AWSミドルウェアの研究　-動的モデル協調層-

塚本修也† 高木良輔† 木村泰輔† 大谷真†
湘南工科大学　情報工学科†

1．　はじめに

現状の自律型 Web サービスによる電子商取引は、システムのどちらかが相手のシステムに合わせてビジネスプロセスモデル(BPM)を構築しなければならない等の取り決めが存在する。

AWS ではこの取り決めを無くし、各自が自由に作成した BPM でも取引を可能にする事を目的としている。本論文では BPM の協調変換部分を司る動的モデル協調層のアルゴリズム、妥当性ついて述べる。

2．　AWS

2．1　　動的モデル協調層の位置づけ

AWS システムは３つの層で構成される。動的モデル協調層（MH 層）はそれぞれのシステムが持つ BPM の交換、協調変換を行い、新たに生成された BPM（協調済 BPM）をフレームワーク層（AF 層）へ提供する。また同時に、取引の流れを制御し、実行可能なオペレーションの候補を指示する。

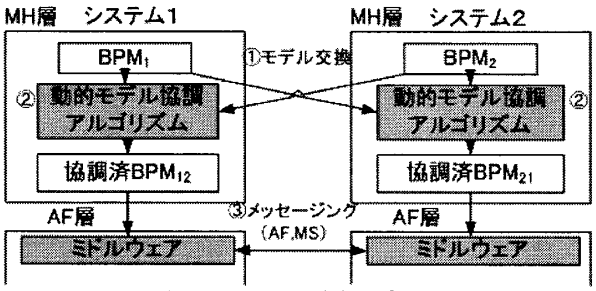

図１．　MH 層の流れ

協調変換とは互いの BPM の各状態の直積を求め、各状態に存在するパスの整合性検査を行い、不整合なパスを削除した後、不要な状態を削除することである。

以上の手順で生成された協調済 BPM は互いに取引が成功するパスだけが残る。システムはこの協調済 BPM に従い取引を行う。

2．2　　開発方針

- モデルはオートマトンの範囲に限定する。
- 利便性を考慮し、非決定性を実装する。
- [2]では正規表現を用いているが、実用性を考慮し状態遷移関数表現とする。
- 変換方式は、オートマトンの直積を取り、各状態に対して再帰的に整合性の検査を行う。

3．　BPM

取引は互いに手順を進め、状態を変化させることであり、これをモデル化したものを BPM と呼ぶ。本研究では BPM を非決定性有限オートマトンで表現する事とした。

3．1　　BPM の形式定義

定義は[2]に従い *BPM=(O,B) O=op*（オペレーションの集合）、*B*=オートマトンとして表現される。*op=(p,t) p=Input/Output*パターン、*t*=メッセージの型である。*B*はオートマトン *(S, O, T, s0, A)*として定義される。

BPM は上記の定義を XML によって表現することとした。XML の主な要素を以下に示す。operation は *op* に相当し、pattern は *p* に相当し、state は *S* に相当し、next は *T* に相当し、first は *s0* に相当し、last は *A* に相当する。

図２に BPM の概略例を示す。

3．2　　BPM 例

```
<operations>      ・・・Oに相当、opの集合
 <operation name="見積依頼">     ・・・opに相当
  <pattern>output</pattern>      ・・・データの入出力パターン
 </operation>
</operations>
<states>
 <state no="0"> ・・・Sに相当。取引の状態。
  <next operation="見積依頼">1</next>     ・・・Tに相当
  <next operation="見積依頼">3</next>     ・・・operation=入力、値=遷移先
 </state>
</states>
<first>0</first>   ・・・s0に相当、初期状態
<last>4</last>     ・・・Aに相当、受理状態
```

図２．　BPM の XML 表現例

4．　動的モデル協調アルゴリズム

4．1　　アルゴリズム

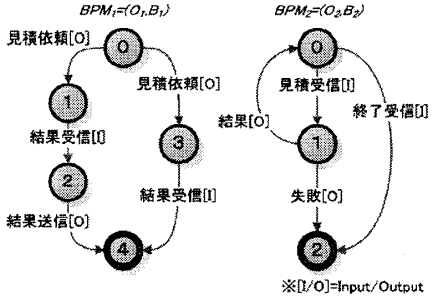

図３．　BPM 例

Study on AWS Middleware-Dynamic Model Harmonize Layer-
† Shuya Tsukamoto, Ryosuke Takagi, Taisuke Kimura, Makoto Oya, Shonan Institute of Technology

$BPM_1=(O_1,B_1)$, $BPM_2=(O_2,B_2)$, $B_1=(S_1,O_1,T_1,S0_1,A_1)$, $B_2=(S_2,O_2,T_2,S0_2,A_2)$とし、図３を用いてアルゴリズムを説明する。

まず、S_1 ,S_2の直積を求め、新たな状態 S ($=S_1 \times S_2$)を作成する。（図４-①） $S0(=S0_1 \cap S0_2)$, $A(=A_1 \times A_2)$が協調済 BPM の開始状態と受理状態である。$S0$から順に BPM_1, BPM_2の（現在参照している位置の）次状態の直積 $S_{next}=(S_{1next} \times S_{2next})$を求め、そのパスを Sへ代入する。パスとは図２、３の状態遷移の矢印の事であり状態遷移関数 T の入力と遷移先である。代入とはパスを S に当てはめる事であり、T の生成である。この代入を行う前に整合性の検査（４．２参照）を行い、整合の場合のみ代入を行う。代入を行った時、そのパスが指す先に対して同様の処理を再帰的に行う（0, 0→1, 1→2, 0 というように）。図４-①は全てのパスの候補を表示しているが、実際に代入されるパスは図４-②の通りである。例えば 3, 1→4, 2 へのパスは I/0 パターンは合致するが型が整合でないため代入されない。

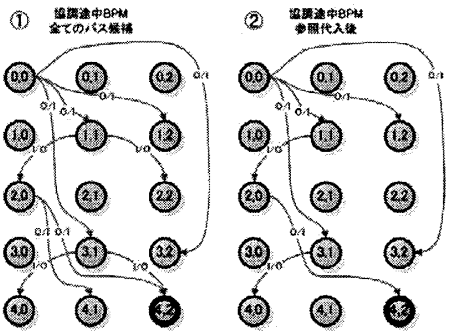

図４．　協調変換アルゴリズム１

次に S の開始状態から再帰的にパスを辿る。手順は図５の通りである。例えば 0, 0→1, 2 のパスは 1, 2 から次状態へのパスが無く、受理状態でも無いのでプログラムは 1, 2 の flag を false とし、return -1 を行う。その結果、0, 0→1, 2 へのパスは削除される。

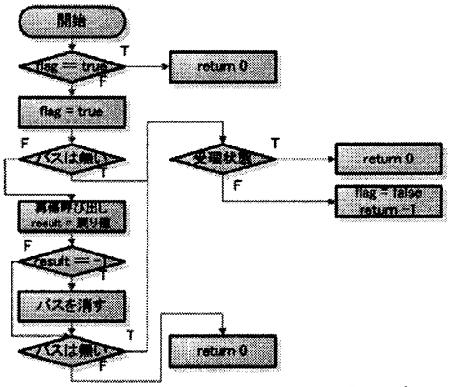

図５．　パスチェックアルゴリズム

上記のパスの削除を行い、モデルは図６-③の状態となる。次に不要な S の削除を行う。S の flag が true で無いものを削除する。flag のデフォルト値は false の為、$S0$ に繋がっていない状態は true になることは一度も無く、削除される。

上記のアルゴリズムを経て残った S、Sに代入されたパスとその op の集合 O、$S0$、A が図６-④の協調済 BPM である。

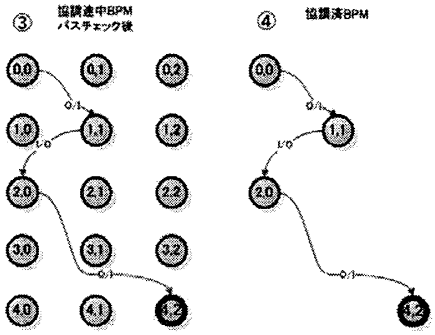

図６．協調変換アルゴリズム２

４．２　整合性検査

パスの代入を決定する整合性の検査は２つ存在する。

１つ目は、２つの BPM のオペレーションに対応するデータの送受パターンの検査である。取引にはデータの送信、受信、２つのパターンがあり、一方が送信の場合、もう一方は受信である必要がある。

２つ目（*tMatch* 関数）は、取引に使用するデータフォーマットの合致具合の検査である。検査結果は３値で表現され、*TRUE*, *FALSE*, *UNKOWN* の３つである。検査には t の名前空間、型名について検査を行うが名前空間については省略する。また、*tMatch* 関数は Web 上から与えられるものであり、検査アルゴリズムは今後の課題である。

５．　評価、まとめ

本研究は [2]の正規表現版プロトタイププログラムのテストケースと同様の結果を出力し、アルゴリズムの妥当性を検証した。

AWS における動的協調変換のアルゴリズム、妥当性について述べた。現在、整合性検査の新たなアルゴリズムを考案中である。本研究は科研費(19500095)の助成を受けたものである。

参考文献

1. Makoto Oya, Masahiro Kinoshita, and Yukinori Kakazu, On Dynamic Generation of Business Protocol in Autonomous Web Services, Systems and Computers in Japan, Vol.37, No.2, 2006, pp37-45
2. M. Oya, Autonomous Web Services Based on Dynamic Harmonization, IFIP I3E2008, Springer, ISBN:978-0-387-8590-2, pp.139-150, 2008
3. Extensible Markup Language (XML) 1.1 (Second Edition) http://www.w3.org/TR/xml11/
4. 高木, 木村, 伊東他, AWS ミドルウェアの研究, 情報処理学会第 71 回全国大会, 2009