Parser and AST Documentation

Anurag Kashyap, Mia de Leon, Austin Kao

February 22, 2021

Description

This is the segment of the compiler which parses the output of the lexer and forms the Abstract Syntax Tree.

Overview

We use some nonterminals in our grammar whose usage is as follows:
fieldlist: Used to generate the fields in a record instantiation (not declaration)
funcsarg: Used to generate the arguments passed to a function call (not declaration)
expseq: Used to generate a sequence of expressions inside a Let block
lvalue:Generates all the lvalues
tyfields: Generates the list of type-annotated parameters for a function declaration or record type declaration
ty: Generates type declarations of all kinds permitted in Tiger

The semantic actions for all these nonterminals are trivial.

Precedence and Associativity

THEN is right associative because in nested *if then else* blocks, the last *else* must chain with the nearest *then*

ELSE if right associative because we want the parser to shift right and pick the first of the two options below whenever possible:
1. IF exp THEN exp . ELSE exp
2. IF exp THEN exp .

MINUS and DIVIDE associate are left associative to follow the rules of basic arithmetic, i.e. $10 - 5 - 2 = (10 - 5) - 2$ and $10/5/2 = (10/5)/2$

Similarly, logical operators are also left associative.

Comparison operators have no associativity and are parsed after their arguments are completely parsed.

Declarations

Mutual Recursion

fundec: It has a right-recursive production which lets us create a sequence of mutually recursive functions, followed by the tail *restdecfun*
tydec: It has a right-recursive production which lets us create a sequence of mutually recursive types, followed by the tail *restdecty*
restdecty: Generates any possible sequence of declarations that don't start with a type declaration. This helps us demarcate where a bunch of possibly mutually recursive type declarations end.
restdecfun: Generates any possible sequence of declarations that don't start with a function declaration. This helps us demarcate where a bunch of possibly mutually recursive function declarations end.
decs: Generates a sequence of declarations
vardec: Does not require any mutual recursion. Escape is set to false by default.

Helper Functions in semantic actions

addTyDecList : This function takes a list of declarations and merges contiguous type declarations.
addFunDecList : This function takes a list of declarations and merges contiguous function declarations.

Error Recovery

We do not perform any error recovery in our parser.