Exercises Coalgebra for Lecture 11

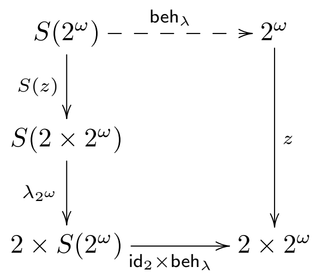
The exercises labeled with **(*)** are optional and more advanced.

1. By 2 we denote the two-elements set $2 = \{0, 1\}$. We define a (pointwise) complement operator $\mathsf{comp}\colon 2^\omega \to 2^\omega$ on binary streams as follows: $\mathsf{comp}(\sigma)(n) = 1$ iff $\sigma(n) = 0$, for all n. We're going to define this as an operation on coalgebras for the functor $B\colon \mathsf{Set} \to \mathsf{Set}$, $B(X) = 2 \times X$.

   (a) Describe comp as an inference rule, just like we did for alt (Equation (2) in the notes).

   (b) Give a functor $S\colon \mathsf{Set} \to \mathsf{Set}$ which captures the syntax (a single unary operator). Give a distributive law $\lambda\colon SB \Rightarrow BS$ which captures the complement operator. Prove that your λ is indeed a natural transformation.

   (c) **(*)** Prove that your distributive law is correct. To this end, let $z\colon 2^\omega \to 2 \times 2^\omega$ be the final B-coalgebra. Just like in the lecture, we use the distributive law λ to define the stream system on the left below:

   Show that the unique homomorphism beh_λ to the final coalgebra is indeed comp.

2. A *partial automaton* is a coalgebra of the form $\langle \epsilon, \delta \rangle\colon X \to 2 \times (X + 1)^A$, where $1 = \{*\}$. For every state and alphabet symbol, we have either a single next state, or no next state. The latter should just mean that no more words with that letter in front should be accepted. (See the notes for details.)

   (a) Show, with a concrete example, that coalgebraic bisimilarity is different from language equivalence in the above sense (where a transition to $*$ just means no arrow).

   (b) Define a determinisation procedure: for each partial automaton $\langle \epsilon, \delta \rangle$, a deterministic automaton $\langle \epsilon^\sharp, \delta^\sharp \rangle$, which makes the triangle on the left commute. How should we define η?

$$\begin{array}{ccccc}
X & \xrightarrow{\eta_X} & X+1 & \xrightarrow{\mathsf{beh}} & \mathcal{P}(A^*) \\
{\scriptstyle\langle\epsilon,\delta\rangle}\downarrow & \swarrow{\scriptstyle\langle\epsilon^\sharp,\delta^\sharp\rangle} & & & \downarrow \\
2\times(X+1)^A & & \xrightarrow[\mathsf{id}\times(\mathsf{beh})^A]{} & & 2\times\mathcal{P}(A^*)^A
\end{array}$$

The coalgebra on the right is the final deterministic automaton; beh is the unique coalgebra homomorphism to it. How does the language semantics of the partial automaton arise in this picture?

(c) In the lecture, we've seen that we could capture determinisation of non-deterministic automata by a distributive law and a monad. For partial automata, we can play a similar game: we'd like to capture $\langle\epsilon^\sharp,\delta^\sharp\rangle$ as a composition

$$S(X) \xrightarrow{S(\langle\epsilon,\delta\rangle)} SBS(X) \xrightarrow{\lambda_{SX}} BSS(X) \xrightarrow{B(\mu_X)} BS(X)$$

where $S(X) = X+1$ and $B(X) = 2\times X^A$. What should μ and λ be?

3. **(*)** In the very last part of the notes, we generalise the determinisation picture a little. In particular, it is claimed that with a functor $B\colon \mathcal{C}\to\mathcal{C}$, a monad (T,η,μ) on $\mathcal{C}$ and a distributive law $\lambda\colon TB \Rightarrow BT$, we get a functor $\overline{T}\colon \mathsf{CoAlg}(BT)\to\mathsf{CoAlg}(B)$, which lifts T.

   (a) Prove this claim.

   (b) Since this is a functor, it preserves homomorphisms. How about bisimulations? What does this say, for instance, about determinisation of non-deterministic automata?

4. **(*)** In the lecture (and the notes), we've seen that, if B has a final coalgebra, then any $\lambda\colon SB\Rightarrow BS$ defines an algebra on this final coalgebra, by finality. Investigate the case that S has an initial algebra: can we define a coalgebra on it using λ? What does this mean in a concrete example (for instance, a few operations on streams)?