c \xrightarrow{\sigma} c' \right\} \qquad \mathrm{Post}(c) = \bigcup_{\sigma \in \Sigma} \mathrm{Post}(c, \sigma)$$

$$\mathrm{Post}(U, \sigma) = \bigcup_{c \in U} \mathrm{Post}(c, \sigma) \qquad \mathrm{Post}(U) = \bigcup_{c \in U} \mathrm{Post}(c)$$

Labeled Transition Systems

Definition

A labeled transition system is a quintuple $\langle C, \mathit{Init}, \mathit{Out}, \Sigma, \rightarrow \rangle$ where :

- C is a set of *configurations*
- $\mathit{Init} \subseteq C$ and $\mathit{Out} \subseteq C$ are sets of *initial* and *exit configurations*
- Σ is a finite set of *actions*
- $\rightarrow \subseteq C \times \Sigma \times C$ is a set of *transitions*

$$\mathrm{Post}(c, \sigma) = \left\{ c' \in C \;\middle|\; c \xrightarrow{\sigma} c' \right\} \qquad \mathrm{Post}(c) = \bigcup_{\sigma \in \Sigma} \mathrm{Post}(c, \sigma)$$

$$\mathrm{Post}(U, \sigma) = \bigcup_{c \in U} \mathrm{Post}(c, \sigma) \qquad \mathrm{Post}(U) = \bigcup_{c \in U} \mathrm{Post}(c)$$

Labeled Transition Systems

Definition

A labeled transition system is a quintuple $\langle C, \mathit{Init}, \mathit{Out}, \Sigma, \rightarrow \rangle$ where :

- C is a set of *configurations*
- $\mathit{Init} \subseteq C$ and $\mathit{Out} \subseteq C$ are sets of *initial* and *exit configurations*
- Σ is a finite set of *actions*
- $\rightarrow \subseteq C \times \Sigma \times C$ is a set of *transitions*

$$\mathrm{Pre}(c, \sigma) = \left\{ c' \in C \;\middle|\; c' \xrightarrow{\sigma} c \right\} \qquad \mathrm{Pre}(c) = \bigcup_{\sigma \in \Sigma} \mathrm{Pre}(c, \sigma)$$

$$\mathrm{Pre}(U, \sigma) = \bigcup_{c \in U} \mathrm{Pre}(c, \sigma) \qquad \mathrm{Pre}(U) = \bigcup_{c \in U} \mathrm{Pre}(c)$$

Semantics of Expressions and Guards

Consider a finite set $\mathtt{X}$ of variables. A valuation is a function $v : \mathtt{X} \to \mathbb{R}$.

Expressions: $[\![e]\!]_v$

$$
\begin{aligned}
[\![c]\!]_v &= c && [c \in \mathbb{Q}] \\
[\![x]\!]_v &= v(x) && [x \in \mathtt{X}] \\
[\![e_1 + e_2]\!]_v &= [\![e_1]\!]_v + [\![e_2]\!]_v \\
[\![e_1 - e_2]\!]_v &= [\![e_1]\!]_v - [\![e_2]\!]_v \\
[\![e_1 * e_2]\!]_v &= [\![e_1]\!]_v \times [\![e_2]\!]_v
\end{aligned}
$$

Guards: $v \models g$

$$
\begin{aligned}
v &\models e_1 < e_2 && \text{if} && [\![e_1]\!]_v < [\![e_2]\!]_v \\
v &\models e_1 \leq e_2 && \text{if} && [\![e_1]\!]_v \leq [\![e_2]\!]_v \\
v &\models e_1 = e_2 && \text{if} && [\![e_1]\!]_v = [\![e_2]\!]_v \\
v &\models e_1 \neq e_2 && \text{if} && [\![e_1]\!]_v \neq [\![e_2]\!]_v \\
v &\models e_1 \geq e_2 && \text{if} && [\![e_1]\!]_v \geq [\![e_2]\!]_v \\
v &\models e_1 > e_2 && \text{if} && [\![e_1]\!]_v > [\![e_2]\!]_v
\end{aligned}
$$

Semantics of Expressions and Guards

Consider a finite set $\mathtt{X}$ of variables. A valuation is a function $v : \mathtt{X} \to \mathbb{R}$.

Expressions: $[\![e]\!]_v$

$$
\begin{array}{rcll}
[\![c]\!]_v & = & c & [c \in \mathbb{Q}] \\
[\![x]\!]_v & = & v(x) & [x \in \mathtt{X}] \\
[\![e_1 + e_2]\!]_v & = & [\![e_1]\!]_v + [\![e_2]\!]_v & \\
[\![e_1 - e_2]\!]_v & = & [\![e_1]\!]_v - [\![e_2]\!]_v & \\
[\![e_1 * e_2]\!]_v & = & [\![e_1]\!]_v \times [\![e_2]\!]_v &
\end{array}
$$

Guards: $v \models g$

$$
\begin{array}{lll}
v \models e_1 < e_2 & \text{if} & [\![e_1]\!]_v < [\![e_2]\!]_v \\
v \models e_1 \leq e_2 & \text{if} & [\![e_1]\!]_v \leq [\![e_2]\!]_v \\
v \models e_1 = e_2 & \text{if} & [\![e_1]\!]_v = [\![e_2]\!]_v \\
v \models e_1 \neq e_2 & \text{if} & [\![e_1]\!]_v \neq [\![e_2]\!]_v \\
v \models e_1 \geq e_2 & \text{if} & [\![e_1]\!]_v \geq [\![e_2]\!]_v \\
v \models e_1 > e_2 & \text{if} & [\![e_1]\!]_v > [\![e_2]\!]_v
\end{array}
$$

Semantics of Operations

The semantics $[\![\mathtt{op}]\!]$ of an operation $\mathtt{op}$ is defined as a binary relation between valuations before $\mathtt{op}$ and valuations after $\mathtt{op}$:

$$[\![\mathtt{op}]\!] \quad \subseteq \quad (\mathtt{X} \to \mathbb{R}) \times (\mathtt{X} \to \mathbb{R})$$

Examples with $\mathtt{X} = \{\mathtt{x}, \mathtt{y}\}$

$$[\![\mathtt{x}{*}\mathtt{y} \leq 10]\!] = \{(v, v) \mid v(\mathtt{x}) \times v(\mathtt{y}) \leq 10\}$$
$$[\![\mathtt{x} := 3{*}\mathtt{x}]\!] = \{(v, v') \mid v'(\mathtt{x}) = 3 \times v(\mathtt{x}) \ \wedge \ v'(\mathtt{y}) = v(\mathtt{y})\}$$

Operations: $[\![\mathtt{op}]\!]$

$$(v, v') \in [\![g]\!] \quad \textit{if} \quad v \models g \ \textit{and} \ v' = v$$

$$(v, v') \in [\![x := e]\!] \quad \textit{if} \quad \begin{cases} v'(x) = [\![e]\!]_v \\ v'(y) = v'(y) \quad \text{for all } y \neq x \end{cases}$$

Semantics of Operations

The semantics $[\![\mathtt{op}]\!]$ of an operation $\mathtt{op}$ is defined as a binary relation between valuations before $\mathtt{op}$ and valuations after $\mathtt{op}$:

$$[\![\mathtt{op}]\!] \quad \subseteq \quad (\mathtt{X} \to \mathbb{R}) \times (\mathtt{X} \to \mathbb{R})$$

Examples with $\mathtt{X} = \{\mathtt{x}, \mathtt{y}\}$

$$[\![\mathtt{x}{*}\mathtt{y} \leq 10]\!] = \{(v, v) \mid v(\mathtt{x}) \times v(\mathtt{y}) \leq 10\}$$
$$[\![\mathtt{x} := 3{*}\mathtt{x}]\!] = \{(v, v') \mid v'(\mathtt{x}) = 3 \times v(\mathtt{x}) \ \wedge \ v'(\mathtt{y}) = v(\mathtt{y})\}$$

Operations: $[\![\mathtt{op}]\!]$

$$(v, v') \in [\![g]\!] \quad \textit{if} \quad v \models g \ \textit{and} \ v' = v$$

$$(v, v') \in [\![x := e]\!] \quad \textit{if} \quad \begin{cases} v'(x) = [\![e]\!]_v \\ v'(y) = v'(y) \quad \text{for all } y \neq x \end{cases}$$

Semantics of Operations

The semantics $[\![\mathtt{op}]\!]$ of an operation $\mathtt{op}$ is defined as a binary relation between valuations before $\mathtt{op}$ and valuations after $\mathtt{op}$:

$$[\![\mathtt{op}]\!] \quad \subseteq \quad (\mathtt{X} \to \mathbb{R}) \times (\mathtt{X} \to \mathbb{R})$$

Examples with $\mathtt{X} = \{\mathtt{x}, \mathtt{y}\}$

$$\begin{aligned} [\![\mathtt{x}{*}\mathtt{y} \leq 10]\!] &= \{(v, v) \mid v(\mathtt{x}) \times v(\mathtt{y}) \leq 10\} \\ [\![\mathtt{x} := 3{*}\mathtt{x}]\!] &= \{(v, v') \mid v'(\mathtt{x}) = 3 \times v(\mathtt{x}) \ \wedge \ v'(\mathtt{y}) = v(\mathtt{y})\} \end{aligned}$$

Operations: $[\![\mathtt{op}]\!]$

$$(v, v') \in [\![g]\!] \quad \textit{if} \quad v \models g \ \textit{ and } \ v' = v$$

$$(v, v') \in [\![x := e]\!] \quad \textit{if} \quad \begin{cases} v'(x) &= [\![e]\!]_v \\ v'(y) &= v'(y) \quad \text{for all } y \neq x \end{cases}$$

Operational Semantics of Control Flow Automata

Definition

The interpretation of a control flow automaton $\langle Q, q_{in}, q_{out}, \mathtt{X}, \rightarrow \rangle$ is the labeled transition system $\langle C, \mathit{Init}, \mathit{Out}, \mathtt{Op}, \rightarrow \rangle$ defined by:

- $C = Q \times (\mathtt{X} \rightarrow \mathbb{R})$
- $\mathit{Init} = \{q_{in}\} \times (\mathtt{X} \rightarrow \mathbb{R})$ and $\mathit{Out} = \{q_{out}\} \times (\mathtt{X} \rightarrow \mathbb{R})$
- $(q, v) \xrightarrow{\mathtt{op}} (q', v')$ if $q \xrightarrow{\mathtt{op}} q'$ and $(v, v') \in [\![\mathtt{op}]\!]$

Two kinds of labeled directed graphs

Control Flow Automata

Use: program source codes

- Syntactic objects
- Finite

Interpretations (LTS)

Use: program behaviors

- Semantic objects
- Uncountably infinite

Operational Semantics of Control Flow Automata

Definition

The interpretation of a control flow automaton $\langle Q, q_{in}, q_{out}, \mathtt{X}, \rightarrow \rangle$ is the labeled transition system $\langle C, \mathit{Init}, \mathit{Out}, \mathtt{Op}, \rightarrow \rangle$ defined by:

- $C = Q \times (\mathtt{X} \rightarrow \mathbb{R})$
- $\mathit{Init} = \{q_{in}\} \times (\mathtt{X} \rightarrow \mathbb{R})$ and $\mathit{Out} = \{q_{out}\} \times (\mathtt{X} \rightarrow \mathbb{R})$
- $(q, v) \xrightarrow{\mathtt{op}} (q', v')$ if $q \xrightarrow{\mathtt{op}} q'$ and $(v, v') \in [\![\mathtt{op}]\!]$

Two kinds of labeled directed graphs

Control Flow Automata

Use: program source codes

- Syntactic objects
- Finite

Interpretations (LTS)

Use: program behaviors

- Semantic objects
- Uncountably infinite

Control Paths, Execution Paths and Runs

A control path is a path in the control flow automaton:

$$q_0 \xrightarrow{\mathrm{op}_0} q_1 \cdots q_{k-1} \xrightarrow{\mathrm{op}_{k-1}} q_k$$

An execution path is a path in the labeled transition system:

$$(q_0, v_0) \xrightarrow{\mathrm{op}_0} (q_1, v_1) \cdots (q_{k-1}, v_{k-1}) \xrightarrow{\mathrm{op}_{k-1}} (q_k, v_k)$$

A run is an execution path that starts with an initial configuration:

$$(q_{in}, v_{in}) \xrightarrow{\mathrm{op}_0} (q_1, v_1) \cdots (q_{k-1}, v_{k-1}) \xrightarrow{\mathrm{op}_{k-1}} (q_k, v_k)$$

Control Paths, Execution Paths and Runs

A control path is a path in the control flow automaton:

$$q_0 \xrightarrow{\mathrm{op}_0} q_1 \cdots q_{k-1} \xrightarrow{\mathrm{op}_{k-1}} q_k$$

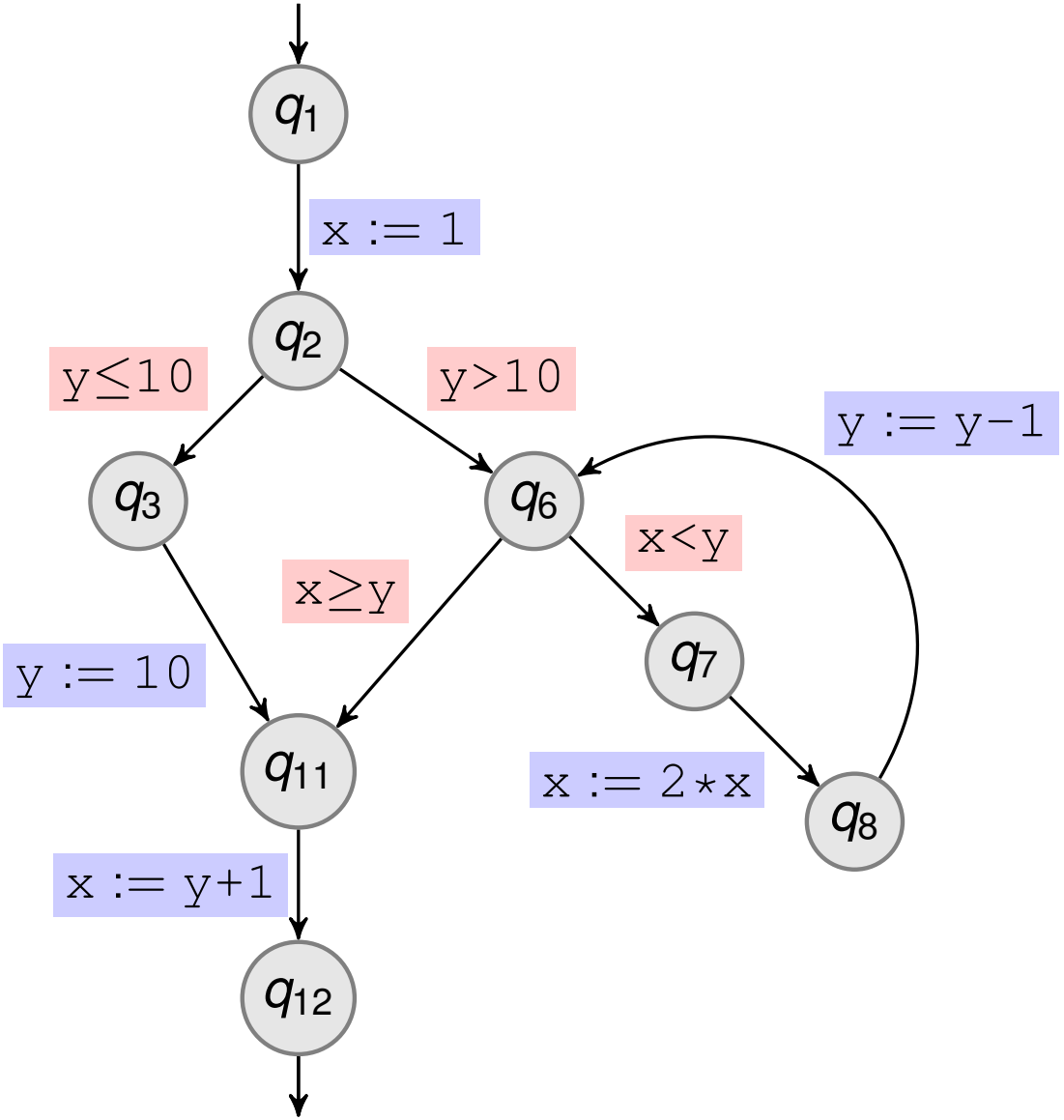
An execution path is a path in the labeled transition system:

$$(q_0, v_0) \xrightarrow{\mathrm{op}_0} (q_1, v_1) \cdots (q_{k-1}, v_{k-1}) \xrightarrow{\mathrm{op}_{k-1}} (q_k, v_k)$$

A run is an execution path that starts with an initial configuration:

$$(q_{in}, v_{in}) \xrightarrow{\mathrm{op}_0} (q_1, v_1) \cdots (q_{k-1}, v_{k-1}) \xrightarrow{\mathrm{op}_{k-1}} (q_k, v_k)$$

Execution Path: Example



$(q_1, -159, 27)$

$\downarrow$ x := 1

$(q_2, 1, 27)$

$\downarrow$ y>10

$(q_6, 1, 27)$

$\downarrow$ x<y

$(q_7, 1, 27)$

$\downarrow$ x := 2∗x

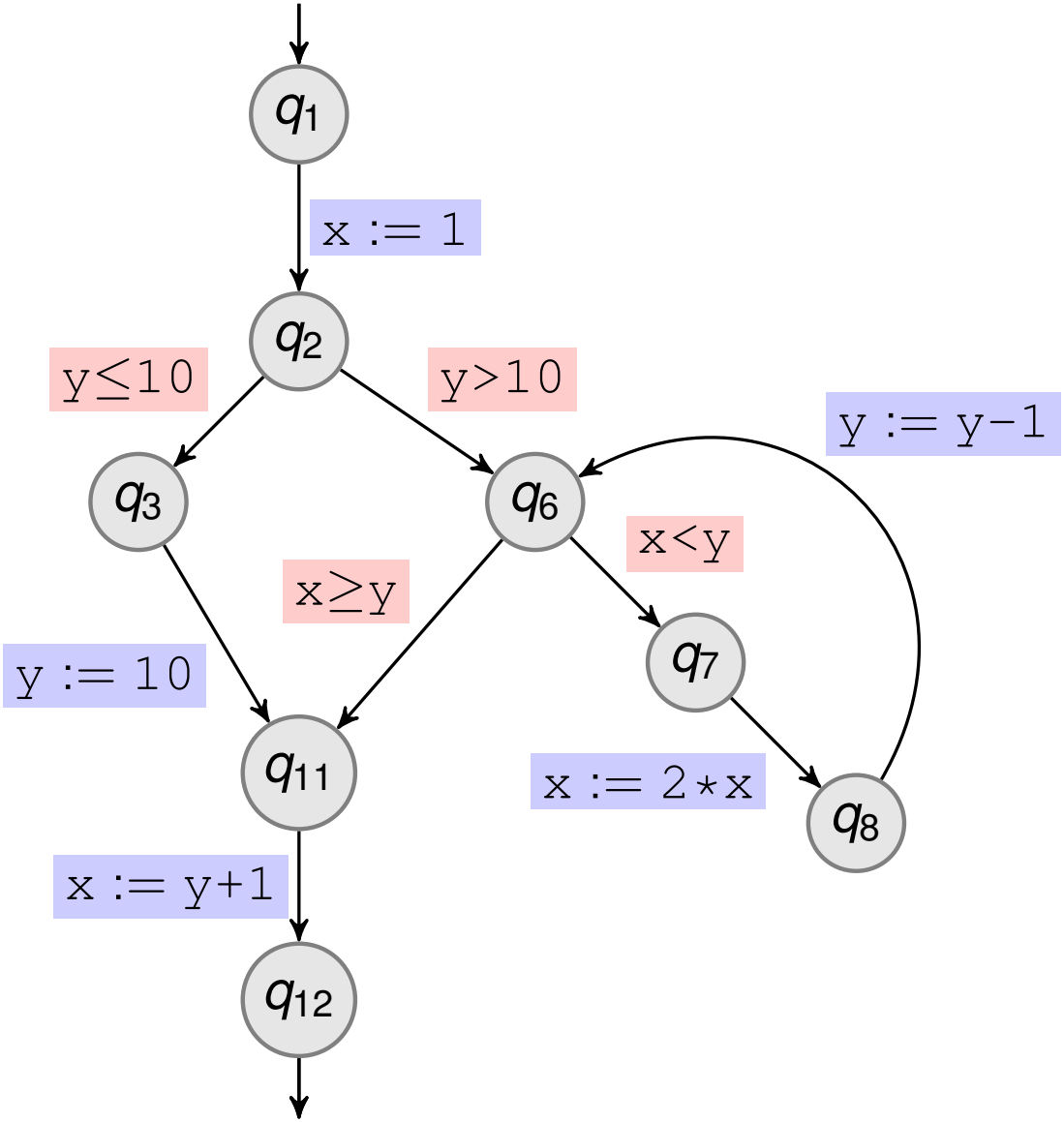
$(q_8, 2, 27)$

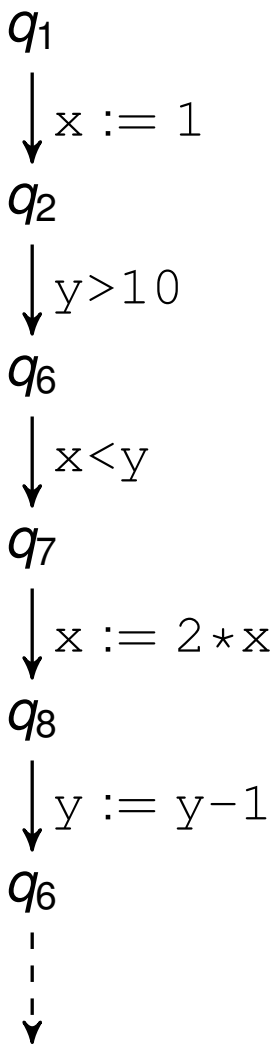
$\downarrow$ y := y−1

$(q_6, 2, 26)$

$\vdots$

Control Path: Example





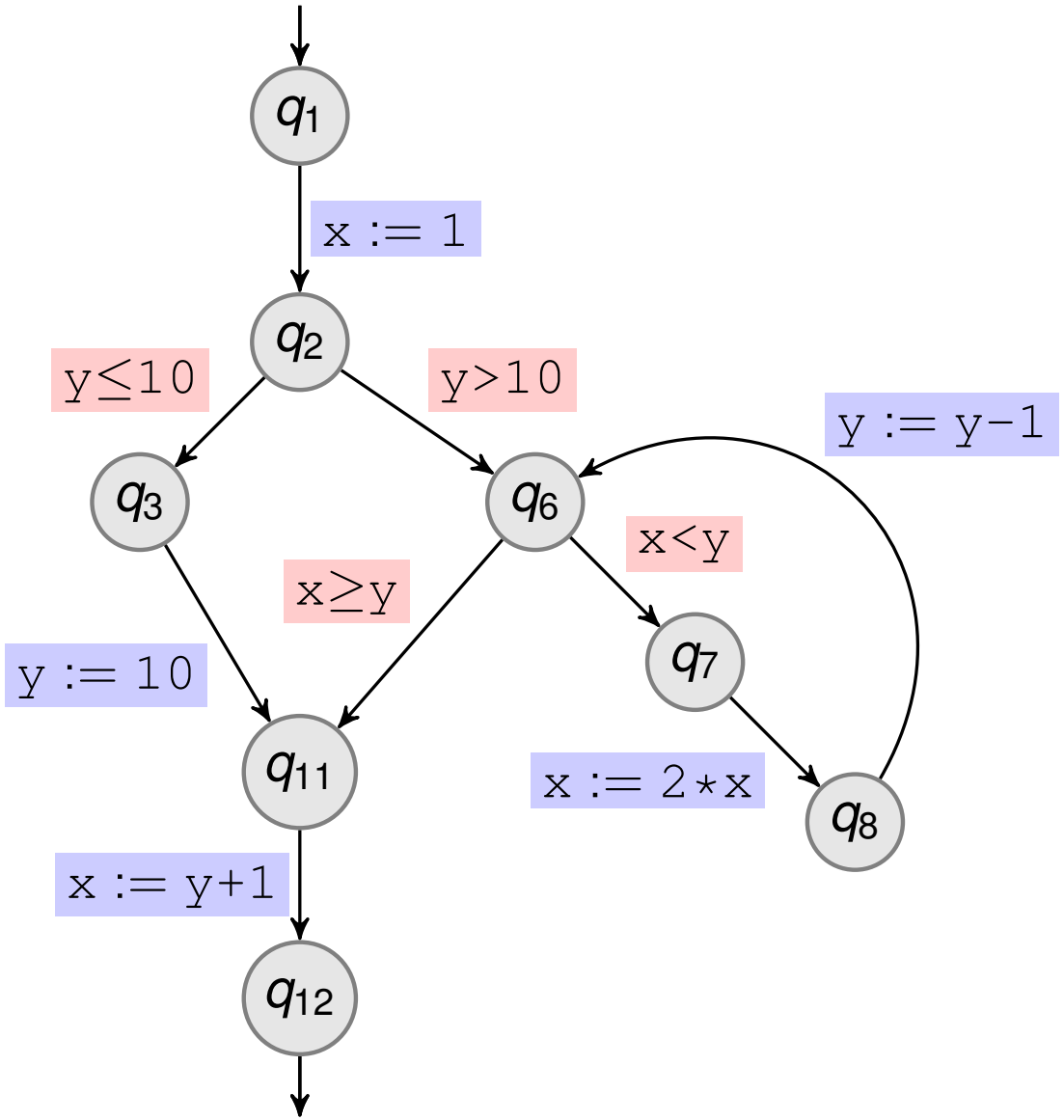
Outline — Control Flow Automata

Forward Reachability Set Post*

Set of all configurations that are reachable from an initial configuration

$$\begin{aligned}
\text{Post}^* &= \bigcup_{\rho\,:run} \{(q,v) \mid (q,v) \text{ occurs on } \rho\} \\
&= \bigcup_{i \in \mathbb{N}} \text{Post}^i(\mathit{Init}) \\
&= \bigcup_{q_{in} \xrightarrow{\mathrm{op}_0} \cdots \xrightarrow{\mathrm{op}_{k-1}} q} \{q\} \times (\llbracket \mathrm{op}_{k-1} \rrbracket \circ \cdots \circ \llbracket \mathrm{op}_0 \rrbracket)\,[(\mathrm{X} \to \mathbb{R})]
\end{aligned}$$

Forward Reachability Set Post* on Running Example



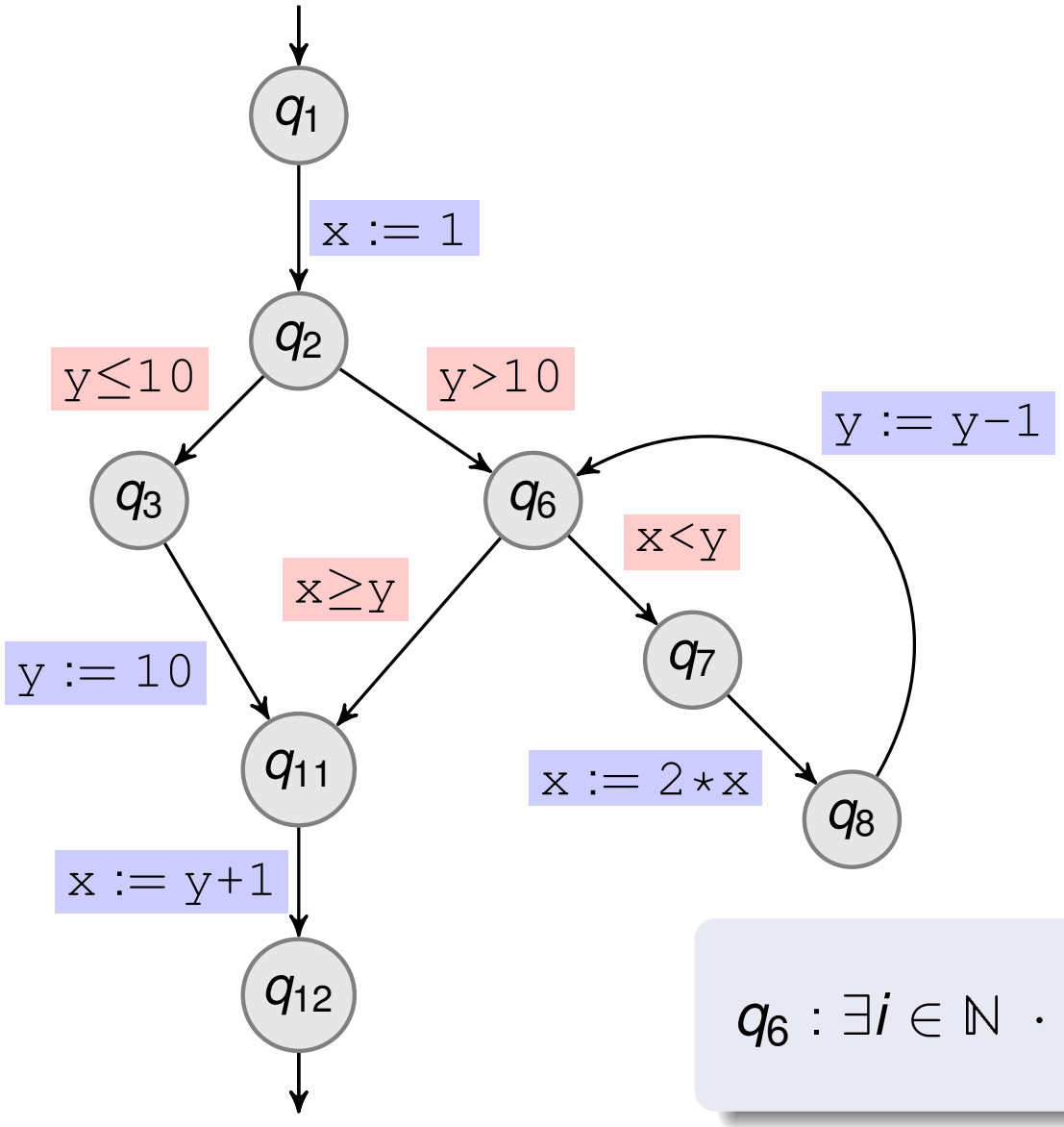
$q_1 : \mathbb{R} \times \mathbb{R}$

$q_2 : \{1\} \times \mathbb{R}$

$q_3 : \{1\} \times]-\infty, 10]$

$q_6 : \{1\} \times]10, +\infty[\cup$
$\{2\} \times]9, +\infty[\cup$
$\{4\} \times]8, +\infty[\cup$
$\ldots$

Forward Reachability Set Post* on Running Example



$q_1 : \mathbb{R} \times \mathbb{R}$

$q_2 : \{1\} \times \mathbb{R}$

$q_3 : \{1\} \times]-\infty, 10]$

$q_6 : \{1\} \times]10, +\infty[\cup$
$\{2\} \times]9, +\infty[\cup$
$\{4\} \times]8, +\infty[\cup$
$\ldots$

$$q_6 : \exists i \in \mathbb{N} \cdot \begin{cases} x = 2^i \ \wedge \ y + i > 10 \ \wedge \\ i \geq 1 \implies 2^{i-1} < y + 1 \end{cases}$$

Backward Reachability Set Pre^*

Set of all configurations that can reach an exit configuration

$$\mathrm{Pre}^* = \bigcup_{i \in \mathbb{N}} \mathrm{Pre}^i(Out)$$

$$= \bigcup_{q \xrightarrow{\mathrm{op}_0} \cdots \xrightarrow{\mathrm{op}_{k-1}} q_{out}} \{q\} \times \left([\![\mathrm{op}_0]\!]^{-1} \circ \cdots \circ [\![\mathrm{op}_{k-1}]\!]^{-1} \right) [(\mathrm{X} \to \mathbb{R})]$$

$$= \bigcup_{q \xrightarrow{\mathrm{op}_0} \cdots \xrightarrow{\mathrm{op}_{k-1}} q_{out}} \{q\} \times \left(([\![\mathrm{op}_{k-1}]\!] \circ \cdots \circ [\![\mathrm{op}_0]\!])^{-1} \right) [(\mathrm{X} \to \mathbb{R})]$$

Verification of Control Flow Automata

Goal (Repetition)

Check that there is no run that visits a location q contained in a given set $Q_{BAD} \subseteq Q$ of bad locations.

Define the set *Bad* of bad configurations by: $Bad = Q_{BAD} \times (\mathbb{X} \rightarrow \mathbb{R})$.

Goal (Equivalent Formulation)

Check that Post^* is disjoint from *Bad*

Undecidability

The *location reachability* and *configuration reachability* problems are both undecidable for control flow automata.

Proof by reduction to location reachability in two-counters machines.

Two-Counters Machines as Control Flow Automata

Two-Counters (Minsky) Machines

Finite-state automaton extended with:

- two counters over nonnegative integers
- test for zero, increment and guarded decrement

Reachability is undecidable for this class.

Any two-counters machine can (effectively) be represented as a control flow automaton in this restricted class:

- two variables: $\mathtt{X} = \{c_1, c_2\}$
- allowed guards: $x = 0$ and $x \neq 0$ for each $x \in \mathtt{X}$
- allowed assignments: $x := x{+}1$ and $x := x{-}1$ for each $x \in \mathtt{X}$

Two-Counters Machines as Control Flow Automata

Two-Counters (Minsky) Machines

Finite-state automaton extended with:

- two counters over nonnegative integers
- test for zero, increment and guarded decrement

Reachability is undecidable for this class.

Any two-counters machine can (effectively) be represented as a control flow automaton in this restricted class:

- two variables: $\mathtt{X} = \{c_1, c_2\}$
- allowed guards: $x = 0$ and $x \neq 0$ for each $x \in \mathtt{X}$
- allowed assignments: $x := x{+}1$ and $x := x{-}1$ for each $x \in \mathtt{X}$

Tentative Solution: Approximation Techniques

Definition

An invariant is any set $Inv \subseteq C$ such that $\text{Post}^* \subseteq Inv$.

Idea:

1. Compute an invariant *Inv* (easier to compute than Post^*)
2. If *Inv* is disjoint from *Bad* then Post^* is also disjoint from *Bad*

Rest of the lecture:

Computation of precise enough invariants

Tentative Solution: Approximation Techniques

Definition

An invariant is any set $Inv \subseteq C$ such that $\text{Post}^* \subseteq Inv$.

Idea:

1. Compute an invariant *Inv* (easier to compute than Post^*)
2. If *Inv* is disjoint from *Bad* then Post^* is also disjoint from *Bad*

Rest of the lecture:

Computation of precise enough invariants

Summary

- Computational model for programs: control flow automata
  - syntax
  - semantics
- Undecidability in general of model-checking for control flow automata
- Tentative solution: computation of invariants

Part III

Data Flow Analysis

Outline — Data Flow Analysis

Outline — Data Flow Analysis

Short Introduction to Data Flow Analysis

Tentative Definition

Compile-time techniques to gather run-time information about data in programs without actually running them

Applications

Code optimization

- Avoid *redundant* computations (e.g. reuse available results)
- Avoid *superfluous* computations (e.g. eliminate dead code)

Code validation

- Invariant generation

Conservative approximations

Live Variables Analysis: Definition

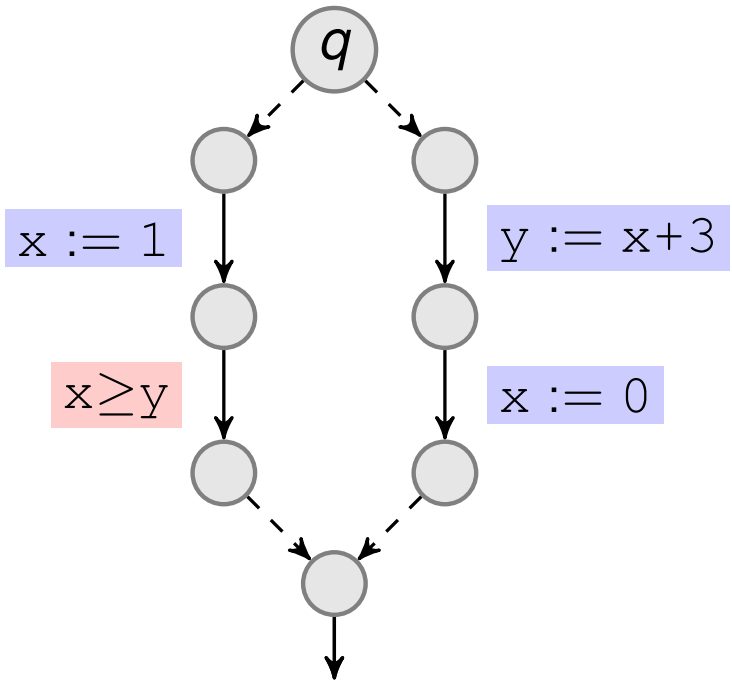
Definition

A variable x is live at location q if there exists a control path starting from q where x is used before it is modified.

Live Variables Analysis: Definition

Definition

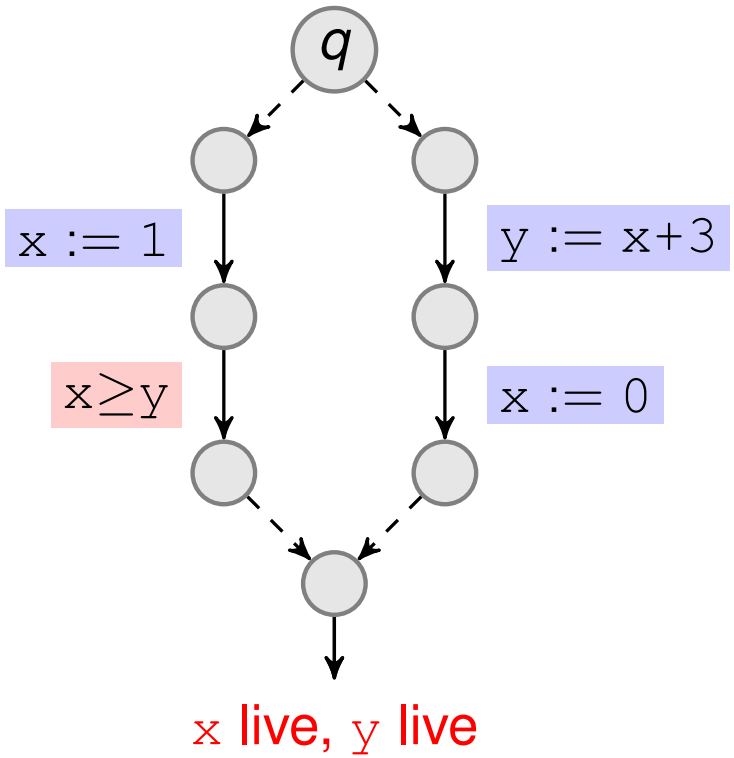
A variable x is live at location q if there exists a control path starting from q where x is used before it is modified.



Live Variables Analysis: Definition

Definition

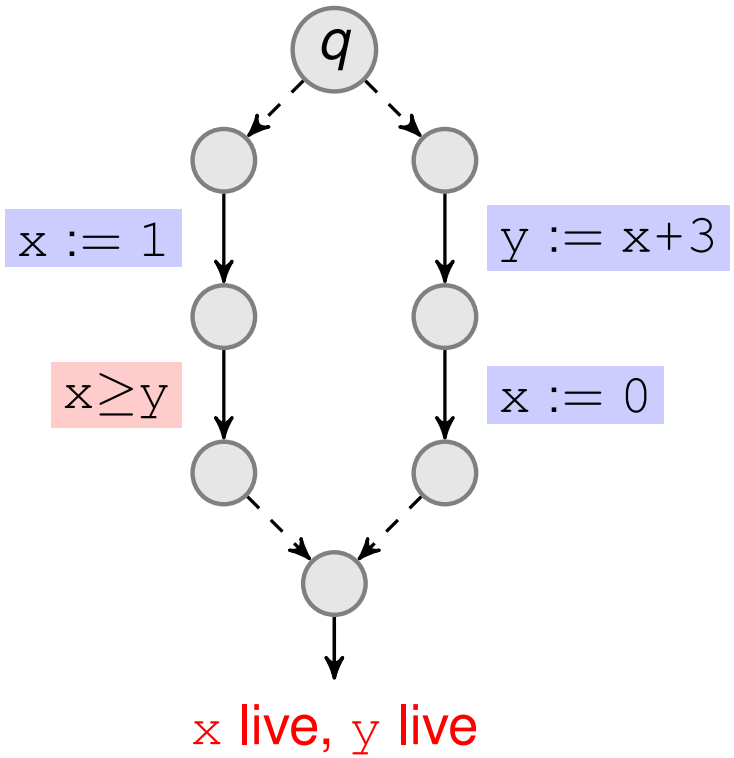
A variable x is live at location q if there exists a control path starting from q where x is used before it is modified.

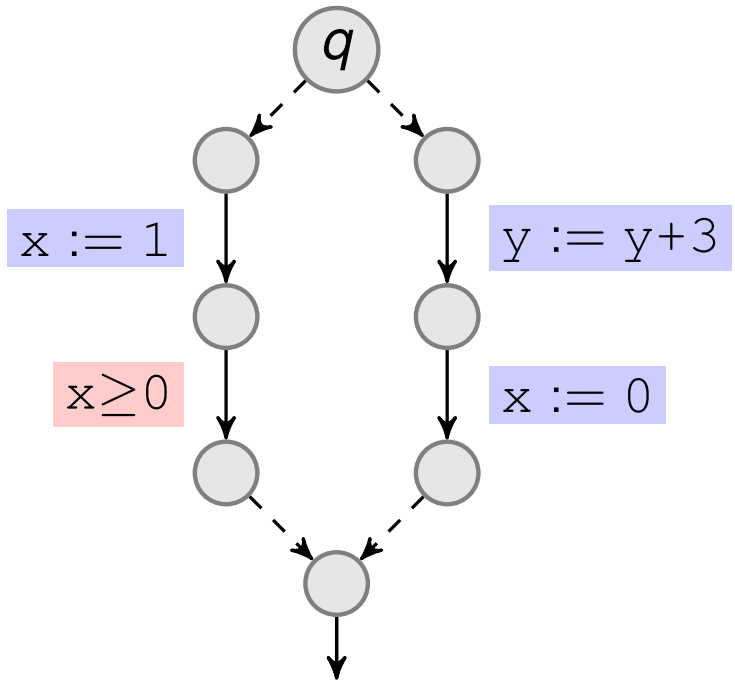


Live Variables Analysis: Definition

Definition

A variable x is live at location q if there exists a control path starting from q where x is used before it is modified.

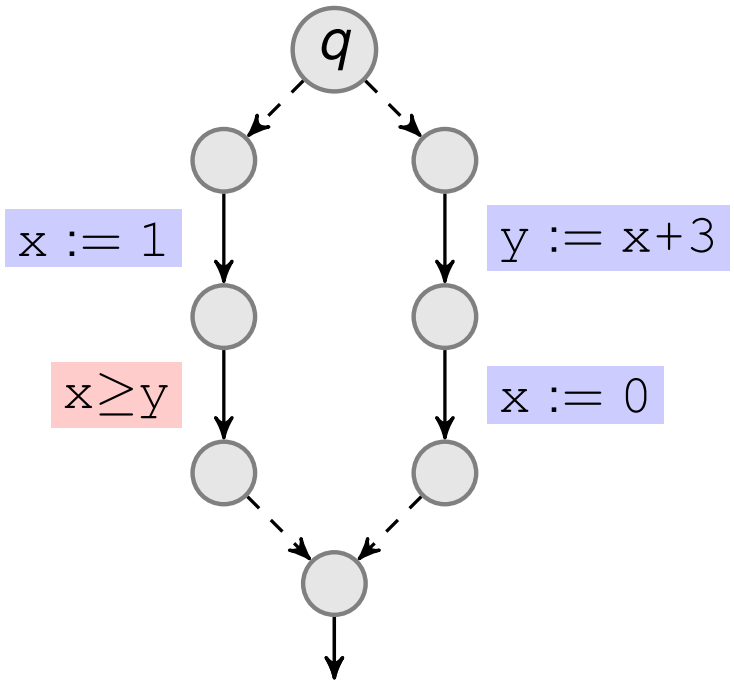




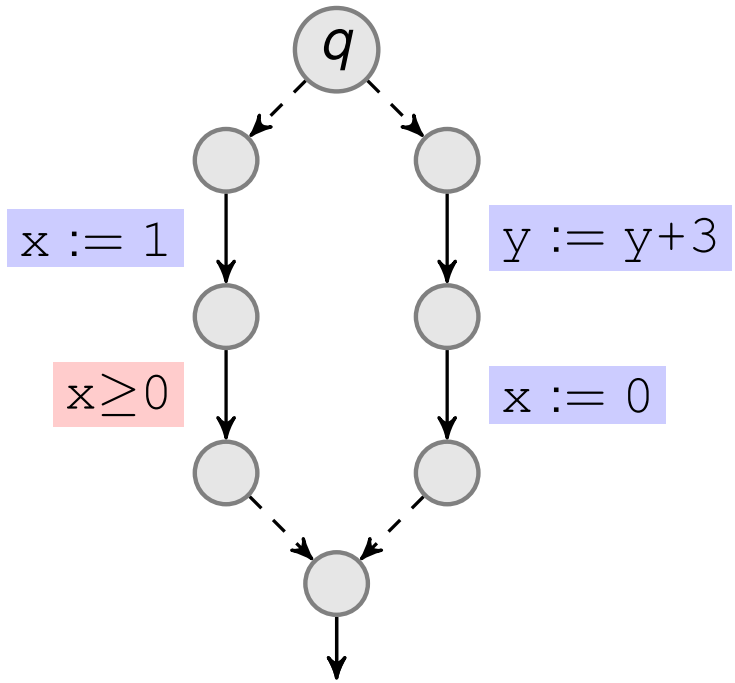
Live Variables Analysis: Definition

Definition

A variable x is live at location q if there exists a control path starting from q where x is used before it is modified.

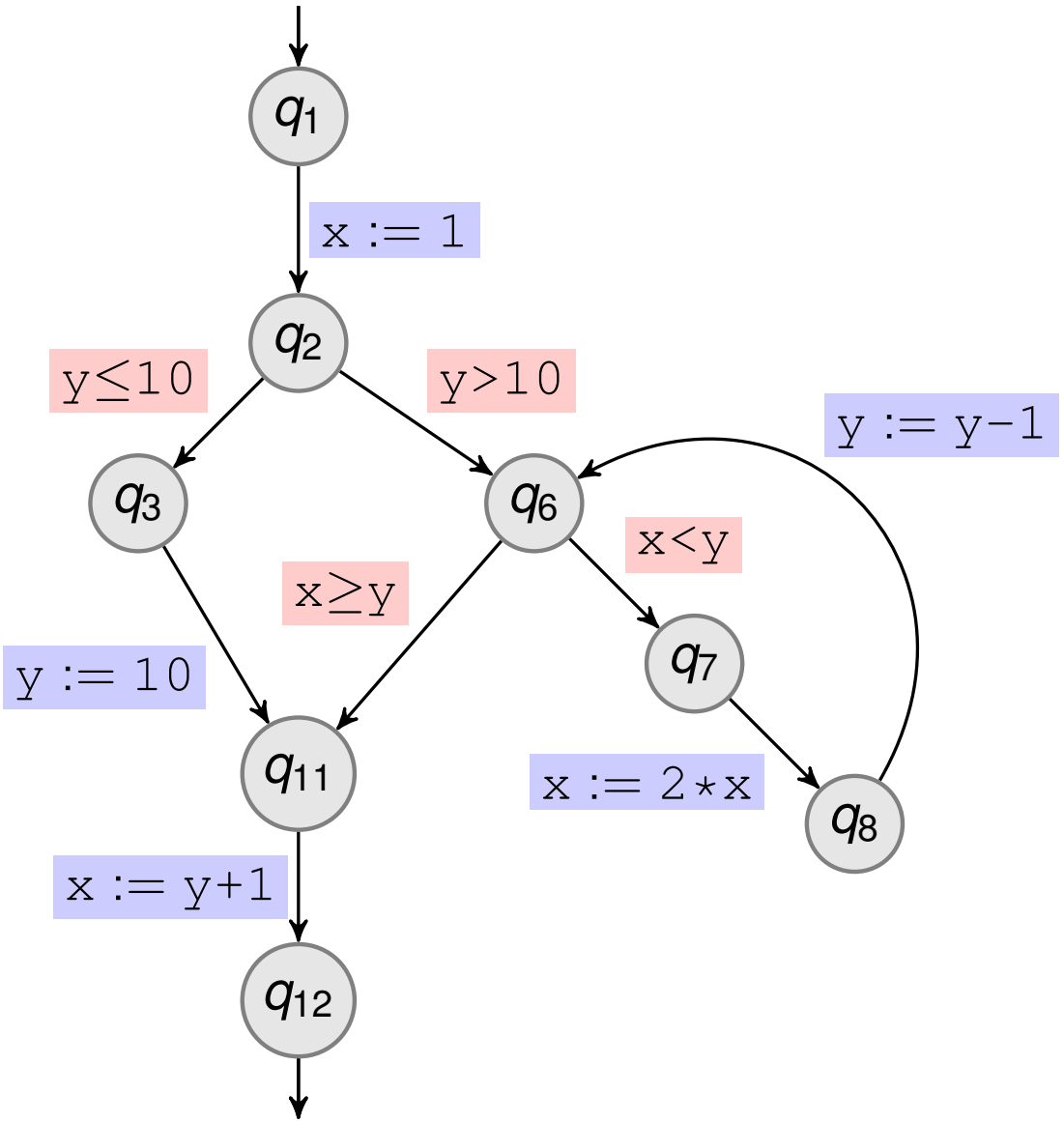


x live, y live

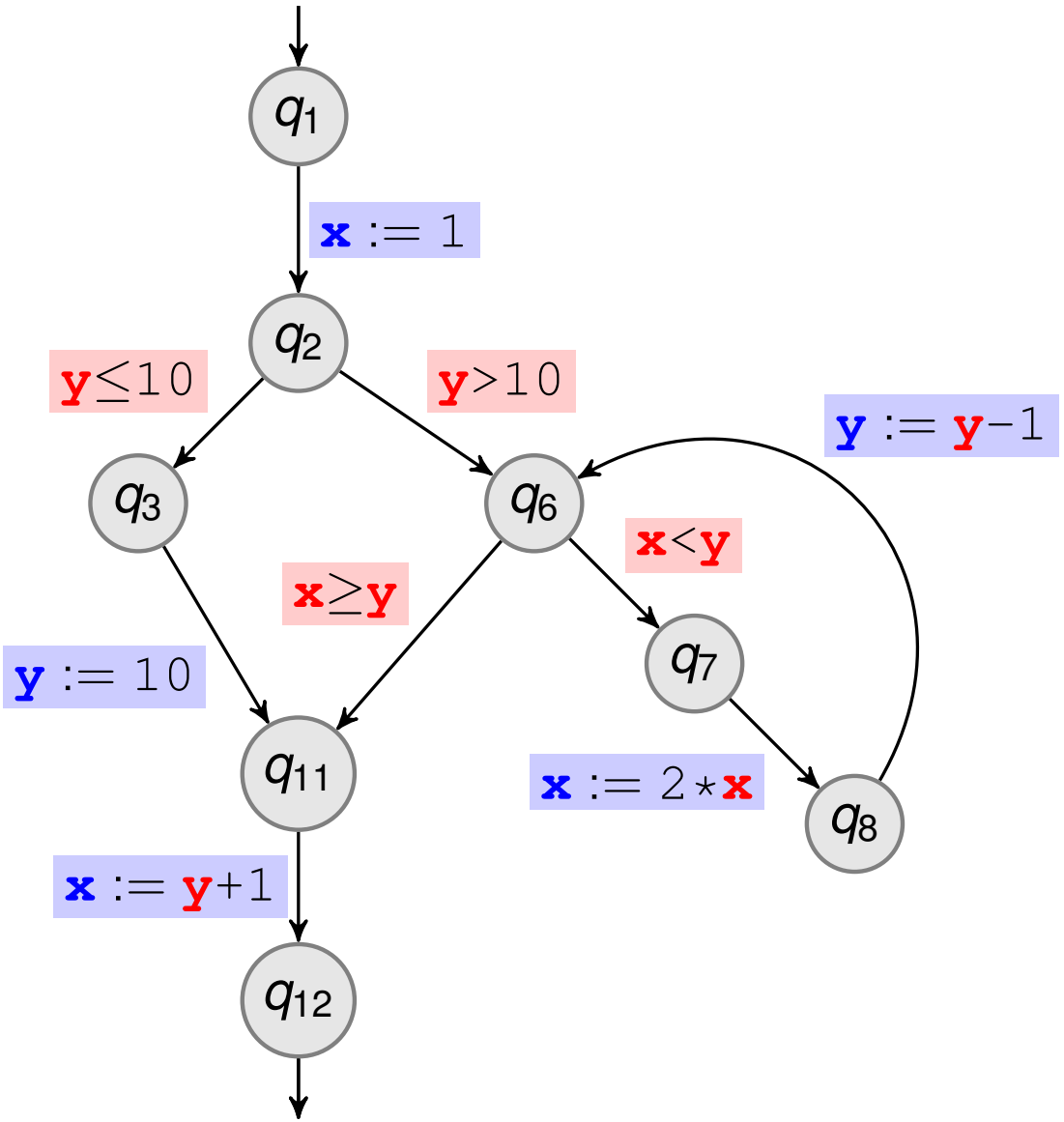


x not live, y live

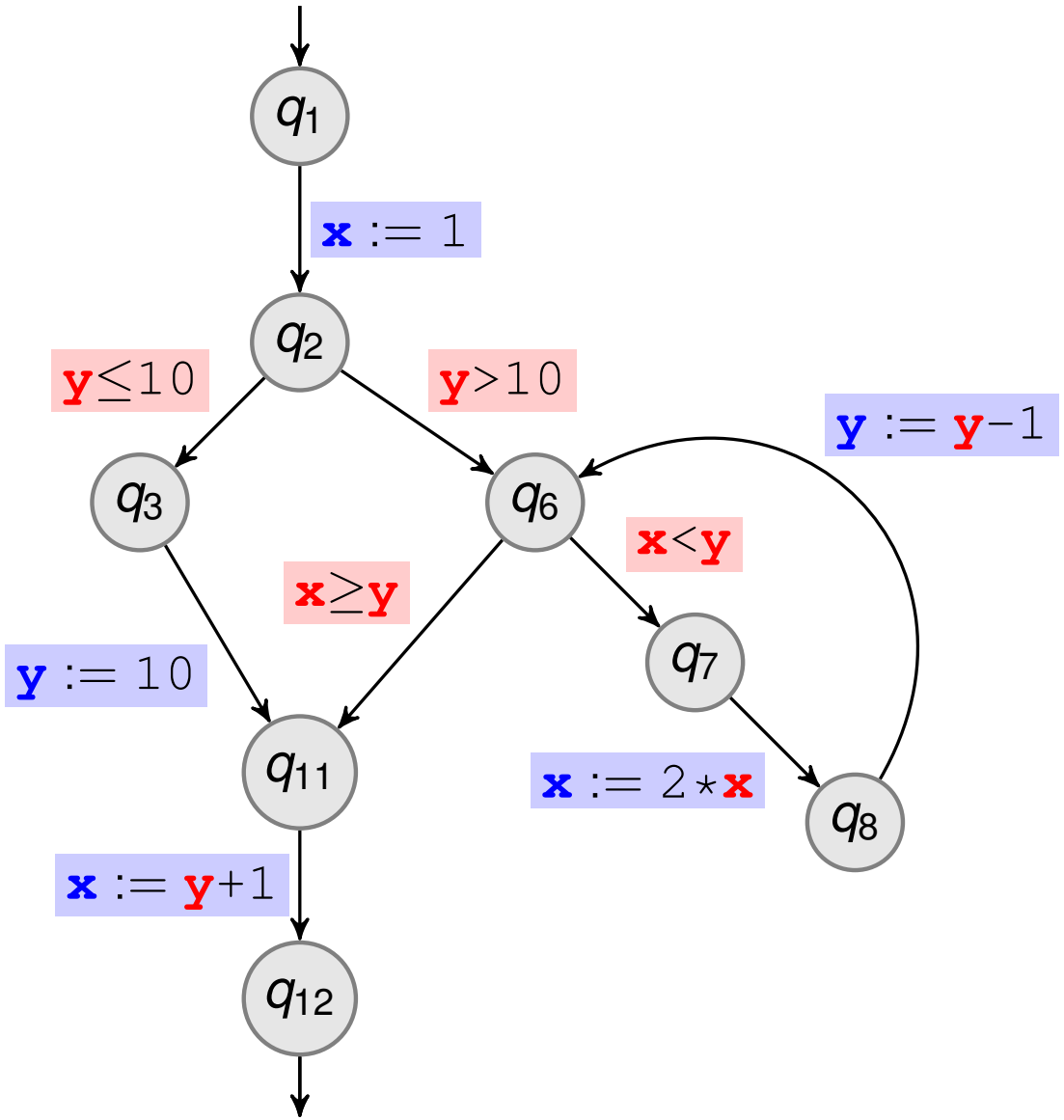
Live Variables Analysis: Running Example



Live Variables Analysis: Running Example



Live Variables Analysis: Running Example



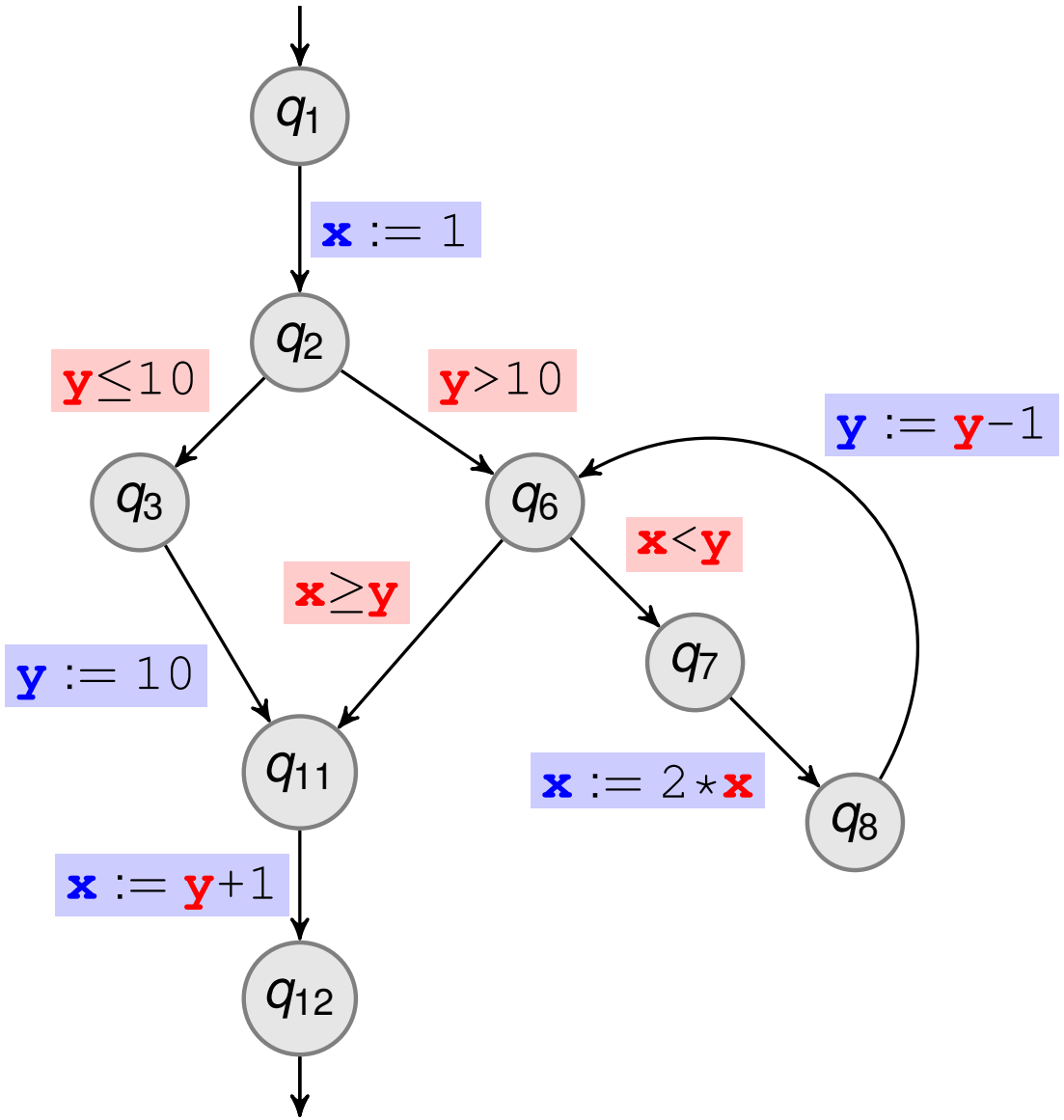
0 : Initialization

1 : Local information

2 : Propagation (←)

	x	y
q_1		
q_2		
q_3		
q_6		
q_7		
q_8		
q_{11}		
q_{12}		

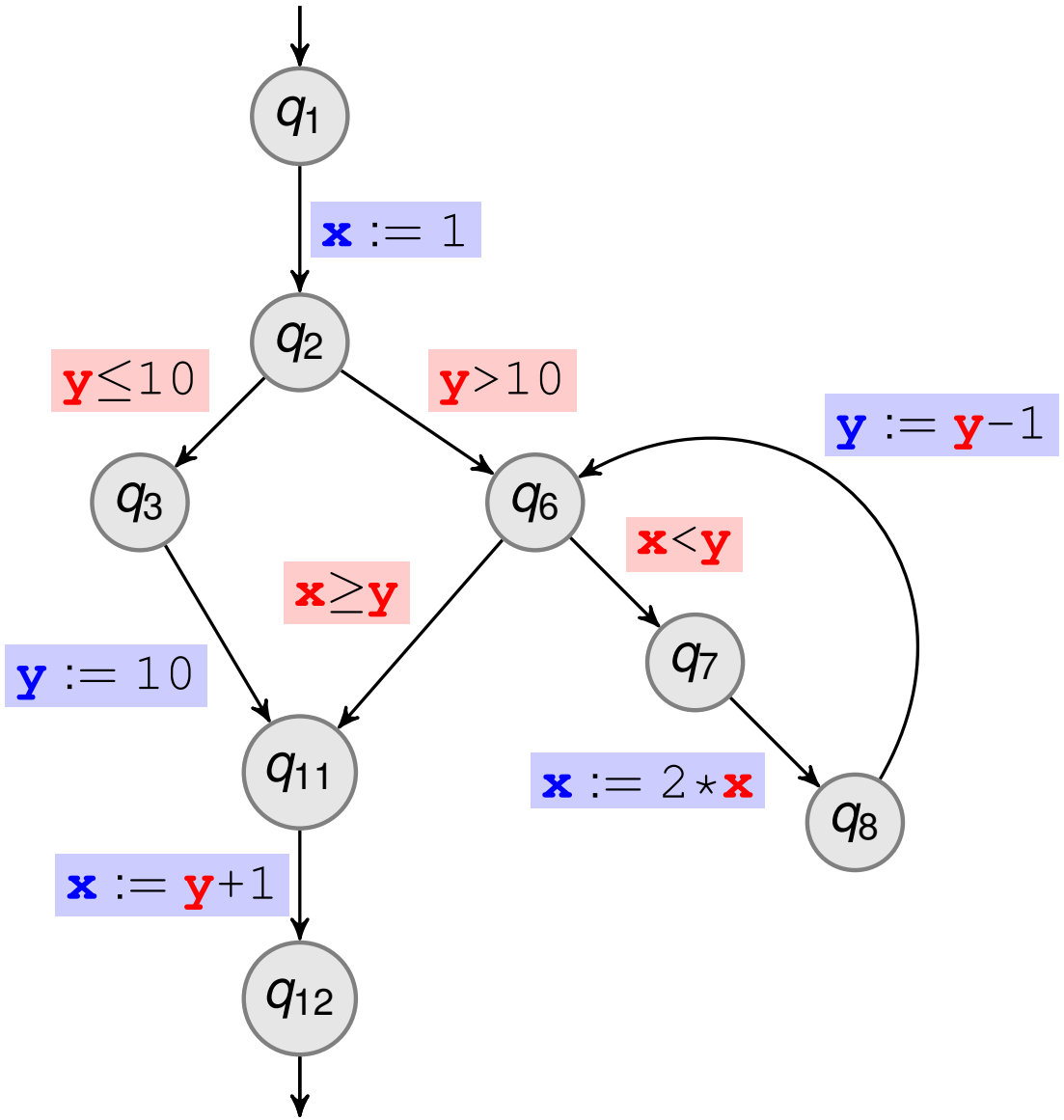
Live Variables Analysis: Running Example



0 : Initialization
1 : Local information
2 : Propagation (←)

	x	y
q_1		
q_2		•
q_3		
q_6	•	•
q_7	•	
q_8		•
q_{11}		•
q_{12}		

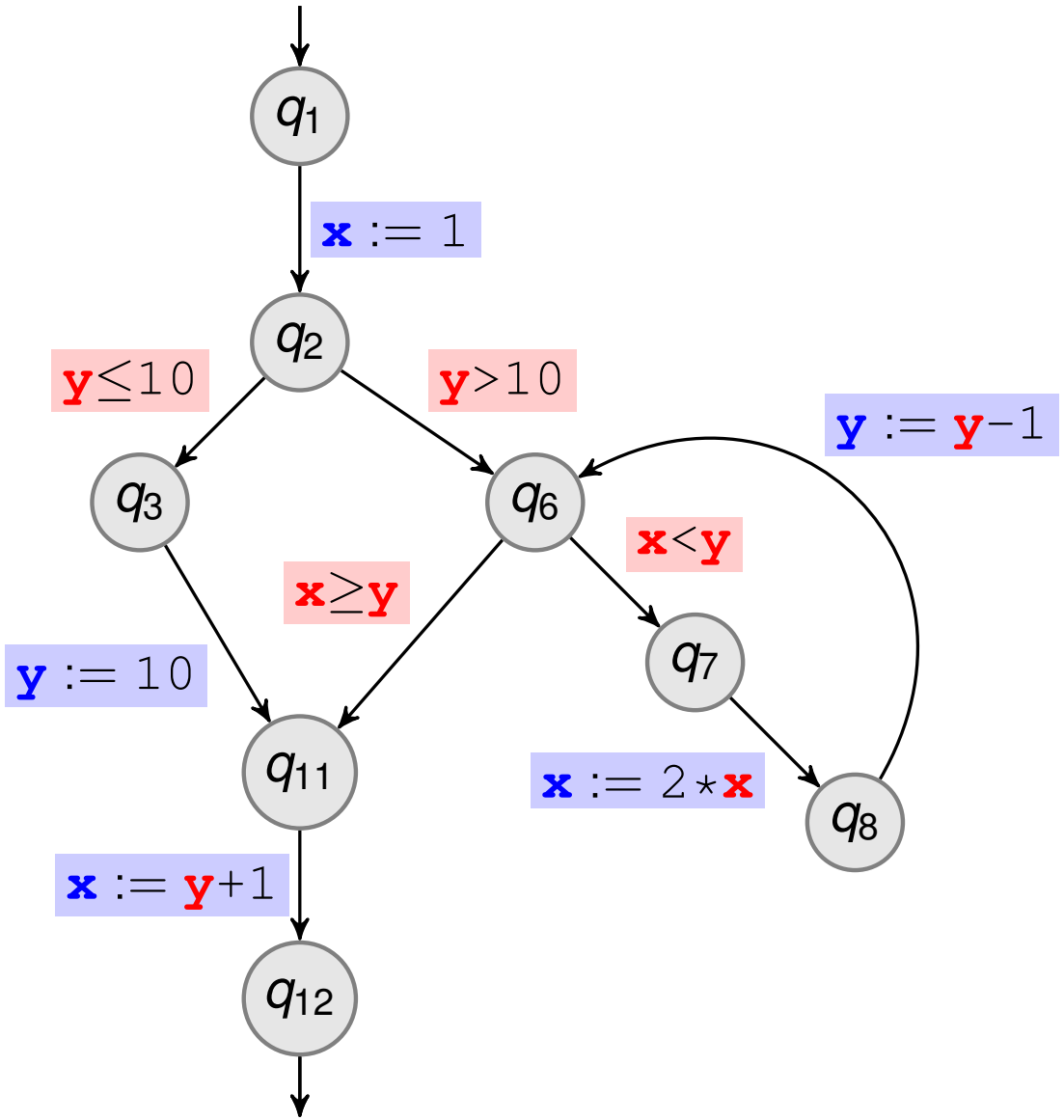
Live Variables Analysis: Running Example



0 : Initialization
1 : Local information
2 : Propagation (←)

	x	y
q_1		
q_2	●	●
q_3		
q_6	●	●
q_7	●	
q_8		●
q_{11}		●
q_{12}		

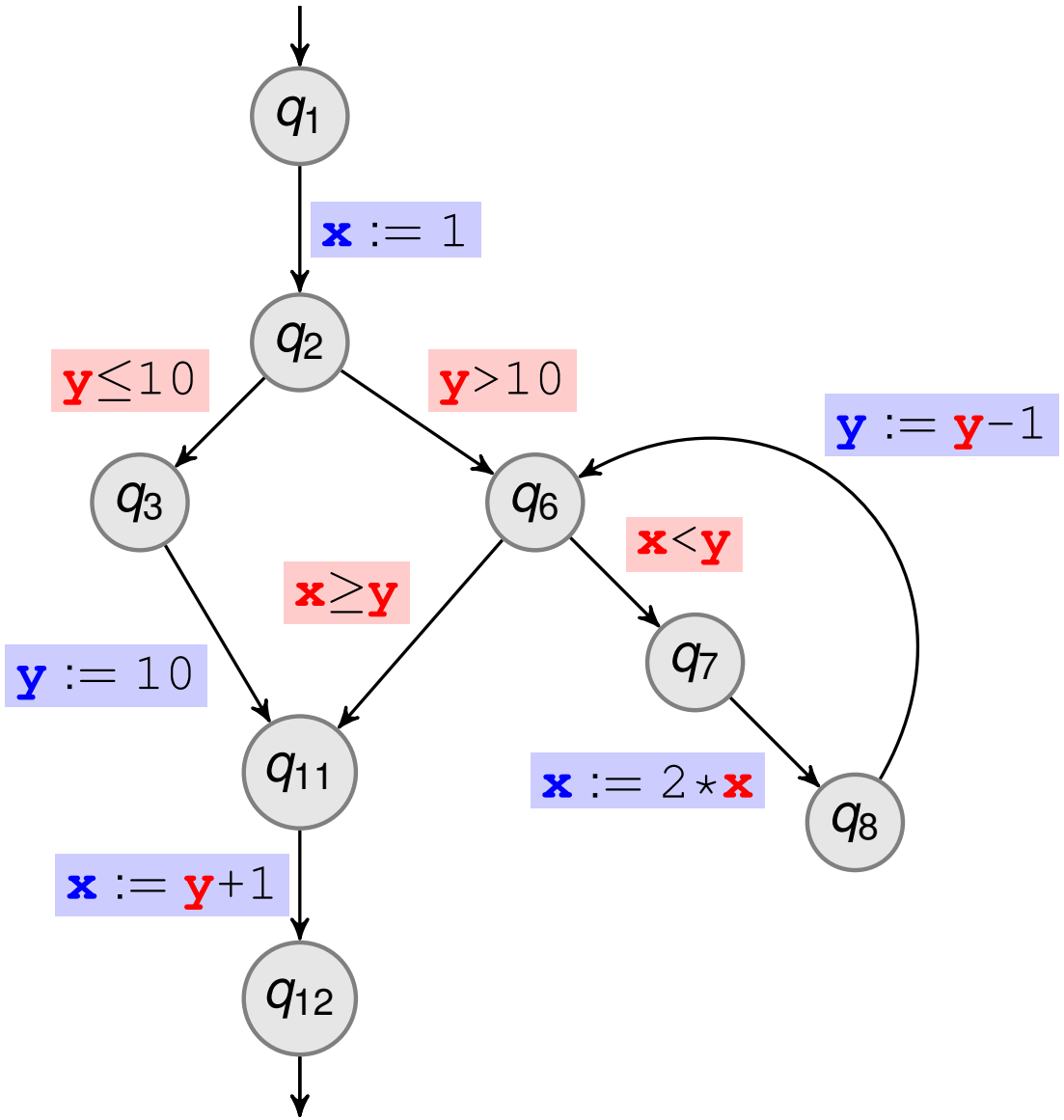
Live Variables Analysis: Running Example



0 : Initialization
1 : Local information
2 : Propagation (←)

	x	y
q_1		
q_2	•	•
q_3		
q_6	•	•
q_7	•	
q_8		•
q_{11}		•
q_{12}		

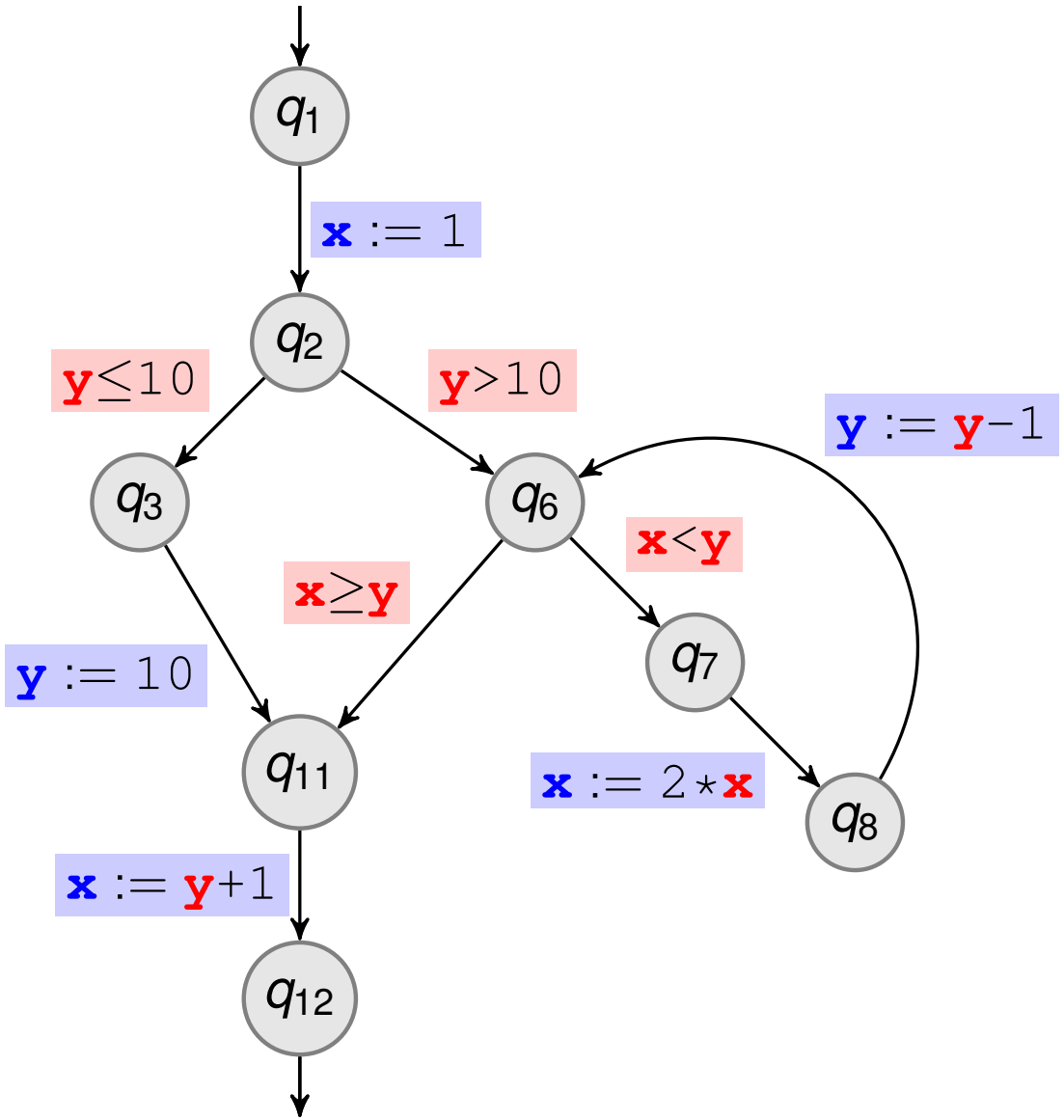
Live Variables Analysis: Running Example



0 : Initialization
1 : Local information
2 : Propagation (←)

	x	y
q_1		●
q_2	●	●
q_3		
q_6	●	●
q_7	●	
q_8		●
q_{11}		●
q_{12}		

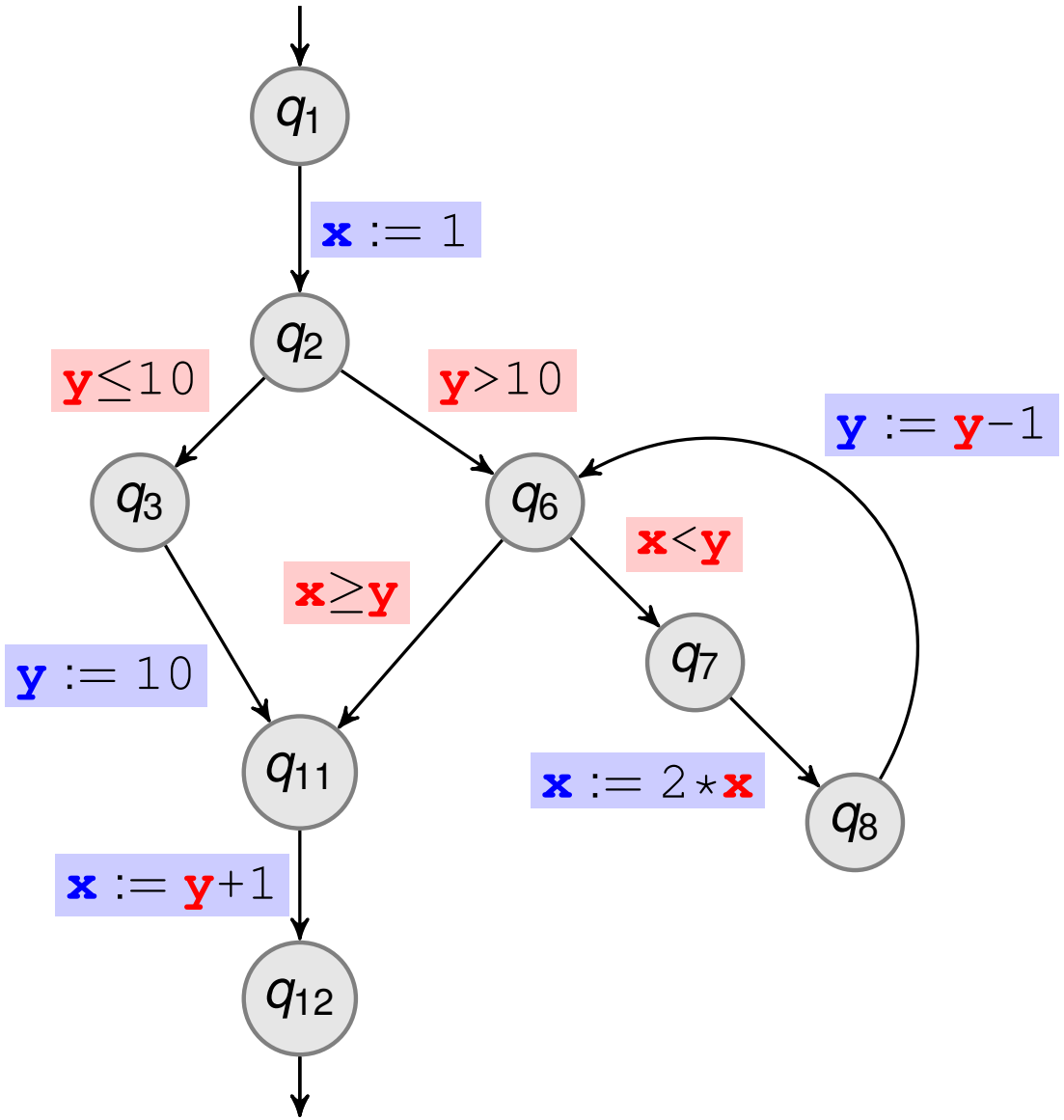
Live Variables Analysis: Running Example



0 : Initialization
1 : Local information
2 : Propagation (←)

	x	y
q_1		•
q_2	•	•
q_3		
q_6	•	•
q_7	•	
q_8		•
q_{11}		•
q_{12}		

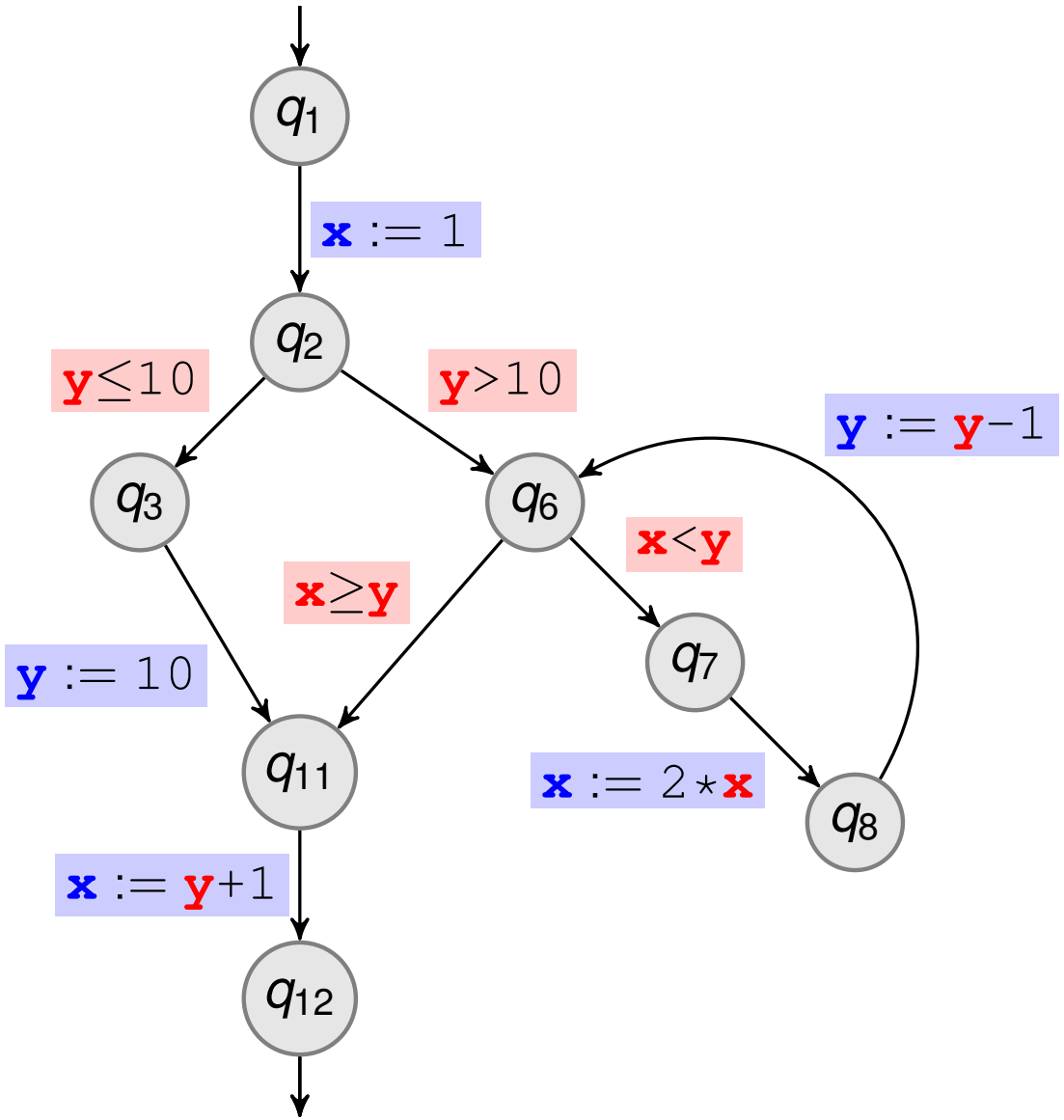
Live Variables Analysis: Running Example



0 : Initialization
1 : Local information
2 : Propagation (←)

	x	y
q_1		•
q_2	•	•
q_3		
q_6	•	•
q_7	•	
q_8	•	•
q_{11}		•
q_{12}		

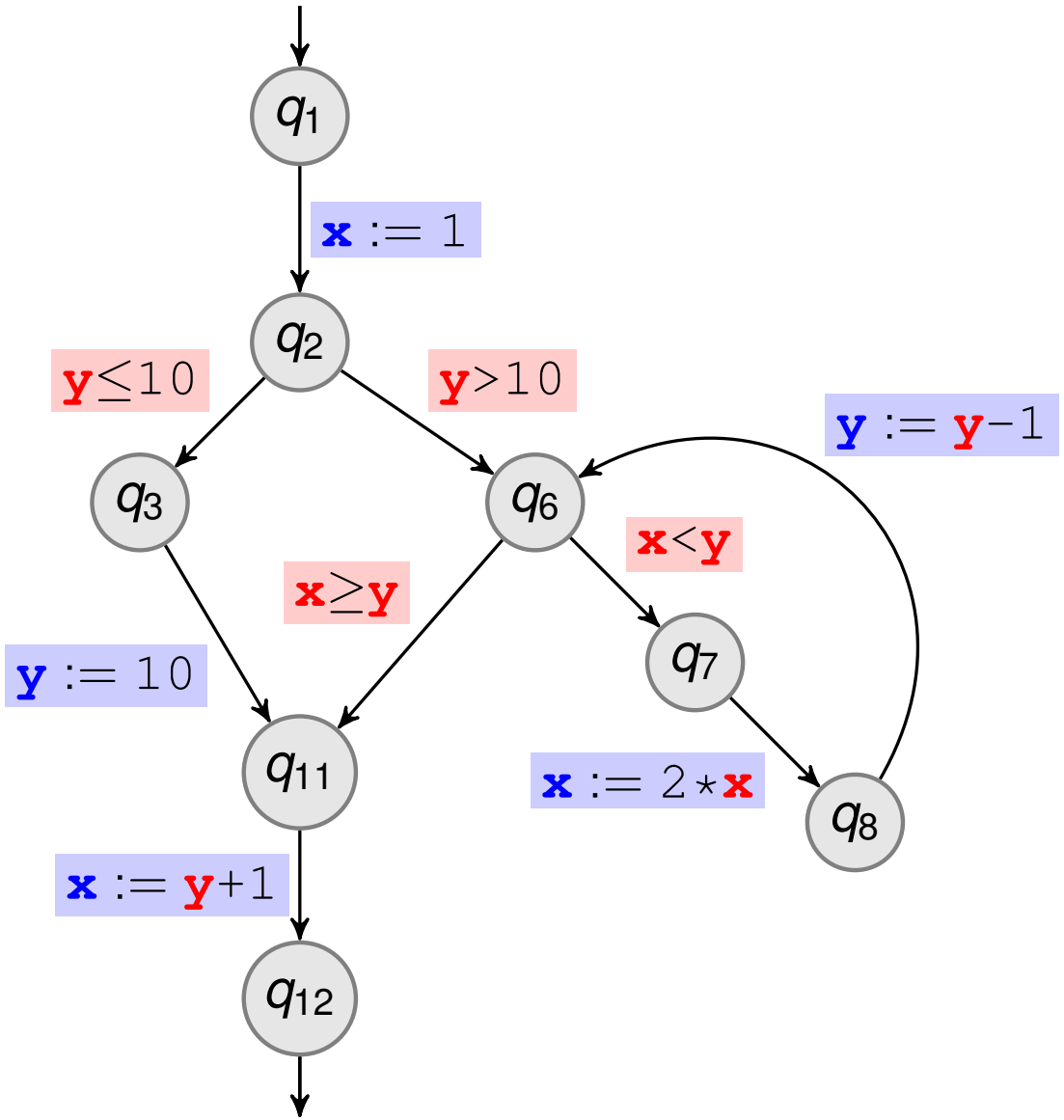
Live Variables Analysis: Running Example



0 : Initialization
1 : Local information
2 : Propagation (←)

	x	y
q_1		•
q_2	•	•
q_3		
q_6	•	•
q_7	•	
q_8	•	•
q_{11}		•
q_{12}		

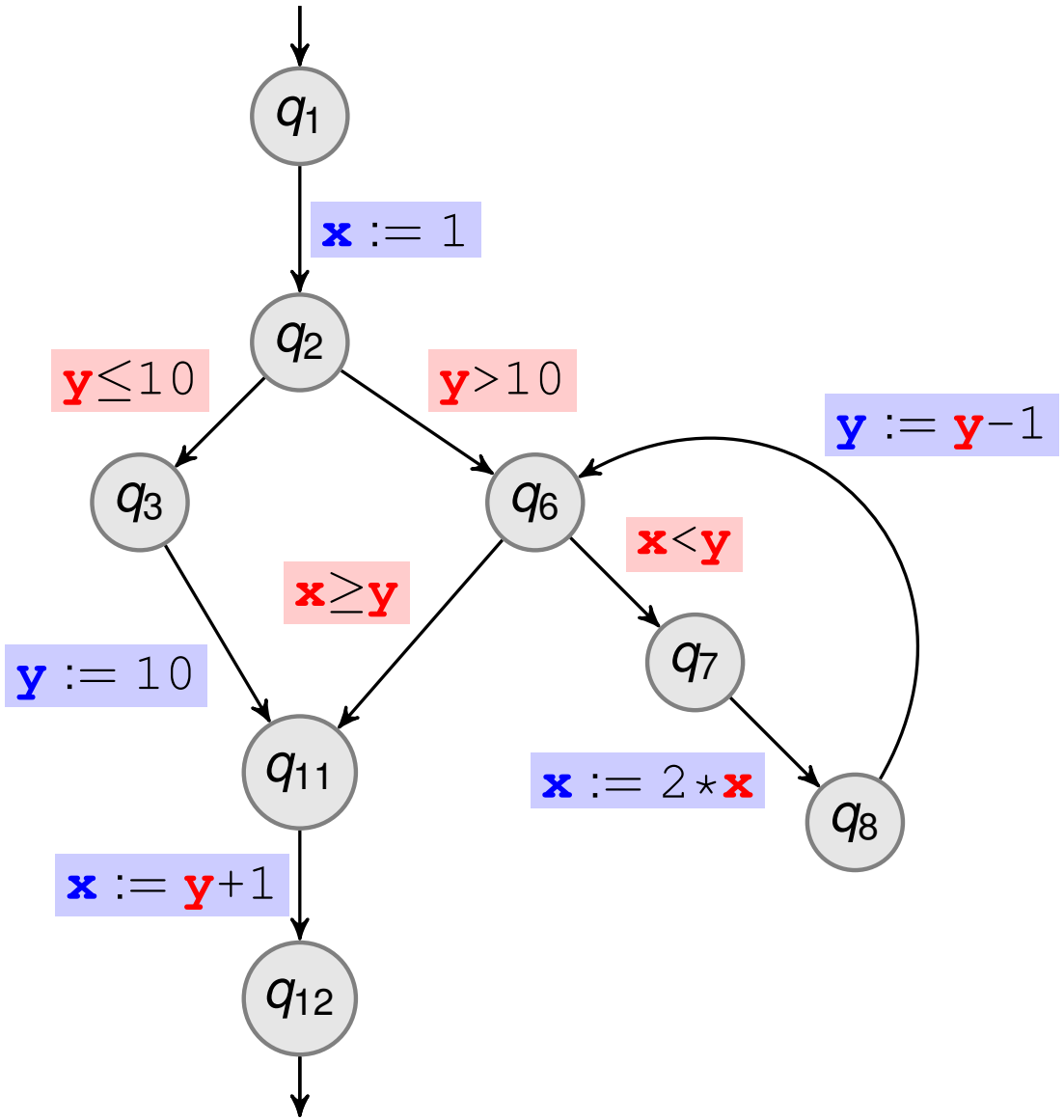
Live Variables Analysis: Running Example



0 : Initialization
1 : Local information
2 : Propagation (←)

	x	y
q_1		●
q_2	●	●
q_3		
q_6	●	●
q_7	●	●
q_8	●	●
q_{11}		●
q_{12}		

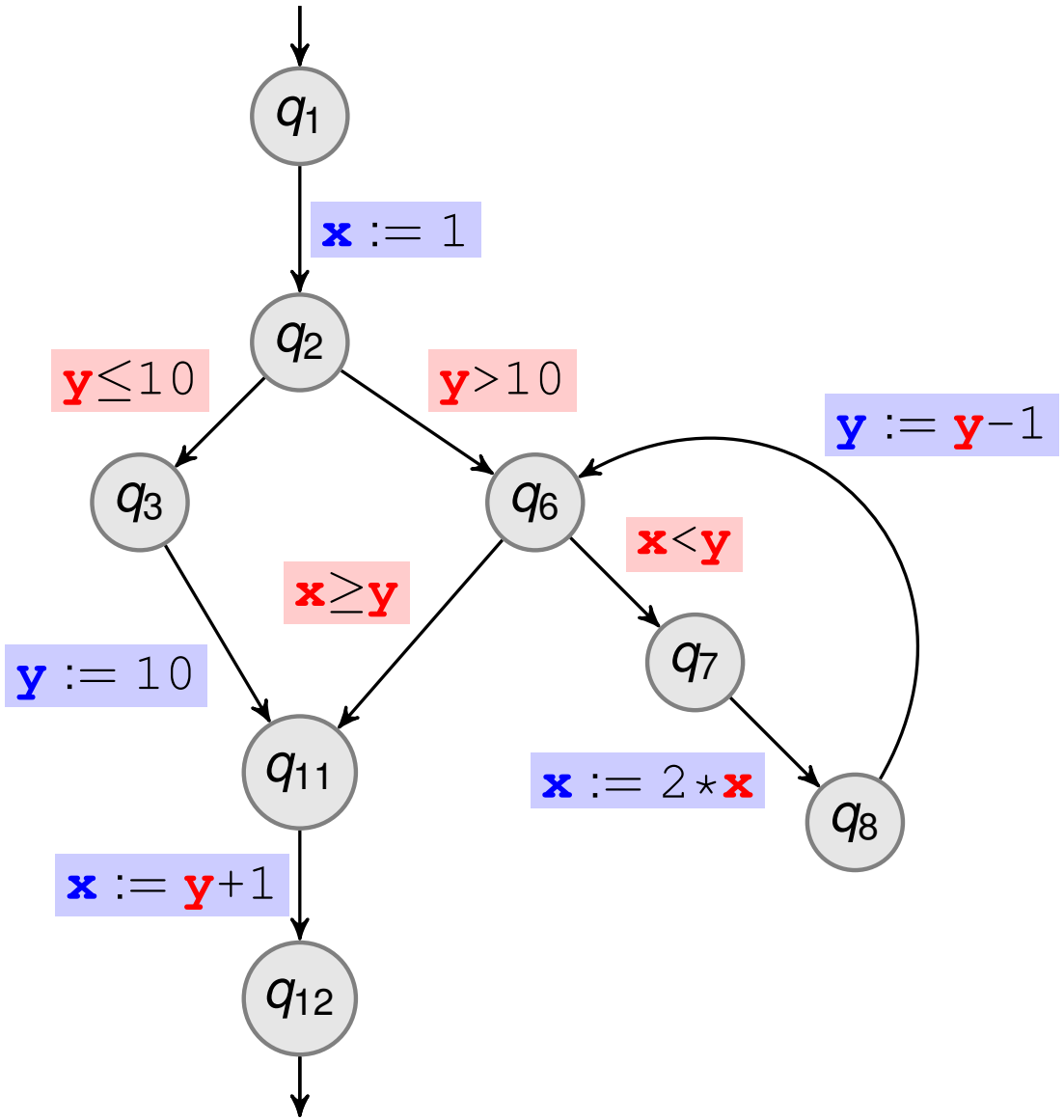
Live Variables Analysis: Running Example



0 : Initialization
1 : Local information
2 : Propagation (←)

	x	y
q_1		•
q_2	•	•
q_3		
q_6	•	•
q_7	•	•
q_8	•	•
q_{11}		•
q_{12}		

Live Variables Analysis: Running Example



0 : Initialization
1 : Local information
2 : Propagation (←)

	x	y
q_1		•
q_2	•	•
q_3		
q_6	•	•
q_7	•	•
q_8	•	•
q_{11}		•
q_{12}		

Live Variables Analysis: Formulation

Control Flow Automaton: $\langle Q, q_{in}, q_{out}, \mathrm{X}, \rightarrow \rangle$

System of equations: variables L_q for $q \in Q$, with $L_q \subseteq \mathrm{X}$

$$L_q = \bigcup_{q \xrightarrow{\mathrm{op}} q'} Gen_{\mathrm{op}} \cup (L_{q'} \setminus Kill_{\mathrm{op}}) \qquad L(q_{out}) = \emptyset$$

$$Gen_{\mathrm{op}} = \begin{cases} Var(g) & \text{if } \mathrm{op} = g \\ Var(e) & \text{if } \mathrm{op} = x := e \end{cases} \qquad Kill_{\mathrm{op}} = \begin{cases} \emptyset & \text{if } \mathrm{op} = g \\ \{x\} & \text{if } \mathrm{op} = x := e \end{cases}$$

$$f_{\mathrm{op}}(X) = Gen_{\mathrm{op}} \cup (X \setminus Kill_{\mathrm{op}})$$

Live Variables Analysis: Formulation

Control Flow Automaton: $\langle Q, q_{in}, q_{out}, \mathtt{X}, \rightarrow \rangle$

System of equations: variables L_q for $q \in Q$, with $L_q \subseteq \mathtt{X}$

$$L_q = \bigcup_{q \xrightarrow{\mathrm{op}} q'} f_{\mathrm{op}}\left(L_{q'}\right) \qquad L(q_{out}) = \emptyset$$

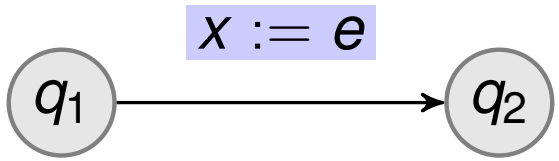
$$Gen_{\mathrm{op}} = \begin{cases} Var(g) & \text{if } \mathrm{op} = g \\ Var(e) & \text{if } \mathrm{op} = x := e \end{cases} \qquad Kill_{\mathrm{op}} = \begin{cases} \emptyset & \text{if } \mathrm{op} = g \\ \{x\} & \text{if } \mathrm{op} = x := e \end{cases}$$

$$f_{\mathrm{op}}(X) = Gen_{\mathrm{op}} \cup (X \setminus Kill_{\mathrm{op}})$$

Live Variables Analysis: Applications

Code Optimization

Dead code elimination



If x is not live at location q_2 then we may remove the assignment $x := e$ on the edge from q_1 to q_2.

This is sound since the analysis is conservative

Available Expressions Analysis: Definition

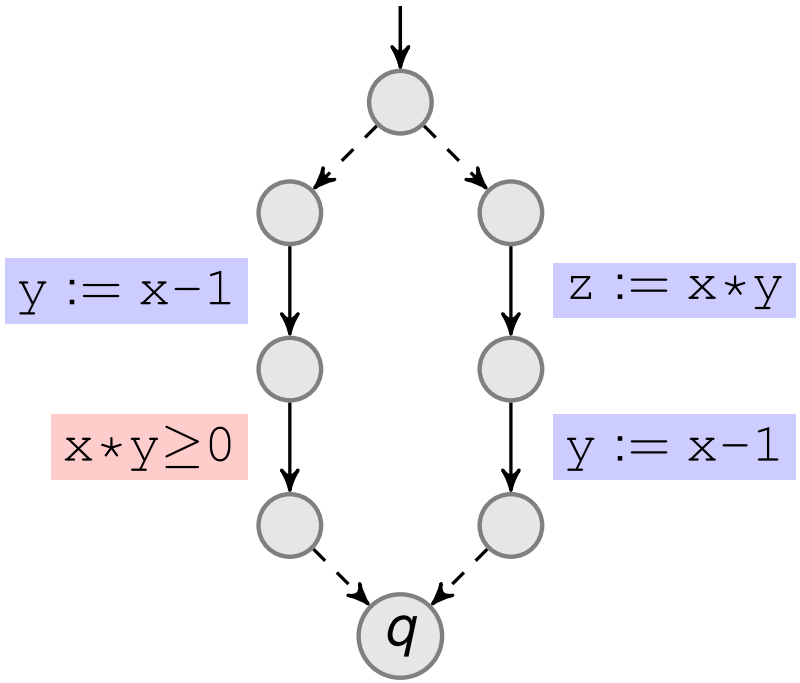
Definition

A expression e is available at location q if every control path from q_{in} to q contains an evaluation of e which is not followed by an assignment of any variable x occurring in e.

Available Expressions Analysis: Definition

Definition

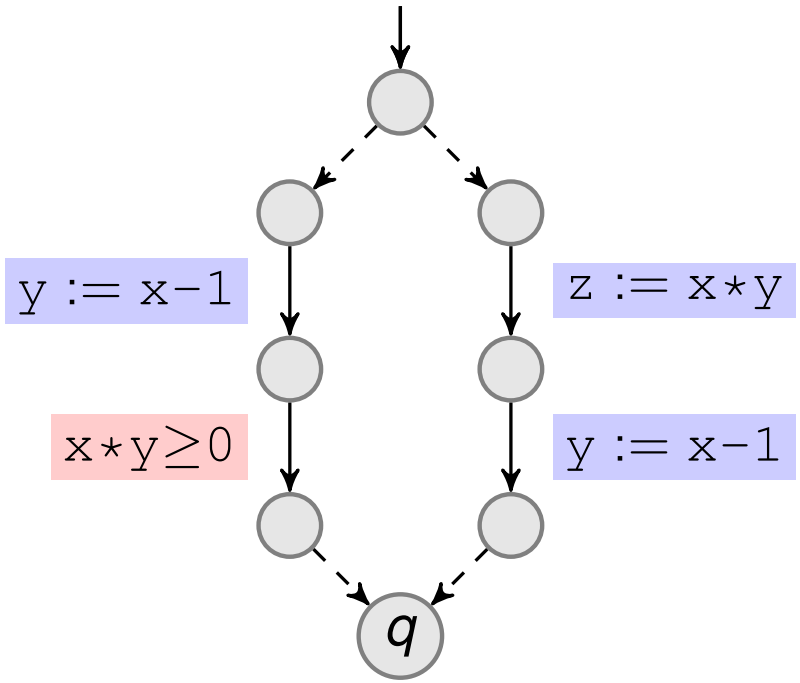
A expression e is available at location q if every control path from q_{in} to q contains an evaluation of e which is not followed by an assignment of any variable x occurring in e.



Available Expressions Analysis: Definition

Definition

A expression e is available at location q if every control path from q_{in} to q contains an evaluation of e which is not followed by an assignment of any variable x occurring in e.

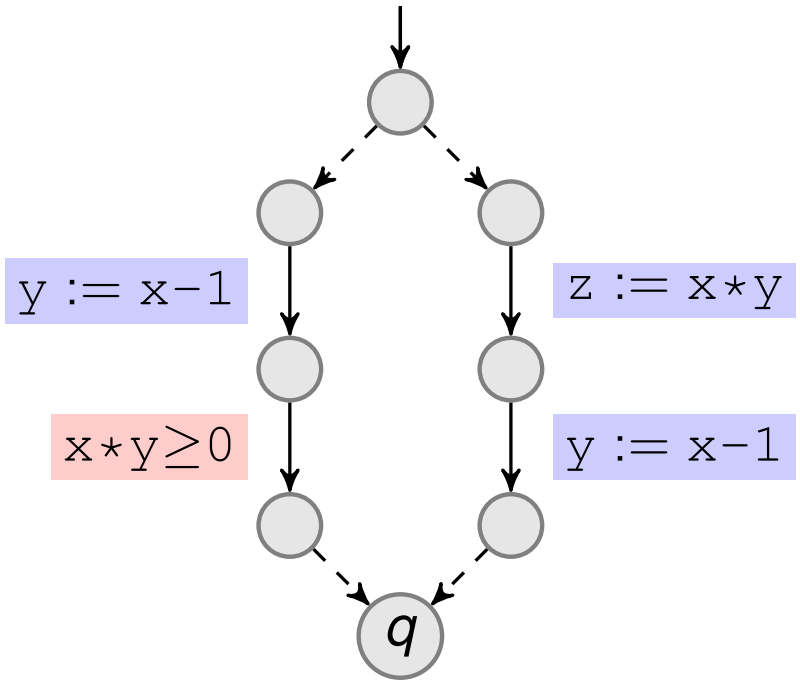


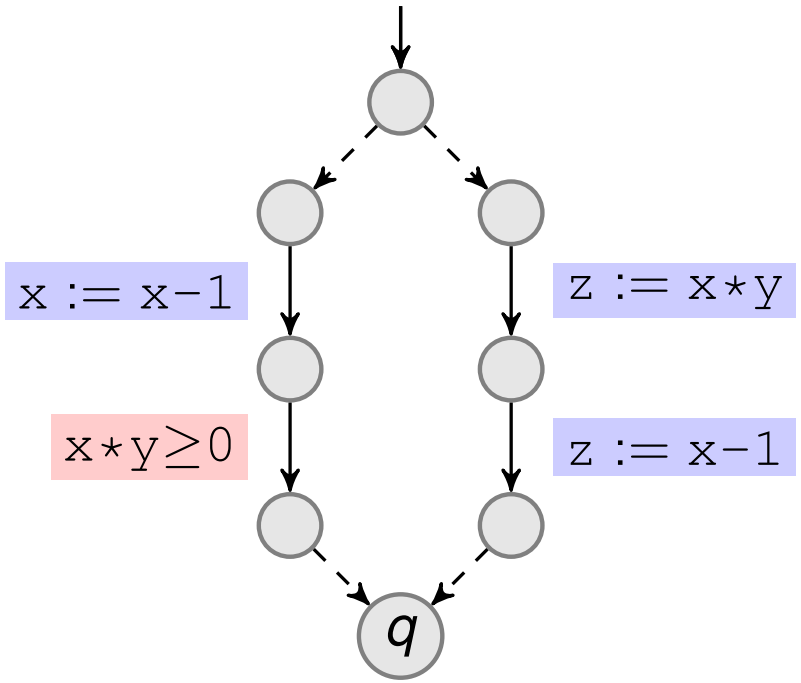
`x-1` available, `x*y` not available

Available Expressions Analysis: Definition

Definition

A expression e is available at location q if every control path from q_{in} to q contains an evaluation of e which is not followed by an assignment of any variable x occurring in e.



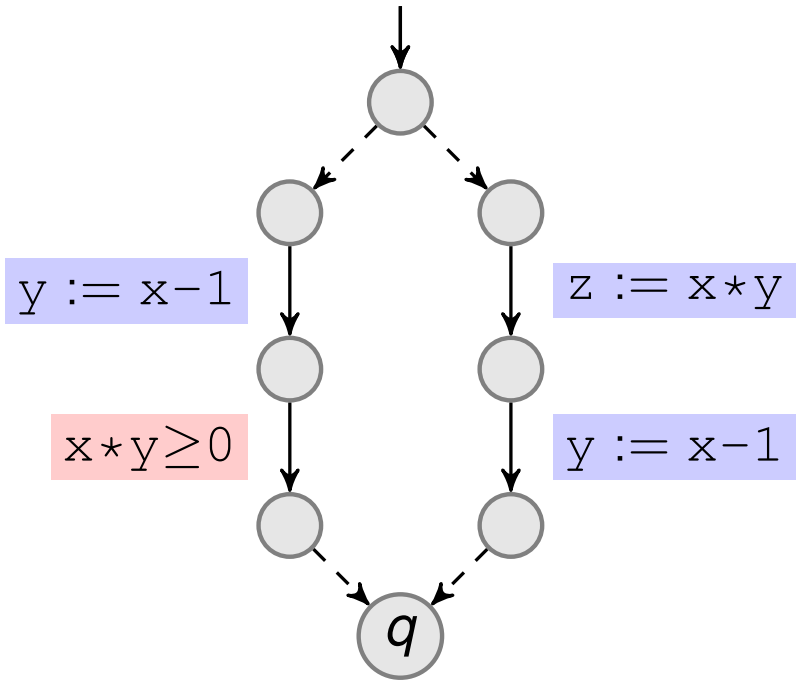


`x-1` available, `x*y` not available

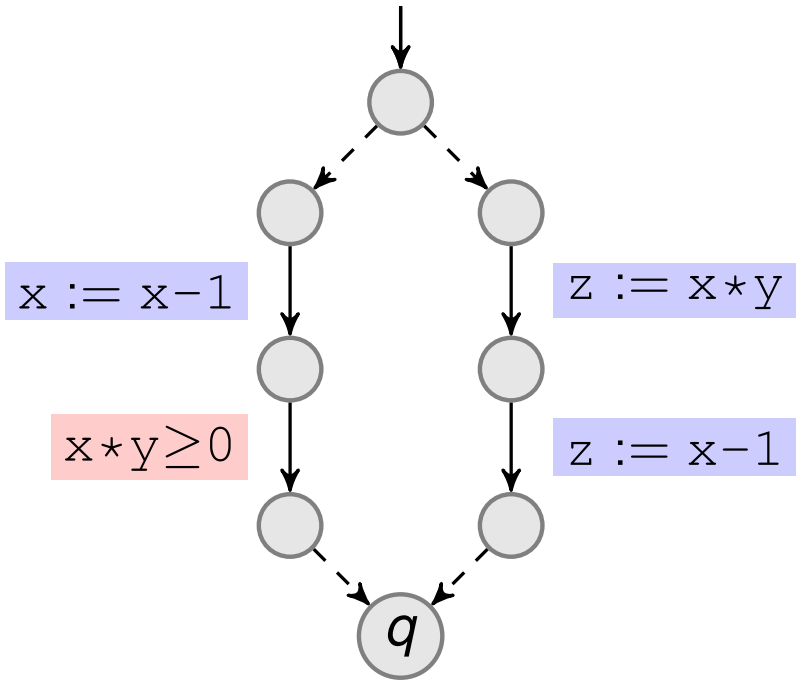
Available Expressions Analysis: Definition

Definition

A expression e is available at location q if every control path from q_{in} to q contains an evaluation of e which is not followed by an assignment of any variable x occurring in e.

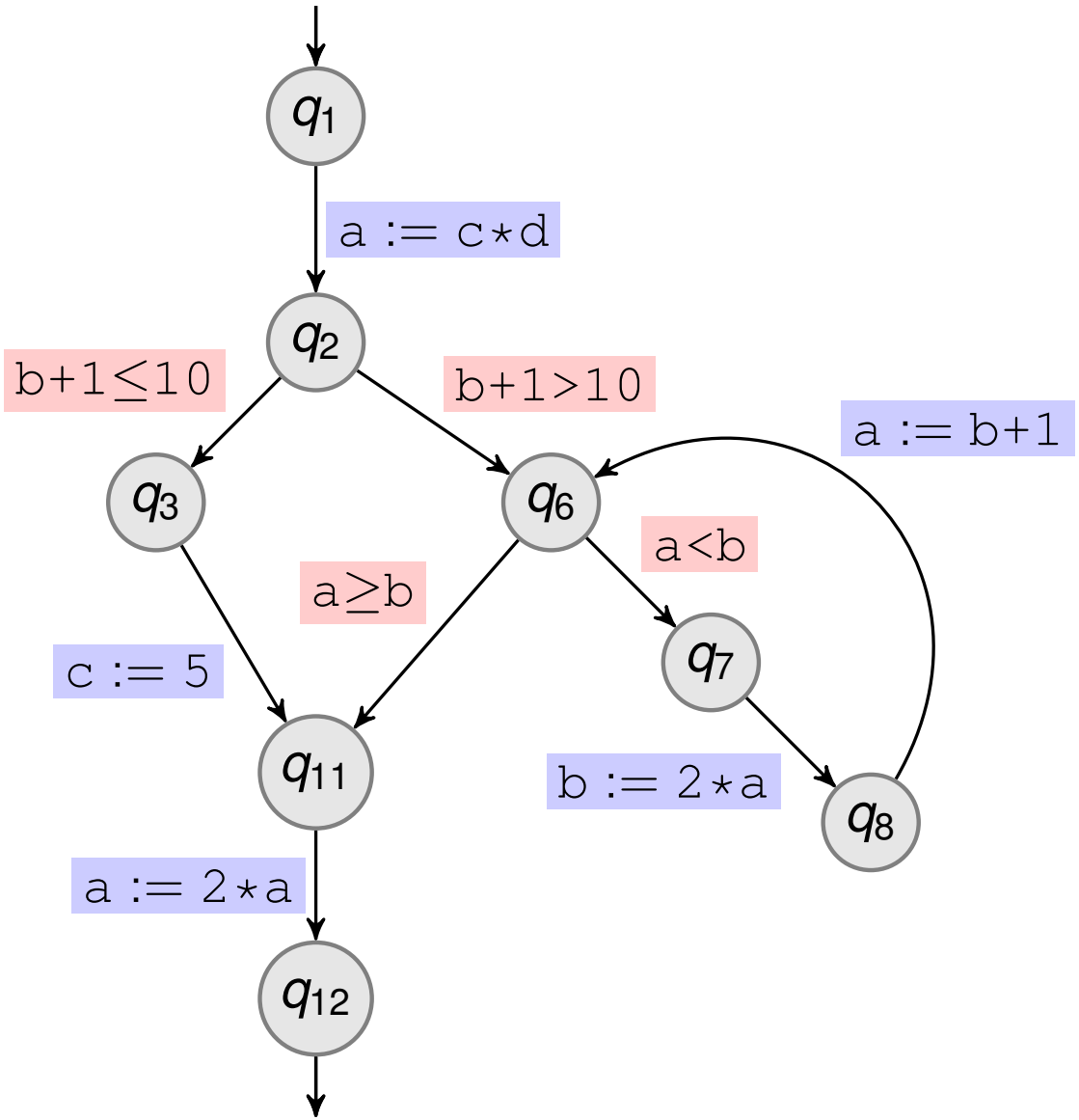


`x-1` available, `x*y` not available

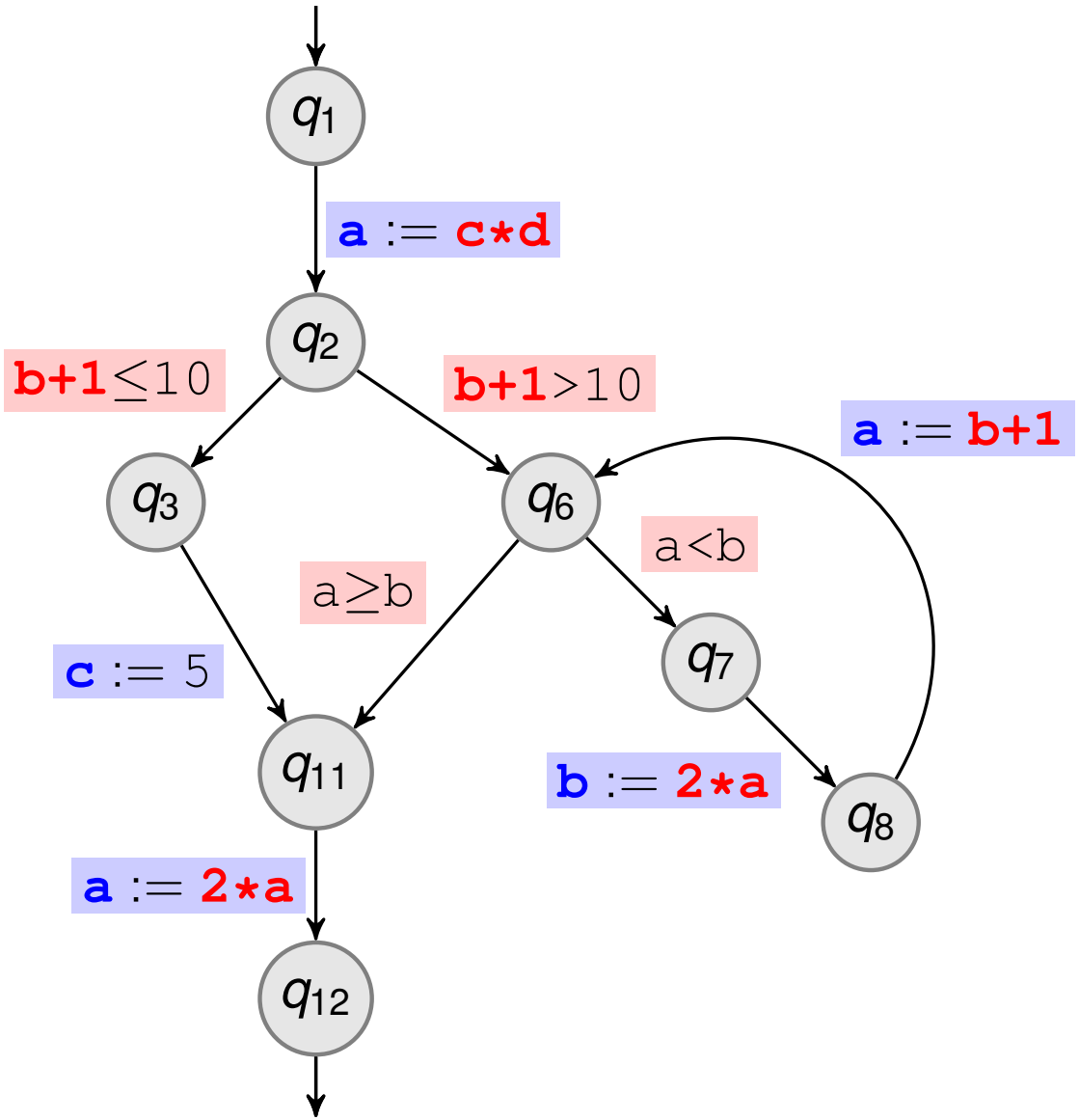


`x-1` not available, `x*y` available

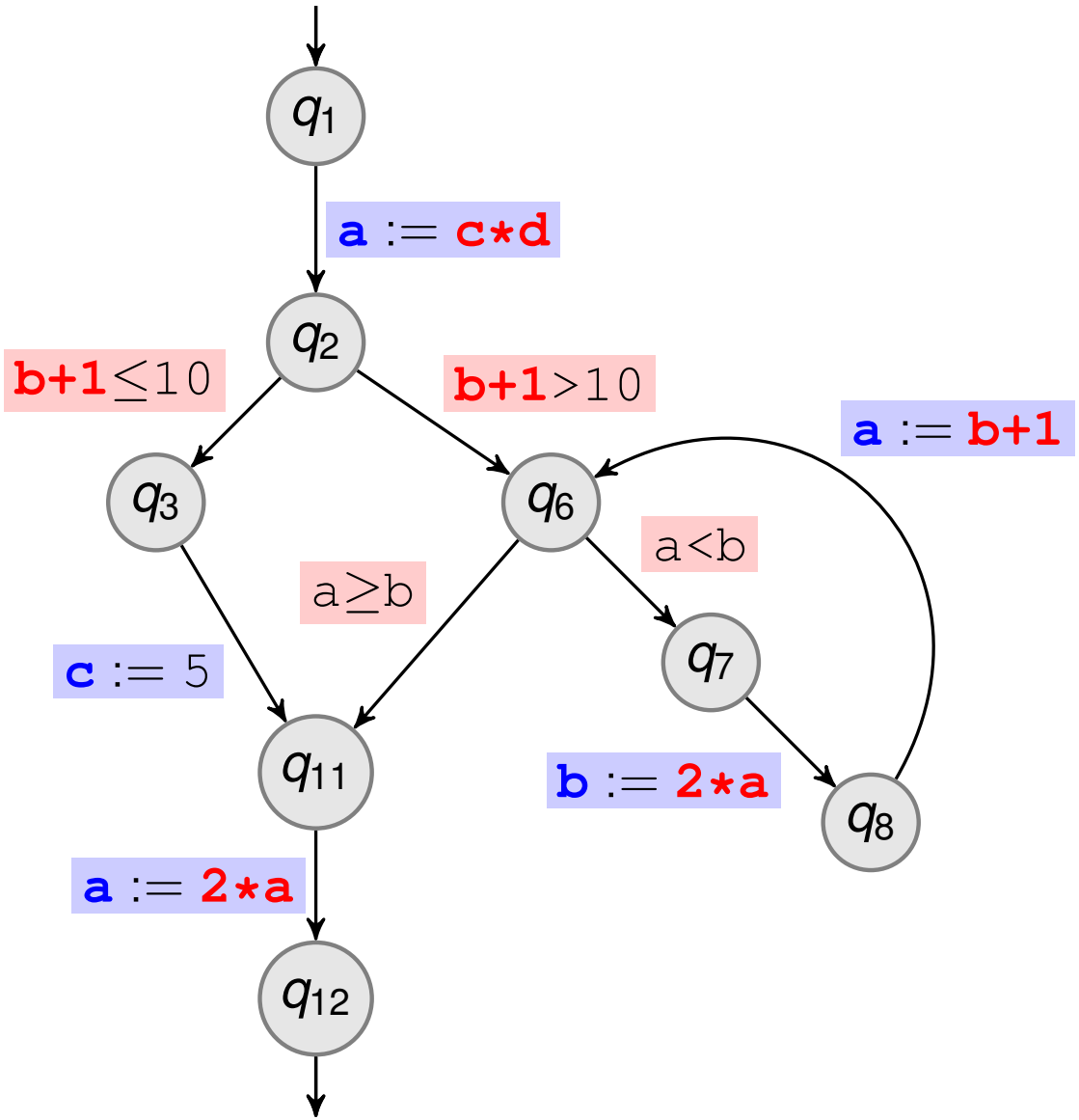
Available Expressions Analysis: Other Example



Available Expressions Analysis: Other Example



Available Expressions Analysis: Other Example



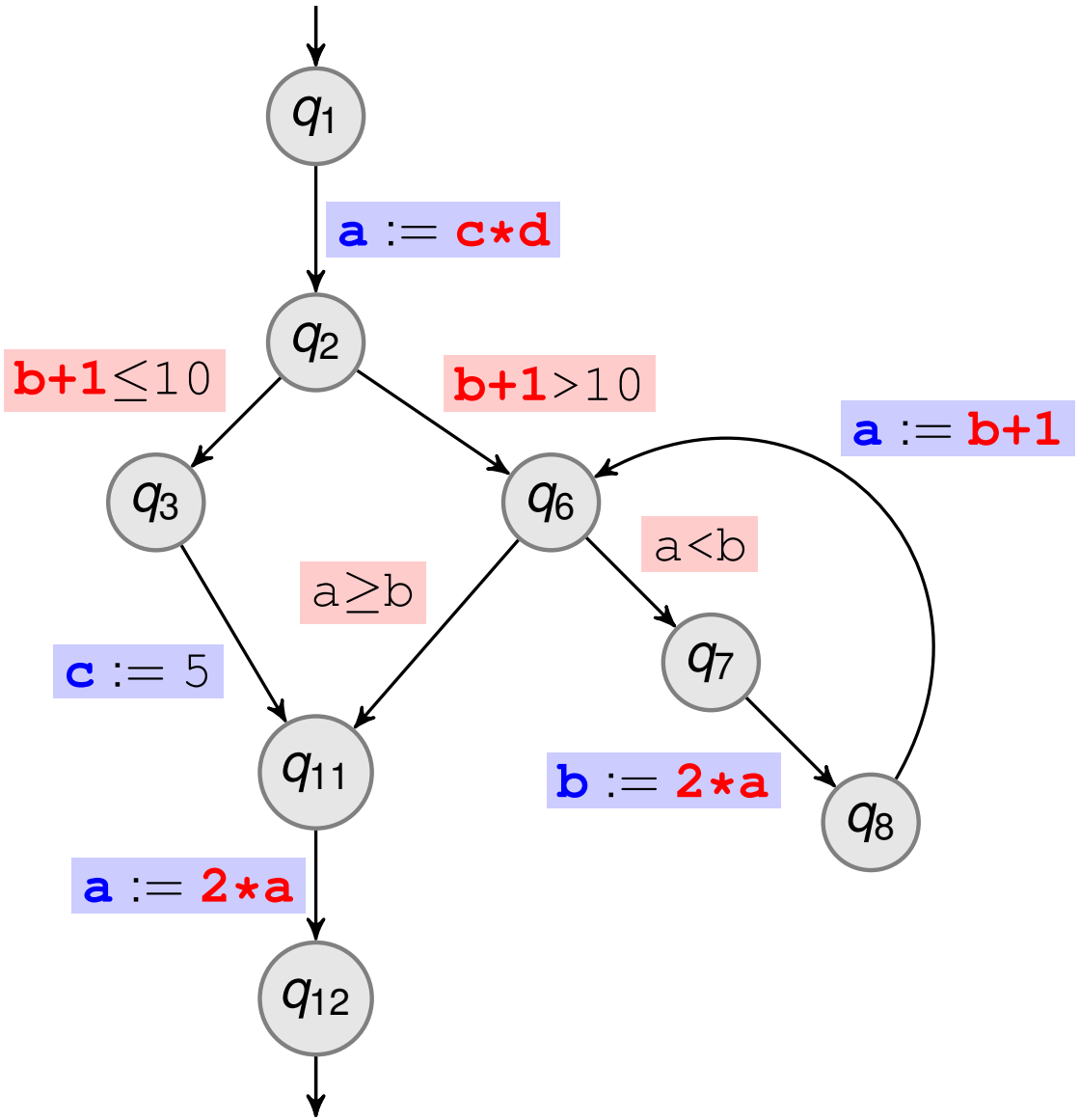
0 : Initialization

1 : Local information

2 : Propagation ($\rightarrow$)

	c*d	b+1	2*a
q_1			
q_2	•	•	•
q_3	•	•	•
q_6	•	•	•
q_7	•	•	•
q_8	•	•	•
q_{11}	•	•	•
q_{12}	•	•	•

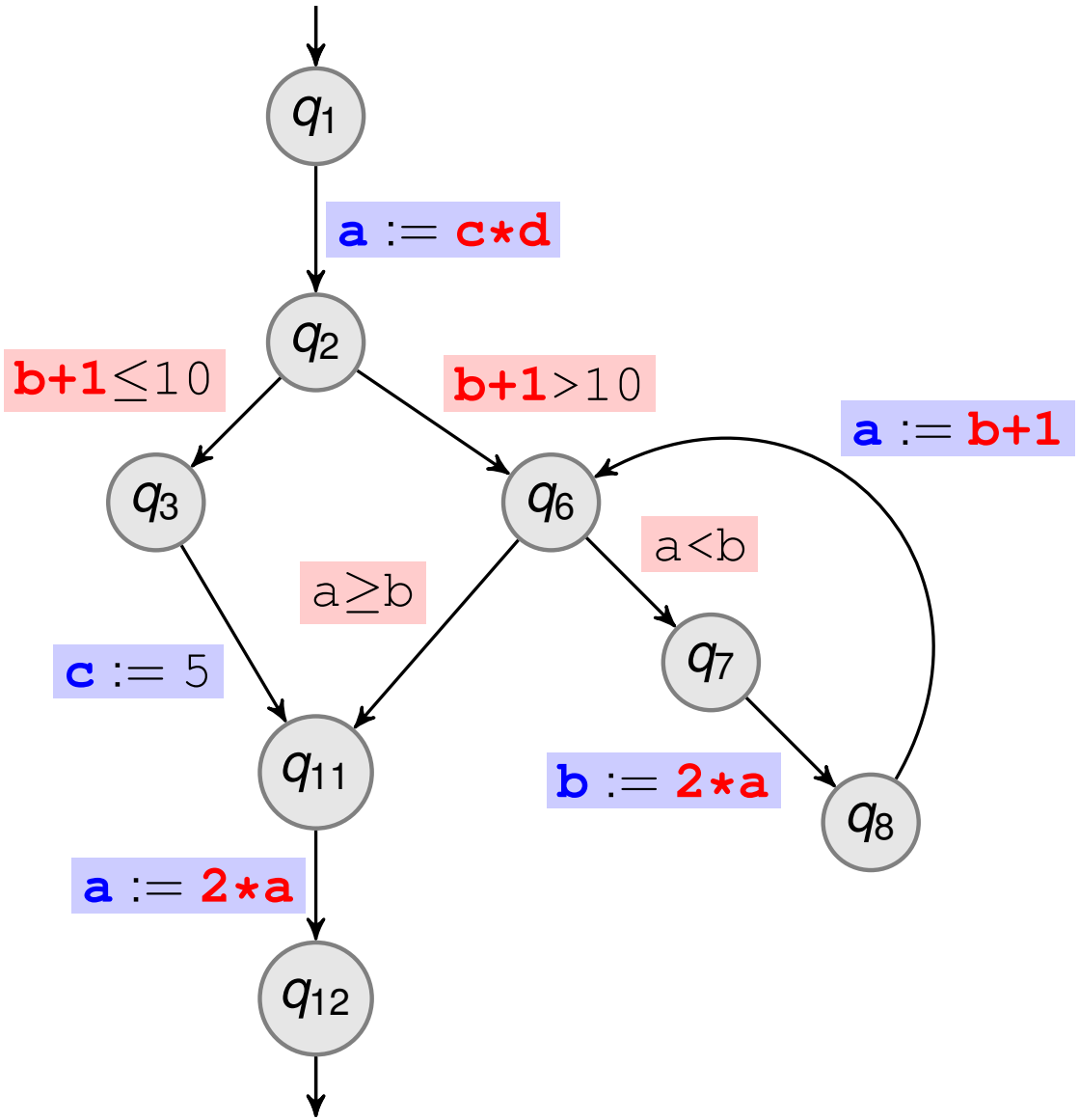
Available Expressions Analysis: Other Example



0 : Initialization
1 : Local information
2 : Propagation ($\rightarrow$)

	c*d	b+1	2*a
q_1			
q_2	•	•	
q_3	•	•	•
q_6	•	•	
q_7	•	•	•
q_8	•		•
q_{11}		•	•
q_{12}	•	•	

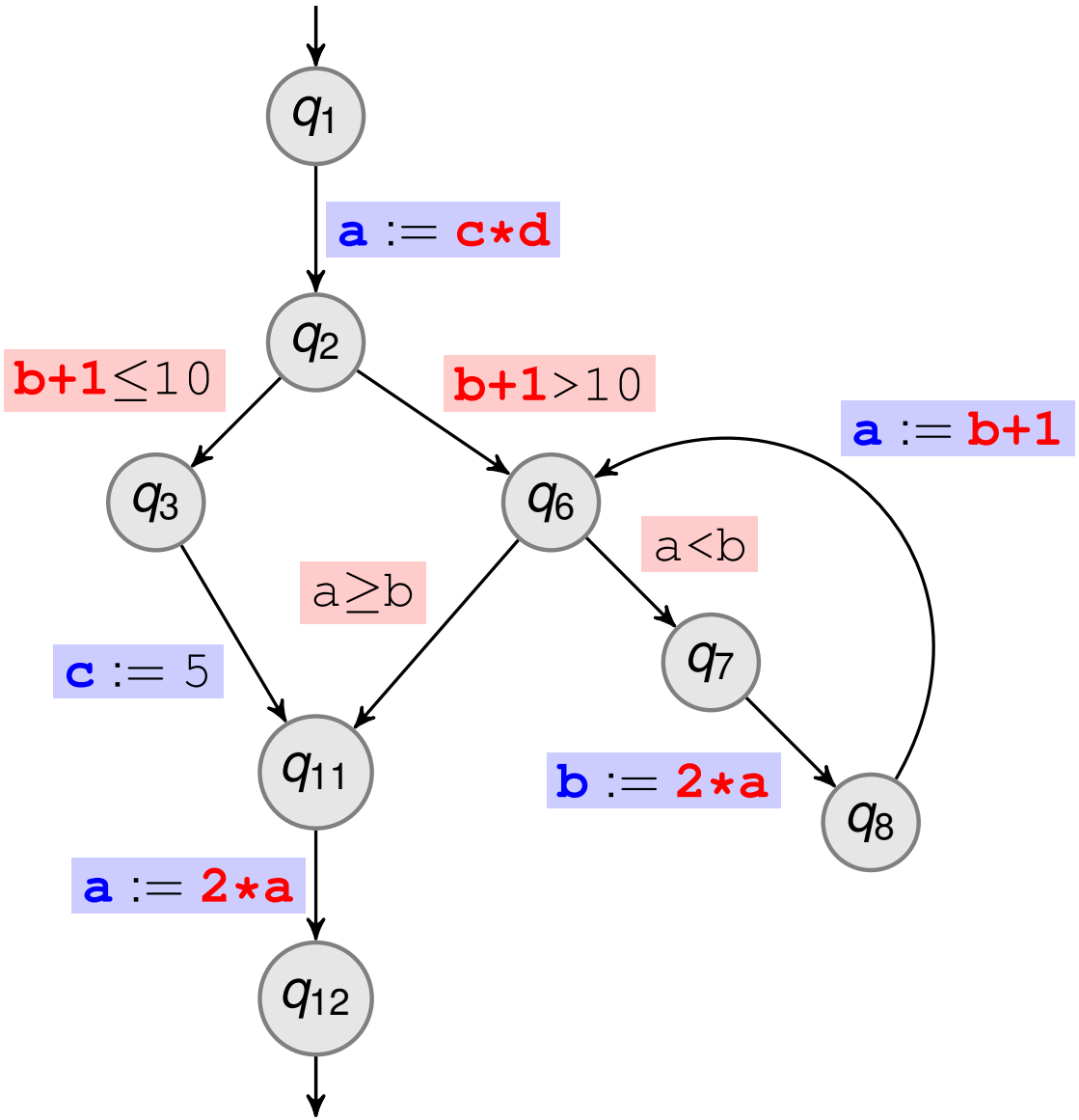
Available Expressions Analysis: Other Example



0 : Initialization
1 : Local information
2 : Propagation ($\rightarrow$)

	c*d	b+1	2*a
q_1			
q_2	•	•	
q_3	•	•	•
q_6	•	•	
q_7	•	•	•
q_8	•		•
q_{11}		•	•
q_{12}	•	•	

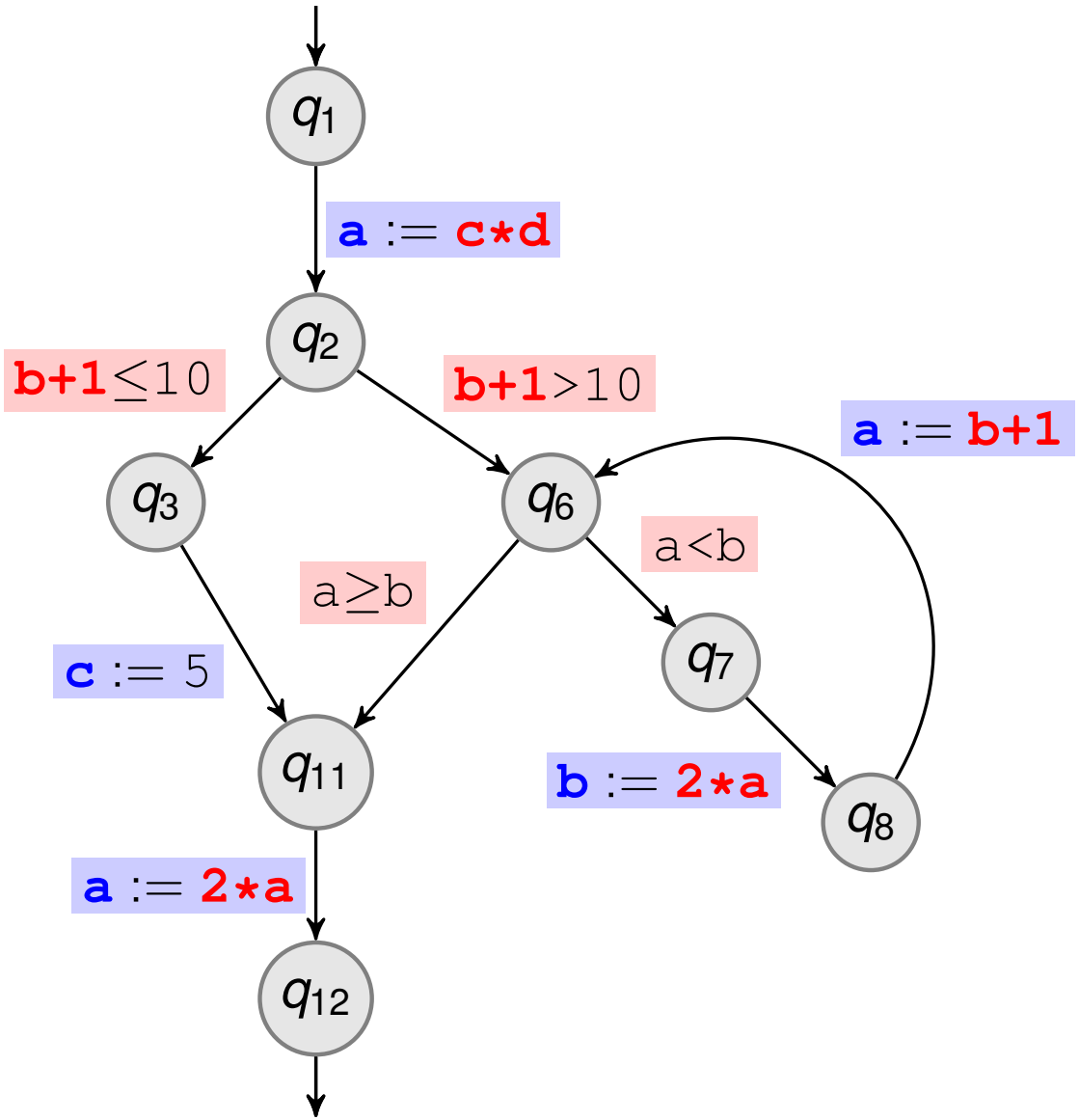
Available Expressions Analysis: Other Example



0 : Initialization
1 : Local information
2 : Propagation ($\rightarrow$)

	c*d	b+1	2*a
q_1			
q_2	•		
q_3	•	•	•
q_6	•	•	
q_7	•	•	•
q_8	•		•
q_{11}		•	•
q_{12}	•	•	

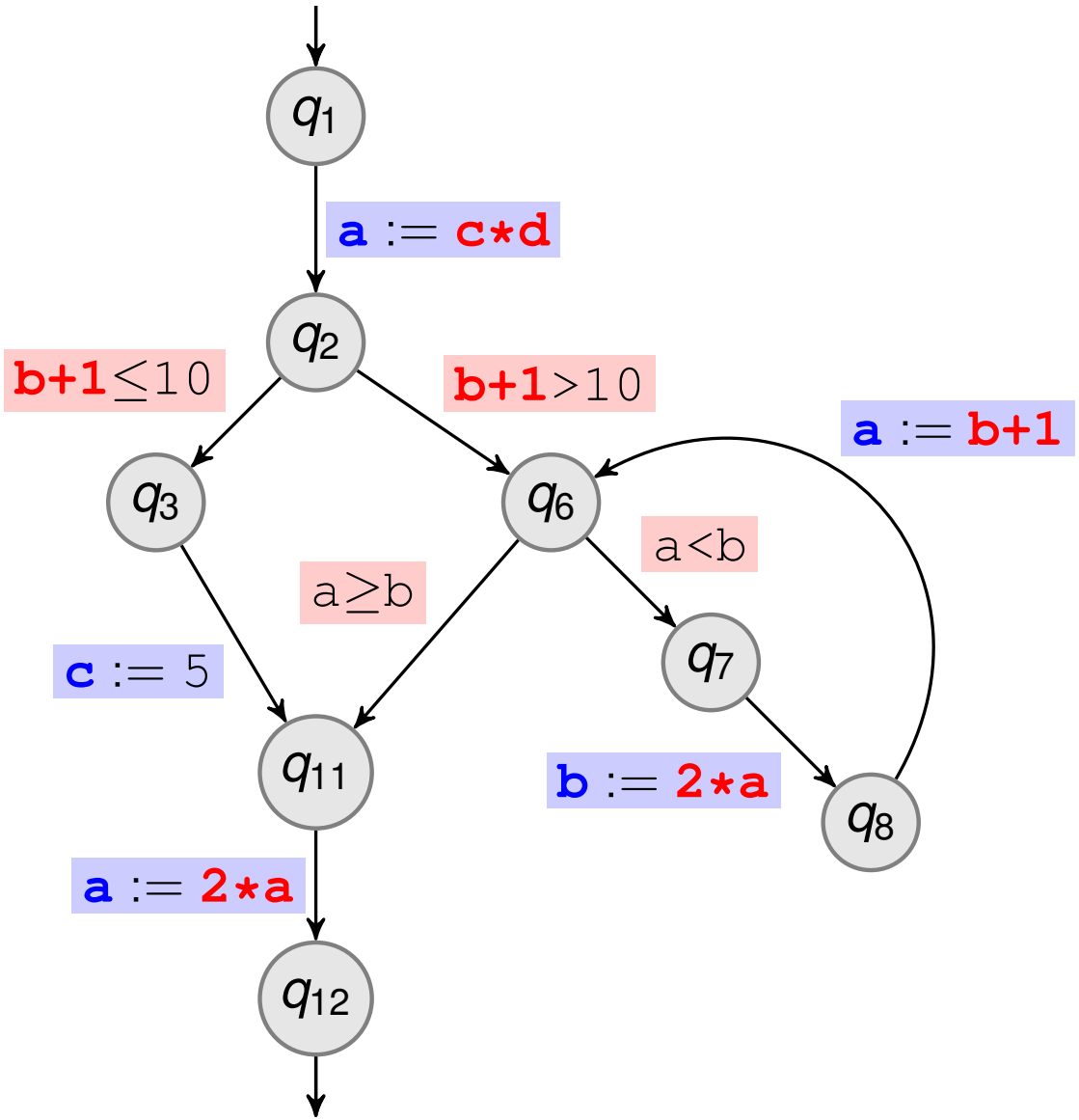
Available Expressions Analysis: Other Example



0 : Initialization
1 : Local information
2 : Propagation ($\rightarrow$)

	c*d	b+1	2*a
q_1			
q_2	•		
q_3	•	•	•
q_6	•	•	
q_7	•	•	•
q_8	•		•
q_{11}		•	•
q_{12}	•	•	

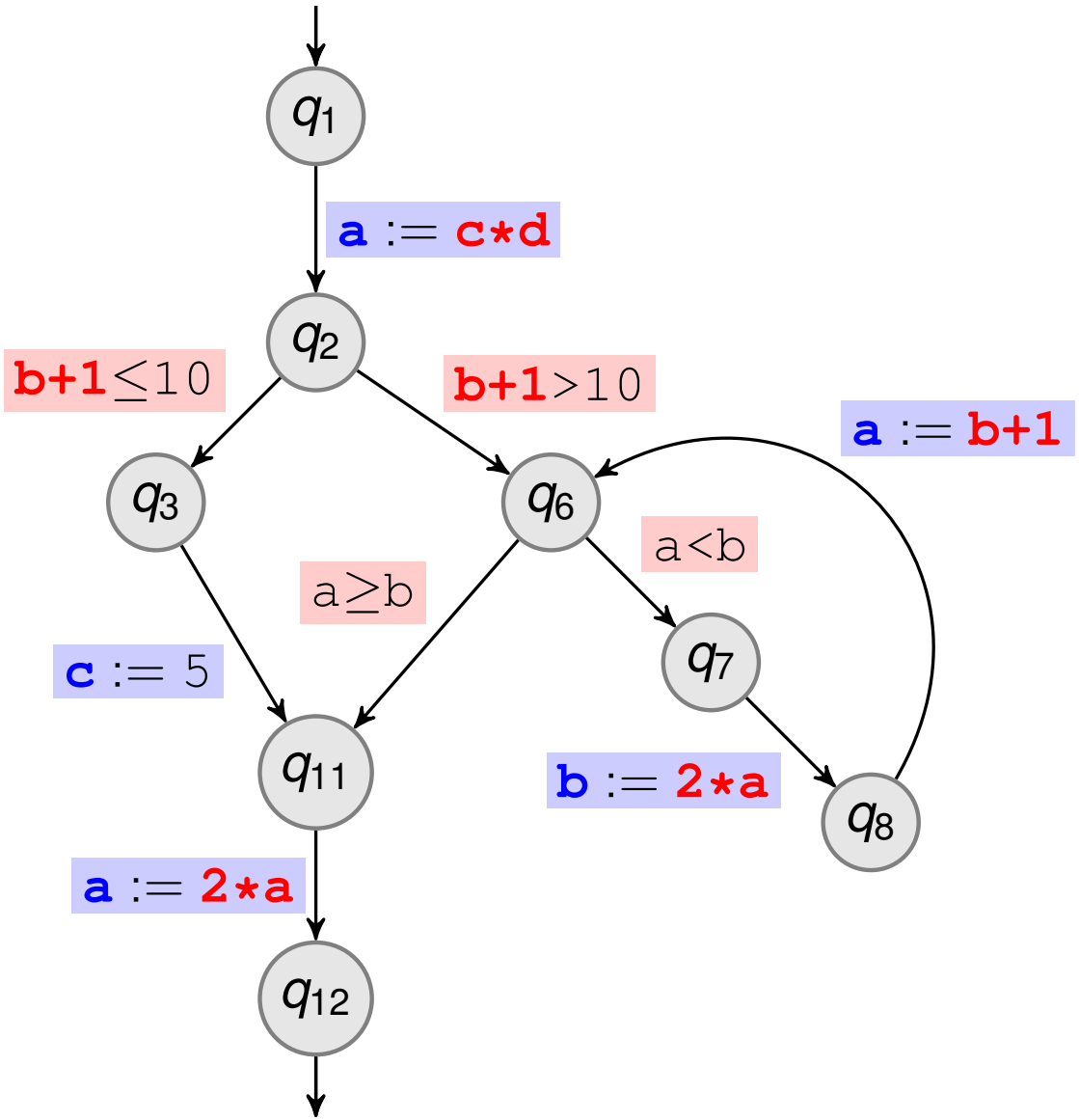
Available Expressions Analysis: Other Example



0 : Initialization
1 : Local information
2 : Propagation ($\rightarrow$)

	c*d	b+1	2*a
q_1			
q_2	•		
q_3	•	•	
q_6	•	•	
q_7	•	•	•
q_8	•		•
q_{11}		•	•
q_{12}	•	•	

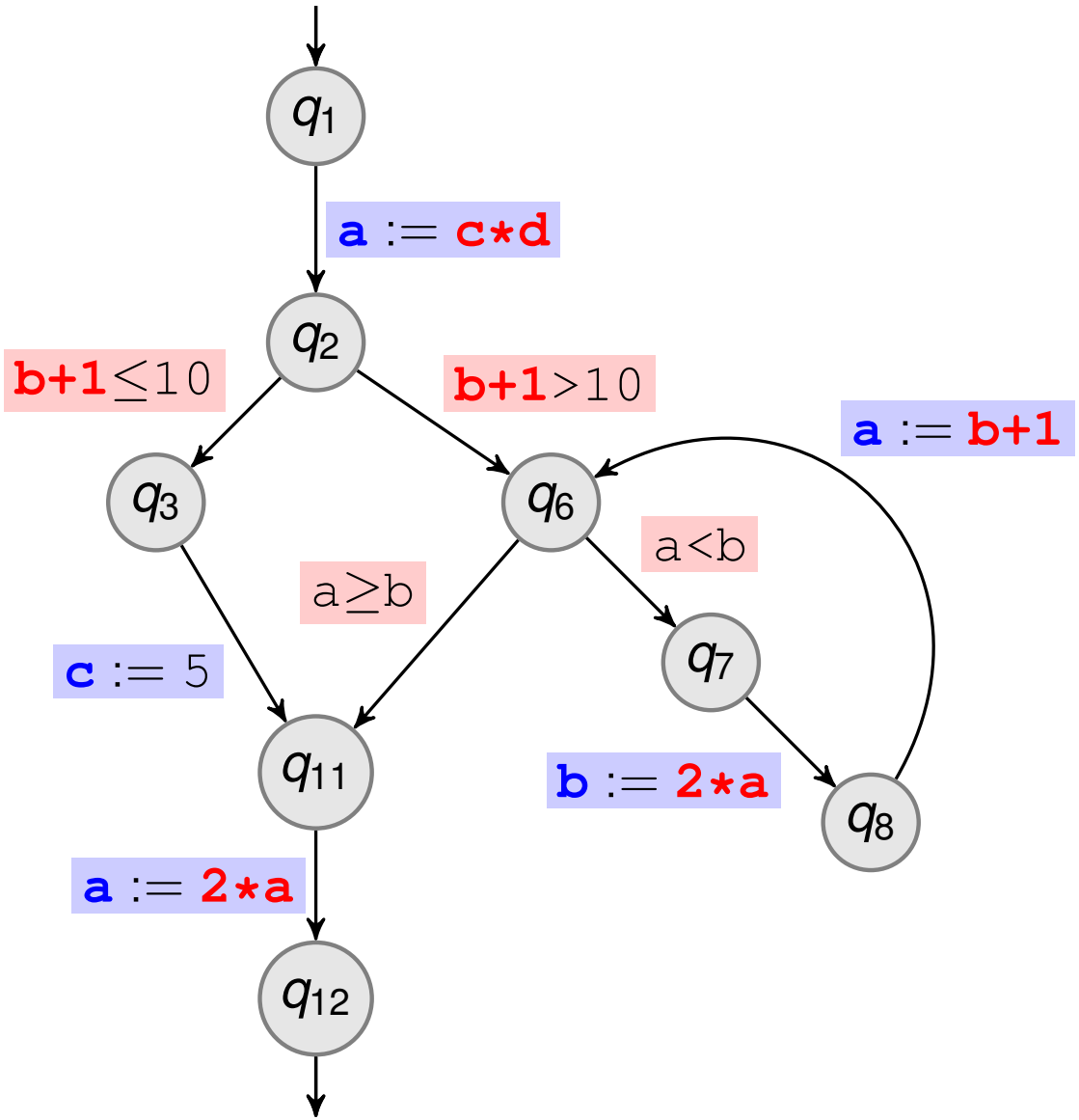
Available Expressions Analysis: Other Example



0 : Initialization
1 : Local information
2 : Propagation ($\rightarrow$)

	c*d	b+1	2*a
q_1			
q_2	•		
q_3	•	•	
q_6	•	•	
q_7	•	•	•
q_8	•		•
q_{11}		•	•
q_{12}	•	•	

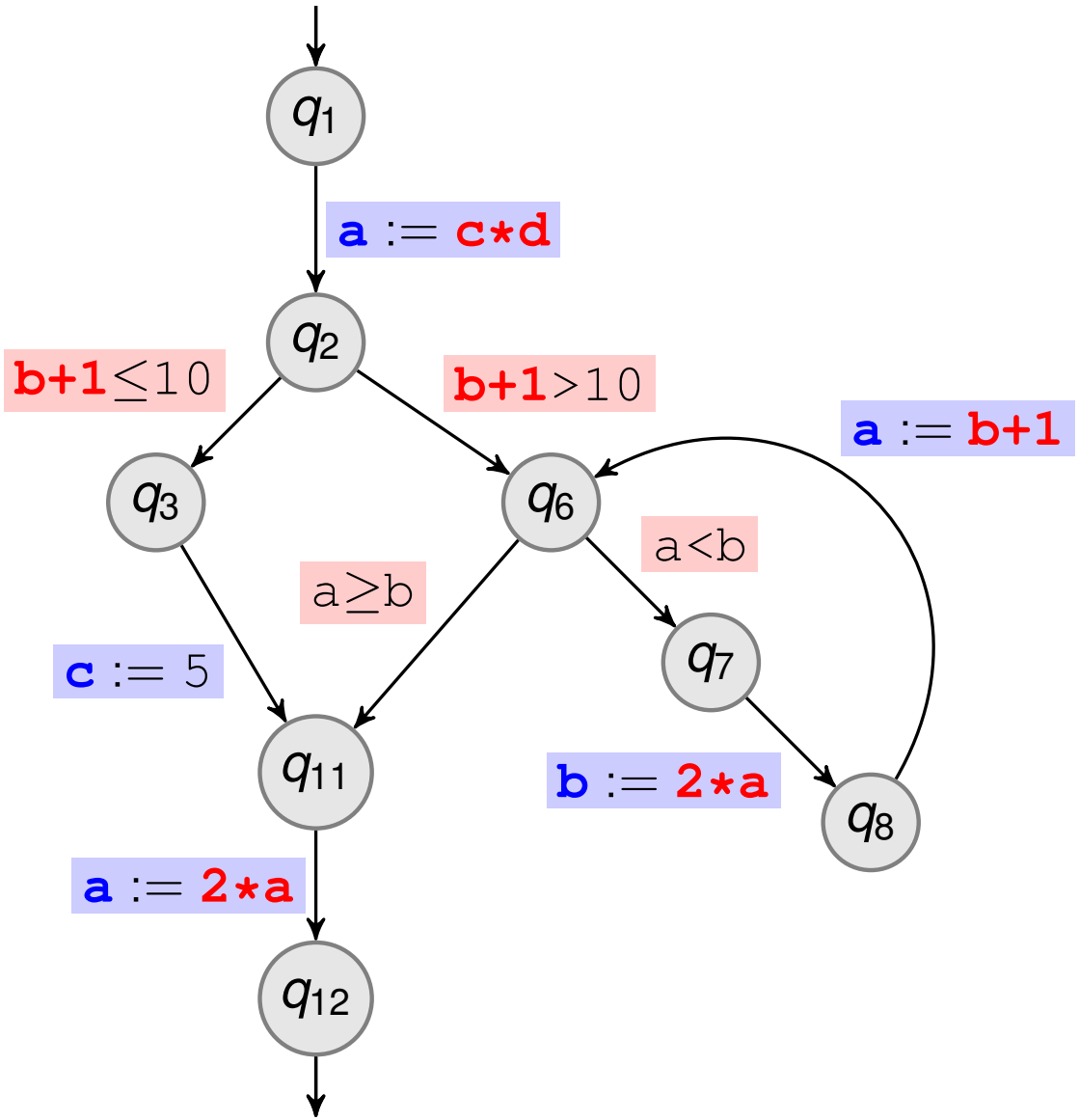
Available Expressions Analysis: Other Example



0 : Initialization
1 : Local information
2 : Propagation ($\rightarrow$)

	c*d	b+1	2*a
q_1			
q_2	•		
q_3	•	•	
q_6	•	•	
q_7	•	•	•
q_8	•		•
q_{11}		•	
q_{12}	•	•	

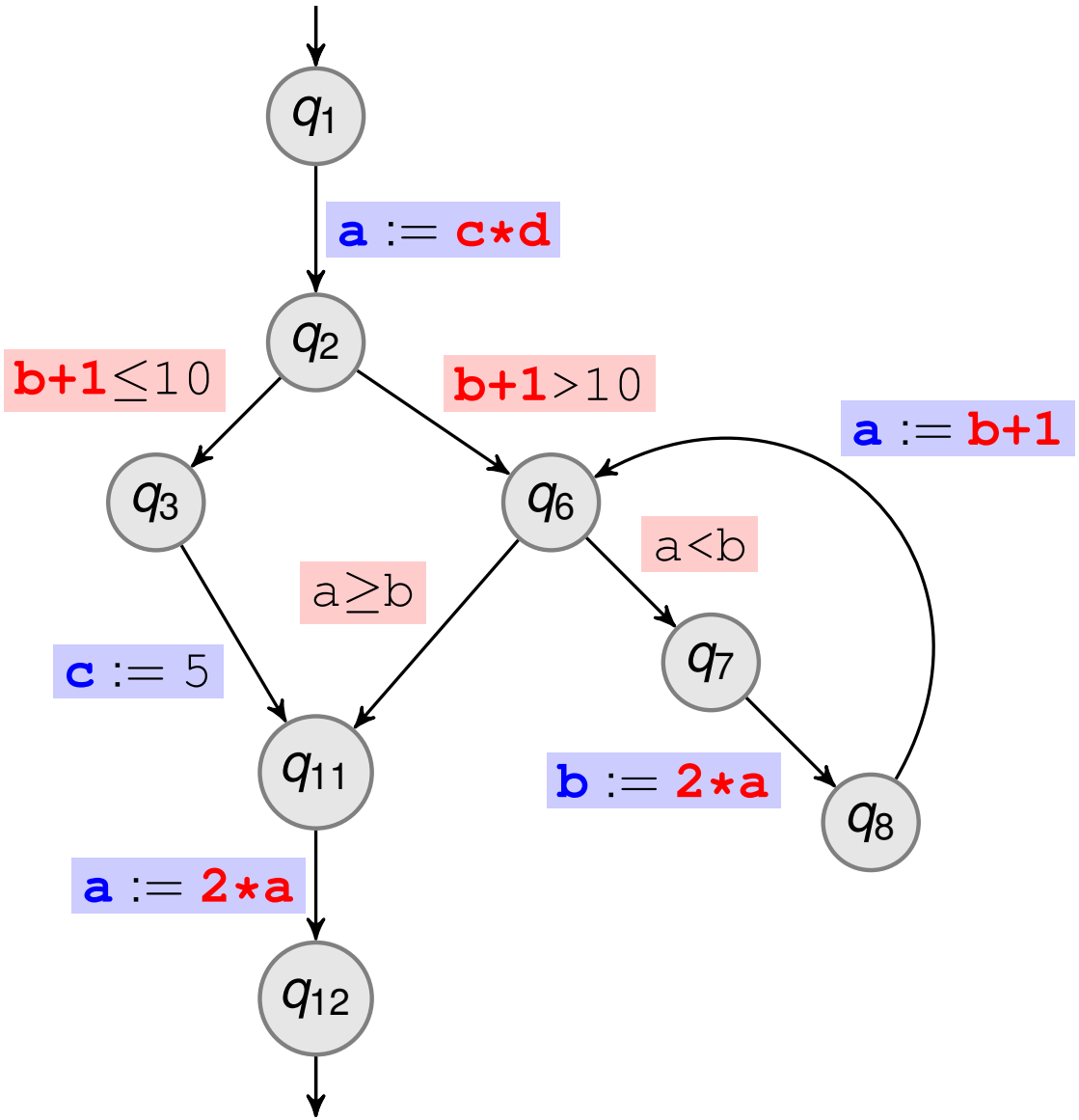
Available Expressions Analysis: Other Example



0 : Initialization
1 : Local information
2 : Propagation ($\rightarrow$)

	c*d	b+1	2*a
q_1			
q_2	•		
q_3	•	•	
q_6	•	•	
q_7	•	•	•
q_8	•		•
q_{11}		•	
q_{12}	•	•	

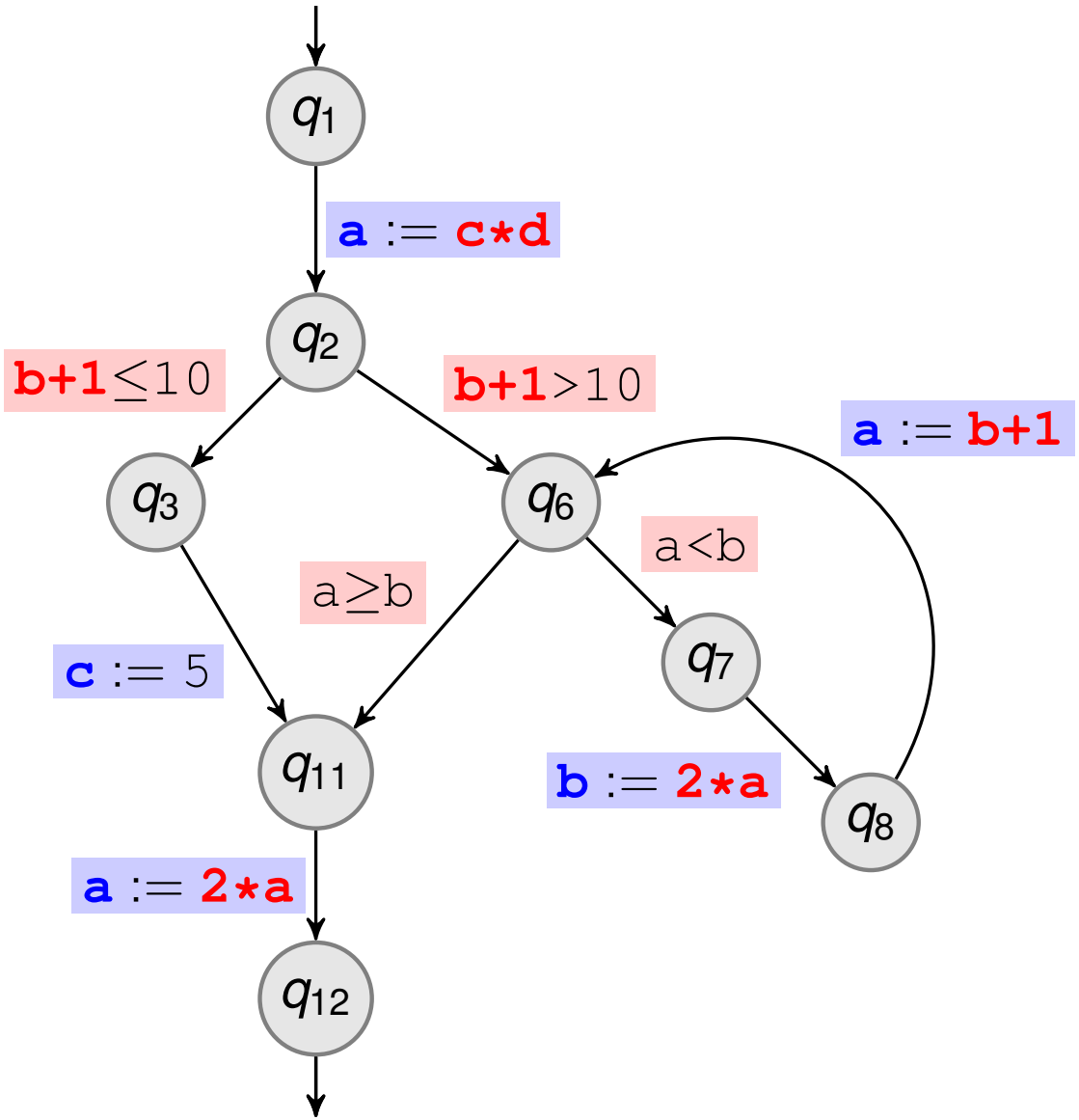
Available Expressions Analysis: Other Example



0 : Initialization
1 : Local information
2 : Propagation ($\rightarrow$)

	c*d	b+1	2*a
q_1			
q_2	•		
q_3	•	•	
q_6	•	•	
q_7	•	•	•
q_8	•		•
q_{11}		•	
q_{12}		•	

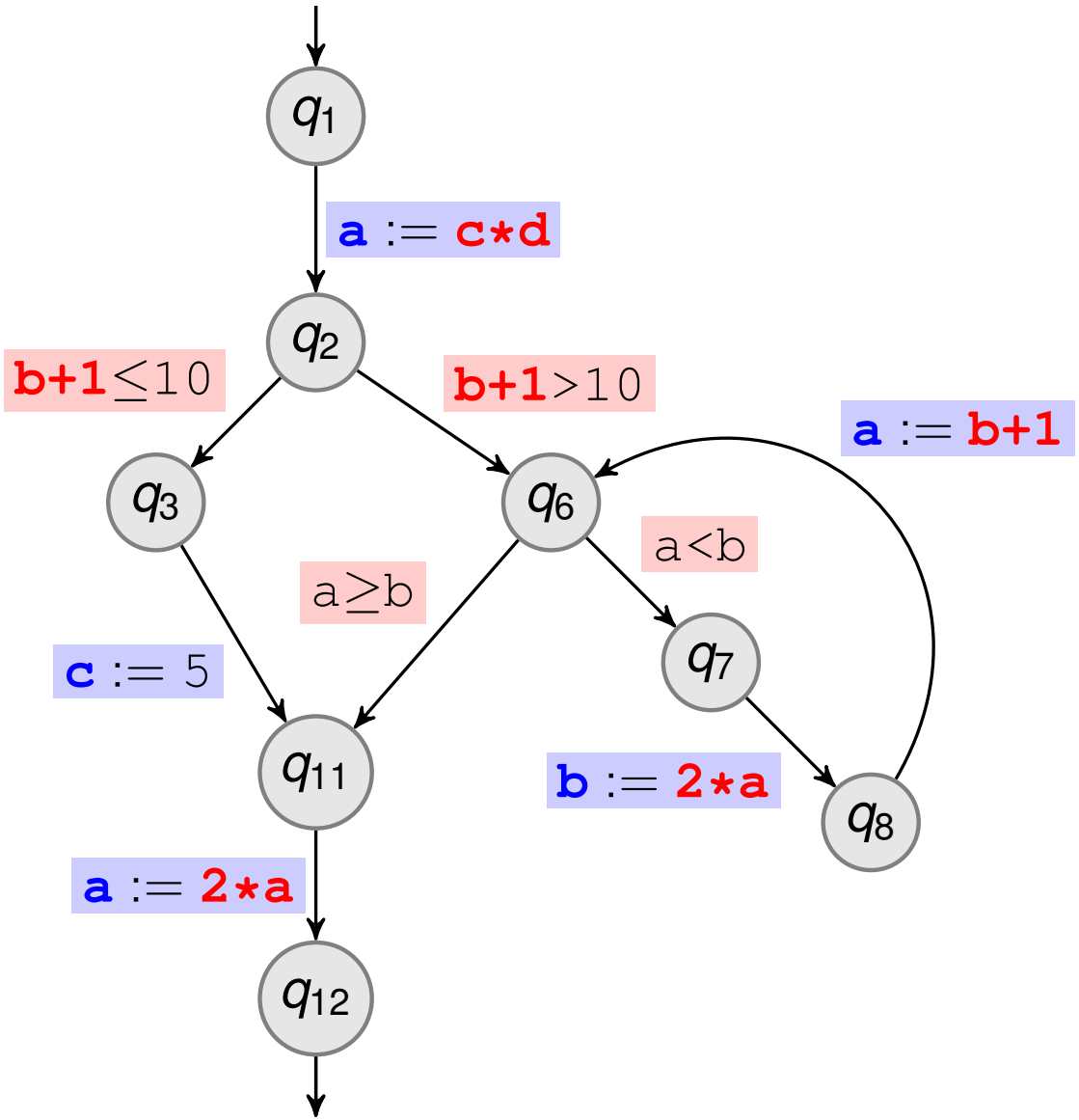
Available Expressions Analysis: Other Example



0 : Initialization
1 : Local information
2 : Propagation ($\rightarrow$)

	c*d	b+1	2*a
q_1			
q_2	•		
q_3	•	•	
q_6	•	•	
q_7	•	•	•
q_8	•		•
q_{11}		•	
q_{12}		•	

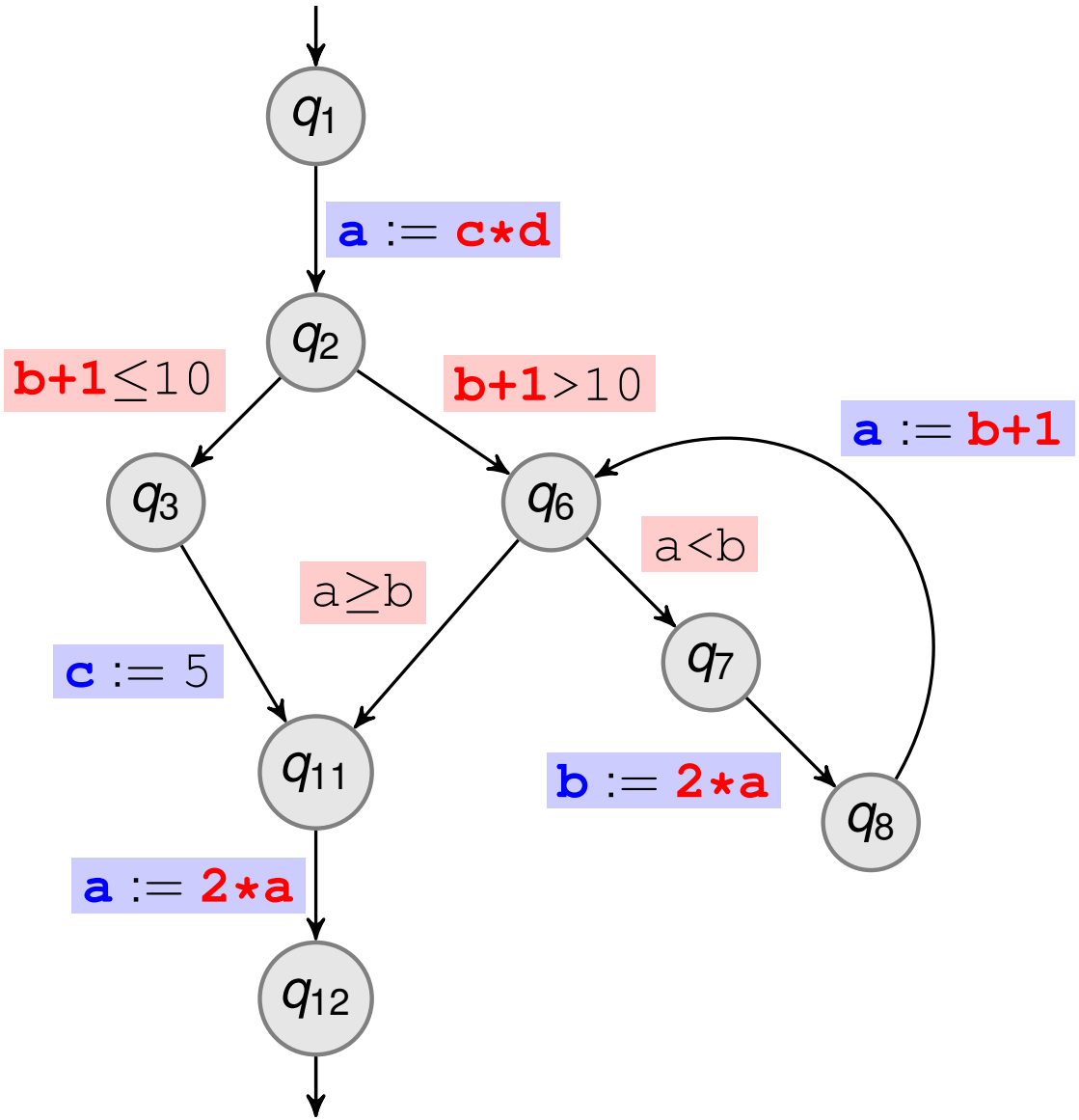
Available Expressions Analysis: Other Example



0 : Initialization
1 : Local information
2 : Propagation ($\rightarrow$)

	c*d	b+1	2*a
q_1			
q_2	•		
q_3	•	•	
q_6	•	•	
q_7	•	•	
q_8	•		•
q_{11}		•	
q_{12}		•	

Available Expressions Analysis: Other Example



0 : Initialization
1 : Local information
2 : Propagation ($\rightarrow$)

	c*d	b+1	2*a
q_1			
q_2	•		
q_3	•	•	
q_6	•	•	
q_7	•	•	
q_8	•		•
q_{11}		•	
q_{12}		•	

Available Expressions Analysis: Formulation

Control Flow Automaton: $\langle Q, q_{in}, q_{out}, \mathtt{X}, \rightarrow \rangle$

System of equations: variables A_q, with $A_q \subseteq SubExp(\rightarrow)$

$$A_q = \bigcap_{q' \xrightarrow{\mathrm{op}} q} Gen_{\mathrm{op}} \cup (A_{q'} \setminus Kill_{\mathrm{op}}) \qquad A(q_{in}) = \emptyset$$

$$Gen_{\mathrm{op}} = \begin{cases} SubExp(g) & \text{if } \mathrm{op} = g \\ \{f \in SubExp(e) \mid x \notin SubExp(e)\} & \text{if } \mathrm{op} = x := e \end{cases}$$

$$Kill_{\mathrm{op}} = \begin{cases} \emptyset & \text{if } \mathrm{op} = g \\ \{e \in SubExp(\rightarrow) \mid x \in Var(e)\} & \text{if } \mathrm{op} = x := e \end{cases}$$

$$f_{\mathrm{op}}(X) = Gen_{\mathrm{op}} \cup (X \setminus Kill_{\mathrm{op}})$$

Available Expressions Analysis: Formulation

Control Flow Automaton: $\langle Q, q_{in}, q_{out}, \mathrm{X}, \rightarrow \rangle$

System of equations: variables A_q, with $A_q \subseteq SubExp(\rightarrow)$

$$A_q = \bigcap_{q' \xrightarrow{\mathrm{op}} q} f_{\mathrm{op}}\left(A_{q'}\right) \qquad A(q_{in}) = \emptyset$$

$$Gen_{\mathrm{op}} = \begin{cases} SubExp(g) & \text{if } \mathrm{op} = g \\ \{f \in SubExp(e) \mid x \notin SubExp(e)\} & \text{if } \mathrm{op} = x := e \end{cases}$$

$$Kill_{\mathrm{op}} = \begin{cases} \emptyset & \text{if } \mathrm{op} = g \\ \{e \in SubExp(\rightarrow) \mid x \in Var(e)\} & \text{if } \mathrm{op} = x := e \end{cases}$$

$$f_{\mathrm{op}}(X) = Gen_{\mathrm{op}} \cup (X \setminus Kill_{\mathrm{op}})$$

Available Expressions Analysis: Applications

Code Optimization

Avoid recomputation of an expression

If e is available at location q_1 then we may reuse its value to evaluate the operation on the edge from q_1 to q_2.

This is sound since the analysis is conservative

Constant Propagation Analysis: Definition

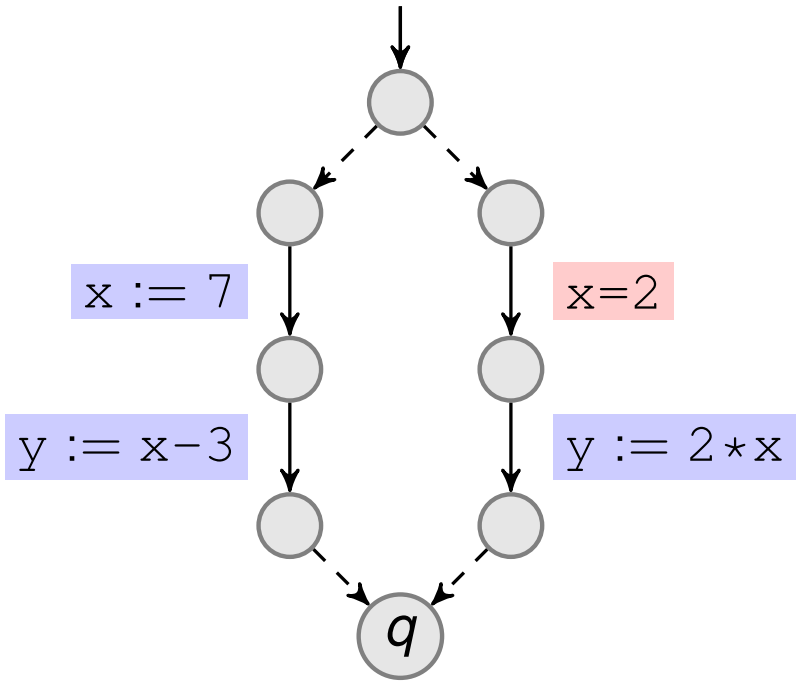
Definition

A variable x is constant at location q if we have $v(x) = v'(x)$ for any two reachable configurations (q, v) and (q, v') in Post*.

Constant Propagation Analysis: Definition

Definition

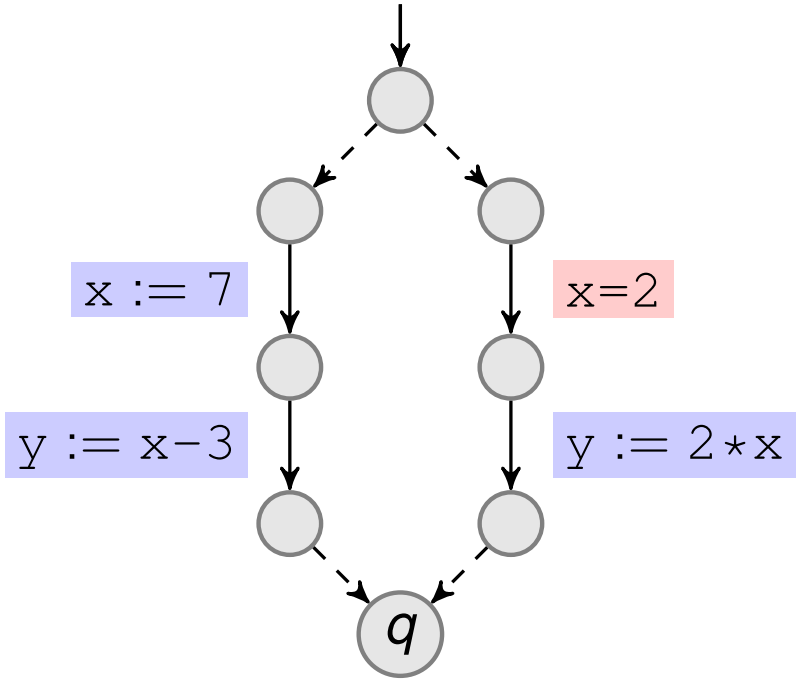
A variable x is constant at location q if we have $v(x) = v'(x)$ for any two reachable configurations (q, v) and (q, v') in Post*.



Constant Propagation Analysis: Definition

Definition

A variable x is constant at location q if we have $v(x) = v'(x)$ for any two reachable configurations (q, v) and (q, v') in Post*.

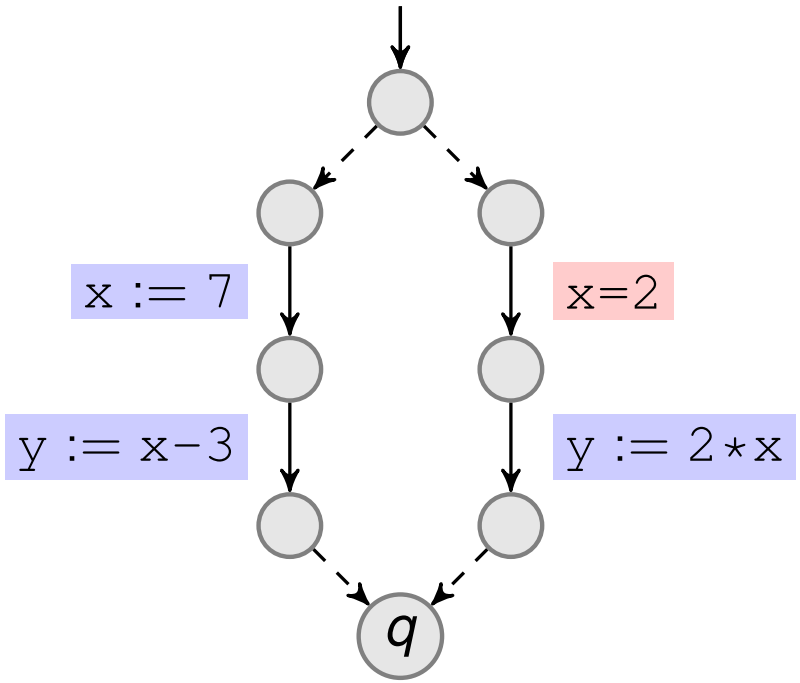


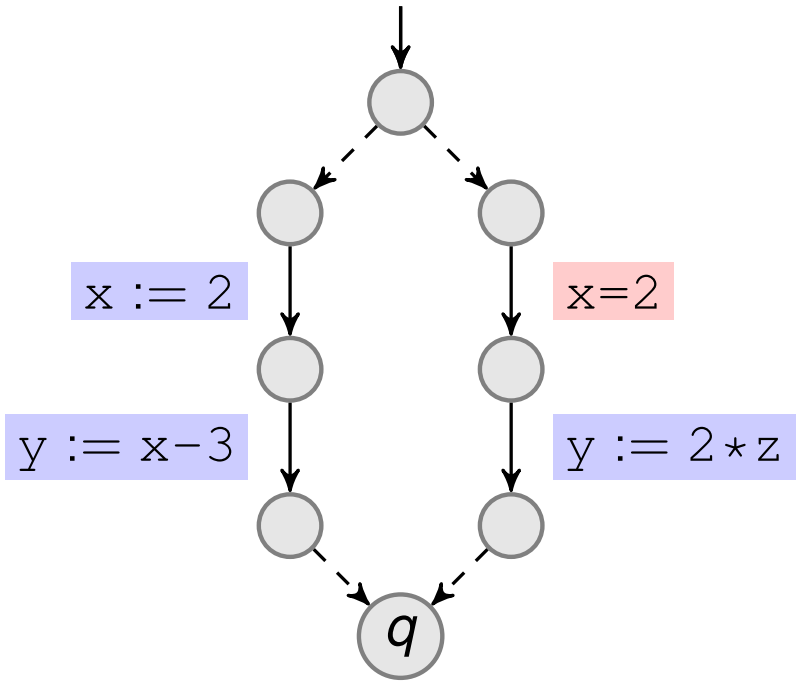
x not constant, y constant

Constant Propagation Analysis: Definition

Definition

A variable x is constant at location q if we have $v(x) = v'(x)$ for any two reachable configurations (q, v) and (q, v') in Post*.



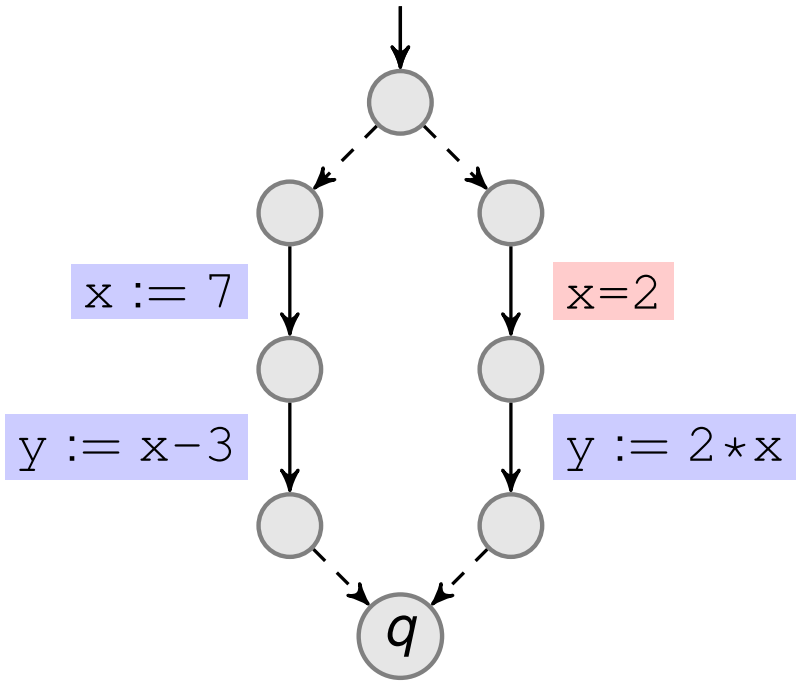


x not constant, y constant

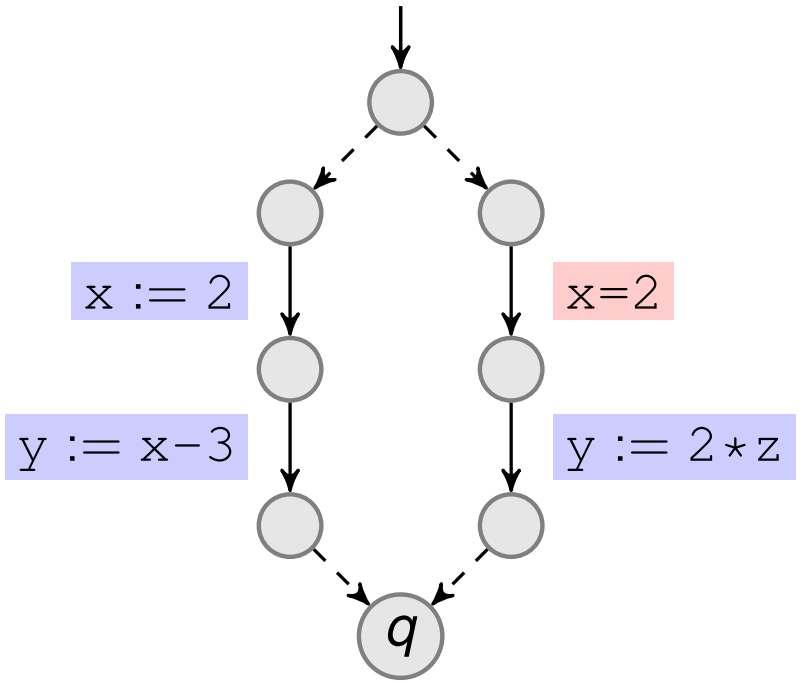
Constant Propagation Analysis: Definition

Definition

A variable x is constant at location q if we have $v(x) = v'(x)$ for any two reachable configurations (q, v) and (q, v') in Post*.

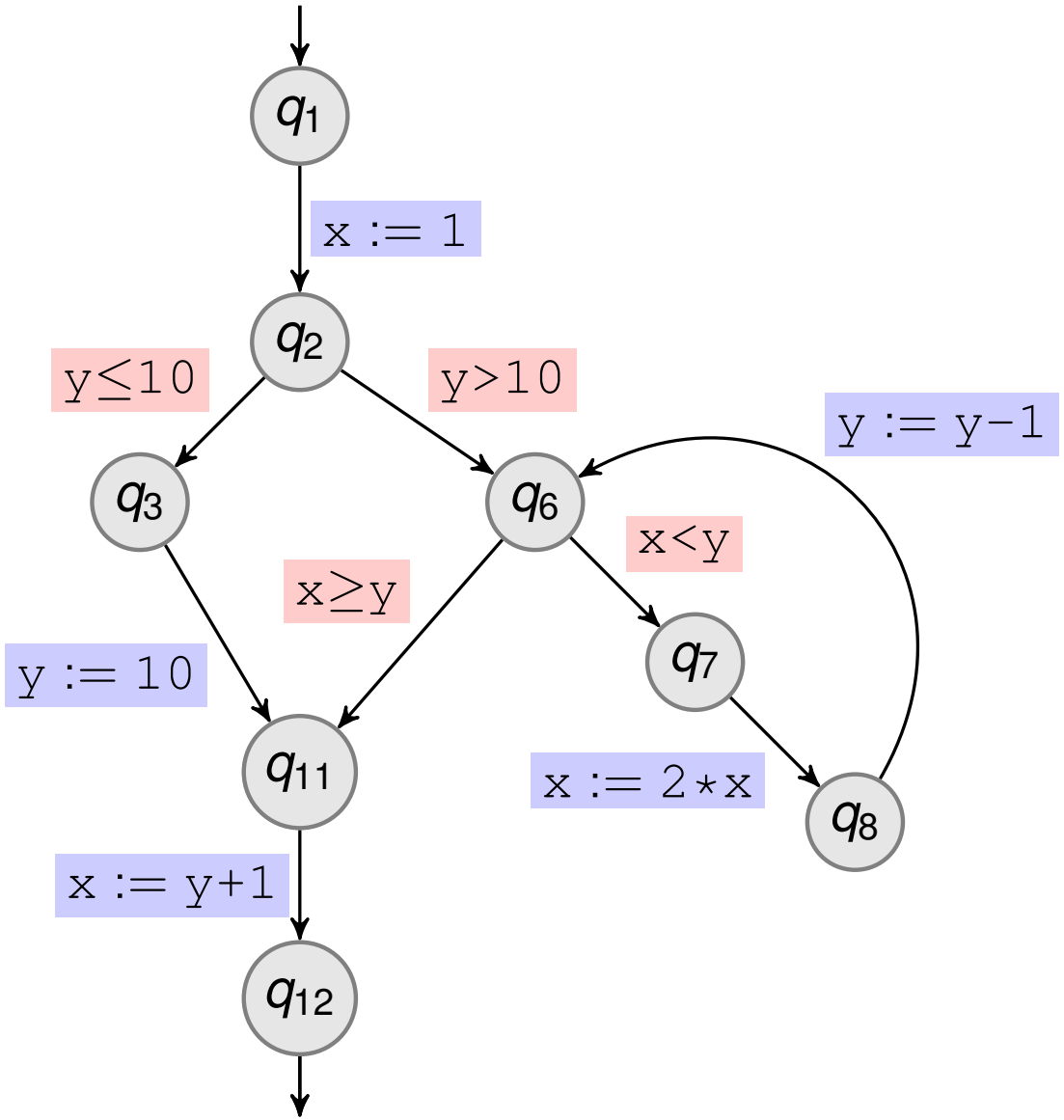


x not constant, y constant

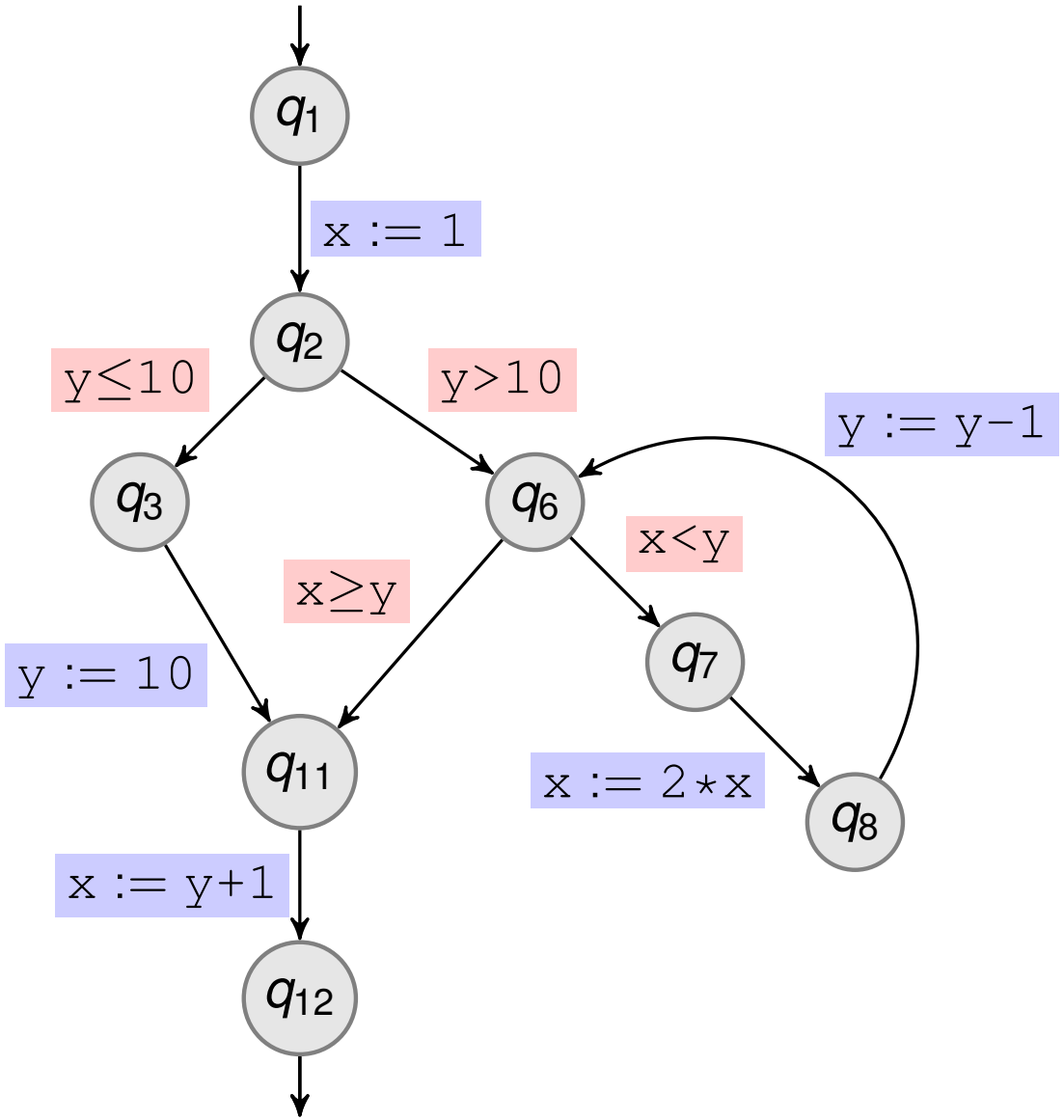


x constant, y not constant

Constant Propagation Analysis: Running Example



Constant Propagation Analysis: Running Example

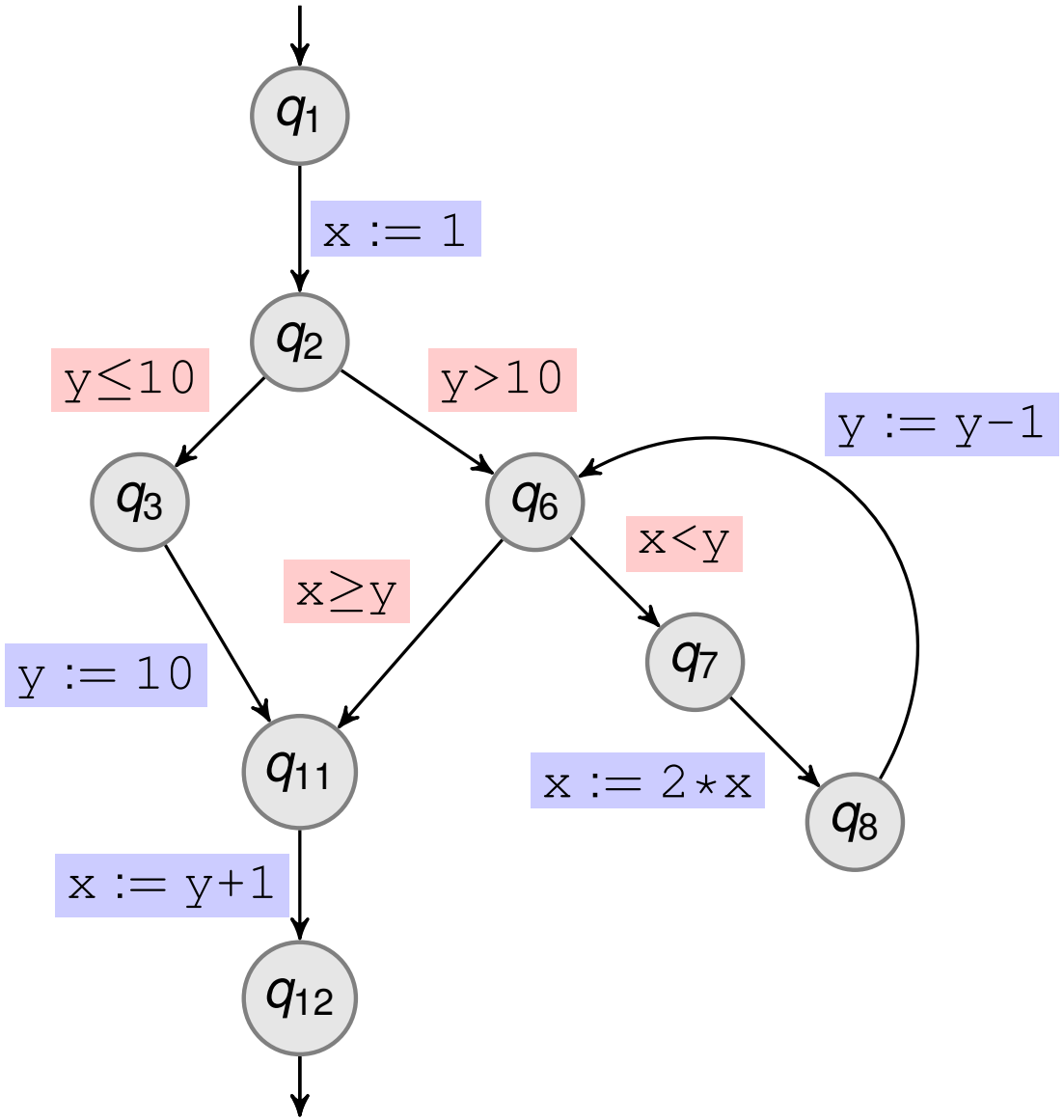


0 : Initialization

1 : Propagation ($\rightarrow$)

	x	y
q_1	$\top$	$\top$
q_2		
q_3		
q_6		
q_7		
q_8		
q_{11}		
q_{12}		

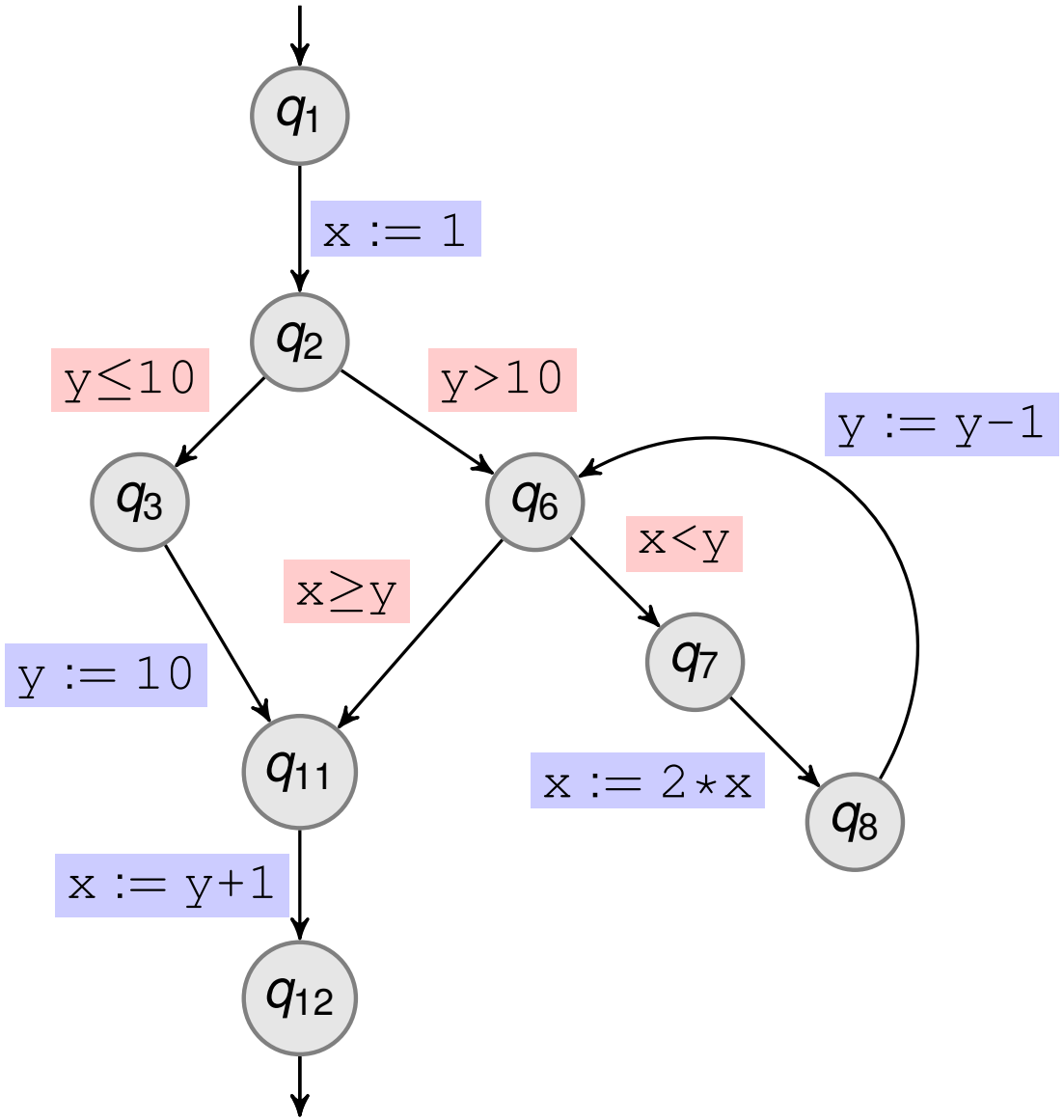
Constant Propagation Analysis: Running Example



0 : Initialization
1 : Propagation ($\rightarrow$)

	x	y
q_1	$\top$	$\top$
q_2	1	$\top$
q_3		
q_6		
q_7		
q_8		
q_{11}		
q_{12}		

Constant Propagation Analysis: Running Example

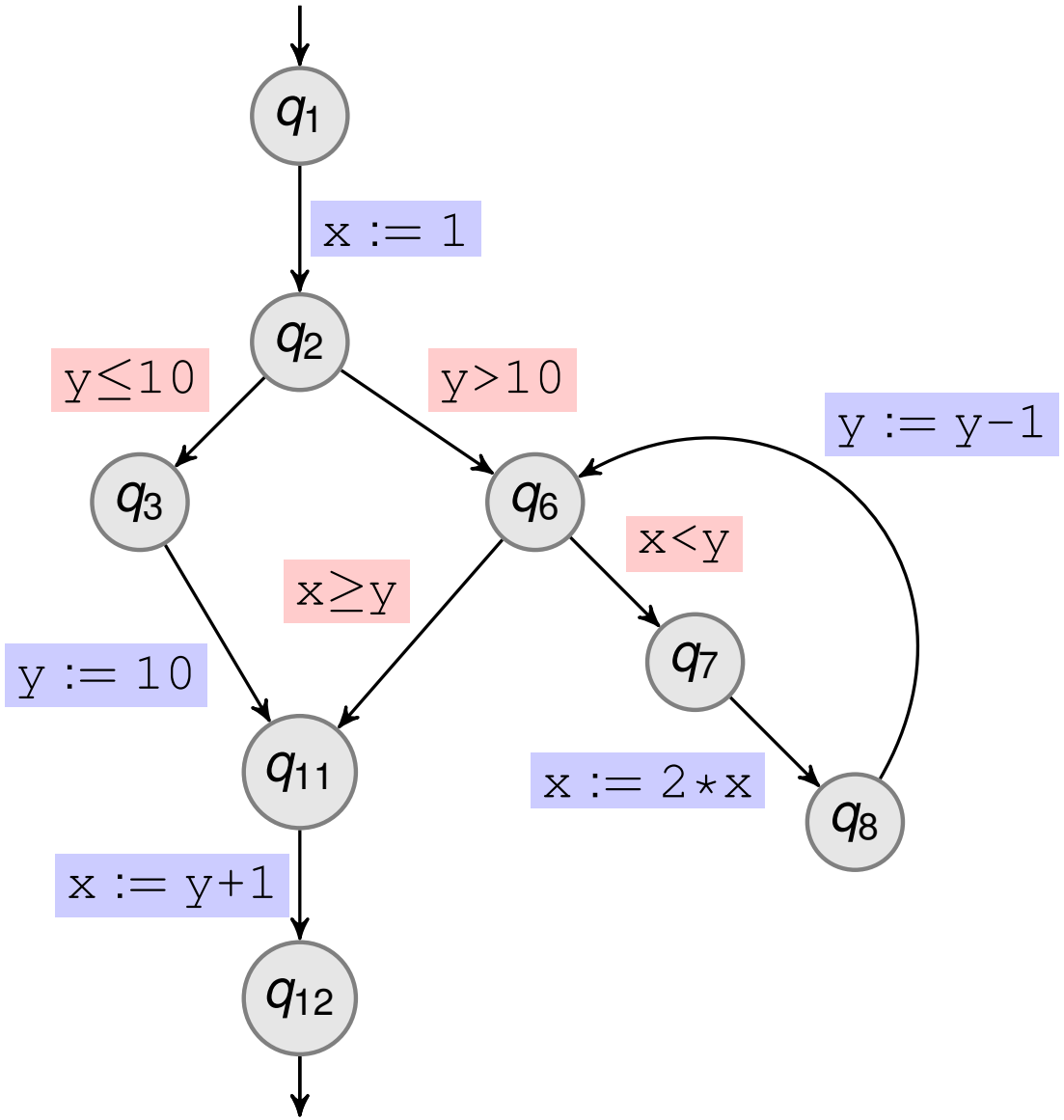


0 : Initialization

1 : Propagation ($\rightarrow$)

	x	y
q_1	$\top$	$\top$
q_2	1	$\top$
q_3		
q_6		
q_7		
q_8		
q_{11}		
q_{12}		

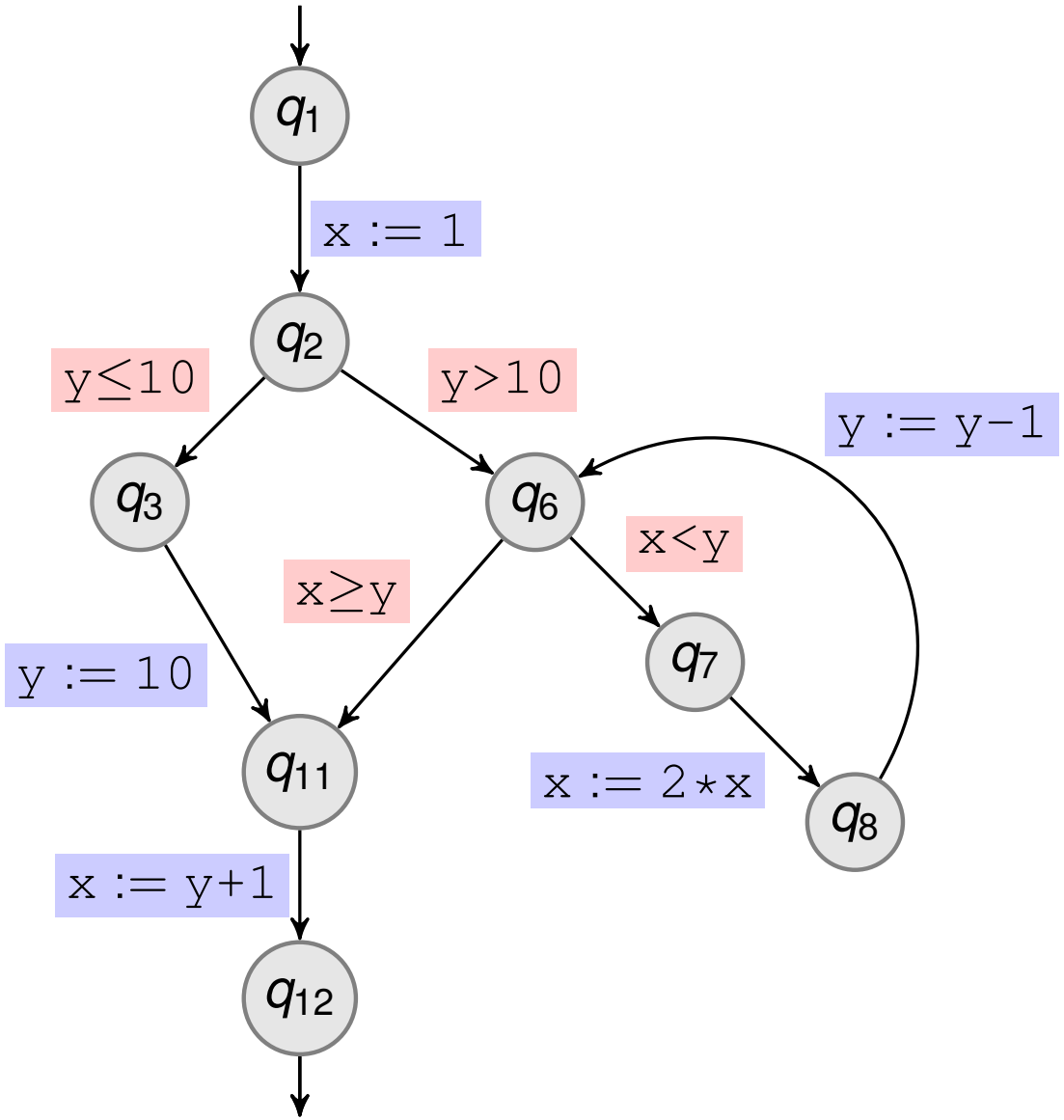
Constant Propagation Analysis: Running Example



0 : Initialization
1 : Propagation ($\rightarrow$)

	x	y
q_1	$\top$	$\top$
q_2	1	$\top$
q_3	1	$\top$
q_6		
q_7		
q_8		
q_{11}		
q_{12}		

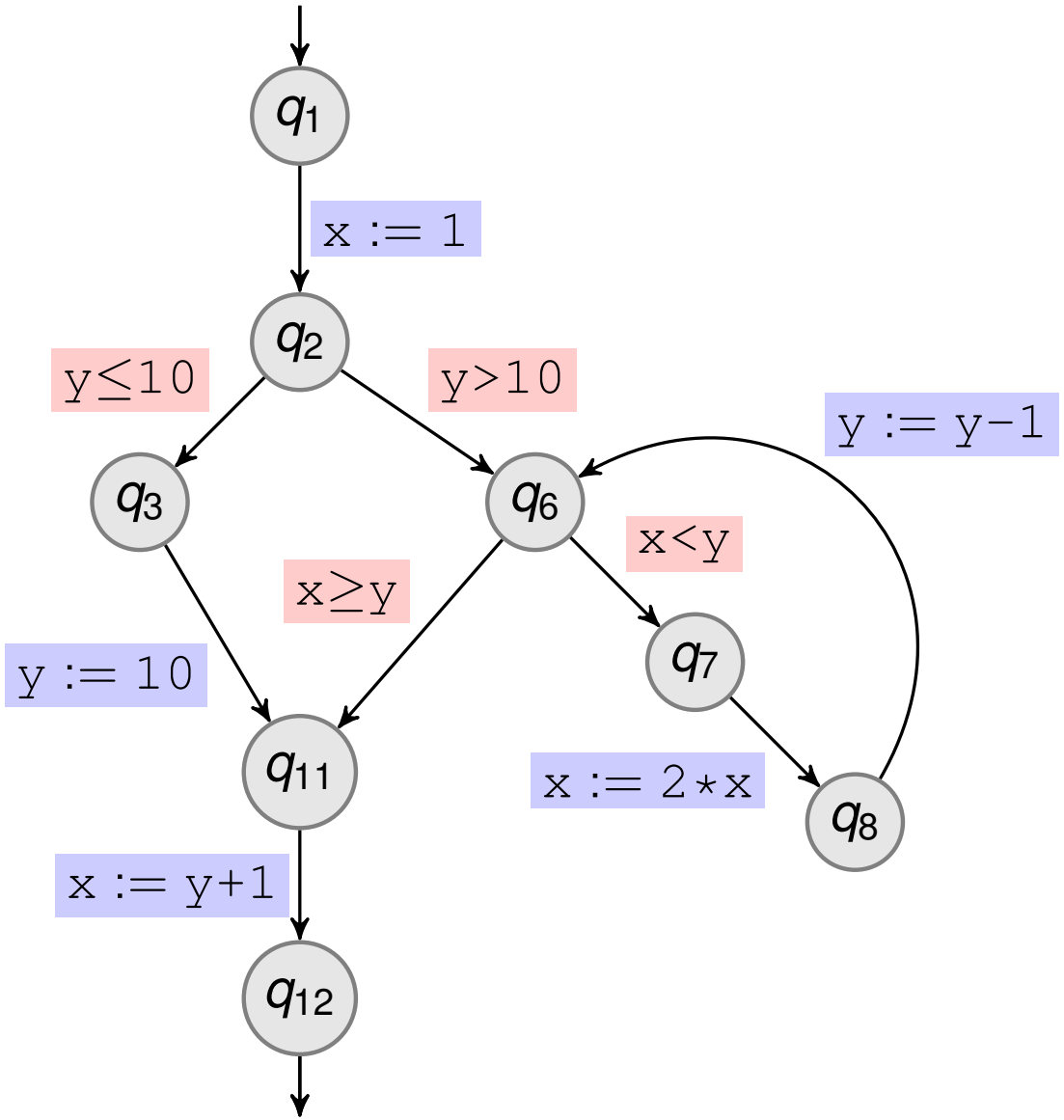
Constant Propagation Analysis: Running Example



0 : Initialization
1 : Propagation ($\rightarrow$)

	x	y
q_1	$\top$	$\top$
q_2	1	$\top$
q_3	1	$\top$
q_6		
q_7		
q_8		
q_{11}		
q_{12}		

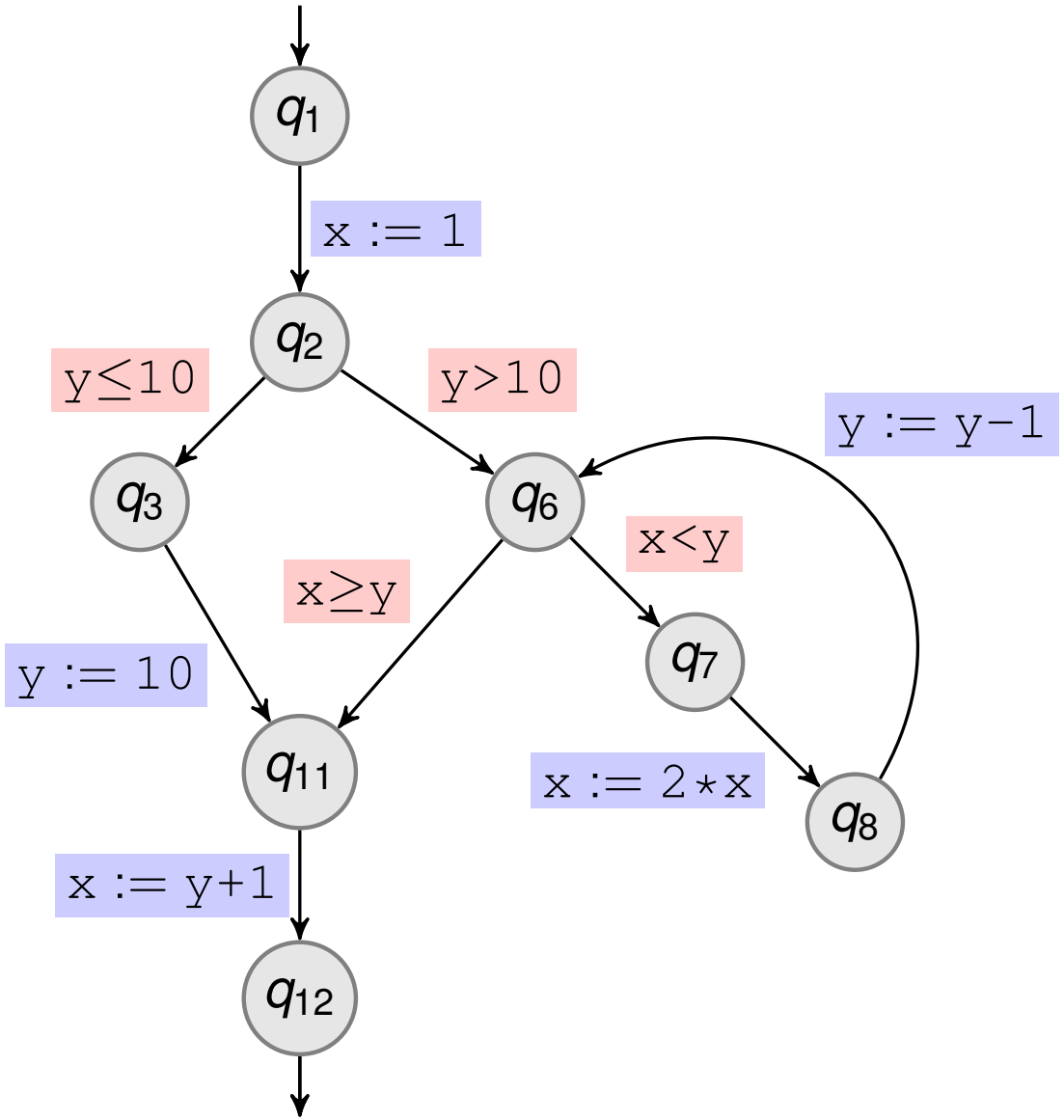
Constant Propagation Analysis: Running Example



0 : Initialization
1 : Propagation ($\rightarrow$)

	x	y
q_1	$\top$	$\top$
q_2	1	$\top$
q_3	1	$\top$
q_6		
q_7		
q_8		
q_{11}	1	10
q_{12}		

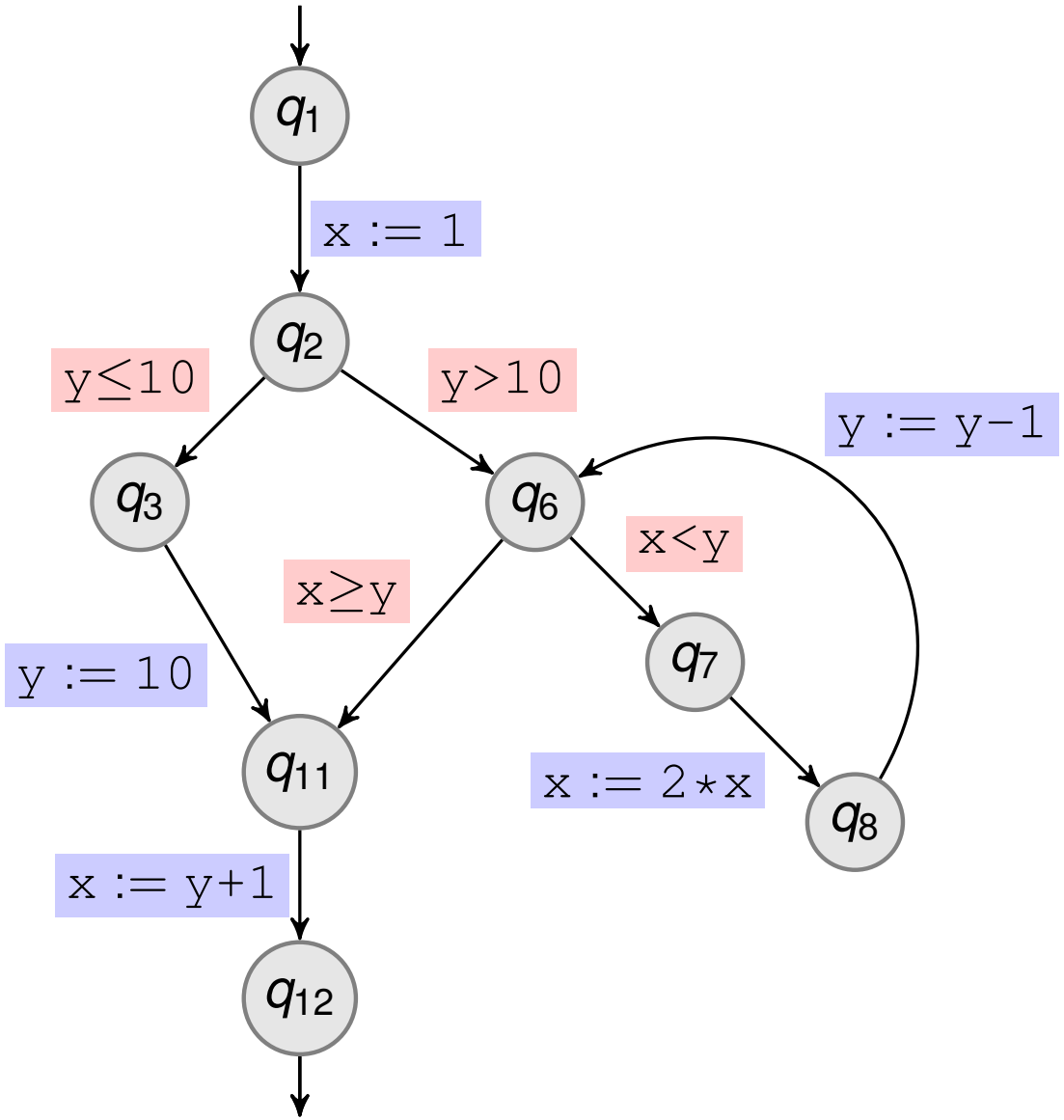
Constant Propagation Analysis: Running Example



0 : Initialization
1 : Propagation ($\rightarrow$)

	x	y
q_1	$\top$	$\top$
q_2	1	$\top$
q_3	1	$\top$
q_6		
q_7		
q_8		
q_{11}	1	10
q_{12}		

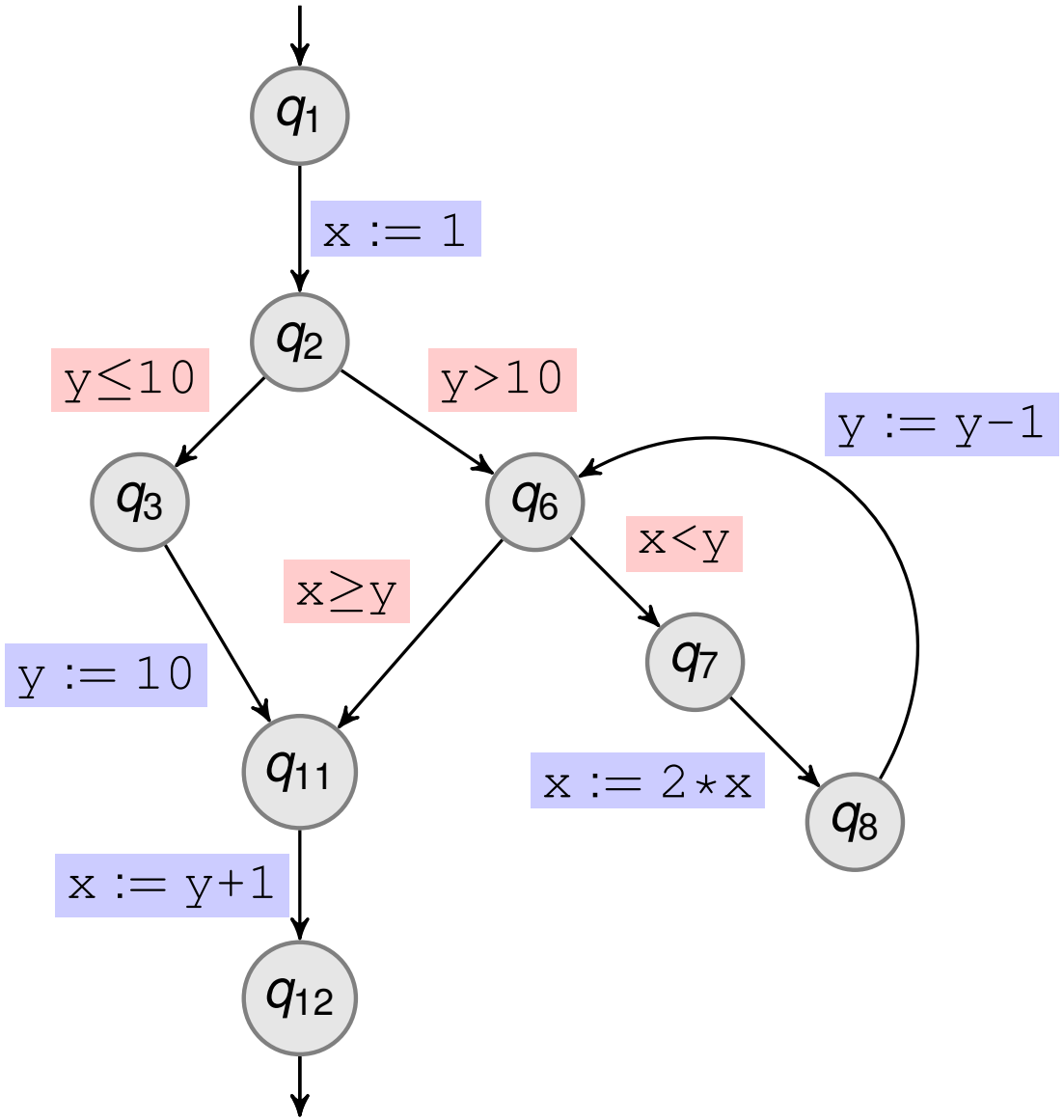
Constant Propagation Analysis: Running Example



0 : Initialization
1 : Propagation ($\rightarrow$)

	x	y
q_1	$\top$	$\top$
q_2	1	$\top$
q_3	1	$\top$
q_6		
q_7		
q_8		
q_{11}	1	10
q_{12}	11	10

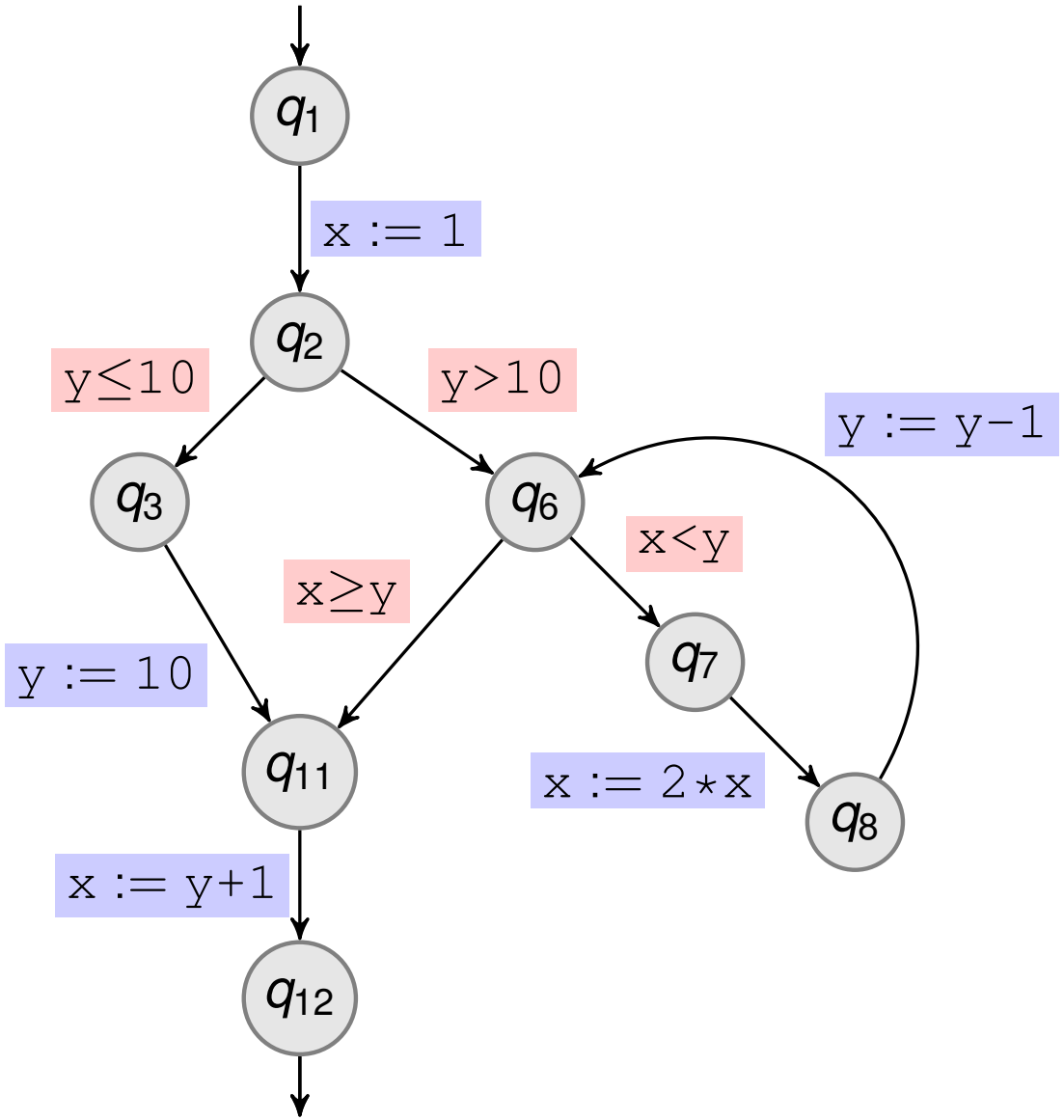
Constant Propagation Analysis: Running Example



0 : Initialization
1 : Propagation ($\rightarrow$)

	x	y
q_1	$\top$	$\top$
q_2	1	$\top$
q_3	1	$\top$
q_6		
q_7		
q_8		
q_{11}	1	10
q_{12}	11	10

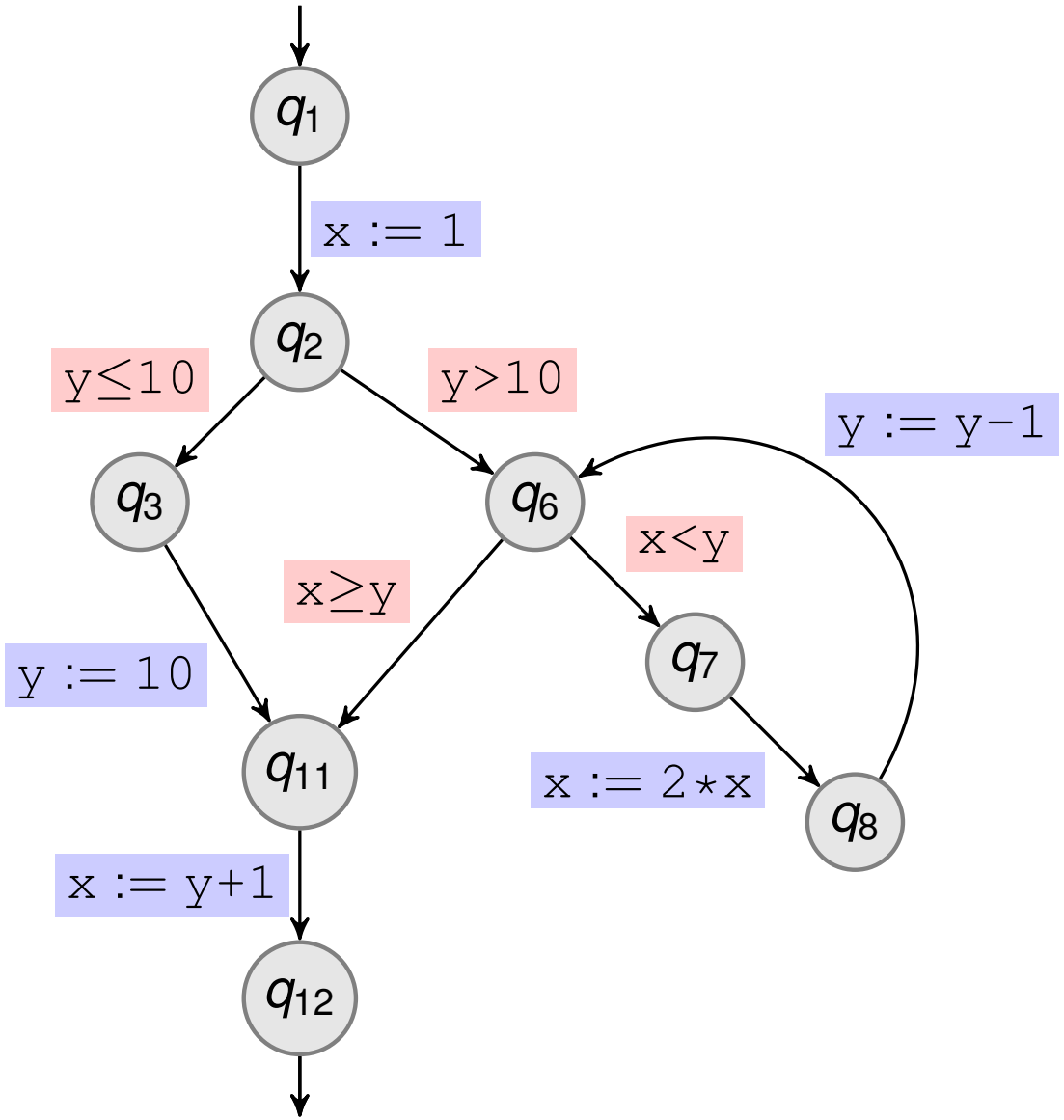
Constant Propagation Analysis: Running Example



0 : Initialization
1 : Propagation ($\rightarrow$)

	x	y
q_1	$\top$	$\top$
q_2	1	$\top$
q_3	1	$\top$
q_6	1	$\top$
q_7		
q_8		
q_{11}	1	10
q_{12}	11	10

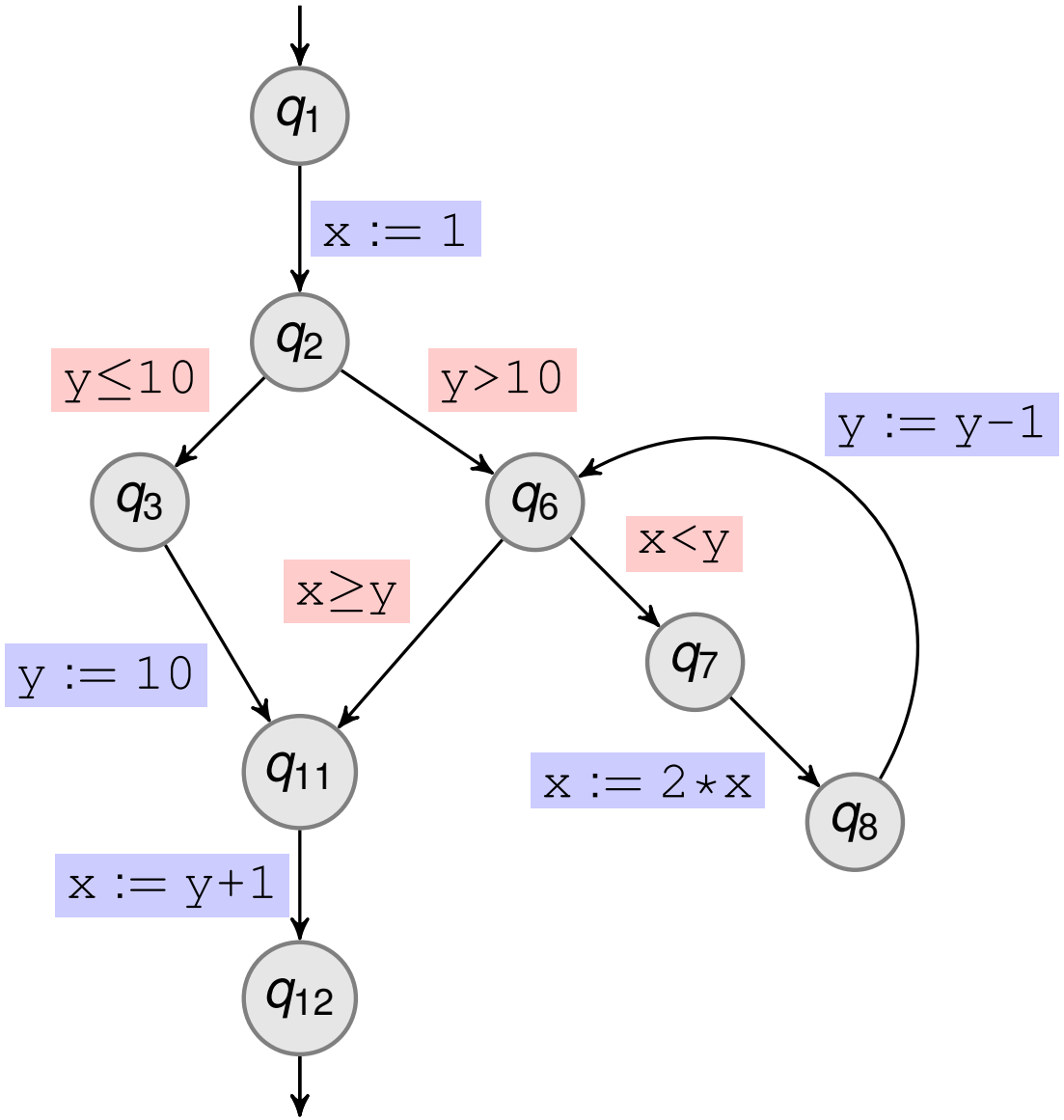
Constant Propagation Analysis: Running Example



0 : Initialization
1 : Propagation ($\rightarrow$)

	x	y
q_1	$\top$	$\top$
q_2	1	$\top$
q_3	1	$\top$
q_6	1	$\top$
q_7		
q_8		
q_{11}	1	10
q_{12}	11	10

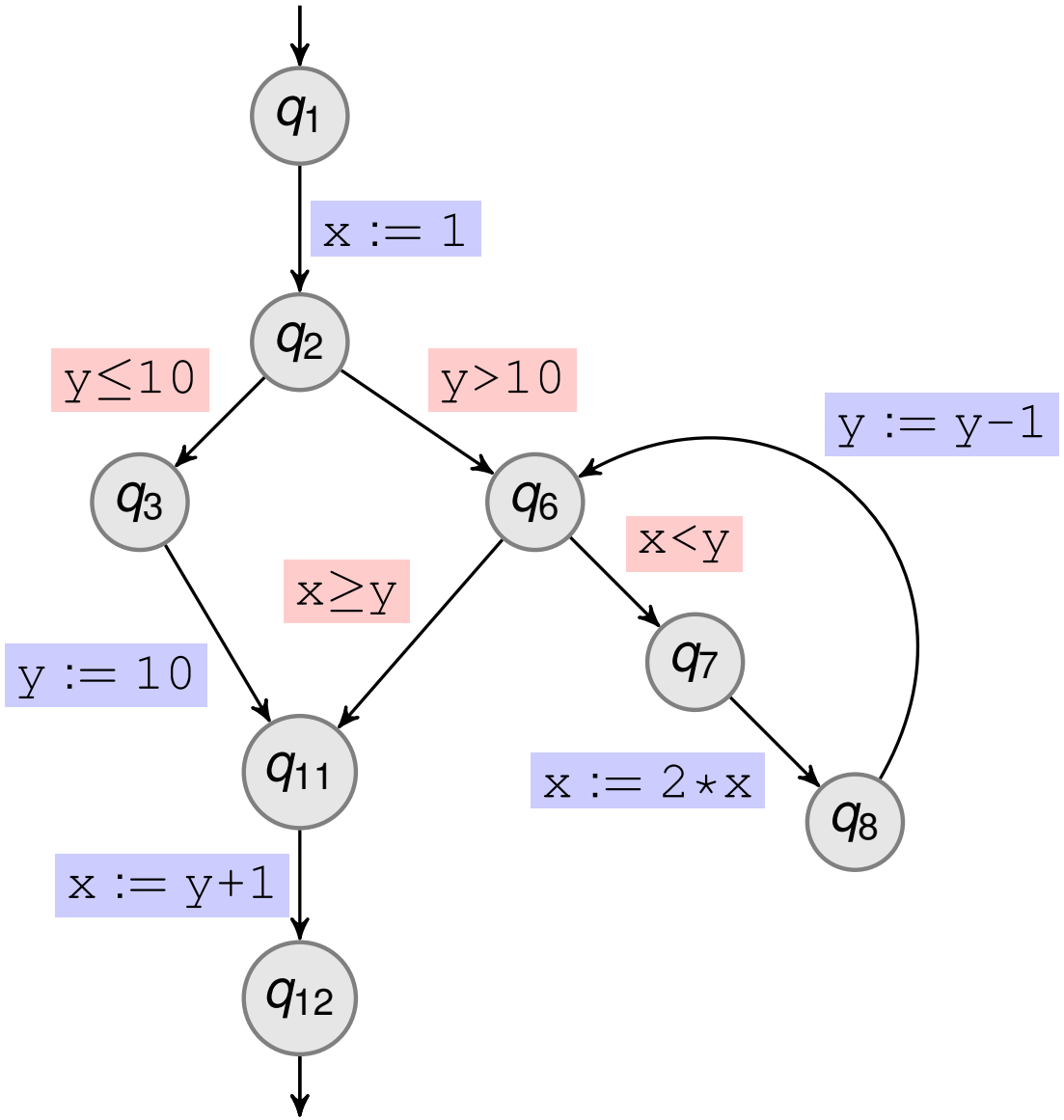
Constant Propagation Analysis: Running Example



0 : Initialization
1 : Propagation ($\rightarrow$)

	x	y
q_1	$\top$	$\top$
q_2	1	$\top$
q_3	1	$\top$
q_6	1	$\top$
q_7		
q_8		
q_{11}	1	10, $\top$
q_{12}	11	10

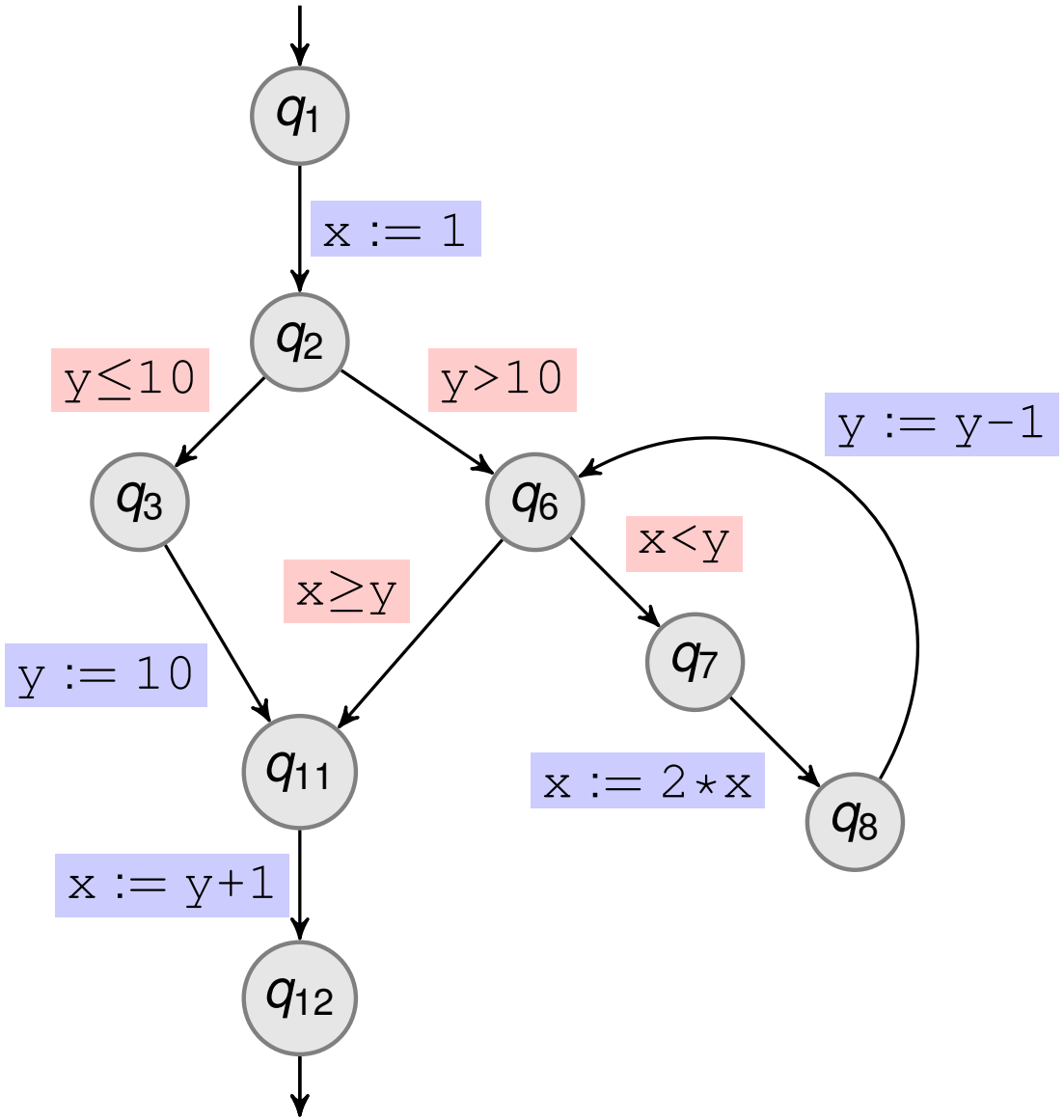
Constant Propagation Analysis: Running Example



0 : Initialization
1 : Propagation ($\rightarrow$)

	x	y
q_1	$\top$	$\top$
q_2	1	$\top$
q_3	1	$\top$
q_6	1	$\top$
q_7		
q_8		
q_{11}	1	$\top$
q_{12}	11	10

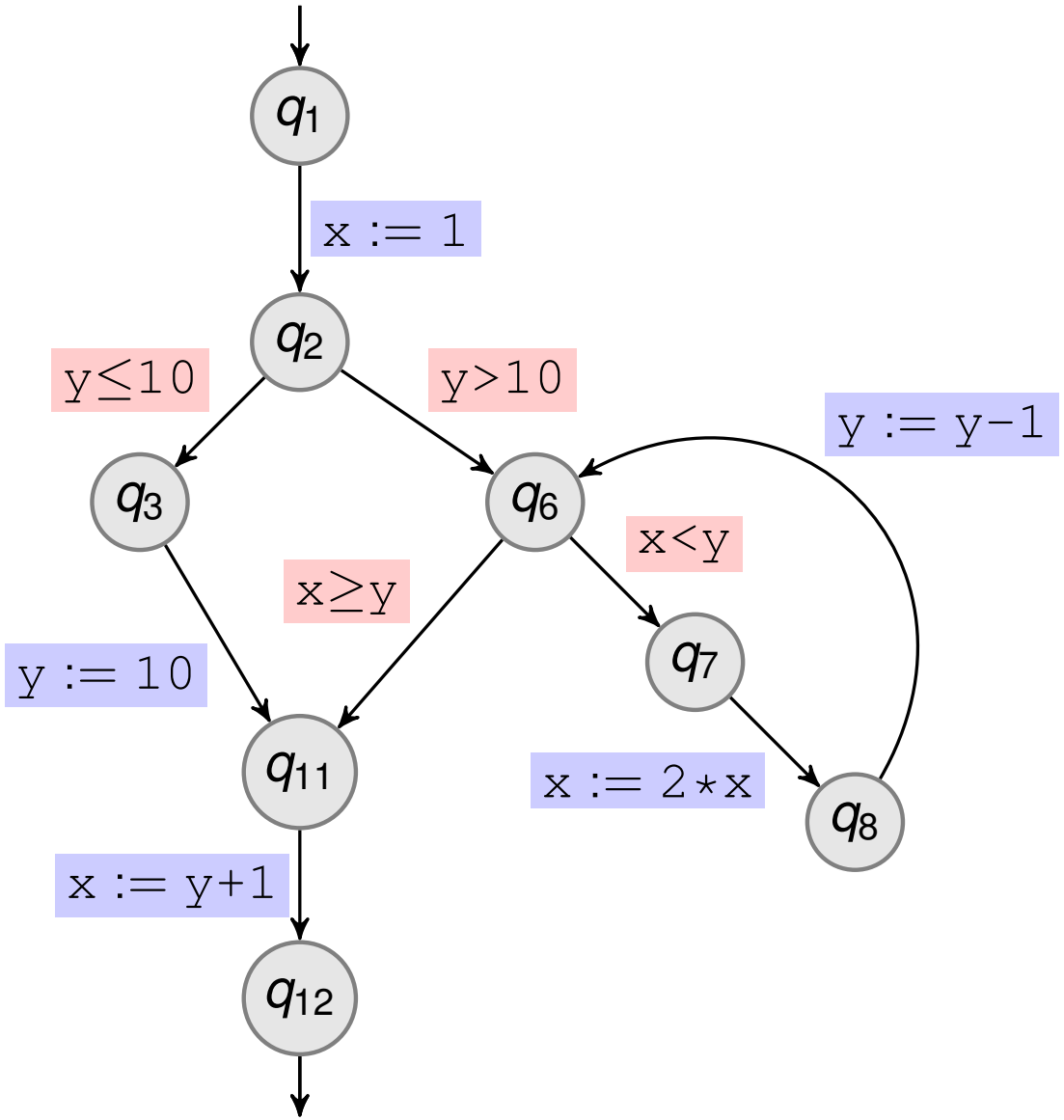
Constant Propagation Analysis: Running Example



0 : Initialization
1 : Propagation ($\rightarrow$)

	x	y
q_1	$\top$	$\top$
q_2	1	$\top$
q_3	1	$\top$
q_6	1	$\top$
q_7		
q_8		
q_{11}	1	$\top$
q_{12}	11, 2	10, $\top$

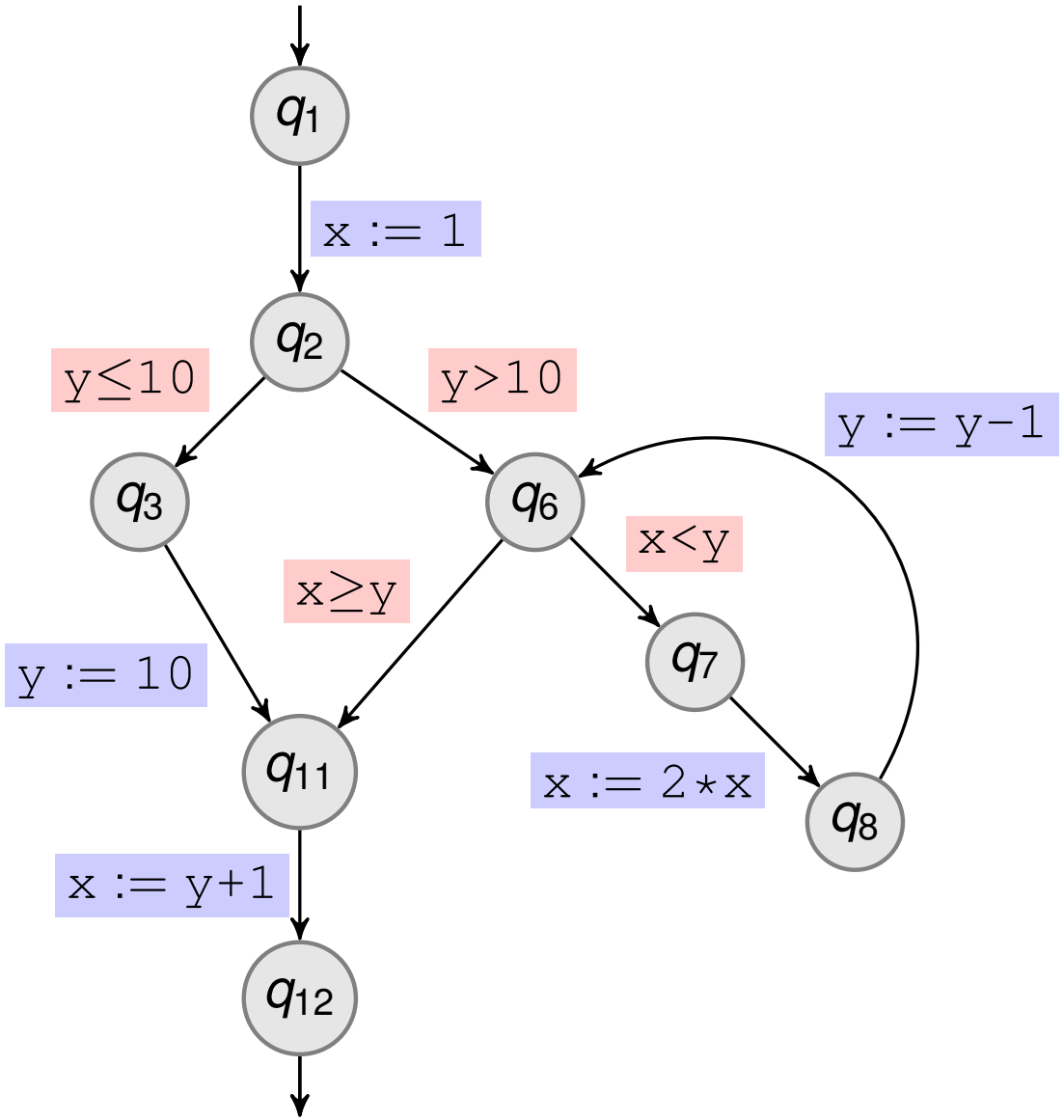
Constant Propagation Analysis: Running Example



0 : Initialization
1 : Propagation ($\rightarrow$)

	x	y
q_1	$\top$	$\top$
q_2	1	$\top$
q_3	1	$\top$
q_6	1	$\top$
q_7		
q_8		
q_{11}	1	$\top$
q_{12}	$\top$	$\top$

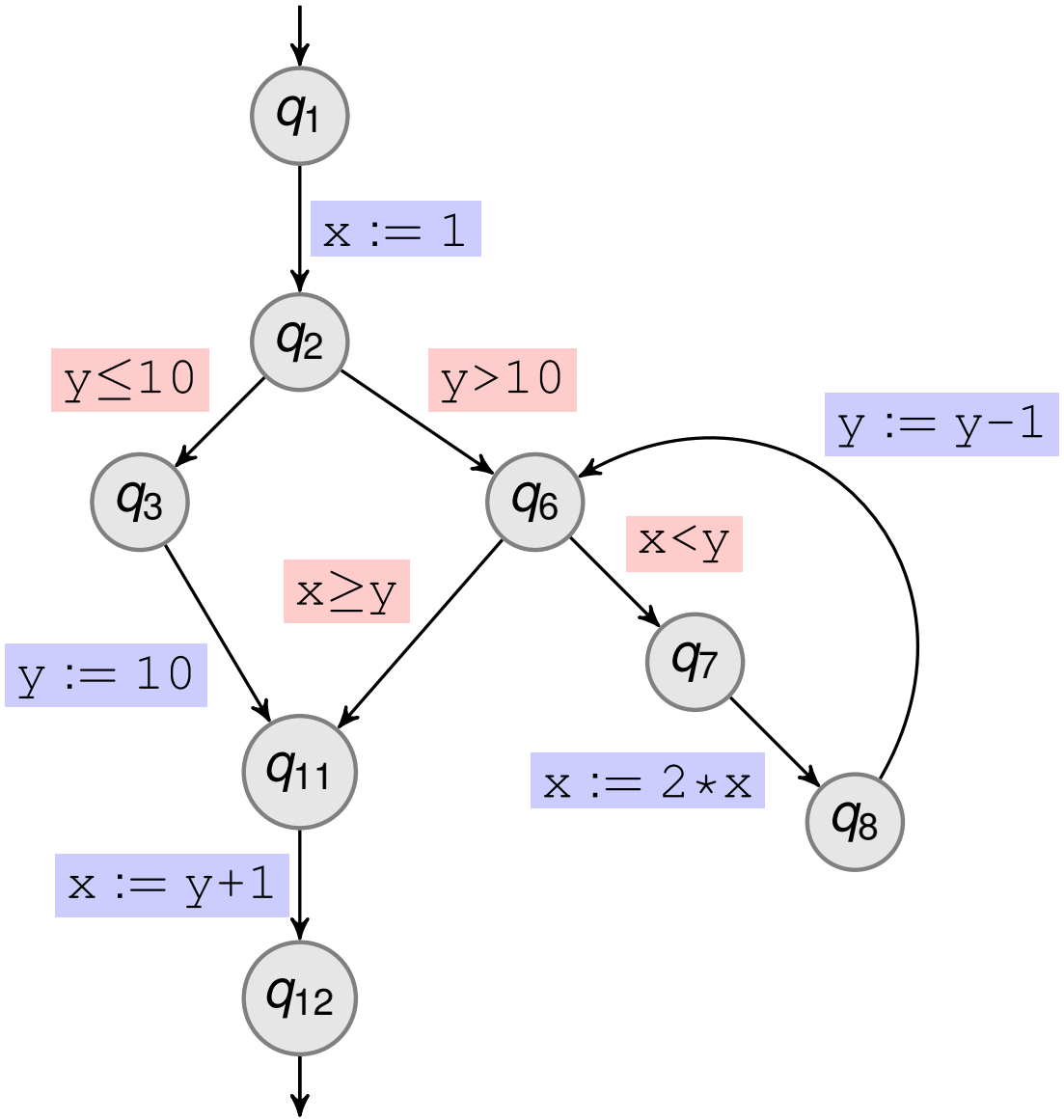
Constant Propagation Analysis: Running Example



0 : Initialization
1 : Propagation ($\rightarrow$)

	x	y
q_1	$\top$	$\top$
q_2	1	$\top$
q_3	1	$\top$
q_6	1	$\top$
q_7	1	$\top$
q_8		
q_{11}	1	$\top$
q_{12}	$\top$	$\top$

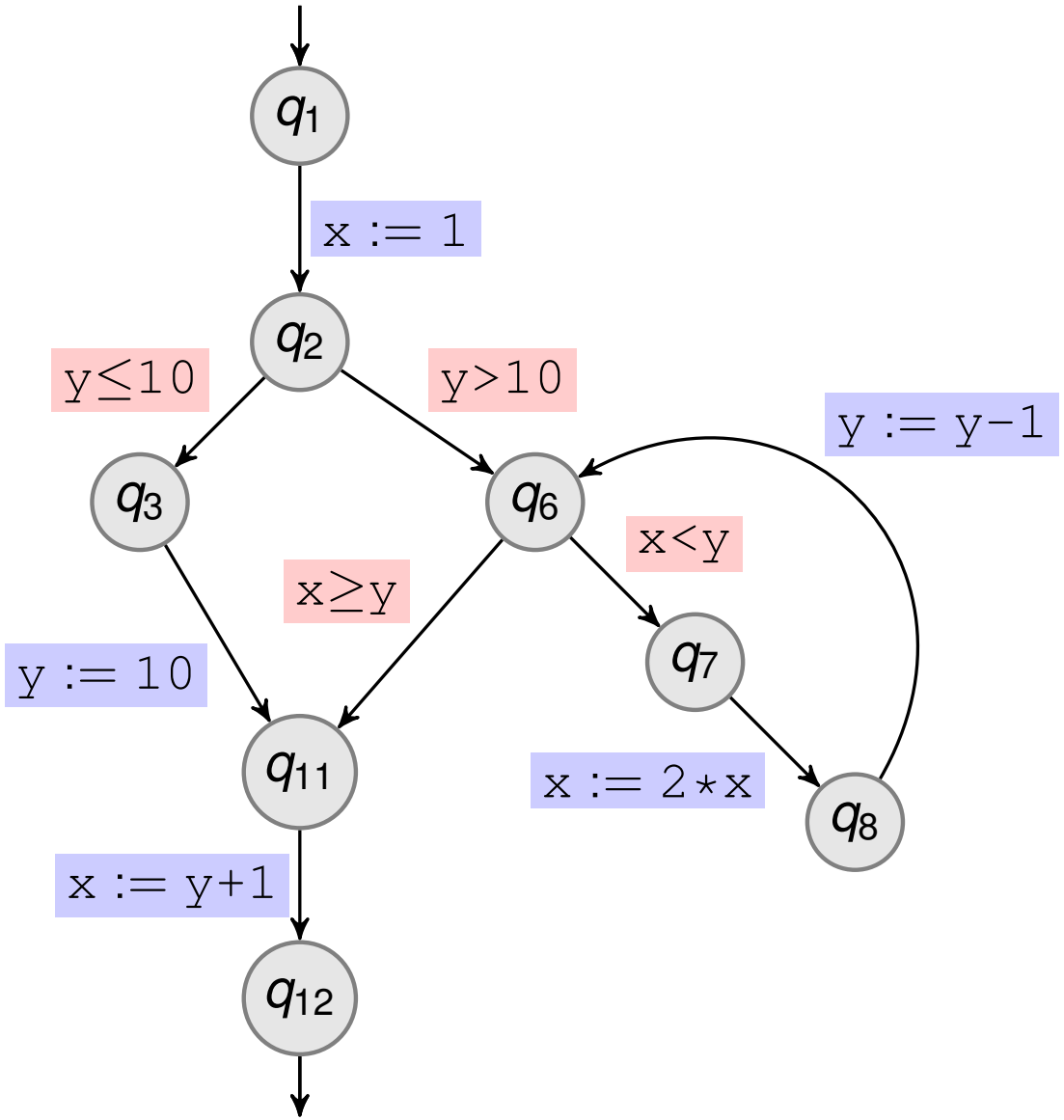
Constant Propagation Analysis: Running Example



0 : Initialization
1 : Propagation ($\rightarrow$)

	x	y
q_1	$\top$	$\top$
q_2	1	$\top$
q_3	1	$\top$
q_6	1	$\top$
q_7	1	$\top$
q_8		
q_{11}	1	$\top$
q_{12}	$\top$	$\top$

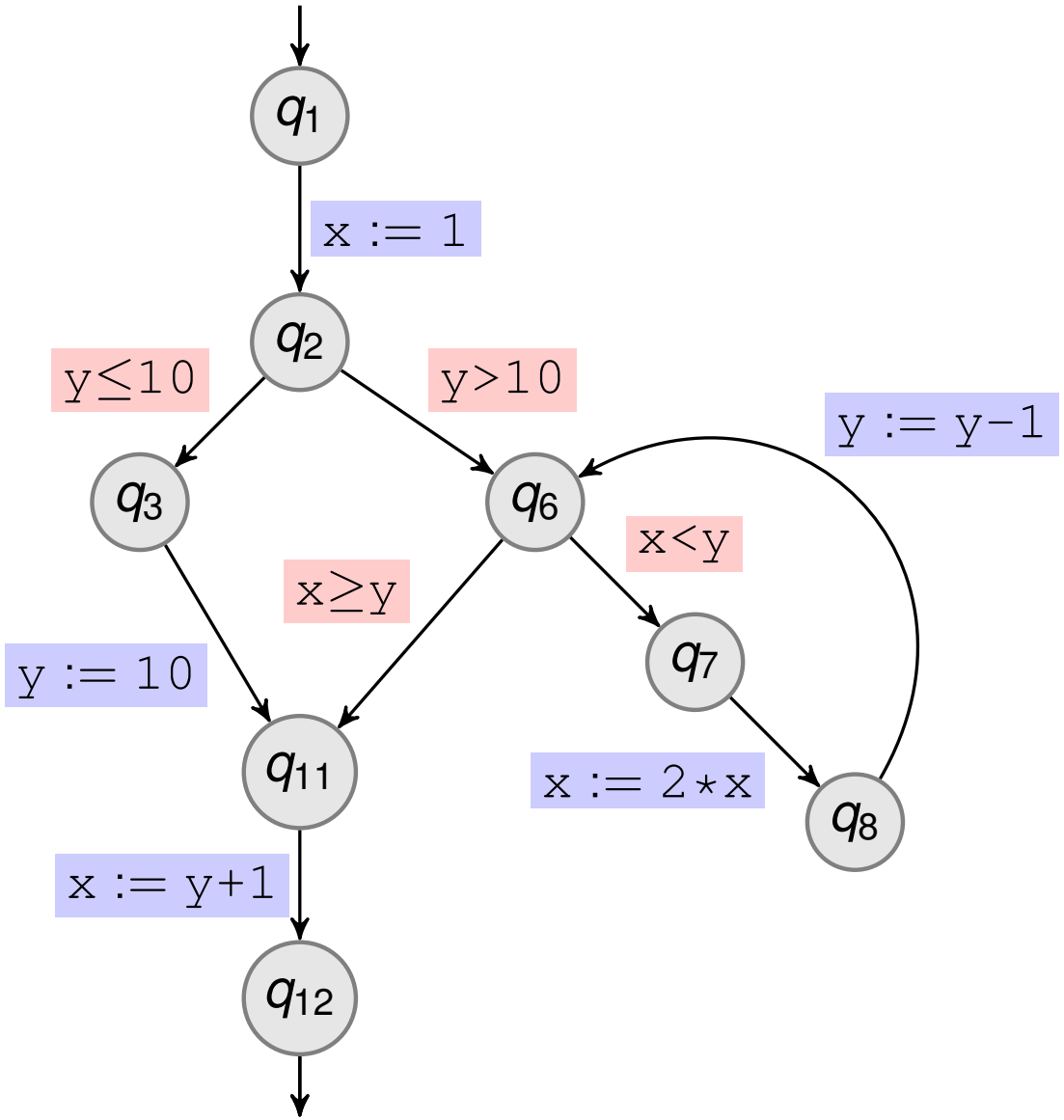
Constant Propagation Analysis: Running Example



0 : Initialization
1 : Propagation ($\rightarrow$)

	x	y
q_1	$\top$	$\top$
q_2	1	$\top$
q_3	1	$\top$
q_6	1	$\top$
q_7	1	$\top$
q_8	2	$\top$
q_{11}	1	$\top$
q_{12}	$\top$	$\top$

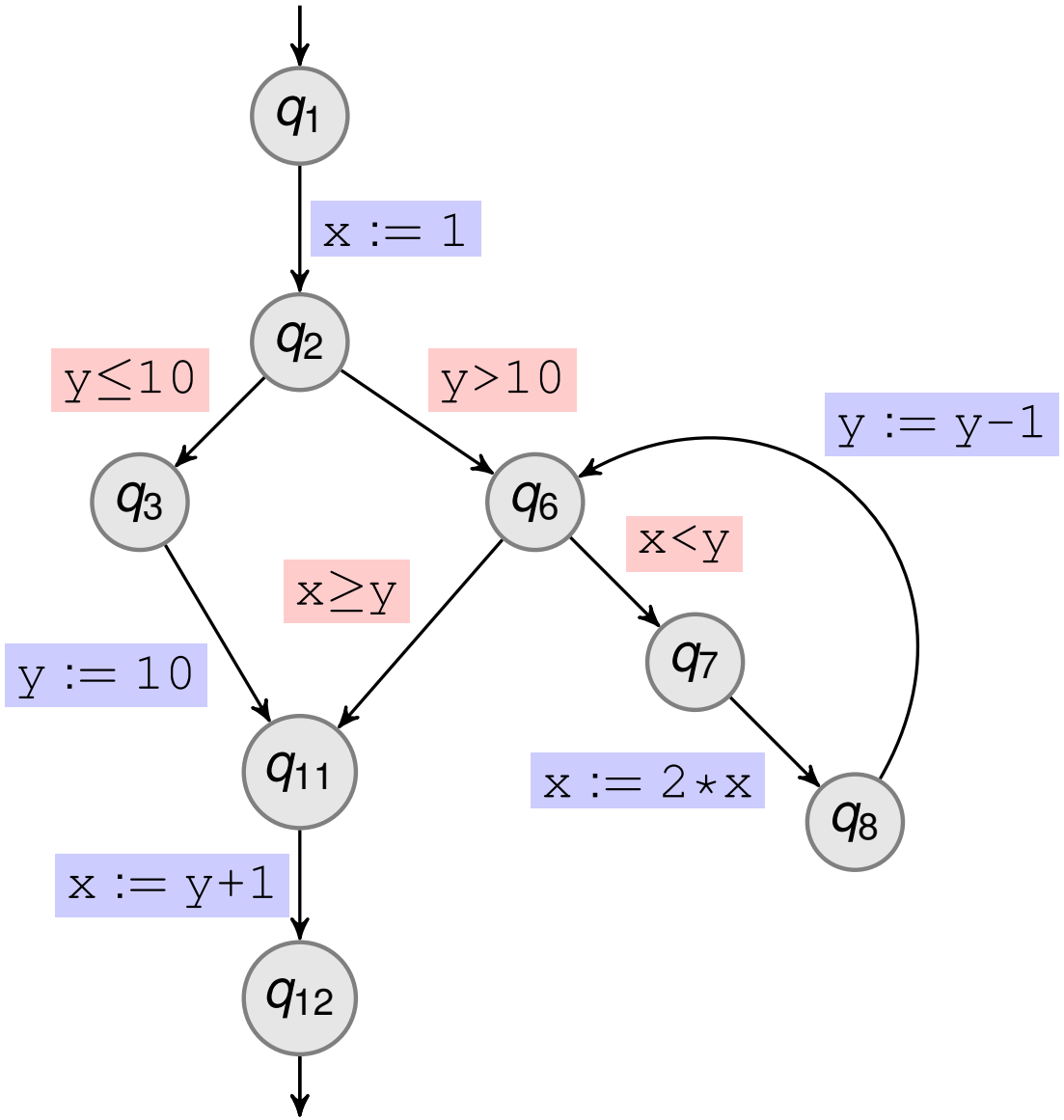
Constant Propagation Analysis: Running Example



0 : Initialization
1 : Propagation ($\rightarrow$)

	x	y
q_1	$\top$	$\top$
q_2	1	$\top$
q_3	1	$\top$
q_6	1	$\top$
q_7	1	$\top$
q_8	2	$\top$
q_{11}	1	$\top$
q_{12}	$\top$	$\top$

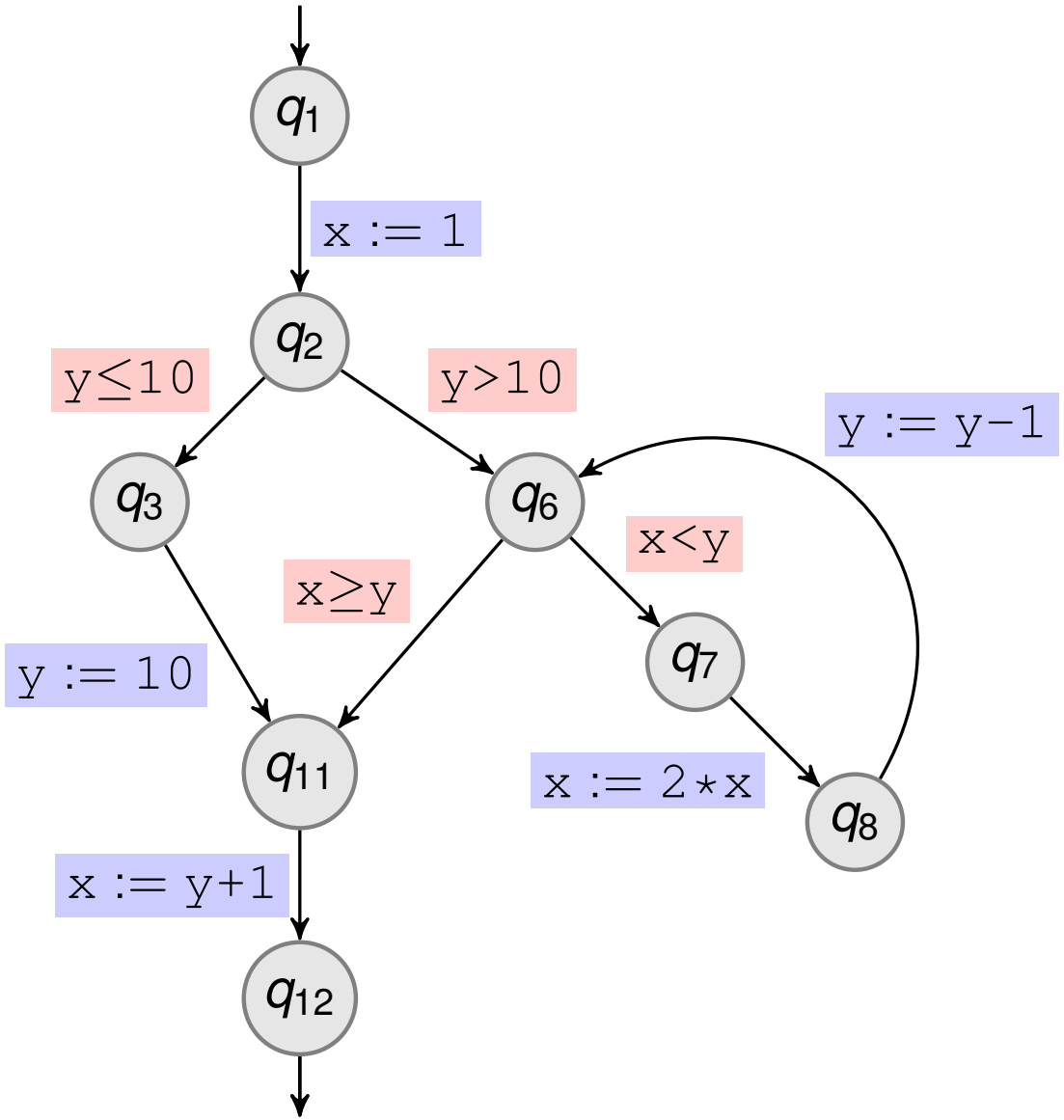
Constant Propagation Analysis: Running Example



0 : Initialization
1 : Propagation ($\rightarrow$)

	x	y
q_1	$\top$	$\top$
q_2	1	$\top$
q_3	1	$\top$
q_6	1, 2	$\top$
q_7	1	$\top$
q_8	2	$\top$
q_{11}	1	$\top$
q_{12}	$\top$	$\top$

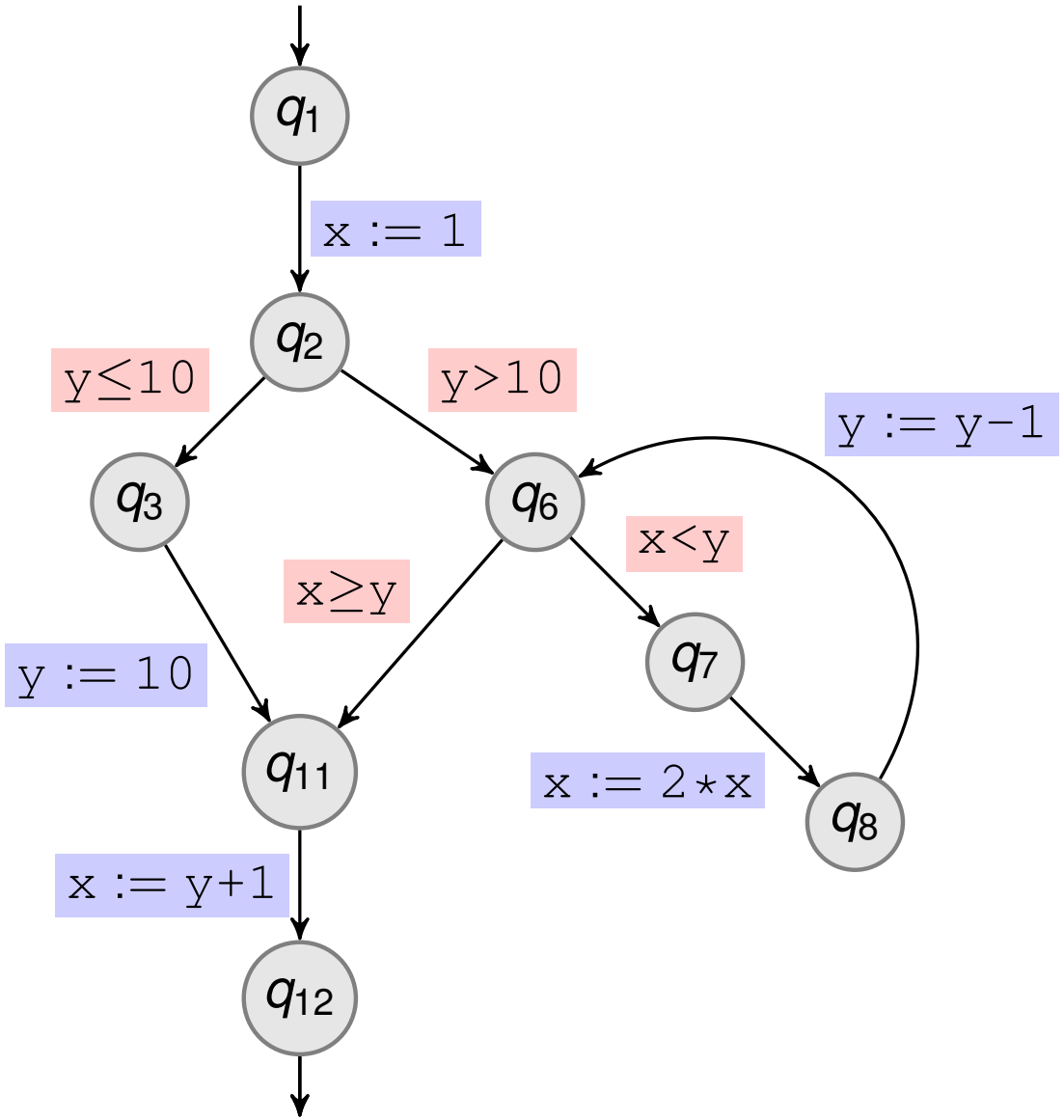
Constant Propagation Analysis: Running Example



0 : Initialization
1 : Propagation ($\rightarrow$)

	x	y
q_1	$\top$	$\top$
q_2	1	$\top$
q_3	1	$\top$
q_6	$\top$	$\top$
q_7	1	$\top$
q_8	2	$\top$
q_{11}	1	$\top$
q_{12}	$\top$	$\top$

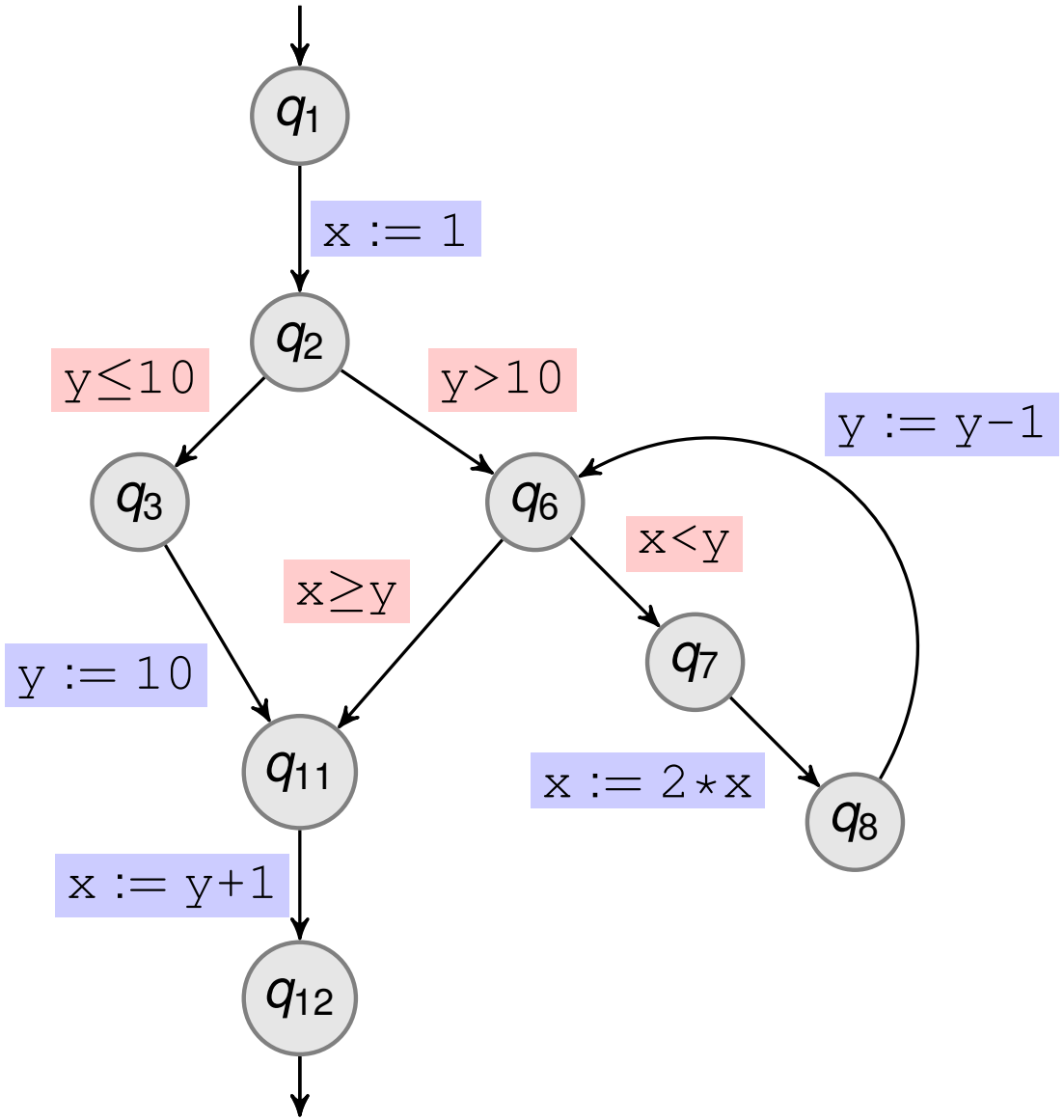
Constant Propagation Analysis: Running Example



0 : Initialization
1 : Propagation ($\rightarrow$)

	x	y
q_1	$\top$	$\top$
q_2	1	$\top$
q_3	1	$\top$
q_6	$\top$	$\top$
q_7	1	$\top$
q_8	2	$\top$
q_{11}	1, $\top$	$\top$
q_{12}	$\top$	$\top$

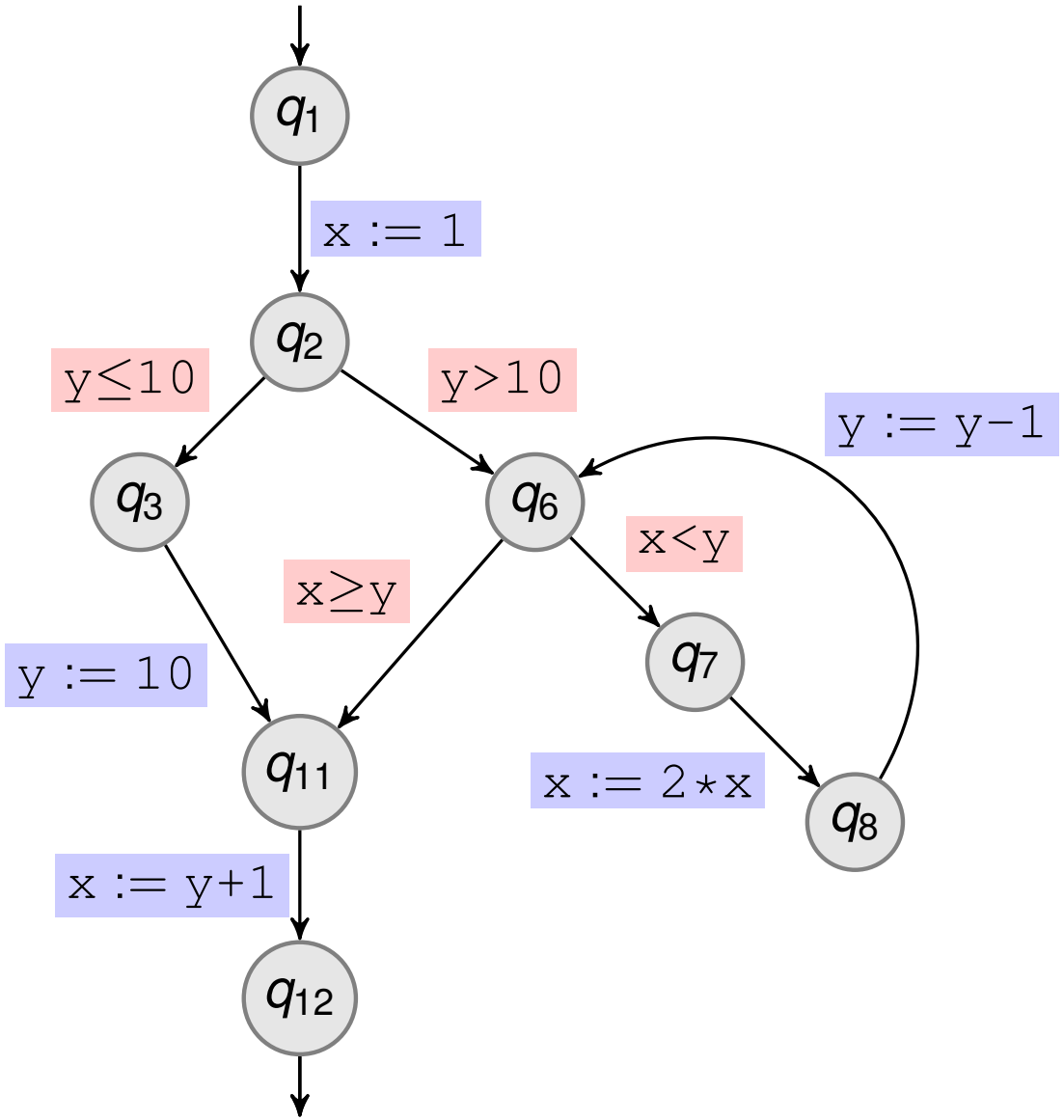
Constant Propagation Analysis: Running Example



0 : Initialization
1 : Propagation ($\rightarrow$)

	x	y
q_1	$\top$	$\top$
q_2	1	$\top$
q_3	1	$\top$
q_6	$\top$	$\top$
q_7	1	$\top$
q_8	2	$\top$
q_{11}	$\top$	$\top$
q_{12}	$\top$	$\top$

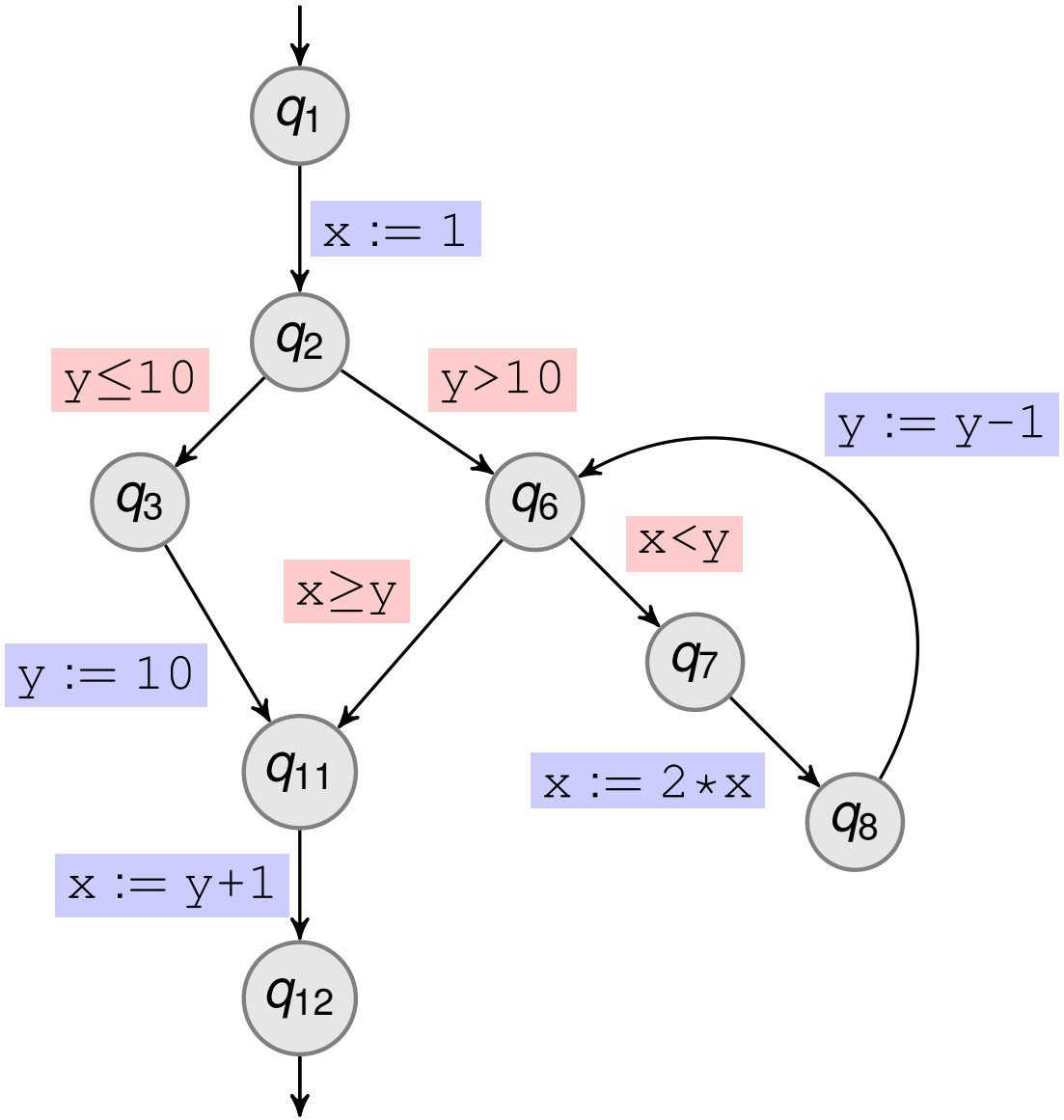
Constant Propagation Analysis: Running Example



0 : Initialization
1 : Propagation ($\rightarrow$)

	x	y
q_1	$\top$	$\top$
q_2	1	$\top$
q_3	1	$\top$
q_6	$\top$	$\top$
q_7	1, $\top$	$\top$
q_8	2	$\top$
q_{11}	$\top$	$\top$
q_{12}	$\top$	$\top$

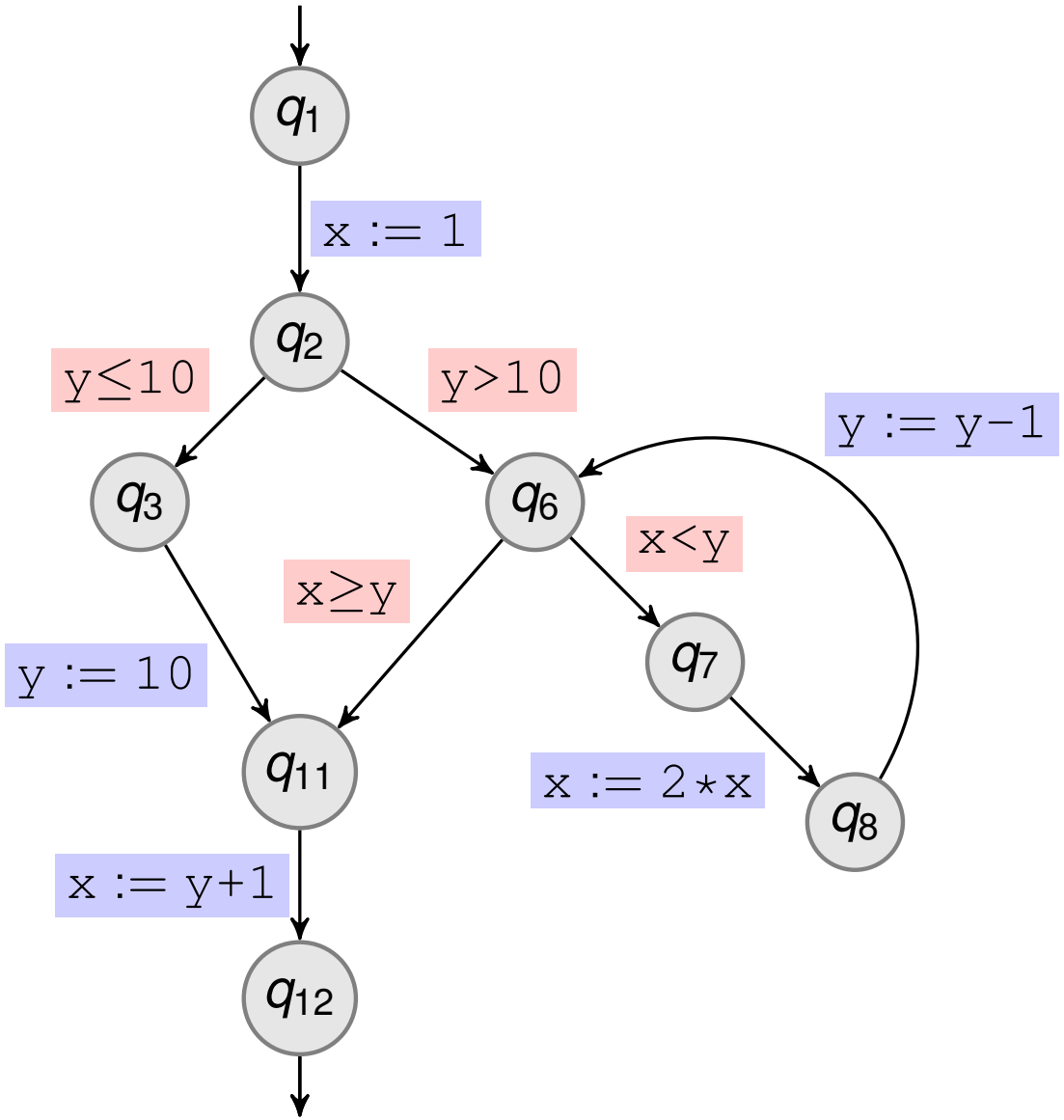
Constant Propagation Analysis: Running Example



0 : Initialization
1 : Propagation ($\rightarrow$)

	x	y
q_1	$\top$	$\top$
q_2	1	$\top$
q_3	1	$\top$
q_6	$\top$	$\top$
q_7	$\top$	$\top$
q_8	2	$\top$
q_{11}	$\top$	$\top$
q_{12}	$\top$	$\top$

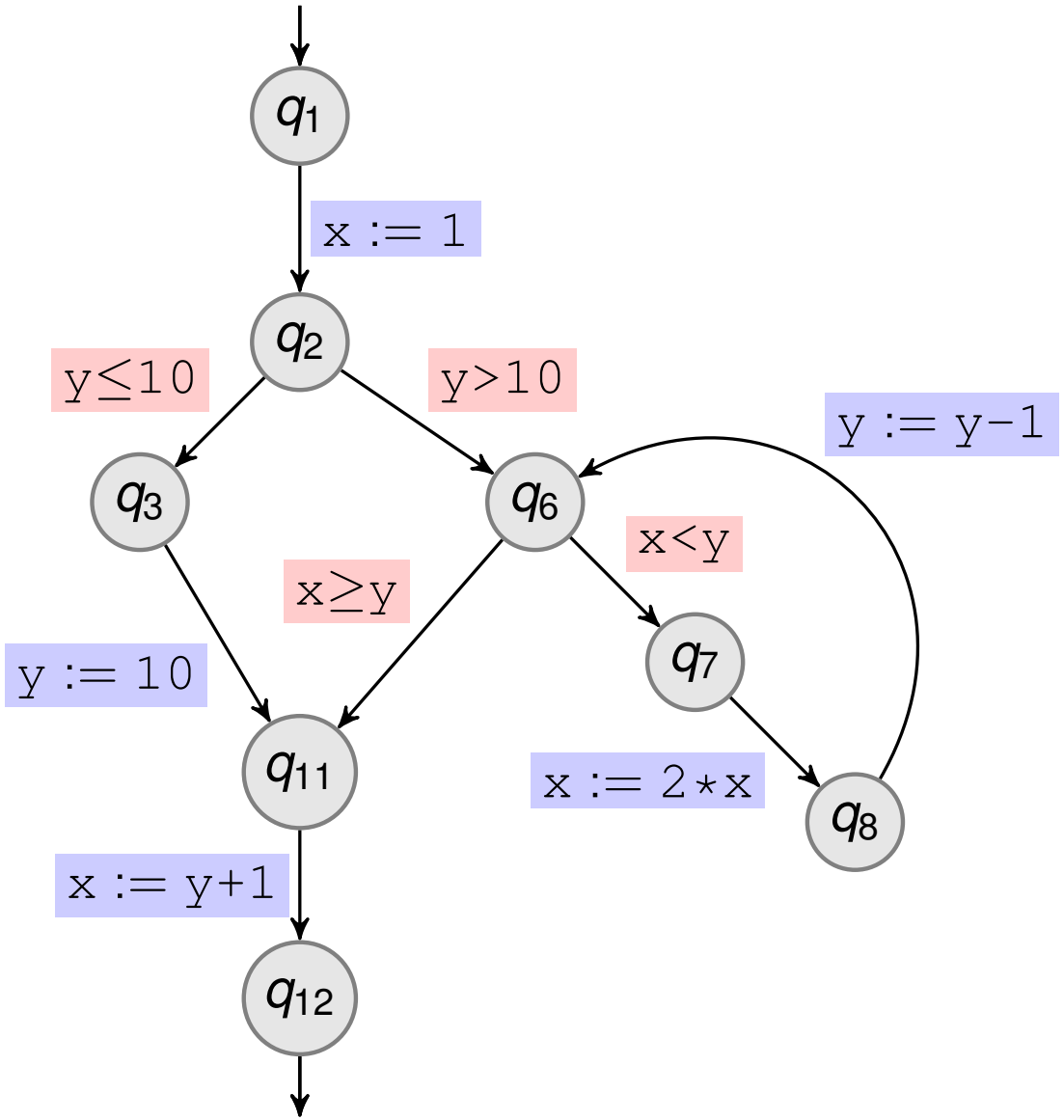
Constant Propagation Analysis: Running Example



0 : Initialization
1 : Propagation ($\rightarrow$)

	x	y
q_1	$\top$	$\top$
q_2	1	$\top$
q_3	1	$\top$
q_6	$\top$	$\top$
q_7	$\top$	$\top$
q_8	2, $\top$	$\top$
q_{11}	$\top$	$\top$
q_{12}	$\top$	$\top$

Constant Propagation Analysis: Running Example



0 : Initialization
1 : Propagation ($\rightarrow$)

	x	y
q_1	$\top$	$\top$
q_2	1	$\top$
q_3	1	$\top$
q_6	$\top$	$\top$
q_7	$\top$	$\top$
q_8	$\top$	$\top$
q_{11}	$\top$	$\top$
q_{12}	$\top$	$\top$

Constant Propagation Analysis: Formulation

Extend $\mathbb{R}$ with a new element $\top$ to account for non-constant values

Extend $+$, $-$ and $\times$ such that $\top$ is absorbent

$$\begin{array}{rcccl} \top + r & = & r + \top & = & \top \\ \top - r & = & r - \top & = & \top \\ \top \times r & = & r \times \top & = & \top \end{array} \qquad \text{for } r \in \mathbb{R} \cup \{\top\}$$

Extend $[\![e]\!]_v$ to valuations from $\mathbb{X}$ to $\mathbb{R} \cup \{\top\}$

Domain of data flow "information"

$$\mathbb{D} \quad = \quad \mathbb{X} \quad \rightarrow \quad (\mathbb{R} \cup \{\top\})$$

Constant Propagation Analysis: Formulation

$$\mathbb{D} \quad = \quad \mathrm{X} \quad \rightarrow \quad (\mathbb{R} \cup \{\top\})$$

System of equations: variables C_q for $q \in Q$, with $C_q \in \mathbb{D}$

$$C_q \quad = \quad \bigotimes_{q' \xrightarrow{\mathrm{op}} q} f_{\mathrm{op}}\left(C_{q'}\right) \qquad\qquad C(q_{in}) \quad = \quad \lambda\, x\,.\, \top$$

$$v \otimes v' \quad = \quad \lambda\, y\,.\, \begin{cases} v(y) & \text{if } v(y) = v'(y) \\ \top & \text{otherwise} \end{cases}$$

Functions f_{op}

$$f_{x:=e}(v) \quad = \quad \lambda\, y\,.\, \begin{cases} v(y) & \text{if } y \neq x \\ [\![e]\!]_v & \text{if } y = x \end{cases} \qquad\qquad f_g(v) \quad = \quad v$$

Constant Propagation Analysis: Formulation

$$\mathbb{D} \quad = \quad \mathrm{X} \quad \to \quad (\mathbb{R} \cup \{\top\})$$

System of equations: variables C_q for $q \in Q$, with $C_q \in \mathbb{D}$

$$C_q \quad = \quad \bigotimes_{q' \xrightarrow{\mathrm{op}} q} f_{\mathrm{op}}\left(C_{q'}\right) \qquad\qquad C(q_{in}) \quad = \quad \lambda\, x\,.\, \top$$

$$v \otimes v' \quad = \quad \lambda\, y\,.\, \begin{cases} v(y) & \text{if } v(y) = v'(y) \\ \top & \text{otherwise} \end{cases}$$

Functions f_{op}

$$f_{x:=e}(v) \quad = \quad \lambda\, y\,.\, \begin{cases} v(y) & \text{if } y \neq x \\ [\![e]\!]_v & \text{if } y = x \end{cases} \qquad\qquad f_g(v) \quad = \quad v$$

Constant Propagation Analysis: Applications

Code Optimization

Constant folding

For each variable y occurring in e, if y is constant at location q_1 then we may replace y with its constant value in e.

This is sound since the analysis is conservative

Common Form of Data Flow Equations

- Domain $\mathbb{D}$ of data flow "information"
  - sets of variables, sets of expressions, valuations, ...
- Variables D_q for $q \in Q$, with value in $\mathbb{D}$
  - D_q holds data-flow information for location q

$$D_q = \bigwedge f\left(D_{q'}\right)$$

- "Confluence" operator $\bigwedge$ on $\mathbb{D}$ to merge data flow information
  - $\cup, \cap, \otimes, \ldots$
- Functions $f : \mathbb{D} \to \mathbb{D}$ to model the effect of operations

Common Form of Data Flow Equations

- Domain $\mathbb{D}$ of data flow "information"
  - sets of variables, sets of expressions, valuations, ...
- Variables D_q for $q \in Q$, with value in $\mathbb{D}$
  - D_q holds data-flow information for location q

$$D_q = \bigwedge f\left(D_{q'}\right)$$

- "Confluence" operator $\bigwedge$ on $\mathbb{D}$ to merge data flow information
  - $\cup$, $\cap$, $\otimes$, ...
- Functions $f : \mathbb{D} \to \mathbb{D}$ to model the effect of operations

Outline — Data Flow Analysis

Partial Order

A partial order on a set L is any binary relation $\sqsubseteq \subseteq L \times L$ satisfying for all $x, y, z \in L$:

$$x \sqsubseteq x \qquad \text{(reflexivity)}$$

$$x \sqsubseteq y \ \wedge \ y \sqsubseteq x \implies x = y \qquad \text{(antisymmetry)}$$

$$x \sqsubseteq y \ \wedge \ y \sqsubseteq z \implies x \sqsubseteq z \qquad \text{(transitivity)}$$

A partially ordered set is any pair $(L, \sqsubseteq)$ where L is a set and $\sqsubseteq$ is a partial order on L.

There can be x and y in L such that $x \not\sqsubseteq y$ and $y \not\sqsubseteq x$.

Partial Order

A partial order on a set L is any binary relation $\sqsubseteq \subseteq L \times L$ satisfying for all $x, y, z \in L$:

$$x \sqsubseteq x \qquad \text{(reflexivity)}$$

$$x \sqsubseteq y \ \wedge \ y \sqsubseteq x \implies x = y \qquad \text{(antisymmetry)}$$

$$x \sqsubseteq y \ \wedge \ y \sqsubseteq z \implies x \sqsubseteq z \qquad \text{(transitivity)}$$

A partially ordered set is any pair $(L, \sqsubseteq)$ where L is a set and $\sqsubseteq$ is a partial order on L.

There can be x and y in L such that $x \not\sqsubseteq y$ and $y \not\sqsubseteq x$.

Lower and Upper Bounds

Consider a partially ordered set $(L, \sqsubseteq)$ and a subset $X \subseteq L$.

Greatest Lower Bound

A lower bound of X is any $b \in X$ such that $b \sqsubseteq x$ for all $x \in X$.

A greatest lower bound of X is any $glb \in X$ such that:

1. glb is a lower bound of X,
2. $glb \sqsupseteq b$ for any lower bound b of X.

If X has a greatest lower bound, then it is *unique* and written $\bigsqcap X$.

Lower and Upper Bounds

Consider a partially ordered set $(L, \sqsubseteq)$ and a subset $X \subseteq L$.

Greatest Lower Bound

A lower bound of X is any $b \in X$ such that $b \sqsubseteq x$ for all $x \in X$.

A greatest lower bound of X is any $glb \in X$ such that: [. . .]

If X has a greatest lower bound, then it is *unique* and written $\bigsqcap X$.

Least Upper Bound

An upper bound of X is any $b \in X$ such that $b \sqsupseteq x$ for all $x \in X$.

A least upper bound of X is any $lub \in X$ such that:

1. lub is an upper bound of X,
2. $lub \sqsubseteq b$ for any upper bound b of X.

If X has a least upper bound, then it is *unique* and written $\bigsqcup X$.

Lower and Upper Bounds: Examples

$(\mathbb{R}, \leq)$

$$\bigsqcup \left\{0, \sqrt{2}, 4\right\} = 4 \qquad \bigsqcap \left\{\frac{1}{2^n} \,\middle|\, n \in \mathbb{N}\right\} = 0$$

But $\{\ldots, -2, -1, 0, 1, 2, \ldots\}$ has no upper bound and no lower bound.

$(\mathcal{P}(\{-1, 0, 1\}), \subseteq)$

$$\bigsqcup \{\{0\}, \{1\}\} = \{0, 1\}$$

$$\bigsqcup \{\{-1\}, \{0, 1\}\} = \{-1, 0, 1\}$$

$$\bigsqcap \{\{-1, 0\}, \{0, 1\}\} = \{0\}$$

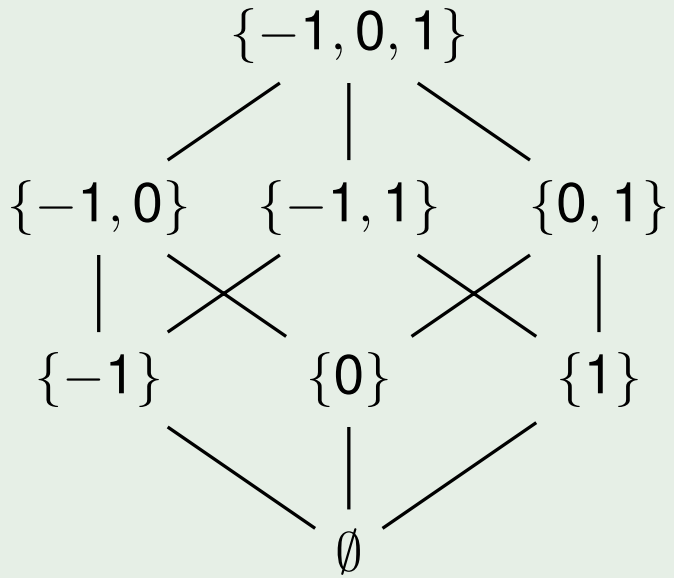
Lower and Upper Bounds: Examples

$(\mathbb{R}, \leq)$

$$\bigsqcup \left\{0, \sqrt{2}, 4\right\} = 4 \qquad \bigsqcap \left\{\frac{1}{2^n} \,\middle|\, n \in \mathbb{N}\right\} = 0$$

But $\{\ldots, -2, -1, 0, 1, 2, \ldots\}$ has no upper bound and no lower bound.

$(\mathcal{P}(\{-1, 0, 1\}), \subseteq)$



$$\bigsqcup \{\{0\}, \{1\}\} = \{0, 1\}$$

$$\bigsqcup \{\{-1\}, \{0, 1\}\} = \{-1, 0, 1\}$$

$$\bigsqcap \{\{-1, 0\}, \{0, 1\}\} = \{0\}$$

Complete Lattice

Definition

A lattice is any partially ordered set $(L, \sqsubseteq)$ where every finite subset $X \subseteq L$ has a greatest lower bound and a least upper bound.

Definition

A complete lattice is any partially ordered set $(L, \sqsubseteq)$ where every subset $X \subseteq L$ has a greatest lower bound and a least upper bound.

The least element $\bot$ and greatest element $\top$ are defined by:

$$\bot = \bigsqcap L = \bigsqcup \emptyset \qquad \top = \bigsqcup L = \bigsqcap \emptyset$$

Example

$(\mathbb{R}, \leq)$ is a lattice, but it is not a complete lattice.

Complete Lattice

Definition

A lattice is any partially ordered set $(L, \sqsubseteq)$ where every finite subset $X \subseteq L$ has a greatest lower bound and a least upper bound.

Definition

A complete lattice is any partially ordered set $(L, \sqsubseteq)$ where every subset $X \subseteq L$ has a greatest lower bound and a least upper bound.

The least element $\bot$ and greatest element $\top$ are defined by:

$$\bot = \bigsqcap L = \bigsqcup \emptyset \qquad \top = \bigsqcup L = \bigsqcap \emptyset$$

Example

$(\mathbb{R}, \leq)$ is a lattice, but it is not a complete lattice.

Complete Lattice

Definition

A lattice is any partially ordered set $(L, \sqsubseteq)$ where every finite subset $X \subseteq L$ has a greatest lower bound and a least upper bound.

Definition

A complete lattice is any partially ordered set $(L, \sqsubseteq)$ where every subset $X \subseteq L$ has a greatest lower bound and a least upper bound.

The least element $\bot$ and greatest element $\top$ are defined by:

$$\bot = \bigsqcap L = \bigsqcup \emptyset \qquad \top = \bigsqcup L = \bigsqcap \emptyset$$

Example

$(\mathbb{R}, \leq)$ is a lattice, but it is not a complete lattice.

Fixpoints

Let $f : L \to L$ be a function on a partially ordered set $(L, \sqsubseteq)$.

Definition

A fixpoint of f is any $x \in L$ such that $f(x) = x$.

Definition

A least fixpoint of f is any $lfp \in X$ such that:

1. lfp is a fixpoint of f,
2. $lfp \sqsubseteq x$ for any fixpoint x of f.

If f has a least fixpoint, then it is *unique* and written $\mathrm{lfp}(f)$.

Definition

A greatest fixpoint of f is any $gfp \in X$ such that:

1. gfp is a fixpoint of f,
2. $gfp \sqsupseteq x$ for any fixpoint x of f.

Fixpoints

Let $f : L \to L$ be a function on a partially ordered set $(L, \sqsubseteq)$.

Definition

A fixpoint of f is any $x \in L$ such that $f(x) = x$.

Definition

A least fixpoint of f is any $lfp \in X$ such that:

1. lfp is a fixpoint of f,
2. $lfp \sqsubseteq x$ for any fixpoint x of f.

If f has a least fixpoint, then it is *unique* and written $\mathrm{lfp}(f)$.

Definition

A greatest fixpoint of f is any $gfp \in X$ such that:

1. gfp is a fixpoint of f,
2. $gfp \sqsupseteq x$ for any fixpoint x of f.

Fixpoints

Let $f : L \to L$ be a function on a partially ordered set $(L, \sqsubseteq)$.

Definition

A fixpoint of f is any $x \in L$ such that $f(x) = x$.

Definition

A least fixpoint of f is any $lfp \in X$ such that: [...]

If f has a least fixpoint, then it is *unique* and written $\mathrm{lfp}(f)$.

Definition

A greatest fixpoint of f is any $gfp \in X$ such that:

1. gfp is a fixpoint of f,
2. $gfp \sqsupseteq x$ for any fixpoint x of f.

If f has a greatest fixpoint, then it is *unique* and written $\mathrm{gfp}(f)$.

Knaster-Tarski Fixpoint Theorem

A function $f : L \longrightarrow L$ on a partially ordered set $(L, \sqsubseteq)$ is monotonic if for all $x, y \in L$:

$$x \sqsubseteq y \implies f(x) \sqsubseteq f(y)$$

Theorem

Every monotonic function f on a complete lattice $(L, \sqsubseteq)$ has a least fixpoint $\mathrm{lfp}(f)$ *and a greatest fixpoint* $\mathrm{gfp}(f)$*. Moreover:*

$$\mathrm{lfp}(f) = \bigsqcap \{x \in L \mid f(x) \sqsubseteq x\}$$

$$\mathrm{gfp}(f) = \bigsqcup \{x \in L \mid f(x) \sqsupseteq x\}$$

Order Duality

If $(L, \sqsubseteq)$ is a partially ordered set then so is $(L, \sqsupseteq)$.

If $(L, \sqsubseteq)$ is a complete lattice then so is $(L, \sqsupseteq)$.

$$\bigsqcap_{(L,\sqsupseteq)} = \bigsqcup_{(L,\sqsubseteq)} \qquad \bot_{(L,\sqsupseteq)} = \top_{(L,\sqsubseteq)}$$

$$\bigsqcup_{(L,\sqsupseteq)} = \bigsqcap_{(L,\sqsubseteq)} \qquad \top_{(L,\sqsupseteq)} = \bot_{(L,\sqsubseteq)}$$

For any monotonic function $f : L \longrightarrow L$ on a complete lattice $(L, \sqsubseteq)$,

$$\mathrm{lfp}_{(L,\sqsubseteq)}(f) = \mathit{gfp}_{(L,\sqsupseteq)}(f)$$

$$\mathrm{gfp}_{(L,\sqsubseteq)}(f) = \mathit{lfp}_{(L,\sqsupseteq)}(f)$$

We shall focus on least fixpoints.

Order Duality

If $(L, \sqsubseteq)$ is a partially ordered set then so is $(L, \sqsupseteq)$.

If $(L, \sqsubseteq)$ is a complete lattice then so is $(L, \sqsupseteq)$.

$$\bigsqcap_{(L,\sqsupseteq)} = \bigsqcup_{(L,\sqsubseteq)} \qquad \bot_{(L,\sqsupseteq)} = \top_{(L,\sqsubseteq)}$$

$$\bigsqcup_{(L,\sqsupseteq)} = \bigsqcap_{(L,\sqsubseteq)} \qquad \top_{(L,\sqsupseteq)} = \bot_{(L,\sqsubseteq)}$$

For any monotonic function $f : L \to L$ on a complete lattice $(L, \sqsubseteq)$,

$$\mathrm{lfp}_{(L,\sqsubseteq)}(f) = \mathit{gfp}_{(L,\sqsupseteq)}(f)$$

$$\mathrm{gfp}_{(L,\sqsubseteq)}(f) = \mathit{lfp}_{(L,\sqsupseteq)}(f)$$

We shall focus on least fixpoints.

Ascending Chain Condition

An ascending chain in a partially ordered set $(L, \sqsubseteq)$ is any infinite sequence $x_0, x_1, \ldots$ of elements of L satisfying $x_i \sqsubseteq x_{i+1}$ for all $i \in \mathbb{N}$.

A partially ordered set $(L, \sqsubseteq)$ satisfies the ascending chain condition if every ascending chain $x_0 \sqsubseteq x_1 \sqsubseteq \cdots$ of elements of L is eventually stationary.

Examples

$(\mathbb{R}, \leq)$ does not satisfy the ascending chain condition.

$(\mathbb{N}, \geq)$ satisfies the ascending chain condition.

Ascending Chain Condition

An ascending chain in a partially ordered set $(L, \sqsubseteq)$ is any infinite sequence $x_0, x_1, \ldots$ of elements of L satisfying $x_i \sqsubseteq x_{i+1}$ for all $i \in \mathbb{N}$.

A partially ordered set $(L, \sqsubseteq)$ satisfies the ascending chain condition if every ascending chain $x_0 \sqsubseteq x_1 \sqsubseteq \cdots$ of elements of L is eventually stationary.

Examples

$(\mathbb{R}, \leq)$ does not satisfy the ascending chain condition.

$(\mathbb{N}, \geq)$ satisfies the ascending chain condition.

Kleene Iteration

Consider a partially ordered set $(L, \sqsubseteq)$ and $f : L \to L$ monotonic.

The Kleene iteration $\left(f^i(\bot)\right)_{i \in \mathbb{N}}$ is an ascending chain:

$$\bot \sqsubseteq f(\bot) \sqsubseteq \cdots \sqsubseteq f^i(\bot) \sqsubseteq f^{i+1}(\bot) \sqsubseteq \cdots$$

For every $k \in \mathbb{N}$, if $f^k(\bot) = f^{k+1}(\bot)$ then $f^k(\bot)$ is the least fixpoint of f.

```
LFP(f : L → L)
x ← ⊥
repeat
    t ← x
    x ← f(x)
until t = x
return x
```

Correction and termination

1. For every monotonic f, if LFP(f) terminates then it returns lfp(f).
2. If L satisfies the ascending chain condition then LFP(f) always terminates (on monotonic f).

Kleene Iteration

Consider a partially ordered set $(L, \sqsubseteq)$ and $f : L \to L$ monotonic.

The Kleene iteration $\left(f^i(\bot)\right)_{i \in \mathbb{N}}$ is an ascending chain:

$$\bot \sqsubseteq f(\bot) \sqsubseteq \cdots \sqsubseteq f^i(\bot) \sqsubseteq f^{i+1}(\bot) \sqsubseteq \cdots$$

For every $k \in \mathbb{N}$, if $f^k(\bot) = f^{k+1}(\bot)$ then $f^k(\bot)$ is the least fixpoint of f.

LFP($\mathtt{f} : L \to L$)
```
x ← ⊥
repeat
    t ← x
    x ← f(x)
until t = x
return x
```

Correction and termination

1. For every monotonic `f`, if LFP(`f`) terminates then it returns lfp(f).
2. If L satisfies the ascending chain condition then LFP(`f`) always terminates (on monotonic `f`).

Constructing Complete Lattices: Power Set

For any set S, the pair $(\mathcal{P}(S), \sqsubseteq)$ is a complete lattice, where $\sqsubseteq = \subseteq$.

$\bigsqcap$, $\bigsqcup$, $\bot$ and $\top$ satisfy:

$$\bigsqcap = \bigcap \qquad \bot = \emptyset$$

$$\bigsqcup = \bigcup \qquad \top = S$$

If S is finite then $(\mathcal{P}(S), \sqsubseteq)$ satisfies the ascending chain condition.

Constructing Complete Lattices: Functions

For any set S and complete lattice $(L, \sqsubseteq)$, the pair $(S \to L, \sqsubseteq)$ is a complete lattice, where $\sqsubseteq$ is defined by:

$$f \sqsubseteq g \quad \text{if} \quad f(x) \sqsubseteq g(x) \quad \text{for all } x \in S$$

$\bigsqcap$, $\bigsqcup$, $\bot$ and $\top$ satisfy:

$$\bigsqcap X = \lambda x . \bigsqcap \{f(x) \mid f \in X\} \qquad \bot = \lambda x . \bot$$

$$\bigsqcup X = \lambda x . \bigsqcup \{f(x) \mid f \in X\} \qquad \top = \lambda x . \top$$

If S is finite and $(L, \sqsubseteq)$ satisfies the ascending chain condition then $(S \to L, \sqsubseteq)$ satisfies the ascending chain condition.

Outline — Data Flow Analysis

Common Form of Data Flow Equations (Recall)

- Domain $\mathbb{D}$ of data flow "information"
  - sets of variables, sets of expressions, valuations, ...
- Variables D_q for $q \in Q$, with value in $\mathbb{D}$
  - D_q holds data-flow information for location q

$$D_q \quad = \quad \bigwedge f\left(D_{q'}\right)$$

- "Confluence" operator $\bigwedge$ on $\mathbb{D}$ to merge data flow information
  - $\cup$, $\cap$, $\otimes$, ...
- Functions $f : \mathbb{D} \to \mathbb{D}$ to model the effect of operations

Monotone Frameworks

Monotone Framework

- Complete lattice $(L, \sqsubseteq)$ of data flow facts
- Set $\mathcal{F}$ of monotonic transfer functions $f : L \rightarrow L$

Partial order $\sqsubseteq$ compares the precision of data flow facts:

- $\phi \sqsubseteq \psi$ means that ϕ is more precise than ψ.
- $\bigsqcup X$ is the most precise fact consistent with all facts $\phi \in X$.

Conservative Approximation

$\phi \sqsubseteq \psi$ means that ψ soundly approximates ϕ.

If $\phi \sqsubseteq \psi$ then it is sound, but less precise, to replace ϕ by ψ.

Monotone Frameworks

Monotone Framework

- Complete lattice $(L, \sqsubseteq)$ of data flow facts
- Set $\mathcal{F}$ of monotonic transfer functions $f : L \to L$

Partial order $\sqsubseteq$ compares the precision of data flow facts:

- $\phi \sqsubseteq \psi$ means that ϕ is more precise than ψ.
- $\bigsqcup X$ is the most precise fact consistent with all facts $\phi \in X$.

Conservative Approximation

$\phi \sqsubseteq \psi$ means that ψ soundly approximates ϕ.

If $\phi \sqsubseteq \psi$ then it is sound, but less precise, to replace ϕ by ψ.

Monotone Frameworks

Monotone Framework

- Complete lattice $(L, \sqsubseteq)$ of data flow facts
- Set $\mathcal{F}$ of monotonic transfer functions $f : L \rightarrow L$

Partial order $\sqsubseteq$ compares the precision of data flow facts:

- $\phi \sqsubseteq \psi$ means that ϕ is more precise than ψ.
- $\bigsqcup X$ is the most precise fact consistent with all facts $\phi \in X$.

Conservative Approximation

$\phi \sqsubseteq \psi$ means that ψ soundly approximates ϕ.

If $\phi \sqsubseteq \psi$ then it is sound, but less precise, to replace ϕ by ψ.

Data Flow Facts: Example for Live Variables Analysis

Semantic Definition of Liveness

A variable x is live at location q if there exists an execution path starting from q where x is used before it is modified.

Consider a control flow automaton with variables $\mathtt{X} = \{\mathtt{x}, \mathtt{y}, \mathtt{z}\}$.

Complete lattice $(L, \sqsubseteq)$ of data flow facts: $(\mathcal{P}(\mathtt{X}), \subseteq)$

The fact $\{\mathtt{x}, \mathtt{z}\}$ means: *the variables that are live are among* $\{\mathtt{x}, \mathtt{z}\}$.

i.e. *the variable* $\mathtt{y}$ *is not live.*

The fact $\{\mathtt{x}\}$ is more precise than $\{\mathtt{x}, \mathtt{z}\}$, but incomparable with $\{\mathtt{y}\}$.

The fact $\{\mathtt{x}, \mathtt{z}\}$ soundly approximates the fact $\{\mathtt{x}\}$.

Data Flow Instances

Data Flow Instance

- Monotone framework $\langle (L, \sqsubseteq), \mathcal{F} \rangle$
- Control flow automaton $\langle Q, q_{in}, q_{out}, \mathtt{X}, \rightarrow \rangle$
- Transfer mapping $f : \mathtt{Op} \rightarrow \mathcal{F}$
- Initial data flow value $\imath \in L$

Notation for transfer mapping: $f_{\mathtt{op}}$ instead of $f(\mathtt{op})$

Two possible directions for data flow analysis: forward and backward

Transfer functions $f_{\mathtt{op}}$ must be defined in accordance with the direction of the analysis.

Data Flow Instances

Data Flow Instance

- Monotone framework $\langle (L, \sqsubseteq), \mathcal{F} \rangle$
- Control flow automaton $\langle Q, q_{in}, q_{out}, \mathtt{X}, \rightarrow \rangle$
- Transfer mapping $f : \mathtt{Op} \rightarrow \mathcal{F}$
- Initial data flow value $\imath \in L$

Notation for transfer mapping: $f_{\mathtt{op}}$ instead of $f(\mathtt{op})$

Two possible directions for data flow analysis: forward and backward

Transfer functions $f_{\mathtt{op}}$ must be defined in accordance with the direction of the analysis.

Data Flow Equations

Consider a data flow instance $\langle (L, \sqsubseteq), \mathcal{F}, Q, q_{in}, q_{out}, \mathtt{X}, \rightarrow, f, \imath \rangle$.

System of equations: variables A_q for $q \in Q$, with $A_q \in L$

Forward Analysis

$$A_q = I_q \sqcup \bigsqcup_{q' \xrightarrow{\mathrm{op}} q} f_{\mathrm{op}}(A_{q'}) \qquad I_q = \begin{cases} \imath & \text{if } q = q_{in} \\ \bot & \text{otherwise} \end{cases}$$

Backward Analysis

$$A_q = I_q \sqcup \bigsqcup_{q \xrightarrow{\mathrm{op}} q'} f_{\mathrm{op}}(A_{q'}) \qquad I_q = \begin{cases} \imath & \text{if } q = q_{out} \\ \bot & \text{otherwise} \end{cases}$$

Data Flow Equations

Consider a data flow instance $\langle (L, \sqsubseteq), \mathcal{F}, Q, q_{in}, q_{out}, \mathtt{X}, \rightarrow, f, \imath \rangle$.

System of equations: variables A_q for $q \in Q$, with $A_q \in L$

Forward Analysis

$$A_q = I_q \sqcup \bigsqcup_{q' \xrightarrow{\mathrm{op}} q} f_{\mathrm{op}}(A_{q'}) \qquad I_q = \begin{cases} \imath & \text{if } q = q_{in} \\ \bot & \text{otherwise} \end{cases}$$

Backward Analysis

$$A_q = I_q \sqcup \bigsqcup_{q \xrightarrow{\mathrm{op}} q'} f_{\mathrm{op}}(A_{q'}) \qquad I_q = \begin{cases} \imath & \text{if } q = q_{out} \\ \bot & \text{otherwise} \end{cases}$$

Minimal Fixpoint (MFP) Solution

The system of data flow equations may have several solutions...

We are interested in the "least solution" to the data flow equations.

Complete lattice $(L, \sqsubseteq)$ extended to $(Q \to L, \sqsubseteq)$

The forward minimal fixpoint solution $\overrightarrow{\text{MFP}}$ of the data flow instance is the least fixpoint of the monotonic function $\overrightarrow{\Delta}$ on $(Q \to L)$:

$$\overrightarrow{\Delta}(a) = \lambda q . \begin{cases} \imath \sqcup \bigsqcup_{q' \xrightarrow{\text{op}} q} f_{\text{op}}(a(q')) & \text{if } q = q_{in} \\ \bigsqcup_{q' \xrightarrow{\text{op}} q} f_{\text{op}}(a(q')) & \text{otherwise} \end{cases}$$

Minimal Fixpoint (MFP) Solution

The system of data flow equations may have several solutions...

We are interested in the "least solution" to the data flow equations.

Complete lattice $(L, \sqsubseteq)$ extended to $(Q \to L, \sqsubseteq)$

The forward minimal fixpoint solution $\overrightarrow{\text{MFP}}$ of the data flow instance is the least fixpoint of the monotonic function $\overrightarrow{\Delta}$ on $(Q \to L)$:

$$\overrightarrow{\Delta}(a) = \lambda q . \begin{cases} \imath \sqcup \bigsqcup_{q' \xrightarrow{\text{op}} q} f_{\text{op}}(a(q')) & \text{if } q = q_{in} \\ \bigsqcup_{q' \xrightarrow{\text{op}} q} f_{\text{op}}(a(q')) & \text{otherwise} \end{cases}$$

Minimal Fixpoint (MFP) Solution

The system of data flow equations may have several solutions...

We are interested in the "least solution" to the data flow equations.

Complete lattice $(L, \sqsubseteq)$ extended to $(Q \to L, \sqsubseteq)$

The backward minimal fixpoint solution $\overleftarrow{\text{MFP}}$ of the data flow instance is the least fixpoint of the monotonic function $\overleftarrow{\Delta}$ on $(Q \to L)$:

$$\overleftarrow{\Delta}(a) = \lambda q . \begin{cases} \iota \sqcup \bigsqcup_{q \xrightarrow{\text{op}} q'} f_{\text{op}}(a(q')) & \text{if } q = q_{out} \\ \bigsqcup_{q \xrightarrow{\text{op}} q'} f_{\text{op}}(a(q')) & \text{otherwise} \end{cases}$$

Constraint-Based Formulation

Consider a data flow instance $\langle (L, \sqsubseteq), \mathcal{F}, Q, q_{in}, q_{out}, \mathrm{X}, \rightarrow, f, \imath \rangle$.

Constraint system: variables A_q for $q \in Q$, with $A_q \in L$

Forward Analysis

$$\overrightarrow{(CS)} \quad \begin{cases} A_{q_{in}} & \sqsupseteq & \imath \\ A_{q'} & \sqsupseteq & f_{\mathrm{op}}(A_q) \quad \text{for each } q \xrightarrow{\mathrm{op}} q' \end{cases}$$

By Knaster-Tarski Fixpoint Theorem,

$$\overrightarrow{\mathrm{MFP}} \quad = \quad \bigsqcap \left\{ a \in Q \rightarrow L \;\middle|\; a \models \overrightarrow{(CS)} \right\}$$

Any solution to $\overrightarrow{(CS)}$ is a sound approximation of $\overrightarrow{\mathrm{MFP}}$.

Constraint-Based Formulation

Consider a data flow instance $\langle (L, \sqsubseteq), \mathcal{F}, Q, q_{in}, q_{out}, \mathtt{X}, \rightarrow, f, \imath \rangle$.

Constraint system: variables A_q for $q \in Q$, with $A_q \in L$

Backward Analysis

$$\overleftarrow{(CS)} \quad \begin{cases} A_{q_{out}} & \sqsupseteq & \imath \\ A_{q'} & \sqsupseteq & f_{\mathrm{op}}(A_q) \quad \text{for each } q' \xrightarrow{\mathrm{op}} q \end{cases}$$

By Knaster-Tarski Fixpoint Theorem,

$$\overleftarrow{\mathrm{MFP}} \quad = \quad \bigsqcap \left\{ a \in Q \rightarrow L \;\middle|\; a \models \overleftarrow{(CS)} \right\}$$

Any solution to $\overleftarrow{(CS)}$ is a sound approximation of $\overleftarrow{\mathrm{MFP}}$.

Live Variables Analysis (Revisited)

Control Flow Automaton: $\langle Q, q_{in}, q_{out}, \mathtt{X}, \rightarrow \rangle$

Monotone Framework

- Complete lattice $(L, \sqsubseteq)$ of data flow facts: $(\mathcal{P}(\mathtt{X}), \subseteq)$
- Set $\mathcal{F}$ of monotonic transfer functions:

$$\mathcal{F} \quad = \quad \{\lambda\, \phi\, .\, \mathit{gen} \cup (\phi \setminus \mathit{kill}) \mid \mathit{gen}, \mathit{kill} \in L\}$$

Data Flow Instance

- Initial data flow value: $\emptyset$
- Transfer mapping: $f_{\mathtt{op}}(\phi) \;=\; \mathit{Gen}_{\mathtt{op}} \;\cup\; (\phi \setminus \mathit{Kill}_{\mathtt{op}})$

Backward analysis

Available Expressions Analysis (Revisited)

Control Flow Automaton: $\langle Q, q_{in}, q_{out}, \mathtt{X}, \rightarrow \rangle$

Monotone Framework

- Complete lattice $(L, \sqsubseteq)$ of data flow facts: $(\mathcal{P}(\mathit{SubExp}(\rightarrow)), \supseteq)$
- Set $\mathcal{F}$ of monotonic transfer functions:

$$\mathcal{F} \quad = \quad \{\lambda\, \phi\, .\, \mathit{gen} \cup (\phi \setminus \mathit{kill}) \mid \mathit{gen}, \mathit{kill} \in L\}$$

Data Flow Instance

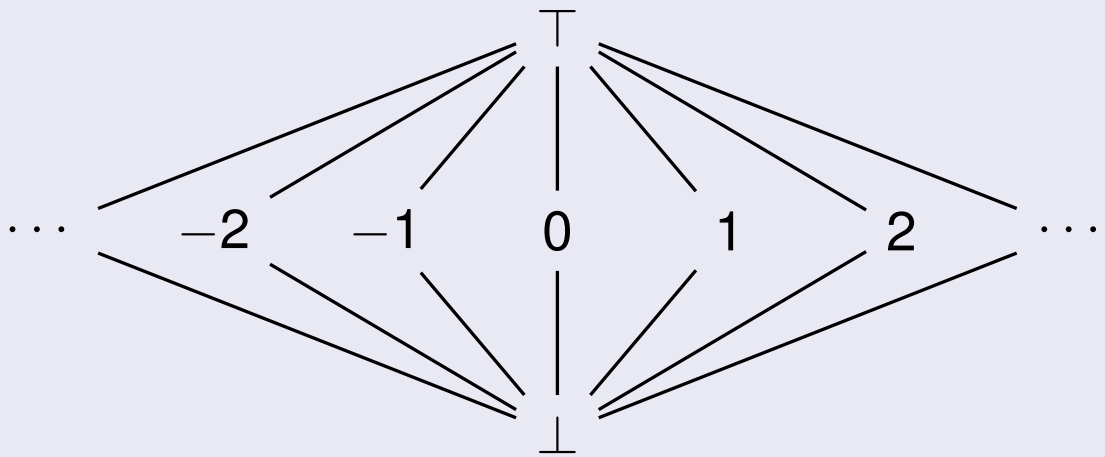
- Initial data flow value: $\emptyset$
- Transfer mapping: $f_{\mathtt{op}}(\phi) \;=\; \mathit{Gen}_{\mathtt{op}} \;\cup\; (\phi \setminus \mathit{Kill}_{\mathtt{op}})$

Forward analysis

Constant Propagation Analysis (Revisited)

Control Flow Automaton: $\langle Q, q_{in}, q_{out}, \mathtt{X}, \rightarrow \rangle$

Constant Propagation Lattice for a Single Variable



$(\mathbb{R} \cup \{\bot, \top\}, \sqsubseteq)$

ϕ	*Meaning*
$\top$	$\mathbb{R}$
$r \in \mathbb{R}$	$\{r\}$
$\bot$	$\emptyset$

Monotone Framework

- Complete lattice $(L, \sqsubseteq)$ of data flow facts: $(\mathtt{X} \rightarrow (\mathbb{R} \cup \{\bot, \top\}), \sqsubseteq)$
- Set $\mathcal{F}$ defined as the set of all monotonic transfer functions on L.

Constant Propagation Analysis (Revisited)

Control Flow Automaton: $\langle Q, q_{in}, q_{out}, \mathbb{X}, \rightarrow \rangle$

Monotone Framework

- Complete lattice $(L, \sqsubseteq)$ of data flow facts: $(\mathbb{X} \rightarrow (\mathbb{R} \cup \{\bot, \top\}), \sqsubseteq)$
- Set $\mathcal{F}$ defined as the set of all monotonic transfer functions on L.

Data Flow Instance

- Initial data flow value: $\top$
- Transfer mapping:

$$f_{x:=e}(\phi) = \lambda y . \begin{cases} \phi(y) & \text{if } y \neq x \\ [\![e]\!]_\phi & \text{if } y = x \end{cases} \qquad f_g(\phi) = \phi$$

Forward analysis

Constant Propagation Analysis (Revisited)

Extension of $[\![e]\!]$ to valuations in $\mathbb{X} \to (\mathbb{R} \cup \{\bot, \top\})$

For $r \in \mathbb{R} \cup \{\top\}$

$$\begin{array}{rcl} \top + r = r + \top & = & \top \\ \top - r = r - \top & = & \top \\ \top \times r = r \times \top & = & \top \end{array}$$

For $r \in \mathbb{R} \cup \{\bot, \top\}$

$$\begin{array}{rcl} \bot + r = r + \bot & = & \bot \\ \bot - r = r - \bot & = & \bot \\ \bot \times r = r \times \bot & = & \bot \end{array}$$

Expressions: $[\![e]\!]_v$

$$\begin{array}{rcll} [\![c]\!]_v & = & c & [c \in \mathbb{Q}] \\ [\![x]\!]_v & = & v(x) & [x \in \mathbb{X}] \\ [\![e_1 + e_2]\!]_v & = & [\![e_1]\!]_v + [\![e_2]\!]_v & \\ [\![e_1 - e_2]\!]_v & = & [\![e_1]\!]_v - [\![e_2]\!]_v & \\ [\![e_1 * e_2]\!]_v & = & [\![e_1]\!]_v \times [\![e_2]\!]_v & \end{array}$$

Constant Propagation Analysis (Revisited)

Extension of $[\![e]\!]$ to valuations in $\mathbb{X} \to (\mathbb{R} \cup \{\bot, \top\})$

For $r \in \mathbb{R} \cup \{\top\}$

$$\begin{array}{lllll} \top + r & = & r + \top & = & \top \\ \top - r & = & r - \top & = & \top \\ \top \times r & = & r \times \top & = & \top \end{array}$$

For $r \in \mathbb{R} \cup \{\bot, \top\}$

$$\begin{array}{lllll} \bot + r & = & r + \bot & = & \bot \\ \bot - r & = & r - \bot & = & \bot \\ \bot \times r & = & r \times \bot & = & \bot \end{array}$$

Expressions: $[\![e]\!]_v$

$$\begin{array}{rcll} [\![c]\!]_v & = & c & [c \in \mathbb{Q}] \\ [\![x]\!]_v & = & v(x) & [x \in \mathbb{X}] \\ [\![e_1 + e_2]\!]_v & = & [\![e_1]\!]_v + [\![e_2]\!]_v & \\ [\![e_1 - e_2]\!]_v & = & [\![e_1]\!]_v - [\![e_2]\!]_v & \\ [\![e_1 * e_2]\!]_v & = & [\![e_1]\!]_v \times [\![e_2]\!]_v & \end{array}$$

(Forward) MFP Computation by Kleene Iteration

Consider a data flow instance $\langle (L, \sqsubseteq), \mathcal{F}, Q, q_{in}, q_{out}, \mathtt{X}, \rightarrow, f, \imath \rangle$.

```
a ← λq.⊥
repeat
    b ← a
    a ← Δ⃗(a)
until b = a
return a
```

```
foreach q ∈ Q
    a[q] ← ⊥
a[q_in] ← ι
repeat
    foreach q ∈ Q
        b[q] ← a[q]
    foreach q ∈ Q
        a[q] ← ⨆_{q' —op→ q} f_op(b[q'])
until (∀ q ∈ Q · b[q] = a[q])
return a
```

Correction and termination

1. Returns $\overrightarrow{\mathrm{MFP}}$ when it terminates
2. Always terminates when $(L, \sqsubseteq)$ satisfies the ascending chain condition

(Forward) MFP Computation by Kleene Iteration

Consider a data flow instance $\langle (L, \sqsubseteq), \mathcal{F}, Q, q_{in}, q_{out}, \mathtt{X}, \rightarrow, f, \imath \rangle$.

```
a ← λq.⊥
repeat
    b ← a
    a ← Δ⃗(a)
until b = a
return a
```

Correction and termination

1. Returns $\overrightarrow{\mathrm{MFP}}$ when it terminates
2. Always terminates when $(L, \sqsubseteq)$ satisfies the ascending chain condition

```
foreach q ∈ Q
    a[q] ← ⊥
a[q_in] ← ι
repeat
    foreach q ∈ Q
        b[q] ← a[q]
    foreach q ∈ Q
        a[q] ← ⨆_{q' —op→ q} f_op(b[q'])
until (∀ q ∈ Q · b[q] = a[q])
return a
```

(Forward) MFP Computation by Kleene Iteration

Consider a data flow instance $\langle (L, \sqsubseteq), \mathcal{F}, Q, q_{in}, q_{out}, \mathtt{X}, \rightarrow, f, \imath \rangle$.

$\mathtt{a} \leftarrow \lambda\, q\,.\, \bot$
repeat
  $\mathtt{b} \leftarrow \mathtt{a}$
  $\mathtt{a} \leftarrow \overrightarrow{\Delta}(a)$
until $\mathtt{b} = \mathtt{a}$
return $\mathtt{a}$

Correction and termination

1. Returns $\overrightarrow{\mathrm{MFP}}$ when it terminates
2. Always terminates when $(L, \sqsubseteq)$ satisfies the ascending chain condition

foreach $q \in Q$
  $\mathtt{a}[q] \leftarrow \bot$
$\mathtt{a}[q_{in}] \leftarrow \imath$
repeat
  foreach $q \in Q$
    $\mathtt{b}[q] \leftarrow \mathtt{a}[q]$
  foreach $q \in Q$
    $\mathtt{a}[q] \leftarrow \bigsqcup_{q' \xrightarrow{\mathrm{op}} q} f_{\mathrm{op}}(\mathtt{b}[q'])$
until $(\forall\, q \in Q \cdot \mathtt{b}[q] = \mathtt{a}[q])$
return $\mathtt{a}$

We can improve!

(Forward) MFP Computation by Round-Robin Iteration

Consider a data flow instance $\langle (L, \sqsubseteq), \mathcal{F}, Q, q_{in}, q_{out}, \mathtt{X}, \rightarrow, f, \imath \rangle$.

```
foreach q ∈ Q
    a[q] ← ⊥
a[q_in] ← ι
do
    change ← false
    foreach q --op--> q'
        new ← f_op(a[q])
        if new ⋢ a[q']
            a[q'] ← a[q'] ⊔ new
            change ← true
while change
return a
```

The foreach loop iterates over transitions in $\rightarrow$.

Propagation of facts

- benefits from previous propagations
- records whether there was a change

Correct and always faster than Kleene iteration

(Forward) MFP Computation by Round-Robin Iteration

Consider a data flow instance $\langle (L, \sqsubseteq), \mathcal{F}, Q, q_{in}, q_{out}, \mathtt{X}, \rightarrow, f, \imath \rangle$.

```
foreach q ∈ Q
    a[q] ← ⊥
a[q_in] ← ι
do
    change ← false
    foreach q --op--> q'
        new ← f_op(a[q])
        if new ⋢ a[q']
            a[q'] ← a[q'] ⊔ new
            change ← true
while change
return a
```

The foreach loop iterates over transitions in $\rightarrow$.

Propagation of facts

- benefits from previous propagations
- records whether there was a change

Correct and always faster than Kleene iteration

(Forward) MFP Computation by Worklist Iteration

```
wl ← nil
foreach q' --op--> q
    wl ← cons((q, op, q'), wl)
foreach q ∈ Q
    a[q] ← ⊥
a[q_in] ← ι
while wl ≠ nil
    (q, op, q') ← head(wl)
    wl ← tail(wl)
    new ← f_op(a[q])
        if new ⋢ a[q']
            a[q'] ← a[q] ⊔ new
            foreach q' --op'--> q''
                wl ← cons((q', op', q''), wl)
return a
```

Vs Round-Robin

- ☺ Less computations
- ☹ Overhead

Worklist structures

- LIFO
- FIFO
- Set
- ...

(Forward) MFP Computation by Worklist Iteration

```
wl ← nil
foreach q' --op--> q
    wl ← cons((q, op, q'), wl)
foreach q ∈ Q
    a[q] ← ⊥
a[q_in] ← ι
while wl ≠ nil
    (q, op, q') ← head(wl)
    wl ← tail(wl)
    new ← f_op(a[q])
        if new ⋢ a[q']
            a[q'] ← a[q] ⊔ new
            foreach q' --op'--> q''
                wl ← cons((q', op', q''), wl)
return a
```

Vs Round-Robin

☺ Less computations

☹ Overhead

Worklist structures

- LIFO
- FIFO
- Set
- ...

Optimization of MFP Computation with SCCs

1. Decompose control flow automaton into strongly connected components
2. Transitions between SCCs induce a partial order between SCCs
3. Compute the MFP solution component after component, following the partial order between SCCs

This optimization often pays off in practice

Further optimizations are possible...

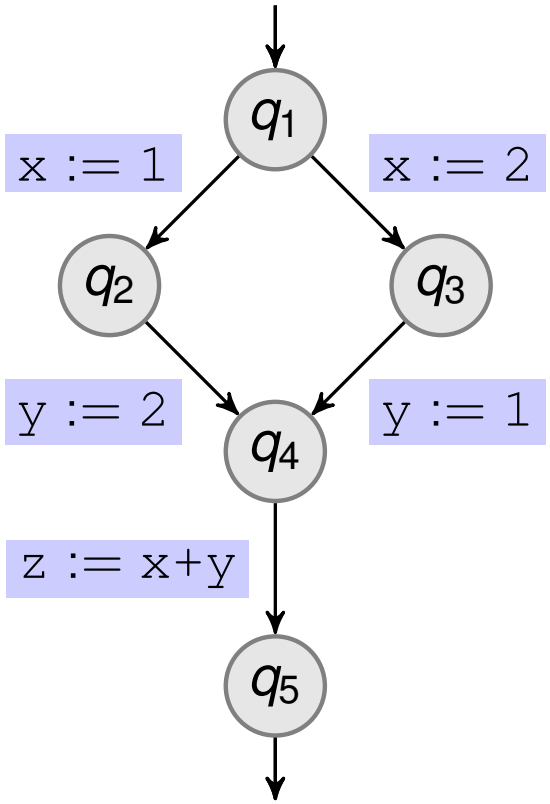
Optimization of MFP Computation with SCCs

1. Decompose control flow automaton into strongly connected components
2. Transitions between SCCs induce a partial order between SCCs
3. Compute the MFP solution component after component, following the partial order between SCCs

This optimization often pays off in practice

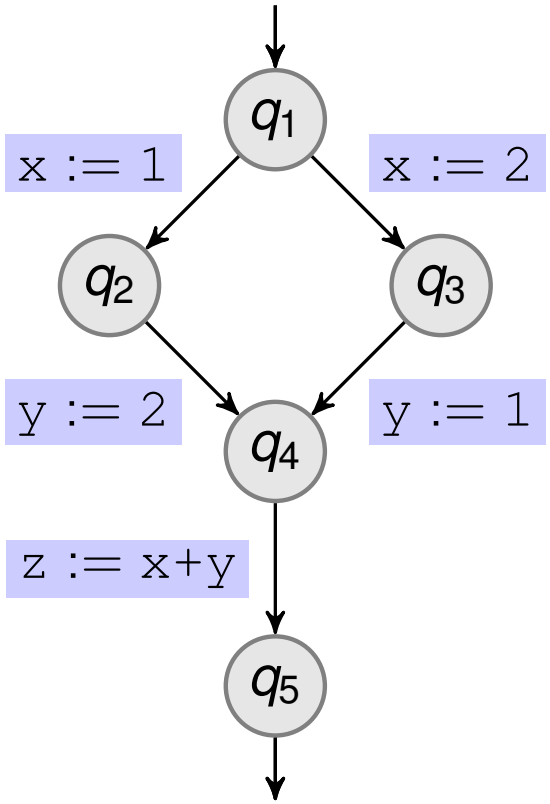
Further optimizations are possible...

Loss of Precision with the MFP Solution



At q_5, we have $z = 3$

Loss of Precision with the MFP Solution



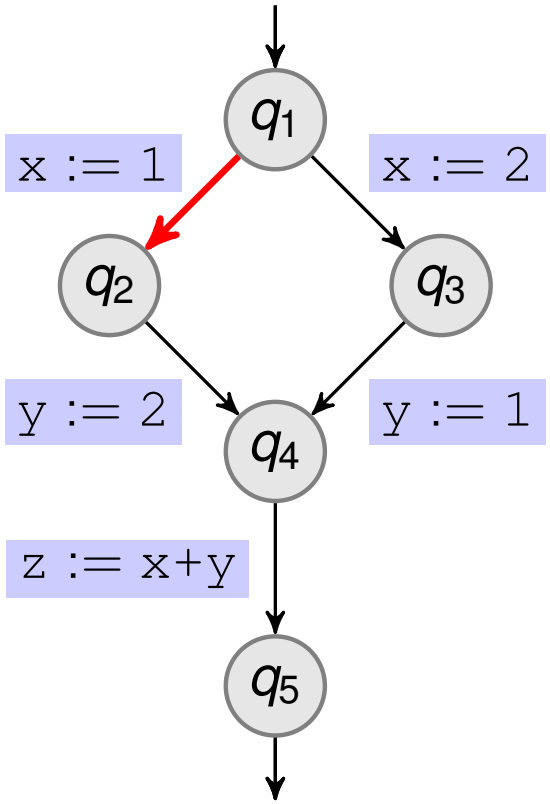
	x	y	z
q_1	$\top$	$\top$	$\top$
q_2	$\bot$	$\bot$	$\bot$
q_3	$\bot$	$\bot$	$\bot$
q_4	$\bot$	$\bot$	$\bot$
q_5	$\bot$	$\bot$	$\bot$

At q_5, we have $z = 3$

Loss of Precision

Cause: application of $\bigsqcup$ at q_4 to merge data flow information

Loss of Precision with the MFP Solution



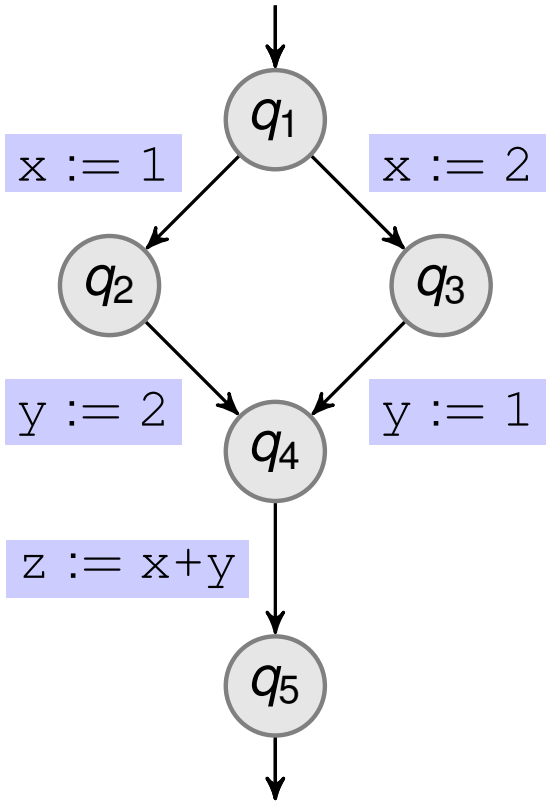
	x	y	z
q_1	$\top$	$\top$	$\top$
q_2	1	$\top$	$\top$
q_3	$\bot$	$\bot$	$\bot$
q_4	$\bot$	$\bot$	$\bot$
q_5	$\bot$	$\bot$	$\bot$

At q_5, we have $z = 3$

Loss of Precision

Cause: application of $\bigsqcup$ at q_4 to merge data flow information

Loss of Precision with the MFP Solution



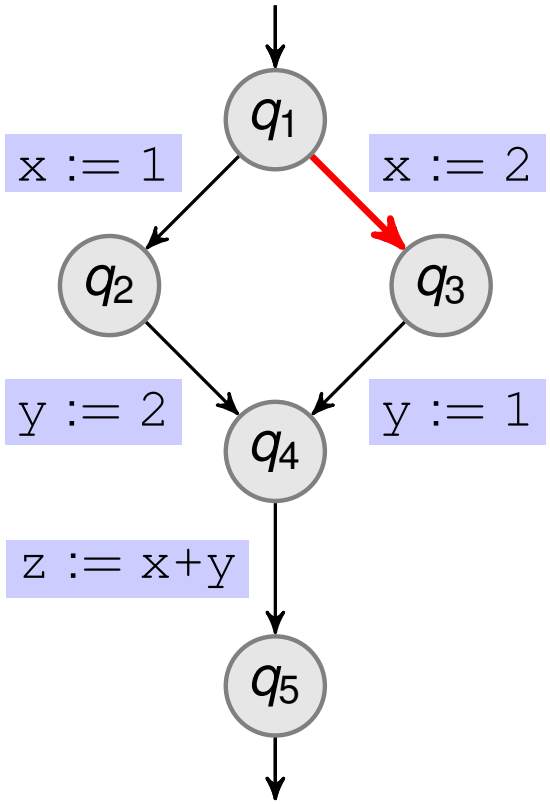
	x	y	z
q_1	$\top$	$\top$	$\top$
q_2	1	$\top$	$\top$
q_3	$\bot$	$\bot$	$\bot$
q_4	$\bot$	$\bot$	$\bot$
q_5	$\bot$	$\bot$	$\bot$

At q_5, we have $z = 3$

Loss of Precision

Cause: application of $\bigsqcup$ at q_4 to merge data flow information

Loss of Precision with the MFP Solution



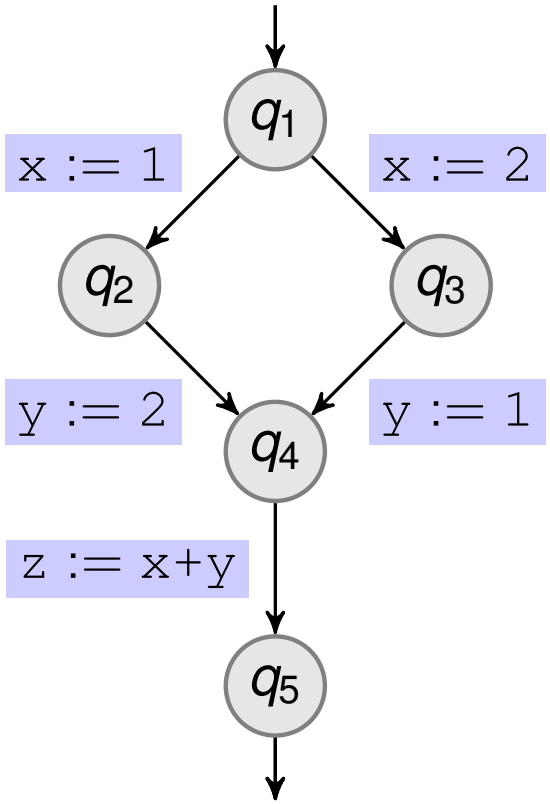
	x	y	z
q_1	$\top$	$\top$	$\top$
q_2	1	$\top$	$\top$
q_3	2	$\top$	$\top$
q_4	$\bot$	$\bot$	$\bot$
q_5	$\bot$	$\bot$	$\bot$

At q_5, we have $z = 3$

Loss of Precision

Cause: application of $\bigsqcup$ at q_4 to merge data flow information

Loss of Precision with the MFP Solution



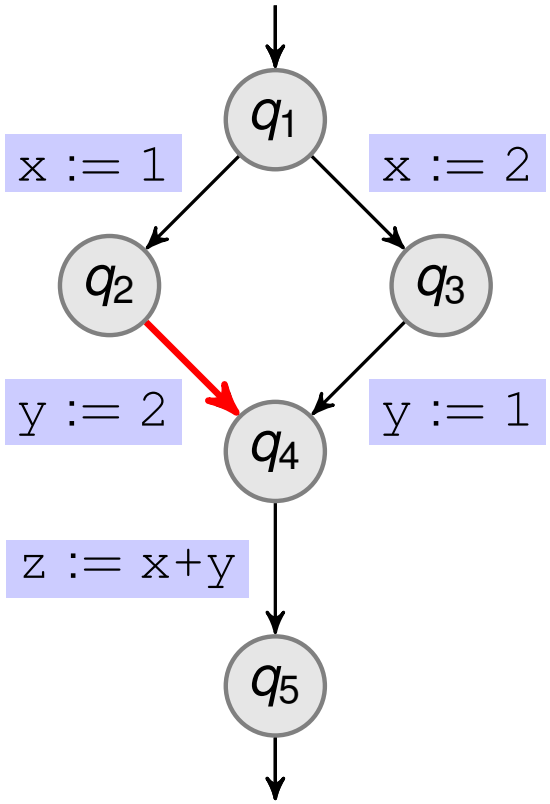
	x	y	z
q_1	$\top$	$\top$	$\top$
q_2	1	$\top$	$\top$
q_3	2	$\top$	$\top$
q_4	$\bot$	$\bot$	$\bot$
q_5	$\bot$	$\bot$	$\bot$

At q_5, we have $z = 3$

Loss of Precision

Cause: application of $\bigsqcup$ at q_4 to merge data flow information

Loss of Precision with the MFP Solution



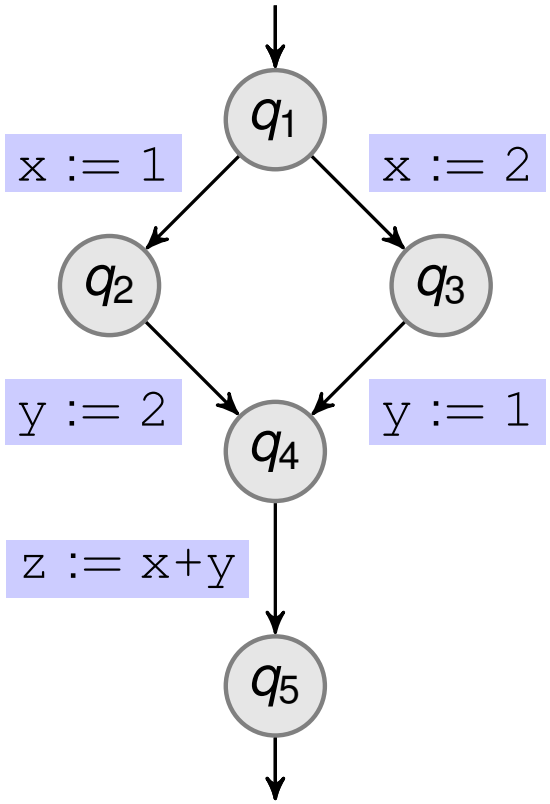
	x	y	z
q_1	$\top$	$\top$	$\top$
q_2	1	$\top$	$\top$
q_3	2	$\top$	$\top$
q_4	1	2	$\top$
q_5	$\bot$	$\bot$	$\bot$

At q_5, we have $z = 3$

Loss of Precision

Cause: application of $\bigsqcup$ at q_4 to merge data flow information

Loss of Precision with the MFP Solution



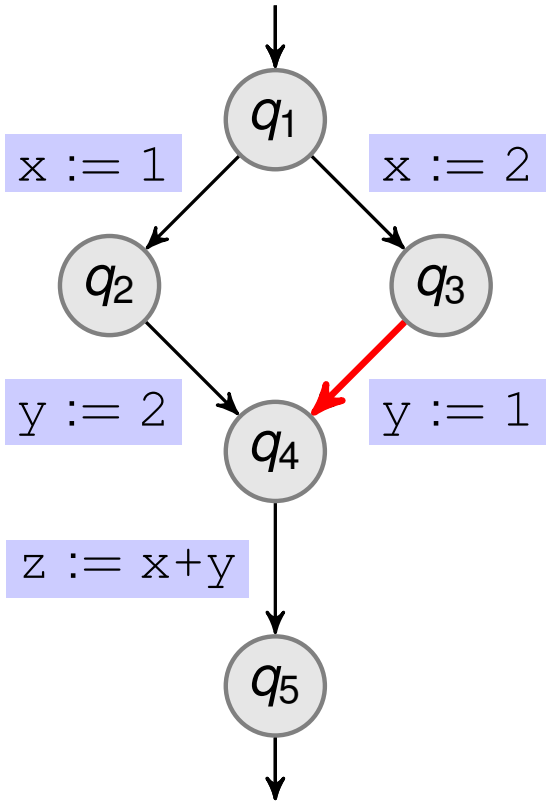
	x	y	z
q_1	$\top$	$\top$	$\top$
q_2	1	$\top$	$\top$
q_3	2	$\top$	$\top$
q_4	1	2	$\top$
q_5	$\bot$	$\bot$	$\bot$

At q_5, we have $z = 3$

Loss of Precision

Cause: application of $\bigsqcup$ at q_4 to merge data flow information

Loss of Precision with the MFP Solution



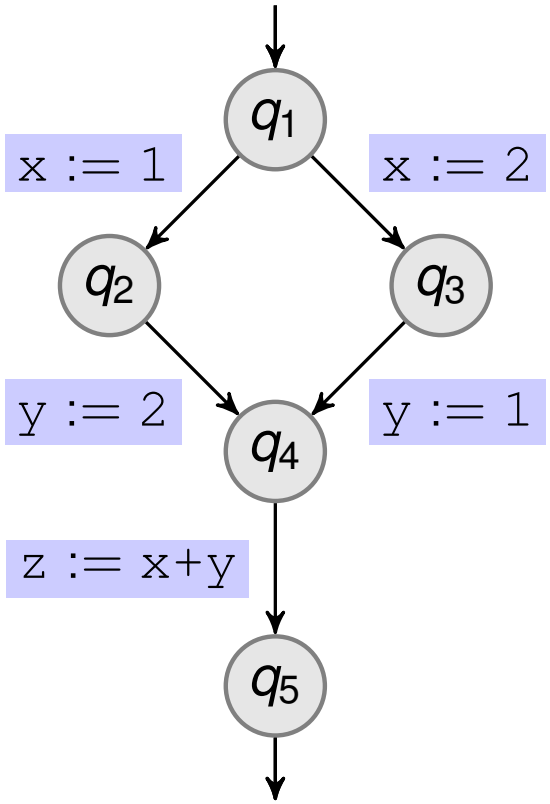
	x	y	z
q_1	$\top$	$\top$	$\top$
q_2	1	$\top$	$\top$
q_3	2	$\top$	$\top$
q_4	$1 \sqcup 2$	$2 \sqcup 1$	$\top$
q_5	$\bot$	$\bot$	$\bot$

At q_5, we have $z = 3$

Loss of Precision

Cause: application of $\sqcup$ at q_4 to merge data flow information

Loss of Precision with the MFP Solution



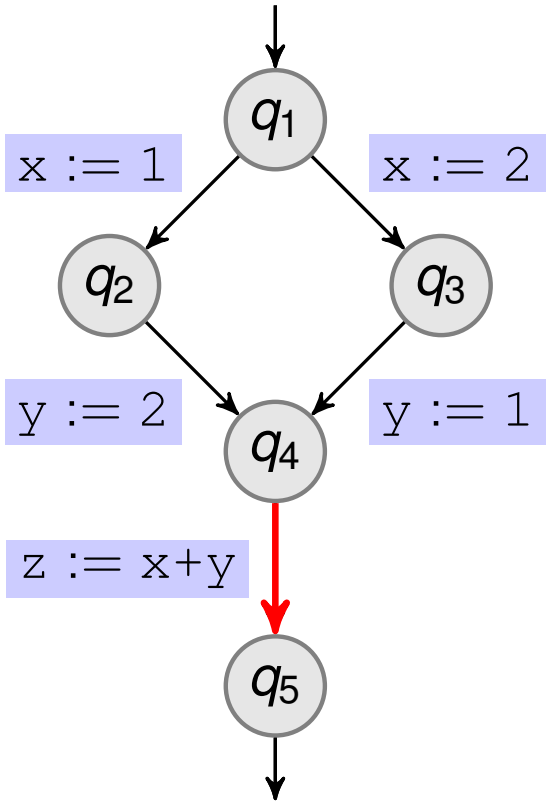
	x	y	z
q_1	$\top$	$\top$	$\top$
q_2	1	$\top$	$\top$
q_3	2	$\top$	$\top$
q_4	$\top$	$\top$	$\top$
q_5	$\bot$	$\bot$	$\bot$

At q_5, we have $z = 3$

Loss of Precision

Cause: application of $\bigsqcup$ at q_4 to merge data flow information

Loss of Precision with the MFP Solution



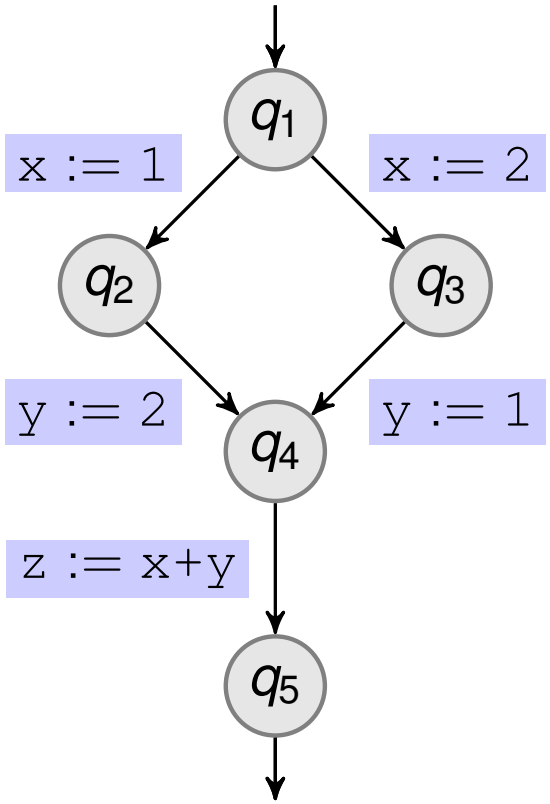
	x	y	z
q_1	$\top$	$\top$	$\top$
q_2	1	$\top$	$\top$
q_3	2	$\top$	$\top$
q_4	$\top$	$\top$	$\top$
q_5	$\top$	$\top$	$\top$

At q_5, we have $z = 3$

Loss of Precision

Cause: application of $\sqcup$ at q_4 to merge data flow information

Loss of Precision with the MFP Solution



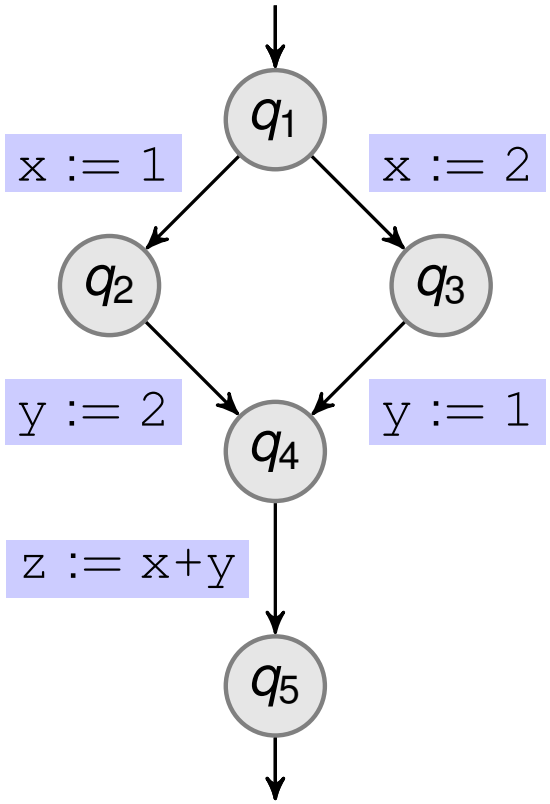
	x	y	z
q_1	$\top$	$\top$	$\top$
q_2	1	$\top$	$\top$
q_3	2	$\top$	$\top$
q_4	$\top$	$\top$	$\top$
q_5	$\top$	$\top$	$\top$

At q_5, we have $z = 3$

Loss of Precision

Cause: application of $\sqcup$ at q_4 to merge data flow information

Loss of Precision with the MFP Solution



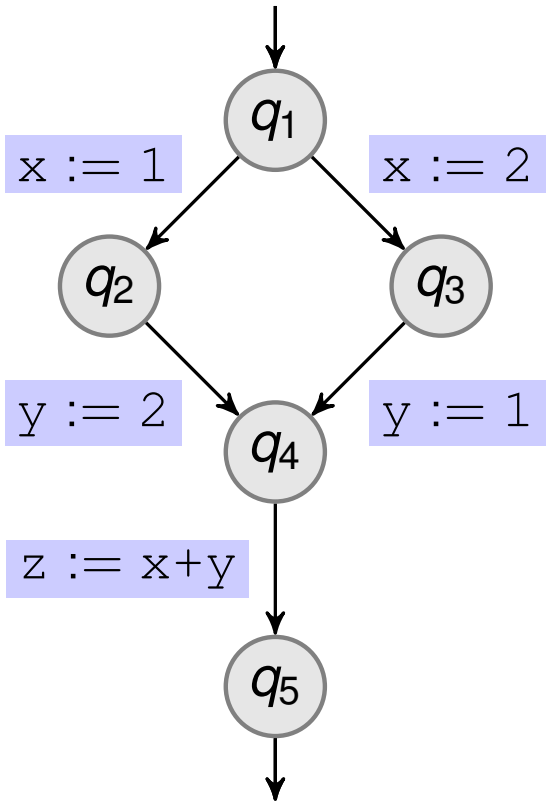
	x	y	z
q_1	$\top$	$\top$	$\top$
q_2	1	$\top$	$\top$
q_3	2	$\top$	$\top$
q_4	$\top$	$\top$	$\top$
q_5	$\top$	$\top$	$\top$

At q_5, we have $z = 3$

Loss of Precision

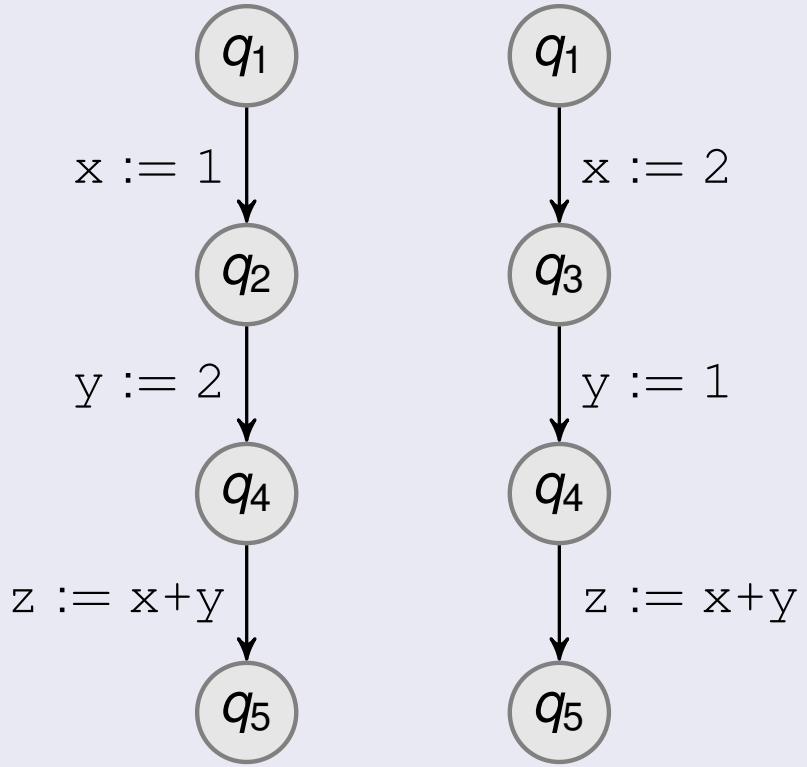
Cause: application of $\bigsqcup$ at q_4 to merge data flow information

Alternative Approach for Better Precision

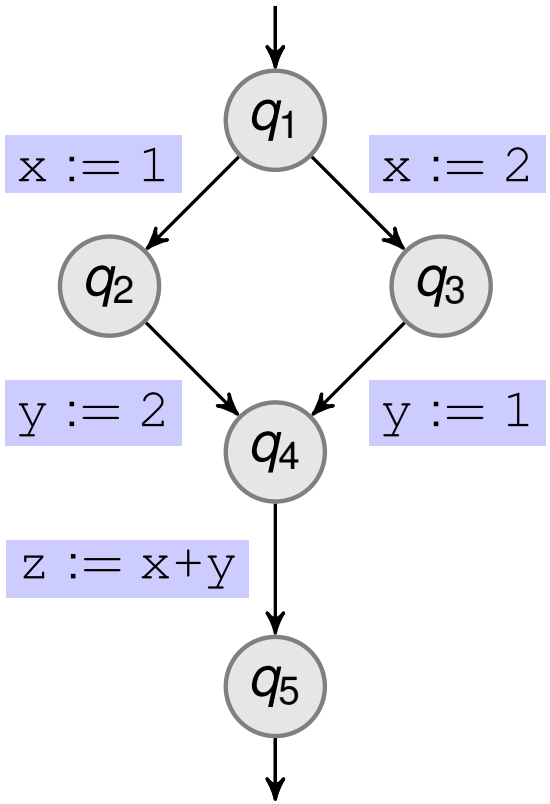


At q_5, we have z = 3

Control Paths from q_1 to q_5

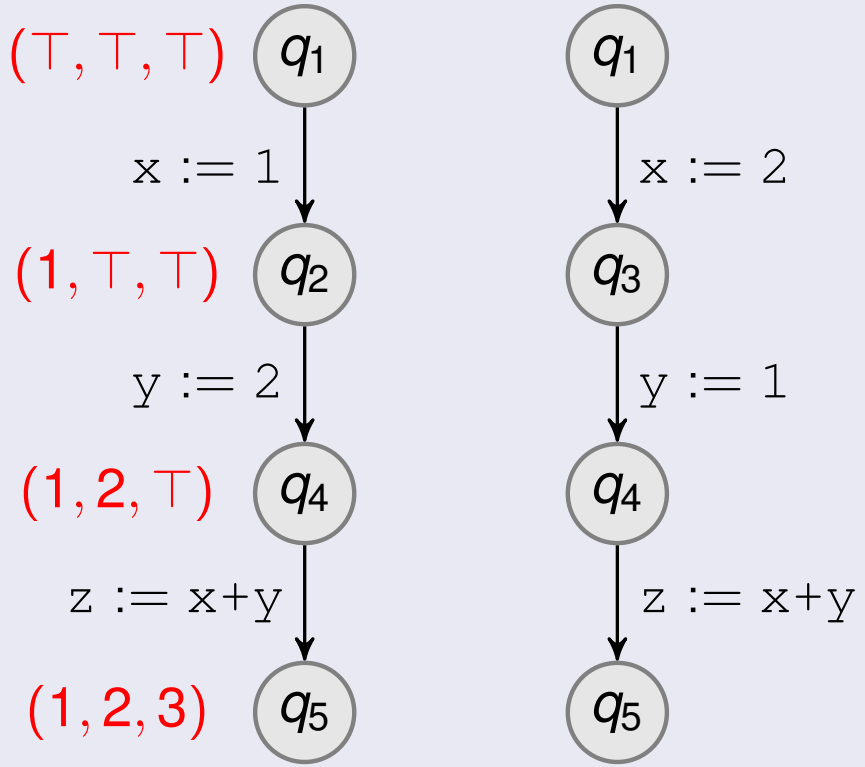


Alternative Approach for Better Precision

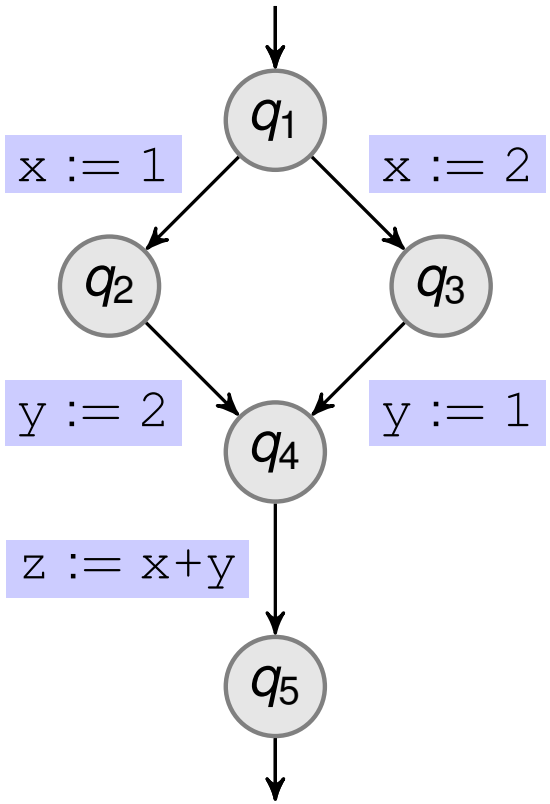


At q_5, we have $z = 3$

Control Paths from q_1 to q_5

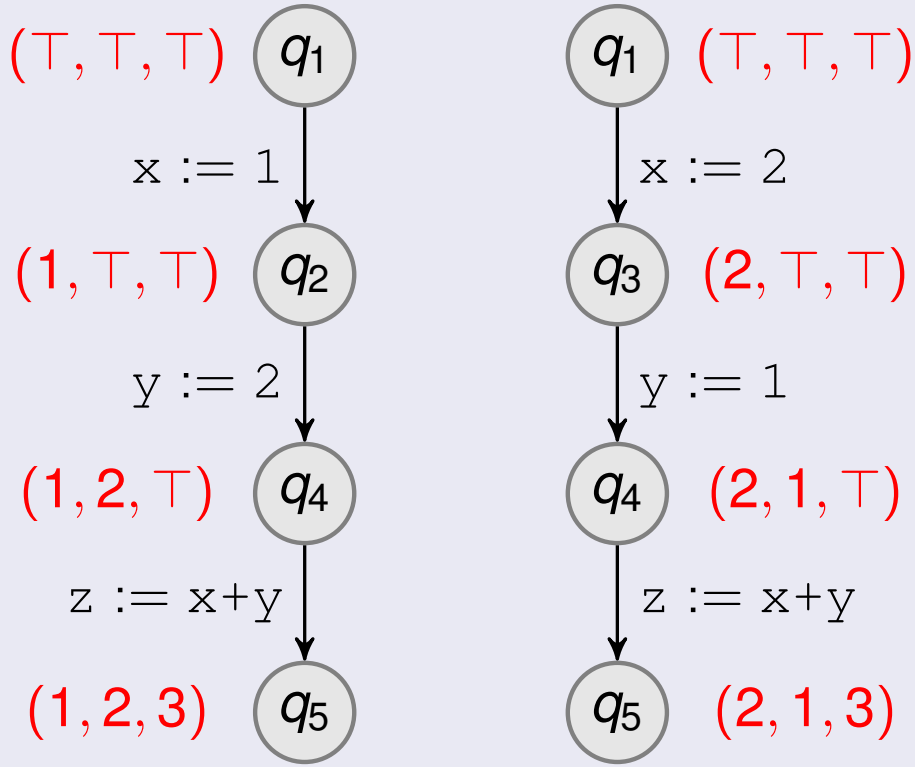


Alternative Approach for Better Precision

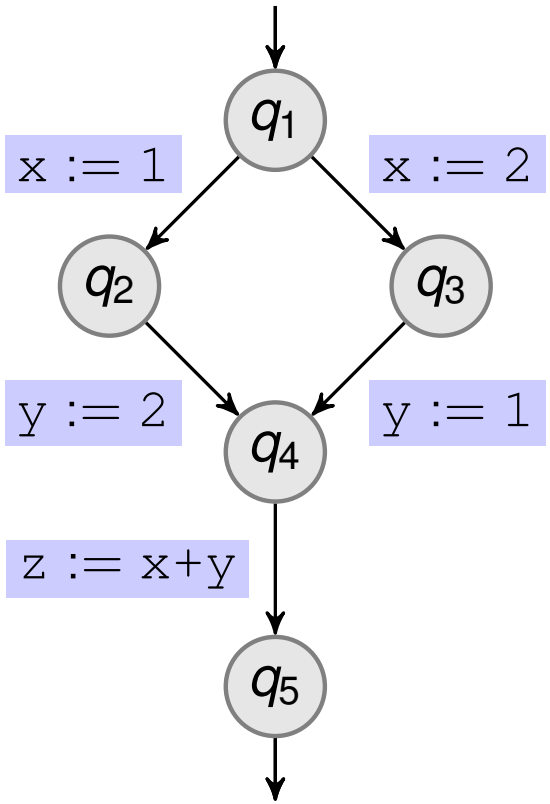


At q_5, we have $z = 3$

Control Paths from q_1 to q_5

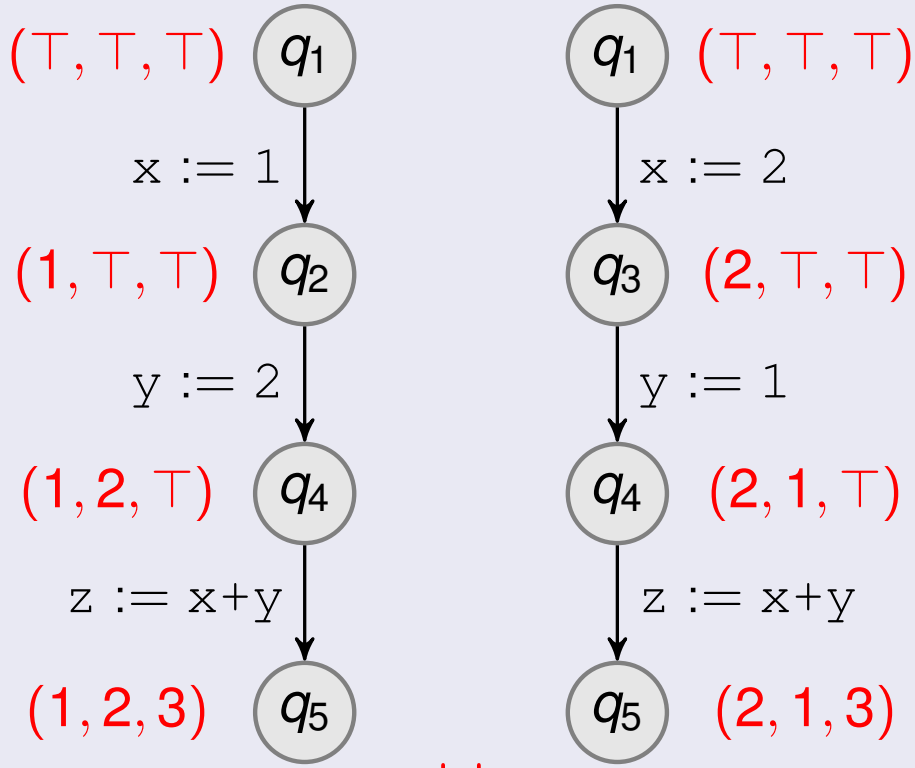


Alternative Approach for Better Precision



At q_5, we have $z = 3$

Control Paths from q_1 to q_5



Meet Over All Paths (MOP) Solution

Consider a data flow instance $\langle (L, \sqsubseteq), \mathcal{F}, Q, q_{in}, q_{out}, \mathrm{X}, \rightarrow, f, \imath \rangle$.

Forward Meet Over All Paths Solution

$$\overrightarrow{\mathrm{MOP}} = \lambda q . \bigsqcup \left\{ f_{\mathrm{op}_k} \circ \cdots \circ f_{\mathrm{op}_0}(\imath) \;\middle|\; q_{in} \xrightarrow{\mathrm{op}_0} q_1 \cdots q_k \xrightarrow{\mathrm{op}_k} q \right\}$$

Backward Meet Over All Paths Solution

$$\overleftarrow{\mathrm{MOP}} = \lambda q . \bigsqcup \left\{ f_{\mathrm{op}_0} \circ \cdots \circ f_{\mathrm{op}_k}(\imath) \;\middle|\; q \xrightarrow{\mathrm{op}_0} q_1 \cdots q_k \xrightarrow{\mathrm{op}_k} q_{out} \right\}$$

More precise than MFP

$$\overrightarrow{\mathrm{MOP}} \sqsubseteq \overrightarrow{\mathrm{MFP}}$$
$$\overleftarrow{\mathrm{MOP}} \sqsubseteq \overleftarrow{\mathrm{MFP}}$$

Not Computable in General

$\overrightarrow{\mathrm{MOP}}(q) \stackrel{?}{=} 1$ is undecidable for constant propagation

Meet Over All Paths (MOP) Solution

Consider a data flow instance $\langle (L, \sqsubseteq), \mathcal{F}, Q, q_{in}, q_{out}, \mathrm{X}, \rightarrow, f, \imath \rangle$.

Forward Meet Over All Paths Solution

$$\overrightarrow{\mathrm{MOP}} = \lambda q . \bigsqcup \left\{ f_{\mathrm{op}_k} \circ \cdots \circ f_{\mathrm{op}_0}(\imath) \;\middle|\; q_{in} \xrightarrow{\mathrm{op}_0} q_1 \cdots q_k \xrightarrow{\mathrm{op}_k} q \right\}$$

Backward Meet Over All Paths Solution

$$\overleftarrow{\mathrm{MOP}} = \lambda q . \bigsqcup \left\{ f_{\mathrm{op}_0} \circ \cdots \circ f_{\mathrm{op}_k}(\imath) \;\middle|\; q \xrightarrow{\mathrm{op}_0} q_1 \cdots q_k \xrightarrow{\mathrm{op}_k} q_{out} \right\}$$

More precise than MFP

$$\overrightarrow{\mathrm{MOP}} \sqsubseteq \overrightarrow{\mathrm{MFP}}$$
$$\overleftarrow{\mathrm{MOP}} \sqsubseteq \overleftarrow{\mathrm{MFP}}$$

Not Computable in General

$\overrightarrow{\mathrm{MOP}}(q) \stackrel{?}{=} 1$ is undecidable for constant propagation

MOP = MFP in Distributive Frameworks

A monotone framework $\langle (L, \sqsubseteq), \mathcal{F} \rangle$ is distributive if every $f \in \mathcal{F}$ is completely additive:

$$f(\bigsqcup X) = \bigsqcup \{f(\phi) \mid \phi \in X\} \qquad \text{(for all } X \subseteq L\text{)}$$

Theorem

For any data flow instance over a distributive monotone framework,

$$\overrightarrow{\text{MOP}} = \overrightarrow{\text{MFP}}$$
$$\overleftarrow{\text{MOP}} = \overleftarrow{\text{MFP}}$$

Intuition

In a distributive framework, applying $\bigsqcup$ "early" does not lose precision:

$$f_{\text{op}_5}\left(f_{\text{op}_2}(\phi) \sqcup f_{\text{op}_3}(\psi)\right) = f_{\text{op}_5} \circ f_{\text{op}_2}(\phi) \sqcup f_{\text{op}_5} \circ f_{\text{op}_3}(\psi)$$

MOP = MFP in Distributive Frameworks

A monotone framework $\langle (L, \sqsubseteq), \mathcal{F} \rangle$ is distributive if every $f \in \mathcal{F}$ is completely additive:

$$f\left(\bigsqcup X\right) \quad = \quad \bigsqcup \{f(\phi) \mid \phi \in X\} \qquad \text{(for all } X \subseteq L\text{)}$$

Theorem

For any data flow instance over a distributive monotone framework,

$$\begin{array}{rcl} \overrightarrow{\text{MOP}} & = & \overrightarrow{\text{MFP}} \\ \overleftarrow{\text{MOP}} & = & \overleftarrow{\text{MFP}} \end{array}$$

Intuition

In a distributive framework, applying $\bigsqcup$ "early" does not lose precision:

$$f_{\text{op}_5}\left(f_{\text{op}_2}(\phi) \sqcup f_{\text{op}_3}(\psi)\right) \quad = \quad f_{\text{op}_5} \circ f_{\text{op}_2}(\phi) \sqcup f_{\text{op}_5} \circ f_{\text{op}_3}(\psi)$$

Examples of Distributive Monotone Frameworks

Gen / Kill Monotone Frameworks

- Complete lattice $(L, \sqsubseteq)$ of data flow facts:

$$L = \mathcal{P}(S) \text{ for some set } S \qquad \sqsubseteq \text{ is } \subseteq \text{ or } \supseteq$$

- Set $\mathcal{F}$ of monotonic transfer functions:

$$\mathcal{F} = \{\lambda \phi \,.\, \mathit{gen} \cup (\phi \setminus \mathit{kill}) \mid \mathit{gen}, \mathit{kill} \in L\}$$

All gen / kill monotone frameworks are distributive

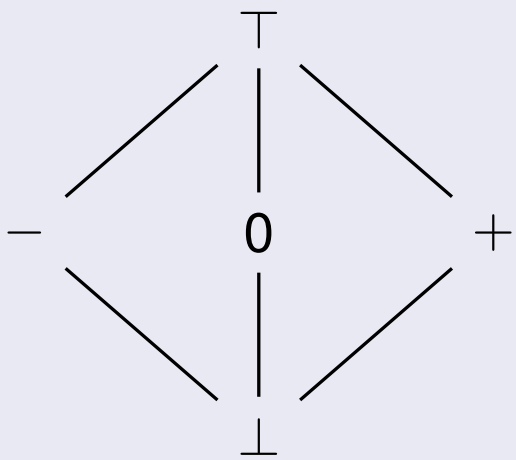
Examples

- Live Variables
- Available Expressions
- Uninitialized Variables
- ...

Sign Analysis: Monotone Framework

Control Flow Automaton: $\langle Q, q_{in}, q_{out}, \mathtt{X}, \rightarrow \rangle$

(Simplified) Sign Lattice for a Single Variable: $(Sign, \sqsubseteq)$



ϕ	*Meaning*
$\top$	$\mathbb{R}$
$-$	$\{r \in \mathbb{R} \mid r < 0\}$
$+$	$\{r \in \mathbb{R} \mid r > 0\}$
0	$\{0\}$
$\bot$	$\emptyset$

Monotone Framework

- Complete lattice $(L, \sqsubseteq)$ of data flow facts: $(\mathtt{X} \rightarrow Sign, \sqsubseteq)$
- Set $\mathcal{F}$ defined as the set of all monotonic transfer functions on L.

Sign Analysis: Data Flow Instance

Control Flow Automaton: $\langle Q, q_{in}, q_{out}, \mathbb{X}, \rightarrow \rangle$

Monotone Framework

- Complete lattice $(L, \sqsubseteq)$ of data flow facts: $(\mathbb{X} \rightarrow \mathit{Sign}, \sqsubseteq)$
- Set $\mathcal{F}$ defined as the set of all monotonic transfer functions on L.

Data Flow Instance

- Initial data flow value: $\top$
- Transfer mapping:

$$f_{x:=e}(\phi) = \lambda y . \begin{cases} \phi(y) & \text{if } y \neq x \\ [\![e]\!]_\phi & \text{if } y = x \end{cases} \qquad f_g(\phi) = \phi$$

Forward analysis

Sign Analysis: Transfer Mapping

Need to define $[\![e]\!]$ for valuations v in $\mathbb{X} \to \{-, 0, +, \bot, \top\}$

Expressions: $[\![e]\!]_v$

$$
\begin{aligned}
[\![c]\!]_v &= sign(c) && [c \in \mathbb{Q}] \\
[\![x]\!]_v &= v(x) && [x \in \mathbb{X}] \\
[\![e_1 + e_2]\!]_v &= [\![e_1]\!]_v \oplus [\![e_2]\!]_v \\
[\![e_1 - e_2]\!]_v &= [\![e_1]\!]_v \ominus [\![e_2]\!]_v \\
[\![e_1 * e_2]\!]_v &= [\![e_1]\!]_v \otimes [\![e_2]\!]_v
\end{aligned}
$$

$$
sign(c) = \begin{cases} - & \text{if } c < 0 \\ 0 & \text{if } c = 0 \\ + & \text{if } c > 0 \end{cases}
$$

"Abstract" Addition

$\oplus$	$\bot$	$-$	0	$+$	$\top$
$\bot$	$\bot$	$\bot$	$\bot$	$\bot$	$\bot$
$-$	$\bot$	$-$	$-$	$\top$	$\top$
0	$\bot$	$-$	0	$+$	$\top$
$+$	$\bot$	$\top$	$+$	$+$	$\top$
$\top$	$\bot$	$\top$	$\top$	$\top$	$\top$

Tables also required for:

- "abstract" subtraction
- "abstract" multiplication

Sign Analysis: Transfer Mapping

Need to define $[\![e]\!]$ for valuations v in $\mathbb{X} \to \{-, 0, +, \bot, \top\}$

Expressions: $[\![e]\!]_v$

$$
\begin{aligned}
[\![c]\!]_v &= sign(c) && [c \in \mathbb{Q}] \\
[\![x]\!]_v &= v(x) && [x \in \mathbb{X}] \\
[\![e_1 + e_2]\!]_v &= [\![e_1]\!]_v \oplus [\![e_2]\!]_v \\
[\![e_1 - e_2]\!]_v &= [\![e_1]\!]_v \ominus [\![e_2]\!]_v \\
[\![e_1 * e_2]\!]_v &= [\![e_1]\!]_v \otimes [\![e_2]\!]_v
\end{aligned}
$$

$$
sign(c) = \begin{cases} - & \text{if } c < 0 \\ 0 & \text{if } c = 0 \\ + & \text{if } c > 0 \end{cases}
$$

"Abstract" Addition

$\oplus$	$\bot$	$-$	0	$+$	$\top$
$\bot$	$\bot$	$\bot$	$\bot$	$\bot$	$\bot$
$-$	$\bot$	$-$	$-$	$\top$	$\top$
0	$\bot$	$-$	0	$+$	$\top$
$+$	$\bot$	$\top$	$+$	$+$	$\top$
$\top$	$\bot$	$\top$	$\top$	$\top$	$\top$

Tables also required for:

- "abstract" subtraction
- "abstract" multiplication

Sign Analysis: Transfer Mapping

Need to define $[\![e]\!]$ f... valuations v in $\mathbb{X} \to \{-, 0, +, \bot, \top\}$

Are these tables correct?

Expressions: $[\![e]\!]_v$

$$
\begin{aligned}
[\![c]\!]_v &= sign(c) && [c \in \mathbb{Q}] \\
[\![x]\!]_v &= v(x) && [x \in \mathbb{X}] \\
[\![e_1 + e_2]\!]_v &= [\![e_1]\!]_v \oplus [\![e_2]\!]_v \\
[\![e_1 - e_2]\!]_v &= [\![e_1]\!]_v \ominus [\![e_2]\!]_v \\
[\![e_1 * e_2]\!]_v &= [\![e_1]\!]_v \otimes [\![e_2]\!]_v
\end{aligned}
$$

...stract" Addition

			0	+	⊤
	⊥				⊥
−	⊥				
0	⊥	−			
+	⊥	⊤	+	+	
⊤	⊥	⊤	⊤	⊤	⊤

$$
sign(c) = \begin{cases} - & \text{if } c < 0 \\ 0 & \text{if } c = 0 \\ + & \text{if } c > 0 \end{cases}
$$

Tables also required for:

- "abstract" subtraction
- "abstract" multiplication

Sign Analysis: Transfer Mapping

Need to define $[\![e]\!]$ f… valuations v in $\mathbb{X} \to \{-, 0, +, \bot, \top\}$

Are these tables correct?

Expressions: $[\![e]\!]_v$

$$[\![c]\!]_v = sign(c) \quad [c \in \mathbb{Q}]$$

$$[\![x]\!]_v$$

…stract" Addition

	0	+	⊤
			⊥

…tion

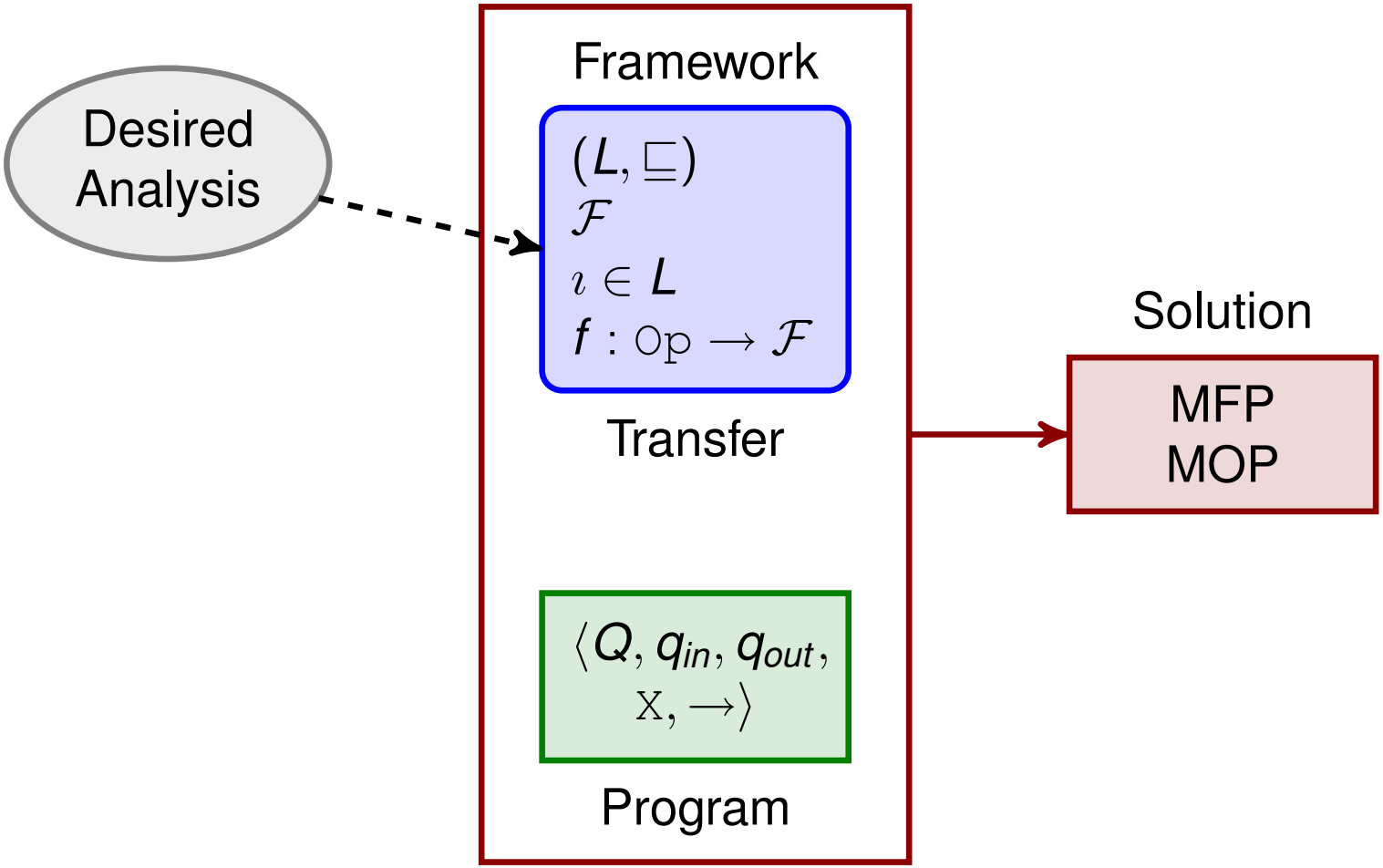
…ultiplication

Does this data flow instance really perform sign analysis?

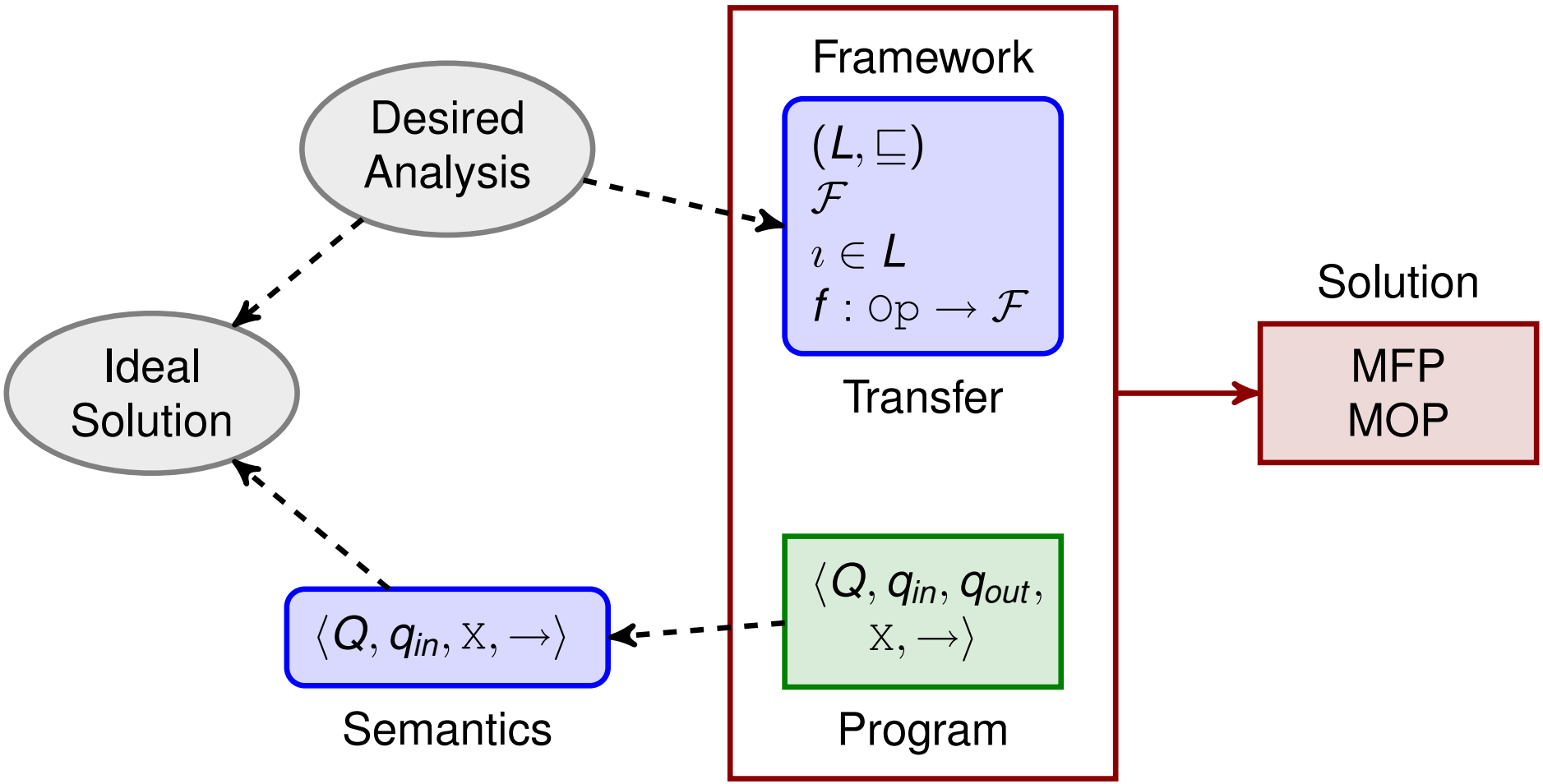
Is the analysis correct?

Is it precise?

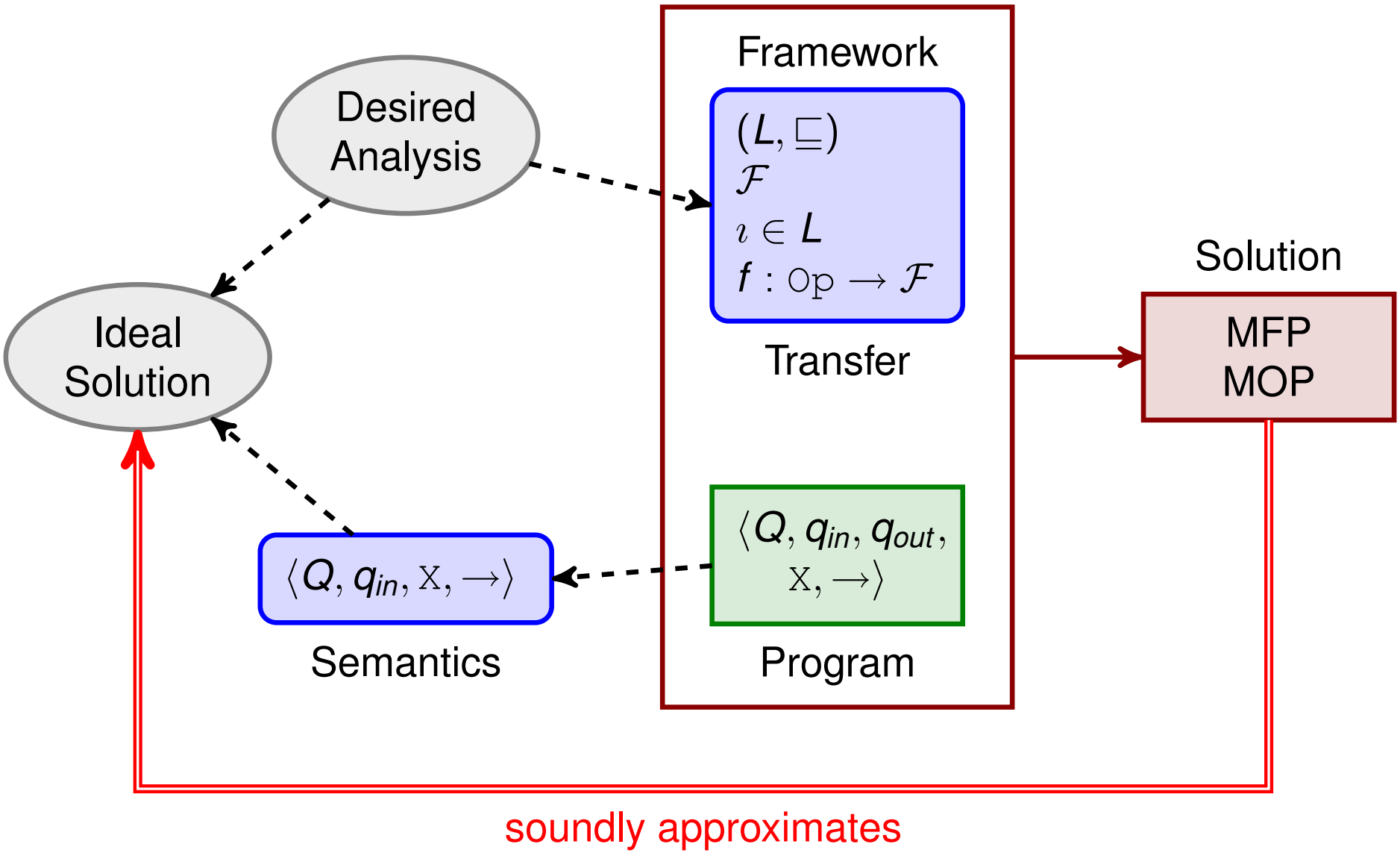
What About Correctness of Data Flow Analyses?



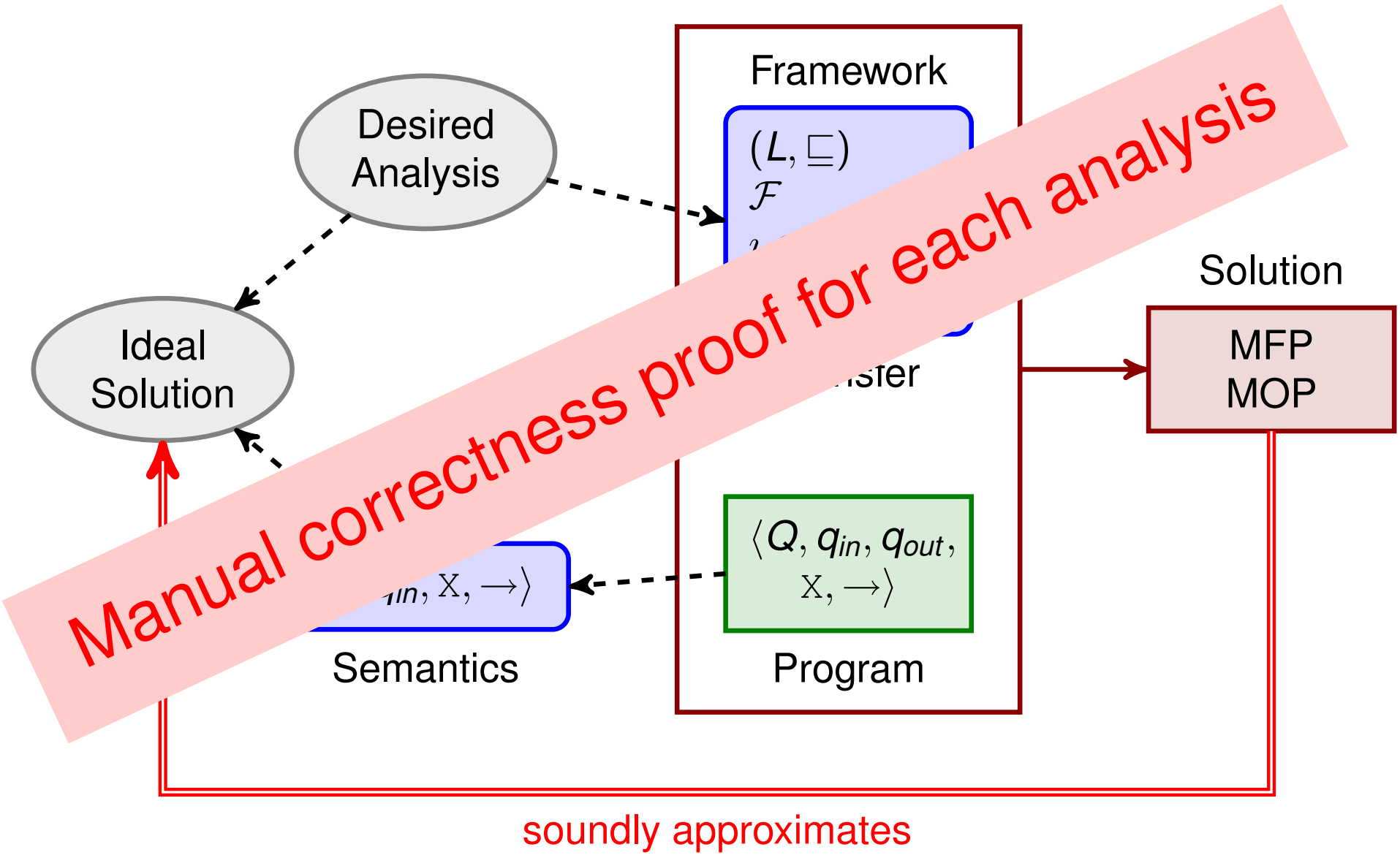
What About Correctness of Data Flow Analyses?



What About Correctness of Data Flow Analyses?



What About Correctness of Data Flow Analyses?



How to Systematically Ensure Correctness?

Data flow facts have an intended meaning.

The transfer mapping is designed according to this intended meaning.

We need a formal link to relate data flow facts and transfer functions with the formal semantics.

Solution: Abstract Interpretation

« This paper is devoted to the systematic and correct design of program analysis frameworks with respect to a formal semantics. »

P. Cousot & R. Cousot. Systematic Design of Program Analysis Frameworks. *Sixth Annual Symposium on Principles of Programming Languages*, 1979.

How to Systematically Ensure Correctness?

Data flow facts have an intended meaning.

The transfer mapping is designed according to this intended meaning.

We need a formal link to relate data flow facts and transfer functions with the formal semantics.

Solution: Abstract Interpretation

« This paper is devoted to the systematic and correct design of program analysis frameworks with respect to a formal semantics. »

P. Cousot & R. Cousot. Systematic Design of Program Analysis Frameworks. *Sixth Annual Symposium on Principles of Programming Languages*, 1979.

Part IV

Abstract Interpretation

Outline — Abstract Interpretation

Outline — Abstract Interpretation

Concrete Lattice & Abstract Lattice: Notations

Concrete lattice

$$(L, \sqsubseteq)$$

Abstract lattice

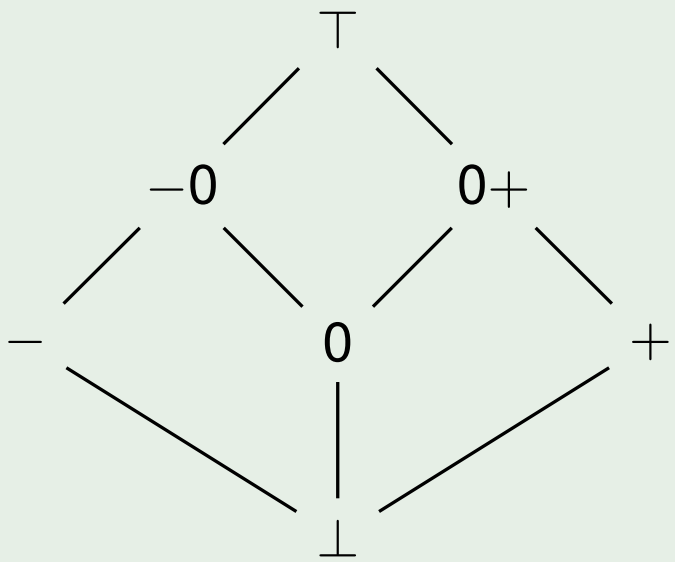
$$(\overline{L}, \overline{\sqsubseteq})$$

Example (Sets of Values)

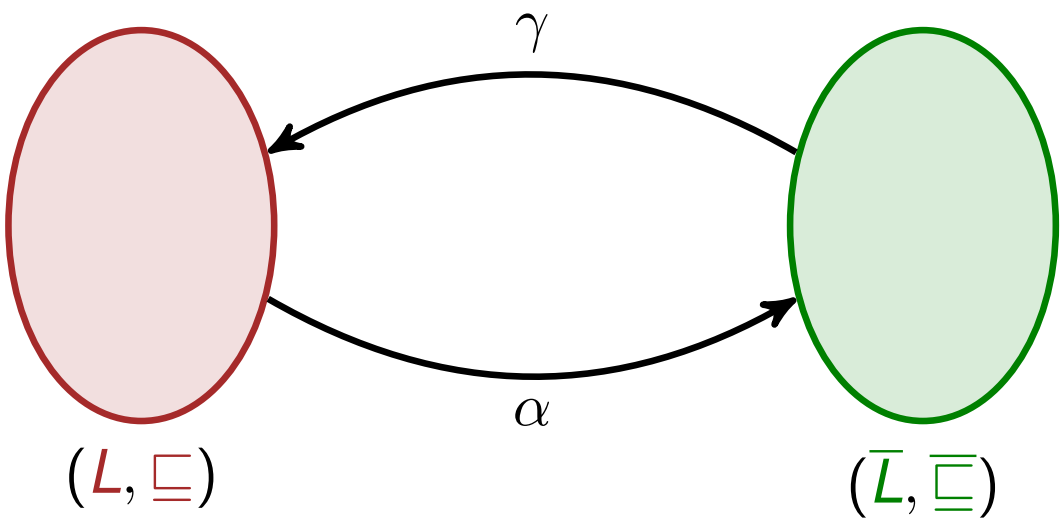
For a variable ranging over a domain $\mathbb{D}$:

$$(\mathcal{P}(\mathbb{D}), \subseteq)$$

Example (Sign Lattice)



Galois Connections: Definition



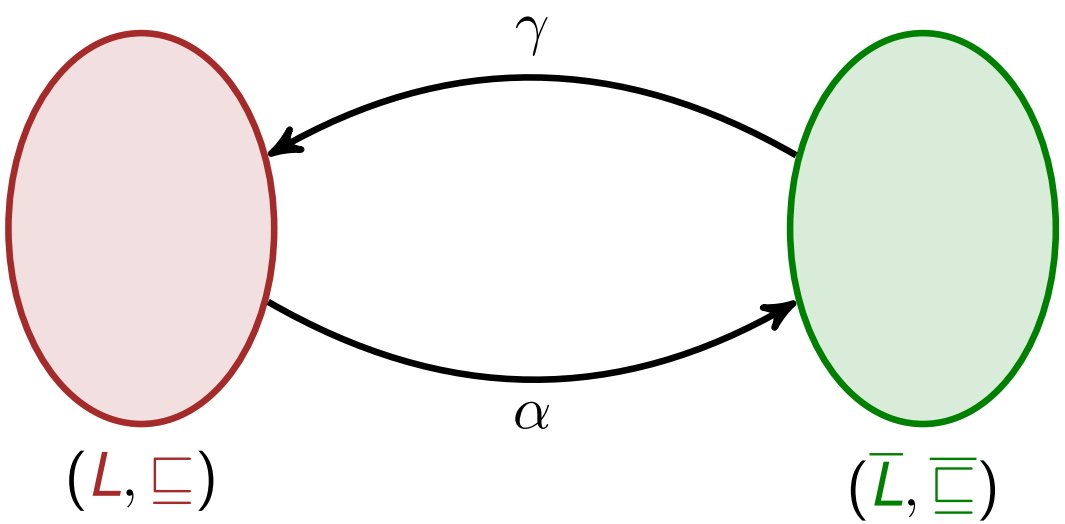
Definition

A Galois connection between a lattice $(L, \sqsubseteq)$ and a lattice $(\overline{L}, \overline{\sqsubseteq})$ is a pair of functions (α, γ), with $\alpha : L \to \overline{L}$ and $\gamma : \overline{L} \to L$, satisfying:

$$\alpha(x) \,\overline{\sqsubseteq}\, \overline{y} \quad \text{iff} \quad x \sqsubseteq \gamma(\overline{y}) \qquad \text{(for all } x \in L, \overline{y} \in \overline{L})$$

Notation for Galois connections: $(L, \sqsubseteq) \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} (\overline{L}, \overline{\sqsubseteq})$

Galois Connections: Definition



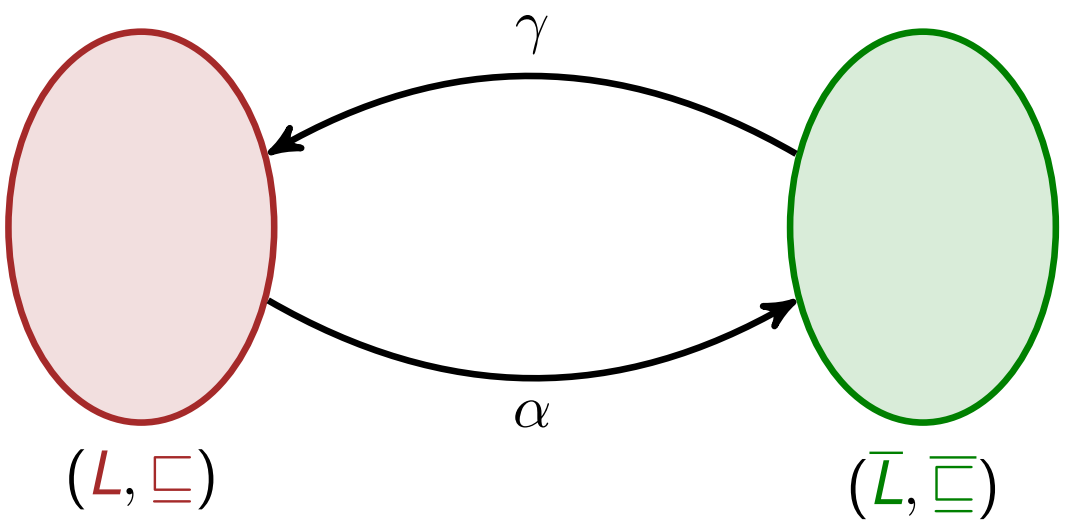
Definition

A Galois connection between a lattice $(L, \sqsubseteq)$ and a lattice $(\overline{L}, \overline{\sqsubseteq})$ is a pair of functions (α, γ), with $\alpha : L \to \overline{L}$ and $\gamma : \overline{L} \to L$, satisfying:

$$\alpha(x) \,\overline{\sqsubseteq}\, \overline{y} \quad \text{iff} \quad x \sqsubseteq \gamma(\overline{y}) \qquad \text{(for all } x \in L, \overline{y} \in \overline{L})$$

Notation for Galois connections: $(L, \sqsubseteq) \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} (\overline{L}, \overline{\sqsubseteq})$

Galois Connections: Intuition



Concretization

γ is the concretization function.

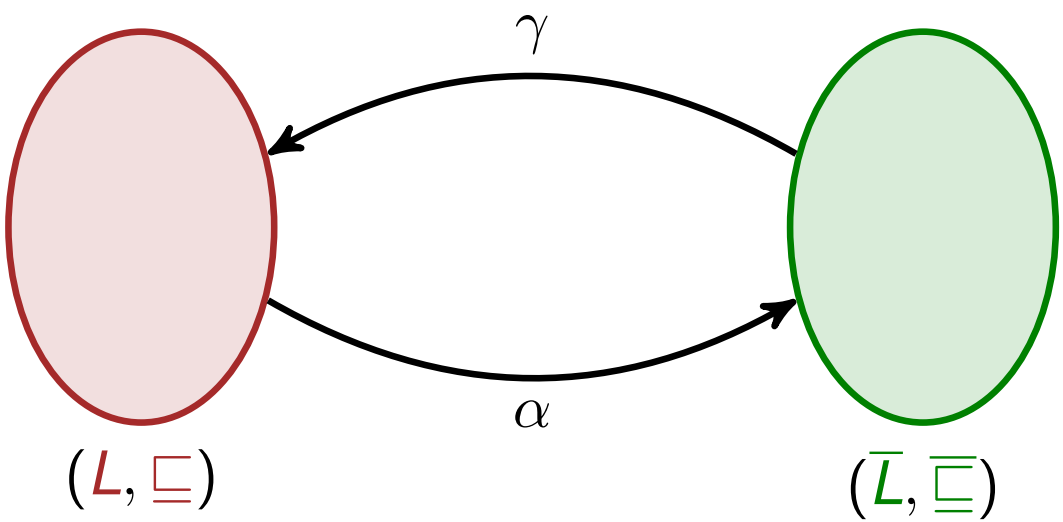
$\gamma(\overline{y})$ is the concrete value in L that is represented by $\overline{y}$.

Abstraction

α is the abstraction function.

$\alpha(x)$ is the most precise abstract value in $\overline{L}$ whose concretization approximates x.

Galois Connections: Intuition



Concretization

γ is the concretization function.

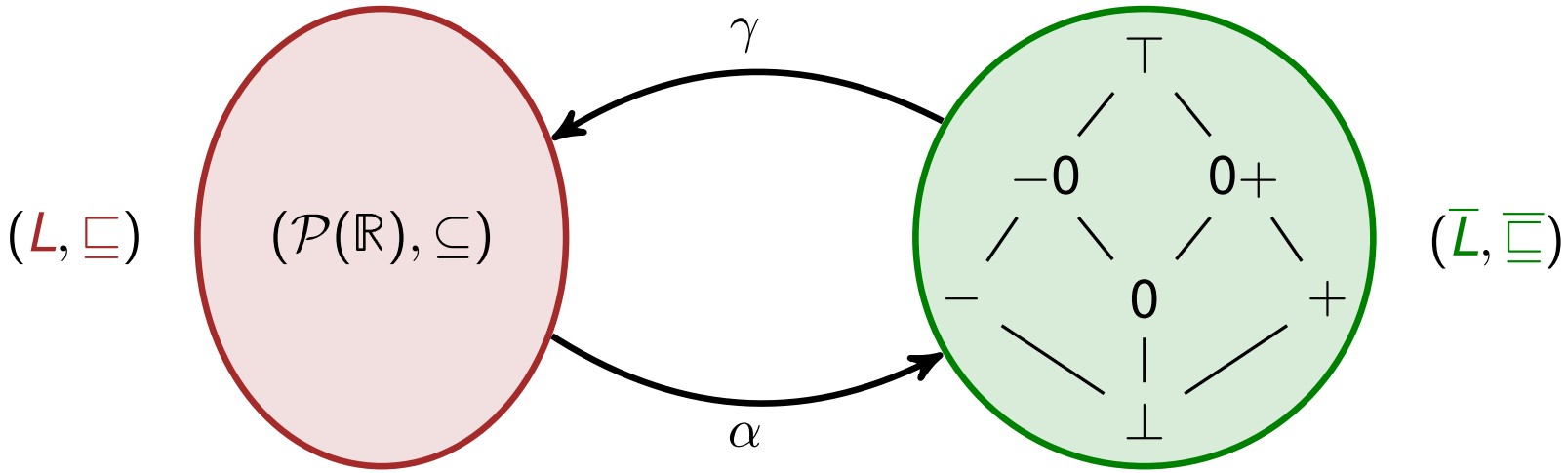
$\gamma(\overline{y})$ is the concrete value in L that is represented by $\overline{y}$.

Abstraction

α is the abstraction function.

$\alpha(x)$ is the most precise abstract value in $\overline{L}$ whose concretization approximates x.

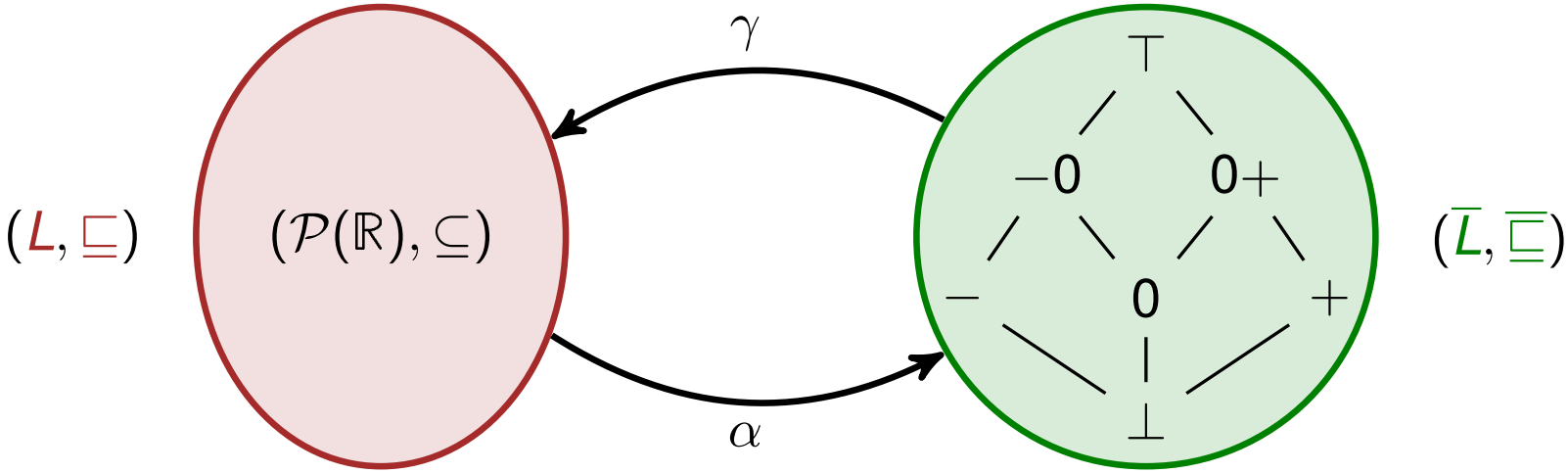
Galois Connections: Example



$$\alpha(x) = \begin{cases} \bot & \text{if } x = \emptyset \\ - & \text{if } x \subseteq \{r \in \mathbb{R} \mid r < 0\} \\ 0 & \text{if } x = \{0\} \\ + & \text{if } x \subseteq \{r \in \mathbb{R} \mid r > 0\} \\ -0 & \text{if } \{0\} \subset x \subseteq \{r \in \mathbb{R} \mid r \leq 0\} \\ 0+ & \text{if } \{0\} \subset x \subseteq \{r \in \mathbb{R} \mid r \geq 0\} \\ \top & \text{otherwise} \end{cases}$$

$\overline{y}$	$\gamma(\overline{y})$
$\bot$	$\emptyset$
$-$	$\{r \in \mathbb{R} \mid r < 0\}$
0	$\{0\}$
$+$	$\{r \in \mathbb{R} \mid r > 0\}$
-0	$\{r \in \mathbb{R} \mid r \leq 0\}$
$0+$	$\{r \in \mathbb{R} \mid r \geq 0\}$
$\top$	$\mathbb{R}$

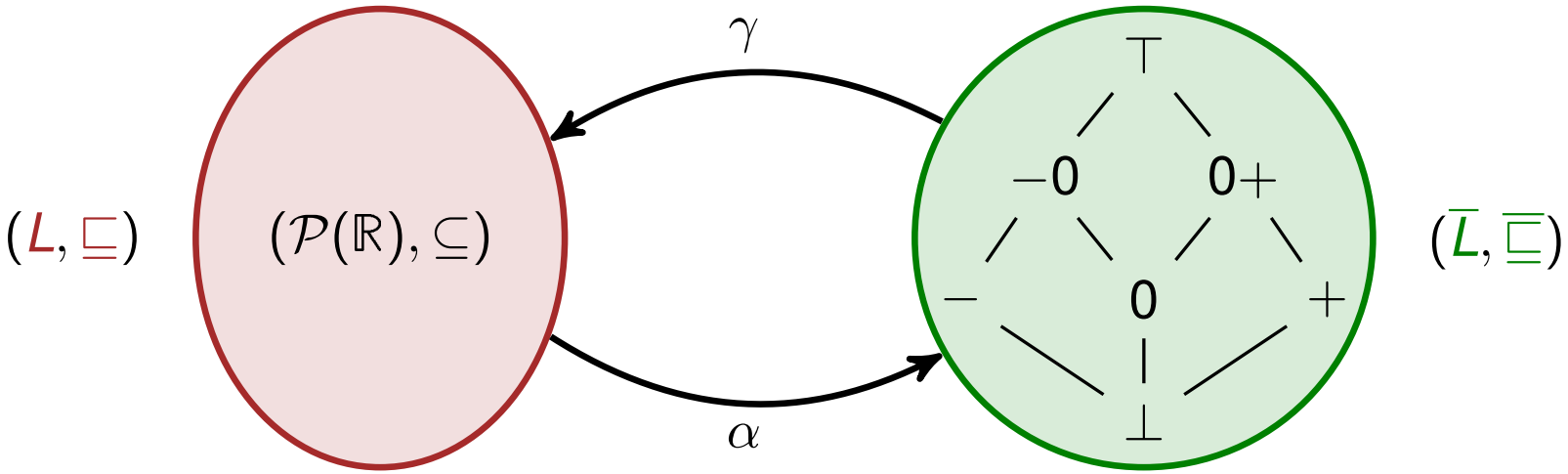
Galois Connections: Example



$$\alpha(x) = \begin{cases} \bot & \text{if } x = \emptyset \\ - & \text{if } x \subseteq \{r \in \mathbb{R} \mid r < 0\} \\ 0 & \text{if } x = \{0\} \\ + & \text{if } x \subseteq \{r \in \mathbb{R} \mid r > 0\} \\ -0 & \text{if } \{0\} \subset x \subseteq \{r \in \mathbb{R} \mid r \leq 0\} \\ 0+ & \text{if } \{0\} \subset x \subseteq \{r \in \mathbb{R} \mid r \geq 0\} \\ \top & \text{otherwise} \end{cases}$$

$\overline{y}$	$\gamma(\overline{y})$
$\bot$	$\emptyset$
$-$	$\{r \in \mathbb{R} \mid r < 0\}$
0	$\{0\}$
$+$	$\{r \in \mathbb{R} \mid r > 0\}$
-0	$\{r \in \mathbb{R} \mid r \leq 0\}$
$0+$	$\{r \in \mathbb{R} \mid r \geq 0\}$
$\top$	$\mathbb{R}$

Galois Connections: Example



$$\alpha(x) = \begin{cases} \bot & \text{if } x = \emptyset \\ - & \text{if } x \subseteq \{r \in \mathbb{R} \mid r < 0\} \\ 0 & \text{if } x = \{0\} \\ + & \text{if } x \subseteq \{r \in \mathbb{R} \mid r > 0\} \\ -0 & \text{if } \{0\} \subset x \subseteq \{r \in \mathbb{R} \mid r \leq 0\} \\ 0+ & \text{if } \{0\} \subset x \subseteq \{r \in \mathbb{R} \mid r \geq 0\} \\ \top & \text{otherwise} \end{cases}$$

$\overline{y}$	$\gamma(\overline{y})$
$\bot$	$\emptyset$
$-$	$\{r \in \mathbb{R} \mid r < 0\}$
0	$\{0\}$
$+$	$\{r \in \mathbb{R} \mid r > 0\}$
-0	$\{r \in \mathbb{R} \mid r \leq 0\}$
$0+$	$\{r \in \mathbb{R} \mid r \geq 0\}$
$\top$	$\mathbb{R}$

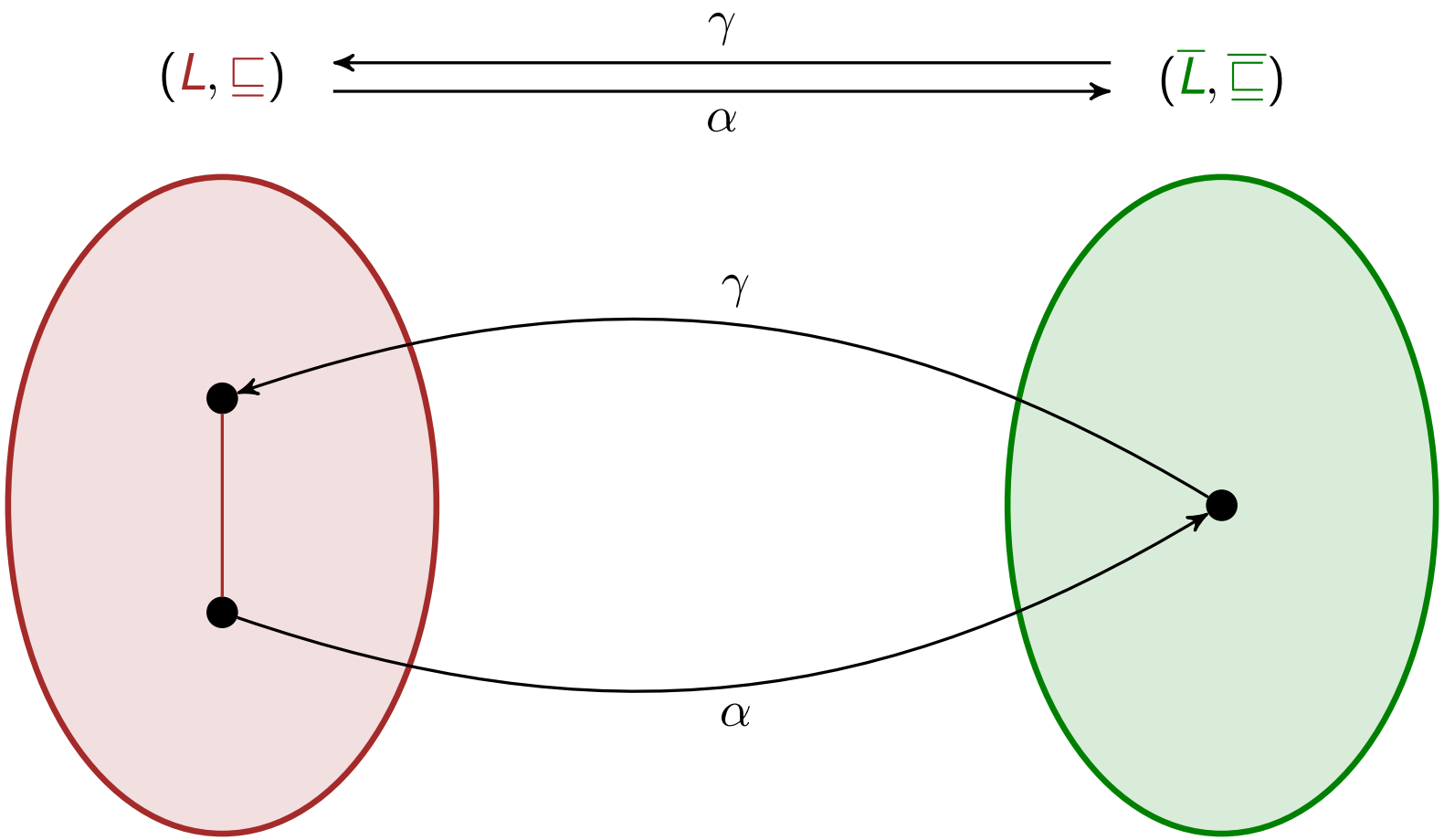
Galois Connections: Characterization

Consider two lattices $(L, \sqsubseteq)$ and $(\overline{L}, \overline{\sqsubseteq})$.

For any two functions $\alpha : L \to \overline{L}$ et $\gamma : \overline{L} \to L$, we have

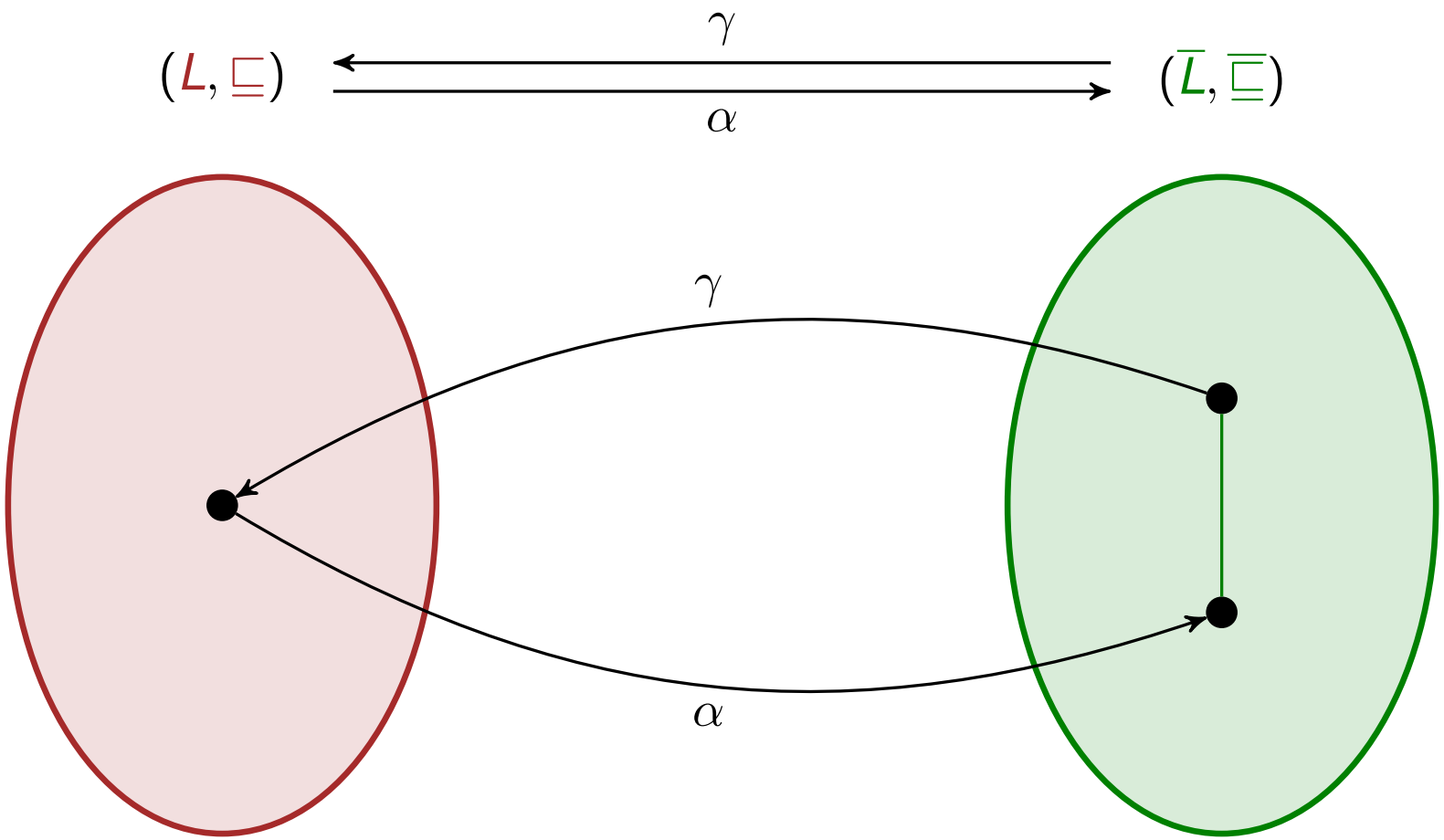
$$(L, \sqsubseteq) \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} (\overline{L}, \overline{\sqsubseteq}) \qquad \text{iff} \qquad \begin{cases} x \sqsubseteq \gamma \circ \alpha(x) & (\text{for all } x \in L) \\ \alpha \circ \gamma(\overline{y}) \,\overline{\sqsubseteq}\, \overline{y} & (\text{for all } \overline{y} \in \overline{L}) \\ \alpha \text{ is monotonic} \\ \gamma \text{ is monotonic} \end{cases}$$

Galois Connections: Characterization



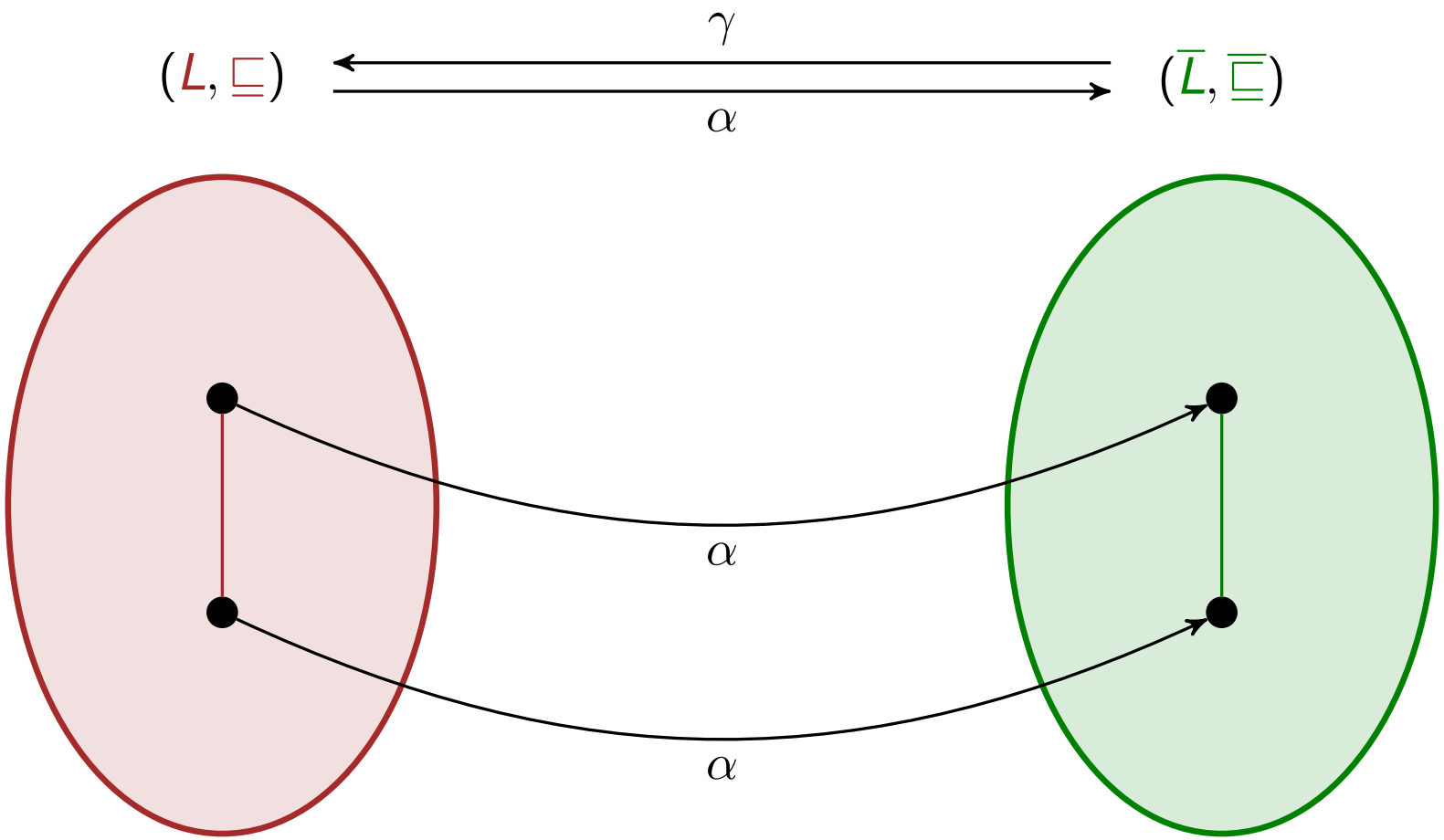
$$x \sqsubseteq \gamma \circ \alpha(x) \qquad (\gamma \circ \alpha \text{ extensive})$$

Galois Connections: Characterization



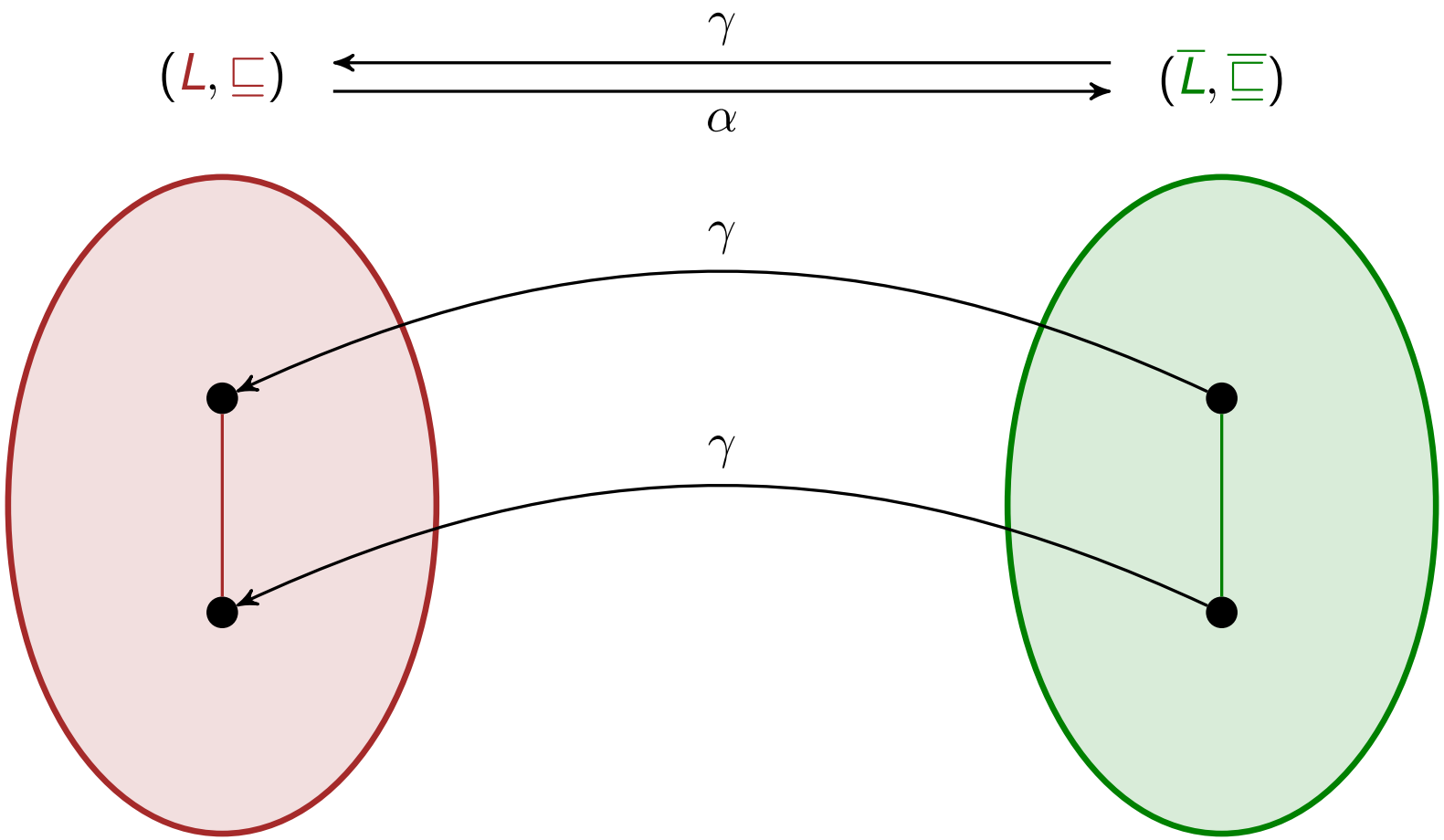
$$\alpha \circ \gamma(\overline{y}) \ \overline{\sqsubseteq} \ \overline{y} \qquad (\alpha \circ \gamma \text{ reductive})$$

Galois Connections: Characterization



α is monotonic

Galois Connections: Characterization



γ is monotonic

Galois Connections: Properties

For any Galois connection $(L, \sqsubseteq) \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} (\overline{L}, \overline{\sqsubseteq})$, we have

$$\alpha = \alpha \circ \gamma \circ \alpha \qquad \gamma = \gamma \circ \alpha \circ \gamma$$

α is surjective iff γ is injective iff $\alpha \circ \gamma = \lambda\, \overline{y}\, .\, \overline{y}$

Definition

A Galois insertion between a lattice $(L, \sqsubseteq)$ and a lattice $(\overline{L}, \overline{\sqsubseteq})$ is any Galois connection $(L, \sqsubseteq) \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} (\overline{L}, \overline{\sqsubseteq})$ where α is surjective.

Notation for Galois insertions: $(L, \sqsubseteq) \underset{\alpha}{\overset{\gamma}{\leftrightarrows\!\!\!\!\!\rightarrow}} (\overline{L}, \overline{\sqsubseteq})$

Galois Connections: Properties

For any Galois connection $(L, \sqsubseteq) \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} (\overline{L}, \overline{\sqsubseteq})$, we have

$$\alpha = \alpha \circ \gamma \circ \alpha \qquad \gamma = \gamma \circ \alpha \circ \gamma$$

α is surjective iff γ is injective iff $\alpha \circ \gamma = \lambda\, \overline{y}\, .\, \overline{y}$

Definition

A Galois insertion between a lattice $(L, \sqsubseteq)$ and a lattice $(\overline{L}, \overline{\sqsubseteq})$ is any Galois connection $(L, \sqsubseteq) \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} (\overline{L}, \overline{\sqsubseteq})$ where α is surjective.

Notation for Galois insertions: $(L, \sqsubseteq) \underset{\alpha}{\overset{\gamma}{\leftrightarrows\!\!\!\!\rightarrow}} (\overline{L}, \overline{\sqsubseteq})$

Galois Connections: Properties

For any Galois connection $(L, \sqsubseteq) \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} (\overline{L}, \overline{\sqsubseteq})$, we have

$$\alpha = \alpha \circ \gamma \circ \alpha \qquad \gamma = \gamma \circ \alpha \circ \gamma$$

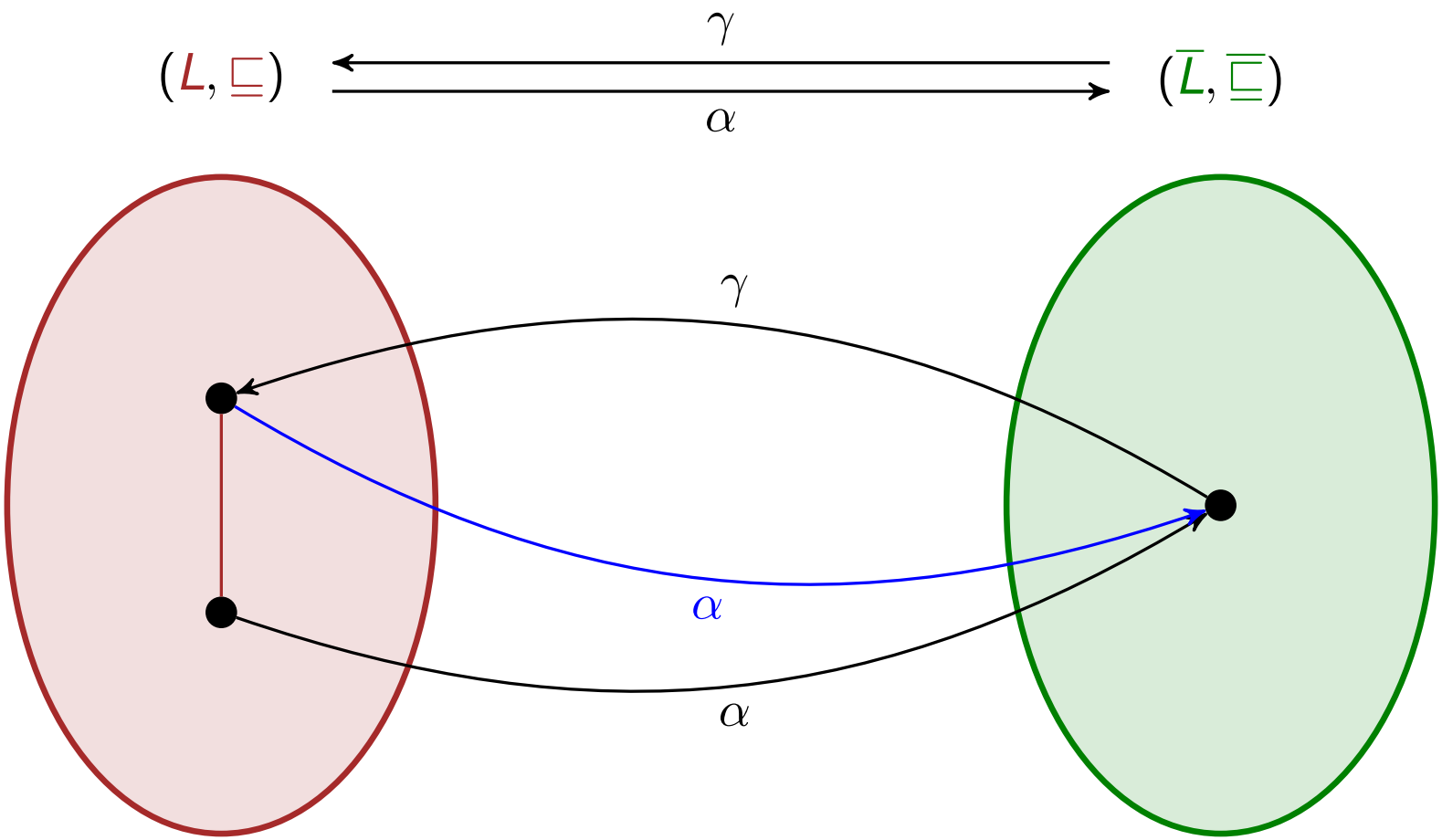
α is surjective iff γ is injective iff $\alpha \circ \gamma = \lambda\, \overline{y}\, .\, \overline{y}$

Definition

A Galois insertion between a lattice $(L, \sqsubseteq)$ and a lattice $(\overline{L}, \overline{\sqsubseteq})$ is any Galois connection $(L, \sqsubseteq) \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} (\overline{L}, \overline{\sqsubseteq})$ where α is surjective.

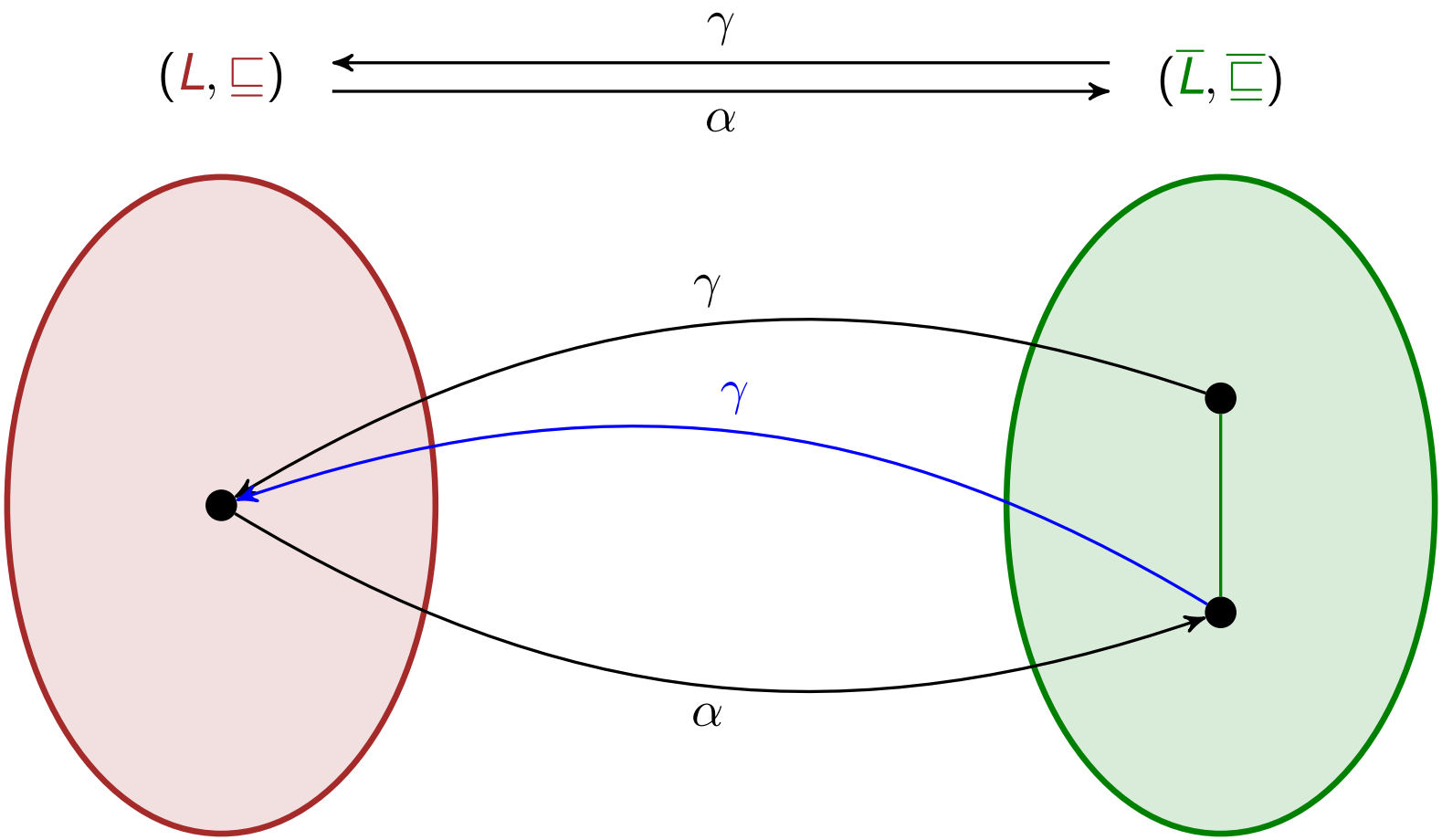
Notation for Galois insertions: $(L, \sqsubseteq) \underset{\alpha}{\overset{\gamma}{\leftrightarrows\!\!\!\!\!\!\rightarrow}} (\overline{L}, \overline{\sqsubseteq})$

Galois Connections: Properties



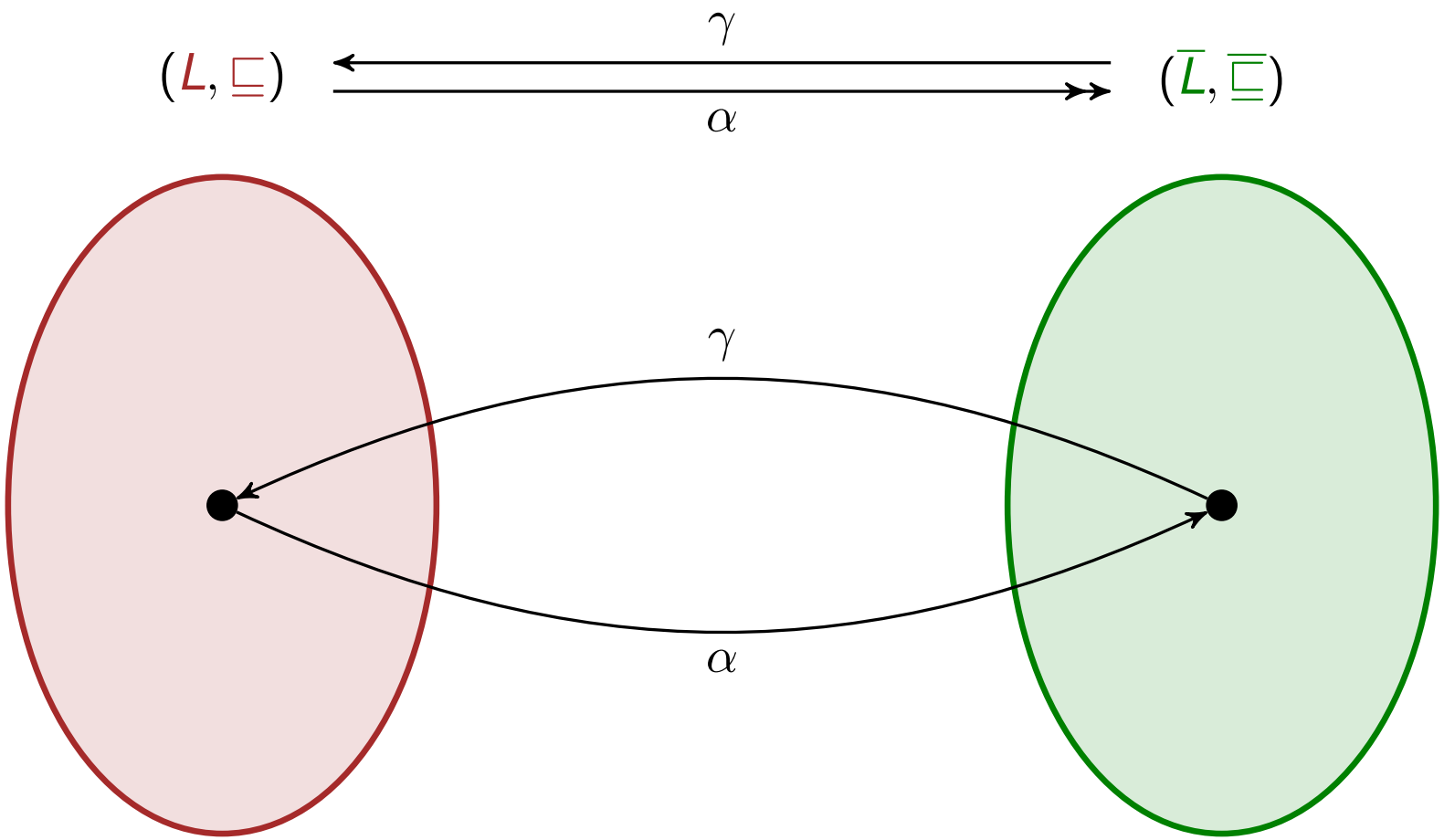
$$\alpha = \alpha \circ \gamma \circ \alpha$$

Galois Connections: Properties



$$\gamma = \gamma \circ \alpha \circ \gamma$$

Galois Connections: Properties



$$\alpha \circ \gamma = \lambda\, \overline{y}\, .\, \overline{y} \qquad \text{(Galois insertion)}$$

Galois Connections: Properties for Complete Lattices

For any Galois connection $(L, \sqsubseteq) \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} (\overline{L}, \overline{\sqsubseteq})$ on complete lattices,

$$\alpha(x) = \overline{\bigsqcap} \{\overline{y} \in \overline{L} \mid x \sqsubseteq \gamma(\overline{y})\} \quad \text{(for all } x \in L\text{)}$$

$$\gamma(\overline{y}) = \bigsqcup \{x \in L \mid \alpha(x) \,\overline{\sqsubseteq}\, \overline{y}\} \quad \text{(for all } \overline{y} \in \overline{L}\text{)}$$

$$\alpha\left(\bigsqcup X\right) = \overline{\bigsqcup} \{\alpha(x) \mid x \in X\} \quad \text{(for all } X \subseteq L\text{)}$$

$$\gamma\left(\overline{\bigsqcap}\, \overline{Y}\right) = \bigsqcap \{\gamma(\overline{y}) \mid \overline{y} \in \overline{Y}\} \quad \text{(for all } \overline{Y} \subseteq \overline{L}\text{)}$$

Informally

α uniquely determines γ and γ uniquely determines α.

α preserves least upper bounds, γ preserves greatest lower bounds.

Best Abstraction of a Monotonic Concrete Function

Consider a Galois connection $(L, \sqsubseteq) \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} (\overline{L}, \overline{\sqsubseteq})$ on complete lattices.

Definition

For any *monotonic* function $f : L \to L$, the best abstraction of f is the *monotonic* function $f^\sharp : \overline{L} \to \overline{L}$ defined by:

$$f^\sharp \quad = \quad \alpha \circ f \circ \gamma$$

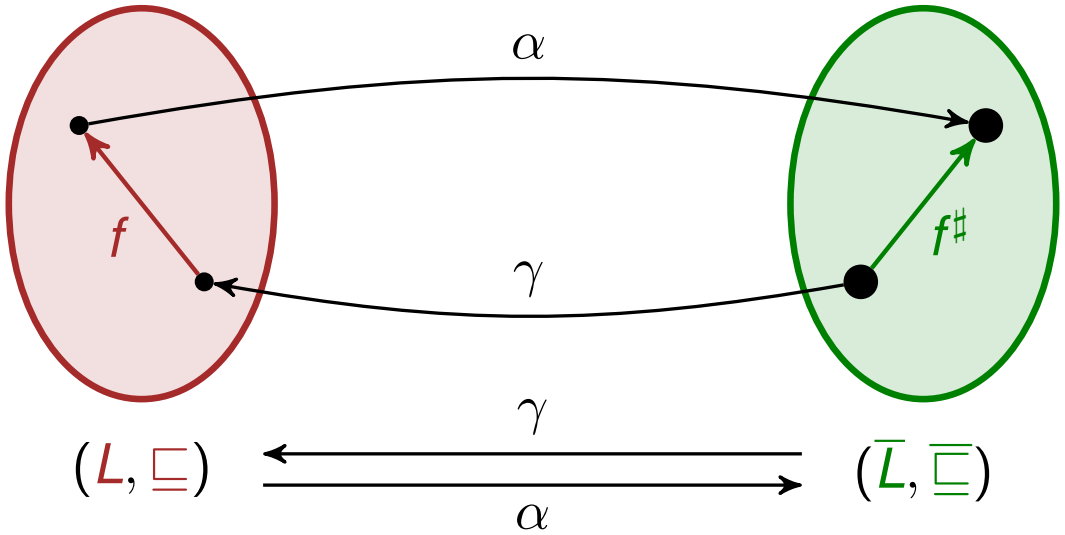
Best Abstraction of a Monotonic Concrete Function

Consider a Galois connection $(L, \sqsubseteq) \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} (\overline{L}, \overline{\sqsubseteq})$ on complete lattices.

Definition

For any *monotonic* function $f : L \to L$, the best abstraction of f is the *monotonic* function $f^\sharp : \overline{L} \to \overline{L}$ defined by:

$$f^\sharp \quad = \quad \alpha \circ f \circ \gamma$$



Best Abstraction: Justification

Given a monotonic function $f : L \longrightarrow L$, we look for a monotonic function $\overline{g} : \overline{L} \longrightarrow \overline{L}$ that is a sound approximation of f:

$$f(x) \sqsubseteq \gamma \circ \overline{g} \circ \alpha(x)$$

or equivalently (when $\overline{g}$ is monotonic):

$$\alpha(x) \mathrel{\overline{\sqsubseteq}} \overline{y} \quad \Longrightarrow \quad \overline{g}(\overline{y}) \mathrel{\overline{\sqsupseteq}} \alpha \circ f(x)$$

The most precise function satisfying the above condition is defined by:

$$\overline{g}(\overline{y}) = \overline{\bigsqcup} \{\alpha \circ f(x) \mid \alpha(x) \mathrel{\overline{\sqsubseteq}} \overline{y}\}$$

Best Abstraction: Justification

Given a monotonic function $f : L \longrightarrow L$, we look for a monotonic function $\overline{g} : \overline{L} \longrightarrow \overline{L}$ that is a sound approximation of f:

$$f(x) \sqsubseteq \gamma \circ \overline{g} \circ \alpha(x)$$

or equivalently (when $\overline{g}$ is monotonic):

$$\alpha(x) \mathrel{\overline{\sqsubseteq}} \overline{y} \quad \Longrightarrow \quad \overline{g}(\overline{y}) \sqsupseteq \alpha \circ f(x)$$

The most precise function satisfying the above condition is defined by:

$$\overline{g}(\overline{y}) = \overline{\bigsqcup} \{\alpha \circ f(x) \mid \alpha(x) \mathrel{\overline{\sqsubseteq}} \overline{y}\}$$

Best Abstraction: Justification

$$\overline{g}(\overline{y}) \quad = \quad \overline{\bigsqcup} \left\{ \alpha \circ f(x) \;\middle|\; \alpha(x) \mathrel{\overline{\sqsubseteq}} \overline{y} \right\}$$

Recall that α preserves least upper bounds, hence:

$$\overline{g}(\overline{y}) \quad = \quad \alpha \left(\bigsqcup \{ f(x) \mid x \in X \} \right)$$

where $X = \left\{ x \in L \;\middle|\; \alpha(x) \mathrel{\overline{\sqsubseteq}} \overline{y} \right\}$. Since f is monotonic,

$$\bigsqcup \{ f(x) \mid x \in X \} \quad \sqsubseteq \quad f \left(\bigsqcup X \right)$$

Recall that $\alpha \circ \gamma(\overline{y}) \mathrel{\overline{\sqsubseteq}} \overline{y}$, hence $\gamma(\overline{y}) \in X$ and we come to:

$$\bigsqcup \{ f(x) \mid x \in X \} \quad \sqsupseteq \quad f(\gamma(\overline{y}))$$

Best Abstraction: Justification

$$\overline{g}(\overline{y}) \quad = \quad \overline{\bigsqcup} \left\{ \alpha \circ f(x) \;\middle|\; \alpha(x) \mathrel{\overline{\sqsubseteq}} \overline{y} \right\}$$

Recall that α preserves least upper bounds, hence:

$$\overline{g}(\overline{y}) \quad = \quad \alpha \left(\bigsqcup \left\{ f(x) \mid x \in X \right\} \right)$$

where $X = \left\{ x \in L \;\middle|\; \alpha(x) \mathrel{\overline{\sqsubseteq}} \overline{y} \right\}$. Since f is monotonic,

$$\bigsqcup \left\{ f(x) \mid x \in X \right\} \quad \sqsubseteq \quad f \left(\bigsqcup X \right)$$

Recall that $\alpha \circ \gamma(\overline{y}) \mathrel{\overline{\sqsubseteq}} \overline{y}$, hence $\gamma(\overline{y}) \in X$ and we come to:

$$\bigsqcup \left\{ f(x) \mid x \in X \right\} \quad \sqsupseteq \quad f(\gamma(\overline{y}))$$

Best Abstraction: Justification

$$\overline{g}(\overline{y}) \quad = \quad \overline{\bigsqcup} \left\{ \alpha \circ f(x) \;\middle|\; \alpha(x) \mathrel{\overline{\sqsubseteq}} \overline{y} \right\}$$

Recall that α preserves least upper bounds, hence:

$$\overline{g}(\overline{y}) \quad = \quad \alpha \left(\bigsqcup \{ f(x) \mid x \in X \} \right)$$

where $X = \left\{ x \in L \;\middle|\; \alpha(x) \mathrel{\overline{\sqsubseteq}} \overline{y} \right\}$. Since f is monotonic,

$$\bigsqcup \{ f(x) \mid x \in X \} \quad \sqsubseteq \quad f \left(\bigsqcup X \right)$$

Recall that $\alpha \circ \gamma(\overline{y}) \mathrel{\overline{\sqsubseteq}} \overline{y}$, hence $\gamma(\overline{y}) \in X$ and we come to:

$$\bigsqcup \{ f(x) \mid x \in X \} \quad \sqsupseteq \quad f(\gamma(\overline{y}))$$

Best Abstraction: Justification

$$\overline{g}(\overline{y}) \quad = \quad \overline{\bigsqcup} \left\{ \alpha \circ f(x) \;\middle|\; \alpha(x) \mathrel{\overline{\sqsubseteq}} \overline{y} \right\}$$

Recall that α preserves least upper bounds, hence:

$$\overline{g}(\overline{y}) \quad = \quad \alpha \left(\bigsqcup \{ f(x) \mid x \in X \} \right)$$

where $X = \left\{ x \in L \;\middle|\; \alpha(x) \mathrel{\overline{\sqsubseteq}} \overline{y} \right\}$. Since f is monotonic,

$$\bigsqcup \{ f(x) \mid x \in X \} \quad \sqsubseteq \quad f \left(\bigsqcup X \right)$$

Recall that $\alpha \circ \gamma(\overline{y}) \mathrel{\overline{\sqsubseteq}} \overline{y}$, hence $\gamma(\overline{y}) \in X$ and we come to:

$$\bigsqcup \{ f(x) \mid x \in X \} \quad \sqsupseteq \quad f(\gamma(\overline{y}))$$

Best Abstraction: Justification

$$\overline{g}(\overline{y}) \quad = \quad \alpha\left(\bigsqcup \{f(x) \mid x \in X\}\right)$$

where $X = \{x \in L \mid \alpha(x) \overline{\sqsubseteq} \overline{y}\}$.

$$f(\gamma(\overline{y})) \quad \sqsubseteq \quad \bigsqcup \{f(x) \mid x \in X\} \quad \sqsubseteq \quad f\left(\bigsqcup X\right)$$

Recall that $\gamma(\overline{y}) = \bigsqcup X$, hence:

$$\bigsqcup \{f(x) \mid x \in X\} \quad = \quad f\left(\bigsqcup X\right) \quad = \quad f(\gamma(\overline{y}))$$

We obtain that:

$$\overline{g}(\overline{y}) \quad = \quad \alpha \circ f \circ \gamma(\overline{y})$$

And... all is well, since $\alpha \circ f \circ \gamma$ is monotonic!

Best Abstraction: Justification

$$\overline{g}(\overline{y}) \quad = \quad \alpha\left(\bigsqcup \{f(x) \mid x \in X\}\right)$$

where $X = \{x \in L \mid \alpha(x) \overline{\sqsubseteq} \overline{y}\}$.

$$f(\gamma(\overline{y})) \quad \sqsubseteq \quad \bigsqcup \{f(x) \mid x \in X\} \quad \sqsubseteq \quad f\left(\bigsqcup X\right)$$

Recall that $\gamma(\overline{y}) = \bigsqcup X$, hence:

$$\bigsqcup \{f(x) \mid x \in X\} \quad = \quad f\left(\bigsqcup X\right) \quad = \quad f(\gamma(\overline{y}))$$

We obtain that:

$$\overline{g}(\overline{y}) \quad = \quad \alpha \circ f \circ \gamma(\overline{y})$$

And... all is well, since $\alpha \circ f \circ \gamma$ is monotonic!

Best Abstraction: Justification

$$\overline{g}(\overline{y}) \quad = \quad \alpha\left(\bigsqcup \{f(x) \mid x \in X\}\right)$$

where $X = \left\{x \in L \;\middle|\; \alpha(x) \,\overline{\sqsubseteq}\, \overline{y}\right\}$.

$$f(\gamma(\overline{y})) \quad \sqsubseteq \quad \bigsqcup \{f(x) \mid x \in X\} \quad \sqsubseteq \quad f\left(\bigsqcup X\right)$$

Recall that $\gamma(\overline{y}) = \bigsqcup X$, hence:

$$\bigsqcup \{f(x) \mid x \in X\} \quad = \quad f\left(\bigsqcup X\right) \quad = \quad f(\gamma(\overline{y}))$$

We obtain that:

$$\overline{g}(\overline{y}) \quad = \quad \alpha \circ f \circ \gamma(\overline{y})$$

And... all is well, since $\alpha \circ f \circ \gamma$ is monotonic!

Galois Connections: Fixpoint Abstraction

Consider a Galois connection $(L, \sqsubseteq) \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} (\overline{L}, \overline{\sqsubseteq})$ on complete lattices.

Recall that for any monotonic function $f : L \to L$, we denote by $f^\sharp$ the monotonic function:

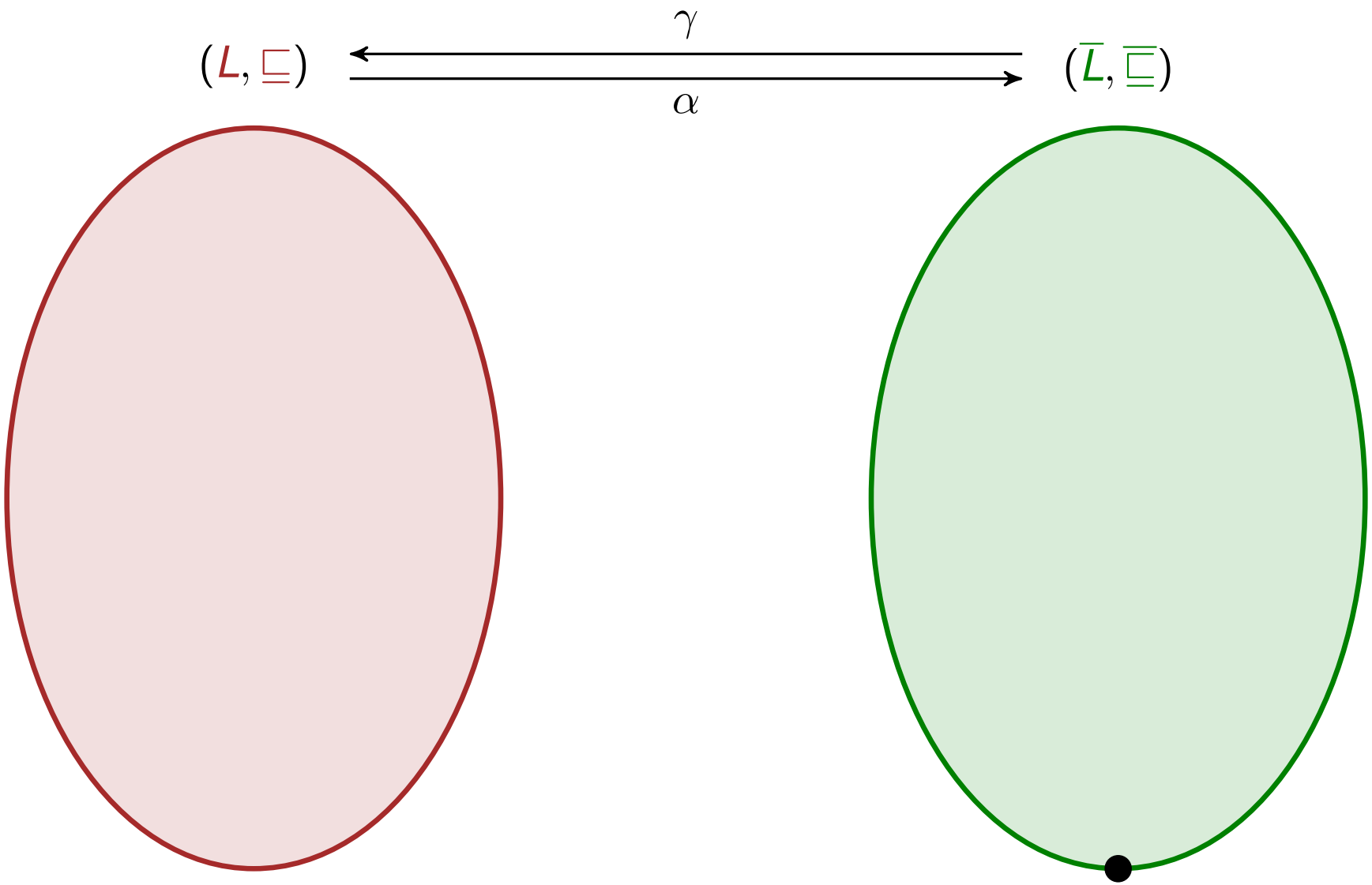
$$f^\sharp \quad = \quad \alpha \circ f \circ \gamma$$

Theorem

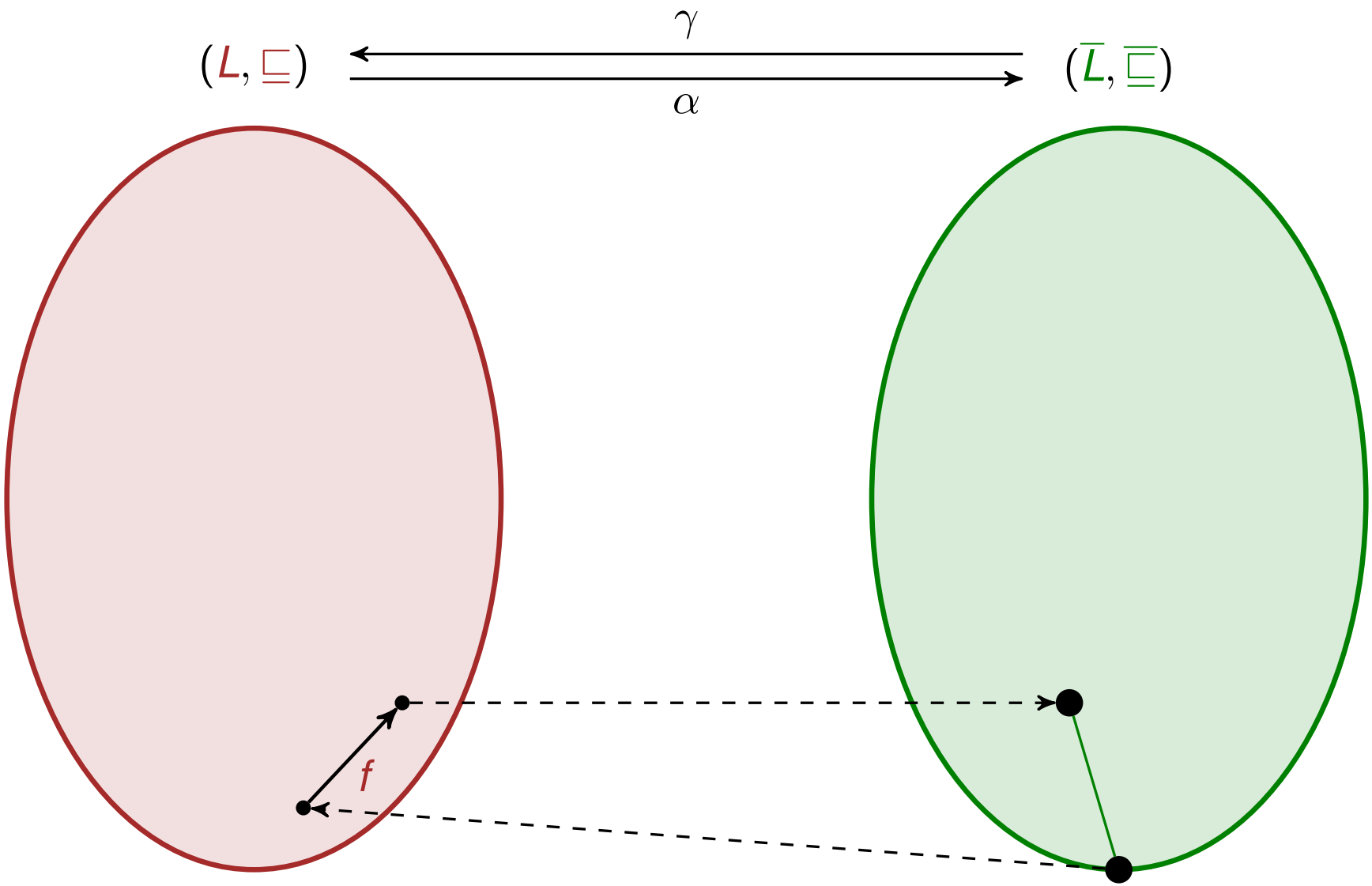
For any monotonic function $f : L \to L$, the least fixpoints of f and $f^\sharp$ satisfy:

$$\mathrm{lfp}(f) \quad \sqsubseteq \quad \gamma\left(\mathrm{lfp}(f^\sharp)\right)$$

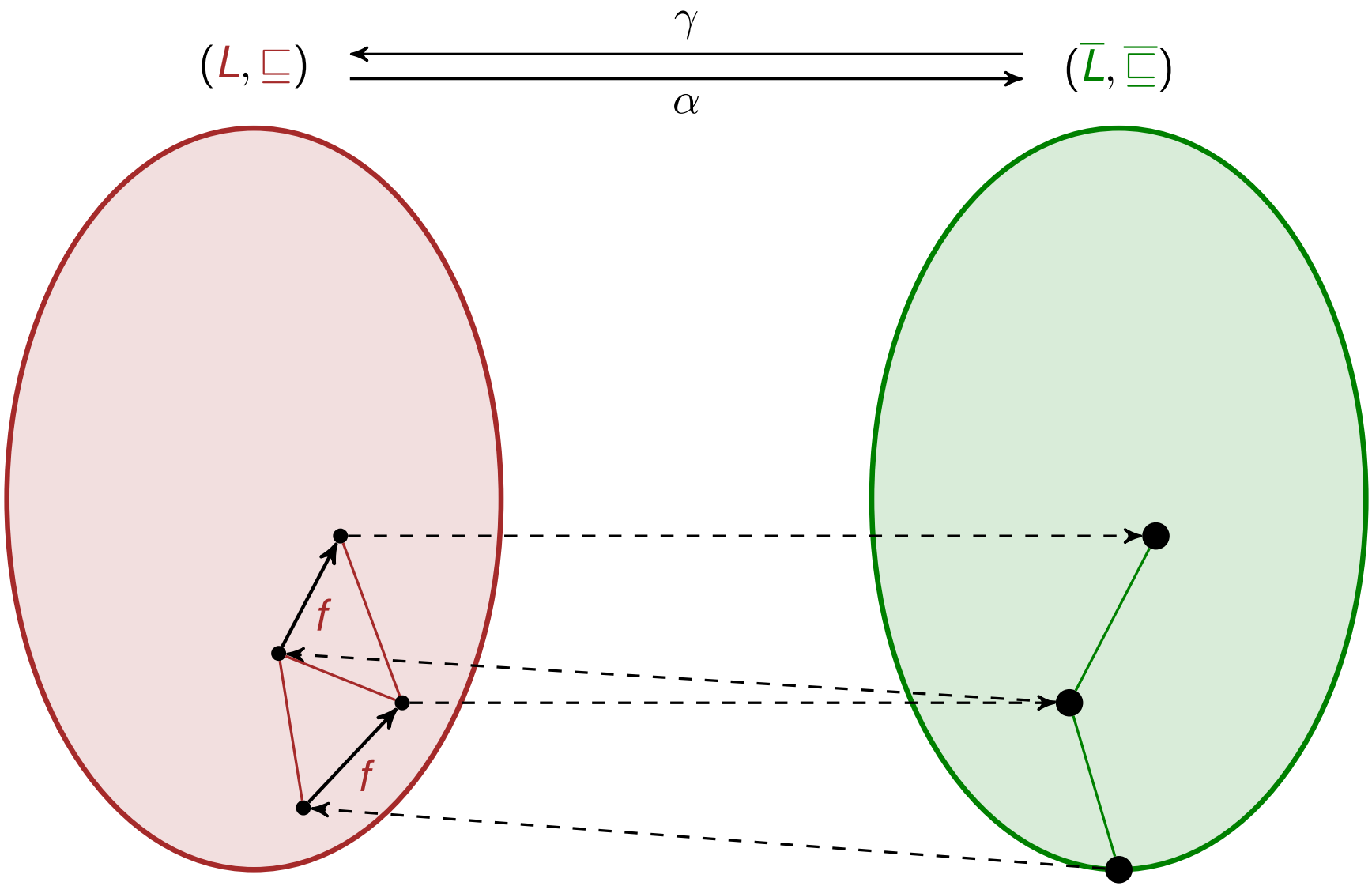
Galois Connections: Fixpoint Abstraction



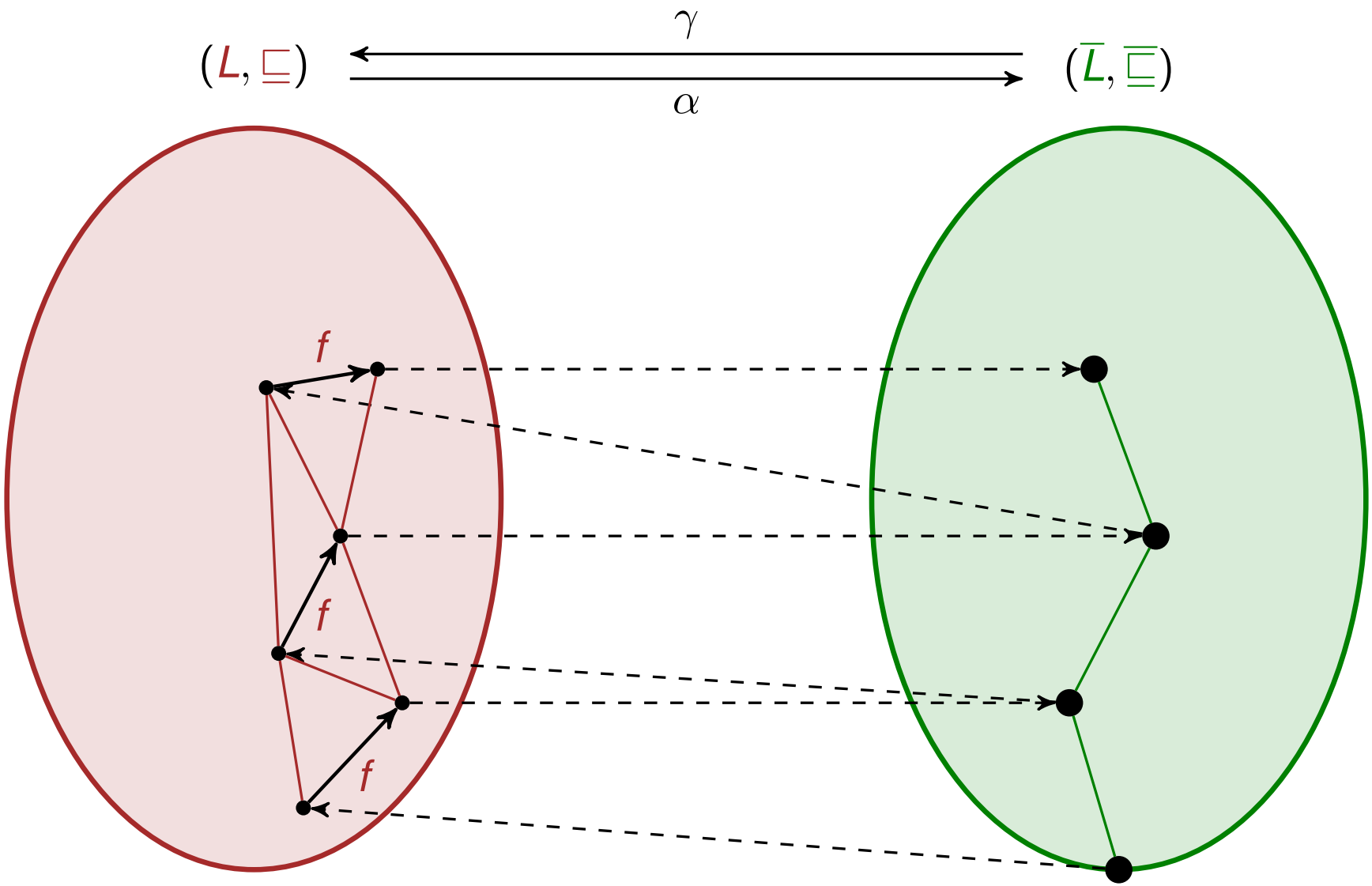
Galois Connections: Fixpoint Abstraction



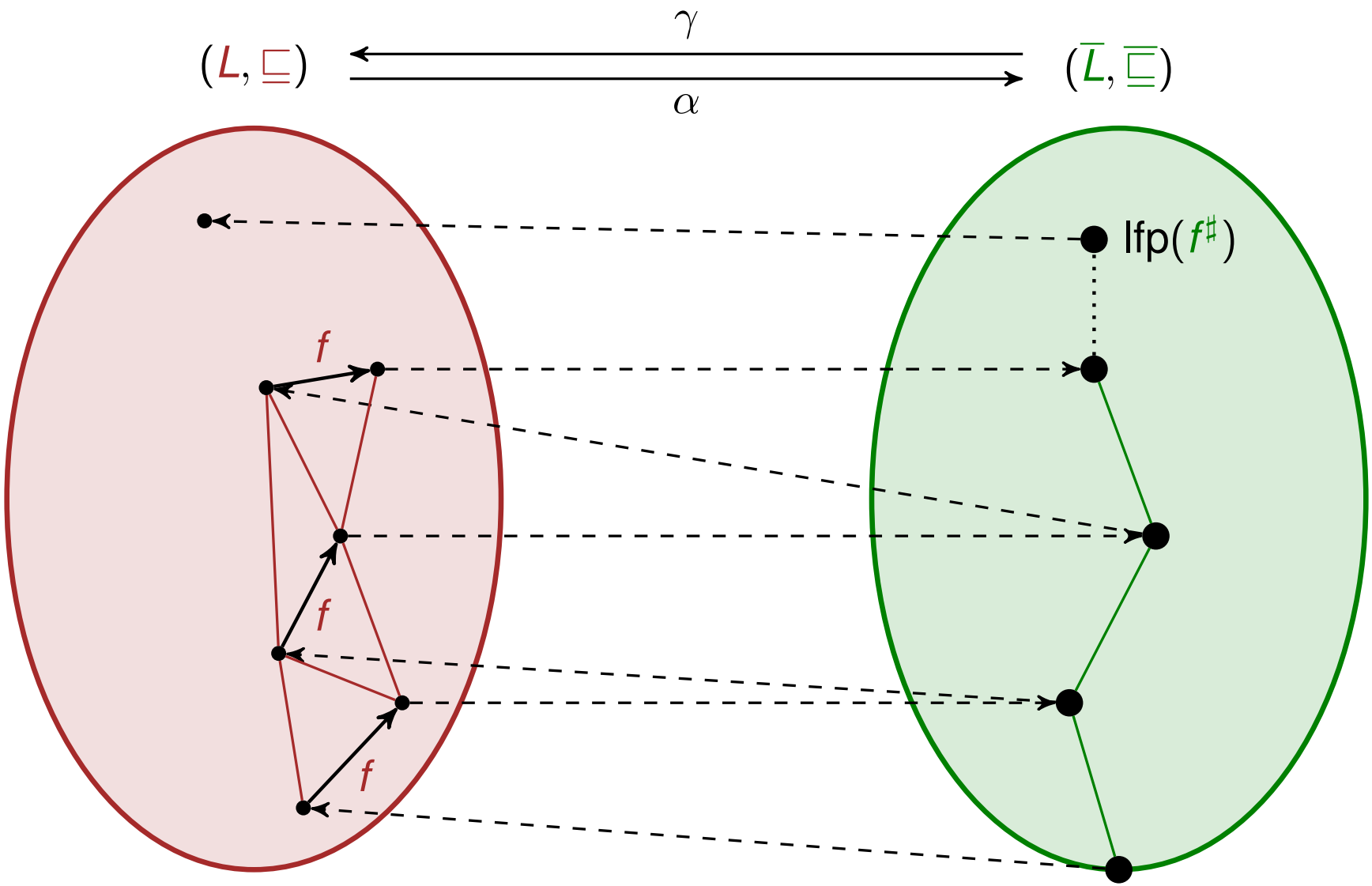
Galois Connections: Fixpoint Abstraction



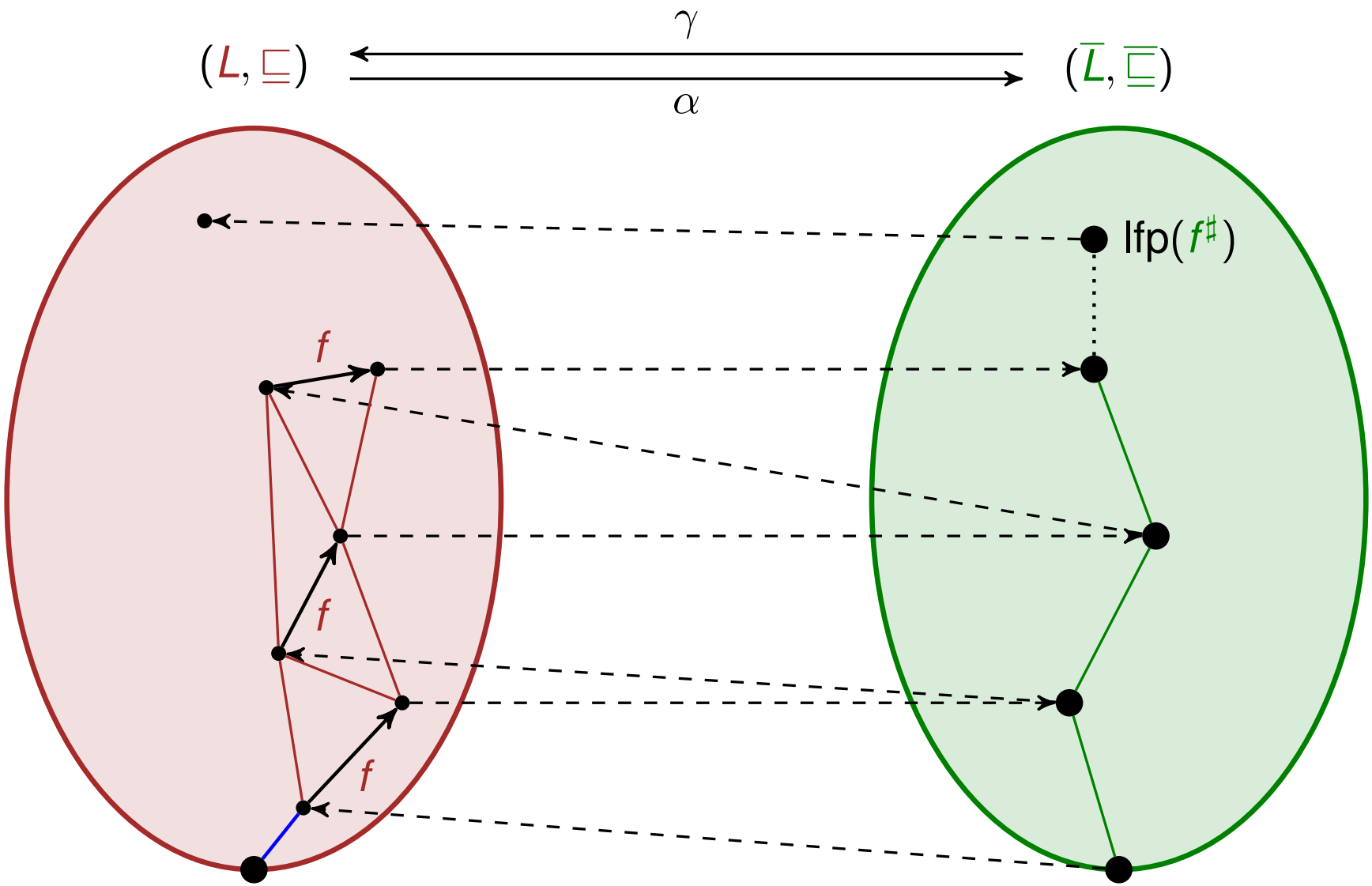
Galois Connections: Fixpoint Abstraction



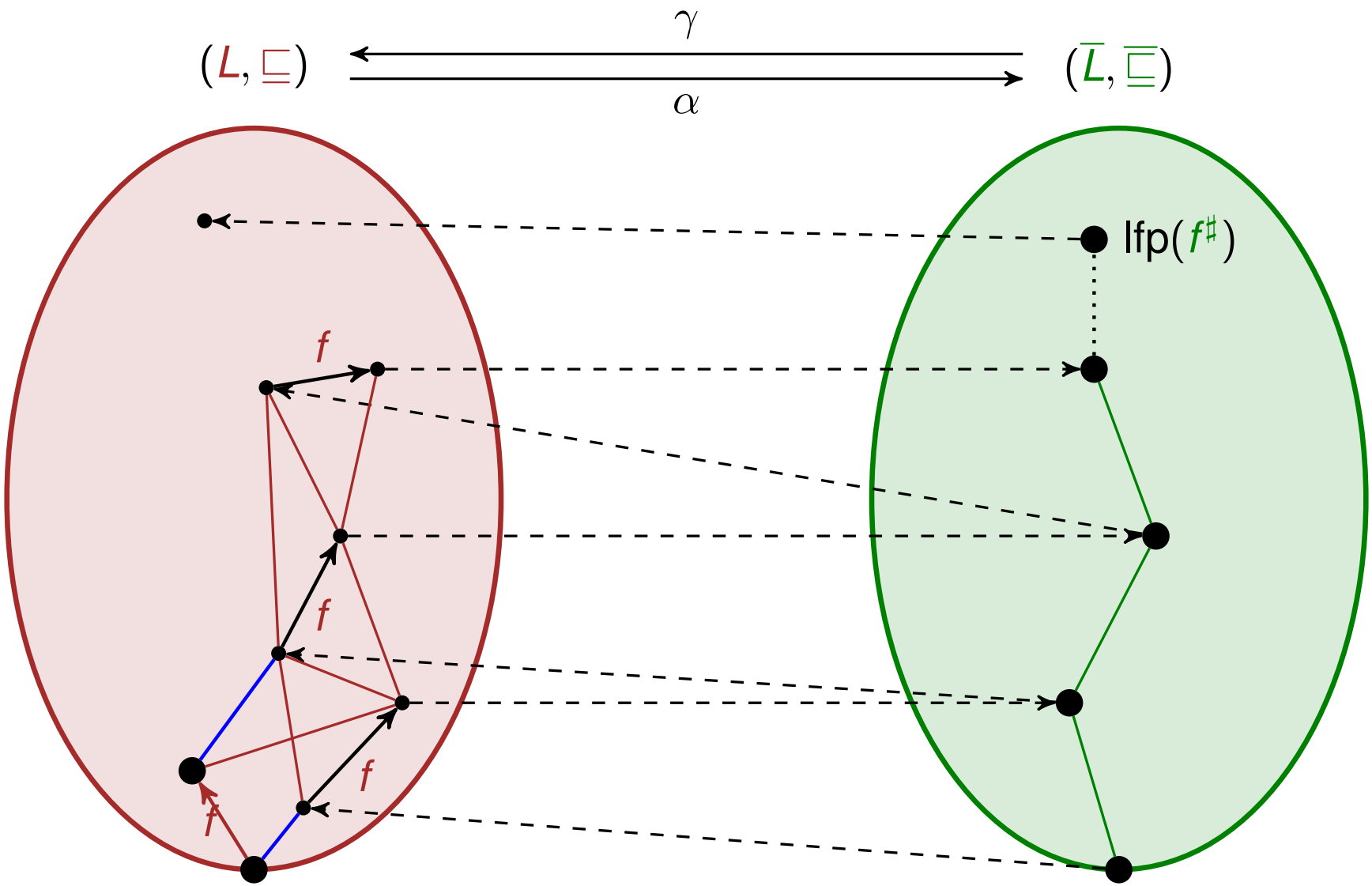
Galois Connections: Fixpoint Abstraction



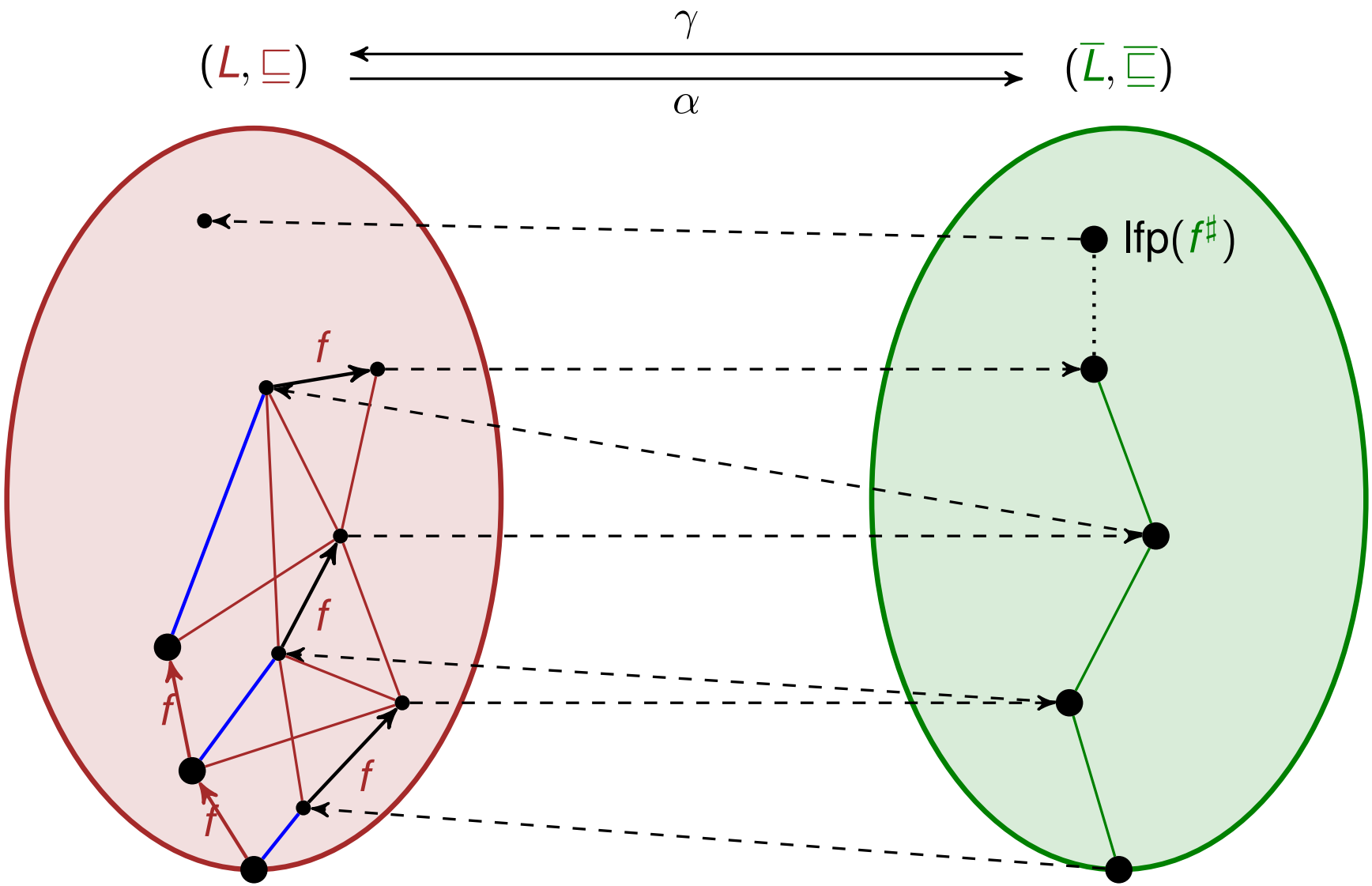
Galois Connections: Fixpoint Abstraction



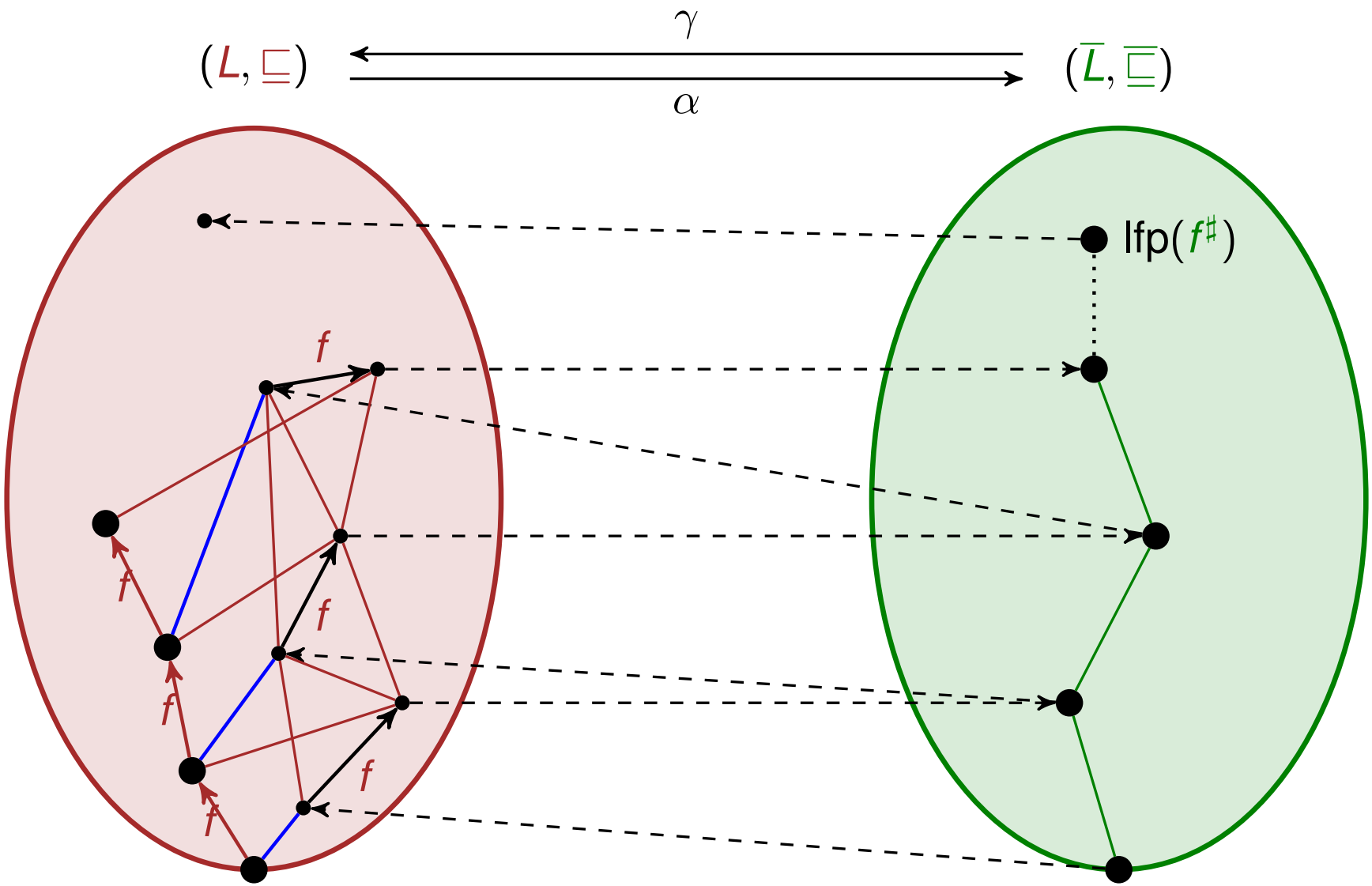
Galois Connections: Fixpoint Abstraction



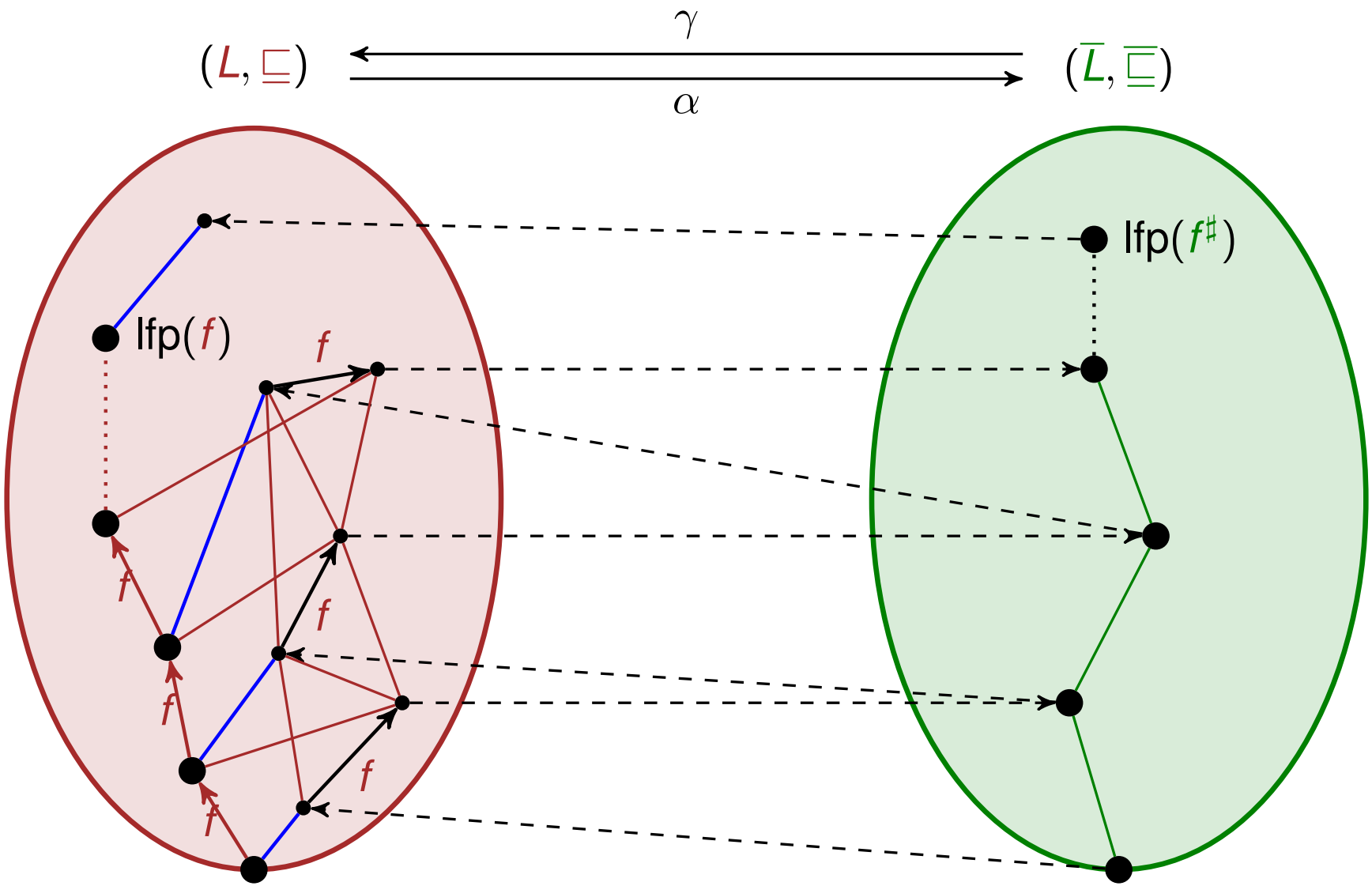
Galois Connections: Fixpoint Abstraction



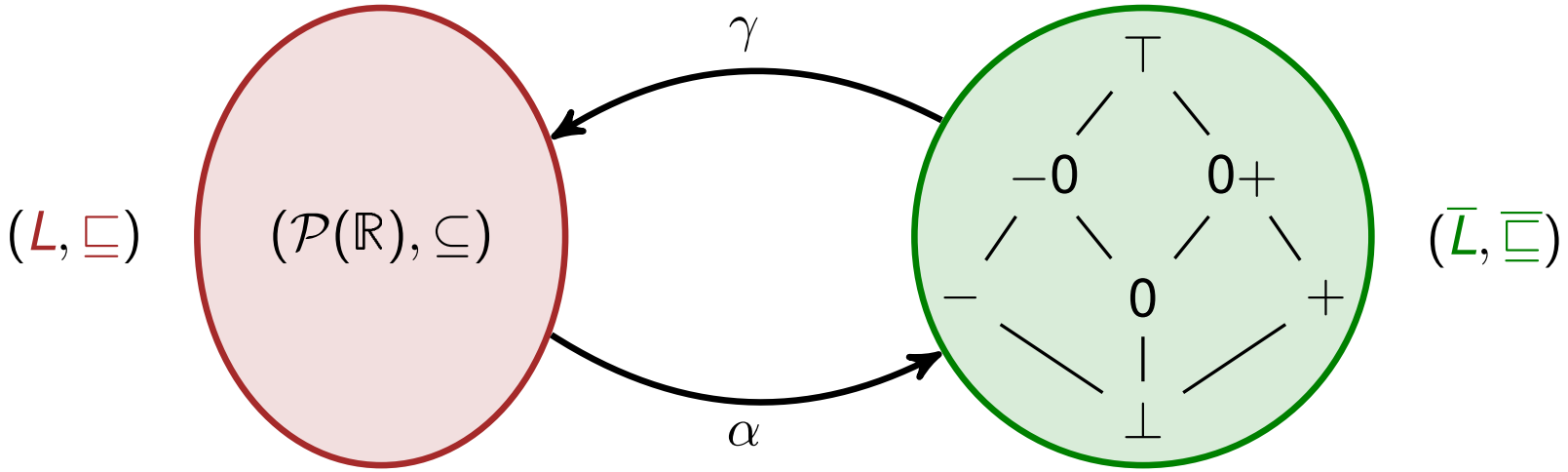
Galois Connections: Fixpoint Abstraction



Galois Connections: Fixpoint Abstraction



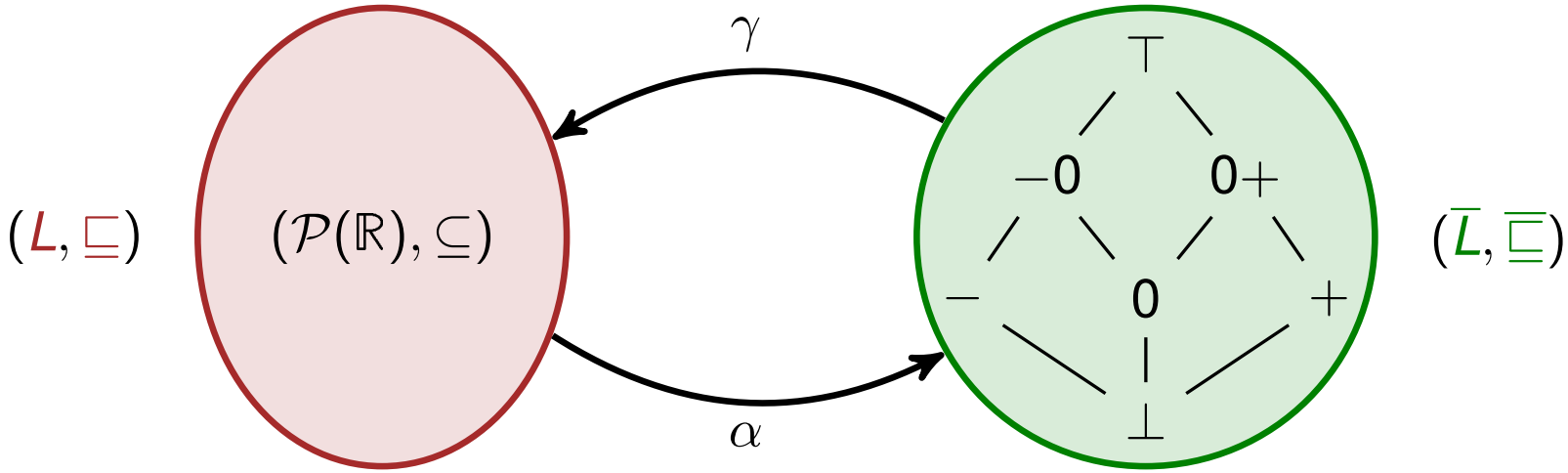
Best Abstraction & Fixpoint Abstraction: Example



f		
$\mathcal{P}(\mathbb{R})$	$\rightarrow$	$\mathcal{P}(\mathbb{R})$
x	$\mapsto$	$\{r+2 \mid r \in x\} \cup \{5\}$

$f^\sharp$							
$\overline{y}$	$\bot$	$-$	0	$+$	-0	$0+$	$\top$
$f^\sharp(\overline{y})$							

Best Abstraction & Fixpoint Abstraction: Example

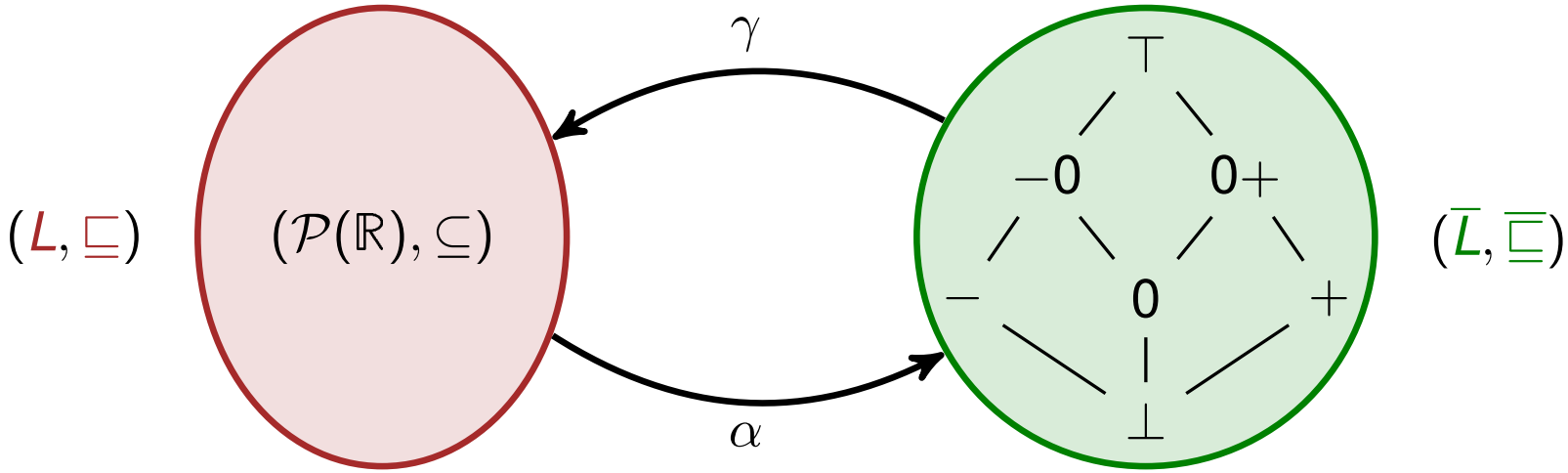


f		
$\mathcal{P}(\mathbb{R})$	$\rightarrow$	$\mathcal{P}(\mathbb{R})$
x	$\mapsto$	$\{r+2 \mid r \in x\} \cup \{5\}$

$f^\sharp$							
$\overline{y}$	$\bot$	$-$	0	$+$	-0	$0+$	$\top$
$f^\sharp(\overline{y})$	$+$						

$$
\begin{aligned}
f^\sharp(\bot) &= \alpha \circ f \circ \gamma(\bot) \\
&= \alpha \circ f(\emptyset) \\
&= \alpha(\{5\}) \\
&= +
\end{aligned}
$$

Best Abstraction & Fixpoint Abstraction: Example

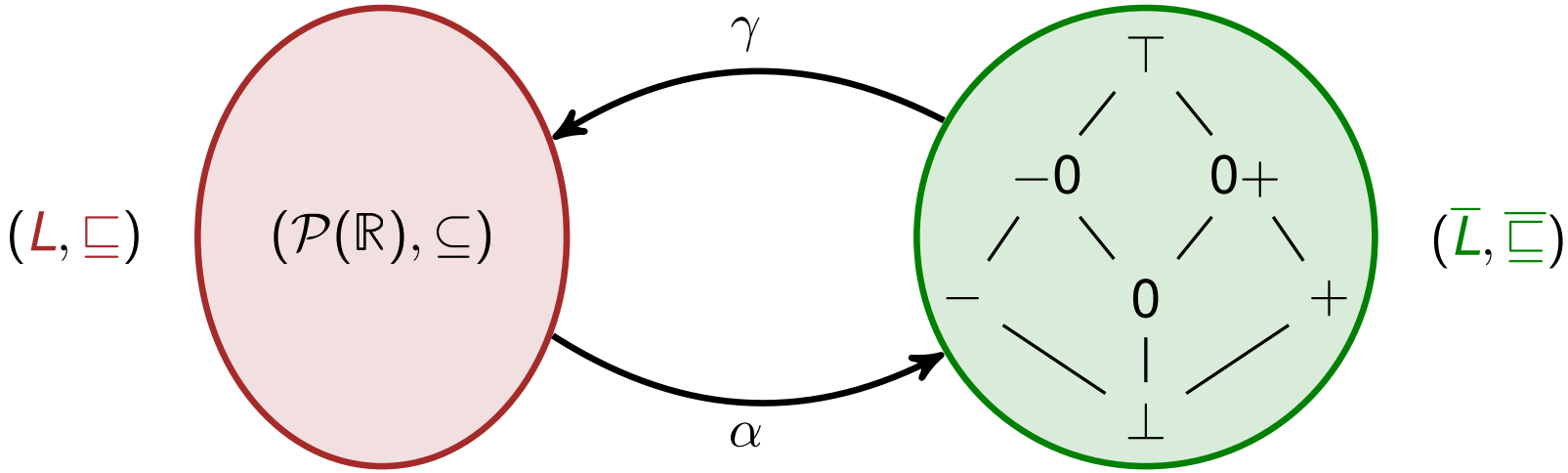


f		
$\mathcal{P}(\mathbb{R})$	$\rightarrow$	$\mathcal{P}(\mathbb{R})$
x	$\mapsto$	$\{r+2 \mid r \in x\} \cup \{5\}$

$f^\sharp$							
$\overline{y}$	$\bot$	$-$	0	$+$	-0	$0+$	$\top$
$f^\sharp(\overline{y})$	$+$	$\top$					

$$
\begin{aligned}
f^\sharp(-) &= \alpha \circ f \circ \gamma(-) \\
&= \alpha \circ f(\{r \in \mathbb{R} \mid r < 0\}) \\
&= \alpha(\{r+2 \mid r < 0\} \cup \{5\}) \\
&= \top
\end{aligned}
$$

Best Abstraction & Fixpoint Abstraction: Example

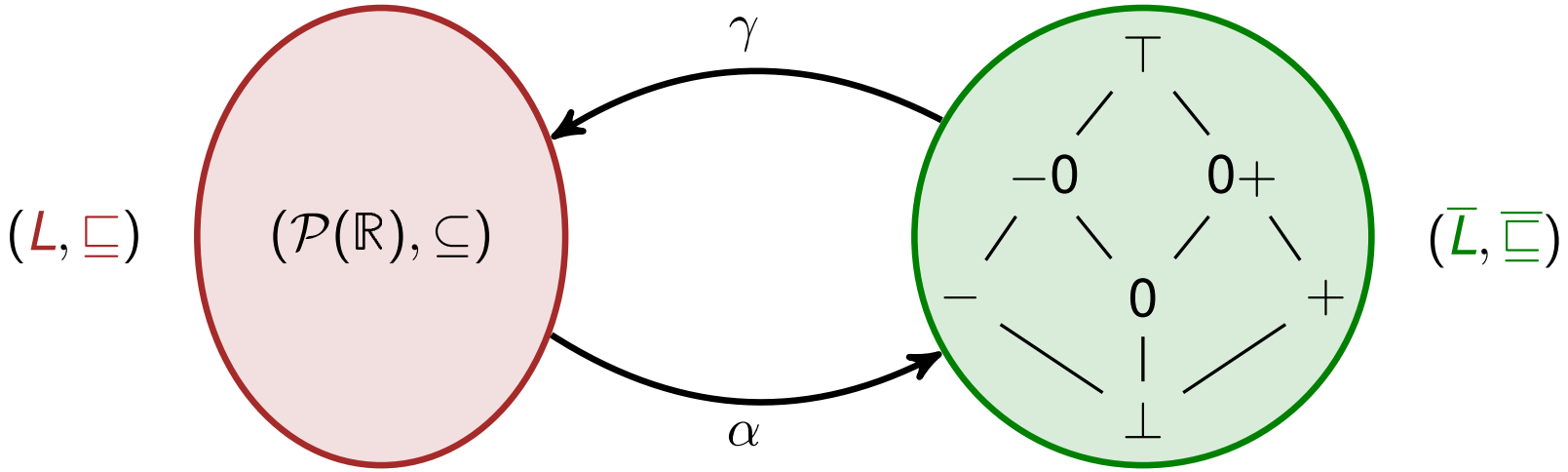


f		
$\mathcal{P}(\mathbb{R})$	$\rightarrow$	$\mathcal{P}(\mathbb{R})$
x	$\mapsto$	$\{r+2 \mid r \in x\} \cup \{5\}$

$f^\sharp$							
$\overline{y}$	$\bot$	$-$	0	$+$	-0	$0+$	$\top$
$f^\sharp(\overline{y})$	$+$	$\top$	$+$				

$$
\begin{aligned}
f^\sharp(0) &= \alpha \circ f \circ \gamma(0) \\
&= \alpha \circ f(\{0\}) \\
&= \alpha(\{2\} \cup \{5\}) \\
&= +
\end{aligned}
$$

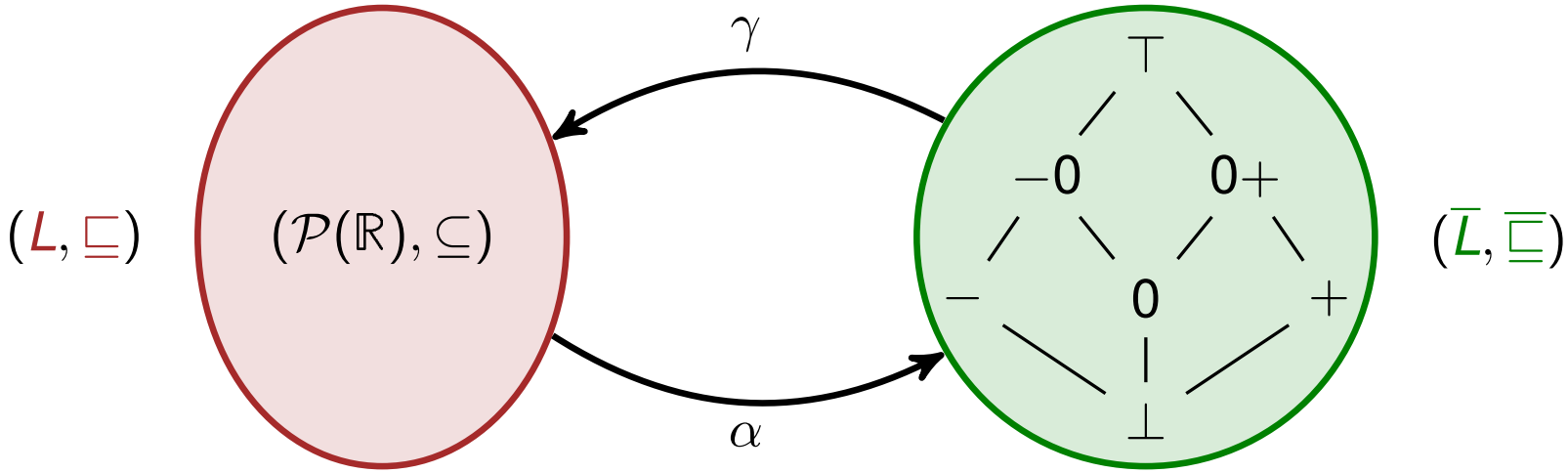
Best Abstraction & Fixpoint Abstraction: Example



f		
$\mathcal{P}(\mathbb{R})$	$\rightarrow$	$\mathcal{P}(\mathbb{R})$
x	$\mapsto$	$\{r+2 \mid r \in x\} \cup \{5\}$

$f^\sharp$							
$\overline{y}$	$\bot$	$-$	0	$+$	-0	$0+$	$\top$
$f^\sharp(\overline{y})$	$+$	$\top$	$+$	$+$	$\top$	$+$	$\top$

Best Abstraction & Fixpoint Abstraction: Example

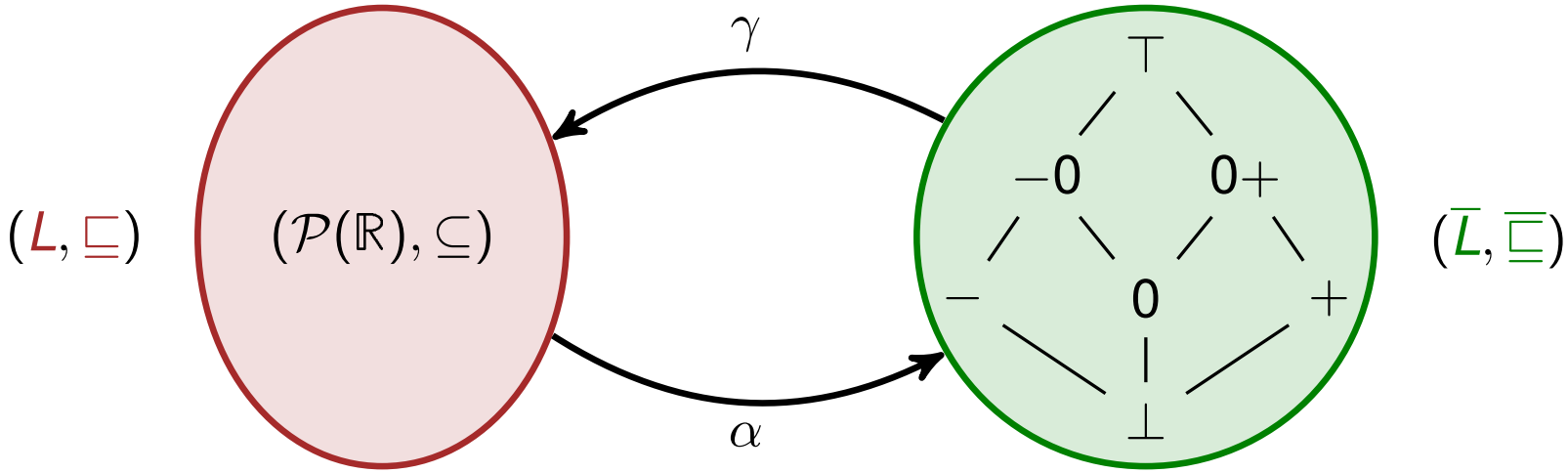


f
$\mathcal{P}(\mathbb{R}) \rightarrow \mathcal{P}(\mathbb{R})$
$x \mapsto \{r+2 \mid r \in x\} \cup \{5\}$

$$
\begin{aligned}
f(\emptyset) &= \{5\} \\
f^2(\emptyset) &= \{5, 7\} \\
f^3(\emptyset) &= \{5, 7, 9\} \\
\text{lfp } f &= \{5 + 2k \mid k \in \mathbb{N}\}
\end{aligned}
$$

$f^\sharp$							
$\overline{y}$	$\bot$	$-$	0	$+$	-0	$0+$	$\top$
$f^\sharp(\overline{y})$	$+$	$\top$	$+$	$+$	$\top$	$+$	$\top$

Best Abstraction & Fixpoint Abstraction: Example



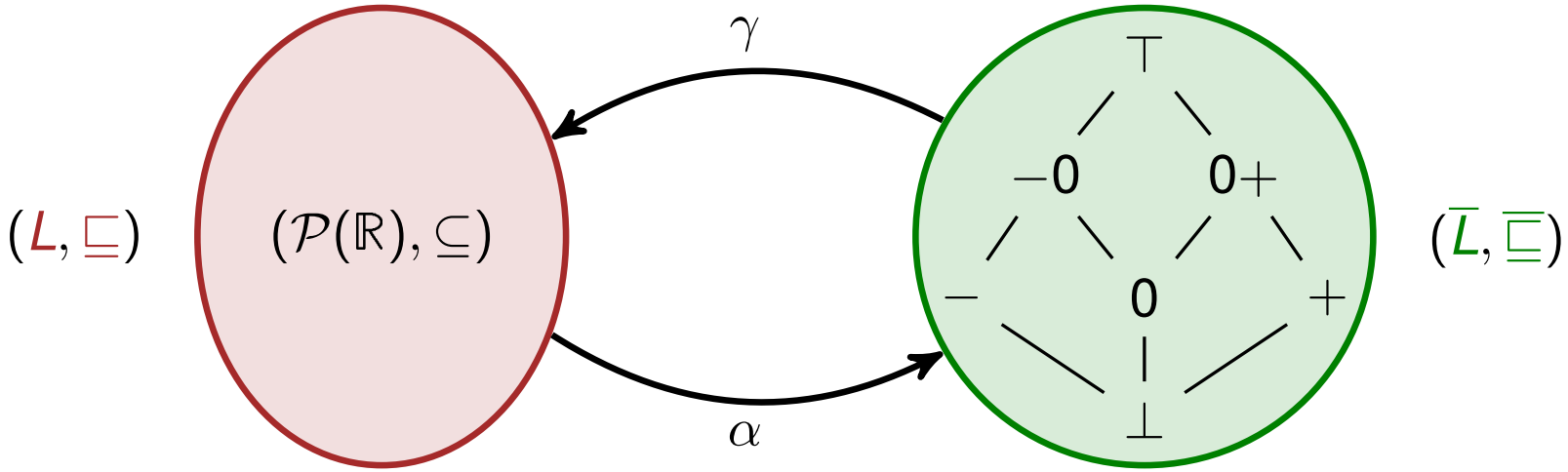
f		
$\mathcal{P}(\mathbb{R})$	$\rightarrow$	$\mathcal{P}(\mathbb{R})$
x	$\mapsto$	$\{r+2 \mid r \in x\} \cup \{5\}$

$$
\begin{aligned}
f(\emptyset) &= \{5\} \\
f^2(\emptyset) &= \{5, 7\} \\
f^3(\emptyset) &= \{5, 7, 9\} \\
\text{lfp}\, f &= \{5 + 2k \mid k \in \mathbb{N}\}
\end{aligned}
$$

$f^\sharp$							
$\overline{y}$	$\bot$	$-$	0	$+$	-0	$0+$	$\top$
$f^\sharp(\overline{y})$	$+$	$\top$	$+$	$+$	$\top$	$+$	$\top$

$$
\begin{aligned}
f^\sharp(\bot) &= + \\
f^{\sharp 2}(\bot) &= + \\
\text{lfp}\, f^\sharp &= +
\end{aligned}
$$

Best Abstraction & Fixpoint Abstraction: Example



f		
$\mathcal{P}(\mathbb{R})$	$\rightarrow$	$\mathcal{P}(\mathbb{R})$
x	$\mapsto$	$\{r+2 \mid r \in x\} \cup \{5\}$

$$
\begin{array}{rcl}
f(\emptyset) & = & \{5\} \\
f^2(\emptyset) & = & \{5, 7\} \\
f^3(\emptyset) & = & \{5, 7, 9\} \\
\mathrm{lfp}\, f & = & \{5 + 2k \mid k \in \mathbb{N}\}
\end{array}
$$

$f^\sharp$							
$\overline{y}$	$\bot$	$-$	0	$+$	-0	$0+$	$\top$
$f^\sharp(\overline{y})$	$+$	$\top$	$+$	$+$	$\top$	$+$	$\top$

$$\mathrm{lfp}\, f^\sharp = +$$

$$\gamma\left(\mathrm{lfp}(f^\sharp)\right) = \{r \in \mathbb{R} \mid r > 0\}$$

Galois Connections: Summary & Application

We want to "compute" the least fixpoint $\text{lfp}(f)$ of monotonic function $f : L \to L$ on a complete lattice $(L, \sqsubseteq)$.

If Kleene iteration $\bot \sqsubseteq f(\bot) \sqsubseteq \cdots \sqsubseteq f^i(\bot) \sqsubseteq \cdots$ diverges then:

1. design an abstract complete lattice $(\overline{L}, \overline{\sqsubseteq})$, simpler than $(L, \sqsubseteq)$, and formalize the "meaning" of abstract values by a Galois connection

$$(L, \sqsubseteq) \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} (\overline{L}, \overline{\sqsubseteq})$$

2. compute $\text{lfp}(f^\sharp)$, where $f^\sharp = \alpha \circ f \circ \gamma$ is the best abstraction of f.

By Fixpoint Abstraction Theorem, $\gamma\left(\text{lfp}(f^\sharp)\right)$ soundly approximates $\text{lfp}(f)$

$$\text{lfp}(f) \sqsubseteq \gamma\left(\text{lfp}(f^\sharp)\right) \quad \text{or equivalently} \quad \alpha(\text{lfp}(f)) \;\overline{\sqsubseteq}\; \text{lfp}(f^\sharp)$$

Outline — Abstract Interpretation

Short Introduction to Abstract Interpretation

Recall that to ensure correctness of data flow analyses. . .

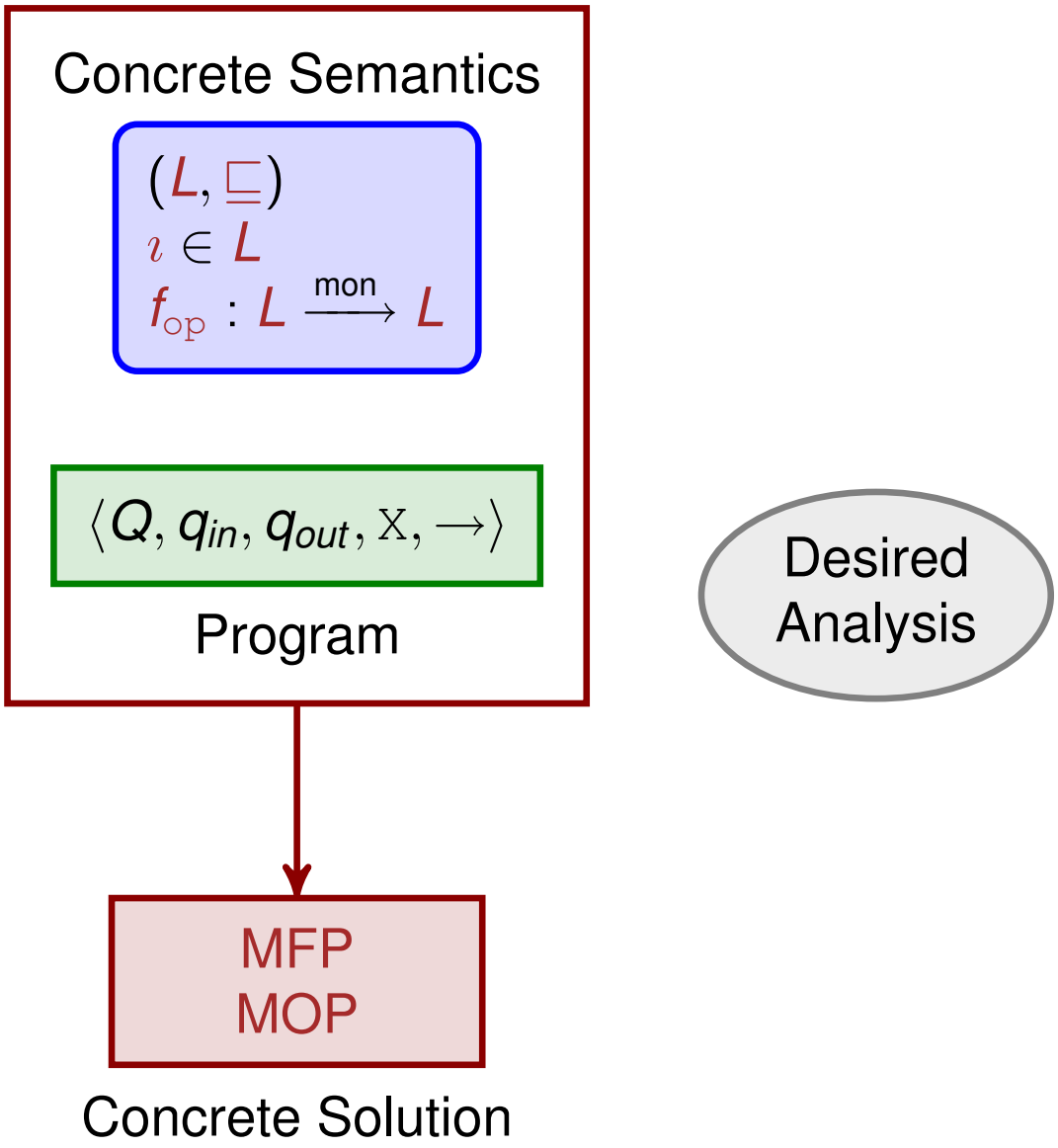
. . . we need a formal link to relate data flow facts and transfer functions with the formal semantics.

Abstract interpretation relies on Galois connections to formally express these relationships.

- Formal meaning of data flow facts by a concretization function
- Transfer mapping that soundly approximates the formal semantics
- Sound fixpoint approximation

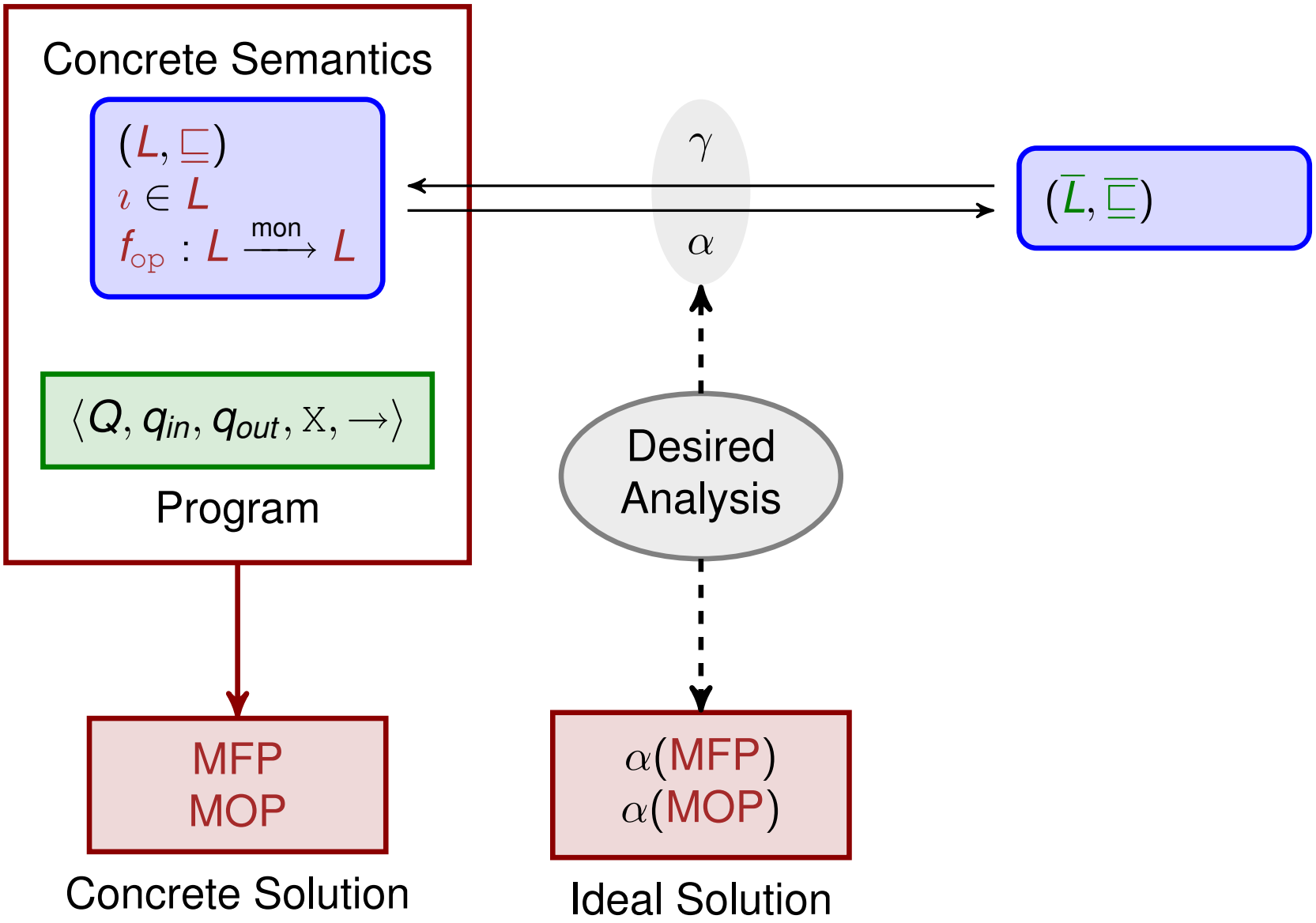
Data flow analyses that are correct by design: crucial for verification!

Systematic Design of Correct of Data Flow Analyses

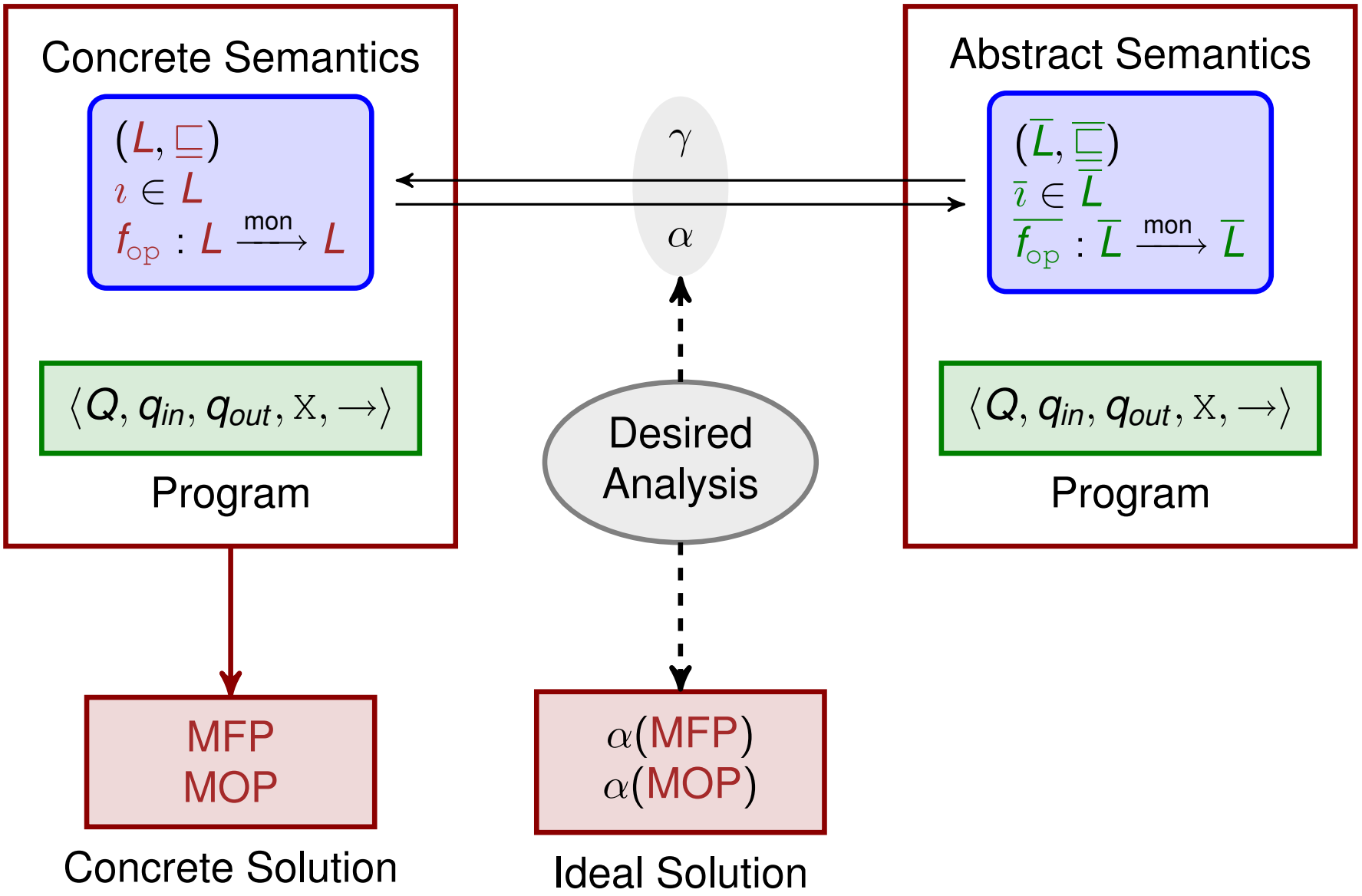


Desired Analysis

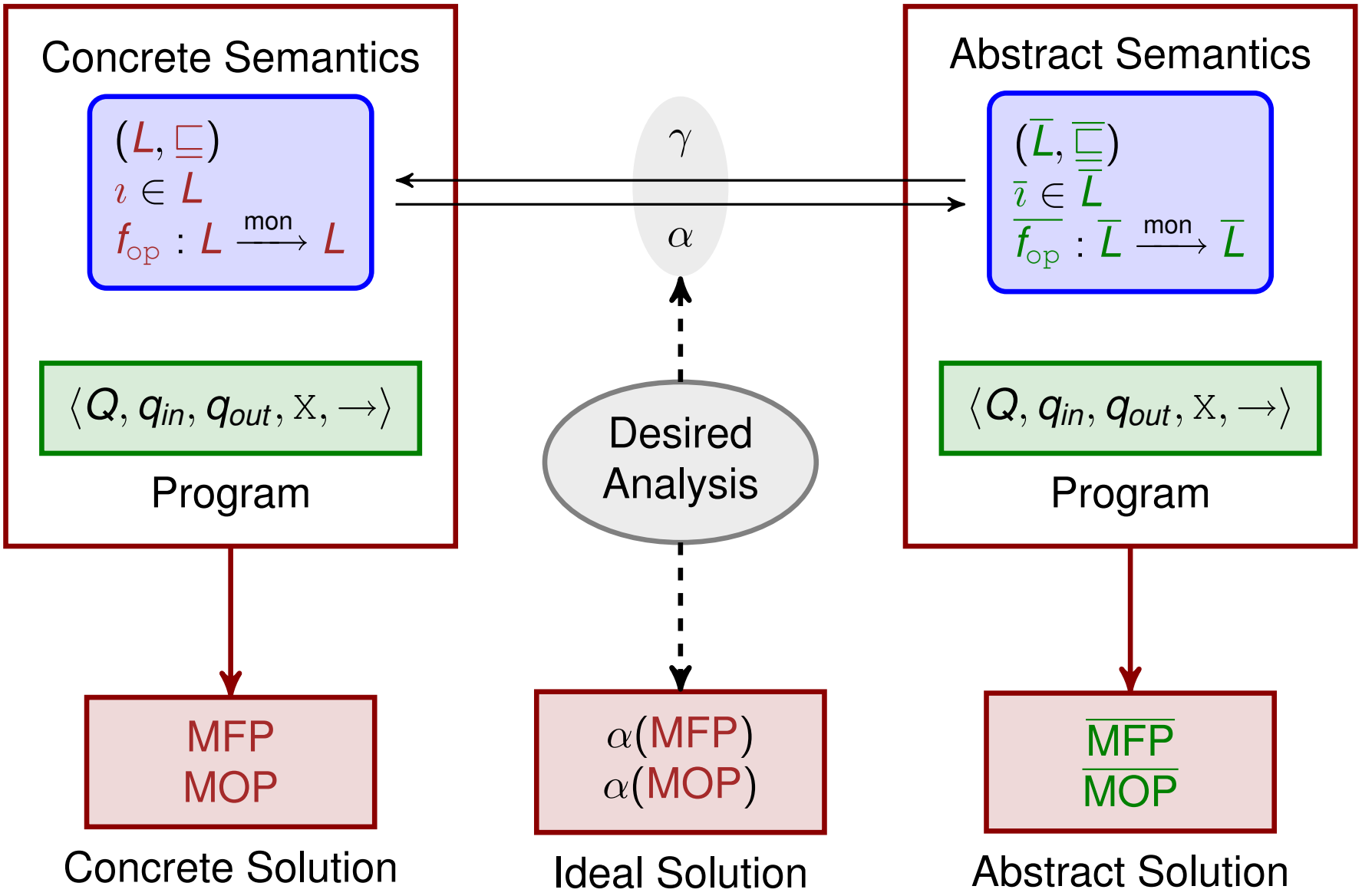
Systematic Design of Correct of Data Flow Analyses



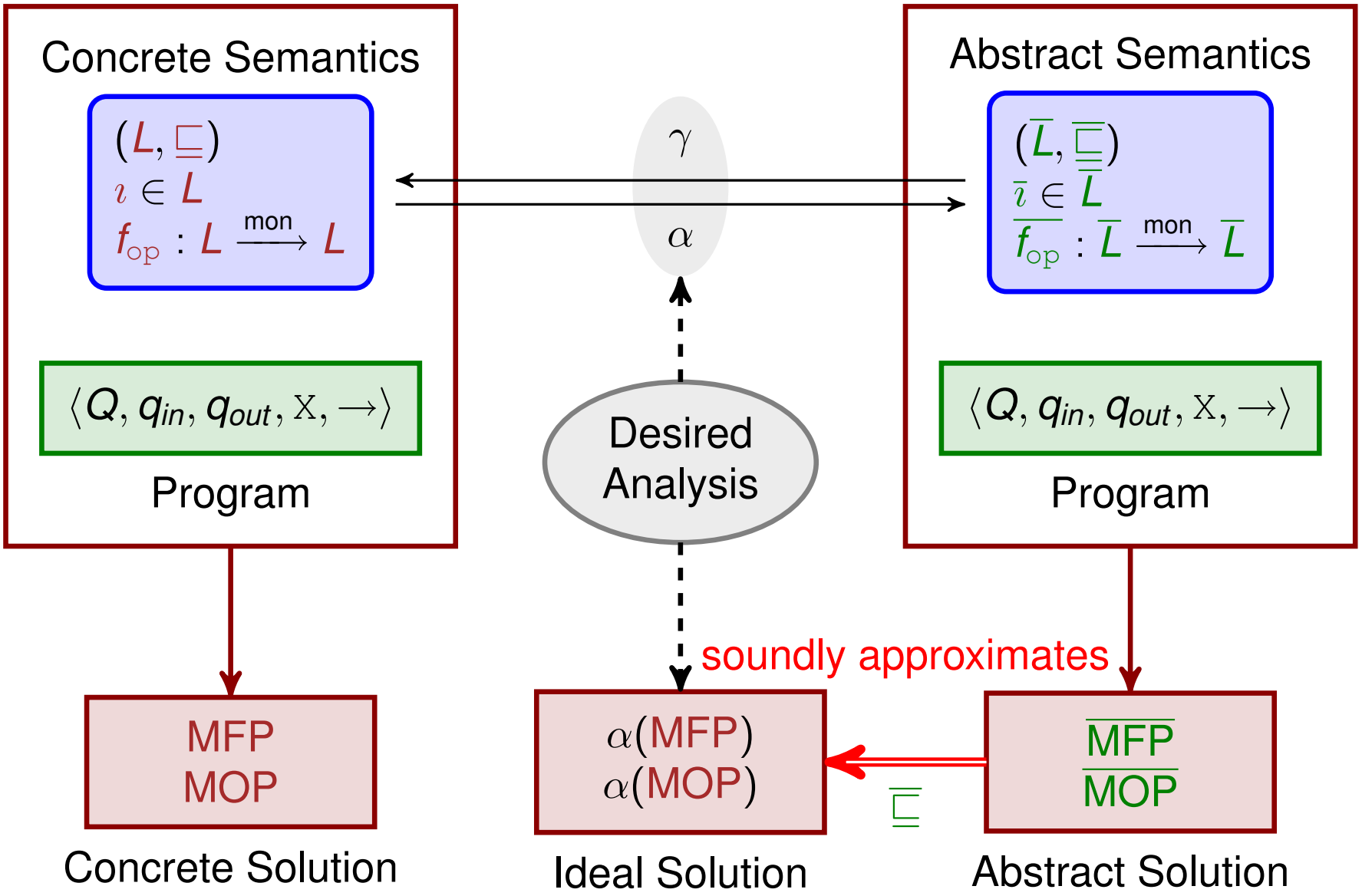
Systematic Design of Correct of Data Flow Analyses



Systematic Design of Correct of Data Flow Analyses



Systematic Design of Correct of Data Flow Analyses



Systematic Design of Correct of Data Flow Analyses

The MOP solution of the concrete semantics is the strongest property (i.e. the most precise fact) that is satisfied by all runs of the program.

The ideal solution to a given analysis is an approximation of the concrete MOP solution.

Natural Limitation

The class of possible analyses depends on the choice of "standard" concrete semantics.

Abstract data flow facts and transfer functions cannot be more precise than concrete ones.

Our operational semantics: $\langle Q \times (\mathtt{X} \to \mathbb{R}), \mathit{Init}, \mathit{Out}, \mathtt{Op}, \to \rangle$

Focus on numerical analyses

Systematic Design of Correct of Data Flow Analyses

The MOP solution of the concrete semantics is the strongest property (i.e. the most precise fact) that is satisfied by all runs of the program.

The ideal solution to a given analysis is an approximation of the concrete MOP solution.

Natural Limitation

The class of possible analyses depends on the choice of "standard" concrete semantics.

Abstract data flow facts and transfer functions cannot be more precise than concrete ones.

Our operational semantics: $\langle Q \times (\mathbb{X} \to \mathbb{R}), \mathit{Init}, \mathit{Out}, \mathtt{op}, \to \rangle$

Focus on numerical analyses

Standard Concrete Semantics

Control Flow Automaton: $\langle Q, q_{in}, q_{out}, \mathtt{X}, \rightarrow \rangle$

Recall: semantics $[\![\mathtt{op}]\!]$ of operations $\mathtt{op} \in \mathtt{Op}$

$$[\![\mathtt{op}]\!] \quad \subseteq \quad (\mathtt{X} \rightarrow \mathbb{R}) \times (\mathtt{X} \rightarrow \mathbb{R})$$

Monotone Framework

- Complete lattice $(L, \sqsubseteq)$ of data flow facts: $(\mathcal{P}(\mathtt{X} \rightarrow \mathbb{R}), \subseteq)$
- Set $\mathcal{F}$ of monotonic transfer functions:

$$\mathcal{F} \quad = \quad \{\lambda\, \phi\, .\, R[\phi] \mid R \subseteq (\mathtt{X} \rightarrow \mathbb{R}) \times (\mathtt{X} \rightarrow \mathbb{R})\}$$

Data Flow Instance $\overrightarrow{\mathcal{S}}$ for Forward Analysis

- Initial data flow value: $\top = \mathtt{X} \rightarrow \mathbb{R}$
- Transfer mapping: $f_{\mathtt{op}}(\phi) \quad = \quad [\![\mathtt{op}]\!][\phi]$

Standard Concrete Semantics

Control Flow Automaton: $\langle Q, q_{in}, q_{out}, \mathtt{X}, \rightarrow \rangle$

Recall: semantics $[\![\mathtt{op}]\!]$ of operations $\mathtt{op} \in \mathtt{Op}$

$$[\![\mathtt{op}]\!] \quad \subseteq \quad (\mathtt{X} \rightarrow \mathbb{R}) \times (\mathtt{X} \rightarrow \mathbb{R})$$

Monotone Framework

- Complete lattice $(L, \sqsubseteq)$ of data flow facts: $(\mathcal{P}(\mathtt{X} \rightarrow \mathbb{R}), \subseteq)$
- Set $\mathcal{F}$ of monotonic transfer functions:

$$\mathcal{F} \quad = \quad \{\lambda\, \phi\, .\, R[\phi] \mid R \subseteq (\mathtt{X} \rightarrow \mathbb{R}) \times (\mathtt{X} \rightarrow \mathbb{R})\}$$

Data Flow Instance $\overleftarrow{\mathbb{S}}$ for Backward Analysis

- Initial data flow value: $\top = \mathtt{X} \rightarrow \mathbb{R}$
- Transfer mapping: $f_{\mathtt{op}}(\phi) \quad = \quad [\![\mathtt{op}]\!]^{-1}[\phi]$

Standard Concrete Semantics

Control Flow Automaton: $\langle Q, q_{in}, q_{out}, \mathtt{X}, \rightarrow \rangle$

Recall: semantics $[\![\mathtt{op}]\!]$ of operations $\mathtt{op} \in \mathtt{Op}$

$$[\![\mathtt{op}]\!] \quad \subseteq \quad (\mathtt{X} \rightarrow \mathbb{R}) \times (\mathtt{X} \rightarrow \mathbb{R})$$

Monotone Framework: Distributive

- Complete lattice $(L, \sqsubseteq)$ of data flow facts: $(\mathcal{P}(\mathtt{X} \rightarrow \mathbb{R}), \subseteq)$
- Set $\mathcal{F}$ of monotonic transfer functions:

$$\mathcal{F} \quad = \quad \{\lambda\, \phi\, .\, R[\phi] \mid R \subseteq (\mathtt{X} \rightarrow \mathbb{R}) \times (\mathtt{X} \rightarrow \mathbb{R})\}$$

Data Flow Instance $\overleftarrow{\mathcal{S}}$ for Backward Analysis

- Initial data flow value: $\top = \mathtt{X} \rightarrow \mathbb{R}$
- Transfer mapping: $f_{\mathtt{op}}(\phi) \quad = \quad [\![\mathtt{op}]\!]^{-1}[\phi]$

Post^* and Pre^* as Data Flow Analysis Solutions

Consider a control flow automaton: $\langle Q, q_{in}, q_{out}, \mathtt{X}, \rightarrow \rangle$. Recall that:

$$\text{Post}^* = \bigcup_{q_{in} \xrightarrow{\mathtt{op}_0} \cdots \xrightarrow{\mathtt{op}_k} q} \{q\} \times (\llbracket \mathtt{op}_k \rrbracket \circ \cdots \circ \llbracket \mathtt{op}_0 \rrbracket)\,[(\mathtt{X} \rightarrow \mathbb{R})]$$

$$\text{Pre}^* = \bigcup_{q \xrightarrow{\mathtt{op}_0} \cdots \xrightarrow{\mathtt{op}_k} q_{out}} \{q\} \times \left((\llbracket \mathtt{op}_k \rrbracket \circ \cdots \circ \llbracket \mathtt{op}_0 \rrbracket)^{-1}\right)\,[(\mathtt{X} \rightarrow \mathbb{R})]$$

$$\text{Post}^* = \overrightarrow{\text{MOP}}\left(\overrightarrow{\mathcal{S}}\right) = \overrightarrow{\text{MFP}}\left(\overrightarrow{\mathcal{S}}\right)$$

$$\text{Pre}^* = \overleftarrow{\text{MOP}}\left(\overleftarrow{\mathcal{S}}\right) = \overleftarrow{\text{MFP}}\left(\overleftarrow{\mathcal{S}}\right)$$

Post^* and Pre^* as Data Flow Analysis Solutions

Consider a control flow automaton: $\langle Q, q_{in}, q_{out}, \mathtt{X}, \rightarrow \rangle$. Recall that:

$$\text{Post}^* = \bigcup_{q_{in} \xrightarrow{\mathtt{op}_0} \cdots \xrightarrow{\mathtt{op}_k} q} \{q\} \times (\llbracket \mathtt{op}_k \rrbracket \circ \cdots \circ \llbracket \mathtt{op}_0 \rrbracket)\,[(\mathtt{X} \rightarrow \mathbb{R})]$$

$$\text{Pre}^* = \bigcup_{q \xrightarrow{\mathtt{op}_0} \cdots \xrightarrow{\mathtt{op}_k} q_{out}} \{q\} \times \left((\llbracket \mathtt{op}_k \rrbracket \circ \cdots \circ \llbracket \mathtt{op}_0 \rrbracket)^{-1}\right)\,[(\mathtt{X} \rightarrow \mathbb{R})]$$

$$\text{Post}^* = \overrightarrow{\text{MOP}}\left(\overrightarrow{\mathcal{S}}\right) = \overrightarrow{\text{MFP}}\left(\overrightarrow{\mathcal{S}}\right)$$

$$\text{Pre}^* = \overleftarrow{\text{MOP}}\left(\overleftarrow{\mathcal{S}}\right) = \overleftarrow{\text{MFP}}\left(\overleftarrow{\mathcal{S}}\right)$$

Abstraction of the Concrete Semantics: Intuition

Concrete Semantics

$$\langle (\mathcal{P}(\mathbb{X} \to \mathbb{R}), \subseteq), \mathcal{F}, Q, q_{in}, q_{out}, \mathbb{X}, \rightarrow, f, \imath \rangle$$

Galois connection

$$(\mathcal{P}(\mathbb{X} \to \mathbb{R}), \subseteq) \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} (\overline{L}, \overline{\sqsubseteq})$$

$\overline{L}$ is a set of machine-representable "properties" of the variables.

Example

$$\overline{L} = \{\mathrm{x} \text{ is even}, \ \mathrm{y} \text{ is odd or negative}, \ \mathrm{x} \geq \mathrm{y} \Rightarrow \mathrm{x} = 2^i\}$$

$\gamma(\overline{\psi})$ is the meaning of an abstract "property" $\overline{\psi}$.

$\alpha(\phi)$ encodes a sound approximation of ϕ, the most precise one.

$\overline{\sqsubseteq}$ corresponds to entailment between "properties", and abstracts $\subseteq$.

Abstraction of the Concrete Semantics: Intuition

Concrete Semantics

$\langle (\mathcal{P}(\mathbb{X} \to \mathbb{R}), \subseteq), \mathcal{F}, Q, q_{in}, q_{out}, \mathbb{X}, \rightarrow, f, \imath \rangle$

Galois connection

$(\mathcal{P}(\mathbb{X} \to \mathbb{R}), \subseteq) \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} (\overline{L}, \overline{\sqsubseteq})$

$\overline{L}$ is a set of machine-representable "properties" of the variables.

Example

$$\overline{L} = \{\mathrm{x} \text{ is even}, \ \mathrm{y} \text{ is odd or negative}, \ \mathrm{x} \geq \mathrm{y} \Rightarrow \mathrm{x} = 2^i\}$$

$\gamma(\overline{\psi})$ is the meaning of an abstract "property" $\overline{\psi}$.

$\alpha(\phi)$ encodes a sound approximation of ϕ, the most precise one.

$\overline{\sqsubseteq}$ corresponds to entailment between "properties", and abstracts $\subseteq$.

Abstract Semantics Induced by a Galois Connection

Consider a data flow instance $\mathcal{A} = \langle (L, \sqsubseteq), \mathcal{F}, Q, q_{in}, q_{out}, \mathtt{X}, \rightarrow, f, \imath \rangle$ and a Galois connection $(L, \sqsubseteq) \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} (\overline{L}, \overline{\sqsubseteq})$.

Definition

The abstract data flow instance $\overline{\mathcal{A}}$ induced by $\mathcal{A}$ and $(L, \sqsubseteq) \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} (\overline{L}, \overline{\sqsubseteq})$ is $\overline{\mathcal{A}} = \langle (\overline{L}, \overline{\sqsubseteq}), \overline{\mathcal{F}}, Q, q_{in}, q_{out}, \mathtt{X}, \rightarrow, \overline{f}, \overline{\imath} \rangle$ where:

$$\begin{aligned} \overline{\mathcal{F}} &= \overline{L} \xrightarrow{\text{mon}} \overline{L} \\ \overline{f} &= \lambda\, \mathrm{op}\, .\, f^{\sharp}_{\mathrm{op}} \\ \overline{\imath} &= \alpha(\imath) \end{aligned}$$

Recall that $f^{\sharp}_{\mathrm{op}} = \alpha \circ f_{\mathrm{op}} \circ \gamma$ is the *best abstraction* of f_{op}.

Correctness of Induced Abstract Data Flow Analysis

Extension of Galois Connections to Functions

For any set Q and Galois connection $(L, \sqsubseteq) \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} (\overline{L}, \overline{\sqsubseteq})$, we have

$(Q \to L, \sqsubseteq) \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} (Q \to \overline{L}, \overline{\sqsubseteq})$ where: $\alpha(a) = \lambda q \,.\, \alpha(a(q))$

$\gamma(\overline{b}) = \lambda q \,.\, \gamma(\overline{b}(q))$

Theorem (Correctness of Induced Abstract Forward Analysis)

For any data flow instance $\mathcal{A}$ and Galois connection $(L, \sqsubseteq) \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} (\overline{L}, \overline{\sqsubseteq})$, the induced abstract data flow instance $\overline{\mathcal{A}}$ satisfies:

$$\overrightarrow{\mathrm{MFP}}(\mathcal{A}) \sqsubseteq \gamma\left(\overrightarrow{\mathrm{MFP}}(\overline{\mathcal{A}})\right) \qquad \alpha\left(\overrightarrow{\mathrm{MFP}}(\mathcal{A})\right) \sqsubseteq \overrightarrow{\mathrm{MFP}}(\overline{\mathcal{A}})$$

$$\overrightarrow{\mathrm{MOP}}(\mathcal{A}) \sqsubseteq \gamma\left(\overrightarrow{\mathrm{MOP}}(\overline{\mathcal{A}})\right) \qquad \alpha\left(\overrightarrow{\mathrm{MOP}}(\mathcal{A})\right) \sqsubseteq \overrightarrow{\mathrm{MOP}}(\overline{\mathcal{A}})$$

Correctness of Induced Abstract Data Flow Analysis

Extension of Galois Connections to Functions

For any set Q and Galois connection $(L, \sqsubseteq) \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} (\overline{L}, \overline{\sqsubseteq})$, we have

$(Q \to L, \sqsubseteq) \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} (Q \to \overline{L}, \overline{\sqsubseteq})$ where: $\alpha(a) = \lambda q \,.\, \alpha(a(q))$

$\gamma(\overline{b}) = \lambda q \,.\, \gamma(\overline{b}(q))$

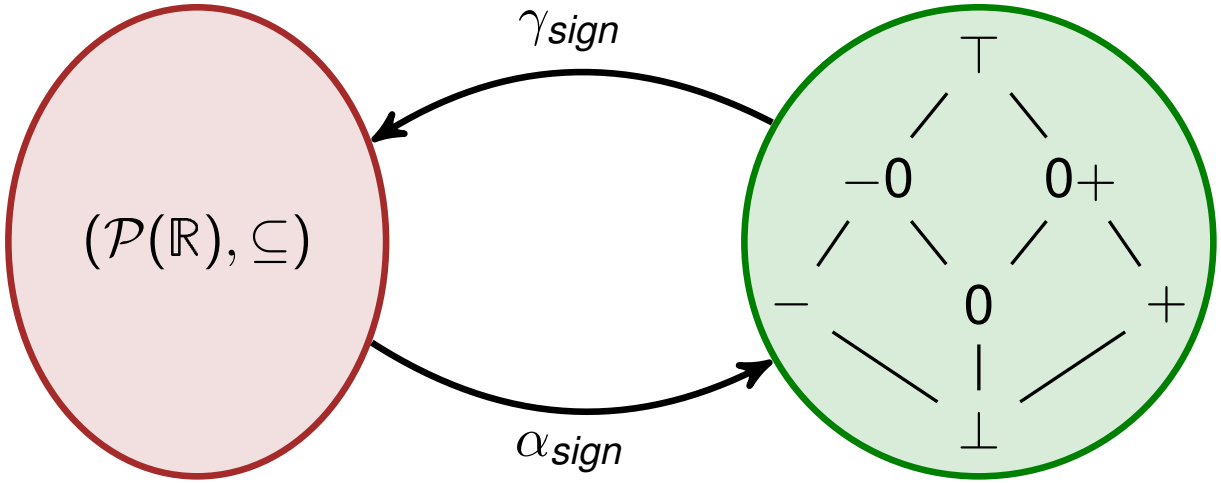
Theorem (Correctness of Induced Abstract Backward Analysis)

For any data flow instance $\mathcal{A}$ and Galois connection $(L, \sqsubseteq) \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} (\overline{L}, \overline{\sqsubseteq})$, the induced abstract data flow instance $\overline{\mathcal{A}}$ satisfies:

$$\overleftarrow{\mathrm{MFP}}(\mathcal{A}) \sqsubseteq \gamma\left(\overleftarrow{\mathrm{MFP}}(\overline{\mathcal{A}})\right) \qquad \alpha\left(\overleftarrow{\mathrm{MFP}}(\mathcal{A})\right) \sqsubseteq \overleftarrow{\mathrm{MFP}}(\overline{\mathcal{A}})$$

$$\overleftarrow{\mathrm{MOP}}(\mathcal{A}) \sqsubseteq \gamma\left(\overleftarrow{\mathrm{MOP}}(\overline{\mathcal{A}})\right) \qquad \alpha\left(\overleftarrow{\mathrm{MOP}}(\mathcal{A})\right) \sqsubseteq \overleftarrow{\mathrm{MOP}}(\overline{\mathcal{A}})$$

Back Again to Sign Analysis: Galois Connection



γ							
$\overline{y}$	$\bot$	$-$	0	$+$	-0	$0+$	$\top$
$\gamma(\overline{y})$	$\emptyset$	$\{r \mid r < 0\}$	$\{0\}$	$\{r \mid r > 0\}$	$\{r \mid r \leq 0\}$	$\{r \mid r \geq 0\}$	$\mathbb{R}$

Objective

Design a Galois Connection between:

- $(\mathcal{P}(\mathbb{X} \to \mathbb{R}), \subseteq)$, concrete data flow facts from standard semantics
- $(\mathbb{X} \to \mathit{Sign}, \overline{\sqsubseteq})$, abstract data flow facts

Intermediate Galois Connection: Projection

Convenient intermediate step for non-relational analyses

Objective of Projection

Design a Galois Connection between:

- $(\mathcal{P}(\mathbb{X} \to \mathbb{R}), \subseteq)$, concrete data flow facts from standard semantics
- $(\mathbb{X} \to \mathcal{P}(\mathbb{R}), \overline{\subseteq})$, projected data flow facts

where $\overline{\subseteq}$ is as expected: $\overline{\psi} \ \overline{\subseteq} \ \overline{\psi'}$ if $\overline{\psi}(x) \ \overline{\subseteq} \ \overline{\psi'}(x)$ for all $x \in \mathbb{X}$.

Fact

$$(\mathcal{P}(\mathbb{X} \to \mathbb{R}), \subseteq) \underset{\alpha_\pi}{\overset{\gamma_\pi}{\leftrightarrows}} (\mathbb{X} \to \mathcal{P}(\mathbb{R}), \overline{\subseteq})$$

where:

$$\alpha_\pi(\phi) = \lambda x . \{v(x) \mid v \in \phi\}$$

$$\gamma_\pi(\overline{\psi}) = \{v \in \mathbb{X} \to \mathbb{R} \mid v(x) \in \overline{\psi}(x) \text{ for all } x \in \mathbb{X}\}$$

Intermediate Galois Connection: Projection

Convenient intermediate step for non-relational analyses

Objective of Projection

Design a Galois Connection between:

- $(\mathcal{P}(\mathbb{X} \to \mathbb{R}), \subseteq)$, concrete data flow facts from standard semantics
- $(\mathbb{X} \to \mathcal{P}(\mathbb{R}), \overline{\subseteq})$, projected data flow facts

where $\overline{\subseteq}$ is as expected: $\overline{\psi} \ \overline{\subseteq} \ \overline{\psi'}$ if $\overline{\psi}(x) \ \overline{\subseteq} \ \overline{\psi'}(x)$ for all $x \in \mathbb{X}$.

Fact

$$(\mathcal{P}(\mathbb{X} \to \mathbb{R}), \subseteq) \underset{\alpha_\pi}{\overset{\gamma_\pi}{\leftrightarrows}} (\mathbb{X} \to \mathcal{P}(\mathbb{R}), \overline{\subseteq})$$

where:

$$\alpha_\pi(\phi) = \lambda x . \{v(x) \mid v \in \phi\}$$

$$\gamma_\pi(\overline{\psi}) = \{v \in \mathbb{X} \to \mathbb{R} \mid v(x) \in \overline{\psi}(x) \text{ for all } x \in \mathbb{X}\}$$

Intermediate Galois Connection: Projection

Convenient intermediate step for non-relational analyses

Objective of Projection

Design a Galois Connection between:

- $(\mathcal{P}(\mathtt{X} \to \mathbb{R}), \subseteq)$, concrete data flow facts from standard semantics
- $(\mathtt{X} \to \mathcal{P}(\mathbb{R}), \overline{\subseteq})$, projected data flow facts

where $\overline{\subseteq}$ is as expected: $\overline{\psi} \ \overline{\subseteq} \ \overline{\psi'}$ if $\overline{\psi}(x) \ \overline{\subseteq} \ \overline{\psi'}(x)$ for all $x \in \mathtt{X}$.

Fact

$$(\mathcal{P}(\mathtt{X} \to \mathbb{R}), \subseteq) \underset{\alpha_\pi}{\overset{\gamma_\pi}{\leftrightarrows}} (\mathtt{X} \to \mathcal{P}(\mathbb{R}), \overline{\subseteq})$$

where:

$$\alpha_\pi(\phi) = \lambda x \,.\, \{v(x) \mid v \in \phi\}$$

$$\gamma_\pi(\overline{\psi}) = \left\{ v \in \mathtt{X} \to \mathbb{R} \;\middle|\; v(x) \in \overline{\psi}(x) \text{ for all } x \in \mathtt{X} \right\}$$

Back Again to Sign Analysis: Galois Connection

Projection

$$(\mathcal{P}(\mathbb{X} \to \mathbb{R}), \subseteq) \underset{\alpha_\pi}{\overset{\gamma_\pi}{\leftrightarrows}} (\mathbb{X} \to \mathcal{P}(\mathbb{R}), \subseteq)$$

Sign

$$(\mathcal{P}(\mathbb{R}), \subseteq) \underset{\alpha_{sign}}{\overset{\gamma_{sign}}{\leftrightarrows}} (\mathit{Sign}, \overline{\sqsubseteq})$$

Extension of Sign to Functions

$$(\mathbb{X} \to \mathcal{P}(\mathbb{R}), \subseteq) \underset{\alpha_{sign}}{\overset{\gamma_{sign}}{\leftrightarrows}} (\mathbb{X} \to \mathit{Sign}, \overline{\sqsubseteq})$$

$$(\mathcal{P}(\mathbb{X} \to \mathbb{R}), \subseteq) \underset{\alpha_{sign} \circ \alpha_\pi}{\overset{\gamma_\pi \circ \gamma_{sign}}{\leftrightarrows}} (\mathbb{X} \to \mathit{Sign}, \overline{\sqsubseteq})$$

The composition of Galois connections is a Galois connection.

Back Again to Sign Analysis: Induced Instance

$$(\mathcal{P}(\mathtt{X} \to \mathbb{R}), \subseteq) \underset{\alpha_\pi}{\overset{\gamma_\pi}{\leftrightarrows}} (\mathtt{X} \to \mathcal{P}(\mathbb{R}), \subseteq) \underset{\alpha_{\mathit{sign}}}{\overset{\gamma_{\mathit{sign}}}{\leftrightarrows}} (\mathtt{X} \to \mathit{Sign}, \overline{\sqsubseteq})$$

$$\begin{array}{rcl} \overline{\mathcal{F}} & = & (\mathtt{X} \to \mathit{Sign}) \xrightarrow{\text{mon}} (\mathtt{X} \to \mathit{Sign}) \\ \overline{f} & = & \lambda\, \mathtt{op}\,.\, f^\sharp_{\mathtt{op}} \\ \bar{\imath} & = & \alpha_{\mathit{sign}} \circ \alpha_\pi(\top) = \lambda\, x\,.\, \overline{\top} \end{array}$$

But this data flow instance looks similar to what we did previously (less painfully) without Galois connections...

What do we get?

The most precise transfer mapping that soundly approximates the standard semantics

Back Again to Sign Analysis: Induced Instance

$$(\mathcal{P}(\mathtt{X} \to \mathbb{R}), \subseteq) \underset{\alpha_\pi}{\overset{\gamma_\pi}{\leftrightarrows}} (\mathtt{X} \to \mathcal{P}(\mathbb{R}), \subseteq) \underset{\alpha_{\mathit{sign}}}{\overset{\gamma_{\mathit{sign}}}{\leftrightarrows}} (\mathtt{X} \to \mathit{Sign}, \overline{\sqsubseteq})$$

$$\begin{array}{rcl} \overline{\mathcal{F}} & = & (\mathtt{X} \to \mathit{Sign}) \xrightarrow{\text{mon}} (\mathtt{X} \to \mathit{Sign}) \\ \overline{f} & = & \lambda\, \mathtt{op}\,.\, f^\sharp_{\mathtt{op}} \\ \bar{\imath} & = & \alpha_{\mathit{sign}} \circ \alpha_\pi(\top) = \lambda\, x\,.\, \overline{\top} \end{array}$$

But this data flow instance looks similar to what we did previously (less painfully) without Galois connections...

What do we get?

The most precise transfer mapping that soundly approximates the standard semantics

Forward Sign Analysis: Transfer Mapping

$$(\mathcal{P}(\mathrm{X} \to \mathbb{R}), \subseteq) \underset{\alpha_\pi}{\overset{\gamma_\pi}{\leftrightarrows}} (\mathrm{X} \to \mathcal{P}(\mathbb{R}), \subseteq) \underset{\alpha_{sign}}{\overset{\gamma_{sign}}{\leftrightarrows}} (\mathrm{X} \to \mathit{Sign}, \overline{\sqsubseteq})$$

$$f^\sharp_{\mathrm{op}}(\overline{\psi}) \quad = \quad \alpha_{sign} \circ \alpha_\pi \left([\![\mathrm{op}]\!] \left[\gamma_\pi \circ \gamma_{sign}(\overline{\psi}) \right] \right)$$

Extensions of $[\![e]\!]$ and $[\![g]\!]$ to subsets of $\mathbb{R}$

$[\![e]\!]$

$$\begin{array}{rcl} \mathcal{P}(\mathrm{X} \to \mathbb{R}) & \to & \mathcal{P}(\mathbb{R}) \\ \phi & \mapsto & \{ [\![e]\!]_v \mid v \in \phi \} \end{array}$$

$[\![g]\!]$

$$\begin{array}{rcl} \mathcal{P}(\mathrm{X} \to \mathbb{R}) & \to & \mathcal{P}(\mathrm{X} \to \mathbb{R}) \\ \phi & \mapsto & \{ v \in \phi \mid v \models g \} \end{array}$$

$$f^\sharp_{x:=e}(\overline{v}) \quad = \quad \lambda\, y \,.\, \begin{cases} \overline{v}(y) & \text{if } y \neq x \\ \alpha_{sign} \circ [\![e]\!] \circ \gamma(\overline{v}) & \text{if } y = x \end{cases}$$

$$f^\sharp_g \quad = \quad \alpha_{sign} \circ \alpha_\pi \circ [\![g]\!] \circ \gamma_\pi \circ \gamma_{sign}$$

Forward Sign Analysis: Transfer Mapping

$$(\mathcal{P}(\mathrm{X} \to \mathbb{R}), \subseteq) \underset{\alpha_\pi}{\overset{\gamma_\pi}{\leftrightarrows}} (\mathrm{X} \to \mathcal{P}(\mathbb{R}), \subseteq) \underset{\alpha_{sign}}{\overset{\gamma_{sign}}{\leftrightarrows}} (\mathrm{X} \to \mathit{Sign}, \overline{\sqsubseteq})$$

$$f^\sharp_{\mathrm{op}}(\overline{\psi}) \quad = \quad \alpha_{sign} \circ \alpha_\pi \left([\![\mathrm{op}]\!] \left[\gamma_\pi \circ \gamma_{sign}(\overline{\psi}) \right] \right)$$

Extensions of $[\![e]\!]$ and $[\![g]\!]$ to subsets of $\mathbb{R}$

$[\![e]\!]$		
$\mathcal{P}(\mathrm{X} \to \mathbb{R})$	$\to$	$\mathcal{P}(\mathbb{R})$
ϕ	$\mapsto$	$\{[\![e]\!]_v \mid v \in \phi\}$

$[\![g]\!]$		
$\mathcal{P}(\mathrm{X} \to \mathbb{R})$	$\to$	$\mathcal{P}(\mathrm{X} \to \mathbb{R})$
ϕ	$\mapsto$	$\{v \in \phi \mid v \models g\}$

$$f^\sharp_{x:=e}(\overline{v}) \quad = \quad \lambda\, y \,.\, \begin{cases} \overline{v}(y) & \text{if } y \neq x \\ \alpha_{sign} \circ [\![e]\!] \circ \gamma(\overline{v}) & \text{if } y = x \end{cases}$$

$$f^\sharp_g \quad = \quad \alpha_{sign} \circ \alpha_\pi \circ [\![g]\!] \circ \gamma_\pi \circ \gamma_{sign}$$

Forward Sign Analysis: Transfer Mapping

$$(\mathcal{P}(\mathrm{X} \to \mathbb{R}), \subseteq) \underset{\alpha_\pi}{\overset{\gamma_\pi}{\leftrightarrows}} (\mathrm{X} \to \mathcal{P}(\mathbb{R}), \subseteq) \underset{\alpha_{sign}}{\overset{\gamma_{sign}}{\leftrightarrows}} (\mathrm{X} \to \mathit{Sign}, \overline{\sqsubseteq})$$

$$f^\sharp_{\mathrm{op}}(\overline{\psi}) \quad = \quad \alpha_{sign} \circ \alpha_\pi \left([\![\mathrm{op}]\!] \left[\gamma_\pi \circ \gamma_{sign}(\overline{\psi}) \right] \right)$$

Extensions of $[\![e]\!]$ and $[\![g]\!]$ to subsets of $\mathbb{R}$

$[\![e]\!]$		
$\mathcal{P}(\mathrm{X} \to \mathbb{R})$	$\to$	$\mathcal{P}(\mathbb{R})$
ϕ	$\mapsto$	$\{[\![e]\!]_v \mid v \in \phi\}$

$[\![g]\!]$		
$\mathcal{P}(\mathrm{X} \to \mathbb{R})$	$\to$	$\mathcal{P}(\mathrm{X} \to \mathbb{R})$
ϕ	$\mapsto$	$\{v \in \phi \mid v \models g\}$

$$f^\sharp_{x:=e}(\overline{v}) \quad = \quad \lambda\, y \,.\, \begin{cases} \overline{v}(y) & \text{if } y \neq x \\ \alpha_{sign} \circ [\![e]\!] \circ \gamma(\overline{v}) & \text{if } y = x \end{cases}$$

$$f^\sharp_g \quad = \quad \alpha_{sign} \circ \alpha_\pi \circ [\![g]\!] \circ \gamma_\pi \circ \gamma_{sign}$$

Forward Sign Analysis: Transfer Mapping

$$(\mathcal{P}(\mathtt{X} \to \mathbb{R}), \subseteq) \underset{\alpha_\pi}{\overset{\gamma_\pi}{\leftrightarrows}} (\mathtt{X} \to \mathcal{P}(\mathbb{R}), \subseteq) \underset{\alpha_{sign}}{\overset{\gamma_{sign}}{\leftrightarrows}} (\mathtt{X} \to \mathit{Sign}, \overline{\sqsubseteq})$$

$$f^\sharp_{x:=e}(\overline{v}) \quad = \quad \lambda\, y\,.\, \begin{cases} \overline{v}(y) & \text{if } y \neq x \\ \alpha_{sign} \circ [\![e]\!] \circ \gamma(\overline{v}) & \text{if } y = x \end{cases}$$

Not easy to compute!

Example

Consider a (non-constant) multivariate polynomial expression e and the operation $\mathtt{op} = \mathtt{x} := e * e$.

$$f^\sharp_{\mathtt{op}}(\overline{\top}) \quad = \quad \lambda\, y\,.\, \begin{cases} \overline{\top} & \text{if } y \neq \mathtt{x} \\ 0+ & \text{if } y = \mathtt{x} \text{ and } e \text{ has a root} \\ + & \text{if } y = \mathtt{x} \text{ and } e \text{ has no root} \end{cases}$$

Forward Sign Analysis: Transfer Mapping

$$(\mathcal{P}(\mathtt{X} \to \mathbb{R}), \subseteq) \underset{\alpha_\pi}{\overset{\gamma_\pi}{\leftrightarrows}} (\mathtt{X} \to \mathcal{P}(\mathbb{R}), \subseteq) \underset{\alpha_{sign}}{\overset{\gamma_{sign}}{\leftrightarrows}} (\mathtt{X} \to \mathit{Sign}, \overline{\sqsubseteq})$$

$$f^\sharp_{x:=e}(\overline{v}) \quad = \quad \lambda\, y\,.\, \begin{cases} \overline{v}(y) & \text{if } y \neq x \\ \alpha_{sign} \circ [\![e]\!] \circ \gamma(\overline{v}) & \text{if } y = x \end{cases}$$

Not easy to compute! Even for the simple *Sign* lattice!

Example

Consider a (non-constant) multivariate polynomial expression e and the operation $\mathrm{op} = \mathtt{x} := e * e$.

$$f^\sharp_{\mathrm{op}}(\overline{\top}) \quad = \quad \lambda\, y\,.\, \begin{cases} \overline{\top} & \text{if } y \neq \mathtt{x} \\ 0+ & \text{if } y = \mathtt{x} \text{ and } e \text{ has a root} \\ + & \text{if } y = \mathtt{x} \text{ and } e \text{ has no root} \end{cases}$$

Forward Sign Analysis: Transfer Mapping

$$(\mathcal{P}(\mathtt{X} \to \mathbb{R}), \subseteq) \underset{\alpha_\pi}{\overset{\gamma_\pi}{\leftrightarrows}} (\mathtt{X} \to \mathcal{P}(\mathbb{R}), \subseteq) \underset{\alpha_{sign}}{\overset{\gamma_{sign}}{\leftrightarrows}} (\mathtt{X} \to \mathit{Sign}, \overline{\sqsubseteq})$$

$$f^\sharp_{x:=e}(\overline{v}) \quad = \quad \lambda\, y\,.\, \begin{cases} \overline{v}(y) & \text{if } y \neq x \\ \alpha_{sign} \circ [\![e]\!] \circ \gamma(\overline{v}) & \text{if } y = x \end{cases}$$

Not easy to compute! Even for the simple *Sign* lattice!

Example

Consider a (non-constant) multivariate polynomial expression e and the operation $\mathrm{op} = \mathtt{x} := e * e$.

$$f^\sharp_{\mathrm{op}}(\overline{\top}) \quad = \quad \lambda\, y\,.\, \begin{cases} \overline{\top} & \text{if } y \neq \mathtt{x} \\ 0+ & \text{if } y = \mathtt{x} \text{ and } e \text{ has a root} \\ + & \text{if } y = \mathtt{x} \text{ and } e \text{ has no root} \end{cases}$$

Forward Sign Analysis: Transfer Mapping

What can be done?

Forward Sign Analysis: Transfer Mapping

What can be done?

Approximate!

But soundly ☺

Forward Sign Analysis: Transfer Mapping

What can be done?

Approximate!

But soundly ☺

Approximate Transfer Mapping

Replace each $f^{\sharp}_{\mathtt{op}}$ with an approximate transfer function $\overline{h_{\mathtt{op}}}$ that

- exploits the structure of operations to obtain
- better performance at the expense of precision.

Last Bit of Lattice Theory

Theorem

For any two monotonic functions f, g on a complete lattice $(L, \sqsubseteq)$,

$$\textit{if} \quad f(x) \sqsubseteq g(x) \quad \textit{for all } x \in L \quad \textit{then} \quad \mathrm{lfp}(f) \sqsubseteq \mathrm{lfp}(g)$$

Proof.

$$\{x \in L \mid f(x) \sqsubseteq x\} \supseteq \{x \in L \mid g(x) \sqsubseteq x\}$$

Hence

$$\mathrm{lfp}(f) = \bigsqcap \{x \in L \mid f(x) \sqsubseteq x\} \sqsubseteq \bigsqcap \{x \in L \mid g(x) \sqsubseteq x\} = \mathrm{lfp}(g)$$

□

Last Bit of Lattice Theory

Theorem

For any two monotonic functions f, g on a complete lattice $(L, \sqsubseteq)$,

$$\textit{if} \quad f(x) \sqsubseteq g(x) \quad \textit{for all } x \in L \quad \textit{then} \quad \mathrm{lfp}(f) \sqsubseteq \mathrm{lfp}(g)$$

Proof.

$$\{x \in L \mid f(x) \sqsubseteq x\} \supseteq \{x \in L \mid g(x) \sqsubseteq x\}$$

Hence

$$\mathrm{lfp}(f) = \bigsqcap \{x \in L \mid f(x) \sqsubseteq x\} \sqsubseteq \bigsqcap \{x \in L \mid g(x) \sqsubseteq x\} = \mathrm{lfp}(g)$$

Correctness of Approximate Transfer Mapping

Consider a data flow instance $\mathcal{A}$ with a set $\mathcal{F}$ of transfer functions and a transfer mapping $f : \mathtt{Op} \to \mathcal{F}$.

For any monotonic function $h : \mathtt{Op} \to \mathcal{F}$ verifying

$$f_{\mathtt{op}}(x) \sqsubseteq h_{\mathtt{op}}(x) \qquad \text{(for all } \mathtt{op} \in \mathtt{Op}, x \in L\text{)}$$

the data flow instance $\mathcal{B}$ obtained from $\mathcal{A}$ by replacing f with h satisfies:

$$\overrightarrow{\mathsf{MFP}}(\mathcal{A}) \sqsubseteq \overrightarrow{\mathsf{MFP}}(\mathcal{B}) \qquad \overrightarrow{\mathsf{MOP}}(\mathcal{A}) \sqsubseteq \overrightarrow{\mathsf{MOP}}(\mathcal{B})$$

Application to Induced Abstract Data Flow Instances

Replace $f^\sharp_{\mathtt{op}}$ with a simpler monotonic $\overline{h_{\mathtt{op}}}$ verifying

$$f^\sharp_{\mathtt{op}}(\overline{x}) \sqsupseteq \overline{h_{\mathtt{op}}}(\overline{x}) \qquad \text{(for all } \mathtt{op} \in \mathtt{Op}, \overline{x} \in L\text{)}$$

Correctness of Approximate Transfer Mapping

Consider a data flow instance $\mathcal{A}$ with a set $\mathcal{F}$ of transfer functions and a transfer mapping $f : \mathtt{Op} \to \mathcal{F}$.

For any monotonic function $h : \mathtt{Op} \to \mathcal{F}$ verifying

$$f_{\mathtt{op}}(x) \sqsubseteq h_{\mathtt{op}}(x) \qquad \text{(for all } \mathtt{op} \in \mathtt{Op}, x \in L\text{)}$$

the data flow instance $\mathcal{B}$ obtained from $\mathcal{A}$ by replacing f with h satisfies:

$$\overleftarrow{\mathrm{MFP}}(\mathcal{A}) \sqsubseteq \overleftarrow{\mathrm{MFP}}(\mathcal{B}) \qquad \overleftarrow{\mathrm{MOP}}(\mathcal{A}) \sqsubseteq \overleftarrow{\mathrm{MOP}}(\mathcal{B})$$

Application to Induced Abstract Data Flow Instances

Replace $f^\sharp_{\mathtt{op}}$ with a simpler monotonic $\overline{h_{\mathtt{op}}}$ verifying

$$f^\sharp_{\mathtt{op}}(\overline{x}) \sqsupseteq \overline{h_{\mathtt{op}}}(\overline{x}) \qquad \text{(for all } \mathtt{op} \in \mathtt{Op}, \overline{x} \in L\text{)}$$

Correctness of Approximate Transfer Mapping

Consider a data flow instance $\mathcal{A}$ with a set $\mathcal{F}$ of transfer functions and a transfer mapping $f : \mathtt{Op} \to \mathcal{F}$.

For any monotonic function $h : \mathtt{Op} \to \mathcal{F}$ verifying

$$f_{\mathtt{op}}(x) \sqsubseteq h_{\mathtt{op}}(x) \qquad \text{(for all } \mathtt{op} \in \mathtt{Op}, x \in L\text{)}$$

the data flow instance $\mathcal{B}$ obtained from $\mathcal{A}$ by replacing f with h satisfies:

$$\overleftarrow{\mathrm{MFP}}(\mathcal{A}) \sqsubseteq \overleftarrow{\mathrm{MFP}}(\mathcal{B}) \qquad \overleftarrow{\mathrm{MOP}}(\mathcal{A}) \sqsubseteq \overleftarrow{\mathrm{MOP}}(\mathcal{B})$$

Application to Induced Abstract Data Flow Instances

Replace $f^\sharp_{\mathtt{op}}$ with a simpler monotonic $\overline{h_{\mathtt{op}}}$ verifying

$$f^\sharp_{\mathtt{op}}(\overline{x}) \sqsupseteq \overline{h_{\mathtt{op}}}(\overline{x}) \qquad \text{(for all } \mathtt{op} \in \mathtt{Op}, \overline{x} \in L\text{)}$$

Design of Approximate Transfer Mapping

Given a Galois connection $(\mathcal{P}(\mathbb{X} \to \mathbb{R}), \subseteq) \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} (\overline{L}, \overline{\sqsubseteq})$ the resulting abstract data flow instance is obtained systematically.

But in practice, $f^\sharp$ is rarely used: an approximate transfer mapping is required.

Tradeoff between computational cost and precision: many possibilities!

General principle: exploit the structure operations

1. define an abstract conservative semantics for arithmetic operators and comparators, ideally the most precise one
2. derive inductively an abstract semantics for operations, as usual

Design of Approximate Transfer Mapping

Given a Galois connection $(\mathcal{P}(\mathbb{X} \to \mathbb{R}), \subseteq) \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} (\overline{L}, \overline{\sqsubseteq})$ the resulting abstract data flow instance is obtained systematically.

But in practice, $f^\sharp$ is rarely used: an approximate transfer mapping is required.

Tradeoff between computational cost and precision: many possibilities!

General principle: exploit the structure operations

1. define an abstract conservative semantics for arithmetic operators and comparators, ideally the most precise one
2. derive inductively an abstract semantics for operations, as usual

Sign Analysis: Abstract Arithmetic Operators

$$(\mathcal{P}(\mathbb{R}), \subseteq) \underset{\alpha_{sign}}{\overset{\gamma_{sign}}{\leftrightarrows}} (Sign, \overline{\sqsubseteq})$$

Extension of Arithmetic Operators to Subsets of $\mathbb{R}$

For each function $* \in \{+, -, \times, \ldots\}$ from $\mathbb{R} \times \mathbb{R}$ to $\mathbb{R}$, define the function $* : (\mathcal{P}(\mathbb{R}) \times \mathcal{P}(\mathbb{R})) \to \mathcal{P}(\mathbb{R})$ by:

$$U * V \quad = \quad \{u * v \mid u \in U, v \in V\}$$

Abstract Arithmetic Operators

Define the best abstraction $*^\sharp : (Sign \times Sign) \to Sign$ of each function $* \in \{+, -, \times, \ldots\}$ by:

$$\overline{x} \; *^\sharp \; \overline{y} \quad = \quad \alpha_{sign}\left(\gamma_{sign}(\overline{x}) * \gamma_{sign}(\overline{y})\right)$$

Abstract Arithmetic Operators: Table for $+^\sharp$

$$\overline{x} +^\sharp \overline{y} = \alpha_{sign}\left(\gamma_{sign}(\overline{x}) + \gamma_{sign}(\overline{y})\right)$$

$+^\sharp$	$\bot$	$-$	0	$+$	-0	$0+$	$\top$
$\bot$	$\bot$	$\bot$	$\bot$	$\bot$	$\bot$	$\bot$	$\bot$
$-$	$\bot$	$-$	$-$	$\top$	$-$	$\top$	$\top$
0	$\bot$	$-$	0	$+$	-0	$0+$	$\top$
$+$	$\bot$	$\top$	$+$	$+$	$\top$	$+$	$\top$
-0	$\bot$	$-$	-0	$\top$	-0	$\top$	$\top$
$0+$	$\bot$	$\top$	$0+$	$+$	$\top$	$0+$	$\top$
$\top$	$\bot$	$\top$	$\top$	$\top$	$\top$	$\top$	$\top$

After mechanical inspection of all cases, we derive the above table.

We can derive similar tables for $-^\sharp$ and $\times^\sharp$.

Sign Analysis: Abstract Semantics of Expressions

$$(\mathcal{P}(\mathbb{X} \to \mathbb{R}), \subseteq) \underset{\alpha_\pi}{\overset{\gamma_\pi}{\leftrightarrows}} (\mathbb{X} \to \mathcal{P}(\mathbb{R}), \subseteq) \underset{\alpha_{sign}}{\overset{\gamma_{sign}}{\leftrightarrows}} (\mathbb{X} \to \mathit{Sign}, \overline{\sqsubseteq})$$

For any abstract valuation $\overline{v} : \mathbb{X} \to \mathit{Sign}$, define $\overline{[\![e]\!]}_{\overline{v}}$ inductively:

$$\begin{array}{rcll}
\overline{[\![c]\!]}_{\overline{v}} & = & \alpha_{sign}(\{c\}) & [c \in \mathbb{Q}] \\
\overline{[\![x]\!]}_{\overline{v}} & = & \overline{v}(x) & [x \in \mathbb{X}] \\
\overline{[\![e_1 + e_2]\!]}_{\overline{v}} & = & \overline{[\![e_1]\!]}_{\overline{v}} +^\sharp \overline{[\![e_2]\!]}_{\overline{v}} & \\
\overline{[\![e_1 - e_2]\!]}_{\overline{v}} & = & \overline{[\![e_1]\!]}_{\overline{v}} -^\sharp \overline{[\![e_2]\!]}_{\overline{v}} & \\
\overline{[\![e_1 * e_2]\!]}_{\overline{v}} & = & \overline{[\![e_1]\!]}_{\overline{v}} \times^\sharp \overline{[\![e_2]\!]}_{\overline{v}} &
\end{array}$$

Fact (Conservative Approximation)

$$\overline{[\![e]\!]}(\overline{v}) \quad \overline{\sqsupseteq} \quad \alpha_{sign} \circ [\![e]\!] \circ \gamma(\overline{v})$$

Sign Analysis: Abstract Semantics of Expressions

$$(\mathcal{P}(\mathbb{X} \to \mathbb{R}), \subseteq) \underset{\alpha_\pi}{\overset{\gamma_\pi}{\leftrightarrows}} (\mathbb{X} \to \mathcal{P}(\mathbb{R}), \subseteq) \underset{\alpha_{sign}}{\overset{\gamma_{sign}}{\leftrightarrows}} (\mathbb{X} \to \mathit{Sign}, \overline{\sqsubseteq})$$

For any abstract valuation $\overline{v} : \mathbb{X} \to \mathit{Sign}$, define $\overline{[\![e]\!]}_{\overline{v}}$ inductively:

$$\begin{array}{rcll}
\overline{[\![c]\!]}_{\overline{v}} & = & \alpha_{sign}(\{c\}) & [c \in \mathbb{Q}] \\
\overline{[\![x]\!]}_{\overline{v}} & = & \overline{v}(x) & [x \in \mathbb{X}] \\
\overline{[\![e_1 + e_2]\!]}_{\overline{v}} & = & \overline{[\![e_1]\!]}_{\overline{v}} +^\sharp \overline{[\![e_2]\!]}_{\overline{v}} & \\
\overline{[\![e_1 - e_2]\!]}_{\overline{v}} & = & \overline{[\![e_1]\!]}_{\overline{v}} -^\sharp \overline{[\![e_2]\!]}_{\overline{v}} & \\
\overline{[\![e_1 * e_2]\!]}_{\overline{v}} & = & \overline{[\![e_1]\!]}_{\overline{v}} \times^\sharp \overline{[\![e_2]\!]}_{\overline{v}} &
\end{array}$$

Fact (Conservative Approximation)

$$\overline{[\![e]\!]}(\overline{v}) \quad \overline{\sqsupseteq} \quad \alpha_{sign} \circ [\![e]\!] \circ \gamma(\overline{v})$$

Sign Analysis: Abstract Arithmetic Comparators

$$(\mathcal{P}(\mathbb{R}), \subseteq) \underset{\alpha_{sign}}{\overset{\gamma_{sign}}{\leftrightarrows}} (\mathit{Sign}, \overline{\sqsubseteq})$$

Extension of Arithmetic Comparators to Subsets of $\mathbb{R}$

For each binary relation $\bowtie \in \{<, \leq, =, \neq, >, \geq, \ldots\}$ on $\mathbb{R}$, define the binary relation $\bowtie$ on $\mathcal{P}(\mathbb{R})$ by:

$$U \bowtie V \quad \text{if} \quad u \bowtie v \text{ for some } u \in U \text{ and } v \in V$$

Abstract Arithmetic Comparators

Define the best abstraction $\bowtie^\sharp \subseteq \mathit{Sign} \times \mathit{Sign}$ of each binary relation $\bowtie \in \{<, \leq, =, \neq, >, \geq, \ldots\}$ by:

$$\overline{x} \bowtie^\sharp \overline{y} \quad \text{if} \quad \gamma_{sign}(\overline{x}) \bowtie \gamma_{sign}(\overline{y})$$

Abstract Arithmetic Comparators: Table for $<^\sharp$

$$\overline{x} <^\sharp \overline{y} \quad \text{if} \quad \gamma_{sign}(\overline{x}) < \gamma_{sign}(\overline{y})$$

$<^\sharp$	$\bot$	$-$	0	$+$	-0	$0+$	$\top$
$\bot$							
$-$		•	•	•	•	•	•
0				•		•	•
$+$				•		•	•
-0		•	•	•	•	•	•
$0+$				•		•	•
$\top$		•	•	•	•	•	•

After mechanical inspection of all cases, we derive the above table.

We can derive similar tables for $\leq^\sharp$, $=^\sharp$, $\neq^\sharp$, $>^\sharp$, and $\geq^\sharp$.

Sign Analysis: Abstract Semantics of Guards

$$(\mathcal{P}(\mathtt{X} \to \mathbb{R}), \subseteq) \underset{\alpha_\pi}{\overset{\gamma_\pi}{\leftrightarrows}} (\mathtt{X} \to \mathcal{P}(\mathbb{R}), \subseteq) \underset{\alpha_{sign}}{\overset{\gamma_{sign}}{\leftrightarrows}} (\mathtt{X} \to \mathit{Sign}, \overline{\sqsubseteq})$$

For any abstract valuation $\overline{v} : \mathtt{X} \to \mathit{Sign}$, define $\overline{v} \models g$ inductively:

$$\begin{array}{lcll}
\overline{v} & \models & e_1 < e_2 & \text{if} \quad \overline{[\![e_1]\!]}_{\overline{v}} <^\sharp \overline{[\![e_2]\!]}_{\overline{v}} \\
\overline{v} & \models & e_1 \leq e_2 & \text{if} \quad \overline{[\![e_1]\!]}_{\overline{v}} \leq^\sharp \overline{[\![e_2]\!]}_{\overline{v}} \\
\overline{v} & \models & e_1 = e_2 & \text{if} \quad \overline{[\![e_1]\!]}_{\overline{v}} =^\sharp \overline{[\![e_2]\!]}_{\overline{v}} \\
\overline{v} & \models & e_1 \neq e_2 & \text{if} \quad \overline{[\![e_1]\!]}_{\overline{v}} \neq^\sharp \overline{[\![e_2]\!]}_{\overline{v}} \\
\overline{v} & \models & e_1 \geq e_2 & \text{if} \quad \overline{[\![e_1]\!]}_{\overline{v}} \geq^\sharp \overline{[\![e_2]\!]}_{\overline{v}} \\
\overline{v} & \models & e_1 > e_2 & \text{if} \quad \overline{[\![e_1]\!]}_{\overline{v}} >^\sharp \overline{[\![e_2]\!]}_{\overline{v}}
\end{array}$$

Fact (Conservative Approximation)

$$\text{if } v \models g \text{ for some } v \in \gamma(\overline{v}) \quad \text{then} \quad \overline{v} \models g$$

Sign Analysis: Abstract Semantics of Guards

$$(\mathcal{P}(\mathtt{X} \to \mathbb{R}), \subseteq) \underset{\alpha_\pi}{\overset{\gamma_\pi}{\leftrightarrows}} (\mathtt{X} \to \mathcal{P}(\mathbb{R}), \subseteq) \underset{\alpha_{sign}}{\overset{\gamma_{sign}}{\leftrightarrows}} (\mathtt{X} \to \mathit{Sign}, \overline{\sqsubseteq})$$

For any abstract valuation $\overline{v} : \mathtt{X} \to \mathit{Sign}$, define $\overline{v} \models g$ inductively:

$$\begin{array}{lcll}
\overline{v} & \models & e_1 < e_2 & \text{if} \quad \overline{[\![e_1]\!]}_{\overline{v}} <^\sharp \overline{[\![e_2]\!]}_{\overline{v}} \\
\overline{v} & \models & e_1 \leq e_2 & \text{if} \quad \overline{[\![e_1]\!]}_{\overline{v}} \leq^\sharp \overline{[\![e_2]\!]}_{\overline{v}} \\
\overline{v} & \models & e_1 = e_2 & \text{if} \quad \overline{[\![e_1]\!]}_{\overline{v}} =^\sharp \overline{[\![e_2]\!]}_{\overline{v}} \\
\overline{v} & \models & e_1 \neq e_2 & \text{if} \quad \overline{[\![e_1]\!]}_{\overline{v}} \neq^\sharp \overline{[\![e_2]\!]}_{\overline{v}} \\
\overline{v} & \models & e_1 \geq e_2 & \text{if} \quad \overline{[\![e_1]\!]}_{\overline{v}} \geq^\sharp \overline{[\![e_2]\!]}_{\overline{v}} \\
\overline{v} & \models & e_1 > e_2 & \text{if} \quad \overline{[\![e_1]\!]}_{\overline{v}} >^\sharp \overline{[\![e_2]\!]}_{\overline{v}}
\end{array}$$

Fact (Conservative Approximation)

$$\text{if} \quad v \models g \quad \text{for some} \quad v \in \gamma(\overline{v}) \qquad \text{then} \qquad \overline{v} \models g$$

Forward Sign Analysis: Approximate Transfer Mapping

$$(\mathcal{P}(\mathtt{X} \to \mathbb{R}), \subseteq) \underset{\alpha_\pi}{\overset{\gamma_\pi}{\leftrightarrows}} (\mathtt{X} \to \mathcal{P}(\mathbb{R}), \subseteq) \underset{\alpha_{sign}}{\overset{\gamma_{sign}}{\leftrightarrows}} (\mathtt{X} \to \mathit{Sign}, \overline{\sqsubseteq})$$

$$\overline{h_{x:=e}}(\overline{v}) = \lambda y . \begin{cases} \overline{v}(y) & \text{if } y \neq x \\ \overline{[\![e]\!]}_{\overline{v}} & \text{if } y = x \end{cases} \qquad \overline{h_g}(\overline{v}) = \begin{cases} \overline{\bot} & \text{if } \overline{v} \not\models g \\ \overline{v} & \text{if } \overline{v} \models g \end{cases}$$

Fact (Conservative Approximation)

$$f^\sharp_{\mathtt{op}}(\overline{x}) \;\overline{\sqsupseteq}\; \overline{h_{\mathtt{op}}}(\overline{x}) \qquad (\textit{for all } \mathtt{op} \in \mathtt{Op}, \overline{x} \in \mathtt{X} \to \mathit{Sign})$$

Vs Transfer Mapping Previously Designed by Hand

- ☺ guaranteed to lead to a correct data flow analysis
- ☺ more precise since the previous one was the identity on guards

Forward Sign Analysis: Approximate Transfer Mapping

$$(\mathcal{P}(\mathtt{X} \to \mathbb{R}), \subseteq) \underset{\alpha_\pi}{\overset{\gamma_\pi}{\leftrightarrows}} (\mathtt{X} \to \mathcal{P}(\mathbb{R}), \subseteq) \underset{\alpha_{sign}}{\overset{\gamma_{sign}}{\leftrightarrows}} (\mathtt{X} \to \mathit{Sign}, \overline{\sqsubseteq})$$

$$\overline{h_{x:=e}}(\overline{v}) = \lambda y . \begin{cases} \overline{v}(y) & \text{if } y \neq x \\ \overline{[\![e]\!]}_{\overline{v}} & \text{if } y = x \end{cases} \qquad \overline{h_g}(\overline{v}) = \begin{cases} \overline{\bot} & \text{if } \overline{v} \not\models g \\ \overline{v} & \text{if } \overline{v} \models g \end{cases}$$

Fact (Conservative Approximation)

$$f^\sharp_{\mathtt{op}}(\overline{x}) \;\overline{\sqsupseteq}\; \overline{h_{\mathtt{op}}}(\overline{x}) \qquad (\textit{for all } \mathtt{op} \in \mathtt{Op}, \overline{x} \in \mathtt{X} \to \mathit{Sign})$$

Vs Transfer Mapping Previously Designed by Hand

- ☺ guaranteed to lead to a correct data flow analysis
- ☺ more precise since the previous one was the identity on guards

Forward Sign Analysis: Approximate Transfer Mapping

$$(\mathcal{P}(\mathtt{X} \to \mathbb{R}), \subseteq) \underset{\alpha_\pi}{\overset{\gamma_\pi}{\leftrightarrows}} (\mathtt{X} \to \mathcal{P}(\mathbb{R}), \subseteq) \underset{\alpha_{sign}}{\overset{\gamma_{sign}}{\leftrightarrows}} (\mathtt{X} \to \mathit{Sign}, \overline{\sqsubseteq})$$

$$\overline{h_{x:=e}}(\overline{v}) = \lambda y . \begin{cases} \overline{v}(y) & \text{if } y \neq x \\ \overline{[\![e]\!]}_{\overline{v}} & \text{if } y = x \end{cases} \qquad \overline{h_g}(\overline{v}) = \begin{cases} \overline{\bot} & \text{if } \overline{v} \not\models g \\ \overline{v} & \text{if } \overline{v} \models g \end{cases}$$

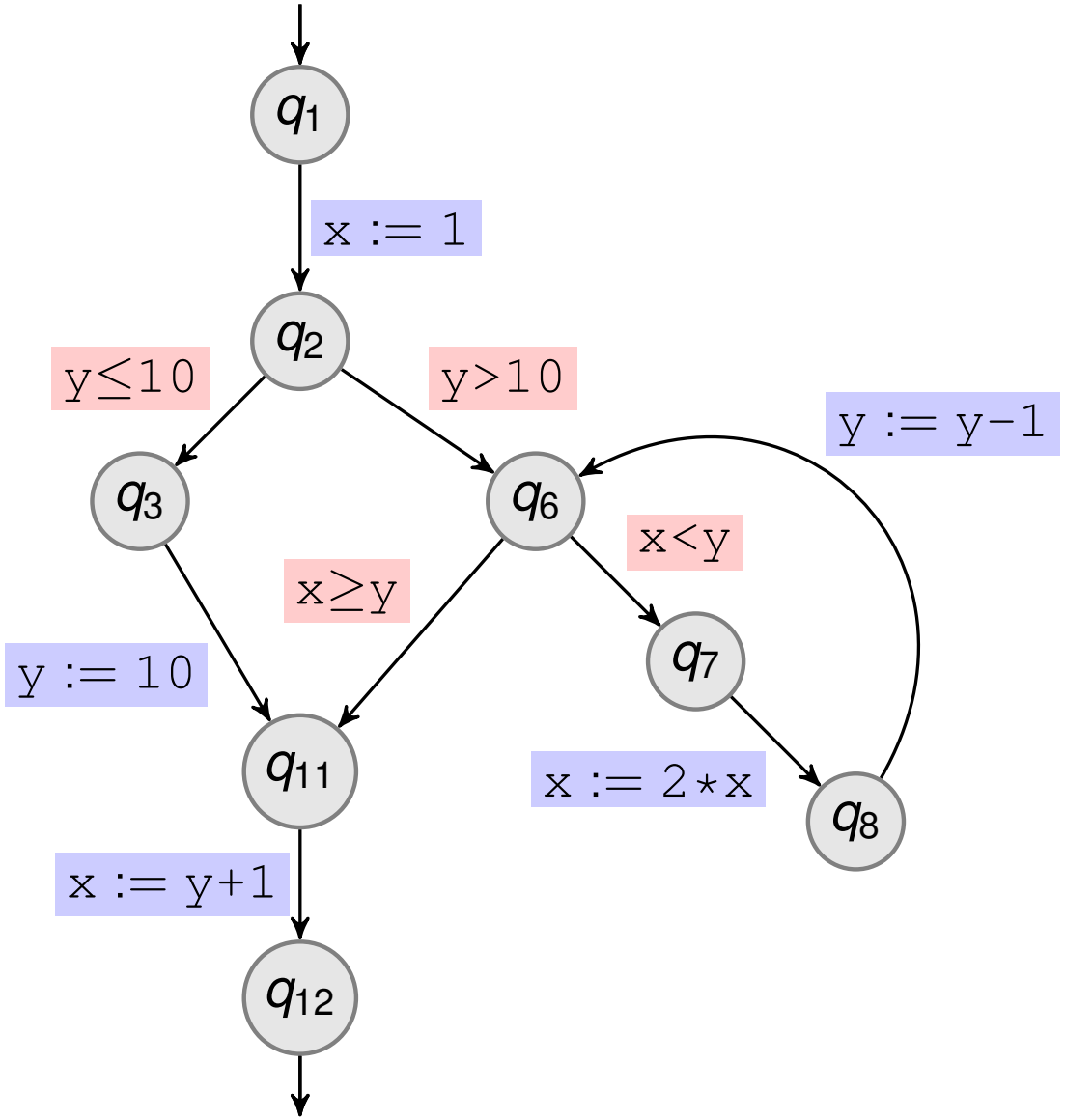
Fact (Conservative Approximation)

$$f^\sharp_{\mathtt{op}}(\overline{x}) \;\overline{\sqsupseteq}\; \overline{h_{\mathtt{op}}}(\overline{x}) \qquad (\textit{for all } \mathtt{op} \in \mathtt{Op}, \overline{x} \in \mathtt{X} \to \mathit{Sign})$$

Vs Transfer Mapping Previously Designed by Hand

- ☺ guaranteed to lead to a correct data flow analysis
- ☺ more precise since the previous one was the identity on guards

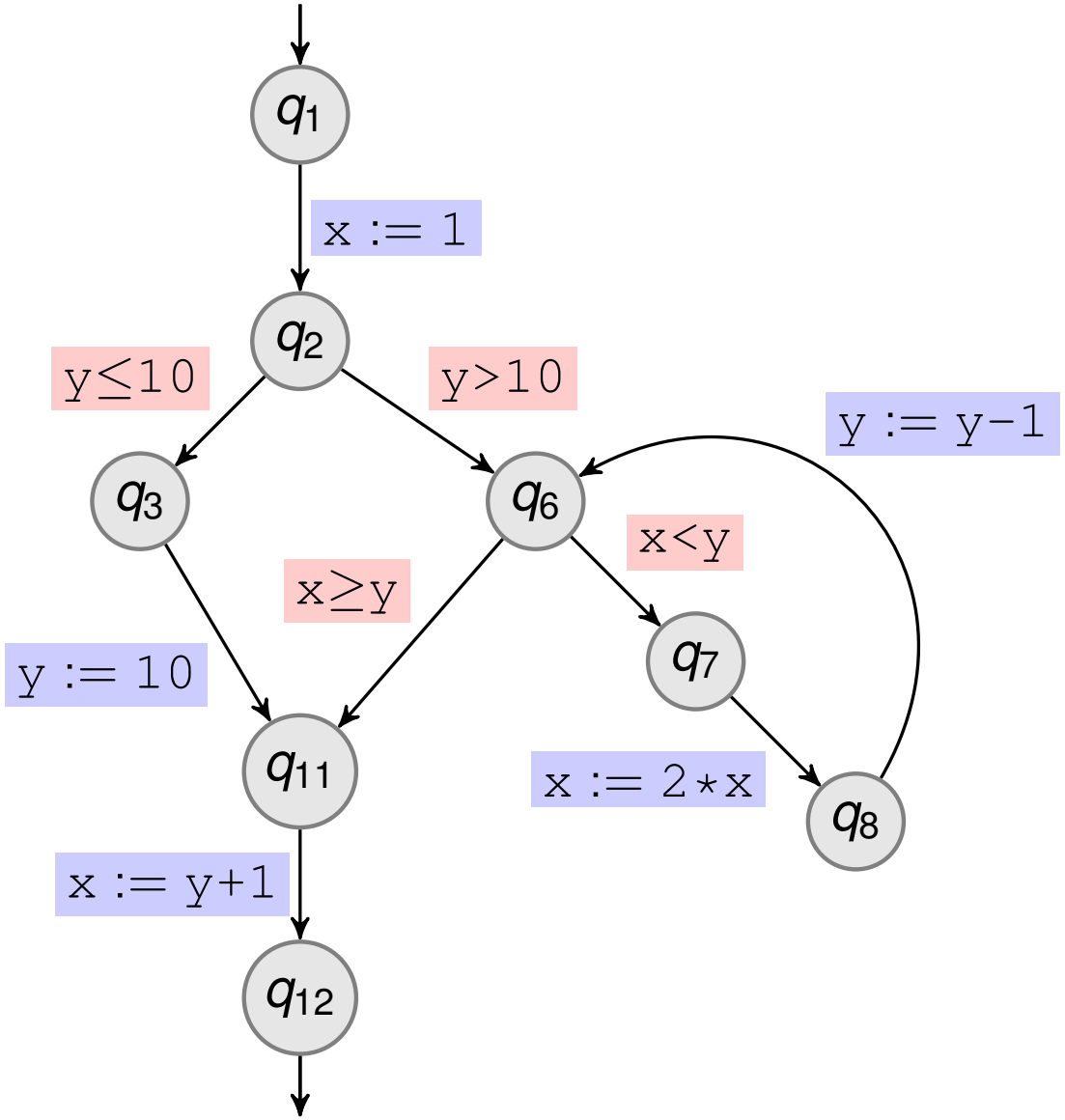
Forward Sign Analysis on Running Example



Goal

Show that $x > 0$ at q_{12}

Forward Sign Analysis on Running Example

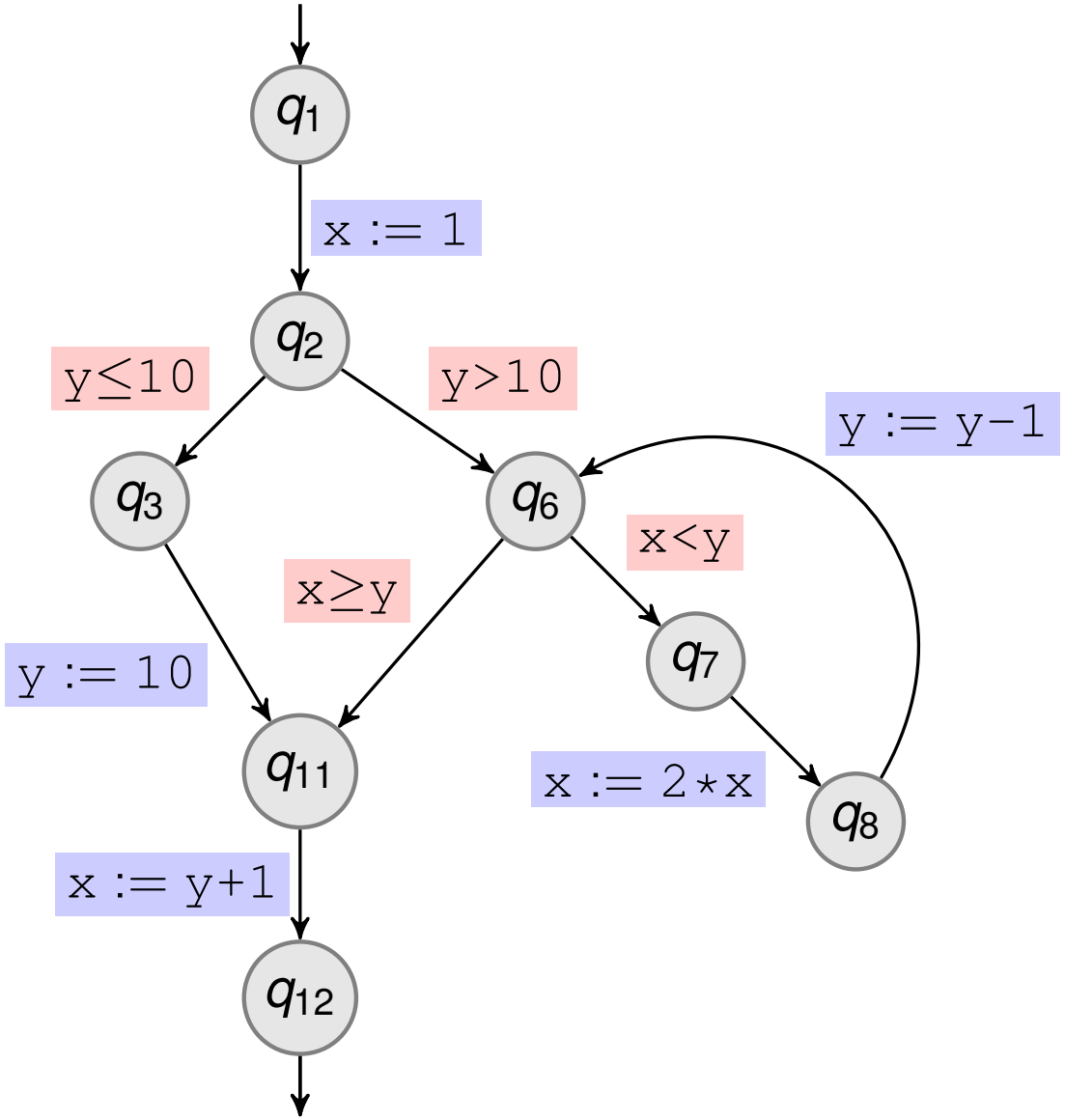


Goal

Show that $x > 0$ at q_{12}

	x	y
q_1	$\top$	$\top$
q_2	$\bot$	$\bot$
q_3	$\bot$	$\bot$
q_6	$\bot$	$\bot$
q_7	$\bot$	$\bot$
q_8	$\bot$	$\bot$
q_{11}	$\bot$	$\bot$
q_{12}	$\bot$	$\bot$

Forward Sign Analysis on Running Example

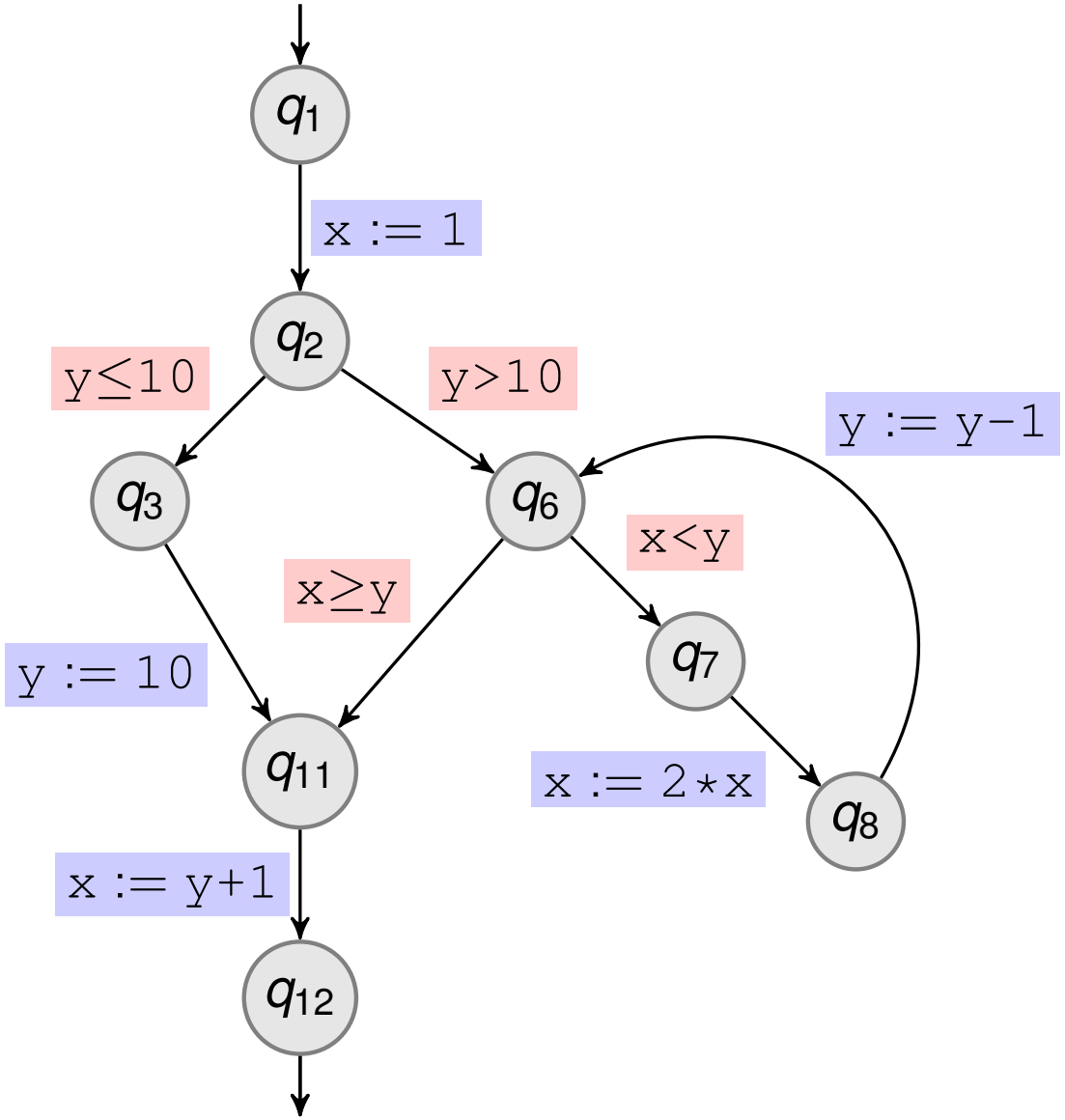


Goal

Show that $x > 0$ at q_{12}

	x	y
q_1	$\top$	$\top$
q_2	$+$	$\top$
q_3	$\bot$	$\bot$
q_6	$\bot$	$\bot$
q_7	$\bot$	$\bot$
q_8	$\bot$	$\bot$
q_{11}	$\bot$	$\bot$
q_{12}	$\bot$	$\bot$

Forward Sign Analysis on Running Example

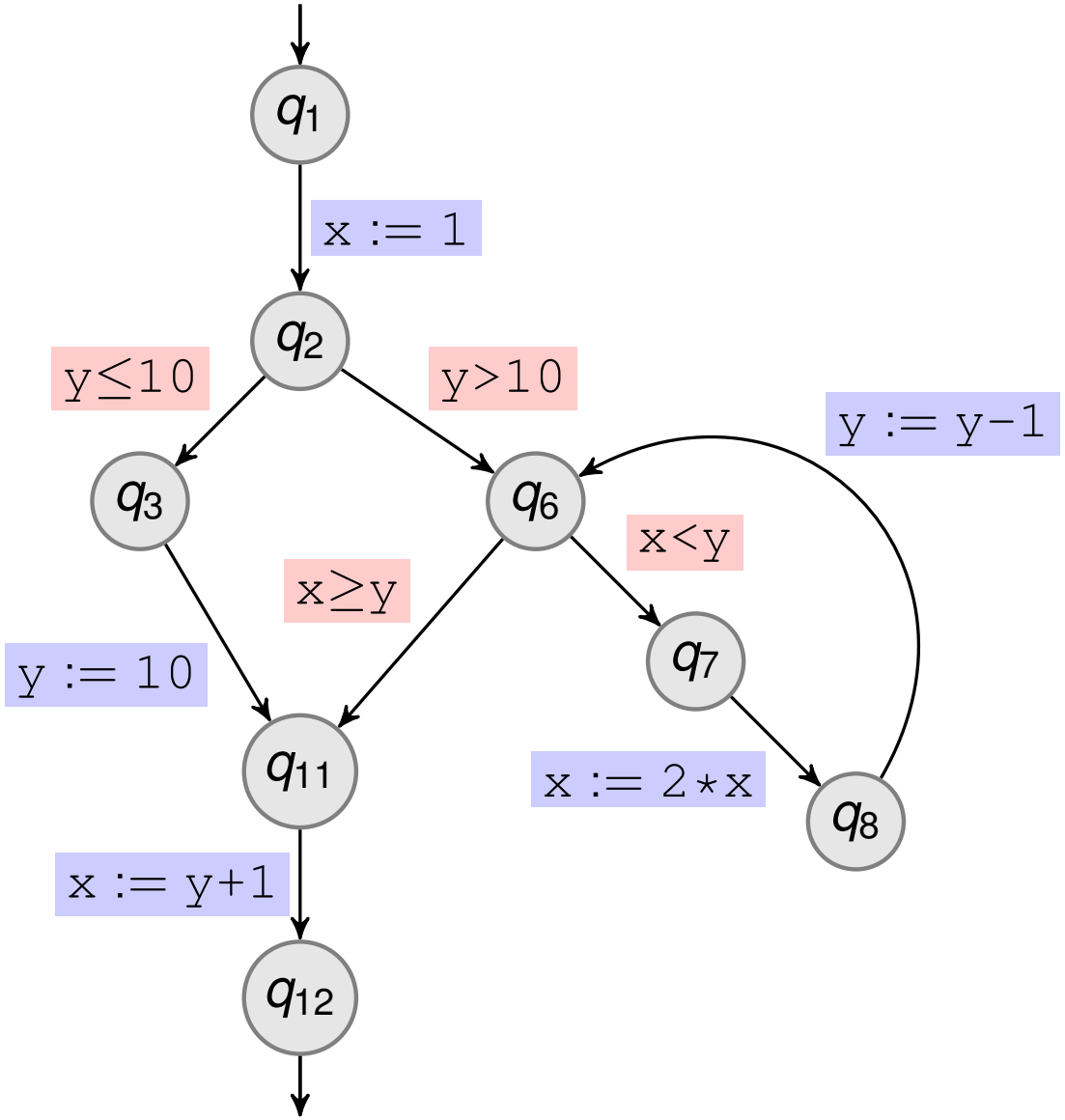


Goal

Show that $x > 0$ at q_{12}

	x	y
q_1	$\top$	$\top$
q_2	$+$	$\top$
q_3	$\bot$	$\bot$
q_6	$\bot$	$\bot$
q_7	$\bot$	$\bot$
q_8	$\bot$	$\bot$
q_{11}	$\bot$	$\bot$
q_{12}	$\bot$	$\bot$

Forward Sign Analysis on Running Example

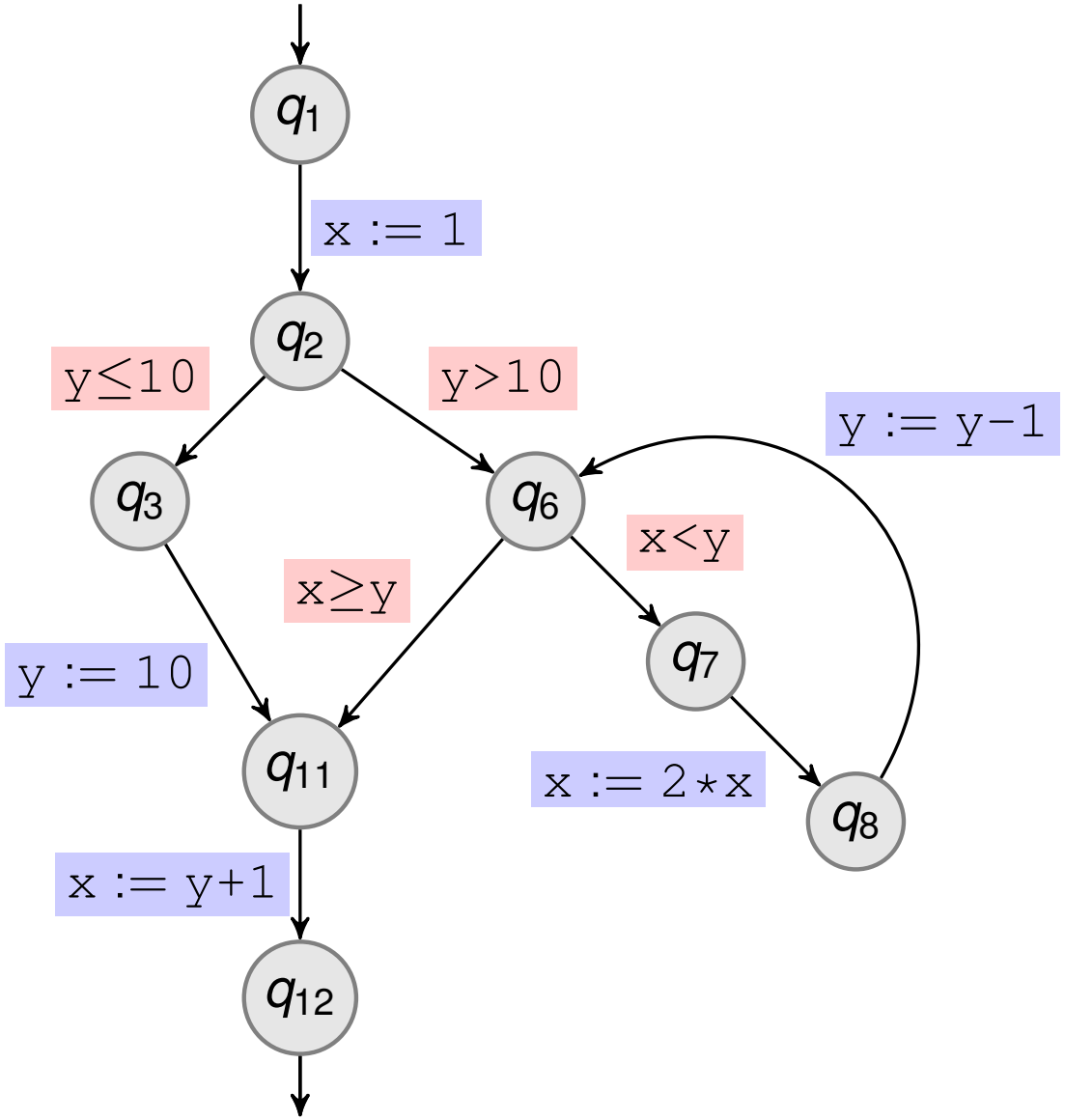


Goal

Show that $x > 0$ at q_{12}

	x	y
q_1	$\top$	$\top$
q_2	$+$	$\top$
q_3	$+$	$\top$
q_6	$\bot$	$\bot$
q_7	$\bot$	$\bot$
q_8	$\bot$	$\bot$
q_{11}	$\bot$	$\bot$
q_{12}	$\bot$	$\bot$

Forward Sign Analysis on Running Example

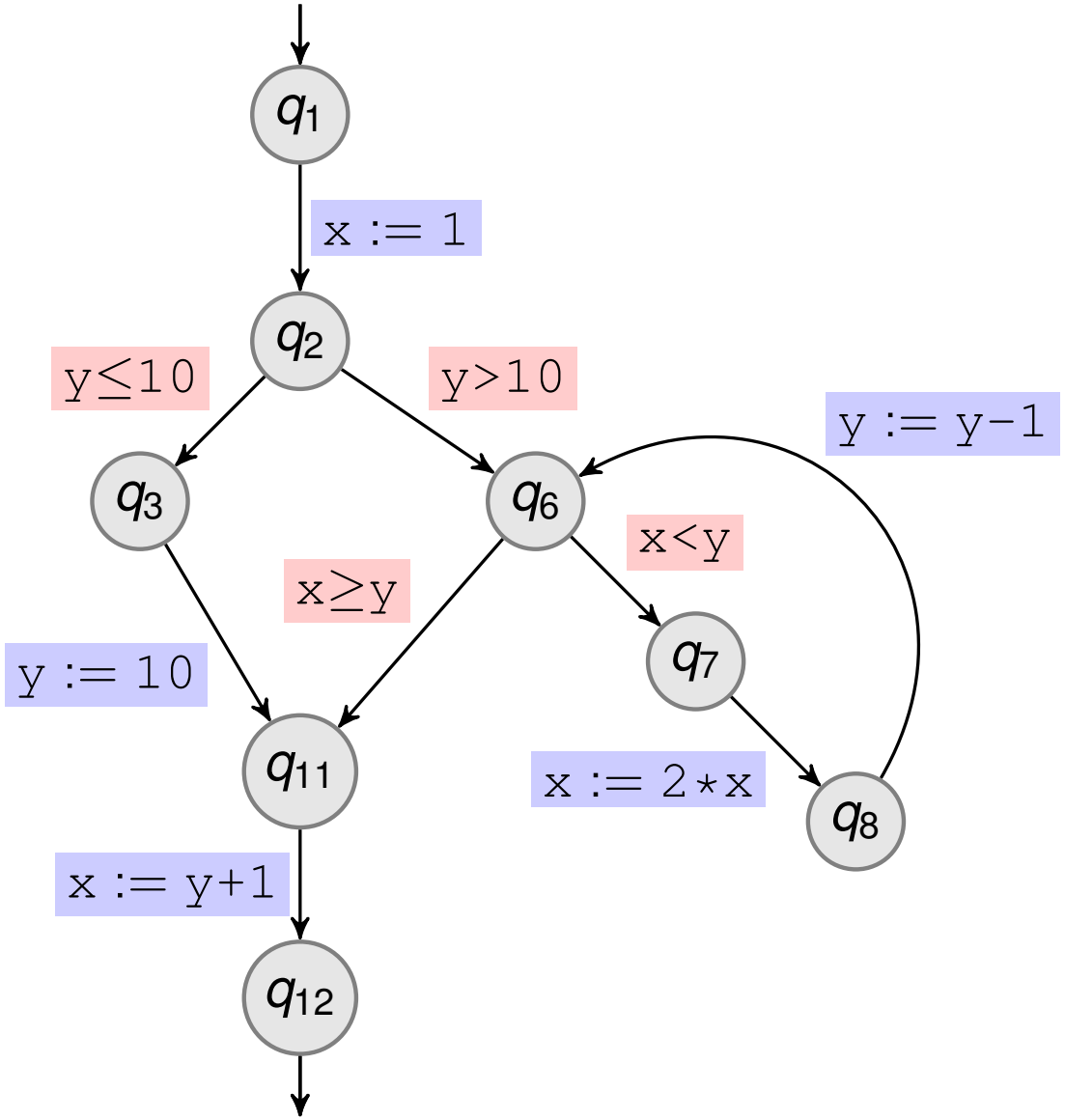


Goal

Show that $x > 0$ at q_{12}

	x	y
q_1	$\top$	$\top$
q_2	$+$	$\top$
q_3	$+$	$\top$
q_6	$\bot$	$\bot$
q_7	$\bot$	$\bot$
q_8	$\bot$	$\bot$
q_{11}	$\bot$	$\bot$
q_{12}	$\bot$	$\bot$

Forward Sign Analysis on Running Example

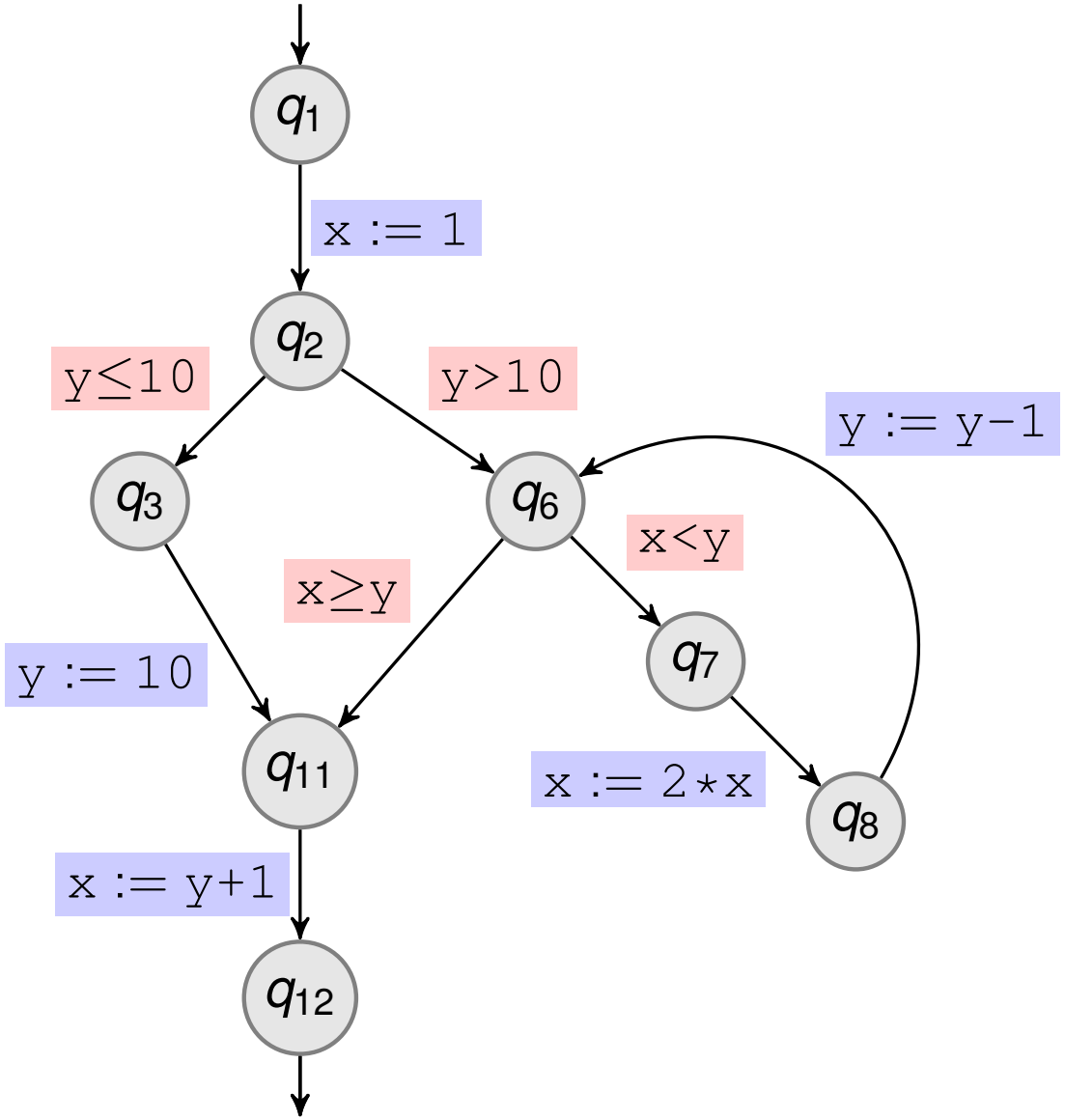


Goal

Show that $x > 0$ at q_{12}

	x	y
q_1	$\top$	$\top$
q_2	$+$	$\top$
q_3	$+$	$\top$
q_6	$\bot$	$\bot$
q_7	$\bot$	$\bot$
q_8	$\bot$	$\bot$
q_{11}	$+$	$+$
q_{12}	$\bot$	$\bot$

Forward Sign Analysis on Running Example

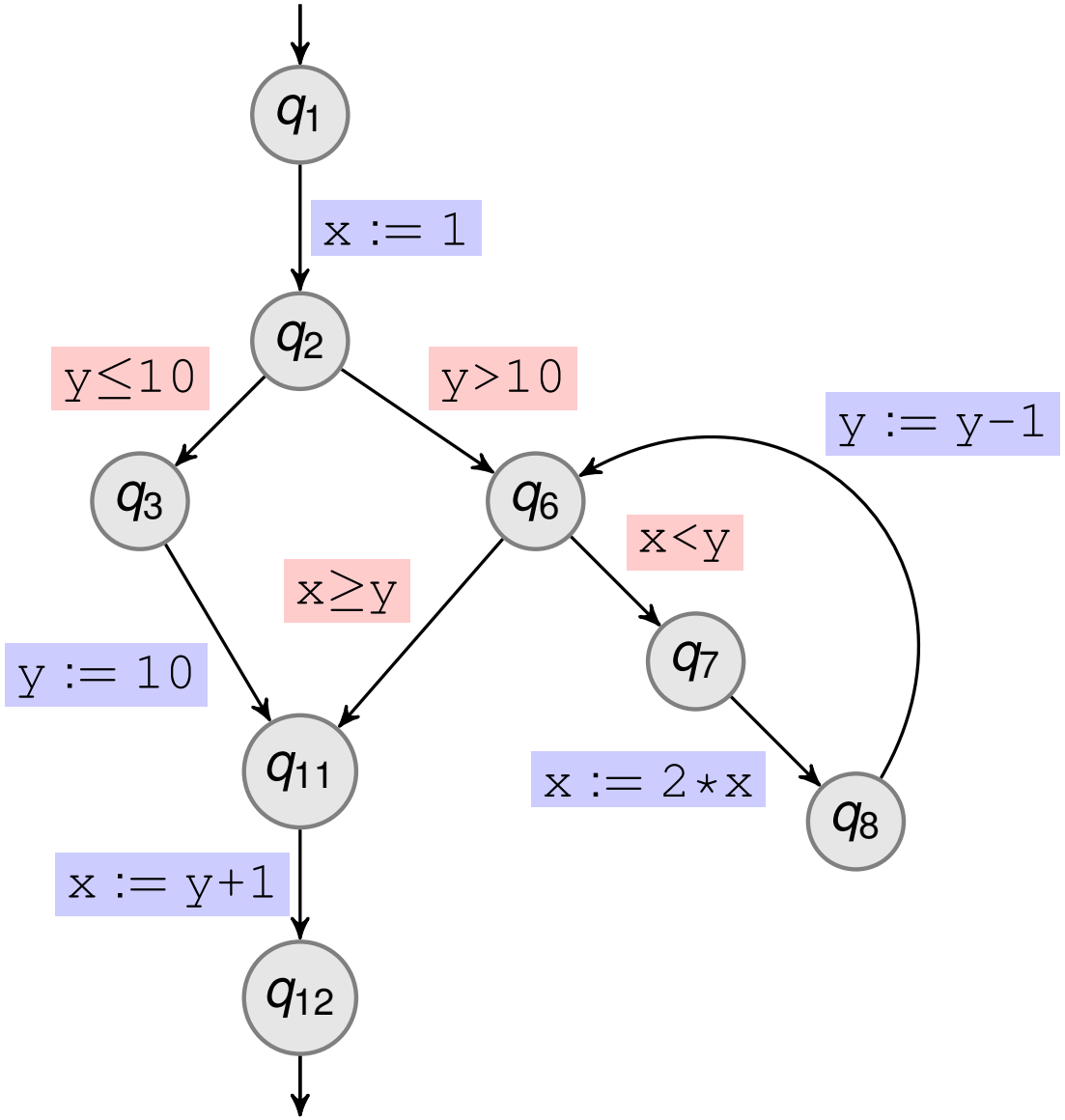


Goal

Show that $x > 0$ at q_{12}

	x	y
q_1	$\top$	$\top$
q_2	$+$	$\top$
q_3	$+$	$\top$
q_6	$\bot$	$\bot$
q_7	$\bot$	$\bot$
q_8	$\bot$	$\bot$
q_{11}	$+$	$+$
q_{12}	$\bot$	$\bot$

Forward Sign Analysis on Running Example

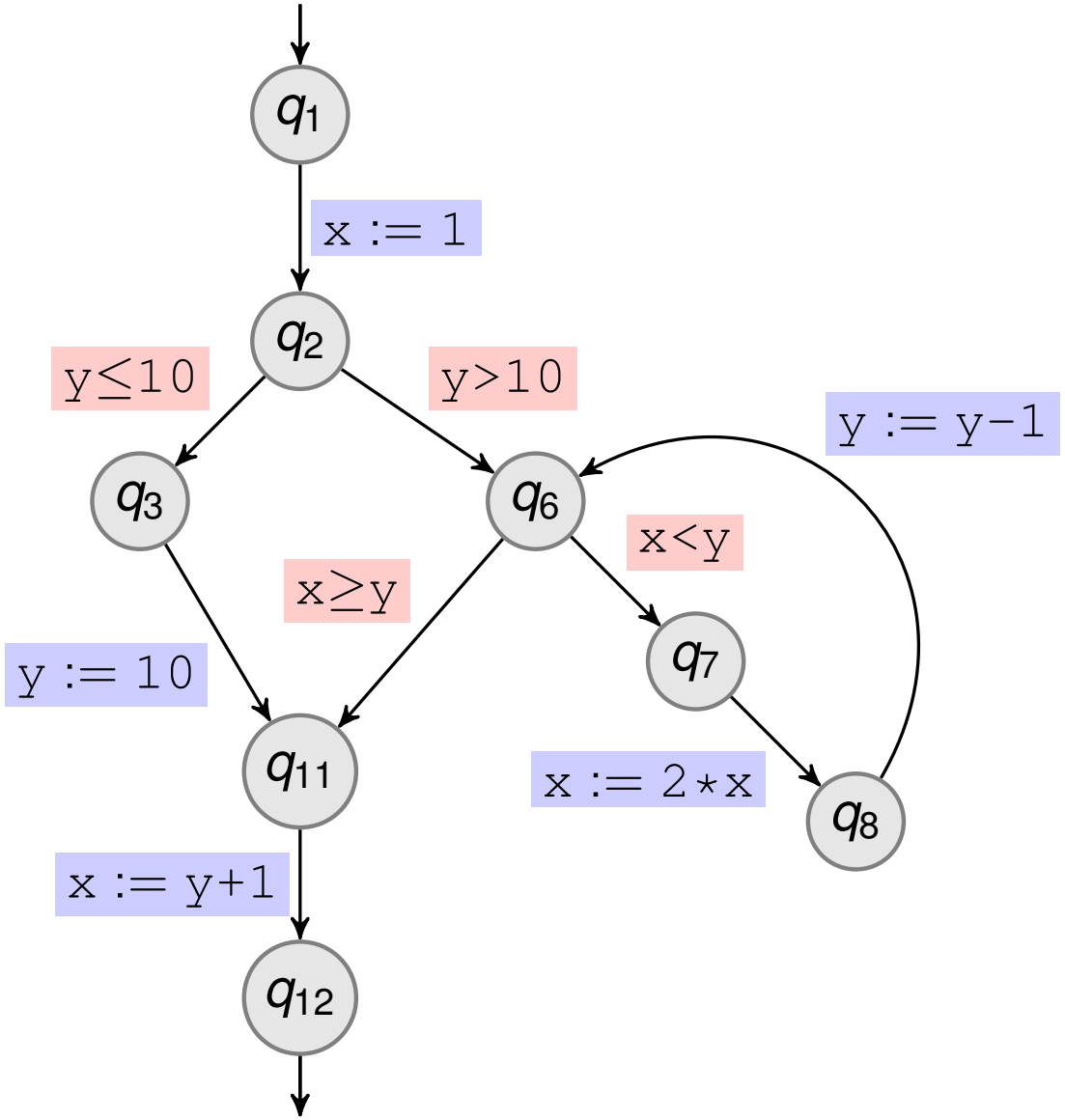


Goal

Show that $x > 0$ at q_{12}

	x	y
q_1	$\top$	$\top$
q_2	$+$	$\top$
q_3	$+$	$\top$
q_6	$\bot$	$\bot$
q_7	$\bot$	$\bot$
q_8	$\bot$	$\bot$
q_{11}	$+$	$+$
q_{12}	$+$	$+$

Forward Sign Analysis on Running Example

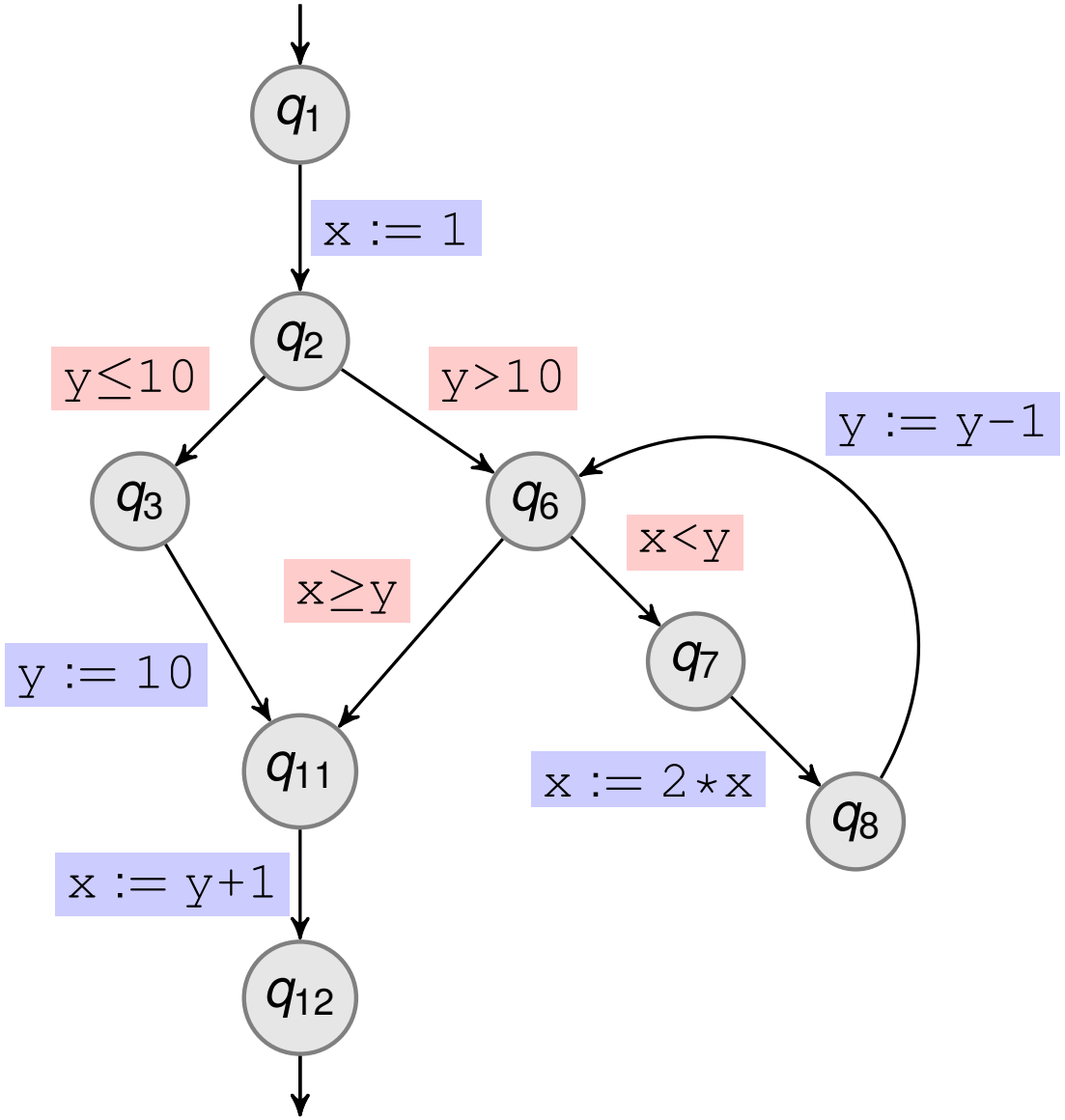


Goal

Show that $x > 0$ at q_{12}

	x	y
q_1	$\top$	$\top$
q_2	$+$	$\top$
q_3	$+$	$\top$
q_6	$\bot$	$\bot$
q_7	$\bot$	$\bot$
q_8	$\bot$	$\bot$
q_{11}	$+$	$+$
q_{12}	$+$	$+$

Forward Sign Analysis on Running Example

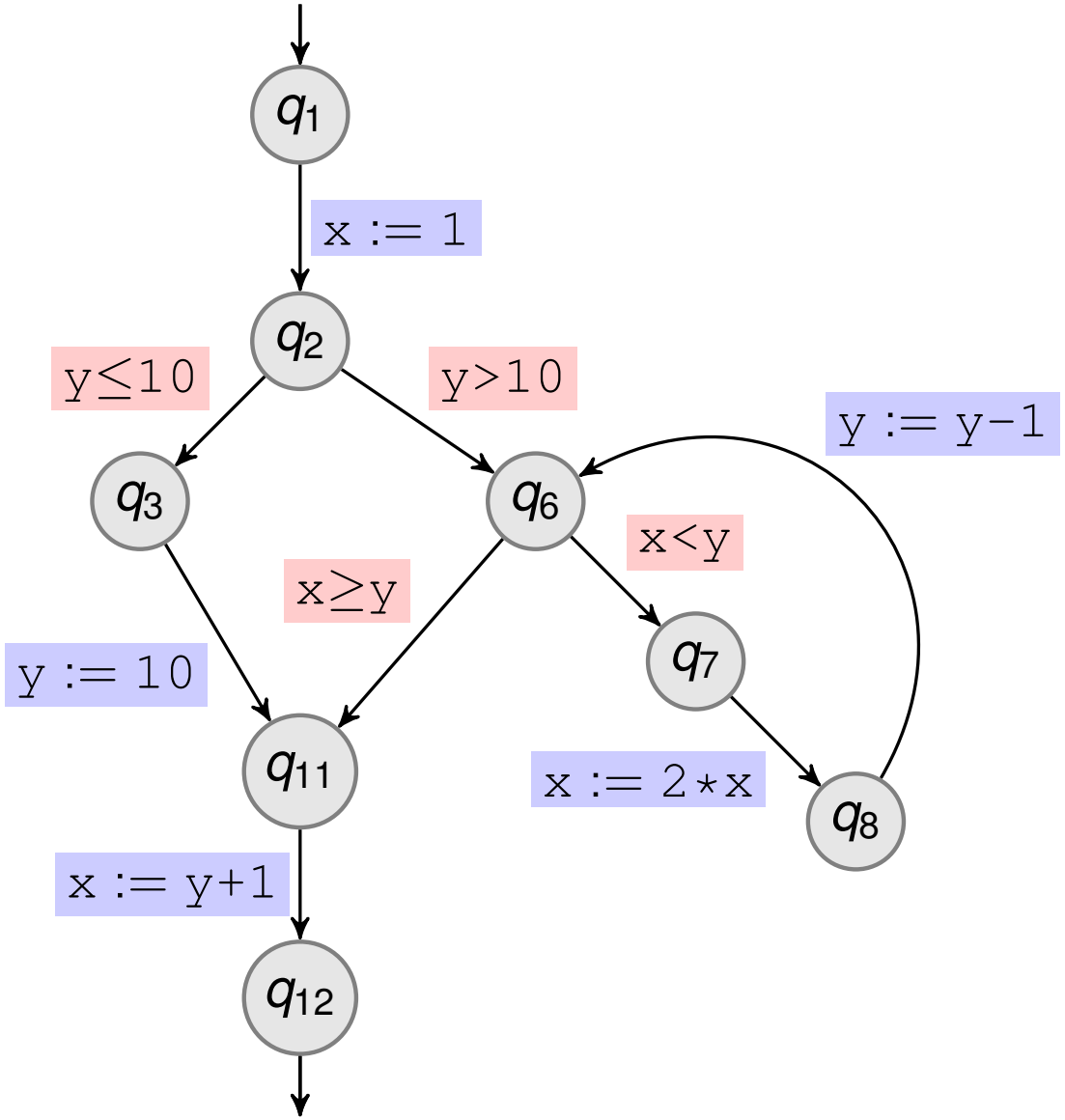


Goal

Show that $x > 0$ at q_{12}

	x	y
q_1	$\top$	$\top$
q_2	$+$	$\top$
q_3	$+$	$\top$
q_6	$+$	$\top$
q_7	$\bot$	$\bot$
q_8	$\bot$	$\bot$
q_{11}	$+$	$+$
q_{12}	$+$	$+$

Forward Sign Analysis on Running Example

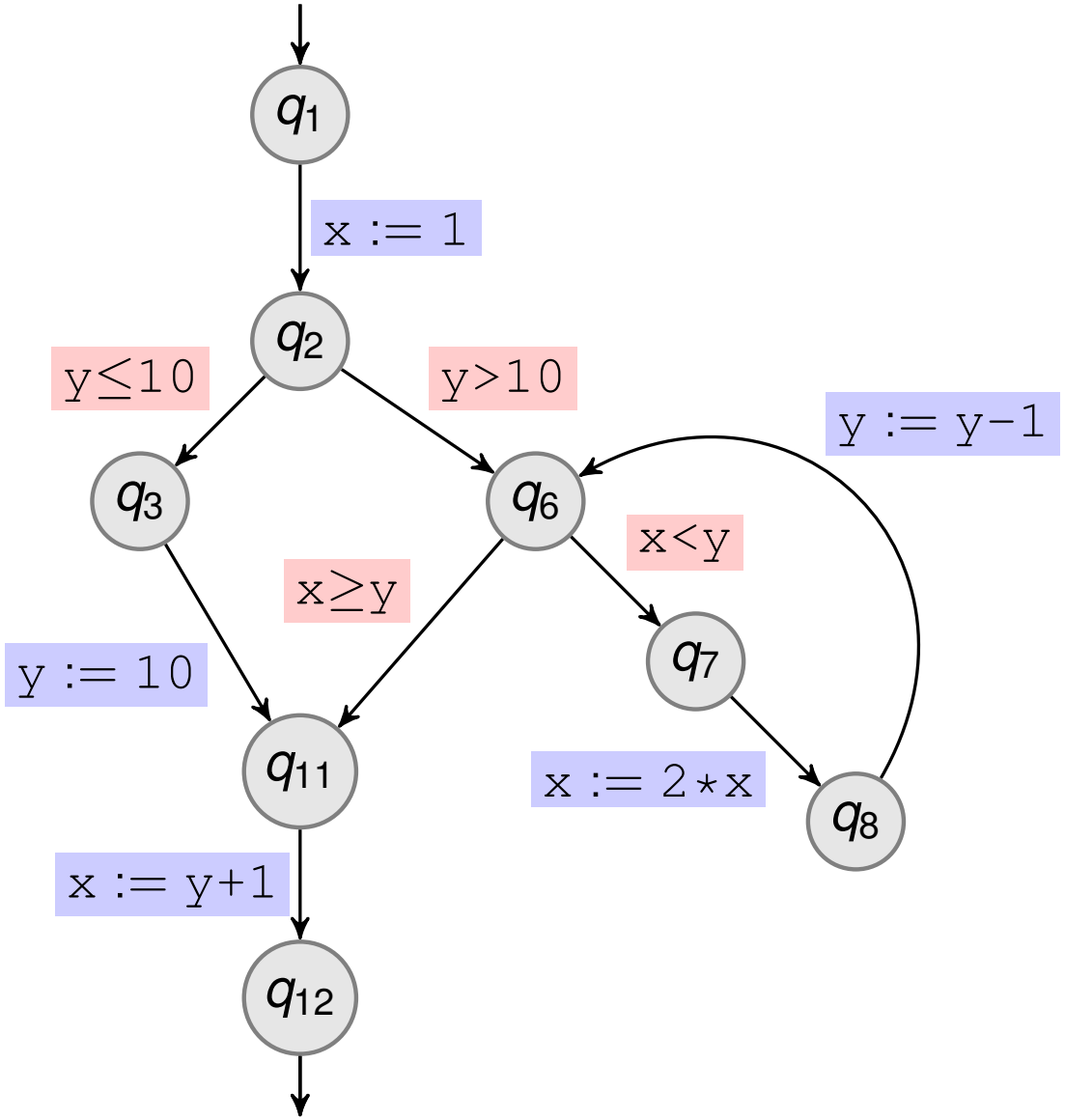


Goal

Show that $x > 0$ at q_{12}

	x	y
q_1	$\top$	$\top$
q_2	$+$	$\top$
q_3	$+$	$\top$
q_6	$+$	$\top$
q_7	$\bot$	$\bot$
q_8	$\bot$	$\bot$
q_{11}	$+$	$+$
q_{12}	$+$	$+$

Forward Sign Analysis on Running Example

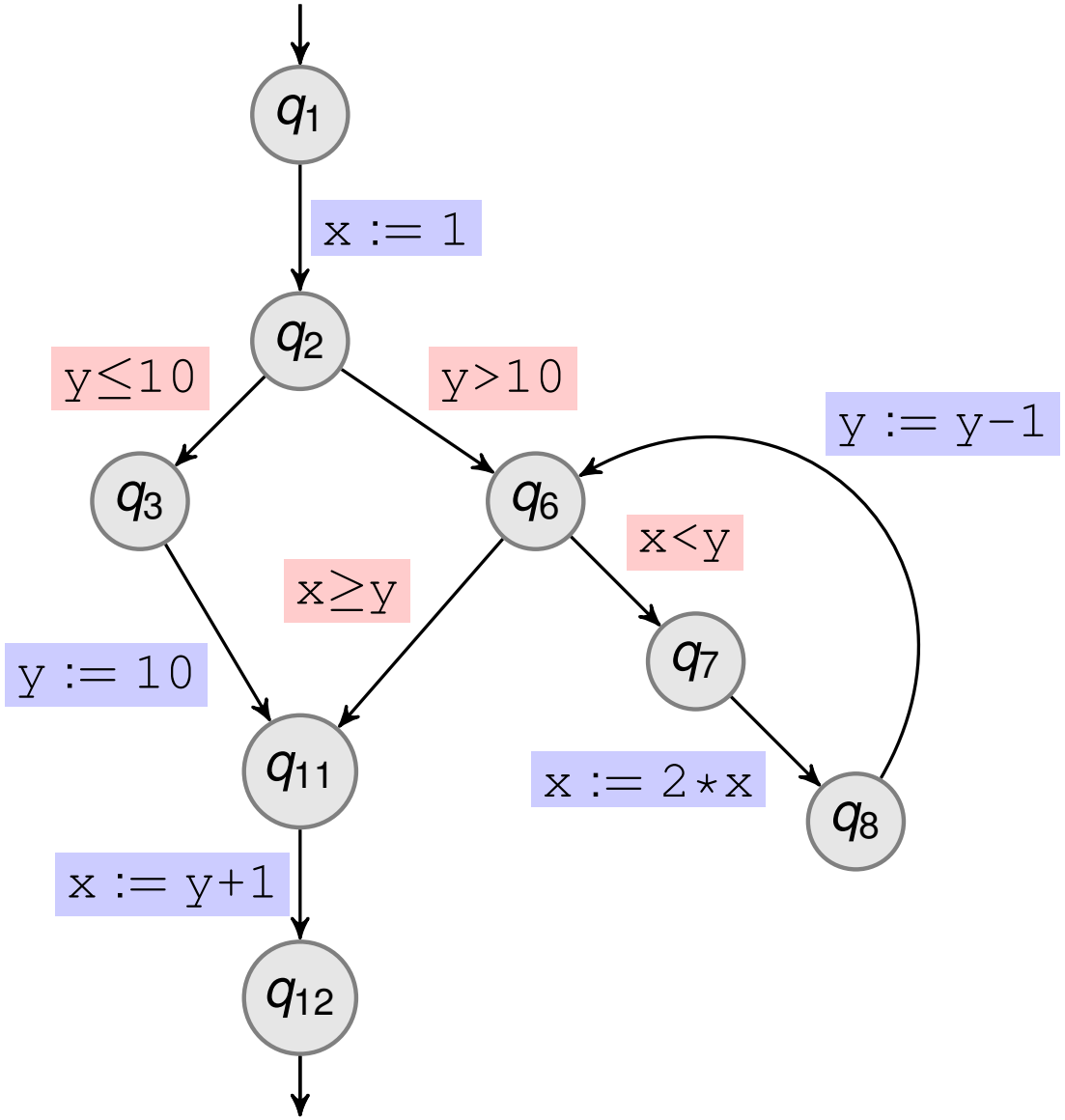


Goal

Show that $x > 0$ at q_{12}

	x	y
q_1	$\top$	$\top$
q_2	$+$	$\top$
q_3	$+$	$\top$
q_6	$+$	$\top$
q_7	$+$	$\top$
q_8	$\bot$	$\bot$
q_{11}	$+$	$+$
q_{12}	$+$	$+$

Forward Sign Analysis on Running Example

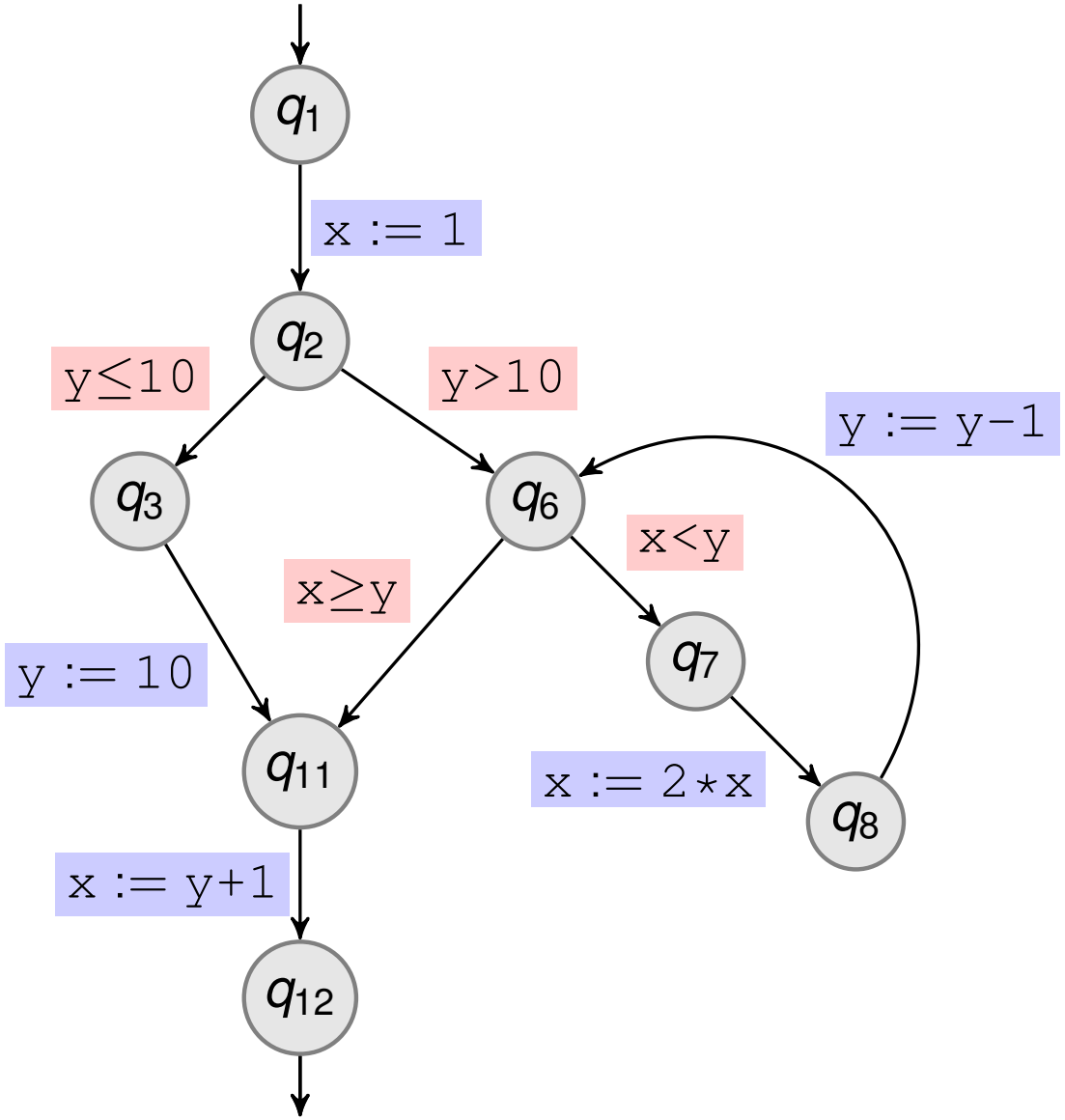


Goal

Show that $x > 0$ at q_{12}

	x	y
q_1	$\top$	$\top$
q_2	$+$	$\top$
q_3	$+$	$\top$
q_6	$+$	$\top$
q_7	$+$	$\top$
q_8	$\bot$	$\bot$
q_{11}	$+$	$+$
q_{12}	$+$	$+$

Forward Sign Analysis on Running Example

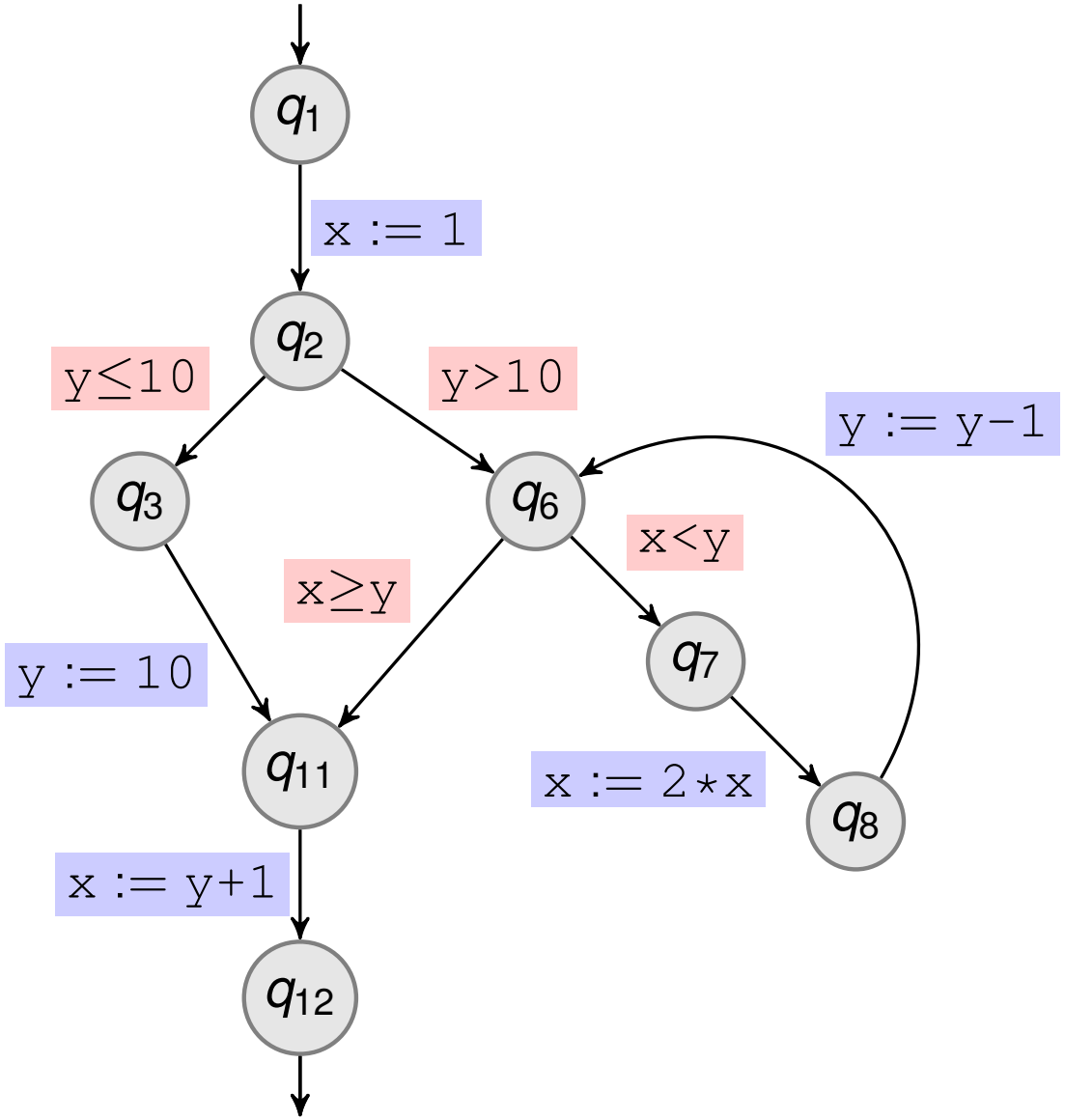


Goal

Show that $x > 0$ at q_{12}

	x	y
q_1	$\top$	$\top$
q_2	$+$	$\top$
q_3	$+$	$\top$
q_6	$+$	$\top$
q_7	$+$	$\top$
q_8	$+$	$\top$
q_{11}	$+$	$+$
q_{12}	$+$	$+$

Forward Sign Analysis on Running Example

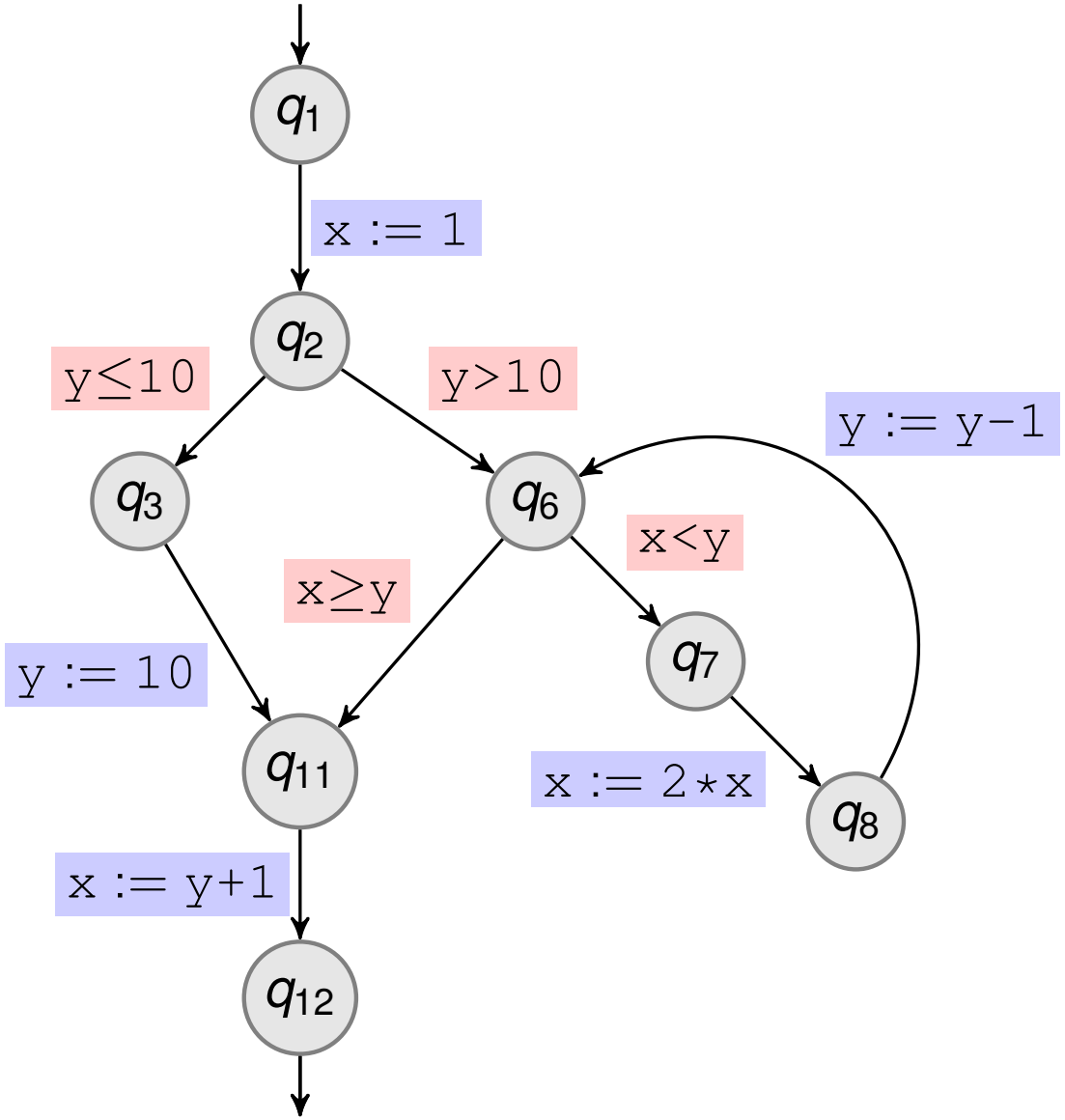


Goal

Show that $x > 0$ at q_{12}

	x	y
q_1	$\top$	$\top$
q_2	$+$	$\top$
q_3	$+$	$\top$
q_6	$+$	$\top$
q_7	$+$	$\top$
q_8	$+$	$\top$
q_{11}	$+$	$+$
q_{12}	$+$	$+$

Forward Sign Analysis on Running Example

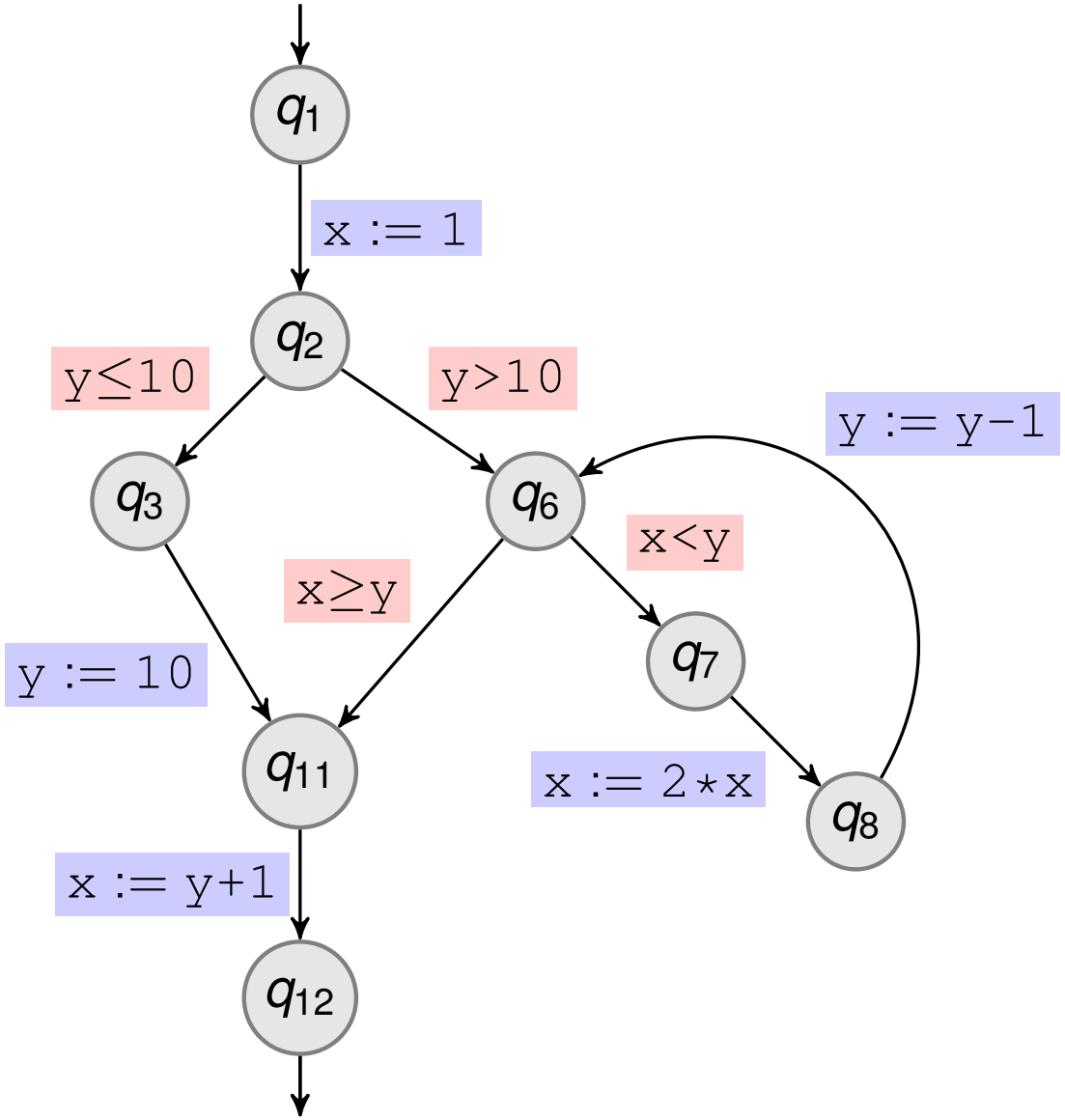


Goal

Show that $x > 0$ at q_{12}

	x	y
q_1	$\top$	$\top$
q_2	$+$	$\top$
q_3	$+$	$\top$
q_6	$+$	$\top$
q_7	$+$	$\top$
q_8	$+$	$\top$
q_{11}	$+$	$\top$
q_{12}	$+$	$+$

Forward Sign Analysis on Running Example

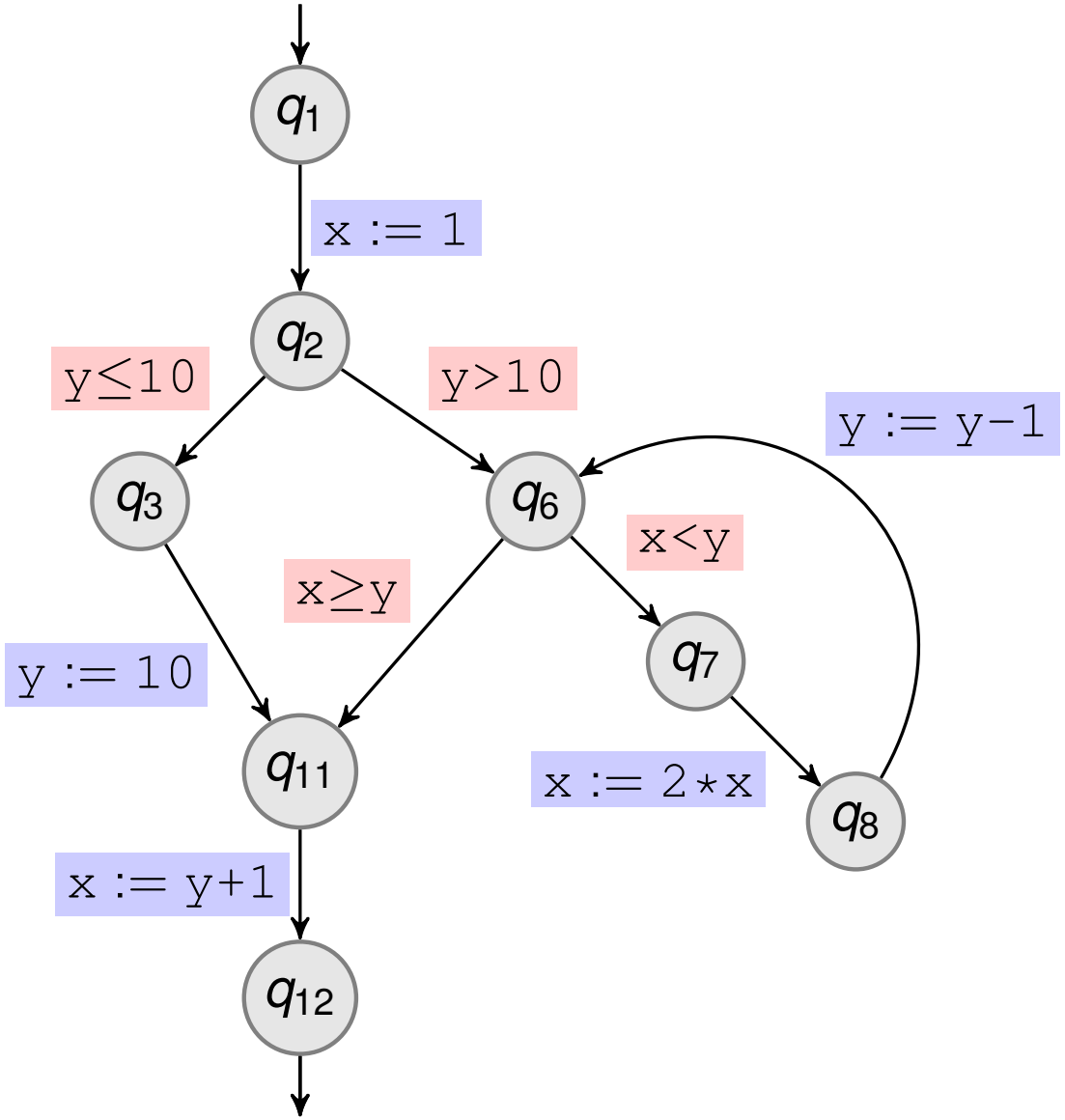


Goal

Show that $x > 0$ at q_{12}

	x	y
q_1	$\top$	$\top$
q_2	$+$	$\top$
q_3	$+$	$\top$
q_6	$+$	$\top$
q_7	$+$	$\top$
q_8	$+$	$\top$
q_{11}	$+$	$\top$
q_{12}	$+$	$+$

Forward Sign Analysis on Running Example

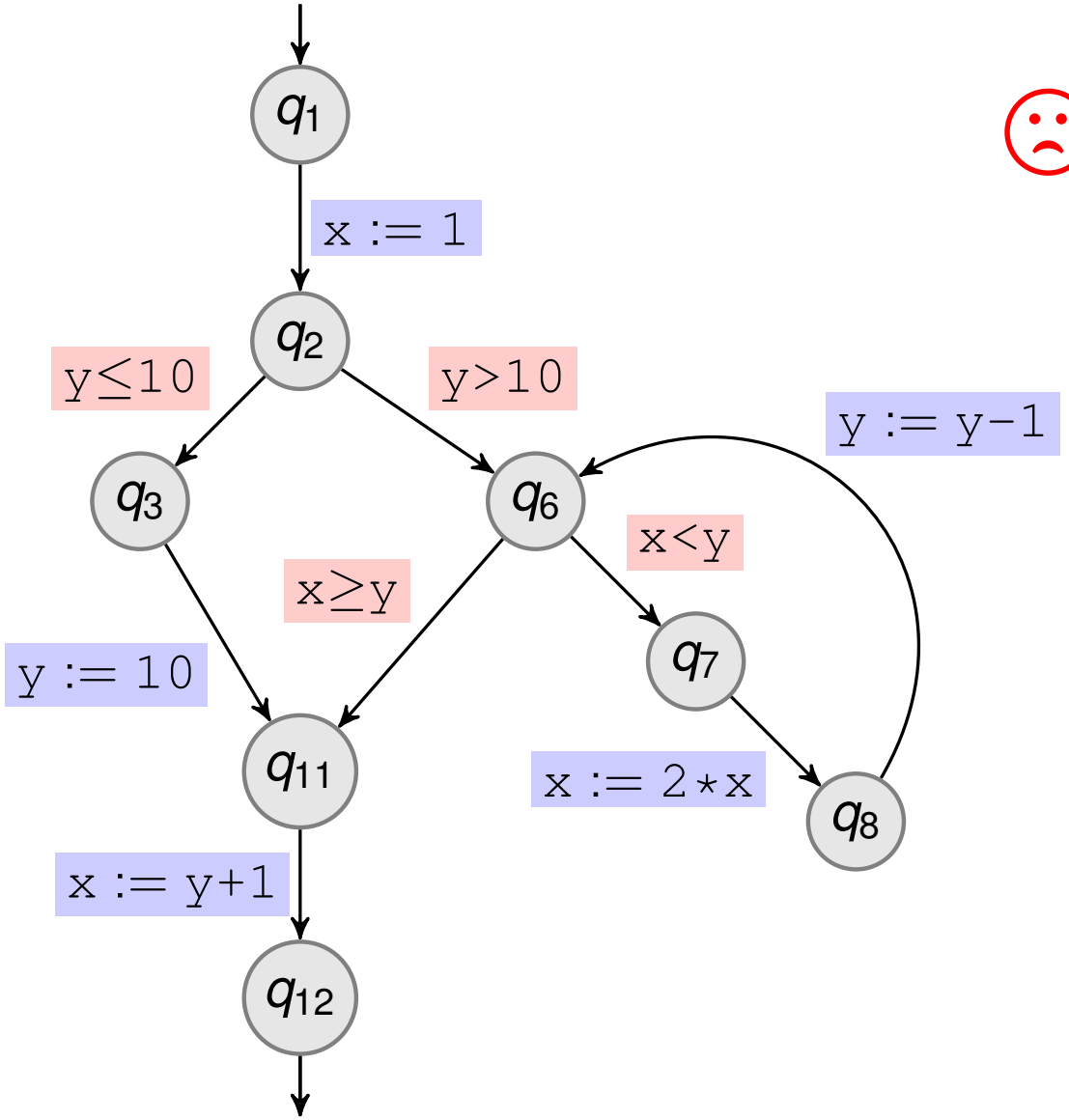


Goal

Show that $x > 0$ at q_{12}

	x	y
q_1	$\top$	$\top$
q_2	$+$	$\top$
q_3	$+$	$\top$
q_6	$+$	$\top$
q_7	$+$	$\top$
q_8	$+$	$\top$
q_{11}	$+$	$\top$
q_{12}	$\top$	$\top$

Forward Sign Analysis on Running Example



Goal

Show that $x > 0$ at q_{12}

	x	y
q_1	$\top$	$\top$
q_2	$+$	$\top$
q_3	$+$	$\top$
q_6	$+$	$\top$
q_7	$+$	$\top$
q_8	$+$	$\top$
q_{11}	$+$	$\top$
q_{12}	$\top$	$\top$

Loss of Precision with Approximate Transfer Mapping

Example (Assignment $\mathtt{op} = \mathtt{x} := \mathtt{z} * \mathtt{z}$)

$$f^\sharp_{\mathtt{op}}(\overline{\top}) = \lambda y . \begin{cases} \overline{\top} & \text{if } y \neq \mathtt{x} \\ 0+ & \text{if } y = \mathtt{x} \end{cases} \qquad \overline{h_{\mathtt{op}}}(\overline{\top}) = \lambda y . \begin{cases} \overline{\top} & \text{if } y \neq \mathtt{x} \\ \overline{\top} & \text{if } y = \mathtt{x} \end{cases}$$

Indeed with $\overline{h_{\mathtt{op}}}$ the new value for $\mathtt{x}$ is: $\overline{[\![\mathtt{z} * \mathtt{z}]\!]}_{\overline{\top}} = \overline{[\![\mathtt{z}]\!]}_{\overline{\top}} \times^\sharp \overline{[\![\mathtt{z}]\!]}_{\overline{\top}} = \overline{\top}$.

Example (Guard $\mathtt{op} = \mathtt{x} = 0$)

$$f^\sharp_{\mathtt{op}}(\overline{\top}) = \lambda y . \begin{cases} \overline{\top} & \text{if } y \neq \mathtt{x} \\ 0 & \text{if } y = \mathtt{x} \end{cases} \qquad \overline{h_{\mathtt{op}}}(\overline{\top}) = \lambda y . \begin{cases} \overline{\top} & \text{if } y \neq \mathtt{x} \\ \overline{\top} & \text{if } y = \mathtt{x} \end{cases}$$

Indeed $\overline{h_{\mathtt{op}}}(\overline{v})$ is either $\overline{\bot}$ (if $\overline{v} \not\models g$) or $\overline{v}$.

Example (Guard $\mathtt{op} = \mathtt{x} > \mathtt{x}$)

$$f^\sharp_{\mathtt{op}}(\overline{\top}) = \overline{\bot} \qquad \overline{h_{\mathtt{op}}}(\overline{\top}) = \overline{\top}$$

Loss of Precision with Approximate Transfer Mapping

Example (Assignment $\mathtt{op} = \mathtt{x} := \mathtt{z} * \mathtt{z}$)

$$f^\sharp_{\mathtt{op}}(\overline{\top}) = \lambda y . \begin{cases} \overline{\top} & \text{if } y \neq \mathtt{x} \\ 0+ & \text{if } y = \mathtt{x} \end{cases} \qquad \overline{h_{\mathtt{op}}}(\overline{\top}) = \lambda y . \begin{cases} \overline{\top} & \text{if } y \neq \mathtt{x} \\ \overline{\top} & \text{if } y = \mathtt{x} \end{cases}$$

Indeed with $\overline{h_{\mathtt{op}}}$ the new value for $\mathtt{x}$ is: $\overline{[\![\mathtt{z} * \mathtt{z}]\!]}_{\overline{\top}} = \overline{[\![\mathtt{z}]\!]}_{\overline{\top}} \times^\sharp \overline{[\![\mathtt{z}]\!]}_{\overline{\top}} = \overline{\top}$.

Example (Guard $\mathtt{op} = \mathtt{x} = 0$)

$$f^\sharp_{\mathtt{op}}(\overline{\top}) = \lambda y . \begin{cases} \overline{\top} & \text{if } y \neq \mathtt{x} \\ 0 & \text{if } y = \mathtt{x} \end{cases} \qquad \overline{h_{\mathtt{op}}}(\overline{\top}) = \lambda y . \begin{cases} \overline{\top} & \text{if } y \neq \mathtt{x} \\ \overline{\top} & \text{if } y = \mathtt{x} \end{cases}$$

Indeed $\overline{h_{\mathtt{op}}}(\overline{v})$ is either $\overline{\bot}$ (if $\overline{v} \not\models g$) or $\overline{v}$.

Example (Guard $\mathtt{op} = \mathtt{x} > \mathtt{x}$)

$$f^\sharp_{\mathtt{op}}(\overline{\top}) = \overline{\bot} \qquad \overline{h_{\mathtt{op}}}(\overline{\top}) = \overline{\top}$$

Loss of Precision with Approximate Transfer Mapping

Example (Assignment $\mathtt{op} = \mathtt{x} := \mathtt{z} * \mathtt{z}$)

$$f^\sharp_{\mathtt{op}}(\overline{\top}) = \lambda y . \begin{cases} \overline{\top} & \text{if } y \neq \mathtt{x} \\ 0+ & \text{if } y = \mathtt{x} \end{cases} \qquad \overline{h_{\mathtt{op}}}(\overline{\top}) = \lambda y . \begin{cases} \overline{\top} & \text{if } y \neq \mathtt{x} \\ \overline{\top} & \text{if } y = \mathtt{x} \end{cases}$$

Indeed with $\overline{h_{\mathtt{op}}}$ the new value for $\mathtt{x}$ is: $\overline{[\![\mathtt{z} * \mathtt{z}]\!]}_{\overline{\top}} = \overline{[\![\mathtt{z}]\!]}_{\overline{\top}} \times^\sharp \overline{[\![\mathtt{z}]\!]}_{\overline{\top}} = \overline{\top}$.

Example (Guard $\mathtt{op} = \mathtt{x} = 0$)

$$f^\sharp_{\mathtt{op}}(\overline{\top}) = \lambda y . \begin{cases} \overline{\top} & \text{if } y \neq \mathtt{x} \\ 0 & \text{if } y = \mathtt{x} \end{cases} \qquad \overline{h_{\mathtt{op}}}(\overline{\top}) = \lambda y . \begin{cases} \overline{\top} & \text{if } y \neq \mathtt{x} \\ \overline{\top} & \text{if } y = \mathtt{x} \end{cases}$$

Indeed $\overline{h_{\mathtt{op}}}(\overline{v})$ is either $\overline{\bot}$ (if $\overline{v} \not\models g$) or $\overline{v}$.

Example (Guard $\mathtt{op} = \mathtt{x} > \mathtt{x}$)

$$f^\sharp_{\mathtt{op}}(\overline{\top}) = \overline{\bot} \qquad \overline{h_{\mathtt{op}}}(\overline{\top}) = \overline{\top}$$

Enhanced Precision with Functional Comparators

Gain information from guards

Functional Extension of Arithmetic Comparators to Subsets of $\mathbb{R}$

For each binary relation $\bowtie \in \{<, \leq, =, \neq, >, \geq, \ldots\}$ on $\mathbb{R}$, define the function $\bowtie : (\mathcal{P}(\mathbb{R}) \times \mathcal{P}(\mathbb{R})) \to (\mathcal{P}(\mathbb{R}) \times \mathcal{P}(\mathbb{R}))$ by:

$$U \bowtie V = (\{u \in U \mid \exists v \in V, u \bowtie v\}, \{v \in V \mid \exists u \in U, u \bowtie v\})$$

Functional Abstract Arithmetic Comparators

Define the best abstraction $\bowtie^\sharp : (\mathit{Sign} \times \mathit{Sign}) \to (\mathit{Sign} \times \mathit{Sign})$ of each function $\bowtie \in \{<, \leq, =, \neq, >, \geq, \ldots\}$ by:

$$\overline{x} \bowtie^\sharp \overline{y} = (\alpha_{sign}(U), \alpha_{sign}(V))$$

$$\text{where} \quad (U, V) = \gamma_{sign}(\overline{x}) \bowtie \gamma_{sign}(\overline{y})$$

Enhanced Precision with Functional Comparators

Gain information from guards

Functional Extension of Arithmetic Comparators to Subsets of $\mathbb{R}$

For each binary relation $\bowtie \in \{<, \leq, =, \neq, >, \geq, \ldots\}$ on $\mathbb{R}$, define the function $\bowtie : (\mathcal{P}(\mathbb{R}) \times \mathcal{P}(\mathbb{R})) \to (\mathcal{P}(\mathbb{R}) \times \mathcal{P}(\mathbb{R}))$ by:

$$U \bowtie V \quad = \quad (\{u \in U \mid \exists v \in V, u \bowtie v\}, \{v \in V \mid \exists u \in U, u \bowtie v\})$$

Functional Abstract Arithmetic Comparators

Define the best abstraction $\bowtie^\sharp : (\mathit{Sign} \times \mathit{Sign}) \to (\mathit{Sign} \times \mathit{Sign})$ of each function $\bowtie \in \{<, \leq, =, \neq, >, \geq, \ldots\}$ by:

$$\overline{x} \bowtie^\sharp \overline{y} \quad = \quad (\alpha_{sign}(U), \alpha_{sign}(V))$$

$$\text{where} \qquad (U, V) \quad = \quad \gamma_{sign}(\overline{x}) \bowtie \gamma_{sign}(\overline{y})$$

Enhanced Precision with Functional Comparators

Gain information from guards

Functional Extension of Arithmetic Comparators to Subsets of $\mathbb{R}$

For each binary relation $\bowtie \in \{<, \leq, =, \neq, >, \geq, \ldots\}$ on $\mathbb{R}$, define the function $\bowtie : (\mathcal{P}(\mathbb{R}) \times \mathcal{P}(\mathbb{R})) \to (\mathcal{P}(\mathbb{R}) \times \mathcal{P}(\mathbb{R}))$ by:

$$U \bowtie V \quad = \quad (\{u \in U \mid \exists v \in V, u \bowtie v\}, \{v \in V \mid \exists u \in U, u \bowtie v\})$$

Functional Abstract Arithmetic Comparators

Define the best abstraction $\bowtie^\sharp : (\mathit{Sign} \times \mathit{Sign}) \to (\mathit{Sign} \times \mathit{Sign})$ of each function $\bowtie \in \{<, \leq, =, \neq, >, \geq, \ldots\}$ by:

$$\overline{x} \bowtie^\sharp \overline{y} \quad = \quad (\alpha_{sign}(U), \alpha_{sign}(V))$$

$$\text{where} \qquad (U, V) \quad = \quad \gamma_{sign}(\overline{x}) \bowtie \gamma_{sign}(\overline{y})$$

Functional Abstract Comparators: Table for $\leq^\sharp$

$$\begin{array}{rrcl} & \overline{x} \bowtie^\sharp \overline{y} & = & \left(\alpha_{sign}(U), \alpha_{sign}(V)\right) \\ \text{where} & (U, V) & = & \gamma_{sign}(\overline{x}) \bowtie \gamma_{sign}(\overline{y}) \end{array}$$

$\leq^\sharp$	$\bot$	$-$	0	$+$	-0	$0+$	$\top$
$\bot$	$(\bot,\bot)$	$(\bot,\bot)$	$(\bot,\bot)$	$(\bot,\bot)$	$(\bot,\bot)$	$(\bot,\bot)$	$(\bot,\bot)$
$-$	$(\bot,\bot)$	$(-,-)$	$(-,0)$	$(-,+)$	$(-,-0)$	$(-,0+)$	$(-,\top)$
0	$(\bot,\bot)$	$(\bot,\bot)$	$(0,0)$	$(0,+)$	$(0,0)$	$(0,0+)$	$(0,\top)$
$+$	$(\bot,\bot)$	$(\bot,\bot)$	$(\bot,\bot)$	$(+,+)$	$(\bot,\bot)$	$(+,0+)$	$(+,\top)$
-0	$(\bot,\bot)$	$(-,-)$	$(-0,0)$	$(-0,+)$	$(-0,-0)$	$(-0,0+)$	$(-0,\top)$
$0+$	$(\bot,\bot)$	$(\bot,\bot)$	$(0,0)$	$(0+,+)$	$(0,0)$	$(0+,0+)$	$(0+,\top)$
$\top$	$(\bot,\bot)$	$(-,-)$	$(-0,0)$	$(\top,+)$	$(-0,-0)$	$(\top,0+)$	$(\top,\top)$

After mechanical inspection of all cases, we derive the above table.

We can derive similar tables for $\leq^\sharp$, $=^\sharp$, $\neq^\sharp$, $>^\sharp$, and $\geq^\sharp$.

Enhanced Approximate Transfer Mapping

$$\overline{h_{x:=e}}(\overline{v}) \;=\; \lambda y \,.\, \begin{cases} \overline{v}(y) & \text{if } y \neq x \\ \overline{[\![e]\!]}_{\overline{v}} & \text{if } y = x \end{cases} \qquad \overline{h_g}(\overline{v}) \;=\; \begin{cases} \overline{\bot} & \text{if } \overline{v} \not\models g \\ \overline{\theta_g}(\overline{v}) & \text{if } \overline{v} \models g \end{cases}$$

$g = x \bowtie x$

$$\overline{\theta_g}(\overline{v}) \;=\; \lambda y \,.\, \begin{cases} \overline{v}(y) & \text{if } \bowtie \in \{=, \leq, \geq\} \\ \overline{\bot} & \text{if } \bowtie \in \{\neq, <, >\} \end{cases}$$

$g = x_1 \bowtie x_2$ with $x_1 \neq x_2$

$$\overline{\theta_g}(\overline{v}) \;=\; \lambda y \,.\, \begin{cases} \overline{t_1} & \text{if } y = x_1 \\ \overline{t_2} & \text{if } y = x_2 \\ \overline{v}(y) & \text{otherwise} \end{cases} \qquad \text{where } (\overline{t_1}, \overline{t_2}) = \overline{[\![x_1]\!]}_{\overline{v}} \bowtie^\sharp \overline{[\![x_2]\!]}_{\overline{v}}$$

Enhanced Approximate Transfer Mapping

$$\overline{h_{x:=e}}(\overline{v}) \;=\; \lambda\, y\,.\begin{cases}\overline{v}(y) & \text{if } y \neq x\\ \overline{[\![e]\!]}_{\overline{v}} & \text{if } y = x\end{cases} \qquad \overline{h_g}(\overline{v}) \;=\; \begin{cases}\overline{\bot} & \text{if } \overline{v} \not\models g\\ \overline{\theta_g}(\overline{v}) & \text{if } \overline{v} \models g\end{cases}$$

$g = x \bowtie x$

$$\overline{\theta_g}(\overline{v}) \;=\; \lambda\, y\,.\begin{cases}\overline{v}(y) & \text{if } \bowtie \in \{=, \leq, \geq\}\\ \overline{\bot} & \text{if } \bowtie \in \{\neq, <, >\}\end{cases}$$

$g = x_1 \bowtie x_2 \quad$ with $x_1 \neq x_2$

$$\overline{\theta_g}(\overline{v}) \;=\; \lambda\, y\,.\begin{cases}\overline{t_1} & \text{if } y = x_1\\ \overline{t_2} & \text{if } y = x_2\\ \overline{v}(y) & \text{otherwise}\end{cases} \qquad \text{where } (\overline{t_1}, \overline{t_2}) = \overline{[\![x_1]\!]}_{\overline{v}} \bowtie^\sharp \overline{[\![x_2]\!]}_{\overline{v}}$$

Enhanced Approximate Transfer Mapping

$$\overline{h_{x:=e}}(\overline{v}) \;=\; \lambda\, y\,.\begin{cases}\overline{v}(y) & \text{if } y \neq x\\ \overline{[\![e]\!]}_{\overline{v}} & \text{if } y = x\end{cases} \qquad \overline{h_g}(\overline{v}) \;=\; \begin{cases}\overline{\bot} & \text{if } \overline{v} \not\models g\\ \overline{\theta_g}(\overline{v}) & \text{if } \overline{v} \models g\end{cases}$$

$g = x \bowtie e$ with e not reduced to a variable

$$\overline{\theta_g}(\overline{v}) \;=\; \lambda\, y\,.\begin{cases}\overline{t} & \text{if } y = x\\ \overline{v}(y) & \text{otherwise}\end{cases} \qquad \text{where} \quad (\overline{t}, \_) \;=\; \overline{[\![x]\!]}_{\overline{v}} \bowtie^{\sharp} \overline{[\![e]\!]}_{\overline{v}}$$

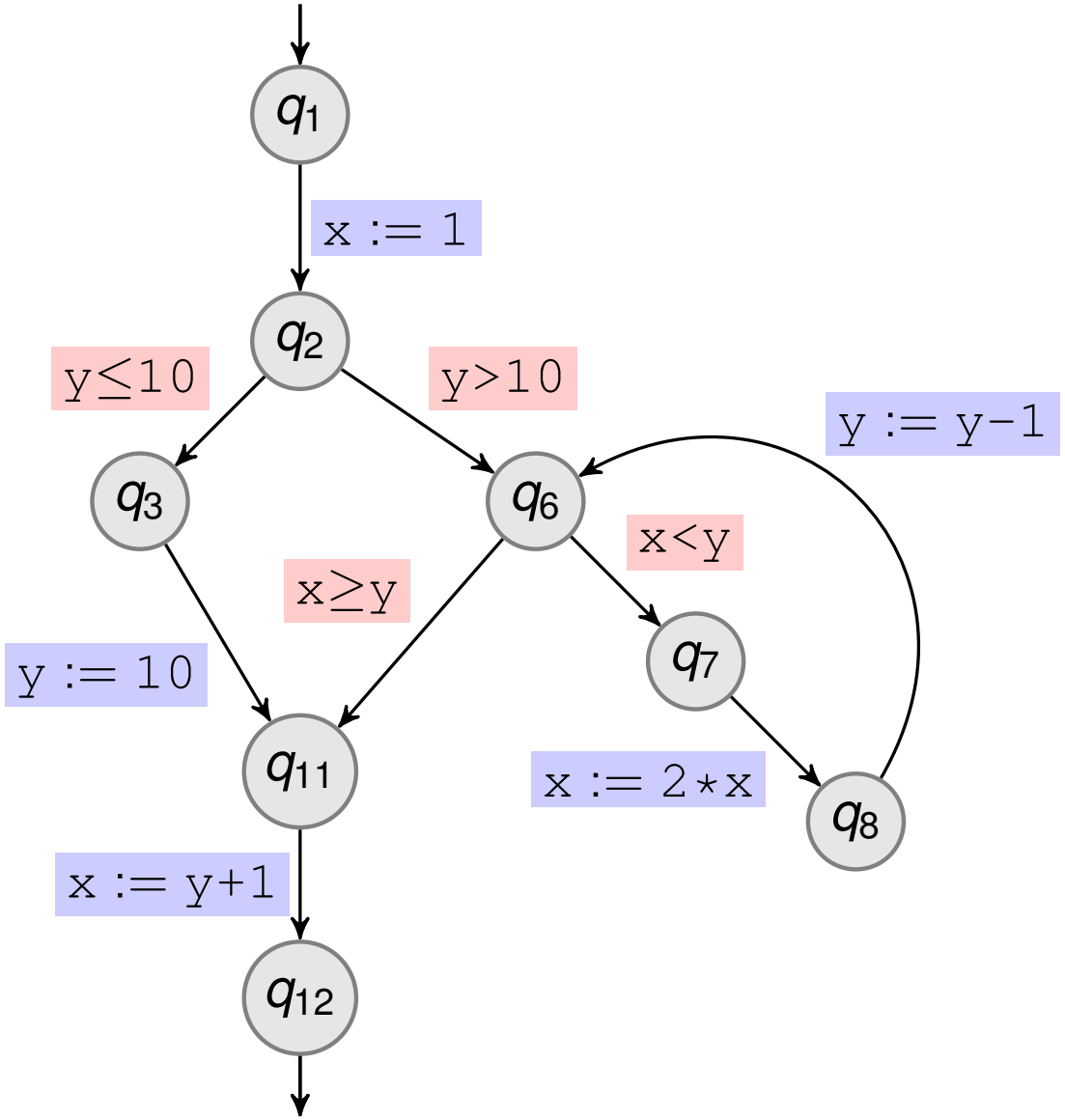
$g = e \bowtie x$ with e not reduced to a variable

$$\overline{\theta_g}(\overline{v}) \;=\; \lambda\, y\,.\begin{cases}\overline{t} & \text{if } y = x\\ \overline{v}(y) & \text{otherwise}\end{cases} \qquad \text{where} \quad (\_, \overline{t}) \;=\; \overline{[\![e]\!]}_{\overline{v}} \bowtie^{\sharp} \overline{[\![x]\!]}_{\overline{v}}$$

$g = e_1 \bowtie e_2$ with e_1, e_2 not reduced to a variable

$$\overline{\theta_g}(\overline{v}) \;=\; \overline{v}$$

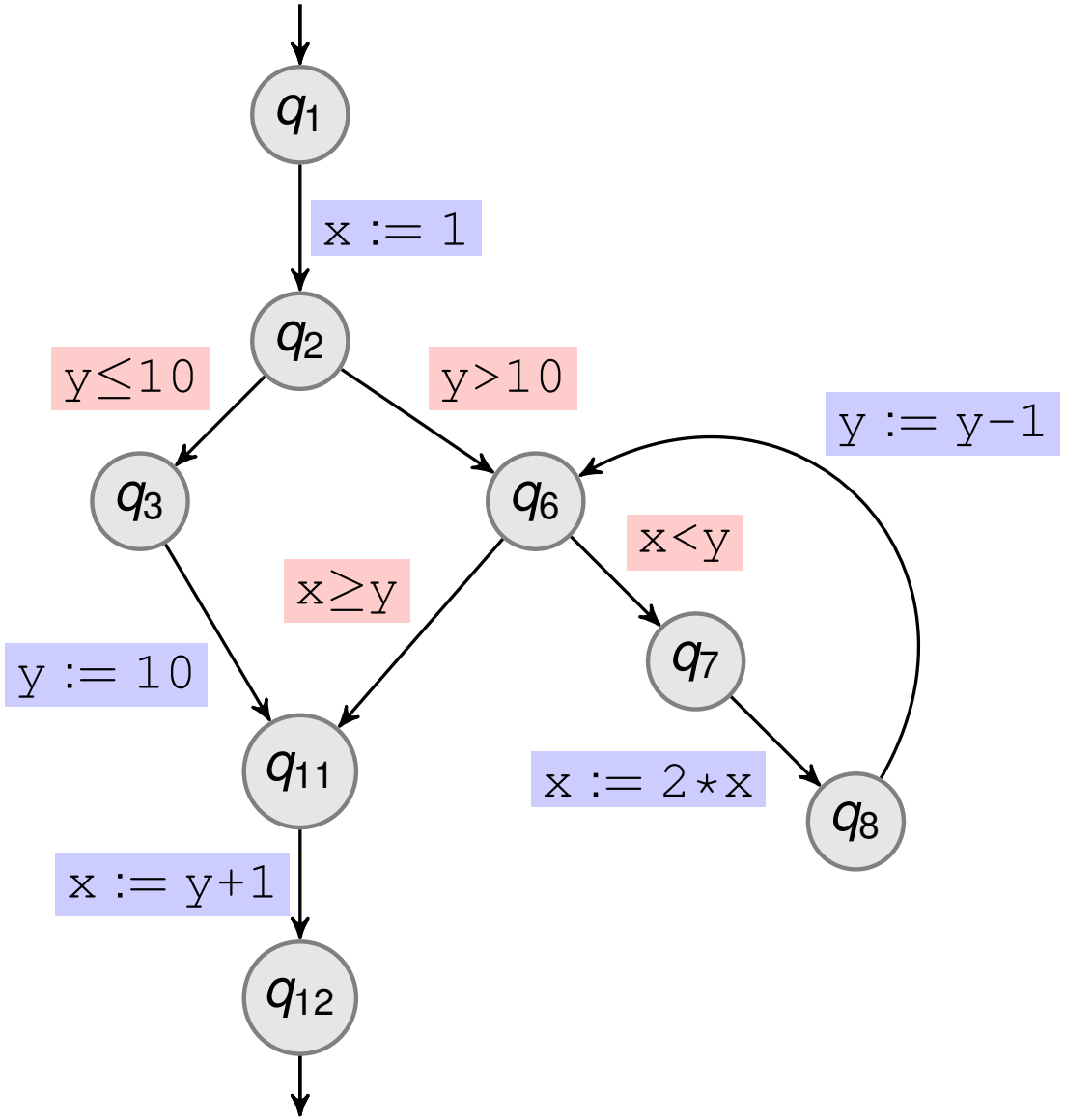
Forward Sign Analysis on Example with Enhanced $\overline{h_{\mathrm{op}}}$



Goal

Show that $\mathtt{x} > 0$ at q_{12}

Forward Sign Analysis on Example with Enhanced $\overline{h_{\text{op}}}$

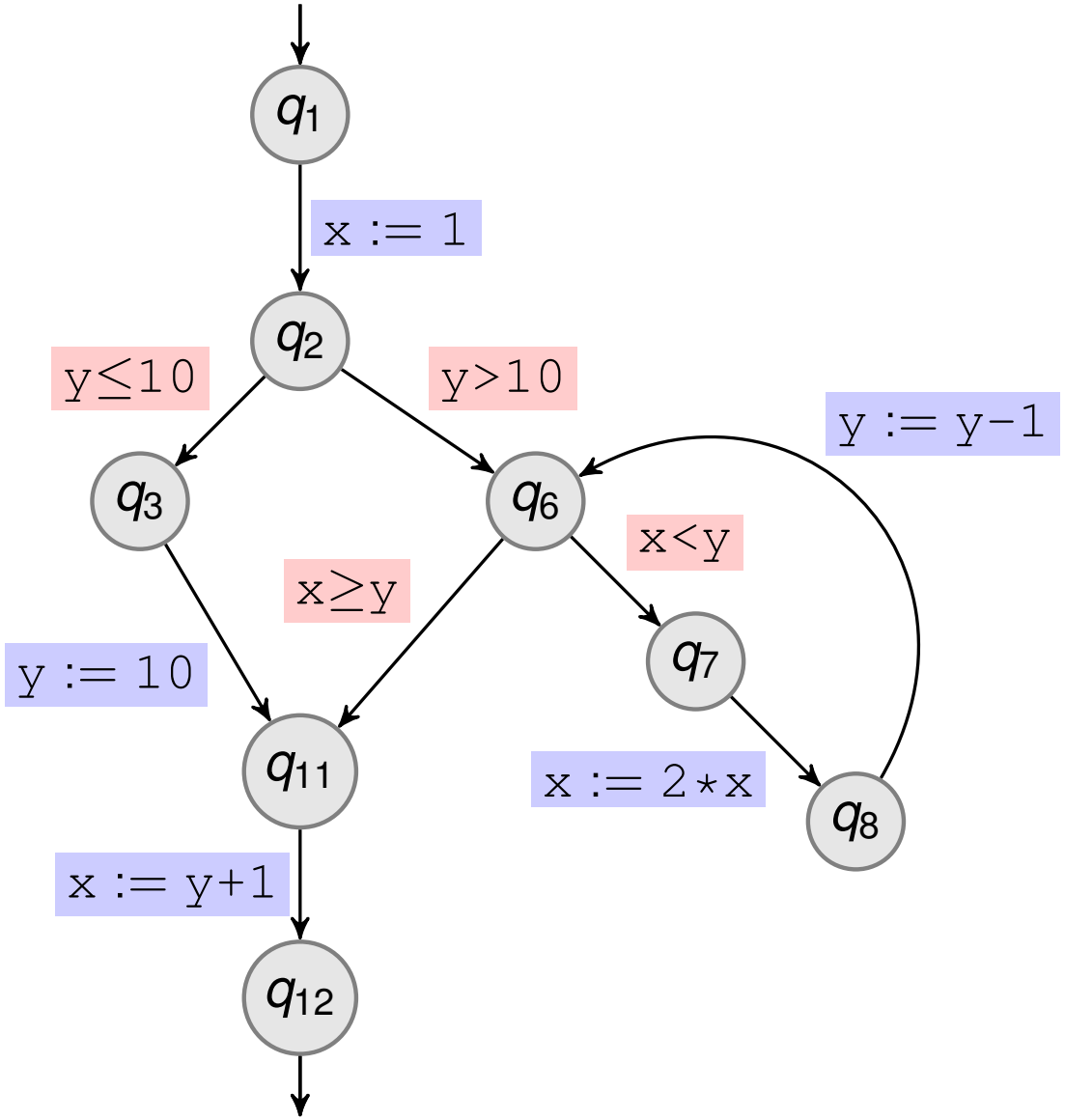


Goal

Show that $x > 0$ at q_{12}

	x	y
q_1	$\top$	$\top$
q_2	$\bot$	$\bot$
q_3	$\bot$	$\bot$
q_6	$\bot$	$\bot$
q_7	$\bot$	$\bot$
q_8	$\bot$	$\bot$
q_{11}	$\bot$	$\bot$
q_{12}	$\bot$	$\bot$

Forward Sign Analysis on Example with Enhanced $\overline{h_{\mathrm{op}}}$

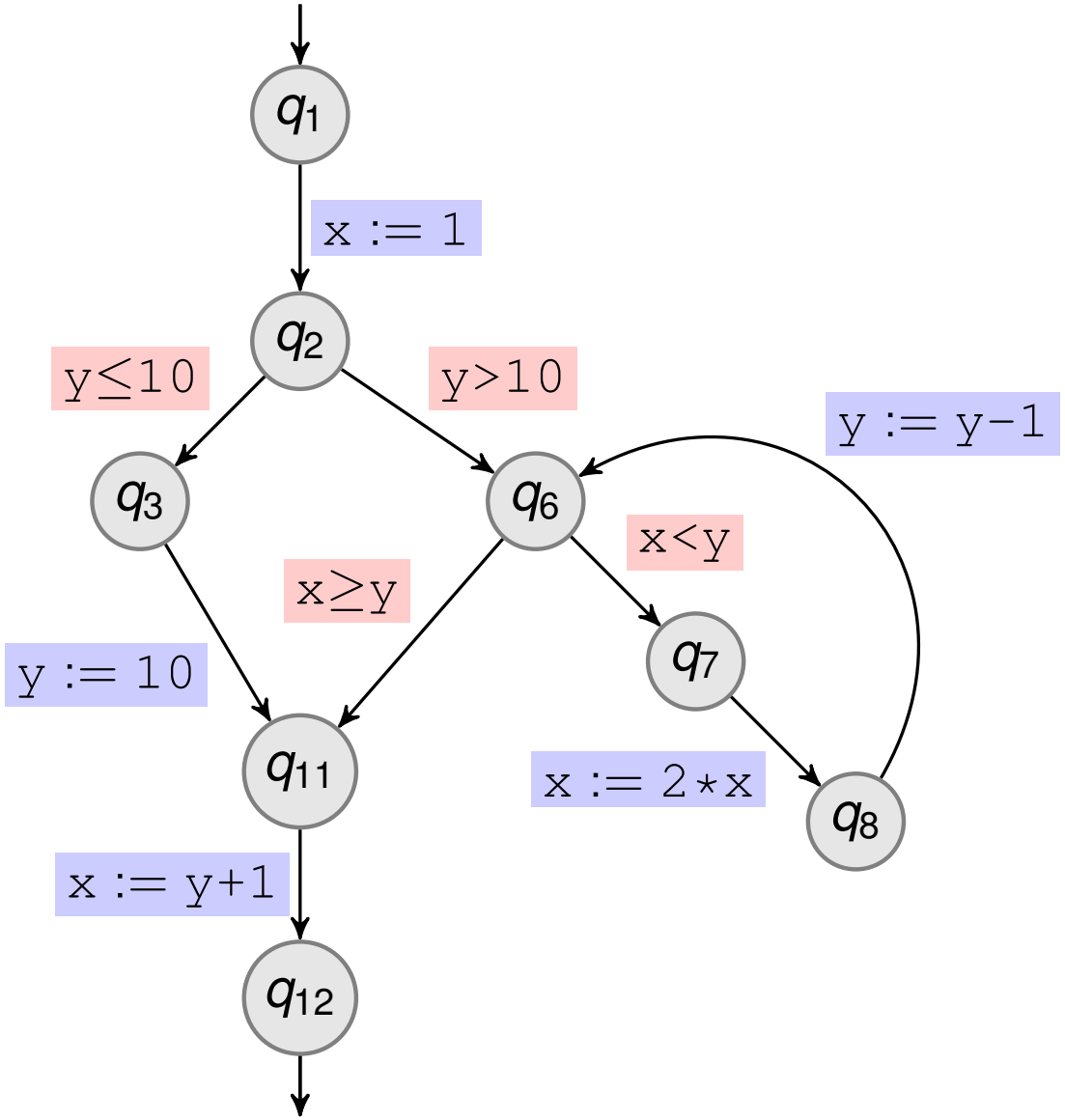


Goal

Show that $x > 0$ at q_{12}

	x	y
q_1	$\top$	$\top$
q_2	$+$	$\top$
q_3	$\bot$	$\bot$
q_6	$\bot$	$\bot$
q_7	$\bot$	$\bot$
q_8	$\bot$	$\bot$
q_{11}	$\bot$	$\bot$
q_{12}	$\bot$	$\bot$

Forward Sign Analysis on Example with Enhanced $\overline{h_{\text{op}}}$

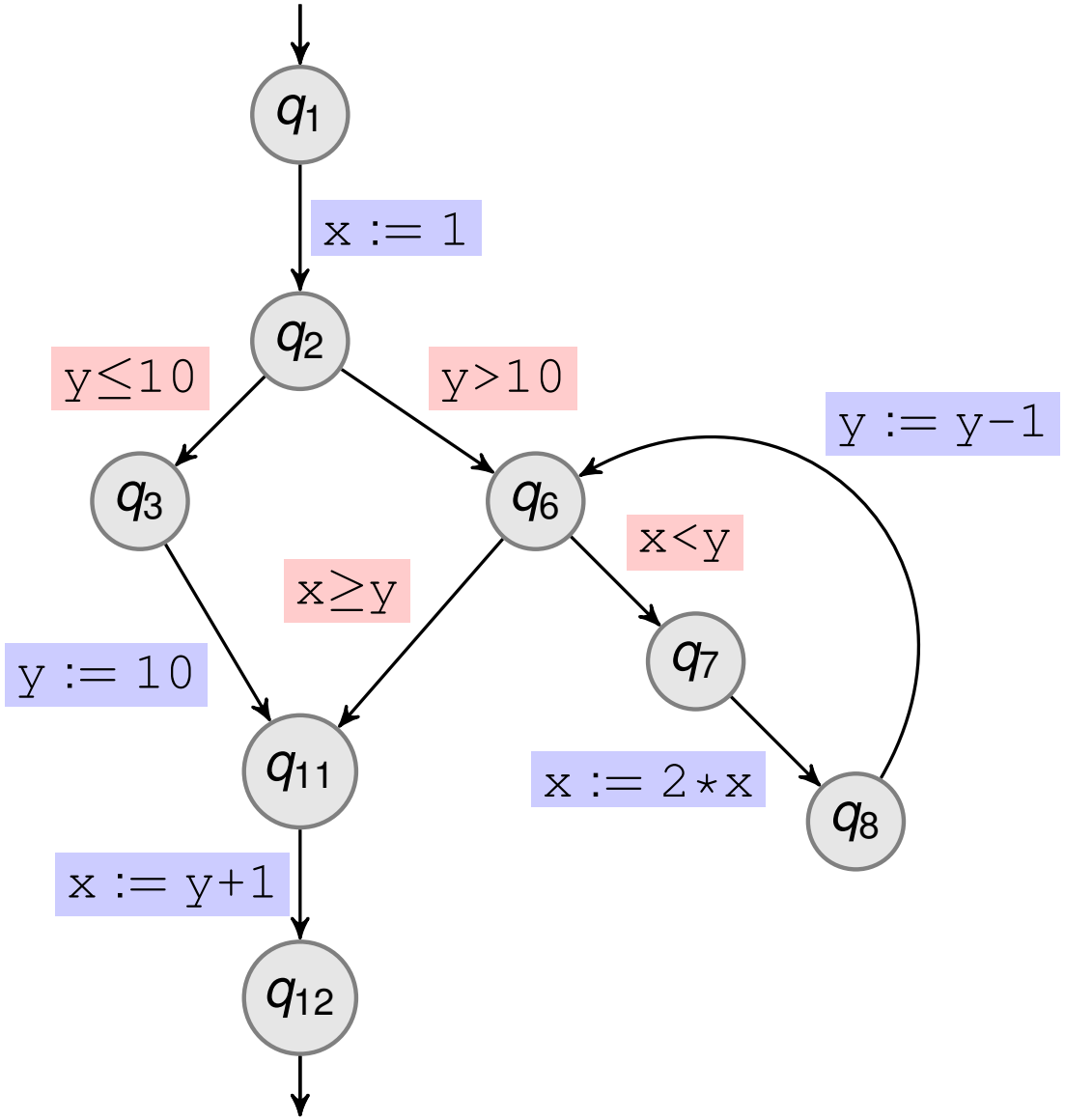


Goal

Show that $x > 0$ at q_{12}

	x	y
q_1	$\top$	$\top$
q_2	$+$	$\top$
q_3	$\bot$	$\bot$
q_6	$\bot$	$\bot$
q_7	$\bot$	$\bot$
q_8	$\bot$	$\bot$
q_{11}	$\bot$	$\bot$
q_{12}	$\bot$	$\bot$

Forward Sign Analysis on Example with Enhanced $\overline{h_{\text{op}}}$

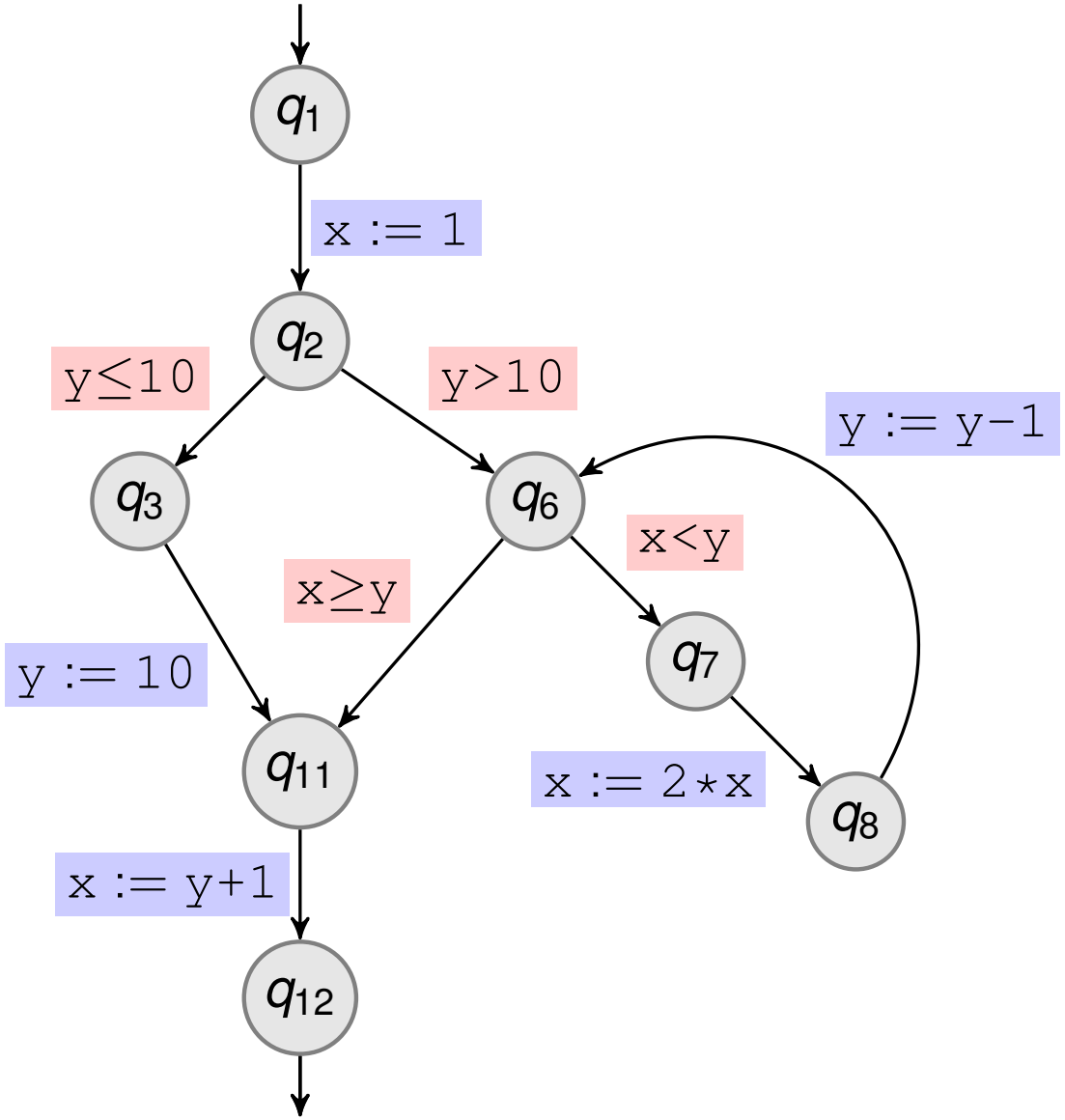


Goal

Show that $x > 0$ at q_{12}

	x	y
q_1	$\top$	$\top$
q_2	$+$	$\top$
q_3	$+$	$\top$
q_6	$\bot$	$\bot$
q_7	$\bot$	$\bot$
q_8	$\bot$	$\bot$
q_{11}	$\bot$	$\bot$
q_{12}	$\bot$	$\bot$

Forward Sign Analysis on Example with Enhanced $\overline{h_{\mathrm{op}}}$

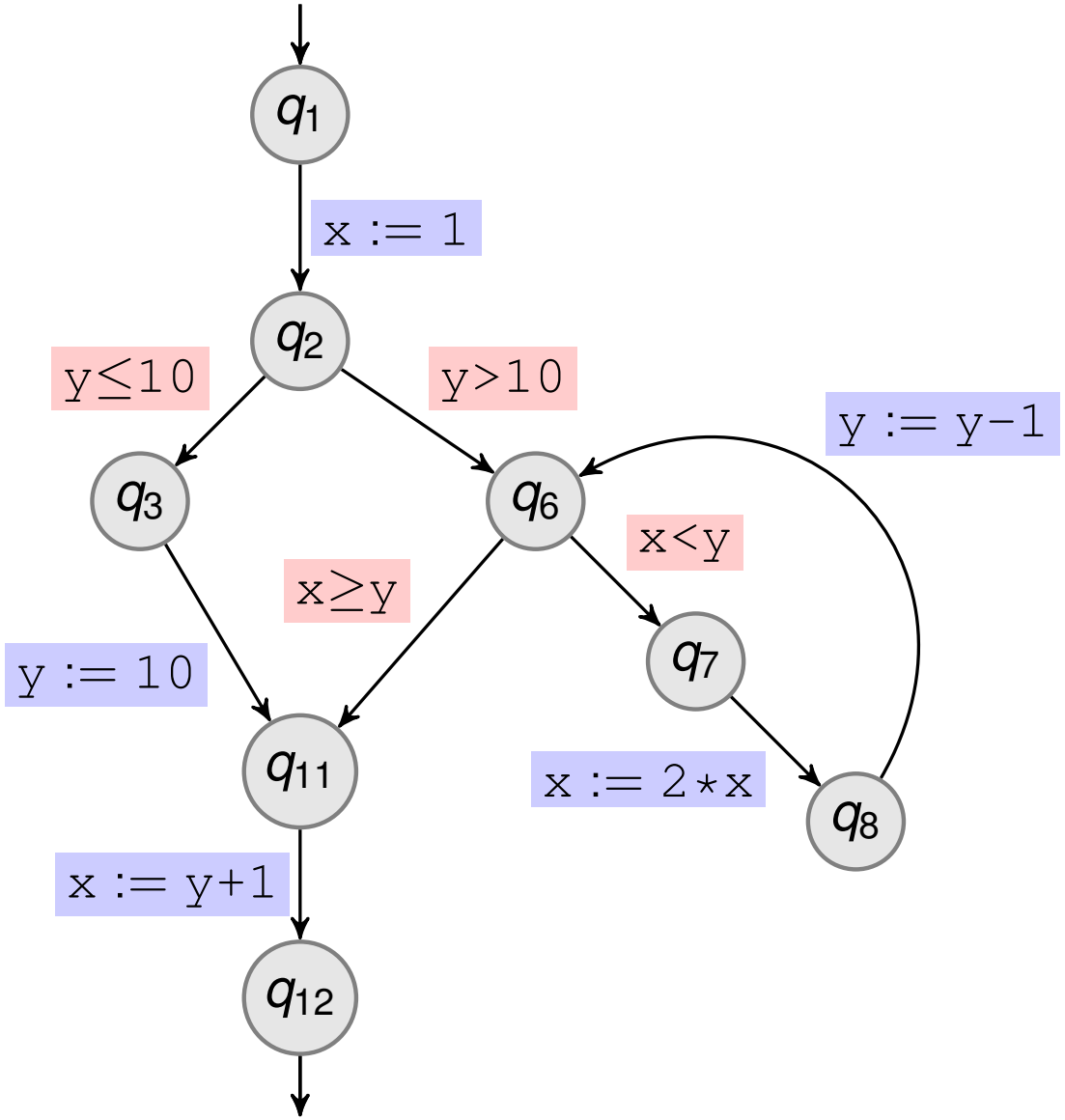


Goal

Show that $\mathtt{x} > 0$ at q_{12}

	x	y
q_1	$\top$	$\top$
q_2	$+$	$\top$
q_3	$+$	$\top$
q_6	$\bot$	$\bot$
q_7	$\bot$	$\bot$
q_8	$\bot$	$\bot$
q_{11}	$\bot$	$\bot$
q_{12}	$\bot$	$\bot$

Forward Sign Analysis on Example with Enhanced $\overline{h_{\text{op}}}$

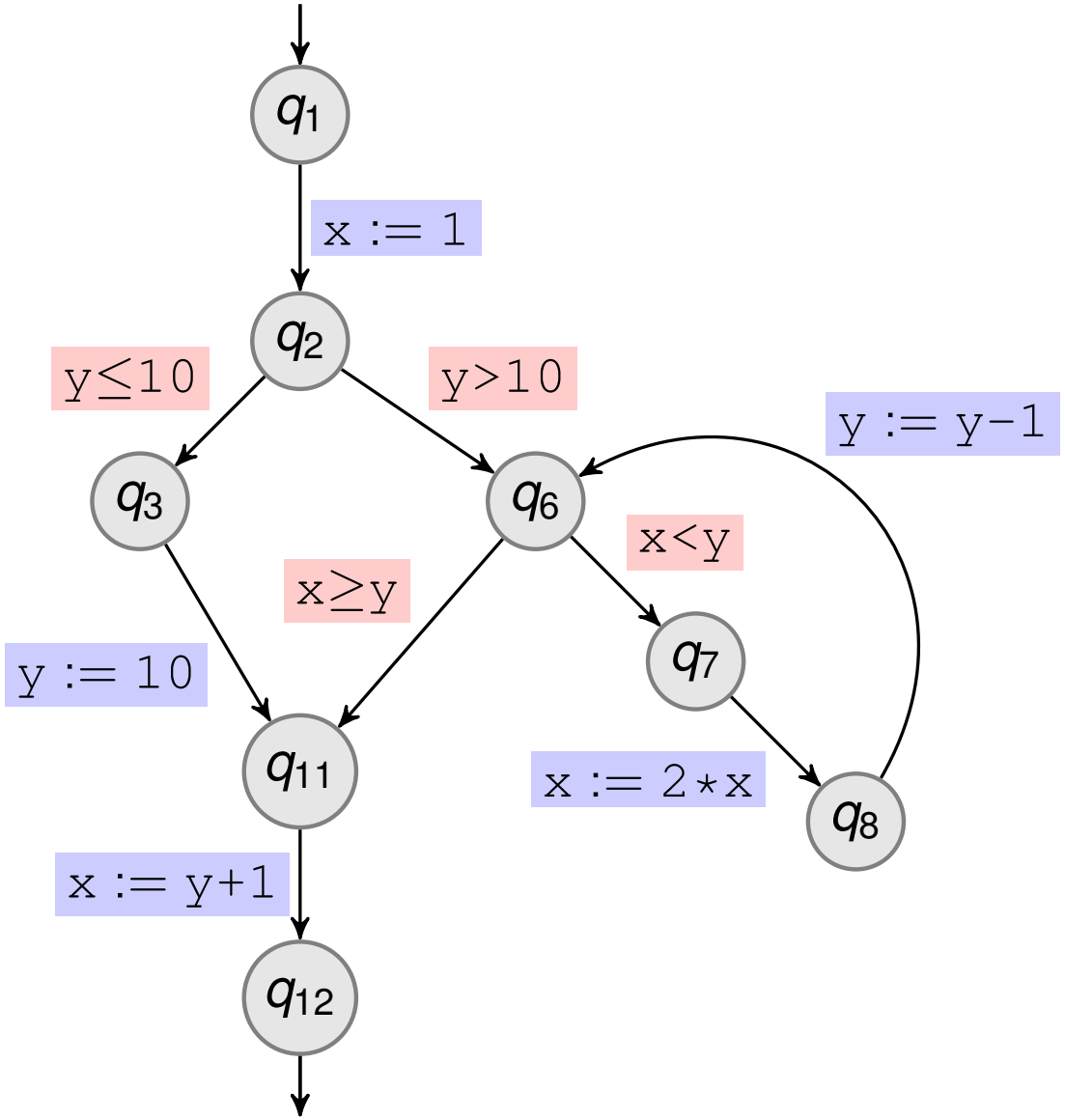


Goal

Show that $x > 0$ at q_{12}

	x	y
q_1	$\top$	$\top$
q_2	$+$	$\top$
q_3	$+$	$\top$
q_6	$\bot$	$\bot$
q_7	$\bot$	$\bot$
q_8	$\bot$	$\bot$
q_{11}	$+$	$+$
q_{12}	$\bot$	$\bot$

Forward Sign Analysis on Example with Enhanced $\overline{h_{\text{op}}}$

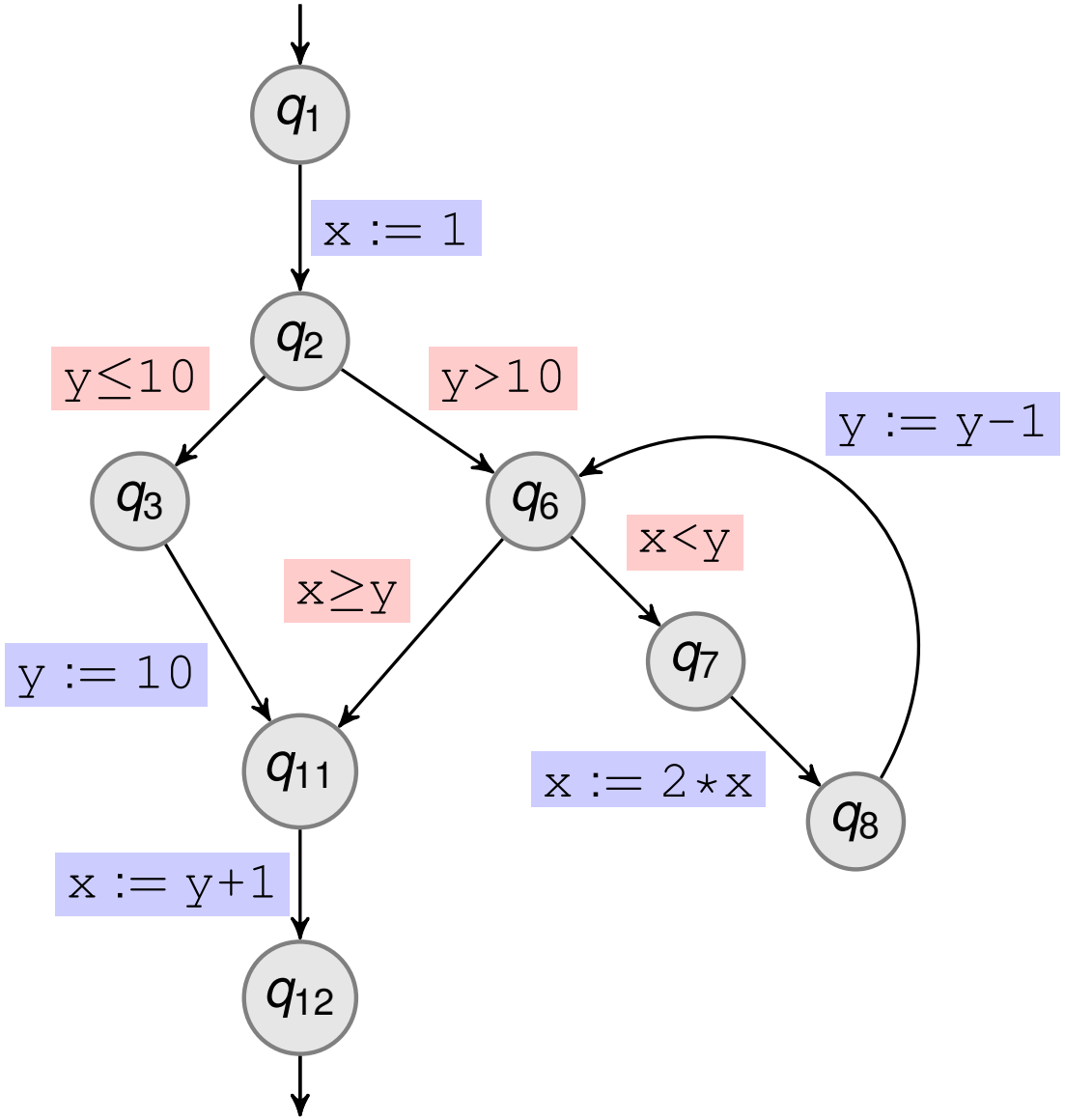


Goal

Show that $x > 0$ at q_{12}

	x	y
q_1	$\top$	$\top$
q_2	$+$	$\top$
q_3	$+$	$\top$
q_6	$\bot$	$\bot$
q_7	$\bot$	$\bot$
q_8	$\bot$	$\bot$
q_{11}	$+$	$+$
q_{12}	$\bot$	$\bot$

Forward Sign Analysis on Example with Enhanced $\overline{h_{\text{op}}}$

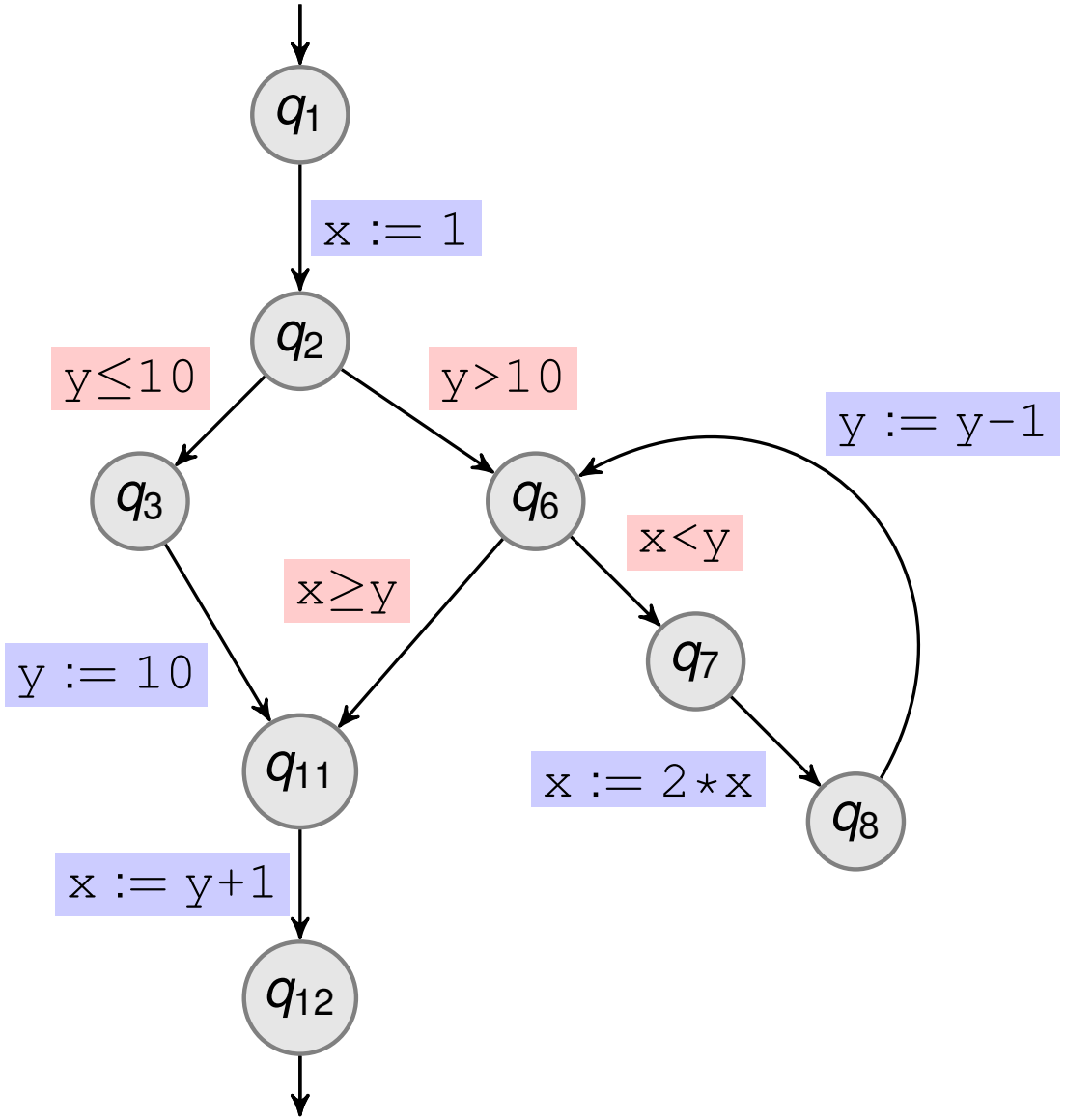


Goal

Show that $x > 0$ at q_{12}

	x	y
q_1	$\top$	$\top$
q_2	$+$	$\top$
q_3	$+$	$\top$
q_6	$\bot$	$\bot$
q_7	$\bot$	$\bot$
q_8	$\bot$	$\bot$
q_{11}	$+$	$+$
q_{12}	$+$	$+$

Forward Sign Analysis on Example with Enhanced $\overline{h_{\text{op}}}$

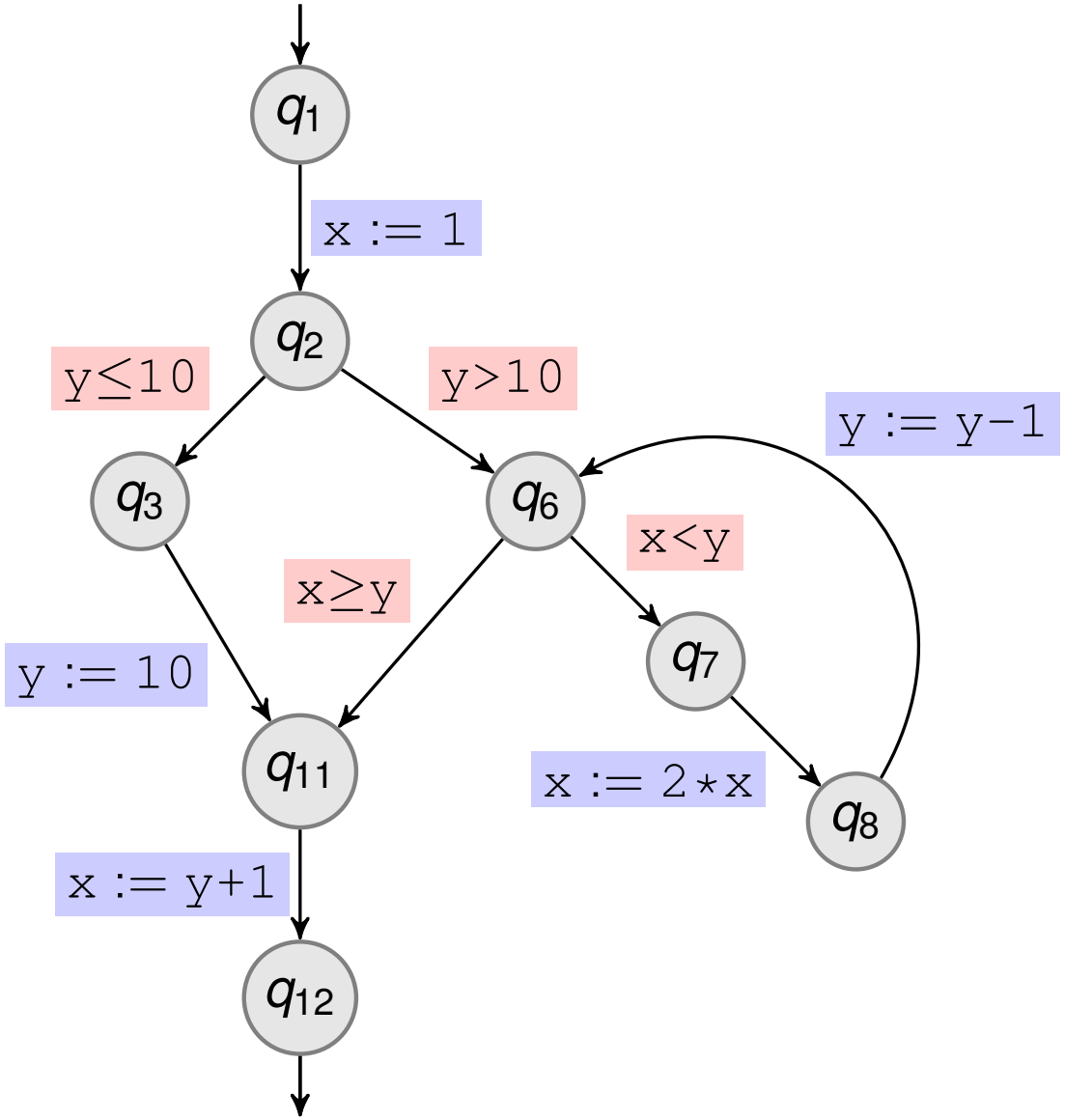


Goal

Show that $x > 0$ at q_{12}

	x	y
q_1	$\top$	$\top$
q_2	$+$	$\top$
q_3	$+$	$\top$
q_6	$\bot$	$\bot$
q_7	$\bot$	$\bot$
q_8	$\bot$	$\bot$
q_{11}	$+$	$+$
q_{12}	$+$	$+$

Forward Sign Analysis on Example with Enhanced $\overline{h_{\text{op}}}$

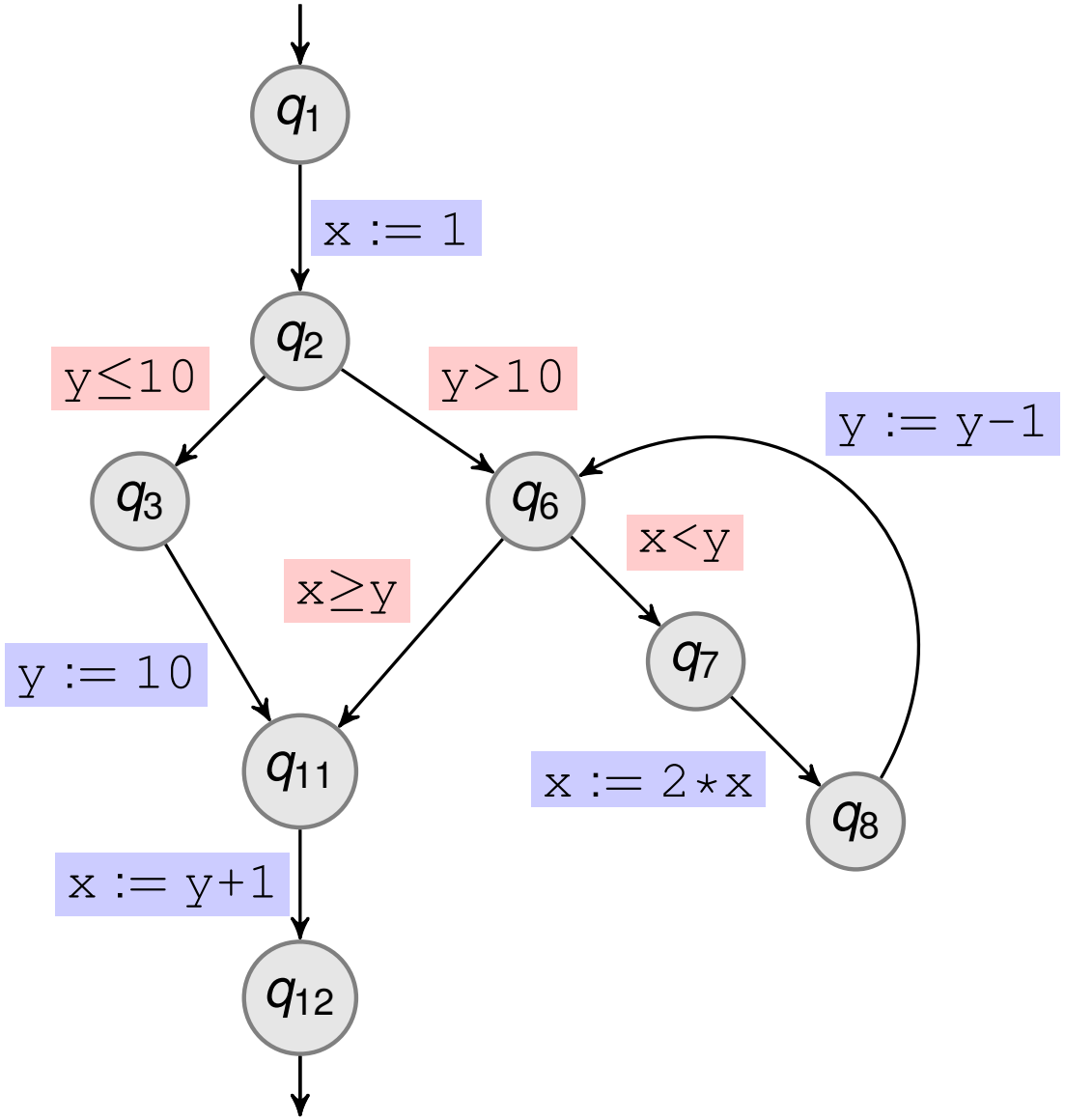


Goal

Show that $x > 0$ at q_{12}

	x	y
q_1	$\top$	$\top$
q_2	$+$	$\top$
q_3	$+$	$\top$
q_6	$+$	$+$
q_7	$\bot$	$\bot$
q_8	$\bot$	$\bot$
q_{11}	$+$	$+$
q_{12}	$+$	$+$

Forward Sign Analysis on Example with Enhanced $\overline{h_{\text{op}}}$

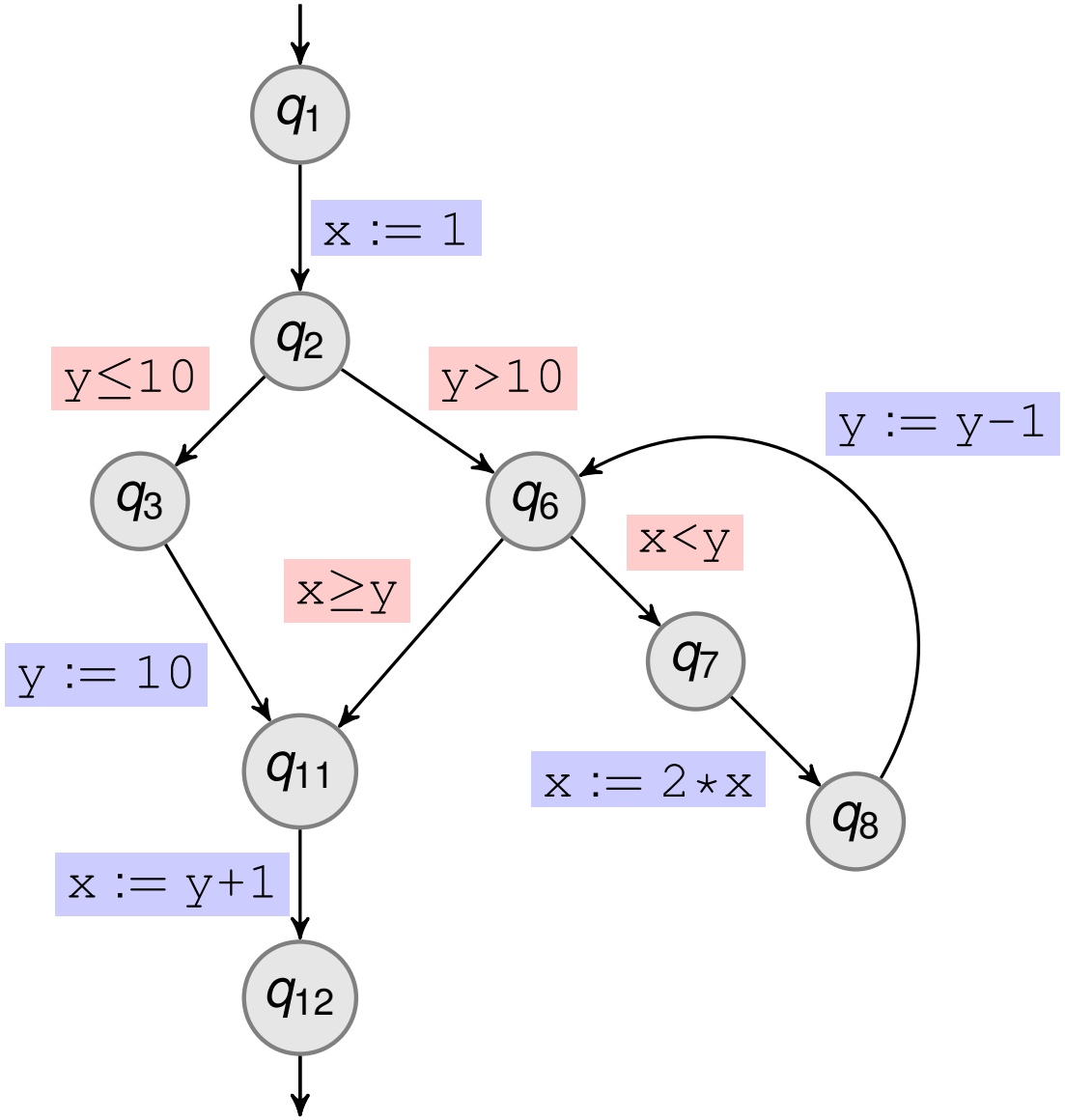


Goal

Show that $x > 0$ at q_{12}

	x	y
q_1	$\top$	$\top$
q_2	$+$	$\top$
q_3	$+$	$\top$
q_6	$+$	$+$
q_7	$\bot$	$\bot$
q_8	$\bot$	$\bot$
q_{11}	$+$	$+$
q_{12}	$+$	$+$

Forward Sign Analysis on Example with Enhanced $\overline{h_{\text{op}}}$

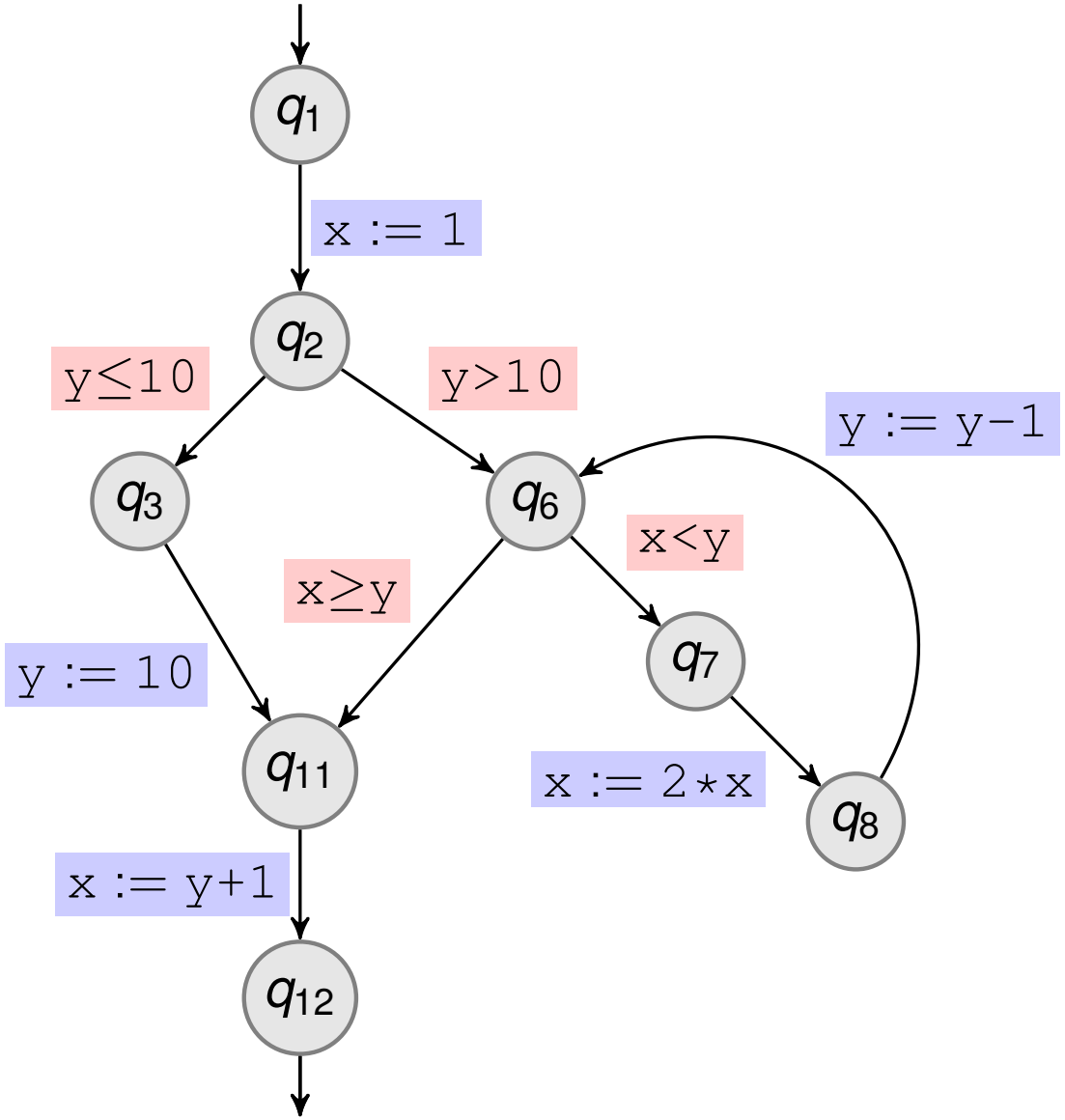


Goal

Show that $x > 0$ at q_{12}

	x	y
q_1	$\top$	$\top$
q_2	$+$	$\top$
q_3	$+$	$\top$
q_6	$+$	$+$
q_7	$+$	$+$
q_8	$\bot$	$\bot$
q_{11}	$+$	$+$
q_{12}	$+$	$+$

Forward Sign Analysis on Example with Enhanced $\overline{h_{\mathrm{op}}}$

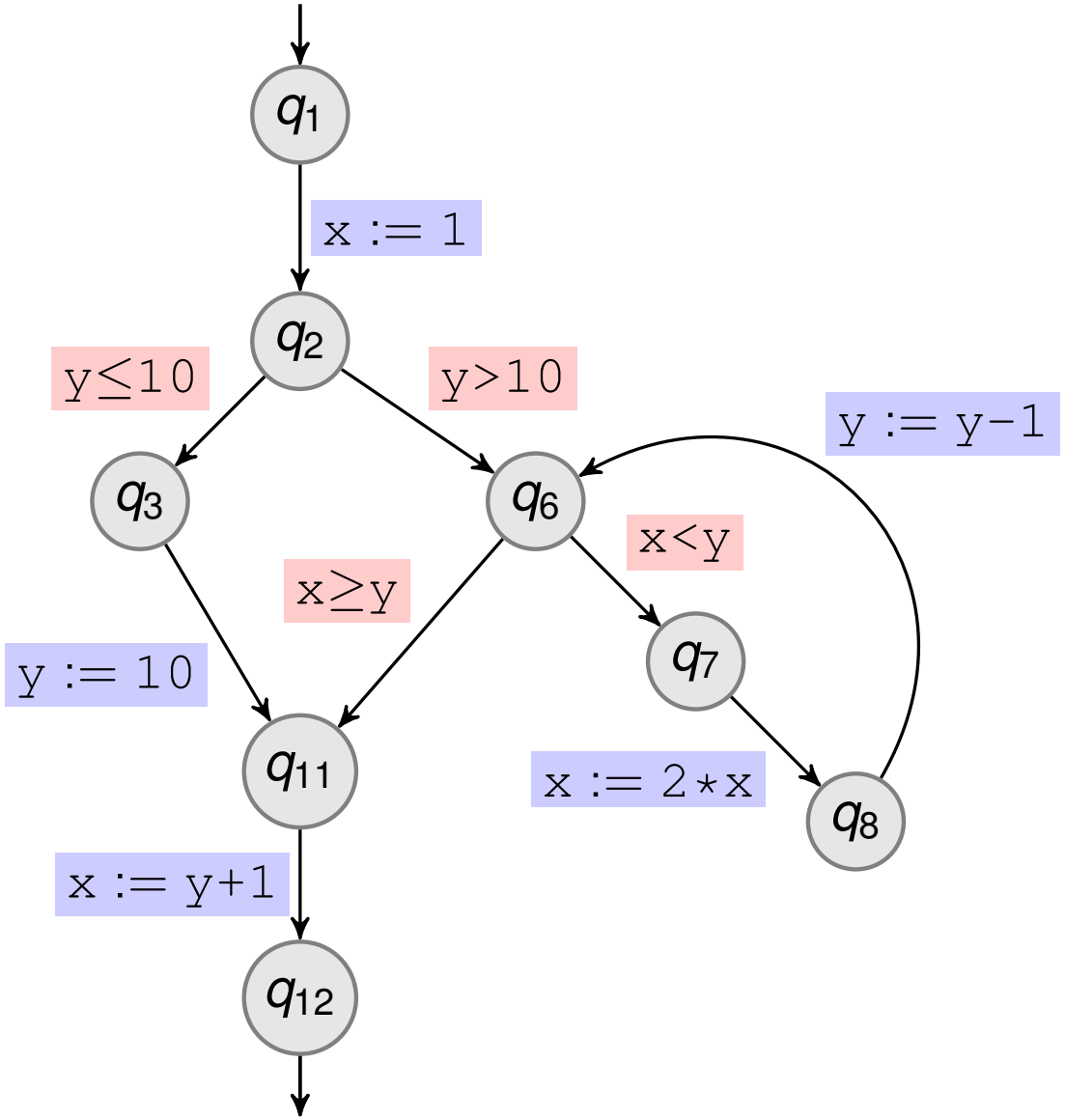


Goal

Show that $x > 0$ at q_{12}

	x	y
q_1	$\top$	$\top$
q_2	$+$	$\top$
q_3	$+$	$\top$
q_6	$+$	$+$
q_7	$+$	$+$
q_8	$\bot$	$\bot$
q_{11}	$+$	$+$
q_{12}	$+$	$+$

Forward Sign Analysis on Example with Enhanced $\overline{h_{\mathrm{op}}}$

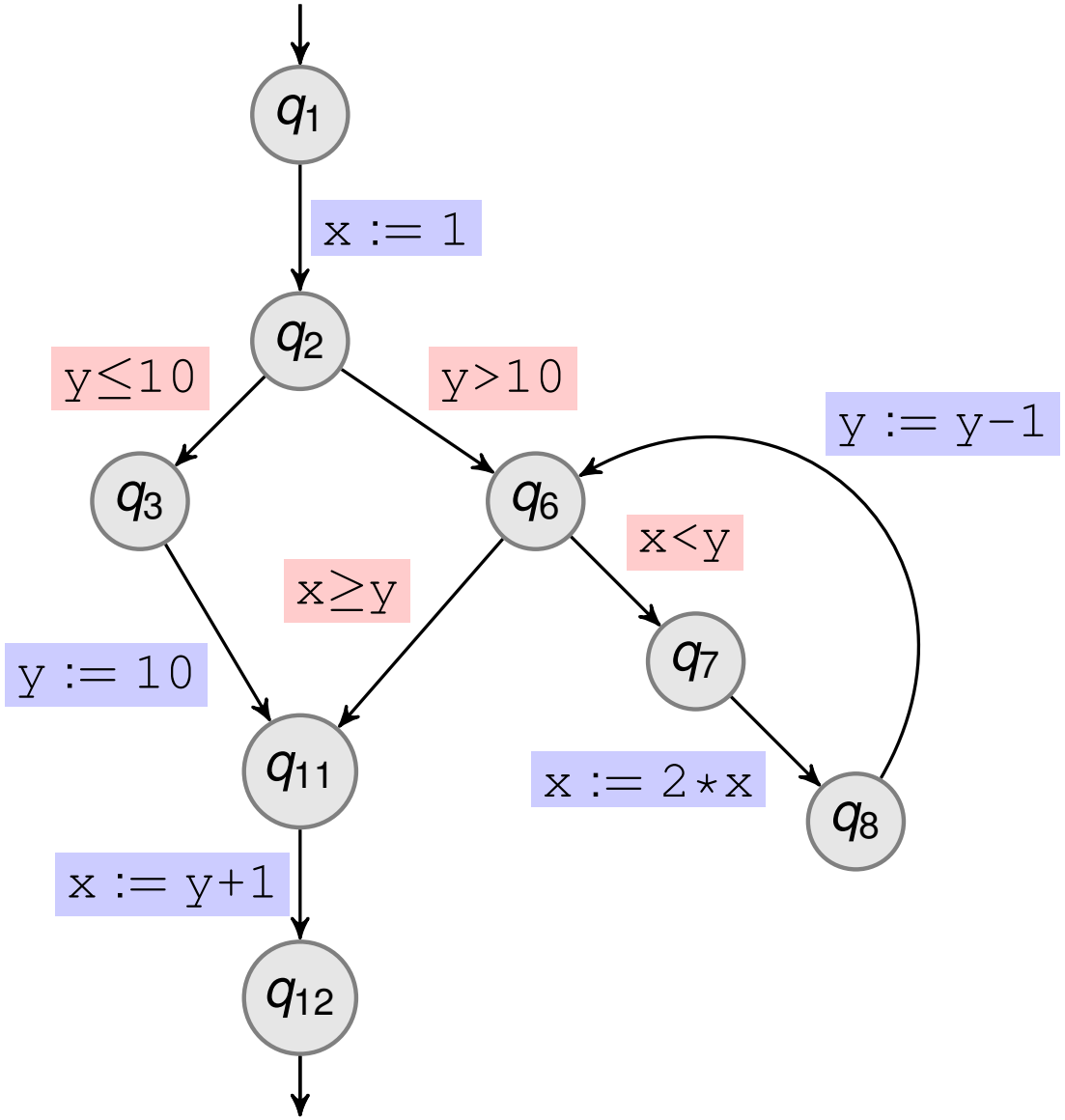


Goal

Show that $x > 0$ at q_{12}

	x	y
q_1	$\top$	$\top$
q_2	$+$	$\top$
q_3	$+$	$\top$
q_6	$+$	$+$
q_7	$+$	$+$
q_8	$+$	$+$
q_{11}	$+$	$+$
q_{12}	$+$	$+$

Forward Sign Analysis on Example with Enhanced $\overline{h_{\mathrm{op}}}$

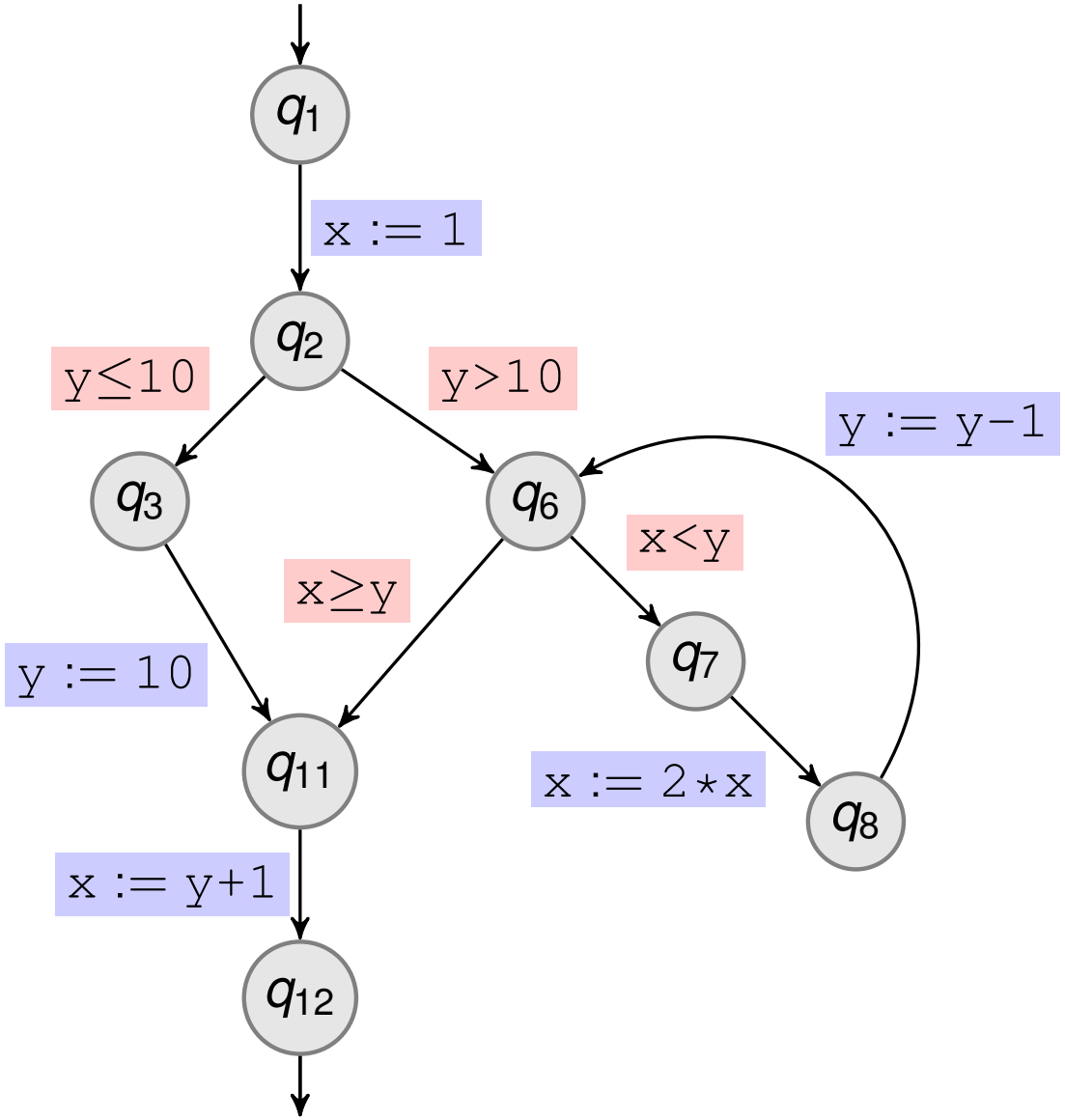


Goal

Show that $\mathtt{x} > 0$ at q_{12}

	x	y
q_1	$\top$	$\top$
q_2	$+$	$\top$
q_3	$+$	$\top$
q_6	$+$	$+$
q_7	$+$	$+$
q_8	$+$	$+$
q_{11}	$+$	$+$
q_{12}	$+$	$+$

Forward Sign Analysis on Example with Enhanced $\overline{h_{\text{op}}}$

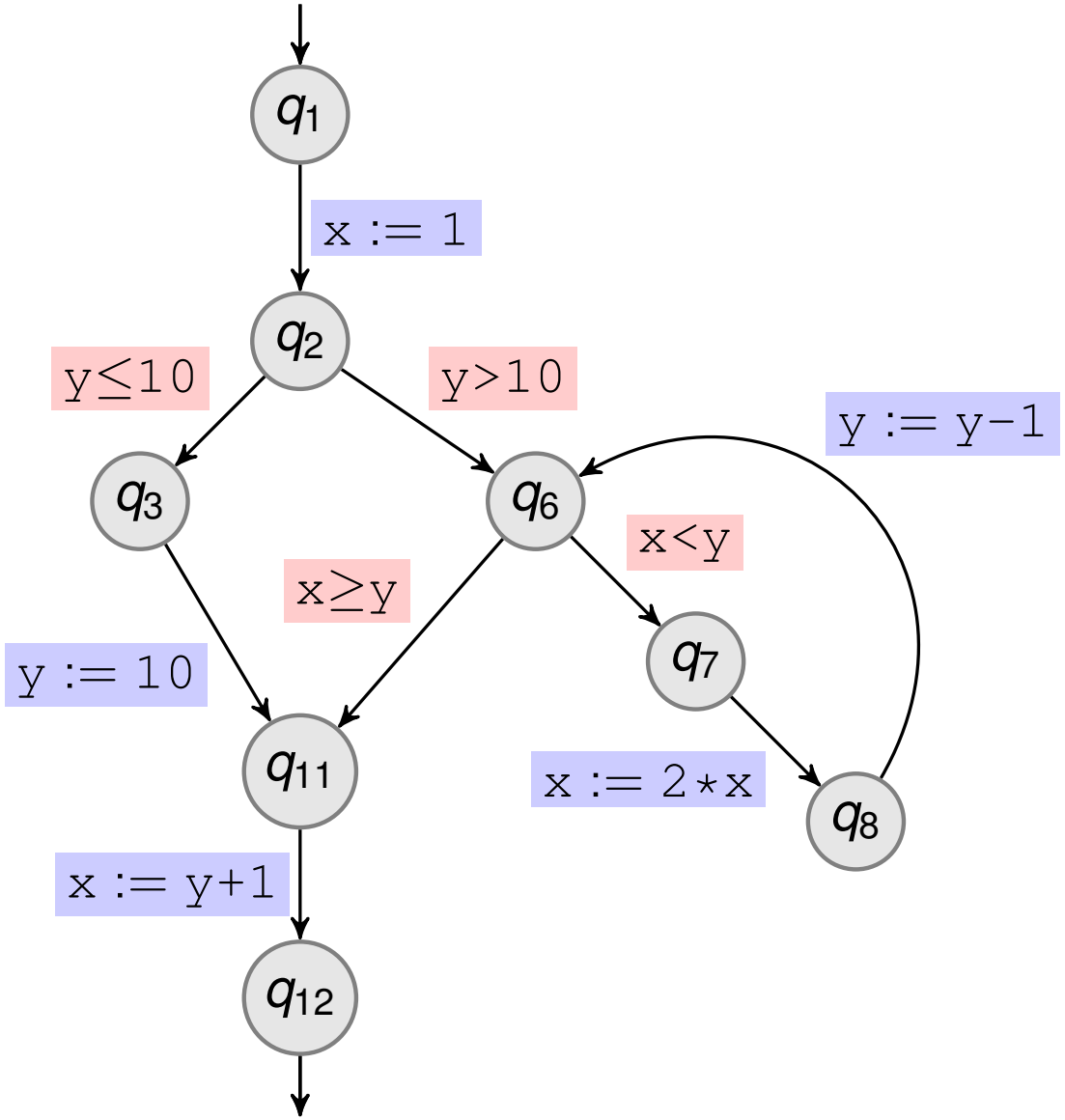


Goal

Show that $x > 0$ at q_{12}

	x	y
q_1	$\top$	$\top$
q_2	$+$	$\top$
q_3	$+$	$\top$
q_6	$+$	$\top$
q_7	$+$	$+$
q_8	$+$	$+$
q_{11}	$+$	$+$
q_{12}	$+$	$+$

Forward Sign Analysis on Example with Enhanced $\overline{h_{\mathrm{op}}}$

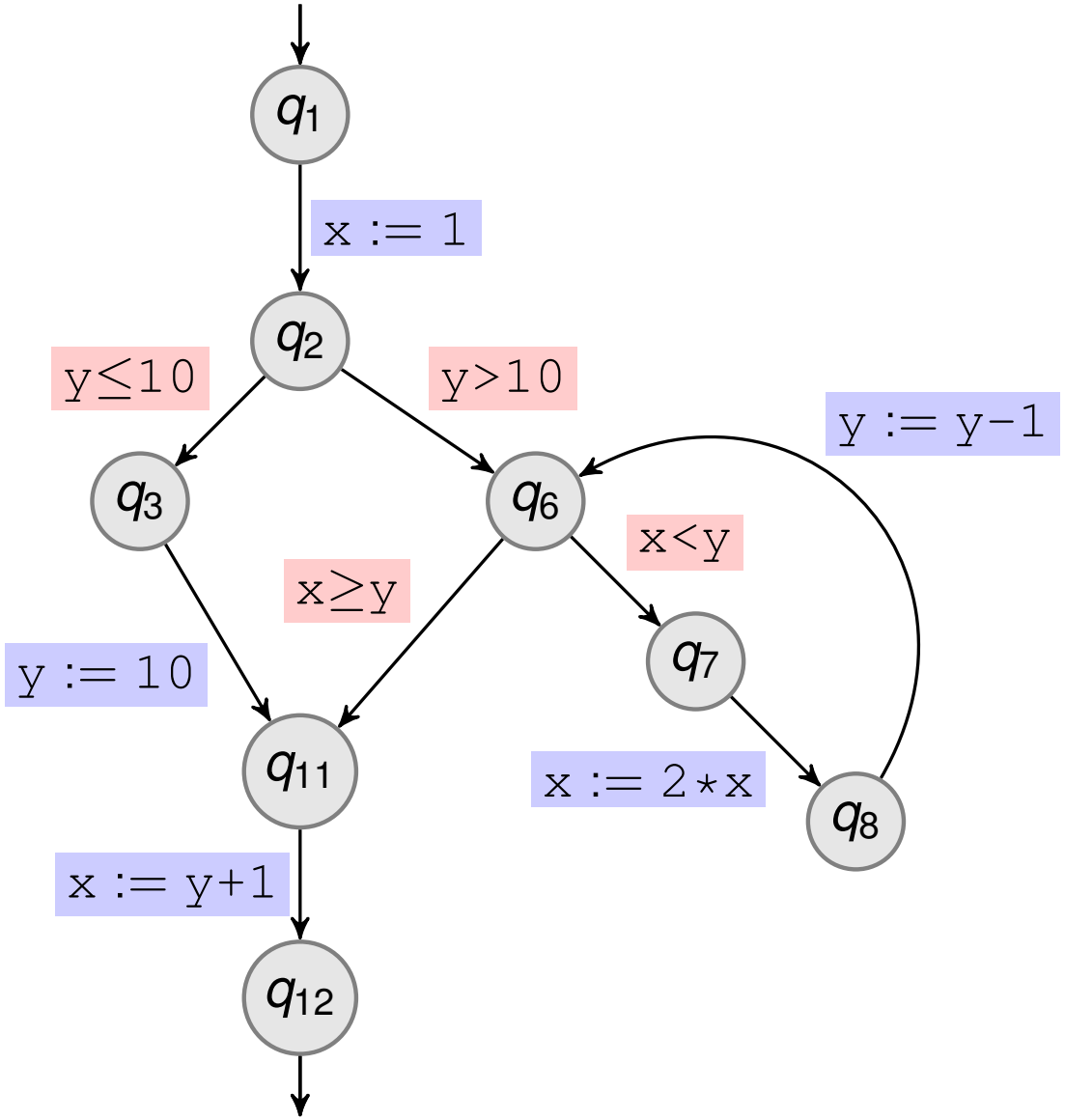


Goal

Show that $\mathtt{x} > 0$ at q_{12}

	x	y
q_1	$\top$	$\top$
q_2	$+$	$\top$
q_3	$+$	$\top$
q_6	$+$	$\top$
q_7	$+$	$+$
q_8	$+$	$+$
q_{11}	$+$	$+$
q_{12}	$+$	$+$

Forward Sign Analysis on Example with Enhanced $\overline{h_{\mathrm{op}}}$

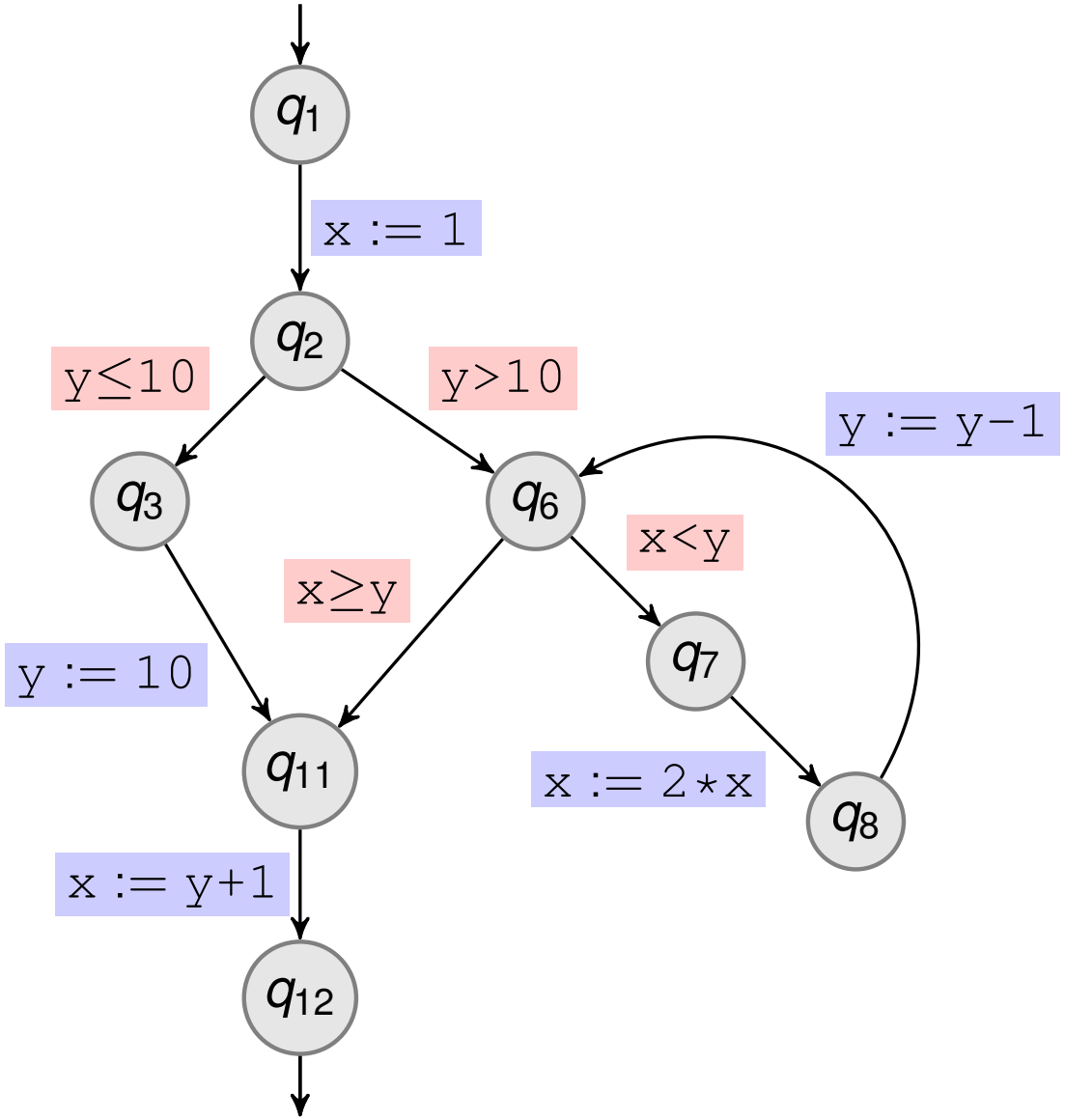


Goal

Show that $x > 0$ at q_{12}

	x	y
q_1	$\top$	$\top$
q_2	$+$	$\top$
q_3	$+$	$\top$
q_6	$+$	$\top$
q_7	$+$	$+$
q_8	$+$	$+$
q_{11}	$+$	$+$
q_{12}	$+$	$+$

Forward Sign Analysis on Example with Enhanced $\overline{h_{\mathrm{op}}}$

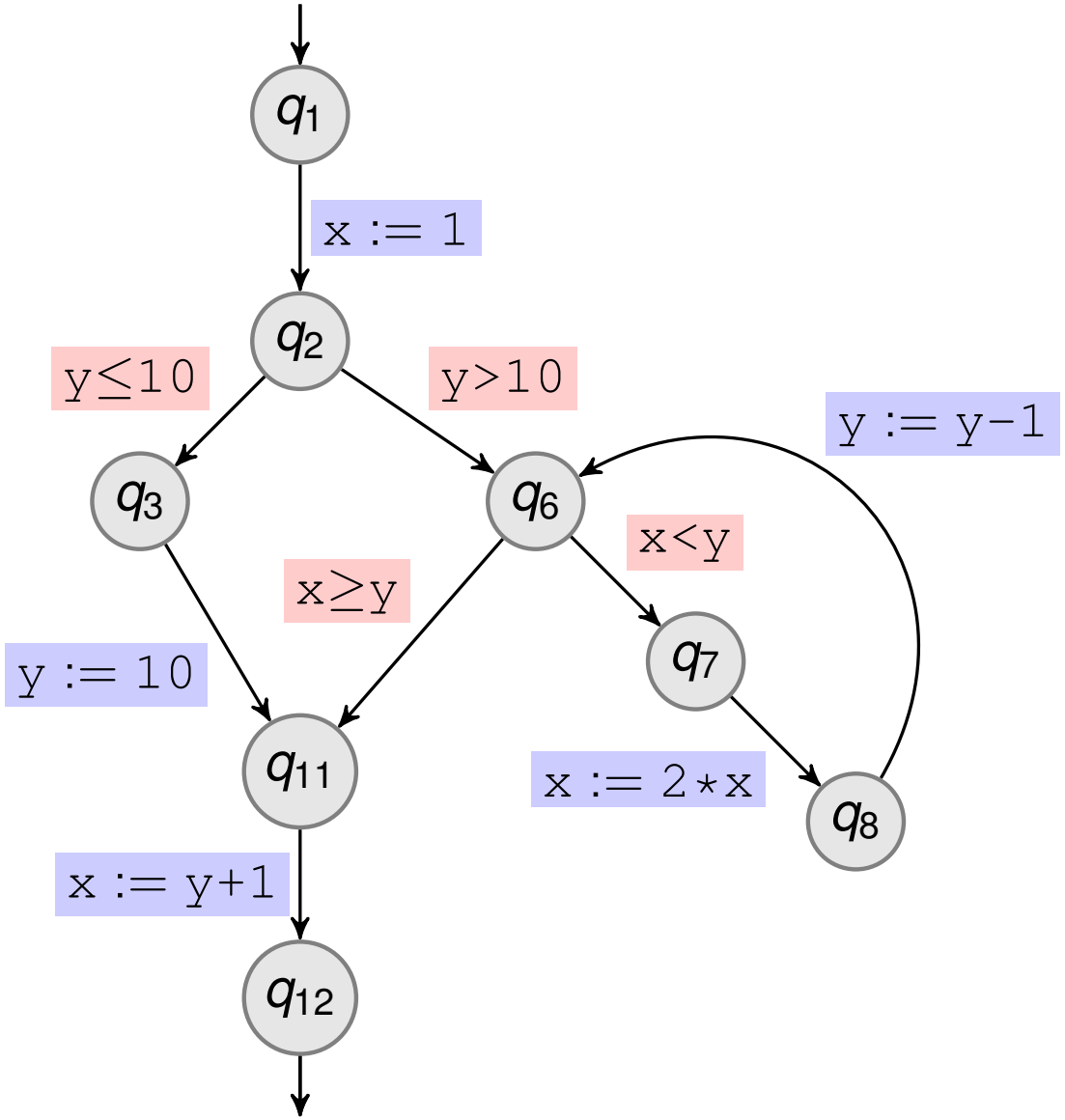


Goal

Show that $\mathtt{x} > 0$ at q_{12}

	x	y
q_1	$\top$	$\top$
q_2	$+$	$\top$
q_3	$+$	$\top$
q_6	$+$	$\top$
q_7	$+$	$+$
q_8	$+$	$+$
q_{11}	$+$	$+$
q_{12}	$+$	$+$

Forward Sign Analysis on Example with Enhanced $\overline{h_{\text{op}}}$

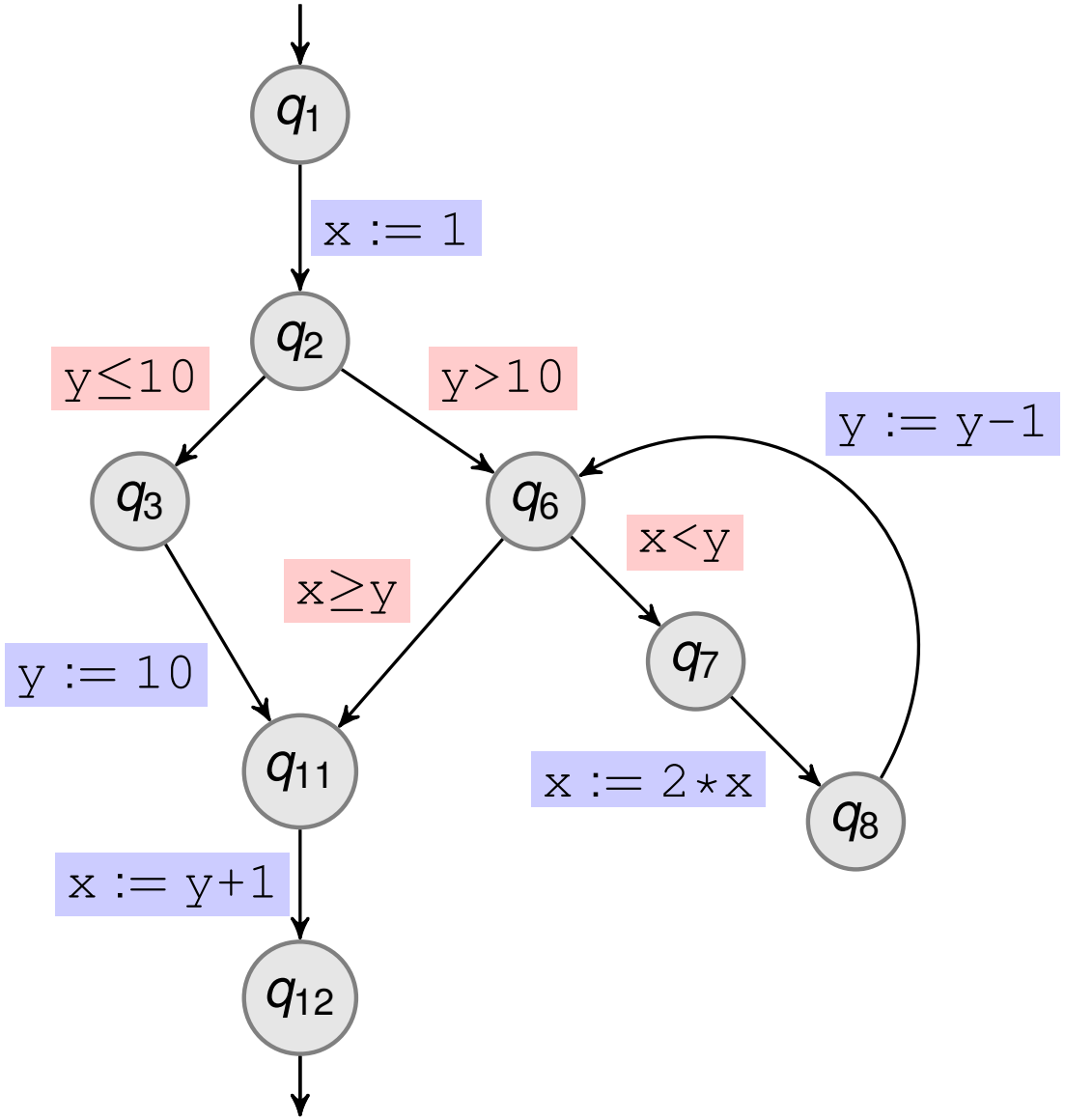


Goal

Show that $x > 0$ at q_{12}

	x	y
q_1	$\top$	$\top$
q_2	$+$	$\top$
q_3	$+$	$\top$
q_6	$+$	$\top$
q_7	$+$	$+$
q_8	$+$	$+$
q_{11}	$+$	$\top$
q_{12}	$+$	$+$

Forward Sign Analysis on Example with Enhanced $\overline{h_{\text{op}}}$

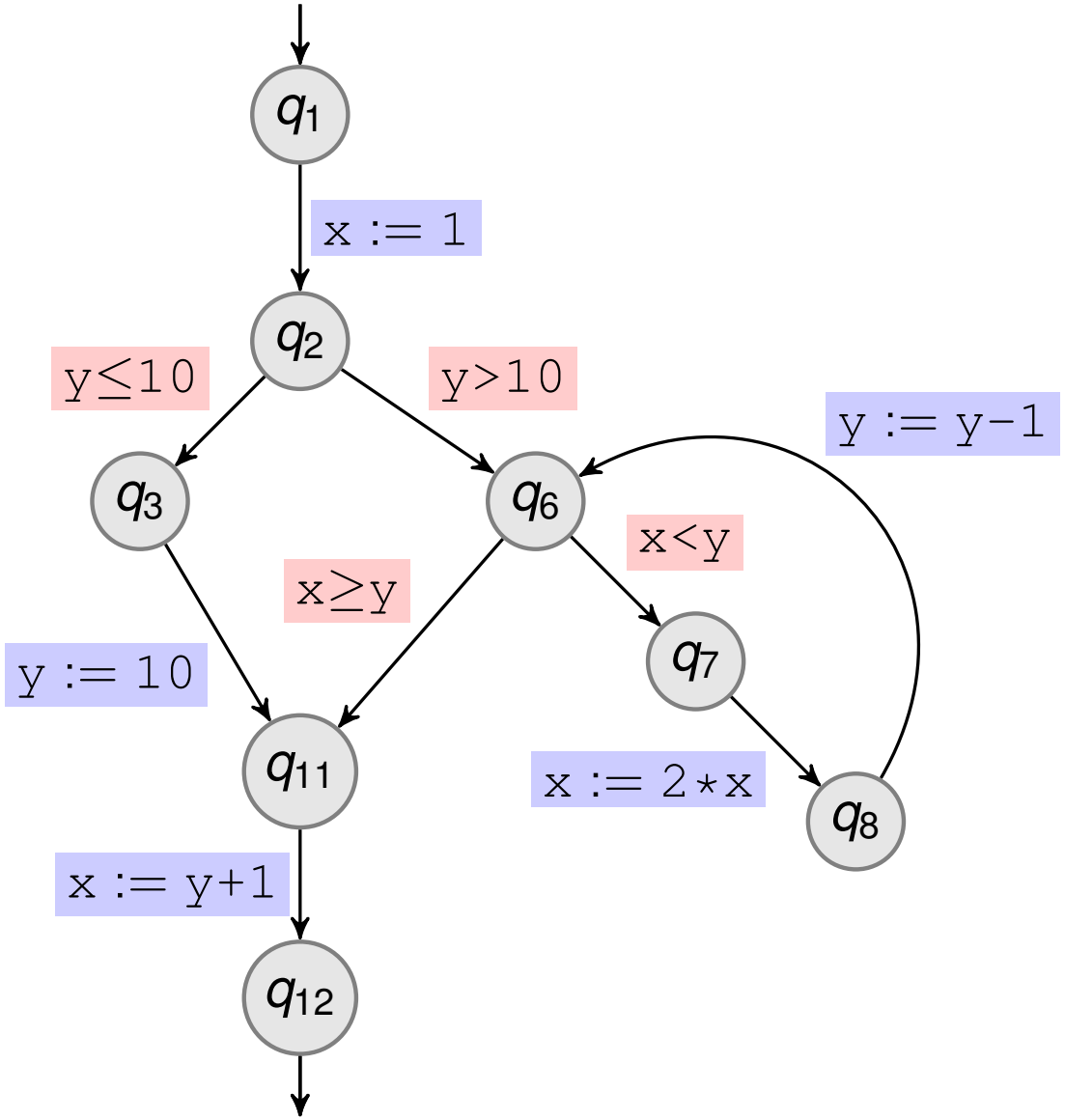


Goal

Show that $x > 0$ at q_{12}

	x	y
q_1	$\top$	$\top$
q_2	$+$	$\top$
q_3	$+$	$\top$
q_6	$+$	$\top$
q_7	$+$	$+$
q_8	$+$	$+$
q_{11}	$+$	$\top$
q_{12}	$+$	$+$

Forward Sign Analysis on Example with Enhanced $\overline{h_{\text{op}}}$

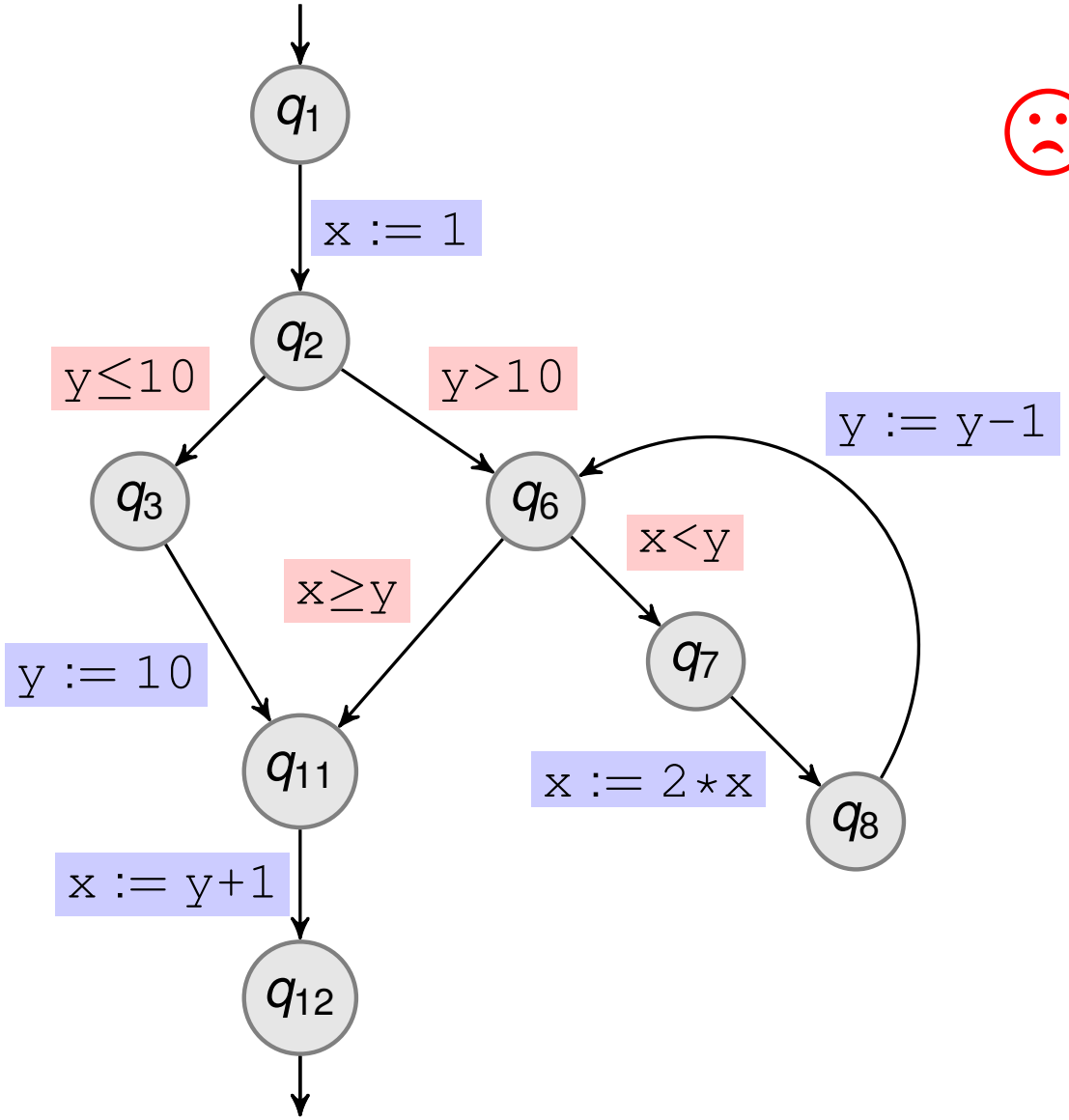


Goal

Show that $x > 0$ at q_{12}

	x	y
q_1	$\top$	$\top$
q_2	$+$	$\top$
q_3	$+$	$\top$
q_6	$+$	$\top$
q_7	$+$	$+$
q_8	$+$	$+$
q_{11}	$+$	$\top$
q_{12}	$\top$	$\top$

Forward Sign Analysis on Example with Enhanced $\overline{h_{\text{op}}}$

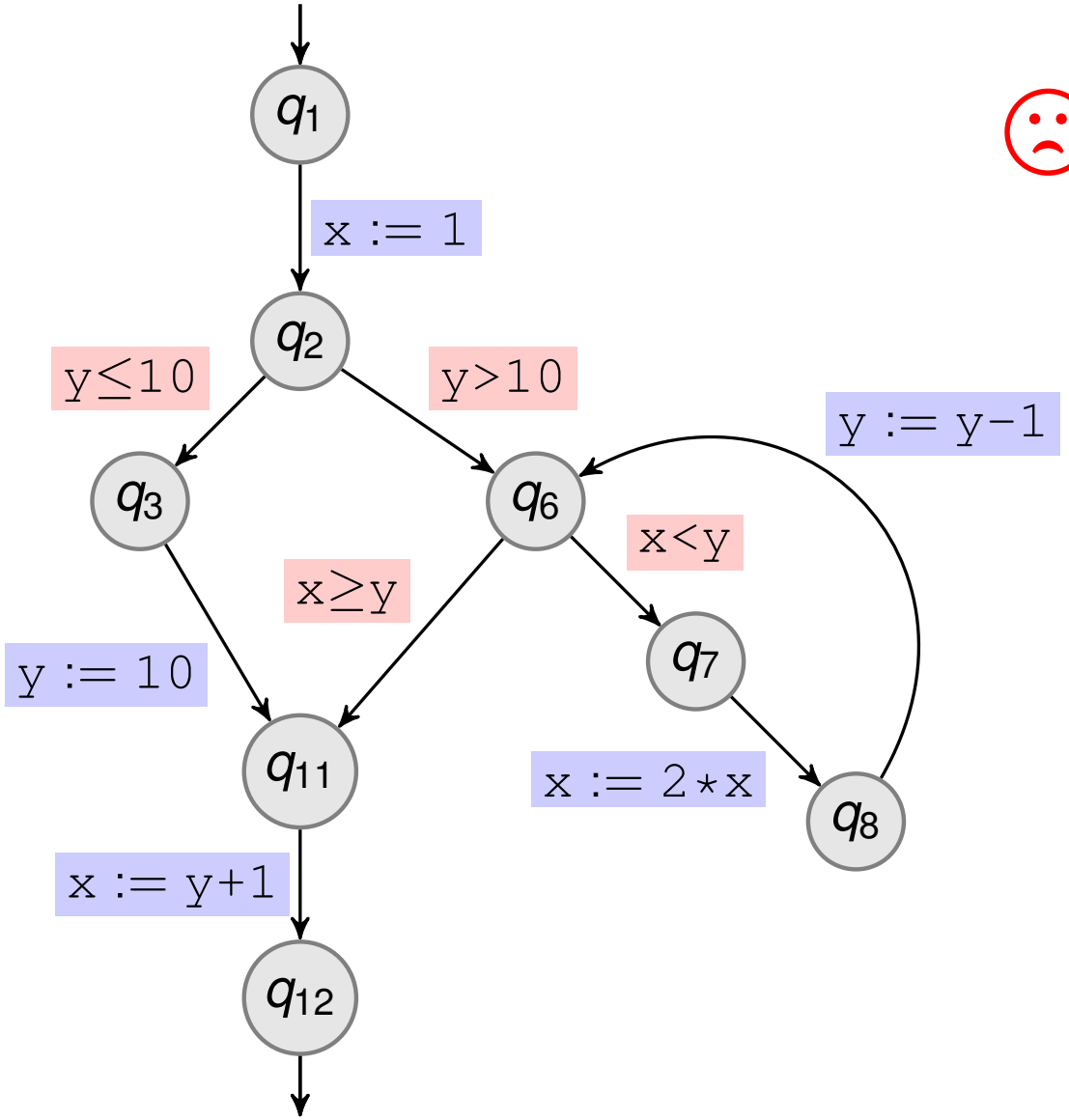


Goal

Show that x $>$ 0 at q_{12}

	x	y
q_1	$\top$	$\top$
q_2	$+$	$\top$
q_3	$+$	$\top$
q_6	$+$	$\top$
q_7	$+$	$+$
q_8	$+$	$+$
q_{11}	$+$	$\top$
q_{12}	$\top$	$\top$

Forward Sign Analysis on Example with Enhanced $\overline{h_{\text{op}}}$



☹

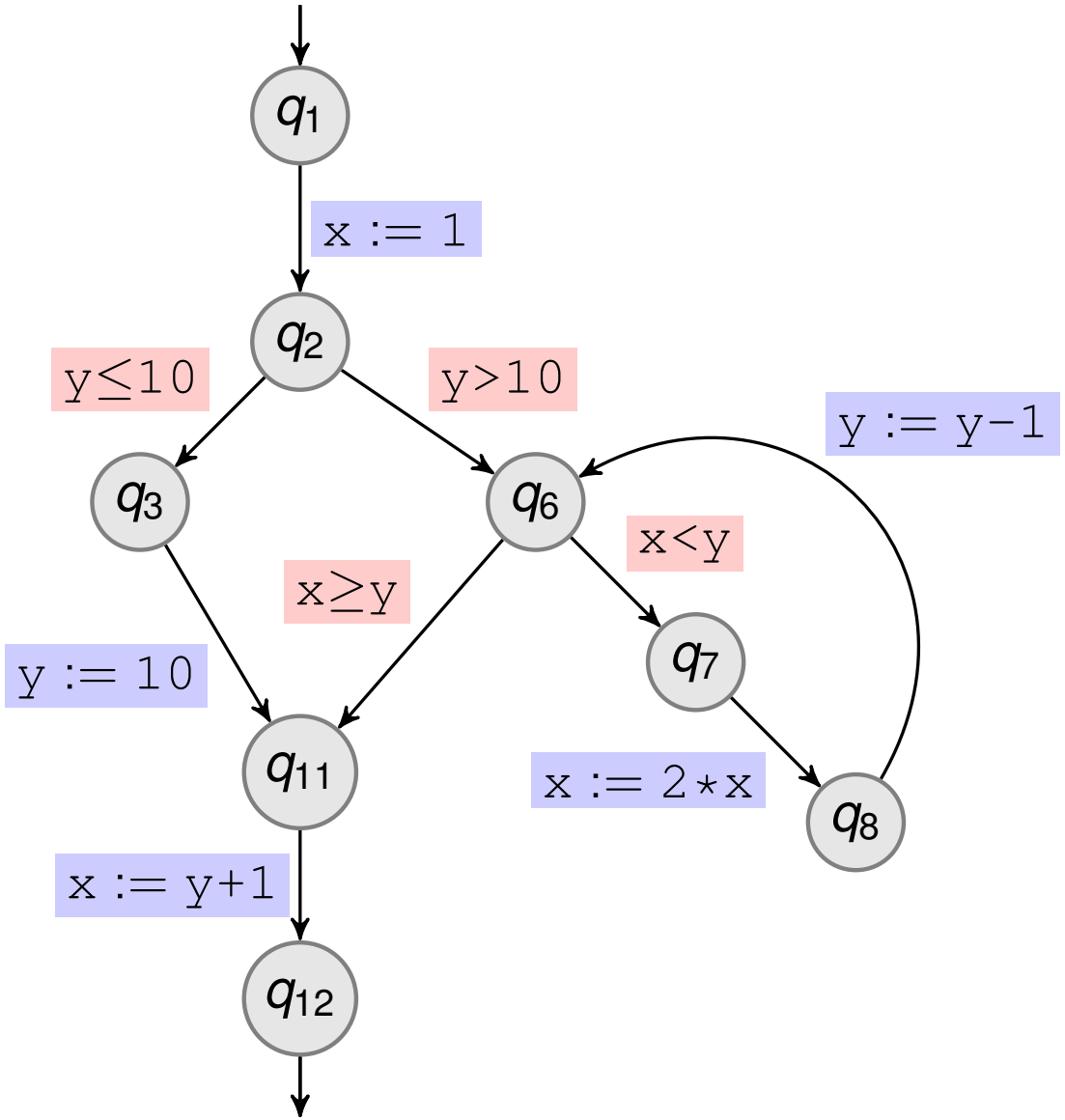
Goal

Show that $x > 0$ at q_{12}

	x	y
q_1	$\top$	$\top$
q_2	+	$\top$
q_3	+	$\top$
q_6	+	$\top$
q_7	+	+
q_8	+	+
q_{11}	+	$\top$
q_{12}	$\top$	$\top$

Getting closer...

Forward Sign Analysis on Example with Enhanced $\overline{h_{\text{op}}}$

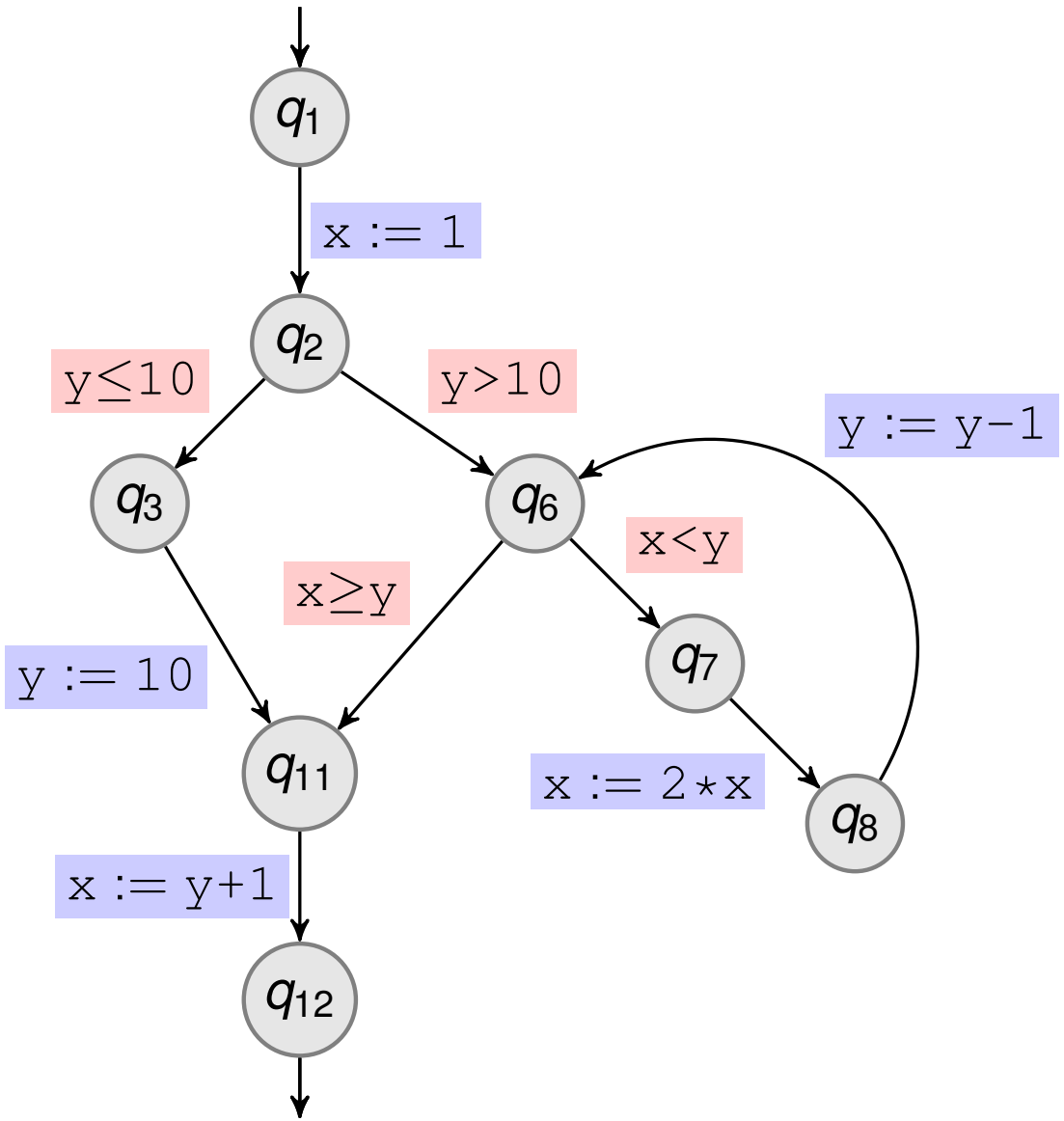


Show that $x > 0$ at q_{12}

Assumption

Variables range over $\mathbb{Z}$

Forward Sign Analysis on Example with Enhanced $\overline{h_{\mathrm{op}}}$



Show that $\mathtt{x} > 0$ at q_{12}

Assumption

Variables range over $\mathbb{Z}$

$$\gamma_{sign}(+) \quad = \quad \{1, 2, \ldots\}$$

Tuned Semantics

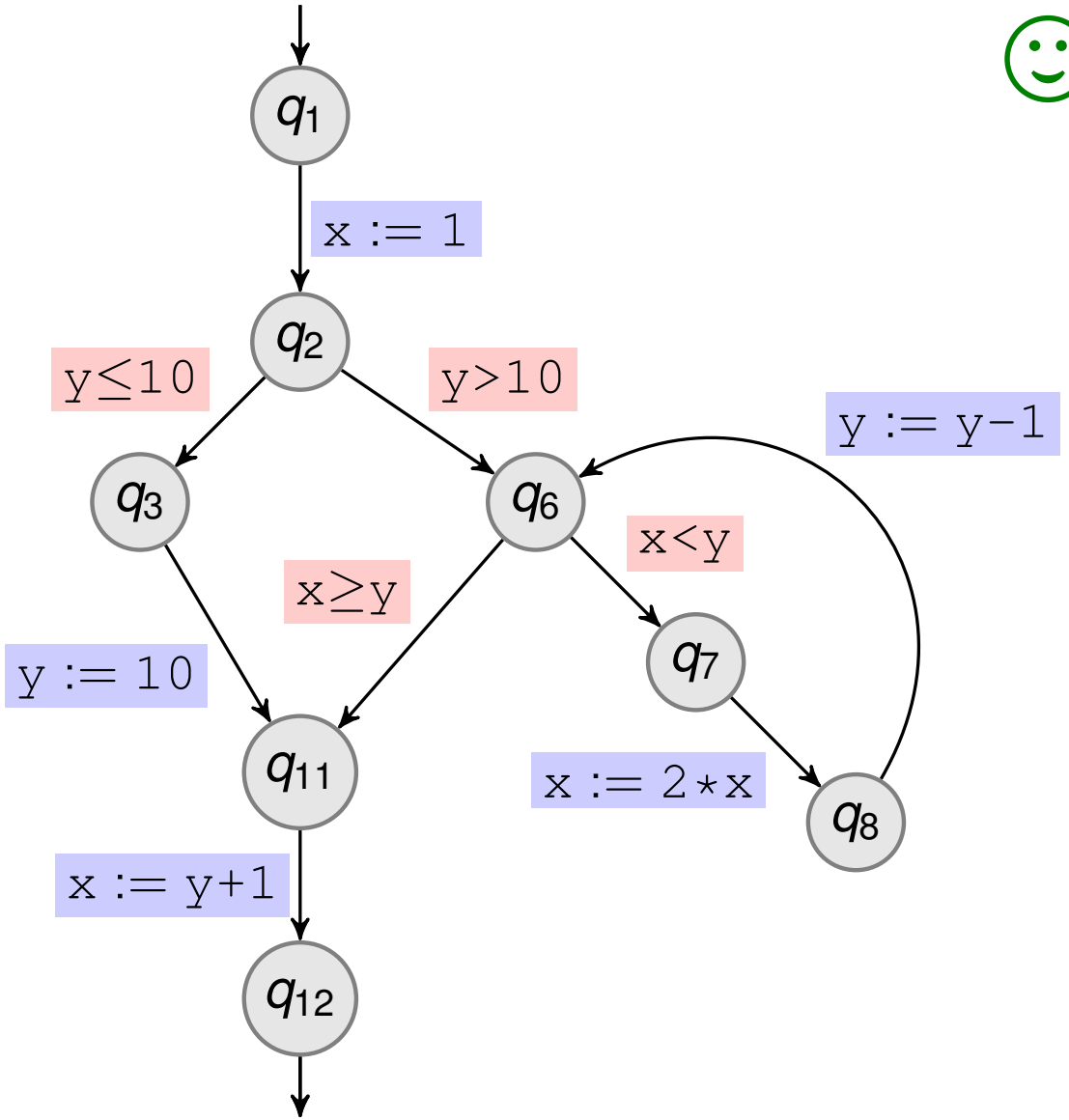
To exploit this new γ_{sign}

$\mathtt{op} \;=\; x := e - 1$

If $\overline{[\![e]\!]}_{\overline{v}} = +$ then

$$\overline{h_{\mathrm{op}}}(\overline{v})(x) \;=\; 0+$$

Forward Sign Analysis on Example with Enhanced $\overline{h_{\text{op}}}$



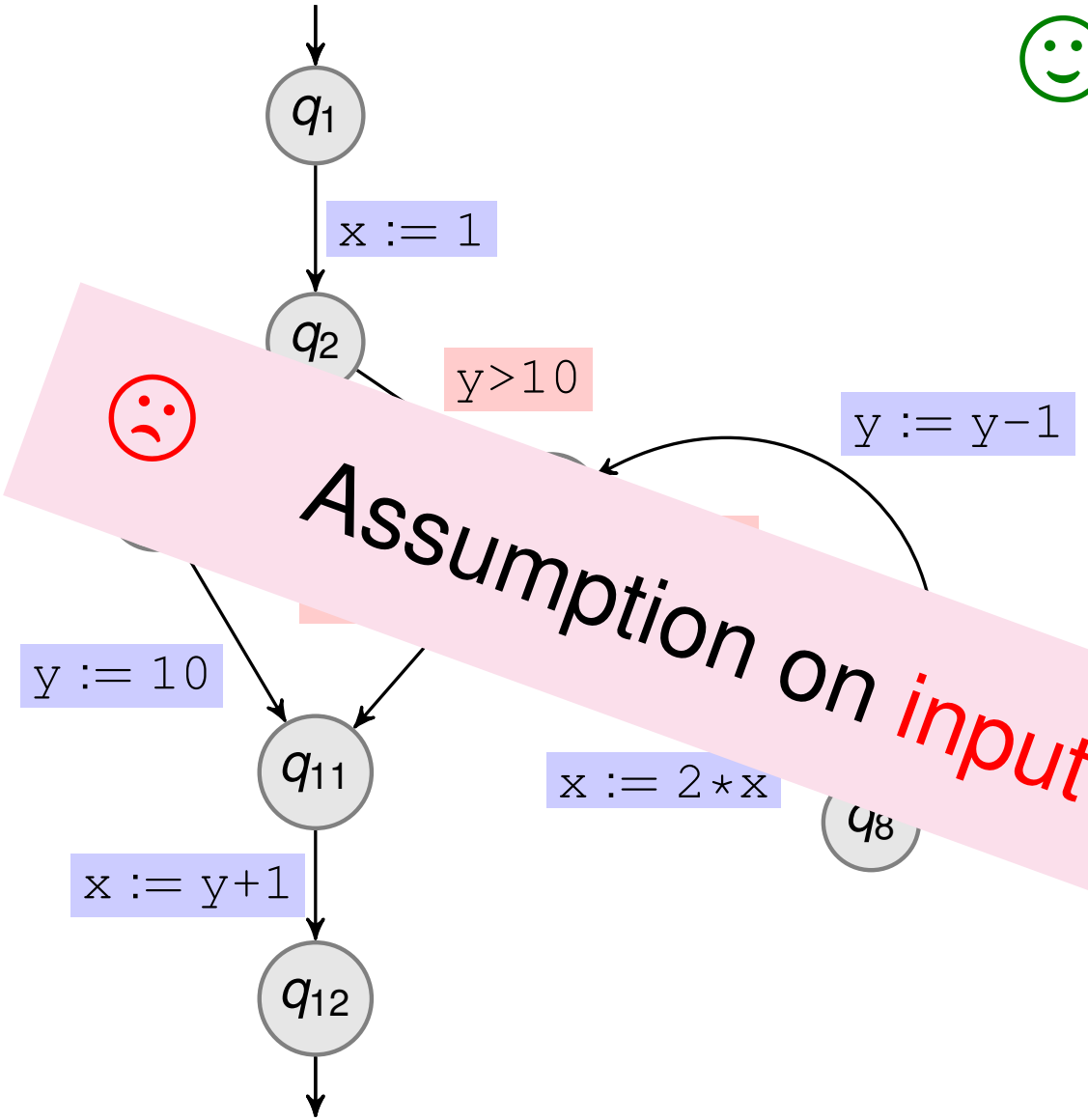
Show that $x > 0$ at q_{12}

Assumption

Variables range over $\mathbb{Z}$

	x	y
q_1	$\top$	$\top$
q_2	+	$\top$
q_3	+	$\top$
q_6	+	0+
q_7	+	+
q_8	+	+
q_{11}	+	0+
q_{12}	+	+

Forward Sign Analysis on Example with Enhanced $\overline{h_{\mathrm{op}}}$



Show that $x > 0$ at q_{12}

Assumption

Variables range over $\mathbb{Z}$

	x	y
q_1	$\top$	$\top$
q_2	+	$\top$
q_3	+	$\top$
	+	0+
		+
q_{11}		
q_{12}	+	

Assumption on input variable y

Beyond Sign Analysis...

A careful inspection of the control flow automaton shows that the desired property $\mathtt{x} > 0$ at q_{12} holds for any real initial value of $\mathtt{y}$.

This property cannot be obtained even with best abstract transfer mapping $f^{\sharp}_{\mathtt{op}}$.

The *Sign* abstract lattice is not sufficient!

Solution

Try with a finer abstract lattice!

Beyond Sign Analysis...

A careful inspection of the control flow automaton shows that the desired property $\mathtt{x} > 0$ at q_{12} holds for any real initial value of $\mathtt{y}$.

This property cannot be obtained even with best abstract transfer mapping $f^{\sharp}_{\mathtt{op}}$.

The *Sign* abstract lattice is not sufficient!

Solution

Try with a finer abstract lattice!

Outline — Abstract Interpretation

Short Introduction to Widening and Narrowing

So far: finite height lattices. Iterative computation of MOP converges.

Finite height lattices not sufficient for (precise) numerical analysis

Software Verification by Invariant Generation

Good precision is required for generation of useful invariants

Infinite height abstract lattices required to obtain good precision

But iterative computation of MOP may diverge in infinite height lattices.

Solution

1. Use widening to compute a sound approximation of MOP.
2. Use narrowing to improve the precision of the approximation.

Short Introduction to Widening and Narrowing

So far: finite height lattices. Iterative computation of MOP converges.

Finite height lattices not sufficient for (precise) numerical analysis

Software Verification by Invariant Generation

Good precision is required for generation of useful invariants

Infinite height abstract lattices required to obtain good precision

But iterative computation of MOP may diverge in infinite height lattices.

Solution

1. Use widening to compute a sound approximation of MOP.
2. Use narrowing to improve the precision of the approximation.

Illustration with Range Analysis

Objective of Range Analysis

Discover for each location the range of possible run-time values that variables can have at that location.

Generalizes both sign analysis and constant propagation analysis.

We will first design a Galois Connection between:

- $(\mathcal{P}(\mathbb{X} \to \mathbb{R}), \subseteq)$, concrete data flow facts from standard semantics
- $(\mathbb{X} \to \overline{L}, \overline{\sqsubseteq})$, where $(\overline{L}, \overline{\sqsubseteq})$ is an abstract lattice to represent range information.

Forward analysis

Complete Lattice $\mathcal{Z}$: Extension of $\mathbb{Z}$ with $-\infty$ and $+\infty$

Let $\mathcal{Z} = \mathbb{Z} \cup \{-\infty, +\infty\}$ and define the partial order $\leq$ on $\mathcal{Z}$ with:

$$-\infty < \cdots < -2 < -1 < 0 < 1 < 2 < \cdots < +\infty$$

$(\mathcal{Z}, \leq)$ is a complete lattice

- Least element is $-\infty$ and greatest element is $+\infty$.
- Least upper bound $\sup X$ is either $\max(X)$ if it exists, or $+\infty$.
- Greatest lower bound $\inf X$ is either $\min(X)$ if it exists, or $-\infty$.

Abstract Lattice of Intervals

$$-\infty < \cdots < -2 < -1 < 0 < 1 < 2 < \cdots < +\infty$$

$$\mathit{Int} \;=\; \{\bot\} \cup \{(l,u) \in (\mathbb{Z} \cup \{-\infty\}) \times (\mathbb{Z} \cup \{+\infty\}) \mid l \leq u\}$$

Define the binary relation $\sqsubseteq$ on Int as follows:

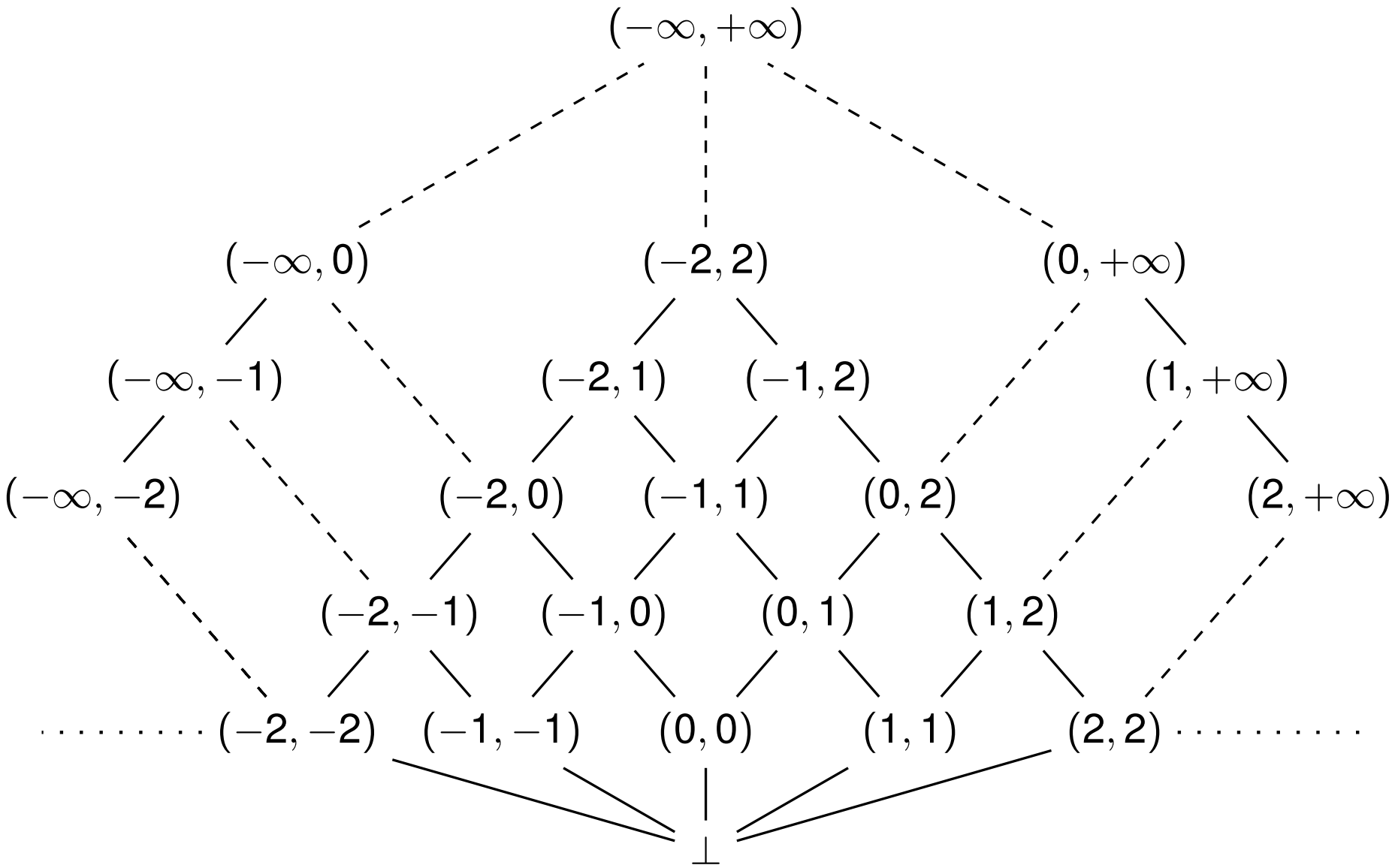
$$\bot \sqsubseteq \bot \qquad\qquad \bot \sqsubseteq (l,u) \not\sqsubseteq \bot$$

$$(l_1,u_1) \sqsubseteq (l_2,u_2) \quad \text{if} \quad l_1 \leq l_2 \leq u_2 \leq u_1$$

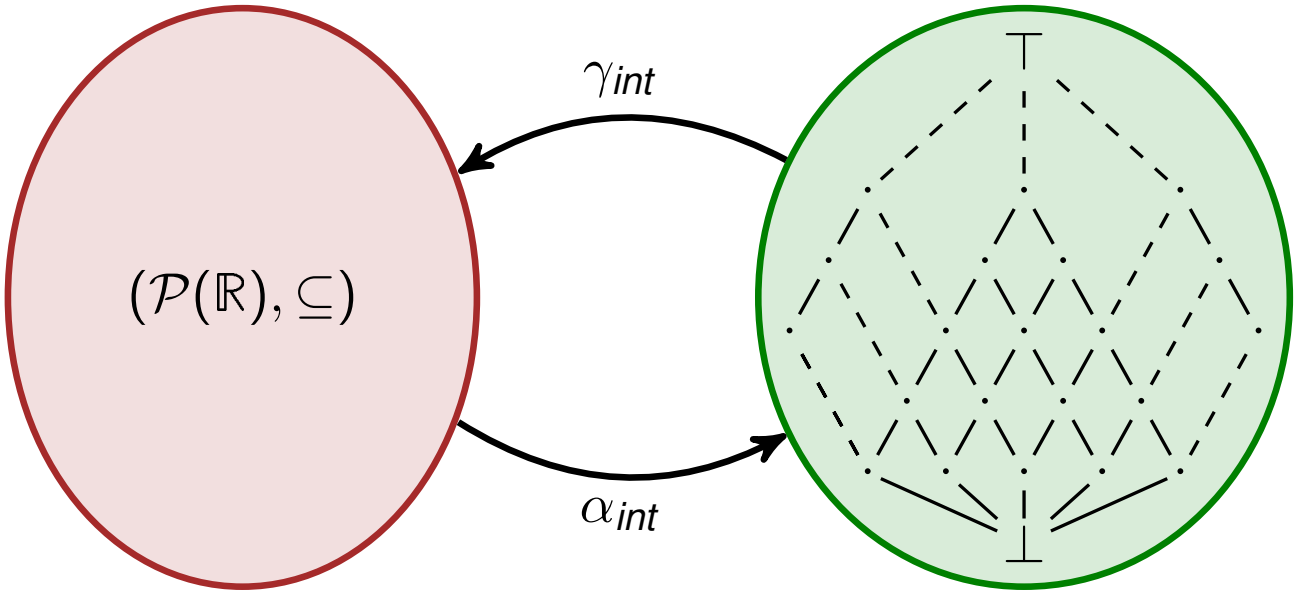
$(\mathit{Int}, \sqsubseteq)$ is a complete lattice

- Least element is $\bot$ and greatest element is $(-\infty, +\infty)$.
- $\bigsqcup X$ is either $\bot$ or $(\inf\{l \mid (l,u) \in X\}, \sup\{u \mid (l,u) \in X\})$.
- $\bigsqcap X$ is either $\bot$ or $(\sup\{l \mid (l,u) \in X\}, \inf\{u \mid (l,u) \in X\})$.

Abstract Lattice of Intervals



Interpretation of Abstract Intervals: Galois Connection



$$\alpha_{int}(\phi) = \begin{cases} \overline{\bot} & \text{if } \phi = \emptyset \\ (\inf\{\lfloor r \rfloor \mid r \in \phi\}, \sup\{\lceil r \rceil \mid r \in \phi\}) & \text{otherwise} \end{cases}$$

$$\gamma_{int}(\overline{\bot}) = \emptyset$$

$$\gamma_{int}(\overline{(l, u)}) = \{r \in \mathbb{R} \mid l \leq r \leq u\}$$

Range Analysis with Intervals: Galois Connection

Projection

$$(\mathcal{P}(\mathbb{X} \to \mathbb{R}), \subseteq) \underset{\alpha_\pi}{\overset{\gamma_\pi}{\leftrightarrows}} (\mathbb{X} \to \mathcal{P}(\mathbb{R}), \subseteq)$$

Intervals

$$(\mathcal{P}(\mathbb{R}), \subseteq) \underset{\alpha_{int}}{\overset{\gamma_{int}}{\leftrightarrows}} (\mathit{Int}, \overline{\sqsubseteq})$$

Extension of Intervals to Functions

$$(\mathbb{X} \to \mathcal{P}(\mathbb{R}), \subseteq) \underset{\alpha_{int}}{\overset{\gamma_{int}}{\leftrightarrows}} (\mathbb{X} \to \mathit{Int}, \overline{\sqsubseteq})$$

$$(\mathcal{P}(\mathbb{X} \to \mathbb{R}), \subseteq) \underset{\alpha_{int} \circ \alpha_\pi}{\overset{\gamma_\pi \circ \gamma_{int}}{\leftrightarrows}} (\mathbb{X} \to \mathit{Int}, \overline{\sqsubseteq})$$

Range Analysis with Intervals: Galois Connection

Projection

$$(\mathcal{P}(\mathrm{X} \to \mathbb{R}), \subseteq) \underset{\alpha_\pi}{\overset{\gamma_\pi}{\leftrightarrows}} (\mathrm{X} \to \mathcal{P}(\mathbb{R}), \subseteq)$$

Intervals

$$(\mathcal{P}(\mathbb{R}), \subseteq) \underset{\alpha_{int}}{\overset{\gamma_{int}}{\leftrightarrows}} (\mathrm{X} \to \mathit{Int}, \overline{\sqsubseteq})$$

...als to Functions

$$(\mathcal{P}(\mathbb{R}), \subseteq) \underset{\alpha_{int}}{\overset{\gamma_{int}}{\leftrightarrows}} (\mathrm{X} \to \mathit{Int}, \overline{\sqsubseteq})$$

Same as sign analysis

$$(\mathcal{P}(\mathrm{X} \to \mathbb{R}), \subseteq) \underset{\alpha_{int} \circ \alpha_\pi}{\overset{\gamma_\pi \circ \gamma_{int}}{\leftrightarrows}} (\mathrm{X} \to \mathit{Int}, \overline{\sqsubseteq})$$

Range Analysis: Induced Abstract Data Flow Instance

$$(\mathcal{P}(\mathrm{X} \to \mathbb{R}), \subseteq) \underset{\alpha_\pi}{\overset{\gamma_\pi}{\leftrightarrows}} (\mathrm{X} \to \mathcal{P}(\mathbb{R}), \subseteq) \underset{\alpha_{int}}{\overset{\gamma_{int}}{\leftrightarrows}} (\mathrm{X} \to \mathit{Int}, \overline{\sqsubseteq})$$

$$\begin{array}{rcl} \overline{\mathcal{F}} & = & (\mathrm{X} \to \mathit{Int}) \xrightarrow{\text{mon}} (\mathrm{X} \to \mathit{Int}) \\ \overline{f} & = & \lambda\, \mathrm{op}\,.\, f^\sharp_{\mathrm{op}} \\ \overline{\imath} & = & \alpha_{int} \circ \alpha_\pi(\top) = \lambda\, x\,.\, \overline{\top} \end{array}$$

$$f^\sharp_{x:=e}(\overline{v}) = \lambda\, y\,. \begin{cases} \overline{v}(y) & \text{if } y \neq x \\ \alpha_{int} \circ [\![e]\!] \circ \gamma(\overline{v}) & \text{if } y = x \end{cases}$$

$$f^\sharp_g = \alpha_{int} \circ \alpha_\pi \circ [\![g]\!] \circ \gamma_\pi \circ \gamma_{int}$$

Range Analysis: Induced Abstract Data Flow Instance

$$(\mathcal{P}(\mathrm{X} \to \mathbb{R}), \subseteq) \underset{\alpha_\pi}{\overset{\gamma_\pi}{\leftrightarrows}} (\mathrm{X} \to \mathcal{P}(\mathbb{R}), \subseteq) \underset{\alpha_{int}}{\overset{\gamma_{int}}{\leftrightarrows}} (\mathrm{X} \to \mathit{Int}, \overline{\sqsubseteq})$$

$$\begin{aligned} \overline{\mathcal{F}} &= (\mathrm{X} \to \mathit{Int}) \xrightarrow{\text{mon}} (\mathrm{X} \to \mathit{Int}) \\ \overline{f} &= \lambda\, \mathrm{op}\,.\, f^\sharp_{\mathrm{op}} \\ \overline{\imath} &= \alpha_{int} \circ \end{aligned}$$

$$\lambda y.\begin{cases} \overline{v}(y) & \text{if } y \neq x \\ \alpha_{int} \circ [\![e]\!] \circ \gamma(\overline{v}) & \text{if } y = x \end{cases}$$

$$f^\sharp_g = \alpha_{int} \circ \alpha_\pi \circ [\![g]\!] \circ \gamma_\pi \circ \gamma_{int}$$

Same as sign analysis

Range Analysis: Approximate Transfer Mapping

$$\overline{h_{x:=e}}(\overline{v}) = \lambda y . \begin{cases} \overline{v}(y) & \text{if } y \neq x \\ \overline{[\![e]\!]}_{\overline{v}} & \text{if } y = x \end{cases} \qquad \overline{h_g}(\overline{v}) = \begin{cases} \overline{\bot} & \text{if } \overline{v} \not\models g \\ \overline{v} & \text{if } \overline{v} \models g \end{cases}$$

Similar to Sign Analysis

- Definition of abstract arithmetic operators and comparators
- Definition of $\overline{[\![e]\!]}_{\overline{v}}$ and of $\overline{v} \models g$
- Precision enhancement by gaining information from guards ("refinement" of $\overline{h_g}(\overline{v})$)

Care about effective computability of $f^\sharp_{\text{op}}$? Not in this section...

We will use $f^\sharp$ for range analysis

Range Analysis: Approximate Transfer Mapping

$$\overline{h_{x:=e}}(\overline{v}) = \lambda y . \begin{cases} \overline{v}(y) & \text{if } y \neq x \\ \overline{[\![e]\!]}_{\overline{v}} & \text{if } y = x \end{cases} \qquad \overline{h_g}(\overline{v}) = \begin{cases} \overline{\bot} & \text{if } \overline{v} \not\models g \\ \overline{v} & \text{if } \overline{v} \models g \end{cases}$$

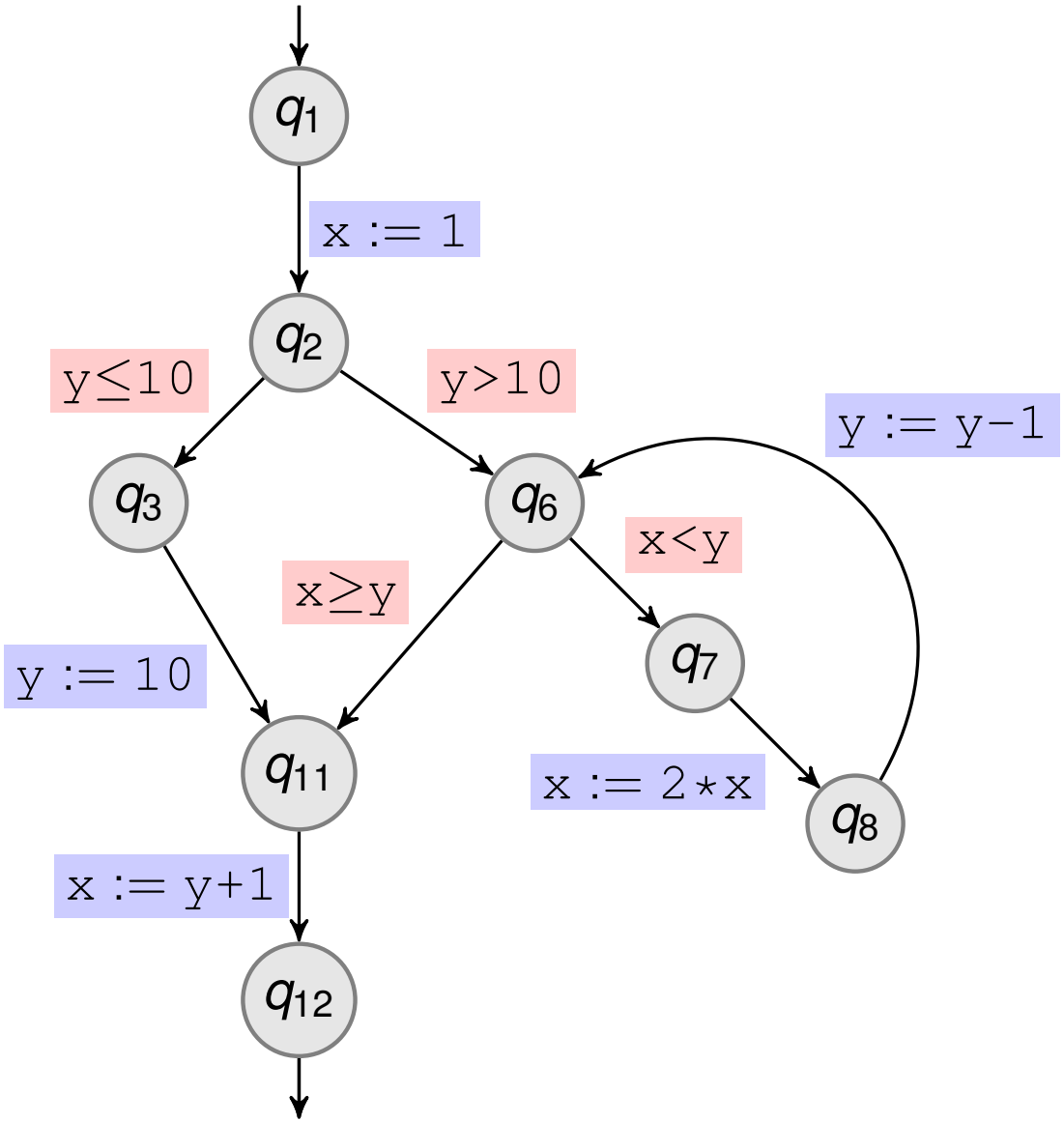
Similar to Sign Analysis

- Definition of abstract arithmetic operators and comparators
- Definition of $\overline{[\![e]\!]}_{\overline{v}}$ and of $\overline{v} \models g$
- Precision enhancement by gaining information from guards ("refinement" of $\overline{h_g}(\overline{v})$)

Care about effective computability of $f^\sharp_{\text{op}}$? Not in this section...

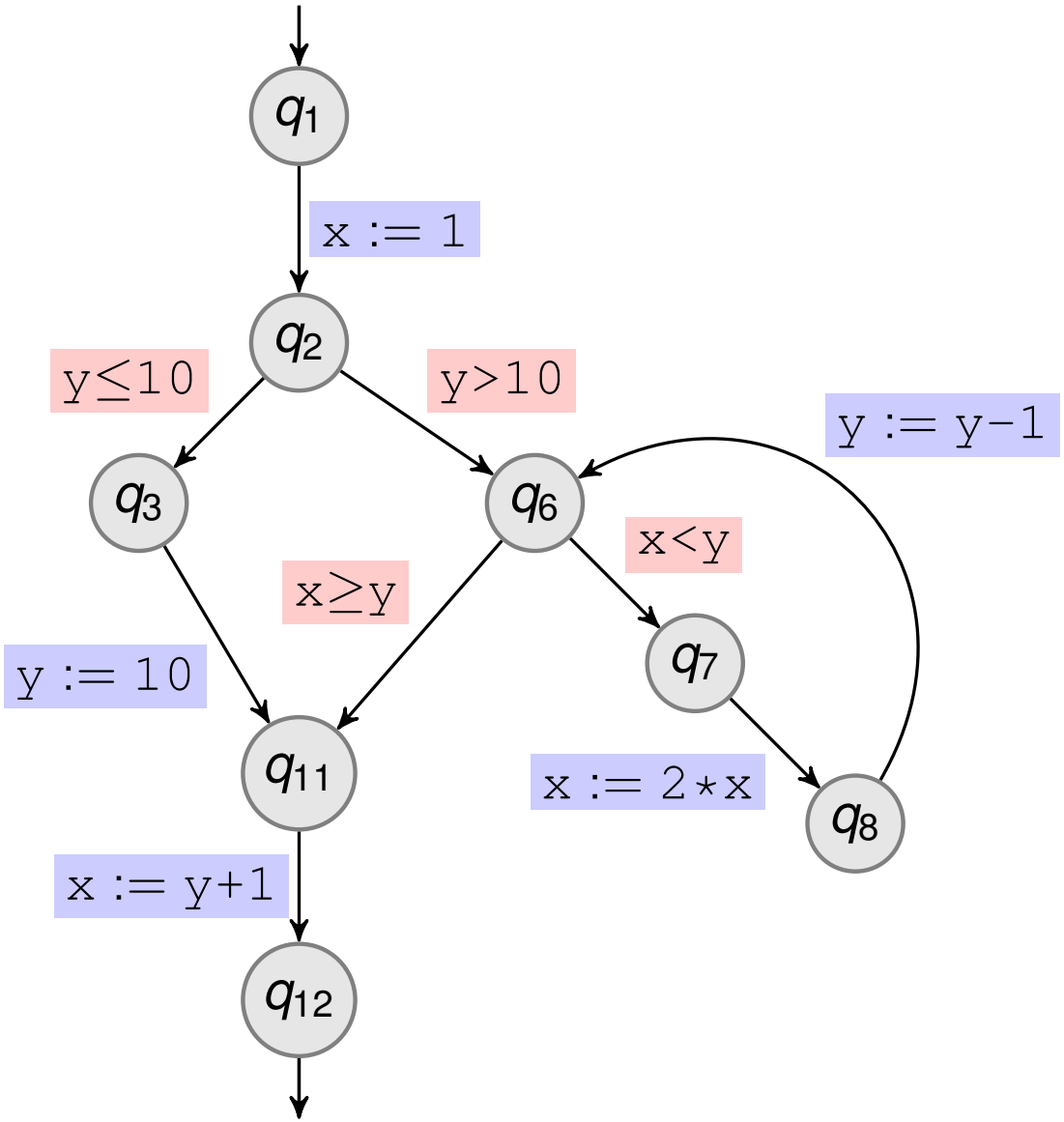
We will use $f^\sharp$ for range analysis

Range Analysis on Running Example



Recall that $\top = (-\infty, +\infty)$

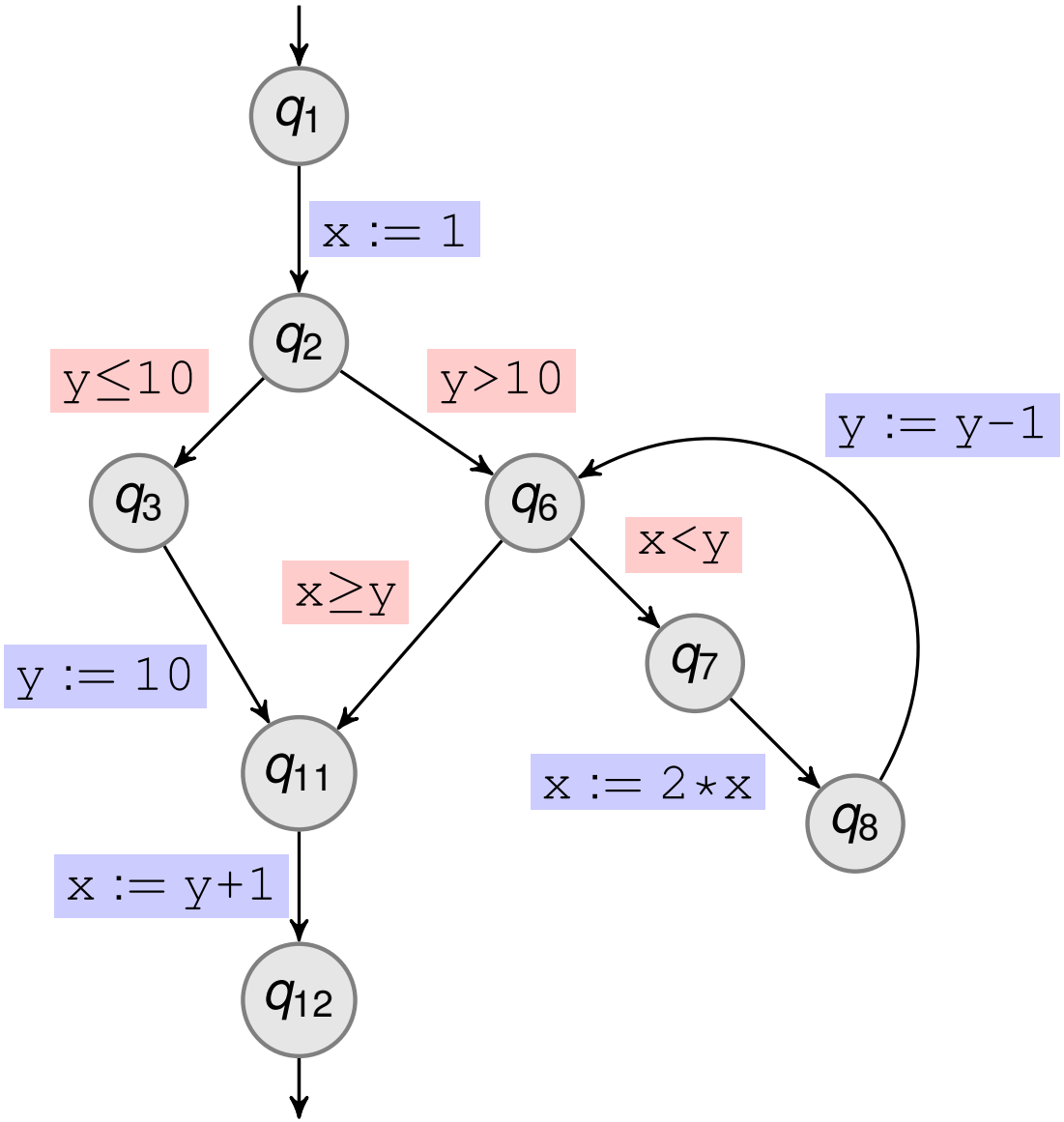
Range Analysis on Running Example



Recall that $\top = (-\infty, +\infty)$

	x		y	
q_1	$-\infty$	$+\infty$	$-\infty$	$+\infty$
q_2	$\bot$	$\bot$	$\bot$	$\bot$
q_3	$\bot$	$\bot$	$\bot$	$\bot$
q_6	$\bot$	$\bot$	$\bot$	$\bot$
q_7	$\bot$	$\bot$	$\bot$	$\bot$
q_8	$\bot$	$\bot$	$\bot$	$\bot$
q_{11}	$\bot$	$\bot$	$\bot$	$\bot$
q_{12}	$\bot$	$\bot$	$\bot$	$\bot$

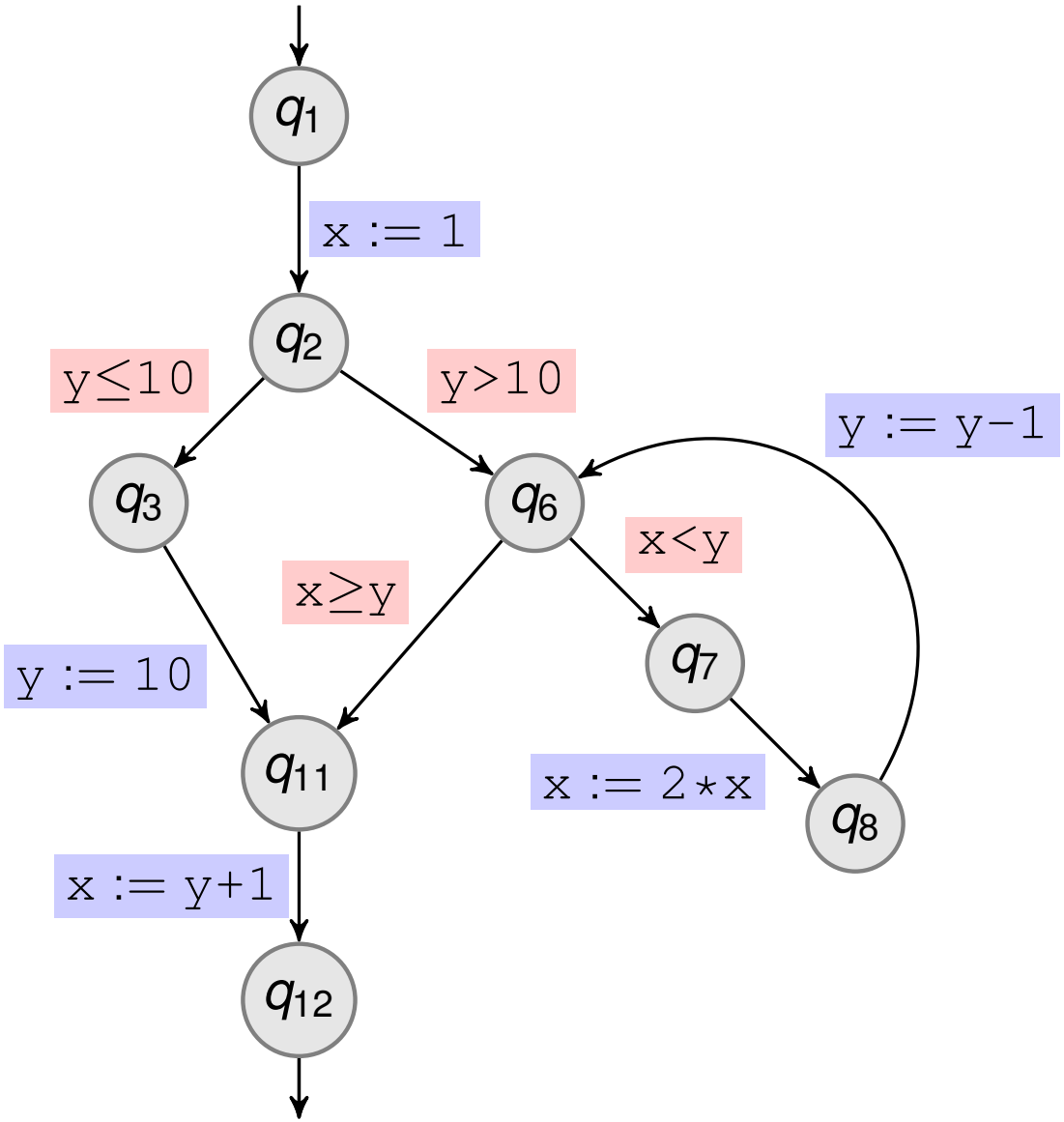
Range Analysis on Running Example



Recall that $\top = (-\infty, +\infty)$

	x		y	
q_1	$-\infty$	$+\infty$	$-\infty$	$+\infty$
q_2	1	1	$+\infty$	$+\infty$
q_3	$\bot$	$\bot$	$\bot$	$\bot$
q_6	$\bot$	$\bot$	$\bot$	$\bot$
q_7	$\bot$	$\bot$	$\bot$	$\bot$
q_8	$\bot$	$\bot$	$\bot$	$\bot$
q_{11}	$\bot$	$\bot$	$\bot$	$\bot$
q_{12}	$\bot$	$\bot$	$\bot$	$\bot$

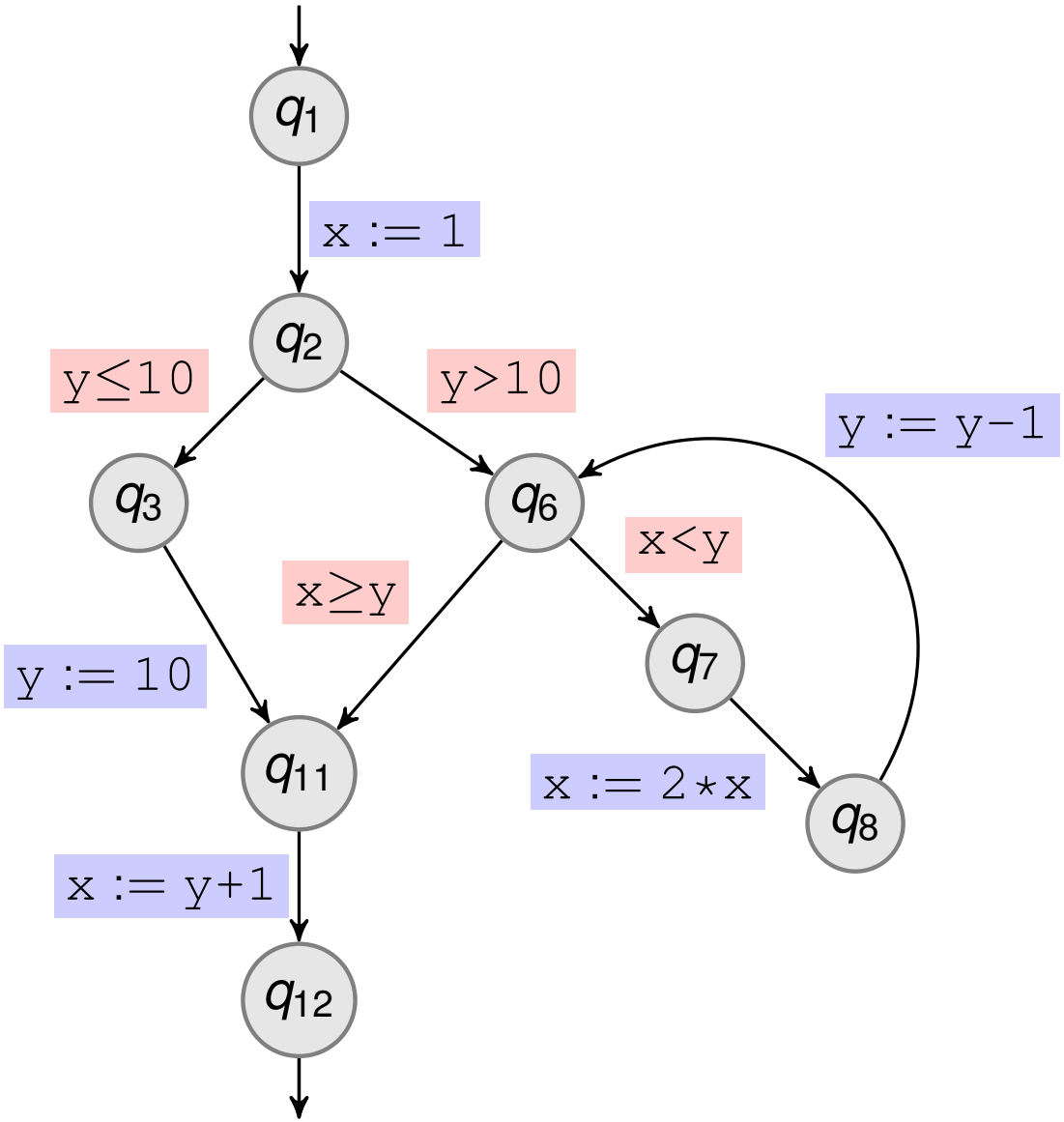
Range Analysis on Running Example



Recall that $\top = (-\infty, +\infty)$

	x		y	
q_1	$-\infty$	$+\infty$	$-\infty$	$+\infty$
q_2	1	1	$+\infty$	$+\infty$
q_3	$\bot$	$\bot$	$\bot$	$\bot$
q_6	$\bot$	$\bot$	$\bot$	$\bot$
q_7	$\bot$	$\bot$	$\bot$	$\bot$
q_8	$\bot$	$\bot$	$\bot$	$\bot$
q_{11}	$\bot$	$\bot$	$\bot$	$\bot$
q_{12}	$\bot$	$\bot$	$\bot$	$\bot$

Range Analysis on Running Example

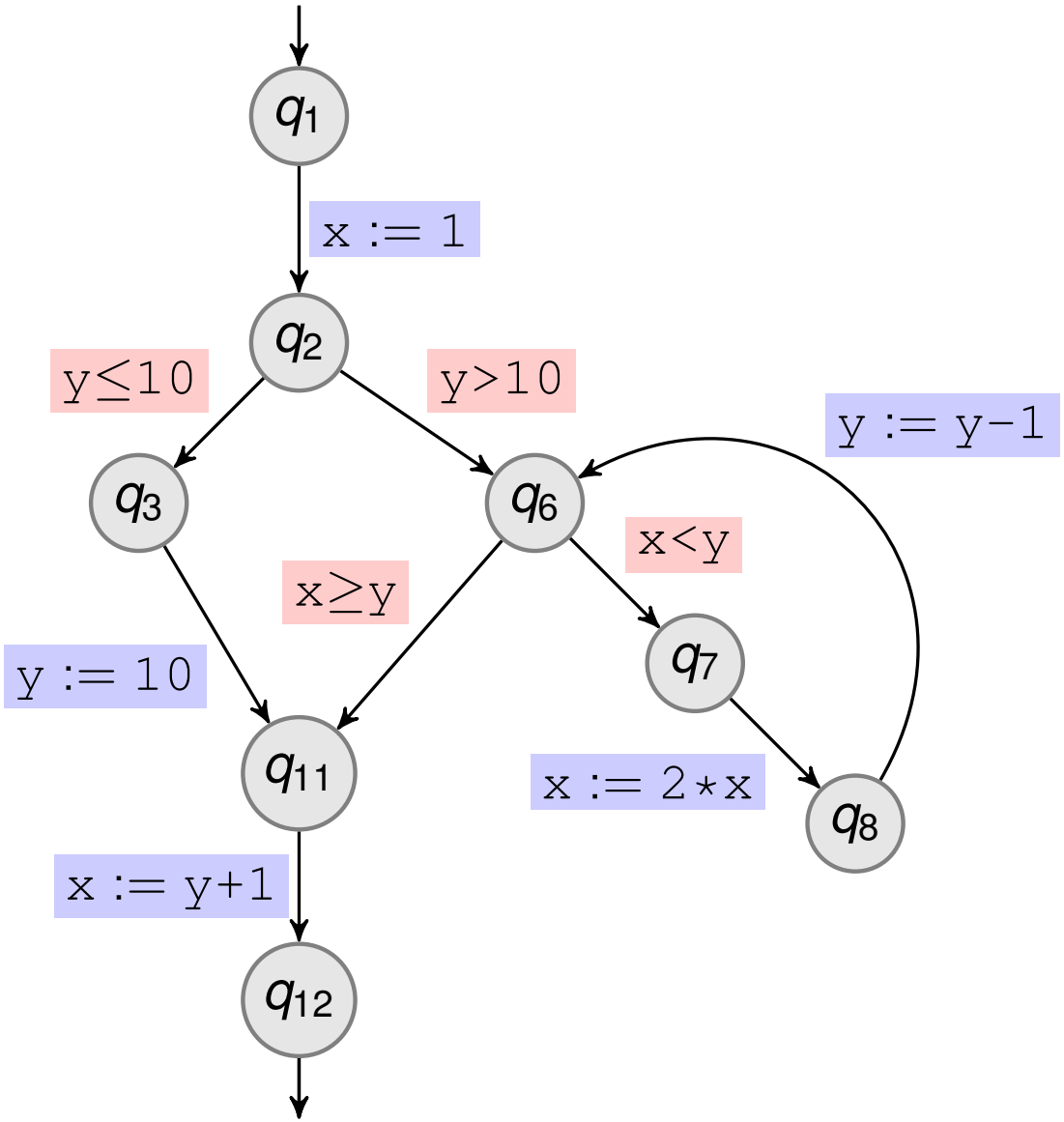


Recall that $\top = (-\infty, +\infty)$

	x		y	
q_1	$-\infty$	$+\infty$	$-\infty$	$+\infty$
q_2	1	1	$+\infty$	$+\infty$
q_3	1	1	$-\infty$	10
q_6	$\bot$	$\bot$	$\bot$	$\bot$
q_7	$\bot$	$\bot$	$\bot$	$\bot$
q_8	$\bot$	$\bot$	$\bot$	$\bot$
q_{11}	$\bot$	$\bot$	$\bot$	$\bot$
q_{12}	$\bot$	$\bot$	$\bot$	$\bot$

Gained from guard $y \leq 10$

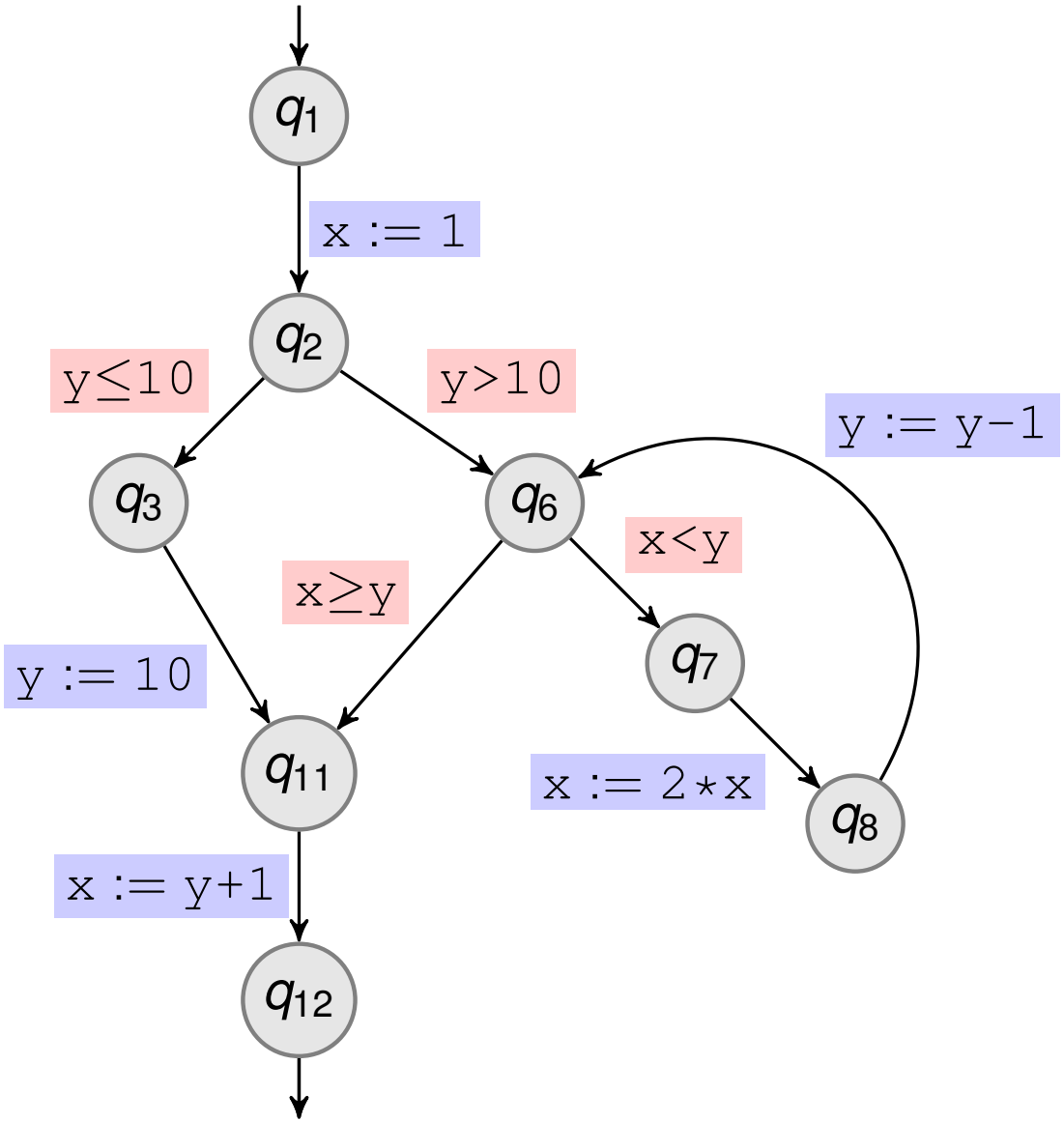
Range Analysis on Running Example



Recall that $\top = (-\infty, +\infty)$

	x		y	
q_1	$-\infty$	$+\infty$	$-\infty$	$+\infty$
q_2	1	1	$+\infty$	$+\infty$
q_3	1	1	$-\infty$	10
q_6	$\bot$	$\bot$	$\bot$	$\bot$
q_7	$\bot$	$\bot$	$\bot$	$\bot$
q_8	$\bot$	$\bot$	$\bot$	$\bot$
q_{11}	$\bot$	$\bot$	$\bot$	$\bot$
q_{12}	$\bot$	$\bot$	$\bot$	$\bot$

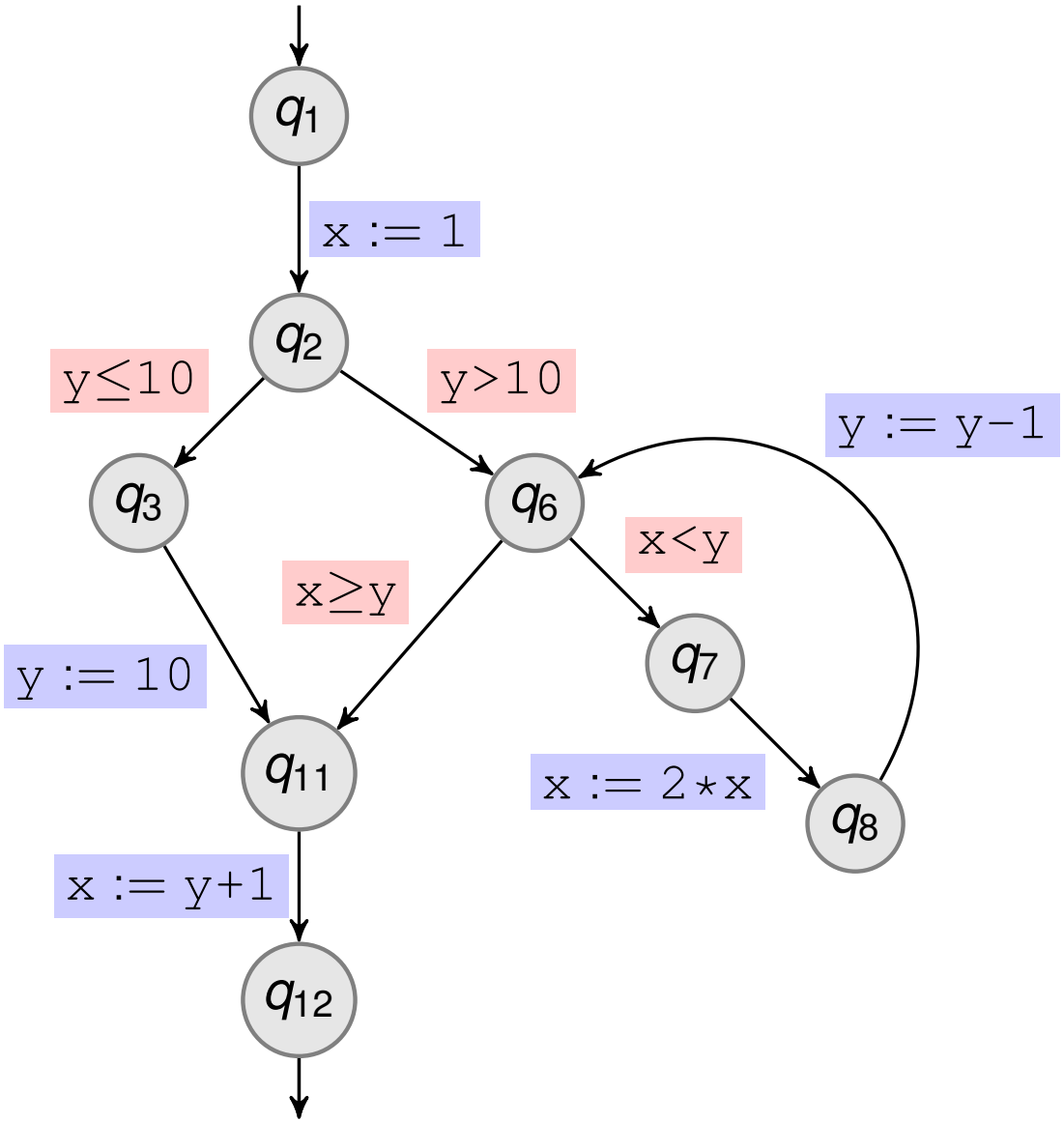
Range Analysis on Running Example



Recall that $\top = (-\infty, +\infty)$

	x		y	
q_1	$-\infty$	$+\infty$	$-\infty$	$+\infty$
q_2	1	1	$+\infty$	$+\infty$
q_3	1	1	$-\infty$	10
q_6	$\bot$	$\bot$	$\bot$	$\bot$
q_7	$\bot$	$\bot$	$\bot$	$\bot$
q_8	$\bot$	$\bot$	$\bot$	$\bot$
q_{11}	1	1	10	10
q_{12}	$\bot$	$\bot$	$\bot$	$\bot$

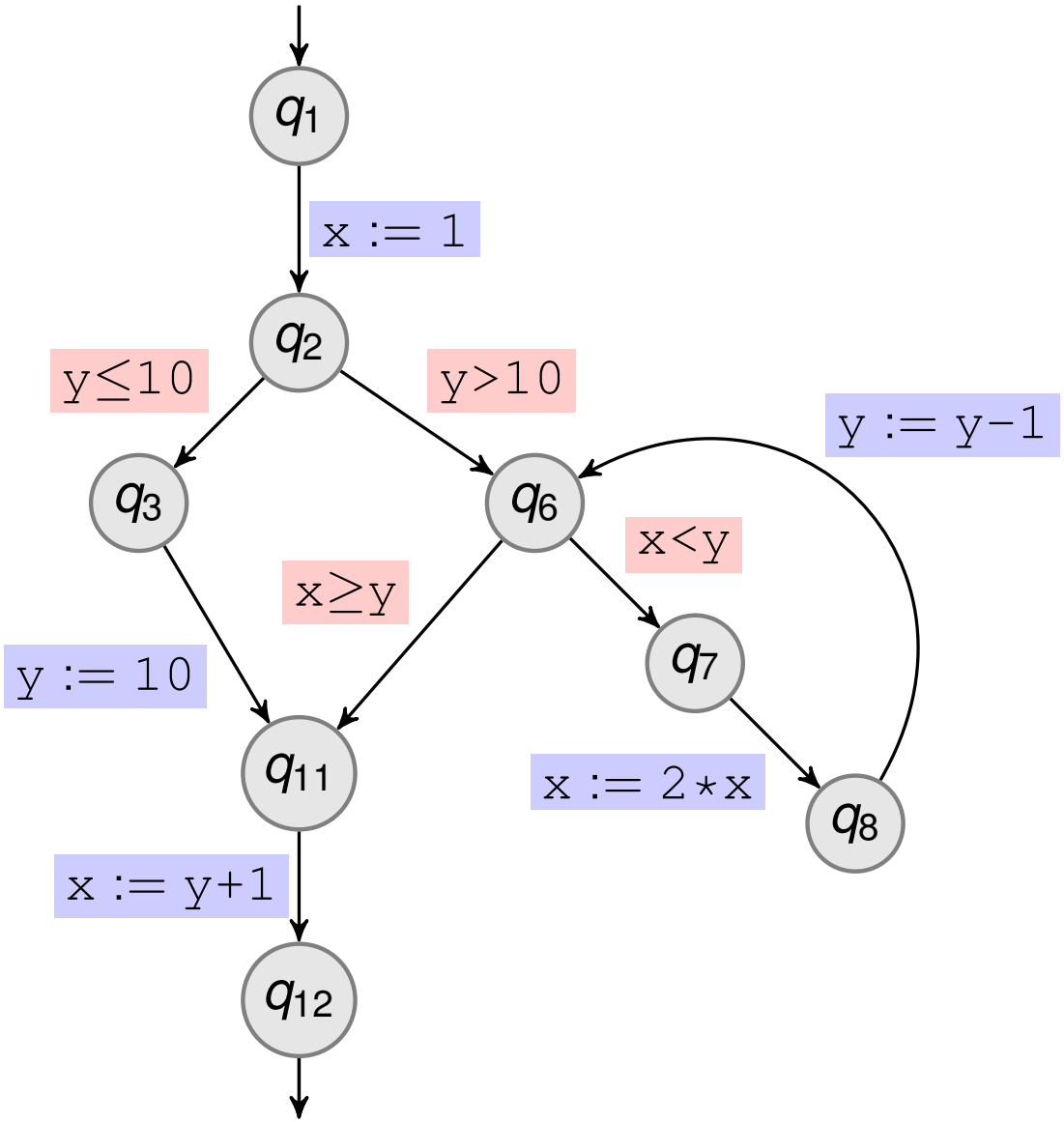
Range Analysis on Running Example



Recall that $\top = (-\infty, +\infty)$

	x		y	
q_1	$-\infty$	$+\infty$	$-\infty$	$+\infty$
q_2	1	1	$+\infty$	$+\infty$
q_3	1	1	$-\infty$	10
q_6	$\bot$	$\bot$	$\bot$	$\bot$
q_7	$\bot$	$\bot$	$\bot$	$\bot$
q_8	$\bot$	$\bot$	$\bot$	$\bot$
q_{11}	1	1	10	10
q_{12}	$\bot$	$\bot$	$\bot$	$\bot$

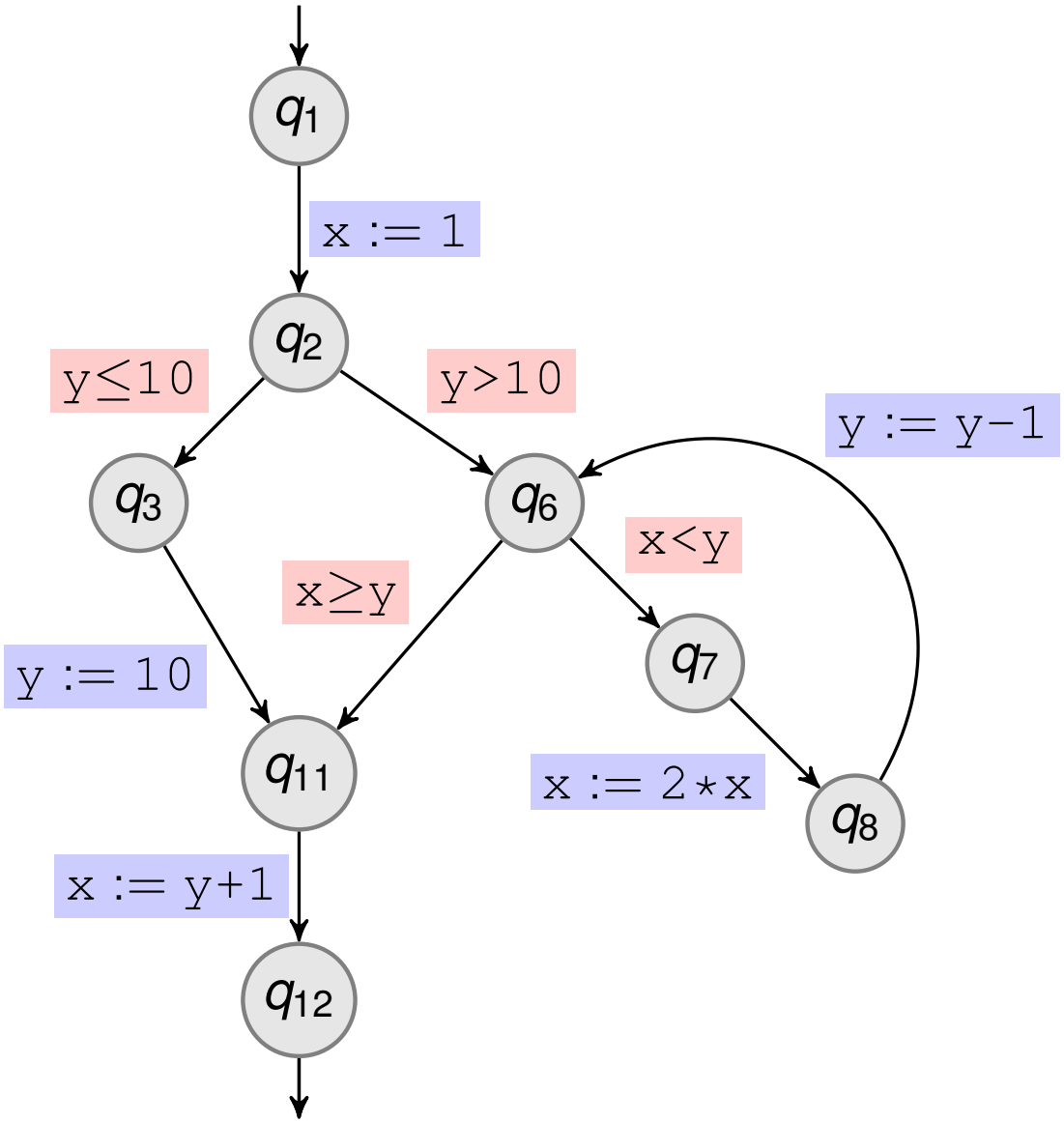
Range Analysis on Running Example



Recall that $\top = (-\infty, +\infty)$

	x		y	
q_1	$-\infty$	$+\infty$	$-\infty$	$+\infty$
q_2	1	1	$+\infty$	$+\infty$
q_3	1	1	$-\infty$	10
q_6	$\bot$	$\bot$	$\bot$	$\bot$
q_7	$\bot$	$\bot$	$\bot$	$\bot$
q_8	$\bot$	$\bot$	$\bot$	$\bot$
q_{11}	1	1	10	10
q_{12}	11	11	10	10

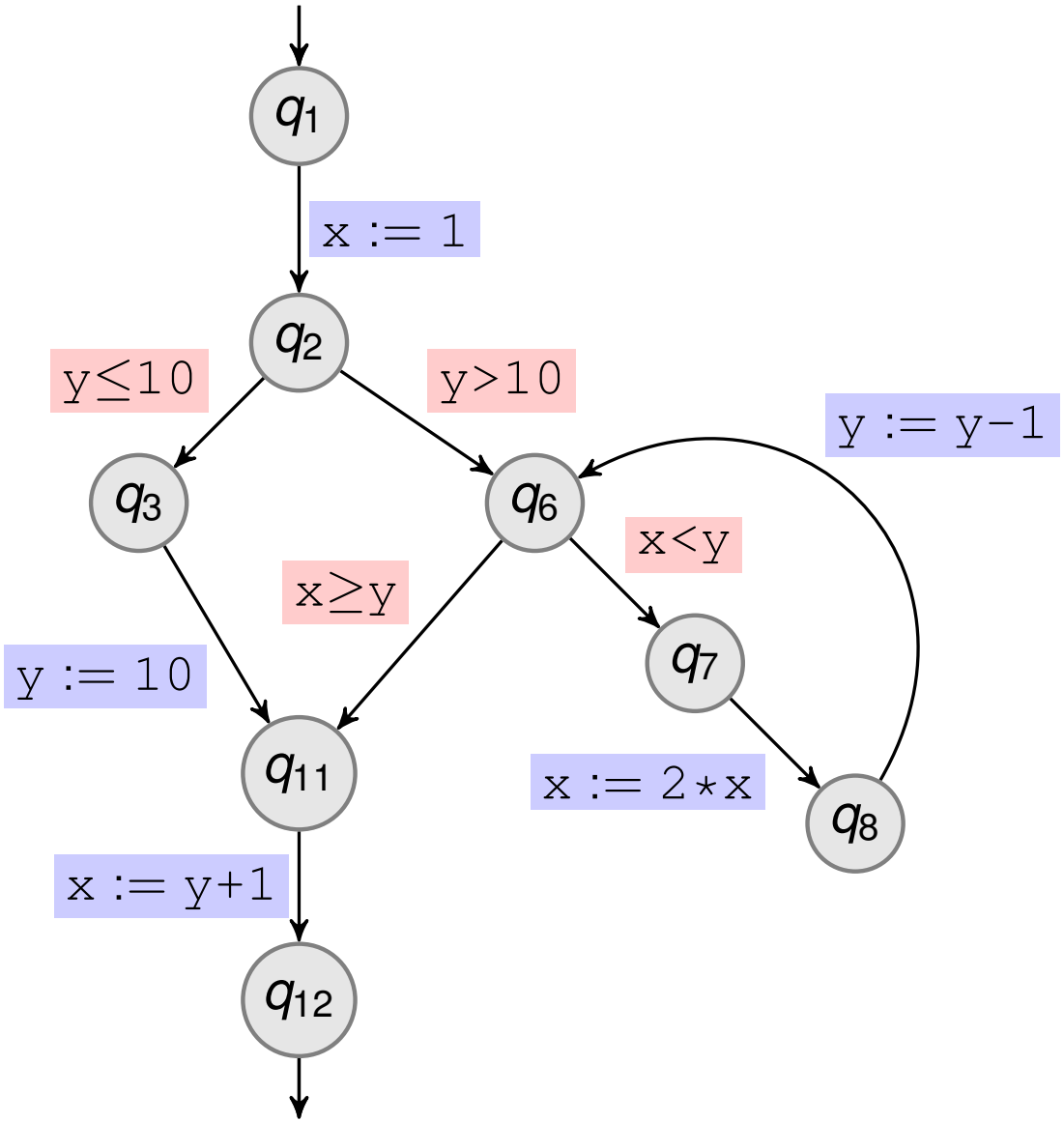
Range Analysis on Running Example



Recall that $\top = (-\infty, +\infty)$

	x		y	
q_1	$-\infty$	$+\infty$	$-\infty$	$+\infty$
q_2	1	1	$+\infty$	$+\infty$
q_3	1	1	$-\infty$	10
q_6	$\bot$	$\bot$	$\bot$	$\bot$
q_7	$\bot$	$\bot$	$\bot$	$\bot$
q_8	$\bot$	$\bot$	$\bot$	$\bot$
q_{11}	1	1	10	10
q_{12}	11	11	10	10

Range Analysis on Running Example

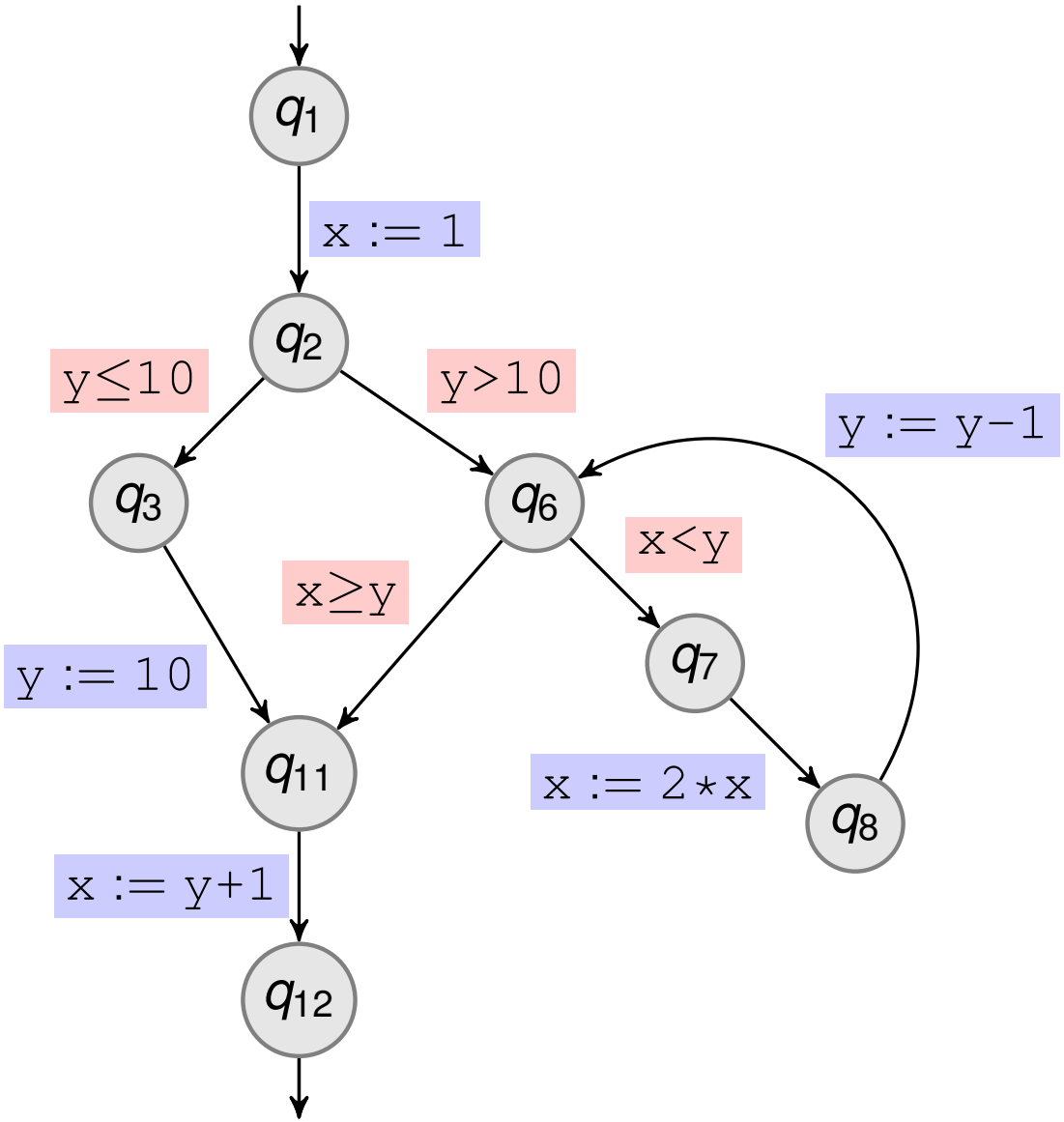


Recall that $\top = (-\infty, +\infty)$

	x		y	
q_1	$-\infty$	$+\infty$	$-\infty$	$+\infty$
q_2	1	1	$+\infty$	$+\infty$
q_3	1	1	$-\infty$	10
q_6	1	1	10	$+\infty$
q_7	$\bot$	$\bot$	$\bot$	$\bot$
q_8	$\bot$	$\bot$	$\bot$	$\bot$
q_{11}	1	1	10	10
q_{12}	11	11	10	10

Gained from guard y > 10

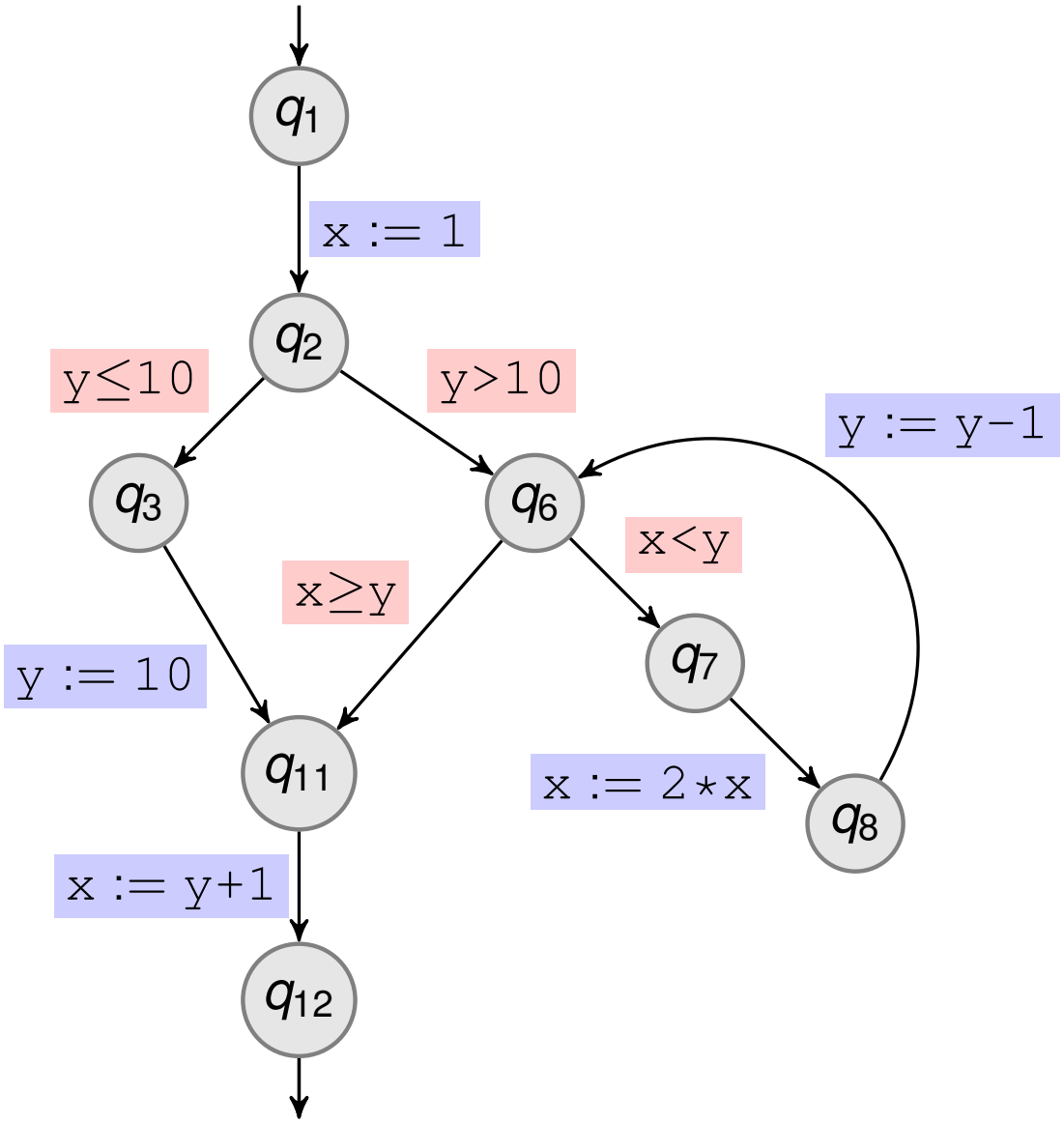
Range Analysis on Running Example



Recall that $\top = (-\infty, +\infty)$

	x		y	
q_1	$-\infty$	$+\infty$	$-\infty$	$+\infty$
q_2	1	1	$+\infty$	$+\infty$
q_3	1	1	$-\infty$	10
q_6	1	1	10	$+\infty$
q_7	$\bot$	$\bot$	$\bot$	$\bot$
q_8	$\bot$	$\bot$	$\bot$	$\bot$
q_{11}	1	1	10	10
q_{12}	11	11	10	10

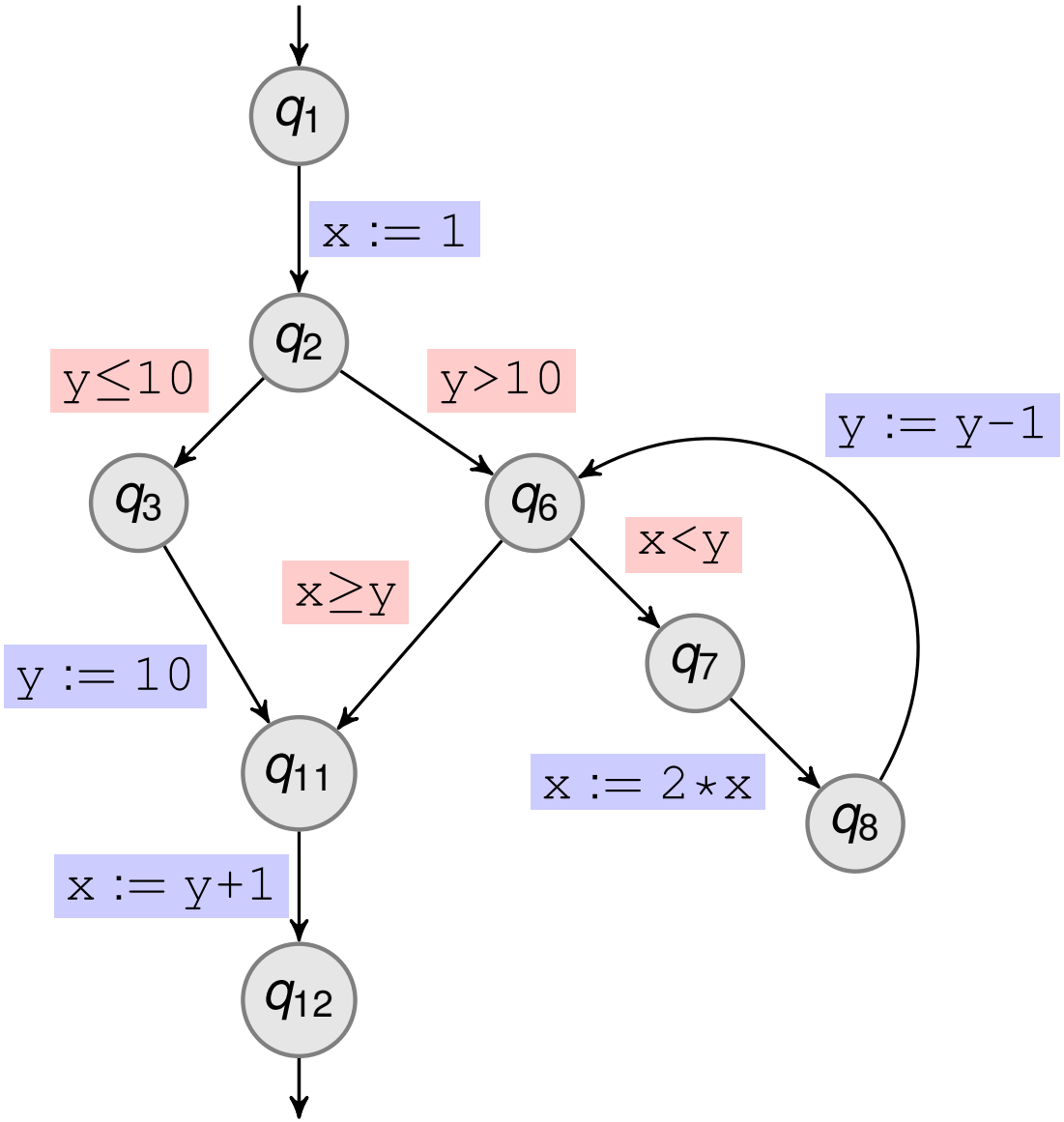
Range Analysis on Running Example



Recall that $\top = (-\infty, +\infty)$

	x		y	
q_1	$-\infty$	$+\infty$	$-\infty$	$+\infty$
q_2	1	1	$+\infty$	$+\infty$
q_3	1	1	$-\infty$	10
q_6	1	1	10	$+\infty$
q_7	1	1	10	$+\infty$
q_8	$\bot$	$\bot$	$\bot$	$\bot$
q_{11}	1	1	10	10
q_{12}	11	11	10	10

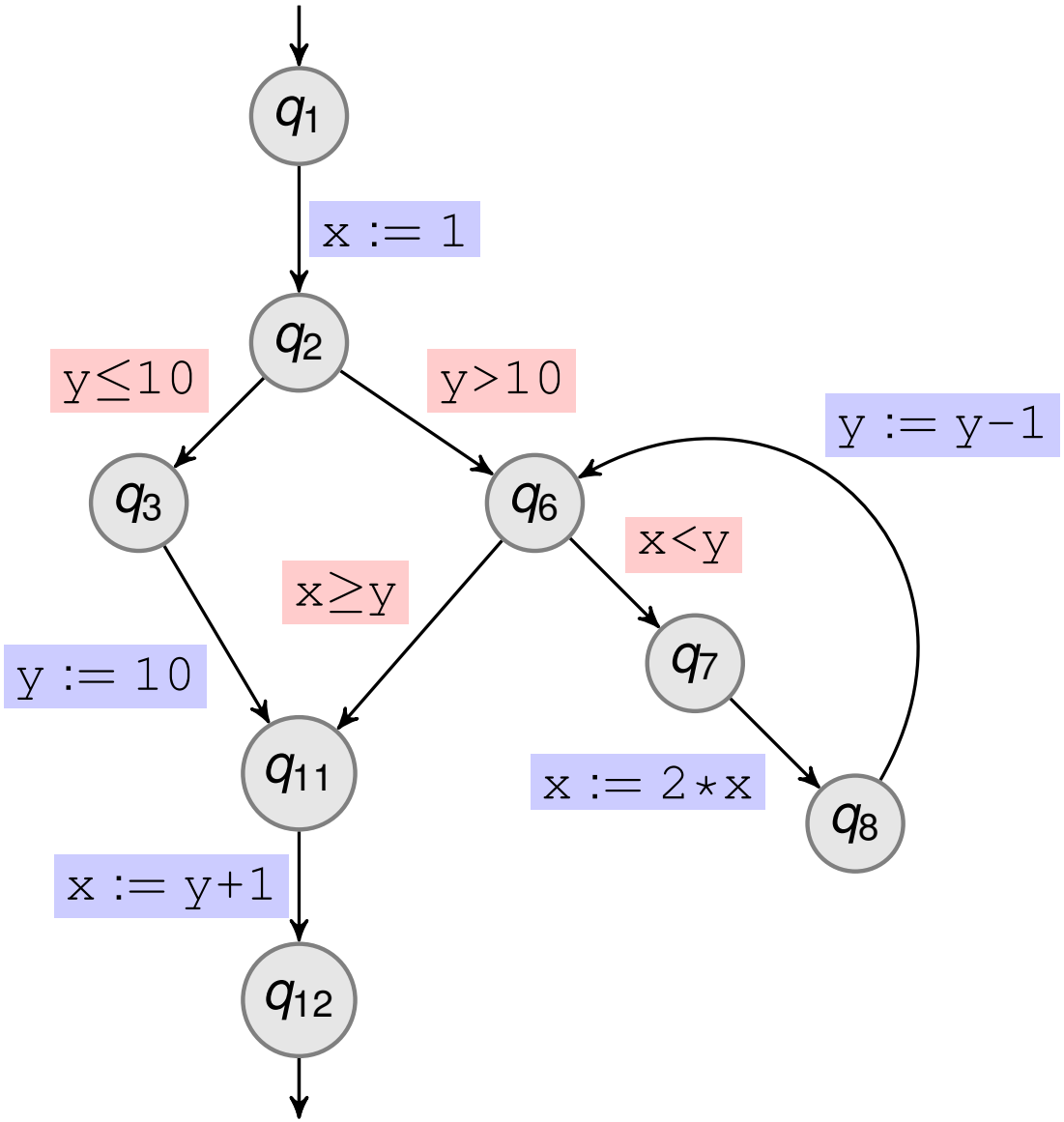
Range Analysis on Running Example



Recall that $\top = (-\infty, +\infty)$

	x		y	
q_1	$-\infty$	$+\infty$	$-\infty$	$+\infty$
q_2	1	1	$+\infty$	$+\infty$
q_3	1	1	$-\infty$	10
q_6	1	1	10	$+\infty$
q_7	1	1	10	$+\infty$
q_8	$\bot$	$\bot$	$\bot$	$\bot$
q_{11}	1	1	10	10
q_{12}	11	11	10	10

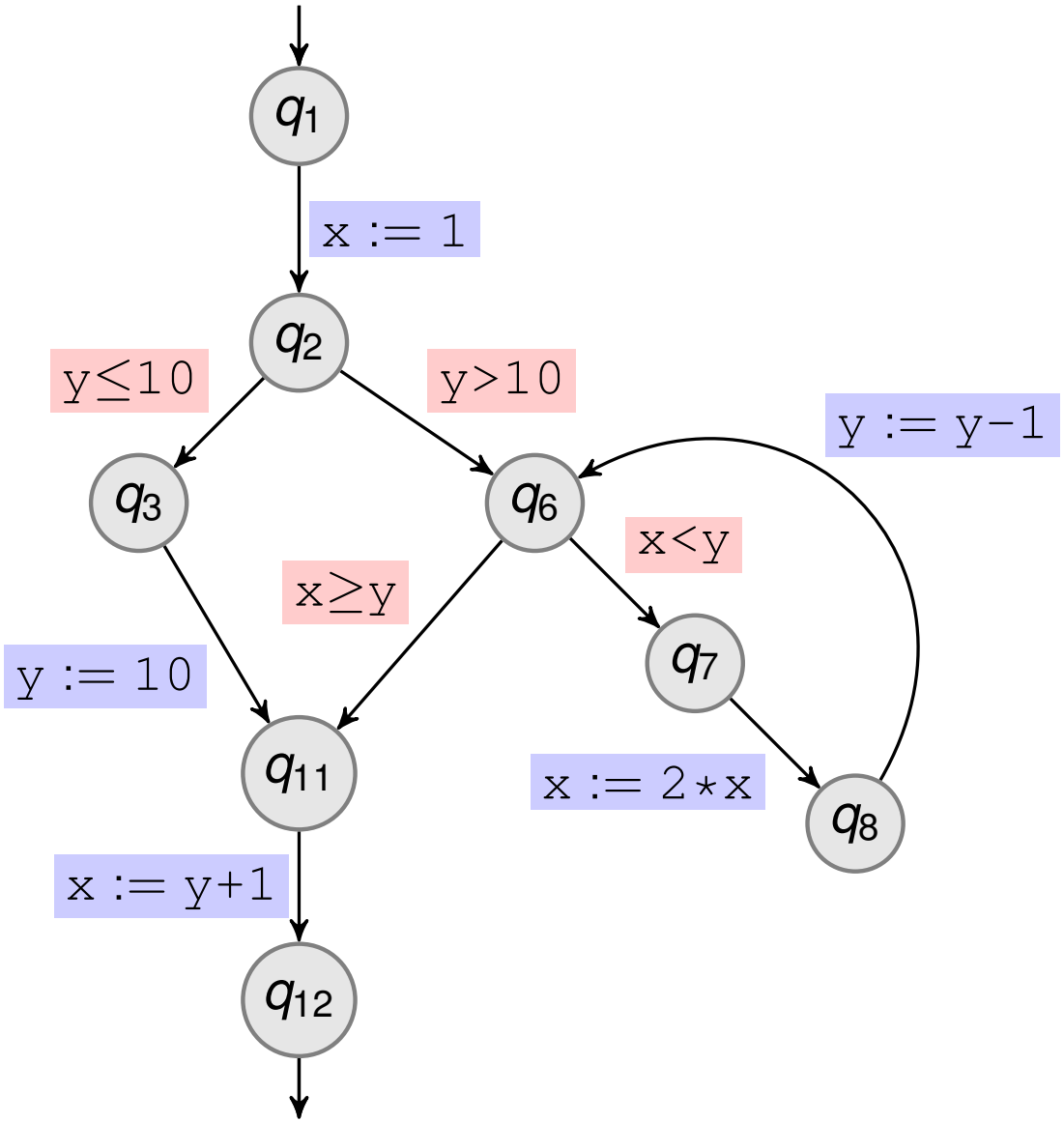
Range Analysis on Running Example



Recall that $\top = (-\infty, +\infty)$

	x		y	
q_1	$-\infty$	$+\infty$	$-\infty$	$+\infty$
q_2	1	1	$+\infty$	$+\infty$
q_3	1	1	$-\infty$	10
q_6	1	1	10	$+\infty$
q_7	1	1	10	$+\infty$
q_8	2	2	10	$+\infty$
q_{11}	1	1	10	10
q_{12}	11	11	10	10

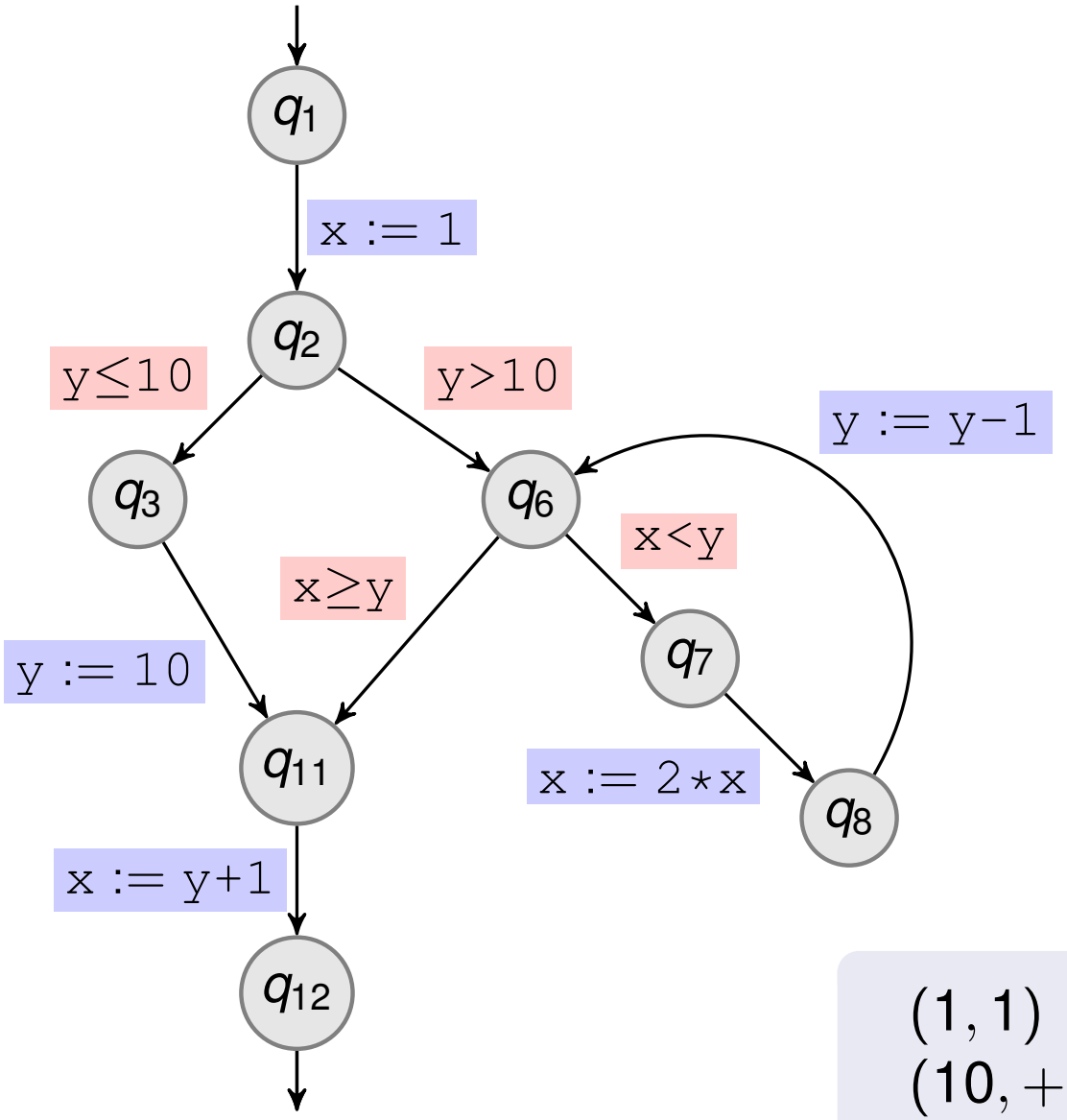
Range Analysis on Running Example



Recall that $\top = (-\infty, +\infty)$

	x		y	
q_1	$-\infty$	$+\infty$	$-\infty$	$+\infty$
q_2	1	1	$+\infty$	$+\infty$
q_3	1	1	$-\infty$	10
q_6	1	1	10	$+\infty$
q_7	1	1	10	$+\infty$
q_8	2	2	10	$+\infty$
q_{11}	1	1	10	10
q_{12}	11	11	10	10

Range Analysis on Running Example

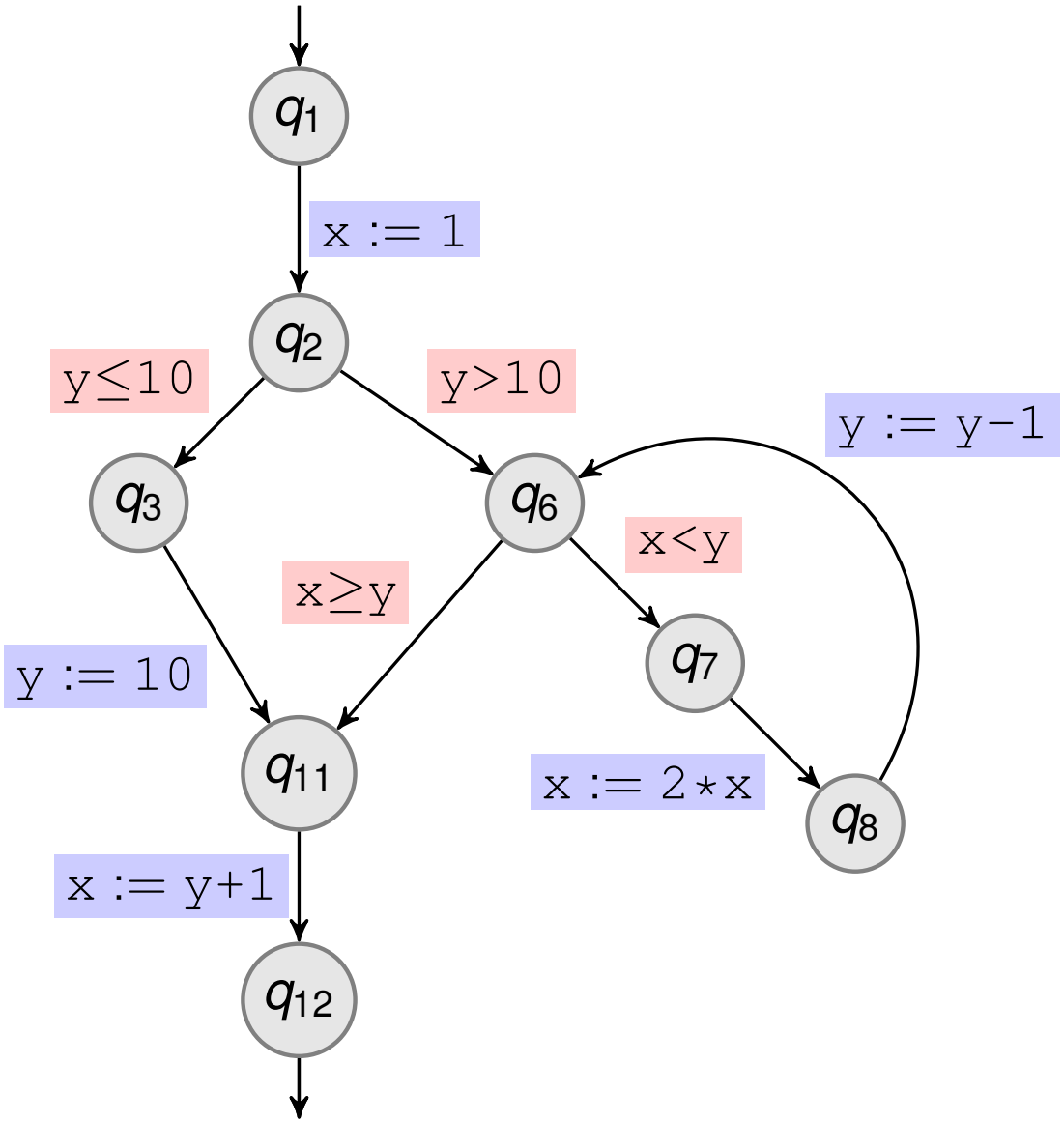


Recall that $\top = (-\infty, +\infty)$

	x		y	
q_1	$-\infty$	$+\infty$	$-\infty$	$+\infty$
q_2	1	1	$+\infty$	$+\infty$
q_3	1	1	$-\infty$	10
q_6	1	2	9	$+\infty$
q_7	1	1	10	$+\infty$
q_8	2	2	10	$+\infty$
q_{11}	1	1	10	10
q_{12}	11	11	10	10

$$(1,1) \sqcup (2,2) = (1,2)$$
$$(10,+\infty) \sqcup (9,+\infty) = (9,+\infty)$$

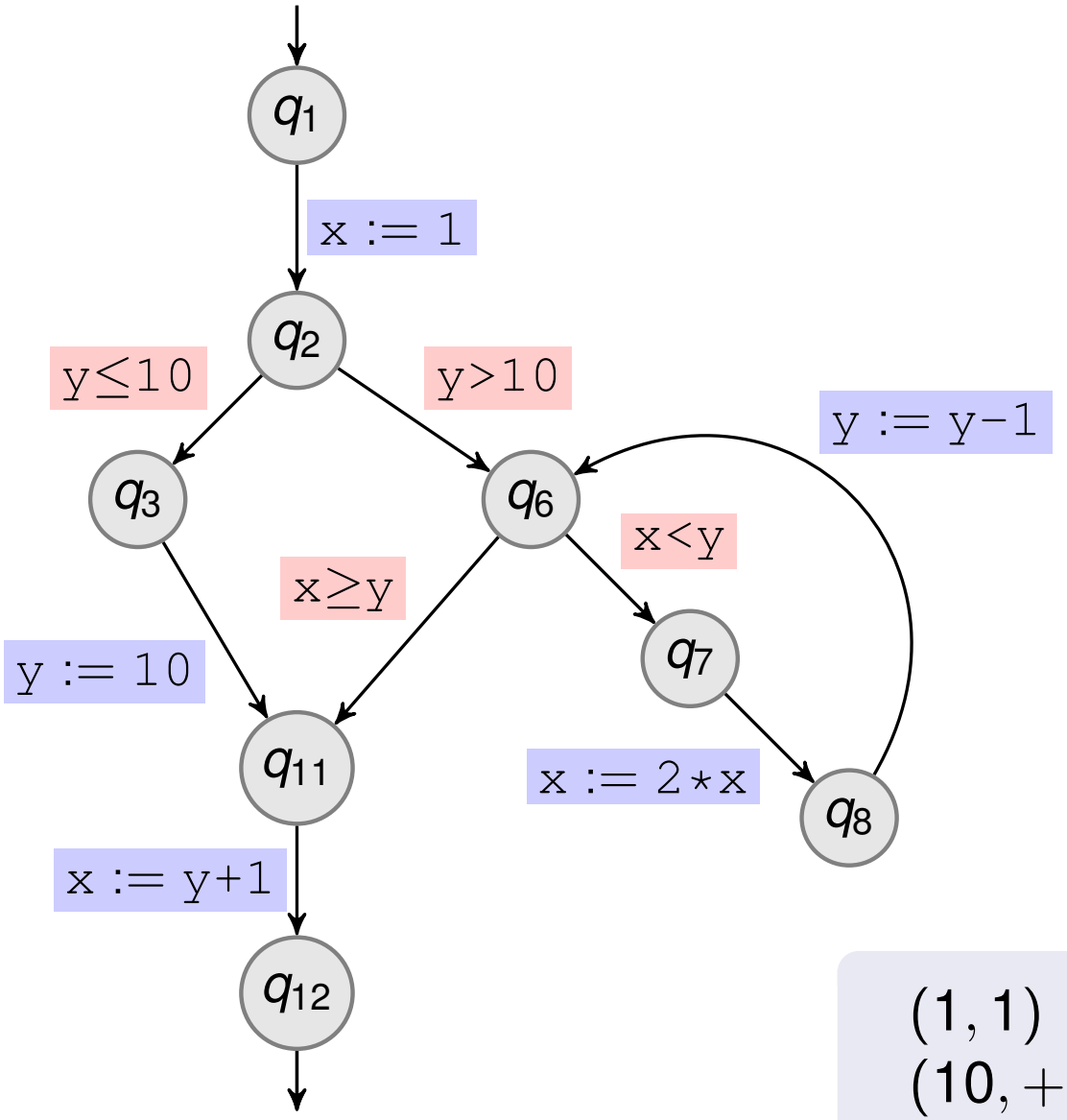
Range Analysis on Running Example



Recall that $\top = (-\infty, +\infty)$

	x		y	
q_1	$-\infty$	$+\infty$	$-\infty$	$+\infty$
q_2	1	1	$+\infty$	$+\infty$
q_3	1	1	$-\infty$	10
q_6	1	2	9	$+\infty$
q_7	1	1	10	$+\infty$
q_8	2	2	10	$+\infty$
q_{11}	1	1	10	10
q_{12}	11	11	10	10

Range Analysis on Running Example

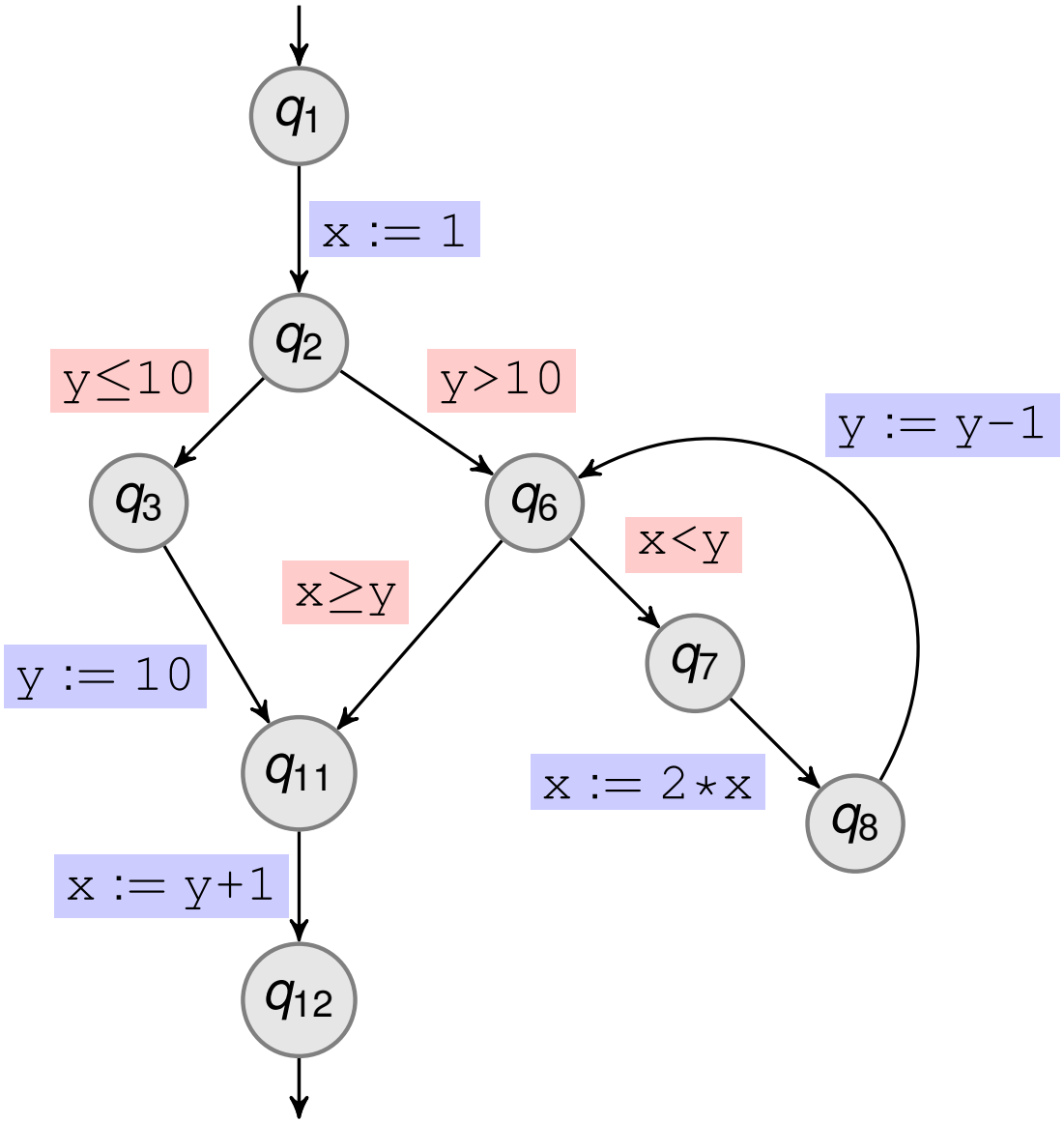


Recall that $\top = (-\infty, +\infty)$

	x		y	
q_1	$-\infty$	$+\infty$	$-\infty$	$+\infty$
q_2	1	1	$+\infty$	$+\infty$
q_3	1	1	$-\infty$	10
q_6	1	2	9	$+\infty$
q_7	1	2	9	$+\infty$
q_8	2	2	10	$+\infty$
q_{11}	1	1	10	10
q_{12}	11	11	10	10

$$(1,1) \sqcup (1,2) = (1,2)$$
$$(10,+\infty) \sqcup (9,+\infty) = (9,+\infty)$$

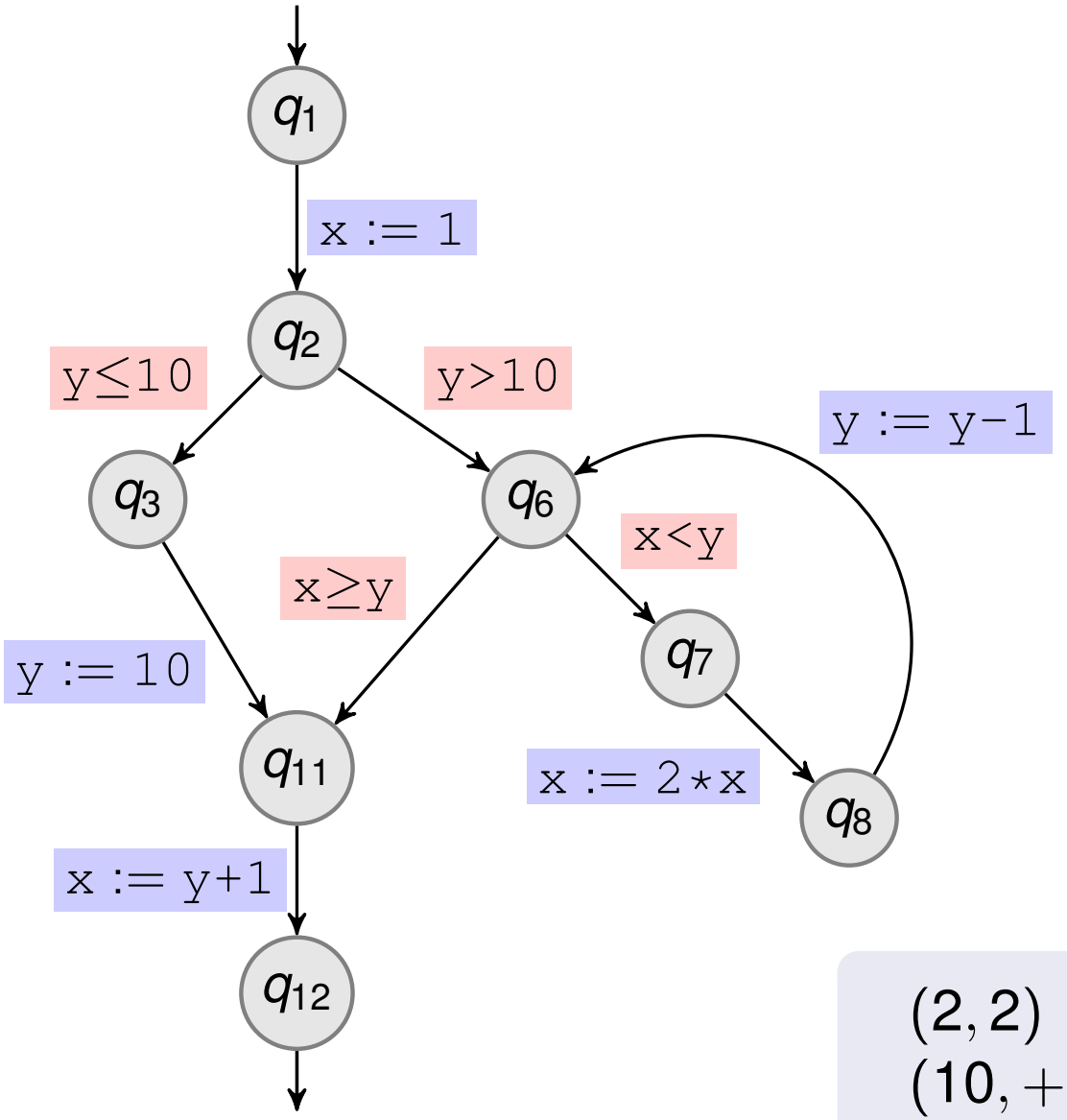
Range Analysis on Running Example



Recall that $\top = (-\infty, +\infty)$

	x		y	
q_1	$-\infty$	$+\infty$	$-\infty$	$+\infty$
q_2	1	1	$+\infty$	$+\infty$
q_3	1	1	$-\infty$	10
q_6	1	2	9	$+\infty$
q_7	1	2	9	$+\infty$
q_8	2	2	10	$+\infty$
q_{11}	1	1	10	10
q_{12}	11	11	10	10

Range Analysis on Running Example

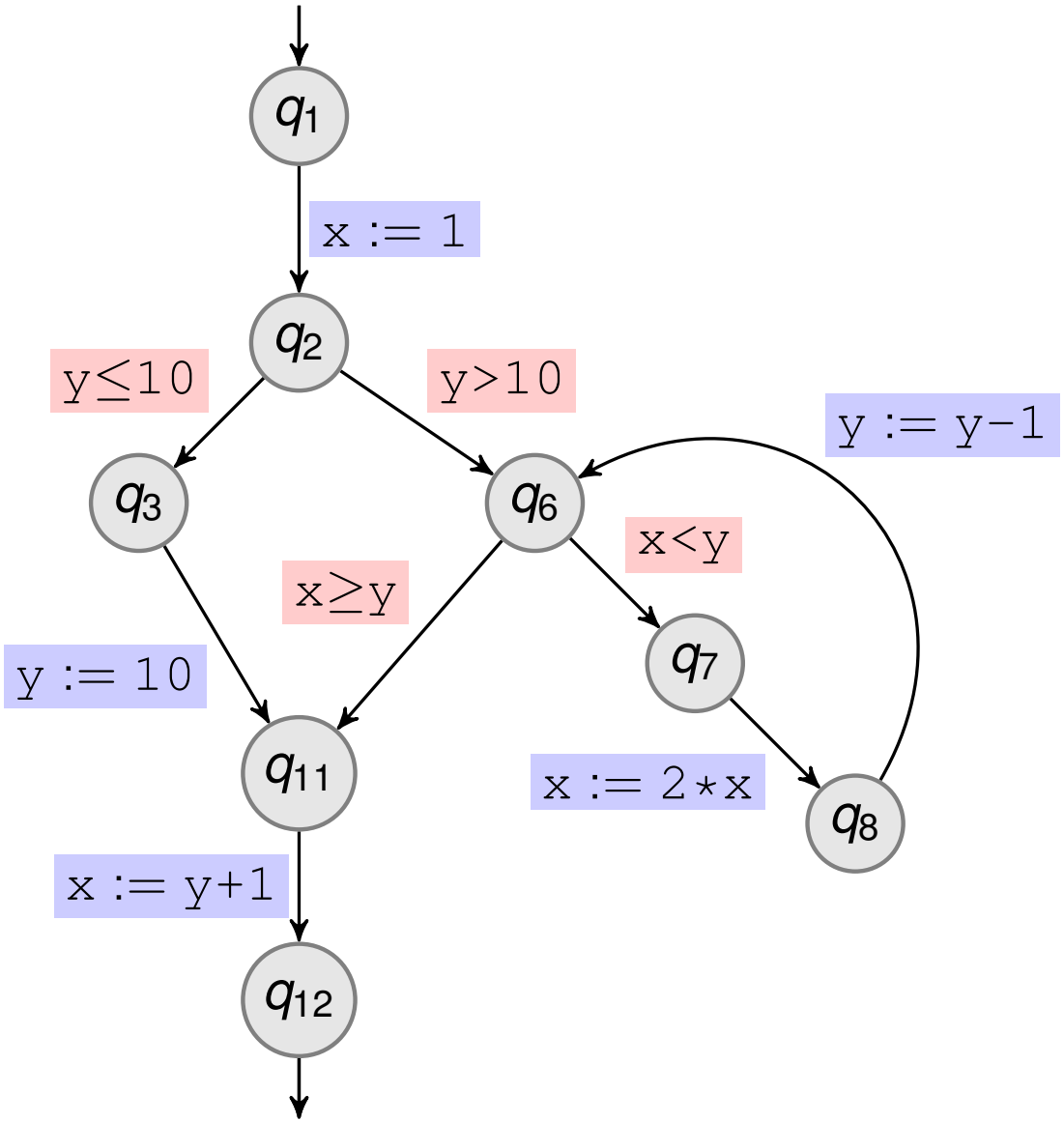


Recall that $\top = (-\infty, +\infty)$

	x		y	
q_1	$-\infty$	$+\infty$	$-\infty$	$+\infty$
q_2	1	1	$+\infty$	$+\infty$
q_3	1	1	$-\infty$	10
q_6	1	2	9	$+\infty$
q_7	1	2	9	$+\infty$
q_8	2	4	9	$+\infty$
q_{11}	1	1	10	10
q_{12}	11	11	10	10

$$(2,2) \sqcup (2,4) = (2,4)$$
$$(10,+\infty) \sqcup (9,+\infty) = (9,+\infty)$$

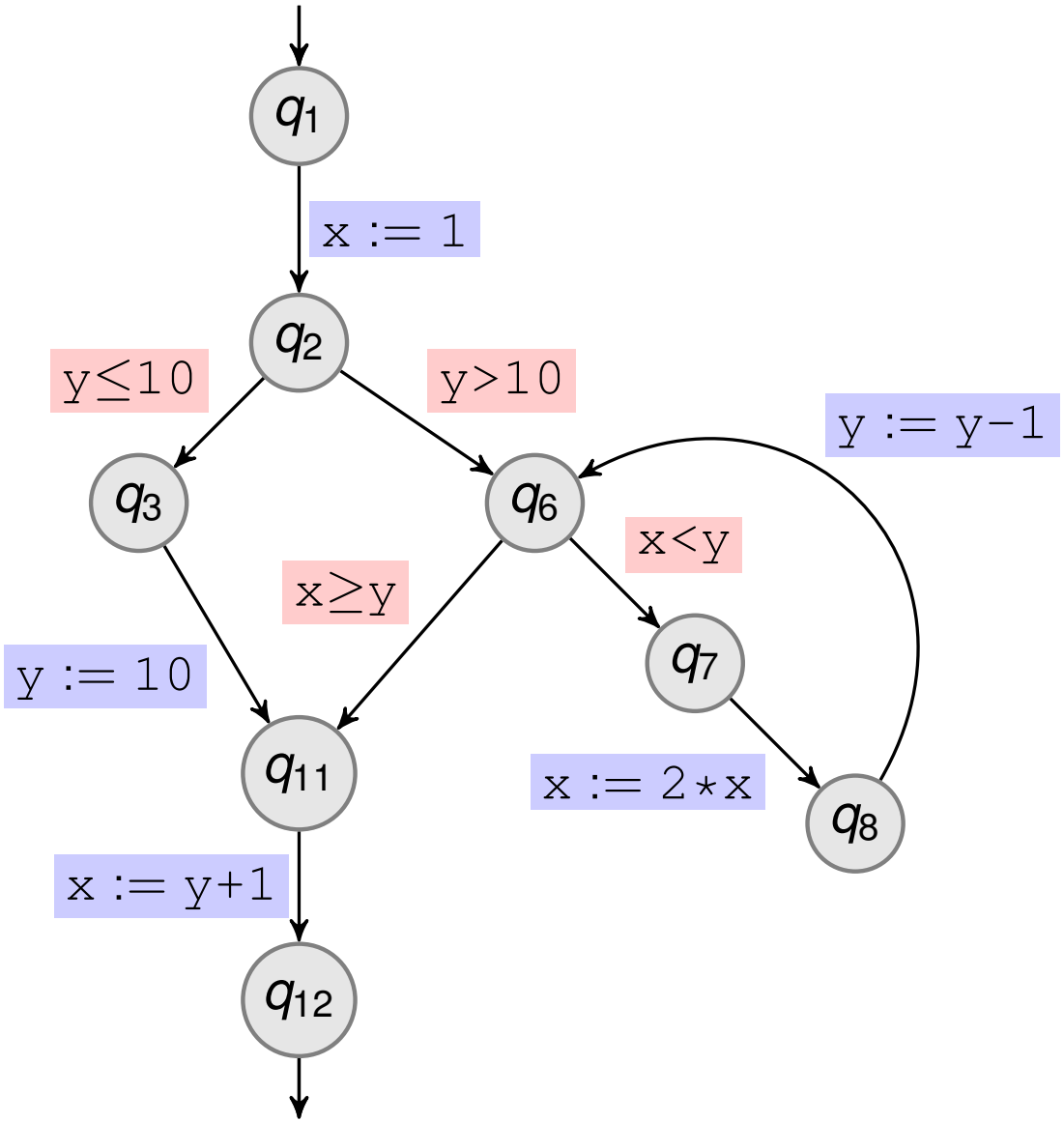
Range Analysis on Running Example



Recall that $\top = (-\infty, +\infty)$

	x		y	
q_1	$-\infty$	$+\infty$	$-\infty$	$+\infty$
q_2	1	1	$+\infty$	$+\infty$
q_3	1	1	$-\infty$	10
q_6	1	2	9	$+\infty$
q_7	1	2	9	$+\infty$
q_8	2	4	9	$+\infty$
q_{11}	1	1	10	10
q_{12}	11	11	10	10

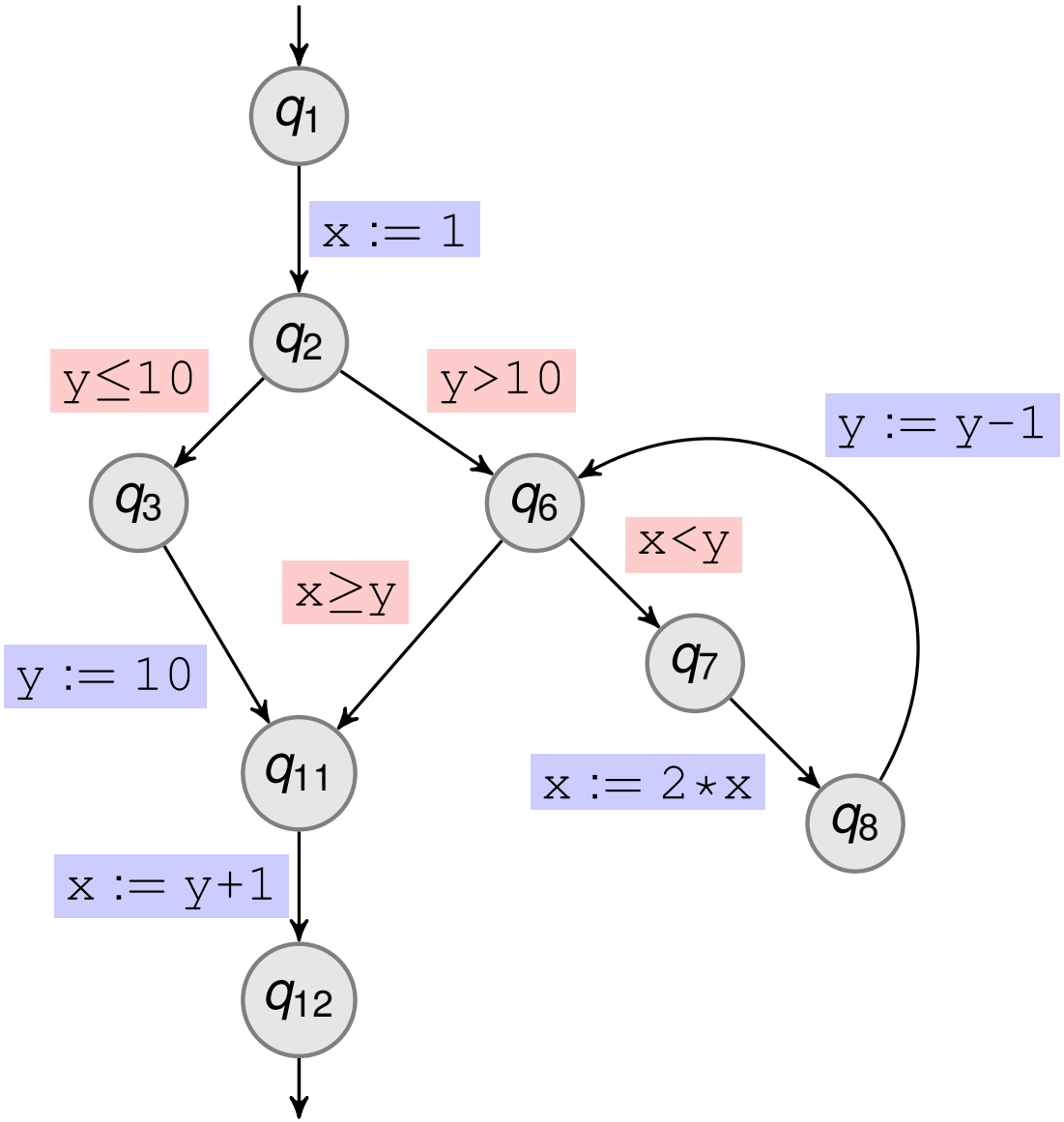
Range Analysis on Running Example



Recall that $\top = (-\infty, +\infty)$

	x		y	
q_1	$-\infty$	$+\infty$	$-\infty$	$+\infty$
q_2	1	1	$+\infty$	$+\infty$
q_3	1	1	$-\infty$	10
q_6	1	4	8	$+\infty$
q_7	1	2	9	$+\infty$
q_8	2	4	9	$+\infty$
q_{11}	1	1	10	10
q_{12}	11	11	10	10

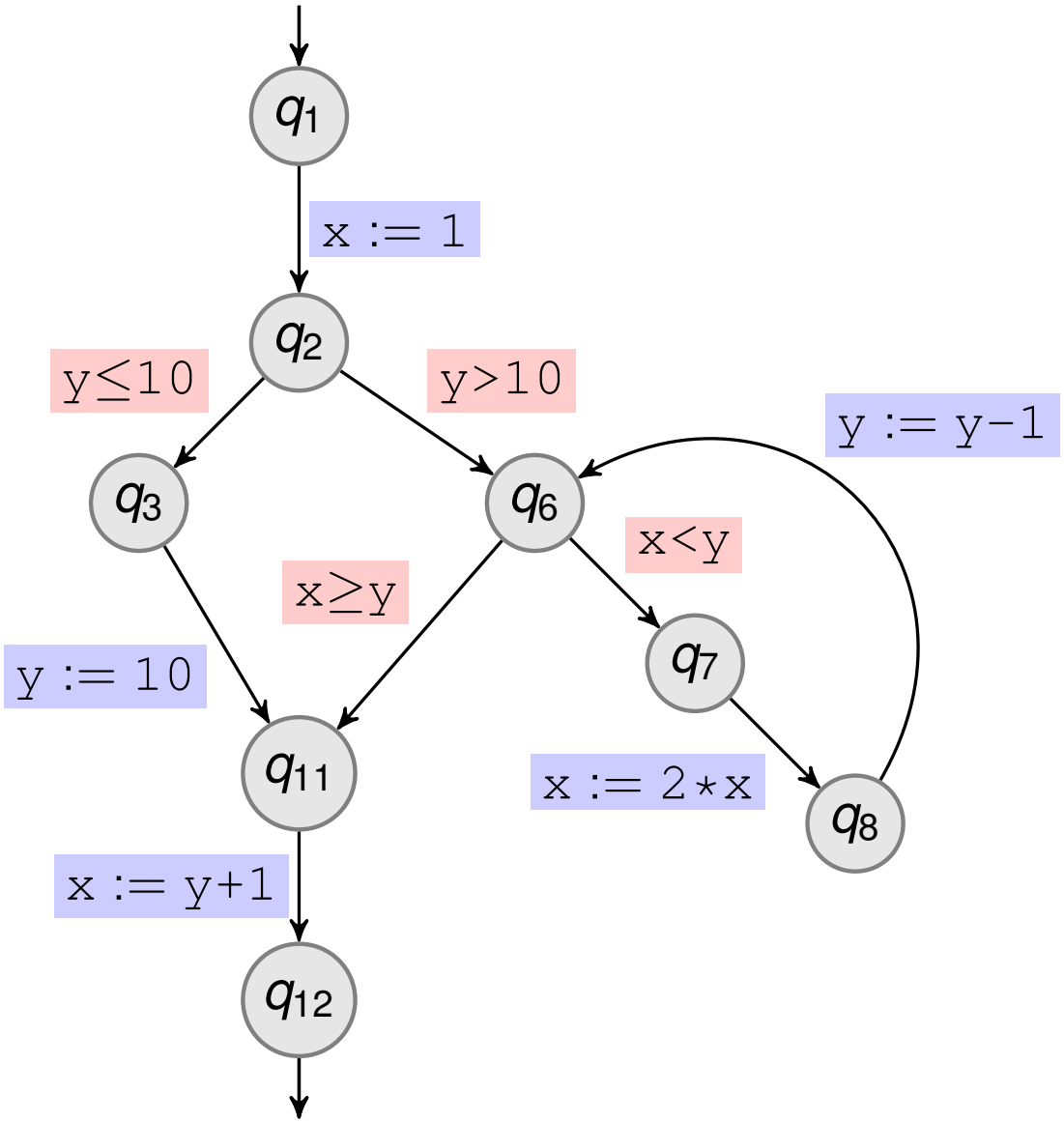
Range Analysis on Running Example



Recall that $\top = (-\infty, +\infty)$

	x		y	
q_1	$-\infty$	$+\infty$	$-\infty$	$+\infty$
q_2	1	1	$+\infty$	$+\infty$
q_3	1	1	$-\infty$	10
q_6	1	4	8	$+\infty$
q_7	1	2	9	$+\infty$
q_8	2	4	9	$+\infty$
q_{11}	1	1	10	10
q_{12}	11	11	10	10

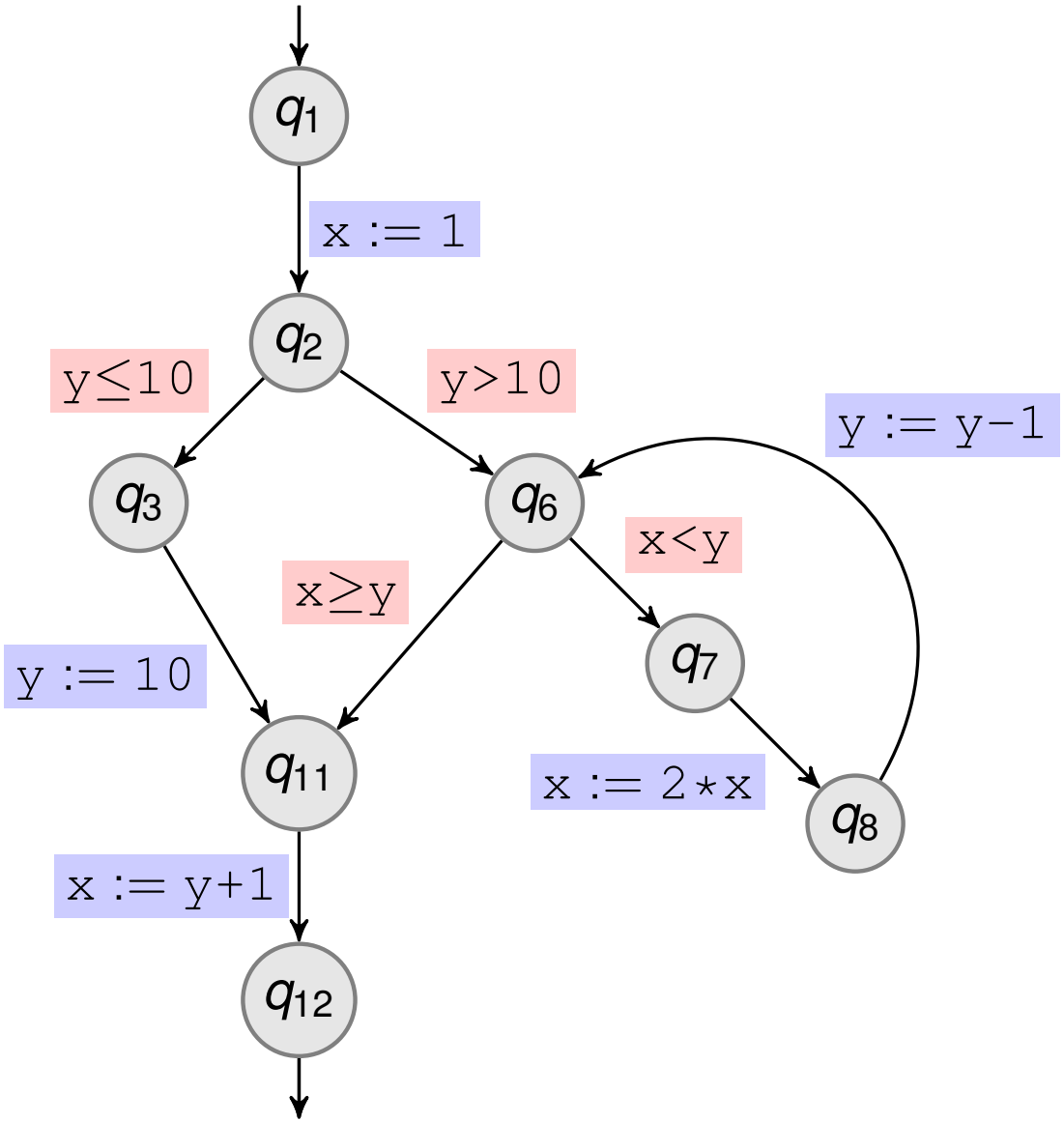
Range Analysis on Running Example



Recall that $\top = (-\infty, +\infty)$

	x		y	
q_1	$-\infty$	$+\infty$	$-\infty$	$+\infty$
q_2	1	1	$+\infty$	$+\infty$
q_3	1	1	$-\infty$	10
q_6	1	4	8	$+\infty$
q_7	1	4	8	$+\infty$
q_8	2	4	9	$+\infty$
q_{11}	1	1	10	10
q_{12}	11	11	10	10

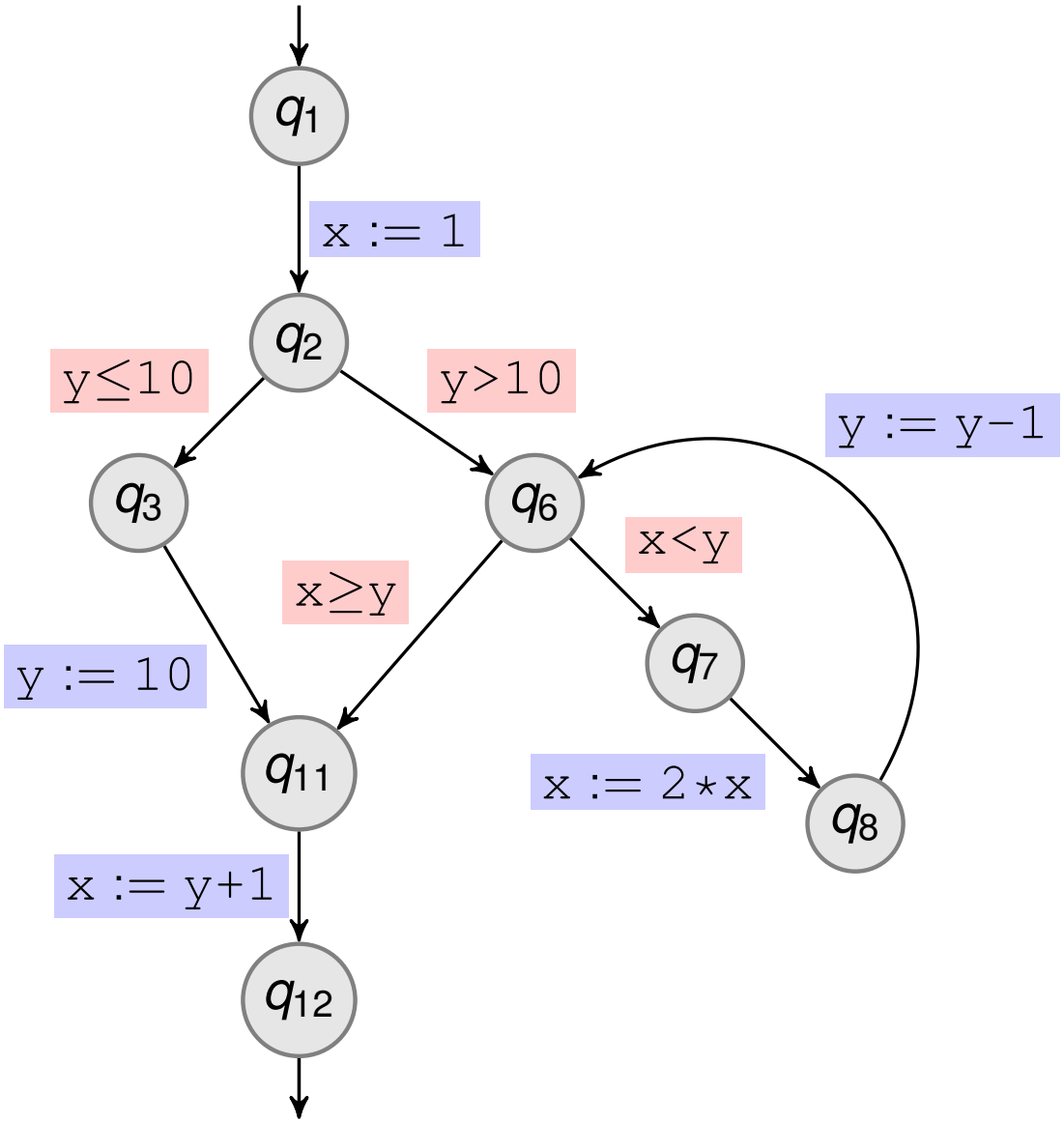
Range Analysis on Running Example



Recall that $\top = (-\infty, +\infty)$

	x		y	
q_1	$-\infty$	$+\infty$	$-\infty$	$+\infty$
q_2	1	1	$+\infty$	$+\infty$
q_3	1	1	$-\infty$	10
q_6	1	4	8	$+\infty$
q_7	1	4	8	$+\infty$
q_8	2	4	9	$+\infty$
q_{11}	1	1	10	10
q_{12}	11	11	10	10

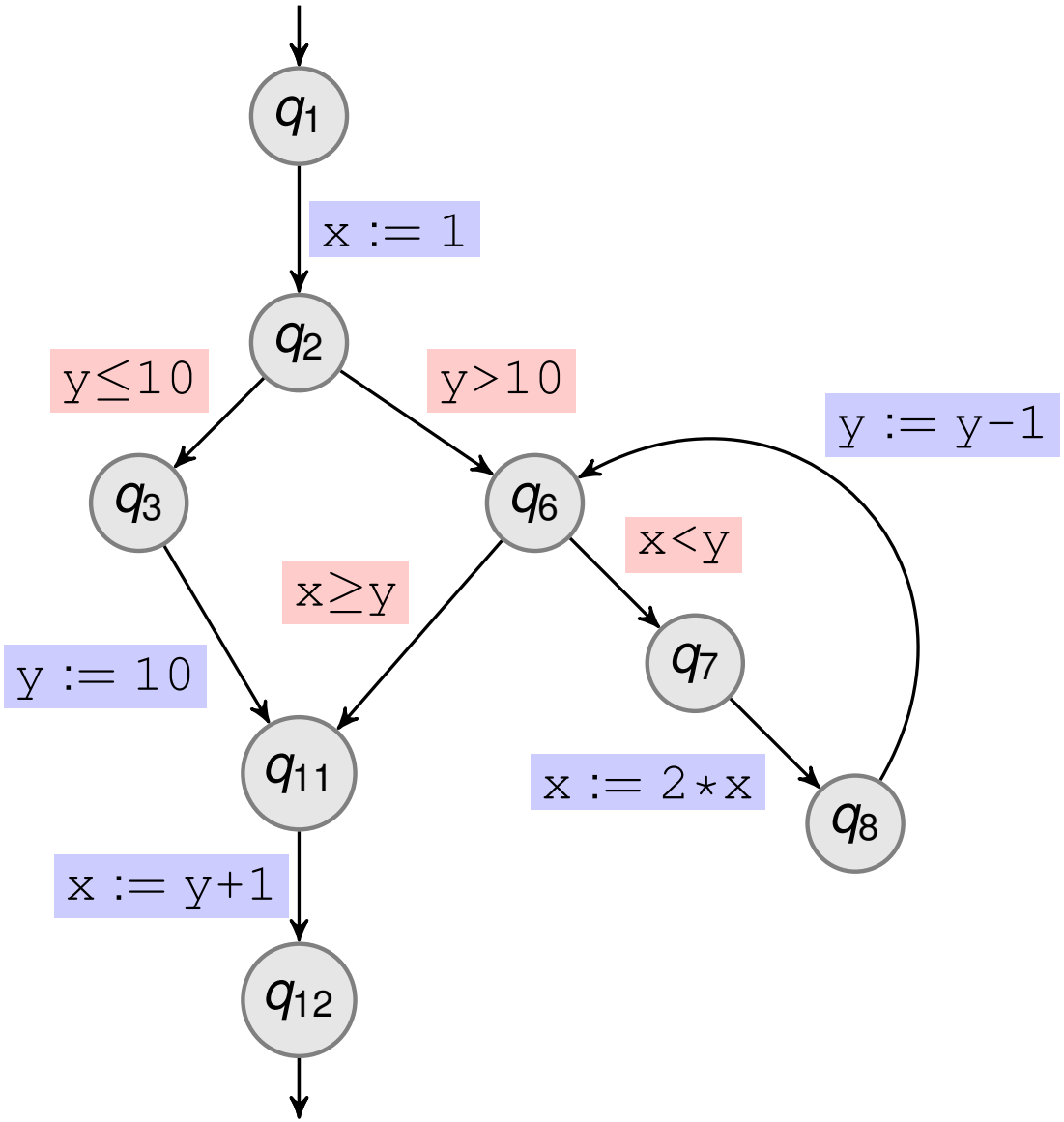
Range Analysis on Running Example



Recall that $\top = (-\infty, +\infty)$

	x		y	
q_1	$-\infty$	$+\infty$	$-\infty$	$+\infty$
q_2	1	1	$+\infty$	$+\infty$
q_3	1	1	$-\infty$	10
q_6	1	4	8	$+\infty$
q_7	1	4	8	$+\infty$
q_8	2	8	8	$+\infty$
q_{11}	1	1	10	10
q_{12}	11	11	10	10

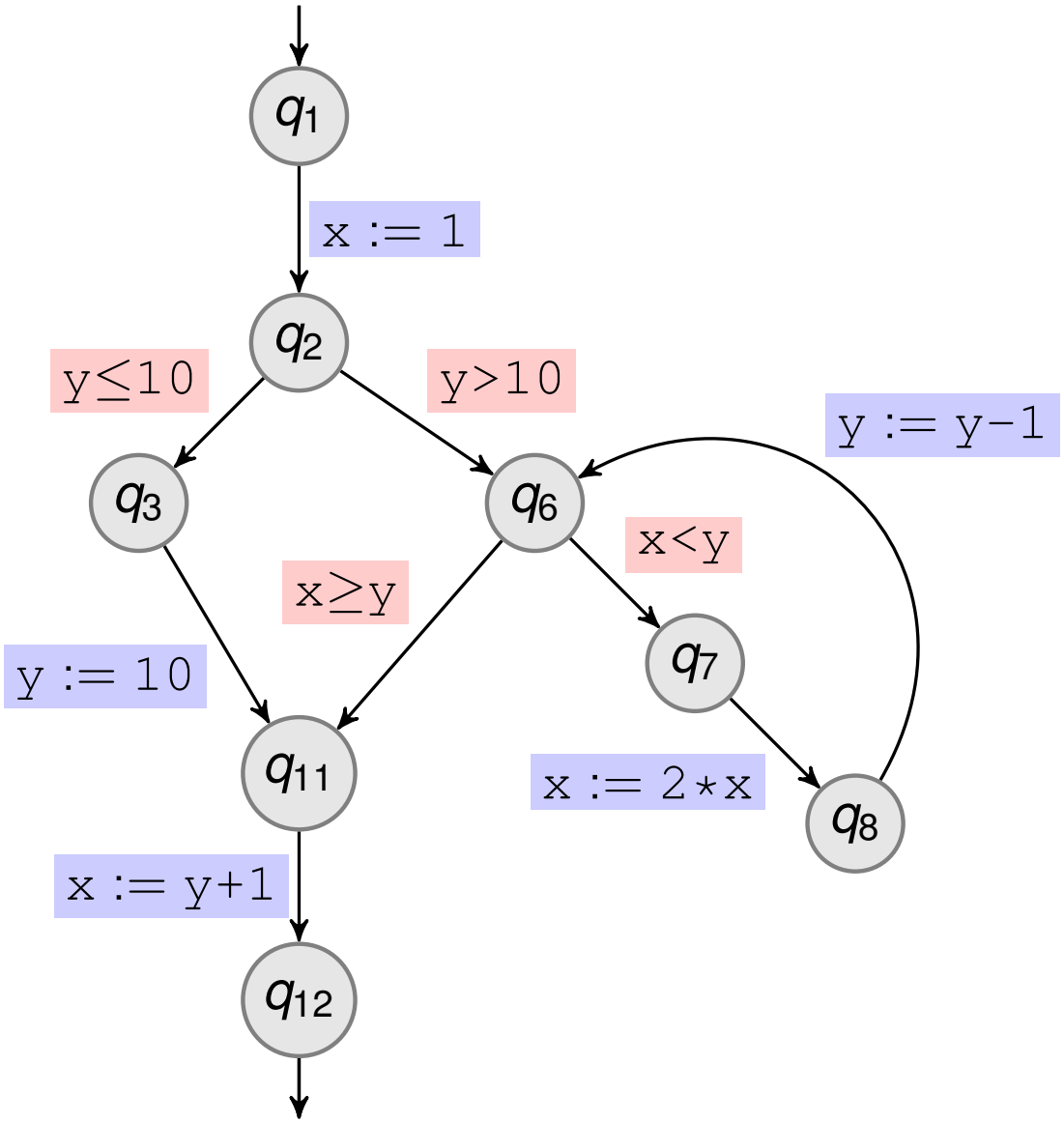
Range Analysis on Running Example



Recall that $\top = (-\infty, +\infty)$

	x		y	
q_1	$-\infty$	$+\infty$	$-\infty$	$+\infty$
q_2	1	1	$+\infty$	$+\infty$
q_3	1	1	$-\infty$	10
q_6	1	4	8	$+\infty$
q_7	1	4	8	$+\infty$
q_8	2	8	8	$+\infty$
q_{11}	1	1	10	10
q_{12}	11	11	10	10

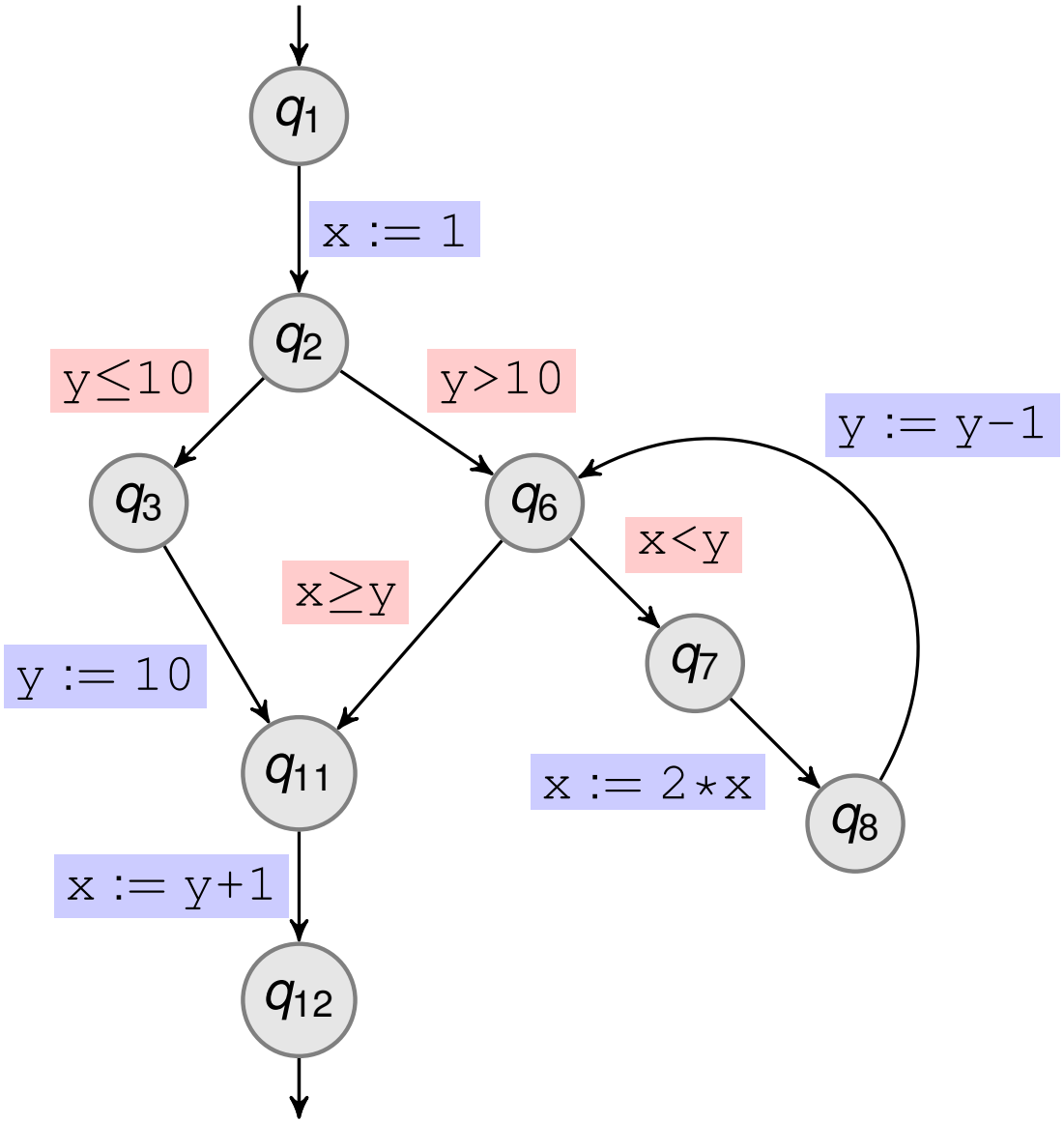
Range Analysis on Running Example



Recall that $\top = (-\infty, +\infty)$

	x		y	
q_1	$-\infty$	$+\infty$	$-\infty$	$+\infty$
q_2	1	1	$+\infty$	$+\infty$
q_3	1	1	$-\infty$	10
q_6	1	8	7	$+\infty$
q_7	1	4	8	$+\infty$
q_8	2	8	8	$+\infty$
q_{11}	1	1	10	10
q_{12}	11	11	10	10

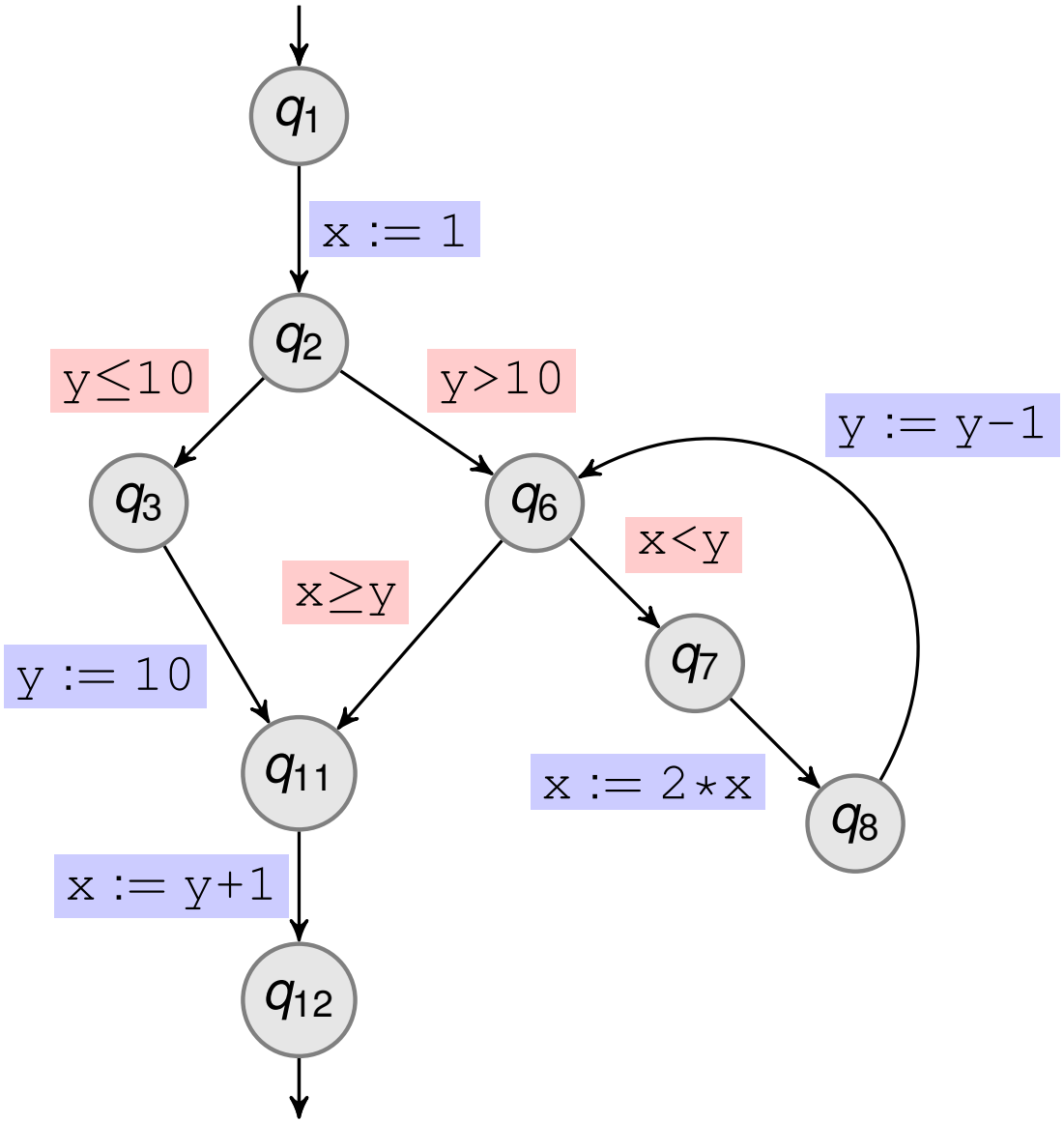
Range Analysis on Running Example



Recall that $\top = (-\infty, +\infty)$

	x		y	
q_1	$-\infty$	$+\infty$	$-\infty$	$+\infty$
q_2	1	1	$+\infty$	$+\infty$
q_3	1	1	$-\infty$	10
q_6	1	8	7	$+\infty$
q_7	1	4	8	$+\infty$
q_8	2	8	8	$+\infty$
q_{11}	1	1	10	10
q_{12}	11	11	10	10

Range Analysis on Running Example

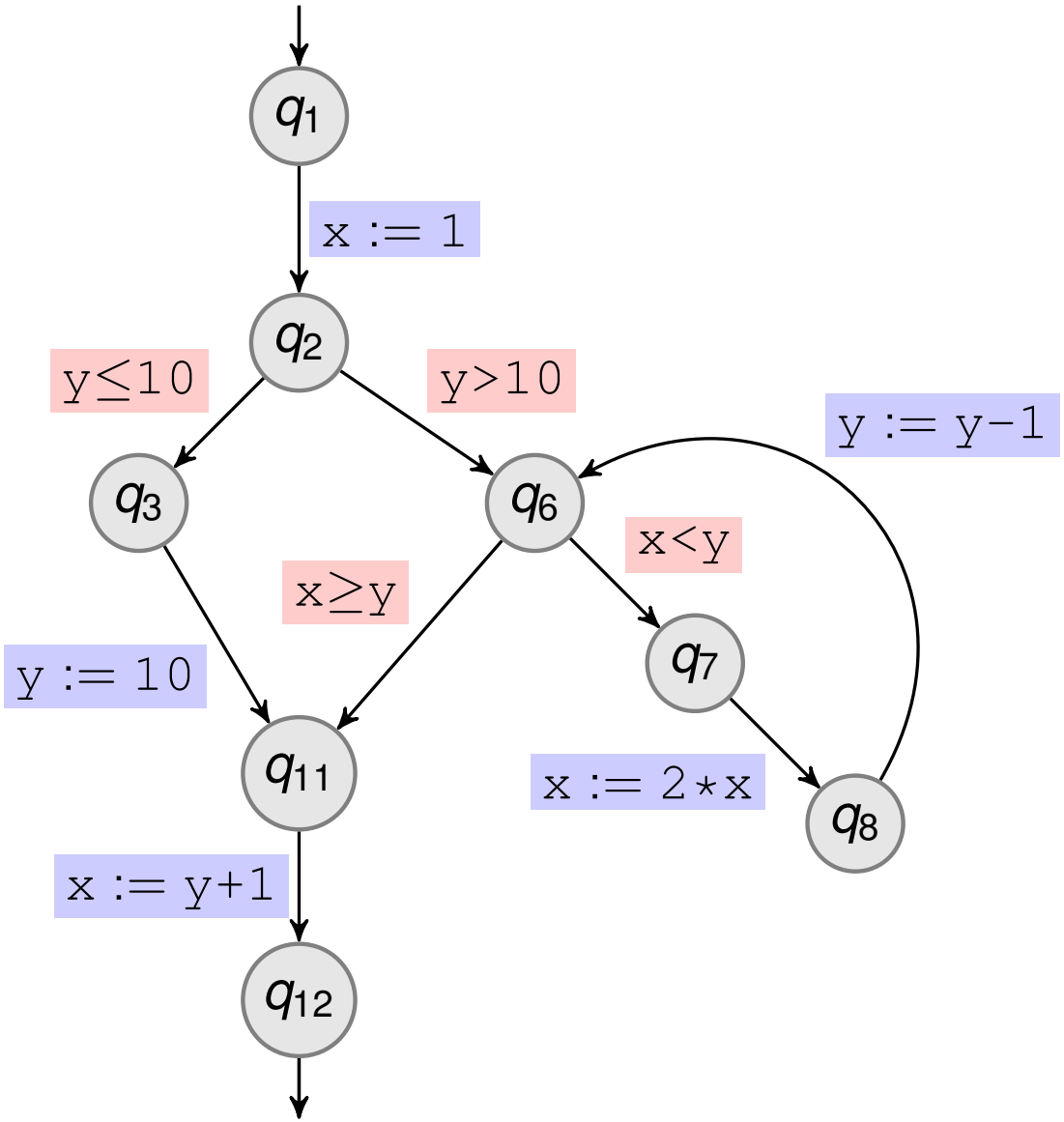


Recall that $\top = (-\infty, +\infty)$

	x		y	
q_1	$-\infty$	$+\infty$	$-\infty$	$+\infty$
q_2	1	1	$+\infty$	$+\infty$
q_3	1	1	$-\infty$	10
q_6	1	8	7	$+\infty$
q_7	1	8	7	$+\infty$
q_8	2	8	8	$+\infty$
q_{11}	1	1	10	10
q_{12}	11	11	10	10

Nothing to be gained from guard $x < y$

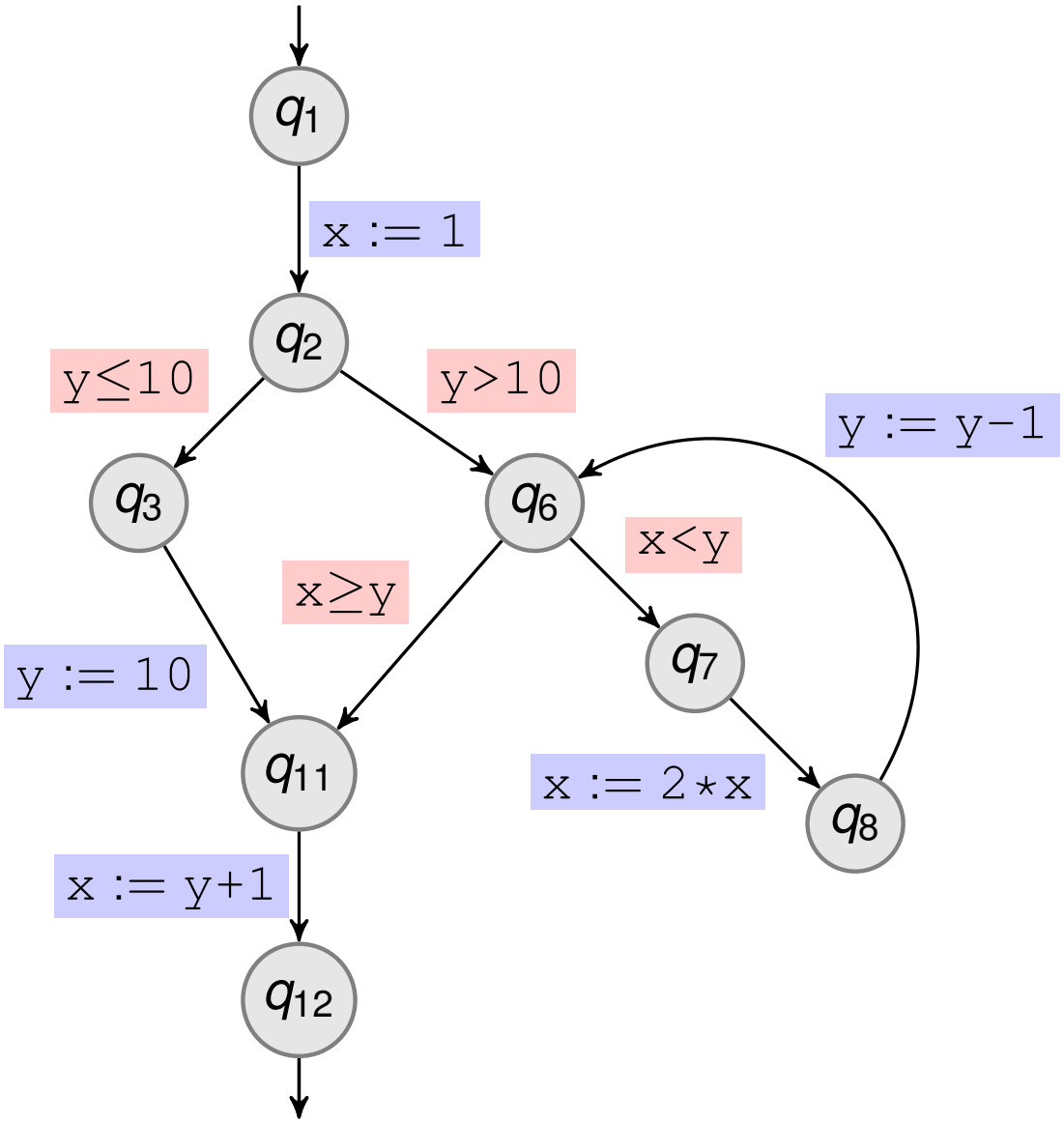
Range Analysis on Running Example



Recall that $\top = (-\infty, +\infty)$

	x		y	
q_1	$-\infty$	$+\infty$	$-\infty$	$+\infty$
q_2	1	1	$+\infty$	$+\infty$
q_3	1	1	$-\infty$	10
q_6	1	8	7	$+\infty$
q_7	1	8	7	$+\infty$
q_8	2	8	8	$+\infty$
q_{11}	1	1	10	10
q_{12}	11	11	10	10

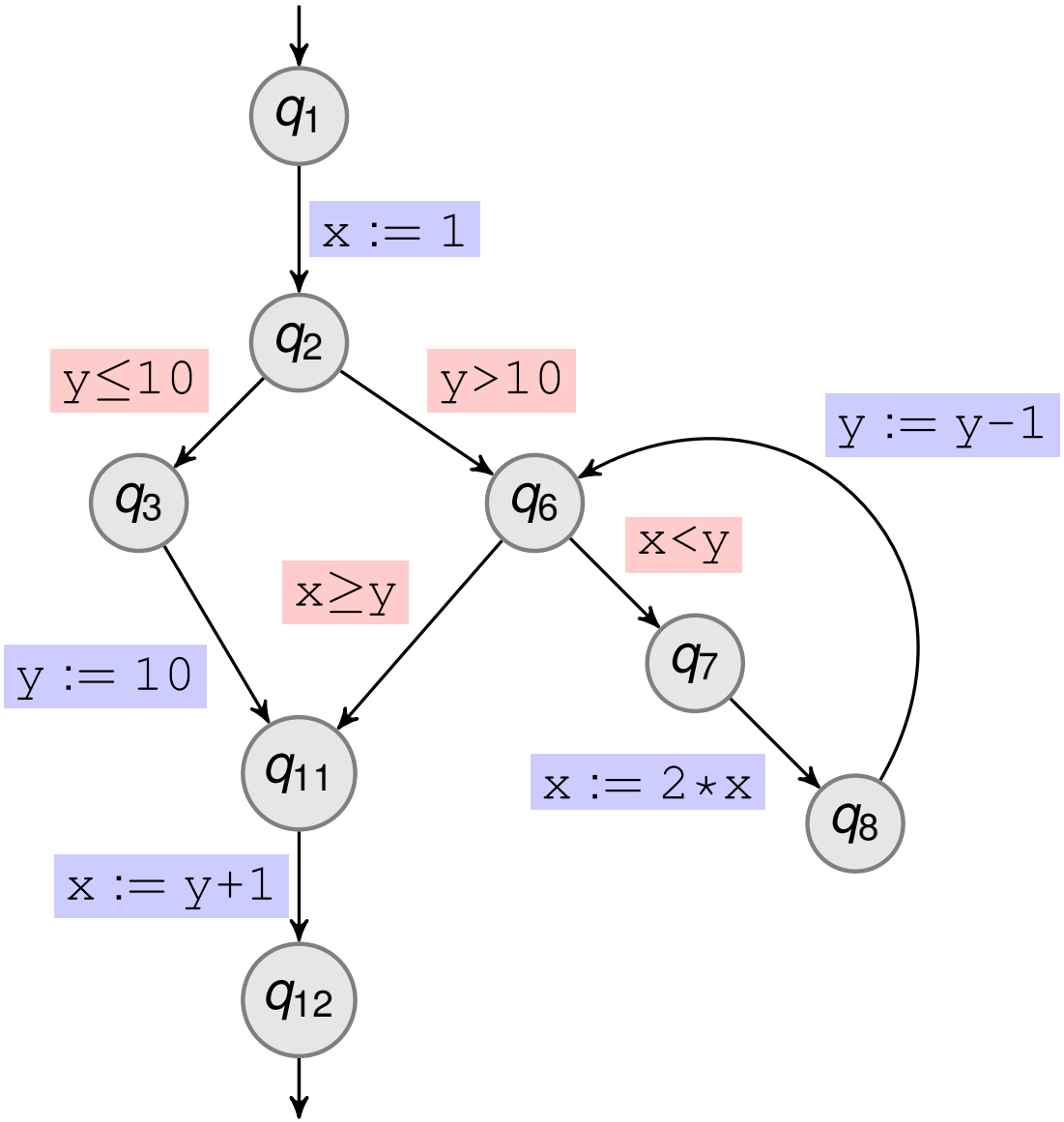
Range Analysis on Running Example



Recall that $\top = (-\infty, +\infty)$

	x		y	
q_1	$-\infty$	$+\infty$	$-\infty$	$+\infty$
q_2	1	1	$+\infty$	$+\infty$
q_3	1	1	$-\infty$	10
q_6	1	8	7	$+\infty$
q_7	1	8	7	$+\infty$
q_8	2	16	7	$+\infty$
q_{11}	1	1	10	10
q_{12}	11	11	10	10

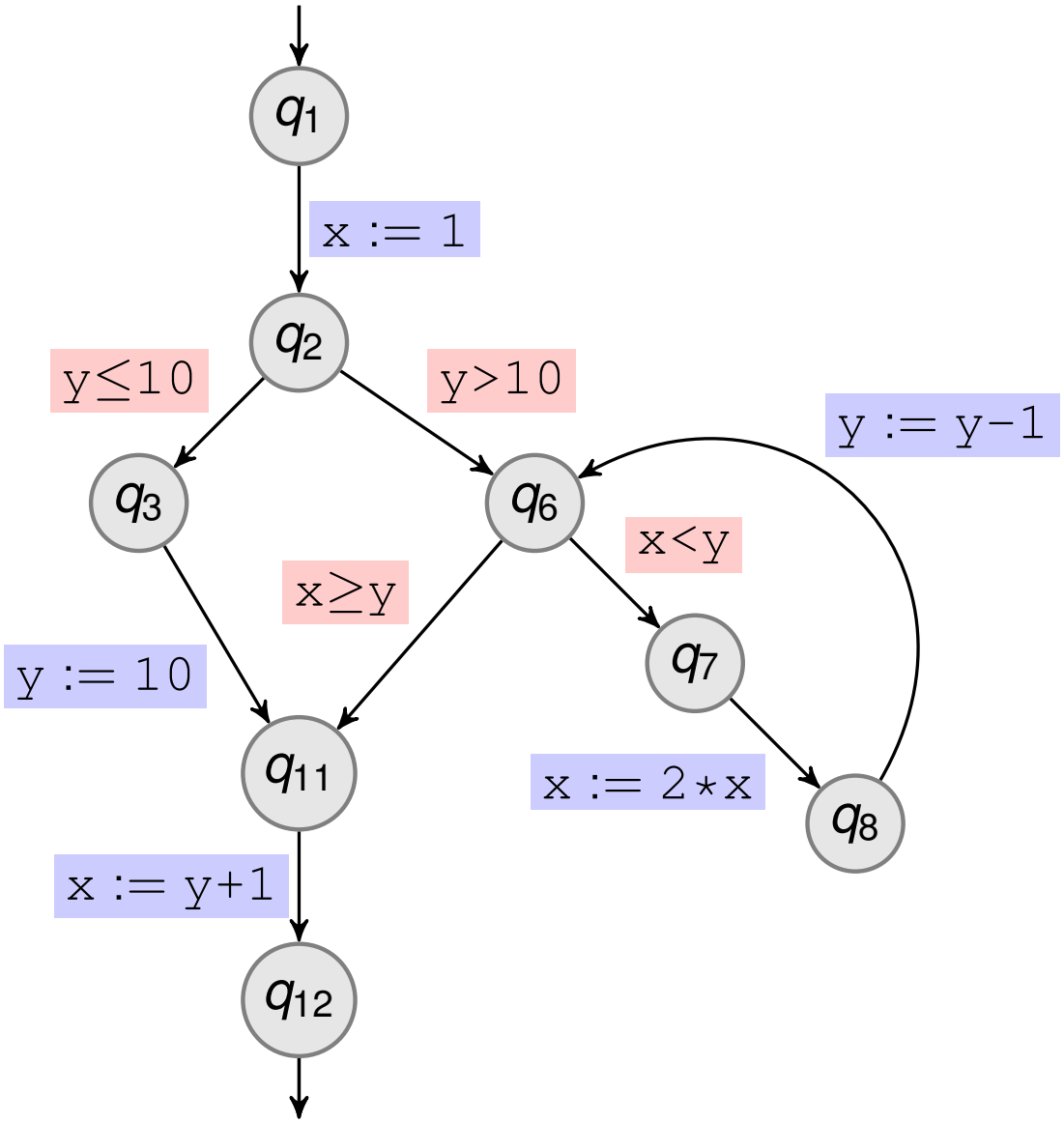
Range Analysis on Running Example



Recall that $\top = (-\infty, +\infty)$

	x		y	
q_1	$-\infty$	$+\infty$	$-\infty$	$+\infty$
q_2	1	1	$+\infty$	$+\infty$
q_3	1	1	$-\infty$	10
q_6	1	8	7	$+\infty$
q_7	1	8	7	$+\infty$
q_8	2	16	7	$+\infty$
q_{11}	1	1	10	10
q_{12}	11	11	10	10

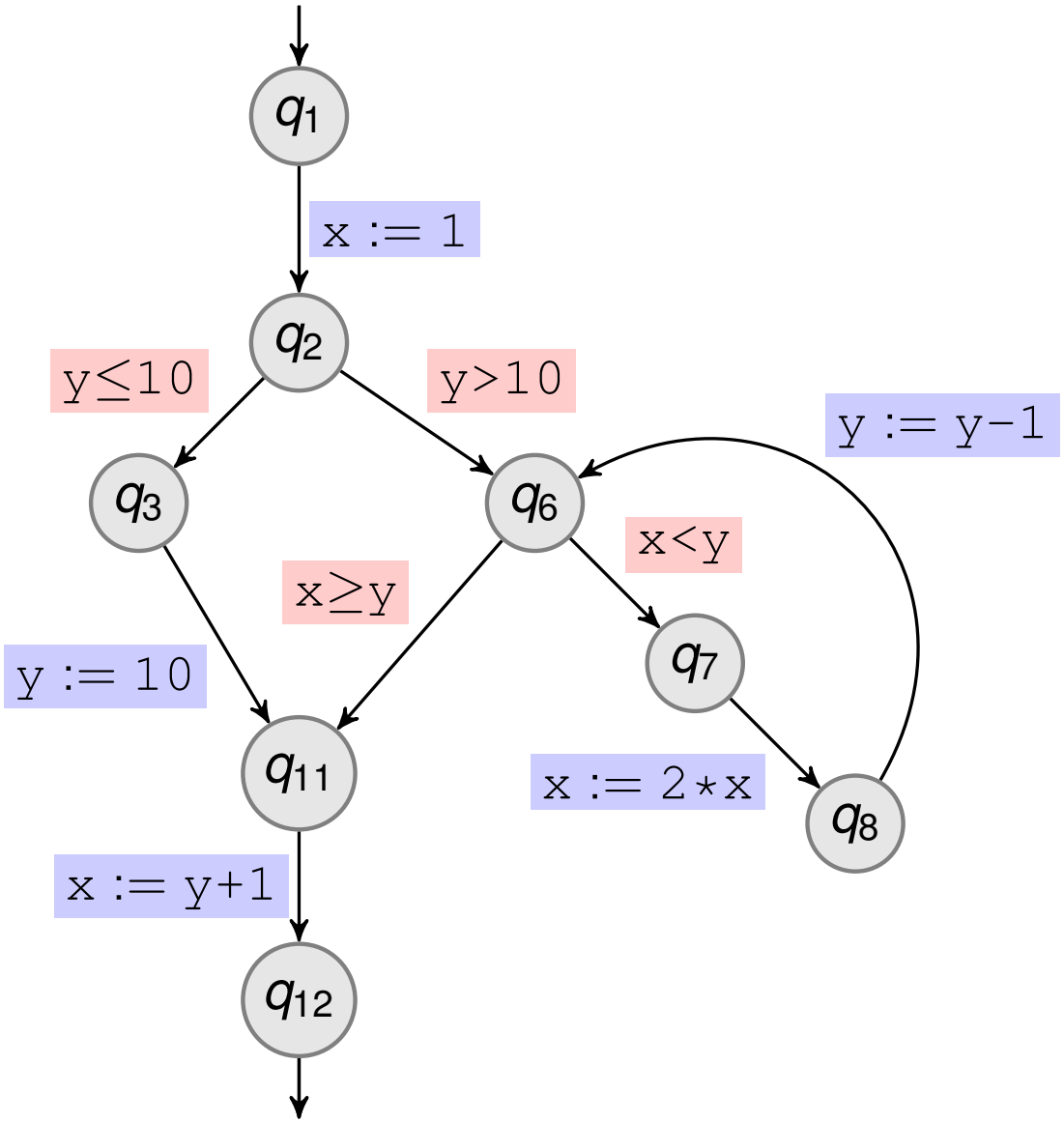
Range Analysis on Running Example



Recall that $\top = (-\infty, +\infty)$

	x		y	
q_1	$-\infty$	$+\infty$	$-\infty$	$+\infty$
q_2	1	1	$+\infty$	$+\infty$
q_3	1	1	$-\infty$	10
q_6	1	16	6	$+\infty$
q_7	1	16	6	$+\infty$
q_8	2	32	6	$+\infty$
q_{11}	1	1	10	10
q_{12}	11	11	10	10

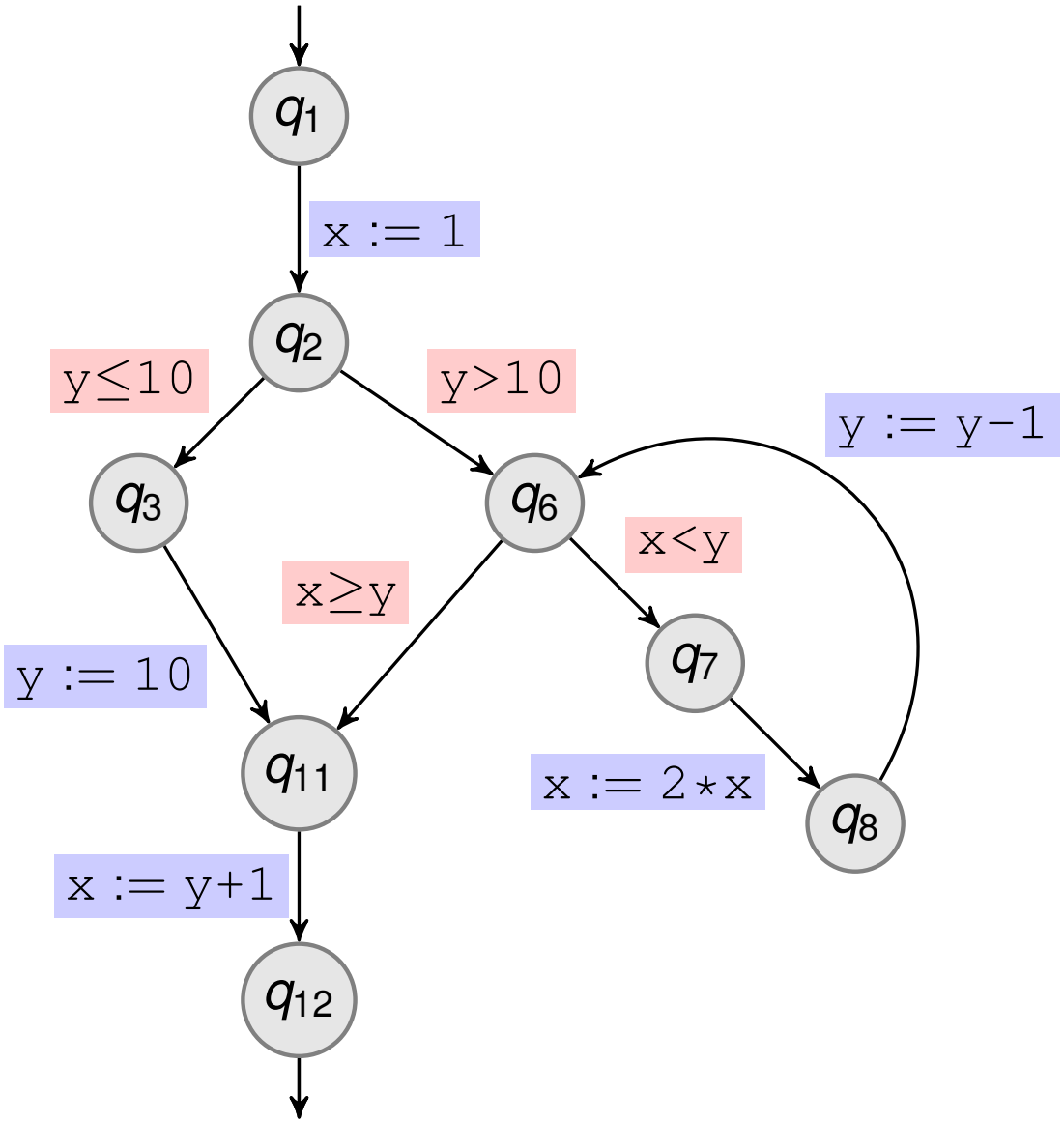
Range Analysis on Running Example



Recall that $\top = (-\infty, +\infty)$

	x		y	
q_1	$-\infty$	$+\infty$	$-\infty$	$+\infty$
q_2	1	1	$+\infty$	$+\infty$
q_3	1	1	$-\infty$	10
q_6	1	32	5	$+\infty$
q_7	1	32	5	$+\infty$
q_8	2	2^6	5	$+\infty$
q_{11}	1	1	10	10
q_{12}	11	11	10	10

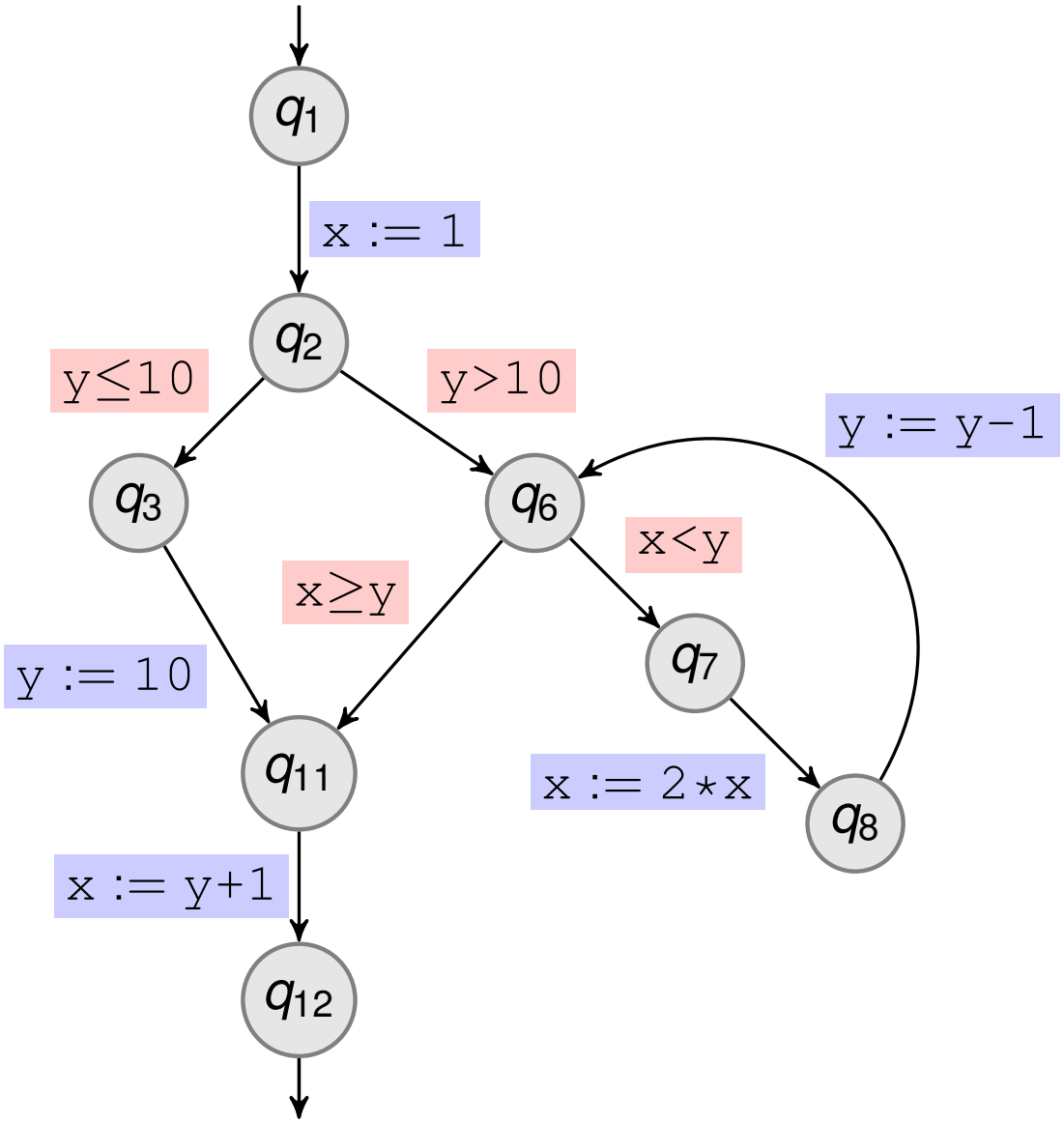
Range Analysis on Running Example



Recall that $\top = (-\infty, +\infty)$

	x		y	
q_1	$-\infty$	$+\infty$	$-\infty$	$+\infty$
q_2	1	1	$+\infty$	$+\infty$
q_3	1	1	$-\infty$	10
q_6	1	2^6	4	$+\infty$
q_7	1	2^6	4	$+\infty$
q_8	2	2^7	4	$+\infty$
q_{11}	1	1	10	10
q_{12}	11	11	10	10

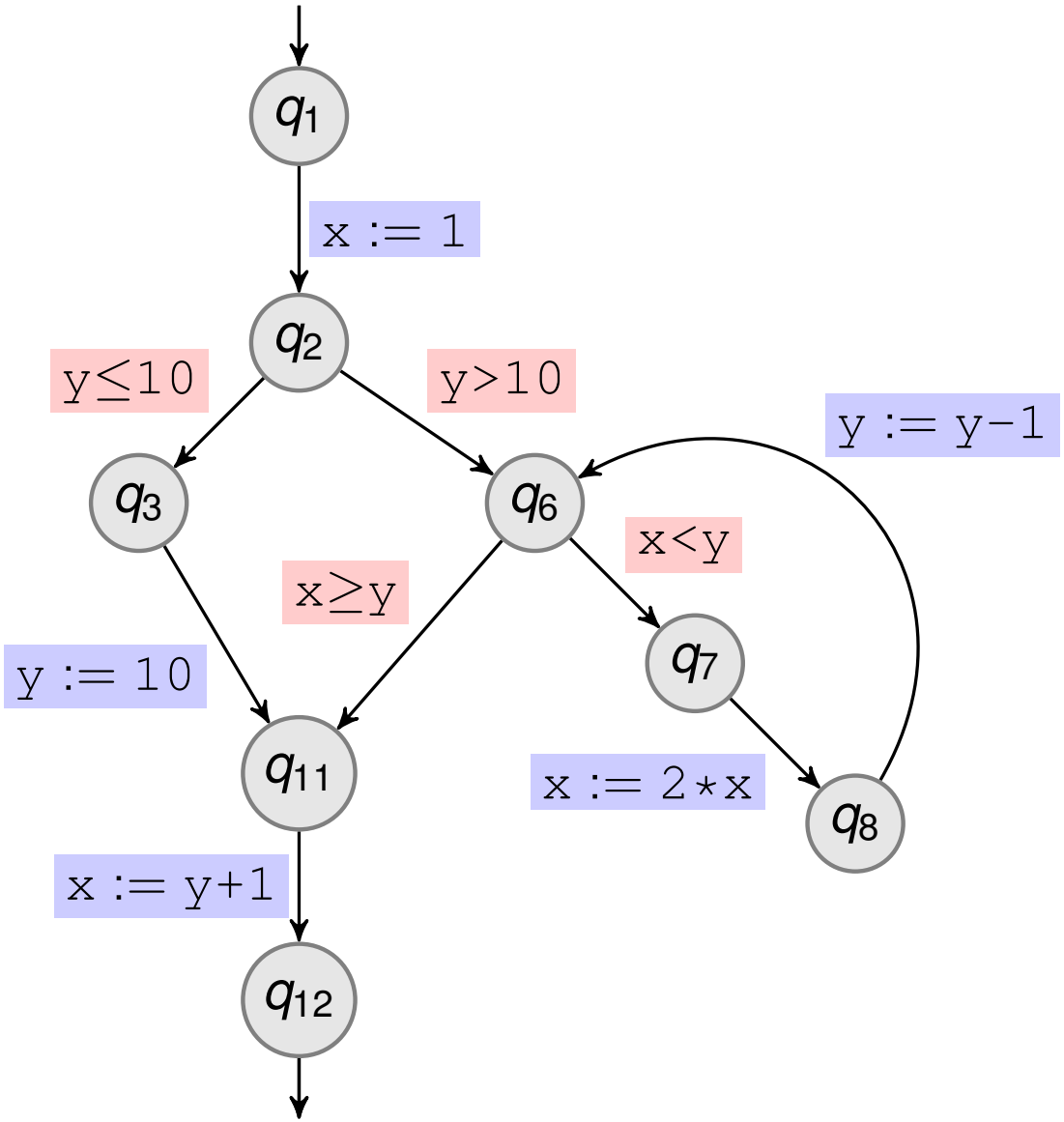
Range Analysis on Running Example



Recall that $\top = (-\infty, +\infty)$

	x		y	
q_1	$-\infty$	$+\infty$	$-\infty$	$+\infty$
q_2	1	1	$+\infty$	$+\infty$
q_3	1	1	$-\infty$	10
q_6	1	2^7	3	$+\infty$
q_7	1	2^7	3	$+\infty$
q_8	2	2^8	3	$+\infty$
q_{11}	1	1	10	10
q_{12}	11	11	10	10

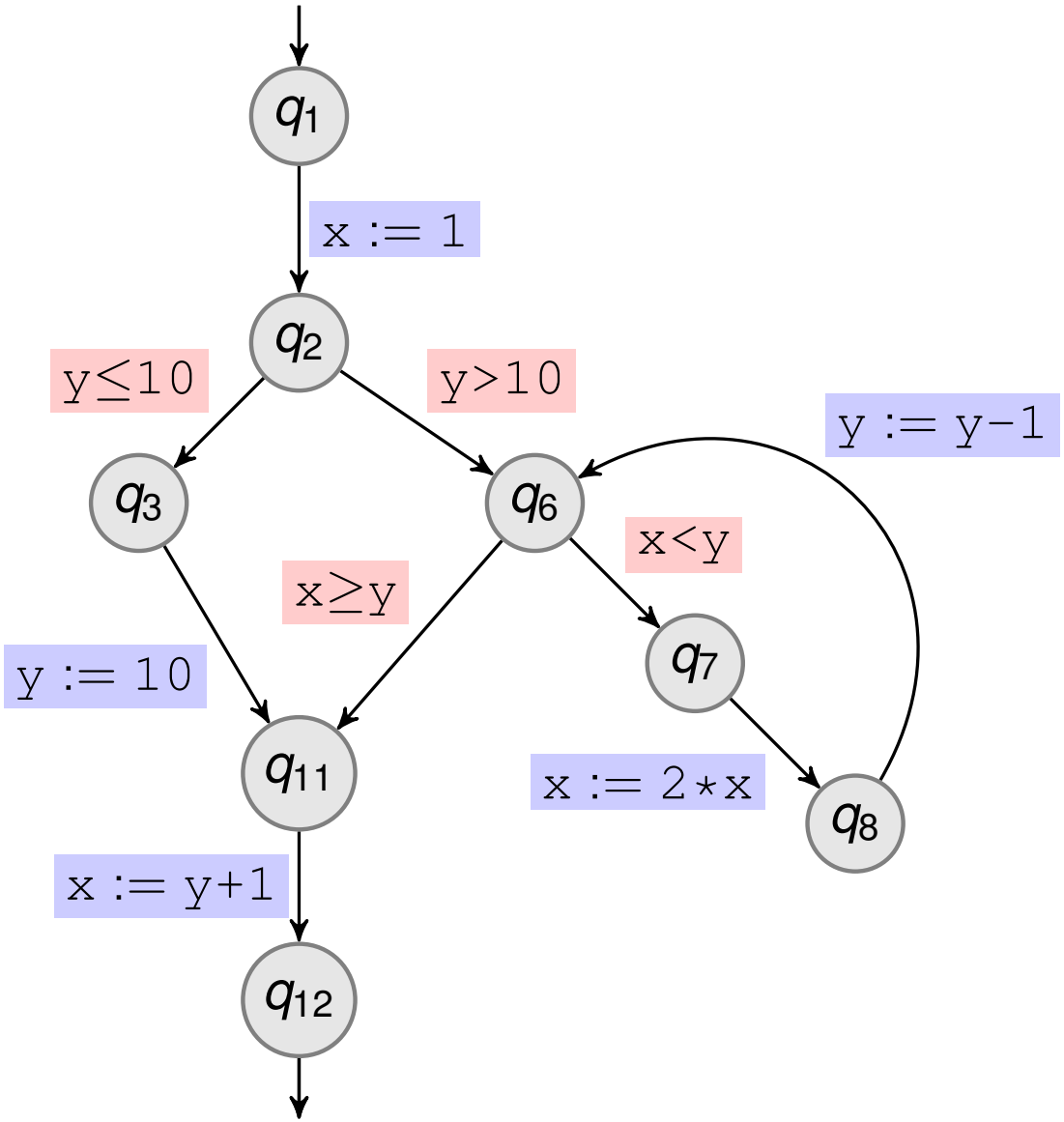
Range Analysis on Running Example



Recall that $\top = (-\infty, +\infty)$

	x		y	
q_1	$-\infty$	$+\infty$	$-\infty$	$+\infty$
q_2	1	1	$+\infty$	$+\infty$
q_3	1	1	$-\infty$	10
q_6	1	2^8	2	$+\infty$
q_7	1	2^8	2	$+\infty$
q_8	2	2^9	2	$+\infty$
q_{11}	1	1	10	10
q_{12}	11	11	10	10

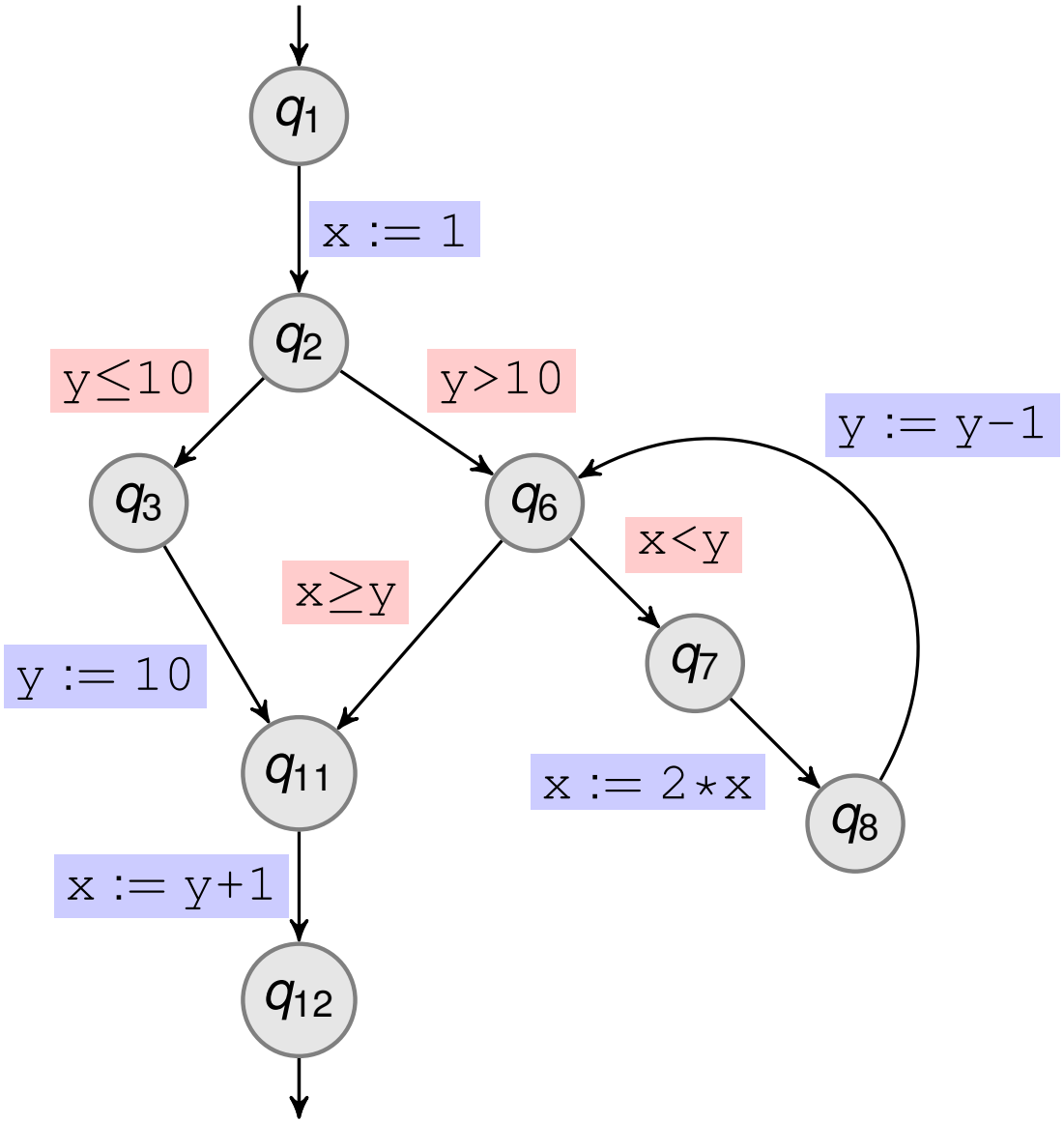
Range Analysis on Running Example



Recall that $\top = (-\infty, +\infty)$

	x		y	
q_1	$-\infty$	$+\infty$	$-\infty$	$+\infty$
q_2	1	1	$+\infty$	$+\infty$
q_3	1	1	$-\infty$	10
q_6	1	2^9	1	$+\infty$
q_7	1	2^9	1	$+\infty$
q_8	2	2^{10}	1	$+\infty$
q_{11}	1	1	10	10
q_{12}	11	11	10	10

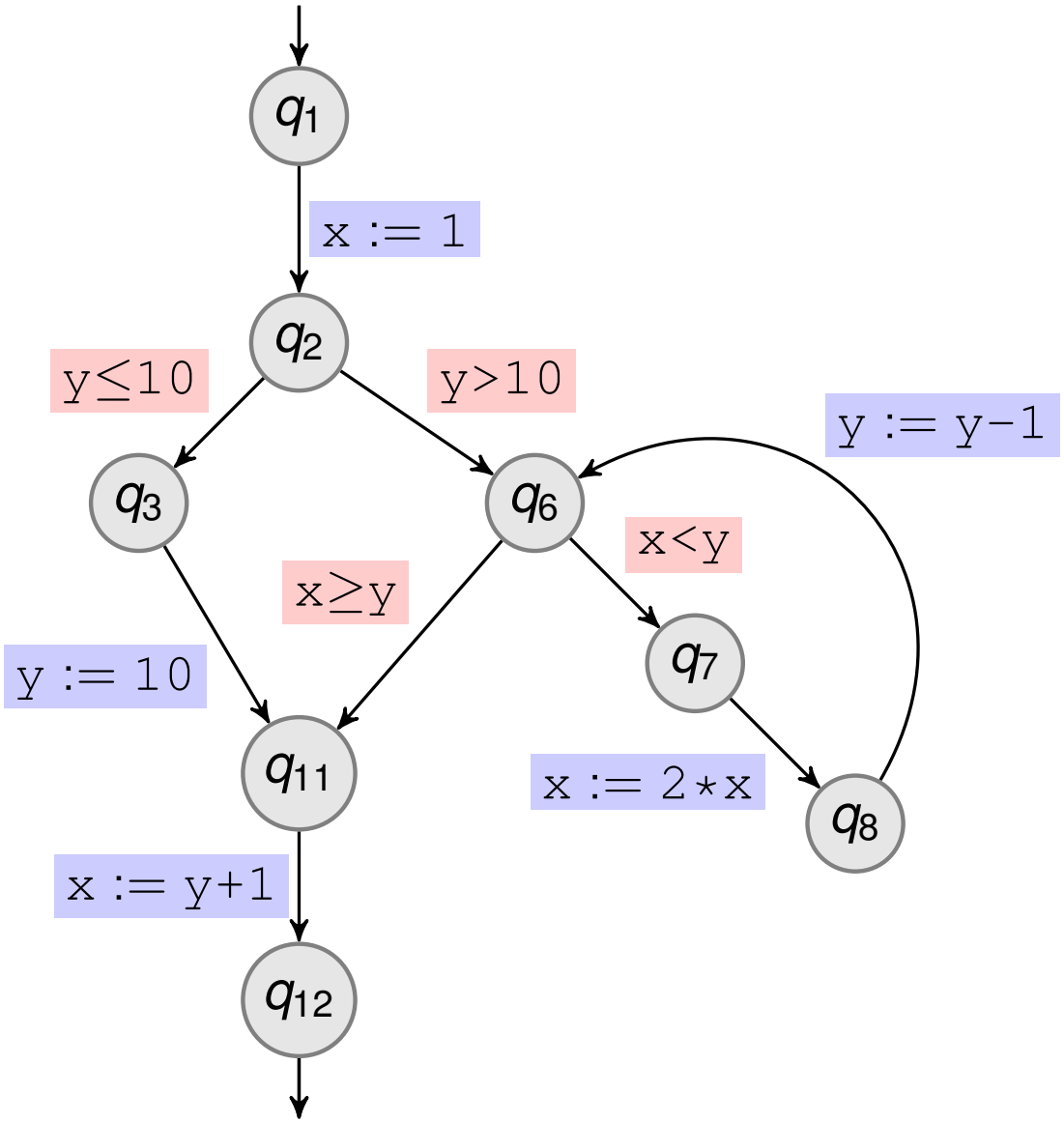
Range Analysis on Running Example



Recall that $\top = (-\infty, +\infty)$

	x		y	
q_1	$-\infty$	$+\infty$	$-\infty$	$+\infty$
q_2	1	1	$+\infty$	$+\infty$
q_3	1	1	$-\infty$	10
q_6	1	2^9	1	$+\infty$
q_7	1	2^9	1	$+\infty$
q_8	2	2^{10}	1	$+\infty$
q_{11}	1	1	10	10
q_{12}	11	11	10	10

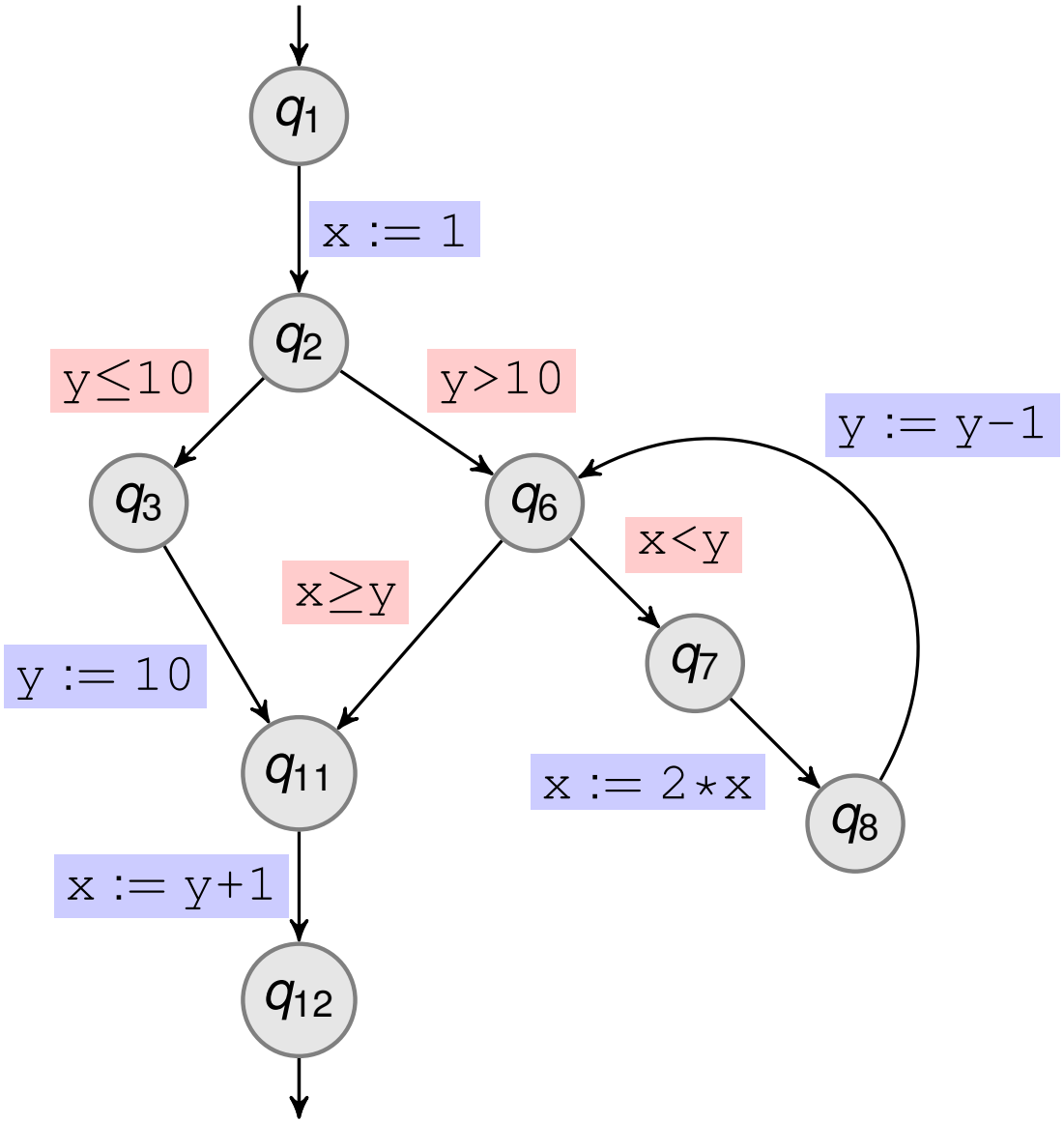
Range Analysis on Running Example



Recall that $\top = (-\infty, +\infty)$

	x		y	
q_1	$-\infty$	$+\infty$	$-\infty$	$+\infty$
q_2	1	1	$+\infty$	$+\infty$
q_3	1	1	$-\infty$	10
q_6	1	2^{10}	0	$+\infty$
q_7	1	2^9	1	$+\infty$
q_8	2	2^{10}	1	$+\infty$
q_{11}	1	1	10	10
q_{12}	11	11	10	10

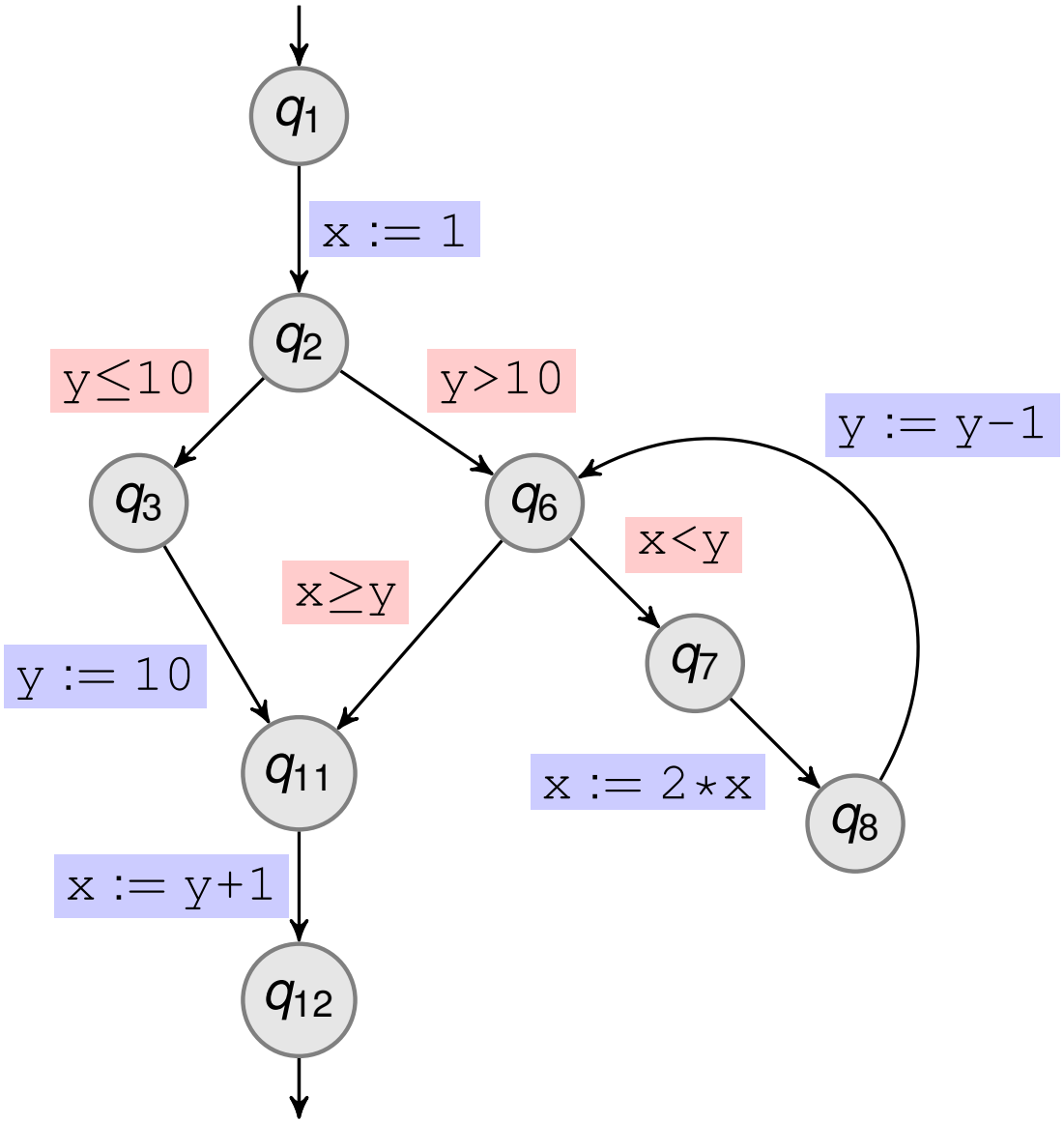
Range Analysis on Running Example



Recall that $\top = (-\infty, +\infty)$

	x		y	
q_1	$-\infty$	$+\infty$	$-\infty$	$+\infty$
q_2	1	1	$+\infty$	$+\infty$
q_3	1	1	$-\infty$	10
q_6	1	2^{10}	0	$+\infty$
q_7	1	2^9	1	$+\infty$
q_8	2	2^{10}	1	$+\infty$
q_{11}	1	1	10	10
q_{12}	11	11	10	10

Range Analysis on Running Example

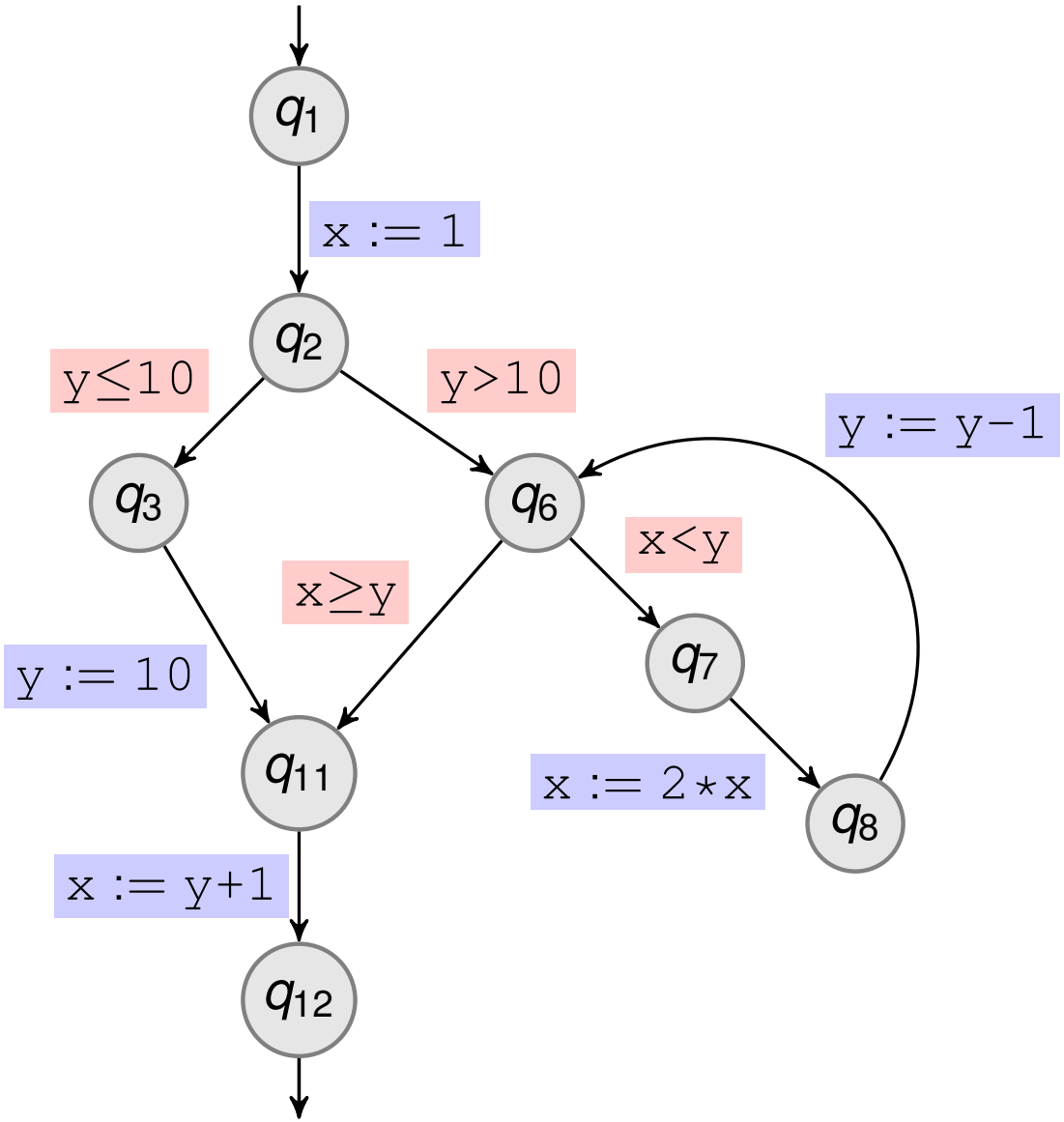


Recall that $\top = (-\infty, +\infty)$

	x		y	
q_1	$-\infty$	$+\infty$	$-\infty$	$+\infty$
q_2	1	1	$+\infty$	$+\infty$
q_3	1	1	$-\infty$	10
q_6	1	2^{10}	0	$+\infty$
q_7	1	2^{10}	1	$+\infty$
q_8	2	2^{10}	1	$+\infty$
q_{11}	1	1	10	10
q_{12}	11	11	10	10

Gained from guard x < y

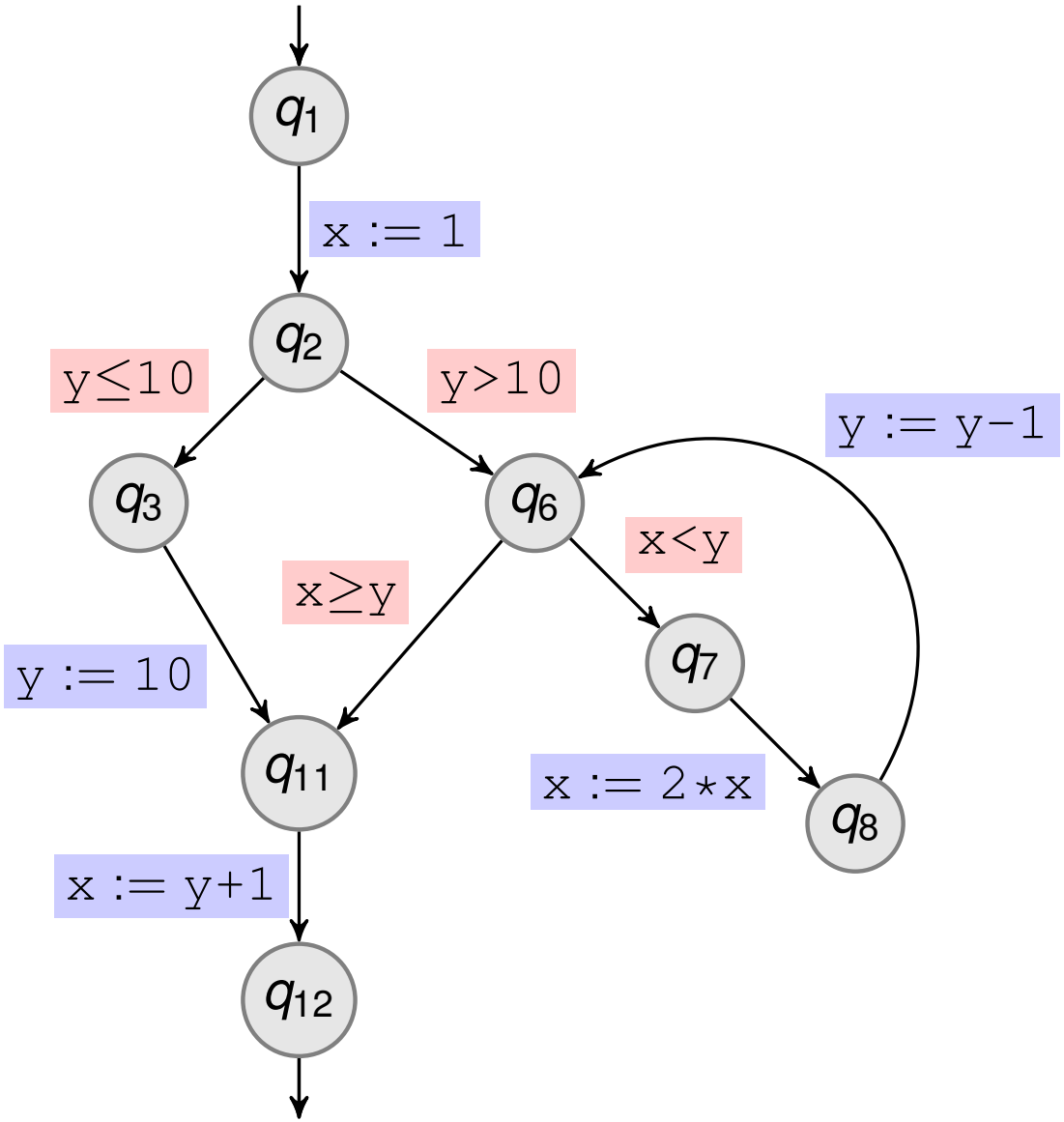
Range Analysis on Running Example



Recall that $\top = (-\infty, +\infty)$

	x		y	
q_1	$-\infty$	$+\infty$	$-\infty$	$+\infty$
q_2	1	1	$+\infty$	$+\infty$
q_3	1	1	$-\infty$	10
q_6	1	2^{10}	0	$+\infty$
q_7	1	2^{10}	1	$+\infty$
q_8	2	2^{10}	1	$+\infty$
q_{11}	1	1	10	10
q_{12}	11	11	10	10

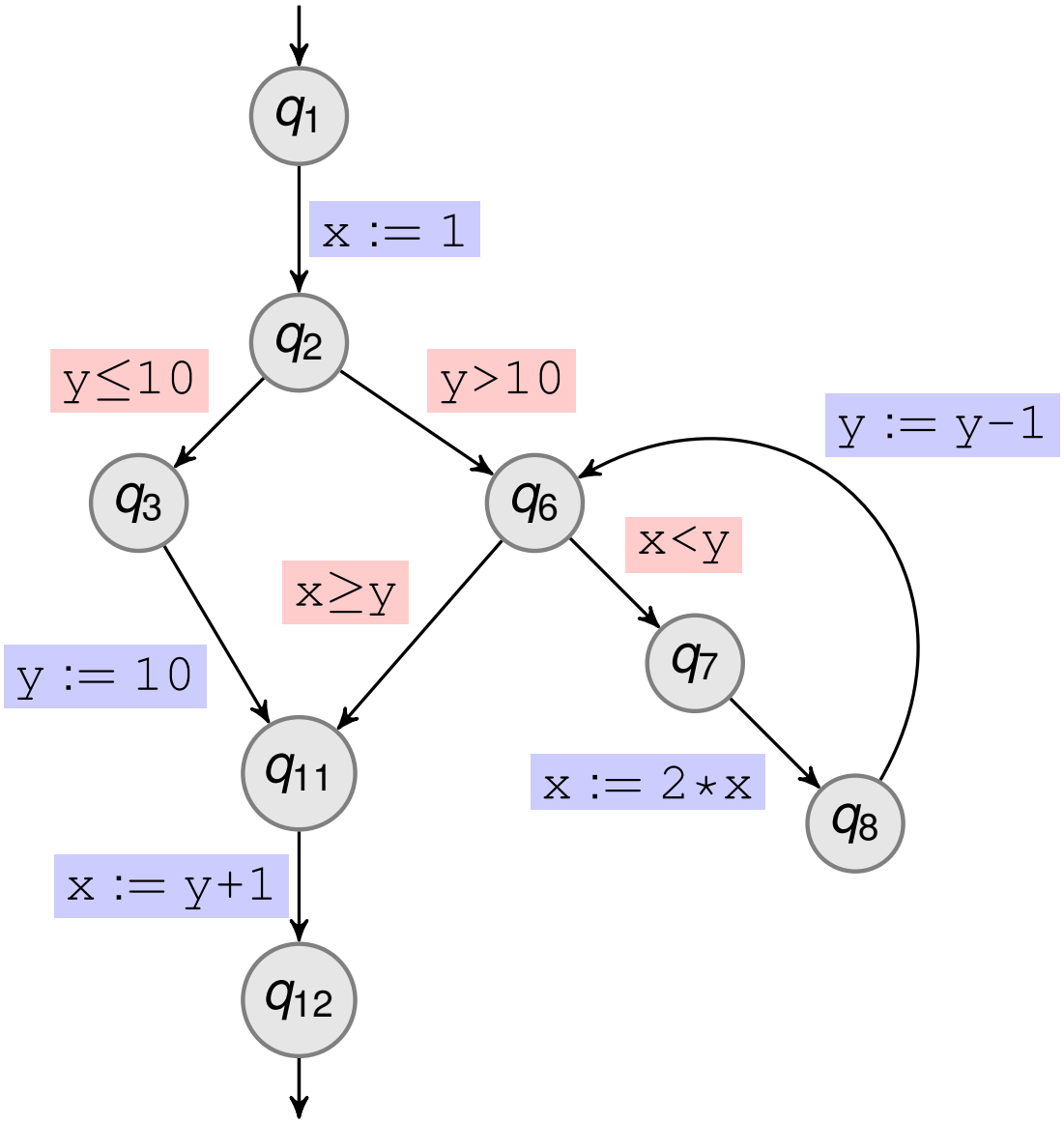
Range Analysis on Running Example



Recall that $\top = (-\infty, +\infty)$

	x		y	
q_1	$-\infty$	$+\infty$	$-\infty$	$+\infty$
q_2	1	1	$+\infty$	$+\infty$
q_3	1	1	$-\infty$	10
q_6	1	2^{10}	0	$+\infty$
q_7	1	2^{10}	1	$+\infty$
q_8	2	2^{11}	1	$+\infty$
q_{11}	1	1	10	10
q_{12}	11	11	10	10

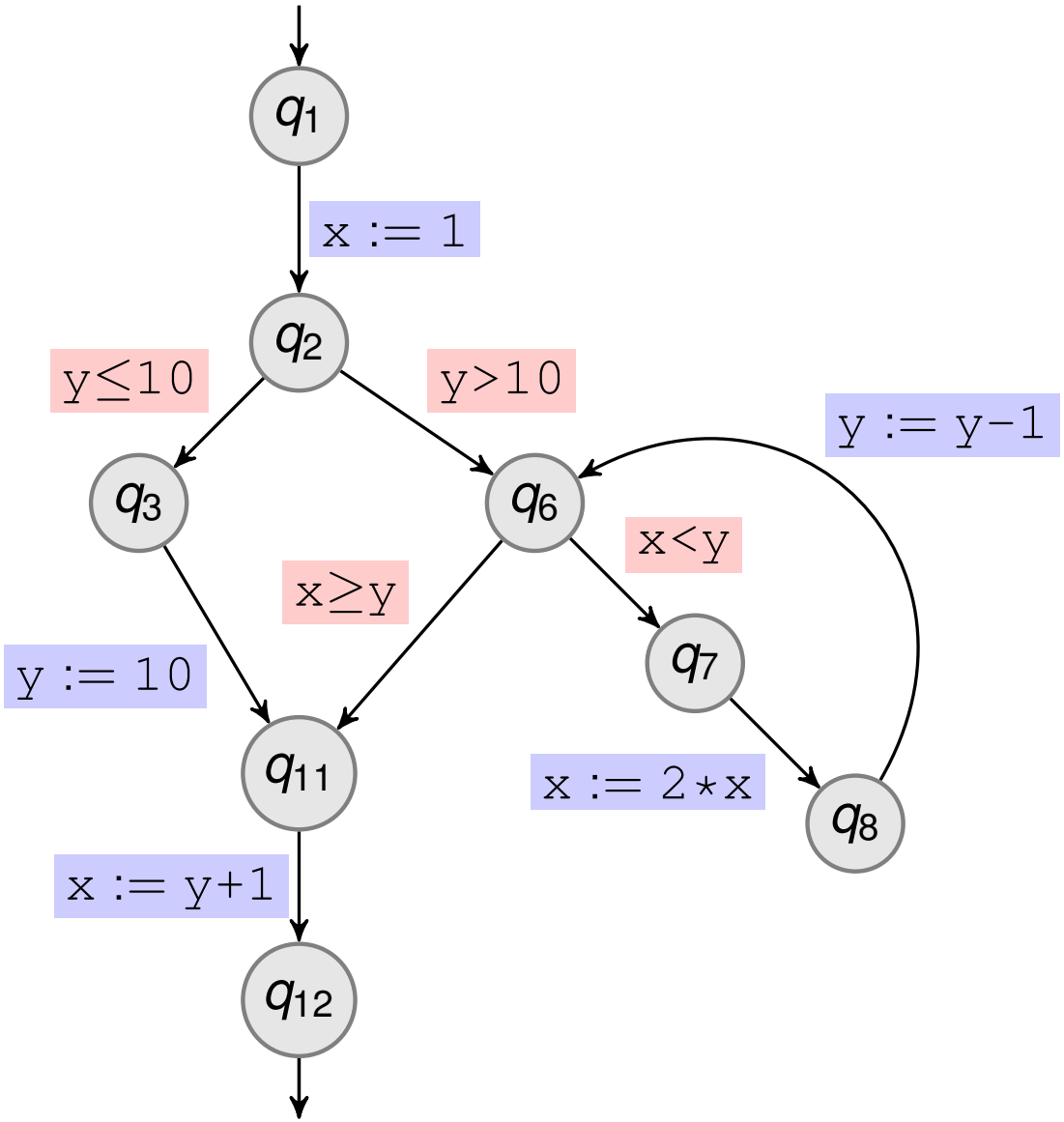
Range Analysis on Running Example



Recall that $\top = (-\infty, +\infty)$

	x		y	
q_1	$-\infty$	$+\infty$	$-\infty$	$+\infty$
q_2	1	1	$+\infty$	$+\infty$
q_3	1	1	$-\infty$	10
q_6	1	2^{10}	0	$+\infty$
q_7	1	2^{10}	1	$+\infty$
q_8	2	2^{11}	1	$+\infty$
q_{11}	1	1	10	10
q_{12}	11	11	10	10

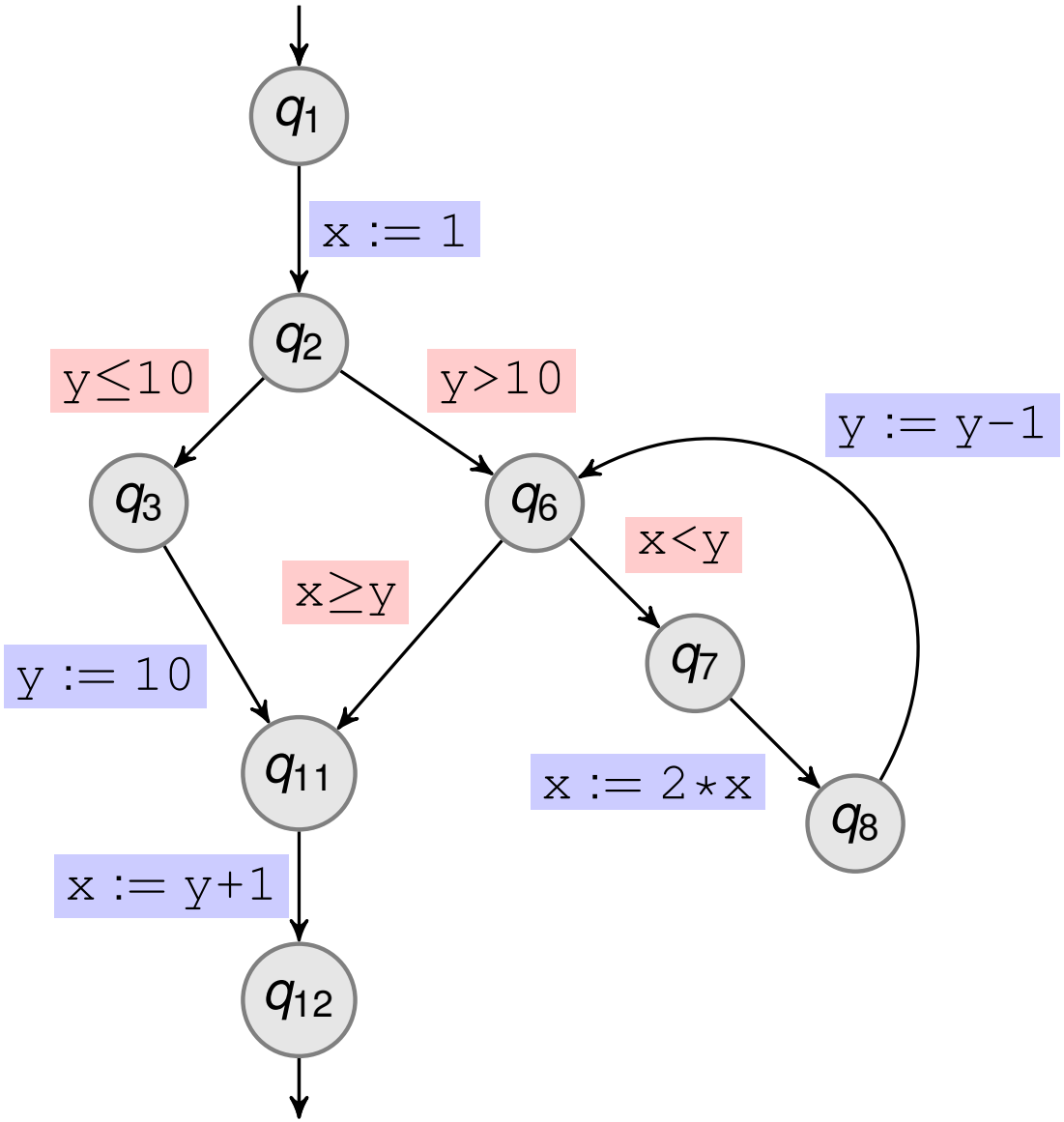
Range Analysis on Running Example



Recall that $\top = (-\infty, +\infty)$

	x		y	
q_1	$-\infty$	$+\infty$	$-\infty$	$+\infty$
q_2	1	1	$+\infty$	$+\infty$
q_3	1	1	$-\infty$	10
q_6	1	2^{11}	0	$+\infty$
q_7	1	2^{11}	1	$+\infty$
q_8	2	2^{12}	1	$+\infty$
q_{11}	1	1	10	10
q_{12}	11	11	10	10

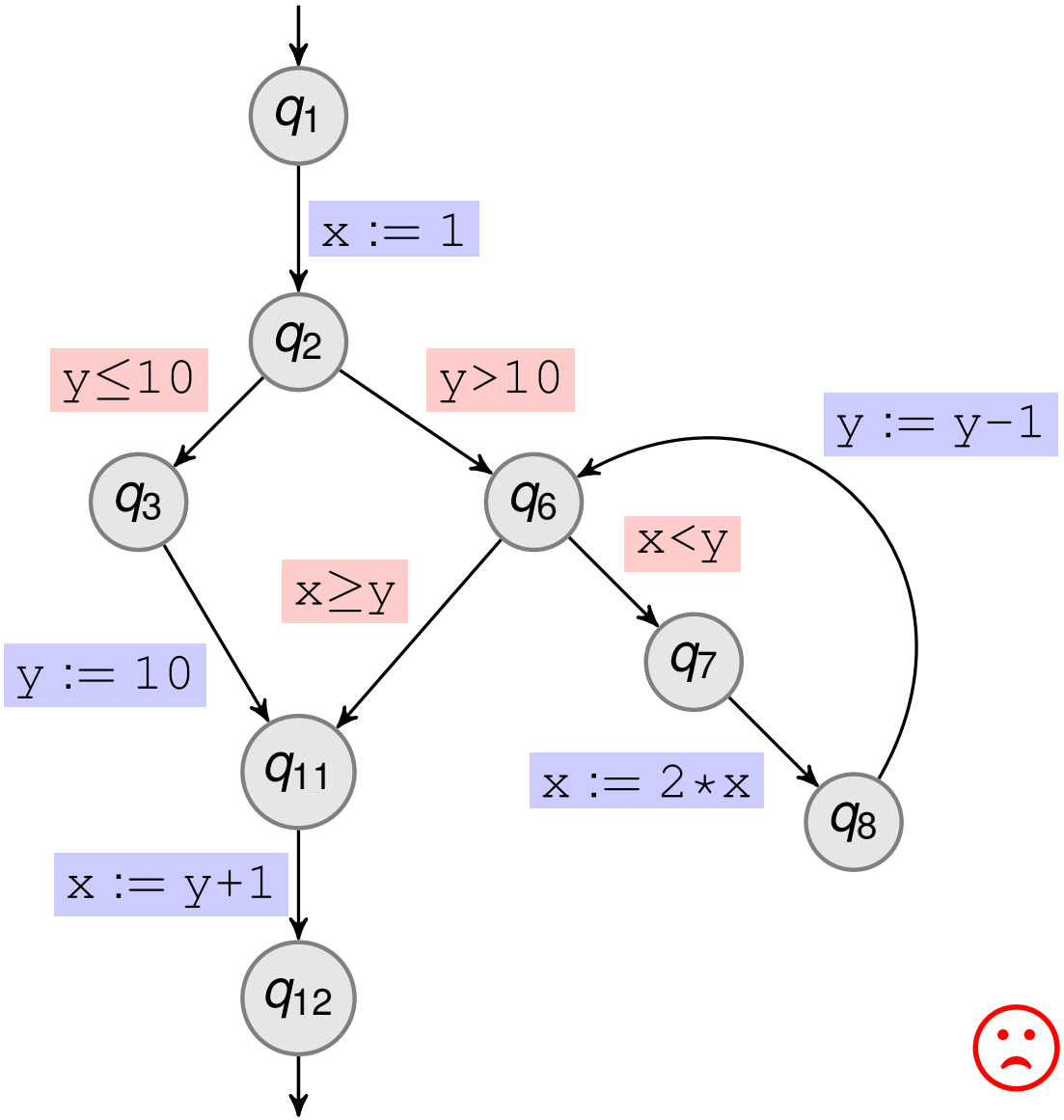
Range Analysis on Running Example



Recall that $\top = (-\infty, +\infty)$

	x		y	
q_1	$-\infty$	$+\infty$	$-\infty$	$+\infty$
q_2	1	1	$+\infty$	$+\infty$
q_3	1	1	$-\infty$	10
q_6	1	2^{12}	0	$+\infty$
q_7	1	2^{12}	1	$+\infty$
q_8	2	2^{13}	1	$+\infty$
q_{11}	1	1	10	10
q_{12}	11	11	10	10

Range Analysis on Running Example



Recall that $\top = (-\infty, +\infty)$

	x		y	
q_1	$-\infty$	$+\infty$	$-\infty$	$+\infty$
q_2	1	1	$+\infty$	$+\infty$
q_3	1	1	$-\infty$	10
q_6	1	···	0	$+\infty$
q_7	1	···	1	$+\infty$
q_8	2	···	1	$+\infty$
q_{11}	1	1	10	10
q_{12}	11	11	10	10

☹

Does not converge!

Dynamic Approximation: Widening Operators

Consider a complete lattice $(L, \sqsubseteq)$.

Objective of Widening Operators

Soundly extrapolate "limits" of ascending chains

Definition

A widening operator for $(L, \sqsubseteq)$ is a function $\nabla : (L \times L) \to L$ such that:

1. $$x \sqcup y \sqsubseteq x \nabla y \qquad \text{(for all } x, y \in L\text{)}$$
2. for any ascending chain $x_0 \sqsubseteq x_1 \sqsubseteq \cdots$ of elements of L, the ascending chain $y_0 \sqsubseteq y_1 \sqsubseteq \cdots$ defined by

$$\begin{cases} y_0 = x_0 \\ y_{i+1} = y_i \nabla x_{i+1} & \text{for all } i \in \mathbb{N} \end{cases}$$

is not strictly increasing (i.e. $y_{i+1} = y_i$ for some $i \in \mathbb{N}$).

Correctness of Kleene Iteration with Widening

Consider a complete lattice $(L, \sqsubseteq)$ and a monotonic function $f : L \to L$.

Theorem

If $\nabla : (L \times L) \to L$ is a widening operator then the ascending chain $x_0 \sqsubseteq x_1 \sqsubseteq \cdots$ defined by

$$\begin{aligned} x_0 &= \bot \\ x_{i+1} &= \begin{cases} x_i & \text{if } f(x_i) \sqsubseteq x_i \\ x_i \nabla f(x_{i+1}) & \text{otherwise} \end{cases} \end{aligned}$$

is eventually stationary, and its limit satisfies $\bigsqcup \{x_i \mid i \in \mathbb{N}\} \sqsupseteq \mathrm{lfp}(f)$.

Application to MFP Approximation in Data Flow Analysis

Replacing $\sqcup$ with ∇ in Kleene / round-robin / worklist algorithms

- guarantees termination, but
- at the expense of precision.

Correctness of Kleene Iteration with Widening

Consider a complete lattice $(L, \sqsubseteq)$ and a monotonic function $f : L \to L$.

Theorem

If $\nabla : (L \times L) \to L$ is a widening operator then the ascending chain $x_0 \sqsubseteq x_1 \sqsubseteq \cdots$ defined by

$$\begin{aligned} x_0 &= \bot \\ x_{i+1} &= \begin{cases} x_i & \text{if } f(x_i) \sqsubseteq x_i \\ x_i \nabla f(x_{i+1}) & \text{otherwise} \end{cases} \end{aligned}$$

is eventually stationary, and its limit satisfies $\bigsqcup \{x_i \mid i \in \mathbb{N}\} \sqsupseteq \mathrm{lfp}(f)$.

Application to MFP Approximation in Data Flow Analysis

Replacing $\sqcup$ with ∇ in Kleene / round-robin / worklist algorithms

- guarantees termination, but
- at the expense of precision.

(Forward) Round-Robin Iteration with Widening

Consider a data flow instance $\langle (L, \sqsubseteq), \mathcal{F}, Q, q_{in}, q_{out}, \mathtt{X}, \rightarrow, f, \imath \rangle$.

```
foreach q ∈ Q
    a[q] ← ⊥
a[q_in] ← ι
do
    change ← false
    foreach q --op--> q'
        new ← f_op(a[q])
        if new ⋢ a[q']
            a[q'] ← a[q'] ∇ new
            change ← true
while change
return a
```

If ∇ is a widening operator on $(L, \sqsubseteq)$ then:

- this algorithm terminates for any data flow instance on $(L, \sqsubseteq)$.
- the returned $a \in Q \rightarrow L$ satisfies:

$$\overrightarrow{\mathrm{MFP}}(q) \sqsubseteq a(q)$$

for every $q \in Q$.

Widening Operator for Range Analysis: Intuition

Objective of Widening Operators

Soundly extrapolate "limits" of ascending chains

Put ∞ when the bound is moving towards ∞

Examples

$$\begin{aligned}
\ldots,\ (1,2),\ (1,3),\ (1,4) &\longrightarrow (1,+\infty)\\
\ldots,\ (1,2),\ (-1,2),\ (-6,2) &\longrightarrow (-\infty,2)\\
\ldots,\ (1,2),\ (-9,3),\ (-19,4) &\longrightarrow (-\infty,+\infty)\\
\ldots,\ (1,+\infty),\ (-9,+\infty),\ (-19,+\infty) &\longrightarrow (-\infty,+\infty)
\end{aligned}$$

∇ only looks at the last two elements of the sequence

Widening Operator for Range Analysis

Widening Operator on the Complete Lattice $(Int, \sqsubseteq)$ of Intervals

$$\bot \nabla \bot = \bot \qquad \bot \nabla (l,u) = (l,u) \nabla \bot = (l,u)$$

$$(l_1,u_1) \nabla (l_2,u_2) = (l_\nabla, u_\nabla) \quad \text{where} \quad \begin{cases} l_\nabla = \begin{cases} -\infty & \text{if } l_2 < l_1 \\ l_1 & \text{otherwise} \end{cases} \\ u_\nabla = \begin{cases} +\infty & \text{if } u_2 > u_1 \\ u_1 & \text{otherwise} \end{cases} \end{cases}$$

Widening Operator on the Complete Lattice $(\mathrm{X} \to Int, \overline{\sqsubseteq})$

Extension $\overline{\nabla}$ of the widening ∇ on $(Int, \sqsubseteq)$ to $(\mathrm{X} \to Int, \overline{\sqsubseteq})$, defined by:

$$\overline{v_1} \;\overline{\nabla}\; \overline{v_2} = \lambda q \,.\, \overline{v_1}(q) \nabla \overline{v_2}(q)$$

Widening Operator for Range Analysis

Widening Operator on the Complete Lattice $(Int, \sqsubseteq)$ of Intervals

$$\bot \nabla \bot = \bot \qquad \bot \nabla (l,u) = (l,u) \nabla \bot = (l,u)$$

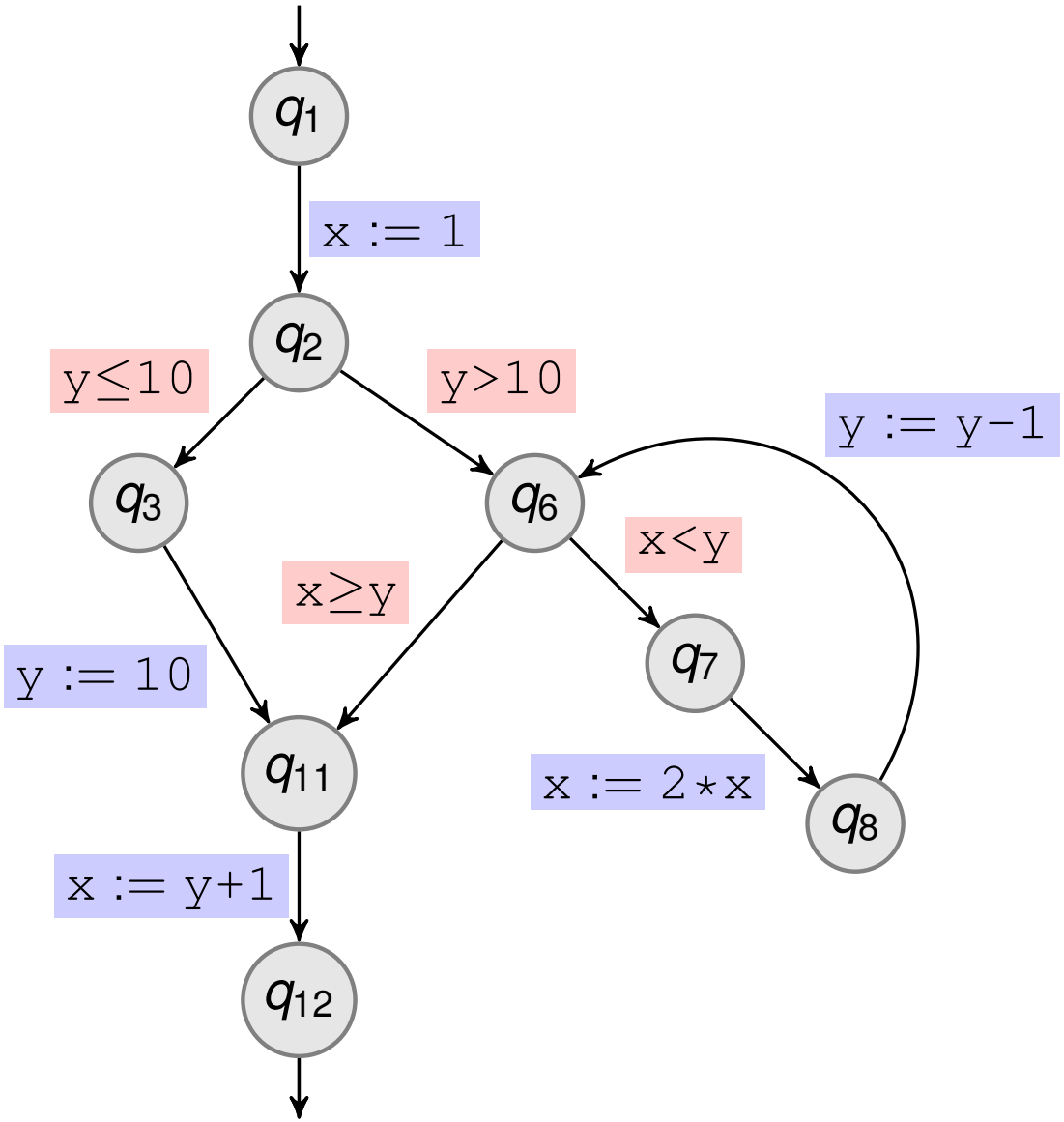
$$(l_1,u_1) \nabla (l_2,u_2) = (l_\nabla, u_\nabla) \quad \text{where} \quad \begin{cases} l_\nabla = \begin{cases} -\infty & \text{if } l_2 < l_1 \\ l_1 & \text{otherwise} \end{cases} \\ u_\nabla = \begin{cases} +\infty & \text{if } u_2 > u_1 \\ u_1 & \text{otherwise} \end{cases} \end{cases}$$

Widening Operator on the Complete Lattice $(\mathbb{X} \to Int, \overline{\sqsubseteq})$

Extension $\overline{\nabla}$ of the widening ∇ on $(Int, \sqsubseteq)$ to $(\mathbb{X} \to Int, \overline{\sqsubseteq})$, defined by:

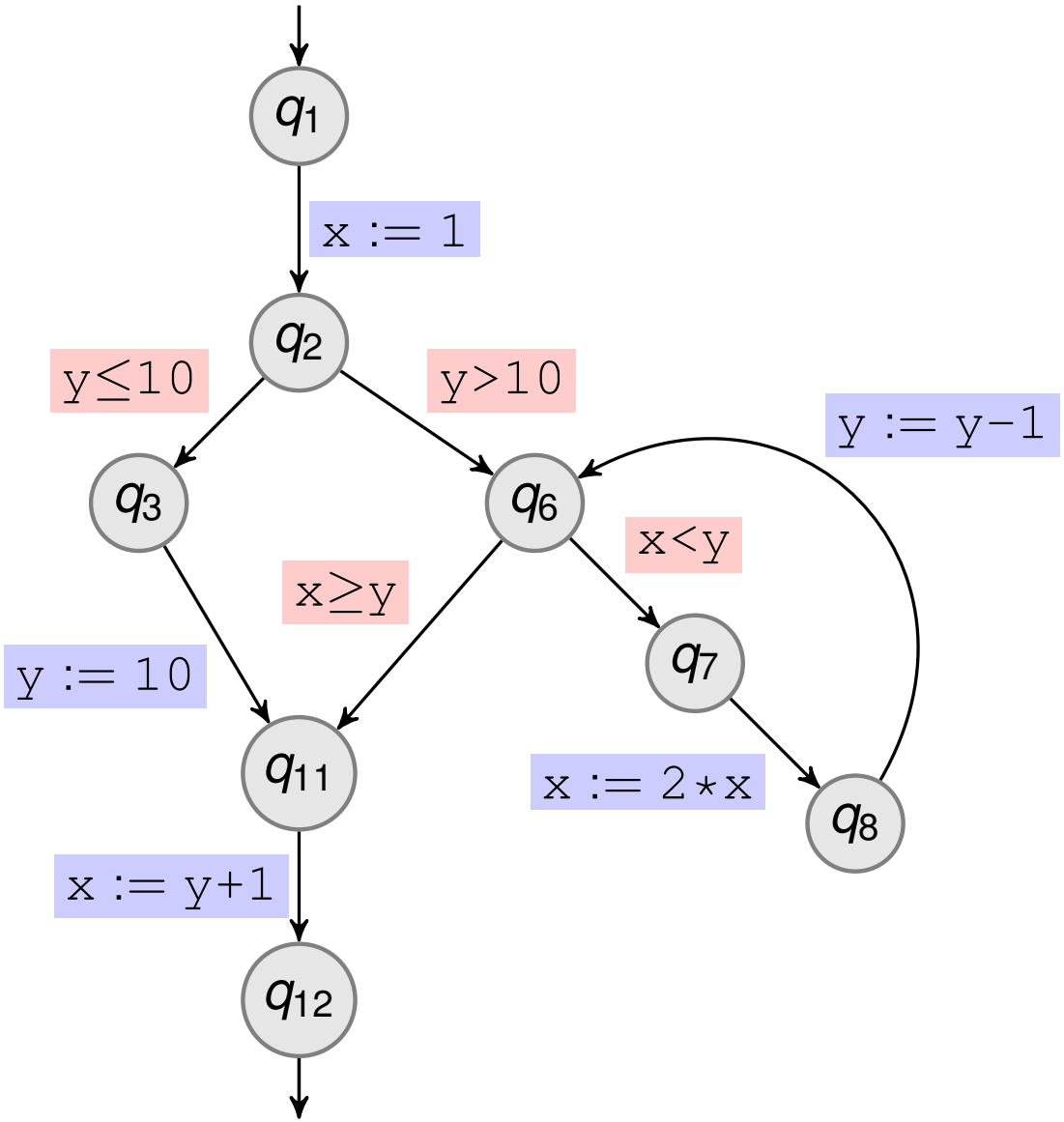
$$\overline{v_1} \;\overline{\nabla}\; \overline{v_2} = \lambda q \,.\, \overline{v_1}(q) \nabla \overline{v_2}(q)$$

Range Analysis on Example with Widening



Show that $x > 0$ at q_{12}

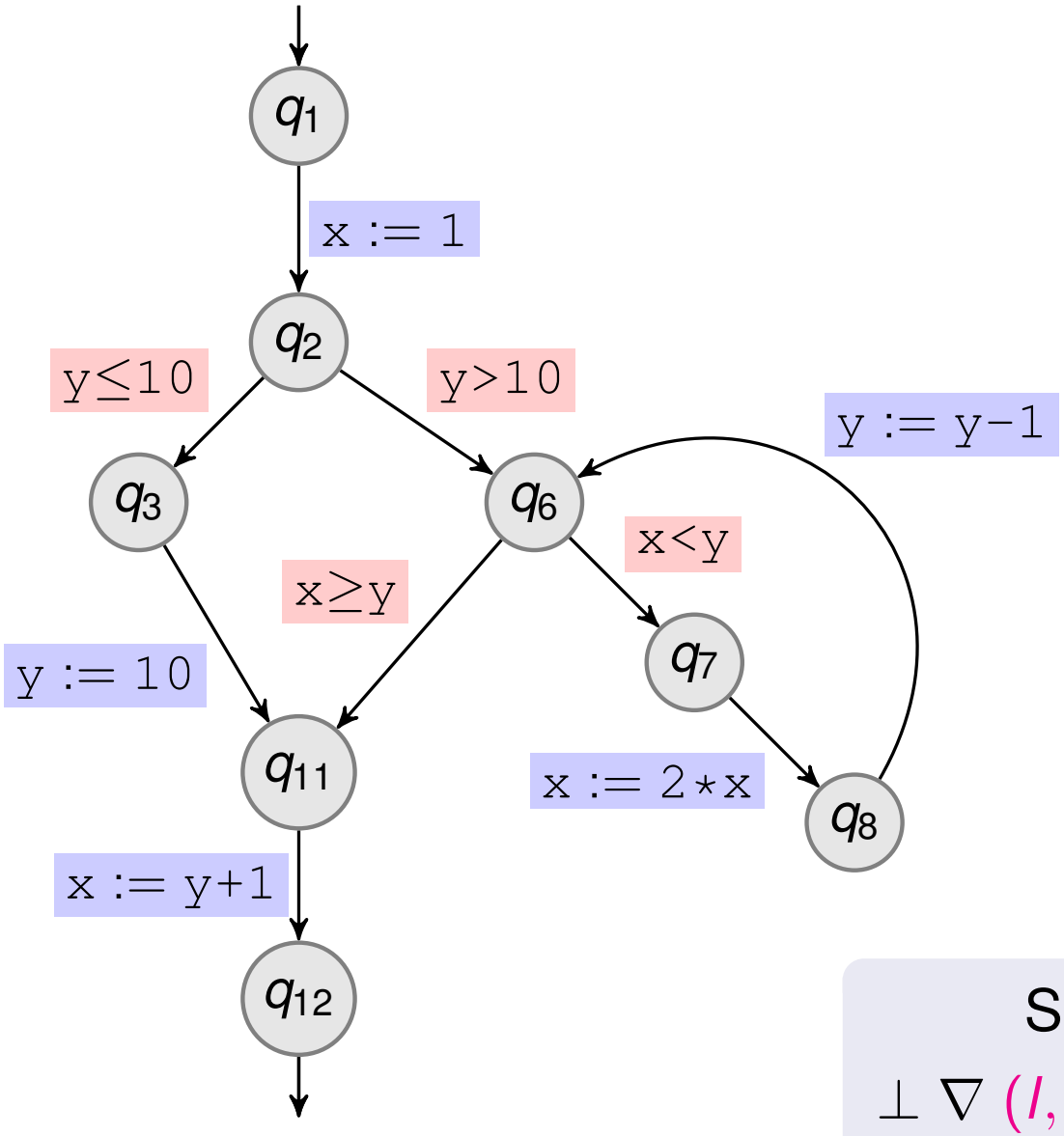
Range Analysis on Example with Widening



Show that $x > 0$ at q_{12}

	x		y	
q_1	$-\infty$	$+\infty$	$-\infty$	$+\infty$
q_2	$\bot$	$\bot$	$\bot$	$\bot$
q_3	$\bot$	$\bot$	$\bot$	$\bot$
q_6	$\bot$	$\bot$	$\bot$	$\bot$
q_7	$\bot$	$\bot$	$\bot$	$\bot$
q_8	$\bot$	$\bot$	$\bot$	$\bot$
q_{11}	$\bot$	$\bot$	$\bot$	$\bot$
q_{12}	$\bot$	$\bot$	$\bot$	$\bot$

Range Analysis on Example with Widening



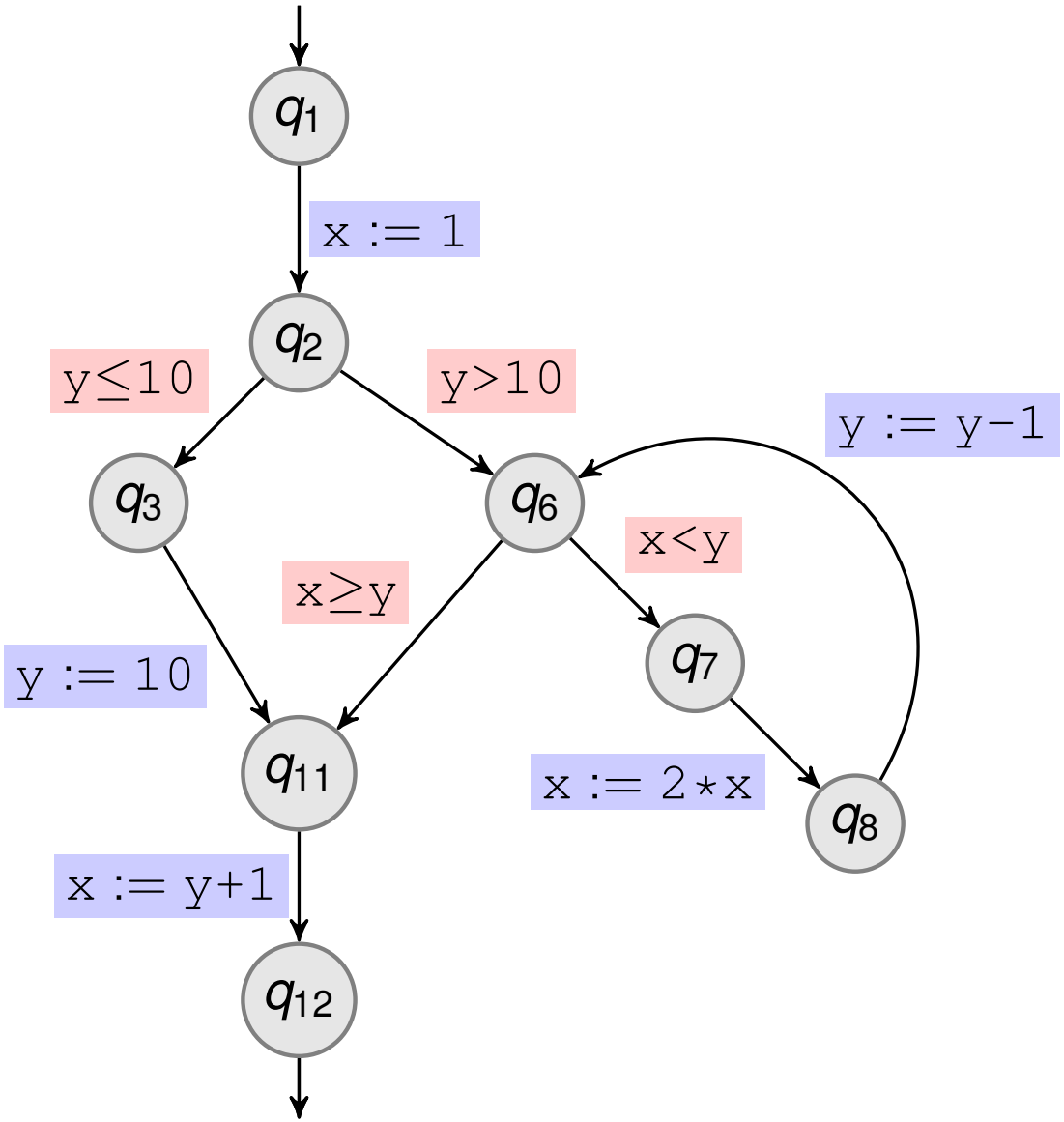
Show that $x > 0$ at q_{12}

	x		y	
q_1	$-\infty$	$+\infty$	$-\infty$	$+\infty$
q_2	1	1	$+\infty$	$+\infty$
q_3	1	1	$-\infty$	10
q_6	1	1	10	$+\infty$
q_7	1	1	10	$+\infty$
q_8	2	2	10	$+\infty$
q_{11}	1	1	10	10
q_{12}	11	11	10	10

Same as without ∇ since

$$\bot \nabla (l, u) = \bot \sqcup (l, u) = (l, u)$$

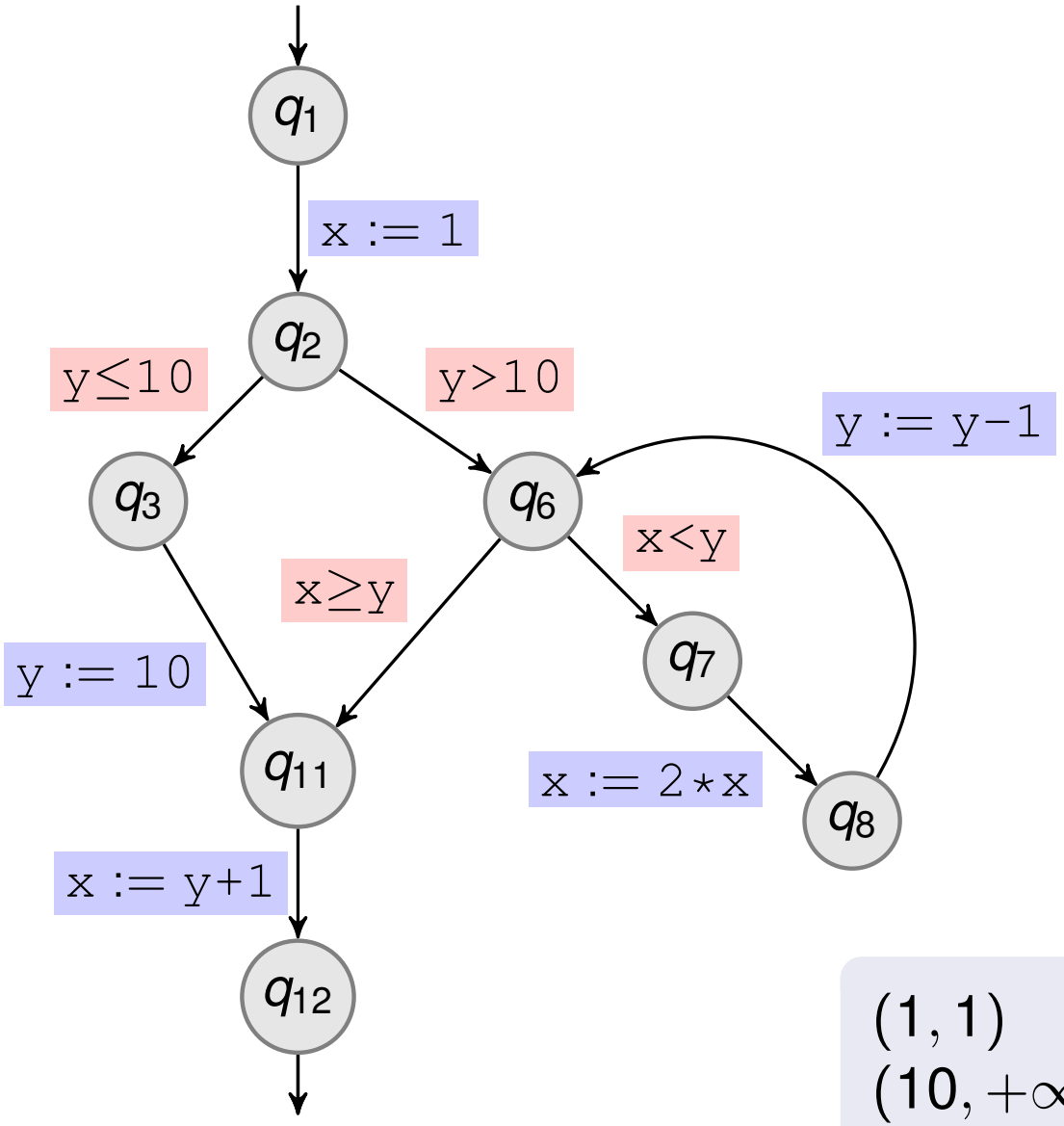
Range Analysis on Example with Widening



Show that $x > 0$ at q_{12}

	x		y	
q_1	$-\infty$	$+\infty$	$-\infty$	$+\infty$
q_2	1	1	$+\infty$	$+\infty$
q_3	1	1	$-\infty$	10
q_6	1	1	10	$+\infty$
q_7	1	1	10	$+\infty$
q_8	2	2	10	$+\infty$
q_{11}	1	1	10	10
q_{12}	11	11	10	10

Range Analysis on Example with Widening

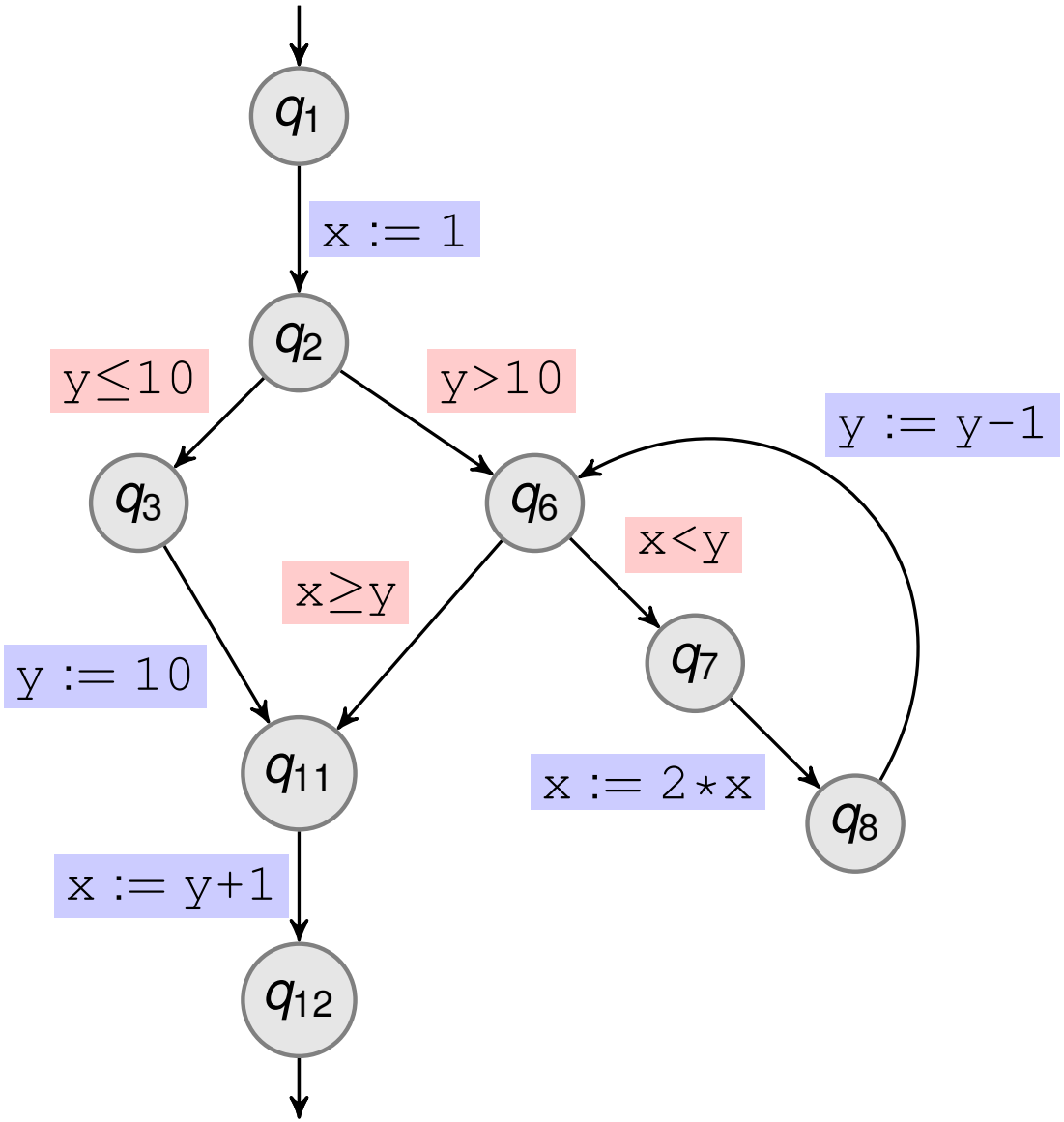


Show that $x > 0$ at q_{12}

	x		y	
q_1	$-\infty$	$+\infty$	$-\infty$	$+\infty$
q_2	1	1	$+\infty$	$+\infty$
q_3	1	1	$-\infty$	10
q_6	1	$+\infty$	$-\infty$	$+\infty$
q_7	1	1	10	$+\infty$
q_8	2	2	10	$+\infty$
q_{11}	1	1	10	10
q_{12}	11	11	10	10

$$(1,1) \quad \nabla \quad (2,2) \quad = \quad (1,+\infty)$$
$$(10,+\infty) \quad \nabla \quad (9,+\infty) \quad = \quad (-\infty,+\infty)$$

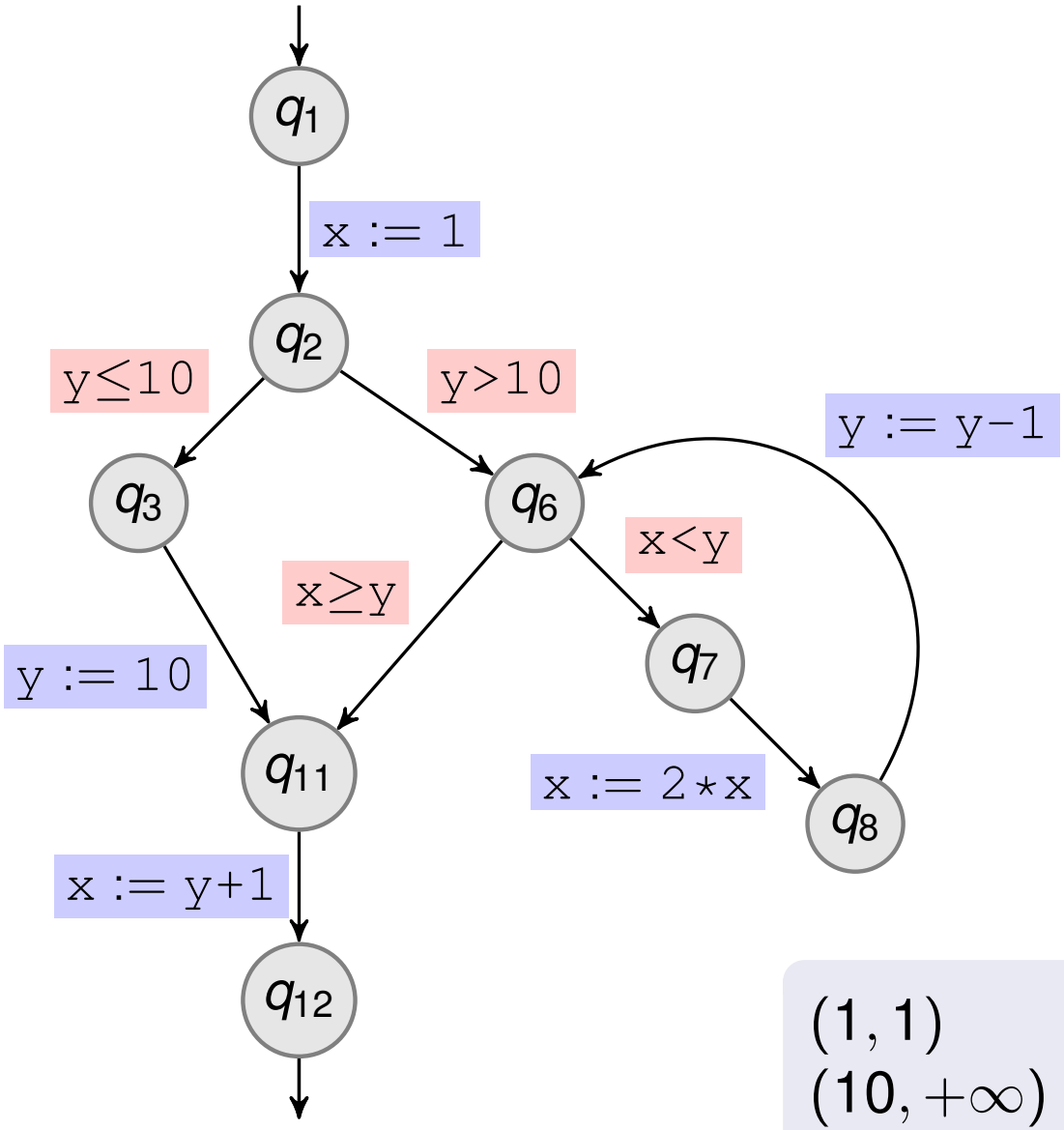
Range Analysis on Example with Widening



Show that $x > 0$ at q_{12}

	x		y	
q_1	$-\infty$	$+\infty$	$-\infty$	$+\infty$
q_2	1	1	$+\infty$	$+\infty$
q_3	1	1	$-\infty$	10
q_6	1	$+\infty$	$-\infty$	$+\infty$
q_7	1	1	10	$+\infty$
q_8	2	2	10	$+\infty$
q_{11}	1	1	10	10
q_{12}	11	11	10	10

Range Analysis on Example with Widening

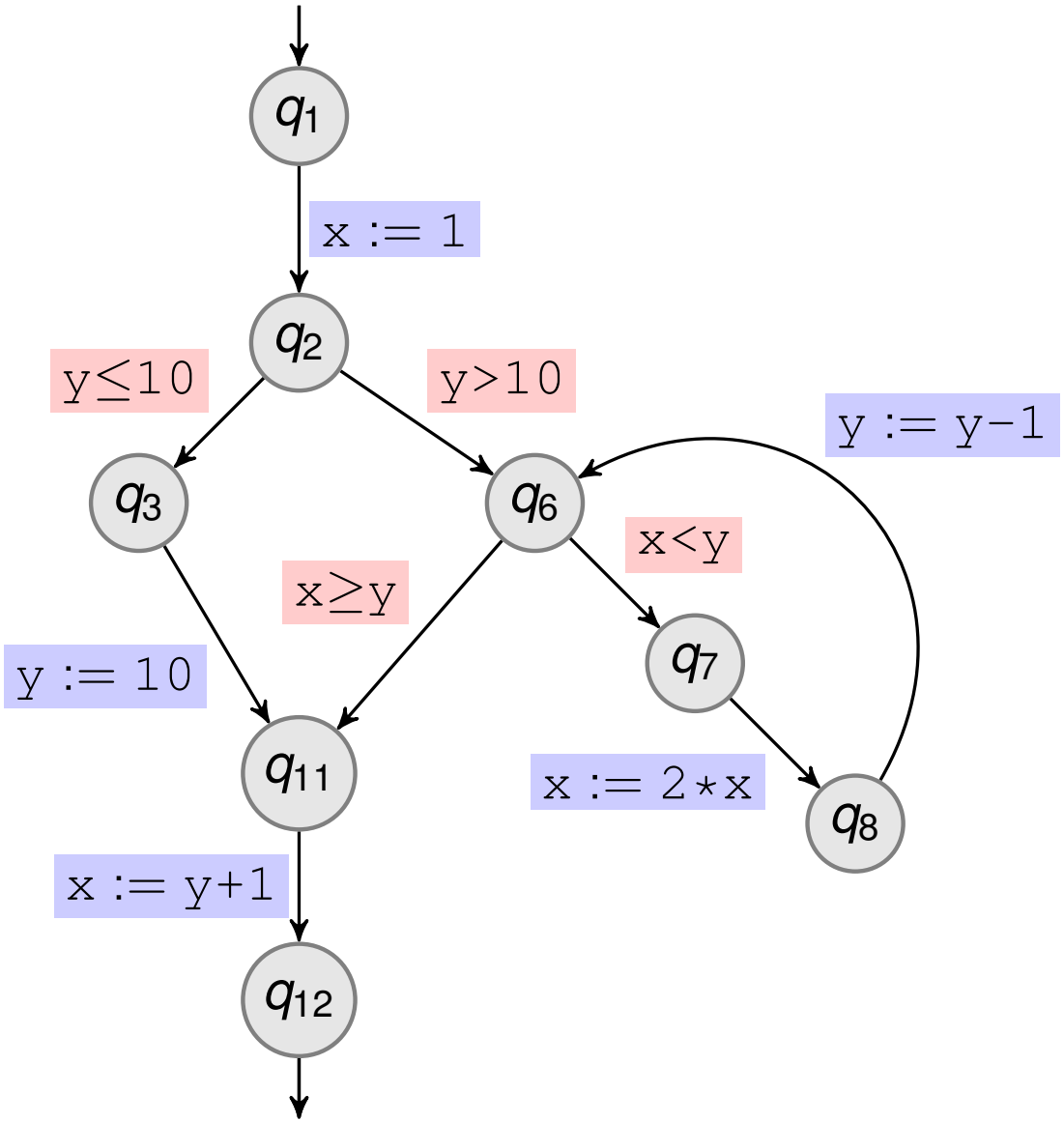


Show that $x > 0$ at q_{12}

	x		y	
q_1	$-\infty$	$+\infty$	$-\infty$	$+\infty$
q_2	1	1	$+\infty$	$+\infty$
q_3	1	1	$-\infty$	10
q_6	1	$+\infty$	$-\infty$	$+\infty$
q_7	1	$+\infty$	$-\infty$	$+\infty$
q_8	2	2	10	$+\infty$
q_{11}	1	1	10	10
q_{12}	11	11	10	10

$$(1,1) \quad \nabla \quad (1,+\infty) \quad = \quad (1,+\infty)$$
$$(10,+\infty) \quad \nabla \quad (-\infty,+\infty) \quad = \quad (-\infty,+\infty)$$

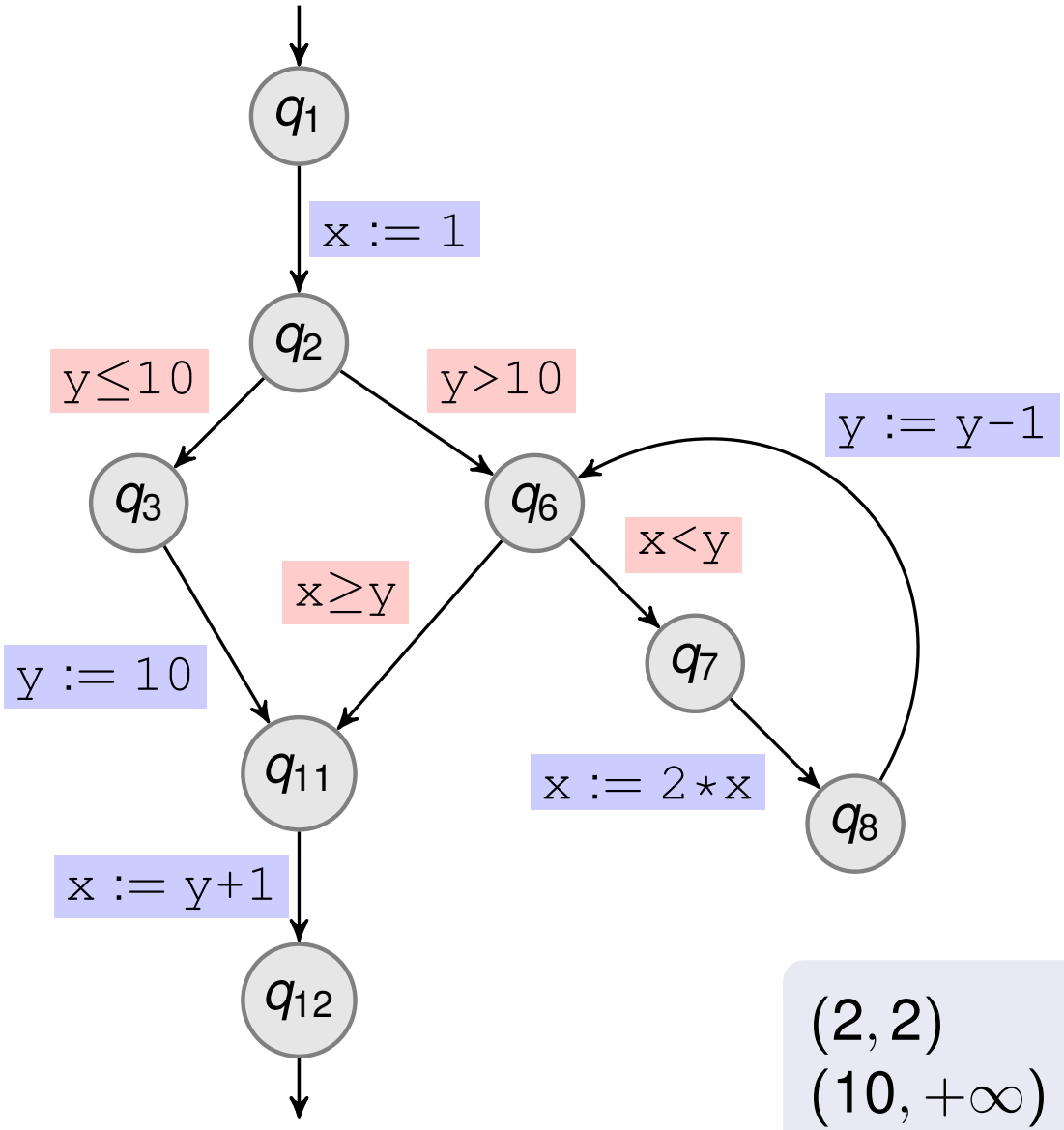
Range Analysis on Example with Widening



Show that $x > 0$ at q_{12}

	x		y	
q_1	$-\infty$	$+\infty$	$-\infty$	$+\infty$
q_2	1	1	$+\infty$	$+\infty$
q_3	1	1	$-\infty$	10
q_6	1	$+\infty$	$-\infty$	$+\infty$
q_7	1	$+\infty$	$-\infty$	$+\infty$
q_8	2	2	10	$+\infty$
q_{11}	1	1	10	10
q_{12}	11	11	10	10

Range Analysis on Example with Widening

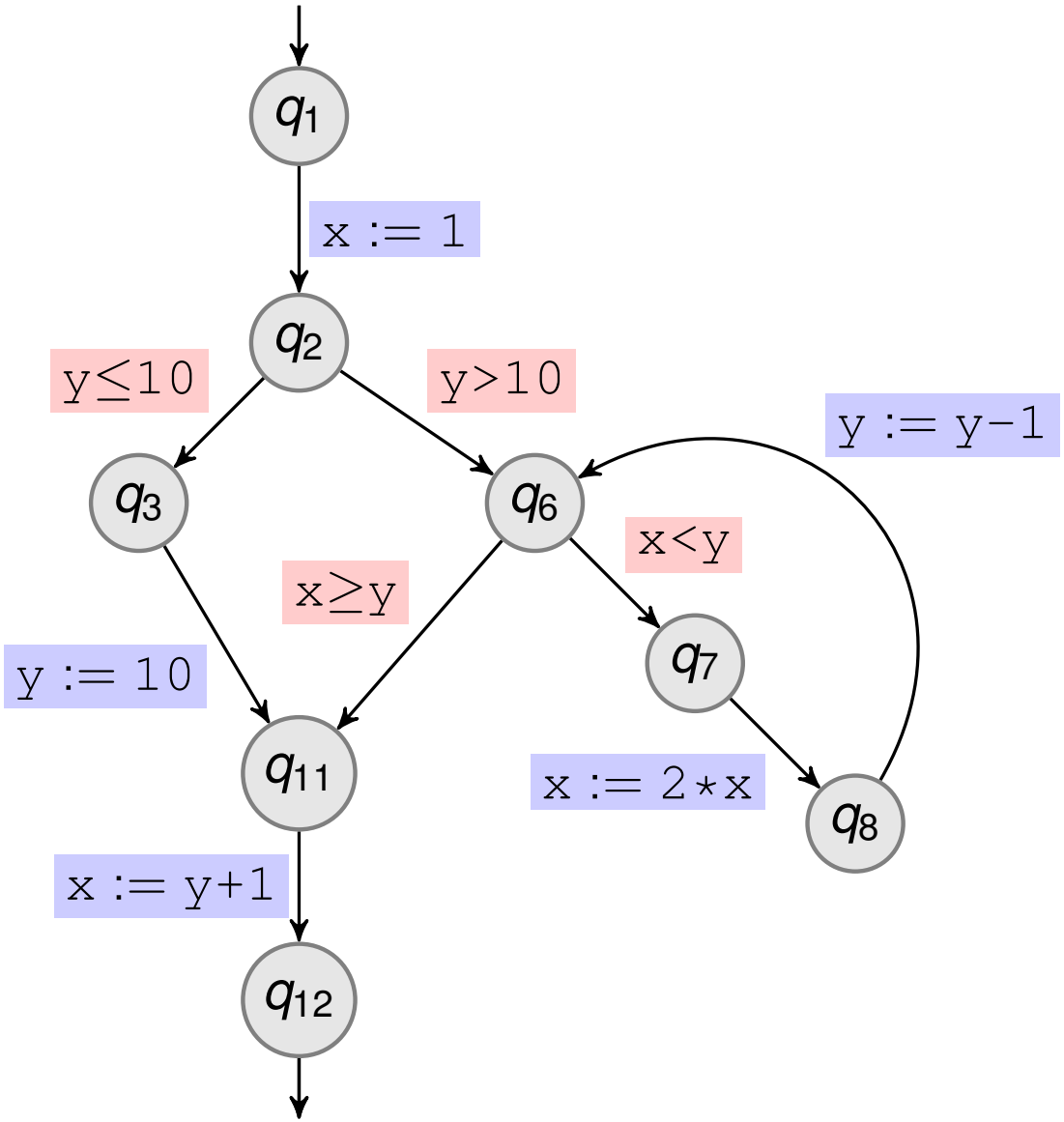


Show that $x > 0$ at q_{12}

	x		y	
q_1	$-\infty$	$+\infty$	$-\infty$	$+\infty$
q_2	1	1	$+\infty$	$+\infty$
q_3	1	1	$-\infty$	10
q_6	1	$+\infty$	$-\infty$	$+\infty$
q_7	1	$+\infty$	$-\infty$	$+\infty$
q_8	2	$+\infty$	$-\infty$	$+\infty$
q_{11}	1	1	10	10
q_{12}	11	11	10	10

$$(2,2) \quad \nabla \quad (2,+\infty) \quad = \quad (2,+\infty)$$
$$(10,+\infty) \quad \nabla \quad (-\infty,+\infty) \quad = \quad (-\infty,+\infty)$$

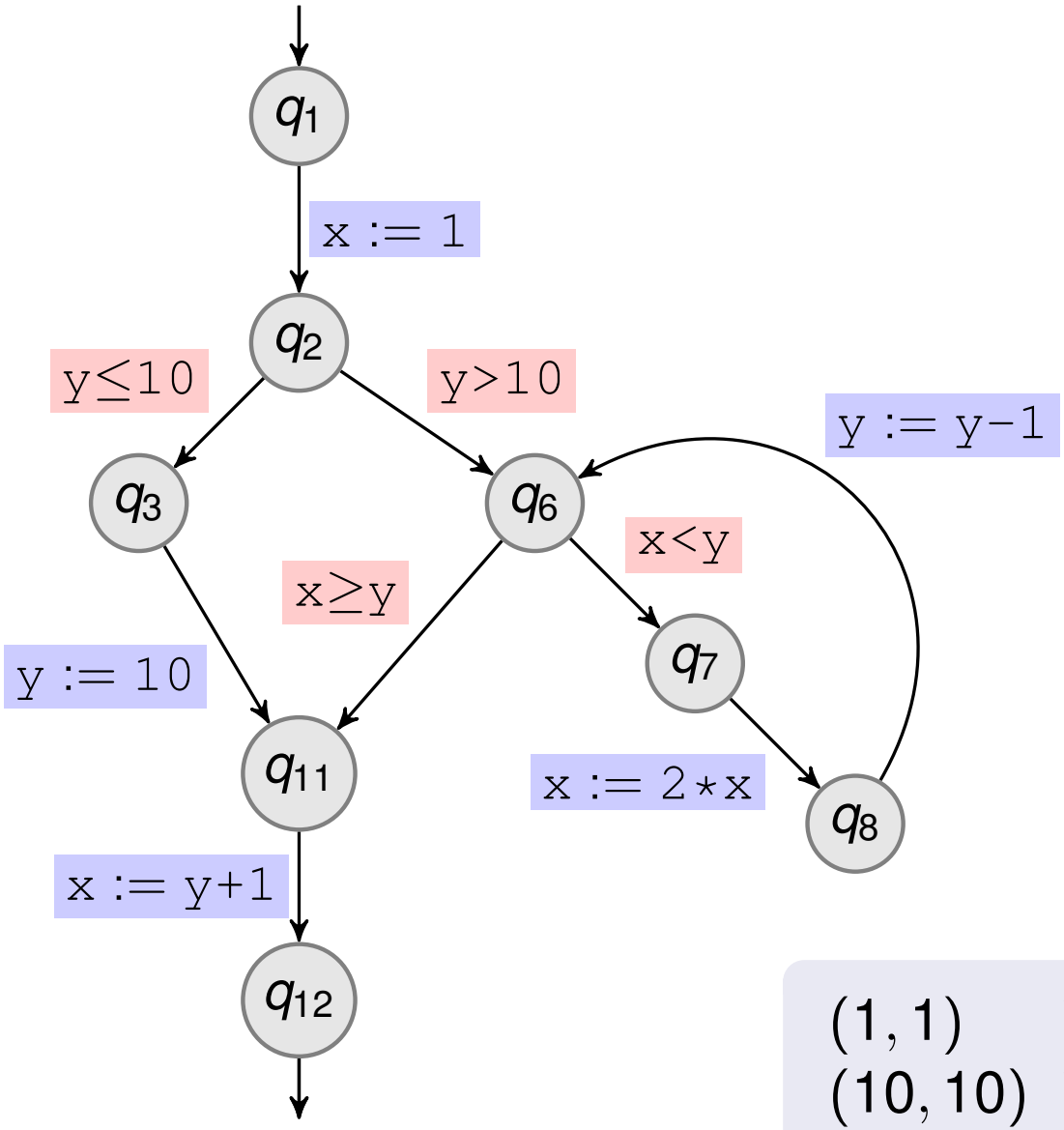
Range Analysis on Example with Widening



Show that $x > 0$ at q_{12}

	x		y	
q_1	$-\infty$	$+\infty$	$-\infty$	$+\infty$
q_2	1	1	$+\infty$	$+\infty$
q_3	1	1	$-\infty$	10
q_6	1	$+\infty$	$-\infty$	$+\infty$
q_7	1	$+\infty$	$-\infty$	$+\infty$
q_8	2	$+\infty$	$-\infty$	$+\infty$
q_{11}	1	1	10	10
q_{12}	11	11	10	10

Range Analysis on Example with Widening

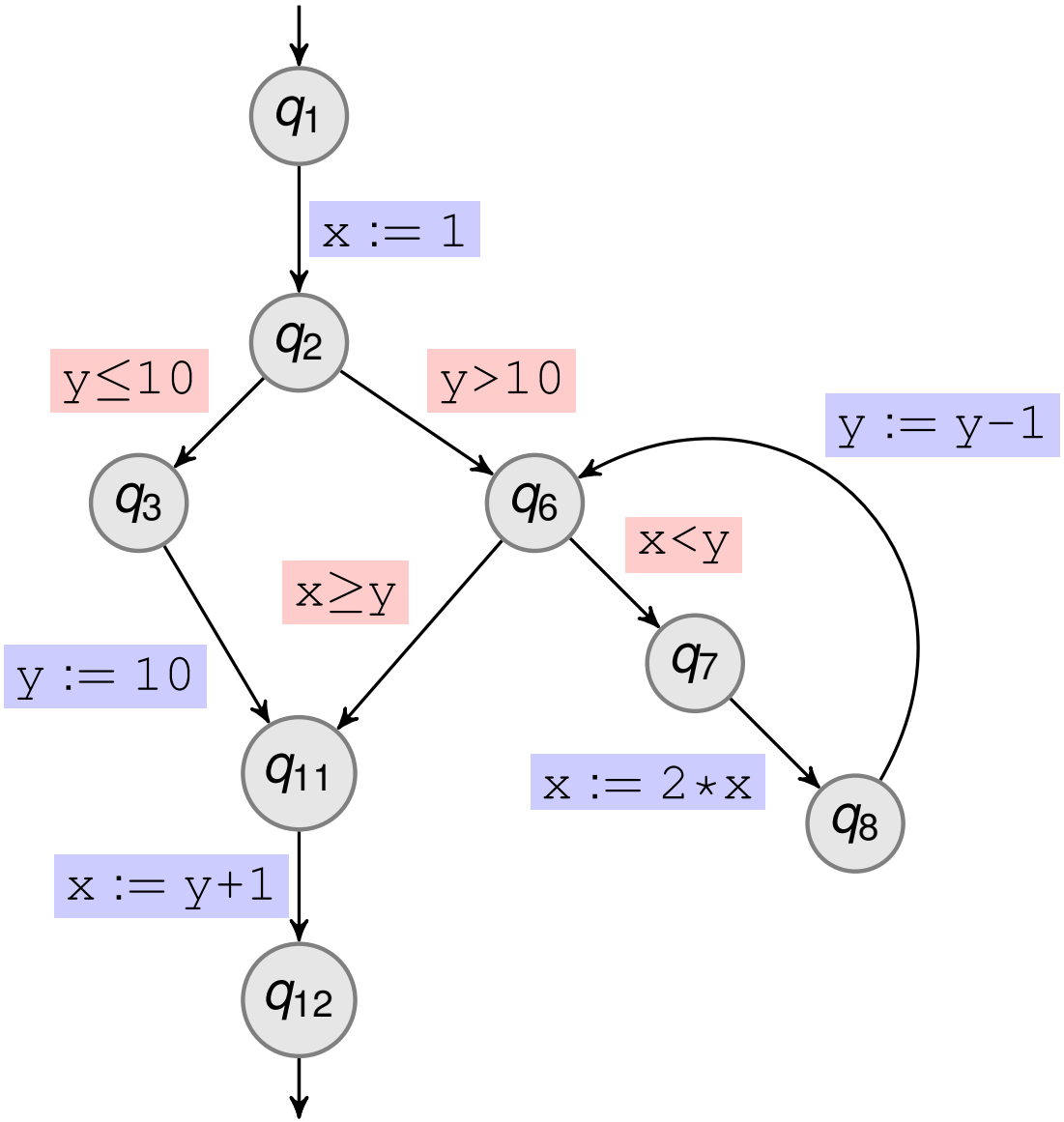


Show that $x > 0$ at q_{12}

	x		y	
q_1	$-\infty$	$+\infty$	$-\infty$	$+\infty$
q_2	1	1	$+\infty$	$+\infty$
q_3	1	1	$-\infty$	10
q_6	1	$+\infty$	$-\infty$	$+\infty$
q_7	1	$+\infty$	$-\infty$	$+\infty$
q_8	2	$+\infty$	$-\infty$	$+\infty$
q_{11}	1	$+\infty$	$-\infty$	$+\infty$
q_{12}	11	11	10	10

$$(1,1) \quad \nabla \quad (1,+\infty) \quad = \quad (1,+\infty)$$
$$(10,10) \quad \nabla \quad (-\infty,+\infty) \quad = \quad (-\infty,+\infty)$$

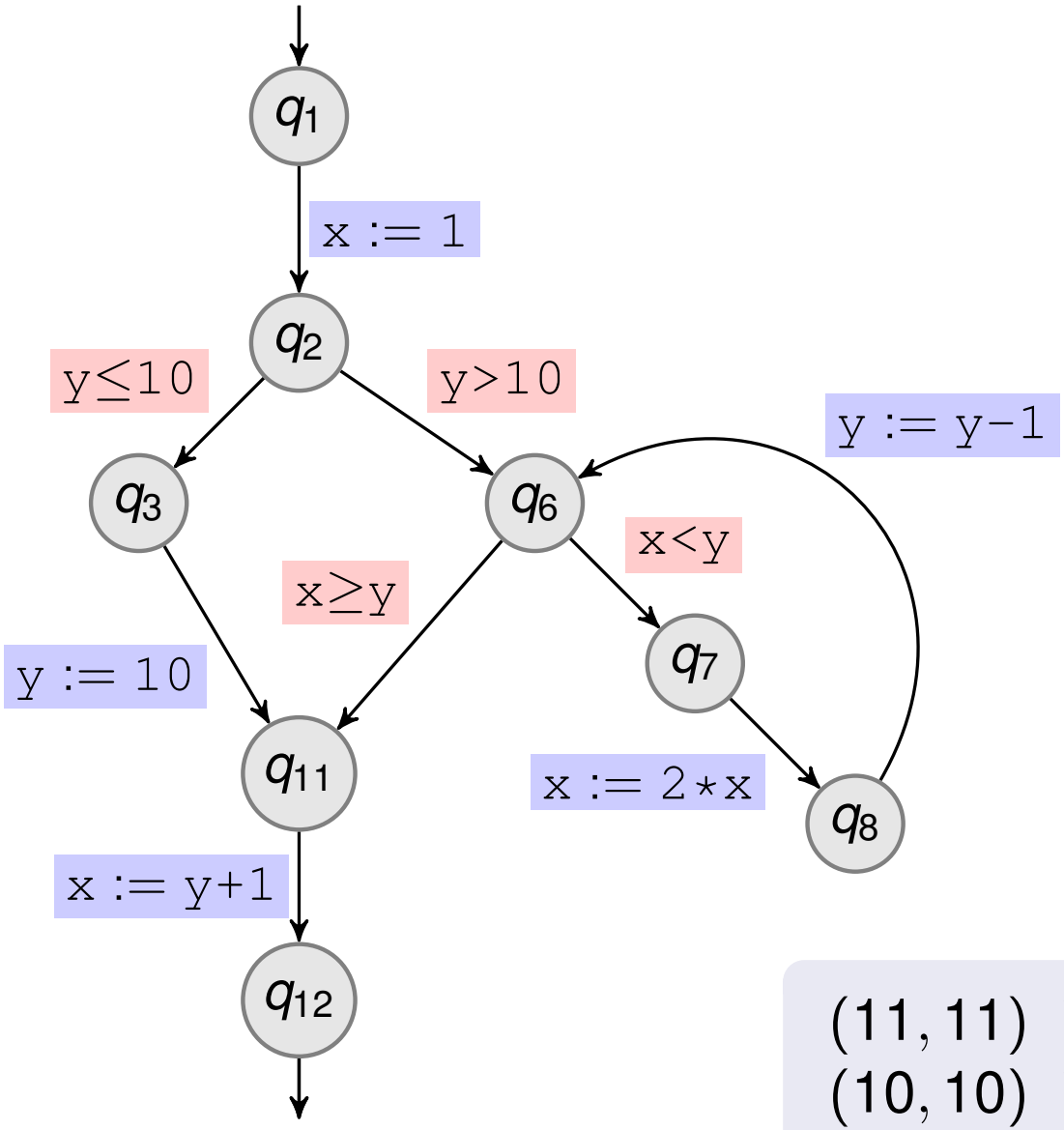
Range Analysis on Example with Widening



Show that $x > 0$ at q_{12}

	x		y	
q_1	$-\infty$	$+\infty$	$-\infty$	$+\infty$
q_2	1	1	$+\infty$	$+\infty$
q_3	1	1	$-\infty$	10
q_6	1	$+\infty$	$-\infty$	$+\infty$
q_7	1	$+\infty$	$-\infty$	$+\infty$
q_8	2	$+\infty$	$-\infty$	$+\infty$
q_{11}	1	$+\infty$	$-\infty$	$+\infty$
q_{12}	11	11	10	10

Range Analysis on Example with Widening

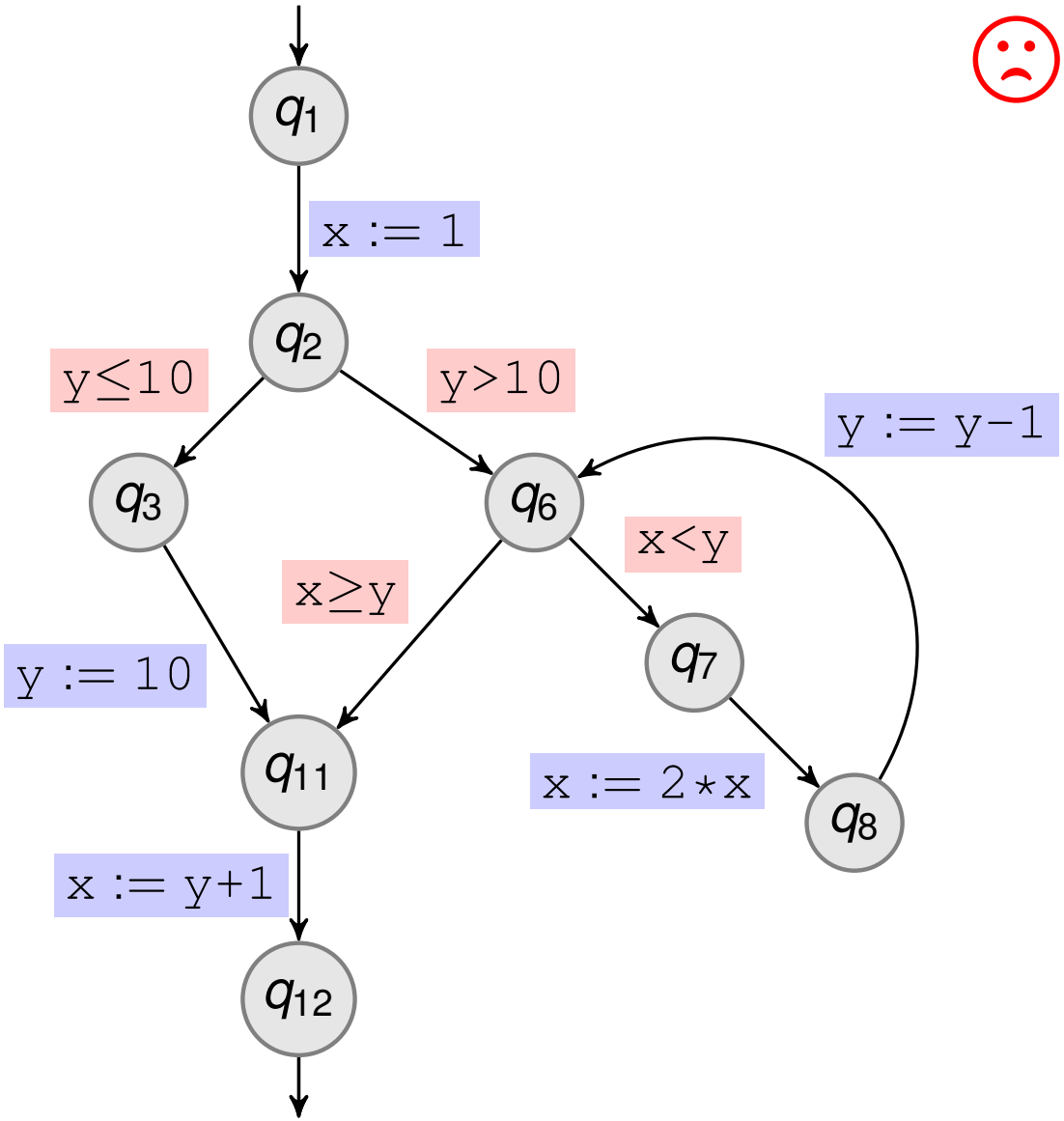


Show that $x > 0$ at q_{12}

	x		y	
q_1	$-\infty$	$+\infty$	$-\infty$	$+\infty$
q_2	1	1	$+\infty$	$+\infty$
q_3	1	1	$-\infty$	10
q_6	1	$+\infty$	$-\infty$	$+\infty$
q_7	1	$+\infty$	$-\infty$	$+\infty$
q_8	2	$+\infty$	$-\infty$	$+\infty$
q_{11}	1	$+\infty$	$-\infty$	$+\infty$
q_{12}	$-\infty$	$+\infty$	$-\infty$	$+\infty$

$$(11,11) \ \nabla \ (-\infty,+\infty) = (-\infty,+\infty)$$
$$(10,10) \ \nabla \ (-\infty,+\infty) = (-\infty,+\infty)$$

Range Analysis on Example with Widening

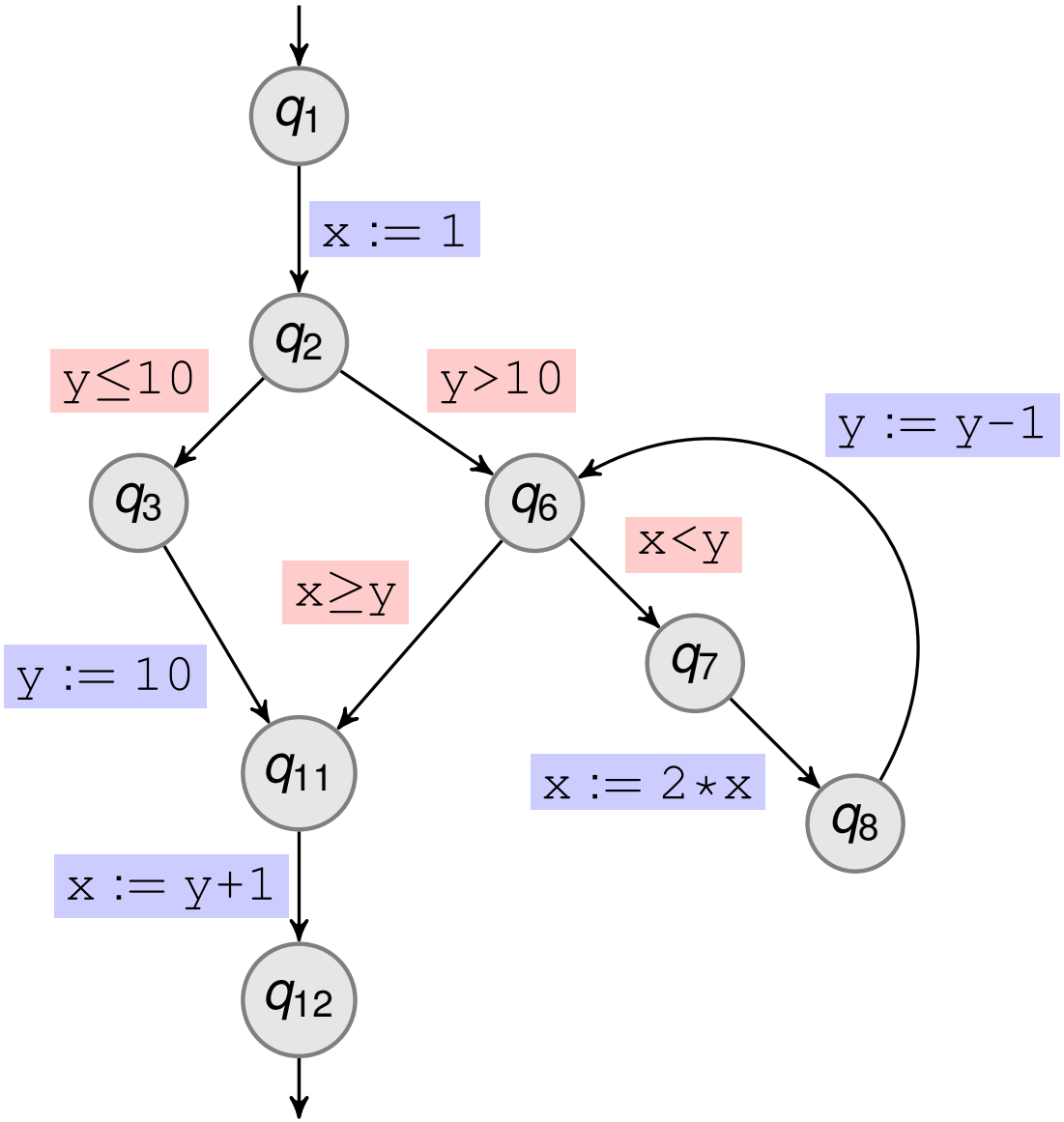


Show that $x > 0$ at q_{12}

	x		y	
q_1	$-\infty$	$+\infty$	$-\infty$	$+\infty$
q_2	1	1	$+\infty$	$+\infty$
q_3	1	1	$-\infty$	10
q_6	1	$+\infty$	$-\infty$	$+\infty$
q_7	1	$+\infty$	$-\infty$	$+\infty$
q_8	2	$+\infty$	$-\infty$	$+\infty$
q_{11}	1	$+\infty$	$-\infty$	$+\infty$
q_{12}	$-\infty$	$+\infty$	$-\infty$	$+\infty$

Too coarse!

Range Analysis on Example with Delayed Widening

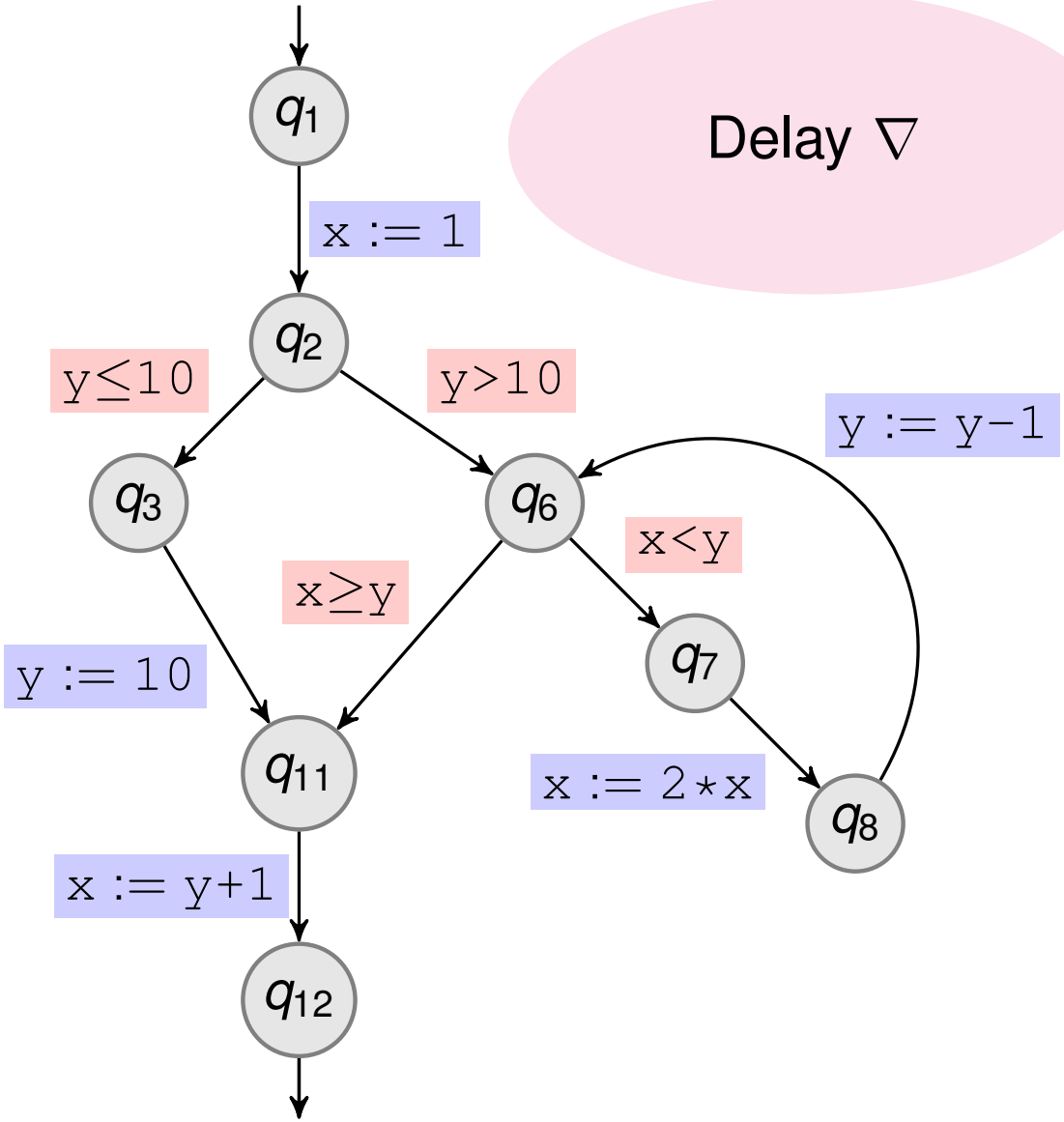


Show that $x > 0$ at q_{12}

Delayed Widening

1. Keep $\sqcup$ for the first iterations
2. Track number of "updates" for each location
3. Switch to ∇ after a suitable "delay"

Range Analysis on Example with Delayed Widening

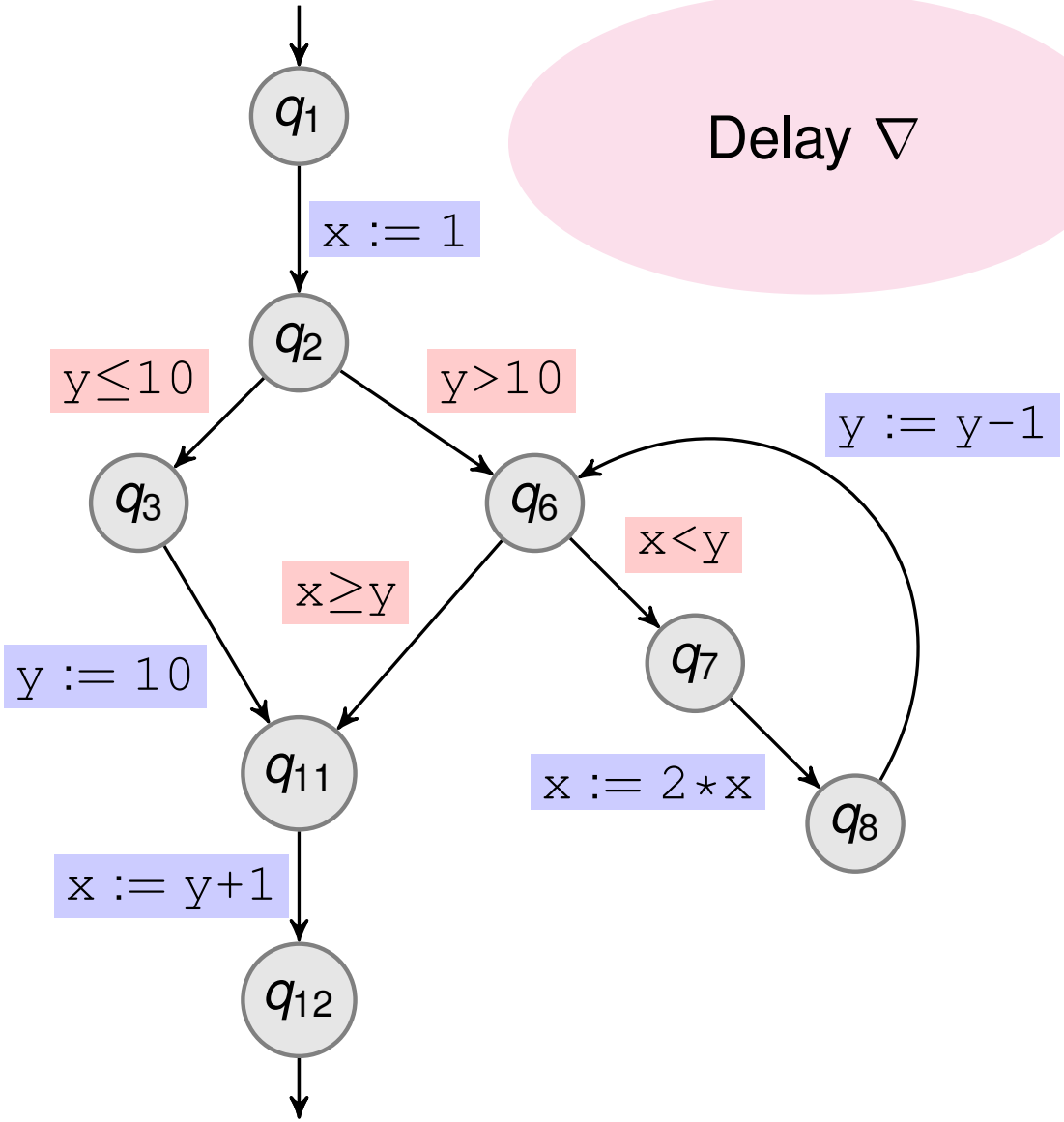


Delay ∇

Show that $x > 0$ at q_{12}

	x		y	
q_1	$-\infty$	$+\infty$	$-\infty$	$+\infty$
q_2	$\bot$	$\bot$	$\bot$	$\bot$
q_3	$\bot$	$\bot$	$\bot$	$\bot$
q_6	$\bot$	$\bot$	$\bot$	$\bot$
q_7	$\bot$	$\bot$	$\bot$	$\bot$
q_8	$\bot$	$\bot$	$\bot$	$\bot$
q_{11}	$\bot$	$\bot$	$\bot$	$\bot$
q_{12}	$\bot$	$\bot$	$\bot$	$\bot$

Range Analysis on Example with Delayed Widening



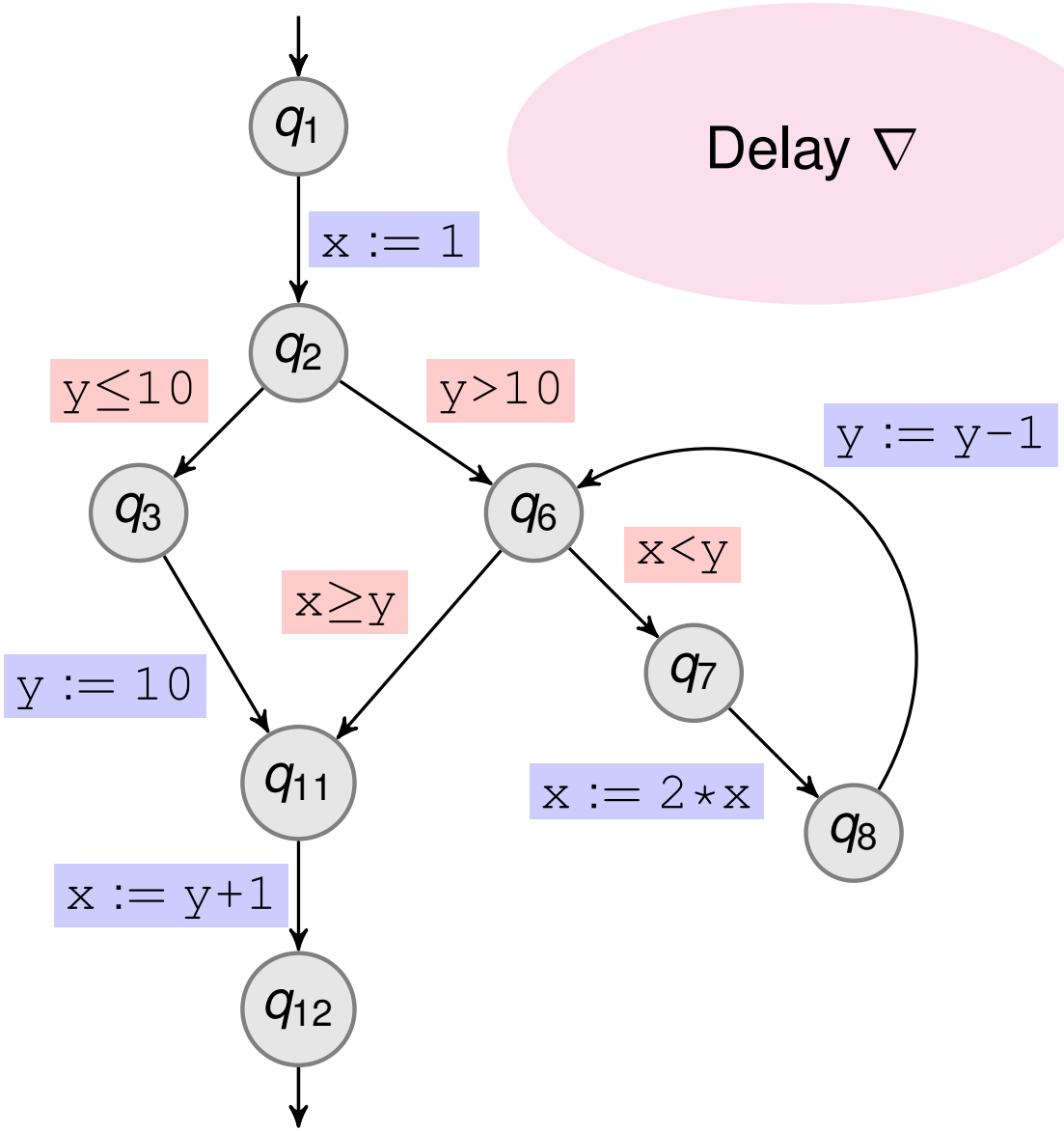
Delay ∇

Show that $x > 0$ at q_{12}

	x		y	
q_1	$-\infty$	$+\infty$	$-\infty$	$+\infty$
q_2	1	1	$+\infty$	$+\infty$
q_3	1	1	$-\infty$	10
q_6	1	1	10	$+\infty$
q_7	1	1	10	$+\infty$
q_8	2	2	10	$+\infty$
q_{11}	1	1	10	10
q_{12}	11	11	10	10

Same as without ∇

Range Analysis on Example with Delayed Widening

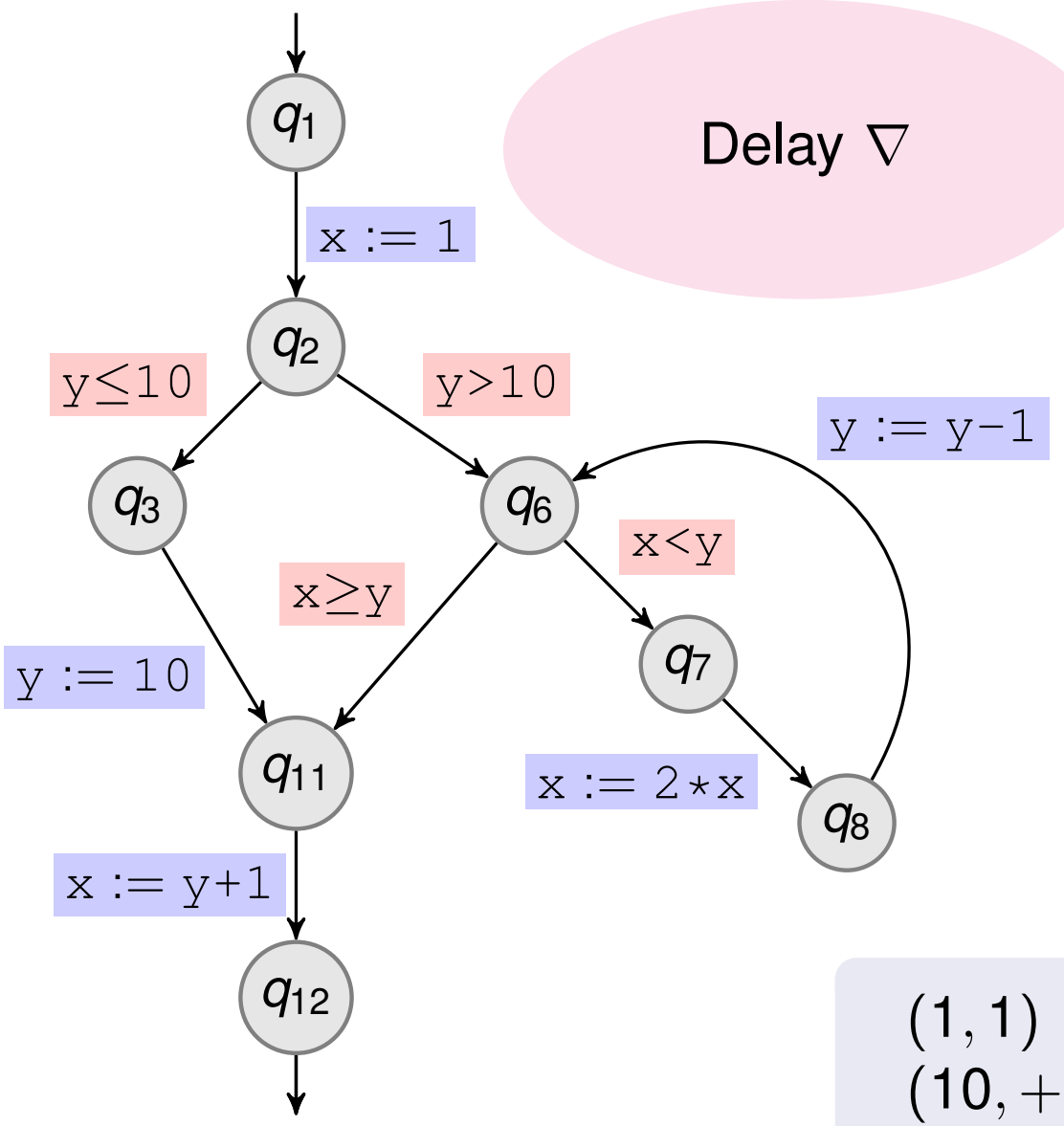


Delay ∇

Show that $x > 0$ at q_{12}

	x		y	
q_1	$-\infty$	$+\infty$	$-\infty$	$+\infty$
q_2	1	1	$+\infty$	$+\infty$
q_3	1	1	$-\infty$	10
q_6	1	1	10	$+\infty$
q_7	1	1	10	$+\infty$
q_8	2	2	10	$+\infty$
q_{11}	1	1	10	10
q_{12}	11	11	10	10

Range Analysis on Example with Delayed Widening



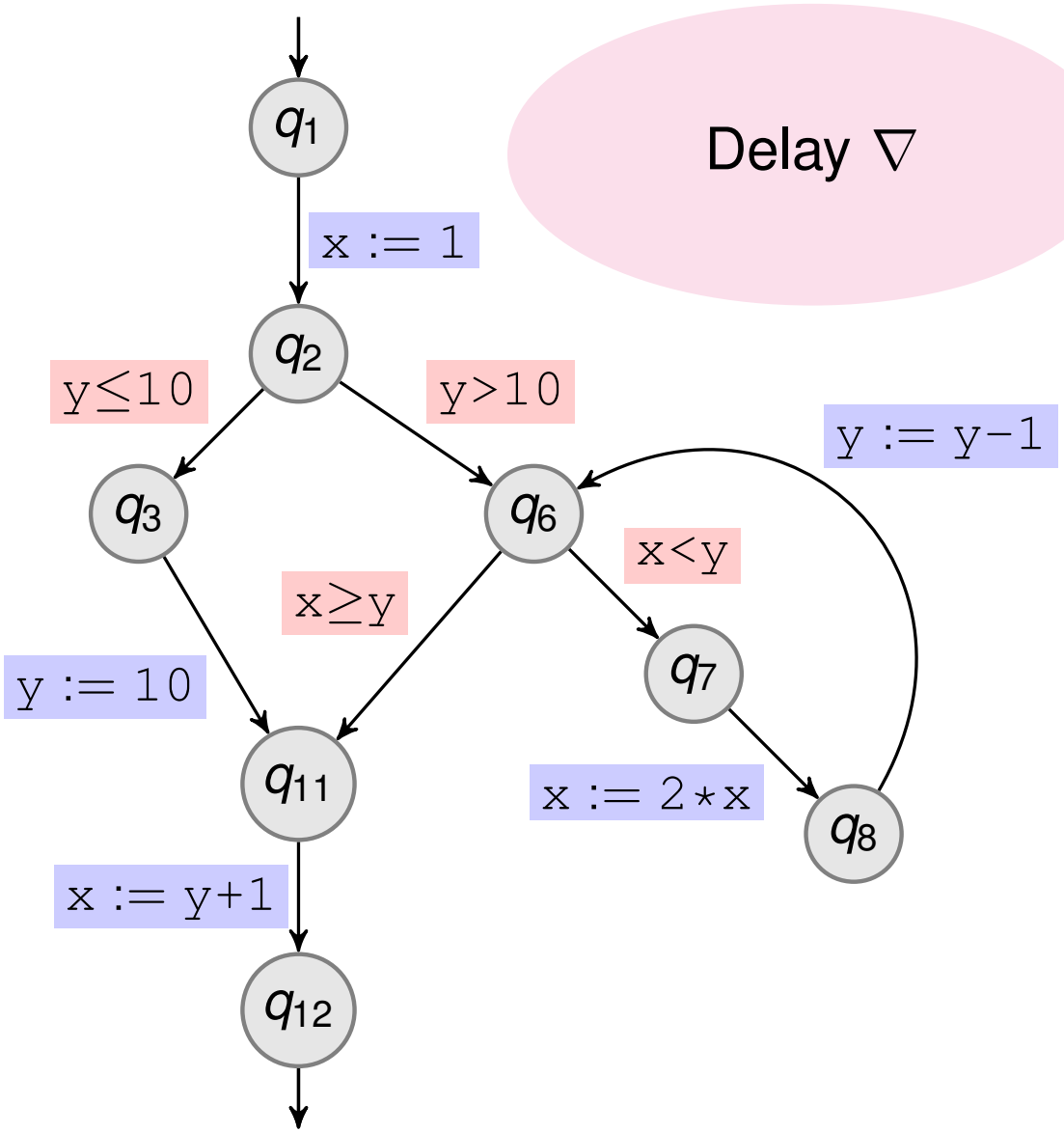
Delay ∇

Show that $\mathtt{x} > 0$ at q_{12}

	x		y	
q_1	$-\infty$	$+\infty$	$-\infty$	$+\infty$
q_2	1	1	$+\infty$	$+\infty$
q_3	1	1	$-\infty$	10
q_6	1	2	9	$+\infty$
q_7	1	1	10	$+\infty$
q_8	2	2	10	$+\infty$
q_{11}	1	1	10	10
q_{12}	11	11	10	10

$$(1,1) \sqcup (2,2) = (1,2)$$
$$(10,+\infty) \sqcup (9,+\infty) = (9,+\infty)$$

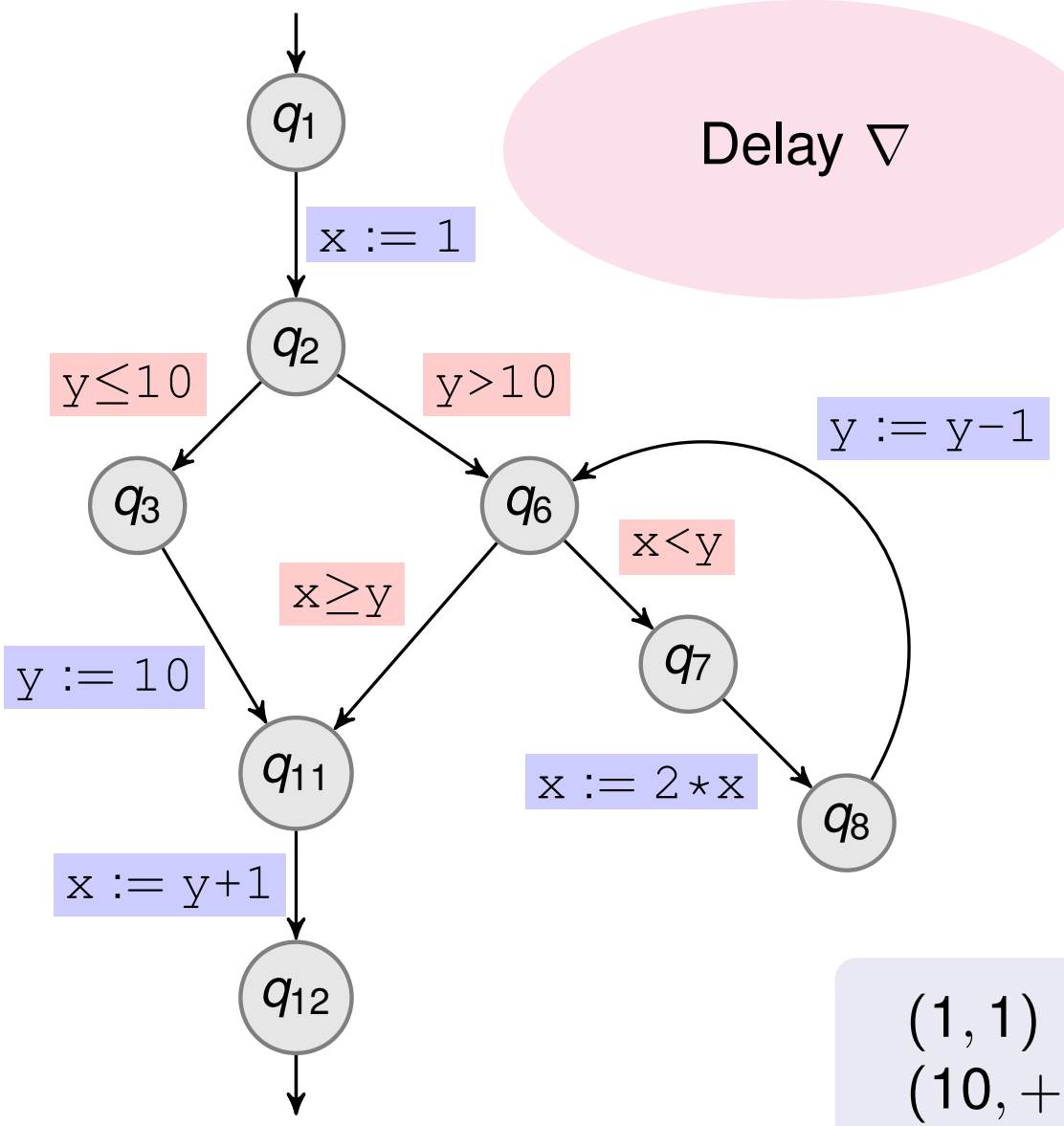
Range Analysis on Example with Delayed Widening



Show that $x > 0$ at q_{12}

	x		y	
q_1	$-\infty$	$+\infty$	$-\infty$	$+\infty$
q_2	1	1	$+\infty$	$+\infty$
q_3	1	1	$-\infty$	10
q_6	1	2	9	$+\infty$
q_7	1	1	10	$+\infty$
q_8	2	2	10	$+\infty$
q_{11}	1	1	10	10
q_{12}	11	11	10	10

Range Analysis on Example with Delayed Widening



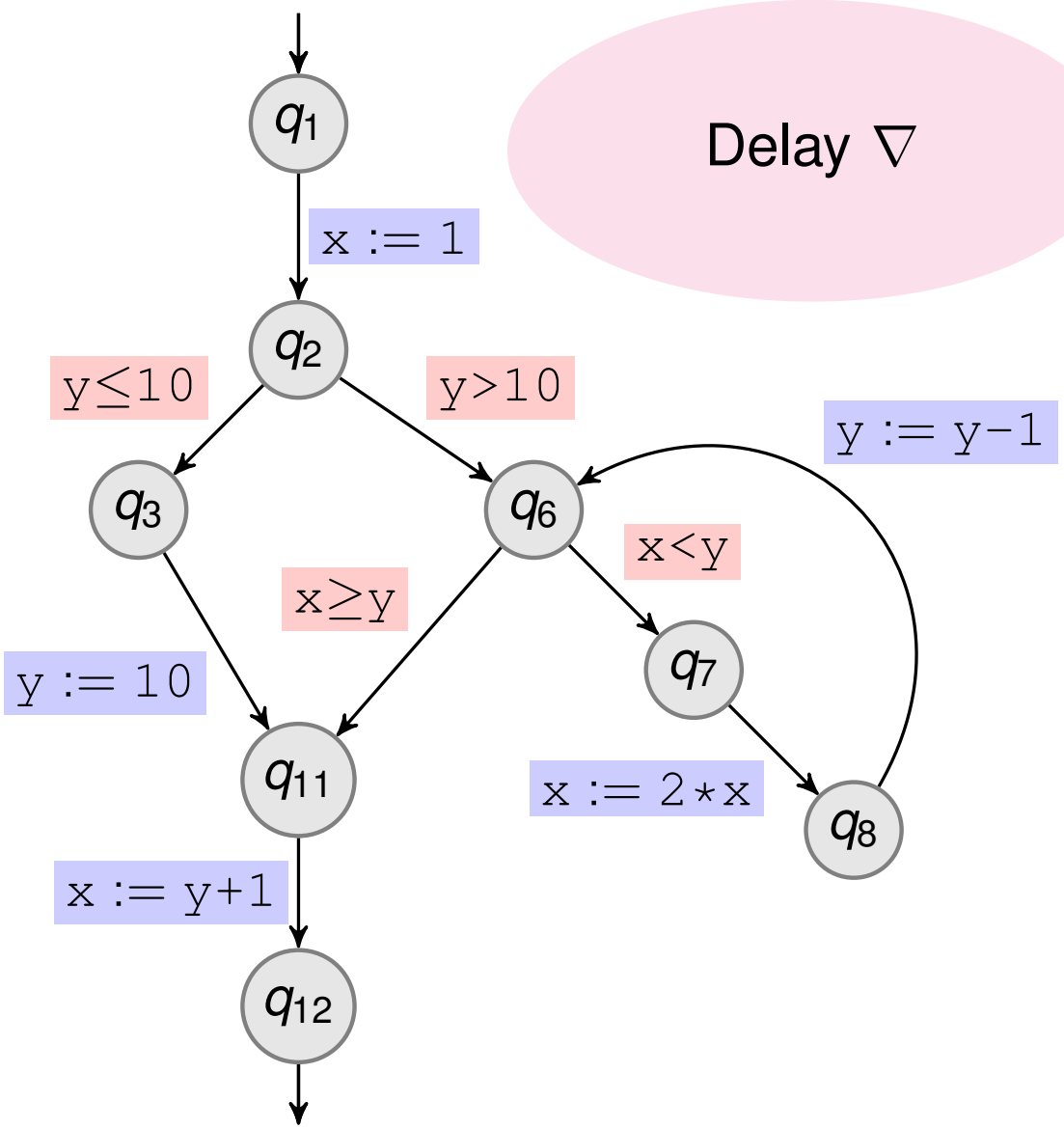
Delay ∇

Show that $\mathtt{x} > 0$ at q_{12}

	x		y	
q_1	$-\infty$	$+\infty$	$-\infty$	$+\infty$
q_2	1	1	$+\infty$	$+\infty$
q_3	1	1	$-\infty$	10
q_6	1	2	9	$+\infty$
q_7	1	2	9	$+\infty$
q_8	2	2	10	$+\infty$
q_{11}	1	1	10	10
q_{12}	11	11	10	10

$$(1,1) \sqcup (1,2) = (1,2)$$
$$(10,+\infty) \sqcup (9,+\infty) = (9,+\infty)$$

Range Analysis on Example with Delayed Widening

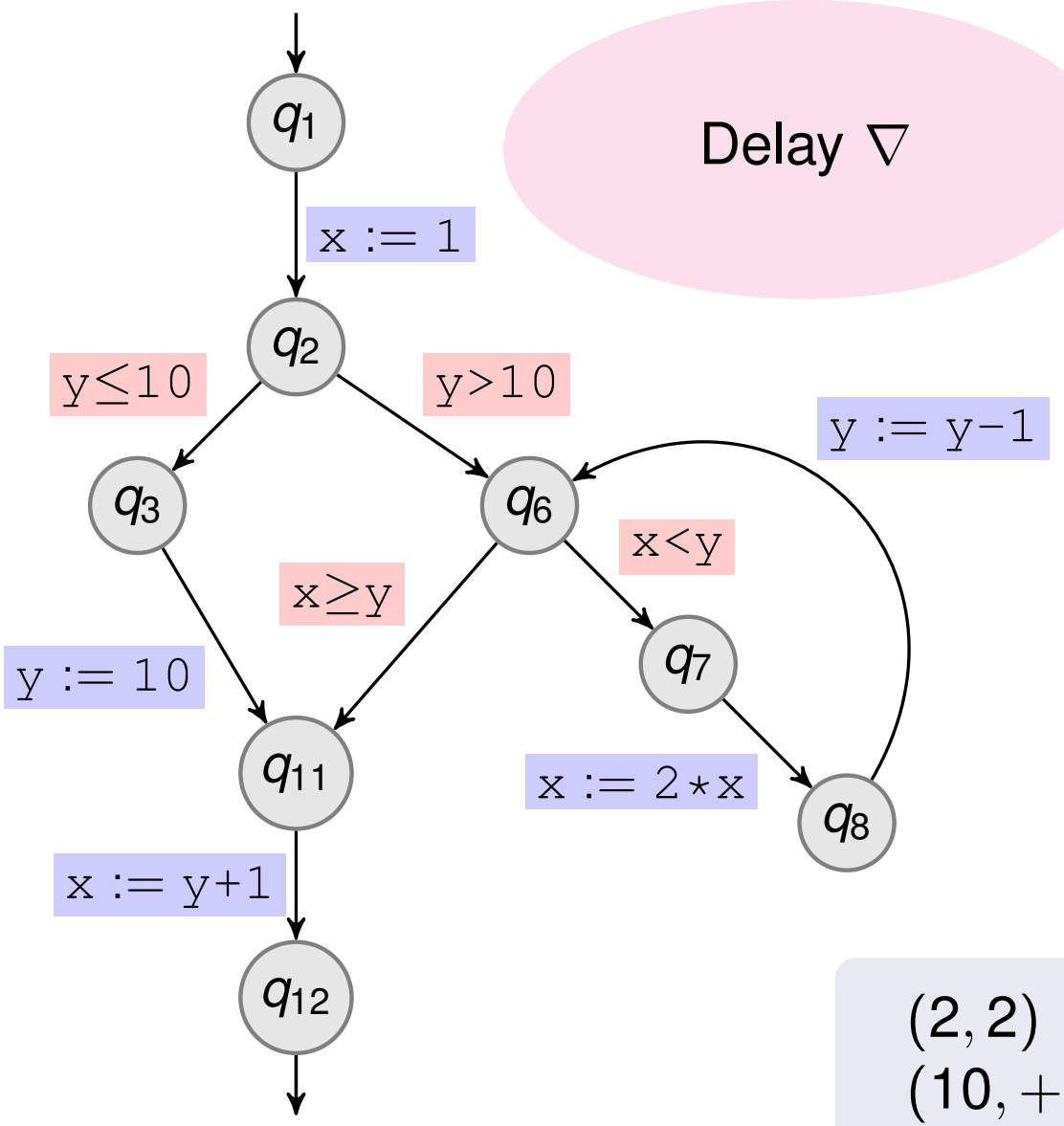


Delay ∇

Show that $x > 0$ at q_{12}

	x		y	
q_1	$-\infty$	$+\infty$	$-\infty$	$+\infty$
q_2	1	1	$+\infty$	$+\infty$
q_3	1	1	$-\infty$	10
q_6	1	2	9	$+\infty$
q_7	1	2	9	$+\infty$
q_8	2	2	10	$+\infty$
q_{11}	1	1	10	10
q_{12}	11	11	10	10

Range Analysis on Example with Delayed Widening



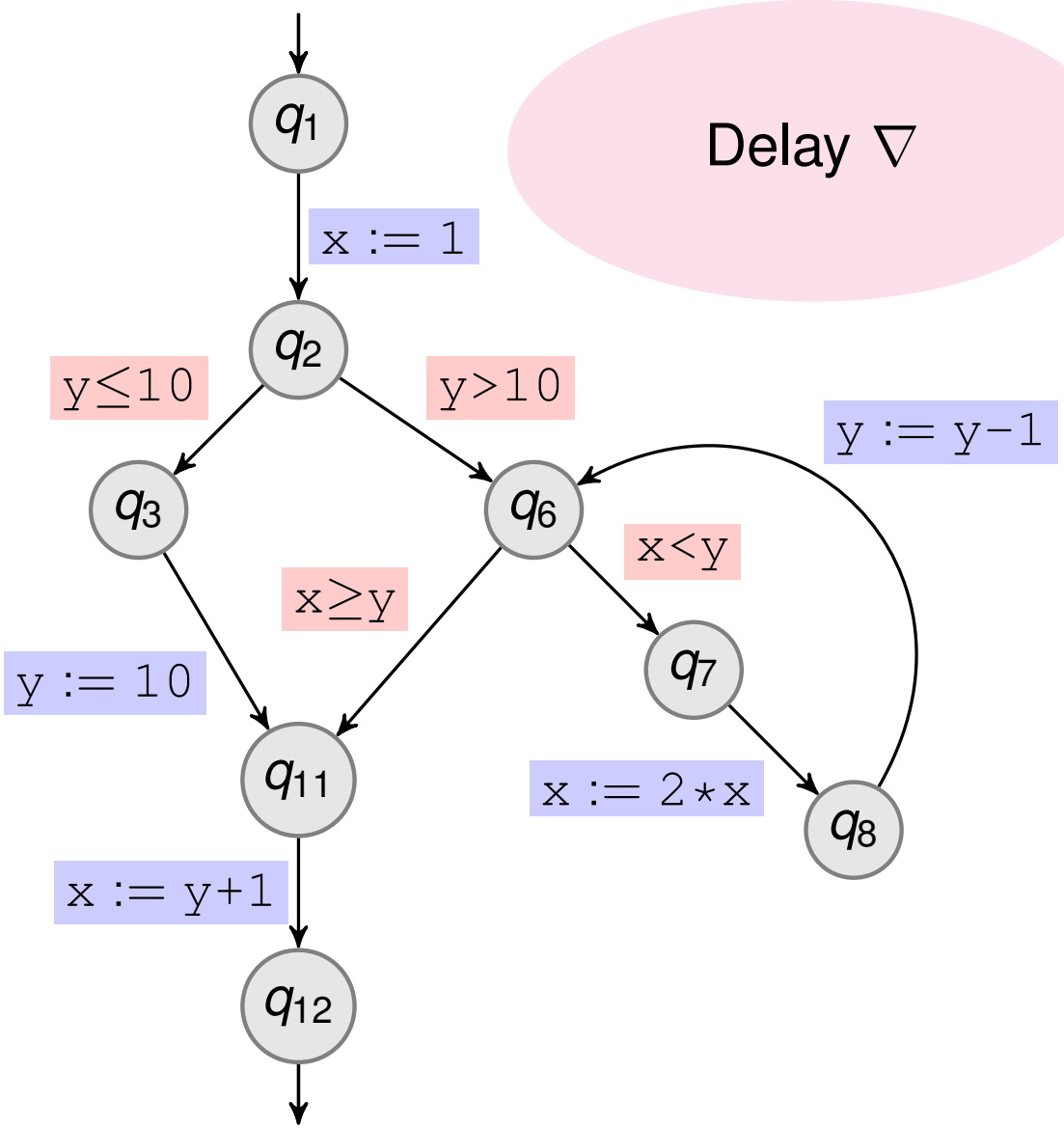
Delay ∇

Show that $x > 0$ at q_{12}

	x		y	
q_1	$-\infty$	$+\infty$	$-\infty$	$+\infty$
q_2	1	1	$+\infty$	$+\infty$
q_3	1	1	$-\infty$	10
q_6	1	2	9	$+\infty$
q_7	1	2	9	$+\infty$
q_8	2	4	9	$+\infty$
q_{11}	1	1	10	10
q_{12}	11	11	10	10

$$(2,2) \sqcup (2,4) = (2,4)$$
$$(10,+\infty) \sqcup (9,+\infty) = (9,+\infty)$$

Range Analysis on Example with Delayed Widening

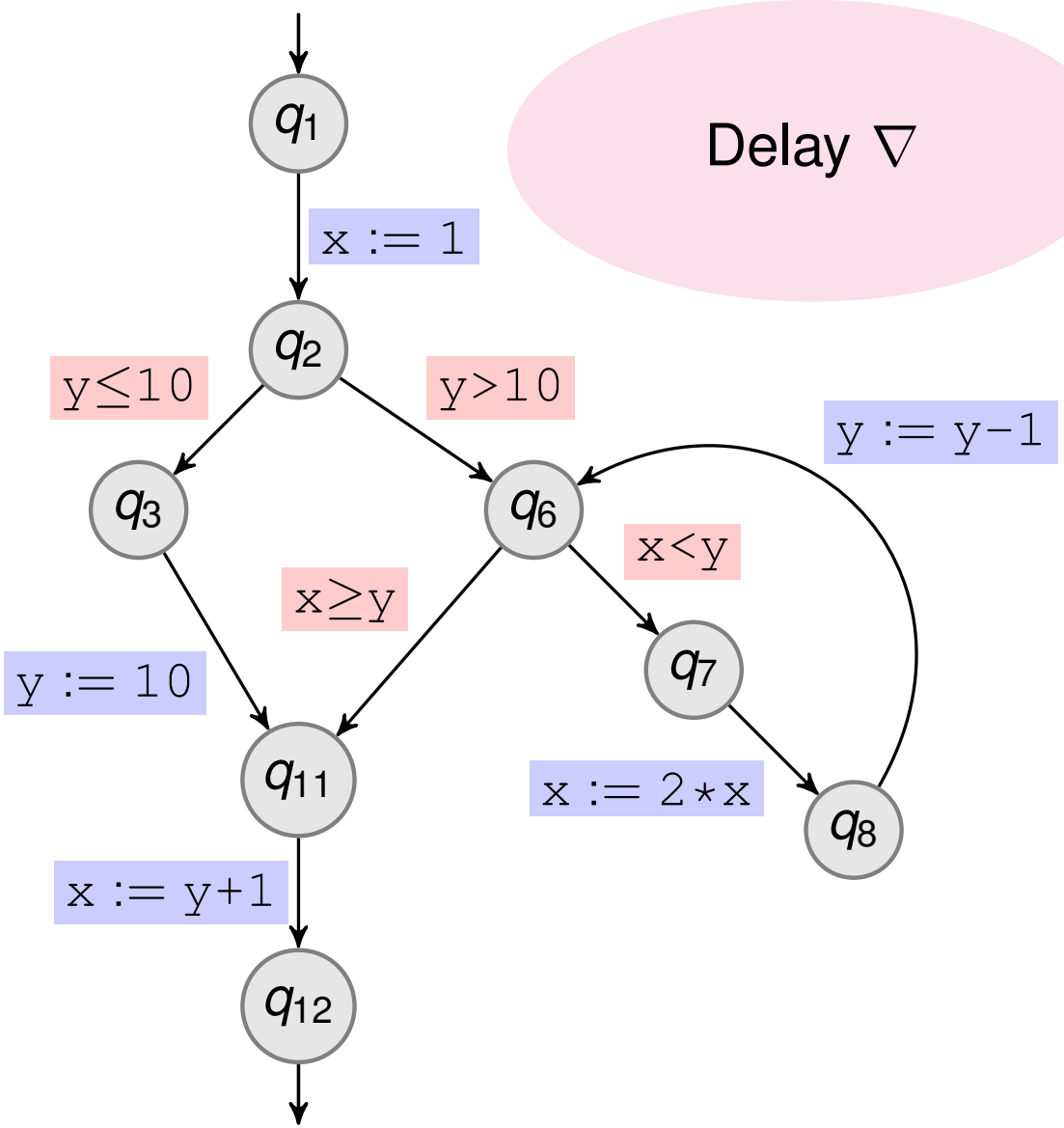


Delay ∇

Show that $x > 0$ at q_{12}

	x		y	
q_1	$-\infty$	$+\infty$	$-\infty$	$+\infty$
q_2	1	1	$+\infty$	$+\infty$
q_3	1	1	$-\infty$	10
q_6	1	2	9	$+\infty$
q_7	1	2	9	$+\infty$
q_8	2	4	9	$+\infty$
q_{11}	1	1	10	10
q_{12}	11	11	10	10

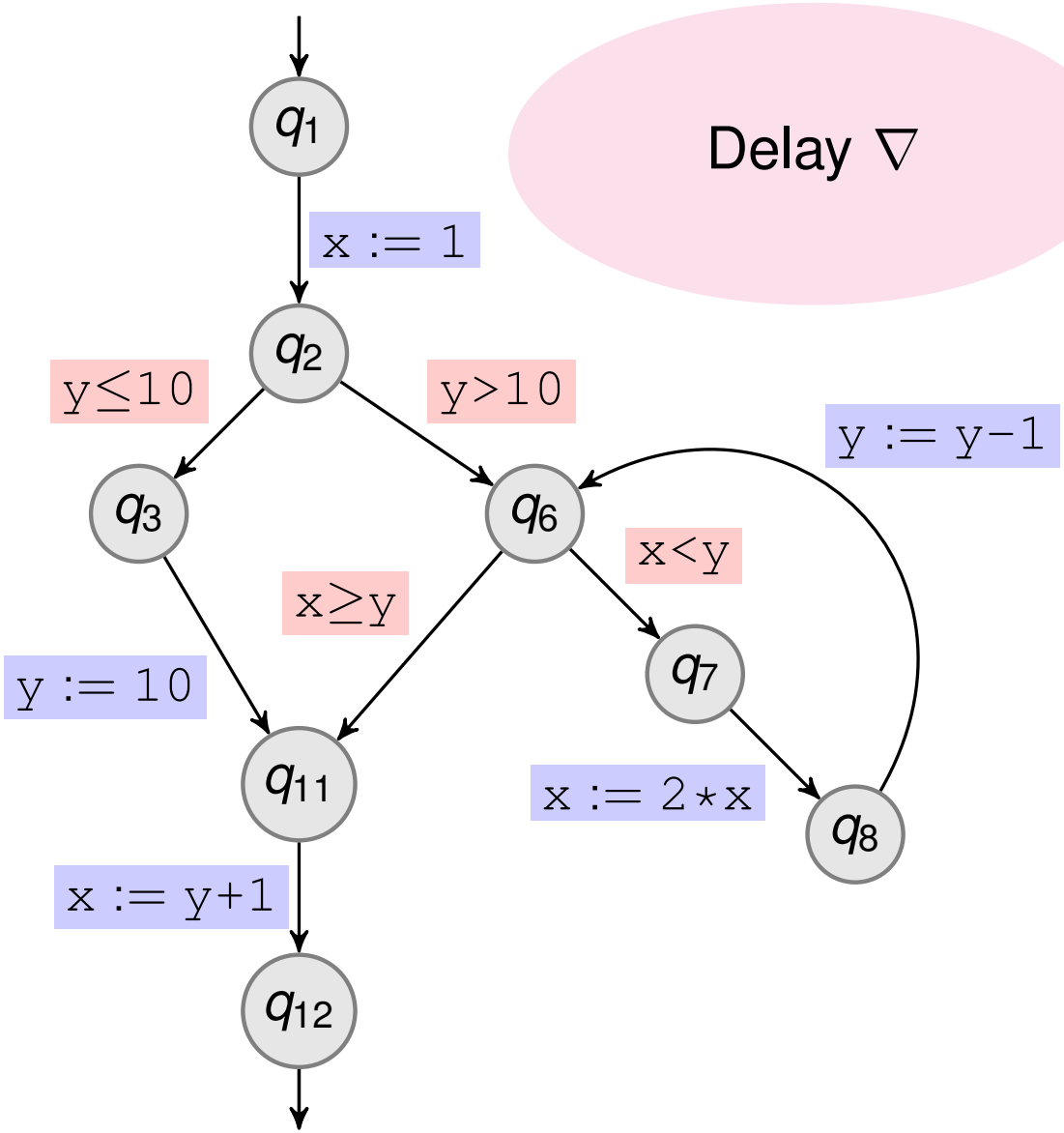
Range Analysis on Example with Delayed Widening



Show that $x > 0$ at q_{12}

	x		y	
q_1	$-\infty$	$+\infty$	$-\infty$	$+\infty$
q_2	1	1	$+\infty$	$+\infty$
q_3	1	1	$-\infty$	10
q_6	1	4	8	$+\infty$
q_7	1	4	8	$+\infty$
q_8	2	8	8	$+\infty$
q_{11}	1	1	10	10
q_{12}	11	11	10	10

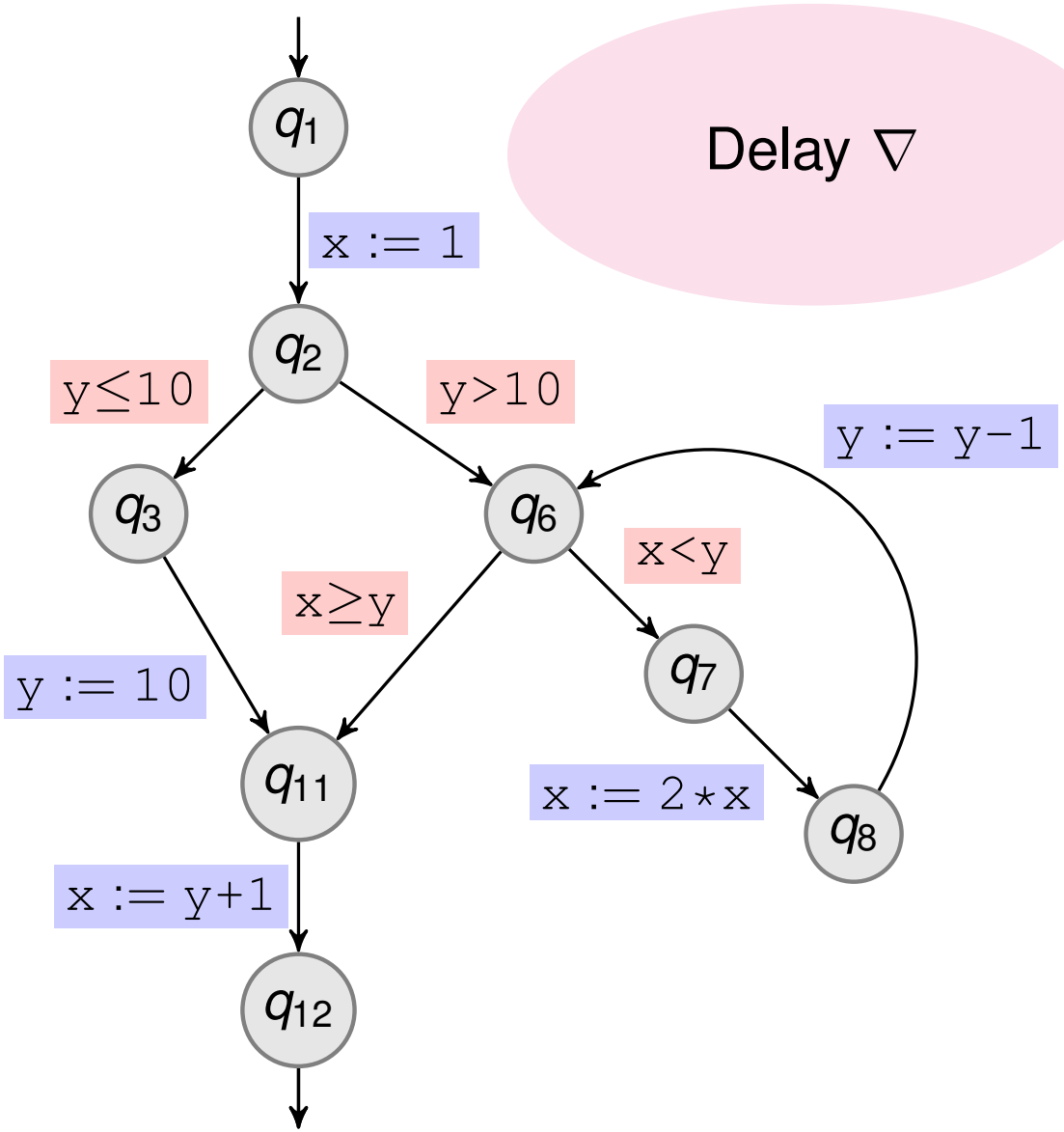
Range Analysis on Example with Delayed Widening



Show that $x > 0$ at q_{12}

	x		y	
q_1	$-\infty$	$+\infty$	$-\infty$	$+\infty$
q_2	1	1	$+\infty$	$+\infty$
q_3	1	1	$-\infty$	10
q_6	1	8	7	$+\infty$
q_7	1	8	7	$+\infty$
q_8	2	16	7	$+\infty$
q_{11}	1	1	10	10
q_{12}	11	11	10	10

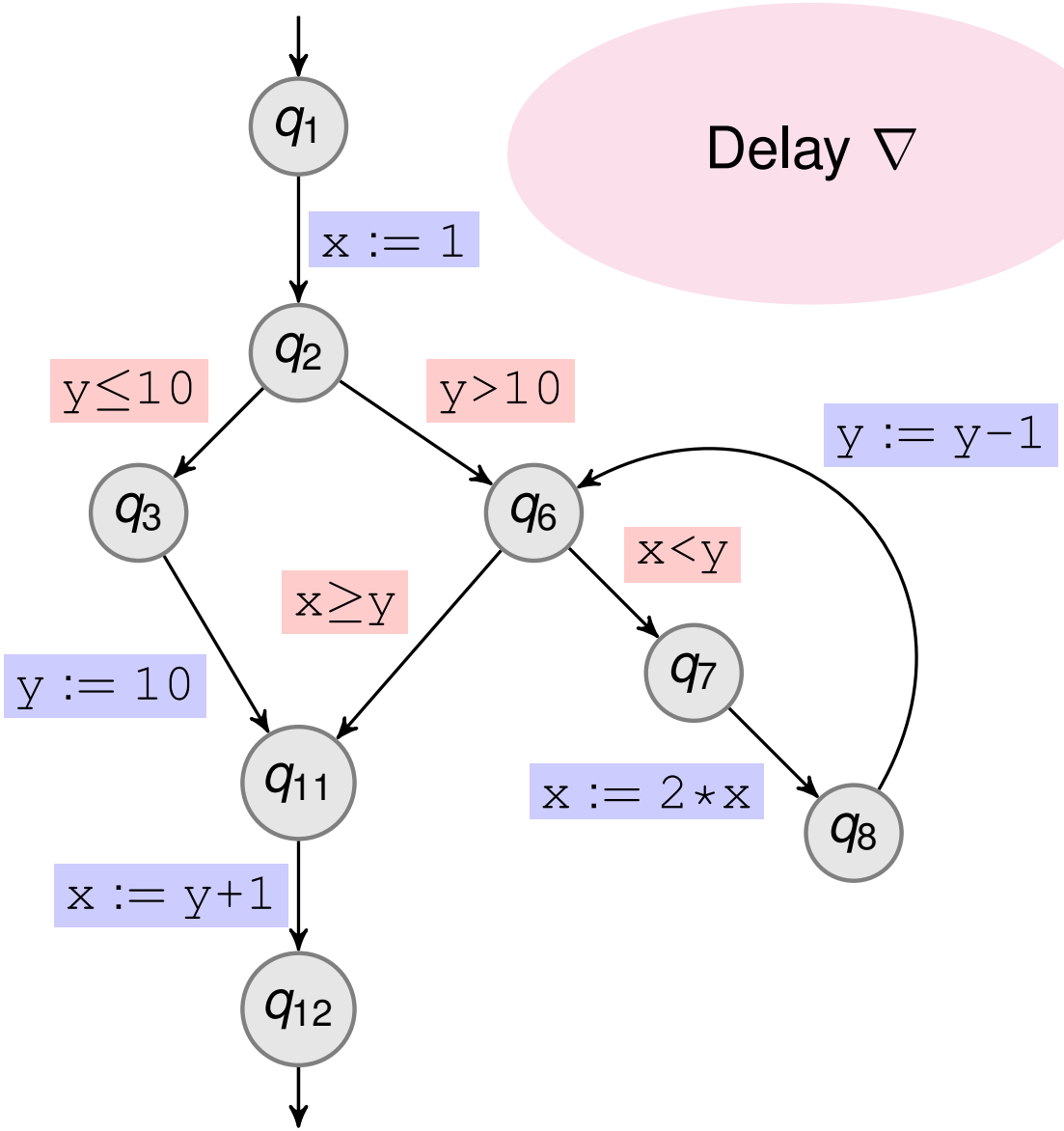
Range Analysis on Example with Delayed Widening



Show that $x > 0$ at q_{12}

	x		y	
q_1	$-\infty$	$+\infty$	$-\infty$	$+\infty$
q_2	1	1	$+\infty$	$+\infty$
q_3	1	1	$-\infty$	10
q_6	1	16	6	$+\infty$
q_7	1	16	6	$+\infty$
q_8	2	32	6	$+\infty$
q_{11}	1	1	10	10
q_{12}	11	11	10	10

Range Analysis on Example with Delayed Widening

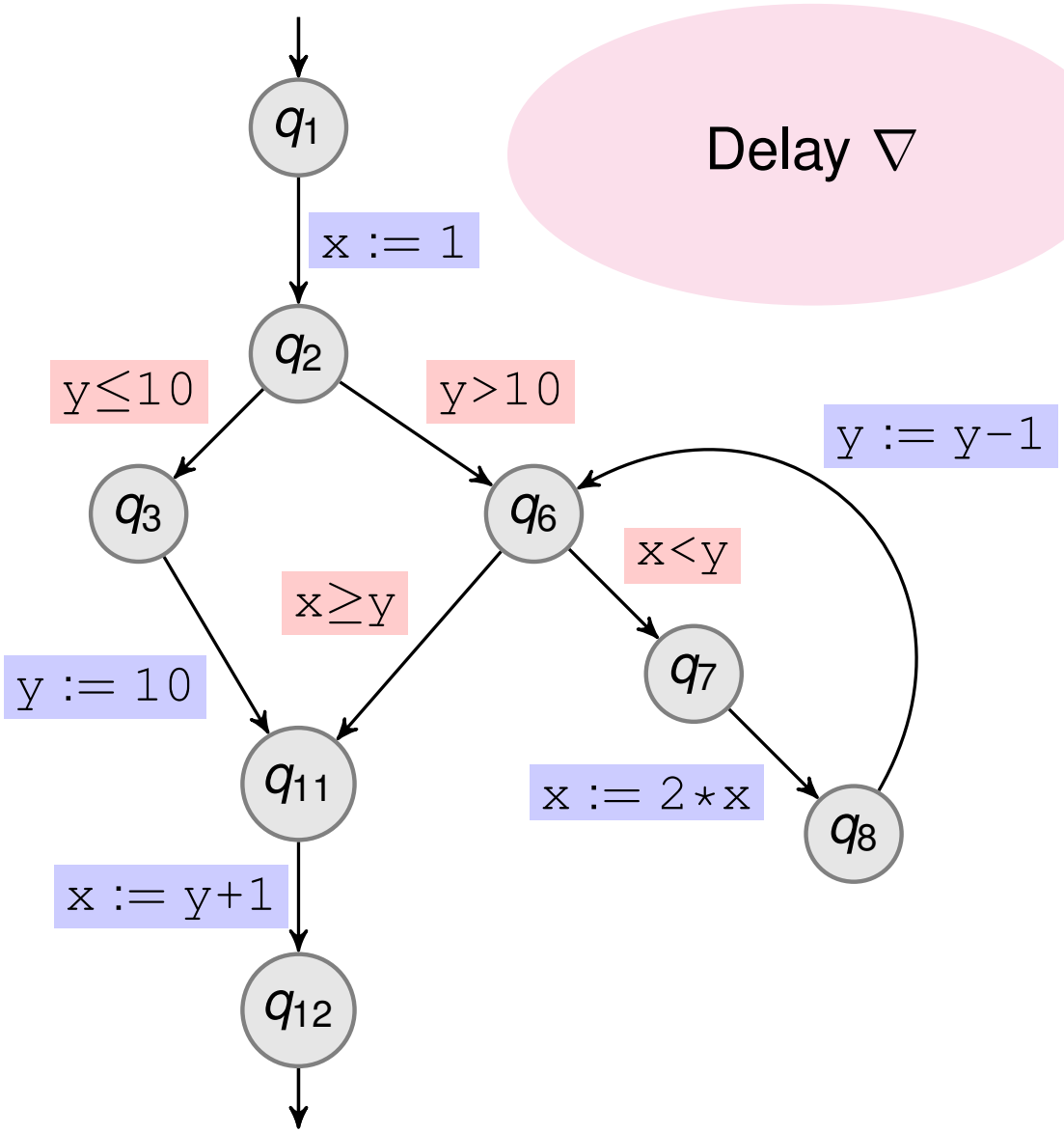


Delay ∇

Show that $x > 0$ at q_{12}

	x		y	
q_1	$-\infty$	$+\infty$	$-\infty$	$+\infty$
q_2	1	1	$+\infty$	$+\infty$
q_3	1	1	$-\infty$	10
q_6	1	32	5	$+\infty$
q_7	1	32	5	$+\infty$
q_8	2	2^6	5	$+\infty$
q_{11}	1	1	10	10
q_{12}	11	11	10	10

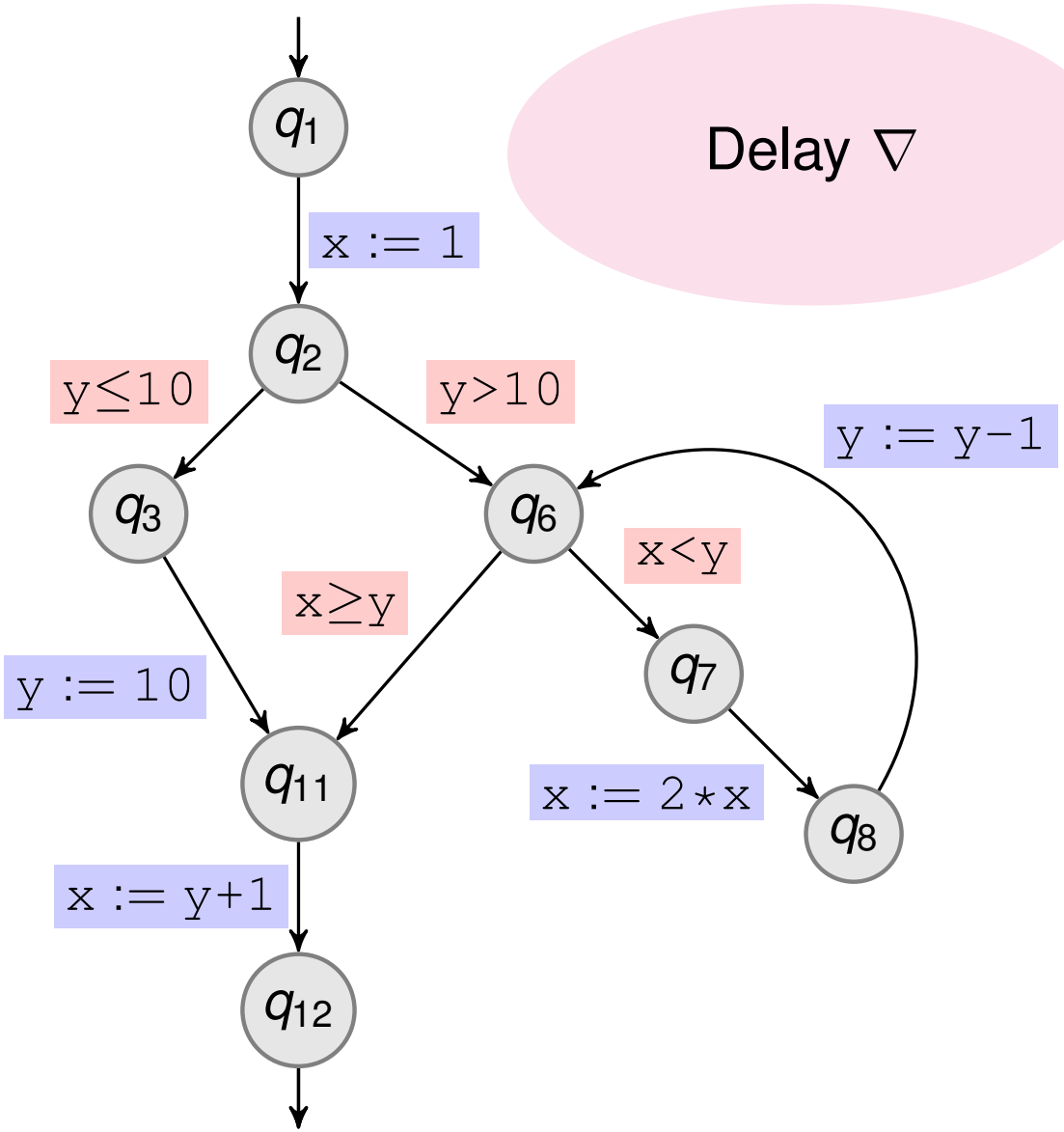
Range Analysis on Example with Delayed Widening



Show that $x > 0$ at q_{12}

	x		y	
q_1	$-\infty$	$+\infty$	$-\infty$	$+\infty$
q_2	1	1	$+\infty$	$+\infty$
q_3	1	1	$-\infty$	10
q_6	1	2^6	4	$+\infty$
q_7	1	2^6	4	$+\infty$
q_8	2	2^7	4	$+\infty$
q_{11}	1	1	10	10
q_{12}	11	11	10	10

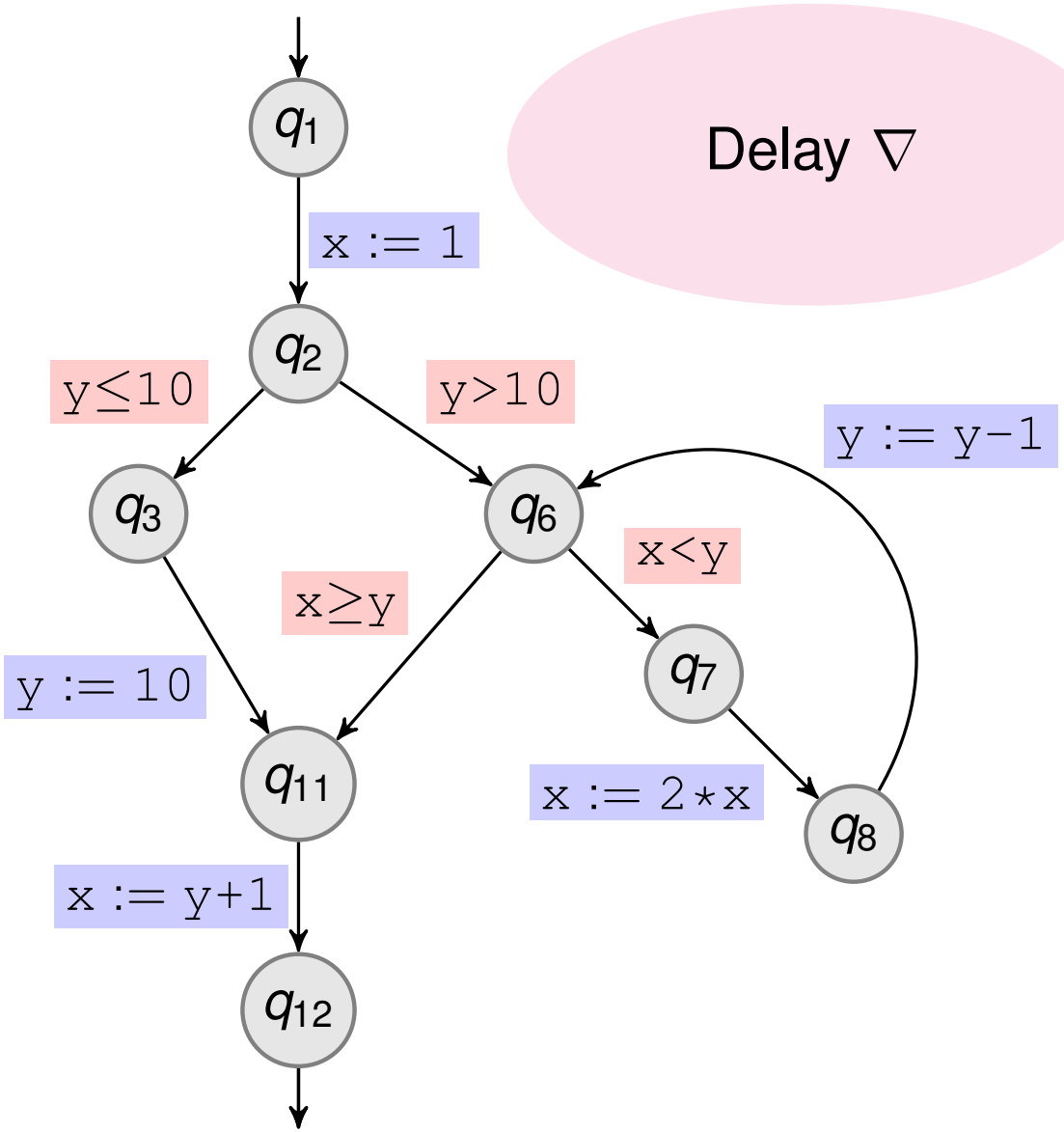
Range Analysis on Example with Delayed Widening



Show that $x > 0$ at q_{12}

	x		y	
q_1	$-\infty$	$+\infty$	$-\infty$	$+\infty$
q_2	1	1	$+\infty$	$+\infty$
q_3	1	1	$-\infty$	10
q_6	1	2^7	3	$+\infty$
q_7	1	2^7	3	$+\infty$
q_8	2	2^8	3	$+\infty$
q_{11}	1	1	10	10
q_{12}	11	11	10	10

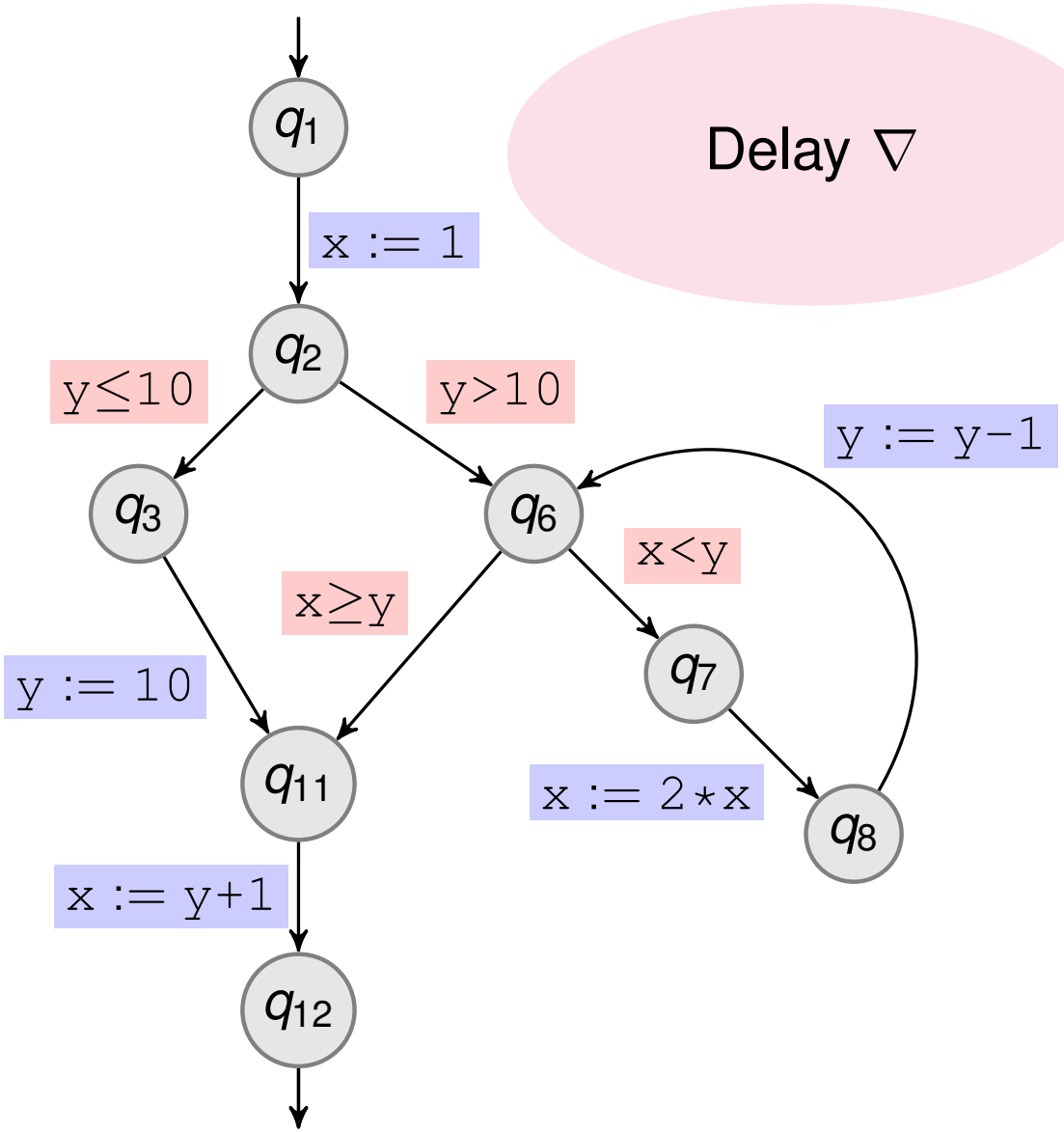
Range Analysis on Example with Delayed Widening



Show that $x > 0$ at q_{12}

	x		y	
q_1	$-\infty$	$+\infty$	$-\infty$	$+\infty$
q_2	1	1	$+\infty$	$+\infty$
q_3	1	1	$-\infty$	10
q_6	1	2^8	2	$+\infty$
q_7	1	2^8	2	$+\infty$
q_8	2	2^9	2	$+\infty$
q_{11}	1	1	10	10
q_{12}	11	11	10	10

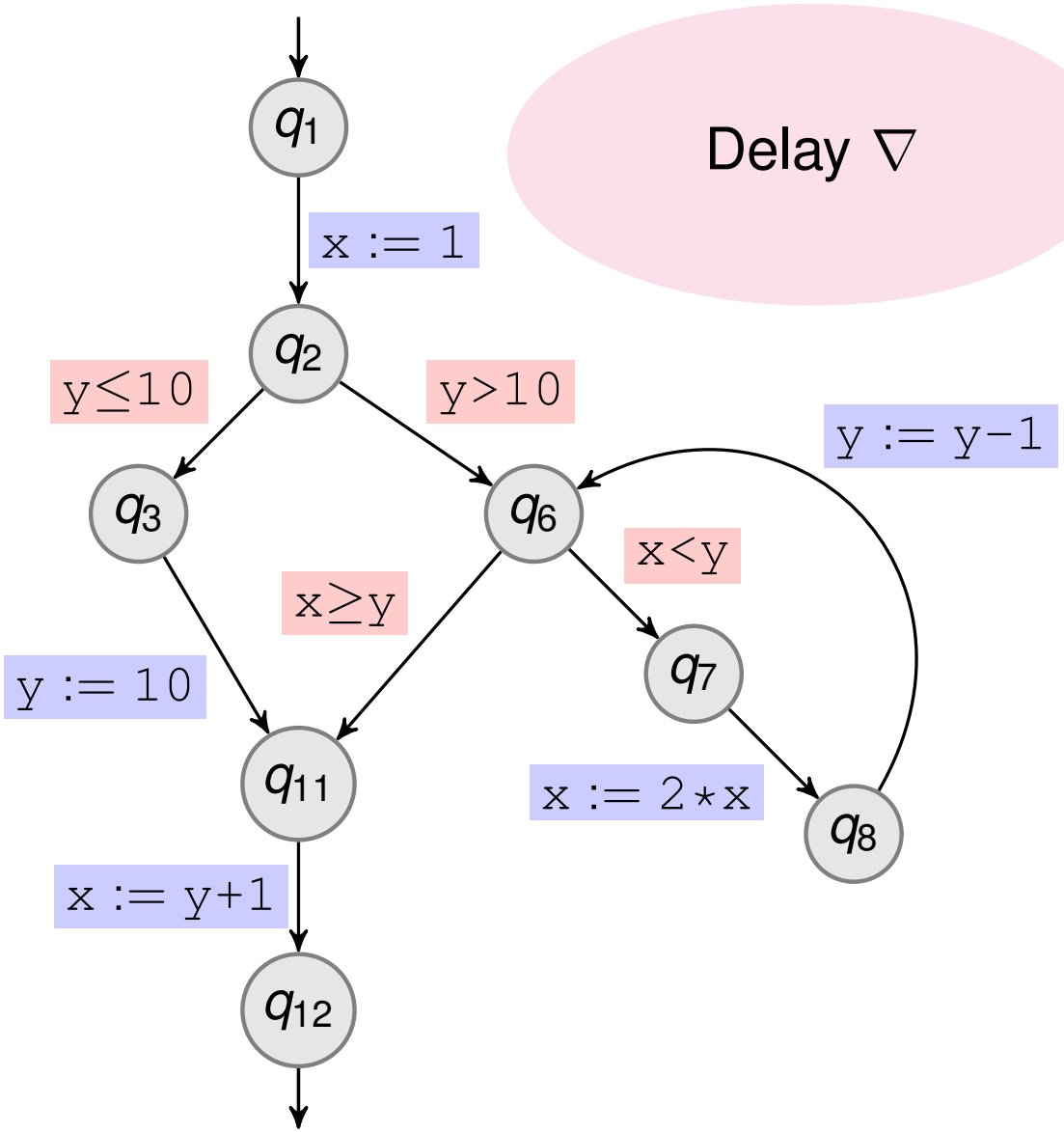
Range Analysis on Example with Delayed Widening



Show that $x > 0$ at q_{12}

	x		y	
q_1	$-\infty$	$+\infty$	$-\infty$	$+\infty$
q_2	1	1	$+\infty$	$+\infty$
q_3	1	1	$-\infty$	10
q_6	1	2^9	1	$+\infty$
q_7	1	2^9	1	$+\infty$
q_8	2	2^{10}	1	$+\infty$
q_{11}	1	1	10	10
q_{12}	11	11	10	10

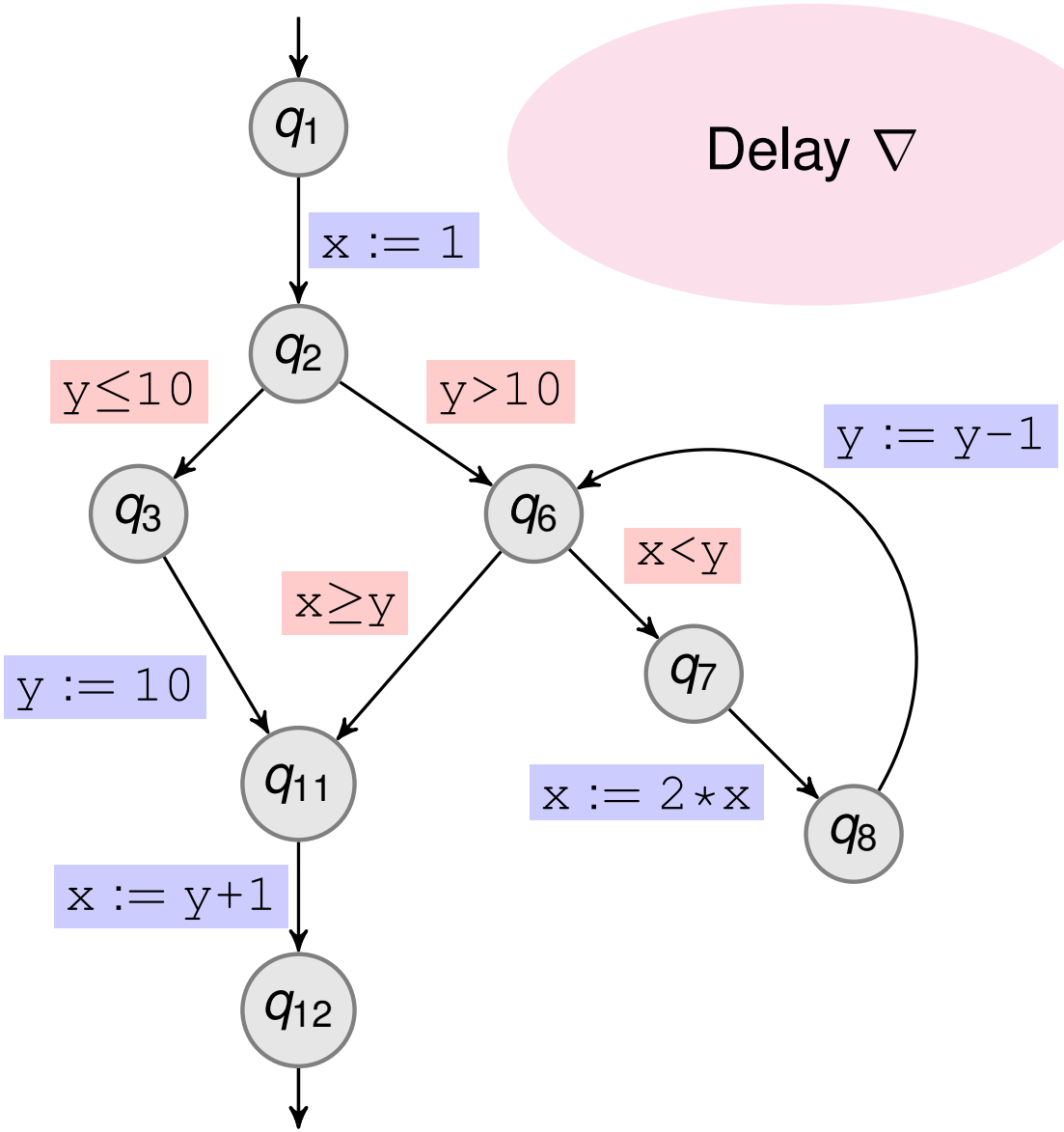
Range Analysis on Example with Delayed Widening



Show that $x > 0$ at q_{12}

	x		y	
q_1	$-\infty$	$+\infty$	$-\infty$	$+\infty$
q_2	1	1	$+\infty$	$+\infty$
q_3	1	1	$-\infty$	10
q_6	1	2^9	1	$+\infty$
q_7	1	2^9	1	$+\infty$
q_8	2	2^{10}	1	$+\infty$
q_{11}	1	1	10	10
q_{12}	11	11	10	10

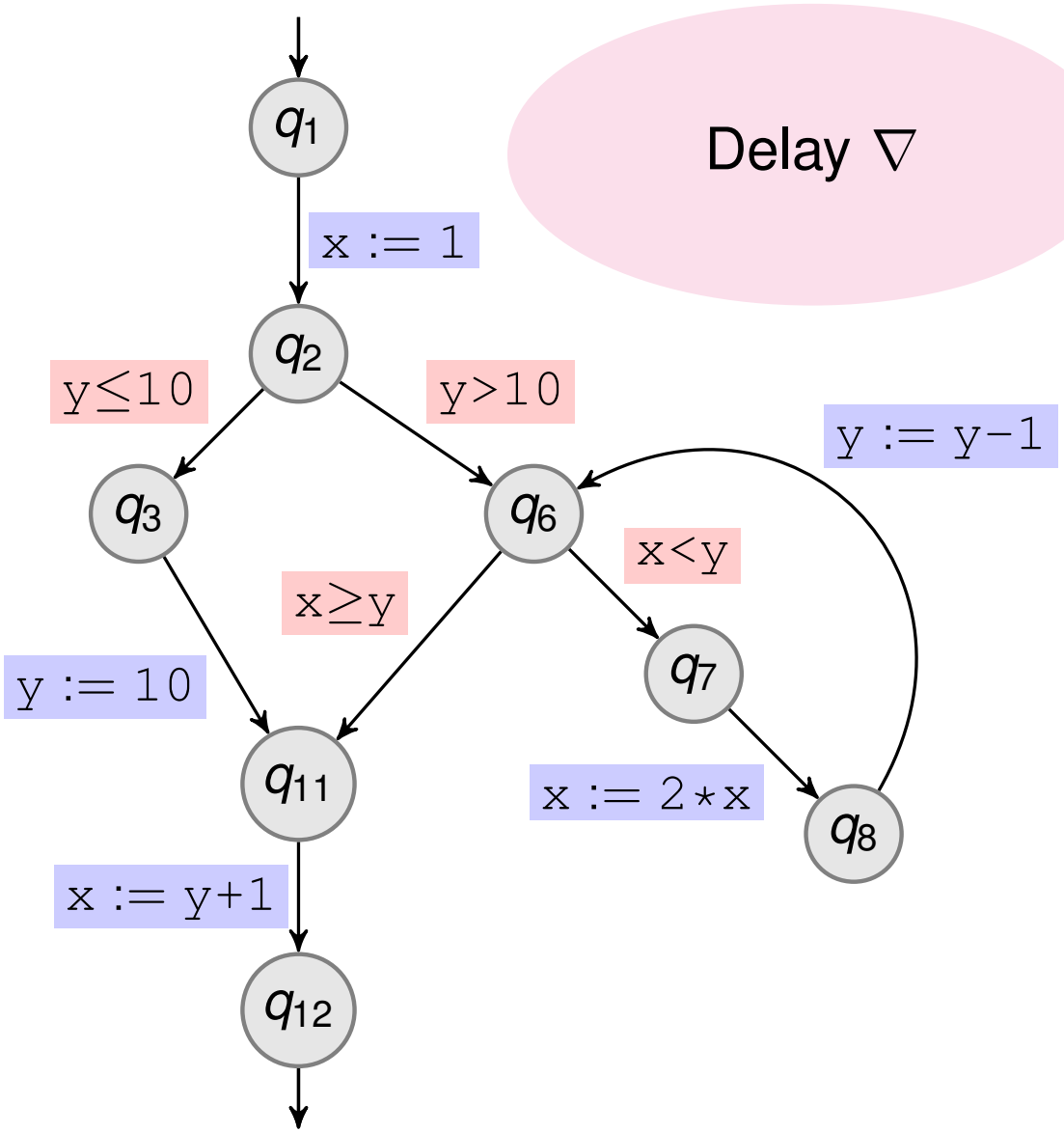
Range Analysis on Example with Delayed Widening



Show that $x > 0$ at q_{12}

	x		y	
q_1	$-\infty$	$+\infty$	$-\infty$	$+\infty$
q_2	1	1	$+\infty$	$+\infty$
q_3	1	1	$-\infty$	10
q_6	1	2^{10}	0	$+\infty$
q_7	1	2^9	1	$+\infty$
q_8	2	2^{10}	1	$+\infty$
q_{11}	1	1	10	10
q_{12}	11	11	10	10

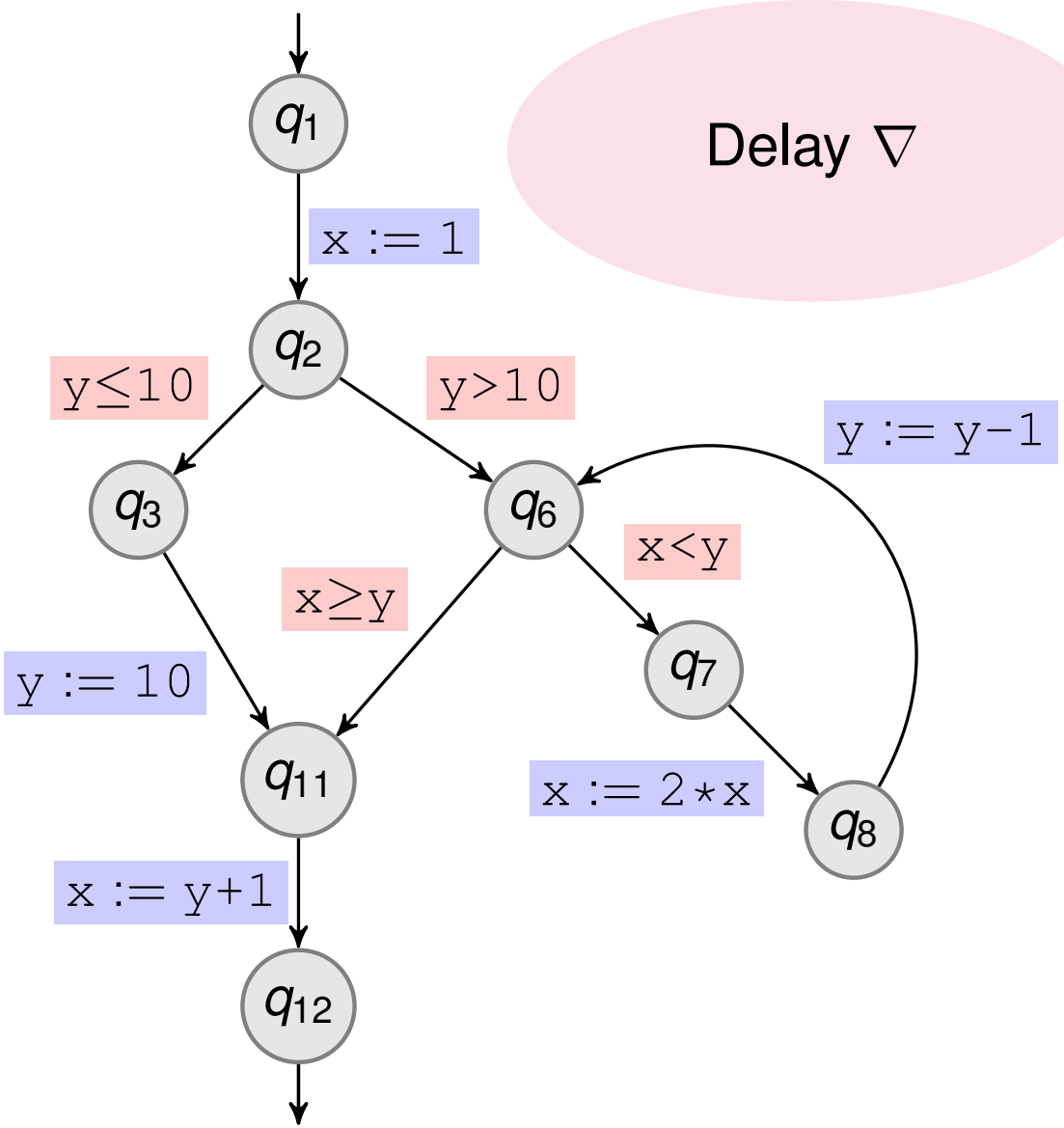
Range Analysis on Example with Delayed Widening



Show that $x > 0$ at q_{12}

	x		y	
q_1	$-\infty$	$+\infty$	$-\infty$	$+\infty$
q_2	1	1	$+\infty$	$+\infty$
q_3	1	1	$-\infty$	10
q_6	1	2^{10}	0	$+\infty$
q_7	1	2^9	1	$+\infty$
q_8	2	2^{10}	1	$+\infty$
q_{11}	1	1	10	10
q_{12}	11	11	10	10

Range Analysis on Example with Delayed Widening

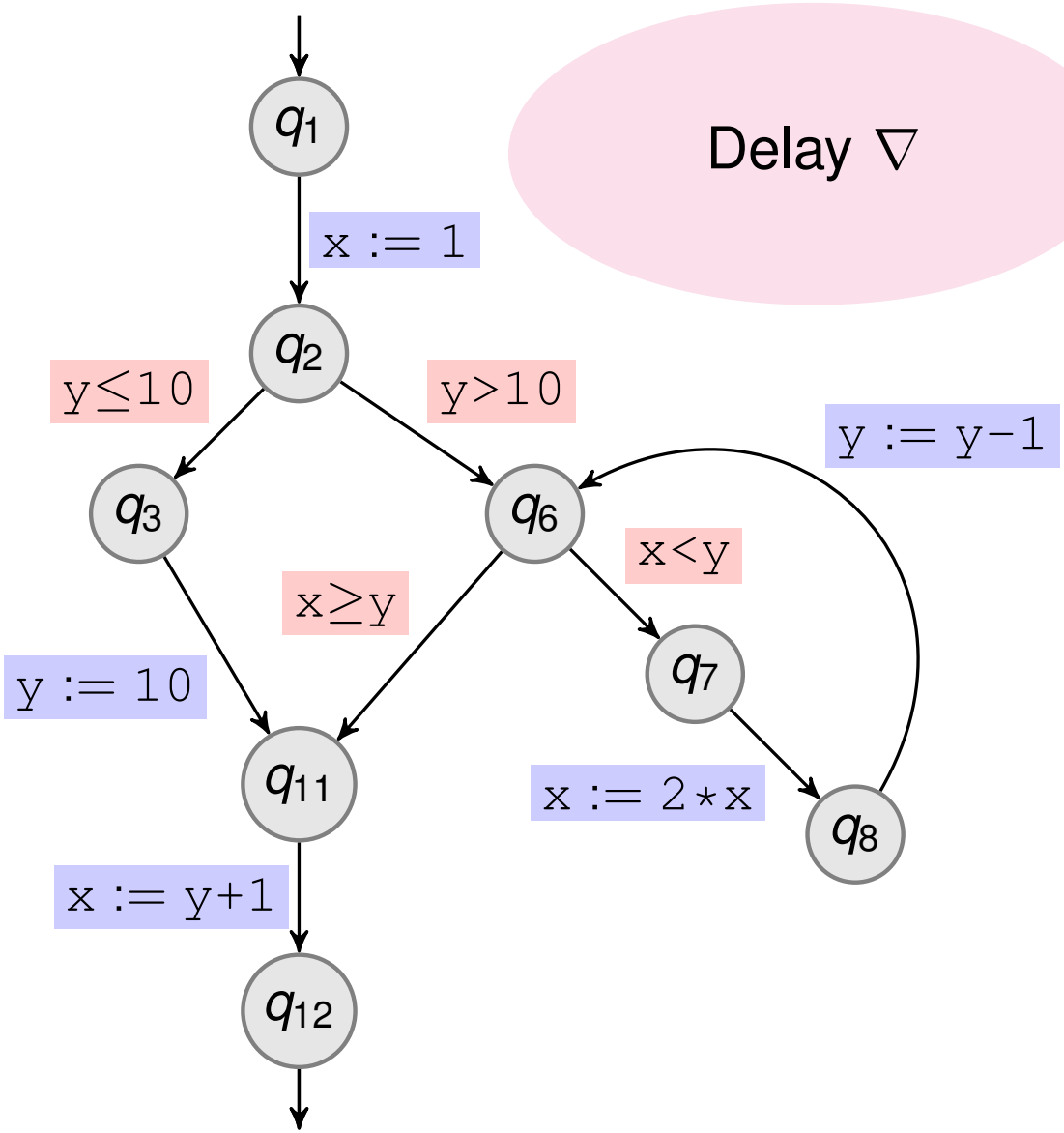


Show that $x > 0$ at q_{12}

	x		y	
q_1	$-\infty$	$+\infty$	$-\infty$	$+\infty$
q_2	1	1	$+\infty$	$+\infty$
q_3	1	1	$-\infty$	10
q_6	1	2^{10}	0	$+\infty$
q_7	1	2^{10}	1	$+\infty$
q_8	2	2^{10}	1	$+\infty$
q_{11}	1	1	10	10
q_{12}	11	11	10	10

Gained from guard $x < y$

Range Analysis on Example with Delayed Widening

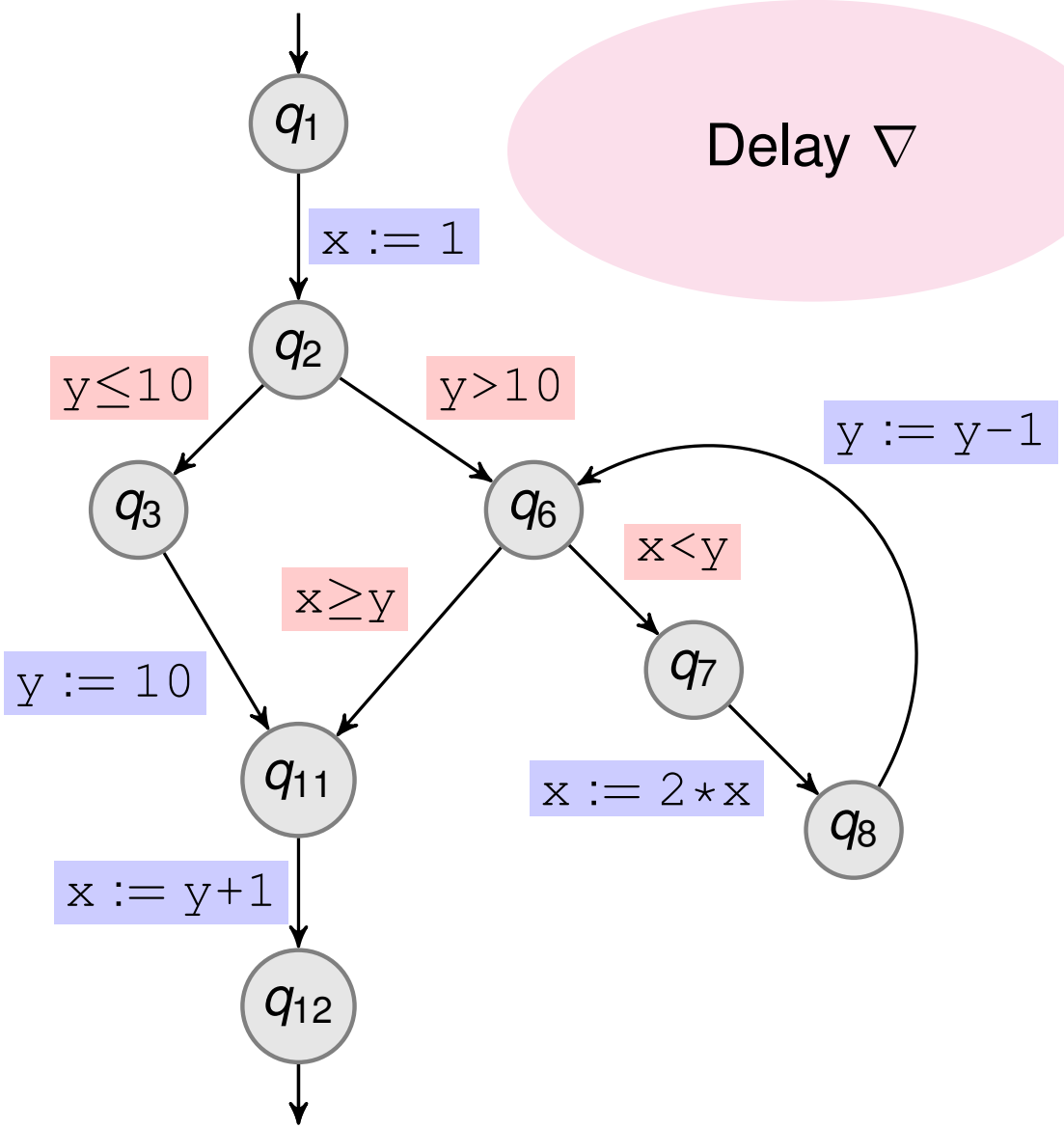


Delay ∇

Show that $x > 0$ at q_{12}

	x		y	
q_1	$-\infty$	$+\infty$	$-\infty$	$+\infty$
q_2	1	1	$+\infty$	$+\infty$
q_3	1	1	$-\infty$	10
q_6	1	2^{10}	0	$+\infty$
q_7	1	2^{10}	1	$+\infty$
q_8	2	2^{10}	1	$+\infty$
q_{11}	1	1	10	10
q_{12}	11	11	10	10

Range Analysis on Example with Delayed Widening

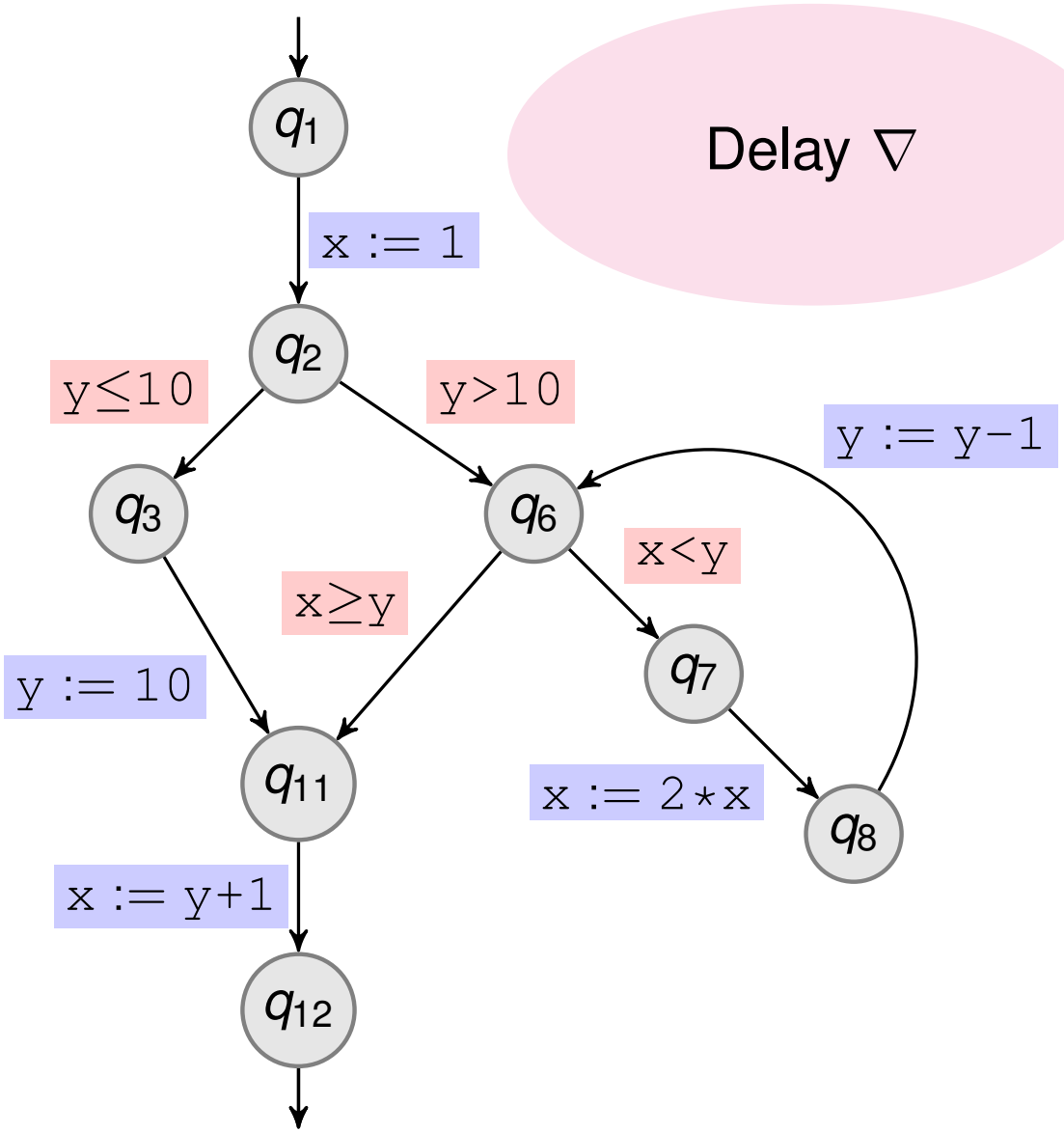


Delay ∇

Show that $x > 0$ at q_{12}

	x		y	
q_1	$-\infty$	$+\infty$	$-\infty$	$+\infty$
q_2	1	1	$+\infty$	$+\infty$
q_3	1	1	$-\infty$	10
q_6	1	2^{10}	0	$+\infty$
q_7	1	2^{10}	1	$+\infty$
q_8	2	2^{11}	1	$+\infty$
q_{11}	1	1	10	10
q_{12}	11	11	10	10

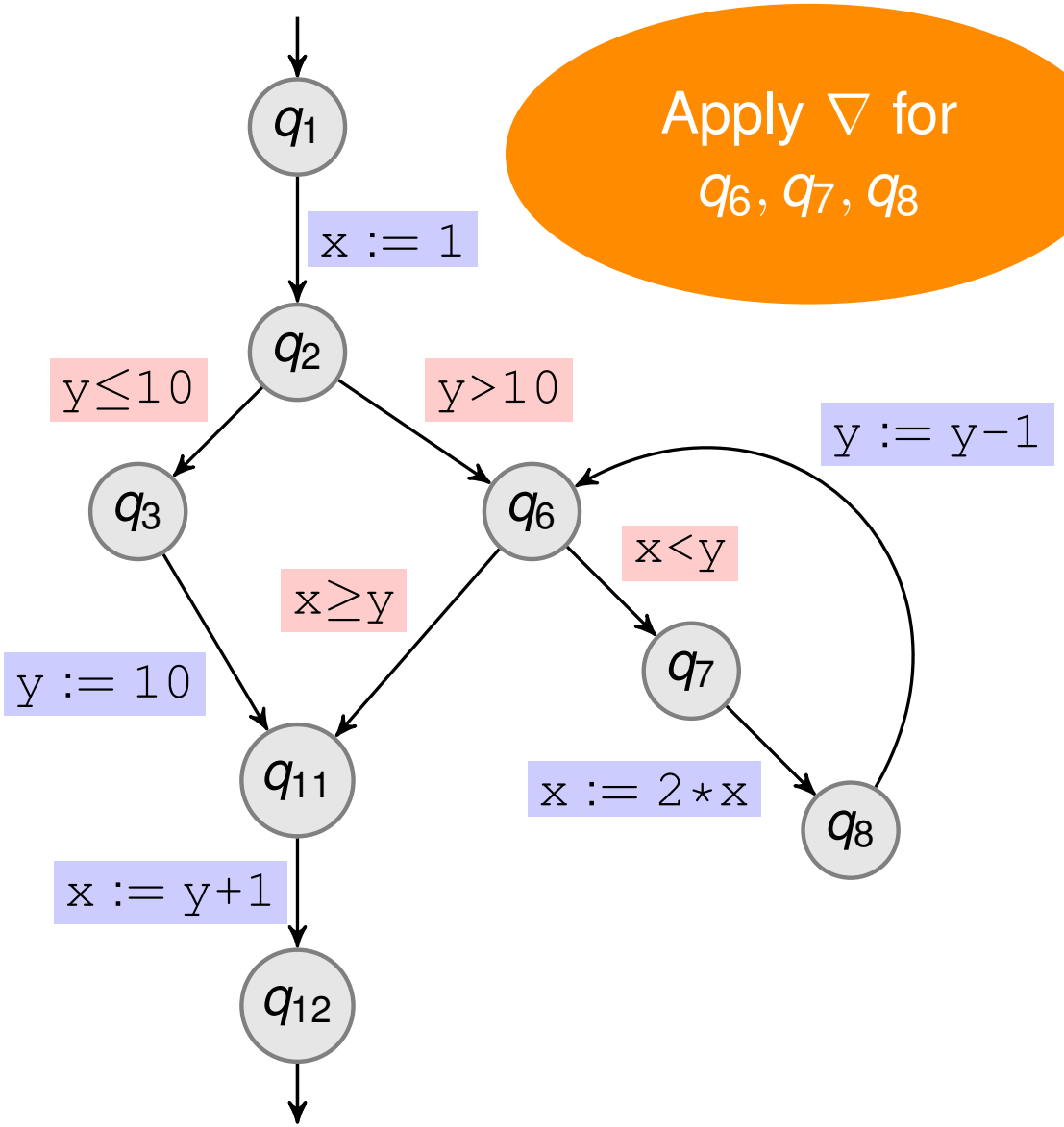
Range Analysis on Example with Delayed Widening



Show that $x > 0$ at q_{12}

	x		y	
q_1	$-\infty$	$+\infty$	$-\infty$	$+\infty$
q_2	1	1	$+\infty$	$+\infty$
q_3	1	1	$-\infty$	10
q_6	1	2^{10}	0	$+\infty$
q_7	1	2^{10}	1	$+\infty$
q_8	2	2^{11}	1	$+\infty$
q_{11}	1	1	10	10
q_{12}	11	11	10	10

Range Analysis on Example with Delayed Widening

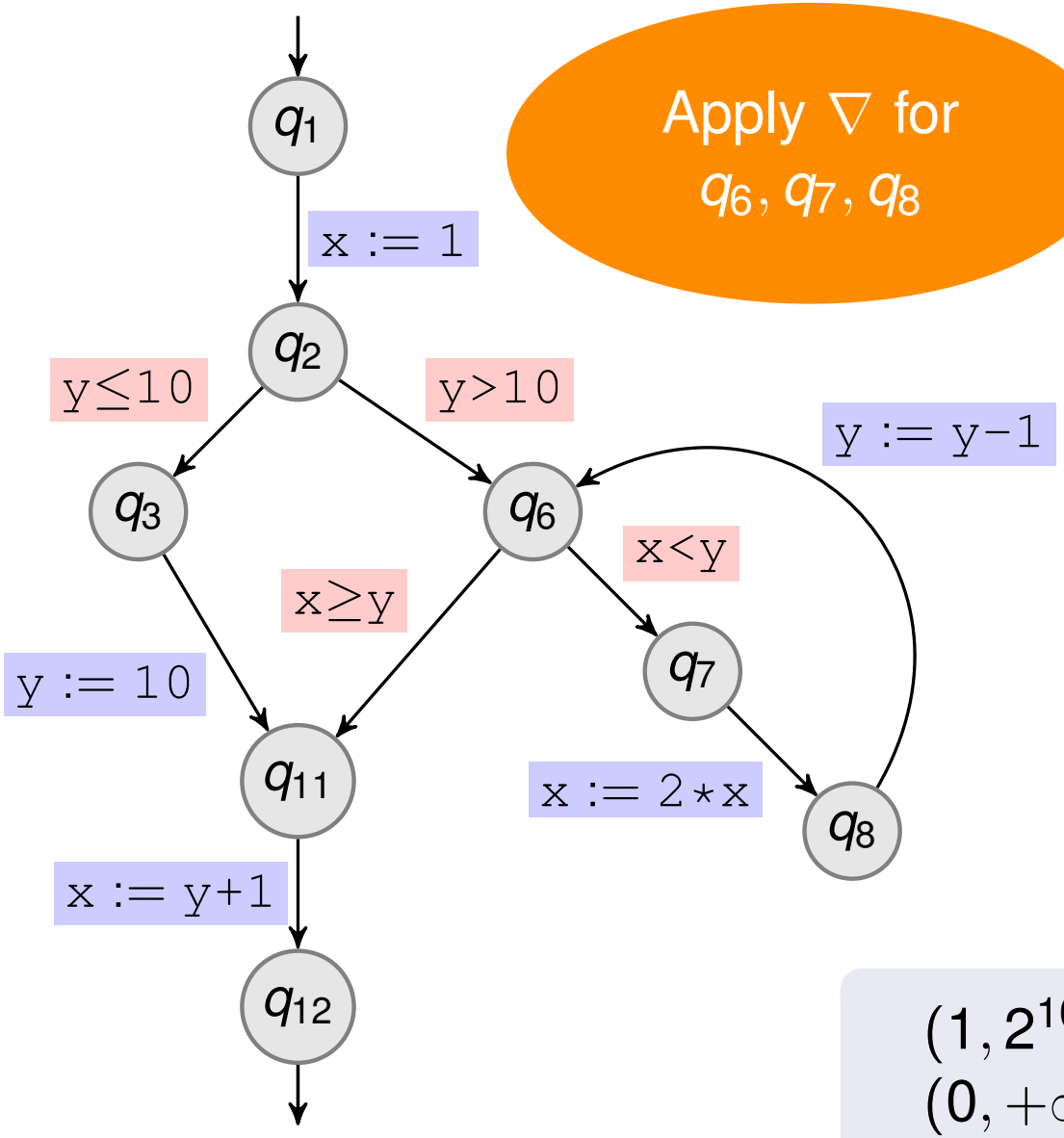


Apply ∇ for q_6, q_7, q_8

Show that $x > 0$ at q_{12}

	x		y	
q_1	$-\infty$	$+\infty$	$-\infty$	$+\infty$
q_2	1	1	$+\infty$	$+\infty$
q_3	1	1	$-\infty$	10
q_6	1	2^{10}	0	$+\infty$
q_7	1	2^{10}	1	$+\infty$
q_8	2	2^{11}	1	$+\infty$
q_{11}	1	1	10	10
q_{12}	11	11	10	10

Range Analysis on Example with Delayed Widening



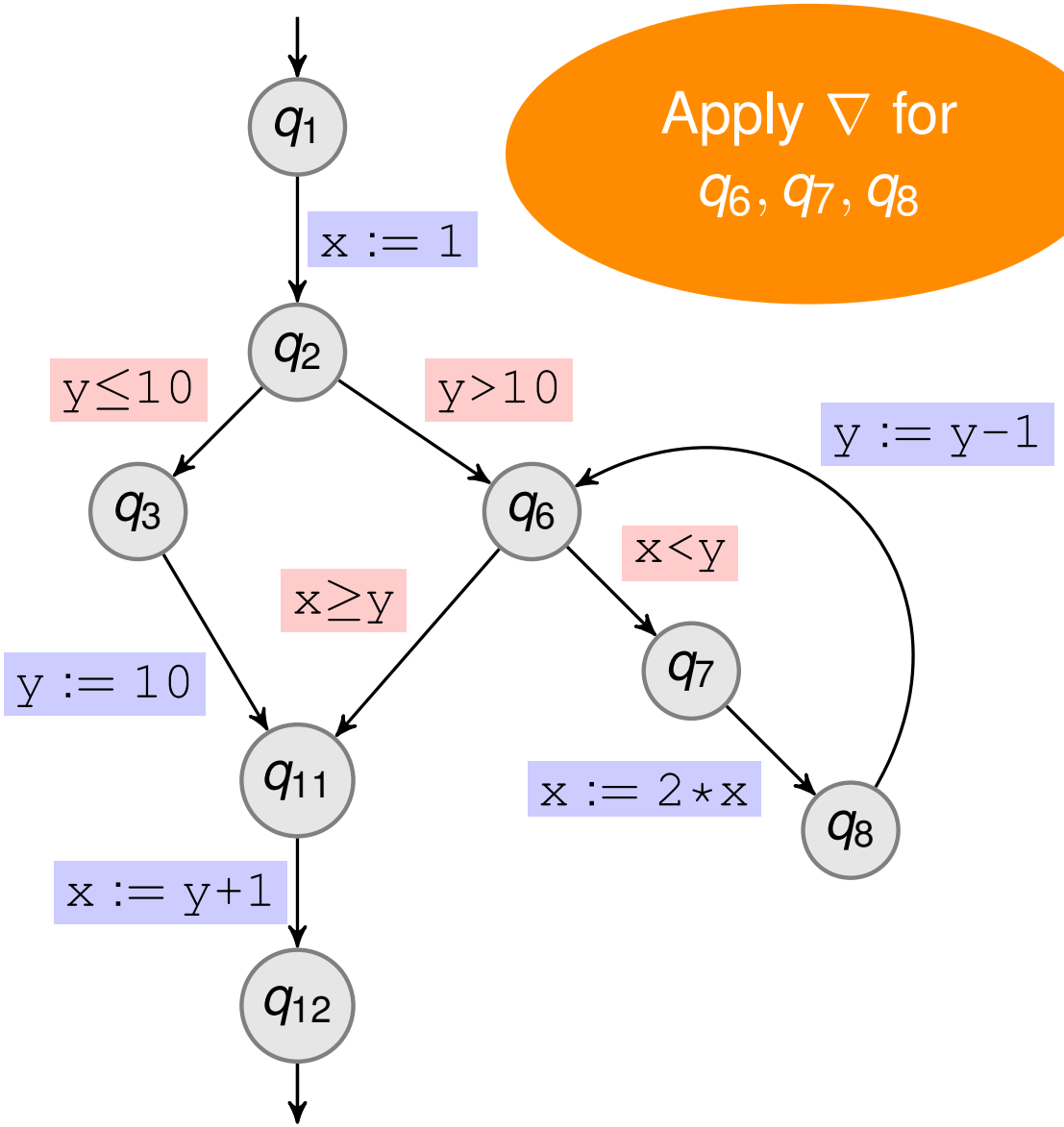
Apply ∇ for q_6, q_7, q_8

Show that $x > 0$ at q_{12}

	x		y	
q_1	$-\infty$	$+\infty$	$-\infty$	$+\infty$
q_2	1	1	$+\infty$	$+\infty$
q_3	1	1	$-\infty$	10
q_6	1	$+\infty$	0	$+\infty$
q_7	1	2^{10}	1	$+\infty$
q_8	2	2^{11}	1	$+\infty$
q_{11}	1	1	10	10
q_{12}	11	11	10	10

$$(1, 2^{10}) \ \nabla \ (2, 2^{11}) = (1, +\infty)$$
$$(0, +\infty) \ \nabla \ (0, +\infty) = (0, +\infty)$$

Range Analysis on Example with Delayed Widening

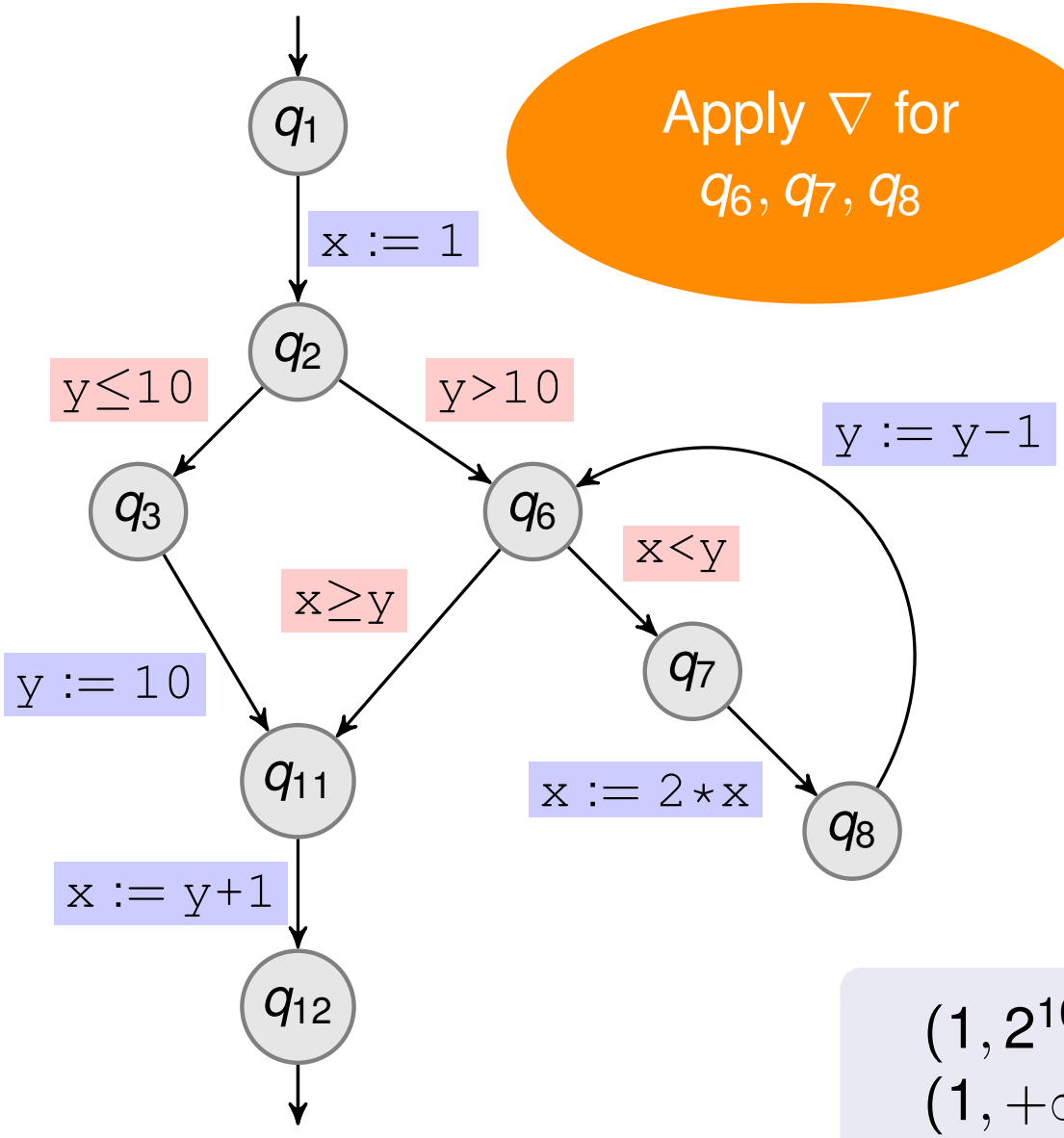


Apply ∇ for q_6, q_7, q_8

Show that $x > 0$ at q_{12}

	x		y	
q_1	$-\infty$	$+\infty$	$-\infty$	$+\infty$
q_2	1	1	$+\infty$	$+\infty$
q_3	1	1	$-\infty$	10
q_6	1	$+\infty$	0	$+\infty$
q_7	1	2^{10}	1	$+\infty$
q_8	2	2^{11}	1	$+\infty$
q_{11}	1	1	10	10
q_{12}	11	11	10	10

Range Analysis on Example with Delayed Widening



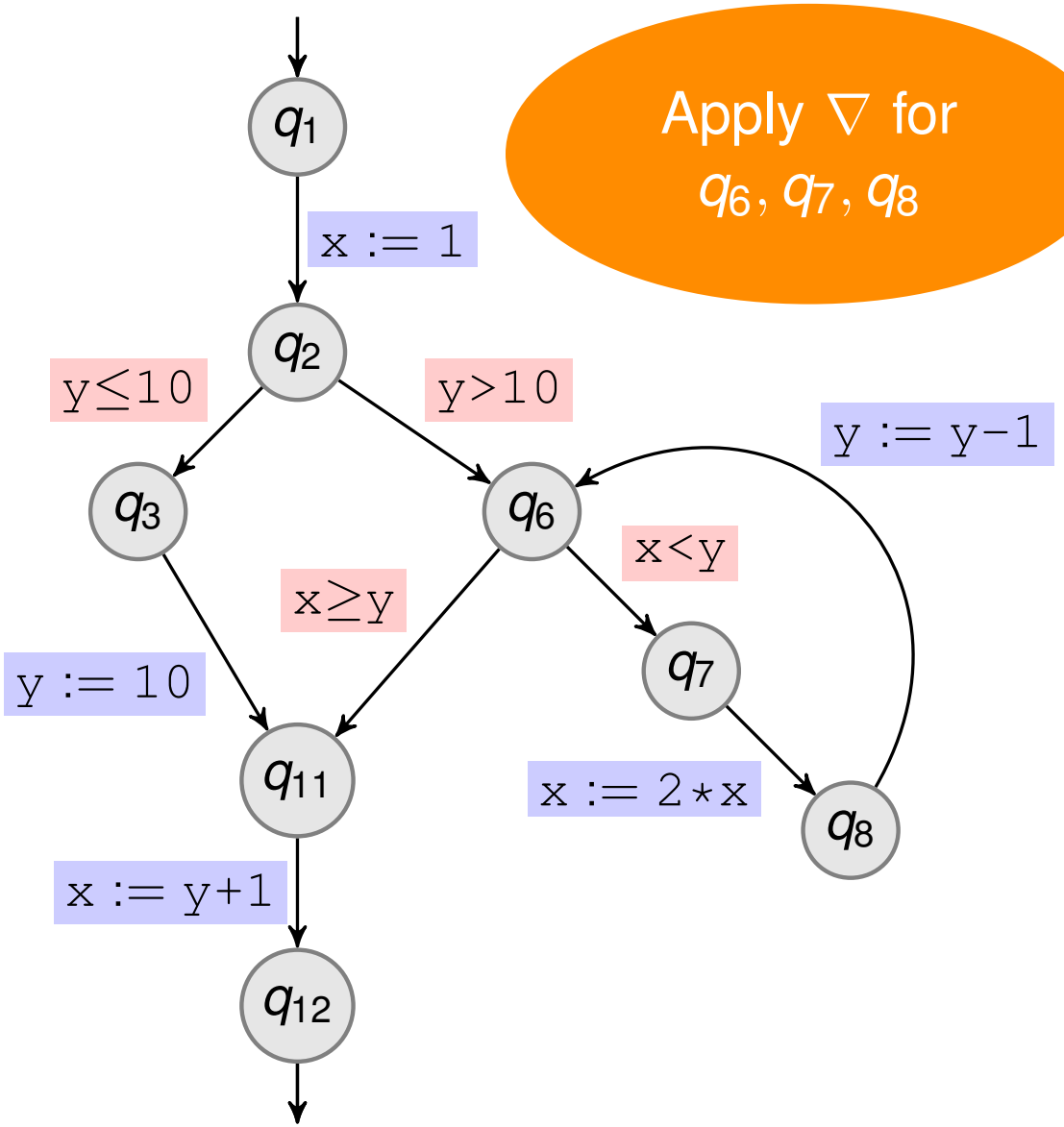
Apply ∇ for q_6, q_7, q_8

Show that $x > 0$ at q_{12}

	x		y	
q_1	$-\infty$	$+\infty$	$-\infty$	$+\infty$
q_2	1	1	$+\infty$	$+\infty$
q_3	1	1	$-\infty$	10
q_6	1	$+\infty$	0	$+\infty$
q_7	1	$+\infty$	1	$+\infty$
q_8	2	2^{11}	1	$+\infty$
q_{11}	1	1	10	10
q_{12}	11	11	10	10

$$(1, 2^{10}) \ \nabla \ (1, +\infty) = (1, +\infty)$$
$$(1, +\infty) \ \nabla \ (1, +\infty) = (1, +\infty)$$

Range Analysis on Example with Delayed Widening

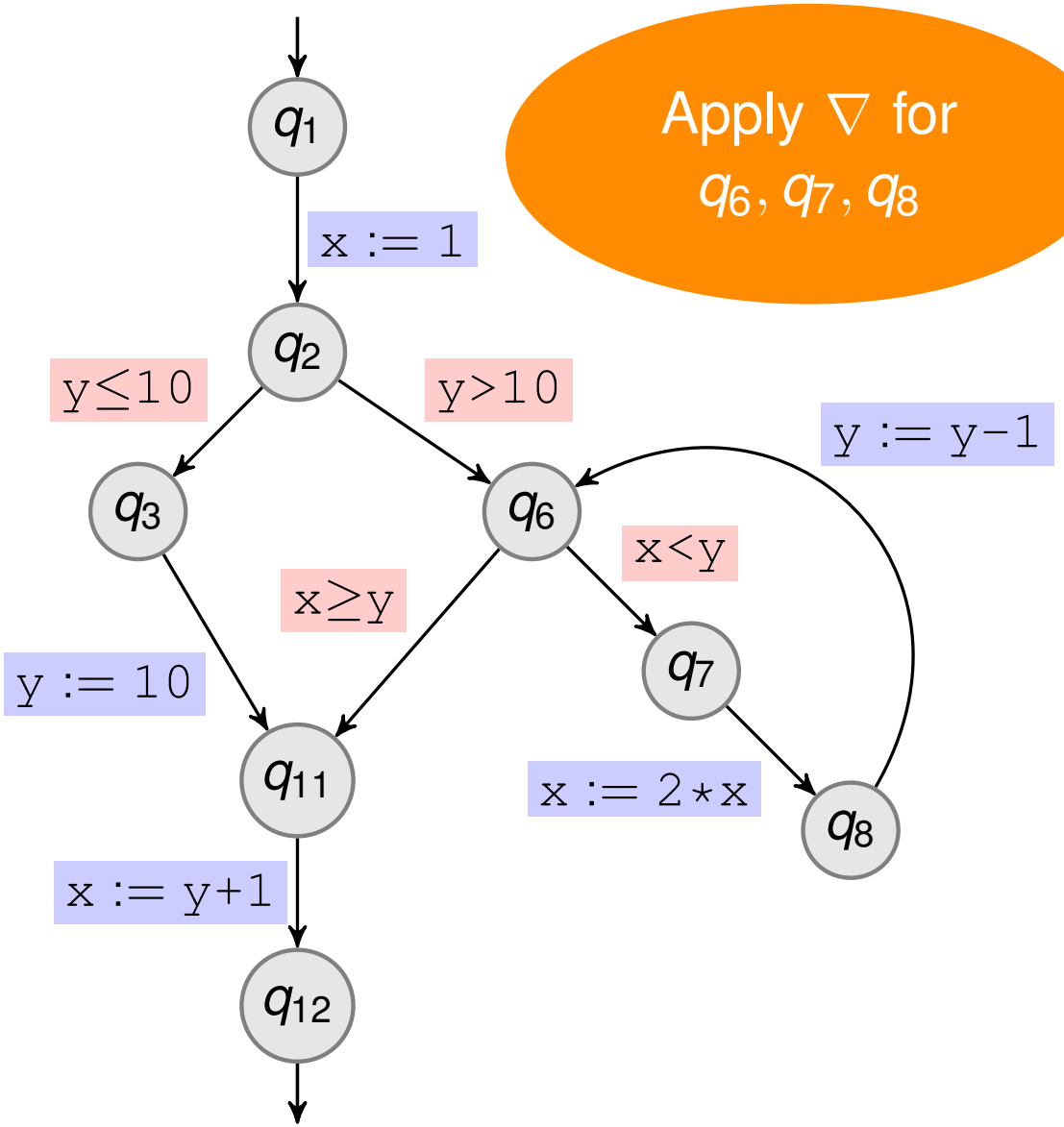


Apply ∇ for q_6, q_7, q_8

Show that $x > 0$ at q_{12}

	x		y	
q_1	$-\infty$	$+\infty$	$-\infty$	$+\infty$
q_2	1	1	$+\infty$	$+\infty$
q_3	1	1	$-\infty$	10
q_6	1	$+\infty$	0	$+\infty$
q_7	1	$+\infty$	1	$+\infty$
q_8	2	2^{11}	1	$+\infty$
q_{11}	1	1	10	10
q_{12}	11	11	10	10

Range Analysis on Example with Delayed Widening

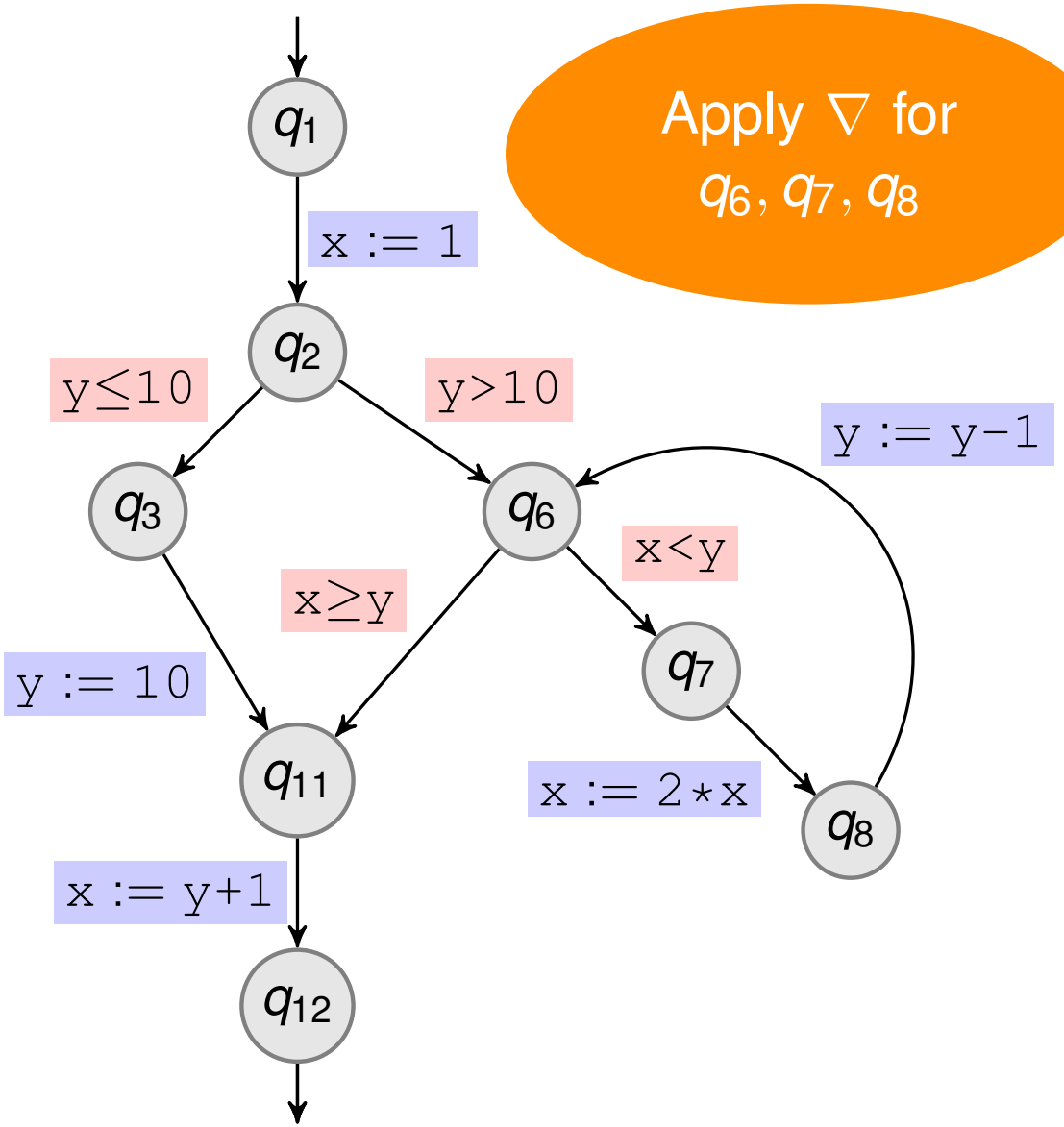


Apply ∇ for q_6, q_7, q_8

Show that $x > 0$ at q_{12}

	x		y	
q_1	$-\infty$	$+\infty$	$-\infty$	$+\infty$
q_2	1	1	$+\infty$	$+\infty$
q_3	1	1	$-\infty$	10
q_6	1	$+\infty$	0	$+\infty$
q_7	1	$+\infty$	1	$+\infty$
q_8	2	$+\infty$	1	$+\infty$
q_{11}	1	1	10	10
q_{12}	11	11	10	10

Range Analysis on Example with Delayed Widening

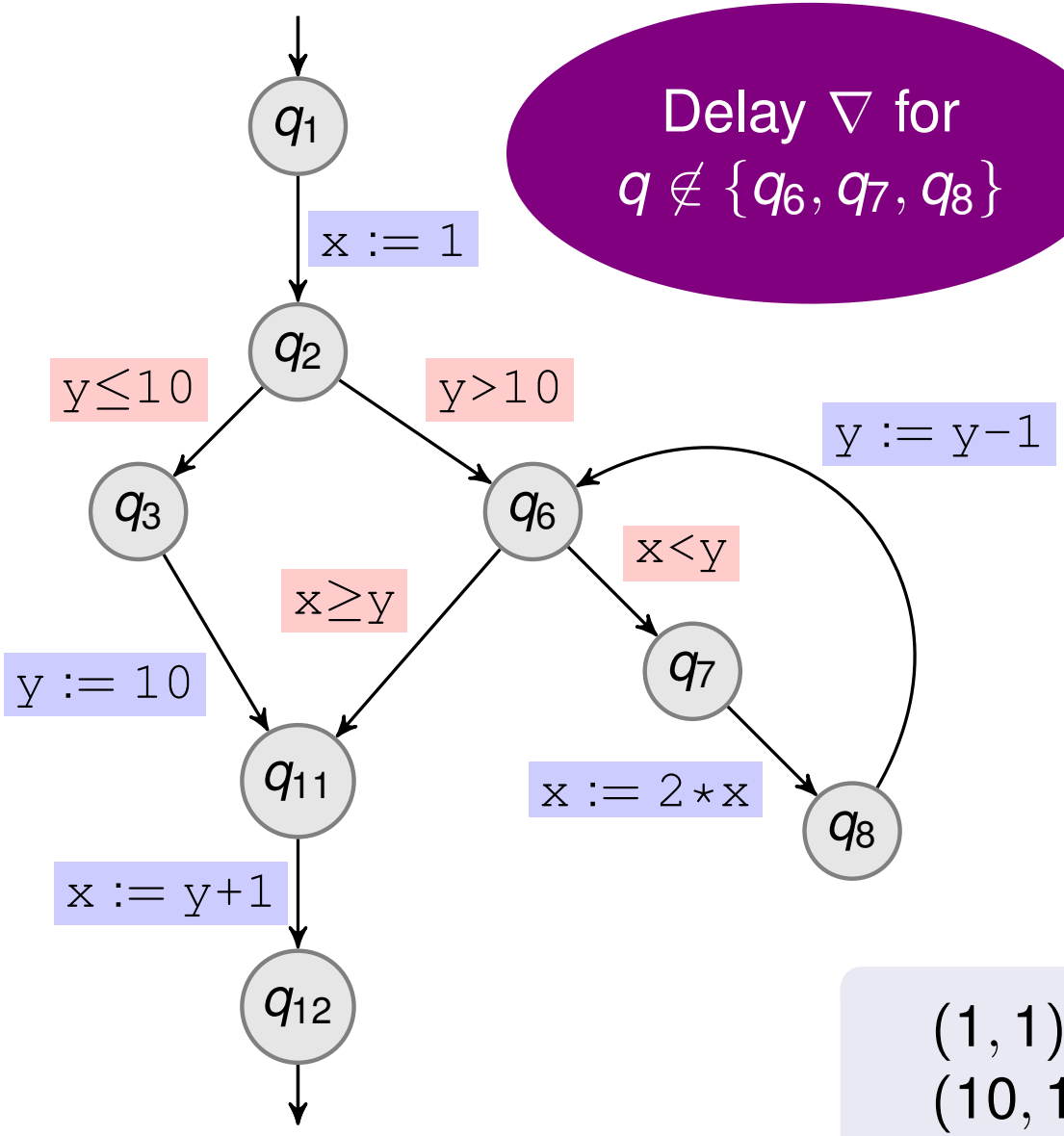


Apply ∇ for q_6, q_7, q_8

Show that $x > 0$ at q_{12}

	x		y	
q_1	$-\infty$	$+\infty$	$-\infty$	$+\infty$
q_2	1	1	$+\infty$	$+\infty$
q_3	1	1	$-\infty$	10
q_6	1	$+\infty$	0	$+\infty$
q_7	1	$+\infty$	1	$+\infty$
q_8	2	$+\infty$	1	$+\infty$
q_{11}	1	1	10	10
q_{12}	11	11	10	10

Range Analysis on Example with Delayed Widening



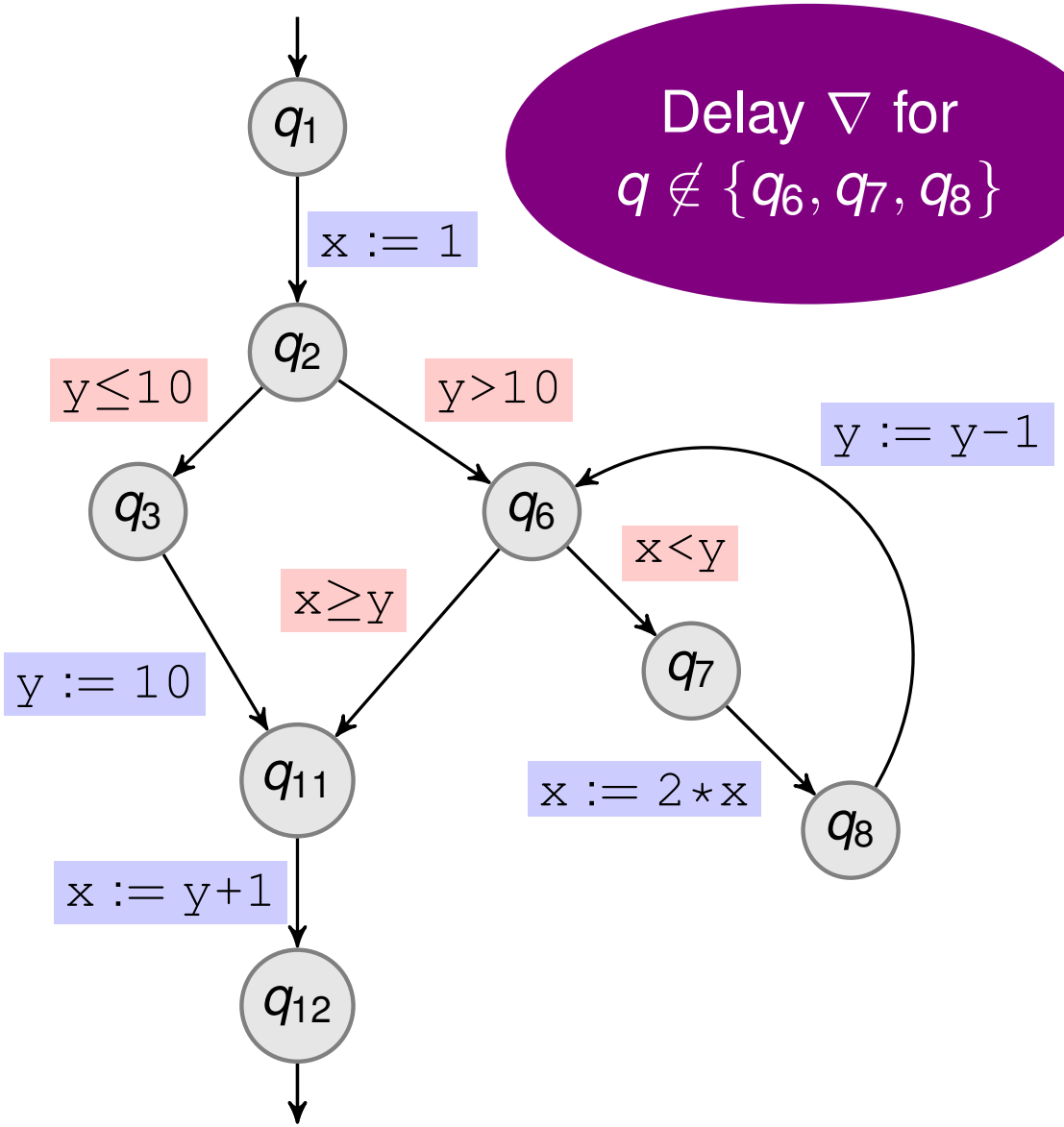
Delay ∇ for $q \notin \{q_6, q_7, q_8\}$

Show that $x > 0$ at q_{12}

	x		y	
q_1	$-\infty$	$+\infty$	$-\infty$	$+\infty$
q_2	1	1	$+\infty$	$+\infty$
q_3	1	1	$-\infty$	10
q_6	1	$+\infty$	0	$+\infty$
q_7	1	$+\infty$	1	$+\infty$
q_8	2	$+\infty$	1	$+\infty$
q_{11}	1	$+\infty$	0	$+\infty$
q_{12}	11	11	10	10

$$(1,1) \sqcup (1,+\infty) = (1,+\infty)$$
$$(10,10) \sqcup (0,+\infty) = (0,+\infty)$$

Range Analysis on Example with Delayed Widening

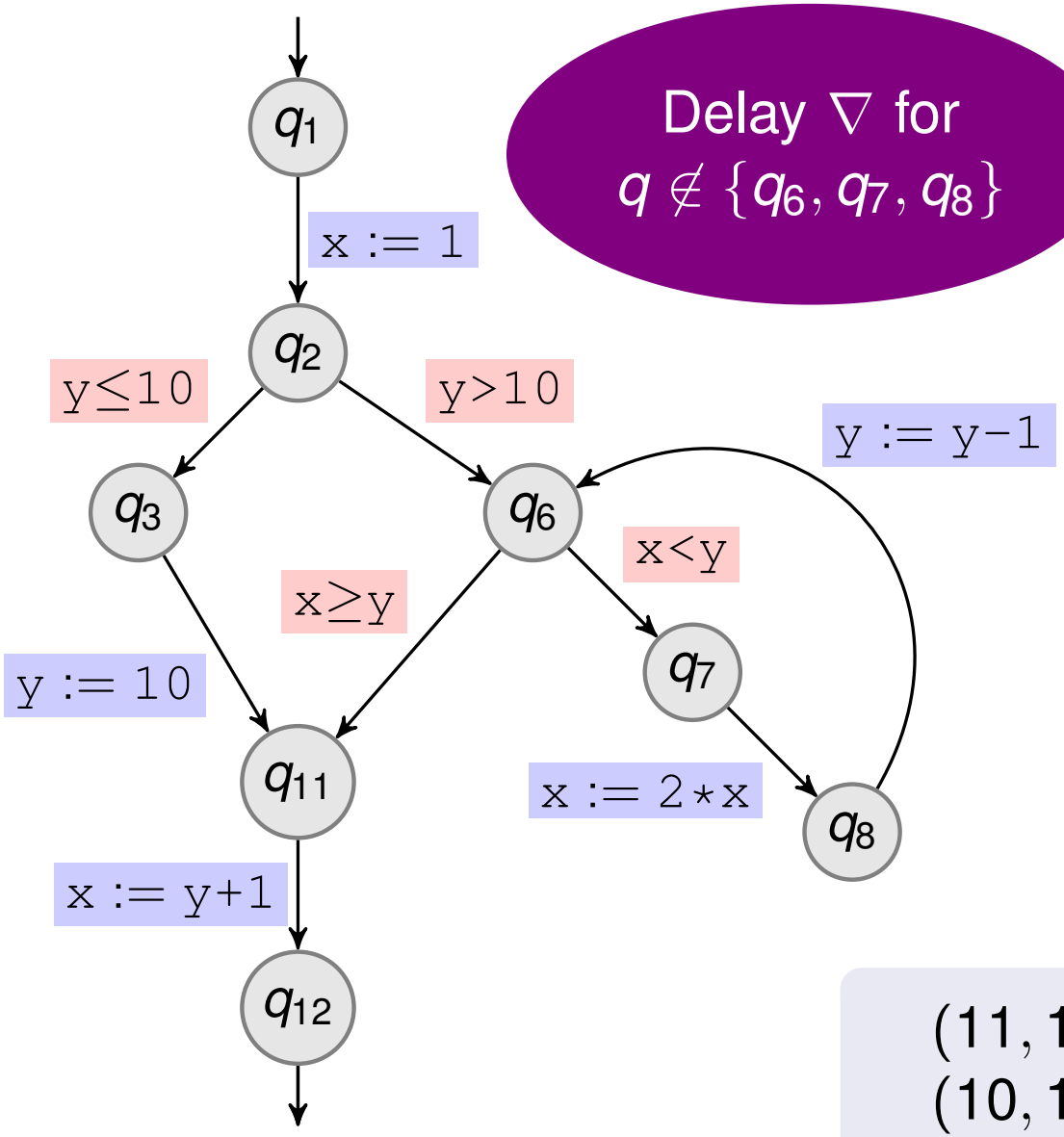


Delay ∇ for $q \notin \{q_6, q_7, q_8\}$

Show that $x > 0$ at q_{12}

	x		y	
q_1	$-\infty$	$+\infty$	$-\infty$	$+\infty$
q_2	1	1	$+\infty$	$+\infty$
q_3	1	1	$-\infty$	10
q_6	1	$+\infty$	0	$+\infty$
q_7	1	$+\infty$	1	$+\infty$
q_8	2	$+\infty$	1	$+\infty$
q_{11}	1	$+\infty$	0	$+\infty$
q_{12}	11	11	10	10

Range Analysis on Example with Delayed Widening



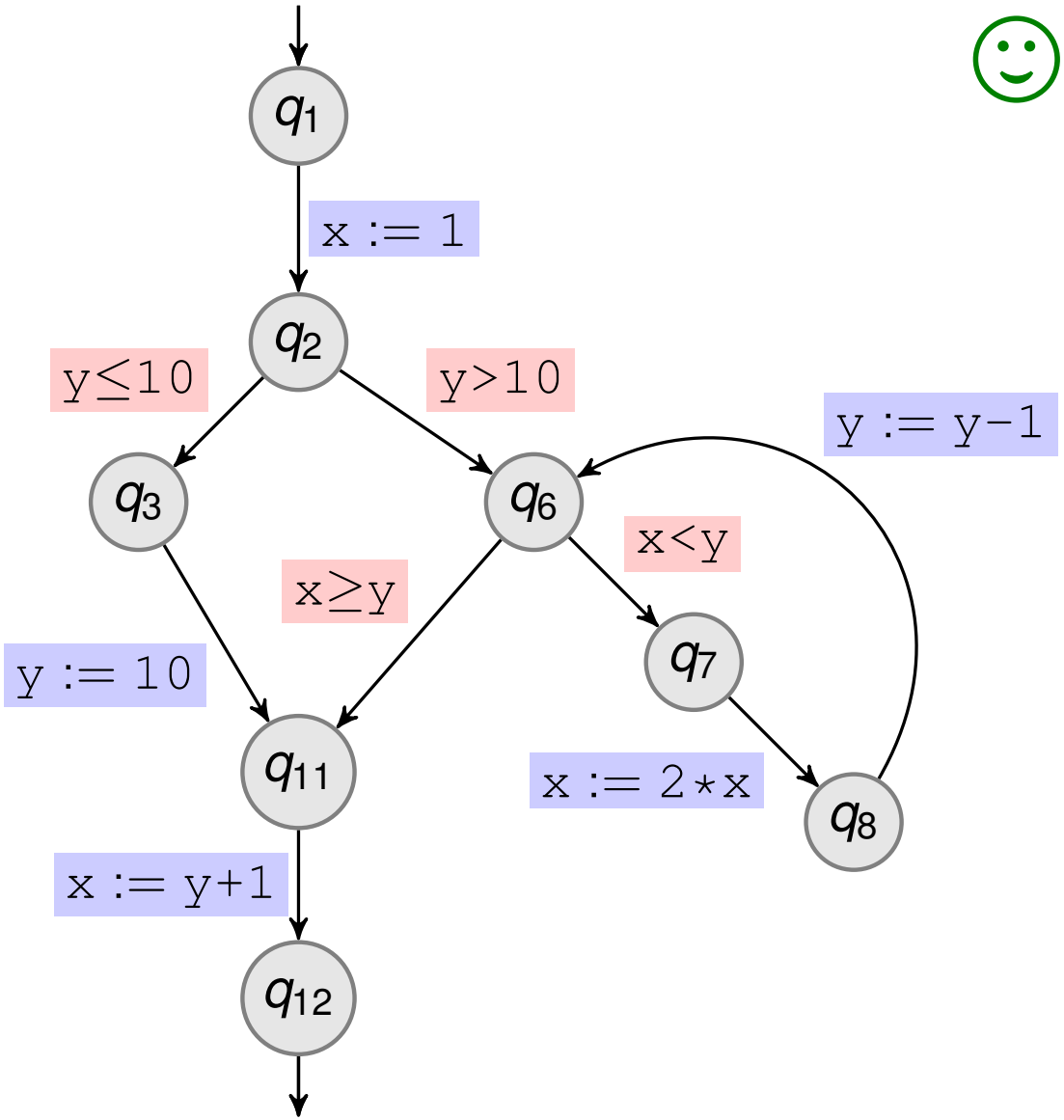
Delay ∇ for $q \notin \{q_6, q_7, q_8\}$

Show that $x > 0$ at q_{12}

	x		y	
q_1	$-\infty$	$+\infty$	$-\infty$	$+\infty$
q_2	1	1	$+\infty$	$+\infty$
q_3	1	1	$-\infty$	10
q_6	1	$+\infty$	0	$+\infty$
q_7	1	$+\infty$	1	$+\infty$
q_8	2	$+\infty$	1	$+\infty$
q_{11}	1	$+\infty$	0	$+\infty$
q_{12}	0	$+\infty$	0	$+\infty$

$$(11, 11) \sqcup (1, +\infty) = (1, +\infty)$$
$$(10, 10) \sqcup (0, +\infty) = (0, +\infty)$$

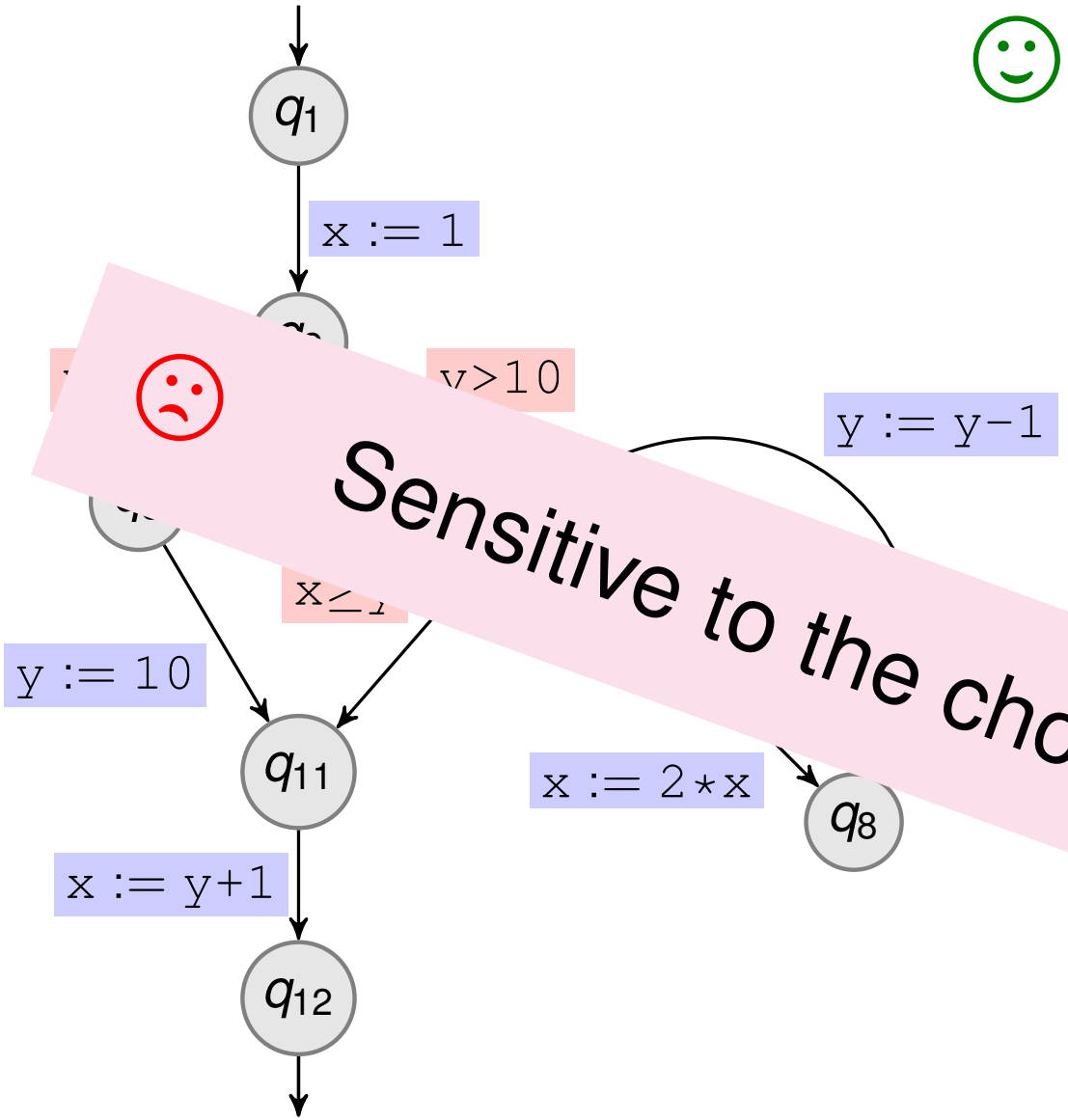
Range Analysis on Example with Delayed Widening



Show that $x > 0$ at q_{12}

	x		y	
q_1	$-\infty$	$+\infty$	$-\infty$	$+\infty$
q_2	1	1	$+\infty$	$+\infty$
q_3	1	1	$-\infty$	10
q_6	1	$+\infty$	0	$+\infty$
q_7	1	$+\infty$	1	$+\infty$
q_8	2	$+\infty$	1	$+\infty$
q_{11}	1	$+\infty$	0	$+\infty$
q_{12}	0	$+\infty$	0	$+\infty$

Range Analysis on Example with Delayed Widening



Show that $x > 0$ at q_{12}

	x		y	
q_1	$-\infty$	$+\infty$	$-\infty$	$+\infty$
q_2	1	1	$+\infty$	$+\infty$
q_3	1	1	$-\infty$	10
q_6	1	$+\infty$	0	$+\infty$
q_7	1	$+\infty$	1	$+\infty$
	2	$+\infty$	1	$+\infty$
		$+\infty$	0	$+\infty$
				$+\infty$

Sensitive to the choice of delay

Precision Improvement with Narrowing

Consider a complete lattice $(L, \sqsubseteq)$.

Objective of Narrowing Operators

Soundly improve the precision of an approximation obtained with ∇

Definition

A narrowing operator for $(L, \sqsubseteq)$ is a function $\Delta : (L \times L) \to L$ such that:

1. $$y \sqsubseteq x \implies y \sqsubseteq (x \,\Delta\, y) \sqsubseteq x \qquad \text{(for all } x, y \in L\text{)}$$
2. for any descending chain $x_0 \sqsupseteq x_1 \sqsupseteq \cdots$ of elements of L, the descending chain $y_0 \sqsupseteq y_1 \sqsupseteq \cdots$ defined by

$$\begin{cases} y_0 = x_0 \\ y_{i+1} = y_i \,\Delta\, x_{i+1} & \text{for all } i \in \mathbb{N} \end{cases}$$

is not strictly decreasing (i.e. $y_{i+1} = y_i$ for some $i \in \mathbb{N}$).

Correctness of Decreasing Iteration with Narrowing

Consider a complete lattice $(L, \sqsubseteq)$ and a monotonic function $f : L \to L$.

A post-fixpoint of f is any element $a \in L$ satisfying $a \sqsupseteq f(a)$.

Theorem

If $\Delta : (L \times L) \to L$ is a narrowing operator then for any post-fixpoint a of f, the descending chain $x_0 \sqsupseteq x_1 \sqsupseteq \cdots$ defined by

$$\begin{aligned} x_0 &= a \\ x_{i+1} &= x_i \,\Delta\, f(x_{i+1}) \end{aligned}$$

is eventually stationary, and its limit satisfies $\bigsqcap \{x_i \mid i \in \mathbb{N}\} \sqsupseteq \mathrm{lfp}(f)$.

Application to Precision Improvement of MFP Approximations

1. Compute an approximation of *MFP* by Kleene iteration with ∇.
2. Then perform a decreasing iteration with Δ to regain precision.

Correctness of Decreasing Iteration with Narrowing

Consider a complete lattice $(L, \sqsubseteq)$ and a monotonic function $f : L \to L$.

A post-fixpoint of f is any element $a \in L$ satisfying $a \sqsupseteq f(a)$.

Theorem

If $\Delta : (L \times L) \to L$ is a narrowing operator then for any post-fixpoint a of f, the descending chain $x_0 \sqsupseteq x_1 \sqsupseteq \cdots$ defined by

$$\begin{aligned} x_0 &= a \\ x_{i+1} &= x_i \,\Delta\, f(x_{i+1}) \end{aligned}$$

is eventually stationary, and its limit satisfies $\bigsqcap \{x_i \mid i \in \mathbb{N}\} \sqsupseteq \mathrm{lfp}(f)$.

Application to Precision Improvement of MFP Approximations

1. Compute an approximation of *MFP* by Kleene iteration with ∇.
2. Then perform a decreasing iteration with Δ to regain precision.

Narrowing Operator for Range Analysis: Intuition

Objective of Narrowing Operators

Soundly improve the precision of an approximation obtained with ∇

∇ may have introduced infinite bounds to accelerate convergence.

Improve infinite bounds when possible (leave the non-infinite ones)

Examples

$$\begin{aligned} (1, +\infty) \,\Delta\, (1, 4) &= (1, 4) \\ (1, 10) \,\Delta\, (1, 4) &= (1, 10) \\ (-\infty, 10) \,\Delta\, (1, 4) &= (1, 10) \end{aligned}$$

Narrowing Operator for Range Analysis

Narrowing Operator on the Complete Lattice $(Int, \sqsubseteq)$ of Intervals

$$\bot \Delta \bot = \bot \qquad \bot \Delta (l, u) = (l, u) \Delta \bot = \bot$$

$$(l_1, u_1) \Delta (l_2, u_2) = (l_\Delta, u_\Delta) \quad \text{where} \quad \begin{cases} l_\Delta = \begin{cases} l_2 & \text{if } l_1 = -\infty \\ l_1 & \text{otherwise} \end{cases} \\ u_\Delta = \begin{cases} u_2 & \text{if } u_1 = +\infty \\ u_1 & \text{otherwise} \end{cases} \end{cases}$$

Narrowing Operator on the Complete Lattice $(\mathbb{X} \to Int, \dot{\sqsubseteq})$

Extension $\dot{\Delta}$ of the narrowing Δ on $(Int, \sqsubseteq)$ to $(\mathbb{X} \to Int, \dot{\sqsubseteq})$, defined by:

$$\overline{v_1} \; \dot{\Delta} \; \overline{v_2} = \lambda q \,.\, \overline{v_1}(q) \Delta \overline{v_2}(q)$$

Narrowing Operator for Range Analysis

Narrowing Operator on the Complete Lattice $(Int, \sqsubseteq)$ of Intervals

$$\bot \,\Delta\, \bot \;=\; \bot \qquad\qquad \bot \,\Delta\, (l,u) \;=\; (l,u) \,\Delta\, \bot \;=\; \bot$$

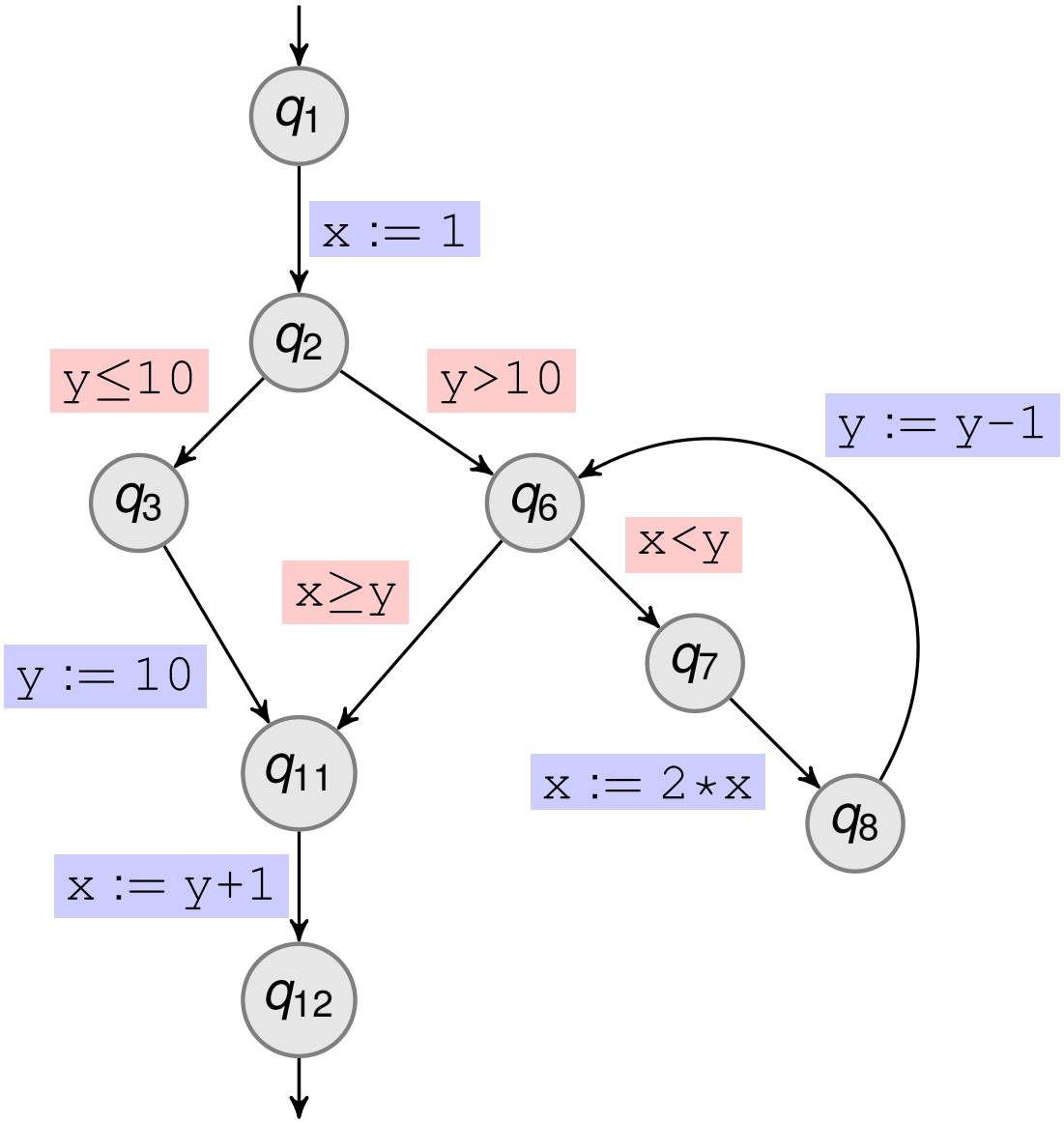
$$(l_1,u_1) \,\Delta\, (l_2,u_2) \;=\; (l_\Delta, u_\Delta) \quad \text{where} \quad \begin{cases} l_\Delta \;=\; \begin{cases} l_2 & \text{if } l_1 = -\infty \\ l_1 & \text{otherwise} \end{cases} \\ u_\Delta \;=\; \begin{cases} u_2 & \text{if } u_1 = +\infty \\ u_1 & \text{otherwise} \end{cases} \end{cases}$$

Narrowing Operator on the Complete Lattice $(\mathbb{X} \to Int, \overline{\sqsubseteq})$

Extension Δ of the narrowing Δ on $(Int, \sqsubseteq)$ to $(\mathbb{X} \to Int, \overline{\sqsubseteq})$, defined by:

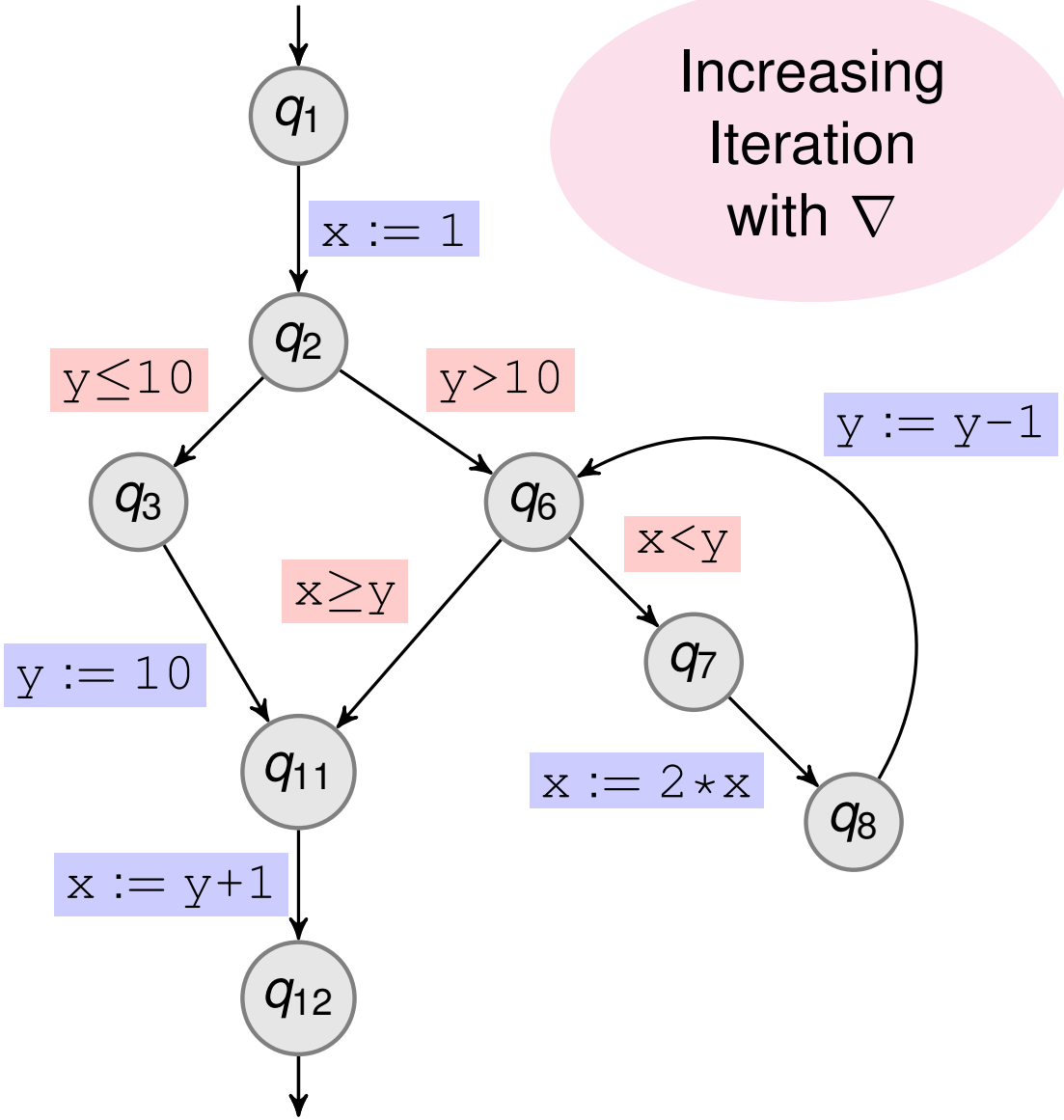
$$\overline{v_1} \;\Delta\; \overline{v_2} \;=\; \lambda\, q \,.\, \overline{v_1}(q) \,\Delta\, \overline{v_2}(q)$$

Range Analysis on Example with ∇ and Δ



Show that $x > 0$ at q_{12}

Range Analysis on Example with ∇ and Δ

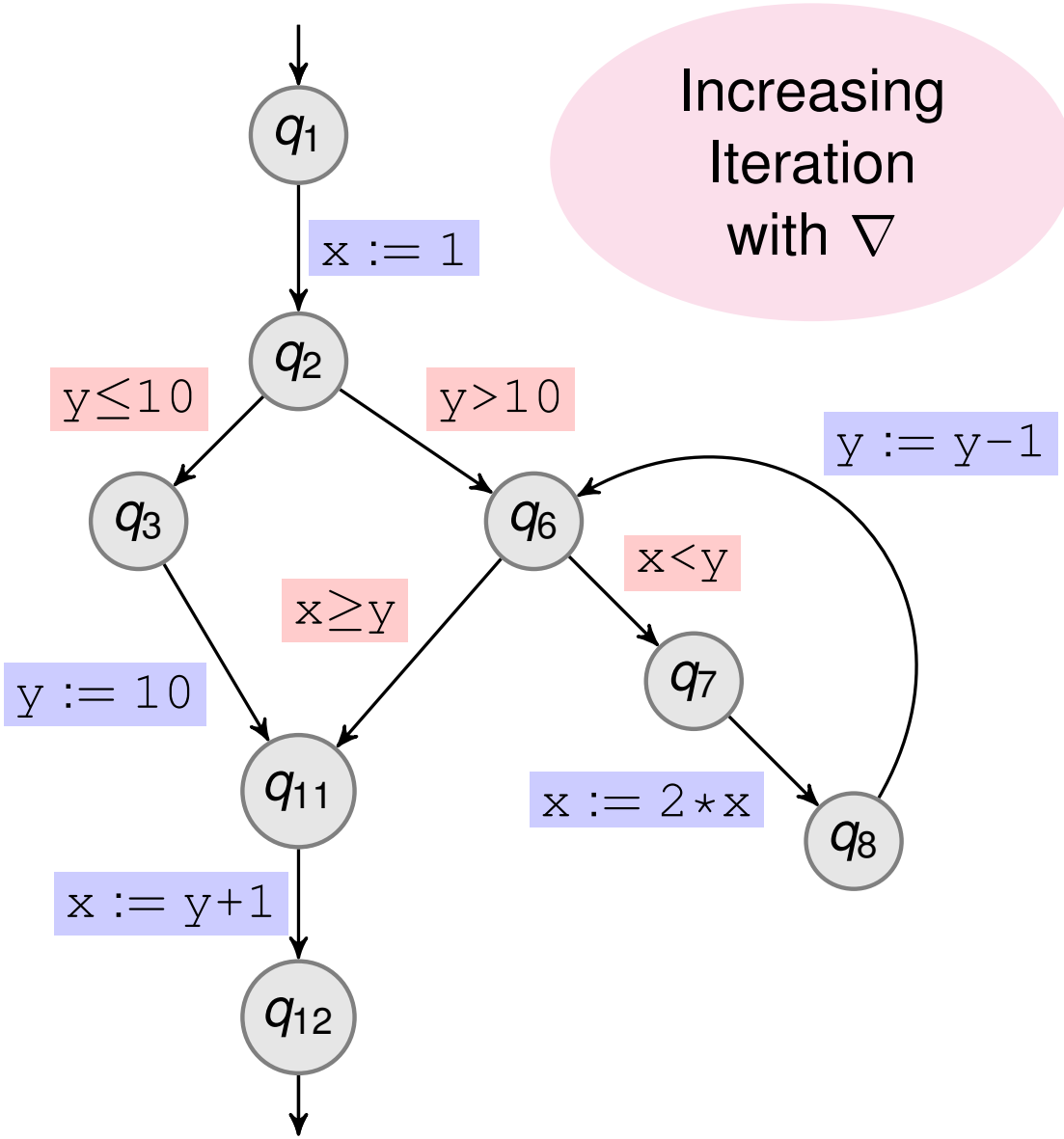


Increasing Iteration with ∇

Show that $x > 0$ at q_{12}

	x		y	
q_1	$-\infty$	$+\infty$	$-\infty$	$+\infty$
q_2	$\bot$	$\bot$	$\bot$	$\bot$
q_3	$\bot$	$\bot$	$\bot$	$\bot$
q_6	$\bot$	$\bot$	$\bot$	$\bot$
q_7	$\bot$	$\bot$	$\bot$	$\bot$
q_8	$\bot$	$\bot$	$\bot$	$\bot$
q_{11}	$\bot$	$\bot$	$\bot$	$\bot$
q_{12}	$\bot$	$\bot$	$\bot$	$\bot$

Range Analysis on Example with ∇ and Δ



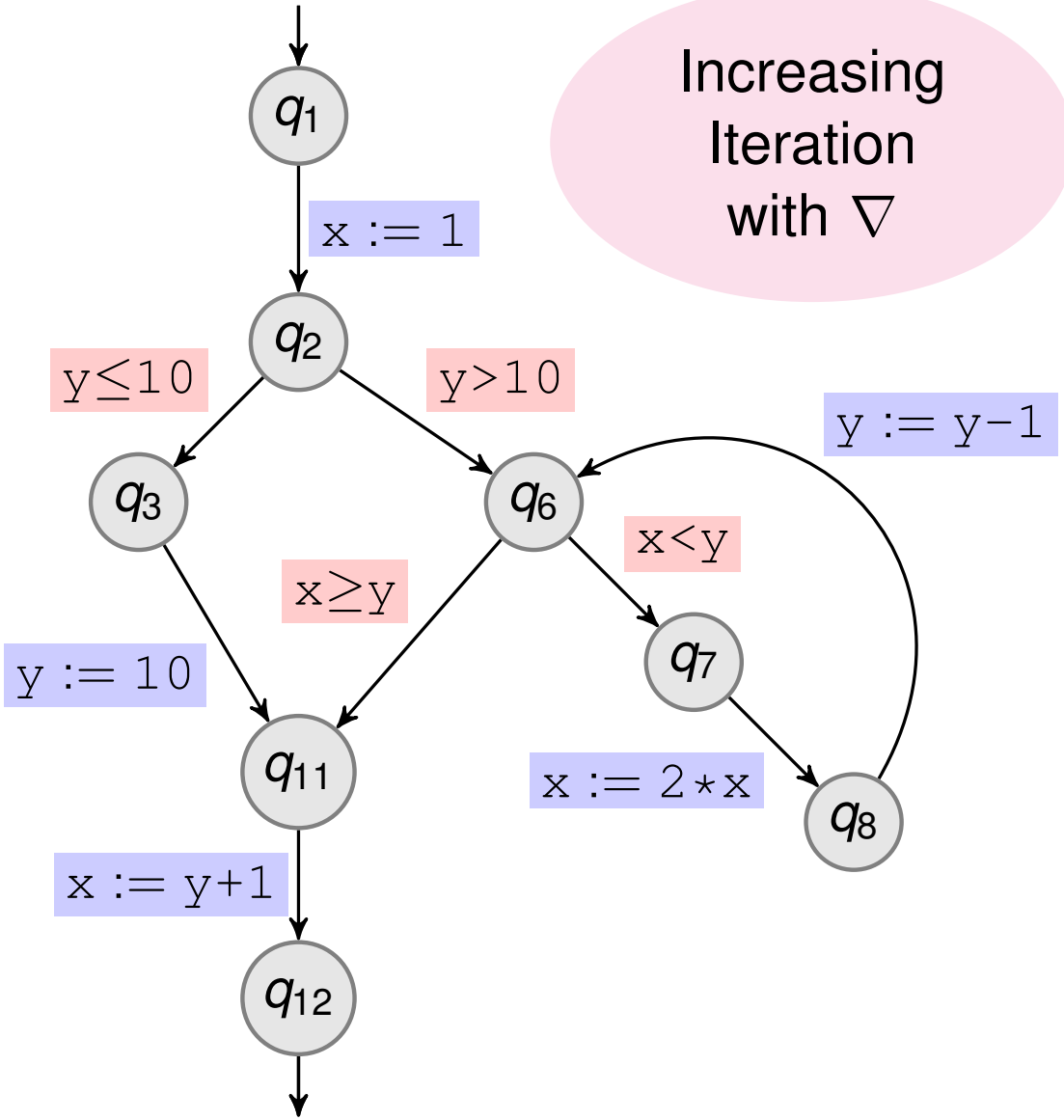
Increasing Iteration with ∇

Show that $x > 0$ at q_{12}

	x		y	
q_1	$-\infty$	$+\infty$	$-\infty$	$+\infty$
q_2	1	1	$+\infty$	$+\infty$
q_3	1	1	$-\infty$	10
q_6	1	1	10	$+\infty$
q_7	1	1	10	$+\infty$
q_8	2	2	10	$+\infty$
q_{11}	1	1	10	10
q_{12}	11	11	10	10

Same as with non-delayed widening

Range Analysis on Example with ∇ and Δ



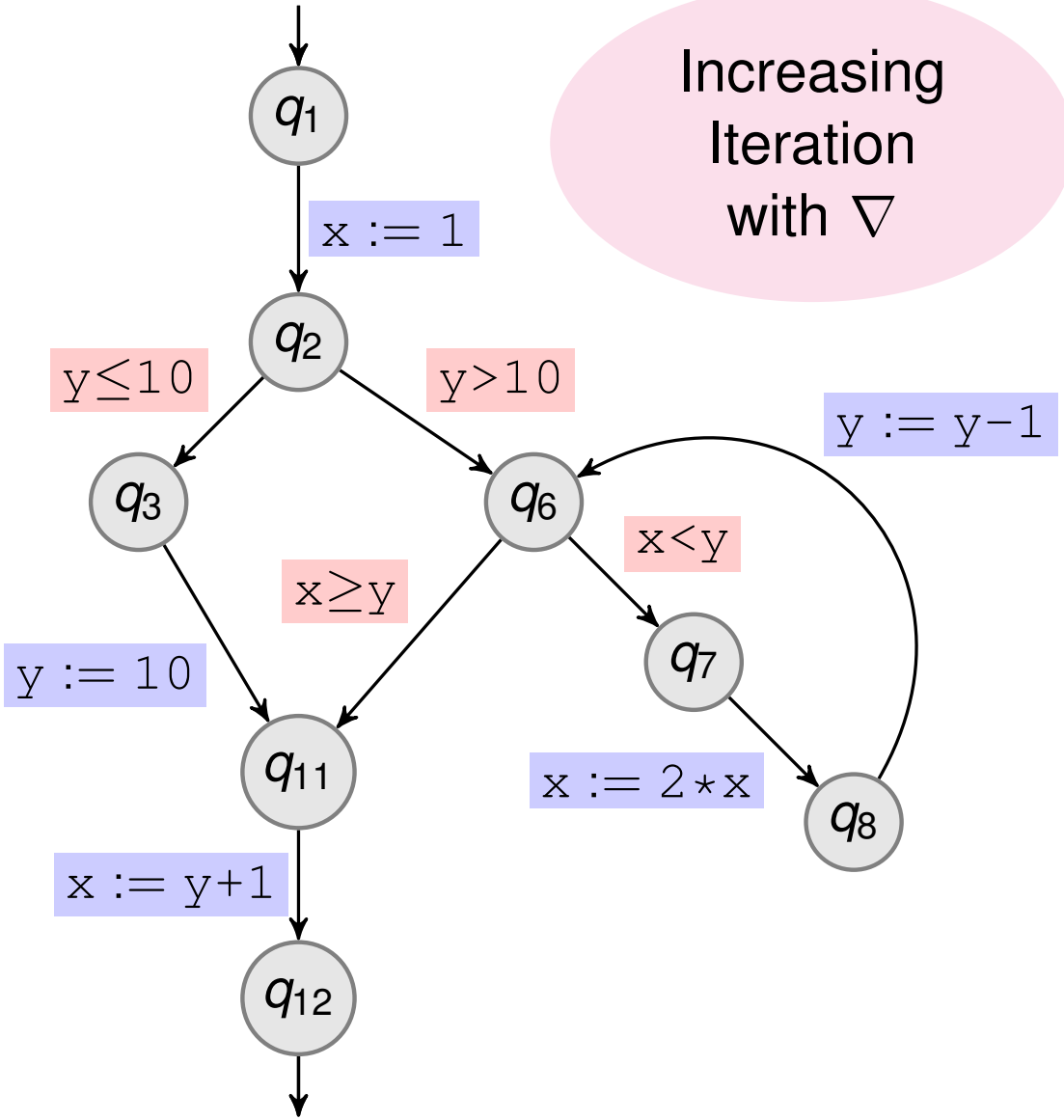
Increasing Iteration with ∇

Show that $x > 0$ at q_{12}

	x		y	
q_1	$-\infty$	$+\infty$	$-\infty$	$+\infty$
q_2	1	1	$+\infty$	$+\infty$
q_3	1	1	$-\infty$	10
q_6	1	1	10	$+\infty$
q_7	1	1	10	$+\infty$
q_8	2	2	10	$+\infty$
q_{11}	1	1	10	10
q_{12}	11	11	10	10

Same as with non-delayed widening

Range Analysis on Example with ∇ and Δ



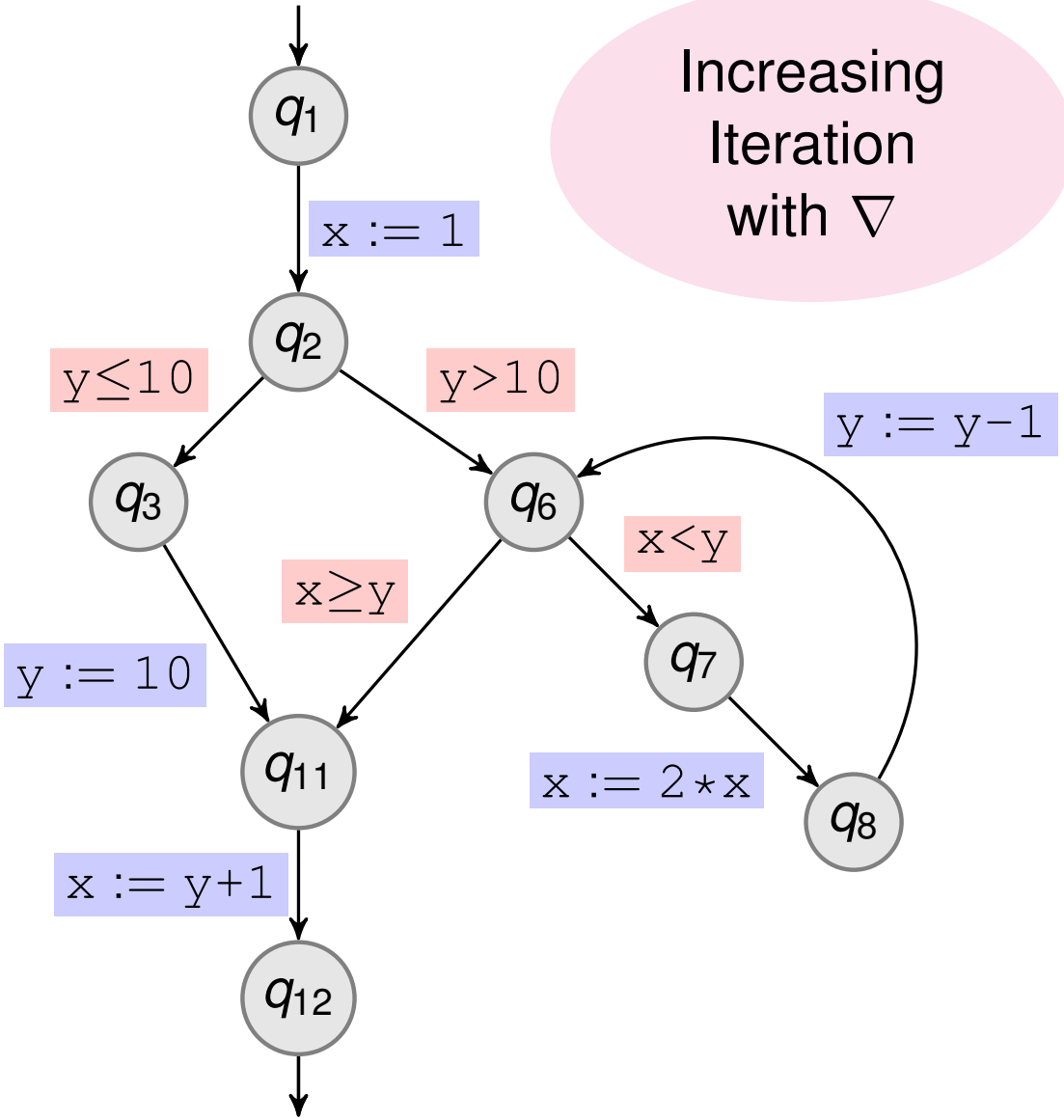
Increasing Iteration with ∇

Show that $x > 0$ at q_{12}

	x		y	
q_1	$-\infty$	$+\infty$	$-\infty$	$+\infty$
q_2	1	1	$+\infty$	$+\infty$
q_3	1	1	$-\infty$	10
q_6	1	$+\infty$	$-\infty$	$+\infty$
q_7	1	1	10	$+\infty$
q_8	2	2	10	$+\infty$
q_{11}	1	1	10	10
q_{12}	11	11	10	10

Same as with non-delayed widening

Range Analysis on Example with ∇ and Δ



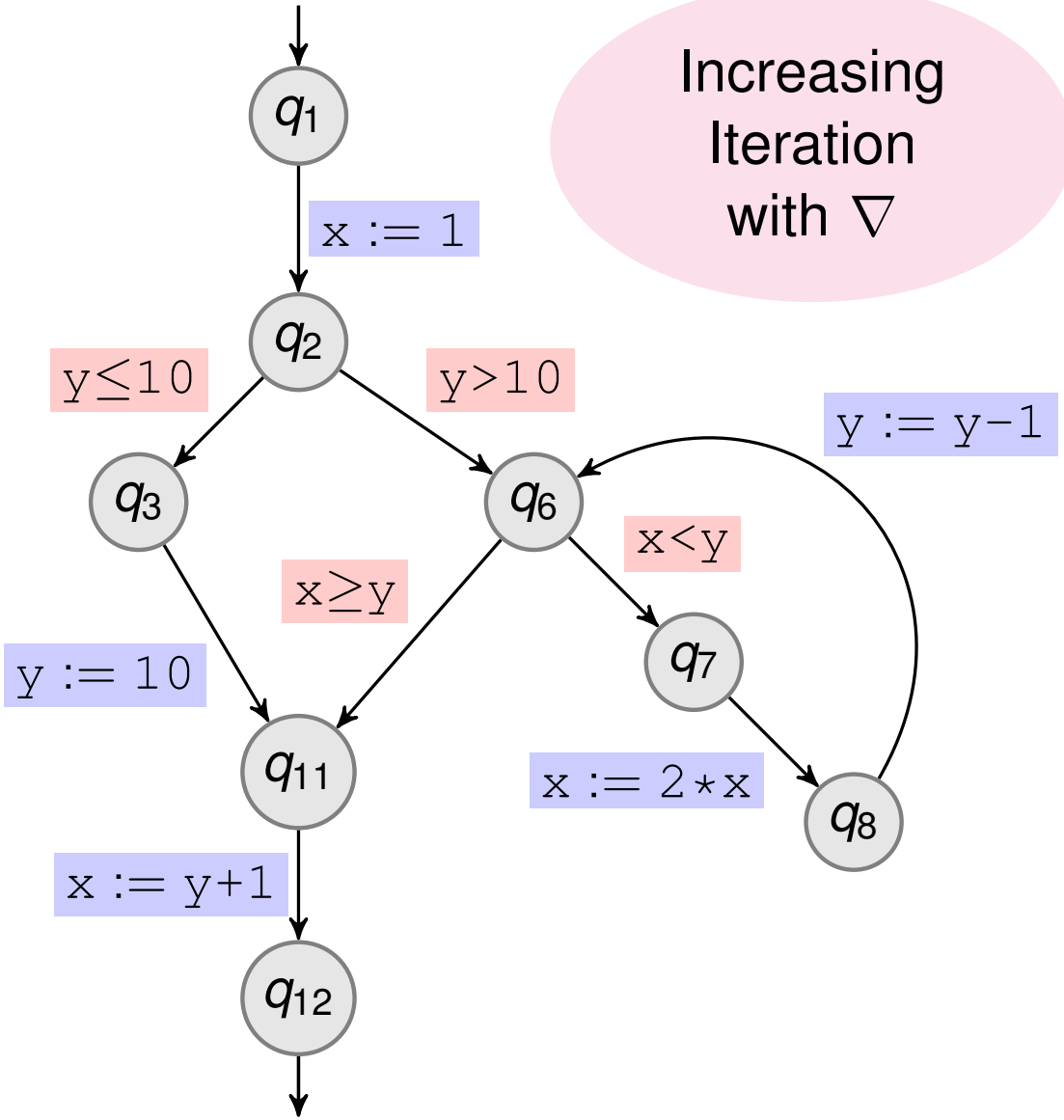
Increasing Iteration with ∇

Show that $x > 0$ at q_{12}

	x		y	
q_1	$-\infty$	$+\infty$	$-\infty$	$+\infty$
q_2	1	1	$+\infty$	$+\infty$
q_3	1	1	$-\infty$	10
q_6	1	$+\infty$	$-\infty$	$+\infty$
q_7	1	$+\infty$	$-\infty$	$+\infty$
q_8	2	2	10	$+\infty$
q_{11}	1	1	10	10
q_{12}	11	11	10	10

Same as with non-delayed widening

Range Analysis on Example with ∇ and Δ



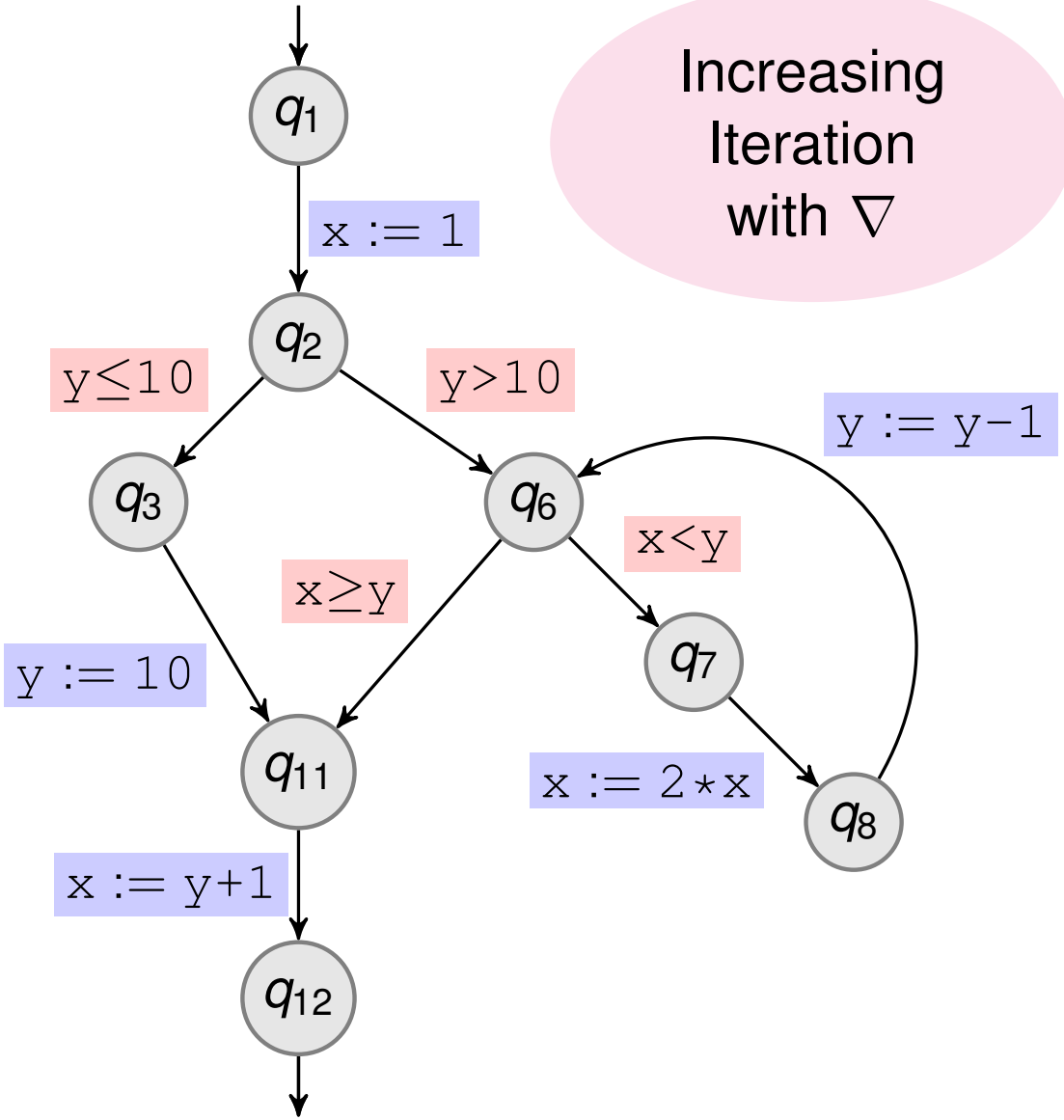
Increasing Iteration with ∇

Show that $x > 0$ at q_{12}

	x		y	
q_1	$-\infty$	$+\infty$	$-\infty$	$+\infty$
q_2	1	1	$+\infty$	$+\infty$
q_3	1	1	$-\infty$	10
q_6	1	$+\infty$	$-\infty$	$+\infty$
q_7	1	$+\infty$	$-\infty$	$+\infty$
q_8	2	$+\infty$	$-\infty$	$+\infty$
q_{11}	1	1	10	10
q_{12}	11	11	10	10

Same as with non-delayed widening

Range Analysis on Example with ∇ and Δ



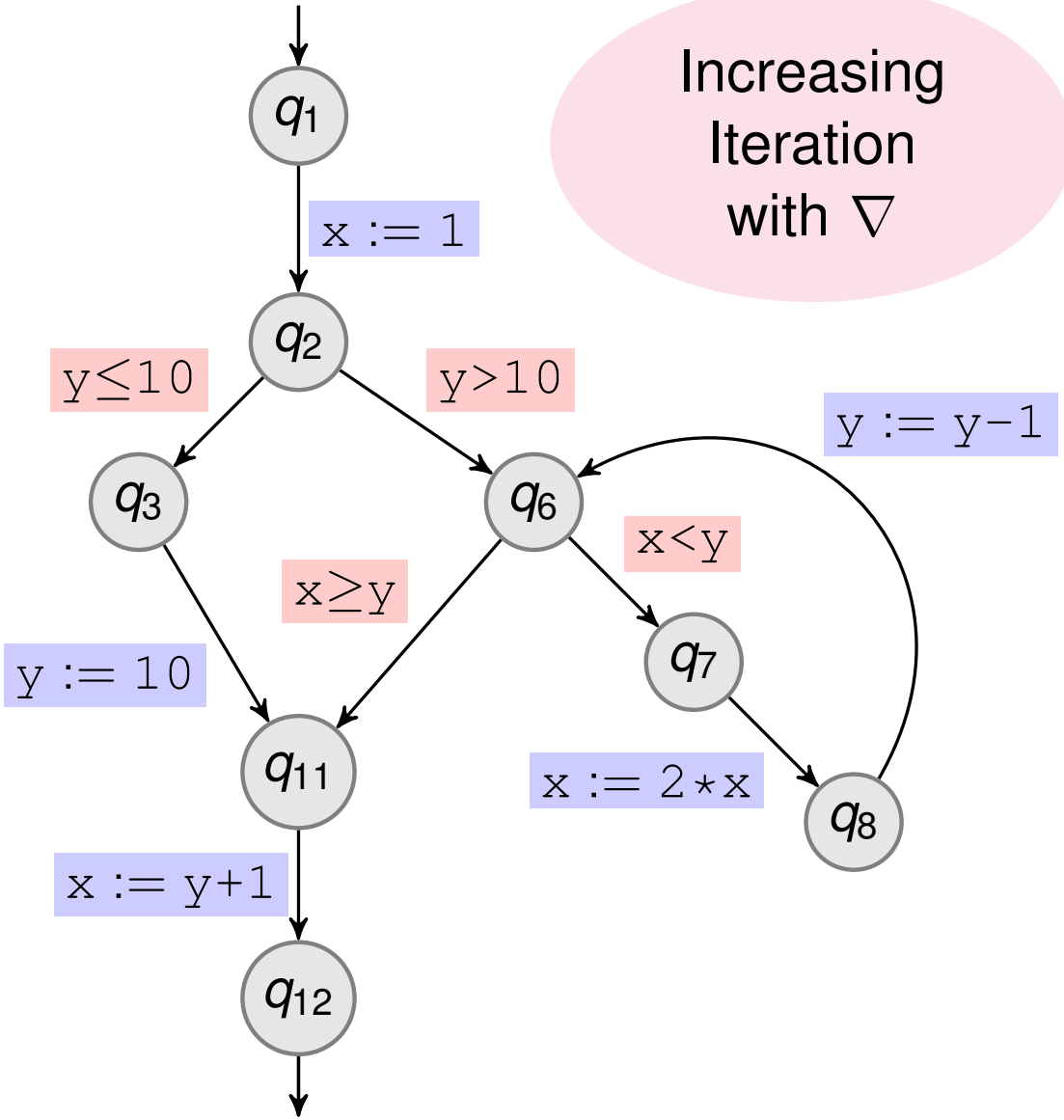
Increasing Iteration with ∇

Show that $x > 0$ at q_{12}

	x		y	
q_1	$-\infty$	$+\infty$	$-\infty$	$+\infty$
q_2	1	1	$+\infty$	$+\infty$
q_3	1	1	$-\infty$	10
q_6	1	$+\infty$	$-\infty$	$+\infty$
q_7	1	$+\infty$	$-\infty$	$+\infty$
q_8	2	$+\infty$	$-\infty$	$+\infty$
q_{11}	1	$+\infty$	$-\infty$	$+\infty$
q_{12}	11	11	10	10

Same as with non-delayed widening

Range Analysis on Example with ∇ and Δ



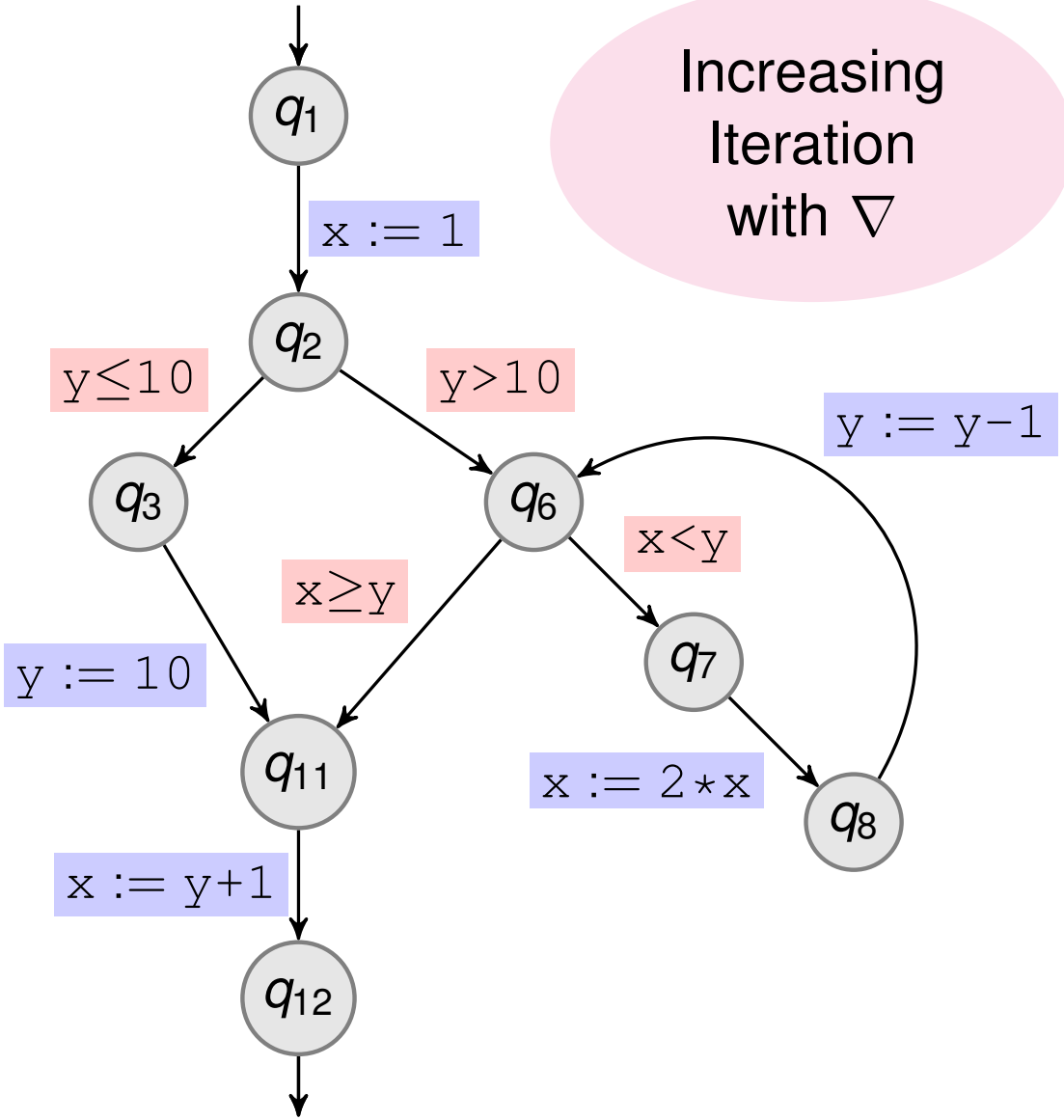
Increasing Iteration with ∇

Show that $x > 0$ at q_{12}

	x		y	
q_1	$-\infty$	$+\infty$	$-\infty$	$+\infty$
q_2	1	1	$+\infty$	$+\infty$
q_3	1	1	$-\infty$	10
q_6	1	$+\infty$	$-\infty$	$+\infty$
q_7	1	$+\infty$	$-\infty$	$+\infty$
q_8	2	$+\infty$	$-\infty$	$+\infty$
q_{11}	1	$+\infty$	$-\infty$	$+\infty$
q_{12}	$-\infty$	$+\infty$	$-\infty$	$+\infty$

Same as with non-delayed widening

Range Analysis on Example with ∇ and Δ

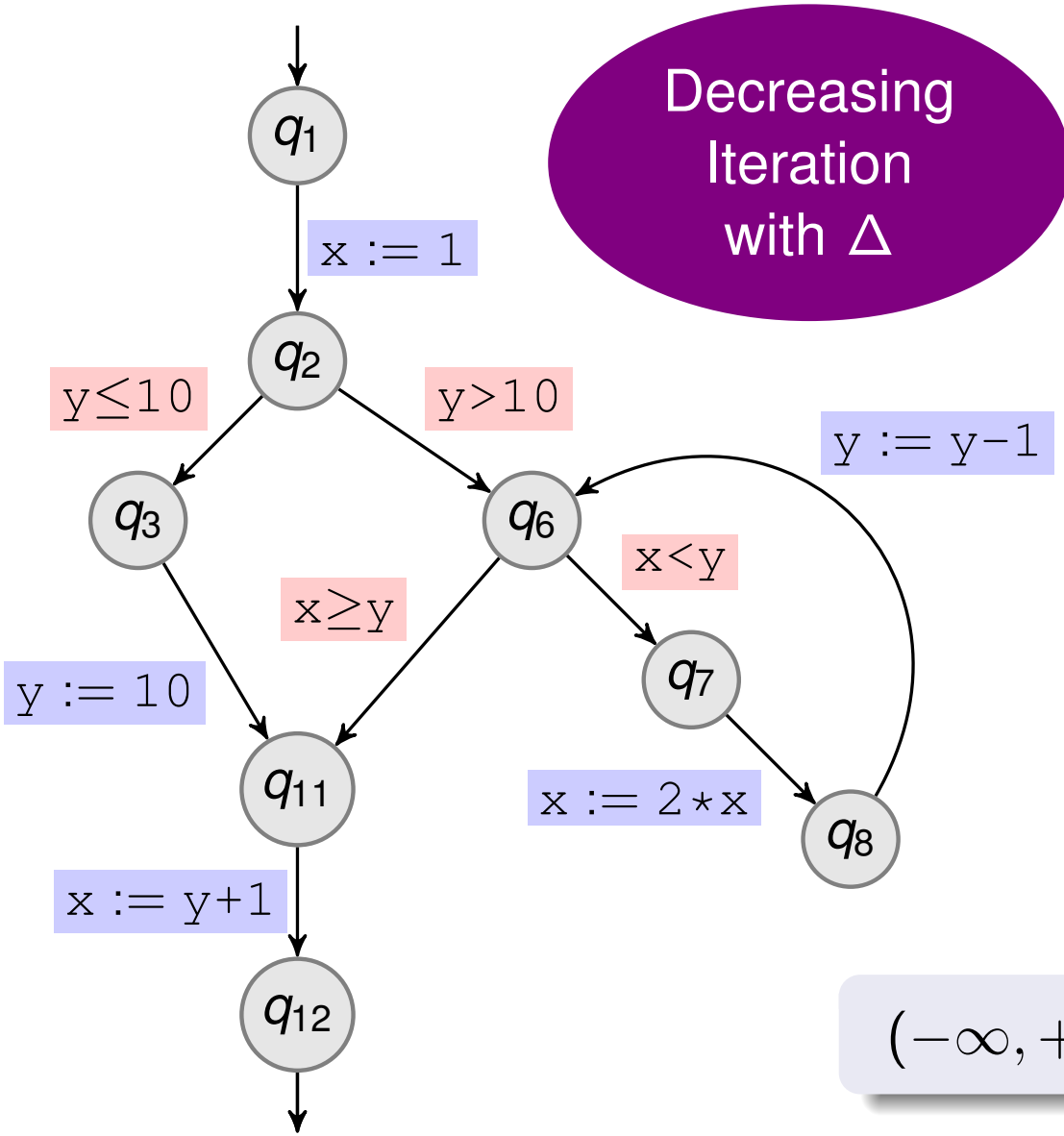


Increasing Iteration with ∇

Show that $x > 0$ at q_{12}

	x		y	
q_1	$-\infty$	$+\infty$	$-\infty$	$+\infty$
q_2	1	1	$+\infty$	$+\infty$
q_3	1	1	$-\infty$	10
q_6	1	$+\infty$	$-\infty$	$+\infty$
q_7	1	$+\infty$	$-\infty$	$+\infty$
q_8	2	$+\infty$	$-\infty$	$+\infty$
q_{11}	1	$+\infty$	$-\infty$	$+\infty$
q_{12}	$-\infty$	$+\infty$	$-\infty$	$+\infty$

Range Analysis on Example with ∇ and Δ

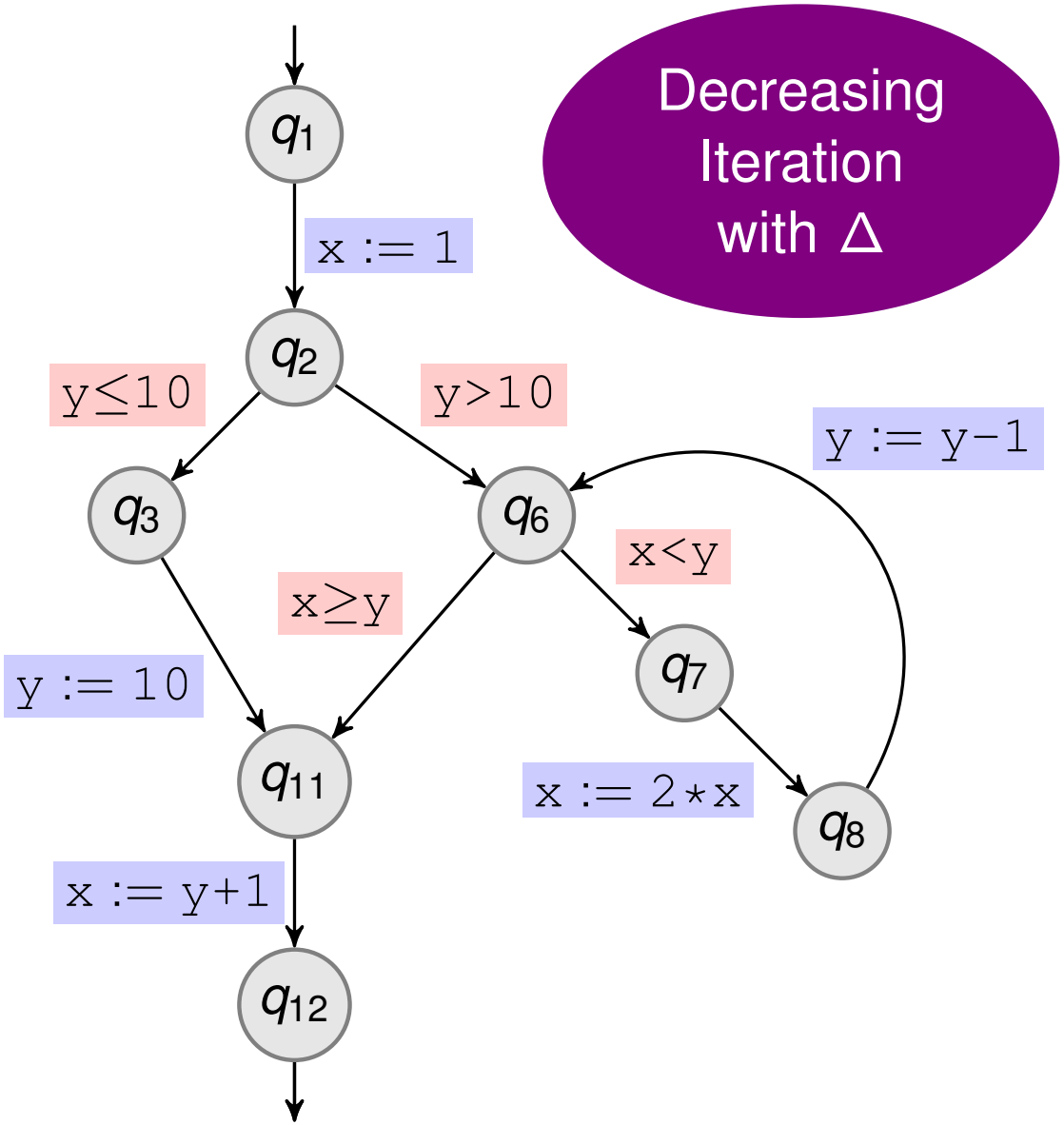


Show that $\mathtt{x} > 0$ at q_{12}

	x		y	
q_1	$-\infty$	$+\infty$	$-\infty$	$+\infty$
q_2	1	1	$+\infty$	$+\infty$
q_3	1	1	$-\infty$	10
q_6	1	$+\infty$	$-\infty$	$+\infty$
q_7	1	$+\infty$	1	$+\infty$
q_8	2	$+\infty$	$-\infty$	$+\infty$
q_{11}	1	$+\infty$	$-\infty$	$+\infty$
q_{12}	$-\infty$	$+\infty$	$-\infty$	$+\infty$

$$(-\infty, +\infty) \ \Delta \ (1, +\infty) \ = \ (1, +\infty)$$

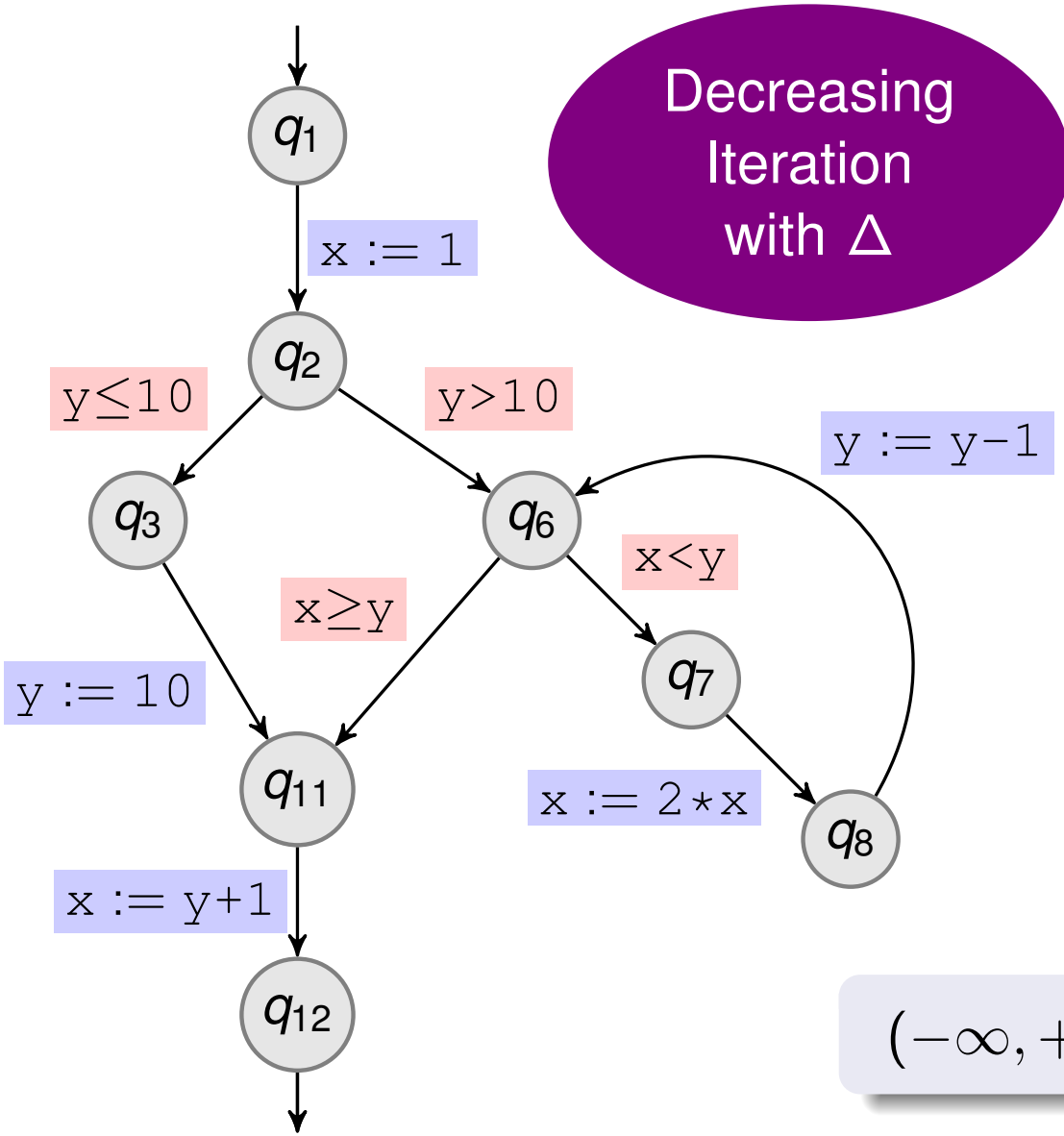
Range Analysis on Example with ∇ and Δ



Show that $x > 0$ at q_{12}

	x		y	
q_1	$-\infty$	$+\infty$	$-\infty$	$+\infty$
q_2	1	1	$+\infty$	$+\infty$
q_3	1	1	$-\infty$	10
q_6	1	$+\infty$	$-\infty$	$+\infty$
q_7	1	$+\infty$	1	$+\infty$
q_8	2	$+\infty$	$-\infty$	$+\infty$
q_{11}	1	$+\infty$	$-\infty$	$+\infty$
q_{12}	$-\infty$	$+\infty$	$-\infty$	$+\infty$

Range Analysis on Example with ∇ and Δ

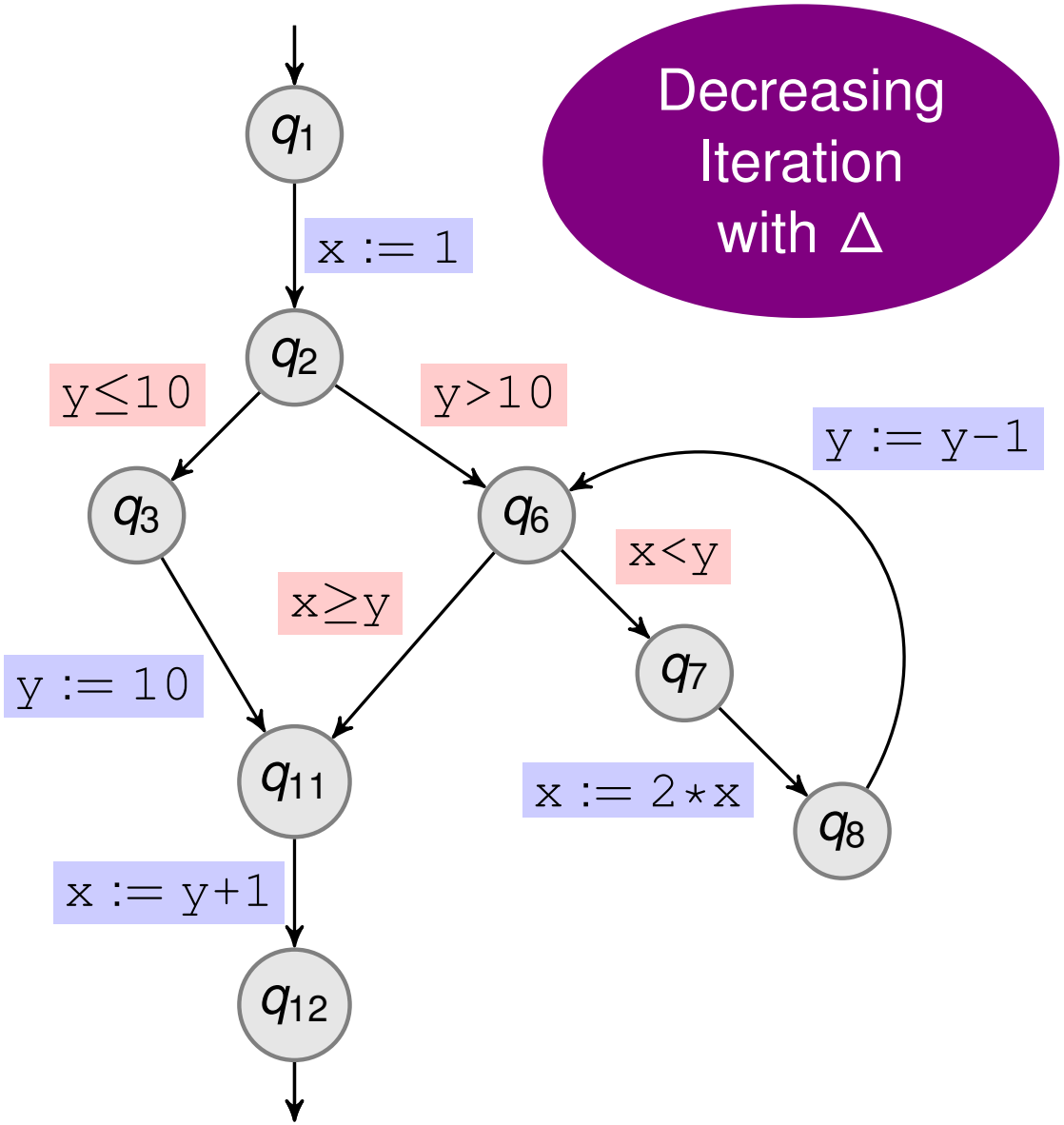


Show that $x > 0$ at q_{12}

	x		y	
q_1	$-\infty$	$+\infty$	$-\infty$	$+\infty$
q_2	1	1	$+\infty$	$+\infty$
q_3	1	1	$-\infty$	10
q_6	1	$+\infty$	$-\infty$	$+\infty$
q_7	1	$+\infty$	1	$+\infty$
q_8	2	$+\infty$	1	$+\infty$
q_{11}	1	$+\infty$	$-\infty$	$+\infty$
q_{12}	$-\infty$	$+\infty$	$-\infty$	$+\infty$

$$(-\infty, +\infty) \ \Delta \ (1, +\infty) \ = \ (1, +\infty)$$

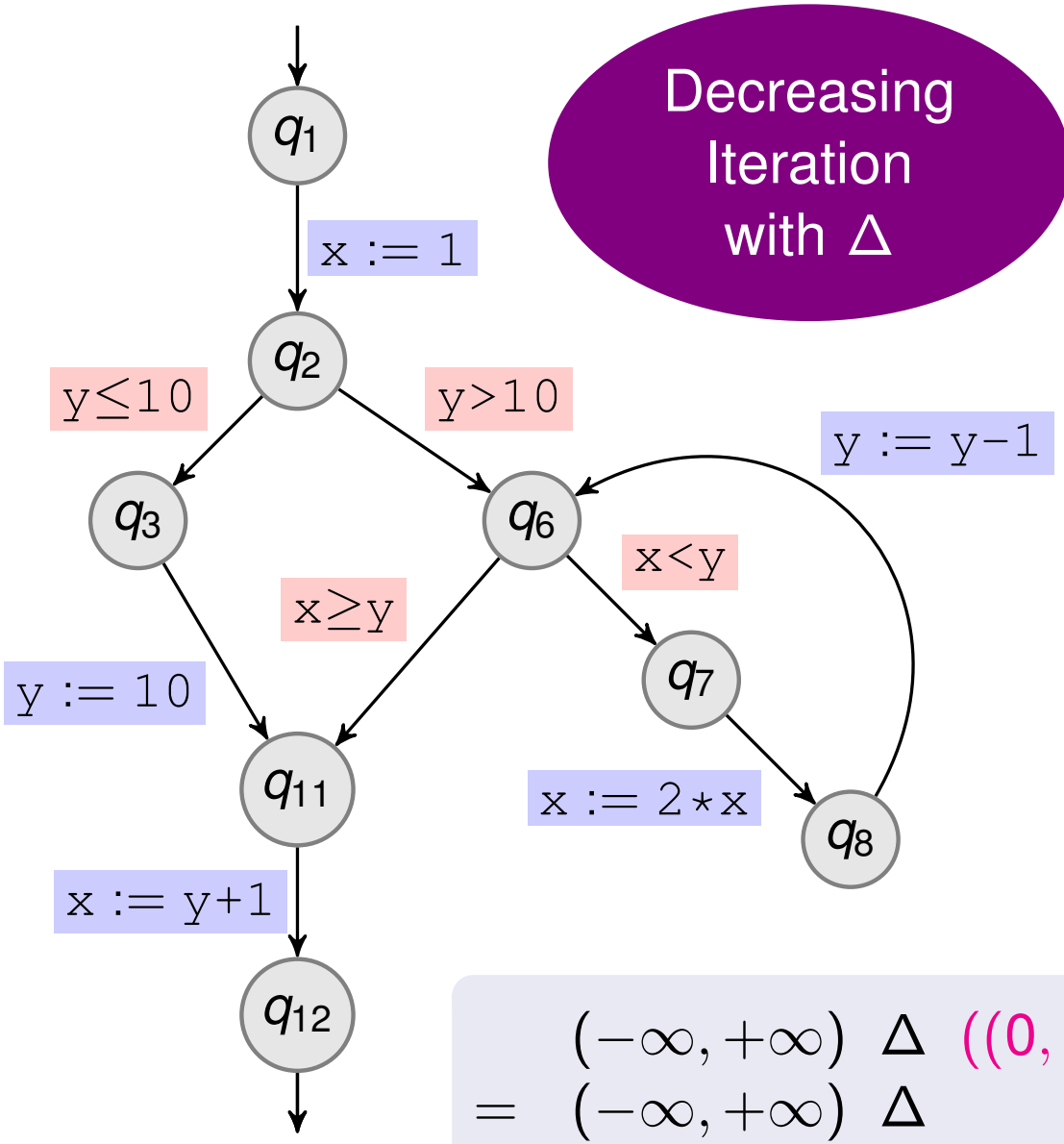
Range Analysis on Example with ∇ and Δ



Show that $x > 0$ at q_{12}

	x		y	
q_1	$-\infty$	$+\infty$	$-\infty$	$+\infty$
q_2	1	1	$+\infty$	$+\infty$
q_3	1	1	$-\infty$	10
q_6	1	$+\infty$	$-\infty$	$+\infty$
q_7	1	$+\infty$	1	$+\infty$
q_8	2	$+\infty$	1	$+\infty$
q_{11}	1	$+\infty$	$-\infty$	$+\infty$
q_{12}	$-\infty$	$+\infty$	$-\infty$	$+\infty$

Range Analysis on Example with ∇ and Δ

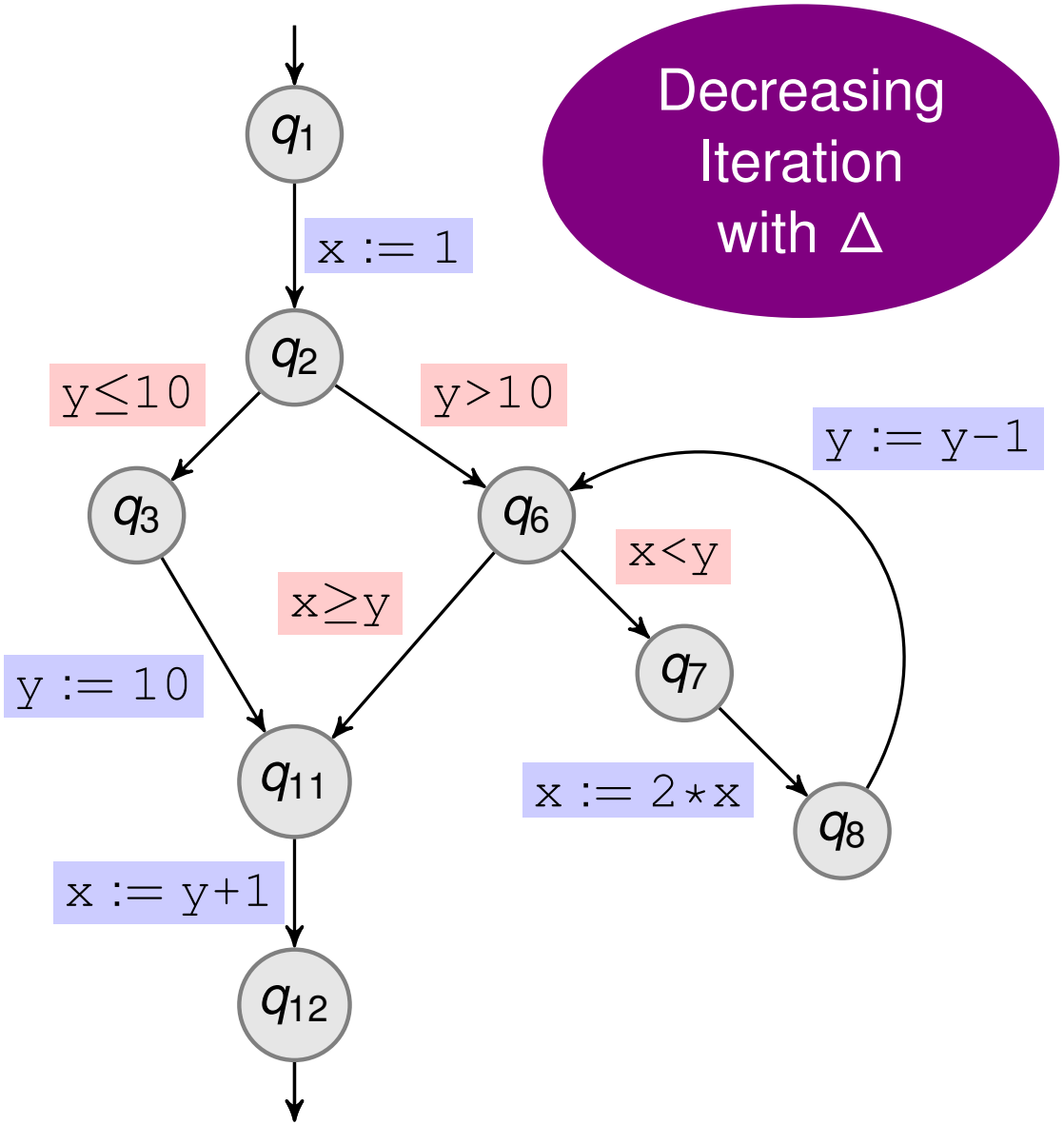


Show that $x > 0$ at q_{12}

	x		y	
q_1	$-\infty$	$+\infty$	$-\infty$	$+\infty$
q_2	1	1	$+\infty$	$+\infty$
q_3	1	1	$-\infty$	10
q_6	1	$+\infty$	0	$+\infty$
q_7	1	$+\infty$	1	$+\infty$
q_8	2	$+\infty$	1	$+\infty$
q_{11}	1	$+\infty$	$-\infty$	$+\infty$
q_{12}	$-\infty$	$+\infty$	$-\infty$	$+\infty$

$$\begin{array}{rcl} & (-\infty,+\infty) \ \Delta \ ((0,+\infty) \sqcup (10,+\infty)) & \\ = & (-\infty,+\infty) \ \Delta \ (0,+\infty) & = (0,+\infty) \end{array}$$

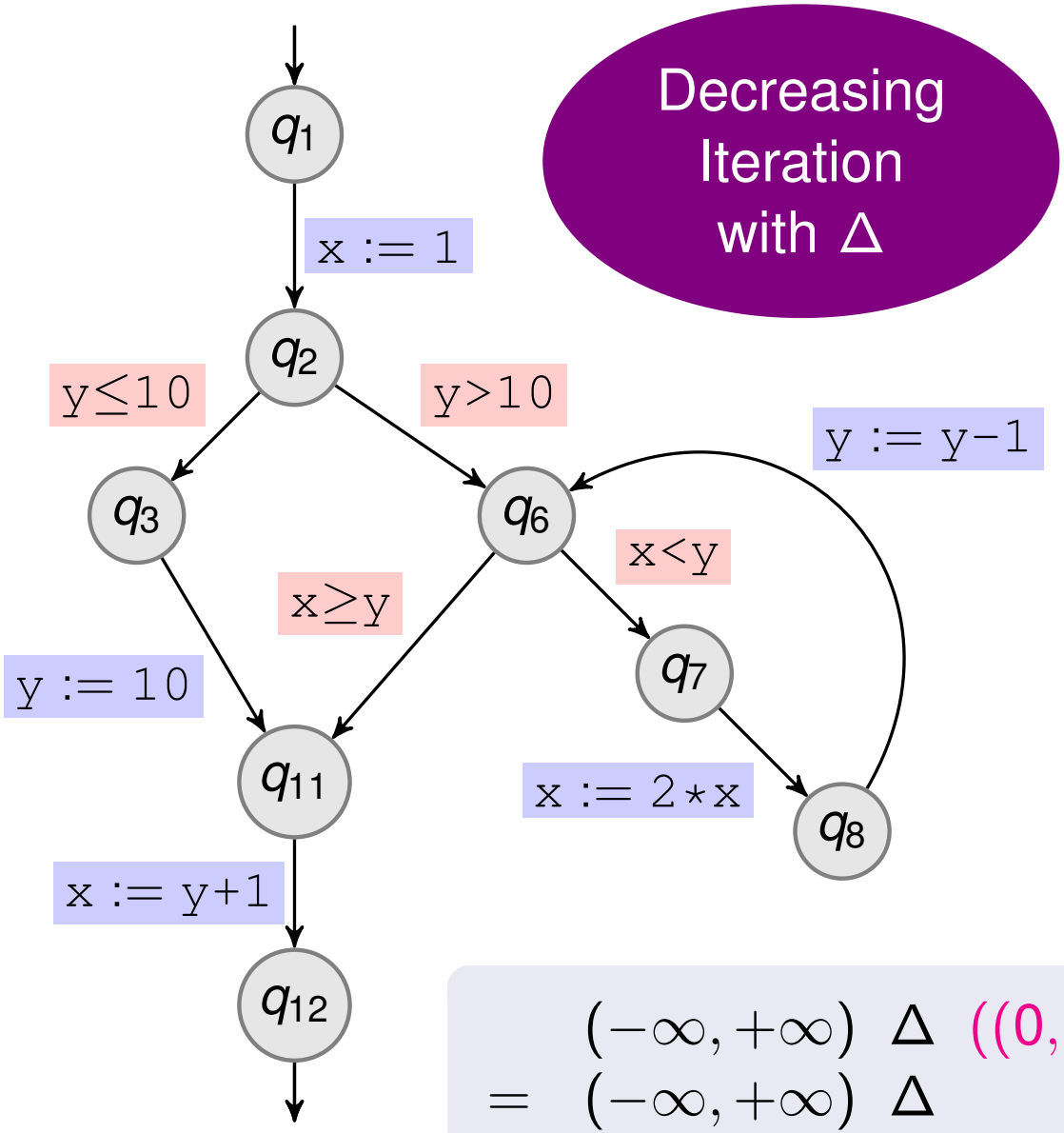
Range Analysis on Example with ∇ and Δ



Show that $x > 0$ at q_{12}

	x		y	
q_1	$-\infty$	$+\infty$	$-\infty$	$+\infty$
q_2	1	1	$+\infty$	$+\infty$
q_3	1	1	$-\infty$	10
q_6	1	$+\infty$	0	$+\infty$
q_7	1	$+\infty$	1	$+\infty$
q_8	2	$+\infty$	1	$+\infty$
q_{11}	1	$+\infty$	$-\infty$	$+\infty$
q_{12}	$-\infty$	$+\infty$	$-\infty$	$+\infty$

Range Analysis on Example with ∇ and Δ

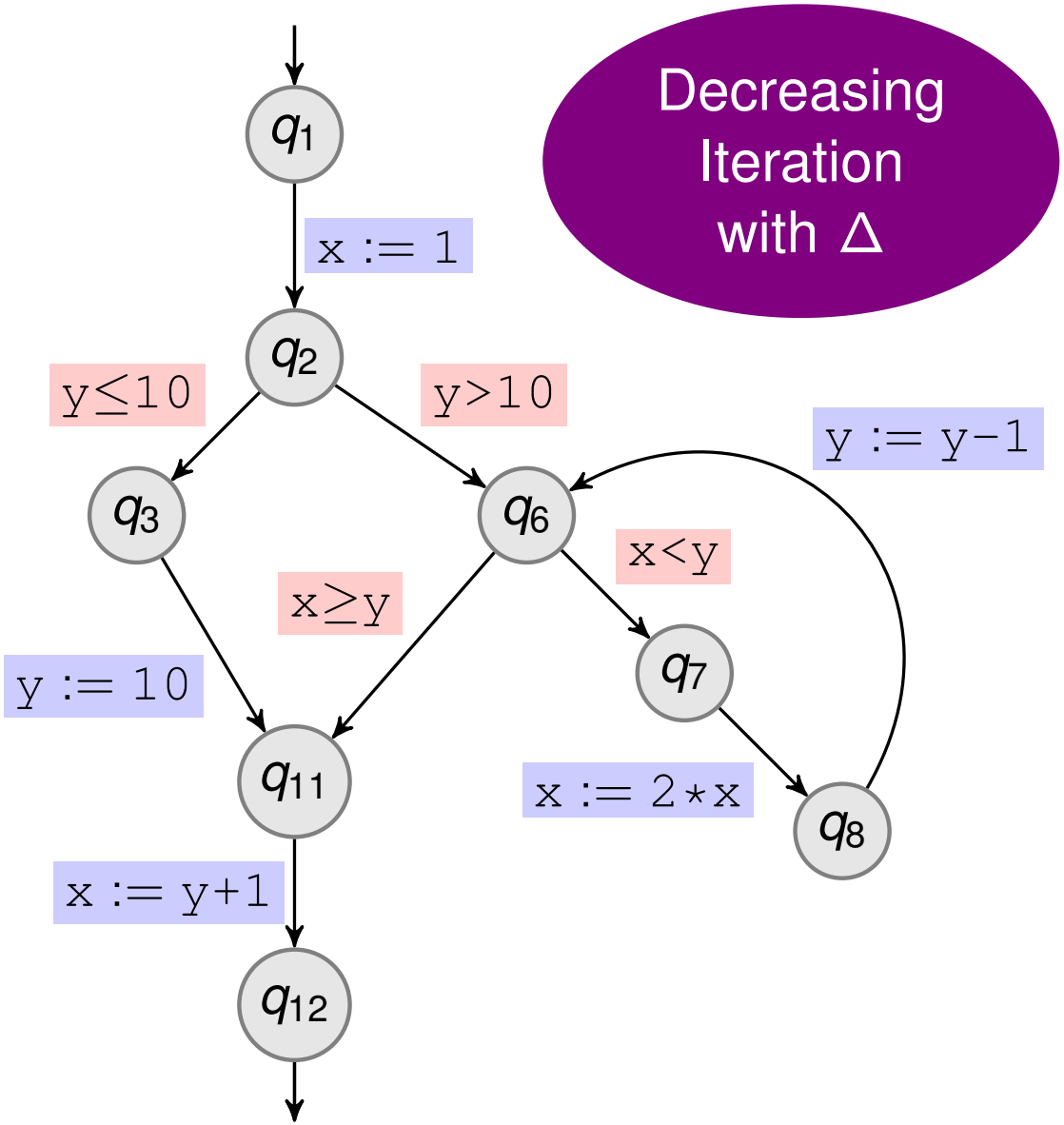


Show that $x > 0$ at q_{12}

	x		y	
q_1	$-\infty$	$+\infty$	$-\infty$	$+\infty$
q_2	1	1	$+\infty$	$+\infty$
q_3	1	1	$-\infty$	10
q_6	1	$+\infty$	0	$+\infty$
q_7	1	$+\infty$	1	$+\infty$
q_8	2	$+\infty$	1	$+\infty$
q_{11}	1	$+\infty$	0	$+\infty$
q_{12}	$-\infty$	$+\infty$	$-\infty$	$+\infty$

$$\begin{array}{rlll} & (-\infty,+\infty) & \Delta & ((0,+\infty) \sqcup (10,10)) \\ = & (-\infty,+\infty) & \Delta & (0,+\infty) \quad = \quad (0,+\infty) \end{array}$$

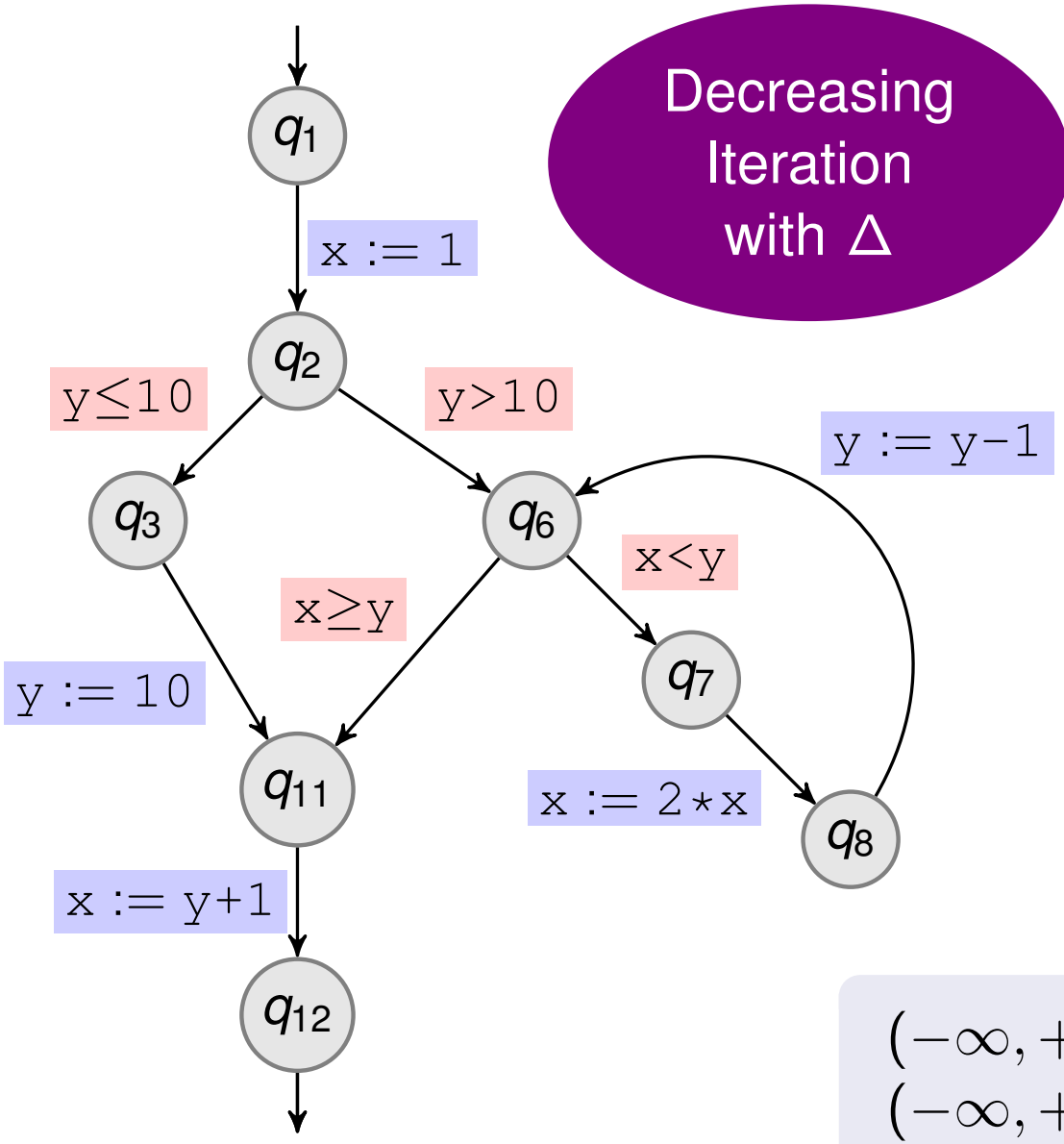
Range Analysis on Example with ∇ and Δ



Show that $x > 0$ at q_{12}

	x		y	
q_1	$-\infty$	$+\infty$	$-\infty$	$+\infty$
q_2	1	1	$+\infty$	$+\infty$
q_3	1	1	$-\infty$	10
q_6	1	$+\infty$	0	$+\infty$
q_7	1	$+\infty$	1	$+\infty$
q_8	2	$+\infty$	1	$+\infty$
q_{11}	1	$+\infty$	0	$+\infty$
q_{12}	$-\infty$	$+\infty$	$-\infty$	$+\infty$

Range Analysis on Example with ∇ and Δ

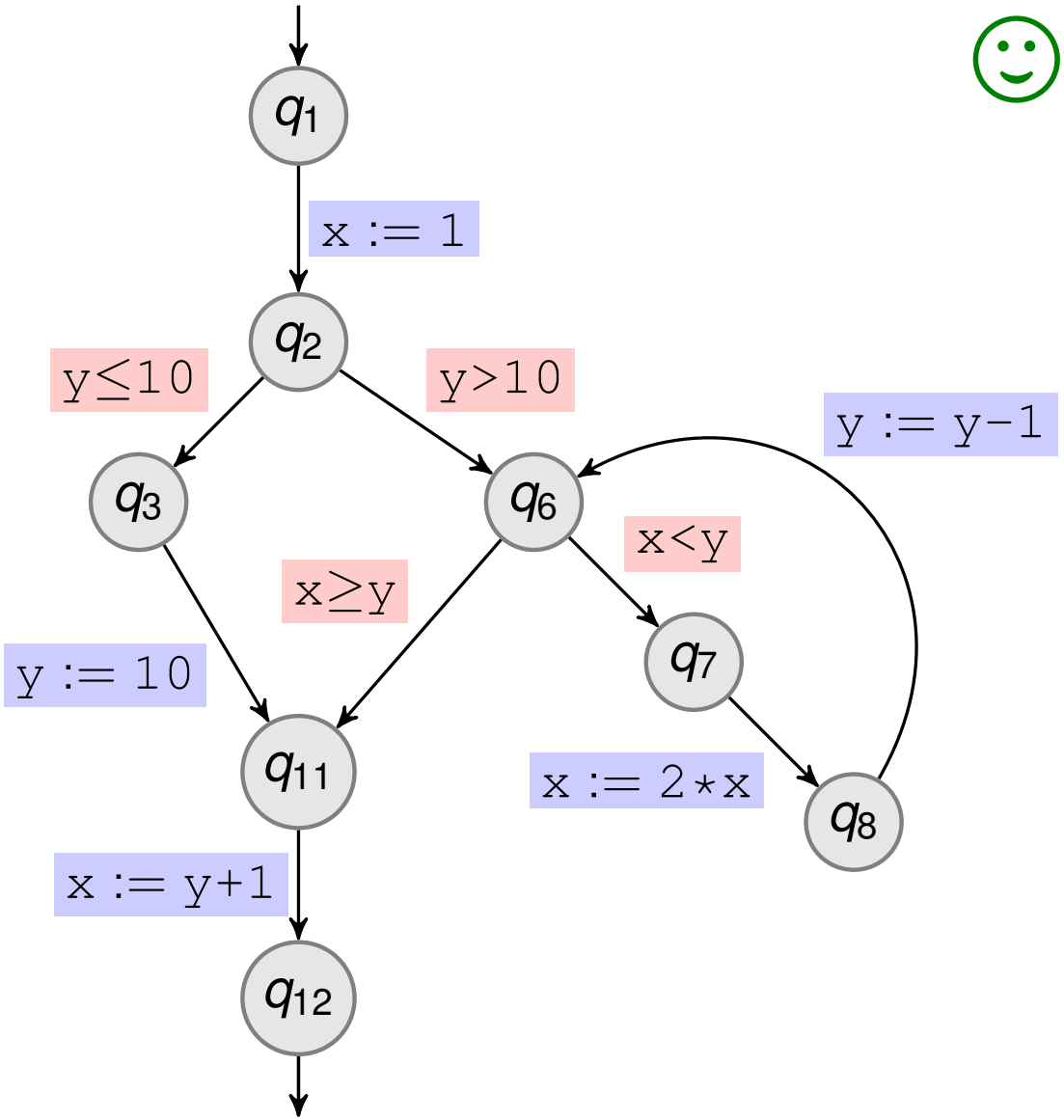


Show that $x > 0$ at q_{12}

	x		y	
q_1	$-\infty$	$+\infty$	$-\infty$	$+\infty$
q_2	1	1	$+\infty$	$+\infty$
q_3	1	1	$-\infty$	10
q_6	1	$+\infty$	0	$+\infty$
q_7	1	$+\infty$	1	$+\infty$
q_8	2	$+\infty$	1	$+\infty$
q_{11}	1	$+\infty$	0	$+\infty$
q_{12}	1	$+\infty$	0	$+\infty$

$$(-\infty, +\infty) \ \Delta \ (1, +\infty) = (1, +\infty)$$
$$(-\infty, +\infty) \ \Delta \ (0, +\infty) = (0, +\infty)$$

Range Analysis on Example with ∇ and Δ



Show that $x > 0$ at q_{12}

	x		y	
q_1	$-\infty$	$+\infty$	$-\infty$	$+\infty$
q_2	1	1	$+\infty$	$+\infty$
q_3	1	1	$-\infty$	10
q_6	1	$+\infty$	0	$+\infty$
q_7	1	$+\infty$	1	$+\infty$
q_8	2	$+\infty$	1	$+\infty$
q_{11}	1	$+\infty$	0	$+\infty$
q_{12}	1	$+\infty$	0	$+\infty$

Selective Application of Widening

Widening introduces imprecision that often cannot be regained by narrowing.

To ensure convergence it is enough to only apply widening at cut points

Cut points: set of locations that cut each loop (in the control flow automaton's graph)

Other Methods to Reduce Precision Loss of Widening

- Delayed widening
- Widening "up to"

  Given a finite set $M \subseteq L$, use $(x \nabla y) \sqcap \bigsqcap \{m \in M \mid a \sqcup b \sqsubseteq m\}$.
- Look-ahead widening
- ...

Selective Application of Widening

Widening introduces imprecision that often cannot be regained by narrowing.

To ensure convergence it is enough to only apply widening at cut points

Cut points: set of locations that cut each loop (in the control flow automaton's graph)

Other Methods to Reduce Precision Loss of Widening

- Delayed widening
- Widening "up to"

  Given a finite set $M \subseteq L$, use $(x \nabla y) \sqcap \bigsqcap \{m \in M \mid a \sqcup b \sqsubseteq m\}$.
- Look-ahead widening
- ...

Part V

Software Verification by Static Analysis

Outline — Software Verification by Static Analysis

Outline — Software Verification by Static Analysis

Summary: Data Flow Analysis

Compile-time techniques to gather run-time information about data in programs without actually running them

- Live Variables
- Available Expressions
- Uninitialized Variables
- Constant Propagation

Monotone Data Flow Analysis Frameworks

Minimal Fixpoint

- ☺ Computable in finite-height lattices
- ☹ Loss of Precision

Meet Over All Paths

- ☺ Most precise solution
- ☹ Undecidable (constant propagation)

Summary: Abstract Interpretation

Semantics-based systematic design of correct data flow analyses

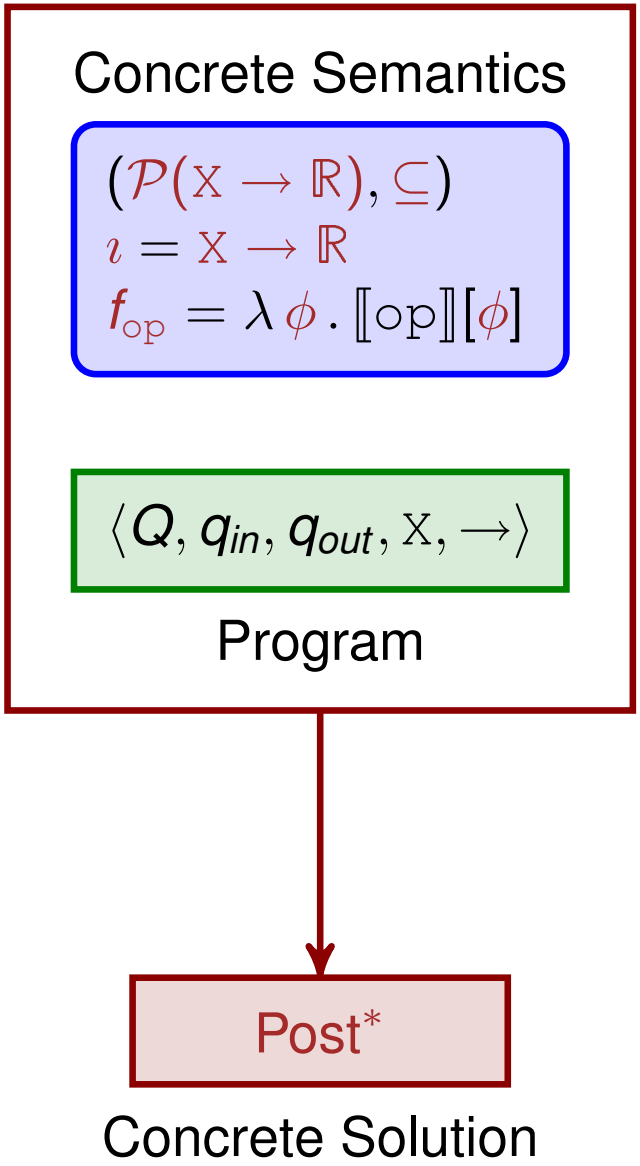
Galois connections to formally relate abstract and concrete semantics

Safe approximations of the “best” abstract semantics

Convergence acceleration with widening and narrowing

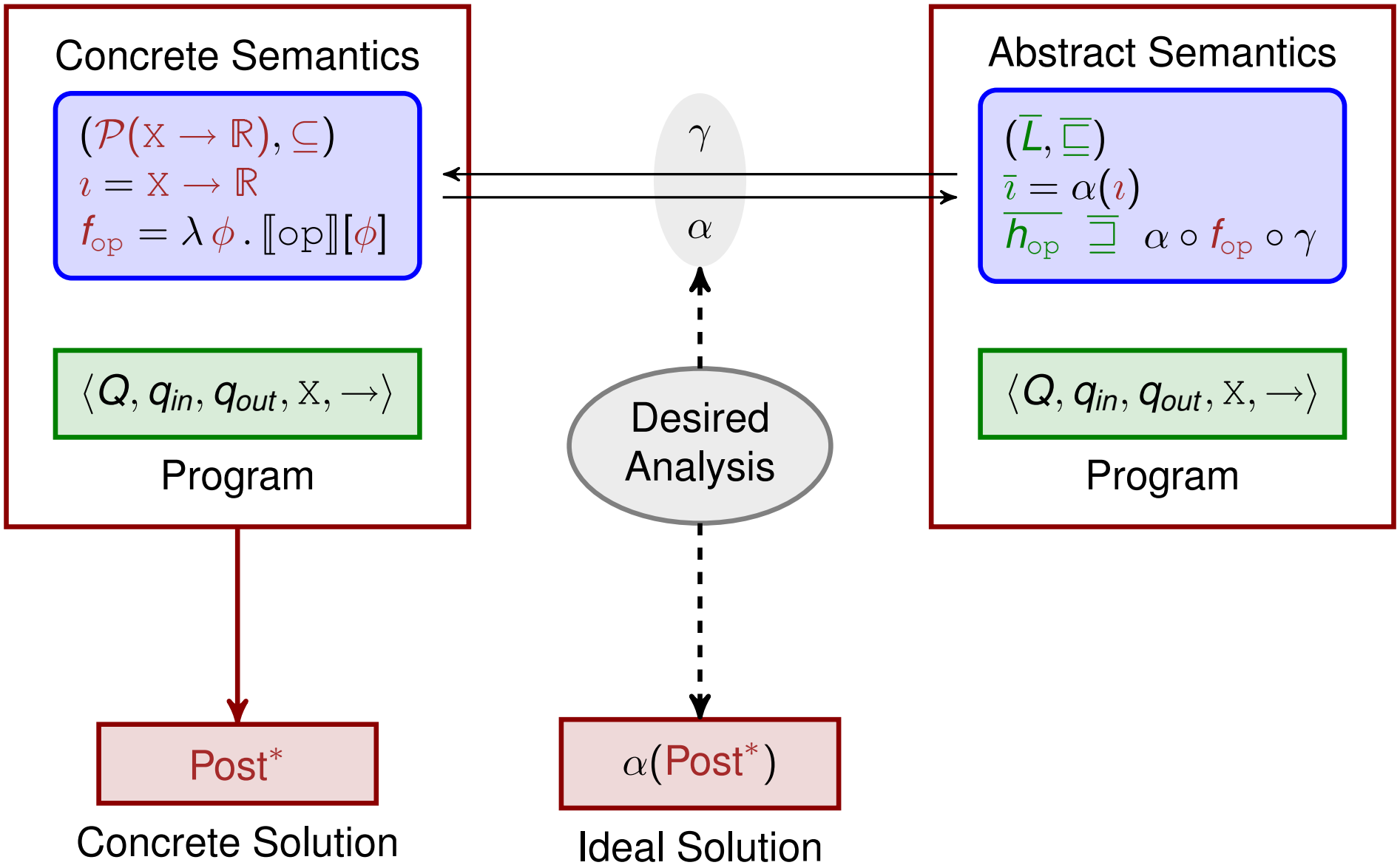
- Sign Analysis
- Range Analysis

Abstract Interpretation-Based Data Flow Analysis

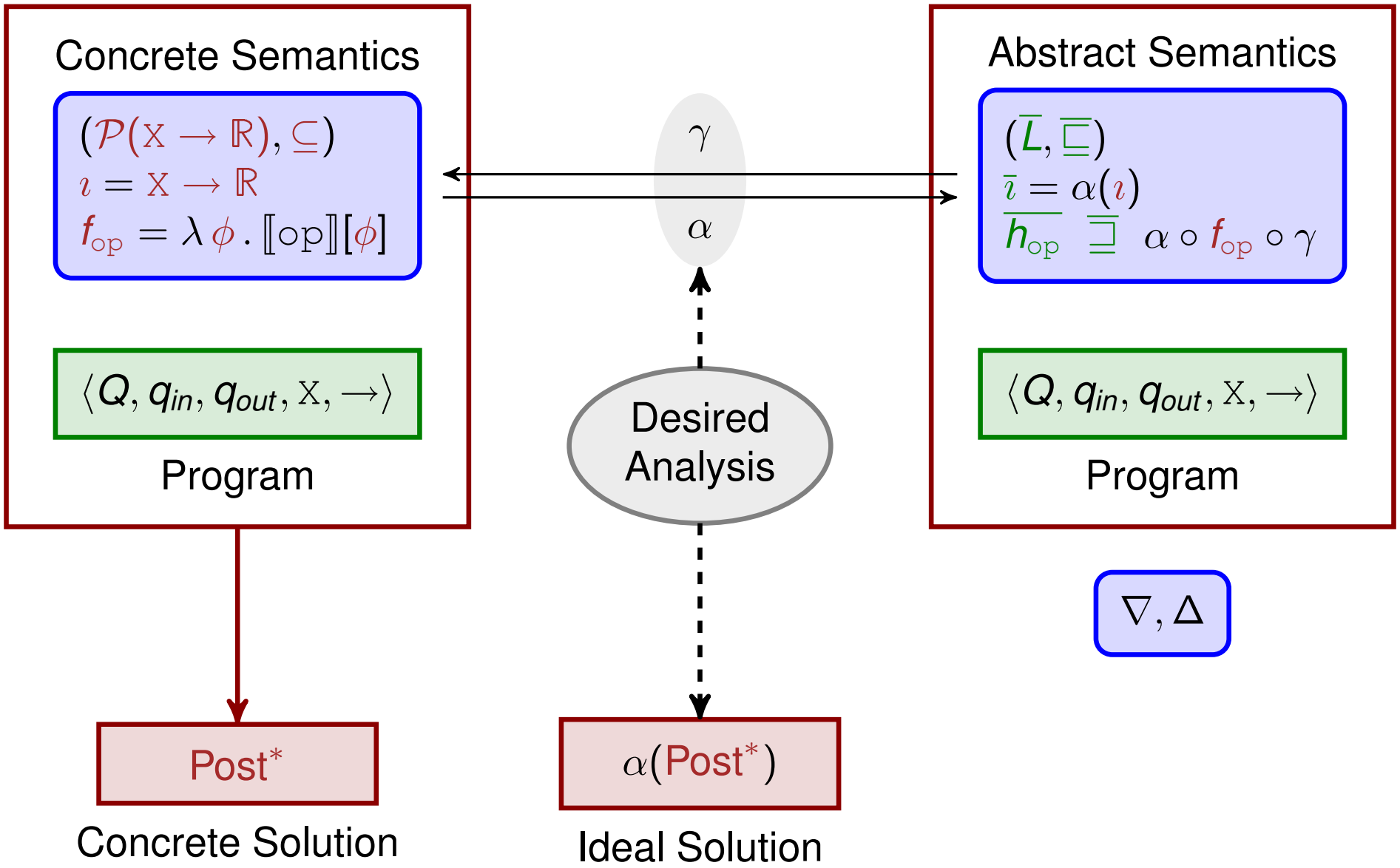


Desired Analysis

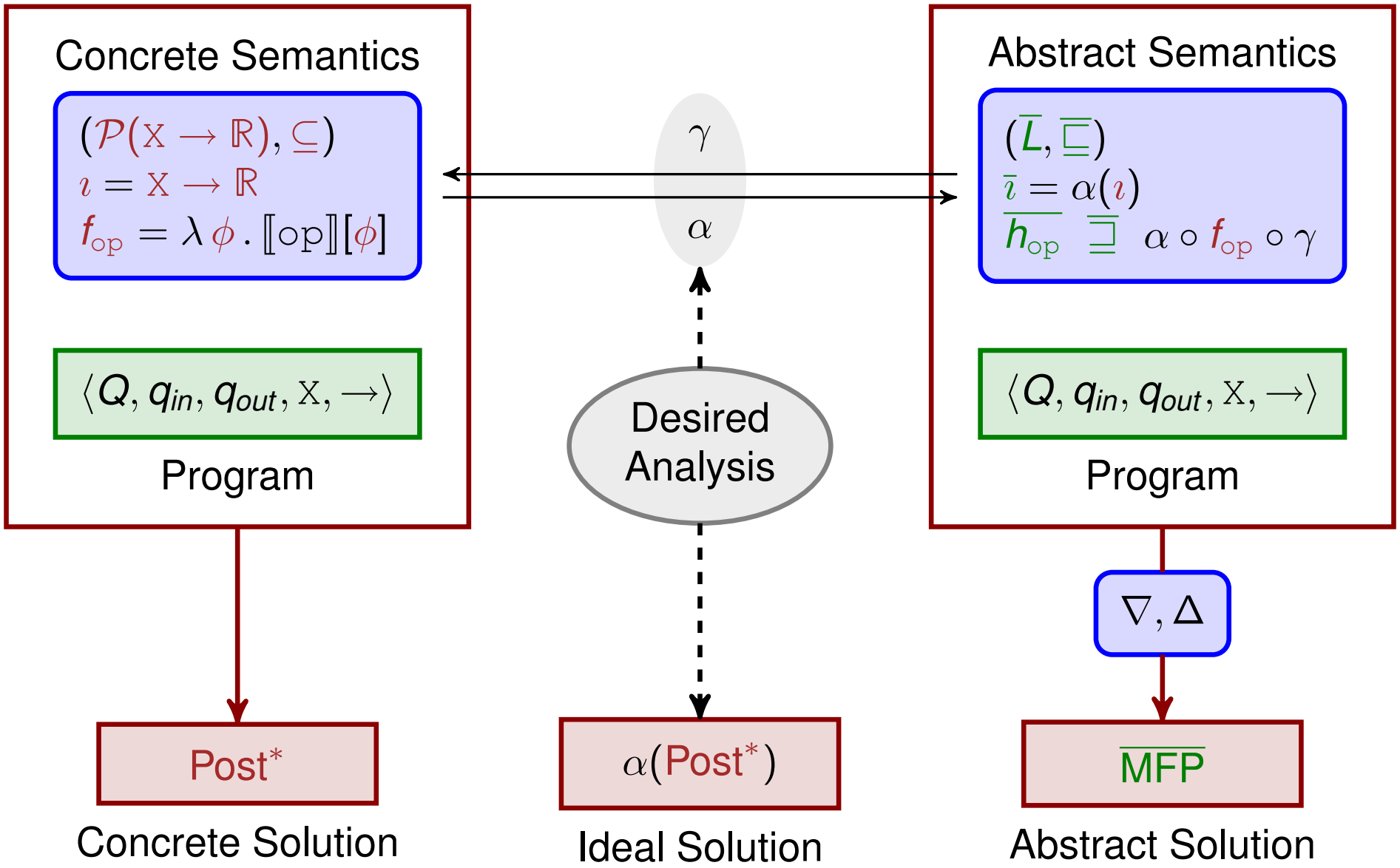
Abstract Interpretation-Based Data Flow Analysis



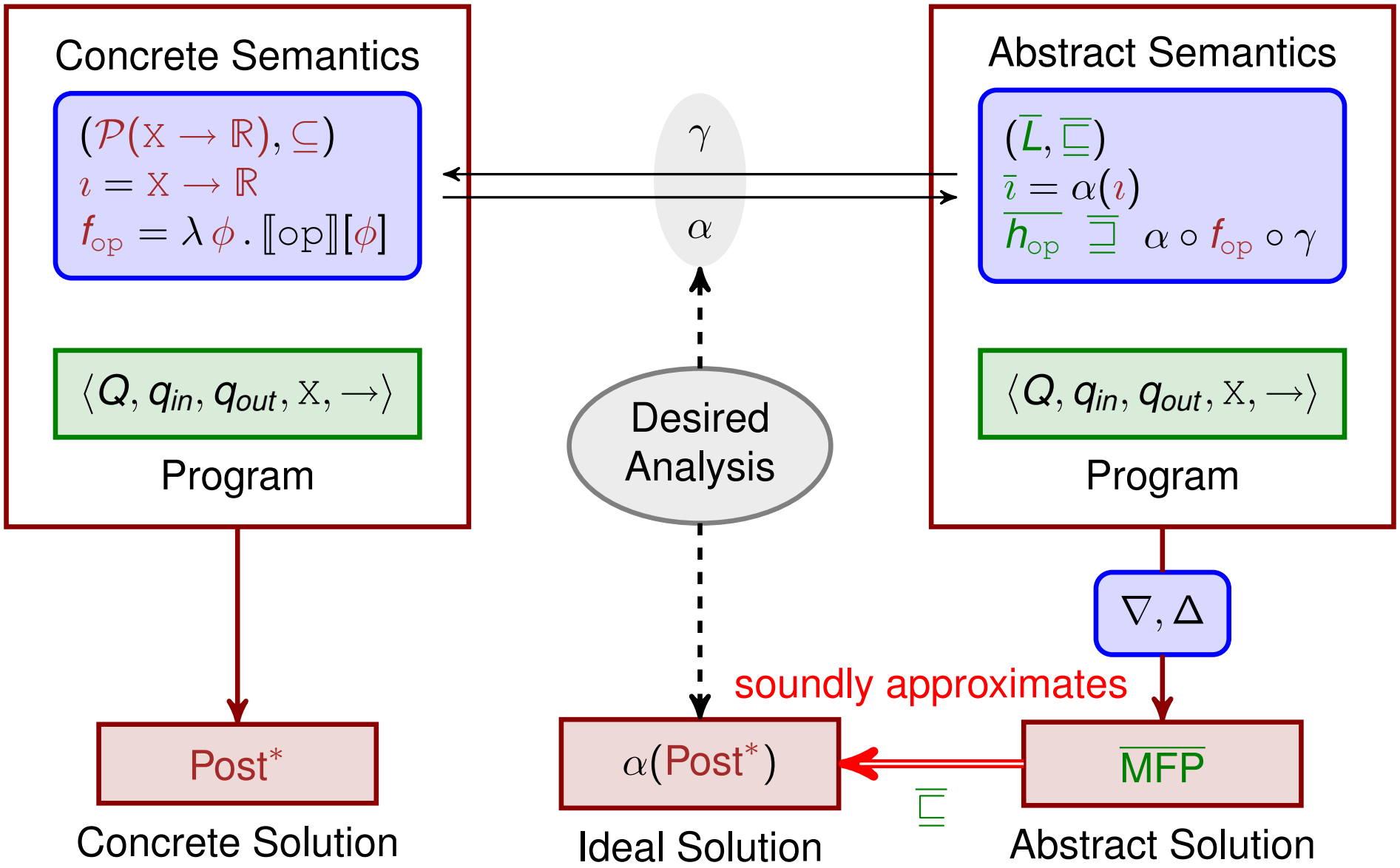
Abstract Interpretation-Based Data Flow Analysis



Abstract Interpretation-Based Data Flow Analysis



Abstract Interpretation-Based Data Flow Analysis



Outline — Software Verification by Static Analysis

Applications of Classical Data Flow Analysis

Very Common Sources of Bugs

Uninitialized variables

Dead code

...

Can be detected by gen / kill data flow analyses

Data flow analysis in every compiler!

Classical Data Flow Analysis in Compilers

```
1 class Foo1 {
2     static void foo1(int x) {
3         int i, y;
4
5         for (i = 0 ; i < x ; i++) {
6             y = y + (i * i);
7         }
8     }
9 }
```

```
$ javac Foo1.java
Foo1.java:6: variable y might not have been initialized
            y = y + (i * i);
                ^
1 error
```

Abstract Interpretation-Based Invariant Generation

Software Verification: Is Post* disjoint from *Bad*?

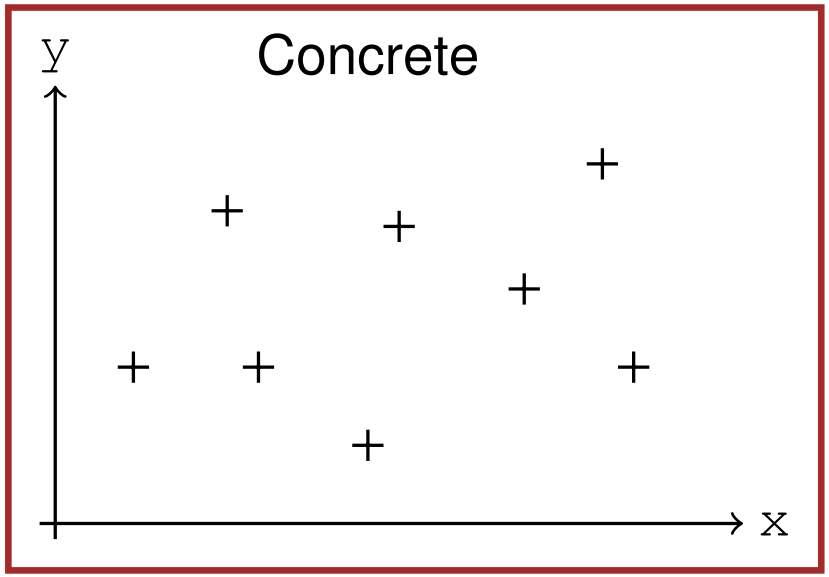
1. Compute an invariant $Inv \supseteq \text{Post}^*$
2. If *Inv* is disjoint from *Bad* then return "program safe"

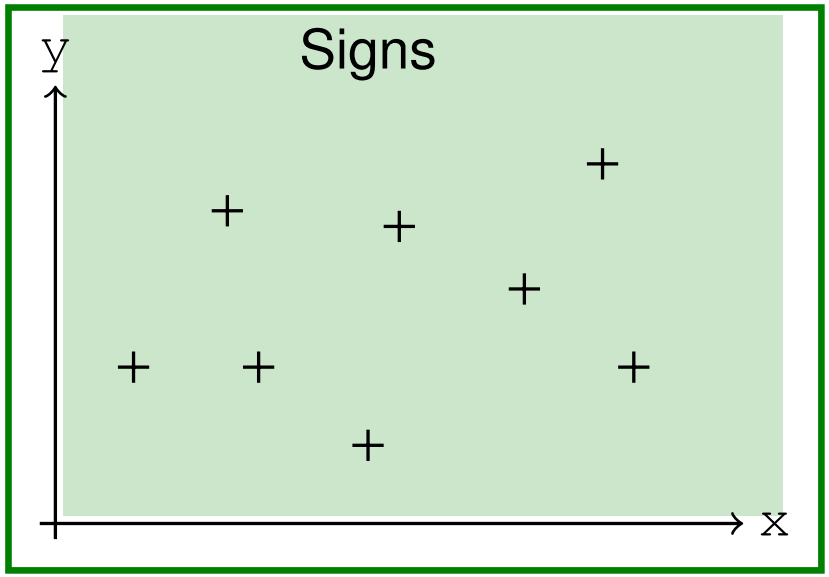
The MFP solution obtained by abstract interpretation is an invariant ☺

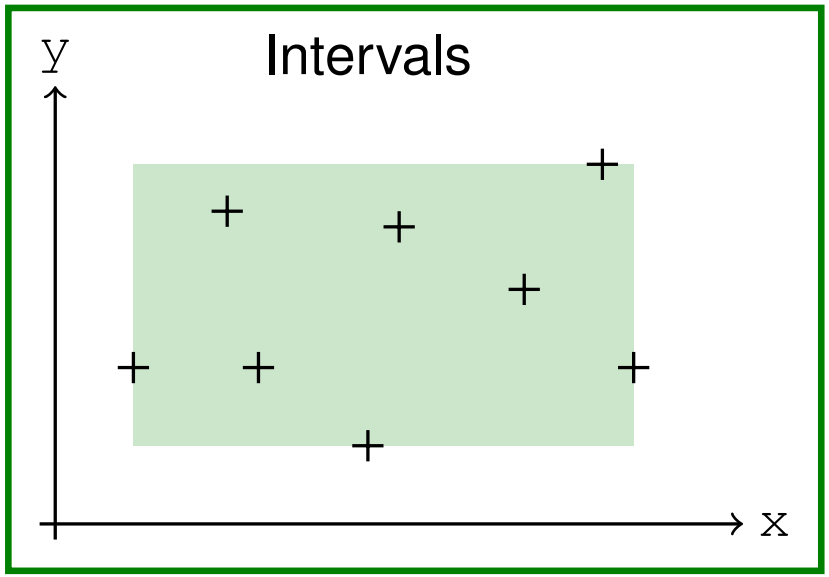
Tradeoff between computational cost and precision

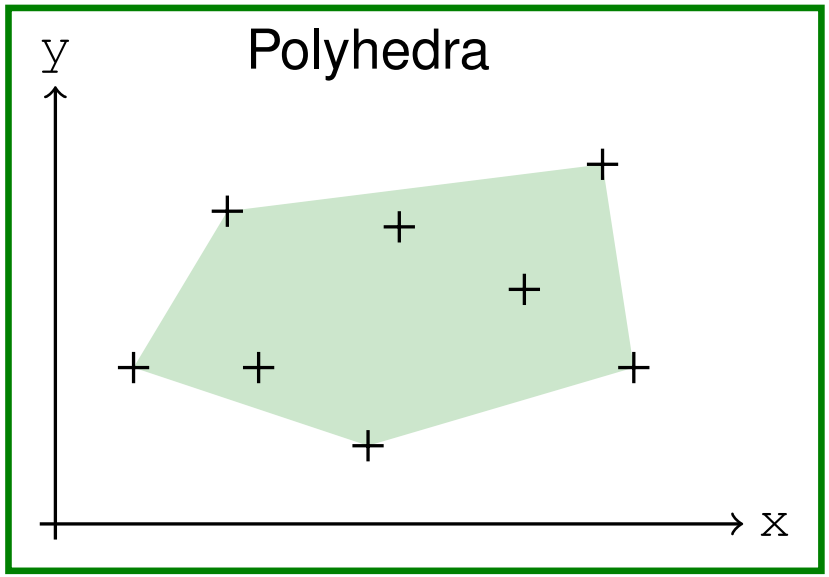
- Numerical abstract domains
- Approximate transfer mappings
- Widenings and narrowings

Some Numerical Abstract Domains









Some Commercial Static Analysis Tools

PolySpace™ Embedded Software Verification, The MathWorks™

> « *PolySpace™ products verify C, C++, and Ada code for embedded applications by detecting run-time errors before code is compiled and executed.* »

Coverity Prevent™ Static Analysis for C/C++, for C#, and for Java

> « *The foundation of Coverity's leading automated approach to identifying and resolving the most critical defects in C, C++, C# and Java source code.* »

Coverity periodically runs Coverity Prevent™ on open source projects

- Program Analyzer Generator (Saarland Univ. & AbsInt GmbH)
- Purify (IBM), Klocwork, ...

Some Academic Static Analysis Tools

ASTRÉE Static Analyzer — P. Cousot, R. Cousot, . . .

- Abstract interpretation-based analysis of C
- Application to safety critical embedded software

Verification of the primary flight control software of the Airbus A340 and A380 fly-by-wire systems

APRON Numerical Abstract Domain Library — B. Jeannet, . . .

- Common interface to various abstract domains
  - includes intervals, polyhedra, octagons, linear congruences
- Online demonstration of the Interproc analyzer
- Open-source, released under the GNU LGPL

Outline — Software Verification by Static Analysis

Main Limitation of Data Flow Analysis

False Positives and False Negatives

Example of False Positive

Variable detected as not initialized, but in fact it is initialized for all runs of the program.

Example of False Negative

No code detected as dead, but in fact some program point cannot be reached by any run.

Main Limitation of Data Flow Analysis

False Positives and False Negatives

Example of False Positive

Variable detected as not initialized, but in fact it is initialized for all runs of the program.

Example of False Negative

No code detected as dead, but in fact some program point cannot be reached by any run.

Main Limitation of Data Flow Analysis

False Positives and False Negatives

Example of False Positive

Variable detected as not initialized, but in fact it is initialized for all runs of the program.

Example of False Negative

No code detected as dead, but in fact some program point cannot be reached by any run.

Classical Data Flow Analysis in Compilers

```
1 class Foo2 {
2     static int foo2(int x) {
3         int y;
4
5         if (x == 0) { y = 5; }
6         else        { y = 2; }
7         return y;
8     }
9 }
```

```
$ javac Foo2.java
$
```

Classical Data Flow Analysis in Compilers

```
1 class Foo3 {
2     static int foo3(int x) {
3         int y;
4
5         if (x == 0) { y = 5; }
6         if (x != 0) { y = 2; }
7         return y;
8     }
9 }
```

```
$ javac Foo3.java
Foo3.java:7: variable y might not have been initialized
        return y;
               ^
1 error
```

Main Limitation of Abstract Interpretation

False Positives

Software Verification: Is Post* disjoint from *Bad*?

1. Compute an invariant $Inv \supseteq \text{Post}^*$
2. If *Inv* is disjoint from *Bad* then return “program safe”
3. If *Inv* intersects *Bad* then return “alarm”

In practice, there might be too many false alarms. . .

Main Limitation of Abstract Interpretation

False Positives

Software Verification: Is Post* disjoint from *Bad*?

1. Compute an invariant *Inv* $\supseteq$ Post*
2. If *Inv* is disjoint from *Bad* then return “program safe”
3. If *Inv* intersects *Bad* then return “alarm”

In practice, there might be too many false alarms. . .

Main Limitation of Abstract Interpretation

False Positives

Software Verification: Is Post* disjoint from *Bad*?

1. Compute an invariant $Inv \supseteq \text{Post}^*$
2. If *Inv* is disjoint from *Bad* then return "program safe"
3. If *Inv* intersects *Bad* then return "alarm"

In practice, there might be too many false alarms...

What can we do about it?

Software Verification by Static Analysis: Workflow

While the analysis returns alarms

1. Inspect alarms to determine whether they are spurious or not
2. If alarms are spurious then refine the analysis to gain precision

Why not automate this process?

Trade termination guarantee with fully automatic model-checking

- ☹ Not acceptable for compile-time static analyses
- ☺ Acceptable for verification

Topic of next part. . .

What can we do about it?

Software Verification by Static Analysis: Workflow

While the analysis returns alarms

1. Inspect alarms to determine whether they are spurious or not
2. If alarms are spurious then refine the analysis to gain precision

Why not automate this process?

Trade termination guarantee with fully automatic model-checking

- ☹ Not acceptable for compile-time static analyses
- ☺ Acceptable for verification

Topic of next part...

Outline — Software Verification by Static Analysis

Some References

F. Nielson, H. R. Nielson, and C. Hankin.
Principles of Program Analysis.
Springer, 1999.

P. Cousot and R. Cousot.
Systematic design of program analysis frameworks.
In *Proc. 6th ACM Symp. Principles of Programming Languages, San Antonio, TX, USA*, pages 269–282. ACM Press, 1979.

- The ASTRÉE Static Analyzer.
  `http://www.astree.ens.fr/`
- The APRON Library for Numerical Abstract Domains.
  `http://apron.cri.ensmp.fr/library/`
- Coverity's Scan.
  `http://scan.coverity.com/`

Part VI

Abstract Model Refinement

Outline — Abstract Model Refinement

Outline — Abstract Model Refinement

Software Verification by Static Analysis (Repetition)

Software Verification: Is Post* disjoint from *Bad*?

1. Compute an invariant $Inv \supseteq \text{Post}^*$
2. If *Inv* is disjoint from *Bad* then return "program safe"
3. If *Inv* intersects *Bad* then return "alarm"

Alarms must be inspected manually ☹

If an alarm is a real bug, then the analysis is useful ☺

Otherwise...

An improved analysis must be designed to eliminate alarms

Software Verification by Static Analysis (Repetition)

Software Verification: Is Post* disjoint from *Bad*?

1. Compute an invariant *Inv* $\supseteq$ Post*
2. If *Inv* is disjoint from *Bad* then return “program safe”
3. If *Inv* intersects *Bad* then return “alarm”

Alarms must be inspected manually ☹

If an alarm is a real bug, then the analysis is useful ☺

Otherwise. . .

An improved analysis must be designed to eliminate alarms

Software Verification by Static Analysis (Repetition)

Software Verification: Is Post* disjoint from *Bad*?

1. Compute an invariant $Inv \supseteq \text{Post}^*$
2. If *Inv* is disjoint from *Bad* then return "program safe"
3. If *Inv* intersects *Bad* then return "alarm"

Alarms must be inspected manually ☹

If an alarm is a real bug, then the analysis is useful ☺

Otherwise...

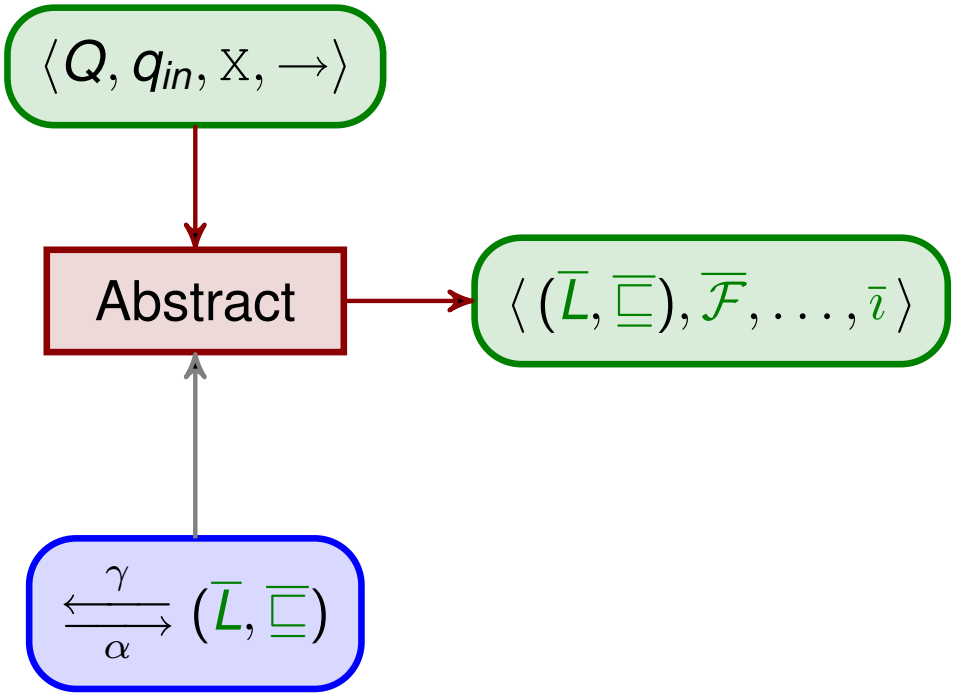
An improved analysis must be designed to eliminate alarms

Software Verification by Static Analysis: Workflow

$\langle Q, q_{in}, \mathrm{X}, \rightarrow \rangle$

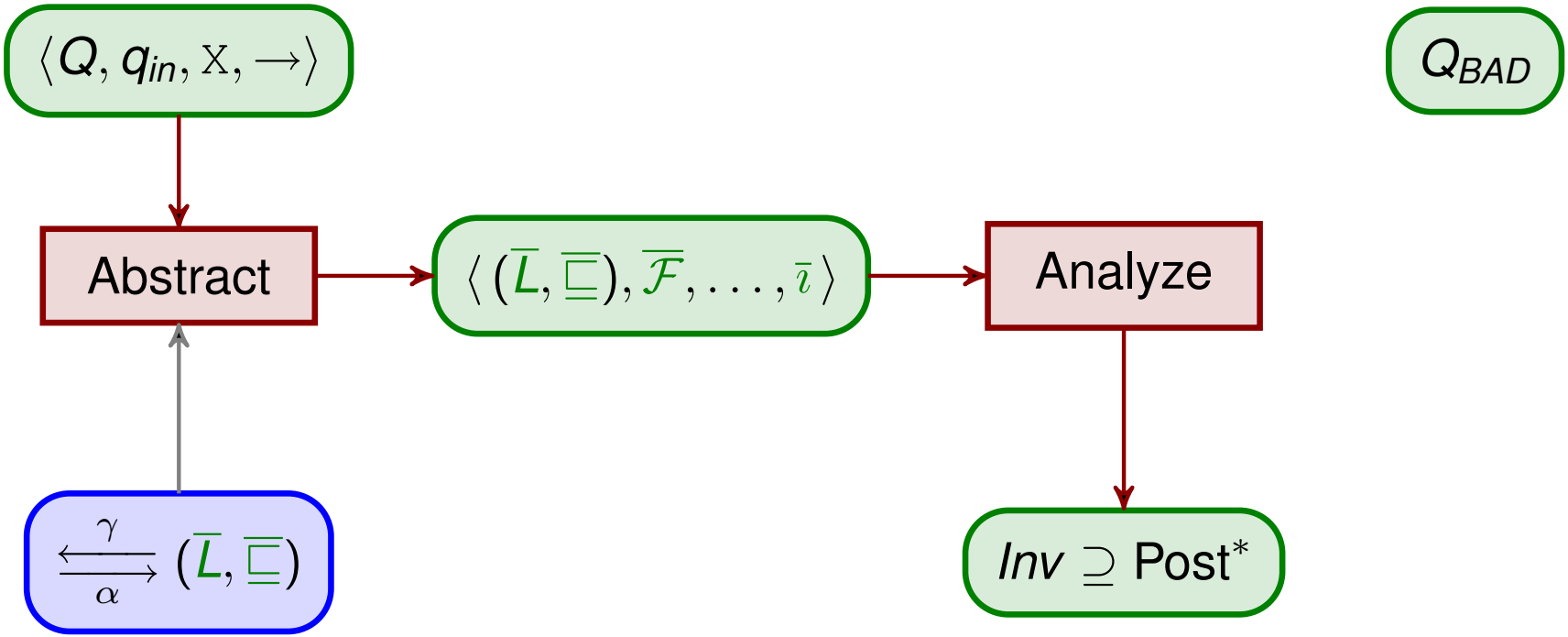
Q_{BAD}

Software Verification by Static Analysis: Workflow

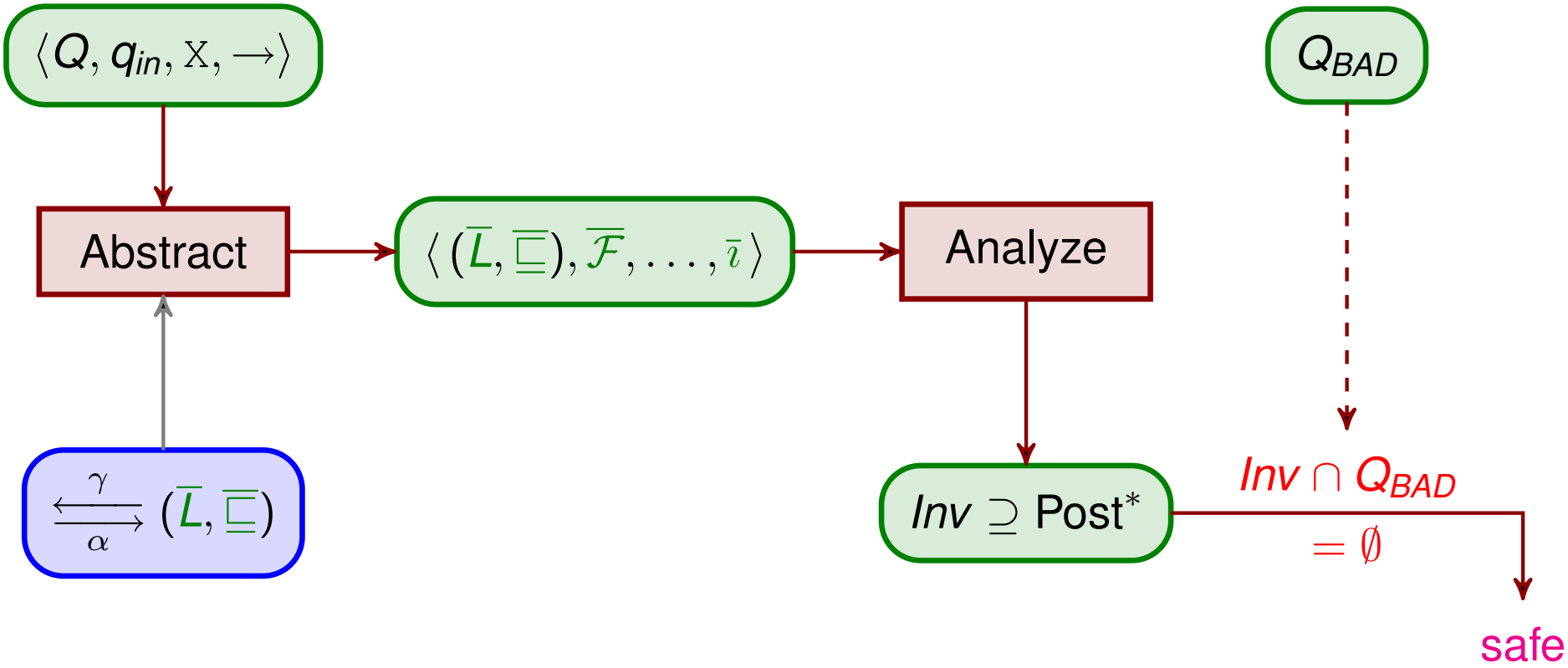


Q_{BAD}

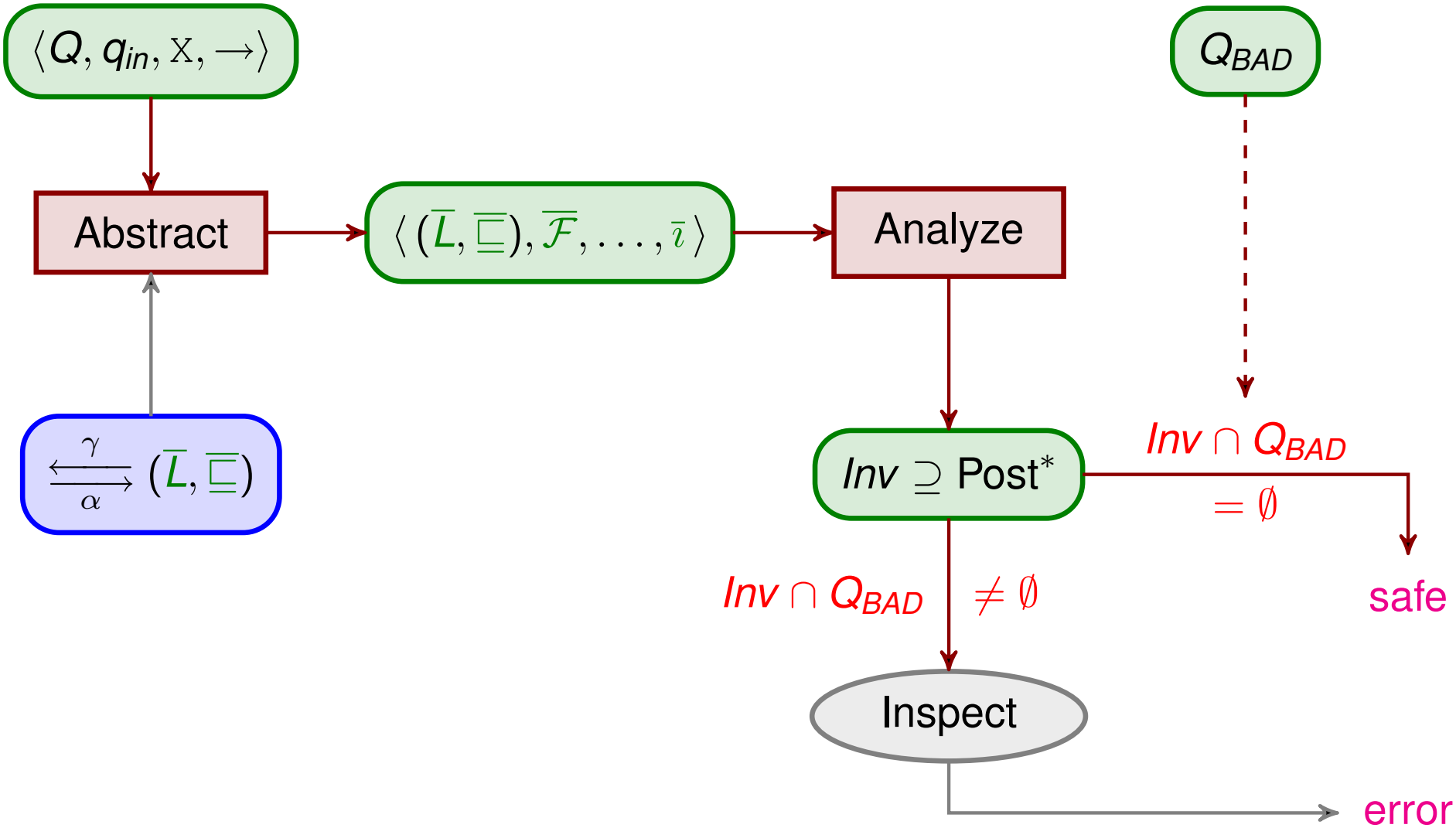
Software Verification by Static Analysis: Workflow



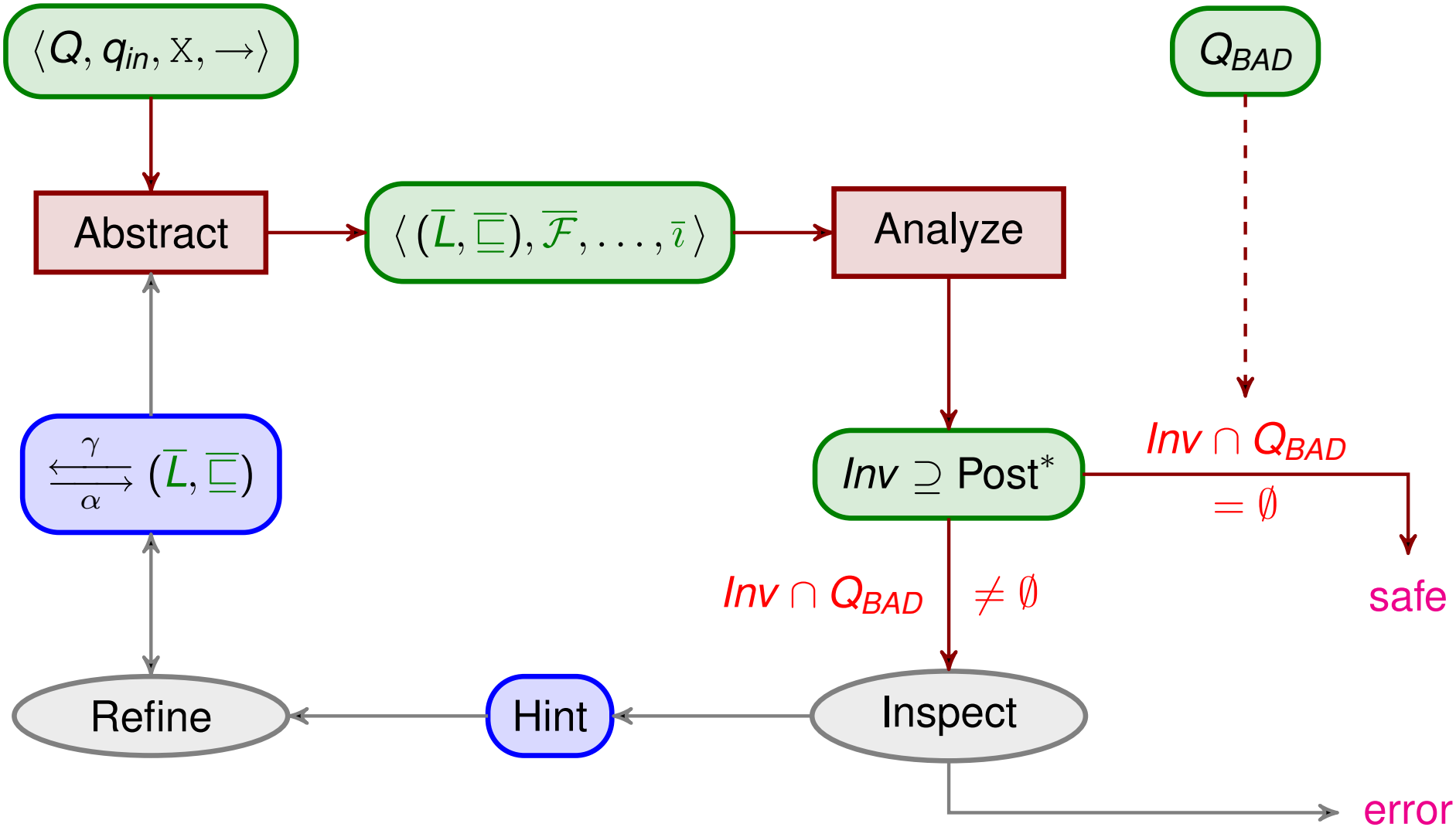
Software Verification by Static Analysis: Workflow



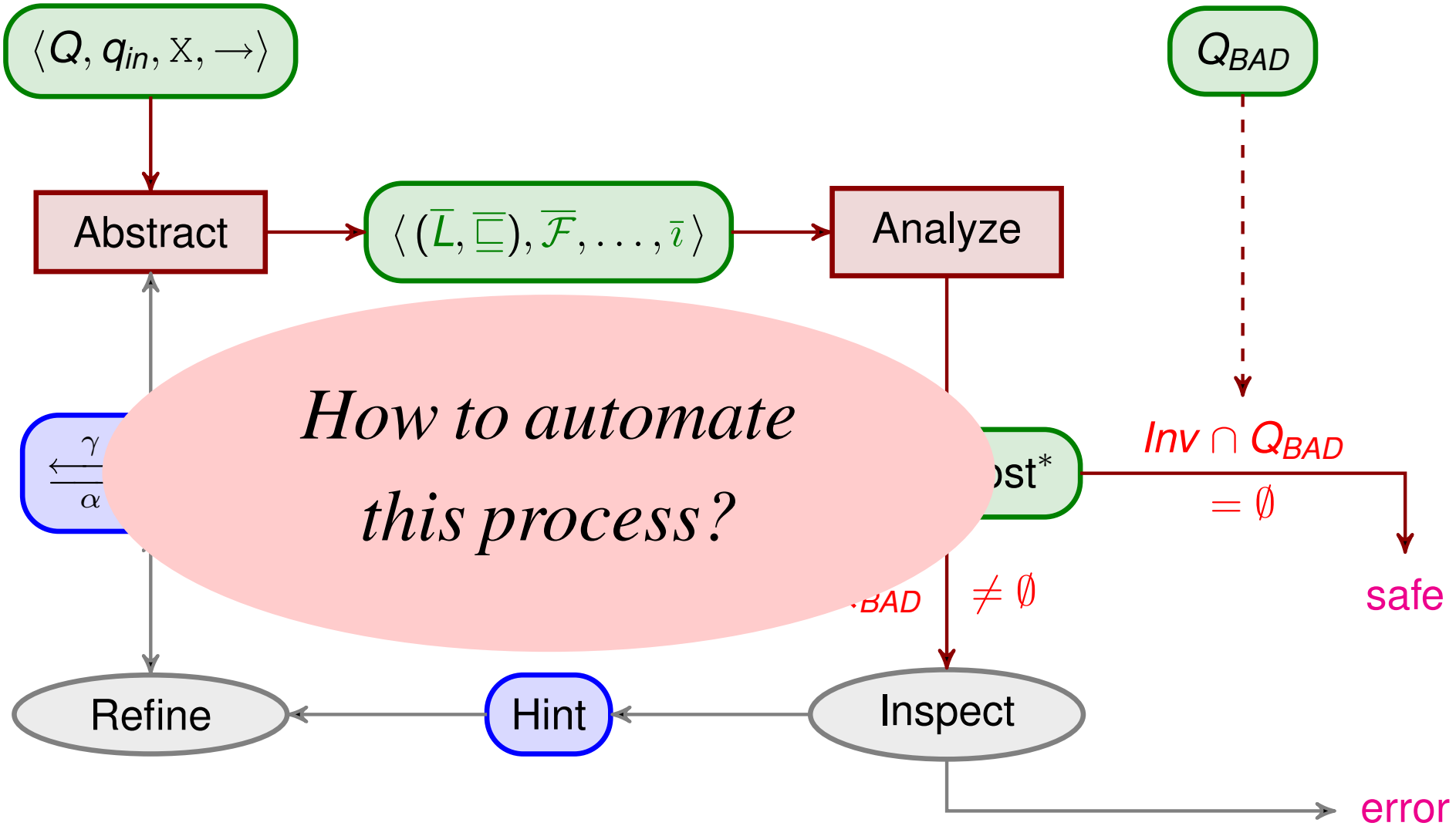
Software Verification by Static Analysis: Workflow



Software Verification by Static Analysis: Workflow



Software Verification by Static Analysis: Workflow



Inspection of Alarms: Not a Simple Task!

Objective

Given an abstract invariant $\overline{Inv}$ whose concretization $\gamma(\overline{Inv})$ intersects $Bad = Q_{BAD} \times (\mathbb{X} \to \mathbb{R})$, determine whether Post* intersects Bad.

All configurations in $\gamma(\overline{Inv}) \cap Bad$ are potentially reachable...

How are these configurations potentially reached?

It would be nice to have an "abstract run" of the form:

$$(q_{in}, \overline{\psi_0}) \xrightarrow{\mathrm{op}_0} \cdots \xrightarrow{\mathrm{op}_{k-1}} (q_k, \overline{\psi_k}) \quad \text{with} \quad \begin{cases} q_k \in Q_{BAD} \\ \gamma(\overline{\psi_k}) \cap Bad \neq \emptyset \end{cases}$$

Checking whether this abstract run is spurious reduces to checking emptiness of the relation: $(\llbracket \mathrm{op}_k \rrbracket \circ \cdots \circ \llbracket \mathrm{op}_0 \rrbracket)$.

Inspection of Alarms: Not a Simple Task!

Objective

Given an abstract invariant $\overline{Inv}$ whose concretization $\gamma(\overline{Inv})$ intersects $Bad = Q_{BAD} \times (\mathbb{X} \to \mathbb{R})$, determine whether Post^* intersects Bad.

All configurations in $\gamma(\overline{Inv}) \cap Bad$ are potentially reachable...

How are these configurations potentially reached?

It would be nice to have an "abstract run" of the form:

$$(q_{in}, \overline{\psi_0}) \xrightarrow{\circ p_0} \cdots \xrightarrow{\circ p_{k-1}} (q_k, \overline{\psi_k}) \quad \text{with} \quad \begin{cases} q_k \in Q_{BAD} \\ \gamma(\overline{\psi_k}) \cap Bad \neq \emptyset \end{cases}$$

Checking whether this abstract run is spurious reduces to checking emptiness of the relation: $(\llbracket \circ p_k \rrbracket \circ \cdots \circ \llbracket \circ p_0 \rrbracket)$.

Inspection of Alarms: Not a Simple Task!

Objective

Given an abstract invariant $\overline{Inv}$ whose concretization $\gamma(\overline{Inv})$ intersects $Bad = Q_{BAD} \times (\mathbb{X} \to \mathbb{R})$, determine whether Post^* intersects Bad.

All configurations in $\gamma(\overline{Inv}) \cap Bad$ are potentially reachable...

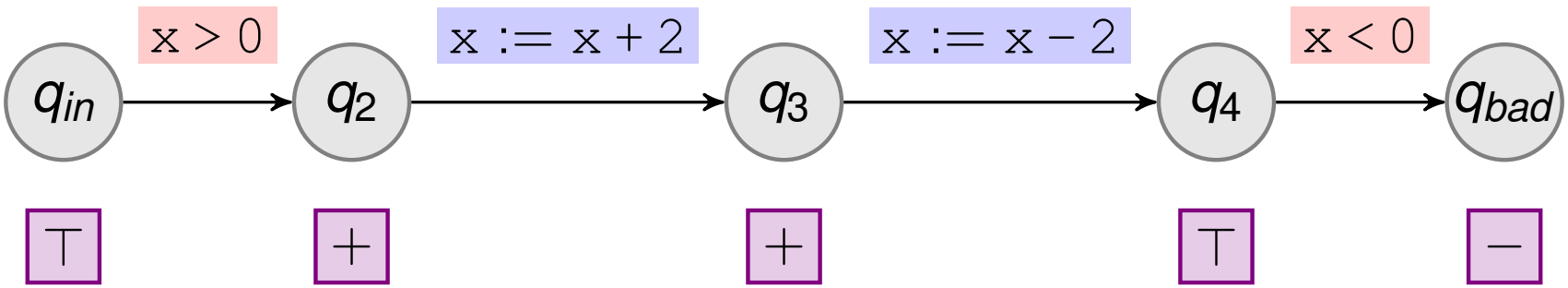
How are these configurations potentially reached?

It would be nice to have an "abstract run" of the form:

$$(q_{in}, \overline{\psi_0}) \xrightarrow{\mathrm{op}_0} \cdots \xrightarrow{\mathrm{op}_{k-1}} (q_k, \overline{\psi_k}) \quad \text{with} \quad \begin{cases} q_k \in Q_{BAD} \\ \gamma(\overline{\psi_k}) \cap Bad \neq \emptyset \end{cases}$$

Checking whether this abstract run is spurious reduces to checking emptiness of the relation: $(\llbracket \mathrm{op}_k \rrbracket \circ \cdots \circ \llbracket \mathrm{op}_0 \rrbracket)$.

Inspection of Abstract Runs: Example



Semantics of Operations (Repetition)

$$(v, v') \in [\![g]\!] \quad \textit{if} \quad v \models g \ \textit{and} \ v' = v$$

$$(v, v') \in [\![x := e]\!] \quad \textit{if} \quad \begin{cases} v'(x) = [\![e]\!]_v \\ v'(y) = v'(y) \quad \text{for all } y \neq x \end{cases}$$

$$[\![\mathrm{x} < 0]\!] \circ [\![\mathrm{x} := \mathrm{x} - 2]\!] \circ [\![\mathrm{x} := \mathrm{x} + 2]\!] \circ [\![\mathrm{x} > 0]\!] = \emptyset$$

$$\mathrm{x} > 0 \ \wedge \ \mathrm{x}' = \mathrm{x} + 2 \ \wedge \ \mathrm{x}'' = \mathrm{x}' - 2 \ \wedge \ \mathrm{x}'' < 0 \quad \textit{unsastisfiable}$$

Refinement of Abstract Domains: Not a Simple Task!

Objective

Given an abstract invariant $\overline{Inv}$ and a subset $U \subseteq \gamma(\overline{Inv}) \setminus \text{Post}^*$, design a new abstract domain where the resulting $\overline{Inv}$ is disjoint from U.

U would be a set of configurations identified as false alarms.

Quite challenging!

More Reasonable Objective

Given a spurious "abstract run", design a new abstract domain that eliminates this "abstract run".

Refinement of Abstract Domains: Not a Simple Task!

Objective

Given an abstract invariant $\overline{Inv}$ and a subset $U \subseteq \gamma(\overline{Inv}) \setminus \text{Post}^*$, design a new abstract domain where the resulting $\overline{Inv}$ is disjoint from U.

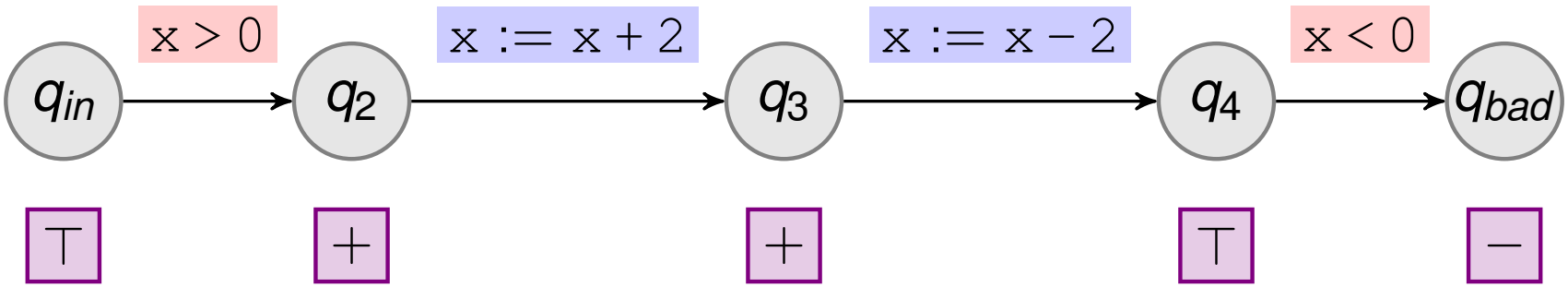
U would be a set of configurations identified as false alarms.

Quite challenging!

More Reasonable Objective

Given a spurious "abstract run", design a new abstract domain that eliminates this "abstract run".

Refinement Based on Abstract Runs: Example

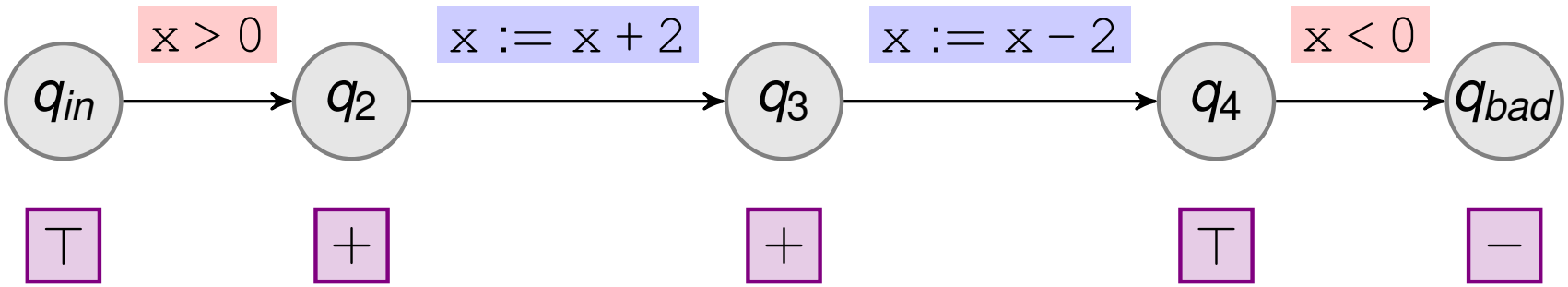


In q_3, the set of reachable valuations is:

$$[\![x := x + 2]\!] \circ [\![x > 0]\!][(X \to \mathbb{R})] \quad = \quad \{v \in X \to \mathbb{R} \mid v(x) > 2\}$$

We lack the "property" $x > 2$. Let us add it (as $2+$) to the *Sign* domain.

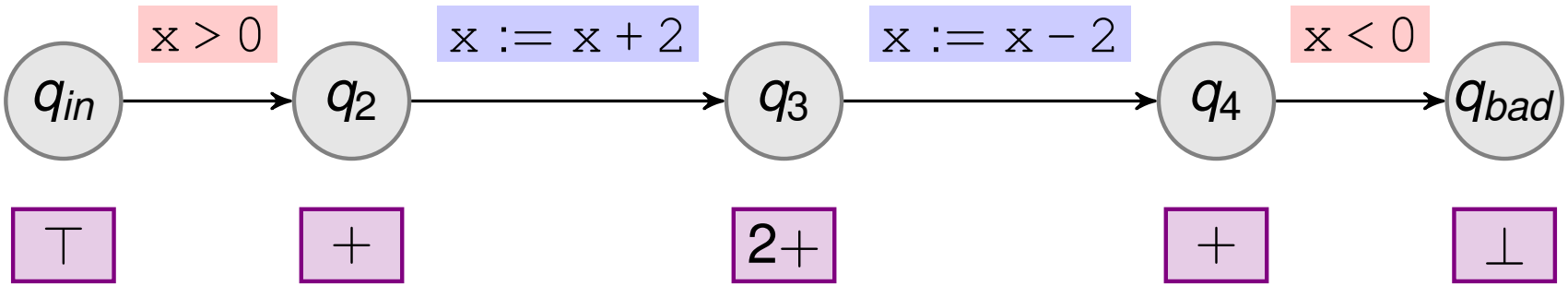
Refinement Based on Abstract Runs: Example



In q_3, the set of reachable valuations is:

$$[\![x := x + 2]\!] \circ [\![x > 0]\!][(X \to \mathbb{R})] \quad = \quad \{v \in X \to \mathbb{R} \mid v(x) > 2\}$$

We lack the "property" $x > 2$. Let us add it (as $2+$) to the *Sign* domain.



Hypothetical Workflow Based on Abstract Runs

Abstract “counterexample” runs are key to:

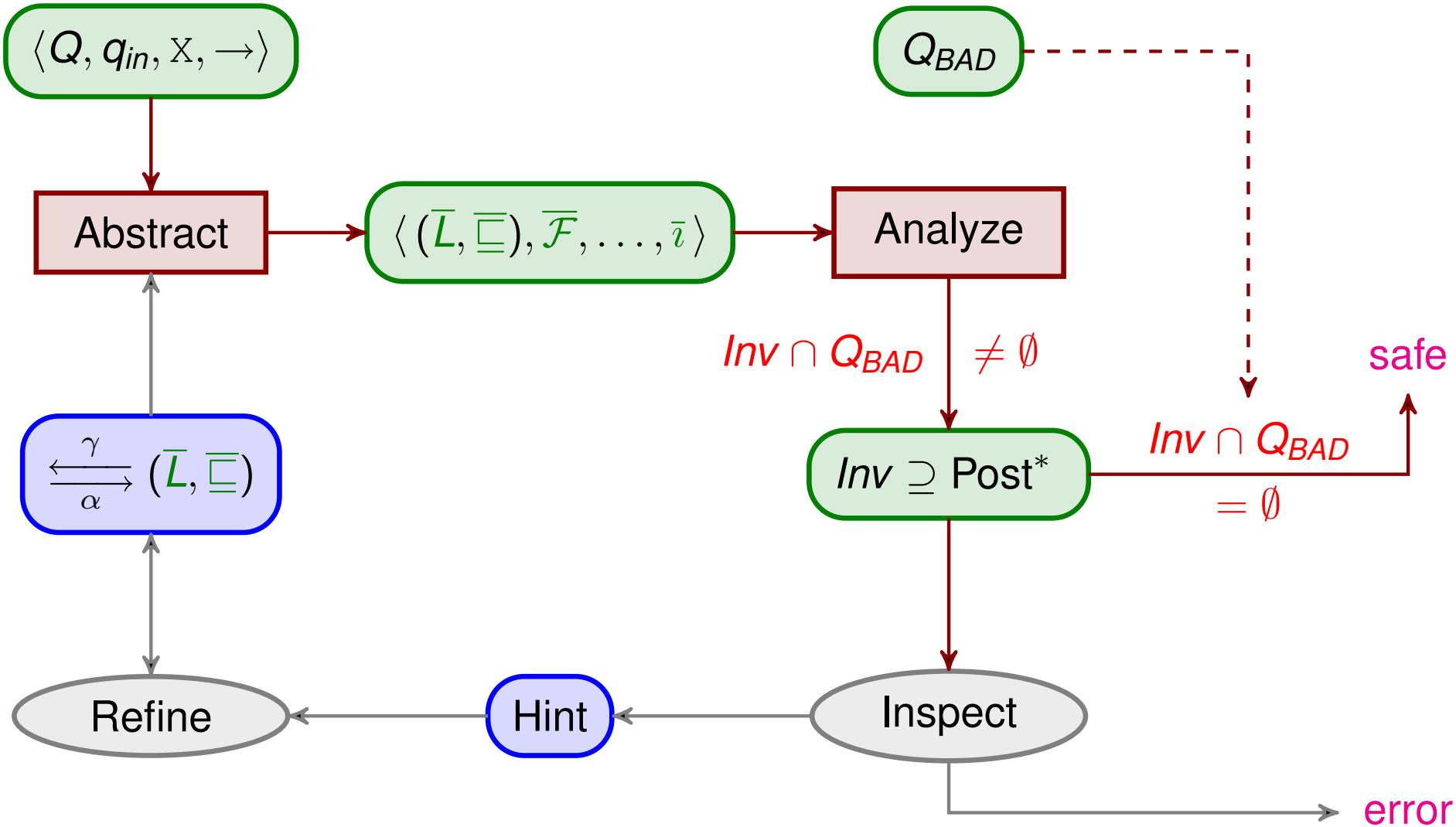
- inspection of alarms
- refinement of abstract domains

Enhanced Workflow Based on Abstract Runs

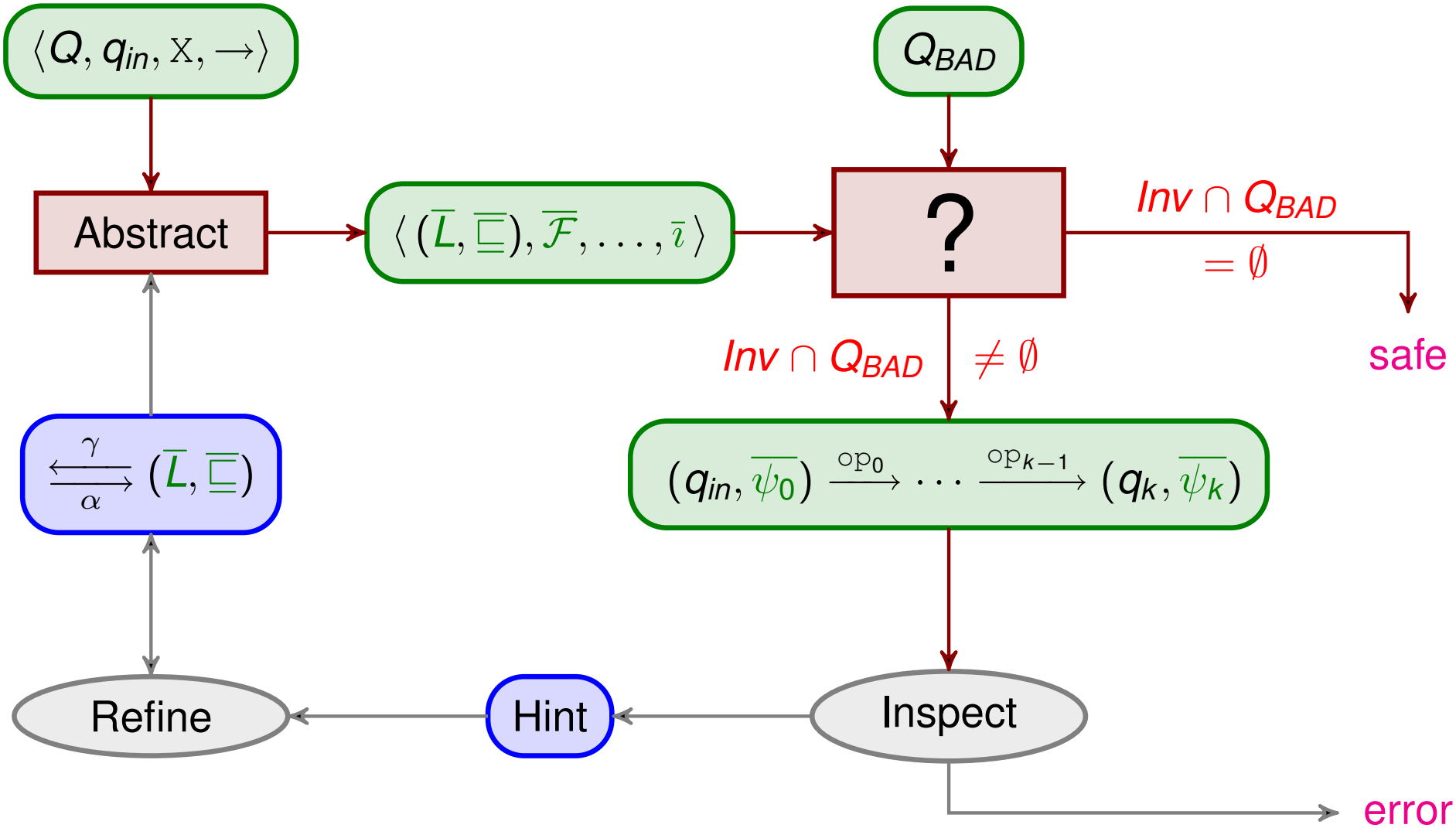
Imagine a hypothetical workflow where the analyzer returns:

- either “program safe” if it finds an invariant *Inv* disjoint from *Bad*
- or “alarm” with an abstract run as a potential counterexample

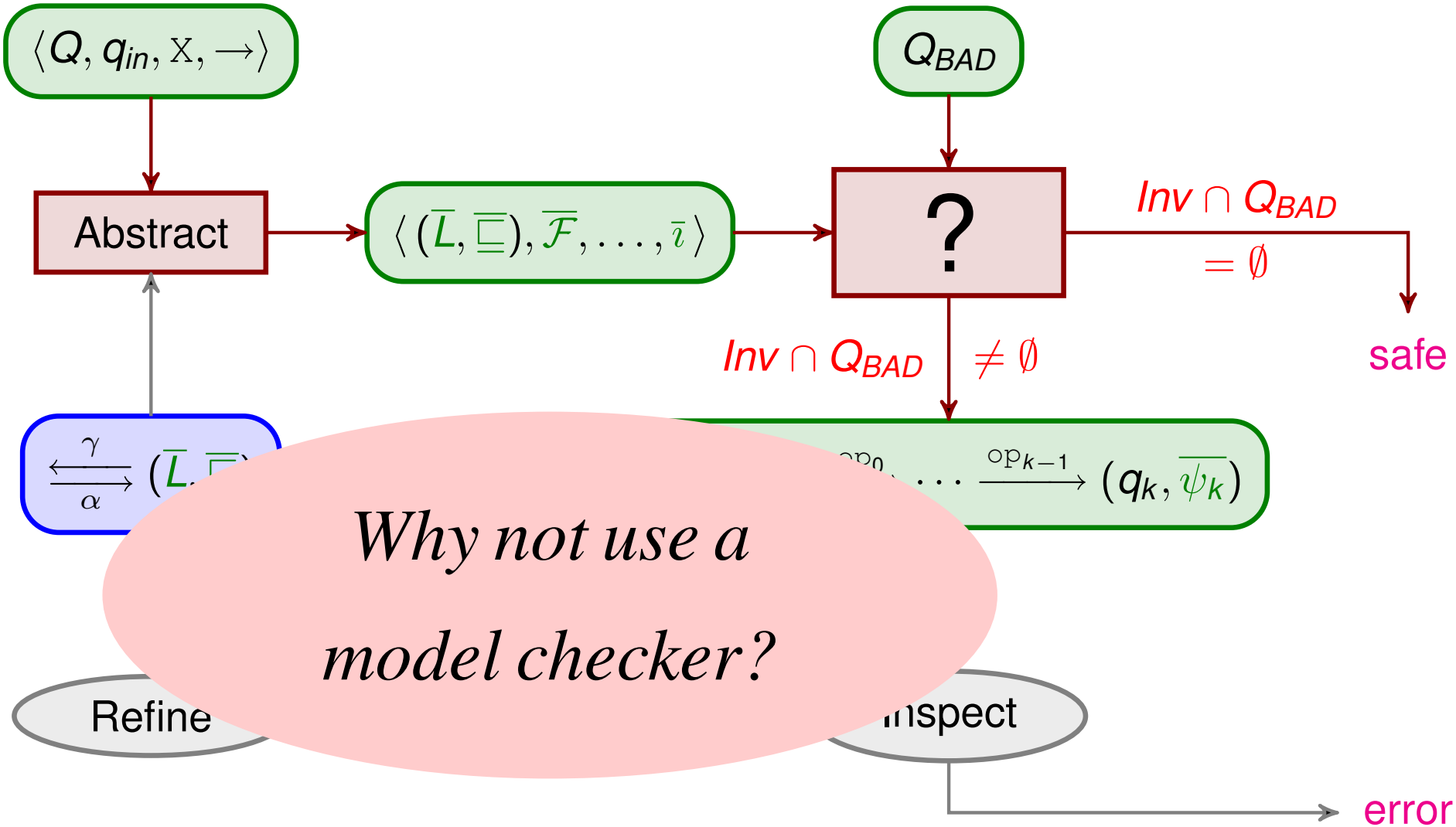
Hypothetical Workflow Based on Abstract Runs



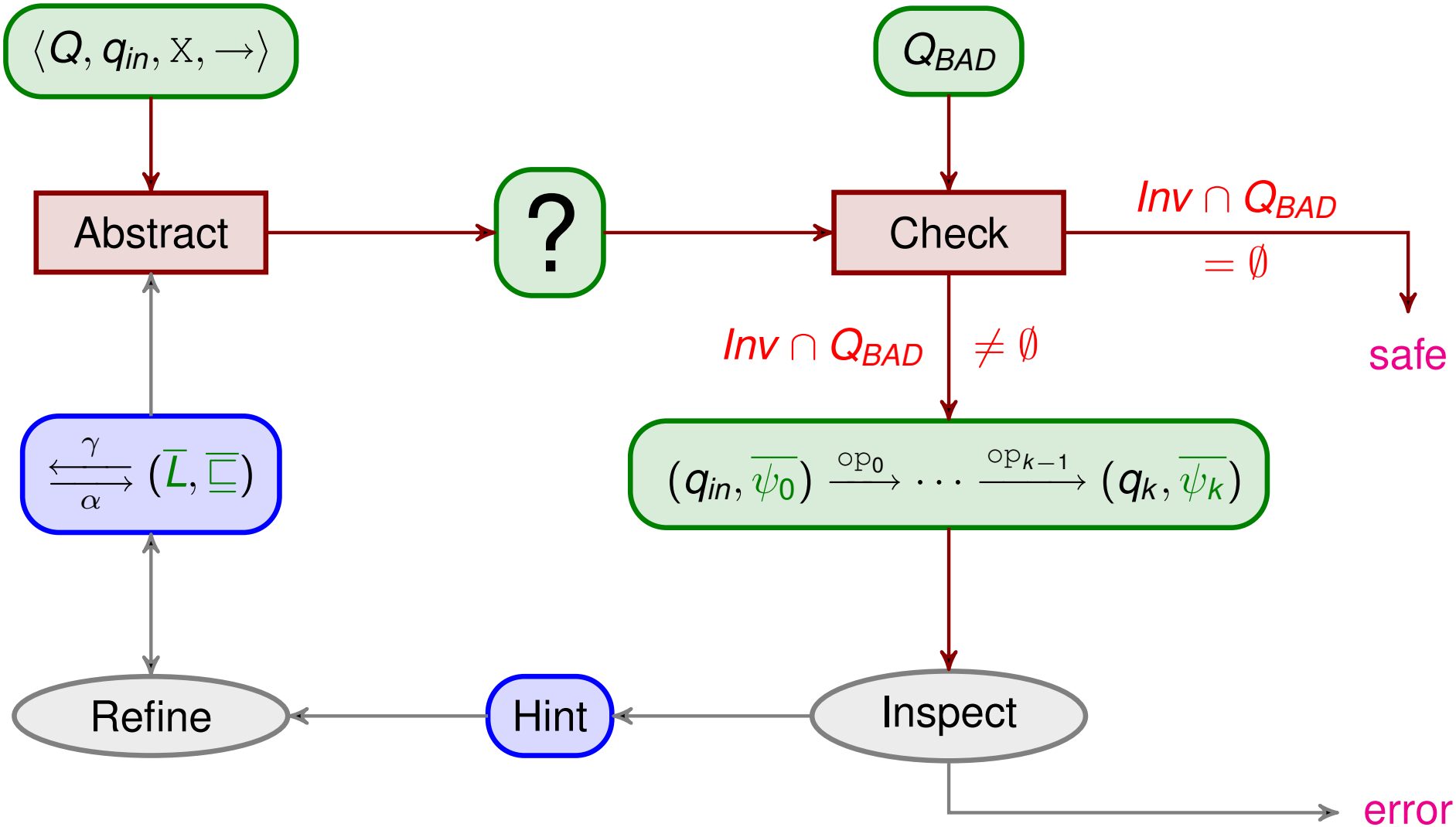
Hypothetical Workflow Based on Abstract Runs



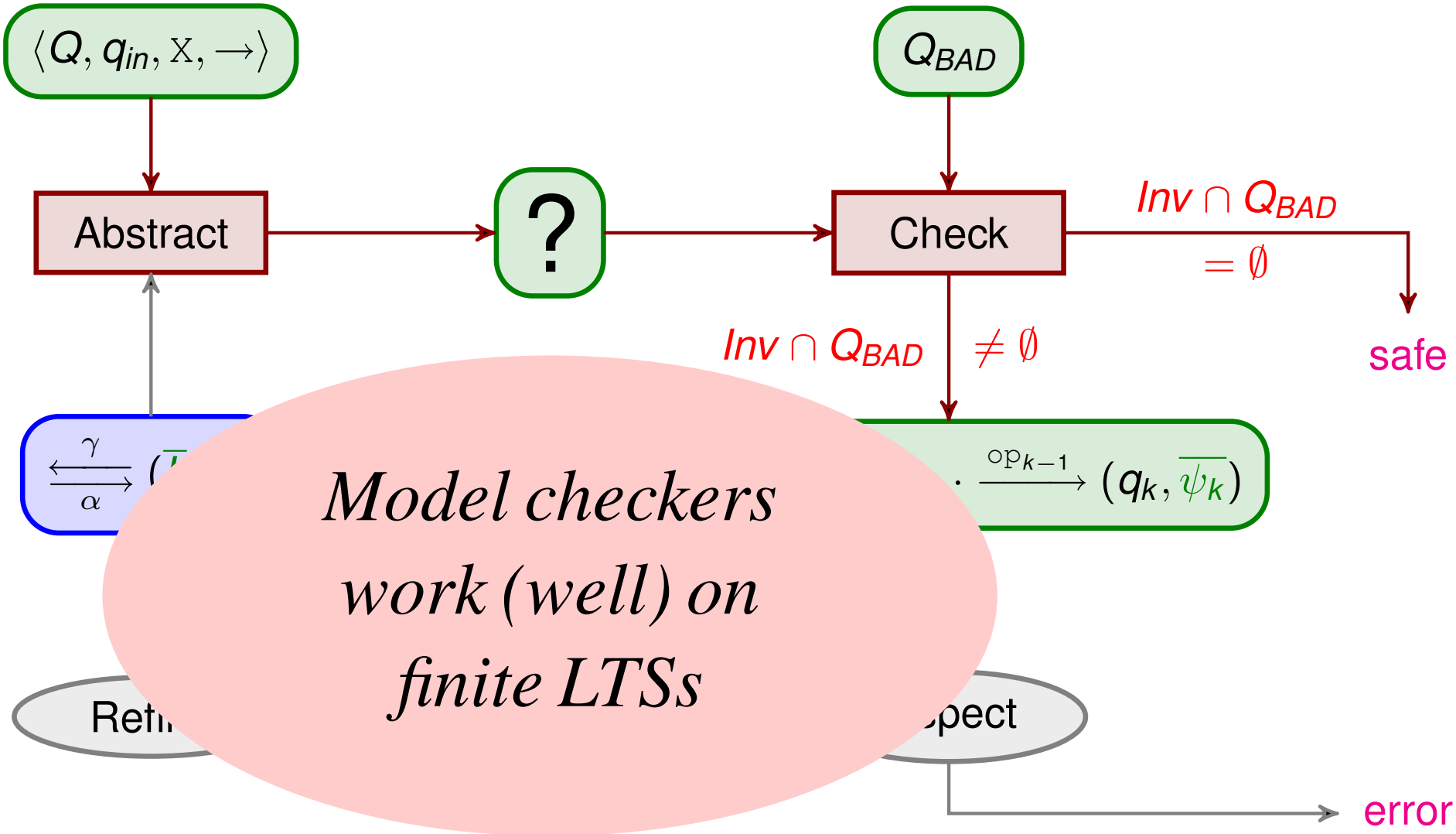
Hypothetical Workflow Based on Abstract Runs



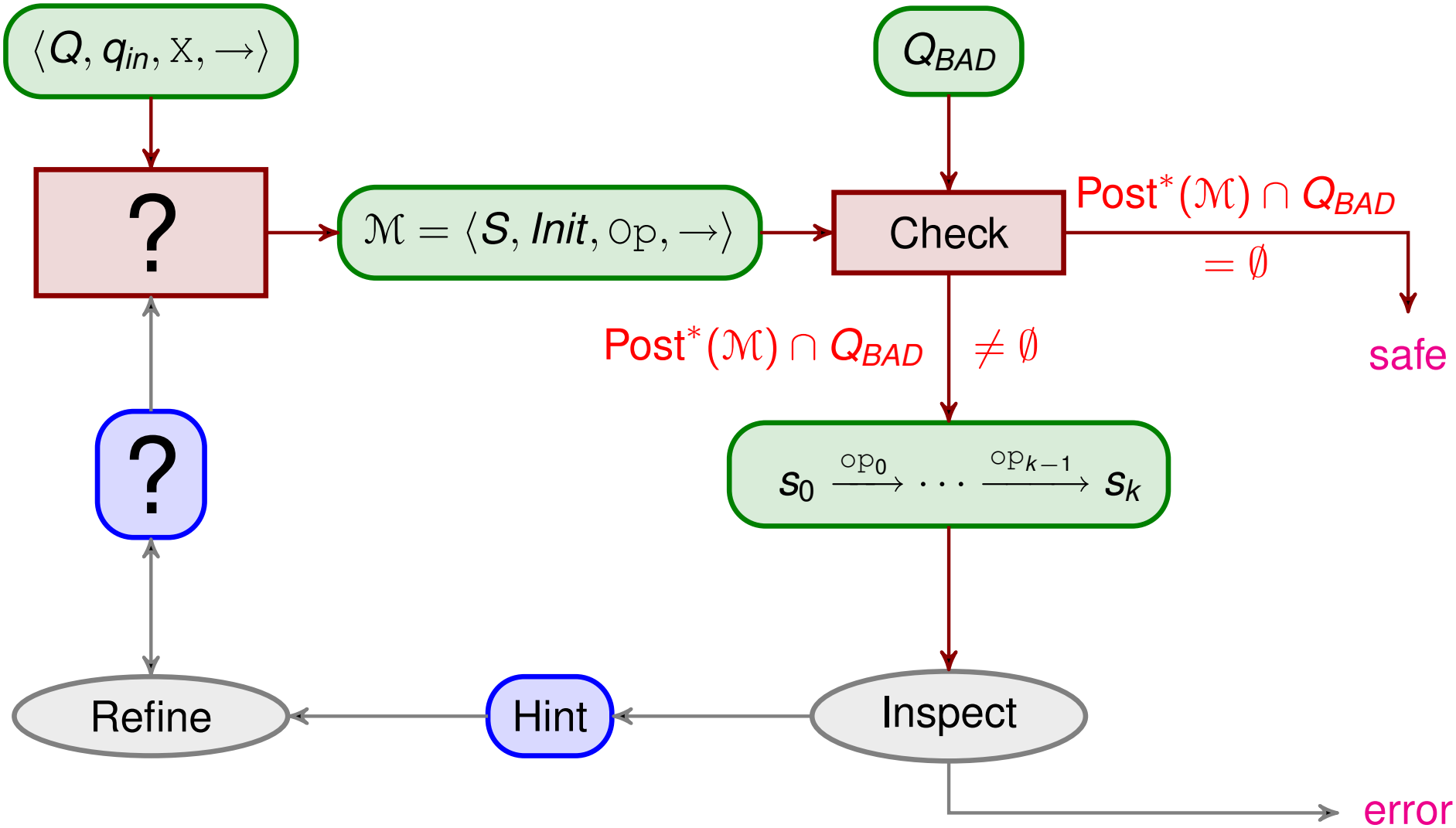
Hypothetical Workflow Based on Abstract Runs



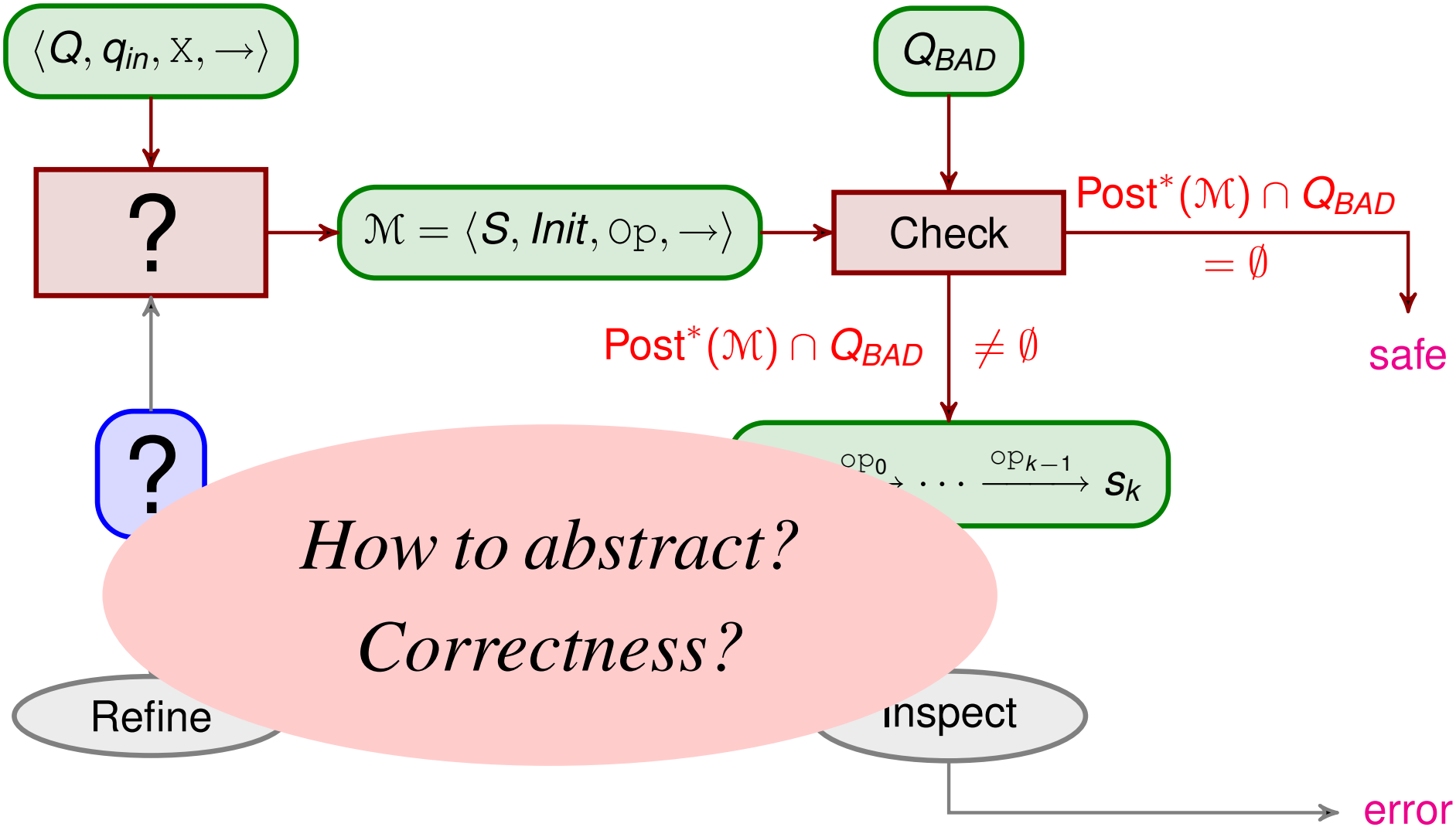
Hypothetical Workflow Based on Abstract Runs



Hypothetical Workflow Based on Abstract Runs



Hypothetical Workflow Based on Abstract Runs



Verification by Model-Checking Abstract Models

This hypothetical workflow... ... is not hypothetical at all!

Automatic Generation of Property-Preserving Abstractions

- First designed for large finite-state concurrent systems
- Inspired from abstract interpretation (use of Galois connections)
- Extended to (infinite-state) programs with theorem provers

Credits: Pioneers (1990's)

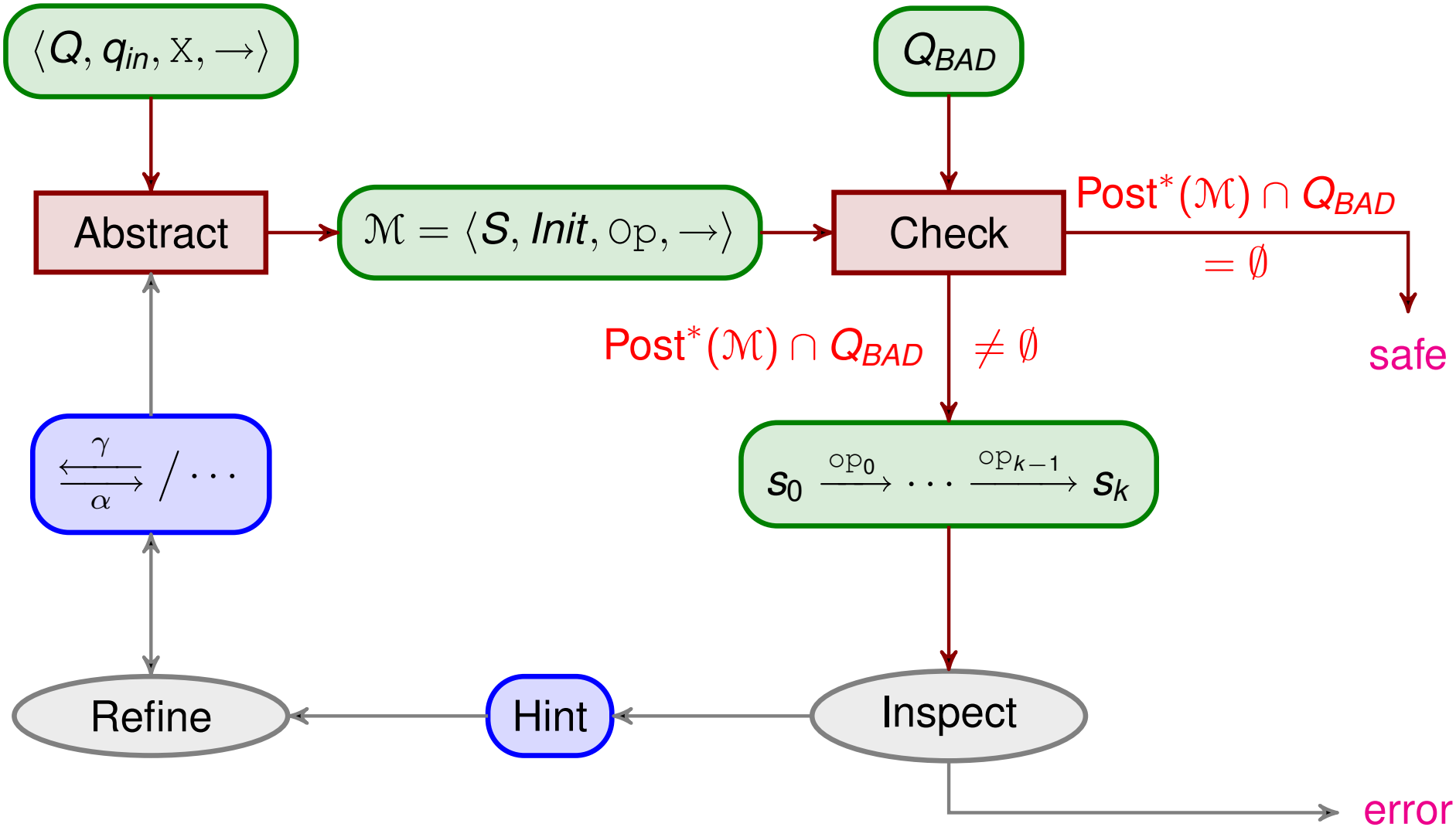
Joseph Sifakis & Claire Loiseaux

Dennis Dams & Rob Gerth & Orna Grumberg

Susanne Graf & Hassen Saïdi

...

Verification by Model-Checking Abstract Models



Automatic Inspection and Refinement: a Dream?

Goal

Automate the tasks Inspect and Refine

Counterexample Guided Refinement (2000)

- First designed for large finite-state systems (hardware)
- Extended to (infinite-state) programs with theorem provers
- Subject of active research

Credits: Pioneers (2000)

Edmund Clarke & Orna Grumberg

Thomas Ball & Sriram Rajamani

...

Summary and Outlook: Key Ingredients

Property-Preserving Abstraction

Conservatively extract finite-state models from programs

Model-Checking

Can use a readily available finite-state model checker ☺

Inspection of Abstract Counterexamples

Reduces to satisfiability checking (use of theorem provers)

Refinement Guided by Abstract Counterexamples

Driven by the safety property to check: precision where required

Monotonic: the model after refinement has less counterexamples

All these tasks can be automated ☺

Outline — Abstract Model Refinement

Objectives of the Basic Theory

Property-Preserving Abstraction

Conservatively extract finite-state models from programs

We focus on safety properties

Model

Labeled Kripke Structure

$=$

LTS + Bad

Notations

Concrete LKS: $\mathcal{M}^c$

Abstract LKS: $\mathcal{M}^a$

Theory Intentionally Limited (Only What We Need...)

- Notions of abstraction and refinement (simpler than $\overset{\gamma}{\underset{\alpha}{\leftrightarrows}}$ ☺)
- Theorem for preservation of safety

Labeled Kripke Structures for Safety

Definition

A labeled Kripke structure is a quintuple $\langle S, \mathit{Init}, \mathit{Bad}, \Sigma, \rightarrow \rangle$ where :

- S is a set of *states*
- $\mathit{Init} \subseteq S$ is a set of *initial states*
- $\mathit{Bad} \subseteq S$ is a set of *bad states*
- Σ is a finite set of *actions*
- $\rightarrow \subseteq S \times \Sigma \times S$ is a set of *transitions*

Simplified Definition!

Kripke structures are classically defined with a mapping from S to $\mathcal{P}(AP)$ where AP is a finite set of atomic propositions.

In our context $AP = \{\mathit{bad}\}$, hence it suffices to take $\mathit{Bad} \subseteq S$.

Labeled Kripke Structures for Safety

Definition

A labeled Kripke structure is a quintuple $\langle S, \mathit{Init}, \mathit{Bad}, \Sigma, \rightarrow \rangle$ where :

- S is a set of *states*
- $\mathit{Init} \subseteq S$ is a set of *initial states*
- $\mathit{Bad} \subseteq S$ is a set of *bad states*
- Σ is a finite set of *actions*
- $\rightarrow \subseteq S \times \Sigma \times S$ is a set of *transitions*

Simplified Definition!

Kripke structures are classically defined with a mapping from S to $\mathcal{P}(AP)$ where AP is a finite set of atomic propositions.

In our context $AP = \{\mathit{bad}\}$, hence it suffices to take $\mathit{Bad} \subseteq S$.

Lab. Kripke Structures Versus Lab. Transition Systems

Labeled Transition System

$$\langle C, \mathit{Init}, \Sigma, \rightarrow \rangle$$

Elements of C are called configurations.

Use: concrete operational semantics of control flow automata.

Labeled Kripke Structures

$$\begin{aligned} \mathcal{M} &= \langle S, \mathit{Init}, \mathit{Bad}, \Sigma, \rightarrow \rangle \\ &= \mathit{LTS} + \mathit{Bad} \end{aligned}$$

Elements of S are called states.

Use: models (in general abstract ones) for abstraction refinement.

Simulation Relation: Definition

Consider two labeled Kripke structures:

$$\mathcal{M}^c = \langle S^c, Init^c, Bad^c, \Sigma, \rightarrow^c \rangle \qquad \mathcal{M}^a = \langle S^a, Init^a, Bad^a, \Sigma, \rightarrow^a \rangle$$

A simulation relation from $\mathcal{M}^c$ to $\mathcal{M}^a$ is any binary relation $\prec \subseteq S^c \times S^a$ satisfying:

$$\forall$$

$$s^c \quad \prec \quad s^a$$

$$\sigma \downarrow$$

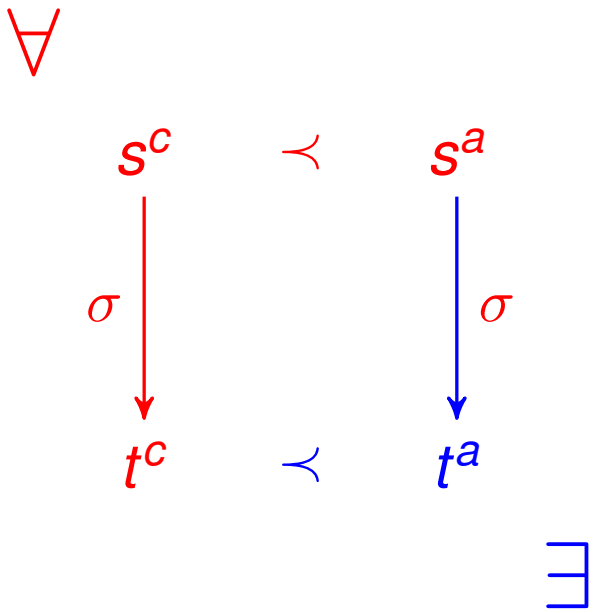
$$t^c$$

Simulation Relation: Definition

Consider two labeled Kripke structures:

$$\mathcal{M}^c = \langle S^c, Init^c, Bad^c, \Sigma, \rightarrow^c \rangle \qquad \mathcal{M}^a = \langle S^a, Init^a, Bad^a, \Sigma, \rightarrow^a \rangle$$

A simulation relation from $\mathcal{M}^c$ to $\mathcal{M}^a$ is any binary relation $\prec \subseteq S^c \times S^a$ satisfying:



Abstraction and Refinement

Consider two labeled Kripke structures:

$$\mathcal{M}^c = \langle S^c, Init^c, Bad^c, \Sigma, \rightarrow^c \rangle \qquad \mathcal{M}^a = \langle S^a, Init^a, Bad^a, \Sigma, \rightarrow^a \rangle$$

If there exists a simulation relation $\prec$ from $\mathcal{M}^c$ to $\mathcal{M}^a$ such that

$$\begin{cases} \forall s^c \in Init^c \ \cdot \ \exists s^a \in Init^a \ \cdot \ s^c \prec s^a \\ \forall (s^c, s^a) \in \prec \ \cdot \ s^c \in Bad^c \implies s^a \in Bad^a \end{cases}$$

then we say that:

$$\mathcal{M}^a \text{ is an } \textcolor{red}{\text{abstraction}} \text{ of } \mathcal{M}^c$$
$$\mathcal{M}^c \text{ is a } \textcolor{red}{\text{refinement}} \text{ of } \mathcal{M}^a$$

Preservation of Safety Properties

A labeled Kripke structure $\mathcal{M} = \langle S, \mathit{Init}, \mathit{Bad}, \Sigma, \rightarrow \rangle$ is safe if it contains no path

$$s_0 \xrightarrow{\sigma_0} s_1 \cdots s_{k-1} \xrightarrow{\sigma_k} s_k \quad \text{with} \quad \begin{cases} s_0 \in \mathit{Init} \\ s_k \in \mathit{Bad} \end{cases}$$

Theorem (Safety Preservation)

For any two labeled Kripke structures $\mathcal{M}^c$ and $\mathcal{M}^a$,

if $\mathcal{M}^a$ is an abstraction of $\mathcal{M}^c$ and $\mathcal{M}^a$ is safe then $\mathcal{M}^c$ is safe.

The converse does not hold.

Preservation of Safety Properties

A labeled Kripke structure $\mathcal{M} = \langle S, \mathit{Init}, \mathit{Bad}, \Sigma, \rightarrow \rangle$ is safe if it contains no path

$$s_0 \xrightarrow{\sigma_0} s_1 \cdots s_{k-1} \xrightarrow{\sigma_k} s_k \quad \text{with} \quad \begin{cases} s_0 \in \mathit{Init} \\ s_k \in \mathit{Bad} \end{cases}$$

Theorem (Safety Preservation)

For any two labeled Kripke structures $\mathcal{M}^c$ and $\mathcal{M}^a$,

if $\mathcal{M}^a$ is an abstraction of $\mathcal{M}^c$ and $\mathcal{M}^a$ is safe then $\mathcal{M}^c$ is safe.

The converse does not hold.

Preservation of Safety Properties: Application

We want to show that a concrete labeled Kripke structure $\mathcal{M}^c$ is safe.

If $\mathcal{M}^c$ cannot be directly model-checked then:

1. design an abstract labeled Kripke structure $\mathcal{M}^a$, simpler than $\mathcal{M}^c$, and exhibit a simulation relation $\prec$ that shows that $\mathcal{M}^a$ is an abstraction of $\mathcal{M}^c$.
2. check that $\mathcal{M}^a$ is safe

If $\mathcal{M}^a$ is safe then $\mathcal{M}^c$ is safe

However, If $\mathcal{M}^a$ is not safe then we cannot conclude that $\mathcal{M}^c$ is not safe.

Preservation of Safety Properties: Completeness

Consider a labeled Kripke structure $\mathcal{M}^c = \langle S^c, Init^c, Bad^c, \Sigma, \rightarrow^c \rangle$.

$$(\mathcal{M}^a) \qquad \begin{array}{rclcrcl} S^a & = & \{reach\} & & Init^a & = & \{reach\} \\ \rightarrow^a & = & \{reach\} \times \Sigma \times \{reach\} & & Bad^a & = & \emptyset \end{array}$$

The relation $\prec \; = \; \text{Post}^*(\mathcal{M}^c) \times \{reach\}$ is obviously a simulation relation from $\mathcal{M}^c$ to $\mathcal{M}^a$. Note that $\mathcal{M}^a$ is safe. Moreover:

$$\text{if } \mathcal{M}^c \text{ is safe then } \begin{cases} \forall s^c \in Init^c \;\cdot\; \exists s^a \in Init^a \;\cdot\; s^c \prec s^a \\ \forall (s^c, s^a) \in \prec \;\cdot\; s^c \in Bad^c \implies s^a \in Bad^a \end{cases}$$

Theorem (Relative Completeness)

For any safe labeled Kripke structure $\mathcal{M}^c$, there exists a finite-state abstraction $\mathcal{M}^a$ of $\mathcal{M}^c$ such that $\mathcal{M}^a$ is safe.

Finite-state abstractions are sufficient to prove safety of any model.

Outline — Abstract Model Refinement

Two Steps

Presentation of abstraction schemes at the Semantic Level

Forget about control flow automata and programs

But keep them in mind for intuitions ☺

Implementation of predicate abstraction for control flow automata

Two Steps

Presentation of abstraction schemes at the Semantic Level

Forget about control flow automata and programs

But keep them in mind for intuitions ☺

Implementation of predicate abstraction for control flow automata

Partition Abstraction: Definition

Consider a labeled Kripke structure $\mathcal{M}^c = \langle S^c, Init^c, Bad^c, \Sigma, \rightarrow^c \rangle$.

Partition given by (S^a, α) where S^a is a finite set and $\alpha : S^c \rightarrow S^a$

Partition Abstraction $\mathcal{M}^a$ Induced by (S^a, α)

$$
\begin{aligned}
Init^a &= \{\alpha(s^c) \mid s^c \in Init^c\} \\
Bad^a &= \{\alpha(s^c) \mid s^c \in Bad^c\} \\
\rightarrow^a &= \{(\alpha(s^c), \sigma, \alpha(t^c)) \mid (s^c, \sigma, t^c) \in \rightarrow^c\}
\end{aligned}
$$

The simulation relation $\prec = \{(s^c, \alpha(s^c)) \mid s^c \in S^c\}$ shows that

$\mathcal{M}^a$ is an abstraction of $\mathcal{M}^c$

Partition Abstraction: Explanation

Partition (S^a, α)

- S^a finite set
- $\alpha : S^c \to S^a$

Partition Abstraction Induced by (S^a, α)

$$\begin{array}{rcll} Init^a & = & \{\alpha(s^c) \mid s^c \in Init^c\} & (Bad^a \ldots) \\ \rightarrow^a & = & \{(\alpha(s^c), \sigma, \alpha(t^c)) \mid (s^c, \sigma, t^c) \in \rightarrow^c\} & \end{array}$$

Induced equivalence relation $\sim$ defined by: $s^c \sim t^c$ if $\alpha(s^c) = \alpha(t^c)$.

Abstraction Function $\alpha : S^c \to S^a$

All concrete states in an equivalence class are merged together.

Induced Concretization Function $\gamma : S^a \to \mathcal{P}(S^c)$

$$\gamma(s^a) = \{s^c \mid \alpha(s^c) = s^a\}$$

Not a Galois Connection

(α, γ) becomes a Galois Connection when lifted to powersets.

Partition Abstraction: Explanation

Partition (S^a, α)

- S^a finite set
- $\alpha : S^c \to S^a$

Partition Abstraction Induced by (S^a, α)

$$\begin{array}{rcll} \mathit{Init}^a & = & \{\alpha(s^c) \mid s^c \in \mathit{Init}^c\} & (\mathit{Bad}^a \ldots) \\ \to^a & = & \{(\alpha(s^c), \sigma, \alpha(t^c)) \mid (s^c, \sigma, t^c) \in \to^c\} & \end{array}$$

Init^a, Bad^a and $\to^a$ are existential lifts of their concrete counterparts:

$$s^a \in \mathit{Init}^a \quad \text{iff} \quad \exists s^c \cdot \begin{cases} \alpha(s^c) = s^a & \wedge \\ s^c \in \mathit{Init}^c & \end{cases}$$

$$(s^a, \sigma, t^a) \in \to^a \quad \text{iff} \quad \exists s^c \; \exists t^c \cdot \begin{cases} \alpha(s^c) = s^a & \wedge \\ \alpha(t^c) = t^a & \wedge \\ (s^c, \sigma, t^c) \in \to^c & \end{cases}$$

Partition Abstraction: Computation of $\mathcal{M}^a$

Consider a labeled Kripke structure $\mathcal{M}^c = \langle S^c, \mathit{Init}^c, \mathit{Bad}^c, \Sigma, \rightarrow^c \rangle$.

Computation of Init^a

$\mathtt{I} \leftarrow \emptyset$
foreach $\mathtt{s}^a \in S^a$
  if $\exists s^c \cdot (s^c \in \gamma(\mathtt{s}^a) \wedge s^c \in \mathit{Init}^c)$
    $\mathtt{I} \leftarrow \mathtt{I} \cup \{\mathtt{s}^a\}$
return $\mathtt{I}$

Computation of $\rightarrow^a$

$\mathtt{R} \leftarrow \emptyset$
foreach $(\mathtt{s}^a, \sigma, \mathtt{t}^a) \in S^a \times \Sigma \times S^a$
  if $\exists s^c \; \exists t^c \cdot (s^c \in \gamma(\mathtt{s}^a) \wedge t^c \in \gamma(\mathtt{t}^a) \wedge (s^c, \sigma, t^c) \in \rightarrow^c)$
    $\mathtt{R} \leftarrow \mathtt{R} \cup \{(\mathtt{s}^a, \sigma, \mathtt{t}^a)\}$
return $\mathtt{R}$

Partition Abstraction: Implementation Issues

- Machine representation of $\alpha : S^c \to S^a$ or $\gamma : S^a \to \mathcal{P}(S^c)$
  - Examples: BDDs (if $S^c = \{0,1\}^n$), NDDs (if $S^c = \mathbb{Z}^n$), ...
- Algorithms to decide the conditions

$$\exists s^c \cdot (s^c \in \gamma(s^a) \wedge s^c \in \mathit{Init}^c)$$

$$\exists s^c \cdot (s^c \in \gamma(s^a) \wedge s^c \in \mathit{Bad}^c)$$

$$\exists s^c \, \exists t^c \cdot (s^c \in \gamma(s^a) \wedge t^c \in \gamma(t^a) \wedge (s^c, \sigma, t^c) \in \to^c)$$

Partial Algorithms (yes / no / ?) Are Sufficient

Safety preservation from $\mathcal{M}^a$ to $\mathcal{M}^c$ still holds if Init^a, Bad^a and $\to^a$ are larger than the "optimal ones". We may soundly consider "?" as "yes".

Partition Abstraction: Refinement

Given two equivalence relations $\sim_1$ and $\sim_2$ on some set S, we say that $\sim_2$ is finer than $\sim_1$ if $\sim_2 \subseteq \sim_1$, or equivalently if each equivalence class of $\sim_1$ is a union of equivalence classes of $\sim_2$.

Consider two partitions (S_1^a, α_1) and (S_2^a, α_2).

If $\sim_2$ is finer than $\sim_1$ then $\mathcal{M}^a(S_2^a, \alpha_2)$ is a refinement of $\mathcal{M}^a(S_1^a, \alpha_1)$.

Informally

To refine a partition abstraction, split some equivalence classes.

Recomputation of $\mathcal{M}^a$ after refinement

- Refinement is local to equivalence classes that are split.
- If $\mathcal{M}^a$ is stored explicitly then the refined $\rightarrow^a$ can be efficiently computed from the previous $\rightarrow^a$.

Predicate Language

Predicates

Formulas in first-order logic over some vocabulary

Example

For control flow automata, take the same vocabulary as in expressions:

$$\langle \ldots, -1, 0, 1, \ldots \ ; \ +, -, * \ ; \ <, \leq, =, \neq, \geq, > \rangle$$

At the semantic level, we view predicates as sets of states.

Predicate Language

Predicates

Formulas in first-order logic over some vocabulary

Example

For control flow automata, take the same vocabulary as in expressions:

$$\langle \ldots, -1, 0, 1, \ldots \ ; \ +, -, * \ ; \ <, \leq, =, \neq, \geq, > \rangle$$

At the semantic level, we view predicates as sets of states.

Boolean Predicate Abstraction: Definition

Consider a labeled Kripke structure $\mathcal{M}^c = \langle S^c, \mathit{Init}^c, \mathit{Bad}^c, \Sigma, \rightarrow^c \rangle$.

Support predicates given by a finite set Φ of subsets of S^c

Characteristic Function of $\phi \in \Phi$

$$\mathbf{1}_\phi : S^c \rightarrow \{0,1\}$$

$$s^c \mapsto \begin{cases} 1 & \text{if } s^c \in \phi \\ 0 & \text{if } s^c \notin \phi \end{cases}$$

Partition (S^a_Φ, α_Φ)

$$S^a_\Phi = \Phi \rightarrow \{0,1\}$$

$$\alpha_\Phi(s^c) = \lambda \phi \,.\, \mathbf{1}_\phi(s^c)$$

Boolean Predicate Abstraction $\mathcal{M}^a$ Induced by Φ

Partition abstraction induced by the partition (S^a_Φ, α_Φ)

Boolean Predicate Abstraction: Explanation

Partition (S^a_Φ, α_Φ)

$$S^a_\Phi = \Phi \to \{0,1\}$$

$$\alpha_\Phi(s^c) = \lambda\,\phi\,.\,\mathbf{1}_\phi(s^c)$$

Intuition

Abstract state: truth value for each predicate

α_Φ merges concrete states that satisfy the same predicates.

Induced Concretization Function $\gamma : S^a \to \mathcal{P}(S^c)$

$$\gamma_\Phi(s^a) = \bigcap_{s^a(\phi)=1} \phi \quad \cap \quad \bigcap_{s^a(\phi)=0} S^c \setminus \phi$$

Not a Galois Connection

$(\alpha_\Phi, \gamma_\Phi)$ becomes a Galois Connection when lifted to powersets.

Boolean Predicate Abstraction: Computation of $\mathcal{M}^a$

$Init^a$, Bad^a and $\rightarrow^a$ can be computed as for partition abstractions, but:

Exponential complexity

Number of abstract states: $2^{|\Phi|}$

Exponential number of decisions $\exists s^c \; \exists t^c \; \cdot \; (\cdots)$ to compute $\rightarrow^a$

Exploit the structure of the partition to get better algorithms (in practice)

Computation of $\alpha(U) = \{\alpha(s^c) \mid s^c \in U\}$ where $U \subseteq S^c$

If $U \subseteq \phi$ then every $s^a \in \alpha(U)$ necessarily satisfies $s^a(\phi) = 1$.

In that case, there is no need to examine candidates where $s^a(\phi) = 0$.

$$\Phi_1 = \{\phi \in \Phi \mid U \subseteq \phi\} \qquad \Phi_0 = \{\phi \in \Phi \mid U \subseteq S^c \setminus \phi\}$$

New complexity linear in $|\Phi_0| + |\Phi_1|$ and exponential in $|\Phi \setminus (\Phi_0 \cup \Phi_1)|$

Boolean Predicate Abstraction: Computation of $\mathcal{M}^a$

$Init^a$, Bad^a and $\rightarrow^a$ can be computed as for partition abstractions, but:

Exponential complexity

Number of abstract states: $2^{|\Phi|}$

Exponential number of decisions $\exists s^c \; \exists t^c \; . \; (\cdots)$ to compute $\rightarrow^a$

Exploit the structure of the partition to get better algorithms (in practice)

Computation of $\alpha(U) = \{\alpha(s^c) \mid s^c \in U\}$ where $U \subseteq S^c$

If $U \subseteq \phi$ then every $s^a \in \alpha(U)$ necessarily satisfies $s^a(\phi) = 1$.

In that case, there is no need to examine candidates where $s^a(\phi) = 0$.

$$\Phi_1 = \{\phi \in \Phi \mid U \subseteq \phi\} \qquad \Phi_0 = \{\phi \in \Phi \mid U \subseteq S^c \setminus \phi\}$$

New complexity linear in $|\Phi_0| + |\Phi_1|$ and exponential in $|\Phi \setminus (\Phi_0 \cup \Phi_1)|$

Boolean Predicate Abstraction: Computation of $\mathcal{M}^a$

$Init^a$, Bad^a and $\rightarrow^a$ can be computed as for partition abstractions, but:

Exponential complexity

Number of abstract states: $2^{|\Phi|}$

Exponential number of decisions $\exists s^c \; \exists t^c \; \cdot \; (\cdots)$ to compute $\rightarrow^a$

Exploit the structure of the partition to get better algorithms (in practice)

Computation of $\alpha(U) = \{\alpha(s^c) \mid s^c \in U\}$ where $U \subseteq S^c$

If $U \subseteq \phi$ then every $s^a \in \alpha(U)$ necessarily satisfies $s^a(\phi) = 1$.

In that case, there is no need to examine candidates where $s^a(\phi) = 0$.

$$\Phi_1 = \{\phi \in \Phi \mid U \subseteq \phi\} \qquad \Phi_0 = \{\phi \in \Phi \mid U \subseteq S^c \setminus \phi\}$$

New complexity linear in $|\Phi_0| + |\Phi_1|$ and exponential in $|\Phi \setminus (\Phi_0 \cup \Phi_1)|$

Boolean Predicate Abstraction: Implementation Issues

Each abstract state is a truth valuation of the predicates.

Sets of abstract states (e.g. $Init^a$, Bad^a) are sets of truth valuations.

Natural Encoding

Propositional Formulas

Introduce propositional variables $\mathrm{p}_\phi, \mathrm{p}'_\phi$ for each predicate ϕ.

$$s^a \leftrightsquigarrow \bigwedge_{\phi \in \Phi} \overline{\mathrm{p}_\phi} \qquad \text{(conjunction of literals)}$$

$$Init^a, Bad^a \leftrightsquigarrow \bigvee \bigwedge_{\phi \in \Phi} \overline{\mathrm{p}_\phi} \qquad \text{(formula on } \mathrm{p}_\phi\text{)}$$

$$\rightarrow^a \leftrightsquigarrow \bigvee \bigwedge_{\phi \in \Phi} \overline{\mathrm{p}_\phi} \bigwedge_{\phi \in \Phi} \overline{\mathrm{p}'_\phi} \qquad \text{(formula on } \mathrm{p}_\phi, \mathrm{p}'_\phi\text{)}$$

Use BDDs to represent these propositional formulas ☺

Boolean Predicate Abstraction: Implementation Issues

Each abstract state is a truth valuation of the predicates.

Sets of abstract states (e.g. $Init^a$, Bad^a) are sets of truth valuations.

Natural Encoding

Propositional Formulas

Introduce propositional variables $\mathrm{p}_\phi, \mathrm{p}'_\phi$ for each predicate ϕ.

$$s^a \quad \leftrightsquigarrow \quad \bigwedge_{\phi \in \Phi} \overline{\mathrm{p}_\phi} \qquad \text{(conjunction of literals)}$$

$$Init^a, Bad^a \quad \leftrightsquigarrow \quad \bigvee \bigwedge_{\phi \in \Phi} \overline{\mathrm{p}_\phi} \qquad \text{(formula on } \mathrm{p}_\phi\text{)}$$

$$\longrightarrow^a \quad \leftrightsquigarrow \quad \bigvee \bigwedge_{\phi \in \Phi} \overline{\mathrm{p}_\phi} \bigwedge_{\phi \in \Phi} \overline{\mathrm{p}'_\phi} \qquad \text{(formula on } \mathrm{p}_\phi, \mathrm{p}'_\phi\text{)}$$

Use BDDs to represent these propositional formulas ☺

Boolean Predicate Abstraction: Refinement

If $\Phi_2 \supseteq \Phi_1$ then $\mathcal{M}^a(\Phi_2)$ is a refinement of $\mathcal{M}^a(\Phi_1)$.

Informally

To refine a boolean predicate abstraction, add new predicates.

Recomputation of $\mathcal{M}^a$ after refinement

- Refinement is global, since it can impact all abstract states.

Cartesian Predicate Abstraction: Introduction

Support predicates given by a finite set Φ of subsets of S^c

Objective

Avoid exponential cost in the abstraction of a set U of concrete states

A monomial is a conjunction of literals $\bigwedge_{\phi \in \Phi'} \overline{\mathrm{p}_\phi}$ for some $\Phi' \subseteq \Phi$.

Solution

Replace disjunctions of abstract states by the most precise monomial.

$$\text{Boolean:} \quad U \subseteq S^c \overset{\alpha}{\rightsquigarrow} \bigvee \bigwedge_{\phi \in \Phi} \overline{\mathrm{p}_\phi}$$

$$\text{Cartesian:} \quad U \subseteq S^c \overset{\alpha}{\rightsquigarrow} \bigwedge_{\phi \in \Phi'} \overline{\mathrm{p}_\phi} \qquad (\Phi' \subseteq \Phi)$$

Cartesian Predicate Abstraction: Trivectors

Encoding of Monomials

Encode $\bigwedge_{\phi \in \Phi'} \overline{\mathtt{p}_\phi}$ as the valuation

$$v(\phi) = \begin{cases} 1 & \text{if } \overline{\mathtt{p}_\phi} = p_\phi \\ 0 & \text{if } \overline{\mathtt{p}_\phi} = \neg p_\phi \\ * & \text{if } \phi \notin \Phi' \end{cases}$$

3-Valued Characteristic Function

$$\mathbf{1}_\phi : \mathcal{P}(S^c) \rightarrow \{0, 1, *\}$$

$$U \neq \emptyset \mapsto \begin{cases} 1 & \text{if } U \subseteq \phi \\ 0 & \text{if } U \subseteq S^c \setminus \phi \\ * & \text{otherwise} \end{cases}$$

Cartesian Abstraction and Concretization Functions

$$S^a_\Phi = \Phi \rightarrow \{0, 1, *\}$$

$$\alpha_\Phi(U) = \lambda \phi \,.\, \mathbf{1}_\phi(U) \qquad (U \neq \emptyset)$$

$$\gamma_\Phi(s^a) = \bigcap_{s^a(\phi)=1} \phi \quad \cap \quad \bigcap_{s^a(\phi)=0} S^c \setminus \phi$$

Cartesian Predicate Abstraction: Definition

Notation: Concrete Post Operator

$$\mathrm{Post}^c(U, \sigma) \;=\; \{t^c \in \mathit{State}^c \mid \exists s^c \in U \;\cdot\; (s^c, \sigma, t^c) \in \rightarrow^c\}$$

Cartesian Predicate Abstraction $\mathcal{M}^a$ Induced by Φ

$$\begin{aligned}
S^a_\Phi &= \Phi \rightarrow \{0, 1, *\} \\
\mathit{Init}^a &= \{\alpha_\Phi(s^c) \mid s^c \in \mathit{Init}^c\} \\
\mathit{Bad}^a &= \{s^a \mid s^a \in S^a, \gamma_\Phi(s^a) \cap \mathit{Bad}^c \neq \emptyset\} \\
\rightarrow^a &= \left\{(s^a, \sigma, \alpha_\Phi \circ \mathrm{Post}^c\,(\gamma_\Phi(s^a), \sigma)) \;\middle|\; s^a \in S^a, \sigma \in \Sigma\right\}
\end{aligned}$$

The simulation relation $\prec \;=\; \{(s^c, s^a) \mid s^c \in \gamma_\Phi(s^a)\}$ shows that

$\mathcal{M}^a$ is an abstraction of $\mathcal{M}^c$

Cartesian Predicate Abstraction: Remarks

Cartesian Predicate Abstraction $\mathcal{M}^a$ Induced by Φ

$$
\begin{aligned}
S^a_\Phi &= \Phi \to \{0, 1, *\} \\
\gamma_\Phi(s^a) &= \bigcap_{s^a(\phi)=1} \phi \quad \cap \quad \bigcap_{s^a(\phi)=0} S^c \setminus \phi \\
\to^a &= \left\{ (s^a, \sigma, \alpha_\Phi \circ \mathsf{Post}^c \circ \gamma_\Phi(s^a)) \;\middle|\; s^a \in S^a, \sigma \in \Sigma \right\}
\end{aligned}
$$

Abstract state: truth value in $\{0, 1, *\}$ for each $\phi \in \Phi$. Not a partition!

The special value $*$ is conservatively treated as "don't know" in γ_Φ.

The transition relation $\to^a$ is deterministic (at most one successor).

Galois Connection

$(\alpha_\Phi, \gamma_\Phi)$ is a Galois Connection (with $0, 1 \sqsubseteq *$).

Cartesian Predicate Abstraction: Computation of $\mathcal{M}^a$

Computation of Init^a

Same as boolean case

Computation of $\alpha(U)$

```
foreach φ ∈ Φ
    if U ⊆ S^c \ φ
        s^a[φ] ← 0
    else if U ⊆ φ
        s^a[φ] ← 1
    else
        s^a[φ] ← *
return s^a
```

Computation of $\rightarrow^a$

```
R ← ∅
foreach (s^a, σ) ∈ S^a × Σ | Post^c(γ(s^a), σ) ≠ ∅
    foreach φ ∈ Φ
        if Post^c(γ(s^a), σ) ⊆ S^c \ φ
            t^a[φ] ← 0
        else if Post^c(γ(s^a), σ) ⊆ φ
            t^a[φ] ← 1
        else
            t^a[φ] ← *
    R ← R ∪ {(s^a, σ, t^a)}
return R
```

Linear number of decisions $\mathrm{Post}^c(\gamma(s^a), \sigma) \subseteq \ldots$ to compute the successor $\rightarrow^a (s^a, \sigma)$ of a given abstract state s^a and action $\sigma \in \Sigma$.

Cartesian Pred. Abstraction: Implementation Issues

Similar to boolean predicate abstraction:

- Encoding with 3-valued propositional variables $\mathrm{p}_\phi, \mathrm{p}'_\phi$
- Representation with TDDs (or BDDs via binary encoding)

For concrete labeled Kripke structures obtained from programs, the cartesian predicate abstraction can be presented as boolean program.

Cartesian Predicate Abstraction: Refinement

If $\Phi_2 \supseteq \Phi_1$ then $\mathcal{M}^a(\Phi_2)$ is a refinement of $\mathcal{M}^a(\Phi_1)$.

Informally

To refine a cartesian predicate abstraction, add new predicates.

Recomputation of $\mathcal{M}^a$ after refinement

- Refinement is global, since it can impact all abstract states.

How about Programs?

Control flow automaton: $\langle Q, q_{in}, \mathrm{X}, \rightarrow \rangle$. Set $Q_{BAD} \subseteq Q$ of bad locations.

Concrete Labeled Kripke Structure $\mathcal{M}^c$

$$
\begin{array}{rclcrcl}
S^c & = & Q \times (\mathrm{X} \rightarrow \mathbb{R}) & & Init^c & = & \{q_{in}\} \times (\mathrm{X} \rightarrow \mathbb{R}) \\
\Sigma & = & \mathrm{Op} & & Bad^c & = & Q_{BAD} \times (\mathrm{X} \rightarrow \mathbb{R}) \\
\rightarrow^c & = & \left\{ ((q, u^c), \sigma, (q', v^c)) \;\middle|\; q \xrightarrow{\mathrm{op}} q' \text{ and } (u^c, v^c) \in [\![\mathrm{op}]\!]^c \right\} & & & &
\end{array}
$$

The usual semantics $[\![\mathrm{op}]\!]$ of operations is now written $[\![\mathrm{op}]\!]^c$.

Nothing Surprising Here!

This is the usual labeled transition system (operational semantics of control flow automata) equipped with the usual bad configurations.

Predicate Language

Control flow automaton: $\langle Q, q_{in}, \mathtt{X}, \rightarrow \rangle$.

Vocabulary

$$\langle \ldots, -1, 0, 1, \ldots \ ; \ +, -, * \ ; \ <, \leq, =, \neq, \geq, > \rangle$$

Additive and multiplicative theory of the reals is decidable.

Finite Set Φ of Support Predicates

(Quantifier-free) first-order formulas with free variables in $\mathtt{X}$

Semantics of Support Predicates

The interpretation $[\![\varphi]\!]$ of a predicate φ is a subset of $\mathtt{X} \rightarrow \mathbb{R}$.

Link With Semantic Level Abstraction Schemes

The interpretations $[\![\varphi]\!]$ replace the "semantic support predicates" ϕ.

Boolean Predicate Abstraction: Definition

Boolean Predicate Abstraction $\mathcal{M}^a$ Induced by Φ

$$
\begin{array}{rclcrcl}
S^a & = & Q \times (\Phi \to \{0,1\}) & & \mathit{Init}^a & = & \{q_{in}\} \times (\Phi \to \{0,1\}) \\
\Sigma & = & \mathrm{Op} & & \mathit{Bad}^a & = & Q_{BAD} \times (\Phi \to \{0,1\}) \\
\to^a & = & \left\{ ((q,u^a),\sigma,(q',v^a)) \;\middle|\; q \xrightarrow{\mathrm{op}} q' \text{ and } (u^a,v^a) \in [\![\mathrm{op}]\!]^a \right\} & & & &
\end{array}
$$

- Concrete valuations in $\mathrm{X} \to \mathbb{R}$ are replaced by abstract valuations in $\Phi \to \{0,1\}$.
- The control flow automaton's graph is kept intact.
- All the work is done in the abstract semantics of operations.

Boolean Predicate Abstraction: Definition

Syntactic Concretization

Concretization formula $\gamma(v^a)$ of a valuation $v^a \in \Phi \to \{0,1\}$ defined by

$$\gamma(v^a) \quad = \quad \bigwedge_{\substack{\varphi \in \Phi \\ v^a(\varphi)=1}} \varphi \quad \wedge \quad \bigwedge_{\substack{\varphi \in \Phi \\ v^a(\varphi)=0}} \neg\varphi$$

Abstract semantics $[\![\mathrm{op}]\!]^a$ of operations defined as a binary relation

$$[\![\mathrm{op}]\!]^a \quad \subseteq \quad (\Phi \to \{0,1\}) \times (\Phi \to \{0,1\})$$

Guards: $(u^a, v^a) \in [\![g]\!]^a$ if $v^a = u^a$ and $\gamma(u^a) \wedge g$ sat.

Assignments: $(u^a, v^a) \in [\![x := e]\!]^a$ if $\gamma(u^a) \wedge \gamma(v^a)[e/x]$ sat.

Boolean Predicate Abstraction: Computation of $\mathcal{M}^a$

Safety checking of $\mathcal{M}^a$ usually performed by forward graph exploration.

Computation of $\{v^a \in \Phi \to \{0,1\} \mid (u^a, v^a) \in [\![g]\!]^a\}$

if $\gamma(\mathrm{u}^a) \wedge g$ is satisfiable
  return $\{\mathrm{u}^a\}$
else
  return $\emptyset$

Computation of $\{v^a \in \Phi \to \{0,1\} \mid (\mathrm{u}^a, v^a) \in [\![x := e]\!]^a\}$

$\mathrm{S} \leftarrow \emptyset$
foreach $\mathrm{v}^a \in \Phi \to \{0,1\}$ (exponential ☹)
  if $\gamma(\mathrm{u}^a) \wedge \gamma(\mathrm{v}^a)[e/x]$ is satisfiable
    $\mathrm{S} \leftarrow \mathrm{S} \cup \{\mathrm{v}^a\}$
return S

Cartesian Predicate Abstraction: Definition

Cartesian Predicate Abstraction $\mathcal{M}^a$ Induced by Φ

$$S^a = Q \times (\Phi \to \{0, 1, *\}) \qquad Init^a = \{q_{in}\} \times \{\lambda\, \varphi\, .\, *\}$$

$$\Sigma = \mathrm{Op} \qquad Bad^a = Q_{BAD} \times \{\lambda\, \varphi\, .\, *\}$$

$$\to^a = \left\{ ((q, u^a), \sigma, (q', v^a)) \;\middle|\; q \xrightarrow{\mathrm{op}} q' \text{ and } v^a = [\![\mathrm{op}]\!]^a(u^a) \right\}$$

Syntactic Concretization

Concretization formula $\gamma(v^a)$ of a valuation $v^a \in \Phi \to \{0, 1, *\}$ defined by

$$\gamma(v^a) = \bigwedge_{\substack{\varphi \in \Phi \\ v^a(\varphi)=1}} \varphi \quad \wedge \quad \bigwedge_{\substack{\varphi \in \Phi \\ v^a(\varphi)=0}} \neg\varphi$$

Cartesian Predicate Abstraction: Definition

Abstract semantics $[\![\mathrm{op}]\!]^a$ of operations defined as a partial function

$$[\![\mathrm{op}]\!]^a : (\Phi \to \{0, 1, *\}) \to (\Phi \to \{0, 1, *\})$$

Guards

If $\gamma(u^a) \wedge g$ is unsatisfiable then $[\![\mathrm{op}]\!]^a(u^a)$ is undefined

Otherwise $[\![\mathrm{op}]\!]^a(u^a) = \lambda\varphi.\begin{cases} 0 & \text{if } (\gamma(u^a) \wedge g) \Rightarrow \neg\varphi \text{ is valid} \\ 1 & \text{if } (\gamma(u^a) \wedge g) \Rightarrow \varphi \text{ is valid} \\ * & \text{otherwise} \end{cases}$

Assignments

If $\gamma(u^a)$ is unsatisfiable then $[\![x := e]\!]^a(u^a)$ is undefined

Otherwise $[\![x := e]\!]^a(u^a) = \lambda\varphi.\begin{cases} 0 & \text{if } \gamma(u^a) \Rightarrow \neg\varphi[e/x] \text{ is valid} \\ 1 & \text{if } \gamma(u^a) \Rightarrow \varphi[e/x] \text{ is valid} \\ * & \text{otherwise} \end{cases}$

Cartesian Predicate Abstraction: Computation of $\mathcal{M}^a$

Safety checking of $\mathcal{M}^a$ usually performed by forward graph exploration.

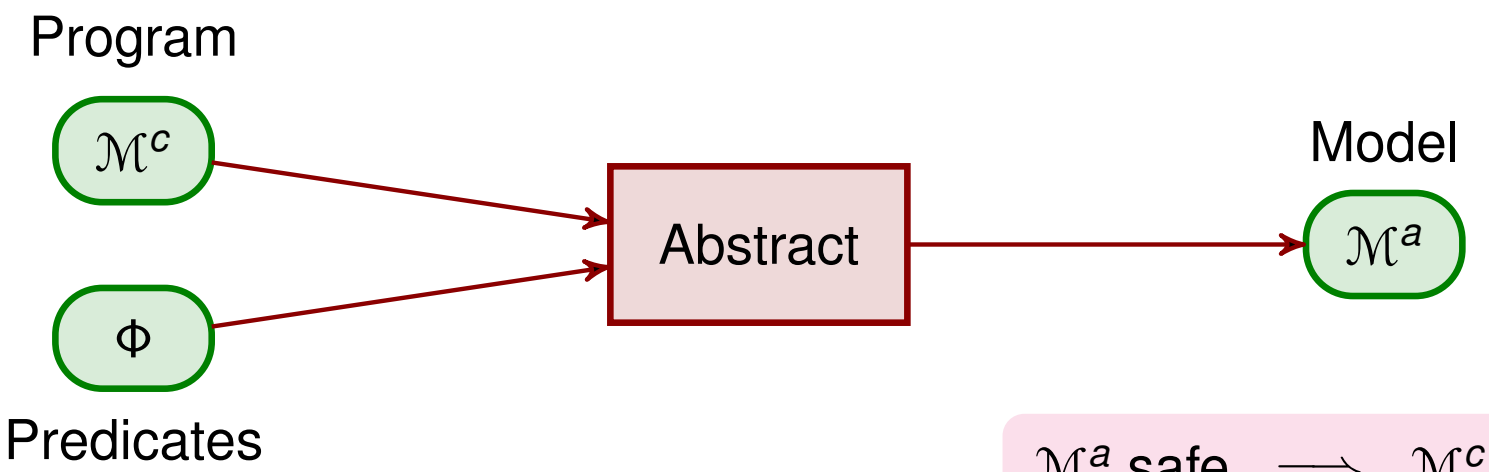
Computation of $[\![g]\!]^a(u^a)$

if $\gamma(\mathrm{u}^a) \wedge g$ is unsatisfiable
  return undefined
foreach $\varphi \in \Phi$ (linear ☺)
  if $\models (\gamma(\mathrm{u}^a) \wedge g) \Rightarrow \neg\varphi$
    $\mathrm{v}^a[\varphi] \leftarrow 0$
  else if $\models (\gamma(\mathrm{u}^a) \wedge g) \Rightarrow \varphi$
    $\mathrm{v}^a[\varphi] \leftarrow 1$
  else
    $\mathrm{v}^a[\varphi] \leftarrow *$
return v^a

Computation of $[\![x := e]\!]^a(u^a)$

if $\gamma(\mathrm{u}^a)$ is unsatisfiable
  return undefined
foreach $\varphi \in \Phi$ (linear ☺)
  if $\models \gamma(\mathrm{u}^a) \Rightarrow \neg\varphi[e/x]$
    $\mathrm{v}^a[\varphi] \leftarrow 0$
  else if $\models \gamma(\mathrm{u}^a) \Rightarrow \varphi[e/x]$
    $\mathrm{v}^a[\varphi] \leftarrow 1$
  else
    $\mathrm{v}^a[\varphi] \leftarrow *$
return v^a

Summary: Automatic Predicate Abstraction



Refinement consists in adding new support predicates

Boolean Abstraction

Partition induced by Φ

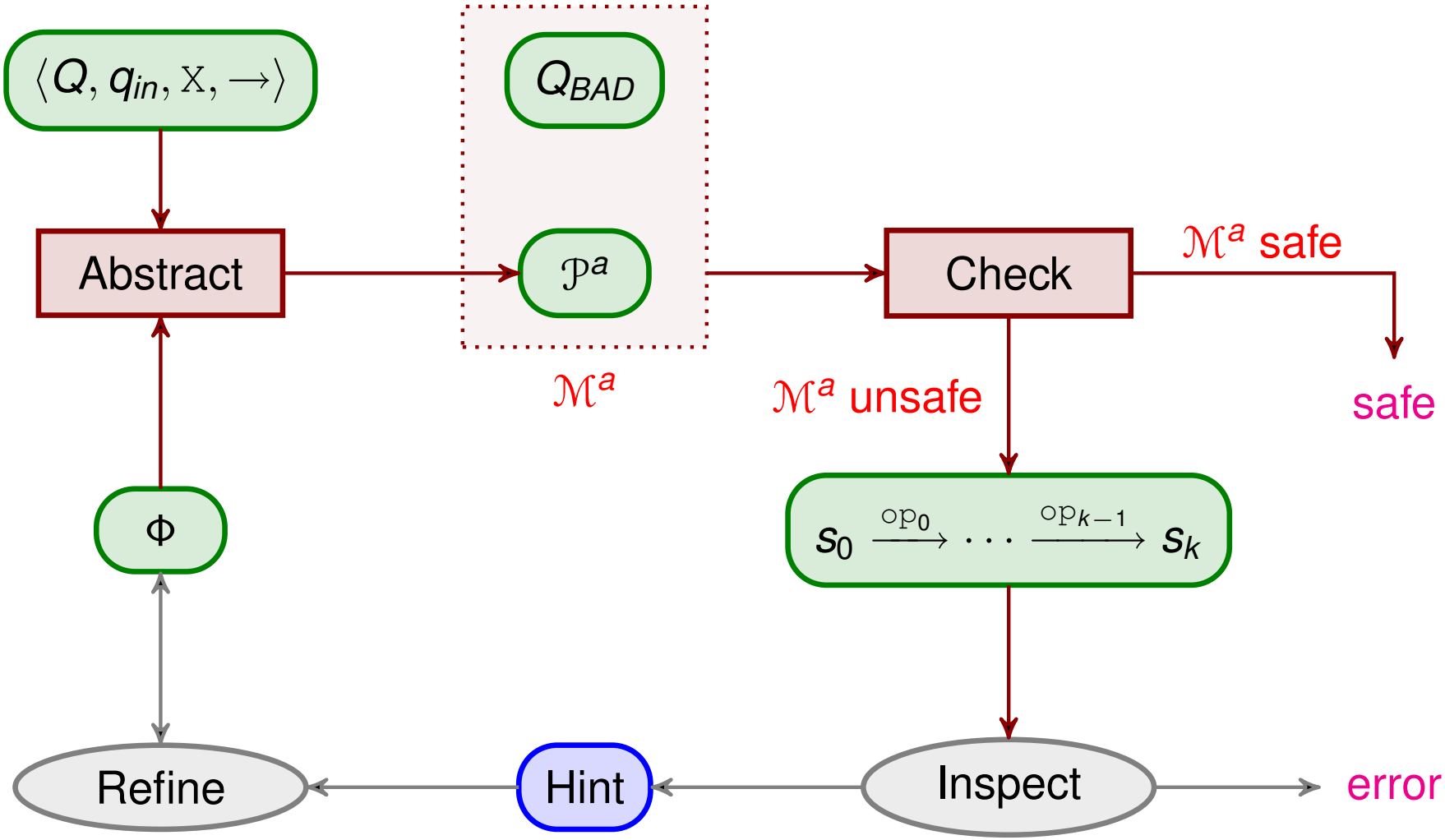
- ☺ Most precise abstraction based of Φ
- ☹ Exponential (for successors)

Cartesian Abstraction

Monomials induced by Φ

- ☹ Less precise than boolean abstraction
- ☺ Linear (for successors)

Verification by Model-Checking Abstract Models



Outline — Abstract Model Refinement

Inspection of Abstract Counterexamples

Control flow automaton: $\langle Q, q_{in}, \mathrm{X}, \rightarrow \rangle$. Set $Q_{BAD} \subseteq Q$ of bad locations.

$\mathcal{M}^a = \langle S^a, Init^a, Bad^a, \Sigma, \rightarrow^a \rangle$ obtained by predicate abstraction

Abstract counterexample

$$(q_{in}, v_0^a) \xrightarrow{\mathrm{op}_0} (q_1, v_1^a) \cdots (q_k, v_k^a) \xrightarrow{\mathrm{op}_k} (q_{bad}, v_{k+1}^a)$$

The abstract counterexample is feasible if there is a concrete run

$$(q_{in}, v_0^c) \xrightarrow{\mathrm{op}_0} (q_1, v_1^c) \cdots (q_k, v_k^c) \xrightarrow{\mathrm{op}_k} (q_{bad}, v_{k+1}^c) \quad \text{with } v_i^c \in \gamma(v_i^a)$$

Better to directly check for all possible abstract predicate valuations!

Objective

Check whether a control path $q_{in} \xrightarrow{\mathrm{op}_0} q_1 \cdots q_k \xrightarrow{\mathrm{op}_k} q_{bad}$ is feasible

Inspection of Abstract Counterexamples

Control flow automaton: $\langle Q, q_{in}, \mathrm{X}, \rightarrow \rangle$. Set $Q_{BAD} \subseteq Q$ of bad locations.

$\mathcal{M}^a = \langle S^a, Init^a, Bad^a, \Sigma, \rightarrow^a \rangle$ obtained by predicate abstraction

Abstract counterexample

$$(q_{in}, v_0^a) \xrightarrow{\mathrm{op}_0} (q_1, v_1^a) \cdots (q_k, v_k^a) \xrightarrow{\mathrm{op}_k} (q_{bad}, v_{k+1}^a)$$

The abstract counterexample is feasible if there is a concrete run

$$(q_{in}, v_0^c) \xrightarrow{\mathrm{op}_0} (q_1, v_1^c) \cdots (q_k, v_k^c) \xrightarrow{\mathrm{op}_k} (q_{bad}, v_{k+1}^c) \quad \text{with } v_i^c \in \gamma(v_i^a)$$

Better to directly check for all possible abstract predicate valuations!

Objective

Check whether a control path $q_{in} \xrightarrow{\mathrm{op}_0} q_1 \cdots q_k \xrightarrow{\mathrm{op}_k} q_{bad}$ is feasible

Inspection of Abstract Counterexamples

Control flow automaton: $\langle Q, q_{in}, \mathrm{X}, \rightarrow \rangle$. Set $Q_{BAD} \subseteq Q$ of bad locations.

$\mathcal{M}^a = \langle S^a, Init^a, Bad^a, \Sigma, \rightarrow^a \rangle$ obtained by predicate abstraction

Abstract counterexample

$$(q_{in}, v_0^a) \xrightarrow{\mathrm{op}_0} (q_1, v_1^a) \cdots (q_k, v_k^a) \xrightarrow{\mathrm{op}_k} (q_{bad}, v_{k+1}^a)$$

The abstract counterexample is feasible if there is a concrete run

$$(q_{in}, v_0^c) \xrightarrow{\mathrm{op}_0} (q_1, v_1^c) \cdots (q_k, v_k^c) \xrightarrow{\mathrm{op}_k} (q_{bad}, v_{k+1}^c) \quad \text{with } v_i^c \in \gamma(v_i^a)$$

Better to directly check for all possible abstract predicate valuations!

Objective

Check whether a control path $q_{in} \xrightarrow{\mathrm{op}_0} q_1 \cdots q_k \xrightarrow{\mathrm{op}_k} q_{bad}$ is feasible

Checking Feasibility of Control Paths

Feasibility at the Semantic Level

$$q_{in} \xrightarrow{\mathrm{op}_0} q_1 \cdots q_k \xrightarrow{\mathrm{op}_k} q_{bad} \text{ feasible} \quad \text{iff} \quad [\![\mathrm{op}_k]\!] \circ \cdots \circ [\![\mathrm{op}_0]\!] \neq \emptyset$$

Recall that expressions e used in guards and assignments are over X.

Syntactic Effect of Operations: Formula $\langle\!\langle \mathrm{op} \rangle\!\rangle$ over X, X'

$$\langle\!\langle g \rangle\!\rangle = g \wedge \bigwedge_{x \in \mathrm{X}} x' = x \qquad \langle\!\langle x := e \rangle\!\rangle = x' = e \wedge \bigwedge_{y \in \mathrm{X}, y \neq x} y' = y$$

For each $\mathrm{op} \in \mathrm{Op}$: $[\![\langle\!\langle \mathrm{op} \rangle\!\rangle]\!] = [\![\mathrm{op}]\!]$

Multiply-primed copies of variables: $x^{(i)}$ is the copy of x with i primes.

Feasibility at the Syntactic Level

$$q_{in} \xrightarrow{\mathrm{op}_0} q_1 \cdots q_k \xrightarrow{\mathrm{op}_k} q_{bad} \text{ feasible} \quad \text{iff} \quad \langle\!\langle \mathrm{op}_0 \rangle\!\rangle^{(0)} \wedge \cdots \wedge \langle\!\langle \mathrm{op}_k \rangle\!\rangle^{(k)} \text{ sat.}$$

Checking Feasibility of Control Paths

Feasibility at the Semantic Level

$q_{in} \xrightarrow{\mathtt{op}_0} q_1 \cdots q_k \xrightarrow{\mathtt{op}_k} q_{bad}$ feasible iff $[\![\mathtt{op}_k]\!] \circ \cdots \circ [\![\mathtt{op}_0]\!] \neq \emptyset$

Recall that expressions e used in guards and assignments are over $\mathtt{X}$.

Syntactic Effect of Operations: Formula $\langle\!\langle \mathtt{op} \rangle\!\rangle$ over $\mathtt{X}, \mathtt{X}'$

$$\langle\!\langle g \rangle\!\rangle \;=\; g \wedge \bigwedge_{x \in \mathtt{X}} x' = x \qquad \langle\!\langle x := e \rangle\!\rangle \;=\; x' = e \wedge \bigwedge_{y \in \mathtt{X}, y \neq x} y' = y$$

For each $\mathtt{op} \in \mathtt{Op}$: $[\![\langle\!\langle \mathtt{op} \rangle\!\rangle]\!] = [\![\mathtt{op}]\!]$

Multiply-primed copies of variables: $x^{(i)}$ is the copy of x with i primes.

Feasibility at the Syntactic Level

$q_{in} \xrightarrow{\mathtt{op}_0} q_1 \cdots q_k \xrightarrow{\mathtt{op}_k} q_{bad}$ feasible iff $\langle\!\langle \mathtt{op}_0 \rangle\!\rangle^{(0)} \wedge \cdots \wedge \langle\!\langle \mathtt{op}_k \rangle\!\rangle^{(k)}$ sat.

Checking Feasibility of Control Paths

Feasibility at the Syntactic Level

$q_{in} \xrightarrow{\mathrm{op}_0} q_1 \cdots q_k \xrightarrow{\mathrm{op}_k} q_{bad}$ feasible iff $\langle\!\langle \mathrm{op}_0 \rangle\!\rangle^{(0)} \wedge \cdots \wedge \langle\!\langle \mathrm{op}_k \rangle\!\rangle^{(k)}$ sat.

Number of variables grows linearly with the length of the control path.

To help the prover, we may replace $\langle\!\langle \mathrm{op} \rangle\!\rangle$ with the weakest precondition

$$\mathrm{wp}(\mathrm{op}, \varphi) = \begin{cases} g \wedge \varphi & \text{if } \mathrm{op} = g \\ \varphi[e/x] & \text{if } \mathrm{op} = x := e \end{cases}$$

Feasibility with Weakest Precondition

$q_{in} \xrightarrow{*} q_{bad}$ feasible iff $\mathrm{wp}(\mathrm{op}_0, \mathrm{wp}(\mathrm{op}_1, \ldots, \mathrm{wp}(\mathrm{op}_k, \mathit{true}) \cdots))$ sat.

But it might actually be better to rely on the prover's powerful engine!

Checking Feasibility of Control Paths

Feasibility at the Syntactic Level

$q_{in} \xrightarrow{\mathrm{op}_0} q_1 \cdots q_k \xrightarrow{\mathrm{op}_k} q_{bad}$ feasible iff $\langle\!\langle \mathrm{op}_0 \rangle\!\rangle^{(0)} \wedge \cdots \wedge \langle\!\langle \mathrm{op}_k \rangle\!\rangle^{(k)}$ sat.

Number of variables grows linearly with the length of the control path.

To help the prover, we may replace $\langle\!\langle \mathrm{op} \rangle\!\rangle$ with the weakest precondition

$$\mathrm{wp}(\mathrm{op}, \varphi) = \begin{cases} g \wedge \varphi & \text{if } \mathrm{op} = g \\ \varphi[e/x] & \text{if } \mathrm{op} = x := e \end{cases}$$

Feasibility with Weakest Precondition

$q_{in} \xrightarrow{*} q_{bad}$ feasible iff $\mathrm{wp}(\mathrm{op}_0, \mathrm{wp}(\mathrm{op}_1, \ldots, \mathrm{wp}(\mathrm{op}_k, \mathit{true}) \cdots))$ sat.

But it might actually be better to rely on the prover's powerful engine!

Refinement Challenge: Finding Relevant Predicates

Assume that the counterexample $q_{in} \xrightarrow{\mathrm{op}_0} q_1 \cdots q_k \xrightarrow{\mathrm{op}_k} q_{bad}$ is spurious

$$\langle\!\langle \mathrm{op}_0 \rangle\!\rangle^{(0)} \wedge \cdots \wedge \langle\!\langle \mathrm{op}_k \rangle\!\rangle^{(k)} \text{ unsatisfiable} \qquad (1)$$

Refinement consists in adding new predicates, but as few as possible.

Goal

Find predicates that remove the counterexample from the abstraction

Practical Approach

Some conjuncts in (1) do not "participate" in unsatisfiability.

Natural idea: try to find a small unsatisfiable subset of useful conjuncts.

For instance pick the leaves in a proof of unsatisfiability.

Might or might not work...

... Let us look back at the goal!

Refinement: Computation of Path Invariants

Consider an unfeasible control path $q_{in} \xrightarrow{\mathrm{op}_0} q_1 \cdots q_k \xrightarrow{\mathrm{op}_k} q_{bad}$.

Path Safety Invariant

Sequence $(\phi_i)_{0 \leq i \leq k+1}$ of subsets of $\mathtt{X} \to \mathbb{R}$ such that

$$\phi_0 = \mathtt{X} \to \mathbb{R} \qquad \phi_{i+1} \supseteq [\![\mathrm{op}_i]\!][\phi_i] \qquad \phi_{k+1} = \emptyset$$

Intuition

A path safety invariant gives an explanation of unfeasibility

Example: Sequence of Reachable Valuations Along the Path

$$\phi_i = [\![\mathrm{op}_{i-1}]\!] \circ \cdots \circ [\![\mathrm{op}_0]\!][\mathtt{X} \to \mathbb{R}]$$

Objective

Compute simple path safety invariants

Refinement: Path Safety Invariants from Proofs

Consider an unfeasible control path $q_{in} \xrightarrow{\mathrm{op}_0} q_1 \cdots q_k \xrightarrow{\mathrm{op}_k} q_{bad}$.

Path Safety Invariant (Syntactic Definition)

Sequence $(\varphi_i)_{0 \leq i \leq k+1}$ of formulas with free variables in X such that

$$\varphi_0 = \mathit{true} \qquad \models \varphi_i \wedge \langle\langle \mathrm{op}_i \rangle\rangle \Rightarrow \varphi_{i+1}^{(1)} \qquad \varphi_{k+1} = \mathit{false}$$

Path safety invariants can be obtained from proofs of unsatisfiability

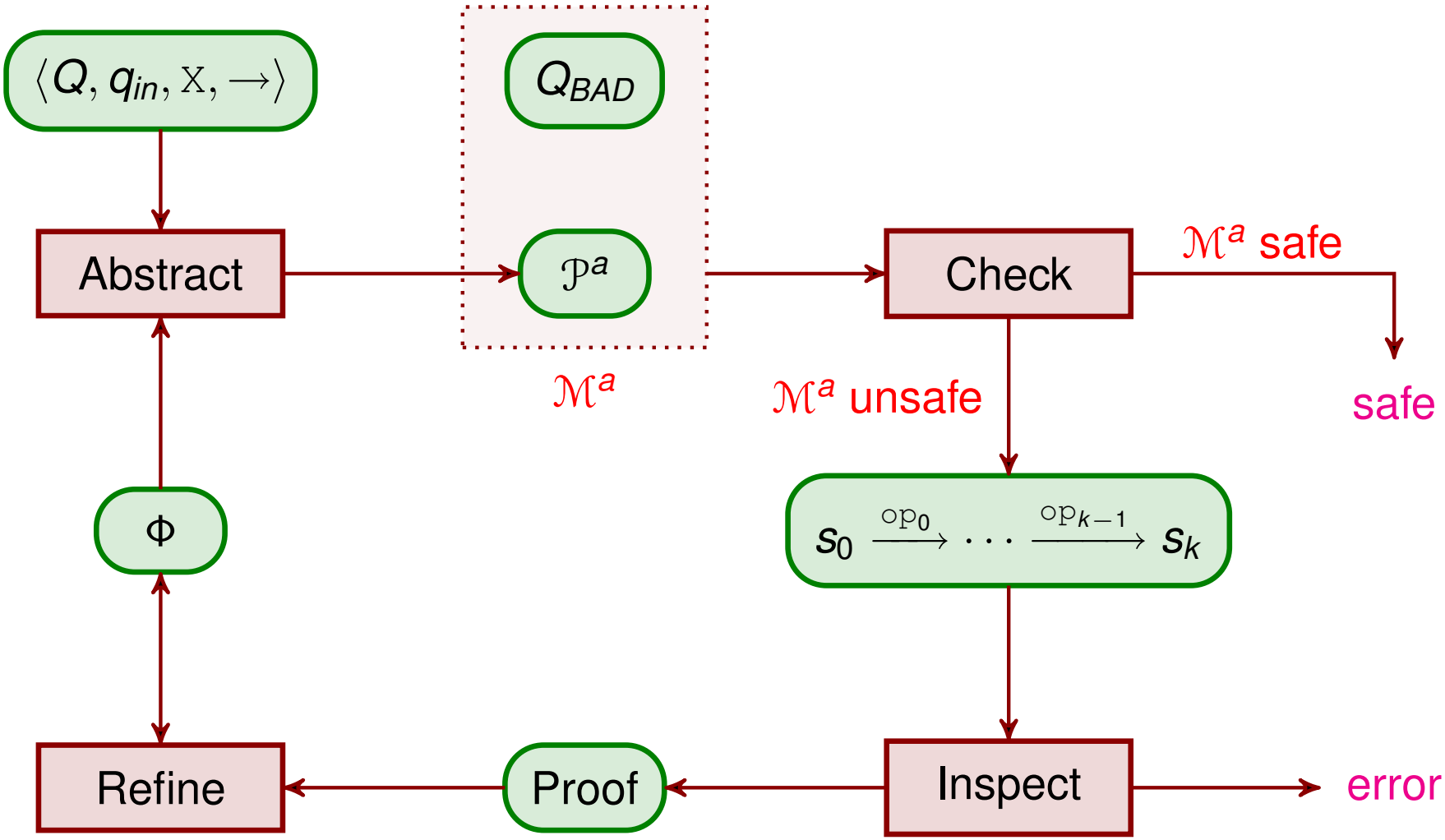
Refinement

New predicates are atomic predicates from the path invariant.

This guarantees that the counterexample will be eliminated.

No quantifier ever introduced! ☺

CounterExample-Guided Abstract model Refinement



Classical CEGAR Algorithm

CEGAR $(\mathcal{P} = \langle Q, q_{in}, \mathrm{X}, \rightarrow \rangle, Q_{BAD}, \Phi_0)$

$\Phi \leftarrow \Phi_0$

while (true)

$\mathcal{M}^a \leftarrow$ PredicateAbstraction $(\langle Q, q_{in}, \mathrm{X}, \rightarrow \rangle, Q_{BAD}, \Phi)$

check $\leftarrow$ ModelCheck $(\mathcal{M}^a)$

if check is $\mathcal{M}^a$ safe

return $\mathcal{P}$ safe

// *check is* $(q_{in}, v_0^a) \xrightarrow{\mathrm{op}_0} (q_1, v_1^a) \cdots (q_k, v_k^a) \xrightarrow{\mathrm{op}_k} (q_{bad}, v_{k+1}^a)$

insp $\leftarrow$ Inspect $(q_{in} \xrightarrow{\mathrm{op}_0} q_1 \cdots q_k \xrightarrow{\mathrm{op}_k} q_{bad})$

if insp is feasible

return $q_{in} \xrightarrow{\mathrm{op}_0} q_1 \cdots q_k \xrightarrow{\mathrm{op}_k} q_{bad}$ feasible

// *insp is unfeasible*

construct a path invariant and extract new predicates Φ' from it

$\Phi \leftarrow \Phi \cup \Phi'$

Drawbacks of the Classical CEGAR Algorithm

Batch-oriented integration

No sharing of data structures

No reuse of previous computations

Re-explores the same error-free parts of the configuration space

Abstraction fully computed before the model-checking phase

Useless expensive work

Some Variants of the Classical CEGAR Algorithm

Lazy CEGAR

Integrated CEGAR loop driven by the model-checker

Builds a reachability tree with abstract successors on demand

- Nodes labeled by support predicates
- Refinement only locally refines subparts of the tree

Lazy Interpolation

Builds a reachability tree with no abstract successor computation

Uses interpolation to:

- rule out each spurious control path
- label counterexample paths in the tree with path invariants

Part VII

Conclusions

Outline — Conclusions

Outline — Conclusions

Summary: Abstract Model Refinement

Fully automatic software verification technique based on model-checking and refinement of finite-state abstractions

Property-Preserving Abstraction

Conservatively extract finite-state models from programs

Inspection of Abstract Counterexamples

Reduces to satisfiability checking

Refinement Guided by Abstract Counterexamples

Based on the construction of path invariants

New predicates obtained from proofs of unsatisfiability

Each of these three phases relies on theorem provers

Outline — Conclusions

Some CEGAR-based Software Verification Tools

SLAM — Thomas Ball, Sriram Rajamani, . . .

Analysis of programs written in C

- ☹ Classical batch-oriented CEGAR algorithm
- ☺ Interprocedural analysis (abstraction into boolean programs)

Now integrated in Static Driver Verifier, part of the Windows Driver Kit

BLAST — Thomas Henzinger, . . .

Analysis of programs written in C

- ☺ Lazy CEGAR algorithm
- ☹ Bounded-recursion interprocedural analysis

Open source, distributed under the BSD license

MAGIC, YASM, . . .

Application: Verification of Device Drivers

Why device drivers?

High Impact

Bugs lead to system crash (e.g. BSOD)

Developed by third-party vendors

Not So Complex

Simple safety properties (e.g. locking discipline)

Only a small part of the code is relevant to the properties

Medium-sized ($\leq$ 25 000 lines)

Outline — Conclusions

Static Analysis and Abstraction Refinement

Verification of software: computation of strong enough invariants

Abstraction Process

Interpret programs according to a simplified, “abstract” semantics.

Property-Preserving Abstraction

Formally relate the “abstract” semantics with the “standard” semantics, so as to preserve relevant properties.

Main challenge: suitable refinement of abstractions

Static Analysis versus Abstraction Refinement

Static Analysis

- ☺ Always terminates
- ☹ False positives
- ☹ Manual refinement
- ☺ Infinite domains
- ☹ Same precision everywhere

Abstraction Refinement

- ☹ May not terminate
- ☺ Definite answer (yes / no)
- ☺ Automatic refinement
- ☹ Finite abstract domains
- ☺ Adaptive precision
- ☺ Driven by the property

Inspection & Refinement

Smart mind

Inspection & Refinement

Smart prover

Static Analysis versus Abstraction Refinement

Static Analysis

- ☺ Always terminates
- ☹ False positives
- ☹ Manual refinement
- ☺ Infinite domains
- ☹ Same precision everywhere

Abstraction Refinement

- ☹ May not terminate
- ☺ Definite answer (yes / no)
- ☺ Automatic refinement
- ☹ Finite abstract domains
- ☺ Adaptive precision
- ☺ Driven by the property

Inspection & Refinement

Smart mind

Inspection & Refinement

Smart prover

Extensions and Remaining Challenges

Not Covered in the Lecture

Computational Models

- Pointer analysis, arrays
- Recursion, threads
- Hybrid systems, ...

Beyond Safety

- Termination
- Liveness properties
- μ-calculus (Modal LKS)

Software Verification remains a challenging problem!

Room for Improvement

- Generation of smart predicates for refinement
- Path invariants for control paths with loops

Outline — Conclusions

Some References

S. Graf and H. Saïdi.
Construction of abstract state graphs with PVS.
In *Proc. 9th Int. Conf. Computer Aided Verification, Haifa, Israel*, LNCS 1254, pages 72–83. Springer, 1997.

E. Clarke, O. Grumberg, S. Jha, Y. Lu, H. Veith.
Counterexample-guided abstraction refinement.
In *Proc. 12th Int. Conf. Computer Aided Verification, Haifa, Israel*, LNCS 1855, pages 154–169. Springer, 2000.

- The SLAM Project.
  `http://research.microsoft.com/slam/`
- The Berkeley Lazy Abstraction Software Verification Tool.
  `http://mtc.epfl.ch/software-tools/blast/`

Thank you!