

Logical Basis of Program Synthesis

京大 計算機科学研究所

謝 章文

1. Logical Program Synthesis

Formal specification から correctness および executability の保証された program を構成する過程で、deductive system における derivability や definability に関する機能を利用する program synthesis の研究方法を logical program synthesis という。

ここでは first-order theory に基づく first-order logical program synthesis の基本概念および基本原理を一般的に考察する。

2. Formal Specification and its Denotation

[定義 (Formal Specification)] Formulas の集合 Σ , 存在限量子で束縛された 1 個以上の変数記号を含む formula ψ , Σ および ψ に出現する述語記号のある集合 Δ とする。 $\Sigma \cup \{\psi\}$ が consistent であるとき、 (Σ, ψ, Δ) を formal specification FS という。 $\square$

話を簡単にするため、 Σ および ψ の formulas は closed formulas, Σ の formulas は存在限量子を含まない、 ψ は Σ に出現しない

関数記号を含まないものとする。 φ は prenex normal form とし、その matrix を φ で表わす。

φ の prefix 中 (I) 全称限量子 (II) 存在限量子で束縛される変数記号は (I) 入力変数 (II) 出力変数を表わす。これを φ :

$\forall \bar{x} \exists \bar{y} \varphi(\bar{x}, \bar{y})$ と記す。 $\bar{x}$ は有限個の x_i の並び, $\bar{x} = (x_1, x_2, \dots, x_k)$ を表わし, その個数は $|\bar{x}| = k$ で表わす。

Σ の Herbrand universe すなわち Σ に出現する関数記号により構成されるすべての ground terms の集合 $H(\Sigma)$ とし, その中集合を $B[H(\Sigma)]$ と記す。

$\text{Graph}(\Sigma, \varphi)$ をつぎのように定める:

$$\text{graph}(\Sigma, \varphi) = \{(\bar{u}, \bar{v}) \mid \Sigma \models \varphi(\bar{u}, \bar{v}), (\bar{u}, \bar{v}) \in H^n(\Sigma)\},$$

$$|\bar{u}| = \ell, |\bar{v}| = m, n = \ell + m.$$

[定義 (Denotation of FS)] $(\Sigma, \varphi, \Delta)$ の Denotation は $\text{graph}(\Sigma, \varphi)$ により定まる $H^\ell(\Sigma) \rightarrow H^m(\Sigma)$ の写像 または $H^\ell(\Sigma) \rightarrow B[H(\Sigma)]^m$ の関数であり, $\text{Den}(\Sigma, \varphi, \Delta)$ または $\text{Den}(\Sigma, \varphi)$ と記す。 $\square$

3. Correctness of Program

[定義 (Completeness of FS)] $\text{Den}(\Sigma, \varphi)$ が (I) total (II) partial (III) unique (IV) multi-valued mapping であるとき, $(\Sigma, \varphi, \Delta)$ は (I) total (II) partial (III) unique (IV) multi-valued specification であるという。 Formal specification が total か

unique であるとき、complete であるという。☒

ここでは、ある計算機構で実行可能な言語による関数の表現を program と考える。Program の executability については4章で考察する。

Σ を nonlogical axioms とする first-order theory を Σ -theory と
いう。

[定義 (Σ -program)] Formulas の集合 Σ とするとき、
program は $H(\Sigma)$ 上の関数を、その denotation (それが表現する関数) の Σ -theory 内構成概念 または Σ -denotation としてもつとき、 Σ -program であるという。☒

Σ -program $\mathcal{P}$ の Σ -denotation の graph および domain を、それぞれ、 $\text{graph}(\mathcal{P}; \Sigma)$, $\text{dom}(\mathcal{P}; \Sigma)$ と記す。

[定義 (Correctness of Program)] Program $\mathcal{P}$, formal specification (Σ, ψ, Δ) とするとき、 $\mathcal{P}$ が Σ -program であり、

$$\text{graph}(\mathcal{P}; \Sigma) \subseteq \text{graph}(\Sigma, \psi)$$

$$\text{かつ (I) } \text{dom}(\mathcal{P}; \Sigma) = \text{dom}(\text{Den}(\Sigma, \psi))$$

$$\text{(II) } \text{dom}(\mathcal{P}; \Sigma) \subsetneq \text{dom}(\text{Den}(\Sigma, \psi))$$

であるとき、 $\mathcal{P}$ は (Σ, ψ, Δ) に関して (I) strongly correct
(II) weakly correct であるという。☒

4. Expressibility of Formal Specification and Program

Program synthesis において、correctness とともに、構成され

る program の executability は重要な概念である。Program の executability は特定の計算機構すなわちその計算機構の固有の言語に依存する。ここでは、あるクラスの計算機構の abstract machine を想定し、executability の作業概念として、その abstract language による expressibility の概念を用いる。

Logical program synthesis において、関数記号および述語記号の executability は、primitive symbols の宣言により定められる。言語の表現形式は recursive type と iterative type に大別される。ここでは、abstract language として logical program synthesis においてより基本的であると考えられる recursive type language について考察する。

$\Gamma: (\Sigma, \Psi, \Delta)$ とするとき、新しく有限個の関数変数記号を導入し、変数記号、 Γ の関数記号および関数変数記号により構成される term を Γ -term, $\Delta \setminus \{=\}$ に属する述語記号および Γ -term により構成される quantifier を含まない formula を Δ -formula という。

[定義 (Γ -expression)] k 個の関数変数記号 $F_1, F_2, \dots, F_k$, $\Gamma: (\Sigma, \Psi, \Delta)$ とする。 $\bar{F}$ および $\bar{F}_i$ を Γ -term の k -tuple, α_i を Δ -formula とするとき、

$$\{(\bar{F}_i, \alpha_i) \mid i=1, 2, \dots, n\} \text{ または } \bar{F}$$

を Γ -expression といい、 $\tau^*[\bar{F}](\bar{x})$ と表わす。 $\bar{x}$ は出現する

ψ の入力変数記号の並び, $\bar{F}:(F_1, F_2, \dots, F_k)$ である。☒

D_1, D_2 を集合とするとき, $D_1 \rightarrow D_2$ のすべての partial function の集合を $[D_1 \rightarrow D_2]$ と記す。

Γ -expression $\tau^*[\bar{F}](\bar{x})$ は, $|\bar{x}|=l$, $|\bar{F}|=k$ とするとき, $[H^l(\Sigma) \rightarrow B[H(\Sigma)]]^k$ 上の functional または $[H^l(\Sigma) \rightarrow B[H(\Sigma)]^k]$ 上の functional を表わす。

[定義 (Δ -recursive expressibility of FS)] Γ -expression $\tau^*[\bar{F}](\bar{x})$ により記述される system of recursive definitions:

$$\bar{F}(\bar{x}) \Leftarrow \tau^*[\bar{F}](\bar{x})$$

で, その least fixed point $\bar{H}:(H_1, H_2, \dots, H_k)$ が存在し, $\bar{H}$ の適当な部分列を $\bar{F}:(F_1, F_2, \dots, F_m)$, $m \leq k$ とするとき,

$$\bar{F} \equiv \text{Den}(\Gamma)$$

なるとき, Γ は Δ に関して recursive expressible であるという。その system of recursive definitions を Γ の complete recursive expression, $\bar{F} \not\equiv \text{Den}(\Gamma)$ なる system of recursive definitions を Γ の incomplete recursive expression という。☒

ここで定義された概念は, formal specification の denotation および graph (Σ, φ) の Δ -recursive expressibility である。

[定義 (Recursive program of FS)] Γ の (I) complete (II) incomplete recursive expression に出現する Γ -expression $\tau^*[\bar{F}](\bar{x})$ から導出される Γ -expression $\tau[\bar{F}](\bar{x})$ が

- i. $[H^k(\Sigma) \rightarrow H^k(\Sigma)]$ 上の functional を表わし, かつ
 ii. その system of recursive definitions $\bar{F}(\bar{x}) \Leftarrow \tau[\bar{F}](\bar{x})$ の
 least fixed point $\bar{h}$ の適当な部分列を $\bar{f}$ とするとき,

$$\forall \bar{x} \quad \bar{f}(\bar{x}) \in \text{Den}(\Gamma)(\bar{x})$$

なるとき, その system of recursive definitions $\bar{F}(\bar{x}) \Leftarrow \tau[\bar{F}](\bar{x})$ を
 Γ の (I) complete (II) incomplete recursive program という。☒
 定義より明らかに つぎの命題が成立する。

〔定理〕 Γ の recursive program は Γ に関して correct である。

5. Principle of Logical Program Synthesis

Formal specification から Γ -recursive expression を構成するための
 deductive system の拡張およびその機能の適用法について考
 察する。

First-order logic に関する Gödel's Completeness Theorem より
 Model theory における logical implication の relation $\models$ と
 Proof theory における derivability の relation $\vdash$ は equivalent
 である。すなわち, Complete な first-order calculus K が存在し
 $\{(\bar{u}, \bar{v}) \mid \Sigma \vdash_K \varphi(\bar{u}, \bar{v}), (\bar{u}, \bar{v}) \in H^n(\Sigma)\} = \text{graph}(\Sigma, \varphi)$
 が成り立つ。

〔Principle I (case division)〕 Formulas の集合 Σ , $\bar{x}$ を
 free variables とある formula φ とあるとき,

$$\Sigma \vdash \exists \bar{x} \varphi(\bar{x})$$

ならば、その proof から定まる logical instance (most general instance) を構成する手続きが存在し、

$$\Sigma \vdash \forall \bar{x} (\alpha(\bar{x}) \supset (\varphi(\bar{t}_1) \vee \varphi(\bar{t}_2) \vee \cdots \vee \varphi(\bar{t}_k)))$$

$$\text{かつ } \text{not} \vdash \forall \bar{x} (\varphi(\bar{t}_i) \supset \varphi(\bar{t}_j)) \quad \text{if } i \neq j$$

$$k \geq 1, \quad \bar{t}_i: \Sigma \text{ の term の有限列.}$$

なる α および $\bar{t}_1, \bar{t}_2, \dots, \bar{t}_k$ を求めることができる。 $k=1$ のとき、元の proof を Basic proof という。さらに、適当な k 個の formulas $\alpha_1, \alpha_2, \dots, \alpha_k$ を求め、 k 個の basic proofs

$$\Sigma \vdash \forall \bar{x} (\alpha_i(\bar{x}) \supset \varphi(\bar{t}_i)), \quad i=1, 2, \dots, k$$

に分けることができる。□

[Principle II (recursion)] i. First-order theory の definability の partial definability への拡張。 ii. First-order theory の extensions by definitions の拡張、すなわち、新しい関数記号とその defining axiom による theory の extension における existence condition および uniqueness condition を削除し、それに代わるものとして、functional による system of recursive definitions とその least fixed point の概念を適用する。□

Principle I, II により、formal specification が partial または multi-valued でも、その recursive expression を求めることが可能になる。Recursion を含むので、 $\text{graph}(\Sigma, \varphi)$ に対する表現能力は Δ によって定まる。

[Principle III (input variables)] (Σ, ψ, Δ) において,
 $\varphi(\bar{x}, \bar{y})$ の入力変数記号 $\bar{x} = (x_1, x_2, \dots, x_e)$ と basic proof
 から導出された $\alpha_i \supset \varphi(\bar{t}_{i1}, \bar{t}_{i2})$, $\bar{t}_{i1} = (t_{i11}, t_{i12}, \dots, t_{i1e})$ か
 ら, Γ -expression の要素 $(\bar{t}_{i2}, \alpha_i \wedge x_1 = t_{i11} \wedge \dots \wedge x_e = t_{i1e})$
 を求める。□

Principle III により, basic proof への制約がなくなる。

[Principle IV (= rule)] Theory の新しい inference rule と
 して, equality introduction rule を追加する。□

6. General Procedure of Logical Program Synthesis

Formal specification $\Gamma: (\Sigma, \psi, \Delta)$ とする。

Step 1. Σ と consistent な 適当な^{#1} 有限個の Σ -formulas
 $\varphi_0 (= \psi), \varphi_1, \dots, \varphi_k$ および, 適当な^{#2} 有限個の関数変数記号
 $F_1, F_2, \dots, F_n$ を導入し, それらの defining axioms により,
 Σ -theory を拡張する (Principle II);

$$\Sigma^+ : \Sigma \cup \{ \forall \bar{x} \varphi(\bar{F}), \forall \bar{x}_1 \varphi_1(\bar{F}_1), \dots, \forall \bar{x}_k \varphi_k(\bar{F}_k) \},$$

φ_i は ψ_i の matrix.

Step 2. 各 i ($i=0, 1, \dots, k$) について, 適当な^{#3} 有限個の
 $\Delta \cup \{=\}$ -formula α_{ij} を構成し, 対応する basic proofs から
 つぎの集合を求める (Principle I, IV):

$$\gamma_i = \{ (\bar{t}_{j1}, \bar{t}_{j2}, \alpha_{ij}) \mid \Sigma^+ \vdash \forall \bar{x}_i (\alpha_{ij} \supset \varphi_i(\bar{t}_{j1}, \bar{t}_{j2})) \},$$

$$\bar{t}_{j1} \in T^{L_i}(\Sigma^+), \bar{t}_{j2} \in T^{M_i}(\Sigma^+), T(\Sigma^+) : \Sigma^+ \text{ の term の集合.}$$

Step 3. 各 $\mathcal{I}_i$ から Γ -expression を求める (Principle III):

$$\tau_i^*[\bar{F}, \bar{F}_1, \dots, \bar{F}_k](\bar{x}_i) = \{(\bar{t}_{ij}, \bar{\alpha}_{ij}^+) \mid \bar{\alpha}_{ij}^+ = \alpha_{ij} \wedge \bar{x}_i = \bar{t}_{j1}\}$$

Step 4. つぎの system of recursive definitions をなす

Γ -recursive expression $\mathcal{P}^*[\bar{F}]$ を求める:

$$\mathcal{P}^*[\bar{F}] : \bar{F}(\bar{x}) = \tau_0^*[\bar{F}, \bar{F}_1, \dots, \bar{F}_k](\bar{x})$$

$$\bar{F}_i(\bar{x}_i) = \tau_i^*[\bar{F}, \bar{F}_1, \dots, \bar{F}_k](\bar{x}_i), \quad i=1, 2, \dots, k.$$

Step 5. つぎの Γ の recursive program を求める:

$$\mathcal{P}[\bar{F}] : \bar{F}(\bar{x}) = \tau_0[\bar{F}, \bar{F}_1, \dots, \bar{F}_k](\bar{x})$$

$$\bar{F}_i(\bar{x}_i) = \tau_i[\bar{F}, \bar{F}_1, \dots, \bar{F}_k](\bar{x}_i), \quad i=1, 2, \dots, k.$$

#1, 2, 3 について: 各々について, 目的物を構成する

algorithm または semi-algorithm の存在 および 手続きの細部は, 採用する deductive system とその拡張の仕方に依存する。

Resolution-refutation base の extended deductive system においては, それらの algorithm または semi-algorithm が知られている。とくに #3 については, 可能なかぎり (Δ に依存する) complete recursive expression を求めるように basic proofs を選ぶ semi-algorithm が存在する。

参考文献

[1] 謝 (1976): プログラムの自動合成と定義可能性, 数理研 講究録 270

[2] 謝 (1975): プログラム・シヤシスのための情報抽出系, 信学会 AL75-13

- [3] 謝 (1975): Mechanical Flowchart Synthesis 1-712, 信学会 AL 75-2
- [4] 謝 (1975): Resolution-Refutation 21-53 M.P.S., 信学会 AL 74-40
- [5] Chang & Lee (1973): Symbolic Logic and Mechanical Theorem Proving, Academic Press
- [6] Carnap, R. (1942): Introduction to semantics, Cambridge, Mass.,
- [7] Carnap, R. (1943): Formalization of logic, Cambridge, Mass.,
- [8] Kleene, S.C. (1967): Introduction to Metamathematics, N.H.
- [9] Kleene, S.C. (1967): Mathematical Logic, John Wiley & Sons.
- [10] Manna, Z. (1974): Mathematical Theory of Computation, McGraw-Hill
- [11] Tarski, A. (1969): Logic, Semantics, Meta-Mathematics, Oxford.
- [12] Takeuti, G. (1975): Proof Theory, North-Holland.
- [13] Davis, M. (1958): Computability and unsolvability, McGraw-Hill
- [14] 相沢輝 (1970): 計算理論の基礎, 総合図書
- [15] 神野, 内井 (1976): 論理学, モデル理論と歴史的背景, ミネルポ書房.
- [16] Wilder, R.L. (1965): Introduction to the Foundations of Math., John Wiley