Inferring Definitive Counterexamples Through Under-Approximation

Jörg Brauer (RWTH Aachen, Germany)
Axel Simon (TU Munich, Germany)

Motivation

Motivation

- proof absence of bugs using static analysis

Motivation

- proof absence of bugs using static analysis
- recover from precision loss due to over-approximating domains

Counterexamples

Counterexamples

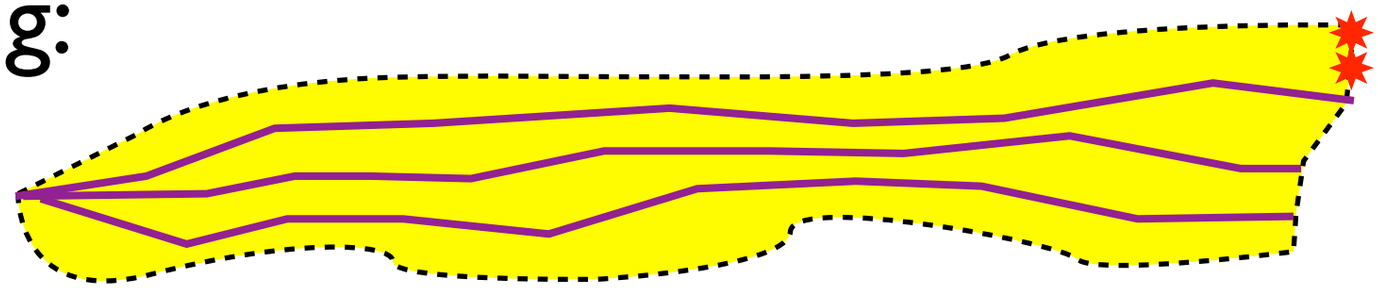
- model checking yields counter examples
  - considered invaluable for fixing defects
    - software model checking: model is an abstraction; is counterexample spurious?

Counterexamples

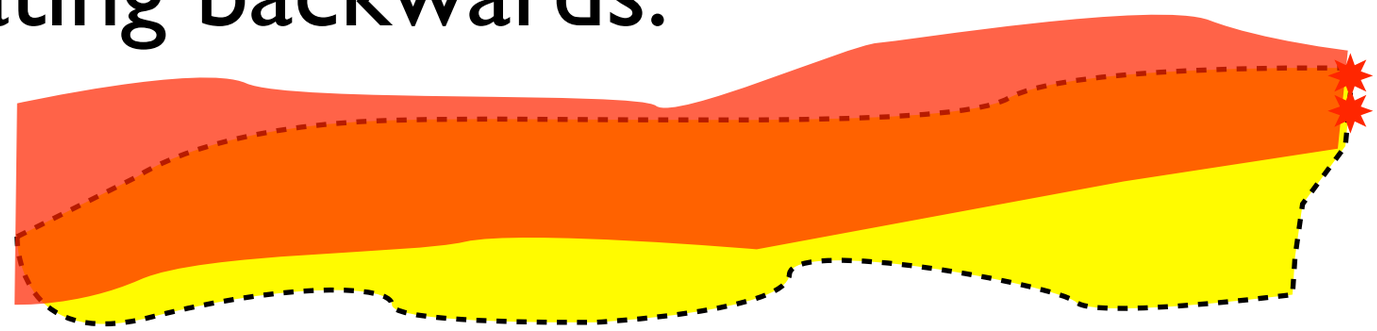
- model checking yields counter examples
  - considered invaluable for fixing defects
    - software model checking: model is an abstraction; is counterexample spurious?
- abstract interpretation merges traces
  - a warning may always be spurious

Revert Merge of Traces

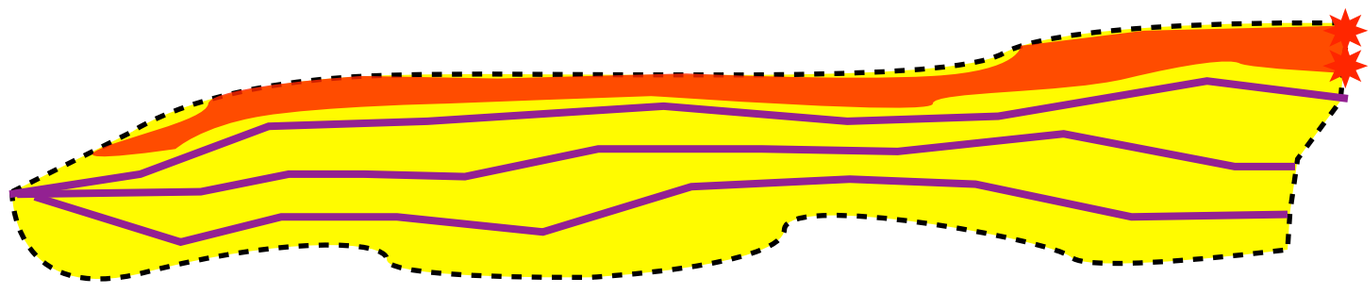
- Spurious warning:



- Over-approximating backwards:



- Under-approximating backwards:



Difficulty of Backward Reasoning

Difficulty of Backward Reasoning

- Need to solve abduction problem:
  - Given B and C, compute A s.th. $A \wedge B \vDash C$
  - A is called pseudo-complement of B relative to C

Difficulty of Backward Reasoning

- Need to solve abduction problem:
  - Given B and C, compute A s.th. $A \wedge B \vdash C$
  - A is called pseudo-complement of B relative to C
- Few domains allow to allow this precisely
  - any domain closed under negation
  - standard numeric domains: no (convexity)

Domains for Backward Reasoning

- Idea: use Domain of Boolean functions
- Express program semantics at the bit-level
- use a relational (input/output) semantics
- `XOR a b` gives $\bigwedge_{i=0}^{31}(\mathbf{a}'[i] \leftrightarrow \mathbf{a}[i] \oplus \mathbf{b}[i])$
- given a post-state `s'` over `a'`, calculate a pre-state `s` over `a`, `b` using SAT-solving

Example

Example

- Suppose an abstract interpreter finds assertion ψ violated in post-state `s'` starting from `s`

Example

- Suppose an abstract interpreter finds assertion ψ violated in post-state s' starting from s
- let φ encode transformer: $s' = \varphi \wedge s$

Example

- Suppose an abstract interpreter finds assertion ψ violated in post-state s' starting from s
- let φ encode transformer: $s' = \varphi \wedge s$
- then $\varphi \wedge s' \wedge \neg\psi$ describe those states in s that violate the assertions

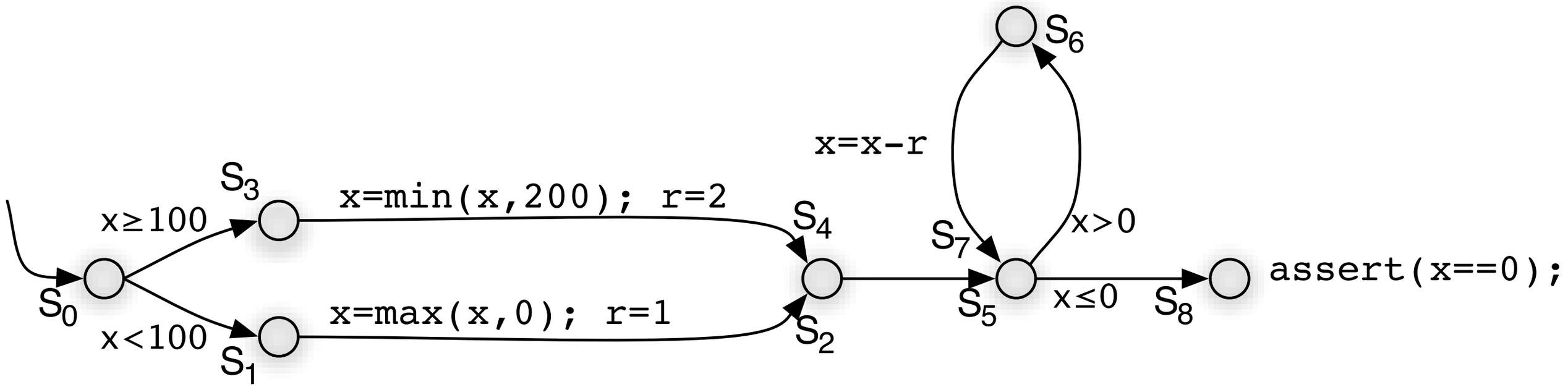
Example

- Suppose an abstract interpreter finds assertion ψ violated in post-state s' starting from s
- let φ encode transformer: $s' = \varphi \wedge s$
- then $\varphi \wedge s' \wedge \neg\psi$ describe those states in s that violate the assertions
- idea: project $\varphi \wedge s' \wedge \neg\psi$ onto variables in s

Example

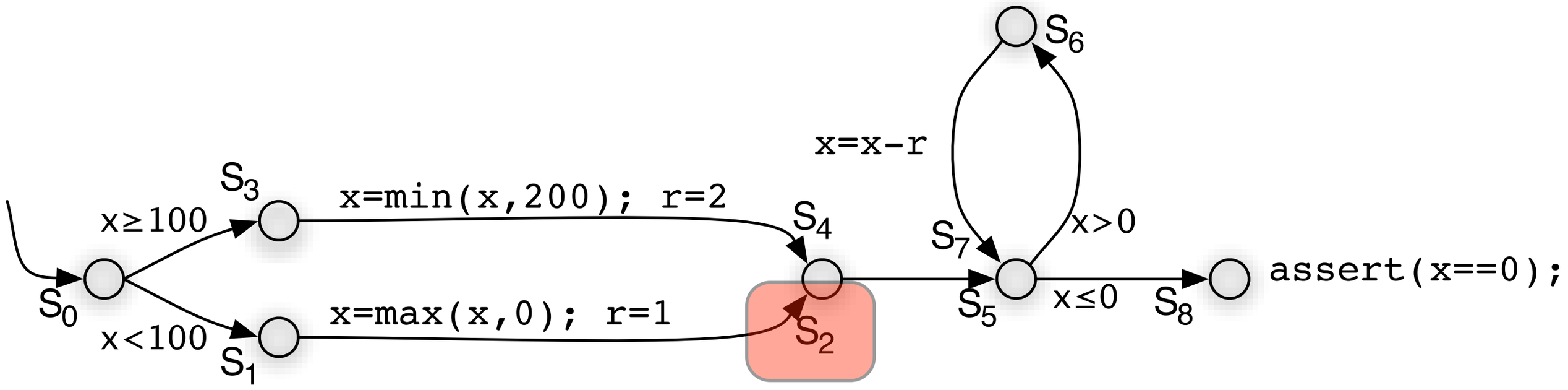
- Suppose an abstract interpreter finds assertion ψ violated in post-state s' starting from s
- let φ encode transformer: $s' = \varphi \wedge s$
- then $\varphi \wedge s' \wedge \neg\psi$ describe those states in s that violate the assertions
- idea: project $\varphi \wedge s' \wedge \neg\psi$ onto variables in s
- φ only represents one step: repeat!

Analysis of Loops



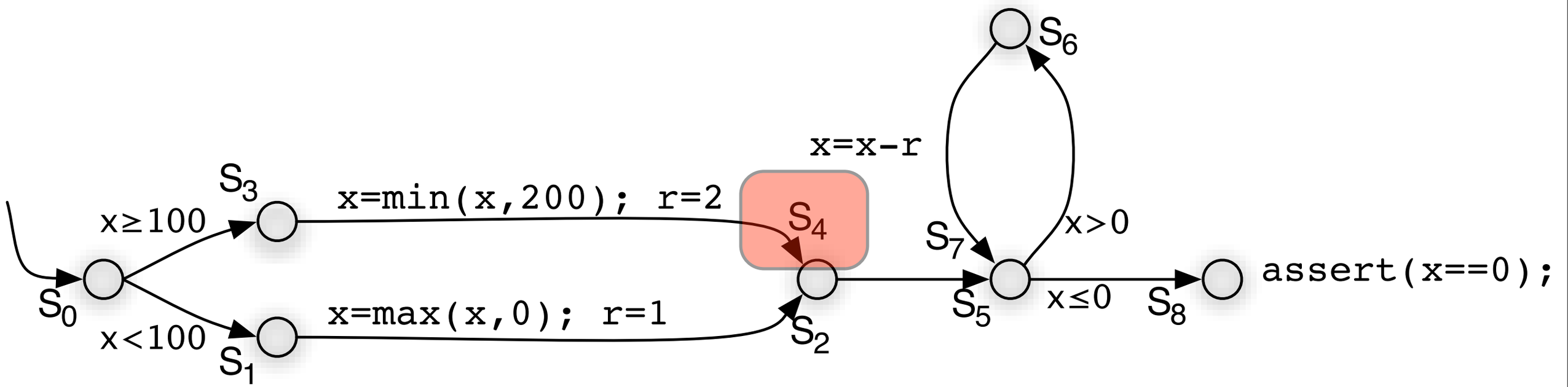
i	0	1	2	3	4	5	6	7	8
$S_i = \begin{cases} x \\ r \end{cases}$	any any	≤ 99 any	$[0,99]$ $[1,1]$	≥ 100 any	$[100,200]$ $[2,2]$	$[0,200]$ $[1,2]$	$[1,200]$ $[1,2]$	$[-1,199]$ $[1,2]$	$[-1,0]$ $[1,2]$
$S'_i = \begin{cases} x \\ r \end{cases}$			$\bot$		$\bot$	1 2	$[0,1]$ $[1,2]$	-1 $[1,2]$	-1 $[1,2]$
$S''_i = \begin{cases} x \\ r \end{cases}$	$[101,125]$ 2		$\bot$	$[101,125]$ 2	$[101,125]$ 2	$[1,125]$ 2			

Analysis of Loops



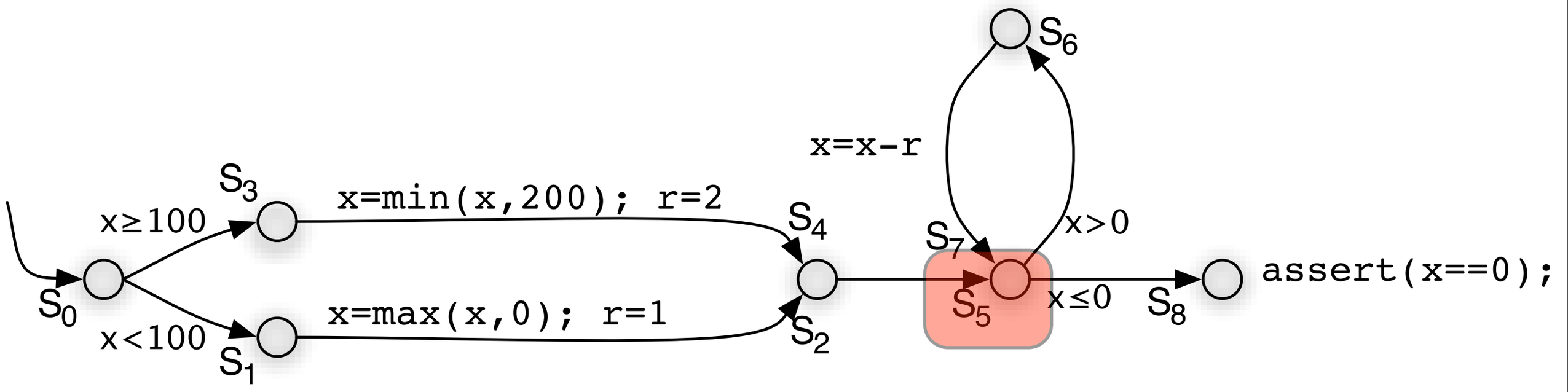
i	0	1	2	3	4	5	6	7	8
$S_i = \begin{cases} x \\ r \end{cases}$	any any	≤ 99 any	$[0,99]$ $[1,1]$	≥ 100 any	$[100,200]$ $[2,2]$	$[0,200]$ $[1,2]$	$[1,200]$ $[1,2]$	$[-1,199]$ $[1,2]$	$[-1,0]$ $[1,2]$
$S'_i = \begin{cases} x \\ r \end{cases}$			$\bot$		$\bot$	1 2	$[0,1]$ $[1,2]$	-1 $[1,2]$	-1 $[1,2]$
$S''_i = \begin{cases} x \\ r \end{cases}$	$[101,125]$ 2		$\bot$	$[101,125]$ 2	$[101,125]$ 2	$[1,125]$ 2			

Analysis of Loops



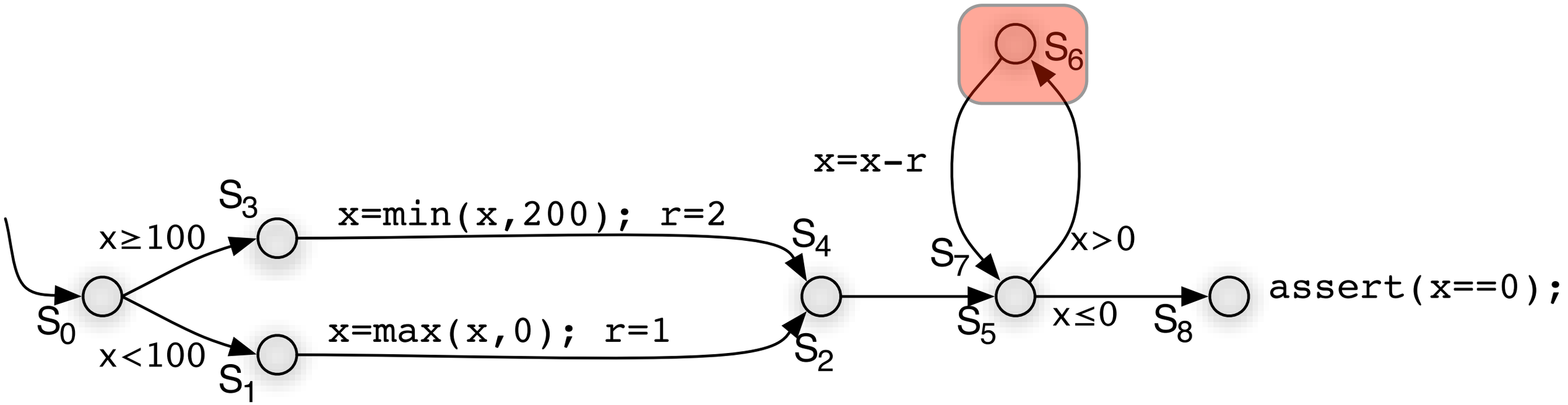
i	0	1	2	3	4	5	6	7	8
$S_i = \begin{cases} x \\ r \end{cases}$	any any	≤ 99 any	$[0,99]$ $[1,1]$	≥ 100 any	$[100,200]$ $[2,2]$	$[0,200]$ $[1,2]$	$[1,200]$ $[1,2]$	$[-1,199]$ $[1,2]$	$[-1,0]$ $[1,2]$
$S'_i = \begin{cases} x \\ r \end{cases}$			$\bot$		$\bot$	1 2	$[0,1]$ $[1,2]$	-1 $[1,2]$	-1 $[1,2]$
$S''_i = \begin{cases} x \\ r \end{cases}$	$[101,125]$ 2		$\bot$	$[101,125]$ 2	$[101,125]$ 2	$[1,125]$ 2			

Analysis of Loops



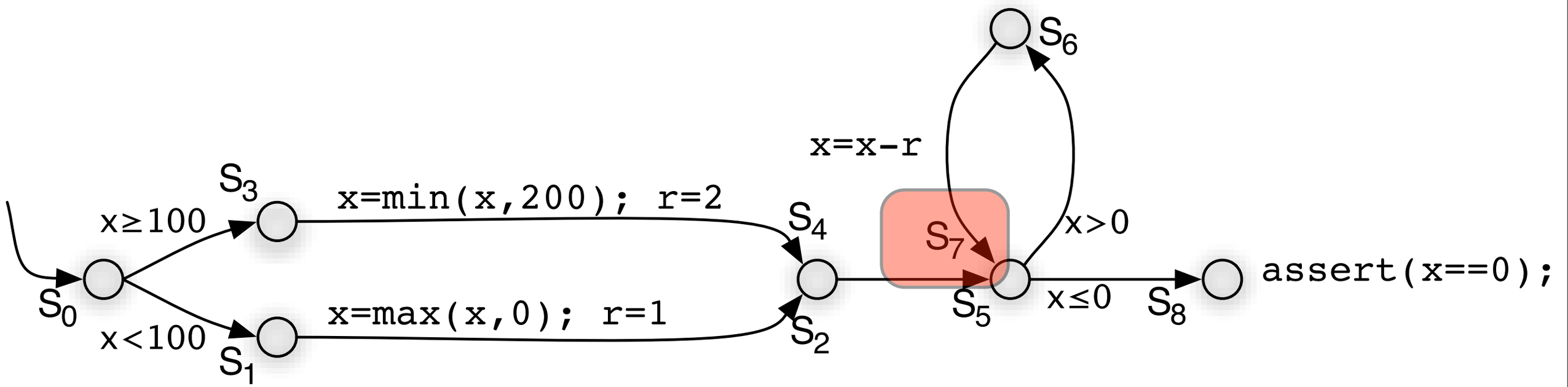
i	0	1	2	3	4	5	6	7	8
$S_i = \begin{cases} x \\ r \end{cases}$	any any	≤ 99 any	$[0,99]$ $[1,1]$	≥ 100 any	$[100,200]$ $[2,2]$	$[0,200]$ $[1,2]$	$[1,200]$ $[1,2]$	$[-1,199]$ $[1,2]$	$[-1,0]$ $[1,2]$
$S'_i = \begin{cases} x \\ r \end{cases}$			$\bot$		$\bot$	1 2	$[0,1]$ $[1,2]$	-1 $[1,2]$	-1 $[1,2]$
$S''_i = \begin{cases} x \\ r \end{cases}$	$[101,125]$ 2		$\bot$	$[101,125]$ 2	$[101,125]$ 2	$[1,125]$ 2			

Analysis of Loops



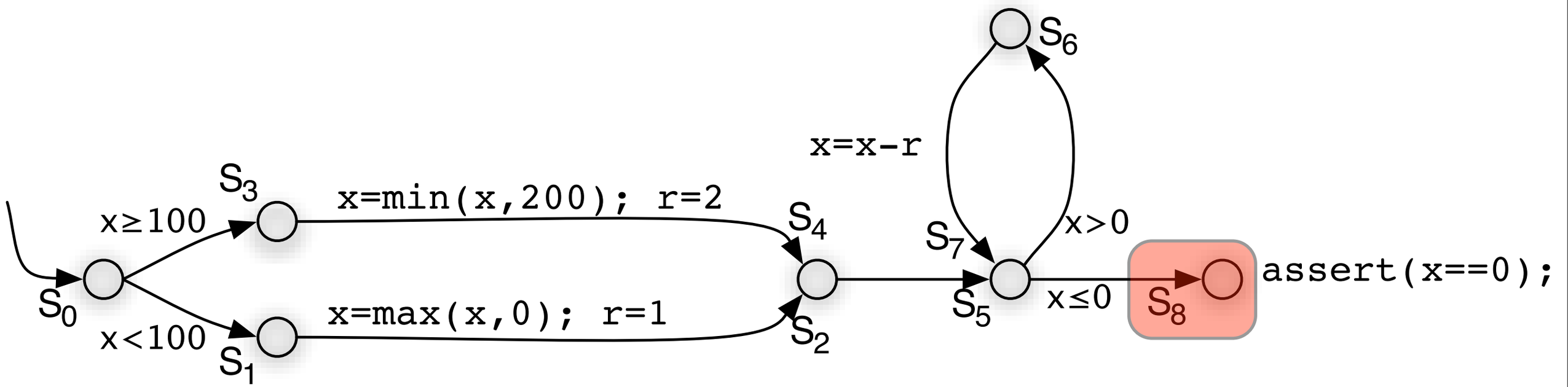
i	0	1	2	3	4	5	6	7	8
$S_i = \begin{cases} x \\ r \end{cases}$	any any	≤ 99 any	$[0,99]$ $[1,1]$	≥ 100 any	$[100,200]$ $[2,2]$	$[0,200]$ $[1,2]$	$[1,200]$ $[1,2]$	$[-1,199]$ $[1,2]$	$[-1,0]$ $[1,2]$
$S'_i = \begin{cases} x \\ r \end{cases}$			$\bot$		$\bot$	1 2	$[0,1]$ $[1,2]$	-1 $[1,2]$	-1 $[1,2]$
$S''_i = \begin{cases} x \\ r \end{cases}$	$[101,125]$ 2		$\bot$	$[101,125]$ 2	$[101,125]$ 2	$[1,125]$ 2			

Analysis of Loops



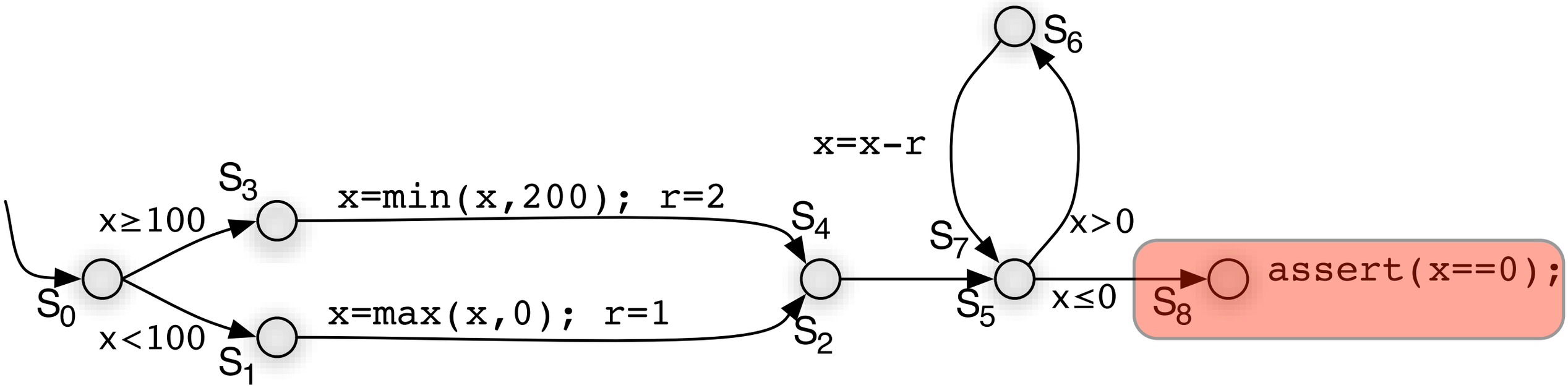
i	0	1	2	3	4	5	6	7	8
$S_i = \begin{cases} x \\ r \end{cases}$	any any	≤ 99 any	$[0,99]$ $[1,1]$	≥ 100 any	$[100,200]$ $[2,2]$	$[0,200]$ $[1,2]$	$[1,200]$ $[1,2]$	$[-1,199]$ $[1,2]$	$[-1,0]$ $[1,2]$
$S'_i = \begin{cases} x \\ r \end{cases}$			$\bot$		$\bot$	1 2	$[0,1]$ $[1,2]$	-1 $[1,2]$	-1 $[1,2]$
$S''_i = \begin{cases} x \\ r \end{cases}$	$[101,125]$ 2		$\bot$	$[101,125]$ 2	$[101,125]$ 2	$[1,125]$ 2			

Analysis of Loops



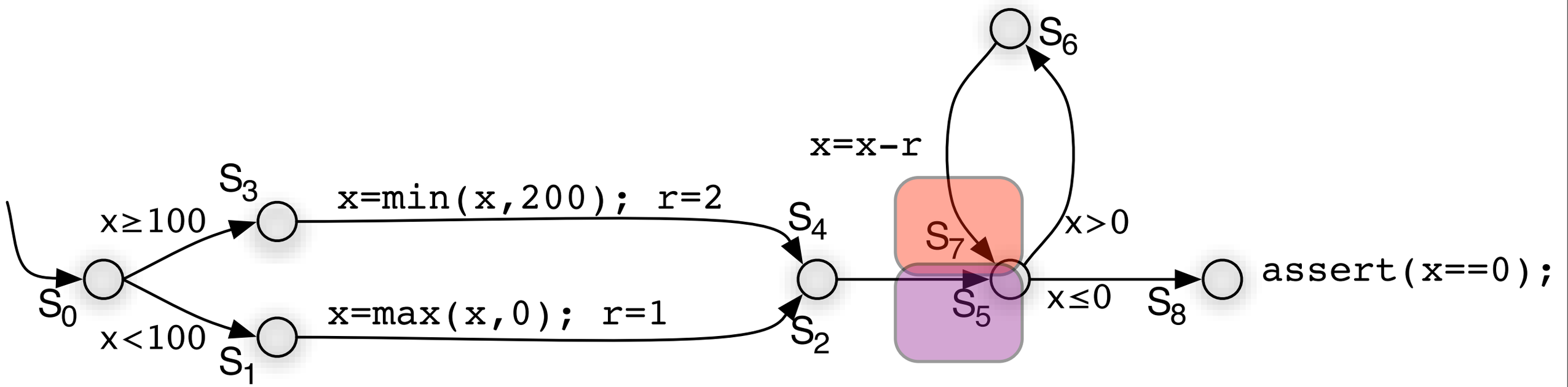
i	0	1	2	3	4	5	6	7	8
$S_i = \begin{cases} x \\ r \end{cases}$	any any	≤ 99 any	$[0,99]$ $[1,1]$	≥ 100 any	$[100,200]$ $[2,2]$	$[0,200]$ $[1,2]$	$[1,200]$ $[1,2]$	$[-1,199]$ $[1,2]$	$[-1,0]$ $[1,2]$
$S'_i = \begin{cases} x \\ r \end{cases}$			$\bot$		$\bot$	1 2	$[0,1]$ $[1,2]$	-1 $[1,2]$	-1 $[1,2]$
$S''_i = \begin{cases} x \\ r \end{cases}$	$[101,125]$ 2		$\bot$	$[101,125]$ 2	$[101,125]$ 2	$[1,125]$ 2			

Analysis of Loops



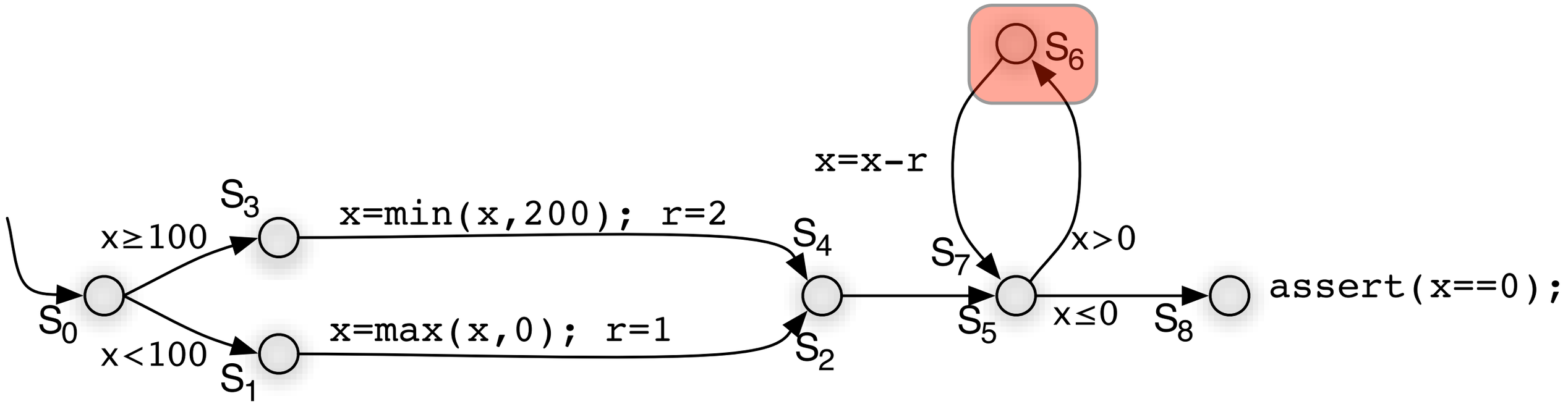
i	0	1	2	3	4	5	6	7	8
$S_i = \begin{cases} x \\ r \end{cases}$	any any	≤ 99 any	$[0,99]$ $[1,1]$	≥ 100 any	$[100,200]$ $[2,2]$	$[0,200]$ $[1,2]$	$[1,200]$ $[1,2]$	$[-1,199]$ $[1,2]$	$[-1,0]$ $[1,2]$
$S'_i = \begin{cases} x \\ r \end{cases}$			$\bot$		$\bot$	1 2	$[0,1]$ $[1,2]$	-1 $[1,2]$	-1 $[1,2]$
$S''_i = \begin{cases} x \\ r \end{cases}$	$[101,125]$ 2		$\bot$	$[101,125]$ 2	$[101,125]$ 2	$[1,125]$ 2			

Analysis of Loops



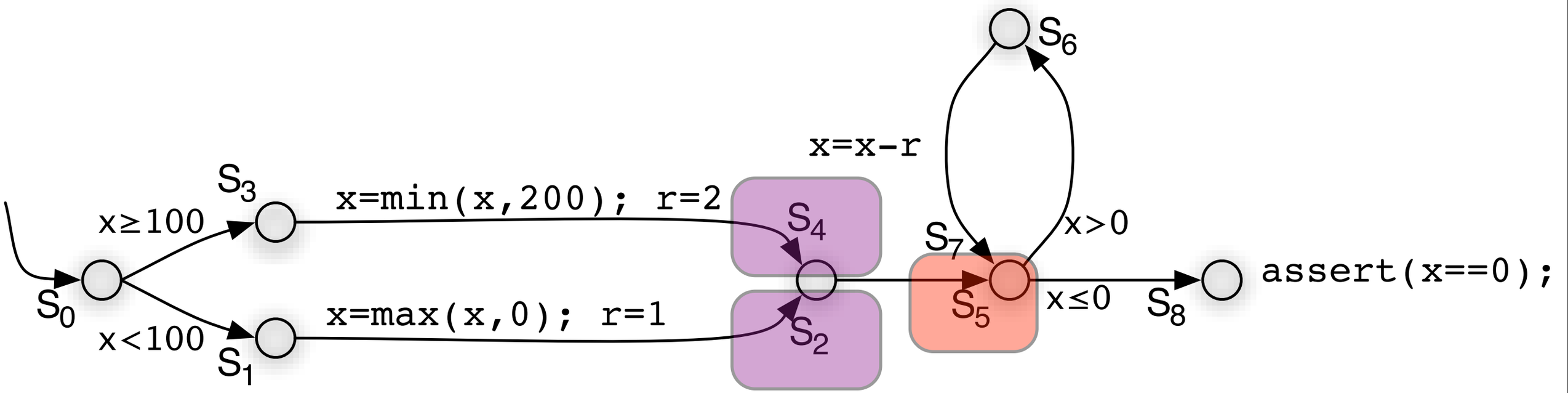
i	0	1	2	3	4	5	6	7	8
$S_i = \begin{cases} x \\ r \end{cases}$	any any	≤ 99 any	$[0,99]$ $[1,1]$	≥ 100 any	$[100,200]$ $[2,2]$	$[0,200]$ $[1,2]$	$[1,200]$ $[1,2]$	$[-1,199]$ $[1,2]$	$[-1,0]$ $[1,2]$
$S'_i = \begin{cases} x \\ r \end{cases}$			$\bot$		$\bot$	1 2	$[0,1]$ $[1,2]$	-1 $[1,2]$	-1 $[1,2]$
$S''_i = \begin{cases} x \\ r \end{cases}$	$[101,125]$ 2		$\bot$	$[101,125]$ 2	$[101,125]$ 2	$[1,125]$ 2			

Analysis of Loops



i	0	1	2	3	4	5	6	7	8
$S_i = \begin{cases} x \\ r \end{cases}$	any any	≤ 99 any	$[0,99]$ $[1,1]$	≥ 100 any	$[100,200]$ $[2,2]$	$[0,200]$ $[1,2]$	$[1,200]$ $[1,2]$	$[-1,199]$ $[1,2]$	$[-1,0]$ $[1,2]$
$S'_i = \begin{cases} x \\ r \end{cases}$			$\bot$		$\bot$	1 2	$[0,1]$ $[1,2]$	-1 $[1,2]$	-1 $[1,2]$
$S''_i = \begin{cases} x \\ r \end{cases}$	$[101,125]$ 2		$\bot$	$[101,125]$ 2	$[101,125]$ 2	$[1,125]$ 2			

Analysis of Loops



i	0	1	2	3	4	5	6	7	8
$S_i = \begin{cases} x \\ r \end{cases}$	any any	≤ 99 any	$[0,99]$ $[1,1]$	≥ 100 any	$[100,200]$ $[2,2]$	$[0,200]$ $[1,2]$	$[1,200]$ $[1,2]$	$[-1,199]$ $[1,2]$	$[-1,0]$ $[1,2]$
$S'_i = \begin{cases} x \\ r \end{cases}$			$\bot$		$\bot$	1 2	$[0,1]$ $[1,2]$	-1 $[1,2]$	-1 $[1,2]$
$S''_i = \begin{cases} x \\ r \end{cases}$	$[101,125]$ 2		$\bot$	$[101,125]$ 2	$[101,125]$ 2	$[1,125]$ 2			

Loop Transformers

Loop Transformers

- Let $f \equiv [\![x' = x - r,\ r' = r]\!]$ denote Boolean formulae of loop transformer

Loop Transformers

- Let $f \equiv [\![x' = x - r,\ r' = r]\!]$ denote Boolean formulae of loop transformer
- pre-calculate $f^2 = f \cdot f$, $f^4 = f^2 \cdot f^2$,...

Loop Transformers

- Let $f \equiv [\![x' = x - r,\ r' = r]\!]$ denote Boolean formulae of loop transformer
- **pre-calculate** $f^2 = f \cdot f$, $f^4 = f^2 \cdot f^2$,...
- **calculate** $\varphi_{i+1} = \varphi_i \vee \varphi_i \cdot f^{2^i}$ **where** $\varphi_0 = id$

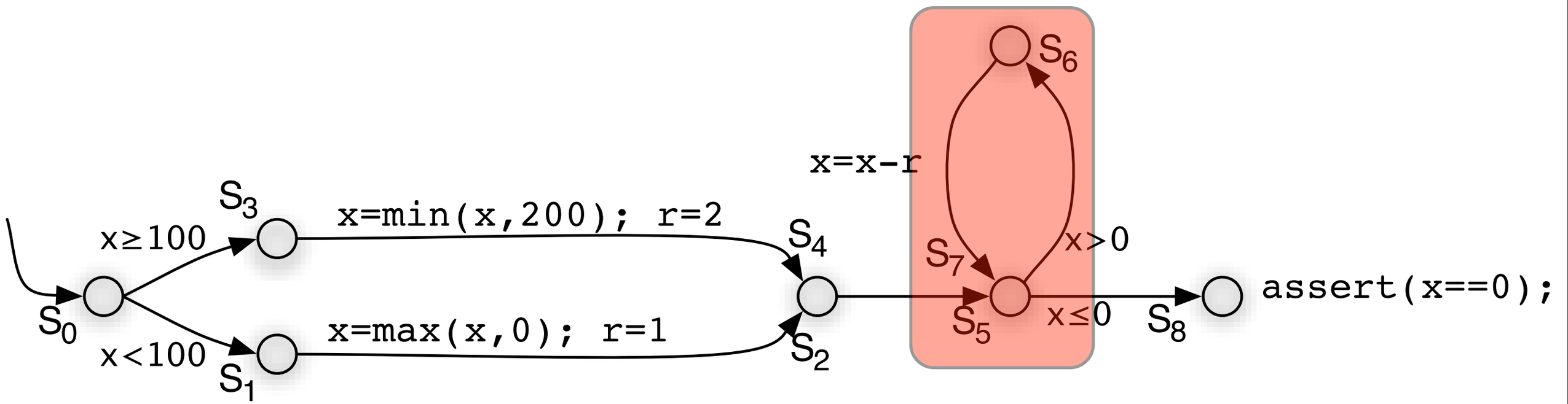
Loop Transformers

- Let $f \equiv [\![x' = x - r,\ r' = r]\!]$ denote Boolean formulae of loop transformer
- **pre-calculate** $f^2 = f \cdot f$, $f^4 = f^2 \cdot f^2$,...
- **calculate** $\varphi_{i+1} = \varphi_i \vee \varphi_i \cdot f^{2^i}$ **where** $\varphi_0 = id$
- in the example, compose loop 2^5 times:

Loop Transformers

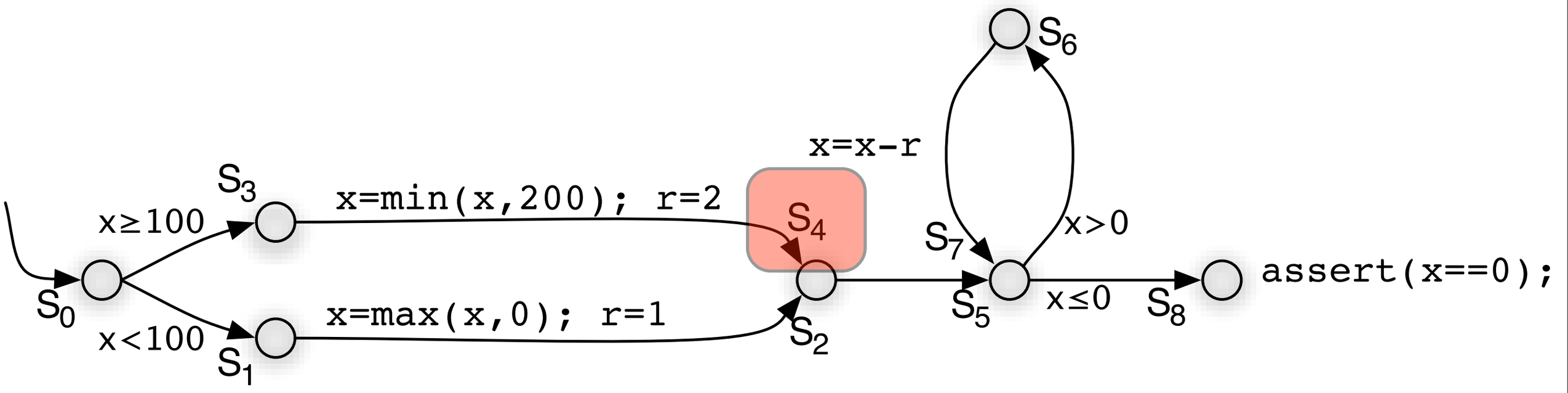
- Let $f \equiv [\![x' = x - r,\ r' = r]\!]$ denote Boolean formulae of loop transformer
- pre-calculate $f^2 = f \cdot f$, $f^4 = f^2 \cdot f^2$,...
- calculate $\varphi_{i+1} = \varphi_i \vee \varphi_i \cdot f^{2^i}$ where $\varphi_0 = id$
- in the example, compose loop 2^5 times:
  - $\varphi_5(S_5) = [\![x = 2n-1, n \in [1,63], r = 2]\!]$

Analysis of Loops



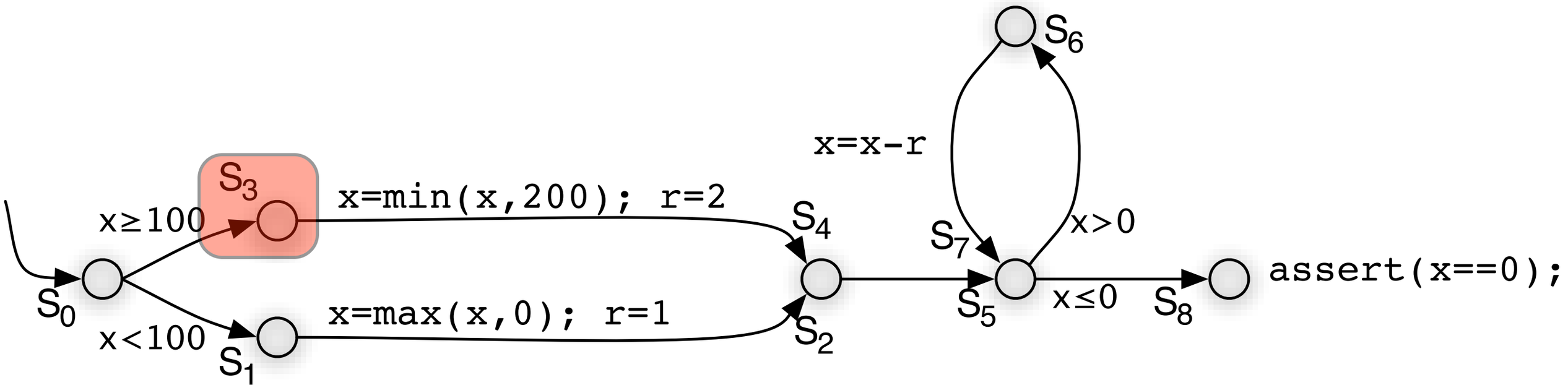
i	0	1	2	3	4	5	6	7	8
$S_i = \begin{cases} x \\ r \end{cases}$	$\begin{matrix} \text{any} \\ \text{any} \end{matrix}$	$\begin{matrix} \leq 99 \\ \text{any} \end{matrix}$	$\begin{matrix} [0,99] \\ [1,1] \end{matrix}$	$\begin{matrix} \geq 100 \\ \text{any} \end{matrix}$	$\begin{matrix} [100,200] \\ [2,2] \end{matrix}$	$\begin{matrix} [0,200] \\ [1,2] \end{matrix}$	$\begin{matrix} [1,200] \\ [1,2] \end{matrix}$	$\begin{matrix} [-1,199] \\ [1,2] \end{matrix}$	$\begin{matrix} [-1,0] \\ [1,2] \end{matrix}$
$S'_i = \begin{cases} x \\ r \end{cases}$			$\bot$		$\bot$	$\begin{matrix} 1 \\ 2 \end{matrix}$	$\begin{matrix} [0,1] \\ [1,2] \end{matrix}$	$\begin{matrix} -1 \\ [1,2] \end{matrix}$	$\begin{matrix} -1 \\ [1,2] \end{matrix}$
$S''_i = \begin{cases} x \\ r \end{cases}$	$\begin{matrix} [101,125] \\ 2 \end{matrix}$		$\bot$	$\begin{matrix} [101,125] \\ 2 \end{matrix}$	$\begin{matrix} [101,125] \\ 2 \end{matrix}$	$\begin{matrix} [1,125] \\ 2 \end{matrix}$			

Analysis of Loops



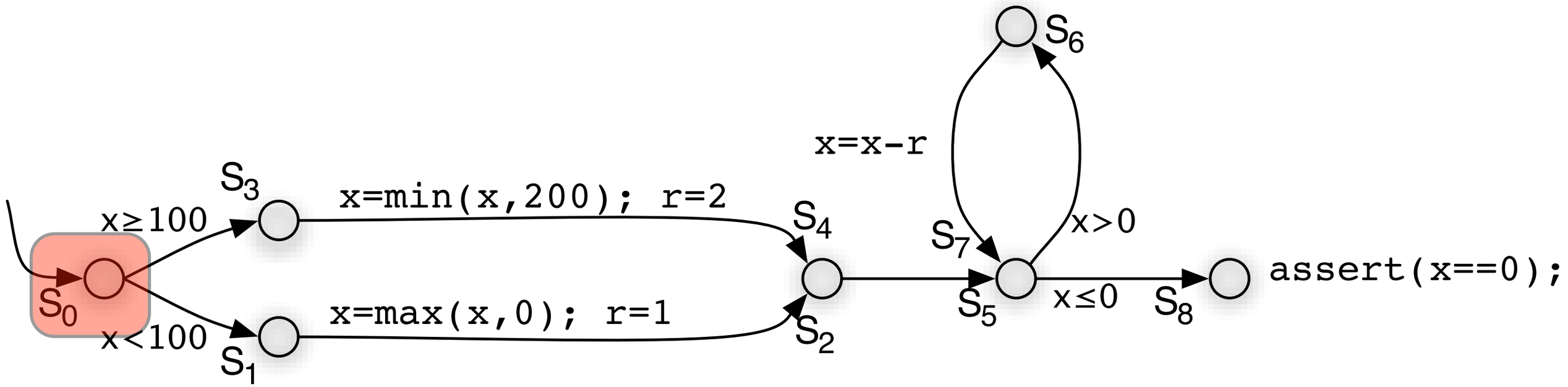
i	0	1	2	3	4	5	6	7	8
$S_i = \begin{cases} x \\ r \end{cases}$	any any	≤ 99 any	$[0,99]$ $[1,1]$	≥ 100 any	$[100,200]$ $[2,2]$	$[0,200]$ $[1,2]$	$[1,200]$ $[1,2]$	$[-1,199]$ $[1,2]$	$[-1,0]$ $[1,2]$
$S'_i = \begin{cases} x \\ r \end{cases}$			$\bot$		$\bot$	1 2	$[0,1]$ $[1,2]$	-1 $[1,2]$	-1 $[1,2]$
$S''_i = \begin{cases} x \\ r \end{cases}$	$[101,125]$ 2		$\bot$	$[101,125]$ 2	$[101,125]$ 2	$[1,125]$ 2			

Analysis of Loops



i	0	1	2	3	4	5	6	7	8
$S_i = \begin{cases} x \\ r \end{cases}$	any any	≤ 99 any	$[0,99]$ $[1,1]$	≥ 100 any	$[100,200]$ $[2,2]$	$[0,200]$ $[1,2]$	$[1,200]$ $[1,2]$	$[-1,199]$ $[1,2]$	$[-1,0]$ $[1,2]$
$S'_i = \begin{cases} x \\ r \end{cases}$			$\bot$		$\bot$	1 2	$[0,1]$ $[1,2]$	-1 $[1,2]$	-1 $[1,2]$
$S''_i = \begin{cases} x \\ r \end{cases}$	$[101,125]$ 2		$\bot$	$[101,125]$ 2	$[101,125]$ 2	$[1,125]$ 2			

Analysis of Loops



i	0	1	2	3	4	5	6	7	8
$S_i = \begin{cases} x \\ r \end{cases}$	any any	≤ 99 any	$[0,99]$ $[1,1]$	≥ 100 any	$[100,200]$ $[2,2]$	$[0,200]$ $[1,2]$	$[1,200]$ $[1,2]$	$[-1,199]$ $[1,2]$	$[-1,0]$ $[1,2]$
$S'_i = \begin{cases} x \\ r \end{cases}$			$\bot$		$\bot$	1 2	$[0,1]$ $[1,2]$	-1 $[1,2]$	-1 $[1,2]$
$S''_i = \begin{cases} x \\ r \end{cases}$	$[101,125]$ 2		$\bot$	$[101,125]$ 2	$[101,125]$ 2	$[1,125]$ 2			

Strategy for Loops

Strategy for Loops

- whenever backwards propagation fails at p
  - for each loop between assertion and p
    - double the number of unrollings
    - until backward propagation succeeds

Strategy for Loops

- whenever backwards propagation fails at p
  - for each loop between assertion and p
    - double the number of unrollings
    - until backward propagation succeeds
- cheaper: infer minimal no of unrollings

Minimal no. of Unrollings

Minimal no. of Unrollings

- many loops have a counter: $f \equiv [\![i' = i + 1]\!]$

Minimal no. of Unrollings

- many loops have a counter: $f \equiv [\![i' = i+1]\!]$
- infer affine transformations in f

Minimal no. of Unrollings

- many loops have a counter: $f \equiv [\![i' = i+1]\!]$
- infer affine transformations in f

- if a loop must be unrolled at least 2^i times:
  - calculate $\varphi_{i+1} = \varphi_i \cdot f^{2^i}$
  - instead of $\varphi_{i+1} = \varphi_i \vee \varphi_i \cdot f^{2^i}$

Remove Spurious Warnings

- there exists an `i` such that $\varphi_{i+1} \not\vdash \varphi_i$
- in this state, the warning is spurious:

```
 1  unsigned int log2(unsigned char c) {
 2          unsigned char i = 0;
 3          if (c==0) return 0; else c--;
 4          while (c > 0) {
 5                  i = i + 1;
 6                  c = c >> 1;
 7          }
 8          assert(i <= 7);
 9          return i;
10  }
```

Projection

Projection

- formula `f` relates input `s` to output `s'`

Projection

- formula f relates input s to output s'
- for $f^2 = f \cdot f$ must rename s, s' to s', s''

Projection

- formula f relates input s to output s'
- for $f^2 = f \cdot f$ must rename s, s' to s', s''
- cannot rename variables forever: calculating $f^4 = f^2 \cdot f^2$ requires two copies of f^2

Projection

- formula f relates input s to output s'
- for $f^2 = f \cdot f$ must rename s, s' to s', s''
- cannot rename variables forever: calculating $f^4 = f^2 \cdot f^2$ requires two copies of f^2
- f^{2^i} requires formula of size $O(2^i)$

Projection

- formula f relates input s to output s'
- for $f^2 = f \cdot f$ must rename s, s' to s', s''
- cannot rename variables forever: calculating $f^4 = f^2 \cdot f^2$ requires two copies of f^2
- f^{2^i} requires formula of size $O(2^i)$
- use projection: $f^2 = \exists_{vars(s')} (f \cdot f)$

Implementation of ∃

- algorithm of [Brauer et al., CAV'11]:
  - given a CNF: $c_1 \wedge \ldots \wedge c_n$,
  - for a subset of variables:
  - calculate d_1, $d_1 \vee d_2$, ..., $d_1 \vee \ldots \vee d_m$ with $d_1 \vee \ldots \vee d_m \models \ldots d_1 \vee d_2 \models d_1$
  - once DNF is complete, apply alg. again

Under-Approximations

Under-Approximations

- strategy: stop calculating DNF d_1, $d_1 \vee d_2$, ..., $d_1 \vee \ldots \vee d_m$ after threshold m

Under-Approximations

- strategy: stop calculating DNF d_1, $d_1 \vee d_2$, ..., $d_1 \vee \ldots \vee d_m$ after threshold m
- sub-sequence is under-approximation

Under-Approximations

- strategy: stop calculating DNF d_1, $d_1 \vee d_2$, ..., $d_1 \vee \ldots \vee d_m$ after threshold m
- sub-sequence is under-approximation
- may be enough to find backward trace

Under-Approximations

- strategy: stop calculating DNF d_1, $d_1 \vee d_2$, ..., $d_1 \vee \ldots \vee d_m$ after threshold m
- sub-sequence is under-approximation
- may be enough to find backward trace
- could stop calculation of DNF at depth m

Under-Approximations

- strategy: stop calculating DNF d_1, $d_1 \vee d_2$, ..., $d_1 \vee \ldots \vee d_m$ after threshold m
- sub-sequence is under-approximation
- may be enough to find backward trace
- could stop calculation of DNF at depth m
- iterative deepening by increasing m?

Incremental Backward Analysis

```
unsigned int hamDist(int x, int y) {
        unsigned int d = 0;
        unsigned int v = x ^ y;
        while (v != 0) {
                d = d + 1;
                v = v & (v - 1);
        }
        assert(d < 32);
        return d;
}
```

Approximation Strategy

Approximation Strategy

- affine relations: at least 32 iterations

Approximation Strategy

- affine relations: at least 32 iterations
- f^{32} is very complex to calculate, so approx!

Approximation Strategy

- affine relations: at least 32 iterations
- f^{32} is very complex to calculate, so approx!
- limiting m doesn't scale: m=1: $2^{31} \leq d \leq 2^{32-1}$

Approximation Strategy

- affine relations: at least 32 iterations
- f^{32} is very complex to calculate, so approx!
- limiting m doesn't scale: m=1: $2^{31} \leq d \leq 2^{32-1}$
- search for traces that emanate near correct/incorrect boundary in ψ

Approximation Strategy

- affine relations: at least 32 iterations
- f^{32} is very complex to calculate, so approx!
- limiting m doesn't scale: m=1: $2^{31} \leq d \leq 2^{32-1}$
- search for traces that emanate near correct/incorrect boundary in ψ
- in example: try $2^5 \leq d \leq 2^6$, $2^6 \leq d \leq 2^7$,...

Implementation

- affine loop transfer function: SAT4J
- projection: MiniSat for CNF→DNF
- BDD for DNF→CNF using CUDD 2.4.2
- tracked times for
  - loop transfer function synthesis (TF)
  - counter example generation (CE)

Experiments

Benchmark	# Instr.	Runtime (Full)		Runtime (Simp.)	
		TF	CE	TF	CE
bit-cnt	26	4.1s	0.9s	0.4s	0.4s
ham-dist	19	4.8s	1.7s	0.8s	0.3s
inc-lshift	14	3.2s	2.7s	0.8s	0.6s
log	22	1.9s	1.3s	0.3s	0.3s
parity	28	8.3s	1.2s	1.2s	0.4s
parity_mit	17	6.2s	2.6s	1.5s	1.2s
randerson	23	8.0s	2.4s	4.2s	0.6s
swap	15	5.9s	1.8s	0.9s	0.5s
loops	207	43.6s	8.0s	13.1s	5.8s

Related Approaches

- CEGAR: counter example in the model is given by model checker
  - need to find trace in concrete semantics
- bounded model checking: limit unrollings
  - we refine/unroll back to front
- backward over-approximation

Future Work

Future Work

- scalability
  - use cheaper domains if possible
  - combine with domains that are able to express abduction (SMT?)
  - allow for user input

Future Work

- scalability
  - use cheaper domains if possible
  - combine with domains that are able to express abduction (SMT?)
  - allow for user input
- lose soundness: allow for qualified traces?