5. Implementing operations on relations using finite automata

We discuss how to implement operations on relations over a (possibly infinite) universe U. Even though we will encode the elements of U as words, when implementing relations it is convenient to think of U as an abstract universe, and not as the set Σ^* of words over some alphabet Σ. The reason is that for some operations we encode an element of X not by one word, but by many, actually by infinitely many. In the case of operations on sets this is not necessary, and one can safely identify the object and its encoding as word.

We shall consider a number of operations on relations, some of which are closely related to operations on sets, which we have discussed above. For other types of operations:

Recall:

$\mathsf{Projection}_1(R)$	:	returns the set $\pi_1(R) = \{x;\ (\exists x)[(x,y) \in R]\}$
$\mathsf{Projection}_2(R)$	:	returns the set $\pi_2(R) = \{y;\ (\exists y)[(x,y) \in R]\}$
$\mathsf{Join}(R,S)$	:	returns $R \circ S = \{(x,z);\ (\exists y)[(x,y) \in R \wedge (y,z) \in S]\}$
$\mathsf{Post}(X,R)$	:	returns $\mathsf{post}_R(X) = \{y \in U;\ (\exists x \in X)[(x,y) \in R]\}$
$\mathsf{Pre}(X,R)$	:	returns $\mathsf{pre}_R(X) = \{y \in U;\ (\exists x \in X)[(y,x) \in R]\}$

Encoding objects

- So far we have assumed for convenience:
  a) every word encodes one object.
  b) every object is encoded by exactly one word.
  We now analyze this in more detail.

- Example: objects → natural number encoding → *lsbf*
  lsbf(5) = 101 *lsbf*(0) = ε.
  Satisfies b), but not a).

- We argue that a) can be easily weakened to:
  a') the set of words encoding objects is a regular language.

- The *lsbf* encoding satisfies a'):
  set of encodings → $\{\varepsilon\} \cup \{w \in \Sigma^* \mid w \text{ ends with } 1\}$

Encoding pairs

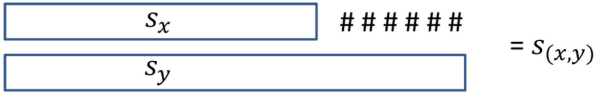
- Extending the implementations to relations requires to encode pairs of objects.
- How should we encode a pair (n_1, n_2) of natural numbers?

- Consider the pair (n_1, n_2) .
- Assume n_1, n_2 encoded by w_1, w_2 in *lsbf* encoding
- Which should be the encoding of (n_1, n_2) ?
  - Cannot be $w_1 w_2$.
    Then same word encodes many pairs, violates b).

- First attempt: use a separator symbol &, and encode (n_1, n_2) by $w_1 \& w_2$.
  - Problem: not even the identity relation gives a regular language!

- Second attempt: encode (n_1, n_2) as a word over $\{0,1\} \times \{0,1\}$ (intuitively, the automaton reads w_1 and w_2 simultaneously).
  - Problem: what if w_1 and w_2 have different length?
  - Solution: fill the shortest one with 0s.
  - Satisfies b) and a'), but not (a):
    - The number k is encoded by all the words of $s_k 0^*$, where s_k is the *lsbf* encoding of k.
  - We call 0 the padding symbol or padding letter.

- So we assume:
  - The alphabet contains a padding letter #, different or not from the letters used to encode an object.
  - Each object x has a minimal encoding s_x .
  - The encodings of x are all the words of $s_x\#^*$.
  - A pair (x, y) of objects has a minimal encoding $s_{(x,y)}$.



  - The encodings of (x, y) are all the words of $s_{(x,y)}\#^*$.

- Question: if objects (pairs of objects) are encoded by multiple words, which is the set of objects (pairs) recognized by a DFA or NFA?

  (We can no longer say: an object is recognized if its encoding is accepted by the DFA or NFA!)

- Question: because of the new definition of "set of objects recognized by an automaton", do we have to change the implementation of the set operations?

Definition 38

Assume an encoding of the universe U over Σ^* has been fixed. Let A be an NFA.

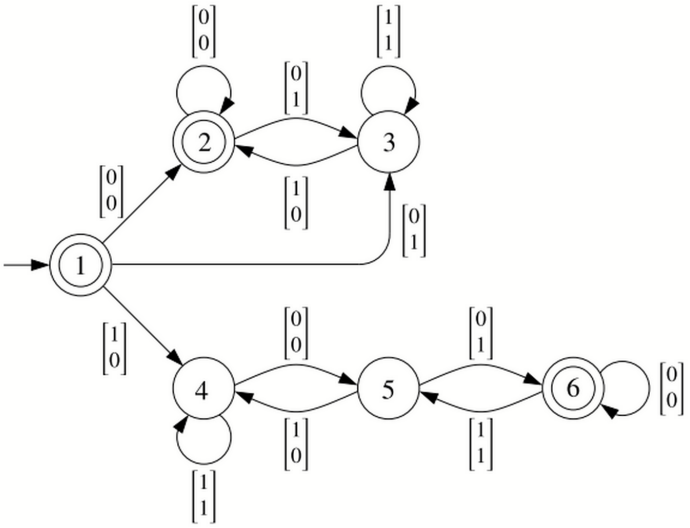
- A accepts $x \in U$ if it accepts all encodings of x.
- A rejects $x \in U$ if it accepts no encoding of x.
- A recognizes a set $X \subseteq U$ if

$$L(A) = \{w \in \Sigma^*;\ \ w \text{ encodes some element of } X\}\ .$$

A set is regular (with respect to the fixed encoding) if it is recognized by some NFA.

Notice that if A recognizes $X \subseteq U$ then, as one would expect, A accepts every $x \in X$ and rejects every $x \notin X$. Hence, with this definition, it may be the case that an NFA neither accepts nor rejects a given x. An NFA is well-formed if it recognizes some set of objects, and ill-formed otherwise.

Transducers



Definition 39
A transducer over Σ is an NFA over the alphabet $\Sigma \times \Sigma$.

Transducers are also called Mealy machines.

According to this definition, a transducer accepts sequences of pairs of letters, but it is convenient to look at it as a machine accepting pairs of words:

Definition 40

Let T be a transducer over Σ. Given words $w_1 = a_1 a_2 \dots a_n$ and $w_2 = b_1 b_2 \dots b_n$, we say that T accepts the pair (w_1, w_2) if it accepts the word $(a_1, b_1) \dots (a_n, b_n) \in (\Sigma \times \Sigma)^*$.

Definition 41

Let T be a transducer.

- T accepts a pair $(x, y) \in X \times X$ if it accepts all encodings of (x, y).
- T rejects a pair $(x, y) \in X \times X$ if it accepts no encoding of (x, y).
- T recognizes a relation $R \subseteq X \times X$ if

$$L(T) = \{(w_x, w_y) \in (\Sigma \times \Sigma)^*;\ (w_x, w_y) \text{ encodes some pair of } R\}\ .$$

A relation is regular if it is recognized by some transducer.

Examples of regular relations on numbers (*lsbf* encoding):

— the identity relation $\{ (n, n) \; ; \; n \in \mathbb{N}_0 \}$

— the relation "is double of" $\{ (n, 2n) \; ; \; n \in \mathbb{N}_0 \}$

Example 42

The Collatz function is the function $f : \mathbb{N} \to \mathbb{N}$ defined as follows:

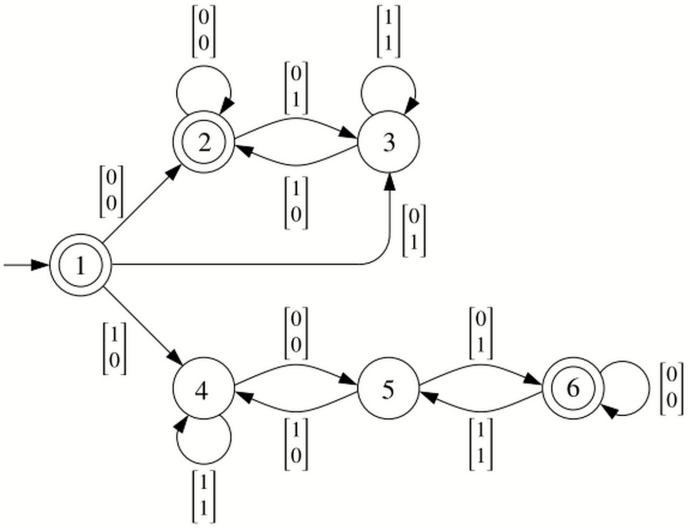
$$f(n) = \begin{cases} 3n+1 & \text{if } n \text{ is odd} \\ n/2 & \text{if } n \text{ is even} \end{cases}$$

We next show a transducer that recognizes the relation $\{(n, f(n));\ n \in \mathbb{N}\}$ with *lsbf* encoding and with $\Sigma = \{0, 1\}$. The elements of $\Sigma \times \Sigma$ are represented as column vectors with two components. The transducer accepts for instance the pair $(7, 22)$ because it accepts the pairs $(111000^k, 011010^k)$, that is, it accepts

$$\begin{bmatrix}1\\0\end{bmatrix}\begin{bmatrix}1\\1\end{bmatrix}\begin{bmatrix}1\\1\end{bmatrix}\begin{bmatrix}0\\0\end{bmatrix}\begin{bmatrix}0\\1\end{bmatrix}\left(\begin{bmatrix}0\\0\end{bmatrix}\right)^k$$

for every $k \geq 0$.

Transducers



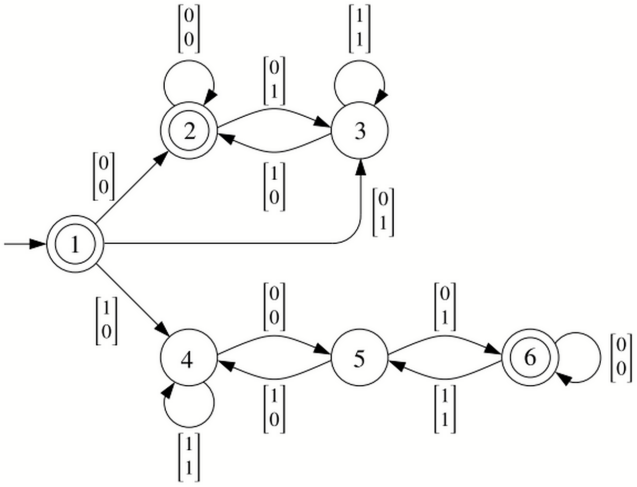
Determinism

- A transducer is deterministic if it is a DFA.
- Observe: if Σ has size n, then a state of a deterministic transducer with alphabet Σ x Σ has n^2 outgoing transitions.
- Warning! There is a different definition of determinism:
  - A letter $\begin{bmatrix} a \\ b \end{bmatrix}$ is interpreted as "output b on input a"
  - Deterministic transducer: only one move (and so only one output) for each input.

- Before implementing the new operations:
  - How do we check membership?
  - Can we compute union, intersection and complement of relations as for sets?

Implementing the operations

Projection



- Deleting the second component is not correct
  - Counterexample: $R = \{ (4,1) \}$
  - $s_{(4,1)} = \begin{bmatrix}0\\1\end{bmatrix}\begin{bmatrix}0\\0\end{bmatrix}\begin{bmatrix}1\\0\end{bmatrix}$
  - DFA for R:

Proj_1(T)
Input: transducer $T = (Q, \Sigma \times \Sigma, \delta, q_0, F)$
Output: NFA $A = (Q', \Sigma, \delta', q_0', F')$ with $\mathcal{L}(A) = \pi_1(\mathcal{L}(T))$

```
1  Q' ← Q; q'_0 ← q_0; F'' ← F
2  δ' ← ∅;
3  for all (q, (a, b), q') ∈ δ do
4     add (q, a, q') to δ'
5  F' ← PadClosure((Q', Σ, δ', q'_0, F''), #)
```

PadClosure$(A, \#)$
Input: NFA $A = (\Sigma \times \Sigma, Q, \delta, q_0, F)$
Output: new set F' of final states

```
1  W ← F; F' ← ∅;
2  while W ≠ ∅ do
3     pick q from W
4     add q to F'
5     for all (q', #, q) ∈ δ do
6        if q' ∉ F' then add q' to W
7  return F'
```

- Problem: we may be accepting $s_x \#^k \#^*$ instead of $s_x \#^*$ and so according to the definition we are not acepting x !

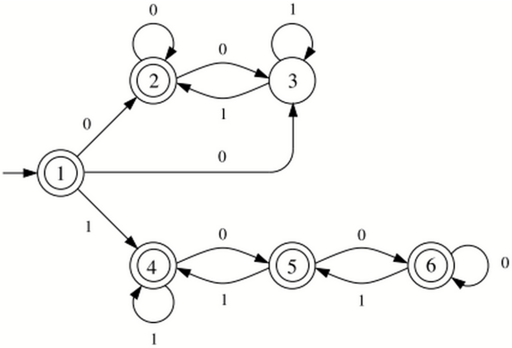
- Solution: if after eliminating the second components some non-final state goes with # ... # to a final state, we mark the state as final.

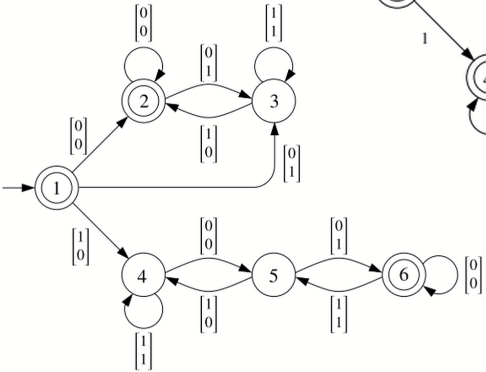
- Complexity: linear in the size of the transducer

- Observe: the result of a projection may be a NFA, even if the transducer is deterministic!!

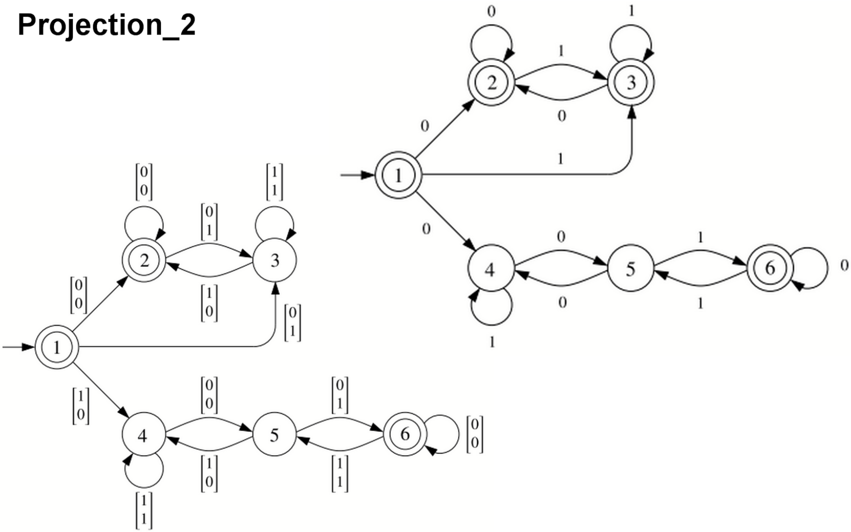
- This is the operation that prevents us from implementing all operations directly on DFAs.

Projection_1





Projection_2



Correctness proof

- Assume: transducer T recognizes a set of pairs
- Prove: the projection automaton A recognizes a set, and this set is the projection onto the first component of the set of pairs recognized by T.

a) A accepts either all encodings or no encoding of an object.
Assume A accepts at least one encoding w of an object x.
We prove it accepts all.
If A accepts w, then T accepts $\begin{matrix} w \\ w' \end{matrix}$ for some w'. By assumption T accepts $\begin{matrix} w \\ w' \end{matrix} \begin{bmatrix} \# \\ \# \end{bmatrix}^*$, and so A accepts $w\ \#^*$.
Moreover, $w = s_x \#^k$ for some $k > 0$, and so, by padding closure, A also accepts $s_x \#^j$ for every $j < k$.

b) A only accepts words that are encodings of objects.
Follows easily from the fact that T satisfies the same property for pairs of objects.

Correctness proof

c) If A accepts an object x , then there is an object y such that T accepts (x, y) .

x accepted by A

$\Rightarrow$ s_x accepted by A (part a)

$\Rightarrow$ $\begin{matrix} s_x \\ w \end{matrix}$ accepted by T for some w

By assumption, T only accepts pairs of words encoding some pair of objects. So w encodes some object y. By assumption, T then accepts all encodings of (x, y). So T accepts (x, y) .

Correctness proof

d) If a pair of objects (x, y) is accepted by T, then x is accepted by A.

(x, y) accepted by T

$\Rightarrow$ $\begin{matrix} w_x \\ w_y \end{matrix}$ accepted by T for some encodings w_x, w_y of x and y

$\Rightarrow$ w_x accepted by A

$\Rightarrow$ x accepted by A (part a))

Remember:

The projection automaton of a deterministic transducer may be nondeterministic.

Joi

- Goal: given transducers T_1, T_2 recognizing relations R_1, R_2, construct a transducer $T_1 \circ T_2$ recog∘nizing the relation $R_1 \circ R_2$.

- First step: construct a transducer T that accepts $\begin{matrix} w \\ v \end{matrix}$ iff there is a "connecting" word u such that

  $\begin{matrix} w \\ u \end{matrix}$ is accepted by T_1 and $\begin{matrix} u \\ v \end{matrix}$ is accepted by T2.

  We slightly modify the pairing construction.

Instead of:

$$\begin{bmatrix} q_{01} \\ q_{02} \end{bmatrix} \xrightarrow{a_1} \begin{bmatrix} q_{11} \\ q_{12} \end{bmatrix} \quad \text{iff} \quad \begin{array}{l} q_{01} \xrightarrow{a_1} q_{11} \\ q_{02} \xrightarrow{a_1} q_{12} \end{array}$$

we now use

$$\begin{bmatrix} q_{01} \\ q_{02} \end{bmatrix} \xrightarrow{\begin{bmatrix} a_1 \\ b_1 \end{bmatrix}} \begin{bmatrix} q_{11} \\ q_{12} \end{bmatrix} \quad \text{iff} \quad \begin{array}{l} q_{01} \xrightarrow{\begin{bmatrix} a_1 \\ c_1 \end{bmatrix}} q_{11} \\ q_{02} \xrightarrow{\begin{bmatrix} c_1 \\ b_1 \end{bmatrix}} q_{12} \end{array}$$

for some letter c1

The transducer T has a run

$$\begin{bmatrix} q_{01} \\ q_{02} \end{bmatrix} \xrightarrow{\begin{bmatrix} a_1 \\ b_1 \end{bmatrix}} \begin{bmatrix} q_{11} \\ q_{12} \end{bmatrix} \xrightarrow{\begin{bmatrix} a_2 \\ b_2 \end{bmatrix}} \begin{bmatrix} q_{21} \\ q_{22} \end{bmatrix} \quad \cdots \quad \begin{bmatrix} q_{(n-1)1} \\ q_{(n-1)2} \end{bmatrix} \xrightarrow{\begin{bmatrix} a_n \\ b_n \end{bmatrix}} \begin{bmatrix} q_{n1} \\ q_{n2} \end{bmatrix}$$

iff T_1 and T_2 have runs

$$\begin{array}{lcllcllcl} q_{01} & \xrightarrow{\begin{bmatrix} a_1 \\ c_1 \end{bmatrix}} & q_{11} & \xrightarrow{\begin{bmatrix} a_2 \\ c_2 \end{bmatrix}} & q_{21} & \cdots & q_{(n-1)1} & \xrightarrow{\begin{bmatrix} a_n \\ c_n \end{bmatrix}} & q_{n1} \\ q_{02} & \xrightarrow{\begin{bmatrix} c_1 \\ b_1 \end{bmatrix}} & q_{12} & \xrightarrow{\begin{bmatrix} c_2 \\ b_2 \end{bmatrix}} & q_{22} & \cdots & q_{(n-1)2} & \xrightarrow{\begin{bmatrix} c_n \\ b_n \end{bmatrix}} & q_{n2} \end{array}$$

- We have the same problem as before.
- Let $R_1 = \{ (2,4) \}$, $\mathrm{R}_2 = \{ (4,2) \}$.
  Then $R_1 \circ R_2 = \{ (2,2) \}$.
- But the operation we have just defined does not yield the correct result.
- Solution: apply the padding closure again with padding symbol $\begin{bmatrix} \# \\ \# \end{bmatrix}$.

$Join(T_1, T_2)$
Input: transducers $T_1 = (Q_1, \Sigma \times \Sigma, \delta_1, q_{01}, F_1)$, $T_2 = (Q_2, \Sigma \times \Sigma, \delta_2, q_{02}, F_2)$
Output: transducer $T_1 \circ T_2 = (Q, \Sigma \times \Sigma, \delta, q_0, F)$

1 $Q, \delta, F' \leftarrow \emptyset$; $q_0 \leftarrow [q_{01}, q_{02}]$
2 $W \leftarrow \{[q_{01}, q_{02}]\}$
3 **while** $W \neq \emptyset$ **do**
4 **pick** $[q_1, q_2]$ **from** W
5 **add** $[q_1, q_2]$ **to** Q
6 **if** $q_1 \in F_1$ and $q_2 \in F_2$ **then add** $[q_1, q_2]$ **to** F'
7 **for all** $(q_1, (a, c), q_1') \in \delta_1, (q_2, (c, b), q_2') \in \delta_2$ **do**
8 **add** $([q_1, q_2], (a, b), [q_1', q_2'])$ **to** δ
9 **if** $[q_1', q_2'] \notin Q$ **then add** $[q_1', q_2']$ **to** W
10 $F \leftarrow$ **PadClosure**$((Q, \Sigma \times \Sigma\delta, q_0, F'), (\#, \#))$

Complexity: similar to pairing

- Example:
  - Let f be the Collatz function.
  - Let $R_1 = R_2 = \{ (n, f(n)) \mid n \geq 0 \}$.
  - Then $R_1 \circ R_2 = \{ (n, f(f(n))) \mid n \geq 0 \}$.

$$f(f((n)) = \begin{cases} n/4 & \text{if } n \equiv 0 \, mod \, 4 \\ 3n/2 + 1 & \text{if } n \equiv 2 \, mod \, 4 \\ 3n/2 + 1/2 & \text{if } n \equiv 1 \, mod \, 4 \text{ or } n \equiv 3 \, mod \, 4 \end{cases}$$