情報科学演習 D
課題 3

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: 09B20033

城間大幹

2022 年 12 月 8 日

1　仕様

意味解析器への入力は Pascal 風プログラムを字句解析した結果 (ts ファイル) であり，出力は意味的な誤りが含まれるかどうかの判定結果である．また，構文的な誤りが含まれるかどうか (課題 2 の内容) も同時に判定する．開発対象となる意味解析器のメソッドは Checker.run(String) である．当該メソッドの仕様は以下の通り．

- 第一引数で指定された ts ファイルを読み込み，意味解析を行う
- 意味的に正しい場合は標準出力に”OK”を出力する
- 意味的に正しくない場合は”Semantic error: line”という文字列とともに，最初の誤りを見つけた 行の番号を標準エラーに出力する
- 構文的な誤りが含まれる場合もエラーメッセージを表示する
- 入力ファイル内に複数の誤りが含まれる場合は，最初に見つけた誤りのみを出力する
- 入力ファイルが見つからない場合は標準エラーに”File not found”と出力して終了する

2　方針と設計

課題 2 の Parser.java のコードをベースにして, コードを多少の改変や追加により実装した．Checker では主に変数周りに関わる所が多かったため, スコープ, 名前, 型などを記録した変数のリストも変数宣言時に作成した．そしてそれらをもとに型の不整合や重複宣言を確認する関数を作成し, それを適切な箇所で呼び出すことにより, Semantic error を検出した．

3　実装プログラム

ここでは Checker で重要な変数のリストと, Semantic error を検出するために用いる, いくつかの関数について説明する．

3.1　変数リスト variable

変数宣言時にリストを作成している．variable は String 型の二次元配列であり, 下記のようにしてリストを作成する．

- variable[0] には変数のスコープを示し, 文字列 global, または副プログラムの手続き名を代入する
- variable[1] には変数の名称を示し, str[0] を代入する
- variable[2] には変数の型を示す．型は integer, char, boolean の 3 種類である
- variable[3] には, 変数が配列であれば文字列 array を代入する．配列でなければ null である

なお副プログラム文の引数は, 副プログラムの手続き名をスコープとしてリストに加えている.

3.2 関数 varDuplicationCheck

変数が重複して宣言されていないかを調べる関数である. 関数 varDuplicationCheck 呼び出し時の scope を満たす variable の要素の内, varName と一致するものがあれば変数の重複宣言とみなし, false を返す.

ソースコードでは関数 SIDENTIFIER 内で, 変数 => コロンと読み込んだ時に関数 varDuplicationCheck を呼び出し, 重複を確認する.

3.3 関数 substitutionTypeCheck

代入文の型の不整合を確認する関数である. varName が左辺の変数名, substitution が右辺の代入されるものの文字列, substitutionType が右辺のトークン名, 変数であればその型を示している.

まず variable を探索して, varName に合致するものの型を取得する. 次に右辺が変数であれば, 再度 variable を探索し, その変数の型を取得して比較する. 一方右辺が, STRUE, SFALSE であれば, varName に対応する型が boolean であるか, SCONSTANT であれば, varName に対応する型が integer であるか, SSTRING であれば, varName に対応する型が char であるかを判定し, 比較する. 比較の結果, 一致しなければ型の不一致とみなし, false を返す.

ソースコードでは関数 SIDENTIFIER 内で, SASSIGN(:=) を読み込んだ時に, 左辺と右辺を引数にして関数 substitutionTypeCheck を呼び出し, 型の一致を確認している.

3.4 関数 checkSCONSTANT

変数の型が integer 型かを調べる関数である. checkSCONSTANT 呼び出し時の scope を満たす variable の要素の内, varName と一致するものの型が integer でなければ, false を返す.

ソースコードでは関数 Brackets 内で, SLBRACKET([ ) を読み込んだ時に, 次に続く変数を引数として, 関数 checkSCONSTANT を呼び出して integer 型であるかを調べ, 配列の添字の型を調べている.

3.5 関数 varTypeCheck

変数の型を調べる関数である. varTypeCheck 呼び出し時の scope を満たす variable の要素の内, varName と一致するものの型を String 型で返す.

ソースコートでは関数 varTypeCheck は関数 SIDENTIFIER 内で呼び出され, 型の違う変数同士の代入を検出している.

また関数 calculation 内でも呼び出され, 型の違う演算子の演算も検出するが, ここでその方法について軽く説明する. 関数 calculation では演算子と被演算子を交互に読み取り構文誤りがないかを調べるが, 被演算子が変数である場合, 関数 varTypeCheck を用いて型を判定し, integer 型であれば integerType フラグを, char 型であれば charType フラグを, boolean 型であれば integerType フラグを立てる. SSTRING に対応する stringType フラグを含め, 一つの演算で, integerType+charType+stringType+booleanType>1 となれば型の不一致が起きた演算であるとわかる.

3.6 関数 procedureCheck

副プログラムの手続き名が呼び出されているかを判断する関数である. リストに追加する変数はコロン, セミコロンの間にある SIDENTIFIER と想定しているため, 副手続き名もリストに入ってしまう. すなわち変数リスト variable にスコープが副手続き名で, 変数名も副手続き名である変数が入ってしまうということだ. したがって関

数 SIDENTIFIER 内で関数 procedureCheck を用いることにより, こうした場合を除外する.

3.7 関数 arrayCheck

変数が配列であるかを調べる関数である. arrayCheck 呼び出し時の scope を満たす variable の要素の内, varName と一致するものの variable[i][3] が array であれば true を返す.

関数 SIDENTIFIER 内などで変数を読み込んだ際に呼び出し, 配列であれば arrayflag を立てる. そして arrayflag=1 であるときに, 次の文字が SLBRACKET([ ) であるかを判定することで, 配列が適切に使用されているかを判定できる.

4 考察や工夫点

意味解析を行う上で変数の扱いは非常に重要であるため, 二次元配列 variavle と, scanner で読み取る str から Pascal 文のスコープを示す, 変数 scope をグローバル変数にすることで, 常に更新し最新の状態を維持した.

また課題 2 と同様に, 変数周りの意味的な誤りを確認するために, それぞれ関数を作成し, その名前も一目見てどういうものか分かるようにした.

5 感想

課題 2 でブロック化した構文定義を関数にして表すという方針のもと開発したため, 課題 3 の実装では元のコードをあまり変更することなく, 比較的簡単に実装できた. またグローバル変数を用いることで, 関数間の連携をうまく取ることができ, 課題 2 で作成した関数をさらに生かすことができた.

6 謝辞

終始熱心なご指導を頂いた情報科学演習 D の松本 真佑, 桝井 晃基担当教員, また TA の方々には心より感謝の意を表します.