自律制御用インタプリタ絡繰（からくり）の概要

4D－11

青木　孝

（株）富士通研究所

1. はじめに

絡繰は，自律制御システム（たとえばロボットや自走車）を動かすために当社で開発したインタプリタ言語の名称である．これは，宇宙で作業を行う部分自律型宇宙ロボットの実現に必要な要素技術として開発された．宇宙ロボットは地上の操作者と無線で通信しながら指示に従って作業を行う．地上の操作者と宇宙ロボットをつなぐ通信回線は，時間遅れが数秒あり，容量が小さいという特徴がある．そのような環境で従来の位置指令によるマスタスレーブ動作を行うことは困難である．さらに宇宙ロボットが備えている各種のセンサ情報に基づいて行動させようとすると，地上にセンサ情報を送りその情報に応じた行動の指示を出すことになり，通信負荷と時間遅れの影響を増大させる．この問題に対して，テレプログラミングと呼ばれる技術が米国のペンシルバニア大学で提案されている[2]．これは地上から，位置指令ではなくロボット動作コマンドを送って作業を行うというものである．彼らの提案しているプログラムはコマンドの列であり，計算機言語のようなデータ／手続き抽象化機能や，実行制御機能は持っていない．

宇宙ロボットに限らず，複数の計算機がネットワークで接続され，計算機間で情報をやり取りして何らかの作業を行うばあいにも言語による作業指示が可能であり，有効だと考える．たとえば，プリンタが決められた文字を印刷するだけの時はパラレルポートへのデータ逐一出力だったものが，レーザプリンタという汎用の印刷機の出現によってPostScript™というページ記述言語による出力に変わり，高品位の印刷が可能になった．また最近では，ネットワークを渡り歩いての作業がRPC(Remote Procedure Call)からTelescript™になろうとしている[3]．同様に，通信回線（あるいはネットワーク）で接続されたロボットが非定型作業を自律的に行う場合には，絡繰のような作業記述言語を使って制御されると考えている．さらに，ロボットの実行状況や，センサデータを絡繰のプログラムにして，ロボットから操作者へと送り返すことも可能である．

2. 絡繰の動作環境

図1は当社が開発したロボットシステム（ASTRA）の構成を示している[1]．ユーザはプラナや高位ロボット言語あるいはマスタアームを使って，絡繰のプログラムを作成する．できたプログラムは，ロボットやシミュレータに送る．ロボットコントローラにもシミュレータにも絡繰インタプリタがあり，受け取ったプログラムを解釈実行して，ロボットやシミュレータの絵を動かす．絡繰を使うことで，自律的動作を記述できる．見方をかえると，絡繰は操作者の意図を受けて実際に動作を行うエージェントを作成しているとも言える．図2に従来の

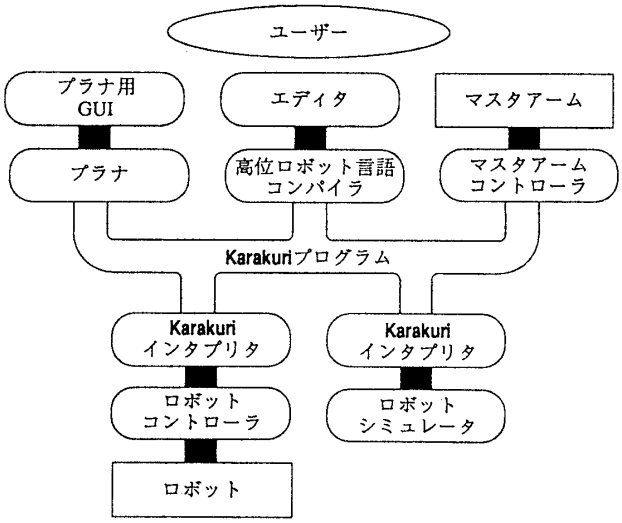

図1　ロボットシステムASTRAの機能構成

Outline of an Interpretive Language 'Karakuri' for Autonomous System Control
Takashi Aoki
Fujitsu Laboratories Ltd.
1015 Kamikodanaka Nakahara-ku, Kawasaki, 211, Japan

状況と，絡繰を使った場合の状況を示した．絡繰エージェントを送り込むことで，ロボットとユーザをつなぐ通信をロボットの動作中ずっと接続状態（オンライン）にする必要がなくなる．また通信量も減り，時間遅れの影響も受けなくなる．

3. 言語仕様

絡繰のシンタックスはPostScript™と同じである．すなわち，逆ポーランド記法で，オペランドスタックを介して関数間の引数のやり取りを行う．変数名や関数名は辞書と呼ばれる表で管理される．組み込みのデータ型は，整数，実数，文字列，配列，手続きである．一般的なPostScript™とは異なり，絡繰の実数は倍精度（64ビット）で表現される．これは，制御のための演算（ロボットの逆キネマティクスやヤコビアンの計算）を倍精度で行うためである．

文字列や配列はヒープ領域に割り付けられ，リファレンスカウントによって管理される．絡繰ではオペランドスタックへのプッシュとポップが頻繁に行われるので，スタックからのリファレンスをいちいちリファレンスカウントに反映していたのでは実行速度が遅くなる．そこで，スタックからのリファレンスはリファレンスカウントに反映しないことにして，そのかわりリファレンスカウントが0になってもすぐには解放せずにゼロカウントリストにつなぐようにしている．ゼロカウントリストはヒープ領域割り付け手順のbにおいてごみ集めの対象となる．ごみ集めの際にはスタックからのリファレンスをチェックしなければならないが，スタックはヒープ領域と比べて小さいので，このチェックは高速に行える．リファレンスカウントだけではヒープ領域の断片化を防げないので，dにおいてヒープの圧縮を行っている．

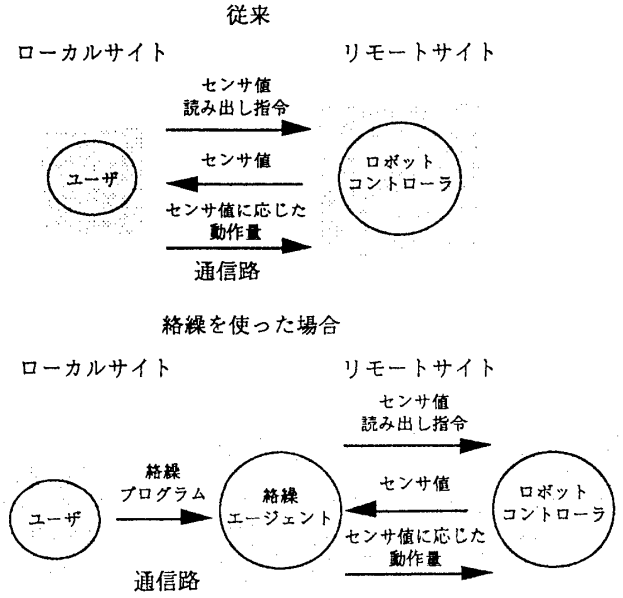

図2　自律動作時の情報のやりとり

［ヒープ領域割り付け手順］

a.フリーリストのなかに十分な大きさの連続領域があればそれを返す．
b.なければごみ集めをする．
c.フリーリストのなかに十分な大きさの連続領域があればそれを返す．
d. なければ圧縮をする．
e.フリーリストのなかに十分な大きさの連続領域があればそれを返す．
f.なければ割り付け失敗．

絡繰ですべてを記述したのでは実行速度が遅い場合があるので，Cのオブジェクトファイル（いわゆる.oファイル）を読み込んで絡繰のオペレータとして登録するダイナミックリンクローダが組み込んである．プログラム中にコード化したオブジェクトファイルを含めて，転送することも可能である．ただし，この場合オブジェクトファイル中の関数が正常に動作するためには，転送先のシステムがオブジェクトの動作環境と一致していなければならない．

4. インプリメント

絡繰インタプリタはすべてCで記述されている．現在動作しているプラットフォームはSunOS4.1.3，vxWorks5.1，IRIX4.0.5の3種類，CPUとしては，SparcとMIPSである．

[参考文献]

[1] A.Ejiri et al, "Satellite Berthing Experiment with a Two-Armed Space Robot", IEEE Int. Conf. on R&A, pp. 3480-3487, 1994.

[2] R. Paul, T. Lindsay and C. Sayers, "Time Delay Insensitive Teleoperation", IEEE/RSJ Int. Conf. on Intelligent Robots and Systems, pp. 247-254, 1992.

[3] "Telescript Technology: The Foundation for the Electronic Marketplace", General Magic White Paper, 1994.