APPLICATION TO GENERIC PREDICATE ABSTRACTION

P. Cousot

Patrick.Cousot@ens.fr http://www.di.ens.fr/~cousot

Biarritz IFIP-WG 2.3 meeting (3)

23 — 28 mars 2003, Hotel Miramar, Biarritz, France

3.1 Generic Predicate Abstraction

Generic Abstract Domains

- A generic abstract domain is parameterized.
- A particular abstract domain instantiation: bind the formal parameters to program dependent actual parameters (constants, variables, control points, etc.)
- Example: Kildall [9]'s generic abstract domain for constant propagation $D(C, V)$ is:

$$D(C, V) = \prod_{\ell \in C} \prod_{\mathtt{X} \in V(\ell)} L \ .$$

- L is Kildall's complete lattice. Given a command C, it is instantiated to $D(\mathsf{lab}[\![C]\!], \mathsf{var}[\![C]\!])$ where
  - $\mathsf{lab}[\![C]\!]$ is the set of labels of command C
  - $\mathsf{var}[\![C]\!](\ell)$ is the set of program variables $\mathtt{X}$ which are visible at this program point ℓ of command C.

Generic Comparison Abstract Domain

We let $\mathcal{D}_{\mathrm{rel}}(X)$ be a generic relational integer abstract domain parameterized by a set X of program and auxiliary variables (such as octagons [12, 13] or polyhedra [7]). This abstract domain is assumed to have abstract operations on $r, r_1, r_2 \in \mathcal{D}_{\mathrm{rel}}(X)$ such as:

- the projection or variable elimination $\exists x \in X : r$,
- disjunction $r_1 \vee r_2$,
- conjunction $r_1 \wedge r_2$,
- abstract predicate transformers for assignments and tests, etc.

Generic Comparison Abstract Domain

Then we define the generic comparison abstract domain:

$$\mathcal{D}_{\mathrm{lt}}(X) = \{\langle \mathrm{lt}(\mathtt{t}, a, b, c, d),\ r\rangle \mid \mathtt{t} \in X \wedge a, b, c, d \notin X \wedge \\ r \in \mathcal{D}_{\mathrm{rel}}(X \cup \{a, b, c, d\})\}\ .$$

Concretization of the Generic Comparison Abstract Domain

The meaning $\gamma(\langle \mathrm{lt}(\mathtt{t}, a, b, c, d),\ r\rangle)$ of an abstract predicate
$\langle \mathrm{lt}(\mathtt{t}, a, b, c, d),\ r\rangle$
is informally that all elements of $\mathtt{t}$ between indices a and b are less than any element of $\mathtt{t}$ between indices c and d and moreover r holds:

$$\gamma(\langle \mathrm{lt}(\mathtt{t}, a, b, c, d),\ r\rangle) = \exists a, b, c, d : \mathtt{t}.\ell \le a \le b \le \mathtt{t}.h \\ \wedge\, \mathtt{t}.\ell \le c \le d \le \mathtt{t}.h \\ \wedge\, \forall i \in [a, b] : \forall j \in [c, d] : \mathtt{t}[i] \le \mathtt{t}[j] \wedge r$$

where $\mathtt{t}.\ell$ is the lower bound and $\mathtt{t}.h$ is the upper bound of the indices i of the array $\mathtt{t}$ with elements $\mathtt{t}[i]$.

Concretization of the Generic Comparison Abstract Domain (cont'd)

More formally, there should be a declaration $\mathtt{t} : \mathtt{array}[\ell, h]\ \mathtt{of\ int}$ so that $\gamma(\langle \mathrm{lt}(\mathtt{t}, a, b, c, d),\ r\rangle)$ defines a set of environments ρ mapping program and auxiliary variables $\mathtt{X}$ to their value $\rho(\mathtt{X})$ for which the above concrete predicate holds:

$$\gamma(\langle \mathrm{lt}(t, a, b, c, d),\ r\rangle) = \{\rho \mid \exists a, b, c, d : \rho(\mathtt{t}).\ell \le a \le b \le \rho(\mathtt{t}).h \\ \wedge\, \rho(\mathtt{t}).\ell \le c \le d \le \rho(\mathtt{t}).h \\ \wedge\, \forall i \in [a, b] : \forall j \in [c, d] : \rho(\mathtt{t})[i] \le \rho(\mathtt{t})[j] \\ \wedge\, \rho \in \gamma(r)\}$$

where the domain of the ρ is $X \cup \{a, b, c, d\}$ and $\gamma(r)$ is the concretization of the abstract predicate $r \in \mathcal{D}_{\mathrm{rel}}(X \cup \{a, b, c, d\})$ specifying the possible values of the variables in X and the auxiliary variables a, b, c, d.

Abstract Logical Operations of the Generic Comparison Abstract Domain

Then the abstract domain must be equipped with abstract operations such as

- implication $\Rightarrow$,
- conjunction $\wedge$,
- disjunction $\vee$, etc.

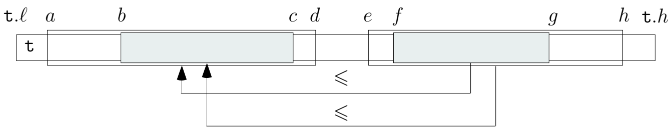
We simply provided a few examples.

Abstract Implication

We have $\langle \mathrm{lt}(\mathtt{t}, a, b, c, d),\ r\rangle \Rightarrow r$. If $r \Rightarrow r'$ and $a \leq b \leq c \leq d$ and $e \leq f \leq g \leq h$ then:

$$\langle \mathrm{lt}(\mathtt{t}, a, d, e, h),\ r\rangle \ \Rightarrow\ \langle \mathrm{lt}(\mathtt{t}, b, c, f, g),\ r'\rangle \qquad (1)$$

as shown below:



Abstract Conjunction

If $\mathtt{t}, i, j, k, \ell \notin \mathrm{var}[\![r]\!]$, then:

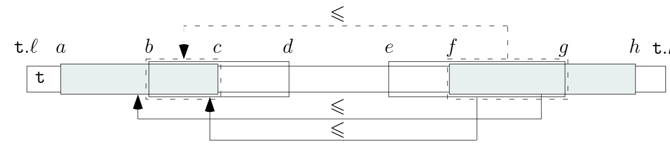
$$r \wedge \langle \mathrm{lt}(\mathtt{t}, a, c, f, h),\ r'\rangle \ =\ \langle \mathrm{lt}(\mathtt{t}, a, c, f, h),\ r \wedge r'\rangle \qquad (2)$$

Abstract Conjunction (Cont'd)

If $a \leq b \leq c \leq d$ and $e \leq f \leq g \leq h$ then we have:

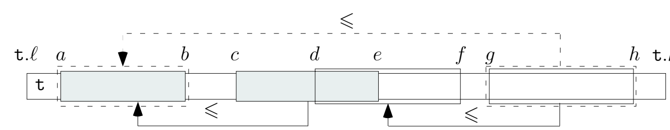
$$\begin{aligned}&\langle \mathrm{lt}(\mathtt{t}, a, c, f, h),\ r\rangle \wedge \langle \mathrm{lt}(\mathtt{t}, b, d, e, g),\ r'\rangle\\ &\quad = \langle \mathrm{lt}(\mathtt{t}, b, c, f, g),\ \exists a, d, e, h : r \wedge r'\rangle\end{aligned}$$

as shown below:



Abstract Conjunction (End)

The same way:



we have:

$$\begin{aligned}&\langle \mathrm{lt}(\mathtt{t}, a, b, c, e),\ r\rangle \wedge \langle \mathrm{lt}(\mathtt{t}, d, f, g, h),\ r'\rangle\\ &\quad = \langle \mathrm{lt}(\mathtt{t}, a, b, g, h),\ \exists c, e, d, f : r \wedge r'\rangle \qquad (3)\end{aligned}$$

when $(r \wedge r') \Rightarrow (c \leq d \leq e \leq f)$.

Abstract Disjunction

We have:

$$\langle \mathrm{lt}(\mathtt{t}, a, b, c, d),\ r\rangle \vee \langle \mathrm{lt}(\mathtt{t}, e, f, g, h),\ r'\rangle \quad = \tag{4}$$
$$\langle \mathrm{lt}(\mathtt{t}, i, j, k, \ell),\ ()\exists a, b, c, d : i = a\ \wedge j = b \wedge k = c \wedge \ell = d \wedge r)$$
$$\vee (\exists e, f, g, h : i = e \wedge j = f \wedge k = g \wedge \ell = h \wedge r')$$

Abstract Disjunction (cont'd)

In case one of the terms does not refer to the array ($\mathtt{t} \notin \mathrm{var}[\![r]\!]$), a criterion must be used to force the introduction of an identically true array term $\mathrm{lt}(\mathtt{t}, i, i, i, i)$. For example if the auxiliary variables d, f, g, h in r' depend upon one selectively chosen variable $\mathtt{I}$, then we have:

$$r \vee \langle \mathrm{lt}(\mathtt{t}, d, f, g, h),\ r'\rangle = \tag{5}$$
$$\langle \mathrm{lt}(\mathtt{t}, i, j, k, \ell),\ (i = j = k = \ell = \mathtt{I} \wedge r) \vee \tag{6}$$
$$(\exists d, f, g, h : i = d \wedge j = f \wedge k = g \wedge \ell = h \wedge r')\rangle$$

This case appears typically in loops, which can also be handled by unrolling, see 3.1.

Abstract Predicate Transformers for the Generic Comparison Abstract Domain

- Then the abstract domain must be equipped with abstract predicate transformers for tests, assignments, etc.
- We consider forward strongest postconditions (although weakest preconditions, which avoid an existential quantifier in assignments, may sometimes be simpler [14]).
- We depart from traditional predicate abstraction which uses a simplifier (or a theorem prover) to formally evaluate the abstract predicate transformer $\alpha \circ F \circ \gamma$ approximating the concrete predicate transformer F.

- The alternative proposed below is traditional in static program analysis and directly provides an over-approximation of the best abstract predicate transformer $\alpha \circ\ F \circ \gamma$ in the form of an algorithm (which correctness must be established formally).
- The simplifier/prover/pattern-matcher is used only to reduce the post-condition in the normal form (??) which is required for the abstract predicates.

Abstract Strongest Postconditions for Tests

$\{ P_1 \}$
if $(\mathtt{t[I]} > \mathtt{t[I+1]})$ **then**
  $\{ P_1 \wedge \langle \mathrm{lt}(\mathtt{t}, i, j, k, \ell),\ i = \mathtt{I} \wedge j = \mathtt{I} + 1 \wedge k = \mathtt{I} \wedge \ell = \mathtt{I} \rangle \}$ (7)
  ...
  $\{ P_2 \}$
else
  $\{ P_1 \wedge \langle \mathrm{lt}(\mathtt{t}, i, j, k, \ell),\ i = \mathtt{I} \wedge j = k = \ell = \mathtt{I} + 1 \rangle \}$ (8)
  ...
  $\{ P_3 \}$
fi
$\{ P_2 \vee P_3 \}$ (9)

Abstract Strongest Postconditions for Assignments

For assignment, assuming $\mathtt{t} \notin \mathrm{var}[\![r]\!]$ and $r \Rightarrow (i = \mathtt{I} \wedge j = \mathtt{I} + 1 \wedge k = \mathtt{I} \wedge \ell = \mathtt{I})$, we have:

$\{ \langle \mathrm{lt}(\mathtt{t}, i, j, k, \ell),\ r \rangle \}$
$\mathtt{t[I]} := \mathtt{t[I+1]}$ (10)
$\{ \langle \mathrm{lt}(\mathtt{t}, m, n, p, q),\ \exists i, j, k, \ell : r \wedge m = \mathtt{I} \wedge n = p = q = \mathtt{I} + 1 \rangle \}$.

Abstract Strongest Postconditions for Assignments (Cont'd)

The same way if $\mathtt{t} \notin \mathrm{var}[\![r]\!]$ and $r \Rightarrow (\mathtt{I} \in [i, j] \wedge \mathtt{J} \in [i, j]) \vee (\mathtt{J} \in [k, \ell] \wedge \mathtt{I} \in [k, \ell])$ then:

$\{ \langle \mathrm{lt}(\mathtt{t}, i, j, k, \ell),\ r \rangle \}$
$\mathtt{t[I]} := \mathtt{t[J]}$ (11)
$\{ \langle \mathrm{lt}(\mathtt{t}, i, j, k, \ell),\ r \rangle \}$

since the swap of the array elements does not interfere with the assertions.

Generic Comparison Widening

Finally the abstract domain must be equipped with a widening (and optionally a narrowing to improve precision) to speed up the convergence of iterative fixpoint computations [4]. We choose to define the widening ∇ as:

$$\langle \mathrm{lt}(\mathtt{t}, i, j, k, \ell),\ r \rangle \nabla \langle \mathrm{lt}(\mathtt{t}, m, n, p, q),\ r' \rangle = \quad (12)$$
$$\text{let } \langle \mathrm{lt}(\mathtt{t}, r, s, t, u),\ r'' \rangle = \langle \mathrm{lt}(\mathtt{t}, i, j, k, \ell),\ r \rangle \vee \langle \mathrm{lt}(\mathtt{t}, m, n, p, q),\ r' \rangle \text{ in}$$
$$\langle \mathrm{lt}(\mathtt{t}, r, s, t, u),\ r \nabla r'' \rangle \ .$$

Generic Comparison Widening (Cont'd)

Typically, when handling loops, one encounters widenings of the form $r \mathbin{\nabla} \langle \mathrm{lt}(\mathtt{t}, m, n, p, q),\ r' \rangle$ where r corresponds to the loop entry condition while the term $\mathrm{lt}(\mathtt{t}, m, n, p, q)$ appears during the analysis of the loop body. There are several ways to handle this situation:

1. Incorporate the term $\mathrm{lt}(\mathtt{t}, i, j, k, \ell)$ in the form of a tautology, as already described in (5) for the abstract disjunction;
2. Use disjunctive completion (see ??) to preserve the disjunction within the loop (which may ultimately lead to infinite disjunctions) or better allow only abstract predicates of the more restricted form $r \vee \langle \mathrm{lt}(\mathtt{t}, m, n, p, q),\ r' \rangle$ (which definitively avoids the previous potential explosion);

3. Use *semantically loop unrolling* (as in [2, Sec. 6.5]) so that the loop:

$$\texttt{while } B \texttt{ do } C \texttt{ od}$$

is handled in the abstract semantics as if written in the form:

$$\texttt{if } B \texttt{ then } C;\ \texttt{while } B \texttt{ do } C \texttt{ od fi}$$

which is equivalent in the concrete semantics. More generally, if several abstract terms of different kinds are considered (like $\mathrm{lt}(\mathtt{t}, i, j, k, \ell)$ and $\mathrm{s}(\mathtt{t}, m, n)$ in the forthcoming 17), a further semantic unrolling can be performed each time a term of a new kind does appear, while all terms of the same king are merged by the widening.

Refined Generic Comparison Abstract Domains

- The generic comparison abstract domain $\mathcal{D}_{\mathrm{lt}}(X)$ of 3.1 may be imprecise since it allows only for one term $\langle \mathrm{lt}(\mathtt{t}, a, b, c, d),\ r \rangle$.
- First we could consider several arrays, with one such term per array.
- Second, we could consider the conjunction of such terms for a given array, which is more precise but may potentially lead to infinite conjunctions within loops (e.g. for which termination cannot be established).
- So we will consider this alternative within tests only, then applying the above abstract domain operators term by term[1].

[1] For short we avoid to resort to semantical loop unrolling which is better adapted to automatization but would yield to lengthy handmade calculations in this section. This technique will be illustrated anyway in the forthcoming 17.

- The same way we could the disjunctive completion of this domain, that is terms of the form $\vee_i \wedge_j \langle \mathrm{lt}(\mathtt{t}, a_{ij}, b_{ij}, c_{ij}, d_{ij}),\ r_{ij} \rangle$. This would introduce an exponential complexity factor, which we prefer to avoid. If necessary, we will use *local trace partitioning* [2, Sec. 6.6] instead.

Generic Comparison Static Program Analysis

Let us consider the following program (where $a \leq b$) which is similar to the inner loop of bubble sort [10]:

```
      var t : array [a, b] of int;
1 :
      I := a;
2 :
      while (I < b) do
3 :
          if (t[I]  >  t[I + 1]) then
4 :
              t[I]  :=:  t[I + 1]
5 :
          fi;
6 :
          I  :=  I + 1
7 :
      od
8 :
```


Generic Choice of the Generic Relational Integer Abstract Domain

- We let P_p^i be the value of the local predicate attached to the program point $p = 1, \ldots, 8$ at the i^{th} iteration.
- Initially, $P_1^0 = (\mathtt{a} \leq \mathtt{b})$ while $P_p^0 = \mathrm{false}$ for $p = 2, \ldots, 8$.
- We choose the octagonal abstract domain [12, 13] as the generic relational integer abstract domain $\mathcal{D}_{\mathrm{rel}}(X)$ parameterized by the set X of program variables $\mathtt{I}$, $\mathtt{J}$,... and auxiliary variables i, j, etc.

Fixpoint Iterates

The fixpoint iterates are as follows:

$P_1^1 = (\mathtt{a} \leq \mathtt{b})$ ≀initialization to P_1^0≀

$P_2^1 = (\mathtt{I} = \mathtt{a} \leq \mathtt{b})$ ≀assignment $(\mathtt{I} := \mathtt{a})$≀

$P_3^1 = (\mathtt{I} = \mathtt{a} < \mathtt{b})$ ≀loop condition $\mathtt{I} < \mathtt{b}$≀

$P_4^1 = \langle \mathrm{lt}(\mathtt{t}, i, j, k, l),\ i = k = \ell = \mathtt{I} = \mathtt{a} < \mathtt{b} \wedge j = \mathtt{I} + 1 \rangle$ ≀by (7) for test condition $(\mathtt{t}[\mathtt{I}] > \mathtt{t}[\mathtt{I}+1])$≀

$P_5^1 = \langle \mathrm{lt}(\mathtt{t}, m, n, p, q),\ \exists i, j, k, \ell : i = k = \ell = \mathtt{I} = \mathtt{a} < \mathtt{b} \wedge j = \mathtt{I} + 1 \wedge$

≀by assignment (10) which, by octagonal projection, simplifies into:≀

$= \langle \mathrm{lt}(\mathtt{t}, m, n, p, q),\ m = \mathtt{I} = \mathtt{a} < \mathtt{b} \wedge n = p = q = \mathtt{I} + 1 \rangle$

$P_6^1 = (P_3^1 \wedge \langle \mathrm{lt}(\mathtt{t}, i, j, k, \ell),\ i = \mathtt{I} = \mathtt{a} < \mathtt{b} \wedge j = k = \ell = \mathtt{I} + 1 \rangle) \vee P_5^1$

≀by (8) for test condition $(\mathtt{t}[\mathtt{I}] > \mathtt{t}[\mathtt{I}+1])$ and join (9)≀

$= (\langle \mathrm{lt}(\mathtt{t}, i, j, k, \ell),\ i = \mathtt{I} = \mathtt{a} < \mathtt{b} \wedge j = k = \ell = \mathtt{I} + 1 \rangle) \vee (\langle \mathrm{lt}(\mathtt{t}, m, n, p, q),\ m = \mathtt{I} = \mathtt{a} < \mathtt{b} \wedge n = p = q = \mathtt{I} + 1 \rangle)$

≀by def. P_3^1 and (2) as well as by def. of P_5^1≀

$= \langle \mathrm{lt}(\mathtt{t}, a, b, c, d),\ (\exists i, j, k, \ell : a = i \wedge b = j \wedge c = k \wedge d = \ell \wedge i = \mathtt{I}$

≀by def. (4) of the abstract union $\vee$≀

$= \langle \mathrm{lt}(\mathtt{t}, a, b, c, d),\ (a = \mathtt{I} = \mathtt{a} < \mathtt{b} \wedge b = c = d = \mathtt{I} + 1) \vee (a = \mathtt{I} =$

≀by octagonal projection≀

$= \langle \mathrm{lt}(\mathtt{t}, a, b, c, d),\ a = \mathtt{I} = \mathtt{a} < \mathtt{b} \wedge b = c = d = \mathtt{I} + 1 \rangle$ ≀by octagonal disjunction≀

$P_7^1 = \langle \mathrm{lt}(t, a, b, c, d),\ a = I - 1 = a < b \wedge b = c = d = I\rangle$ $\wr$by invertible assignment I := I + 1$\int$

$= \langle \mathrm{lt}(t, a, b, c, d),\ I = a + 1 = a + 1 \le b \wedge b = c = d = I\rangle$ $\wr$octagonal simplification$\int$

$P_3^2 = (P_2^1 \vee P_7^1) \wedge (I < b)$ $\wr$loop condition $I < b$ and absence of widening on first iterate$\int$

$= ((I = a \le b) \vee (\langle \mathrm{lt}(t, a, b, c, d),\ I = a + 1 = a + 1 \le b \wedge b = c = d$
$(I < b)$ $\wr$def. P_2^1 and $P_7^1$$\int$

$= ((\langle \mathrm{lt}(t, i, j, k, \ell),\ (i = j = k = \ell = I = a \le b) \vee (\exists a, b, c, d : i = a$
$(I < b)$

$\wr$def. (5) of abstract disjunction, the octagonal predicate depending only on I, a and b which leads to the selection of I, the only of these variables which is modified within the loop body$\int$ (13)

$= ((\langle \mathrm{lt}(t, i, j, k, \ell),\ (i = j = k = \ell = I = a \le b) \vee (I = i + 1 = a +$
$(I < b)$ $\wr$by octagonal projection$\int$

$= ((\langle \mathrm{lt}(t, i, j, k, \ell),\ (i = j = k = \ell = I = a < b) \vee (I = i + 1 = a +$
$\wr$by octagonal conjunction$\int$

$= \langle \mathrm{lt}(t, i, j, k, \ell),\ i = a \le j = k = \ell = I \le a + 1 \le b\rangle$ $\wr$by octagonal disjunction$\int$

$P_3^3 = P_3^2\ \nabla\ \langle \mathrm{lt}(t, i, j, k, \ell),\ i = a \le j = k = \ell = I \le a + 2 \le b\rangle$ $\wr$in absence of stabilization of the iterates, by a similar computation at the next iteration$\int$

$= \langle \mathrm{lt}(t, i, j, k, \ell),\ i = a \le j = k = \ell = I < b\rangle$ $\wr$by def. (12) of the widening $\nabla$$\int$

$P_4^3 = P_3^3 \wedge \langle \mathrm{lt}(t, m, n, p, q),\ m = p = q = I \wedge n = I + 1\rangle$ $\wr$by (7) for test condition (t[I] > t[I + 1])$\int$

$= \langle \mathrm{lt}(t, i, j, k, \ell),\ i = a \le j = k = \ell = I < b\rangle \wedge$
$\langle \mathrm{lt}(t, m, n, p, q),\ m = p = q = I \wedge n = I + 1\rangle$ $\wr$by def. P_4^3, the conjunction being left symbolic since it cannot be simplified, see 3.1$\int$

$P_5^3 = \langle \mathrm{lt}(t, i, j, k, \ell),\ i = a \le j = k = \ell = I < b\rangle \wedge$
$\langle \mathrm{lt}(t, i, j, k, \ell),\ \exists m, n, p, q : m = p = q = I \wedge n = I + 1 \wedge i = I \wedge$
$\wr$by (11) and (10) where $t \notin \mathrm{var}[\![d]\!]$ and $d \Rightarrow m = p = q = I \wedge n = I + 1$$\int$

$= \langle \mathrm{lt}(t, i, j, k, \ell),\ i = a \le j = k = \ell = I < b\rangle \wedge$
$\langle \mathrm{lt}(t, i', j', k', \ell'),\ i' = I \wedge j' = k' = \ell' = I + 1\rangle$ $\wr$by octagonal projection$\int$

$= \langle \mathrm{lt}(t, i, j, k, \ell),\ i = a \le j = k = \ell = I + 1 \le b\rangle$ $\wr$by def. (3), of conjunction and octagonal projection$\int$

$P_6^3 = (\langle \mathrm{lt}(t, i, j, k, \ell),\ i = a \le j = k = \ell = I < b\rangle \wedge$
$\langle \mathrm{lt}(t, i', j', k', \ell'),\ i' = I \wedge j' = k' = \ell' = I + 1\rangle) \vee$
$\langle \mathrm{lt}(t, i'', j'', k'', \ell''),\ i'' = a \le j'' = k'' = \ell'' = I + 1 \le b\rangle$
$\wr$by $P_6^3 = (P_3^3 \wedge (t[I] \le t[I + 1])) \vee P_5^3$ and (8)$\int$

$= \langle \mathrm{lt}(t, i, j, k, \ell),\ i = a \le j = k = \ell = I + 1 \le b\rangle \vee$
$\langle \mathrm{lt}(t, i'', j'', k'', \ell''),\ i'' = a \le j'' = k'' = \ell'' = I + 1 \le b\rangle$
$\wr$by def. (3), of conjunction and octagonal projection$\int$

$= \langle \mathrm{lt}(t, i, j, k, \ell),\ i = a \le j = k = \ell = I + 1 \le b\rangle$ $\wr$by $P \vee P = P$$\int$

$P_7^3 = \langle \mathrm{lt}(t, i, j, k, \ell),\ i = a \le j = k = \ell = I \le b\rangle$ $\wr$by assignment I := I + 1$\int$

Now the iterates have stabilized since:

$$(P_2^3 \vee P_7^3) \wedge (I < b)$$

$$= (P_2^1 \vee P_7^3) \wedge (I < b) \qquad \wr\text{since } P_2^3 = P_2^1 \text{ is stable}\int$$

$$
\begin{array}{ll}
= & ((\mathtt{I} = \mathtt{a} \leq \mathtt{b}) \vee \langle \mathrm{lt}(\mathtt{t}, i, j, k, \ell),\ i = \mathtt{a} \leq j = k = \ell = \mathtt{I} \leq \mathtt{b}\rangle) \wedge \\
& (\mathtt{I} < \mathtt{b}) \qquad \wr\text{def. } P_2^1 \text{ and } P_7^3 \int \\
= & ((\langle \mathrm{lt}(\mathtt{t}, i, j, k, \ell),\ (i = j = k = \ell = \mathtt{I} = \mathtt{a} \leq \mathtt{b}) \vee (\exists a, b, c, d : i = a \\
& (\mathtt{I} < \mathtt{b}) \ \wr\text{def. } (5) \text{ of abstract disjunction with selection of } \\
& \mathtt{I} \text{ as in } (??)\int \\
= & ((\langle \mathrm{lt}(\mathtt{t}, i, j, k, \ell),\ (i = j = k = \ell = \mathtt{I} = \mathtt{a} \leq \mathtt{b}) \vee (j = k = \ell = \mathtt{I} = \\
& (\mathtt{I} < \mathtt{b}) \qquad \wr\text{by octagonal projection}\int \\
= & ((\langle \mathrm{lt}(\mathtt{t}, i, j, k, \ell),\ i = \mathtt{a} \leq j = k = \ell = \mathtt{I} \leq \mathtt{b} \wedge \mathtt{a} \leq \mathtt{b}\rangle) \quad \wedge \\
& (\mathtt{I} < \mathtt{b}) \qquad \wr\text{by octagonal disjunction}\int \\
= & \langle \mathrm{lt}(\mathtt{t}, i, j, k, \ell),\ i = \mathtt{a} \leq j = k = \ell = \mathtt{I} < \mathtt{b}\rangle \quad \wr\text{by abstract} \\
& \text{conjunction } (2)\int \\
\Rightarrow & P_3^3 \qquad \wr\text{by def. } (1) \text{ of abstract implication}\int
\end{array}
$$

It remains to compute the loop exit invariant:

$$(P_2^3 \vee P_7^3) \wedge (\mathtt{I} \geq \mathtt{b})$$

$$
\begin{array}{ll}
= & ((\langle \mathrm{lt}(\mathtt{t}, i, j, k, \ell),\ i = \mathtt{a} \leq j = k = \ell = \mathtt{I} \leq \mathtt{b} \wedge \mathtt{a} \leq \mathtt{b}\rangle) \quad \wedge \\
& (\mathtt{I} \geq \mathtt{b}) \qquad \wr\text{by octagonal disjunction}\int \\
= & \langle \mathrm{lt}(\mathtt{t}, i, j, k, \ell),\ i = \mathtt{a} \leq j = k = \ell = \mathtt{I} = \mathtt{b}\rangle \quad \wr\text{by abstract} \\
& \text{conjunction } (2)\int
\end{array}
$$

The static analysis has therefore discovered the following invariants:


```
      var t : array [a, b] of int;
1 :   {a ≤ b}
      I := a;
2 :   {I = a ≤ b}
      while (I < b) do
3 :       {lt(t, a, I, I, I) ∧ I < b}
          if (t[I] > t[I + 1]) then
4 :           {lt(t, a, I, I, I) ∧ I < b ∧ lt(t, I, I + 1, I, I)}
              t[I] :=: t[I + 1]
5 :           {lt(t, a, I + 1, I + 1, I + 1) ∧ I + 1 ≤ b}
          fi;
6 :       {lt(t, a, I + 1, I + 1, I + 1) ∧ I + 1 ≤ b}
          I := I + 1
7 :       {lt(t, a, I, I, I) ∧ I ≤ b}
      od
8 :   {lt(t, a, I, I, I) ∧ I = b}
```


Generic Sorting Abstract Domain

Then we define the generic sorting abstract domain:

$$\mathcal{D}_s(X) = \{\langle \mathrm{s}(\mathtt{t}, a, b),\ r\rangle \mid \mathtt{t} \in X \wedge a, b \notin X \wedge r \in \mathcal{D}_{\mathrm{rel}}(X \cup \{a, b\})\} .$$

The meaning $\gamma(\langle \mathrm{s}(\mathtt{t}, a, b),\ r\rangle)$ of an abstract predicate $\langle \mathrm{s}(\mathtt{t}, a, b),\ r\rangle$ is, informally that the elements of $\mathtt{t}$ between indices a and b are sorted:

$$
\begin{aligned}
\gamma(\langle \mathrm{s}(\mathtt{t}, a, b),\ r\rangle) = {} & \exists a, b : \mathtt{t}.\ell \leq a \leq b \leq \mathtt{t}.h \wedge \\
& \forall i, j \in [a, b] : (i \leq j) \Rightarrow (\mathtt{t}[i] \leq \mathtt{t}[j]) \wedge r .
\end{aligned}
$$

Generic Comparison and Sorting Abstract Domain

The analysis of sorting algorithms involves the reduced product [5] of the generic comparison abstract domain of 3.1 and sorting abstract domain of 14, that is triples of the form:

$$\langle \mathrm{lt}(\mathtt{t}, a, b, c, d),\ \mathrm{s}(\mathtt{t}, e, f),\ r\rangle \ .$$

Reduction

The reduction involves interactions between terms such as, e.g.:

$$\mathrm{lt}(\mathtt{t}, a, b-1, b-1, b-1) \wedge \mathrm{lt}(\mathtt{t}, a, b, b, b) \Rightarrow \mathrm{s}(\mathtt{t}, b-1, b) \wedge \mathrm{lt}(\mathtt{t}, a, b-1, b-1, b) \quad (15)$$

$$\mathrm{s}(\mathtt{t}, b+1, c) \wedge \mathrm{lt}(\mathtt{t}, a, b+1, b+1, c) \wedge \mathrm{lt}(\mathtt{t}, a, b, b, b) \Rightarrow \mathrm{s}(\mathtt{t}, b, c) \wedge \mathrm{lt}(\mathtt{t}, a, b, b, c) \quad (16)$$

$$\mathrm{lt}(\mathtt{t}, a, a+1, a+1, b) \wedge \mathrm{s}(\mathtt{t}, a+1, b) \Rightarrow \mathrm{s}(\mathtt{t}, a, b) \quad (17)$$

The reduction [5] also involves the refinement of abstract predicate transformers (see a.o. [3, 11]) which would be performed automatically e.g. if the abstract predicate transformers are obtained by automatic simplification of the formula $\alpha \circ F \circ \gamma$ (where F is the concrete semantics) by the simplifier of a theorem prover.

Generic Comparison & Sorting Static Program Analysis

Let us consider the bubble sort [10]:


```
var t : array [a, b] of int;
J := b;
while (a < J) do
    I := a;
    while (I < J) do
        if (t[I] > t[I + 1]) then
            t[I] :=: t[I + 1]
        fi;
        I := I + 1
    od;
    J := J - 1
od
```

Fixpoint Approximation

The fixpoint approximation is as follows ($P_p^{i,k}$ denotes the local assertion attached to program point p at the i^{th} iteration and k^{th} loop unrolling, $P_p^i = P_p^{i,0}$ where $k = 0$ means that the decision to semantically unroll the loop is not yet taken):

$$
\begin{array}{lll}
P_1^0 = & (\mathtt{a} \leq \mathtt{b}) & \wr\text{initialization}\wr \\
P_i^0 = & \text{false},\ i = 2, \ldots, 8 & \\
P_1^1 = & P_1^0 & \\
= & (\mathtt{a} \leq \mathtt{b}) & \wr\text{def. } P_1^0\wr \\
P_2^1 = & (\mathtt{a} \leq \mathtt{b} = \mathtt{J}) & \wr\text{assignment } \mathtt{J} := \mathtt{b}\wr \\
P_3^{1,0} = & (\mathtt{a} < \mathtt{b} = \mathtt{J}) & \wr\text{test } (\mathtt{a} < \mathtt{J})\wr
\end{array}
$$

...

$P_{10}^{1,0}$ = $\mathrm{lt}(\mathtt{t}, \mathtt{a}, \mathtt{I}, \mathtt{I}, \mathtt{I}) \wedge \mathtt{a} < \mathtt{b} = \mathtt{I} = \mathtt{J}$ [2] ≀as in 3.1 since the inner loop does not modify a, b or I≀

$\Rightarrow$ $\mathrm{lt}(\mathtt{t}, \mathtt{a}, \mathtt{J}, \mathtt{J}, \mathtt{b}) \wedge \mathtt{a} < \mathtt{b} = \mathtt{J}$ ≀by elimination (octagonal projection) of program variable I which is no longer live at program point 10≀

$P_{11}^{1,0}$ = $\mathrm{lt}(\mathtt{t}, \mathtt{a}, \mathtt{J}+1, \mathtt{J}+1, \mathtt{b}) \wedge \mathtt{a} < \mathtt{b} \wedge \mathtt{J} = \mathtt{b} - 1$ ≀postcondition for assignment $\mathtt{J} := \mathtt{J} - 1$≀

$P_3^{1,1}$ = $\mathrm{lt}(\mathtt{t}, \mathtt{a}, \mathtt{J}+1, \mathtt{J}+1, \mathtt{b}) \wedge \mathtt{a} < \mathtt{J} = \mathtt{b} - 1$ ≀by semantical loop unrolling (since a new symbolic "lt" term has appeared, see 3.1,) and test $(\mathtt{a} < \mathtt{J})$≀

...

$P_{10}^{1,1}$ = $\mathrm{lt}(\mathtt{t}, \mathtt{a}, \mathtt{J}+1, \mathtt{J}+1, \mathtt{J}+1) \wedge \mathtt{a} < \mathtt{J} = \mathtt{b} - 1 \wedge \mathrm{lt}(\mathtt{t}, \mathtt{a}, \mathtt{I}, \mathtt{I}, \mathtt{I}) \wedge \mathtt{I} = \mathtt{J}$

≀as in 3.1 since the inner loop does (18) not modify a, b or I and the swap $\mathtt{t}[\mathtt{I}] :=: \mathtt{t}[\mathtt{I}+1]$ does not interfere with $\mathrm{lt}(\mathtt{t}, \mathtt{a}, \mathtt{J}+1, \mathtt{J}+1, \mathtt{J}+1)$ according to $\mathtt{a} \leq \mathtt{I} < \mathtt{I}+1 \leq \mathtt{J} < \mathtt{J}+1$ so $\mathtt{I}, \mathtt{I}+1 \in [\mathtt{a}, \mathtt{J}+1]$ and (11)≀

$\Rightarrow$ $\mathrm{lt}(\mathtt{t}, \mathtt{a}, \mathtt{J}+1, \mathtt{J}+1, \mathtt{J}+1) \wedge \mathrm{lt}(\mathtt{t}, \mathtt{a}, \mathtt{J}, \mathtt{J}, \mathtt{J}) \wedge \mathtt{a} < \mathtt{J} = \mathtt{b} - 1$ ≀by elimination of I is dead at program point 10≀

$\Rightarrow$ $\mathrm{s}(\mathtt{t}, \mathtt{J}, \mathtt{b}) \wedge \mathrm{lt}(\mathtt{t}, \mathtt{a}, \mathtt{J}, \mathtt{J}, \mathtt{b}) \wedge \mathtt{a} < \mathtt{J} = \mathtt{b} - 1$ ≀by reduction (15)≀

$P_{11}^{1,1}$ = $\mathrm{s}(\mathtt{t}, \mathtt{J}+1, \mathtt{b}) \wedge \mathrm{lt}(\mathtt{t}, \mathtt{a}, \mathtt{J}+1, \mathtt{J}+1, \mathtt{b}) \wedge \mathtt{a} \leq \mathtt{J} = \mathtt{b} - 2$ ≀by assignment $\mathtt{J} := \mathtt{J} - 1$≀

$P_3^{1,2}$ = $\mathrm{s}(\mathtt{t}, \mathtt{J}+1, \mathtt{b}) \wedge \mathrm{lt}(\mathtt{t}, \mathtt{a}, \mathtt{J}+1, \mathtt{J}+1, \mathtt{b}) \wedge \mathtt{a} < \mathtt{J} = \mathtt{b} - 2$ ≀by semantical loop unrolling (since a new symbolic "s" term has appeared, see 3.1,) and test $(\mathtt{a} < \mathtt{J})$≀

...

$P_{10}^{1,2}$ = $\mathrm{s}(\mathtt{t}, \mathtt{J}+1, \mathtt{b}) \wedge \mathrm{lt}(\mathtt{t}, \mathtt{a}, \mathtt{J}+1, \mathtt{J}+1, \mathtt{b}) \wedge \mathtt{a} < \mathtt{J} = \mathtt{b} - 2 \wedge \mathrm{lt}(\mathtt{t}, \mathtt{a}, \mathtt{I}, \mathtt{I}, \mathtt{I}) \wedge \mathtt{I} = \mathtt{J}$ ≀by 3.1 and non interference, see (18)≀

$\Rightarrow$ $\mathrm{s}(\mathtt{t}, \mathtt{J}+1, \mathtt{b}) \wedge \mathrm{lt}(\mathtt{t}, \mathtt{a}, \mathtt{J}+1, \mathtt{J}+1, \mathtt{b}) \wedge \mathtt{a} < \mathtt{J} = \mathtt{b} - 2 \wedge \mathrm{lt}(\mathtt{t}, \mathtt{a}, \mathtt{J}, \mathtt{J}, \mathtt{J})$ ≀since I is dead≀

$\Rightarrow$ $\mathrm{s}(\mathtt{t}, \mathtt{J}, \mathtt{b}) \wedge \mathrm{lt}(\mathtt{t}, \mathtt{a}, \mathtt{J}, \mathtt{J}, \mathtt{b}) \wedge \mathtt{a} < \mathtt{J} = \mathtt{b} - 2$ ≀by reduction (16)≀

$P_{11}^{1,2}$ = $\mathrm{s}(\mathtt{t}, \mathtt{J}+1, \mathtt{b}) \wedge \mathrm{lt}(\mathtt{t}, \mathtt{a}, \mathtt{J}+1, \mathtt{J}+1, \mathtt{b}) \wedge \mathtt{a} \leq \mathtt{J} = \mathtt{b} - 3$ ≀by assignment $\mathtt{J} := \mathtt{J} - 1$≀

$$
\begin{aligned}
P_3^{2,2} &= (P_3^{1,2} \mathbin{\nabla} (P_{11}^{1,2} \wedge (\mathtt{a} < \mathtt{J}))) \wedge (\mathtt{a} < \mathtt{J}) && \wr\,\text{loop unrolling stops in absence of new abstract term and widening speeds-up convergence}\,\int \\
&= ((\mathtt{s}(\mathtt{t}, \mathtt{J}+1, \mathtt{b}) \wedge \mathtt{lt}(\mathtt{t}, \mathtt{a}, \mathtt{J}+1, \mathtt{J}+1, \mathtt{b}) \wedge \mathtt{a} < \mathtt{J} = \mathtt{b}-2) \mathbin{\nabla} (\mathtt{s}(\mathtt{t}, \mathtt{J}+1, \mathtt{b}) \wedge \mathtt{lt}(\mathtt{t}, \mathtt{a}, \mathtt{J}+1, \mathtt{J}+1, \mathtt{b}) \wedge \mathtt{a} \leq \mathtt{J} = \mathtt{b}-3 \wedge (\mathtt{a} < \mathtt{J}))) \wedge (\mathtt{a} < \mathtt{J}) && \wr\,\text{def. } P_3^{1,2} \text{ and } P_{11}^{1,2}\,\int \\
&= \mathtt{s}(\mathtt{t}, \mathtt{J}+1, \mathtt{b}) \wedge \mathtt{lt}(\mathtt{t}, \mathtt{a}, \mathtt{J}+1, \mathtt{J}+1, \mathtt{b}) \wedge ((\mathtt{a} < \mathtt{J} = \mathtt{b}-2) \mathbin{\nabla} (\mathtt{a} < \mathtt{J} = \mathtt{b}-3)) \wedge (\mathtt{a} < \mathtt{J}) && \wr\,\text{by def. widening}\,\int \\
&= \mathtt{s}(\mathtt{t}, \mathtt{J}+1, \mathtt{b}) \wedge \mathtt{lt}(\mathtt{t}, \mathtt{a}, \mathtt{J}+1, \mathtt{J}+1, \mathtt{b}) \wedge \mathtt{a} < \mathtt{J} \leq \mathtt{b}-2 && \wr\,\text{by def. octagonal widening and conjunction}\,\int
\end{aligned}
$$

...

$$
\begin{aligned}
P_{10}^{2,2} &= \mathtt{s}(\mathtt{t}, \mathtt{J}+1, \mathtt{b}) \wedge \mathtt{lt}(\mathtt{t}, \mathtt{a}, \mathtt{J}+1, \mathtt{J}+1, \mathtt{b}) \wedge \mathtt{a} < \mathtt{J} \leq \mathtt{b}-2 \wedge \mathtt{lt}(\mathtt{t}, \mathtt{a}, \mathtt{I}, \mathtt{I}, \mathtt{I}) \wedge \mathtt{I} = \mathtt{J} && \wr\,\text{by 3.1 and non interference, see (18)}\,\int
\end{aligned}
$$

$$
\begin{aligned}
&= \mathtt{s}(\mathtt{t}, \mathtt{J}+1, \mathtt{b}) \wedge \mathtt{lt}(\mathtt{t}, \mathtt{a}, \mathtt{J}+1, \mathtt{J}+1, \mathtt{b}) \wedge \mathtt{a} < \mathtt{J} \leq \mathtt{b}-2 \wedge \mathtt{lt}(\mathtt{t}, \mathtt{a}, \mathtt{J}, \mathtt{J}, \mathtt{J}) && \wr\,\text{by elimination of the dead variable I}\,\int \\
&\Rightarrow \mathtt{s}(\mathtt{t}, \mathtt{J}, \mathtt{b}) \wedge \mathtt{lt}(\mathtt{t}, \mathtt{a}, \mathtt{J}, \mathtt{J}, \mathtt{b}) \wedge \mathtt{a} < \mathtt{J} \leq \mathtt{b}-2 && \wr\,\text{by reduction (16)}\,\int \\
P_{11}^{2,2} &= \mathtt{s}(\mathtt{t}, \mathtt{J}+1, \mathtt{b}) \wedge \mathtt{lt}(\mathtt{t}, \mathtt{a}, \mathtt{J}+1, \mathtt{J}+1, \mathtt{b}) \wedge \mathtt{a} \leq \mathtt{J} \leq \mathtt{b}-3 && \wr\,\text{by assignment } \mathtt{J} := \mathtt{J}-1\,\int
\end{aligned}
$$

Now $(P_{11}^{2,2} \wedge \mathtt{a} < \mathtt{J}) \Rightarrow P_3^{1,2}$ so that the loop iterates stabilize to a post-fixpoint. On loop exit, we must collect all cases following from semantic unrolling:

$$
\begin{aligned}
P_{12}^2 = {} & (P_2^1 \wedge \mathtt{a} \geq \mathtt{J}) && \wr\,\text{no entry in the loop}\,\int \\
& \vee (P_{11}^{1,0} \wedge \mathtt{a} \geq \mathtt{J}) && \wr\,\text{loop exit after one iteration}\,\int
\end{aligned}
$$

[2] Notice that this notation is a shorthand for the more explicit notation $\exists i, j, k, \ell : \mathrm{lt}(t, i, j, k, \ell) \wedge i = \mathtt{a} \wedge j = \mathtt{I} \wedge k = \mathtt{I} \wedge \ell = \mathtt{I}) \wedge \mathtt{a} < \mathtt{b} \wedge \mathtt{b} = \mathtt{J} \wedge \mathtt{I} = \mathtt{J}$ as used in 3.1, so that, in particular, we freely replace i, j, k and ℓ in $\mathrm{lt}(t, i, j, k, \ell)$ by equivalent expressions.

$$
\begin{aligned}
& \vee (P_{11}^{1,1} \wedge \mathtt{a} \geq \mathtt{J}) && \wr\,\text{loop exit after two iterations}\,\int \\
& \vee (P_{11}^{2,2} \wedge \mathtt{a} \geq \mathtt{J}) && \wr\,\text{loop exit after three iterations or more}\,\int \\
= {} & (\mathtt{a} = \mathtt{J} = \mathtt{b}) \vee (\mathtt{s}(\mathtt{t}, \mathtt{J}+1, \mathtt{b}) \wedge \mathtt{lt}(\mathtt{t}, \mathtt{a}, \mathtt{J}+1, \mathtt{J}+1, \mathtt{b}) \wedge \mathtt{a} = \mathtt{J} \leq \mathtt{b}-1) && \wr\,\text{def. abstract disjunction}\,\int \\
= {} & (\mathtt{a} = \mathtt{J} = \mathtt{b}) \vee (\mathtt{s}(\mathtt{t}, \mathtt{a}+1, \mathtt{b}) \wedge \mathtt{lt}(\mathtt{t}, \mathtt{a}, \mathtt{a}+1, \mathtt{a}+1, \mathtt{b}) \wedge \mathtt{a} < \mathtt{b}) && \wr\,\text{elimination of dead variable J}\,\int \\
= {} & (\mathtt{a} = \mathtt{b}) \vee (\mathtt{s}(\mathtt{t}, \mathtt{a}, \mathtt{b}) \wedge \mathtt{a} < \mathtt{b}) && \wr\,\text{by reduction (17)}\,\int \\
= {} & \mathtt{s}(\mathtt{t}, \mathtt{a}, \mathtt{b}) \wedge \mathtt{a} \leq \mathtt{b} && \wr\,\text{by definition of abstract disjunction similar to (5)}\,\int
\end{aligned}
$$

The sorting proof would proceed in the same way by proving that the final array is a permutation of the original one.


```
     var t : array [a, b] of int;
 1 : J := b;
 2 : while (a < J) do
 3 :    I := a;
 4 :    while (I < J) do
 5 :       if (t[I] > t[I + 1]) then
 6 :          t[I] :=: t[I + 1]
 7 :       fi;
 8 :       I := I + 1
 9 :    od;
10 :    J := J − 1
11 : od
12 : {s(t, a, b) ∧ a ≤ b}
```

Conclusion

- Observe that *generic predicate abstraction* is defined for a programming language as opposed to *ground predicate abstraction* which is specific to a program, a usual distinction between abstract interpretation based static program analysis (a generic abstraction for a set of programs) and abstract model checking (an abstract model for a given program).
- Notice that the so-called *polymorphic predicate abstraction* of [1] is an instance of symbolic relational separate procedural analysis [6, Sec. 7] for *ground* predicate abstraction.
- The generalization to generic predicate abstraction is immediate since it only depends on the way concrete predicate transformers are defined (see [6, Sec. 7]).

Bibliography

[1] T. Ball, T. Millstein, and S.K. Rajamani. Polymorphic predicate abstraction. Technical report MSR-TR-2001-10, Microsoft Reasearch, Redmond, Washington, United States, 17 June 2002. 21 p. 49

[2] B. Blanchet, P. Cousot, R. Cousot, J. Feret, L. Mauborgne, A. Miné, D. Monniaux, and X. Rival. Design and implementation of a special-purpose static program analyzer for safety-critical real-time embedded software, invited chapter. In T. Mogensen, D.A. Schmidt, and I.H. Sudborough, eds, *The Essence of Computation: Complexity, Analysis, Transformation. Essays Dedicated to Neil D. Jones*, LNCS 2566, pages 85–108. Springer, 2002. 22, 24

[3] A. Cortesi, B. Le Charlier, and P. van Hentenryck. Combinations of abstract domains for logic programming: open product and generic pattern construction. *Science of Computer Programming*, 38(1–3):27–71, 2000. 38

[4] P. Cousot and R. Cousot. Abstract interpretation: a unified lattice model for static analysis of programs by construction or approximation of fixpoints. In *Conference Record of the Fourth Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, pages 238–252, Los Angeles, California, 1977. ACM Press, New York, New York, United States. 20

[5] P. Cousot and R. Cousot. Systematic design of program analysis frameworks. In *Conference Record of the Sixth Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, pages 269–282, San Antonio, Texas, 1979. ACM Press, New York, New York, United States. 37, 38

[6] P. Cousot and R. Cousot. Modular static program analysis, invited paper. In R.N. Horspool, editor, *Proceedings of the Eleventh International Conference on Compiler Construction, CC '2002*, pages 159–178, Grenoble, France, April 6—14 2002. Lecture Notes in Computer Science 2304, Springer-Verlag, Berlin, Germany. 49

[7] P. Cousot and N. Halbwachs. Automatic discovery of linear restraints among variables of a program. In *Conference Record of the Fifth Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, pages 84–97, Tucson, Arizona, 1978. ACM Press, New York, New York, United States. 4

[8] S. Das and D.L. Dill. Counter-example based predicate discovery in predicate abstraction. In M. Aagaard and J.W. O'Leary, editors, *Proceedings of the Fourth International Conference on Formal Methods in Computer-Aided Design, FMCAD 2002*, Portland,, Oregon, United States, Lecture Notes in Computer Science 1633, pages 19–32. Springer-Verlag, Berlin, Germany, November 2002.

[9] G. Kildall. A unified approach to global program optimization. In *Conference Record of the First Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, pages 194–206, Boston, Massachusetts, October 1973. ACMpress. 3

[10] D.E. Knuth. Sorting and searching. In *The Art of Computer Programming*, volume 3. Addison-Wesley Pub. Co., Reading, Massachusetts, United States, 1973. 25, 39

[11] S. Lerner, D. Grove, and C. Chambers. Composing dataflow analyses and transformations. In *Conference Record of the Twentyninth Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, pages 270–282, Portland, Oregon, January 2002. ACM Press, New York, New York, United States. 38

[12] A. Miné. A new numerical abstract domain based on difference-bound matrices. In 0. Danvy and A. Filinski, editors, *Proceedings of the Second Symposium PADO '2001, Programs as Data Objects*, Århus, Denmark, 21–23 May 2001, Lecture Notes in Computer Science 2053, pages 155–172. Springer-Verlag, Berlin, Germany, 2001. http://www.di.ens.fr/~mine/publi/article-mine-padoII.pdf. 4, 26

[13] A. Miné. A few graph-based relational numerical abstract domains. In M. Hermenegildo and G. Puebla, editors, *SAS'02*, volume 2477 of *Lecture Notes in Computer Science*, pages 117–132. Springer-Verlag, Berlin, Germany, 2002. http://www.di.ens.fr/~mine/publi/article-mine-sas02.pdf. 4, 26

[14] J.H. Morris and B. Wegbreit. Subgoal induction. *Communications of the Association for Computing Machinary*, 20(4):209–222, April 1977. 15

THE END

More references at URL www.di.ens.fr/~cousot.