Project CP 4 – CS 536: Programming Language Design
Instructor: Rose Bohrer (pronouns: she/her)
Time: T,Th | 4:00 PM – 5:20 PM
Classroom: Higgins Labs 114
Instructor Office: Fuller Labs 139

Overview

This course is an ungraded, project-based, team-based course. This means that you identify a team, design your own project together, and assign yourself a grade based on how well your project meets the course goals. In understanding the role of assignments, we should distinguish "structure" from "requirements." If you already know how you want to run your project or what you want to do, you should do so – be as radical as you like. You choose how to engage with this structure, but these homeworks provide you a structure so that you have lots of guidance as your starting point.

When I review your submissions, I **only** provide formative feedback, never a grade. You are the person who assigns your final grade, you are in control.

Goals

- Make significant implementation progress
- Make initial report-writing progress
- Demonstrate understanding of operational semantics and type systems

Logistics

Teams are at least 3 people. There is no upper limit on size. It is recommended to keep the same team all semester. Checkpoints are due at the times indicated on the Schedule section of the syllabus.

As a group, fill out the form on the following pages and submit it. Submit all files for your (work-in-progress) project and submit (the beginnings of) your project report. Submit one copy for your whole group. Many students save their copies of the individual forms, to help with end-of-term self-evaluations.

Task Instructions

- Before the meeting:
  - **Contact the professor to schedule a meeting with your team**
  - Do the work you said you would do
  - Fill out the pre-meeting form
- At the meeting:
  - Follow the meeting form
  - Take notes
  - Review exercises
  - Review project progress
  - Help each other with both of those things
  - Review the status of your project files. Do your best to have it “presentable” for feedback. I.e., does it compile and run, and can I do that easily?
  - Read the handout to see what is planned next week.
  - Decide what work each person will do over the next week.
  - Submit your work on Canvas (one person, on behalf of group)
    - Make sure you submit your project files. If they’re hosted in a (e.g. GitHub) repository, include a link to it every time.
    - Make sure to submit your writeup draft

Exercises

Week 8: Operational Semantics

1. There are lots of definitions this week. Write a glossary, i.e., write definitions of the following terms from Lecture 15 and 16:
    a. **Value**
    b. **Expression**
    c. **Definition**
    d. **Judgement**
    e. **Inference Role**
    f. **Small Step Semantics**
    g. **Big Step Semantics**
    h. Premise
    i. Conclusion
    j. Side Condition
    k. (Lecture 16 starts here)
    Environment

And Lecture 16:

2. Recall the program hierarchy "Well-formed", "Executable," "Well-typed," "Correct" from Lecture 15 (ignoring the "Happy" programs). For each of the following Scala programs, where does it fit in the hierarchy? Give the most specific answer, e.g., don't put "Well-formed" for a "Well-typed" program. If it belongs to none of these places in the hierarchy, put "Not well-formed"
    a.
```
// Compute the square of x
def sqr(x : Int): Int = x*x
```
    b.
```
def mystery(areaCode : Int) : String = {
  val phoneNumber : Int = areaCode - 555 - 1234
  // Call the phoneNumber and enter 1 at the prompt ??? (jocular)
  phoneNumber(1)
}
```
    c.
```
 -*C / \ /* ee&*
```
    d.
```
val x : Int = 1 / 0
```
    e.
```
// Compute the reciprocal of square root of x
def rsqrt(x : Int): Int = x*x
```
3. For each of the following expressions, is it (a) a declaration/definition (b) a value, or (c) an expression that is not a value?
    a. 1-2
    b. "1-2"
    c. val x = "1-2"
    d. 0.5
    e. 1.0 / 2.0
    f. false

4. We introduced the following judgements: e |-> e, e |->* e, e = e, and e⇓v (In plain text, you can just write e⇓v as "e EvaulatesTo v"). I gave this as an example of a rule that combines two of those judgements:

$$\text{StepsEq}\ \frac{e1\ \text{|->*}\ e2}{e1 = e2}$$

Write a new rule of your own that relates at least two of the judgements:

YourRule ----------

Lecture 16 Starts Here:

This lecture covers environments and errors.

5. Write the Eval rule for evaluating a function applied to an argument

6. Every expression needs error-propagation rules that say a large program fails when subprograms fail, for example:

$$\text{EPropOpL}\ \frac{E\ \text{|-}\ e1\ \text{error}}{E\ \text{|-}\ (e1\ \text{op}\ e2)\ \text{error}} \qquad \text{EPropOpR}\ \frac{E\ \text{|-}\ e2\ \text{error}}{E\ \text{|-}\ (e1\ \text{op}\ e2)\ \text{error}}$$

Write the error propagation rules for **let** (there are 2)

---------- ----------

Week 9: Type Systems (Covering Tuesday's lecture, since assignment is due Thursday)

1. When programmers say "type" they might mean either static or dynamic types. Which one do type theorists (almost always) mean when they say type?
2. "Types are compositional" means we can build up big types using connectives such as function types (->). To show that types are compositional, write out three different types using ->.
3. In a sentence, what is the relationship between a type context and an environment?
4. In lecture, we had separate typing judgements for expressions vs. definitions, so giving a type to a function definition required two rules:

$$\textbf{DefFun}\ \frac{\textbf{C,(x:t1) |- e : t2}}{\textbf{C |- def f(x:t1):t2 = e : (f:t1->t2)}}$$

And

$$\text{TyDef}\ \frac{C1 \vdash d : C2 \quad C1, C2 \vdash e : t}{C1 \vdash \text{let } d \text{ in } e : t}$$

But that's not the only way we could design a working type system. Instead, we could just have one judgement for expressions and no separate judgement for declarations.
Write a new typing rule for expressions with function definitions, i.e., expressions of form
"let f(x:t1):t2 = e1 in e2"

Before-Meeting Form

Fill this out to prepare for your meeting

1. **At the last meeting, what project work did you assign yourself?:**

   _____

   _____

2. **What went well in your own work? If you feel off track, what can the team do to get on track?:**

   _____

3. **As a whole, is the team on track? If not, what can you do to get on track?**

   _____

4. **If there's anything course staff could do to help my team, bring that up in the team meeting**

Student Structured Meeting Form

Meet with all your teammates, follow the meeting instructions, and fill out the form.

5. **Discuss the lecture exercises + answers with each other. Write notes from your discussion here. Any sources of confusion? Were they resolved? Any questions you'd like to ask course staff?:**

   _____

   _____

6. **Update (e.g. during meeting) your report document based on these forms and your conversations.**
7. **Discuss what each team member planned to do, what they did well, and where they need help:**

   _____

   _____

8. **Now help them, and/or get help from course staff.**
9. **Prepare your submission**
10. **I want to support you. A great way to do this is if you proactively ask questions and share comments/concerns with me on each checkin, at least once per CP, so I can give you answers in feedback. Put questions/comments here.**

    _____

    _____

11. **What do you (each) plan to do on your project in the next week?**

    _____

    END OF FORM

Instructor Feedback Rubric

I will use this rubric for giving feedback:

- What questions and concerns can I answer for the students?
- Was the work detailed enough that I felt I could provide good feedback? If not, your main feedback will be to increase the level of detail
- I read the answers to the lecture exercises. What mastery was demonstrated? What areas should they review, if any?
- Did they help each other understand the lecture exercises?
- Are they working well together?
- Did they say what each individual team member will do in the next week?
- I will give feedback on core course skills:
    - Did you correctly use regular expressions and context-free grammars to define your language?
    - Did you submit a program that compiles and runs and works on >=1 tiny example?
    - If not, did you document your problems and show me your plan going forward?
    - How has your implementation matured since last time? E.g., bug fixes, new features, better examples or documentation.
    - I will highlight students' strengths and progress on the above points.
    - Highlight students' strengths and progress on the above points.