実時間制約を考慮したマルチタスキングのモデル化

Modeling multi-tasking for real-time systems

清野 貴博* 緒方 和博† 二木 厚吉‡ 日比野 靖§

Summary. TBCMs are UNITY computational models evolved by introducing clock variables so as to model real-time systems. But, TBCMs are not suitable for modeling design details of real-time systems that take account of the number of processors used, their performance and scheduling policies. Therefore, we have extended TBCMs to define *fp*-TBCMs that can model such design details of real-time systems. This paper describes *fp*-TBCMs and their descriptions in CafeOBJ.

1 はじめに

隠蔽代数に基づく振舞仕様は，抽象機械などを自然に，かつ実装に依存しない高い抽象レベルで記述できる．振舞仕様は元来，オブジェクト指向におけるオブジェクトを記述するために設計されており，振舞仕様を記述する言語として CafeOBJ [1] [4] が提案されている．一方，いくつかの事例研究 [8] [10] により，振舞仕様を基にしたシステム記述が，並行・分散システムにも適切に応用できることを確認した．それら事例研究では，並行・分散システムを UNITY [3] 等の状態遷移機械でモデル化し，そのモデルを CafeOBJ で記述し，システムが望ましい性質を有することを CafeOBJ システム支援の下で検証している．さらに，時間の経過を表す時計変数を用いることで，状態遷移機械で実時間システムをモデル化できることが知られている [2] [7]．このため，並行・分散システムと同様に，実時間システムの記述にも，振舞仕様を基にしたシステム記述が適切に応用できることを，いくつかの事例研究によって確認した [9] [11]．それら事例研究では，UNITY に時間変数を導入した TBCM で実時間システムをモデル化し，そのモデルを CafeOBJ で記述し，システムが望ましい性質を有することを CafeOBJ システム支援の下で検証している．

一方で，実時間システムを実装する設計者は，プロセッサの数や性能，またオペレーティングシステムのスケジューリングポリシーなどの条件の下で，その実装が仕様で示された実時間制約を満たせるかどうかを検証しなければならない．TBCM では，それらの条件は明示的には示されていないため，どのように実装するかは，設計者に任されている．本稿では，TBCM を拡張し，これらの条件を考慮にいれて実時間システムをモデル化できる計算モデル *fp*-TBCM を提案するとともに，*fp*-TBCM でモデル化した実時間システムの CafeOBJ による記述方法，つまり，*fp*-TBCM に基づく振舞仕様の作成方法についても提案する．

*Takahiro Seino, 北陸先端科学技術大学院大学

†Kazuhiro Ogata, 北陸先端科学技術大学院大学

‡Kokichi Futatsugi, 北陸先端科学技術大学院大学

§Yasushi Hibino, 北陸先端科学技術大学院大学

2 実装を考慮した実時間システムのモデル: *fp*-TBCM

TBCM は，実装に関する情報，例えば，いくつのタスクに分割し，どんな性能のプロセッサをいくつ使って動かすかといった情報を持たない．逆に言うと，TBCM は十分な性能のプロセッサを無限個用いて動かした場合をモデル化していると見なすこともできる．以下では，TBCM を拡張し，ある性能のプロセッサを有限個用いて動かすような，より実装に近いモデル化ができる *fp*-TBCM を提案する．

fp-TBCM は，**プロセッサの集合** $\mathcal{P}$，*fp*-TBCM の状態空間 Σ を構成する**共有変数の集合** $\mathcal{V}$ ，**初期条件** $\mathcal{I}$ および**遷移規則の集合** $\mathcal{T} \cup \{tick\} \cup \{idle_p : p \in \mathcal{P}\} \cup \{update_\tau : \tau \in \mathcal{T}\}$ で定義される状態機械である．各 $\tau \in \mathcal{T}$ は一般に**遷移可能条件** c_τ を伴って定義され，c_τ が真のとき，τ が適用されると $\mathcal{V}$ が更新され，偽のときは，τ が適用されても $\mathcal{V}$ は変化しない．また，c_τ は後述するシステム変数 $\mathcal{D}$ のみが現れる**事前条件** p_τ を含んでいる．

$\mathcal{V}$ は**システム変数** $\mathcal{D}$ と**時計変数** $\mathcal{C}$ に分類され，$\mathcal{V} = \mathcal{D} \cup \mathcal{C}$ である．$\mathcal{C}$ は TBCM による拡張で，実時間制約を表現するために導入する変数である．$\mathcal{C}$ は now，u_τ および $l_\tau : \tau \in \mathcal{T}$ に加え，*fp*-TBCM では，$s_\tau : \tau \in \mathcal{T}$ がある．now はシステムの計算が始まってからの経過時間を保持し、型は $R^+ \cup \{0\}$ である．u_τ と l_τ は，それぞれ**上限**と**下限**と呼ばれ，τ の実行をしなければならない最も遅い時刻と，τ の実行ができる最も早い時刻を保持し，型は，それぞれ $R^+ \cup \{\infty\}$ と $R^+ \cup \{0\}$ である．s_τ は，上限と下限は，**最大遅延時間** d_{max_τ} と**最小遅延時間** d_{min_τ} という定数から設定される．τ のスケジューリングを制御するための変数であり，型は $R^+ \cup \{0, \infty\}$ である．$p \in \mathcal{P}$ において，τ が実行可能なときは，τ が実行できるようになる時刻を，そうでない時は，∞ を保持する．

$\mathcal{I}$ は，$\mathcal{D}$ に対する初期条件に加え，以下に示す $\mathcal{C}$ に対する各条件を満たす．

- $now = 0$.
- 各々の $\tau \in \mathcal{T}$ について，p_τ が真の時，$u_\tau = d_{max_\tau}$，$l_\tau = d_{min_\tau}$ および $s_\tau = 0$ と設定する．偽ならば $u_\tau = \infty$ ， $l_\tau = 0$ および $s_\tau = \infty$ とする．

fp-TBCM はマルチタスキングを表現できなければならない．*fp*-TBCM では，各タスクを遷移規則としてモデル化する．各タスクはそれが実行されるプロセッサや，その優先度が静的に決定されていることを仮定する．また，タスクの優先度に基づいたスケジューリングポリシーを採用する．以下では，優先度 $r \in \{Lo, Mid, Hi\}$ を持つ遷移規則を τ^r ，その集合を $\mathcal{T}^r$ とする．またプロセッサの集合は $\mathcal{P}$ と表記し、プロセッサ $p \in \mathcal{P}$ で動作する遷移規則を τ_p，その集合を $\mathcal{T}_p$ とする．また，$\tau_p^r \in \mathcal{T}_p^r$ のようにも表記し，これは p 上で動作する優先度 r を持つ遷移規則およびその集合として用いる．

τ_p は，それが実行可能となるとき，言い換えると，τ_p の事前条件 p_{τ_p} が真になったとき，τ_p の実行に必要なすべてのデータは揃っており，τ_p はそれらに基づき決定的な計算を行う．*fp*-TBCM では，τ_p の実行時間を e_{τ_p} と表現し，これはプロセッサ p の性能を表現しているとみなせる．τ_p が実行された時刻を t とすると，$t + e_{\tau_p}$ の時刻まで，τ および他の遷移規則 $\mathcal{T}_p$ を新たに実行することはできない．また，ある状態において，$\mathcal{T}_p$ に事前条件が真である遷移規則が複数あるとき，より優先度の高い遷移規則が選ばれ，実行される．また，事前条件が真である遷移規則のうち，最も

表**1**　状態 s で τ が適用され s' に遷移したときの，τ に対する u_τ ，l_τ と s_τ の設定

	$p_\tau(s') = \text{false}$	$p_\tau(s') = \text{true}$
$p_\tau(s) = \text{true}$	$u_\tau := \infty$ $l_\tau := 0$ $s_\tau := \infty$	$u_\tau := now + d_{max_\tau}$ $l_\tau := now + d_{min_\tau}$ $s_\tau := now + e_\tau$

表**2**　状態 s で τ が適用され s' に遷移したときの，$\tau'(\neq \tau)$ に対する $u_{\tau'}$ ，$l_{\tau'}$ と $s_{\tau'}$ の設定

	$p_{\tau'}(s') = \text{false}$	$p_{\tau'}(s') = \text{true}$
$p_{\tau'}(s) = \text{true}$	$u_{\tau'} := \infty$ $l_{\tau'} := 0$ $s_{\tau'} := \infty$	$u_{\tau'}$ (変化させない) $l_{\tau'}$ (変化させない) $s_{\tau'} := now + e_\tau$
$p_{\tau'}(s) = \text{false}$	$u_{\tau'} := \infty$ $l_{\tau'} := 0$ $s_{\tau'} := \infty$	$u_{\tau'} := now + d_{max_{\tau'}}$ $l_{\tau'} := now + d_{min_{\tau'}}$ $s_{\tau'} := now + e_\tau$

優先度が高い遷移規則が複数ある時は，それらのうち，どちらかが選ばれる．実行できる遷移規則が全くない時は，特別な優先度 *Lowest* を持った遷移規則 $idle_p^{Lowest}$ が実行され，プロセッサがアイドル状態であることを表現する．

τ_p は，それがモデル化している計算の実行結果を変数に書き戻す．τ_p が更新する変数が $\mathcal{T}_p$ からのみ参照されているならば，*fp*-TBCM では，τ_p が変数を更新するタイミングは，$t + e_{\tau_p}$ の時刻までであれば，いつでもよい．そうでない時は，変数の更新を表現する特別な遷移規則 $update_{\tau_p}$ を加える．

上記を実現する $\tau_p \in \mathcal{T}_p$ は次のようになる．c_{τ_p} は p_{τ_p} ，TBCM による実時間制約 $l_{\tau_p} \leq now$ に加え，次の2つの不等式を結んだものである．1つは，$s_{\tau_p} \leq now$ である．2つ目は各 $\tau'_p \in \{\tau | \tau \in \mathcal{T}_p^{r'} \wedge r \prec r'\}$ に対し $now < s_{\tau'_p}$ である．また，$\tau \in \mathcal{T}$ による $\mathcal{C}$ の設定は，表1および表2の通りである．なお，これらの表では，p_τ が偽であった時は，$\mathcal{V}$ は不変であるので，省略している．また，$idle_p^{Lowest}$ は，$p_{idle_p^{Lowest}}$ が常に真であり，$u_{idle_p^{Lowest}}$ および $l_{idle_p^{Lowest}}$ を持たないことを除けば，$\mathcal{T}$ と同じ定義である．

tick は 時間の進みを表現する遷移規則である．*tick* は，各$\tau \in \mathcal{T}$ の u_τ および s_τ を超えない任意の時刻まで now を進めることができる．

$update_\tau$ は，次のように定義する．$update_\tau$ に対して，変数 p_{update_τ}，u_{update_τ} および l_{update_τ} が与えられる．初期状態では p_{update_τ}，u_{update_τ} および l_{update_τ} は それぞれ偽，∞ および 0 である．τ の実行によって，p_{update_τ} が真となる．この時，u_{update_τ} と l_{update_τ} は，表1に示すように，設定される．c_{update_τ} は $p_{update_\tau} \wedge l_{update_\tau} \leq now$ であり，この条件を満たす時，$update_\tau$ は目的の変数を更新し，p_{update_τ} は偽に，u_{update_τ} と l_{update_τ} は，表1に示すように，設定される．

3　CafeOBJ による *fp*-TBCM の記述

CafeOBJ は主に**始代数** (*initial algebra*) と**隠蔽代数** [6] (*hidden algebra*) に基づいている．始代数は整数やスタックなどの抽象データ型の記述に，隠蔽代数はオブジェ

クト指向のオブジェクトの記述に用いる．始代数では**可視ソート**がデータを，隠蔽代数では**隠蔽ソート**がオブジェクトの状態空間を表現する．隠蔽ソート上には，オブジェクト指向でのメソッドに対応する 2種類の演算を定義する．**操作演算** (*action*) はオブジェクトの状態を変える演算であり，引数に 1つの隠蔽ソートと可視ソートで表現された 0個以上のデータを取り，別の状態を表現する (引数の隠蔽ソートと同じ) 隠蔽ソート上の項を返す．**観測演算** (*observation*) はオブジェクト中のデータを観測する演算であり，引数に 1つの隠蔽ソートと可視ソートで表現された 0個以上のデータを取り，オブジェクト中のデータを返す．

オブジェクトを遷移機械とみなすと，*fp*-TBCM は CafeOBJ で自然に記述できる．変数は観測演算によって，それらの初期値は等式によって表現できる．遷移規則は操作演算によって，それらの振舞いは等式によって表現できる．

4 まとめと今後の課題

本稿では，実時間制約を記述できる TBCM を拡張した *fp*-TBCM を定義し，有限のプロセッサ上で，遷移規則の優先度や実行時間を考慮に入れたモデルを提案した．また，*fp*-TBCM を実際に記述する言語として CafeOBJ を選び，*fp*-TBCM を記述する方法を示した．

TBCM から *fp*-TBCM への拡張の際には，CCS を基本にしたマルチタスキングのモデル化 [5] を参考にした． [5] では *fp*-TBCM における下限にあたる変数があるが，上限がないため，実時間制約の検証には適さない．一方で， [5] ではプロセッサの横取りを扱っているが，*fp*-TBCM には取り入れていない．*fp*-TBCM での横取りの扱いは今後の課題である．記述した *fp*-TBCM が *non-Zeno* であることは，重要な性質である．記述した *fp*-TBCM が *non-Zeno* であるための十分条件か，もしくは個々のモデルが *non-Zeno* を満たすことを証明する手法の提案も，今後の課題である．

参考文献

[1] CafeOBJ web page. `http://www.ldl.jaist.ac.jp/cafeobj/`.

[2] M. Abadi and L. Lamport. An old-fashioned recipe for real time. *ACM TOPLAS*, 16(5):1543–1571, 1994.

[3] K. M. Chandy and J. Misra. *Parallel Program Design: A Foundation*. Addison-Wesley, 1988.

[4] R. Diaconescu and K. Futatsugi. *CafeOBJ report*. Number 6 in AMAST Series in Computing. World Scientific, 1998.

[5] C. J. Fidge. The algebra of multi-tasking. In *Proceedings of AMAST 2000 (LNCS 1816)*, pages 213–227. Springer-Verlag, 2000.

[6] J. Goguen and G. Malcolm. *A hidden agenda*. Number 245 in Theoretical Computer Science. 2000.

[7] N. A. Lynch. *Distributed Algoritms*. Morgan Kaufmann, 1996.

[8] K. Ogata and K. Futatsugi. Formal verification of the MCS list-based queuing lock. In *Proceedings of ASIAN'99 (LNCS 1742)*, pages 281–293. Springer-Verlag, 1999.

[9] K. Ogata and K. Futatsugi. Specifying and verifying a railroad crossing with CafeOBJ. In *6th International Workshop on Formal Methods for Parallel Programming: Theory and Applications (Proc. of IPDPS'2001, IEEE Computer Sciety)*, 2001.

[10] T. Seino, K. Ogata, and K. Futatsugi. Specification and verification of a single-track railroad signaling in CafeOBJ. *IEICE TRANSACTIONS on Fundamental of Electronics, Communications and Computer Science*, E84-A(6):1471–1478, 2001.

[11] T. Seino, K. Ogata, and K. Futatsugi. Specification and verification of tokenless blocking systems with CafeOBJ. In *Proceedings of ITC-CSCC 2001*, volume II, pages 807–811, 2001.