Coalgebraic Semantics Lecture 1

Farzaneh Derakhshan, Tao Gu, Aditya Oak

17 June, 2019

This lecture is a first introduction to coalgebra. We first recall the inductive data types and the principle of induction. Then we introduce some basic coinductive data types and the principle of coinduction.

1 Finite and infinite data types

Throughout this note we fix an arbitrary alphabet set $\mathcal{A}$. The elements in $\mathcal{A}$ are called letters.

Example 1. Our first example is the inductive data structure $\mathcal{A}^*$ consisting of all finite words on $\mathcal{A}$. There are several ways of describing $\mathcal{A}^*$:

- In OCaml, one can write:

$$\textbf{type list a = nil | cons a l}$$

- By derivation rules:

$$\frac{}{\epsilon \in \mathcal{A}^*} \qquad \frac{a \in \mathcal{A} \qquad u \in \mathcal{A}^*}{a \cdot u \in \mathcal{A}^*}$$

Given a letter in $\mathcal{A}$ and a word in $\mathcal{A}^*$, we can *construct* a new word in $\mathcal{A}^*$ by appending the letter to the word. The above definition does not only define $\mathcal{A}^*$ using its constructors, but also provides a way to inductively define functions operating on $\mathcal{A}^*$. In the following Example **??** we use constructors of $\mathcal{A}^*$ to define the function len.

Example 2. We define function $\mathsf{len} \colon \mathcal{A} \to \mathbb{N}$ inductively using the constructors of $\mathcal{A}^*$:

$$\mathsf{len}(\epsilon) = 0, \qquad \mathsf{len}(a \cdot u) = 1 + \mathsf{len}(u).$$

Intuitively len simply counts the length of a finite word. By the inductive construction of finite words, it is enough to explain the behaviour of the function (i) on the empty word, and (ii) on $a \cdot u$ assuming its behaviour on u.

Example 3. Concatenation function ; : $\mathcal{A}^* \times \mathcal{A}^* \to \mathcal{A}^*$ is defined by structural induction on its first argument as:

$$\epsilon; u = u, \qquad (a \cdot u); v = a \cdot (u; v).$$

Similar to the previous example, we used recursive call to define concatenation.

We can also prove properties of functions defined on inductive types using their constructors:

Example 4. We want to prove $\mathsf{len}(u; v) = \mathsf{len}(u) + \mathsf{len}(v)$ given the definitions of len and ; functions in examples **??** and **??**. The proof goes by induction on the first argument:

Base case. $\mathsf{len}(\epsilon; v) = \mathsf{len}(v) = 0 + \mathsf{len}(v) = \mathsf{len}(\epsilon) + \mathsf{len}(v)$

Inductive case. $\mathsf{len}(a \cdot u; v) = \mathsf{len}(a \cdot (u; v)) = 1 + \mathsf{len}(u; v)$ by definitions of len and concatenation. By inductive hypothesis, $\mathsf{len}(u; v) = \mathsf{len}(u) + \mathsf{len}(v)$. Thus $\mathsf{len}(a \cdot u; v) = 1 + \mathsf{len}(u; v) = 1 + \mathsf{len}(u) + \mathsf{len}(v) = \mathsf{len}(a \cdot u) + \mathsf{len}(v)$.

Inductive datatypes come with an inductive principle that works on their constructors. For the coinductive infinite data types we have a similar situation. But instead of defining the values of constructors, for coinductive types we define the value of *deconstructors* in a dual manner.

2 Infinite Sequences over $\mathcal{A}$

The set $\mathcal{A}^\omega$ of all infinite sequences over $\mathcal{A}$ is defined as $\{\sigma \mid \sigma \colon \mathbb{N} \to \mathcal{A}\}$. We use the notation $\sigma = (\sigma_0, \sigma_1, \dots)$. We define deconstructing functions

$$\mathsf{head}(\sigma) \coloneqq \sigma(0)$$
$$\mathsf{tail}(\sigma)(n) \coloneqq \sigma(n+1)$$

Coalgebraic semantics: It can be considered as a way to reason about the programs that operate on infinite data structures. More details on this will be provided in the following lectures.

2.1 Functions over $\mathcal{A}^\omega$

We can define some basic operations on $\mathcal{A}^\omega$:

- $\mathsf{even} \colon \mathcal{A}^\omega \to \mathcal{A}^\omega$,

$$\mathsf{even} \colon (\sigma_0, \sigma_1, \dots) \mapsto (\sigma_0, \sigma_2, \sigma_4, \dots)$$

- $\mathsf{odd} \colon \mathcal{A}^\omega \to \mathcal{A}^\omega$

$$\mathsf{odd} \colon (\sigma_0, \sigma_1, \dots) \mapsto (\sigma_1, \sigma_3, \sigma_5, \dots)$$

- $\mathsf{merge}\colon \mathcal{A}^\omega \times \mathcal{A}^\omega \to \mathcal{A}^\omega$

$$\mathsf{merge}\colon (\sigma_0, \sigma_1, \dots), (\tau_0, \tau_1, \dots) \mapsto (\sigma_0, \tau_0, \sigma_1, \tau_1, \dots)$$

Recursive definitions of the above given functions:

- $\begin{cases} \mathsf{head}(\mathsf{merge}(\sigma, \tau)) = \mathsf{head}(\sigma) \\ \mathsf{tail}(\mathsf{merge}(\sigma, \tau)) = \mathsf{merge}(\tau, \mathsf{tail}(\sigma))) \end{cases}$
- $\begin{cases} \mathsf{head}(\mathsf{even}(\sigma)) = \mathsf{head}(\sigma) \\ \mathsf{tail}(\mathsf{even}(\sigma)) = \mathsf{even}(\mathsf{tail}(\mathsf{tail}(\sigma))) = \mathsf{odd}(\mathsf{tail}(\sigma)) \end{cases}$
- $\begin{cases} \mathsf{head}(\mathsf{odd}(\sigma)) = \mathsf{head}(\mathsf{tail}(\sigma)) \\ \mathsf{tail}(\mathsf{odd}(\sigma)) = \mathsf{tail}(\mathsf{even}(\mathsf{tail}(\sigma))) \end{cases}$

Proposition 5. $\mathsf{merge}(\mathsf{even}(\sigma), \mathsf{odd}(\sigma)) = \sigma$

Proof. Let $\rho = \mathsf{merge}(\mathsf{even}(\sigma), \mathsf{odd}(\sigma))$. Now we show that $\forall n \in \mathbb{N}, \rho(n) = \sigma(n)$:

Base case.

$$\begin{aligned} \rho(0) &= \mathsf{head}(\mathsf{merge}(\mathsf{even}(\sigma), \mathsf{odd}(\sigma))) \\ &= \mathsf{head}(\mathsf{even}(\sigma)) \\ &= \mathsf{head}(\sigma) \\ &= \sigma(0) \end{aligned}$$

Inductive case.

$$\begin{aligned} \rho(n+1) &= \mathsf{merge}(\mathsf{odd}(\sigma), \mathsf{tail}(\mathsf{even}(\sigma)))(n) \\ &= \mathsf{merge}(\mathsf{odd}(\sigma), \mathsf{odd}(\mathsf{tail}(\sigma)))(n) \\ &= \mathsf{merge}(\mathsf{even}(\mathsf{tail}(\sigma)), \mathsf{odd}(\mathsf{tail}(\sigma)))(n) \qquad \text{(IH)} \\ &= \mathsf{tail}(\sigma)(n) \\ &= \sigma(n+1) \end{aligned}$$

□

Proposition 6. $\mathsf{even}(\mathsf{merge}(\sigma, \tau)) = \sigma$

Proof. Let $\rho = \mathsf{even}(\mathsf{merge}(\sigma, \tau))$.

Base case.

$$\begin{aligned} \rho(0) &= \mathsf{merge}(\sigma, \tau)(0) \\ &= \sigma(0) \end{aligned}$$

Inductive case.

$$\begin{aligned}\rho(n+1) &= \mathsf{merge}(\sigma,\tau)(2(n+1)) \\ &= \sigma(n+1)\end{aligned}$$

□

Similarly we can show $\mathsf{odd}(\mathsf{merge}(\sigma,\tau)) = \tau$.

3 Bisimulation and Coinduction

Proof by induction is common and useful on inductive data structures. Similarly, one would expect coinductive proof on coinductive data structures. One convenient way for this is to find a bisimulation, and then apply the coinductive principle. To get the feeling of coinduction, we will focus on infinite streams $\mathcal{A}^*$ here.

Definition 7. A relation $R \subseteq \mathcal{A}^* \times \mathcal{A}^*$ is a *bisimulation* if $\forall(\sigma,\tau) \in R$,

1. $\mathsf{head}(\sigma) = \mathsf{head}(\tau)$
2. $(\mathsf{tail}(\sigma), \mathsf{tail}(\sigma)) \in R$

We say that two infinite sequences σ and τ are *bisimilar*, denoted as $\sigma \sim \tau$, if there exists some bisimulation between them. Bisimulation provides a way of proving by coinduction:

Lemma 8 (Coinductive Principle for $\mathcal{A}^*$)**.** *For all $\sigma, \tau \in \mathcal{A}^*$,*

$$\sigma \sim \tau \Rightarrow \sigma = \tau$$

Proof. Suppose $R \subseteq \mathcal{A}^* \times \mathcal{A}^*$ is a bisimulation such that $(\sigma,\tau) \in R$. It suffices to show that $\sigma(n) = \tau(n)$, for all $n \in \mathbb{N}$. We prove by induction on n.

- For $n = 0$, this is exactly (1) in definition **??**.
- For $n+1$, note that by (2) in definition **??** we have $\mathsf{tail}(\sigma) \sim \mathsf{tail}(\tau)$. So we have:

$$\begin{aligned}\sigma(n+1) &= \mathsf{tail}(\sigma)(n) \\ &= \mathsf{tail}(\tau)(n) && \text{(IH)} \\ &= \tau(n+1)\end{aligned}$$

Therefore $\sigma = \tau$. □

In some cases, we may not have a bisimulation at first sight. It is common that we start from the conclusion we want to proof, and construct a bisimulation by adding the necessary pairs.

Example 9. Consider the equation

$$\mathsf{merge}(\mathsf{even}(\sigma), \mathsf{odd}(\sigma)) = \sigma \tag{1}$$

To derive (**??**) using the coinduction principle, we would like to construct some bisimulation R which contains the pair consisting of the streams on both sides of (**??**). Consider the relation

$$R_0 = \{(\mathsf{merge}(\mathsf{even}(\sigma), \mathsf{odd}(\sigma)), \sigma) \mid \sigma \in \mathcal{A}^\omega\}$$

To ensure condition (2) in definition **??**, we need to add the following pairs to the relation:

$$R_1 = \{(\mathsf{tail}(\mathsf{merge}(\mathsf{even}(\sigma), \mathsf{odd}(\sigma))), \mathsf{tail}(\sigma)) \mid \sigma \in \mathcal{A}^\omega\}$$

Proceeding like this, we get a sequence of relations, and the union of all these relations should give a bisimulation. Fortunately we do not need to add a lot, since R_1 is already included in R_0:

$$\begin{aligned}\mathsf{tail}(\mathsf{merge}(\mathsf{even}(\sigma), \mathsf{odd}(\sigma))) &= \mathsf{merge}(\mathsf{odd}(\sigma), \mathsf{tail}(\mathsf{even}(\sigma))) \\ &= \mathsf{merge}(\mathsf{even}(\mathsf{tail}(\sigma)), \mathsf{odd}(\mathsf{tail}(\sigma)))\end{aligned}$$

For any σ, consider $\tau = \mathsf{tail}(\sigma)$, and we know that $(\mathsf{merge}(\mathsf{even}(\tau), \mathsf{odd}(\tau)), \tau) \in R_0$.

Example 10. Consider the equation

$$\mathsf{even}(\mathsf{merge}(\sigma, \tau)) = \sigma \tag{2}$$

Following the above strategy, we construct a sequence of relations:

$$\begin{aligned}R_0 &= \{(\mathsf{even}(\mathsf{merge}(\sigma, \tau)), \sigma) \mid \sigma, \tau \in \mathcal{A}^\omega\} \\ R_1 &= \{(\mathsf{tail}(\mathsf{even}(\mathsf{merge}(\sigma, \tau))), \mathsf{tail}(\sigma)) \mid \sigma, \tau \in \mathcal{A}^\omega\} \\ R_2 &= \{(\mathsf{tail}(\mathsf{tail}(\mathsf{even}(\mathsf{merge}(\sigma, \tau)))), \mathsf{tail}(\mathsf{tail}(\sigma))) \mid \sigma, \tau \in \mathcal{A}^\omega\} \\ &\dots\end{aligned}$$

Note that

$$\begin{aligned}\mathsf{tail}(\mathsf{even}(\mathsf{merge}(\sigma, \tau))) &= \mathsf{odd}(\mathsf{tail}(\mathsf{merge}(\sigma, \tau))) \\ &= \mathsf{odd}(\mathsf{merge}(\tau, \mathsf{tail}(\sigma))) \\ \mathsf{tail}(\mathsf{odd}(\mathsf{merge}(\sigma, \tau))) &= \mathsf{even}(\mathsf{tail}(\mathsf{merge}(\sigma, \tau))) \\ &= \mathsf{even}(\mathsf{merge}(\tau, \mathsf{tail}(\sigma)))\end{aligned}$$

Then it is easy to see that $R_0 \supseteq R_2 \supseteq \cdots$ and $R_1 \supseteq R_3 \supseteq \cdots$. So the union of the whole sequence terminates as $R \coloneqq R_0 \cup R_1$. R is then a bisimulation containing all $(\mathsf{even}(\mathsf{merge}(\sigma, \tau)), \sigma)$.

Homework

Suppose $R, S \subseteq \mathcal{A}^\omega \times \mathcal{A}^\omega$ are bisimulations.

1. $R;S$ is bisimulation, where $(x,z) \in R;S$ iff $\exists y$ such that $(x,y) \in R$ and $(y,z) \in S$.
2. R^o is bisimulation, where $(x,y) \in R^o$ iff $(y,x) \in R$.