Numerical and Structural Abstractions

Jan Midtgaard

Week 5, Abstract Interpretation

Aarhus University, Q4 - 2012

Last time

More approximation methods for abstract interpretation:

- □ Partitioning
- □ Relational and attribute independent analysis
- □ Inducing, abstracting, approximating fixed points
- □ Widening, narrowing
- □ Forwards/backwards analysis

+ analysis of Plotkin's three counter machine

Today

A catalogue of abstractions

- □ Toolbox abstractions
- □ Structural abstractions: sums, pairs/tuples, . . .
- □ Numerical abstractions: constants, intervals, polyhedra
- □ Concretization-based abstract interpretation, briefly

A retrospective on the 3 counter machine analysis, incl. constraint extraction

Toolbox abstractions

Warm up: Collapsing abstractions

The collapsing abstraction into a two element lattice:

$$\begin{aligned} \alpha(\emptyset) &= \bot \\ \alpha(S) &= \top \quad \text{if} \quad S \neq \emptyset \\ \gamma(\bot) &= \emptyset \\ \gamma(\top) &= S \end{aligned}$$

$$\langle \wp(S); \subseteq \rangle \xleftrightarrows[\alpha]{\gamma} \langle \{\top, \bot\}; \sqsubseteq \rangle$$

is slightly better than the completely collapsing abstraction:

$$\begin{aligned} \alpha(S) &= \bot \\ \gamma(\bot) &= S \end{aligned}$$

$$\langle \wp(S); \subseteq \rangle \xleftrightarrows[\alpha]{\gamma} \langle \{\bot\}; \sqsubseteq \rangle$$

Subset abstraction

Given a set C and a strict subset $A \subset C$ hereof, the restriction to the subset induces a Galois connection:

$$\langle \wp(C); \subseteq \rangle \underset{\alpha_\subset}{\overset{\gamma_\subset}{\leftrightarrows}} \langle \wp(A); \subseteq \rangle$$

$$\alpha_\subset(X) = X \cap A$$

$$\gamma_\subset(Y) = Y \cup (C \setminus A)$$

For example, in a *control-flow analysis* of untyped functional programs one can choose to focus on functional values (closures) and not model numbers:

$$\langle \wp(Clo + Num); \subseteq \rangle \underset{\alpha_\subset}{\overset{\gamma_\subset}{\leftrightarrows}} \langle \wp(Clo); \subseteq \rangle$$

(Note: by a sum $A + B$ we mean the disjoint union)

Elementwise abstraction

Let an elementwise operator @ $: C \to A$ be given.
Define

$$\alpha(P) = \{@(p) \mid p \in P\}$$
$$\gamma(Q) = \{p \mid @(p) \in Q\}$$

Then

$$\langle \wp(C); \subseteq \rangle \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} \langle \wp(A); \subseteq \rangle$$

In particular, if @ is onto, we have

$$\langle \wp(C); \subseteq \rangle \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} \langle \wp(A); \subseteq \rangle$$

For example, Parity is isomorphic to an elementwise abstraction.
Q: what would A and @ be in this case?

Structural abstractions

Structural?

> How is $State^{\#}$ constructed? It is possible to invent $State^{\#}$, and then the pair of adjoined functions. Another approach consists in inducing $State^{\#}$ from the structure of $State$.
>
> —Alain Deutsch, POPL'90

Abstracting sums as a product

We can abstract sums by first utilizing a simple isomorphism:

$$\langle \wp(A+B); \subseteq \rangle \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} \langle \wp(A) \times \wp(B); \subseteq_\times \rangle$$

where

$$\alpha(S) = (\{a \mid a \in S \cap A\}, \{b \mid b \in S \cap B\})$$

This isomorphism will typically enable further approximation.

For example, the values of a mini-Scheme language could be such a disjoint sum: closure or number

Componentwise abstraction

We can abstract a Cartesian product (e.g., the outcome of the previous isomorphism) componentwise:

$$\frac{\langle \wp(C_i); \subseteq\rangle \underset{\alpha_i}{\overset{\gamma_i}{\leftrightarrows}} \langle A_i; \sqsubseteq_i\rangle \qquad i \in \{1, \ldots, n\}}{\langle \wp(C_1) \times \ldots \times \wp(C_n); \subseteq_\times\rangle \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} \langle A_1 \times \ldots \times A_n; \sqsubseteq_\times\rangle}$$

with

$$\alpha(\langle X_1, \ldots, X_n\rangle) = \langle \alpha_1(X_1), \ldots, \alpha_n(X_n)\rangle$$
$$\gamma(\langle x_1, \ldots, x_n\rangle) = \langle \gamma_1(x_1), \ldots, \gamma_n(x_n)\rangle$$

and writing $\sqsubseteq_\times$ for componentwise inclusion.

For example, last week we used the “triple version” for abstracting the 3 Counter Machine memory.

Abstracting pairs, coarsely

We can approximate a set-of-pairs by an abstract pair:

$$\frac{\langle \wp(C_1); \subseteq \rangle \xrightleftharpoons[\alpha_1]{\gamma_1} \langle A_1; \leq_1 \rangle \qquad \langle \wp(C_2); \subseteq \rangle \xrightleftharpoons[\alpha_2]{\gamma_2} \langle A_2; \leq_2 \rangle}{\langle \wp(C_1 \times C_2); \subseteq \rangle \xrightleftharpoons[\alpha]{\gamma} \langle A_1 \times A_2; \leq_\times \rangle}$$

where

$$\alpha(S) = \langle \alpha_1(\{a \mid (a,b) \in S\}),\ \alpha_2(\{b \mid (a,b) \in S\}) \rangle$$

For example, we used this approach to abstract the three memory registers of the 3CM.

Abstracting pairs, better

Utilizing the well-known isomorphism

$$\langle \wp(C_1 \times C_2); \subseteq \rangle \xrightleftharpoons[\alpha]{\gamma} \langle C_1 \to \wp(C_2); \dot{\subseteq} \rangle$$

we can approximate the set-of-pairs as a function between abstract domains:

$$\frac{\langle \wp(C_1); \subseteq \rangle \xrightleftharpoons[\alpha_1]{\gamma_1} \langle A_1; \leq_1 \rangle \qquad \langle \wp(C_2); \subseteq \rangle \xrightleftharpoons[\alpha_2]{\gamma_2} \langle A_2; \leq_2 \rangle}{\langle \wp(C_1 \times C_2); \subseteq \rangle \xrightleftharpoons[\alpha]{\gamma} \langle A_1 \to A_2; \dot{\leq}_2 \rangle}$$

where

$$\alpha(S) = \dot{\bigsqcup} \{ [\alpha_1(\{a\}) \mapsto \alpha_2(\{b\})] \mid \langle a,\, b \rangle \in S \}$$

Abstracting pairs, relationally

Finally we can go all-in and approximate the set-of-pairs as an abstract set-of-pairs:

$$\frac{\langle \wp(C_1); \subseteq \rangle \underset{\alpha_1}{\overset{\gamma_1}{\leftrightarrows}} \langle A_1; \leq_1 \rangle \qquad \langle \wp(C_2); \subseteq \rangle \underset{\alpha_2}{\overset{\gamma_2}{\leftrightarrows}} \langle A_2; \leq_2 \rangle}{\langle \wp(C_1 \times C_2); \subseteq \rangle \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} \langle \wp(A_1 \times A_2)/_{\equiv}; \subseteq \rangle}$$

where $\quad \alpha(S) = \{\langle \alpha_1(\{a\}),\, \alpha_2(\{b\})\rangle \mid \langle a,\, b\rangle \in S\}$

Note: this requires a domain reduction, equating all elements with the same meaning, e.g., in $\wp(Par \times Par)$, $\{\langle \top,\, even\rangle\} \equiv \{\langle odd,\, even\rangle, \langle even,\, even\rangle\}$.

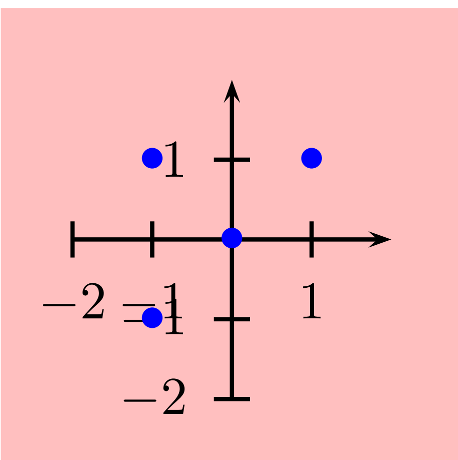
Perhaps a fun project abstracting the 3CM in this manner?

Comparing the three pair abstractions

Suppose we abstract the signs of the following set

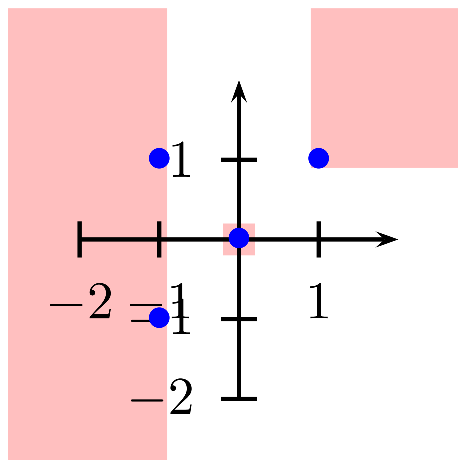
$$S = \{\langle -1,\, -1\rangle, \langle 0,\, 0\rangle, \langle 1,\, 1\rangle, \langle -1,\, 1\rangle\}$$

coarsely:



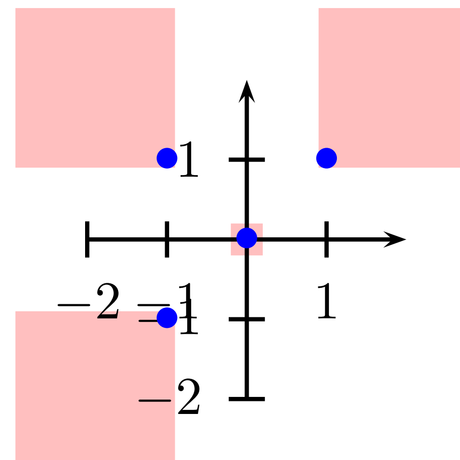
$$\alpha(S) = \langle \top,\, \top\rangle$$

better:



$$\alpha(S) = [\mathit{neg} \mapsto \top,\ 0 \mapsto 0,\ \mathit{pos} \mapsto \mathit{pos}]$$

relationally:



$$\alpha(S) = \{\langle \mathit{neg},\, \mathit{neg}\rangle, \langle 0,\, 0\rangle, \langle \mathit{pos},\, \mathit{pos}\rangle, \langle \mathit{neg},\, \mathit{pos}\rangle\}$$

Abstracting monotone functions

Similar to the 'better abstraction' of pairs, we can approximate monotone functions by monotone abstract functions:

$$\frac{\langle C_1; \subseteq_1 \rangle \underset{\alpha_1}{\overset{\gamma_1}{\leftrightarrows}} \langle A_1; \leq_1 \rangle \qquad \langle C_2; \subseteq_2 \rangle \underset{\alpha_2}{\overset{\gamma_2}{\leftrightarrows}} \langle A_2; \leq_2 \rangle}{\langle C_1 \xrightarrow{m} C_2; \dot{\subseteq}_2 \rangle \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} \langle A_1 \xrightarrow{m} A_2; \dot{\leq}_2 \rangle}$$

where $X \xrightarrow{m} Y$ are the monotone functions from X to Y and

$$\alpha(f) = \alpha_2 \circ f \circ \gamma_1$$
$$\gamma(g) = \gamma_2 \circ g \circ \alpha_1$$

Abstracting sequences

We can abstract a set of sequences (rather crudely) by collapsing their elements:

$$\frac{\langle \wp(C); \subseteq \rangle \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} \langle A; \leq \rangle}{\langle \wp(C^*); \subseteq \rangle \underset{\alpha^*}{\overset{\gamma^*}{\leftrightarrows}} \langle A; \leq \rangle}$$

where

$$\alpha^*(S) = \alpha(\{x \mid x \in s \ \wedge \ s \in S\})$$

Numerical abstractions

Numerical abstractions

We've already come across a few numerical abstract domains: parity, signs, intervals, . . .

All of these were attribute independent (or non-relational): they don't express relations between (the values of) variables.

Let's recap what we have seen and supplement with some new ones, both non-relational and relational.

What is a numerical abstract domain?

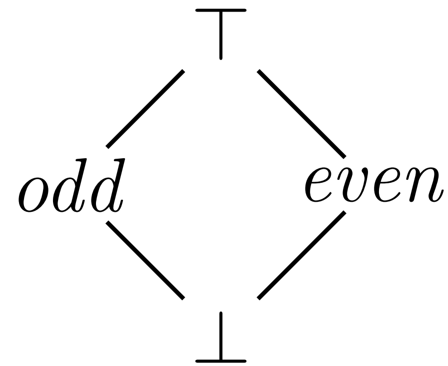
A computer-representable property, with

- top and bottom: $\top$, $\bot$
- join, meet, and comparison operators: $\sqcup$, $\sqcap$, and $\sqsubseteq$
- widening and narrowing operators (optional, for domains with infinite strictly incr./decr. chains)
- some primitive operations: `+`, `-`, `*`, `/`
- other basic operations: test, assignment
- with matching backwards operations (optional, for (forwards/) backwards analysis)
- a γ-function mapping elements to their meaning (mathematical, not necessarily computable)

The parity domain

$$Par = \{\top, odd, even, \bot\}$$

$$\langle \wp(\mathbb{N}_0); \subseteq \rangle \xrightleftharpoons[\alpha]{\gamma} \langle Par; \sqsubseteq \rangle$$



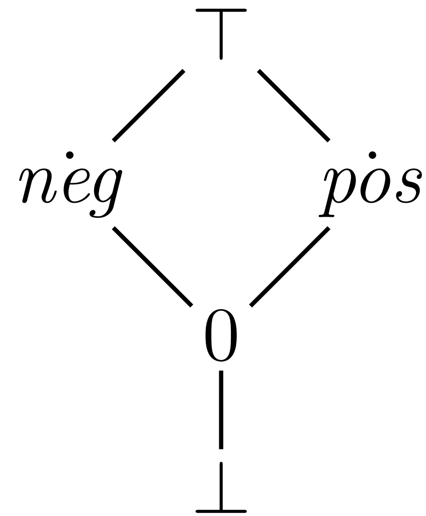
where

$$\begin{aligned} \gamma(\bot) &= \emptyset \\ \gamma(odd) &= \{n \in \mathbb{N}_0 \mid n \mod 2 = 1\} \\ \gamma(even) &= \{n \in \mathbb{N}_0 \mid n \mod 2 = 0\} \\ \gamma(\top) &= \mathbb{N}_0 \end{aligned}$$

A simple sign domain

$$Sign = \{\top, p\dot{o}s, n\dot{e}g, 0, \bot\}$$

$$\langle \wp(\mathbb{Z}); \subseteq \rangle \overset{\gamma}{\underset{\alpha}{\leftrightarrows}} \langle Sign; \sqsubseteq \rangle$$



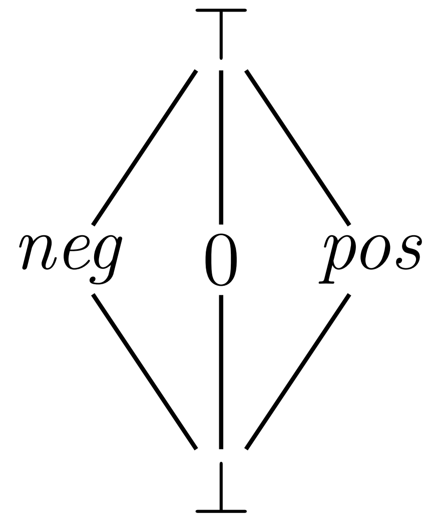
where

$$\begin{aligned} \gamma(\bot) &= \emptyset \\ \gamma(0) &= \{0\} \\ \gamma(p\dot{o}s) &= \{n \in \mathbb{Z} \mid n \geq 0\} \\ \gamma(n\dot{e}g) &= \{n \in \mathbb{Z} \mid n \leq 0\} \\ \gamma(\top) &= \mathbb{Z} \end{aligned}$$

Another simple sign domain

$$Sign = \{\top, pos, neg, 0, \bot\}$$

$$\langle \wp(\mathbb{Z}); \subseteq \rangle \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} \langle Sign; \sqsubseteq \rangle$$



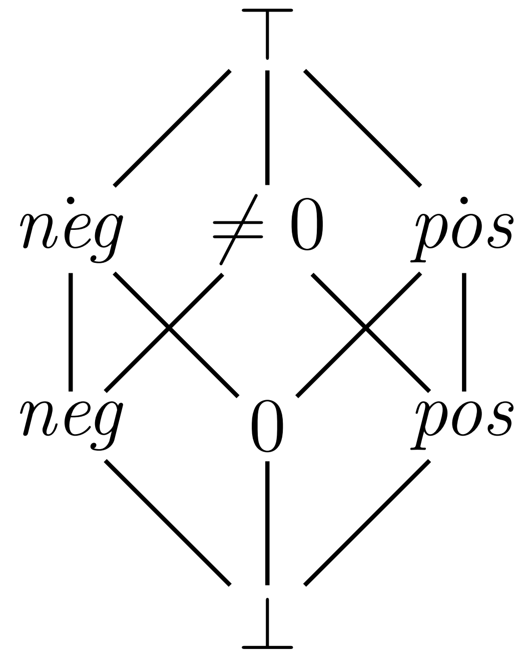
where

$$\begin{aligned}
\gamma(\bot) &= \emptyset \\
\gamma(0) &= \{0\} \\
\gamma(pos) &= \{n \in \mathbb{Z} \mid n > 0\} \\
\gamma(neg) &= \{n \in \mathbb{Z} \mid n < 0\} \\
\gamma(\top) &= \mathbb{Z}
\end{aligned}$$

The improved sign domain

$$Sign = \{\top, \neq 0, p\dot{o}s, n\dot{e}g, pos, neg, 0, \bot\}$$

$$\langle \wp(\mathbb{Z}); \subseteq \rangle \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} \langle Sign; \sqsubseteq \rangle$$



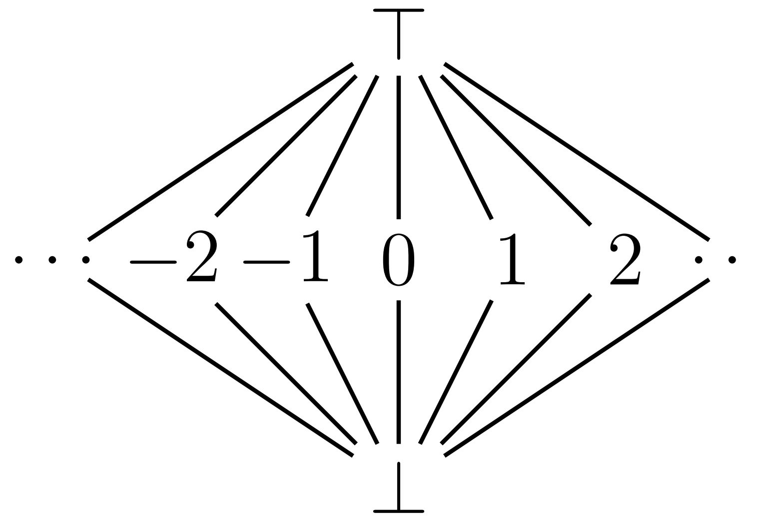
where

$$
\begin{aligned}
\gamma(\bot) &= \emptyset \\
\gamma(0) &= \{0\} \\
\gamma(pos) &= \{n \in \mathbb{Z} \mid n > 0\} \\
\gamma(neg) &= \{n \in \mathbb{Z} \mid n < 0\} \\
\gamma(p\dot{o}s) &= \{n \in \mathbb{Z} \mid n \geq 0\} \\
\gamma(n\dot{e}g) &= \{n \in \mathbb{Z} \mid n \leq 0\} \\
\gamma(\neq 0) &= \{n \in \mathbb{Z} \mid n \neq 0\} \\
\gamma(\top) &= \mathbb{Z}
\end{aligned}
$$

The constant propagation domain (Kildall:73)

$$Const = \mathbb{Z} \cup \{\top, \bot\}$$

$$\langle \wp(\mathbb{Z}); \subseteq \rangle \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} \langle Const; \sqsubseteq \rangle$$



where

$$\gamma(\top) = \mathbb{Z} \qquad \alpha(\{n_1, n_2, \ldots\}) = \top$$
$$\gamma(n) = \{n\} \qquad \alpha(\{n\}) = n$$
$$\gamma(\bot) = \emptyset \qquad \alpha(\emptyset) = \bot$$

Simple congruences (Granger'89)

$$Cong = \{\bot\} \cup \{a + b\mathbb{Z} \mid a, b \in \mathbb{Z} : (b = 0) \;\vee\; (0 \leq a < b)\}$$

$$\langle \wp(\mathbb{Z}); \subseteq \rangle \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} \langle Cong; \sqsubseteq \rangle$$

$$\gamma(\bot) = \emptyset$$
$$\gamma(a + b\mathbb{Z}) = \{a + bz \mid z \in \mathbb{Z}\}$$

Ordering:

$$\bot \sqsubseteq (a + b\mathbb{Z})$$
$$(a + b\mathbb{Z}) \sqsubseteq (a' + b'\mathbb{Z}) \iff (b' \mid \gcd(|a - a'|, b))$$

```
            0+1Z
          /  |    \
   0+2Z   1+2Z   1+3Z ...
    / \       \   /
... ...    1+6Z

     ... ... ...

... 0+0Z 1+0Z 2+0Z ...
   \   \   |   /   /
          bot
```

$$x \equiv a \mod b$$

Simple congruences, continued

Join:
$$\begin{aligned} \bot \sqcup (a + b\mathbb{Z}) &= a + b\mathbb{Z} \\ (a + b\mathbb{Z}) \sqcup \bot &= a + b\mathbb{Z} \\ (a + b\mathbb{Z}) \sqcup (a' + b'\mathbb{Z}) &= (\min(a, a') + \gcd(|a - a'|, b, b')\mathbb{Z}) \end{aligned}$$

Note: there are no infinite, strictly increasing chains.
However there are infinite, strictly decreasing chains:

$$0 + 1\mathbb{Z} \sqsupset 1 + 2\mathbb{Z} \sqsupset 1 + 6\mathbb{Z} \sqsupset 1 + 12\mathbb{Z} \sqsupset \dots$$

hence we may need a narrowing…

Simple congruences, continued

Join:
$$\bot \sqcup (a + b\mathbb{Z}) = a + b\mathbb{Z}$$
$$(a + b\mathbb{Z}) \sqcup \bot = a + b\mathbb{Z}$$
$$(a + b\mathbb{Z}) \sqcup (a' + b'\mathbb{Z}) = (\min(a, a') + \gcd(|a - a'|, b, b')\mathbb{Z})$$

Note: there are no infinite, strictly increasing chains.
However there are infinite, strictly decreasing chains:

$$0 + 1\mathbb{Z} \sqsupset 1 + 2\mathbb{Z} \sqsupset 1 + 6\mathbb{Z} \sqsupset 1 + 12\mathbb{Z} \sqsupset \ldots$$

hence we may need a narrowing...

Q: what do the elements $0 + 2\mathbb{Z}$ and $1 + 2\mathbb{Z}$ represent together with $\bot$ and $0 + 1\mathbb{Z}$?

Simple congruences, continued

Join:
$$\bot \sqcup (a + b\mathbb{Z}) = a + b\mathbb{Z}$$
$$(a + b\mathbb{Z}) \sqcup \bot = a + b\mathbb{Z}$$
$$(a + b\mathbb{Z}) \sqcup (a' + b'\mathbb{Z}) = (\min(a, a') + \gcd(|a - a'|, b, b')\mathbb{Z})$$

Note: there are no infinite, strictly increasing chains.
However there are infinite, strictly decreasing chains:

$$0 + 1\mathbb{Z} \sqsupset 1 + 2\mathbb{Z} \sqsupset 1 + 6\mathbb{Z} \sqsupset 1 + 12\mathbb{Z} \sqsupset \dots$$

hence we may need a narrowing...

Q: what do the elements $0 + 2\mathbb{Z}$ and $1 + 2\mathbb{Z}$ represent together with $\bot$ and $0 + 1\mathbb{Z}$?

Q: what about $\dots, 0 + 0\mathbb{Z}, 1 + 0\mathbb{Z}, 2 + 0\mathbb{Z}, 3 + 0\mathbb{Z}, \dots$ together with $\bot$ and $0 + 1\mathbb{Z}$?

Simple congruence operations

The arithmetic operators over congruences, e.g., addition:

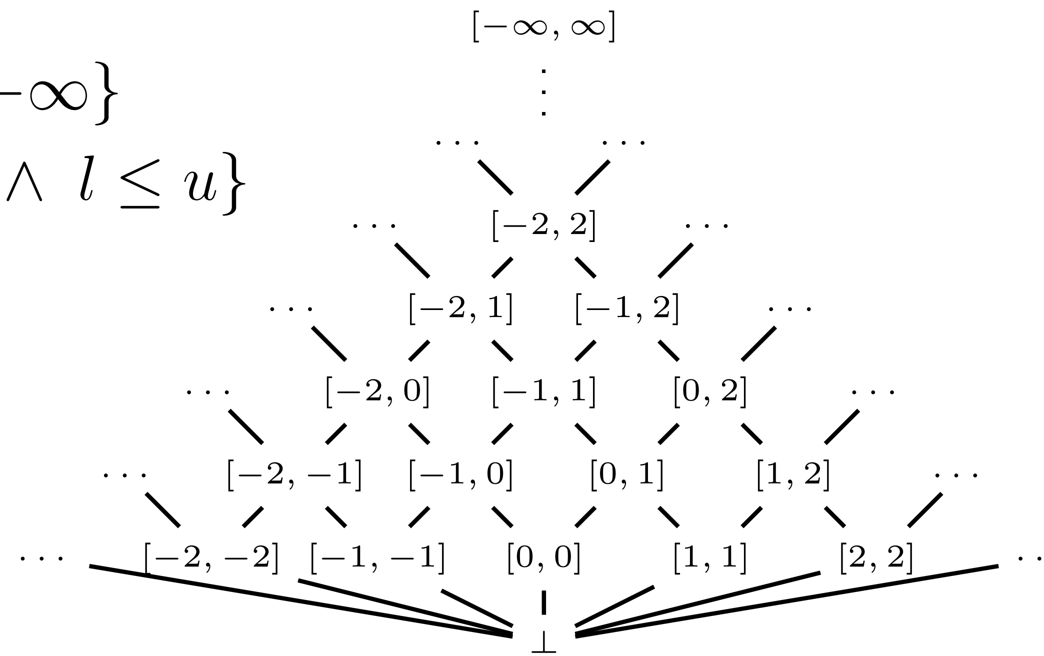
$$\begin{aligned}
(a + b\mathbb{Z}) + \perp &= \perp \\
\perp + (a + b\mathbb{Z}) &= \perp \\
(a + b\mathbb{Z}) + (c + d\mathbb{Z}) &= ((a + c) \mod \gcd(b, d)) + \gcd(b, d)\mathbb{Z}
\end{aligned}$$

and multiplication:

$$\begin{aligned}
(a + b\mathbb{Z}) * \perp &= \perp \\
\perp * (a + b\mathbb{Z}) &= \perp \\
(a + b\mathbb{Z}) * (c + d\mathbb{Z}) &= (ac \mod \gcd(ad, bc, bd)) \\
&\quad + \gcd(ad, bc, bd)\mathbb{Z}
\end{aligned}$$

Intervals (Moore'66, Cousot-Cousot'76)

$$Interval = \{\bot\} \cup \{[l,u] \mid l \in \mathbb{Z} \cup \{-\infty\} \wedge u \in \mathbb{Z} \cup \{+\infty\} \wedge l \leq u\}$$



$$\langle \wp(\mathbb{Z}); \subseteq \rangle \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} \langle Interval; \sqsubseteq \rangle$$

$$\gamma(\bot) = \emptyset \qquad \alpha(\emptyset) = \bot$$

$$\gamma([a,b]) = \{n \in \mathbb{Z} \mid a \leq n \leq b\} \qquad \alpha(S) = [\min S, \max S]$$

Note: intervals over $\mathbb{R}$ also work, however over $\mathbb{Q}$ the resulting domain is not complete.

Intervals, continued (2/3)

Least upper bounds:

$$
\begin{aligned}
X \sqcup \bot &= X \\
\bot \sqcup Y &= Y \\
[a, b] \sqcup [c, d] &= [\min(a, c), \max(b, d)]
\end{aligned}
$$

Greatest lower bounds:

$$
\begin{aligned}
X \sqcap \bot &= \bot \\
\bot \sqcap Y &= \bot \\
[a, b] \sqcap [c, d] &= \begin{cases} [\max(a, c), \min(b, d)] & \text{if} \quad \max(a, c) \leq \min(b, d) \\ \bot & \text{otherwise} \end{cases}
\end{aligned}
$$

Intervals, continued (3/3)

Interval addition:

$$
\begin{aligned}
\bot + X &= \bot \\
X + \bot &= \bot \\
[a, b] + [c, d] &= [a + c, b + d]
\end{aligned}
$$

Widening and narrowing:

$$
\begin{aligned}
\bot \mathbin{\triangledown} I &= I & I \mathbin{\triangledown} \bot &= I \\
[a, b] \mathbin{\triangledown} [c, d] &= \left[\begin{cases} -\infty & c < a \\ a & c \geq a \end{cases}, \begin{cases} +\infty & d > b \\ b & d \leq b \end{cases} \right] \\
\bot \mathbin{\vartriangle} I &= \bot & I \mathbin{\vartriangle} \bot &= \bot \\
[a, b] \mathbin{\vartriangle} [c, d] &= \left[\begin{cases} c & a = -\infty \\ a & \text{otherwise} \end{cases}, \begin{cases} d & b = +\infty \\ b & \text{otherwise} \end{cases} \right]
\end{aligned}
$$

Interval widening example

Widening with $\perp$ yields identity:

$$\emptyset \mathbin{\triangledown} [1, 100] = [1, 100]$$

Increasing upper bounds expand to ∞:

$$[1, 100] \mathbin{\triangledown} [1, 101] = [1, \infty]$$

Decreasing lower bounds expand to $-\infty$:

$$[1, \infty] \mathbin{\triangledown} [0, 102] = [-\infty, \infty]$$

Convex Polyhedra

Convex Polyhedra (1/2)

We can use inequalities to describe the relationship between numerical variables of a program, e.g.:

$$\mathrm{y} \geq 1 \quad \wedge \quad \mathrm{x} + \mathrm{y} \geq 3 \quad \wedge \quad -\mathrm{x} + \mathrm{y} \leq 1$$

for two variables `x` and `y`.

The inequalities represent a *convex polyhedron*.

These form the abstract values of the *polyhedra* domain, which is a *relational* abstract domain.

```
y ^     /
  |    /
3 |   /
  |  /
2 |  \
  |   \
1 |    ---------------
  |
0 +---------------------->
  0 1 2 3 4 5 6 ...    x
```

Representation (implementation)

Convex polyhedra are represented using *double description* (with variables $X = \{\mathtt{x}_1, \ldots, \mathtt{x}_n\}$):

- a *system of inequalities* (A, B) where A is an $m \times n$ matrix, B is an m vector, and $\gamma(A, B) = \{X \mid AX \geq B\}$
- a *system of generators* (V, R) of vertices and rays where $V = \{V_1, \ldots, V_k\}$, $R = \{R_1, \ldots, R_l\}$, and
$$\gamma(V, R) = \{\Sigma_{i=1}^{k} \lambda_i V_i + \Sigma_{i=1}^{l} \mu_i R_i \mid \lambda_i \geq 0 \ \wedge \ \mu_i \geq 0 \ \wedge \ \Sigma_{i=1}^{k} \lambda_i = 1\}$$

An domain implementation will typically translate back and forth between the two, trying to minimize the number of conversions.

Representation example

For example, we can represent

$$\mathtt{y} \geq 1 \quad \wedge \quad \mathtt{x} + \mathtt{y} \geq 3 \quad \wedge \quad -\mathtt{x} + \mathtt{y} \leq 1$$

as a system of inequalities: $AX \geq B$

$$\begin{bmatrix} 0 & 1 \\ 1 & 1 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \geq \begin{bmatrix} 1 \\ 3 \\ -1 \end{bmatrix}$$

as a system of generators:

$$V = \{V_0 : (2,1), V_1 : (1,2)\}$$
$$R = \{R_0 : (1,0), R_1 : (1,1)\}$$

```
y ^     /
  |
  |   _
3 |   /| R1
  |V1/
2 |  \
  |   \  R0
1 |    x-> - - - - - -
  |   V0
0 +---------------------->
  0 1 2 3 4 5 6 ...     x
```

Convex Polyhedra (2/2)

Operations, some of which are easier on one representation, rather than the other:

- $\sqcup$ – returns a *convex hull*, which is an over-approximation of the union of two polyhedra.

  Easily expressed as a union of the corresponding generators.

- $\sqcap$ – returns the polyhedron representing the intersection of two polyhedra.

  Easily expressed as the conjunction of the two constraint systems.

But there is a catch

The polyhedra lattice is not complete: there exists strictly infinite chains for which the limit is not in the domain. Example: a disk.

Hence for some sets, e.g., a disk, there is no best abstraction.

As a consequence the abstraction to polyhedra is not a Galois connection.

A possible relaxation is to consider only concretization functions…

Concretization-based abstract interpretation

Proposition. Assume $\langle C; \sqsubseteq, \sqcup \rangle$ is a poset, $F : C \to C$ is a continuous function, $\bot_c \in C$ such that $\bot_c \sqsubseteq F(\bot_c)$, and $\bigsqcup_{n \in \mathbb{N}} F^n(\bot_c)$ exists.

Assume A is a set, $\gamma : A \to C$ is a function, $\leq$ is a preorder, defined as: $c \leq c' \iff \gamma(c) \sqsubseteq \gamma(c')$, $\bot_a \in A$ such that $\bot_c \sqsubseteq \gamma(\bot_a)$, $G : A \to A$ is a monotone function such that $F \circ \gamma \sqsubseteq \gamma \circ G$ and $\triangledown$ is a widening operator.

Then the upward iteration sequence with widening is ultimately stationary with limit a, such that $\mathrm{lfp}\, F \sqsubseteq \gamma(a)$ and $G(a) \leq a$.

As an alternative, Miné suggests a framework based on *partial Galois connections*, in which α is a partial function.

Want more abstractions?

There are many more numerical abstractions, see, e.g., Miné's thesis or this link:

`http://bugseng.com/products/ppl/abstractions`

The *Two Variables per Inequality* (TVPI) domain is a restricted form of polyhedra, only expressing relations between two variables: $a_{ij}\mathtt{x}_i + b_{ij}\mathtt{x}_j \leq c_{ij}$

Miné's *Octagon* domain is another restricted form of polyhedra, also expressing relations between two variables: $\pm\mathtt{x}_i \pm \mathtt{x}_j \leq c_{ij}$

Q: what do we get by restricting to one variable per inequality?

Numerical domains, botanically

ATTRIBUTE INDEPENDENT DOMAINS (NON-RELATIONAL):

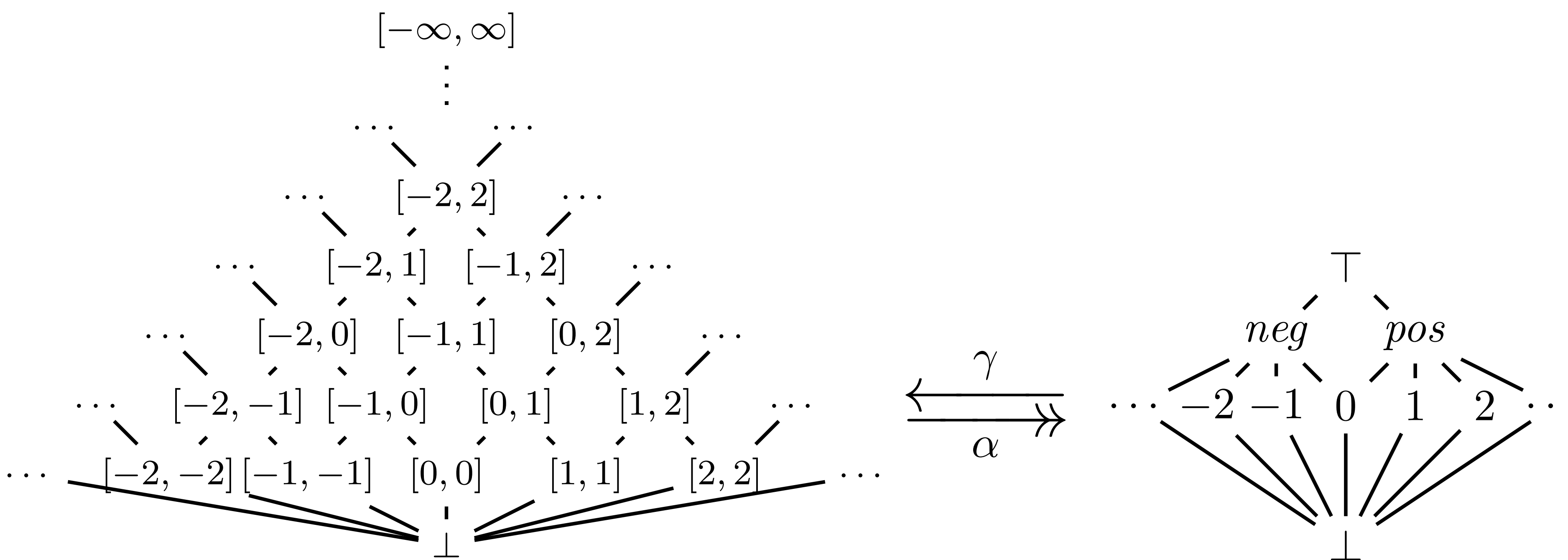
Parity, Sign, Constants, Simple Congruences, Intervals, . . .

RELATIONAL DOMAINS:

Polyhedra, Octagons, TVPI, . . .

A few connections between numerical abstractions

From intervals to a constant/sign combination

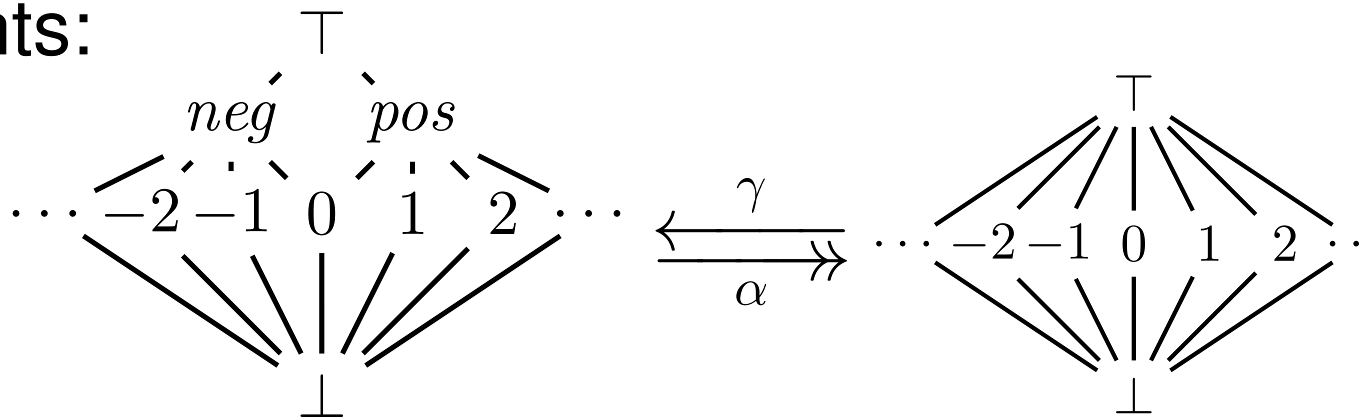


where

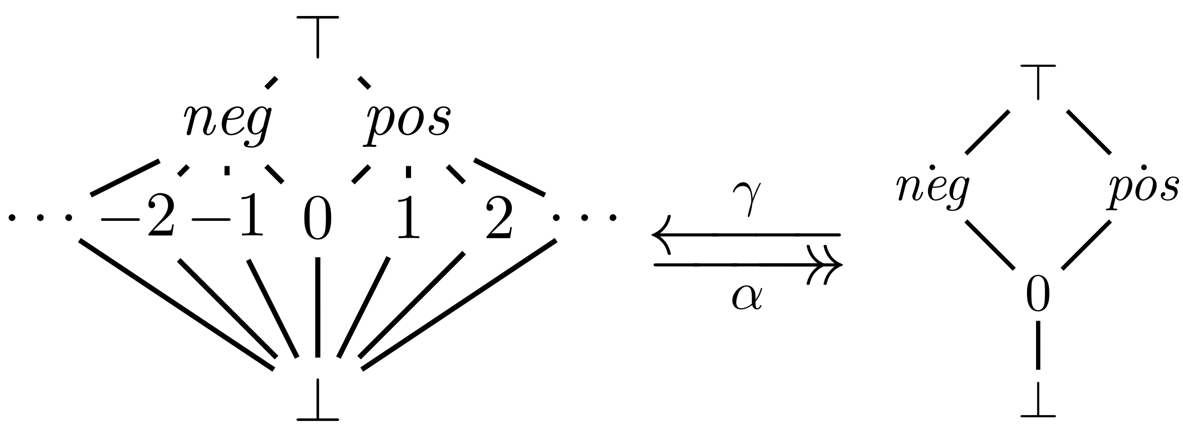
$$\alpha(\perp) = \perp$$
$$\alpha([a, a]) = a$$
$$\alpha([a, b]) = \begin{cases} pos & \text{if } a \geq 0 \\ neg & \text{if } b \leq 0 \\ \top & \text{otherwise} \end{cases}$$

From the constant/sign combination to . . .

This domain can (naturally) be abstracted to both constants:

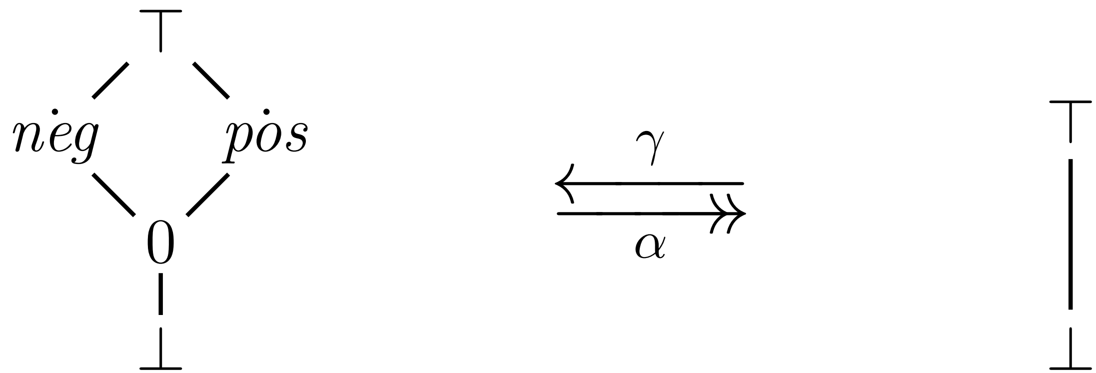


and signs:

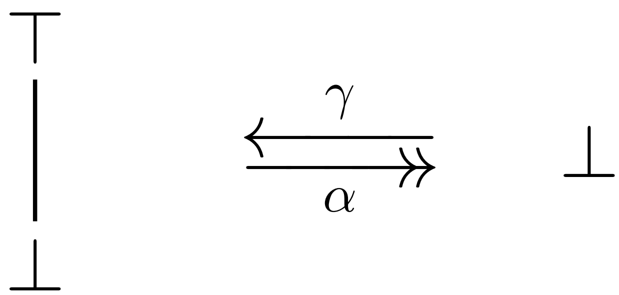


From the constant/sign combination to . . .

Both can be abstracted into a simple two-point domain:



and all the way down to a one-point lattice:



Connection summary

To summarize:

$$
\begin{tikzcd}
& & \mathit{Const} \arrow[dl, dash] \arrow[dr, dash] & & \\
\mathit{Interval} \arrow[r, dash] & \mathit{ConstSign} \arrow[dr, dash] & & \mathit{TopBot} \arrow[dl, dash] \arrow[r, dash] & \perp \\
& & \mathit{Sign} & &
\end{tikzcd}
$$

A nice lattice of lattices! $\ddot\smile$

The 3 counter machine analysis, revisited

The 3 counter machine analysis, revisited

We arrived at an abstract transition function $\mathtt{T}$, but the analysis is the least fixed point lfp of $\mathtt{T}$.

Q: Which fixed point theorem(s) of the three from last time are we using?

Design choices of the analysis

The resulting analysis associates an abstract memory to each program point:

$$\wp(PC \times \mathbb{N}_0 \times \mathbb{N}_0 \times \mathbb{N}_0) \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} PC \to (Parity \times Parity \times Parity)$$

Design choices of the analysis

The resulting analysis associates an abstract memory to each program point:

$$\wp(PC \times \mathbb{N}_0 \times \mathbb{N}_0 \times \mathbb{N}_0) \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} PC \to (Parity \times Parity \times Parity)$$

Alternatively we could have abstracted the components separately as follows:

$$\wp(PC \times \mathbb{N}_0 \times \mathbb{N}_0 \times \mathbb{N}_0) \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} \wp(PC) \times (Parity \times Parity \times Parity)$$

Design choices of the analysis

The resulting analysis associates an abstract memory to each program point:

$$\wp(PC \times \mathbb{N}_0 \times \mathbb{N}_0 \times \mathbb{N}_0) \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} PC \rightarrow (Parity \times Parity \times Parity)$$

Alternatively we could have abstracted the components separately as follows:

$$\wp(PC \times \mathbb{N}_0 \times \mathbb{N}_0 \times \mathbb{N}_0) \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} \wp(PC) \times (Parity \times Parity \times Parity)$$

Q: how would you characterize the first analysis using static analysis terminology?

Design choices of the analysis

The resulting analysis associates an abstract memory to each program point:

$$\wp(PC \times \mathbb{N}_0 \times \mathbb{N}_0 \times \mathbb{N}_0) \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} PC \to (Parity \times Parity \times Parity)$$

Alternatively we could have abstracted the components separately as follows:

$$\wp(PC \times \mathbb{N}_0 \times \mathbb{N}_0 \times \mathbb{N}_0) \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} \wp(PC) \times (Parity \times Parity \times Parity)$$

Q: how would you characterize the first analysis using static analysis terminology?

Q: how would you characterize the second?

Alternative 3 counter machine analyses?

Q: What changes if we want to switch to a different numerical abstraction (intervals, congruences, . . .)?

or rather,

Q: which assumptions about Parity did we rely on?

Alternative 3 counter machine analyses?

Q: What changes if we want to switch to a different numerical abstraction (intervals, congruences, . . .)?

or rather,

Q: which assumptions about Parity did we rely on?

$$\dfrac{\dfrac{\dfrac{}{\wp(\mathbb{N}_0 \times \mathbb{N}_0 \times \mathbb{N}_0) \leftrightarrows \wp(\mathbb{N}_0) \times \wp(\mathbb{N}_0) \times \wp(\mathbb{N}_0)} \qquad \dfrac{\dfrac{}{\wp(\mathbb{N}_0) \mathrel{\substack{\longleftarrow \\ \twoheadrightarrow}} Par} \quad \dfrac{}{\wp(\mathbb{N}_0) \mathrel{\substack{\longleftarrow \\ \twoheadrightarrow}} Par} \quad \dfrac{}{\wp(\mathbb{N}_0) \mathrel{\substack{\longleftarrow \\ \twoheadrightarrow}} Par}}{\wp(\mathbb{N}_0) \times \wp(\mathbb{N}_0) \times \wp(\mathbb{N}_0) \leftrightarrows Par \times Par \times Par}}{\wp(\mathbb{N}_0 \times \mathbb{N}_0 \times \mathbb{N}_0) \leftrightarrows Par \times Par \times Par}}{PC \to \wp(\mathbb{N}_0 \times \mathbb{N}_0 \times \mathbb{N}_0) \leftrightarrows PC \to Par \times Par \times Par}$$

From fixed points to constraints

Recall: a fixed point of T satisfies: $T(S^\#) = S^\#$

and a post-fixed point satisfies: $T(S^\#) \sqsubseteq S^\#$

Any post-fixed point of T is a sound approximation of $\operatorname{lfp} T$

In our case, T is defined as a big (pointwise) join:

$$T(S^\#) = X_1 \mathbin{\dot\sqcup} X_2 \mathbin{\dot\sqcup} \ldots \mathbin{\dot\sqcup} X_n \mathrel{\dot\sqsubseteq} S^\#$$

which is equivalent to:

$$X_1 \mathrel{\dot\sqsubseteq} S^\# \ \wedge\ X_2 \mathrel{\dot\sqsubseteq} S^\# \ \wedge\ \ldots\ \wedge\ X_n \mathrel{\dot\sqsubseteq} S^\#$$

Extracting 3 counter machine constraints (1/4)

```
T(S#) =  ( <bot,bot,bot>. [1 -> <top, even, even> ] )

  U.
            U.         ( <bot,bot,bot>. [pc+1 -> [x++]#(S#(pc))] )
      pc in Dom(S#)
      P_pc = inc x

            (...and for y and z)

  U.
            U.         ( <bot,bot,bot>. [pc+1 -> [x--]#(S#(pc))] )
      pc in Dom(S#)
      P_pc = dec x

            (...and for y and z)

  U.
            U.         ( <bot,bot,bot>. [pc' -> [x==0]#(S#(pc))] )
      pc in Dom(S#)         U. ( <bot,bot,bot>. [pc'' -> [x<>0]#(S#(pc))] )
  P_pc = zero x pc' else pc''

            (...and for y and z)

     C. S#
```

Extracting 3 counter machine constraints (2/4)

```
            ( <bot,bot,bot>. [1 -> <top, even, even> ] )  C.  S#

/\
               U.        ( <bot,bot,bot>. [pc+1 -> [x++]#(S#(pc))] ) C.  S#
       pc in Dom(S#)
       P_pc = inc x

               (...and for y and z)

/\
               U.        ( <bot,bot,bot>. [pc+1 -> [x--]#(S#(pc))] ) C.  S#
       pc in Dom(S#)
       P_pc = dec x

               (...and for y and z)

/\
               U.        ( <bot,bot,bot>. [pc' -> [x==0]#(S#(pc))] )      C.  S#
       pc in Dom(S#)         U. ( <bot,bot,bot>. [pc'' -> [x<>0]#(S#(pc))] )
  P_pc = zero x pc' else pc''

               (...and for y and z)
```

Extracting 3 counter machine constraints (3/4)

```
           <top, even, even>  C  S#(1)

/\
       for all pc in Dom(S#), such that P_pc = inc x :

            [x++]#(S#(pc))  C  S#(pc+1)

            (...and for y and z)

/\
       for all pc in Dom(S#), such that P_pc = dec x :

            [x--]#(S#(pc))  C  S#(pc+1)

            (...and for y and z)

/\
       for all pc in Dom(S#), such that P_pc = zero x pc' else pc'' :

            [x==0]#(S#(pc))  C  S#(pc')

              /\  [x<>0]#(S#(pc))  C  S#(pc'')

            (...and for y and z)
```

...not that far from the constraint-based analyses of 'dOvs' and 'Static Analysis'.

Extracting 3 counter machine constraints (4/4)

```
            <top, even, even>  C  S#(1)

/\
        for all "inc x" instructions in P with program counter pc :

             [x++]#(S#(pc))  C  S#(pc+1)

             (...and for y and z)

/\
        for all "dec x" instructions in P with program counter pc :

             [x--]#(S#(pc))  C  S#(pc+1)

             (...and for y and z)

/\
        for all "zero x pc' else pc''" instructions in P with program counter pc

             [x==0]#(S#(pc))  C  S#(pc')

               /\  [x<>0]#(S#(pc))  C  S#(pc'')

             (...and for y and z)
```

. . . not that far from the constraint-based analyses of 'dOvs' and 'Static Analysis'.

Summary

Summary

A catalogue of abstractions

- □ Toolbox abstractions
- □ Structural abstractions: sums, pairs/tuples, ...
- □ Numerical abstractions: constants, intervals, congruences, polyhedra, ...
- □ Concretization-based abstract interpretation, briefly

A retrospective on the 3 counter machine analysis, incl. constraint extraction