動的環境におけるエージェント知識の効率的モデル検査法

梅村晃広[†,☆] 勝野裕文[†,☆☆]

有限状態の遷移で記述される動的環境下で一連の事象が起きた後に，その環境にいるエージェントがどのような知識を持つかという問題を，van der Meydenは様相論理式のモデル検査の形で定式化した．彼は，完全な記憶を持つ分散同期システムにおいて，状態系列 r が与えられたとき，その最終時点で知識様相命題が真か偽かの判定（モデル検査）を，r の長さ（つまり状態変化の回数）$|r|$ について $O(|r|)$ の時間計算量で計算する方法を示した．しかし，$|r|$ の係数は状態の数 $|W|$ に依存し，本論文で示すように最悪の場合 $|W|$ に関しては指数オーダ以上で増大する．本論文では，上と同様の設定において，「知らない」ということが表せないように制限された知識命題論理式（単調知識命題論理式）に限定した場合に，時間計算量が $|r|$ に関して線形時間であると同時に $|W|$ に関して多項式時間であるモデル検査が可能であることを示す．

An Efficient Model Checking Method of Agent Knowledge under Dynamic Environments

AKIHIRO UMEMURA[†,☆] and HIROFUMI KATSUNO[†,☆☆]

We study the problem of deciding what knowledge an agent in a dynamic environment with finite states has after a series of events. van der Meyden formalized this as the model checking problem of modal formulas by representing the environment as a synchronous distributed system with perfect recall. He showed that the time complexity of his decision algorithm is linear in $|r|$, where $|r|$ is the length of a run r (i.e., the number of events). But the time complexity depends on $|W|$, where $|W|$ is the number of states. We show that the worst case time complexity increases in at least exponential order with respect to $|W|$. We present that it can be decided in time polynomial in $|W|$ and linear in $|r|$ if we restrict modal formulas to cases where 'not knowing' cannot be represented.

1. はじめに

ここ数年で，計算機ネットワークは一般の人にとっても身近なものとなりつつある．専門家ではない多くの人が計算機ネットワーク上の情報を有効に利用しようとするとき，その情報処理を担うエージェントにはより知的であることが求められる．これから，多くのエージェントが混在する場において，それぞれのエージェントが行動のための独自の判断を行うという計算形態の研究の重要性は日ごとに増大しているといえる[5),6)]．しかしながら，こういった新しい計算形態を実現するうえで必要な要素技術の考察はまだ十分に行われていない．たとえば，動的に変化する環境の中で，複数エージェントが相互の知識・信念を計算できれば，それらのエージェントの間のコミュニケーションが円滑に進むことが期待できるが，その計算の具体的方法はまだ部分的にしか解明されていない．本論文では，動的環境下でエージェント相互の知識を効率的に計算する問題を議論する．

有限状態の遷移で記述される動的環境下で一連の事象が起きた後に，その環境にいるエージェントがどのような知識を持つかという問題を，van der Meyden[7),8)]は様相論理式のモデル検査の形で定式化した．彼はFaginら[1)]にならって，各エージェントが完全な記憶を持つ（perfect recall）分散同期システムの問題を取りあげ，そのシステムを様相論理の有限ストラクチャと結び付けている．そして，このシステムの状態系列（run）r が与えられたとき，その最終時点で知識様相命題 φ が真か偽かの判定（モデル検査）を，$O(|r|)$（すなわち $C \cdot |r|$）の時間計算量で計算する方法を示した．

† NTT基礎研究所
NTT Basic Research Laboratories
☆ 現在，株式会社NTTデータオープンシステムセンタ
Presently with Open Systems Center, NTT DATA Corporation
☆☆ 現在，NTTコミュニケーション科学基礎研究所
Presently with NTT Communication Science Laboratories

本論文では，上の評価で使われる定数 C に着目し，状態の数 $|W|$ に関しては最悪の場合 C は指数関数以上の速さで増大する関数であることを示す．この結果は，van der Meydenが提案したモデル検査法を状態数が多い場合にエージェントの知識の計算に使うことは現実的でないことを示唆している．本論文では，「知らない」ということが表せないように制限された知識命題論理式を単調知識命題論理式と呼ぶことにする．そして，単調知識命題論理式に限れば，上のモデル検査が $|W|$ に関して多項式時間のオーダで行えることを示す．

2. 準　　備

本論文は文献7)，8)で提案された枠組みに基づくものである．この章では文献7)，8)の枠組みを紹介する．なお後の章での議論を行いやすくするため，いくらか変更してある部分もある．

2.1 知識命題

n エージェントについての知識命題を定義する（本論文では，$n \geq 2$ の場合のみ考える）．A を1から n までの整数，すなわちエージェントの集合とする．基本命題の集合を $\mathcal{P}$ とする．基本命題から n 個の知識様相記号 K_i（ただし $i \in A$）と $\wedge$，$\neg$ を用いて作られる論理式を**知識命題**とする．知識命題のうち，知識様相記号が現れない命題を**平命題**と呼ぶ．平命題から知識様相記号と $\wedge$ で構成される知識命題（つまり，知識様相記号の外側に $\neg$ が現れない知識命題）を**単調知識命題**と呼ぶ．

知識命題 φ の中の基本命題，論理記号，様相記号の数の総和を φ の**大きさ** $|\varphi|$ とする．たとえば，$A = \{1,2\}$ で p，q が基本命題であるとき，知識命題 $\varphi_1 = K_1(K_2 p \wedge K_2 K_1 \neg q)$ は単調知識命題であり，$|\varphi_1| = 8$ である．また，$\varphi_2 = K_1 \neg K_1 p$ は単調知識命題ではなく，$|\varphi_2| = 4$ である．

次に知識命題に対して，知識様相の交代深さを定義する．これを厳密に行うために，以下の手順をとる．まず，各知識命題 φ に対して，エージェントと自然数の組の集合を対応付ける関数 f を以下のように定義する．

(1) φ が基本命題のとき，
$$f(\varphi) = \{\langle 1,0\rangle, \ldots, \langle n,0\rangle\}.$$
(2) $f(\neg\varphi) = f(\varphi)$.
(3) $f(\varphi \wedge \varphi') = f(\varphi) \cup f(\varphi')$.
(4) $f(K_i\varphi) = \{\langle i,m\rangle \mid \langle i,m\rangle \in f(\varphi)\} \cup \{\langle i,m+1\rangle \mid \exists j \in A.j \neq i \wedge \langle j,m\rangle \in f(\varphi)\}$.

このとき φ に対して $f(\varphi)$ の要素であるすべての組 $\langle i,m\rangle$ の中での m の最大値を φ の**知識様相の交代深さ**，あるいは単に**深さ**といい，$KD(\varphi)$ と表す．φ が平命題でない場合 $KD(\varphi)$ は直観的には次のような数に対応している．知識様相記号に注目して φ の外側から内側へたどる（一般に複数のたどり方がある）ときにできる知識様相記号の列ごとに，その列の中で隣りあう知識様相記号が異なる回数を求め，その回数の最大値が $KD(\varphi) - 1$ である．

たとえば，先述の φ_1，φ_2 に対して，$KD(\varphi_1) = 3$，$KD(\varphi_2) = 1$ である．$KD(\varphi_2)$ の値は2ではないことに注意されたい．

2.2 環境と Kripke 構造

本論文では，動的な環境の下でのエージェントの知識を議論する．環境は形式的に以下のように定義される．

定義1 **環境** $E = (W, T, V, \sim_1, \ldots, \sim_n)$ は次のものから成る．

- W は状態の有限集合である．
- T は W 上の二項関係，すなわち $T \subset W \times W$ であり，状態 w_1 から状態 w_2 へ遷移可能であるとき $\langle w_1, w_2\rangle \in T$ とする．
- $V : W \times \mathcal{P} \to \{\mathtt{true}, \mathtt{false}\}$ は，状態と基本命題を引数としてとり，真理値を返す関数である．すなわち V は各状態における基本命題の真偽を定めるものである．
- 各 $\sim_i$ は W 上の同値関係である．エージェントはそれぞれ自分のまわりしか観測できないので，2つの状態の w_1 と w_2 の違いがエージェント i の観測範囲外であるとき，i は w_1 と w_2 を区別できない．このような各エージェントの観測可能性を定めるのがこの同値関係である．エージェント i にとって状態 w_1 と状態 w_2 が区別できないとき，$w_1 \sim_i w_2$ とする．

例2 以降の議論に使用する環境の例をここで定義しておく（**図1**）．エージェント1，2，3があり，状態の集合を $W = \{w_1, w_2, w_3\}$ とする．遷移可能関係を表す集合 T は $\langle w_1, w_2\rangle$，$\langle w_1, w_3\rangle$，$\langle w_2, w_3\rangle$，$\langle w_3, w_1\rangle$ から成る．基本命題として p，q があり，V は $V(w_1, p) = V(w_3, q) = \mathtt{true}$ で，それ以外はすべて $\mathtt{false}$ とする．また各エージェントにとっての状態の同値関係 $\sim_i$ は次のように定義する．$w_1 \sim_1 w_2$，$w_2 \sim_2 w_3$，$w_3 \sim_3 w_1$ で，それ以外は自分自身を除いて同値ではないものとする．

状態の集合 W の要素の有限列 $w_1 w_2 \cdots w_m$ で，隣りあうすべての2つの状態 w_j と w_{j+1} が $\langle w_j, w_{j+1}\rangle \in T$ であるものを**状態系列**（run）と呼ぶ．環境 E 下

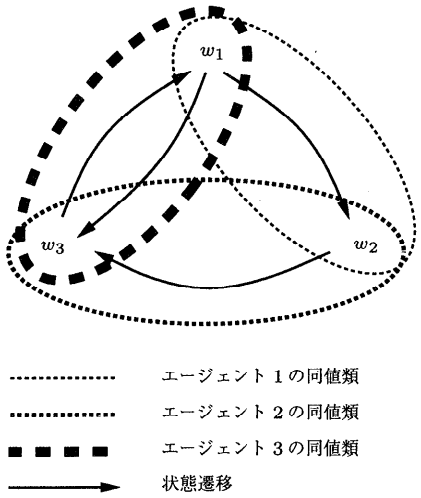


図 1 環境の例
Fig. 1 Example of an environment.

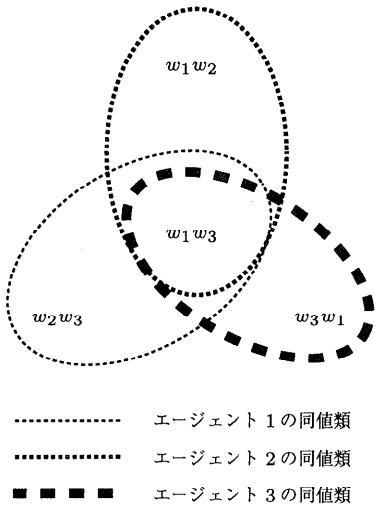


図 2 長さ 2 の状態系列の Kripke 構造
Fig. 2 Kripke structure of runs in length 2.

での状態系列全体の集合を $Runs(E)$ とする．状態系列 $r = w_1w_2 \cdots w_m$ に対して m を r の**長さ**と呼び，$|r|$ で表す．またこのとき，各 w_j を $r[j]$ と書く．さらに r の最後の状態 w_m を $last(r)$ で表す．長さが m である状態系列全体の集合を $Runs_m(E)$ とする．

環境 $E = (W, T, V, \sim_1, \ldots, \sim_n)$ に対する Kripke **構造** $M(E)$ は

$$M(E) = (Runs(E), V', \mathcal{K}_1, \ldots, \mathcal{K}_n)$$

とする．ここで $V' : Runs(E) \times \mathcal{P} \to \{\mathtt{true}, \mathtt{false}\}$ は $V'(r,p) = V(last(r), p)$ と定義する．また各 $\mathcal{K}_i$ $(1 \leq i \leq n)$ は，$Runs(E)$ 上の同値関係であり，次のように定義される☆：$r\mathcal{K}_i r'$ iff $|r| = |r'|$ かつ，すべての $1 \leq j \leq |r|$ について $r[j] \sim_i r'[j]$ であること．

例 3 例 2 の環境から作られる状態系列の Kripke 構造のうち，長さが 2 である状態系列に関係する部分だけを**図 2** に示す．

「Kripke 構造のもとで状態系列が知識命題を満たす」という関係が通常のように以下のとおり定義される．

定義 4 環境 E に対する Kripke 構造 $M = M(E)$ と状態系列 r，および知識命題 φ が与えられたときに，$M, r \models \varphi$ を φ の構造について帰納的に定義する：

- 基本命題 p に対して $M, r \models p$ iff $V'(r,p) = \mathtt{true}$.
- $M, r \models \neg\varphi$ iff $M, r \not\models \varphi$.
- $M, r \models \varphi \wedge \psi$ iff $M, r \models \varphi$ かつ $M, r \models \psi$.
- $M, r \models K_i\varphi$ iff $r\mathcal{K}_i r'$ を満たすすべての r' に対して $M, r' \models \varphi$.

上で定義した Kripke 構造 $M(E)$ による意味論は，$S5_n$ と呼ばれる公理系によって健全で完全な公理化ができることが知られている[1]．

☆ 本論文では，A は B の必要十分条件であることを A iff B と略記する．また定義の文脈で A iff B と書いた場合には A の定義を B とすることを表す．

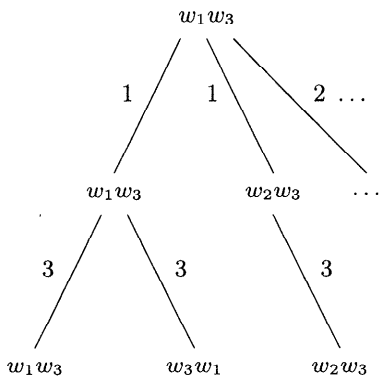


図 3 長さ 2 の状態系列の木
Fig. 3 Tree of runs in length 2.

2.3 木構造によるモデル検査

定義 4 に基づいて，モデル検査によって Kripke 構造上での知識命題の真偽を決定する場合を考える．例として，例 2 の環境から構成される状態系列の Kripke 構造 M において，$M, w_1w_3 \models K_1K_3q$ の計算を試みよう．このとき，**図 3** に示したように，節点が状態系列を表し，枝がエージェント名でラベル付けされている木構造を用いることができる．ここで，エージェント名 i のラベルが付けられた枝は $\mathcal{K}_i$ により同値となる状

態系列どうしを結ぶものである．$M, w_1w_3 \models K_1K_3q$ の評価は，根から1でラベル付けされた枝で到達可能な状態系列からさらに3でラベル付けされた枝で到達可能なすべての状態系列おける q の真偽値を調べることによって得られる．

ここで，長さ l の状態系列の総数は最悪の場合 l について指数関数的に増大するので，状態系列を節点とする木を使ったモデル検査法では，l について指数関数の時間がかかる場合があると見積もられる．ところが，van der Meyden[7]は，本論文のように状態系列の最終状態によって決まる基本命題のみを扱う場合，節点が状態系列ではなく最終状態を表す木を考えればよいことを示し，その結果同一視できる枝を刈ることにより，l について線形時間でモデル検査ができることを示した．以下では，この木構造を定義する．

なお，Kripke 構造をそのまま使わずに，木構造を導入する理由は，1つ前の時点の構造をもとにして今の時点の構造を得ることが簡単にできるからである．このことは，2.4 節で導入される **update** という関数で具体的に立証されている．

今，技術的な都合からエージェントの集合 A に0という擬似エージェントを加えた $A_0 = A \cup \{0\}$ を考え，これを用いて木の集合を定義する（この擬似エージェントは，ここでの木の定義や後の重子木の定義に用いるだけである．様相記号としての K_0 は考えない）．

定義 5 W を状態の集合とするエージェント $i \in A_0$ についての深さ k の木の集合 $\mathcal{T}_{k,i}(W)$ を k について帰納的に定義する☆：

- $\mathcal{T}_{0,i}(W) = W$;
- $\mathcal{T}_{k+1,i}(W) = \{\langle w, C_1, \ldots, C_n\rangle \mid w \in W, C_j \subset \mathcal{T}_{k,j}(W),\ i \neq 0 \text{ ならば } C_i = \emptyset\}$．ここで $\emptyset$ は空集合である．

特に $\mathcal{T}_{k,0}(W)$ を $\mathcal{T}_k(W)$ と省略する．$k \geq 1$ のとき，$t \in \mathcal{T}_{k,i}(W)$ は $t = \langle w, C_1, \ldots, C_n\rangle$ と書けるが，w を t の**根**，各 $t' \in C_j$ を t の j-**子**と呼び，それぞれ $w = root(t)$，$C_j = Chld_j(t)$ と書く．ただし便宜上，深さ0の木 w に対しては $root(w) = w$ とし，各 $Chld_j(w)$ は空集合とする．

上記の木の定義で，$i = 0$ のとき以外は i-子の集合を空集合としていることに注意する．これは，一般に $S5_n$ では「$K_iK_i\varphi \iff K_i\varphi$」がつねに成立することを利用して，検査する木の大きさをより小さくする工夫である．このために，次の関係 $\models$ の定義が通常より複雑なものとなる☆☆．

定義 6 知識命題 φ が $KD(\varphi) \leq k$ で $t \in C \subset \mathcal{T}_{k,i}(W)$ であるとき（$i \in A_0$），$t, C \models_i \varphi$ を次のように定義する：

- 基本命題 p に対して $t, C \models_i p$ iff $V(root(t), p) =$ **true**.
- $t, C \models_i \neg\varphi$ iff $t, C \not\models_i \varphi$.
- $t, C \models_i \varphi \wedge \phi$ iff $t, C \models_i \varphi$ かつ $t, C \models_i \phi$.
- $t, C \models_i K_i\varphi$ iff すべての $t' \in C$ について $t', C \models_i \varphi$ である．
- $t, C \models_i K_j\varphi$（ただし $i \neq j$）iff すべての $t' \in C$ について，すべての $t'' \in Chld_j(t')$ について $t'', Chld_j(t') \models_j \varphi$ である．

特に $t, \{t\} \models_0 \varphi$ を $t \models \varphi$ と書く．

直観的には上記の定義における $C \subset \mathcal{T}_{k,i}$ と $\models_i$ は，t が検査されている文脈を保存するものであり，同じ様相記号が二段続く場合を特別扱いしている．

状態系列 r での命題のモデル検査に用いる木を定める．$k \geq 0$ と $i \in A_0$ について $\mathrm{kt}_{k,i}(r) \in \mathcal{T}_{k,i}(W)$ を k について帰納的に以下のように定義する：$\mathrm{kt}_{0,i}(r) = last(r)$; $\mathrm{kt}_{k+1,i}(r) = \langle last(r), C_1, \ldots, C_n\rangle$，ただし $i \neq j$ である j について $C_j = \{\mathrm{kt}_{k,j}(r') \mid r\mathcal{K}_jr'\}$ で，$i \neq 0$ ならば C_i は空集合とする．特に $\mathrm{kt}_{k,0}(r)$ を $\mathrm{kt}_k(r)$ と書く．

このとき，文献7)で示されているように，次の命題が成立する．

命題 7 知識命題 φ について $KD(\varphi) \leq k$ である任意の k について $M, r \models \varphi$ iff $\mathrm{kt}_k(r) \models \varphi$ である．

図 4 は例3の条件下での $\mathrm{kt}_2(w_1w_3)$ である（根から1–3のパスでたどることができる枝以外は省略してある）．

2.4 木の更新

van der Meyden[7]はモデル検査を効率的に行うために，$\mathrm{kt}_{k,i}(r)$ を r の長さ方向に1ステップずつ更新していく方法を提案した．今，木 t から各 $w \in W$ ごとに木を作る操作 $\mathbf{update}(t, w)$ を t の深さについての帰納法で定義する：

- $\mathbf{update}(w', w) = w$;
- $\mathbf{update}(\langle w', C_1', \ldots, C_n'\rangle, w) = \langle w, C_1, \ldots, C_n\rangle$，ただしここで $C_i = \{\mathbf{update}(t', v) \mid t' \in C_i', v \sim_i w, \langle root(t'), v\rangle \in T\}$ である．

このとき，文献7)で示されているように，次の命題が成立する．

☆ ここで定義している木は，$i = 0$ のとき文献7)での k-tree にあたり，$i \neq 0$ のとき文献7)での i-objective k-tree にあたる．

☆☆ 文献7)は，その最終版[8]で定義の方法を変えているが，本質的な違いではない．以後の議論をより簡単に進めるため，ここではもとの定義方法を採用した．

命題 8 $\mathbf{update}(\mathrm{kt}_k(r), w) = \mathrm{kt}_k(rw)$ である．

命題 7 は，$M, r \models \varphi$ を決定するために $\mathrm{kt}_k(r)$ が使えることを示し，命題 8 は $\mathrm{kt}_k(w_1)$ から始めて，$\mathrm{kt}_k(r'w) = \mathbf{update}(\mathrm{kt}_k(r'), w), \ldots$ と **update** を使って順番に $\mathrm{kt}_k(r)$ を生成できることを示す．

図 5 に **update** の例を示す．この図は，一部の枝のみ表示したものである．

2.5 モデル検査の時間計算量

文献 7) は，$\mathbf{update}(\mathrm{kt}_k(r'), w)$ の 1 ステップが $|r'|$ に依存しない定数でできることから，$\mathrm{kt}_k(r)$ の生成が $O(|r|)$ ででき，したがって，$M, r \models \varphi$ が $\mathrm{kt}_k(r)$ を使って $O(|r|)$ で決定できることを示している．しかし $|r|$ 以外のパラメータに関する考察はなされていない．本論文では以下で $|W|$ について注目する．

まず，この議論を行うために木 t の**大きさ** $size(t)$ を，木の深さ k についての帰納法で定義する：$k = 0$ のとき $size(t) = 1$ とする；$k = k' + 1$ のとき $t = \langle x, C_1, \ldots, C_n \rangle$ とおける．各 C_i について $forest\_size(C_i) = \sum_{t' \in C_i}(size(t'))$ とする．このとき

$$size(t) = \Sigma_{1 \le i \le n} forest\_size(Chld_i(t)) + 1$$

とする．

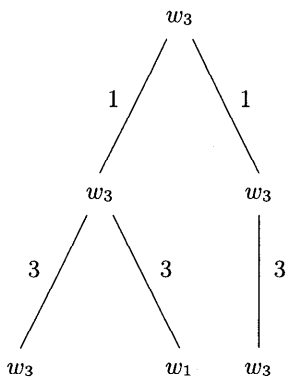


図 4 木 $\mathrm{kt}_2(w_1w_3)$
Fig. 4 The tree $\mathrm{kt}_2(w_1w_3)$.

一般に Kripke 構造に対応する木の大きさは状態の数 $|W|$ が増大するにつれ，$|W|$ に関する指数オーダ以上の速さで急速に増大することが予想される．これは木の総数すなわち $|\mathcal{T}_k(W)|$ の上界が文献 7) に示されているように，$|W|$ に関して急速に増大するからである．

しかし，木を $\mathrm{kt}_k(r)$ に限定すると，その大きさの評価は慎重に行わなければならない．なぜならば，状態遷移 T によって $\mathrm{kt}_k(r)$ として生成される木の種類は限定されているので，大きさの小さい木しか出現しない可能性があるからである．

付録 A.2 では，r の長さが十分に長いときには $\mathrm{kt}_k(r)$ の大きさが最悪の場合少なくても $2^{|W|}$ のオーダで増大することを示している．このことから，$\mathbf{update}(\mathrm{kt}_k(r), w)$ の計算は最悪の場合少なくても $O(2^{|W|})$ の時間がかかることになる．

3. 単調知識命題の効率的なモデル検査

まず，計算にかかる時間についてより正確な議論をするために，2.3 節で定義した木をあらためて見直す．2.3 節では，木は子木の集合をその構成要素として持っていたが，実際に集合を作る操作をインプリメントするためには，2 つの要素の同一性をチェックする必要がある．ここで集合の要素も木であるから，再帰的にチェックすることが必要になり，結局同一性のチェックには木の大きさの程度の時間がかかることが推定さ

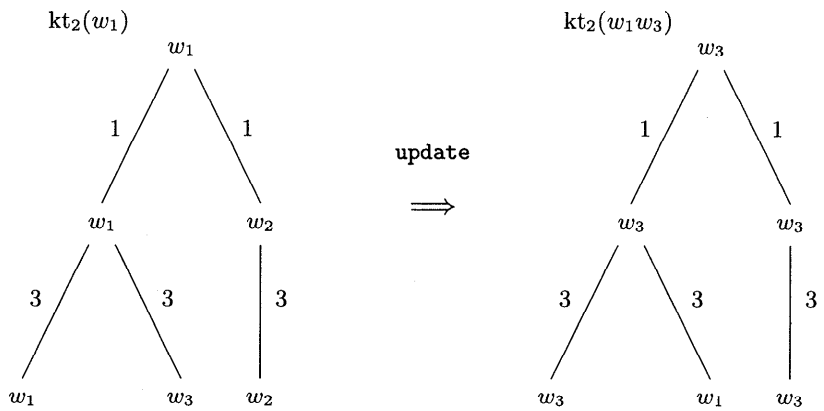


図 5 木の更新の例
Fig. 5 Example of tree update.

れる．

本章ではこの問題を避けるため，2つの木の同一性のチェックを直接には行わないで済む構造を定義する．そのためにマルチセットに基づいた木を使う．ここではこれを重子木と呼ぶ．これに対して2.3節で定義したものを単に木と呼ぶ．

本章では，まず，重子木とそれによる命題のモデル検査を定義する（3.1節）．次に，単調知識命題についてのモデル検査結果を変化させない，重子木の木への変形操作 *pack* を定義する（3.2節，3.3節）．そして，この *pack* を使って，木の更新手続きを定義する（3.4節）．最後に，この手続きによるモデル検査の時間計算量を議論する（3.5節）．

3.1 重子木の定義

一般に X を有限集合とするとき，$Multi(X)$ を X 上の有限マルチセットの集合とする（つまり $Y \in Multi(X)$ の中には，X のそれぞれの要素が0回以上有限回出現する．任意の $x \in X$ に対して $Y_1 \in Multi(X)$ における x の出現回数と $Y_2 \in Multi(X)$ における x の出現回数が一致するとき $Y_1 = Y_2$ とする）．$x \in X$ と $Y \in Multi(X)$ について $x \in Y$ とは，Y において x が1回以上出現することとし，このとき x は Y の要素であるという．有限マルチセットのサイズ $|Y|$ は Y のすべての要素の出現回数の和とする．$Y_1, Y_2 \in Multi(X)$ に対して $Y_1 \sqcup Y_2$ を Y_1 と Y_2 の和とする．すなわち，$Y_1 \sqcup Y_2$ における $x \in X$ の出現回数は Y_1 における x の出現回数と Y_2 における x の出現回数の和とする．これはマルチセットの有限族 $\{Y_j\}_{j \in J}$ の和 $\bigsqcup_{j \in J} Y_j$ に拡張される（インデックス J も有限マルチセットであってもよい）．

$i \in A_0$ とするとき，深さ k の i についての**重子木**全体の集合 $\mathcal{MT}_{k,i}(W)$ を k について帰納的に次のように定義する：$\mathcal{MT}_{0,i}(W) = W$; $\mathcal{MT}_{k+1,i}(W) = \{\langle w, C_1, \ldots, C_n\rangle \mid w \in W, C_j \in Multi(\mathcal{MT}_{k,j}(W)), (i \neq 0 \text{ ならば } C_i = \emptyset)\}$．特に $\mathcal{MT}_{k,0}(W)$ を $\mathcal{MT}_k(W)$ と書く．ただし $\emptyset$ は空集合（マルチセットと見なせる）である．木の場合と同様の方法で，重子木に対しても根，i-子，重子木の大きさが定義される．

今，マルチセット Y において各要素がたかだか1回しか出現しないとき，Y を**単出形**であるということにする．この単出形という概念を重子木に拡張する．重子木の深さ k について帰納的に定義する：$w \in \mathcal{MT}_{0,i}(W)$ は単出形である；$t = \langle w, C_1, \ldots, C_n\rangle \in \mathcal{MT}_{k+1,i}(W)$ について，各 C_j が単出形であり，C_j の各要素も単出形であるとき，t は単出形である．単出形の重子木を同じ深さの木と同一視できることは自明である．

次に，重子木による命題のモデル検査を定義する．定義6をそのままなぞることにより，重子木 t，重子木のマルチセット C，エージェント $i \in A_0$，知識命題 φ の間に関係 $t, C \models_i \varphi$ が定義できる．木は重子木の特殊な場合であるから，この定義は定義6の拡張である．

ここで，後に単調知識命題についてのモデル検査の性質を証明するために必要となる補題を1つ用意しておく．まず，$C \in Multi(\mathcal{MT}_{k,i}(W))$ であるとき，「すべての $t \in C$ について $t, C \models_i \varphi$ であること」を $C \models_i \varphi$ と書くことにする．このとき次が成立する．

補題9 次の4つの性質が成立する．

(1) φ が平命題のとき $C \models_i \varphi$ iff すべての $t \in C$ について $root(t) \models \varphi$;
(2) $C \models_i \varphi \wedge \phi$ iff $C \models_i \varphi$ かつ $C \models_i \phi$;
(3) $C \models_i K_i\varphi$ iff $C \models_i \varphi$;
(4) $C \models_i K_j\varphi$（ただし $i \neq j$）iff すべての $t' \in C$ について，$Chld_j(t') \models_j \varphi$.

［証明］ 定義から自明である． □

ここで注意しなければならないのは，$C \models_i \neg\varphi$ iff $C \not\models_i \varphi$ は一般には成立しないことである．このことが，最終定理25が一般の知識命題ではなく単調知識命題に限定されたものになる原因である．

3.2 重子木の木への変換

次に，重子木 t を，単出形の重子木，すなわち木に変換する操作 $pack(t)$ を定義する．まず，重子木のマルチセット C に対して，根を共通とする C の重子木をまとめて1つにする操作 $Merge(C)$ を定義して，これをもとに *pack* を定義する☆．

定義10 深さが k の重子木のマルチセット $C \in Multi(\mathcal{MT}_{k,i}(W))$ について，$Merge(C)$ を次のように定義する．まず，C に対して $Rootset(C) \subset W$ を次のように定義する：$Rootset(C) = \{root(t) \in W \mid t \in C\}$ とする．次に C を $Rootset(C)$ の要素によって次のように分割する．$w \in Rootset(C)$ に対してマルチセット $[w]$ を定義する：C の要素 t のうち $root(t) = w$ であるものが，C における出現回数と同じだけ出現するマルチセットを $[w]$ とする．各 $[w]$ に対して $mc([w]) \in \mathcal{MT}_{k,i}(W)$ を次のように定める：$root(mc([w])) = w$; $Chld_j(mc([w])) = \bigsqcup_{t' \in [w]} Chld_j(t')$．このとき，$Merge(C)$ は

☆ なお混乱を避けるため，重子木に対する操作に対しては小文字で始まる名前をつけ，その集合やマルチセットに対する操作には大文字で始まる名前をつけることを原則とした．

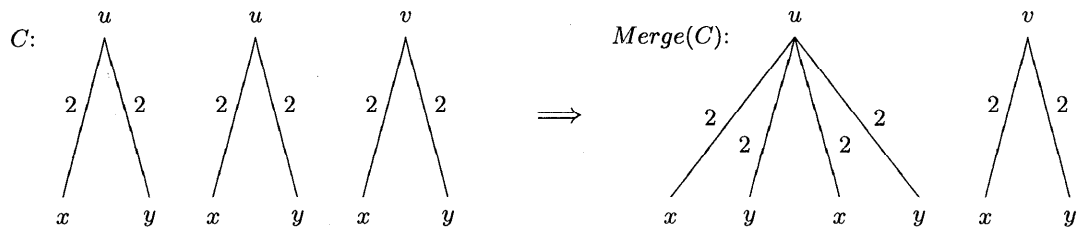


図6 *Merge* の例
Fig. 6 Example of *Merge*.

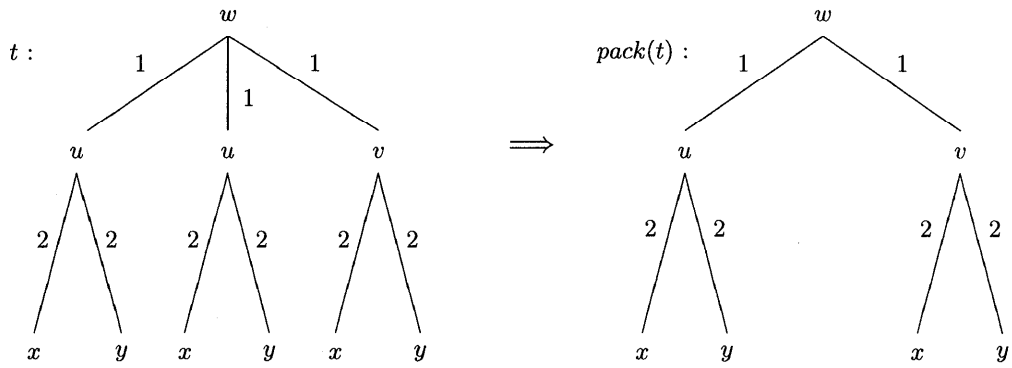


図7 *pack* の例
Fig. 7 Example of *pack*.

$$Merge(C) = \bigsqcup_{w \in Rootset(C)} \{mc([w])\}$$

とする．

例11 C として**図6**に示す3個の深さ1の木からなるマルチセットを考える．このとき，$[u]$ は $\langle u, \emptyset, \{x, y\}\rangle$ という木2個から成るマルチセットである．また，$mc([u])$ は $\langle u, \emptyset, \{x, x, y, y\}\rangle$ という重子木になる．したがって，$Merge(C)$ は図6に示すかたちになる．

定義から *Merge* について次のことがいえる．

(1) $C \in Multi(\mathcal{MT}_{k,i}(W))$ のとき $Merge(C) \in Multi(\mathcal{MT}_{k,i}(W))$ である（これは $k = 0$ のときにも成立することに注意する）．

(2) 任意の C に対して $Merge(C)$ はマルチセットとして単出形である．これから $Merge(C)$ の定義式は

$$Merge(C) = \bigcup_{w \in Rootset(C)} \{mc([w])\}$$

と書き直すことができる．

(3) $Merge(C)$ は C より次の意味で簡約化されている：

$$forest\_size(Merge(C)) \leq forest\_size(C).$$

定義された *Merge* を帰納的に使って *pack* を定義する．

定義12 $t \in \mathcal{MT}_{k,i}(W)$ について，$pack(t) \in \mathcal{MT}_{k,i}(W)$ を k について帰納的に定義する：

- $k = 0$ のとき，$pack(t) = t$ とする．
- $k = k' + 1$ のとき，$t = \langle x, C_1, \ldots, C_n\rangle$ とおける．このとき $pack(t) = \langle x, C'_1, \ldots, C'_n\rangle$，ただし $C'_i = [pack(t') \mid t' \in Merge(C_i)]$ とする（ここで $[pack(t') \mid t' \in Merge(C_i)]$ とは，各 $t' \in Merge(C_i)$ の各出現に対して $pack(t')$ の D における1つの出現が1対1で対応付けられるマルチセット D のこととする．同様の記法を以下でも使用する）．

例として**図7**に示す重子木 t に対する *pack* を考える．t を $\langle u, C_1, \emptyset\rangle$ と書くと，C_1 は例11で使った C にほかならないから，$Merge(C_1)$ は図6に示したようになる．今，$Merge(C_1)$ の要素である2個の重子木のうち，根が u である方を t_1，根が v である方を t_2 とする．すると，$pack(t_1) = \langle u, \emptyset, \{x, y\}\rangle$ であり，$pack(t_2) = t_2$ なので，$pack(t)$ は図7に示したようになる．

上の定義と，先の *Merge* の性質から，*pack* について次のことがいえる．

(1) 定義12の中での $C'_i = [pack(t') \mid t' \in Merge(C_i)]$ は単出形である．したがって，$C'_i = \{pack(t') \mid t' \in Merge(C_i)\}$ とおきかえて

よい.

(2) 上記(1)から帰納的に，各 $t \in \mathcal{MT}_{k,i}(W)$ に対して $pack(t)$ が単出形であることがいえる.

(3) したがって，$pack$ は重子木から木を作る操作であると見なすことができる.

3.3 単調知識命題のモデル検査の $pack$ による不変性

単調知識命題を考える場合は，t のもとでのモデル検査と $pact(t)$ のもとでのモデル検査の意味が変わらないことを，パスという新たな道具を使って示す．直観的には，パスは t の根 w から t の途中の節点（である重子木の根）までの経路を示すものである．より正確にするために，次のように定義する.

定義 13 長さ l の**パス**とは，$w_0 i_1 w_1 i_2 w_2 \cdots i_l w_l$ という形の列である．ただし各 $w_j \in W$，$i_j \in A$ で，各 j について $i_j \neq i_{j+1}$ とする．$Path$ をパス全体の集合とする．パス p_1 と p_2 の i による結合を $p_1 \cdot ip_2$ と書く（$i \in A$）.

パスの集合 P が**プレフィクスについて閉じている**とは，$p \cdot iw \in P$ ならば $p \in P$ であることとする．$\mathcal{P}athset$ をプレフィクスについて閉じているパスの有限集合全体の集合とする．$P \in \mathcal{P}athset$ のとき，P は有限集合であるから，P が空集合でないかぎり P の要素の中に最長の長さを持つものが存在する．このとき P の中の最長のパスの長さを P の**深さ**と呼ぶ．また P が空集合のときは，P の深さを 0 と定義する．$\mathcal{P}athset$ の要素で深さが k 以下であるもの全体の集合を $\mathcal{P}athset_k$ と書く.

$P \in \mathcal{P}athset_k$ と $w \in W$，$1 \leq i \leq n$ に対して，$P|_{w,i}$ を次のように k について帰納的に定義する：$k = 0$ のとき $P|_{w,i}$ は空集合；$k = k' + 1$ のとき $P|_{w,i} = \{p \mid w \cdot ip \in P\}$．定義から $P \in \mathcal{P}athset_{k+1}$ に対して $P|_{w,i} \in \mathcal{P}athset_k$ となる.

重子木のマルチセット $C \in Multi(\mathcal{MT}_{k,i}(W))$ に対する $Paths(C) \in \mathcal{P}athset_k$ と，$t \in \mathcal{MT}_{k,i}(W)$ に対する $paths(t) \in \mathcal{P}athset_k$ を同時に k について帰納的に定義する：$k = 0$ のとき，$paths(w) = \{w\}$，$Paths(C) = \bigcup_{w \in C} paths(w)$．$k \geq 1$ のとき $paths(t) = \{root(t)\} \cup \{root(t) \cdot ip \mid 1 \leq i \leq n, p \in Paths(Chld_i(t))\}$，$Paths(C) = \bigcup_{t \in C} paths(t)$.

次に $\mathcal{P}athset_k$ の要素についても関係 $\models$ を定義する．$P \in \mathcal{P}athset_k$，$i \in A_0$ と，深さ k 以下の単調知識命題 ψ に対して $P \models_i \psi$ を次のように ψ の構造について帰納的に定義する：

- ψ が平命題のとき $P \models_i \psi$ iff すべての長さ 1 のパス $w \in P$ について $w \models \psi$;
- $P \models_i \psi_1 \wedge \psi_2$ iff $P \models_i \psi_1$ かつ $P \models_i \psi_2$;
- $P \models_i K_i \psi$ iff $P \models_i \psi$.
- $P \models_i K_j \psi$（ただし $i \neq j$）iff すべての長さ 1 のパス $w \in P$ について，$(P|_{w,j}) \models_j \psi$.

特に $P \models_0 \psi$ を $P \models \psi$ と書く.

定義から次の 2 点がいえる（P，Q がプレフィクスについて閉じている点に注意する）：

(a) $P \subset Q$ のとき $Q \models_i \psi$ ならば $P \models_i \psi$;

(b) $P \models_i \psi$ かつ $Q \models_i \psi$ ならば $P \cup Q \models_i \psi$.

これから，次が成立する.

命題 14 $t \in \mathcal{MT}_k(W)$ で，ψ が単調知識命題，$KD(\psi) \leq k$ とするとき $t \models \psi$ iff $paths(t) \models \psi$.

［証明］ より一般的な主張：「$i \in A_0$，$C \subset \mathcal{MT}_{k,i}(W)$，$KD(\psi) \leq k$ のとき $C \models_i \psi$ iff $Paths(C) \models_i \psi$」(*) を示す．命題 14 は $C = \{t\}$，$i = 0$ の場合である．上の主張 (*) を補題 9 を利用して ψ の構造についての帰納法で示す.

ψ が平命題のときは自明である．また $\psi = \psi_1 \wedge \psi_2$ のときと $\psi = K_i \psi'$ のときは，帰納法の仮定からすぐにいえる.

$\psi = K_j \psi'$（$j \neq i$）のときを考える．定義から $Paths(C)|_{w,j} = \bigcup \{Paths(Chld_j(t')) \mid t' \in C,\ root(t') = w\}$ である．これから上の (a)，(b) により $Paths(C) \models_i K_j \psi'$ iff $\bigcup_{t' \in C} Paths(Chld_j(t')) \models_j \psi'$ となる．また帰納法の仮定から，各 $t' \in C$ について，$Chld_j(t') \models_j \psi'$ iff $Paths(Chld_j(t')) \models_j \psi'$ である．したがって，もう一度上の (a)，(b) を使って $Paths(C) \models_i K_j \psi'$ iff $C \models_i K_j \psi'$ となる． □

今，$t, t' \in \mathcal{MT}_k(W)$ に対して $paths(t) = paths(t')$ であるとき $t \equiv t'$ と書くと，上の命題から次が成立する.

系 15 $t, t' \in \mathcal{MT}_k(W)$，$\psi$ が単調知識命題で $KD(\psi) \leq k$ のとき $t \equiv t'$ ならば $t \models \psi$ iff $t' \models \psi$.

また，付録に証明を示したように次の命題が成立する.

命題 16 $t \in \mathcal{MT}_{k,i}(W)$ について $t \equiv pack(t)$.

系 15 と命題 16 から次が成立する.

系 17 $t \in \mathcal{MT}_k(W)$，ψ が単調知識命題で $KD(\psi) \leq k$ のとき $t \models \psi$ iff $pack(t) \models \psi$.

系 17 から，単調知識命題の木 t のもとでのモデル検査は，$pack(t)$ の上で行えることが分かる．$pack(t)$ の大きさは次のように評価できる.

命題 18 任意の木 $t \in \mathcal{MT}_{k,i}(W)$ に対して，

$$size(pack(t)) \leq (n|W| + 1)^k$$

である.

［証明］ $pack(t)$ の i-子がたかだか $|W|$ 個しかないこ

とから，k についての帰納法で証明できる． □

3.4 *pack* を利用した木の更新

2.4 節で述べたように，kt を使って $\mathrm{kt}_k(r) \models \psi$ を決定することを考えた場合，$\mathrm{kt}_k(r)$ の大きさは最悪の場合，少なくとも $|W|$ に関して指数オーダで増大する．以下では $\mathrm{kt}_k(r)$ のかわりに *pack* を利用した木を考えることにより，$|W|^k$ のオーダで同じ単調知識命題のモデル検査ができることを示す．

まず，update を重子木に対応させた mupdate を定義する：

$$\mathbf{mupdate}(w', w) = w;$$
$$\mathbf{mupdate}(\langle w', C'_1, \ldots, C'_n\rangle, w) = \langle w, C_1, \ldots, C_n\rangle,$$

ただしここで

$$C_i = [\mathbf{mupdate}(t', w'') \mid t \in C'_i,\ w'' \sim_i w,\ \langle root(t'), w''\rangle \in T]$$

である．この意味は，条件を満たす t'，w'' の各組に対して $\mathbf{mupdate}(t', w'')$ を，t' の C'_i における出現回数だけ出現させてできるマルチセットである．

定義から，任意の $t \in \mathcal{T}_{k,i}(W)$ に対して t は重子木でもあるが，$\mathbf{mupdate}(t, w) \equiv \mathbf{update}(t, w)$ となることは明らかである．

さらに，$t \equiv t'$ ならば $\mathbf{mupdate}(t, w) \equiv \mathbf{mupdate}(t', w)$ になることを示すが，帰納法を使うためには，重子木のマルチセットに対する命題として証明する方がやさしい．そこで $C, D \in Multi(\mathcal{MT}_{k,i}(W))$ に対して $C \equiv D$ iff $Paths(C) = Paths(D)$ と定義して，次の命題を証明する．

命題 19 $C, D \in Multi(\mathcal{MT}_{k,i}(W))$ かつ $C \equiv D$ であるとする．また $w_1, w_2 \in W$ についての述語 $P(w_1, w_2)$ があるとする．このとき

$$[\mathbf{mupdate}(t, w) \mid t \in C, P(root(t), w)] \equiv [\mathbf{mupdate}(t, w) \mid t \in D, P(root(t), w)].$$

［証明］ k についての帰納法で示す．$k = 0$ のとき，C と D はマルチセットとして要素は等しく，出現数が違うだけである．したがって，$P(root(t), w)$ を満たす t，w の組の集合は両辺で等しくなり，この式は成立する．

$k \geq 1$ のときを考える．$F(t, w, j, k)$ を

$$F(t, w, j, k) = [\mathbf{mupdate}(t', w') \mid t' \in Chld_j(t),\ w' \sim_j w, \langle root(t'), w'\rangle \in T]$$

とおく．すると，証明すべき式の左辺 $Left$ は mupdate の定義から

$$Left = [\langle w, F(t, w, 1, k-1), \ldots, F(t, w, n, k-1)\rangle \mid t \in C,\ P(root(t), w)]$$

となる．各 w ごとの j-子のマルチセットを $Left(w, j)$ とすると $Left(w, j) = [F(t, w, j, k-1) \mid t \in C, P(root(t), w)]$ であり，

$$Left \equiv [\langle w, Left(w, 1), \ldots, Left(w, n)\rangle \mid w \in W]$$

である．同様に右辺 $Right$ に対する $Right(w, j)$ を定義する．$C \equiv D$ より $\{root(t) \mid t \in C\}$ と $\{root(t) \mid t \in D\}$ は一致するので，$Left \equiv Right$ を示すには，各 j について $Left(w, j) \equiv Right(w, j)$ を示せばよい．

ここで，$Left(w, j)$ を F の定義を使って書き換えると

$$Left(w, j) = [\mathbf{mupdate}(t', w') \mid t' \in UChld_j(C, w, P),\ w' \sim_j w, \langle root(t'), w'\rangle \in T]$$

となる．$Right(w, j)$ も同様である．ただし

$$UChild_j(C, w, P) = [t' \mid t \in C, P(root(t), w), t' \in Chld_j(t)]$$

とする．この定義から $Paths(UChild_j(C, w, P)) = \bigcup_{P(v,w)} Paths(C)|_{v,j}$ となる．今，$C \equiv D$ であるから

$$Paths(UChild_j(C, w, P)) = Paths(UChild_j(D, w, P))$$

となる．よって帰納法の仮定より，$Left(w, j) \equiv Right(w, j)$ となる（ただし $w_2 \sim_j w$ かつ $\langle w_1, w_2\rangle \in T$ となることを新たな述語 $P(w_1, w_2)$ とした）． □

系 20 $t \equiv t'$ ならば

$$\mathbf{mupdate}(t, w) \equiv \mathbf{mupdate}(t', w)$$

である．

［証明］ 命題 19 で $C = \{t\}$，$D = \{t'\}$ ととり，述語 $P(w_1, w_2)$ を $w_2 = w$ とすればよい． □

さて $\mathrm{kt}'_k(r)$ を $\mathrm{kt}'_k(w_1) = pack(\mathrm{kt}_k(w_1))$ から始めて順次

$$\mathrm{kt}'_k(r'w) = pack(\mathbf{mupdate}(\mathrm{kt}'_k(r'), w))$$

と定義する．このとき次のことが示せる．

命題 21 $\mathrm{kt}_k(r) \equiv \mathrm{kt}'_k(r)$

［証明］ r の長さについての帰納法で証明する．$|r| = 1$ のときは明らかである．$|r| > 1$ のとき，命題 16，系 20，および帰納法の仮定を使うと $r = r'w$ のとき

$$\mathrm{kt}_k(r'w) = \mathbf{update}(\mathrm{kt}_k(r'), w) \equiv \mathbf{mupdate}(\mathrm{kt}_k(r'), w) \equiv \mathbf{mupdate}(\mathrm{kt}'_k(r'), w) \equiv pack(\mathbf{mupdate}(\mathrm{kt}'_k(r'), w)) = \mathrm{kt}'_k(r'w)$$

となる（**図 8** 参照）． □

以上で，$pack(t)$ の形の木によってモデル検査ができることが分かった．

例 22 図 5 に対応する重子木の更新を**図 9** に示す．根から 1–3 のパスでたどることができる枝以外は省略

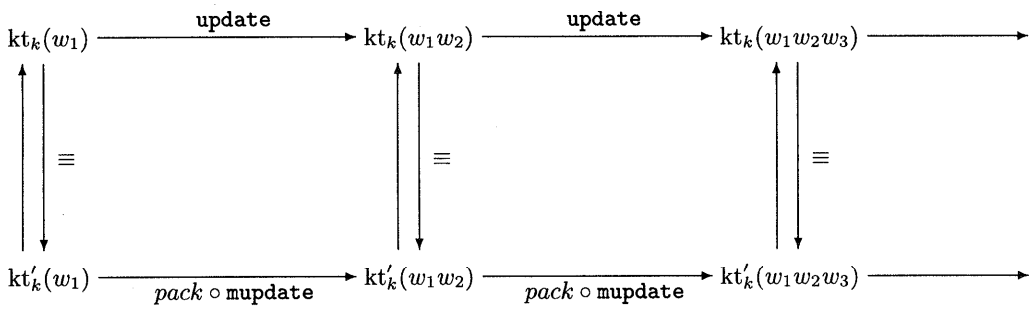


図 8 $\mathrm{kt}_k(r)$ と $\mathrm{kt}'_k(r)$ の関係
Fig. 8 Relation between $\mathrm{kt}_k(r)$ and $\mathrm{kt}'_k(r)$.

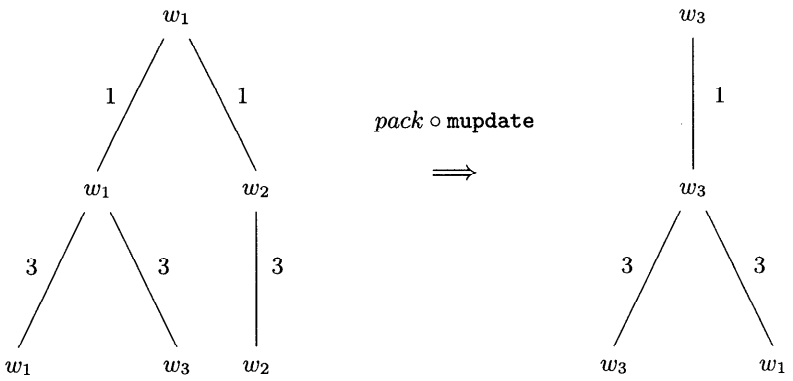


図 9 $pack \circ \mathtt{mupdate}$ による更新の例
Fig. 9 Example of $pack \circ \mathtt{mupdate}$.

してある．図 5 と比較すると，更新後の木が単純化されていることが分かる．もちろん，この木によるモデル検査がすべての命題について図 5 のものと一致するわけではない．たとえばこの木は $K_1\neg K_3 q$ を満たす．単調知識命題に限定すれば図 5 の更新後の木とこの木は同一のモデル検査結果を返す．

3.5 単調知識命題のモデル検査の計算量

$pack \circ \mathtt{mupdate}$ の 1 ステップにかかる時間計算量を調べ，これをもとに，モデル検査全体の時間計算量を求める．以下の議論では W の 2 つの要素 w_1，w_2 が，$w_1 = w_2$ であるか否かを定数時間で決定できると仮定する．

命題 23 $t \in \mathcal{T}_{k,i}(W)$ に対して $pack(t)$ を求めることは $O(size(t)|W|)$ の時間でできる．

［証明］ k についての帰納法で証明する．$k = 0$ のとき，$t = pack(t)$ であるから，定数時間である．

$k \geq 1$ の場合を示す．一般に $C \in Multi(\mathcal{MT}_k(W))$ に対して $Mappack(C) = \{pack(t) \mid t \in C\}$ とおく．$t = \langle w, C_1, \ldots, C_n\rangle$ とすると，

$$pack(t) = \langle w, C'_1, \ldots, C'_n\rangle,$$

ただし $C'_j = Mappack(Merge(C_j))$ である．ここで $Merge(C_j)$ が $O(forest\_size(C_j)|W|)$ でできることを示す．まず定義 10 に従い C_j の要素を根ごとに分ける．これは $|C_j||W|$ でできる．次に各根 w ごとに，w を根とする子木をまとめたマルチセット $[w]$ を作る．この過程には全体としての子木の数だけかかる．子木の数 $< forest\_size(C_j)$ であるから $Merge(C_j)$ は $O(forest\_size(C_j)|W|)$ でできる．次に，$Mappack(C_j)$ の計算が $O(forest\_size(C_j)|W|)$ でできることを示す．$Merge(C_j)$ の各要素 t' に対する $pack$ は帰納法の仮定から $O(size(t')|W|)$ でできる．したがって，$Mappack(C_j)$ 全体では $O(forest\_size(Merge(C_j))|W|)$ である．これは，$|Merge(C_j)| \leq |C_j|$ なので，$O(forest\_size(C_j)|W|)$ より小さい．このことから C'_j を求めるのは

$$O(forest\_size(C_j)|W|)$$

でできる．すべての j についてこれを行うから $pack(t)$ 全体では $O(size(t)|W|)$ である． □

命題 24 $t \in \mathcal{MT}_{k,i}(W)$ に対して

$$size(\mathtt{mupdate}(t, w)) \leq size(t)|W|^k$$

であり，この計算にかかる時間も $O(size(t)|W|^k)$ である．

［証明］ k についての帰納法で示す．$k = 0$ のときは自明である．また，$k = k' + 1$ のとき $\mathbf{mupdate}(\langle w_0, C_1, \ldots, C_n\rangle, w) = \langle w, C'_1, \ldots, C'_n\rangle$，ただしここで

$$C'_j = [\mathbf{mupdate}(t', w') \mid t' \in C_j, w' \sim_j w, \langle root(t'), w'\rangle \in T]$$

である．帰納法の仮定から $\mathbf{mupdate}(t', w')$ の大きさは $size(t')|W|^{k'}$ 以下，かつ計算時間は $O(size(t')|W|^{k'})$ である．また t' と w' の選び方は $|C_i| \times |W|$ である．これから求める結果が得られる． □

以上をまとめると，次のことがいえる．

定理 25 単調知識命題 ψ に対して $M, r \models \psi$ を決定するのに必要な時間計算量は，$KD(\psi) = k$ のとき $O(n^k(|W|^{2k+1}|r| + |W|^k|\psi|))$ である．

［証明］ $r = w_1 \cdots w_m$ であるとする．$\mathrm{kt}'_k(w_1)$ の生成がこの木の大きさのオーダの時間でできることはすぐに分かる．これは $O(n^k|W|^k)$ である．次に

$$\mathrm{kt}'_k(r'w_j) = pack(\mathbf{mupdate}(\mathrm{kt}'_k(r'), w_j))$$

を使う各段階を考える．このときは，まず **mupdate** を行い，次に *pack* を行う．**mupdate** の入力（$\mathrm{kt}'_k(r')$）の大きさは命題 18 から $n^k|W|^k$ のオーダであるから，前命題より **mupdate** には $O(n^k|W|^{2k})$ の時間がかかる．またこれから *pack* の入力の木の大きさは $n^k|W|^{2k}$ のオーダである．したがって，命題 23 により *pack* には $O(n^k|W|^{2k+1})$ かかる．以上から，$\mathrm{kt}'_k(r')$ から $\mathrm{kt}'_k(r'w)$ を作る 1 ステップにはそれぞれ $O(n^k|W|^{2k+1})$ かかる．

最終的に完成した $\mathrm{kt}'_k(r)$ を使って $\mathrm{kt}'_k(r) \models \psi$ を決定するのにかかる時間が $O(size(\mathrm{kt}'_k(r))|\psi|)$ であることはすぐに分かるので，この定理の結果となる． □

4. おわりに

本論文では，複数のエージェントが存在する動的環境を完全な記憶を持った同期システムとモデル化した場合に，一連の事象が起きた後で，各エージェントがどのような単調知識命題を正しいと考えているかを効率的に計算する方法を示した．具体的には，問題を様相論理のモデル検査の問題に変換し，この問題が状態系列の長さに関して線形時間かつシステムの状態数に関して多項式時間で解けることを示した．

扱える論理式を単調知識命題に制限したので，本論文の適用範囲は限られたものになる．たとえば，エージェント 1 と 2 が競合関係にあるとしよう．このとき，エージェント 2 が p を知らないことをエージェント 1 が知っている場合のみ，エージェント 1 は戦略 1 を採用するとしよう．もし一般の知識命題が扱えるプログラミング言語であれば，このことは

if $K_1\neg K_2 p$ **then** take Strategy 1

と書くことができる．ここで $K_1\neg K_2 p$ は単調知識命題ではないので，本論文で提案した方法では扱えない．

しかし，本論文の手法でも単調知識命題の否定の場合は「知らない」という条件が扱える．たとえば，エージェント 2 が p を知っていることをエージェント 1 が知らない場合のみ，エージェント 1 が p をエージェント 2 に送る，という条件は

if $\neg K_1 K_2 p$ **then** send p from 1 to 2

と表せ，$M, r \not\models K_1K_2p$ と $M, r \models \neg K_1K_2p$ は同値なので，本論文の手法が使える．

なお，本論文では環境 E と状態系列 r が与えられたときに，E から構成される Kripke 構造 $M(E)$ の状態 r の下でのモデル検査について議論し，このモデル検査を利用して上のような知識命題が扱えるプログラミング言語で書かれたプログラムを各エージェントがどのように実行するか，という問題には直接触れなかった．この問題については，応用問題に関する議論を含めて別途稿を改めて議論する予定でいる．

ここでは上の場合に限った直観的な説明を補足する．各エージェント i は実際の状態系列 r を特定することはできないが，自分の観測結果をもとにして各時点での実際の状態（w とする）の候補の集合 $\{w' \mid w' \sim_i w\}$ が認識できる．このことから，エージェント i は初期状態 w_1 で $\mathrm{kt}'_k(w_1)$ の i-子の集合 $(Chld_i(\mathrm{kt}'_k(w_1)))$ を構成することができる．そして，本論文で $\mathrm{kt}'_k(r)$ を求めたように，各時点ごとにエージェント i の観測結果から得られる，実際の状態の候補の情報を使って，$Chld_i(\mathrm{kt}'_k(w_1))$ を **mupdate** と *pack* で書き換えていくことにより，各時点でエージェント i は $Chld_i(\mathrm{kt}'_k(r))$ の情報を管理でき，それを使うことにより $M, r \models \neg K_1K_2p$ などの真偽値を判定できる．

一方，磯崎[2]は知識ではないが信念の場合に単調信念命題を使って，話者が聴者に関する自分の信念をもとにして，「よ」「よね」などの終助詞を使い分ける方法を提案している．ただし，そこで使われる論理モデルは磯崎・勝野[4]で考察されている非標準構造の特別な場合に基づいているので，本論文の結果と比較する場合は注意が必要である．

本論文で扱ったように，様相論理の $S5_n$ の体系で定式化される知識を使う場合は，稀にしか起こらない状

態遷移もすべて考慮しなければならないので，ある時点で考えなければならない状態数は一般に増大する．その結果，エージェントが持つ確定的な知識，すなわちその時点で考えなければならないすべての状態で成り立つ知識は限られたものになる可能性がある．たとえば，状態 w_1 から p が成立する状態 w_2 に遷移する確率が 99.9%であり，状態 w_1 から p が成立しない状態 w_3 に遷移する確率は 0.01%であれば，w_3 への遷移を無視して，p が成り立つと信じて処理を進め，矛盾が生じた時点で，しかるべき矛盾の解消を行うことも考えられる．そこで，$KD45_n$ などの様相論理の体系で定式化される信念を用いて通常起こりうる場合のみを検討の対象にすることも考えられる．この場合も，基本的には本論文と同様に議論を展開することができるが，$S5_n$ の場合のように，与えられた状態間の $KD45_n$ 的関係を自然な形で状態系列間の $KD45_n$ 的関係に拡張できないので，いくつかの点で変更が必要である．信念の場合の取扱いは稿を改めて議論する予定である．

なお，本論文の枠組みと似た動的環境の下での信念を扱う研究では，磯崎・勝野[3),4)]がエージェントの信念を推測するアルゴリズムを提案し，その論理的意味を考察している．そこでは，エージェントが現在観測している情報から得られる信念を重要視し，またアルゴリズムの効率を優先している．したがって，観測された事象の事前条件を考慮していない．このことは，本論文の枠組みに翻訳すると，どの状態からも任意の状態に遷移可能な場合を考えていることになる．その結果，起こりえない状態遷移が関係する場合には，磯崎・勝野[3),4)]の方法ではエージェントが我々の直観に合わない信念の推定をすることがある．

本論文で扱ったような問題の解決を積み重ねて，知識指向プログラミング[5)]やエージェント指向プログラミング[6)]のパラダイムを実現しその有効性を検証することは，今後に残された大きな課題である．

謝辞 本研究に関連して貴重なコメントをいただいた，真鍋義文主任研究員，磯崎秀樹主任研究員に感謝いたします．また，的確なコメントを下さった査読者の方々に感謝いたします．最後に，日頃ご指導いただく石井健一郎企画担当部長に深謝いたします．

参考文献

1) Fagin, R., Halpern, J.Y., Moses, Y. and Vardi, M.Y.: *Reasoning about knowledge*, MIT Press (1995).
2) 磯崎秀樹：信念推定算法の終助詞選択への適用，電子情報通信学会，Vol.NLC 96-6 (1996).
3) Isozaki, H. and Katsuno, H.: A Semantic Characterization of an Algorithm for Estimating Others' Beliefs from Observation, *Proc. National Conference on Artificial Intelligence*, pp.543–549, MIT Press (1996).
4) 磯崎秀樹，勝野裕文：マルチエージェント環境における遡行的信念推定アルゴリズム，情報処理学会論文誌，Vol.38, No.3, pp.429–442 (1997).
5) Moses, Y. and Kislev, O.: Knowledge-oriented Programming (Exteneded Abstract), *Proc. ACM Symposium on Principles of Distributed Computing*, pp.261–270 (1993).
6) Shoham, Y.: Agent-oriented Programming, *Artificial Intelligence*, Vol.60, No.1, pp.51–92 (1993).
7) van der Meyden, R.: Common Knowledge and Update in Finite Environments. I (Extended Abstract), *Proc. 5th Conference on Theoretical Aspects of Reasoning About Knowledge*, pp.225–242, Morgan Kaufmann (1994).
8) van der Meyden, R.: Common Knowledge and Update in Finite Environments., *Information and Computation*, Vol.140, No.2, pp.115–157 (1998).

付　録

A.1 *pack* の性質（命題 16）の証明

$pack(t) \equiv t$ を示す．

命題 26 $C \in Multi(\mathcal{T}_{k,i}(W))$ について
$Paths(C) = Paths(Merge(C))$.

［証明］ $p \in Paths(C)$ iff $p \in Paths(Merge(C))$ を p の長さ l についての場合分けによって証明する $(0 \leq l \leq k)$．

長さが 0 であるパスは C に含まれる重子木の根であり，これが $Merge(C)$ に含まれる重子木の根と一致することは $Merge$ の定義から自明である．

$k \geq 1$ で $1 \leq l \leq k$ のとき，$p = w \cdot jp'$ とおける．このとき (1)「$p \in Paths(C)$ である」ことと，(2)「$t \in C$ が存在して $w = root(t)$ かつ $p' \in Paths(Chld_j(t))$ となる」ことは同値である．一方，(3)「$p \in Paths(Merge(C))$ である」ことと (4)「ある $t' \in Merge(C)$ が存在して $w = root(t')$ かつ $p' \in Paths(Chld_j(t'))$ となる」ことも同値である．以下で (2) と (4) の同値性を示す．

(2) が成立するとき，$Merge$ の定義からある $t' \in Merge(C)$ が存在して $root(t') = root(t) = w$, $Chld_j(t') \sqsupset Chld_j(t)$ となる．ただしここで一般に，2 つのマルチセット A, B について $A \sqsupset B$ とは，

各 $x \in B$ について x の A における出現回数が B における出現回数と同じか，それを超えていることとする．$Paths$ について一般に $A \sqsupset B$ ならば $Paths(A) \supset Paths(B)$ が成立することは定義から明らかであるから (4) が成立する．

逆に (4) が成立するとする．$Paths$ が一般に $Paths(A \sqcup B) = Paths(A) \cup Paths(B)$ を満たすことは定義からすぐに分かるから，$Merge$ が $\bigsqcup$ によって定義されていることより，特定の $t'_c \in Chld_j(t')$ が存在して $p' \in Paths(\{t'_c\})$ となる．C の要素 t のうち $root(t) = root(t')$ となるもの全体の集合を C_w とおく．すると $Chld_j(t') = \bigsqcup_{t \in C_w} Chld_j(t)$ であるから，特定の $t \in C_w$ について $t'_c \in Chld_j(t)$ となる．このとき $p' \in Paths(Chld_j(t))$ である．$t \in C$, $root(t) = w$ であるから (2) が成立する． □

命題 23 の証明と同様に $C \in Multi(\mathcal{MT}_{k,i}(W))$ に対して $Mappack(C) = \{pack(t) \mid t \in C\}$ とおく．

命題 27 $C \in Multi(\mathcal{MT}_{k,i}(W))$ に対して $Paths(C) = Paths(Mappack(C))$.

[証明] k についての帰納法で示す．

$k = 0$ のとき $Mappack(C) = C$ であるから自明である．$k = k' + 1$ のとき，長さ 0 のパス $p = w$ については $p \in Paths(C)$ iff $p \in Paths(Mappack(C))$ は自明である．長さが 1 以上のパス $p = w \cdot jp'$ について考える．

$p \in Paths(C)$ とすると，ある $t \in C$ があって $w = root(t)$, $p' \in Paths(Chld_j(t))$ である．$t' = pack(t)$ を考えると $root(t') = root(t) = w$, $Chld_j(t') = Mappack(Merge(Chld_j(t)))$ である．ここで前命題と帰納法の仮定から

$$Paths(Chld_j(t')) = Paths(Chld_j(t))$$

である．したがって，$p' \in Paths(Chld_j(t'))$. これから $p \in paths(pack(t)) \subset Paths(Mappack(C))$ となる．逆に $p \in Paths(Mappack(C))$ とするとある $t' \in Mappack(C)$ が存在して $root(t') = w$ かつ $p \in Paths(Chld_j(t'))$ となる．$Mappack$ の定義からある $t \in C$ が存在して $t' = pack(t)$ となり，前命題と帰納法の仮定から $Paths(Chld_j(t')) = Paths(Chld_j(t))$ が成立する．したがって，$p' \in Paths(Chld_j(t))$ である．ゆえに $p \in paths(t) \subset Paths(C)$ となる． □

命題 28 (= 命題 16) $t \in \mathcal{MT}_{k,i}(W)$ について $t \equiv pack(t)$.

[証明] 命題 27 において $C = \{t\}$ とおく． □

A.2 木 $\mathbf{kt}_k(r)$ の大きさ

$size(\mathrm{kt}_2(r))$ が $|W|$ に関して指数オーダで増大する場合が実際に存在することを示す．2 人のエージェントから成る環境 $E = (W, T, V, \sim_1, \sim_2)$ を次のように定める．$W_1 = \{u_1, u_2, \ldots, u_{2k}\}$ かつ $W_2 = \{v_1, v_2, \ldots, v_{2k-1}\}$ とし，$W = W_1 \cup W_2$ とする．状態遷移は W_1 の内部での遷移は $\langle u_i, u_{i+1}\rangle \in T$ (ただし $1 \le i \le 2k-1$) と $\langle u_{2k}, u_1\rangle \in T$ のみであり，W_1 から W_2 への遷移は $\langle u_i, v_i\rangle \in T$ (ただし $1 \le i \le 2k-1$) と $\langle u_{2k}, v_{2k-1}\rangle \in T$ に限り，W_2 から W_1 への遷移は $\langle v_i, u_i\rangle \in T$ (ただし $1 \le i \le 2k-1$) であり，W_2 内での遷移は存在しないとする．同値関係 $\sim_1$ は 2 つの同値類 W_1 と W_2 からなるとする．同値関係 $\sim_2$ は W 全体が 1 つの同値類となるような関係，すなわちエージェント 2 は状態の区別がいっさいできないとする．

今，長さが $5k$ の状態系列の任意の 1 つを r_0 とし，$\mathrm{kt}_2(r_0)$ の大きさが少なくとも 2^k 以上になることを示す．$\mathrm{kt}_2(r_0)$ の根の 2-子の 1 つを t とすると，t は深さ 1 の木なので，$\langle w, C_1, \emptyset\rangle$ (ここで C_1 は W の部分集合) と書ける．$states(t) = \{w\} \cup C_1$ と定義するとき，t として $\mathrm{kt}_2(r_0)$ の根の 2-子すべてを考えた場合に $states(t)$ のとりうる値を考える．すなわち，

$$B(r_0) = \{states(t) \mid t \in Chld_2(\mathrm{kt}_2(r_0))\}$$

と定義する．このとき，後で示すように W_1 の k 個のもとから成る任意の集合は $B(r_0)$ に含まれることが示せる．W_1 の k 個のもとから成る集合は全部で ${}_{2k}C_k$ 個存在する．k に関する帰納法を使えば，容易に分かるように ${}_{2k}C_k \ge 2^k$ である．したがって，$\mathrm{kt}_2(r_0)$ の根は少なくとも 2^k 個の 2-子を持つので，$\mathrm{kt}_2(r_0)$ の大きさは少なくとも 2^k 以上になる．

以下では，W_1 の k 個のもとから成る任意の集合が $B(r_0)$ に含まれることを示す．まずエージェント 2 の立場になって事態を考えよう．エージェント 2 は状態の区別ができないので，r_0 が実際に起きたときに，長さ $5k$ の任意の状態系列 r_1 を実際に起きた候補として考える．すなわち，$r_0 \sim_2 r_1$ である．一方，エージェント 1 は，各遷移が W_1 と W_2 にまたがるものか，W_1 内のものかをエージェント 1 が区別できる．今，状態系列 $r = w_1 w_2 \cdots w_m$ に対して，$[r]$ は各 w_i が W_1 と W_2 のどちらに属しているかを示す 1, 2 の列とする．すなわち，$w_i \in W_j$ ならば $e_i = j$ とし，$[r] = e_1 e_2 \cdots e_m$ と定義する．すると，$r_1 \sim_1 r_2$ の必要十分条件は $[r_1] = [r_2]$ である．

長さ $5k$ の任意の状態系列 r_1 に対して，$\mathrm{kt}_2(r_0)$ の根の 2-子 $t = \langle w, C_1, \emptyset\rangle$ で，$last(r_1) = w$ となるものが存在する．このとき，

$$states(t) = \{last(r_2) \mid [r_2] = [r_1]\}$$

が成り立つ．上式の右辺を $ends(r_1)$ と書くと，

$B(r_0) = \{ends(r_1) \mid |r_1| = 5k\}$ となる．

今，$2k \geq i_k > i_{k-1} > \ldots > i_1 \geq 1$ として，$[r_1]$ が

$$(1211)^{2k-i_k}1(1211)^{i_k-i_{k-1}-1}1\cdots$$

$$\cdots(1211)^{i_2-i_1-1}1(1211)^{i_1-1}$$

となる場合を考える．ただし，$(1211)^m$ は 1211 を m 回繰り返す列とする（この列の長さが $5k$ であることを確認されたい）．このとき，次が成立する，

(1) 上のような r_1 は必ず存在する．

(2) $ends(r_1) = \{u_{i_1}, u_{i_2}, \ldots, u_{i_k}\}$.

上の (2) から W_1 の k 個の元からなる任意の集合が $B(r_0)$ に含まれることが分かる．

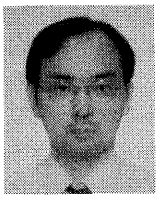
上の (1)，(2) の証明の概略を示す．今，状態系列 r' が $[r'] = (1211)^{2k-i_k}1$ という条件を満たす r_1 の最初の部分であるとして，$ends(r')$ がどのようになるかを調べる．このため，$ends$ の記法を少し拡張して，α が 1 または 2 の有限列であるとき，$ends(\alpha) = \{last(r) \mid [r] = \alpha\}$ と書くことにする．すると，$ends(1) = W_1$ になる．また，遷移の条件 T を考慮すると，$ends(1211) = \{u_2, \ldots, u_{2k}\}$ になる．よって，

$$ends((1211)^{2k-i_k}1) = \{u_1, u_{2k-i_k+2}, \ldots, u_{2k}\}$$

となる．

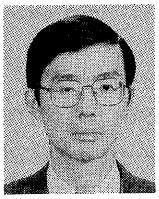
一般に，$(1211)^m$ は可能な最終状態の数を m 個減らす役目を果たし，また $(1211)^m$ の直後の 1 は直観的には最後に残る状態を1つ保存する役目を果たしている．これらに注意すれば，最終的に $ends(r_1) = \{u_{i_1}, u_{i_2}, \ldots, u_{i_k}\}$ を示すことができる．また，最終状態の候補が空集合ではないので，上のような r_1 が実際に存在することが分かる．

(平成 10 年 1 月 14 日受付)
(平成 11 年 9 月 2 日採録)



梅村 晃広（正会員）

1965 年生．1990 年東北大学大学院工学研究科情報工学専攻博士前期二年課程修了（修士相当）．同年日本電信電話（株）入社．ソフトウェア工学，理論計算機科学，AI 理論の研究に従事．1998 年 NTT データ通信（株）〔現（株）NTT データ〕に転籍．電子透かし，情報共有環境の研究開発に従事．ソフトウェア科学会会員．



勝野 裕文（正会員）

1952 年生．1976 年東京大学大学院理学系研究科数学専攻修士課程修了．同年電電公社入社．1988 年から 1989 年にかけてカナダトロント大学計算機科学科客員研究員．現在 NTT コミュニケーション科学基礎研究所人間情報研究部主幹研究員．データベース，人工知能等の研究に従事．電子情報通信学会，人工知能学会，ACM，AAAI 各会員．