

コンパイラにおける字句解析・構文解析過程の視覚化*

リサーチ 9

丹羽 直輝[†] 楠目 勝利[‡] 佐々 政孝[§]東京工業大学 情報理工学研究科 数理・計算科学専攻[¶]

1 はじめに

コンパイラに関する研究は、これまで各フェーズに対する新しいアルゴリズムの開発や定式化法の確立などが中心となっており、コンパイラの動きをわかりやすく視覚化してみせる学習者・開発者教育としての研究はあまりおこなわれていないように思われる。

そこで、コンパイラの各フェーズの動作やその協調作業を視覚化するための第一歩として、字句解析器・構文解析器の動作を視覚化するシステムを実現した。これにより、開発者は自分が作成した字句・構文解析器の動作を視覚的に確認することができる。また教育者が講義などにおいて本システムを利用することにより、学習者の理解を助けることが容易となる。

2 本システムの特徴

今回実現したシステムの特徴は、1. 対象とするプログラム言語が自由に選べる、2. システムの利用が容易である、3. 実行時環境にできるだけ依存しない、の3点である。本システムでは1を実現するために公開ソフトウェアの字句解析生成系 flex [2]、構文解析器生成系 Bison [3] を利用し、flex が生成する決定性有限オートマトン (DFA) と Bison が生成する LALR(1) 構文解析器の動作を視覚化している。これにより本システムの利用者は、扱うプログラム言語の flex と Bison の記述を用意すればよく、このことから2も実現している。また、3を実現するために Java 言語を用いてシステムを記述しており、ネットワーク越しにアニメーションを見せることができる。

3 視覚化する情報

DFA による字句解析と LALR 構文解析の一般的な動作について詳しくは省くが [1][4]、字句解析器の動作は「正規表現に基づいてレキシムを認識し、トークンに付随する属性を決める」というものであり、構文解析器の動作は「スタックと先読み記号を見て、構文解

析表を引き、その結果決まるシフト (shift) と還元 (reduce) を繰り返して状態を遷移しつつ構文を解析する」というものである。このことから、本システムでは表示する情報として、ソース中の先読み記号、DFA 遷移グラフ (拡大版、縮小版)、正規表現、スタック、LR 状態、文法中での還元時に適用された生成規則、解析木を用意した。実際には、それぞれを図2のように表示する。

4 本システムの構成

本システムは、字句解析部と構文解析部の2つに分けられる。構文解析器生成系には Bison に少し変更を加えたもの (“Bison 改” と呼ぶ) を利用している。これらを用いた本システム全体の構成は図1のようである。

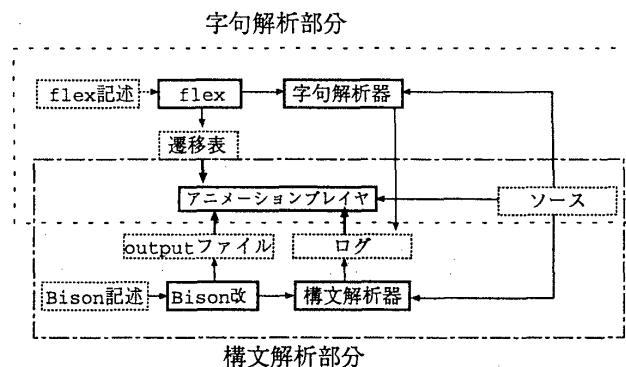


図1: システム全体の構成

図1の遷移表とは、正規表現を flex に与えて出力されたファイルである。“output ファイル”とは、Bison から出力されるファイルで、LR 状態や各 LR 状態における動作が記述された *.output という名前のファイルのことである。“ログ”とは、字句・構文解析器がソースプログラムを解析した際におこなった動作を出力したファイルのことである。これら3つのファイルをアニメーションプレイヤーへの入力としてアニメーションをおこなう。

5 視覚化の流れ

あるプログラム言語についてアニメーションを実行するために必要となるファイルは、字句解析のための flex 記述 と構文解析のための Bison 記述である。アニ

*Visualization of lexical and syntactic analysis in the compiler

[†]Naoki Niwa(niwa@is.titech.ac.jp)

[‡]Katsutoshi Kusume(kusume@is.titech.ac.jp)

[§]Masataka Sassa(sassa@is.titech.ac.jp)

[¶]Dept. of Math. Comp. Sci., Tokyo Institute of Technology

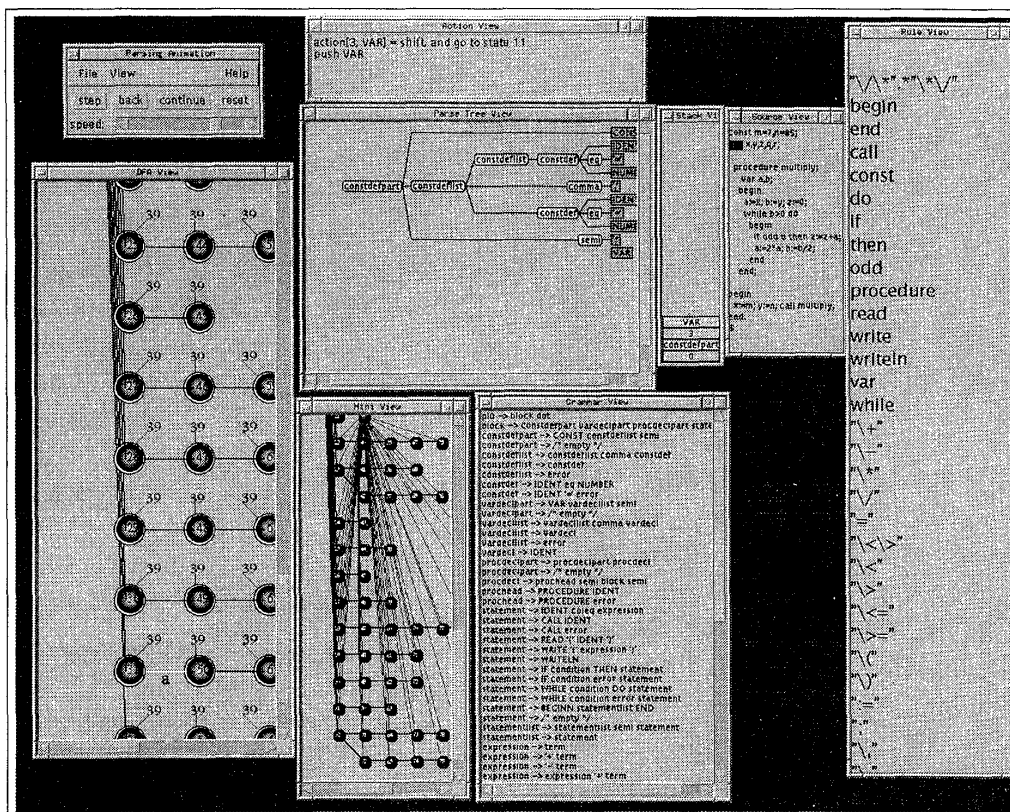


図 2: 実行時の画面

メーションをおこなうまでの流れは次のようになる。

1. flex 記述、Bison 記述 のファイルを用意する
2. flex 記述から字句解析器と“-Tf”オプションをつけてDFAの遷移表を生成し、Bison 記述から構文解析器と“-v”オプションをつけて“output ファイル”を生成しておく
3. 構文解析器を字句解析器とともにコンパイルして実行形式を得る
4. 生成した字句・構文解析器の実行形式でソースプログラムを解析し、ログファイルを得る
5. 遷移表、ログファイル、output ファイル、ソースファイルをプレイヤーの入力としてアニメーションをおこなう

1 から 3 までは、あるプログラム言語に対して一回だけ必要となる処理である。仮にこの処理が困難なユーザであっても、PL/0 言語コンパイラなどの字句解析器、遷移表、構文解析器、output ファイルをいくつか用意してあるので、ユーザはソースプログラムの記述をするだけで、ブラウザなどから本システムを利用できる。

6 今後の課題

今回実現したシステムでは、例えばビュー上のあるノードをクリックしたら、そのノードに対応したソー

ス上の範囲が反転表示されるといった機能はない。また、字句解析と構文解析のビューの連携も不十分である。今後はこのような各ビューが互に関連した表示を提供するようにしたい。さらに、1つのビューに対して複数の表示方法を用意するなど、情報の提供方法をより柔軟にしていきたい。

今回は字句解析と構文解析の動作を視覚化したが、ビューの見やすさを改善し、字句解析、構文解析以降のフェーズの動作を視覚化することも実現していきたい。

参考文献

- [1] A. V. Aho, R. Sethi, and J. D. Ullman. *Compilers-Principles, Techniques, and Tools*. Addison-Wesley, 1986.
- [2] V. Paxson *FLEX. Lexical scanner generator*. at Lawrence Berkeley Laboratory, Berkeley, California, 1996.
- [3] C. Donnelly and R. Stallman. *Bison Reference Manual*. Free Software Foundation, 1991.
- [4] 佐々政孝. プログラミング言語処理系. 岩波書店, 1989.