Abstract interpretation, reloaded

Jan Midtgaard

Week 3, Abstract Interpretation

Aarhus University, Q4 - 2012

Last time

Semantics overflow:

- The three counter machine
- An abstract machine for CPS terms
- A flow-chart semantics for IMP (non-deterministic!)
- A JVM-like semantics for a bytecode instruction set (objects,classes,methods,fields,. . .)

Finally we had another look at collecting semantics.

Today

- Approximation methods for AI (Cousot-Cousot:JLP92)
  - Lattice and fixed point theory
    - fixed points,
    - Galois connections
  - The Galois approach (p.11-. . .)
- From collecting semantics to static analysis
- Fun with Plotkin's three counter machine

Fixed points, reloaded

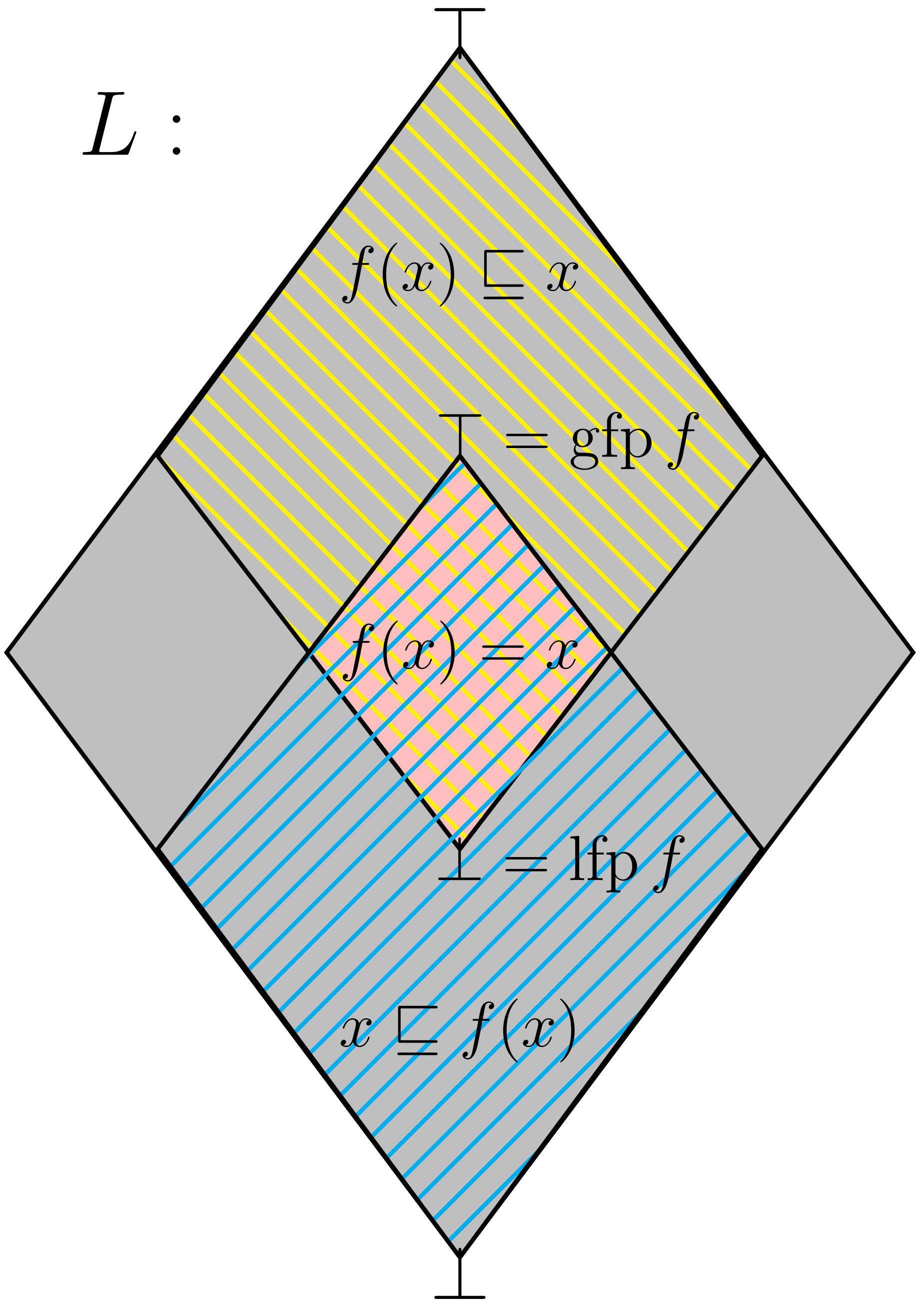
Tarski's fixed-point theorem

Theorem. (Tarski:PJM55) Let L be a complete lattice $\langle L; \sqsubseteq, \bot, \top, \sqcup, \sqcap \rangle$, and let f be a monotone function. Then the set P of all fixed points of f forms a complete lattice $\langle P; \sqsubseteq, \mathrm{lfp}\, f, \mathrm{gfp}\, f, \sqcup, \sqcap \rangle$ where

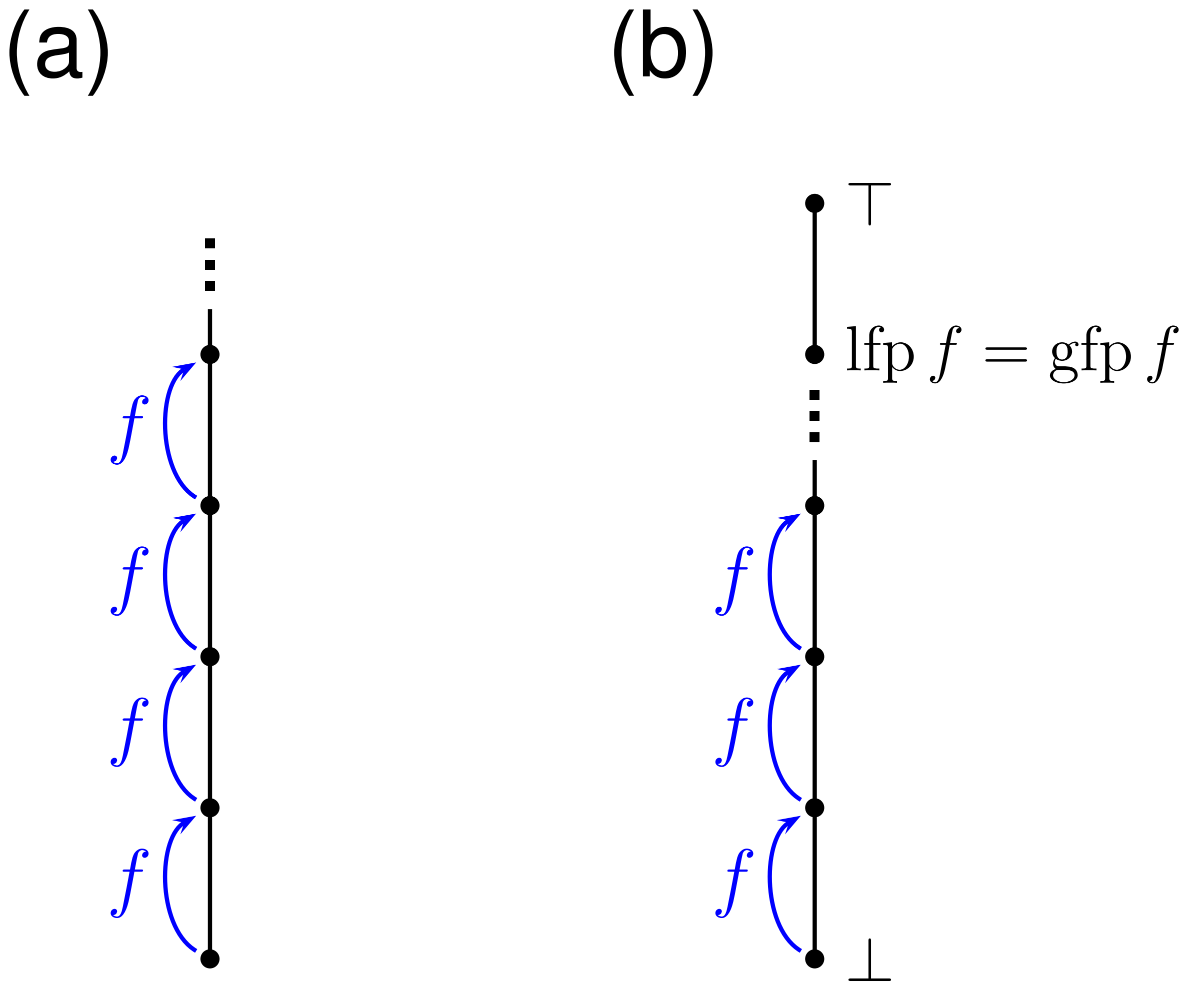
- $P = \{x \in L \mid x = f(x)\}$
- $\mathrm{lfp}\, f = \bigsqcap \{x \in L \mid f(x) \sqsubseteq x\}$
- $\mathrm{gfp}\, f = \bigsqcup \{x \in L \mid x \sqsubseteq f(x)\}$

Note: (1) $\mathrm{lfp}\, f$ is greatest lower bound of the set of post fixed points of f, and (2) $\mathrm{gfp}\, f$ is least upper bound of the set of pre fixed points of f.

Tarski's fixed point theorem, graphically



Fixed points, intuition



(a) On a poset a monotone function is not guaranteed to have a fixed point, (b) lfp and gfp may coincide, or (c) the fixed points may form a sub-lattice.

Galois connections, reloaded

Galois connection motivation

Partial orders model precision of properties: $a \sqsubseteq a'$ if the properties a and a' are *comparable* and a is *more precise* than a'.

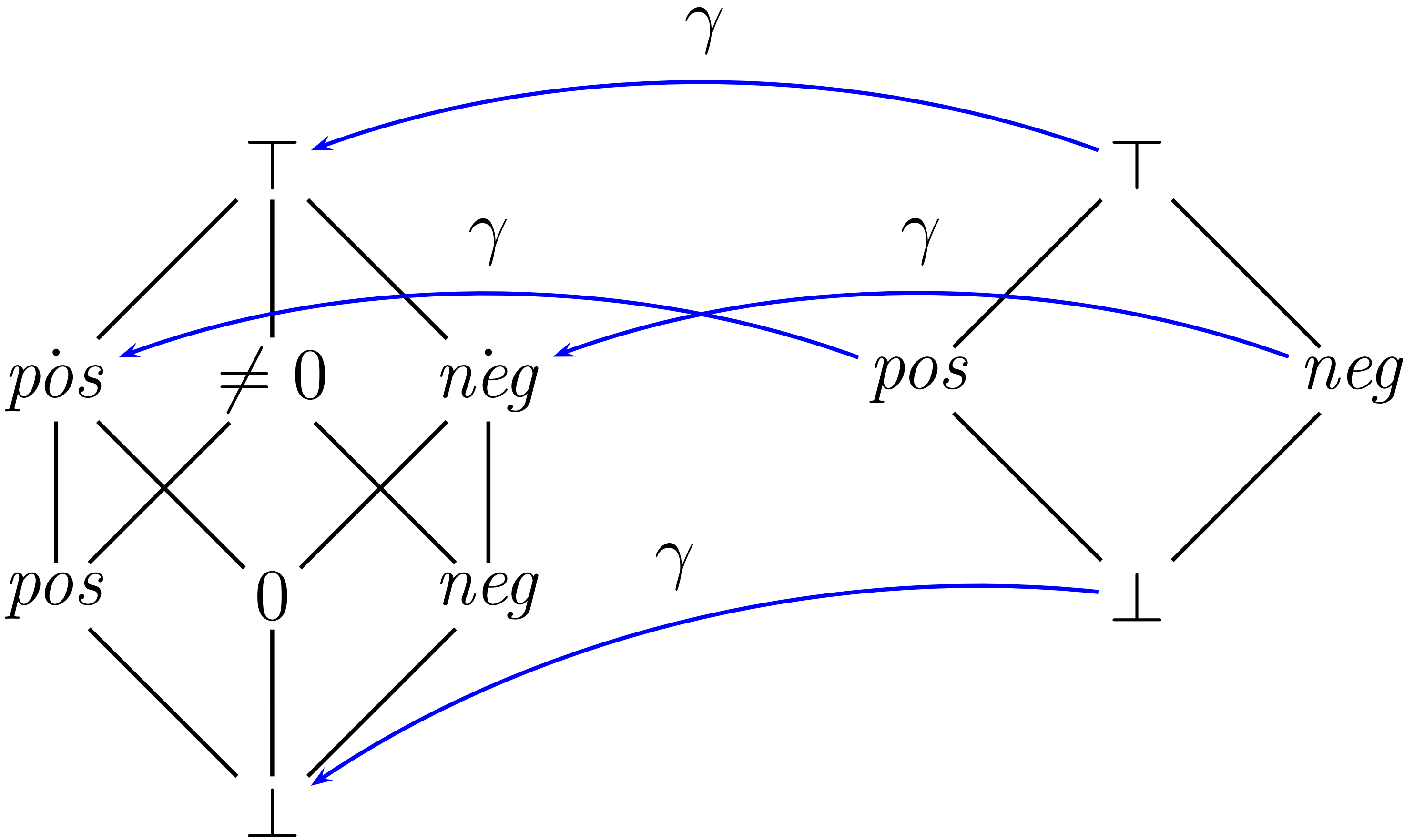
Example. Recall from the Parity domain:

The property $even$ meaning $\{n \in \mathbb{N} \mid n \text{ is even}\}$ is more precise than the property $\top$ meaning $\mathbb{N}$

The meaning of an abstract property is expressed by the concretization function γ.

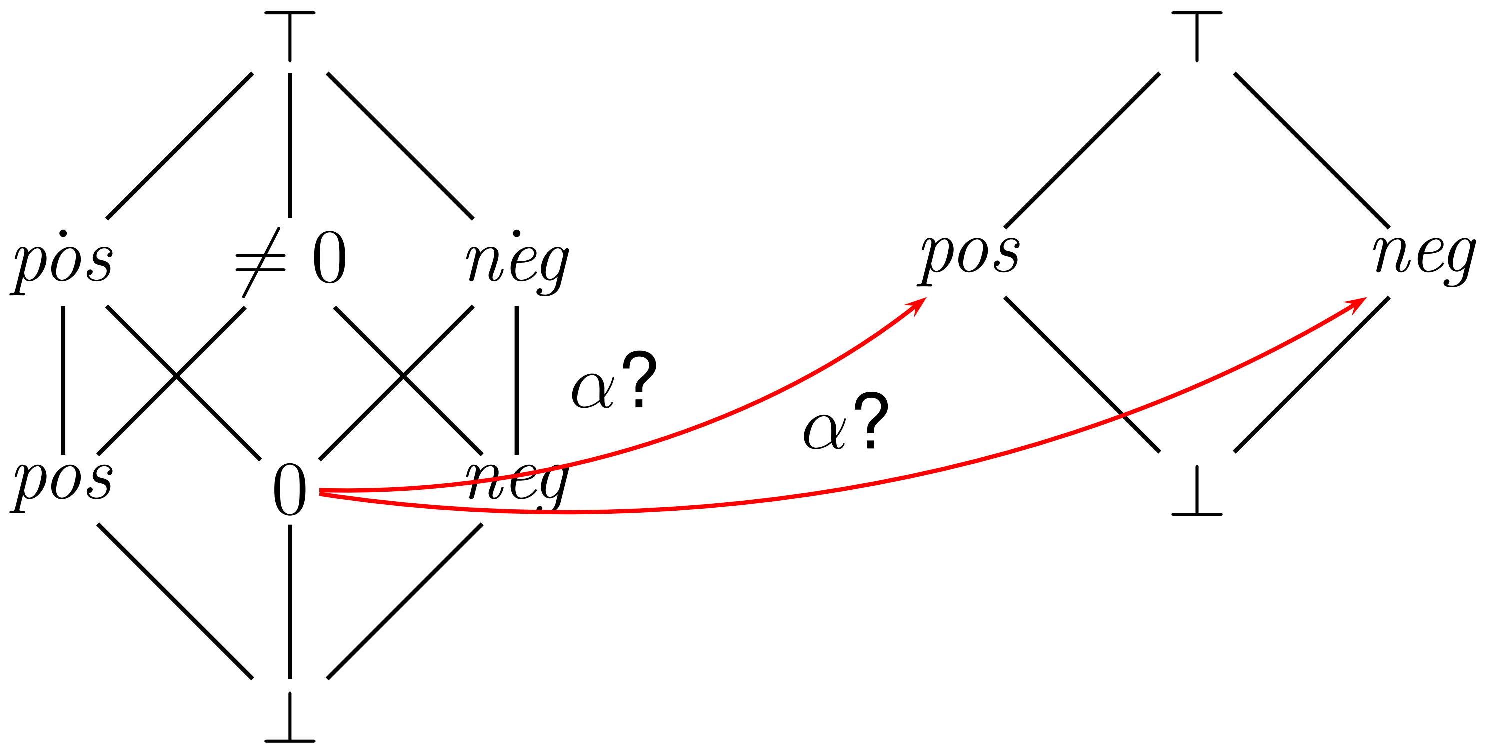
Approximation is captured by the abstraction function α: it maps each concrete property to its *best* abstract counterpart.

Galois connection non-example



γ assigns meaning to each abstract element.

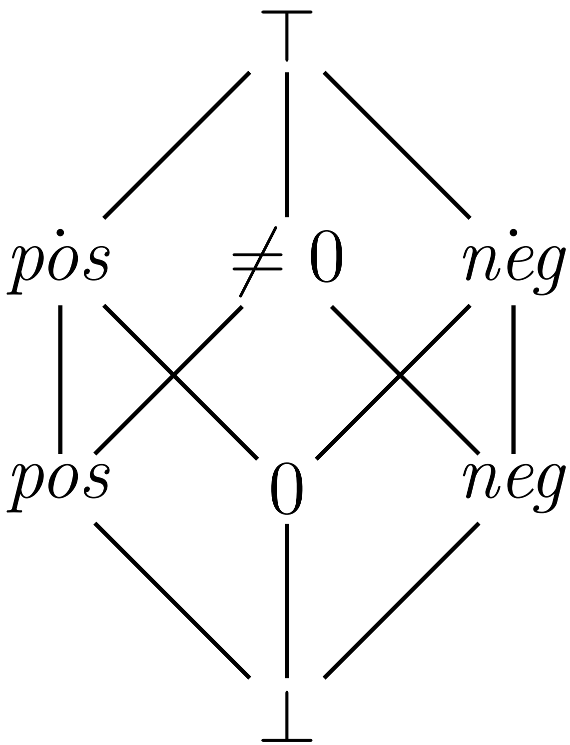
Galois connection non-example

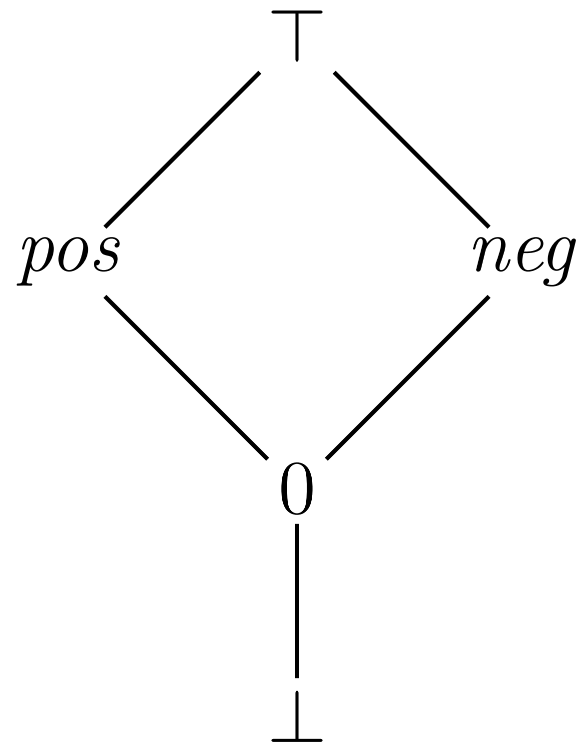


γ assigns meaning to each abstract element.

Problem: however there is no best (unique) abstraction for 0!

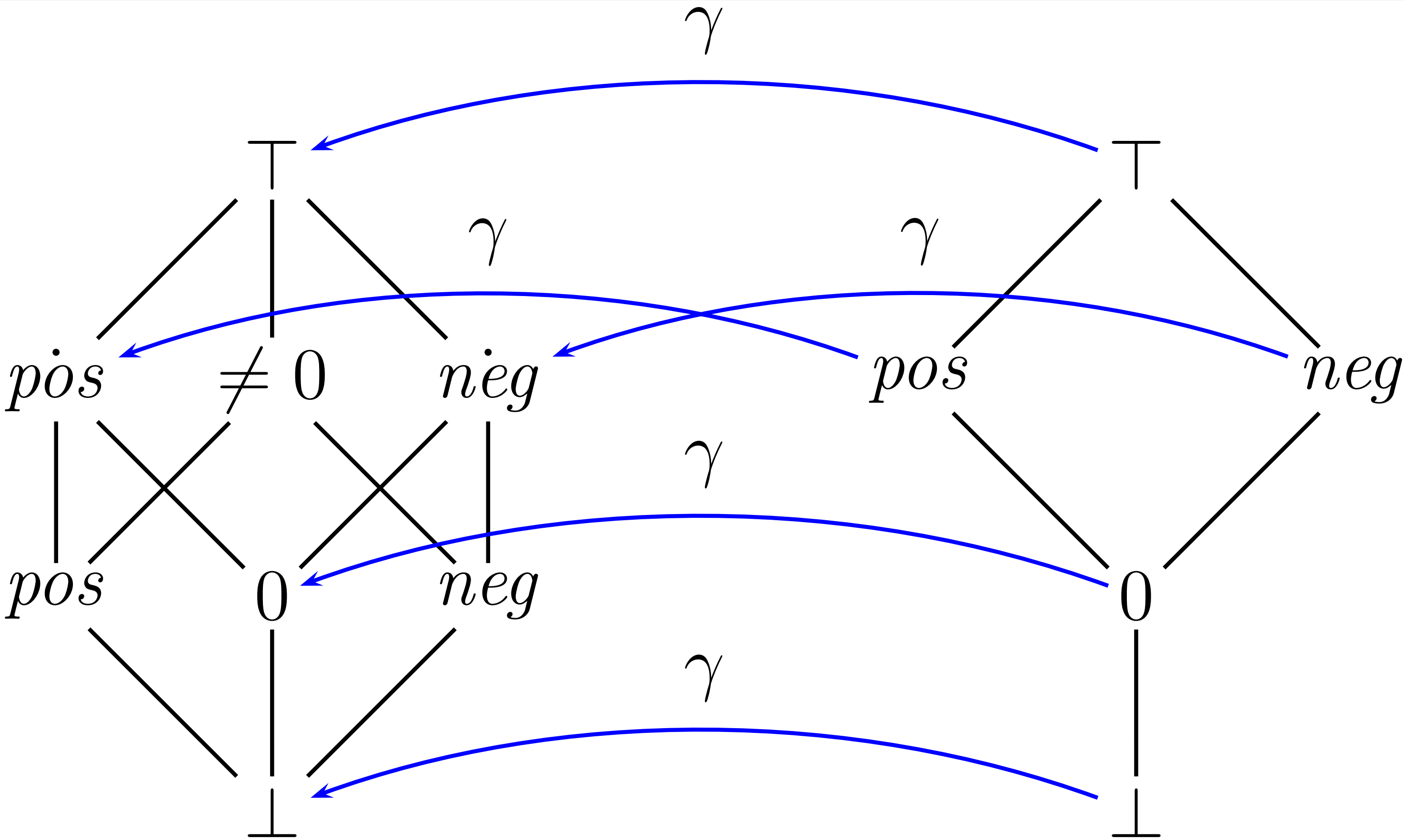
Galois connection example, fixed





We fix it by adding an element corresponding to 0.

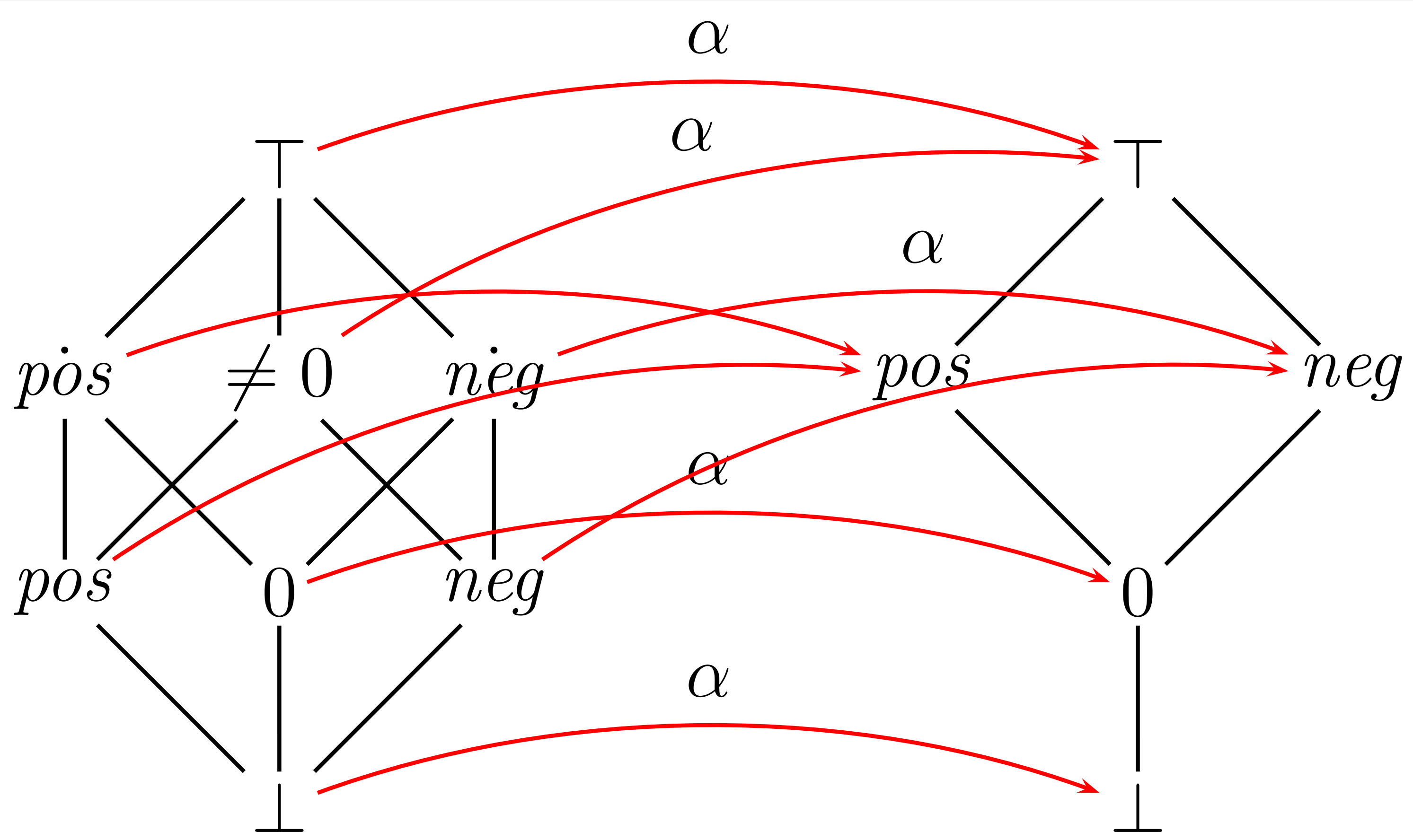
Galois connection example, fixed



γ assigns meaning to each abstract element.

Notice how γ is injective (one-to-one).

Galois connection example, fixed



α maps each element to its best abstraction.

Notice how α is surjective (onto), hence we have a Galois surjection.

Also notice the information loss.

Two soundness conditions

Condition 1:
If $a \leq a'$ for some c where $\alpha(c) = a$, then a' is a sound albeit less precise approximation of c.

Condition 2:
If $c' \sqsubseteq c$ for some a where $\gamma(a) = c$, then a is a sound albeit less precise approximation of c'.

When the two conditions are equivalent:

$$\alpha(c) \leq a' \iff c' \sqsubseteq \gamma(a)$$

we have a Galois connection.

Galois connection properties (1/2)

Observation 1: $\gamma \circ \alpha$ is extensive

Intuition: loss of information by α is sound

Observation 2: $\alpha \circ \gamma$ is reductive

Intuition: γ loses no information, i.e., α is as precise as possible

Observation 3: α and γ are monotone

Intuition: α and γ are order, i.e., soundness preserving

Galois connection properties (2/2)

Theorem. The inverse of a Galois connection is itself a Galois connection (under reverse order):

$$\frac{\langle C; \sqsubseteq \rangle \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} \langle A; \leq \rangle}{\langle A; \geq \rangle \underset{\gamma}{\overset{\alpha}{\leftrightarrows}} \langle C; \sqsupseteq \rangle}$$

Note how we have typeset the theorem as an *inference rule*.

Galois connection properties (2/2)

Theorem. The inverse of a Galois connection is itself a Galois connection (under reverse order):

$$\frac{\langle C; \sqsubseteq\rangle \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} \langle A; \leq\rangle}{\langle A; \geq\rangle \underset{\gamma}{\overset{\alpha}{\leftrightarrows}} \langle C; \sqsupseteq\rangle}$$

Note how we have typeset the theorem as an *inference rule*.

By the *duality principle* all results on posets have a dual.

Hence this extends to Galois connections if we replace

- $\sqsubseteq$, $\sqsubset$, $\bot$, $\top$, $\sqcap$, and $\sqcup$ with
- $\sqsupseteq$, $\sqsupset$, $\top$, $\bot$, $\sqcup$, and $\sqcap$

Alternative 1: Closure operators (1/2)

Definition. A function $\rho : S \to S$ on a poset $\langle S; \sqsubseteq \rangle$ is a(n upper) closure operator if ρ is monotone, extensive, and idempotent: $\forall s \in S : \rho(\rho(s)) = \rho(s)$

Similarly ρ is a *lower* closure operator if it is monotone, *reductive*, and idempotent.

Corollary. A Galois connection $\langle C; \sqsubseteq \rangle \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} \langle A; \leq \rangle$ induces

- □ an upper closure operator $\gamma \circ \alpha$ on C and
- □ a lower closure operator $\alpha \circ \gamma$ on A

Alternative 1: Closure operators (2/2)

Theorem. A closure operator $\rho : S \to S$ on a poset $\langle S; \sqsubseteq \rangle$ induces a Galois connection

$$\langle S; \sqsubseteq \rangle \underset{\rho}{\overset{1}{\leftrightarrows}} \langle \rho(S); \sqsubseteq \rangle$$

(1 being the identity function on S).

Hence it is equivalent to stay in the concrete domain and formulate abstract interpretation in terms of closure operators!

Alternative 2: Moore families

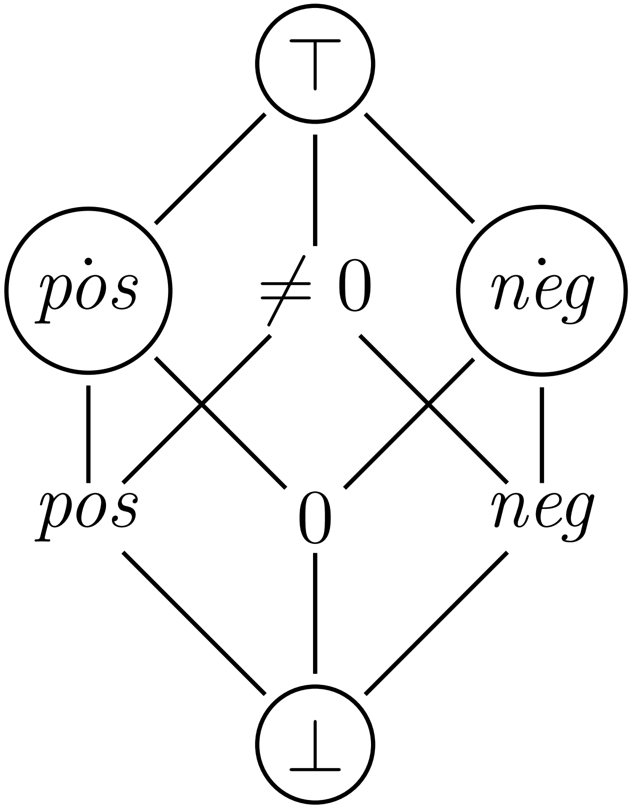
Definition. Let $\langle P; \sqsubseteq \rangle$ be a poset with a top element $\top$. A *Moore family* is a subset $S \subseteq P$ such that

- $\top \in S$
- If $X \subseteq S$ then $\sqcap X$ exists in P and $\sqcap X \in S$

Proposition. If $\langle C; \sqsubseteq \rangle \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} \langle A; \leq \rangle$ is a Galois connection and $\langle C; \sqsubseteq, \bot, \top, \sqcup, \sqcap \rangle$ is a complete lattice, then $\gamma(A) = \{\gamma(a) \mid a \in A\}$ is a Moore family.

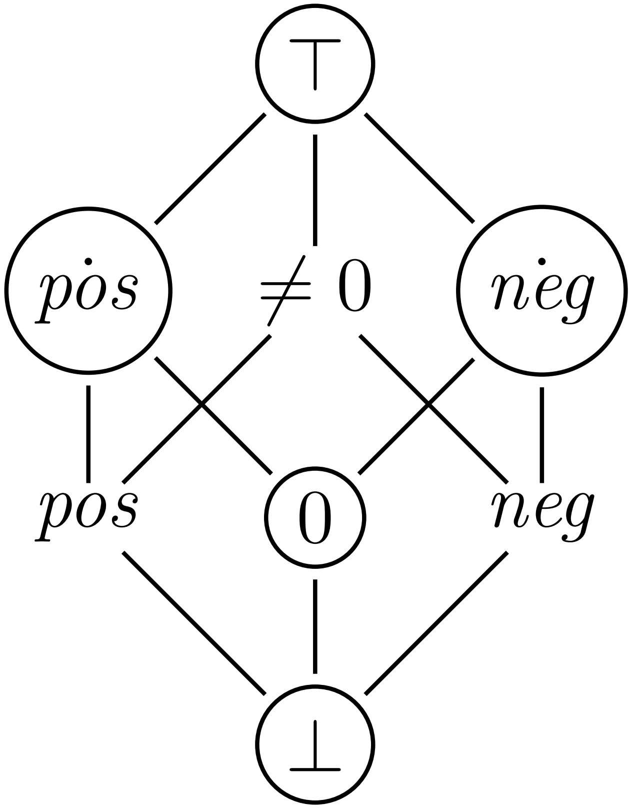
Hence, Moore families can provide a sanity check for an abstract domain.

Alternative 2: Moore family non-example



The greatest lower bound $\dot{pos} \sqcap \dot{neg}$ exists, but not in the above subset.

Alternative 2: Moore family example



The greatest lower bound $\dot{pos} \sqcap \dot{neg}$ exists, and belongs to the above subset.

More Galois connection properties

Each function uniquely determines the other:

Proposition. If $\langle C; \sqsubseteq \rangle \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} \langle A; \leq \rangle$ and $\langle C; \sqsubseteq \rangle \underset{\alpha'}{\overset{\gamma'}{\leftrightarrows}} \langle A; \leq \rangle$ then $\alpha = \alpha'$ if and only if $\gamma = \gamma'$

Each function expresses the other:

Proposition. If $\langle C; \sqsubseteq \rangle \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} \langle A; \leq \rangle$ then

- for all $c \in C : \alpha(c) = \bigwedge\{a \mid c \sqsubseteq \gamma(a)\}$
- for all $a \in A : \gamma(a) = \bigsqcup\{c \mid \alpha(c) \leq a\}$

Galois surjections and injections reloaded

Definition. A *Galois surjection* (or insertion) $\langle C; \sqsubseteq\rangle \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} \langle A; \leq\rangle$ is a Galois connection where α is surjective (equivalently γ is injective, or $\forall a \in A : \alpha \circ \gamma(a) = a$).

Definition. A *Galois injection* $\langle C; \sqsubseteq\rangle \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} \langle A; \leq\rangle$ is a Galois connection in which γ is surjective (or equivalently α is injective, or $\forall c \in C : \gamma \circ \alpha(c) = c$).

Proposition. If $\langle C; \sqsubseteq\rangle \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} \langle A; \leq\rangle$ is a Galois surjection and C is a complete lattice $\langle C; \sqsubseteq, \bot, \top, \sqcup, \sqcap\rangle$ then A is a complete lattice.

Reduction of an abstract domain

By equating abstract elements with the same concretization, we obtain a Galois surjection:

Proposition. If $\langle C; \sqsubseteq \rangle \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} \langle A; \leq \rangle$ is a Galois connection, then

- $a \equiv a' = (\gamma(a) = \gamma(a'))$ is an equivalence relation, such that
- $\langle C; \sqsubseteq \rangle \underset{\alpha_{\equiv}}{\overset{\gamma_{\equiv}}{\leftrightarrows}} \langle A/_{\equiv}; \leq_{\equiv} \rangle$ is a Galois surjection,

$$\text{where } X \leq_{\equiv} Y \text{ if } (\exists a \in X : \exists a' \in Y : a \leq a')$$
$$\alpha_{\equiv}(c) = \{a \mid a \equiv \alpha(c)\}$$
$$\gamma_{\equiv}(X) = \gamma(a) \ \text{ where } \ a \in X$$

Example: intervals

Consider the abstract domain of *intervals*.

Elements are of the form $[a, b]$ with $a, b \in \mathbb{Z} \cup \{-\infty, \infty\}$

Ordering: $[a, b] \sqsubseteq [a', b']$ if $a' \leq a \ \wedge \ b \leq b'$

Concretization: $\gamma([a, b]) = \{n \mid a \leq n \leq b\}$

All elements $[a, b]$ for which $a > b$ denote the empty set $\emptyset$. Usually this reduction has already (implicitly) taken place.

For example, $\emptyset = \gamma([32, 0]) = \gamma([5, 4]) = \emptyset$

Compositional design of Galois connections

Known composition from week 1

Theorem. The composition of two Galois connections $\langle C; \sqsubseteq \rangle \underset{\alpha_1}{\overset{\gamma_1}{\leftrightarrows}} \langle B; \subseteq \rangle$ and $\langle B; \subseteq \rangle \underset{\alpha_2}{\overset{\gamma_2}{\leftrightarrows}} \langle A; \leq \rangle$ is itself a Galois connection:

$$\langle C; \sqsubseteq \rangle \underset{\alpha_2 \circ \alpha_1}{\overset{\gamma_1 \circ \gamma_2}{\leftrightarrows}} \langle A; \leq \rangle$$

The above theorem typeset as an inference rule:

$$\frac{\langle C; \sqsubseteq \rangle \underset{\alpha_1}{\overset{\gamma_1}{\leftrightarrows}} \langle B; \subseteq \rangle \qquad \langle B; \subseteq \rangle \underset{\alpha_2}{\overset{\gamma_2}{\leftrightarrows}} \langle A; \leq \rangle}{\langle C; \sqsubseteq \rangle \underset{\alpha_2 \circ \alpha_1}{\overset{\gamma_1 \circ \gamma_2}{\leftrightarrows}} \langle A; \leq \rangle}$$

The Cartesian product of Galois connections

Theorem. Let $\langle C_1; \sqsubseteq_1\rangle \underset{\alpha_1}{\overset{\gamma_1}{\leftrightarrows}} \langle A_1; \leq_1\rangle$ and $\langle C_2; \sqsubseteq_2\rangle \underset{\alpha_2}{\overset{\gamma_2}{\leftrightarrows}} \langle A_2; \leq_2\rangle$ be Galois connections. Then we can form a Galois connection between the Cartesian product of the concrete and abstract domains:

$$\langle C_1 \times C_2; \sqsubseteq_1 \times \sqsubseteq_2\rangle \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} \langle A_1 \times A_2; \leq_1 \times \leq_2\rangle$$

where

$$\alpha(\langle c_1,\, c_2\rangle) = \langle \alpha_1(c_1),\, \alpha_2(c_2)\rangle$$
$$\gamma(\langle a_1,\, a_2\rangle) = \langle \gamma_1(a_1),\, \gamma_2(a_2)\rangle$$

The Cartesian product of Galois connections

Theorem. (same, now typeset as inference rule)

$$\frac{\langle C_1; \sqsubseteq_1 \rangle \underset{\alpha_1}{\overset{\gamma_1}{\leftrightarrows}} \langle A_1; \leq_1 \rangle \qquad \langle C_2; \sqsubseteq_2 \rangle \underset{\alpha_2}{\overset{\gamma_2}{\leftrightarrows}} \langle A_2; \leq_2 \rangle}{\langle C_1 \times C_2; \sqsubseteq_1 \times \sqsubseteq_2 \rangle \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} \langle A_1 \times A_2; \leq_1 \times \leq_2 \rangle}$$

where

$$\alpha(\langle c_1,\, c_2 \rangle) = \langle \alpha_1(c_1),\, \alpha_2(c_2) \rangle$$
$$\gamma(\langle a_1,\, a_2 \rangle) = \langle \gamma_1(a_1),\, \gamma_2(a_2) \rangle$$

The Cartesian product of Galois connections

Theorem. (same, now typeset as inference rule)

$$\frac{\langle C_1; \sqsubseteq_1 \rangle \underset{\alpha_1}{\overset{\gamma_1}{\leftrightarrows}} \langle A_1; \leq_1 \rangle \qquad \langle C_2; \sqsubseteq_2 \rangle \underset{\alpha_2}{\overset{\gamma_2}{\leftrightarrows}} \langle A_2; \leq_2 \rangle}{\langle C_1 \times C_2; \sqsubseteq_1 \times \sqsubseteq_2 \rangle \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} \langle A_1 \times A_2; \leq_1 \times \leq_2 \rangle}$$

where

$$\alpha(\langle c_1,\, c_2 \rangle) = \langle \alpha_1(c_1),\, \alpha_2(c_2) \rangle$$
$$\gamma(\langle a_1,\, a_2 \rangle) = \langle \gamma_1(a_1),\, \gamma_2(a_2) \rangle$$

Example: we can abstract a pair of natural number sets to a Parity pair:

$$\frac{\langle \wp(\mathbb{N}); \subseteq \rangle \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} \langle Par; \sqsubseteq \rangle \qquad \langle \wp(\mathbb{N}); \subseteq \rangle \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} \langle Par; \sqsubseteq \rangle}{\langle \wp(\mathbb{N}) \times \wp(\mathbb{N}); \subseteq \times \subseteq \rangle \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} \langle Par \times Par; \sqsubseteq \times \sqsubseteq \rangle}$$

Reduced product

A *reduced product* improves two (or more) abstractions of the same domain:

Theorem. Let $\langle C; \sqsubseteq \rangle \underset{\alpha_1}{\overset{\gamma_1}{\leftrightarrows}} \langle A_1; \leq_1 \rangle$ and $\langle C; \sqsubseteq \rangle \underset{\alpha_2}{\overset{\gamma_2}{\leftrightarrows}} \langle A_2; \leq_2 \rangle$ be Galois connections between complete lattices. Then the reduced product is a Galois surjection:

$$\langle C; \sqsubseteq \rangle \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} \langle A_1 \times A_2; \leq_1 \times \leq_2 \rangle$$

$$\text{where} \qquad \begin{aligned} \alpha(c) &= \langle \alpha_1(c),\, \alpha_2(c) \rangle \\ \gamma(\langle a_1,\, a_2 \rangle) &= \gamma_1(a_1) \sqcap \gamma_2(a_2) \end{aligned}$$

Note: the paper contains a much more general version

Example: reduced product

Imagine we abstract an integer variable x using both *Sign* and *Parity* abstract domains.

If $\mathrm{x} = 0$ from the Sign domain ($\gamma(0) = \{0\}$) and x is odd from the Parity domain ($\gamma(odd) = \{1, 3, 5, \dots\}$), we can gain information by combining them.

A reduction tells us, no integers are 0 *and* odd, hence we reduce to $\gamma(0) \cap \gamma(odd) = \emptyset$.

Note: Not transferring information from one domain to the other corresponds to running the analyses separately.

From concrete to abstract semantics

Correctness, optimality, and completeness

Definition. If $\alpha \circ F \mathrel{\dot{\leq}} F^{\#} \circ \alpha$ we say $F^{\#}$ is a (locally) correct (or sound) approximation of F

Definition. If $F^{\#} = \alpha \circ F \circ \gamma$ we say $F^{\#}$ is an optimal approximation of F

Intuitively we can't do better with the available abstract information.

Definition. If $\alpha \circ F = F^{\#} \circ \alpha$ we say $F^{\#}$ is a complete approximation of F (no loss of information)

Intuitively we can't do better with the available concrete information.

These definitions generalize to n-ary functions F and $F^{\#}$.

Example

Consider addition over the abstract Sign domain.

Addition is not complete, e.g.:

$$
\begin{aligned}
0 &= \alpha(42 + (-42)) \\
&\sqsubseteq \alpha(42) + \alpha(-42) = pos + neg = \top
\end{aligned}
$$

However addition is an optimal approximation, e.g.:

$$
\begin{aligned}
&\alpha(\gamma(pos) + \gamma(neg)) \\
&= \alpha(\{n \mid n \geq 0\} + \{n \mid n \leq 0\}) \\
&= \alpha(\{n + n' \mid n \geq 0 \ \wedge \ n' \leq 0\}) \\
&= \alpha(\mathbb{Z}) = \top
\end{aligned}
$$

From concrete to abstract operator, constructively

These definitions lead us to the following two “recipes” for approximating a concrete operator F:

1. Push α’s under the function definition:

$$\alpha \circ F(c) = \cdots = F^{\#}(\alpha(c))$$

   (geared towards complete approximation, however it is still correct/sound if we upward judge underway)

2. Compose F with α and γ:

$$\alpha \circ F \circ \gamma(a) = \cdots = F^{\#}(a)$$

   (geared towards optimal approximation, however it is still correct/sound if we upward judge underway)

Fun with the three counter machine

Recall Plotkin's three counter machine (1/2)

There are 3 variables (or registers):

$$var \in Var = \{\mathtt{x}, \mathtt{y}, \mathtt{z}\} \qquad \text{(variables)}$$

$$\begin{aligned} Inst ::= {} & \mathtt{inc}\ var \\ \mid {} & \mathtt{dec}\ var \\ \mid {} & \mathtt{zero}\ var\ m\ \mathtt{else}\ n \\ \mid {} & \mathtt{stop} \end{aligned} \qquad \text{(instructions)}$$

$$P = Inst^* \qquad \text{(programs)}$$

$$pc \in PC = \mathbb{N} \qquad \text{(program counter)}$$

$$States = PC \times \mathbb{N}_0 \times \mathbb{N}_0 \times \mathbb{N}_0 \qquad \text{(states)}$$

Initial states: $\{\langle 1,\ i,\ 0,\ 0\rangle \mid i \in \mathbb{N}_0\}$ (for program P with input i)

Final states: $\{\langle pc,\ 0,\ yv,\ 0\rangle \mid pc \in PC\ \wedge\ yv \in \mathbb{N}_0\ \wedge\ P_{pc} = \mathtt{stop}\}$
(with yv being the result)

Recall Plotkin's three counter machine (2/2)

Transition relation:

$$
\begin{array}{rcll}
\langle pc,\, xv,\, yv,\, zv\rangle & \longrightarrow & \langle pc+1,\, xv+1,\, yv,\, zv\rangle & \text{if } P_{pc} = \texttt{inc x} \\
- & \longrightarrow & \langle pc+1,\, xv,\, yv+1,\, zv\rangle & \text{if } P_{pc} = \texttt{inc y} \\
- & \longrightarrow & \langle pc+1,\, xv,\, yv,\, zv+1\rangle & \text{if } P_{pc} = \texttt{inc z} \\
\\
\langle pc,\, xv,\, yv,\, zv\rangle & \longrightarrow & \langle pc+1,\, xv-1,\, yv,\, zv\rangle & \text{if } P_{pc} = \texttt{dec x} \ \wedge\ xv > 0 \\
- & \longrightarrow & \langle pc+1,\, xv,\, yv-1,\, zv\rangle & \text{if } P_{pc} = \texttt{dec y} \ \wedge\ yv > 0 \\
- & \longrightarrow & \langle pc+1,\, xv,\, yv,\, zv-1\rangle & \text{if } P_{pc} = \texttt{dec z} \ \wedge\ zv > 0 \\
\\
\langle pc,\, xv,\, yv,\, zv\rangle & \longrightarrow & \langle pc',\, xv,\, yv,\, zv\rangle & \text{if } P_{pc} = \texttt{zero x}\ pc'\ \texttt{else}\ pc'' \ \wedge\ xv = 0 \\
- & \longrightarrow & \langle pc'',\, xv,\, yv,\, zv\rangle & \text{if } P_{pc} = \texttt{zero x}\ pc'\ \texttt{else}\ pc'' \ \wedge\ xv \neq 0 \\
\\
\langle pc,\, xv,\, yv,\, zv\rangle & \longrightarrow & \langle pc',\, xv,\, yv,\, zv\rangle & \text{if } P_{pc} = \texttt{zero y}\ pc'\ \texttt{else}\ pc'' \ \wedge\ yv = 0 \\
- & \longrightarrow & \langle pc'',\, xv,\, yv,\, zv\rangle & \text{if } P_{pc} = \texttt{zero y}\ pc'\ \texttt{else}\ pc'' \ \wedge\ yv \neq 0 \\
\\
\langle pc,\, xv,\, yv,\, zv\rangle & \longrightarrow & \langle pc',\, xv,\, yv,\, zv\rangle & \text{if } P_{pc} = \texttt{zero z}\ pc'\ \texttt{else}\ pc'' \ \wedge\ zv = 0 \\
- & \longrightarrow & \langle pc'',\, xv,\, yv,\, zv\rangle & \text{if } P_{pc} = \texttt{zero z}\ pc'\ \texttt{else}\ pc'' \ \wedge\ zv \neq 0
\end{array}
$$

Plotkin's three counter machine in ASCII...

Transition relation:

```
<pc, xv, yv, zv> --> <pc+1, xv+1, yv, zv>                  if P_pc = inc x
        -        --> <pc+1, xv, yv+1, zv>                  if P_pc = inc y
        -        --> <pc+1, xv, yv, zv+1>                  if P_pc = inc z

<pc, xv, yv, zv> --> <pc+1, xv-1, yv, zv>          if P_pc = dec x /\ xv>0
        -        --> <pc+1, xv, yv-1, zv>          if P_pc = dec y /\ yv>0
        -        --> <pc+1, xv, yv, zv-1>          if P_pc = dec z /\ zv>0

<pc, xv, yv, zv> --> <pc', xv, yv, zv>      if P_pc = zero x pc' else pc''
                                               /\ xv=0
        -        --> <pc'', xv, yv, zv>     if P_pc = zero x pc' else pc''
                                               /\ xv<>0

<pc, xv, yv, zv> --> <pc', xv, yv, zv>      if P_pc = zero y pc' else pc''
                                               /\ yv=0
        -        --> <pc'', xv, yv, zv>     if P_pc = zero y pc' else pc''
                                               /\ yv<>0

<pc, xv, yv, zv> --> <pc', xv, yv, zv>      if P_pc = zero z pc' else pc''
                                               /\ zv=0
        -        --> <pc'', xv, yv, zv>     if P_pc = zero z pc' else pc''
                                               /\ zv<>0
```

Implementation of the three counter machine

Quick tour of implementation:

- ☐ AST
- ☐ Lexer
- ☐ Parser
- ☐ Wellformedness (checks out of bounds)
- ☐ Interpreter

Each of the above reside in their own module (and file).

To build from scratch run:

`make depend` and `make`

An epigram from Perlis

Beware of the Turing tar-pit in which everything is possible but nothing of interest is easy.

— Alan Perlis

Formulating the collecting semantics

Recall the reachable states collecting semantics:

$$T(\Sigma) = I \cup \{\sigma \mid \exists \sigma' \in \Sigma : \sigma' \to \sigma\}$$

Let's write the specialized version...

Abstracting the collecting semantics

We abstract the collecting semantics to a set valued function using the well-known Galois connection:

$$\langle \wp(A \times B); \subseteq, \emptyset, A \times B, \cup, \cap \rangle \xleftarrow[\alpha]{\gamma} \langle A \to \wp(B); \dot{\subseteq}, \lambda x.\emptyset, \lambda x.B, \dot{\cup}, \dot{\cap} \rangle$$

$$\text{where} \qquad \begin{aligned} \alpha(R) &= \lambda a.\{b \mid (a, b) \in R\} \\ \gamma(F) &= \{(a, b) \mid b \in F(a)\} \end{aligned}$$

Note: in our case $A = PC$ and $B = \mathbb{N}_0 \times \mathbb{N}_0 \times \mathbb{N}_0$.

We will use the first recipe of “pushing alphas”. . .

Result

```
T#(S#) =  Ø. [1 -> { <i,0,0> | i in N_0 }]

  U.
            U.         Ø. [pc+1 -> { <xv+1, yv, zv> }]
   { <xv, yv, zv> } C S#(pc)
      P_pc = inc x               (...and for y and z)
  U.
            U.         Ø. [pc+1 -> { <xv-1, yv, zv> }]
   { <xv, yv, zv> } C S#(pc)
      P_pc = dec x
          xv>0                   (...and for y and z)
  U.
            U.         Ø. [pc' -> { <xv, yv, zv> }]
   { <xv, yv, zv> } C S#(pc)
  P_pc = zero x pc' else pc''
           xv=0                  (...and for y and z)
  U.
            U.         Ø. [pc'' -> { <xv, yv, zv> }]
   { <xv, yv, zv> } C S#(pc)
  P_pc = zero x pc' else pc''
          xv<>0                  (...and for y and z)
```

What happened?

We systematically massaged the transition function of the collecting semantics

$$T : \wp(PC \times \mathbb{N}_0 \times \mathbb{N}_0 \times \mathbb{N}_0) \to \wp(PC \times \mathbb{N}_0 \times \mathbb{N}_0 \times \mathbb{N}_0)$$

into a transition function over a related domain

$$T\# : (PC \to \wp(\mathbb{N}_0 \times \mathbb{N}_0 \times \mathbb{N}_0)) \to (PC \to \wp(\mathbb{N}_0 \times \mathbb{N}_0 \times \mathbb{N}_0))$$

by surfing on the Galois connections.

Note: a least fixed point of the resulting function is still not computable (in general), so we are not quite there yet. . .

Cliffhanger...

To be continued...

Summary

Summary

We've taken a more in depth look at AI based on Cousot-Cousot:JLP92.

- □ Foundations: Fixed points, Galois connections, . . .
- □ The Galois approach and friends: closure operators, Moore families, . . .
- □ From collecting semantics to analysis
- □ The first step towards analysing Plotkin's 3 counter machine