ダイクストラの検証法

ダイクストラ Edger Wybe Dijkstra (1930.5.11-2011.8.6)

構造化プログラミング, goto有害論, 分散プログラミング,

形式的検証, プログラム導出 etc.

ダイクストラの方法

最弱前条件twp(P, B)を用いて検証を行う.

非決定的プログラム言語を用いる.

ちなみに, ダイクストラ法とは

グラフ理論における最短経路を求めるアルゴリズムの一つであり,

プログラム理論とは無関係なものである.

最弱前条件と表明付きプログラム

定義

$pwp_M(P, B) = \{\rho \mid Exec_M(P, \rho, \rho')$なるどんな$\rho'$に対しても, M, $\rho' \models B\}$

$twp_M(P, B) = \{\rho \mid$ある$\rho'$が存在して $Exec_M(P, \rho, \rho') \wedge \rho \in pwp_M(P, B)\}$

状態の集合として定義されている.

「前条件として最も弱い命題」を表す, すなわち

$$M \models_{tot} \langle\!\langle A \rangle\!\rangle P \langle\!\langle B \rangle\!\rangle \Leftrightarrow \text{どんな}\rho\text{に対しても } M, \rho \models A \text{ ならば } \rho \in twp_M(P, B)$$

である.

(ϕがψより弱い(weaker)とは$\psi \supset \phi$が成り立つことである.)

$twp_M(P, B)$を表明として使うことができるのであれば

(すなわち, 言語が充分な記述能力を持つとすれば)

$$M \models_{tot} \langle\!\langle A \rangle\!\rangle P \langle\!\langle B \rangle\!\rangle \Leftrightarrow M \models (A \supset twp_M(P, B))$$

$\langle\!\langle A \rangle\!\rangle P \langle\!\langle B \rangle\!\rangle$のかわりに $A \supset twp_M(P, B)$を用いる.

以降, モデル M は固定されているものとし，$Exec_M$, twp_M における M の表記は省略し，Exec, twpと表す.

最弱前条件の一般法則

$$\frac{B_1 \supset B_2}{twp(P, B_1) \supset twp(P, B_2)}$$

すなわち

$M \models_{tot} (B_1 \supset B_2) \Rightarrow M \models (twp(P, B_1) \supset twp(P, B_2))$

$$twp(P, A \wedge B) = twp(P, A) \wedge twp(P, B)$$

$$twp(P, A \vee B) = twp(P, A) \vee twp(P, B)$$

$$twp(P, False) = False$$

最弱前条件の特徴付けと検証

ホーア論理では公理と推論規則を使うところを,

ダイクストラの方法ではtwp(P, B)を特徴つける等式を用いる.

代入文の特徴付けは

$$\mathrm{twp}(a=t, B) = B[t/a]$$

である.

ここから

$$B[t/a] \supset \mathrm{twp}(a=t, B)$$

が成り立つ.

これは代入文の公理

$$《 B[t/a] 》\ a=t\ 《 B 》$$

に対応する.

同様に,

$twp(skip, B) = B$

$twp(\text{if } C \{P\} \text{ else } \{Q\}, B) = (C \supset twp(P, B)) \wedge (\neg C \supset twp(Q, B))$

$twp(P_1; \ldots ; P_n, B) = twp(P_1, \ldots twp(P_{n-1}, twp(P_n, B))\ldots)$

$twp(\text{while } C \{P\}, B) = \exists n H_n(C, P, B)$

ただし, H_nは

$H_0(C, P, B) = \neg C \wedge B$

$H_{n+1}(C, P, B) = C \wedge twp(P, H_n(C, P, B))$

で定義される.

条件文の規則をダイクストラ流に表現

$$\frac{(C \wedge A) \supset twp(P, B) \quad (\neg C \wedge A) \supset twp(Q, B)}{A \supset twp(\text{if } C \{P\} \text{ else } \{Q\}, B)}$$

これを証明する.

この推論の前提は

$$((C \wedge A) \supset twp(P, B)) \wedge ((\neg C \wedge A) \supset twp(Q, B))$$

同値変形すると

$$A \supset ((C \supset twp(P, B)) \wedge (\neg C \supset twp(Q, B)))$$

条件文の特徴付けの等式

$$twp(\text{if } C \{P\} \text{ else } \{Q\}, B) = (C \supset twp(P, B)) \wedge (\neg C \supset twp(Q, B))$$

を用いると

$$A \supset twp(\text{if } C \{P\} \text{ else } \{Q\}, B)$$

これは条件文の推論の結論である.

同様に, while文の規則をダイクストラ流に表現

$$\frac{(C \wedge A) \supset twp(P, A)}{A \wedge twp(\text{while } C \ \{P\}, True) \supset twp(\text{while } C \ \{P\}, \neg C \wedge A)}$$

twp(P, True)は,「現在の状態でPを実行すると停止する」を意味する.
A⊃twp(while C{P}, True)はAの元での停止性を表す.
すなわち, この推論規則の結論はプログラムが停止することを仮定している.

一方, A⊃twp(while C {P}, ¬C∧A)はAが成り立つ状態でwhileを実行すると
停止して後条件が¬C∧Aであることを表す.
したがって, この結論は$\vdash_{tot}$ ⦇A⦈ while C {P}⦇¬C∧A⦈と同値.
完全正当性の推論規則である.

非決定的プログラムとその検証

ガード付きコマンド（guarded command）

条件文の代わりに以下の文を用いる

IF ≡ if $C_1 \to P_1$ □ $C_2 \to P_2$ □ … □ $C_n \to P_n$ fi

C_i : ガード

$C_i \to P_i$: ガード付きコマンド

条件C_iが成り立つときにPiを実行する

ガード付きコマンドの実行は非決定的

C_iが成り立つのならどれを実行してもよい.

成り立つガードがないときは停止しない.

$Exec(IF, \rho, \rho') \equiv$

$\rho \models C_1 \wedge Exec(P_1, \rho, \rho') \vee \rho \models C_2 \wedge Exec(P_2, \rho, \rho') \vee \ldots \vee \rho \models C_n \wedge Exec(P_n, \rho, \rho')$

whileの代わりに以下のDOを用いる.

$$DO \equiv \text{do } C_1 \to P_1 \ \square \ C_2 \to P_2 \ \square \ \cdots \ \square \ C_n \to P_n \text{ od}$$

ガードC_iが成立すればどのガード付きコマンド $C_i \to P_i$ を実行してもよい.
成り立つガードがある限り繰り返す.
成り立つガードがなくなれば正常終了する.

$twp(IF, B) = (C_1 \vee \ldots \vee C_n) \wedge (C_1 \supset twp(P_1, B)) \wedge \ldots \wedge (C_n \supset twp(P_n, B))$

$twp(DO, B) = \exists n K_n(DO, B)$

$K_0(DO, B) = B \wedge \neg(C_1 \vee \ldots \vee C_n)$

$K_{n+1}(DO, B) = twp(IF, K_n(DO, B)) \vee K_0(DO, B)$

ここで

$IF \equiv \text{if } C_1 \to P_1 \ \square \ C_2 \to P_2 \ \square \ \cdots \ \square \ C_n \to P_n \text{ fi}$

$DO \equiv \text{do } C_1 \to P_1 \ \square \ C_2 \to P_2 \ \square \ \cdots \ \square \ C_n \to P_n \text{ od}$

のことである.

while文の規則に類似したDO文の規則

$$\frac{((C_1 \vee \ldots \vee C_n) \wedge B) \supset twp(IF, B) \qquad ((C_1 \vee \ldots \vee C_n) \wedge B \wedge bound=b) \supset twp(IF, bound<b)}{B \supset twp(DO, B \wedge \neg(C1 \vee \ldots \vee Cn))}$$

twpの計算

twpの特徴付けを用いて, twp(P, B)が計算できる.

例えば, 代入文の特徴付けはtwp(a=t, B) =B[t/a]であったので,

具体的にBを与えることにより,

$$twp(a=a*3,\ a=3) = (a=3)[a*3/a] = (a*3=3)$$

となる.

最弱前条件の特徴付け(再掲)

$twp(a=t, B) = B[t/a]$

$twp(skip, B) = B$

$twp(\text{if } C \{P\} \text{ else } \{Q\}, B) = (C \supset twp(P, B)) \wedge (\neg C \supset twp(Q, B))$

$twp(P_1; \ldots ; P_n, B) = twp(P_1, \ldots twp(P_{n-1}, twp(P_n, B))\ldots)$

$twp(\text{while } C\{P\}, B) = \exists n H_n(C, P, B)$

ただし,

$H_0(C, P, B) = \neg C \wedge B$

$H_{n+1}(C, P, B) = C \wedge twp(P, H_n(C, P, B))$

で定義される.

$twp(IF, B) = (C_1 \vee \ldots \vee Cn) \wedge (C_1 \supset twp(P_1, B)) \wedge \ldots \wedge (Cn \supset twp(Pn, B))$

ただし, $IF \equiv \text{if } C_1 \to P_1 \ \square \ C_2 \to P_2 \ \square \ \cdots \ \square \ C_n \to P_n \text{ fi}$

$twp(DO, B) = \exists n K_n(DO, B)$

ただし,

$DO \equiv \text{do } C_1 \to P_1 \ \square \ C_2 \to P_2 \ \square \ \cdots \ \square \ C_n \to P_n \text{ od}$

$K_0(DO, B) = B \wedge \neg(C_1 \vee \ldots \vee C_n)$

$K_{n+1}(DO, B) = twp(IF, K_n(DO, B)) \vee K_0(DO, B)$

例題

DO: do　　x>2 → x=x-1; i=i+1

　　□　x>3 → x=x-2; i=i+1 od

B: i≤2

を考える. ただしx, iは自然数とする.

twp(DO, B)を計算する.

twp(DO, B) = $\exists n K_n$(DO, B)

K_0(DO, B) = i≤2∧¬(x>2∨x>3)=x≤2∧i≤2

K_1(DO, B) = twp(IF, K0(DO, B))∨ K0(DO, B) = twp(IF, i≤2∧x≤2)∨x≤2∧i≤2

twp(IF, i≤2∧x≤2) = (x>2∨x>3) ∧(x>2⊃twp(x=x-1; i=i+1 , x≤2∧i≤2))

　　∧(x>3⊃twp(x=x-2; i=i+1 , x≤2∧i≤2))

twp(x=x-1; i=i+1 , x≤2∧i≤2)=

　　twp(x=x-1, twp(i=i+1, x≤2∧i≤2)) = twp(x=x-1, i+1≤2∧x≤2) = i≤1∧x≤3

twp(x=x-2; i=i+1 , x≤2∧i≤2)= i≤1∧x≤4

∴ twp(IF, i≤2∧x≤2) = (x>2∨x>3)∧(x>2⊃ i≤1∧x≤3)∧(x>3⊃ i≤1∧x≤4)

　　= (x=3∨x=4)∧i≤1

∴ K_1(DO, B) = (x=3∨x=4)∧i≤1 ∨ x≤2∧i≤2

同様に

$K_2(DO, B) = twp(IF, K_1(DO, B)) \vee K0(DO, B)$

ただし $K1(DO, B) = (x=3 \vee x=4) \wedge i \leq 1 \vee x \leq 2 \wedge i \leq 2$

$twp(IF, K_1(DO, B))$

$= (x>2 \vee x>3) \quad \wedge (x>2 \supset twp(x=x-1; i=i+1, K_1(DO, B)))$

$\wedge (x>3 \supset twp(x=x-2; i=i+1, K_1(DO, B)))$

$= (x>2 \vee x>3) \wedge (x>2 \supset (x=4 \vee x=5) \wedge i \leq 0 \vee x \leq 3 \wedge i \leq 1)$

$\wedge (x>3 \supset (x=5 \vee x=6) \wedge i \leq 0 \vee x \leq 4 \wedge i \leq 1)$

$= x=3 \wedge i \leq 1 \vee x=4 \wedge i \leq 1 \vee x=5 \wedge i \leq 0 \vee x=6 \wedge i \leq 0$

$\therefore K_2(DO, B) = x \leq 2 \wedge i \leq 2 \vee x=3 \wedge i \leq 1 \vee x=4 \wedge i \leq 1 \vee x=5 \wedge i=0 \vee x=6 \wedge i=0$

$K_3(DO, B) = twp(IF, K_2(DO, B)) \vee K_0(DO, B)$

$twp(IF, K_2(DO, B))$

$= (x>2 \vee x>3) \wedge (x>2 \supset twp(x=x-1; i=i+1, K2(DO, B)))$

$\wedge (x>3 \supset twp(x=x-2; i=i+1, K_2(DO, B)))$

$= (x>2 \vee x>3) \wedge (x>2 \supset x \leq 3 \wedge i \leq 1 \vee x=4 \wedge i \leq 0 \vee x=5 \wedge x \leq 0 \vee x=6 \wedge i=-1 \vee x=7 \wedge i=-1)$

$\wedge (x>3 \supset x \leq 4 \wedge i \leq 1 \vee x=5 \wedge i \leq 0 \vee x=6 \wedge x \leq 0 \vee x=7 \wedge i=-1 \vee x=8 \wedge i=-1)$

$\therefore K_3(DO, B) = x \leq 2 \wedge i \leq 2 \vee x=3 \wedge i \leq 1 \vee x=4 \vee i \leq 1 \vee x=5 \wedge i=0 \vee x=6 \wedge i=0$

$n \geq 2$のとき$K_n(DO, B) = K_2(DO, B)$となるので,

$twp(DO, B) = K_2(DO, B) = x \leq 2 \wedge i \leq 2 \vee x=3 \wedge i \leq 1 \vee x=4 \wedge i \leq 1 \vee x=5 \wedge i=0 \vee x \geq 6 \wedge i=0$

$= x \leq 2 \wedge i \leq 2 \vee 3 \leq x \leq 4 \wedge i \leq 1 \vee i=0$

演習

DO: do $z \neq x \rightarrow z=z+1;\ y=y*z$ od

B: $y=x!$

に対してtwp(DO, B)を計算せよ.

演習

DO: do $x>2 \to x=x-1 \ \square \ x>3 \to x=x-2$ od

B: $x=0$

に対してtwp(DO, B)を計算せよ.

最終レポート

1. aを配列，ρ，ρ'を状態としたとき，aへの代入a[i]=tに対する実行関係
   $Exec_M$(a[i]=t，ρ，ρ')が成り立つためには，ρに対してどのようなρ'であればよいか．

2. 以下のプログラムDOを考える．ただし変数は全て自然数値をとるものとする．

   DO: do x>2 → x=x-1 □ x>3 → x=x-2 od

また，Bをx=5とする．
このとき，ダイクストラの方法でtwp(DO, B)を求めよ．

締切　　8月18日(月) 9:00　通常と曜時間が異なるので注意．
提出先　　manaba