ロボットによる組立作業のための作業スキルの表現とその実装方法 *

長　井　達一郎**
荒　牧　重　登**

The Representation Method and the Programming Technique of Task Skill for Robotic Assembly Task

Tatsuichiro NAGAI** and Shigeto ARAMAKI**

The skill of robotic task has been studied in many research institutes. We have been researching the skill for the assembly of the consumer products. We has developed the original knowledge representation for robotic assembly task with the position and force control. This paper describes representation of a task skill for an assembly robot and programming technique of the system. We propose the representation method of the skill that a human has and the construction method of a knowledge base for robotic assembly task. In this paper, the concept of CRS(Constraint Reduction System) is also introduced for making a robot task program. The CRS is realized by “Prolog” language extended. All the action and primitives are considered as process and can be uniformly described by the reduction rule of the CRS. Actually, we could confirm the effectiveness by making a robot perform peg_in_hole task.

Key Words : Task Skill, Hierarchical Intelligent Control, Landmark, Fine Motion Planning, Constraint Reduction System

1. 緒言

近年，産業用ロボットは工場などで製品の組立や自動車のボディの塗装やウェハーの搬送など様々な作業を行っている．これらの作業の動作教示は，ロボットをティーチング・ペンダントなどのコントローラで直接動かし教示するティーチングプレイバック方式や低レベルロボット言語による教示,また作業環境を仮想空間に構築し，その空間内でロボットの動作の軌道を作成する教示方法が採用されている．これらの教示の手間は膨大であるだけでなく，工業製品の組立作業などは多品種少量生産のため，頻繁に製品がモデルチェンジし教示も変更が必要になり，また１つの製造ラインで数種類の製品の組立を行う場合などは組立動作を製品ごとに対応しなくてはならない．そのため，教示の手間を削減するために，ロボットによる組立作業計画 [1] やその表現技術などのソフトウェア面の研究・支援が求められている．

組立作業をロボットに行わせるためには，作業対象である部品の形状，位置，姿勢，部品間の結合関係などの情報が必要となる．これらの情報取得の方法として，視覚システム [2] などの感覚器やそれを基に作成した環境モデル [3] を用いる方法がある．これは，未知環境での作業や学習には非常に有効である．一方，工業製品の組立の場合，既知環境であるため設計情報から部品等の形状は入手可能であり，作業対象の位置も特定に位置に置かれていると考えられる．

また，組立作業は組付け部品を初期状態から目標状態までの接触状態を遷移させるプロセスとして考えられる．よって，組立作業を接触状態遷移として容易に表現することが可能で，かつ組立作業を効率的に行うために人の持つ組立の作業スキルも表現できるシステムが求められる．組立作業全体を表現する手法として，接触状態遷移ネットワーク [4] などの状態遷移グラフによる表現

* 平成 22 年 2 月 3 日受付
** 電子情報工学科

[5][6][7][8]，実演から接触状態を抽出した研究 [9] やビジョンデータから正しい接触状態とその遷移を推定する研究 [10] がある．これらの状態遷移による動作計画には，ロボットの位置決め誤差や部品の公差などを考慮した，対象への順応機能 (Fine Motion) が必要である．人間は組立作業中の不確定性に対処できる技能 (スキル) を持っている．このスキルに関しては，基本動作 (スキル) の組み合わせで作業を定義する研究 [11] やルールベースとな人工技能システムを構築した研究 [12] などがある．これらの研究では，作業のデータ解析からスキルの作成やシステム構築までを統一的に述べられていない．システムの構築手法については，遠隔操作で作業を教示する際にインピーダンス制御にマッピングする手法でスキルを定義しロボットに自律作業を行わせるシステムの研究 [13] や RT ミドルウェアによる作業構造の実装の研究 [14] などがある．RT ミドルウェアが提供するロボットプログラムの環境は新しく，工場の組立作業に適しているかはさらなる研究が待たれる．

ユーザによる教示を容易にし，あとはシステムが自動的に組立作業を行うことを可能にするため，本研究では，組立対象物の状態とロボットのセンサ値を定性的な値に量子化したものを用いて組立作業の状態空間を定義する．この定性的な値により，制御モードや制御方向を切替える．また，動作手順を組立の目標位置の系列ではなく，他の部品と相互作用し機能を発現する部品の部分である機能素 [15] をランドマーク [16] の系列として定義する．この手法により，力センサと位置センサを用いてロボットの作業状態を精密に監視でき，かつロボットの動作計画 (Fine Motion Planning) が可能となる．これらによって定義される状態空間と動作は知的制御階層構造として定義される．この知識制御階層構造の各レベルは FSA(Finite State Automaton) [17] の直積や連接で定義される．このモデリング手法の有効性を確認するために，論理型言語 Prolog[18] を用いてこれらの試作システムを構築し，実機による実験を行う．さらに，本論文では，知的制御階層構造で定義されるスキルの各レベルをプロセスとみなし拘束条件リダクション法 [19] を導入し，プロセスリダクションによりスキルの実行を試みる．

2. 組立部品と作業スキル

今回対象としている民生機器の組立品は，いくつかの部品から構成される．それぞれの部品は材質や加工法等の属性の他に，組立のために機能を発現する軸 (shaft) や穴 (hole) のような機能素 (Functional element) [20] を持っている．機能素も位置や直径等の属性を持つ．

民生機器の量産品の場合，寸法に公差が付加された部品図とともに作業の種類の用語を用いた組立手順と組立図が，設計情報の一部として与えられる．図 1 (a) のような組立の場合，設計情報として与えられる組立手順は「part b を part a に取り付け (affix(shaft,hole))」程度の指示である (図 1(b))．しかし，実際の人手による組立では，直接 part b を part a に挿入しようとせず，一度穴の近くに突き当て，面上を一定の力で押しつけながら穴にはめ合い，挿入するといった対象物の動作自由度を拘束していくような作業スキルを用いた手順などが考えられる．このように，設計情報として与えられる組立手順は主として人手の組立を前提としており，ロボットの組立手順としては不十分である．そこで，図 1(a) に示すように，approach，work などの，ロボットによる組立作業特有の機能素を追加し，動作計画が容易になるよう図 1(b) を図 1(c) のように詳細化する．

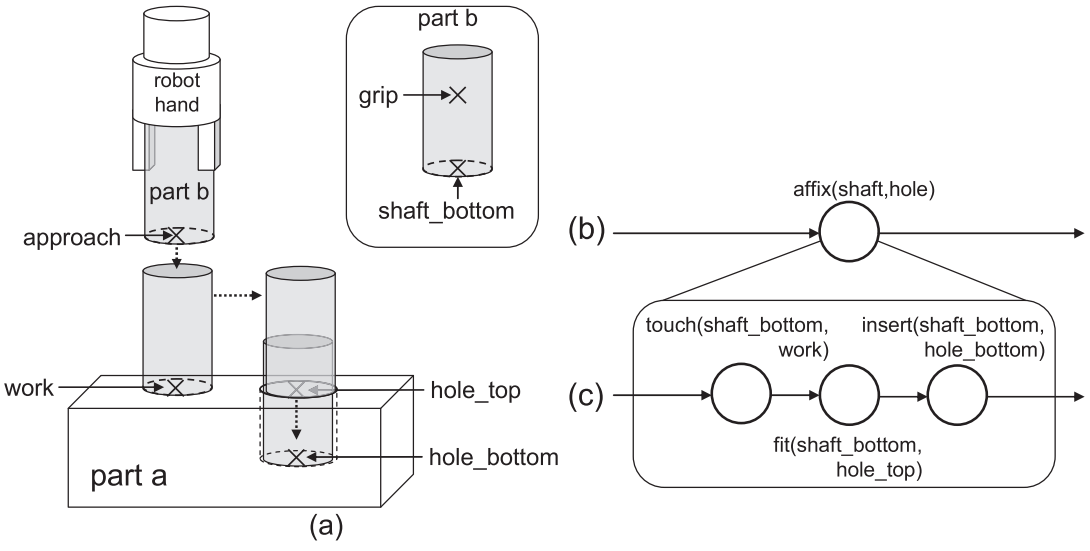


図 1 部品と組立手順

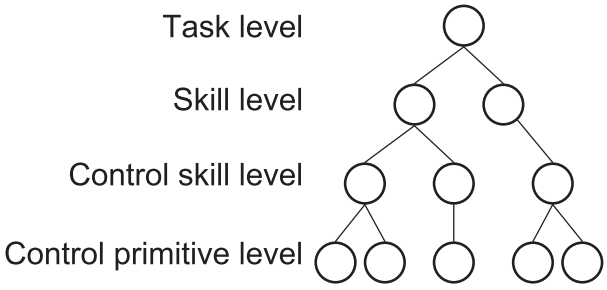


図 2 知的制御階層構造の階層

3. 知的制御階層構造

作業対象の状態空間を位置センサ及び力センサの状態として定性的に表現した FSA を要素とし，これらを用いて構成される動作手順をもとに 4 つの階層を定義する．この階層を知的制御階層構造とよぶ．以下に知的制御階層構造の概略を示す．

タスクレベル：図 1(c) に示すようなロボットによる組立手順をスキルの組合せにより定義する階層．

スキルレベル：制御方向の割り当て，制御モードの設定を行い制御系の枠組を定義する階層．

制御スキルレベル：1 方向について制御系の目標値を設定し分解可能な制御モードの設定を行う階層．

制御プリミティブレベル：フィードバックループを形成し制御系を実行する階層．

知的制御階層構造の概要を図 2 に示す．組立の動作手順はタスクレベルと呼ばれスキルレベルの組合せで定義される．スキルレベルは制御スキルレベルの組合せで，制御スキルは制御プリミティブの組合せでそれぞれ定義される．

3.1. 制御プリミティブレベル

制御プリミティブレベルは，組立作業中のロボットハンドや作業対象物の一方向の位置または力の状態遷移を管理する階層であるロボットハンドもしくは作業対象物の位置及び力の状態を *m*(過大状態), *s*(適正状態), *l*(過小状態) とする．センサ入力を量子化し定性的に *less*(過小入力), *suit*(適正入力), *more*(過大入力) とし，動作の出力を *up*(上昇), *down*(下降), *hold*(維持) とする．また，センサ入力方向と動作方向は同一とする．

例として，図 3(a)に示す任意の1方向D1の位置の状態をsに導くFSAについて説明する．出力をもつFSAは，5つ組S = {K, Σ, Δ, δ, λ}(K: 有限集合，Σ:入力アルファベット，Δ: 出力アルファベット，δ: 出力関数，λ: 遷移関数を表す)で与えられる．ある状態で入力を受け出力をした結果，遷移先は一意に定まらず非決定的であるが，これを部分集合作成及び状態数の最小化[17]を行い，決定的なFSAで表現する．これにより，位置の状態をsに導くFSAはK = {m, s, l, msl} (mslはセンサの不定状態)，Σ = {more,suit,less}，Δ = {up, down, hold} で，δとλは図 3(b)で定義される．既報[21]では，図 3(b)のようなFSAにおいて，s状態からmsl状態を経由せず，直接mやl状態に遷移するリンクがあったが，このリンクは複数のFSAの直積を利用する上でmslとそれ以外の状態が同時に存在するため，動作が安定しないことがしばしば起こった．そこで，本研究では図 3(b)のように，必ず一度msl状態を経由して他の状態に遷移するFSAを提案する．さらに，全ての状態から全てのセンサ入力に対応できるように拡張した．

位置及び力に関して制御プリミティブは表 1 のように分類できる．位置や力の状態を s，m，l 状態に導く

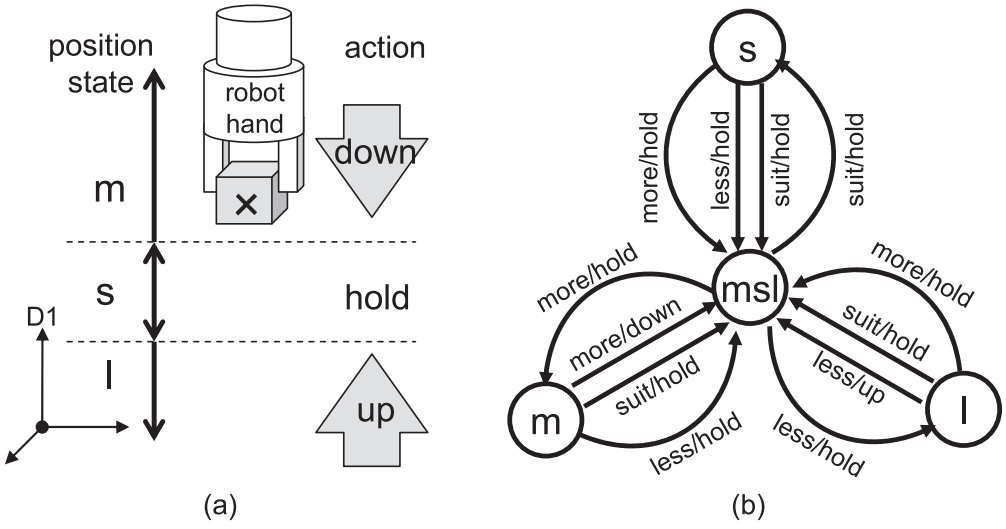


図 3 位置の領域と位置の状態を s に導く FSA

表 1 制御プリミティブの分類

Objective state	Position sensor	Force sensor
s	position_to_s(D1)	force_to_s(D1)
m	position_to_m(D1)	force_to_m(D1)
l	position_to_l(D1)	force_to_l(D1)
None	position_check(D1)	force_check(D1)

プリミティブと位置や力の状態をチェックするプリミティブがある．これらの制御プリミティブは，目標とする状態に導くだけで，上位のレベルで初期状態，終了状態を与えることにより意味を持たせる．

3.2. 制御スキルレベル

制御スキルは制御する 1 方向の制御プリミティブの組合せで定義し，その目標値を設定する．

まず，図 1(a) に示すように，ロボットによる組立用の機能素をランドマークとして，目標状態の切替を行う境界を定義する．ここでは，図 1(a) に示す組立作業のうち，part b の shaft_bottom を part a の approach から work へ，一定の反力がかかるまで 1 方向に動作させるスキル「突き当て (touch(D1))」を例にする (図 4).

スキル touch(D1) は shaft_bottom が，approach から work への移動し接触する状態遷移として定義できる．突き当てた状態は，位置が目標とする領域内にあり，かつ適正な反力を検出することにより達成される．

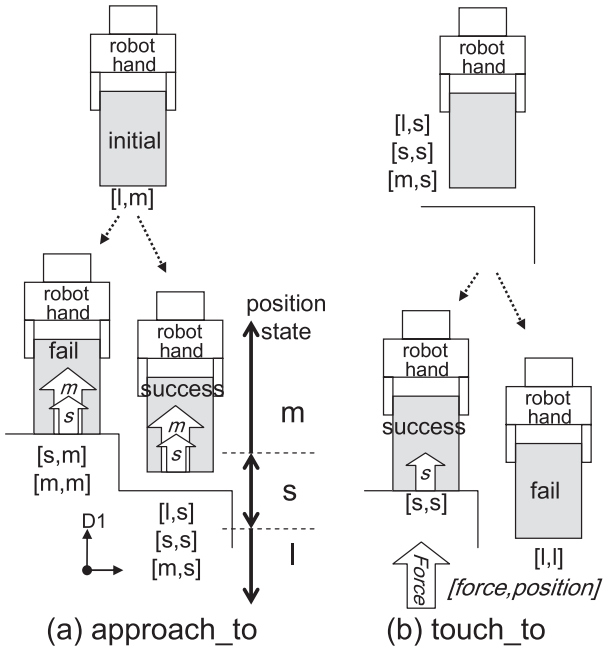


図 4 touch(D1) を構成する動作と各状態

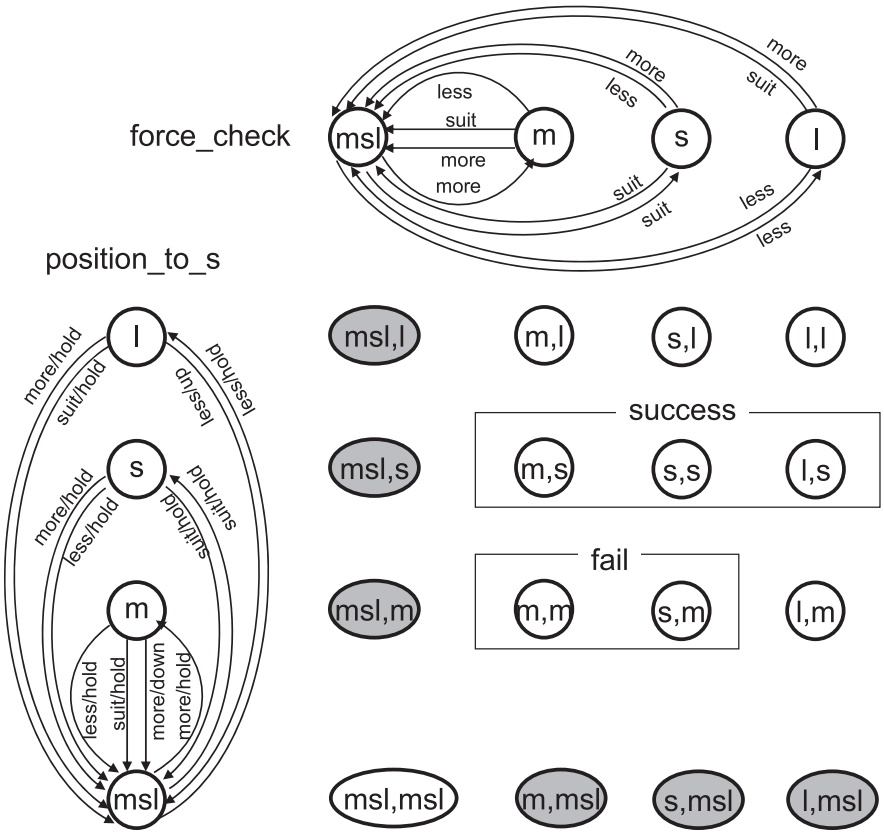


図 5 位置と力の制御プリミティブの直積の FSA

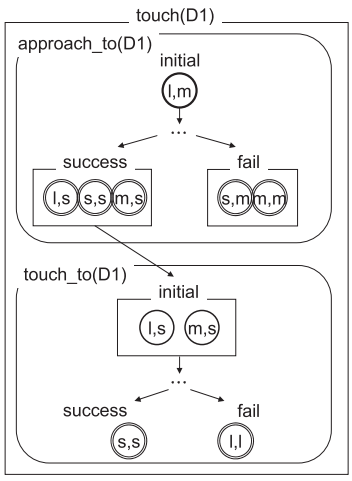


図 6 touch(D1) の状態遷移グラフ

よって終了状態 (goal) は，機能素 work と機能素 shaft_bottom の一致の状態として，[s, s] (力の状態∩位置の状態) と定義され，初期状態 (initial) はまだ機能素 shaft_bottom が機能素 approach の位置にある状態なので [l,m] と定義される．

図 4 に示すように，大まかな位置決めは高速に位置制御 (制御スキル approach_to(D1)) で，細かな位置決めは微妙な力の調節による力制御 (制御スキル touch_to(D1)) で行うといった，目的に応じた制御モードの切替により組立作業を行う作業スキルを考える．ここで，位置制御は位置の制御プリミティブにのみ出力を持たせ，力の制御プリミティブは状態の確認を行う．力制御はこれと逆に力の制御プリミティブに出力を持たせる．このように，それぞれの目的に応じた出力アクションを割り当てる．

approach_to(D1) は，図 5 に示すように制御プリミティブ position_to_s(D1) と force_check(D1) の FSA の直積をとり定義する．図中, 灰色の状態は存在しない．既報 [21] では，直積の FSA は 16 状態だったが，本論文では図 3(b) の FSA の採用により 10 状態となる．これにより，[msl,msl] 以外の msl を含む状態 (例えば，[msl,m]) が作業対象の現状態を判断できないループが発生することを回避できた．

approach_to(D1) は，initial の [l,m] から位置の状態が s になれば成功とする．すなわち [l, s]|[s, s]|[m, s] (| は∪を表す) 状態になれば成功状態 (success) である．ここでは，位置の状態が s になる前に障害物に衝突などして「手前で突き当て」という失敗状態 (fail) が考えられる．この fail は，位置が m でありかつ力の状態が接触と判断する [s, m]|[m, m] として定義できる．これらを，図 6 に示す状態遷移グラフで表現する．ここで，図 6 のアークには，図 5 に示す position_to_s(D1) と force_check(D1) の直積の 10 状態の FSA が相当する．すなわち制御スキルレベルでは，1 方向について位置及び力の制御プリミティブの直積をとり，制御モードを設定し initial 及び goal として, success と fail を定義する．

touch_to(D1) については，position_check(D1) と force_to_s(D1) の直積をとり図 6 に示すように定義する．touch_to(D1) は十分に近付いた状態から適正に接触させるため，initial は位置の状態が s である [l, s]|[m, s] であり，success は位置及び力の状態がともに s である [s, s] として定義する．また，[l, l] は接触させようとしたが目標を外れたりしておこる「突き当て失敗」が考えられる．

3.3. スキルレベル

まず，1 方向の分解可能な制御モードである制御スキルを利用してスキルを定義する．スキル touch(D1) の状態遷移グラフを図 6 に示す．スキル touch(D1) は，initial が [l,m]，goal が [s, s] への状態遷移として定義

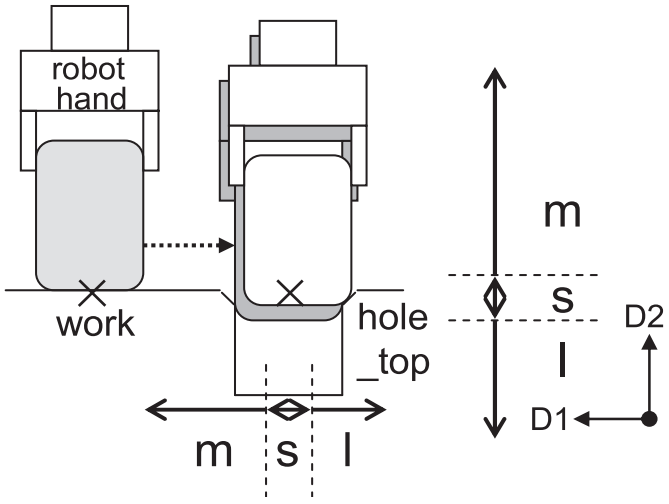


図 7 fit(D1,D2) を構成する動作と各状態

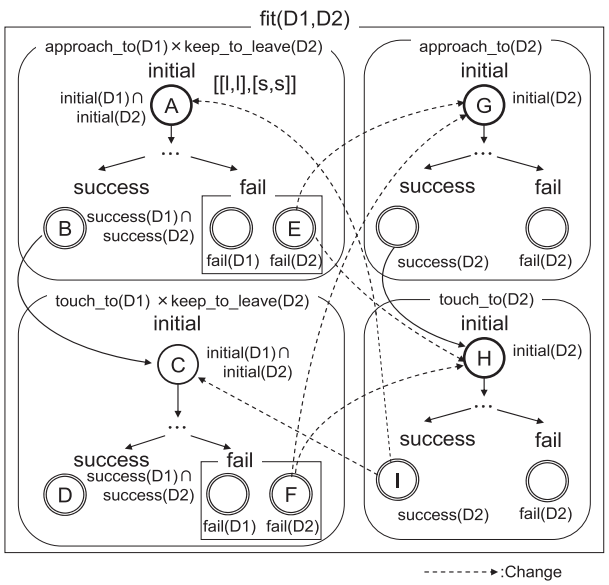


図 8 fit(D1,D2) の状態遷移グラフ

する．そして，独立に定義した制御スキル approach_to(D1) と touch_to(D1) の連接で touch(D1) を定義する．touch(D1) は，まず位置制御で大まかな位置決め approach_to(D1) を実行する．この，approach_to(D1) は goal である success ∪ fail になれば終了となる．ここで，approach_to(D1) が goal になった状態を Final とする．この Final に対し Final ⊂ (touch(D1) の goal) のとき，すなわち Final = [s, s] の場合，touch(D1) は成功となり終了する．また，Final が touch_to(D1) の initial に対し，Final ⊂ (touch_to(D1) の initial) が成り立つとき，すなわち Final = [l, s]|[m, s] の場合，制御モードの切替を行い力制御で touch_to(D1) を実行する．そして，同様に touch_to(D1) が success になれば touch(D1) は成功となり終了する．approach_to(D1) または touch_to(D1) で fail になった場合は，切替える制御スキルがないため touch(D1) は失敗となる．すなわち，制御スキルレベルの success，fail は，制御モード切替を行うためのランドマークである．

次に，図 7 に示す「はめ合い (fit(D1, D2))」を例に，多方向についても同様に定義できることを示す．動作方向を分解して考えると D1 方向は突き当て，D2 方向は維持から離脱 (keep_to_leave(D2)) と 2 方向の制御として定義する．図 7 に示すように，D1，D2 方向の位置 (力は省略) の状態を定義する．fit(D1, D2) の goal は，面取り部同士の突き当て，またはクリアランスが大きい場合などには突き当てる前に偶然入ることが考えられる．

fit(D1,D2) の状態遷移グラフによる表現を図 8 に示す．図 8 の initial(D1)，success(D1)，fail(D1) は，それぞれ制御スキルの D1 方向の initial，success，fail を意味し，initial(D2)，success(D2)，fail(D2) についても同様である．fit(D1, D2) も touch(D1) と同様に，approach_to(D1)×keep_to_leave(D2) から touch_to(D1)×keep_to_leave(D2) へ切替を行い実行する．これらのアークには 10^2 状態の FSA が対応する．スムーズに作業が進む場合は，図 8 中の A → B → C → D と状態が遷移し，fit(D1, D2) は success で終了する．しかし，D2 方向に関しては，位置及び力の状態を s に維持し続けるため，調整動作が必要となる場合がある．D2 方向が fail(図 8 中 E,F) になった場合，そのときの状態が approach_to(D2) 又は touch_to(D2) の initial(図 8 中 G,H) に含まれる場合のみ調整可能と判断し調整動作を行う．多方向の制御スキルの直積では，initial 及び success は各方向の AND(initial(D1) ∩ initial(D2) 及び success(D1) ∩ success(D2))，fail は 1 方向が fail になれば切替えが必要となるため各方向の OR (fail(D1) ∪ fail(D2)) で定義し制御方向及び制御モードの切替を行う．これにより，D2 方向が fail になれば 1 方向に切替え，approach_to(D2) または touch_to(D2) により調整動作を行う．調整動作が success(図 8 中 I) の状態に遷移し

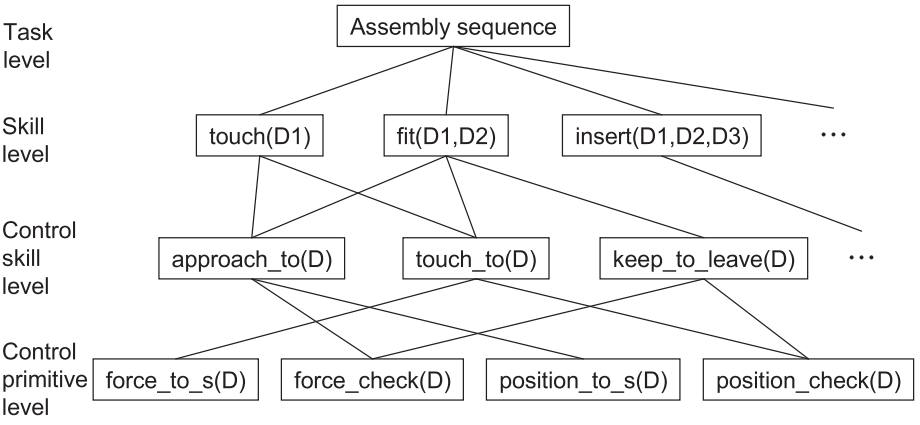


図 9 知的制御階層構造の例

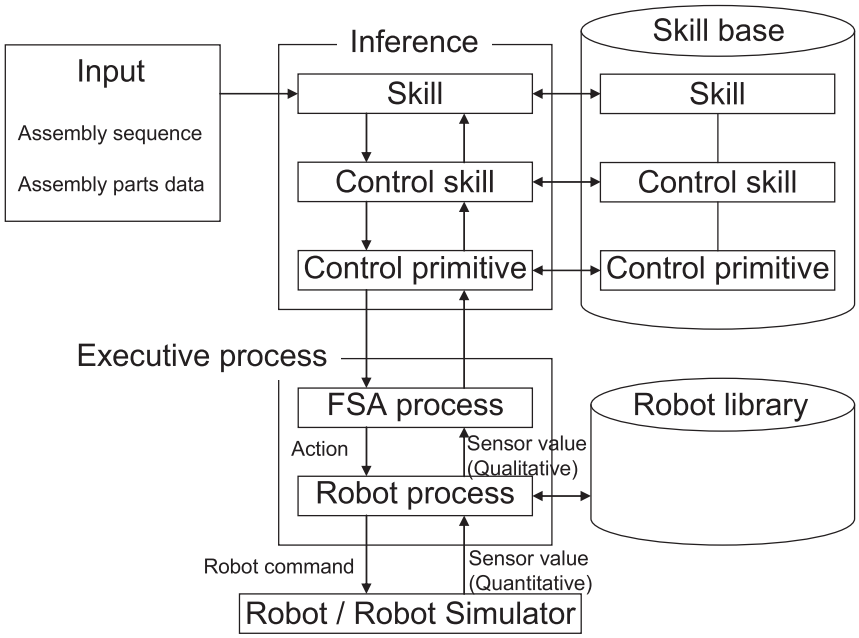


図 10 システム構成

た場合，その状態が含まれる initial(図 8 中 A,C) に切替わる．

3.4. 知的制御階層構造

作業スキルを用いたロボットによる組立作業は，タスクレベルはスキルの連接で表現でき，スキルレベルの連接で表現でき，スキルレベルは制御スキルレベルの連接と直積で表現でき，制御スキルレベルは 1 方向の制御プリミティブの直積で表現できる．この階層構造を知的制御階層構造と呼ぶ．知的制御階層構造の例を図 9 に示す．例えば，touch(D1) は approach_to と touch_to で構成され，approach_to は force_check と position_check で構成されている．このように，下位レベルの組合せで上位レベルを定義できるため，容易に新しい制御スキルやスキルを定義することができる．

4. CRS によるスキルの表現

4.1. システム構成

本研究の作業スキルの有効性を確認するため，Prolog を用いた試作システムを構築した．システムの構成を図 10 に示す．本システムは，スキルベース，ロボットライブラリ，入力部，推論部，実行プロセス部で構成されており，入力部から組立の動作手順がスキルの連接で与えられると，推論部でスキルベースを参照し，必要なスキル，制御スキル，制御プリミティブを呼び出す．実行プロセス部では，制御プリミティブの FSA を目標状態に導くための動作命令をロボットに送り，センサの値を定性値に量子化して FSA プロセスの入力ストリームとして返す．ロボットの動作命令は，ロボットライブラリを参照し動作させるロボットの命令に変換し，出力される．ロボットライブラリとロボットプロセスを用意することで，多種多様なロボットやシミュレータへの接続が容易になる．

図 10 の実行プロセスの詳細を図 11 に示す．実行プロセスでは 4 つのプロセス（FSA，Robot，Sensor，Coordinate transformation プロセス）のループによって作業スキルが実行される．実行中これらのプロセスは，互いにメッセージ伝達によって通信している．ロボット制御用のコマンドは “Robot” プロセスで生成される．“Sensor” プロセスの中では，センサ値が “FSA” の中で使われる more，suit，less のいずれかに変換される．作業スキルを任意の制御方向に対して実行するため

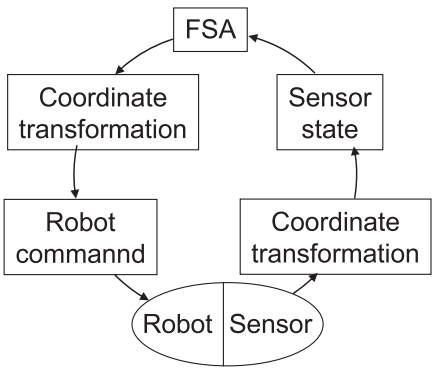


図 11 実行プロセスのループ

には，座標変換が必要である．座標変換はロボットの位置及び姿勢を表すマトリックスをロボットの動きに合わせて動的に変更することによって行われる．

4.2. プロダクションシステム

試作システムでは，CRS（Constraint Reduction System）[19] の概念をロボット作業の実行のために導入する．CRS では，“拘束条件リダクション法”と呼ばれる推論メカニズムが使用される．このメカニズムは，拡張されたプログラム言語 Prolog によって実現されている．CRS の中ではメッセージによって通信しているプロセスがルールに応じて多くのプロセスにリダクションされる（図 12）．リダクションされたすべてのプロセスが実行され消滅したときその源プロセスは完全に実行されたものとみなされる．

例えば，touch(D1) は approach_to(D1) と touch_to(D1) の連接のプロセスにリダクションされる．さらに，approach_to(D1) は position_to_s(D1) と force_check(D1) の直積のプロセスにリダクションされる．これらは，順番に実行されていき，最終的に touch_to(D1) のリダクションされたプロセスが完了すると，touch(D1) が完了したことになる．

このように，スキル，制御スキル，制御プリミティブのどの階層もプロセスとみなしリダクションルールを用いて記述することができる．

4.3. FSA の直積の表現

FSA の直積は並行なプロセスに分解されて実行される．例として，ロボットの手先の大まかな位置制御を行う制御スキル（approach_to）は位置を S（適正）へ導く制御プリミティブ（position_to_s）と力がかかっているかどうかをチェックする制御プリミティブ（force_check）の直積で表現される．これをリダクションルールで記述したものを図 13 に示す．

図中の D1，D2 は方向を，R1，R2 は各方向の適正範囲を，In は入力状態を，Out は結果の出力状態をそれぞれ表す．システムは (1) と (4) に挟まれるプロセス (2)，(3) を平行なプロセスとして処理する．

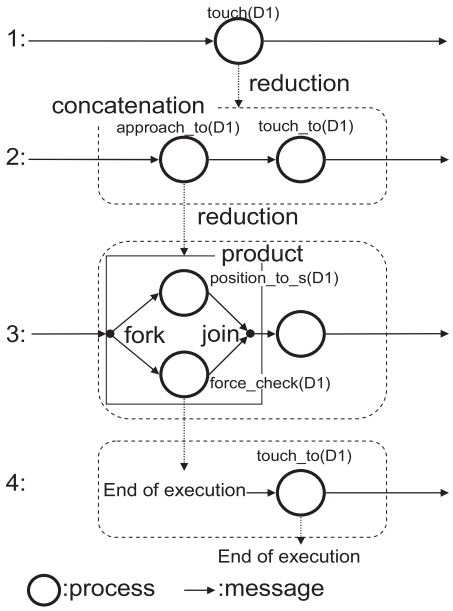


図 12 プロセスリダクション

```
approach_to( ?{D1,D2}, ?{R1,R2},
             ?In, @Out ):-
  true |
  fork(In,In1,In2),                              …(1)
  trans(force_check,D1,R1,In1,Out1,l,l),         …(2)
  trans(position_to_s,D2,R2,In2,Out2,m,s),       …(3)
  join(Out1,Out2,Out).                           …(4)
```

図 13 approach_to のプログラム

4.4. FSA の連接の表現

FSA の連接は，メッセージで結ばれた直列のプロセスに分解されて実行される．例として当て付けを行うスキル（touch）は大まかな位置制御の制御スキル（approach_to）と接触状態に導く制御スキル（touch_to）の連接で表現される．これをリダクションルールで記述したものを図 14 に示す．

図中，D1，D2 は方向を，R1，R2 は各方向の適正範囲を，InS，InE，InM は各プロセスの初期状態を，OutS，OutE，OutM はプロセスの終了した時の状態をそれぞれ表す．プロセスは (1), (2) の順番で処理される．

このようにプロセスリダクションと CRS によって，提案した制御概念を統一的に記述することができる．

5. 作業実験

作業プログラムはリダクションルールを使ってすべての作業スキルを記述することによってなされる．例え

```
touch( ?{D1,D2}, ?{R1,R2},
       ?InS-InE, @OutS-OutE):-
   true |
   approach_to(D1,R1,InS-InM,OutS-OutM),   …(1)
   touch_to(D2,R2,InM-InE,OutM-OutE).      …(2)
```

図 14 touch のプログラム

```
peg_in_hole( ?{D1,D2,D3}, ?{R1,R2,R3},
             ?InS-InE, ?OutS-OutE ):-
      true |
      touch(D1,R1,InS-In2,OutS-Out2),
      fit(D2,R2,In2-In3,Out2-Out3),
      insert(D3,R3,In3-InE,Out3-OutE).
```

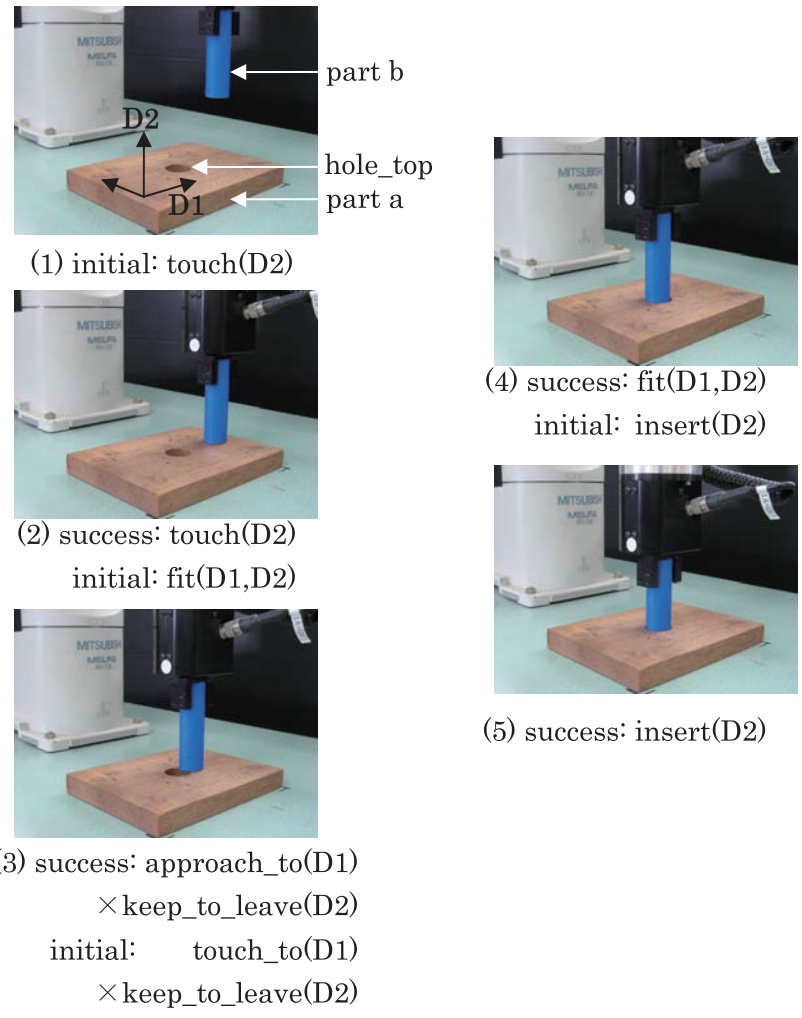
図 15 peg_in_hole のプログラム

ば図 1 に示した作業 “peg_in_hole” はスキルレベルの “touch”，“fit”，“insert” の連接によって図 15 ようにCRS を用いて表現される．

これらのプログラムは，図 10 に示した各ステップを通してロボットを動かす制御用コマンドに変換される．例えば，“touch” は “approach_to” と “touch_to” に分解される（図 9）．“approach_to” はさらに “force_check” と “position_to_s” に分解される．ロボットは “force_check” と “position_to_s” の FSA の直積に基づいて作業を行う．すなわち，ロボットの動き（up や down など）は現在の腕の位置と力センサの値に基づき FSA のルールにしたがって決定される．この動作はロボットの制御コマンドに変換される．

今回，試作システムを用いて，“peg_in_hole” の動作の確認を行った．実験環境は次の通り．ロボットは産業用ロボット MELFA　RV-1A(三菱電機製)．力センサは 6 軸力センサ IFS-67M25A15-I40 (ニッタ製)．

実験結果を図 16 に示す．図中 (1) に作業対象物 part a と part b とその作業座標系を示している．作業は (1) から (5) まで順番に touch(D2)，fit(D1,D2)，insert(D2) が実行されたことを示している．これにより，今回提案した作業スキルが CRS を用いて表現され，正しく実行できることが確認できた．また，既報 [21] の FSA の場



(1) initial: touch(D2)

(2) success: touch(D2)
initial: fit(D1,D2)

(3) success: approach_to(D1)
×keep_to_leave(D2)
initial: touch_to(D1)
×keep_to_leave(D2)

(4) success: fit(D1,D2)
initial: insert(D2)

(5) success: insert(D2)

図 16 作業実験

合は，組立作業の成功率は 50% 程度であったが，本論文の FSA ではほぼ 100% の成功率であった．

6. 結言

本論文では，簡単な位置制御と力制御による状態遷移の直積と制御モードの切替により，複雑な組立作業スキルの表現手法について示した．簡単なプリミティブの組合わせにより複雑な組立作業が行える知的制御階層構造について述べた．この構造は，新たなスキルも既存の制御スキルを組合わせることにより容易に作成できる．さらに，拘束条件リダクションシステム（CRS）を導入することで，スキル，制御スキル，制御プリミティブをそれぞれプロセスとしてとらえ，プロセスリダクションにより知的制御階層構造の各レベルが表現できることを示した．最後に，CRS を用いて試作したシステムにおいて作業実験を行い CRS による知的制御階層構造で定義された作業スキルの有効性を示した．

今後は，さらにスキルや制御スキルの種類の充実と，人の持つ作業スキルをいかに FSA のレベルに変換するかの研究が必要である．また，知的制御階層構造のプログラム編集環境の充実が必要である．

ACKNOWLEDGMENT

We would like to thank to Isao Nagasawa in Kyushu Institute of Technology, Hidetoshi Tsukihara in Sumitomo Heavy Industries, Ltd. and Takahiro Nakata in Anritsu Corporation who discussed our research.

参 考 文 献

[1] “「組立作業計画」特集”, Journal of Robotics Society of Japan, Vol.11, No.2, 1993.

[2] K. Ikeuchi, T. Suehiro: Task Model for Assembly Plan from Observation System, Journal of Robotics Society of Japan, Vol.11, No.2, pp.281-290, 1993

[3] T. Hasegawa, T. Suehiro, K. Takase: An Integrated Robot System with a Geometric Environment Model and Manipulation Skills, Journal of Robotics Society of Japan, Vol.9, No.1, pp.66-74, 1991

[4] S.Hirai, H.Asada and H.Tokumaru, Kinematic Analysis of Contact State Transitions in Assembly Operations and Automatic Generation of Transition Network, Trans. of SICE, Vol.24, No.4, pp.84-91, 1988.

[5] Xiao, j. and Ji, X.:A Divide-and-Merge Approach to Automatic Generation of Contact States and Planning of Contact Motion, IEEE Int. Conf. on Robotics and Automation, pp.750-756, 2000

[6] Ong, C. and Gilbert, E. G.: Growth Distance: New Measures for Object separation and penetration, IEEE Int. Trans. on Robotics and Automation, Vol.12, No.6, pp.888-903, 1996

[7] Goeree, B.B., Fasse, E.D. and Marefat, M.M.: Determining Feasible Contact State of Pairs of Spatial Polyhedra, IEEE Int. Conf. on Robotics and Automation, pp.1396-1401, 2000

[8] Pan, F. and Schimmels, J. M.: Efficient Contact State Graph Generation for Assembly Application, IEEE Int. Conf. on Robotics and Automation, pp.2592-2598, 2003

[9] M. Tsuda, T. Takahashi, H. Ogata：Generation of an Assembly-Task Model Analyzing Human Demonstration, Journal of Robotics Society of Japan, Vol.18, No.4, pp.73-82, 2000.

[10] J. Takamatsu, K, Ogawara, H. Kimura, K. Ikeuchi: Abstraction of Assembly Tasks to Automatically Generate Robot Motion from Observation, Transactions of Information Processing Society of Japan, Vol.46, No.SIG_9(CVIM_11), pp.41-55, 2005.

[11] T. Suehiro, K. Takase: Skill Based Manipulation System, Journal of Robotics Society of Japan, Vol.8, No.5, pp.551-562, 1990

[12] Y. Murakawa, M. Uchiyama: An Approach to the Design of a Skill System for Assembly Tasks, Journal of Robotics Society of Japan, Vol.9, No.1, pp.11-17, 1991

[13] W. Yoon, T. Suehiro, H. Onda, K. Kitagaki: Task Skill Transfer Method Using a Bilateral Teleoperation, Journal of Robotics Society of Japan, Vol.25, No.1, pp.155-165, 2007.

[14] J. Ooga, R. Katsuki, T. Yoshimi: Universal-design for manipulation framework ～ manipulation framework of servo level ～ , The 25th Annual Conference of the Robotics Society of Japan, 1I17, 2007.

[15] T. Tomiyama, H. Yoshikawa: “Theory of Design Process Model : An Application of Pai-number to Machine Design in General Design Theory”, Journal of the Japan Society of Precision Engineering, Vol.49, No.4, pp.441-446, 1983.

[16] B.Kuipers,YT Byun: “A Robust Qualitaive Method for Robot Spatial Learning”, In Proccedings AAAI88 pp.775-779, 1988

[17] S.Arikawa, S.Miyano: “ オートマトンと計算可能

性 ”, BAIFUKAN CO., LTD, 1986.
[18] T.Kurokawa: “Prolog のソフトウェア作法 ”, Iwanami Shoten, 1985.
[19] I. Nagasawa, Y. Furukawa and S. Aramaki : “ADL : A Designer's Language Based on Logic Programming”, Transactions of Information Processing Society of Japan, Vol.25, No.4, pp.606-613,1984.
[20] M. Mochizuki, I. Nagasawa, M. Umeda, T. Higuchi, and T. Ojima. A knowledge representation language for tolerance analyses and its programming techniques. Transactions of Information Processing Society of Japan, 35(9):1922-1935, 1994.
[21] H.Tsukihara, I.Nagasawa, S.Aramaki and T.Nakata, Method of Generating Assembly Action Using Assembly Architecture and Task Skill, Trans. Of JSME, Vol.60, No.578, pp.328-333, 1994.
[22] Shigeto Aramaki, et.al.: “The Knowledge Representation and Programming for Robotic Assembly task”, Proc. of ISATP ’99, pp.256-261, 1999.