Programming Languages: Project 2

This document describes the second project for the course Programming Languages (2020/21).

This file might be updated before the deadline (last change: 24 May 2021, v0.1). Any changes that occur will be announced in Slack.

Quick Summary

- Goals:
  1. Extend the Simply Typed Lambda Calculus studied with additional features
  2. Implement a typechecking algorithm
  3. Implement the operational semantics defined in point 1. as a function
- Deadline: **7 June 2021**
- To be done in groups of three students
- Submission is via Fénix (see instructions below)
- If you have any questions, please do not hesitate to contact João. You are encouraged to ask questions in the course's Slack (channel `#projects`)!

Tasks

0. Download the Project Pack from Fénix

You must work on the provided project pack (`P2-PL.zip`). The pack contains files that were taken/adapted from the Software Foundations book (Volume 2). You are required to change the following files:

- `MoreStlc.v`: the extensions to the STLC (and associated proofs) will be encoded in this file
- `Typechecking.v`: the typechecking algorithm (and associated proofs) will be encoded in this file
- `README.md`: file that you should fill in with your group's information and contribution

The files are marked with TODO comments to identify the places where you are supposed to work. You are also required to add information to the `README.md` file.

You can use the Makefile to compile the project (but note that it will only compile after you complete some of the tasks). If you add new files, you should edit the file `_CoqProject` and regenerate the Makefile (see the official documentation)

1. Extend the Simply Typed Lambda Calculus

You are required to extend the STLC as described in the file `MoreStlc.v` (module `STLCExtended`). Besides the features mentioned in the original chapter, you are required to extend the STLC with a new binary operator that allows non-deterministic choice between two programs. If we denote that operator as `!!`, then `c1 !! c2` is a program that chooses non-deterministically between `c1` and `c2`.

In particular, you should edit the file `MoreStlc.v` and perform the following tasks:

1. Do the exercise `STLCE_definitions`, extending the syntax and notations to support non-deterministic choice, and completing the formalizations of substitution, reduction, and the typing relation.
2. Go through the exercise marked as `STLCE_examples` and ensure that all examples work (if they don't, it might be due to errors in your definitions). Add one example (as a `Module`) for non-deterministic choice in the same style as the examples already present (including typechecking and reduction examples).
3. Prove the listed properties of typing by solving the exercises marked as `STLCE_progress`, `STLCE_subst_preserves_typing`, and `STLCE_preservation`.

2. Typechecking algorithm

Extend the typechecker provided to deal with the extended features mentioned above. In particular, you should edit the file `Typechecking.v` and perform the following tasks:

1. Do the exercise `typechecker_extensions` and complete the function `type_check`
2. Provide an example for each of the new cases that you implemented
3. Prove soundness and completeness, by completing the proofs of `type_checking_sound` and `type_checking_complete`

3. Operational semantics as a Coq function

Do the exercise `stlc_step_function`, where you are expected to:

1. Encode the operational semantics as a function.
2. Prove soundness and completeness (i.e., prove the properties `sound_stepf` and `complete_stepf`).

Note that non-determinism will cause some difficulties. You can leave this case unimplemented. However, you are required to include an explanation of how it could be done. Please also consider the extras below.

4. Extras

You are encouraged to extend your work with more features. In terms of grades, the extensions might only be considered if everything else was attempted.

Here are some suggestions for extra features:

1. Fully implement the case of non-determinism in the function described in Task 3.
2. Extend the STLC with even more features such as records and references (and with subtyping). See chapters of the Software Foundations book (volume 2).
3. Do the exercise `stlc_impl` included in the file `Typechecking.v`
4. ... Be creative! ;)

Submission

The project is due on the **7th of June, 2021**. You should follow the following steps:

- Submit only one file per group. Make sure your submitted file is named `P2-PL-GNN-2021.zip`, where `NN` is the group number. Always use two digits (e.g., Group 8's submitted file should be named `P1-PL-G08-2021.zip`)
- `PL-P2-GNN-2021.zip` is a zip file containing the solution and a `README.md` file filled in as required
- Upload the file to Fénix before the deadline.

Assessment

To assess your submission, the following grid will be used:

Task	Marks (max)
README file properly filled in	0,5
Task 1	
Extend syntax and notation	1
Substitution	2
Reduction relation	2
Typing relation	2

Task	Marks (max)
STLCE_Examples (non-determinism)	1
Proof: STLCE_progress	1,5
Proof: STLCE_subst_preserves_typing	1
Proof: STLCE_preservation	1
Task 2	
Typechecker function	2
Examples	1
Sound and completeness	2
Task 3	
Function `stlc_step_function`	2
Sound and completeness	1

If any of the above items is only partially developed, the grade will be given accordingly. If you are unable to finish a proof, you can hand in partially developed proofs by using `admit` or `Admitted`.

You are encouraged to comment your submission, so that we can understand your decisions. You might get additional points for that (e.g., if you describe in a comment exactly what needs to be done, even though a proof is incomplete).

Other Forms of Evaluation

After submission, you may be asked to present individually your work or to develop the solution of a problem similar to the one used in the project. This decision is solely taken by the teaching team.

Also, students whose grade in the test is lower than this project grade by more than 5 may be subject to an oral examination.

Fraud Detection and Plagiarism

The submission of the project assumes the commitment of honour that the project was solely executed by the members of the group that are referenced in the files/documents submitted for assessment. Failure to stand up to this commitment, i.e., the appropriation of work done by other groups or someone else, either voluntarily or involuntarily, will have as consequence the immediate failure of all students involved (including those who facilitated the occurrence).