Interpretare abstractă

11 mai 2004

Marius Minea

Interpretare abstractă: Introducere

O metodă pentru definirea unei *semantici abstracte* a unui program, care poate fi utilizată pentru a analiza programul și a produce informatii despre comportamentul său în execuție. [Cousot & Cousot '77]

Inspirată din:
– analiza fluxului de date
sistematizează noțiunile de proprietăți analizate și caracteristicile fundamentale ale metodelor de analiză
– semantică denotațională
formalizează noțiunea de *aproximare* a semanticii unui program (corectitudinea e nedecidabilă ⇒ avem nevoie de aproximare)

Exemplu simplu: abstracția semn

Considerăm un limbaj care conține doar întregi și înmulțire: $e ::= i \mid e * e$
Definim semantica printr-o funcție care dă valoarea unei expresii:
$\mu : Exp \to Int$, $\mu(i) = i$, $\mu(e_1 * e_2) = \mu(e_1) * \mu(e_2)$
Definim o funcție de abstracție $\alpha : Exp \to \{-, 0, +\}$:

întâi pentru întregi: $\alpha(i) = \begin{cases} - & \text{dacă } i < 0 \\ 0 & \text{dacă } i = 0 \\ + & \text{dacă } i > 0 \end{cases}$

și pentru expresii prin tabelul:

$\bar{\cdot}$	$-$	0	$+$
$-$	$+$	0	$-$
0	0	0	0
$+$	$-$	0	$+$

La fel se poate introduce operatorul minus unar:

	$-$	0	$+$
$\bar{-}$	$+$	0	$-$

Abstracția semn: continuare

Introducem operația de adunare $\Rightarrow$ abstracția nu mai e precisă

$\bar{+}$	$-$	0	$+$
$-$	$-$	$-$	$\top$
0	$-$	0	$+$
$+$	$\top$	$+$	$+$

Apar valori care nu pot fi determinate precis (se pierde precizia)

$\Rightarrow$ e necesară introducerea lui $\top = \{-, 0, +\}$

$\Rightarrow$ apare din nou noțiunea de *latice*

Abstracţia semn: continuare

Similar, la introducerea operatorului de împărţire $/$:
nu putem împărţi la zero.

$\bar{/}$	$-$	0	$+$	$\top$
$-$	$+$	$\perp$	$-$	$\top$
0	0	$\perp$	0	0
$+$	$-$	$\perp$	$+$	$\top$
$\top$	$\top$	$\perp$	$\top$	$\top$

şi extindem toate operaţiile şi la $\perp$: $\perp \ op \ x = x \ op \ \perp = \perp$

Interpretarea abstractă: considerații fundamentale

Semantica unui program: un model matematic (formal) al tuturor comportamentelor posibile ale unui sistem de calcul care execută acel program, în interacțiune cu orice mediu posibil [Cousot]

Semantica unui limbaj de programare: definește semantica oricărui program

Abordare generală: semantica unui program poate fi definită ca soluție a unei *ecuații de punct fix* (corespunzătoare iterației).
$\Rightarrow$ toate semanticile unui program pot fi organizate într-o ierarhie, după nivelul lor de *abstracție*.

Exemple de abstracţii

– secvenţe de execuţie (finite sau infinite): descriu în fiecare punct valoarea variabilelor din program
– semantică operaţională: descrie comportamentul la nivel de stări şi tranziţii (ca şi automat)
– semantică denotaţională: descrie rezultatul execuţiei (incl. neterminare)
– semantică naturală: ca mai sus, ignoră aspectul neterminării

Exemple efective (numerice) de abstracţii

– abstracţia semn
– abstracţia pe intervale
– abstracţia poliedrală (înfăşurătoarea convexă a valorilor)
– abstracţia octogonală (ecuaţii de forma $\pm x \leq c$, $\pm y \leq c$, $\pm x \pm y \leq c$)
– abstracţia modulo un număr, punctual sau pe intervale

– abstracţii nerelaţionale (ex. abstracţia carteziană): calculează abstracţia individual pentru fiecare variabilă, neţinând cont de corelare
– abstracţii relaţionale: păstrează corelarea între variabile

Abstractizare și concretizare

Constă în:

– un domeniu concret D și un domeniu abstract A, legate printr-o *conexiune Galois*:

– o funcție de abstracție $\alpha : D \rightarrow A$

– o funcție de concretizare $\gamma : A \rightarrow \mathcal{P}(D)$

(asociază fiecărei valori abstracte o mulțime de valori concrete)

– a.î. $\forall x \in \mathcal{P}(D) \,.\, x \subseteq \gamma(\alpha(x))$ și $\forall a \in A \,.\, a = \alpha(\gamma(a))$

(abstractizarea urmată de concretizare introduce aproximare; concretizarea urmată de abstractizare e exactă)

Aceasta e proprietatea fundamentală care exprimă corectitudinea abstracției (*soundness*): ea poate genera valori suplimentare, dar niciodată nu va omite valori $\Rightarrow$ e o abstracție conservatoare.

Abstracții ale funcțiilor

Fiind dată o funcție (transformare) in programul $f : D \to D$ în programul concret, ce corespondent are aceasta în programul abstract ?

Răspuns: $f^\sharp = \alpha \circ f \circ \gamma$

Pentru un element abstract $x \in A$:
– producem întâi mulțimea de valori concrete $\gamma(x)$
– aplicăm funcția concretă f fiecărei valori, obșinând o mulțime de valori (concrete)
– abstractizăm mulțimea de valori concrete într-o valoare abstractă (cu obișnuita posibilă pierdere de precizie)

Abstracții de punct fix

Semantica unui program e dată în general de o ecuație de punct fix (datorată ciclurilor din program)
– am văzut deja ecuații de punct fix pentru analiza de flux de date

Avem:
– punctul fix al funcției f în domeniul concret
– punctul fix al funcției $f^\sharp$ în domeniul abstract
– concretizarea punctului fix al lui $f^\sharp$
Atunci, $\mathsf{lfp} f \sqsubseteq \gamma(\mathsf{lfp} f^\sharp)$

Probleme de punct fix și terminare

Problemă: dacă lanțul ascendent în calculul de punct fix are lungime infinită
(laticea e de înălțime infinită în raport cu funcția/transformarea dată), nu se poate determina un punct fix într-un număr finit de pași

```
while (x >= 0)
  x = x + 1;
```

Ex. analiza codului de mai sus chiar cu intervale.
Pe de altă parte, “ghicind” intervalul $[0, \infty)$ e clar că acesta e punct fix (și punctul fix minimal).

Lărgire și restrângere (widening și narrowing)

Permite eliminarea lanțurilor de lungime infinită la calculul de punct fix, în doi pași:
– la început, o aproximare mai grosieră a lanțului ascendent, care conduce la convergenă mai rapidă (în număr finit de pași)
– apoi, o revenire la precizie mai bună, printr-un lanț de aproximări descendente, spre punctul fix propriu-zis