The Goblin

Vesal Vojdani
University of Tartu

What is the Goblin?

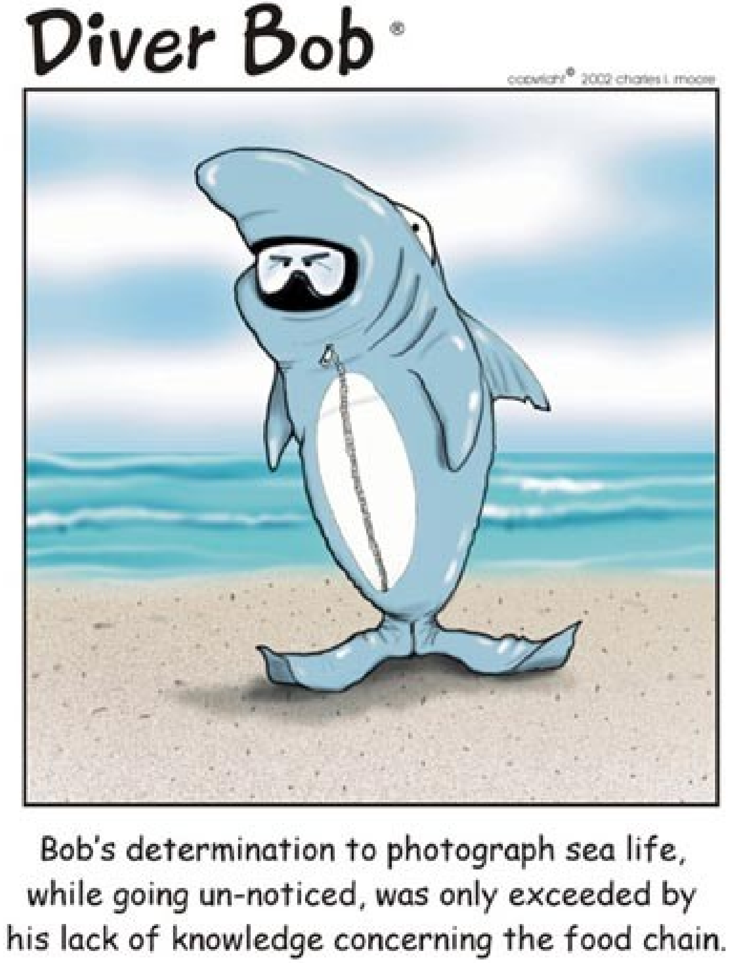
- General
  - A general analysis framework
- O'Caml
  - Analyses are written in Objective Caml
  - It analyzes C code
- Brogram
  - Uhm, well in Estonian bank is written pank
- Linter
  - Such as splint

Meet Alice

- She writes compilers
- She knows the semantics of her language very well
- She can check lots of things with her compiler
- Because she does lots of static analysis
- She is happy :)

Introducing Driver Bob

- Bob works on device drivers
- His code must satisfy properties that Alice is unaware of
- Bob does not know how to write compilers
- He can't check his code
- He is unhappy :(

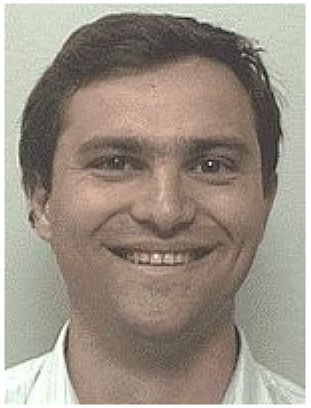

Diver Bob®
copyright © 2002 charles l. moore
Bob's determination to photograph sea life, while going un-noticed, was only exceeded by his lack of knowledge concerning the food chain.

Fulfilling Customer Needs

How can Alice please Bob?

Solution: Meta-Compilation!

- In particular
  - Coverity Prevent™
  - Based on research at $tanford
- In general
  - Static Analysis for the millions
  - Let Bob specify the analysis
- Meanwhile at Berkeley: CIL
  - It is open source
  - But industry strength

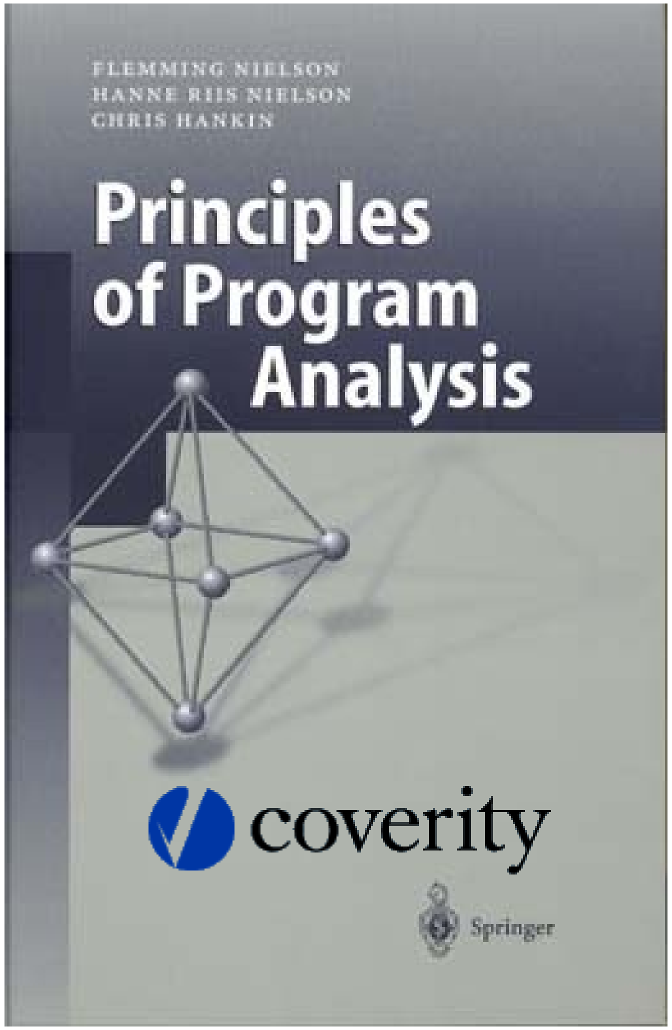


Alice and Bob finally meet

```
Alice> You can write compiler extensions
  in this cool language MetaL

Bob> How?

Alice> You know Finite State Automata?

Bob> Yeah sort of...

Alice> Good, you're all set.

Bob> OK

Cousot> Is it sound?

Engler> Not important!
```

Sound or unsound?

- Approach / philosophy
  - Bug detection
  - Software verification
- Practical analysis result
  - As many bugs as possible
  - All bugs of type X
- Can we have both???
  - As expressive as Engler
  - Almost as sound as Cousot



What bugs me about PAG

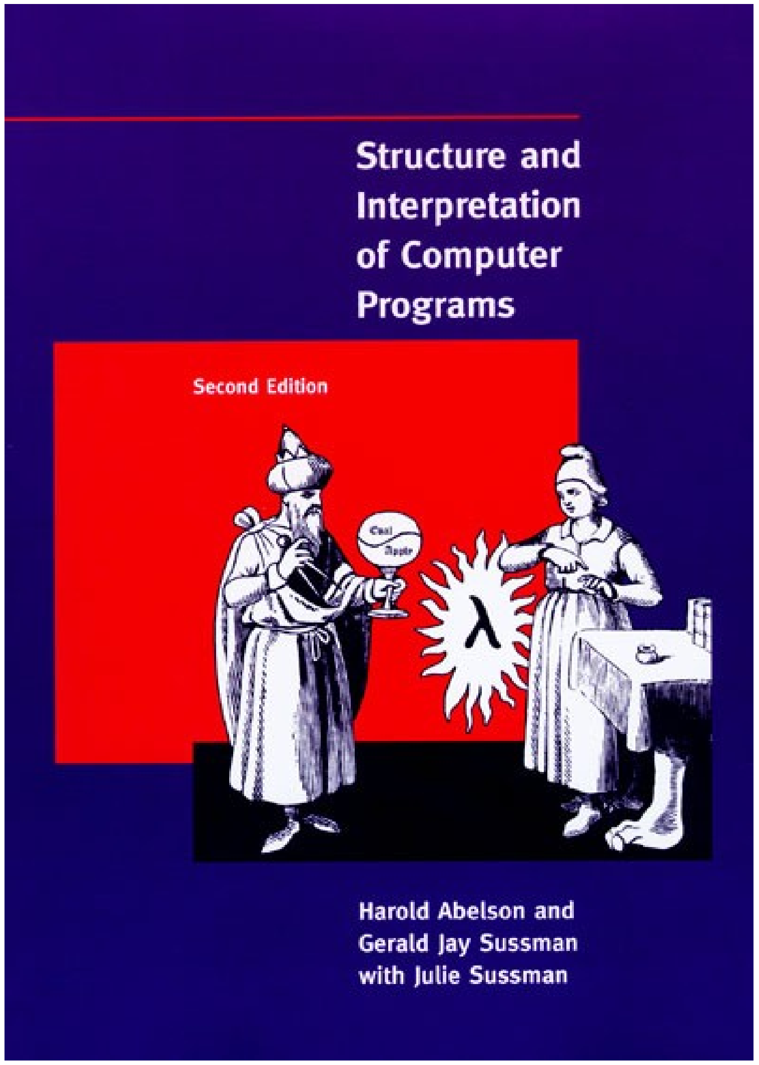
- Let's start from a sound framework
- PAG is an example of a sound program analysis framework
- You specify the abstract domain and transfer functions
- Out comes an efficient analyzer!
- BUT when the primitives are not enough, you have to program in C

Pluggable domains!

- A very nice idea by Cooprider and Regehr (cXprop)
- Use a high level language like O'Caml
- Alice specifies the domain interface
- Bob plugs in his domain and there you go
- Except Bob has to write a lot of O'Caml code
- Can we combine the ideas?
  - The ease of PAG
  - The freedom of Pluggable domains

Remember the 80s?

- Recall Abelson and Sussman's meta-linguistic abstraction
  - Layers of progressively domain-specific facilities
  - The layers are transparent
- Why do we love the Domain Specific Languages hosted by Haskell?
  - Because they're still Haskell!


Structure and Interpretation of Computer Programs
Second Edition
Harold Abelson and Gerald Jay Sussman with Julie Sussman

The Goblin DSLs

- Like PAG
  - Goblin Domain Definition Language
  - Goblin Transfer Function Language
- Towards MetaL
  - Goblin Analysis Patterns
  - Goblin Analysis Transformers
- If only my development team was as efficient as my marketing department ...

Goblin Domain Definition Language

- The GDDL is hosted by the O'Caml module system
- Functors are wonderful *(when they work)*
- The syntax is very similar to PAG's DATLA
- A simple interval domain can be specified as

```
Product (Reverse (Lift (Integers)))
        (Lift (Integers))
```

Goblin Transfer Function Language

- The GTFL is used to specify the effect of C expressions on the state
- You can analyze all of C (mainly thanks to CIL) by giving definitions for:
  - Assignments
  - Simple branches
  - Function calls
- The best part:
  GTFL is nothing else than O'Caml itself!

Goblin Analysis Patterns

- GAPs create analyses from very simple definitions
- Like the cXprop functor
  - takes an abstract value domain X
  - create a conditional X propagation analysis
- Some other common patterns are
  - Operation A must always precede B
  - All functions that do A must also do B before returning

Goblin Analysis Transformers

- GATs are functors that combine entire analyses
- The hottest one takes two analyses as input
  - Some form of constant propagation, usually Goblin's built in analysis
  - Simple user supplied analysis X on a finite domain
- The result is an as-path-sensitive-as-necessary X analysis!
- *It would be interesting to put Peter & Ilja into that functor!*

Alice and Bob meet again

Alice> You can write user analyses using this cool framework called the Goblin!

Bob> How?

Alice> Well, it's like Lego!

Bob> Oh, I love Lego.

Alice> Good, you're all set.

Bob> OK, thanks!

Varmo> Does it work?

Vesal> Yes, it will!

Goblin's current features

- Heavy base analysis
  - Conditional Constant Propagation with point-to analysis (intertwined)
  - Uses the Trier value domain (good for case expressions)
  - Granular structs and arrays
- Interprocedural multithreaded analysis
  - Functional approach
  - The Trier approach to multithreading
  - Data Race Analyzer

Goblin Implementation

Recursive

ML Module
Mania

Conclusions: Work

- Passed work
  - Why don't people program in the While language?
- Present work
  - Make things work!
  - Make things look good!
  - Make the source code look good!
- Future work
  - Statistical post-processing seems interesting
  - Try some original ideas on the goblin