知識工学と人工知能 53－5
（１９８７．７．１０）

作業モデルによる動作計画システムの試作

大嶽 能久 *，隅田 敏**，水谷 博之 *

（株）東芝総合研究所 * 情報システム研究所
（株）東芝生産技術研究所**

次世代組立ロボットのプロトタイプ・システムの問題解決機構について述べる．このシステムはある構造物をサンプルとして提示すると，ステレオ・ヴィジョンによってその構造を推定し，その組立て方を計画し，同じ構造のものを組立てる．

問題解決を行うプラニング・モジュールは２つのモジュールによって構成されている．１つは目標構造物の組立て手順を計画するモジュールで，これを作業プラナと呼ぶ．もう１つは各組立て作業の実行方法を計画し，その結果をロボット・プログラムとして出力するモジュールで，これを動作プラナと呼ぶ．

作業プラナは最小拘束アプローチに基いた階層計画システムである．動作プラナの問題解決には作業のモデルと呼ぶ，作業の仕方の知識が使われる．また動作プラナは“何をどうする”といった作業レベルのロボット言語の処理系としての機能を持つものである．

組立て玩具を使って，サンプルと同じ構造の構造物をロボットに自律的に組立てさせる実験を行った．その実行結果を例として同時に示す．

A MOTION PLANNING SYSTEM WITH TASK MODEL

Yoshihisa OHTAKE *, Satoshi SUMIDA** and Hiroyuki MIZUTANI *

* Information Systems Laboratory, R&D Center, TOSHIBA Corporation
1, Komukaitoshiba-cho, Saiwai-ku, Kawasaki, 210, Japan

**Manufacturing Engineering Laboratory, TOSHIBA Corporation
8, Shinsugita-cho, Isogo-ku, Yokohama, 231, Japan

We report the problem solving system used in ARI (Assembly Robot with Intelligence) which is developed in our laboratory as a prototype system of advanced assembly robot system. This robot system is a test case of autonomous assembly robot.

Planning system consists of two modules. One is a task planner which plans the construction sequence of parts. The other is a motion planner which plans the robot's manipulation procedure for each parts' assembly motion. The motion planner outputs complete program for the robot.

Task planner is an hierarchical planing system based on the least commitment approach. Motion planner uses the task model which is a knowledge base for mani-pulation to complete each task. And at the same time, the motion planner has a function of task level robot language interpreter.

We executed an experiment of automatic assembly by this robot system with toy blocks. The results are also presented in this paper.

1. はじめに

組立用ロボットなどへの作業の教示方法は，従来のティーチング・プレイバック方式からロボット言語によるプログラミング方式へと現在移行しつつある．しかしこの言語はロボットの個々のプリミティヴな動作を指示することによって，ロボットの動きを記述するものである．ロボットへの指示を“何をどうする”といった形式で，単にその作業目標だけを指示するものにすることができれば，プログラミングの労力を大幅に省くことができる．更に“これらの部品を使ってこういうものを組立てろ”などのように，単にその目標状態を指示するだけでロボットが自律的にその状態を達成してくれるならば，まさに理想的である．このような理想的なものも１つのテスト・ケースとしては実現可能であることを次世代組立ロボットのプロトタイプ・システムの開発を通して確認することができた．今回はその問題解決機能について述べる．

2. ＡＲＩ

ＡＲＩ（Assembly Robot with Intelligence）は東芝生産技術研究所と東芝総合研究所情報システム研究所との共同研究として開発された，次世代組立ロボットのプロトタイプ・システムである．このシステムは大まかに以下の３つのモジュールから構成されている．

① ステレオ・ヴィジョンによって対象物の構造を推定する視覚モジュール．
② 対象物と同じ構造をした構造物の組立て作業を計画するプラニング・モジュール．
③ ロボットを制御して実際に組立て作業を実行させる制御モジュール．ここでは冗長自由度を有するアームによる障害物回避動作や，位置補正のための視覚フィード・バックや力覚フィード・バック等の制御も行う．

今回１つのテスト・ケースとして，プラスチック製の組立て玩具の長・短２種類のピース数個を使って，提示された構造物と同じ構造のものを自律的に組み上げさせる実験を，幾つかのサンプルについて行った．

3. プラニング・モジュール

プラニング・モジュールは二つのモジュールによって構成されている．一つは目標構造物の組立て手順を計画する，つまり目標構造物を構成している各部品をどのような順序で組付けてゆくかを計画するモジュール．これを仮に作業プラナと呼ぶ．もう一つは各組立て作業の実行方法，つまり部品を組付けるための作業の仕方を計画し，その結果を動作単位の記述によるロボット・プログラムとして出力するモジュール．これを仮に動作プラナと呼ぶ．作業プラナと動作プラナの役割分担についての絶対的な境界線は無いが，動作プラナの問題解決には作業の仕方の知識が必要であり，動作プラナが計画すべき単位作業に余り多くの「目的」を含めてしまうと，その作業の仕方を知識化するのが困難となる．

4．　作業プラナ

このモジュールは最小拘束(least-commitment)アプローチに基いた，階層的計画システム(hierarchical planning system)である．階層的計画法とは大まかな計画からより詳細な計画へと，段階的にトップダウンに計画を精密化させてゆく手法である．これは複雑な問題解決をする場合に，オペレータの組合せの数が指数的に増加する組合せ的爆発を，明らかに不適切な組合せの探索を各階層で枝刈りすることによって回避する手法である．また最小拘束とは副目標間あるいは抽象オペレータ間の予見可能な相互作用を回避する等の十分に正当な理由がないかぎり，それらの間の順序関係に拘束を与えないという手法である．これは不適切な順序付けの修正のための後戻りを少なく，あるいはまったく無くす効果があるばかりでなく，各オペレータ間の並列性が陽に保存されるので，許される資源の容量に応じた並列処理の計画作成が可能となる点で優れたアプローチである．

我々はこのモジュールを同様の思想の基いて開発された他のシステムをベースとして開発した．従って問題空間を“手続きネット”によって表現し，ＴＯＭＥ(Table Of Multiple Effects)を使って副目標間の相互作用を発見する等の手法も継承して採用している．ただし問題解決のための知識を定義する枠組みとして①オペレータ・スキーマ②エクスパンジョン・スキーマ③パターン・スキーマの３種類のスキーマを用いている．以下それぞれについて簡単に説明する．

オペレータ・スキーマ

抽象オペレータを定義するためのスキーマ．その記述内容は，

pos-(neg-)conditions:	成立していなければならない（いてはならない）状態．
add-(del-)effects:	新たに成立する（成立しなくなる）状態．
expansion:	一段下位の階層へ詳細化するための展開規則．
etc.	

対象世界の状態は一階述語形式で，真である正リテラルの連言によって記述される．前提条件(pos-/neg-conditions) は対象世界の状態の記述から真でなければ，またはであってはならない論理式で表される．また効果(add-/del-effects)は対象世界の状態の記述に追加される，または削除されるリテラルで表される．

エクスパンジョン・スキーマ

同一階層内での問題解決において，オペレータ間の順序付だけでは満足されない前提条件を，新たなオペレータを手続ネット中の適切な位置に追加・挿入することによって達成させる方法を定義するためのスキーマ．新たに追加・挿入されるオペレータと，それらのオペレータ間の順序関係及び挿入位置等が記述される．

パターン・スキーマ

オペレータ・スキーマ，エクスパンジョン・スキーマは，それが達成しようとする目標や作用を表すパターン記述によって代表され，それによって検索される．従って１つのパターン記述によって同様の目的・機能を持った幾つかのスキーマを代表させ，状況に応じてそれらを選択し使い分けることができる．そのようなスキーマのグルーピングと，適切なスキーマの選択方法を定義するためのスキーマ．

図４．１に今回の実験に用いたオペレータ・スキーマを例として示す．作業プラナも動作プラナも（全体ではないが）オブジェクト指向でインプリメントされており，オペレータ・スキーマはクラスとして定義され，そのインスタンスが各階層での手続ネットを構成する．クラスメソッドはクラスop-schema から継承される．図４．１のスキーマはピースをワークに組付ける操作を表すもので，変数$lower-pieces,$above-pieces はそれぞれ目標状態で，対象ピース（$piece）の下方にあって隣接するピースと，対象ピースの上方にあってその組付けの障害となるピースである．また$with-piece は部分組立てによって対象ピースに既に組付けられているピースである．インスタンスではパターン・スキーマによってこれらの変数にそれぞれ対応する値が与えられる．また*refはその変数値の参照，*mapは第１項の第２項へのマッピング，*if はLISPのifと同様である．

```
(make-frame 'connect
  (type 'class)
  (isa 'op-schema)
  (pos-conditions '((*map (?x (*ref $lower-pieces)) (assembled ?x))))
  (neg-conditions '((*map (?x (*ref $above-pieces)) (assembled ?x))))
  (add-effects '((assembled (*ref $piece))
                 (*if (*ref $with-piece)
                      (assembled (*ref $with-piece)))))
  (resource 'robot)
  (instance-method (out-com '#'connect-out-com)))
```

図４．１　オペレータ・スキーマ connect

５．　作業プランの実行例

作業プランの実行例として今回の実験に使用したサンプルの一つについてのプラニングの様子を示す．

図５．１が組立ての目標となるサンプル．図５．２は最初の展開によって生成される手続ネットである．

PIECE-G は不安定なため通常

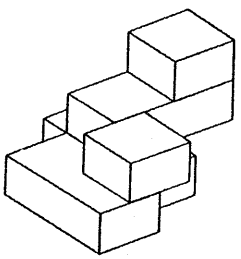

図５．１　目標構造物

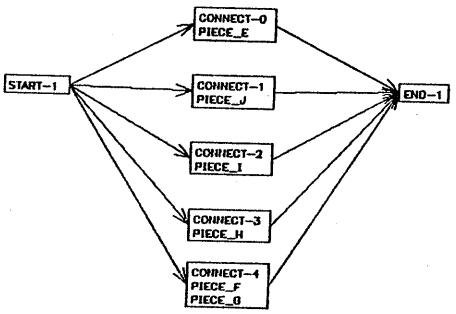

図５．２　初期ネット

の組付けができない．そこでPIECE-F と部分組立てをしてからワークに組付ける操作を行うオペレータCONNECT-4 がパターン・スキーマによって選ばれている．

次にオペレータ間の相互作用が排除され，しかも各オペレータの前提条件が満足されるようような順序付を行って，この階層での問題解決を完了する．その様子を図５．３，図５．４に示す．

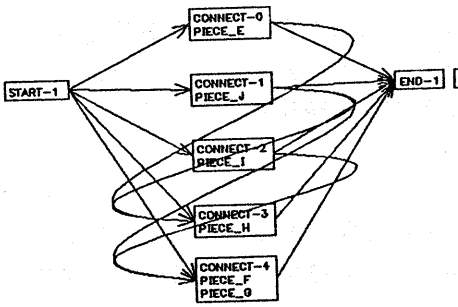

図５．３　相互作用の排除

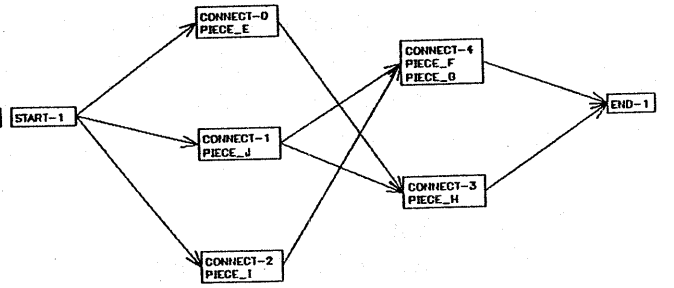

図５．４　最小拘束による順序付の結果

６．　動作プラナ

上記の例での作業プラナの出力を図６．１に示す．リストの第１項は組付け作業の種類を指示する作業名．第２項は組付けるピース．第４項はその組付け位置の４×４行列での記述．第６項は目標位置の領域指定．第８項は使用するアームの指定．第９項以降は部分組立てを指示する記述である．このように動作プラナへの入力は“何をどうする”といった作業の記述である．これを解釈し，ロボットが理解できるレベルの動作指示に翻訳するのが動作プラナである．

```
(ATTACH PIECE_I TO
        #2A((1.0S0 0.0S0 0.0S0 32.0S0) (0.0S0 1.0S0 0.0S0 48.0S0)
            (0.0S0 0.0S0 1.0S0 0.0S0) (0.0S0 0.0S0 0.0S0 1.0S0))
        OF WORK_SP WITH RIGHT)
(ATTACH PIECE_J TO
        #2A((1.0S0 0.0S0 0.0S0 32.0S0) (0.0S0 1.0S0 0.0S0 16.0S0)
            (0.0S0 0.0S0 1.0S0 0.0S0) (0.0S0 0.0S0 0.0S0 1.0S0))
        OF WORK_SP WITH RIGHT)
(ATTACH PIECE_E TO
        #2A((0.0S0 -1.0S0 0.0S0 32.0S0) (1.0S0 0.0S0 0.0S0 0.0S0)
            (0.0S0 0.0S0 1.0S0 0.0S0) (0.0S0 0.0S0 0.0S0 1.0S0))
        OF WORK_SP WITH RIGHT)
(WH-ATTACH1 PIECE_F TO
    #2A((1.0S0 0.0S0 0.0S0 32.0S0) (0.0S0 1.0S0 0.0S0 32.0S0)
        (0.0S0 0.0S0 1.0S0 19.0S0) (0.0S0 0.0S0 0.0S0 1.0S0))
    OF WORK_SP WITH RIGHT THROUGH
    ((PIECE_F TO
            #2A((1.0 0.0 0.0 0.0) (0.0 1.0 0.0 0.0) (0.0 0.0 1.0 0.0)
                (0.0 0.0 0.0 1.0))
            OF AIR_SP)
     (PIECE_G TO
            #2A((1.0 0.0 0.0 32.0) (0.0 1.0 0.0 0.0)
                (0.0 0.0 1.0 19.0) (0.0 0.0 0.0 1.0))
            OF AIR_SP)))
(ATTACH PIECE_H TO
        #2A((1.0S0 0.0S0 0.0S0 16.0S0) (0.0S0 1.0S0 0.0S0 0.0S0)
            (0.0S0 0.0S0 1.0S0 19.0S0) (0.0S0 0.0S0 0.0S0 1.0S0))
        OF WORK_SP WITH RIGHT)
```

図６．１　動作プラナへの入力例

現在実用化されているロボット言語のように，作業内容をロボットの個々の基本動作によって記述する言語を動作レベルの言語と呼ぶことにする．それに対して図６．１に示したような，ある目的を持った一纏まりの動作を単位として記述する言語を作業レベル言語と呼んでいる．従って動作プラナは作業レベル言語の処理系として機能するものである．

作業の指示は個々の対象世界に強く依存するような記述ではなく，多様な解釈が可能であるが，実行環境に応じた適切な解釈が処理系によって生成されるようなものでなければ有用でない．そのため作業レベル言語の処理系は，個々の対象世界に依存する部分が極力分離され，それらを知識として利用するように構成されている必要がある．どのような情報が知識としてまとめられるかを列挙してみると以下のように整理することができる．

① 作業環境内に関する知識．

ロボットの作業環境内に存在する物体（ロボット自身も含む）についての形状や位置の情報を基本とし，ロボットがある動作をした場合にその環境に与える影響・効果を推論するための知識．

② 作業対象物に関する知識．

個々の部品等の操作対象をロボットに適切にマニピュレートさせるための情報．

③ 作業の仕方に関する知識．

作業の仕方の知識，及び作業を実行する各動作に対する意味付を与えるための情報．

④ センサに関する知識．

各種のセンサから得られる信号情報を総合的に解釈・利用する方法に関する知識．

⑤ ロボットに関する知識．

個々のロボットに依存する機能や制約条件などの情報．

以上の各情報を処理系から分離し，しかも相互の独立性の高い形で知識化することが，汎用性の高い作業レベル言語処理系を構成するための最も重要な課題である．我々が今回開発したシステムの特徴は上記③の知識の構成方法にある．

７． 動作プラナの問題解決

作業命令からそれを実行する動作プログラムを生成する方法として，定型作業についてのプログラムのテンプレートを用意しておき，パラメータ化された部分を実行環境の状況に応じて埋めるといった方式が比較的簡単に実現できる．しかしこのような方法では動作パターンの類似したマクロには同じ作業に対してもそれぞれ別のテンプレートを用意しておかなければならないので，実用的な汎用性を得るためには膨大な数のテンプレートを予め用意しておかなければならない．そこで我々は作業の仕方を知識化し，これに基いて作業方法を計画する方式をとっている．つまりある作業からいきなりそれに対応する動作列への対応付けをしようとするのではなく，各種の作業の間に成立する構成的な関係を単位作業について分類・整理し，マクロな単位作業をよりミクロな単位作業に詳細化する，作業の展開規則として作業の仕方を知識化するというアプローチをとった．

従って動作計画のプロセスは生成規則に基いて文を生成するように，作業の展開規則に基いて作業命令から動作命令列が生成される．このような作業の展開規則の集合を“作業のモデル”と呼ぶことにする．

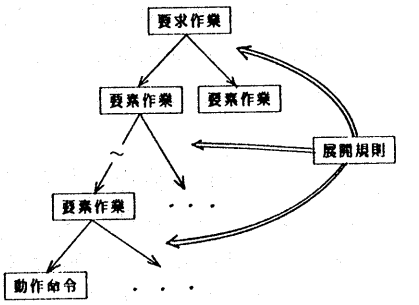

図７．１動作命令の生成過程

このような方式をとることによってモジュール性の高い形で作業の仕方を知識化することができる．しかもこの方法では戦略的に同じものであれば，作業の展開過程のある段階での詳細化の方法を適当に選択することによって，実行環境に応じたミクロな作業方法の変更に柔軟に対応することができる．例として前記の例で用いた，１つのピースをワークに組付ける作業ATTACHを，アームやカメラ系等の各コントローラに対する動作命令に展開したものを図７．２に示す．作業ATTACHはまずLIFT,MOUNTに展開され，LIFTは更にG-APPRO 等に展開され，最終的に四角の枠で囲んだ各動作命令へと展開される．動作プランの結果も図のような手続ネットとして生成されるので，各動作間の並列性が陽に表現されている．

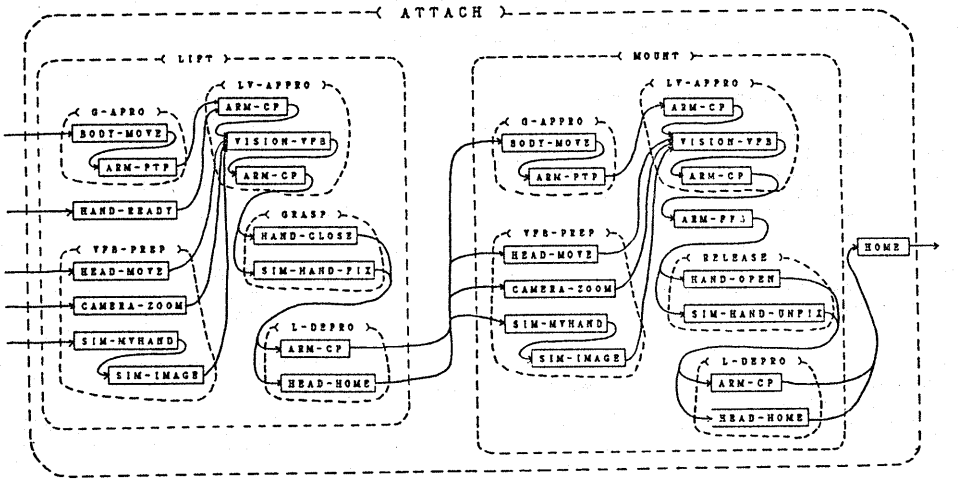

図７．２　作業ATTACHの展開結果

展開によって動作命令列を生成する過程は，同時に展開された各作業にある種の意味付を与えてゆく過程でもある．動作プラナは展開した各単位作業のインスタンスをリンクによって結合した，目標作業を根としそれを実行する動作命令群を葉とする木構造の情報を同時に生成する．これを以下“作業の構造木”と呼ぶ．図７．３は先ほどの作業ATTACHを単純化した別の展開結果による作業の構造木である．

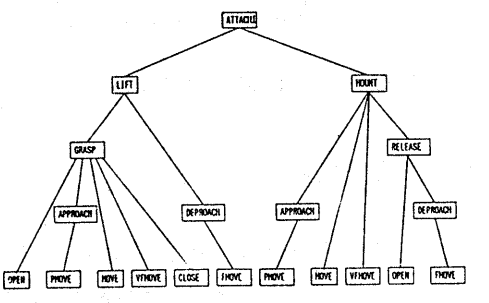

図７．３　作業ATTACHの構造木

このような作業の構造木は現在動作レベルのプログラムを生成するためにだけに使われているが，これは制御モジュールにとっても作業の実行上極めて有益な情報を提供しうるものである．そのことについて以下に簡単に言及しておく．

① センサ情報の高度な解釈

前記④の知識と作業の構造木に基いて，各種のセンサから得られる情報の総合的かつ作業の実行段階に応じた解釈を推論することが可能となる．

② 高度なエラー処理

作業の実行時に何等かのエラーが検出された場合，その時点で実行中であった動作の位置付けを作業の構造木から判断し，それに応じた処理方法を推論することが可能となる．

8． 動作プログラムの生成

動作プログラムの生成については，単に前記の例について生成されるプログラムの一部とそのシミュレーションの結果をそれぞれ図8．1，図8．2に示すにとどめる．前記の例では図8．1に示すような形式で約3000行のプログラムが生成される．図8．2の各直方体はピースを，三角柱はロボットの指を表している．

```
(@ARM-CP-0_1
    (AND (WAIT '@ARM-PTP-0) (WAIT '@HAND-READY-0))
        (LAMBDA (X)
          (START-ARM-CP 'RIGHT
              '(#2A((0.0 -1.0 0.0 111.0) (-1.0 0.0 0.0 -74.0)
                    (0.0 0.0 -1.0 56.3) (0.0 0.0 0.0 1.0))
                PIECE_SP)
              (REFER-BB '(ROBOT ARM-SPEED HIGH))))
)

(@ARM-CP-0
    (AND (WAIT '@ARM-CP-0_1) (WAIT-ARM-MOVE))
        DO-NOTHING
)

(@VISION-VFB-0_1
    (AND (WAIT '@ARM-CP-0) (WAIT '@HEAD-MOVE-0) (WAIT '@CAMERA-ZOOM-0)
         (WAIT '@SIM-IMAGE-0))
        (LAMBDA (X) (START-VFB 'GRASP 'NIL 'RIGHT))
)

(@VISION-VFB-0_2
    (AND (WAIT '@VISION-VFB-0_1) (WAIT-VFB))
        (LAMBDA (X)
          (START-ARM-CP 'RIGHT
              '(#2A((0.0 -1.0 0.0 111.0) (-1.0 0.0 0.0 -74.0)
                    (0.0 0.0 -1.0 56.3) (0.0 0.0 0.0 1.0))
                PIECE_SP)
              (REFER-BB '(ROBOT ARM-SPEED LOW))))
)
```

図8．1 生成される動作プログラムの例

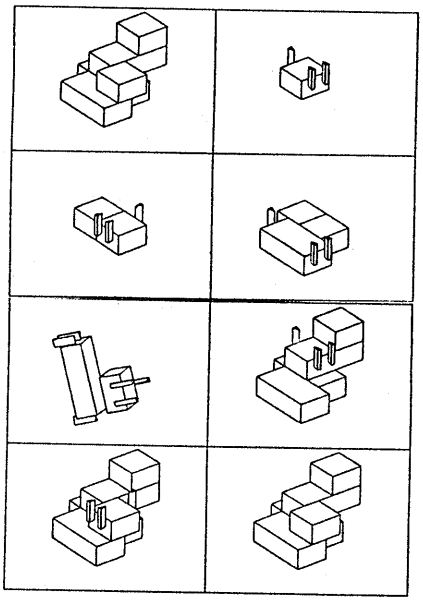

図8．2 シミュレーション結果

9． おわりに

以上我々が開発した次世代組立てロボットのプロトタイプ・システムの問題解決機構について述べた．本システムの開発を通して得られた技術を，今後の人工知能技術を応用したＦＡシステムの開発に反映させてゆくつもりである．