1 Cool Constructors

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

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

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

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.