Design and Analysis of Real-Time Systems

Foundations of Abstract Interpretation I

Jan Reineke

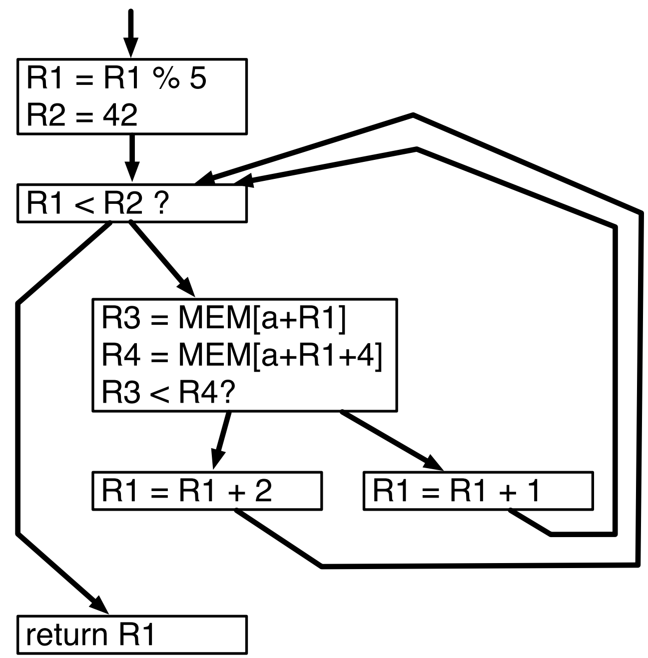
Advanced Lecture, Summer 2013

Recap: Value Analysis

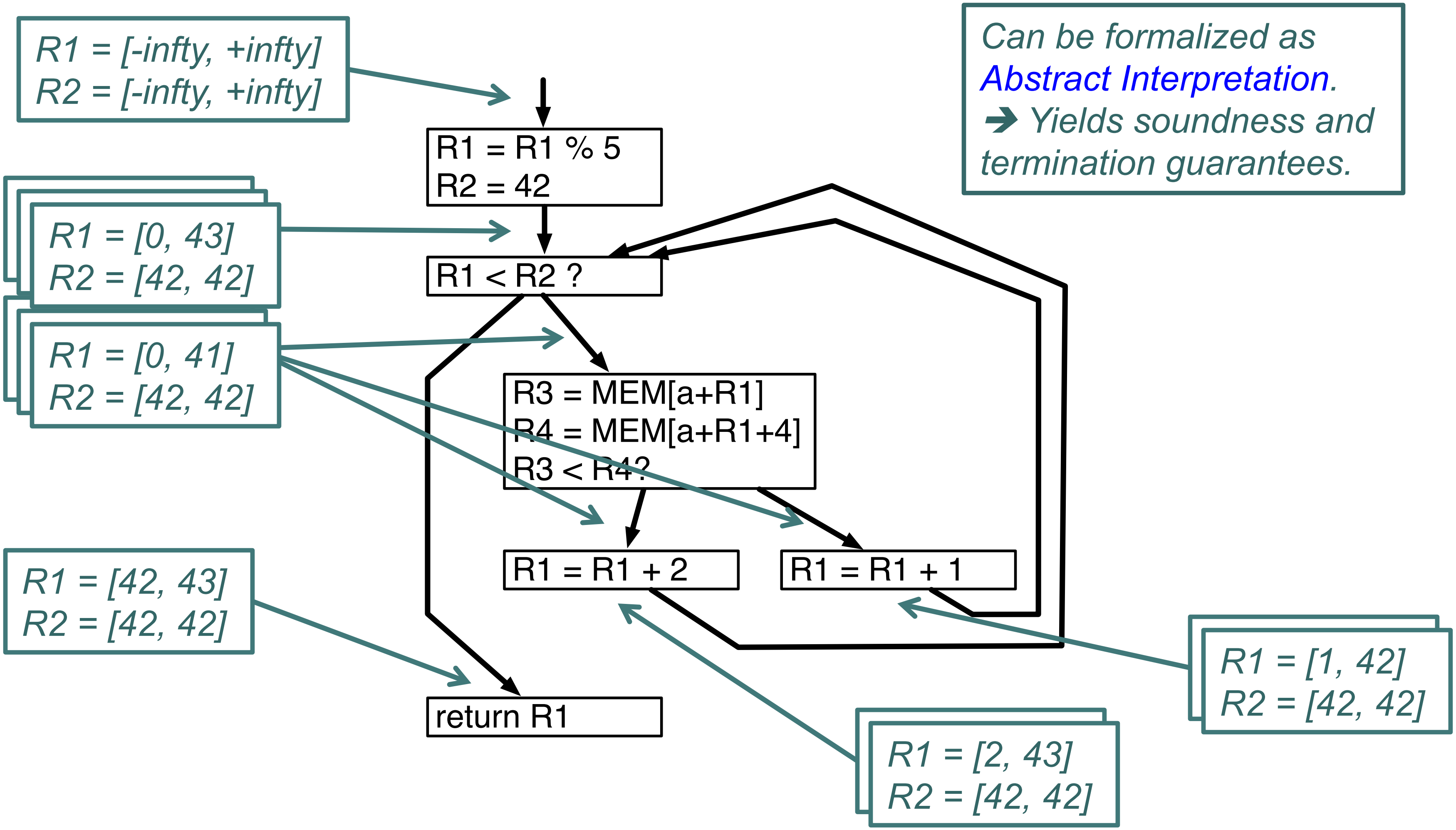
Determines invariants on values of registers at different program points. Invariants are often in the form of enclosing intervals of all possible values.

Where is this information used?

- Microarchitectural Analysis
  - Pipeline Analysis
  - Cache Analysis
- Control-Flow Analysis
  - Detect infeasible paths
  - Derive loop bounds



Value Analysis
Intuition of Interval Analysis



Can be formalized as Abstract Interpretation.
➔ Yields soundness and termination guarantees.

Abstract Interpretation

- Semantics-based approach to program analysis
- Framework to develop provably correct and terminating analyses

Ingredients:

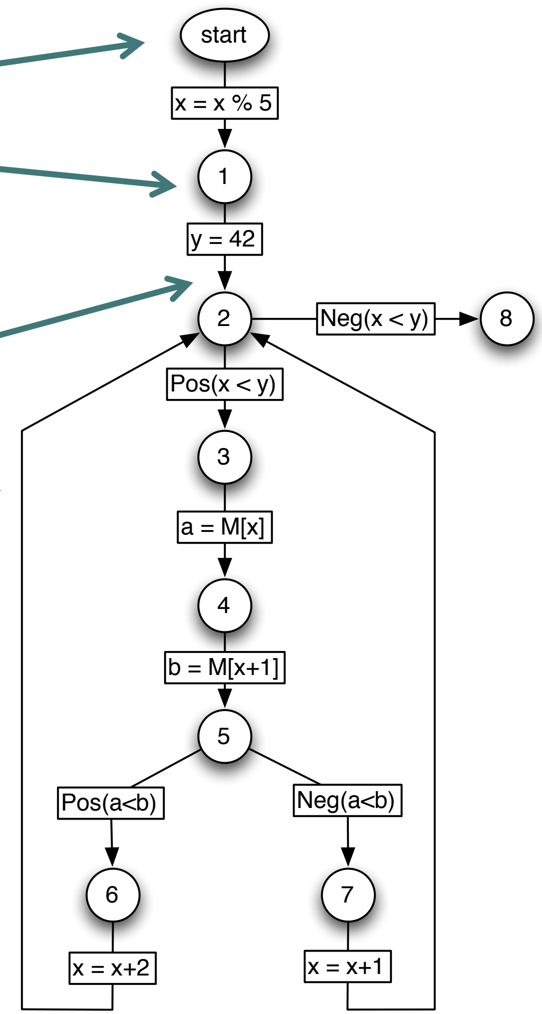
- Concrete semantics: Formalizes meaning of a program
- Abstract semantics
- Both semantics defined as fixpoints of monotone functions over some domain
- Relation between the two semantics establishing correctness

Representing Programs by Control-Flow Graphs

Vertices

$$CFG\ G = (V, E, start, labeling)$$

Edges



$$start \in V$$

$$E \subseteq V \times V$$

$$labeling : E \rightarrow Statement$$

Four Kinds of Statements

1. Assignment: $R = e$
2. Load: $R = M[e]$
3. Store: $M[e_1] = e_2$
4. Test: $Pos(e)$ or $Neg(e)$

Meaning of Statements

States consist of variables and memory:

$$s = (\rho, \mu) \in States$$

$$\rho : Vars \to int$$

Values of Variables

$$\mu : \mathbb{N} \to int$$

Contents of Memory

Execution of a statement transforms states:

$$[\![statement]\!] \subseteq States \times States$$

Meaning of Statements

States consist of variables and memory:

$$s = (\rho, \mu) \in States$$

$$\rho : Vars \to int$$

Values of Variables

$$\mu : \mathbb{N} \to int$$

Contents of Memory

Execution of a statement transforms states:

$$[\![statement]\!] \subseteq States \times States$$

$$[\![R = e]\!] := \{((\rho, \mu), (\rho[R \mapsto [\![e]\!]\rho], \mu)) \mid (\rho, \mu) \in States\}$$

Meaning of Statements

States consist of variables and memory:

$$s = (\rho, \mu) \in \mathit{States}$$

$$\rho : \mathit{Vars} \to \mathit{int}$$ Values of Variables

$$\mu : \mathbb{N} \to \mathit{int}$$ Contents of Memory

Execution of a statement transforms states:

$$[\![\mathit{statement}]\!] \subseteq \mathit{States} \times \mathit{States}$$

$$[\![R = e]\!] := \{((\rho, \mu), (\rho[R \mapsto [\![e]\!]\rho], \mu)) \mid (\rho, \mu) \in \mathit{States}\}$$

$$[\![R = M[e]]\!] := \{((\rho, \mu), (\rho[R \mapsto \mu([\![e]\!]\rho)], \mu)) \mid (\rho, \mu) \in \mathit{States}\}$$

Meaning of Statements

States consist of variables and memory:

$$s = (\rho, \mu) \in States$$

$$\rho : Vars \to int$$ *Values of Variables*

$$\mu : \mathbb{N} \to int$$ *Contents of Memory*

Execution of a statement transforms states:

$$[\![statement]\!] \subseteq States \times States$$

$$[\![R = e]\!] := \{((\rho, \mu), (\rho[R \mapsto [\![e]\!]\rho], \mu)) \mid (\rho, \mu) \in States\}$$

$$[\![R = M[e]]\!] := \{((\rho, \mu), (\rho[R \mapsto \mu([\![e]\!]\rho)], \mu)) \mid (\rho, \mu) \in States\}$$

$$[\![M[e_1] = e_2]\!] := \{((\rho, \mu), (\rho, \mu[[\![e_1]\!]\rho \mapsto [\![e_2]\!]\rho])) \mid (\rho, \mu) \in States\}$$

Meaning of Statements

States consist of variables and memory:

$$s = (\rho, \mu) \in \mathit{States}$$

$$\rho : \mathit{Vars} \to \mathit{int}$$ *Values of Variables*

$$\mu : \mathbb{N} \to \mathit{int}$$ *Contents of Memory*

Execution of a statement transforms states:

$$[\![\mathit{statement}]\!] \subseteq \mathit{States} \times \mathit{States}$$

$$[\![R = e]\!] := \{((\rho, \mu), (\rho[R \mapsto [\![e]\!]\rho], \mu)) \mid (\rho, \mu) \in \mathit{States}\}$$

$$[\![R = M[e]]\!] := \{((\rho, \mu), (\rho[R \mapsto \mu([\![e]\!]\rho)], \mu)) \mid (\rho, \mu) \in \mathit{States}\}$$

$$[\![M[e_1] = e_2]\!] := \{((\rho, \mu), (\rho, \mu[[\![e_1]\!]\rho \mapsto [\![e_2]\!]\rho])) \mid (\rho, \mu) \in \mathit{States}\}$$

$$[\![\mathit{Pos}(e)]\!] := \{((\rho, \mu), (\rho, \mu)) \mid (\rho, \mu) \in \mathit{States} \wedge [\![e]\!]\rho \neq 0\}$$

Meaning of Statements

States consist of variables and memory:

$$s = (\rho, \mu) \in \mathit{States}$$

$$\rho : \mathit{Vars} \to \mathit{int}$$ Values of Variables

$$\mu : \mathbb{N} \to \mathit{int}$$ Contents of Memory

Execution of a statement transforms states:

$$[\![\mathit{statement}]\!] \subseteq \mathit{States} \times \mathit{States}$$

$$[\![R = e]\!] := \{((\rho, \mu), (\rho[R \mapsto [\![e]\!]\rho], \mu)) \mid (\rho, \mu) \in \mathit{States}\}$$

$$[\![R = M[e]]\!] := \{((\rho, \mu), (\rho[R \mapsto \mu([\![e]\!]\rho)], \mu)) \mid (\rho, \mu) \in \mathit{States}\}$$

$$[\![M[e_1] = e_2]\!] := \{((\rho, \mu), (\rho, \mu[[\![e_1]\!]\rho \mapsto [\![e_2]\!]\rho])) \mid (\rho, \mu) \in \mathit{States}\}$$

$$[\![\mathit{Pos}(e)]\!] := \{((\rho, \mu), (\rho, \mu)) \mid (\rho, \mu) \in \mathit{States} \wedge [\![e]\!]\rho \neq 0\}$$

$$[\![\mathit{Neg}(e)]\!] := \{((\rho, \mu), (\rho, \mu)) \mid (\rho, \mu) \in \mathit{States} \wedge [\![e]\!]\rho = 0\}$$

Meaning of Expressions

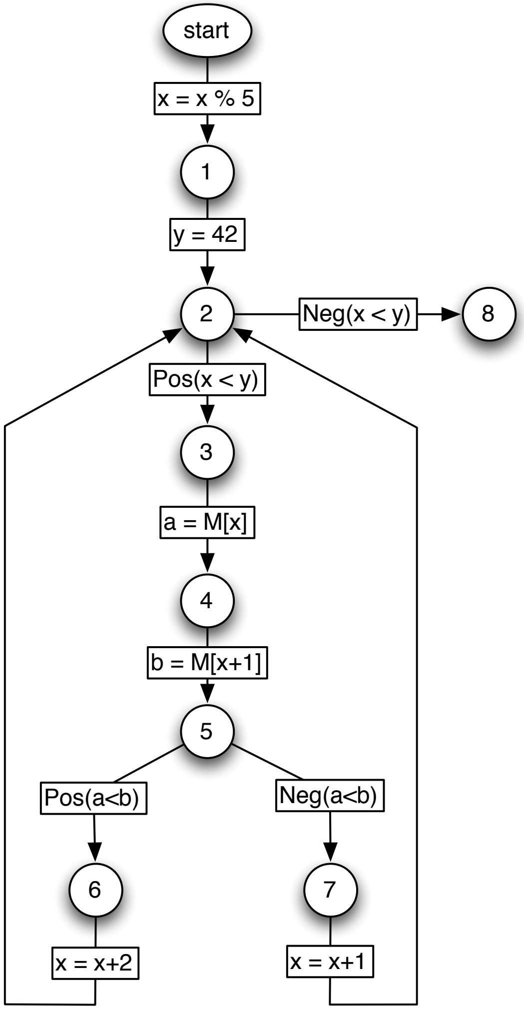
Evaluation of expressions is as expected:

$$
\begin{aligned}
[\![a]\!]\rho &:= \rho(a) & \qquad\qquad \textit{if } a \in \textit{Vars} \\
[\![e_1 \otimes e_2]\!]\rho &:= [\![e_1]\!]\rho \otimes [\![e_2]\!]\rho \\
[\![a < b]\!]\rho &:= \begin{cases} 1 & : [\![a]\!]\rho < [\![b]\!]\rho \\ 0 & : \textit{otherwise} \end{cases} \\
&\ldots
\end{aligned}
$$

Concrete Semantics

Different semantics are required for different properties:

- "Is there an execution in which the value of x alternates between 3 and 5?" ➔ Trace Semantics
- "Is the final value of x always the same as the initial value of x?" ➔ "Input/Output" Semantics
- "May x ever assume the value 45 at program point 7?" ➔ Reachability Semantics



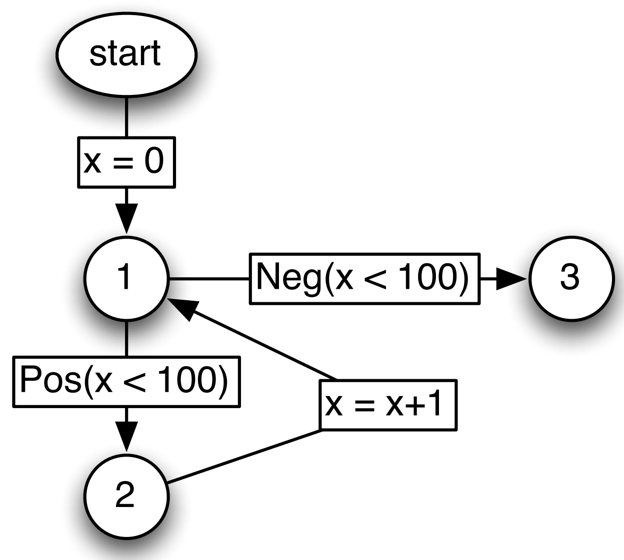
Concrete Semantics

- Trace Semantics: Captures set of traces of states that the program may execute.
- Input/Output Semantics: Captures the pairs of initial and final states of execution traces.
  - Abstraction of Trace Semantics
- Reachability Semantics: Captures the set of reachable states at each program point
  - Abstraction of Trace Semantics

Reachability Semantics

Captures the set of reachable states at each program point. Formally: $Reach : V \to \mathcal{P}(States)$

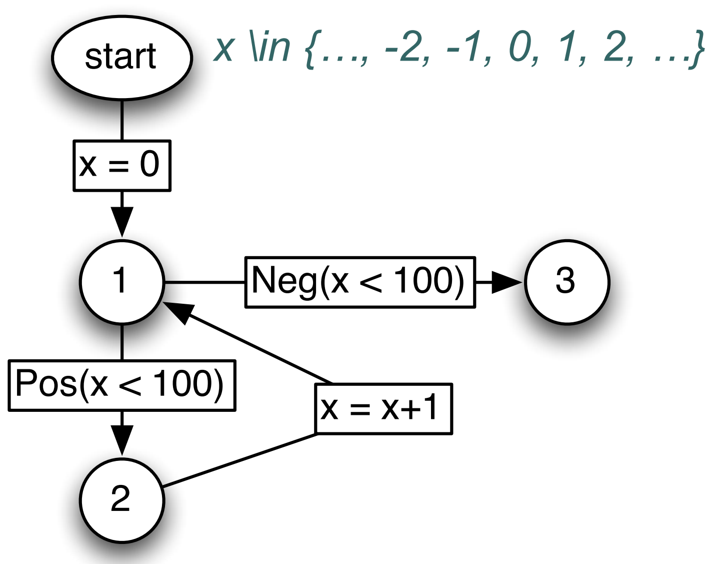
Example:



Reachability Semantics

Captures the set of reachable states at each program point. Formally: $Reach : V \to \mathcal{P}(States)$

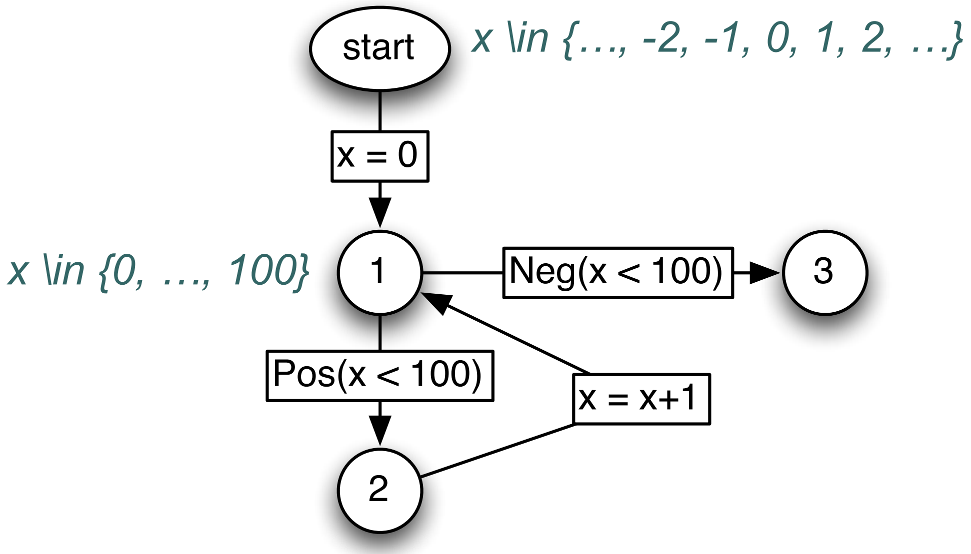
Example:



Reachability Semantics

Captures the set of reachable states at each program point. Formally: $Reach : V \to \mathcal{P}(States)$

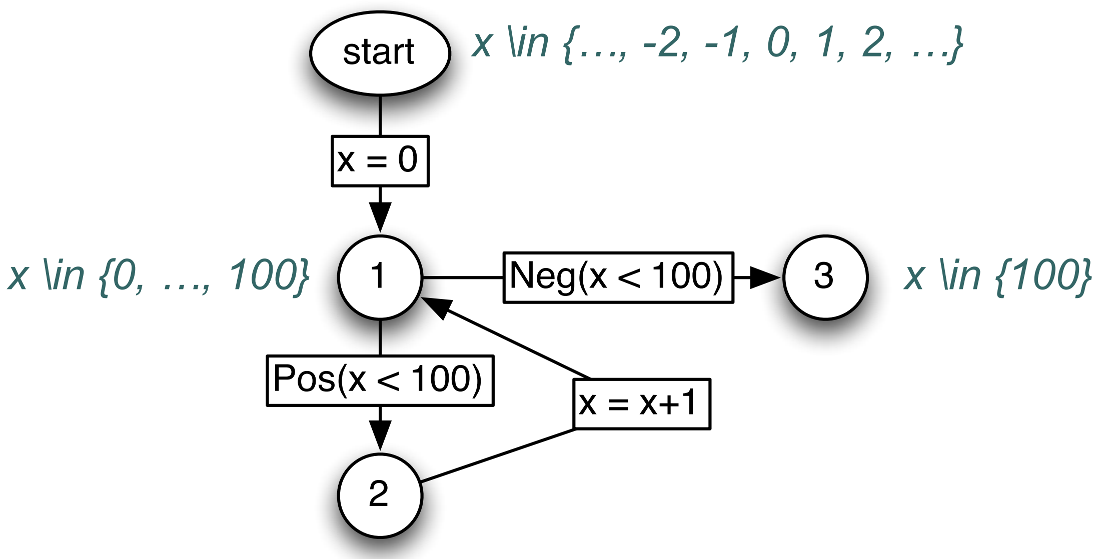
Example:



Reachability Semantics

Captures the set of reachable states at each program point. Formally: $Reach : V \to \mathcal{P}(States)$

Example:

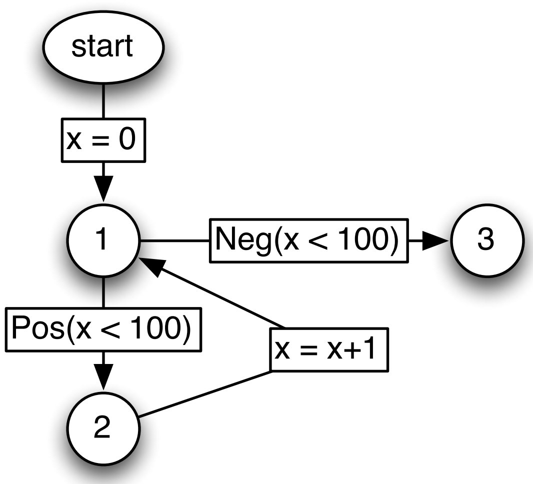


Reachability Semantics

Can be captured as the least solution of:

$$Reach(start) = States$$

$$\forall v' \in V \setminus \{start\} : Reach(v') = \bigcup_{v \in V, (v,v') \in E} [\![labeling(v, v')]\!](Reach(v))$$



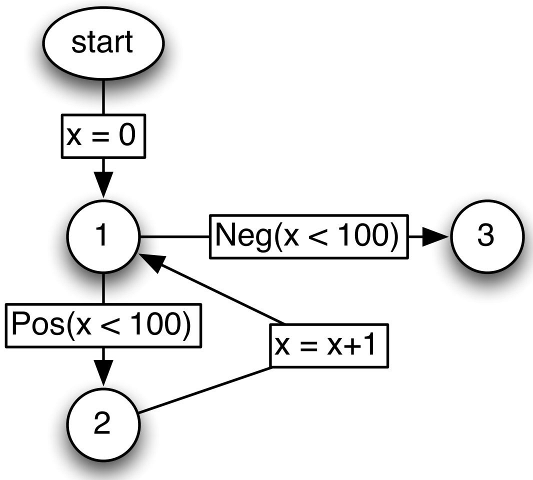
$$Reach(1) = [\![labeling(start, 1)]\!](Reach(start)) \cup [\![labeling(2, 1)]\!](Reach(2))$$
$$Reach(2) = [\![labeling(1, 2)]\!](Reach(1))$$
$$Reach(3) = [\![labeling(1, 3)]\!](Reach(1))$$

Reachability Semantics

Can be captured as the least solution of:

$$Reach(start) = States$$

$$\forall v' \in V \setminus \{start\} : Reach(v') = \bigcup_{v \in V, (v,v') \in E} [\![labeling(v, v')]\!](Reach(v))$$



$$Reach(1) = [\![labeling(start, 1)]\!](Reach(start)) \cup [\![labeling(2, 1)]\!](Reach(2))$$
$$Reach(2) = [\![labeling(1, 2)]\!](Reach(1))$$
$$Reach(3) = [\![labeling(1, 3)]\!](Reach(1))$$

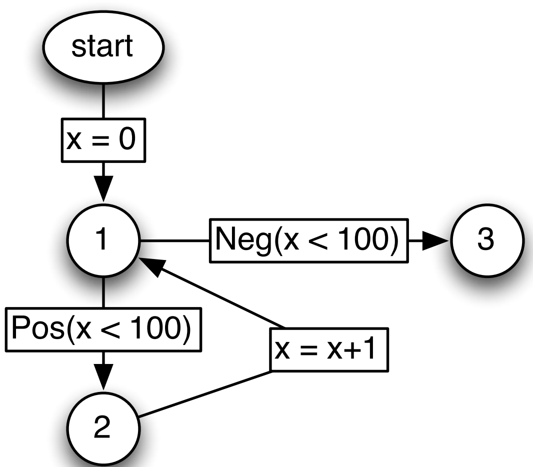
$$Reach(1) = [\![x = 0]\!](Reach(start)) \cup [\![x = x + 1]\!](Reach(2))$$
$$Reach(2) = [\![Pos(x < 100)]\!](Reach(1))$$
$$Reach(3) = [\![Neg(x < 100)]\!](Reach(1))$$

Reachability Semantics

Can be captured as the least solution of:

$$Reach(start) = States$$

$$\forall v' \in V \setminus \{start\} : Reach(v') = \bigcup_{v \in V, (v,v') \in E} [\![labeling(v, v')]\!](Reach(v))$$



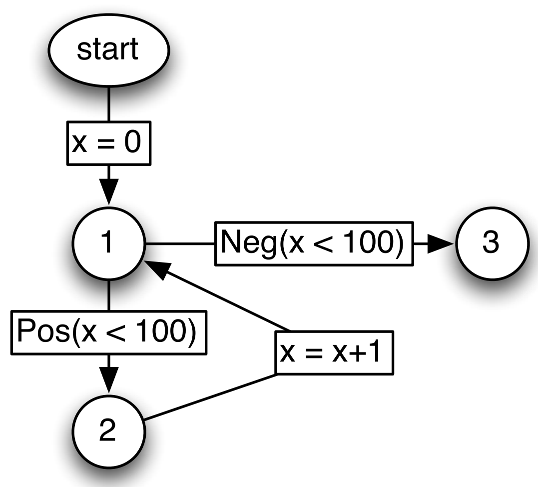
$$\begin{aligned} Reach(1) &= [\![labeling(start, 1)]\!](Reach(start)) \cup [\![labeling(2, 1)]\!](Reach(2)) \\ Reach(2) &= [\![labeling(1, 2)]\!](Reach(1)) \\ Reach(3) &= [\![labeling(1, 3)]\!](Reach(1)) \end{aligned}$$

$$\begin{aligned} Reach(1) &= [\![x = 0]\!](Reach(start)) \cup [\![x = x + 1]\!](Reach(2)) \\ Reach(2) &= [\![Pos(x < 100)]\!](Reach(1)) \\ Reach(3) &= [\![Neg(x < 100)]\!](Reach(1)) \end{aligned}$$

$$\begin{aligned} Reach(1) &= \{0\} \cup \{v + 1 \mid v \in Reach(2)\} \\ Reach(2) &= Reach(1) \cap \{\ldots, 98, 99\} \\ Reach(3) &= Reach(1) \cap \{100, 101, \ldots\} \end{aligned}$$

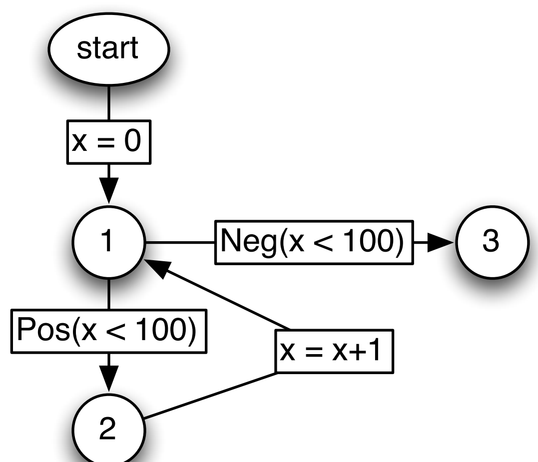
Questions

- Why the least solution?
- Is there more than one solution?
- Is there a unique least solution?
- Can we systematically compute it?



Answers

- Is there more than one solution? Yes!
- Is there a unique least solution? Yes!
- Can we systematically compute it? Yes and No.



Why? Knaster-Tarski Fixpoint Theorem!

THEOREM 1 (KNASTER-TARSKI, 1955).
Assume $(D, \leq)$ *is a* complete lattice. *Then every* monotonic *function* $f : D \to D$ *has a* least fixed point $d_0 \in D$.

Raises more questions:

- What is a complete lattice?
- What is a monotonic function?
- What is a fixed point?

Monotone Functions

Let $(D, \leq)$ be *partially-ordered set*.
For example: $D = \mathbb{N}$ and $\leq$ the order on natural numbers.

Function $f : D \to D$ is *monotone* (order-preserving) iff

$$\text{for all } d_1, d_2 \in D : d_1 \leq d_2 \Rightarrow f(d_1) \leq f(d_2).$$

Monotone Functions

Let $(D, \leq)$ be *partially-ordered set.*
For example: $D = \mathbb{N}$ and $\leq$ the order on natural numbers.

Function $f : D \to D$ is *monotone* (order-preserving) iff

$$\text{for all } d_1, d_2 \in D : d_1 \leq d_2 \Rightarrow f(d_1) \leq f(d_2).$$

Examples:

$$f(x) = x$$

$$g(x) = -x$$

$$h(x) = x - 1$$

Which of these are monotone?

$$F(X) = \{f(x) \mid x \in X\}$$

$$G(X) = \{y \mid x \in X \wedge (x, y) \in R\}$$

Monotone Functions

Let $(D, \leq)$ be *partially-ordered set.*
For example: $D = \mathbb{N}$ and $\leq$ the order on natural numbers.

Function $f : D \to D$ is *monotone* (order-preserving) iff
$$\text{for all } d_1, d_2 \in D : d_1 \leq d_2 \Rightarrow f(d_1) \leq f(d_2).$$

Examples:

$$f(x) = x$$
$$g(x) = -x$$
$$h(x) = x - 1$$
$$F(X) = \{f(x) \mid x \in X\}$$
$$G(X) = \{y \mid x \in X \wedge (x, y) \in R\}$$

Which of these are monotone?

Need to know what the order is.

Partial Orders

A binary relation $\leq : D \times D$ is a *partial order*, iff for all $a, b, c \in D$, we have that:

- $a \leq a$ (reflexivity),
- if $a \leq b$ and $b \leq a$ then $a = b$ (antisymmetry),
- if $a \leq b$ and $b \leq c$ then $a \leq c$ (transitivity).

A set with a partial order is called a *partially-ordered set*.

Partial Orders: Examples I

- The natural numbers ordered by the standard less-than-or-equal relation: $(\mathbb{N}, \leq)$.
- The set of subsets of a given set (its powerset) ordered by the subset relation: $(\mathcal{P}(A), \subseteq)$.
- The set of subsets of a given set (its powerset) ordered by the superset relation: $(\mathcal{P}(A), \supseteq)$.
- The natural numbers ordered by *divisibility*: $(\mathbb{N}, |)$.

Partial Orders: Examples II

The vertex set V of a directed acyclic graph $G = (V, E)$ ordered by reachability (reflexive, transitive closure of edge relation).

The vertex set V of an arbitrary graph $G = (V, E)$ ordered by reachability.

For a set X and a partially-ordered set P, the function space $F : X \to P$, where $f \leq g$ if and only if $f(x) \leq g(x)$ for all x in X.

Partial Orders: Examples II

The vertex set V of a directed acyclic graph $G = (V, E)$ ordered by reachability (reflexive, transitive closure of edge relation).

~~The vertex set V of an arbitrary graph $G = (V, E)$ ordered by reachability.~~

For a set X and a partially-ordered set P, the function space $F : X \to P$, where $f \leq g$ if and only if $f(x) \leq g(x)$ for all x in X.

Partial Orders: Examples II

The vertex set V of a directed acyclic graph $G = (V, E)$ ordered by reachability (reflexive, transitive closure of edge relation).

~~The vertex set V of an arbitrary graph $G = (V, E)$ ordered by reachability.~~

For a set X and a partially-ordered set P, the function space $F : X \to P$, where $f \leq g$ if and only if $f(x) \leq g(x)$ for all x in X.

What about $\mathit{Reach} : V \to \mathcal{P}(\mathit{States})$ *?*

Complete Lattices

A partially-ordered set $(L, \leq)$ is a *complete lattice* if every subset A of L has both a *least upper bound* (denoted $\bigsqcup A$) and a *greatest lower bound* (denoted $\bigsqcap A$).

Complete Lattices

A partially-ordered set $(L, \leq)$ is a *complete lattice* if every subset A of L has both a *least upper bound* (denoted $\bigsqcup A$) and a *greatest lower bound* (denoted $\bigsqcap A$).

What is an upper bound of a set A?

An element x is an upper bound of a set A if x if for every element a of A, we have $a \leq x$.

Complete Lattices

A partially-ordered set $(L, \leq)$ is a *complete lattice* if every subset A of L has both a *least upper bound* (denoted $\bigsqcup A$) and a *greatest lower bound* (denoted $\bigsqcap A$).

What is an upper bound of a set A?

An element x is an upper bound of a set A if x if for every element a of A, we have $a \leq x$.

What is the least upper bound (also: join, supremum) of a set A?

x is the *least upper bound* of A, denoted $\bigsqcup A$, if

1. x is an upper bound of A,
2. for every upper bound y of A, we have $x \leq y$.

Least Upper Bounds: Examples I

Partially-ordered set $(D, \leq)$	$A \subseteq D$	$\bigsqcup A$	$\bigsqcap A$
$(\mathbb{N}, \leq)$	$\{1, 2, 3\}$	?	?
$(\mathbb{R}, \leq)$	$\{x \in \mathbb{R} \mid x < 1\}$	?	?
$(\mathbb{R}, \leq)$	$\{x \in \mathbb{R} \mid x \leq 1\}$	?	?
$(\mathbb{Q}, \leq)$	$\{x \in \mathbb{Q} \mid x^2 \leq 2\}$	?	?
$(\mathbb{N}, \leq)$	$\{x \in \mathbb{N} \mid x \ is \ odd\}$	?	?

Least Upper Bounds: Examples I

$Partially\text{-}ordered\ set\ (D, \leq)$	$A \subseteq D$	$\bigsqcup A$	$\bigsqcap A$
$(\mathbb{N}, \leq)$	$\{1, 2, 3\}$	?	?
$(\mathbb{R}, \leq)$	$\{x \in \mathbb{R} \mid x < 1\}$	?	?
$(\mathbb{R}, \leq)$	$\{x \in \mathbb{R} \mid x \leq 1\}$	?	?
$(\mathbb{Q}, \leq)$	$\{x \in \mathbb{Q} \mid x^2 \leq 2\}$	?	?
$(\mathbb{N}, \leq)$	$\{x \in \mathbb{N} \mid x\ is\ odd\}$	?	?

Which of these are complete lattices?

Least Upper Bounds: Examples II

$Partially\text{-}ordered\ set\ (D, \leq)$	$A \subseteq D$	$\bigsqcup A$	$\bigsqcap A$
$(\mathcal{P}(\mathbb{N}), \subseteq)$	$\{\{1,2\},\{2,4,5\}\}$	?	?
$(\mathcal{P}(\mathbb{N}), \supseteq)$	$\{\{1,2\},\{2,4,5\}\}$	?	?
$(\mathbb{N}, \mid)$	$\{3,4,5\}$	?	?
$(A \to \mathbb{N}, \leq)$	$\{f, g, h\}$	?	?

Least Upper Bounds: Examples II

Partially-ordered set $(D, \leq)$	$A \subseteq D$	$\bigsqcup A$	$\bigsqcap A$
$(\mathcal{P}(\mathbb{N}), \subseteq)$	$\{\{1,2\},\{2,4,5\}\}$	?	?
$(\mathcal{P}(\mathbb{N}), \supseteq)$	$\{\{1,2\},\{2,4,5\}\}$	?	?
$(\mathbb{N}, \mid)$	$\{3,4,5\}$	?	?
$(A \to \mathbb{N}, \leq)$	$\{f, g, h\}$	?	?

Which of these are complete lattices?

Properties of Complete Lattices

Every complete lattice $(D, \leq)$ has

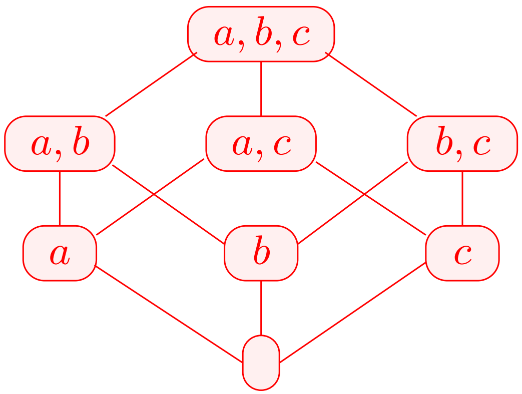
- a *least* element (*bottom* element): $\bot = \bigsqcup \emptyset$, and
- a *greatest* element (*top* element): $\top = \bigsqcup D$.

Generic Lattice Constructions: Power-set Lattice

For any set S, its power set $(\mathcal{P}(S), \subseteq)$ with set inclusion is a lattice:

$$
\begin{array}{rccc}
\textit{“join”:} & \bigsqcup A & = & \bigcup A \\
\textit{“meet”:} & \bigsqcap A & = & \bigcap A \\
\textit{“top”:} & \top & = & S \\
\textit{“bottom”:} & \bot & = & \emptyset
\end{array}
$$

Graphical representation (Hasse diagram):



Generic Lattice Constructions: Total Function Space

For any set S and complete lattice $(L, \leq_L)$, the total function space $(S \to L, \leq)$ is a complete lattice, with $f \leq g :\Leftrightarrow \forall\, s \in S : f(s) \leq g(s)$:

$$\begin{array}{rccc}
\text{“}join\text{”:} & \bigsqcup A & = & \lambda s. \bigsqcup_{f \in A} f(s) \\
\text{“}meet\text{”:} & \bigsqcap A & = & \lambda s. \bigsqcap_{f \in A} f(s) \\
\text{“}top\text{”:} & \top & = & \lambda s. \top_L \\
\text{“}bottom\text{”:} & \bot & = & \lambda s. \bot_L
\end{array}$$

Generic Lattice Constructions: Total Function Space

For any set S and complete lattice $(L, \leq_L)$, the total function space $(S \rightarrow L, \leq)$ is a complete lattice, with $f \leq g :\Leftrightarrow \forall\, s \in S : f(s) \leq g(s)$:

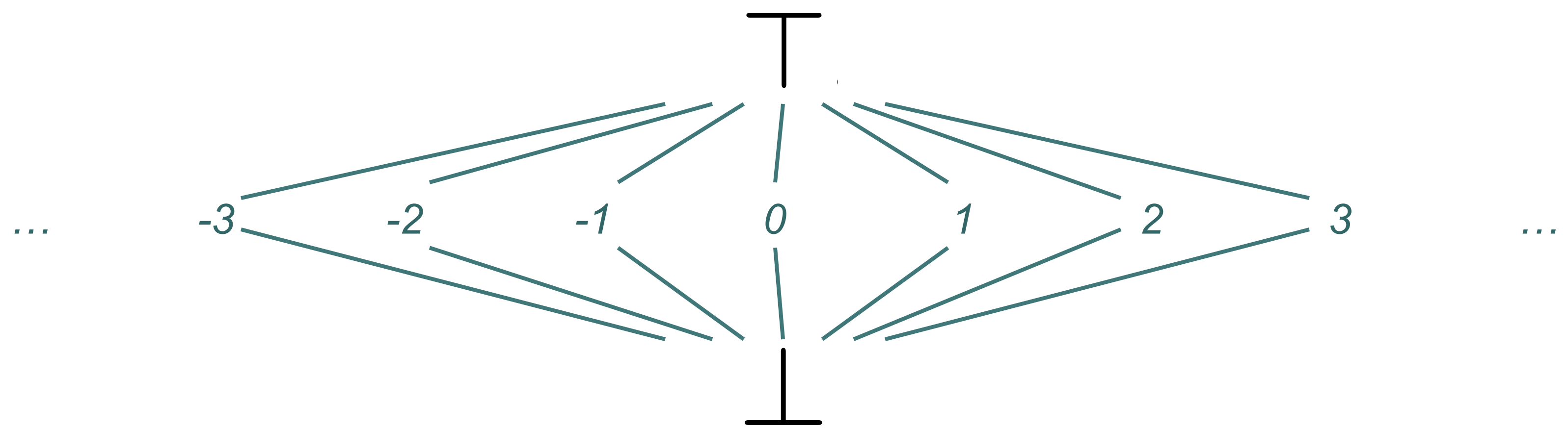
$$
\begin{array}{rrcl}
\text{"}join\text{"}: & \bigsqcup A & = & \lambda s. \bigsqcup_{f \in A} f(s) \\
\text{"}meet\text{"}: & \bigsqcap A & = & \lambda s. \bigsqcap_{f \in A} f(s) \\
\text{"}top\text{"}: & \top & = & \lambda s. \top_L \\
\text{"}bottom\text{"}: & \bot & = & \lambda s. \bot_L
\end{array}
$$

What about $Reach : V \rightarrow \mathcal{P}(States)$*?*

Generic Lattice Constructions: Flat Lattice

For any set S the flat lattice $(S \cup \{\bot, \top\}, \leq)$ is a complete lattice, with $a \leq b :\Leftrightarrow a = b \vee a = \bot \vee b = \top$.

Graphical representation (Hasse diagram) with $S = \mathbb{Z}$*:*



Fixed Points

A fixed point of a function $f : D \to D$ is an element $x \in D$ with $x = f(x)$.

Has multiple fixed points:

$$\{1, 2, 3\}$$
$$\{1, 2, 3, 4\}$$
$$\{1, 2, 3, 5\}$$
$$\{1, 2, 3, 4, 5\}$$

Fixed Points

A fixed point of a function $f : D \to D$ is an element $x \in D$ with $x = f(x)$.

Example:

$$f : \mathcal{P}(\{1,2,3,4,5\}) \to \mathcal{P}(\{1,2,3,4,5\})$$
$$f(X) = \{1,2,3\} \cup X$$

Has multiple fixed points:

$$\{1,2,3\}$$
$$\{1,2,3,4\}$$
$$\{1,2,3,5\}$$
$$\{1,2,3,4,5\}$$

Fixed Points

A fixed point of a function $f : D \to D$ is an element $x \in D$ with $x = f(x)$.

Example:

$$f : \mathcal{P}(\{1,2,3,4,5\}) \to \mathcal{P}(\{1,2,3,4,5\})$$
$$f(X) = \{1,2,3\} \cup X$$

Has multiple fixed points:

$$\{1,2,3\}$$
$$\{1,2,3,4\}$$
$$\{1,2,3,5\}$$
$$\{1,2,3,4,5\}$$

But a unique least fixed point.

$$\{1,2,3\}$$

Fixed Points

A fixed point of a function $f : D \to D$ is an element $x \in D$ with $x = f(x)$.

Example:

$$f : \mathcal{P}(\{1,2,3,4,5\}) \to \mathcal{P}(\{1,2,3,4,5\})$$
$$f(X) = \{1,2,3\} \cup X$$

Has multiple fixed points:

$$\{1,2,3\}$$
$$\{1,2,3,4\}$$
$$\{1,2,3,5\}$$
$$\{1,2,3,4,5\}$$

But a unique least fixed point.

$$\{1,2,3\}$$

The *least fixed point* l, denoted *lfp* f, of a function $f : D \to D$ over a lattice $(D, \leq)$, is a fixed point of f, such that for every fixed point x of f: $l \leq x$.

Knaster-Tarski Fixpoint Theorem

THEOREM 1 (KNASTER-TARSKI, 1955).
Assume $(D, \leq)$ *is a* complete lattice. *Then every* monotonic *function* $f : D \to D$ *has a* least fixed point $d_0 \in D$.

Raises more questions:

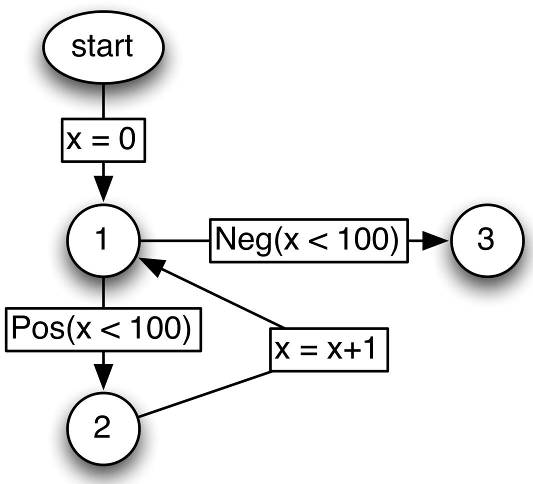
- What is a complete lattice? ✓
- What is a monotonic function? ✓
- What is a fixed point? ✓

Back to the Reachability Semantics

Can be captured as the least fixed point of:

$$Reach(start) = States$$

$$\forall v' \in V \setminus \{start\} : Reach(v') = \bigcup_{v \in V, (v,v') \in E} [\![labeling(v,v')]\!](Reach(v))$$



$$Reach(1) = [\![x = 0]\!](Reach(start)) \cup [\![x = x+1]\!](Reach(2))$$
$$Reach(2) = [\![Pos(x < 100)]\!](Reach(1))$$
$$Reach(3) = [\![Neg(x < 100)]\!](Reach(1))$$

Monotone?

$$Reach(1) = \{0\} \cup \{v+1 \mid v \in Reach(2)\}$$
$$Reach(2) = Reach(1) \cap \{\ldots, 98, 99\}$$
$$Reach(3) = Reach(1) \cap \{100, 101, \ldots\}$$