TU Dortmund University – Software Engineering – Prof. Dr. Jakob Rehof
Anna Vasileva

# Exercise 4: Combinator Implementation in Scala

1. Download and unpack the exercises
   `http://www-seal.cs.tu-dortmund.de/seal/downloads/teaching/warsaw17/Exercise4.zip`

2. Import the project into IntelliJ via `File -> New -> Project From Existing Sources`. Select `Import project from external model -> SBT`. The other settings don't need to be changed.

3. You find the combinators in:
   `core/src/main/scala/org/combinators/guidemo/repository.scala`
   The definitions in this file correspond to the setting known from ex. 3:

$$\begin{aligned}
&\mathrm{WF} = \{\{\alpha \mapsto \mathit{DropDown}\}, \{\alpha \mapsto \mathit{RadioButtons}\}\} \\
&\Gamma = \{ \quad \texttt{customerForm} : (\texttt{String} \to \texttt{java.net.URL} \to \texttt{OptionSelection} \to \texttt{Form}) \cap \\
&\qquad (\mathit{Title} \to \mathit{Location}(\mathit{Logo}) \to \mathit{ChoiceDialog}(\alpha) \to \mathit{OrderMenu}(\alpha)) \\
&\qquad \texttt{dropDownSelector} : (\texttt{java.net.URL} \to \texttt{OptionSelection}) \cap \\
&\qquad (\mathit{Location}(\mathit{Database}) \to \mathit{ChoiceDialog}(\mathit{DropDown})) \\
&\qquad \texttt{radioButtonSelector} : (\texttt{java.net.URL} \to \texttt{OptionSelection}) \cap \\
&\qquad (\mathit{Location}(\mathit{Database}) \to \mathit{ChoiceDialog}(\mathit{RadioButtons})) \\
&\qquad \texttt{companyTitle} : \texttt{String} \cap \mathit{Title} \\
&\qquad \texttt{databaseLocation} : \texttt{java.net.URL} \cap \mathit{Location}(\mathit{Database}) \\
&\qquad \texttt{logoLocation} : \texttt{java.net.URL} \cap \mathit{Location}(\mathit{Logo}) \\
&\qquad \texttt{alternateLogoLocation} : \texttt{java.net.URL} \cap \mathit{Location}(\mathit{Logo}) \quad \}
\end{aligned}$$

   At the top of the file you find the kindig declaration for variable $\alpha$, WF in mathematical notation. Than the combinator definitions follow, where every combinator has an `apply` method annotated by its native type. It also has a semantic type. Your first task is to fill in the semantic type of combinator `customerForm` (replace `???`).
   In Scala syntax constants and type constructors have to be prefixed by single quotation mark: $\mathit{Location}(\mathit{Database})$ becomes `'Location('Database)`. Arrows are represented by `=>:`, e.g. $\sigma \to \tau$ becomes `sigma =>: tau`. Intersections are represented by `:&:`, e.g. $\sigma \cap \tau$ becomes `sigma :&: tau`.

4. You find the inhabitation request in:
   `core/src/main/scala/org/combinators/guidemo/productline.scala`
   `lazy val resultsFromRequests: Results = Results.add(Gamma.inhabit[Form](Omega))`
   specifies the request $\Gamma \vdash ? : \texttt{Form} \cap \omega$
   To run the request select `Run -> Run... -> Edit Configurations...`, click on the + in the upper left, select `SBT Task`. On the right side enter name `run` and tasks: `inhabitation-demos/run`. Under `Before launch: Activate tool window` select `Build` and click on − to remove it. Then click on the `Run` button. Open `http://localhost:9000/guidemo` in your web browser to see the inhabitation results. Click on `Compute` for any variation to get the combinatory term.
   To obtain the synthesized result in IntelliJ click on `File -> New -> Project form Version Control -> Git` and use `http://localhost:9000/guidemo/guidemo.git` as Git Repository URL. In the new project open `build.sbt` and select `import project` (top right). To test your

result open
`src/main/java/org/combinators/guidemo/CustomerForm.java`
right click in to the code and select `Run 'CustomerForm.main()'` To test different variations compute them in the browser, right click on the project in IntelliJ, select `Git -> Repository -> Fetch` and then `Git -> Repository -> Branches`, where each branch corresponds to one variation.

5. Inspect the different variation closely. What is wrong with some of them?
   Change `Omega` in:
   `lazy val resultsFromRequests: Results = Results.add(Gamma.inhabit[Form](Omega))`
   to a more specific semantic type avoiding these problems.

6. Implement a new combinator that inserts your university logo from `http://en.uw.edu.pl/wp-content/themes/uw/library/img/logo-pi.png`.

7. **(Optional)** Add semantic types such that you can specify the logo in the request. Hint: declare a new variable
   `lazy val beta = Variable("beta")`
   update the kinding

   ```
   lazy val kinding: Kinding =
       Kinding(alpha)
         .addOption('DropDown).addOption('RadioButtons)
         .merge(Kinding(beta).addOption(???).addOption(???))
   ```

   replacing `???` appropriately, and use `beta` in your combinators.