A TAXONOMY OF OBSERVABLES FOR POSITIVE LOGIC PROGRAMS

GIORGIO LEVI

Dipartimento di Informatica, Università di Pisa

LEVI@DI.UNIPI.IT

http://www.di.unipi.it/di/groups/epl/

joint work with

MARCO COMINI

MARIA CATARA MEO
A SEMANTIC FRAMEWORK BASED ON ABSTRACT INTERPRETATION

- Comini & Neo, Compositionality properties of SCD-calculi, TCS 1998
Goals

• a semantic framework for definite logic programs to reason about properties of \textit{SLD-derivations} and their abstractions (observables)
  
  – relation between operational semantics and denotational semantics
  
  – existence of a (goal-independent) denotation
  
  – properties of the denotation, such as precision, correctness, minimality and compositionality

• a taxonomy of observables
  
  – classes are characterized by sets of axioms
  
  – for all the observables in a class we guarantee the validity of some general theorems
  
  – reconstruction of several "precise"
    

and “approximated” semantics (data-flow analysis)
Abstraction is handled by abstract interpretation

- the kernel (collecting) semantics
  - collects, for each goal, all the SLD-derivations
  - is specified in two different styles
    * operational, transition system, top-down
    * denotational, bottom-up
    * the transition system and the denotational semantics are given in terms of four semantic operators, which are directly related to the syntactic structure of the language

- observables are Galois insertions

- abstract interpretation theory to study the relation between observables and to (automatically) derive the abstract transition system and the abstract denotational semantics

- each class in the taxonomy is characterized in terms of axioms relating the (concrete) semantic operators and the Galois insertion
Concrete and abstract behaviors: precision and approximation

- the concrete behaviors
  - $\mathcal{B}[G \text{ in } P]$ is the set of all the derivations for the goal $G$ in $P$
  - $\Omega[G \text{ in } P]$ is the corresponding denotational definition
  - $\mathcal{B}[G \text{ in } P] = \Omega[G \text{ in } P]$

- the observable is denoted by the abstraction function $\alpha$

- the abstract behaviors
  - $\mathcal{B}_\alpha[G \text{ in } P]$ and $\Omega_\alpha[G \text{ in } P]$

- an abstract behavior is precise if
  - for all $G$ and $P$, $\alpha(\mathcal{B}[G \text{ in } P]) = \mathcal{B}_\alpha[G \text{ in } P]$

- an abstract behavior is a (correct) approximation if
  - for all $G$ and $P$, $\alpha(\mathcal{B}[G \text{ in } P])$ is more precise than $\mathcal{B}_\alpha[G \text{ in } P]$

"...abstract computations..."
Abstract (goal-independent) denotations and their properties

- **bottom-up denotation**
  - the abstract denotational semantics of the set of clauses
    \[ \mathcal{F}_\alpha[P] = \operatorname{lfp} \mathcal{P}_\alpha[P] = \mathcal{P}_\alpha[P] \uparrow \omega \]

- **top-down denotation**
  - the observables for most general atomic goals
    \[ \mathcal{O}_\alpha[P] = \sum \{ \mathcal{B}_\alpha[p(x) \text{ in } P] \}_{p(x) \in \text{Goals}} \]

- **correctness** of a denotation
  - if \( \mathcal{O}_\alpha[P_1] = \mathcal{O}_\alpha[P_2] \), then, for all \( G \),
    \( \alpha(\mathcal{B}[G \text{ in } P_1]) = \alpha(\mathcal{B}[G \text{ in } P_2]) \)
  - if \( P_1 \) and \( P_2 \) have the same abstract denotation, then they
    cannot be distinguished by looking at the abstractions of their behaviors

- **minimality** (full abstraction) of a denotation
  - if, for all \( G \), \( \alpha(\mathcal{B}[G \text{ in } P_1]) = \alpha(\mathcal{B}[G \text{ in } P_2]) \), then
    \( \mathcal{O}_\alpha[P_1] = \mathcal{O}_\alpha[P_2] \)

- the observable \( \alpha \) is **condensing** if the abstract behavior (for all
  the goals) can be derived from the goal-independent abstract denotation

- a denotation is **AND-compositional** if the semantics of a conjunctive
  goal can be derived from the semantics of its conjuncts

- a denotation is **OR-compositional** if the semantics of a union
  of programs can be derived from the semantics of the programs
Use of the semantic framework

- to reconstruct an existing semantics or to define a new semantics

1. formalize the property you want to model as a Galois insertion \( \langle \alpha, \gamma \rangle \) between SLD-derivations and the property domain

2. verify some algebraic axioms relating \( \langle \alpha, \gamma \rangle \) and the basic semantic operators on SLD-derivations, to assign the observable to the right class

3. depending on the class, you get automatically the new denotational semantics, transition system, top-down and bottom-up denotations, together with several theorems (equivalence, compositionality w.r.t. the various syntactic operators, correctness and minimality of the denotations)

- used for semantics-based program analysis (abstract interpretation, abstract diagnosis, etc.)
Plan of the Talk

• the collecting semantics ($SLD$-derivations)
  
  – transition system, denotational semantics, semantic properties

• observables as Galois insertions

• a taxonomy of (condensing) observables
  
  – perfect observables
    * precise and equivalent abstract transition system and abstract denotational semantics
    * correct, minimal, AND-compositional and OR-compositional top-down and bottom-up denotations

  – denotational observables
    * precise abstract denotational semantics
    * correct, minimal and AND-compositional bottom-up denotation

  – semi-perfect observables
    * (correctly) approximated and equivalent abstract transition system and abstract denotational semantics
    * AND-compositional and OR-compositional top-down and bottom-up denotations

  – semi-denotational observables
    * the most precise (correctly) approximated abstract semantics is the denotational one
    * AND-compositional bottom-up denotation
The denotational collecting semantics

- the semantic domain (a complete lattice)
  - equivalence classes (variance) of pairs composed of goals and SLD-trees represented as sets of derivations (leftmost selection rule)
  - a preorder $\preceq$ on derivations (prefix)

- the denotational semantics (main definitions)
  \[ \mathcal{O}(G \text{ in } P) = \mathcal{G}(G)_{fp} \mathcal{P}(P) \]
  \[ \mathcal{G}[A, G]_I = A[A]_I \times \mathcal{G}[G]_I \]
  \[ \mathcal{A}[A]_I = A \cdot I \]
  \[ \mathcal{P}([c] \cup P)_I = \mathcal{C}[c]_I + \mathcal{P}[P]_I \]
  \[ \mathcal{C}[p(t) :- B]_I = \text{tree}(p(t) :- B) \Join \mathcal{G}[B]_I \]

- the basic semantic operators
  1. $A \cdot D$ is the instantiation of $D$ with $A$
  2. $D_1 \times D_2$ is the product of $D_1$ and $D_2$ (semantic version of the syntactic conjunction)
  3. $D_1 \Join D_2$ is the replacement of $D_2$ in $D_1$ (semantic version of the syntactic implication)
  4. $\sum \{ D_i \}_{i \in I}$ is the sum of a set of elements $\{ D_i \}_{i \in I}$ (semantic version of the syntactic disjunction)

- the usual denotational definitions (and $T_P$ operators) are much more abstract
  - define computed answers (ground instances of computed answers) rather than SLD-trees
The operational collecting semantics

- a transition system $\mathcal{T} = (\mathcal{D}, \xrightarrow{P})$ defined using the same semantic operators used in the denotational definition

- initial states of $\mathcal{T}$: all the collections of $SLD$-derivations of length zero

- final states of $\mathcal{T}$: all the collections of all $SLD$-refutations and finite failures

- $D \xrightarrow{P} D \Join \sum\{(A \cdot \text{tree}(P)) \times \text{Id}\}_{A \in \text{Atoms}}$

- the behavior of $P$: all the $SLD$-derivations of a query $G$ in $P$
  
  $- \mathcal{B}[G \text{ in } P] = \sum\{ D \mid \langle G, \{ G \} \rangle \xrightarrow{P}^* D \}$
  
  $- \xrightarrow{P}^*$ is the reflexive and transitive closure of $\xrightarrow{P}$

- $\mathcal{B}[G \text{ in } P]$ and $\mathcal{Q}[G \text{ in } P]$ are equivalent

- the usual operational semantics are more abstract
  
  $- \text{states are frontiers of the } SLD\text{-tree rather than sets of } SLD\text{-derivations}$
The goal-independent denotation

- the top-down denotation
  - collecting only the behaviors for all most general atomic goals (behaviors of the procedures with no constraints on the inputs)
  
  \[ \mathcal{O}[P] = \sum \{ \mathcal{B}[p(x) \text{ in } P] / \equiv \}_{p(x) \in \text{Goals}} \]

- the bottom-up denotation
  - the semantics of the program as a set of definite clauses (procedure declarations)
  - \( \mathcal{P}[P] \) is the "bottom-up" immediate consequences operator in the case of derivations
  
  \[ \mathcal{F}[P] = \text{lfp} \mathcal{P}[P] = \mathcal{P}[P] \uparrow \omega \]

- \( \mathcal{F}[P] \) and \( \mathcal{O}[P] \) are equivalent
  - \text{SLD}-derivations are condensing
    * the (goal-independent) denotation is meaningful
  - the denotations are
    * correct
    * minimal
    * AND-compositional
    * OR-compositional
Observables

- observable
  - a property which can be extracted from SLD-derivations together with an ordering relation (approximation)
  - formalized according to abstract interpretation theory
    * the concrete domain $(\mathcal{D}, \sqsubseteq)$ (a complete lattice)
    * the abstract domain $(\mathcal{D}, \leq)$ (a complete lattice)
    * $(\alpha, \gamma) : (\mathcal{D}, \sqsubseteq) \Rightarrow (\mathcal{D}, \leq)$ is a Galois insertion
      1. $\alpha$ and $\gamma$ are monotonic
      2. $\forall x \in \mathcal{D}, x \sqsubseteq (\gamma \circ \alpha)(x)$
      3. $\forall y \in \mathcal{D}, (\alpha \circ \gamma)(y) = y$

- from the concrete semantics to the abstract semantics
  - concrete semantics: the least fixpoint of a semantic function $F : \mathcal{D} \rightarrow \mathcal{D}$
  - $f : \mathcal{D}^n \rightarrow \mathcal{D}$ a "primitive" semantic operator
  - $\tilde{f}$ its abstract version
    * $\tilde{f}$ is (locally) correct w.r.t. $f$ if
      $\forall x_1, \ldots, x_n \in \mathcal{D}, f(x_1, \ldots, x_n) \sqsubseteq \gamma(f(\alpha(x_1), \ldots, \alpha(x_n)))$
  - an abstract semantic function $\tilde{F} : \mathcal{D} \rightarrow \mathcal{D}$ is correct if
    $\forall x \in \mathcal{D}, F(x) \leq \gamma(\tilde{F}(\alpha(x)))$
  - local correctness of all the primitive operators implies the global correctness
  - if we replace the concrete operators by locally correct abstract versions, we obtain a correct abstract semantics
Towards a systematic construction of the optimal abstract semantics

• optimality and precision
  
  – for each operator $f$, there exists an optimal (most precise) locally correct abstract operator $\tilde{f}$ defined as
  $$\tilde{f}(y_1, \ldots, y_n) = \alpha(f(\gamma(y_1), \ldots, \gamma(y_n)))$$
  – the composition of optimal operators is not necessarily optimal
  
  – $\tilde{f}$ is precise if $\forall x_1, \ldots, x_n \in \mathbb{D}$,
    $$\alpha(f(x_1, \ldots, x_n)) = \tilde{f}(\alpha(x_1), \ldots, \alpha(x_n))$$
    * the optimal abstract operator $\tilde{f}$ is precise if
      $$\alpha(f(x_1, \ldots, x_n)) = \alpha(f((\gamma \circ \alpha)(x_1), \ldots, (\gamma \circ \alpha)(x_n)))$$
    * the precision of the optimal abstract operators can be formulated in terms of properties of $\alpha$, $\gamma$ and the corresponding concrete operators

• our approach
  
  – take the optimal abstract versions of the concrete operators
  – check under which conditions (on the observable) the resulting abstract semantics is optimal
Perfect observables

- the abstract denotational and operational semantics are equivalent and precise

- the axioms

  1. $\alpha(A \cdot D) = \alpha(A \cdot (\gamma \circ \alpha)D)$
  2. $\alpha(D_1 \times D_2) = \alpha((\gamma \circ \alpha)D_1 \times (\gamma \circ \alpha)D_2)$
  3. $\alpha(D_1 \Join D_2) = \alpha((\gamma \circ \alpha)D_1 \Join (\gamma \circ \alpha)D_2)$

  - for any Galois insertion
    $\alpha(\sum\{ D_i \}_{i \in I}) = \alpha(\sum\{ (\gamma \circ \alpha)D_i \}_{i \in I})$

- the properties

  - $\mathcal{B}_\alpha[G \text{ in } P] = \mathcal{Q}_\alpha[G \text{ in } P] = \alpha(\mathcal{B}[G \text{ in } P])$
  - $\mathcal{O}_\alpha[P] = \mathcal{F}_\alpha[P] = \alpha(\mathcal{O}[P])$

  - perfect observables are condensing
  - the denotation $\mathcal{O}_\alpha[P] = \mathcal{F}_\alpha[P]$ is correct, minimal, AND-compositional and OR-compositional

- examples of perfect observables

  - computed resultants
  - proof trees (Heyting semantics)

- computed answers and frontiers are not perfect
From the observable to the abstract semantics

- the optimal abstract operators
  \[ \sum \{ S_i \}_{i \in I} = \alpha (\sum \{ \gamma(S_i) \}_{i \in I}) \]
  \[ A \times S = \alpha(A \cdot \gamma(S)) \]
  \[ S_1 \times S_2 = \alpha(\gamma(S_1) \times \gamma(S_2)) \]
  \[ S_1 \otimes S_2 = \alpha(\gamma(S_1) \otimes \gamma(S_2)) \]

- abstract denotational semantics
  \[ O_\alpha[G \text{ in } P] = \mathcal{G}_\alpha[G]_{\text{lfp } P_\alpha[P]} \]
  \[ \mathcal{G}_\alpha[A, G]_S = \mathcal{A}_\alpha[A]_S \times \mathcal{G}_\alpha[G]_S \]
  \[ \mathcal{A}_\alpha[A]_S = A \times S \]
  \[ P_\alpha[\{c\} \cup P]_S = \mathcal{C}_\alpha[c]_S + P_\alpha[P]_S \]
  \[ \mathcal{C}_\alpha[p(t) : - B]_S = \alpha(\text{tree}(p(t) : - B)) \otimes \mathcal{G}_\alpha[B]_S \]

- abstract operational semantics
  \[ S \xrightarrow{P} \sum \{ (A \times \alpha(\text{tree}(P))) \times \alpha(\text{Id}) \}_{A \in \text{Atoms}} \]

- behavior and abstract denotations
  \[ B_\alpha[G \text{ in } P] = \sum \{ S \mid \alpha(\langle G, \{ G \} \rangle) \xrightarrow{P} \} \]
  \[ O_\alpha[P] = \sum \{ B_\alpha[p(x) \text{ in } P] \}_{p(x) \in \text{Goals}} \]
  \[ F_\alpha[P] = \text{lfp } P_\alpha[P] = P_\alpha[P] \uparrow \omega \]
Denotational observables

• in several interesting observables $\otimes$ is not precise
  
  − we can obtain a more precise semantics by choosing the optimal abstractions of higher level concrete operators
  
  − in the denotational semantics $\otimes$ is only used inside the semantic function $\mathcal{C}$

  − take the optimal abstraction $\hat{\mathcal{C}}$

• relax the third axiom (a non-precise $\otimes$)

• the new axioms

  1. $\alpha(A \cdot D) = \alpha(A \cdot (\gamma \circ \alpha)D)$
  
  2. $\alpha(D_1 \times D_2) = \alpha((\gamma \circ \alpha)D_1 \times (\gamma \circ \alpha)D_2)$

  3. $\alpha(D_1 \otimes D_2) = \alpha(D_1 \otimes (\gamma \circ \alpha)D_2)$

• if we replace $\mathcal{C}_\alpha$ by the optimal abstraction
  
  $\hat{\mathcal{C}}[c] = \alpha \circ \mathcal{C}[c] \circ \gamma$, we obtain a precise denotational semantics

• the properties

  − $\mathcal{Q}_\alpha[G \text{ in } P] = \alpha(\mathcal{B}[G \text{ in } P])$

  − $\mathcal{F}_\alpha[P] = \alpha(\mathcal{O}[P])$

  − the denotation $\mathcal{F}_\alpha[P]$ is correct, minimal and AND-compositional

• examples of denotational observables

  − ground instances of computed answers (least Herbrand model), instances of computed answers (c-semantics), computed answers (s-semantics), partial answers, call patterns
THE OPERATIONAL SEMANTICS
OF DENOTATIONAL OBSERVABLES

- The intuition system is not precise
  - $Q_{ad}[\text{true}] = \alpha (B[I\text{true}]) \leq B_{a}[I\text{true}]
  - $T_{a}[\text{false}] = \alpha (O[I\text{false}]) \leq O_{a}[I\text{false}]

- We cannot compute answers by abstracting at each transition step

  - We need to compute with a more concrete observable (e.g. values) and abstract to
    computed answers in the realm of the computation
Introducing abstract computations with approximation

- observables used in (static) program analysis lead to a loss of precision to obtain finitely computable semantics
- the abstract semantics is required to be a correct approximation of the concrete one, yet it is not precise
  - as a consequence, we have to give up correctness and minimality of the denotation
- semi-perfect observables
  - the properties
    * $\alpha(\mathcal{B}[G \text{ in } P]) \leq \mathcal{B}_a[G \text{ in } P] = \mathcal{O}_a[G \text{ in } P]
    * $\alpha(\mathcal{O}[P]) \leq \mathcal{O}_a[P] = \mathcal{F}_a[P]
    * semi-perfect observables are condensing
    * the denotation $\mathcal{O}_a[P] = \mathcal{F}_a[P]$ is AND-compositional and OR-compositional
  - examples: SLD-derivations and computed resultants, with concrete substitutions abstracted to elements of POS or to types
- semi-denotational observables
  - the properties
    * $\alpha(\mathcal{B}[G \text{ in } P]) \leq \mathcal{O}_a[G \text{ in } P] \leq \mathcal{B}_a[G \text{ in } P]
    * $\alpha(\mathcal{O}[P]) \leq \mathcal{F}_a[P] \leq \mathcal{O}_a[P]
    * the denotation $\mathcal{F}_a[P]$ is AND-compositional
  - examples: call patterns and computed answers, with concrete substitutions abstracted to elements of POS or to types

17
Open problems

- the axioms allow us to handle separately precision and the various compositionality properties
  - more classes of observables, with weaker properties
    * for example, non-condensing
- the lattice of observables and the sublattices of perfect, denotational, ... observables
  - how to combine observables (glb and lub on specific classes should have stronger properties)
  - how to choose the most abstract among the observables more concrete than \( \alpha \) belonging to a suitable class