プログラミング言語処理系論 (3)
Design and Implementation of Programming Language Processors

佐藤周行
（情報基盤センター/電気系専攻融合情報学コース）

今回の予定

- 「規格を読む」訓練をしましょう

- テキストはXMLの定義文書
  - http://www.w3.org/TR/2006/REC-xml11-20060816/
  - BNFの読み方を理解しましょう
  - 規格独特の言い回しを理解しましょう

次回の予定

- 文法から、パーサを作る
- BNFをそのまま解釈する
- 出力として、いろいろなものを考える
  - インタープリタで実行
  - 解析木を解釈
  - 機械語に翻訳して実行
- BISON,YACCの基礎知識は不要です

What is XML?

- やみくもに規格を読んでもわかりません
- で、XMLってなに？

- http://www.w3.org/

```
<?xml version="1.0" ?>  <!-- 住所録  -->
- <addressbook>
- <personal number="0001">
<name>Aさん</name>
<address>神奈川県</address>
<tel>00-0000-0000</tel>
<email addr="xxx@xxx.xx.xx" />
</personal>
- <personal number="0002">
<name>Bさん</name>
<address>東京都</address>
<tel>11-1111-1111</tel>
<email />
</personal>
</addressbook>
```

What is XML for?

- Web間/Webとブラウザ間を流れるデータの共通フォーマットとして設計
  - インターネット上のデータ流通に特化
- マークアップ言語 として定義
  eXtensible Markup Language
- Webだから、「文書」の形式
  - HTML
  - SGML

蛇足

- 実質的には
  - タグ <address> … </address>,
    <h1>…</h1>など
- マークアップ
  文書に関するメタな情報の指定（もともとは編集用語）
  - 段落指定、フォントサイズ指定などを行なう
  - 指定するものをタグという

具体例をいくつか

```
<?xml version="1.0" ?>
<!-- 住所録 -->
<!DOCTYPE addressbook SYSTEM
      "addressbook.dtd">
<addressbook>
   <personal number="0001">
      <name>Aさん</name>
      <address>神奈川県</address>
      <tel>00-0000-0000</tel>
      <email addr="xxx@xxx.xx.xx"/>
   </personal>
   <personal number="0002">
      <name>Bさん</name>
      <address>東京都</address>
      <tel>11-1111-1111</tel>
      <email/>
   </personal>
</addressbook>
```

```
<!ELEMENT addressbook (personal*)>
<!ELEMENT personal
      (name,address,tel,email)>
<!ATTLIST personal number CDATA
      #REQUIRED>
<!ELEMENT name (#PCDATA)>
<!ELEMENT address (#PCDATA)>
<!ELEMENT tel (#PCDATA)>
<!ELEMENT email EMPTY>
<!ATTLIST email addr CDATA
      #IMPLIED>
```

XMLの構造

- <tag>ではじまり、</tag/>でおわる
  - HTMLは違うけど、XMLでは許されないか？
  - <t
  - る
- “abc”は引用符で囲まなければならないのか？
- HTMLでは違うぞ
- タグはネストしてよい

これで、大体はわかるけど、ができ
大体でいいの？

XMLの構造

- 言語を定義するときの標準的な手法を説明する
  - 何を定義すると言語が定義されるのか？
    - 言語の構造（Syntaxを含む）
    - 言語の解釈（Semantics）
  - どのように定義すると「定義」になるのか？
    - BNF
  - 定義の手法に標準的な方法はあるか？
    - 言い回し

XMLの仕様書

- まずは目次を見てみる
- 1 Introduction
  2 Documents
  3 Logical Structures
  4 Physical Structures
  5 Conformance
  6 Notation

- ここからわかること
  - XMLは、Documentを定めている
  - XMLには、Logical StructureとPhysical Structureがある
  - Notationがあるということは、わからない表記があったらこの章を見ればよいということだな

通常のプログラミング言語では

- Fortran2003
  1. Overview
  2. Fortran terms and Concepts
  3. Characters, lexical tokens, and source form
  4. Types
  5. Data object declarations and specifications
  6. Use of data objects
  7. Expressions and assignment
  8. Execution control
  9. Input/output statements
  10. Input/output editing

- 続き
  11. Program units
  12. Procedures
  13. Intrinsic procedures and modules
  14. Exceptions and IEEE arithmetic
  15. Interoperability with C.
  16. Scope, association, and definition

- 大体のあたりをつけられるか？慣れればOK
- 例えば、logical structureとphysical structureはDocumentsで説明されている

  Physically, the document is composed of units called entities. An entity may refer to other entities to cause their inclusion in the document. A document begins in a "root" or document entity.

- Logically, the document is composed of declarations, elements, comments, character references, and processing instructions, all of which are indicated in the document by explicit markup.

Notation？

- Notationをざっと見ると、どのような記述技術が使われているかわかる。

- The formal grammar of XML is given in this specification using a simple Extended Backus-Naur Form (EBNF) notation. Each rule in the grammar defines one symbol, in the form

  symbol ::= expression

XMLの構造

- 形式文法の理論を用いる（Chomsky et.al.）
  - 言語を生成する規則の集合（文法）を与える

    S1 S2 S3…Sn → T1 T2 … Tm

  - 左辺にマッチした記号列を規則→を用いて右辺に変換する
  - 記号には、これ以上の変換ができない終端記号と、そうでない非終端記号がある
  - 有限回の変換で終端記号の列を生成する
  - 生成される列の集合を「言語」という

簡単な例を

- 言語と文法の例
  S → P
  P → a P a
  P → b P b
  P →
- 言語の生成の例
  S -> P -> a P a -> ab P ba -> aba P aba
  -> abaaba

言語のクラス

- Regular Expression
  - |左辺| = 1, 右辺は terminal+non-terminalのみ
- Context Free Grammar
  - |左辺|=1
- Context Sensitive Grammar
  - |左辺| ≦|右辺|
- Turing Machine
  - なんでもあり

具体的に言語を定義する

- 簡単な式を定義する

```
lines : lines expr '¥n'
      | lines 'n'
      |
      ;
expr : expr '+' term
      | term
      ;
term : term '*' factor
      | factor
      ;
factor : '(' expr ')'
      |DIGIT
      ;
```

ここからどのような言語が生成されるかを観察してみよう

- Fortran規格 p.519ffを見てみよう(01539-01_2004.pdf)
  - Context Free Grammarで記述されていることが観察される

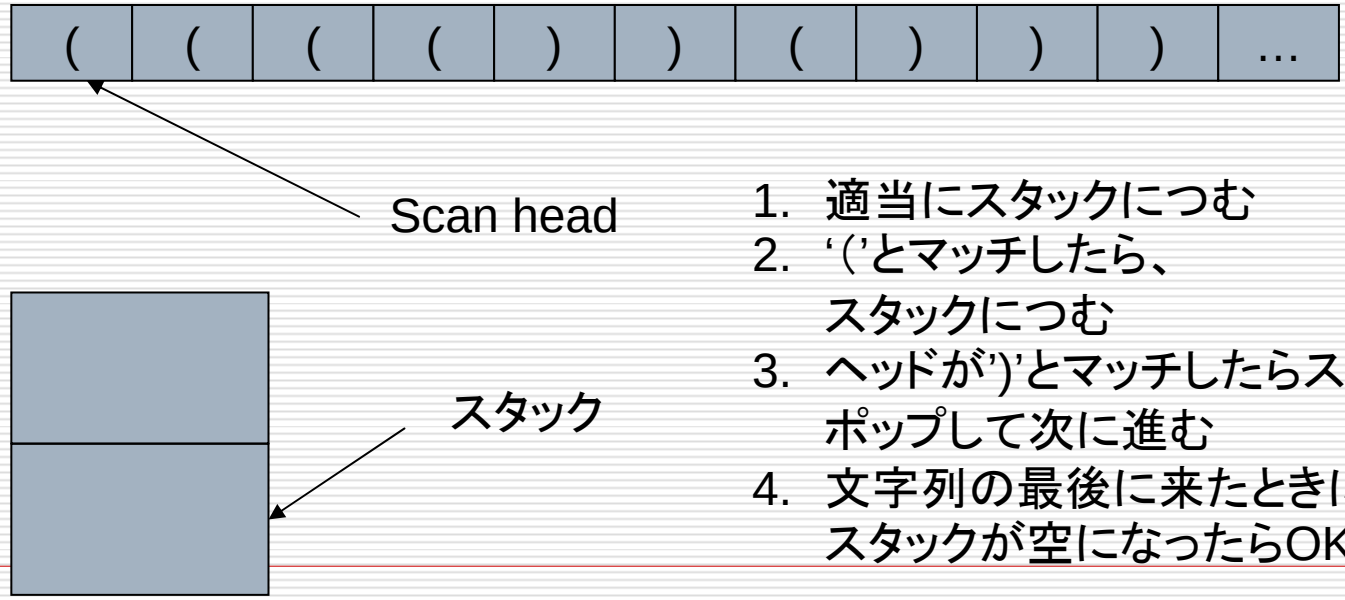
問題

- 文法と、具体的にひとつの文字列が与えられたときに、その文字列が文法に則って生成されたかどうかをチェックせよ

- 自明な解として、「文法によって生成される文字列を列挙して、その中に当該文字列が含まれているかどうかをチェックする」
  - やってられない

通常の解

- 文法に応じて、文字列を入力として、Yes/Noを出力する「受理機械」を作る
- 例:



1. 適当にスタックにつむ
2. ‘(’とマッチしたら、スタックにつむ
3. ヘッドが’)’とマッチしたらスタックをポップして次に進む
4. 文字列の最後に来たときにスタックが空になったらOK

- 「受理機械」は、言語の生成と逆のことを往々にしてやります
- この話（Parse）は来週にします

さてと、

プログラミング言語の定義

- ほとんどすべてのプログラミング言語は、CFGを用いて定義されます
- 定義に用いる記法をBNFといいます。BNFはこの意味でメタな意味の言語です
- BNFは　CFG＋α　で定義されます

BNFの（＋α）部分

Syntax ::= Alt1 | Alt2 (選択肢)
Syntax ::= [A] B ([ ] は省略可能 A?とも)
Syntax ::= A+, A* (一回以上の繰り返し、ゼロ回以上の繰り返し)

その他にも、アドホックな拡張がなされることがあります。規格で使う記法については25ページ以降をみましょう

- 言語はBNFだけで定義されるのではありません
  Document
- ***[1] document ::= (prolog element Misc*) - (Char* RestrictedChar Char*) Matching the document production implies that:***
- ***It contains one or more elements.***
- ***[Definition: There is exactly one element, called the root, or document element, no part of which appears in the content of any other element.] For all other elements, if t… e content of another element, the …ntent of the same element. Mor… elements, delimited by start- and en…ags, nest properly within each other.***
- ***[Definition: As a consequence of this, for each non-root element C in the document, there is one other element P in the document such that C is in the content of P, but is not in the content of any other element that is in the content of P. P is referred to as the parent of C, and C as a child of P.]***

この部分は何？

- 文法＋制約条件が言語の定義のこつです
  - 「制約条件」を軽視してはいけません。ここにCFGだけでは記述しきれないさまざまなルールが自然言語で記述されています
  - 文法部分をシンプルに保ちながら、自然言語の記述力を活かす事で、プログラミング言語の定義の手法は完成しました（制約が守られているかどうかのチェックにはチェック用のプログラムを書くことが必要になります）

言語の規格

- では、ちょっと読んでみましょう

- すぐつまります。なぜ？
- 規格で使う独特の言い回しについては、最初に解説があります。ここを軽視してはいけません
- RFC2119（http://www-sato.cc.u-tokyo.ac.jp/SATO.Hiroyuki/PLDI2010/rfc2119.txt ）
- 規格ローカルな言い回しの定義は最初のほうに記述があります。

言語の規格の読み方

- 規格の書かれ方
  - 概念の定義
  - Syntaxの定義
  - 制約
    - Well-formedness constraint
    - Validity constraint
  - 解説

- SyntaxはBNFで書く
- 例:

[39] element ::=
EmptyElemTag |
STag content ETag
[WFC: Element Type Match]
[VC: Element Valid]

- 例えば、以下のものがelementになる

 Tab

- すると、STAGはくではじまって…
  - 実際

[40] STag ::= ‘<’ Name (S Attribute)* S? ‘>’

[WFC: Unique Att Spec]

- Contentは、elementを含むデータの列で…
- ETAGは…
  - 実際

[43] content ::= CharData? ((element | Reference | CDSect | PI | Comment) CharData?)*

[42] ETag ::= '</' Name S? '>'

- STAGとETAGに関する制約がさらに加わる

Well-formedness constraint:
Element Type Match

The Name in an element's end-tag MUST match the element type in the start-tag.

- 憶えるべきことはこれですべて
- あとは、理解力と想像力
- 規格が読めると他人と議論ができます
  - このプログラムは文法上正しいかどうかは、コンパイラのエラーメッセージで判断するのではなく、規格書をみて判断するものです。
  - 規格が読めない人は「判断の根拠」がいつまでたってもあいまいです。
  - コンパイラが間違っているかどうかの判断もできます。

JIS規格でも話はほとんど一緒

- Fortranの文法はどう定義されているか

  01539-01_2004.pdf

- Fortranは411のルールと468の制約で定義されている
- （XMLのルールは83）

今回のまとめ

- 言語を定義するにはCFG＋制約を用います
- 規格の読み方にはこつがあります。
  - 定義されているものは概念と文法と意味
  - あとは理解力と想像力
- Fortranは411のルールと468の制約で定義されています