Program Schemes: Early results

I. Guessarian

Workshop on Higher-Order Recursion Schemes & Pushdown Automata

OUTLINE

- Origins
- Algebraic semantics
- Classes of interpretations

Origin

IANOV 1958

The logical schemes of algorithms

Inspired two mainstreams of research

- Category Theory (first paper Elgot 1968? 1970?)
- Logic and Universal Algebra (first paper Scott 1969? 1971?)

Algebraic and Category theory approach

Elgot(1971) Algebraic Theories and Program Schemes

Semantics of Loop-Free diagrams using categories

- Algebraic Theories
- semantics of iterative and recursive program schemes
- unique fixed points in initial algebras

ADJ group: Goguen Thatcher Wagner Wright, Elgot, Bloom, Meseguer, Tindell, Esik, . . .

Ianov Flow Diagrams

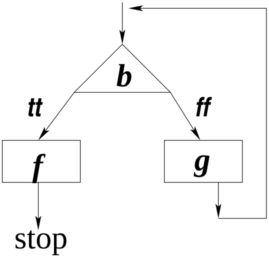


Figure: A Flow Diagram

Semantics

– operational
(computation rule) ??

– mathematical ??

Mathematical semantics

Algebraic, Denotational, Fixpoint,...
Goals:

- **give precise meaning to programs**
- **prove equivalences and properties of programs**

Decisive step: Scott (1971) The Lattice of Flow Diagrams

Algebraic meaning for iterative program schemes à la Ianov

Semantics

- complete lattice of (finite and infinite) flow diagrams
- meaning of a program (with iterations): a **least fixed point**

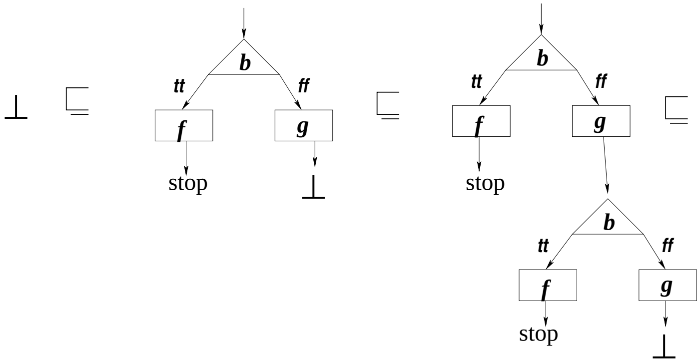


Figure: Meaning of a flow diagram (unfoldings)

Fixed point theorems

Theorem (Knaster, Tarski, Kantorovich, ...?)

A monotone function $F\colon D \longrightarrow D$ on a complete lattice D has a least fixed point $\mu x.f(x) = \inf\{x | f(x) \leq x\}$.

More appropriate: $(D, \leq)$ Scott domain

- every directed set has a least upper bound
- every bounded set has a least upper bound
- every element in D is the least upper bound of a directed set of finite elements (algebraic domain)

If f continuous on D Scott domain, then:

$$\mu x.f(x) = \sup\{f^n(\perp) | n \in \mathbb{N}\}$$

Interpretation

Assume a set F of function symbols, an interpretation is a Scott domain D where all function symbols in F are interpreted as continuous functions $f_i \colon D^n \longrightarrow D$ (n the arity of f).

The notion of interpretation is expressed by a second order formula.

Scott induction

Scott induction:

IF $f : D \longrightarrow D$ continuous , D Scott domain **and** R "well-behaved" property. **THEN:**

$$\left[\ \forall x\ \left(R(x) \Rightarrow R(f(x))\right)\ \right] \Longrightarrow R(\mu x.f(x))$$

Example

Let $X = f(X)$ and $\mu X.f(X) = \sup\{f^n(\bot)\} = f^\infty$

Let $Y = g(Y)$ and $\mu Y.g(Y) = \sup\{g^n(\bot)\} = g^\infty$

hypotheses: $f(\bot) \leq g(\bot)$ and $f(g(X)) \leq g(f(X))$

Then: $\left[f(X) \leq g(X) \Rightarrow f(g(X)) \leq g(f(X) \leq g(g(X))\right]$

$$\Longrightarrow f(g^\infty) \leq g(g^\infty)$$

Scott ... or Park... or...?

Park 1970

Fixpoint Induction and Proofs of program Properties

- meaning of **recursive schemes** as least fixpoints
- proofs of properties of schemes by fixpoint induction
- considers also greatest fixpoints

Nivat 1975

On the Interpretation of Recursive Polyadic Program Schemes

Example: $\left[f(\perp) \leq g(\perp) \text{ and } f(g(X)) \leq g(f(X))\right]$
$\Rightarrow \forall n\ f^n(\perp) \leq g^n(\perp) \qquad \Longrightarrow f^\infty \leq g^\infty$

Semantics in the 70s

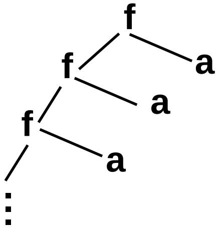
Many researchers worked on the algebraic, logic and automata theory approach to semantics:
Nivat, Park, Scott, Strachey, Luckham, Paterson, Plaisted, Milner, Ashcroft, Hennessy, Winskell, Milne, Morris, Strong, de Bakker, de Roever, Courcelle, Berry, Kotts, Downey, Engelfriet, Indermark, Damm, Fehr, Arnold, Dauchet, Guessarian, Gallier, Manna, Chandra, Vuillemin, Cadiou, Raoult, Burstall, Darlington, Andreka, Nemeti, Tiuryn, ... and even Knuth.

Initial algebra (free algebra, Herbrand model)

$$
\begin{tikzcd}
\textbf{PROGRAM} \arrow[r, "\textit{abstract}"] & \textbf{PROGRAM SCHEME} \arrow[d, "\text{solve in free algebra}"] \\
\textbf{function computed by PROGRAM} & \textbf{Infinite tree} \arrow[l, "\textit{morphism}"']
\end{tikzcd}
$$

Iterative program scheme

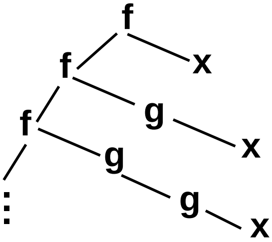
X=f(X,Y) Y=a



Regular (level 0) tree

Recursive program scheme

$$F(x)=f(F(g(x)),x)$$



Algebraic (level 1) tree

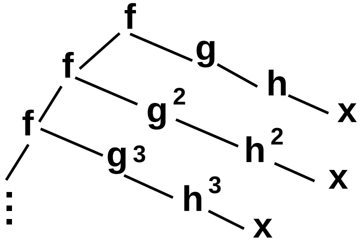
Level 2 program schemes

$$F(x) = \varphi(g, h)(x)$$

$$\varphi(F_1, F_2)(x) = f(\varphi(F_1 \circ g, F_2 \circ h)(x), F_1 \circ F_2(x))$$

Correspond to procedures with procedures as parameters

Level 2 tree of the previous example



Tree of a level 2 program

Semantics

The initial algebra semantics of a deterministic program scheme is an infinite tree which is the least fixed point of the system of equations associated with the program scheme. It can be considered as the value of the program in the "Herbrand" interpretation.

Algebraic trees

An infinite tree T is algebraic iff it is the solution of a deterministic level 1 system of equations of the form: $\varphi_i(x_1, \ldots, x_i) = t_i$ where all function symbols in $F \cup \Phi$ are level 1, *i.e.* of type $D^n \longrightarrow D$, $n \in \mathbb{N}$

Theorem (Courcelle)

T is algebraic iff the set of its branches is a deterministic context-free language.

Program schemes equivalence

Two program schemes are equivalent iff they compute the same function in every interpretation: this implies that they compute the same function in the Herbrand interpretation, *i.e.* that the associated algebraic trees are equal.

Theorem (Sénizergues)

The equivalence problem for deterministic recursive program schemes is solvable.

Other more interesting equivalences (unfortunately more complex).

Higher order schemes

- Fixed point semantics can be adapted by using more complex signatures: each level n tree is the image by a canonical morphism of a regular level (n+1) tree
- additional problem:
  - non-deterministic programs are studied
  - we must distinguish “call by name” **OI** and “call by value” **IO**.

Engelfriet-Schmidt, Damm, ...

Classes of interpretations

Equivalence: P, P' program schemes , T, T' the associated trees,

$$P \approx P' \iff T = T'$$

Too restrictive: relax the demand that P and P' should compute the same function for every interpretation.

$\mathcal{C}$ **a class of interpretations:** $P \approx_{\mathcal{C}} P'$ iff P and P' compute the same function for every interpretation in $\mathcal{C}$.

$\approx_{\mathcal{C}}$ equivalence modulo $\mathcal{C}$

- Captures more interesting equivalences.
- Even harder to study.

wishes

- $\approx_{\mathcal{C}}$ is decidable
- $\approx_{\mathcal{C}}$ is always provable by induction: *i.e.* $\mathcal{C}$ is algebraic
- $\mathcal{C}$ is first-order definable

Algebraic classes

To prove $T \approx_{\mathcal{C}} T'$ it suffices to prove $T \leq_{\mathcal{C}} T'$ and $T' \leq_{\mathcal{C}} T$

Definition: $T \leq_{\mathcal{C}} T'$ iff for every interpretation I in $\mathcal{C}$, $T_I \leq T'_I$ (T_I function computed by T in I).

Intuition: $\mathcal{C}$ is algebraic iff $\leq_{\mathcal{C}}$ is always provable by induction. Formally, t finite tree:

$$t \leq_{\mathcal{C}} \sup_{n \in \mathbb{N}} t'_n \quad \text{iff} \quad \exists n \; t \leq_{\mathcal{C}} t'_n$$

Some results

- $\mathcal{D}$ (the class of discrete interpretations) is algebraic and $\leq_{\mathcal{D}}$ is decidable
- any first-order definable class is algebraic.
- relational classes are algebraic, *i.e.* $R \subset FTrees \times FTrees$ then $\mathcal{C}_R = \{I \mid \forall\ t, t' \in R \quad t_I \leq_I t'_I\}$ is algebraic.

THANKS FOR YOUR ATTENTION