CMSC 22600 **Compilers for Computer Languages** **Homework 2**
Autumn 2016 **Due Nov 22**

This homework assignment is a written assignment about parsing and typechecking. Please turn in your completed homework at the **beginning** of class on Tuesday, November 22.

1. Consider the following augmented grammar G of expressions.

$$\begin{array}{lcl} S' & \rightarrow & E \\ E & \rightarrow & \textbf{id} \\ E & \rightarrow & \textbf{id}\ \textbf{(}\ E\ \textbf{)} \\ E & \rightarrow & E\ \textbf{*}\ \textbf{id} \end{array}$$

   (a) Calculate First and Follow for G.

   (b) Build the LR(0) DFA (*i.e.*, states and goto edges) for the grammar. Your answer should clearly define the set of LR(0) items for each state and include a diagram of the DFA.

   (c) Give the LR(0) action and goto tables for the grammar (remember that the goto table is different from the goto edges!). Is this grammar LR(0)? If not, why?

   (d) Is this grammar SLR? If not, why?

   (e) Is this grammar LR(1)? If not, why?

2. Recall the discussion in Handout 4 (*Basic Polymorphic Typechecking*). Assume that we have both `int` and `real` as base types. To extend the typechecker to support overloaded functions (*e.g.*, "+") on integers and reals, we need to allow type variables that are restricted to be members of some set. For example, the type of "+" could be written as

$$\forall\alpha \in \{\mathtt{int}, \mathtt{real}\}.(\alpha \times \alpha) \rightarrow \alpha$$

   We can model this semantics by changing the representation of type-variable kinds:

```
and tvar_kind
  = INSTANCE of ty
  | UNIV of int
  | NUMKIND
```

   Give a modified version of the destructive unification algorithm given in Figure 5 that deals with this new kind representation. The idea is that if a type variable has `NUMKIND` kind, then it can only be unified with other type variables or with numeric base types.