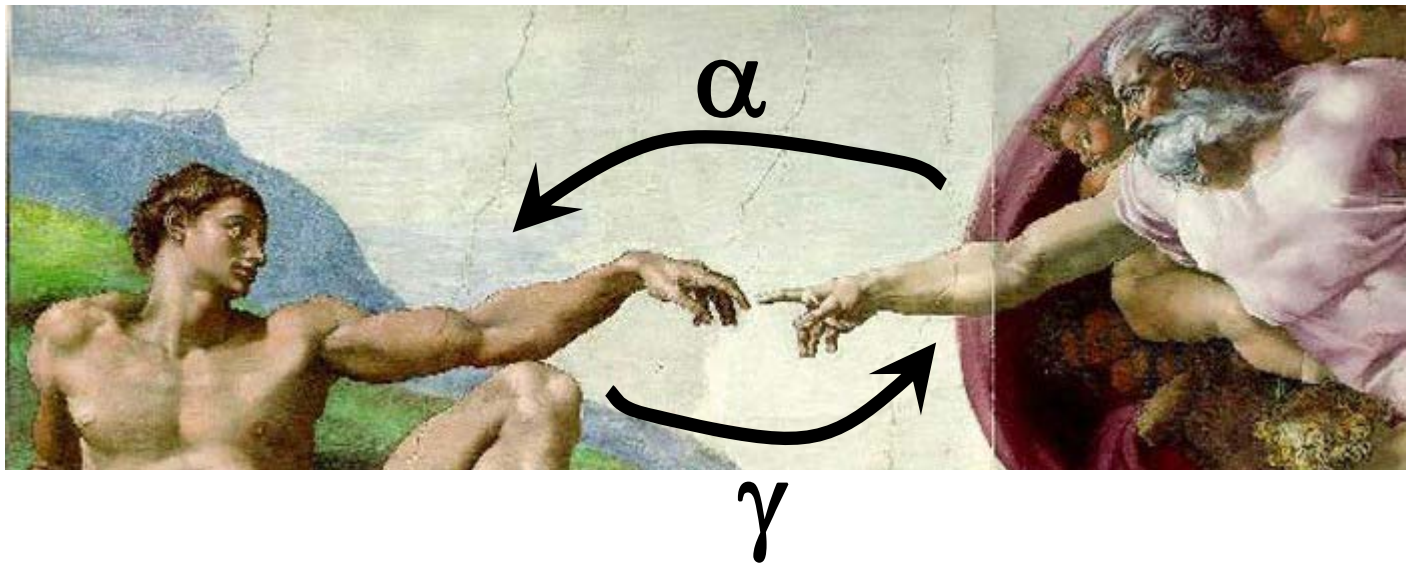
Program Transformation, Semantic Modelling and AI
A Personal Perspective

David Sands

Chalmers University of Technology

Sweden

15YNAAI



The Imperial Chapter

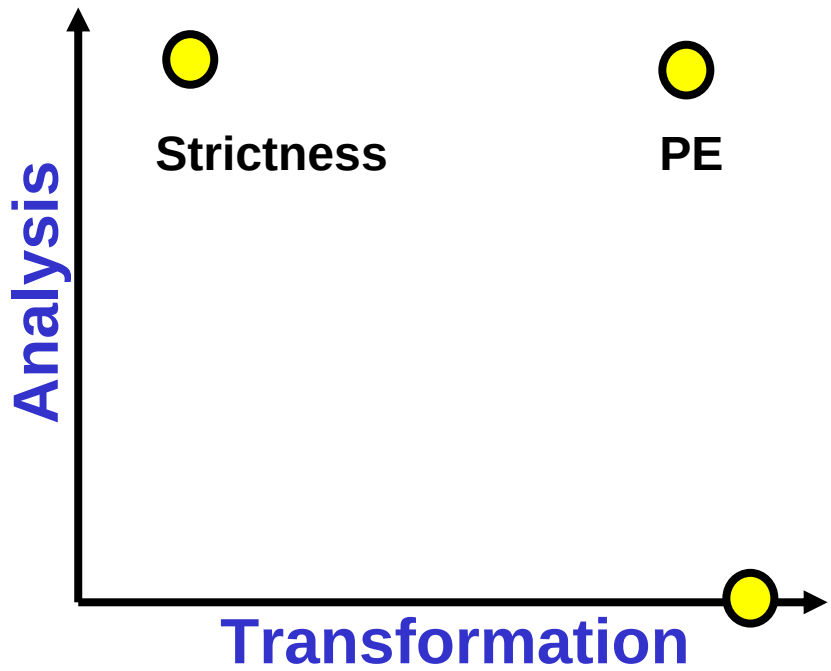
Program Transformation is cool, I think I'll try and do a PhD…

Imperial AI (Hankin, Burn, Abramsky)

- Interesting AI (strictness)
- The EU BRA "Semantique"

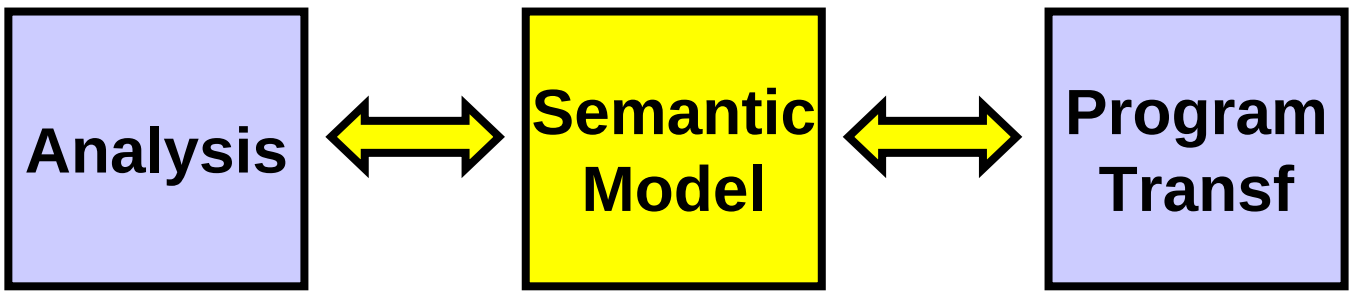
The DIKU Chapter

- Time to focus on interesting transformations (UF, deforestation,…).
- Decided to look more at Partial Evaluation



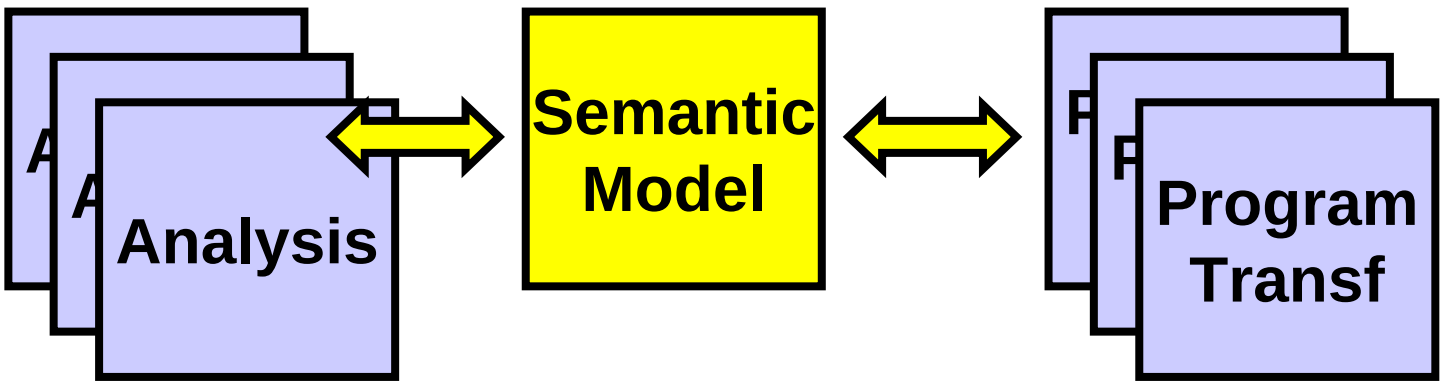
How to Prove Correctness of PT

- Monolithic approach
  - "The analysis is correct if the transformation works!" [Mitch Wand]
- Model-based Approach

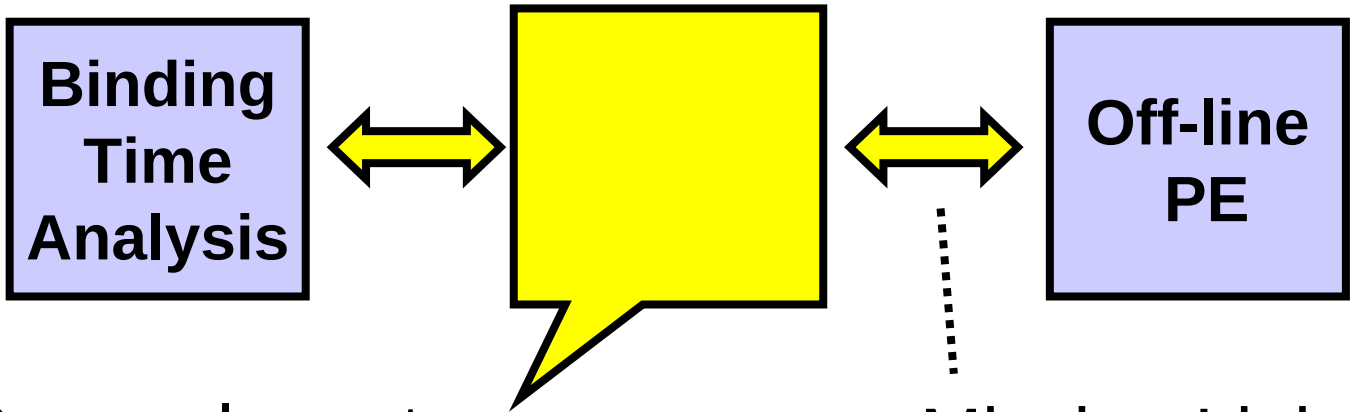


How to Prove Correctness of PT

- Monolithic approach
  - Modularity: independent variation of components
  - "The analysis is correct if the transformation works!" [Mitch Wand]
  - AI arguments: construction, combination
- Model-based Approach



Off-line Partial Evaluation



- Some elegant models of binding times
  - Domain projections [Launchbury,89]
  - PERS [Hunt90,HS91]
- Missing Link between models and PE

The missing link

- …Is broken! The model was not correct
  - semantically sound binding times could make the PE go wrong!

The missing link

- A new model

"A Semantic Model of Binding Times for Safe Partial Evaluation" [Henglein&Sands '95]

- Technical highlights:
  - Dealing with program annotations without "sticky" semantics
  - A "type" system parameterised by an AI

The Chalmers Chapter

- Haskell compilers packed with program transformations enabled by static analyses
- An example
  - Inlining: let x = M in N $\rightarrow$ N[x:=M]
  - trivially “correct” - but only used when it is believed to be beneficial…

Work-Space Safety of Inlining

- Correctness = work & space safety wrt a space and time semantics
- Analysis: usage analysis

  "this binding will be used at most once"
- Many instances but nothing even close to a safety proof for inlining
  - hint at why it is hard: $x + y \rightarrow y + x$ …

The missing link…

…is partly broken

- Semantic guarantees provided by several systems need to be strengthened to guarantee space-safe inlining
- Some usage analyses are unsafe for inlining
  - the analysis can't be smarter than the GC when it comes to the meaning of "live"

  [A Foundation for Space-safe transformations of call-by-need, Gustavsson& Sands '99]

Lesson learned

For program transformation, semantic modelling is great, but

we need the *right* models!

Challenge

- Top down design starting with the intended transformation rather than what we *hope* is the right semantic property
- Can we constructively build analyses from the transformations rather than the concrete-abstract domains?

Further Challenges

- Security/safety increasingly important
- The role of AI and Security analysis
  - e.g. Giacobazzi & Mastroeni: AI as a generalized noninterference framework;

    Hunt & Sands: used AI to understand flow-sensitive security type systems
- Security models get complex
  - but often there is no formal way to validate the correctness of the models

Conclusion

"Ideas are spread by people not papers"

- PC

Thanks to

- the AI community (in particular ESPRIT BRA "Semantique" 1989-2002)
- Coauthors Seb Hunt, Fritz Henglein, Jörgen Gustavsson, Andrei Sabelfeld