Reinhard Wilhelm

Patrick Cousot
cims.nyu.edu/~pcousot
pcousot@cims.nyu.edu

Farewell Colloquium on the Occasion of Reinhard Wilhelm's 68th birthday
Saarbrücken, November, 28th 2014

You said Reinhard Wilhelm?

But who is Reinhard Wilhelm?
But who is Reinhard Wilhelm?

1. You have understood the limitations of "Big data" and "Advanced machine learning"

2. This is THE Reinhard Wilhelm:

   Reinhard Wilhelm is a German computer scientist. Wikipedia
   Born: June 5, 1946 (age 68), Finntrop, Germany
   Education: University of Münster

...sorry, this was 2 months ago on Wikipedia, thanks to the true Reinhard Wilhelm for updating his picture last month!
There is only one, the proof is by Google

And more …

Great Achievements
Great Achievements of Reinhard (I)

Great Achievements of Reinhard (II)

Jun 24, 2015
Great Achievements of Reinhard (III)

Great Achievements of Reinhard (IV)

Not forgetting…Wilhelm Reinhard


Reinhard Wilhelm: GESAMT-Architektur, GI-Informatik-Jahrestagung, Giessen, April 1984, pp. 363-369


1974 - 1976

Reinhard Wilhelm: Graphische Benutzerschnittstellen (Grafik-Programme), Informationstechnik aktuell, Springer 1977, 17-30


1976


1977


Great sustainable productivity


Reinhard Wilhelm: GESAMT-Architektur, GI-Informatik-Jahrestagung, Giessen, April 1984, pp. 363-369


Reinhard Wilhelm: Graphische Benutzerschnittstellen (Grafik-Programme), Informationstechnik aktuell, Springer 1977, 17-30


1976


Reinhard Wilhelm: GESAMT-Architektur, GI-Informatik-Jahrestagung, Giessen, April 1984, pp. 363-369


Reinhard Wilhelm: Graphische Benutzerschnittstellen (Grafik-Programme), Informationstechnik aktuell, Springer 1977, 17-30


Too much to read so let’s have numbers

• The hit parade (even cited before published!):

• The flop:

An abstract machine for an object-oriented language with top-level classes
C Bösch, C Fecht, AV Hesse, R Wilhelm

yes, but cited 11 years before the pretend publication date!

Science
Main contributions

- Coming **number one in static analysis, world-wide**:

  ![Google Search Screenshot]

  - **Search Query**: label:static_program_analysis
  - **Search Results**

What is static analysis?

**Static program analysis**

From Wikipedia, the free encyclopedia

*Static program analysis* is the analysis of computer software that is performed without actually executing programs (analysis performed on executing programs is known as dynamic analysis).

at least the static analyzer must execute!

by a computer

The very first static analysis

**Brahmagupta** (Sanskrit: ब्रह्मगुप्त; listen (help-info)) (598–c.670 CE) was an Indian mathematician and astronomer who wrote two important works on Mathematics and Astronomy: the *Brahmasphutasiddhanta* (Extensive Treatise of Brahma) (628), a theoretical treatise, and the *Khanda khadyaka*, a more practical text.

![Brahmagupta Image]

- **Birth**: 598 CE
- **Death**: c.670 CE
- **Fields**: Mathematics, Astronomy
- **Known for**: Mathematics, Astronomy, Number system
The rule of signs by Brahmagupta (628)

18.30. [The sum] of two positives is positives, of two negatives negative;

• The abstraction is that you do not (always) need to known the absolute value of the arguments to know the sign of the result;
• Sometimes imprecise (don’t know the sign of the sum of a positive and a negative)
The rule of signs by Brahmagupta (628)

18.30. [The sum] of two positives is positives, of two negatives negative;

- The abstraction is that you do not (always) need to known the absolute value of the arguments to know the sign of the result;
- Sometimes imprecise (don’t know the sign of the sum of a positive and a negative)
- Useful in practice (if you know what to do when you don’t know the sign)
- e.g. in compilation: do not optimize (a division by 2 into a shift when positive*)

(*) Unless processor uses 2’s complement and can shift the sign.

18.32. A negative minus zero is negative, a positive [minus zero] positive; zero [minus zero] is zero. When a positive is to be subtracted from a negative or a negative from a positive, then it is to be added.

18.33. The product of a negative and a positive is negative, of two negatives positive, and of positives positive; the product of zero and a negative, of zero and a positive, or of two zeros is zero.
The rule of signs by Michel Sintzoff (1972)

For example, \( a \times a + b \times b \) yields the value 25 when \( a = 3 \) and \( b = -4 \), and when \( + \) and \( \times \) are the arithmetic multiplication and addition. But \( a \times a + b \times b \) yields always the object "pos" when \( a \) and \( b \) are the objects "pos" or "neg", and when the valuation is defined as follows:

- \( pos \times pos = pos \), \( pos \times neg = pos \), \( neg \times pos = pos \), \( neg \times neg = neg \).

The valuation of \( a \times a + b \times b \) yields "pos" by the following computations:

- \( V(a) = pos \), \( V(b) = pos \), \( V(a \times a + b \times b) = pos \times pos = pos \).

This valuation proves that the result of \( a \times a + b \times b \) is always positive and hence allows to compute its square root without any preliminary dynamic test on its sign. On the other hand, the

The rule of signs by Michel Sintzoff (1972)

For example, \( a \times a + b \times b \) yields the value 25 when \( a = 3 \) and \( b = -4 \), and when \( + \) and \( \times \) are the arithmetic multiplication and addition. But \( a \times a + b \times b \) yields always the object "pos" when \( a \) and \( b \) are the objects "pos" or "neg", and when the valuation is defined as follows:

- \( pos \times pos = pos \), \( pos \times neg = pos \), \( neg \times pos = pos \), \( neg \times neg = neg \).

The valuation of \( a \times a + b \times b \) yields "pos" by the following computations:

- \( V(a) = pos \), \( V(b) = pos \), \( V(a \times a + b \times b) = pos \times pos = pos \).

This valuation proves that the result of \( a \times a + b \times b \) is always positive and hence allows to compute its square root without any preliminary dynamic test on its sign. On the other hand, the
The rule of signs by Reinhard Wilhelm (2012/13)

**Idea:**
- We want to determine the signs of the values of expressions.
- For some sub-expressions, the analysis may yield \([-1, -0, -1, +1, 0, +1, +0, 0, -0, +0, -0, +1, -1, +1, +0, +0, 0, 0, 0] \), which means it couldn't find out.

We replace the concrete operators \( \oplus \) working on values by abstract operators \( \oplus \) working on signs.

The abstract operators allow to define an abstract evaluation of expressions:
\[
E : (V \mapsto \text{Signs}) \mapsto \text{Signs}
\]

Determining the sign of expressions in a Sign-environment works as follows:
\[
E[\oplus] \begin{cases} \text{if } c > 0 & \text{if } c = 0 & \text{if } c < 0 \\ \text{+} & \text{0} & \text{-} \end{cases}
\]

We want to determine the signs of the values of expressions.

Thus, we obtain the following effects of edges:
\[
\begin{array}{|c|c|}
\hline
\text{edge} & \text{effect} \\
\hline
1 & D \\
2 & \odot D = D \\
3 & \odot 0 \odot D = D \\
4 & \odot -D = \odot \odot 0 \odot D \\
5 & D 0 \odot 0 \odot D = \odot \odot -D \\
6 & D \odot 0 \odot -D = \odot \odot 0 \odot D \\
7 & \odot 0 \odot \odot 0 \odot \odot 0 \odot D = \odot \odot -D \\
\hline
\end{array}
\]

\[...\text{whenever } D \neq \perp \]

\[E[\oplus] \]

**Attention to details**

That's where you recognize a great scientist: make simple what is complicated!

Suggestions for an happy retirement
Have ambitious objectives!

- Move Dagstuhl close to an airport (or an airport close to Dagstuhl)

Remain active in science!

- Start working on cyberimbedded systems
Remain active in science!

- Start working on cyberimbedded systems
- Consider dynamic methods for static analysis
- Write a book on decompilation, by duality

Time for a serious conclusion
Thanks a lot for 30 years of friendship

Thanks a lot for 30 years of friendship, with lots of problems!

The End, thank you
The beginning, thank you

of retirement
The beginning, thank you