Static Analysis of Floating-Point Numbers by Abstract Interpretation
Current Status and Problems

Eric Goubault
Joint Work with Matthieu Martel, Sylvie Putot
Partially done under the auspices of RTD project IST-1999-20527 “DAEDALUS”
CEA - Recherche Technologique
Eric.Goubault@cea.fr
Road map

- The problem through examples: experiments, and some mathematics
- Language and standard semantics,
- Comparison of floating-point and real number semantics using non-standard semantics
- Abstractions, iteration strategies, widening operations
- Related and future work.
What about floating-point numbers?

- Machines do not use real numbers but a finite approximation of them: floating-point numbers (here the IEEE754 norm)
- This is error-prone (not that different from “runtime bugs”)
- Very difficult to get some confidence in numerical codes! Very subtle properties (stability etc.)
Problem

Computers compute wrong results...

- transcendental numbers...
- non-associativity of elementary operations (+, -, *, / etc.),
- \[ f(a, b) = 333.75b^6 + a^2(11a^2b^2 - b^6 - 121b^4 - 2) + 5.5b^8 + \frac{a}{2b} \]

\[
\begin{align*}
    f_{SU N}(77617, 33096) &= 1.172603... \\
    f_{E X A C T}(77617, 33096) &= -0.827396...
\end{align*}
\]
The IEEE 754 norm

Memory representation (and not register representation...),

<table>
<thead>
<tr>
<th>Sign</th>
<th>Exponent</th>
<th>Mantissa</th>
</tr>
</thead>
</table>

Standard numbers,

\[ r = s \times n \times 2^{k+1-N} \] with \( s = -1, 1; 1 - 2^K < k < 2^K; -2^N < n < 2^N \)

normalised so that \( r = s \times 2^k (1 + f) \) with \( f < 1 \)

come in several versions (simple, double, double extended...)

- 5 -
Other characteristics

- Denormalised numbers (to manage “underflow” in a gradual manner),

\[
\begin{align*}
  r &= s \times n \times 2^{k+1-N} = s \times 2^k (0 + f) \text{ with } k = 2 - 2^K, \\
  &\quad 0 < n < 2^{N-1} \text{ i.e. } 0 < f < 1
\end{align*}
\]

- \( +\infty \) and \(-\infty \) (notice that their inverses, \( +0 \) et \(-0 \) are also there),

- NaN “Not a Number” signed or not (for instance \( 0 \times \infty \)),

- you can control roundoff methods...
Facts and numbers

For a simple float,

- Maximum normalised: \( \sim 3.40282347 \times 10^{38} \)
- Minimum positive normalised: \( \sim 1.17549435 \times 10^{-38} \)
- Maximum denormalised: \( \sim 1.17549421 \times 10^{-38} \)
- Minimum positive denormalised: \( \sim 1.40129846 \times 10^{-45} \)
- Around 1, maximal error ("ulp") = \( 2^{-23} \sim 1.19200928955 \times 10^{-7} \)

Norm specifies: +, −, *, /, \( \sqrt{\cdot} \) computed with an imprecision \( < \frac{ulp}{2} \) around the exact result (when no overflow)...
Examples

- 25th of February 1991 (Gulf war) A Patriot missile misses a Scud in Dharan and crashes on an American barracks, killing 28 soldiers.

- (GAO/IMTEC-92-26) Time conversion from $\frac{1}{10}$ seconds to seconds in a 24 bits register.

\[
\frac{1}{10} = 0.00011001100110011001100\cdots
\]

<table>
<thead>
<tr>
<th>Error Type</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Truncating error</td>
<td>$\sim 0.000000095$ (decimal)</td>
</tr>
<tr>
<td>100 hours drift</td>
<td>$\sim 0.34s$</td>
</tr>
<tr>
<td>Localization error</td>
<td>$\sim 500m$</td>
</tr>
</tbody>
</table>
Example: cancellation

Area of flat triangle ($a$ is very close to $b + c$):

$$A = \sqrt{s(s - a)(s - b)(s - c)} \quad s = \frac{a + b + c}{2}$$

Then $s - a$ is almost irrelevant...
Example - instability

main()
{
    float x, y, z;
    int i;
    x=1;
    y=(sqrt(5)-1)/2;
    for (i=1; i<=20; i++) {
        z=x;
        x=y;
        y=z-y;
        printf("phi^%d=%f\n", i, x);
    }
}

main()
{
    float t;
    int i;
    t=1;
    for (i=1; i<=20; i++) {
        t=t*(sqrt(5)-1)/2;
        printf("phi^%d=%f\n", i, t);
    }
}
Execution

\[ \phi^1 = 0.618034 \]
\[ \phi^2 = 0.381966 \]
\[ \phi^3 = 0.236068 \]
\[ \ldots \]
\[ \phi^{17} = 0.000306 \]
\[ \phi^{18} = 0.000131 \]
\[ \phi^{19} = 0.000176 \]
\[ \phi^{20} = -0.000045 \]
Stable example

\[ x=1; \]
\[ y=-1.0/3.0; \]
\[ z=x; \]
\[ x=y; \]
\[ y=(x+z)/6; \]

for (i=1;i<=20;i++) {
  \[ z=x; \]
  \[ x=y; \]
  \[ y=(x+z)/6; \]
}

\[ \phi^1=-0.333333 \]
\[ \phi^2=0.111111 \]
\[ \phi^3=-0.037037 \]
\[ ... \]
\[ \phi^{19}=-0.000000 \]
\[ \phi^{20}=0.000000 \]
Conditioning of a linear system

Everything in $R$ below!

- Given $F : R^n \rightarrow R^m$ a linear functional,

- $K(F) = \sup\{ \| F(u) \| / \| u \| = 1 \}$

- In the linear case, initial errors are magnified by at most a factor $K(F)$ at the $i$th iterate of $F$
Linear case

- For instance, if $F$ is diagonalizable (diagonal matrix being $B$), at the $i$th iterate, the “magnification” of the errors is (in a suitable basis):

$$B^n = \begin{pmatrix} c_1^n & 0 \\ 0 & c_2^n \end{pmatrix}$$

- Hence the iterative system is stable (meaning the initial error will not be magnified) if all eigenvalues of $F$ are strictly less than 1.

In the previous example, one of the two eigenvalues was greater than one ($\frac{\sqrt{5}+1}{2}$).
A Precision-Sensitive Stable Filter (Output)

Our aim is slightly different from stability: we are not only interested in the effect of initial errors on data (there is of course a strong link)

\[ y_n = x_n + \frac{3}{4} y_{n-1} - \frac{1}{8} y_{n-2} \quad y_0 = 1 \quad y_1 = \frac{1}{6} \quad x_n = \frac{1}{6^n} \]
A Precision-Sensitive Stable Filter (Relative Error)

\[ y_n = x_n + \frac{3}{4} y_{n-1} - \frac{1}{8} y_{n-2} \quad y_0 = 1 \quad y_1 = \frac{1}{6} \quad x_n = \frac{1}{6^n} \]

Stable but many cancellations!!!
Strategy

We assume a toy imperative language...

“specification” is the real number semantics
“implementation” is the floating-point semantics
Idea about mismatch+contribution of control points?
Plan

- Define (a bunch of) operational semantics
- Find suitable abstraction/concretization pairs
- How to compute the lfp of the abstract functional? (iteration strategies)
- How to ensure termination of the lfp computation? (extrapolation - or widening)
Some difficulties of a static analysis

- Must take care of
  - conversions
  - order of evaluation...
  - memory to register operations...
  - not fully specified functions (transcendental...)

- Also: very subtle problem (as numerical analysts do know!)
Standard semantics

- Evaluation of expressions (IEEE754 style)...
- Assignments have the following semantics:
  
  \[[X = \text{Expr}]^f_\rho = \rho[X \leftarrow [\text{Expr}]^f_\rho]\]

- Transitions from state (Instr; Prog, \rho) to (Prog, \rho') are now rather easy to write down... (SOS style for instance),

- Similar kind of semantics for real number semantics.
“Comparison” semantics

(synchronized) “product” of the floating-point and real semantics:

environments: $\rho(X) = (X_{FLOAT}, X_{REAL})$

Only trouble: unstable tests,

```
(if (x < 0) x = x + 1 else x = x - 1, (x ← -10^{-37}, x ← 0))
  ↓
  (x = x + 1, (x ← -10^{-37}, ⊥))
  ↓
  (∅, (x ← 1, ⊥))
```
Unstable tests

What to do when, because of a test, the real values can follow a different path than the floating-point values!?

(1) Follow only the floating-point value path, and only signal any unstable test?

(2) Follow the different paths, and take results of the real path as reference to compute additional errors (“sure”, but overapproximated)?

Current implementation chose (1) (for the time being).
What’s next?

- Let write \((X_{\text{float}}, X_{\text{real}})\) as \(X_{\text{float}} \epsilon + (X_{\text{real}} - X_{\text{float}}) \epsilon_g \ldots\)

- Let us try to “explain” where the difference \(X_{\text{real}} - X_{\text{float}}\) comes from,

- By decomposing the difference on all roundings that take place in the computation of a variable (hence decompose \(\epsilon_g\) using \(\epsilon_{l_1}, \ldots, \epsilon_{l_n}\)),

- So as to spot the main contributions to the imprecision errors

Hence change the semantics of evaluation of expressions (some links with affine arithmetic). The correctness of our intentions will be given by Galois connections with the difference semantics.
### FP-numbers: tracking the errors

<table>
<thead>
<tr>
<th>Operation</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>[\times^{\ell_3}]</td>
<td>[799.6131\varepsilon\cdot\varepsilon]</td>
</tr>
<tr>
<td>+</td>
<td>[0.05\varepsilon_{\ell_1}]</td>
</tr>
<tr>
<td>+</td>
<td>[0.0005\varepsilon_{\ell_2}]</td>
</tr>
<tr>
<td>+</td>
<td>[0.06435\varepsilon\cdot\varepsilon_{\ell_1}]</td>
</tr>
<tr>
<td>+</td>
<td>[0.31065\varepsilon\cdot\varepsilon_{\ell_2}]</td>
</tr>
<tr>
<td>+</td>
<td>[0.000025\varepsilon_{\ell_1}\cdot\varepsilon_{\ell_2}]</td>
</tr>
</tbody>
</table>

**Result**

- Error due to \(r_{1}^{\ell_1}\)
- Error due to \(r_{2}^{\ell_2}\)
- Second order error term

**Machine result**

- Error due to \(r_{1}^{\ell_1}\)
- Error due to \(r_{2}^{\ell_2}\)
- Second order error term
- Error introduced by \(\times^{\ell_3} = \uparrow_o (r_1 \times r_2)\)

- Error introduced by \(\times^{\ell_3} = \downarrow_o (r_1 \times r_2)\)
**Representation of values**

A concrete semantics to modelize the propagation of errors in finite precision computations:

\[ r \in \mathbb{R} \rightarrow f \tilde{\varepsilon} + \sum_{u \in \overline{L}} \omega^u \tilde{\varepsilon}_u \in F(\mathbb{R}, \overline{L}^*) \]

- \( f \in \mathbb{F} \) is the f.p. number used by the computer instead of \( r \)
- \( \overline{L} = \text{Lab(prg)} \cup \{\text{os}\} \):
  - for \( u = \ell_i \in \text{Lab(prg)} \), \( \tilde{\varepsilon}_u \) is a formal variable related to the first order error due to point \( \ell_i \)
  - \( \tilde{\varepsilon}_{\text{os}} \) is a formal variable related to all errors of order greater than one
- \( \omega^u \in \mathbb{R} \) is the coefficient of \( \tilde{\varepsilon}_u \)
Elementary Operations

\[ r_1 + \ell^i r_2 \overset{\text{def}}{=} \uparrow_0 (f_1 + f_2)\bar{e} + \sum_{u \in \mathcal{L}} (\omega_1^u + \omega_2^u)\bar{e}_u + \downarrow_0 (f_1 + f_2)\bar{e}_{\ell^i} \]

\[ r_1 - \ell^i r_2 \overset{\text{def}}{=} \uparrow_0 (f_1 - f_2)\bar{e} + \sum_{u \in \mathcal{L}} (\omega_1^u - \omega_2^u)\bar{e}_u + \downarrow_0 (f_1 - f_2)\bar{e}_{\ell^i} \]

\[ r_1 \times \ell^i r_2 \overset{\text{def}}{=} \uparrow_0 (f_1 f_2)\bar{e} + \sum_{u \in \mathcal{L}} (f_1 \omega_2^u + f_2 \omega_1^u)\bar{e}_u + \sum_{u \in \mathcal{L}, v \in \mathcal{L}} \omega_1^u \omega_2^v \bar{e}_{os} + \downarrow_0 (f_1 f_2)\bar{e}_{\ell^i} \]
Elementary Operations

\[
(r_1)^{-1_{\ell_i}} \overset{\text{def}}{=} \uplus (f_1^{-1}) \vec{e} - \frac{1}{f_1} \sum_{\ell \in \mathcal{L}} \frac{\omega^\ell}{f_1} \vec{e}_\ell + \frac{1}{f_1} \sum_{n \geq 2} (-1)^n \left( \sum_{\ell \in \mathcal{L}} \frac{\omega^\ell}{f_1} \right)^n \vec{e}_s + \downarrow (f_1^{-1}) \vec{e}_{\ell_i}
\]

\[
(r_1 \div_{\ell_i} r_2) \overset{\text{def}}{=} r_1 \times_{\ell_i} (r_2)^{-1_{\ell_i}}
\]

\[
\sqrt{r}^{\ell_i} = \uplus (\sqrt{f}) \vec{e} + \downarrow (\sqrt{f}) \vec{e}_{\ell_i} + \sum_{n \geq 1} \left[ \frac{\frac{1}{2}(-\frac{1}{2}) \ldots (\frac{3}{2} - n)}{n!} \sqrt{f} \left( \sum_{u \in \mathcal{N}^+} \frac{\omega^u}{f} \vec{e}_u \right)^n \right]
\]

(only with suitable restrictions, due to the convergence radius)
799.988125 = 799.6\bar{e} + 0.06435\bar{e}_{\ell_1} + 0.31065\bar{e}_{\ell_2} + 0.000025\bar{e}_{\ell_1\ell_2} + 0.0131\bar{e}_{\ell_3}
**Abstractions**

- Abstract the order of the errors one would like to see, e.g.:

\[ 10 + 0.5\epsilon_{l_1} + \ldots + 0.5\epsilon_{l_6} + 0.05\epsilon_{hi} \]

- Abstract the grain of errors one would like to see, e.g.:

\[ 10 + 1.5\epsilon_{J_1} + 1.5\epsilon_{J_2} + 0.05\epsilon_{hi} \]

- Abstract the numerical values, of values and of the errors (for a class of executions), e.g.:

\[ [5, 15] + [1, 2]\epsilon_{J_1} + [1, 2]\epsilon_{J_2} + [0, 0.1]\epsilon_{hi} \]

It is useful to think of the \( \epsilon \) as formal variables that can take values in

\[
\left\lfloor \frac{ulp(1)}{2}, \frac{ulp(1)}{2} \right\rfloor
\]
Error order

Galois connection for the error order (amelioration of EG SAS’01, in MM ESOP’02):

\[
\langle \varphi(\mathcal{F}(\mathbb{R}, \overline{L}^n)), \subseteq \rangle \overset{\gamma_{m,n}}{\equiv} \langle \mathcal{F}(\mathbb{R}, \overline{L}^m), \subseteq \rangle
\]

\( \varphi(X) \) denotes the power set of \( X \) and \( \subseteq \) denotes the componentwise ordering on formal series. \( \alpha_{n,m} \) and \( \gamma_{m,n} \) are defined by

\[
\alpha_{n,m} \left( \left\{ \sum_{u \in \overline{L}^n} \omega_i^u \varepsilon_u, i \in I \right\} \right) \overset{\text{def}}{=} \sum_{u \in \overline{L}^m \setminus \{\varsigma\}} \left( \bigvee_{i \in I} \omega_i^u \right) \varepsilon_u + \bigvee_{i \in I} \left( \sum_{u \in (\overline{L}^n \setminus \overline{L}^m) \cup \{\varsigma\}} \omega^u \right) \varepsilon_u
\]

\[
\gamma_{m,n} \left( \sum_{u \in \overline{L}^m} \nu^u \varepsilon_u \right) \overset{\text{def}}{=} \left\{ \sum_{u \in \overline{L}^n} \omega^u \varepsilon_u : \omega^u \leq \nu^u \text{ if } u \in \overline{L}^m \setminus \{\varsigma\} \right\}
\]

\[
\sum_{u \in (\overline{L}^n \setminus \overline{L}^m) \cup \{\varsigma\}} \omega^u \leq \nu^s
\]
Change of partition

- Let $\mathcal{J} = \{J_1, J_2, \ldots, J_p\} \in \mathcal{P}(\mathcal{L})$ be a partition of the program points.

- We consider now the words on the alphabet $\mathcal{J}$.

- $\mathcal{J}^n$ denotes the words of maximal length $n$ and $\overline{\mathcal{J}}^n = (\mathcal{J}^n / \sim) \cup \{\varsigma\}$.

- For a maximal order of error $n \in \mathbb{N}$, we consider the family of domains $(\mathcal{F}(R, \overline{\mathcal{J}}^n))_{\mathcal{J} \in \mathcal{P}(\mathcal{L})}$. 
Change of partition

A value $r^{\mathcal{J}^n} \in \mathbb{R}^{\mathcal{J}^n}$ is written

$$r^{\mathcal{J}^n} = f \tilde{\varepsilon}^* + \sum_{u \in \mathcal{J}^{n+}} \left( \sum_{v = \ell_1 \ldots \ell_k \in \mathcal{L}^n} \omega^v \right) \tilde{\varepsilon}_u = f \tilde{\varepsilon}^* + \sum_{u \in \mathcal{J}^{n+}} \omega^u \tilde{\varepsilon}_u$$

$$u = J_1 \ldots J_k \quad \forall i, \ 1 \leq i \leq k, \ \ell_i \in J_i$$

If $|u| = 1$, the word $u = J$ is related to the first order error due to the operations whose label belongs to $J$. 
Comparison between partitions

Let $\mathcal{J}_1$ and $\mathcal{J}_2$ be two partitions of the set $\mathcal{L}$. We say that $\mathcal{J}_1$ is a coarser partition of $\mathcal{L}$ than $\mathcal{J}_2$ and we write $\mathcal{J}_1 \preceq \mathcal{J}_2$ iff

$$\forall J_2 \in \mathcal{J}_2, \ \exists J_1 \in \mathcal{J}_1 : J_2 \subseteq J_1$$
Galois Connections

\[ \langle \varphi(F(R, J_2^n)), \subseteq \rangle \overset{\gamma J_1^n, J_2^n}{\Rightarrow} \langle F(R, J_1^n), \subseteq \rangle \]

defined by

\[ \alpha J_2^n, J_1^n \left( \left\{ \sum_{u \in J_2^n} \omega_i^u \vec{e}_u, i \in I \right\} \right) \overset{\text{def}}{=} \sum_{u \in J_2^n} \left( \bigvee_{i \in I} \omega_i^u \right) \vec{e}_{\tau J_2^n, J_1^n(u)} \]

\[ \gamma J_1^n, J_2^n \left( \sum_{v \in J_1^n} \nu_i^u \vec{e}_v \right) \overset{\text{def}}{=} \left\{ \sum_{u \in J_2^n} \omega_i^u \vec{e}_u : \sum_{\tau J_2^n, J_1^n(u) = v} \omega_i^u \leq \nu_i^u \right\} \]
Grid of abstractions

\[ S^\mathcal{I}_0^* \iff \ldots \iff \gamma^{n,n+1}_{\alpha^{n+1,n}} \quad S^\mathcal{I}_0^n \iff \gamma^{n-1,n}_{\alpha^{n,n-1}} \iff \ldots \iff S^\mathcal{I}_0^0 \]

\[ \alpha^\mathcal{I}_0^n, \mathcal{I}_k^n \uparrow \downarrow \gamma^\mathcal{I}_k^n, \mathcal{I}_0^n \]

\[ S^\mathcal{L}_k^* \iff \ldots \iff \quad S^\mathcal{L}_k^n \iff \ldots \iff S^\mathcal{L}_k^0 \]

\[ S^\mathcal{L}_0^* \iff \ldots \iff \quad S^\mathcal{L}_0^n \iff \ldots \iff S^\mathcal{L}_0^0 \]
Abstractions of the numerical values

We have to abstract a set of power series: we can abstract the sets of coefficients in a:

- Non-relational way
- Or relational way
Non-relational: for instance, intervals

This is the current implementation:

- Use the classical non-relational abstraction on each of the coefficients of the power series (floating-point value+errors):
  \[ \Gamma : D \rightarrow \wp(R \cup \{\infty, -\infty\}) \] by
  \[ \Gamma(a_0 + \sum_{i \in L} a_i \varepsilon_i) = \gamma(a_0) + \sum_{i \in L} \gamma(a_i) \ast ulp(1). \]

- Implementation: use `rounddown` and `roundup` for operations computing the lower and upper bounds of intervals respectively:
  \[ [a, b] \oplus [c, d] = [M'_-(a +^f c), M'_+(b +^f d)] \]

- Implementation: be careful with the representation of errors, the analyzer also makes mistakes! (in our case, use of MPFR)
Relational Analyses

Still using affine forms:

- Taking care of dependencies between errors and the f.p. value:
  an analysis with ratios

\[
\frac{\omega_i}{f}
\]

may be useful to narrow also the error intervals in case of tests or thresholds

- Taking care of linear dependencies between errors: with abstract variables and dependency coefficients instead of intervals (first proposal in EG SAS’01 paper, further work by SP and by MM)
Practicalities, for linear approximations

- Abstract concrete functionals by matrices with interval coefficients
- Determine the conditioning of the abstract functional using “Gershgorin discs”
Gershgorin discs

- For a matrix $A$, the eigenvalues are contained in $D_1 \cap D_2$, where $D_1$ and $D_2$ are the disks of the complex plane $\mathbb{C}$ defined by:

$$D_1 = \bigcup_{1 \leq i \leq m} D_{1,i} \quad D_2 = \bigcup_{1 \leq j \leq m} D_{2,j}$$

$D_{1,i}$ is the disk of center $a_{ii}$ and radius $r_{1,i} = \sum_{1 \leq j \leq m, j \neq i} |a_{ij}|$. Similarly, $D_{2,j}$ is the disk of center $a_{jj}$ and radius $r_{2,j} = \sum_{1 \leq i \leq m, i \neq j} |a_{ij}|$.

- So, an upper bound of the module of the greatest eigenvalues of $A$ is given by:

$$G(A) = \max(|a_{ii}| + \max(r_{1,i}, r_{2,i}) : 1 \leq i \leq m)$$
Non-linear case: Lyapunov exponents

• Let \( f : \mathbb{R}^m \rightarrow \mathbb{R}^m \) and \( J_n = D f^n(v_0) \).

• For all \( k = 1, \ldots, m \), we write \( r^n_k \) for the length of the \( k \)th longest orthogonal axis of the ellipse \( J_n U \).

• The \( k \)th Lyapunov exponent is:

\[
\lambda_k = \lim_{n \to \infty} \frac{\ln r^n_k}{n}
\]

• The function is stable if the greatest Lyapunov exponent is a negative number.
Lyapunov exponents in a static analyzer

- Amelioration of EG SAS’01 by MM SAS’02
- $Df^k(x_0)$ is a matrix of intervals and we use the Gerschgorin disks to find an upper bound of the greatest module of its eigenvalues.
- The greatest Lyapunov exponent after $k$ iterations is bounded by the abstract Lyapunov exponent:

$$\lambda^#_k = \frac{\ln \left( G(Df^k(x_0)) \right)}{k}$$
Resolution of the lfp equations

- Problems (imprecision of the lfp computation!):
  - Static and dynamic partitioning
  - Widenings
Parameters

- **unfoldloopnumber:** virtual unfolding of loops

```c
int i; float t=1;
for (i=0; i<20; i++)
    t=t*.618
```

- with `unfoldloopnumber=1`; equivalent to the same program;
- with `unfoldloopnumber=2`; equivalent to:

```c
int i; float t=1;
for (i=0; i<10; i++)
    { t=t*.618;
      t=t*.618;   }
```
Need for a virtual unfolding of loops

$t = 1$;
for (i=1 ; i<=20 ; i++)
    $t = t * 0.618;$ (epsilon)

- Result without unfolding : $t \in [0, 0.618]$, error $\in ]-\infty, +\infty[$

$i = 1$ : $t_1 = [0.618, 0.618] + 0.618 \text{ulp}(1)[-1, 1]\epsilon$

$i = 2$ : \[
\begin{cases}
    t_2 = [0.618^2, 0.618^2] + 2 \times 0.618^2 \text{ulp}(1)[-1, 1]\epsilon \\
    t_2 := t_1 \cup t_2 = [0.618^2, 0.618] + 2 \times 0.618^2 \text{ulp}(1)[-1, 1]\epsilon \quad (2 \times 0.618^2 > 0.618)
\end{cases}
\]

$i = 3$ : $t_3 = [0.618^3, 0.618] + (2 \times 0.618^3 + 0.618^2) \text{ulp}(1)[-1, 1]\epsilon$

The computed error increases while the error really committed decreases

(could be solved with a good widening : limit of error $\frac{1}{1-0.618} \text{ulp}(1)$)
Virtual unfolding of loops

A possible solution: unfold twice the loop

\[ i = 1 : \begin{cases} 
    t_1 = [0.618, 0.618] + 0.618 \cdot ulp(1) [-1, 1] \varepsilon_1 \\
    t_2 = [0.618^2, 0.618^2] + 0.618^2 \cdot ulp(1) [-1, 1] \varepsilon_1 + 0.618^2 \cdot ulp(1) [-1, 1] \varepsilon_2 
\end{cases} \]

\[ i = 2 : \begin{cases} 
    t_3 = [0.618^3, 0.618^3] + 2 \cdot 0.618^3 \cdot ulp(1) [-1, 1] \varepsilon_1 + 0.618^3 \cdot ulp(1) [-1, 1] \varepsilon_2 \\
    t_4 = [0.618^4, 0.618^4] + 2 \cdot 0.618^4 \cdot ulp(1) [-1, 1] \varepsilon_1 + 2 \cdot 0.618^4 \cdot ulp(1) [-1, 1] \varepsilon_2 
\end{cases} \]

\[ \begin{cases} 
    t_3 := t_3 \cup t_1 = [0.618^3, 0.618] + 0.618 \cdot ulp(1) \varepsilon_1 + 0.618^3 \cdot ulp(1) \varepsilon_2 \\
    t_4 = t_4 \cup t_2 = [0.618^4, 0.618^2] + 0.618^2 \cdot ulp(1) \varepsilon_1 + 0.618^2 \cdot ulp(1) \varepsilon_2 
\end{cases} \]

Convergence
Another Example

\[ y_n = \frac{5}{4} y_{n-1} - \frac{3}{8} y_{n-2} \]: if not enrolled, will pretend to be unstable.

Whereas if you unroll six times at least \((y_{n+6}\) as a function of \(y_n\) and \(y_{n-1}\) has all its coefficients strictly less than one): this is because:

\[
\begin{pmatrix}
    y_n \\
y_{n-1}
\end{pmatrix} =
\begin{pmatrix}
    3 & 1 \\
    4 & 2
\end{pmatrix}
\begin{pmatrix}
    \frac{3}{4} & 0 \\
    0 & \frac{1}{2}
\end{pmatrix}
\begin{pmatrix}
    3 & 1 \\
    4 & 2
\end{pmatrix}^{-1}
\begin{pmatrix}
    y_n \\
y_{n-1}
\end{pmatrix}
\]

So,

\[
\begin{pmatrix}
    y_n \\
y_{n-1}
\end{pmatrix} =
\begin{pmatrix}
    3 & 1 \\
    4 & 2
\end{pmatrix}
\begin{pmatrix}
    \frac{3^6}{4} & 0 \\
    0 & \frac{16}{2}
\end{pmatrix}
\begin{pmatrix}
    3 & 1 \\
    4 & 2
\end{pmatrix}^{-1}
\begin{pmatrix}
    1 & -\frac{1}{2} \\
    -2 & \frac{3}{2}
\end{pmatrix}
\begin{pmatrix}
    y_{n-6} \\
y_{n-7}
\end{pmatrix}
\]

Hence,

\[
\begin{pmatrix}
    y_n \\
y_{n-1}
\end{pmatrix} =
\begin{pmatrix}
    \frac{2059}{4096} & -\frac{1995}{8192} \\
    \frac{665}{1024} & -\frac{537}{2048}
\end{pmatrix}
\begin{pmatrix}
    y_{n-6} \\
y_{n-7}
\end{pmatrix}
\]
Our current unfolding

The union (or widening operator) is applied in a cyclic manner: if $F$ is the functional associated with the body of the loop, $N$ is the unfold number, we (try to) compute the lfp of the functional

$$
\begin{pmatrix}
X_1 \\
X_2 \\
\vdots \\
X_N
\end{pmatrix}
= \begin{pmatrix}
X_1 \cup F(X_N) \\
F(X_1) \\
\vdots \\
F(X_{N-1})
\end{pmatrix}
$$

(in fact, we do not exactly syntactically unfold the body of the loop)
Beyond cyclic unfolding: dynamic partitioning?

Idea: try to partition dynamically any “abstract trace”: unfold any lines (and corresponding $\epsilon$) and use dynamic change of abstract domains to determine “best” partitioning.

Suppose the iterates in the computation of $x$ (no unions) are:

\[
\begin{align*}
x^0 &= f^0 \epsilon + \omega_1^0 \epsilon_1 \\
x^1 &= f^1 \epsilon + \omega_1^1 \epsilon_1 + \omega_2^1 \epsilon_2 \\
\ldots &= \ldots \\
x^n &= f^n \epsilon + \omega_1^n \epsilon_1 + \ldots + \omega_n^n \epsilon_n + \omega_{n+1}^n \epsilon_{n+1}
\end{align*}
\]

(where $\epsilon_{i+1}$ is created by the $i$th instance of the computation of $x$)
Formally

For instance, look at the problem of changing the grain of the $\epsilon$ in a loop, to achieve quicker stability of the least fixed point computation.

For fixed $N$, the problem is to find a maximal (in the number of subsets) partition of $\{1, \ldots, N\}$, $P_1, \ldots, P_k$ such that we reach a postfixed point when projecting out to the corresponding abstraction:

$$\forall i, \sum_{j \in P_i} \omega_{j}^{n+1} \subseteq \sum_{j \in P_i} \omega_{j}^{n}$$

We conjecture this is NP-complete but has nice heuristics.
More generally

A possibility: try to find $n$ together with a cyclic partitioning:
Compute $x^0$ to $x^{2n-1}$; try to find the maximal partition such that the abstract criterion for getting to a postfixpoint is met:

$$\forall i, F_{P_i}(x^n, \ldots, x^{2n-1}) \subseteq F_{P_i}(x^0, \ldots, x^{n-1})$$

Are there optimal dynamic programming or greedy algorithms?

Similar to software pipelining (cyclic scheduling) in concurrency.
Some directions of research

- (From classical static analysis) Combine widenings with other analyses,

- (From classical numerical analysis)
  - Probabilistic methods?
  - Classical extrapolation methods (Aitken’s method in numerical analysis)?

- (From discrete control theory) Policy iteration algorithms?
An example

(1) x=0; (2)
    while (x < 100)
    (3) x++; (4)

Use interval abstraction, the abstract equation on (3) is:

\[ P_3 = \left[ -\infty, 99 \right] \cap (P_2 \cup P_4) \]
\[ = \left[ -\infty, 99 \right] \cap ([0, 0] \cup (P_3 \oplus [1, 1])) \]
\[ = F(P_3) \]
Without widening...

\[
\begin{align*}
X^0 &= \perp \\
X^1 &= [0, 0] \\
X^2 &= [0, 1] \\
X^3 &= [0, 2] \\
\cdots &= \cdots & \cdots \\
X^{100} &= [0, 99] \\
X^{101} &= X^{100}
\end{align*}
\]
With the usual widening...

\[ [a_1, b_1] W [a_2, b_2] = \begin{cases} 
\text{if } a_2 < a_1 \text{ then } -\infty & \text{otherwise } a_1, \\
\text{if } b_2 > b_1 \text{ then } +\infty & \text{otherwise } b_1
\end{cases} \]

It goes (if we apply it after two iterations for instance):

\[
\begin{align*}
X^0 & = \bot \\
X^1 & = [0, 0] \\
X^2 & = [0, 1] \\
X^3 & = [0, +\infty] \\
X^4 & = X^2
\end{align*}
\]
And with “narrowing M”...

\[ [a_1, b_1] M [a_2, b_2] = \begin{cases} 
    a_2 & \text{if } a_1 = -\infty \text{ otherwise } min(a_1, a_2), \\
    b_2 & \text{if } b_1 = +\infty \text{ otherwise } max(b_1, b_2) 
\end{cases} \]

\[
Y^0 = X^3 \\
Y^1 = [0, 99] \\
Y^2 = Y^1
\]
Other such widenings?

[Classical example: sign + interval]

\[ [a_1, b_1] W[a_2, b_2] = \begin{cases} 
0 & \text{if } 0 \leq a_2 < a_1 \\
-\infty & \text{else if } a_2 < a_1 \\
a_1 & \text{otherwise} \\
0 & \text{if } 0 \geq b_2 > b_1 \\
+\infty & \text{else if } b_2 > b_1 \\
b_1 & \text{otherwise} 
\end{cases} \]

Here combine with analysis of powers of two: get to the next power of two in each direction if it increases.
Policy iteration methods

- Newton like method, originally developed for computing eigenvalues of Markov transformations
- Any function on $R^n$ made of iterated $\min$ and $\max$ and "elementary functions" for which we know how to compute fixed points (here, arithmetic functions, for which we just solve systems of equations)

Work done in collaboration with Stéphane Gaubert (INRIA)
Policies

Historically, Howard in the 60s, case of functional $F : \mathbb{R}^n \to \mathbb{R}^n$ with:

$$F_i(x) = \max_{u \in U_i} (r_i^u + p_i^u x)$$

- Policies are maps $\pi : \{1, \cdots, n\} \to \bigcup_{1 \leq i \leq n} U_i$ such that $\pi(i) \in U_i$,
- To each $\pi$ we associate “derivative” $F^\pi$ with

$$F_i^\pi(x) = r_i^{\pi(i)} + p_i^{\pi(i)} x$$

In fact can generalize to other functions...
Principle of the algorithm

(0) Choose $\pi_0$ (random),

(1) Find (maybe by recursive application of the method)

$$x_0 = \text{lfp}(F^{\pi_0})$$

(2) Let $I' = \{i \mid 1 \leq i \leq n, F_i(x_0) > x_{0,i}\}$. Find another $\pi$ such that $\pi(i)$ for $i \in I'$ is different than $\pi_0(i)$ and loop on (1).

At most $n^{\max_i |U_i|}$ policies (but you can stop at any post-fixed point before...).
Partial Conclusion

- **Pragmatical** methods on simple numerical domains (like Monte-Carlo+simple widenings),

- New **promising** methods with policy iteration algorithms (work to be done for more general lattices).
  - Find a good elementary class of functions $F$.
  - Some work to be done for generalizing to other lattices.

- Complementary to indispensable **relational** approaches in our case.
First prototype “FLUCTUAT”

(see EG, MM, SP ESOP’02)

• For each scalar variable, presentation (last control point of the function under analysis) of:
  – Over-approximation (safe wrt the floating-point semantics) of the range of values of all known scalar variables,
  – Over-approximation of the imprecision of computation with respect to the “real” computation, decomposed by contribution of, at most, all C lines,
  – Higher-order errors (agglomerated - just a check!)

• Interprocedural analysis of a fragment of ANSI C; few library functions.
Presentation of the result
Some benchmarks

On a Pentium 4, 1GHz, 512 Mo [3rd version→4th version]:

<table>
<thead>
<tr>
<th>Program</th>
<th>Unfold</th>
<th>LOC</th>
<th>time (s) 3rd/4th V</th>
<th>Max mem (MB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>D8</td>
<td>100</td>
<td>547</td>
<td>201.93→113.9</td>
<td>199</td>
</tr>
<tr>
<td>D6</td>
<td>50</td>
<td>364</td>
<td>3.51→2.6</td>
<td>24</td>
</tr>
<tr>
<td>D9</td>
<td>1000</td>
<td>807</td>
<td>1102.39→47.94</td>
<td>136</td>
</tr>
<tr>
<td>prog</td>
<td>1</td>
<td>8796</td>
<td>107.22→111.98</td>
<td>316</td>
</tr>
</tbody>
</table>

(The need in memory could be reduced...with an expense in computation time)
Some improvements we are working on

- Unfolding and dynamic partitioning [experiments and theory well advanced]
- Relational analyses [in progress]
- Non-linear considerations (Lyapunov exponents - some experiments)
- New widenings
Related work

• CADNA, implementing the CESTAC method (probabilistic model of errors, dynamic analysis, first order)

• PRECISE (F. Chatelin) - backwards method (still probabilistic)

• Ameur et al. (for Aquarels, CNES): simple abstract interpretation

• New arithmetics (multiprecision, intervals, affine intervals, domain theory etc.)

• Specific algorithms (in algorithmic geometry for instance)