

遅延属性文法¹

3U-7

松田裕幸²

東京工業大学総合情報処理センター

1 はじめに

属性文法がコンパイラの仕様記述以外に広く応用されはじめるにつれ [2], それまで実用的クラスとしてはもっとも大きなクラスと考えられてきた絶対非循環属性文法の枠内では扱えない問題が多く登場するようになってきた。特に、属性同士の依存関係に循環 (circularity) を許すことは、データフロー解析 [3] や VLSI 設計 [7], あるいは属性文法自身をプログラムとして直接実行する文法プログラミング [9] 等において必須の条件になっている。

本稿では、すでに報告がなされている属性間の依存関係に循環を許す循環 (circular) 属性文法 (CAG) [3, 7] に対し、遅延属性文法 (LAG) なる新しい文法クラスを提案する。LAG の最大の特徴は、属性式の評価においてその引数にあたる属性の値が事前にすべて定義されている必要がない (laziness) という点にある。

このことは、これまで循環属性文法で試みられてきた様々な評価器実現アルゴリズムに対し、(属性文法形式を関数型の枠組みに変換することで) これまで遅延関数型言語の世界で確立されてきたグラフィダクション [6] の技術がそのまま使えることを意味する。当然のごとく、この技術は循環を含まない絶対非循環の文法クラスに対してもそのまま適用できる。さらに、属性の値に関し遅延を許すということは、属性式の値が確定する範囲において未定義な属性も許すことをも意味し、これまで属性文法で当然と考えられてきた属性値に関する基本的立場を大きく変えるものとなる。

2 定義

属性式の左辺の属性は右辺の属性に依存する (depend on) という。左辺の属性値を計算するにあたり右辺の属性値がすでに確定している時、この式は強制評価 (eager evaluation) されるといい、実際にその値を必要とするまで式の評価が遅延される時、この式は遅延評価 (lazy evaluation) されるといい。さらに遅延評価においては、永久に値が確定しない未定義属性 (undefined attributes) も許される。

属性同士の依存関係は属性依存グラフを形成する。属性グラフ上のある属性に対し、直接、間接的に自分自身に依存するパスが存在する場合、このグラフには循環属性 (circular attributes) が存在するといい、こうした依存関係を持った文法を循環属性文法と呼ぶ。依存関係がその評価において単調 (monotonic) な場合には、評価値に最小不動点が存在し、値が求まる。一般に、循環属性に対し解は存在しない。

文法規則左辺の合成属性の値が左辺の相続属性および右辺の合成属性のみから決定でき、かつ、右辺非終端記号 N_i の相続属性値が左辺の相続属性および右辺非終端記号 $N_j, 1 < j < i-1$ の相続属性のみから決定される時、この評価を LR 順 (Left-to-Right order) と呼ぶ。

属性付き解析木を再帰下降に評価するとして、属性間の依存関係を次の 3 種類に分ける。1) LR 順 (LINEAR), 2) LR 順ではないが、循環をつくらない。すなわち依存グラフにおいて依存関係に関し一筆書きが可能 (STROKE), 3) 不動点を持つ循環をつくる (CIRCULAR)。

3 遅延属性文法

すべての属性に対し遅延評価を行う文法を遅延属性文法 (LAG) と呼ぶ。遅延評価は delayed evaluation と呼ばれることがあるが、これは従来 call-by-name と呼ばれたものと同じで、ここでいう lazy evaluation のよ

¹Lazy Attribute Grammars²Hirokyu Matsuda, Tokyo Institute of Technology

うに、評価グラフにおいて同一項 (term) に対する共有は起きない。共有が起きるということは、一度評価した値は再度評価されないことを意味する。

LAG は、すべての非循環属性文法および最小不動点を持つ循環属性文法を含む。そして、上記に述べた3種類の属性依存関係 (LINEAR, STROKE, CIRCULAR) を持つ属性文法クラスに対し、AG と、LL(1) あるいは ALL(1)[8] を組み合わせることで、構文解析時に属性付き木を実際に作ることなく、ルート属性を求めることができる。

さらに、定義されない属性 $\perp$ があっても、評価の結果に関与しない時には計算は定義され、解が求まる：

$$\begin{aligned} X \uparrow a &= \text{head}(Y \uparrow b) \\ Y \uparrow b &= \text{cons}(1, Z \uparrow c) \\ Z \uparrow c &= \perp \end{aligned}$$

これは従来の属性文法の評価方式では許されない。

4 さいごに

現時点では、LAG に対する評価器は、LAG を完全遅延関数型の枠組みに置き換え、属性に関する依存グラフをグラフリダクションによって計算するアイデア [6] を採用している。この際、各非終端記号に付随するすべての属性は、相続属性から合成属性への関数 (これ自身合成属性) に変換される。当然、この関数属性の引き数は遅延評価される。評価のイメージは直観的には構文解析中は関数属性の呼出しに対応する仮想の関数呼出し木が形成され、実際の評価が始まる (属性値を求める) につれ、この木が縮退して行く。これはリダクションの過程に相当する。

ALL(1) は LR(1) と同等の能力を持つことが知られており [1]、これと LAG と組み合わせることは実用的面から見て実質的にはほぼ最大のクラスを手に入れたことになる。この組み合わせをベースにして、属性文法をプログラムとみなす研究も行われている [4, 9]。

参考文献

- [1] op den Akker, R. etc.: Attribute Evaluation and Parsing, in *Attribute Grammars, Applications and Systems*, Lecture Notes in Computer Science 545, Springer-Verlag, 1991, pp.187-214.
- [2] Deransart, P., Jourdan, M. and Lorho, B.: *Attribute Grammars: Definitions, Systems and Bibliography*, Lecture Notes in Computer Science 323, Springer-Verlag, 1988.
- [3] Farrow, R.: Automatic Generation of Fixed-Point-Finding Evaluators for Circular, but Well-Defined, Attribute Grammars, in *Proc. of the 1986 Symp. on Compiler Construction, SIGPLAN Not.(ACM)*, Vol.21, No.7 (1986), pp.85-98.
- [4] Frost, R. A.: Guarded Attribute Grammars, *Softw. Pract. Exper.*, Vol.23, No.10 (1993), pp.1139-1156.
- [5] Johnsson, T.: Attribute Grammars as a Functional Programming Paradigm, in *Functional Programming Languages and Computer Architecture*, G. Kahn(Ed.), Lecture Notes in Computer Science 274, Springer-Verlag, 1987, pp.154-173.
- [6] Jones, S.L.P.: *The Implementaion of Functional Programming Languages*, Prentice-Hall, 1987.
- [7] Jones, L.G.: Efficient Evaluation of Circular Attribute Grammars, *ACM TOPLAS*, Vol.12, No.3 (1990), pp.429-462.
- [8] Milton, D.R. and Fischer, C. N.: LL(k) Parsing for Attribute Grammars, in *6th ICALP*, H.A. Maurer(Ed), Lecture Notes in Computer Science 71, Springer-Verlag, 1971, pp.422-430.
- [9] 松田裕幸: 文法プログラミング, コンピュータソフトウェア, Vol.10, No.6 (1993), pp.35-53.