

Operations on relations

Operations on relations

Universe of objects U , relations R, S on objects, set of objects X

Operations on relations

- Projection_1**(R) : returns the set $\pi_1(R) = \{x \mid \exists y (x, y) \in R\}$.
- Projection_2**(R) : returns the set $\pi_2(R) = \{y \mid \exists x (x, y) \in R\}$.
- Join**(R, S) : returns the relation $R \circ S = \{(x, z) \mid \exists y \in X (x, y) \in R \wedge (y, z) \in S\}$
- Post**(X, R) : returns the set $post_R(X) = \{y \in U \mid \exists x \in X (x, y) \in R\}$.
- Pre**(X, R) : returns the set $pre_R(X) = \{y \in U \mid \exists x \in X (y, x) \in R\}$.

Encoding pairs

- Using automata to represent relations requires to encode **pairs** of objects.
- How should we encode a pair (n_1, n_2) of natural numbers?

Encoding pairs

- Assume n_1, n_2 are encoded by w_1, w_2 in the *lsbf* encoding
- Which should be the encoding of (n_1, n_2) ?
- Cannot be $w_1 w_2$, then the same word encodes different pairs
- **First attempt**: use a separator symbol $\&$, and encode (n_1, n_2) by $w_1 \& w_2$.
 - **Problem**: not even the identity relation is encoded as a regular language!

Encoding pairs

- **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.

Example: the encoding of $(10, 35)$ is $\begin{bmatrix} 0 \\ 1 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \end{bmatrix}$

- We accept that 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**.

Encoding pairs

- 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 $s_{(x,y)}\#^*$.

Redefining acceptance

- **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, because now there are multiple encodings)

- **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?

Redefining acceptance

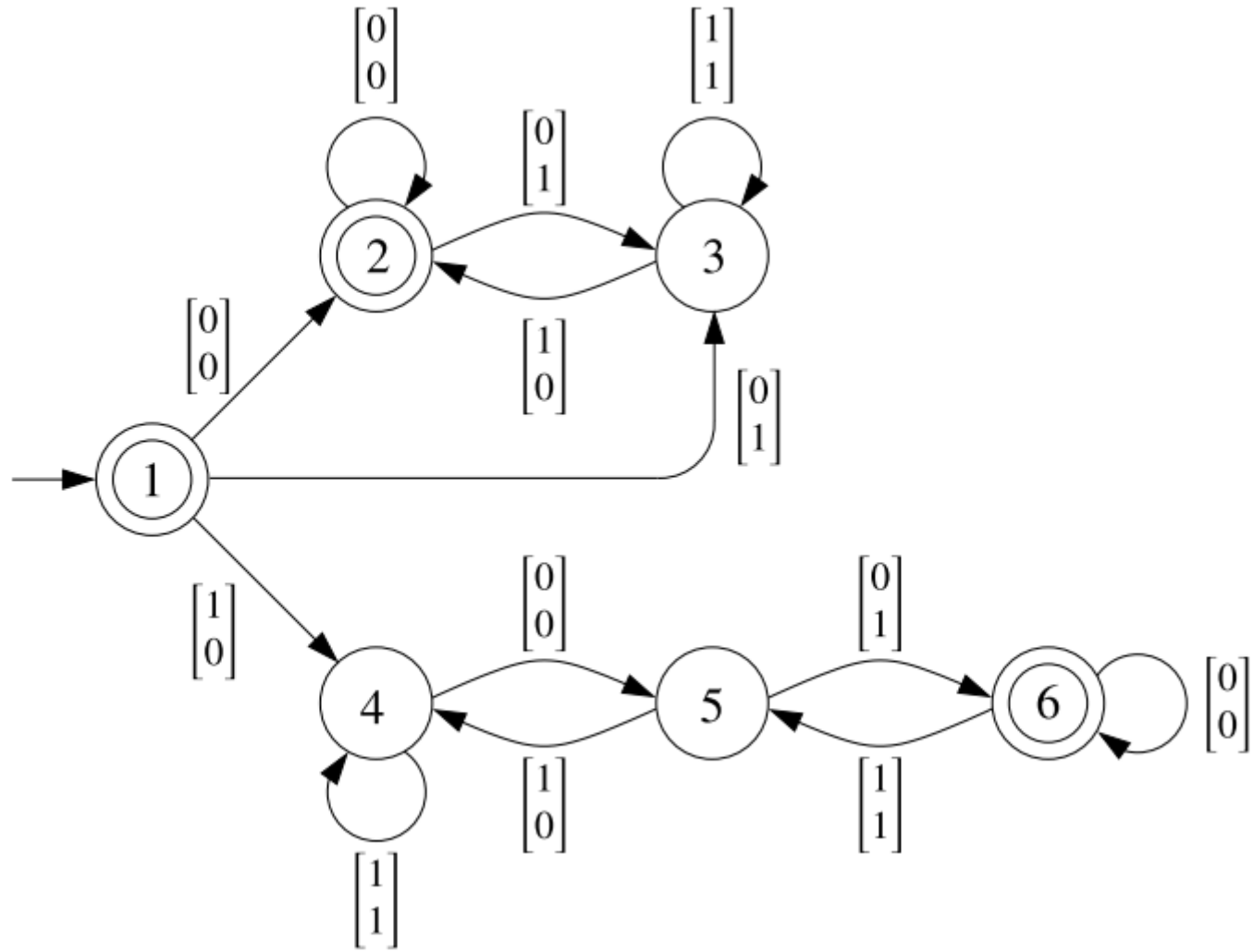
- **Definition:** Assume an encoding of objects as words has been fixed. We say
 - An automaton **accepts** an object x if it accepts **all** encodings of x .
 - An automaton **rejects** an object x if it accepts **no** encoding of x .
 - An automaton recognizes a set of objects X if it accepts every object of X and rejects every other object.
- Observe: if an automaton accepts some, but not all the encodings of an object, then the automaton does not recognize any set. We say that such an automaton is **ill formed**. Automata that do recognize some set of objects are **well formed**.

Redefining acceptance

- The operations we have defined so far still work, in the following sense:
 - If the input(s) is (are) well formed, then the output is well formed
 - The output still satisfies the specification.
- **Example:** If A_1, A_2 are well formed NFAs recognizing sets of objects X_1, X_2 then the automaton $A := \text{inter}(A_1, A_2)$ is well formed and recognizes $X_1 \cap X_2$.

Proof of well formedness: If A recognizes an encoding w of an object x , then by definition of A both A_1 and A_2 recognize w . Since A_1 and A_2 are well formed they recognize all encodings of x , and so A also recognizes all encodings of x .

Transducers



Transducers

- A **transducer over Σ** is an NFA over the alphabet $\Sigma \times \Sigma$.
- We write $(a, b) \in \Sigma \times \Sigma$ as $\begin{bmatrix} a \\ b \end{bmatrix}$
- A transducer **accepts a pair $(a_1 \dots a_n, b_1 \dots, b_n)$ of words** if

it accepts the word $\begin{bmatrix} a_1 \\ b_1 \end{bmatrix} \cdots \begin{bmatrix} a_n \\ b_n \end{bmatrix}$.

- A transducer **accepts a pair of objects** if it accepts all its encodings (which are pairs of words).
- A relation is **regular** if it is recognized by some transducer.

Examples of regular relations

- Examples of regular relations on numbers (*lsbf* encoding):
 - The identity relation $\{ (n, n) \mid n \in \mathbb{N} \}$
 - The relation $\{ (n, 2n) \mid n \in \mathbb{N} \}$
 - The relation $\{ (n, f(n)) \mid n \in \mathbb{N} \}$ where $f: \mathbb{N} \rightarrow \mathbb{N}$ is the Collatz function given by:

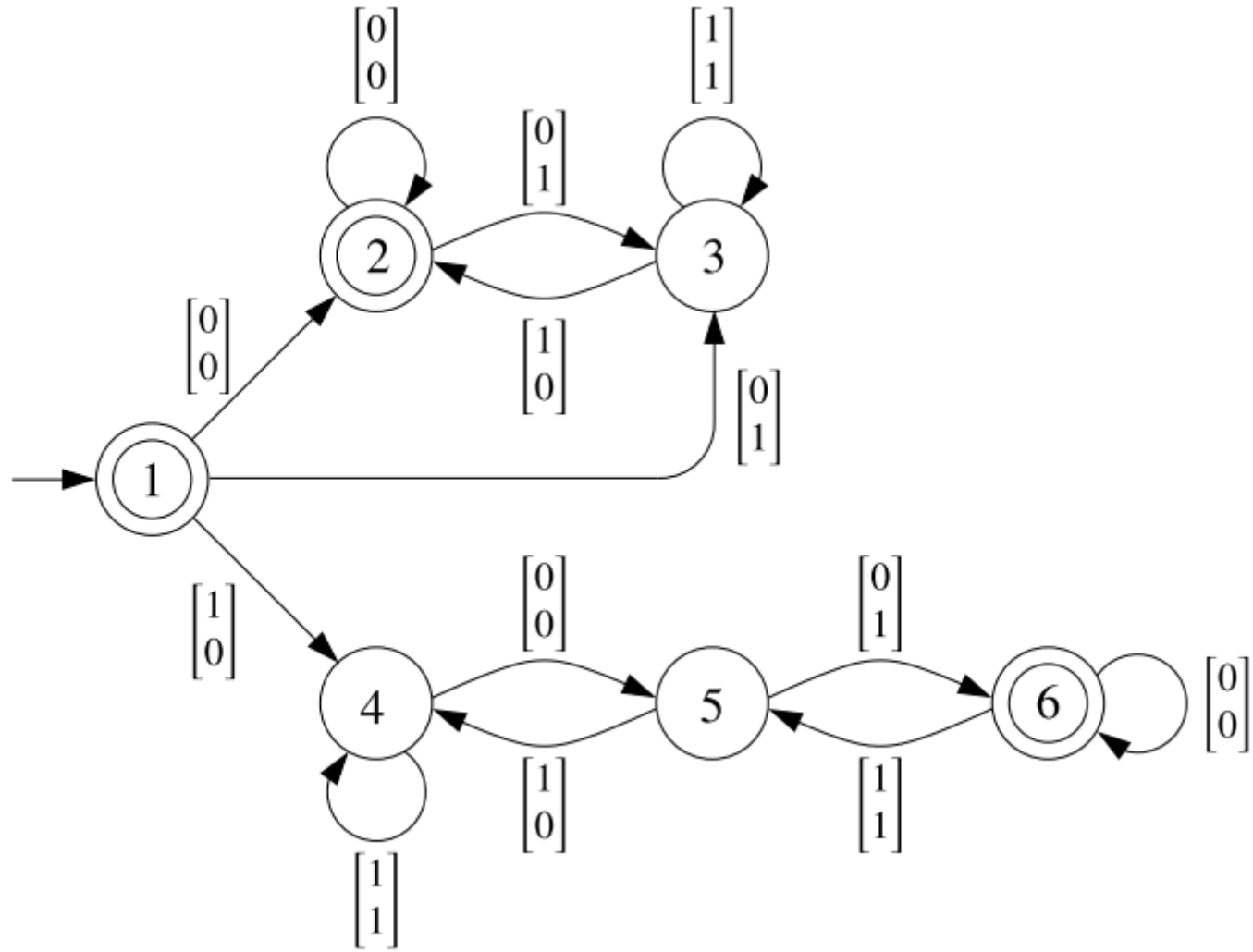
$$f(n) = \begin{cases} 3n + 1 & \text{if } n \text{ is odd} \\ n/2 & \text{if } n \text{ is even} \end{cases}$$

Deterministic transducers

- A transducer is **deterministic** if it is a DFA.
- **Observe:** if Σ has size n , then a state of a deterministic transducer with alphabet $\Sigma \times \Sigma$ 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.

Implementing the operations

Computing projections



Computing projections

- Deleting the second component is **incorrect**
 - 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 :

Computing projections

- **Problem:** we may be accepting $s_x \#^k \#^*$ instead of $s_x \#^*$ and so according to the definition we are not accepting x !
- **Solution:** if after eliminating the second components some non-final state goes with $\# \dots \#$ 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.

Correctness

- **Assume:** transducer T recognizes a relation
- **Prove:** the projection automaton A recognizes a set, and this set is the projection onto the first component of the relation 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 $j < k$.

Correctness

- 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.
- c) If A accepts an object x , then T accepts (x, y) for some y .

x is accepted by A
 $\Rightarrow s_x$ is accepted by A (part a)
 $\Rightarrow \begin{matrix} s_x \\ w \end{matrix}$ is 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

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

(x, y) is accepted by T

\Rightarrow w_x
 w_y is accepted by T for some

encodings w_x, w_y of x and y

\Rightarrow w_x is accepted by A

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

Computing joins

- **Goal:** given transducers T_1, T_2 recognizing relations R_1, R_2 , construct a transducer $T_1 \circ T_2$ recognizing 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 T_2 .
- We slightly modify the pairing construction.

Computing joins

Pairing construction

$$\begin{bmatrix} q_1 \\ q_2 \end{bmatrix} \xrightarrow{a} \begin{bmatrix} q'_1 \\ q'_2 \end{bmatrix} \quad \text{iff} \quad \begin{array}{l} q_1 \xrightarrow{a} q'_1 \\ q_2 \xrightarrow{a} q'_2 \end{array}$$

Join construction

$$\begin{bmatrix} q_1 \\ q_2 \end{bmatrix} \xrightarrow{\begin{bmatrix} a \\ b \end{bmatrix}} \begin{bmatrix} q'_1 \\ q'_2 \end{bmatrix} \quad \text{iff} \quad \begin{array}{l} q_1 \xrightarrow{\begin{bmatrix} a \\ c \end{bmatrix}} q'_1 \\ q_2 \xrightarrow{\begin{bmatrix} c \\ b \end{bmatrix}} q'_2 \end{array}$$

for some $c \in \Sigma$

Computing joins

- With the join construction, transducer T has a run

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

iff T_1 and T_2 have runs

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

Computing joins

- **Second step:** We have the same problem as before.
 - Let $R_1 = \{ (2,4) \}$, $R_2 = \{ (4,2) \}$.
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 $\begin{bmatrix} \# \\ \# \end{bmatrix}$.

Computing joins

Join(T_1, T_2)

Input: transducers $T_1 = (Q_1, \Sigma \times \Sigma, \delta_1, Q_{01}, F_1)$,

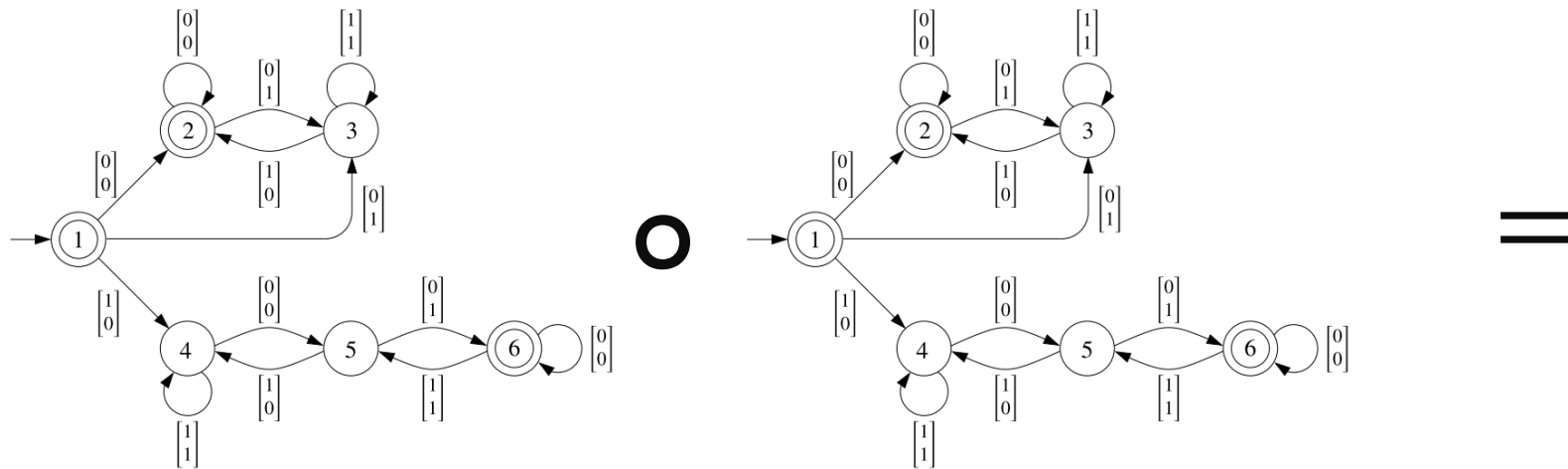
$T_2 = (Q_2, \Sigma \times \Sigma, \delta_2, Q_{02}, F_2)$

Output: transducer $T_1 \circ T_2 = (Q, \Sigma \times \Sigma, \delta, Q_0, F)$

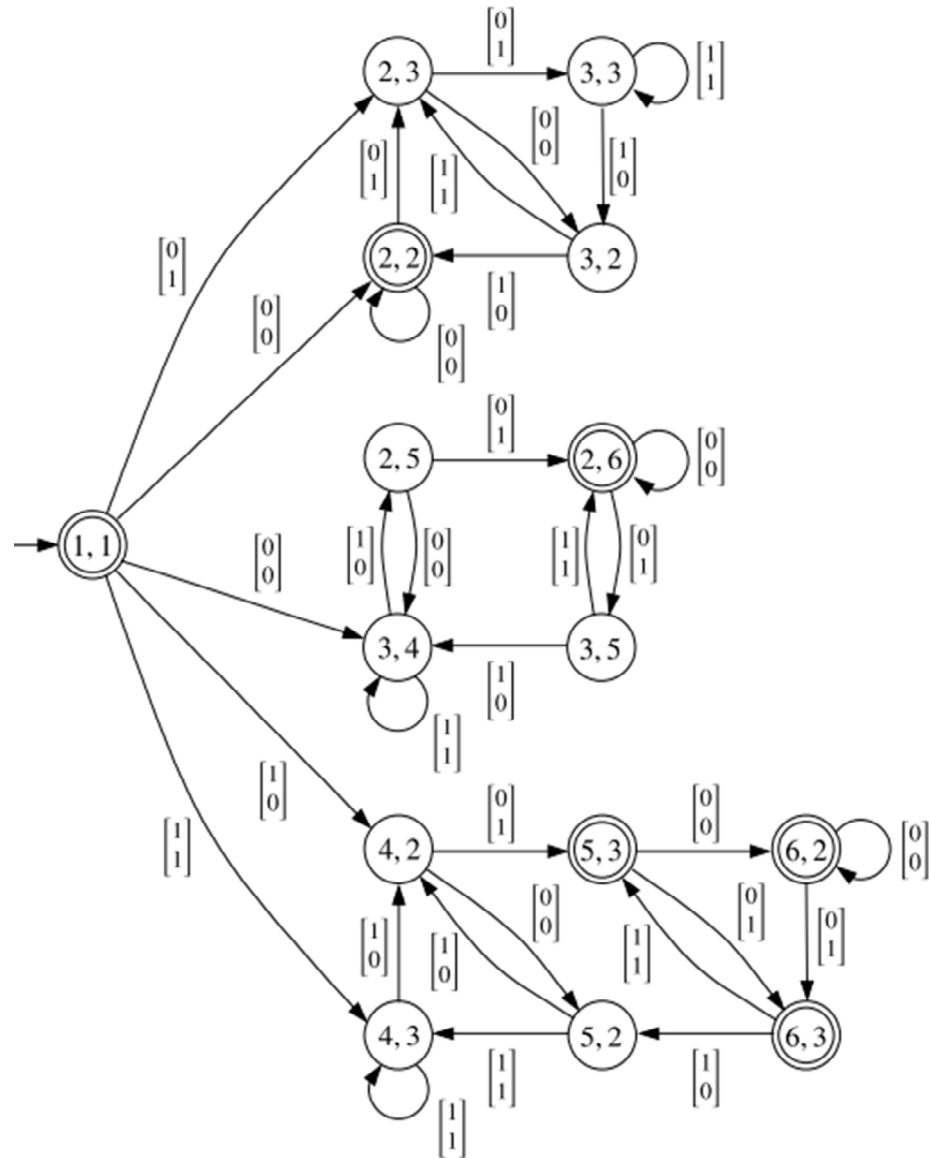
- 1 $Q, \delta, F' \leftarrow \emptyset; Q_0 \leftarrow Q_{01} \times Q_{02}$
- 2 $W \leftarrow Q_0$
- 3 **while** $W \neq \emptyset$ **do**
- 4 **pick** $[q_1, q_2]$ **from** W
- 5 **add** $[q_1, q_2]$ **to** Q
- 6 **if** $q_1 \in F_1$ and $q_2 \in F_2$ **then add** $[q_1, q_2]$ **to** F'
- 7 **for 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 $F \leftarrow \mathbf{PadClosure}((Q, \Sigma \times \Sigma, \delta, Q_0, F'), (\#, \#))$

Computing joins

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



Computing joins



Computing Pre and Post

- **Goal** (for Post): given
 - an automaton A recognizing a set X , and
 - a transducer T recognizing a relation R

construct an automaton B recognizing the set

$$\text{Post}(X, R) = \{ y \mid \exists x \in X : (x, y) \in R \}$$

We slightly modify the construction for join.

Computing Pre and Post

Join construction

$$\begin{bmatrix} q_1 \\ q_2 \end{bmatrix} \xrightarrow{\begin{bmatrix} a \\ b \end{bmatrix}} \begin{bmatrix} q'_1 \\ q'_2 \end{bmatrix} \quad \text{iff} \quad \begin{array}{l} q_1 \xrightarrow{\begin{bmatrix} a \\ c \end{bmatrix}} q'_1 \\ q_2 \xrightarrow{\begin{bmatrix} c \\ b \end{bmatrix}} q'_2 \end{array}$$

for some $c \in \Sigma$

Post construction

$$\begin{bmatrix} q_1 \\ q_2 \end{bmatrix} \xrightarrow{b} \begin{bmatrix} q'_1 \\ q'_2 \end{bmatrix} \quad \text{iff} \quad \begin{array}{l} q_1 \xrightarrow{a} q'_1 \\ q_2 \xrightarrow{\begin{bmatrix} a \\ b \end{bmatrix}} q'_2 \end{array}$$

for some $a \in \Sigma$

Computing Pre and Post

Join(T_1, T_2)

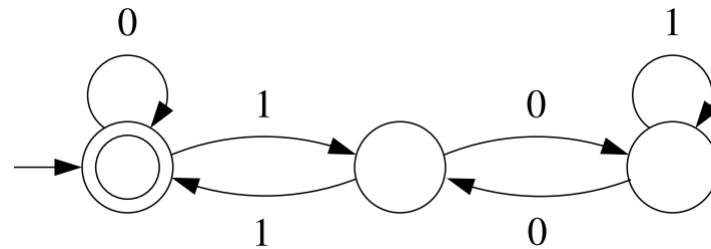
Input: transducers $T_1 = (Q_1, \Sigma \times \Sigma, \delta_1, Q_{01}, F_1)$,
 $T_2 = (Q_2, \Sigma \times \Sigma, \delta_2, Q_{02}, F_2)$

Output: transducer $T_1 \circ T_2 = (Q, \Sigma \times \Sigma, \delta, Q_0, F)$

```
1   $Q, \delta, F' \leftarrow \emptyset$ ;  $Q_0 \leftarrow Q_{01} \times Q_{02}$ 
2   $W \leftarrow Q_0$ 
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  $\delta$ 
9      if  $[q'_1, q'_2] \notin Q$  then add  $[q'_1, q'_2]$  to  $W$ 
10    $F \leftarrow \text{PadClosure}((Q, \Sigma \times \Sigma, \delta, Q_0, F'), (\#, \#))$ 
```

Computing Pre and Post

- Example.
 - Let f be the Collatz function.
 - We compute the set $\{f(n) \mid n \text{ is a multiple of } 3\}$
- DFA for the multiples of 3 in *lsfb* encoding



Post() =

Computing Pre and Post

