Abstract-Interpretation-based Static Analysis of Safety-Critical Embedded Software

Patrick Cousot

Impromptu Invited Talk (in last minute replacement of Allen Emerson) at VMCAI'2009

Savannah, GA, USA

1. Motivation: bugs are everywhere

The factorial program (`fact.c`)

```
#include <stdio.h>
int fact (int n ) {
  int r, i;
  r = 1;
  for (i=2; i<=n; i++) {
    r = r*i;
  }
  return r;
}
int main() { int n;
  scanf("%d",&n);
  printf("%d!=%d\n",n,fact(n));
}
```

$\leftarrow$ $\mathtt{fact}(n) = 2 \times 3 \times \cdots \times n$

$\leftarrow$ read n (typed on keyboard)

$\leftarrow$ write $n\,! = \mathtt{fact}(n)$

Execution of the factorial program (fact.c)

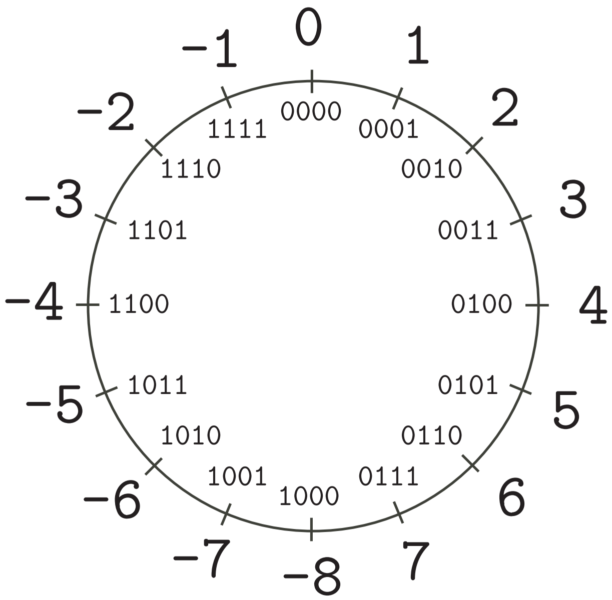
```
#include <stdio.h>
int fact (int n ) {
  int r, i;
  r = 1;
  for (i=2; i<=n; i++) {
    r = r*i;
  }
  return r;
}
int main() { int n;
  scanf("%d",&n);
  printf("%d!=%d\n",n,fact(n));
}
```

```
% gcc fact.c -o fact.exec
% ./fact.exec
3
3!  = 6
% ./fact.exec
4
4!  = 24
% ./fact.exec
100
100!  = 0
% ./fact.exec
20
20!  = -2102132736
```

Bug hunt

– Computers use integer modular arithmetics on n bits (where $n = 16, 32, 64$, etc)

– Example of an integer representation on 4 bits (in *complement to two*) :

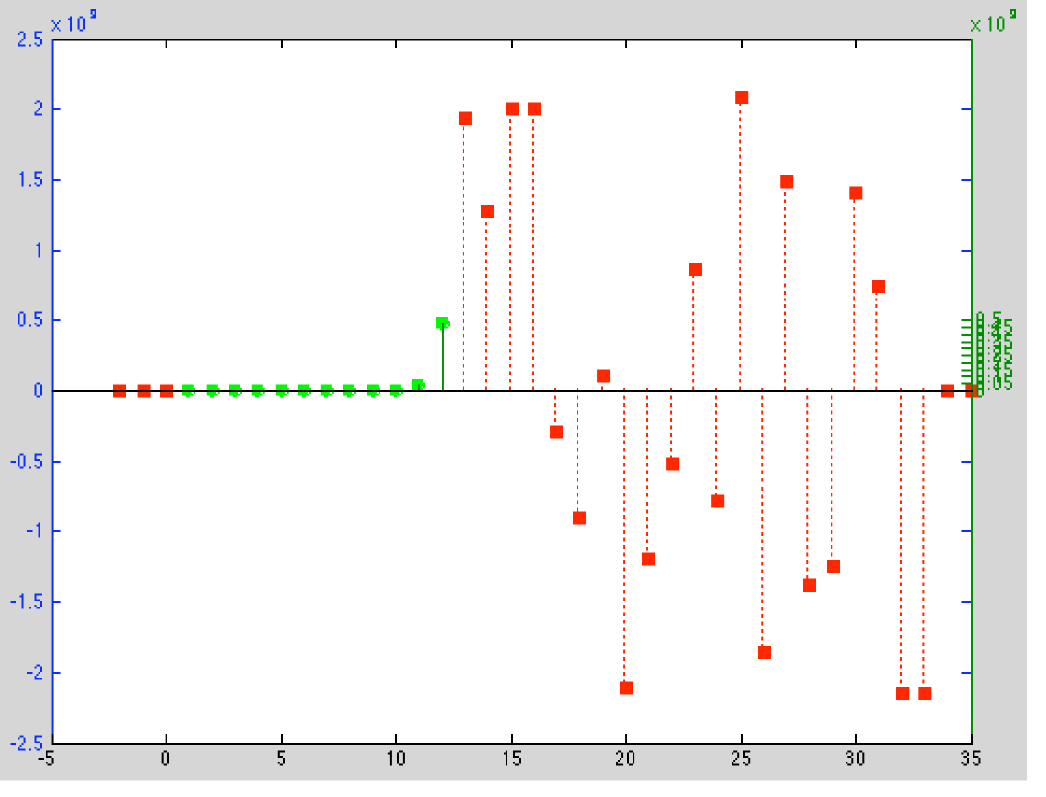


– Only integers between -8 and 7 can be represented on 4 bits

– We get $7 + 2 = -7$
$7 + 9 = 0$

The bug is a failure of the programmer

In the computer, the function `fact(n)` coincide with $n! = 2 \times 3 \times \ldots. \times n$ on the integers only for $1 \leqslant n \leqslant 12$:



And in OCaml the result is different!

```
let rec fact n = if (n = 1) then 1 else n * fact(n-1);;
```

fact(n)	C	OCaml
fact(1)	1	1
...	...	...
fact(12)	479001600	479001600
fact(13)	1932053504	−215430144
fact(14)	1278945280	−868538368
fact(15)	2004310016	−143173632
fact(16)	2004189184	−143294464
fact(17)	−288522240	−288522240
fact(18)	−898433024	−898433024
fact(19)	109641728	109641728
fact(20)	−2102132736	45350912
fact(21)	−1195114496	952369152

fact(22)	−522715136	−522715136
fact(23)	862453760	862453760
fact(24)	−775946240	−775946240
fact(25)	2076180480	−71303168
fact(26)	−1853882368	293601280
fact(27)	1484783616	−662700032
fact(28)	−1375731712	771751936
fact(29)	−1241513984	905969664
fact(30)	1409286144	−738197504
fact(31)	738197504	738197504
fact(32)	−2147483648	0
fact(33)	−2147483648	0
fact(34)	0	0

Absence of runtime error

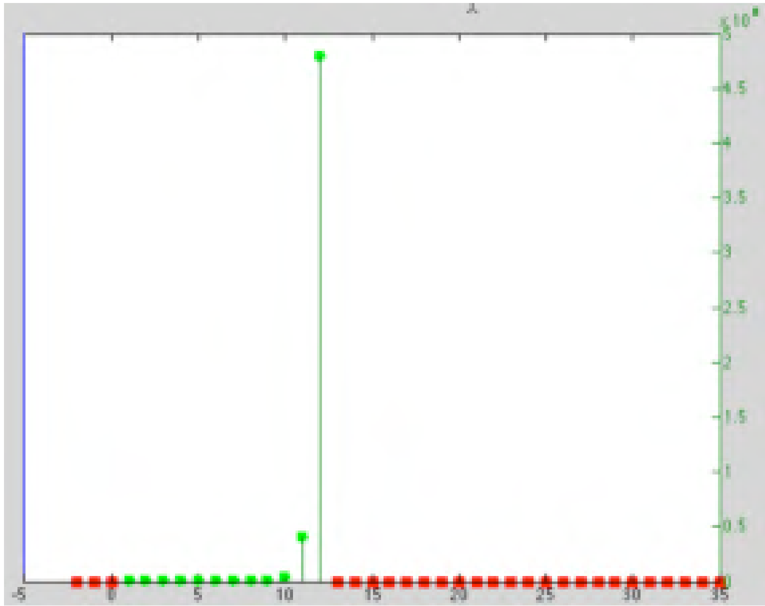
```
int fact (int n ) {
  int r, i;
  r = 1;
  for (i=2; i<=n; i++) {    ← no overflow of i++
    r = r*i;                ← no overflow of r*i
  }
  return r;
}
```

Proof of absence of runtime error by static analysis

```
% cat -n fact_lim.c
int MAXINT = 2147483647;
int fact (int n) {
   int r, i;
   if (n < 1) || (n = MAXINT) {
       r = 0;
   } else {
       r = 1;
       for (i = 2; i<=n; i++) {
           if (r <= (MAXINT / i)) {
               r = r * i;
           } else {
               r = 0;
           }
       }
   }
   return r;
}

int main() {
   int n, f;
   f = fact(n);
}
% astree -exec-fn main fact_lim.c |& grep WARN
%
```

→ No alarm!



2. Varieties of Static Analyses

Static Analysis

– In general static analysis means " *the fully automatic* verification of properties of program executions using the program text only" (excluding running programs)

– But for trivial cases, it is undecidable

– Alternatives to impossible total verification:

  - under-verification (testing, bounded model-checking, bug pattern mining, etc): bug finding, misses bugs, never ends

  - over-verification (typing, dataflow analysis,etc): no bug missed but false alarms

– Challenge: total verification for a given category of properties and a given family of programs (no bug missed, no false alarm but not for all possible properties of all programs)

3. Abstract Interpretation

Programs

$$
\begin{array}{lcll}
\ell & \in & \mathbb{L}, & \text{labels} \\
\mathsf{x} & \in & \mathbb{V}, & \text{variables} \\
\mathsf{E} & \in & \mathbb{E}, & \text{expressions} \\
\mathsf{B} & \in & \mathbb{B}, & \text{conditions} \\
\mathsf{C} & \in & \mathbb{C}, & \text{commands} \\
\mathsf{C} & ::= & {}^{\ell}\texttt{skip} & \\
 & | & {}^{\ell}\texttt{X := }\mathsf{E} & \\
 & | & \texttt{if }{}^{\ell}\mathsf{B}\texttt{ then }\mathsf{C}_1\texttt{ else }\mathsf{C}_2\texttt{ fi} & \\
 & | & \mathsf{C}_1\ ;\ \mathsf{C}_2 & \\
 & | & \texttt{while }{}^{\ell}\mathsf{B}\texttt{ do }\mathsf{C}_1\texttt{ od} & \\
\mathsf{P} & ::= & \mathsf{C}^{\ell}\,. & \text{programs}
\end{array}
$$

```
¹x := ? ;
while ²(1 < x) do
  ³x := x - 1
od⁴.
```

Initial label

$\mathbf{i}[\![\mathsf{C}]\!] \in \mathbb{L}$: initial label where execution of command C starts

$$
\begin{aligned}
\mathbf{i}[\![{}^{\ell}\mathtt{skip}]\!] &\triangleq \ell \\
\mathbf{i}[\![{}^{\ell}\mathtt{X} := \mathsf{E}]\!] &\triangleq \ell \\
\mathbf{i}[\![\mathtt{if}\ {}^{\ell}\mathsf{B}\ \mathtt{then}\ \mathsf{C}_1\ \mathtt{else}\ \mathsf{C}_2\ \mathtt{fi}]\!] &\triangleq \ell \\
\mathbf{i}[\![\mathsf{C}_1\ ;\ \mathsf{C}_2]\!] &\triangleq \mathbf{i}[\![\mathsf{C}_1]\!] \\
\mathbf{i}[\![\mathtt{while}\ {}^{\ell}\mathsf{B}\ \mathtt{do}\ \mathsf{C}_1\ \mathtt{od}]\!] &\triangleq \ell \\
\mathbf{i}[\![\mathsf{C}^{\ell}.]\!] &\triangleq \mathbf{i}[\![\mathsf{C}]\!]
\end{aligned}
$$

Final label

$\mathsf{f}[\![\mathsf{C}]\!] \in \mathbb{L}$: final label where execution of command C finishes

$$
\begin{array}{lll}
\mathsf{P} ::= \mathsf{C}_1{}^{\ell}. & & \mathsf{f}[\![\mathsf{P}]\!] \triangleq \ell \\
 & & \mathsf{f}[\![\mathsf{C}_1]\!] \triangleq \mathsf{f}[\![\mathsf{P}]\!] \\
\mathsf{C} ::= {}^{\ell}\mathtt{skip} & & \mathsf{f}[\![{}^{\ell}\mathtt{skip}]\!] \triangleq \mathsf{f}[\![\mathsf{C}]\!] \\
\quad \mid \ {}^{\ell}\mathtt{X} := \mathsf{E} & & \mathsf{f}[\![{}^{\ell}\mathtt{X} := \mathsf{E}]\!] \triangleq \mathsf{f}[\![\mathsf{C}]\!] \\
\quad \mid \ \mathtt{if}\ {}^{\ell}\mathsf{B}\ \mathtt{then}\ \mathsf{C}_1 & & \mathsf{f}[\![\mathtt{if}\ {}^{\ell}\mathsf{B}\ \mathtt{then}\ \mathsf{C}_1\ \mathtt{else}\ \mathsf{C}_2\ \mathtt{fi}]\!] \triangleq \mathsf{f}[\![\mathsf{C}]\!] \\
\qquad \mathtt{else}\ \mathsf{C}_2\ \mathtt{fi} & & \mathsf{f}[\![\mathsf{C}_1]\!] \triangleq \mathsf{f}[\![\mathsf{C}]\!] \\
 & & \mathsf{f}[\![\mathsf{C}_2]\!] \triangleq \mathsf{f}[\![\mathsf{C}]\!] \\
\quad \mid \ \mathsf{C}_1\ ;\ \mathsf{C}_2 & & \mathsf{f}[\![\mathsf{C}_1\ ;\ \mathsf{C}_2]\!] \triangleq \mathsf{f}[\![\mathsf{C}]\!] \\
 & & \mathsf{f}[\![\mathsf{C}_1]\!] \triangleq \mathsf{i}[\![\mathsf{C}_2]\!] \\
 & & \mathsf{f}[\![\mathsf{C}_2]\!] \triangleq \mathsf{f}[\![\mathsf{C}]\!] \\
\quad \mid \ \mathtt{while}\ {}^{\ell}\mathsf{B}\ \mathtt{do}\ \mathsf{C}_1\ \mathtt{od} & & \mathsf{f}[\![\mathtt{while}\ {}^{\ell}\mathsf{B}\ \mathtt{do}\ \mathsf{C}_1\ \mathtt{od}]\!] \triangleq \mathsf{f}[\![\mathsf{C}]\!] \\
 & & \mathsf{f}[\![\mathsf{C}_1]\!] \triangleq \ell
\end{array}
$$

Semantics: set of observations of program executions

states:

$\nu \in \mathcal{V}$, values (of variables $\mathtt{x} \in \mathbb{V}$)

$\rho \in \mathcal{E}$, environments

$\mathcal{E} \triangleq \mathbb{V} \mapsto \mathcal{V}$

$\sigma \in \mathcal{S}$, states

$\mathcal{S} \triangleq \mathbb{L} \times \mathcal{E}$

traces:

$$\mathcal{P}^n \triangleq \{\pi_0 \ldots \pi_{n-1} \mid \forall i \in [0, n-1] : \pi_i \in \mathcal{S}\} \quad \text{traces of length } n \geqslant 1$$

$$\mathcal{P} \triangleq \bigcup_{n \geqslant 1} \mathcal{P}^n \quad \text{finite traces}$$

semantics: set of traces formalizing finite observations of the program execution (from initial states).

Example: execution trace of `fact(4)`

```
int fact (int n ) {
  int r = 1, i;
  for (i=2; i<=n; i++) {
    r = r*i;
  }
  return r;
}
```

- $\texttt{n} \leftarrow 4;\ \texttt{r} \leftarrow 1;$
- $\texttt{i} \leftarrow 2;\ \texttt{r} \leftarrow 1 \times 2 = 1;$
- $\texttt{i} \leftarrow 3;\ \texttt{r} \leftarrow 2 \times 3 = 6;$
- $\texttt{i} \leftarrow 4;\ \texttt{r} \leftarrow 6 \times 4 = 24;$
- $\texttt{i} \leftarrow 5;$
- $\texttt{return}\ 24;$

Structural semantics

skip:

$$\mathbf{S}[\![^{\ell}\texttt{skip}]\!] \triangleq \{\langle \ell,\ \rho\rangle \mid \rho \in \mathcal{E}\} \\ \cup \{\langle \ell,\ \rho\rangle\langle \mathsf{f}[\![^{\ell}\texttt{skip}]\!],\ \rho\rangle \mid \rho \in \mathcal{E}\}$$

assignment:

$$\mathbf{S}[\![^{\ell}\texttt{X} := \mathsf{E}]\!] \triangleq \{\langle \ell,\ \rho\rangle \mid \rho \in \mathcal{E}\} \\ \cup \{\langle \ell,\ \rho\rangle\langle \mathsf{f}[\![^{\ell}\texttt{X} := \mathsf{E}]\!],\ \rho[\texttt{X} \leftarrow \mathbf{E}[\![\mathsf{E}]\!]\rho]\rangle \mid \rho \in \mathcal{E}\}$$

Structural semantics (cont'd)

conditional:

$$\begin{aligned}&\mathbf{S}[\![\texttt{if}\ {}^{\ell}\mathsf{B}\ \texttt{then}\ \mathsf{C}_1\ \texttt{else}\ \mathsf{C}_2\ \texttt{fi}]\!] \triangleq\\ &\quad \{\langle \ell,\ \rho\rangle \mid \rho \in \mathcal{E}\}\\ &\cup \{\langle \ell,\ \rho\rangle\langle \mathsf{i}[\![\mathsf{C}_1]\!],\ \rho\rangle \mid \mathit{tt} \in \mathbf{B}[\![\mathsf{B}]\!]\rho\} \circ \mathbf{S}[\![\mathsf{C}_1]\!]\\ &\cup \{\langle \ell,\ \rho\rangle\langle \mathsf{i}[\![\mathsf{C}_2]\!],\ \rho\rangle \mid \mathit{ff} \in \mathbf{B}[\![\mathsf{B}]\!]\rho\} \circ \mathbf{S}[\![\mathsf{C}_2]\!]\end{aligned}$$

where $X \circ Y \triangleq \{\pi\sigma\pi' \mid \pi\sigma \in X \wedge \sigma\pi' \in Y\}$

sequence:

$$\mathbf{S}[\![\mathsf{C}_1\ ;\ \mathsf{C}_2]\!] \triangleq \mathbf{S}[\![\mathsf{C}_1]\!] \cup (\mathbf{S}[\![\mathsf{C}_1]\!] \circ \mathbf{S}[\![\mathsf{C}_2]\!])$$

Structural semantics (cont'd)

iteration:

$$\mathbf{S}[\![\texttt{while}\ {}^{\ell}\mathsf{B}\ \texttt{do}\ \mathsf{C}\ \texttt{od}]\!] \triangleq \mathsf{lfp}^{\subseteq}\ \mathbf{F}[\![\texttt{while}\ {}^{\ell}\mathsf{B}\ \texttt{do}\ \mathsf{C}\ \texttt{od}]\!]$$

where the transformer $\mathbf{F}[\![\texttt{while}\ {}^{\ell}\mathsf{B}\ \texttt{do}\ \mathsf{C}\ \texttt{od}]\!]$ is

$$\begin{array}{l}\mathbf{F}[\![\texttt{while}\ {}^{\ell}\mathsf{B}\ \texttt{do}\ \mathsf{C}\ \texttt{od}]\!](X) \triangleq \\ \quad \{\langle \ell,\ \rho\rangle \mid \rho \in \mathcal{E}\} \\ \cup\ X \circ \{\langle \ell,\ \rho\rangle\langle \mathsf{i}[\![\mathsf{C}]\!],\ \rho\rangle \mid \mathit{tt} \in \mathbf{B}[\![\mathsf{B}]\!]\rho\} \circ \mathbf{S}[\![\mathsf{C}]\!] \\ \cup\ X \circ \{\langle \ell,\ \rho\rangle\langle \mathsf{f}[\![\texttt{while}\ {}^{\ell}\mathsf{B}\ \texttt{do}\ \mathsf{C}\ \texttt{od}]\!],\ \rho\rangle \mid \mathit{ff} \in \mathbf{B}[\![\mathsf{B}]\!]\rho\}\end{array}$$

($\mathbf{F}[\![\texttt{while}\ {}^{\ell}\mathsf{B}\ \texttt{do}\ \mathsf{C}\ \texttt{od}]\!]$ is $\subseteq$-monotone on a complete lattice $\langle \wp(\mathcal{P}),\ \subseteq,\ \emptyset,\ \mathcal{P},\ \cup,\ \cap\rangle$ hence does exist.)

Properties/specifications: set of possible semantics

properties/specifications:
Program properties are sets of possible semantics of the program.

$$\begin{array}{rll} \mathcal{P} & & \text{traces} \\ \mathbf{S}[\![\mathbf{C}]\!] \in \wp(\mathcal{P}) & & \text{semantics (set of traces)} \\ \wp(\wp(\mathcal{P})) & & \text{program properties (set of sets of traces)} \end{array}$$

strongest program property of a command **C**

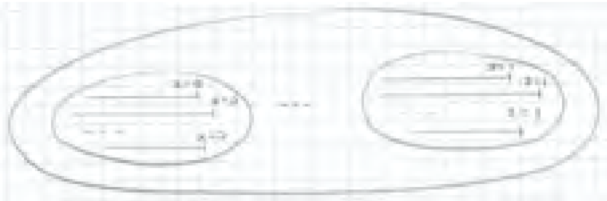
$$\{\mathbf{S}[\![\mathbf{C}]\!]\} \in \wp(\wp(\mathcal{P}))$$

structure of properties:

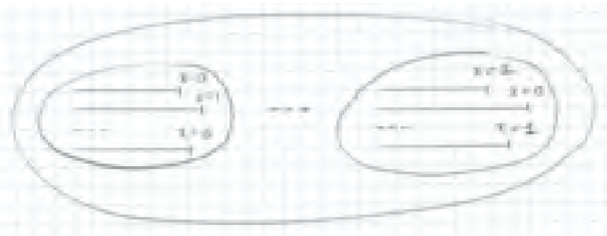
$$\langle \wp(\wp(\mathcal{P})),\ \subseteq,\ \emptyset,\ \mathcal{P},\ \cup,\ \cap \rangle$$

Examples of program properties

– If execution terminates then the final value of variable `x` is always 0 or is always 1 [1]. In pictures



– If execution terminates then the final value of variable `x` is always 0 or 1. In pictures



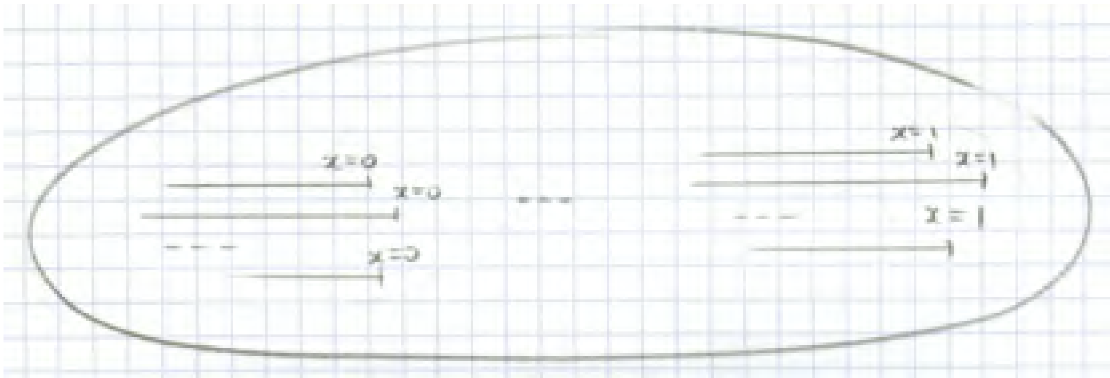
[1] Neither a safety nor liveness property.

Abstraction to traces

The *trace abstraction* of a program property to a trace property

$$\alpha^\pi \in \wp(\wp(\mathcal{P})) \mapsto \wp(\mathcal{P})$$
$$\alpha^\pi(P) \triangleq \bigcup P$$

Both example properties abstract to the same trace property [2]:

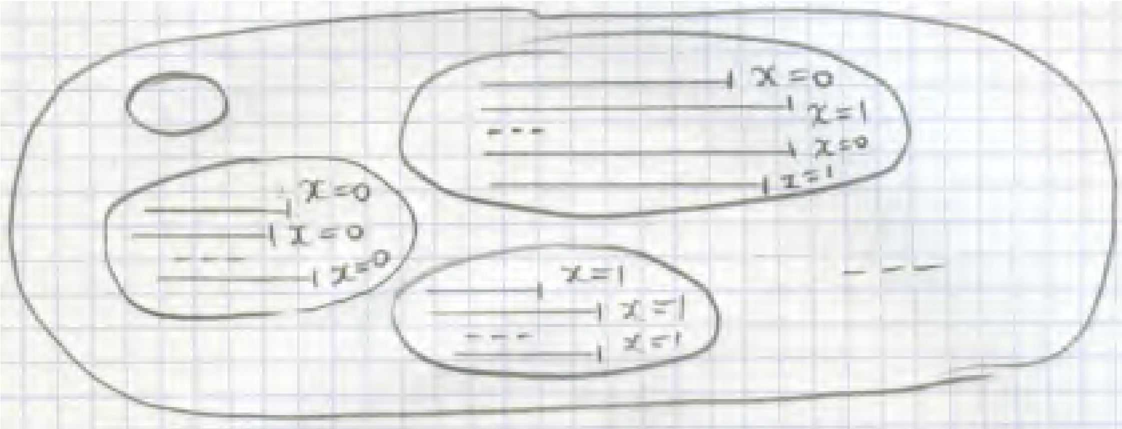


[2] indeed a safety property.

Overapproximation

Abstraction of a property always over-approximates the possible concrete property, as shown by the inverse *concretization*

$$\gamma^\pi \in \wp(\mathcal{P}) \mapsto \wp(\wp(\mathcal{P}))$$
$$\gamma^\pi(T) \triangleq \wp(T)$$



Abstraction to transitions

The *transition abstraction* abstracts sets of traces to a transition relation between a state and its possible successors in one computation step.

$$\alpha^{\tau} \in \wp(\mathcal{P}) \mapsto \wp(\mathcal{S} \times \mathcal{S})$$
$$\alpha^{\tau}(T) \triangleq \{\langle \pi_i, \pi_{i+1} \rangle \mid \exists n \geqslant 1 : \pi \in T \cap \mathcal{P}^n \wedge 0 \leqslant i < n - 1\}$$

Abstraction to initial/current state relation

The *relational abstraction* (or *initial/current state abstraction*) of a trace property just records the first and last state of traces.

$$\alpha^R \in \wp(\mathcal{P}) \mapsto \wp(\mathcal{S} \times \mathcal{S})$$
$$\alpha^R(T) \triangleq \{\langle \pi_0,\ \pi_{n-1} \rangle \mid \exists n \geqslant 1 : \pi \in T \cap \mathcal{P}^n\}$$

$$\gamma^R \in \wp(\mathcal{S} \times \mathcal{S}) \mapsto \wp(\mathcal{P})$$
$$\gamma^R(R) \triangleq \bigcup_{n \geqslant 1} \{\pi \in \mathcal{P}^n \mid \langle \pi_0,\ \pi_{n-1} \rangle \in R\}$$

Abstraction to reachable states

The *reachability abstraction* of a trace property just records the last state of traces.

$$\begin{aligned}\alpha^r &\in \wp(\mathcal{P}) \mapsto \wp(\mathcal{S})\\ \alpha^r(T) &\triangleq \{\pi_{n-1} \mid \exists n \geqslant 1 : \pi \in T \cap \mathcal{P}^n\}\end{aligned}$$

$$\begin{aligned}\gamma^r &\in \wp(\mathcal{S}) \mapsto \wp(\mathcal{P})\\ \gamma^r(I) &\triangleq \bigcup_{n \geqslant 1} \{\pi \in \mathcal{P}^n \mid \pi_{n-1} \in I\}\end{aligned}$$

Abstraction to local invariants

The *invariance abstraction* maps the reachable states to local invariants on memory states attached to program points ($\mathcal{S} \triangleq \mathbb{L} \times \mathcal{E}$ so $\wp(\mathcal{S})$ is isomorphic to $\mathbb{L} \mapsto \wp(\mathcal{E})$)

$$\alpha^I \in \wp(\mathcal{S}) \mapsto (\mathbb{L} \mapsto \wp(\mathcal{E}))$$
$$\alpha^I(R)\ell \triangleq \{\rho \in \mathcal{E} \mid \langle \ell, \rho \rangle \in R\}$$

Predicate abstraction

Given a *finite* set $\mathcal{P}$ of predicates $P \in \mathcal{P}$ (with interpretation $\mathbf{I} \in \mathcal{E} \mapsto \mathbb{B}$, $\mathbb{B} \triangleq \{\mathrm{tt}, \mathrm{ff}\}$), the *predicate abstraction* records at each program point the conjunction of predicates of $\mathcal{P}$ which are invariant at that point.

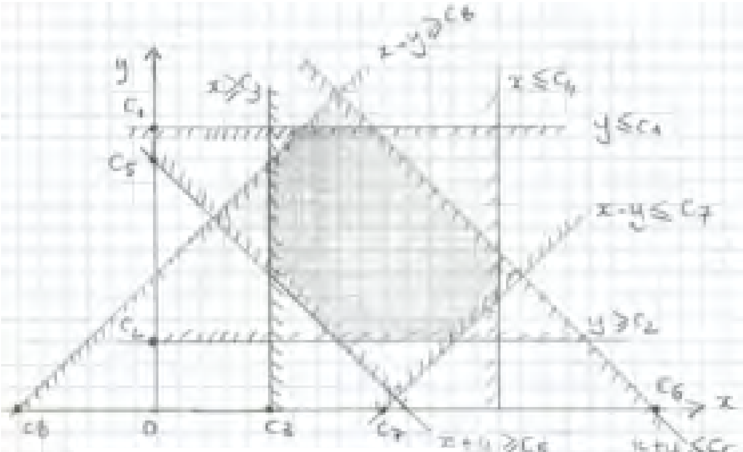
$$\begin{aligned} \alpha^{\mathcal{P}} &\in (\mathbb{L} \mapsto \wp(\mathcal{E})) \mapsto (\mathbb{L} \mapsto \wp(\mathcal{E})) \\ \alpha^{\mathcal{P}}[I](\ell) &\triangleq \bigcap_{P \in \mathcal{P}} \{\rho \mid \mathbf{I}[\![P]\!]\rho \wedge I(\ell) \subseteq \{\rho' \mid \mathbf{I}[\![P]\!]\rho'\}\} \end{aligned}$$

Octagon abstraction

The *octagon abstraction* just records inequalities of the form $c \leq x \pm y \leq c'$ and $c \leq x \leq c'$ between pairs x and y of values of variables x and y.

$$\alpha^o \in (\mathbb{L} \mapsto \wp(\mathcal{E})) \mapsto (\mathbb{L} \mapsto \wp(\mathcal{E})), \qquad \mathcal{E} \triangleq \mathbb{V} \mapsto \mathcal{V}$$

$$\begin{aligned}\alpha^o[I](\ell) \triangleq \bigcap_{\mathrm{x},\mathrm{y}\in\mathbb{V}} & \{\rho \mid \max\{c \mid \{\rho' \mid c \leqslant \rho'(\mathrm{x}) \pm \rho'(\mathrm{y})\} \subseteq I(\ell)\} \\ & \leqslant \rho(\mathrm{x}) \pm \rho(\mathrm{y}) \leqslant \\ & \min\{c' \mid \{\rho' \mid \rho'(\mathrm{x}) \pm \rho'(\mathrm{y}) \leqslant c'\} \subseteq I(\ell)\}\} \\ \cap & \{\rho \mid \max\{c \mid \{\rho' \mid c \leqslant \rho'(\mathrm{x})\} \subseteq I(\ell)\} \\ & \leqslant \rho(\mathrm{x}) \leqslant \\ & \min\{c' \mid \{\rho' \mid \rho'(\mathrm{x}) \leqslant c'\} \subseteq I(\ell)\}\}\end{aligned}$$



The coefficients have to be determined by the analysis among infinitely many possibilities

Cartesian abstraction

The *cartesian abstraction* just records the values of variables, ignoring the relationships betwen the values of such variables.

$$\mathcal{E} \triangleq \mathbb{V} \mapsto \mathcal{V}$$

$$\alpha^c \in (\mathbb{L} \mapsto \wp(\mathcal{E})) \mapsto (\mathbb{L} \mapsto (\mathbb{V} \mapsto \wp(\mathcal{V})))$$
$$\alpha^c[I](\ell)\mathrm{x} \triangleq \{\rho(\mathrm{x}) \mid \rho \in I(\ell)\}$$

Interval abstraction

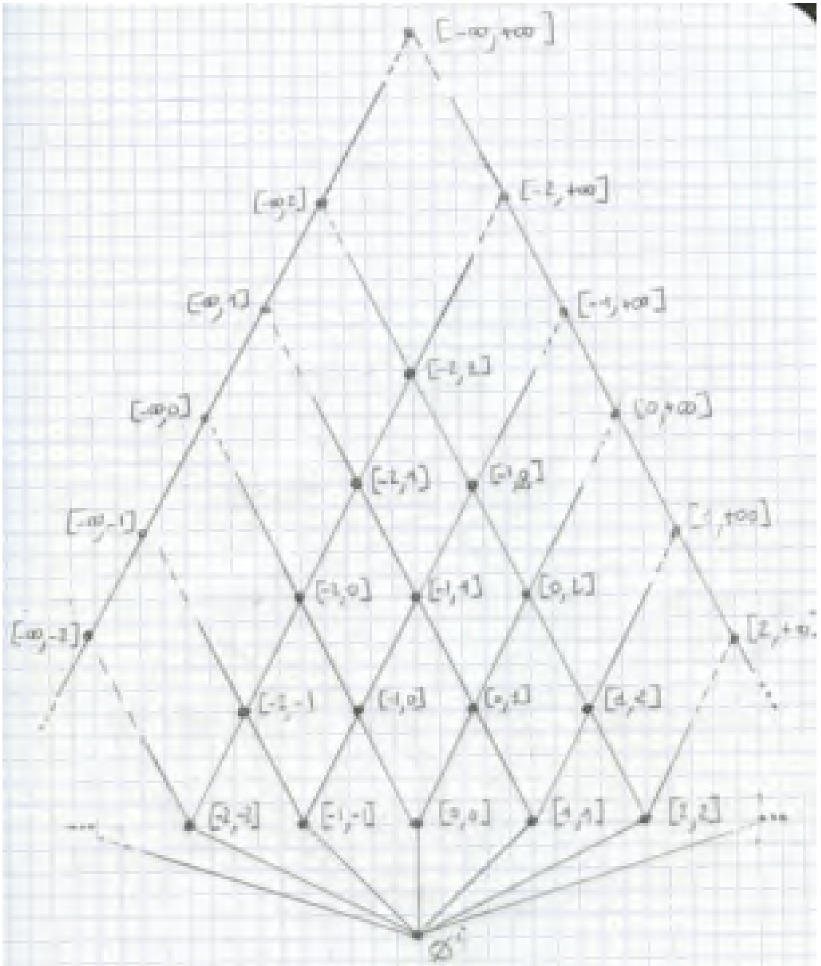
The *interval abstraction* just records the minimum and maximum value of numerical variables.

$$\alpha^{\imath} \in (\mathbb{L} \mapsto (\mathbb{V} \mapsto \wp(\mathbb{I}))) \mapsto (\mathbb{L} \mapsto (\mathbb{V} \mapsto (\mathbb{I}^{\infty} \times \mathbb{I}^{\infty})))$$
$$\alpha^{\imath}[C](\ell)\mathrm{x} \triangleq [\min C(\ell)(\mathrm{x}),\ \max C(\ell)(\mathrm{x})]$$

where

$$\begin{aligned} \min \mathbb{I} &\triangleq -\infty \\ \max \mathbb{I} &\triangleq \infty \\ \mathbb{I}^{\infty} &\triangleq \mathbb{I} \cup \{-\infty, \infty\} \\ [l,\ h] &= \emptyset \quad \text{whenever } h < l \end{aligned}$$

Interval lattice



Sign abstraction

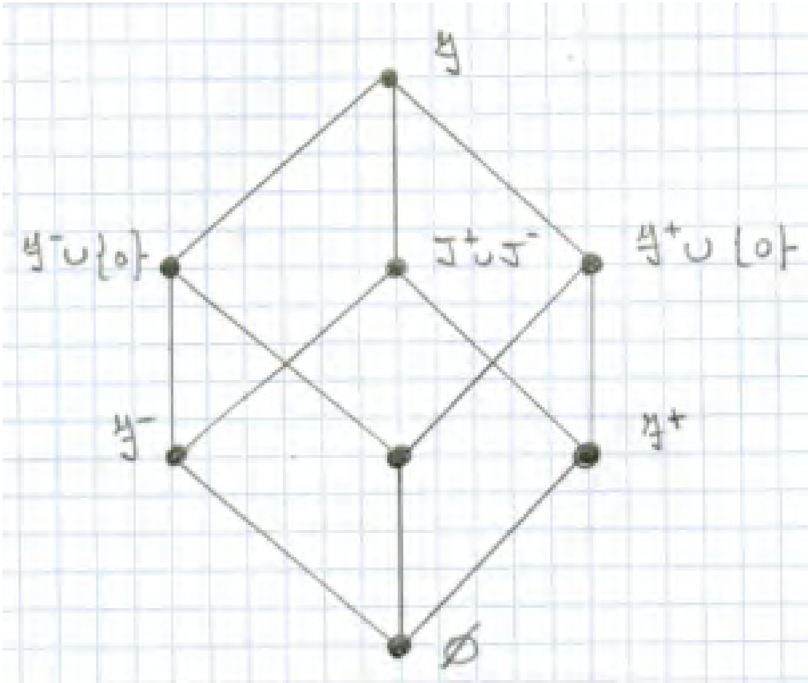
The *sign abstraction* just records the sign of variables.

$\mathbb{I}$	set of integers
$\mathbb{I}^-$	set of strictly negative integers
$\mathbb{I}^+$	set of strictly positive integers

$$\alpha^s \in (\mathbb{L} \mapsto (\mathbb{V} \mapsto \wp(\mathbb{I}))) \mapsto (\mathbb{L} \mapsto (\mathbb{V} \mapsto \bigcup_{I \subseteq \{\emptyset, \mathbb{I}^-, \{0\}, \mathbb{I}^+\}} I))$$

$$\begin{aligned}\alpha^s[C](\ell)\mathrm{x} \triangleq \bigcup & \{\mathbb{I}^- \mid C(\ell)\mathrm{x} \cap \mathbb{I}^- \neq \emptyset\} \cup \{\{0\} \mid 0 \in C(\ell)\mathrm{x}\} \cup \\ & \{\mathbb{I}^+ \mid \mathbb{I}^+ \cap C(\ell)\mathrm{x} \neq \emptyset\}\end{aligned}$$

Sign lattice



Parity abstraction

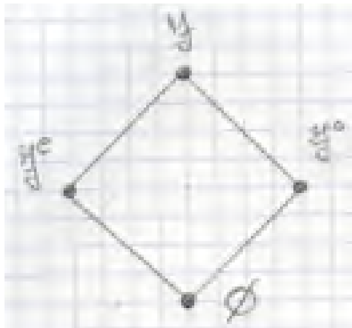
The *parity abstraction* just records the parity of variables.

$\mathbb{I}^{\mathrm{e}}$	set of even integers
$\mathbb{I}^{\mathrm{o}}$	set of odd integers

$$\alpha^p \in (\mathbb{L} \mapsto (\mathbb{V} \mapsto \wp(\mathcal{V}))) \mapsto (\mathbb{L} \mapsto (\mathbb{V} \mapsto \{\emptyset, \mathbb{I}^{\mathrm{o}}, \mathbb{I}^{\mathrm{e}}, \mathbb{I}\}))$$

$$\alpha^p[C](\ell)\mathrm{x} \triangleq \bigcup\{\mathbb{I}^{\mathrm{o}} \mid C(\ell)(x) \cap \mathbb{I}^{\mathrm{o}} \neq \emptyset\} \cup \{\mathbb{I}^{\mathrm{e}} \mid C(\ell)(x) \cap \mathbb{I}^{\mathrm{e}} \neq \emptyset\}$$

Parity lattice



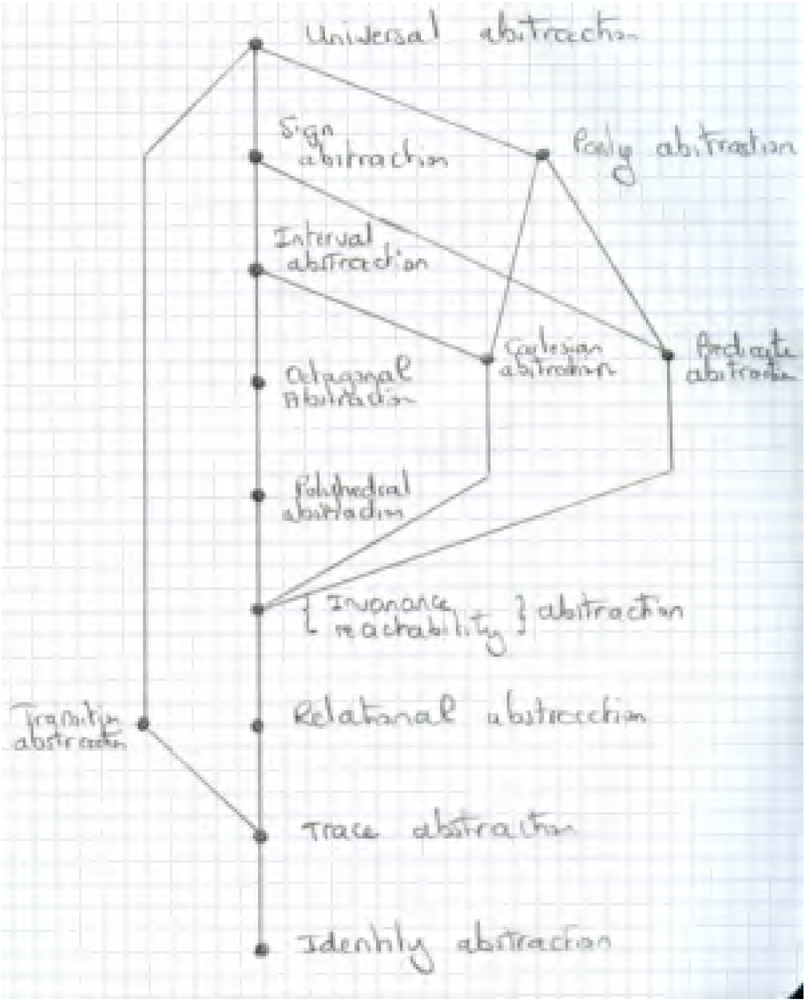
Abstraction $\langle L,\ \subseteq\rangle \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} \langle A,\ \sqsubseteq\rangle$

- The *concrete properties* form a poset $\langle L,\ \subseteq\rangle$
- The *abstract properties* form a poset $\langle A,\ \sqsubseteq\rangle$
- The *abstraction* α is monotonic (to preserve concrete implication $\subseteq$)
- The *concretization* γ is monotonic (to preserve abstract implication $\sqsubseteq$)
- The abstraction is an *overapproximation*: $P \subseteq \gamma(\alpha(P))$
- In case of existence of a *best abstraction*:

$$P \subseteq \gamma(Q) \implies \alpha(P) \sqsubseteq Q$$

we have a *Galois connection* (so α/γ uniquely determines the other).

Hierarchy of abstractions



Structural abstract semantics

abstraction (of sets of traces)

$$\langle \wp(\mathcal{P}),\ \subseteq,\ \cup\rangle \xleftarrow[\alpha]{\gamma} \langle A,\ \sqsubseteq,\ \sqcup\rangle$$

skip:

$$\begin{aligned}\mathbf{S}^\sharp[\![{}^\ell\mathtt{skip}]\!] \triangleq\ & \alpha(\{\langle \ell,\ \rho\rangle \mid \rho \in \mathcal{E}\})\\ & \sqcup\ \alpha(\{\langle \ell,\ \rho\rangle\langle \mathsf{f}[\![{}^\ell\mathtt{skip}]\!],\ \rho\rangle \mid \rho \in \mathcal{E}\})\end{aligned}$$

assignment:

$$\begin{aligned}\mathbf{S}^\sharp[\![{}^\ell\mathtt{X} := \mathsf{E}]\!] \triangleq\ & \alpha(\{\langle \ell,\ \rho\rangle \mid \rho \in \mathcal{E}\})\\ & \sqcup\ \alpha(\{\langle \ell,\ \rho\rangle\langle \mathsf{f}[\![{}^\ell\mathtt{X} := \mathsf{E}]\!],\ \rho[\mathtt{X} \leftarrow \mathbf{E}[\![\mathsf{E}]\!]\rho]\rangle \mid \rho \in \mathcal{E}\})\end{aligned}$$

Structural abstract semantics (cont'd)

conditional:

$$\begin{array}{l}\mathbf{S}^\sharp[\![\texttt{if}\ {}^\ell\mathsf{B}\ \texttt{then}\ \mathsf{C}_1\ \texttt{else}\ \mathsf{C}_2\ \texttt{fi}]\!] \triangleq \\ \quad \alpha(\{\langle \ell,\ \rho\rangle \mid \rho \in \mathcal{E}\}) \\ \sqcup\ \alpha(\{\langle \ell,\ \rho\rangle\langle \mathbf{i}[\![\mathsf{C}_1]\!],\ \rho\rangle \mid tt \in \mathbf{B}[\![\mathsf{B}]\!]\rho\}) \circ^\sharp \mathbf{S}^\sharp[\![\mathsf{C}_1]\!] \\ \sqcup\ \alpha(\{\langle \ell,\ \rho\rangle\langle \mathbf{i}[\![\mathsf{C}_2]\!],\ \rho\rangle \mid ff \in \mathbf{B}[\![\mathsf{B}]\!]\rho\}) \circ^\sharp \mathbf{S}^\sharp[\![\mathsf{C}_2]\!]\end{array}$$

where $X \circ^\sharp Y \triangleq \alpha(\gamma(X) \circ \gamma(Y))$

sequence:

$$\mathbf{S}^\sharp[\![\mathsf{C}_1\ ;\ \mathsf{C}_2]\!] \triangleq \mathbf{S}^\sharp[\![\mathsf{C}_1]\!] \sqcup (\mathbf{S}^\sharp[\![\mathsf{C}_1]\!] \circ^\sharp \mathbf{S}^\sharp[\![\mathsf{C}_2]\!])$$

Structural abstract semantics (cont'd)

iteration:

$$\mathbf{S}^\sharp[\![\texttt{while}\ {}^\ell\mathsf{B}\ \texttt{do}\ \mathsf{C}\ \texttt{od}]\!] \triangleq \mathsf{lfp}^{\sqsubseteq}\ \mathbf{F}^\sharp[\![\texttt{while}\ {}^\ell\mathsf{B}\ \texttt{do}\ \mathsf{C}\ \texttt{od}]\!]$$

where the transformer $\mathbf{F}^\sharp[\![\texttt{while}\ {}^\ell\mathsf{B}\ \texttt{do}\ \mathsf{C}\ \texttt{od}]\!]$ is

$$\begin{aligned}
&\mathbf{F}^\sharp[\![\texttt{while}\ {}^\ell\mathsf{B}\ \texttt{do}\ \mathsf{C}\ \texttt{od}]\!](X) \triangleq \\
&\quad \alpha(\{\langle \ell,\ \rho\rangle \mid \rho \in \mathcal{E}\}) \\
&\sqcup X \circ^\sharp \alpha(\{\langle \ell,\ \rho\rangle\langle \mathsf{i}[\![\mathsf{C}]\!],\ \rho\rangle \mid \mathit{tt} \in \mathrm{B}[\![\mathsf{B}]\!]\rho\}) \circ^\sharp \mathbf{F}^\sharp[\![\mathsf{C}]\!] \\
&\sqcup X \circ^\sharp \alpha(\{\langle \ell,\ \rho\rangle\langle \mathsf{f}[\![\texttt{while}\ {}^\ell\mathsf{B}\ \texttt{do}\ \mathsf{C}\ \texttt{od}]\!],\ \rho\rangle \mid \mathit{ff} \in \mathrm{B}[\![\mathsf{B}]\!]\rho\})
\end{aligned}$$

Fixpoint iteration

Example after invariant abstraction:

$\{y \geqslant 0\}$ $\leftarrow$ hypothesis

```
x := y
```

$\{I(x, y)\}$ $\leftarrow$ loop invariant

```
while (x > 0) do
  x := x - 1;
od
```

Abstract fixpoint equation:

$$I(x,y) = x \geqslant 0 \land (x = y \lor I(x+1, y)) \qquad (\textit{i.e.}\ I = \mathbf{F}^\sharp(I)\ ^{(3)})$$

Equivalent Floyd-Naur-Hoare verification conditions:

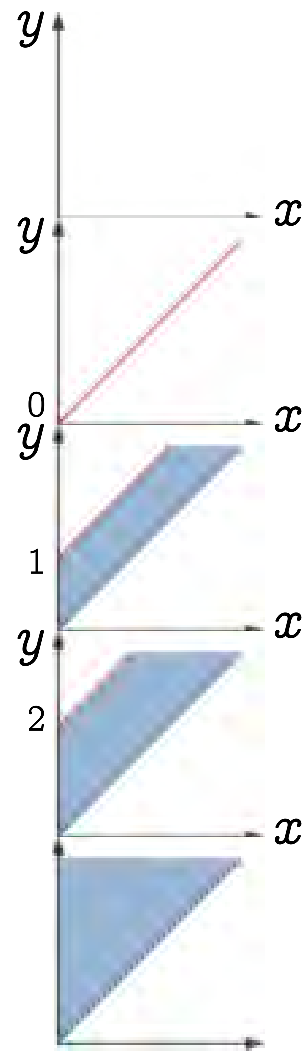
$$(y \geqslant 0 \land x = y) \Longrightarrow I(x,y) \qquad \textit{initialisation}$$

$$(I(x,y) \land x > 0 \land x' = x - 1) \Longrightarrow I(x', y) \qquad \textit{iteration}$$

(3) We look for the most precise invariant I, implying all others, that is $\mathsf{lfp}^{\Longrightarrow}\ \mathbf{F}^\sharp$.

Accelerated Iterates $I = \bigsqcup_{n\to\infty} \mathbf{F}^{\sharp^n}(\text{false})$

$$
\begin{aligned}
I^0(x,y) &= \text{false}\\
I^1(x,y) &= x \geqslant 0 \wedge (x = y \vee I^0(x+1,y))\\
&= 0 \leqslant x = y\\
I^2(x,y) &= x \geqslant 0 \wedge (x = y \vee I^1(x+1,y))\\
&= 0 \leqslant x \leqslant y \leqslant x+1\\
I^3(x,y) &= x \geqslant 0 \wedge (x = y \vee I^2(x+1,y))\\
&= 0 \leqslant x \leqslant y \leqslant x+2\\
I^4(x,y) &= I^2(x,y) \,\nabla\, I^3(x,y) \leftarrow \text{widening}\\
&= 0 \leqslant x \leqslant y\\
I^5(x,y) &= x \geqslant 0 \wedge (x = y \vee I^4(x+1,y))\\
&= I^4(x,y) \quad \text{fixed point!}
\end{aligned}
$$



The invariants are computer representable with octagons!

Convergence acceleration

Overapproximate $\mathsf{lfp}^{\sqsubseteq}\ \mathbf{F}^{\sharp}$ by $\mathsf{lfp}^{\sqsubseteq}\ \mathbf{F}^{\triangledown}$ where

$$\mathbf{F}^{\triangledown}(X) \triangleq \ \mathsf{si}\ \mathbf{F}^{\sharp}(X) \sqsubseteq X \text{ then } X \text{ else } X \mathbin{\nabla} \mathbf{F}^{\sharp}(X)$$

where the widening ∇ overapproximates

$$x \sqsubseteq x \mathbin{\nabla} y$$
$$y \sqsubseteq x \mathbin{\nabla} y$$

and enforces convergence

> For all $x_0 \sqsubseteq x_1 \sqsubseteq \ldots \sqsubseteq x_n \sqsubseteq \ldots$ the increasing sequence $y_0 = x_0, \ldots, y_{n+1} = y_n \mathbin{\nabla} x_n, \ldots$ is ultimately stationary.

Soundness theorem

$$\forall \mathbf{C} : \mathbf{S}[\![\mathbf{C}]\!] \subseteq \gamma(\mathbf{S}^\sharp[\![\mathbf{C}]\!])$$

Verification in the abstract

- Objective: Given an abstract specification $S \in A$, prove that $\mathbf{S}[\![\mathbf{C}]\!] \subseteq \gamma(S)$
- Abstraction: Prove $\mathbf{S}^\sharp[\![\mathbf{C}]\!] \sqsubseteq S$ in the abstract
- Soundness: $(\mathbf{S}^\sharp[\![\mathbf{C}]\!] \sqsubseteq S) \Longrightarrow (\mathbf{S}[\![\mathbf{C}]\!] \subseteq \gamma(S))$
- Incompleteness: $\exists \mathbf{C} : \mathbf{S}[\![\mathbf{C}]\!] \subseteq \gamma(S) \wedge \mathbf{S}^\sharp[\![\mathbf{C}]\!] \not\sqsubseteq S$ (always false alarms for some programs by undecidability)

Design choices

– Choice of abstractions $\alpha = \alpha^{\imath} \circ \cdots \circ \alpha^{I} \circ \cdots \circ \alpha^{\pi}$

– Choice of widenings ∇ (and narrowings)

– Choice of compact computer representations of abstract properties

– Design of efficient algorithms for elementary abstract operations and transformers $\sqsubseteq$, $\sqcup$, $\circ^{\sharp}$, etc

– Compositional design:

  - by composition of abstractions
  - by reduction of abstractions (see later)
  - by structural induction on program syntax

4. Scaling up

The difficulty of scaling up

- The abstraction must be coarse enough to be effectively computable with reasonable resources

- The abstraction must be precise enough to avoid false alarms

- Abstractions to *infinite domains with widenings* are more expressive than abstractions to *finite domains* (when considering the analysis of a programming language) [CC92a]

- Abstractions are ultimately incomplete (even intrinsically for some semantics and specifications [CC00])

Abstraction/refinement by tuning the cost/precision ratio in ASTRÉE

– Approximate reduced product of a choice of coarsenable/refinable abstractions

– Tune their precision/cost ratio by

  - Globally by parametrization
  - Locally by (automatic) analysis directives

  so that the overall abstraction is not uniform.

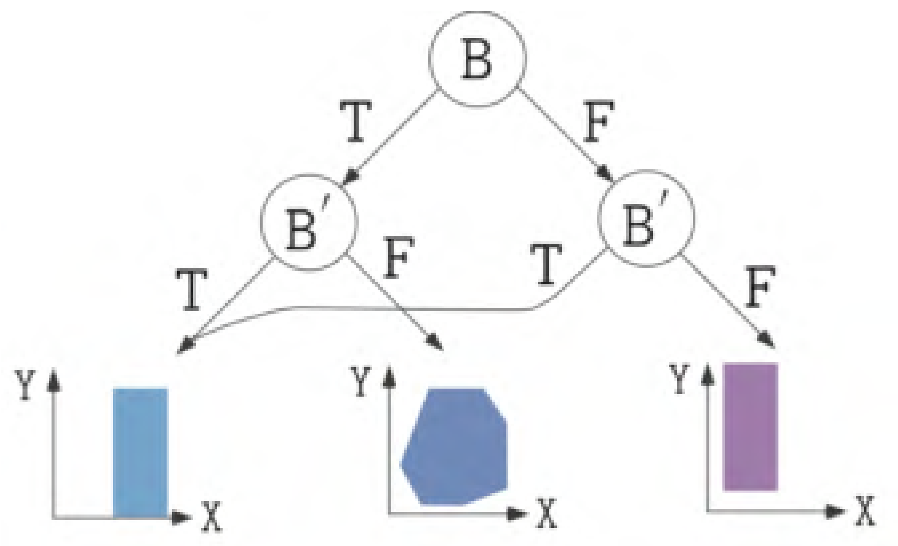
Example of abstract domain choice in ASTRÉE

```
/* Launching the forward abstract interpreter */
/* Domains:  Guard domain, and Boolean packs (based on Absolute
value equality relations, and Symbolic constant propagation
(max_depth=20), and Linearization, and Integer intervals, and
congruences, and bitfields, and finite integer sets, and Float
intervals), and Octagons, and High_passband_domain(10), and
Second_order_filter_domain (with real roots)(10), and
Second_order_filter_domain (with complex roots)(10), and
Arithmetico-geometric series, and new clock, and Dependencies
(static), and Equality relations, and Modulo relations, and
Symbolic constant propagation (max_depth=20), and Linearization,
and Integer intervals, and congruences, and bitfields, and
finite integer sets, and Float intervals.  */
```

Example of abstract domain functor in ASTRÉE: decision trees

– Code Sample:

```
/* boolean.c */
typedef enum {F=0,T=1} BOOL;
BOOL B;
void main () {
  unsigned int X, Y;
  while (1) {
    ...
    B = (X == 0);
    ...
    if (!B) {
      Y = 1 / X;
    }
    ...
  }
}
```



The boolean relation abstract domain is parameterized by the height of the decision tree (an analyzer option) and the abstract domain at the leafs

Reduction [CC79, CCF+08]

Example: reduction of intervals by simple congruences

```
% cat -n congruence.c
     1 /* congruence.c */
     2 int main()
     3 { int X;
     4   X = 0;
     5   while (X <= 128)
     6     { X = X + 4; };
     7   __ASTREE_log_vars((X));
     8 }
% astree congruence.c -no-relational -exec-fn main |& egrep "(WARN)|(X in)"
direct = <integers (intv+cong+bitfield+set): X in {132} >
```

Intervals : $X \in [129, 132]$ + congruences : $X = 0 \mod 4 \Longrightarrow X \in \{132\}$.

Parameterized abstractions

– Parameterize the cost / precision ratio of abstractions in the static analyzer

– Examples:

  - array smashing: `--smash-threshold` n (400 by default) $\rightarrow$ smash elements of arrays of size $> n$, otherwise individualize array elements (each handled as a simple variable).

  - packing in octogons: (to determine which groups of variables are related by octagons and where)
    - `--fewer-oct`: no packs at the function level,
    - `--max-array-size-in-octagons` n: unsmashed array elements of size $> n$ don't go to octagons packs

Parameterized widenings

– Parameterize the rate and level of precision of widenings in the static analyzer

– Examples:

  - delayed widenings: `--forced-union-iterations-at-beginning` n (2 by default)

  - thresholds for widening (e.g. for integers):

```
let widening_sequence =
 [ of_int 0; of_int 1; of_int 2; of_int 3; of_int 4; of_int 5;
   of_int 32767; of_int 32768; of_int 65535; of_int 65536;
   of_string "2147483647"; of_string "2147483648";
   of_string "4294967295" ]
```

Analysis directives

- Require a local refinement of an abstract domain
- Example:

```
% cat repeat1.c
typedef enum {FALSE=0,TRUE=1} BOOL;
int main () {
  int x = 100; BOOL b = TRUE;

  while (b) {
    x = x - 1;
    b = (x > 0);
  }
}
% astree -exec-fn main repeat1.c |& egrep "WARN"
repeat1.c:5.8-13::[call#main@2:loop@4>=4:]: WARN: signed int arithmetic
range [-2147483649, 2147483646] not included in [-2147483648, 2147483647]
%
```

Example of directive (cont'd)

```
% cat repeat2.c
typedef enum {FALSE=0,TRUE=1} BOOL;
int main () {
  int x = 100; BOOL b = TRUE;
 __ASTREE_boolean_pack((b,x));
  while (b) {
    x = x - 1;
    b = (x > 0);
  }
}
```

```
% astree -exec-fn main repeat2.c |& egrep "WARN"
%
```

The insertion of this directive could be automated in ASTRÉE (if the considered family of programs has "repeat" loops).

Automatic analysis directives

– The directives can be inserted automatically by static analysis

– Example:

```
% cat p.c
int clip(int x, int max, int min) {
 if (max >= min) {
  if (x <= max) {
   max = x;
  }
  if (x < min) {
   max = min;
  }
 }
 return max;
}
void main() {
 int m = 0; int M = 512; int x, y;
 y = clip(x, M, m);
  __ASTREE_assert(((m<=y) && (y<=M)));
}
% astree -exec-fn main p.c |& grep WARN
%
```

```
% astree -exec-fn main p.c -dump-partition
...
int (clip)(int x, int max, int min)
{
   if ((max >= min))
   {  __ASTREE_partition_control((0))
      if ((x <= max))
      {
         max = x;
      }
      if ((x < min))
      {
         max = min;
      }
      __ASTREE_partition_merge_last(());
   }
   return max;
}
...
%
```

Adding new abstract domains

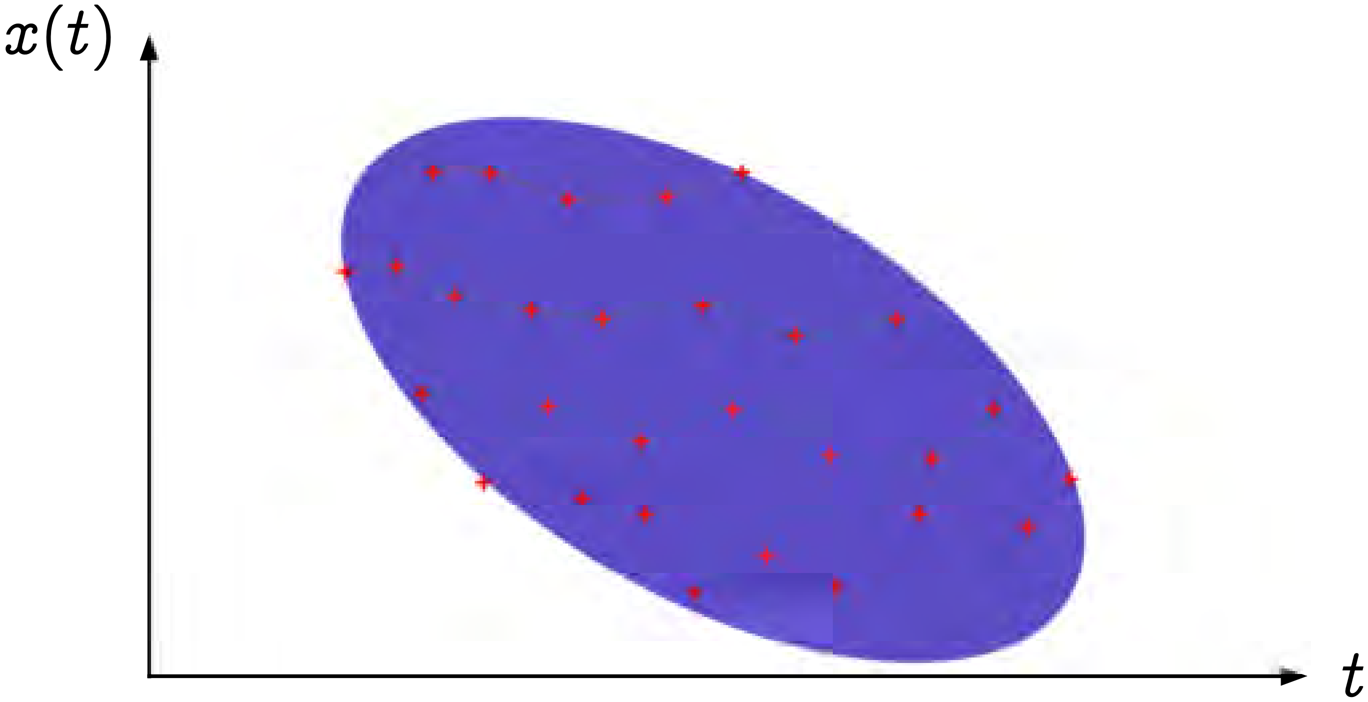
– The weakest invariant to prove the specification may not be expressible with the current refined abstractions $\Rightarrow$ false alarms cannot be solved

– No solution, but adding a new abstract domain:

  - representation of the abstract properties
  - abstract property transformers for language primitives
  - widening
  - reduction with other abstractions

– Examples : ellipsoids for filters, exponentials for accumulation of small rounding errors, quaternions, ...

Abstraction by ellipsoid for filters

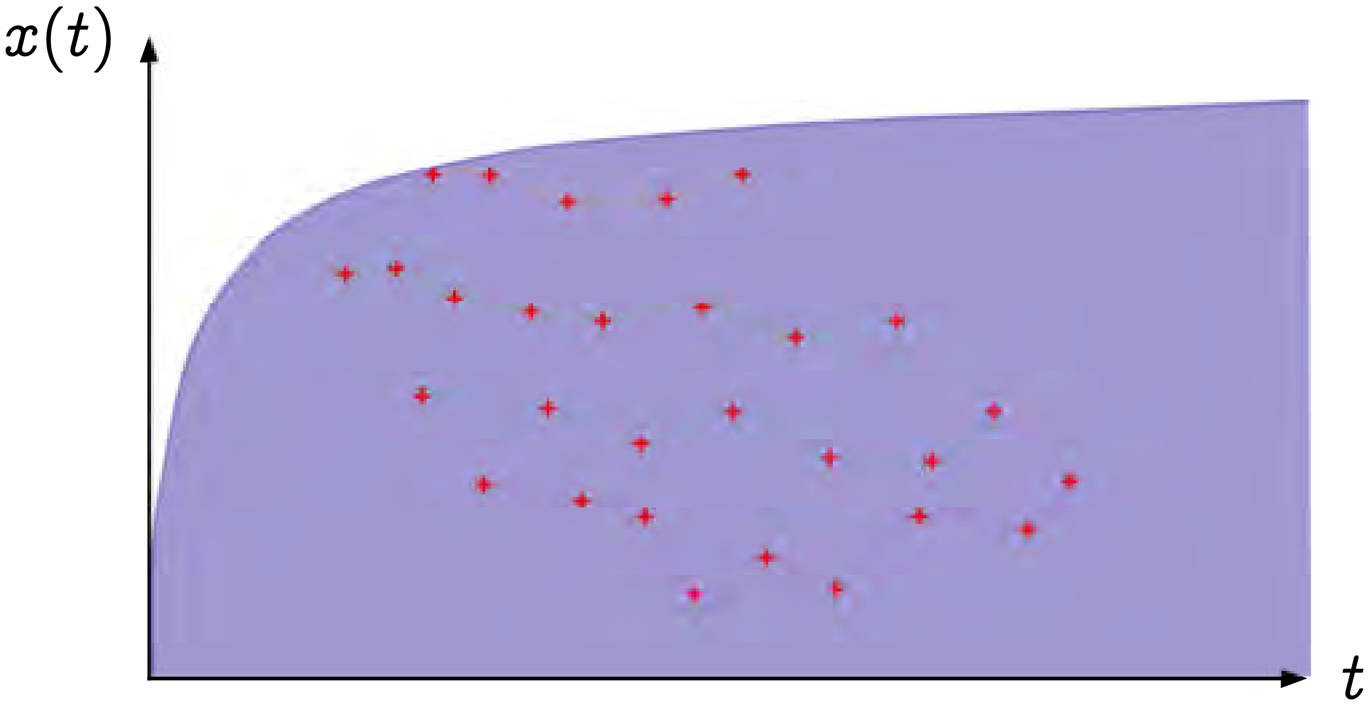


Ellipsoids $(x - a)^2 + (y - b)^2 \leq c$

Example of analysis by ASTRÉE

```
typedef enum {FALSE = 0, TRUE = 1} BOOLEAN;
BOOLEAN INIT; float P, X;
void filter () {
  static float E[2], S[2];
  if (INIT) { S[0] = X; P = X; E[0] = X; }
  else { P = (((((0.5 * X) - (E[0] * 0.7)) + (E[1] * 0.4))
              + (S[0] * 1.5)) - (S[1] * 0.7)); }
  E[1] = E[0]; E[0] = X; S[1] = S[0]; S[0] = P;
  /* S[0], S[1] in [-1327.02698354, 1327.02698354] */
}
void main () { X = 0.2 * X + 5; INIT = TRUE;
  while (1) {
    X = 0.9 * X + 35; /* simulated filter input */
    filter (); INIT = FALSE; }
}
```

Abstraction by exponentials for accumulation of small rounding errors



Exponentials $a^x \leq y$

Example of analysis by ASTRÉE

```
% cat retro.c
typedef enum {FALSE=0, TRUE=1} BOOL;
BOOL FIRST;
volatile BOOL SWITCH;
volatile float E;
float P, X, A, B;

void dev( )
{ X=E;
  if (FIRST) { P = X; }
  else
    { P =  (P - ((((2.0 * P) - A) - B)
                 * 5.0e-03)); };
  B = A;
  if (SWITCH) {A = P;}
  else {A = X;}
}
```

```
void main()
{ FIRST = TRUE;
  while (TRUE) {
    dev( );
    FIRST = FALSE;
    __ASTREE_wait_for_clock(());
  }}
% cat retro.config
__ASTREE_volatile_input((E [-15.0, 15.0]));
__ASTREE_volatile_input((SWITCH [0,1]));
__ASTREE_max_clock((3600000));
```

```
astree -exec-fn main -config-sem retro.config
retro.c |& grep "|P|" | tail -n 1
|P| <=1.0000002*((15.  +
5.8774718e-39/(1.0000002-1))*(1.0000002)ĉlock -
5.8774718e-39/(1.0000002-1)) + 5.8774718e-39 <=
23.039353
```

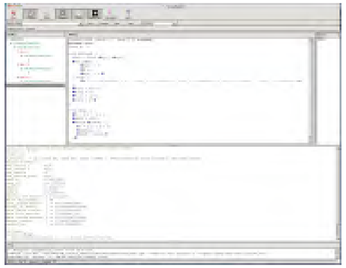
5. Industrial Application of Abstract Interpretation

Examples of sound static analyzers in industrial use

– For C critical synchronous embedded control/command programs (for example for Electric Flight Control Software)

– aiT [FHL⁺01] is a static analyzer to determine the Worst Case Execution Time (to guarantee synchronization in due time)

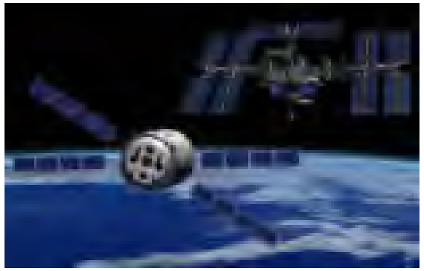
– ASTRÉE [BCC⁺03] is a static analyzer to verify the absence of runtime errors



Industrial results obtained with ASTRÉE

– Automatic proofs of absence of runtime errors in Electric Flight Control Software:
  – A340/600: 132.000 lines of C, 40mn on a PC 2.8 GHz, 300 Mb (Nov. 2003)
  – A380: 1.000.000 lines of C, 34h, 8 Gb (Nov. 2005)

  no false alarm, **World premières !**



– Automatic proofs of absence of runtime errors in the ATV software [4]:
  – C version of the automatic docking software: 102.000 lines of C, 23s on a Quad-Core AMD Opteron™ processor, 16 Gb (Apr. 2008)

[4] the Jules Vernes Automated Transfer Vehicle (ATV) enabling ESA to transport payloads to the International Space Station.

6. Applications of Abstract Interpretation

The Theory of Abstract Interpretation

- A theory of sound approximation of mathematical structures, in particular those involved in the behavior of computer systems
- Systematic derivation of sound methods and algorithms for approximating undecidable or highly complex problems in various areas of computer science
- Main practical application is on the safety and security of complex hardware and software computer systems
- Abstraction: extracting information from a system description that is relevant to proving a property

Applications of Abstract Interpretation

- Static Program Analysis (or Semantics-Checking) [CC77], [CH78], [CC79] including Dataflow Analysis; [CC79], [CC00], Set-based Analysis [CC95], Predicate Abstraction [Cou03], ...
- Grammar Analysis and Parsing [CC03];
- Hierarchies of Semantics and Proof Methods [CC92b], [Cou02];
- Typing & Type Inference [Cou97];
- (Abstract) Model Checking [CC00];
- Program Transformation (including compile-time program optimization, partial evaluation, etc) [CC02];

Applications of Abstract Interpretation (cont'd)

- Software Watermarking [CC04];
- Bisimulations [RT04, RT06];
- Language-based security [GM04];
- Semantics-based obfuscated malware detection [PCJD07].
- Databases [AGM93, BPC01, BS97]
- Computational biology [Dan07]
- Quantum computing [JP06, Per06]

All these techniques involve sound approximations that can be formalized by abstract interpretation

7. Conclusion

Conclusion

– Vision: to understand the numerical world, different levels of abstraction must be considered

– Theory: abstract interpretation ensures the coherence between abstractions and offers effective approximation techniques to cope with infinite systems

– Applications: the choice of effective abstraction which are coarse enough to be *computable* and precise enough to be *avoid false alarms* is central to master undecidability and complexity in model and program verification

The futur

– Software engineering : Manual validation by control of the software design process will be complemented by the verification of the final product

– Complex systems : abstract interpretation looks to apply equally well to the analysis of systems with discrete/hybrid evolution (image analysis [Ser94], biological systems [DFFK07, DFFK08, Fer07], quantum computation [JP06], etc)

THE END

Thank you for your attention

8. Bibliography

Short bibliography

[AGM93] G. Amato, F. Giannotti, and G. Mainetto. Data sharing analysis for a database programming language via abstract interpretation. In R. Agrawal, S. Baker, and D.A.Bell, editors, *Proc. 19th Int. Conf. on Very Large Data Bases*, pages 405–415, Dublin, IE, 24–27 Aug. 1993. MORGANKAUFMANN.

[BCC$^+$03] B. Blanchet, P. Cousot, R. Cousot, J. Feret, L. Mauborgne, A. Miné, D. Monniaux, and X. Rival. A static analyzer for large safety-critical software. In *Proc. ACM SIGPLAN '2003 Conf. PLDI*, pages 196–207, San Diego, CA, US, 7–14 June 2003. ACM Press.

[BPC01] J. Bailey, A. Poulovassilis, and C. Courtenage. Optimising active database rules by partial evaluation and abstract interpretation. In *Proc. 8th Int. Work. on Database Programming Languages*, LNCS 2397, pages 300–317, Frascati, IT, 8–10 Sep. 2001. Springer.

[BS97] V. Benzaken and X. Schaefer. Static integrity constraint management in object-oriented database programming languages via predicate transformers. In M. Aksit and S. Matsuoka, editors, *Proc. 11th European Conf. on Object-Oriented Programming, ECOOP '97*, LNCS 1241. Springer, Jyväskylä, FI, 9–13 June 1997.

[CC77] P. Cousot and R. Cousot. Abstract interpretation: a unified lattice model for static analysis of programs by construction or approximation of fixpoints. In *4th POPL*, pages 238–252, Los Angeles, CA, 1977. ACM Press.

[CC79] P. Cousot and R. Cousot. Systematic design of program analysis frameworks. In *6th POPL*, pages 269–282, San Antonio, TX, 1979. ACM Press.

[CC92a] P. Cousot and R. Cousot. Comparing the Galois connection and widening/narrowing approaches to abstract interpretation, invited paper. In M. Bruynooghe and M. Wirsing, editors, *Proc. 4th Int. Symp. on PLILP '92*, Leuven, BE, 26–28 Aug. 1992, LNCS 631, pages 269–295. Springer, 1992.

[CC92b] P. Cousot and R. Cousot. Inductive definitions, semantics and abstract interpretation. In *19th POPL*, pages 83–94, Albuquerque, NM, US, 1992. ACM Press.

[CC95] P. Cousot and R. Cousot. Formal language, grammar and set-constraint-based program analysis by abstract interpretation. In *Proc. 7th FPCA*, pages 170–181, La Jolla, CA, US, 25–28 June 1995. ACM Press.

[CC00] P. Cousot and R. Cousot. Temporal abstract interpretation. In *27th POPL*, pages 12–25, Boston, MA, US, Jan. 2000. ACM Press.

[CC02] P. Cousot and R. Cousot. Systematic design of program transformation frameworks by abstract interpretation. In *29th POPL*, pages 178–190, Portland, OR, US, Jan. 2002. ACM Press.

[CC03] P. Cousot and R. Cousot. Parsing as abstract interpretation of grammar semantics. *Theoret. Comput. Sci.*, 290(1):531–544, Jan. 2003.

[CC04] P. Cousot and R. Cousot. An abstract interpretation-based framework for software watermarking. In *31st POPL*, pages 173–185, Venice, IT, 14–16 Jan. 2004. ACM Press.

[CCF+07] P. Cousot, R. Cousot, J. Feret, L. Mauborgne, A. Miné, D. Monniaux, and X. Rival. Varieties of static analyzers: A comparison with ASTRÉE, invited paper. In M. Hinchey, He Jifeng, and J. Sanders, editors, *Proc. 1st TASE '07*, pages 3–17, Shanghai, CN, 6–8 June 2007. IEEE Comp. Soc. Press.

[CCF+08] P. Cousot, R. Cousot, J. Feret, L. Mauborgne, A. Miné, D. Monniaux, and X. Rival. Combination of abstractions in the ASTRÉE static analyzer. In M. Okada and I. Satoh, editors, *11th ASIAN 06*, pages 272–300, Tokyo, JP, 6–8 Dec. 2006, 2008. LNCS 4435, Springer.

[CH78] P. Cousot and N. Halbwachs. Automatic discovery of linear restraints among variables of a program. In *5th POPL*, pages 84–97, Tucson, AZ, 1978. ACM Press.

[Cou97] P. Cousot. Types as abstract interpretations, invited paper. In *24th POPL*, pages 316–331, Paris, FR, Jan. 1997. ACM Press.

[Cou02] P. Cousot. Constructive design of a hierarchy of semantics of a transition system by abstract interpretation. *Theoret. Comput. Sci.*, 277(1—2):47–103, 2002.

[Cou03] P. Cousot. Verification by abstract interpretation, invited chapter. In N. Dershowitz, editor, *Proc. Int. Symp. on Verification – Theory & Practice – Honoring Zohar Manna's 64th Birthday*, pages 243–268. LNCS 2772, Springer, Taormina, IT, 29 June – 4 Jul. 2003.

[Dan07] V. Danos. Abstract views on biological signaling. In *Mathematical Foundations of Programming Semantics, 23rd Annual Conf. (MFPS XXIII)*, 2007.

[DFFK07] V. Danos, J. Feret, W. Fontana, and J. Krivine. Scalable simulation of cellular signaling networks. In Zhong Shao, editor, *Proc. 5th APLAS '2007*, pages 139–157, Singapore, 29 Nov. –1 Dec. 2007. LNCS 4807, Springer.

[DFFK08] V. Danos, J. Feret, W. Fontana, and J. Krivine. Abstract interpretation of cellular signalling networks. In F. Loggozzo, D. Peled, and L.D. Zuck, editors, *Proc. 9th Int. Conf. VMCAI 2008*, pages 83–97, San Francisco, CA, US, 7–9 Jan. 2008. LNCS 4905, Springer.

[DS07] D. Delmas and J. Souyris. ASTRÉE: from research to industry. In G. Filé and H. Riis-Nielson, editors, *Proc. 14th Int. Symp. SAS '07*, Kongens Lyngby, DK, LNCS 4634, pages 437–451. Springer, 22–24 Aug. 2007.

[Fer07] J. Feret. Reachability analysis of biological signalling pathways by abstract interpretation. In T.E. Simos and G. Maroulis, editors, *Computation in Modern Science and Engineering: Proc. 6th Int. Conf. on Computational Methods in Sciences and Engineering (ICCMSE'07)*, volume American Institute of Physics Conf. Proc. 963 (2, Part A & B), pages 619–622. AIP, Corfu, GR, 25–30 Sep. 2007.

[FHL+01] C. Ferdinand, R. Heckmann, M. Langenbach, F. Martin, M. Schmidt, H. Theiling, S. Thesing, and R. Wilhelm. Reliable and precise WCET determination for a real-life processor. In T.A. Henzinger and C.M. Kirsch, editors, *Proc. 1st Int. Work. EMSOFT '2001*, volume 2211 of *LNCS*, pages 469–485. Springer, 2001.

[GM04] R. Giacobazzi and I. Mastroeni. Abstract non-interference: Parameterizing non-interference by abstract interpretation. In *31st POPL*, pages 186–197, Venice, IT, 2004. ACM Press.

[JP06] Ph. Jorrand and S. Perdrix. Towards a quantum calculus. In *Proc. 4th Int. Work. on Quantum Programming Languages, ENTCS*, 2006.

[PCJD07] M. Dalla Preda, M. Christodorescu, S. Jha, and S. Debray. Semantics-based approach to malware detection. In *34th POPL*, pages 238–252, Nice, France, 17–19 Jan. 2007. ACM Press.

[Per06] S. Perdrix. *Modèles formels du calcul quantique : ressources, machines abstraites et calcul par mesure.* PhD thesis, Institut National Polytechnique de Grenoble, Laboratoire Leibniz, 2006.

[RT04] F. Ranzato and F. Tapparo. Strong preservation as completeness in abstract interpretation. In D. Schmidt, editor, *Proc. 30th ESOP '04*, volume 2986 of *LNCS*, pages 18–32, Barcelona, ES, Mar. 29 – Apr. 2 2004. Springer.

[RT06] F. Ranzato and F. Tapparo. Strong preservation of temporal fixpoint-based operators by abstract interpretation. In A.E. Emerson and K.S. Namjoshi, editors, *Proc. 7th Int. Conf. VMCAI 2006*, pages 332–347, Charleston, SC, US, 8–10 Jan. 2006. LNCS 3855 , Springer.

[Ser94] J. Serra. Morphological filtering: An overview. *Signal Processing*, 38:3–11, 1994.