Flat GHC プログラムの抽象解釈

堀内 謙二

(財) 新世代コンピュータ技術開発機構

1. はじめに

抽象解釈 (abstract interpretation) はあるプログラムの様々な特性を解析するための統一的な枠組として適している．抽象解釈とは，プログラムを近似的に実行することにより，実際の実行を効率的にシミュレートするものである．近似の度合いに応じて解析される内容は変化するが，その基本的な抽象解釈の枠組は同じものが使える．この統一的な性質が抽象解釈の大きな武器である．

ここでは，FGHC プログラムのプログラム解析のための統一的な枠組として不動点意味論に基づく抽象解釈の検討を行い，モード付きタイプ推定の例とその応用について考察した．

2. 抽象解釈の枠組

抽象解釈の最大の目的は有限回の近似的な実行で任意のプログラムから何か有益な情報を得ることである．その情報が有益であるために必要な抽象解釈の正当性の条件について簡単に解説する．

プログラム P の意味が単調で連続な (具体) 意味関数 $T_P : D \to D$ の最小不動点 ($lfp(T_P)$) で与えられるとする．D(具体領域と呼ぶ) の各要素はプログラムの計算を表現している．次に，D を抽象化し抽象領域 $\underline{D}$ を定義する．$\underline{D}$ は有限集合でその各要素が D の部分集合を表現している．また，D と $\underline{D}$ 間には次の条件を満足する関数 $\alpha : D \to \underline{D}$ と $\gamma : \underline{D} \to D$ が定義さなければならない．

$$(\forall \underline{d} \in \underline{D}(\underline{d} = \alpha(\gamma(\underline{d})))) \wedge (\forall d \in D(d \subseteq \gamma(\alpha(d))))$$

抽象解釈は，$\underline{D}$ の設計に加えて，T_P に対応する抽象意味関数 $\underline{T}_P : \underline{D} \to \underline{D}$ を定義することによって実現される．$\underline{D}$ は有限集合なので，有限回の変換で $\underline{T}_P$ の最小不動点 $lfp(\underline{T}_P)$ が求まる．この時のプログラム P の $\underline{T}_P$ による最小不動点 ($lfp(\underline{T}_P)$) を抽象意味論と呼ぶ．

抽象意味論の具体意味論に関する正当性は

$$lfp(T_P) \subseteq \gamma(lfp(\underline{T}_P))$$

で与えられ，この式が成り立つためには $D, T_P, \underline{D}, \underline{T}_P$ の間に次の条件が成り立っていれば十分であることが分かっている [1]．

$$\forall \underline{d} \in \underline{D}(\{T_P(d) | d \in \gamma(\underline{d})\} \subseteq \gamma(\underline{T}_P(\underline{d})))$$

解析したい特性毎にこれらの条件を満足する $\underline{D}, \underline{T}_P$ を定義 (設計) すれば良い．

3. Flat GHC の意味論

GHC や FCP などの commited choice 型の言語の意味論は，これまで色々と提案されている．ここで提案する意味論は，あるゴールの動作を入出力を区別するための annotation を付加した代入の列で表現している．

まず，基本的な記法について述べる．項，アトム，代入及びそれらへの各種操作などの定義や記法は通常の定義を用いる．詳しくは [3],[2] などを参照されたい．

代入 θ に $+$ または $-$ を付加したものを反応 (reaction) と呼び，θ^+ または θ^- で表す．反応の列を反応列と呼び，$\delta_1\delta_2\ldots\delta_n$ または Δ などで表す．ここで，各 δ_i は反応である．また，空の反応列を $\Box$ で表わす．アトム A と反応列 Δ の対 (A, Δ) をアトム反応と呼ぶ．反応列が空のアトム反応 $(A, \Box)$ を特に (アトム A の) 初期アトム反応と呼ぶ．アトム反応 (A, Δ) はゴール A の動作のある状態を表現している．

FGHC プログラム P が与えられた時，P に現れる全ての述語記号，関数記号を用いて得られるすべてのアトム反応の集合を R_P で表す．また，あるアトム G が与えられた時，各要素がアトム G の初期アトム反応を含むような R_P のベキ集合の部分集合をプログラム P のゴール G に対する具体領域とし，$D_{P,G}$ で表す．

$$D_{P,G} = \{I \mid I \in 2^{R_P} \wedge (G, \Box) \in I\}$$

FGHC プログラム P のゴール G に対する具体意味関数 $T_{P,G} : D_{P,G} \to D_{P,G}$ を次で与える．

$$\begin{aligned}
&T_{P,G}(I) = \\
&\quad \{(s = t, \theta^-) \mid (s = t, \Box) \in I \wedge \theta = mgu(s, t)\} \cup \\
&\quad \{(B_i\sigma\theta_g, \Box) \mid \exists (A, \Box) \in I \wedge \exists \sigma \exists (H :\!- G \mid B) \in P \\
&\qquad\qquad \text{such that } \sigma = mgu(A, H)_{|var(H)} \wedge \\
&\qquad\qquad \exists \theta_g \in mgu(G\sigma) \wedge \exists B_i \in B\} \cup \\
&\quad \{(A, \theta_g^+\Delta) \mid \exists (A, \Box) \in I \wedge \exists \sigma \exists (H :\!- G \mid B) \in P \\
&\qquad\qquad \text{such that } \sigma = mgu(A, H)_{|var(H)} \wedge \\
&\qquad\qquad \exists \theta_g \in mgu(G\sigma) \wedge \\
&\qquad\qquad \forall B_i \in B \exists (B_i\sigma\theta_g, \Delta_i) \in I \\
&\qquad\qquad (\Delta = \Delta_1 \| \Delta_2 \| \ldots \| \Delta_{n|var(A)})\}
\end{aligned}$$

ここで，$mgu(U)$ は unification goal の対または集合 U の most general unifier を表し，$\|$ は並行合成を表し，$\Delta_1 \| \Delta_2 =$

$$\{\delta\Delta \mid \Delta_i = \delta\Delta_i' \wedge \Delta = \Delta_i' \| (\Delta_j)\delta \wedge 1 \le i \ne j \le 2\}$$

で定義される．$(\Delta)\delta$ は Δ への δ の適用である．

この具体意味関数 $T_{P,G}$ の最小不動点 $lfp(T_{P,G})$ によって，プログラム P のゴール G に対する意味を特徴付ける．

4. モード付きタイプ推定

モード付きタイプ推定を例にとり，前節の具体領域 $D_{P,G}$ や意味関数 $T_{P,G}$ を抽象化する．$func(P)$ はプログラム P に現れるすべての関数記号または定数記号の集合であるとする．ここで用いられるモード付きタイプ推定のため

Abstract Interpretation for Flat GHC Programs

Kenji HORIUCHI

Institute for New Generation Computer Technology

の抽象領域を構成する抽象項は，以下で定義される根ノード付き有向グラフである．

(1) ノードは $f/n^+, f/n^-$ または _ であり，$f/n^+, f/n^-$ は n 個の子ノードを持ち，_ は子ノードを持たない．ここで，$f/n \in func(P)$.

(2) あるノードの子ノードがそのノードの 祖先ノードかも知れない (循環パスが存在するかも知れない).

(3) 自分以外の全てのノードへのパスを持つノードが少なくとも 1 つあり，そのノードが根ノードである．

一般にはこの定義による抽象項の種類は無限に存在し得るが，ある種の正規化手続きで有限に抑えることができる．これは，例えば，グラフ内の循環パス上に現れる同一のノードを有限個に限定するなどである．

抽象項は項の集合を表現している．例えば，抽象項 $f^+(\cdot)$ の表現する項の集合を $\gamma(f^+(\cdot))$ と表すと $\gamma(f^+(\cdot))$ は以下のような項の集合である．

$$\gamma(f^+(\cdot)) = \{f(\_), f(f(\_)), f(f(f(\_))), \ldots\}.$$

以下では，反応，反応列の抽象項を用いた抽象化，抽象化された反応列の合成 $\|$ や適用を例を用いて簡単に述べる．ここで反応 δ や反応列 Δ の抽象化をそれぞれ $\alpha(\delta)$, $\alpha(\Delta)$ で表す．

(1) 反応 δ の抽象化 ($\underline{\delta}$)

反応 $\{\ldots, X \leftarrow t, \ldots\}^+$ は $\{\ldots, X \leftarrow t^+, \ldots\}$ と抽象化される．ここで，t^+ は抽象項である．

例

$$\alpha(\{X \leftarrow f(Y)\}^+) = \{X \leftarrow f^+(\_)\}$$
$$\alpha(\{X \leftarrow f(f(f(f(Y)))))\}^+) = \{X \leftarrow f^+(\cdot)\}$$

(2) 反応列 Δ の抽象化 ($\underline{\Delta}$)

反応列 Δ の抽象化は，Δ 内の各反応の合成の抽象化の列である．

例

$$\alpha(\ldots\{X \leftarrow f(Y)\}^+\{Y \leftarrow g(Z)\}^- \ldots) =$$
$$\ldots\{X \leftarrow f^+(\_)\}\{X \leftarrow f^+(g^-(\_))\}\ldots$$

(3) 抽象化反応列 $\underline{\Delta}$ への抽象化反応の適用 ($(\underline{\Delta})\underline{\delta}$)

$\underline{\delta}$ が $\{X \leftarrow \underline{t_1}\}$ で，Δ 内に抽象化反応が $\{X \leftarrow \underline{t_2}\}$ あるなら，それを $\{X \leftarrow \underline{t}\}$ とする．ここで $\underline{t}$ は

$\gamma(t) \supseteq \{t \mid t_1 \in \gamma(t_1) \wedge t_2 \in \gamma(t_2) \wedge t = mgci(t_1, t_2)\}$

なる抽象項である．またここで，$mgci(t_1, t_2)$ は項 t_1, t_2 の共通の instance で最も general なもの (most general common instance) である [2]，一般に $\underline{t}$ はノード f/n^- は f/n^+ の instance であると考えて，$\underline{t_1}$ と $\underline{t_2}$ の対応するノードを擦り合わせることによって求まる．

例

$$(\ldots\{X \leftarrow f^+(g^-(\cdot))\}\ldots)\{X \leftarrow f^-(g^+(\_))\}$$

は

$$(\ldots\{X \leftarrow f^-(g^-(f^+(g^-(\cdot)))))\}\ldots)$$

となり，もし，またさらに，

$$\{X \leftarrow f^-(g^+(\cdot))\}$$

を適用したならば，

$$(\ldots\{X \leftarrow f^-(g^-(\cdot))\}\ldots)$$

となる．

(4) 抽象化反応列の並行合成 ($\underline{\Delta_1}\|\underline{\Delta_2}$)

抽象化反応列の並行合成は前節で示した具体領域上の並行合成の定義から自然に定義できる．

5. 多重書き込みゴールの発見

異なるゴールが共有変数の同じ部分にそれぞれが何か項を書き込もうとすることを多重書き込み (multiple write) と呼ぶ．多重書き込みを許さないように制限された Flat GHC を moded FGHC と呼び，効率の良い処理方式が提案されている [4]．また moded FGHC は，(1) ゴールが失敗しない，(2) モード解析が容易であるなどの利点があるので，今後の FGHC のプログラミングスタイルとしては有力である．

一般に多重書き込みが必要なプログラミング例は少ないと思われるが，中には多重書き込みが有用なプログラミング技法もあり，例えば，stop signal がそれである．

ここでは前節のモード付きタイプ推定の応用例として，多重書き込みの可能性を調べ，もしある場合プログラムのどの部分が多重書き込みに加担しているかを発見する方法について述べる．次のプログラムは stop signal を実現している非常に単純な例である．関数記号の添字はその出現を区別するための ID である．

```
g(T,F) :- true | q(T), p(T, F).
p(t1(_,a1,_),F) :- true | F=f1.
p(_,f2) :- true | true.
p(t2(L,b1,R),F) :- true | p(L,F), p(R,F).
q(T) :- true | T=t3(L,N,R), q1(N), q(L), q(R).
q1(N) :- true | N=a2.
q1(N) :- true | N=b2.
```

プログラム q はノードが a，b のいずれかである 2 進木を生成しており，プログラム p は a とラベル付けされているノードを再帰的に探索している．また各 p は第 2 引数を stop signal 用に共有している．上記のプログラムをゴール g(T,F) に関してモード付きタイプで解析を行う．ここで用いられる抽象項は前節のそれの少し変形で，関数記号の出現 ID を付加情報として持つ．すなわち，$\Delta = \ldots\{X \leftarrow a_2^-\}\ldots$ に $\underline{\delta} = \{X \leftarrow a_1^+\}$ を適用した時，$(\underline{\Delta})\underline{\delta} = \{X \leftarrow a_{21}^-\}$ となるような抽象項である．

− 付きの抽象項に − 付きの抽象項を適用することは多重書き込みを意味する．上の例では，

$$(g(T,F), \ldots\{F \leftarrow t_{11}^+\}\ldots)$$

なる抽象化アトム反応がモード付きタイプ推定で得られるのでゴール g(T,F) は多重書き込みを引き起こす可能性が有り，かつそれを引き起こすのは関数記号 t_1 を書き込む unification goal であることが分かる．

参考文献

[1] Cousot, P. et al, "Abstract Interpretation: A Unified Framwork for Static Analysis of Programs by Construction or Approximation of Fixpoints", Proc. of the 4th POPL, 1977

[2] Lassez, J. L., et al, "Unification Revised", Proc. of the W'shop on Logic and Data Bases, 1987.

[3] Lloyd, J.W., "Foundation of Logic Programming", Second, Extended Edition, Springer-Verlag, 1987.

[4] Ueda, K. et al, "A New Implementation Technique for Flat GHC", Proc. of the 7th ICLP, The MIT Press, 1990.