MELL のカット除去規則に基づく階層グラフ書換え言語の拡張

田久 健人　上田 和紀

階層グラフ書換えとは，グラフ構造に，階層表現やグループ化のための箱状の構造 (Box) を加えた構造と，その書換えにより計算を表現する手法で，高い表現力を有するモデリング・計算手法である．一方で，論理体系との対応が自明ではない．そこで，Box を持つ Multiplicative Exponential Linear Logic (MELL) の Proof Nets に着目した．二者間の対応関係の考察にあたり，階層グラフ書換え言語 LMNtal による MELL Proof Nets のカット除去規則のエンコードを試みたが，接続先を明示しない不特定多数のエッジを含む Box の複製や消去の表現が自明ではなかった．本研究では，LMNtal 上においてこれらの操作の仕様を定義し，処理系に実装した上で，これらの操作の階層グラフ書換えにおける有用性を論ずる．さらに，LMNtal の状態空間探索機能などを用いて，LMNtal の Proof Nets のワークベンチとしての有用性を示す．

1　はじめに

1.1　背景

接続構造と階層構造は，コンピュータサイエンスや日常の様々な事象のモデリングにおいて普遍的に現れる．例えば，企業の組織図や，クラスの継承関係，システム構成図やアーキテクチャ図等，様々な事象の図的表現において，しばしば各構成要素の接続関係を線で表現し，それとは別に構成要素の階層分けやグループ化に関して円形や四角形で囲む表現方法が用いられる．このような，接続構造をグラフで，階層構造を箱状の構造 (以降 Box と呼ぶ) で表現する，**階層グラフ**と呼ばれるデータ構造がある．上記の通り階層グラフは普遍的な構造であるため，これを形式的に扱う計算モデルがあるのが望ましい．

階層グラフの書換えを用いて計算を行う手法は階層グラフ書換えと呼ばれ，その体系はいくつか存在している．並行計算の文脈で登場した Chemical Abstract Machine (CHAM) [2]，Bigraphical Reactive System (BRS) [12]，階層グラフ書換え言語 LMNtal [20] や，pushout を用いたグラフ変換の定義に階層性を追加した Hierarchical Graph Transformation [6]，グラフィカルなポートグラフ書換えツール PORGY [16] 向けに階層表現を追加した Attributed Hierarchical Port Graphs (AHP) [7] などが提案されている．また，これらの体系の多くには、形式的な定義のほかに実装も存在している．中でも LMNtal は，コンパクトな構造的操作的意味論を持ち，オープンソースの処理系を提供している †1．

一般に，ある計算モデルの研究としての位置付けを明確にするために，他の計算モデルとの対応関係を示すことは重要である．階層グラフ書換えにおいても同様で，各階層グラフ書換え系がそれぞれいくつかの計算モデルとの関係を示している．CHAM では CCS，BRS では非同期 π 計算や CCS，CSP との関係が示されている．LMNtal では，ラムダ計算，π 計算，Ambient 計算などの計算モデルのエンコードが

* Enhancing a Hierarchical Graph Rewriting Language based on MELL Cut Elimination.

Kento Takyu, 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.

†1 https://github.com/lmntal

可能であることが示されている[25] ほか，G-Machine の記述[17] や，C プログラムへの変換[24] などの実用的なプログラミング言語との対応も示されている．

また，論理体系との対応関係の整備は，Curry-Howard 同型等で見られるように，仕様の明瞭化や，論理体系に基づく良い性質が得られるため重要である．しかしながら，階層グラフ書換え系と論理体系の対応は非自明である．文献[20] では，LMNtal プログラムの線形論理式による解釈を試みているが，膜に関する扱いは，膜を平坦化し，膜のない flat LMNtal に帰着させて解釈している．そのため，Box 構造の動作に対する直接的な論理学的解釈は依然として自明ではない．

階層グラフ書換えと関連があると考えられる論理体系として，Multiplicative Exponential Linear Logic (MELL) が挙げられる．線形論理では，論理式をグラフ構造で表現する Proof Nets という手法が存在し，中でも指数的演算子 (exponentials) に関する推論規則は，Proof Nets においては Promotion Box と呼ばれる Box 構造で表現される．また，Proof Nets におけるカット除去は，グラフの書換えによって表現され，中でも指数的演算子に関するカット除去規則は，Promotion Box の階層移動，複製，消去を行うものがある．Proof Nets から出発したグラフ書換え系としては，Interaction Nets [11] やその派生の Box を持つ体系[1]，Hypernet [14] が挙げられる．これらは関数型言語の計算モデルとしては有用であるが，表現可能なグラフ構造が限定されており，モデリング言語としては十分とはいえない．

本研究では，階層グラフ書換え系から出発し，階層グラフ書換え系と論理体系との対応を考察する．対象とする階層グラフ書換え系としては，構文や意味論が明瞭であり，多数の計算モデルとの対応が考察されていて，オープンソースの処理系が提供されている LMNtal を選択した．LMNtal における階層グラフの書換えと，MELL Proof Nets のカット除去の対応関係を考察するにあたり，LMNtal による MELL Proof Nets のエンコードを試みた[18]．一般に，階層グラフ書換えにおいて，不特定多数のエッジを含む Box の操作は自明ではない．LMNtal においては，不特定多数のエッジを扱う構文があるため，大多数の操作は直接的にエンコードできた．しかしながら，Box の複製，消去に関しては，ルール前後での Box の内部要素 (ノード，エッジ) の個数の変化が伴うため，要素の出現回数に関する構文上の制約によって接続関係の安全性を保証している LMNtal においては，直接的なエンコードが難しい．

1.2　目的

本研究では，LMNtal において，MELL Proof Nets のカット除去規則で現れる，不特定多数のエッジを含む Box の複製や消去機能を導入することで，LMNtal と MELL の対応関係を明瞭にする．これらの機能の仕様を形式的に記述し，処理系に実装したうえで，

1. これらの機能を用いていくつかの例題を記述し，階層グラフ書換えにおける，MELL Proof Nets に基づく Box 操作の有用性を論ずる．また，
2. これらの機能を用いて，MELL Proof Nets が直接的にエンコードできることを示す．これにより，LMNtal の MELL Proof Nets のワークベンチとしての有用性を示す．

階層グラフ書換え側だけでなく，MELL Proof Nets の研究においても有用な成果となることが期待される．

1.3　本論文の構成

2 節では，階層グラフ書換え言語 LMNtal について説明する．このとき，4 節および 5 節で用いる構文や機能に絞って説明する．3 節では，線形論理の一つである Multiplicative Exponential Linear Logic (MELL) と，Proof Nets について説明する．4 節では，LMNtal の拡張について述べる．また，例題の記述を通して，導入した機能の有用性を述べる．5 節では，拡張 LMNtal を用いた MELL のカット除去規則のエンコードについて述べる．最後に，6 節でまとめを述べる．

なお，本文において，青地の Box を用いた図は LMNtal グラフ，赤地の Box を用いた図は MELL Proof Nets を表す．

プロセス

P ::=	$\mathbf{0}$	(空)
|	$p(X_1, \ldots, X_m)$	(アトム)
|	P, P	(分子)
|	$\{P\}$	(膜)
|	$[\,RuleName\,@@\,]\ T :\text{-}\ T$	(ルール)

プロセステンプレート

T ::=	$\mathbf{0}$	(空)
|	$p(X_1, \ldots, X_m)$	(アトム)
|	T, T	(分子)
|	$\{T\}$	(膜)
|	$[\,RuleName\,@@\,]\ T :\text{-}\ T$	(ルール)
|	$@p$	(ルール文脈)
|	$\$p[X_1, \ldots, X_m \mid A]$	(プロセス文脈)
|	$p(*X_1, \ldots, *X_m)$	(アトム集団)

剰余引数

A ::=	[]	
|	$*X$	(リンク束)

図 1　LMNtal の構文

2　階層グラフ書換え言語 LMNtal

本節では，文献 [20] の内容をもとに階層グラフ書換え言語 LMNtal について簡単に説明する．なお，LMNtal には，ルール適用に関する操作的意味論や，プロセスに関する構造合同規則があるが，本稿ではこれらに関する説明は文献 [20] 等に委ねる．

2.1　構文

LMNtal の構文を図 1 に示す．言語の基本要素は，**アトム，リンク，膜，ルール**からなる．

2.1.1　アトム，リンク，膜

LMNtal における**アトム**，**リンク**は，それぞれグラフ理論における頂点 (ノード)，辺 (エッジ) に対応する．

p という名前で m 本のリンク $X_1, \ldots, X_m$ を持つアトムを $p(X_1, \ldots, X_m)$ と表記する．これらのリンクを，アトムの引数といい，引数には全順序がある．例えば，`a(X,Y)` は，2 価の `a` アトムで，第一引数は `X`，第二引数は `Y` である．このような，引数が固定個で全順序の付いている頂点からなるグラフは，ポートグラフ (port graph) [7] と呼ばれる．なお，英字の大文字から始まる名前はリンク名，そうでないものはアトム名として解釈される．同じリンク名の対によってエッジの接続を表す．このため，直感的にはプロセス中に同じリンク名はたかだか 2 回しか出現しない．LMNtal では，この制約を構文上の制約として次のように課している (**リンク条件**)：

> プロセスに同名のリンクは高々2 回出現できる．

このとき，2 回出現するリンクは両端がアトムと繋がっている**局所リンク**であり，1 回のみ出現するリンクは一端が無接続である (または，外部と接続している) **自由リンク**である．この制約により，リンクの不正な接続を排除している．

LMNtal においては，アトムの多重集合によって無向多重グラフを表現する．つまり，多重辺や自己ループの出現を許す．例えば，

```
a(X,X), a(Y,Z), a(Y,Z).
```

は，図 2 のような無向多重グラフを表す．このとき，図中の矢印は第一引数を指し，そこから矢印の向きに従って引数に順序がついていることを示す．なお，図 2 で行ったように，LMNtal のコードはグラフ構造として図示可能である． アトムやリンクに関する略記法はいくつか存在する．中でもよく用いられる記法として，リンクの左側に `+` をつけた，`+X` のような表記が挙げられる．これは 1 価アトム `'+'(X)` の前置演算子記法である．

LMNtal における**膜**は，一般的なグラフ理論には無い，グラフの階層構造やグルーピングを表現するための構造である．この構造は，膜計算 [2] 等に由来している．膜は，$\{P\}$ という形で表記される．膜は入れ

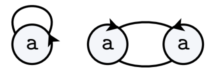

図 2　多重辺や自己ループを許す，LMNtal グラフの例

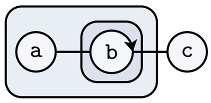

図 3　膜を貫通するリンクのある LMNtal グラフの例

子構造をなすことができ，リンクは膜の境界を越えてアトムをつなぐことができる．例えば，

```
{a(X), {b(Y,X)}}, c(Y).
```

は，図 3 のような構造を表す．このような，アトム，リンクからなる無向多重グラフに膜を加えた構造を，**LMNtal グラフ**と呼ぶ．

2.1.2　ルール

ルールは，部分グラフから部分グラフへの書換えを表す書換え規則である．例えば，`a(X) :- b(X)` は，1 個の `a` アトムを，1 個の `b` アトムに書き換えるルールである．便宜上，ルールの名前として *RuleName* をルールの前に付すこともある．

ルールの左辺と右辺には，部分グラフを記述する．そのため，しばしば自由リンクが現れる．ルール各辺における自由リンクは，接続先を明記しないものの，他の部分と接続している．このような自由リンクを含むグラフの変換ルールにおいて，ルール自体には自由リンクが存在しない，つまりルール上に出現するリンクは，そのルール上にちょうど 2 回出現しなければならないという制約が設けられている．つまり，左辺に自由リンクとして出現したリンクは，右辺でも同様に自由リンクとして出現する必要がある (この構文条件の詳細は [20](Occurrence Conditions) を参照)．直感的には，線形型によるポインタ制御と同様の役割を果たしている．この制約により，書換え時に不正な接続 (ダングリングポインタ) が生成されるのを防いでいる．

2.1.3　リンク束

リンク束は，`*X` のような形式で記述される，不特定多数の自由リンクとマッチするワイルドカード記法である．このとき，リンク束は，自由リンクの集合ではなく，順序のある列を表すことに留意する．また，記法 $|{*X}|$ によって，リンク束とマッチしたリンクの本数を表す．

2.1.4　プロセス文脈

プロセス文脈は，膜の中のルール以外のプロセスのうち，明示的に指定されていないもの全体とマッチする．

プロセス文脈は，$\$p[X_1,\ldots,X_m\,|\,A]$ のような形式で記述する．引数 $X_1,\ldots,X_m$ は，その文脈が持っていなければならない自由リンクであり，これを明示的な自由リンクと呼ぶ．A にはリンク束 $*X$ を記述することができ，これは，明示的な自由リンク以外の 0 本以上の自由リンクを持つことを表し，このような自由リンクを明示的でない不特定多数の自由リンクと呼ぶ．例えば，ルール：

```
o(B), {i(A,B),$p[A|*X]} :- n(A),$p[A|*X].
```

は，初期状態：

```
o(B), {i(A,B),a(A,G1),b(G2)},g(G1),g(G2).
```

に対して，図 4 のような書換えを行う．

また，リンク束やプロセス文脈に関しても，リンクと同様に構文上の出現回数の制約が設けられていて，不正な接続を防いでいる．図 4 の例では，`*X` は，ルール上にちょうど 2 回出現していて，これによりルール前後での `*X` リンクの接続が安全に保存されている．

以上のように，プロセス文脈は，一般的なプログラミング言語におけるワイルドカードのように振る舞うが，接続関係を指定できるという点が，階層グラフを扱う言語ならではの特徴である．

2.1.5　アトム集団

アトム集団 (aggregate) $p({*X_1},\ldots,{*X_m})$ $(m>0)$ は，リンク束 $*X_i$ が表すリンクの本数と同じ個数の m 価のアトムを表す．アトム集団に関して，リンクの接続関係を安全に保つためのいくかの制約が設け

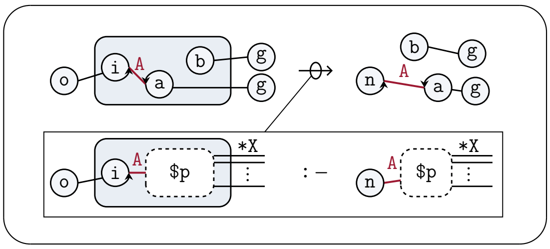

図 4　プロセス文脈，リンク束を含む書換えの例

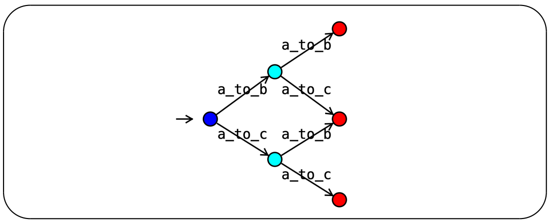

図 5　状態空間の可視化例

られている [20]．アトム集団によって，不特定多数個のアトムを表現することができるが，その複雑さから，現行の LMNtal 処理系上では代替の機能として 4.1 節で述べる `nlmem` ライブラリを提供している．

2.2　非決定性

LMNtal によるルールの適用は，評価戦略が定められていないため，非決定的である．例えば，

```
a_to_b@@ a :- b.
a_to_c@@ a :- c.
```

の，2 つのルールがプロセス中に存在し，これを `a,a.` という初期状態に適用すると，2 つある `a` アトムはそれぞれ `b`，`c` に書換えられる可能性があり，終了状態は，`b,b`，`b,c`，`c,c` の 3 通りが考えられる．

LMNtal 処理系 SLIM [22] では，このような非決定性を扱うための**非決定モード**が用意されている．非決定モードでは，全ての可能性を列挙し，状態空間を構築する．また，構築した状態空間は，可視化ツール StateViewer によって可視化することができる [21]．例えば，先の例で構築した状態空間は，図 5 のように可視化される．また，構築した状態空間に対して，LTL モデル検査を行うことができる [22]．

3　MELL と Proof Nets

3.1　Multiplicative Exponential Linear Logic

本研究では，Multiplicative Exponential Linear Logic (**MELL**) [8] を対象にする．MELL は，線形論理の最も基本的な (弱い) 断片である Multiplicative Linear Logic(MLL) に，指数的演算子 !，? を加えたものである．

定義 3.1 (MELL の論理式)．MELL の論理式 F は以下のように定義される：

$$\text{formula } F ::= X \mid F^{\perp} \mid F \otimes F \mid F \parr F \mid !F \mid ?F$$

ここで，X は原子論理式 (atomic formula) を表す．また，否定 $^{\perp}$ に関して以下のように挙動を定める：

$A^{\perp\perp} := A \quad (A \otimes B)^{\perp} := A^{\perp} \parr B^{\perp} \quad (A \parr B)^{\perp} := A^{\perp} \otimes B^{\perp}$
$(!A)^{\perp} := ?(A^{\perp}) \quad (?A)^{\perp} := !(A^{\perp}) \quad A \multimap B := A^{\perp} \parr B$

また，含意 → は，

$$A \to B \mapsto (!A \multimap B)$$

のように埋め込まれる．$\diamond$

図 6 に，MELL の推論規則 (one-sided) を示す．ここで，A，B は論理式，Γ，Δ は論理式の列である．構造規則のうち，Exchange 規則は認められているが，Weakening，Contraction は，指数的演算子 !，? を通してのみ定義されている．これにより，Γ，Δ は，!，? 内では集合のように振る舞うが，それ以外では多重集合のように振る舞う．乗法的連言 $\otimes$ と，一般的な (加法的な) 連言の違いは推論規則を見ると明らかで，一般的な連言は 1 つの文脈から導出されるのに対して，$\otimes$ は 2 つの文脈から導出される．Weakening，Contraction が認められている場合は，加法的連言と乗法的連言は同値であるが，これらの構造規則が認められていない場合は，異なる挙動を示す．具体的には，乗法的連言の場合は，それぞれの文脈から A，B のみを取り出す，すなわち “消費” している．このように，資源管理の観点から 2 つの連言は異なる意味を持つ．

MELL の証明体系は，カット除去定理が成り立つ

$$\frac{}{\vdash A, A^{\perp}}(ax) \qquad \frac{\vdash \Gamma, A \qquad \vdash \Delta, A^{\perp}}{\vdash \Gamma, \Delta}(cut)$$

$$\frac{\vdash \Gamma, A, B}{\vdash \Gamma, A \parr B}(\parr) \qquad \frac{\vdash \Gamma, A \qquad \vdash \Delta, B}{\vdash \Gamma, \Delta, A \otimes B}(\otimes)$$

$$\frac{\vdash ?\Gamma, A}{\vdash ?\Gamma, !A}(!) \qquad \frac{\vdash \Gamma, A}{\vdash \Gamma, ?A}(?d) \qquad \frac{\vdash \Gamma, A}{\vdash \Gamma, ?A}(?w) \qquad \frac{\vdash \Gamma, ?A, ?A}{\vdash \Gamma, ?A}(?c)$$

図 6　MELL の推論規則

ことが知られている [8]．図 7 に，MELL のカット除去規則の例を示す．なお，全てのカット除去規則は Appendix に添付する．このとき，推論規則中で二重線で示したルール適用は，そのルールを複数回必要なだけ適用することを意味する．多くの規則は直接カットを除去していないが，それらの規則は，カットを証明図の上部に移動させることにより正規化を助けている．

3.2　MELL Proof Nets

3.2.1　証明構造と整合性条件

次に，MELL のシークエントに対応する Proof Nets [8] を定義する．まず，**証明構造**を定義する．図 8 に，MELL の証明構造の構成要素を示す．各構成要素は，MELL の推論規則でラベル付けされたセル (ノード) と MELL 論理式でラベル付けされたワイヤー (エッジ) からなる．このとき，セル $\otimes$，$\parr$ のみ 2 つの入力が順序づけられている．

また，シークエント計算の (!) 規則を扱うための構造として，**Promotion Box** が導入される (図 9)．Box の内部を変換から保護し，簡約を順序付ける．Promotion Box は，必ず互いに分離しているか入れ子になっている必要がある．$?\Gamma$ は，不特定多数の ? 付き論理式の列を表す．

証明構造は，これらの要素を組み合わせることで構成される．

定義 3.2 (MELL Proof Structure)**.** MELL の証明構造は，図 8 に示すセル，ワイヤーと，図 9 に示す Promotion Box からなる，有向非巡回多重グラフである．　$\diamond$

このとき，ワイヤーは必ずしも閉じている必要はない．これは，自由リンクの存在を許すと言い換えることができる．

次に，証明構造と証明木の対応関係を定める．証明構造は，一般に対応する証明木を持つとは限らないが，整合性条件と呼ばれる条件を満たすクラスの証明構造に関しては，これに対して解釈されるようなシークエント計算の証明が存在する．この，整合性条件を満たす証明構造を **Proof Nets** と呼ぶ．

整合性条件としては，同値である複数の条件が知られているが，本研究においては，**Danon-Regnier の整合性条件** [5] を採用する．

定義 3.3 (スイッチング)**.** 証明構造 S に対する**スイッチング** σ とは，S 中の，図 10(a) に示すような，$\parr$，$?c$ の左右の選択である．**スイッチンググラフ** S_σ とは，S の全ての $\parr$，$?c$ セルを，σ に従って書換え，図 10(b) のように全ての Promotion Box の外枠を剥がしたものである．　$\diamond$

スイッチングは，$\parr$，$?c$ セルに対して，左右どちらかを選択し切断する操作で，非決定的な操作である．つまり，証明構造 S に対して，スイッチンググラフ S_σ は，S の $\parr$，$?c$ セルの個数を n として，2^n 個だけ存在する．

Proof Nets はスイッチングを用いて定義される．

定義 3.4 (Proof Nets)**.** 証明構造 S の任意のスイッチング S_σ が非巡回であるとき，S は **Proof Net** である．　$\diamond$

Danos-Regnier の整合性条件は，証明構造の幾何的な条件によって Proof Nets を定義する．

図 11 に，MELL Proof Net の例を示す (拡大版は

(a) promotion-nested

$$\dfrac{\dfrac{\vdash ?\Gamma, A}{\vdash ?\Gamma, !A}\,(!) \qquad \dfrac{\vdash ?\Delta, B, ?A^{\perp}}{\vdash ?\Delta, !B, ?A^{\perp}}\,(!)}{\vdash ?\Gamma, ?\Delta, !B}\,(cut) \quad \rightsquigarrow \quad \dfrac{\dfrac{\dfrac{\vdash ?\Gamma, A}{\vdash ?\Gamma, !A}\,(!) \qquad \vdash ?\Delta, B, ?A^{\perp}}{\vdash ?\Gamma, ?\Delta, B}\,(cut)}{\vdash ?\Gamma ?\Delta, !B}\,(!)$$

(b) promotion-weakening

$$\dfrac{\dfrac{\vdash ?\Gamma, A}{\vdash ?\Gamma, !A}\,(!) \qquad \dfrac{\vdash \Delta}{\vdash \Delta, ?A^{\perp}}\,(?w)}{\vdash ?\Gamma, \Delta}\,(cut) \quad \rightsquigarrow \quad \genfrac{}{}{0pt}{}{\vdash \Delta}{=\!=\!=\!=}\genfrac{}{}{0pt}{}{}{} \dfrac{\vdash \Delta}{\vdash ?\Gamma, \Delta}\,(?w)^{*}$$

(c) promotion-contraction

$$\dfrac{\dfrac{\vdash ?\Gamma, A}{\vdash ?\Gamma, !A}\,(!) \qquad \dfrac{\vdash \Delta, ?A^{\perp}, ?A^{\perp}}{\vdash ?\Delta, ?A^{\perp}}\,(?c)}{\vdash ?\Gamma, ?\Delta}\,(cut) \quad \rightsquigarrow \quad \dfrac{\dfrac{\dfrac{\vdash ?\Gamma, A}{\vdash ?\Gamma, !A}\,(!) \qquad \dfrac{\dfrac{\vdash ?\Gamma, A}{\vdash ?\Gamma, !A}\,(!) \qquad \vdash ?\Delta, ?A^{\perp}, ?A^{\perp}}{\vdash ?\Gamma, \Delta, ?A^{\perp}}\,(cut)}{\vdash ?\Gamma, ?\Gamma, \Delta}\,(cut)}{\vdash ?\Gamma, \Delta}\,(?c)^{*}$$

図 7　MELL のカット除去規則の一部

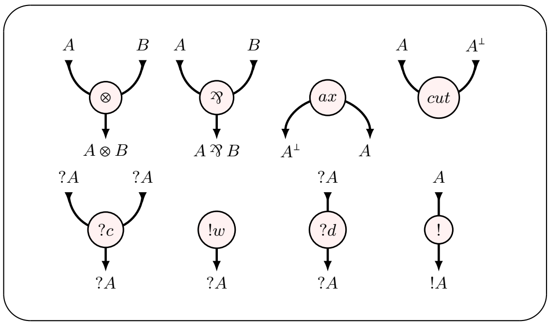

図 8　証明構造の構成要素：セルとワイヤー

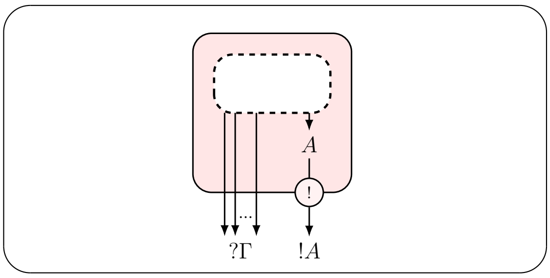

図 9　証明構造の構成要素：Promotion Box

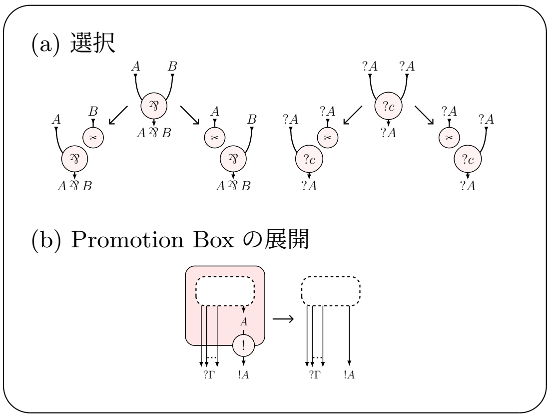

図 10　スイッチング操作

図 24 に添付)．巡回している箇所にはいずれも ⅋ セルが含まれていて，いずれのスイッチンググラフも非巡回であることから，Danos-Regnier の整合性条件を満たしている．対応する証明木は Appendix に添付する．

3.2.2　カット除去

シークエントと同様に，Proof Nets においてもカット除去が定義される．図 12 に，MELL Proof Nets におけるカット除去規則の例を示す．それぞれ図 7 の各規則に対応している．なお，全てのカット除去規則は Appendix に添付する．MELL Proof Nets におけるカット除去は，このように部分グラフの書換えによって表される．

MELL Proof Nets では，カット除去に関して，以

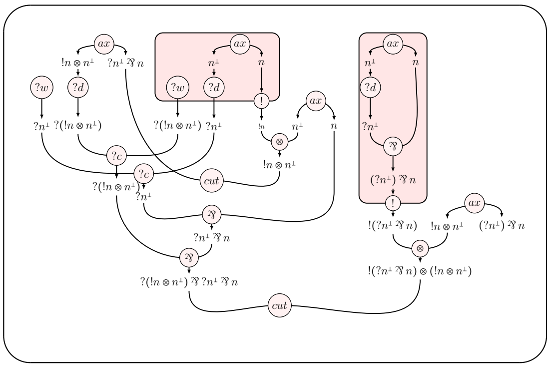

図 11　MELL Proof Net の例

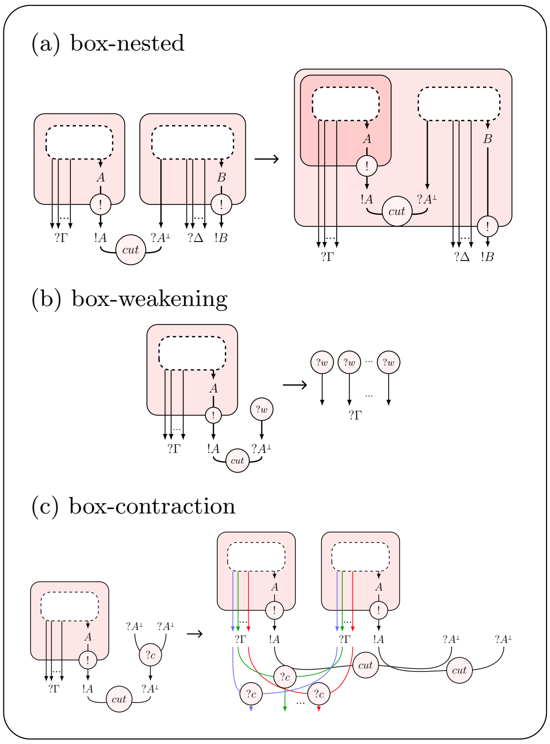

図 12　MELL Proof Nets のカット除去規則の例

下の定理が成り立つ [8] [9] [15]．

定理 3.1 (カット除去定理)．MELL Proof Nets において，任意の Proof Nets に対してカット除去が可能である．

定理 3.2 (安定性)．任意の MELL Proof Net は，カット除去後も MELL Proof Net である．

定理 3.3 (合流性)．MELL Proof Nets のカット除去は合流性を持つ．

定理 3.4 (強正規化性)．MELL Proof Nets のカット除去は強正規化性を持つ．

以上のような，Promotion Box で出現するような，不特定多数のエッジを含む階層構造の書換え規則を一般的なグラフ書換え系で表現する方法は，自明ではない．

4　LMNtal の拡張

本節では，MELL Proof Nets のカット除去で出現する，不特定多数のエッジを含む階層構造や，その書換え規則を，LMNtal 上で記述できるように拡張する．

4.1　nlmem

LMNtal の構文において，不特定多数個の自由リンクは記述できるものの，現行の処理系では，アトムや膜が不特定多数存在することを表す記法は提供してしない．よって，MELL Proof Nets の box-weakening や box-contraction で出現するような，不特定多数個のセルを用いた書換え規則を記述することはできない．関連研究として，量化子を用いた言語拡張 [13](to appear) が挙げられるが，これは主に不特定多数のアトムに関する表現力の拡張を目的としていて，本研究で扱う不特定多数の自由リンクに関する表現力の拡張は行われていない．

SLIM 上には非線形膜ライブラリ nlmem (non-linear membrane) が実装されている [25]．nlmem ライブラリ自体は，階層グラフ書換えにおける不特定多数のエッジを含む膜に関する操作のため，MELL Proof Nets とは無関係の文脈で実装された．nlmem ライブラリには，不特定多数の自由リンクを持つ膜の消去を行う `nlmem.kill` と，消去を行う `nlmem.copy` が実装されている．各機能の動作の概略図を図 13 に示す．

`nlmem.kill` では，第一引数に消去対象の膜を，第二引数には，膜の消去後に接続先を失う自由リンクの先に繋げるアトム (図中では `n` アトムを取っている) を取る．`nlmem.copy` では，第一引数に複製対象の膜を，第二引数には，膜の複製後に浮いてしまう自由リ

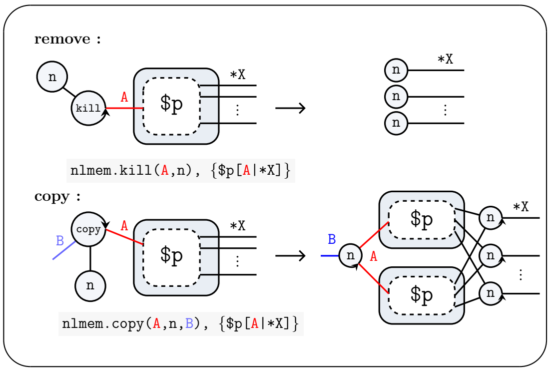

図 13　nlmem ライブラリの動作概略図

ンクを保護するためのアトムを，第三引数には，複製した主ポート A を第二引数のアトムでまとめた際の，主ポートの元の接続先を保存するための自由リンクを取る．これらの操作は，アトム集団によって仕様を記述することができる．アトム集団の用途をこれらの操作に限定し，ライブラリ機能として提供している．

しかしながら，kill，copy ともに，接続先不定の自由リンクを塞ぐための構造がアトムに限定されていたり，contraction のカット除去規則で行われているような主ポートの扱いができないなど，MELL Proof Nets のカット除去規則と直接的に対応が取れない部分がある．

4.2　不特定多数の自由リンクを持つ膜の消去

以降では，前節の課題を解決した，新たな自由リンクの膜の複製，消去機能を持つ LMNtal の拡張について述べる．基本的には，新たな構文は追加せずに，既存の構文を用いて記述するが，表現記法として [20] のアトム集団 (aggregate) を拡張した以下の記法を導入する：

$$\$p[*X_1,*X_2,\ldots,*X_m] \quad (m>0)$$

ここで，各 $*X_i$ はすべて同じ名前の (ルール左辺または右辺の) プロセス文脈に現れるリンク束であり，$|{*X_1}| = |{*X_2}| = \ldots = |{*X_m}|$ である．

これは，アトム集団をプロセス文脈に拡張したもので，m 本の自由リンクを持つ $|{*X_i}|$ 個のプロセス文脈を表す．この表現を用いることで，不特定多数個のプロセス文脈を仕様中に記述することができる．また，

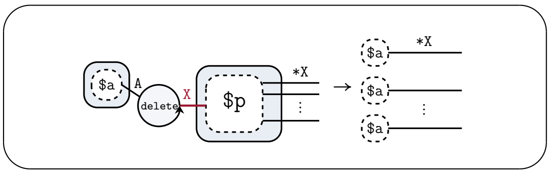

図 14　MELL のカット除去に基づく膜の消去 delete

実装に関しても，この記法自体を新たな構文として実装する方法は自明でないが，nlmem の場合と同様に，用途を限定することで，ライブラリ機能として実装することが可能である．

まず，自由リンクを持つ膜の消去機能を定義する．仕様は以下のように記述される．

```
mell.delete(X,A), {$p[X|*X]}, {$a[A]}
                 → $a[*X]
```

対応する図を図 14 に示す．delete アトムの第 1 引数には消去対象の膜を，第 2 引数には膜の消去後に接続先を失う自由リンクの先に繋げる，1 引数の構造 $a[A] を取る．$a[A] には，アトムだけでなく，膜や，それらを組み合わせた a(A,B),b(B,C),c(C) のような構造も取ることができることに留意する．このとき，プロセス文脈の型付けに関する問題 [23] を回避するため $a[A] は膜で囲む．反応後は，消去対象の膜が消え，*X だけが残り，本来 *X が膜側で接続していた接続していた各リンクが，それぞれ $a[A] のコピーに接続される．このとき，$a[A] を囲っていた膜は剥がされる．このように，mell.delete は，nlmem.kill の第二引数にアトム以外の構造を取れるように拡張したものとして定義される．

4.3　不特定多数の自由リンクを持つ膜の複製

次に，不特定多数の自由リンクを持つ膜の複製機能を定義する．仕様は以下のように記述される．

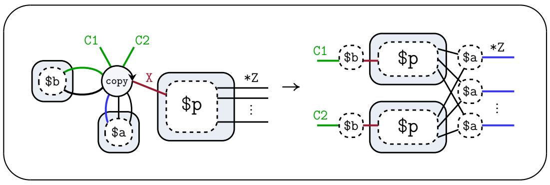

図 15　MELL のカット除去に基づく膜の複製 copy

```
mell.copy(X,A1,A2,A3,B1,B2,C1,C2),
{$p[X|*Z]}, {$a[A1,A2,A3]} ,{$b[B1,B2]}
        → {$p[X'|*Z']}, {$p[X''|*Z'']},
          $a[*Z',*Z'',*Z],
          $b[X',C1], $b[X'',C2].
```

対応する図を図 15 に示す．copy アトムの第 1 引数は複製対象の膜を取る．このとき，copy アトムに直接つながる X の接続元を**主ポート**と呼ぶ．以降の 5 個の引数は，膜を複製した際に，接続先が浮いてしまう自由リンクに繋げるための構造につながるリンクを取る．いずれも，プロセス文脈の型に関する問題を回避するために，膜で囲む．第 2，3，4 引数は，主ポート以外につながるリンク *X を保護するための構造につながるリンクをとる．第 5，6 引数は，主ポートを保護するための構造につながるリンクを取る．第 7，8 引数は，主ポートの外向きのリンクと接続するためのリンクを取る．反応後は，複製対象の膜が複製され，{$p[X'|*Z']}，{$p[X''|*Z'']} が生成される．$a を展開して得られたプロセス文脈群は，*Z'，*Z'' を結び，元の自由リンクである *Z を再現する．$b は，主ポートのリンク X'，X'' を，それぞれ C1，C2 に接続する．このように，mell.copy は主ポートとその他の扱いを分離する．

4.4　例題

プロセス計算における，プロセスの複製や強制終了で，膜の複製や消去を行う例を示す．なお，mell ライブラリは LMNtal 処理系 SLIM 上で実装し，これらの例題について動作確認を行なった．

4.4.1　同期 π 計算

同期 π 計算の，通信に関する規則は以下のように記述される．

$$(x(y).P + M \mid \bar{x}\langle z\rangle.Q + N) \rightarrow P[z/y] \mid Q$$

ここで，通信前 (左辺) において，通信を行うプロセス P，Q と同時に出現するプロセス M，N は，通信後には消去される．プロセスを膜で表現したとき，M，N は膜の消去を意味するが，M，N にも，上記の規則に明示的に現れないチャネルが存在する可能性があるため，不特定多数の自由リンクを持つ膜の消去に対応する．

文献 [25] では，nlmem.kill を用いた以下のようなエンコードを行った．

```
comm@@
{$x,+C1,+C2},{get(C1,Y,{$p[Y|*V1]}),$m},
{snd(C2,Z,{$q}),$n}
  :- {$x},$p[Z|*V1],$q,nlmem.kill({$m,$n}).
```

nlmem.kill を用いていた部分は，mell.delete を用いて以下のように記述できる．

```
comm@@
{$x,+C1,+C2},{get(C1,Y,{$p[Y|*V1]}),$m},
{snd(C2,Z,{$q}),$n}
  :- {$x},$p[Z|*V1],$q,mell.delete({$m,$n},d).
```

4.4.2　Ambient 計算

Ambient 計算 [3] におけるプロセスの複製を例に示す．Ambient 計算とは，ambient と呼ばれる Box 構造を扱う，プロセス計算の一種である．多くのプロセス計算は，プロセスの複製機能 $!P$ を持つ．($!P$ は構造合同規則 $!P \equiv P \mid !P$ によって定義される)．Ambient 計算においてもプロセスの複製が導入されることがあるが，ほとんど $!(\mathrm{open}\ m.P)$ の形式でのみ使用され，以下のような規則で記述される：

$$!(\mathrm{open}\ m.P) \mid m[Q] \rightarrow P \mid Q \mid !(\mathrm{open}\ m.P)$$

直感的には，ambient は膜によってエンコード可能だが，名前管理の詳細な解釈のため，文献 [19] では以下のようなエンコードが行われた．

```
open_repl@@
open_repl(M,{$p}),{amb(M1),
{id,+M1,-M2,$mm},$q,@q},{id,+M,+M2,$m}
  :- nlmem.copy({$p},A1,A2,A3,B1,B2,remove,P),
     {cp(A1,A2,A3)},{copies(B1,B2)},
     $q, {id,+M3,$m,$mm}, open_repl(M3,P).

open_repl_aux@@
copies(A,B), copies({$p},remove) :- A=B, $p.
```

このエンコードにおいては，膜は ambient 構造のエンコードと，名前管理のための構造の，2 とおりの役割を持つ．$P \mapsto$ \$p，$Q \mapsto$ \$q である．m に関しては，名前参照の動的な移動に対応するために，参照を膜を用いた木構造で管理している．プロセスの複製は，`nlmem.copy` を用いて記述されている．この規則は，`mell.copy` を用いて以下のように記述できる．

```
open_repl@@
open_repl(M,{$p}),{amb(M1),
{id,+M1,-M2,$mm},$q,@q},{id,+M,+M2,$m}
  :- mell.copy({$p},A1,A2,A3,B1,B2,remove,P),
     {cp(A1,A2,A3)},{B1=B2},
     $q, {id,+M3,$m,$mm}, open_repl(M3,P).

open_repl_aux@@
remove({$p}):- $p.
```

以上のように，各種プロセス計算におけるプロセスの複製や消去が，Promotion Box の操作と関連する不特定多数の自由リンクを持つ膜の複製や消去で表現できることが示された．Proof Nets における Promotion Box は，内部の構造をアトミックに操作するため，Optimal Reduction 等の局所的書換えに基づく書換え系では排除されることが多いが，並行性が求められる動作の記述や，階層構造を持つ概念の記述においては有用である．

5　MELL Proof Nets の LMNtal へのエンコード

本節では，MELL Proof Nets を拡張 LMNtal にエンコードする方法について述べる．特に，カット除去規則に関して，4 節で導入した機能を用いて簡潔にエンコードできることが期待される．これにより，

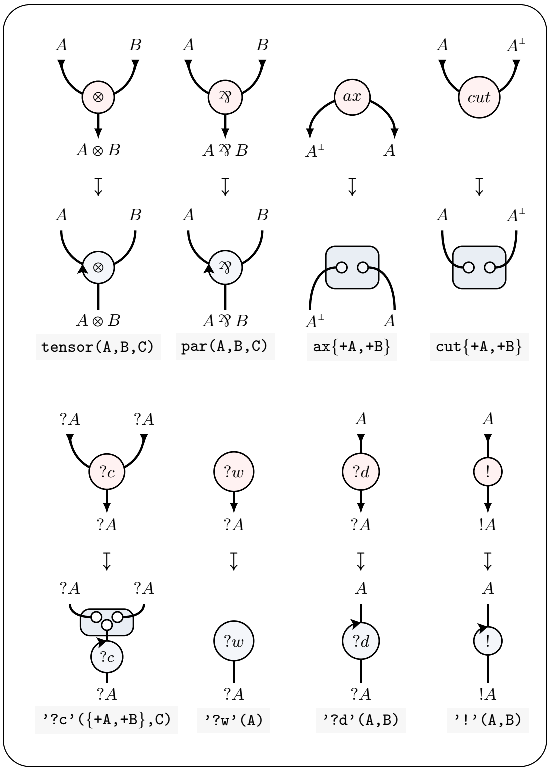

図 16　セル・ワイヤーのエンコード

LMNtal のモデル検査機能が Proof Nets の性質の検証に利用できるようになり，LMNtal が Proof Nets の有用なワークベンチとなることが示される．

5.1　エンコード手法

5.1.1　MELL Proof Nets のエンコード

まず，証明構造の構成要素 (図 8) をエンコードする．セルとワイヤーは，図 16 のようにエンコードする．$\otimes$，$\parr$ セルの入力の順序づけは，アトムで表現する．ax，cut セルは入力に順序がないので，膜で表現する．$?c$ セルは，入出力の区別はアトム，順序のない入力は膜で表現する．また，論理式を表す閉じていないワイヤーは，後述のエンコード例に示すように，アトム `formula` を繋げる．

次に，Promotion Box (図 9) は，図 17 のようにエンコードする．Promotion Box の外枠は膜で表現する．文脈 ?Γ はリンク束 `*X` で，空白部分はプロセス文脈 `$p[X1|*X]` で表現する．

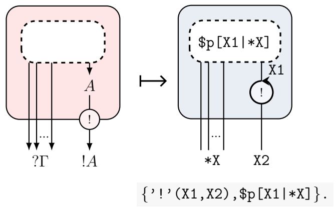

図 17　Promotion Box のエンコード

以上のように，全ての構成要素が直接的にエンコードできた．証明構造は，これらの要素の組み合わせで構成されるので (定義 3.2)，これらのエンコードを組み合わせることで，証明構造全体をエンコードできる．

証明構造のエンコード例を示す．図 11 に示した Proof Net は，図 18 の LMNtal コードで表される．このとき，図 11 右端の $(?n^{\perp})⅋n$ に対応する自由リンクを，アトム `formula` で終端させる (16 行目)．

```
1  ax{+A1,+A2},'?d'(A1,A3),'?w'(A4).
2  {
3    ax{+B1,+B2},'?d'(B1,B3).
4    '?w'(B4),'!'(B2,B5).
5  }.
6  '?c'({+A3,+B4}+C1),'?c'({+A4,+B3},C2).
7  tensor(B5,T1,D2),ax{+T1,+T2}.
8  cut{+A2,+D2}.
9  par(C2,T2,P1),par(C1,P1,F).
10 {
11   ax{+E1,+E2},'?d'(E1,E3).
12   par(E3,E2,E4),'!'(E4,E5).
13 }.
14 tensor(E5,T3,D4),ax{+T3,+T4}.
15 cut{+F,+D4}.
16 formula(T4).
```

図 18　Proof Net のエンコード例

証明構造に関する整合性判定は，LMNtal 上でエンコード可能である．

図 10 で示したスイッチング操作は，部分グラフの変換規則なので，LMNtal のルールで図 19 のように簡潔にエンコードできる．このとき，contraction に関しては，`C1`，`C2` に順序がないため，1 本のルールで記述される．定義 3.3 では，全てのパターンのスイッチンググラフが要求されるが，これは，LMNtal の非決定モードにより，図 19 のルールの全ての適用パターンの終了状態より得られる．

また，Danos-Regnier の整合性条件のような，グラフの形式に関する規則も，図 20，図 21 のようなグラフの変換規則としてエンコードできる．♯アトムは，2 引数のアトム，つまり ⅋，!，$?d$，$?c$ アトムを表す．♭アトムは，1 引数のアトム，つまり $?w$，v，f アトムを表す．`t` アトムは，グラフ構造に流すトークンを表す．`ti` アトムは，最初に 1 度だけ場にトークンを生成するための，チケットのような役割を持つアトムである．まず最初に，いずれかのアトムが 1 度だけ `ti` アトムと反応し，場にトークンを生成する (図 20)．その後トークンは，通ったアトムを消費しながらグラフ構造を走査する．最終的に構造の端，つまり♭アトムに到達すると，そのトークンは 0 価の `ok` アトムに変化する．一方で，構造に循環がある場合は，いずれ 2 つのトークンが衝突して `ng` アトムに変化する (図 21)．このように，token-passing ルールを用いてグラフ構造の整合性を判定することができる．

```
switching_par_r@@
par(A,B,AB) :- par(A,AB),v(B).

switching_par_l@@
par(A,B,AB) :- par(B,AB),v(A).

switching_contraction@@
'?c'({+C1,+C2},X3) :- '?c'(C1,X3),v(C2).

switching_box@@
{'!'(X1,X2),$p[X1|*X]} :- $p[X2|*X]
```

図 19　スイッチング規則のエンコード

5.1.2　MELL Proof Nets カット除去規則のエンコード

続いて，カット除去規則 (図 12) のエンコードを行う．グラフの変換規則は，LMNtal のルールで表現される．

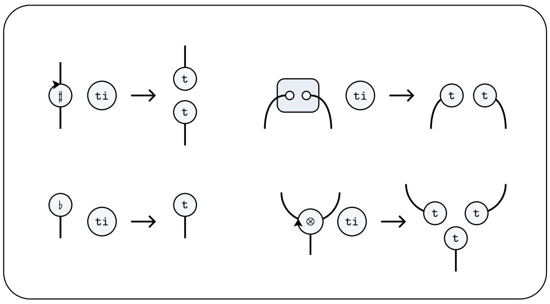

図 20　token-passing による DR 判定：トークンの初期生成

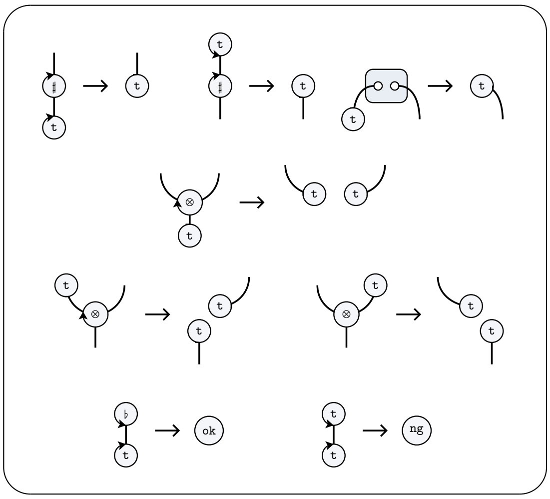

図 21　token-passing による DR 判定

box-nested 規則は，以下のようにエンコードする．

```
box_nested@@
{'!'(X1,X2),$g1[X1|*X]},{$g2[X3|*Y]},
cut{+X2,+X3}
  :- {
      {'!'(X1,X2),$g1[X1|*X]},$g2[X3|*Y]
      cut{+X2,+X3}
     }.
```

図 22 に対応図を示す．このような，ワイヤーの束や階層構造を含む規則は，LMNtal のリンク束と膜を用いて簡潔に記述できる．また，Occurrence Conditions (2.1.1 節) のもとで，このルールはポインタ安全であ

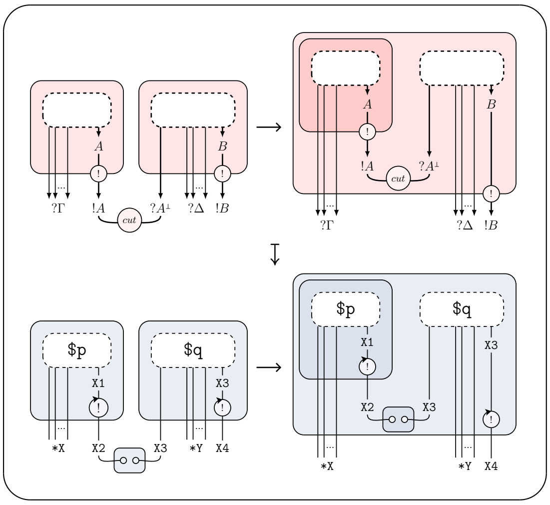

図 22　box-nested 規則のエンコード

ることが保証される．

その他の規則も同様にエンコードできるが (全てのエンコード結果は Appendix に添付する)，Weakening，Contraction の規則に関しては，前述の通り非自明な操作を含んでおり，4 節で導入した機能を用いてエンコードする必要がある．

box-weakening 規則は，以下のようにエンコードする．

```
box_weakening@@
{'!'(X1,X2),$g[X1|*X]},cut{+X2,+X3},'?w'(X3)
  :- mell.kill(X,A),{$p[X|*X]},{'?w'(A)}.
```

図 23(a) に対応図を示す．`mell.delete` を用いて，Promotion Box の内側のリンクに $?w$ アトムを繋ぐようにする．

box-contraction 規則は，以下のようにエンコードする．

```
box_contraction@@
{'!'(X1,X2),$g[X1|*X]},'?c'({+C1,+C2},X3)
cut{+X2,+X3},
  :- mell.copy(X2,A1,A2,A3,B1,B2,C1,C2),
     {'!'(X1,X2),$g[X1|*X]},
     {'?c'({+A1,+A2},A3)},{cut{+B1,+B2}}.
```

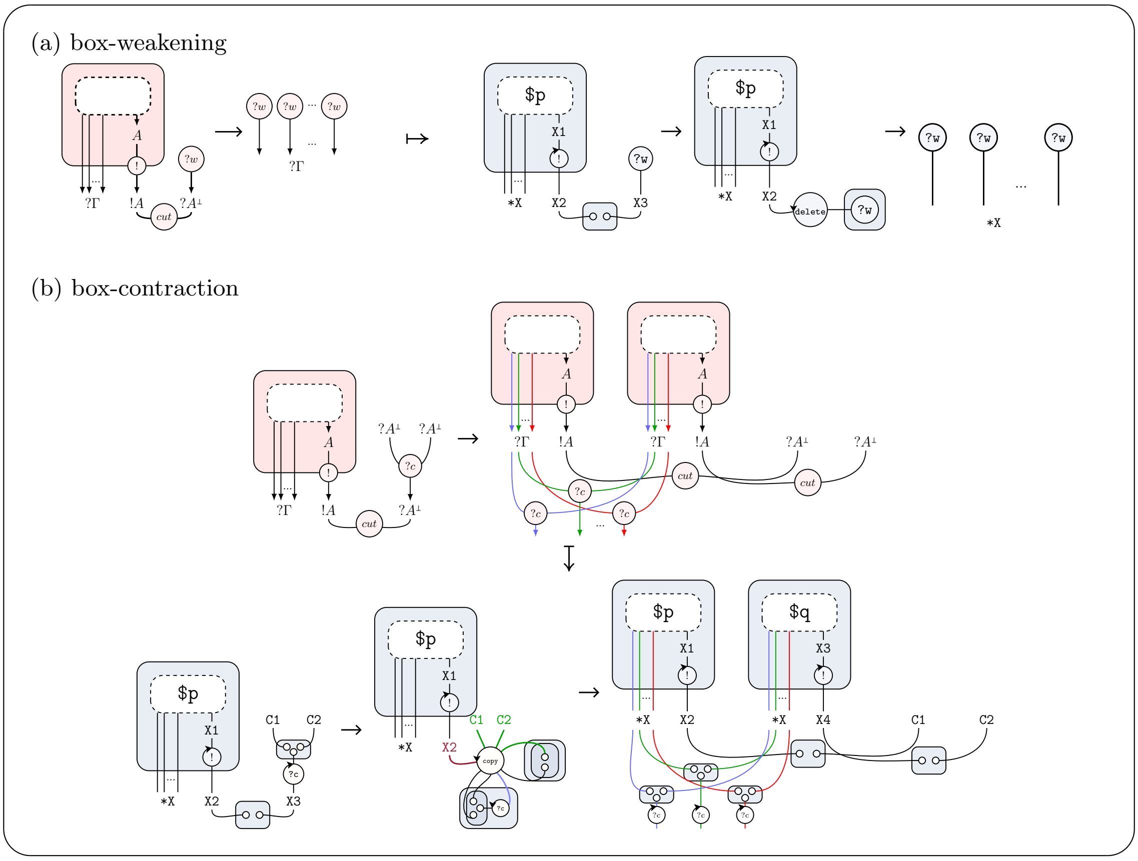

図 23　`mell` **ライブラリを用いたカット除去規則のエンコード**

図 23(b) に対応図を示す．`mell.copy` の第 2，3，4 引数は contraction に対応する構造を取る．`mell.copy` において自由リンクを保護するための構造として，アトム以外も取れるようにしたのは，主にこの部分のエンコードのためである．第 5，6 引数は Cut に対応する構造を，第 7，8 引数は，カット除去前の Contraction の入力側のリンク 2 つを取る．これにより，カット除去後の主ポート周りの構造をエンコードする．`nlmem.copy` では，このようなアトム以外の構造による自由リンクの保護や，主ポートの扱いを行うことができず，エンコードには複数のルールが必要だったが，`mell.copy` の導入によって，1 つのルールでエンコード可能になった．

5.2　エンコードの正しさに関する議論

エンコードの正しさに関して，一般的には，

1. 仕様が与えられ，
2. それを満たすアルゴリズムを記述し，
3. アルゴリズムの健全性，完全性，決定性などを証明する．

のような議論が行われるが，本研究においては，アルゴリズムが限りなく仕様 (証明構造 (定義 3.2)，カット除去規則 (図 12)) に近い形で記述されている．どちらの体系にも対応する図表現が存在しており，エンコードの正しさは，非形式的には図の比較より明らかである．形式的証明が必要な場合には，それぞれの数学的表現の間の対応を取る必要があるが，本論文では詳細を省く．

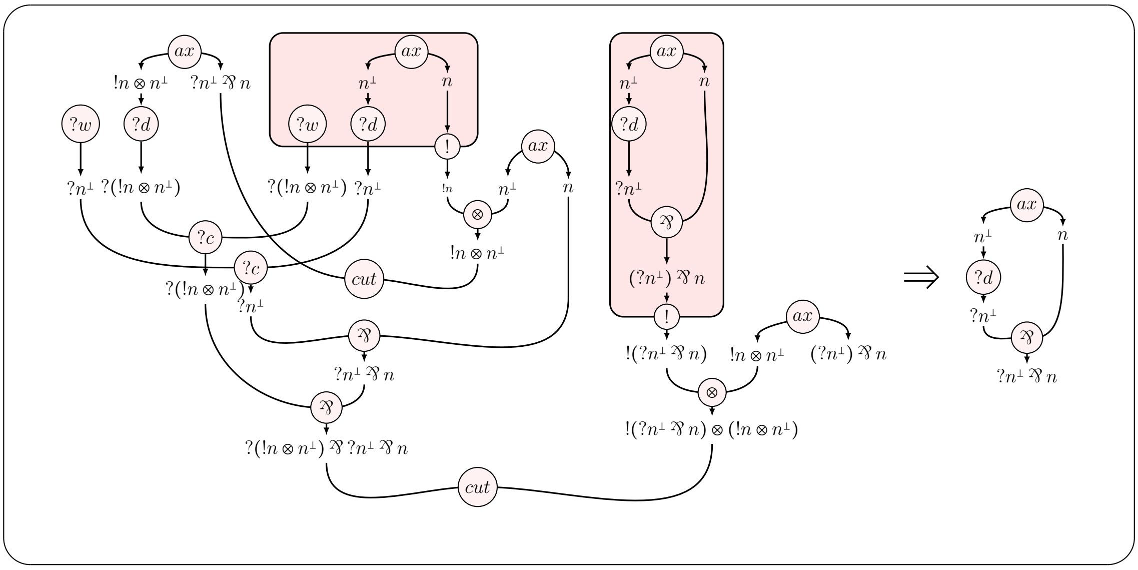

図 24　Proof Net の β 簡約の例

$$
\dfrac{
  \dfrac{
    \dfrac{
      \dfrac{
        \dfrac{}{f : n \to n, x : n \vdash f : n \to n}\,\text{T-Var}
        \qquad
        \dfrac{}{f : n \to n, x : n \vdash x : n}\,\text{T-Var}
      }{f : n \to n, x : n \vdash fx : n}\,\text{T-App}
    }{f : n \to n \vdash \lambda x : n.fx : n \to n}\,\text{T-Abs}
  }{\vdash \lambda f : n \to n.\lambda x : n.fx : n \to n \to n \to n}\,\text{T-Abs}
  \qquad
  \dfrac{\dfrac{}{x : n \vdash x : n}\,\text{T-Var}}{\vdash \lambda x : n.x : n \to n}\,\text{T-Abs}
}{\vdash (\lambda f : n \to n.\lambda x : n.fx)(\lambda x : n.x) : n \to n}\,\text{T-App}
$$

$$
\rightsquigarrow^{*} \quad \dfrac{\dfrac{}{x : n \vdash x : n}\,\text{T-Var}}{\vdash \lambda x : n.x : n \to n}\,\text{T-Abs}
$$

図 25　$(\lambda f{:}n \to n\,.\,\lambda x{:}n\,.\,f\,x)\,(\lambda x{:}n\,.\,x) \to_\beta (\lambda x{:}n\,.\,x)$

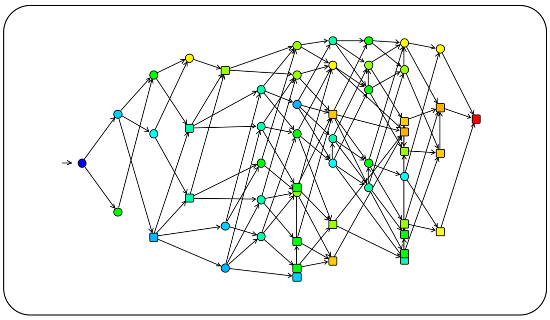

図 26　図 18 へのカット除去適用の状態空間

5.3　実行例：単純型付きラムダ計算の β-簡約

5.3.1　カット除去による β-簡約の状態空間構築

単純型付きラムダ計算を Proof Nets で表現する．$A \to B \;\mapsto ?A^{\perp} \parr B$ より，関数型の型付けは MELL Proof Nets に埋め込むことができる [8] [10]．このとき，カット除去は β-簡約に対応し，カットを全て除去した後の Net は，β-簡約を全て行った，正規形のラムダ式に対応する．

図 24 に，単純型付きラムダ計算 $(\lambda f{:}n \to n\,.\,\lambda x{:}n\,.\,f\,x)\,(\lambda x{:}n\,.\,x) \to (\lambda x{:}n\,.\,x)$ の Proof Net による表現を示す．初期状態の Net は，図 11 と同様のものである (そのエンコードは図 18 に示し

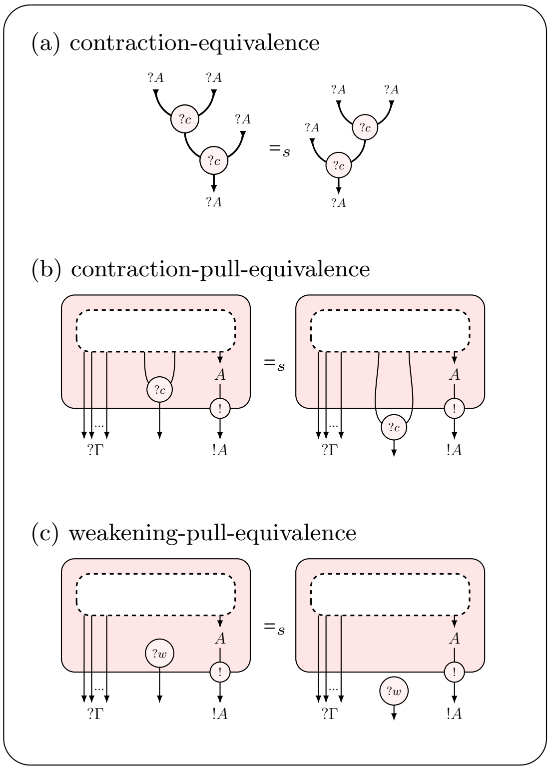

図 27　Pull-equivalence

```
contraction_equation@@
'?c'({+C1,+C2},CC),'?c'({+CC,+C3},C4)
    :- '?c'({+C2,+C3},CC),'?c'({+C1,+CC},C4)

box_contraction_pull@@
{'!'(X1,X2),'?c'({+X3,+X4},X5),$p[X1,X3,X4|*X]}
  :- {'!'(X1,X2),$p[X1,X3,X4|*X]},
     '?c'({+X3,+X4},X5).

box_contraction_push@@
{'!'(X1,X2),$p[X1,X3,X4|*X]},'?c'({+X3,+X4},X5)
  :- {
       '!'(X1,X2),$p[X1,X3,X4|*X]
       '?c'({+X3,+X4},X5),
     }.

box_weakening_pull@@
{'!'(X1,X2),'?w'(X3),$p[X1|*X]}
  :- {'!'(X1,X2),$p[X1|*X]},'?w'(X3).

box_weakening_push@@
{'!'(X1,X2),$p[X1|*X]},'?w'(X3)
  :- {'!'(X1,X2),'?w'(X3),$p[X1|*X]}.
```

図 28　図 27 の各規則の LMNtal エンコード

た) が，これは図 25 の上側の証明図に対応する．$(\lambda f\!:\!n \to n\,.\,\lambda x\!:\!n\,.\,f\,x)(\lambda x\!:\!n\,.\,x)$ は，β-簡約によって正規形 $\lambda x\!:\!n\,.\,x$ に簡約される．$\lambda x\!:\!n\,.\,x$ に対応する証明図は図 25 の下側であり，それに対応する Net は，図 24 の右側の Proof Net である．

図 26 に，図 18 にカット除去規則を適用した状態空間図を示す．初期状態の Net は，最短 10 ステップで正規形 $\lambda x\!:\!n\,.\,x$ に対応する Net に簡約される．また，全ての簡約ルートが，最終的に一つの終了状態 (図中の赤いノード) で停止することから，この例題において，定理 3.3，定理 3.4 の性質が満たされているのが確認できる．

5.3.2　Pull-equivalence の追加

LMNtal でカット除去規則などの Proof Nets に関する変換規則を直接的にエンコードできたことの利点の一つとして，既存の規則の変更や，新たな規則の追加が容易に行えることが挙げられる．

例として，Pull-equivalence と呼ばれる図 27 のような同値関係 $=_s$ [4] を導入し，カット除去と同時並行での適用を許した場合の合流性や正規性への影響を検証する．

図 27 の LMNtal エンコードを図 28 に示す．同値関係を，両向きの書換え規則として表現した．また，(a) contraction-equivalence に関しては，入力の順序がないため，1 本のルールで記述した．(b)，(c) においては，右向き (Promotion Box から要素を取り出す方向) のルールは pull と，左向き (要素を Promotion Box に入れる方向) のルールは push と呼ばれることが多いため，それに従って命名した．

適用対象の Proof Nets として，$(\lambda f\!:\!n \to n\,.\,\lambda x\!:\!n\,.\,f\,(f\,x))\,(\lambda x\!:\!n\,.\,x)$ に対応する，図 29 を用いる．これらの規則をカット除去規則に加えていき，図 29 のエンコードに対して適用した際の状態空間の変化 (全状態の数，終了状態の数) を調査した．表 1 に，各ルールの適用と状態空間の変化を示す．1 段目は，ルールを何も追加せずにカット除去規則のみを適用したもので，状態数は 476 個で，終了状態は 1 個であった．2 段目以降は，図 27 のルールを追

加していく．2 段目は，(a) contraction-equivalence を追加した．状態数と終了状態に変化はなかった．本エンコードにおいては，$?c$ の入力に順序はないため，図 27(a) の両辺は同じ状態としてカウントされる．そのため，contraction-equivalence を追加しても，同じ状態から同じ状態への遷移が追加されるだけであり，状態数は変化しない．3–5 段目は，(b) contraction-pull-equivalence を追加した．3 段目は push のみ追加したが，1 度も適用されることはなく，状態数はカット除去のみ適用したときと同じ 476 個であった．4 段目は pull のみ追加した．状態数は 1808 個に増加し，終了状態は変わらず 1 個で，かつ全ての簡約経路は有限であった (定理 3.3，定理 3.4 が成り立っている)．5 段目は両方同時に追加した．状態数は 1808 個で，終了状態は 1 個であった．状態数に関しては，pull のみ追加した場合と同じであるが，push ルールの適用として，pull ルールの適用の逆向きのみの遷移が追加された．これにより，無限の簡約経路が現れ，強正規化性 (定理 3.4) が失われた．6–8 段目は，(c) weakening-pull-equivalence を追加した．6 段目は push のみ追加した．状態数は 41216 個に増加し，終了状態は 16 個となり，合流性 (定理 3.3) が失われる結果となった．ただし，全ての簡約経路は有限であった．状態数が大幅に増加した原因としては，weakening の pull は，対象となる $?w$ セルが Promotion Box 内に存在するものに限定されているが，push に関しては，対象となる $?w$ セルが Promotion Box 内に存在しないということのみを要求していて，適用対象となる $?w$ セルが増加したためである．7 段目は pull のみ追加した．状態数は 756 個に増加し，終了状態は 1 個で，全ての簡約経路は有限であった (定理 3.3，定理 3.4 が成り立っている)．8 段目は両方同時に追加した．ルール適用は止まることがなく，状態数は爆発した．つまり，強正規化性 (定理 3.4) が失われる結果となった．

以上のように，ルールの追加と，それに伴う書換え系としての性質の変化を，LMNtal 上で容易に調査することができた．

6 まとめ

本研究では，MELL Proof Nets のカット除去規則で行われていた，不特定多数の自由リンクを含む Box 構造の消去，複製操作を，階層グラフ書換え言語 LMNtal に導入することにより，(i) 階層グラフ書換えにおいて，並行性や階層性の表現で Promotion Box が有用であること，(ii) LMNtal が Proof Nets のワークベンチとして有用であることを示した．また，MELL Proof Nets とは無関係な文脈で，同様の操作を実現するために実装された `nlmem.kill`，`nlmem.copy` を，Proof Nets の観点から整理することができた．

謝辞 本研究の一部は科学研究費補助金 (23K11057) および早稲田大学特定課題研究助成費 (2024C-432) の助成を得て行われた

参 考 文 献

[1] Alves, S., Fernández, M., and Mackie, I.: A new graphical calculus of proofs, *Electronic Proceedings in Theoretical Computer Science*, Vol. 48(2011).
[2] Berry, G. and Boudol, G.: The chemical abstract machine, *Theoretical Computer Science*, Vol. 96, No. 1(1992), pp. 217–248.
[3] Cardelli, L. and Gordon, A. D.: Mobile ambients, *Theoretical Computer Science*, Vol. 240, No. 1(2000), pp. 177–213.
[4] Danos, V.: La Logique Linéaire appliquée à l'étude de divers processus de normalisation (principalement du Lambda-calcul), 1990.
[5] Danos, V. and Regnier, L.: The Structure of Multiplicatives, *Archive for Mathematical Logic*, Vol. 28, No. 3(1989), pp. 181–203.
[6] Drewes, F., Hoffmann, B., and Plump, D.: Hierarchical Graph Transformation, *Journal of Computer and System Sciences*, Vol. 64, No. 2(2002), pp. 249–283.
[7] Ene, N. C., Fernández, M., and Pinaud, B.: Attributed hierarchical port graphs and applications, *Electronic Proceedings in Theoretical Computer Science, EPTCS*, Vol. 265(2018), pp. 2–19. Publisher: Open Publishing Association.
[8] Girard, J.-Y.: Linear logic, *Theoretical Computer Science*, Vol. 50, No. 1(1987), pp. 1–101.
[9] Girard, J.-Y.: Linear Logic: A Survey, 1993.
[10] Guerrini, S.: Proof Nets and the λ-Calculus, *Linear Logic in Computer Science*, Ehrhard, T., Girard, J.-Y., Ruet, P., and Scott, P.(eds.), London Mathematical Society Lecture Note Series, Cambridge University Press, 2004, pp. 65–118.
[11] Lafont, Y.: Interaction Nets, *Proceedings of*

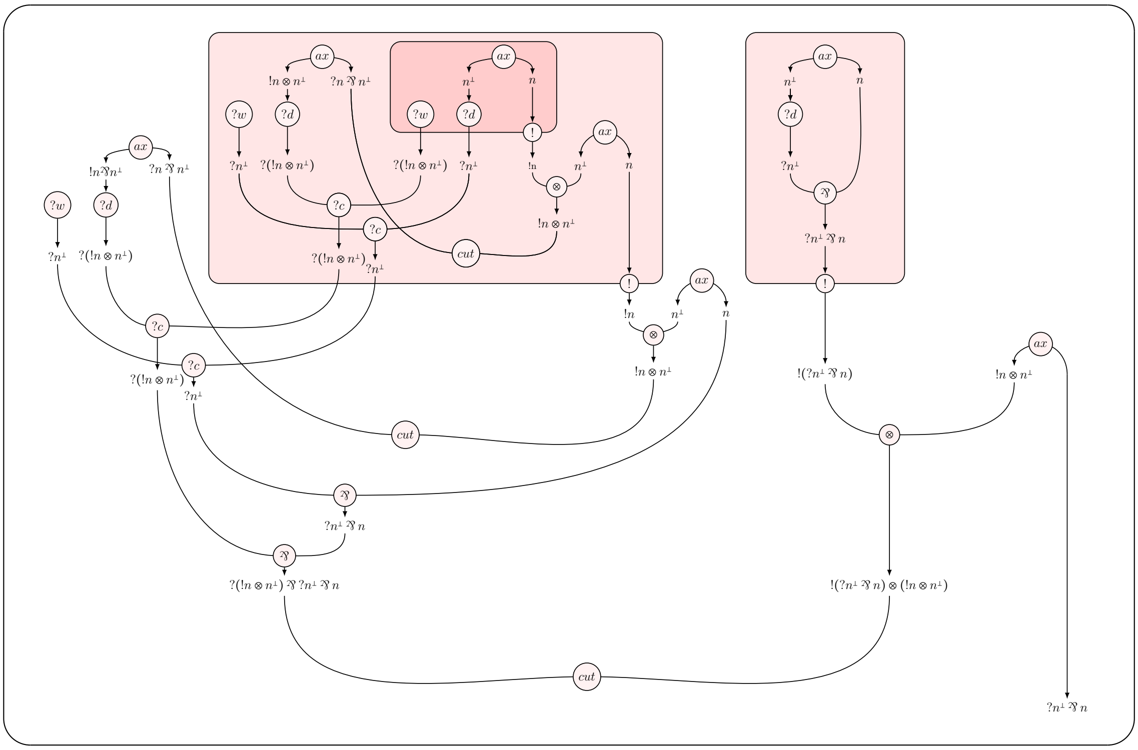

図 29　Proof Net: $(\lambda f\!:\!n \to n\,.\,\lambda x\!:\!n\,.\,f\,(f\,x))\,(\lambda x\!:\!n\,.\,x)$

表 1　各ルール適用時の状態空間

$?c_{\mathrm{eq}}$	$?c_{\mathrm{push}}$	$?c_{\mathrm{pull}}$	$?w_{\mathrm{push}}$	$?w_{\mathrm{pull}}$	状態数	終了状態数
					476	1
✓					476	1
	✓				476	1
		✓			1808	1
	✓	✓			1808	1
			✓		41216	16
				✓	756	1
			✓	✓	∞	—

the 17th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL '90, New York, NY, USA, Association for Computing Machinery, 1989, pp. 95–108.

[12] Milner, R.: Bigraphical Reactive Systems, *Proceedings of the 12th International Conference on Concurrency Theory*, CONCUR '01, Berlin, Heidelberg, Springer-Verlag, 2001, pp. 16–35.

[13] Mishina, H. and Ueda, K.: Introducing Quantification into a Hierarchical Graph Rewriting Language, 34th International Symposium on Logic-Based Program Synthesis and Transformation (LOPSTR 2024), Milano, Italy, September 2024.

[14] Muroya, K.: *Hypernet semantics of programming languages*, Ph.D. thesis, University of Birmingham, 2020.

[15] Pagani, M. and Falco, L. T. d.: Strong normalization property for second order linear logic, *Theoretical Computer Science*, Vol. 411, No. 2(2010), pp. 410–444.

[16] Pinaud, B., Melançon, G., and Dubois, J.: PORGY: A Visual Graph Rewriting Environment for Complex Systems, *Computer Graphics Forum*, Vol. 31, No. 3(2012), pp. 1265–1274. Publisher: Wiley.
[17] Sano, J.: Implementing G-Machine in HyperLMNtal, *CoRR*, Vol. abs/2103.14698(2021). arXiv: 2103.14698.
[18] Takyu, K. and Ueda, K.: Encoding MELL Cut Elimination into a Hierarchical Graph Rewriting Language, *The 21st Asian Symposium on Programming Languages and Systems SRC&Posters*, 2023.
[19] Ueda, K.: Encoding Distributed Process Calculi into LMNtal, *Electronic Notes in Theoretical Computer Science*, Vol. 209(2008), pp. 187–200.
[20] Ueda, K.: LMNtal as a hierarchical logic programming language, *Theoretical Computer Science*, Vol. 410, No. 46(2009), pp. 4784–4800.
[21] 綾野貴之, 堀泰祐, 岩澤宏希, 小川誠司, 上田和紀: 統合開発環境による LMNtal モデル検査, コンピュータソフトウェア, Vol. 27, No. 4(2010), pp. 4_197–4_214.
[22] 後町将人, 堀泰祐, 上田和紀: LMNtal 実行時処理系の並列モデル検査器への発展, コンピュータ ソフトウェア, Vol. 28, No. 4(2011), pp. 4_137–4_157.
[23] 工藤晋太郎, 加藤紀夫, 上田和紀: LMNtal 処理系におけるグラフ構造の操作機能の設計と実装, 情報科学技術レターズ, Vol. 4, August 2005, pp. 9–12.
[24] 冨岡太一, 上田和紀: グラフ書換え言語 LMNtal からの C プログラムの自動生成, 日本ソフトウェア科学会第 *34* 回大会, 2017.
[25] 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. 1_124–1_150.

A 図 11 に対応する MELL の証明木

図 30 に，図 11 に対応する証明木を示す.

B 全てのカット除去規則

図 31 に，MELL のシークエント計算の全てのカット除去規則を，図 32 に，MELL Proof Nets における全てのカット除去規則を，図 33 に，その LMNtal エンコードを示す.

$$
\begin{prooftree}
\AxiomC{}
\RightLabel{(ax)}
\UnaryInfC{$\vdash !n \otimes n^{\perp}, ?n^{\perp} \parr n$}
\RightLabel{$(?d)$}
\UnaryInfC{$\vdash ?(!n \otimes n^{\perp}), ?n^{\perp} \parr n$}
\RightLabel{$(?w)$}
\UnaryInfC{$\vdash ?(!n \otimes n^{\perp}), ?n^{\perp}, ?n^{\perp} \parr n$}
\AxiomC{}
\RightLabel{(ax)}
\UnaryInfC{$\vdash n^{\perp}, n$}
\RightLabel{$(?d)$}
\UnaryInfC{$\vdash ?n^{\perp}, n$}
\RightLabel{$(?w)$}
\UnaryInfC{$\vdash ?(!n \otimes n^{\perp}), ?n^{\perp}, n$}
\RightLabel{$(!)$}
\UnaryInfC{$\vdash ?(!n \otimes n^{\perp}), ?n^{\perp}, !n$}
\AxiomC{}
\RightLabel{(ax)}
\UnaryInfC{$\vdash n, n^{\perp}$}
\RightLabel{$(\otimes)$}
\BinaryInfC{$\vdash ?(!n \otimes n^{\perp}), ?n^{\perp}, n, !n \otimes n^{\perp}$}
\RightLabel{(cut)}
\BinaryInfC{$\vdash ?(!n \otimes n^{\perp}), ?(!n \otimes n^{\perp}), ?n^{\perp}, ?n^{\perp}, n$}
\RightLabel{$(?c)$}
\UnaryInfC{$\vdash ?(!n \otimes n^{\perp}), ?(!n \otimes n^{\perp}), ?n^{\perp}, n$}
\RightLabel{$(?c)$}
\UnaryInfC{$\vdash ?(!n \otimes n^{\perp}), ?n^{\perp}, n$}
\RightLabel{$(\parr)$}
\UnaryInfC{$\vdash ?(!n \otimes n^{\perp}), ?n^{\perp} \parr n$}
\RightLabel{$(\parr)$}
\UnaryInfC{$\vdash ?(!n \otimes n^{\perp}) \parr ?n^{\perp} \parr n$}
\AxiomC{}
\RightLabel{(ax)}
\UnaryInfC{$\vdash n^{\perp}, n$}
\RightLabel{$(?d)$}
\UnaryInfC{$\vdash ?n^{\perp}, n$}
\RightLabel{$(\parr)$}
\UnaryInfC{$\vdash ?n^{\perp} \parr n$}
\RightLabel{$(!)$}
\UnaryInfC{$\vdash !(?n^{\perp} \parr n)$}
\AxiomC{}
\RightLabel{(ax)}
\UnaryInfC{$\vdash !n \otimes n^{\perp}, ?n^{\perp} \parr n$}
\RightLabel{$(\otimes)$}
\BinaryInfC{$\vdash !(?n^{\perp} \parr n) \otimes (!n \otimes n^{\perp}), ?n^{\perp} \parr n$}
\RightLabel{(cut)}
\BinaryInfC{$\vdash ?n^{\perp} \parr n$}
\end{prooftree}
$$

図 30　図 11 に対応する証明木

ax-cut

$$
\begin{prooftree}
\AxiomC{}
\RightLabel{(ax)}
\UnaryInfC{$\vdash A, A^{\perp}$}
\AxiomC{$\vdash \Gamma, A$}
\RightLabel{(cut)}
\BinaryInfC{$\vdash \Gamma, A$}
\end{prooftree}
\quad \rightsquigarrow \quad \vdash \Gamma, A
$$

tensor-par

$$
\begin{prooftree}
\AxiomC{$\vdash \Gamma, A, B$}
\RightLabel{$(\parr)$}
\UnaryInfC{$\vdash \Gamma, A \parr B$}
\AxiomC{$\vdash \Delta, A^{\perp}$}
\AxiomC{$\vdash \Sigma, B^{\perp}$}
\RightLabel{$(\otimes)$}
\BinaryInfC{$\vdash \Delta, \Sigma, A^{\perp} \otimes B^{\perp}$}
\RightLabel{(cut)}
\BinaryInfC{$\vdash \Gamma, \Delta, \Sigma$}
\end{prooftree}
\quad \rightsquigarrow \quad
\begin{prooftree}
\AxiomC{$\vdash \Gamma, A, B$}
\AxiomC{$\vdash \Delta, A^{\perp}$}
\RightLabel{(cut)}
\BinaryInfC{$\vdash \Gamma, \Delta, B$}
\AxiomC{$\vdash \Sigma, B^{\perp}$}
\RightLabel{(cut)}
\BinaryInfC{$\vdash \Gamma, \Delta, \Sigma$}
\end{prooftree}
$$

promotion-nested

$$
\begin{prooftree}
\AxiomC{$\vdash ?\Gamma, A$}
\RightLabel{$(!)$}
\UnaryInfC{$\vdash ?\Gamma, !A$}
\AxiomC{$\vdash ?\Delta, B, ?A^{\perp}$}
\RightLabel{$(!)$}
\UnaryInfC{$\vdash ?\Delta, !B, ?A^{\perp}$}
\RightLabel{(cut)}
\BinaryInfC{$\vdash ?\Gamma, ?\Delta, !B$}
\end{prooftree}
\quad \rightsquigarrow \quad
\begin{prooftree}
\AxiomC{$\vdash ?\Gamma, A$}
\RightLabel{$(!)$}
\UnaryInfC{$\vdash ?\Gamma, !A$}
\AxiomC{$\vdash ?\Delta, B, ?A^{\perp}$}
\RightLabel{(cut)}
\BinaryInfC{$\vdash ?\Gamma, ?\Delta, B$}
\RightLabel{$(!)$}
\UnaryInfC{$\vdash ?\Gamma ?\Delta, !B$}
\end{prooftree}
$$

promotion-dereliction

$$
\begin{prooftree}
\AxiomC{$\vdash ?\Gamma, A$}
\RightLabel{$(!)$}
\UnaryInfC{$\vdash ?\Gamma, !A$}
\AxiomC{$\vdash \Delta, A^{\perp}$}
\RightLabel{$(?d)$}
\UnaryInfC{$\vdash \Delta, ?A^{\perp}$}
\RightLabel{(cut)}
\BinaryInfC{$\vdash ?\Gamma, \Delta$}
\end{prooftree}
\quad \rightsquigarrow \quad
\begin{prooftree}
\AxiomC{$\vdash ?\Gamma, A$}
\AxiomC{$\vdash \Delta, A^{\perp}$}
\RightLabel{(cut)}
\BinaryInfC{$\vdash ?\Gamma, \Delta$}
\end{prooftree}
$$

promotion-weakening

$$
\begin{prooftree}
\AxiomC{$\vdash ?\Gamma, A$}
\RightLabel{$(!)$}
\UnaryInfC{$\vdash ?\Gamma, !A$}
\AxiomC{$\vdash \Delta$}
\RightLabel{$(?w)$}
\UnaryInfC{$\vdash \Delta, ?A^{\perp}$}
\RightLabel{(cut)}
\BinaryInfC{$\vdash ?\Gamma, \Delta$}
\end{prooftree}
\quad \rightsquigarrow \quad
\begin{prooftree}
\AxiomC{$\vdash \Delta$}
\doubleLine
\RightLabel{$(?w)$}
\UnaryInfC{$\vdash ?\Gamma, \Delta$}
\end{prooftree}
$$

promotion-contraction

$$
\begin{prooftree}
\AxiomC{$\vdash ?\Gamma, A$}
\RightLabel{$(!)$}
\UnaryInfC{$\vdash ?\Gamma, !A$}
\AxiomC{$\vdash \Delta, ?A^{\perp}, ?A^{\perp}$}
\RightLabel{$(?c)$}
\UnaryInfC{$\vdash ?\Delta, ?A^{\perp}$}
\RightLabel{(cut)}
\BinaryInfC{$\vdash ?\Gamma, ?\Delta$}
\end{prooftree}
\quad \rightsquigarrow \quad
\begin{prooftree}
\AxiomC{$\vdash ?\Gamma, A$}
\RightLabel{$(!)$}
\UnaryInfC{$\vdash ?\Gamma, !A$}
\AxiomC{$\vdash ?\Gamma, A$}
\RightLabel{$(!)$}
\UnaryInfC{$\vdash ?\Gamma, !A$}
\AxiomC{$\vdash ?\Delta, ?A^{\perp}, ?A^{\perp}$}
\RightLabel{(cut)}
\BinaryInfC{$\vdash ?\Gamma, \Delta, ?A^{\perp}$}
\RightLabel{(cut)}
\BinaryInfC{$\vdash ?\Gamma, ?\Gamma, \Delta$}
\doubleLine
\RightLabel{$(?c)$}
\UnaryInfC{$\vdash ?\Gamma, \Delta$}
\end{prooftree}
$$

図 31　MELL のカット除去規則

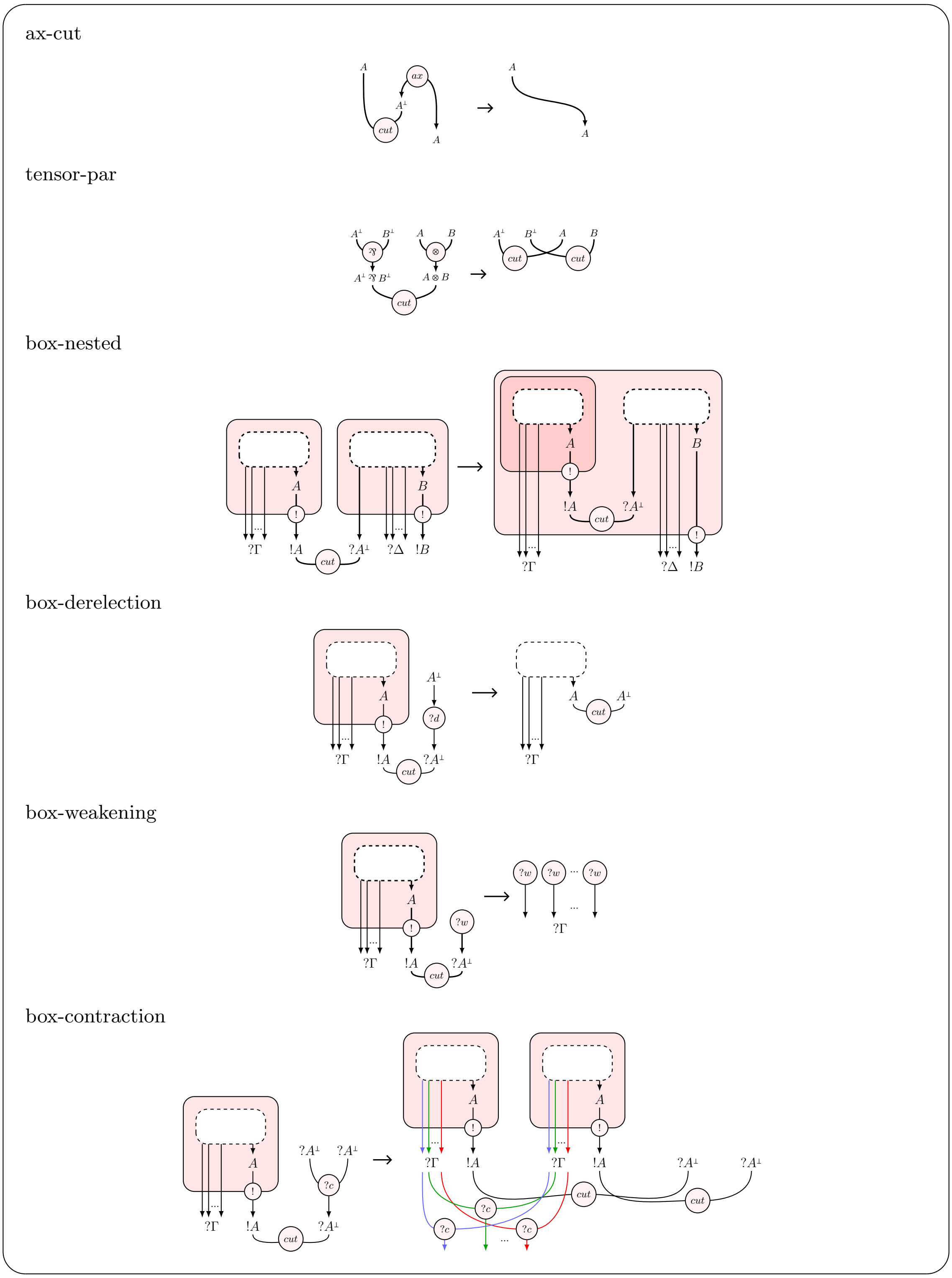

図 32　MELL Proof Nets のカット除去規則

```
//// ax-cut
ax_cut@@
cut{+X,+Y},ax{+Y,+Z}
  :- X=Z.

//// tensor-par
tensor_par@@
tensor(X1,Y1,XY1),par(X2,Y2,XY2),cut{+XY1,+XY2}
  :- cut{+X1,+X2},cut{+Y1,+Y2}.

// box
//// box-nested
box_nested@@
{'!'(X1,X2),$g1[X1|*X],@r1},cut{+X2,+X3},{$g2[X3|*Y],@r2}
  :- {{'!'(X1,X2),$g1[X1|*X],@r1},cut{+X2,+X3},$g2[X3|*Y],@r2}.

//// box-dereliction
box_dereliction@@
{'!'(X1,X2),$g[X1|*X],@r},cut{+X2,+X3},'?d'(X4,X3)
  :- cut{+X1,+X4}, $g[X1|*X],@r.

////  box-weakening
box_weakening@@
{'!'(X1,X2),$g[X1|*X]},cut{+X2,+X3},'?w'(X3)
  :- mell.delete(X1,W),{$g[X1|*X]},{'?w'(W)}.

//// box-contration
box_contraction@@
{'!'(X1,X2),$g[X1|*X]},cut{+X2,+X3},'?c'({+C1,+C2},X3)
  :- mell.copy(X2,A1,A2,A3,B1,B2,C1,C2),{'!'(X1,X2),$g[X1|*X]},
      {'?c'({+A1,+A2},A3)},{cut{+B1,+B2}}.
```

図 33　カット除去規則の LMNtal エンコード (full)