Set Constraints

Leszek Pacholski

CP'97

University of Wrocław

Set-based analysis

=

(Set) constraints generation

+

Constraint resolution

A problem:

What are set constraints?

Set Constraints

Set of constructors $C = \{b, c, \ldots\}$

Set of variables $V = \{\alpha, \beta, \ldots\}$

The set expressions with projections:

$$E \quad ::= \quad 0 \mid \alpha \mid E_1 \cup E_2 \mid E_1 \cap E_2 \mid \neg E_1 \mid c(E_1, \ldots, E_n) \mid c_i^{-1}(E_1)$$

A system S of set constraints:

$$\bigwedge_{i \in I} E_i \subseteq E_i' \wedge \bigwedge_{j \in J} E_j \not\subseteq E_j'$$

Question: *What are the solutions of S?*

Semantics of Set Expressions

Each set expression denotes a subset of H (the Herbrand universe of terms.)

An assignment σ is a function $V \to 2^H$.

Extend σ to expressions:

$$
\begin{aligned}
\sigma(0) &= \emptyset \\
\sigma(E_1 \cup E_2) &= \sigma(E_1) \cup \sigma(E_2) \\
\sigma(E_1 \cap E_2) &= \sigma(E_1) \cap \sigma(E_2) \\
\sigma(\neg E) &= H - \sigma(E) \\
\sigma(c(E_1, \ldots, E_n)) &= \{c(t_1, \ldots, t_n) \mid t_i \in \sigma(E_i)\} \\
\sigma(c_i^{-1}(E)) &= \{t_i \mid c(t_1, \ldots, t_n) \in \sigma(E)\}
\end{aligned}
$$

σ is a solution of constraints

$$\bigwedge_{i \in I} E_i \subseteq E_i' \wedge \bigwedge_{j \in J} E_j \not\subseteq E_j' \quad \text{iff}$$

$$(\forall i \in J)\, \sigma(E_i) \subseteq \sigma(E_i') \;\wedge\; (\forall j \in J)\, \sigma(E_j) \not\subseteq \sigma(E_j')$$

Set-based program analysis

```
X := cons(a,cons(b,cons(c,cons(d,nil))))
Y := nil
A
while (car(X) ≠ c) do
    B
    Y := cons(car(X),Y)
    X := cdr(X)
    C
D
```

$\mathtt{X}^A \supseteq \mathtt{cons(a,cons(b,cons(c,cons(d,nil))))}$

$\mathtt{Y}^A \supseteq \mathtt{nil}$

$\mathtt{X}^B \supseteq \mathtt{X}^A - \mathtt{cons(c,1)}$ $\quad$ $\mathtt{Y}^B \supseteq \mathtt{Y}^A$

$\mathtt{X}^B \supseteq \mathtt{X}^C - \mathtt{cons(c,1)}$ $\quad$ $\mathtt{Y}^B \supseteq \mathtt{Y}^C$

$\mathtt{X}^C \supseteq \mathtt{cdr(X}^B\mathtt{)}$ $\quad$ $\mathtt{Y}^C \supseteq \mathtt{cons(car(X}^B\mathtt{),Y}^B\mathtt{)}$

$\mathtt{X}^D \supseteq \mathtt{X}^C \cap \mathtt{cons(c,1)}$ $\quad$ $\mathtt{Y}^D \supseteq \mathtt{Y}^C$

Set-based analysis, contd.

Solution:

$\mathtt{X}^A = \{\mathtt{cons(a,cons(b,cons(c,cons(d,nil))))}\}$

$\mathtt{Y}^A = \{\mathtt{nil}\}$

$\mathtt{X}^B = \{\mathtt{cons(b,cons(c,cons(d,nil)))}\} \cup \mathtt{X}^A$

$\mathtt{Y}^B =$ lists constructed from $\mathtt{a}$ and $\mathtt{b}$

$\mathtt{X}^C = \{\mathtt{cons(b,cons(c,cons(d,nil)))},$
$\quad \mathtt{cons(c,cons(d,nil))}\}$

$\mathtt{Y}^C =$ non-empty lists constructed from $\mathtt{a}$ and $\mathtt{b}$

$\mathtt{X}^D = \{\mathtt{cons(c,cons(d,nil))}\}$

$\mathtt{Y}^D = \mathtt{Y}^B$

Building and applying

J.C. Reynolds, N.D. Jones, S.S. Muchnick

N. Heintze, J. Jaffar, A. Aiken,
T.K. Lakshman, B. Murphy, E. Wimmers,
M. Fähndrich, A. Deutsch, J. Palsberg,
F. Henglein, J. S. Foster,
F.D. Damm, P. Mishra, U. Reddy, E. Wang,
P.N. Hilfinger, D. McAllester, . . .

Set constraints

- Herbrand
- Herbrand with (various) syntactic restrictions
- Herbrand / Equational theory
- Tarskian
- Higher Order

Tarskian

$f(C_1,\ldots,C_n)$ — $n+1$-ary relation

$f(C_1,\ldots,C_n)$ denotes

$$\{y : \exists x_1 \in C_1,\ldots,x_n \in C_n, \langle x_1,\ldots,x_n,y\rangle \in f\}$$

$f(C_1,\ldots,C_n)$ may be empty

f is deterministic if

$$\text{card}(f(a_1,\ldots,a_n)) = 1$$

Tarskian, contd.

- without deterministic functions — EXPTIME-complete
- without deterministic constants (but with deterministic functions) — NEXPTIME-complete
- full — NEXPTIME-complete (?)

with recursion (least fixpoint)

- without deterministic functions — EXPTIME-complete
- full — undecidable

Tarskian, contd.

Related (apply) to:

- modal logics
- propositional dynamic logic
- modal μ-calculus
- knowledge representation
- natural languages
- concept languages

Jónnson, Tarski, McAllester, Givan, Witty, Kozen

Higher-order set constraints

"We use the terms set expressions and type expressions interchangeably"
[Fähndrich, Aiken]

$$E ::= 0 \mid \alpha \mid f(E_1,\ldots,E_n) \mid E_1 \cap E_2 \mid E_1 \to E_2$$

$$\boxed{\sigma : \text{Var} \to ?}$$

Additional operation $\to$ but no $\neg$.

$$\sigma(E_1 \to E_2) \quad = \quad \{f \mid f \text{ is a function from } \sigma(E_1) \text{ into } \sigma(E_2)\}$$

Problem: requires working in a domain theory; no theoretical results.
Type inference (soft typing): Aiken, Wimmers, Lakshman, Damm, Heintze, McAllester,...

Semantic domains

$$\mathbb{D} \;=\; \{\perp\} \cup (\mathbb{D} \to \mathbb{D}) \cup \bigcup_{c \in C} c(\mathbb{D} - \{\perp\}, \ldots, \mathbb{D} - \{\perp\}) \cup \{\mathtt{ERROR}\}$$

$\mathbb{D}$ is the union of sets of finite elements $\mathbb{D}_0 \subseteq \mathbb{D}_1 \subseteq \ldots$, where

$$\mathbb{D}_0 \;=\; \{\perp\}$$

$$\mathbb{D}_{i+1} \;=\; \mathbb{D}_i \cup (\mathbb{D}_i \to \mathbb{D}_i) \cup \bigcup_{c \in C} c(\mathbb{D}_i - \{\perp\}, \ldots, \mathbb{D}_i - \{\perp\}) \cup \{\mathtt{ERROR}\}$$

Note: $\mathbb{D}_i \to \mathbb{D}_i$ denotes here the set of finite, strict monotonic functions from $\mathbb{D}$ to $\mathbb{D}$.

SBA analysis

$$e \quad ::= \quad x \mid (e_1 e_2) \mid \lambda x.e \mid c(e_1,\ldots,e_n) \mid \mathtt{case}(e_0\,\mathtt{of}\,p_1 : c_1, p_2 : c_2, \ldots, p_n : c_n)$$

$$((\lambda x.e)v) \;\to\; e[v/x] \qquad (\beta)$$

$$(c(v_1,\ldots,v_n)v) \;\to\; \mathtt{ERROR} \qquad (\beta\text{-error})$$

$$\mathtt{case}(u\,\mathtt{of}\,p_1 : c_1, p_2 : c_2, \ldots, p_n : c_n) \;\to\; \sigma(e_i)$$

where $u = \sigma(p_i)$ and $\sigma(x) = x$ for $x \notin p_i$

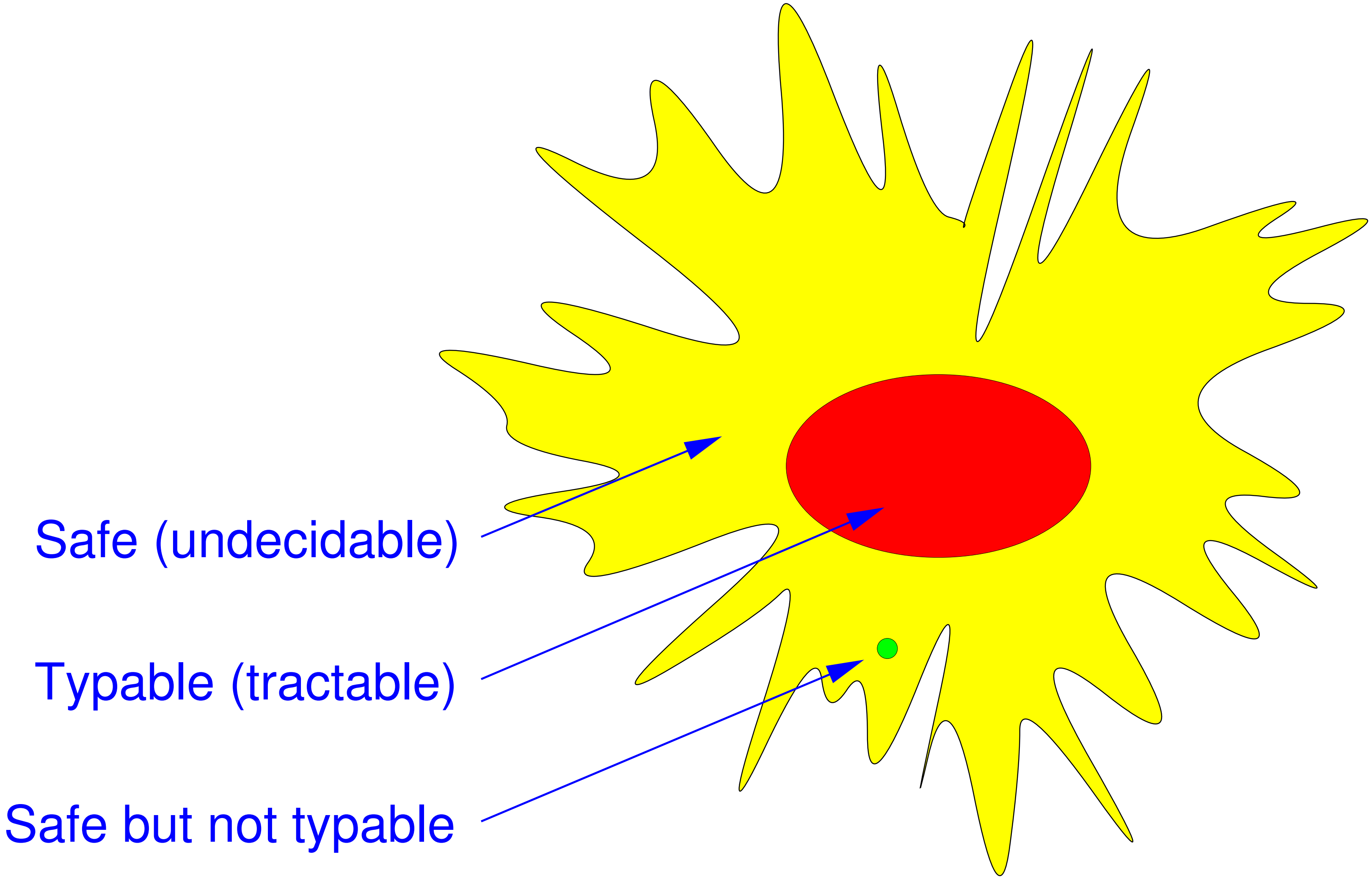
$$\mathtt{case}(u\,\mathtt{of}\,p_1 : c_1, p_2 : c_2, \ldots, p_n : c_n) \;\to\; \mathtt{ERROR}$$

if p_j clashes with u for $1 \leq j \leq n$

Semantic function: $\mu(e) \;=\;$ normal form of e

e is safe iff $e \not\to \mathtt{ERROR}$

Safe vs. typable



type system

$$\frac{}{A \cup \{x : a\}, S \vdash x : a}$$

$$\frac{A \cup \{x : \tau_1\}, S \vdash e : \tau_2}{A, S \vdash \lambda x.e : \tau_1 \to \tau_2}$$

$$\frac{A, S \vdash e_i : \tau_i \quad 1 \leq i \leq n}{A, S \cup \{c(\tau_1, \ldots, \tau_n) \qquad \alpha\} \vdash c(e_1, \ldots, e_n) : \alpha}$$

$$\frac{A, S \vdash e_1 : \tau_1, e_2 : \tau_2}{A, S \cup \{\tau_1 \qquad \alpha \to \beta, \tau_2 \qquad \alpha\} \vdash e_1 e_2 : \beta}$$

Hindley-Milner

$=$

$=$ $=$

Aiken-Wimmers

$\subseteq$

$\subseteq$ $\subseteq$

Polymorphic types

$$\frac{A, S \vdash e : \tau \ \text{ and } \ \mathsf{Sol}(S) \neq \emptyset, \ \ \alpha \text{ not fee in } A}{A, \emptyset \vdash e : \forall \alpha.\tau \,\texttt{where}\, S}$$

$$\frac{A, S \vdash e : \forall \alpha.\tau \,\texttt{where}\, S'}{A, S \cup S'[\tau/\alpha] \vdash e : \tau[\tau'/\alpha]}$$

$$\frac{A, S \vdash e : \tau \quad A \cup \{x : \tau\}, S \vdash e' : \tau'}{A, S \vdash \texttt{let}\, x = e \,\texttt{in}\, e' : \tau'}$$

Typing á la Aiken-Wimmers

Type expressions:

$$\tau \quad ::= \quad \alpha \mid 0 \mid \tau_1 \rightarrow \tau_2 \mid c(t_1,\ldots,t_n) \mid a \mid \tau_1 \cup \tau_2 \mid \tau_1 \& \tau_2$$

e has type τ ($e : \tau$) iff

$$\{v \in \mathbb{D} \mid v \leq \mu(e)\} \subseteq \tau$$

Thm. Every pure λ-term is A-W typable.

Proof. Set type of each variable to α_0, where $\alpha_0 = \alpha_0 \rightarrow \alpha_0$.

Hindley-Milner vs. Aiken-Wimmers

$$\begin{aligned}
\mathsf{I} &= \lambda x.x \\
\mathsf{Double} &= \lambda f.\lambda x.f(fx) \\
\mathsf{Y} &= \lambda u.(\lambda x.u(xx))(\lambda x.u(xx)) \\
\mathsf{Z} &= \mathsf{Y}(\mathsf{Double}\ \mathsf{I})
\end{aligned}$$

	Hindley-Milner	Aiken-Wimmers
I	$\forall\alpha.\alpha \to \alpha$	$\forall\alpha.\alpha \to \alpha$
Double	$\forall\alpha.(\alpha \to \alpha) \to \alpha \to \alpha$	$\forall\alpha\beta\gamma.((\beta \to \alpha)\&(\alpha \to \gamma)) \to \beta \to \gamma$
Y	not typable	$\forall\alpha\beta.((\alpha \to \beta)\&(\alpha \to \alpha)) \to \beta$
Z	not typable	0

Solving S.C. — general tools

- Rewriting of constraints

  Inductive closures
  — adding consequences

- Working on abstract data structures

  tree set automata
  or
  hypergraphs
  or
  monadic class

  +

  combinatorial arguments

Proper systems of SC

A type X is upward closed if

$$X = \mathrm{Up}(X), \qquad \text{where } \mathrm{Up}(X) = \{\perp\} \cup \bigcup_{x \in X - \{\perp\}} \{y \mid y \geq x\}$$

Given X, $\overline{X}$ denotes the smallest type expression without variables which is upward closed and such that $\sigma(X) \subseteq \overline{X}$ for each substitution σ.

$$L ::= 0 \mid \alpha \mid c(L_1, \ldots, L_n) \mid R \to L \mid L_1 \cap \overline{L_2} \mid L_1 \cup L_2$$
$$R ::= 0 \mid \alpha \mid c(R_1, \ldots, R_n) \mid L \to R \mid R_1 \cap R_2 \mid R_1 \cup R_2 \text{ where } \overline{R_1 \cap R_2} = 0$$

Proper constraints: $L_i \subseteq R_i$

Inductive systems

TLVar(E) denotes the set of top level variables of E.

A system S of constraints is inductive if

1. $S \;=\; \{L_i \subseteq \alpha_i \subseteq U_i \mid i = 1, \ldots, n\}$
2. $\mathrm{TLVar}(L_i) \cup \mathrm{TLVar}(U_i) \;\subseteq\; \{\alpha_1, \ldots, \alpha_{i-1}\}$ for $1 \leq i \leq n$
3. For all $i_0 = 1, \ldots, n$ and all integers j

$$\left.\begin{array}{r} (\forall i = 1, \ldots, i_0 - 1)\ (L_i \cap \mathbb{D}_j \subseteq \alpha_i \cap \mathbb{D}_j \subseteq U_i \cap \mathbb{D}_j) \\ \text{and} \\ (\forall i = i_0, \ldots, n)\ (L_i \cap \mathbb{D}_{j-1} \subseteq \alpha_i \cap \mathbb{D}_{j-1} \subseteq U_i \cap \mathbb{D}_{j-1}) \end{array}\right\} \text{ implies } L_{i_0} \cap \mathbb{D}_j \subseteq U_{i_0} \cap \mathbb{D}_j$$

Theorem. Every proper system $\{L_i \subseteq R_i : i = 1, \ldots, n\}$ is equivalent to a finite set of inductive systems.

Cascading equations

A set $\mathrm{Eq} = \{\alpha_1 = E_1, \ldots, \alpha_n = E_n\}$ of equations is cascading if $\mathsf{TLVar}(E_i) \cap \{\alpha_i, \ldots, \alpha_n\} = \emptyset$.

Let $S = \{U_i \subseteq \alpha_i \subseteq U_i \mid i = 1, \ldots, n\}$ be an inductive system of constraints. Then S is equivalent to a system $\{\alpha_i = U_i' \cup (\beta_i \cap U_i) \mid i = 1, \ldots, n\}$ of cascading equations.

If $\mathrm{Eq} = \{\alpha_1 = E_1, \ldots, \alpha_n = E_n\}$ is a set of cascading equations and σ is a substitution for variables other than $\{\alpha_1, \ldots, \alpha_n\}$, then there is a unique extension σ' of σ which is a solution of Eq.

Solving set constraints

The algorithm

1. Simplify $\overline{X}$ and $\neg\overline{X}$, *e.g.*, replace $\overline{X \cup Y}$ by $(\overline{X} \cap \neg\overline{Y}) \cup (\overline{X} \cap \overline{Y}) \cup (\neg\overline{X} \cap \overline{Y})$.
2. Normalize terms (bring to a disjunctive normal form).
3. Simplify constraints,
   e.g., replace $S \cup \{\alpha \cap \overline{L} \subseteq \alpha\}$ by S,
   and $S \cup \{L \subseteq R_1 \cup R_2\}$ by $S \cup \{L \cap \neg\overline{R}_1 \subseteq R_2, L \cap \neg\overline{R}_2 \subseteq R_1\}$
4. Saturate, *e.g.*, add $L \subseteq R$ if $L \subseteq \alpha$, $\alpha \subseteq R$ are in S.
5. Either solve an inductive system or report “no solution”.

Set-based analysis, contd.

Note: no interdependence of variables

Environments $\rightsquigarrow$ Set environments

$$\left\{\begin{array}{l}[\mathtt{X} \mapsto 1, \mathtt{Y} \mapsto 2] \\ [\mathtt{X} \mapsto 3, \mathtt{Y} \mapsto 4]\end{array}\right. \quad \rightsquigarrow \quad \left\{\begin{array}{l}\mathtt{X} \mapsto \{1,3\} \\ \mathtt{Y} \mapsto \{2,4\}\end{array}\right.$$

$$(\mathtt{X},\mathtt{Y}) \mapsto \{(1,2),(3,4)\} \qquad \boxed{\begin{array}{ll}(1,2) & (3,4) \\ (1,4) & (3,2)\end{array}}$$

$$(\mathtt{X},\mathtt{X}) \mapsto \{(1,1),(3,3)\} \qquad \boxed{\begin{array}{ll}(1,1) & (3,3) \\ (1,3) & (3,1)\end{array}}$$

Environment evaluation

A suspended computation is a pair $\langle u, \sigma \rangle$, where u is a term and σ is a substitution.

$$\frac{\begin{array}{l} \text{EVAL}((f,w),\rho) \\ \langle f,\rho\rangle \to \langle \lambda x.u, \sigma\rangle \\ \langle w,\rho\rangle \to v \end{array}}{\langle (f,w),\rho\rangle \to \langle u, \sigma[x := v]\rangle} \quad \text{BETA}$$

$$\frac{\text{EVAL}((f,w),\rho)}{\text{EVAL}(f,\rho), \text{EVAL}(w,\rho)} \quad \text{BETA-CALL}$$

$$\frac{\begin{array}{l} \text{EVAL}((f,w),\rho) \\ \langle f,\rho\rangle \to c(v_1,\ldots,v_n) \end{array}}{\texttt{ERROR}} \quad \text{BETA-ERR}$$

$$\frac{\text{EVAL}(x,\rho)}{\langle x,\rho\rangle \to \rho(x)} \quad \text{VAR} \qquad \frac{\langle u,\rho\rangle \to \langle w,\sigma\rangle}{\text{EVAL}(w,\sigma)} \quad \text{CALL}$$

$$\frac{\langle u,\rho\rangle \to \langle w,\sigma\rangle \quad \langle w,\sigma\rangle \to v}{\langle u,\rho\rangle \to v} \quad \text{RETURN}$$

The SBA rule

$$\frac{\begin{array}{l}\mathsf{EVAL}(u,\rho)\\ \langle u,\rho\rangle \to v\\ \mathsf{EVAL}(u,\gamma)\end{array}}{\langle u,\gamma\rangle \to v}\quad \text{SBA}$$

Flow rules

(Environment omitted)

$$\frac{\mathsf{EVAL}((f,w)) \quad f \to \lambda x.u}{x \to w \quad (f,w) \to u}\ \text{BETA}$$

$$\frac{\mathsf{EVAL}((f,w)) \quad f \to c(v_1,\ldots,v_n)}{\texttt{ERROR}}\ \text{BETA-ERR}$$

SBA safety, contd.

Thm. SBA safety for terms with deep, parallel and strict semantics of case is co-NP hard.

Thm. SBA safety for terms with deep, serial semantics of case is EXPTIME-hard.

Thm. SBA safety for terms with non-overlaping case statements is in $O(n^3)$.

Implementations

- Heintze: set-based analysis for ML
- Aiken: Illyria
- Foster: CLP(SC)
- Yi: detection of uncaught exceptions
- Fähndrich, Aiken: BANE (The Berkeley ANalysis Engine)

Empirical results

- Dynamic Typing (Henglein)
  - near linear algorithm
  - 1000 lines/sec
- Set-Based Analysis for ML
  - $O(n^3)$ complexity
  - 500 lines/sec
- OO inference for SELF
  (Agesen, Palsberg, Schwartzbach)
  - $O(n^3)$ complexity
  - very efficient in practice
- Set-Based Analysis for LP
  - EXPTIME complexity
  - 10-50 lines/sec
- Analysis of FL
  - EXPTIME complexity
  - 10-20 lines/sec