情報科学科演習 D
課題 1

基礎工学部情報科学科ソフトウェア科学コース
学籍番号: 09B20001

イグレシウスエドゥアルド

2022 年 10 月 15 日

1　システムの仕様

1. 字句解析器は Lexer.run(String, String) から実行される。
2. プログラムの入力としては読み込み用のデータパス（ps ファイル）と書き込み用のデータパス（ts ファイル）である。
3. 未完成の文字列、つまり「'aaa」などの場合は文字列とする。
4. プログラムの出力として書き込み用のファイルに書き込んだデータ、つまり、読み込み用のデータパスに指定されたファイルを事前に定義されたトークンごとに分割して、指定された形式で書き出されたファイル（字句解析ファイル）である。
5. プログラムの端末の出力として、プログラムが正常に終了する場合、”OK”を出力して、入力ファイルが見つからない場合は”File Not Found”を出力する。

2　プログラムの方針・設計

最初に、ワードを処理するために、8 つの状態（S0、S1、S2、S3、S4、S5、S6、S7）を定義して、各々の状態の状態変遷図を図 1 に示される。

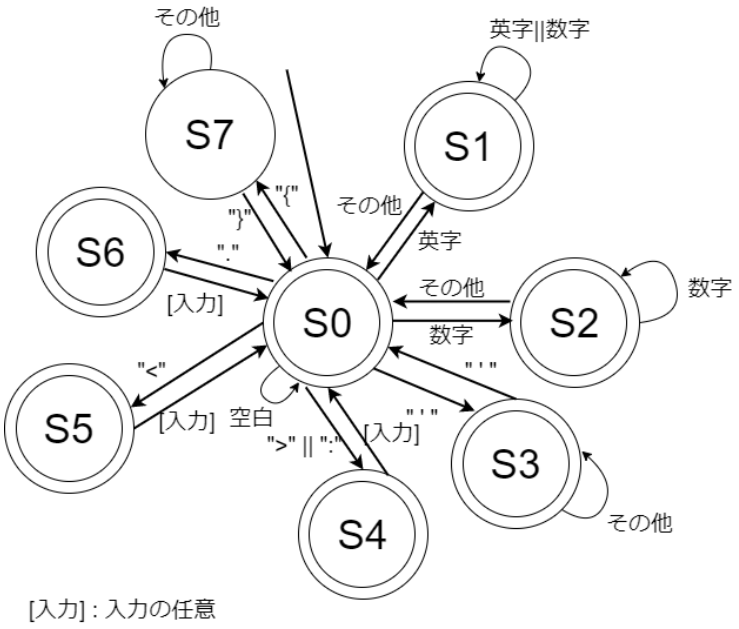

図 1　プログラムの状態遷移図

それぞれの状態は以下に説明する。

- S0、S0 は他の状態へ行くためのピンポイントと一文字記号に対応する状態である。S0 の実際の処理は図 2 に示される。

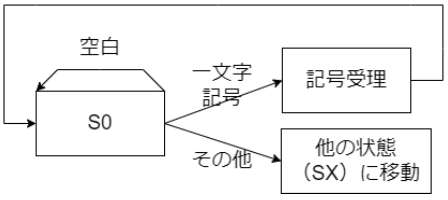

図 2　S0 内の処理

ここで、一文字記号とは絶対一文字のみ構成される記号、つまり、’=’、’*’、’/’、’+’、’-’、’;’、’(’、’)’、’[’、’]’、’,’ のことである。また、空白とは’\n’、’\t’、’ ’ のことである。

- S1、S1 は予約語と識別子に対応する状態である。S1 の実際の処理は図 3 に示される。

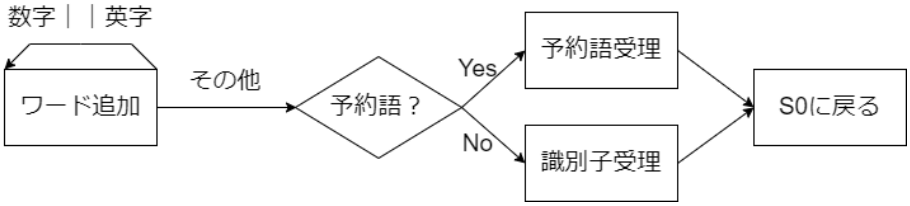

図 3　S1 内の処理

- S2、S2 は数字に対応する状態である。S2 の実際の処理は図 4 に示される。

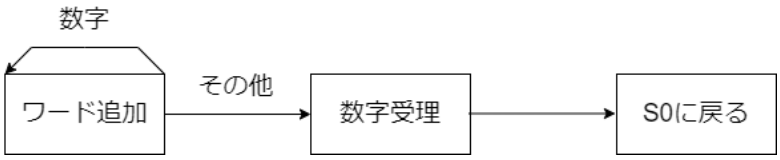

図 4　S2 内の処理

- S3、S3 は文字列に対応する状態である。S3 の実際の処理は図 5 に示される。

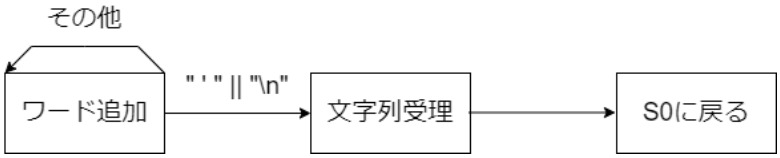

図 5　S3 内の処理

- S4 と S5 と S6、S4 と S5 と S6 は二文字可記号に対応する状態である。
  - S4、S4 は記号’>’ と’:’ に対応する。追加記号は同じ（’=’）なので、一つの状態にまとめる。
  - S5、S5 は記号’<’ に対応する。追加記号は’> と’=” である。
  - S6、S5 は記号’.’ に対応する。追加記号は’.’ である。

  S4、S5、S6 の実際の処理は図 6 に示される。

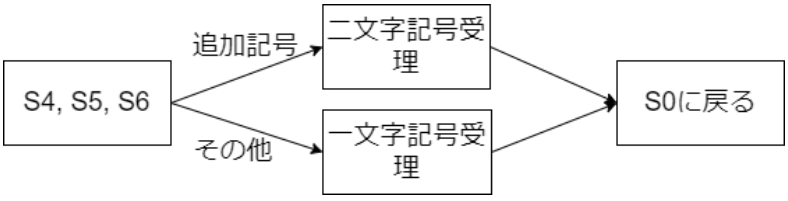

図 6　S4、S5、S6 内の処理

- S7、S7 はコメント用の状態である。S7 の実際の処理は図 7 に示される。

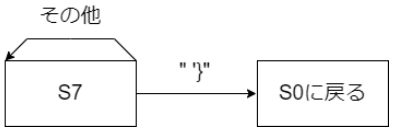

図 7　S7 内の処理

以上のオートマトンを使って、入力されたファイルの中身を文字列にして、そこから一文字ずつ取り出して、オートマトンに入れていくという流れになる。また、ここでの「受理」とはトークン表に確認して、出力ファイルに書き出すという処理を行うことである。

3　実装プログラム

実際のプログラムを説明するためにはトークン表の作り方、ファイルから文字列へ、オートマトン、ファイルに書き出し方という流れに分けて、それぞれ説明していく。

3.1　トークン表の作り方

トークンオブジェクトを作るために、新しいクラス (Token) を作った。Token の中身は図 8 に示される。

図 8　オブジェクト Token の中身

また、オブジェクトにコンストラクタを作り、トークンの ID とソースコードの字句とトークン名を引数として受ける。

以上のオブジェクトを用いて、トークンの一覧に示された 42 のトークンを add() メソッドを使って、arrayList に格納する。以上の処理は一つずつ入れていくしかないので、run() 関数を長くしないように、別の自作関数 (initalizeDictionary()) 関数に任せる。

3.2　ファイルから文字列へ

与えられた文字列のパスを Path.of() メソッドを使って、読み込んで、Path 型のパスを得る。また、そのパスを用いて、Files.readString() メソッドを使って、そのパスに指定されたファイルの中身を文字列として格納する。ただし、与えられた文字列のパスは実際に存在しない可能性があるので、try-catch に囲まれて、パスの読み込み処理を行う。

3.3　オートマトン

次に、得られたファイルの文字列から for 文と counter 変数を用いて、一文字ずつ取り出して（変数 c に格納して）オートマトンに入れていく。本プログラムではオートマトンは全部 run() メソッドで行う。また、簡単のため、図 1 に示された全ての状態名を新しい Type(State) として定義される。状態を表すために currentState という変数を宣言して、初期値として、State.S0 とする。ワードを格納するために、word という変数も宣言する。

次に、switch 文を用いて、currentState の値によりそれぞれの処理を行う。

1. S0、与えられた変数 c の値によって、それぞれの処理を行う。
   （a）c は英字の場合、S1 に移行して、word 変数にその文字を追加する。
   （b）c は数字の場合、S2 に移行して、word 変数にその文字を追加する。
   （c）c は一文字記号の場合、その記号を出力ファイルに書き出す。
   （d）c は” ’ ”の場合、S3 に移行して、word 変数にその文字を追加する。
   （e）c は”>”又は”:”の場合、S4 に移行して、word 変数にその文字を追加する。

（f）c は”<”の場合、S5 に移行して、word 変数にその文字を追加する。
（g）c は”.”の場合、S6 に移行して、word 変数にその文字を追加する。
（h）c は”{”の場合、コメントの始まりなので、word 変数に追加せずに、S7 に移行する。

2. S1、状態 S1 では取り出された文字が英字又は数字である限り、word にその文字を追加するが、そうでない場合は counter を 1 で減らして、出力ファイルに書き出して、状態 S0 に移行する。
3. S2、状態 S2 では取り出された文字が数字である限り、word にその文字を追加するが、そうでない場合は出力ファイルに書き出して、counter を場合により*1 で減らして、状態 S0 に移行する。
4. S3、状態 S2 では取り出された文字が”\n”でない限り、word にその文字を追加するが、””又は”\n”である場合は出力ファイルに書き出して、状態 S0 に移行する。
5. S4、状態 S4 では取り出された文字が”=”である場合、word にその文字を追加して、出力ファイルに書き出して、状態 S0 に移行する。そうでない場合は counter を場合により*1 で減らして、状態 S0 に移行する。
6. S5、状態 S5 では取り出された文字が”>”又は”=”である場合、word にその文字を追加して、出力ファイルに書き出して、状態 S0 に移行する。そうでない場合は counter を場合により*1 で減らして、状態 S0 に移行する。
7. S6、状態 S6 では取り出された文字が”.”である場合、word にその文字を追加して、出力ファイルに書き出して、状態 S0 に移行する。そうでない場合は counter を場合により*1 で減らして、状態 S0 に移行する。
8. S7、状態 S7 では取り出された文字が”}”である時だけ、状態 S0 に移行する。

*取り出された文字が改行である場合は、改行を表す変数 (lineNumber) にバグを発生する可能性があるので、その時に、counter を減らさない。改行以外は 1 で減らす。

最後に、word にまだ残っている文字列がある時に、その文字列を出力ファイルに書き出す。

3.4 字句のファイルへの書き出し方

取り出された字句は次に以下の処理を行う。

- 字句が数字や文字列の場合はそれぞれのトークンを生成して、ファイルに書き出す。
- そうでない場合は、3.1 章で生成されたトークン表を用いて、どのトークンに対応されるかを見つける。
  - 見つける場合、そのトークンを出力ファイルに書き出す。
  - 見つけない場合、識別子として新しいトークンを生成して、出力ファイルに書き出す。

ファイルへの書き出し方については FileWriter() と BufferedWriter() メソッドを用いて、指定されたフォーマットに従い、ファイルの各行に書き出していく。

4 考察・工夫点

C 言語と比べて、Java のプログラムは読み手にとっても、作成者にとっても分かりやすく作れる。例えば、状態は新しいタイプとして、定義できて、プログラム全体がもっとわかりやすくすることができることがわかる。また、自分の環境（windows）で正しく実行できるが、jenkins で改行のところでエラーが生じる。これはなぜかというと、windows の環境では改行は”\r\n”と表すが、linux では”\n\n”とあらわす。よって、windows では改行が一回だけ計算されるが、jenkins でのテストは二回計算されてしまう。そのバグを防ぐために、新しい関数（reduceCounter()）を宣言して、取り出された文字が改行でない時にだけ、counter 変数を 1 で減らす。

5 感想

本課題を通して、Java を復習して、constructor やオブジェクトをもう一度復習した。ただ、改行の場合を考えず、プログラムを jenkins で確認しようと思ったのだが、全テストを落ちてしまって、そこで解決する時間がかかってしまった。また、単体テストの存在も完全に忘れてしまったので、今度はそれを使って、プログラムを実際の jenkins で確認する前に、バグを見つけようと思う。