universität
innsbruck

Advanced Functional Programming | WS 2025/2026 | LVA 703139

Exercise Sheet 9, 10 points | Deadline: Tuesday, December 9, 2025, 4pm

- Solve the tasks in files `Exercise09*.hs` and upload only these files in OLAT.
- Mark the solved exercises in OLAT.
- Your modified `Exercise09*.hs` files must compile with `ghci` without error messages.

## Task 1 *Applicative Parsing, file `Exercise09_SMT_Answer_Parser.hs`* **3 p.**

Write a parser for the *get-value* answer of z3 *in applicative style*. You should also generalize the parser in a way that it can deal with Booleans and (positive or negative) integers.
An example answer looks as follows:

```
(
 (x ( - 5 ) )
 ( y true )
 (z 178) (u false))
```

Your parser should stop to consume input immediately after reading the final closing parenthesis.

## Task 2 *SMT Connection* **5 p.**

1. Restructure the design of the SMT connection so that all the communication with z3 is encapsulated in the SMT module. Think of a suitable interface, so that the SMT connection is easily accessible for generic encoding tasks.

   You find a template file in `Exercise_09_SMT.hs`. (3 points)

2. Adjust `lpoSolver` in file `Exercise_09_LPO_Main.hs`, so that it makes use of the new interface. (2 points)

## Task 3 *Exceptions* **2 p.**

Integrate exception handling, e.g., there might be problems that `createProcess` fails since z3 is not available, or z3 might crash or deliver unexpected answers which cannot be parsed. The implementation should work as follows:

- define a dedicated exception type for SMT related problems in file `Exercise09_SMT.hs`
- in `runSmtSolver`, parse errors of z3's output or `createProcess` exceptions should be converted into suitable SMT exceptions that contain a brief problem description (hint: use throw inside handler)
- write a wrapper around `lpoSolver` that catches SMT exceptions and returns one of three results without throwing an exception: YES(with precedence) or NO(not solvable by LPO) or MAYBE(problem description is printed to stderr)

If all tasks have been implemented, you should be able to run and see, which TRSs from the ARI database are terminating via some LPO-proof.

```
cabal repl
ghci> :l Exercise09_LPO_Main
ghci> searchLPOariFiles "ariTRSs.txt"
```