[redacted]

1986 Lisp and Functional Programming 会議報告(その2)

当協会　システム担当

Session 7

① Connection Graphs

Alan Bawden(Massachusetts Institute of Technology)

プログラミング言語を開発する際に，適切な抽象計算モデルを選択することが重要であるという観点から，共有メモリーを持たない局所的演算を特徴とする並列計算機の抽象モデルとして connection graph grammer を提案している。Connection graph は，電子回路のようなラベル付き有向グラフである。このグラフ言語を生成する文法規則は methocl と呼ばれ，ある部分グラフを別の部分グラフに変換する。connection graph を並列計算モデルとして使う場合，connection graph grammar を変換機構として使用する。さらに methocl の左辺の部分グラフは，2つの節点（並列計算機を構成するプロセッサ要素に対応）を1本の有向弧で結んだグラフに制限した2節点 methocl により connection graph を変換し，並列計算プロセスを表現する。connection graph grammer は，グラフ・リダクション計算モデルに似ている。グラフ・リダクション計算モデルでは，任意の要素と通信可能だが，connection graph grammar はその通信も局所化している。connection graph grammar は Connection Machine のような massibly parallel computer のプログラミング・ツールとして作られた。現在のインプリメントの状況は，connection graph プログラミング言語のためのプロトタイプ・コンパイラを Symbolics 上にインプリメントした。Connection Machine のコード生成は現在開発中である。さらに，この言語で，並列プログクションシステムのプログラムを開発中である。

② Implementing Functional Languages in the Categorical Abstract Machine

Michel Mauny, Ascander Suarez (INRIA)

関数型言語の効率的なインプリメントのため，Categorical Abstract Machine(CAM) 上で関数型言語をインプリメントした報告である。CAM は，typed 入計算項をカテゴリカル・コンビネータへの変換することを基礎とした仮想マシンである。関数型プログラミング言語のセマンテックスを定め，そのセマンテックスに従い関数型プログラミング言語をCAMのコードへコンパイルする。さらに，CAMへの最適コード生成法についても報告している。この方法を INRIA で現在開発しているML言語をインプリメントするために使った。すなわち，ML言語を抽象マシンであるCAMのコードへコンパイルする。ML言語の厳密なインプリメントは2番目の著者により行われており，本報告はML言語の lagy 版である。

③ Connection Machine LISP : Fine-Grained Parallel Symbolic Processing

Guy L. Steele, Jr., W. Daniel Hillis (Thinking Machines Corporation)

Connection Machine Lisp(Cm Lisp) は，微細

12

(fine-grained) な，データ指向的な並列記号処理のための Lisp である。この並列処理は，Connection Machine や NON-VON のような微細構造並列計算機により実現できる。Cm Lisp では，Xapping と呼ぶ新しいデータ構造が導入され，さらにこのデータ構造に関数を並列的に作用させるためのα記法や，演算子βなどが導入されている。Xapping とは，<index>→<value>の型の順序対の集合（関数の集合表現）である。index はプロセッシング要素（PE）に，value はPEのメモリーの値に対応させて考えてもよい。このXapping の各要素に対する関数評価を並列に実行させるために，αと∘記法が導入されている。αは関数の並列適用を示し，∘はαの効果を消すための記法である。αは関数の Xapping や定数 Xapping を生成する。βリダクションは，並列分散されたデータを指定された関数に従い評価し1つに統合したり，データの置換を行ったりする。論文では，これらの記法やβリダクションを使った基本的な関数例が与えられている。Cm Lisp のサブセットの metacircular インタプリタが2つ提案されている。1つは簡易型であり，αをそれ自身により定義したため，その基本的性質が不明瞭になったが，もう1つのインタプリタはより大きいが，これらの基本的性質をうまく表現していると述べている。さらに，異なる index 間で互いにデータを参照しながら関数を実行するとき，同期をとらなければならない場合がある。関数評価の同期性を言語表現の中にどうとり入れるべきかが未解決の重要な問題として提示されている。Cm Lisp と他の言語との比較が行われており，Cm Lisp は，APL, FP, NIALにおける多くのアイデアとイディオムを利用しているが，Cm Lisp の基本的なデータ構造である Xapping が状態をもち，各要素が Lisp object である点がこれらの言語と異なる。Cm Lisp はデータ構造に関する並列性で，SIMD 向きである。一方，QLAMBDA, Multi Lisp は制御構造に関する並列性で，MIMD 向きである。Cm Lisp のインプリメントの状況は，3種類の Xapping (constant, universal, lazy) をサポートするインタプリタを作成し，さらに Connection Machine でのインプリメントは計画されてはいるが，それがいつになるかは明言できないと述べている。

Session 8

Panel : Object Oriented Programming in Lisp

8月5日午後3時45分より2時間に渡って

- オブジェクト指向プログラミングとは何か？
- オブジェクト指向プログラミングと Lisp の関係は何か？

についてパネル討論が行われた。パネリストは

- L. Peter Deutsch (Xerox Parc and Code-Smith Technology Inc.)…議長，Smalltalk の開発者
- Daniel Bobrow (Xerox)…Loopsc Common Loops の開発者
- Dauid Moon (Symbolics)…Flavors の開発者
- Aean Snyder (HP)…Common Object の開発者
- Gary Drescher (LMI)…Object Lisp の開発者

以下，多少断片的になるが重要と思われる議論と質疑応答について記す。

Deutsch

- オブジェクト指向の重要な点は，データとオペレーションを一体化した概念で設計できること (conceptualization design data and operation together) である。
- オブジェクト指向について通常特徴と言われている点は，
  - abstract polymorphism
    異なるクラスに同じ名前のメソッドを用意でき，使う人はクラス間の相違を気にせず操作を指示できる。
  - parlance
    似たような動きをするオブジェクトを subclass を定義することで簡単に定義できる。
  - linguistic support
    言語がシンタックス的，セマンティックス的にオブジェクト指向のためにどんなサポートをしているか？Smalltalkの場合は全てがオブジェクトという簡単明瞭なサポートである。

さらにオブジェクト指向を Lisp で行うためには次のような問題があると思う。

・ オブジェクト指向で言う message passing と Lisp の polymorphic function の関係

・ データ構造. Lisp の場合データ構造以前のベースとして cons セルがある。

・ オブジェクト指向で言う class 階層という設計上の単位と, Lisp の関数, ファイル, パッケージといった単位の関係

これらについてパネリストに議論してもらいたい。

Drescher

・ オブジェクト指向の重要な点は, 徐々に新らしいオブジェクトを定義できること (incrementally define new object) だ。Object Lisp のオブジェクトは Lisp の closure であり, 継承は nested closure で行う。instance と class に本質的な差はないが, この区別は慣例的には重要である。

Snyder

・オブジェクト指向を Lisp に取り入れるとどこがありがたいか?

・ Common Loops はオブジェクト指向システムのカーネルではあるが, ユーザ言語とは違うのではないか?

・ Deutsch は abstraction や encapsulation はオブジェクト指向システムの考えでは無いと言ったが自分はそうは思わない。

・ generic function とオブジェクト指向システムは全然別物だ。

・ 既存の Lisp のタイプ階層とクラスの関係や, パッケージは問題だと思う。

・ Common Object は実験的プログラミングではなく実用的プログラミングをサポートする。

・ Common Object のオブジェクトは encapsulated pieces of data である。

Moon

・ encapsulation はソフトウェア工学的に重要なツールであるがオブジェクト指向システムではない。

・ generic function は function と似ているが, ボディが一つの function ではなくいくつかのモジュールを混ぜたものとしてコンパイラにより処理される。

・ Flavors の linguistic support としてはメソッドの中からはインスタンス変数を通常のLisp変数のようにアクセスできることがある。

・ Flavors は効率的であり, 既存のライブラリ・モジュールを組み合わせて開発できるようにしている。

Bobrow

・ オブジェクト指向システムとして重要な点は前のパネリストたちの言った通りだ。

・ 単に linguistic support だけでなく残りの全システムとの統合が重要である。例えば Loops の browser のようなものが重要である。

・ Common Loops では generic function によってオブジェクト指向スタイルと手続き的スタイルを統合している。

・ Lisp の既存のタイプ・システムと class を分けるのは拡張性のためによくない。

・ Common Loops にはメタクラスというのがあって, Common Object や Object Lisp のカーネルも取り込めるよう検討した。

・ この拡張性は, スタンダードとして使いたい人と変なる実験を試みる人にも使えるようにする意味で重要だ。

質疑応答

① Common Lisp のタイプ階層とクラス階層を融合する利点と欠点について

・ タイプ階層は表現ではなく動作で分けられるべきだ, Common Object でも融合しようとしたがうまくいかなかった (Snyder)。

・ Object Lisp では融合していない。Common Loops では融合している。これは Object Lisp の弱点だ (Dreschei)。

② message senaing と generic function は違うのではないか?

・ Object Lisp では関数呼び出しと send を分けているが, send はオブジェクトのもっている関数を呼び出すことだ。Common Loops のようにすべてのデータがオブジェクトなら関数

呼び出しのシンタックスでうまくいくだろう。generic function がないのも Object Lisp の弱点だ (Drescher)。

・ generic function はオブジェクト指向だ (Bobrow)。

③ generic function とは overloading のことか，それとも polymorphism のことか？

・ Cmplicit Gnversion である (Snyder)。

・ discriminate function である (Bobrow)。

・ discrimination operation である (Deutsch)。

④ それぞれのオブジェクト・システムはどのようなアプリケーションに向くか？

・ Common Loops は実験的なプログラミングに向く (Drescher)

・ Common Object は security と encapsulation が必要なものに向く (Bobrow)

⑤ method combination にどんな問題があるか？

・ Flavers では考えは良かったがマニュアルにうまく説明できなかったため，使う人もソースを見るしかなかった (Moon)。

・ method combination は複雑である。やりたいことを before や after を使って制度化 (institutionalize) するのは難かしい。Loops や Common Loops のように手続き的にやる方が簡単である。

⑥ 今 Lisp 用に提案されているオブジェクト指向システムには並列プログラミングの雰囲気がない。

⑦ persistent local state はオブジェクト指向に必要か？

⑧ 変更の伝播について

Session 9

① A Protocol for Distributed Reference Counting
Claus-Werner Lermen, Dieter Maurer (Universitat des Saarlandes)

本文では分散環境における参照カウンタのプロトコルについて述べ，これの正しいことを証明する。この方法をどう使って直接修正できるようなオブジェクトを検出できるかについても述べる。プロセッサにまたがって参照をコピーする時と，このようにしてコピーされた参照を削除する時だけメッセージを記録すればよいような実現方法について述べる。このような参照を削除する時に同期が必要である。

② A Semantic Model of Reference Counting and its Abstraction (Detailed Summary)
Paul Hudak (Yale University)

関数型言語におけるオブジェクトの共有（参照カウンタ）を調べることにより単独参照の場合はオブジェクトを直接修正したり，セマンテック・コンパイラによりコンパイル時ガベッジ・コレクション（参照カウントをしないコードの生成）が可能になる。このように参照カウンタを調べることは実用的に意義があるが，これまで形式的取り扱いがなされていない。本文では1階関数型言語のインタプリタにおける参照カウンタの厳密な意味的モデルを示す。さらに有限領域上でのモデルの抽象化と決定可能な推論アルゴリズムを示す。この抽象化の有用性を推論参照カウンタを用いてプログラムを最適化することで示す。

③ Distributed Copying Garbage Collection
Martin Rudalics (Johannes Kepler Universität)

分散リスト処理におけるゴミ回収のアルゴリズムについて述べる。このアルゴリズムはリスト操作，システムのアーキテクチャと通信に関する限り，既存のアルゴリズムより一般的である。この利点は配置したオブジェクトをもう少し長く保持しておくことで可能になる。オブジェクトを断続的に局所的に回収し，不必要に保持されるオブジェクトの量を減らし，実時間で動作するような方法を述べる。

3. LISP STANDARDIZATION MEETING

August 5th
Kresge Auditorium (MIT)
p.m. 8：30—11：00
司 会 Julian Padget (University of Bath)

15

出席者　約50名，アメリカ，フランス，イギリス…日本からは4名のみ

Robert Mathis …ANSI の LISP 標準化委員会の委員長

Guy L. Steele Jr. …Common Lisp the Language の著者

Scott Fahlman …Spice Lisp の開発者

Daniel Bobrow …Common Loops の開発者

Jerome Chailloux …フランスの EnLisp の代表

Masayuki Ida …日本 Lisp 技術専門委員会委員長

その他

ミーティングは Lisp Conference の第2日目の夜8時30分から始められた。司会はイギリスの Bath 大学の Padget。この委員会の開催の背景には，Common Lisp をＩＳＯにすることを考えているアメリカ側と，それに反対しているヨーロッパ勢との確執があり，それをなんとか一本化しようと，あるいは一本化ができなくても，両者の意見交換が図れればという意図で開催された。実際，この conference の初日にはヨーロッパ勢から Lisp の標準化に関する発表が行われ，そのなかで，現在の Common Lisp の問題点をあげ，同時に仕様のレベル分け（レベル0：基礎，レベル1：核，レベル2：実用）が提案されている。しかしながらアメリカ側とすれば，de facto standard になりつつある現在の Common Lisp を徹回する気はなく，反対を唱えているヨーロッパ勢をなんとか説得しようという意図がこのミーティング全体を通して感じられた。

ミーティングはまず Lisp 標準化の活動についての各国からの話で始まった。

・ヨーロッパ…イギリス，フランスでWG活動

・アメリカ　…ANSI の委員会が9／23に開催される

・日本　…ＪＩＳと JEIDA においての活動

この時，日本の井田助教授（青学大）から JEIDA における Common Lisp サブセットWG活動の成果である Common Lisp/Core (CL/core) の紹介があった。この CL/Core はすでに ARPANET 上には送られていて，だれでも参照可能であることも紹介された。またハードコピーも数部その場で配布された。

このあとは，標準化をめぐる各人の自由発言が行われた。正確に復元することは不可能なので断片的に，述べられた意見を記す。

・研究者は一生 Common Lisp にかかわりきるわけではない。
しかし企業にとって標準は非常に重要であり，企業間でよく話合って決めるべし。

・ヨーロッパの意見で Lisp のレベルを3つに分ける理由が知りたい。そんなにおいつめるべきではない，それぞれのレベルに意義はある。
仕様の連続性を考慮すべき。はたしてレベル0，1は必要か？

・われわれ（フランス）は Common Lisp がＩＳＯに決ってもそれに従いたくない。

・ＩＳＯで Lisp を2つ定めたらどうか。
しかしやはり Lisp は1つであるべき，おかしい。

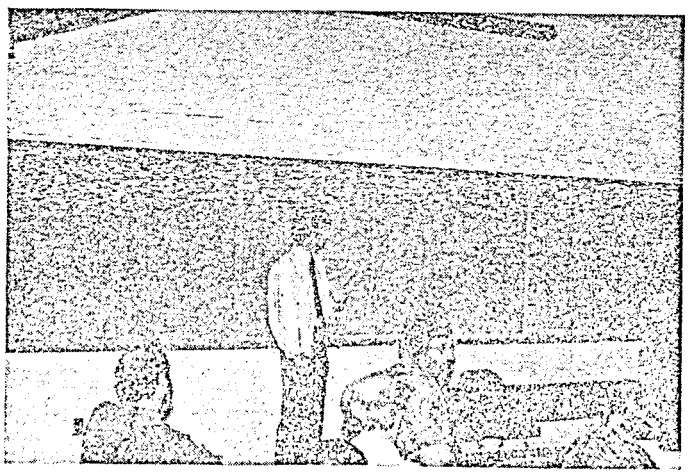

写真　ミーティング

16

・Common Lisp の名前が悪い。
・Common Lisp は DARPA から金が出ている。
・標準化は DARPA のためにやるのではない，技術上そしてユーザのためにやる。
・Common Lisp をＩＳＯにするのはやめて de facto standard にすればよい。これでヨーロッパが救われる。JEIDA 案もあるし。しかし標準は国際的に決めるべきである，国際的商業流通の促進の点からも。
・ＩＳＯが決まればＪＩＳは受け入れる。ヨーロッパ側としては，第2レベルは CL/Core でよいと思う。
・Eu Lisp（ヨーロッパ Lisp）ではレベル1が重要と考えており，これからも検討してゆく。
・自由競争には標準が必要，EuLisp 側はもっと EuLisp の良さをアピールする努力をすべき。
・教育用サブセットとしての標準なら無意味，各々の教師のスタイルがあるから（フランス勢のＩＳＯ化反対が続く…）

結局時間切れ解散となる（終了は11時）。ヨーロッパ勢も Common Lisp には反対していても標準の重要性は勿論良く認識しているし，しかも新しい Lisp の仕様作成，処理系の開発，などの「実績」を武器にした反対でないだけに，アメリカ側の緩急取り混ぜた説得工作に旗色が悪い様子であった。特に，日本などで具体的に Common Lisp を認めた形でのサブセット仕様の検討が行われていることなど，世界の方向は既に大きく変えようのない情勢になっている。これはあの巨大な言語仕様とその処理系を難なく造りあげてしまうアメリカのパワーの成せる業であるという以外に形容のしようがない。

4. Lisp のオブジェクト指向の標準化についての会議

8月6日午後2時より1時間に渡って Lisp のオブジェクト指向の標準化についての会議が開かれた。

Kenneth Kahn（Xerox）を議長とし，Fahlman（CMU），Bobrow（Xerox），Moon（Symbolics），Gabriel（Lucid），Kiczales（Xerox）らが中心となって発言し，Bobrow らの提唱する Common Loops が標準化される見通しとなった。本文では全体の流れを簡単に紹介する。

Fahlman

85年12月の会議からこちら Flavors や New Flavors が Common Loops と合流する動きがあり，これがアメリカの共通の土台となりそうである。特にCMUにおいては Common Loops の理解が深まるとともにこの丘が登るべき正しい丘だと認識されて来ている。私にとって興味深いことは Common Loops の世界と Flavors の世界，そしてその他の世界の間で多くの考えの交流があり，今や一種の標準を作り上げようという気運が高まっている。私はこの動きは前進させられるべきであり，正しい丘に登っているのだと確信している。

Gabriel

Lisp の世界でもオブジェクト指向の考えを取り入れたシステムがいろいろ作られ，ある種の考え方の合意を見ようとしている。機は熟しているように思われ，Common Loops と New Flavors がいい時期に目を開き，2つのグループが両方の考えを取り入れたシステムを作るべく，一緒に仕事をしようとしている。この動きを私はサポートする。

Moon

過去，Symbolics の考えはまだ標準化は早すぎるというものだった。しかし長期的見地からは事情は少し違う。私は今でも2，3年先になると今のシステムよりもっとよい考えが出ると思っている。問題はこのままじっとしてよいシステムが出るのを待つか，5年先はともかく今すでに必要としている人に何かするかである。今の主流となっている Symbolics とか Xerox の人々が作ろうとしているシステムは見かけ上違っていても実は同じモデルのものなのだ。昨日のパネルで Gary が言っていたように Object Lisp は大変異なっているように見えるが，実は Common Loops で作れ

るし，その逆もできるのだ。だから今や互いに協力して標準を作るのが正しい道だと考えている。ただし単に使えるものを作るのではなく，過去の経験を活かして完璧なものになるよう努力しなければならない。

Bobrow

Common Loops が Xerox だけのものではなくなって来たという責任を感じている。皆がエネルギーを注ぐものになったのだ。合流が時期を得て起こった。これは重要なことだ。ＰＣＬについても Xerox 外からもいろんな指摘をしてもらった。われわれの所へ来て励ましてくれた人や貢献してくれた人にお礼が言いたい。

問

Common Loops に対して KEE, ART, Knowledge Craft を作っている人はどう思っているか？

Fahlman

それらの人にはＰＣＬの初期バージョンを見せてコメントを出せる機会を与えた。この仕様がどんな風に役に立つかについては聞けなかった。

問

私の友人にカーネギー・グループの人がいて，その人は Common Lisp の上にシステムを作りたいと思っていてオブジェクト指向の現状を知りたがっていた。彼らのシステムではオブジェクト指向の部分は小さいが，その仕様を標準化するというので驚いていた。

問

KEE, ART, Strobe を作っている人たちはオブジェクト用の部分に大きな時間をさく余裕がない。

問

共通の層（common substrate）を提供するだけでは結局互換性がなくなる。Common Loops において普通のプログラムが知らなければならない部分とシステム・インプリメンタの知らなければならない部分をドキュメントで分けてほしい。

Kiczales

われわれは普通のユーザ用にデフォールト・メタクラスを用意している。単にオブジェクト用の共通の層を用意しているのではない。メタレベルは全く異なるシステムを書く人だけが使う。こんな人は少ないはずだ。色々な異なるオブジェクト・システムがはびこるとは思っていない。

Bobrow

アイデアを持っている多くの人に参加してほしい。エネルギーを集中しよう。

問

この領域ではコンセンサスがまだない。Lisp は20数年も標準化しなかったのに，今 Common Loops のオブジェクト指向を標準化しようとするのは時期早尚ではないか？

Moon

Common Loops はよく考えられたものだ。

問

インプリメンタの意見だけ聞いてユーザの意見を聞いていないので実用的かどうか分からない。

Fahlman

標準化を早くやりすぎることは危険であるが，標準化をしないことも危険である。昨日のパネルでも分かったように多くの人がオブジェクト・システムを使っている。これらの人へのサービスになるはずだ。

問

簡明に書かれたマニュアルがほしい。言葉の定義をまず明確にしてほしい。

Drecher

Common Loops にはメタクラスというのがあって，そのままでも使えるし，explore もできるので，よい Compromise だ。

Moon

私も Common Lisp 標準化のプロセスを学びたい。

問

ＰＣＬを使うことができれば Common Loops が本当に良いかどうか分かる。

Bobrow

もうすぐ第1版が出る。これにフィードバックをかけてもらえれば皆の努力を集中できる。Common Loops を新らしい Common Lisp の標準に入れる予定である。

問

抽象データ型，Smalltalk 型やセマンティック・ネットなどは意味的に近い。Common Loops は Smalltalk 型のオブジェクト指向だが，抽象データ型，セマンティック・ネットとしても使えるのか？

Bobrow

その質問はもっともだ。それにも使えるような仕様にするようにしよう。今の仕様は90%の人の役に立つが，もっと explore が必要である。

問

標準を決めても Lisp はまだ新らしい試みがされているように，オブジェクト・システムで標準を決めても新らしい試みがなくなることはない。

Kiczales

ＰＣＬは直接 Xerox から海外に出すわけにはいかないが，既に広く世界にあるのでそこから取ればよい。

井田

日本でもオブジェクト指向機能は大変興味をもっておりアメリカの努力をフォローしている。Common Loops は one of the best だと思う。

Gabriel

9月23日に Common Loops の新らしいドキュメントができる。

19