3. 数学の証明の計算機による実施のためのプログラミング

高 須 達（電気通信研究所）

§ 数学的方法

定理と，それを証明するに充分なだけの定義を与えて，一定の方針に従つて証明を構成するという名古屋大学黒田教授の方法による。

定義 $\forall w . w \varepsilon \mid a \equiv \exists xy . w = \langle xy \rangle \wedge w \varepsilon \sigma \wedge x \varepsilon a$

定理 $[\forall x . x \varepsilon a \to x \varepsilon b] \to [\forall w . w \varepsilon \sigma \mid a \to w \varepsilon \sigma \mid b]$

証明

$[\forall x . x \varepsilon a \to x \varepsilon b] \to [\forall w . w \varepsilon \sigma \mid a \to w \varepsilon \sigma \mid b]$

――――――――――――――――――――

(1) $\neg \forall x . \ x \varepsilon a \to x \varepsilon b$

$- \forall w . w \varepsilon \sigma \mid a \to w \varepsilon \sigma \mid b$

――――――――――――――――――――

$- w \varepsilon \sigma \mid a \to w \varepsilon \sigma \mid b$

――――――――――――――――――――

$- w \not\varepsilon \sigma \mid a$

$- w \varepsilon \sigma \mid b$

――――――――――――――――――――

$- \neg \exists xy . w = \langle xy \rangle \wedge w \varepsilon \sigma \wedge x \varepsilon a$

(2) $\exists xy . w = \langle xy \rangle \wedge w \varepsilon \sigma \wedge x \varepsilon b$

――――――――――――――――――――

$- \neg \exists y . w = \langle ry \rangle \wedge w \varepsilon \sigma \wedge r \varepsilon a$

――――――――――――――――――――

$- \neg w = \langle rs \rangle \wedge w \varepsilon \sigma \wedge r \varepsilon a$

――――――――――――――――――――

(3) $w \neq \langle rs \rangle$

$- \neg w \varepsilon \sigma \wedge r \varepsilon a$

――――――――――――――――――――

(4) $w \not\varepsilon \sigma$

(5) $r \not\varepsilon a$

(1) ――――――――――――――――――――

$- \neg r \varepsilon a \to r \varepsilon b$

――――――――――――――――――――

$r \varepsilon a$ (5) (6) $r \not\varepsilon b$ (2) ――――

$- \exists y . w = \langle ry \rangle \wedge w \varepsilon \sigma \wedge r \varepsilon b$

――――――――――――――――――――

$- w = \langle rs \rangle \wedge w \varepsilon \sigma \wedge r \varepsilon b$

――――――――――――――――――――

$$\frac{w=\langle r\,s\rangle}{(3)} \qquad \frac{-w\,\varepsilon\,\sigma\wedge r\,\varepsilon\,b}{w\,\varepsilon\,\sigma \quad r\,\varepsilon\,b}$$
$$(4) \qquad (6)$$

論理記号と証明の方針

〔1〕 ∨：（左）∨（右）の時（左），（右）を縦に直列的にならべる。

¬∧：¬〔（左）∧（右）〕の時両側の否定をとつて，直列的にならべる。

→：（左）→（右）の時，（左）の否定と（右）を直列的にならべる。

$\forall$：$\forall x$の時xの代りに自分の糸の中でそれまでに使わなかつた文字を入れる。

¬∃：$\neg\exists x$の時，xの代りに自分の糸の中で，それまでに使われなかつた文字を入れ，否定をとる。

〔II〕¬∨：〔（左）∨（右）〕の時，両側の否定をとつて横に並列的にならべる。

¬∧：（左）∧（右）の時，（左），（右）を横に並列的に並べる。

↛：（左）↛（右）の時，（左）はそのまま，（右）は否定をとつて横に並列的に並べる。

¬$\forall$：$\neg\forall x$の時，xの代りに自分の糸の中でそれまでに使つた文字の適当なものを入れる。そして否定をとる。

∃：¬$\forall$と同じにして，否定はとらない。

〔III〕 ε，=，⊄，≠ 定義があれば，それを証明の中に導入し，無ければ，primitive として，cancelする対を作るかどうかを調べる。

証明の任意性

証明を構成するための分解の順序は，グループ〔III〕に属する定義型のもの，グループ〔I〕に属する直列型のもの，グループ〔II〕に属する並列型のもの，の順に分解する必要がある。

この際に，

(1) $\exists x$，$\neg \forall x$，の x の代りに入れるべき適当な文字の選択を誤れば，証明は失敗する。

(2) 枝別れを生ずるような論理記号がいくつか，使い残っている場合に，次にどれをとつて分解するかは，任意であると共に，悪い場合は証明は失敗する。

証明が失敗するというのは，相殺を生じないで，証明が無限に長くなることを意味する。（$\exists$，$\neg \forall$ は同じものを何回分解してもよいから）

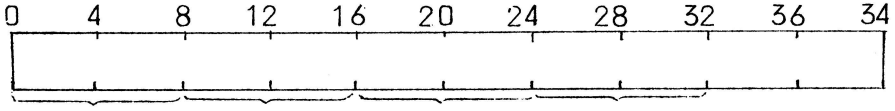
従つて，計算機に証明を実施させる場合には，証明の長さを制限して，その制限の長さを超えれば，上に注意した(1)，(2)についてそれぞれヤリナオシをする必要がある。

なお，証明の長さの指定は，証明すべき定理に対して，あまり大きくすると，時間を非常に浪費する可能性がある。特に定理が成立しない場合に，一定の長さの範囲で証明が存在しないことを結論するためには，この事が大きな障害となるであろう。

§ プログラミングの方針

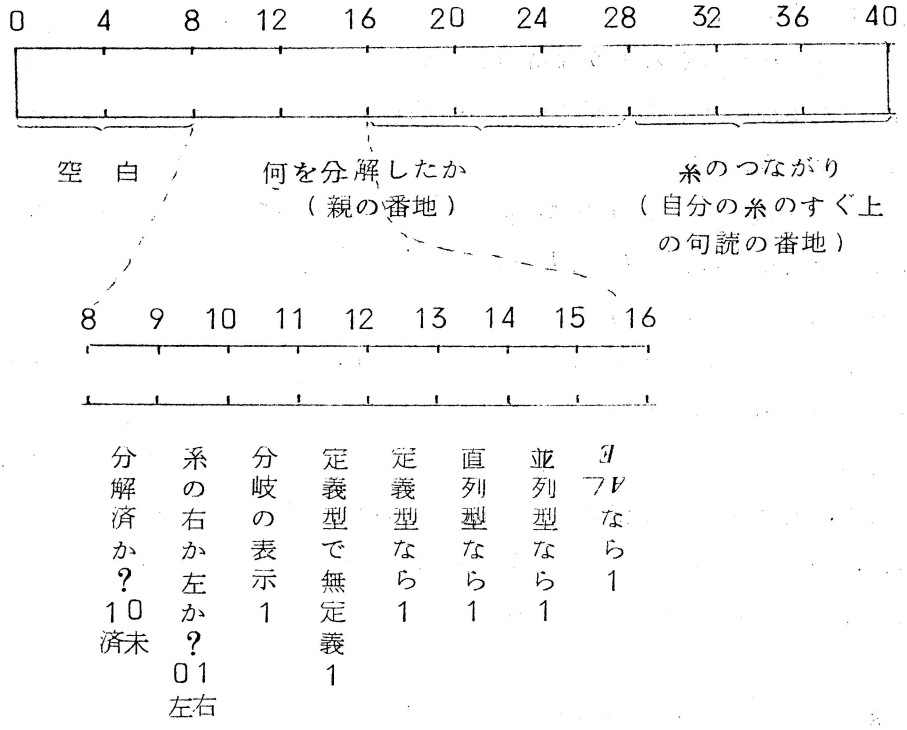
(1) Formula の表現の仕方

1 long word 40 bit



記号	左	右	自
ε	x	a	1
ε	x	b	2
$\rightarrow$	1	2	3

⑵ 句 読 点



1	0	0	0	定義型
0	1	0	0	直列型
0	0	1	0	並列型
0	0	1	1	ヨコア，アタ

⑶ メモリーの使い方

0～9 一時記憶

10. 証明を次に書きこむべき番地

11. 次に分解すべきformulaの番地

12. 系の長さのcurrent value

13. 今考えているのは系の左か右か？

14. 今まで何回並列型が登場したか

15. 何回分解したか。

16. ∃ , ∀ が何回登場したか

17. 11.の予備

18. 系の長さの制限値

19. 定義のリストの一番下の番地

20~27. 文字記憶

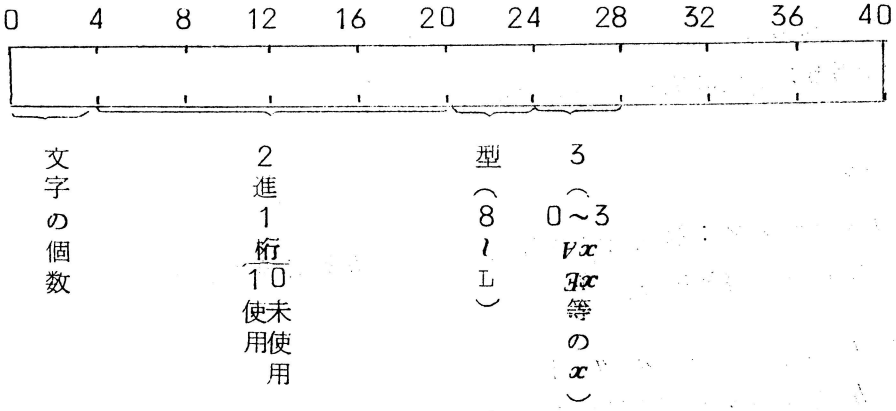
type Ⅰ の文字 128~143(16進80~8L)

〃 Ⅱ 〃 144~159(〃 90~9L)

〃 Ⅲ 〃 160~175(〃 KO~KL)

....

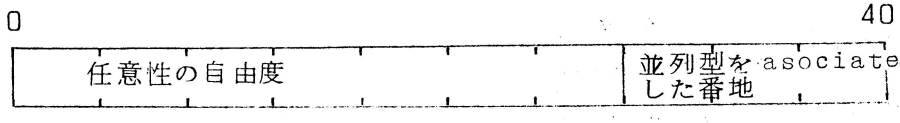
〃 Ⅷ 〃 240~255(〃 LO~LL)

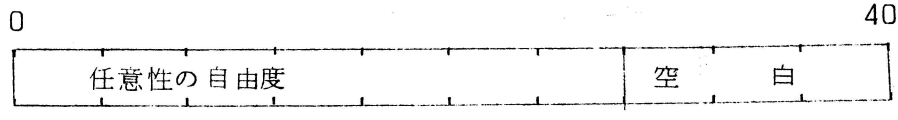


28~47. テープ読込と証明の長さのチエック

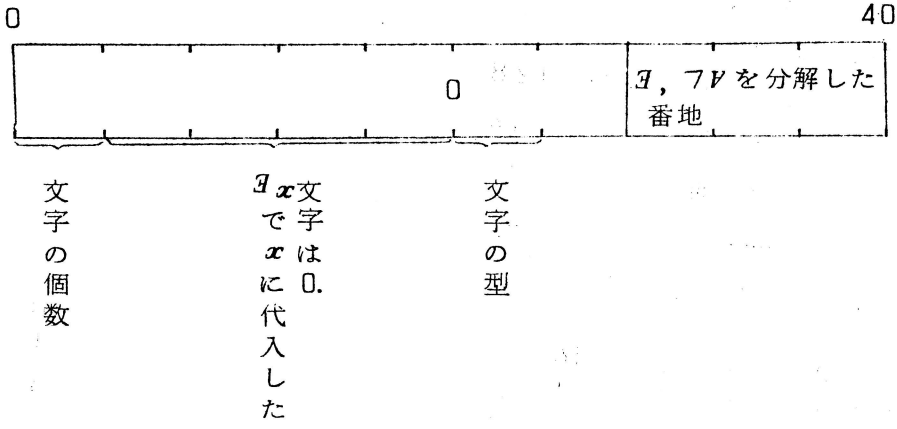
48~99 サブルーチン

100~104 並列記憶





105~110 $\exists, \forall$ 記憶



111~121　定義

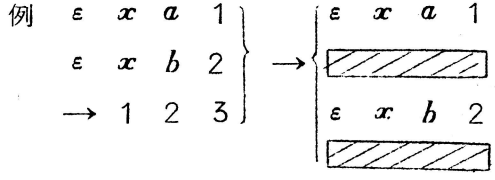
122~134　定理

135~255　証明

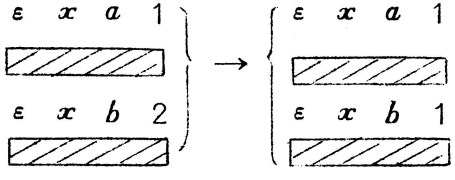
注意　100~255：定理に応じて variable

(4)　サブルーチン

(I)　〔1〕左右分離ルーチン：論理記号の右側の系統の語と左側の系統の語を分離して証明に書き込む。



〔2〕番号附け変え



〔3〕 $\rightarrow$, $\neg\wedge$, $\vee$: $\nrightarrow$, $\wedge$, $\neg\wedge$ に応じて否定をとるか又はとらない。

(II)〔4〕 句読点作製

(III)〔5〕 何を次に分解するかを判断するルーチン

句読点を自分の系でたどつて，定義型，直列型，並列型の先行順序で未使用のものを分解する。

(IV)〔6〕 $\exists$, $\neg\forall$ 証明への書き込み

〔7〕 文字調整

〔8〕 Substitution

(V)〔6〕 $\exists$, $\neg\forall$ 証明への書き込み

〔9〕 文字調整

〔8〕 Substitution

(VI)〔10〕 cancel する対の検出

〔11〕 系の右側を探して，証明の完了か，又は右側へ移つて，分解を続行する。

(VII)〔12〕 定義導入

〔13〕 定義用Substitution

(VIII)〔14〕 $\exists$, $\neg\forall$ についてのヤリナオシ

〔15〕 並列型の分解順序についてのヤリナオシ

§ 結　び

今後の問題

(1) 公理系から出発して，自然発生的に定理を計算機に作らせる方針についても，ある程度発展させて，今考えているような方法と結合させること。

(2) 与えられた定理に対して，それを証明するに充分な定義を計算機に作らせることは出来ないであろうか？

最後に，御指導を頂いた名古屋大学　黒田先生，通研　喜安次長ならびに　MUSASHINO　I　の保守にあたられた通研電子応用研究室計算機グループの各位に，厚く謝意を表させて頂きます。

本 PDF ファイルは 1960 年発行の「第 1 回プログラミング–シンポジウム報告集」をスキャンし、項目ごとに整理して、情報処理学会電子図書館「情報学広場」に掲載するものです。

この出版物は情報処理学会への著作権譲渡がなされていませんが、情報処理学会公式 Web サイトの https://www.ipsj.or.jp/topics/Past_reports.html に下記「過去のプログラミング・シンポジウム報告集の利用許諾について」を掲載して、権利者の捜索をおこないました。そのうえで同意をいただいたもの、お申し出のなかったものを掲載しています。

過去のプログラミング・シンポジウム報告集の利用許諾について

情報処理学会発行の出版物著作権は平成 12 年から情報処理学会著作権規程に従い、学会に帰属することになっています。

プログラミング・シンポジウムの報告集は、情報処理学会と設立の事情が異なるため、この改訂がシンポジウム内部で徹底しておらず、情報処理学会の他の出版物が情報学広場 (=情報処理学会電子図書館) で公開されているにも拘らず、古い報告集には公開されていないものが少からずありました。

プログラミング・シンポジウムは昭和 59 年に情報処理学会の一部門になりましたが、それ以前の報告集も含め、この度学会の他の出版物と同様の扱いにしたいと考えます。過去のすべての報告集の論文について、著作権者（論文を執筆された故人の相続人）を探し出して利用許諾に関する同意を頂くことは困難ですので、一定期間の権利者捜索の努力をしたうえで、著作権者が見つからない場合も論文を情報学広場に掲載させていただきたいと思います。その後、著作権者が発見され、情報学広場への掲載の継続に同意が得られなかった場合には、当該論文については、掲載を停止致します。

この措置にご意見のある方は、プログラミング・シンポジウムの辻尚史運営委員長 (tsuji@math.s.chiba-u.ac.jp) までお申し出ください。

加えて、著作権者について情報をお持ちの方は事務局まで情報をお寄せくださいますようお願い申し上げます。

期間：2020 年 12 月 18 日 ～ 2021 年 3 月 19 日
掲載日：2020 年 12 月 18 日

プログラミング・シンポジウム委員会

情報処理学会著作権規程
https://www.ipsj.or.jp/copyright/ronbun/copyright.html