グラフ書換え言語における数値制約を伴う型の静的型検査

山本 直輝　上田 和紀

グラフ書換え言語 LMNtal は，グラフの書換えによって計算を表現するプログラミング言語としての側面と，一般の複雑なグラフ構造を扱えるモデリング言語としての側面とを兼ね備えた言語である．そして，LMNtal の処理系は通常のプログラム実行とモデル検査の双方に対応した，強力な機能を提供している．グラフ書換え言語では，関数型言語で扱える代数的データ構造よりも広範な構造を扱えるため，静的な型検査の定式化は自明でない．そこで，書換え規則の型検査の手法として，グラフを対象とする静的型検査手法である Structured Gamma を基にした，LMNtal ShapeType が提案されている．本論文では，LMNtal ShapeType に対し，LMNtal の拡張機能である型付きプロセス文脈を援用することにより，添字付きの非終端記号を表現し，赤黒木などの数値制約を伴うグラフ型を対象とした書換え規則の型検査を行う手法について述べる．

A graph rewriting language LMNtal has aspects of both a *programming* language that expresses computation by graph rewriting and a *modeling* language that can handle complicated graph structures. Furthermore, its implementation integrates both regular program execution and model checking. In graph rewriting languages, we can handle a broader class of graph structures than algebraic data structures that can be handled by functional languages. As a type checking method for rewrite rules, LMNtal ShapeType was proposed based on Structured Gamma, a static type checking method for graphs. This paper proposes a type checking method for rewrite rules targeting graph types with numeric constraints (e.g., red-black trees) by using typed process context, which is an extension of LMNtal, as the index of nonterminal symbols.

1　はじめに

ネットワーク構造は，インターネットをはじめとして，情報網や交通網，あるいは人間関係など，様々な分野でみられる．あるいは，ミクロな視点で見ると，例えばコンピュータは各部品が接続関係にあるネットワークであり，それらの部品を構成する電子回路は素子が相互に接続されたネットワークである．こうしたネットワークの構造を抽象化・モデル化したデータ構造がグラフである．

一般のグラフは，代数的データ型（リストや木構造）よりも複雑な構造をもち，それ故に既存のプログラミング言語では扱うことが難しい．例えば，C 言語のようなポインタを扱える言語であれば，ポインタの接続構造としてグラフを表現することができるが，依然としてダングリングポインタなどの不正なポインタが発生する危険性がある．

そこで，グラフを第一級オブジェクトとしてもち，それを書き換えることによって計算を表現する，**グラフ書換え言語**というパラダイムが提案されている．

LMNtal[16] は，グラフ書換え言語の一つであり，アトム（ノード）やルール（書換え規則）といった最小限の言語要素からなる言語モデルである．しかしながら，LMNtal は十分な表現力・計算能力をも持ち合わせている．実際，LMNtal はチューリング完全であり，ラムダ計算などの他の言語体系を LMNtal プログラムにエンコードする手法が存在している [9] [10]．

---

* Static Type Checking for Types with Numeric Constraints in Graph Rewriting Languages.

Naoki Yamamoto, 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.

また，LMNtal はプログラミング言語としての側面の他に，モデリング言語としての側面を持ち合わせている．実際，LMNtal の実行時処理系である SLIM [17] [15] には，これらの 2 つの側面とそれぞれ対応する 2 種類の実行モードがある．すなわち，LMNtal をプログラミング言語とみなす場合には SLIM は実行時処理系となり，モデリング言語とみなす場合には SLIM はモデル検査器となる．特にモデル検査器としての SLIM にはグラフの書換えによって遷移しうる全状態を出力する機能があるほか，LTL モデル検査などの機能も充実している．

LMNtal グラフは well-formed であるかぎり，不正なポインタを含まない．この意味で，LMNtal は言語仕様のレベルにおいて不正なポインタを排除している（すなわち，ポインタ安全である）といえる．一方で，LMNtal には静的型の概念がないため，書換えの結果がプログラマの意図しないものとなることがある．

例えば，スキップリスト [6] は線形リストに対して途中の幾つかのノードを通過するようなエッジを追加した，図 1 (a) に示すようなデータ構造である．これは LMNtal グラフとして図 1 (b) のように表現できる．

ここで，スキップリストの `n2` ノードが 2 つ並んだところがあったら，右側を `n1` に書き換える，という書換え規則を考え，次の 2 通りを書いたとする．

```
n2(X,A,B,C,D), n2(Y,E,F,B,A)
  :- n2(X,A,E,C,D), n1(Y,F,A).
```

```
n2(X,A,B,C,D), n2(Y,E,F,B,A)
  :- n2(X,A,F,C,D), n1(Y,E,A).
```

このように，特にテキスト表現においては，どちらが正しい書換え規則であるかをひと目で判断することは難しい．なお，これらを図に直すと，前者は図 2 (a)，後者は図 2 (b) にそれぞれ示す通りの書換え規則になっている．後者は右側のエッジの繋がり方が変わってしまっているので，このルールを適用するとスキップリストの形状が破壊されてしまうことになる．

こうした問題点を解決するため，これまでにも LMNtal を対象とする幾つかの静的型検査手法が提案されている．例えば，[10] の手法ではノード間の局所的な繋がりに着目し，入出力の方向性を解析し，それを −1 から 1 までの実数値（極性）として表現することで型を付ける．

一方，本研究で扱う **LMNtal ShapeType** [13] は，Structured Gamma [3] およびそのサブセットである Shape types [2] を基とした型検査手法である．この手法では，LMNtal のルールを生成規則として，生成文法により型の定義を行う．そのため，SLIM のモデル検査機能を用いることによって，LMNtal 自身で LMNtal の型検査を行えるという利点がある．ある言語の型をその言語自身で記述し，その言語自身によって型検査を行うという取り組みが統一性の観点から興味深いだけでなく，LMNtal の柔軟な表現力をそのまま型の定義に活かすことができるのは大きな強みである．

本論文では，LMNtal ShapeType に対し，ベース言語 LMNtal の拡張機能である型付きプロセス文脈を援用することにより，添字付きの非終端記号を表現し，高さ n の完全二分木や（平衡している）赤黒木，あるいは長さ n のリストなどの数値制約を伴うグラフ型を対象とした書換え規則の型検査を行う手法について述べる．また，ルールの型検査を既存手法より容易に実装するための手法についても述べ，その正当性を示す．

2 関連研究

本節では，まずグラフを対象とする既存の型検査手法を紹介する．Graph types [5] は正規表現を基にしたグラフの型検査手法である．また，Structured Gamma [3] は文脈自由文法を基にした，生成規則によって型を定義する型検査手法である．Shape Types [2] はこの Structured Gamma のサブセットであり，型検査の完全性が満たされるように型定義の範囲を制限したものである．LMNtal ShapeType [13] は Shape Types の技法を LMNtal に対して応用し，書換え規則の型検査を行う手法である．以上の手法では，「グラフの書換え操作を 1 ステップ進めても，構造が破壊されない」ことの検査・保証を行うことができる．しか

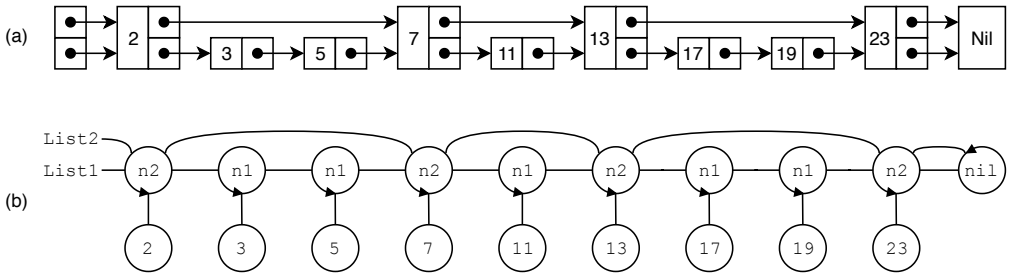

図 1　スキップリスト：(a) グラフによる表現 および (b) LMNtal グラフによる表現

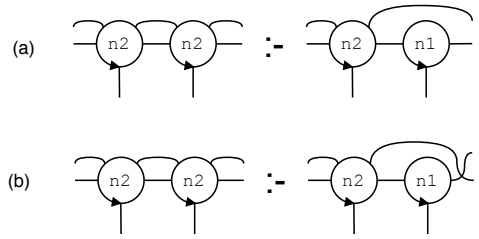

図 2　スキップリストの書換え規則：(a) 正しいもの および (b) 誤ったもの

し，その対象となる型の表現力には後述する文脈自由性による制限があり，完全二分木や赤黒木などの数値制約の入った型を表現することはできていなかった．

また,「二分木を入力するとリストが出力される」のように操作の前後で型が変化する場合や,「リストを 2 本入力すると 1 本のリストが出力される」のように複数の構造を入力にとる場合について対応していなかった．そこで [14] では，LMNtal でよく用いられるデザインパターンの「関数的アトム」に着目して，以上のような複数ステップを要し，かつ型の変化を伴うようなグラフ操作について型検査を行う手法について述べた．

一方で，関数型言語をベースとした型体系では，型に対して述語による数値制約を掛ける手法として篩型（refinement types [12]）が知られている．また，この篩型を述語抽象化によって自動的に推論する手法として Liquid Types [7] が存在する．他のアプローチとして，赤黒木の型を高階型などの機能を用いて表現する研究 [4] もある．

3　グラフ書換え言語 LMNtal

本節では，グラフ書換え言語 LMNtal について簡単に説明する．なお，本来 LMNtal には膜による階層や，ルールの発火に関する制約条件を記述するためのガード及びプロセス文脈といった機能が備わっているが，本論文では簡単のためこれらを除外したサブセット言語を扱う．また，LMNtal の拡張言語である HyperLMNtal [11] において導入された拡張機能である，超辺を表すハイパーリンクについても扱わない．しかしながら，これらの言語機能をもたない LMNtal もまたチューリング完全であり，プログラミング言語としてもモデリング言語としても十分強力である．

3.1　構文

LMNtal の構文を図 3 に示す． LMNtal プログラムは，**グラフ**（アトムの多重集合）と**ルールセット**（ルールの多重集合）の組として表現され，これを**プロセス**と呼ぶ．p という名前で， m 本のリンク $X_1, \ldots, X_m$ をもつアトムを $p(X_1, \ldots, X_m)$ と書く．これらのリンクをアトムの引数といい，順序がついている．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．アトム名と引数の本数（価数）の組を**ファンクタ**といい，p/m のように書き,「m 価の p アトム」のように読む．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節点（ノード）・辺（エッジ）と対応している．

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

プロセス

$P ::= G,R$

グラフ

$G ::= \mathbf{0}$ （空）

| $p(X_1,\ldots,X_m)$ （アトム, $m \geq 0$）

| G,G （分子）

ルールセット

$R ::= \mathbf{0}$ （空）

| [$RuleName$ `@@`] G `:-` G （ルール）

| R,R （分子）

図 3　LMNtal の構文

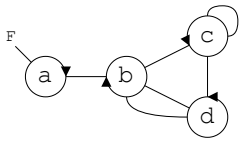

図 4　LMNtal グラフの例

プの出現を許す）．例えば，

```
a(A,F), b(A,C,L1,L2), c(C,D,S,S), d(D,L1,L2)
```

というアトムの多重集合は，図 4 に示すような無向グラフを意味する．図中において矢印は第 1 引数の場所を指しており，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．

グラフはいずれも次の**リンク条件**を満たす：

> グラフ中に同名のリンクは高々 2 回出現する．

グラフ中に 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．自由リンクのないグラフは**閉じている**という．2 つのグラフをカンマで結合する際は，局所リンク名が衝突しないように α 変換されたものとみなす．

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

(E1)　$\mathbf{0},P \equiv P$

(E2)　$P,Q \equiv Q,P$

(E3)　$P,(Q,R) \equiv (P,Q),R$

(E4)　$P \equiv P[Y/X]$

（X は P の局所リンク）

(E5)　$P \equiv P' \Rightarrow P,Q \equiv P',Q$

(E7)　$X{=}X \equiv \mathbf{0}$

(E8)　$X{=}Y \equiv Y{=}X$

(E9)　$X{=}Y,P \equiv P[Y/X]$

（P はアトム，X は P の自由リンク）

図 5　LMNtal グラフ上の構造合同関係

1 価の `a` アトムを，1 価の `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．また可読性のために，ルールはカンマで区切るかわりに，ピリオドで終わる形で書かれることもある．なお，書換えの前後において自由リンクの本数が増減することはあり得ないので，ルール中に同じリンクはちょうど 2 回だけ出現する．

3.2　意味論

LMNtal の意味論は，**構造合同**と**遷移規則**からなる．これらについて順に説明する．

3.2.1　構造合同

グラフ理論における “同型” にあたるものとして，LMNtal にはグラフ間の構造合同関係 “$\equiv$” が存在する．この関係は，図 5 の規則を満たす最小の同値関係として定義される．構造合同な LMNtal プロセスは互いに 0 ステップで変換可能である．なお，$P[Y/X]$ はグラフ P に出現するリンク X をリンク Y に置き換えることを意味する．ただし，(E6), (E10) は膜に関する規則であるので省略した．(E1)–(E3), (E5) はプロセス計算にもみられる一般的な規則であり，(E4) は局所リンク名の α 変換である．(E7) から (E9) までは特別な 2 価のアトムである**コネクタ**に関する規則である．$=(X,Y)$ は二つのリンク X, Y を接続するという意味を持ち，中置記法で $X{=}Y$ とも書く．(E7)

$$\text{(R1)}\quad \frac{G_1 \xrightarrow{T\ :\!-\ U} G_1'}{G_1,G_2 \xrightarrow{T\ :\!-\ U} G_1',G_2}$$

$$\text{(R3)}\quad \frac{G_2 \equiv G_1 \quad G_1 \xrightarrow{T\ :\!-\ U} G_1' \quad G_1' \equiv G_2'}{G_2 \xrightarrow{T\ :\!-\ U} G_2'}$$

$$\text{(R6)}\quad T \xrightarrow{T\ :\!-\ U} U$$

図 6　LMNtal グラフ上の遷移関係

は一つのリンクの両端が繋がって輪になっているものは空とみなして良いこと，(E8) はリンクが対称である（つまり無向である）こと，(E9) は直接接続されている二つのリンクが一つのリンクに縮約できることを表している．

3.2.2　遷移規則

LMNtal における本質的な計算ステップを表す，グラフ間の二項関係として，**ルール T :- U によるグラフの遷移関係** "$\xrightarrow{T\ :\!-\ U}$" が存在する．これは，図 6 に示す遷移規則をみたす最小の二項関係として定義される．ただし，(R2), (R4), (R5) は膜に関する規則であるので省略した． 最も本質的で重要な規則は (R6) である．これは,「ルールと，その左辺にパターンマッチするプロセスが存在したら，その左辺を右辺に書き換えてよい」ということを言っている．

また，この定義を自然にルールセットへと拡張することができる．すなわち，$\exists r \in R.\ G \xrightarrow{r} G'$ が成り立つとき「ルールセット R によって，グラフ G が（1 ステップで）G' に遷移する」といい，単に $G \xrightarrow{R} G'$ と書く．

なお，通常の LMNtal においては，アトムとルールの多重集合であるプロセスが，遷移関係に基づいて**おのずから書き換わっていく**．そのため,「ルールによる作用を受けてグラフが書き換わる」とは考えない．しかしながら，書換えに関する静的な性質について定義および議論をするにあたっては，書換えの主体たるルールと客体たるグラフとは明確に分離されていたほうが都合が良い．よって，本論文では LMNtal の構文・意味論を簡明な形に等価変換して扱っている．

3.3　略記法

より簡潔な記述を可能にするため，次の二つの略記法が認められている．

1. アトムの引数にアトムを書いた場合は，そのアトムの最終引数に繋がっているものと解釈される．つまり，$p(X_1,\ldots,q(Y_1,\ldots,Y_n),\ldots,X_m)$ は，$p(X_1,\ldots,L,\ldots,X_m)$，$q(Y_1,\ldots,Y_n,L)$ と解釈される．ただし，L は使用されていない適当なリンク名とする．例えば，`a(b(c),d)` は `a(B,D),b(C,B),c(C),d(D)` を意味する．
2. 引数リストを省略した場合は引数が 0 個であると解釈される．つまり，p は p`()` の意味である．

4　LMNtal ShapeType

本節では，LMNtal を対象とする静的な型検査手法である LMNtal ShapeType を紹介する．まず，形式的な定義を先に示しておく．なお，以下で "記号" と言うときは，LMNtal のファンクタのことを指す．

定義 1. LMNtal ShapeType における型（以下，これを単に "ShapeType" という）は，3 つ組 (S,P,N) である．ここで，

- $S = t/m$ は，**開始記号**と呼ばれるファンクタ
- P は，**生成規則**と呼ばれるルールの有限集合
- N は，**非終端記号**と呼ばれるファンクタの有限集合

とする．ただし，$S \in N$ とする． ◇

4.1　構文

ShapeType の構文を図 7 に示す．生成規則は，LMNtal のルールとして well-formed である必要がある．生成規則の左辺は 1 つ以上の（コネクタでない）非終端アトムのみからなる．また，LMNtal と同様の略記法（3.3 節）を認める．

`nonterminal{`N`}` は S 以外の非終端記号の宣言であり，無ければ省略して良い．S や N の要素はファンクタであってアトムではないので，本来であれば f/n の形式で記述するべきである．しかし，以下の 2 つの理由から非終端記号はファンクタの形式ではなく自由リンク名を明示したアトムの形式で記述するほうが表記上自然である：

ShapeType
::= `defshape` S `{` P `}` [`nonterminal{` N `}`]

開始記号
S ::= $p(X_1, \ldots, X_m)$

生成規則の有限集合
P ::= $\mathbf{0} \mid [RuleName$ `@@` $]\ A$ `:-` $A \mid P, P$

非終端記号の有限集合
N ::= A

アトムの有限集合
A ::= $\mathbf{0} \mid p(X_1, \ldots, X_m) \mid A, A$

図 7　ShapeType の構文

1. $s(L_1, \ldots, L_n)$ という非終端記号（開始記号を含む）は，そこから生成されるグラフの全体，すなわち
$$\{G \mid s(L_1, \ldots, L_n) \xrightarrow{P}{}^{*} G\}$$
を代表しており，$L_1, \ldots, L_n$ はそのように生成されるグラフ G の自由リンクの名前としても用いられる
2. 代数的データ型（二分木やリスト等）ではデータ構造の根が一つであるのに対して，LMNtal ShapeType が対象とするグラフ構造の根（自由リンク）は一つとは限らず，かつそれらの自由リンクにはそれぞれ異なる役割がある（非可換である）

例えば，1 節で紹介したスキップリストの場合，次のように型定義できる．

```
defshape skiplist(List2,List1){
  p0@@ skiplist(L2,L1) :- nil(L2,L1).
  p1@@ skiplist(L2,L1)
    :- n1(X1,L1), skiplist(L2,X1).
  p2@@ skiplist(L2,L1)
    :- n2(X1,X2,L2,L1), skiplist(X2,X1).
}
```

なお，図 1 (b) で示したスキップリストの各ノードには整数型の要素が繋がっているが，以降の本論文中ではグラフの形状を重視するために，こうしたデータ構造に含まれるデータは省略して考える．例えば，上記の型をもつグラフは図 8 に示すような形状をしている．

以上のように定義された型を，開始記号 $S = t/m$ のアトム名 t と，適当なリンク名 $L_1, \ldots, L_m$ を用いて $t(L_1, \ldots, L_m)$ 型などと呼ぶことにする．ただし，リンク名が重要でないときは開始記号のファンクタを用いて t/m 型（あるいは m 価の t 型）と呼び，誤解のおそれがないときは価数も省略して単に t 型と呼ぶ．

4.2　意味論

型付け関係 “:” を定義するにあたって，補助的な型付け関係 “$\lhd$” を先に定義する．以下では，$L_1, \ldots, L_m$ だけを自由リンクに持つグラフを $G[L_1, \ldots, L_m]$ と書く．また，グラフ G 中で使用されているファンクタ全体の集合を，Funct(G) と表す．

定義 2. 型 $(t/m, P, N)$ について，
$$t(L_1, \ldots, L_m) \xrightarrow{P}{}^{*} G[L_1, \ldots, L_m]$$
であるとき，かつそのときに限り，$G[L_1, \ldots, L_m]$ は $t(L_1, \ldots, L_m)$ 型によって**生成される**といい，$G[L_1, \ldots, L_m] \lhd t(L_1, \ldots, L_m)$ と書く． ◇

これは直観的には，t 型の開始記号から生成規則によって 0 ステップ以上遷移した先のグラフだけが t 型によって生成されることを表す．

次に，型付け関係 “:” を定義する．

定義 3. 型 $(t/m, P, N)$ について，
$$G[L_1, \ldots, L_m] \lhd t(L_1, \ldots, L_m)$$
かつ
$$\mathrm{Funct}(G[L_1, \ldots, L_m]) \cap N = \varnothing$$
であるとき，かつそのときに限り，$G[L_1, \ldots, L_m]$ は $t(L_1, \ldots, L_m)$ 型を**もつ**といい，$G[L_1, \ldots, L_m] : t(L_1, \ldots, L_m)$ と書く． ◇

これは直観的には，t 型によって生成されるグラフのうち，非終端記号を含まないグラフのみが，t 型をもつことを表す．

また，前節で述べた通り，開始記号および（型付けの対象である）グラフの自由リンクの名前と順序の対応関係は重要である．例えば，

`defshape t(X,Y) { t(X,Y) :- a(X,Y) }`,

という型があったときに，`a(X,Y)` $\lhd$ `t(X,Y)` であるが，`a(Y,X)` $\lhd$ `t(X,Y)` ではない．

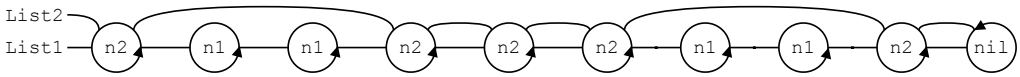

図 8　要素を無視したスキップリスト型グラフの例

4.3　型検査

LMNtal ShapeType には，2 つの基本的な型検査手法がある．一つは，**グラフの型検査**である．これは，与えられたグラフが，与えられた型に属しているかを検査するものである．もう一つは，**ルールの型検査**である．これは，ルールと型が与えられたときに，ルールの適用によって型の構造が破壊されないことを検査するものである．

ルールの型検査について，もう少し形式的に解説する．まず，型 t に関する LMNtal ルール r の型保存性を定義する．

定義 4. ルール r と型 t/m およびリンク列 $L_1, \ldots, L_m$ について，全ての $G : t(L_1, \ldots, L_m)$ が，

$$G \xrightarrow{r} G' \Rightarrow G' : t(L_1, \ldots, L_m)$$

を満たすとき，かつそのときに限り，r は t 型を**保存する**という．　◇

自然言語で表現すると，「t 型をもつ全てのグラフ G は，ルール r を（可能なら）適用した後も，t 型を持っている」という性質である．このルールの型保存性の検査のことを，単にルールの型検査という．

4.4　具体的な検査手法

LMNtal ShapeType においては，生成規則が LMNtal 自身のルールとして記述されていることから，LMNtal を対象言語とするモデル検査器 SLIM [17] を用い，生成規則の逆実行により実際にグラフの型検査・ルールの型検査を行うことができる．

定義 5. LMNtal ルール $r = T :- U$ の**反転** r^{inv} を，$r^{\mathrm{inv}} \triangleq U :- T$ で定める．同様に，LMNtal ルールセット R の反転 R^{inv} を，$R^{\mathrm{inv}} \triangleq \{r^{\mathrm{inv}} \mid r \in R\}$ で定める．　◇

つまり，反転とはルールの左辺と右辺を入れ替えたものである．ここで，$G \xrightarrow{r} G'$ と $G' \xrightarrow{r^{\mathrm{inv}}} G$ が同値であることが LMNtal の遷移規則および構造合同規則から容易に示せる．

グラフの型検査は，開始記号に生成規則を 0 回以上適用して検査対象グラフ G に遷移できるかを検査すればよい．そのため，逆に検査対象グラフ G から出発し，生成規則の反転を 0 回以上適用して開始記号に遷移できるか否かを調べればよい．これは一見すると同じことのようであるが，生成規則の適用により生成される状態空間は（その型をもつグラフの個数が有限でない限り）一般に無限であるので，後者のほうが検証しやすいと言える．

一方で，ルールの型検査はより複雑である．本論文では，文献 [13] や Structured Gamma [3] の手法とは異なる，実装が容易な検査手法を紹介する．この検査手法では，検査対象のルールそれ自体を書き換えていき，**LMNtal ルールを状態とみなして状態空間を生成**する．この考え方は，前提と結論のペアを一つのまとまりとして捉えるという点で，シーケント計算の考え方と近い．

まず，各生成規則 $\alpha :- \beta$ について，$\beta \equiv \beta_1, \beta_2$ かつ β_1 が空でないようなグラフ β_1, β_2 をとる．ここで，状態 $L :- R$ に対し，$L \equiv \beta_1, L'$ であるならば，この状態を $\alpha, L' :- \beta_2, R$ へと書き換えるような書換え規則を作成する．以上のようにして得られる全ての書換え規則をルールセットとし，検査対象ルールを初期状態として状態空間を生成する．こうしてできた状態空間において，初期状態から出発して，左辺が開始記号であるような状態へと到達するような任意の経路上に，必ず reducible（左辺から右辺へと生成規則を 0 回以上適用して遷移可能）な状態が存在することを検証すれば，型保存性を検査できる．

型検査の正当性に関連する詳細な定義と証明は付録にあるが，概略は次の通りである．まず，以上の手順で生成される状態空間を Kripke 構造の形で $\mathcal{S} = (\mathcal{W}, \mathcal{T}, \mathcal{L})$ と定義する．ただし，$\mathcal{W}$ は LMNtal ルール全体の集合であり，$\mathcal{L} : \mathcal{W} \to 2^{\{\mathrm{s},\mathrm{r}\}}$ は状態へのラベル付け関数である．ここで原子命題 s は “start

symbol” の頭文字であり，左辺が開始記号であることを指す．$\mathtt{r}$ は前述の “reducible” を意味する．その上で，以下の定理を示す．ただし，T は検査対象の型，r は検査対象の書換え規則を表す．

定理 1. $\mathcal{S}, r \models \neg \mathtt{s}\,\mathsf{W}\,\mathtt{r}$ ならば，r は T 型を保存する．

ここで，W は “weak until” を意味する時相論理演算子であり，$P\,\mathsf{W}\,Q$ は「Q が成り立つまでずっと P が成り立つ，またはいつでも P が成り立つ（つまり，$(P\,\mathsf{U}\,Q) \vee \mathsf{G}\,P$）」を意味する．よって，ここでの $\mathcal{S}, r \models \neg \mathtt{s}\,\mathsf{W}\,\mathtt{r}$ は，「r から出発して Kripke 構造 $\mathcal{S}$ に従って遷移できる全ての経路上において，reducible な状態にたどり着くまでは左辺が開始記号となることはない．ただし，左辺が開始記号となる状態が存在しない経路では，reducible な状態が存在しなくても良い」ことを表している．

文献 [13] や Structured Gamma [3] における手法では，(i) 検査対象ルールの左辺を基にそれとマッチしうるグラフのパターンを導出するために，必要なグラフを補うことを許して生成規則により逆向きに遷移して状態空間を構築し，(ii) 開始記号に到達するような全ての経路を列挙して，(iii) その経路上で左辺に補ったグラフと同じものを右辺にも補い，開始記号から右辺に遷移できることを確かめる，という手順を踏んでいた．この場合，左辺に補ったグラフを状態空間のエッジ上に記録しておく必要があり，かつ左辺の自由リンクと補ったグラフの自由リンクとの接続関係を，右辺に補う際に適切に再現することが実装上困難であった．実際，文献 [13] でも，実装のためには SLIM に何らかの改造を施す必要がある，とされていた．

一方で，本論文の手法では左辺を基にした逆向きの遷移と，右辺へのグラフの補完とを同時に行っており，かつそれらの手続きは全て LMNtal の上で完結しているので，実装が容易である．

なお，本論文の手法で生成される状態空間はそのままでは一般に有限とならない．そのため，実際の実装においては左辺が同じ状態は同一とみなして状態空間を構築する．ただし，このように左辺の構造合同関係で商を取った状態空間で，初期状態から左辺が開始記号であるような状態へ至る経路上に閉路が出現する場合，その閉路に進入した最初の状態において reducibility の検査を行い，それ以上は遡らない．これにより完全性が幾分損なわれるが，もとよりルールの型検査は決定不可能問題であることが [3] において示されており，かつ [3] においても同様に探索の打ち切りを行っていることから，型検査の能力を損なっていることにはならない．

また，ルール自体を書き換える書換え規則や，左辺が同じ状態を同一とみなしての状態空間構築などは，LMNtal メタインタプリタ [8] の機能を利用すれば実現できる．

4.5 生成規則及び型の文脈自由性

LMNtal ShapeType および 2 節で紹介したグラフを対象とする既存手法では，生成規則の形に次のような制限を加えることでルールの型検査の停止性を担保していた．

定義 6. 生成規則 $p = \alpha$:- β について，α がただ一つのアトムからなり，β がコネクタ以外のアトムを一つ以上含むとき，p は**文脈自由**な生成規則であるという．また，生成規則が全て文脈自由である型を**文脈自由な型**という． ◇

文脈自由な型についてのルールの型検査における状態空間が必ず有限となることは既に分かっているが，これより広い型のクラスについては自明でない．

5 数値制約のある型

ここまでに述べた手法により，二分木や線形リストといった代数的データ型についてはもちろん，根（自由リンク）が複数あるスキップリストや差分リスト，あるいはルートが一つもないリング構造のように，代数的データ型では表現ができないグラフの型も表現することができる．

しかし，高さ n の完全二分木や（平衡している）赤黒木，あるいは長さ n のリストといったような数値制約の付いた型は表現することが難しい．もちろん，文脈自由でない型を許せばこうした型を表現することはできるが，その場合は型検査の停止性が自明でなくなる．

一方で，数値制約が定数であるもの，例えばちょうど高さが 3 の赤黒木は次のように文脈自由な型で表

すことができる．

```
defshape rbtree3(R){
  init@@ rbtree3(R) :- btree3(R).
  bb3@@ btree3(R) :- b(tree2,tree2,R).
  bb2@@ btree2(R) :- b(tree1,tree1,R).
  bb1@@ btree1(R) :- b(tree0,tree0,R).
  bl@@ btree0(R) :- leaf(R).
  tb2@@ tree2(R) :- b(tree1,tree1,R).
  tb1@@ tree1(R) :- b(tree0,tree0,R).
  tr2@@ tree2(R) :- r(btree2,btree2,R).
  tr1@@ tree1(R) :- r(btree1,btree1,R).
  tr0@@ tree0(R) :- r(btree0,btree0,R).
  tl@@ tree0(R) :- leaf(R).
} nonterminal {
  rbtree3(R), btree3(R), btree2(R),
  btree1(R), btree0(R),
  tree2(R), tree1(R), tree0(R)
}
```

ここで登場した非終端記号のうち，btreeN は高さ N かつ根が黒である赤黒木，treeN は黒高さ N の赤黒木（根は赤でも黒でもよい）を表し，終端記号の r は赤ノード，b は黒ノード，leaf は葉ノードを表す．なお，黒高さとは根から葉に至るまでの経路上に出現する黒ノードの数である．赤黒木の性質上，黒高さは葉によらず一定であることが要求される．

この定義において，bb3 から bb1，tb2 から tb1，tr2 から tr0 はそれぞれほとんど同一の生成規則であり，非終端記号の添字が変わっているにすぎない．よって，次のように簡略化することができると考えられる．

```
defshape rbtree(R){
  init@@ rbtree(R) :- btree($n,R).
  bb@@ btree($n,R)
    :- $m = $n-1 | b(tree($m),tree($m),R).
  bl@@ btree(0,R) :- leaf(R).
  tb@@ tree($n,R)
    :- $m = $n-1 | b(tree($m),tree($m),R).
  tr@@ tree($n,R)
    :- r(btree($n),btree($n),R).
  tl@@ tree(0,R) :- leaf(R).
} nonterminal {
  rbtree(R), btree($n,R), tree($n,R)
}
```

ここで新たに $n といった記法を導入した．これは型付きプロセス文脈 [18] と呼ばれる LMNtal の拡張機能に由来するものである．本来の型付きプロセス文脈はガード部（:- と | に挟まれた部分）で指定される制約を満たす任意のグラフにマッチする機構である．しかし，ここでは全ての型付きプロセス文脈は自然数 †1 にのみマッチするものとする．この拡張では，生成規則の左辺に非終端記号 1 つの他に，任意個の自然数や型付きプロセス文脈の出現を許す．これにより「生成規則の左辺はただ一つのアトムのみからなる」という文脈自由性の前提を崩しているが，この自然数は元々非終端記号の添字の役割を担っており，非終端記号の付属物であるとみなすことができるので，本質的には文脈自由である．なおこうした添字付きの非終端記号という着想は，形式言語理論におけるインデックス文法（indexed grammar [1]）から得たものである．

これらのプロセス文脈は任意の自然数値をとる変数のようなものである．よって，無限通りの値をとることが考えられ，状態空間が爆発する危険がある．そのため，状態空間構築においては非終端記号の添字だけの違いは無視し，同一状態とみなすこととする．これにより，見かけ上は添字を導入する前と同一の状態空間が生成されるため，型検査の停止性が保証されるが，前述のちょうど高さが 3 の赤黒木の例のように文脈自由な型の範囲で表現された型の場合と比べると，複数の状態が一つの状態に抽象されるため，4.4 節で述べたのと同様に，型検査の完全性を少し損なうこととなる．しかしながら，添字を導入することは型定義の記述の簡潔性からも重要であり，かつ添字を導入せずに任意の高さの赤黒木を表現することは不可能であるから，全体としては型の表現力は向上している．

5.1 型検査の手順

ここでは，例として以下のルールの型検査を行う．

```
b(S,leaf,R) :- b(S,r(leaf,leaf),R)
```

これは赤黒木へのノードの挿入のうち最も単純な（木の回転を必要としない）例である．まず，生成規則 tl を逆向きに使って，左辺の leaf を tree に戻す．

†1 LMNtal における数は，その数をアトム名に持つ 1 価のアトムとして表現される．

```
b(S,tree(0),R) :- b(S,r(leaf,leaf),R)
```

続いて，自由リンク S の先に tree(0) を補いつつ，生成規則 tb を逆向きに使って，左辺全体を tree に戻す．

```
tree(1,R) :- b(tree(0),r(leaf,leaf),R)
```

このルールは reducible である．実際，次の手順で左辺から右辺へと遷移できる．

tree(1,R)
$\xrightarrow{\texttt{tb}}$ b(tree(0),tree(0),R)
$\xrightarrow{\texttt{tr}}$ b(tree(0),r(btree(0),btree(0)),R)
$\xrightarrow{\texttt{bl}}$ b(tree(0),r(leaf,btree(0)),R)
$\xrightarrow{\texttt{bl}}$ b(tree(0),r(leaf,leaf),R)

また，生成規則 tb の代わりに bb を逆向きに使うと，次を得る．

```
btree(1,R) :- b(tree(0),r(leaf,leaf),R)
```

これもまた reducible である．実際の遷移経路は，先ほどのものとほとんど同一であるので割愛する．これ以外に初期状態から bl を使って戻ることもできるが，その先は行き詰まり状態であって左辺が開始記号となることはないので，この経路は考えなくて良い．以上により，ルールの型保存性が示された．

補うグラフに添字付き非終端記号が含まれる場合は，その添字に任意性があるため上記のように単純ではない．例えば，次のルールを考える．

```
b(A,B,R) :- b(B,A,R)
```

これは単に黒ノードの左右の部分木を入れ替えるルールである．あまり実用的でなくかつ型保存性が直感的には明らかであるが，この型検査にあたっては添字の取扱いが難しい．まず自由リンク A と B の先に黒高さ n の tree を補いつつ，生成規則 bb を逆向きに使い，次を得る．

```
btree(n+1,R) :- b(tree(n),tree(n),R)
```

このルールは生成規則 bb により reducible である．同様に，初期状態から生成規則 tb を使った場合以下を得るが，これも生成規則 tb により reducible である．

```
tree(n+1,R) :- b(tree(n),tree(n),R)
```

以上によりルールの型保存性が検証できた．

以上のような，具体値が必ずしも定まっていないが制約条件が存在するような数を含むグラフに対する操作は，一見すると自然なように思われる．しかし，こうした不定数を記号的に扱う機能は現在の LMNtal の組み込み機能には存在しない．そのため実装にあたっては，現在の型検査実装に用いている LMNtal メタインタプリタを拡張して，数値に関する記号制約を扱えるようにするなどの工夫が必要である．

6 まとめと今後の課題

本論文では，型付きプロセス文脈により添字付きの非終端記号を表現することで，数値制約付きの型を対象とした書換え規則について型検査を行う手法について述べた．また，ルールの型検査の実装をより容易にする手法についても述べ，その正当性を示した．

今後の課題としては，型付きプロセス文脈に対応したプロトタイプの実装および応用が挙げられる．既にルールの型検査のプロトタイプは実装済みであり，4.3 節で述べた手法により初期グラフの生成を除く全部分を LMNtal で実装した．しかし，これは 5 節で述べた型付きプロセス文脈による拡張には未対応である．この実装ができれば，[14] で述べた関数的アトムの型検査と組み合わせることにより，赤黒木への挿入操作のように，数値制約を伴う型を対象とした，複数ステップを要し，あるいは型の変化を伴うようなグラフ操作についても型検査が可能になると考えられる．

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] Aho, A. V.: Indexed grammars – An extension of context free grammars, *8th Annual Symposium on Switching and Automata Theory (SWAT 1967)*, 1967, pp. 21–31.
[2] Fradet, P. and Métayer, D. L.: Shape types, *Proc. POPL'97*, ACM, 1997, pp. 27–39.

[3] Fradet, P. and Métayer, D. L.: Structured Gamma, *Science of Computer Programming*, Vol. 31, No. 2(1998), pp. 263–289.
[4] KAHRS, S.: Red-black trees with types, *Journal of Functional Programming*, Vol. 11, No. 4(2001), pp. 425–432.
[5] Klarlund, N. and Schwartzbach, M. I.: Graph Types, *Proc. POPL'93*, 1993, pp. 196–205.
[6] Pugh, W.: Skip lists: A probabilistic alternative to balanced trees, *Commun. ACM*, Vol. 33, No. 6(1990), pp. 668–676.
[7] Rondon, P. M., Kawaguci, M., and Jhala, R.: Liquid Types, *Proc. PLDI 2008*, ACM, 2008, pp. 159–169.
[8] Tsunekawa, Y., Tomioka, T., and Ueda, K.: Implementation of LMNtal Model Checkers: A Metaprogramming Approach, *Journal of Object Technology*, Vol. 17, No. 1(2018).
[9] Ueda, K.: Encoding the Pure Lambda Calculus into Hierarchical Graph Rewriting, *Proc. RTA2008*, Voronkov, A.(ed.), LNCS, Vol. 5117, Springer, 2008, pp. 392–408.
[10] Ueda, K.: Towards a Substrate Framework of Computation, *Concurrent Objects and Beyond*, Agha, G. et al.(eds.), LNCS, Vol. 8665, Springer, 2014, pp. 341–366.
[11] 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.
[12] Vazou, N., Seidel, E., Jhala, R., Vytiniotis, D., and Peyton Jones, S.: Refinement Types For Haskell, *ACM SIGPLAN Notices*, Vol. 49(2014).
[13] 吉元佑介, 上田和紀: グラフ書換え系における静的グラフ型検査, 日本ソフトウェア科学会第 *32* 回大会 *(2015* 年度*)* 講演論文集, 2015.
[14] 山本直輝, 上田和紀: グラフ書換え言語におけるグラフ操作の静的型検査, 日本ソフトウェア科学会第 *36* 回大会 *(2019* 年度*)* 講演論文集, 2019.
[15] 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.
[16] 上田和紀, 加藤紀夫: 言語モデル LMNtal, コンピュータ ソフトウェア, Vol. 21, No. 2(2004), pp. 126–142.
[17] 石川力, 堀泰祐, 村山敬, 岡部亮, 上田和紀: 軽量な LMNtal 実行時処理系 SLIM の設計と実装, 情報処理学会第 *70* 回全国大会講演論文集, (2008), pp. 153–154.
[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.

A ルールの型検査の正当性証明

LMNtal ルール $\alpha_1 :- \beta_1$ と $\alpha_2 :- \beta_2$ との間の遷移関係 $\mathcal{T}$ を，

$$\begin{aligned}\alpha_1 :- \beta_1 \xrightarrow{\mathcal{T}} \alpha_2 :- \beta_2 \\ \text{iff} \quad \exists \alpha_p :- \beta_p \in P.\ \exists \gamma, \gamma'. \\ \alpha_2 \xrightarrow{\alpha_p :- \beta_p} \alpha_1, \gamma \ \wedge\ \beta_2 \equiv \beta_1, \gamma \\ \wedge\ \beta_p \equiv \gamma, \gamma' \ \wedge\ \gamma' \not\equiv \mathbf{0}\end{aligned}$$

で定める．また，$\mathcal{W}$ を LMNtal ルール全体の集合とし，ラベル付け関数 $\mathcal{L} : \mathcal{W} \to 2^{\{\mathtt{s},\mathtt{r}\}}$ を次で定める．なお，原子命題 $\mathtt{s}$ は “start symbol”，$\mathtt{r}$ は “reducible” を意味する．ただし，ここでの $\equiv$ は自由リンク名の違いを無視する．

$$\begin{aligned}\mathtt{s} \in \mathcal{L}(\alpha :- \beta) \quad \text{iff} \quad \alpha \equiv T \\ \mathtt{r} \in \mathcal{L}(\alpha :- \beta) \quad \text{iff} \quad \alpha \xrightarrow{P}{}^{*}\ \beta\end{aligned}$$

このとき，Kripke 構造 $\mathcal{S} = (\mathcal{W}, \mathcal{T}, \mathcal{L})$ を考える．この Kripke 構造の意味する状態空間は，4.3 節で述べた手法により構築される状態空間と同一である．以下，$\mathcal{S}, r \models \neg \mathtt{s}\,\mathsf{W}\,\mathtt{r}$ が型保存性の十分条件であることを示す．

補題 1. $\alpha :- \beta$ が reducible で，$\alpha :- \beta \xrightarrow{\mathcal{T}} \alpha' :- \beta'$ なら，$\alpha' :- \beta'$ も reducible.

Proof. 仮定より $\alpha \xrightarrow{P}{}^{*}\ \beta$ である．また，$\mathcal{T}$ の定義から $\exists \gamma.\ \alpha' \xrightarrow{P} \alpha, \gamma\beta' \equiv \beta, \gamma$ である．よって，$\alpha' \xrightarrow{P} \alpha, \gamma \xrightarrow{P}{}^{*}\ \beta, \gamma \equiv \beta'$，すなわち $\alpha' \xrightarrow{P}{}^{*}\ \beta'$ である． □

補題 2. $P, Q \equiv R, S$ のとき，$P \equiv A_1, A_2$，$Q \equiv A_3, A_4$，$R \equiv A_1, A_3$，$S \equiv A_2, A_4$ を満たす A_1, A_2, A_3, A_4 が存在する．

Proof. 構造合同規則より明らか. □

補題 3. $P \xrightarrow{\alpha :- \beta} Q$ のとき，$P \equiv C, \alpha$，$Q \equiv C, \beta$ を満たす C が存在する．

Proof. 構造合同規則と遷移規則から明らか. □

補題 4. $X \xrightarrow{p} Y \equiv \alpha, C$ $(p = \alpha_p :- \beta_p \in P)$ のとき，以下のいずれかが成り立つ．

- $\forall \beta.\ \exists \alpha', \beta', C_1, C_2.\ \alpha :- \beta \xrightarrow{\mathcal{T}} \alpha' :- \beta'$

$\wedge\, C \equiv C_1,C_2 \wedge X \equiv \alpha',C_2 \wedge \beta' \equiv \beta,C_1$

- $\exists C'.\ X \equiv \alpha,C' \wedge C' \xrightarrow{p} C$

Proof. $X \xrightarrow{p} Y$ と補題 3 より，$X \equiv \alpha_p,C_p,\ Y \equiv \beta_p,C_p$ なる C_p が存在する．$Y \equiv \beta_p,C_p \equiv \alpha,C$ なので，補題 2 より $C \equiv C_1,C_2,\ \alpha \equiv C_3,C_4,\ \beta_p \equiv C_1,C_3,\ C_p \equiv C_2,C_4$ なる C_1,C_2,C_3,C_4 が存在する．

$C_3 \not\equiv \mathbf{0}$ の場合

ここで $\alpha' = \alpha_p,C_4,\ \beta' = \beta,C_1$ とおく．$\alpha' \equiv \alpha_p,C_4 \xrightarrow{p} \beta_p,C_4 \equiv C_1,C_3,C_4 \equiv \alpha,C_1$ なので，C_1 を $\mathcal{T}$ の定義における γ，C_3 を γ' と見做すと，α :- $\beta \xrightarrow{\mathcal{T}} \alpha'$:- β' である．また，$X \equiv \alpha_p,C_p \equiv \alpha_p,C_2,C_4 \equiv \alpha',C_2$ である．

$C_3 \equiv \mathbf{0}$ の場合

$C_1 \equiv \beta_p$ なので，$C \equiv \beta_p,C_2$ である．よって，$\alpha_p,C_2 \xrightarrow{p} C$ となる．ここで $C' \equiv \alpha_p,C_2$ とおくと，$C' \xrightarrow{p} C$ である．一方，$\alpha \equiv C_4$ なので，$C_p \equiv C_2,\alpha$ である．よって，$X \equiv \alpha_p,C_p \equiv \alpha_p,C_2,\alpha \equiv \alpha,C'$ である． □

定理 2. $\mathcal{S},r \models \neg\mathtt{s}\,\mathsf{W}\,\mathtt{r}$ ならば，r は T 型を保存する．

Proof. $G \lhd T$ および $G \xrightarrow{r} G'$ ならびに $\mathcal{S},r \models \neg\mathtt{s}\,\mathsf{W}\,\mathtt{r}$ を仮定して $G' \lhd T$ を示す．

$G \lhd T$ のとき，非負整数 n とグラフ $X_0,\ldots,X_n$ $(X_0 = G,\ X_n = T)$ および $p_0,\ldots,p_{n-1} \in P$ があって，$\forall i < n.\ X_{i+1} \xrightarrow{p_i} X_i$ である．

$G \xrightarrow{r} G'$ なので，$r = \alpha$:- β とすると，$G \equiv \alpha,C,\ G' \equiv \beta,C$ なる $C \in \mathcal{G}(N \cup \Sigma)$ が存在する．

次に，$i = 0,\ldots,n-1$ について，$X_i \equiv \alpha_i,C_i$ ならば以下を満たす $\alpha_{i+1},\beta_{i+1},C_{i+1}$ が存在することを示す．

$$\begin{aligned}&\alpha_i \text{ :- } \beta_i \xrightarrow{\mathcal{T}}{}^{*}\ \alpha_{i+1} \text{ :- } \beta_{i+1}\\ \wedge\ &X_{i+1} \equiv \alpha_{i+1},C_{i+1}\\ \wedge\ &\beta_{i+1},C_{i+1} \xrightarrow{P}{}^{*}\ \beta_i,C_i\end{aligned}$$

$X_{i+1} \xrightarrow{p_i} X_i$ と補題 4 より，以下のいずれかが成り立つ．

1. $\exists \alpha_{i+1},\beta_{i+1},C_i',C_{i+1}.$
   α_i :- $\beta_i \xrightarrow{\mathcal{T}} \alpha_{i+1}$:- $\beta_{i+1} \wedge C_i \equiv C_i',C_{i+1}$
   $\wedge\, X_{i+1} \equiv \alpha_{i+1},C_{i+1} \wedge \beta_{i+1} \equiv \beta_i,C_i'$
2. $\exists C_{i+1}.\ X_{i+1} \equiv \alpha_i,C_{i+1} \wedge C_{i+1} \xrightarrow{p_i} C_i$

1. の場合，$\beta_{i+1},C_{i+1} \equiv \beta_i,C_i',C_{i+1} \equiv \beta_i,C_i$ となることから明らか．一方，2. の場合は $\alpha_{i+1} = \alpha_i,\ \beta_{i+1} = \beta_i$ とおけば明らか．

以上から，α :- $\beta \xrightarrow{\mathcal{T}}{}^{*}\ \alpha_n$:- β_n かつ $T \equiv \alpha_n,C_n$ かつ $\beta_n,C_n \xrightarrow{P}{}^{*}\ \beta,C$ なる α_n,β_n,C_n が存在する．ここで，T は開始記号の（自己ループを含まない）アトム一つからなるので，$C_n \equiv \mathbf{0}$ であり，$T \equiv \alpha_n$ である．よって，$\mathtt{s} \in \mathcal{L}(\alpha_n$:- $\beta_n)$ である．

$r \xrightarrow{\mathcal{T}}{}^{*}\ \alpha_n$:- β_n と $\mathcal{S},r \models \neg\mathtt{s}\,\mathsf{W}\,\mathtt{r}$ および補題 1 より，$\mathtt{r} \in \mathcal{L}(\alpha_n$:- $\beta_n)$ でもある．つまり $\alpha_n \xrightarrow{P}{}^{*}\ \beta_n$ である．よって，$T \equiv \alpha_n \xrightarrow{P}{}^{*}\ \beta_n \xrightarrow{P}{}^{*}\ \beta,C \equiv G'$ なので，$G' \lhd T$ となる． □