Distributed Concurrent LISP

3D－2

風間吉之　阪口哲男　杉本重雄　田畑孝一

図書館情報大学

1. はじめに

Distributed Concurrent LISP（以下DCLと略す）は分散環境における協同処理の記述を指向したプログラミング言語である。DCLの開発は並行処理プログラミング言語であるLexically Scoped Concurrent LISP[1][2](以下LS/CLと略す)に基づき行っている。本稿では、分散環境上でのプロセス間通信方式、LISPオブジェクトの共有方式、プロセスの管理方式を中心にDCLについて述べる。

2. Lexically Scoped Concurrent LISP

LS/CLは多重プロセス機構を備えた、協同処理記述用プログラミング言語である。

LS/CLでは「プロセスはそれだけで完備した能力で形式(form)を評価する実体である」と定義する。LS/CLは逐次処理機能に関してCommon LISPに準拠し、共有変数とメッセージ送受によるプロセス間通信機能、プロセスの動的生成機能等の多重プロセス機構を備えている。

3. Distributed Concurrent LISPの言語仕様

3.1 概要

DCLはLS/CLを拡張し、ネットワークにより相互接続された複数のホスト上のプロセス間における協同処理の記述を可能にしたプログラミング言語である。

3.2 プロセス

DCLにおけるプロセスの定義はLS/CLのものを継承する。DCLにおけるプロセスは以下に挙げる特徴を持つ。

・プロセスはスペシャルフォームstartevalを実行することにより、その実行を行ったホスト上で、もしくは指定したホスト上で動的に起動される。そしてstartevalを実行したプロセスと生成されたプロセスは親子関係を持つ。

・プロセスは親子関係に基づく相対的関係か、もしくは直接各々のプロセスを表すプロセス記述子により特定される。

Distributed Concurrent LISP
Yoshiyuki KAZAMA, Tetsuo SAKAGUCHI,
Shigeo SUGIMOTO, Koichi TABATA
University of Library and Information Science

starteval {[:host *hostname*]
(*form*$_1$ [*procname* {(*var form*$_2$)}*])}+

startevalは*hostname*で表されたホスト上に*form*$_1$の評価を行うプロセスを生成する。:hostによる指定がない場合はstartevalが実行されたホスト上にプロセスが生成される。*procname*が与えられた場合はそれが生成されたプロセスの名前となる。*var*が指定された場合、*form*$_2$を値とする大域変数*var*が生成されたプロセスに与えられる。

他のプロセスとは排他的に式を評価するためのスペシャルフォームとしてDCLではcr (critical region), ccr (conditional critical region)を次のように定義した。

cr [:global | :local] {*form*}+

crは:globalが指定されたときは処理系全体において排他的に*form*を評価する。一方:localが指定されたときはcrを実行したプロセスが属するホストにおいて*form*を排他的に評価する。

ccr [:global | :local] condition {*form*}+

ccrは与えられた*condition*がnilでなくなるのを待ち、排他的に*form*の評価を行う。またcr, ccrにおいて:globalもしくは:localの指定がない場合は、:localが指定されたものと見なす。

3.3 プロセス間通信

DCLではLISPオブジェクトの共有もしくはメッセージ送受によりプロセス間通信を行うことができる。

●LISPオブジェクトの共有

LISPオブジェクトはシンボルやリストなど評価の対象となるデータである。異なるホスト上の複数のプロセスによる協同処理を実現する上で、LISPオブジェクトを共有することは有効である。DCLでは任意のLISPオブジェクトを共有の対象とすることができる。同一ホスト上で働くプロセス間で共有するLISPオブジェクトは、そのホストのメモリ上に格納され、LS/CLより継承した共有変数[1][2]を通してアクセスされる。DCLでは異なるホストの上で働くプロセス間で共有するLISPオブジェクトを格納するために一つのデータ領域を設ける。この領域を共通領域と呼ぶ。

共通領域上のLISPオブジェクトは各ホスト上

のセル、もしくは共通領域上のセルから参照される。一方、共通領域上のセルから各ホスト上のLISPオブジェクトを参照することはできない(図1参照)。DCLではパッケージ名を次のように定義した。共通領域上のシンボルに対するパッケージの名前はDCL処理系全体において唯一に定まる。また、個々のホスト上のメモリに格納されるシンボルに対するパッケージの名前はそのホスト内において唯一に定まる。このことからどのホストからも同一の名称をもって共通領域上の同一のシンボルを指し示すことができる。各ホスト上のプロセスから共通領域上のLISPオブジェクトへの参照は共通領域上の変数(共通変数)を通して行うことが可能である。

●メッセージ送受

メッセージ送受では送信側のプロセスで指定したLISPオブジェクトと等価なLISPオブジェクトを受信側のプロセスに渡す。

同一ホスト上のプロセス同士でメッセージ送受を行う場合、受信側のプロセスにはメッセージの内容として、送信側のプロセスが指定したLISPオブジェクトを表すポインタが渡される。互いに異なるホスト上に存在するプロセスの間においてメッセージ送受を行う場合、送信側のプロセスで指定したLISPオブジェクトの印字表現が送られる。受信側のプロセスには送られた印字表現をもとに再構築されたLISPオブジェクトを表すポインタが渡される。

4. DCL処理系(図2参照)

DCLの処理系はネットワーク上にある複数のホストの上で作動するDCLプロセサ、および処理系を起動したホスト上で作動する共通領域マネージャにより構成される。

・DCLプロセサ: DCLプロセスにおける処理の実行やプロセスの管理·制御を行う。コンパイラ、インタプリタ、仮想マシンよりなる。

DCLプロセサはDCLプロセスを次のように生成する。startevalを実行したプロセスを司るDCLプロセサは、指定されたホスト上のDCLプロセサへリクエストを送る。そしてリクエストを受信したDCLプロセサはプロセスを生成する。:hostパラメタを省略した場合は、startevalを実行したプロセスを司るDCLプロセサ上でプロセスを生成する。

またDCLプロセサはプロセスの排他制御やスケジューリングなどのプロセスの管理·制御を、必要により他のDCLプロセサや共通領域マネージャと協調して行う。

・共通領域マネージャ: あるプロセスが共通領域上のLISPオブジェクトにアクセスするとき、そのプロセスが属するDCLプロセサは共通領域マネージャに対してアクセスリクエストを送る。それに応じて共通領域マネージャは対象となるLISPオブジェクトの操作を行い、レスポンスとして操作結果を返す。

5. おわりに

本稿ではDCLの言語仕様およびその処理系の構成について述べた。これまでにDCL処理系の設計を終え、現在UNIXワークステーション上で実現を進めている。

参考文献

[1] 阪口哲男, ほか: Lexically Scoped Concurrent LISPシステム. プログラミング−言語·基礎·実践−. 情報処理学会研究報告. 1991. p.85-94.

[2] Sugimoto, S. et al: Concurrent LISP Based on Lexical Scope. COMPSAC'89, IEEE Computer Society. p.700-707.

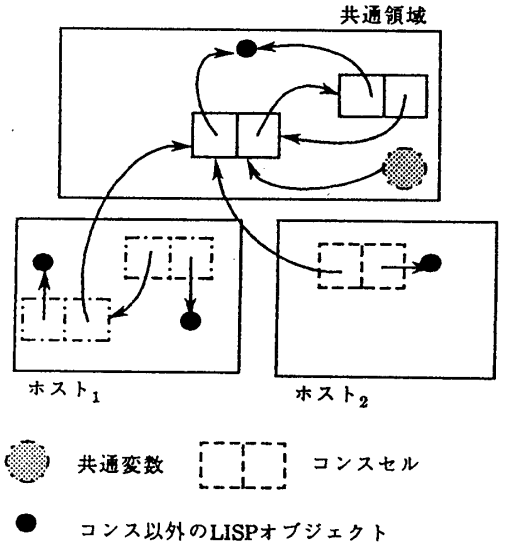

図1 LISPオブジェクトの共有

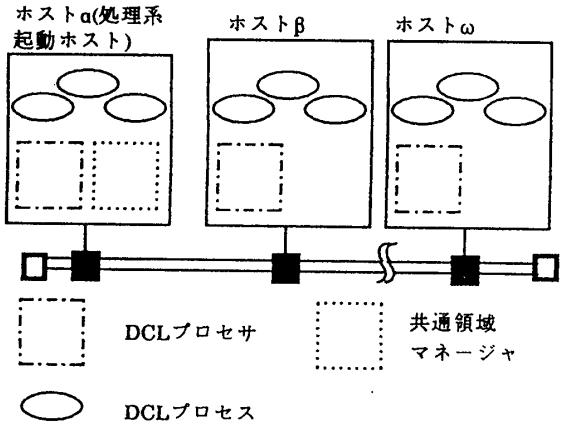

図2 DCL処理系の構成