HW5 – Object-Oriented Languages

CS 476, Fall 2021

1 Instructions

Begin by downloading the file hw5-base.ml from the course website and renaming it to hw5.ml. Then fill in your answers to the problems, adding or modifying definitions as you see fit. Submit your completed hw5.ml via Gradescope. As always, please don’t hesitate to ask for help on Piazza (https://piazza.com/class/ksknvqg6ogb2kc).

2 Typechecking Object-Oriented Programs

In the second half of HW2, you wrote a typechecking function for arithmetic and boolean expressions. In this homework, you’ll fill in pieces of a similar typechecker for a simple object-oriented language.

The file hw5-base.ml defines the types exp of expressions and cmd of commands for a simple Java-like language. It also defines two core functions: type_of, which takes an expression and returns its type, and typecheck_cmd, which takes a command and checks whether it is well-typed (returning true if it is and false otherwise). Both type_of and typecheck_cmd also take a class table ct, which maps class names to their definitions, and a type context gamma, which holds the types of variables. Mathematically, type_of ct Γ e should return Some t exactly when Γ ⊢ e : t, and typecheck_cmd ct Γ c should return true exactly when Γ ⊢ c : ok. The following problems will ask you to complete the implementation of these two functions.

The file also includes the following helper functions:

- fields, which takes a class table and a class, and returns the list of fields of that class (including those defined in superclasses)
- methods, which takes a class table and a class, and returns the list of methods of that class (including those defined in superclasses)
- types_of_params, which takes a list of parameters/field definitions and returns just their types
- field_type, which takes a class table, class, and field name, and returns the type of that field of the class, if it exists
- lookup_method, which takes a class table, class, and method name, and returns the declaration of that method of the class, if it exists
• **typecheck_list**, which takes a class table, a type context, a list of expressions, and a list of types, and returns true if each expression in the list has the corresponding type according to **type_of**.

1. (5 points) Implement the **subtype** function, which takes a class table `ct` and two types `t1` and `t2`, and returns **true** if `t1` is a subtype of `t2` and **false** if it is not. `t1` is a subtype of `t2` if either: 1) `t1` and `t2` are the same type, or 2) `t1` is a class type, and its superclass is a subtype of `t2`.

   Once you have completed this problem, `subtype ct0 (ClassTy "Square") (ClassTy "Object")` should return **true**.

2. (5 points) Extend the provided **type_of** function with a case for **GetField**, the field access expression, according to the following rule:

   \[ \Gamma \vdash e : C \quad field_{\text{type}} C \ f = \tau \]

   \[ \Gamma \vdash e.f : \tau \]

   Note that the class table `ct` is implicit in this and the following rules; it does not appear in the rules, but it still needs to be passed as an argument to any relevant functions.

   Once you have completed this problem, `type_of ct0 gamma0 exp2` should return **Some IntTy**.

3. (5 points) Extend the provided **typecheck_cmd** function with a case for **New**, the object creation command, according to the following rule:

   \[ \Gamma(x) = \tau_0 \quad \text{fields } C = \tau_1 f_1, \ldots, \tau_n f_n \quad \Gamma \vdash e_1 : \tau_1 \quad \ldots \quad \Gamma \vdash e_n : \tau_n \quad C \ <: \tau_0 \]

   \[ \Gamma \vdash x := \text{new } C(e_1, \ldots, e_n) : \text{ok} \]

   Note that we can assign the new object of class `C` to `x` as long as `C` is a subtype of the type of `x` (written `C \ <: \tau_0` in the rule). The function **types_of_params** can be used to extract the types from the list of fields of a class, and **typecheck_list** can be used to check whether a list of expressions matches a list of types.

   Once you have completed this problem, `typecheck_cmd ct0 gamma0 cmd3` should return **true**.

4. (for graduate students) Extend the provided **typecheck_cmd** function with a case for **Invoke**, the method invocation command, according to the following rule:

   \[ \Gamma(x) = \tau_0 \quad \Gamma \vdash e : C \quad \text{lookup\_method } C \ m = \tau \quad m(\tau_1 x_1, \ldots, \tau_n x_n) \]

   \[ \Gamma \vdash e_1 : \tau_1 \quad \ldots \quad \Gamma \vdash e_n : \tau_n \quad \tau \ <: \tau_0 \]

   \[ \Gamma \vdash x := e.m(e_1, \ldots, e_n) : \text{ok} \]

   Once you have completed this problem, `typecheck_cmd ct0 gamma1 cmd4` should return **true**.