

正誤表

以下、×欄に間違った記述を載せ、○欄に訂正内容を載せる。必要ならば、その後に関連事項を解説する。なお、内容に関する重要な訂正には項目の前に☆を付けたので注意してほしい。

☆ 7 ページ, 1.2.2 項, 下から 6 行目

- × 「ALGOL が実際に使われることはなかった。」
- 「ALGOL が実際に使われることはほとんどなかった。」

日本においては ALGOLIP という名称のコンパイラが実際に作成され、使用されたことを識者から指摘された。「ALGOLIP」をインターネット検索すると当時の状況を記したメモなどがヒットする。

13 ページ, 1.3.1 項, 6 行目

- × 「別のプログラミング L_T で書かれた」
- 「別のプログラミング言語 L_T で書かれた」

☆ 47 ページ, 3.2.3 項, 図 3.3

- × —
- 図 3.3 に、状態 11 からピリオド “.” を読んで状態 13 へ遷移する枝を追加する。
上記追加を行わない場合、“0.”で始まる文字列 “0.1”などを正しく解析できない。

92 ページ, 6.4 節, 2 行目

- × 「(章末問題 2 参照)」
- 上記を削除する。

都合により、92 ページ、1 行目の文章「前節の仮定は... 受け入れられるものではない。」の趣旨に関連する章末問題は掲載していない。

☆ 99 ページ, 6.4.4 項, 表 6.7

- ×

$Z \rightarrow A \text{ EoF}$	$\{\text{NUM, SEMI}\}$
-------------------------------	------------------------
- | | |
|-------------------------------|-----------------------------|
| $Z \rightarrow A \text{ EoF}$ | $\{\text{NUM, SEMI, EoF}\}$ |
|-------------------------------|-----------------------------|

☆ 99 ページ, 6.4.4 項, 表 6.8

- ×

Z	$Z \rightarrow A \text{ EoF}$	$Z \rightarrow A \text{ EoF}$	
---	-------------------------------	-------------------------------	--
- | | | | |
|---|-------------------------------|-------------------------------|-------------------------------|
| Z | $Z \rightarrow A \text{ EoF}$ | $Z \rightarrow A \text{ EoF}$ | $Z \rightarrow A \text{ EoF}$ |
|---|-------------------------------|-------------------------------|-------------------------------|

表 6.7, 表 6.8 は上記の通り訂正するが、100 ページの図 6.9 のプログラムに変更はない。と言うのも、90 ページの (2) の約束により、同じ左辺を持つ構文規則が唯一の場合には switch 文を用いないからである。

106 ページ, 6.7.1 項, 図 6.11

× 「0: Z → tt Exp EoF」

○ 「0: Z → Exp EoF」

☆ 121 ページ, 7.2.1 項, 下から 8 行目

× 「LR(0) 項 $[Z \rightarrow S \bullet \text{EoF}]$ のみを持つ LR 状態を最終状態と呼ぶ。」

○ 「のみ」を削除し, 「... を持つ LR 状態を最終状態と呼ぶ。」へ変更する。

元の文法の開始記号 S が構文規則の右辺に現れる場合には, LR(0) 項 $[Z \rightarrow S \bullet \text{EoF}]$ を含む状態が他の LR(0) 項を含むこともある。あるいは, 元の文法の開始記号 S が構文規則の右辺には現れないと仮定すれば, 上の修正は不要である。構文規則の右辺に元の文法の開始記号 S が現れる場合には, 新しい開始記号 S' を導入し, 構文規則 $S' \rightarrow S$ を追加すれば, この仮定が成り立つ。

210 ページ, あとがき, 9 行目

× 「A. W. Apple」

○ 「A. W. Appel」

その他の注意事項

bison について `bison` は, `yacc` の上位互換に相当し, しかも様々な拡張機能を有する構文解析器生成系である。本書は `yacc` のみを考慮して記述しており, `bison` については触れていない。上位互換性のため, 本書のほとんどの内容は `bison` を用いる場合にも有効であるが, いくつかの点で注意が必要である。注意点は `bison` のバージョンにも依存すると思われるが, 現時点で判っている点を以下に述べる。

生成ファイル名 `yacc` でコンパイルすると, C プログラム・ファイル `y.tab.c` が生成される。これに対して, `bison` で, たとえば `hoge.y` というファイルをコンパイルとすると, C プログラム・ファイル `hoge.tab.c` が生成される。生成ファイルを `yacc` 流にする場合には, コンパイル・オプションに `-y` または `--yacc` を追加すればよい。このコマンド・オプションは, `bison` の動作を伝統的な `yacc` に適合させるためのオプションである。

トークンの宣言 本書の宣言方法

```
%token ID,NUM,REAL;
```

を用いて `bison` でコンパイルすると, `bison` がエラーまたは警告が発する可能性がある。その場合には, 宣言を以下のように修正する。

```
%token ID NUM REAL
```

トークン種別名の並びの区切りには空白を用いる。なお, このエラーまたは警告はコマンド・オプション `-y` または `--yacc` を用いても解消しない。

`yyval` `bison` では変数 `yyval` は構文解析関数 `yyparse()` の局所変数として宣言されている。そのため, 図 9.17 のように `yyval` を `yyparse()` の外で使用するようなプログラムをコンパイルすると, C コンパイラが変数の未宣言エラーを発する。これを解消するには, たとえば `yacc` 記述の宣言部 (図 9.15) の `%{ %}` 内に

```
YYSTYPE myval;
```

と大域変数 `myval` を宣言しておき、`yacc` 記述の構文規則部の開始記号の意味値を求める箇所では、`$$ = ...` を `myval = ...` に置換し¹、`yyparse()` の外では `yyval` の代わりに `myval` を用いればよい。なお、このエラーはコマンド・オプション `-y` または `--yacc` を用いても解消しない。

¹この場合、その開始記号は各構文規則の右辺には現れないと仮定する。