

Homework # 11

due November 18

1 Reading

Please read Chapters 20 and 21 in your textbook.

2 Problems

Please do the following problems

- Exercise 20.1.2
- Exercise 20.2.1 (convert your code for the previous question)
- Exercise 21.2.2
- Exercise 21.3.4

Please check your answers against the solutions in the book. Do not turn in your answers.

3 Proofs

Prove the type soundness of recursive types using SASyLF. Start with the skeleton file `homework11.slf.SKEL`.

4 Modelling recursion

Using the `fullisorec` checker, *WITHOUT* the “fix” keyword, implement the recursive plus function (from Homework #7). Put your result in file `myfix.f`. Demonstrate that it works adding 2 and 3.

NB: The textbook (page 273) gives the wrong definition of `fix`: it will cause nontermination. (Furthermore it assumes equi-recursive types. Your code needs to work with iso-recursive types.) You need to use the call-by-value version on page 65. A major part of this question is figuring out how to handle these aspects.

5 Modelling objects

In Homework #10 (see the solution), we showed how objects could be modelled to take the self object as a parameter. Recursive types can be used to give valid types to the `SetCounter` class in our answer to Question 4.3 of Homework #9 (`object.f`). Do that. Leave your answer in an updated `object.f` in this homework directory. (You will need to use the `everything` checker because you need subtyping, recursive types and references.)

However, you will be unable to add types to get the `InstrSetCounter` class to type check. Look at the book’s S-AMBER rule and explain why the “obvious” typing for the `InstrSetCounter` class does not check. Is the type error necessary? That is, can you come up with an example which will break type soundness if this type error were not checked? Explain.