

1K-2

Knuth-Bendix完備化手続きにおける
等式選択の戦略について

株式会社 東芝 システム・ソフトウェア技術研究所
水島 由美子 ・ 松本 一教

1. はじめに

Knuth-Bendixの完備化手続きとは、与えられた等式集合を、これと同値な完備な項書換システムに変換する手続きである。この手続きは、等式集合から等式を一つずつ選択し書換規則に変換していくのであるが、等式の選択順序が効率上きわめて重要な問題となる。

[Sakai]には、”等式 $s = t$ について $\max \{s$ 中に現れる関数記号の数, t 中に現れる関数記号の数} が最小になるような等式を選択するのが効率的である”と記されている。

その理由としては第一に、関数記号の少ない項は一般的であり、書換規則としての”力が大きい”事。第二に、関数記号の少ない方がそれから作られる要対が少ない事。第三に、等式の選択が公平(いつまでも選択されずに残っている等式が存在しない)である事。が挙げられている。([Sakai])

ここでは実行結果をふまえて、どのような等式選択がより効率的であるかを検討する。

2. 実験

[表1]はQuintus prologを用いた実行結果である(アルゴリズムは[坂井]に挙げられていたものに若干の工夫を行った)。項の間の順序とし

ては(左から右への)辞書式部分項順序を用いた。等式選択の方法としては、等式集合から以下の(1)~(6)の6つの方法で、等式を選択する事を検討した。

等式選択

以下のものが最小となる等式 $l = r$ を選択。

- (1). $r = l$ 中に現れる関数記号の数
- (2). $\max \{r$ 中に現れる関数記号の数, l 中に現れる関数記号の数}
- (3). r と l の順序の大きい方を s とする。
 s 中に現れる関数記号の数
- (4). $r = l$ の記号列としての長さ
- (5). $\max \{r$ の記号列としての長さ, l の記号列としての長さ}
- (6). r と l の順序の大きい方を s とする。
 s の記号列としての長さ

等式の選択は上記の他にも存在するが、完備化アルゴリズムの正当性([Huet])という面から、選択は公平でなければならない。ここで、与えられた数よりも少ない関数記号を用いて構成される項は有限個しか存在しないので、上記の(1)~(6)の選択はいずれも公平である。

	group			group2			groupoid			
	step	crit pair	rule	step	crit pair	rule	step	crit pair	rule	ordering fail
(1)	96	166	18	95	161	19	71	135	23	1
(2)	61	90	14	64	103	16	80	175	25	2
(3)	59	90	14	64	103	16	79	163	24	1
(4)	96	129	19	83	112	17	108	214	28	0
(5)	68	89	15	75	111	17	94	152	24	3
(6)	66	89	15	69	96	16	132	229	30	9

[表1]

On the experimental result of equation selection strategy of the Knuth-Bendix algorithm

Yumiko Mizushima ・ Kazunori Matsumoto
TOSHIBA Corp. Systems&Software Engineering Lab.

例題としては、よく知られている以下の3つのものを用いた。

例題

$E_{group} =$ $\{0+X=X, (-X)+X=0, (X+Y)+Z=X+(Y+Z)\}$ $E_{group2} =$ $\{X+0=X, X+(-X)=0, (X+Y)+Z=X+(Y+Z)\}$ $E_{groupoid} =$ $\{(X+Y)+(Y+Z)=Y, (X+(X+X))+Y=X+Y,$ $X+(X+Y)=r(X), (Z+X)+Y=l(X,Y)\}$
--

3. 実験結果

[表1]は実験の結果を表している。stepは等式選択の回数を、ruleは完備化手続きを通して作成された規則の総数を、crit pairは作成された要対の数を表している。またordering failは選択した等式が向き付けに失敗した回数を表す。(本システムは向き付けに失敗した等式を等式集合に返し、新たに別の等式を選択するようにしてある)

4. 考察

完備化手続きが効率的であるとはどういう事であろうか。実行時間が短い、生成される要対の数が少ない事も挙げられるが、ここではまず、必要な規則がいかに早く獲得されるか(つまり、余分な書換規則がいかに少ないか)に注目した。

E_{group} , E_{group2} は10の、 $E_{groupoid}$ は7の書換規則を生成し完備化手続きを終了する。

[表2]はそれぞれの場合の余分な(生成された既約でない)書換規則の数を表している。

各例によってばらつきはあるが、合計を見ると(4),(5),(6)よりも(1),(2),(3)の方が効率的である。これは記号列としての長さより、項を構成する関数記号の数で項の複雑さを定義する方が妥当である事を表している。また、(1)よりも(2),(3)の方が(4)よりも(5),(6)の方が効率的であり、等式を全体として捉えるより、等式が書換規則になった時の事を考慮し選択を行う方が有効である事がわかる。この事を考慮すると、理論的には(2)よりも(3)の方が効率的なはずであるが、実験結果は両者ともほぼ同じである。これは $s \Rightarrow t$ という書換規則において、多くの場合 t 中の関数記号の数よりも s 中の関数記号の数の方が多いためである。

	E_{group}	E_{group2}	$E_{groupoid}$	計
(1)	8	9	16	33
(2)	4	6	18	28
(3)	4	6	17	27
(4)	9	7	21	37
(5)	5	7	17	29
(6)	5	6	23	34

[表2]

では、実行時間についてはどうであろうか。以下はC言語を用いた E_{group} 等式選択(2),(3)の実行時間を測定したものである。

(2).		
TOTAL CPU TIME	:	7
CPU TIME FOR UNIFICATION(SUCCESS)	:	0.03
CPU TIME FOR UNIFICATION(FAILURE)	:	1.43
CPU TIME FOR REWRITING	:	4.62
CPU TIME FOR ORDERING	:	0.00
CPU TIME FOR SELECTING EQUATION	:	0.07
(3).		
TOTAL CPU TIME	:	9
CPU TIME FOR UNIFICATION(SUCCESS)	:	0.17
CPU TIME FOR UNIFICATION(FAILURE)	:	1.38
CPU TIME FOR REWRITING	:	4.72
CPU TIME FOR ORDERING	:	0.03
CPU TIME FOR SELECTING EQUATION	:	1.37

(2)では等式選択にかかる時間はほとんど無視することができるが、(3)では実行時間の15%にも及んでいる。

以上の実験により、作成される規則の面からも、実行速度からも(2)のように選択するのが良い事が結論づけられる。

参考文献

- [Sakai] Sakai, K: An ordering method for term rewriting systems, ICOT TR-062, 1984.
 [坂井] 坂井 公: Knuth-Bendixの完備化手続きとその応用, コンピュータソフトウェア, vol. 4, No. 1, pp. 2-22, 1987.
 [Huet] Huet, G.: A complete proof of correctness of the Knuth-Bendix completion algorithm, J. Comput. Syst., Vol. 23, No. 1, pp. 11-21, 1981.