㉟ 最短経路問題 —ダイクストラのアルゴリズム

内田 剛
浦 昭二

頂点 i \ 頂点 j	1	2	3	4	5	6
1	∞	7	∞	∞	1	2
2	∞	∞	1	∞	∞	∞
3	∞	∞	∞	∞	∞	5
4	∞	∞	5	∞	∞	∞
5	∞	∞	∞	3	∞	∞
6	∞	6	∞	∞	∞	∞

図 3 有向グラフの隣接行列

表 1 実行例

	S	w	$D[2]$	$D[3]$	$D[4]$	$D[5]$	$D[6]$	$P[2]$	$P[3]$	$P[4]$	$P[5]$	$P[6]$
1	{1}		7	∞	∞	1	2	1	—	—	1	1
2	{1, 5}	5	7	∞	4	*1	2	1	—	5	1	1
3	{1, 5, 6}	6	7	∞	4	*1	*2	1	—	5	1	1
4	{1, 4, 5, 6}	4	7	9	*4	*1	*2	1	4	5	1	1
5	{1, 2, 4, 5, 6}	2	*7	8	*4	*1	*2	1	2	5	1	1
6	{1, 2, 3, 4, 5, 6}	3	*7	*8	*4	*1	*2	1	2	5	1	1

れる．$D[j]:=C[1,j]$ とする．

$P[j]$ の初期化．$j\neq 1$ で $D[j]:\neq\infty$ の j の P に 1 を入れる．

ステップ 2. $V-S$ の中から $D[w]$ が最小となる頂点 w を選ぶ．S に w を付け加える．

ステップ 3. $V-S$ の中の各頂点 v について，$D[v]$ と $D[w]+C[w,v]$ とを比較して，小さいほうを改めて $D[v]$ とする．

$D[w]+C[w,v]$ のほうが小さかったなら，$P[v]:=w$ とする．

$V-S$ がなければ終わり，あればステップ 2 へ戻る．

❖例

ダイクストラのアルゴリズムを図 2 の有向グラフに適用してみよう．アルゴリズムの動きを**表 1** に示す．

第 1 行は，ステップ 1 での初期状態の設定である．2 行目から 6 行目までは，ステップ 2 とステップ 3 の結果を示している．D 欄の値で * 印がついているものは，S に含まれていることを示す．

2 行目では，前行の $D[j]$ の中で $D[5]=1$ が最小なので，w を 5 として，S に含める．次に，$D[v]$ を小さくできないかを調べる．$D[4]=\infty$ であったが，頂点 5 を通れば $D[4]=4$ とできる．これに伴い，$P[4]=5$ とする．以下，同様にして，第 6 行まで進む．

6 行目では，S にはすべての頂点が含まれ，各 D には最終的な最短経路の長さが示される．最短経路は P を見ることによりわかる．$P[j]=i$ とは頂点 j の前は頂点 i であることを示す．頂点 4 に行く最短経路を調べるには，$P[4]=5$, $P[5]=1$ と見る．これにより，$1\rightarrow 5\rightarrow 4$ であることがわかる．

❖まとめ

ダイクストラのアルゴリズムでは，逐次，特別な経路の最短のものを選んでいくことにより，全体をみても最短経路のものを得ている．目先の最善を選ぶ方針で導いたアルゴリズムを欲張りアルゴリズムというが，ダイクストラのアルゴリズムは欲張りアルゴリズムがうまくいく例になっている．

アルゴリズムのステップ 2 で，特別な経路の最短のものを選ぶ．このとき，最短な経路は特別な経路以外にないのかという疑問が湧く．実際には，他に最短路はない．詳しくは参考文献 1) を参照されたい．

実行時間は隣接行列の大きさと弧の数に依存する．頂点数が n 個のとき，隣接行列は $n\times n$ の大きさになる．弧の数が n^2 に近いときは配列を用い，実行時間は $O(n^2)$ となる．

出発点を一つに定めた最短経路問題の他に，任意の頂点と任意の頂点の最短経路をすべて求めたい場合がある．これには，すべての頂点を出発点としてダイクストラのアルゴリズムを用いることもできるが，他に，フロイド (R. W. Floyd) のアルゴリズムがある．

❖参考文献

1) A. V. エイホ, J. E. ホップクロフト, J. D. ウルマン：『データ構造とアルゴリズム』, 大野義夫訳, 培風館, 1987.
2) E. W. Dijkstra: "A Note on Two Problems in Connexion with Graphs", *Numerische Mathematik*, 1, pp. 269-271 (1959).

（うちだ ごう, うら しょうじ 慶應義塾大学）

1987.11

Common Lispアラカルト14

対談：Lispマシンとパソコンの狭間で

John Teeter
井田 昌之

金鉱と田園志向プログラミング

I（井田） もうどのくらい日本にいるのですか．

T（Teeter） 6月の初旬からで約3カ月になります．月曜日（8月31日）に帰ります．

I もう，あと少しですね．

T ハイ．3カ月いたけれど，やらなければならないことはまだまだ残っています．

I これ，もってきましたよ．（Common Lisp/coreの資料を渡す．）

T ありがとう．参考にします．スモールモデルがメモリにのりきらないので．

I さっそくですが，今話題になっているCL（Common Lisp）のウィンドウ・インタフェースの話はどう思いますか？

T いいえ，まだ見ていません．しかし，オブジェクトの場合などもいい一例なのですが，我々は基本的に，優れた研究者達が討論を繰り返して作り，また合意した仕様というものを尊重します．どんなものであっても，標準へのプロポーザルとしてまとまってきたものは良いものだという認識をもっています．

I ウィンドウについてはXの開発者 Scheifler 氏自身が CLX というスペックを書いていますね．また，これはいくつかのCLの処理系での実験が始まっています．

T そうですか．それはいいですね．

I Scheifler 氏とのコンタクトはありますか．

T まだありません．今はマイクロソフトのOSがありますからマイクロソフトのウィンドウ・システムを念頭におきながら，一方でXにも興味をもっています．マイクロソフトXかなあ…．

I あなたの会社は MIT のすぐそばですね．

T はい．

I だからXウィンドウの話はしやすいわけですね，西海岸の人達よりも．（笑）

T はい．我々は既にXウィンドウ・システムのコピーをもっています．インタフェースのテストをしています，Goldworks なんかとの．CLX のプロポーザルはまだ見ていませんが，いずれにせよ，それは良い出発点のはずです．標準のプロポーザルがなければ話も始まらないというのが我々の立場です．でも，なにしろ米国を3カ月離れてしまっているので…．2カ月はいいけど，やっぱり3カ月になるとちょっと（情報収集の点で）きついですね．この間，多くのことが起きたようです．

I ここに編集部でまとめた質問事項があるので，それをちょっと読んでみましょう．

1. AI用のハードウェアとして汎用機対専用機のどちらが良いかというような議論があります．その中でなぜGHC（Gold Hill Computers）社は PC をターゲットとしているのですか？ 特に，実用上の問題はないのですか．スピードとかメモリとか，価格とか．

T まいったなあ…．（笑）

I 2番目は何故，インテルラインに従っているのですか．モトローラや RISC チップなどは考えないのですか．3番目は，Common Lisp の今後についてあなたはどう考えますか．

4番目は，米国での Lisp 文化の状況，特に Prolog 文化などとの比較ですね．

5番目は Concurrency（並列性）ということについてです．GHC 社にはコンカレント Common Lisp がありますね．6番目はハッカーと呼ばれる人達のことをどう思うか，ということです．**bit** では Stallman, Crispin, Scheifler などの各氏との座談会をやっています．Stallman の freeware という考え方もあります．

T それは社会学的な質問ですね．（笑）

I 7番目は，GHC 社はケンブリッジにあるので，そのエリアはコンピュータ・サイエンス，AI にとって重要なエリアです．そのエリアの状況などを紹介して下さい．

8番目は，日本の印象．

9番目は，できれば日本の **bit** の読者へのメッセージを，というのが **bit** の編集部からの注文です．

T OK，一番目のテーマが重要です．今聞いた質問のうちのいくつかははっきりと回答できるけど，あとのいくつかはうまく答えられませんね．（笑）

I たしか Teeter さんの家は，どちらかといえば郊外ですね．

T そうです．それがいくつかの質問に答えられない理由です．私はあまり都会らしい所が好きでないので，例えばケンブリッジの様子とかハッカーメンタリティとかは知らないのです．

I そう，郊外の家と比べればケンブリッジはゴミゴミしてますね．家の広さはどのくらいでしたっけ．

T 私の家は60フィート×40フィート（約18m×12m）です．

米国では室内のとりかたが違います．日本とくらべて安いから，たくさん部屋をとれます．

I たしか野原とプールとそれと，山と….

T いや，私はケンブリッジには山はもっていません．アイダホにいけばありますけど．

I Gold Hill ですね．

T そうです．それが会社の名前の由来です．

I 金が出るから Gold Hill ですか．

T そうです．金山があったのです．1840年頃から金の採掘が始まり，今でも5つ金山があり，そのうち1つは今でも採鉱しています．1905～1920年は一番さかんでした．今やっているのは1人です．（笑）

Gold Hill という名はたくさんあります．あちこちで金がとれましたから．

私が高校の時，ドリリング，探鉱などもやったことがあります．

I 都会にいるよりも，広々とした自然の中のほうが好きなんですね．

T Gold Hill という社名はいろいろな意味で，探索すべきアイデアなどをこつこつと探り出すイメージがあります．これが GHC のカンパニーカルチャになっています．

量産マイクロと共に歩む

I GHC 社の社員人数は今増えているときいていますが．

T はい，去年大変増えました．今100人の従業員がいます．正確には101人ですが．

I どこからきているのですか．

T 多くの人は LMI からきました．彼らの多くは大変優秀な Lisp 屋たちです．

I GCLISP の GMACS は結構いいと思いますが，どういう開発になっているのですか．

T GMACS の開発は3つの世代があります．最初のものは Purdue 大学の学部学生が作ったものです．2番目のものは Dan Brutsky が作ったもので，彼は今 San Marco Associates にいます．3番目のものは Bob Jeen Commo，彼は LMI からきたのですが，彼が作ったものです．

ボストンの Lisp コミュニティは大変小さく，みんなお互いによく知っています．だから，GHC 社の製品が良くなるにしたがって皆に知られ，Lisp ハッカーのほとんどは PC 用の Common Lisp として GHC をすすめてくれるようになりました．最初は今も持っている Symbolics マシンで始めました．そして Symbolics の上にインタプリタを作り，クロスコンパイルで GCLISP を作り始めました．

I 3年前に行った時，Jerry B（Jerry Barber 氏．GHC 社創立者の一人）がクロスでやっているのを直接見ましたよ．

T はい，クロスコンパイルは意外と大変でした．なぜかというと，環境が Symbolics と PC ではまったく違うからです．

I Symbolics や TI などの Lisp マシンを今どう見ていますか？

T この2つのマシンはそれ自体違った面をもっています．マシン自体はそれ自身どんどん進化しているでしょう．大きくなり，能力も増え，カスタムマシンとしても良くなっていくでしょう．これらのマシンでできることのいくつかは，いわゆるそのへんにあるマシン，マ

井田昌之氏（青山学院大学）

イクロでは難しくなってきています．実行時のタイプチェックなどがその例ですが．大きな開発環境というのはまだまだ進んでいくでしょう．それは単に計算能力の進歩ということではありません．我々も前に進みつづけます．そういえば，LMI チップは100倍も Symbolics より速いそうです．

I　100倍？

T　そう．私はまだ動いているのを見たわけではありませんが，こうしたマシンは売られるようになるでしょうから，そうなると，市場もどんどん変わるでしょう．また，そうした会社の経済的な状況も好転するでしょう．

また，ネットワーキングも重要です．カスタマ・サービスなどもよくできるようになります．

(注：LMI 社の倒産と Gigamos 社による再建がこの辺の会話の背景にある．)

GHC 社は専用チップというものを採用する方向は取っていません．なぜなら，我々はなおも一般市場に大量にあるマイクロプロセッサを重要に思っていて，かつそれはハイエンドのアプリケーションにも使える日がくると思っているからです．インテル8086ラインのものは訓練・教育用と簡単な応用のためで，開発用のマシンではないと思われていますが，しかし，4年の間に8086，286，386と進んでいるのです．

I　386のあとはどうなのですか？　たとえば IBM が386のあとは特殊なチップで専用のアーキテクチャにしてしまうという可能性もある．

T　それはウワサでしょう．いずれにしても，インテルからはもっとすぐれたチップが出てくるでしょう．モトローラからもね．我々には我々の席があり，極端に IBM 追従というわけではありません．3～5年あるいはそれより少ない期間にチップテクノロジーが進むことはあるでしょうが，こうしたテクノロジーの進歩は，チップの中により良い機能を組み込むということで競争をしています．RISC マシンというのは極端なものですが，もう1つの流れは機能をたくさん入れようというもので，386はその典型的な例といえます．

I　IBM の新しい OS の出荷は遅れているといわれていますが．

T　我々は386チップのマーケットも既にとりかかっています．386のボードを売っています．チップ自身は多少問題があるらしいが，6カ月やそこらの問題はたいして気にしない．新しい安定した状態になるでしょう．ボードとして完全に機能すればいいのです．もし IBM が386をカスタマイズしたものをやっているとして，我々がこの先どうするかははっきりしませんが，たぶん1つの可能性としては，Generic Supplier が部品として386を使った良いものをたくさん出し，選択の余地も広がり，競争によって価格も下がるでしょう．

また，少なくとも過去においては PC マーケットに対して IBM はオープンな政策をとってきました．たくさんの人々が入ってこれるようにしていました．

I　GCLISP は MS-DOS/PC-DOS に依存していますが，その意志決定にはどういうことがあるのでしょうか？　IBM が PC-DOS だからということでしょうか？

T　IBM が選択した OS を GHC 社は常に選択していきます．

Gold Hill の市場というのは，もちろん Lisp それ自身と教育，PC，ということがありますが，主なものはビジネス指向の AI アプリケーションです．そのマーケットは基本的に IBM の顧客です．そこで IBM が選択する OS を使うことになります．これは単に数の問題です．UNIX が増えていれば UNIX へいく．クローンが多くなければその対応もします．

I　フェニックス BIOS でも問題はないのですか．

T　問題はないです．最初にそれをクリアしています．我々のシステムは PC 互換性のテストソフトにもなっています．(笑) 我々のソフトはローレベルの機能をよく使っているので，PC互換というマシンの多くで，BIOS 製作者達は GCLISP をテストに使っているのです．コンパック BIOS ではいくつかの問題があったのですが，それも我々のソフトを動かすことで発見し修正しました．NCR とか，たくさんのマシンの非互換性を GCLISP は見つけました．

I　日本の PC などでは困難はありますか．BIOS の

な命令セットのマシンを高速にして，大変強力なマシンになることを示しています．

I　昨日，日本でも Sun 4 の発表がありました．

T　ほう，それは知らなかった．

I　10MIPS マシンですね．

T　大変に速くて大変にシンプルです．

I　たしか日本円で1700万円ですね．12万ドルくらいですか．

T　10MIPS という値には気をつけないといけません．なぜかというと，命令数というのは秒当りの命令数であり，その命令の能力には関係がなく，別のアーキテクチャの 1MIPS のマシンより10倍速いというわけではありません．だから値そのものには注意する必要があります．もちろん Sun4 が大変速いということに変わりはありませんが．

I　モトローララインをベースにした Sun3 から Sun4 への連続性についてはどう思いますか．

T　Sun はその連続性の確保のためにあらゆる努力をしていると思います．Sun はいい会社だと思います．しかし，まだ彼らが連続性の保持を完全に成功しているとは思っていません．すべてのツールが Sun4 で行なえるわけではありません．Sun4 は，大変印象深いハードウェアなので，Sun のもっているソフトウェアのサブセットだけがその上で使えるという事態に疑問をもっています．今後はどうなるでしょうか．いずれにせよ，RISC マシンの上で Lisp が速く動き，ベンチマークでは Symbolics より良い性能が出ることを示したのは重要なことです．

I　でも，Sun4 の RISC には Lisp 向きの機構，たとえば NIL フラグとか，そういったものはついていませんね．

T　そう．それがカスタムチップと RISC の大きな違いです．また，RISC チップでも，インテルやモトローラが出すとすれば，大量に出回り人口も増えますが，Sun の出している RISC は Sun だけで出るからそう大量でもありません．したがって，そうしたチップのコストはワークステーションのコスト中で，かなりの割合を占めています．386 チップでもまだ高くコストのかなりの部分になります．量産の効果があがって 386 チップの価格が Sun4 の RISC チップよりももっと安くなることを期待しています．

量産のチップは低価格への道が開けており，RISC チップには高速化，Lisp カスタムチップにはより高機能化という道があるのです．実行時のタイプチェックとか

いろいろ Trade-off があります．Symbolics は High-end へ，RISC は高速化とベストツールへ，Sun はワークステーションから AI マーケットに入り，Symbolics より全体の価格を下げています．比べれば，機能の豊富さは劣りますが速度は速い．Gold Hill は，下から上がってきます．底から上がってきます．大変小さなマシンから機能を上げていき386を使い，Symbolics に匹敵する速度を出してきています．そして安い．速度の比較表を見てほしいです．

1万ドルの AI エンジンを目指して

I 386を使った HummingBoard は速いと聞いていますが，HummingBoard は誰が作ったのですか？

T Elick Soly と Gred Papedarls が作ったのです．Gred がハードボードを，Elick がアーキテクチャを設計しました．

I HummingBoard のクロックは 20 MHz でした？

T そう，20または16です．それは基本的にはスタンドアローン・ボードなのですが，実は Lisp 向きの特殊な回路をもっています．簡単なものなのですが，タイプチェックなどをやります．タイプコードをしらべるのはマイクロでは高価な仕事なのです．そしてそれをしなければなりません．メモリ参照の際にはいつも必要になります．また，いくつかの種類のオブジェクトは即値としてポインタそれ自身に入っていて，メモリ参照がへらされています．そして処理の統一性も必要です．また，ある種のアドレスに対しては値のデコードをさぼることもしたいです．物理メモリがない所です．

I タグフィールドは32ビットポインタの中ですか，外ですか．

T 外です．それが大きな違いです．

Symbolics ではタイプコードはポインタの中です．ポインタは32ビットより大きくなります．我々のものではポインタはタイプコードをポイントします．それをとるのに余分なメモリ参照がいるわけです．

I TI explorer では32ビットの中にタイプコードを入れて，実際のアドレス空間は小さかったですね．たしか Symbolics は 40 ビットポインタで 8 ビットのコードと32ビットのアドレスに分かれていたと思います．

T そう，アドレス空間は Symbolics は広がっているが，ポインタの処理は32ビット幅ではすまなくなる．

I 同じ方式をとるのはマイクロ向きの設計ではないですね．

T そう．Symbolics について正確なことは知りませんが，私は Symbolics はそうしなくてもいいのだけれどソース互換性のために即値を使っていると理解してます．

I Symbolics も386ボードを出し始めるようですね．IBM PC の仕事が Symbolics でできるようになるのでしょうか．また，多くの西海岸の Lisp ベンダも 386 上での実現をはじめていますね．だから 386 市場はこれらの主戦場の１つではないですか？

T そう思います．286 はほとんど完全に Gold Hill の領分です．そして386など32ビットマシン，68020などにくると，たとえば Symbolics も32ビット市場，IBM の動きなどに気づくようになったのではないでしょうか．たとえば Inttelicorp，KEE の会社，6000～10000 ドルの規模のものをやっている会社が 2000～5000 ドルの規模のものを始めるというのは会社の構造を変えなければやっていけないという要素が出てくるのです．それに成功するかは私はよくわかりませんが，会社の内部の構造がやっぱり違うのです．

I Lucid に行った時，彼らは彼らの会社のあるフロアーはかつて Inttelicorp がいた所だと言っていました．

T そう，いろいろとこの２つの会社は関係があるようですね．KEE と Sun は彼らのカギです．我々の仕事は Goldworks です．（笑）

I あなたが今くれた GHC 社の企業戦略の文書をかいつまんで説明してくれませんか．

T これは AAAI '87 に出したものです．米国から送ってもらいました．たくさんの人が我々に，さあ次は何をするのかい，と聞きます．答えは，我々は完璧に Common Lisp を実現する．そして Goldworks を成功させる，と言います．また他の既存のアプリケーションとのインタフェース，特に Lotus とか dBASE とかそのインタフェースをやります．ウィンドウシステムとのインタフェースもやります．また，分散型のシステムについても開発を進めます．コンカレンシー，コンカレント Common Lisp は１つの製品ですが，我々はもっと一般的な汎用のアーキテクチャにも興味があります．分散型の AI です．スタンドアローン・マシンというのはやっぱり高価です．各々機能をもったマシンの連係がこれから重要です．

I あれ，GCLISP 3.0 では PCL (Portable Common Loops) が動くのですか？（資料を見ながら）

T はい，PCL が動くようになりました．

I うん，PCL が動くというのはそれが Common

Lisp であるということの1つの証拠として使われています．というのは PCL のソースには Common Lisp 的なやり口が多数入っているからです．

T そう，知っています．1つのデファクト・スタンダードです．

I 3月の時はダメでしたね．

T はい．

I Gregor (Gregor Kiczales, Xerox Parc) は，その頃 Gold Hill の連中ががんばろうとしているらしいと，こっそり教えてくれましたよ．(笑) そうそう3月に Parc を訪問したら，あとから Carl Hewitt が入ってきましたね．(笑)

T そう，Carl はあっちこっち行くんです．日本にも何回かきています．

T 私は Common Lisp 処理系の戦争については，我々の 286 版は最高速というわけでもなく，また完璧な Common Lisp でもありません．汎用性，共通性のために速度を犠牲にしてきたこともいくつかあります．今，速度をがんばっています．でも共通性というのは大事です．

I PC 上の AI の将来はどう思いますか．

T もっとスタンドアローンで使えるようになり，もっと自分自身での機能が充実するでしょう．

I ネットワーキングの中で？ 外で？

T ネットワーキングの中では機能分担ということになります．外では，PC はあまりたくさんのハードをもっていませんが，そこではそれに応じたことをさせるわけです．そう，我々は1万ドル以下の AI 環境を目指しています．たとえば2000台のマシンで AI を各々やるとします．各々のコストの差は2000台になると大変です．したがってそこに我々の回答があります．PC 間の通信とネットワークも重要です．データをコンシステントにとっておいたり，連絡したり．

I 先月号では外部言語インタフェースのことを書いておられますが，やはりユーザからそうしたクリティカルなニーズがあるのですか．

T そうです．既に別のアプリケーションが動いていて，それと連絡する必要もあります．また，こういうこともあります．暗号化されたデータの扱いが必要でその暗号化/復号化の機構はC言語で書いてあり，それをもう一度 Lisp で書き直すことはダメだというわけです．

私の個人的な意見は，外部インタフェースはすべて（ネットワーク的な意味での）メッセージ通信指向となるべきだと思っています．関数呼出し型のインタフェー

スよりもいいと思います．データの型の扱いもきれいにいきますし．

I　数値計算のコードを Lisp で書くことはどう思いますか？　伝統的に Lisp では数値は各々メモリに格納されるからセルを使います．ガベジコレクションの頻度が増すわけです．

T　そう思います．Cなんかでのように中間結果をレジスタにとっておくイメージを Lisp の共通仕様の中でできたらいいでしょうが，もちろんコンパイラはある程度これをやりますが，やはり一般性の点ではなかなかうまくいかないでしょう．別の手続きで作った浮動小数点数を他の手続きで利用するといった場合，Lisp ではどうしても余計な手間がかかることになります．こんな点でもCとのインタフェースをもつことの意味があります．言語自身の特性からCのほうが計算途中の値の処理は楽です．Lisp では，たとえば hash-table もそれほど便利ではありません．

ネットワークに対応する

I　TCP/IP インタフェースもたしか GCLISP にはありましたね．ユーザには提供されていますか．

T　ネットワークは我々はなるべく層化して抽象をもたしてユーザからは見えるようにしています．もちろん各層へのユーザからのアクセスも考えています．

I　たしか Gold Hill は ARPAnet は入っていないですね．どうしてですか．GHC 社のネットワーキングソフトが良くないからではないかという邪推も聞きましたが．

T　2つの側面があるんです．1つは ARPAnet に入るということは，政府の資金による仕事をしているということです．Gold Hill はけっして政府資金をもらった仕事はしていません．今，CSnet には入ろうとしています．

I　そりゃいい．

T　2つ目はお金です．

I　今どういう形態でネットワーク接続をしているのですか．

T　Symbolics Dialnet です．電子メールのネットアドレス・経路を見てもらえばわかります．

I　そうですね．

T　でも今 CSnet を考えていますから，少しすれば状況は良くなります．GHC の中では TCP/IP も CHAOS も使っています．Symbolics はファイルサーバにもなっています．

I　しかし，誰が言ったか忘れましたが，米国には2種類の大学がある．ネットワークにつながっている所とそうでない所と．（笑）これと同じ感覚で GHC 社もネットワークに入っているといいですね．

T　今 MIT とつないでいるけれど，まだ不充分です．ARPAnet を使うということは会社にとって重大な意味があるんです．

I　たしか live-oak とつないでいましたね．live-oak はパブリックマシンなのですか？

T　よく知らないけど，Symbolics Dialnet でつないでいます．井田先生のところは Dialnet は？

I　いや，まだなんです．

T　Dialnet は uucp のように point to point で簡単ですよ．日本でのネットワークについていえば，NEC とか Fujitsu とか，各社の内部のネットワークはしっかりしてますね．

I　でも，JUNET というのがあって，相互に通信できますよ．

T　でもそれはローカルエリアネットではないでしょ．全体のネットは重要です．日本でも GHC 社は分散環境での製品を考えています．NEC スタイルのネットとか，他のとか各々を考えています．トランスポートレベルは Lisp 自身から分けています．TCP/IP，IBM スタイル，iPSC インタフェースなど違っているので各々対応させています．日本はもっとまともなネットを張るべきです．

I　私の厚木の教室では60台の NEC-PC を NEC スタイルのネットでつないでいます．1Mbps でつないでいます．

T　本当ですか．

I ええ，同軸リンクです．

T ほんと！

I イーサとはちょっと違いますが．

T じゃ GCLISP でも対応できますね．基本的な機能はどうなっているんですか．

I そう，ROM にある程度あって，それから提供ソフトがその上にのっかるんです．

T GCLISP では EPI スタイルのインタフェースがあって，これは他言語というだけでなくネットワーク・インタフェースでもあるんです．だから Lisp 外のネットワークコードともうまくいきます．その辺のプロジェクトを青山でやったらどうですか．(笑)

Stallman 流 freeware は？

I 別の話になりますが，R. Stallman 氏は freeware の概念を主張しています．著作権は残しているがパブリックドメインにある．それはソフトウェア会社にとって大変重要なできごとだと思いますが．

T GHC 社としての答えと私個人の答を言いましょう．2つの異なったフォーカスがあるでしょう．GHC の答えとしては，常にすべての組織はコストのかからないソフトを使いたいという選択の自由があります．コストファクタの点でも，また品質の点でも，ソフトが完全な高品質のものとして提供されたいとは必ずしも思わないということがあります．

一方で，長期的なサポートを必要とするという考え方もあります．長期的に安定したドキュメント，安定したユーザサポートが必要な会社もあります．それは製品の名前という象徴的な問題になります．ある会社ではマイクロソフト・ウィンドウを使っています．X-ウィンドウは使いません．なぜかというと，マイクロソフトウィンドウは，マイクロソフトという名前がついているので，それは PC では非常に信頼された名前ですからよく知られています．一方，X-ウィンドウは PC ではまだ信頼されていません．

だから，たくさんの種類の意志決定や状況があるのです．品質の保証は，ユーザの知識レベル，カストマサポート，原著者が製作改良の努力をつづけるか，どれだけ内部努力をしているか，いろいろなことがあります．これが GHC 社としての答えでしょう．彼はいい選択をしていると言えるでしょう．freeware を信奉する人もそうでない人も，両方とも共存できます．freeware の人はソフトウェアをどんどん良くしていきます．いいものを入

れていきます．non-free soft の人は，より良いカストマサポート，より良いドキュメント，より良いフィールドサポートに努力をしていきます．

I ユーザは選択できるようになるということですね．

T 私個人の答としては，私は両方の種類のソフトを書くのがすきです．(笑) ときどき freeware を書きます．技術者として，freeware を提供したいと思います．また，自分のソフトを書くのに良い freeware soft が使えれば開発も楽になるし，いいものを作ることができます．もちろん我々はいいものを使うことにお金を払います．それは我々のソフトを良くすることにもなります．

I 環境のソフトとか．

T そうです．また，私個人は多くの種類のソフトウェアは freeware におくべきだと考えています．会社で作ったものもです．ARPAnet その他にメイリングリストがあり，そこには良いソフトがのっています．自分でもそうしたソフトを使っています．そういうのがあるので，良いソフトを自分で作るときの支えになっています．

I 京大の GMW，Wnn など，日本でも始まっているように思いますが．

T それはいいですね．京大の Common Lisp，KCL はいい例です．これは必ずしも freeware ではありませんがいい例です．US では KCL は品質の保証を要しない CL ユーザに多く使われています．大学でのソフトウェアの開発は freeware に向くでしょう．Lisp について言えば PC 用に XLisp というのがあって，これは小さな処理系ですが，そうしたニーズに対応しています．

I　日本にもきてますね．

T　freeware の1つの危険というのは，大変な多くの種類のソフトがありますが，あの種の品質のコントロールがあってフリーウェア・ライブラリに入ることが必要と思います．なぜかというと，汚染（pollution）の危険があるからです．くずのコードが入ってしまう可能性があります．汚染の浸透という問題です．ネットワークでいきますから．

Small ZetaLisp が発端

I　3, 4年前の GHC 社の様子を教えて下さい．というのは，この数年で非常に大きくなった会社だと思うので．

T　この3，4年というのは大変でした．1983年には15か16人の従業員でした．我々は製品もなく，自社製品を出そうとしている所でした．懸命に働きました．

I　そう，Stephany（Teeter 夫人）は私にこぼしてましたよ．そのころはほとんど仕事ばかりで家のことをかまってくれなかったって．（笑）

T　時々そんなこともありましたっけ．（笑）

I　日本のビジネスマンと同じだったって言ってましたよ．（笑）ほとんど夏休みをとらなかったって．

T　最初のころは GCLISP のコードを書きながら，製品計画を立て，テクニカルサポートの電話をうけ，ボストン地区での製品販売，配達を全部していましたから．それで思うのは，会社をおこして大きくしていくには，大変よいビジネスマネジャが必要だということです．技術者はやはりビジネスマンではありませんから．

I　思い出したけれど，Jerry B に会ったとき彼は最初の版の GCLISP は彼が INRIA にいて，そこで作った Lisp がもとになっていると言っていました．

T　そう，元になった版は Jerry B のフランスでのアイデアですね．

I　そして Jerone Cailloux が，そのあと INRIA にきた（?）Jerry B がそのままいたら INRIA は Common Lisp をやっていたかも．(笑) Jerone は LeLisp をやっている．

T　実をいうと，最初我々は ZetaLisp 互換とするつもりだったんです．Symbolics の世界を IBM PC にということで．でもそのころ *Byte* マガジンで Lisp の特集があり，Common Lisp の話が出ていました．それを飛行機の中で見て方向を少し変えたわけです．ZetaLisp が事実上の標準と思っていましたが，その年の12月に Steele の Common Lisp のドラフトを見ました．そしてそれを読んで Common Lisp へと向かったのです．

I　1983年12月ですね．

T　そうです．最初は Small ZetaLisp にしようと思っていたんです．実際に商品として出したのは1984年11月です．その時からコンパイラを作り始めました．

I　Pattrick Winston の *Lisp* 2nd edition と GCLISP の関係について教えて下さい．その中には GCLISP の例がついていますから．

T　Jerry B は MIT の AI 研を出たのです．1983年10月にフランスから帰ってくると，彼は AI 研でリサーチアシスタントになったのです．1985年までその職にいまして，彼の滞在中に Patt は Lisp の教科書を書き出し，2nd edition を始めていました．そのころ会社を始めていましたから，会話型のチュートリアルをほしいと思っていました．そんなわけで関係があるのです．

I　ということは，これは San Marco Explorer の起源でもあるわけですね．

T　そうです，実際の S. M. E の起源はこうです．2nd edition のための実例を実際の機械の上で作って動かす必要があり，小さなマシンの上でそれを動かしてみたいと彼は言いました．そこで我々はテスト環境をとと

のえました．そして彼に提供したのです．今 Patt は第3版の準備をしていると思います．まあ，2版の例は実際に Common Lisp 的というほどではありませんでしが（笑），3版ではある程度また変わるでしょう．これは GHC 社を励ましています．

I Patt の家とあなたの家と個人的なつきあいがあるんですね．たしか一緒に競走馬をもっていましたね．

T そうです，4月に始めたんです．今はうちにおいてあるのでよくきます．そうすると，奥さん同志は馬の世話をして，私と Patt はバスケットボールで遊んだりしています．（笑）

I そりゃ，いいコミュニケーションですね．（笑）

T ちょうどインテルが新版の iPSC を出したんです．今までのは point to point の結合でメッセージは中継ノードのメモリに書き込んでつないでいきます．そして割込みで知らせていきます．デイジーチェインです．新しいボードはルーティングが違います．中継ノードをわずらわせることはありません．またプロセッサは割込みの必要がなくなってきて，また速くなりました．全体で前のより100倍速くなりました．

I その話はもうオープンにされているのですか．

T はい，今週の IJCAI でやっているはずです．

I それじゃ，ちょうど今日かもしれませんね．

T AAAI でアナウンスがあって，IJCAI で見せているはずです．それは16MB メモリの386をノードにしています．彼らは32ビット Lisp ではなく16ビット Lisp を，動かすと主張しています．そうすると性能が Cray より速くなるからです．

I Cray より速い？

T そう．ガブリエルベンチマークをやると Cray より速いんです．

I ほんと！（笑）

T そうそう iPSC は Cray より速い．

I カブリエルベンチをやるには同じソースでなければ意味がないけれど，iPSCのコンカレント Lisp は並列マシンだからどうしました？　そのままですか．

T そう，少し変えました．小さな違いがあります．それは三角形を作って上のは下の2つをコントロールするのです．16ノードでこれをやっています．木状に置いています．これは非常に簡単な分解です．

I まあ自然な方法ですね．

T 現在のコンカレント Lisp とその構造では非常に自然な方法です．メモリの共有が要素にありません．それが入ってくるとまた少し変わってくるでしょう．

I GHC の製品では例えば ACTOR モデルは入っていませんね．Carl は GHC を支えているけれども，関数コールは普通の考え方ですね．GCLISP 自身にはその意味で並列性がまだ基本的に組み込まれているわけではないのではないですか．

T そうかもしれない．普通の関数呼出しに対して我々は remote 関数呼出し，remote apply をおき，オブジェクトの import/export をおき，静的束縛の関数形式でシンボルの並列的なやりとりをしていくことができます．それは，まだ ACTOR モデルのような抽象的レベルまでいっていません．そのレベルまでいくと，すべてのことはオブジェクトのメッセージ送信に帰結するはずです．Common Lisp の逐次実行構造もそこまでいくはずです．

Common Lisp の推進

I CLOS や LOOP やエラー処理などの Lisp 拡張案をどう思いますか？

T まだエラー処理拡張案はよく見てません．LOOP は，私の見方では，技術的な話を除いてみますが，標準があるというのは無いよりいいということです．これが基本です．GHC 社は，基本的にそれぞれの案に対してこれはいい，これはダメ，ここを直せということを主張しません．案は Symbolics や Lucid や Xerox や TI や HP など多くの優秀な設計者達がいっしょうけんめい考えてくれているのですから．

I なにしろ早く決まってくれ，というのが GHC 社の基本線ですね．これは日本の多くの企業の考えに通ずる所がありますね．（笑）

T GHC 社がそうしている理由は簡単です．GHC にはマンパワーが足りませんから．そうした活動に人を出せないのです．また我々のもっている市場からくる性質もあります．標準が決まってから動き出すということもあります．

I TCP/IP なんかへの対応も同じような意味づけがあるのですね．

T そう．市場での要求が我々の製品の性格を決めています．漢字のサポートもそうです．市場が漢字を要求している．だから漢字を入れる．それは標準なものがいい．そこで今，井田先生から出て ANSI に出ようとしているものを参考にしたいのです．

I 6月の X3J13 で私の代わりに塩田さんが行き，ILA の人がそれをフォローし，ANSI で書き直そうと

しているのを知っていますか.

T ほう，それはいいや．GHC 社では，そう今から5週間でその件をやるのです．10月中旬にできるはずです.

I 米国の Common Lisp 処理系作成者の多くは漢字処理を重要と考え，今やっていると思います．いいタイミングですね.

T もっとも重要なことは，日本のマーケットは日本のものであってまちがいなく日本だということです．製品を日本向きに直していくことはやります.

I この2週間，私は学生と Teach-Zmacs の日本語版を作っていました.

T そりゃいい．それは Symbolics と一緒にですか？

I いやそうじゃありません．うちの Symbolics の上ですが.

T CEC では実は我々のオンラインドキュメントを日本語化することをほとんど終わっています．だから GMACS の中などでドキュメントが必要になった時，それは日本語で出てくるようにしているのです.

画面エディタと OS としての Lisp

I vi と Emacs のどっちが使われているか，なんていう議論はどうでしょう.

T そうですね，UNIX 外の人間は vi を知りませんね！

I アッハッハ，そうだった．PC のスクリーンエディタでは？

T そう Wordstar がやっぱり一番使われていますね．一番好きなエディタは何ですかという質問がよくありますね．それはやっぱり個人の嗜好に強く依存していますね．だから，自分の気に入っているものが一番いいと言います．AI 屋にとって，Lisp 屋にとっては Emacs がいいわけです．C プログラマにとってはやはり vi でしょう．そして2つの環境の融合として vi に Lisp シンタックスを入れたり GMACS に vi キーボードシークエンスを入れたりすることがあります．そして人々に残されたものは機能的な違いというよりも嗜好の違いということになってくるでしょう.

I 日本では混乱というか戦いというか，Stallman が来て，彼は強いから，GNU Emacs が UNIX 上で大きな話題になっています．いいことです.

T いずれにしても UNIX というのはCがベースです．Lisp をその上にもってくると考えると，そこにベースの違いが出てくるでしょう.

I 日本ではまだあまりそうした観点での話は出ていませんね.

T vi もシンプルでいいんじゃないですか．でも Lisp には機能が低い.

I GNU Emacs はエディタというよりも一種の OS だと思うんですが．それ自身 UNIX とは全く違うものです.

T Emacs は UNIX からきたものでなく，他の文化からきていますね.

I Emacs の中では UNIX などの OS のコマンドをたたかずに，OS 的な仕事ができます.

T GMACS でもそうですね.

I その意味では，GMACS-GCLISP は UNIX や MS-DOS をおきかえるものの1つの候補じゃないんですか？(笑)

T たしかにそうかもしれない．理想的にはそうです.

I 実は，私はそんなイメージをもって10年ちょっと前からパソコンと Lisp とを育ててきているつもりなんですが．(笑)

T Lisp それ自身 OS になっていく傾向があります．この意味では，UNIX の上でも UNIX コマンドでなく，Lisp 中で済むようにしたいでしょう．けれども現実の世界ではリアリティがないでしょう.

I 私がこの質問をしたのは，Sun 3～4 というラインの上にいい Lisp そして環境がのってくると，それはある意味で GCLISP の先にあるものでしょう．Lisp 環境の構造的概念は似ていませんか．UNIX と PC-DOS の違いで.

T 開発環境という点で似た仕組みですね．私の基本的な仕事です．でも OS というイメージには実行時のオブジェクトのチェックなどということはありません．しかし AI 屋にとってこれらは重要なことです．Sun の上で UNIX のシェルではなく Lisp の環境を直接呼び出すという状況はありえないことではないですね.

I Sun の電源をいれると AI 環境になるというわけですね．AI ターンキーシステムですね.

T この辺のイメージというのは，これからがおもしろい所です.

Prolog

I ところで Prolog などの他の言語のことをどう思

いますか？

T 他の言語ね．Prologは，ある種のアプリケーションでは大変簡単にそれを書くことができます．そして簡単に書けていいでしょう．論理型のプログラミング・ユニフィケーションによる機構などLispよりもいいものをもっています．しかし，エキスパートシステム，システムという点で考えてみると，私の見方では違ったものがあります．Lispは柔軟な制御構造があり，便利です．アプリケーションが違うということです．LispとCとPrologとを比べてみると，Lispよりローレベルにがいて，Lispより上のレベルにPrologがいる．下にあるほうが柔軟性が高くなるが，反面もっていない機能になるとユーザがなすべきことが増えます．宣伝にもなりますが，Prologの上にGoldworks，これはLispで書いた我々のシステムですが，そういうレベルづけになります．

今思い出しましたが，OSのビデオディスクのデバイスドライバ，これをPrologで書いたのがあります．これを最初みた時はびっくりしました．(笑)

I 誰がやったのですか？

T HPLab.です．

I 本当に？

T そうです．(笑) たしか7月の日本のAIショーでHPの人がやっていました．HPの人はPSLでやり，それをPLISPで書き直し，それをPrologに書き直したと言っていました．これは言語間の比較の良いテーマを与えてくれるでしょう．

おわりに

I 最後に読者に，特に，AIなどというわけでもなくマイクロというわけでもなく，これからコンピュータ・サイエンス一般を志す若い人達へのメッセージをお願いしたいのですが．

T たくさんの機会を生かすことですね．自分のキャリア，機会，専門，他の種類のこと，その上で環境として大きい会社，小さい会社，開発，研究，ラインなどたくさんの選択があります．私は，高校の頃からやっていた井田先生と違って大学を出る頃からソフトを始めました．ミンスキー教授が言っていますが，ソフトウェア・サイエンスの将来は娯楽（エンタテインメント）にあると．(笑)

小さい子供も含めて，多くの人は娯楽を通してコンピュータに接しています．そして満足したオーナになっています．うちの子供もファミコンを米国にもっていて，それを今回もってきて楽しんでいます．ビジネスの話は大変です．しかし最終的には，ほとんどの人は娯楽でコンピュータを使うんじゃないですか？ 次世代のゲームを考えるのもソフト産業の大きな仕事でしょう．結論的なことをいうとすれば，オープンマインドでいる，これが重要だと思いますね．チャンスはたくさんあります．

I どうもありがとうございました．

(1987年8月26日収録)

(ジョン・ティーター，Gold Hill Computers 社副社長
いだ まさゆき　青山学院大学)

まにゅある倶楽部

6．読者を知る

高橋　剛

マニュアルに限らず文章で誰かに何かを伝えようとする人は，必ず読者を想定しています．読者の関心事，知識レベル，心理的欲求，文章の読み方と内容に対する反応などさまざまな側面から読者を検討し，それから文章を作り始めると思います．例えば，文学では心理的側面に訴えるための読者の研究が重要ですし，マニュアルの世界では読者の関心事と知識レベル，そして理解の過程を知ることが最も重要になります．ところが，この“読者を知る”ことが実は大変難しい．読者の想定さえ的確にできればマニュアル作成の70%は終了したといってもよいくらいです．残りの30%はテクニカルライターの努力と技量でカバーできるのです．

マニュアルで伝える情報は読者に理解してほしいものばかりです．商品を使って仕事をし，効果を得るための必要なすべての知識をマニュアルに盛り込んでいるのです．しかし，書店には市販ソフトの「よくわかる…」が氾濫し，またメーカや販売店には，煩わしい問合せの電話がかかっているのです．つまり，既存のマニュアルは読者を満足させていないということです．結局のところマニュアルを作る側で読者の想定を誤っているのです．我々の（マニュアル作成）業界では読者想定を誤った古典的なエピソードとしてフロッピディスクの取扱いの例があります．「フロッピディスクをケースから取り出してドライブ1に差し込んでください．」これを読んだあるユーザはディスクのあの黒くて堅いケースを破り，中の円盤そのものを取り出した，というのです．本誌の読者にはこんな人はいないと思いますが，世の中にはいたのです．これはそんな読者が悪いのではなく，そういう人も読むのだ，ということを想定できなかったマニュアル作成者の責任です．

マニュアル作成段階で読者を的確に想定できない理由は主に次の二つです．一つは，読者を知る有効な手段が確立しておらず，はっきり言って情報不足である，ということ．もう一つは，誰に何のために使ってもらうのか商品自体の性格があい昧である，ということです．後者の場合はマニュアル作成側の問題ではないのでここでは省略します．余談ですが，マニュアルを作成する者たちのある集まりで，マニュアルの定量的評価の方法について議論しました．頼りになる評価基準に乏しいとか，誰が評価するのが最適かとか，諸説紛々でしたが最終的には読者の評価を吸い上げるのが一番，ということに落ち着いたのです．逆に言うと，読者がどう評価しているかマニュアル作成側では誰も知らなかったのです．もっと正確に言えば，マニュアル作成者個人個人はごく限られた範囲で読者の評価を知っているが，統一された評価として世間に認識されてはいない，ということです．マニュアルの品質を全体としてレベルアップさせるためには，読者を知る方法の確立は急務の課題なのです．

読者を知る方法として，アンケートはがきをマニュアルの巻末につけることが従来から行なわれています．この方法は景品付きの場合以外は回収率が低く，またその内容にも偏りがあり，あまり有効に使われていないようです．待っていても仕方がないので直接読者にインタビューする手もあります．出荷台数の少ない商品なら良いかもしれませんが，数千台も出回るものでは時間ばかりかかってしまいます．

商品が出荷される前にマニュアルテストと同時に読者の反応をつかむことも試みられています．これはメーカ側で想定した読者（主に社員）に実際にマニュアルを見ながら商品を使ってもらい，その読者の行動を観察して読者の反応を知る，というものです．この方法は心理学者などがよく試みています．この方法はかなり手の込んだものでアカデミックな分析をするのに適しているように思います．しかし想定した読者の数に限りがあり，想定した読者の全体像をつかむのにはかなりの時間が必要になります．

以上のように，短時間で読者を知ることは難しく，やはり日ごろのデータの蓄積に勝るものはないようです．ところが読者の指向，欲求，関心が時間とともに移り変わり，せっかく蓄積したデータもうっかりすると時代遅れになりかねません．

最後になりますが，本誌の読者と編集者にお願いがあります．我々の蓄積するデータをいつも新鮮なものにするために，読者に満足してもらえるマニュアルを作るために，マニュアルに対する文句，称賛をもっと声を大にして叫んで欲しいのです．一番効果的なのはコンピュータ雑誌の投稿欄です．私は，目を皿のようにして，毎号各誌の投稿欄を見ています．どこぞのコピーではありませんが，「あなたを知りたい，知らせてください」．

（たかはし つよし　セイコー電子工業(株)）