4R−7　協調活動の動的有限オートマトンによる記述手法

国島丈生　上林彌彦

京都大学工学部

1　はじめに

複数の人間が協調処理を行う場合、依頼する活動の正確な内容の伝達、活動の自動化などを行うためには、活動の進行するプロセスなど、活動そのものに関するデータを明確に記述する手段が必要である。

このとき考慮しなければならないことは、活動は動的に変更され得るということである。つまり、それまでの活動の結果、環境の変化、交渉の結果などの影響を受けてその時点からの活動が決定されることがある。しかも、変更の詳細はあらかじめ予測できないことがある。本稿では、このような協調活動の動的側面を考慮した協調モデルを提案する。

従来、協調対象としてプロセスを想定して協調活動のプロトコルを記述するモデルとしては、有限オートマトン[5, 4, 1, 2, 3]、ペトリネット、並行プログラミング、分散人工知能などが研究されている。本稿の協調モデルは、これらのうち、動作が把握しやすく、かつ最も単純な有限オートマトンを基にしており、さらに有限オートマトンの構造変更操作を定義している。

2　有限オートマトンに基づく協調モデル

以下では、協調対象のことをエージェントと呼ぶ。

2.1　協調モデルの構造

一つのエージェントは、そのエージェントの行う活動を表すFA(活動有限オートマトン、AFA)を複数持ち、そのエージェントの内部状態(またはその一部を表す値)、そのエージェント内部の状態の変更、外部からの入力(他エージェントからの事象・通信)、時刻事象を入力としてこれらのFAの状態を遷移させ、出力を他のエージェントに送信することによって通信を行う。また、出力を用いて自分の持つ他のAFAを遷移させることもできる。

AFAは、Moore型の非決定性有限オートマトンであり、その入力記号i、出力記号oは$\langle s, A\rangle$と定義される。ここに、sはあらかじめ定められた記号であり、AはAFAの集合である。特に入力記号の場合、Aの要素は一つだけとする。あるAFA aから出力記号$o = \langle s, A\rangle$が出力されると、記号sがAのすべての要素に送信される。sを受信したAFAは$\langle s, \{a\}\rangle$を入力記号として状態遷移を起こす。sとしてどのような記号が許されるかは、通信を互いに行うエージェント間であらかじめ合意ができているものとする。

A Description Method for Cooperative Activities using Dynamic Finite Automata
Takeo KUNISHIMA and Yahiko KAMBAYASHI
Kyoto University

2.2　構造変更操作

実際の協調活動では、例えば次のような活動変更が行われる。

- 活動の下請け依頼
- 活動の転送(結果のとりまとめなども含めて任せる)
- 活動に関する諸条件(デッドラインなど)の変更

AFA上では、このような活動変更はAFA自身の構造変更として表されるため、協調モデルはAFAの構造を変更できる必要がある。

そこで、ここではAFAの構造に対する基本変更操作を定義し、それらによって上記のような活動変更を構成する。基本変更操作としては次のようなものを用意する。

- *newActivity(M)*
  MというAFAがない場合、新たなAFA Mを作る。そのエージェントがすでにMという名前のAFAを持っている場合エラーとなり、新たに作られることはない。
- *deleteActivity(M)*
  AFA Mを削除する。そのエージェントがMを持っていない場合、エラーとなり、削除は行われない。
- *addState(M, s, o)*
  AFA Mに状態sがない場合、新たな状態sを付け加え、$\lambda(s) = o$とする。sがすでにある場合にはエラーとなり、付け加えられない。
- *deleteState(M, s)*
  AFA Mの状態sを削除する。sがM中に存在しない場合エラーとなり、削除は行われない。
- *addEdge(M, e, s_1, s_2, i)*
  AFA Mに遷移枝eがない場合、状態s_1からs_2に遷移枝eを新たに付け加え、そのラベルをiとする。eがすでにある場合にはエラーとなり、付け加えられない。
- *deleteEdge(M, e)*
  AFA Mの遷移枝eを削除する。Mにeが存在しない場合エラーとなり、削除は行われない。
- *changeStateLabel(M, s, o, o')*
  AFA Mの状態sについて、$\lambda(s)$をoからo'に変更する。$o' = \langle s, A\rangle$がAのどの要素の入力記号でもない場合エラーとなり、ラベルは変更されない。

- *changeEdgeLabel(M, e, i, i')*
  AFA M の遷移枝 e のラベルを i から i' に変更する。$i' = \langle s, \{a\}\rangle$ が a の出力記号でないときエラーとなり、ラベルは変更されない。

- *send(A, operation)*
  AFA 集合 A の各要素に対して、操作 *operation* を実行するよう送信する。ここに、*operation* は *send* 以外の基本変更操作である。

ここでの協調モデルは複数の AFA が協調して活動を実行していくため、ある AFA の構造を変更すると他の AFA の構造も変更しなければならない場合が生じ、一般にはこのときエージェント間で交渉により変更内容について合意をとる必要がある。本節で述べるモデルでは、簡単のため、合意を取る方法については考慮しておらず、AFA 自身の構造変更は、どのような変更を行うかについて協調を行うエージェント間で合意が得られてから行うものとする。

先に述べた基本変更操作で下請け依頼、転送を構成した例を以下に示す。

```
下請け依頼 (E,a)
# 遷移枝集合 E で表される活動をエージェント a に
# 下請け依頼する
foreach eij = <si,sj> in E
do if (send(a,newActivity(Me))) {
      # Me は遷移枝 eij に対応する AFA
      createState(M,sk,'<i を依頼, a>')
      createEdge(M,eik,si,sk,'<i を依頼,self>')
      # self はこの AFA 自身
      createEdge(M,ekj,sk,sj,'<i が終了,a>')
      deleteEdge(M,eij)
   }
end

転送 (E,a)
# 遷移枝集合 E で表される活動をエージェント a に
# 転送する
foreach e = <si,sj> in E
do if (send(a, newActivity(Me)) {
      changeEdgeLabel(M,e,'<i,self>','<i,a>')
      foreach a' in A'
      do  # A' は sj の出力を送信する AFA 集合
        x = name(self)
        send(a',
        changeEdgeLabel(M',e','<i,x>','<i,a>')
      end
   }
end
```

3 トリガ処理への応用

実際の協調作業の中では、すべての活動結果をエージェント a_1 に転送する、エージェント a_2 から依頼された活動はすべてデッドラインなどの変更を加える、などというように、活動の種類とは関係なく、何らかの条件が満たされるとある処理が行われる場合がある。

本稿の協調モデルでは AFA は一連の活動に対して一つ定義されるので、このような処理は一般には複数の AFA に共通で行われる。この処理を起こす条件が AFA の入力記号系列で表せる場合は、これを受理して適当な出力記号を出力するような AFA を新たに一つ設けることによって対処できる。

例えば、先の例の場合、前者は、活動結果として出てきた出力 $\langle s, A\rangle$ に対応して $\langle s, A \cup \{a_1\}\rangle$ を出力するような AFA を設ければよい。また、後者の場合は、a_2 からの入力についてデッドラインを変更した後、送信されるべき AFA へ送信してやればよい。

ただし、このような AFA を設ける場合、本来の送信元の AFA と送信先の AFA について、遷移枝のラベルを変更しておく必要がある。先の例の前者の場合、新たに付け加える AFA を M_{new}、M_{new} を付け加えるエージェントを a_3 とすれば、

- a_3 の持つすべての AFA のすべての遷移枝のラベル $\langle s, A\rangle$ を $\langle'\langle s, A\rangle', M_{new}\rangle$ に

- a_3 から出力が送られるすべての AFA 中の遷移枝のラベルのうち、$\langle s, M_{a_3}\rangle$($M_{a_3}$ はエージェント a_3 の持つ AFA) というものをすべて $\langle s, M_{new}\rangle$ に

それぞれ直さなければならない。

4 おわりに

本稿で提案した協調モデルの性質を明らかにしていかなければならない。

例えば、実際にある活動を受け持つエージェントが、その AFA で定義された入力・出力の構造をローカルに変更したい場合がある。しかし、AFA は一般に他の AFA と入出力を通信するので、変更を自由に行ってよいわけではない。他のエージェントとの合意なしに構造変更をローカルに行ってよいための条件を明らかにすることは、エージェントの自律性の点からも重要である。

参考文献

[1] D. Brand and P. Zafiropulo. On Communicating Finite-State Machines. *J. ACM*, Vol. 30, No. 2, pp. 323–342, Apr. 1983.

[2] T. L. Casavant and J. G. Kuhl. A Communicating Finite Automata Approach to Modeling Distributed Computation and Its Application to Distributed Decision-Making. *IEEE Trans. Comput.*, Vol. 39, No. 5, pp. 628–639, May 1990.

[3] M. H. Nodine and S. B. Zdonik. Cooperative Transaction Hierarchies: Transaction Support for Design Applications. *VLDB J.*, Vol. 1, No. 1, pp. 41–80, July 1992.

[4] R. R. Tenney and N. R. Sandell, Jr. Strategies for Distributed Decisionmaking. *IEEE Trans. Syst., Man, and Cybernetics*, Vol. 11, No. 8, pp. 527–538, Aug. 1981.

[5] R. R. Tenney and N. R. Sandell, Jr. Structures for Distributed Decisionmaking. *IEEE Trans. Syst., Man, and Cybernetics*, Vol. 11, No. 8, pp. 517–527, Aug. 1981.