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

Formal Models of Connected Systems in Model Theory Approach

旭　　貴朗

1　はじめに
2　オートマトン
3　結合システム
4　分割合成問題
5　おわりに

1　はじめに

本稿では，モデル理論アプローチによるシミュレーション（旭 2009）において開発の基礎となる，結合システムのモデルと分割合成問題（Decomposition Problem）について考察する．シミュレーション開発においては，複雑なシステムを要素システムの組み合わせとして構成していくことになる．もしもその構成の一般的な共通枠組みがあるならば，その枠組みはシステム開発に対してよりどころを与えるものとなるだろう．本稿では，シミュレーションシステムの一般的な枠組みを提示したい．考察の対象はオートマトンである．また，分割合成問題とは，与えられたオートマトンを，複数の（より小さな）オートマトンに分割したあと，結合して，もとのオートマトンに再構成できるかを問うことである．これを解くことにより，本稿では，むしろ要素システム間の相互作用の意味を明らかにしたい．

モデル理論アプローチの特徴のひとつは，モデルを集合と関係で定義し，関係（関数を含む）の定義は形式言語の意味の論理式で表わすことである（高原ほか 2007）．本稿では入力や出力の時系列やそれらの間の関係に注目したいので，まずは入力出力システムを集合として定義する．次に論理式を用いてオートマトンの定義を与える．制御理論等の工学的なものと対象は同じであるが，定義の外見は異なるし，結合に関する命題や証明を与えつつ議論を進める．論理式を用いて議論を進める理由は，モデル理論アプローチにおける開発言語 CAST（旭ほか 2008）が論理式を記述するのに適しており，考察の結果がシミュレーション開発にうまくつながるからである（旭 2011）．

以下，第2節でオートマトンと状態機械を定義し，第3節で並列処理の結合システムを定義する．第4節で分割合成問題を解く．第5節は考察である．

2　オートマトン

(1)　入力出力システム

考察の対象は離散時間システムなので，時間の集合を T（0 から始まる自然数の集合）とする．システムに対する入力の値の集合を A と書くことにすると，入力の時系

列は時間関数 x：T→A で表わすことができる（本稿では入力列と呼ぶ）．もちろん任意の時刻 t∈T に対して x(t)∈A である．入力列の集合を A^T と書く．また，出力の値の集合を B と書くことにすると，出力列は時間関数 y: T→B である（y(t)∈B）．出力列の集合を B^T と書く．また，記号 $A^T \times B^T$ は時間関数の集合 A^T と B^T の直積といい，直感的には入力列 x と出力列 y の組 (x,y) のあらゆる組合わせの集合を意味するものである．以下では高原（1974）に従って，定義を行なっていく．

システムの行動 S は，対象システムによって定まる入力列 x と出力列 y の組 (x,y) の集合で表わすことができる．ただし，実際には組 (x,y) の間には何らかの関係があり，あらゆる組合わせが起こるとは限らず，システムの特性に応じて定まった組合わせしか発生しないことは明白である．したがって，システムの行動 S は集合 $A^T \times B^T$ の部分集合である（$S \subseteq A^T \times B^T$）と定義できる．数学では，直積の部分集合を「関係」と呼んでいる．対象システムの入力出力側面での（外部から見た）行動を「関係」で表現したものを，以下では**入力出力システム**と呼ぶことにする．

(2) オートマトンと状態機械

一般に，ある時刻の出力が過去に依存せず，その時刻の入力値のみに依存して定まるような入力出力システムを**静的システム**（static system）と呼び，そうでないものを**動的システム**（dynamic system）と呼ぶ．システム論では，静的システムは，ある関数 h(t):A→B が存在し「$(x,y) \in S \Leftrightarrow (\forall t)(\, y(t) = h(t)\,(x(t))\,)$」と定義される．同時刻に同じ入力値があるなら，同じ出力値を返すシステムである．直感的に言えば，静的システムは何ものも蓄積せず入力の値だけに依存して出力値を決定するシステムのことである．ただし，本稿では時間不変の関数のみを考える（h(t) = h(t')）ので，システムと関数 h:A→B を区別せず，混乱のない限り同じ記号で h と書くことにする．

これに対して，動的システムは，現在以外の入力値（実際には過去の入力値）にも依存して出力値が定まるシステムのことである．初期状態が同じでも，入力履歴が異なれば遷移後の状態が異なり，現在の状態が異なれば同じ入力でありながら出力も異なる場合がある．次で定義する Moore 型オートマトンは，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モデルであるので，オートマトンは動的システムのモデルである．

定義 1（Moore 型オートマトン）

入力出力システム $S \subseteq A^T \times B^T$ に対して，ある集合 C とふたつの関数 $\delta : C \times A \to C$, $\lambda : C \to B$ を定めることができて次の関係が成り立つとき，「S はオートマトンである」あるいは「S はオートマトンとしてモデル化できる」と称する．

$$(x,y) \in S \Leftrightarrow (\exists c \in C^T)(\forall t \in T)(\, c(t+1) = \delta(c(t), x(t)) \text{ かつ } y(t) = \lambda(c(t))\,).$$

ただし，記号 ⇔,∃,∀は，それぞれ論理学における「必要十分条件」,「存在限定詞」「全称限定詞」である．このとき，$<A, B, C, \delta, \lambda>$ を S のオートマトンモデルと呼び，$S = <A, B, C, \delta, \lambda>$ と書く．このとき，関数 δ, λをそれぞれ**状態遷移関数**，**出力関数**と呼ぶ．また，集合 A, B, C をそれぞれ，**入力集合**，**出力集合**，**状態集合**と呼

ぶ．文脈から明らかなときは各集合を省略し，$S=<\delta,\lambda>$と書くことにする．関数δの定義域が$C\times A$であることから，直ちに$(\forall x\in A^T)(\exists y\in B^T)((x,y)\in S)$が成立つ．

さて，通常は，オートマトンは同じ出力でありながら内部の状態が異なる場合も想定されている（$\lambda(c(t))=\lambda(c'(t))$）．しかし場合によっては，出力関数が恒等関数であるようなオートマトンもある．内部の状態をそのまま出力しているシステムであり，出力$y(t)$＝状態$c(t)$となるオートマトンである．出力関数が恒等関数であるようなオートマトンを状態機械（state machine）と呼ぶ．

定義２（状態機械）

入力出力システム$S\subseteq A^T\times C^T$に対して，ある関数$\delta:C\times A\to C$を定めることができて次の関係が成り立つとき，「S は状態機械である」あるいは「状態機械としてモデル化できる」と称する．

$$(x,c)\in S \Leftrightarrow (\forall t)(c(t+1)=\delta(c(t),x(t))).$$

状態機械はもはや出力関数λについて考える必要はないので，$S=<A,C,\delta>$あるいは略記して$S=<\delta>$と書くことにする．当然ながら，二つの状態システム$S=<\delta>, S'=<\delta'>$に対して，$\delta=\delta'$が成立つならば，S と S' は同じものである（$S=S'$）．

【例 1】任意の時刻の入力 x(t)を，そのまま次の時刻に出力するようなシステムを考える．これを状態機械としてモデル化すると，

$$\delta(c(t),x(t))=c(t+1) \Leftrightarrow c(t+1)=x(t).$$

となる．すると，このオートマトン$S=<\delta>$は，常に次の状態が入力の値と等しくなるようなシステムである．入力を一時的に保持するという意味の一次記憶システムである．

【例 2】ニュートン力学では，空間上にあるひとつの物体の運動を次のようにモデル化する．時刻 t における物体の質量を m，位置を u(t)，速度を v(t)とすると，時刻 t における「状態」は$c(t)=(m, u(t), v(t))$である．物体に加えられる力を x(t)とすると，状態遷移関数は次のように書くことができる．

$$\begin{aligned}\delta(c(t),x(t))=c(t+h) \Leftrightarrow\\ a(t)&=(1/m)\cdot x(t),\\ v(t+h)&=v(t)+h\cdot a(t),\\ u(t+h)&=u(t)+h\cdot v(t+h),\\ c(t+h)&=(m,u(t+h),v(t+h)). \quad \text{--- (1)}\end{aligned}$$

ただし，式(1)は論理学でいう論理式であり，式後のコンマは論理学の「かつ（and）」をあらわす省略記号である．「または（or）」は or で表わすこととする（旭 2012）．

ここでは連続時間軸の微分方程式ではなく，離散時間軸で近似した差分モデルを考えており，現在の時刻 t から微小時間 h が経過した時刻（t+h）を「次の時刻」と見

なしている．式(1)は離散近似した運動方程式である．力は物体の状態を変化させるという意味で入力である．力が加速度 a(t)を生み，加速度は次の時刻での速度を決定し，その結果として位置が変化している．もちろん，力が働かないとき（x(t) = 0 のとき）は，速度が一定（v (t+h) = v(t)）となり，等速運動を行なうモデルとなっている．

【例 3】時刻が進むにつれて，状態が 1 ずつ増加するシステムを考える．

$\delta(c(t),x(t)) = c(t+1) \Leftrightarrow c(t+1) = c(t) + 1.$

このオートマトン $S = <\delta>$ は，入力がなくとも自律的に動作するシステムである．たとえば初期状態が 0 だったとすると，時刻 n では状態が n となる．左辺の入力 x(t) が右辺で使用されないので，むしろ，左辺を $\delta(c(t)) = c(t+1)$ と書いたほうがふさわしい．本稿では，入力を考える必要のないシステムを**閉じたシステム**（closed system）と呼ぶことにする．

3　結合システム

複数の入力出力システムが結合してできるシステムを結合システムと呼ぶ．結合形態の視点から分類すると，各要素システムが結合する基本的な形態には，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 1ー図 3）．

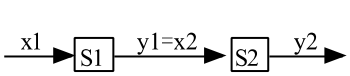

図 1　直列結合

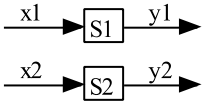

図 2　並列結合

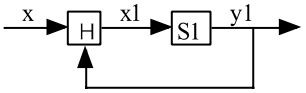

図 3　フィードバック結合

以下では，**並列処理システム**（concurrent processing system）について考えていく．並列処理システムとは，結合の形態にかかわりなく，「すべての要素がいつでも自律的に動いている」ような結合システムのことを言う．例えばインターネットには無数のコンピュータが結合している．各コンピュータは自律的に活動しているので，それら自律的要素がつながったネットワークは並列処理システムである．

定義 3（結合システム）

1）2 つの入力出力システム $S1 \subseteq A1^T \times B1^T$, $S2 \subseteq A2^T \times B2^T$ が与えられているとき，**直列結合システム** $S1 \circ S2 \subseteq A1^T \times B2^T$ を，次のように定義する．

$(x,y) \in S1 \circ S2 \Leftrightarrow (\exists z \in B1^T)((x,z) \in S1$ かつ $(z,y) \in S2)$.

2）2 つの入力出力システム $S1 \subseteq A1^T \times B1^T$, $S2 \subseteq A2^T \times B2^T$ が与えられているとき，**並列結合システム** $S1 * S2 \subseteq (A1 \times B1)^T \times (A2 \times B2)^T$ を，次のように定義する．

$((x1,y1),(x2,y2)) \in S1 * S2 \Leftrightarrow (x1,y1) \in S1$ かつ $(x2,y2) \in S2$.

3）入力出力システム $S1 \subseteq A1^T \times B1^T$ および関数 $h:A \times B1 \to A1$ が与えられているとき，**フィードバック結合システム** $S^h \subseteq A^T \times B1^T$ を，次のように定義する．

$(x,y) \in S^h \Leftrightarrow x1 = h(x,y)$ かつ $(x1,y) \in S$.

直列結合の定義により，例えば $B1 \subseteq A2$ ならば，直列結合は空集合とはならない．一方，例えば $B1 \cap A2 = \phi$（空集合）ならば，S1 の出力 z が S2 の入力になりえないので，直列結合 S1◦S2 は空集合となる．しかしながら，いずれにしても数学的に定義可能であることは確かである．次に，並列結合の定義では実際には結合していないことに注意したい．複数のシステムを並べてひとつのシステムと見なす場合のモデルになっている．またフィードバック結合の定義にある関数 h は出力を入力に結合するための「接着剤」に相当するものである．

【結合システムの一般構造】

定義 3 は 2 要素の結合を定義しているが，実際には任意個数の結合を考えることができる．以下では，これら 3 種類の結合形態を任意に組合わせた結合システムを考える．

一般的には，どんなに複雑に結合したシステムであっても，オートマトンを要素とするすべての（並列処理の）結合システムは，図 4 のように，並列結合とフィードバック結合の組み合わせで図式化することができる．また，結合の形態（直列，並列，フィードバックおよびそれらの組合わせ）は結合関数で記述できる．本稿では，外部からの入力を扱う「入力結合関数 in」と外部への出力を扱う「出力結合関数 out」に分けることとする．

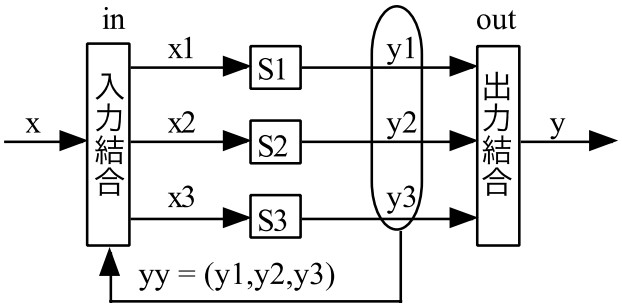

図 4　結合システムの一般的構造（要素数 n=3 のとき）

例えば，S1, S2, S3 を結合したシステム（図 4）の状態遷移関数 δ と出力関数 λ は，$C=C1 \times C2 \times C3$，$yy(t) \in B1 \times B2 \times B3$ とすると，次のようになる．

$$
(2)\ \begin{cases}
\delta(c(t),x(t))= c(t+1) \Leftrightarrow \\
\qquad yy(t)=(\lambda1(c1(t)),\lambda2(c2(t)),\lambda1(c3(t))), \\
\qquad (x1(t),x2(t),x3(t))=in(x(t),yy(t)), \\
\qquad c(t+1)=(\delta1(c1(t),x1(t)),\delta2(c2(t),x2(t)),\delta1(c3,x3(t))). \\
\lambda(c(t))=y(t) \Leftrightarrow \\
\qquad yy(t)=(\lambda1(c1(t)),\lambda2(c2(t)),\lambda1(c3(t))), \\
\qquad y(t)=out(yy(t)).
\end{cases}
$$

【例 4】3 つのオートマトン，$S1\subseteq A1^T\times B1^T$, $S2\subseteq A2^T\times B2^T$, $S3\subseteq A3^T\times B3^T$ が直列に結合した結合システム $S1\circ S2\circ S3\subseteq A1^T\times B3^T$ を考え（図 5），そのすべての要素がいつでも自律的に動いているものとする（並列処理システム）.

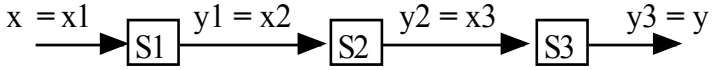

図 5　並列処理直列結合システム

このとき，(2)式に従って δ,λ を定義すると，全体システム $S=<A1,B3,C1\times C2\times C3,\delta,\lambda>$ は集合として直列結合システム $S1\circ S2\circ S3$ と一致するので，$S1\circ S2\circ S3$ のオートマトンモデルは $<A1,B3,C1\times C2\times C3,\delta,\lambda>$ である．実際，$C=C1\times C2\times C3$, $yy(t)\in B1\times B2\times B3$ とし，入力結合と出力結合を，

$in(x(t),yy(t)) = (x1(t),x2(t),x3(t)) \Leftrightarrow x1(t)=x(t),\ x2(t)=y1(t),x3(t)=y2(t).$

$out(yy(t)) = y(t) \Leftrightarrow y(t)=y3(t).$

と定義すると，次が成立つ.

$$
\begin{aligned}
&(x,y)\in S \\
&\quad\Leftrightarrow (\exists c\in C^T)(\forall t\in T)(\, c(t+1)=\delta(c(t),x(t)),\ y(t)=\lambda(c(t))\,) \\
&\quad\Leftrightarrow (\exists c\in C^T)(\forall t\in T)(\\
&\qquad\qquad yy(t)=(\lambda1(c1(t)),\lambda2(c2(t)),\lambda1(c3(t))), \\
&\qquad\qquad y(t)=out(yy(t)), \\
&\qquad\qquad (x1(t),x2(t),x3(t))=in(x(t),yy(t)), \\
&\qquad\qquad c(t+1)=(\delta1(c1(t),x1(t)),\delta2(c2(t),x2(t)),\delta1(c3(t),x3(t))) \\
&\qquad\qquad) \\
&\quad\Leftrightarrow (\exists c\in C^T)(\forall t\in T)(\\
&\qquad\qquad (y1(t),y2(t),y3(t))=(\lambda1(c1(t)),\lambda2(c2(t)),\lambda1(c3(t))), \\
&\qquad\qquad y(t)=y3(t), \\
&\qquad\qquad (x1(t),x2(t),x3(t))=(x(t),y1(t),y2(t)), \\
&\qquad\qquad c(t+1)=(\delta1(c1(t),x1(t)),\delta2(c2(t),x2(t)),\delta1(c3(t),x3(t))) \\
&\qquad\qquad) \\
&\quad\Leftrightarrow (\exists c\in C^T)(\forall t\in T)(\\
&\qquad\qquad c(t+1)=(\delta1(c1(t),x(t)),\ \delta2(c2(t),y1(t)),\ \delta1(c3(t),y2(t))),
\end{aligned}
$$

(y1(t),y2(t),y(t))=(λ1(c1(t)), λ2(c2(t)), λ1(c3(t)))
)
⇔ (∃c=(c1,c2,c3)∈C^T)(∀t∈T)(
c1(t+1)=δ1(c1(t), x(t)), y1(t) =λ1(c1(t)),
c2(t+1)=δ2(c2(t),y1(t)), y2(t) =λ2(c2(t)),
c3(t+1)=δ3(c3(t),y2(t)), y(t) =λ3(c3(t))
)
⇔ (∃y1∈B1^T) (∃y2∈B2^T) ((x,y1)∈S1, (y1,y2)∈S2, (y2,y)∈S2)
⇔ (x,y)∈S1◦S2◦S3.

よって S1◦S2◦S3 = S = < δ , λ > である．つまり，S1◦S2◦S3 は Moore 型オートマトンであり，< δ , λ > がそのオートマトンモデルである．

以上，直列結合を例にして述べてきたが，並列結合とフィードバック結合についても同様である．これらをまとめると次のようになる．

命題 1

n 個のオートマトン Si を要素とする任意の結合システム S が与えられたとき，結合関数 in と out を適切に定義すれば S = ((S1＊S2＊…＊Sn)^in)◦out と書くことができ，それは C1×C2 ×…×Cn を状態集合とするオートマトンである．例えば，n = 3 の場合，図 4 のように図式化でき，その状態遷移関数 δ と出力関数 λ は式(2)で与えられる．

次に，要素システムが状態機械である場合 (Si = < δ i >) の結合システムを考える．状態機械は出力関数を考えない，あるいは状態＝出力とみなせるようなオートマトンのことである．したがって，命題 1 を適用することができて，結合システムを図 4 のように書け，それは C1×C2 ×…×Cn を状態集合とするオートマトンである．また，全体システムの状態遷移関数 δ と出力関数 λ は，例えば n = 3 のとき，

δ(c(t),x(t))= c(t+1) ⇔
(x1(t),x2(t),x3(t))=in(x(t),c(t)),
c(t+1)=(δ1(c1(t),x1(t)),δ2(c2(t),x2(t)),δ1(c3,x3(t))).
λ(c(t))=y(t) ⇔ y(t)=out(c(t)).

となる．しかしながら，全体システムの状態をそのまま出力すること (λ(c(t)) = c(t)) にすれば，出力関数 λ を考える必要はない (図 6)．すなわち，状態機械を結合したシステムを状態機械 < δ >としてモデル化することができる．

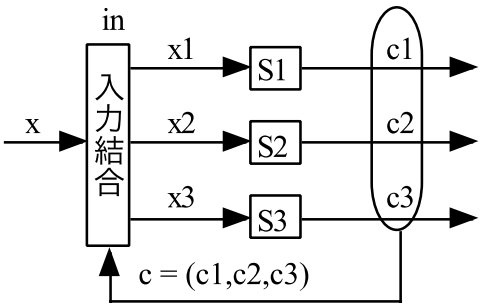

図6　状態機械の結合システム

系2

n 個の状態機械 Si を要素とする任意の結合システム S が与えられたとき，結合関数 in を適切に定義すれば $S = (S1 * S2 * \cdots * Sn)^{in}$ と書くことができ，それは C1×C2×…×Cn を状態集合とする状態機械である．例えば，n＝3 の場合，図6のように図式化でき，その状態遷移関数 δ は次の式(3)で与えられる．

$$\delta(c(t),x(t)) = c(t+1) \Leftrightarrow$$
$$(x1(t),x2(t),x3(t)) = in(x(t),c(t)),$$
$$c(t+1) = (\delta 1(c1(t),x1(t)),\delta 2(c2(t),x2(t)),\delta 1(c3,x3(t))). \quad \text{--- (3)}$$

【例5】二体力学系の場合，上記の入力結合関数 in として，引力の相互作用を考えることができる．空間上に二つの物体があるとき，それらの物体の間には相互に引力が働き，その引力が各物体への入力となる．実際，例2のように，各物体の状態を c1(t) = (m1, u1(t), v1(t)) ， c2(t) = (m2, u2(t), v2(t))とすると，全体システムの状態は c(t) = (c1(t),c2(t))である．また，物体1への物体2からの引力 x1(t)，および物体2への物体1からの引力 x2(t)は，それぞれの質量 m1, m2 と位置ベクトル u1(t) - u2(t)を使って計算できる．したがって，x1(t) =h1(c(t))，x2(t) =h2(c(t)) と書くことができる．そこで入力結合関数を in(c(t)) = (h1(c(t)),h2(c(t)))と定義すれば，これが二つのシステムを結合する結合関数である．例2によれば，そもそも各物体の運動方程式（状態遷移関数 δ 1, δ 2）は式(1)で表わせるので，全体システム（二体力学系）は閉じた状態機械となり，その状態遷移関数は次のようになる．詳細は旭(2012)を参照されたい（そこでは入力結合関数を相互作用 θ と呼んでいる）．

$$\delta(c(t)) = c(t+h) \Leftrightarrow$$
$$(x1(t),x2(t)) = in(c(t)),$$
$$c1(t+h) = \delta 1(c1(t),x1(t)),$$
$$c2(t+h) = \delta 2(c2(t),x2(t)).$$

4　分割合成問題

前節では複数の状態機械<δi>を結合したシステム<δ>が，やはり状態機械として

モデル化できることを見た．本節では，その逆に，与えられた状態機械<δ>を複数の要素システム（状態機械<δi>）に分割し，それらを組み合わせて元の状態機械<δ>を合成できるかという問題を扱う．これを分割合成問題と呼ぶ．ただし注目するのは閉じた状態機械やオートマトンである．まず，状態が 2 項からなる場合 c(t) = (c1(t), c2(t)) を考える（つまり，C = C1×C2 の場合）．

補題 3

状態集合が C = C1×C2 である閉じた状態機械< C, δ>が与えられたとする．状態遷移関数 c(t+1) = δ(c(t)) に対して，通常の関数の分解 (c1(t+1), c2(t+1)) = (δ1(c(t)), δ2(c(t))) を考える．このとき 3 つの関数を，

h(c1(t),c2(t)) = (x1(t), x2(t)) ⇔ x1(t)=c2(t), x2(t) = c1(t).

P1(c1(t), x1(t)) = c1(t+1) ⇔ c1(t+1) = δ1(c1(t), x1(t)).

P2(c2(t), x2(t)) = c2(t+1) ⇔ c2(t+1) = δ2(x2(t), c2(t)).

と定義すると，次式が成り立つ.

$$
\begin{aligned}
&\delta(c(t)) = c(t+1) \Leftrightarrow \\
&\qquad (x1(t), x2(t)) = h(c(t)), \\
&\qquad c1(t+1) = P1(c1(t), x1(t)), \\
&\qquad c2(t+1) = P2(c2(t), x2(t)). \qquad \text{--- (4)}
\end{aligned}
$$

証明

式(4)の左辺を仮定する．関数 h の定義から c2 (t)= x1(t), c1(t) = x2(t) であるので,

$$
\begin{aligned}
&\delta(c(t)) = c(t+1) \\
\rightarrow\ &(c1(t+1), c2(t+1)) \\
&\qquad = (\delta 1(c(t)), \delta 2(c(t))) \\
&\qquad = (\delta 1(c1(t),c2(t)), \delta 2(c1(t),c2(t))) \\
&\qquad = (\delta 1(c1(t),x1(t)), \delta 2(x2(t),c2(t))) \\
&\qquad = (P1(c1(t),x1(t)), P2(c2(t),x2(t))). \\
\rightarrow\ &c1(t+1) = P1(c1(t),x1(t)), \\
&\qquad c2(t+1) = P2(c2(t), x2(t)).
\end{aligned}
$$

逆に式(4)の右辺を仮定する．関数 h の定義から x1(t) = c2(t), x2(t) = c1(t) であるので,

$$
\begin{aligned}
&c(t+1) = (c1(t+1), c2(t+1)) \\
&\qquad = (P1(c1(t),x1(t)), P2(c2(t),x2(t))) \\
&\qquad = (\delta 1(c1(t),x1(t)), \delta 2(x2(t),c2(t))) \\
&\qquad = (\delta 1(c1(t),c2(t)), \delta 2(c1(t),c2(t))) \\
&\qquad = (\delta 1(c(t)), \delta 2(c(t))) = \delta(c(t)). \qquad \text{証明終了}
\end{aligned}
$$

この補題は，状態集合が C = C1×C2 であるすべての閉じた状態機械 S = < δ > は，2 つの状態機械 S1 = <P1>, S2 = <P2>に分割できて，「それらを図 6 のように（ただし要素システムは 2 つとして）結合すれば」，元の状態機械 S と等価になることを意味している．もちろん S1, S2 は，それぞれ c1, c2 を状態とし，それぞれ x1, x2 を入力とする状態機械である．

また h は結合関数であるが，いささか特徴的である．各 Si への入力 xi として，Si 以外のシステムの状態を渡しているのである（x1=c2, x2=c1）．このことは状態 C = C1×C2×…×Cn をもつ閉じたシステムへ拡張することができる．すなわち，

h(c) = (x1,…,xn) ⇔ 各 xi は c=(c1,…,cn)から xi を除いた n-1 項組.

と定義すると，補題 3 を一般化することができる．記法の簡便のため，c=(c1,…,cn)から xi を除いた n-1 項組を $\underline{ci}$ と書き，また δ i[ci, $\underline{ci}$] = δ i(c)と定義する．

命題 4

C = C1×C2×…×Cn を状態集合とする閉じた状態機械 S = < C, δ >が与えられたとする．このとき，n 個の状態機械 Si = <Ai, Ci, Pi> を構成できて，S = (S1＊…＊Sn)^h である．

証明

状態遷移関数 c(t+1) = δ (c(t)) に対して，通常の関数の分解 (c1(t+1),…,cn(t+1)) = (δ 1(c(t)),…, δ n(c(t))) を考え，i = 1,2,3,…,n に対して，

Ai = $\underline{Ci}$.

h(c(t)) = (x1(t),…,xn(t)) ⇔ x1(t)=$\underline{c1}$(t),…, xn(t) = $\underline{cn}$(t). --- (5)

Pi(ci(t), xi(t)) = ci(t+1) ⇔ ci(t+1)= δ i[ci(t), xi(t)].

Si = <Ai, Ci, Pi>.

と定義する．このとき，補題 3 と同様にして，

δ (c(t)) = c(t+1) ⇔

(x1(t),…,xn(t)) = h(c(t)),

任意の i に対して ci(t+1) = Pi(ci(t),xi(t)).

が成り立つ．系 2 より，S = (S1＊…＊Sn)^h である． 証明終了

この命題は，「状態が n 項からなるすべての閉じた状態機械 S = < δ > は，n 個の要素システム S1 = <P1>,…, Sn = <Pn> に分割できて，それらを図 6 のように結合すれば，元の状態機械 S と等価になる」ことを意味している．以上の議論から，次が成り立つ．

定理 5

C = C1×C2×…×Cn を状態集合とする閉じたオートマトン S = < C, B, δ , λ >が与えられたとする．このとき，Ci を状態集合とする状態機械 Si (i = 1, 2,…,n)となんらかの結合関数 h が構成できて，S = ((S1＊…＊Sn)^h)◦ λ である．すなわち，S は図 4

のように分割合成することができる.

証明
与えられたオートマトン S = < C, B, δ , λ > に対して，状態機械 δ= < C,δ>と静的システム λ:C→B を考えると，S はδとλの直列結合である（S =δ◦λ）．また命題 4 より，状態機械δは関数 Pi と結合関数 h を使って図 6 のように分割合成することができる． δ = (S1＊…＊Sn)h．以上により，S = ((S1＊…＊Sn)h)◦λである． 証明終了

これまでは，システムを結合する関数を結合関数と呼んできたが，システム論の観点からは結合関数はシステム間の相互作用のひとつの表現であると考えることができる．定理 5 は閉じたオートマトン<δ , λ>が与えられたとき，複数の要素システム Si に分割し，さらになんらかの結合関数 h が構成できて，それらを図 4 のように結合することにより，元のオートマトン<δ , λ>を合成できることを証明している．関数 h が相互作用である.

命題 4 の式(5)にあるように，構成された結合関数 h の特徴は，各 Si への入力 xi(t) として，Si 以外のシステムの状態 ci(t)を渡している点にある．しかし，この結合関数 h の構成はひとつの方法にすぎず，他にも構成方法があるかもしれないことに注意すべきである．例えば，n 項の状態をちょうど n 個に分割する必要もない．2 個にグループ化して分割することも可能であり，開発者の自由である．分割は non-cohesive であるようにすれば良い（Mesarovic & Takahara 1989 p.372）．また，二体問題（例 5）における結合関数は質量と位置ベクトルから計算される引力 h(c)であり，状態そのもののやり取りではなかった．このことは,「システムの相互作用には複数の表現方法がある」ことを意味する．その理由は，何を要素システムとするかをシステム開発者が決めることができるからである.

しかしながら，相互作用というものは，たとえ直接的に状態そのもののやり取りではないにせよ，間接的には他の要素システムの状態の影響を受けていることは確かである．つまり，ひとつの要素システムは，自身と結合している他の要素システムの「状態の影響」を受ける．システム論の観点から言えば，それが相互作用の本質である．式(5)はそのことを直截に示していることになる.

5 おわりに

本稿では，モデル理論アプローチによるシミュレーションにおいて基本的に必要となる，結合システムの形式的モデルについて考察した．各種のモデルは論理式で表わされている.

第 3 節では，命題 1 により，シミュレーションモデルの基本構造が図 4 であることを確認した．経験上，多くのシミュレーション開発では，まず要素システムの動作 S1, S2, S3 を決定し，その後，それら要素システムの間の相互作用（結合関数）を決定していく．図 4 の結合システムは，S = ((S1＊S2＊S3)in)◦out と書くことができる．もしも S が自分の構成したいイメージと異なるならば，修正を加えることになる（S’ =

((S1’ ＊S2’ ＊S3’)in’)◦out’). 修正を繰り返して，望ましいシステムになるまでシミュレーション開発を行なうことになる．また，結合システムはひとつのオートマトンであるので，複数の結合システムを組合せて，より大きな結合システムを構成していくことの保証が得られる．そのことは，オートマトンの標準理論からすでに知られたことである．

しかしながら，モデル理論アプローチの観点からは，むしろ結合システムのモデルの「論理式による表現」が式(2)の形をしていることが重要である．実際，モデル理論アプローチにおけるモデル記述言語 CAST による結合システムの実装構造は式(2)とまったく同じであり，式(2)を具体的に記述すれば，動くシミュレーションを実際に作成することができる（旭 2013 Web 公開資料，図 10）．すなわち，モデル理論アプローチによるシミュレーションに対する本稿の意義は，実践に対して理論の裏付けを与えたことにある．

第 4 節では，閉じたオートマトンに限定したが，分割合成問題を解いた（定理 5）．さらに，分割よりも，むしろ合成において，どのように結合関数を構成するかに注目した．命題 4 の式(5)で構成した結合関数 h は，各 Si への入力 xi(t)として，Si 以外のシステムの状態 ci(t)を渡している．第 4 節の最後で述べたように，相互作用の観点からみれば，結合関数の本質は，自身と結合している他の要素システムの「状態の影響」を直接的・間接的に表現することにある．

【参考文献】

旭貴朗，高原康彦，中野文平ほか（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（2011）「モデル理論アプローチによるシミュレーション ― 自律分散システムの開発手順 ―」経営論集 78 号，pp.177-188.

旭貴朗(2012)「二体力学系の相互作用のシミュレーションモデル」東洋大学経営論集 80 号，pp.31-38.

高原康彦（1974）『システム工学の理論』日刊工業新聞社

高原康彦ほか（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

Mesarovic & Takahara (1989), *Abstract Systems Theory,* Springer-Verlag.

旭貴朗「技術資料：Moore 型オートマトン多層ネットワークのモデル」
http://www2.toyo.ac.jp/~asahi/research/simulation/docs/mooreNetwork.doc (2013.9.5)

（2013 年 9 月 6 日受理）