

マルチエージェントによる協調ロボットシステム

3D-7

- ロボット言語の設計 - *

溝口文雄[†] 飯塚圭一[†] 西山裕之[†]東京理科大学 理工学部[‡]

1 はじめに

複数台のロボットが協調して作業を行なう実世界に対応したマルチエージェントシステム (MARS) の構築 [2] において、エージェント間の交渉 [1] が複雑になる。

そこで、エージェント間の交渉を容易に記述可能なロボット言語の設計を行なった。この言語は、マルチエージェントロボットシステムに必要とされる並列性、同期制御を得意とする並列論理型言語 KL1 により設計され、UNIX 上で並列実行可能である。本稿では、片付け問題を対象に、複数台のロボットの協調作業を実現し、本ロボット言語の有効性を実験的に明らかにする。

2 片付け問題

構築するマルチエージェントロボットシステムの対象として環境上の積木を協調して箱に入れる片付け問題を考える。図 1 は、作業環境の様子を示している。

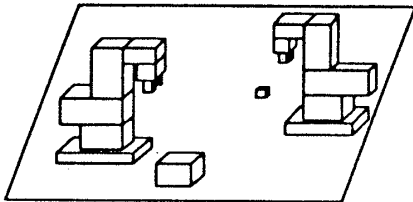


図 1: 片付け問題

各々のロボットは他のロボットの情報（環境情報、稼働状況）を知り得ない。以上のような状況で、積木を複数台のロボットが協調して片付けようとすると、お互いの状況を知り、タスクを要請する必要が生じる。このような交渉は、ロボット台数の増加に伴い複雑になる。

3 ロボット言語の設計

本ロボット言語は、環境監視部、実行制御部、実行計画部が存在し、UNIX 上で並列実行可能である。

*Cooperative Robot System by Multi Agent - Designing of Robot Language -

[†]Fumio MIZOGUCHI, Keiichi IIZUKA, Hironori NISHIYAMA

[‡]Faculty of Sci. and Tech. Science University of Tokyo

3.1 モジュール間の通信、並列性

ロボットのモジュール間、あるいは、ロボット間の通信は、並列言語 KL1 におけるストリーム通信を用いる。

```
main :- get_env(Block)@node(1),  
        plan(Block)@node(2).
```

まず、述語 `get_env` が積木データを獲得した際、`Block` というリストの先頭に獲得したデータを入れると、ストリーム通信により、共有変数 `Block` を保持している述語 `plan` がデータを受けとることができる。また `@node(Node)` により、それぞれの述語は並列実行可能となる。

3.2 作業環境の把握

本ロボット言語では、次の述語により作業環境を把握可能である。

```
get:enviro(Block,Box)
```

環境監視モジュールでは、現在の作業環境が表されているファイルを常に読んでいる。これにより、作業環境に変化があった場合、ストリーム通信により他に知らせ、一度知らせた作業環境情報は蓄えておき、同じデータを送らない。

また、それぞれのロボットの支配領域の環境の変化のみを抽出しロボットに伝えることも可能である。

3.3 ロボットの実行制御

ロボットの制御は、実行制御部により行なわれる。実行制御部では、ロボットを直接制御している C 言語により作成されたプログラムとソケット通信を行なうことでロボットを制御する。

```
connect:robot(Robot_num,Command_List)
```

上の述語により指定したロボットに対して、コマンドリスト `Command_List` の内、先頭から 1 つずつコマンドが処理される。

3.3.1 協調作業

複数台のロボットが協調的な動作をする際、同期制御が必要となる。同期制御を用い、以下のようにして、`r1` から `r2` に受渡し作業が可能である。

```
connect:robot(r1,C1), connect:robot(r2,C2),
C1 = [mp(450,0,300,0,0),ok(OK1),OK2,go,
      mp(400,0,300,0,0),ok(OK3)|C11],
C2 = [mp(400,0,300,0,90),OK1,
      mp(450,0,300,0,90),gc,ok(OK2),OK3|C21],
      :
```

リスト中に"ok(X)"を入れ、その引数とX同じ変数を協調させるロボットのコマンドリストに入れることにより同期を取ることが可能になっている。

3.3.2 オペレータ

本言語では、よりユーザの負担を軽減させるために以下のようなオペレータを用意している。

pickup(X,Y,R): ブロックを取る。

putdown(X,Y,R): ブロックを置く。

release(X,Y): 高さ 20cm で物体を放す。

deliver(R1,R2,[X,Y,Z],OK): R1 から R2 に受渡す。

3.4 実行計画部

ユーザは以上のコマンド・オペレータにより、複数台ロボットを制御可能であるが、動的環境において全てコマンドを与えるのは不可能であり、ロボット自身により実行コマンドを生成する計画作成部を作成する必要がある。この実行計画においては、他のロボットとの交渉が非常に重要となってくる。ユーザは、次の述語の後に実行計画部を記述する。

```
plan(C_L,Block,Box,Out,In,Robot) :-
```

ここで、C_Lは使用するロボット、Blockは積木情報、Boxは箱情報、Outは他のロボットへの依頼、Inは他のロボットからの依頼、Robotは環境内に存在するロボットの座標を表す。また、他のロボットへの依頼は、次のように記述する。

```
Out = [ [R,Req] | Out1 ]
```

ここでRは要請するロボット(r1,r2,...,all)であり、Reqは要請(質問)する内容を表す。同様にして依頼を受けとる(In)ことも可能で、その際Rは要請されたロボット、Reqは要請(質問)された内容を表す。

3.5 動的環境への対応

実行制御部において単にコマンドリストの先頭からコマンドの実行を行っていたのでは、一度決定したコマンドの中止は不可能であり、環境の変化に対応しきれない。そこで、実行制御部ではコマンドリストの先読みを行なうことで、リスト中の異常を感知することが可能となり、環境の変化に対応可能になっている。

```
C_L = [ no(COM_LIST) | C_L_R ]
```

以上のようなコマンドの入力により、直ちに現在の行動を中止し、実行しなかったコマンドをCOM_LISTに入れ送り返し、次の新たなコマンドを待つ。

4 MARS の構築

本言語により、図2のアルゴリズムに基づき4台のロボットによる片付け問題を対象にMARSを作成した。

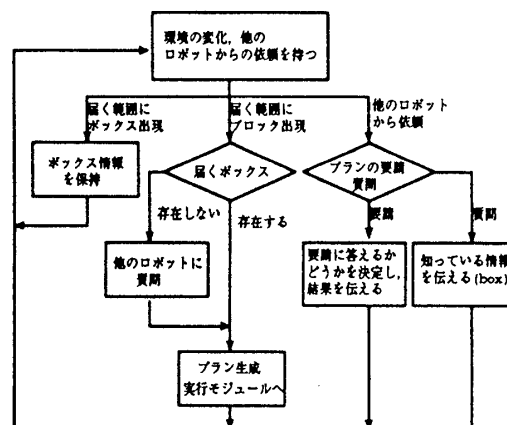


図2: アルゴリズム

今回作成したプログラムでは、環境監視部を1つのプロセッサで、また実行計画部、実行制御部は、ロボット毎に各々プロセッサに割り当て、並列実行を行なった。

これにより、複数台のロボットが協調して積木を箱に片付けることができた。

5 評価

本言語により、実環境で動作するMARSを構築できた。この際ロボット間の交渉は、交渉内容とその交渉の答を対応づけてプログラミングすることで、容易にシステムの構築が可能であり、ロボットシステムのために作成したプログラムは、ロボットの台数に依存しないものであった。

本来、ロボットの台数に従って複雑になる交渉が、本言語を用いることで、ロボットの台数に依存しないシステム並びに、容易に並列実行可能なシステムが構築可能となり、本言語は有効であるといえる。

6 おわりに

本稿では、マルチエージェントロボットシステムを構築する上で、最も困難であるといえる、ロボット間の交渉を容易に行なえるロボット言語を提案した。今後より容易に、より宣言的にシステムを構築可能なロボット言語の開発を行なう必要がある。

参考文献

- [1] Jeffrey S. Rosenschein, "Consenting Agents: Negotiation Mechanisms for Multi-Agent Systems" IJCAI-93, pp.792-799.
- [2] 溝口, 飯塚, 西山, "KLICによるマルチエージェントロボットシステム-実装と評価-", 第9回人工知能学会全国大会, 1995.