Thirty Years of Abstract Interpretation [1]

Patrick Cousot

École normale supérieure

45 rue d'Ulm, 75230 Paris cedex 05 (France)

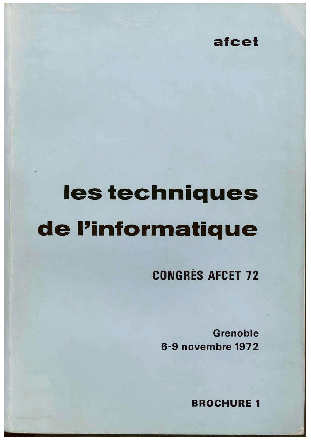
Patrick.Cousot@ens.fr www.di.ens.fr/~cousot

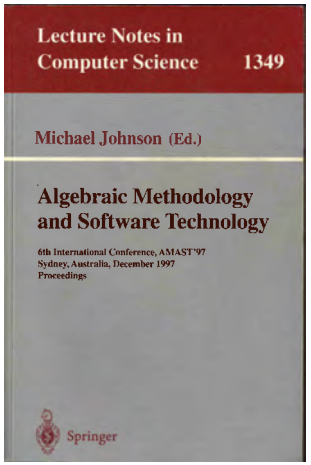
San Francisco, USA

[1] Joint work with Radhia Cousot.

1. Early days in Grenoble

Grammar analysis

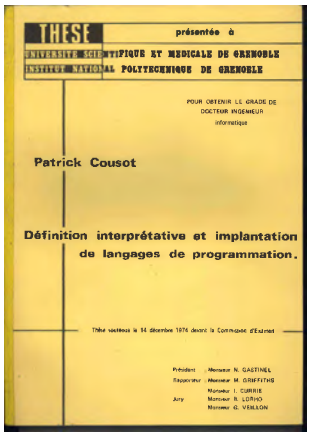




- My initial interest was in parsing, the fashionable subject of the time
- Designed a parsing algorithm by optimization of Earley's algorithm thanks to a static analysis of the grammar [2] [Cou72]
- This interest in grammars and grammar analysis has perpetuated [CC97]
- The understanding that parsing is an abstract interpretation is more recent [CC03, CC06]!

[2] my present understanding!

Operational semantics & compile-time optimization


THESE
présentée à
UNIVERSITE SCIENTIFIQUE ET MEDICALE DE GRENOBLE
INSTITUT NATIONAL POLYTECHNIQUE DE GRENOBLE
POUR OBTENIR LE GRADE DE DOCTEUR INGENIEUR
informatique
Patrick Cousot
Définition interprétative et implantation de langages de programmation.
Thèse soutenue le 14 décembre 1976 devant la Commission d'Examen
Président Monsieur N. GASTINEL
Rapporteur Monsieur M. GRIFFITHS
Monsieur I. CURRIE
Jury Monsieur R. LOTHO
Monsieur G. VERJUS

- Interpreter-based operational semantics of programming languages [Cou74]
- Procedure/library optimization by compile-time pre-evaluation [Cou74]
- Understand that program *semantic* analysis is required for program transformation
- The formalization of program transformation by abstract interpretation is more difficult and recent [CC02b]

The origin

– Radhia comes to my office to explain that she wants to check intervals by backward propagation [3]

– I try to convince her that forward propagation would be better, like an execution

– Everything goes well but for convergence in loops!

– We invent *widening* (denoted ∇ reminiscent of $\cup$)

– We get our first contract (IRIA-SESORI-75-035)

[3] à la Dijkstra's weakest precondition [Dij76]

– Bernard Lorho, reviewer, comes with Kildall's thesis [Kil72] and Wegbreit's report [Weg73] (pointing back to Sintzoff [Sin72], himself to Naur [Nau63, Nau65])

– Having discovered everything by ourselves, we are somewhat disappointed, but ...

– Our static analysis framework is more general [4], with arbitrary lattices [5] thanks to widening, and a correctness proof [6] with respect to a trace semantics.

---

[4] e.g. Kildall assumes distributivity while his main example, constant propagation, is not distributive.

[5] Naur and Sintzoff assume finiteness, Wegbreit and Kildall assume the ascending chain condition, later shown to be a severe limitation [CC92a].

[6] e.g. Sintzoff's rule of signs was wrong.

Widening

Excerpt form the first IRIA-SESORI-75-035 contract report [7]:

A6 : $E \times E \overset{\nabla}{\rightarrow} E$

A7 : $(x \cup y) \leq (x \nabla y)$

Hypothèse - H17

$\{\forall(v_1, \ldots, v_n, \ldots) \in V_a^\infty$,

$\forall(s_o, \ldots, s_n, \ldots) \in V_a^\infty$ |

$(s_o = \square, s_1 = s_o \bar{\nabla} v_1, \ldots, s_n = s_{n-1} \bar{\nabla} v_n, \ldots\}$

$\Rightarrow$ {la suite $s_o, s_1, \ldots, s_n, \ldots$ n'est pas infinie et strictement croissante pour $\bar{<}$ }

Théorème T401

La procédure "interprétation abstraite" termine pour tout graphe de programme.

[7] $\square$ is now $\bot$, we ignored the existence of denotational semantics at the time!

Example of widening [8]

```
(x ∇̄ y) = cas x ∈ V_a, y ∈ V_a alors
        | □, ? => y ;
        | ?, □ => x ;
        | [n1,m1],[n2,m2] =>
            [si n2 < n1 alors
                si v_i ≤ n2 alors v_i sinon -∞ fsi
             sinon n1 fsi ;
            si m2 > m1 alors
              si m2 ≤ v_s alors v_s sinon +∞ fsi
            sinon m1 fsi ; ]
        fincas ;
```

[8] with thresholds extracted from variable interval declarations `var v :` $v_i..v_s$.

Galois connections [9]

Définition - DEF1 - @ : abstraction d'ensembles de valeurs concrètes

$2^{V_c} \xrightarrow{@} V_a$

Hypothèse - H3

{@ est un homomorphisme de $(2^{V_c}, \cup)$ dans $(V_a, \bar{\cup})$}

$\overset{\text{def}}{\Longleftrightarrow}\{\forall (v_1, v_2) \in (2^{V_c})^2, @(v_1 \cup v_2) = @(v_1) \bar{\cup} @(v_2)\}$

Définition - DEF4 - γ : concrétisation de valeurs abstraites

$V_a \xrightarrow{\gamma} 2^{V_c}$

Les fonctions @ et γ sont liées par les relations suivantes :

Hypothèse - H5

$\{\forall v \in 2^{V_c}, \{v \subseteq \gamma(@(v))\}\}$

Hypothèse - H6

$\{\forall v \in V_a, \{v = @(\gamma(v))\}\}$

[9] The link with semi-dual Galois correspondances was not even done at that time!

Soundness (with respect to a trace semantics)

Quel que soit P un programme concret, notons :

$$P_{(a_1,\ldots,a_m)}(i)$$

la valeur de la variable i sur l'arc a_m, à la suite d'une exécution quelconque du programme P, le long du chemin $(a_1, \ldots, a_m)$, à partir de l'arc initial a_1 de P.

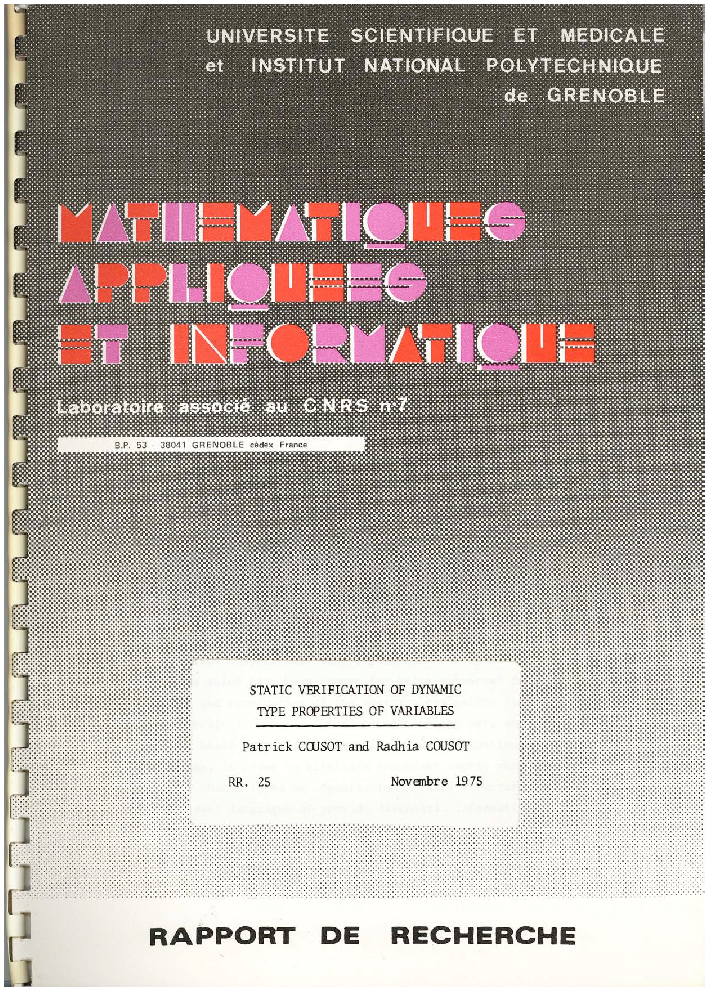
Notons C_{a_m}, le contexte local associé à l'arc a_m du graphe de P, par la procédure d'interprétation abstraite de P. La correction de la procédure d'interprétation découle du théorème suivant :

Théorème - T502

$\forall (a_1, \ldots, a_m), \quad \forall i \in I :$

$$P_{(a_1, \ldots, a_m)}(i) \in \gamma(C_{a_m}(i))$$

The abstract interpretor in the first research report [CC75]


UNIVERSITE SCIENTIFIQUE ET MEDICALE
et INSTITUT NATIONAL POLYTECHNIQUE
de GRENOBLE

MATHEMATIQUES APPLIQUEES ET INFORMATIQUE

Laboratoire associé au C.N.R.S. n°7

B.P. 53 38041 GRENOBLE cedex France

STATIC VERIFICATION OF DYNAMIC TYPE PROPERTIES OF VARIABLES

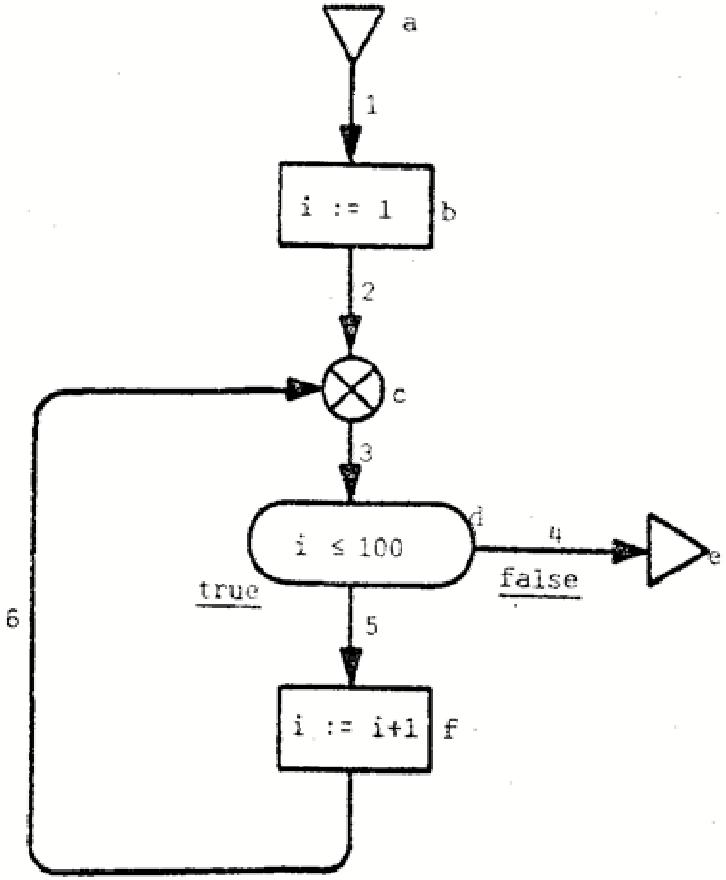
Patrick COUSOT and Radhia COUSOT

RR. 25 Novembre 1975

RAPPORT DE RECHERCHE

```
procedure abstract interpretation (graph) ;
begin
    for each arc of graph do local context (arc) := ϕ repeat ;
    execution paths := {entry-arc (entry-node (graph))} ; junctions := ∅ ;
    while (execution paths ≠ ∅) do
        while (execution paths ≠ ∅) do
            {choose an execution path}
            input arc := choose (execution paths) ;
            execution paths := execution paths - {input arc} ;
            node := final-end (input arc) ;
            case node of
                assignment node →
                assign output context (exit-arc (node), ∇(node,
                                                local context (input arc))) ;
                test node →
                    (C_T, C_F) := ∇(node, local context (input arc)) ;
                assign output context (true-exit-arc (node), C_T) ;
                assign output context (false-exit-arc (node), C_F) ;
                simple or loop junction node → junctions := junctions ∪{node} ;
                exit node , ;
            end ;
        repeat ;
        for each junction node of junctions do
            output context := ⊔ local context (input arc)
                              input arc ∈ entry-arc (junction node)
            if ¬(output context ⊆ local context (exit-arc(junction node))) then
                case junction node of
                    simple junction node →
                    assign output context (exit-arc(junction node),
                                           output context) ;
                    loop junction node →
                    assign output context (exit-arc(junction node),
                    local context (exit-arc(junction node)) ∇
                    output context) ;
                end ;
            fi ;
        repeat ;
        junctions := ∅ ;
    repeat ;
    return ;
    procedure assign output context (output arc, output context) ;
        if ¬(output context ⊆ local context (output arc)) then
            local context(output arc) := output context ;
            execution paths := execution paths ∪ {output arc} ;
        fi ;
end ;
```

All ideas come out from a simple example

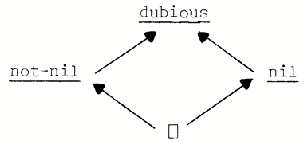


step	node	input arc	local context (input arc)	output context	execution paths	junctions
1	a				{1}	∅
2	b	(1)	∅	{i, [1, 1]}	{2}	∅
3	c	2	{i, [1, 1]}		{ }	{c}
4	c	2	{i, [1, 1]}			
				{i, [1, 1]}	{3}	∅
		6	∅			
5	d	3	{i, [1, 1]}	{i, [1, 1]}	{5}	∅
6	f	5	{i, [1, 1]}	{i, [2, 2]}	{6}	∅
7	c	6	{i, [2, 2]}		{ }	{c}
8	c	2	{i, [1, 1]}	{i, [1, 1] $\bar{\vee}$ [1, 2]}		
		6	{i, [2, 2]}	= {i, [1, +∞]}	{3}	∅
9	d	(3)	{i, [1, +∞]}	{i, [101, +∞]}	{4	∅
					,	
				{i, [1, 100]}	5}	
10	e	(4)	{i, [101 +∞]}		{5}	∅
11	f	(5)	{i, [1, 100]}	{i, [2, 101]}	{6}	∅
12	c	6	{i, [2, 101]}		{ }	{c}
13	c	(2)	{i, [1, 1]}	{i, [1, +∞] $\bar{\vee}$ ([1, 1] $\bar{\cup}$ [2, 101])}		
		(6)	{2, [2, 101]}	$\bar{\leq}$ {i, [1, +∞]). end.		

But apply to “significant” examples!

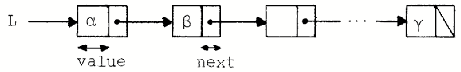
```
lwb := 1 ; upb := 100 ;
{(lwb, [1, 1]), (upb, [100, 100])}

L : {(lwb, [1, +∞]), (upb, [-∞, 100]), (m, [-∞, +∞])}

if upb < lwb then
     {(lwb, [1, +∞]), (upb, [-∞, 100]), (m, [-∞, +∞])}
       unsuccessfull search ;
fi ;

{(lwb, [1, 100]), (upb, [1, 100]), (m, [-∞, +∞])}
m := (upb + lwb) ÷ 2 ;

{(lwb, [1, 100]), (upb, [1, 100]), (m, [1, 100])}

if K = R(m) then
     successfull search ;

elsif K < R(m) then
     upb := m - 1 ;
     {(lwb, [1, 100]), (upb, [0, 99]), (m, [1, 100])}

else
     lwb := m + 1 ;
     {(lwb, [2, 101]), (upb, [1, 100]),(m, [1, 100])}
fi ;

{(lwb, [1, 101]), (upb, [0, 100]), (m, [1, 100])}

go to L ;
```

Binary search (numerical)



In the case of a finite number of abstract values, the widening $\bar{\nabla}$ is taken to be $\bar{\cup}$.

The problem consists in finding the K^{th} value of a linear linked list L :



The intended solution, with its analysis is the following :

```
     {(K, [-∞, +∞]), (L, dubious)}
     if K ≤ 0 then stop fi ;
     cursor := L ;
B :  {(K, [1, +∞]), (cursor, dubious), (L, dubious)}
     if K ≠ 1 then
          {(K, [2, +∞]), (cursor, dubious), ...}
          K := K - 1 ;
          {(K, [1, +∞], (cursor, dubious), ...}
          if cursor = nil then
               stop
          else
               {(K, [1, +∞]), (cursor, not-nil), ...}
[A]            cursor := next (cursor)
               {..., (cursor, dubious), ...}
          fi ;
     {(K, [1, +∞]), (cursor, dubious), (L, dubious)}
     go to B
     fi ;
     {(K, [1, 1]), (cursor, dubious), (L, dubious) ;
[B]  ... value (cursor) ...
```

Pointer analysis (symbolic)

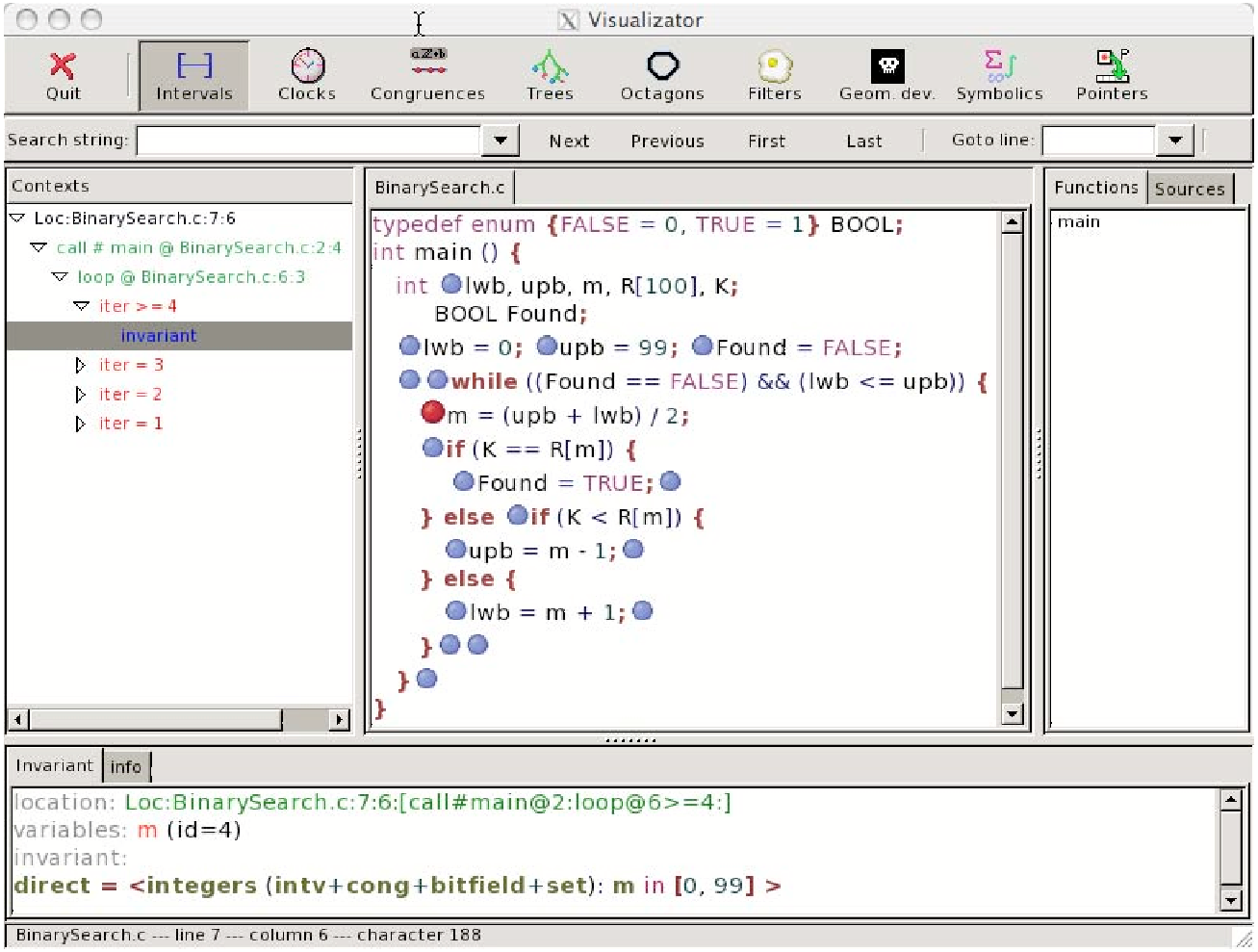
The first publication [CC76], now with narrowing

step	node	input arc	local context (input arc)	output context	execution paths	junctions
1	a			ψ	{1}	∅
2	b	(1)	ψ	{i, [1,1]}	{2}	∅
3	c	2	{i, [1,1]}		∅	{c}
4	c	2	{i, [1,1]}			
				ψ	{3}	∅
		6	ψ			
5	d	3	ψ	{i, [101,+∞]}	{4	∅
				{i, [-∞,100]}	5}	
6	e	4	{i, [101,+∞]}		{5}	∅
7	f	(5)	{i, [-∞,100]}	{i, [-∞,101]}	{6}	∅
8	c	6	{i, [-∞,101]}		∅	{c}
9	c	(2)	{i, [1,1]}			
				{i, [-∞,+∞]∆ [-∞,101]}		
		(6)	{i, [-∞,101]}	={i, [-∞,101]}	{3}	∅
10	d	(3)	{i, [-∞,101]}	{i, [101,101]}	{4}	
				{i, [-∞,100]} = {i, [-∞,100]}. end.		
11	e	(4)	{i, [101,101]}		∅	∅

The “significant” example with widening and narrowing

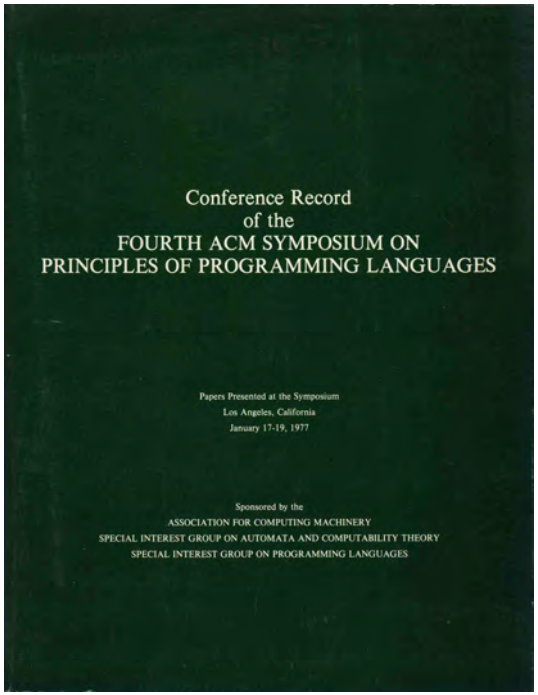

```
lwb := 1 ; upb := 100 ;
{(lwb, [1,1]), (upb, [100, 100])}
L : {(lwb, [1,101]), (upb, [0,100]), (m, [1,100])}
if upb < lwb then
      {(lwb, [1,101]), (upb, [0,100]), (m, [1,100])}
        unsuccessfull search ;
fi ;
{(lwb, [1,100]), (upb, [1,100]), (m, [1,100])}
m := (upb + lwb) ÷ 2 ;
{(lwb, [1,100]), (upb, [1,100]), (m, [1,100])}
if K = R(m) then
        successfull search ;
elsif K < R(m) then
        upb := m - 1 ;
       {(lwb, [1,100]), (upb, [0,99]), (m, [1,100])}
else
        lwb := m + 1 ;
        {(lwb, [2, 101]), (upb, [1,100]), (m, [1,100])}
fi ;
{(lwb, [1,101]), (upb, [0,100]), (m, [1,100])}
go to L ;
```

The binary search example ... with ASTRÉE

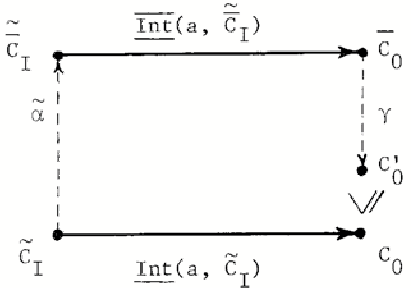


2. The great leap forward

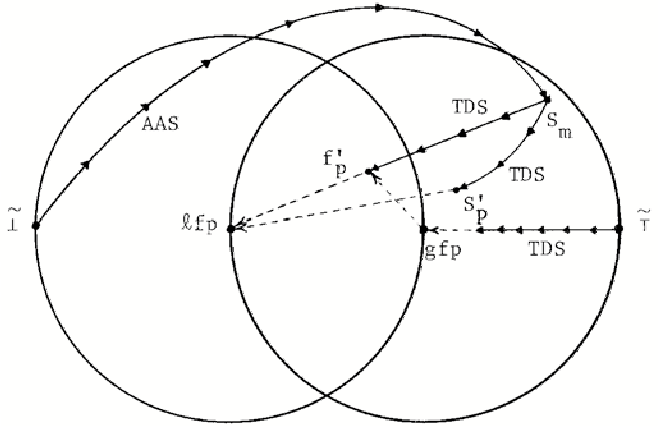
“Program analyzes are abstract interpretations” (POPL 77)



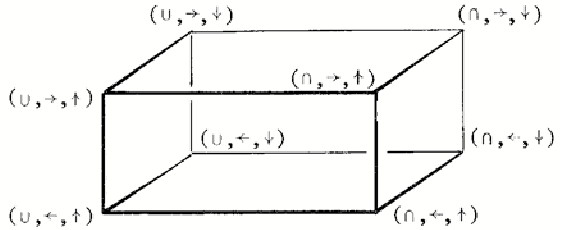
Abstraction:



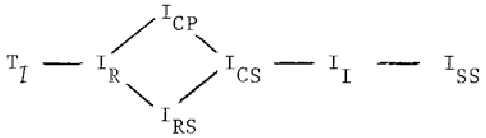
Fixpoint iteration with ∇ and Δ:



sp [10] + lfp [11] + inverse, dual, & conjugate [12]:



Lattice of Abstract Interpretations:

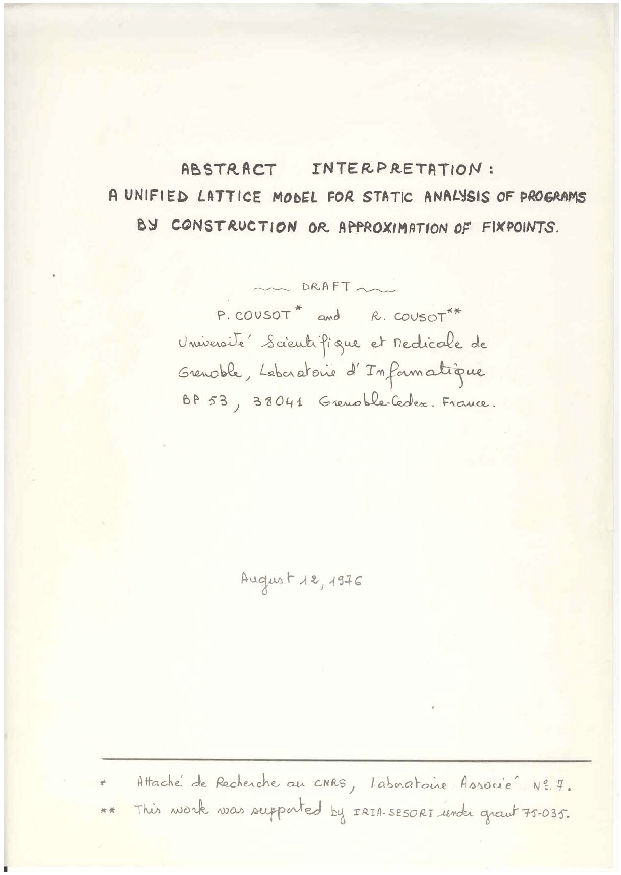


10 strongest postcondition

11 least fixpoint

12 forward $\leftrightarrow$ backward, lfp $\leftrightarrow$ gfp, & $f \leftrightarrow \neg f \neg$

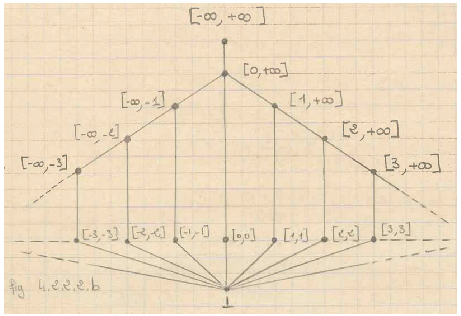
How to get your abstract accepted

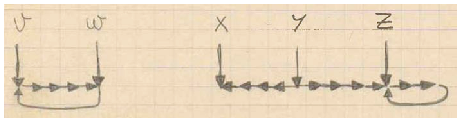


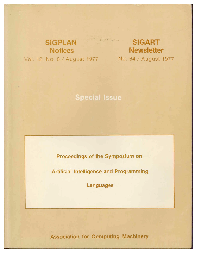
ABSTRACT INTERPRETATION :
A UNIFIED LATTICE MODEL FOR STATIC ANALYSIS OF PROGRAMS
BY CONSTRUCTION OR APPROXIMATION OF FIXPOINTS.

~~~ DRAFT ~~~

P. COUSOT * and R. COUSOT **
Université Scientifique et Medicale de
Grenoble, Laboratoire d'Informatique
BP 53, 38041 Grenoble Cedex. France.

August 18, 1976

* Attaché de Recherche au CNRS, Laboratoire Associé N°. 7.
** This work was supported by IRIA-SESORI under grant 75-035.

- Send 100 pages (instead of a 10 pages abstract)
- Handwritten, in wrong format (A4)
- At the last minute [13]
- Hoping that
  - the chairman will erase all names and references in abstracts
  - all program committee members will read the abstracts
  - and at least one will be enthousiastic [14]

[13] A storm drown down the car and we arrived just 5 mn before the post office closing time!

[14] During the conference in LA, Zohar Manna took us aside and advised us never to do that again!
~~~

Left out of the 100 pages





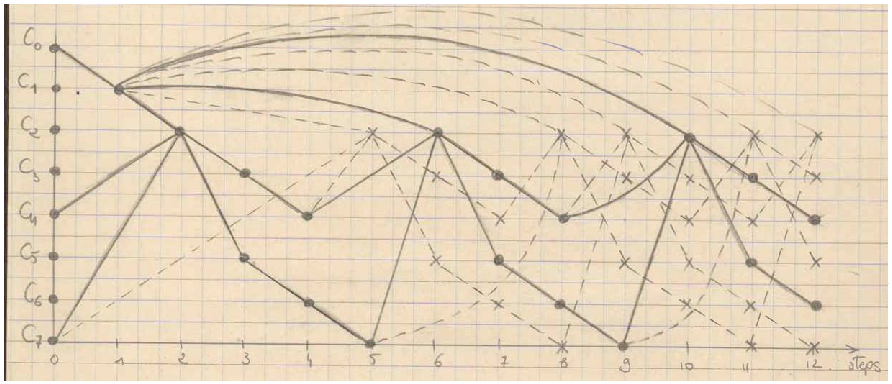


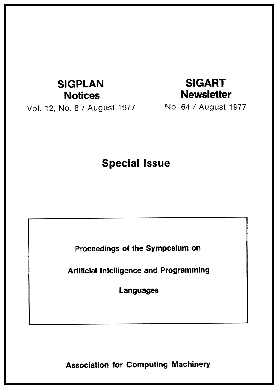
- Trace and denotational semantics (see [Cou02], in POPL'77 the analyzes were proved correct [15] with respect to a reachability semantics, called *static semantics* [16])
- Symbolic execution, see [CC77b]
- Types (abstracting sets of values, see [CC77c])
- Performance analysis
- Some examples like live expressions, extended constant propagation, heap reachability analysis [CC77c], etc.

[15] with the exception of available expression using symbolic traces

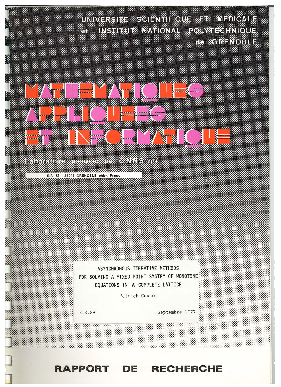
[16] collecting semantics

Chaotic and asynchronous iterations

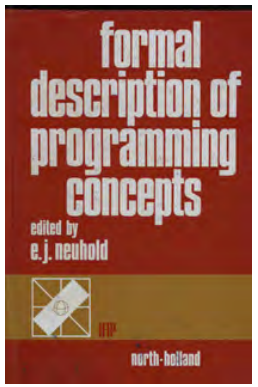




[CC77b]



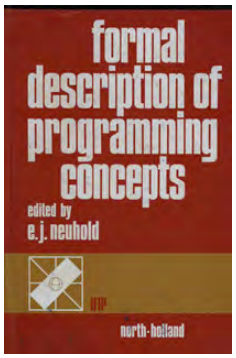
[Cou77]

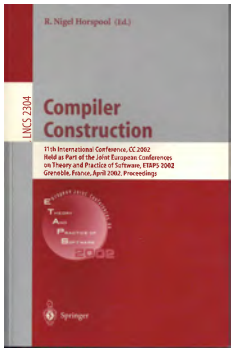


[CC77d]

– Parallel monotonic static analyses always produce the same result
– Still a problem in case of non-monotonicity (widening)

Recursive procedures



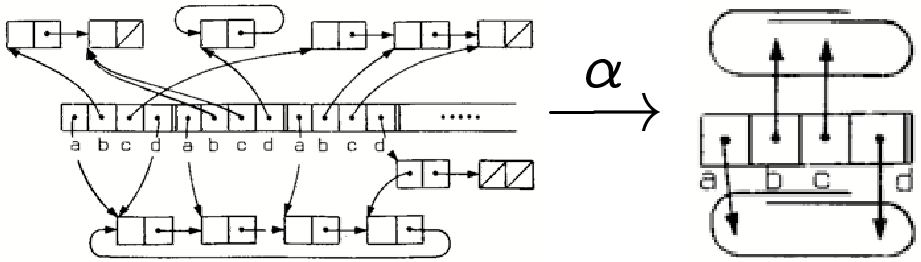


– [CC77d] introduces a topological view of abstract interpretation [17], later wrapped into *closure operators* [CC79]
– Applied to context-sensitive and separate analysis of recursive procedures (using functional chaotic iterations)
– Later extended at higher-order [CC94]
– Must still scale up for precise analyzes [CC02a]

[17] Discouraging most potential readers, later abandonned

Example: heap pointer reachability [CC77d]

Stack/heap abstraction:



Example (reversed copy):

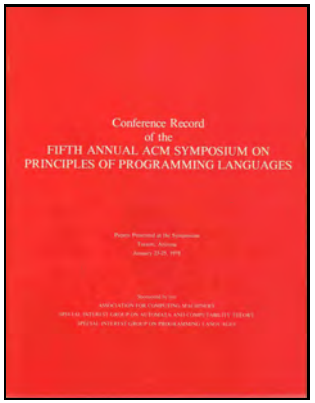
```
type node = record val∈ℤ ; next∈↑node end;
proc rev(value x,y∈↑node ; result z∈↑node) =
     if x=nil then
        z:=y;
     else
        begin new t∈↑node := allocate(node);
           t.val:=x.val;
           t.next:=y;
           rev(x.next,t;z);
        end;
     fi;
```

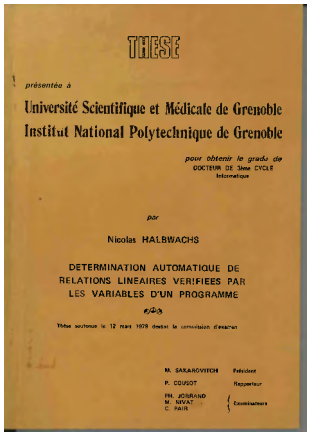
ϕ_1 ϕ_2 ϕ_3

$\phi_1(/a,x/b,y/z/) = (/a/x/b,t,y,z/)$

proving that whenever a and b reference disjoint collections then rev(a,b;c) may cause b and c to indirectly reference the same record whereas a shares no record with b or c.

Relational numerical static analysis



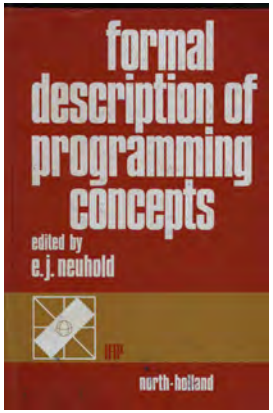


- Polyhedral/inequality analysis [CH78], [Hal79] [18] generalizing linear equalities [Kar76]. Polyhedral analysis must still scale up!
- At the origin of many numerical relational analyzes

```
procédure Hanoï (entier valeur n ; entier valeur-résultat a,b,c ;
                 entier tableau [*] TA,TB, TC) =
  {n=n0, a=a0, b=b0, c=c0}
  si n=1 alors
    b:=b+1 ; TB[b]:=TA[a]; a:=a-1
    {n=n0-1, a=a0-1, b=b0+1, c=c0}
  sinon
    début entier m ; m:=n-1 ;
      {n=n0-m+1, a=a0, b=b0, c=c0}
      Hanoï (m,a,c,b,TA,TC,TB) ;
      {n=n0-m+1, m≥1, a=a0-m, b=b0, c=c0+m}
      b:=b+1 ; TB[b]:=TA[a]; a:=a-1 ;
      {n=n0-m+1, m≥1, a=a0-m-1, b=b0+1, c=c0+m}
      Hanoï (m,c,b,a,TC,TB,TA) ;
      {n=n0-m+1, m≥1, a=a0-m-1, b=b0+m+1, c=c0}
    fin ;
    {n=n0, n0≥2, a=a0-n0, b=b0+n0, c=c0}
  finsi ;
  {n=n0, n0≥1, a=a0-n0, b=b0+n0, c=c0}
finproc ;
```

[18] also a.o. [ACH+95], [BRZH02], [BHRZ03], [BHZ05], [BHMZ05], [BHRZ05], [BHZ06], [SV95], [Hal93], [Hal94], [Hal98], [HMCV03], [HPR94], [HPR97], [Jea03], [Jea], [JHR99], [JM], [Kar76], [MM92], [SFM07], [Soh94], [VG90], [VDS91], ...

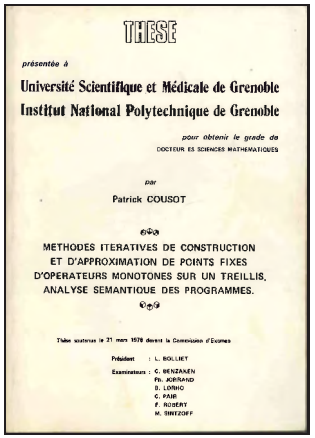
Relational symbolic static analysis



- Non-relational heap analyzes such as nullity [CC75, CC76]
- Relational heap analyzes such as heap-reachability [CC77c, CC77d], the ancestor of *storeless* analysis methods [19]
- Later extended to shape analysis using formal languages [CC95]
- Despite an abundant literature, must still scale up

[19] Initially designed in complete ignorance of Neil Jones' *store-based* methods!

Mathematical foundations of abstract interpretation [Cou78, CC79]


THESE

présentée à

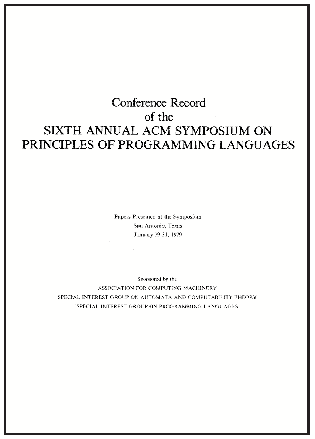
Université Scientifique et Médicale de Grenoble
Institut National Polytechnique de Grenoble

pour obtenir le grade de
DOCTEUR ES SCIENCES MATHEMATIQUES

par

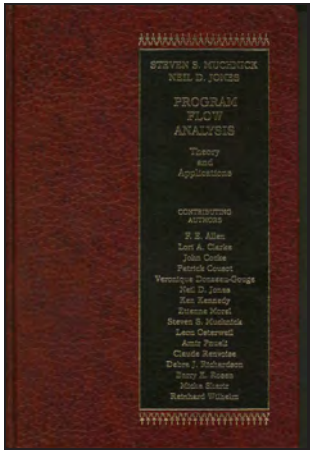
Patrick COUSOT

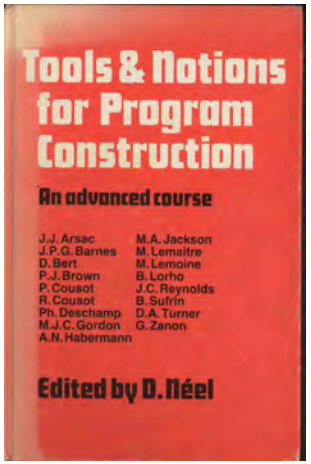
METHODES ITERATIVES DE CONSTRUCTION
ET D'APPROXIMATION DE POINTS FIXES
D'OPERATEURS MONOTONES SUR UN TREILLIS,
ANALYSE SEMANTIQUE DES PROGRAMMES.


Conference Record
of the
SIXTH ANNUAL ACM SYMPOSIUM ON
PRINCIPLES OF PROGRAMMING LANGUAGES

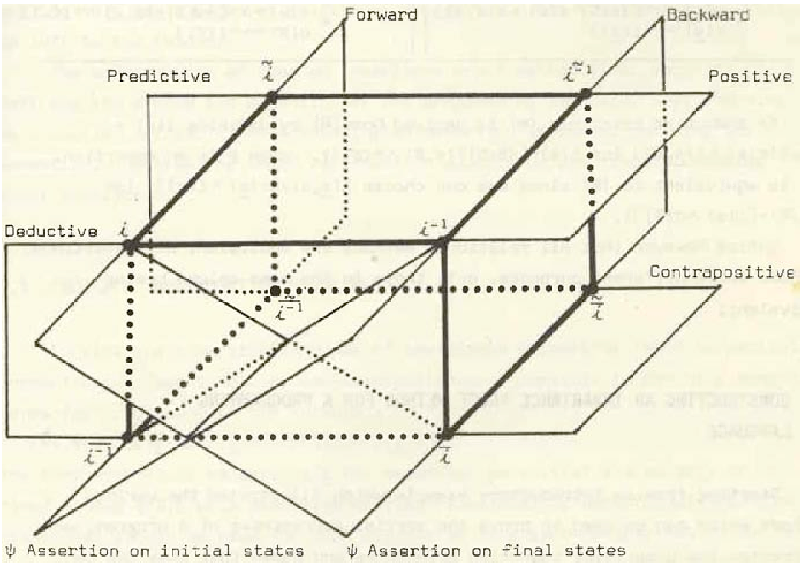
- Consider transition systems and their relational/reachability semantics in fixpoint form
- Equivalent formulations of abstraction (closures, Moore families, Galois connections,...), best abstraction
- Static analysis reduced to fixpoint abstraction and approximation (widening/narrowing and duals)
- Formal derivation of abstract semantics, abstract domains, hierarchy of abstractions
- Soundness, completeness, sufficient conditions
- Combinations of abstract domains (reduced product, etc), forward/backward analyzes

Proof methods [Cou81, CC82]





– Understanding proof methods as complete abstractions of an operational semantics of programs
– Reduction of proof methods to (equivalent) induction principles by abstraction
– Formal design of proof methods from the operational semantics, the abstraction, and the induction principle [20]



[19] Leads to the idea of computational design of abstract interpretors