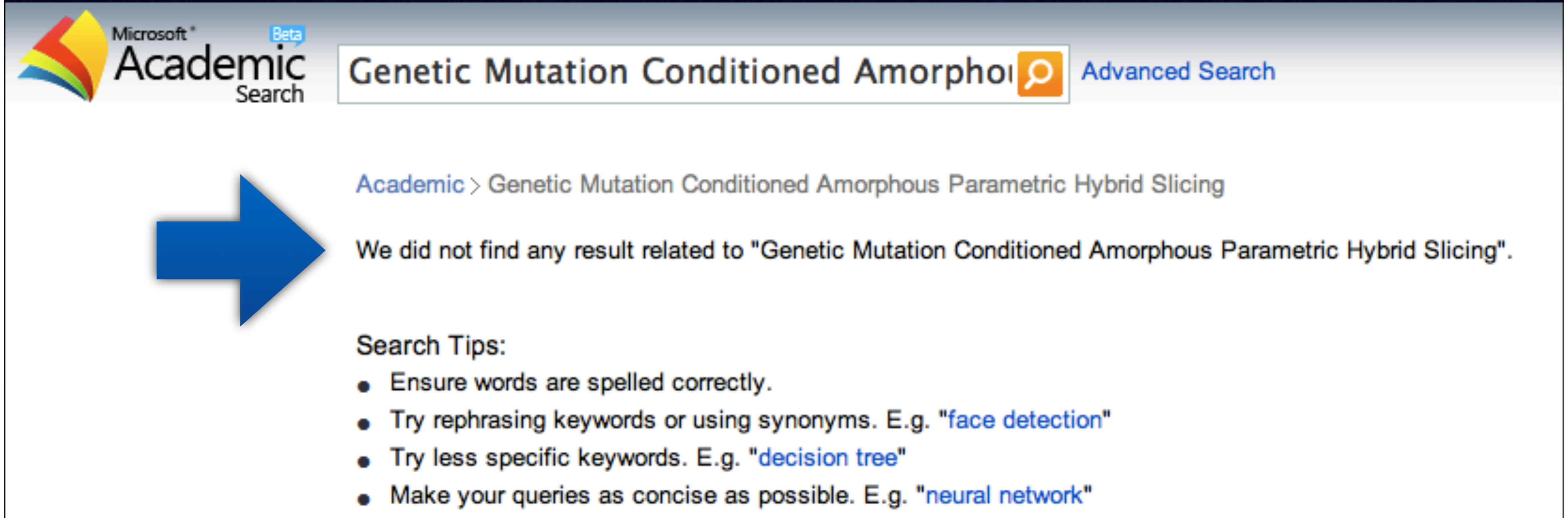
Genetic Mutation Conditioned Amorphous Parametric Hybrid Slicing

Jens Krinke
CREST, Department of Computer Science
University College London

London, 25/01/2011

What is it?

Genetic Mutation Conditioned Amorphous Parametric Hybrid Slicing


Microsoft Academic Search Beta
Genetic Mutation Conditioned Amorphou
Advanced Search

Academic > Genetic Mutation Conditioned Amorphous Parametric Hybrid Slicing

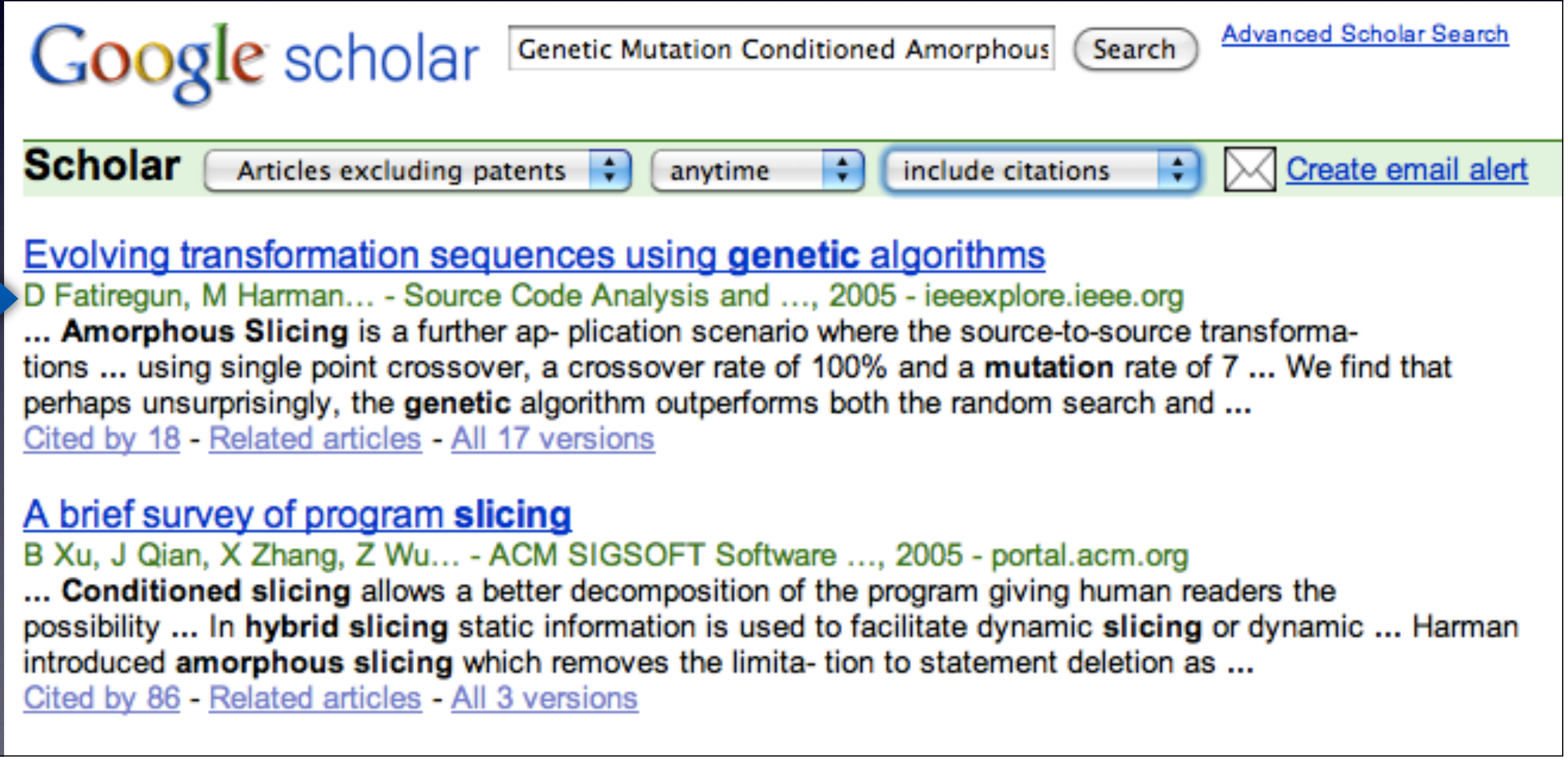
We did not find any result related to "Genetic Mutation Conditioned Amorphous Parametric Hybrid Slicing".

Search Tips:

- Ensure words are spelled correctly.
- Try rephrasing keywords or using synonyms. E.g. "face detection"
- Try less specific keywords. E.g. "decision tree"
- Make your queries as concise as possible. E.g. "neural network"

What is it?

Genetic Mutation Conditioned Amorphous Parametric Hybrid Slicing



Evolving transformation sequences using **genetic** algorithms
D Fatiregun, M Harman... - Source Code Analysis and ..., 2005 - ieeexplore.ieee.org
... **Amorphous Slicing** is a further ap- plication scenario where the source-to-source transforma- tions ... using single point crossover, a crossover rate of 100% and a **mutation** rate of 7 ... We find that perhaps unsurprisingly, the **genetic** algorithm outperforms both the random search and ...
Cited by 18 - Related articles - All 17 versions

A brief survey of program **slicing**
B Xu, J Qian, X Zhang, Z Wu... - ACM SIGSOFT Software ..., 2005 - portal.acm.org
... **Conditioned slicing** allows a better decomposition of the program giving human readers the possibility ... In **hybrid slicing** static information is used to facilitate dynamic **slicing** or dynamic ... Harman introduced **amorphous slicing** which removes the limita- tion to statement deletion as ...
Cited by 86 - Related articles - All 3 versions

100 Article Titles

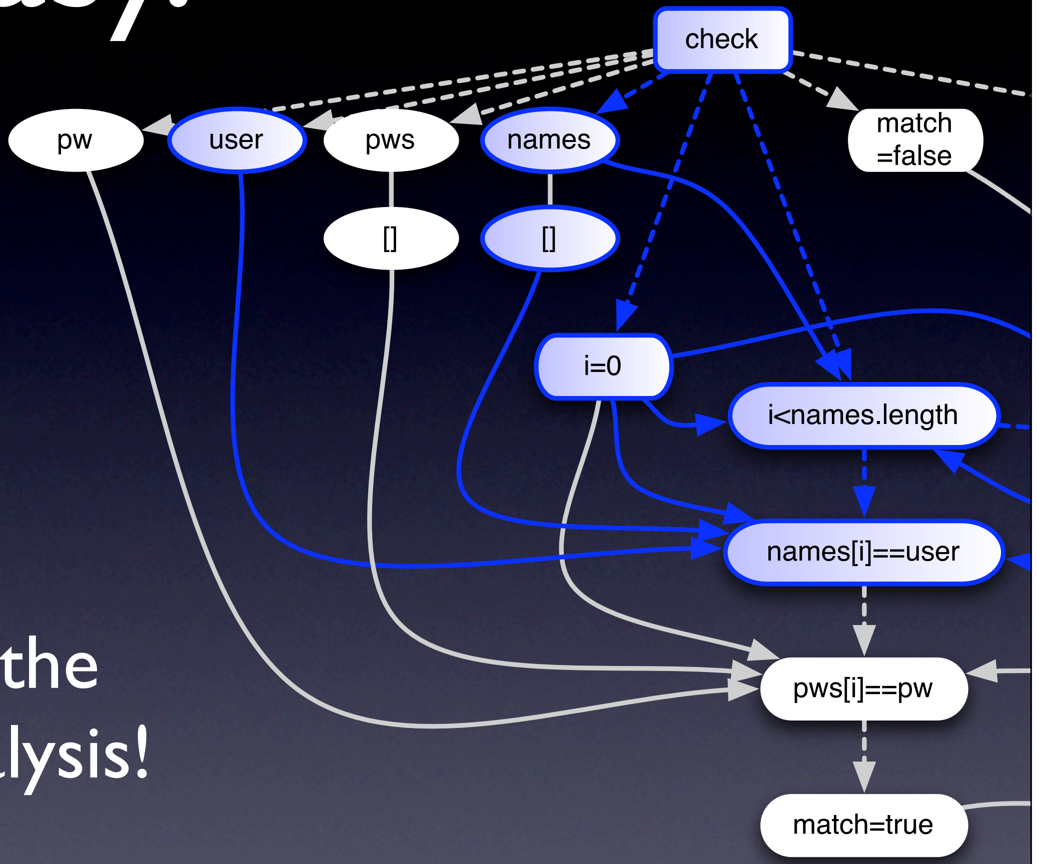
dynamic
debugging
using
application
analysis
based
techniques
understanding
comprehension
technique
evaluation
testing
survey
maintenance
approach
software
semantics
relationship
localization
dependence
algorithmic
identifying
specialization
abstract
automatic
description
regression
evaluating
hierarchical
classification
empirical
interpretation
checking
specification
function
bug
via
vhdl
test
tool
amorphous
effective
fault
lightweight
graphs
ripple
flow
study
generation
extraction
brief
static
mutants
hardware
atpg-based
integration
data
reusable
dependence-cache
framework
comments
descriptions
experimental
complexity
component
analyze
source
towards
simplify
overview
cost
role
model
executable
methods
provenance
improving
evolution
pointer
code
recovery
extracting
process
evaluate
system
beyond
comparative
source-level
denotational
engineering
traditional
faster
aspect
loop
transformation
theoretical
forward
effect
object-o
transformations
preconditions
execution
support
invariant
combination
points-to
behaviors
sigsoft
reverse
case
subsets
xml
fermat
unifying
relations
information
wcet
semantic
assignment
ansi
assisted
presence
node
explicitly
problems
algorithm
functions
applications
parametric
parallel
high
formalisation
location
composition
bound
ming
restructuring
oriented
eng
method
aspect-oriented
softw
new
specc

Where is Slicing used?

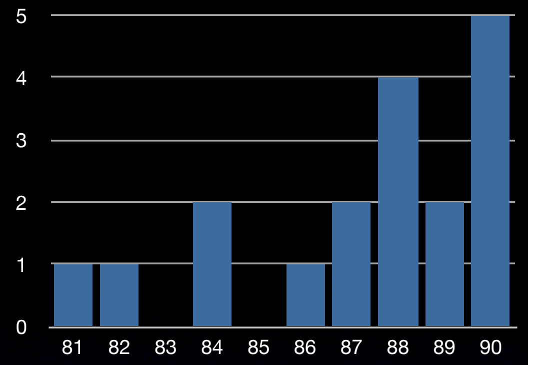
- Debugging:
  Which statements may have caused a fault?
- Evolution:
  What is a change's impact?
- Testing:
  Which tests have to be rerun?

Slicing is easy.

- Slicing is just a traversal of dependences.
- The hard part is the Dependence Analysis!



First 10 years



79, 81, 82, 84 - Mark Weiser's articles

84 - Slicing in Dependence Graphs

86 - Dicing

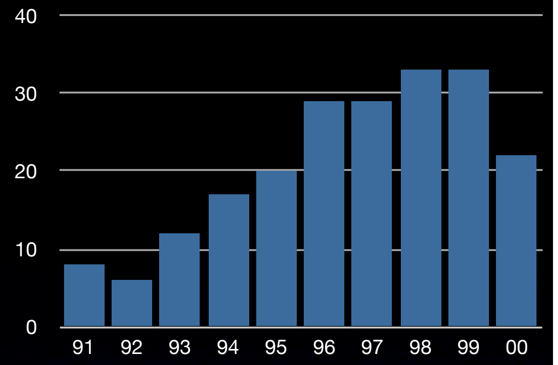
87 - Fault Localisation

88 - Dynamic Slicing

88 - Applications: Maintenance, Differencing

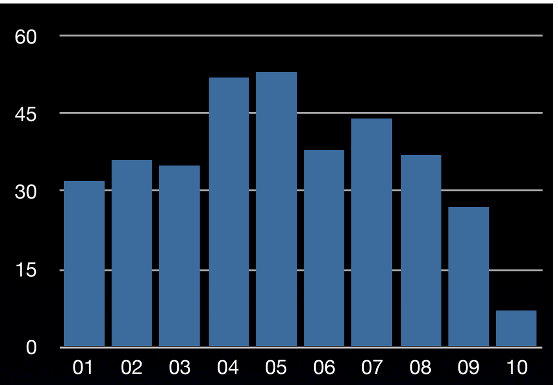
88 - Semantics

Busy 10 years



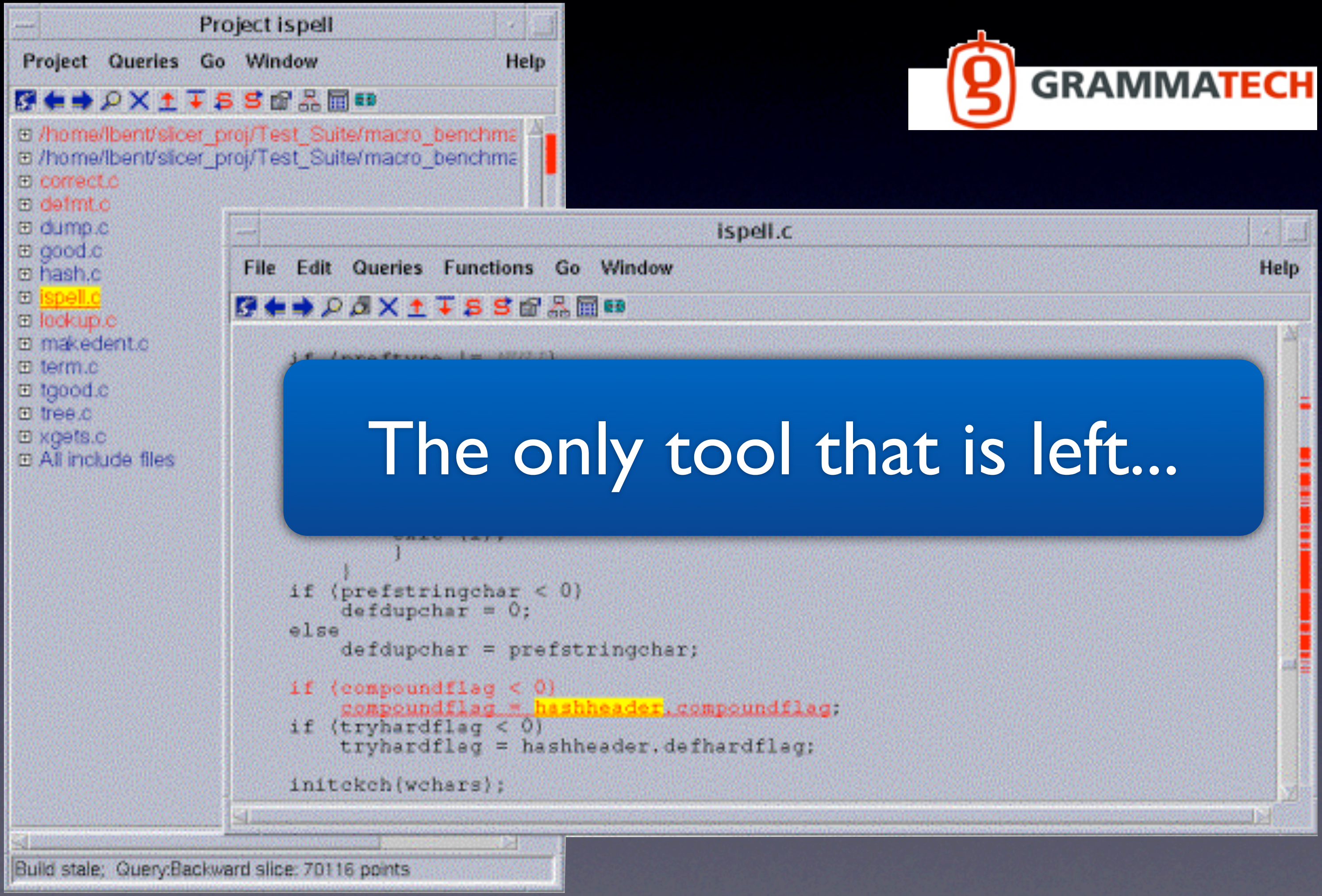
- 91 - Quasi-static slicing
- 92 - Testing
- 93 - Pointers
- 93 - Concurrency
- 93 - Specifications
- 93 - Functional Languages
- 93 - Function Extraction
- 94 - Chopping
- 94 - OOP
- 95 - Parametric Slicing
- 95 - Frank Tip's Survey
- 96 - Prolog
- 96 - VHDL
- 97 - Amorphous Slicing
- 98 - Conditioned Slicing
- 98 - State Machines

Stable 10 years



- Improvements in precision, efficiency, applications, usability, applicability, ...
- Empirical studies
- Tool(s): CodeSurfer and some prototypes (Kaveri, JSlice, Sprite, Unravel)

CodeSurfer


Project ispell
Project Queries Go Window Help
ispell.c
File Edit Queries Functions Go Window Help

```
/home/lbent/slicer_proj/Test_Suite/macro_benchma
/home/lbent/slicer_proj/Test_Suite/macro_benchma
correct.c
defmt.c
dump.c
good.c
hash.c
ispell.c
lookup.c
makedent.c
term.c
tgood.c
tree.c
xgets.c
All include files
```

```
    if (preftype != NULL)
            exit (1);
            }
        }
    if (prefstringchar < 0)
        defdupchar = 0;
    else
        defdupchar = prefstringchar;

    if (compoundflag < 0)
        compoundflag = hashheader.compoundflag;
    if (tryhardflag < 0)
        tryhardflag = hashheader.defhardflag;

    initckch(wchars);
```

Build stale; Query:Backward slice: 70116 points

The only tool that is left...

Two Events

2001:
William Griswold's Keynote at PASTE:
"Making Slicing Practical: The Final Mile"

2005:
Dagstuhl Seminar
"Beyond
Program Slicing"

Good News!

Most technical problems on Griswold’s list have been solved!

But…

Where are the Slicers?

- Program Slicers are *still* not widely used.
- Program Slicers are not matching
  the needs of software engineers.
- Slicers are too general and to complex.
- CodeSurfer's use in research:
  not the tool itself is used,
  but its infrastructure and its scripting API.

A Tool User

Your tool can solve all sorts of problems for us. But it'll have to analyse our entire 1 MLOC program, which is written in 4 languages and doesn't compile right now. I want the results as fast as compilation, with an intuitive graphical display linked to the source and integrated into our IDE. I want to save the results, and have them automatically updated as I change the program. By the way, I use Windows and some of my colleagues use Unix.

Griswold's Slide in 2001

Challenges
(see Wolfgang's talk)

- Distributed applications
- Exhaustive analyses are impossible, source code is not available or compilable.
- Systems programmed in various languages, including scripting and configurations.

Challenges
Who solves them?

- Almost no advances in the past 10 years!
- Academic research cannot solve these large scale challenges - too expensive.
- Only industrial research can solve them, if a paying client has a specific problem.

Slicing in 2011

- Slicing has been replaced by Dependence Analysis.
- Many new techniques use some kind of dependence traversal adapted to their specific needs.
- Dependence is measured and ranked, binary information is insufficient.

Current Applications Static

- **Slicing Models**
  State Machines, UML, etc.
- **Security and Information Flow**
  Taint Analysis, Non-Interference

Current Applications
Dynamic

- **Fault Localization**
  Slicing vs. Tracing
- **Impact Analysis**
  Slicing vs. Tracing

Current Trends

- Analyses are neither sound nor complete, optimistic instead of conservative.
- Dynamic Analyses (Tracing)
- Abstract Interpretation
- Symbolic Execution
- Transformation instead of just analysis

Conclusions

- Technical problems to slice simple systems are all solved.
- Complex systems are still a challenge!
- Research is (and should be) task driven
- Dynamic analyses like Tracing are used instead of language-based Slicing.
- Dependence Analysis, not Slicing, is established, often used, and successful.