座　談　会

コンパイラの将来への展望

野口　広[1]　米田　信夫[2]　石井　康雄[3]
藤野　喜一[4]　大駒　誠一[5]　(司会)戸川　隼人[6]

1. 方言と標準語

司会　まずは，すでにある言語とかコンパイラにケチをつけるあたりから始めましょう．一番批判的な米田さんからお願いします．

米田　わたくしは別に批判的なわけではないのですが――．言語には，それぞれ何か工夫があって，それぞれの言語を作った人には大へん使いやすいことばでしょうけれども……(笑)．作った人，乃至はそれと精神的に共鳴できるような人には使いやすいが，一般にはむずかしい．人間にはいろいろあって，どういうことばが使いやすいかというのは，人によって違うかもしれない．将来は教育によって，プログラムに対して統一的な精神構造を持たすようにすれば，一つの言語でいいかもしれないが…….

野口　その点は，わたくしは非常に批判的で，自然言語のいまの状態をみても，こんな，無数に自然言語があること自身非常に無駄ですよね．でも，それが少しも改善されないのは，やはり，人間の宿命か何かでしょうがないような気もする．

米田　ただね，何を話すかによって，言語が本質的に違うということではないかもしれないが，何かを話してるんだから，こういう部分は省略されるのが当然だといったぐあいに，いろいろ方言が出てくることはあると思うんです．

石井　たしかに，人間の場合から類推すると，職業方言とか，身分の方言とか，いろんな方言がある．方言というとぐあいが悪く，術語といえばなんとなく高級な話になりますが，いろんな術語があって，複雑な内容を非常に圧縮して表現できる．それは，術語を使える人にとっては便利なわけです．けれども，何がわかりやすいのか，というのは，コンパイラに関しての教育のレベルで違うし，慣れてくるとだんだん簡単に表現したくなる．自分にとって簡単だと，他人にとってわかりにくい，そういうこともあるんじゃないですかね．

米田　この頃，コンパイラを作るほうで手を抜こうとする考えが出てきているようですが，少ない手間でやるのはいいが，中身は手を抜かずに能率よく計算機が使えるように，人間の知恵をもっとどんどん働かせることが必要です．たとえば，オプチミゼーションなどについて，ちゃんと規則を作るのは大へんむずかしいかもしれないが，どういうときにはどういうオプチミゼーションができるということを指定できるといいと思います．簡単なコンパイラが何でもかんでもスタックを使い，ランタイムにもスタックを使うというのは，能率がよくないわけです．実験用にはそれでいいが，実用のコンパイラを作るにはもっとうまい記述言語を考え，どうしたらオプチミゼーションになるか，などを記述できるものを工夫しなければならない．たとえば FORTRAN の場合，添字付きのものがあって，それに1をたすなんてことが書いてあって，それが左辺にも右辺にも出てくる場合，同じだから，一べんアドレスを作っておけば，左辺にも右辺にも，両方に使える．同じだということを，積極的に言語の中で指定するような言語も考えられないことはないが，そうではなくても，コンパイラのほうはどんなところが重複しているかをちゃんと見てやるような方法がもっとあっていいんじゃないか．

藤野　そういうことをやるのに，プログラムの構造をトポロジカルな性質を使って解析して，オプチイミゼーションをする考え方が大分進んでいるようですが，プログラムの理論みたいなものができれば，もっとうまくいくようになるんじゃないでしょうか．

野口　自然言語と情報処理に使われている言語と並べてみて，いろいろな差違があると思うんです．また，

(1)早稲田大学　(2)学習院大学　(3)日本ソフトウェア(株)
(4)日本電気(株)　(5)慶応義塾大学　(6)航空宇宙技術研究所

それを比較することがいいことかどうかは別の問題だと思いますが，自然言語では，わりにイデオムが重要な役割を果たしていますね．コンパイラ言語でも，そういったものが存在するんですか．

石井 DO ループの DO はイデオムとは違うが，最初あれを見れば，これはイデオムかなと思う人がいるかもしれませんね．

米田 イデオムは，普通，人間の場合にはどこかではやって，ちゃんとした意味の定義がなくても，使われる場所から類推して，こういう意味をここしばらくは持つらしい，というように経験によって，そのイデオムを使う．それは計算機にはむずかしい．夢としてはそういうことがほしいかもしれませんが．だけど，差当りはイデオムの意味の定義ができる，たとえば，プロシージャの名前がつけられる，長いものを短く書く機能がそこにあるわけですね．この頃は，エクステンシブル・ランゲイジとかいって，オペレータの定義ができる．文法構造の定義もちょっと……(笑)，ある程度できたりして，自分なりにことばを変えながら使うことができるような面が出てきている．非常に基本的なものだけあって，あとはそれぞれ勝手に定義して使えと初めから割り切っている言語もあるでしょう．しかし，それがほんとうのイデオムかどうか——．人間は，イデオムを使うときに，あまり，具体的な意味を反省しないで使っていますね．それで，ほんとうのイデオムの意味を記述してくれといわれると困る．

藤野 エクステンシブル・ランゲイジについて，去年の SJCC と一緒にシンポジウムがあったりして大分関心があるようですが，比較的かんたんな基礎言語とそれを拡張してやるために使うメタ言語を処理する拡張メカニズムとから成り立つのが一般的な構成のようです．基礎言語がうまくできていて，拡張メカニズムがやさしく使えるのができれば，ユーザーの希望にぴったりしたパーソナルコンパイラといったようなものを自分で作れるようになるかもしれませんね．コンパイラ・コンパイラなども一種のエクステンシブルランゲイジとみることもできるのではないかと思います．

2. コンパイラとは，そもそも何か？

野口 コンパイラというのは，人間にわかりやすい言語があって，それを，機械がわかるマシン・ランゲイジに移す，一種の変換であると思っていいのでしょうか．

石井 ぼくは，いまの人工言語というのは，そもそも機械向きの言語だと思う．人間は非常にパターン認識の能力があるから，機械にわかるものだったら原理的には人間にもわかる．紙テープにパンチしてあるマシンコードをみて，スーッと読める男がいます．ただ，そのわかり方にあまり負担のかからないのがいまのコンパイラ言語ですね．つまり，いまのソース言語はやはり機械にわかる，機械向きの言語じゃないかと思います．

米田 コンパイラの目標は，機械語に翻訳することである，というのはせまい意味ではそうですが，広くいえば，外側では人間に近いようなことばを受け入れて，あとはそれが意図しているような，計算なりをやってしまうところまで含めて考えてもいいんでしょうね．コンパイラというよりむしろプロセッサかな．

野口 エデイタとは違うんでしょうね．本を作る会社で，外国では，エディタとコンパイラとライタがいるみたいですね．コンパイラは中身まではいじらない．カッコよく並べる．

石井 アッセンブルというのは集めるんですね．

大駒 JIS ではコンパイラのことを翻訳ルーチンといっていますから，野口さんのいわれたようなことだと思います．コンパイラ言語を何か別の言語に置き換えるのが翻訳ルーチン，JIS から受ける感じはそうです．

3. 汎用言語は飾りもの？

司会 いままで出た話は，人間がやってもらいたいことをわかってくれることが一つと，それをなるべく能率よくやってくれることが，もう一つ出たわけですね．そうすると，あと，表現できる範囲をどのへんまで取ったらいいかということがあると思うんです．

野口 それは，広ければ広いほど，いいんじゃないかしら．

米田 そうもいえませんよ．

石井 ぼくも，反対ですね．

大駒 汎用言語というのは，重たくなりすぎ，小回りがきかなくて，実用的でないような気がするんですけれど……．PL/I などは何でもできるといっているが，同じことを FORTRAN で書くと，実行時間が何倍か早くなる．遅くて当りまえだと思うんですが．

石井 ぼくもそう思います．ぼんとうに，書くのに時間がかかり，コンパイルに時間がかかり，仕事をするのに時間がかかる．だから，汎用言語は伝家の宝刀です．

石井 いよいよというときにないと困るが……．も

ちろん使いますけれど.

藤野 PL/I は, ユーザがずいぶん長い間, こんな事ができればよい——かければよいと考えてきたような機能を相当に含んでいると思うんです. だからユーザにとってはその機能がいつも全部ほしいわけではないかもしれません. そこで自分の使いたい部分だけが使えるような——言語の分割がうまくいくようになれば, PL/I のような汎用言語はやはり特長が発揮できるようになるんじゃないかと思いますけど.

米田 汎用言語の使われ方は, 汎用言語で書かれたものがある程度はいったら, それが残っていて, 前と同じ環境の中で, ほかの人が使うことができるような……. たとえば, 誰か偉い人が, 自分のまわりの人が仕事をするときには, こういうことばはこういう意味に使う, というような定義をすっかり入れてしまって, あとは, 入れちゃったテキストをこなしたようなコンパイラになっている, そういったかたちの汎用言語なら役に立つ. いろんなローカルな言語をその上に作り得る配慮…….

司会 ローカルな言語が作れて, しかも能率のいいものができなくちゃいけないんですね. それは, できる見通しはあるわけですか.

米田 大へんむずかしいけれども, だんだん作っていくべきであり, ある程度はできると思います. しかし, 何でもできるといっても, ほんとうに何でもということは原理的に不可能だと思います. やはり, 制約があって, この程度のことならば…….

石井 たしかに, アセンブラ言語としても, ほんとうに, 何でも, とはいいにくくなりましたね.

4. 機能追加の必要性

司会 その何でも, というのは, どのへんまであったらいいか……. だいぶ昔, COBOL がまだ無かった頃に, 事務用の言語は, どんな機能をもたせたらいいかわからないと, 誰かが, しきりに聞いていましたね. いまは, それがはっきりしてきた. そのかわり今度, パターン認識なんてことが出てきて, パターン認識には何ができればいいんだ, と.

藤野 使い易い言語というのは, それが対象とする問題の領域にかなり影響されるものだと思います. 何々用言語というときには, その何々という専門領域を十分研究した上で, いいかえれば, 使いこんだ上でないと, よい言語は作れないんじゃないですか.

野口 歴史的にいって, FORTRAN——科学計算用と, いま出ている事務用と称する, その二つの言語およびコンパイラが, 一応安定してきたところへいったわけですが. 順序としては…….

大駒 COBOL はまだ安定しておりませんね. 今でも 6 週間おきに, CODASYL* で会議が開かれて, その度にどんどん変っていっちゃう.

石井 最近 COBOL がどんどんアップデートされるのは, 確かに専門家の立場からすると, そのとおりだと思うのですが, 一般大衆からすると COBOL 65 はおろか, 61 か 62 か……そのへんで手一杯でついていけない.

大駒 でも, 計算機が進歩すれば, 当然, 言語もそれにつれて変っていかなければならないので, いまの JIS FORTRAN は少しずつ追加が必要なんじゃないかなと思っています. JIS 3000 とか, 小さいレベルは変えなくてもいいと思うのですが. 上のほうのレベルでは, たとえば, 非同期処理ができるような機能が入ってもいいんじゃないですが.

石井 ハードウェアと人間との間にコンパイラが入っている. 人間の思想をうまく表わしたいという要求と, せっかくハードウェアをよくしたんだから, これを, フルに使いたいという要求を, どこでマッチさせるか. たしかに, JIS FORTRAN でも, ランダムの処理などを考えないと, これからは困るでしょうね.

米田 何か, 画期的に違う計算機に対しては, 言語の性質もまた違ってき得る気もしますね. 計算機に割り込みの能力があるとか, パラレル処理の能力があるとかいうと, それをうまく使うために, そのための表現がいるようになる. やはり, いまは人間が計算機に何かやらせたいと思うときに, 計算機は何でもできるんだと思って頼むのではなくて, 計算機には, このくらいできるらしいから, そのときには, こういって頼もう, と相手を踏まえた表現をしているわけだから.

石井 ある言語が普及するかどうか, ということは金と人と時間をいかに投入するか, メーカが力を入れているかどうかが, 勝負の決め手になっていた. メーカというのは, 自分のコンピュータを売りたいために, ヒョッとすると, 自社のコンピュータのこういう機能が使えるから自社の言語が良い, それが, 実はとてもいいものなんだと思わせているかも知れませんね.

5. CAI 用言語

* Conference On Data System Languages—データ組織言語会議

司会 この間，藤野さんから，CAIのための言語というお話が出ましたね．

石井 たとえば，IBMでいうと，コースライタみたいな言語は，教育に経験のある人が，その言語を習えば，教育しようと思うことを，わりと簡明に書ける．

藤野 数学にしろ，物理にしろ，人間は手続きを考えますね．まず解析やモデル化があって，その次にアルゴリズムができるでしょう．そのアルゴリズムをさらにコンピュータにわかるようにするところから，コンパイラが必要となるわけですね．そうすると，CAI用言語なんかは，いま，できているコンパイラが登場する一つ前の段階の，手続きとかアルゴリズムというものを表現する言語になるのではないかと思うんです．だから，汎用言語よりも，もうちょっと人間側に近いんじゃないか．

米田 手続きとか，アルゴリズムとかは，相当，コンパイラの側に覚えられていて，何々といわれたときにはこうという手続きを送り出せばいいという仕掛けですね．

藤野 そうなれば専門向き言語とか何とかになるのではないか．これからコンピュータが普及すればするほど，そういう言語が必要になるのではないか．それの一つとしてCAI用言語なんてものがあるんでしょうね．

野口 CAI以外の，たとえば，FORTRAN, COBOLなど，われわれはエグゼキューションのデータを見てどうこういうんですが，CAIのアウトプットは人間が反応を起こすんです．こういうことをやりなさいとか，何とか，そういうものを入れる．そうして出てきたものは，一種の教科課程になるわけでしょう．だから，DOとか，そういう命令だけではたりない．人間の反応に直接関係する言語だから，だいぶ違うんじゃないかと思うんです．

米田 そこの問題点は，人間の最後の反応は，そのプログラムが動いたときのDOのかたちのものなんでしょう．

野口 ええ．CAIのアウトプットとは，要するに練習帳か教科の予習帳か，そういった類のものが，パッと出てくるような言語なんじゃないか．ここまでやって，お前は成績が悪いから10ページ前に戻れ，もう1回定義を勉強せよ，という命令が出るんですね．そのときに，いままでどおりの冷たい命令一本やりのものが出てきたら，それでトレーニングされる学生が，非常に冷たい感じを受けてしまって……．

石井 CAIは，相手が小学生のこともあるし，大学生もあり，いろんなレベルがありますね．野口先生がおっしゃるのは，“Try 何とか”，とか“Do you understand?”なんていわないで，“Can you try——”と，やわらかく書いたらどうかということですね．

野口 具体的にいえばそういうことです．

米田 CAI用言語で大切なことは，計算機をどう動かすかではなくて，相手にどういうことを教えたいかを記述することでしょう．そうすると，たとえば，先生がどうしろといっても，計算機はそれを少しやわらげたことばにして，向うへ伝える……．

石井 普通のプログラミング言語の中に，コンパイラのソース言語と，コンパイラを書くための記述言語とがあるように，CAIの場合も，IBMのコースライタみたいなものと，コンピュータやCAIターミナルとのコミュニケーションのための言語と二つあると思うんです．いま野口先生がおっしゃったのは，人間向きのところで，どういうコミュニケーション用の言語がいいかということだと思うんです．同じようなことはコンパイラでもありますね．なれてくると，コンパイラのエラー・メッセージは，番号だけ出てくれたほうがいいが，初心者用にはなるべく冗長に出てきたほうがいい．しかも，偉い人に見せるときだけは，丁寧なことばで出てほしい．なんて……(笑)．

6. TSS用言語

大駒 特にTSS用の言語なんかはそうですね．同じメッセージがたくさん出てくるんですけど，途中まで出れば，あとはわかるわけです．だけども，アウトプットとしているうちは仕事ができませんね．

司会 グラフィックなんかも，10分の1ぐらい書いたら，これはくだらないとわかりますが，もう止めろというのが書けないことがある．

石井 クイット機能をもち，しかも，クイットも2種類やれるターミナルがあるんですよ．だから，クイット中にもう一ぺんクイットできる……．人間の場合はクイットなのか，本当のことをいっているか，パターンでわかっちゃうからいいんですね．

米田 人間との連絡のためには，昔のハードウェアだとこうなっているからこう使わざるを得ない，ということが多いんだけれども，だんだんソフトウェアでどうでもできるように変ってきているんじゃないですかね．

大駒 計算機は，ある程度のレベルにいかなければ

だめなんじゃないですか．たとえば，時計がなければTSS なんか絶対にできっこないとか……．

石井 時計といえば，今のコンパイラでは，時計は扱えるんですか．たとえば，このDOグループは5秒以上やったら強引にこういうメッセージをいえなんていうプログラムは組めるんですか．あまり見ないですね．

大駒 そういうランゲイジは見たことがない．

司会 CALL・CLOCK で IF を使うと絶対時間はわかりますが……．

大駒 IF の中に入れて，自分で時計を読んで判断するのではだめで，たとえば，COBOL の SORT 命令は一つで10分も20分もかかる．その間で時間をはかることはできない．

藤野 たとえば，CAI の場合には，ある問題を生徒に解かせる．その時に，打切の条件として30秒以上考えたらもうだめだとか，だけどその間に50人の生徒の 60% 解けたらとか，その二つの条件のどちらかが成り立ったら通過する．ほおうっておいたら何分でもかかるわけです．そういう場合には30秒とか，かけないと困るんです．

米田 CAI でも考えようによっては，制御用のコンピュータで，何かほかで動いているものといっしょに動かす場合には，時計というのは相当必要なものですよね．

石井 CAI システムの場合は OS がターミナルごとに何秒のリプライ・タイムとか，お点が何点とか，みんな勘定していますから一応のコントロールはできている．ターミナル側のほうで，「わたしは，これは持ち時間30秒で解いてみせます」というのは聞いたことはないが，ヒョッとすると，そのうちにそういうのが出るかもしれませんね．

野口 入力がグラフィックになると，将来のコンパイラとしてどうなるんでしょうかね．

米田 グラフィックの場合に問題になり得ることは，タイプライタで打ったら，ディスクリートでどれを打つかキッチリ決めてしまうでしょう．ところが，人間がフリーハンドで書いて機械が受け付けてくれるかというと問題ですね．ちゃんと製図してやるとか，フローチャート自身を，キーボードみたいなもので指示しないと受つけてくれないのではつまらない．人間の場合はお互いに何か話をしながら絵を書く．その絵はあらっぽい絵でもわかりますね．

野口 そういったコンパイラができたら，あらっぽい絵というのが，実は，コンパイラ言語に匹敵するんじゃないでしょうかね．

米田 そうですね．そういうふうになってくれればいい．

石井 たしかに，人間の表現はイタラティブに変えられる．非常にあらっぽい話をして，実は，といってもうちょっと細かくできる．ところが，今のコンパイラのように，宣言は全部前もってしろ，なんていわれると，全部できないと書けない．

米田 それは，カンバセイショナルな言語で，フィードバックが早くできれば宣言しなくても……．

石井 そうではなくて，非常にラフに，自分はこうしたらいいということをコンピュータに入れる．コンピュータがそれを覚えていてくれて，一部をなおすとだんだん組み立てられて，ほんとうの会話型ができるような表現方法があるといいなと思うんです．人間の思想を茫漠と表現するというのに，自然言語が大へんぐあいがいいんですけどね．

米田 計算機のほうも茫漠としてくれればいいんですがね……(笑)．いま，せめてできることはやれといわれましたが，ここのところがわからないのでできませんというメッセージが返ってくる．そういうことならできるんですね．エラー・メッセージだと思わないで，わかりませんから教えてくださいという注文だ．

7. コンパイラー自動作成の問題点

司会 このへんでコンパイラを作るほうの話を一席やってください．

大駒 いま，一人の人がプログラムを書くのは，アセンブラでもわりと高度な言語でも，設計からドキュメンテーション完了まで入れて，1カ月に数百ステップぐらいだと思いますが．

石井 そうでしょうね．

大駒 コンパイラの長さは，大体数万ステップぐらいですか．

石井 そうですね．PL/I だと10万を超える．

大駒 そうすると数十人年ぐらいになりますか．いままでのコンパイラは，ほとんどアセンブラで作っていたんですが，マンパワーがものすごくかかるのでそれを減らそうという努力がされているわけですが，その手段は大きく二つにわかれていると思います．その一つは，コンパイラを記述する言語を作るということで，いま盛んに試みられている．もう一つは，コンパイラを作るコンパイラ，すなわち，コンパイラ・コンパ

イラというのが考えられているんですが，ほんとうの意味でのコンパイラ・コンパイラというのはなかなかないようです．いま，たいていのメーカはコンパイラ・コンパイラを作るよりも，コンパイラ記述言語を整理するほうに主力をおいているような気がする．

石井　そうですね．

大駒　コンパイラ・コンパイラという言葉はあいまいなまま非常に広い意味で使われていますが，はっきり定義しなければいけないと思うんです．ぼくの定義によると，まず，コンパイラは，さっき，野口さんのいわれたようなものを言うとして，コンパイラ・コンパイラはインプットとして，言語のスペックですか——文法とか，シンタックスとか，セマンティックス——を入れると，出力としてコンパイラが出てくるもの，だと思うんです．そうすると，今のところ，できたといえるものはまずないんじゃないですか．最も簡単にみえることができないんですね．なぜかというといまは文法を定義するのに，われわれは，ALGOL 以来，バッカス・ノーテーションに慣れていますが，あれでやると Context Sensitive のほうはまるっきりだめなんですね．たとえば，FOFTRAN の GO TO 10 という文があったときに，その 10 が FORMAT 文以外のどこかで定義してなければならないといったようなことは，バッカス・ノーテーションでは記述できない．GO TO 10 をジャンプのAとおきかえ，Aをどこかで定義すれば，それをアセンブラでつなげることは簡単なんですが，このような identification もできないのはコンパイラ・コンパイラとはいえないと思っているんです．

米田　そうですね．コンパイラ・コンパイラというときも，コンパイラの中で相当一般的に，普通のコンパイラではこういうものはこういうふうに使うというものは入れておいてしまわないといけない．たとえば identify というのは離れたところで同じものを指す知恵だということを入れておかないとだめでしょうね．この間の萩原さんのやっていらっしゃったものでは，そういうところが少しはいってると思います．ブロック構造まではいってないかな．

大駒　そういう意味では，ちゃんとできるのはなかなかないし，当分できない．できても，普通の手作りのコンパイラには，とても，能率的に及ばないと思う．

8. シンタックスとセマンテイックスの境界はあいまい

大駒　シンタックスとか，セマンティックスという話がありますけれども，去年の夏のコンパイラ自動作成シンポジウムのとき，どこまでがシンタックスで，どこまでがセマンティックスかが問題になった——．たとえば，ALGOL 60 でセマンティックスだったものが，ALGOL 68 ではかなりシンタックスに入ってしまったんだすね．あのシンポジウムでは，シンタックスとセマンテックスの定義は，フォーマルに記述できたらシンタックスである．それ以外で，自然言語で書かなくちゃいけないような部分はセマンティックだという定義にしよう，ということにしたんです…(笑)．

米田　ぼくは，むしろそれは反対のほうです．いまの話は島内さんですね．

大駒　そうです．

野口　米田先生の見解はどうですか．

米田　それは，はっきり分けようとすると，とてもむずかしい話なんですけどね．

大駒　はっきり決まらないから，時々刻々変わって構わないような定義になっている．記述できたら，それはシンタックスという……．

米田　たとえば，何かへんなものを書くと，エラー・メッセージが出てくるが，こう書いたら，こういうエラー・メッセージが出てくるのはシンタックスかセマンテックスか……．その次はコンパイル・タイムにはねられるようなものがありますが，どういうものがはねられるかというのは，シンタックスといえるでしょう．ところでその次に，エクゼキューション・タイムにこういうものが出たらはねられる，というものがありますね．ぼくの感じは，どれだけが書いたものとして合法的であるかというのがシンタックスであって，書いたものがどういうふうに動いていくか，書いたものが何を意味するか（普通の意味）というのがセマンティックスのほうだと思うんです．

ところが，書いたものの，実行が定義されないというのがダイナミックな条件で決ってくるものがあると，そこの部分は書いたものに意味があると思うべきか，ないと思うべきか，はっきり区別できなくなってくるわけです．たとえば，計算をしていてゼロで割ったらこれはいけない．ゼロで割ったらいけないというのはシンタックスかセマンティックスか——．ゼロで割るようなことが起こるプログラムはシンタックスでちゃんと記述できればいいのですが，そういうわけにはいかない．結局何か，あるレベルで切って，そこまでではねるか，はねないかの範ちゅうを決めて，その時はねることにした．はねる範ちゅうで切ったところま

でがシンタックスで，そこからあとはセマンティックス．

大駒 言語が同じでも，作る人によって違ってくるわけですね．

石井 コンパイラ・コンパイラを作るとなると，そのへんのところをはっきりしておかないと困るでしょうね．

9. 現実的なアプローチ

石井 コンパイラ・コンパイラができたとすると，文法を入れると，スッとコンパイラができちゃうものなんでしようけれども，そこまでいかなくて，半製品ができるようなコンパイラ・コンパイラ・ダッシュみたいなものはないんですか．セマンティックスの分野なんかは，人間があとでモディファイするつもりになる……．

大駒 どっちみち，いま完全なものはないわけですから，試作段階でいろんなレベルのものがあり得ると思うんです．

米田 そうですね．汎用のコンパイラ・コンパイラといっても，もとの言語の性格がなんでもいいというわけにはいかないね．

大駒 そうでしょうね．FORTRAN もできて，記号処理言語もできて，というのは大へんだ．コンパイラ・コンパイラが，もし仮りにできたとしても，FORTRAN とか，COBOL のように一般のユーザにどんどん使われることはないのではないかと思うんです．一部のプロだけが使う．

米田 使って，生産性が上がればいいですね．

野口 だけど，それをやらないと，実際に，さつきの話で，数10人年という労力になるというお話ですから……．

石井 早くそういうものができるといいな，と思っているんですが……．

米田 コンパイラ作りのときに，人間が学んできたいろんなテクニックなどが集積されていて，そういうものを適当に結び合わせてくれるコンパイラ・コンパイラがあるといい．コンパイラを作るのに，人間が本質的に，何か考えなければならない部分を機械にやらせるのは，無理ですね．

石井 ちょっと無理ですね．

米田 まとめてしまって，これだけしか使えませんよ，とするならいいが……．

石井 何か，夢みたいな話ですが，ユーザース・プログラムでもいいから，とにかく，手作りで部品を全部作っておいて，やっと集めてアセンブルするとできちゃう，という，本当の意味のアセンブラ——機械工業なんかと同じような——そういう生産手段がプログラミングで取れれば，経験を積めば機械化もできる．やがてデザイン・オートメーションをやり，工場の流れができる．そういうのが，いままでの工業発展の経過だと思うんです．この間，魚木さんも，そういうことを前から思っていた．というんですが，プログラムの部品を考えると，プログラミング・テクニックで類別して，たぶん，数千ぐらいのものがあると思うんです．そうすると，スペックを書くときにこの部品をこういうふうに使いなさい，というのがうまく表現できれば，何も知らない人でも，棚からプログラム・シートをサッと集めて来てコンパイラができちゃう．それまで行けば．プログラミングの機械化ができると思うんですがね．

野口 その人が本当のコンパイラでしょうね(笑)．

米田 その部分品のスペックをすっかり書き尽くすのがむずかしいですね．うっかりすると，前のリスティングそのままが一番いいもので……(笑)．

そういうものがだんだんふえてくるとすると，「あれをやりたいなら，これを使えばいい」という生き辞引きが必要で，「こういう性質を生み出すようなプログラムを寄こせ」というと，既存の中から取り出してくるか，そういうものが必要だと思います．しかし IR の場合でも，頼んでも思ったようなものが出てこないことがある．それと同じように，何がほしいかということをすっかり書きつくせるかどうかわからないから，大体このようなものが出てくる．人間が中身を見ないで，それだけをつなぎ合わせて動くかというと……．

10. 70点コンパイラ

石井 ぼくがときどき考えるのは，自分が作ったプログラムのうち，何%を本当にお客が使っているだろうか，ということです．ヒョッとすると，半分以上も使わないうちに寿命がきてお蔵に入るようなことがあるんじゃないか．だけど，それがないと，もしかすると1万回に1回通ったときにドカンとかいって動かないと大へんだから，そういうところまで苦労して作るわけですが，もし 99% のところでがまんすることができれば納期は早くなるし，安くなる．

大駒 しかし，1億回に1回でも困るようなアプリケーションがあると思うんです．人間衛星を飛ばすよ

うなときに…….

石井 そうですね，ああいうのはリアル・タイムですから．しょうがない．

野口 そういうものでなければ，スペシャル・ケースのブランチの行先は，アスク・ミスター・イシイでいいんじゃないですか……(笑)．コンパイラは気楽にいこうという態度から出発しているから，非常に簡潔な表現に近づいていくでしょうね．だから，使う側としても，いつでも100%動くということは，仮定するほうがちょっと，ぐあいが悪いんじゃないでしょうか．

石井 いい意見ですね．

野口 シェークスピアだって，何をいっているかわからない英文だってあると思うんですよ．そこいらはがまんしていただいて，とにかく，普通に使ったら，70% ぐらいいくということで…….

米田 そうすると，計算機のメーカとしては，出荷するときに保険をつけて……(笑)，この計算機においてソフトのために変なことが起こったら損害をどのくらい払います，と……(笑)．

野口 結局，作る人の費用と保険の費用がバランスすればいいんじゃないですか．

石井 そうなると，品質管理なんていう概念が出てくるかもしれませんね．過剰品質はよくないとか…….

米田 いまは，計算機は間違えないつもりで使うのが普通なんだけれども，もう少し信用を落として，そのかわりコストを安くする．

石井 いまのコンパイラを作ってデバッグするときは，全数検査を目標にして検査すればいいが，それはできない？

大駒 ちょっとできませんね．ユーザが思いもかけない組み合わせを使ったときに，変な結果になることは年中ありますね．

野口 コンピュータの言語とはそういうものだと，最初から割り切るといいが——いい過ぎかな……(笑)

米田 それは卓見ではあるが，ちょっとすぐには…

大駒 ユーザとしてはやはりいやですね．マニュアルに書いてあるとおりに動いてくれないと困る．

11. 第五世代のソフトウェア

大駒 ところで，計算機は10進法で，もう1桁早くなるんですか．

石井 それは確実になるでしょうね．

大駒 いまは？

石井 50ナノセカンド．

大駒 それを1桁上げると5ナノセカンドでできるんですか．

石井 光のスピードに限界があるからどっちみちだめなんですが，プロセッサの数をどんどんふやしていく．淵さんがこの間いってたんですが，OS の国際シンポジウムとかいうのに出たら，第五世代のソフトウェアでは，プログラムを全部バラに分けてしまってやるようなソフトウェア・システムができるかも知れない．もし，できるとすると，やはりプロセッサの数をふやせば得だということがあるかもしれない．プロセッサを256倍にするのも，LSI を使えば確実に見込みがありますね．もっとも ILLIAC IV は問題があるといわれていますが．

米田 機械の能力を分けたときに，単純に増すのならいいが，何か特殊なことをして増してくる場合にはユーザのほうで，あれとこれを組み合わせれば早くなるなど，ということは考えたくないですね．

石井 それはそうですね．

米田 そうすると，ソフトウェアとしては，それをどういうふうに割り振ったら早くなるかを考えなければなりませんね．

大駒 それをハードがやってくれないとかなわない．

米田 計算機全体が早くなってくれば，オプチミゼーションの問題にしろ，全体のバランスの問題で，計算機が2倍早くなったから，それをソフトの負担軽減にまわすか，それともユーザのほうの能率向上にまわすかで，ある程度はハードの向上をソフトが食っちゃっていいと思うんです．タクシーの値上げみたいなものですが．

野口 大体人間の能力はナノセカンドなんて動いていない．スピート・アップは結構だが人間の思考のスピードに合わせる．間に合うくらいのところまでくれば，あとは，余力はソフトウェア・グループの過剰負担を軽減するところに持っていくべきでしょうね．

石井 だけどぼくは，コンピュータというのは遅いなアと常に思うんですよ(笑)．たとえば，ものすごく早くなったといっていますね．数年間で10倍ずつ早くなる——．しかし，いまから10年前は WRITE の実行は1命令で1行かけたんです．いまは WRITE と書けば1,000ステップ走ってはじめてチョコッと書ける．CALL とやると，以前はジャンプで飛べたが，いまは SVC で，何千ステップ走るかよくわからない．いままでは1命令でいったのが，今はすぐ数千命令走

る．コンピュータはものすごく速くなったはずなのにちっとも速くない．それを考えると，余程速くないとだめなんじゃないか．

大駒 ところで，いまある大容量記憶装置はみんな物理的に動くものでなきゃいけませんね．ディスクでも，データセルでも，磁気テープでも，大体動いてヘッドの下へいかないと読めない．

石井 レーザ・メモリーというのを，去年，SJCCかどこかで，発表した会社がありました．

大駒 容量は……．

石井 数億バイトですね．ただレーザ・メモリーですから，一ぺん書いたら，もうアウトです．固定メモリーになっちゃうんです．

石井 レーザだから非常に分解能がいい．そして書くときには1の光でスッと書くわけです．小さい穴があきます．それを2分の1のもので読むと穴があいてるかどうかが検出できる．そんな原理でやるらしいんですよ．

米田 その穴が，時間とともにだんだん小さくなって，しばらくたてば見えなくなるといいが……(笑)．

司会 でも，そういうのができたら，また，コンパイラ・コンパイラを作りなおさなければいけませんね．一度書いたらもう書けない．

大駒 何回でも書きなおせるのが，われわれの記憶装置ですからね．

石井 昔の紙テープにあけたらだめだというのと同じだ．

米田 セマンティックスの部分を，現在の計算機のことばに密接して記述していると，計算機のほうでやることが違ったら……．

石井 そんな画期的なメモリーが本当に実用化されたら，それをどうするかを考えないと，スワッピングなんかのイメージが全然変わってしまいますね．

大駒 そうですね．スワップするたびに，捨てていたらかなわない．

石井 もう一つは，CPU のテーブル・ハンドリング機能が，もし非常にうまく作れれば，コンパイラの方も相当よくなるわけでしょう．ところが，いまはハード屋さんが考えるテーブル・ハンドリング機能は，一生懸命作っても，コンパイラから見ると，タイトにでき過ぎていてうまくない．そのへんがうまくできると，OS のスピードなんかは違うと思うんです．

12. 現代数学とコンパイラ

司会 きょうは，数学の先生がお二人おいでですから，コンパイラ・コンパイラとか，コンピュータとかを数学のほうから見ると，数学の現代理論が使えるのかどうか……．

米田 ぼくは，数学とコンピュータとは別のものだと思っている．数学的な考え方というのは両方に使えるが，数学の理論が直接こっちに応用できるように数学ができているとは思わないし，また逆に，コンピュータでこういうことができたから数学に役に立つというのも，まだまだということで，あまり連絡は取れないと思う．

藤野 数学とコンピュータが違うのだとすれば，いわゆる，応用ということは当然考えられますね．

米田 数値計算のための数学といった方面では立派なものもあるでしょう．だけど，数値計算のためのものは数学ではないという考え方があったり，逆に数値計算のための本を読んでいますと，さっきの 70% じゃないが，うまくいかない場合の処理ができていない．

あまり計算機が万能だと思うからいけないので，アルゴリズムの前提条件が満たされているかどうかぐらいは，人間のほうでチェックしてからにしろ，ということかもしれませんよ．

野口 もう一つ，数学とコンピュータということについて考える点は，従来の数学というのは大体，力学と物理学に非常に影響と源とを持っているんですね．インスピレーションをそこから相当受けている．たとえば，流体のある問題は偏微分方程式が解ければよい．またその解は物理的実験で多少の見当はつくなど．

これから，70年代になって，また，数学も，新しい発展をしなければならないでしょうし，従来通りの発展も，もちろんあるべきだと思うんですけれども，ここでコンピュータを通して，生物学とか心理学とか物理学以外のサイエンスの分野が，数学発展の原動力になってくれるんじゃないかな．だから，そこで対象になるのは，実数とか複素数ではないかもしれないが，何か新しい対象と方法が数学の中に芽生えてくるのではないか．誇張していうと，それがサイエンスの窮極目的である，宇宙とは何であるか，人間とは何であるか，という問に対する数学的な方向からの一つのアプローチを開くのではないかという見方もあると思うんです．

大駒 いい結論ですね．

司会 どうもありがとうございました．