ユーザモデルにおけるオートマトン型の判定

—モデル理論アプローチによるシミュレーション—

Evaluating Automaton Type of User Models

:Model Theory Approach for Simulation

旭　　貴朗

1. はじめに

本稿の目的は，シミュレーション実行環境（実行エンジン）の設計に向けて，モデル理論アプローチにおけるシミュレーションの立場から，ユーザが記述したモデルのオートマトン型を自動判定する方法を明らかにすることである．

モデル理論アプローチによるシミュレーション（Model Theory Approach for Simulation）では，複数の要素システムをオートマトンでモデル化し，それらを結合させてシミュレーションモデルを開発する（高原他 2007, 2016）（旭 2013）．これをユーザモデルと呼んでおり，ユーザモデルを言語 CAST で記述することが特徴である．言語 CAST は論理式を記述する言語であり，数理モデルとほとんど類似のモデル記述が可能である（旭他 2008）．もう一つの特徴は，ユーザモデルとそれを実行するソフトウェア（実行エンジンと呼んでいる）が分離していることである．特にシミュレーションのための実行エンジンは，標準的なもの（stdAutomatonEngine.p）があらかじめ開発実行環境 MTA-SDK.ova の中に装備されており、コンパイル時にユーザモデルと合成され，実行可能なシステムとなる（旭 2017）．そのため，ユーザはモデル作成に注力することができ，そのモデルを実行するためのエンジンの作成は不要である（旭 2017）．本稿では，ユーザが記述した（ソースコードに相当する）ユーザモデルを考察対象とする．

一方，次節で述べるように，オートマトンには二種類のモデル型（Mealy 型と Moore 型）があり，それぞれの種類のモデルを実行するアルゴリズムが異なる．つまり，型に合わせた異なる実行アルゴリズムを適用する必要がある．単純に考えると，使用すべき実行アルゴリズムをユーザがあらかじめ指定し，ユーザモデルを実行させればよい．そのためには，ユーザモデルの中にオートマトン型をあらかじめ「記述」しておく方法や，ユーザモデルを実行するときにオートマトン型を「選択」する方法などが考えられる．

しかしながら，プログラミング入門者や教育の現場では，ユーザモデルを一つ

の実行エンジンで実行できることが望ましい．なぜなら，オートマトン型の指定を忘れてモデルを作成した場合，あるいは間違えて指定して実行開始した場合でも，実行エンジンそのものが何らかの対処ができることが必要だからである．そのためには，ユーザモデルのオートマトン型をある程度自動判定できることが望ましい．

次の 2 節では，まず二種類のオートマトン型を数理的に定義する．また，複数の要素を結合したシステムの定義，および処理方法として逐次処理と並行処理の二種類を示す．3 節では，結合システムのモデルの「記述上の」特徴を明らかにする．4 節では，3 節の結果を元に，ユーザモデルのオートマトン型を自動判別するためのアルゴリズムを提案する．5 節はまとめである．

2. 諸定義

まず，要素システムの二種類のモデルとそれらを結合したシステムのモデルおよび処理方法の定義を行う．

(1) 要素システム

1）Mealy 型オートマトン（Mealy Type Automaton）

Mealy 型オートマトンは，現在の状態と現在の入力に応じてその時刻での出力が定まり，次の時刻における状態に遷移する入力出力システムのモデルである．以下に正確な定義を述べる．

任意の時刻 t (t = 1,2,3,…)でのシステム S への入力を a∈A，出力を b∈B，状態を c∈C とする．また，次の時刻 t+1 での状態を c’と書くことにする．与えられたシステム S に対して，二つの関数 δ, λ が存在し，これらの変数の間に，

$\delta(c, a) = c'$

$\lambda(c, a) = b$

という関係が，任意の時刻で常に成り立つとき，S は Mealy 型オートマトンである（S は Mealy 型オートマトンとしてモデル化できる）といい，A を入力集合，B を出力集合，C を状態集合，δ を状態遷移関数，λ を出力関数と呼ぶ．

このとき，5 項組 $<A, B, C, \delta, \lambda>$ を S の Mealy 型オートマトンモデルと呼び，$S = <A, B, C, \delta, \lambda>$と書く．本稿では，集合を略して $S = <\delta, \lambda>$ と書く．

Mealy 型オートマトンの動作は次のようになる．任意の時刻 t における状態が c であるとき，そこに a が入力されると，$b = \lambda(c, a)$ が出力され，次の時刻 t+1 における状態は $c' = \delta(c, a)$ となりシステムに記憶される．定義により初期出力はなく，何らかの入力によって状態遷移を行うときに何らかの出力が発生するのである．

2）Moore 型オートマトン（Moore Type Automaton）

一方，**Moore 型オートマトン**は，現在の状態と入力に応じて状態が遷移し，遷移後の状態に対して出力が決まるような入力出力システムのモデルである．正確に定義するならば，与えられたシステム S に対して，二つの関数 δ, λ が存在し，上記の変数の間に，

$\delta(c, a) = c'$

$\lambda(c) = b$

という関係が任意の時刻で常に成り立つとき，S は Moore 型オートマトンであると呼び，モデルは 1）と同じく，5 項組 $S = <A, B, C, \delta, \lambda>$で表現し，略して $S = <\delta, \lambda>$ と書く．

Moore 型オートマトンの動作は次のようになる．時刻 t における状態が c のとき，その時刻での出力は $\lambda(c)$ である．そこに a が入力されると，次の時刻 t+1 における状態は $c' = \delta(c, a)$ となりシステムに記憶されるとともに，$b' = \lambda(c')$ が出力される．定義により，初期状態 c0 における出力が存在し $\lambda(c0)$ である．その後の出力は必ず遷移後の状態に応じた出力がなされる．

3）状態機械（state machine）

上記二つのオートマトン型に拘らず，出力関数が状態 c をそのまま出力する（$\lambda(c, a) = c$ あるいは $\lambda(c) = c$）場合は，そのオートマトンを**状態機械**と呼ぶ．状態機械のモデルはもはや出力関数 λ を明示する必要はなく，状態遷移関数 $<\delta>$ だけを明示して理論展開することもできる．出力関数を明示しない場合は，Mealy 型でもあり Moore 型でもあるとして，本稿では両者に所属するものとして扱う．

図表 1　オートマトン型（筆者作成）

名称	状態遷移関数	出力関数
Mealy 型	$\delta(c,a) = c'$	$\lambda(c,a) = b$
Moore 型	$\delta(c,a) = c'$	$\lambda(c) = b$
（状態機械）	$\delta(c,a) = c'$	－

理論的には，二つのオートマトン型 1）2）は互いに等価にモデル変換できる（同等のモデリング能力を持つ）ことが知られている．しかし実践では，型の区別はユーザの裁量の範囲内にある．シミュレーション対象の特徴に応じて，モデル化しやすい方をユーザが選ぶのである．

4）問題の所在

図表 1 によると，単体システム（一つの要素からなるシステム）ならば，モデルの形式から，次のルールにより直ちに判定することが可能である．

R1）2 変数の出力関数 $\lambda(c, a) = b$ が定義されていれば，そのモデルは出力関数を持つ Mealy 型オートマトンである．

R2）1 変数の出力関数 $\lambda(c) = b$ が定義されていれば，そのモデルは出力関数を持つ Moore 型オートマトンである．

すなわち，単体システムに対しては，出力関数の引数（変数）の数によって型の判定が可能である．正確に言うならば，そのような型であるとして実行エンジンがユーザモデルを実行すれば良い．注意すべき点は，出力関数がたとえ恒等関数（$\lambda(c, a) = c$ あるいは $\lambda(c) = c$）であっても，それが存在する（ユーザモデル中

に記述されている）ならば，実行エンジンが出力関数を計算してしまうことである．しかし，その計算結果として状態と出力が等しくなるのであるから，シミュレーション上は何も問題はなく，オートマトン型に応じたアルゴリズムで実行すれば良いだけである．

【コンピュータ実装】

理論的には上記のように単純な判定条件で，モデルを数理的に操作することができようが，コンピュータ上で実行エンジンを設計する（作成する）立場から考えると，やや複雑なことになる．

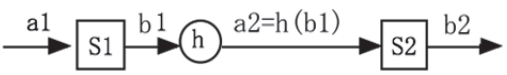
第 1 の問題は状態機械の扱いである．たとえ対象システムが状態 c をそのまま出力するシステムであっても，出力関数を $\lambda(c, a) = c$ あるいは $\lambda(c) = c$ と明示しているならば，実行エンジンはオートマトン型に合わせたは 2 つのアルゴリズムだけを作成すれば良い．しかしながら，ユーザ（モデル作成者）が状態機械を定義するとき，出力関数を明示しないこともあり，その場合はもう 1 つアルゴリズムが必要である．つまり合計 3 つのアルゴリズムを実装することになる．従って，本稿では，「通常の数理的なオートマトン理論とは異なり，ユーザモデル中の出力関数の引数（変数）の数だけでなく，出力関数の存在の有無も考慮する」ことになる．

第 2 は構成要素システムの問題である．シミュレーション実践では，次項(2)のように，複数の要素を結合したシステムを扱うことが一般的である．つまり，複数の要素システム $Si = <\delta i, \lambda i>$ を結合して，複合システム $S = <\delta, \lambda>$ を構成する．そのとき，複合システム S の出力関数の特徴だけで型を判定できない状況が発生する．例えば，複合システムが状態機械であっても，要素システムが出力関数を持つことがあり，単純にはモデルのオートマトン型を推定することができないのである（第 3 節(3)で詳述する）．本稿の主題は，単体システムでなく，複合システムのオートマトン型の推定方法を明らかにすることである．

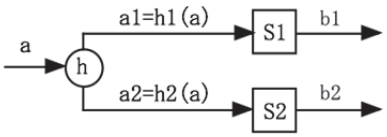
(2) 複合システム

通常のシミュレーション開発では，複数の入力出力システムを結合させて複雑なシステムを構築していく．基本的な結合方式には直列結合，並列結合，フィードバック結合の 3 種類がある（図表 2, 3, 4）．ただし，図中の○印で表わす結合関数 h は複数のシステムを結合するための接着剤に相当する．本稿では，これら 3 つの基本的な結合を任意に有限回適用したシステムを複合システムと呼ぶ．複合システムのモデルは次節で扱う．

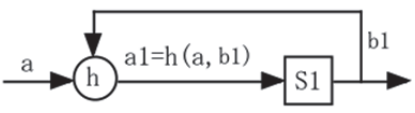
図表 2 直列結合 S1◦S2（筆者作成）



図表 3 並列結合 S1＊S2



図表 4 feed back 結合 $S1^{h}$



また，本稿では，逐次処理システムおよび同期型の並行処理システムについても考察する．**逐次処理**システム（sequential processing system）とは，並びの順に要素が動作する（一度に一つの要素しか動作しない）システムのことである．例えば図表 2 の直列結合システムでは，a1 が要素 S1 に入力され，S1 が動作し，後に a2 = h(b1) が要素 S2 に入力され，S2 が動作することになる．一方，**同期型の並行処理**システム（synchronized concurrent processing system）は，すべての要素システムが同一タイミングだが自律して動作しているシステムである．あるいは，自律的に動作する複数の要素の結合体として複雑なシステムを把握するのである．それぞれの要素は，他の要素の動作を待つことなく，各時刻における入力を直接受けるものと考えればよい．

(3) モデルの型と前提条件

以上，二種類のオートマトン型と三つの結合方法および二つの処理方法を定義した．それらの組み合わせを考慮すると，様々な複合システムに対して異なる処理アルゴリズムが考えられる．しかしながら，並列結合の場合はそれぞれの要素の出力が他の要素に影響を与えず，フィードバック結合には一つの要素しかないので，動作の結果は処理方法が異なっても同じである．よって，実質的に処理方法の異同が意味を持つのは直列結合の場合だけであり，考察対象を絞ること（ある程度の場合分けの削減）ができる．

また本稿では，型の一貫性と型の限定性を議論の前提とする．**型の一貫性**とは，一つのオートマトン型で一貫してモデルを構築することであり，システム設計上の前提である．設計者の意図するオートマトン型に応じて，例えば Moore 型でモデル化するときは，対象システムに属する複数の要素すべてを Moore 型（ただし，その中には出力関数がないもの，すなわち状態機械を含む）でモデル化するという意味である．したがって，本稿で「モデルが Moore 型である」というときは，「複数の構成要素 Si およびそれらを結合した全体システム S がすべて Moore 型でモデル化されている」ことを意味することとする．モデルが Mealy 型であるというときも同様である．

次に，**型の限定性**とは，ユーザモデルのオートマトン型の種類が上記の二種類しかないと仮定することである．したがって，Mealy 型でないモデルは Moore 型であるし，逆に，Moore 型でないモデルは Mealy 型である．ただし，状態機械は両方の型に属するものであるから，論理的には用語の使用に注意が必要である．正確には，「出力関数を持つモデルが与えられているとき，それが Mealy 型でないならば Moore 型であり，逆に，Moore 型でないならば Mealy 型である」となる．以下では，上記の前提条件を仮定して議論を展開する．

3. 複合システムの特徴

この節では，まず複数の Moore 型オートマトンあるいは状態機械を結合した，逐次処理あるいは並行処理を行う複合システムを考察し，複合システムのモデル

の特徴から要素システムの種類を特定できることを明らかにする.

(1) Moore 型，逐次処理

二つの Moore 型オートマトン $S1 = < \delta 1, \lambda 1 >$, $S2 = < \delta 2, \lambda 2 >$ が与えられているとき，前節の基本的な結合方法によって結合された逐次処理システム S のモデル $< \delta, \lambda >$ は次のようになる（図表 5）.

種類	状態 c	状態遷移関数	出力関数
直列結合	(c1,c2)	$\delta(c, a) = (\delta 1(c1, a), \delta 2(c2, h(\lambda 1(\delta 1(c1,a)))))$	$\lambda(c) = \lambda 2(c2)$
並列結合	(c1,c2)	$\delta(c, a) = (\delta 1(c1, h1(a)), \delta 2(c2, h2(a)))$	$\lambda(c) = (\lambda 1(c1), \lambda 2(c2))$
feedback 結合	c = c1	$\delta(c, a) = \delta 1(c, h(a,b1))$	$\lambda(c) = \lambda 1(c)$

図表 5 Moore 型逐次処理複合システムのモデル（筆者作成）

図表 5 を見れば分かるように，オートマトンを結合したシステムもまたオートマトンである．また，そのオートマトン型も保存されている．すなわち「Moore 型オートマトンの逐次処理複合システムは Moore 型オートマトンである」.（この事情は次項の並行処理の場合でも同じである.）さらに本稿では，これら 3 つの基本的な結合を繰り返し適用した複合システムを対象として考察するが，上記のことから,「オートマトン型は，複合システムにおいても保存される」ことが分かる．さらに次の言明が成り立つ.

(*) モデルが Moore 型であり、少なくとも一つの要素が恒等関数でない出力関数を持つならば, 必ず逐次処理の複合システムも恒等関数でない出力関数を持つ.

実際，図表 5 を見ながら，まず並列結合について考察する．並列結合システムの状態は c =(c1,c2)であり，その出力関数は，$\lambda(c) = (\lambda 1(c1), \lambda 2(c2))$である．このとき，$\lambda 1$ と $\lambda 2$ の少なくとも一つが恒等関数でないならば，λ は恒等関数ではない．すなわち結合システムも出力関数を持つ．例えば，S1 が状態機械で S2 が Moore 型オートマトンであるとき，$\lambda 1(c1) = c1$ となるが，$\lambda 2(c2)$が恒等関数でないので，全体として $\lambda(c)$は恒等関数でない．次に，フィードバック結合の場合は，要素は一つであり，$\lambda(c) = \lambda 1(c)$であるから，明らかに $\lambda 1$ が恒等関数でないならば λ も恒等関数ではない.

最後に直列結合の場合を考える．直列結合システムの出力関数は $\lambda(c) = \lambda 2(c2)$ であるから，出力関数は $\lambda 1$ の影響を受けない（敷衍すると，何個の要素が結合しているとしても，最後の要素の出力だけが結合システム全体の出力である）．したがって $\lambda 1$ が何であっても $\lambda 2$ が恒等関数であるなら，$\lambda(c) = c2$ となる．しかしながら，λ は恒等関数ではないことに注意すべきである．結合システム全体の状態

はc =(c1, c2) であるから，λはcからc2を取り出す関数(射影関数: (c1, c2)→c2) なのである．すなわち，直列結合の場合は，たとえすべての要素が状態機械であっても，結合システムは恒等関数でない出力関数を持つのである[1]．すなわち，「Moore 型の要素システム Si がどのような出力関数を持っていても，それらを直列結合したシステム S は状態機械ではない」のである．したがって，論理学的には「少なくとも一つの要素が恒等関数でない出力関数を持つならば，直列結合システムは恒等関数でない出力関数を持つ」という言明は真である．

以上により，Moore 型オートマトンを要素とする逐次処理システムにおいては，基本的な（直列，並列，フィードバック）結合システムは言明（*）を満足しているのである（もちろん，それらの結合を有限回繰り返した複合システムにおいてもその言明を満足している）．

さらに，出力関数が恒等関数であるシステムを状態機械と呼ぶのであるから，言明(*) の「逆」をとると，次の言明(**)が成り立つ．

(**) Moore 型モデルにおいて，逐次処理複合システムが状態機械ならば，すべての要素システムは状態機械である．

(2) Moore 型，並行処理

複数の Moore 型オートマトン $Si = <\delta i, \lambda i>$ $(i = 1,2,3,\cdots,n)$ を要素とし並行処理を行う複合システム S のモデル $<\delta, \lambda>$に関しては，むしろ一般的な構造を見いだすことができる．

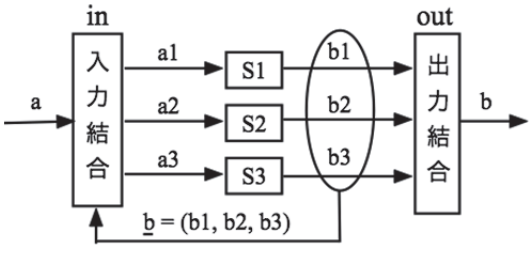
補題 1 (旭, 2013, 命題 1)

1) n 個の Moore 型オートマトン Si を結合した並行処理システム S が与えられたとき，S は (c1, c2,⋯,cn) を状態とするオートマトンとしてモデル化できる．

2) そのモデルは，要素システムを並列に並べ，フィードバック結合を組み合わせて，結合関数 in と out を適切に定義すれば，標準的に作成できる．

3) 例えば，図表 6 は三つの要素を持つ並行処理システムの図式である．結合関数は, 外部からの入力を扱う「入力結合 in」と外部への出力を扱う「出力結合 out」に分けている．結合の基本形態は結合関数で記述できる．特に出力結合 out は射影（projection）である．射影とは引数のいくつかを抜き出す関数のことである．例えば out(b1, b2, b3) = (b2, b3) は射影である．

図表 6　Moore 型並行処理システムのモデル(旭, 2013)



4）このとき，複合システム S の状態遷移関数 δ と出力関数 λ は論理式（1)で与えられる．

$$(1)\quad \begin{cases} \delta(c, a) = c' \Leftrightarrow \\ \qquad \underline{b} = (\lambda 1(c1), \lambda 2(c2), \lambda 3(c3)\ ), \\ \qquad (a1, a2, a3) = in(a, \underline{b}), \\ \qquad c' = (\delta 1(c1, a1), \delta 2(c2, a2), \delta 3(c3, a3)\ ). \\ \lambda(c) = b \Leftrightarrow \\ \qquad \underline{b} = (\lambda 1(c1), \lambda 2(c2), \lambda 3(c3)\ ), \\ \qquad b = out(\underline{b}). \end{cases}$$

モデル理論アプローチでは，通常のオートマトン理論とは異なり，数理モデルを論理式（等号を含む一階述語論理における well-formed formula）で記述する．例えば式(1)の中の，記号 ⇔ は述語論理における必要十分条件を意味する（左辺を右辺で定義していると考えればよい）．また，括弧内のコンマは区切り文字であり，行末のコンマは論理学における連言（and かつ）の略記号である．

補題 2

複数の要素システムを S1, S2, S3 とし, それらが結合した並行処理システムを S とする．複合システム S のモデルは Moore 型であるとする．このとき次の 3 つが成り立つ．

1）結合システム S の出力関数が恒等関数ならば，S を構成するすべての要素システム Si の出力関数は恒等関数である．

2）少なくとも一つの要素システム Si の出力関数が恒等関数でないならば，並行処理システム S の出力関数は必ず恒等関数でない．

3）並行処理複合システム S が状態機械であるならば，S を構成する要素システム Si はすべて状態機械である．

証明

1）補題 1 から，結合システム S の出力関数は，なんらかの出力結合関数 out を用いて，λ(c) = out(λ1(c1), λ2(c2), λ3(c3)) と定義される．これが恒等関数であるならば，結合関数 out( ) が射影であることから，(λ1(c1), λ2(c2), λ3(c3)) = (c1,c2,c3) でなくてはならない．するとすべての出力関数 λi(ci)が恒等関数となる．

2）は 1）の「逆」である．また，出力関数が恒等関数であるオートマトンならば状態機械であることから，3）は 1）を「言い換えた」だけである．　証明終

(3) 状態機械

Moore 型モデルに関する上記の議論をまとめると次のようになる．

命題 3

複数の要素システム Si が結合した複合システムを S とし, そのモデルが Moore 型であるとする．このとき複合システム S が状態機械であるならば，S を構成するすべての要素システム Si は状態機械である．

証明

S が逐次処理システムである場合，第 3 節(1)の言明(**)より，S が状態機械であるならば，すべての要素システム Si は状態機械である．一方，S が並行処理システムである場合，補題 2 の 3)より，S が状態機械であるならば，すべての要素システム Si の出力関数は恒等関数である（すなわち状態機械）である．証明終

この命題 3 は, たとえモデルが Moore 型オートマトンで記述されていると知らされていても，そして「逐次処理と並行処理のどちらであっても」，複合システム S が状態機械であるなら，すべての要素 Si は状態機械であることを意味する．

上記は Moore 型モデルの場合の考察であるが，しかし，Mealy 型のモデルの場合は全体が状態機械であっても構成要素が出力関数を持つことがあるので，注意が必要である. 例えば, 逐次処理を行う Mealy 型直列結合システムの出力関数は, 一般的には $\lambda(c, a) = \lambda 2(c2, \lambda 1(c1, a))$ であるが, $\lambda 1$ と $\lambda 2$ を特別に定義すれば $\lambda(c, a) = c$ となるように構成できる．例えば，$a2 = b1 = \lambda 1(c1, a) = \sqrt{c1}$ および $\lambda 2(c2, a2) = (a2*a2, c2)$ とすればよい（ただし，$c1 > 0$ とする）．このとき，$\lambda(c, a) = \lambda 2(c2, \lambda 1(c1, a)) = \lambda 2(c2, \sqrt{c1}) = (c1, c2) = c$ となる．つまり，要素が出力関数を持つにもかかわらず，全体の出力関数は恒等関数であり，すなわち全体は「結果として」状態機械である．出力関数 $\lambda(c, a)$は存在するけれども恒等関数である．しかしながら，以下に見るように，Moore 型のみが本稿の議論に必須である．

4. オートマトン型の判定

最後に，主題であるオートマトン型の判定方法について考察する．ここでは第 2 節 4)「問題の所在」で述べたように，コンピュータ実装の観点から，オートマトン型だけでなく，状態機械であるが出力関数を記述しない場合も区別する．

複数の要素システムを,「型の一貫性」を保ちつつ結合を繰り返して構成したシステムを S とする．ただし，S のモデルのオートマトン型は不明であるとする．このとき，逐次型処理であっても並行処理であっても，型の判定に関して次が成り立つ．

定理 4（オートマトン型の判定）

複数の要素 Si $(n = 1, \ldots, n)$ から構成されたシステム S のモデル $<\delta, \lambda>$ があり，そのオートマトン型が不明であるとする．このとき，次の 2 つが成り立つ．

1）もしもモデルに 1 変数の出力関数 $\lambda(c)$ の記述があるならば，そのモデルは Moore 型オートマトンで構成されている．

2）もしもモデルに 1 変数の出力関数 λ(c) の記述がないならば，そのモデルは Mealy 型オートマトンで構成されている．

証明
1）モデルに 1 変数の出力関数 λ(c) の記述があると仮定する．図表 1 から，引数 1 の出力関数 λ(c)を持つものは，Moore 型オートマトンのみである．したがって，複合システム全体は出力関数 λ(c)を持つ Moore 型オートマトンであり，型の一貫性から，構成要素も Moore 型オートマトンである．
2）モデルに 1 変数の出力関数 λ(c) の記述がないと仮定する．全体システムの出力関数 λ(c)の記述がないことから，全体システムは状態機械である．ただし，全体が状態機械であるからといって，要素システムも状態機械であると単純には言えないことに留意する（前項で述べたように，要素が Mealy 型であり，それらを結合した最終的な全体システムが状態機械である可能性もある）．

そこで，仮にモデルが Moore 型オートマトンであると仮定するならば，命題 3 の 3）より，逐次処理であろうが並行処理であろうが，要素も全体もすべて状態機械である．したがって，Mealy 型であるとも言える．一方，モデルが Moore 型オートマトンでないと仮定すると，型の限定性（第 2 節(3)）より，モデルは Mealy 型オートマトンで構成されている（本稿では二種類のみを考察対象としている）．

証明終

この定理は，モデルのオートマトン型を判定するには「1 変数の出力関数 λ(c) がモデルに記述されているかを最初にチェックすればよい」ことを意味する．また，もしも λ(c)がモデルに記述されていなければ，証明中の 2）から，次に λ(c, a) が記述されているかをチェックすれば良いことになる．これが自動判定の手順である．

この方法により，次のように，曖昧さの問題と添字の問題が解決されている．第 1 に，Mealy 型の出力関数 λ(c, a)の有無によってモデルの型の判定（特に状態機械の判定）ができないことに注意したい．第 3 節(3)で述べたように，たとえ複合システムが状態機械であっても，その要素システムが出力関数を持つこともあるので，2 変数出力関数の有無にもとづく判定に曖昧さが残る．しかし，出力関数 λ(c)の有無ならば，チェックの方法に曖昧さはない．

第 2 に，次のように，出力関数の「変数の数」だけで型を判定できない状況として添字の問題がある．「添字の問題」とは，要素システム $Si = <\delta_i, \lambda_i>$には添字がついており，その添字付き関数を，プログラム内でどのように定義するかが問題なのである．

モデル理論アプローチが提供するモデル記述言語 CAST では，例えば，添字付き関数を delta(i, c, a)や lambda(i, c)などと定義することが可能である．つまり，2 変数の添字付き状態遷移関数は 3 変数関数として定義できる．同様に 1 変数の

添字付き出力関数は 2 変数関数として定義できる．簡単に言えば，添字を正式な変数として扱うのである．その結果，出力関数について言えば，（変数の数が同じであるため）lambda(c, a)と lambda(i, c)の区別がつかなくなってしまう．人間ならば，「変数 c が状態を表し，lambda(c, a)は全体システム S の Mealy 型の 2 変数出力関数であり，一方，変数 i が添字を表し，lambda(i, c)は要素システム Si の Moore 型の 1 変数出力関数である」ということを文脈から理解するが，コンピュータは意味や文脈を理解せず，単純に変数の数だけを比較するため，これらを区別できないのである．ところが，出力関数 lambda(c) の記述がない場合は，定理 4 により，Mealy 型であることが確定できるので，lambda(c, a)が lambda(i, c)と混同されることはない．

以上，定理 4 はこれらの問題を回避している点で重要である．定理の中の λ(c)は要素システムでなく全体システムの Moore 型の 1 変数出力関数に限られることから，曖昧さがなく，判定条件としてふさわしいのである．

5. おわりに

本稿では，シミュレーションモデルを実行するためのプログラム（開発実行環境あるいは実行エンジン）の設計に関して，モデル理論アプローチにおけるシミュレーション開発の立場から，モデルのオートマトン型（Moore 型または Mealy 型）を判定する方法を明らかにした．

第 3 節命題 3 では，もしも Moore 型複合システム S が状態機械であるならば，S を構成するすべての要素システム Si は状態機械であることを示した．これにより，本稿の主たる結果として，第 4 節定理 4 が導出された．この定理は，モデルのオートマトン型を判定するには「1 変数の出力関数 λ(c)がモデルに記述されているかどうかだけを見ればよい」ことを述べている．簡潔に言えば，全体システム S の出力関数が 1 変数（Moore 型）であるかどうかが決め手である．多数の要素からなるシステムの, 要素一つ一つをチェックする必要がない点が重要である．

また，定理 4 の後で考察しているように，Mealy 型モデルの特殊性や要素のモデル記述における添字の問題から，2 変数の出力関数 λ(c, a) の存在の有無では型の判定は不十分であることも明らかにした．つまり，出力関数に注目するのは当然であるように見えるが，しかし λ(c) 以外は決め手にならないことを明らかにした．

なお，モデル理論アプローチによる情報システム開発のための開発実行環境 MTA-SDK.ova は，Web サイトにて配布している．

https://sites.google.com/a/theoreticalapproach.net/cast/download/

本稿の結果はその中の実行エンジン stdAutomatonEngine.p の修正に当たって利用され，この実行エンジンは，ユーザモデルが Moore 型でも Mealy 型であっても，型を宣言することなく実行することができる．本稿の結果は実践のための理論的裏付けを提示したものである．

[1]上記のように, Moore 型の要素を直列結合した逐次処理複合システムは必ず恒等関数でない出力関数を持つ. すなわち Moore 型逐次処理直列結合は状態機械になり得ない．よって，Moore 型の要素を結合した逐次処理複合システムが状態機械ならば，その内部における結合の種類は並列結合またはフィードバック結合のみである．

参考文献

旭貴朗，高原康彦，中野文平他（2008）「経営情報システム開発のためのモデル記述言語 CAST」『経営情報学会誌』, Vol. 16, No. 4, pp.19-30.

旭貴朗（2009）「モデル理論アプローチによるシミュレーション --- 開発実行環境 Simcast」東洋大学『経営論集』73 号, pp. 33-51.

旭貴朗（2013）「モデル理論アプローチにおける結合システムの形式モデル」東洋大学『経営論集』No. 82, pp. 101-112.

旭貴朗（2017）「モデル理論アプローチによるシミュレーション --- モデル記述言語の改良および開発環境の統合」東洋大学『経営論集』89 号, pp. 35-44.

高原康彦，齋藤敏雄，旭貴朗，柴直樹（2007）『形式手法 モデル理論アプローチ：情報システム開発の基礎』日科技連出版.

高原康彦，齋藤敏雄，旭貴朗，柴直樹，竹田信夫，高木徹（2016）『形式手法 モデル理論アプローチ【第２版】実践編：情報システム開発の基礎』日科技連出版.

（2018 年 1 月 8 日受理）