スキャナ・ジェネレータＳＧＥＮの開発

3M−2

山口　裕子　　久島　伊知郎　　神野　俊昭

（株）日立製作所　システム開発研究所

１．はじめに

コンパイラの自動生成の研究の一環としてスキャナ・ジェネレータＳＧＥＮ（Scanner GENerator ）を開発した。コンパイラの中でスキャナは入力されるソース・プログラム中の全ての文字を読む唯一のフェーズであり、最も時間のかかるところである。生成するスキャナの実行性能がよく、入力仕様の書き易さ等の点で使い勝手の良いものを開発することを目的とした。

２．概要

ＳＧＥＮは、プログラミング言語の字句の正規表現に基づく形式的仕様を入力とし、決定性有限オートマトンモデルに基づくスキャナを生成する（**図１参照**）。スキャナは字句を認識したら対応する字句番号を返す。字句解析と同時に行う処理は、意味動作として入力仕様中に記述できる。生成されるスキャナの記述言語にはＣとPascalがある。

３．スキャナの処理

ＳＧＥＮが生成するスキャナは文字を読み込み、文字と現在の状態から、オートマトンを表す遷移表上の次の状態をひく遷移表検索手続きを呼び出して次の動作を決めていく。

文字をみるだけで済む場合は、なるべく遷移表検索手続きを呼ばない方法をとる。これにより、全て検索手続きを呼び出す場合に比べ、５～２０％高速化している。

４．ＳＧＥＮへの入力仕様

ＳＧＥＮへの入力例の一部を**図２**に示す。字句の定義部（「#token」以下）には、字句の仕様と対応する字句番号を記述する（何も書いてないものには直前のものに１足した値を自動設定する）。これにより、スキャナの動作として字句番号を返すという意味動作を書かずに済み、記述が簡潔になる。また、字句の出現回数を数えるといった通常のスキャナ以外の用途にも適用できるように、字句番号を返す動作をしないようにも指定できる。

キーワードは字句定義部でも定義できるが、遷移表を小さくするために専用のキーワード定義部（「#keyword」以下）を設けた。

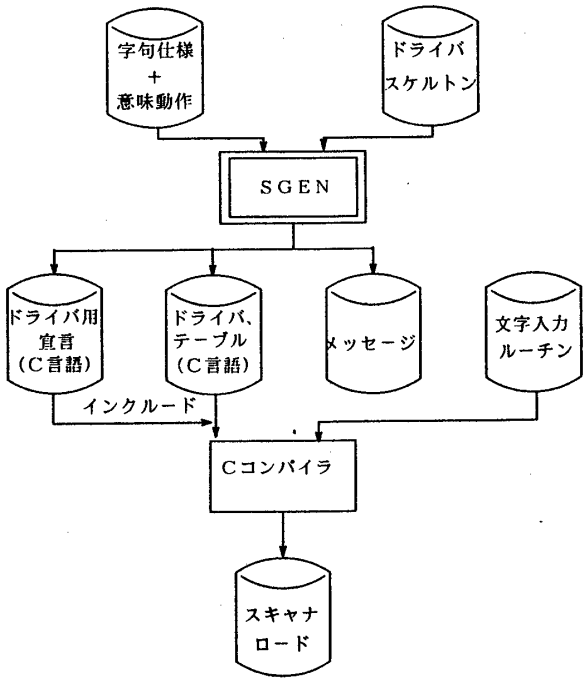

図1　SGEN　C版によるスキャナの生成

Design and Implementation of Scanner Generator SGEN

Yuko Yamaguchi, Ichiro Kyushima, Toshiaki Kohno

Systems Development Laboratory, Hitachi Ltd.

```
#keyword
  'sizeof' 17 #permit 1;   --①
     :
  'while'      #recover 'whle';
#lexicon                    --②
  ID = letter(letter|digit)*;
#token
  ';'     1    #refact action;
  ','          #refact action;
     :
  ID           #refact IsKeyword
               #limit 14;    --③
```

図２　ＳＧＥＮへの入力例の一部

４．１．エラー訂正のための仕様

字句解析時に字句エラーを訂正をすることは一般に困難であり、誤った訂正をする恐れもある。安全に字句エラーを訂正するために、入力仕様で字句毎にどの程度のつづり誤りを訂正するかを指定し処理する方式を考案した。つづり誤りを訂正するか、あるいはエラーメッセージ出力に留めるかはスキャナ実行時にユーザが指定する。

通常、スキャナは文字列を切り出したら、定義されたキーワードを登録したキーワードテーブルを検索して該当するものがあればこの字句番号を返し、なければ識別子とする。このためキーワードのつづり誤りは識別子とみなされることが多い。

ＳＧＥＮでは、実際の字句のつづり誤りは１文字違いという場合が多いことに着目して、キーワードテーブルを検索する際に１文字違いのものがあったら、つづり誤りとみてこの字句番号を返すことにした。このとき問題になるのは、キーワードと１文字違う字句がつづりを誤ったものではなく、識別子だった場合である。このような不正な誤り訂正を避けるため、１文字違いをつづり誤りとみなしてよいキーワードを入力仕様中で指定させることにした。この指定をしたときユーザは自分で指定したキーワードと紛らわしい識別子を、字句解析するソース中で用いないようにしなければならない。

１文字違いのつづり誤りを許すキーワードに対しては**図２**の①のように「#permit 1」と書く。字数の多いキーワードは２文字違いまで許すとすれば、「#permit 2」と指定する（３文字以上違っている場合は、つづり誤りというより識別子とみるべきなので、「#permit」の後には２までしか書けない）。

また、**図２**の②のようにユーザがよく間違えるパターンを指定する仕方も用意した。「#recover 'whle'」と指定してあるとスキャナは「whle」を「while」の誤りとみなす（キーワードテーブルには'whle'を登録してあるので、テーブル検索は「#permit 1」と指定した場合より、高速になる）。これは指定した文字列「whle」を本来の字句「while」に置き換える機能という見方もできる。ユーザは自分用のスキャナを生成して自分好みの字句を用いたプログラムが処理できるようになる。

４．２．その他

一般に識別子や文字列リテラルは字数の上限値が定められているが、正規表現ではこれを表すことができない。字句の字数の上限値をチェックするための指定方法を用意した。**図２**の③の「#limit 14」は識別子(ID)の上限値を１４文字としてチェックすることを指定している。

５．おわりに

エラー訂正機能とスキャナの高性能性を特徴とするスキャナ・ジェネレータＳＧＥＮを開発し適用に供している。

参考文献

H.Mossenbock, Alex--A Simple and Efficient Scanner Generator, SIGPLAN Notices 21(12),139-148(1986)