9F−7

CCSプロセスのネット表現*

川本 真一　伊藤 貴康†
東北大学 工学部 情報工学科‡

1 はじめに

同時進行性の表現に優れたモデルとして、ペトリネットがある。著者らは、このペトリネットの上に、プロセスの代数的な性質やモジュラー性をも表現可能な体系である構造化ネット[1]を提案している。

本発表では、この構造化ネットを用いて、CCS[2]プロセスの表現を行う。まず、CCSプロセス動作式に対するネット表現を行う。これによって、本来インタリーブに基づいたプロセスのふるまいが同時進行的に解釈される。次にCCSプロセスのエージェントとしての側面も構造化ネットで表現可能なことを示す。

2 構造化ネット

本稿ではネットを (P,T,A,δ) という4項組として考える。ただし、P はプレースの集合、T はトランジションの集合、A はアークの集合、δ はトランジションラベルの集合である。

構造化ネットは、並列プロセスの代数的性質とモジュラー性を表現する枠組としてネット上に構築された体系で、プリミティブネット、構造化ネットオペレーション、インタフェースより構成される。

例えば、次のような構文によって定まる構造化並列言語の構造化ネットによる表現は図1となる。

P	:=	**begin** S_1 **end**	(ブロック化)
	|	**begin** $S_1;\cdots;S_n$ **end**	(逐次)
	|	**altbegin** $S_1+\cdots+S_n$ **altend**	(選択)
	|	**if** b **then** S_1 **else** S_2 **fi**	(条件分岐)
	|	**parbegin** $S_1\|\cdots\|S_n$ **parend**	(並列)
	|	**while** b **do** S_1 **od**	(繰り返し)
S_i	:=	*primitive*	
	|	P	
b	:=	*boolean*	

プリミティブ a に対しては(1)のようなプリミティブネットによって表し、記号では $N(a)$ とする。ブロック化構文 に対しては(2)のように表現される。$N(S_1)$ の入出力に、λ と λ^{-1} というネットが付いており、これがbeginとendによるブロック化を表す。これらを逐次インタフェースと呼び、Λ, Λ^{-1} という記号で表す。(2)のネット全体は、$\Lambda\odot N(S_1)\odot\Lambda^{-1}$ という記号によって表される。逐次構文に対しては(3)のようになる。$N(S_1),\cdots,N(S_n)$ を逐次合成([;])し、それに Λ と Λ^{-1} を付けたもので、記号表現は $\Lambda\odot(N(S_1)[;]\cdots[;]N(S_n))\odot\Lambda^{-1}$ とする。選択は、(4)のようになる。$N(S_1)$ から $N(S_n)$ を選択([+])演算によって合成し、それに Λ と Λ^{-1} を付ける。記号表現は、$\Lambda\odot(N(S_1)[+]\cdots[+]N(S_n))\odot\Lambda^{-1}$ とする。条件分岐に関しては、(5)となる。記号表現は、$\Lambda\odot((N(b)[;]N(S_1))[+](N(\neg b)[;]N(S_n)))\odot\Lambda^{-1}$ とする。並列実行に関しては、(6)のようになる。$N(S_1)$ から $N(S_n)$ のネット表現を並列合成([|||])によって合成し、その入出力に λ_p と λ_p^{-1} というネットを付ける。これらを並列インタフェースと呼び、記号では Λ_p, Λ_p^{-1} と表す。これらは、parbegin, parendによるブロック化を表現する。記号表現は、$\lambda_p\odot(N(S_1)[|||]\cdots[|||]N(S_n))\odot\lambda_p^{-1}$ とする。繰り返しは、(7)のようになる。$N(b)$ と $N(S_1)$ の逐次合成に繰り返し[*]を適用し、それに $N(\neg b)$ を逐次合成し、Λ と Λ^{-1} を付ける。記号表現は、$\Lambda\odot((N(b)[;]N(S_1))^{[*]}[;]N(\neg b))\odot\Lambda^{-1}$ とする。以後、構造化ネットの記号表現を構造化ネット式と呼ぶ。

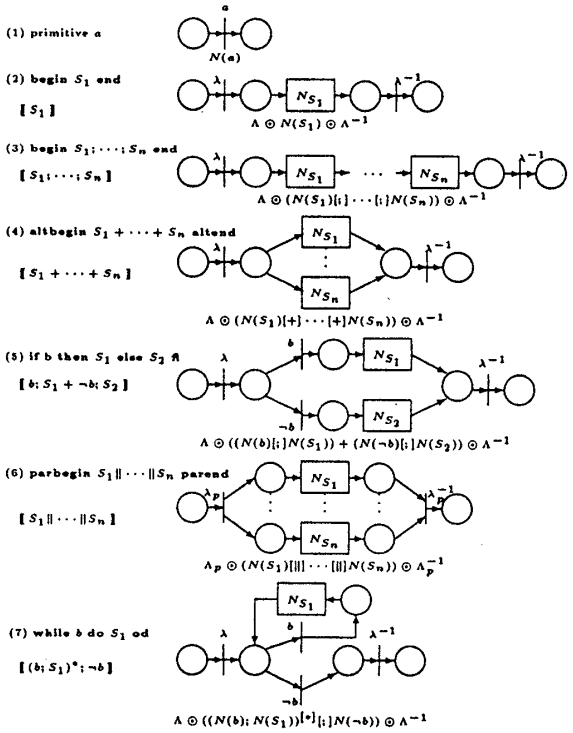


図1: 構造化並列言語の構造化ネット表現

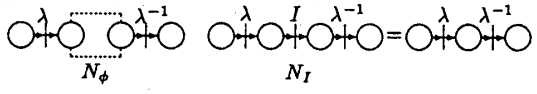
構造化ネットの合成

構造化ネットに対する構造化ネットオペレーションの適用は次の規則に従う。

$$(\Lambda\odot N_1\odot\Lambda^{-1})[;](\Lambda\odot N_2\odot\Lambda^{-1}) \Rightarrow \Lambda\odot(N_1[;]N_2)\odot\Lambda^{-1}$$
$$(\Lambda\odot N_1\odot\Lambda^{-1})[+](\Lambda\odot N_2\odot\Lambda^{-1}) \Rightarrow \Lambda\odot(N_1[+]N_2)\odot\Lambda^{-1}$$
$$(\Lambda\odot N_1\odot\Lambda^{-1})[|||](\Lambda\odot N_2\odot\Lambda^{-1}) \Rightarrow \Lambda_p\odot(N_1[|||]N_2)\odot\Lambda_p^{-1}$$
$$(\Lambda\odot N\odot\Lambda^{-1})^{[*]} \Rightarrow \Lambda\odot N^{[*]}\odot\Lambda^{-1}$$
$$(\Lambda\odot N)^{[\infty]} \Rightarrow \Lambda\odot N^{[\infty]}$$

空ネットと単位ネット

- 空ネット N_ϕ: $\Lambda\odot\phi\odot\Lambda^{-1}$
- 単位ネット N_I: $\Lambda\odot I\odot\Lambda^{-1}=\Lambda\odot\Lambda^{-1}$



インタフェースの簡約

インタフェースは次の規則に従って簡約可能である。

$$\Lambda\odot(\Lambda\odot N\odot\Lambda^{-1})\odot\Lambda^{-1}\Rightarrow\Lambda\odot N\odot\Lambda^{-1}$$

構造化ネットの代数的性質

零元: $N_\phi[;]N=N[;]N_\phi=N_\phi$

単位元: $N_I[;]N=N[;]N_I=N$

$N_\phi[+]N=N[+]N_\phi=N$

$(N_1[;]N_2)[;]N_3=N_1[;](N_2[;]N_3)=N_1[;]N_2[;]N_3$

$N[+]N=N$

$N_1[+]N_2=N_2[+]N_1$

$N_1[|||]N_1=N_2[|||]N_1$

---

*Net Representation of CCS Processes

†Shinichi KAWAMOTO, Takayasu ITO
‡Department of Information Engineering, Faculty of Engineering, Tohoku University

例

altbegin altbegin a+b altend + altbegin c+d altend altend の構造化ネット式は、

$\Lambda\odot((\Lambda\odot(N(a)[+]N(b))\odot\Lambda^{-1})[+](\Lambda\odot(N(c)[+]N(d))\odot\Lambda^{-1})\odot\Lambda^{-1}$

$\Rightarrow \Lambda\odot(\Lambda\odot(N(a)[+]N(b)[+]N(c)[+]N(d))\odot\Lambda^{-1})\odot\Lambda^{-1}$

$\Rightarrow \Lambda\odot(N(a)[+]N(b)[+]N(c)[+]N(d))\odot\Lambda^{-1}$

と簡約され、もともとプロセスの構造を反映した図2(a)の構造化ネットが、a,b,c,d のうちのどれかが選択されるという意味を表現した(b)の構造化ネットへと変換される。

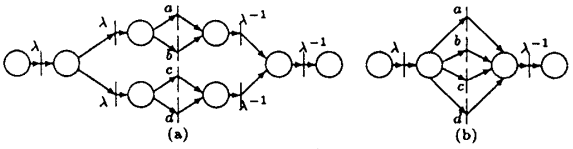


図2

3 CCS プロセスの構造化ネット表現

CCS は、プロセスを相互に通信しあうエージェントとしてモデル化し、そのエージェントのふるまいをプロセス動作式によって表現する代数的な体系である。CCS プロセスの構造化ネットによる表現としては、プロセス動作式を構造化ネットによって表現する方法と、エージェントを構造化ネットによって表現する方法の2つが考えられる。ここでは、この両方のアプローチから CCS プロセスの構造化ネットによる表現を行う。

3.1 CCS プロセス動作式の構造化ネット表現

CCS のプロセス動作式 E は以下のように定義される。

$$E = 0 \mid x \mid \alpha.E \mid E+E' \mid E|E' \mid \mu x.E$$

0 は無動作プロセス。$\alpha.E$ は通信 α を行い E のようにふるまう。$E+E'$ は E かまたは E' のどちらかのようにふるまう。$E|E'$ は E と E' を並列に実行する。$\mu x.E$ は E のようにふるまうが、E に存在するすべての x は、$\mu x.E$ への再帰を表している。

CCS プロセス動作式のネット表現は、まずポート通信を1つのアクションと考え、これをプリミティブネットで表現する。また、プロセス演算のアクション前置は逐次合成 [;] によって表され、非決定的選択 + は選択 [+] によって表される。再帰 $\mu x.$ は無限繰り返し [∞] によって表される。合成 | は、並列合成 [|||] によって表される。ただし [|||] は、相補ポートに対応する2つのトランジションを融合する。例えば、図3(a) と (b) はプロセス P とプロセス Q が相補ポート b と $\overline{b}$ を持つので、これらが並列合成されると、トランジションの融合が起り図3(c) のような構造化ネットが得られる。また、CCS の合成 | には結合則が成り立つので、[|||] に関し次の規則を定める。

$$(\Lambda_p\odot N_1\odot\Lambda_p^{-1})[|||](\Lambda\odot N_2\odot\Lambda^{-1})\Rightarrow\Lambda_p\odot(N_1[|||]N_2)\odot\Lambda_p^{-1}$$
$$(\Lambda\odot N_1\odot\Lambda^{-1})[|||](\Lambda_p\odot N_2\odot\Lambda_p^{-1})\Rightarrow\Lambda_p\odot(N_1[|||]N_2)\odot\Lambda_p^{-1}$$
$$(\Lambda_p\odot N_1\odot\Lambda_p^{-1})[|||](\Lambda_p\odot N_2\odot\Lambda_p^{-1})\Rightarrow\Lambda_p\odot(N_1[|||]N_2)\odot\Lambda_p^{-1}$$

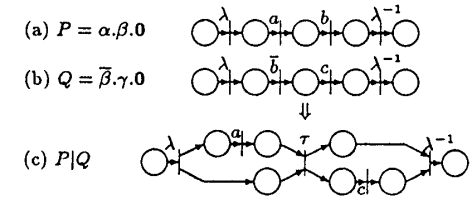


図3: 相補ポートを表現するトランジションの融合

図4の中央は、その左の CCS プロセス動作式に対する構造化ネット表現である。一方、右側は Olderog[3] の方法によるネット表現である。

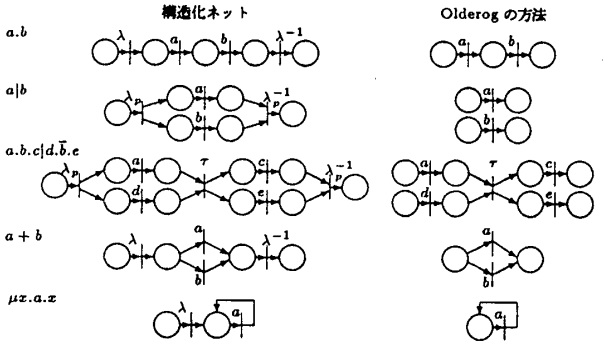


図4: CCS プロセスの構造化ネットによる表現

例えば、プロセス $a|b$ の Olderog によるネット表現は、独立した2つのネットが存在するだけで、これが1つの並列に動作するプロセスであることは明確ではないが、構造化ネットではインタフェースによってモジュール化されており、並列に動作する1つのプロセスであることが明確に表現される。

3.2 CCS エージェントの構造化ネット表現

CCS エージェントは、プロセス間の通信に使用される入出力ポートのみを記述したものである。この通信ポートを、構造化ネットのインタフェースによって表現する。各ポートを用いた通信の順番には特に決まりがなく、同時進行的に発生しても良い。このエージェントを、プロセス動作式で表現すると、通信は逐次的 (インタリーブ) になってしまうが、構造化ネットを用いた記述によって、本来の同時進行的な表現が可能となる。

図5の (a)(b) はエージェント P,Q の構造化ネットによる表現を示している。α や β などの入力ポートは、λ_α , λ_β などのラベルの付いた入力インタフェースによって表現される。一方出力ポート $\overline{\beta}$ は、λ_β^{-1} というラベルの付いたトランジションによって表現される。これらのネットが並列合成されると、図5(c) のように、相補ポートを表すインタフェース Λ_β と Λ_β^{-1} が縮退され、1つのプレースとなる。このプレースは、エージェントの相補ポート間の同期通信動作である τ を表現している。また複数の入出力ポートがあれば、それらは並列インタフェース Λ_p と Λ_p^{-1} によってまとめられ、並列に動作するエージェントによって行われる1つのプロセスであることが明確に表現される。

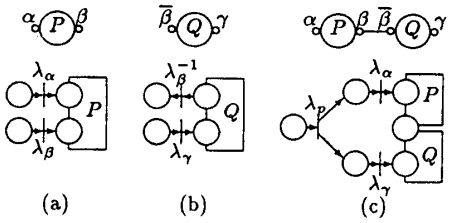


図5: エージェントのネット表現

4 まとめ

構造化ネットを用いて、CCS プロセスの2つの側面であるプロセス動作式とエージェントの表現を行った。構造化ネットによるプロセス動作式の表現は従来の方法に比べ、より明確な表現であることを示した。また従来の方法では明確に取り扱うことのできなかったエージェントの表現が構造化ネットでは可能であることを示した。

[1] 川本真一 伊藤貴康, 並列プロセスのネット表現, 日本ソフトウェア科学会第9回大会論文集, 1992
[2] Robin Milner, Communication and Concurrency, Prentice Hall, 1989
[3] Ernst-Rüdiger Olderog, *Operational petri net semantics for CCSP*, Advances in Petri Nets 1987, LNCS 266, 1987.

本研究は科研費一般 (A)01420029 及び (A02)02249102 の一環として行われたものである.