数理解析研究所講究録
第421巻 1981年 268-283

LISPプログラム自動合成のーシステム

電気通信大学　永井　雅人
西澤　輝泰

1. 序

プログラム自動合成と呼ばれるものは、述語論理式等の計算方法を与えていない表現からプログラムを導くものと、有限個の入出力例からプログラムを推測するものに大別される。後者ではオートマトン論の諸結果が重要な役割を果しており、前者では主としてresolutionによる定理の自動証明の手法を応用している([1]~[6])。しかし、前者についても我々はオートマトン論に着目してよい。なぜならば、正規表現の微分から有限オートマトンを得る過程は明らかに1つのプログラム自動合成である。この自動合成過程は場合分けと帰納法の適用によって構成されている。我々はこうした考え方をLISPプログラムの自動合成に適用し、AE型の述語論理式に対し、場合分けを行う変形規則と帰納法の適用等による変形を行い、if-then-elseの入れ子構造を持つプログラムを合成するシス

テムを試作した。このために、2種類の場合分けの構造（if-then-else と IF-THEN-ELSE）を持つN－表現なるものを考案し、用いることとした。

我々のシステムでは、対象となる述語と関数の引数値及び関数値はLISPのS式であるが、ここでのS式はリストヌはアトムのいずれかであると限定している。但し、nil は空リストであって、アトムとは考えない。又、述語の値はT(真)，F(偽)のいずれかとする。

2. N－表現の定義

(1) 式

述語記号 … 任意の記号 (atom, equalを含む)。

関数記号 … 任意の記号 (car, cdr, consを含む)

変数 … 任意の記号。

変数は自由変数と束縛変数に区別され、束縛変数はさらにA型束縛変数（∀で束縛される変数）とE型束縛変数（∃で束縛される変数）に分けられる。

但し、以上の記号に混同があってはならない。又、変数の種別はN－表現の入力時に宣言されている。述語記号と関数記号の区別は特に意識しない（入力時に陰に宣言されていると考えられる）。

項，原子式 … 上記の記号を用いた通常の定義。

基本式 … 原子式，¬原子式，T又はF（T，Fは真偽を表わす定数）。

式 … 基本式と∧，∨，¬及びカッコを用いた通常の論理式。

(2) 制限

F型基本式 … 束縛変数を含まない基本式。

A型基本式 … A型束縛変数を含み，E型束縛変数を含まない基本式。

E型基本式 … E型束縛変数を含み，A型束縛変数を含まない基本式。

F型制限 … 空，又はF型基本式と∧からなる論理式。

A型制限 … 空，又はA型基本式と∧からなる論理式。

E型制限 … 空，又はE型基本式と∧からなる論理式。

(3) 基本N－表現

節 … （節本体，E型制限）

節本体 … 式。

基本N－表現 … A型基本N－表現又はE型基本N－表現。

Wを **IF**（A，r_E）**THEN**（B，r_E）**ELSE**（C，r_E'）なる形式とするとき，A型基本N－表現は［W；r_A；r_F］，E型基本N－表現は［W；r_F］なる形式である。但し，(A，r_E)，(B，r_E)，

(C, r_E')：節，r_A：A型制限，r_F：F型制限である。又、E型基本N－表現の場合，WはA型束縛変数を含まないものとする。これらの意味は次により定められる。

W：**IF** (A, r_E) **THEN** (B, r_E) ***ELSE*** (C, r_E')

の意味$\widetilde{W}$は，

$(\exists\eta)(A \wedge B \wedge r_E) \vee (\neg(\exists\eta)(A \wedge r_E) \wedge (\exists\eta)(C \wedge r_E'))$

（但し，ηはWに現われるE型束縛変数全ての系列）となり、A型及びE型基本N－表現$[W; r_A; r_F]$および$[W; r_F]$の意味はそれぞれ

$r_F \supset (\forall\xi)(r_A \supset \widetilde{W})$　，　$r_F \supset \widetilde{W}$

（但し、ξはW及びr_Aに現われるA型束縛変数全ての系列）である。

(4) N－表現

F型式 … 束縛変数を含まない式。

N－表現 … F型式，基本N－表現又は(**if** N－表現 **then** N－表現 **else** N－表現），

(**if** U **then** V **else** W) の意味は、N－表現U，V，Wの意味を$\widetilde{U}$，$\widetilde{V}$，$\widetilde{W}$とすると

$(\widetilde{U} \wedge \widetilde{V}) \vee (\neg\widetilde{U} \wedge \widetilde{W})$

となる。

3. 変形規則

基本N－表現に対して場合分けを行う変形規則について述べる前に、それを正当化する基本定理を掲げる。

以下において、$(\exists\eta)[\mathrm{IF}\ A\ \mathrm{THEN}\ B\ \mathrm{ELSE}\ C]$ （A, B, C：論理式，η：変数列）の意味は

$$(\exists\eta)(A\wedge B)\vee(\neg(\exists\eta)A\wedge(\exists\eta)C)$$

である。

基本定理

p, W, A, B, C, r, r'：論理式，ξ, η：変数列．

f-1) $W \leftrightarrow (\mathrm{if}\ p\ \mathrm{then}\ W\ \mathrm{else}\ W)$

$\leftrightarrow (\mathrm{if}\ p\ \mathrm{then}\ p\supset W\ \mathrm{else}\ W)$

$\leftrightarrow (\mathrm{if}\ p\ \mathrm{then}\ W\ \mathrm{else}\ \neg p\supset W)$

$\leftrightarrow (\mathrm{if}\ p\ \mathrm{then}\ p\supset W\ \mathrm{else}\ \neg p\supset W)$

a-1) $(\forall\xi)W \leftrightarrow (\mathrm{if}\ (\forall\xi)[p\supset W]\ \mathrm{then}\ (\forall\xi)[\neg p\supset W]\ \mathrm{else}\ F)$

e-1) $(\exists\eta)[\mathrm{IF}\ A\wedge r\ \mathrm{THEN}\ B\wedge r\ \mathrm{ELSE}\ F]$

$\leftrightarrow (\mathrm{if}\ (\exists\eta)[\mathrm{IF}\ A\wedge r\wedge p\ \mathrm{THEN}\ B\wedge r\wedge p\ \mathrm{ELSE}\ F]$

$\mathrm{then}\ T$

$\mathrm{else}\ (\exists\eta)[\mathrm{IF}\ A\wedge r\wedge\neg p\ \mathrm{THEN}\ B\wedge r\wedge\neg p\ \mathrm{ELSE}\ F])$

e-2) $(\exists\eta)[\mathrm{IF}\ A\wedge r\ \mathrm{THEN}\ T\wedge r\ \mathrm{ELSE}\ C\wedge r']$

$\leftrightarrow (\mathrm{if}\ (\exists\eta)[\mathrm{IF}\ A\wedge r\wedge p\ \mathrm{THEN}\ T\wedge r\wedge p\ \mathrm{ELSE}\ F]$

$\mathrm{then}\ T$

$\mathrm{else}\ (\exists\eta)[\mathrm{IF}\ A\wedge r\wedge\neg p\ \mathrm{THEN}\ T\wedge r\wedge\neg p\ \mathrm{ELSE}\ C\wedge r'])$

e-3) $(\exists\eta)$[IF $A_{\wedge r}$ THEN $F_{\wedge r}$ ELSE $C_{\wedge r'}$]

$\leftrightarrow$ (if $(\exists\eta)$[IF $A_{\wedge r\wedge p}$ THEN $F_{\wedge r\wedge p}$ ELSE T]

then $(\exists\eta)$[IF $A_{\wedge r\wedge\neg p}$ THEN $F_{\wedge r\wedge\neg p}$ ELSE $C_{\wedge r'}$]

else F)

e-4) $(\exists\eta)$[IF $A_{\wedge r}$ THEN $B_{\wedge r}$ ELSE $C_{\wedge r'}$]

$\leftrightarrow$ (if $(\exists\eta)$[IF $A_{\wedge r}$ THEN $B_{\wedge r}$ ELSE F]

then T

else $(\exists\eta)$[IF $A_{\wedge r}$ THEN $F_{\wedge r}$ ELSE $C_{\wedge r'}$])

次に前提基本式を定義する。基本式 p が以下の①と②又は①と②′を満足するとき、p は前提基本式である。

① p が目的述語又は目的関数を含むならば、その引数の少くとも1つが car と cdr の空でない系列 α による $\alpha[x]$（x:変数）で表わされている。（但し、目的述語（関数）は主目的述語（関数）と副目的述語（関数）があり、主目的述語（関数）は入力時に宣言され、副目的述語（関数）はシステムによる合成過程で発生し、その時点で宣言される。）

② p は F 型基本式である。

②′ p は A 型又は E 型基本式であり、p が

atom[x], equal[x; α], equal[α; x]

（x:束縛変数, α:束縛変数を含まない項）のいずれかの

形又はこれらに¬のついた形をしている。

前提基本式pはpの型によりF型，A型，E型に分類される。

変形規則で用いる4つの変換μ，τ_F，τ_A，τ_Eを定義する。

°$\mu[A;p]$　（A：式，p：基本式）

Aに含まれるpをTで，$\neg p$をFで置き換える。

°$\tau_F[E;p]$，$\tau_A[E;p]$，$\tau_E[E;p]$

（E：基本N-表現，p：前提基本式）

pが$equal[x;\alpha]$のとき，Eの全てのxをαで置き換える。

pが$\neg equal[x;nil]$であり，Eの制限が$\neg atom[x]$を含むとき，

τ_F：xを$cons[car[x];cdr[x]]$で置き換え，制限に$\neg atom[cdr[x]]$を加える。

τ_A,τ_E：xを$cons[x_1;x_2]$（x_1,x_2はxと同じ型）で置き換え，制限に$\neg atom[x_2]$を加える。

pがそれ以外のとき，Eの制限にpを加える。

これらの変換については，以下のことが成り立つ。

$\mu[A;p]\wedge p \leftrightarrow A\wedge p$，　$p\supset\mu[A;p] \leftrightarrow p\supset A$

$\tau_F[[W;r_A;r_F];p] \leftrightarrow [W;r_A;r_F\wedge p]$

$\tau_F[[W;r_F];p] \leftrightarrow [W;r_F\wedge p]$

$\tau_A[[W;r_A;r_F];p] \leftrightarrow [W;r_A\wedge p;r_F]$

$\tau_E[[\mathbf{IF}\ (A,r_E)\ \mathbf{THEN}\ (B,r_E)\ \mathbf{ELSE}\ (C,r_E');r_F];p]$

$\leftrightarrow$ [IF $(A, r_{E \wedge p})$ THEN $(B, r_{E \wedge p})$ ELSE $(C, r_{E'})$; r_F]

以上により次の変形規則が正当化される（即ち、N－表現に現われる基本N－表現に、次のような変形を加えても、全体の意味は不変である）。

変形規則

f-1) pをF型前提基本式とする。

[IF (A, r_E) THEN (B, r_E) ELSE $(C, r_{E'})$; r_A; r_F]

$\rightarrow$ (if p

then τ_F[[IF $(\mu[A; p], r_E)$ THEN $(\mu[B; p], r_E)$

ELSE $(\mu[C; p], r_{E'})$; r_A; r_F]; p]

else τ_F[[IF $(\mu[A; \neg p], r_E)$ THEN $(\mu[B; \neg p], r_E)$

ELSE $(\mu[C; \neg p], r_{E'})$; r_A; r_F]; $\neg p$])

E型基本N－表現についても同様。

a-1) pをA型前提基本式とする。

[IF (A, r_E) THEN (B, r_E) ELSE $(C, r_{E'})$; r_A; r_F]

$\rightarrow$ (if τ_A[[IF $(\mu[A; p], r_E)$ THEN $(\mu[B; p], r_E)$

ELSE $(\mu[C; p], r_{E'})$; r_A; r_F]; p]

then τ_A[[IF $(\mu[A; \neg p], r_E)$ THEN $(\mu[B; \neg p], r_E)$

ELSE $(\mu[C; \neg p], r_{E'})$; r_A; r_F]; $\neg p$]

else F)

e-1) pをE型前提基本式とする。

[IF (A, r_E) THEN (B, r_E) ELSE (F,); r_F]

→ (if τ_E[[IF (μ[A; p], r_E) THEN (μ[B; p], r_E)

ELSE (F,); r_F]; p]

then T

else τ_E[[IF (μ[A; ¬p], r_E) THEN (μ[B; ¬p], r_E)

ELSE (F,); r_F]; ¬p])

e-2) ～ e-4) (略. 基本定理 e-2)～ e-4) に基づく)

4. 制限の除去

入力された基本N－表現を変形規則等で変形を行った結果のN－表現に含まれる基本N－表現において, 以下のように各制限を取除くことが可能となる (即ち、変形の過程におけるN－表現において、それに現われる任意の基本N－表現に次のような変形を加えても、全体のN－表現の値は不変である)。

節本体にA型束縛変数が存在しないとき、

[IF (A, r_E) THEN (B, r_E) ELSE (C, r_E'); r_A; r_F]

→ [IF (A, r_E) THEN (B, r_E) ELSE (C, r_E'); r_F]

節本体に束縛変数が存在しないとき.

[IF (A, r_E) THEN (B, r_E) ELSE (C, r_E'); r_F]

→ (if A then B else C)

5. 帰納法適用

まず、擬変数を

〈擬変数〉 ::=〈自由擬変数〉|〈束縛変数〉

〈自由擬変数〉 ::=〈自由変数〉| car[〈自由擬変数〉]|
cdr[〈自由擬変数〉]

と定義する。

今、ある基本N－表現E_0とそれを変形して得られるN－表現に含まれる基本N－表現Eにおいて、次のような擬変数に対する代入σが存在するとき、Eに対してE_0に関する帰納法を適用する。

i) $E_0\cdot\sigma = E$

ii) σにおいて、束縛変数は同型の束縛変数に、自由擬変数πはcarとcdrの系列αによる$\alpha[\pi]$に対応し、少なくとも1つの自由擬変数において系列αが空でない。

例.

sublist[x; y]に相当する基本N－表現E_0:

[IF (equal[y; append[u; append[x; v]]], ¬atom[u] ∧ ¬atom[v])

THEN (T, ¬atom[u] ∧ ¬atom[v]) **ELSE** (F,); ¬atom[x] ∧ ¬atom[y]]

E_0を変形して得られるN－表現に含まれる基本N－表現E:

[IF (equal[cdr[y]; append[u_2; append[x; v]]], ¬atom[u_2] ∧ ¬atom[v])

THEN (T, ¬atom[u_2] ∧ ¬atom[v]) **ELSE** (F,); ¬atom[x] ∧ ¬atom[cdr[y]]]

(x, y: 自由変数, u, v, u_2: E型束縛変数)

このとき E を以下のように変形する。

[IF (sublist[x; cdr[y]],) THEN (T,) ELSE (F,);]

E_0 が表わす述語（関数）が存在しないときは、新たに述語記号（関数記号）を導入し、補助述語（補助関数）を構成する。

6. 等式，基本式の変形

節本体に equal[α;β] なる原子式が存在するとき、α, β を各各、cons[α_1; α_2], cons[β_1; β_2] と表わすことが可能ならば、

equal[cons[α_1; α_2]; cons[β_1; β_2]] → equal[α_1; β_1] ∧ equal[α_2; β_2]

とすることができる。問題となるのは

α → cons[α_1; α_2], β → cons[β_1; β_2]

なる変形である。このために、システムに登録されている知識（述語，関数等の性質）を用いる。

知識の例

(append[x; y].) → ((y . equal[x; nil])

(cons[car[x]; append[cdr[x]; y]]. ¬equal[x; nil]))

変形の例

equal[y; append[x; v]]

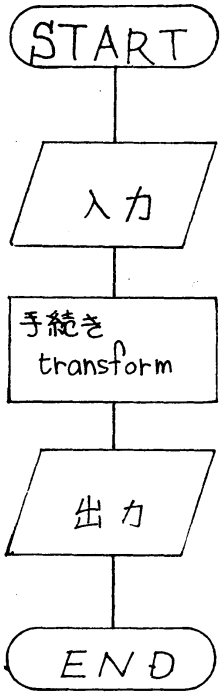
→ (equal[x; nil] ∧ equal[y; v])

∨ (¬equal[x; nil] ∧ equal[y; cons[car[x]; append[cdr[x]; v]]])

この知識を用いた変形を上のように等式に適用したときを

等式の変形，基本式に適用したときを基本式の変形という。

7. アルゴリズム概要



手続き transform

終了判定* Yes → 終り

↓No

帰納法適用可能 Yes → 帰納法適用

↓No

前提基本式が存在 Yes → 変形規則の適用

↓No

等式が存在 Yes → 等式の変形

↓No

基本式の変形

＊終了判定は、各節本体に束縛変数が存在しないということ。

8. 合成例

sublist[x; y] : "リストyにリストxがサブリストとして存在する" という述語。

入力

280

predicate: sublist[x; y];

$(\exists u, v)$ [**IF** (equal[y; append[u; append[x; v]]], ¬atom[u] ∧ ¬atom[v])

THEN (T, ¬atom[u] ∧ ¬atom[v])

ELSE (F,); ¬atom[x] ∧ ¬atom[y]]

ここで用いる知識

(x .) → ((nil . equal[x; nil]) (x . ¬equal[x; nil] ∧ atom[x])

(cons[car[x]; cdr[x]] . ¬equal[x; nil] ∧ ¬atom[x]))

(append[x; y] .) → ((y . equal[x; nil])

(cons[car[x]; append[cdr[x]; y]] . ¬equal[x; nil]))

変形の過程 (但し、ここでは THEN と ELSE の間にある節の E 型制限は省略する。又、不必要な制限も省略する。)

[**IF** (equal[y; append[u; append[x; v]]], ¬atom[u] ∧ ¬atom[v])

THEN (T,) **ELSE** (F,); ¬atom[x] ∧ ¬atom[y]]

等式の変形

→[**IF** ((equal[u; nil] ∧ equal[y; append[x; v]])

∨ (equal[u; nil] ∧ equal[y; cons[car[u]; append[cdr[u]; append[x; v]]]]),

¬atom[u] ∧ ¬atom[v])

THEN (T,) **ELSE** (F,); ¬atom[x] ∧ ¬atom[y]]

e-1 (equal[u; nil]) …これは前提基本式 equal[u; nil] を用いた変形規則 e-1 の適用を示す。以下同様。

→ (**if** [**IF** (equal[y; append[x; v]], ¬atom[v]) **THEN** (T,)

ELSE (F,); ¬atom[x] ∧ ¬atom[y]]
①

then T

else [IF (equal[y; cons[u1; append[u2; append[x; v]]]], ¬atom[v] ∧ ¬atom[u2])

THEN (T,) ELSE (F,); ¬atom[x] ∧ ¬atom[y]])
⑤

①からの変形

等式の変形

→[IF ((equal[x; nil] ∧ equal[y; v])

∨(¬equal[x; nil] ∧ equal[y; cons[car[x]; append[cdr[x]; v]]]), ¬atom[v])

THEN (T,) ELSE (F,); ¬atom[x] ∧ ¬atom[y]]

f-1 (equal[x; nil])

→(if equal[x; nil]

then [IF (equal[y; v], ¬atom[v]) THEN (T,) ELSE (F,); ¬atom[y]]
②

else [IF (equal[y; cons[car[x]; append[cdr[x]; v]]], ¬atom[v])

THEN (T,) ELSE (F,); ¬atom[y] ∧ ¬atom[cdr[y]]])
③

②からの変形

e-1(equal[y; v])

→(if [IF (T,) THEN (,) ELSE (F,);]

then T else [IF (F,) THEN (T,) ELSE (F,);])

③からの変形を終けると以下のようなN-表現を得る。

(if ¬equal[y; nil]

then (if equal[car[y]; car[x]]

then [IF (equal[cdr[y]; append[cdr[x]; v]], ¬atom[v])
THEN (T,) ELSE (F,); ¬atom[cdr[x]]∧¬atom[y]] ④
else [IF (F,) THEN (T,) ELSE (F,);])
else [IF (F,) THEN (T,) ELSE (F,);])

ここで④は①に関して帰納法適用可能であるから、新たに述語記号 sublist1 を導入して④を以下のように変形する。

[IF (sublist1[cdr[y]; cdr[x]],) THEN (T,) ELSE (F,);]

⑤からの変形

略

出力

sublist[x; y]
= (if sublist1[y; x] then T
else (if ¬equal[y; nil] then sublist[x; cdr[y]] else F))

sublist1[y; x]
= (if equal[x; nil] then T
else (if ¬equal[y; nil]
then (if equal[car[y]; car[x]] then sublist1[cdr[y]; cdr[x]]
else F)
else F))

謝辞

本研究に関して、京大数理解析研究所高須達教授には、数次にわたるセミナーを設けていただき、数々の御批判、御助言をいただきました。ここに厚く御礼申し上げます。

参考文献

[1] Nilsson, N.J. : Problem-Solving Methods in Artificial Intelligence (1971) McGraw-Hill

[2] Chang, C.L. & R.C.T. Lee : Symbolic Logic and Mechanical Theorem Proving (1973) ACADEMIC PRESS

[3] Cartwright, R. & J. McCarthy : Recursive Program as Function in a First Order Theory (1979) Lecture Note in Computer Science 75 springer-Verlag

[4] Lee, R.C.T., C.L. Chang & R.J. Waldinger : An Improved Program-Synthesizing Algorithm and Its Correctness (1974) C.A.C.M. vol. 17. No. 4

[5] Luckham, D. & N.J. Nilsson : Extracting Information from Resolution Proof Trees (1971) Artificial Intelligence 2

[6] Darlington, J.L. : Automatic Program Synthesis in Second Order Logic (1973) Proc. I.J.C.A.I.