2A-05

グラフ書き換え系における token passing を用いたグラフ型検査

山田 啓太† 山本 直輝† 上田 和紀†

早稲田大学 基幹理工学研究科 情報理工・情報通信専攻

1 はじめに

グラフ書き換え言語 LMNtal [1] は, 様々なグラフ構造を簡潔に扱うことができるプログラミング言語である. LMNtal への型概念の導入法として, グラフの生成規則を用いて型付けを行う LMNtal ShapeType [2] が提案されている. LMNtal ShapeType におけるグラフ型は, 素朴には生成規則の逆実行により検査可能であるが, 一般には効率が十分ではない. 本論文では, この課題を解決するため, token passing を用いたグラフ型検査手法を提案し, グラフの生成規則からの変換アルゴリズムを示す. 本手法に基づく型検査により, 従来の型検査と比べ, 状態空間が削減されることを確認した.

2 LMNtal

LMNtal では, アトムとリンクで構成されるグラフの書き換えを繰り返すことによって処理を進める. LMNtal ではさらに, 膜と呼ばれる階層構造が利用でき, 多彩なデータ構造の表現が可能である. LMNtal では, グラフの書き換えはルールと呼ばれる書き換え規則によって表現する. ルールは, グラフの部分構造を表す左辺と右辺からなり, 書き換え対象となるグラフとルールの左辺のグラフ構造がパターンマッチした場合, その部分を右辺のグラフに置き換える形で書き換えが進行する.

LMNtal の実行時処理系として, 一般に SLIM [3] が用いられる. SLIM は, 通常のプログラム実行に加え, 全ての書き換え順序を試みて状態空間を出力する非決定実行の機能を併せ持つ.

3 LMNtal ShapeType

LMNtal ShapeType [2] は, Shape Types [4] を基に設計された, LMNtal グラフを対象とする静的型検査手法である. LMNtal ShapeType では, 図 1 のスキップリストの例が示すように, グラフの型を LMNtal ルールの形で記述された生成規則によって定義する. 生成規則によって書き換えられるアトムを**非終端アトム**, 生成された後はそれ以上書き換えられないアトムを**終端アトム**と呼ぶ.

図 1 内の点線で表されるような, 対象グラフ外部との接続部

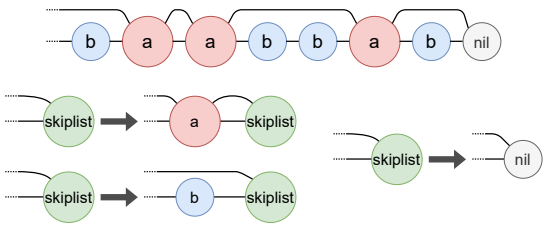


図 1: スキップリスト（上）とその型定義（下）

Token-passing-style graph type checking in graph rewriting systems

† Keita Yamada, Naoki Yamamoto and Kazunori Ueda, Waseda University

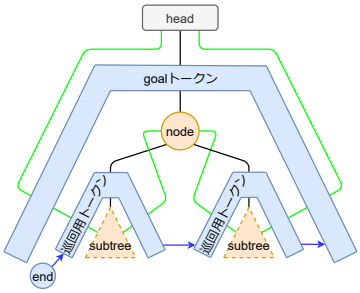


図 2: トークン配置例（threaded tree 型）

分にあたるリンクを**自由リンク**と呼ぶ. LMNtal ShapeType の特徴は, 複数本の自由リンク（根）をもつグラフ構造を扱うことができる点である.

LMNtal ShapeType におけるグラフの型所属性は, 素朴には型定義である生成規則の非決定的な逆適用により検査可能であるが, 状態空間爆発を招く例がある点が課題となっていた.

4 Token passing を用いたグラフ型検査

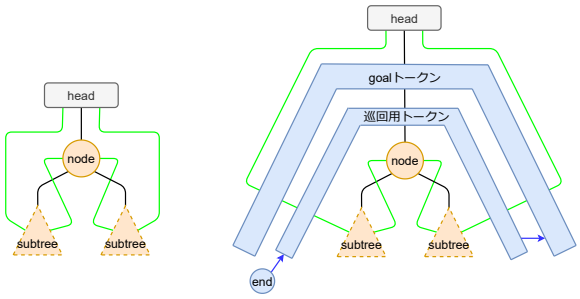
Token passing [5] は, 対象のグラフ構造内をトークンと呼ばれる要素に巡回させ, そのトークンを目印にしてグラフの解析等を行う手法である. トークンの巡回ルールは, トークンが出会ったグラフ要素ごとに設定される. したがって, 同じグラフ要素が複数現れる再帰的グラフ構造の解析に適しており, 処理の流れも明瞭であるという利点がある.

また, トークン自体は対象とするグラフから独立した要素であり, 対象グラフに影響せず自由にデータを持たせることができる. そのため, 解析に必要なデータを持ち歩きながらトークンを巡回させることで, より高度なグラフ解析にも利用できる.

4.1 扱うトークンの種類

本論文では token passing 手法を基に, LMNtal ShapeType における新たなグラフ型検査手法を提案する.

本手法では, 図 2 に示す配置例のように, グラフ内を移動する**巡回用トークン**と, 巡回用トークンの出発地点に配置される**goal トークン**の二種類のトークンを用いる. 巡回用トークンは, 対象とする型定義に現れる非終端アトムごとに対応したものを用意し, トークン名によって識別できるようにする. 巡回用トークンはグラフ内に同時に複数個存在でき, 図 2 にて矢印で表される**帰還用リンク**と呼ばれるリンクによって一列に接続されている. そのうち, end アトムから伸びる帰還用リンクを持つ巡回用トークンのみがグラフ内を動くことができる. このような巡回用トークンを**アクティブな**巡回用トークンと呼ぶ. 一方で, goal トークンは型検査中は常に一つしか存在せず, 配置されてから移動等はしない. グラフの型検査が進み end アトムが goal トークンに直接接続された場合, 型検査は成功となる. 本手法のトークンは通常のトークンとは異なり, グラフ内の複数本のリンクを保持しつつグラフ内を移動する点が特徴である.



(a) 入力グラフ　　**(b)** 開始処理後

図 3: 入力グラフと開始処理後の状態（threaded tree）

4.2　対象となるグラフ型

本手法を適用するために，グラフ型に以下の制約を課す．

1. 1 本以上の自由リンクを持つ連結なグラフを生成すること．
2. 文脈自由な生成規則のみを含むこと．
3. 複数の非終端アトムを生成する生成規則について，非終端アトム間を直接結ぶリンクが生成されないこと．
4. 生成規則右辺の全ての終端アトムが，終端アトムのみを辿っていずれかの自由リンクへ至る経路を持つこと．

これらの制約を満たすグラフ型には，リスト型や木構造型などの代数的データ型や，末尾が終端されておらず，両端が自由リンクとなっている差分リスト型などに加え，共有や循環などの構造をもつ多くの実用的データ型も含まれる．

4.3　型検査プログラムの導出

型検査プログラムは，開始処理として，図 3 に例示するように，入力グラフの内部に，アクティブな巡回用トークンおよび goal トークンが帰還用リンクによって繋がれた構造を埋め込む．開始処理の終了後，グラフ型の生成規則から変換される巡回ルールに従ってトークンを巡回させることで型検査を進める．

生成規則から型検査ルールへの変換アルゴリズムは，非終端アトムを生成する生成規則と，生成しない生成規則によって異なる．これらを図 4 に示す．この変換アルゴリズムを，その型の定義を構成する全ての生成規則に適用し，生成された型検査ルールを入力グラフに対して適用する．end アトムと goal アトムが直接繋がった状態が生成されたら，入力グラフがその型に含まれることになる．

5　実行結果

本手法の適用例として，LMNtal ShapeType で定義された threaded tree 型を本手法にしたがって型検査した結果と，同じ入力グラフを従来の非決定逆実行によって型検査した結果を図 5 に示す．入力グラフにはあらかじめ threaded tree 型に属すことが分かっているものを用いた．どちらの手法でも，状態遷移の終点として正しく入力グラフを受理する結果が得られる．従来手法では途中状態が爆発している一方で，本手法は最終状態まで決定的に遷移が進み，状態数を抑えられている．

6　まとめ

LMNtal ShapeType におけるグラフの型検査について，token passing アルゴリズムを用いた新たな手法を提案し，グラフ型定義から型検査プログラムを導出する方法を示した．今後は，グラフの形状に関する数値制約を含むより強力なグラフ型に本手法を一般に適用する方法の確立が検討課題である．

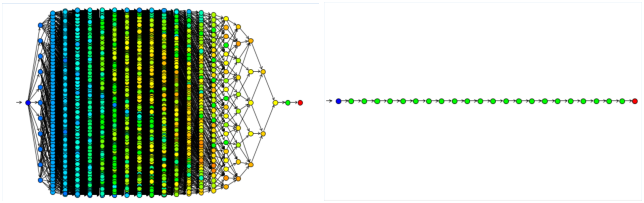
非終端アトムを生成する生成規則の場合

1. 生成規則右辺から非終端アトムを消したグラフ g_{pre} を生成する．
2. 右辺で非終端アトムに繋がるものも含め，左辺の非終端アトム a が持つ全ての自由リンクを束ねた，a に対応するアクティブな巡回用トークンを g_{pre} に加える．こうしてできたグラフ g_{left} を型検査プログラムの左辺とする．
3. 生成規則右辺の非終端アトムに，任意の優先度を付ける．
4. g_{pre} について，生成規則右辺の各非終端アトムが持っていたリンクを束ねる形で，各非終端アトムに対応する巡回用トークンを配置する．
5. 配置した巡回用トークン間を，優先度の高い方から低い方に辿るように，帰還用リンクで接続する．優先度が最高の巡回用トークンには，end アトムからの帰還用リンクを付ける．優先度が最低の巡回用トークンからは，g_{left} 内に設置された巡回用トークンが持っていた帰還用リンクを伸ばす．このグラフを型検査プログラムの右辺とする．

非終端アトムを生成しない生成規則の場合

1. 生成規則右辺のグラフに，全ての自由リンクを束ねるような形で，アクティブな巡回用トークンを配置する．
2. 1 で生成したグラフを型検査ルールの左辺として，巡回用トークンから伸びる帰還用リンクに end トークンを繋ぎ変える処理を実装する．この時，巡回用トークンは削除し，元のグラフ構造に戻すようにリンクを接続し直す．

図 4: 変換アルゴリズム



(a) 従来手法（非決定逆実行）　　**(b)** 本手法（token passing）

図 5: threaded tree 型の型検査実行結果の比較

謝辞 本研究の一部は科研費 23K11057 の補助を得て実施した．

参考文献

[1] 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, pp. 126–142, 2004.

[2] Naoki Yamamoto and Kazunori Ueda. Engineering grammar-based type checking for graph rewriting languages. *IEEE Access*, Vol. 10, pp. 114612–114628, 2022.

[3] 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，pp. 153–154, 2008.

[4] P. Fradet and D. L. Metayer. Shape types. In *POPL 1997*, pp. 27–39, 1997.

[5] François-Régis Sinot. Call-by-name and call-by-value as token-passing interaction nets. In *TLCA 2005*, Vol. 3461 of *LNCS*, pp. 386–400, 2005.