4N-03 プロセスデータモデル

大矢野 潤 †
千葉短期大学

1 はじめに

インターネット上で展開されるサービスの利用者数の増大、アクセスポイントの広域化、利用形態の多様化に伴い、一つのサービスをさまざまな形態で提供する必要性が出てきている。また、ユーザの移動、複数の団体への所属にともない、一人のユーザに対応するデータが複数のドメインや、データベースに存在する傾向があり、データの一元管理という点に疑問が生じてきている。つまり、サービスは複数のサーバマシンによって提供され、さらにそのサービスの根底にあるデータの一元性、一貫性は一つのデータベース管理システム (以下、DBMS) の中だけではなく、サービス全体で保証される必要がある。しかし、それぞれのドメインのネットワークへの接続状況、ユーザ数、管理ポリシー、そして構成機器、搭載しているOSや、DBMSが異なるため、既存のRDB、OODBなどのモデルよりも粒度の細かい、動的な性質を記述できるデータモデルが必要となる。

本論文では、CCS(Calculus of Concurrenet System)[4] のプロセスに値をバインドし、そのプロセスの値に注目したプロセスデータモデルを提案し、その実装と応用について議論する。

2 CCSの拡張

本論文では、データモデルを記述する枠組としてCCSを用いる。CCSはR. Milnerによって提唱された並列プロセスの記述と、双模倣 (bi-simulation) によって定義される同値関係に基づいた代数的計算の枠組である。ここでは、まず、一般的なCCSのプロセスを定義し、その上でCCSを拡張していく。

2.1 CCS

CCSは一般的にはラベルつき遷移システム (labelled transition system) としてとらえることができ、次のように記述する。

$$\mathcal{T} = (S, T, \{\xrightarrow{t} : t \in T\})$$

ここで、S は状態の集合、T は遷移ラベルの集合、$\xrightarrow{t} \subseteq S \times S$ はそれぞれの $t \in T$ に対する遷移関係を表す。

そして、$A \xrightarrow{\alpha} A'$ は α アクションを選択することができ、その結果 A' に遷移できることを表している。さらに、「$E \xrightarrow{a} E'$ であるならば、E と F の合成である $E \mid F$ は $E' \mid F$ に遷移できる」ということを次の形で記述することにする。

$$\frac{E \xrightarrow{\alpha} E'}{E \mid F \xrightarrow{\alpha} E' \mid F}$$

CCSで利用する他の記号と意味は次のようになる。

(1)	$\mathrm{a}(x).E, \overline{\mathrm{a}}(e).E, \tau.E$	プリフィックス
(2)	$\Sigma_{i \in I} E_i$	非決定和
(3)	$E_1 \mid E_2$	合成
(4)	$E \backslash L$	制限 ($L \subseteq \mathcal{L}$)
(5)	$E[f]$	リラベリング
(6)	**if** b **then** E	条件

2.2 Valued CCS

ここでは、前述のCCSの遷移システムで定義されるプロセスに値をバインドした遷移システム $\mathcal{T}_{\mathcal{B}}$、およびその上で定義される値つきCCS(以下、VCCSとする) を定義しよう。

$$\mathcal{T}_{\mathcal{B}} = (S, T_f, \{\xrightarrow{t} : t \in T_f\}, \mathcal{B})$$

S を状態の集合、V を状態のとり得る値の集合とし、$\mathcal{B}$ は $S \to V$ の関数とする。ここで、$\mathcal{B}(s)$ は状態 $s \in S$ にバインドされている値を、逆に $\mathcal{B}(s) := v$ は状態 s に新しい値 $v \in V$ をバインド (assign) するものとする。また、通常のプログラミング言語のようにこの代入文は := の右から評価するものとするため、$\mathcal{B}(s) := 2 \times \mathcal{B}(s)$ は、現在状態 s に割り当てられている値を2倍したものを新たに s の値とするものとする。更に、表現 E についても同様の定義、つまり、$\mathcal{B}(E)$ も許すものとする。また、その表現を代表する状態を参照するための特別な定数をSelfとする。

次にトランジション T を拡張してみる。拡張したトランジション T_f は $l : f(x)$ という形をしており、l はラベル集合 $\mathcal{L}$ の要素、f は関数の集合 $\mathcal{F}$ の要素である。ここで $\mathcal{F}$ の要素には、通常の算術関数、前述のプロセスに値をバインドする関数、そして、集合を取り扱う関数から構成される関数であるとする。それぞれの関数については適宜導入していく。

ここで、特に $l : \{\mathcal{B}(\mathrm{Self}) := x\}(x)$ という形を $l(x)$ と書くことにする。また、特に関数を必要としない場合には単に l、$\overline{l}$ と書くことにする。

これらの議論を元に、遷移システムのセマンティクスは図1であたえる。

The Process Data Model
† Jun Oyano
Chiba Junior College
1-3-1 Konodai,Ichikawa,Chiba 272-8512,Japan

$$\mathbf{ACT}\ \frac{}{\alpha : f(x).E \xrightarrow{\alpha} E} \qquad \mathbf{Sum}_j\ \frac{E_j \xrightarrow{\alpha} E'_j}{\Sigma_{i\in I} E_i \xrightarrow{\alpha} E'_j}(j \in I)$$

$$\mathbf{Com}_1\ \frac{E \xrightarrow{\alpha} E'}{E \mid F \xrightarrow{\alpha} E' \mid F} \qquad \mathbf{Com}_2\ \frac{F \xrightarrow{\alpha} F'}{E \mid F \xrightarrow{\alpha} E \mid F'}$$

$$\mathbf{Com}_3\ \frac{E \xrightarrow{l} E' \quad F \xrightarrow{\bar{l}} F'}{E \mid F \xrightarrow{\tau} E' \mid F'}$$

$$\mathbf{Res}\ \frac{E \xrightarrow{\alpha} E'}{E\backslash L \xrightarrow{\alpha} E'\backslash L}(\alpha, \bar{\alpha} \notin L) \qquad \mathbf{Rel}\ \frac{E \xrightarrow{\alpha} E'}{E[f_l] \xrightarrow{f_l(\alpha)} E'[f_l]}$$

$$\mathbf{Con}\ \frac{P \xrightarrow{\alpha} P'}{A \xrightarrow{\alpha} P'}(A \stackrel{\text{def}}{=} P)$$

図 1: トランジションセマンティクス

例 1($\Sigma_{i=0}^{\infty} i$): ここで、次の B と C のプロセスがあるものとする。

$$\begin{aligned} B &\stackrel{\text{def}}{=} \text{in}_B : \{\mathcal{B}(B) := \mathcal{B}(B) + x\}(x).B \\ &\quad +\overline{\text{out}_B} : \{\mathcal{B}(B)\}.B \\ C &\stackrel{\text{def}}{=} \overline{\text{out}_C} : \{\mathcal{B}(C)\}.\tau : \{\mathcal{B}(C) := \mathcal{B}(C) + 1\}.C \end{aligned}$$

また、プロセス B、C にそれぞれ初期値としてバインドされている値 $\mathcal{B}(B)$、$\mathcal{B}(C)$ はともに 0 であるとする。このとき、これらのプロセスを合成したプロセス $(B \mid C)[\text{out}_C/\text{in}_B]$ の出力列は、終らない計算 $0 + 1 + 2 + \ldots$ の途中の値、例えば $0, 1, 1, 3, 6, 6, 6, 10, 15, 15, 21, \ldots$ などとなる。

3 関係モデルとの対応

ここでは、VCCS を使ってデータのセルを表現するプロセスと、そのプロセスの構造を決めるプロセスを記述することで、関係データモデルの機能である選択の動作を記述することを試みてみる。

例 2(選択): まず、データのセルとして振舞うプロセスを $B_i \stackrel{\text{def}}{=} \overline{out_{B_i}} : \{\mathcal{B}(B_i)\}.B_i$ とし、これらのプロセスからデータを収集するプロセス C、スキーマを決めるプロセス DB を次のように定義する。

$$\begin{aligned} C &\stackrel{\text{def}}{=} \Sigma_{j\in J}\text{in}_{C_j} : \{\mathcal{B}(C) := \mathcal{B}(C) \cup \{x\}\}(x).C \\ &\quad +\overline{\text{out}_C} : \{\mathcal{B}(C)\}.C \\ DB &\stackrel{\text{def}}{=} (\Pi_{i\in I} B_i \mid C)[\text{in}_{C_k}/\text{in}_{B_k}] \\ &\quad \textbf{where } k \in I \cap J \end{aligned}$$

このとき、$k \in I \cap J$ に対する B_k の持つ値が $\mathcal{B}(C)$ に収集される。この例ではプロセス C の値に集合の包含関係の上限が“意図した”値となる。しかし、例 1 と同様にプロセス DB は停止性の性質[1]を持たない。さらに、データのセルへの値の更新を許す表現を追加した場合[2]には、データの一貫性すら保証されないため、RDB の一貫性性質などの検証の必要がでてくる。

[1] mu-calculus[3] の言葉を使うと $\mu Z.([-]\text{ff} \vee \langle - \rangle Z)$。

4 実装

現在、本論文で提案した、VCCS の実行系のプロトタイプを開発中である。この実行系は次の部分から成る。

モデル: 前述の $\mathcal{T}_{\mathcal{B}}$ の構造をもつ perl のオブジェクトとして直接記述している。

インターフェース: モデルの正当性を検証するための補助ツールとして、モデルの状態を可視化し、実際にユーザが発火可能なトランジションのうち適当なものを実行できるインターフェイスを持つ。

コントロール: 遷移、通信可能なプロセスを選択し、実際に実行するディスパッチャーの部分である。

これらは perl-Tk と、perl に組み込まれている関数を用いて実装している。このため、通信部に通常の Socket を利用したり、また、実際のデータ管理部にファイルシステム、DBM、PostgreSQL などを利用しながら、実際に検証が必要なところのみを VCCS を用いて表現し、検証やデバッグを行なうことを目的としている。

5 まとめ

ネットワーク上に展開される多様な要求を充足するサービスは、一つのサービスを複数のサーバが協調して提供する仕組みが必要となる。反面、そのサービスの一貫性を、サーバの境界を越えて検証する必要性がでてくる。

本論文では、複数のプロセスが協調して動作するためのモデルを提案した。しかし、本研究の本来の目的[1, 2]は柔軟で安全なサービスの提供であり、数学的な検証論の展開と同時に、新たなサービスの実現がさらなる課題として残されている。

参考文献

[1] 大矢野 潤, “コンピュータ上の投票システムの開発”, 公共選択の研究 第 27 号,1996

[2] 大矢野 潤, 柏木 将宏 “層の概念を導入したデータベースの教育環境への応用”, 情報処理学会第 58 回全国大会講演論文集,1999

[3] Colin Stirling, “Modal and Temporal Logics of Processes”, Logics for Concurrency, LNCS Tutorial pp.149-237, Springer,1996

[4] Robin Milner, “Communication and Concurrency”, Prentice Hall,1989

[2]
$$\begin{aligned} B_i &\stackrel{\text{def}}{=} \overline{out_{B_i}} : \{\mathcal{B}(B_i)\}.B_i \\ &\quad +in_{B_i} : \{\mathcal{B}(B_i) := \mathcal{B}(B_i) \cup \{x\}\}(x).B_i \end{aligned}$$