Thirty Years of Abstract Interpretation
A UK Perspective

Chris Hankin

Department of Computing, Imperial College London

San Francisco, January 2008

Outline

Outline

Outline

Outline

Outline

Some History

1981 Mycroft – Applicative Programs

1984 Nielson – Domain Theory

1985 Canterbury workshop (Higher-order Strictness Analysis)

1987 Abstract Interpretation of Declarative Languages published

1987 Hughes and Wadler – Projections

1989 First Semantique project

Some History – continued

1991 PEPM launched

1991 Hunt – PERS

1992 Semantique 2 Working Group

1994 SAS launched

1999 Principles of Program Analysis published

2001 Parma Polyhedral Library launched

Outline

Foundations

- Domains (Burn, Hill, King, Mellish, Mycroft, Wadler)
- Projections and PERS (Hughes, Hunt, Wadler)
- Stone Duality – Strictness Types (Jensen, Benton)
- Game Semantics (Malacaria, Hankin)
- Quantitative Analysis (Clark, Hankin, Hunt, Malacaria, Wiklicky)

Algorithms

- Frontiers Algorithms (Clack, Peyton-Jones, Hankin, Hunt, Martin) – Approximation methods (Hankin, Hunt)
- Minimal Function Graphs (Mycroft, Rosendahl)
- Lazy Types (Hankin, Le Métayer)
- Polyhedra (Hill, King)
- BDDs (King)

Applications

Functional: Strictness Analysis; Binding Time Analysis; Escape Analysis; Control Flow

Logic: Groundedness Analysis; Dependency Analyses; Circularities – Occurs Check avoidance; Type Analysis

Imperative: Pointers; Numerical analyses

Object-Oriented: Control Flow; Escape Analysis; Security

Language Based Security: Confinement; Protocol Analysis; Buffer Over-runs

Applications

Functional: Strictness Analysis; Binding Time Analysis; Escape Analysis; Control Flow

Logic: Groundedness Analysis; Dependency Analyses; Circularities – Occurs Check avoidance; Type Analysis

Imperative: Pointers; Numerical analyses

Object-Oriented: Control Flow; Escape Analysis; Security

Language Based Security: Confinement; Protocol Analysis; Buffer Over-runs

Applications

Functional: Strictness Analysis; Binding Time Analysis; Escape Analysis; Control Flow

Logic: Groundedness Analysis; Dependency Analyses; Circularities – Occurs Check avoidance; Type Analysis

Imperative: Pointers; Numerical analyses

Object-Oriented: Control Flow; Escape Analysis; Security

Language Based Security: Confinement; Protocol Analysis; Buffer Over-runs

Applications

Functional: Strictness Analysis; Binding Time Analysis; Escape Analysis; Control Flow

Logic: Groundedness Analysis; Dependency Analyses; Circularities – Occurs Check avoidance; Type Analysis

Imperative: Pointers; Numerical analyses

Object-Oriented: Control Flow; Escape Analysis; Security

Language Based Security: Confinement; Protocol Analysis; Buffer Over-runs

Applications

Functional: Strictness Analysis; Binding Time Analysis; Escape Analysis; Control Flow

Logic: Groundedness Analysis; Dependency Analyses; Circularities – Occurs Check avoidance; Type Analysis

Imperative: Pointers; Numerical analyses

Object-Oriented: Control Flow; Escape Analysis; Security

Language Based Security: Confinement; Protocol Analysis; Buffer Over-runs

Outline

Cambridge (Mycroft)

- Program analysis and compilation techniques applied to hardware
- Semantics and compilation using continuations
- Inverse program analysis – extracting program corrections from program analysis inconsistencies

City, Kings and Queen Mary (Hunt, Clark and Malacaria)

- Quantitative Information Flow – language based security using Shannon's Information Theory
- Flow-sensitive security types (with Sands)
- PERs and Abstract Non-Interference (with Mastroeni)

Kent (King)

- Backward analysis for reasoning about legacy code, concurrency, determinacy and verification
- Representation of Boolean functions for groundedness analysis, dependency analyses and Reduced-Ordered BDD approximation techniques
- Security issues and buffer-overun
- Polyhedral techniques

Leeds (Hill)

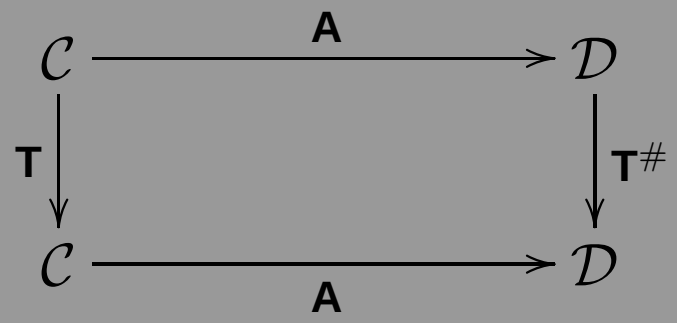
- A Library of Polyhedra – free software distributed under the terms of the GNU General Public License
- Domain Construction and Grids for Software Analysis
- Analyzing the Natural Semantics of Programs
- Analyzing Object-Oriented Languages
- Numerical Stability of software – detecting numerical instability in software caused by the approximate binary or floating point representations of numbers.

Imperial (Di Pierro, Hankin and Wiklicky)

- Analysis of protocols for ad hoc wireless networks (with Nanz)
- Pointer analysis for C (with Pearce and Kelly)
- Probabilistic Abstract Interpretation (with Di Pierro and Wiklicky)

Probabilistic Abstraction

Consider a Concrete Domain $\mathcal{C}$ and an Abstract Domain $\mathcal{D}$:

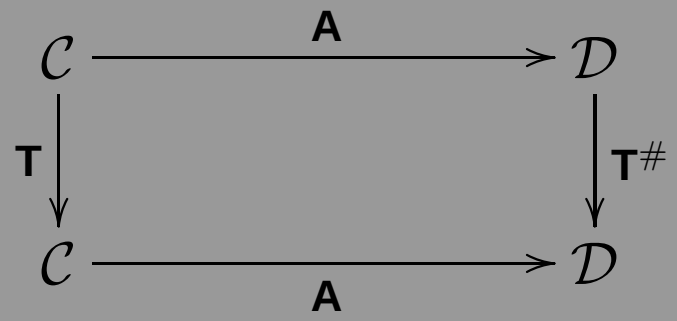


With an abstraction $\mathbf{A} : \mathcal{C} \to \mathcal{D}$ and a concretisation $\mathbf{G} : \mathcal{D} \to \mathcal{C}$:

$$\mathbf{T}^{\#} = \mathbf{GTA}$$

Probabilistic Abstraction

Consider a Concrete Domain $\mathcal{C}$ and an Abstract Domain $\mathcal{D}$:



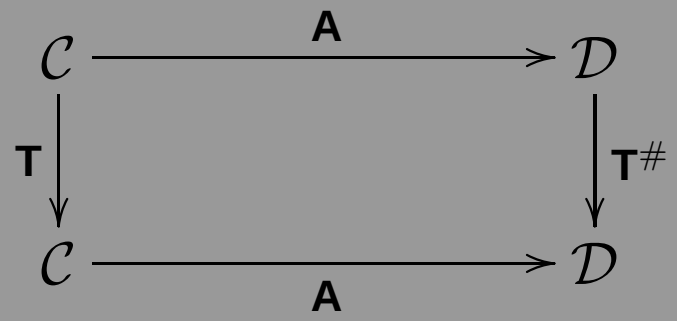
With an abstraction $\mathbf{A} : \mathcal{C} \to \mathcal{D}$ and a concretisation $\mathbf{G} : \mathcal{D} \to \mathcal{C}$:

$$\mathbf{T}^{\#} = \mathbf{GTA}$$

Abstract Interpretation: $(\mathbf{A}, \mathbf{G})$ form a Galois Connection.

Probabilistic Abstraction

Consider a Concrete Domain $\mathcal{C}$ and an Abstract Domain $\mathcal{D}$:



With an abstraction $\mathbf{A} : \mathcal{C} \to \mathcal{D}$ and a concretisation $\mathbf{G} : \mathcal{D} \to \mathcal{C}$:

$$\mathbf{T}^{\#} = \mathbf{GTA}$$

Probabilistic Abst.Int.: $(\mathbf{A}, \mathbf{G})$ Moore-Penrose Pseudo-Inverse.

Moore Penrose Pseudo-Inverse

Definition

Let $\mathcal{C}$ and $\mathcal{D}$ be two Hilbert spaces and $\mathbf{A} : \mathcal{C} \to \mathcal{D}$ a bounded linear map. A bounded linear map $\mathbf{A}^\dagger = \mathbf{G} : \mathcal{D} \to \mathcal{C}$ is the Moore-Penrose pseudo-inverse of $\mathbf{A}$ iff

(i) $\mathbf{A} \circ \mathbf{G} = \mathbf{P}_A$,

(ii) $\mathbf{G} \circ \mathbf{A} = \mathbf{P}_G$,

where $\mathbf{P}_A$ and $\mathbf{P}_G$ denote orthogonal projections onto the ranges of $\mathbf{A}$ and $\mathbf{G}$.

An (orthogonal) projection is an operator $\mathbf{P}$ with $\mathbf{P}^* = \mathbf{P} = \mathbf{PP}$.

Moore Penrose Pseudo-Inverse

Definition

Let $\mathcal{C}$ and $\mathcal{D}$ be two Hilbert spaces and $\mathbf{A} : \mathcal{C} \to \mathcal{D}$ a bounded linear map. A bounded linear map $\mathbf{A}^\dagger = \mathbf{G} : \mathcal{D} \to \mathcal{C}$ is the Moore-Penrose pseudo-inverse of $\mathbf{A}$ iff

(i) $\mathbf{A} \circ \mathbf{G} = \mathbf{P}_A$,

(ii) $\mathbf{G} \circ \mathbf{A} = \mathbf{P}_G$,

where $\mathbf{P}_A$ and $\mathbf{P}_G$ denote orthogonal projections onto the ranges of $\mathbf{A}$ and $\mathbf{G}$.

An (orthogonal) projection is an operator $\mathbf{P}$ with $\mathbf{P}^* = \mathbf{P} = \mathbf{PP}$.

Moore-Penrose Pseudo-Inverse II

Definition

An operator $\mathbf{A} \in \mathcal{B}(\mathcal{H})$ is Moore-Penrose invertible if there exists an element $\mathbf{G} \in \mathcal{B}(\mathcal{H})$ such that:

(i) $\mathbf{AGA} = \mathbf{A}$,

(ii) $\mathbf{GAG} = \mathbf{G}$,

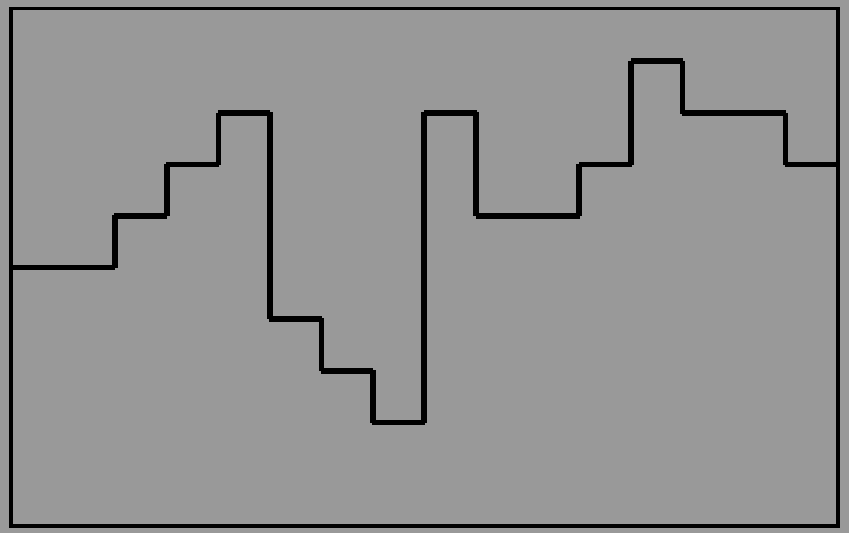
(iii) $(\mathbf{AG})^* = \mathbf{AG}$,

(iv) $(\mathbf{GA})^* = \mathbf{GA}$.

If it exists $\mathbf{G} = \mathbf{A}^\dagger$ is called Moore-Penrose pseudo-inverse.

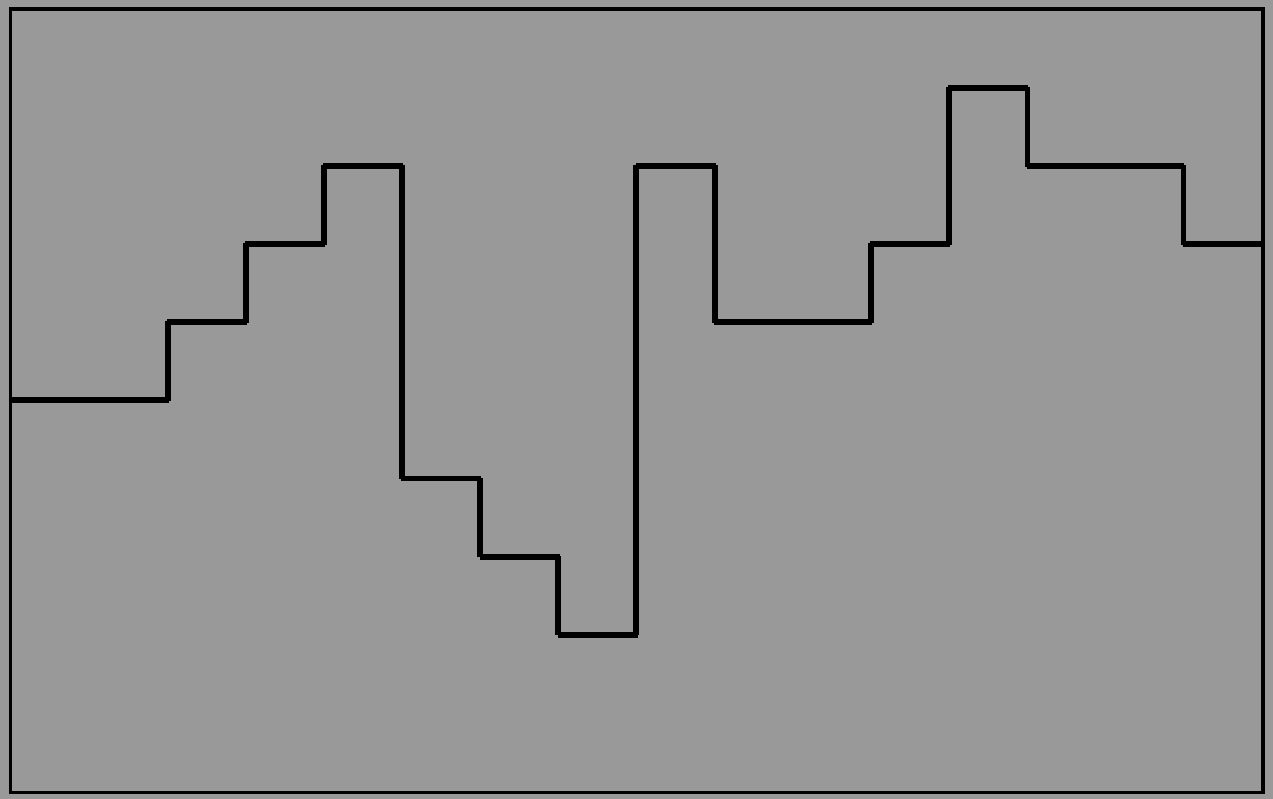
Example: Function Approximation

Concrete and abstract domain are step-functions on $[a, b]$.
The set of (real-valued) step-function $\mathcal{T}_n$ is based on the sub-division of the interval into $n = 16, 8, 4, 2, 1$ sub-intervals.
Each step function in $\mathcal{T}_n$ corresponds to a vector in $\mathbb{R}^n$.

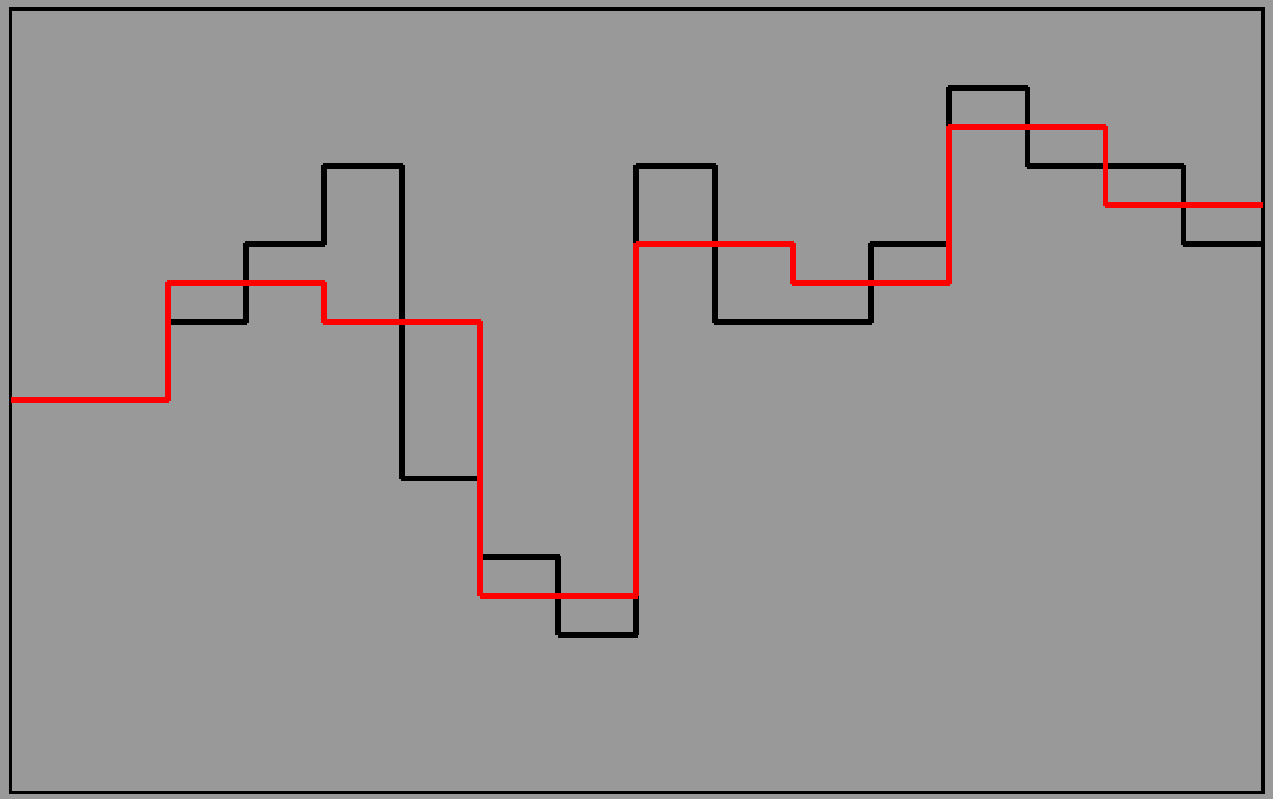


$$\begin{pmatrix} 5 & 5 & 6 & 7 & 8 & 4 & 3 & 2 & 8 & 6 & 6 & 7 & 9 & 8 & 8 & 7 \end{pmatrix}$$

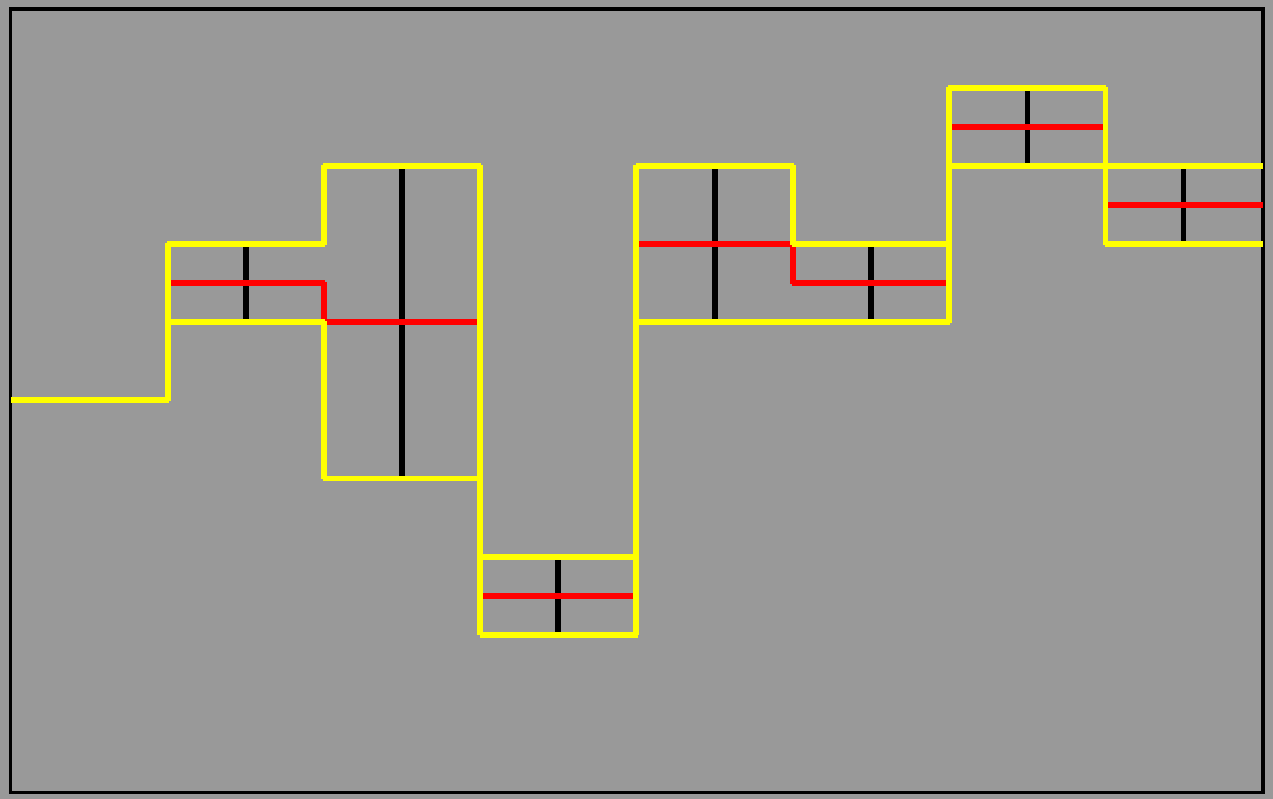
Example: Geometric Comparison



Example: Geometric Comparison



Example: Geometric Comparison



Applications

- Approximate confinement
- Virus propagation in distributed systems
- Padding to remove timing leaks
- Probabilistic pointer analysis

Outline

Congratulations to Patrick on 30 years of Abstract Interpretation!

The End