▸ ANTON SETZER, *A model of computation for single threaded sequential interactive programs.*
Dept. of Computer Science, Swansea University, Swansea SA1 8EN, UK.
E-mail: a.g.setzer@swansea.ac.uk.
URL Address: http://www.cs.swan.ac.uk/~csetzer/.

We present two models of computation derived from our formalisation (together with Peter Hancock) of interactive programs in dependent type theory, which define the IO monad using weakly final coalgebras.

The first model covers non-state-dependent interactive programs. An interface consists of commands $\mathrm{C} \in \mathcal{P}(\mathbb{N})$ and responses $\mathrm{R} : \mathrm{C} \to \mathcal{P}(\mathbb{N})$. Examples of commands are the printing of a string with response set a singleton set, or reading input from console with response the string being read. Instructions to actuators and reading from sensors can be represented similarly.

The set of interactive programs for an interface (C, R) is the largest set IO of pairs $\langle c, f\rangle$ with $c \in \mathrm{C}$ and $\{f\} : \mathrm{R}(c) \to \mathrm{IO}$. In order to define a monadic version $\mathrm{IO}(A)$, we add to C termination commands $\mathrm{return}(a)$ for $a \in A$ with $\mathrm{R}(\mathrm{return}(a)) = \emptyset$. One can define monadic composition $\_\gg\_ : (\mathrm{IO}(A) \times (A \to \mathrm{IO}(B))) \to \mathrm{IO}(B)$.

The second model adds a state to the interface, which determines the set of commands available, and which changes depending on commands and responses issued. So, we have states $\mathrm{S} \in \mathcal{P}(\mathbb{N})$, $\mathrm{C} \in \mathrm{S} \to \mathcal{P}(\mathbb{N})$, $\mathrm{R} \in \prod s \in \mathrm{S}.\mathrm{C}(s) \to \mathcal{P}(\mathbb{N})$ and $\mathrm{next} \in \prod s \in \mathrm{S}.\prod r \in \mathrm{C}(s).\mathrm{R}(s, r) \to \mathrm{S}$. We define $\mathrm{IO}(s)$ as the largest set of pairs $\langle c, f\rangle$ with $c \in \mathrm{C}(s)$ and $\{f\} \in \prod r \in \mathrm{R}(s, r).\mathrm{IO}(\mathrm{next}(s, c, r))$. A monadic version $\mathrm{IO}(s, A)$ for $A \in \mathrm{S} \to \mathcal{P}(\mathbb{N})$ can be defined similarly. Equality is bisimulation.

[1] Andreas Abel, Stephan Adelsberger and Anton Setzer, *Interactive programming in Agda – Objects and graphical user interfaces*, ***J Functional Programming***, vol. 27 (2017), E8.