Cool Abstract Syntax Trees

1 Cool Constructors

The Cool AST is made up of nodes with the following constructors:

```plaintext
class constructor program(classes : Classes) : Program;
class constructor class_decl(name : Symbol; parent: Symbol; features : Features; filename : Symbol): Class;
class constructor method(overridep : Boolean; name : Symbol; formals : Formals; return_type : Symbol; expr : Expression) : Feature;
class constructor attr(name, of_type : Symbol) : Feature;
class constructor formal(name, of_type: Symbol) : Formal;
class constructor branch(name, local_type: Symbol; expr: Expression): Case;
class constructor assign(name : Symbol; expr : Expression) : Expression;
class constructor static_dispatch(expr: Expression; type_name : Symbol; name : Symbol; actuals : Expressions) : Expression;
class constructor dispatch(expr : Expression; name : Symbol; actuals : Expressions) : Expression;
class constructor cond(pred, then_exp, else_exp : Expression): Expression;
class constructor loop(pred, body: Expression) : Expression;
class constructor typecase(expr: Expression; cases: Cases): Expression;
class constructor block(body: Expressions) : Expression;
class constructor let(identifier, local_type: Symbol; init, body: Expression): Expression;
class constructor add(e1, e2: Expression) : Expression;
class constructor sub(e1, e2: Expression) : Expression;
class constructor mul(e1, e2: Expression) : Expression;
class constructor div(e1, e2: Expression) : Expression;
class constructor neg(e1: Expression) : Expression;
class constructor lt(e1, e2: Expression) : Expression;
class constructor leq(e1, e2: Expression) : Expression;
class constructor comp(e1: Expression) : Expression;
class constructor int_lit(token: Symbol) : Expression;
class constructor bool_lit(value: Boolean) : Expression;
class constructor string_lit(token: Symbol) : Expression;
class constructor alloc(type_name: Symbol): Expression;
class constructor unit(): Expression;
class constructor nil(): Expression;
class constructor no_expr(): Expression;
class constructor variable(name: Symbol): Expression;
```

The types of the constructor parameters are other nodes, or sequences of nodes except for the simple types Boolean and Symbol. When drawing an AST, the only nodes drawn should be instances of these constructors:
- Do not use a “role” (such as “body”) as a node name; it gives the relation between a node and its parent. It is not a constructor.

- Do not use a “phylum” (such as “Expression”) as a node name; it is a node type, not a constructor.

- Do not draw symbols as nodes. The type “Int” is not a node. The name “foo” is not a node. These are symbols associated with a node, not a child of that node.

In parsing Cool, some syntactic sugar is removed. See the Cool manual for the precise definition, but here is the summary:

- The class parameters and initializers are captures in a method named the same as the class which returned “this.”

- Constructor calls are converted into a dispatch on an alloc node.

- A local variable with initialization is converted into a let node with the rest of the block being the body of the let.

- An receiver of this is used if there is no explicit receiver.

2 Example

```{var i: Int = j/2;
   i = i*3;
   print(i)
}
```

```
let(i, Int)
    ↘
   div
      ↘
     var(j)
       ↓
      int_lit(2)
        ↓
       assign(i)
         ↓
        dispatch(print)
          ↘
         var(this)
           ↓
          var(i)
            ↓
         int_lit(3)
```

The following aspects are often done incorrectly by students:

- The let node has the initialization and body as its two children. The initialization is not an assignment!

- Only the last two parts of the block are in an AST block.

- An assign node has only one child.

- The implicit receiver this should be made explicit.