Lightly doctored!

Introduction to Kleene Algebra	~~Undoctored~~ Class Notes March 7,2024
CS 6861 Spring 2024	instructor: Dexter Kozen, scribe: Mark Moeller

Remark 1. *Before we get started:*

1. *Deciding equations in Kleene Algebra (KA) is PSPACE-complete (as you would learn in CS4810 and CS6810).*
2. *The examples we have seen so far use traditional finite automata, but the techniques are much more general. See the PhD Thesis of Alexandra Silva.*
3. *Looking ahead, we will later deal with KA with more bells and whistles, for example:*
   - *KAT*
   - *NetKAT*
   - *PL constructs such as while loops, if/then/else*
   - *Hoare Logic*
   - *GKAT*
   - *Decision procedures and completeness results for all of the above, by reduction to completeness of KA.*

1 Brzozowski Derivatives

Last time we saw that automata are coalgebras for the functor:

$$GX = 2 \times X^{\Sigma},$$

and such a coalgebra is given by a pair of functions:

$$\begin{gathered}(\epsilon, \delta)\colon X \to 2 \times X^{\Sigma^*}\\ \epsilon\colon X \to 2\\ \delta_a\colon X \to X.\end{gathered}$$

In particular, the final coalgebra is $(2^{\Sigma^*}, E, D)$, where:

$$\begin{gathered}E\colon 2^{\Sigma^*} \to 2 \qquad E(A) = \begin{cases}1 & \epsilon \in A\\ 0 & \epsilon \notin A\end{cases}\\ D_a\colon 2^{\Sigma^*} \to 2^{\Sigma^*} \qquad D_a(A) = \{x \mid ax \in A\}.\end{gathered}$$

The structure map (E, D) of the final coalgebra is also called the *Semantic Brzozowski Derivatve.*

It turns out $(\mathsf{Reg}\Sigma, E, D)$ is a subalgebra and a subcoalgebra of $(2^{\Sigma^*}, E, D)$, which means that there are both algebra and coalgebra homomorphisms from $(\mathsf{Reg}\Sigma, E, D)$ to $(2^{\Sigma^*}, E, D)$. This means they are both *bialgebras*, but we will defer any more commentary on this concept for now.

Now we will show that $\mathsf{Exp}\Sigma/\equiv$ has coalgebraic structure as well (Actually, we will ignore the congruence classes $/\equiv$, because they do not affect the construction).

The coalgebraic structure is $(\mathsf{Exp}\Sigma, e, d)$, where:

$$\begin{aligned} e &\colon \mathsf{Exp}\Sigma \to 2 \\ d_a &\colon \mathsf{Exp}\Sigma \to \mathsf{Exp}\Sigma \end{aligned}$$

Note that we need the definitions to be such that for each $s \in \mathsf{Exp}\Sigma$, we will have $e(s) = E(\mathsf{R}_\Sigma(s))$. We define e inductively:

$$\begin{aligned} e(a) &= 0, a \in \Sigma \\ e(0) &= 0 \\ e(1) &= 1 \\ e(s+t) &= e(s) + e(t) \\ e(s \cdot t) &= e(s) \cdot e(t) \\ e(s^*) &= 1 \end{aligned}$$

Remark 2. *We have seen this definition before! This is the empty word property of Salomaa.*

Now we need to give the definition of d so that the following diagram commutes:

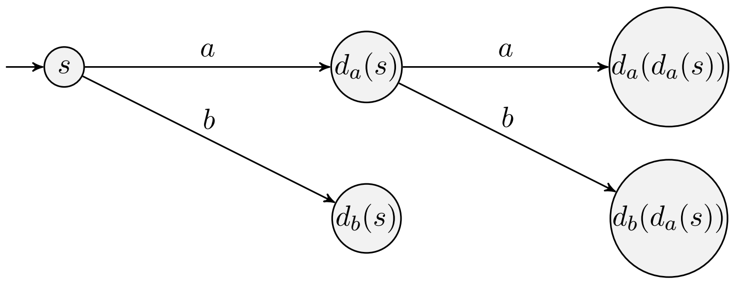
$$\begin{tikzcd} s \arrow[r, "\mathsf{R}_\Sigma"] \arrow[d, "d_a"'] & \mathsf{R}_\Sigma(s) \arrow[d, "D_a"] \\ d_a(s) \arrow[r, "\mathsf{R}_\Sigma"] & D_a(\mathsf{R}_\Sigma(s)) \end{tikzcd}$$

In particular, it needs to be the case that $D_a(\mathsf{R}_\Sigma(s)) \triangleq \{x \mid ax \in \mathsf{R}_\Sigma(s)\} = \mathsf{R}_\Sigma(d_a(s))$. Here is the definition that makes this work:

$$\begin{aligned} d_a(b) &= 0, a \neq b \\ d_a(a) &= 1 \\ d_a(0) &= 0 \\ d_a(1) &= 0 \\ d_a(s+t) &= d_a(s) + d_a(t) \\ d_a(s \cdot t) &= d_a(s) \cdot t + e(s) \cdot d_a(t) \\ d_a(s^*) &= d_a(s) \cdot s^* \end{aligned}$$

These (e, d) are called the *Syntactic Brzozowski Derivative.*

Brzozowski used the syntactic derivative as the basis for an algorithm for converting regular expressions to finite automata. Starting from an expression s, we repeatedly "explore outwards" by taking derivatives:


$$\begin{tikzcd}
\arrow[r] & s \arrow[r, "a"] \arrow[dr, "b"] & d_a(s) \arrow[r, "a"] \arrow[dr, "b"] & d_a(d_a(s)) \\
& & d_b(s) & d_b(d_a(s))
\end{tikzcd}$$

…and so on…

Remark 3. *Brzozowski proved that this process terminates as long as we normalize the expressions during the process by associativity, commutativity, and idempotence (ACI). Without doing this, the process of taking derivatives repeatedly might not terminate.*

2 Bisimulations

We now develop a notion of *bisimulation*, a relation on coalgebras with some additional structure. We will also see that *bisimulation equivalences* form the kernels of coalgebra homomorphisms in the same way that congruences are the kernels of algebra homomorphisms.

Let $(X, \mathsf{obs}, \mathsf{cont}), (Y, \mathsf{obs}, \mathsf{cont})$ be coalgebras for the set functor $\Sigma \times -$. Thus for $x \in X$ and $y \in Y$, $\mathsf{obs}(x) \in \Sigma$, $\mathsf{obs}(y) \in \Sigma$, $\mathsf{cont}(x) \in X$, and $\mathsf{cont}(y) \in Y$.

Definition 1 (Bisimulation for streams). *A relation $\approx \subseteq X \times Y$ is a* bisimulation *if for any $x \in X, y \in Y$ such that $x \approx y$ we have:*

1. $\mathsf{obs}(x) = \mathsf{obs}(y)$*, and*
2. $\mathsf{cont}(x) \approx \mathsf{cont}(y)$*.*

Note that under this definition, $\emptyset$ is a bisimulation.

Proposition 1. *Let A be any set of bisimulations between X and Y. Then $\bigcup A = \{(s,t) \mid \exists {\approx} \in A \colon s \approx t\}$ is a bisimulation between X and Y.*

Proof.

$$\begin{aligned} (s,t) \in \bigcup A &\Rightarrow (s,t) \in {\approx} \text{ for some } {\approx} \in A \\ &\Rightarrow (s,t) \text{ statisfies (1) and (2) above.} \end{aligned}$$

Since $(s,t) \in \bigcup A$ was arbitrary, $\bigcup A$ statisfies (1) and (2) above. □

Proposition 2. *There is always a unique maximal bisimulation between X and Y, which we denote by ($\approxeq$).*

By the Proposition 1, the union of *all* bisimulations between X and Y is a bisimulation between X and Y, and it is the maximal bisimulation because it contains all other bisimulations.

Definition 2 (Bisimilar). *We call $s \in X$ and $t \in Y$* bisimilar *if they are related by the masximal bisimulation, i.e., $s \approxeq t$.*

Definition 3 (Autobisimulation)**.** *A bisimulation* $\approx \subseteq X \times X$ *is called an* autobisimulation *on* X*.*

The way we have defined it here, a bisimulation is not necessarily an equivalence relation. If we close an autobisimulation under reflexivity, symmetry, and transitivity, the resulting relation is still a bisimulation.

Whenever a bisimulation is an equivalence relation, it is also called a *bisimulation equivlance.*

Proposition 3. $s \approx t \Rightarrow s \approxeq t$*.*

We leave the remaining facts as exercises:

1. The kernel of a coalgebra homomorphism is a bisimulation equivalence, where:
$$\mathsf{ker}(h) = \{(s,t) \mid h(s) = h(t)\}$$
2. Any bisimulation equivalence is $\mathsf{ker}(h)$ for some homomorphism h (the proof is via the quotient construction).
3. The maximal autobisimulation ($\approxeq$) is the kernel of the unique homomorphism to the final coalgebra:

   That is, for the unique homormorphism:
$$(X, \mathsf{obs}, \mathsf{cont}) \to (\Sigma^\omega, \mathsf{hd}, \mathsf{tl})$$
   We have:
$$\begin{aligned} x \approxeq y \Rightarrow \forall n\ \mathsf{obs}(\mathsf{cont}^n(x)) &= \mathsf{obs}(\mathsf{cont}^n(y)) \\ &\Rightarrow (\mathsf{obs}(x), \mathsf{obs}(\mathsf{cont}(x)), \mathsf{obs}(\mathsf{cont}(\mathsf{cont}(x))), \ldots) \\ &\quad = (\mathsf{obs}(y), \mathsf{obs}(\mathsf{cont}(y)), \mathsf{obs}(\mathsf{cont}(\mathsf{cont}(y))), \ldots) \end{aligned}$$

What does all this mean for automata?

Recall that automata are a coalgebra using the functor
$$GX = 2 \times X^\Sigma$$

Definition 4 (Bisimulation for automata)**.** *A relation* $\approx \subseteq (X, \epsilon, \delta) \times (Y, \epsilon, \delta)$ *is a* bisimulation *if for any* $x \in X, y \in Y$ *such that* $x \approx y$ *we have:*

1. $\epsilon(x) = \epsilon(y)$*, and*

2. $\delta_a(x) \approx \delta_a(y)$*, for each* $a \in \Sigma$*.*

This definition is equivalent to the definition we gave for the Myhill-Nerode relation on states of automata in a previous lecture. That is, for all strings $x \in \Sigma^*$, we have:
$$\hat{\delta}(s,x) \in F \iff \hat{\delta}(t,x) \in F$$
(old notation) for automaton states (s,t) exactly whenever we have:
$$\epsilon(\hat{\delta}_x(s)) = \epsilon(\hat{\delta}_x(t)),$$
where $\hat{\delta}_\epsilon = \mathsf{id}$, and $\hat{\delta}_{xa} = \delta_a \circ \hat{\delta}_x$.

3 Coinduction

Coinduction is a technique we can use for both definitions and proofs in coalgebra. We can think of coinduction as “induction without a basis.” Whereas in inductive reasoning we are “guilty until proven innocent,” (structures do not exist unless derived from base cases), in coinductive reasoning we take the view that we are “innocent until proven guilty” (structures are valid unless we find a contradiction).

We will conclude the discussion with an example. Suppose we have a stream $\sigma = (x_0, x_1, \ldots)$. We want to write a function that splits streams into even- and odd-indexed elements, i.e.

$$\mathsf{split}(\sigma) = ((x_0, x_2, x_4, \ldots), (x_1, x_3, x_5, \ldots)).$$

So our function will have type:

$$\mathsf{split}\colon \Sigma^\omega \to \Sigma^\omega \times \Sigma^\omega$$

And we can define it *corecursively* (or coinductively) as follows:

$$\mathsf{split}(x_0 :: x_1 :: \tau) \triangleq (x_0 :: \pi_1(\mathsf{split}(\tau)), x_1 :: \pi_2(\mathsf{split}(\tau)))$$

This function has no basis! But it is a perfectly good *coinductive* definition.

In the other direction we want a function with type:

$$\mathsf{merge}\colon \Sigma^\omega \times \Sigma^\omega \to \Sigma^\omega$$

We can define this function (again, coinductively) as follows:

$$\mathsf{merge}(x :: \sigma, y :: \tau) \triangleq x :: y :: \mathsf{merge}(\sigma, \tau)$$

Proposition 4. *For any stream σ, $\mathsf{merge}(\mathsf{split}(\sigma)) = \sigma$.*

Proof.

$$\begin{aligned}
\mathsf{merge}(\mathsf{split}(x_0 :: x_1 :: \sigma)) &= \mathsf{merge}(x_0 :: \pi_1(\mathsf{split}(\sigma)), x_1 :: \pi_2(\mathsf{split}(\sigma))) \\
&= x_0 :: x_1 :: \mathsf{merge}(\pi_1(\mathsf{split}(\sigma)), \pi_2(\mathsf{split}(\sigma))) \\
&= x_0 :: x_1 :: \mathsf{merge}(\mathsf{split}(\sigma)) \\
&= x_0 :: x_1 :: \sigma
\end{aligned}$$

The last step uses the *coinductive hypothesis* $\mathsf{merge}(\mathsf{split}(\sigma)) = \sigma$, which is the assumption that the tails for the streams satisfy the property we are trying to prove. We are allowed to assume the property holds of the tails in the proof that $\mathsf{merge}(\mathsf{split}(x_0 :: x_1 :: \sigma)) = x_0 :: x_1 :: \sigma$. □