Operations on relations: Implementation on NFAs

Projection_1(R) : returns the set $\pi_1(R) = \{x \mid \exists y\ (x, y) \in R\}$.
Projection_2(R) : returns the set $\pi_2(R) = \{y \mid \exists y\ (x, y) \in R\}$.
Join(R_1, R_2) : returns $R_1 \circ R_2 = \{(x, z) \mid \exists y \in X\ (x, y) \in R_1 \wedge (y, z) \in R_2\}$

Post(Y, R) : returns $post_R(Y) = \{x \in X \mid \exists y \in Y : (y, x) \in R\}$.
Pre(Y, R) : returns $pre_R(Y) = \{x \in X \mid \exists y \in Y' : (x, y) \in R\}$.

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 numbers
encoding --> lsbf: 5 --> 101, 0 --> epsilon.

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.

Satisfied by the lsbf encoding:

set of encodings --> {epsilon} U words ending with 1

Encoding pairs

Extending the implementations to relations requires to encode pairs of objects.

How should we encode a pair (n1,n2) of natural numbers?

Consider the pair (n1, n2).
Assume n1, n2 encoded by w1, w2 in lsbf encoding

Which should be the encoding of (n1,n2) ?

- Cannot be w1w2
  (then same word encodes many pairs, violates (b)).

- First attempt: use a separator symbol &, and encode (n1, n2) by w1&w2

  Problem: not even the identity relation gives a regular language!

- Second attempt:

Encode (n1, n2) as a word over {0,1} x {0,1}
(intuitively, the automaton reads w1 and w2 simultaneously)

Problem: what if w1 and w2 have different length?
Solution: fill the shortest one with 0s.

Satisfies (b).
Satisfies (a'), but not (a):
A number k is encoded by all the words of s_k0*,
where s_k is the lsbf encoding of k.

We call 0 the padding symbol or letter.

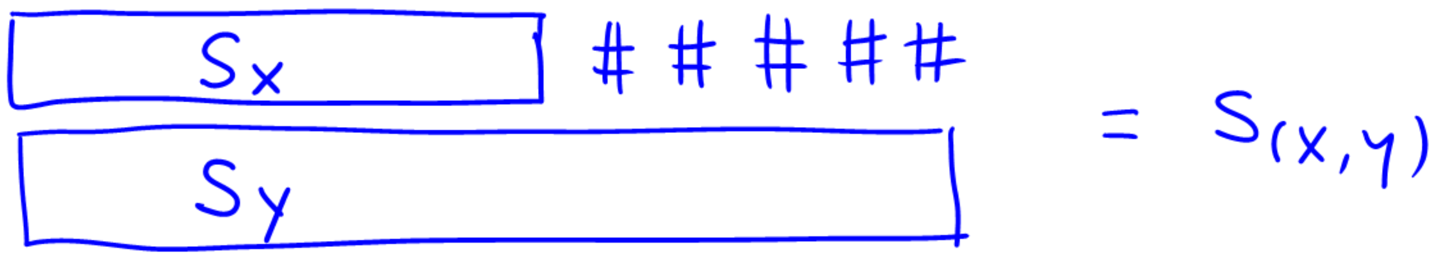
So we assume:

- 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 an object are all the words of s_x #*.

- A pair (x,y) of objects has a minimal encoding s_(x,y)



- The encodings of the pair (x,y) are 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!)

Definition 5.2 *Assume an encoding of X over Σ^* has been fixed. Let A be an NFA.*

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

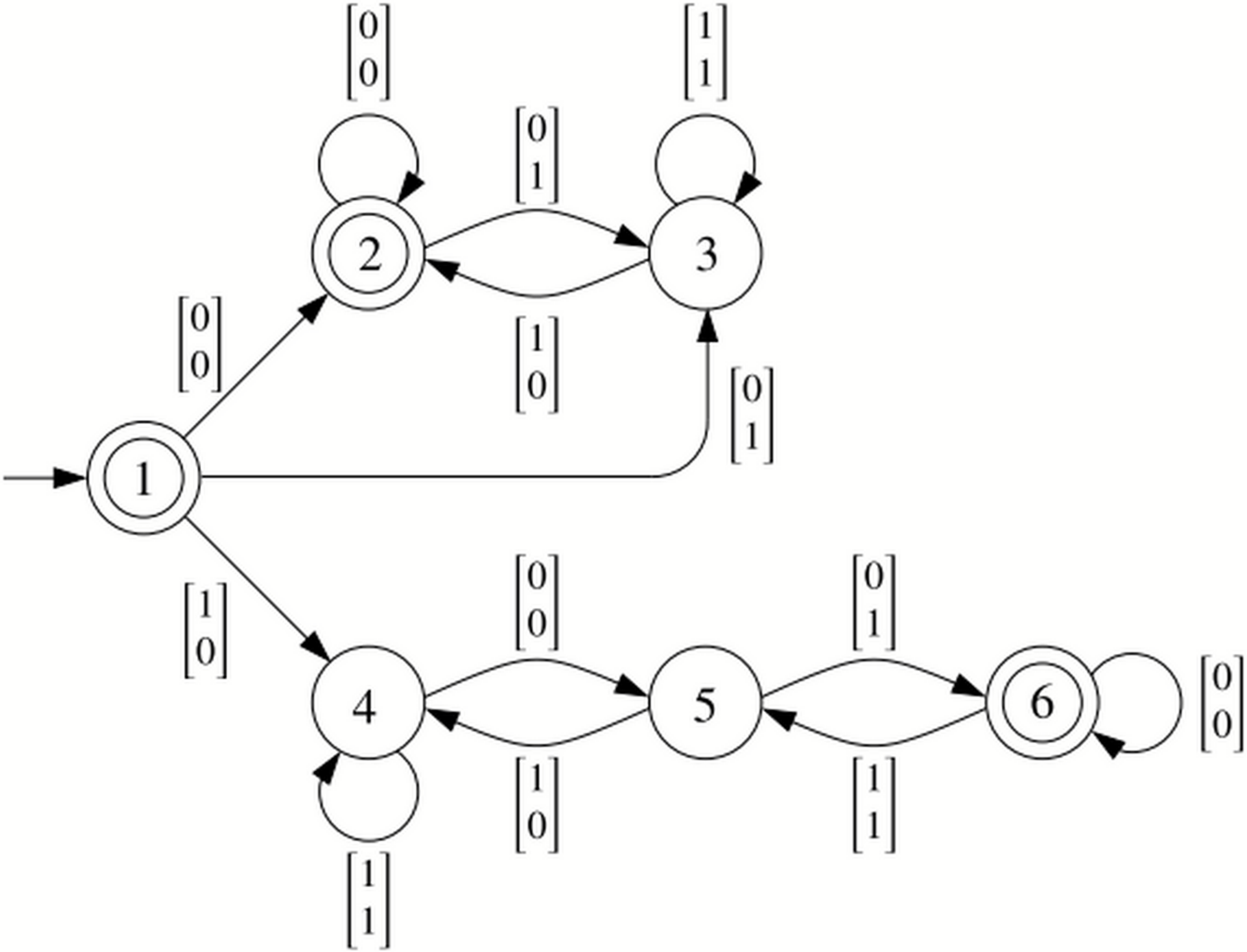
$$\mathcal{L}(A) = \{w \in \Sigma^* \mid w \text{ encodes some element of } Y\} .$$

A subset $Y \subseteq X$ is regular *(with respect to the fixed encoding) if it is recognized by some NFA.*

Notice that with this definition a NFA may neither accept nor reject a given x. In this case the NFA does not recognize any subset of X.

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?

Transducers



Definition 5.3 *A* transducer *over* Σ *is an NFA over the alphabet* $\Sigma \times \Sigma$.

Definition 5.4 *Let* T *be a transducer over* Σ. *Given words* $w_1 = a_1 a_2 \ldots a_n$ *and* $w_2 = b_1 b_2 \ldots b_n$, *we say that* T accepts the pair (w_1, w_2) *if it accepts the word* $(a_1, b_1) \ldots (a_n, b_n) \in (\Sigma \times \Sigma)^*$.

Definition 5.5 *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*

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

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 **N** }

- The relation { (n, 2n) | n in **N** }

Example 5.6 The *Collatz function* is the function $f : \mathbb{N} \rightarrow \mathbb{N}$ defined as follows:

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

Determinism

A transducer is deterministic if it is a DFA.
Observe: if A has size n, then a state of a deterministic transducer with alphabet A x A has n^2 outgoing transitions.

Warning! There is a different definition of determinism:

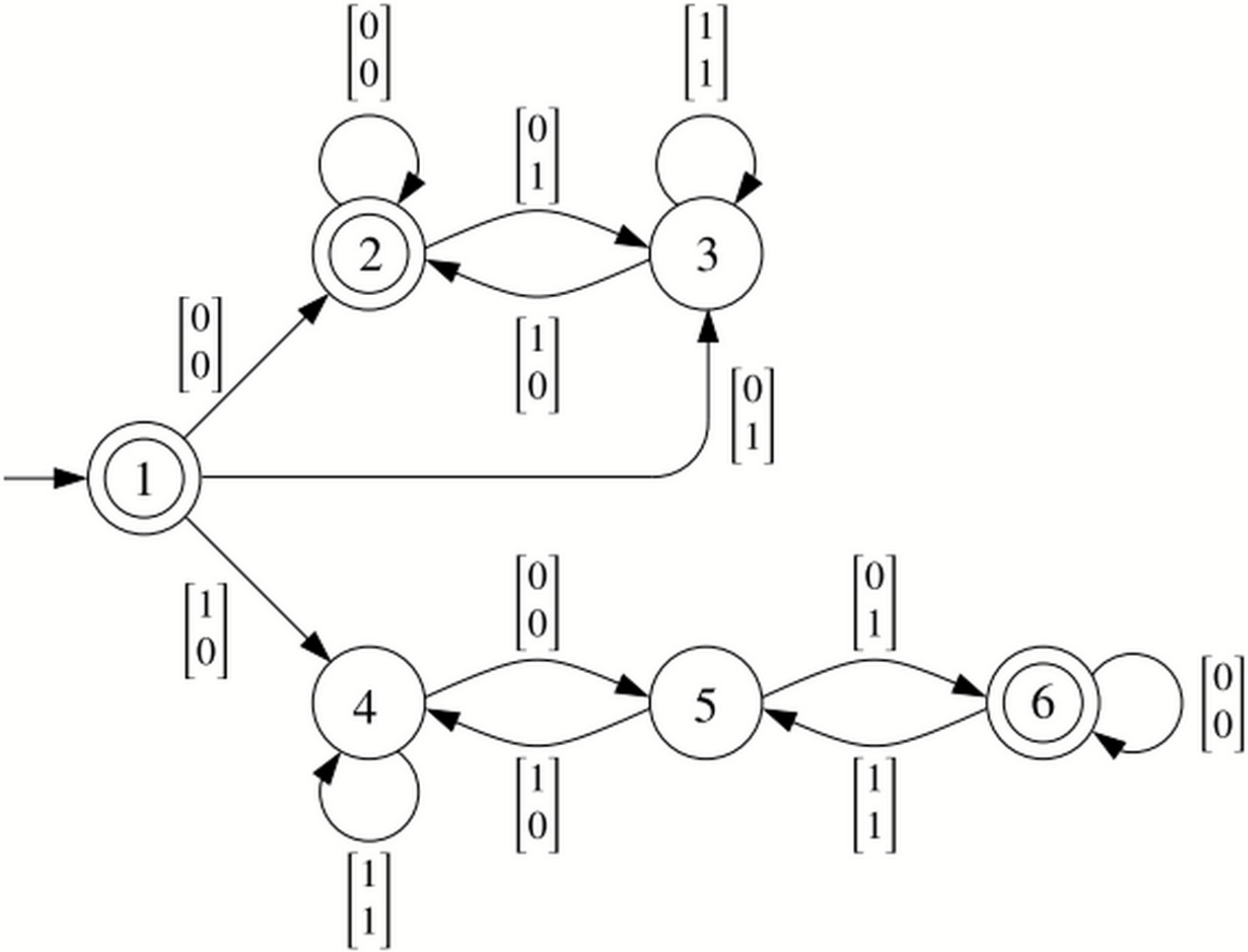
- pair (a,b) interpreted "output b on input a"
- deterministic: only one move (and so one possible output) for each input

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 componet is not correct

Counterexample: R={ (4,1) }

$S_{(4,1)} =$

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 $\mathcal{L}(A) = \pi_1(\mathcal{L}(T))$

$Q' \leftarrow Q$; $q'_0 \leftarrow q_0$; $F'' \leftarrow F$
$\delta' \leftarrow \emptyset$;
for all $(q, (a, b), q') \in \delta$ **do**
  add (q, a, q') **to** δ'
$F' \leftarrow PadClosure((Q', \Sigma, \delta', q'_0, F''), \#)$

PadClosure(A, #)

Input: NFA $A = (\Sigma \times \Sigma, Q, \delta, q_0, F)$

Output: new set F' of final states

$W \leftarrow F$; $F' \leftarrow \emptyset$;
while $W \neq \emptyset$ **do**
  pick q **from** W
  add q **to** F'
  for all $(q', \#, q) \in \delta$ **do**
    if $q' \notin F'$ **then add** q' **to** W
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!

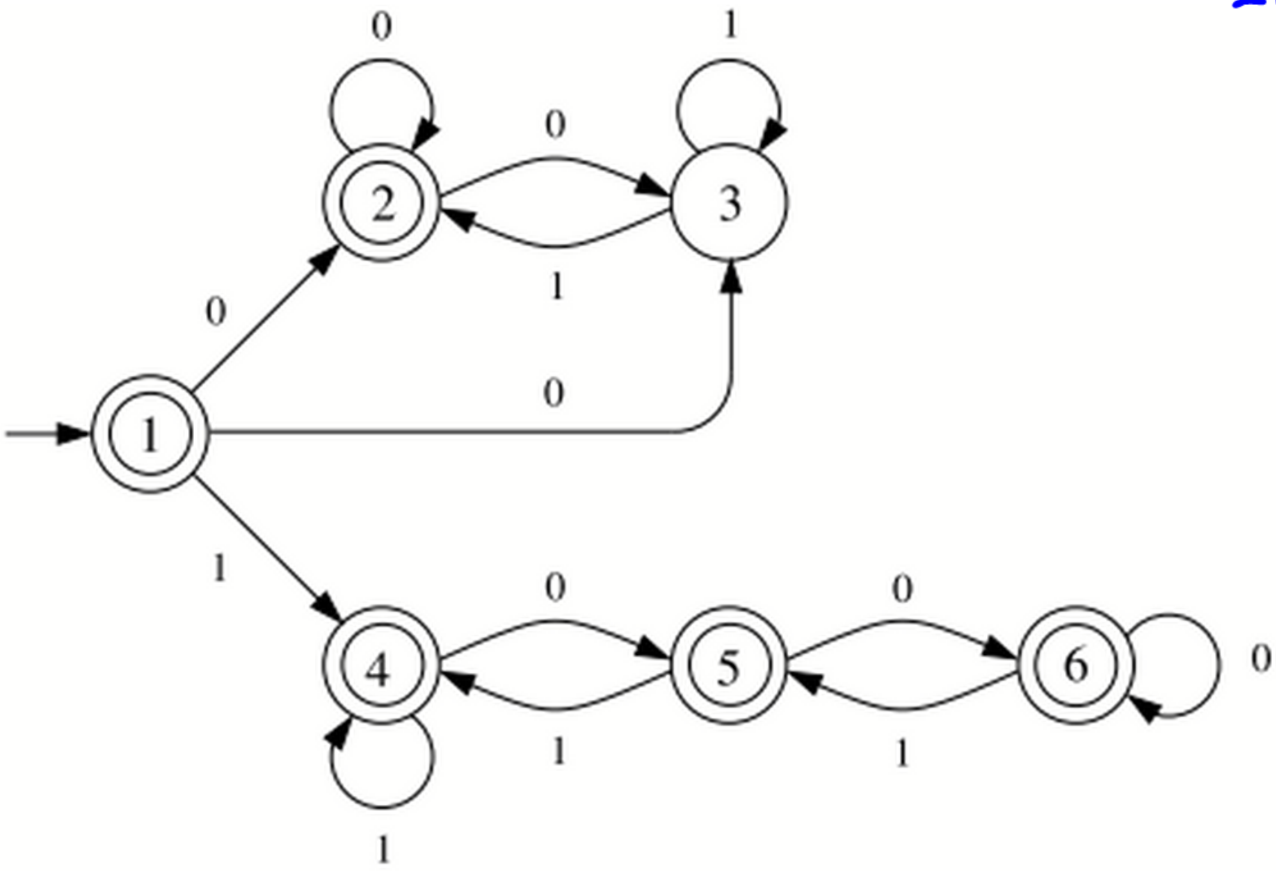
Solution: if after eliminating the second components a non-final state goes with #...# to a final state, we mark the state as final.

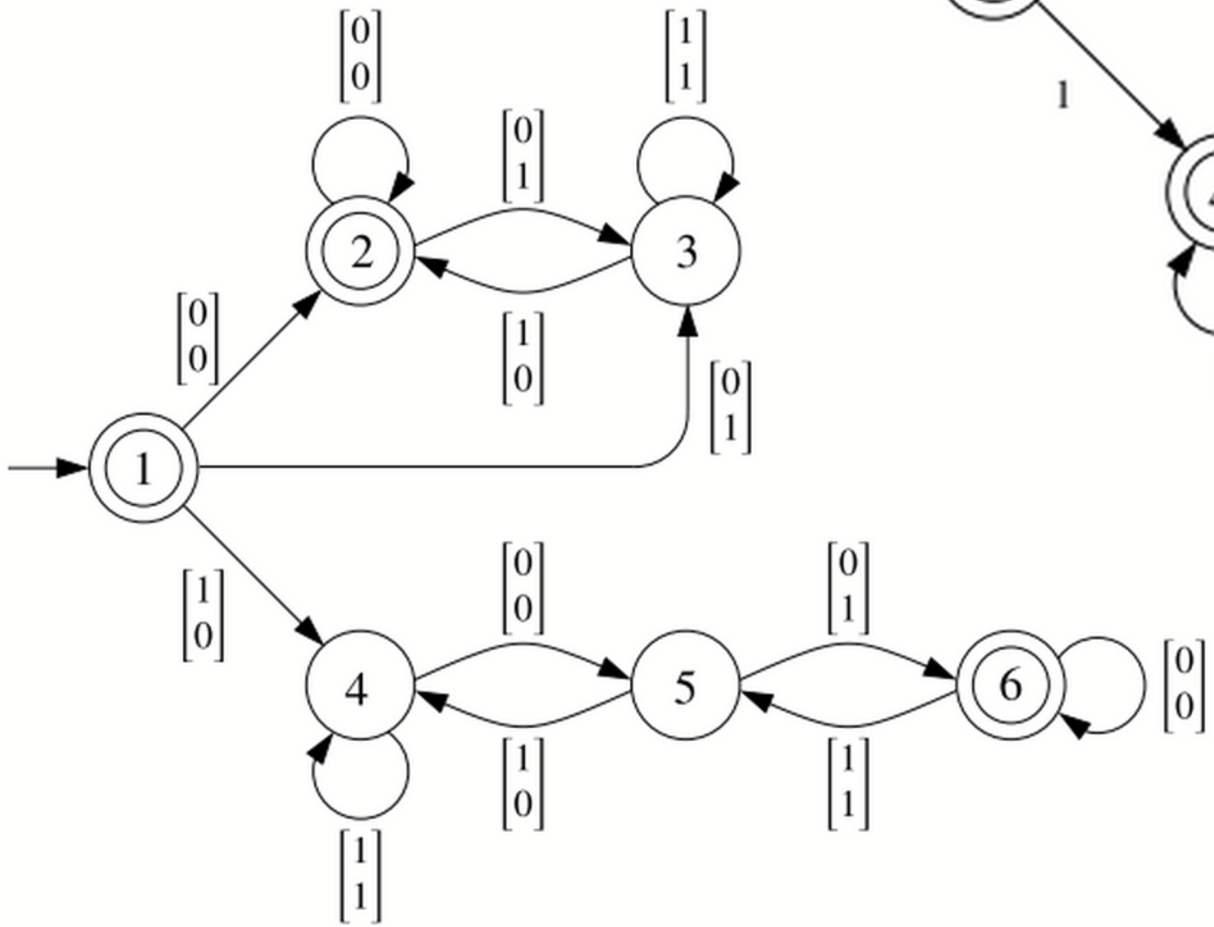
Complexity: linear in the size of the transducer

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

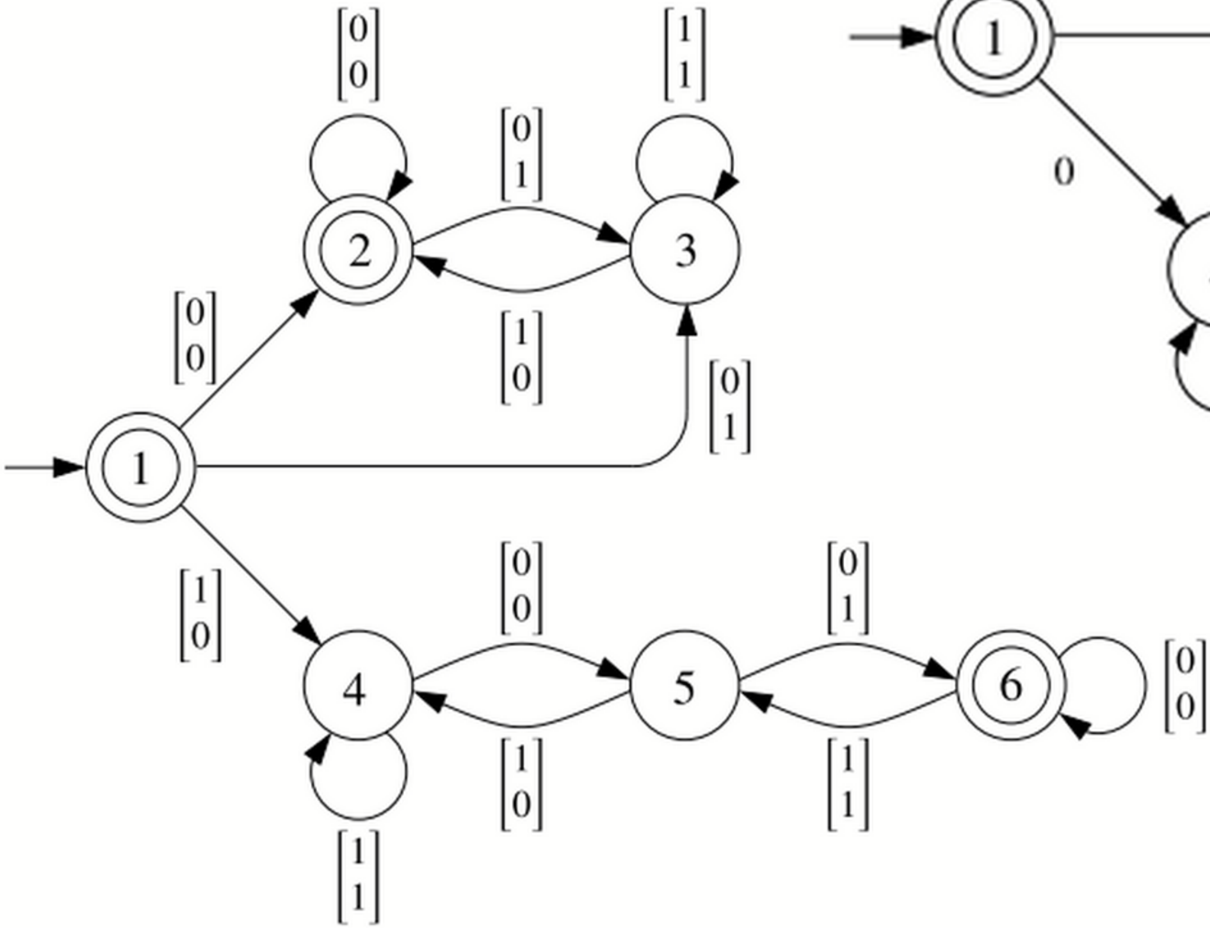
This is the operation that prevents us from implementing all operations directly on DFAs.

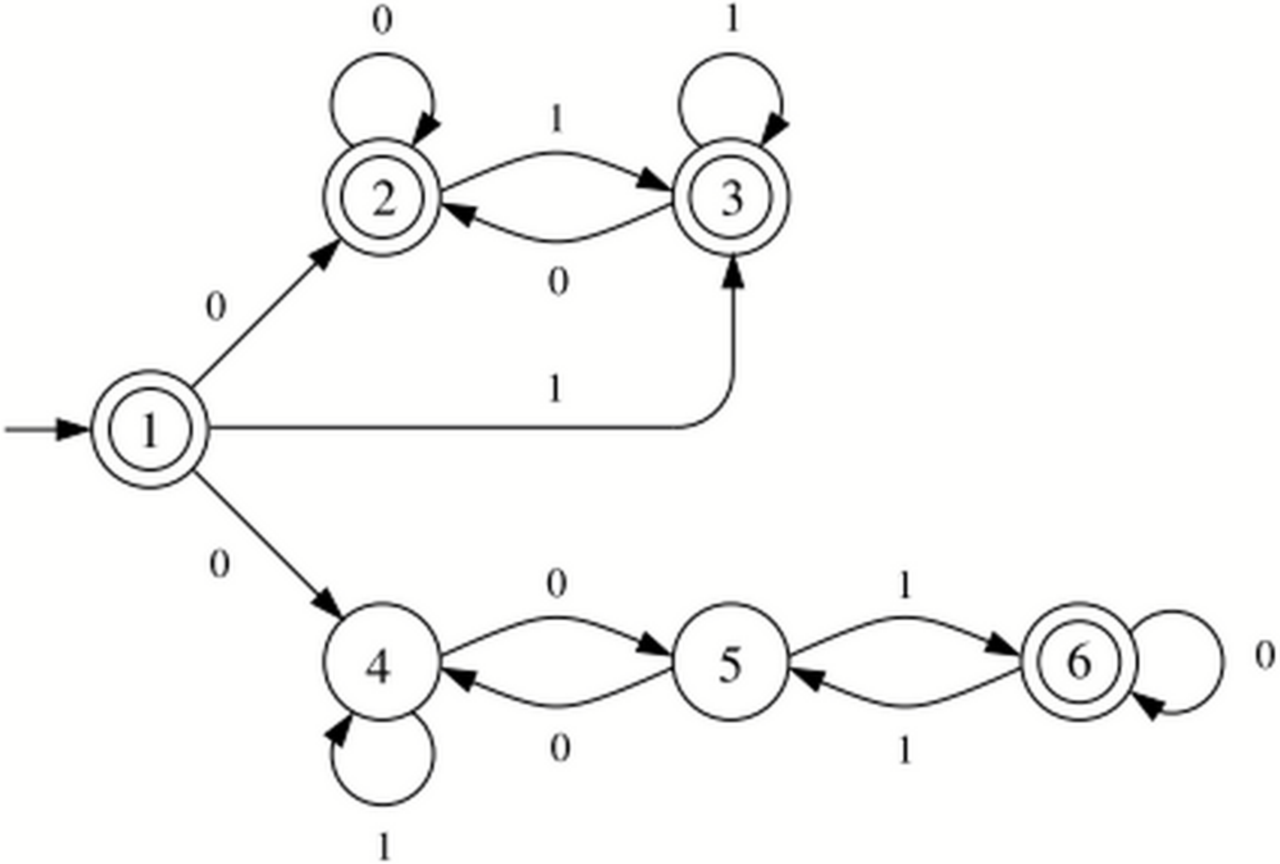
Projection_1





Projection_2





Correctness proof:

Assume: transducer T recognizes a set of pairs

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

Correctness proof:

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 (w,w') for some w'. By assumption T accepts (w,w')(#,#)*. So A accepts w#*.
Moreover, by padding closure, if w = s_x #^k for some k >0, and so 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 object x accepted by A , then there is an object y such that (x,y) accepted by T .

x accepted by A
=> (part (a))
s_x accepted by A
=>
(s_x, w) accepted by T for some w

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

Correctness proof:

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

(x,y) accepted by T

=>

(w_x, w_y) accepted by T for some encodings w_x , w_y of x and y

=>

w_x accepted by T

=> (part (a))

x accepted by A.

Remember:

The projection automaton of a deterministic transducer may be nondeterministic.

Join

Goal: given transducers T1, T2 recognizing relations R1, R2, construct a transducer T1 o T2 recognizing the relation R1 o R2.

First step: construct a transducer T that accepts (w, v) iff there is a "connecting" word u such that

- (w, u) is accepted by T1, and
- (u, v) is accepted by T2.

We slightly modify the pairing construction.

Instead of:

$$\begin{bmatrix} q_{01} \\ q_{02} \end{bmatrix} \xrightarrow{a_1} \begin{bmatrix} q_{11} \\ q_{12} \end{bmatrix}$$

iff

$$q_{01} \xrightarrow{a_1} q_{11}$$
$$q_{02} \xrightarrow{a_1} q_{12}$$

we now use

$$\begin{bmatrix} q_{01} \\ q_{02} \end{bmatrix} \xrightarrow{\begin{bmatrix} a_1 \\ b_1 \end{bmatrix}} \begin{bmatrix} q_{11} \\ q_{12} \end{bmatrix}$$

iff

$$q_{01} \xrightarrow{\begin{bmatrix} a_1 \\ c_1 \end{bmatrix}} q_{11}$$
$$q_{02} \xrightarrow{\begin{bmatrix} c_1 \\ b_1 \end{bmatrix}} q_{12}$$

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

$$q_{01} \xrightarrow{\begin{bmatrix} a_1 \\ c_1 \end{bmatrix}} q_{11} \xrightarrow{\begin{bmatrix} a_2 \\ c_2 \end{bmatrix}} q_{21} \quad \cdots \quad q_{(n-1)1} \xrightarrow{\begin{bmatrix} a_n \\ c_n \end{bmatrix}} q_{n1}$$

$$q_{02} \xrightarrow{\begin{bmatrix} c_1 \\ b_1 \end{bmatrix}} q_{12} \xrightarrow{\begin{bmatrix} c_2 \\ b_2 \end{bmatrix}} q_{22} \quad \cdots \quad q_{(n-1)2} \xrightarrow{\begin{bmatrix} c_n \\ b_n \end{bmatrix}} q_{n2}$$

We have the same problem as before:

Let R1 = { (2,4) } , R2 = { (4,2) }

Then R1 o R2 = { (2,2) }

But the operation we have just defined does not yield the correct result.

Solution: apply the padding closure again with padding symbol [#,#].

$Join(T_1, T_2)$

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 $T_1 \circ T_2 = (Q, \Sigma \times \Sigma, \delta, q_0, F)$

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

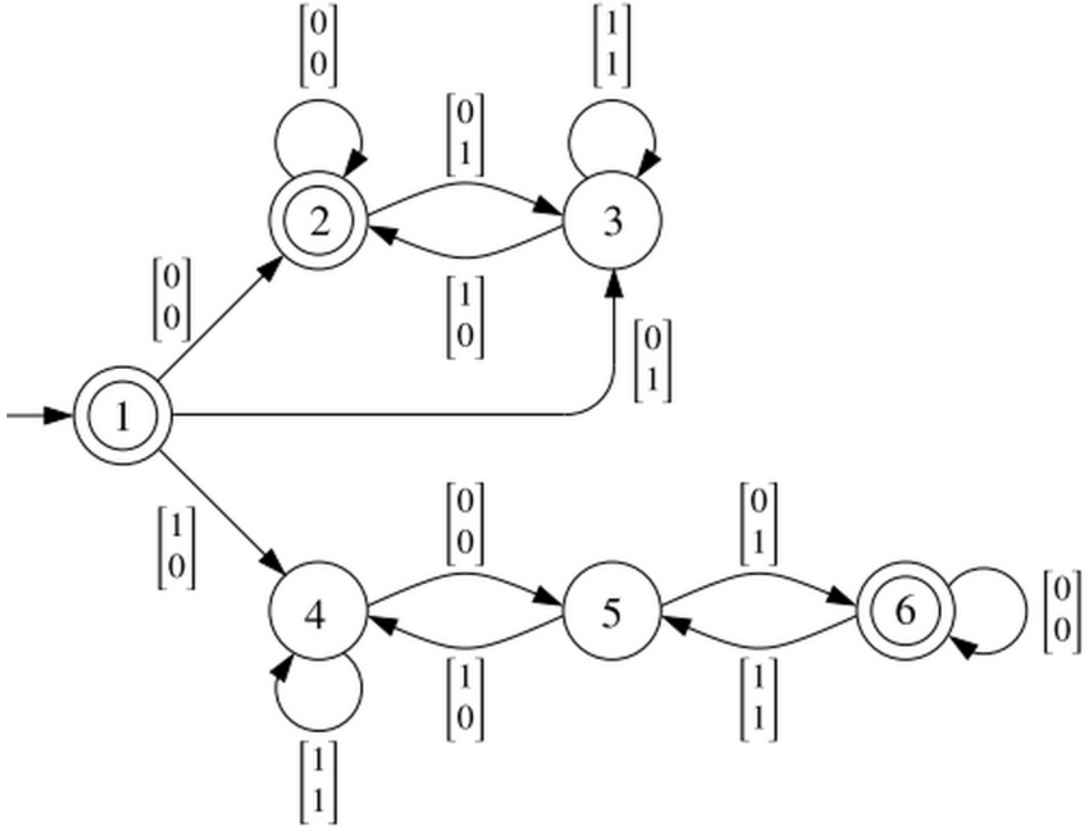
Complexity: similar to pairing

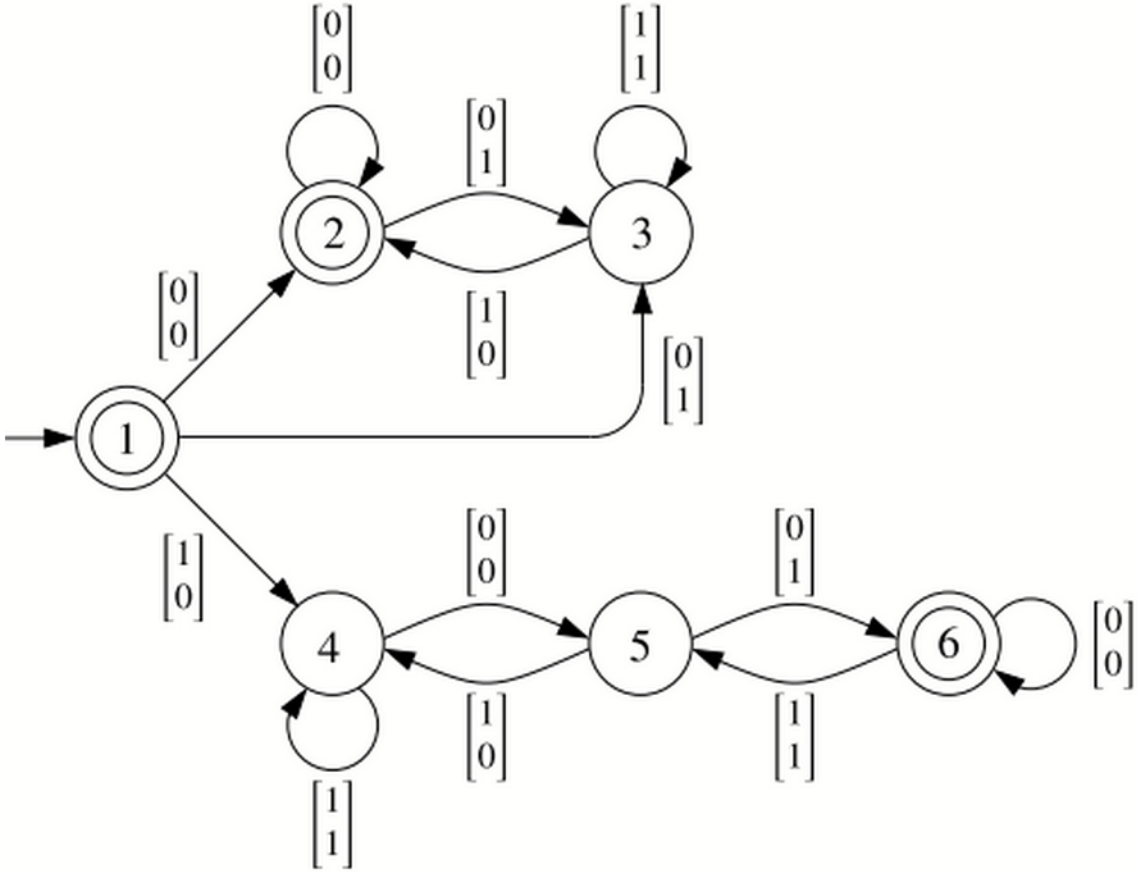
Example:

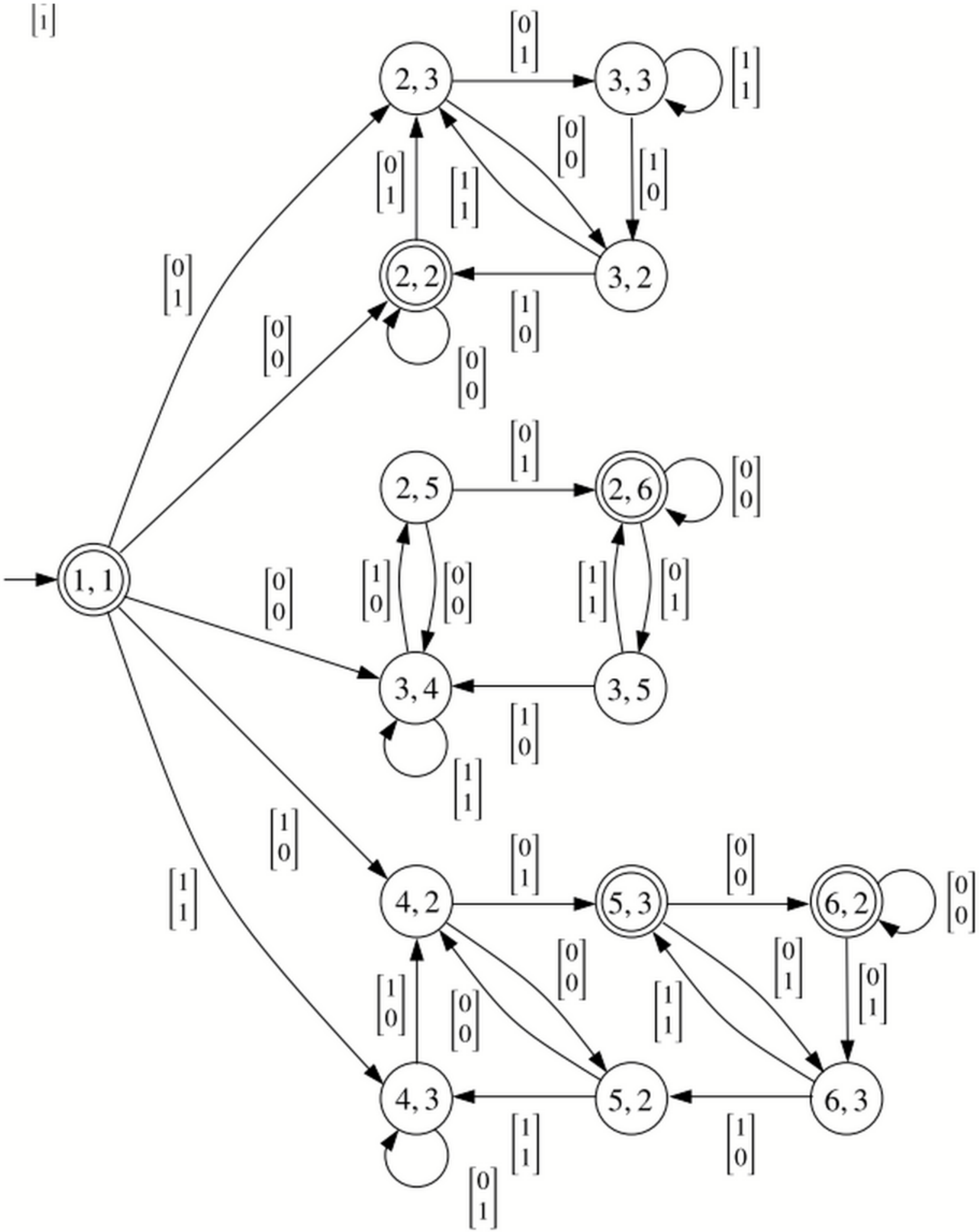
Let f be the Collatz function.
Let R1 = R2 = { (n, f(n)) | n >= 0 }

Then R1 o R2 = { (n, f(f(n))) | n >= 0 }

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


1
2
3
4
5
6


1
2
3
4
5
6


1, 1
2, 3
3, 3
2, 2
3, 2
2, 5
2, 6
3, 4
3, 5
4, 2
5, 3
6, 2
4, 3
5, 2
6, 3

Pre and Post

Goal (for post): given

automaton A recognizing set X and
transducer T recognizing relation R

construct automaton B recognizing the set

{ y | exists x in X such that (x, y) in R }

We slightly modify the construction for join.

Instead of:

$$\begin{bmatrix} q_{01} \\ q_{02} \end{bmatrix} \xrightarrow{\begin{bmatrix} a_1 \\ b_1 \end{bmatrix}} \begin{bmatrix} q_{11} \\ q_{12} \end{bmatrix}$$

iff

$$q_{01} \xrightarrow{\begin{bmatrix} a_1 \\ c_1 \end{bmatrix}} q_{11}$$

$$q_{02} \xrightarrow{\begin{bmatrix} c_1 \\ b_1 \end{bmatrix}} q_{12}$$

for some letter c1

we now use

$$\begin{bmatrix} q_{01} \\ q_{02} \end{bmatrix} \xrightarrow{b_1} \begin{bmatrix} q_{11} \\ q_{12} \end{bmatrix}$$

iff

$$q_{01} \xrightarrow{a_1} q_{11}$$

$$q_{02} \xrightarrow{\begin{bmatrix} a_1 \\ b_1 \end{bmatrix}} q_{12}$$

for some letter a_1

From Join to Post

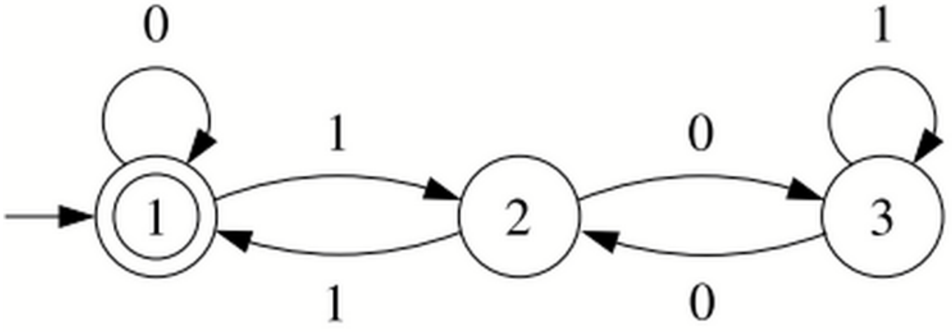
$Join(T_1, T_2)$

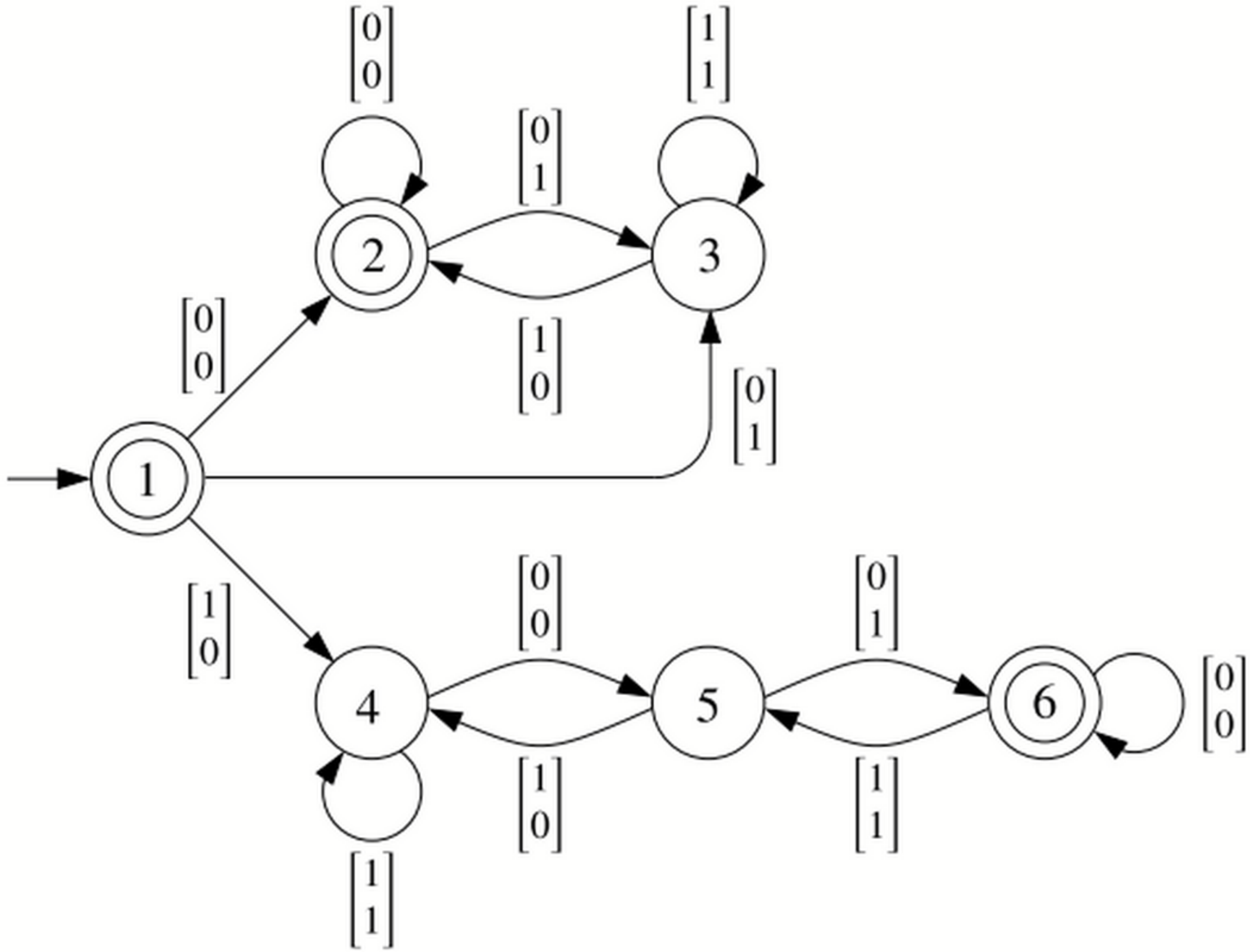
Input: transducers $T_1 = (Q_1, \Sigma \times \Sigma, \delta_1, q_{01}, F_1)$, $T_2 = (Q_2, \Sigma \times \Sigma, \delta_2, q_{02}, F_2)$

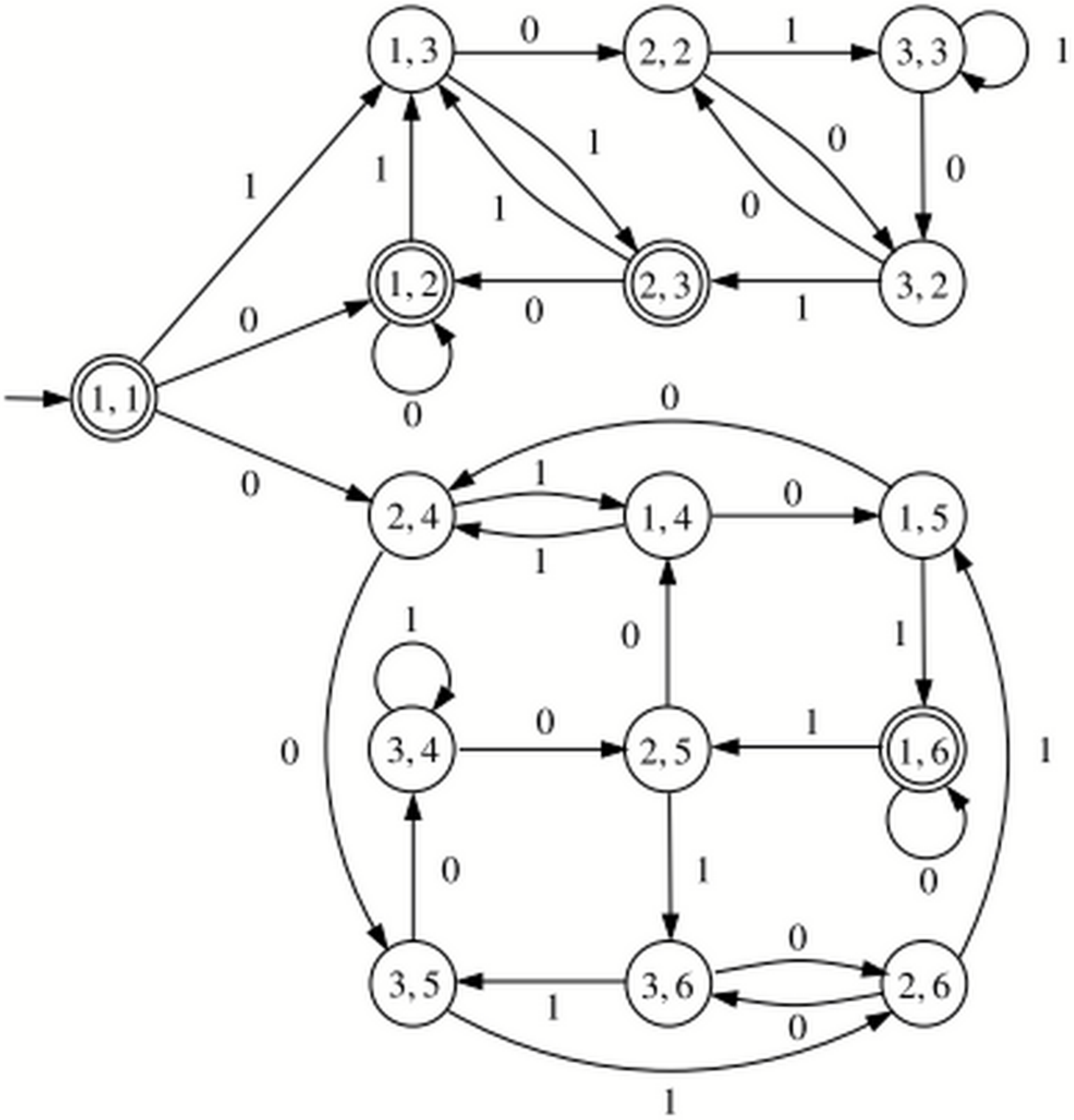
Output: transducer $T_1 \circ T_2 = (Q, \Sigma \times \Sigma, \delta, q_0, F)$

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

Example: compute the set { f(n) | n multiple of 3 }






1, 3
2, 2
3, 3
1, 2
2, 3
3, 2
1, 1
2, 4
1, 4
1, 5
3, 4
2, 5
1, 6
3, 5
3, 6
2, 6