Finite-state transducers and regular relations

A finite-state transducer (FST) is a finite automaton with the labels on its transitions doubled and allowed to be $\epsilon$, for a transition table $\delta \subseteq Q \times (\Sigma \cup \{\epsilon\}) \times (\Sigma \cup \{\epsilon\}) \times Q$ with

$$\delta(q, x, x', q')$$ written $q \xrightarrow{x:x'} q'$.

The fst $\langle \rightarrow, F \rangle$ computes the relation

$$\{(x_1 \ldots x_n, x'_1 \ldots x'_n) \in \Sigma^* \times \Sigma^* \mid (\exists q_1 \ldots q_n) q_0 \xrightarrow{x_1:x'_1} q_1 \xrightarrow{x_2:x'_2} \ldots q_n \xrightarrow{x_n:x'_n} q_n \in F\}.$$
Finite-state transducers and regular relations

A **finite-state transducer** (FST) is a finite automaton with the labels on its transitions doubled and allowed to be $\epsilon$, for a transition table $\delta \subseteq Q \times (\Sigma \cup \{\epsilon\}) \times (\Sigma \cup \{\epsilon\}) \times Q$ with

$$\delta(q, x, x', q') \text{ written } q \xrightarrow{x:x'} q'.$$

The fst $\langle \rightarrow, F \rangle$ **computes** the relation

$$\{(x_1 \ldots x_n, x'_1 \ldots x'_n) \in \Sigma^* \times \Sigma^* \mid (\exists q_1 \ldots q_n) q_0 \xrightarrow{x_1:x'_1} q_1 \xrightarrow{x_2:x'_2} \ldots \xrightarrow{x_n:x'_n} q_n \in F\}.$$

**N.B.** $x_1 \ldots x_n$ and $x'_1 \ldots x'_n$ may have different lengths, as an $x_i$ and/or $x'_i$ can be $\epsilon$ (of length 0).
Finite-state transducers and regular relations

A finite-state transducer (FST) is a finite automaton with the labels on its transitions doubled and allowed to be $\epsilon$, for a transition table $\delta \subseteq Q \times (\Sigma \cup \{\epsilon\}) \times (\Sigma \cup \{\epsilon\}) \times Q$ with

$$\delta(q, x, x', q') \text{ written } q \xrightarrow{x:x'} q'.$$

The fst $\langle \rightarrow, F \rangle$ computes the relation

$$\{(x_1 \ldots x_n, x'_1 \ldots x'_n) \in \Sigma^* \times \Sigma^* \mid (\exists q_1 \ldots q_n) q_0 \xrightarrow{x_1:x'_1} q_1 \xrightarrow{x_2:x'_2} \ldots q_n \xrightarrow{x_n:x'_n} q_n \in F\}.$$

**N.B.** $x_1 \ldots x_n$ and $x'_1 \ldots x'_n$ may have different lengths, as an $x_i$ and/or $x'_i$ can be $\epsilon$ (of length 0).

A relation between strings is regular if it is computed by some fst.
Define
\[ \text{fst}(+\text{Input}, +\text{Trans}, +\text{Final}, ?\text{Output}) \]

\textbf{Caution:} consider \text{fst} computing \(1 \times 1^+\) given by
\[ \text{Trans} = [[[q_0, []], 1, q_0], [q_0, 1, 1, q_1]] \]
\[ \text{Final} = [q_1] \]

That is, the length \(n\) of a pair \((x_1 \ldots x_n, x'_1 \ldots x'_n)\) no longer bounds a run computing it.
Some regular relations

1. The factor relation

\[ s \text{ hasFactor } s' \iff (\exists u, v) \quad s = us'v \]
Some regular relations

1. The factor relation

\[ s \text{ hasFactor } s' \iff (\exists u, v) \ s = us'v \]

2. The accepting runs of a finite automaton \( \rightarrow, F \)

\[ \{ \langle a_1 a_2 \cdots a_n, q_1 q_2 \cdots q_n \rangle \mid q_0 \xrightarrow{a_1} q_1 \xrightarrow{a_2} q_2 \cdots \xrightarrow{a_n} q_n \in F \} \]

mixing symbols/actions \( a_i \) with states/situations \( q_i \)
Some regular relations

1. The factor relation

\[ s \text{ hasFactor } s' \iff (\exists u, v) \ s = us'v \]

2. The accepting runs of a finite automaton \( \rightarrow, F \)

\[ \{ \langle a_1 a_2 \cdots a_n, q_1 q_2 \cdots q_n \rangle \mid q_0 \xrightarrow{a_1} q_1 \xrightarrow{a_2} q_2 \cdots \xrightarrow{a_n} q_n \in F \} \]

mixing symbols/actions \( a_i \) with states/situations \( q_i \)

3. The diagonal \( \Delta_L \) of a regular language \( L \)

\[ \Delta_L := \{ (s, s) \mid s \in L \} \]
Some regular relations

1. The factor relation

\[ s \text{ hasFactor } s' \iff (\exists u, v) \ s = us'v \]

2. The accepting runs of a finite automaton \( \rightarrow, F \)

\[ \{ \langle a_1a_2\cdots a_n, q_1q_2\cdots q_n \rangle \mid q_0 \xrightarrow{a_1} q_1 \xrightarrow{a_2} q_2 \cdots \xrightarrow{a_n} q_n \in F \} \]

mixing symbols/actions \( a_i \) with states/situations \( q_i \)

3. The diagonal \( \Delta_L \) of a regular language \( L \)

\[ \Delta_L := \{(s, s) \mid s \in L\} \]

4. \( A \)-string-meronym \( \succeq_A \) on \((2^A)^*\)

\[ \alpha_1 \cdots \alpha_n \succeq_A x_1 \cdots x_n \iff x_i = \epsilon \text{ or } x_i \subseteq \alpha_i \text{ for } 1 \leq i \leq n \]

for \( \alpha_1 \cdots \alpha_n \in (2^A)^* \).
Some closure properties

1. If $R$ is regular, so is its inverse $R^{-1}$. 
Some closure properties

1. If $R$ is regular, so is its inverse $R^{-1}$.
2. If $R$ and $R'$ are regular, so are its union $R \cup R'$ and relational composition

   $R; R' := \{(s, s') \mid (\exists s_0) sRs_0 \text{ and } s_0R's'\}$.
Some closure properties

1. If $R$ is regular, so is its inverse $R^{-1}$.
2. If $R$ and $R'$ are regular, so are its union $R \cup R'$ and relational composition

$$R; R' := \{(s, s') \mid (\exists s_0) sRs_0 \text{ and } s_0R's'\}.$$ 

3. The restriction $R_L$ of a regular relation $R$ to a regular language $L$

$$R_L := \{(s, s') \in R \mid s \in L\}$$
Some closure properties

1. If $R$ is regular, so is its inverse $R^{-1}$.

2. If $R$ and $R'$ are regular, so are its union $R \cup R'$ and relational composition

   \[ R; R' := \{(s, s') \mid (\exists s_0) sRs_0 \text{ and } s_0 R's'\}. \]

3. The restriction $R_L$ of a regular relation $R$ to a regular language $L$

   \[ R_L := \{(s, s') \in R \mid s \in L\} \]

If $R$ is a regular relation, then its image $\{s' \mid (\exists s) sRs'\}$ is regular including

\[ L \cap L' = \text{image}(\Delta_L; \Delta_{L'}) \]

and the Peirce product

\[ R^{-1}L := \{s \mid (\exists s' \in L) sRs'\} = \text{image}(R^{-1}_L) \]

12 / 21
Regular relations are not Boolean-closed

Regular relations are *not* closed under intersection —

\[
\{\langle 0^n, 1^n 2^m \rangle \mid n \geq 0, m \geq 0 \} \quad \text{and} \quad \{\langle 0^n, 1^m 2^n \rangle \mid n \geq 0, m \geq 0 \}
\]

are regular, but their intersection has image \( \sum_{n \geq 0} 1^n 2^n \).
Regular relations are not Boolean-closed

Regular relations are *not* closed under intersection —

\[
\{\langle 0^n, 1^n2^m \rangle \mid n \geq 0, m \geq 0\} \quad \text{and} \quad \{\langle 0^n, 1^m2^n \rangle \mid n \geq 0, m \geq 0\}
\]

are regular, but their intersection has image \( \sum_{n \geq 0} 1^n2^n \).

Hence, the complement \( \overline{R} \) of a regular relation \( R \) need not be regular, as

\[
R \cap R' = \overline{R \cup R'}
\]
TM-actions via finite-state transducers

moveRight$(q, a, q')$
\[ \vec{l}qar \leadsto \vec{l}aq'r \]

moveLeft$(q, a, q')$
\[ \vec{l}a'qar \leadsto \vec{l}q'a'ar \]

write$(q, a, a', q')$
\[ \vec{l}qar \leadsto \vec{l}q'a'r \]

For any Turing machine $M$, step$M$ is regular (finite tuples)
TM-actions via finite-state transducers

moveRight\((q, a, q')\) \quad \vec{l}qa\vec{r} \rightsquigarrow \vec{la}q'\vec{r}

moveLeft\((q, a, q')\) \quad \vec{l}a'qa\vec{r} \rightsquigarrow \vec{la}q'\vec{a}r

write\((q, a, a', q')\) \quad \vec{l}qa\vec{r} \rightsquigarrow \vec{l}q'a'\vec{r}

tape infinite to the right \quad \vec{l}q \approx \vec{l}q\#

tape infinite to the left \quad q\vec{r} \approx \vec{#q}\vec{r}
TM-actions via finite-state transducers

moveRight \((q, a, q')\):
\[ \vec{l}q a \bar{r} \leadsto \vec{l}a q' \bar{r} \]
moveRight \((q, \#, q')\):
\[ \vec{l}q \leadsto \vec{l}\# q' \]
moveLeft \((q, a, q')\):
\[ \vec{l}a' q a \bar{r} \leadsto \vec{l}q' a' a \bar{r} \]
moveLeft \((q, a, q')\):
\[ q a \bar{r} \leadsto q' \# a \bar{r} \]
moveLeft \((q, \#, q')\):
\[ \vec{l}a q \leadsto \vec{l}q' a \]
moveLeft \((q, \#, q')\):
\[ q \leadsto q' \]
write \((q, a, a', q')\):
\[ \vec{l}q a \bar{r} \leadsto \vec{l}q' a' \bar{r} \]
write \((q, \#, a, q')\):
\[ \vec{l}q \leadsto \vec{l}q' a \]

tape infinite to the right:
\[ \vec{l}q \approx \vec{l}q\# \]

tape infinite to the left:
\[ q \bar{r} \approx \# q \bar{r} \]
TM-actions via finite-state transducers

\[
\begin{align*}
\text{moveRight}(q, a, q') & \quad \vec{t}qa\vec{r} \sim \vec{t}aq'\vec{r} \\
\text{moveRight}(q, \#, q') & \quad \vec{t}q \sim \vec{t}\#q' \\
\text{moveLeft}(q, a, q') & \quad \vec{t}a'qa\vec{r} \sim \vec{t}q'a'ar\vec{r} \\
\text{moveLeft}(q, a, q') & \quad qar\vec{r} \sim q'\#ar\vec{r} \\
\text{moveLeft}(q, \#, q') & \quad \vec{t}aq \sim \vec{t}q'a \\
\text{moveLeft}(q, \#, q') & \quad q \sim q' \\
\text{write}(q, a, a', q') & \quad \vec{t}qa\vec{r} \sim \vec{t}q'a'\vec{r} \\
\text{write}(q, \#, a, q') & \quad \vec{t}q \sim \vec{t}q'a \\
\text{tape infinite to the right} & \quad \vec{t}q \approx \vec{t}q\# \\
\text{tape infinite to the left} & \quad q\vec{r} \approx \#q\vec{r}
\end{align*}
\]

For any Turing machine $M$, $\text{step}_M$ is regular (finite tuples)
Finite-state approximations

Output extraction $\leadsto$ via finite-state transducer

$$\text{halt}(q, a) \quad \vec{l}qa\vec{r} \leadsto \text{unpad}(\vec{l}ar)$$

$$\text{halt}(q, \#) \quad \vec{l}q \leadsto \text{unpad}(\vec{l})$$

$n$-step approximation of a TM $M$

$$M_n := \{(s, s') \mid q_0s \overset{n}{\text{step}}_M; \leadsto s'\}$$

$$(\exists x) \ q_0s \overset{n}{\text{step}}_M x \text{ and } x \leadsto s'$$

Bounded iterations (time-out clock) are regular

$$\text{input/output}(M) = \bigcup_{n \geq 0} M_n$$
% Solution to fst exercise (slide 4 above)

fst(Input, Trans, Final, Output):-
    sfst([[q0,Input,[]]], Trans, Final, RevEps),
    reverseEps(RevEps,Output).

sfst([[State,[]],RE]|_], _, Final, RE) :-
    member(State, Final).

sfst([Node|More], Trans, Final, RE) :-
    findall(Next, arc(Node,Next,Trans), Children),
    append(More,Children,New),
    sfst(New, Trans, Final, RE).

arc([[State,In,PartOut], [Next,In,[X|PartOut]], Trans) :-
    member([[State, [], X, Next]], Trans).

arc([[State,[H|T],PartOut], [Next,T,[X|PartOut]], Trans) :-
    member([[State, H, X, Next]], Trans).
reverseEps(RevEps, Out) :- reverseEps(RevEps, [], Out).

reverseEps([], Ac, Ac).

reverseEps([[] | More], Ac, Out) :- !,
    reverseEps(More, Ac, Out).

reverseEps([X | More], Ac, Out) :-
    reverseEps(More, [X | Ac], Out).

% test for 1 \times 1^+
test(Out):- fst([1], [[q0, [], 1, q0], [q0, 1, 1, q1]], [q1], Out).