

WORD

From College of Information Science

学類誌今昔物語

Bash わかるよ その気持ち

Coq の基礎の基礎

博士号回収記録

オンラインでソフトウェイイベントを開くときのテクニック集

プリティーシリーズ視聴RTA（プリパラ～プリ☆チャン，映画分岐全制覇，敗北EDカット）

29d22h36m08s

クソ長いテキストから超高速かつ超省メモリに部分文字列の出現回数を数えたい



WORD は大学新聞の
学類誌特集から
ハブられました号

2022.06

51

目次

学類誌今昔物語	Azumabashi	1
Bash わかるよ その気持ち	coord_e	34
Coq の基礎の基礎	public_yusuke	46
博士号回収記録	cely chan/tactilium	59
オンラインでソフトウェアイベントを開くときのテクニック集	puripuri2100	63
プリティーシリーズ視聴 RTA (プリパラ～プリ☆チャン, 映画分岐全制覇, 敗北 ED カット) 29d22h36m08s	n.takana	67
クソ長いテキストから超高速かつ超省メモリに部分文字列の出現回数を数えたい..	塚本	72
WORD 編集部へのお誘い	public_yusuke	81
編集後記		82

学類誌今昔物語

文 編集部 Azumabashi

1 WORD 編集部に激震走る

2022 年 4 月 6 日、WORD 編集部に激震が走りました。前日に発行された筑波大学新聞第 369 号^{*1}で学内誌の特集が組まれたのは良いが、我らが WORD がないではないか！しかも 2016 年の学類誌特集^{*2}には WORD があったのに！

wordian は激怒した。必ず、かの邪智暴虐^{*3}の大学新聞を除かなければならぬと決意した。wordian には学内事情がわからぬ。wordian は、WORD 編集部の住人である。学類誌の原稿を書き、授業を受けて暮して來た^{*4}。けれども邪惡に対しては、人一倍に敏感であった。（以下略）

そこで本稿では、過去に発行されていた学類誌のうち、現在確認できる範囲のものを概観しようと思います。

2 学類誌の基準

本稿での学類誌の定義は、次の定義を採用します。

学類誌とは、「各学類（専門学群・総合学域群を含む。以下同様）の学生が主体となって発行していて、かつその学類の名のもとに発刊されている（ないし発刊された）雑誌」をいう。ただし、新歓パンフレットの類は除く。

すなわち、各サークルの広報誌や、各学類のパンフレットなどは学類誌に含みません。あくまでも WORD のような雑誌のみを指すことにします。

3 情報の出所について

情報の出所は、以下のいずれかです。

1. (学類誌の現物を確認できていれば) 学類誌の現物
2. 学類誌協議会編『大学ノート』(1987 年 10 月 9 日発行分および 1988 年 10 月 8 日発行分)。表記の簡単のため、それぞれ『87 年ノート』および『88 年ノート』と略記することにします。
3. 「筑波大学新聞」第 59 号 (1982 年 2 月 23 日発行), 第 65 号 (1982 年 12 月 22 日発

*1 <https://www.tsukuba.ac.jp/about/public-newspaper/pdf/369.pdf>

*2 <https://www.tsukuba.ac.jp/about/public-newspaper/330.pdf>

*3 針小棒大にも程があるが、このままにしておきます。

*4 単位を取れたとは言っていない。



図1 『88年ノート』の表紙.

行), 第330号(2016年10月3日発行)*5. 筑波大学新聞の第n号が出典の情報は,簡単のために, 情報の出所は「大学新聞n号」と書くことにします。

これら以外の媒体を出所とする情報は, その都度情報の出典を示します。

とくにこれらの文献の中でも, 『大学ノート』に負うところが大きいです。『大学ノート』とは何かについては, 『88年ノート』の「発刊の辞」の説明がわかりやすいので, これを引用します。

皆さん、こんにちわ。(引用者注: 原文ママ) 1年ぶりの大学ノートです。社会学類誌『そおしあーる』(引用者注: 原文ママ) の提言で創られ(引用者注: 原文ママ) 筑波大学学類誌協議会も今年で3年目を迎えました。学類誌協議会は各学類の学類誌編集委員が意見、情報を交換し合いながら、各自の誌面を充実させる目的で毎月定例会を開いています。この大学ノートはその公開の場であります。

『大学ノート』を経由することで、現物の残っていない学類誌の情報や、かつての学類誌の規模の情報を得ることができました。昔の学類誌の雰囲気を味わいたい読者は、まずは『大学ノート』に当たってみることをおすすめしたいと思います。

*5 <https://www.tsukuba.ac.jp/about/public-newspaper/330.pdf>



図2 『社会学類誌』「じゅんびごう」の表紙.

4 色々な学類誌

ここでは、色々な学類誌を紹介します。確認できた中で^{*6} 最新の号の発行日が古い順番に紹介していきます。また、日付はすべて西暦に改めています。詳細な分析は史学が専門の人にお任せするとして^{*7}、ここでは大雑把な記述に留めます。また、それぞれの学類誌がどのような方針のもと創刊されたを浮き彫りにするため、入手可能なものについては発刊の辞ないしこれに類するものを掲載しました。

ここでは、個別の学類誌に対する記述に留めます。学類誌全体の傾向は、後で掘んでみます。

4.1 社会学類誌

はっきり言って謎の多いのがこの『社会学類誌』です。発行年不明、後続の『そおしある』との関連性不明とわからないこと尽くし。しかも「創刊号」が第1号かと思ったら「じゅんびごう」が出ています。この「じゅんびごう」は「11月10日（月）」発行と示されています。試しに、筑波大学開学の年である1973年以降の、11月10日が月曜日である年を列挙すると、1975, 1980, 1986, 1997, 2003, 2008, 2014となります。『そおしある』と同時に刊行されていたことはまずないだろうと思われる所以、1975年または1980年の創刊でしょうか。仮にこの『社会学類誌』が改組されて『そおしある』になったと仮定する

^{*6}当然未確認の号があるかもしれません。

^{*7}そういうのが得意そうという偏見があります。

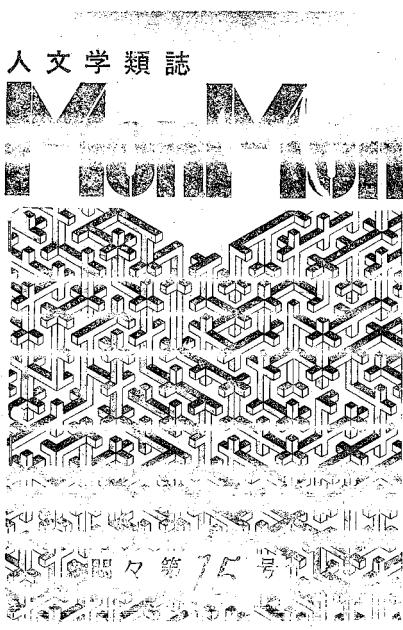


図3 『MonMon』第15号の表紙。

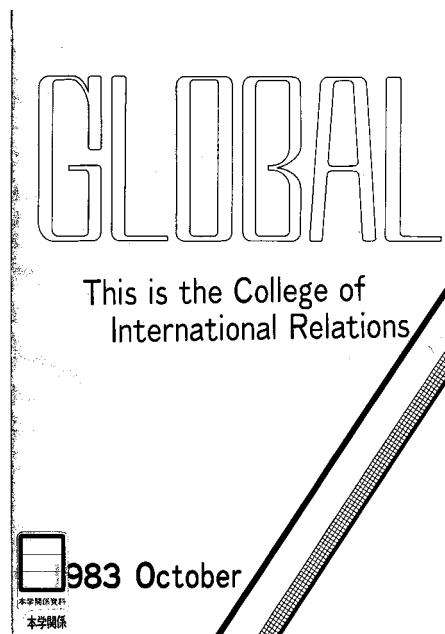


図4 『GLOBAL』第1号の表紙。

と、1980年発行説が有力になります^{*8}。いずれにせよ、かなり古いもので、学類誌としては初期のものと思われます。

確認できる最新刊の創刊2号は「2月16日(月)」発行とされています。こちらも発行年が確認できませんし、発行頻度が不明ですので「じゅんびごう」の日付から割り出すこともできません。

4.2 人文学類誌『MonMon』

人文学類誌『MonMon』は、1980年12月22日に第3号が発行された後、1982年10月4日に第15号が発行されました。人文日報(4.5節)の記述により、1983年11月に休刊したことがわかります。表紙に「悶々」ないし「悶悶」の文字が見えることから、卒論やレポートなどで苦しむ様から命名されたのでしょうか。人文学類誌編集委員会の手により編集され、人文学類クラス代表者会議の手により発行されたようです。

4.3 國際關係學類誌？『GLOBAL』

1983年10月に第1号が発刊されたようですが、肝心の第1号しか残っていません。発行主体も謎ですが、「刊行にあたって」では

さて、わが國際關係學類では、劇、喫茶、雑誌の三本立てで学園祭に臨むことになり、本誌の編集はその一環として行われたものである。

^{*8}1975年発行の場合、『そおしあ～る』の創刊と間が空きすぎと思われます。



図5 『人文日報』(号数不明)の表紙.

と書かれていることから、そんなに長続きさせる意図はなかったのではないかと思われます。このような刊行背景があったことから学類誌に分類するのには躊躇がありますが、表紙にデカデカと「This is the College of International Relations」と書かれていることから学類誌に分類してしまうことにします。

4.4 比較文化学類誌『COM』

比較文化学類誌『COM』は、存在はしたようですが、『87年ノート』および後継の比較文化学類誌『ize』の本文中にしか登場しない、謎の学類誌です。しかも、『87年ノート』はそれぞれの学類誌が記事を持ち寄ってひとつの冊子を構成していましたが、『COM』だけは『COM』が寄贈した記事が見あたらないことが、さらに『COM』を謎な存在にしています。発刊時期は明らかではありませんが、『ize』の刊行（1989年）当時には既に終刊されていたようです（詳細は『ize』の項目で示します）。

1987年当時は、年3~4回、300~400部程度の学類誌を発行していたようです。この規模から考えるに、現代の『WORD』と大差ない規模の学類誌であったことが窺えます。

4.5 人文学類誌『人文日報』

人文学類誌『人文日報』は、1987年5月に創刊されました。その後第8巻までの刊行が確認されていますが、あいにく刊行日の記載が確認できませんでした。先述した『MonMon』の発行とは、恐らく時間的な違いがあります（こちらのほうが遅い時期に発行された）。1987年5月に刊行された学類誌には「創刊」と書かれていたので、おそらく『MonMon』廃止後に改めて創刊されたものと思われます。発行主体は「人文公社」と書かれています。

なお、紙媒体で発行された号は確認できていませんが、『87年ノート』および『88年ノート』には人文日報の名が見えます。この時期は、ほぼ季刊で300部程度発行していたようです。

残念ながら創刊号は残存していないようですが、『87年ノート』に「じんぶんにっぽうのれきし」と題された文章が掲載されていたので、引用しておきます。

1986年4月1日、現在第一線を（一応）退き最低顧問（引用者注：原文ママ）となっている前委員長・社長らの約3名がxxxxxx（引用者注：原文ママ）の進出により、xx学類誌（引用者注：原文ママ）の横行する人文学類の現状を憂い、1983年11月以来休刊していた『MONMON』の復刊を断行、人文学類誌今ここに蘇らんとの気炎を上げた。この錦の御旗の下に続々と結集した憂学の志士たちは、先達の偉大なる栄光を再興せんと新学類誌名を公募し、現社員であるSYATO明俊の投稿『人文日報』を公社中央創刊準備計画会議席上で満場一致の快挙をもって可決した。しかし、数年間の空白はいかんともしがたく、新入社員の怠慢、人文学類誌復刊を恐れる某学類誌側工作員の謀略などにより、復刊を目指して奮戦努力する我々の前途は決して明るいものではなかった。このような諸事情により復刊への道程は困難を極めたが、我々人文公社社員の徹夜の連続により、86年7月1日付をもってかろうじて復刊準備号が発行された。しかしこの後の経過も、社員の脱落及び夏休みボケ等によりおもわしいものではなく、日々はいたずらに過ぎて行ったが、我々社員は新生への陣痛ともいべき辛酸の日々に耐えたのである。かくして86年10月1日、我々の努力は創刊号に結実した。僅か148部という部数ながら創刊号はその斬新な企画内容、過激なスクープ記事等がパクスソーシアーナに浸り切っていた業界関係者及び学類生たちに甚大なる衝撃を与え心胆を寒からしめた。ここにパクスソーシアーナは終えんを迎えるが、これはにわかに顕在化した学類誌活動の拡大によって泥沼の学類誌闘争と化し、業界は弱肉強食の戦国時代を迎えたのである。業界の混迷は大学ノオトの発行により一応の妥協をみた形となったが、全学を制覇せんとする人文日報の意気込みは、日報新世紀の到来を告げる時の声であろう。

いちいち話のスケールがやたら壮大ですが、要は「xx学類誌」である「xxxxxx」に対抗して復刊した、ということでしょう。ところでこの「xxxxxx」ですが、おそらく社会学類誌の『そおしあ～る』(4.16節参照、1982年創刊)のことと思われます。その理由は、ちょうど文字数が一致するだけでなく、「パクスソーシアーナ」という単語にあります。調べてみると、「パクス-ローマーナ(ラテン語で Pax Romana)」という言葉があるようです。辞典を紐解くと、

(「ローマの平和」の意) ローマ帝国初期のアウグストゥス時代から五賢帝時代末期までの約二〇〇年間(前二七-後一八〇)。動乱や戦争は収まり、文化的発展のめざましい時代だったのでいう。

と説明されています^{*9}。これをもじったフレーズが「パクスソーシアーナ」であり、「ソーシア」が social のことだと考えると辻褄が合います。我らが WORD もこの時期には既に発刊していたはずですが、当時の「ナンバー学群^{*10}」の垣根を超えることはできなかった模様です。

なお、これは完全に余談ですが、パクス-ローマーナがラテン語由来の言葉であることを考えると、「パクスソーシアーナ」は音だけしか見ていない（ラテン語の文法をガン無視している）ことがわかります。どこのご家庭にもあるラテン語の辞書^{*11}で「社会」を引いてみると、「humana societas [consorito], hominum communitas [consociato], civilis [civium] societas, consociati homines」と出ており、「社会」をいうためには2語以上が必要であることがわかります。「社会の」としても事情は変わらず、「ad hominum societatem pertinens」という訳語が掲載されています。英語の social に似ているラテン語の societas を引いてみると、

- 1 仲間関係、共同、提携。
- 2 団体、結社、組合
- 3 同盟。
- 4 密接な関係。
- 5 〔か〕修道会。

と登録されており、現代でいうところの「社会」の意味からは逸脱しています。しかも Romana は第1類形容詞 Romanus の女性名詞形活用形^{*12}の主格である一方、societas は名詞ですし、形容詞形 socius は第1類形容詞のため同じ活用をしますが、「社会」の意味からは離れてしまいます^{*13}。もっとも、パクス-アメリカーナという言葉もあるらしいので、発音しか見なくてもよいのかもしれませんが……。

現在の人文学類は学類誌を発行していないため、今までのどこかで休刊ないし廃刊されたようです。

*9 広辞苑第七版による。

*10かつての筑波大学は、第一学群・第二学群・第三学群と若干の専門学群から構成されていました。この時代の第一学群・第二学群・第三学群を俗に「ナンバー学群」ということがあります。人文学類は第一学群、情報学類（当時）は第三学群の所属でした。

*11 和羅辞典改訂版（研究社）による。

*12なぜなら pax が女性名詞なので。

*13 この文章の筆者（Azumabashi）は初級ラテン語の初步の初步だけかじったことがあります、このあたりの事情はほんの僅かにわかります。もっとももうほとんど忘れてしましたが……。

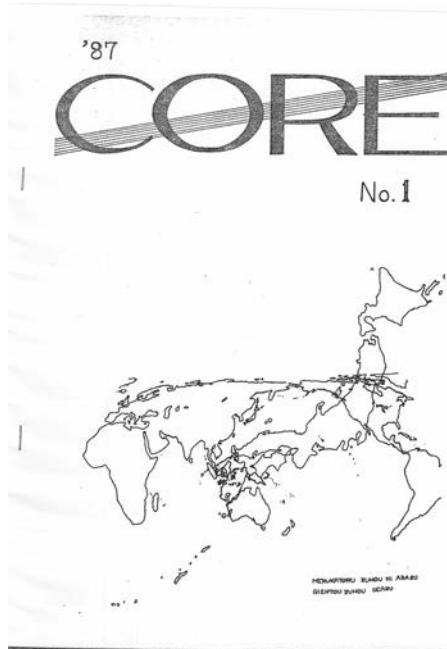


図 6 『CORE』1987 年 No. 1 の表紙.

4.6 基礎工学類誌『CORE』

1986 年 12 月 18 日付で昭和 61 年の No.4^{*14} が発刊され、1987 年 2 月 19 日付で昭和 62 年の No.1 まで確認できました。『CORE』は、年度ごとに巻数がリセットされる方式で番号がつけられていたため、通算何号なのかは不明です。『87 年ノート』によれば創刊は 1979 年 10 月とのことです。発行主体は「基礎工学類学類誌編集委員会」です。

基礎工学類は、工学基礎学類（1998 年）を経て現在は応用理工学類（2007 年）になっています^{*15}が、これらの学類に引き継がれたかは不詳です。『88 年ノート』には『CORE』が登場すること、および現在の応用理工学類は学類誌を発行していないことから、1988 年以降のどこかのタイミングで休刊ないし終刊した模様です。

『87 年ノート』によれば、「CORE」の名称の由来は、

なんでも Engineering の CORE (核心) に近づきたい、ということから命名されたそうだ。

ということです。思ったより命名の由来に基礎工学類らしさがなかった。



図 7 人間学類誌『さやにんげん』のタイトルロゴ（『88年ノート』より抜粋）。かわいい。

4.7 人間学類誌『さやにんげん』

人間学類誌『さやにんげん』は、存在はしたようですが、『87年ノート』および『88年ノート』にしか登場しない、謎の学類誌です。『88年ノート』によれば、「今年で創刊4年目」ということなので、1984年創刊と思われます。1988年当時は、年5回、250部を発行していたようです。名前の由来は言うまでもなく「さやいんげん」と人間学類の「人間」のダジャレと思われます。

現在の人間学群各学類は、学類誌を発行していないので、1988年以降に廃刊したものと思われます。

4.8 生物学類誌『HIPPOPOTAMUS』

生物学類誌『HIPPOPOTAMUS』は、大学新聞330号によれば、1978年創刊のようです。『88年ノート』によれば、学期に1-2回、400部を配布していたようです。

タイトルの「HIPPOPOTAMUS」については補足が必要でしょう。『87年ノート』によれば「ひぼぼたます」と読むようです。『87年ノート』に詳しい解説があるので、引用します。

この Hippopotamus：生物学辞典を紐といてみると・・・・

脊椎動物門	IX 紹	Hammalia	(ほにゅうるい)
IX-3 亜綱		Theria	(じゅう るい)
下綱		Eutheria	(真じゅうるい)
目 26		Artiodactyla	(ぐうているい)
亜目 I		Suiformes	(イノシシるい)

に属する単なるカバです。

因にこのカバ：ヒスワリ語（引用者注：原文ママ）では『カリ』と呼ばれ気分に邑のある危険な動物です。群れているときは大人しく、一匹になると危険な動物なんで

*14奥付にはNo.3とありますが、表紙および訂正紙にはNo.4があるため、ここではNo.4という表記に従います。

*15学類改組の年度は <https://sse.tsukuba.ac.jp/outline/history/> をもとにしました。

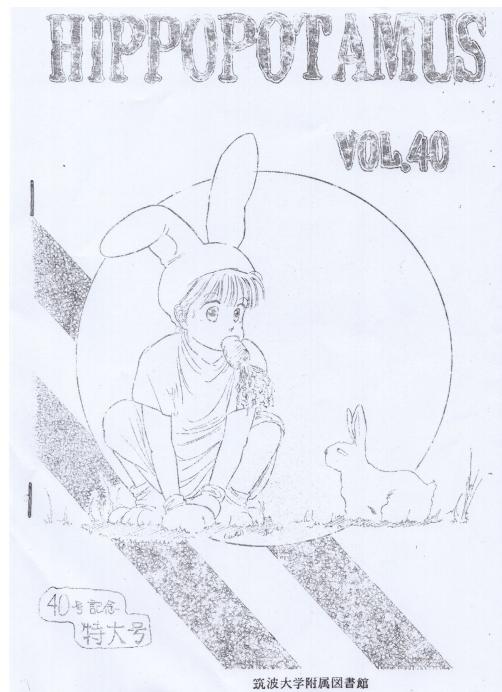


図 8 生物学類誌『HIPPOPOTAMUS』vol.40 の表紙. 『87年ノート』ではカバの絵があしらわれていたが、この号はなぜかウサギである。

す。（わかる人には、作者の愛読書がわかる。）

表紙のカバの大口をあけた絵にはそーゆ意味があったんです。わかつてましたか？

このような記述があります。いかにも生物学類らしい題名および表紙のチョイスではないでしょうか。

この HIPPOPOTAMUS ですが、当初は少し揉め事があったようで、1979 年刊行の「第 5 号」でその事情が報告されています。少し引用しておきます（＊印は判別不能な箇所を示します）。

生物学類の編集活動許可さらに発行が「申し入れ」成立を通じて公的に許可され、ここに第一号を発行できることになりました。そこで、「なぜここまで発行が遅れたか」、言い訳がましいけれども少少（引用者注：原文ママ）のべたいと思います。

まず、5月11日付厚生補導審議会決定「学類専門学群誌（紙）発行に関する取扱いについて」が出され学類誌刊行に関しての要項が必要になってきました。そこで5月末日に発足した編集委員会では、一学期第一号発行を目指して要項作成を開始した由です。その時、前田先生（学生担当教官）から要項提示され、それに対して私達の方でも要項を作成し、編集側の案として提出したのです。ここまで発行が遅れた理由は、要項作成段階で学担側としては「要項決定は教員会議が行う」としていたのに対し、私達の方は独特的の案を認めてほしかった点。さらに学担側は「編集段階で指

導助言をしたい、また教育活動の一環としたい」としていたのに対し、私達の方では「校費使用を要望している点で学担の指導はしかたないにしても、それは編集の最終段階にしてほしい」というものでした。やはり学担の介入ができるだけ避け、学生の自主的な活動を認めてほしかったのです。さらに、学担側と編集委員会側の意図する学類誌の在り方に相違もありました。大まかに言うとこの3点となります。特に第一点が最大の問題で10月半ばまでもめていた由です。結局、要項は私達の要望をある程度受け入れた形で教員会議が決定するが、それに対し私達の要項は「申し入れ」という形をとってはどうか、と平林先＊＊＊類誌担当教官）から要請されたのです。10月末に私達は要項を受け入れ、さらに11月上旬クラス代表者会議も承認、学類誌刊行に関する申し＊＊＊学類長に認められ、その後、要項が教員会議で承認されたことにより、学類誌が正式に認められ、編集活動にGOサインが出た由です。

以上ですが、長いブランクの後の学類誌の発行に際して、学類長の＊木先生、さらに前田先生、平林先生、また座長である角本さんに「学類誌に望むもの」として、前の二人には寄稿を、後の二人にはインタビューをし、それをまとめたものを次に載せたいと思います。

1978年創刊で、1979年時点でのこのような「揉め事」によるブランクを「長い」と判断するか否かは見解の分かれるところかもしれません。また、「学類専門学群誌（紙）発行に関する取扱いについて」の出された経緯も判然としませんが、とにかくこのようなことがあったようです。それでは1978年の創刊第一号はどういう経緯で出せたのかが気になるところですが、残念ながら1988年の時点で既に散逸してしまったようです。

その後、1988年9月30日付でvol.40を迎えたようです。vol.40には、当時学内に存在した学類誌のうち『人文日報』『そおしある』『CREVIX』『WORD』からのお祝いの言葉（図9）が掲載されました。確認できた最新の号は「別冊89'正月号」ですが、印刷は1988年内に行っていたようですので、1989年に『HIPPOPOTAMUS』が活動していたかどうかを確認することは残念ながらできません。

現在の生物学類では学類誌を発行していないため、1988年以降に廃刊したようです。

4.9 医学専門学群誌『CERVIX』

医学専門学群誌『CERVIX』は、『88年ノート』でのみ確認できる学類誌です。1987年12月25日に創刊されたようです。『88年ノート』に掲載された紹介文を引用しておきます。

CREVIXは、去年の12月25日、クリスマスの日に創刊したばかりの、まだまだ赤ん坊の学群誌です。生まれたては、20代前半のお兄ちゃん2人が苦労してくれて、なんとか息ができたのですが、ひどく難産で、お兄ちゃん達に言わせれば、周りも心配する程の未熟児だったそうです。

情報を提供し、物事を考えるいとぐちを与えてくれるようなコになって欲しい……という願いのもと、栄養と愛情を注いでくれる人が増え、なんとか2ヶ月3ヶ月と体重も増していました。6ヶ月後には、「生まれた時にはどうなることかと思った

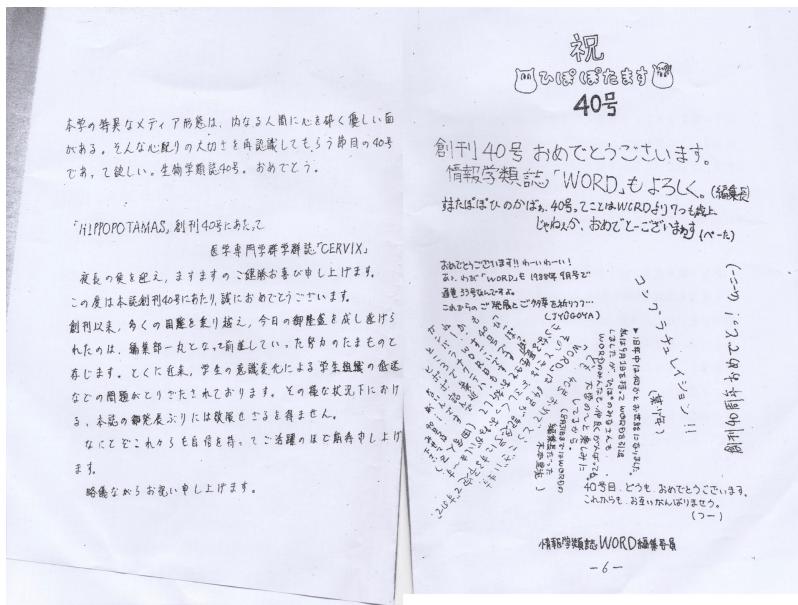


図9 『HIPPOPOTAMUS』vol.40を記念した『CREVIX』および『WORD』からのお祝いの言葉。ちなみに、お祝いの言葉はほとんどの学類誌が『CREVIX』のようなマトモな体裁で書いていたが、唯一ふざけていた（？）のが『WORD』でした。この当時から『WORD』らしさはあったようです。

が、ずいぶん大きくなつて…健康になつたぢやないか（引用者注：原文ママ）…うるうる……」と産婦人科のお医者さんが言ったか言わないかは知りませんが、なんとか形になってきました。

一歳にあと少しという今日、このごろ…………。もっと多くの人と交流をもちたいなあ……。というところ

医者を養成する専門学群（当時）らしい筆致ではないでしょうか。

1988年頃は、不定期に150部を発行していたようです。現在の医学群各学類では学類誌を発行していないことから、その後終刊した模様です。

ちなみに、「cervix」の語を英和辞典で引くと『解』くび、頸、頸部；子宮頸」という意味が出てきます¹⁶。前者の意味で取ったのか後者の意味で取ったのかは記述がないため定かではありません。個人的には、なぜこのような語を学類誌のタイトルに据えようと思ったのかはかなり気になるところではあります。先に引用した文章に出てきた「産婦人科」になぞらえるのであれば後者の意味で取るのがよいのでしょうか。

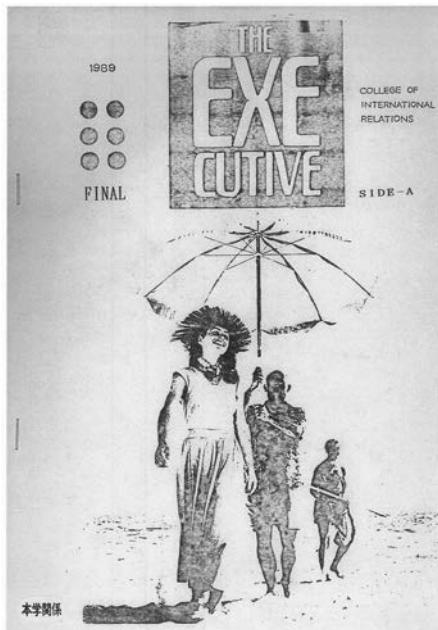


図 10 『THE EXECUTIVE』 vol.33 (SIDE A) の表紙。

4.10 國際關係學類誌『THE EXECUTIVE』

1983 年 12 月に vol. 1 が発刊^{*17}され、1989 年 1 月の vol. 33 で終刊となりました。最終巻の vol. 33 は珍しく 2 分冊で発刊され、『THE EXECUTIVE』との別れを惜しむ声が多数掲載されていました。2 分冊になったのは製本の都合でしょうか。国際關係學類時代に終刊したので、国際総合学類（1995 年発足^{*18}）には引き継がれなかった模様です。

最終巻の vol. 33 には「EXE 廃刊特集にあたって」という文章が掲載されていたので、引用しておこうと思います。

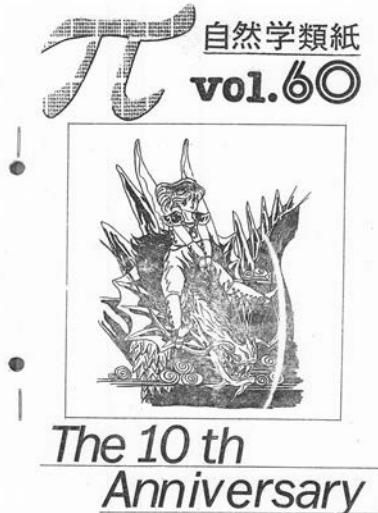
廃刊を記念して初めて EXE 自身の特集を組んでみました。特集を進める上で EXE を創刊号からすべて揃えてみましたが、バックナンバーを並べてみると EXE の変遷と伝統が見えてきて、興味深いと同時に EXE の歴史を絶ってしまうことに少なからず寂しさを感じます。

EXE は 1983 年 12 月に草創期の国闘における“学級新聞”的存在として創刊されました。詳しくはこれから始まる特集を読んでいただきたいのですが、EXE は回り道をしながらも確実に発展してきたと思います。「盛者必衰の理」というのはおこがましいと思いますが、現在の 3、4 年生が支えてきた体制に限界が見え始めてきたと同時に、EXE の体質で出来ることは、ほぼこなしたと判断するに至りました。また、手前味噌で申し訳在りませんが、4 月号から本誌で連載している「いま『国闘』」を

*16 リーダーズ英和中辞典第 2 版（研究社）による。

*17 vol. 33 の記述によります。vol. 1 は現存しないようです。

*18 <https://sse.tsukuba.ac.jp/outline/history/> より。

図 11 『 π 』 vol.60 の表紙.

問う」（引用者注：原文ママ）において、指摘されていたように、大幅な定員増と年月が経過は（引用者注：原文ママ）国閥を筑波大学に定着せしめたという状況から、国閥自体が大きく変化し、また変化しつつあると思います。そうした中でいい意味でも悪い意味でも国閥草創期の名残をいろ濃く残す“THE EXECUTIVE”という名前は、消滅してしかるべき時期にきていると判断しました。

この特集およびEXE33号を今までお世話になった方々、長い間愛読していただいた読者のみなさんに捧げたいと思います。どうもありがとうございました。そして、（引用者注：原文ママ）さようなら。

これを読む限りでは、編集部の判断での終刊となったようです。

4.11 自然学類紙『 π 』

自然学類紙は、1982年9月21日にvol.37が発行された後、1989年6月19日にvol.69が発行されています。第69巻が、確認できた最新の自然学類紙です。発行日の記載がありませんが、「なんばあ～14」も現存しています。「誌」ではなく「紙」が正式な表記のようで、『88年ノート』には『自然学類誌 π 』を『自然学類紙 π 』に訂正した痕跡が見られます。名前の『 π 』はvol.60で採用されたようです。ちなみに、このvol.60は1986年10月に発行され、これが10周年記念号のようですので、創刊号の発行は1976年……かと思ったら、

The 10th Anniversary — 10周年。これはウソです。本当は9周年です。でも昭和52年7月創刊だから、10年目と言い換えられるでしょ。それに今回で学類紙は60号。区切りがいいから10年記念しちゃいます。（本当にいいのかなー）

とのことです。何はともあれ創刊時期は 1977 年で確定としてよいでしょう。

自然科学類の後継の学類は、いずれも学類誌を刊行していないことから、1989 年以降に休刊または廃刊されたものと思われます。

ところで、旧自然科学類は現在でいうところの数学類・化学類・物理学類・地球学類に相当しますが、その中で数学ないし物理で頻出する π が選ばれたのはどうしてなのでしょうか。vol. 60 には

π —— パイ、円周率、3.14…

ついに決まりました！ 学類紙の名前。今年の 1 月からいろいろとおさわがせしましたが、この名前に決まってしまいました。どおしよう。(引用者注：原文ママ)

何はともあれ、決まってしまったのだから仕方がない。この名前でよろしくお願ひします。

(実はこの名前一週間前に決まったものだからまだ理由考えてない。だから次号まで待とう。)

とあります。肝心の vol. 61 が確認できていないので π の由来はわかりません。ただこの書き方から推測するに、そんなに深い理由はないのかもしれません。

少しおもしろい特徴としては、先生方の文章が掲載されていたこともあることでしょうか。『87 年ノート』には、物理学系（当時）の先生の手による「巨人ファンは科学者になれるか」と題された、次のような文章が掲載されています（元々は同年 1 月発行の学類誌に掲載されたものの改訂版のようです）。

定義：

世上「巨人ファン」と呼ばれている人達には、共通した体質 G_i ($i = 1, 2, \dots$) があります。以下では、 G_i を「巨人ファン症候」と呼びます。要素 G_i の全体は代数学的に一つの群 G を形成しますが、この G を「巨人ファン症候群」と呼ぶ人もあります。他方、—疑似ではなくて真正の—科学者は、幾つかの条件を備えていなければなりませんが、便宜上これらの条件を S_j ($j = 1, 2, \dots$) と書き、 S_j の集合を S します。さて、群 G に対して次の定理が成り立ちます。

定理：

G の任意の要素 G_i は、 S の適当な要素 S_j と矛盾する。

このあとに定理の証明が続きますが、今はそれは割愛します。この定理の主張するところは、「巨人ファンは科学者になれない（！）」というところでありましょう。元々筆者の先生は「“皆さん、もし巨人ファンであるならば、将来大科学者になり得る確率は非常に小さい”のではないか」との懸念を抱いていらしたようです。それを堂々と証明しようとするのがこの定理です。言うまでもなくこの定理はジョークの類だとは思いますが……。

本学関係

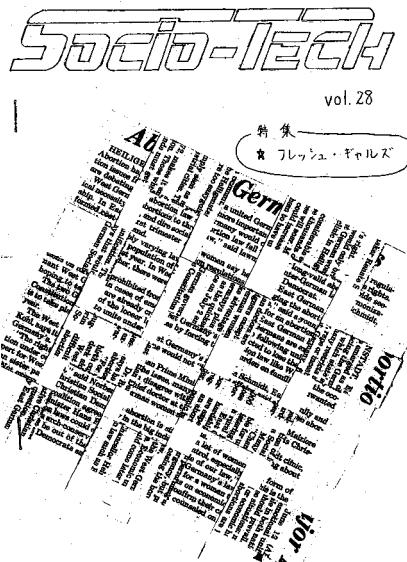


図 12 『socio-tech』 vol.28 の表紙。個人的には題字がカッコイイと思う。

4.12 社会工学類誌『socio-tech』

1985年1月25日付でvol. 0^{*19}が発行されました。確認できた最新の号は、1990年6月25日付のvol. 28です。「socio-tech」の名はvol. 0には登場しておらず、vol. 1以降に登場しています。現在では発行していないようなので、1990年以降のどこかのタイミングで休刊または終刊した模様です。

いつもの通り、vol. 0の「発刊にあたり」を引用しておきます。

学類誌なるものを見て、「なんだこりゃ？」（引用者注：原文ママ）と思う人は少ないでしょう。左様、学類誌はあなたのお察しの通りのものなのです。では、「なんでこんなもの作るんだ？」（引用者注：原文ママ）と言うと、答えは——。単純に、「作りたいから作るんだよ！」という答えが返ってくるのです。

事実、我々編集を行っている者どもは、各自様々な目的を持って作っています。
「みんなに何かをアピールしたい。」「面白そうじゃない。」「…作りたい…」etc.

でも1つ大きな目的としては、「もっと社工内でワイワイした雰囲気を作ろう。」そういうと、「人間関係を意図的に作るなんて、オレ（私）はヤダ」なんて思っている人がいるかもしれません、そう堅いコト言わずに……

「そのためには、学類誌なんて使えるの？」「大いに使えます！」

それは、あくまでも、みなさんが色々な意見や反論、批評等を学類誌に言ってくれ

^{*19}socio-techだけ「vol. 0」が存在します。0-indexedを好んだのでしょうか。だとしたら親近感が湧くような気もします。



図 13 『ize』 vol.1 の表紙。

たり、投稿すれば、その成果はかなりあると思います。僕らが興味を持っている人間で、まだ知らない人は、たくさんいます。まあ、中には「ワイワイした雰囲気なんぞなくともかまわん」と思う人がいるかもしれません、そんな人も、ちょっとは「学類誌」をのぞいてみてくださいな。

とにかく、平凡な言いまわしですが、「みんなで学類誌を作っていく」こういうことなんです。投稿待ってます！

発刊の動機は「作りたいから」、何と素直な動機ではないでしょうか。

4.13 比較文化学類誌『ize』

名前は『ize』ですが、表紙の発音記号を見る限り、また vol. 1 の記述（後述）によれば、タイトルの「ize」は「…にする[なる]」「…化する」などの意味をもつ²⁰“ize”のことのようなので、読み方は [aiz] で良いのでしょうか。1989 年 10 月に vol. 1 が発刊し、1992 年 2 月の vol. 8 をもって終刊したと、短命に終わりました。vol. 1 に掲載された「学類誌復刊にあたって」の文章を掲載します。

比較文化学類の学類誌が復刊されることになった。我々はこの作業を、今まで「COM」として馴れ親しまれてきた雑誌を「IZE」として生まれ変わらせることから始めた。それは今までの比文学類誌の流れとは一線を画すという意味があるのだが、それ以上に「IZE」という題名に込めた意味というものを読み取ってもらいたい。

²⁰リーダーズ英和中辞典第 2 版（研究社）による。

「IZE」というのは英語において、名詞、形容詞を動詞化させる機能を持つ接尾語である。さらにソシュール言語学にしたがって、言語を差異の体系と見るときに音声的に差異の無い、「EYES」という言葉も含意されている。つまり、物事を静態的にではなく、動態的にあくまでもそのダイナミズムのもとで見る目を持つ、といったような意味を「IZE」に込めているのである。このことはよりもなおさず、従来の学問体系がともすれば静態的に成り、物事のダイナミズムというものを見過ごしがちであったということを克服するべく登場してきた「比較文化学類」という学類の性格を象徴するとともに、この大変動の時代に、つまりは物事のダイナミズムというものを最も享受でき、かつしなければならない時代にあって、氾濫する情報に押し流され、ついにはそうした情報を静態的に、固定化したものとしてしか認識できなくなっている状況に対してのアンチテーゼでもある。一見逆説的ではあるがこういう時代には、氾濫する情報に対して一步立ち止まってじっくりと考えてみる必要があるのではないだろうか。こうした態度によってこそ物事のダイナミズムというものが見えてくるに違いない。さらにこうした態度というのは具体的には「自分の言葉で語る」ということに現われてくるのではないだろうか。

以上のような文脈を考えたとき「学類誌」というものが持つ意義というものは自ずと明らかになってくるだろう。

ともすれば時代の流れに流されがちな自分というものをしっかりと捕まえて、現在という状況のなかで主体的に生きていこうとする人たちの拠り所として学類誌はあるのである。つまりこの学類誌というのは、我々編集部の、あるいはそれを含めた一部の人のための雑誌ではなく、比較文化学類に所属する全ての人のための学類誌であり、そうなってこそ学類誌復刊という企ては成功するのである。この学類誌が成功するかしないかは読者諸賢にかかっている。そしてこのことはよりもなおさず、読者自身がこの状況のなかで主体的、積極的に生きていこうとする、その第一歩となるはずである。だからこの企てを対岸の火事として傍観することなく、積極的に参加してくれることを望むものである。

非常にアツい復刊にあたっての言葉ではないでしょうか。文字数にして 1,000 文字オーバーです。しかしながら、先述の通り『IZE』は 3 年ほどで廃刊となってしまいました。一体何があったのかは、vol. 8 の最後のページに掲載された次の文章から窺うことができるかもしれません。

二年半にわたって続けてきた IZE ですが、この号をもって終刊ということになりました。創刊当初から、僕たち一代限りで終わりにするつもりでしたが、ここまで急激にテンションが下がってしまうとは、正直言って思っていませんでした。

終刊にあたって、言いたいことはいろいろありますが、言うべきことは何一つないという気もします。一つだけ僕個人の気持ちを言わせてもらえば、僕は、あなたを怖がらせたり、あなたを黙らせるために、ことばを発してきたつもりは全くありま

せん。

もちろん、おきまりのあいさつと、ありきたりの会話と、見え透いた愛想笑いを際限なく続ける中で相手の腹の中を探るような、日常的な「おつきあい」がかったるくてしようがないから、ほんとうにいちばんたいせつだと思うことを、まっすぐに話したいから、クソみたいな「今」を蹴り飛ばして、もっときれいなところへ行きたいから、文章を書いているのである以上、日常べったりのことばづかいをすることはできません。

それが結果としてはかえって、僕のことばがあなたを拒絶しているように見えたのだとしたら、それは僕の文章力の稚拙さのせいだと思います。けれど、まるでただをこねるようですが、僕は、僕個人の文章にも、学類誌自体にも、もっといろんなアクションがほしかった。

まあ、もうすんだことをうだうだ言ってもしかたありません。みなさんも、そのためなら「自分」なんてどうなってもいい、というようなものをはやく見つけて、それを目一杯たいせつにしていってください、でないと、後は生きながら死んでいるような飼い殺しの毎日が待っているだけです。

少ししゃべりすぎたようです。最後になりましたが、今号の発行が遅れ、締め切り通りに投稿してくださった方にご迷惑をおかけしたことをおわびします。また、発行人になってくださった小澤先生、辻村先生、インタビューに応じてくださった諸先生、そして、今までご愛読してくださったみなさん、ありがとうございました。さようなら。

発行期間の短さと動いていた学生の変化があまりなかったためか、『ize』は例外的にすべての号が残存しています。

4.14 芸術専門学群広報誌『南大門』

『南大門』は、学類誌ではなく「広報誌」を名乗っていますが、その実体は学類誌に近いように思われる所以、ここで扱います。『南大門』の名は、5C 棟の形状から取っているようです^{*21}。なお、大学新聞 59 号では『南大門ニュース』の名前で掲載されていましたが、ここでは確認できた現物の表記にならって『南大門』と示します。「南大門」の読み方は「なんだいもん」でしょうか。

『南大門』は、大学新聞 59 号の記述によれば、1980 年にクラ代の決定を告知する広報誌として創刊されましたが、その後ミニコミとして発刊されていました。確認できた範囲では、1983 年 2 月 26 日に第 2*号^{*22}が発行され、その後 1994 年 9 月（発行日不明）に第 63 号が発行されたようです。この期間を考えると活潑に発行されていたことがわかります。現在の芸術専門学群では広報誌を発行していないので、その後に休刊または廃刊されたようです。発行主体は判然としませんが、「南大門編集室」と表記されている号もあります。

*21 大学新聞 59 号より。

*22 一の位の数字は判読不能でした。

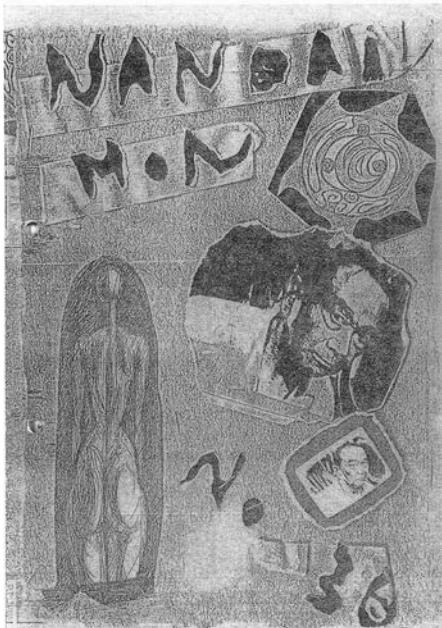


図 14 『南大門』 No.36 の表紙.



図 15 『のーのーりんりん』 第 8 号の表紙.

4.15 農林学類誌『のーのーりんりん』

農林学類誌は、1989 年に『農林学類誌』の名称で創刊されたようです。1991 年 4 月 30 日発行の第 8 号のころから『のーのーりんりん』という名称を利用しているようですが、どちらが正式名称なのは不明です。「のーのーりんりん」の由来は、第 8 号の表紙から推測するに「農林」の「農」と「林」を重ねたものようです。1998 年 6 月 9 日発行の第 34 号が最新の号ですが、第 34 号には次回予告があるため、これ以降も発行が続いたのではないかと思われます。

農林学類の後継である生物資源学類では、学類誌を発行していないことから、1998 年以降のタイミングで休刊ないし廃刊になったものと思われます。

4.16 社会学類誌『そしあへる』

比較的長生きした学類誌です。通称は「そしあ」のようです。大学新聞 65 号によれば、名前の由来は「ソーシャル」のようです。社会学類の英語名称 “College of Social Science” の social でしょうか。2009 年には WORD 編集部との対談が実現しています^{*23}。

1982 年 6 月 21 日^{*24} に創刊されました。創刊当初は人文学類誌編集委員会が発行に協力していたようです。当初の発行主体は「社会学類誌広報委員会」でしたが、1983 年 6 月 13

*23 対談の様子は <https://www.word-ac.net/post/2009/1030/> に掲載されています。

*24 WORD 第 11 号（2009 年 10 月発行、<https://www.word-ac.net/post/2009/1030/>）では 1981 年 6 月 1 日創刊とっていますが、実際の「vol.1」は 1982 年 6 月 21 日発行とされているため、ここではこの日付を創刊日とします。もっとも、『社会学類誌』よろしく創刊号の前に 1 号出ていたのかもしれません……。

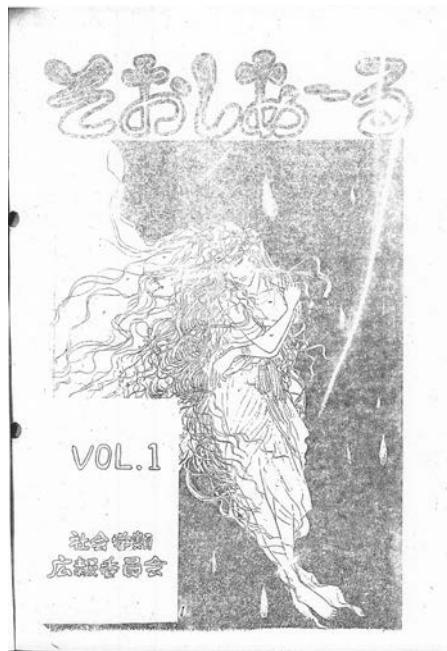


図 16 『そおしあ～る』 vol.1 の表紙.

日発行の vol. 3 以降は「社会学類広報委員会“そおしあ～る”出版社」名義で発行されています。この名義では少なくとも 1992 年頃までは発行していたようですが、2011 年以降は「社会学類誌委員会」が発行名義になっています。この 10 年の間に「“そおしあ～る”出版社」が取れたのでしょうか。

折角なので vol. 1 の巻頭言を引用しておきましょう。判読不能な文字は「＊」で示します。

ようやくのことで、社会学類誌を出す運びとなりました。「社会学類には活動的な人が多いのに、なぜ学類誌がないの？」と、友達や事務官の方からよく言われます。その原因については、諸説の飛び交うところなのでしょうが……。

とにかく、“新生”広報委員会になって初の学類誌発行にこぎつけて、今はホッとしているところです。

今“新生”広報委と言いましたが、それには理由があります。

去年までの広報委員会は、体質的に、クラス代表者会議の下部機関といった色あいが強く、広報委のメンバーはそれこそ「編集協力者」であり、「学類活動への協力」という大義名分の為に、大学生活の貴重な時間を削る、いわゆる奉仕者だったので。そういう事は、『広報委がしっかりしてくれんと、ワシらが困る』とか『学類誌が出ないから、学類活動が進まんのだ』(引用者注：このカッコは原文ママ)といった、クラス代表者会議や学類企画のお偉方のオコトバにも、アリアリとうかがえます。これ＊は余ほど、ヒマな人か、奉仕精神豊かな人でない限り、広報委に一年

も付き合ってられないでしょうし、だいたい個性豊かなツワモノの社会学類生がそんな事に始め（引用者注：原文ママ）から見向きもするハズがありません。だというのに、学類誌が出んあいのは、“広報委の怠慢”ということで安易にかたづけられてきました……。タワケ！ 問題はそんな所にはない！

「オレ達、私達は、学類活動の“書記”なんかではないぞ！」

これが新しい広報委員会の原点です。

私達は、学類活動に奉仕するつもりは全くありません。声の大きな諸君の主張の伝達なんか死んでもやる気になりません。（そんなのは、テメエらでビラを作れ！）大学当局とかいう人々と嫌悪なカケ引きの舞台に学類誌をするなんてマッピラ。

- ミニコミなのですから、社会学類生から遊離した存在になるのはイヤです。（引用者注：強調箇所は原文では文字サイズが大きくなっている。以下同様）
- 又、今回集ったメンバーには、それぞれにインタレストがあります。或者（引用者注：原文ママ）は編集好きだから、ある人はジャーナリスト志向、又ある人はタイプ打ちに興味があるので、又ある人は友だちが入ったから……。私達は、くら～い役所的なコトには興味なくて、和気アイアイとしたサークルの雰囲気でやる為に集まったのです。やりたいからやるのです。命令されたり、サイソクされてやるのはイヤです。

こうした、私達の姿勢に不満な方々もいて、広報委のメンバーを引き込んだり、あの手この手でメディアを利用しようと必死の活動家もいる様です。しかし、私達は（何度も言いますが）活動家のビラの編集ではなく、学類生のコミュニケーションと、広報委員間相互の親ボク（引用者注：原文ママ）（樂しければいい！）を目的としていることを宣言します。

まだ不慣れでミニコミとしてはまだですが、それなりに作ってゆきますので、今後ともよろしくお願ひします。

なかなか当時の苦労が滲み出る巻頭言ではないでしょうか。

2012年5月26日付のvol. 131は発行を確認しました。第132号以降は確認できていませんが、第131号には次号予告が掲載されていたので、その後もしばらくは発行していたものと思われます。大学新聞330号によれば、2014年に休刊し、そのまま現在に至っています。残念ながら最後に発行された号は確認できませんが、第132号はWORDとほぼ同様の体裁で発行されていたため、学類誌全盛期の面影を色濃く残していたものと思われます。

学類誌が発行されていた当時は、webページも開設していたようです。現在では閉鎖されていますが、<https://web.archive.org/web/20050115050927/http://www.first.tsukuba.ac.jp/social/top.shtml>でかつてのページを見ることができます。



図 17 『MAST』vol.1 の表紙（情報メディア創成学類の web ページで公開されているバックナンバーより）。

4.17 情報メディア創成学類誌『MAST』

情報メディア創成学類誌『MAST』は、2009年9月に創刊されました。名称は情報メディア創成学類の略称からと思われます。情報メディア創成学類の web ページ^{*25}によれば、2016年より紙媒体の発行を停止し、Web マガジン化していましたが、2020年度をもって休刊したようです。紙媒体時代のバックナンバーは情報メディア創成学類の web ページ^{*26}に掲載されています。

大学新聞330号によれば、web マガジン化の原因は編集員の減少と紙媒体制作作業の負担ということのようです。結局 web マガジン化した4年後に休刊になったことから、web マガジン化したもの編集員の減少はどうしようもなかったものと思われます。Web マガジン時代は <https://magazine.mast.tsukuba.ac.jp/> で記事が公開されていましたが、現在では閲覧できなくなっています。当時の web ページは、例えば <https://web.archive.org/web/20210318103647/https://magazine.mast.tsukuba.ac.jp/> で閲覧できます。当時の web ページを見る限り、最後の最後まで研究室紹介号などは発刊していたようですが、研究室紹介号の PDF は2017年入学者の管理するページに設置されていたようで、やはり休刊原因は「編集員がいなくなってしまった」とと思われます。

*25 <https://www.mast.tsukuba.ac.jp/outline/campuslife.html>

*26 <https://www.mast.tsukuba.ac.jp/outline/campuslife.html>

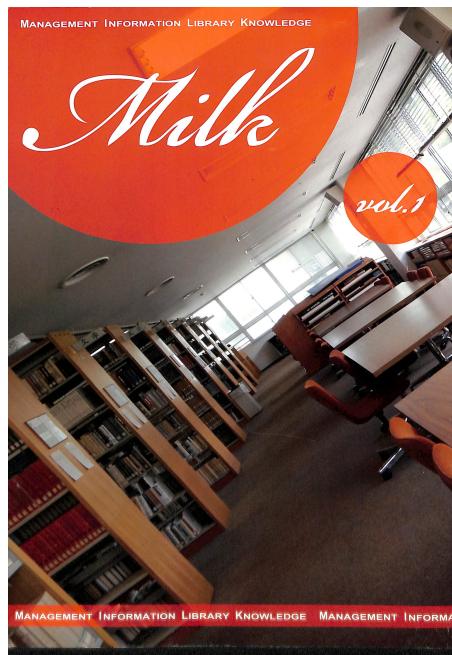


図 18 『MILK』 vol.1 の表紙 (MILK 公式 web ページのバックナンバーより)。

4.18 知識情報・図書館学類誌『MILK』

現在発行されている学類誌は我らが WORD のみ……と思ったら、知識情報・図書館学類の学類誌『MILK』も現在でも発行を続けている学類誌のひとつです。2011年9月に創刊されました。「MILK」は management, information, library, knowledge の頭文字を取ったものようです。management, information, knowledge はそれぞれ知識情報・図書館学類の情報資源経営主専攻・知識情報システム主専攻・知識科学主専攻から取ってきたものでしょう。残る library は知識情報・図書館学類の図書館の部分だと思われます。

バックナンバーは <https://milk.klis.tsukuba.ac.jp/> で閲覧できます。後発の学類誌らしく、グラフィカルな紙面デザインが特徴です。最新号は 2021 年 12 月 1 日発行のようです。

4.19 情報学類誌・情報科学類誌『WORD』

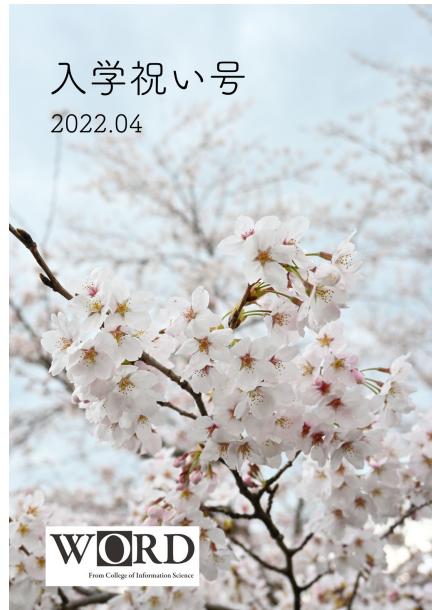
我らが『WORD』はここに位置します。1979年に創刊され^{*27}、現在に至るまで発行を続けています。途中に情報学類から情報科学類への改組があったようですが、『WORD』は引き継がれそのまま発行され続けています。直近では 2022 年 4 月 1 日に「入学祝い号」が発行されました^{*28}。

*27とはいえバックナンバー (<https://www.word-ac.net/backnumbers/>) で閲覧できるのは 2007 年以降の記事です。それ以前の記事は電子化が間にあってないのでしょうきっと。

*28WORD は慣例的に「第 N 号」というナンバリングはしないようです。なお、WORD のナンバリングについては後でまた触れます。



図 19 『WORD』創刊号の表紙.

図 20 本号が出る前の最新号だった
『WORD』入学祝い号（2022 年 4 月 1
日発行）の表紙.

2009 年には創刊 30 周年を記念したイベントが開催されたようです^{*29}。現在まで発行を続けている学類誌の中では最も老舗ということになります。また、『MILK』や『MAST』など、後発の学類誌はグラフィカルな組版を特徴をしている反面、『WORD』は LATEX ベースの簡素な組版と紙ベースでの発行を維持し続けており、学類誌全盛期の面影を色濃く残しています。

「WORD」という題字ですが、Wikipedia 日本語版の「筑波大学」の項^{*30}では、

タイトルの由来は情報量のワードであり、Microsoft Word からではない。

と記述されていますが、真偽のほどは不明です。

実際には、全代会の発行している『Campus』が 1975 年創刊^{*31}であるため、『WORD』はそれよりかは若い雑誌ということになります。とはいえ、全代会は学則上定期的に人間が補充されることが確実であるのに対し、『WORD』はそうではないことから、『WORD』が例外的に長期間に渡って発行し続けられてきたことの特異さが浮き彫りになります。ここまで特異的に長期間に渡って発行が続けられている理由は謎です。

第 1 号の「発刊にあたって」を引用しておきましょう。

*29 <https://www.word-ac.net/post/2009/0916/>

*30 <https://ja.wikipedia.org/wiki/%E7%AD%91%E6%B3%A2%E5%A4%A7%E5%AD%A6>

*31 大学新聞 369 号より。

「学類誌」として雑誌を発刊する事が出来る迄には、色々なことがありました。

それは約一年前、現在の一年生が丁度受験勉強に忙しかったであろう頃にクラス会議でまとまらなかったことに始まり（当時には、現在では既に解決されてしまっている問題があったのだけれど）、その後、今年度の一学期に有志によって、学類誌を意識しながらもまた別の立場から雑誌が出されました。それから、二学期になってクラス会議での承認が得られ、若干メンバー不足ではありますが、約十名の編集委員があちこち走り回り、どうにか今回の創刊に至ることができた。というのが、これ迄のいきさつです。

色々な苦労はありましたが、学類誌が情報学類学生間及び学生と先生方とのコミュニケーションを強めることに役立つように努力して行きたいと思います。しかし、乗務員不足の船はまだ港を出たばかりです。これから続く長い航海において座礁することのないように、しっかりと舵をとって行きたいと思います。また、皆様にもいろいろと御助言いただければ幸いです。

創刊時は苦労したようです。『HIPPOPOTAMUS』(4.8 節) で触れたような「揉め事」が影響していたものと思われます（ちょうど規則が出た年と WORD 創刊の時期が被っています）。そういう苦労があったものの、WORD は今日も「長い航海において座礁することのないように、しっかりと舵をと」り続けています。

さて、WORD の歴史を振り返ると、WORD にとって的一大転機のひとつは、バックグラウンドにいる学類の変更、すなわち情報学類の情報科学類への改組だったでしょう。事実、このタイミングで WORD が多少リニューアルしているようです。そこで、2007 年 5 月 30 日発行『WORD』Hello Word!号の「編集長挨拶」を引用してみましょう。

さて、本日のお日柄がよいのかどうかは天気予報に任せるとしまして、それはともかくにもどうも。情報学類誌から情報科学類誌に変わる際、役職も引き継いだため、情報科学類誌の初代編集長にもなってしまった Tait-you です。まあ、今回と次号だけの間ですが…。

と、言うわけで学群再編によってリニューアルを果たしました情報科学類誌 WORD、今回がリニューアル後の記念すべき一回目なので挨拶をしろと言われたのでしょうがなく書いている今日この頃です。別にこれといって変わっている部分は今のところ無いような気がしますけどねえ…。

まあ、せっかくリニューアルを果たしたことだし、何か大きく変わることもあると思います。とりあえず、タイトルは変わりました。また、今まで年度が変わることに No.1 にリセットされたナンバリングが今回からリニューアル創刊から何号目かを表すようになりました。って、表紙だけじゃーん中身はどうなの？ と言われるといつもどおりのような気がします。相変わらずほどほどにお馬鹿な記事を書いて、一体誰に需要があるのか分からぬ技術系の記事を書いていくという基本スタイルには一切の変更はありません。…と言いましたがこれを書いているのはなにぶん赤入



図 21 情報学類誌時代の WORD のタイトルロゴ。



図 22 情報科学類誌時代（現行）の WORD のタイトルロゴ。

れ前、どんな記事があるのかは現時点自分もさっぱり把握していません。それでいいのかと言わればそれでどうにかなっちゃうのが WORD のすごいところかもしれません。

…と与太話はこの辺にしておきましてここからは結構真面目な話。相変わらず軽い展開で進んでいく WORD ですが、結構重い話が無かったわけではありません。そして、この先もいつものようにのほほんと過ごしていくか、と尋ねられると非常に微妙な部分もあるかもしれません。この先、筑波大学が潰えるその時まで WORD が生き延びられる確証もありませんし、下手したら今年で廃刊なんて可能性も無きにしもあらずな訳です。いや、冗談抜きで。

ですが、このように長い間、当編集部が WORD を発行を続けてきていたのは今、冊子を手に取っているあなたがいるからなのです。WORD にニーズがあるからこそ、これからも WORD は新しい記事を提供して、時にはへえという感嘆の言葉や爆笑、あるいはクスリとくる笑いを提供していくことが出来るのです。ですから、これからもどうぞ、WORD を愛し続けてくれることを自分は切に願います。

…って随分壮大な事をののしちゃったけどこれってどうなのよ？ そんなでっかいバックフィールドは本当にあるの？ と最後にしっかりと惚けておいてこれにて締めとさせて頂きます。

Hello WORD!!

どうも過去には WORD 存続の危機もあったらしいです。現在は今のところ安泰な気もしますが……。

WORD のナンバリングについては、学類改組時に通番で振り直しているようです。とはいえ、この号が「Hello WORD 号」と銘打たれていることからもわかるように、通番で振り直したのは有名無実化しているようです。そして、これは現在でも変わっていません。

「タイトルは変わりました」とあるのは補足が必要でしょう。現在の「WORD」のタイトルロゴは、実はこのときに制定されたもので、情報学類時代はまた別のタイトルロゴ（図 21）を利用していたようです。学類誌の名称は『WORD』のまま変わらないので、「タイトルは変わりました」というのは、おそらくこのことを指すのでしょう。

表 1 1987–88 年頃、および 2016 年頃の WORD の情報. 『87 年ノート』『88 年ノート』および大学新聞 330 号より筆者作成.

項目	1987–88 年頃	2016 年頃
発行間隔	月 1 回 +α	年 5, 6 回
入手場所	3C213	第 3 エリア A・C 棟、図書館など
活動場所	3C213 (情報控室)	情報科学類 WORD 編集室
部数	500	変動あり

余談ですが、WORD は現在でも学類の公認のもとに活動しており、2022 年には学類 Web ページに掲載された「学類長からのご挨拶」*32 にも WORD が登場するようになりました。その文章の一部を引用します（強調は引用者による）。

情報科学類では、こうした、よい意味での自由な雰囲気を持ちながら、学生が自主的に様々な活動を行っています。たとえば、学類で運営している教育用計算機システムを学生自ら管理に携わっています。また、学類広報誌 WORD の編集や、産学間連携推進室では学外のベンチャー企業の皆さんと一緒にになって自由な発想で様々な活動を行っています。

WORD は広報誌だった……？ *33

WORD の今昔

せっかく本稿が WORD に掲載されているので、1987–88 年頃の WORD について少し突っ込んだことを書いておこうと思います。本小節の情報源は『87 年ノート』および『88 年ノート』です。

まず、1987–88 年当時の WORD のスペックが掲載されていたので、表 1 に抜粋して示します。このスペックは 87 年・88 年で同じ内容でした。表 1 には、参考までに大学新聞 330 号の情報をもとにして、2016 年現在の情報も付加しています。

まず目につくのは、発行間隔が今に比べれば遙かに短いことでしょう。現在の WORD はほとんど不定期発行状態になっていますが、当時は月 1 回のペースで発行していたようです。また、発行部数も近年の発行部数より多くなっています。近年は 2^n 部（典型的には 256 部）を発行することが多くなっていますが、「500 部」が「512 部」を丸めたものなのかどうかはわかりません*34。2016 年に何部発行していたのかは大学新聞には明記がありませんでしたが、この頃のバックナンバーを見る限りでは 192, 256, 386 など様々な部数で発行していました。どうやら綺麗に 2^n 部印刷し出したのは最近の傾向なのかもしれません。

入手場所および活動場所は、現在と大差ないようです。厳密には現在の活動場所は 3C212

*32 <https://www.coins.tsukuba.ac.jp/education/chairs/>

*33 そんなにオフィシャル度合いの高いものを出しているつもりはなかった……。

*34 キッカリ 500 部印刷していた可能性もありますが、他の学類誌の発行部数も含めて 50 部単位で掲載されていることから、50 部単位に発行部数を丸めてから掲載した可能性は捨て切れません。

であり、3C213（情報科学類ラウンジ）とは別の部屋番号が割り当てられていますが、この程度であれば誤差でしょう。当時からWORDが現在の情報科学類ラウンジ付近で活動していたことが読み取れます。ただし、当時の3C213室がどのような環境だったかはわかりません。そのため、現在のWORDの活動スペースとは異なる環境だった可能性も十分にあります。2016年ではラウンジと独立して「情報科学類WORD編集室」ができていることから、また大学新聞330号に掲載された写真からも、現在とほぼ同様の環境であったことが推察されます。

「入手場所」については、現在こそ3C棟2階の廊下のみで配布されている状態^{*35}ですが、昔は3A棟1階でも配布していたようで、現在でもWORDが設置したと思われるアンケートボックスが残っています。2016年の情報にある「第3エリアA棟」とはこのブースのことでしょう。また、1988年以降2016年以前のどこかのタイミングで図書館にWORDを置き始めて、2017年以降にそれが中止されたこともわかります^{*36}。附属図書館にもWORDがあった時代があったとは……。

当時の情報学類

さて、『大学ノート』に掲載された投稿からは、情報学類（当時）の雰囲気を読み取ることができます。『88年ノート』から抜粋しましょう。

しかし、情報学類というと伝統的に『暗い、ダサい、変態』という無茶苦茶なイメージを持たれているようですが、そんなことは断じてありません！！確かに我々は文系の人には嫌われがちな「こんぴゅーたー」なる機械を操っていますし、眼鏡をかけている人（またはコンタクトレンズをしている人）が異様に多いことも知られていますし、某JAMJAMでは明るさが最低値の40（絶対この数字は間違いでいると言っている私）ですので取り付きにくいのかもしれません。またごく一部にいる「1日5時間はコンピューターにアクセスしないと手が震える」とか、「酒買う金があったらフロッピーディスク買っちゃうもんね～」などという訳のわからんちょっとアブない人もいますので、その印象が強いのかもしれません。でも、損はないと思います。大多数の人はフツーの大学生なのです（うっ、思わず言い訳してしまいました）。ぜひお気軽にお声をおかけください、ほんとにいい人ぞろいであることがわかるはずです。

「フロッピーディスク」など時代を感じる記述もありますが、概ね現代とあまり変わらない気も……。もしかしたら、情報産業が盛んになり、全学的に情報教育がなされている現代のほうがマシな印象になってるのかもしれません。

*35だから人目につきにくく大学新聞にシカトされるんだぞ、と言われればその通りです。

*36多分面倒になったから置かなくなつたのでしょう。

4.20 未確認の学類誌

大学新聞 330 号によれば、2016 年当時は、生物資源学類が『資季彩菜』^{しきさいさい} という学類誌を発行していたようです。とはいっても、2016 年当時で一般向けの発行部数が極めて少なく、かつ現在まで発行しているかどうかもわからないため、現物をまだ入手できていません。大学新聞 330 号によれば、配布場所は生物資源学類の学生控え室とされていますが、そこを訪問してもそれらしきものはありませんでした。

また、大学新聞 330 号によれば、1978 年に社会工学類誌『海嘯』が発刊されたようですが、こちらも現物を確認できていません。

5 学類誌の特徴

ここでは、4 章で紹介してきた学類誌の特徴を示します。

- 学類誌の内容は、WORD と大差ありません。各学類誌の発行母体の専門に応じた内容で執筆されています。小説やマンガ、アンケートや読者の寄稿欄が設けられている雑誌もあります。初期の『そおしあ～る』のように、専門的な論文が掲載されていたケースもあります
- 昔に掲載された学類誌では、手書きで執筆された学類誌もありました。時代が下るにつれコンピュータで組版された学類誌も登場しました。これはイメージ通りだと思います。
- 多くの学類誌がモノクロ印刷ですが、『MAST』『MILK』に限っては InDesign などで組版されたと思われるフルカラーのページが特徴です。
- 多くの学類誌では、印刷されたページを綴じただけの装丁ですが、『南大門』などは（印刷されたページを綴じただけに比べれば）豪華な表紙がついていました。

6 学類誌を閲覧したい

これらの学類誌は、次の手順により閲覧できます。

- Web ページ上で公開されているものは、既に示した URL から閲覧できます（URL が示されていないものは、Web ページ上では公開されていません）。
- 中央図書館 2 階に大学関係の資料がまとまっている部屋があるので、そこに行くと閲覧できます。大学関係の資料の多くはきちんとしたハードカバーのついた装丁ですが、それに比べれば学類誌は明らかに粗末な装丁で、雑にフォルダケースに入っていたりバインダーに綴じられたりしているので、すぐにわかると思います。

7 学類誌の盛衰を概観する

ここでは、今まで述べてきたような、個別の学類誌に着目した記述ではなくて、一步引いてみて、学類誌の登場・廃止の変遷を辿ってみたいと思います。ただし、4.20 節で紹介した学類誌は、ここでは取り扱わないことにします。

表2 本稿で取り上げた学類誌のうち、確認できた最も古い年および最新の年の一覧。

『社会学類誌』は、発行年を確定できないため掲載していない。また、年にカッコを付したものは、ここに示した年以前（または以降）に学類誌が発行されていた可能性があることを示す。芸術専門学群誌『南大門』の「最古（年）」欄の1980年は、ミニコミの前身のクラ代の広報誌の創刊時期であるが、定義上これを『南大門』の創刊年として扱う。

タイトル	学類・専門学群	最古（年）	最新（年）
MonMon	人文学類	(1980)	1983
GLOBAL	国際関係学類	1983	(1983)
COM	比較文化学類	(1987)	(1987)
人文日報	人文学類	1986	(1988)
CORE	基礎工学類	1979	(1988)
さやにんげん	人間学類	1984	(1988)
HIPPOPOTAMUS	生物学類	1978	(1988)
CERVIX	医学専門学群	1987	(1988)
THE EXECUTIVE	国際関係学類	1983	1989
π	自然科学類	1977	(1989)
socio-tech	社会工学類	1985	(1990)
ize	比較文化学類	1989	1992
南大門	芸術専門学群	1980	(1994)
のーのーりんりん	農林学類	1989	(1998)
そおしあ～る	社会学類	1982	2014
MAST	情報メディア創成学類	2009	2021
MILK	知識情報・図書館学類	2011	(2021)
WORD	情報科学類	1979	2022

まずは、今まで登場した学類誌について、確認できた最古の年および最新の年をまとめると、表2のようになります。表だけではわかりづらいと思うので、この表をガントチャート式に図示すると、図23のようになります。

ここから、次のことを読み取ることができます。

- 少なくとも1987-88年頃に、様々な学類で学類誌が発行されていました。この時期を筑波大学における学類誌の全盛期と解釈できます。各々の学類誌の創刊時期ははっきりとしないところがありますが、開学（1973年）から15年後に学類誌が全盛を迎えることを考慮すると、どこかで「学類誌ブーム」が起こっていても不思議ではないでしょう。大学新聞330号では、1970年代後半から80年代前半にかけて「創刊ブーム」が起こった結果、1980年時点で14の学類誌があったとしています。この情報を信用すれば、ここで明らかになった学類誌以外にも様々な学類誌があった可能性があります。

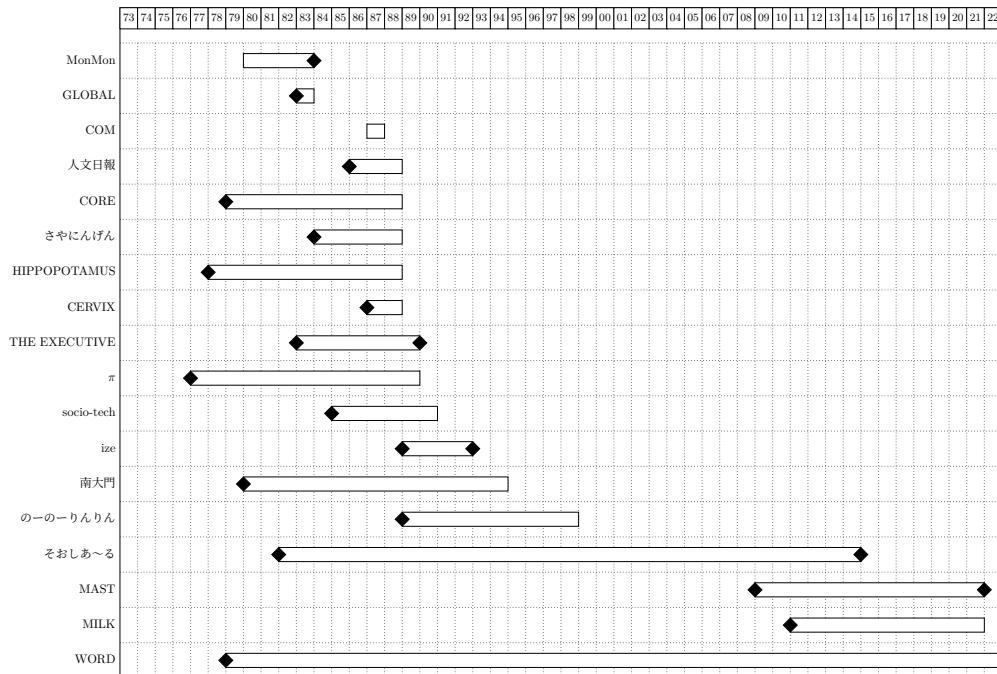


図 23 表 2 をガントチャートを用いて可視化したもの。◆印は、その年より前（または以後）に、その学類誌が発刊されたことがない（と推定される）ことを表す（表 2 ではカッコについていない箇所に◆印をついているので、注意してほしい）。

- 2000 年を跨いで発刊し続けられていたのは、『そおしあ～る』と『WORD』のみで、他の学類誌はその前に確認できなくなりました。その後 2000 年代前半は、この 2 誌のみが発行し続けていました。
- 2010 年前後になって、新興勢力である『MAST』および『MILK』が創刊されました。これらの学類誌は、新興勢力らしく、フルカラーのグラフィカルな組版を特徴としています。

8 学類誌のこれから

学類誌は、以上で見てきたように、現在では『WORD』および『MILK』のみが発行されています。これ以上学類誌が増えるか……と言われると、その可能性は低いと思われます。

学類誌の主な代替となっているのが、いわゆる「○つくば³⁷」というものです。情報系の人にわかりやすい言葉で翻訳すると、LT (Lightning Talk)³⁸ でしょう。「○つくば」の詳細な説明は、大学新聞 363 号³⁹ に譲ります。現在では、過去の学類誌が担っていた各学生

*37 この「○」は隠語ではなく、正規表現の「.+」の意味と思われます。

*38 ものによっては “lightning” とは言い難い時間発表していることもありますが、雰囲気的にはほぼこのような近似ができるとして構わないでしょう。

*39 <https://www.tsukuba.ac.jp/about/public-newspaper/pdf/363.pdf>

の研究成果発表は、「〇つくば」が担っていると考えられます。

「〇つくば」以外では、

- 人文社会系に関しては、^{そうほううろんそう} 雙峰論叢^{*40} が担っていると考えられます。
- また、それ以外の雑多なコンテンツは、各種ブログサービスや Twitter が発信を担っていると考えられます。
- または、各種サークルの発行する雑誌や、全代会『Campus』も代替として挙げられるかもしれません。

学類誌は、各学類がバックアップにつく以上、発行前の校閲については慎重に行われているため、発行までのコストが高いと考えられます。それに比べれば、より手軽に発表できる方法が確保されている以上、新規刊行の動機は薄いのではないかと思われます^{*41}。

個人的には、往年の学類誌文化を継承する、『WORD』および『MILK』が未永く発行されることを願うばかりです。

*40 <https://sohoronso.wicurio.com/>

*41 薄いにもかかわらず、噂によれば、いくつかの学類で学類誌の新規創刊・復刊の動きがあるようです。とはいえる、筆者本人はこれの裏は取れていません。

Bash わかるよ その気持ち

文 編集部 coord_e

本記事中のコードブロックは、特に明記がない限り Bash 4 以上で実行されることが想定されています。また、シェルの話をしていると POSIX sh との互換性が気になって仕方がない人がいることは存じ上げておりますが、この記事では Bash にしか興味を持たないのでグッと堪えて顔を Bash に馴染ませていただけると幸いです。

1 シェル

みなさん端末エミュレータを開いてコマンドを打っていますが、あれってどこに向かって打っているものかご存知ですか？つまり誰が俺たちのコマンドラインを吸ってプロセスの作成をしてくれているのか… そうシェルです。シェルというのが、バイナリのパスをいちいち覚えていられない俺たちのために、変数の値をすべて覚えていられない俺たちのために、便利なインターフェースを提供してくれているわけですね（やさしい）。

さてそのシェル 現在実行中のシェルがなんなのか、もしあなたが POSIX 準拠のシェルを使っているなら、次のコマンドで確認できます（そう、このコマンドを打っているシェルが何なのか^{*1*2}、です）。

```
1 $ echo $0  
2 bash
```

バニラの GNU/Linux だと大抵 bash が出てくるでしょう。最近の macOS では zsh だったりするかもしれません。もっともこの記事は Bash の話をするので Bash 以外に興味がない！ bash 以外が出てきた人は Bash を実行するなりして Bash に興味を持ってほしいところです^{*3}。

1.1 Bash

Bash はシェルの一種です。多くの GNU/Linux ディストリビューションにおいてデフォルトのシェルであり、意識しなくとも一度は触ったことがあるのではないかと思います。さて、多くの環境で利用可能であるということは、Bash に関する知識は非常に応用が効くということでもあります。たとえばサーバー上で一時的なバッチ処理を行うスクリプトが必要になったとしましょう、Bash に関する知識があればそのままの環境でスクリプトが書けるで

^{*1}正確にはコマンド名 argv[0] が出てくるだけなので、あんまりなんですが、まあ

^{*2}もっとも実行しているシェルに限らなければ、ユーザーのシェルのパスは \$SHELL とか getent passwd "\$USER" とかで出てきたりします

^{*3}とはいえる、この記事の内容は Z Shell (zsh) にも通ずるところは多い気がします。fish とかはちょっとわからない、ダメかもしれない。

しょうが、そうでなければわざわざ Ruby なり Python なりをインストールしなければいけません*4！嫌ですよね。嫌かどうかは置いておいても、環境に前提をおかないスクリプトの中では Bash のシェルスクリプトのシェアはかなり大きいと思います。

このようにシェアが広い Bash ですが、それに対して正しい知識があまり広まっていない印象があります。そんな Bash についての細かい知識を習得して Bash に理解のあるオタクくん顔をするのがこの記事の趣旨です。

1.2 シェルスクリプト

シェルスクリプトの話を先にしてしまった。シェルスクリプトというのはシェルコマンドを並べたテキストファイルです。行頭に `#!/bin/bash` とか書いてある*5 のを見たことがあると思いますが、それです。

2 基本

Bash でよく知られている基本的な機能の説明をしていきます。

2.1 変数

Bash には変数(**variable**)があります。変数は代入文(**assignment statement**)、`name=value` の形で代入ができる*6、`$name` の形で参照ができます。

```
1 $ a=hello
2 $ echo $a
3 hello
```

変数はわたしたちが代入文で設定するもの以外にも、シェルが自動的に設定する **シェル変数**(**shell variable**)が存在します。シェル変数は便利ではあるんですがそこまでおもしろいものでもないのでここでは説明しません、詳しくは `bash(1)` の Shell Variables の節を参照してください。

パラメータ

変数はパラメータ(**parameter**)の一種です。変数は名前で参照され先述の構文で代入ができますが、変数ではないパラメータは数字や記号で参照され代入文での代入ができません。例えば 1 章でも登場した `$0` は、シェルの名前が格納されている(変数ではない)パラメータです。

ゼロ 1 つ以外の数字列で構成されるパラメータを **位置パラメータ**(**positional parameter**)と呼びます。位置パラメータにはシェルが起動されたときの引数が順番に格納されています。シェルスクリプトであればそのシェルスクリプトの実行時の引数です。それ以外にも特

*4 正直 Bash の説明をする動機付けがしたいだけなので妄言を吐いている。Bash があるところには Perl とかもある気がする…

*5 これはシバン(shebang)と呼ばれているもので、`execve(2)` の仕様…らしい

*6 有名な話ですが、これは=の前後にスペースを許容しません

特殊パラメータ (special parameter) が存在し、シェルによって特別に扱われます。特殊パラメータの一覧とその展開のされかたを 表 1 に示します (展開については詳しく 3 章で説明します)。

表 1 特殊パラメータの一覧

\$*	\$1 から順番に位置パラメータに展開されます。ダブルクオートで囲まれている場合は展開後の位置パラメータは 1 つの語に、囲まれていない場合は別々の語に展開されます*7.
\$@	\$1 から順番に位置パラメータに展開されます。ダブルクオートで囲まれている場合は展開後の位置パラメータもダブルクオートで囲われ、囲まれていない場合は囲まれません (i.e. word splitting の対象になります)*8.
\$#	位置パラメータの数を十進数で表現して展開されます。
\$?	直近で実行された (非同期なものを除く) コマンドの終了コードに展開されます。
\$-	実行されているシェルに設定されているオプションフラグに展開されます。
\$\$	実行されているシェルの PID に展開されます*9.
\$!	直近で非同期に実行されたコマンドの PID に展開されます。
\$0	実行されているシェルかシェルスクリプトの名前に展開されます。

読み取り専用変数

Bash の変数は、他の一般的な手続的プログラミング言語の変数と同じく、最初の代入後に値を変更すること (再代入) が可能ですが、一方で定数を扱う場合や、趣味嗜好の問題で変数の値を変更をしたくないこともあるでしょう。Bash では `readonly` 組み込みコマンドを用いて*10、図 1 のように変数を読み取り専用にすることができます。

```

1 $ readonly CONSTANT=hello
2 $ echo $CONSTANT
3 hello
4 $ CONSTANT=1
5 bash: read-only variable: CONSTANT

```

図 1 読み取り専用変数

*7 1 つの語への変換の仕方や別々の語に展開された後の挙動などは `bash(1)` の Special Parameters の節を参照してください。

*8 展開が語の中で起きた場合の挙動、位置パラメータが空の場合の挙動などは `bash(1)` の Special Parameters の節を参照してください。

*9 1 章で実行中のシェルの名前しか出せませんでしたが、これを使えば (PID から実行可能ファイルのフルパスを求める方法が OS によりまちまちですが) 実行中のシェルのフルパスを求めることができるでしょう。

*10 `local` や `declare` に `-r` を渡すことでも同様の効果が得られます。

配列変数

Bash には配列変数があり^{*11*12}, いわゆる配列のデータ構造を扱うことができます。図 2 のように `name=(values...)` で配列変数に代入し, `${name[index]}` で値を参照することができます。意外なことに負の数値による^{*13} インデックスにも対応しており, Python などのそれと同様に末尾から数えてくれます。便利。

```

1 $ array=(a b c)
2 $ echo ${array[0]}
3 a
4 $ array+=(d)
5 $ echo ${array[3]}
6 d
7 $ echo ${array[-2]}
8 c

```

図 2 配列変数

さらに図 3 のように連想配列変数もあり, いわゆる連想配列のデータ構造を扱うこともできます。連想配列変数の宣言は `declare` や `readonly` といった変数を宣言する組み込みコマンドに `-A` フラグを渡して行う必要があります。

```

1 $ declare -A array=([a]=1 [b]=2 [c]=3)
2 $ echo ${array[b]}
3 2

```

図 3 連想配列変数

2.2 関数

よく知られた事実ではあるのですが, Bash ではプログラミング言語風の機能として, 関数を定義して実行することができます。`fname () compound-command` か `function fname ()` `compound-command` の形式で `fname` という名前の関数を定義できます。() とか書いておきながら引数を宣言するわけではない様子が愉快ですね。では引数はどう受け取るかというと, 2.1 節で説明した位置パラメータ`$1`, `$2`, ... が関数の引数として扱えるようにな

^{*11}配列の値があるわけではなくて, 変数に配列が格納できるだけ

^{*12}配列変数であって配列パラメータの概念は存在しないので 2.1 節の\$@パラメータとかが配列としてではなく特別扱いされるのすごいヤバいと思う。伝わりますかこの気持ち

^{*13}数値！？ Bash に数値は存在しないのでは！？ と思ったあなたの鋭い！ 配列変数の参照における添字部分は 3.4 節で説明する算術展開の対象になっており, 算術展開の結果として数値に評価される式が入ることになっています

ります。呼び出しも字句的には `fname` という名前のコマンドを実行することになるので、スクリプトの中で有効な子スクリプトを定義しているような感じになるわけですね。

ローカル変数

Bash の変数は、デフォルトでは図 4 のようにスコープを持たず、実行環境中で保持されます。このスコープのない変数は環境変数としての役割では適切であってもプログラミング言語の変数としては取り回しが悪いと感じることが多いでしょう。`local` 組み込みコマンドを用いることで、図 5 のように変数を関数スコープにすることが出来ます。

```
1 $ echo $var1 # unset
2
3 $ function f() { var1="I'm in f"; echo $var1; }
4 $ f
5 I'm in f
6 $ echo $var1 # var1 leaks from f
7 I'm in f
```

図 4 スコープのない変数

```
1 $ echo $var2 # unset
2
3 $ function g() { local var2="I'm in g"; echo $var2; }
4 $ g
5 I'm in g
6 $ echo $var2 # unset
7
```

図 5 関数スコープの変数

3 展開

Bash では、コマンドの実行前にコマンドラインの字句に対して **展開 (expansion)** が行われます。今まで自然と説明していたパラメータの参照 (`$parameter`) も、この展開プロセスの一環として処理されています。Bash には 7 種の展開が存在し、次の順番で処理が進みます。

1. ブレース展開
2. チルダ展開
3. パラメータ展開

4. 算術展開
5. コマンド置換
6. word splitting
7. パス名展開

ただし、語句の出現場所によって各種展開が起きる場合と起きない場合があるため注意が必要です^{*14}。以降、各種展開についてそれぞれの効果を説明していきます。

3.1 ブレース展開, パス名展開, チルダ展開

これらは実際に利用したことがある人も多いはず、スクリプティング以外の用途でも便利な機能です。ブレース展開 (**brace expansion**) は語句に含まれたブレース ({}) の中に書かれた文字列に応じて語句を展開します。ドット 2つ .. による展開とカンマ区切りによる展開があり、それぞれ 図 6 のように動作します。これらは規則的に名前がついたファイル/ディレクトリへの操作などに応用が効き便利なことは容易に想像できると思います。

```
1 $ echo test_{1..5}
2 test_1 test_2 test_3 test_4 test_5
3 $ echo test_{a,b,c}
4 test_a test_b test_c
```

図 6 ブレース展開の例

パス名展開 (**pathname expansion**) は * ? [] を含む語句についてファイルシステム上のエントリに基づいた展開を行います。いわゆるグロブ (glob) 展開といったものです。これはかなり日常的に使っている人も多いのではないでしょうか。図 7 に例を示します。* が任意の文字列、? が任意の 1 文字、[] が囲われた中で指定された任意の 1 文字にマッチし、マッチするファイル、ディレクトリ名の列に展開されます。

チルダ展開 (**tilde expansion**) はチルダ (~) で始まる語句についてディレクトリ名の補完を行うものです。~ がユーザーのホームディレクトリに展開されるのは有名な話です。それ以外にもチルダの後に続く文字列によって様々な展開がありますが、私があまり使ったことがないのでここでは説明を省略します。多分便利だと思うので詳しくは bash(1) の Tilde Expansion の節を参照してください。

3.2 パラメータ展開

Bash を語るうえでは外せないパラメータ展開について説明します。パラメータ展開 (**parameter expansion**) は \$ の後にパラメータを指定することで発生する展開です。パラメータをブレースで囲うこともでき、一部の文脈では続く語句とパラメータを区別するため

^{*14} どこで何が起きるみたいなのは私は網羅的にまとめることができなかった、注意しながら逐一調べるのがいいと思います

```

1 $ ls
2 file_1 file_2 file_3 file_a file_abc file_b
3 $ echo file_*
4 file_1 file_2 file_3 file_a file_abc file_b
5 $ echo file_?
6 file_1 file_2 file_3 file_a file_b
7 $ echo file_[0-9]
8 file_1 file_2 file_3

```

図 7 パス名展開の例

にプレースが必要になることがあります。

これは今まで `$variable` や `$0` のようにして展開してきた構文のことを指しているのですが、実はこのパラメータ展開は単純にパラメータの値を展開するだけではない豊富な機能を備えています。全機能は `bash(1)` の Parameter Expansion の節を参照していただくことにして、ここではいくつか特に使い所が多い機能を紹介します。

まずデフォルト値の展開。これは `${parameter:-default}` と書いたときに、基本は `$parameter` に展開しつつ `$parameter` が空だった際に `default` に展開するものです。例えばコマンドライン引数を受け取るときに `${1:-default}` とすることでコマンドライン引数が指定されなかったときのデフォルト値を簡単に指定できたりと、思ったより応用範囲は広いです。また 5.2 節で紹介する `-u` フラグを使用する際は未定義の変数参照がエラーになるため、この展開は必須級の機能になります。

次に先頭・末尾のパターンによる削除。 `${parameter#pat}` と書いたときに `$parameter` の展開のうち `pat` に一致する先頭部分を削除したものに展開されます。さらに `${parameter%pat}` と書いたときに `$parameter` の展開のうち `pat` に一致する末尾部分を削除したものに展開されます。どちらも記号を 2 回繰り返すと (`${parameter##pat}`, `${parameter%%pat}`) `pat` が最長一致になります(デフォルトは最短一致)。`pat` のパターンの構文とマッチ方法はパス名展開のパターンと同様です(glob パターン)。

最後にパターンによる置換。これは先ほどの削除と似ていますが、マッチした文字列の置換を行います。 `${parameter/pat/replacement}` と書いたときに `$parameter` の展開のうち `pat` に最初にマッチした部分を `replacement` に置換します。 `${parameter//pat/replacement}` と書くと、最初のマッチだけでなくすべてのマッチに対して置換を行います。

これらを用いるとシェルスクリプト上の文字列処理が非常に簡単になります。図 8 に応用例を示します。こういったユースケースでは `sed(1)` の利用が検討されがちですが、パラメータ展開を利用することでコードは単純になりますし、可搬性を向上しやすくなります。

```

1 $ echo ${non_existing:-default}
2 default
3 $ var=docker.io/library/bash:4.0
4 $ echo "name=${var%:*}"
5 name=docker.io/library/bash
6 $ echo "tag=${var##*:}"
7 tag=4.0
8 $ echo "ecr=${var/docker.io/public.ecr.aws/docker}"
9 ecr=public.ecr.aws/docker/library/bash:4.0

```

図 8 パラメータ展開の応用例

3.3 コマンド置換

コマンド置換 (**command substitution**) ではコマンドをコマンドの実行結果に展開します。具体的には, \$(command) または `command` と書くと^{*15}, command の標準出力に展開されます。

```

1 $ rev=$(git rev-parse HEAD)
2 $ echo "{\"revision\": \"$rev\"}" | jq . | tee revision.json
3 {
4   "revision": "3f786850e387550fdab836ed7e6dc881de23001b"
5 }
6 $ revision_json=< revision.json
7 $ echo "$revision_json"
8 {
9   "revision": "3f786850e387550fdab836ed7e6dc881de23001b"
10 }

```

図 9 コマンド置換の例

3.4 算術展開

算術展開 (**arithmetic expansion**) は算術式を評価してその結果に展開を行います。算術式^{*16}については詳しく解説しないので bash(1) の ARITHMETIC EVALUATION の章を読んでほしいですが、基本的には四則演算や論理演算を固定長の整数を値として行うことがで

^{*15}\$()の方がネストがやりやすいので基本的にこっちを使うといいでしょう

^{*16}Bash の中で式と呼ばれるもの（値に評価される再帰的に定義された構造、程度の感覚）はこの算術式と条件式しか存在しない！

きます。図 10 に算術展開の例を示します。算術式の中ではパラメータ展開を介さずに変数の値を参照することができます^{*17}。

```

1 $ not_num=1000
2 $ echo $not_num
3 1000
4 $ not_num=$(( not_num+500 ))
5 $ echo $not_num
6 1500

```

図 10 算術展開

また, `declare` や `local` といった変数宣言のためのコマンドに `-i` フラグをつけることで図 11 のようにその変数の代入時に自動的に算術展開を行うことができます。

```

1 $ declare -i num=1000
2 $ echo $num
3 1000
4 $ num=num+500 # arithmetic expansion without $(( ))
5 $ echo $num
6 1500

```

図 11 数値変数

3.5 word splitting

Bash はダブルクオートで囲まれておらず、代入文の右辺ではないパラメータ展開、コマンド置換、算術展開の結果について word splitting を行います。word splitting^{*18} は IFS 変数の値をデリミタとして 1 つの語を複数の語に展開します。IFS 変数には基本的にはホワイトスペースが入っているので、パラメータ展開、コマンド置換、算術展開の結果にホワイトスペースが含まれていて、かつそれがダブルクオートで囲まれていないと複数の語に自動的に分割されるということになります。一般的には word splitting はその挙動の分かりにくさから避けるのがよしとされパラメータ展開やコマンド置換はダブルクオートで囲むのが慣習になっていますが、挙動を理解していれば word splitting で適度に楽をすることも可能だった

^{*17} 算術展開の前に算術式がパラメータ展開含む各種展開を受けるため、パラメータ展開の形式で変数を参照することもできます

^{*18} 日本語表記が一般的に浸透していないような用語に怪しい日本語訳を勝手に当てるのが好きでこの記事でもずっとそれをやっているのだけど、word splitting だけはそれっぽい日本語を作れなかった…

ります*19.

4 代入

さて、2.1 節で変数と代入の方法について説明しました。そこで代入文という概念が登場しましたが、その代入文は実は Bash の構文に直接現れる構造ではありません。Bash がトップレベルで受け付けるのはコマンドリストか複合コマンド*20 のみであって、代入文と名のついた構文は存在しないのです。ではどこが代入文を受けているかというと、まずトップレベルで直接代入文を書いた場合はコマンドリストの構成要素である**単純コマンド (simple command)** の一部として代入文は処理されます。

単純コマンドは代入文を先頭に取ることができ、その後に語とりダイレクトの列が続く構文です。`bash(1)` 曰く：

A simple command is a sequence of optional variable assignments followed by blank-separated words and redirections, and terminated by a control operator.

つまり単純コマンドの構文は次の通りです*21：

```
1 | [name=[value] ...] [word1 word2 ...]
```

わたしたちが今まで代入文だと思っていたものは、単純コマンドのコマンド部分が省略されたものだったんですね。さて、代入文が付随したコマンドに実行形式に見覚えがある方もいると思います。例えばスクリプトの実行時に環境変数をセットする：

```
1 | $ RELEASE=1 ./build.sh
```

この時、コマンド部分に対する展開は変数の代入による環境の変更前に行われます。これは図 12 のように非直感的な挙動につながるため注意が必要です。

```
1 | $ var=1 echo $var # expands to "echo ", so output nothing
2 |
```

図 12 非直感的な展開順序

4.1 コマンド置換と終了コード

単純コマンドの終了コードは次のように決定されます。

1. コマンド部分の展開の結果、コマンド名が残らなかった場合:

*19配列変数の初期化を改行区切りのコマンド出力で行いたい時など...(Bash 4.4 以上なら `mapfile` でいいのだが、Bash 3 でも動く必要があるとだいぶ難しい顔になってしまうので)

*20それぞれの具体的な構文については執筆時間の都合から説明しません

*21ここではリダイレクトを無視して話を進めます

- (a) 前半の代入文にコマンド置換が含まれていれば最後のコマンド置換の終了コードをこの単純コマンドの終了コードとする
 - (b) 前半の代入文にコマンド置換が含まれていなければ終了コードは 0 とする
2. コマンド名が残った場合: 前半の代入文によって環境を変更した状態でコマンドの実行をし, その終了コードをこの単純コマンドの終了コードとする

このアルゴリズムはいくつかの興味深い不整合を引き起こします。代入文の右辺でコマンド置換を行った場合の終了コードについて、次の例を考えます^{*22}:

```
1 var1=$((false)  
2 echo $? # 1
```

これは、コマンド部分が存在しない単純コマンドの終了コードが最後のコマンド置換の終了コードであるとされていることから理解できます。一方で、次の例を考えます:

```
1 readonly var2=$((false)  
2 echo $? #0
```

こちらは代入文を `readonly` ビルトインの引数として用いています。この場合は単純コマンドのコマンド名が残っている場合として処理され、単に `readonly` コマンドを代入文を引数として実行することになります（前半の代入文は存在しない）。そして `readonly` コマンドは引数が不正でない限り 0 で終了するため、ここでは単純コマンド全体としてみたときの終了コードは 0 になります。

このように代入を行うにしてもコマンドを用いるか用いないかによって終了コードが伝播したりしなかったりします。これは 5.2 節で説明する `-e` フラグを用いたときに特に注意する必要がある挙動です。

5 役に立つほうの知識

これまで説明してきた知識のうち、役に立つ要素はこれから挙げるツールやおまじないでほとんどカバーされているのであまり身に着ける意味はないでしょう。

5.1 ShellCheck

`ShellCheck`^{*23} を使ったことはありますか？ ないなら今すぐ使ったほうがいいです。`ShellCheck` はシェルスクリプトのための静的検査ツールで、シェルスクリプトに関する専門的な知識がなくてもシェルスクリプトを書くうえで犯しがちなミスを検知して警告を生成します。基本的には本当にほとんどの落とし穴は `ShellCheck` でカバーされているイメージがあります、この記事を読むよりあなたの CI に `ShellCheck` を組み込むほうがかなり生産的な行為です。

*22 `false` は常に失敗の終了コードを返すコマンド

*23 <https://github.com/koalaman/shellcheck>

5.2 おまじない

シェルスクリプトを書くうえで書いておいて損がないおまじないを図 13 に示します。これは次の 3 点についてシェルの挙動を変更し、より直感的にシェルスクリプトが動作するようにします。

- 特定の場所を除いて、終了コードが 0 以外でコマンドが終了した際に、スクリプト自身を終了させる (-e)
- 未定義のパラメータを参照した際に、空文字列に展開するのではなくエラーとする (-u)
- パイプライン^{*24}について、最右のコマンドの終了コードを用いるのではなく、すべてのコマンドの終了コードが 0 でない限り成功しないようにする (-o pipefail)

```
1 | set -euo pipefail
```

図 13 おまじない

ただしこのおまじないについて、関数をコマンド置換の中で利用する場合に注意が必要です。Bash は(POSIX 互換モードではない場合に)外側の -e フラグをクリアしてコマンド置換を実行します。そのため関数をコマンド置換から実行すると、その関数は -e なしで実行されることになります。たとえスクリプトの先頭で set -e されていたとしても、関数の先頭では set -e しておくと良いでしょう。

さらに、readonly や local, declare といった代入系ビルトインを利用する場合に注意が必要です。4.1 節で説明したとおり代入系ビルトインを利用した場合は代入文の右辺のコマンド置換の終了コードが隠されてしまいます。できる限り代入系ビルトインは代入ではなく変数の属性を変更するためだけに使用し、代入は代入文を直接使って行うと良いでしょう^{*25}。

6 おわりに

私が詳しくないので書きませんでしたがリダイレクト周りにまだ知識の余地がある気がします…

7 参考文献

- BASH(1) - General Commands Manual
- GNU Bash Manual <https://www.gnu.org/software/bash/manual/>
- Bash の代入^{*26} <https://coord-e.github.io/slides-coins201t-assignments-in-bash/>

*24 ここでは|で繋げられたコマンドの列と思ってよい

*25 とはいってもこれも ShellCheck が指摘してくれるのでもあといった感じではありますね

*26 おいしい手前味噌

Coq の基礎の基礎

文 編集部 public_yusuke

1 はじめに

こんにちは、無事に B3 になった public_yusuke です。読者の方々も新生活に慣れてきた頃でしょうか。

さて、情報科学類生の卒業要件には、選択科目の中に「論理と形式化^{*1}」という授業が入っています^{*2}。私や私の友達の多くがこの授業を履修していました。この授業の前半では述語論理における命題の証明問題に取り組むのですが、私は初めて証明図を書くときにこれを難しく感じ、もうちょっと楽をしながら書けないものかと考えていました。

そこで、Coq という証明支援システムを使って証明を書いてみたところ、これがパズルのようで結構楽しく、やっているうちに証明図もサラサラと書けるようになりました^{*3}。

この記事では、述語論理でのさまざまな命題を Coq で証明できるようになるところまで、Coq の使い方を紹介します^{*4}。これを最後まで読めば、論理と形式化で A+ を取れるかもしれません^{*5}。

2 Coq とは

Coq とは、定理証明支援系の一つです。問題と証明を書いて与えると、各部分の論証が正当であるかどうかを確かめてくれます。また、Coq はインタラクティブな利用をすることが可能であり、証明中に次に示すべきことや現在導入されている仮定などを表示させながら証明を書いていくことができます。つまり、証明を書くのにとても便利なソフトウェアであるということなのです。

2.1 とりあえず使ってみよう

Coq は通常あなたのコンピュータにインストールして利用するソフトウェアです。しかし、軽くお試しをしたいだけなのにわざわざディスク容量を埋めたくないという方もいらっしゃるでしょう。そこで、この記事ではブラウザで動作する Coq である **jsCoq** (<https://coq.vercel.app/scratchpad.html>) を用いることにします。

*1 <https://kdb.tsukuba.ac.jp/syllabi/2022/GB12601/jpn/>

*2 <https://www.tsukuba.ac.jp/education/ug-courses-directory/2022/pdf/5-2.pdf> 81 ページ参照

*3 個人差があります

*4 Coq の内部的な部分や型の話などは書きません

*5 保証はできません……。

このページの左側では、

Alt + ↓ で証明を進める
Alt + ↑ で証明を戻す
Alt + Enter (または Alt + →) で証明を現在のカーソル位置まで進める

といった操作が可能です。

いま、試しにページの左側に次のようなコードを入力してみましょう。コードの意味はわからなくて大丈夫です。

```

1 Goal forall A B : Prop, A \vee B -> B \vee A.
2 Proof.
3   intros.
4   destruct H.
5   right.
6   exact H.
7   left.
8   exact H.
9 Qed.
```

入力中に `forall` や `\vee` の部分の表示が自動的に \forall や \vee に変わることがわかります。これを止めたい方は、右上の三点リーダをクリックして “Enable company-coq” のチェックを外すと無効化できます。

上記のコードを入力し、キーボード操作で最後の `Qed.` の部分まで証明を進めることができたら、次に進みましょう。

3 基本的な命題の証明

ここからは実際に証明を書いてみましょう。その前に、簡単のため、これから書く証明は以下の `Section` で囲まれたスコープの中に記述することにしてください⁶。

```

1 Section coq_kiso.
2 Variable A B : Prop.
3
4 (* ここに証明が入る *)
5
6 End coq_kiso.
```

⁶ ちなみに、(* ... *) で囲われた中の部分はコメントとして扱われます

3.1 「 \implies の導入」 intro, intros

まずは $A \implies B \implies A$ を証明してみましょう。通常、この命題の証明図は以下のようになるはずです。

$$\frac{\frac{[A]_1}{B \implies A} \implies I_{(2)}}{A \implies B \implies A} \implies I_{(1)}$$

この証明図に対応する Coq のソースコードは次のようにになります。

```

1 Goal A -> B -> A.
2 Proof.
3   intro.
4   intro.
5   exact H.
6 Qed.
```

3 行目の `intro.` を実行すると、Coq の画面に表示されている証明の対象が $A \implies B \implies A$ から $B \implies A$ に変化し、矢印の左側にあった命題 A が現在使える仮定として H という名前とともに追加されます^{*7}。したがって、`intro.` は \implies の導入に対応する命令であるとわかります^{*8}。

Goals
1 goal
$A, B : P$
$\frac{}{A \rightarrow B \rightarrow A}$

Goals
1 goal
$A, B : P$
$H : A$
$\frac{}{B \rightarrow A}$

図 1 2 行目の `Proof.` まで進めた状態

図 2 3 行目の `intro.` まで進めた状態

4 行目の `intro.` を実行すると、証明の対象は A に変化します。このとき、仮定 H は「 A が成り立つ」ことを意味しているので、`exact H.` で仮定 H がそのままゴールに対応していることを表せば、証明は完了します。

Goals
1 goal
$A, B : P$
$H : A$
$H_0 : B$
$\frac{}{A}$

Goals
No more goals.

図 3 4 行目の `intro.` まで進めた状態

図 4 5 行目の `exact H.` まで進めた状態

*7 `intro foo.` と書くと仮定に foo という名前を付けることができる

*8 この操作で矢印が消えて仮定が追加されているので、 \implies の導入っぽくないというか、むしろその逆のことをしているように感じるかもしれません、その考えは正しいです。Coq では証明図を下から書いていくように証明を進めるため、そのように見えるのです

証明中で使った `intro` や `exact` のような命令のことをタクティック (Tactic) といいます。この例で、タクティック `intro` は「 \implies の導入」に対応していることがわかりました。

複数の `intro`. をまとめる `intros`.

先程の証明は、次のように書くこともできます。

```

1 Goal A -> B -> A.
2 Proof.
3   intros.
4   exact H.
5 Qed.
```

タクティック `intros` は、タクティック `intro` をできるだけ繰り返し適用します。何かを証明したい場合、たいていの場合は仮定となるものがいくつか最初に付いていることが多いので、とりあえず `intros`. を証明の最初に置いておくと、準備として便利なことが多いです。以降の証明でも、証明の最初には `intros`. を書きます。

3.2 「 \wedge の導入・除去」 `split`, `destruct`

次は、 $(A \wedge B) \implies (B \wedge A)$ を証明してみましょう。証明図は以下のようになります。

$$\frac{\frac{[A \wedge B]_1}{B} {}^{\wedge E_2} \quad \frac{[A \wedge B]_1}{A} {}^{\wedge E_1}}{\frac{B \wedge A}{A \wedge B \rightarrow B \wedge A}} {}^{\rightarrow I^{(1)}}$$

この証明図に対応する Coq のソースコードは次のようになります。

```

1 Goal (A /\ B) -> (B /\ A).
2 Proof.
3   intros.
4   split.
5   destruct H.
6   exact H0.
7   destruct H.
8   exact H.
9 Qed.
```

4行目の `split`. を実行すると、証明すべきものが B と A に変わります。これは、「 \wedge の導入」に対応します。

Goals
1 goal
$A, B : P$
$H : A \wedge B$

Goals
2 goals
$A, B : P$
$H : A \wedge B$

図 5 3 行目の `intro.` まで進めた状態図 6 4 行目の `split.` まで進めた状態

最初に証明すべきは B ですが、仮定の中には B がありません。困りましたね。しかし、仮定 H から明らかに B が成り立つと言えそうです。ここで `destruct H.` を実行すると、仮定として A と B が追加され、証明を進めることができます。これは「 \wedge の除去」に対応します。

Goals
2 goals
$A, B : P$
$H : A$
$H_0 : B$

Goals
1 goal
$A, B : P$
$H : A \wedge B$

図 7 5 行目の `destruct H.` まで進め
た状態図 8 6 行目の `exact H0.` まで進めた状態

後は、同様の方法で A も証明することで、全体の証明が完了します。

3.3 「 \vee の導入・除去」(`left`, `right`), `destruct`

まだまだ続きます。次は、 $(A \vee B) \implies (B \vee A)$ を証明してみましょう。証明図は以下のようになります。

$$\frac{[A \vee B]_1 \quad \frac{\frac{[A]_2}{B \vee A}^{\vee I_2} \quad \frac{[B]_2}{B \vee A}^{\vee I_1}}{B \vee A}^{\vee E(2)}}{A \vee B \rightarrow B \vee A}^{\rightarrow I(1)}$$

この証明図に対応する Coq のソースコードは次のようになります。

```

1 Goal (A \vee B) -> (B \vee A).
2 Proof.
3 intros.
4 destruct H.
5 right.
6 exact H.
7 left.

```

8 exact H.
9 Qed.

4 行目の `destruct H.` は、「 \vee の除去」に対応します。これによって、「 A が成り立つときと B が成り立つときそれぞれで $B \vee A$ を証明する」状態に入ります。



図 10 4 行目の `destruct H.` まで進

図 9 3 行目の `intros.` まで進めた状態

5 行目の `right.` は、「 \vee の導入」に対応します。ここでは「 A が成り立つ」という仮定 H があるので、 $B \vee A$ の右側の A が成り立つことを証明すればよいだろう、という考えで使います。すると「 A が成り立つときの $B \vee A$ の証明」が完了し、次は「 B が成り立つときの $B \vee A$ の証明」をする状態に入ります。



図 11 5 行目の `right.` まで進めた状態

図 12 6 行目の `exact H.` まで進めた状態

7 行目の `left.` は、「 \vee の導入」に対応します。ここでは「 B が成り立つ」という仮定 H があるので、 $B \vee A$ の左側を証明することを選びました。あとは `exact H.` を実行することで「 B が成り立つときの $B \vee A$ の証明」が完了します。これで、全体の証明が完了しました。

3.4 「 \Rightarrow の除去」apply

最後に、 $\neg A \Rightarrow (\neg A \Rightarrow B) \Rightarrow B$ を証明してみましょう。証明図は以下のようになります。

$$\frac{\frac{[\neg A \rightarrow B]_2 \quad [\neg A]_1}{B} \rightarrow_E}{\frac{(\neg A \rightarrow B) \rightarrow B \rightarrow I^{(2)}}{\neg A \rightarrow (\neg A \rightarrow B) \rightarrow B \rightarrow I^{(1)}}}$$

この証明図に対応する Coq のソースコードは次のようにになります。

```

1 Goal  $\sim A \rightarrow (\sim A \rightarrow B) \rightarrow B$ .
2 Proof.
3   intros.
4   apply H0.
5   exact H.
6 Qed.
```

`intros.` の後のサブゴールは B となっています。このとき、仮定 $H0$ より「 $\sim A$ が証明できれば $\sim A \implies B$ を使って B を証明できる」ことがわかるので、`apply H0.` でサブゴールを $\sim A$ に変えます。これは「 \implies の除去」に対応します。そうしたら、仮定に $\sim A$ が成り立つことが含まれるので、それを `exact H.` で適用して証明が終了します。

Goals
1 goal
$A, B : P$
$H : \sim A$
$H_0 : \sim A \rightarrow B$
<hr/>
B

Goals
1 goal
$A, B : P$
$H : \sim A$
$H_0 : \sim A \rightarrow B$
<hr/>
$\sim A$

図 13 3 行目の `intros.` まで進めた状態 図 14 4 行目の `apply H0.` まで進めた状態

「 \sim の除去」と「 \sim の導入」は？

$\sim A$ が $A \implies \perp$ と同値であることを示してみましょう。なお、Coq では命題の同値関係は \leftrightarrow を使うことで表すことができます。

```

1 Goal ( $A \rightarrow \text{False}$ )  $\leftrightarrow \sim A$ .
2 Proof.
3   unfold iff. (* "A  $\leftrightarrow$  B" の記法を "(A  $\rightarrow$  B)  $\wedge$  (B  $\rightarrow$  A)" に直す。なくてもよい *)
4   split.
5   intros.
6   intro.
7   apply H.
8   exact H0.
9   intros.
10  apply H.
11  exact H0.
12 Qed.
```

5 行目に `intros` があるのにもかかわらず 6 行目で `intro` を実行しているのは冗長に見えるかもしれません。これによって仮定に A が入り、サブゴールが \perp となります。実は Coq での $\sim A$ は $A \implies \perp$ の別表記でしかありません⁹。ただし、この `intro` については

⁹このことは、証明中の好きなところで `unfold not.` を実行することで確かめることができます

「 \neg の導入」と同等なものとして見ることができる、すなわち「証明の最中に \perp が現れてしまつたので、仮定に入っていた A を否定した $\neg A$ が成り立つことにする」といった議論と同じことを示しているという点において興味深いです。

Goals
2 goals
$A, B : \text{P}$
$H : A \rightarrow \text{False}$
$\neg A$
subgoal 2 is:
$\neg A \rightarrow A \rightarrow \text{False}$

図 15 5 行目の intros. まで進めた状態

Goals
2 goals
$A, B : \text{P}$
$H : A \rightarrow \text{False}$
$H_0 : A$
$\neg A$
subgoal 2 is:
$\neg A \rightarrow A \rightarrow \text{False}$

図 16 6 行目の intro. まで進めた状態

また、10 行目の apply H. は「 \neg の除去」と同等なものとして見ることができます^{*10}。

Goals
1 goal
$A, B : \text{P}$
$H : \neg A$
$H_0 : A$
$\neg A$
False

図 17 9 行目の intros. まで進めた状態

Goals
1 goal
$A, B : \text{P}$
$H : \neg A$
$H_0 : A$
$\neg A$
A

図 18 10 行目の apply H. まで進めた状態

4 演習問題

次の問題を解いてみましょう。ただし、それぞれの証明は各問題に付いているソースコードを使って書き始めてください。

1. $A \vee A \implies A$
 - Goal forall A : Prop, A \vee A -> A.
2. $A \implies A \vee A$
 - Goal forall A : Prop, A -> A \vee A.
3. $(A \vee B) \wedge C \implies (A \wedge C) \vee (B \wedge C)$
 - Goal forall A B C : Prop, (A \vee B) /\ C -> (A /\ C) \vee (B /\ C).

以下、それぞれの解答です。解けましたか？

```
1 | Goal forall A : Prop, A \vee A -> A.
2 | Proof.
```

*10 ちなみに、命題 A について、仮定として A と $\neg A$ が同時に入っているときには contradiction というタクティックを使うことで証明をすぐに完了することができます

```

3   intros.
4   destruct H.
5   exact H.
6   exact H.
7 Qed.
8
9 Goal forall A : Prop, A -> A \vee A.
10 Proof.
11   intros.
12   left.
13   exact H.
14 Qed.
15
16 Goal forall A B C : Prop, (A \vee B) /\ C -> (A /\ C) \vee (B /\ C).
17 Proof.
18   intros.
19   destruct H.
20   destruct H.
21   left.
22   split.
23   exact H.
24   exact H0.
25   right.
26   split.
27   exact H.
28   exact H0.
29 Qed.

```

4.1 証明をもっと短く書く

これらの証明はちょっと冗長に感じるところがあります。例えば、1つ目の証明では `destruct H.` によって生成された新たな2つのサブゴールどちらに対しても `exact H.` を適用していますが、これらは次のように書くことでまとめることができます。

```

1 Goal forall A : Prop, A \vee A -> A.
2 Proof.
3   intros.
4   destruct H; exact H.
5 Qed.

```

`tactic1; tactic2` のように書くと、「`tactic1` によって生成された新たなサブゴールそれぞれに対して `tactic2` を適用する」という意味になります。これで証明を短くすることができます。

3つ目の証明は、かなり短くすることができるので、その完成形だけ掲載します。

```

1 Goal forall A B C : Prop, (A \ / B) /\ C -> (A /\ C) \ / (B /\ C).
2 Proof.
3   intros.
4   repeat destruct H; [left | right]; split; assumption.
5 Qed.
```

`repeat tactic1` を使うと、`tactic1` をできるだけ適用します。これで `destruct H.` を2回書かなくて済みました。

また、`[tactic1 | tactic2 | ... | tacticN]` のように書くと、セミコロンで渡ってきた X 番目のサブゴールに `tacticX` を適用することができます。ここでは、`repeat destruct H;` によって渡されたサブゴールの1つ目に `left`、2つ目に `right` を適用しています。

その後、`split` に対して2つのサブゴールを渡しています。この `split` によって、各サブゴールについて2つのサブゴールが新たに作られるので、`assumption` タクティックに渡されるサブゴールは合計で $2 \times 2 = 4$ つになります。

最後に、`assumption.` によってすべてのサブゴールの証明が完了します。このタクティックは何をしているのでしょうか。`assumption` タクティックは、サブゴールと同じものが仮定の中に含まれているならば、それを適用して証明を終了させるという動作をします。短縮バージョンの前の証明では、各サブゴールの証明は最終的に `exact` タクティックを用いて完了していたこと、また短縮前の証明中の `exact H.` と `exact H0.` の部分は `assumption.` に置換できるということから、この技を使うことができました。

5 おわりに

ここまで知識を使えば、論理と形式化で扱われる論理式の証明を書けるようになるでしょう。 \forall と \exists の導入と除去については述べませんでしたが、これまでのことが身に付いていれば、おまけとして最後に貼ってある「証明規則とタクティックの対応表」を見ながら証明を書くことができるはずです。

授業を受けていると、証明図を書くときに何をどう書けばいいか流れがつかめないとき、証明図をすぐに書き終わってしまって暇などがあると思います。Coq で遊びながら授業を受けると、一限の授業でも眠くなるのを防げるし、書いた証明を友達にチャットでシェアしながら一緒に楽しむこともできると思います。授業を楽しみながら受ける工夫として、ぜひ一度試してみてください！

なお、記事中の証明図は `ipc_solver`*¹¹ を利用して生成しました。

この記事は、過去の LT で喋った内容を記事に起こしました*¹²

*¹¹https://github.com/qnighy/ipc_solver

*¹²https://docs.google.com/presentation/d/1nBuARG1KvV1NWakoWxYCEExE3AtphH9UyVkJVzS_7xOKo/edit#slide=id.gd971327930_0_170

おまけ1 自動的に証明をさせる

さて、ここまででは人が証明を手で書いてきましたが、そろそろ「命題論理の証明問題ぐらいコンピュータが自動的に証明してくれないのか」という気持ちになってきたと思います。筆者もそう思います。なんと、Coq はそれをやってくれます。3つ目の演習問題を証明させてみましょう。

```

1 Goal forall A B C : Prop, (A \wedge B) \wedge C -> (A \wedge C) \vee (B \wedge C).
2 Proof.
3   tauto.
4 Qed.
```

`tauto` タクティックによって、証明を完了することができました。本当に証明できているのか、という疑問が出てくるので、`tauto.` の次の行で `Show Proof.` を実行し、証明を見てみましょう。すると、画面右下に次のように表示されます。

```

1 (fun (A0 B0 C : Prop) (H : (A0 \wedge B0) \wedge C) =>
2   and_ind
3     (fun (H0 : A0 \wedge B0) (H1 : C) =>
4       or_ind (fun H2 : A0 => or_introl (conj H2 H1))
5         (fun H2 : B0 => or_intror (conj H2 H1)) H0)
6   H)
```

これは、次のような証明と同等です。

```

1 Goal forall A B C : Prop, (A \wedge B) \wedge C -> (A \wedge C) \vee (B \wedge C).
2 Proof.
3   intros.
4   induction H. (* ← and_ind に対応 *)
5   (* ↓ or_ind に対応 *)
6   induction H; [left | right]; split; assumption.
7 Qed.
```

`induction H.` は、仮定 H についての数学的帰納法で証明を進めることを表すためのタクティックですが、この証明問題でわざわざ \vee や \wedge のことをそのような目で見て証明する人はほとんどいないでしょう。このように、自動的に生成された証明では、あまり人が書いた証明とは思えない結果となることがあります。

または <https://bit.ly/3sDYJTA>

おまけ2 自然数に関する定理を証明してみる

Coq は、様々な分野の定理を証明するのに使えます^{*13}。ここでは n が自然数のとき、 $n \leq n^2$ となることを証明してみます。

```

1 Require Import Arith.
2 Goal forall n : nat, n <= n*n.
3 Proof.
4   intros.
5   induction n.
6   - reflexivity.
7   - rewrite <- Nat.add_1_r.
8     rewrite Nat.mul_add_distr_r.
9     rewrite Nat.mul_add_distr_l.
10    rewrite Nat.mul_add_distr_l.
11    rewrite Nat.mul_1_l.
12    rewrite Nat.mul_1_l.
13    rewrite Nat.mul_1_r.
14    rewrite Nat.add_assoc.
15    apply Plus.plus_le_compat_r.
16    apply Plus.le_plus_trans.
17    apply Plus.le_plus_trans.
18    exact IHn.
19 Qed.
```

この証明について詳細な説明はしませんが、このように簡単な命題だけでなく別の分野の証明も可能です。

ちなみに、この定理も Coq に自動的に証明させることができます。

```

1 Goal forall n : nat, n <= n*n.
2 Proof.
3   intros.
4   induction n; lia.
5 Qed.
```

標準の自然数ライブラリに慣れないまま自分で証明を書いたときには1時間半ぐらいかかったのですが、タクティック lia を使ったら1秒もかからずに証明が構築されました。すごい。

^{*13}Coq で四色定理を証明した例がある。参考：“A computer-checked proof of the Four Colour Theorem” by Georges Gonthier (Microsoft Research Cambridge), <http://www2.tcs.ifi.lmu.de/~abel/lehre/WS07-08/CAFR/4colproof.pdf>

おまけ3 証明規則とタクティックの対応表

	導入 (introduction)	除去 (elimination)
\rightarrow \Rightarrow	intro. $\frac{[A] \leftarrow \text{仮定が増える}}{A \Rightarrow B}$ $B \leftarrow \text{サブゴールが変化}$	apply H. $\frac{[A \Rightarrow B]_H}{B}$ $A \leftarrow \text{サブゴールが変化}$
\wedge \wedge	split. $\frac{A \quad B \leftarrow \text{サブゴールが2つに増える}}{A \wedge B}$	$\frac{[A \wedge B]_H}{A}$ destruct H. $\frac{[A \wedge B]_H}{B}$
\vee \vee	left. $\frac{A \leftarrow \text{サブゴールが変化}}{A \vee B}$ right. $\frac{A \leftarrow \text{サブゴールが変化}}{B \vee A}$	destruct H. $\frac{[A] \quad [B] \leftarrow \text{サブゴールが2つに増える}}{C}$ $C \leftarrow \text{仮定が増える} \rightarrow B$
\sim \neg	intro. $\frac{\perp \leftarrow \text{サブゴールが変化}}{\neg A}$ $\neg A \leftarrow \text{仮定が増える} \rightarrow \perp$	apply H2. apply H1. $\frac{[A]_{H1} \quad [\neg A]_{H2}}{\perp}$ (または contradiction)
False \perp		exfalse. $\frac{\perp \leftarrow \text{サブゴールが変化}}{A}$
forall \forall	intro. $\frac{F(a) \leftarrow \text{仮定が増える}}{\forall x F(x)}$	specialize (H a). $\frac{[\forall x F(x)]_H}{F(a)}$ 仮定が変化 $\rightarrow F(a)$
exists \exists	exists t. $\frac{F(t) \leftarrow \text{サブゴールが変化}}{\exists x F(x)}$	destruct H. $\frac{[\exists x F(x)]_H}{C}$ $C \leftarrow \text{サブゴールが変化} \rightarrow F(a)$

図 19 証明規則とタクティックの対応表

Coq で証明を書いた後に証明図に起こしたいときや、どんなタクティックが使えるか迷ったときに役立つかかもしれません。

博士号回収記録

文 編集部 cely chan/tactilium



図1 博士号画像コレクション。左: 出現直後（2013年ごろ）に松見池を泳ぐ博士号。人が搭乗しており、機体後部から水流が出ていることがわかる。中央: 越冬に備え、陸上で装備を整える博士号。恥ずかしさのためか、身をブルーシートで覆っている。右: 最近の博士号。1H棟付近の排水口に、極太チェーン2本を用いて係留されている。

1 博士号とは

博士号とは、2013年10月15日に筑波大学天王台地区にある松見池に突如出現したアヒル・ボートである^{*1}。学生によって設置されたとされているが、詳細を知る者は2022年現在ほとんどいない。インターネット上の史料によると、かつて松見池からの撤去を目的として、クレーンによる吊り上げ、ヘリウム風船による浮上といった大規模作業が行われたが、現在も設置当時と同様に、松見池のほとりに佇んでいる。

この博士号を松見池に設置できたのは、こちらの記事 [[hatena](#)] で説明されているように、「筑波大学は国立大学法人化した後も依然として「行政主体」である。」ことと、「一般的な大学と異なり、筑波大学のキャンパスは、自他共に公共の公園として慣習的に認知されている。」ことから、「筑波大学の公園部分の中にある池は「行政主体が管理する公共の池」として共用開始された池である」ことが言え、「筑波大学の池で一般使用（水浴びやボート遊びなどの許可不要な使用）を行うことは、筑波大学関係者のみならず、学外の一般市民にとっても可能である」ということができる。」ためである（括弧内は記事より引用）。

博士号設置当時の様子を知る人物によると、撤去時にはクレーンや風船などの道具を用いて博士号を移動させたが、設置時には人力で博士号を運んだそうだ。Twitter上の、筑波大学にゆかりのあるけしからんアカウントのヘッダ画像は、博士号らしきアヒル・ボートを車で牽引している写真が使われている。断言することは困難だが、写真に写っている規模の自動車を池に横付けすることは困難であるから、博士号設置の際にはまず原産地から車で輸送し、最後は人力でもってして松見池に投げ込んだと考えられる。

^{*1}スワン・ボートだと述べる不届者が学内に一定数存在するが、そんな戯言に耳を傾けてはいけない。



図2 筑波大学第三エリア某所より発掘された博士号搭乗券。T-ACTに採用されていることから、大学当局が博士号を好意的に捉えていたことが窺える。

2 それ池っ！ アヒルが逃げたぞ！

前述の博士号は、ここ数年松見池の隅に係留されていたが、2021年7月1日に突如脱走了。Twitter上には、10:47に最初のものと思われる写真付きTweetが投稿された。その後続々と写真付きTweetが投稿され、人々は博士号の脱走に注目した²。

博士号は、南京錠を用いて強固に係留されており、これは記事[hatena]において説明されている、「ボートを動かせないように固定または施錠することは、公園の一部の土地（池の中の水域）を私人が「占用」することにあたるため、公園管理者（松美池の場合は、筑波大学）の事前の許可が必要になる。」に該当する。つまりは、放置されているように見えて、実際はなんらかのあやしい集団が、公園管理者たる筑波大学から許可を受けて係留しているのである。そして、脱走した博士号はこのあやしい集団の手によって回収されたのである。



図3 左: 脱走した博士号。中央: どこからともなく現れた特殊部隊。右: 閉鎖ネットワーク上に投稿された画像。

回収作業に従事していた特殊部隊によると、あやしい集団は、博士号が脱走した噂を11:30ごろに把握し、12:40に目視で確認したそうだ。同集団は、博士号の脱走により第四学群への隠し通路が人目に晒され、筑波大学の秘密が漏れてしまうことを好ましく思わぬ

²*2当時の様子を知るためにには、Twitterで次に示す語句を用いて検索すれば良い：博士号 since:2021-7-1 until:2021-7-2

かったため、特殊部隊を回収作業に向かわせた。図 3 中央は、WORD 編集部員が捉えた特殊部隊の様子である。彼らはどこからともなく現れ、回収作業後すぐに姿を消した。



図 4 回収の様子 1。左: 脱走した博士号を写真に収める隊員。作業中の表示のある看板と博士号を一枚に収めている。中央: 胴長を着用し松見池に侵入する隊員。右: うんこしょ。どっこいしょ。押してもアヒルは動きません。

博士号のいる松見池にはヘドロが堆積しており、並大抵の装備で侵入することは困難である。そのため、隊員は胴長を装備して作業に当たっていた。図 4 中央は、松見池に侵入し、アヒル目がけて一步ずつ確実に歩んでいく隊員の様子である。

まず隊員は、図 4 右に示すように、博士号を後ろから押して移動させることを試みていた。しかしながら、博士号が滞留していた領域は水深が浅かったためか、この方法ではどんどん支えてしまい、移動させることは困難であるように見受けられた。



図 5 回収の様子 2。左: 博士号を牽引する隊員。中央: 博士号から下船する隊員。右: ロープを用いて博士号を牽引する隊員たち。

博士号前部にはフックが存在するが、図 5 左に示すような、これを人力で引っ張って回収する作業は、堆積したヘドロの山をかき分けて歩く必要があったため困難を極めていた。そのため隊員は博士号に乗り込み、ペダルを漕いで運転しようとしていた。博士号は数年ぶりに波を立てたが、経年劣化のためか前傾しており、その部分が湖底のヘドロに引っかかるようで、ほとんど前進してなかった。図 5 中央は、漕いだ後に下船する隊員を捉えたものである。最終的には岸よりロープが投入され、それを前部フックに取り付けることで回収作業が行われた。図 5 右は、ロープを用いて博士号を牽引する隊員である。



図 6 回収の様子 3。左: 博士号をバックで押し込む隊員。中央: 博士号になにやら細工をする隊員。右: 係留された博士号。

その後博士号は、図 6 左に示すように、バックによって所定の位置に係留された。係留後には、図 6 中央に示すように、特殊部隊が博士号になんらかの細工を施す様子が観察された。しかしながら、編集部員が確認する限りでは、博士号に特段の変化は見られなかつた。このようにして博士号は回収されたのである。

3 終わりに

本記事では、あまり詳細の知られていなかった博士号回収の様子を取り上げた。本記事をきっかけとして博士号に興味を持っていただけたら幸いである。

執筆に際して、WORD 編集部の moneto 氏、public_yusuke 氏より写真の提供をして頂いた。ここに御礼申し上げる。また、本記事中の博士号脱走はノンフィクションであるが、博士号回収に関わった組織、名称などについてはフィクションである。フィクションということにしなければ、大学内に存在するあやしい集団による検閲を通らないのである。

オンラインでソフトウェアイベントを開くときのテクニック集

文 編集部 puripuri2100

はじめに

特定のプログラミング言語やソフトウェアを使用するコミュニティに所属しているであろう皆さんの悩みは何と言っても「このコミュニティを盛り上げたいぜ！！！」だと思います。

多くのコミュニティで行われている手っ取り早く盛り上げるための手法として「ソフトウェアカンファレンス」というものがあります。 \LaTeX ではTUGやTeX Confなどがあり、他のコミュニティでは Haskell Day や RubyKaigi など、様々な例があります。今回はこのソフトウェアイベント、特にオンラインでのそれを行うときのテクニックなどを紹介したいと思います。

オンラインの良さはいくつかありますが、開催地の縛りが無くなり遠方の方が気軽に参加できることがとても大きいです。また、学生という弱い立場の人間がリアル会場を使うことのリスクとコストはかなりありますが、オンラインではそれが薄れます。ただし、リアルでの交流というのも大事ではあるので、そういった面でのバランスは気を付ける必要はあるとは思います。どのようなツールを使うと良いかについては、各方面から知見が蓄積されていくことでしょう。

筆者は SATySFi という Better \LaTeX を目指している新しい組版ソフトウェアのソフトウェアカンファレンスを 2020 年と 2021 年に主催し、2022 年も主催予定です (SATySFi Conf 2022 はまだ参加者募集段階で未開催)。また、講演者・聴衆としていくつかのオンラインイベントに参加しており、知見はある方ではないかと思っています。

では、さっそくテクを見ていきましょう。

テク：関係各所に確認を取りまくる

当たり前ですが、大事です。特に「ソフトウェアの作者を呼びたいぜ！ できれば招待講演もしたいぜ！」みたいなときに相手に確認を取らずに勝手に話を進めると後で大変なことがあります。気を付けましょう。

ただし、確認を取ったのに相手が予定を忘れてしまい、リスケジュールになったり相手が反省して人の気持ちを学びにコミュニティを離れてしまったりする可能性もあります^{*1}。

確認を取る先としては

*1 <https://gigazine.net/news/20181023-linus-torvalds-return-to-linux/>

- ・共同で主催する人
- ・コミュニティの人たち
- ・何らかの特殊なサービスを使う場合はその提供側
- ・もし内定している発表者が居る場合はその発表者

などが考えられます。

テク：参加者募集は connpass を使うと便利

connpass の回し者ではないです。

枠ごとに参加者を募集できたり時間や開催サービスの表示ができたりとかなり便利です。また、様々な技術イベントが connpass で募集を行っているため、そういった意味でもデファクトスタンダードになっているものに乗っておくことに損はないと思います。

テク：Twitter で宣伝しまくる

そのコミュニティの中で使われている Slack や Discord などのコミュニケーションツールで宣伝するのは当然ですが、新規参入を増やすという意味では、コミュニティに所属していない人にも届ける必要があります。Twitter にはリツイート (RT) という素晴らしい機能があるため、「知り合いの知り合い」にまでイベントが開催される旨を伝えられる可能性が高まります。積極的に使っていきましょう。

特に、宣伝ツイートを定期的にセルフ RT したり引用 RT したり、言及してくれたツイートを RT することは大事です。宣伝の効果を高めるためにはかなり必要なことです。

テク：YouTube で同時配信すると良い

Zoom や cluster といったサービスで発表することは多いですが、「少しその分野に興味がある」という薄い気持ちで参加する人にとってソフトウェアイベントのためだけに特定のソフトウェアをインストールしたりアカウント登録をしてもらうというのは大きい障壁になります。ですので、障壁がかなり低い YouTube で配信をすると参加者の裾野を広げることができます。

また、「リアルタイムで参加できなかったけれども後から見たい」という需要もかなり存在するため、アーカイブを残すことをおすすめしておきます。

その際の注意点ですが、必ず発表者には発表を全世界に配信してアーカイブが残ることを伝え、同意を取りましょう（発表者募集の時点での条件として明記しておくと手間が省けます）。極まれに著作権侵害をしているスライド（アニメのスクショを貼るなど）を作成する方がいらっしゃいますので、それを事前にさせないことは重要です。

テク：音声・スライドチェックは十分にやるべき

突然の音声トラブルが発生することもあり、特にオンラインイベントの場合のそれは機材の融通が効かないため、とても致命的です。ですが、開場前と発表前に二重のチェックをすることでトラブルの発見がしやすくなります。

発見したところで解消できなければ意味はありませんが、事前に発見できれば一旦発表順を変更してその間にPCの再起動をしてもらったり、最悪チャットで発表をしてもらったりなどの対応をスムーズに取ることができます。

過去には「スライドに埋め込んだ動画が再生できない」や「音声が出なくなった」、「マイクが雑音を拾いまくる」などのトラブルがありましたが、これらは事前にチェックしておけば発見でき、対処できたかもしれません。

参加者と発表者にストレスと緊張を与えないためにも、念入りにテストしましょう。

テク：休憩・交代の時間を長めにとると良い

前述の音声・スライドチェックをするための時間として10分程度設けておくと良いです。また、交代時間を十分に設けておくことで前の発表が若干延びてしまったとしても休憩時間中に十分吸収でき、後ろの発表の人に迷惑をかけることが無くなります。

テク：参加者アンケートを取ると参考になる

多様な視点から見た意見を集約して参考にすることで、より良い運営に繋げていくことができます。特に発表枠や運営手法に関する意見は役に立ちます。もちろん、全ての意見を実現することはできませんし、確固たる思想に基づいて実施していることを変にゆがめる事はしない方が良いですが。

テク：発表者にスライドの公開をさせる

知見をインターネット上に蓄積していくことは大事です。スライドを公開してもらいましょう。使用するサービスはなんでも良いとは思いますが、最近SlideShareの制約が強くなったりと色々あるので、選定には注意を払うべきです。

テク：イベント終了後に振り返り記事を書くと良い

イベントに関する情報が散らばりすぎていると後から調べたくなった人が困ります。ですので、

- 発表者の名前
- 発表のタイトル
- 発表の概要
- 発表に使われたスライドへのリンク
- 発表の様子の動画へのリンク

などの情報をまとめたブログなどを作成することをお勧めします。

自分は以下のようにまとめた記事を出しています。

- SATySFi Conf 2020 まとめ：<https://puripuri2100.hatenablog.com/entry/2020/07/26/164203>
- SATySFi Conf 2021 まとめ：<https://puripuri2100.hatenablog.com/entry/>

2022/01/20/205344

テク：定期的に開催する

とてつもなく大事です。

間隔がひと月になるか半年になるか一年になるかは主催者の気合とコミュニティの規模に左右されますが、定期的に開催することで良い刺激剤となり、コミュニティの活性化に大きく役立ちます。

テク：開催を恐れない

「イベントをやっても誰も参加してくれないんじゃないだろうか」という恐れは最初にやろうとするときには必ず抱くものですが、そこで諦めずに一歩を踏み出してみてください。意外とみんな気軽に参加してくれると思います。

これは参加・発表側にも言え、コミュニティを活性化させたいときには気軽にイベントに参加していきましょう。また、発表したいときも「こんな雑な成果で発表したら叩かれるんじゃないかな」と躊躇うのではなく、思い切って発表してみてください。

意外とみんな優しいです。

おわりに

SATySFi Conf 2022 にぜひ参加してください！！！！！！！

<https://connpass.com/event/247712>

プリティーシリーズ視聴 RTA（プリパラ～プリ☆チャン， 映画分岐全制覇， 敗北 ED カット） 29d22h36m08s

文 編集部 n.takana

まあ倫理的に考えて真矢クロの相部屋は『マズい』よな…（千葉県北西部の方言で皆さんお元気ですかの意）

春休み中にプリパラ～アイドルタイムプリパラ～キラッとプリ☆チャンまで一気に観たので第3エリアにお住まいの女児の皆様にも履修方法を共有しようと思います。君もこの記事を読んでプリパラとプリ☆チャンとプリマジの履修、しよう！（ダイマ）

1 レギュレーションの策定

なんか既に走っている人も居そうですが（そうか？）気にせず新規にレギュレーションを作成します。筆者は既にプリマジを視聴してたのでプリパラ（全141話）→アイドルタイムプリパラ（全51話）→キラッとプリ☆チャン（全153話）で走ることにしました。

え？ プリティーリズム？ なんのこったよ（すっとぼけ）

プリティーリズムシリーズは大学への準備とかあって春休み中（というか3月の1ヶ月）で走らなくてはならなかったので諦めました（自分の予定でレギュレーションを決める走者の肩）。プリティーリズム履修済み女児兄貴は僕にダイマしてください。

本レギュでは映画ちゃんと観ます。対象範囲では3本あるので見逃さずに行きましょう。映画は女児先輩が観るからか1本あたりの上映時間が1時間程度と短いです（悲しい）。

しかし、驚くべきことにプリパラの映画では上映する週によって話が分岐していて、3パターンにコースが別れています。もちろんこれは全部制覇します。というか全部良いので観てくれ。dアニメストアはオタクに優しいのでストーリーが分岐しているところで前後半を分けてくれていますのでちょっとだけタイムを短縮できます（申し訳程度のRTA要素）。

その他レギュとして再生速度は等倍、OPカットEDカットは無しで行きます。RTAとはいえる今回の主目的はプリパラとプリ☆チャンをめいいっぱい楽しむことです。タイムよりも大切なものがそこにあるのです（RTAとは？）。

ただし、本記事のタイトルにもある通り敗北EDは飛ばしても良いことにします。敗北EDとはEDに実写映像が入る恐怖現象のこと、アイドルが踊るものから本物の女児先輩が踊って投稿するもの、クオリティが非常に微妙な着ぐるみが踊るものなど多岐に渡ります。怖いねえ…¹ 異常成人男性であるところの我々走者はかわいいアニメキャラの女の子が見たいのであって、EDでいきなり成人女性や遊園地のイベントでしか出てこないような不気味な着ぐるみが踊ってるところを見せつけられるのは非常にダメージがデカいです。

*1現在放送中の『ワッチャ！ プリマジ』でもつい最近（5月8日放送分）敗北EDが来てしまいました…（絶望）

プリティーシリーズ観聴 RTA (プリパラ～プリ☆チャン, 映画分岐全制覇, 敗北 ED

カット) 29d22h36m08s

ということで今回は敗北 ED はカットです (無情). 流石に各 ED1 回は観ました^{*2} が…

2 チャートの作成

最低でも 1 ヶ月以内に観聴しなくてはならない^{*3} のでチャートは真剣に組みます. 紹密な計算によって全部で何話あるか計算すると,

$$140 + 51 + 153 = 344$$

なんと 344 話あるそうです. どうやら 1 ヶ月で観られそうですね！

3 月は 31 日まであるので 1 日あたりの話数を工夫して計算すると,

$$344 \div 31 = 11.0967 \dots$$

余裕ですか (慢心). 卒業式など避けられないイベントがあって観られない日があったり映画を観たりする時間も考えると 1 日あたり 12 話くらいが妥当そうです.

観聴順は表 1 の通りにしました.

表 1 観聴順

観聴順	タイトル	話数 or 時間
1	プリパラ	38 話
2	プリパラ 2nd season	51 話
3	プリパラ 3rd season	51 話
4	アイドルタイムプリパラ	51 話
5	映画プリパラ み～んなのあこがれ♪レッツゴー☆プリパリ	60 分
6	劇場版プリパラ み～んなでかがやけ！ キラリン☆スターライブ！	57 分
7	キラッとプリ☆チャン	153 話
8	劇場版 プリパラ&キラッとプリ☆チャン ～きらきらメモリアルライブ～	60 分

ネタバレが嫌で Wikipedia を見ずに組んだので公開順とか全然考慮できてないです… (は？) これから走る人はプリパラ → プリパラ映画 2 本 → アイドルタイムプリパラ → プリ☆チャンを 12 話くらいまで観る → プリパラ&プリ☆チャンの映画を観る → 残りのプリ☆チャンを観る → プリマジを最新話まで観る → アイドルランドプリパラを観るのチャートが良いと思います. 忘れないようにチャートにちゃんと書いておきましょう.

基本的にプリパラとプリ☆チャン (とプリマジ) は話的には独立しているのでどちら観ても良いんですが、前作のネタが入ってたり歴代のキャラがシレッと出てきたりするのでやっぱり時系列順に観るのが一番だと思います. どれか 1 つだけと言うなら一番取つきやすいのはプリ☆チャンですかね. ちなみに、観聴後に凄まじいロス^{*4} が発生するので映画を後に取っておくというチャートも全然アリです.

*2薄目で.

*3さもなくば引っ越しの準備などができず全部がおしまいになってしまふ.

*4時間的な意味ではなく心的な意味で.

3 視聴のための準備

d アニメストアに入れ！（メイドインワリオ）*5

4 大履修

あとは普通にアニメを観るだけです。ネタバレになってしまふあれだし経過だけ書きます（は？）。

表2 履修タイムライン

タイトル	日付	話数	経過
プリパラ	3/1	1話	履修スタート
	3/3	25話	25話を見聴し，“涙”…
	3/4	39話	2nd season に突入
	3/5	45話	ありよま～～～！（発作）
	3/6	52話	実写 ED の連鎖から開放。ここで敗北 ED は終わりだと思ってました（感慨）
	3/8	78話	あろみかだけが本当の“愛”であることに気づく
	3/8	90話	3rd season に突入
	3/8	91話	らあらは私の母になってくれるかもしれない女性だ！
	3/9	97話	（絶句）
	3/9	105話	今サムネを見るだけでも泣けます
アイドルタイムプリパラ	3/12	137話	決勝ライブがガチで良かった。オタクは今すぐ観た方がいいよ。
	3/12	140話	プリパラを完走…
	-	-	アイドルタイムプリパラ
プリパラ映画 2 本	3/12	1話	プリパラと話が繋がってるのでそのまま観始める
	3/15	41話	41話でいきなりクソデカい百合が生えてきてビビる
	3/15	51話	完走
プリパラ映画 2 本	3/16	-	女児向けだからか展開がコンパクトにまとまってギャグに全振り するのが良かった（女児並感）
キラッとプリ☆チャン	3/16	1話	プリ☆チャン視聴開始
	3/16	9話	なんかあからさまに百合じゃないこれえ？（気づき）
	3/17	-	関東が停電し、まさかの大ロス
	3/21	50話	泣いた。いや、啼いた。
	3/23	61話	一瞬自分の願望が生み出した幻覚かと思った。確認のために観返してロス。
	3/24	78話	新 OP/ED が麗しすぎて発狂
	3/24	88話	期待してた展開がモロに来て確認のために観返してロス
	3/25	94話	<i>What the customer really needed.</i>
	3/26	111話	分かっててやってんだろこれエ！！！（ED の映像を指差しながら）
	3/26	112話	院試の存在を思い出して偽物の鬱病になった
劇場版 プリパラ&プリ☆チャン	3/28	132話	制作陣（理解のある彼）
	3/30	153話	完走
劇場版 プリパラ&プリ☆チャン	3/30	-	タイマーストップ

自分でもビックリするくらいコンスタントに視聴できました。

*5 ちなみに d アニメストアは日割りしてくれないので月の頭に入るとオトクらしいです。この RTA を長期休暇に走るのにピッタリですね…（チラチラと読者の方を見ながら）

プリティーシリーズ観聴 RTA（プリパラ～プリ☆チャン， 映画分岐全制覇， 敗北 ED カット）29d22h36m08s

記録は 29 日と 22 時間 36 分 08 秒でした。まあ厳密に計ったわけじゃなくてツイートから逆算しただけだけど照*6。

前半の方が勢いが良くて後半の方が若干鈍いのはつくばに引っ越すために色々準備をしてて時間が取れなかったからです。家具を買ったり女の子しか出てこない漫画や雑誌をダンボールにギチギチに詰める作業などをしたら時間が溶けました。なので記録更新の余地は充分にあると思います。記録更新の余地は充分にあると思います。

5 完走した感想

え？ 完走した完走ですか？

マジで最高だった。

普通に女児向けであることをすっかり忘れて楽しめました。こんなに面白いアニメを 7 年間毎週逃してたのかと思うと非常に惜しい気持ちでいっぱいになります。女児向けのアニメってなんとなく説教臭かったり道徳的にするためにストーリーの詰めが甘かったりするイメージがあったんですが、全然そんなことはなく*7 純粋にキャラや展開が優れていて面白かったです。女児向けアニメなんて初めて観たけど良いなーとか思ってたんですが、よくよく考えると中学時代にカードキャプターさくらを観てマジでデカい声を出していたのであんまり変わってねえわ照*8。

あとやっぱり特筆すべきは曲ですね。プリパラもプリ☆チャンも醍醐味はやはりライブで、豊富な衣装と優れた楽曲が毎回楽しめるのは大変優れています。同じ曲でもストーリーが進むとまた違った印象を受けるので全然飽きません。Spotify でライブ曲と公式リミックスアルバム（！）が配信されているので、これらを聴きまくることで無限に狂うことができます。オトク！

全体的な総括ですが、1 ヶ月に渡って集中して観られたので話も良くつかめたと思います。深夜アニメとかでもそうですが、1 週間空くと忘れちゃうんですよね。普段の自分なら気づかなそうな細かいネタまで拾えたのは良かったです。特に長いアニメは整理するのが大変なので一気に集中して観たのは正解だったと思います。

しかし、ちょっとしたデメリットもありました。それは非常に深刻なプリパラロス&プリ☆チャンロスです。1 ヶ月間毎日観ていたものを急に失うと人間は壊れます。筆者は RTA が終わった 4 月に入ってからも毎日プリパラやプリ☆チャンを観聴し直したり関連動画を漁ったり週に 1 話だけ供給されるプリマジを観て心を癒やしておりました。そもそも 7 年かけて放送されていたものを十ヶ月で観ようとするのは非常に危険です。
~~過剰摂取ダメ、ゼッタイ。男なら自己破壊を。ロスがあるなら再走すれば良いのです。君も~~
カメラロールがラーメンと女児向けアニメのキャプで埋まってる異常独身男性になろう！

*6 RTA なのにまとまに計測さえしてないのか…（呆れ）

*7 登場人物が（というかドロシーが）視聴者の思っていることをド直球で代弁してくれることで上手いこと折り合いをつけているのは見事だと思う。

*8 古典的名作であるカードキャプターさくらの全話観聴もお待ちしてます。

6 終わりに

この節を読んでいる頃には皆さんチャートをとっくに作り終わってプリパラの3話を観ている途中辺りでしょうがいかがでしたでしょうか。本稿を通して1つのことに打ち込むことの素晴らしさ、ひたむきに何かを積み重ねていくことの美しさを読者女児に伝えられたなら幸いです（建前）オタクはプリパラとプリ☆チャンとプリマジを観て♡（本音）

クソ長いテキストから超高速かつ超省メモリに部分文字列の出現回数を数えたい

文 編集部 塚本

1 はじめに

はじめまして、情報科学類3年の塚本です。突然ですが、「クソ長いテキストから超高速かつ超省メモリに部分文字列の出現回数を数えたいなあ・・・」と思った事はありませんか？僕はあります。この記事では、クソ長いテキストから超高速かつ超省メモリに部分文字列の出現回数を数える方法を皆さんにお伝えしたいと思います。

2 問題の具体化

抽象的な問題を解こうとしてもどうにもならないので、まずは今回の要求を厳密に定義していきます。今回の問題で扱う「テキスト」は「**文字列**」の一つであり、厳密には「**文字列集合**」の一要素です。「文字列」とは、0個以上の「**文字**」からなる列の事です。それぞれの「文字」は文字集合 Σ の要素となります。ここで、文字集合のサイズを $\sigma = |\Sigma|$ とします。全ての「文字」の集合 Σ を定義した上で、全ての「文字列」の集合も Σ^* のように定義することができます。

次に「超高速かつ超省メモリに部分文字列を検索したいなあ・・・」という発言の意味について考えてみます。まず、今後の分かりやすさのために、「検索元となっている長い文字列」を「**テキスト文字列（テキスト）**」と呼び、「検索対象となっている短い文字列」を「**パターン文字列（パターン）**」と呼ぶ事にします。テキスト文字列 $T \in \Sigma^*$ に対して「パターン P がテキスト T の部分文字列である」とは、 $\exists i \in \{1, 2, \dots, |T| - |P| + 1\}, T_i, T_{i+1}, \dots, T_{i+|P|-1} = P$ が成り立つ事を意味します。具体例を挙げると、文字列 $P = "ab"$ は文字列 $T = "ababc"$ の部分文字列であり、 i が 1, 3 の時に上述の条件を満たします。「部分文字列を検索する」とは、テキスト T とパターン P を入力に、 P が T の部分文字列であるかどうか判定を行い、もしそうであれば対応する i の一覧を出力する事を指します。通常の RAM モデルでは1回の操作で複数の文字にアクセスする事ができないため、この検索操作の理論上最適な計算量は $O(\min(|P|, |S|))$ 時間となります^{*1}。

しかし、長さ 10^9 のテキストから長さ 10 のパターンを検索する時に毎回 10^9 回のメモリアクセスを行っているとクソ遅いため、テキストに対する $O(n \text{ polylog } n)$ 時間^{*2} の前処理を許して検索時にかかる計算量を $O(m \text{ polylog } n)$ に抑える事を考えます^{*3}。このような操

*1 計算モデルが異なる場合はそうとは限りません

*2 polylog n は n の polylogarithmic function を表します。有限長の正数列 A が存在し、ある定数 n_0 以上の全ての n に対して $f(n) \leq \sum_{k=1}^{|A|} A_k \log^k n$ が成り立つ時、 $f(n) \in O(\text{polylog } n)$ です

*3 ここで n はテキスト長を、 m はパターン長を表しています

作が行えるデータ構造の事を便宜上 **全文索引** と呼ぶことにします。

全文索引にもいくつか種類がありますが、今回はその中でも特に「超高速かつ超省メモリ」な物を探したいです。テキスト長に依存しない + パターン長に対して線形時間で部分文字列が検索できればなんとなく「超高速」っぽいのですが、テキスト長の 10 倍のメモリ使用量を使う索引は「超省メモリ」とは言えない気がします。そこで、今回は「テキスト長とほぼ同じメモリ使用量の索引」を「超省メモリな索引」と定義して、超省メモリな索引とは何かを考えていきたいと思います。ここでの「テキスト長とほぼ同じメモリ使用量」とは、データ構造を表現するのに必要な空間計算量が文字列長 n に対して $n \log \sigma + o(n \log \sigma)$ bits^{*4} の範囲内である事を指します。この条件を満たす索引は、「あるデータを表現する時の情報理論的下限が t bits である時に $t + o(t)$ bits で表現できるデータ構造」となります。このような条件を満たすデータ構造を **簡潔データ構造** と呼びます。

ここまで整理する事で、「クソ長いテキストから超高速かつ超省メモリ部分文字列を検索したいなあ・・・」という願望は、高速かつ簡潔な^{*5} 全文索引について学ぶ事で解決できるかもしれません事が分かりました。それでは、ここまで考えてきた事を問題として定式化してみましょう。

3 導入

この章では今後使う定義や表記・問題設定等をまとめて導入していきます。次の章以降で出てきた表記の意味が分からぬ場合は適宜ここに戻って確認してください。

3.1 基本的な定義

数列 A は $(A_1, A_2, \dots, A_{|A|})$ のように表記します。 A_i は数列 A の i 要素目を表します。文字の全順序集合は Σ と表記し、文字集合のサイズを $\sigma = |\Sigma|$ と表記します。また、 Σ 中で c より小さい文字の集合を $\Sigma_{<c}$ と表記します。0 個以上の文字からなる列 $S \in \Sigma^*$ を **文字列** と呼びます。一般の列は $A = (A_1, A_2, \dots, A_{|A|})$ のように表記しますが、文字列の場合は特別に $S = S_1 S_2 \dots S_{|S|}$ と表記します。

文字列 S に対して、 $S_l S_{l+1} \dots S_{r-1} S_r$ ($1 \leq l \leq r \leq |S|$) を S の **部分文字列 (substring)** と呼び、 $S[l, r]$ で表します。特に $l = 1$ の時は S の **接頭辞 (prefix)** と呼ばれ、 $r = |S|$ の時は S の **接尾辞 (suffix)** と呼ばれます。また、任意の文字列の組 S, T に対して以下のような全順序関係 $<$ を定義し、この全順序を **辞書順 (lexicographical order)** と呼びます。

$$S < T \Leftrightarrow \exists i \in \{1, 2, \dots, |T|\}, (\forall 1 \leq j < i, S_j = T_j) \wedge ((S_i < T_i) \vee (|S| < i)) \quad (1)$$

文字列の組 T, P に対して $\text{match}(T, P)$ を以下のように定義します。

$$\text{match}(T, P) = \{1 \leq i \leq |T| - |P| + 1 \mid T_i T_{i+1} \dots T_{i+|P|-1} = P\} \quad (2)$$

^{*4} $o(f(n))$ は感覚的には「 $f(n)$ より厳密に小さい計算量」を意味しています。具体的には、ある定数 $c > 0$ が存在し、ある定数 n_0 以上の全ての n に対して $f(n) \leq c g(n)$ が成り立つ時、 $f(n) \in O(g(n))$ です

^{*5}簡潔データ構造であるような、という意味です

3.2 問題の定義

今回の目標は以下の問題として定式化する事ができます。

問題 1 長さ n の文字列 T に対するデータ構造 $\mathcal{D}(T)$ であり、以下の要件を満たす物を構築せよ。

1. T が与えられた時、 $O(n \text{ polylog } n)$ 時間で $\mathcal{D}(T)$ を構築できる
2. 長さ m の文字列 P と $\mathcal{D}(T)$ が与えられた時、 $|\text{match}(T, P)|$ を $O(m \text{ polylog } n)$ 時間で計算できる
3. $\mathcal{D}(T)$ を空間計算量 $n \log \sigma + o(n \log \sigma)$ bits で表現できる

ただし、ここで今後の説明の分かりやすさのために、 Σ 中で順序が最小の特殊文字 $\$ \in \Sigma$ を定義し、問題の入力に以下の制約を追加します。

- $T[1, |T| - 1], P$ の中には文字 $\$ \in \Sigma$ が出現しない
- $T_{|T|} = \$$ である

この制約が存在しないバージョンの問題も n と σ をそれぞれ 1だけ増やす事で制約付きの問題に変換する事ができるため、ただただ今後の説明で冗長な場合分けを行わなくて済むようにするためにある制約だと思ってください。

4 FM-Index の関連知識

4.1 FM-Index の概要

問題定義と基本的な定義の導入が済んだので、今回の記事で扱うデータ構造は何なのかについて説明します。今回の記事で扱うデータ構造は **FM-Index** です。このデータ構造は、前述の問題中の \mathcal{D} として要求される要件を全て満たしています。具体的には、

1. $O(n \lceil \log \sigma \rceil)$ 時間で構築可能
2. $|\text{match}(T, P)|$ が $O(m \lceil \log \sigma \rceil)$ 時間で計算可能
3. $n \lceil \log \sigma \rceil + o(n \lceil \log \sigma \rceil)$ bits で表現可能

であり、前述の問題中の要件と照らし合わせて見ると、問題中で求められている内容が全て満たされている事が分かると思います。

では、これから「FM-Index とは一体なんなのか」「FM-Index がどのように高速な検索を実現しているか」を説明していきましょう。FM-Index とは、「テキスト文字列に Burrows-Wheeler Transform をかけて作った文字列を Wavelet Tree や Wavelet Matrix を用いて保存したデータ構造」であり、「LF-mapping の性質と文字列上の rank/select 演算を用いて」高速な検索を実現しています。知らない単語が突然たくさんてきてびっくりしましたね。それでは、上の文章を理解するために必要な知識をこれから身に付けていきましょう。

4.2 rank/select と索引構造

数列・文字列 A に対して、 $\text{rank}_c(A, i)$, $\text{select}_c(A, x)$ を以下のように定義します。

$$\text{rank}_c(A, i) = |\{1 \leq j \leq i \mid A_j = c\}| \quad (3)$$

$$\text{select}_c(A, k) = i \Leftrightarrow (A_i = c) \wedge (\text{rank}_c(A, i) = k) \quad (4)$$

$\text{rank}_c(A, i)$ は A の先頭 i 要素における c の出現回数を、 $\text{select}_c(A, k)$ は A 中での c の k 回目の出現位置を表します。

ではこの操作がどのような時間・空間計算量で実現できるのかについて説明します。本来ならこの節で紹介するデータ構造の詳細についても説明するべきなのですが、全て網羅的に紹介しようとすると紙面がどれだけあっても足りなくなってしまうため、「この操作がこの計算量でできるデータ構造があるんだなあ」という感じで理解して頂けると嬉しいです。

まずビット列 B 上の rank/select 操作ですが、これは空間計算量 $|A| + o(|A|)$ bits の簡潔データ構造（完備辞書）を用いて定数時間で実現する事ができます。また、値域が $[0, m - 1]$ であるような一般の整数列 A 上の rank/select 操作は **Wavelet Tree** や **Wavelet Matrix** と呼ばれるデータ構造を用いると空間 $|A|\lceil\log m\rceil + o(|A|\lceil\log m\rceil)$ bits かつ $O(\log \sigma)$ 時間で実現する事ができます。Wavelet Tree や Wavelet Matrix を用いると、 $\sum_{j=0}^{x-1} \text{rank}_j(A, |A|)$ なども $O(\log m)$ 時間で求める事ができます。これは「 A 中での x より小さい数の出現回数」を表しています。文字列は値域が $[0, \sigma - 1]$ の整数列として扱う事ができるため、文字列 S 上の前述の操作も $|S|\lceil\log \sigma\rceil + o(|S|\lceil\log \sigma\rceil)$ bits *6 の索引構造で実現できます。

4.3 Suffix Array (SA)

長さ n の文字列 S に対して、 n 要素の数列 SA^S を以下の条件を満たす順列として定義します。

$$\forall (i, j) \in \{1, 2, \dots, n\}^2, i < j \Leftrightarrow S[\text{SA}_i^S, n] < S[\text{SA}_j^S, n] \quad (5)$$

SA^S は S の全ての接尾辞を辞書順で並べ替えた時の順序を表します。この数列を S の **接尾辞配列 (Suffix Array)** と呼びます。

Suffix Array 自体はただの数列ですが、 S と SA^S を合わせて保存するとそれだけで全文索引となります。これは一体どういう事なのでしょうか？ テキスト文字列 T 中でのパターン文字列 P の出現位置を求める手続きを下に示します。

1. 二分探索を用いて $P \leq T[\text{SA}_l^T, |T|]$ となる最小の $1 \leq l \leq |T|$ を求める
2. 二分探索を用いて $P < T[\text{SA}_r^T, |T|]$ となる最小の $1 \leq r \leq |T|$ を求める
3. そのような l, r が存在しない場合空集合を出力する
4. $l < r$ である場合 $\{l, l + 1, \dots, r - 2, r - 1\}$ を出力する

この手続きの計算量と正当性について考えてみましょう。まず二分探索の各ステップの

*6 $|S|\lceil\log \sigma\rceil \leq |S|(\log \sigma + 1)$ である事に気を付けると、これは $|S|\log \sigma + o(|S|\log \sigma)$ bits とも書く事ができます

計算量ですが、文字列 P の文字数以上の比較は行われないので $O(|P|)$ となります。また、二分探索のイテレーションの回数ですが、 l, r の取りうる値域が 1 から $|T|$ までの整数であり、各イテレーションの度に探索範囲が半分となるため、イテレーション回数は $O(\log_2 |T|) = O(\log |T|)$ となります。従って、全体の計算量は $O(|P| \log |T|)$ となり、全文索引として要求される要件を満たしています。次に正当性についてです。SA の定義より、 $i < j$ ならば $T[\text{SA}_i^T, |T|] < T[\text{SA}_j^T, |T|]$ となります。従って $T[\text{SA}_i^T, |T|]$ は i に対して単調であり、二分探索で求めた区間に對応する全ての接尾辞について、その先頭 $|P|$ 文字が P と一致する事が分かります。これにより、アルゴリズムの正当性についても理解する事ができました。

文字列 T から SA^T を $O(|T| \text{ polylog } |T|)$ 時間で求める方法は自明ではありませんが、実は SA^T は $O(|T|)$ 時間かつ実用上高速に求める事ができる事が知られています^{*7}。従って、構築・検索計算量・検索正当性の全てについて全文索引として要求される要件を満たしている事がわかり、組 (T, SA^T) が全文索引な事が分かりました。

5 Burrows-Wheeler Transform (BWT)

5.1 BWT の概要

Burrows-Wheeler Transform (BWT) は文字列に対する可逆変換です。長さ n の文字列 S に対して BWT を適用した後の文字列 (**BWT 配列**) $\text{BWT}^S = \text{BWT}_1^S \text{BWT}_2^S \dots \text{BWT}_n^S$ は、数列 $\text{BWIdx}^S = (\text{BWIdx}_1^S, \text{BWIdx}_2^S, \dots, \text{BWIdx}_n^S)$ を用いて以下のように定義されます。

$$\text{BWIdx}_i^S = \begin{cases} n & (\text{SA}_i^S = 1) \\ \text{SA}_i^S - 1 & (\text{otherwise}) \end{cases} \quad (6)$$

$$\text{BWT}_i^S = S_{\text{BWIdx}_i^S} \quad (7)$$

BWT 配列は、「接尾辞を Suffix Array の順に並べた時の、各接尾辞の直前に現れる文字」を並べた列です。

Burrows-Wheeler Transform は可逆変換であるため、 BWT^S の情報のみから元の文字列 S を復元することができます。また、この配列は SA と S から $O(n)$ 時間で計算できるほか、接尾辞配列を経由せずに直接構築する方法も知られています。

5.2 LF-mapping の定義

Burrows-Wheeler Transform にはいくつかの良い性質があります。その一つが **LF-mapping** です。LF-mapping を用いる事で元文字列の高速な復元などを行う事ができ、FM-Index でも LF-mapping の性質を利用して検索を行っています。

LF-mapping では、「 i が与えられた時、 $\text{SA}_i^S - 1$ が SA^S 上でどの index に存在するか」を BWT のみから求める事ができます。 S の各文字を辞書順に並べ替えた文字列を F として S の BWT 配列を L とした時にこの操作が L 上の index を F 上の index に変換する操作に対

^{*7}SA-IS というアルゴリズムを用いる事が多いです

応しているため、LF-mapping という名前がついています。

これからは LF-mapping を与える関数 LF を導出してみたいと思います。まずは上で説明した LF-mapping の定義を問題として落とし込んでいきます。

問題 2 長さ n の文字列 S に対する関数関数 $\text{LF}^S : \{1, 2, \dots, n\} \rightarrow \{1, 2, \dots, n\}$ であり、任意の $1 \leq i \leq n$ に対して $\text{SA}_{\text{LF}^S(i)}^S = \text{SA}_i^S - 1$ を満たす関数 LF^S を構築せよ。ただし、計算時には BWT^S 以外の情報を用いる事ができないものとする。

これにより解くべき問題が定まったので、次の節でこの要件を満たす関数 LF^S について考えてていきましょう。

5.3 LF-mapping の導出

$\text{BWidth}_i^S \neq 0$ の時、 $S[\text{BWidth}_i^S, n] = \text{BWT}_i^S S[\text{SA}_i^S, n]$ は「 S 中で i 番目に小さい接尾辞より一文字だけ長い接尾辞」を表します。ここで、 SA^S が接尾辞の順序を表している事を考えると、以下の式が成り立つ事が分かります。

$$\forall (i, j) \in \{1, 2, \dots, n\}^2, \forall (p, q) \in \Sigma^2, pS[\text{SA}_i^S, n] < qS[\text{SA}_j^S, n] \Leftrightarrow (p, i) < (q, j) \quad (8)$$

上の式中の p, q を $\text{BWT}_i^S, \text{BWT}_j^S$ に置き換えて考えると、以下の式も同様に成り立つ事が分かります。

$$\forall (i, j) \in \{1, 2, \dots, n\}^2, S[\text{BWidth}_i^S, n] < S[\text{BWidth}_j^S, n] \Leftrightarrow (\text{BWT}_i^S, i) < (\text{BWT}_j^S, j) \quad (9)$$

上の式は「1つ長い接尾辞同士の順序関係が組 (BWT_i^S, i) 間の順序関係と一致する」事を意味しています。 $\text{SA}_i^S = n$ の時に $\text{BWT}_i^S = \$$ となる事を考えると、これは全ての (i, j) の組に対して成り立つ事が分かります。つまり、「 i が与えられた時、 $\text{SA}_i^S - 1$ が SA^S 上でどの index に存在するか」を返す問題は、「 i が与えられた時、 $(\text{BWT}_j^S, j) \leq (\text{BWT}_i^S, i)$ を満たす j はいくつ存在するか」を返す問題に変換することができます。

SA^S 上の問題が (BWT_i^S, i) 間の問題に帰着できたところで、帰着後の問題をどうやって解くかを考えてみましょう。 $(\text{BWT}_j^S, j) \leq (\text{BWT}_i^S, i)$ を満たす j は必ず以下の条件のうちどちらか片方を満たし、条件を満たさないなら $(\text{BWT}_j^S, j) > (\text{BWT}_i^S, i)$ となります。

i	SA_i^S	BWT_i^S	$S[\text{SA}_i^S, n]$
1	6	b	\$
2	4	c	ab\$
3	1	\$	abcab\$
4	5	a	b\$
5	2	a	bcab\$
6	3	b	cab\$

表 1 $S = \text{abcab\$}$ に対する $\text{SA}^S, \text{BWT}^S$ の例

1. $\text{BWT}_j^S < \text{BWT}_i^S$
2. $\text{BWT}_j^S = \text{BWT}_i^S$ かつ $j \leq i$

前者の条件は、 BWT 配列上での $c \in \text{BWT}_i^S$ であるような c の出現回数を表しており、これは BWT 配列を走査すれば簡単に求まりますし、rank演算を用いて $\sum_{c \in \Sigma_{<\text{BWT}_i^S}} \text{rank}_c(\text{BWT}_S, n)$ と書く事もできます。一方、後者の条件は BWT 配列の先頭 i 要素における BWT_i^S の出現回数を表しており、これも同様に BWT 配列の走査によって求まるほか、 $\text{rank}_{\text{BWT}_i^S}(\text{BWT}_S, i)$ と表現する事もできます。したがって、関数 LF は以下の式のように書く事ができ、問題中の要件を全て満たす関数を作る事ができました。

$$\text{LF}^S(i) = \sum_{c \in \Sigma_{<\text{BWT}_i^S}} \text{rank}_c(\text{BWT}_S, n) + \text{rank}_{\text{BWT}_i^S}(\text{BWT}_S, i) \quad (10)$$

6 FM-Index

6.1 データ構造の概要

クソ長い導入がやっと終ったので、ここから本題の FM-Index に入っていきます。再掲となります、FM-Index とは「テキスト文字列に Burrows-Wheeler Transform をかけて作った文字列を Wavelet Tree や Wavelet Matrix を用いて保存したデータ構造」であり、「LF-mapping の性質と文字列上の rank/select 演算を用いて」高速な検索を実現しています。

長さ n のテキスト文字列 T が与えられた時、 BWT^T を Wavelet Tree や Wavelet Matrix で保存した物が T の FM-Index となります^{*8}。Wavelet Tree や Wavelet Matrix の時間・空間計算量を考えると、FM-Index は $O(n \log \sigma)$ 時間で構築でき、空間計算量は $n[\log \sigma] + o(n[\log \sigma])$ bits である事が分かります。

6.2 文字列検索の手続きと正当性

長さ n のテキスト文字列 T と長さ m のパターン文字列 P が与えられた時、 $\text{match}(T, P)$ は必ず SA 上のある区間に對応します。FM-Index では以下の手順で $\text{match}(T, P)$ に対応する SA 上の区間 $[l, r]$ を求めます。

1. $l = 1, r = n$ とする
2. $i = m$ とし、 $i = 0$ となるまで以下の操作を行う
 - (a) $l = \sum_{c \in \Sigma_{< P_i}} \text{rank}_c(\text{BWT}_T, n) + \text{rank}_{P_i}(\text{BWT}_T, l - 1) + 1$ とする
 - (b) $r = \sum_{c \in \Sigma_{< P_i}} \text{rank}_c(\text{BWT}_T, n) + \text{rank}_{P_i}(\text{BWT}_T, r)$ とする
 - (c) $l > r$ であるなら空集合を返す
 - (d) $i \leftarrow i - 1$ とする
3. 区間 $[l, r]$ を返す

t 回目のイテレーションが終った後の区間 $[l, r]$ は、 $\text{match}(S, P[m - t + 1, m])$ に対応

*8各 $c \in \Sigma$ に対して $\sum_{c' \in \Sigma_{< c}} \text{rank}_{c'}(\text{BWT}^S, n)$ を別に保持している資料が多いですが、この値は Wavelet Tree 等を用いて計算可能であるため、このような定義をしています

するような SA 上の区間を表しています。各イテレーションで区間 $[l, r]$ を徐々に狭めていきながら検索を行う事がこのアルゴリズムの特徴です。

このアルゴリズムの正当性について考えてみましょう。 $\text{match}(S, P[m-t+1, m])$ に対応する SA 上の区間 $[l, r]$ に対して上の手続きを 1 イテレーション行った後の区間を $[l', r']$ とします。この時の区間 $[l', r']$ が $\text{match}(S, P[1, t+1])$ に対応するような SA 上の区間となる事が証明できれば、帰納法によりアルゴリズムの正当性を示す事ができます。

まずは l' について考えます。 l' は $\text{BWT}_i^S S[\text{SA}_i^S, n]$ の先頭 $t+1$ 文字が辞書順で $P[m-t, m]$ 未満であるような i の個数 +1 を表していて欲しいです^{*9}。ここで、LF-mapping の時の議論と式 8, 9 を思い出すと、そのような i は必ず以下のどちらかの条件を満たす事が言えます。

1. $\text{BWT}_i^S < P_{m-t}$
2. $\text{BWT}_i^S = P_{m-t}$ かつ $i < l$

1 つ目の条件を満たす物の個数は $\sum_{c \in \Sigma_{< P_{m-t}}} \text{rank}_c(\text{BWT}_T, n)$ と書く事ができますし、2 つ目の条件を満たす物の個数は $\text{rank}_{P_{m-t}}(\text{BWT}_T, l-1)$ と書く事ができます。したがって、上の手続きで求めた l' は区間の始点に正しく対応している事が言えました。

r' の正当性についてですが、これも同様の議論で示す事ができます。 r' は $\text{BWT}_i^S S[\text{SA}_i^S, n]$ の先頭 $t+1$ 文字が辞書順で $P[m-t, m]$ 以下であるような i の個数を表していて欲しいです。これも l' と同様に 2 つの条件で書く事ができ、それぞれの条件が $\sum_{c \in \Sigma_{< P_i}} \text{rank}_c(\text{BWT}_T, n)$ と $\text{rank}_{P_i}(\text{BWT}_T, r)$ に対応しているため、 r' は区間が終点に正しく対応している事が示せます。

手続き中の rank を求める操作は愚直に行うと $O(n)$ 時間かかってしまいますが、FM-Index は BWT^S を Wavelet Tree 等のデータ構造で表現しているため、これを $O(\log \sigma)$ 時間で計算する事ができます。したがって、FM-Index の検索の時間計算量が $O(m \log \sigma)$ である事が示せました。

ここまで説明で、FM-Index が

1. $O(n \lceil \log \sigma \rceil)$ 時間で構築可能
2. $|\text{match}(T, P)|$ が $O(m \lceil \log \sigma \rceil)$ 時間で計算可能
3. $n \lceil \log \sigma \rceil + o(n \lceil \log \sigma \rceil)$ bits で表現可能

である事をそれぞれ示す事ができました。したがって、これで「クソ長いテキストから超高速かつ超省メモリに部分文字列の出現回数を数えたい」という要求を満たすデータ構造を構築する事ができました。

7 おわりに

いかがでしたか？ FM-Index はうまい形で圧縮する事もできて超楽しいのですが、著者の気力の都合でその部分を載せる事ができなかったので興味を持ったなら調べてみてください

^{*9}厳密には $\text{BWT}_i^S S[\text{SA}_i^S, n]$ が $t+1$ 文字未満である場合を考慮しなければいけません

クソ長いテキストから超高速かつ超省メモリに部分文字列の出現回数を数えたい

さい。文字列周りは全てがクソ面白くて最高なので、この記事を読んでいる皆さんも是非
文字列をやりましょう、レツツ文字列！

WORD 編集部へのお誘い

文 編集部 public_yusuke

我々 WORD 編集部は、情報科学類の学類誌「WORD」を発行している情報科学類公認の団体です。

WORD は「情報科学」から「カレーの作り方」までを手広くカバーする総合的な学類誌です。年に数回発行しており、主に第三エリア 3A、3C 棟などで配布しています。過去の記事は、公式ホームページ (<https://www.word-ac.net>) 上にアップロードしておりますので、是非読んでみてください。

編集部の拘束時間には週 1 回の編集会議^{*1}と、年に数回の赤入れや製本作業等発行に伴う作業があります。日常的に活動する必要はありませんので、他サークルの掛け持ちの障壁にはなりません。実際、多くの編集部員が他サークルと掛け持ちで在籍しています。

現在 WORD 編集部には、情報科学類生や数学類生、比較文化学類生、大学院生などが在籍しています。

少しでも WORD に興味を持った方は是非、情報科学類学生ラウンジ隣の編集部室 (3C212) へいつでも見学に来てください。時間を問わず常に開いています（人がいない時間帯もありますので、事前に連絡いただけたとより確実に案内ができます）。

質問がある方は、word@coins.tsukuba.ac.jp か、Twitter アカウントをお持ちの方は @word_tsukuba までお気軽にお問い合わせください。

^{*1}通常であれば編集部室に集まって行いますが、新型コロナウィルス感染症が流行しているため、今年度の編集会議をどのような形態で行うかについては未定です

情報科学類誌

WORD

From College of Information Science

WORD は大学新聞の学内誌特集からハブられました号

発行者	情報科学類長
編集長	一木 祐介
	筑波大学情報学群
	情報科学類 WORD 編集部
制作・編集	(第三エリアC棟212号室)

2022年6月5日 初版第1刷発行

(128部)