ハイパーグラフ書き換え系への構文駆動で compositional な構文・意味論の提案

佐野 仁　上田 和紀

グラフ書換え系における一般的なハイパーグラフの定義は，頂点集合，辺集合，頂点から辺への対応とラベリング関数などからなり，サブグラフへのマッチングや生成はそれらへの射を用いて定義される．ただし，これは λ 計算や π 計算のように構文駆動な意味論とは言い難い．そこで，本研究ではグラフ書き換えに基づく計算モデル Flat LMNtal にプロセス代数における名前の隠蔽の構文・意味論を組み込むことで，ハイパーグラフ書き換え系の構文駆動で compositional な構文・意味論を提案する．また，さらにこの上でいくつかの性質について証明を行い，その妥当性を確認した．

1　はじめに

ハイパーグラフ[3] とは，頂点と 2 つの頂点を接続する辺の集合からなるグラフを，辺がそれぞれ 1 つ以上の頂点を結ぶことができるように拡張したものである．ハイパーグラフはコンピュータサイエンスに普遍的に出現するデータ構造であり，インターネットなどに代表されるようなマクロなネットワーク構造から，言語処理系の実装などの低レイヤに至るまで，多くのものがハイパーグラフを用いて抽象化可能である．例えば，C 言語などでのポインタは $n\,(n \geq 0)$ 個のポインタが一つのアドレスを共有するという意味で $n\,(+1)$ 個の端点を結ぶ,（有向）ハイパーリンクになっている．関数型言語においては，これらの言語上で提供されるデータ構造としては木しかないものの（とはいえ，木もグラフである），その実装においては，基本的にハイパーグラフを扱う必要がある．これらを形式的に扱うには，ハイパーグラフを第一級に扱うシンプルで精密な計算モデルがあるのが望ましい．

（ハイパー）グラフを第一級に扱うプログラミング言語または計算モデルのパラダイムとしてグラフ書き換え系[4][13][12] がある．グラフ書き換え系は，書き換え規則に基づき，グラフのマッチング及びマッチングしたサブグラフの書き換えを行って計算を進める．文献[3] にあるように，ハイパーグラフは一般的には頂点 ID の集合（頂点集合）と頂点 ID の集合の多重集合（辺集合）の組で定義される．グラフ書き換え系における（ハイパー）グラフの定義は様々なバリエーションがあるが，典型的には前述したハイパーグラフの定義の派生であり，頂点 ID の集合と辺 ID の集合と辺から頂点への対応に加えて，頂点や辺のラベリング関数などが付随する[4]．書き換え規則は典型的には (L, K, R) のように，マッチングを行うサブグラフ，書き換え規則における左辺と右辺の頂点や辺の対応，生成するサブグラフの三つ組からなり，書き換えは，グラフへの射を用いて L とマッチし，K によって保存する頂点や辺やラベルを対応させながら R によって新たなグラフを生成する *pushout* を行う．

ただし，これは *Structural Operational Semantics*（SOS）[11] とは大きく異なり，

- 構文から直接的に操作的意味論が定義できてい

---

* Introducing a syntax-driven and compositional syntax and semantics to hypergraph rewriting system This is an unrefereed paper. Copyrights belong to the Author(s).

Jin Sano, Kazunori Ueda, 早稲田大学基幹理工学研究科情報理工・情報通信専攻, Dept. of Computer Science and Communications Engineering, Graduate School of Fundamental Science and Engineering, Waseda University.

るわけではない，つまり構文駆動でないため，実際のプログラムとの対応がそこまで自明でない．

- 圏論などの数学的素養を要求し，少なくとも一般のプログラマにとっては複雑である．
- compositional でない．ここで，compositional とは部分から全体を構成可能だという意味である．先述のようなグラフの定義では，例えば頂点集合と辺集合の組からなるグラフを二つ持ってきたときに，それらの合成がどのように定義されるのかは自明でないため，素直に compositional とは言い難い．また，グラフ書き換え系におけるグラフのマッチングに用いる射は，部分から全体を構成するようなものではないため，これも compositional でない．

対して，広く受け入れられている計算モデル（λ 計算や π 計算[14] など）はほとんど全て，構文駆動な SOS に基づいて定義されており，構文と直結していてシンプルで，構造的帰納法などを用いることにより性質の証明が比較的しやすいという特徴がある．

階層グラフ書き換えに基づく SOS を持つプログラミング言語として，LMNtal が存在する[16]．ただし，LMNtal におけるグラフの辺（リンク）はハイパーリンクではないため，端点を高々二つしかもてない．LMNtal の拡張として，これにハイパーリンクの機能を加えた HyperLMNtal[17] が存在している．ただし，LMNtal には形式的な定義があるが，HyperLMNtal [17] では LMNtal からの拡張を非形式的に述べているだけにとどまり，精密な意味論を与えているわけではない．ここで，HyperLMNtal は LMNtal 独自の機能であるグラフの階層化（膜）などにも対応しており，また通常のリンクとハイパーリンクは区別され，それらの接続も可能など，相当に複雑である．

そこで，本研究では，LMNtal から階層化の機能を省いた計算モデルである Flat LMNtal のリンクを，プロセス代数の名前の隠蔽の構文・意味論を導入することで，構文駆動性を保ったままハイパーリンクに拡張した計算モデルとして *Flat HyperLMNtal* を定義し，提案する．これはミニマルなハイパーグラフ書き換え系の形式的定義になっていると主張する．

1.1 本稿の構成

まず，2 節で関連研究について述べ，3 節で提案する Flat HyperLMNtal の意味論の概要を述べる．次に，4 節でこれを精密に定義する．また，5 節でこの意味論が構文駆動な計算モデルに普遍的に見られる性質を満たすことを検証する．そして，6 節では，これを応用して階層ハイパーグラフ書き換え言語である HyperLMNtal の形式化を行う．最後に，7 節で本稿をまとめる．

2 関連研究

グラフ書き換えに基づくプログラミング言語として，GP 2[1]，Dactl[6] などが存在する．GP 2 はグラフのマッチングと書き換え以外にもいくつかの制御構造を持ち，それらに関しては SOS が定義されているが，グラフのマッチング及び書き換えは，典型的なグラフ書き換え系の意味論である double-pushout approach[13] に基づくもので，1 節で触れた通り，構文駆動でない．

Dactl は論理型言語や関数型言語などの項書き換えに基づく言語のコンパイラのターゲット言語（CTL）を目指したグラフ書き換え言語である．ここで，Dactl が計算に用いるグラフは，辺の始点（ノードが持つポインタ）に全順序のついた有向グラフであり，複数の辺が同じ頂点を始点としてもつことが許されるため，ハイパーグラフとなっている．Dactl においてもグラフのマッチング及び書き換えはグラフへの射を援用するものとなっており，構文駆動でない[2]．ここで，注目に値するのは *matching*（ルール左辺のマッチ），*building*（ルール右辺のグラフ生成）を行った後に *redirection* を行うことである．Redirection とは，他のノードが持つ参照をあるアドレスの方へ向け直すことで，これは，無向グラフ書き換え系においては，ある辺を他の辺と融合させることに対応する．

Redirection は特に論理型言語や関数型言語の実装において普遍的に見られる概念であり，例えば遅延評価を行う関数型言語の仮想機械 G-Machine[8] においても重要な役割を果たす．G-Machine の実装をハイパーグラフ書き換え言語 HyperLMNtal[17] を用いて行った研究[15] では redirection を行うためのノード，

indirection ノードをハイパーリンクの融合を用いて簡潔に実装できることを示した．従って，ハイパーグラフ書き換え系を定式化するにあたって，ハイパーリンクの融合が精密に形式化できることは重要であると言える．

ネットワークを扱う SOS で定義される並行計算モデルとして，π 計算などが存在する．π 計算はモバイルネットワークをモデル化するために提案された計算モデルであるから，この意味論がハイパーグラフ書き換え系と共通項を持つのは自然である．実際，文献[9] においては multi-arc な（ハイパーリンクであることを許す） *flow graph* を用いて例題の説明がなされている．しかし，π 計算はあくまでネットワークを介した通信をモデル化する計算モデルであり，グラフのマッチングや書き換えを行うグラフ書き換え系とは異なる．ここで，π 計算を先述した典型的なグラフ書き換え系の意味論を用いて定義した研究は存在している[5] ものの，ハイパーグラフ書き換え系を π 計算の意味論を援用して定式化したという研究は著者の知る限り存在しない．

また，π 計算の派生として，Fusion calculus がある[10]．Fusion calculus では π 計算とは異なり，チャネルを受信する際に受信したチャネルを局所的に束縛するのではなく，大域的にそれらのチャネルを同じものとする，**チャネルの融合（fusion）** を行う．Fusion calculus における fusion はハイパーリンクの融合と直に対応する概念であるので，これは重要な関連研究である．ただし，これはあくまで π 計算の派生であって，グラフ書き換え系の定式化と直に結びつくわけではない．

グラフ書き換えに基づく構文駆動な計算モデルかつプログラミング言語として，LMNtal が存在する[16]．ただし，LMNtal におけるグラフの辺（リンク）はハイパーリンクではないため，端点を高々二つしかもてない．LMNtal にはグラフや書き換え規則からなるプロセスを階層化する膜と呼ばれる機能が存在し，この機能を用いてリンクの端点のノードを集約させることによって，ハイパーリンクをエンコードすることは可能である．ただし，階層化は単なるハイパーリンクの表現よりも強力な機能であるために，特に膜を貫くリンクに関する意味論は相応に複雑になっている．また，プログラムも複雑化する．

3 提案する意味論の概要

本研究では階層化の機能を省いた LMNtal（Flat LMNtal）をベースに拡張する形でハイパーグラフ書き換え系を形式化した．LMNtal には connector と呼ばれる，$=$ を名前（ラベル）に持つ特殊なアトム（ノード）が存在する．これは構造合同規則 (E9) によって，以下のようにリンクの融合を行う．

$$X = Y, P \quad \equiv \quad P[Y/X]$$

（ただし X は P の自由リンク，P はアトム）

ここで，$P[Y/X]$ は**リンク代入**で，P に出現する全てのリンク X を Y で置き換えるものである．また，あるプロセスにおける自由リンクとはそのプロセスに 1 回のみ出現するリンクである．プロセスに 2 回出現したリンクは局所リンクと呼ばれる．LMNtal のリンクはハイパーリンクではないため．高々2 回しか出現できないという構文制約がある．そのため，このように，リンクの出現回数を数えるだけで局所リンク（局所変数）と自由リンク（自由変数）を見分けることができる．ただし，ハイパーリンクは出現回数に制限がないため，このように見分けることはできない．

ハイパーリンク代入もハイパーでないリンクと同様に $P\langle Y/X\rangle$ で，P に出現する全てのハイパーリンク X を Y で置き換えるものと定義したとする．ここで，ハイパーリンクに局所性を定義できていなかった場合は，ハイパーリンクの融合（fusion）がハイパーリンク代入を行うとき，代入すべき全てのハイパーリンクを代入し終えたかどうかが判別できない．このため，fusion，$\bowtie$ に関する構造合同規則を，単純に

$$X \bowtie Y, P \equiv P\langle Y/X\rangle$$

のように定義すると，例えば，

$$(a(X), (X \bowtie Y, b(X, Y))) \tag{1}$$

は

$$(a(Y), b(Y, Y)) \tag{2}$$

と構造合同であるべきだが，

$$(a(X), b(Y, Y)) \tag{3}$$

とは構造合同であってはならないはずなのに，これが

導出できてしまう．ここで，(2) ≡ (1) ≡ (3) を認めてしまうと，任意の異なるハイパーリンク（この例では X, Y）を同じハイパーリンクへ書き換えたものと構造合同だと言えてしまい，プログラムに出現する全てのハイパーリンクが等しいことが導出できてしまって，グラフ書き換え系としての意味をなさなくなる．

これを防ぐためには，X を Y と融合する（ハイパーリンク代入を行う）ときに P **の外で** X **が出現しないという保証** が必要である．ただし，これは「プログラム全体」について言及してしまっているため，部分だけで意味を定義することができておらず，このままでは compositional でない．また，これは P の構文から導かれる制約となっておらず，構文駆動でない．

LMNtal では (E9) の付帯条件「ただし X は P の自由リンク」が，X が $X = Y, P$ のどちらにも出現していることを要求し，$X = Y, P$ において **局所リンク** であることを保証するので，この問題は発生しない．

そこで，本研究では，局所リンクと自由リンクという概念を構文駆動かつ compositional にハイパーリンクに持ち込むために，λ 計算における $\lambda x.E$ や π 計算における $(\nu x)P$ (*channel creation*) のように．明示的なスコープ（*hyperlink creation*）を新たな構文として導入することにした．

提案する構文・意味論においては，構文 $\nu X.P$ がハイパーリンク X が P において局所ハイパーリンクであることを保証する．そして，LMNtal における (E9) に対応する規則，(E6) は

$$\nu X.(X \bowtie Y, P) \equiv \nu X.P\langle Y/X\rangle$$

（ただし X または Y が P に自由出現する）

のようになる．

4 提案するハイパーグラフ書き換え系の構文と操作的意味論

構文駆動かつ compositional なハイパーグラフ書き換え系のミニマムな形式的定義として，以下に述べる Flat HyperLMNtal を提案する．

プロセス			
P	::=	$\mathbf{0}$	Null
	|	$p(X_1, \ldots, X_m)$	Atom, $m \geq 0$
	|	(P, P)	Molecule
	|	$\nu X.P$	Hyperlink creation
	|	$(P :\text{-}\ P)$	Rule

図 1　Flat HyperLMNtal の構文

$$\begin{aligned}
fn(\mathbf{0}) &\stackrel{\text{def}}{=} \emptyset \\
fn(p(X_1, \ldots, X_m)) &\stackrel{\text{def}}{=} \bigcup_{i=1}^{m} \{X_i\} \\
fn((P, Q)) &\stackrel{\text{def}}{=} fn(P) \cup fn(Q) \\
fn(\nu X.P) &\stackrel{\text{def}}{=} fn(P) \setminus \{X\} \\
fn((P :\text{-}\ Q)) &\stackrel{\text{def}}{=} \emptyset
\end{aligned}$$

図 2　Flat HyperLMNtal のプロセスの自由ハイパーリンクの集合

4.1 Flat HyperLMNtal の構文

4.1.1 プロセス

Flat HyperLMNtal は二つの構文要素から成り立つ：

- X はハイパーリンクを表す．
- p はアトム名を表す．唯一の予約名は $\bowtie$ である．

Flat HyperLMNtal の構文は図 1 に示される．アトム $X \bowtie Y$ は *fusion* と呼び，直感的には引数のハイパーリンク同士を繋げる機能を持つ．プロセス P における自由ハイパーリンクの集合を $fn(P)$ で表し，図 2 のように帰納的に定義する．

4.1.2 ルール

ルール $(P :\text{-}\ Q)$ において，P はルールの左辺と呼び，Q はルールの右辺と呼ぶ．ルール $(P :\text{-}\ Q)$ は以下に示す条件を満たす必要がある．

1. ルールは P に出現してはならない．
2. $fn(P) \supseteq fn(Q)$.

二つ目の条件は，直感的には，右辺で新しく出現するハイパーリンクは ν (new) で宣言してやる必要があると読める．

$$\mathbf{0}\langle Y/X\rangle \stackrel{\text{def}}{=} \mathbf{0}$$

$$p(X_1,\ldots,X_m)\langle Z/Y\rangle \stackrel{\text{def}}{=} p(X_1\langle Z/Y\rangle,\ldots,X_m\langle Z/Y\rangle)$$

ただし $X_i\langle Z/Y\rangle \stackrel{\text{def}}{=} \begin{cases} Z & \text{if } X_i = Y \\ X_i & \text{if } X_i \neq Y \end{cases}$

$$(P,Q)\langle Y/X\rangle \stackrel{\text{def}}{=} (P\langle Y/X\rangle, Q\langle Y/X\rangle)$$

$$(\nu X.P)\langle Z/Y\rangle \stackrel{\text{def}}{=} \begin{cases} \nu X.P & \text{if } X = Y \\ \nu X.P\langle Z/Y\rangle & \text{if } X \neq Y \wedge X \neq Z \\ \nu W.(P\langle W/X\rangle)\langle Z/Y\rangle & \text{if } X \neq Y \wedge X = Z \\ & \wedge W \notin fn(P) \wedge W \neq Z \end{cases}$$

$$(P :\!- Q)\langle Y/X\rangle \stackrel{\text{def}}{=} (P :\!- Q)$$

図 3　Flat HyperLMNtal のハイパーリンク代入

4.2　Flat HyperLMNtal の操作的意味論

まず，プロセス間の構造合同規則 ($\equiv$) を定義し，それを用いて遷移関係 ($\longrightarrow$) を定義する．

4.3　構造合同規則

$\equiv$ を図 4 の関係を満たすプロセス上の最小の同値関係として定義する．ただし， $P\langle Y/X\rangle$ はハイパーリンク代入であり，これは図 3 に示されるように，全ての自由出現する X を Y で置き換えるものである．ここで， X が Y に置き換えられた時に自由出現しない場所に出現しているときは，事前に α 変換が必要になる可能性があることに注意する．また， 以降，Flat HyperLMNtal の意味論の記述においては， $=$ をプロセスや構文要素が**構文的に等しい**という意味で用いる．

(E1)–(E4) は LMNtal と全く同様である．(E5) は ν を構造合同関係にするための規則である

(E6)，(E7) は fusion のための規則で，これが本研究における核心部分である．(E6) の左辺において，$\nu X.$ が必要な理由はすでに第 3 節において説明した通りである．この右辺において $\nu X.$ が必要なのは，X と Y がどちらも同じ名前のハイパーリンクであったときのためで，これがないと，例えば

$$\nu X.(X \bowtie X, a(X)) \equiv a(X)$$

となり，この例では，左辺では自由ハイパーリンクは出現しないが，右辺では X が自由ハイパーリンクとなり，$\equiv$ の両辺で自由ハイパーリンク（自由変数）の集合が変化してしまう．これは構文駆動な計算モデルにおいては望ましくない性質であるため，$\nu X.$ を規則の右辺にも補っている．

同様に，付帯条件「ただし $X \in fn(P) \vee Y \in fn(P)$」も構造合同なプロセス間で自由ハイパーリンクの集合が等しくなるようにするためである．この付帯条件がないと，例えば，

$$\nu X.(X \bowtie Y, a) \equiv a$$

が導出でき，この例では左辺では自由ハイパーリンク Y が出現するが，右辺では自由ハイパーリンクは出現せず，$\equiv$ の両辺で自由ハイパーリンクの集合が変化してしまう．

こういったコーナーケースがいくつか存在したため，$\equiv$ の関係にあるプロセス間において，自由ハイパーリンクの集合が等しいということは，定理 3 において証明した．

ただし，この付帯条件のために，ハイパーリンクが $\bowtie$ の両辺にしか出現しない場合に，(E6) のみでは fusion を吸収できない．そこで，このケースに対応するためにさらに (E7) を用意した．ここで，(E7)

(E1)	$(\mathbf{0}, P)$	$\equiv$	P
(E2)	(P, Q)	$\equiv$	(Q, P)
(E3)	$(P, (Q, R))$	$\equiv$	$((P, Q), R)$
(E4)	$P \equiv Q$	$\Rightarrow$	$(P, R) \equiv (Q, R)$
(E5)	$P \equiv Q$	$\Rightarrow$	$\nu X.P \equiv \nu X.Q$
(E6)	$\nu X.(X \bowtie Y, P)$	$\equiv$	$\nu X.P\langle Y/X\rangle$
	（ただし $X \in fn(P) \vee Y \in fn(P)$）		
(E7)	$\nu X.\nu Y.X \bowtie Y$	$\equiv$	$\mathbf{0}$
(E8)	$\nu X.\mathbf{0}$	$\equiv$	$\mathbf{0}$
(E9)	$\nu X.\nu Y.P$	$\equiv$	$\nu Y.\nu X.P$
(E10)	$\nu X.(P, Q)$	$\equiv$	$(\nu X.P, Q)$
	（ただし $X \notin fn(Q)$）		

図 4　Flat HyperLMNtal プロセスの構造合同

(R1)	$\dfrac{P \longrightarrow P'}{(P,Q) \longrightarrow (P',Q)}$
(R2)	$\dfrac{P \longrightarrow P'}{\nu X.P \longrightarrow \nu X.P'}$
(R3)	$\dfrac{Q \equiv P \qquad P \longrightarrow P' \qquad P' \equiv Q'}{Q \longrightarrow Q'}$
(R4)	$(P,(P :\text{-}\ Q)) \longrightarrow (Q,(P :\text{-}\ Q))$

図 5　Flat HyperLMNtal プロセスの遷移関係

も X と Y が同じハイパーリンク名のときも適用可能であることに注意されたい．

(E8)–(E10) は ν のための規則で，これらは π 計算などの並行計算モデルにおいて普遍的に見られる規則である．

Flat HyperLMNtal では局所ハイパーリンクの α 変換による合同関係は他の定義から導出可能であったのであえて定義せず，定理 1 としてのちに証明する．

4.4　遷移関係

プロセス間の遷移関係 $\longrightarrow$ を図 5 を満たす最小の関係として定義する．

計算モデル，プログラミング言語の意味論においては，関数の仮引数と実引数，ルールとグラフのハイパーリンクの対応などの，マッチングする際の変数の対応がきちんと定義できていることが重要である．Flat HyperLMNtal の場合は局所ハイパーリンクは α 変換可能であることを後で示すものの，自由ハイパーリンクは α 変換できないため，これが正しくマッチングできるように定義されているかが鍵となってくる．

では，ルール

$$(p(X,Y) :\text{-}\ q(X,Y))$$

がアトム $p(X,X)$ を書き換えることは可能であろうか．より正確には，

$$(p(X,X),(p(X,Y) :\text{-}\ q(X,Y)))$$

は遷移することが可能か．ルールは

$$(p(X,X) :\text{-}\ \ldots)$$

のような形に α 変換することはできない．しかし，アトム $p(X,X)$ は (E7) とのちに証明する補題 1 を用いて $\nu Y.(Y \bowtie X, p(X,Y))$ の形に変換できる．

従って，これは以下のように遷移が可能である

$$\begin{aligned}
&(p(X,X),(p(X,Y) :\text{-}\ q(X,Y))) \\
&\equiv_{\text{補題 1}} \nu Y.(p(X,X),(p(X,Y) :\text{-}\ q(X,Y))) \\
&\equiv_{\text{E6}} \nu Y.(Y \bowtie X,(p(X,Y),(p(X,Y) :\text{-}\ q(X,Y)))) \\
&\longrightarrow \nu Y.(Y \bowtie X,(q(X,Y),(p(X,Y) :\text{-}\ q(X,Y)))) \\
&\equiv_{\text{E6}} \nu Y.(q(X,X),(p(X,Y) :\text{-}\ q(X,Y))) \\
&\equiv_{\text{補題 1}} (q(X,X),(p(X,Y) :\text{-}\ q(X,Y)))
\end{aligned}$$

このように，fusion に関する構造合同関係を援用することによって，自由ハイパーリンクのマッチング，特に非単射的マッチングも可能である．

5　Flat HyperLMNtal の性質

λ 計算や π 計算は，α 変換が（構造）合同性を保つことや，自由変数の集合が（構造）合同なプロセス間で変化しないこと，自由変数の集合が遷移に伴い広義単調に減少することといった性質を満たす †1．これらは構文駆動な計算モデルにおいて普遍的な性質だと言える．そこで，この節では，これらの性質が Flat HyperLMNtal においても成り立つことを証明する．

5.1　構造合同規則に関する性質の証明

提案した Flat HyperLMNtal の構造合同規則では LMNtal における構造合同規則と比較して

- (E4) $P \equiv P[Y/X]$
  (ただし X は P の自由リンク)
- (E8) $X = Y \equiv Y = X$

に対応するものが欠けている．

これはこれらの規則が他の規則から導出できること，すなわち admissible であることがわかったためである．これらを定理 1 と定理 2 に示す．また，それらの定理を用いて自由ハイパーリンクの集合が合同なプロセス間で変化しないことを定理 3 に示す．

補題 1 (自由出現しないハイパーリンクを束縛する ν の除去)**.**

$$\nu X.P \equiv P \text{ where } X \notin fn(P)$$

†1　ただし，π 計算では，構造合同規則 Sc-Mat においてのみ，構造合同なプロセス間で自由変数の集合が変化する可能性がある [14]

Proof.

$$
\begin{array}{lll}
 & \nu X.P & \\
\equiv_{\mathrm{E5}} & \nu X.(\mathbf{0}, P) & \because P \equiv_{\mathrm{E1}} (\mathbf{0}, P) \\
\equiv_{\mathrm{E10}} & (\nu X.\mathbf{0}, P) & \because X \notin \mathit{fn}(P) \\
\equiv_{\mathrm{E4}} & (\mathbf{0}, P) & \because \nu X.\mathbf{0} \equiv_{\mathrm{E8}} \mathbf{0} \\
\equiv_{\mathrm{E1}} & P &
\end{array}
$$

□

補題 2 (同じハイパーリンクを代入しても構文的に等しい)**.**

$$P\langle X/X\rangle = P$$

これは見かけほどは自明ではない．なぜなら，ハイパーリンク代入を行う際に α 変換をされてしまい，合同ではあるが構文的には異なるプロセスになってしまう可能性が素朴には否定できないからである．

Proof. プロセスにおいて，構造帰納法を用いて証明する．ここで， $\mathbf{0}$， $p(X_1, \ldots, X_m)$，(P, Q)，$(P \mathrel{:-} Q)$ の場合は帰納法の仮定より明らかである．

Case $\nu Y.P$:

$$(\nu Y.P)\langle X/X\rangle \stackrel{\mathrm{def}}{=} \begin{cases} \nu Y.P & \text{if } Y = X \\ \nu Y.P\langle X/X\rangle & \text{if } Y \neq X \\ = \nu Y.P \because \text{帰納法の仮定} & \end{cases}$$

$Y \neq X \wedge Y = X$ は絶対に起こり得ないため, 変数捕捉の回避のために α 変換（これにより構文的な等しさが失われる可能性があった）を行う可能性はない．

□

定理 1 (α 同値)**.**

$$\nu X.P \equiv \nu Y.P\langle Y/X\rangle \text{ where } Y \notin \mathit{fn}(P)$$

Proof. この証明には補題 1 と補題 2 を用いる．

代入される自由ハイパーリンクが出現している場合としていない場合に分けて証明を行う．後者は自明に思えるが，ハイパーリンク代入による α 変換が起きる可能性があるため，これは自明ではない．前者を先に証明し，前者を用いることのできる形に変形してやることで証明を行う．

Case $X \in \mathit{fn}(P)$:

$$
\begin{array}{l}
\nu X.\nu Y.(Y \bowtie X, (X \bowtie Y, P)) \\
\equiv_{\mathrm{E5,\ E3}} \nu X.\nu Y.((Y \bowtie X, X \bowtie Y), P) \\
\equiv_{\mathrm{E5,\ E10}} \nu X.(\nu Y.(Y \bowtie X, X \bowtie Y), P) \\
\qquad \because Y \notin \mathit{fn}(P) \\
\equiv_{\mathrm{E5,\ E6}} \nu X.(\nu Y.X \bowtie X, P) \\
\qquad \because (X \bowtie Y)\langle X/Y\rangle = X \bowtie X \\
\equiv_{\mathrm{E5,\ E10}} \nu X.\nu Y.(X \bowtie X, P) \\
\qquad \because Y \notin \mathit{fn}(P) \\
\equiv_{\mathrm{E9}} \nu Y.\nu X.(X \bowtie X, P) \\
\equiv_{\mathrm{E6,\ 補題\ 2}} \nu Y.\nu X.P \\
\qquad \because P\langle X/X\rangle = P \\
\equiv_{\mathrm{補題\ 1}} \nu X.P \\
\qquad \because Y \notin \mathit{fn}(P)
\end{array}
$$

であり，また

$$
\begin{array}{l}
\nu X.\nu Y.(Y \bowtie X, (X \bowtie Y, P)) \\
\equiv_{\mathrm{E2,\ E3,\ E5,\ E9}} \nu Y.\nu X.(X \bowtie Y, (Y \bowtie X, P)) \\
\equiv_{\mathrm{E5,\ E6}} \nu Y.\nu X.(Y \bowtie Y, P\langle Y/X\rangle) \\
\equiv_{\mathrm{E5,\ 補題\ 1}} \nu Y.(Y \bowtie Y, P\langle Y/X\rangle) \\
\qquad \because X \notin \mathit{fn}((Y \bowtie Y, P\langle Y/X\rangle)) \\
\equiv_{\mathrm{E6}} \nu Y.P\langle Y/X\rangle \\
\qquad \because \text{補題 2 より } (P\langle Y/X\rangle)\langle Y/Y\rangle = P\langle Y/X\rangle
\end{array}
$$

であるため，$\nu X.P \equiv \nu Y.P\langle Y/X\rangle$

Case $X \notin \mathit{fn}(P)$:

この場合においては，(E7) を用いて，まず自由ハイパーリンク X を付加させることで，先の証明を活用する．

$$
\begin{array}{l}
\nu X.P \\
\equiv_{\mathrm{補題\ 1}} P \\
\equiv_{\mathrm{E1}} (\mathbf{0}, P) \\
\equiv_{\mathrm{E4,\ E7}} (\nu X.\nu X.X \bowtie X, P) \\
\equiv_{\mathrm{E4,\ 補題\ 1}} (\nu X.X \bowtie X, P) \\
\qquad \because X \notin \mathit{fn}(\nu X.X \bowtie X) \\
\equiv_{\mathrm{E10}} \nu X.(X \bowtie X, P) \\
\qquad \because X \notin \mathit{fn}(P) \\
\equiv_{\mathrm{先の証明}} \nu Y.(X \bowtie X, P)\langle Y/X\rangle \\
\qquad \because X \in \mathit{fn}((X \bowtie X, P)) \\
= \nu Y.(Y \bowtie Y, P\langle Y/X\rangle)
\end{array}
$$

$\equiv_{\text{E10}} (\nu Y.Y \bowtie Y, P\langle Y/X\rangle)$

$\because X \notin fn(P)$ なので $Y \notin fn(P\langle Y/X\rangle)$

$\equiv_{\text{E4, 補題 1}} (\nu Y.\nu Y.Y \bowtie Y, P\langle Y/X\rangle)$

$\equiv_{\text{E7}} (\mathbf{0}, P\langle Y/X\rangle)$

$\equiv_{\text{E1}} P\langle Y/X\rangle$

$\equiv_{\text{補題 1}} \nu Y.P\langle Y/X\rangle$

□

ここで，定理 1 の証明に (E1)–(E10) の全ての構造合同規則を用いていることは注目に値する．定理 1 やこれから述べる定理 2 が構造合同規則から導出できることにより，LMNtal における規則（グラフ書き換え系に必要な規則）が全て導出できること，つまり，定義した構造合同規則が十分であることはわかる．が，これらが必要であること，つまり admissible な規則が存在しないことを証明するのは難しい．実際に，LMNtal においては構造合同規則 (E8) が admissible であることは，本研究で初めて示された（以下に述べる定理 2 とほぼ同じように証明できる）．しかし，定理 1 の導出において定義した全ての構造合同規則を用いたことは，提案する意味論における構造合同規則がどれも必要であるということの直感的な根拠を与えるものである．

定理 2 ($\bowtie$ の対称性)**.**

$$X \bowtie Y \equiv Y \bowtie X$$

Proof.

$\nu Z.(Z \bowtie X, Z \bowtie Y)$

$\equiv_{\text{E6}} \nu Z.(X \bowtie Y)$

$\because (Z \bowtie Y)\langle X/Z\rangle = X \bowtie Y$

$\equiv_{\text{補題 1}} X \bowtie Y$

であり，また

$\nu Z.(Z \bowtie X, Z \bowtie Y)$

$\equiv_{\text{E2, E5}} \nu Z.(Z \bowtie Y, Z \bowtie X)$

$\equiv_{\text{E6}} \nu Z.(Y \bowtie X)$

$\because (Z \bowtie X)\langle Y/Z\rangle = Y \bowtie X$

$\equiv_{\text{補題 1}} Y \bowtie X$

従って，$X \bowtie Y \equiv Y \bowtie X$.

□

定理 3 (構造合同なプロセスにおける自由ハイパーリンクの集合の等価性)**.**

$$fn(P) = fn(Q) \text{ if } P \equiv Q$$

Proof. 構造合同規則とプロセスにおいて構造帰納法を用いて証明を行う．

Case $(\mathbf{0}, P) \equiv P$:

$fn((\mathbf{0}, P)) = fn(\mathbf{0}) \cup fn(P) = fn(P)$

Case $(P, Q) \equiv (Q, P)$:

$fn((P, Q)) = fn(P) \cup fn(Q) = fn(Q) \cup fn(P) = fn((Q, P))$

Case $(P, (Q, R)) \equiv ((P, Q), R)$:

$fn((P, (Q, R))) = fn(P) \cup fn(Q) \cup fn(R) = fn(((P, Q), R))$

Case $P \equiv P' \Rightarrow (P, Q) \equiv (P', Q)$:

帰納法の仮定より $fn(P) = fn(P')$ if $P \equiv P'$ であるので

$fn((P, Q)) = fn(P) \cup fn(Q) = fn(P') \cup fn(Q) = fn((P', Q))$

Case $P \equiv P' \Rightarrow \nu X.P \equiv \nu X.P'$:

帰納法の仮定より $fn(P) = fn(P')$ if $P \equiv P'$ であるので

$fn(\nu X.P) = fn(P) \setminus \{X\} = fn(P') \setminus \{X\} = fn(\nu X.P')$

Case $\nu X.(X \bowtie Y, P) \equiv \nu X.P\langle Y/X\rangle$ （ただし $X \in fn(P) \vee Y \in fn(P)$） :

付帯条件より, $fn(P\langle Y/X\rangle)$ は $(fn(P) \setminus \{X\}) \cup \{Y\}$ と等しい.

従って,

$fn(\nu X.(X \bowtie Y, P))$

$= (fn(P) \cup \{Y\}) \setminus \{X\}$

$= ((fn(P) \setminus \{X\}) \cup \{Y\}) \setminus \{X\}$

$= fn(P\langle Y/X\rangle) \setminus \{X\} = fn(\nu X.P\langle Y/X\rangle)$

Case $\nu X.\nu Y.X \bowtie Y \equiv \mathbf{0}$:

$fn(\nu X.\nu Y.X \bowtie Y)$

$= fn(X \bowtie Y) \setminus \{X, Y\}$

$= \{X, Y\} \setminus \{X, Y\} = \emptyset = fn(\mathbf{0})$

Case $\nu X.\mathbf{0} \equiv \mathbf{0}$:

$fn(\nu X.\mathbf{0}) = \emptyset \setminus \{X\} = \emptyset = fn(\mathbf{0})$

Case $\nu X.\nu Y.P \equiv \nu Y.\nu X.P$:

$fn(\nu X.\nu Y.P) = (fn(P) \setminus \{Y\}) \setminus \{X\} = (fn(P) \setminus \{X\}) \setminus \{Y\} = fn(\nu Y.\nu X.P)$

Case $\nu X.(P,Q) \equiv (\nu X.P,Q)$（ただし $X \notin fn(Q)$）:

$$\begin{aligned} & fn(\nu X.(P,Q)) \\ &= (fn(P) \cup fn(Q)) \setminus \{X\} \\ &= (fn(P) \setminus \{X\}) \cup fn(Q) \quad \because X \notin fn(Q) \\ &= fn((\nu X.P,Q)) \end{aligned}$$

□

5.2 遷移規則に関する性質の証明

提案する意味論において，プロセスの自由ハイパーリンクは遷移に伴い，単調に減少することを 定理 4 で示す．つまり，これは新たな自由ハイパーリンクが遷移に伴い出現することがないことを保証するものである．

定理 4 (プロセスの遷移に伴う自由ハイパーリンクの広義単調減少性)**.**

$$fn(P) \supseteq fn(Q) \text{ if } P \longrightarrow Q$$

Proof. プロセスと遷移関係において構造帰納法を用いて証明する．

Case $\dfrac{P \longrightarrow P'}{(P,Q) \longrightarrow (P',Q)}$:

帰納法の仮定より $fn(P) \supseteq fn(P')$ if $P \longrightarrow P'$ であるので

$fn((P,Q)) = fn(P) \cup fn(Q) \supseteq fn(P') \cup fn(Q) = fn((P',Q))$

Case $\dfrac{P \longrightarrow P'}{\nu X.P \longrightarrow \nu X.P'}$:

帰納法の仮定より $fn(P) \supseteq fn(P')$ if $P \longrightarrow P'$ であるので

$fn(\nu X.P) = fn(P) \setminus \{X\} \supseteq fn(P') \setminus \{X\} = fn(\nu X.P')$

Case $\dfrac{Q \equiv P \qquad P \longrightarrow P' \qquad P' \equiv Q'}{Q \longrightarrow Q'}$:

帰納法の仮定より

$fn(P) \supseteq fn(P')$ if $P \longrightarrow P'$ であり，また定理 3 から $fn(Q) = fn(P)$ if $Q \equiv P$ かつ $fn(P') = fn(Q')$ if $P' \equiv Q'$ であるので

$fn(Q) = fn(P) \supseteq fn(P') = fn(Q')$

Case $(P,(P \text{ :- } Q)) \longrightarrow (Q,(P \text{ :- } Q))$:

ルールの二つ目の構文条件により $fn(P) \supseteq fn(Q)$ であるので

$fn((P,(P \text{ :- } Q))) = fn(P) \supseteq fn(Q) = fn((Q,(P \text{ :- } Q)))$

□

6 HyperLMNtal の構文・意味論

この節では構文駆動な意味論を持つ階層グラフ書き換え言語 LMNtal へハイパーリンクを形式的に組み込む．Flat LMNtal は非常にミニマルな計算モデルであったが，LMNtal は「実用プログラミング言語」としての側面が強く [18],（ほとんどの他の実用プログラミング言語よりは遥かに簡明であるものの）表現力の高さと引き換えに意味論は相応に複雑になっている．

LMNtal には，Flat LMNtal の諸機能に加えて，「膜」と呼ばれるプロセスの階層化の機能が存在する．膜を貫く自由リンクに関する意味論は相応に複雑であり，また膜の中の書き換え規則の集合にマッチするルール文脈や膜の中の残りのグラフにマッチするプロセス文脈などの複雑なパターンマッチのための構文や意味論が付随する．

さらに，現実装の HyperLMNtal は 通常の LMNtal に特殊な単引数のアトムとしてハイパーリンクを実装したものであり，これは通常のリンクの直接的な拡張ではない．この特殊なアトムはハイパーリンクの ID を持ち，これを用いてマッチングやグラフ同型性判定 [7] などを行う．これは処理系からすると単なるアトムとも取れるため，通常のリンクと接続することも可能であるし，通常の自由リンクで，通常のリンクの先でハイパーリンクと接続されたグラフをマッチングすることも可能である．これを定式化するためには，ハイパーリンクをリンクの拡張ではなく，アトムと同じような構文要素として定義する必要がある．

本節ではそのように複雑な SOS を持つプログラミング言語においても，提案する意味論が容易に組み込めることを確認する．

6.1 HyperLMNtal の抽象構文

HyperLMNtal の構文要素を三つで定める：

- 通常のリンク名 X.
- ハイパーリンク名 H. ただし，ハイパーリンクは通常のリンクと区別するために基本的に！をつけて表記される.
- アトム名 p. アトム名の予約名は $=$ と $\bowtie$ である.

抽象構文を図 6 で与える. ここで，LMNtal に新しく追加された構文要素を † で示す. また，LMNtal と同様の略記法を扱えるものとする. 詳細は文献[16] を参照されたい. 例えば，省略しても曖昧さの残らない括弧は省略可能である. また，$!H_1(X), X \bowtie Y, !H_2(Y)$ は $!H_1 \bowtie !H_2$ のように略記できる.

HyperLMNtal のプロセスに出現する自由ハイパーリンクの集合を図 7 のように定義する. また，プロセステンプレートにおける自由ハイパーリンクも同様に定義する. ここで，通常のリンクとハイパーリンクの名前空間は別であることに注意されたい. 例えば「X が P の自由リンクである」といったときは，プロセス P において通常のリンクがただ一回のみ出現したことを表し，そのプロセスにおけるハイパーリンクの出現に関しては全く言及していない.

また，LMNtal と Flat HyperLMNtal における構文条件をどちらとも満たさなくてはいけない. 例えば，通常のリンクの出現数は高々2 回のみでなくてはならない. また，ルール T :- U は $fn(T) \supseteq fn(U)$ を満たさなくてはならない.

6.2 HyperLMNtal の構造合同規則

図 9 は HyperLMNtal のプロセスにおける構造合同規則を示す. LMNtal から新しく追加された規則は † で示した. ここで，$P[Y/X]$ は通常のリンクの代入を示し，通常のリンク X を通常のリンク Y で置き換えるものである. 対して，$P\langle H_2/H_1\rangle$ は 表 8 に定義されるようなハイパーリンク代入である. Flat HyperLMNtal において既に議論されたように事前の α 変換が要求される可能性があることに注意する.

図 10 は HyperLMNtal の遷移関係を示す. 新しく追加された遷移規則は † で示す. 規則 (R7) における θ はルールのマッチングにおける置換を表し，プロセス文脈やルール文脈などのマッチングに利用される. この詳細は文献[16] を参照されたい. ただし，ここで，プロセス文脈はハイパーリンクにマッチ可能であるが，hyperlink creation を含むプロセスにはマッチ可能でないものと定義する. これによって，プロセス文脈を用いたプロセスのコピーにおいてハイパーリンクは新しく生成されることなく，元のものから共有されることになる.

プロセス	
$P ::= \mathbf{0}$	Null
$\mid\ p(X_1, \ldots, X_m)$	Atom, $m \geq 0$
$\mid\ (P, P)$	Molecule
$\mid\ \{P\}$	Cell
$\mid\ \nu H.P$	Hyperlink creation †
$\mid\ !H(X)$	Hyperlink †
$\mid\ (T \text{ :- } T)$	Rule
プロセステンプレート	
$T ::= \mathbf{0}$	Null
$\mid\ p(X_1, \ldots, X_m)$	Atom, $m \geq 0$
$\mid\ (T, T)$	Molecule
$\mid\ \{T\}$	Cell
$\mid\ \nu H.T$	Hyperlink creation †
$\mid\ !H(X)$	Hyperlink †
$\mid\ (T \text{ :- } T)$	Rule
$\mid\ @p$	Rule context
$\mid\ \$p\,[X_1, \ldots, X_m \mid A]$	Process context, $m \geq 0$
リンク束	
$A ::= []$	Empty
$\mid\ *X$	Bundle

図 6 HyperLMNtal の構文

7 まとめ

本論文では，プロセス代数における隠蔽の構文・意味論をグラフ書き換えに基づく計算モデル Flat LMNtal に組み込むことで，ハイパーグラフ書き換え言語 Flat HyperLMNtal の構文・意味論を定義し

$$
\begin{aligned}
fn(\mathbf{0}) &\stackrel{\text{def}}{=} \emptyset \\
fn(p(X_1,\ldots,X_m)) &\stackrel{\text{def}}{=} \emptyset \\
fn((P,Q)) &\stackrel{\text{def}}{=} fn(P) \cup fn(Q) \\
fn(\{P\}) &\stackrel{\text{def}}{=} fn(P) \\
fn(\nu H.P) &\stackrel{\text{def}}{=} fn(P) \setminus \{H\} \\
fn(!H(X)) &\stackrel{\text{def}}{=} \{H\} \\
fn((P :\text{-}\ Q)) &\stackrel{\text{def}}{=} \emptyset
\end{aligned}
$$

図 7　HyperLMNtal プロセスの自由ハイパーリンクの集合

$$
\mathbf{0}\langle H_2/H_1\rangle \stackrel{\text{def}}{=} \mathbf{0}
$$

$$
p(X_1,\ldots,X_m)\langle H_2/H_1\rangle \stackrel{\text{def}}{=} p(X_1,\ldots,X_m)
$$

$$
(P,Q)\langle H_2/H_1\rangle \stackrel{\text{def}}{=} (P\langle H_2/H_1\rangle, Q\langle H_2/H_1\rangle)
$$

$$
(\nu H.P)\langle H_2/H_1\rangle
\stackrel{\text{def}}{=} \begin{cases}
\nu H.P & \text{if } H = H_1 \\
\nu H.P\langle H_2/H_1\rangle & \\
\qquad\qquad \text{if } H \neq H_1 \wedge H \neq H_2 & \\
\nu H_3.(P\langle H_3/H\rangle)\langle H_2/H_1\rangle & \\
\qquad \text{if } H \neq H_1 \wedge H = H_2 & \\
\qquad\quad \wedge H_3 \notin fn(P) \wedge H_3 \neq H_2 &
\end{cases}
$$

$$
!H(X)\langle H_2/H_1\rangle \stackrel{\text{def}}{=} \begin{cases}
!H_2(X) & \text{if } H = H_1 \\
!H(X) & \text{if } H \neq H_1
\end{cases}
$$

$$
(P :\text{-}\ Q)\langle H_2/H_1\rangle \stackrel{\text{def}}{=} (P :\text{-}\ Q)
$$

図 8　HyperLMNtal のハイパーリンク代入

(E1)	$\mathbf{0}, P \equiv P$
(E2)	$P, Q \equiv Q, P$
(E3)	$P, (Q, R) \equiv (P, Q), R$
(E4)	$P \equiv P[Y/X]$
	（ただし X は P の局所リンク）
(E5)	$P \equiv P' \Rightarrow P, Q \equiv P', Q$
(E6)	$P \equiv P' \Rightarrow \{P\} \equiv \{P'\}$
(E7) †	$P \equiv Q \Rightarrow \nu H.P \equiv \nu H.Q$
(E8)	$X = X \equiv \mathbf{0}$
(E9)	$X = Y, P \equiv P[Y/X]$
	（ただし P はアトム，X は P の自由リンク）
(E10)	$\{X = Y, P\} \equiv X = Y, \{P\}$
	（ただし X または Y のどちらか一方が P の自由リンク）
(E11) †	$\nu H_1.(!H_1 \bowtie !H_2, P) \equiv \nu H_1.P\langle H_2/H_1\rangle$
	（ただし $H_1 \in fn(P) \vee H_2 \in fn(P)$）
(E12) †	$\nu H_1.\nu H_2.(!H_1 \bowtie !H_2) \equiv \mathbf{0}$
(E13) †	$\nu H.\mathbf{0} \equiv \mathbf{0}$
(E14) †	$\nu H_1.\nu H_2.P \equiv \nu H_2.\nu H_1.P$
(E15) †	$\nu H.\{P\} \equiv \{\nu H.P\}$
(E16) †	$\nu H.(P, Q) \equiv (\nu H.P, Q)$
	（ただし $H \notin fn(Q)$）

図 9　HyperLMNtal プロセスの構造合同

た．これは構文から直接的に操作的意味論を定義できており，圏論などの深い数学的素養を要求せず，一般のプログラマにとってもシンプルでわかりやすい定義となっていると主張する．また，これが他の構文駆動な計算モデルに普遍的に見られる性質を持つことを証明し，SOS を持つより複雑なプログラミング言語に容易に組み込むことが可能であることを示した．

グラフ書き換え言語は現状，研究者にとっても，広く受け入れられているプログラミングパラダイムとは言いがたい．これには実装上の効率などの要因もあると思われるが，一番の原因はそのコア部分に広く普及している計算モデルの意味論（SOS）が適用できていなかったことだと考える．しかし，このようにハイパーリンクの融合も第一級に扱う強力なハイパーグラフ書き換え言語として，ミニマムな SOS を定義したことによって，ハイパーグラフ書き換え系の普及及び，SOS を持つ他の計算モデルやプログラミング言語への組み込みが容易になると主張する．

謝辞　本研究の一部は，科学研究費基盤研究 (B) 18H03223 の助成を受けて実施した．

参 考 文 献

[1] Bak, C.: *GP 2: Efficient Implementation of a Graph Programming Language*, PhD Thesis, Uni-

(R1)	$\dfrac{P \longrightarrow P'}{P, Q \longrightarrow P', Q}$
(R2)	$\dfrac{P \longrightarrow P'}{\{P\} \longrightarrow \{P'\}}$
(R3) †	$\dfrac{P \longrightarrow P'}{\nu H.P \longrightarrow \nu H.P'}$
(R4)	$\dfrac{Q \equiv P \quad P \longrightarrow P' \quad P' \equiv Q'}{Q \longrightarrow Q'}$
(R5)	$\{X = Y, P\} \longrightarrow X = Y, \{P\}$ （ただし X と Y は $\{X = Y, P\}$ の自由リンク）
(R6)	$X = Y, \{P\} \longrightarrow \{X = Y, P\}$ （ただし X と Y は P の自由リンク）
(R7)	$T\theta, (T :- U) \longrightarrow U\theta, (T :- U)$

図 10　HyperLMNtal プロセスの遷移関係

versity of York, September 2015.

[2] Banach, R.: DACTL Rewriting is Categorical, Computer Science Department, Manchester University, 1991.

[3] Claude, B.: *Hypergraphs: Combinatorics of Finite Sets*, Elsevier Science Publishers B.V., USA, 1989.

[4] Ehrig, H., Ehrig, K., Prange, U., and Taentzer, G.: *Fundamentals of Algebraic Graph Transformation (Monographs in Theoretical Computer Science. An EATCS Series)*, Springer, 2006.

[5] Gadducci, F.: Graph rewriting for the π-calculus, *Mathematical Structures in Computer Science*, Vol. 17, No. 3(2007), pp. 407–437.

[6] Glauert, J., Kennaway, J., and Sleep, M.: Dactl - an Experimental Graph Rewriting Language, *Proc. Graph Grammars 1990*, LNCS, Vol. 532, Springer, 1991, pp. 378–395.

[7] Gocho, M., Hori, T., and Ueda, K.: Evolution of the LMNtal Runtime to a Parallel Model Checker, *Computer Software*, Vol. 28, No. 4(2011), pp. 4_137–4_157.

[8] Kieburtz, R. B.: The G-machine: A fast, graph-reduction evaluator, *Functional Programming Languages and Computer Architecture*, Springer Berlin Heidelberg, 1985, pp. 400–413.

[9] Milner, R., Parrow, J., and Walker, D.: A calculus of mobile processes, I, *Information and Computation*, Vol. 100, No. 1(1992), pp. 1–40.

[10] Parrow, J. and Victor, B.: The fusion calculus: expressiveness and symmetry in mobile processes, *Proc. LICS 1998*, 1998, pp. 176–185.

[11] Plotkin, G.: A Structural Approach to Operational Semantics, *J. Log. Algebr. Program.*, Vol. 60-61(2004), pp. 17–139.

[12] Raoult, J. C.: On graph rewritings, *Theoretical Computer Science*, Vol. 32, No. 1(1984), pp. 1–24.

[13] Rozenberg, G.: *Handbook of Graph Grammars and Computing by Graph Transformation*, World Scientific, 1997.

[14] Sangiorgi, D. and Walker, D.: *The Pi-Calculus: A Theory of Mobile Processes*, Cambridge University Press, 2001.

[15] Sano, J.: Implementing G-Machine in HyperLMNtal, Bachelor's thesis, Waseda University, 2021. https://arxiv.org/abs/2103.14698.

[16] Ueda, K.: LMNtal as a hierarchical logic programming language, *Theoretical Computer Science*, Vol. 410, No. 46(2009), pp. 4784–4800.

[17] Ueda, K. and Ogawa, S.: HyperLMNtal: An Extension of a Hierarchical Graph Rewriting Model, *KI - Künstliche Intelligenz*, Vol. 26, No. 1(2012), pp. 27–36.

[18] 乾敦行, 工藤晋太郎, 原耕司, 水野謙, 加藤紀夫, 上田和紀: 階層グラフ書換えモデルに基づく統合プログラミング言語 LMNtal, コンピュータ ソフトウェア, Vol. 25, No. 1(2008), pp. 124–150.