Transducers and Their Decision Problems

Sarah Winter

Université libre de Bruxelles, Belgium

16 December, 2022
No free lunch seminar @ UAntwerp

String-to-String Transductions

$$f : \Sigma^* \hookrightarrow \Sigma^*$$

$$f : \Sigma^* \hookrightarrow \Sigma^*$$

Append a # $\quad$ $abbab \mapsto abbab\#$

$$f : \Sigma^* \hookrightarrow \Sigma^*$$

Append a #	$abbab \mapsto abbab\#$
Delete all b	$abbab \mapsto aa$

$$f : \Sigma^* \hookrightarrow \Sigma^*$$

Append a #	$abbab \mapsto abbab\#$
Delete all b	$abbab \mapsto aa$
Squeeze all white space sequences ≥ 2	$friday␣␣␣16 \mapsto friday␣16$

$$f : \Sigma^* \hookrightarrow \Sigma^*$$

Append a #	$abbab \mapsto abbab\#$
Delete all b	$abbab \mapsto aa$
Squeeze all white space sequences ≥ 2	$friday␣␣␣16 \mapsto friday␣16$
Add a parity bit	$0100101 \mapsto \mathbf{1}0100101$

$$f : \Sigma^* \hookrightarrow \Sigma^*$$

Append a #	$abbab \mapsto abbab\#$
Delete all b	$abbab \mapsto aa$
Squeeze all white space sequences ≥ 2	$friday␣␣␣16 \mapsto friday␣16$
Add a parity bit	$0100101 \mapsto \mathbf{1}0100101$
Mirror the input word	$antwerp \mapsto prewtna$

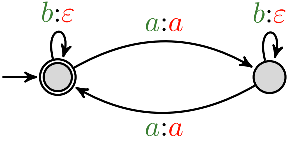
$$f : \Sigma^* \hookrightarrow \Sigma^*$$

Append a #	$abbab \mapsto abbab\#$
Delete all b	$abbab \mapsto aa$
Squeeze all white space sequences ≥ 2	$friday\text{␣␣␣}16 \mapsto friday\text{␣}16$
Add a parity bit	$0100101 \mapsto \mathbf{1}0100101$
Mirror the input word	$antwerp \mapsto prewtna$
Copy the input word	$yes \mapsto yesyes$

Transductions are defined by Transducers

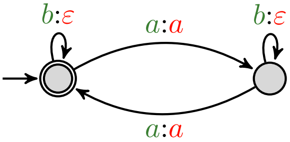
Transducers: Automata with output

f_{del} :



Transducers: Automata with output

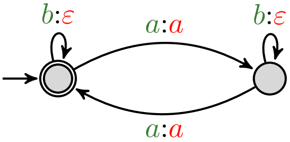
f_{del} :



$$aabaa \quad \mapsto \quad aaaa$$

Transducers: Automata with output

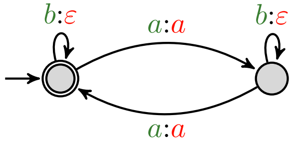
f_{del} :



$$\begin{array}{rcl} aabaa & \mapsto & aaaa \\ aaba & \mapsto & \text{undefined} \end{array}$$

Transducers: Automata with output

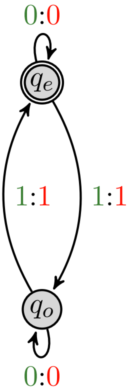
f_{del} :



$$\begin{array}{rcl} aabaa & \mapsto & aaaa \\ aaba & \mapsto & \text{undefined} \\ dom(f_{del}) & = & \text{`even number of } a\text{'} \end{array}$$

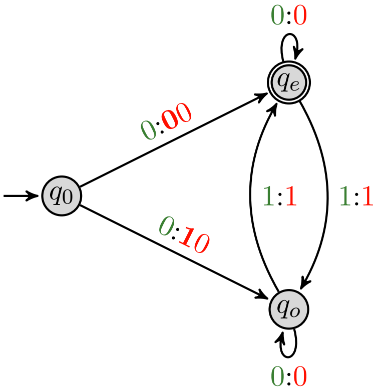
Parity bit

$01101 \mapsto \mathbf{1}01101, 01111 \mapsto \mathbf{0}01111$



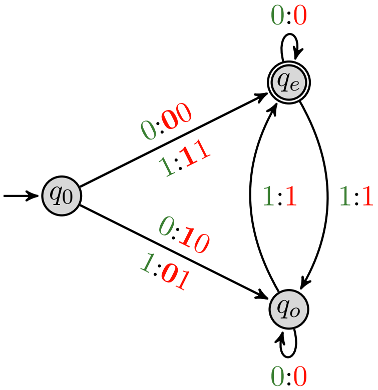
Parity bit

$01101 \mapsto \mathbf{1}01101, 01111 \mapsto \mathbf{0}01111$



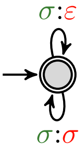
Parity bit

$01101 \mapsto \mathbf{1}01101, 01111 \mapsto \mathbf{0}01111$



Non-determinism and relations

In general, transducers define binary *relations* in $\Sigma^* \times \Sigma^*$



defines $\{(u, v) \mid v \text{ is a subword of } u\}$

Formal Definition

Definition

A (finite state) *transducer* is a tuple $T = (\Sigma, Q, I, F, \Delta)$ where:

- Σ is a finite alphabet
- Q is a finite set of states
- $I \subseteq Q$ are the initial states and $F \subseteq Q$ are the final states
- $\Delta \subseteq Q \times \Sigma \times \Sigma^* \times Q$ is the transition relation.

Formal Definition

Definition

A (finite state) *transducer* is a tuple $T = (\Sigma, Q, I, F, \Delta)$ where:

- Σ is a finite alphabet
- Q is a finite set of states
- $I \subseteq Q$ are the initial states and $F \subseteq Q$ are the final states
- $\Delta \subseteq Q \times \Sigma \times \Sigma^* \times Q$ is the transition relation.

Semantics

A *run* is a sequence of transitions

$$r = q_0 \xrightarrow{\sigma_1 : v_1} q_1 \ldots q_{n-1} \xrightarrow{\sigma_n : v_n} q_n \qquad \sigma_i \in \Sigma$$

Its input is $in(r) = \sigma_1 \ldots \sigma_n$ and its output $out(r) = v_1 \ldots v_n$.
The (rational) *relation* defined by T is:

$$[\![T]\!] = \{(in(r), out(r)) \mid r \text{ is an accepting run}\}$$

Application: Natural Language Processing

Table of nouns and their plural forms

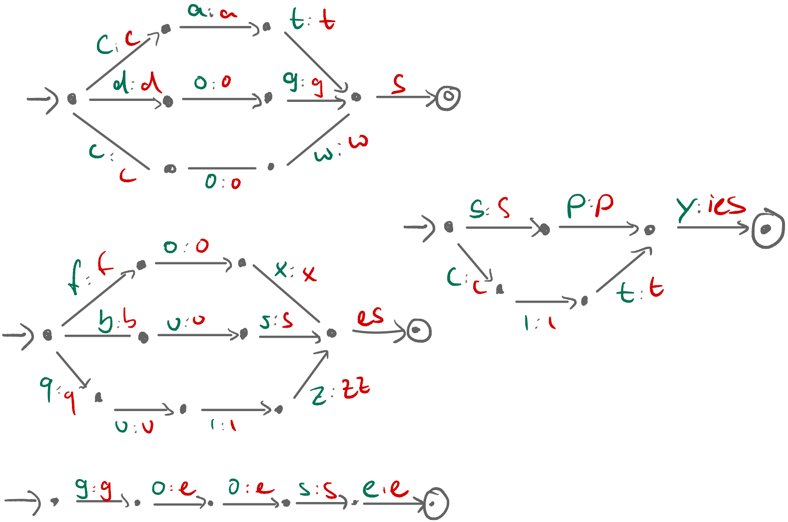
Noun singular	Noun plural
cat	cats
dog	dogs
cow	cows
fox	foxes
bus	buses
quiz	quizzes
goose	geese
spy	spies
city	cities
...	...

Table of nouns and their plural forms

Noun singular	**Noun plural**
cat	cats
dog	dogs
cow	cows
fox	foxes
bus	buses
quiz	quizzes
goose	geese
spy	spies
city	cities
...	...

- Inefficient representation

Lexicon + spelling rules: Transducer for plural forms



Properties of Transducers

Closure Properties

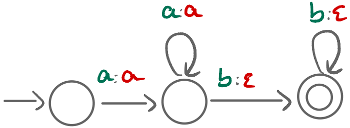
- Union

Closure Properties

- Union ✓
- Intersection

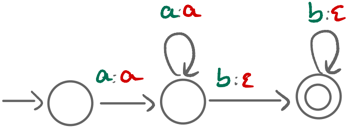
Closure Properties: Intersection

1. show that $\{(a^n b^m, a^n) \mid n, m > 0\}$ is rational.

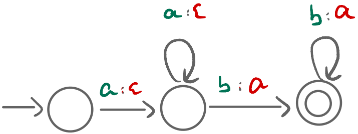


Closure Properties: Intersection

1. show that $\{(a^n b^m, a^n) \mid n, m > 0\}$ is rational.

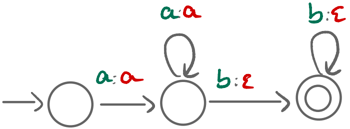


2. show that $\{(a^n b^m, a^m) \mid n, m > 0\}$ is rational.

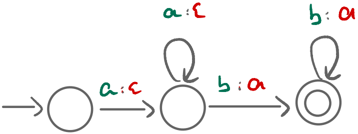


Closure Properties: Intersection

1. show that $\{(a^n b^m, a^n) \mid n, m > 0\}$ is rational.



2. show that $\{(a^n b^m, a^m) \mid n, m > 0\}$ is rational.



3. are rational relations closed under intersection ? why ?

$$\cap := \{(a^n b^m, a^n) \mid n = m > 0\}$$

Closure Properties

- Union ✓
- Intersection ✗
- Complement

Closure Properties

- Union ✓
- Intersection ✗
- Complement ✗
  Proof via De-Morgan's Law of sets : $\overline{\overline{A \cap B}} = \overline{\overline{A} \cup \overline{B}}$

Closure Properties

- Union ✓
- Intersection ✗
- Complement ✗
  Proof via De-Morgan's Law of sets : $\overline{\overline{A} \cap \overline{B}} = \overline{\overline{A} \cup \overline{B}}$
- Composition :
  $R_2 \circ R_1 = \{(u, w) \mid \exists (u, v) \in R_1, (v, w) \in R_2\}$.

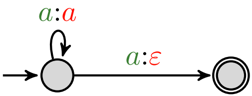
Closure Properties

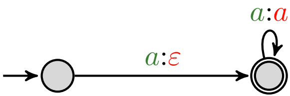
- Union ✓
- Intersection ✗
- Complement ✗
  Proof via De-Morgan's Law of sets : $\overline{\overline{A} \cap \overline{B}} = \overline{\overline{A} \cup \overline{B}}$
- Composition :
  $R_2 \circ R_1 = \{(u, w) \mid \exists (u, v) \in R_1, (v, w) \in R_2\}$. ✓

Transducer vs Automata

Transducer vs Automata

T :





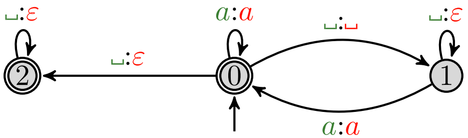
- Consider r_1, r_2 two runs on a^3. We have $(in(r_1), out(r_1)) = (in(r_2), out(r_2))$ but different in-out words:
$$(a, a)(a, a)(a, \varepsilon) \neq (a, \varepsilon)(a, a)(a, a)$$

Transducer vs Automata

T :

- Consider r_1, r_2 two runs on a^3. We have $(in(r_1), out(r_1)) = (in(r_2), out(r_2))$ but different in-out words:
$$(a, a)(a, a)(a, \varepsilon) \neq (a, \varepsilon)(a, a)(a, a)$$
- Transducers are *asynchronous*
- Make most transducer problems conceptually difficult (and even computationally).

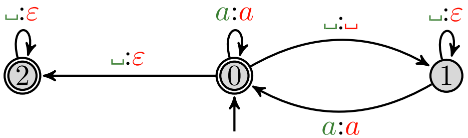
Determinizability: Example



␣ = white space

Determinizability: Example

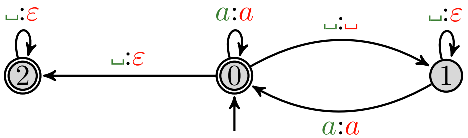
␣ = white space



$␣␣aa␣␣␣a␣␣ \quad \mapsto \quad ␣aa␣a$

Determinizability: Example

␣ = white space

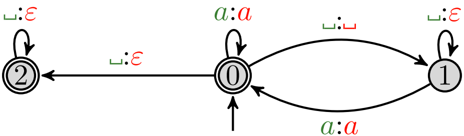


␣␣*aa*␣␣␣*a*␣␣ ↦ ␣*aa*␣*a*

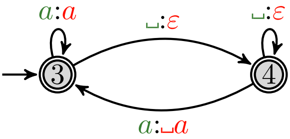
Is non-determinism needed ?

Determinizability: Example

␣ = white space



$␣␣aa␣␣␣a␣␣ \mapsto ␣aa␣a$



Is non-determinism needed ? No.

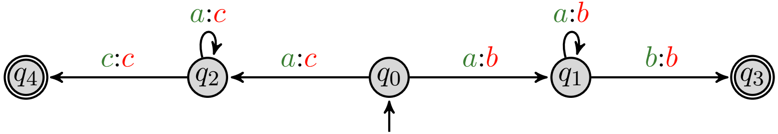
Can we always get an equivalent deterministic FT?

Can we always get an equivalent deterministic FT?

- not in general: input-deterministic transducers are less expressive than functional ones

Can we always get an equivalent deterministic FT?

- not in general: input-deterministic transducers are less expressive than functional ones

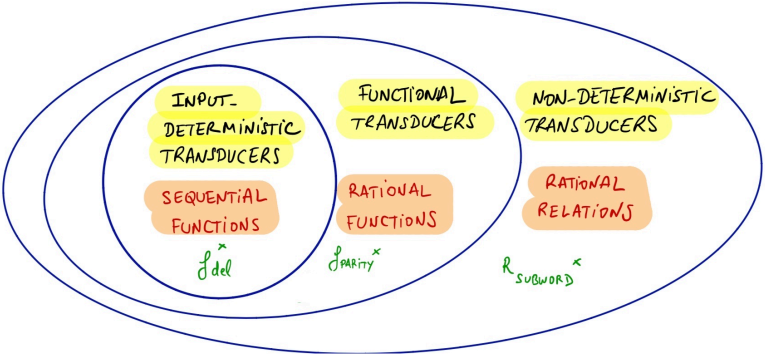


Semantics

$$[\![T]\!] : \begin{cases} a^n b \mapsto b^{n+1} \\ a^n c \mapsto c^{n+1} \end{cases}$$

functional but not determinizable

Different classes of transductions



Class Membership Problems

Functionality

Def Given a transducer T, is $[\![T]\!]$ a function?

Class Membership Problems

Functionality

Def Given a transducer T, is $[\![T]\!]$ a function?

Thm (Gurari, Ibarra 83, Carton, Beal, Prieur,Sakarovitch 03).
Functionality is decidable in PTime.

Class Membership Problems

Functionality

Def Given a transducer T, is $[\![T]\!]$ a function?

Thm (Gurari, Ibarra 83, Carton, Beal, Prieur,Sakarovitch 03).
Functionality is decidable in PTime.

Determinizability

Def Given a transducer T, does there exist an input-deterministic transducer T' such that $[\![T]\!] = [\![T']\!]$?

Class Membership Problems

Functionality

Def Given a transducer T, is $[\![T]\!]$ a function?

Thm (Gurari, Ibarra 83, Carton, Beal, Prieur,Sakarovitch 03).
Functionality is decidable in PTIME.

Determinizability

Def Given a transducer T, does there exist an input-deterministic transducer T' such that $[\![T]\!] = [\![T']\!]$?

Thm (Choffrut 77, Weber, Klemm 95).
Determinizability is decidable in PTIME.

Another Fundamental Problem: Equivalence

Def Given two transducers T_1, T_2, does $[\![T_1]\!] = [\![T_2]\!]$ hold?

Another Fundamental Problem: Equivalence

Def Given two transducers T_1, T_2, does $[\![T_1]\!] = [\![T_2]\!]$ hold?

Case of functional transducers

- Equivalence reduces to functionality:
  1. test whether $dom(T_1) = dom(T_2)$ PSPACE
  2. test whether $T_1 \uplus T_2$ is functional. PTIME

Another Fundamental Problem: Equivalence

Def Given two transducers T_1, T_2, does $[\![T_1]\!] = [\![T_2]\!]$ hold?

Case of functional transducers

- Equivalence reduces to functionality:
  1. test whether $dom(T_1) = dom(T_2)$
  2. test whether $T_1 \uplus T_2$ is functional.

General case

- Undecidable (Griffith 68),
- even if one alphabet is unary (Ibarra 78)

Summary – Transducers

Expressiveness:

input-deterministic	functional	non-deterministic
f_{del}	$<$ f_{parity}	$<$ $R_{subword}$

PTime PTime

Summary – Transducers

Expressiveness:

input-deterministic	functional	non-deterministic
f_{del} $<$	f_{parity} $<$	$R_{subword}$

PTime PTime

Equivalence: ($dom(T_1) = dom(T_2)$ is known)

input-deterministic	functional	non-deterministic
PTime	PTime	undec