計測自動制御学会東北支部 第 163 回研究集会 (1996.10.19)
資料番号 163-11

ペトリネットによる双腕ロボットの協調制御

Cooperative Control of Dual-Arm Robots Using Petri Net

○工藤 亨*, 渡部慶二*, 山田 功*, 斎藤周次*

○ S.Kudo*,K.Watanabe*,K.Yamada*,S.Saitoh*

*山形大学

*Yamagata University

キーワード： 協調制御 (Cooperative control), ペトリネット (Petri net), 双腕ロボット (Dual-arm robot)

連絡先： 〒992 米沢市城南 4-3-16 山形大学工学部 電子情報工学科 渡部研究室
工藤 亨, Tel.: (0238)26-3326, Fax.: (0238)24-2752, E-mail: kudo@ewky.yz.yamagata-u.ac.jp

1. はじめに

近年、ロボット工学の進歩により、さまざまな分野において自動化、省力化を目的としてロボットが使用されるようになった。しかし、すべての作業を1台で行えるような汎用性のある多機能ロボットは現段階ではコンピュータの処理能力やコストなどの点で実用化が困難な状況にあり、作業内容の限定された単機能ロボットを使用するのが普通である。そこで、高性能な単機能ロボットを目的に応じて数多く開発し、それらを組み合わせることで汎用性を持たせることを考える。この場合に問題となるのが複数台ロボットの協調制御である。つまり、1台のロボットでは作業不可能な場合でも、複数のロボットが集まれば作業可能であるといった場合は多く、たとえば、1台のロボットアームでは持ち上げることができない大きなものでも、2台のロボットアームがあれば持ち上げることができるかもしれない。このような場合、各ロボットアームがバラバラな動きをしたのでは目的を達成することはできないので、2台のロボットアームが協調して作業するように制御するのが協調制御である。

本発表では、複数のロボットで構成されるシステムにおいて、各ロボットの状態、データの流れ、制御の流れなどを統合的に扱うことができるペトリネットを用いた協調制御のアーキテクチャを提案し、双腕ロボットに対して適用する。

2. 複数ロボットシステム

ここでは、各自が制御用コンピュータ (PC) を持つ複数のロボット (ユニット) で構成される複数ユニット協調作業システムを想定する (*Fig*1)。

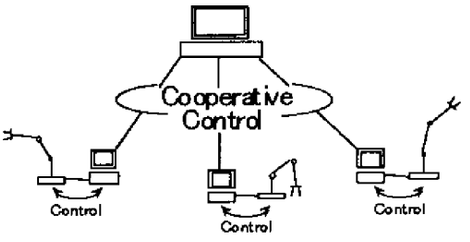

Fig. 1　複数ロボットシステム

メインコンピュータは各ユニットの状態やシステム全体で共有するべきデータを管理し、ユニットが協調して作業するような指令を出す、いわば指令塔の役割を果たす。このメインコンピュータを通してオペレータはシステムを制御する。オペレータはメインコンピュータから各ユニットに対して作業司令をだすが、司令後の各ユニットの動作は各PCによって制御される。基本的に各ユニットは独立しているものとする。

システムのイメージ図を *Fig2*に示す。

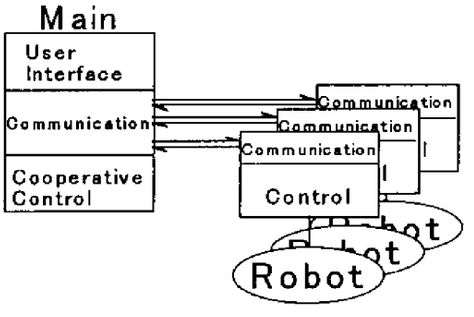

Fig. 2　システムのイメージ図

*Fig2*が示すように、指令塔となるメインコンピュータは、オペレーターとの接点となるユーザ・インタフェース、各ユニットと双方向で通信を行なうための機能、各ユニットを制御、管理する機能を持たなければならない。このようなシステムに協調作業を行なわせる場合、実際にシステムを利用し、作業命令を考えるのは人間である。オペレータの意志を正確に伝達し、思っているとおりにシステムを動かすためには、ユーザインタフェースは簡単なものでなければならない。しかし、自然言語のような人間にとって理解しやすいものというのはコンピュータに対してかなりの処理能力を要求する。したがって、ユーザ・インタフェースとして望まれるのは、人間から見れば仕様がそのまま表現でき、かつ、変更や追加が容易なこと、コンピュータ側からはコンピュータが解釈し、実行しやすい簡単なものである、ということである。

協調制御の特徴でもある複数ユニットによる並列動作を考えると複数のユニットが同時に作業をしたい場合は適切なタイミングで作業開始の合図や作業内容や必要なデータを同時に複数のユニットへ送らなければならない。

このように、ユーザインタフェース、通信、制御の３つの機能はそれぞれが専門の知識を必要とし、これまでは独立に議論されてきた。しかし、このようなシステムでは *Fig2*が示すように、１つのマンロボットインタフェースとして扱うべきものである。これら３つの機能を統合して扱えれば、つまり、ユーザインタフェースや通信や制御という別々のものを意識せずに扱えれば、それぞれ個別の知識を必要とせず、管理者への負担はかなり軽減することができる。この点を考慮に入れた協調制御のアーキテクチャを提案する。

3.　協調制御アーキテクチャ

ここで、マンロボットインタフェースとしてペトリネットを用いる。計算機を介してオペレータとロボットとが相互に通信しあうことをマンロボットインタフェースという。オペレータとロボットとのインタフェースを良く

するためには、プログラムが書きやすく、オペレータの意志を誤りなくロボットに伝達することができなければいけないし、また、ロボットの状態を常に監視することができなければならない[3]。

ペトリネットは、非同期的でかつ並列的に振る舞うシステムに対して、その中の情報の流れや制御を記述し解析するために考え出されたものである。そして、いくつかの事象が並列的に発生する中で、それらの発生の順序、頻度などにある制約が与えられているようなシステムをモデル化するために主として用いられてきた[1]。

したがって、複数のユニットによる並列動作を特徴とするような協調制御へのペトリネットの導入は適していると考えられる。

3.1 ペトリネットの定義

ペトリネットは動的なグラフィックモデルであり、形式的に次のように定義される[1]。

定義 1 5項組 $N=(P,T;F,C,W)$ は、次の場合に限りネットであるという。

(1) $(P,T;F)$ はネットである。ただし、Pの要素はプレースと呼ばれ、Tの要素はトランジションと呼ばれる。$F \subseteq (P\times T)\cup(T\times P)$ は2項関係でネット Nにおける流れの関係である。

(2) $C:P\to N^+\cup\{\infty\}$ は**容量関数**

(3) $W:F\to N^+$ は**重み関数**

ただし、N^+は正の自然数の集合を表す。

ここでは、$|P|<\infty$、$|T|<\infty$ であるような有限なペトリネットのみについて考える。

つまり、ペトリネットは*Fig*3に示す4つの部分からなっている。

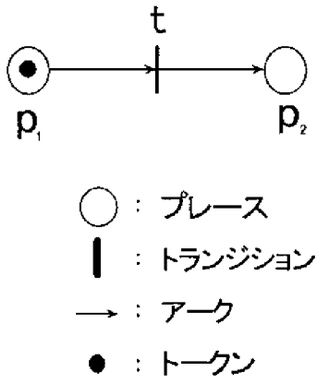

Fig. 3 ペトリネットの構成要素

まず、プレースと呼ばれる円があり、次にトランジションと呼ばれる棒がある。ダイヤグラム上では、プレースやトランジションであることを表すのに、たとえばプレースにp、トランジションにtを書き添えておく。

3つめはトランジションからプレース（あるいはプレースからトランジション）に向かう有効アークである。アークはネットにおける流れの方向を示している。矢印の先端はそれが指し示しているところへの入力を意味し、矢印の尾はその抜け出しているところからの出力を意味する。

これら、プレース、トランジション、アークによって構成されるネットのなかを流れているものがトークンである。トークンはトランジションの発火により プレースからプレースへ動き回る。これによってダイヤグラムの動的な性質を表す。トランジションの発火は次のような定義にしたがって実行される。

定義 2 $N=(P,T;F,C,W)$ をネットとする。

(1) Nは、もし$\forall(p,t)\in P\times T:(p,t)\in F\Rightarrow(t,p)\notin F$ であるならば、そのときに限り**純**であるという。

(2) 関数 $M:P\to N$は Nのマーキングと呼ばれる。

(3) トランジション$t \in T$は、もし$\forall p \in I(t)$: $M(p) \geq W(p,t)$かつ$\forall p \in O(t)$: $M(p) + W(t,p) \leq C(p)$であるならば、そのときに限りNのマーキングMのもとで発火可能である。

(4) トランジション$t \in T$はNのマーキングMのもとで発火可能であるとする。そのとき、tは次のように与えられるNの新しいマーキングM'を発生しうる。

$$M'(p) = M(p) - W(p,t)$$
(すべての$p \in I(t)$に対して)

$$M'(p) = M(p) + W(t,p)$$
(すべての$p \in O(t)$に対して)

$$M'(p) = M(p)$$
(すべての$p \notin I(t) \cup O(t)$に対して)

M'をMの直接次段マーキングと呼ぶ。一般に"Mの次段マーキングの集合"とは、直接次段マーキングに続くマーキングの集合を意味する。

3.2 複数ロボットシステムのためのペトリネットの定義

複数ロボットシステムを記述するために、プレース、トランジションおよびトークンを次のように対応させる。

ペトリネットの定義

Table 1 ネットの構成要素の定義

ペトリネット		システム
プレース	⇔	状態
トランジション	⇔	作業
トランジションの発火	⇔	作業の開始
トークン	⇔	状態の成立
アーク	⇔	状態の遷移

また以下のような発火規則に従ってトークンがネット内を移動するものとする。

発火規則

[1] トークンはペトリネットのトランジションを発火させることによりネット内を移動する。

[2] トランジションを発火させるためには、トランジションが発火可能でなければならない。トランジションのすべての入力プレースにトークンがあるとき、そのトランジションは**発火可能**である、という。

[3] トランジションが発火すると、そのトランジションのすべての入力プレースから1個ずつトークンを取り去り、T秒後（作業終了後）にすべての出力プレースに1個のトークンを生成する。ここで、トランジションの発火からトークンを生成するまでの間を**発火中**という状態にあるものとする。

ペトリネットとシステムの対応イメージ図を*Fig4*に示す。

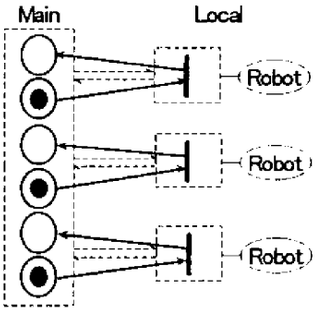

Fig. 4 ペトリネットとシステムの対応イメージ

ここではプレースによってメインコンピュータの状態を、トランジションによって各ユニットを表すことにする。つまりトランジションが発火可能であるという状態はそのユニットへ作業命令を出せる状態が成立していることを意味する。トークンはメインコンピュー

タがどのような状態にあるかということを表し、トークンのあるプレースがその時の状態である。また、トークンに内部情報を持たせることによってデータの受け渡しを表すことにする。

したがって、目的に応じてペトリネットで表すことにり、その構造は作業の工程を、プレースはメインコンピュータが各ロボットに対して作業命令を出せる状態にあるのかどうか、トランジションはどのロボットにどのような作業をさせるのか、トークンの位置は現在の作業の進行状況を、その動きはデータの動きを表すことができる。

3.3 同期と競合

複数ユニットの並列動作で問題となるのは同期と競合である。同期は複数のトランジションを同時に発火させることで表現することにする。たとえば、*Fig*5のように記述すればよい。

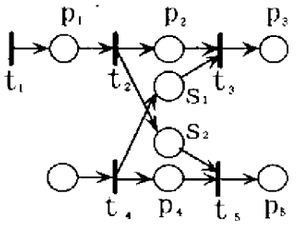

Fig. 5　複数ロボット間の同期

*Fig*5は、同期用のプレース s_1, s_2を導入することで、トランジション t_3と t_5が同時に発火する。つまり、t_3と t_5に対応する作業が同時に開始される。

また、ペトリネットの構造に、「プレースからの出力は常に1である ($|O(p)| = 1$)」という制約を設ける。これは、プレースの複数の出力トランジションによるトークンの奪い合いを避けるためであり、これにより、構造的な競合の禁止を行う。

3.4 例1

「ロボットハンド1がA地点にある物体を掴んで、B地点でロボットハンド2に渡し、ロボットハンド2がC地点に置く」という作業について考える。ロボットハンドがあらかじめ次のような番号で識別される内部コマンドを持っているものとする。

0：あらかじめきめられた原点に移動する。

1：A地点に移動する。

2：B地点に移動する。

3：C地点に移動する。

4：ロボットハンドを閉じる。

5：ロボットハンドを開く。

また、プレース p_i, s_i、トランジション t_ijには次のような意味を持たせる。

p_i: メインコンピュータが作業命令を出せる状態にある。

s_i: 各ユニット間で同期を取るためのプレース。

t_{ij}: ユニット iが上述した内部コマンド jを開始し、作業が終わるまで発火中の状態を維持する。

たとえば、t_{12}はロボットハンド1が作業2(つまり、B地点に移動)を行うことを意味している。また、プレース pにつけられた添字はそれぞれを区別するためのもので、任意に与えることができる。

これらを用いると、上述した作業は

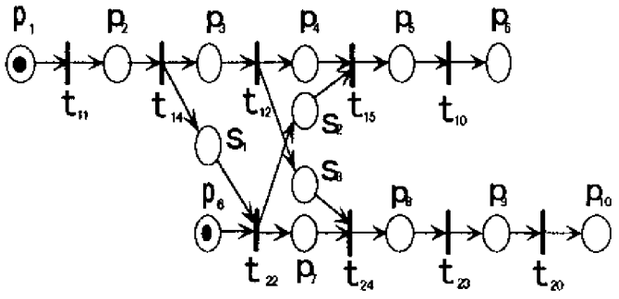

Fig. 6　ペトリネット表現

とあらわすことができる。

*Fig7*のマーキングは初期状態をあらわしている。初期状態ではメインコンピュータが作業命令を出せる状態が成立していて、トランジションt_{11}が発火可能である。つまり、ロボットハンド１へ内部コマンド１に対応する作業命令を出すことができる。トランジションの発火とともにロボットハンド１は作業を開始する。その後、作業終了とともに新たなトークンがプレースp_2へ生成される。これは、ロボットハンド１が作業終了メッセージをメインコンピュータに送信したことを意味している。そして順次トランジションの発火が進み、それとともにロボットは作業を進めていく。

3.5　例２

「カメラにより物体の位置を計算し、ロボットアーム１とロボットアーム２で物体をはさんで持ち上げ、Ｂ地点を通ってＡ地点に置く」という作業について考える。

各ユニットはあらかじめ次のような番号で識別される内部コマンドを持っているものとする。

0：あらかじめきめられた原点に移動する。

1：A 地点に移動する。

2：B 地点に移動する。

3：指示された地点に移動する。

4：ロボットハンドを閉じる。

5：ロボットハンドを開く。

6：物体の位置の情報を計算する。

また、プレースp_i, s_i、トランジションt_ijには次のような意味を持たせる。

p_i: メインコンピュータが作業命令を出せる状態にある。

s_i: 各ユニット間で同期を取るためのプレース。

t_{ij}: ユニットiの内部コマンドjに対応する作業。ユニット番号は１がロボットアーム１、２がロボットアーム２、３がカメラ。

これらを用いると、上述した作業は

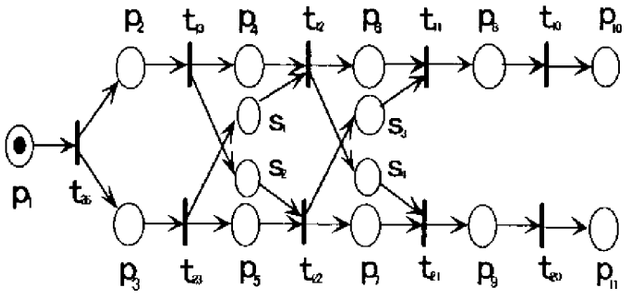

Fig. 7　ペトリネット表現

とあらわすことができる。

*Fig7*のマーキングは初期状態をあらわしている。まず、カメラが物体の位置の情報を計算する(t_{36})。そのデータはトークンによってまずメインコンピュータに返され、そのご各ユニットへ送られる。各ユニットは得られた位置のデータを基にその地点へ行く(t_{13}, t_{23})。この作業はロボットアーム１、ロボットアーム２同時に行なわれる。というように作業は進められる。

4. おわりに

本発表では、使用者の負担を減らし、簡単にシステムの記述やデータの流れ等を統合して扱うことができるようなものを目標に、ペトリネットを用いた協調制御アーキテクチャを提案した。しかし、このままではペトリネットによる表記はシステムが複雑になればなるほど、そして構成ユニットが増えれば増えるほどその大きさが大きくなってしまう、という欠点を持っている。この点はカラーペトリネットの概念を導入すれば解決できるのではないかと考えている。また、ここ想定した集中処理型システムは、一部の故障によるシステム全体への影響が大きく、また現代の流れに沿ったものであるとは言い難い。そこで、メインコンピュータを必要としない、分散処理型システムへの拡張が今後の課題となる。

参考文献

1) 椎塚久雄著、実例ペトリネット、コロナ社

2) 関口隆共著、シーケンス制御工学—新しい理論と設計法—、電気学会

3) 長田正著、基礎ロボット工学-知能情報編、昭晃堂