SADの文法(1)

2011/06/01

森田 昭夫

文法とは(1)

言語処理系の構成要素

- 字句解析(Lexical Analysis)
  - 入力記号列の語彙への分割
  - Tokenizeとも呼ばれる
- 構文解析(Syntax Analysis)
  - 語彙列に、規則に当てはめ、文の構造(構文木)を特定する
  - Parseとも呼ばれる
- 意味解析(Semantics Analysis)
  - 構文木に意味を与える
  - コンパイラでは目的マシンのコード生成、インタプリタでは入力の実行・評価
  - Code Generation, Evaluationなど処理系によって様々に呼ばれる

文法とは(2)

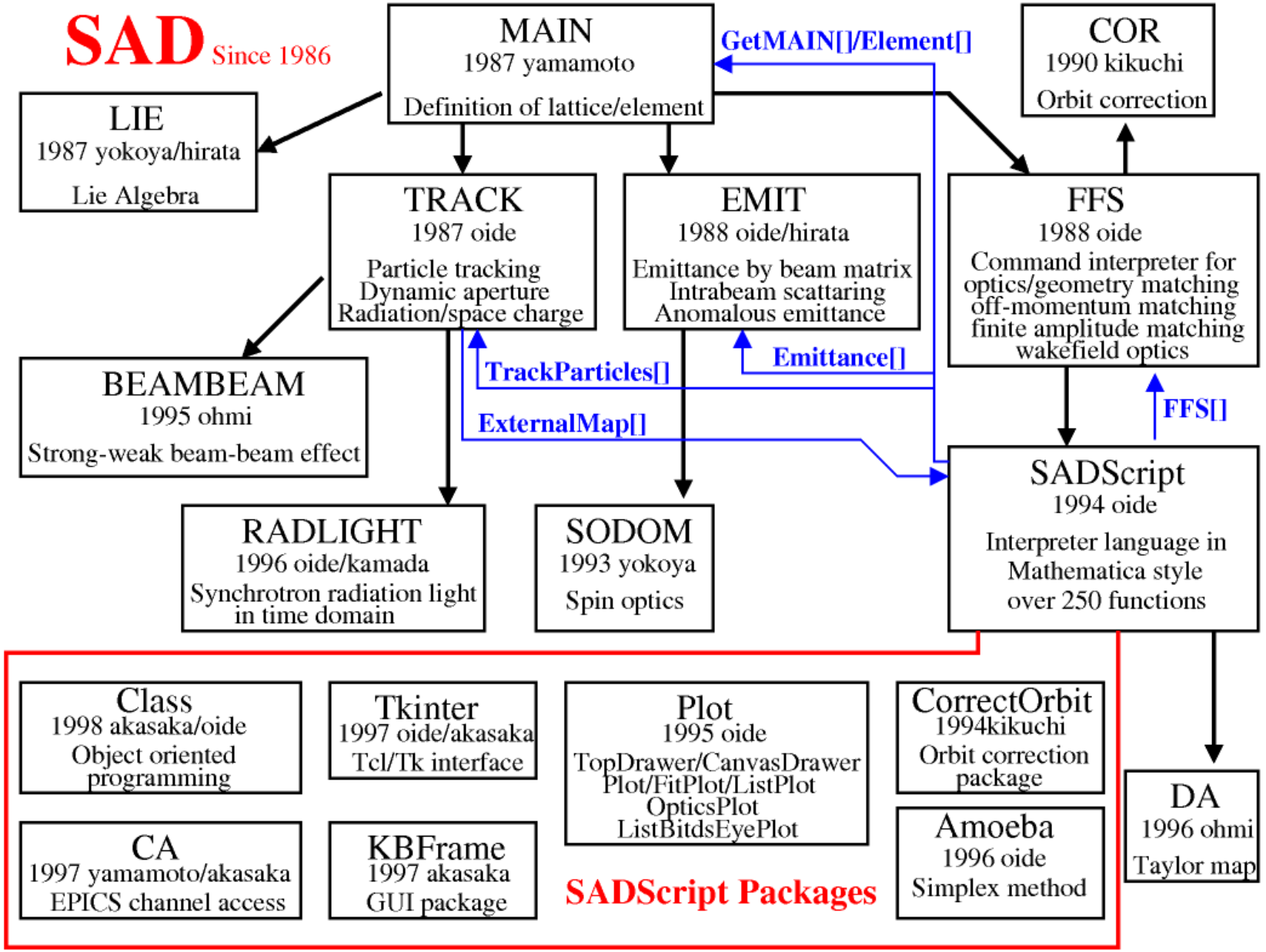
- 文法とは、構文規則集のこと
  - 構文解析を行う際の構文規則の集合
  - 生成文法(言語を生成する文法)などとも呼ぶ
- 覚えると便利？
  - 正しい文の書き方が分かる
    - 複雑な文を正しく書き下せるようになる
    - 複雑な文を正しく読み解けるようになる
  - これだけで、プログラムを書ける訳ではない！
    - 構文木の意味論が必要
    - 実装する内容(アルゴリズム)を考える必要も…

頭の固い計算機に、正しく此方の意図を知らせるのには必須！

SADの実態

- Kitchen Sink Approach
  - 必要そうなものは、片っ端から放り込んである
  - その場その場で、新しい機能を追加している
  - トップダウン的な一貫した機能分類ではない
- 歴史的経緯を引きずった建て増し構造
  - 各部のデザインが一貫しないというか、管理されていない
  - 外部仕様書やマニュアルの整備が追いついてない
    - 実装内容の詳細はソースを追うしかない
- 慢性的なテスト不足
  - どこまで正しく動くかは、誰も知らない
    - 系統的なテストが行われていない
    - 比較する外部仕様書が無い
  - ソースレベルで分かるバグも多数ある
    - レビュー不足

SADの構造(1)



1st SAD Workshopのブロック図を整理したもの

SADの構造(2)

良く使われる部分

- MAIN Level
  - 最も古い環境であり、SAD起動時の初期環境
  - 他のLevelへの遷移を行える
  - ビームラインの定義記述に多用される
- FFS Level　　(Final Focus Systemが語源)
  - FFTB設計用に開発された対話的な光学計算・最適化環境
- SADScript
  - FFSに埋め込まれた、本格的なプログラミング言語処理系
  - 文法と意味論は、Wolfram Researchの Mathematicaに影響を受けてるというか、パクリ
  - チューリング完全なので、何でも出来る(計算理論的には)

相互に関係を持ってはいるが、個別の名前空間、
字句・構文規則を持った別の言語である

MAIN Level(1)

- 参考資料
  - Tokenizer
    - src/gettok.f
  - Parser
    - src/toplvl.f
    - src/doelem.f(element_definition構文)
    - src/rdkwdl.f(element_body構文)
    - src/doline.f(line_definition構文)
    - src/lread.f(line_body構文)
    - src/dAssgn.f(assignment構文)
    - src/doACT.f(acction構文)
    - src/calc.y(expr構文[LALR(1)文法])
  - 予約語定義部
    - src/initbl.f
    - src/initb1.f

MAIN Level(2)

- 文字集合
  - spaces　　空白、タブ文字、改行記号
  - delimiters　[-+*/'`"(){}<>=!#%@:;?,~^:spaces:]
  - alphabet　[a-zA-Z]
  - digit　[0-9]
- 字句規則
  - !から改行までと、字句に前置されるspacesは読み飛ばす
  - identifier(識別子)
    - [:alphabet:][^:delimiters:]*
  - numeric(数値)
    - ([:digit:]+(.[:digit:]*)?|.[:digit:]+)([dDeE][-+]?[:digit:]*)?
  - string(文字列)
    - ('([^']+|''[^']*)*'|"([^"]+|""[^"]*)*")

MAIN Level(3)

■ 構文規則(抜粋)

```
statement :        element_definition | line_definition |
                   action_statement | reserved_statement | assignment
element_definition:  element_type { element_body* }
          |            element_type element_body* ;
          |            element_body ;
element_body:    element_name = [({] ( keyword = expr )* [})]
line_definition :  ( LINE | line ) { line_body* }
          |        ( LINE | line ) line_body* ;
          |        line_body ;
line_body:         line_name = ( ( (numeric * )? [-+]? ( line_name | element_name ) )* )
element_type:    identifier
element_name:    identifier
keyword:         identifier
line_name:       identifier
assignment :       identifier = ( expr (, expr)* ) ;
          |        identifier = expr ;
```

statementは、必ずidentifierで始まる

接頭identifierによって文の種別が決まる

identifierの前方参照は、禁止されている

MAIN Level(4)

- element_typeとなりえる identifier
  - DRIFT
  - BEND, QUAD, SEXT, OCT, DECA, DODECA, MULT
  - UND, WIG
  - CAVI, TCAVI
  - SOL, MAP, COORD, BEAMBEAM, PHSROT, INS, APERT
  - MARK, MONI
  - TEST
    - lower caseも有効
- keywordとなりえる identifier
  - element_typeに依存している
  - 概略はSADHelp、完全な定義は src/initb1.fを参照
    - 50行目付近から keywordとなり得る identifierの定義
    - 300行目付近から element_type毎に有効なkeywordの定義

MAIN Level(5)

- expr
  - 計算機上の一般的な数式表現(完全な定義は、src/calc.y)
    - 算術演算子　　*, /, +, -, ^(累乗)
    - 論理演算子　　&, |, ~(論理否定)
    - 比較演算子　　<, >, =, <=, >=, <>
    - SQRT, EXP, LN(自然対数), LOG(常用対数)
    - SIN, COS, TAN, SINH, COSH, TANH, ATAN, ATAN2
  - 一般的ではない項としてnumeric UNITがある
- UNIT
  - 数値の単位を表す identifier
  - src/initb1.fにて icUNITが現れる行で定義されている
  - T, JOULE, HZ, KHZ, MHZ, GHZ, V, KV, MV, GV, %
  - eV, KeV, MeV, GeV
  - EV, KEV, MEV, GEV
  - rad, mrad, deg, m, cm, mm, gauss
  - RAD, MRAD, DEF, M, CM, MM, GAUSS

MAIN Level(6)

- action_statement
  - 接頭identifier
    - PLOT, GRAPH, LIE, TRACK, FFS, QUICK, EMIT
  - 構文
    - action_statement: identifier (parameter = value)* ;
    - value: (val_list | val)
    - val_list: (val (, val)*)
    - val: (expr | T | F | string)
- reserved_statement
  - 接頭identifier
    - EXPAND, PRINT, READ, LIST, REVERSE, STOP, ON, OFF
  - 構文は個別に実装