構文解析器の開発支援環境

笠原 史郎　　大久保弘崇　　粕谷英人　　山本 晋一郎

愛知県立大学 情報科学部 情報システム学科

要 旨

構文解析器生成ツール yacc の入力ファイル（yacc プログラム）の理解を支援するため，構文規則に着目し，その理解手法について議論する．次いで，Web ブラウザを用いて構文規則の対話的な表示を行う理解支援システムを提案する．本システムは，yacc プログラムのもつさまざまな側面からの理解を支援するための，例文表示機能を含む 5 つの機能を備える．本システムの与える理解支援について評価する．

The interactive browser and comprehension assistant for yacc grammar files

Shiro KASAHARA, Hirotaka OHKUBO, Hideto KASUYA, and Shinichiro YAMAMOTO

Department of Information Systems, Faculty of Information Science, Aichi Prefectual University

Abstract

We discuss the methods to understand the context-free grammar for yacc. Then we propose an comprehension assistant for the yacc grammar, which interactively displays syntax rules. The system has 5 features including example sentence presentation, to assist comprehension from various sides of view. It evaluates about the understanding support which this system gives.

1　概要

既存の yacc[1] プログラムを利用したり，改良したりする前には，まずその内容を理解する必要がある．表 1 に例を示すとおり，実用的な言語を構文解析するための yacc プログラムは，他の実用的なプログラム同様，相当な規模であり，C 言語の CASE ツールに SPIE[2] があるように，yacc プログラムにも CASE ツールによる支援が必要であると考えられる．

特に，yacc プログラムは構文規則部と C 言語によるアクション記述部が混在していることが，単一言語によるプログラムより，その理解を一層難しいものとしている．

表 1: yacc プログラムの規模

言語	C 言語	Java	Ruby
終端記号	58	78	103
非終端記号	63	100	105
構文規則	211	278	456
行数	417	608	563

本稿では，yacc プログラムのうちの構文規則部を重視し，構文規則を中心とした yacc プログラムの閲覧ツールを提案する．また単なる閲覧にとどまらない能動的な理解支援のための機能を組み込み，構文解析器の開発支援環境とする．

```
%%
stmt : expr ';' { printf("%d\n",$1);};
expr : t_NUM
     | t_ID
     ;
%%
```

図 1: example.xml

2 yacc プログラム CASE ツールに必要な機能

2.1 yacc2html

既存の yacc プログラムの理解支援ツールに yacc2html[3] がある．yacc2html は，lex プログラムと yacc プログラムを入力として，プログラムテキスト中に参照部から宣言部へジャンプするハイパーリンクを挿入した HTML ファイルを生成する．

図 1 の例では，1 行めの `stmt`，2 行めの `expr` に

```
<a name="stmt">stmt</a>
<a name="expr">expr</a>
```

とリンク先となる名前をタグ付けし，1 行めの expr に

```
<a href="#expr">expr</a>
```

とハイパーリンクをタグ付けする．

また対応する lex プログラムを同時に与えたときは，2 行めの `t_NUM`，3 行めの `t_ID` に対し，lex プログラムのアクション部で対応する箇所（return t_NUM; などとなっている行）へのリンクがタグ付けされる．

yacc2html では C 言語プログラムの記述に関しての解析は行っていない．

yacc2html が与える理解支援について考察する．プログラマは yacc プログラムに記述されている構文規則の全体像を捉えるため，ルートとなる構文規則から構文規則に出現する非終端記号を辿り，どのような構文を還元する規則であるかを推察する．この作業はルート以外の構文規則においても再帰的に行われる．この点において，yacc2html はプログラマの構文規則の探索作業を支援している．

逆に，プログラマの注目している構文規則が表す非終端記号が上位のどの構文規則に現れるかを理解することも重要である．しかし，yacc2html のタグ付けするハイパーリンクは使用から定義への一方通行であり，この意味で十分な理解支援を与えているとはいえない．

2.2 理想的な閲覧機能

ここでは理解支援となりうる閲覧機能について考察し，図 1 の example.y を用いてその例を挙げる．

1. 非終端記号の関連付け
   プログラマは構文規則から非終端記号を辿り，どのような構文を還元する規則であるかを推察する．yacc2html の機能と同様に，構文規則間の移動を支援する機能は必要である．
   例）stmt ルールの expr から expr ルールへ移動する．
2. 構文規則の関係
   ある構文規則に注目したとき，その構文規則を還元する非終端記号が現れる構文規則を親と呼び，構文規則に現れる非終端記号の構文規則を子と呼ぶことにする．
   この親と子の構文規則間の関係を理解することは，構文規則間でアクションがどのように連携して動作するか，ある規則に対して親は複数存在しうることから、特に擬似変数$$,$1,$2,... でどのような値を受け渡すか，を理解するために重要である．
   例 1）expr の親として stmt を提示する．
   例 2）stmt の子として expr を提示する．
3. 構文規則の表現する意味
   ある構文規則がどのような文を受理するかを知ることができれば，それは構文規則を理解するうえで有効である．つまり，注目している非終端記号が還元するトークンの並びを例示できればその非終端記号の意味をりかいする上で大きな手助けとなる．
   例）非終端記号 stmt の表現する意味として，

   ```
   t_NUM ';'
   ```

   と表示する．

4. アクション
構文規則を読むとき，記述されているアクションがプログラム理解の妨げとなることがある．構文規則に記述されているアクション部分の表示を一時的に折り畳む．
例）stmt ルールのアクションを折り畳み，以下のように表示する．

```
stmt : expr ';' <action> ;
```

3　理解支援システム

ユーザの目的とする構文規則を選択し，各機能によって動的に生成された HTML ファイルを Web ブラウザで閲覧する．

本システムは以下の提示方法で yacc プログラムを表示し，プログラマの理解を支援する．

- 全規則表示（ALL）

  全ての構文規則を表示する．構文規則に現れる非終端記号をクリックすることで，その非終端記号の構文規則に移動する．

- 親規則表示（PARENT）

  非終端記号を 1 つ選択し，その親の構文規則全てを表示する．

- 子規則表示（CHILD）

  非終端記号を 1 つ選択し，その子の構文規則全てを表示する．

- 親と子規則表示（PARENT&CHILD）

  非終端記号を 1 つ選択し，その親と子の構文規則全てを表示する．

- 受理トークン列の例示（EXAMPLE）

  非終端記号を 1 つ選択し，それに還元するトークンの並びを表示する．

3.1　画面構成

システムの画面構成を図 2 に示す．

機能指示部には非終端記号と機能を選択するフォームがある．このフォームを使用してユーザが要求した結果を表示部に表示する．

図 2: 画面構成

4　実装

4.1　システム構成

本システムでは，動的な HTML の生成方法として XML,XSLT,DOM を用いた．また，サーバーコンテナには Tomcat を用いた．システム構成を図 3 に示す．yacc プログラムを yacc2xml により XML 化としておく．ユーザからの要求に応じ，この XML を利用して XSLT,DOM により出力動的に生成する．

4.2　yacc2xml

yacc2html は yacc プログラムを読み込み，構文規則の構造，非終端記号の ID，アクションの ID 等の情報を持った XML ファイルを生成する．yacc2xml は実装に flex と bison を使用している．yacc2xml に example.y を入力したときのの出力

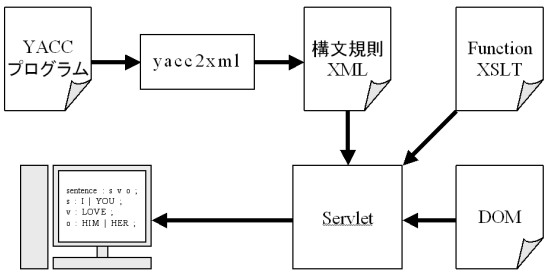

図 3: システム構成

を図 4 に示す．各タグの意味を表 2 に示す．

4.2.1 各表示機能の実現

yacc2xml で生成された output.xml を利用して 3.1 節で示した各機能を実現する．ALL，PARENT 等の出力には，さらに次の nontermnial がクリックされたとき，それを機能指示部にも反映させる機能をもたせた．この実現には javascript を用いた．

- ALL

  全ての GrammarRule を表示する．表示の順序は yacc プログラム中の記述順のままである．

- PARENT

  非終端記号の ID を受け取り，GrammarRule の中で，その RightRule 中にある nonterm で，idref 属性が入力された ID と等しいものを取り出し，表示する．さらに，選択した非終端記号の規則を末尾に表示する．それは，GrammarRule で，左辺を意味する nonterm の id 属性が与えられた ID であるものである．

- CHILD

  非終端記号の ID を受け取り，まずその非終端記号の規則を表示する．次にこの GrammarRule の RightRules に含まれる全ての nonterm について，その idref 属性で参照される GrammarRule を全て表示する．

- PARENT&CHILD

  PARENT 機能と CHILD 機能は同じ XSL ファイルで処理を行っている．機能の種類 sort を受け取り，sort が Parent か Child か，あるいはいずれでもないかで動作を切り換えている．

  sort が Parent でも Child でもないとき PARENT 機能と CHILD 機能を併せ，出力する．その分岐動作を図 5 に示す．

以上の表示では，yacc プログラムの一部あるいは全体を HTML で表示する．非終端記号を辿るリンクを実現するため，nonterm 要素で id 属性を持つものには name 属性，idref 属性を持つ

```
<GrammarRules>
 <GrammarRule>
  <nonterm id="s0">stmt</nonterm>
  <op>:</op>
  <RightRules>
   <RightRule>
    <nonterm idref="s1">expr</nonterm>
    <term>';'</term>
    <action actid="k1">{
        printf("%d\n",$1);
    }</action>
   </RightRule>
  </RightRules>
  <op>;</op>
 </GrammarRule>
 <GrammarRule>
  <nonterm id="s1">expr</nonterm>
  <op>:</op>
  <RightRules>
   <RightRule>
    <term>t_NUM</term>
   </RightRule>
   <op>|</op>
   <RightRule>
    <term>t_ID</term>
   </RightRule>
  </RightRules>
  <op>;</op>
 </GrammarRule>
</GrammarRules>
```

図 4: output.xml

表 2: output.xml の要素

要素名	意味
GrammarRules	構文規則全体
	GrammarRule の並び
GrammarRule	構文規則
	nonterm op RightRules の並び
RightRules	ある構文規則の右辺全体
	RightRule が	で区切られた並び
RightRule	ある構文規則の右辺
	以下の要素が入る
term	終端記号
nonterm	非終端記号 属性 id,idref を持つ
op	演算子
action	アクション 属性 actid を持つ

```
<xsl:if test="$sort!='Child'">
 PARENT 機能の部分
</xsl:if>
 選択した非終端記号の構文規則
<xsl:if test="$sort!='Parent'">
 CHILD 機能の部分
</xsl:if>
```

図 5: 表示機能の分岐

```
stmt
 |expr
 | | t_ID
 |';'
EOF
```

図 6: EXAMPLE 機能の例

ものには href 属性のアンカータグをつける．また，後者については onclick 属性で javascript を利用し，注目している非終端記号が機能指示部に反映されるようにすることによって，他の機能への移行を支援する．

構文規則にあるアクション部は<action>表示によって折り畳まれた状態で表示される．クリックにより，C 言語プログラムが展開した状態と相互に切り換えることができる．action 要素の actid 属性を用いてアクションを区別し，javascript で実装する．

- EXAMPLE

  選択された非終端記号から構文規則を再帰的に辿り，トークンを探索する．ループ対策のため，トークンの探索は深さ 16 で打ち切る．探索結果の抽象構文木を非終端記号を含めて HTML 化して表示する．実現には Java と DOM を用いた．出力例を図 6 に示す．

5 評価

2.2 節で考察した機能を実装できたか検討する．

1. 非終端記号の関連付け
   これは主に ALL 機能で実現した．yacc2html と同様に構文規則に現れる非終端記号をクリックすることで，その非終端記号の構文規則へ移動することができる．
2. 構文規則間の関係
   構文規則間の関係を知るための機能として PARENT,CIHLD,PARENT&CHILD 機能を実装した．また，これらの機能では構文規則を還元する非終端記号をクリックすることで，注目する非終端記号を変更し構文規則の表示を更新することができる．これらの機能を利用することで，構文規則間の関係を閲覧しながら親や子の構文規則を辿ることができる．
3. 構文規則の表現する意味
   構文規則の大意を直観的に知るための機能として，EXAMPLE 機能を実装した．EXAMPLE 機能では，選択した非終端記号に還元する例文の表示を行い，どのような構文規則であるかを推察することができる．同時に，探索した非終端記号を木構造で表示することでトークンまでの探索経路を知ることができる．
4. アクション
   アクションについては<action>という表示によって C 言語の記述を隠した．構文規則の理解のみに注目したときこの機能は有効である．また，アクションの内容を知りたいときは<action>と切り替えることが可能である．

2.2 節で挙げた機能についてはほぼ実装できた．また，これらの機能は本稿で着目した yacc プログラムの構文規則に関しての理解を支援した．

yacc2html と比較すると，yacc2html では lex プログラムと yacc プログラムを対象としているが，本システムは yacc プログラムの構文規則を対象とし，その理解に特化している．yacc2html の生成する出力は静的な HTML ファイルであるため，ブラウザのみで閲覧でき，手軽である．本システムでは動的に XHTML を生成するため，JSP コンテナなど大がかりな準備が必要である．

6 今後の課題

本稿で実現したシステムにより，yacc プログラムの理解支援を実現したが，よりよい支援を

行うために次のような課題がある．

6.1 構文規則以外の情報の取得

現在の yacc2xml は，構文規則のみを取り出して XML 化しているが，yacc プログラムにはこの他に宣言部とプログラム部がある．

また yacc は主に lex と併用して使用するので，lex プログラムの情報の取得することも有効である．これらの情報を取得し活用することは構文規則の理解支援だけではなく，yacc プログラム全体の理解支援になる．

6.2 ソースプログラムとの対応

本システムはソースプログラムに記述されている通りに表示するのではなく，構文規則ごとに ':','|',';' を整列したいわゆる pretty print を行っている．そのためユーザによっては見づらい表示形式であるかもしれない．またソースプログラムと見比べたとき，ソースプログラム上のどの位置に記述されている構文規則であるかがわかりづらいといった問題がある．したがって，ソースプログラムの記述をそのまま表示し，かつ本稿で実装した機能を使用できるような仕組が必要である．

現在の yacc2xml では構文規則の構造，非終端記号の ID，アクションの ID を取得し，得られた情報から XML ファイルを生成している．この手法ではスペースやタブ，改行文字を読み飛ばしているため，ソースプログラムをそのまま表示する機能を実装することができない．この問題を解決するために，ソースプログラムにタグを挿入する方法がある．タグを挿入することで，ソースプログラムの内容をそのまま残すことができる．今までの手法でスペースなどの情報を取得することも考えられるが，タグを挿入する手法をとる方が効率的である．

6.3 例文表示機能の改良

EXAMPLE 機能の現状の実装では，さまざまな例文を出力するために乱数をもしいている．文法の構造を考慮して，探索の打ち切りをなるべく少なくする例文生成のアルゴリズムを導入することが考えられる．しかし，言語には文法と意味があり文脈自由文法だけからでは，言語として意味のある例文の生成は行えず，探索アルゴリズムの改良によるアプローチには限界がある．

構文解析器への意味の正しい入力サンプルを別に用意できれば（例:C パーサに対して，実行可能な C プログラム），これを用いて意味のある例文表示機能が実現できる．そのためには，与えられた yacc プログラムからアクション部を削除し，例文表示機能のための情報取得アクションを代わりに挿入してサンプルを構文解析し，データベースに保存するという一連の動作を自動的に実行するシステムが必要である．

7 おわりに

構文解析器の開発支援のため，構文規則の理解支援について議論し考察した．また，構文規則の理解を支援する機能を備えたシステムを作成した．

ALL 機能を使用することで構文規則間の移動の手間を省くことができる．また PARENT 機能，CHILD 機能，PARENT&CHILD 機能では，必要な構文規則のみを閲覧することができる．全ての機能に共通して，非終端記号をクリックすることで機能指示部の非終端記号を選択し直す．これにより，注目している非終端記号の構文規則に対して容易に他の機能の実行を可能にした．

謝辞

御指導頂いた，愛知県立大学情報科学部 稲垣康善教授と名古屋大学院情報科学研究科 阿草清滋教授に感謝します．

参考文献

[1] Aho, Alfred V., Ravi Sethi, and Feffrey D. Ullman. 「Comilers:Principles, Techniques, and Tools.」 Addison-Wesley, 1986.

[2] 大橋 洋貴, 古山 将佳寿, 吉田 敦, 山本 晋一郎, 阿草 清滋: ソースプログラム・アーカイブ・サイト Sappy , オープンソースまつり'99 in 秋葉原 (1999/11)

[3] jutta: yacc2html, （5 Jan 1999）, http://kbs.cs.tu-berlin.de/~jutta/