Adding type classes to Helium

Jeroen Weijers - 0422894 (jweijers@cs.uu.nl)

In this short report I'll briefly discuss the changes made to the Helium compiler for adding type classes. Currently type classes are supported by the parser, and the static checks. The type-inferencer and code generator still need to be adapted.

Changes in the helium compiler

Parsing phase

parser/Parser.hs:
The topdecl parser is extended with two new sorts of top level declarations, namely class and instance.
The class and instance both contain some declarations. These are parse by cdecls and idecls, both are restricted versions of idecls.

A class may contain:
- Type signatures
- Function declarations

In an instance only functions can be defined that were declared in the class. According to the Haskell 98 specification it is not allowed to provide type-signatures for these functions (these were already given in the class definition).

parser/ParseLibrary.hs:
Added the instance keyword to the lexer.

Static analysis

syntax/UHA_Utils.hs:
Added helper function that extracts the name from a declaration.

staticanalysis/staticchecks/Collect.ag:
A lot of new attributes that collect and distribute information on classes and instances are declared in the Collect.ag, most are used for constructing the export environment.
The exported environment from a module contains the class environment that was collected in the module as well as an environment providing information on the members of a module.
These two environments are collected by the following attributes:
The synthesised attribute for Body, Declaration and Declarations: classEnv
The synthesised attribute for Body, Declarations and Declaration: collectClassMemberEnv

The collectClassMemberEnv attribute is combined with the collectClassMemberEnv of all the imported modules and inherited by all declarations in the module.

Context items have a synthesised attribute typeVariables containing all type variables that occur in the context.

modulesystem/ImportEnvironment.hs:
Added two fields to the import environment: classEnvironment and classMemberEnvironment. The classEnvironment is a class environment as defined by Top. The ClassMemberEnvironment is a map that provides a list of all of the class members and if there is a default definition defined for that member.

staticanalysis/staticchecks/MiscErrors.ag:
In this ag file all static errors related to typeclasses are generated. The checks performed in this module are:

- A function definition without a type signature is not allowed in a class
- A function cannot be declared in multiple classes
- A type class can only have one function parameter
- An instance declaration containing type variables may only use each type variable once. For example an instance for tuples (a, a) is not allowed, this should be (a, b). The occurrence of multiple type variables is not a form of multi-parameter type classes, we instantiate the type class for just one type namely the tuple type containing an a and a b.
- The context of a class member may not put further restrictions on the class variable
- The context of a class may only use the class type variable
- All type signatures in a class must use the class type variable
- A class must have a unique name
- A class may not have a name that is equal to the name of an imported class
- A class member may not have a name that is already a top-level declaration
- A class instance cannot be instantiated with a type synonym
- Instances of classes that are not declared are not allowed
- An instance may not contain functions that were not declared by the class
- An instance type must be a simple type
- Instances are only allowed if instances exist of all super classes defined in the class
- The context of an instance may only use a type variable that occurs in the instance type
- Overlapping instances are not allowed

staticanalysis/messages/StaticErrors.hs:
All new type class related error messages were added here, also the error message associated to them are defined here.

staticanalysis/staticchecks/Warnings.ag:
A warning was added for missing declarations in an instance, when no default definition was provided by the Class.

staticanalysis/messages/Warnings.hs:
The warning and message asociated to it, on missing declarations on instances was added here.

Remaining tasks

In order to enable type classes in the helium compiler the type-checker/-inferencer and the code generator need to be changed.
In the type checker most work has to be done in: *staticanalysis/inferencers/ TypeInferenceRules.ag.* For classes and instances the attribute binding group needs to be defined, as well as constraints and assumptions.
Type checking classes is done by checking the types of the default functions provided in the class. When type checking a default function for a class *C* with type variable *a*, the context of all type-signatures in the class is extended with *C a.* When type-checking

instances the type variable of the class has to be replaced with the instance type in all type signatures of the class. The context off all type signatures also need some additions.
As patterns are not allowed in type classes and instances the *PatternMatchWarnings.ag* also has to be adapted to prevent unnecessary warnings from being displayed.