DSLsofMath 2019: Assignment 1

Patrik Jansson and Daniel Schoepe and Maximilian Algehed

1 DSLsofMath 2019: Assignment 1

In this assignment the focus is on the following three learning outcomes:

- organize areas of mathematics in DSL terms
- develop adequate notation for mathematical concepts
- discuss and compare different software implementations of mathematical concepts

1.1 DSLs, sets and von Neumann

In this assignment you will build up a domain specific language (a DSL) for finite sets. The domain you should model is pure set theory where all members are sets.

Define a datatype $SET\ v$ for the abstract syntax of set expressions with variables of type v and a datatype $PRED\ v$ for predicates over pure set expressions.

Part 1. *SET* should have constructors for

- the *Empty* set
- the one-element set constructor *Singleton*
- *Union*, and *Intersection*
  - you can also try *Powerset*
- set-valued variables ($Var :: v \to SET\ v$)

PRED should have contructors for

- the two predicates *Elem*, *Subset*
- the logical connectives *And*, *Or*, *Implies*, *Not*

Part 2. A possible semantic domain for pure sets is

$$\textbf{newtype}\ \mathit{Set} = S\ [\mathit{Set}]$$

Implement the evaluation functions

$$\begin{array}{l} \mathit{eval}\ \ :: \mathit{Eq}\ v \Rightarrow \mathit{Env}\ v\ \mathit{Set} \to \mathit{SET}\ v \quad \to \mathit{Set} \\ \mathit{check} :: \mathit{Eq}\ v \Rightarrow \mathit{Env}\ v\ \mathit{Set} \to \mathit{PRED}\ v \to \mathit{Bool} \end{array}$$

$$\textbf{type}\ \mathit{Env}\ \mathit{var}\ \mathit{dom} = [(\mathit{var}, \mathit{dom})]$$

Note that the type parameter v to SET is for the type of variables in the set expressions, not the type of elements of the sets. (You can think of pure set theory as "untyped" or "unityped".)

Part 3. The von Neumann encoding of natural numbers as sets is defined recursively as

$$\begin{array}{ll} \mathit{vonNeumann}\ 0 & = \mathit{Empty} \\ \mathit{vonNeumann}\ (n+1) & = \mathit{Union}\ (\mathit{vonNeumann}\ n) \\ & \qquad\qquad (\mathit{Singleton}\ (\mathit{vonNeumann}\ n)) \end{array}$$

Implement *vonNeumann* and explore, explain and implement the following "pseudocode" claims as functions in Haskell:

$$\begin{array}{ll} \mathit{claim1}\ \mathit{n1}\ \mathit{n2} & = \{\text{- if } (\mathit{n1} \leqslant \mathit{n2}) \text{ then } (\mathit{n1} \subseteq \mathit{n2}) \text{ -}\} \\ \mathit{claim2}\ n & = \{\text{- } n = \{0, 1, ..., n-1\} \text{ -}\} \end{array}$$

You need to insert some embeddings and types and you should use the *eval* and *check* functions. (For debugging it is useful to implement a *show* function for *Set* which uses numerals to show the von Neumann naturals.)

Admin:

- *Submission*: Assignments are to be submitted via Canvas
- *Deadline*: 2019-02-05
- *Grading*: Discussions with each of the teams during the slot 2019-02-11, 10-12 or 15-17

Note: The examination will be in English.