MFoCS Seminar: Regular Transductions

Daan Spijkers

Jan 2025

Introduction

- We will be dealing with a particular kind of *string-to-string* functions, called regular transductions.
- These can be considered a generalization of regular languages.
- Instead of accepting or rejecting a string $w \in \Sigma^*$, for each string we produce an output string.
- This output can be in Σ^* or in some other Γ^*.
- Regular languages also have many different formalisms (regular expressions, automatons, etc)

Regular Transductions

- There are many ways to define functions over strings, say for example homomorphisms, for which:

$$f(w \cdot v) = f(w) \cdot f(v)$$

- This is very limiting.
- In fact, a homomorphism h is uniquely defined by its application over the letters of Σ.

Regular Transductions

- Homomorphisms can only change the letters in each word.
- Regular transductions can do significantly more
- Examples include for, say $w \in \{a, b, c\}^*$,

$$w \mapsto rev(w)$$

$$w \mapsto ww$$

$$w \mapsto (\text{longest c-free prefix in } w) \cdot (\text{longest c-free suffix in } w)$$

- We will be looking at several models.

Papers

- Mikołaj Bojańczyk and Lê Thành Dũng Nguyễn. Algebraic Recognition of Regular Functions, 2023.

Papers

- Mikołaj Bojańczyk and Lê Thành Dũng Nguyễn. Algebraic Recognition of Regular Functions, 2023.
- Rajeev Alur and Pavol Černý. Expressiveness of streaming string transducers, 2010.

SST

- A *streaming string transducer* (SST) is a model which uses several variables (or registers) to define a function.
- It processes the input string from left-to-right (streaming).
- Each letter updates the registers.

SST

- Say $\Sigma = \{a, b\}$, the input is *aaab*, registers $R = \{x\}$.
- What does $l \mapsto x = xl$ do?

$aaab$	$x = \epsilon$
aab	$x = a$
ab	$x = aa$
b	$x = aaa$
	$x = aaab$

SST

- Instead of $l \mapsto x = xl$, what about $l \mapsto x = lx$?

SST

- Instead of $l \mapsto x = xl$, what about $l \mapsto x = lx$?

$aaab$	$x = \epsilon$
aab	$x = a$
ab	$x = aa$
b	$x = aaa$
	$x = baaa$

SST

- A finite set of states Q with initial state $q_0 \in Q$.
- A finite set of variables X.
- State and variable transition functions:

$$\delta_1 : Q \times \Sigma \to Q$$
$$\delta_2 : Q \times X \times \Sigma \to (X + \Gamma)^*$$

- A partial output function $F : Q \to (X + \Gamma)^*$.

SST

- A finite set of states Q with initial state $q_0 \in Q$.
- A finite set of variables X.
- State and variable transition functions:

$$\delta_1 : Q \times \Sigma \to Q$$
$$\delta_2 : Q \times X \times \Sigma \to (X + \Gamma)^*$$

- A partial output function $F : Q \to (X + \Gamma)^*$.
- F and δ_2 must be copyless.

SST

- For the semantics we use some (q, s) where

$$s : X \to \Gamma^*$$

  We can easily extend this to $s : (X + \Gamma)^* \to \Gamma^*$.
- For the initial configuration this sends all registers to ϵ.

$$(q_0, s_0)$$

SST

- For the semantics we use some (q, s) where

$$s : X \to \Gamma^*$$

  We can easily extend this to $s : (X + \Gamma)^* \to \Gamma^*$.
- For the initial configuration this sends all registers to ϵ.

$$(q_0, s_0)$$

- The transition function is defined by

$$\begin{aligned} \delta(q, s)(a) &= (\delta_1(q, a), s') \\ s'(x) &= s(\delta_2(q, a, x)) \end{aligned}$$

SST

- For the semantics we use some (q, s) where

$$s : X \to \Gamma^*$$

  We can easily extend this to $s : (X + \Gamma)^* \to \Gamma^*$.
- For the initial configuration this sends all registers to ϵ.

$$(q_0, s_0)$$

- The transition function is defined by

$$\begin{aligned} \delta(q, s)(a) &= (\delta_1(q, a), s') \\ s'(x) &= s(\delta_2(q, a, x)) \end{aligned}$$

- Now for some word w we apply this transition function and take the output:

$$(q, s) = \delta^*((q_0, s_0), w)$$

$$s(F(q))$$

SST Examples

- An example we use throughout is

$$f(w) = w \cdot rev(w)$$

- $Q = \{q_0\}$ and $X = \{x, y\}$.
- The update functions are

$$\begin{aligned} \delta_1(q_0, l) &= q_0 \\ \delta_2(q_0, x, l) &= xl \\ \delta_2(q_0, y, l) &= ly \end{aligned}$$

- Output function is $q_0 \mapsto xy$.

Monadic Second Order Logic

- As input in this model we transform a word into an edge-labeled graph
- The output graph will then be defined using MSO formulas over this input graph.

MSO

- Monadic Second order logic is an extension of first order logic where we can quantify over unary (monadic) relations.

$$\forall P \forall x (Px \vee \neg Px)$$

- For our purposes this means we can quantify over both nodes and sets of nodes in the graph.

MSO

- Monadic Second order logic is an extension of first order logic where we can quantify over unary (monadic) relations.

$$\forall P \forall x (Px \vee \neg Px)$$

- For our purposes this means we can quantify over both nodes and sets of nodes in the graph.
- In MSO we have:
  - Atomic formulas $x \in X$, $x = y$, $a(x, y)$
  - Boolean connectives such as $\vee, \wedge, \neg, \rightarrow$.
  - Quantifiers $\exists, \forall$ for nodes x, y and sets of nodes X, Y.

Example MSO Formulas

Examples of MSO formulas for a graph with edge labels Σ:

- There is an edge between x and y with label a

$$q(x, y) = a(x, y)$$

Example MSO Formulas

Examples of MSO formulas for a graph with edge labels Σ:

- There is an edge between x and y with label a

$$q(x, y) = a(x, y)$$

- There exists an outgoing edge with label a for this vertex:

$$q(x) = \exists y . a(x, y)$$

Example MSO Formulas

Examples of MSO formulas for a graph with edge labels Σ:

- There is an edge between x and y with label a

$$q(x, y) = a(x, y)$$

- There exists an outgoing edge with label a for this vertex:

$$q(x) = \exists y . a(x, y)$$

- There exists an outgoing edge with any label for this vertex:

$$q(x) = \exists y . \bigvee_{a \in \Sigma} a(x, y)$$

MSO Model

Definition

For some finite copy set C, we define vertex and edge formulas. That is,

- v^c is in the output graph if $\phi^c(v)$ is true
- There is an edge $v^c \overset{a}{\mapsto} u^d$ if $\phi_a^{c\ d}(v, u)$ is true.

MSO Examples

Say we want to define $f(w) = rev(w)$. We only need a copy set with one element to do this $C = \{1\}$.

- We want to keep all the vertices, so $\phi^1(x) = \top$.
- All edges should be reversed, so $\phi_a^{1\ 1}(x, y) = a(y, x)$ for all $a \in \Sigma$.

MSO Examples

In the case of $f(w) = w \cdot rev(w)$ we will need more vertices, so $C = \{1, 2\}$.

- In the first copy set we keep all vertices except the last one, in the second one we keep everything. That is,

$$\phi^1(x) = \neg \left(\bigvee_{a \in \Sigma} \exists y. a(x, y) \right)$$
$$\phi^2(x) = \top$$

For convenience we will re-use ϕ^1 as *out*. We can also define *in* accordingly.

MSO Examples

For the edge formulas we do something similar.

MSO Examples

For the edge formulas we do something similar.

- In $(1,1)$ we keep all edges except the last one, which has no outgoing edge:

$$\phi_a^{1\ 1}(x,y) = a(x,y) \wedge out(y)$$

MSO Examples

For the edge formulas we do something similar.

- In $(1, 1)$ we keep all edges except the last one, which has no outgoing edge:

$$\phi_a^{1\ 1}(x, y) = a(x, y) \wedge out(y)$$

- In $(2, 2)$ we reverse all the edges:

$$\phi_a^{2\ 2}(x, y) = a(y, x)$$

MSO Examples

For the edge formulas we do something similar.

- In $(1, 1)$ we keep all edges except the last one, which has no outgoing edge:

$$\phi_a^{1\ 1}(x, y) = a(x, y) \wedge out(y)$$

- In $(2, 2)$ we reverse all the edges:

$$\phi_a^{2\ 2}(x, y) = a(y, x)$$

- And in $(1, 2)$ we connect the last vertex we kept from set 1 to the last vertex from set 2. This is the edge we discarded.

$$\phi_a^{1,2}(x, y) = a(x, y) \wedge \neg out(y)$$

MSO Examples

For the edge formulas we do something similar.

- In $(1,1)$ we keep all edges except the last one, which has no outgoing edge:

$$\phi_a^{1\ 1}(x,y) = a(x,y) \wedge out(y)$$

- In $(2,2)$ we reverse all the edges:

$$\phi_a^{2\ 2}(x,y) = a(y,x)$$

- And in $(1,2)$ we connect the last vertex we kept from set 1 to the last vertex from set 2. This is the edge we discarded.

$$\phi_a^{1,2}(x,y) = a(x,y) \wedge \neg out(y)$$

- $\phi_a^{2,1} = \bot$.

MSO Equivalence

- We want to create a MSO model from an SST model.
- We do this by representing the update function δ_2 in several nodes.
- This is best shown by example.

MSO Equivalence

- Let us take the model $f(w) = w \cdot rev(w)$, with input aba. For some letter $l \in Sigma$ this has the following register updates:

$$X = Xl$$
$$Y = lY$$

- We will represent each right-hand side of a register update by several nodes.
- Letters will be represented by two nodes without edges, while letters will have an edge between them.
- We then need to connect these nodes, following the path that each variable took.

Transducer Semigroup

- First, recall the definition of a semigroup:
  - Some set M of elements.
  - An associative operation $\cdot : M \times M \to M$:

$$a \cdot (b \cdot c) = (a \cdot b) \cdot c$$

- To map between two semigroups we use a homomorphism:

$$f : M \to N$$
$$f(a \cdot_M b) = f(a) \cdot_N f(b)$$

Functor

- A homomorphism f is between two specific semigroups.
- For our purposes we need a new construction which works for every semigroup.
- We will use a semigroup functor.
- This is a mapping from semigroups to semigroups, and homomorphisms to homomorphisms.
- Say f is a homomorphism, and F a semigroup functor:

$$f : X \to Y$$
$$F(f) : FX \to FY$$

Functor

Examples of semigroup functors include:

- Mapping a semigroup M to a tuple of $M \times M$, with:

$$(l_1, r_1) \cdot_{M \times M} (l_2, r_2) = (l_1 \cdot_M l_2, r_1 \cdot_M r_2)$$

- Mapping M to lists of elements M^*.
- Mapping a semigroup to its opposite semigroup, which we will denote M^{-1}.

$$a \cdot_{M^{-1}} b = b \cdot_M a$$

Transducer Semigroup

Definition

A transducer semigroup onsists of the following:

- A semigroup-to-semigroup functor F
- An output mechanism $out_A : FA \to A$. This is a collection of functions such that the following diagram commutes for any homomorphism h:

$$\begin{array}{ccc} FA & \xrightarrow{Fh} & FB \\ \downarrow{\scriptstyle out_A} & & \downarrow{\scriptstyle out_B} \\ A & \xrightarrow{h} & B \end{array}$$

Transducer Semigroup

Definition

A transducer semigroup onsists of the following:

- A semigroup-to-semigroup functor F
- An output mechanism $out_A : FA \to A$. This is a collection of functions such that the following diagram commutes for any homomorphism h:

$$\begin{array}{ccc} FA & \xrightarrow{Fh} & FB \\ \downarrow{\scriptstyle out_A} & & \downarrow{\scriptstyle out_B} \\ A & \xrightarrow{h} & B \end{array}$$

  This is a natural transformation.

Transducer Semigroup

Definition

A function $f : A \to B$ between semigroups is recognized by a transducer semigroup (F, out) if it can be decomposed as

$$A \xrightarrow{h} FB \xrightarrow{out_B} B$$

For some semigroup homomorphism h.

Transducer Semigroup

Definition

A function $f : A \to B$ between semigroups is recognized by a transducer semigroup (F, out) if it can be decomposed as

$$A \xrightarrow{h} FB \xrightarrow{out_B} B$$

For some semigroup homomorphism h.

We will exclusively deal with the case where:

- The function is string-to-string
- The functor is finiteness-preserving

Transducer Semigroup Examples

To define $f(w) = w \cdot rev(w)$ let us first define

$$double(w) = ww$$
$$rev(w)$$

in terms of transducer semigroups.
We will combine these into the proper transducer semigroup.

Transducer Semigroup Example 1

For *double*(w) we define it as follows:

- The tuple functor,

$$FX = X \times X$$
$$F(f) : X \times X \to Y \times Y$$
$$F(f)(x, x) = (f(x), f(x))$$

With operation mentioned previously:

$$(l_1, r_1) \cdot_{M \times M} (l_2, r_2) = (l_1 \cdot_M l_2, r_1 \cdot_M r_2)$$

- Homomorphism $h : \Sigma^* \to \Sigma^* \times \Sigma^*$ as $h(w) = (w, w)$.
- Output function $out_X(w, w) = w \cdot w$ which concatenates the words.

Transducer Semigroup Double

$$aaabb \overset{h}{\mapsto} (aaabb, aaabb) \xmapsto{out_{\Sigma^*}} aaabbaaabb$$

Transducer Semigroup Example 1

To show this is a valid transducer semigroup we need to show the following things:

- F is a valid semigroup functor.
- $h(w) = (w, w)$ is a homomorphism from Σ^* to $F\Sigma^*$:

$$h(w \cdot r) = (wr, wr) = (w, w) \cdot (r, r) = h(w) \cdot h(r)$$

- The output mechanism is natural.

Transducer Semigroup Example 1

Now we define $rev(w)$ as follows:

- Functor F maps a semigroup Σ^* to its opposite semigroup, $(\Sigma^*)^{-1}$.

Transducer Semigroup Example 1

Now we define $rev(w)$ as follows:

- Functor F maps a semigroup Σ^* to its opposite semigroup, $(\Sigma^*)^{-1}$.
- Homomorphism h:

$$
\begin{aligned}
h(l_1 \cdot l_2 \cdot \cdots \cdot l_n) &= l_1 \cdot_{-1} l_2 \cdot_{-1} \cdots -1 \cdot l_n \\
&= l_n \cdots l_2 l_1
\end{aligned}
$$

  This is actually just $rev(w)$!
- Output mechanism $out_{\Sigma^*}(w) = w$.

Transducer Semigroup Double

$$
\begin{aligned}
aaabb &\overset{h}{\mapsto} a \cdot_{-1} a \cdot_{-1} a \cdot_{-1} b \cdot_{-1} b \\
&= bbaaa \\
&\xmapsto{out_{\Sigma^*}} bbaaa
\end{aligned}
$$

Transducer Semigroup Example 1

We can combine these into the function $f(w) = w \cdot rev(w)$:

- Functor $F(M) = M \times M^{-1}$.
- Homomorphism $f(w) = (w, rev(w))$
- Output mechanism $out_M((w, r)) = w \cdot r$.

$$M \xrightarrow{double} M \times M \xrightarrow{id \times rev} M \times M^{-1} \xrightarrow{out_M} M$$

Transducer Semigroup Equivalence

- Say we have a transducer semigroup (F, out, h).
- Proving the exact equivalence is quite difficult.
- Our goal:

$$update : \Sigma^* \to U^*$$
$$\delta : R \times U \to R$$

- We will define what these updates U and registers R are in the process.

Transducer Semigroup Equivalence

Our plan:

1. Track what happens to each input letter in the output (origin information).
2. Turn this output with origin information into a list of updates.
3. Define what the register and transition functions look like.

Semigroup Coproduct

Definition

A coproduct of two semigroups A, B, denoted $A \oplus B$ is the following semigroup:

- Elements are disjoint union of elements in A and B, limited to ones where they are alternating;

$$ab \cdot aa \cdot bba$$

- Operation is defined in the obvious way:

$$(ab \cdot aa \cdot bba) \cdot (a \cdot b) = ab \cdot aa \cdot bbaa \cdot b$$

Semigroup Coproduct

- We use this to separate each input letter into its own semigroup.
- Say $f(w) = w \cdot rev(w)$, with input aab

$$
\begin{aligned}
aab &\mapsto (a, a, b) && : (\Sigma^*)^3 \\
&\overset{h}{\mapsto} (h(a), h(a), h(b)) && : (\Sigma^* \times \Sigma^*)^3 \\
&= ((a, a), (a, a), (b, b)) && : (\Sigma^* \times \Sigma^*)^3 \\
&\mapsto ((\color{red}{a}, \color{red}{a}), (a, a), (\color{cyan}{b}, \color{cyan}{b})) && : ((\Sigma_1^* \oplus \Sigma_2^* \oplus \Sigma_3^*) \times (\Sigma_1^* \oplus \Sigma_2^* \oplus \Sigma_3^*))^3 \\
&\overset{\cdot}{\mapsto} (\color{red}{a} \cdot a \cdot \color{cyan}{b}, \color{cyan}{b} \cdot a \cdot \color{red}{a}) && : ((\Sigma_1^* \oplus \Sigma_2^* \oplus \Sigma_3^*) \times (\Sigma_1^* \oplus \Sigma_2^* \oplus \Sigma_3^*))^3 \\
&\overset{out}{\longmapsto} \color{red}{a} \cdot a \cdot \color{cyan}{bb} \cdot a \cdot \color{red}{a} && : (\Sigma_1^* \oplus \Sigma_2^* \oplus \Sigma_3^*)
\end{aligned}
$$

Status

- We have a way to keep track of letter origins.
- To define the updates from this we will need several operations.

Merge

- Say we have a coproduct $A_1 \oplus \cdots A_n$
- With some $I \subseteq \{1, \cdots n\}$ coordinates having the same semigroup A.
- We can then merge these coordinates:

$$ab \cdot aa \cdot bb \cdot bab \mapsto abaa \cdot bb \cdot bab$$

- This operation is of type

$$A_1 \oplus \cdots \oplus A_n \to A \oplus \bigoplus_{k \notin I} A_k$$

Shape

- We map each coordinate to the semigroup 1.
- We denote such a mapping as $! : A \to 1$.

$$ab \cdot aa \cdot bb \cdot bab \mapsto 1 \cdot 1 \cdot 1 \cdot 1$$

- This operation is of type

$$A_1 \oplus \cdots \oplus A_n \to 1 \oplus \cdots \oplus 1$$

View

- Pick a coordinate i.
- Apply ! and merge all other coordinates.

$$ab \cdot aa \cdot bb \cdot bab \mapsto 1 \cdot aa \cdot 1$$

- We took the view of the blue coordinate.
- For A_i this is type

$$A_1 \oplus \cdots \oplus A_n \to 1 \oplus A_i$$

Reconstruction

- A coproduct can be reconstructed using its *views* and *shape*.

$$\begin{matrix} ab & & & 1 & & & bab \\ 1 & & aa & & & 1 & \\ & 1 & & & bb & & 1 \\ 1 & & 1 & & 1 & & 1 \end{matrix} \mapsto ab \cdot aa \cdot bb \cdot bab$$

Transducer Semigroup Equivalence

- To create the updates we merge all coordinates *before* and *after* each i into different 1 semigroups.
- Say that for some transducer semigroup

$$a \cdot a \cdot b \mapsto ab \cdot aa \cdot bb \cdot bab$$

- Then we get the following:

$$ab \cdot aa \cdot bb \cdot bab \mapsto \begin{array}{cccccc} ab & & & 1_n & & bab \\ 1_p & & aa & & 1_n & 1_p \\ & 1_p & & & bb & 1_p \end{array}$$

- Updates are in $1 \oplus A \oplus 1$.
- The register will be in $1 \oplus A$.

Transducer Semigroup Equivalence

1

ab		1_n		bab
1_p	aa		1_n	1_p
	1_p		bb	1_p

Transducer Semigroup Equivalence

1

$ab \quad 1 \quad bab$

$\mapsto$

$ab \quad 1_n \quad bab$

$1_p \quad aa \quad 1_n \quad 1_p$

$1_p \quad bb \quad 1_p$

$1_p \quad aa \quad 1_n \quad 1_p$

$1_p \quad bb \quad 1_p$

Transducer Semigroup Equivalence

$$1 \mapsto \begin{matrix} ab & & 1_n & & bab \\ 1_p & aa & 1_n & & 1_p \\ & 1_p & bb & & 1_p \end{matrix}$$

$$ab \quad 1 \quad bab \mapsto \begin{matrix} 1_p & aa & 1_n & 1_p \\ & 1_p & bb & 1_p \end{matrix}$$

$$ab \quad aa \quad 1 \quad bab \mapsto 1_p \quad bb \quad 1_p$$

Transducer Semigroup Equivalence

1	$\mapsto$	$ab \quad 1_n \quad bab$ $1_p \quad aa \quad 1_n \quad 1_p$ $1_p \quad bb \quad 1_p$
$ab \quad 1 \quad bab$	$\mapsto$	$1_p \quad aa \quad 1_n \quad 1_p$ $1_p \quad bb \quad 1_p$
$ab \quad aa \quad 1 \quad bab$	$\mapsto$	$1_p \quad bb \quad 1_p$
$ab \quad aa \quad bb \quad bab$		

Transducer Semigroup Equivalence

- From a semigroup transducer, we now have the functions we wanted:

$$\begin{aligned} \mathit{update} &: \Sigma^* \to (1 \oplus A \oplus 1)^* \\ \delta &: (1 \oplus A) \times (1 \oplus A \oplus 1) \to (1 \oplus A) \end{aligned}$$

- We have succesfully transformed a transducer semigroup into something approximating an SST.

Recap

- Regular transductions are a class of string-to-string functions.
- There are many equivalent models available.
- We have covered SST, MSO, and transducer semigroups.

Recap

- Regular transductions are a class of string-to-string functions.
- There are many equivalent models available.
- We have covered SST, MSO, and transducer semigroups.
- Questions?