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

Encodings

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).

Encodings

We have argued 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

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).

- 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.

But then: a number is no longer encoded by only one word.

We call 0 the padding symbol or letter.

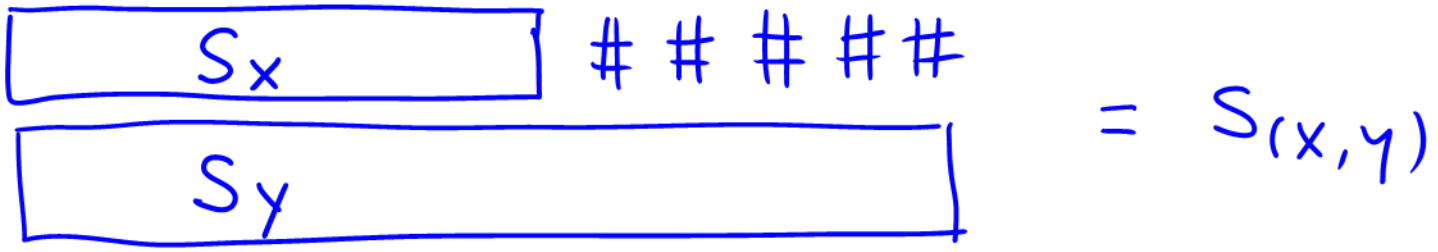
So we assume:

- The alphabet contains a padding letter #, which may or may not be different 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*

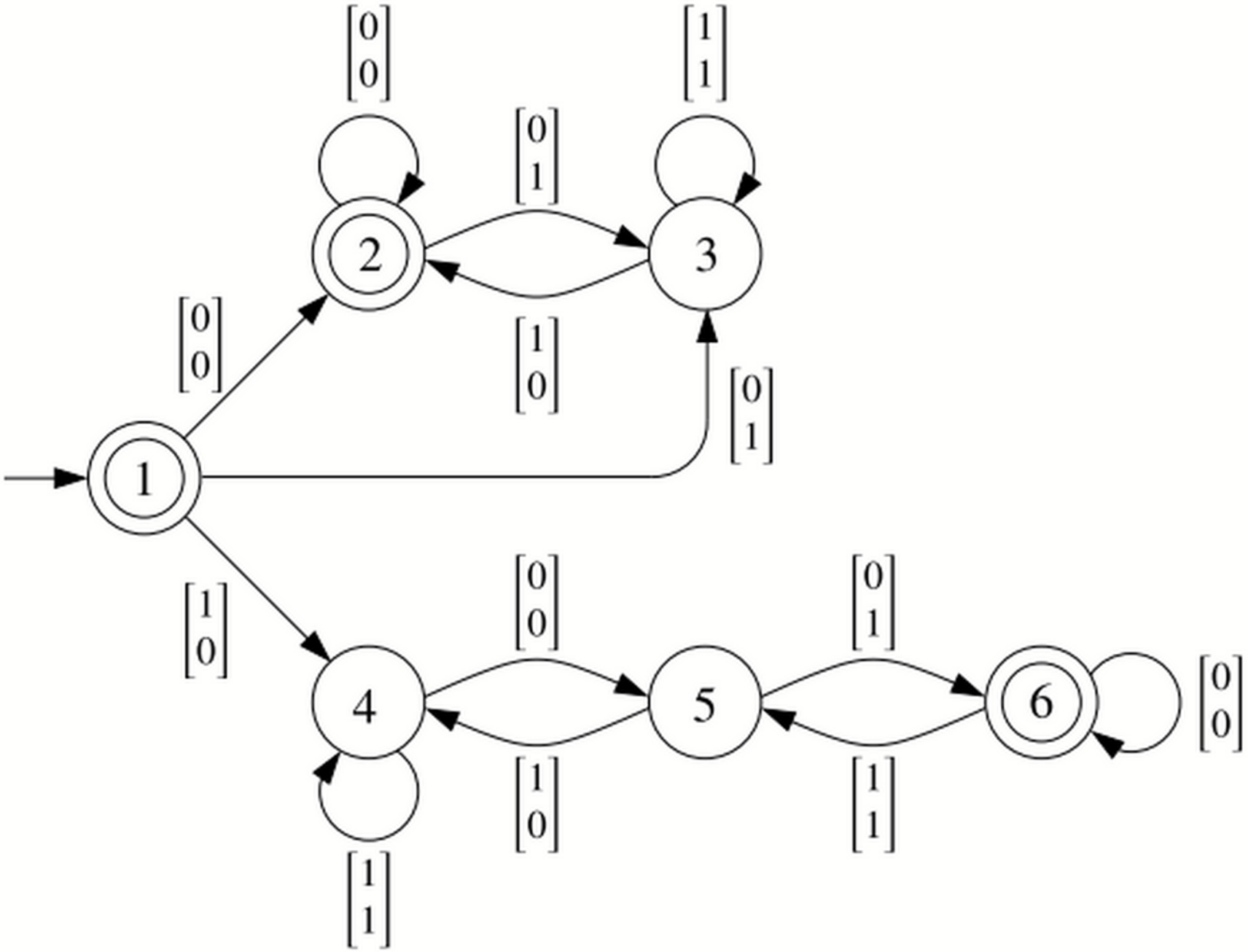
$$\mathcal{L}(A) = \{w \in \Sigma^* \mid \textit{ } w \textit{ encodes some element of } Y\} \ .$$

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* $(a_1, b_1)...(a_n, b_n) \in (\Sigma \times \Sigma)^*$.

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 $f\colon \mathbb{N} \to \mathbb{N}$ defined as follows:

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

Determinism A transducer is *deterministic* if it is a DFA. In particular, a state of a deterministic transducer over the alphabet $\Sigma \times \Sigma$ has exactly $|\Sigma|^2$ outgoing transitions. The transducer of Figure 5.1 is deterministic in this sense, when an appropriate sink state is added.

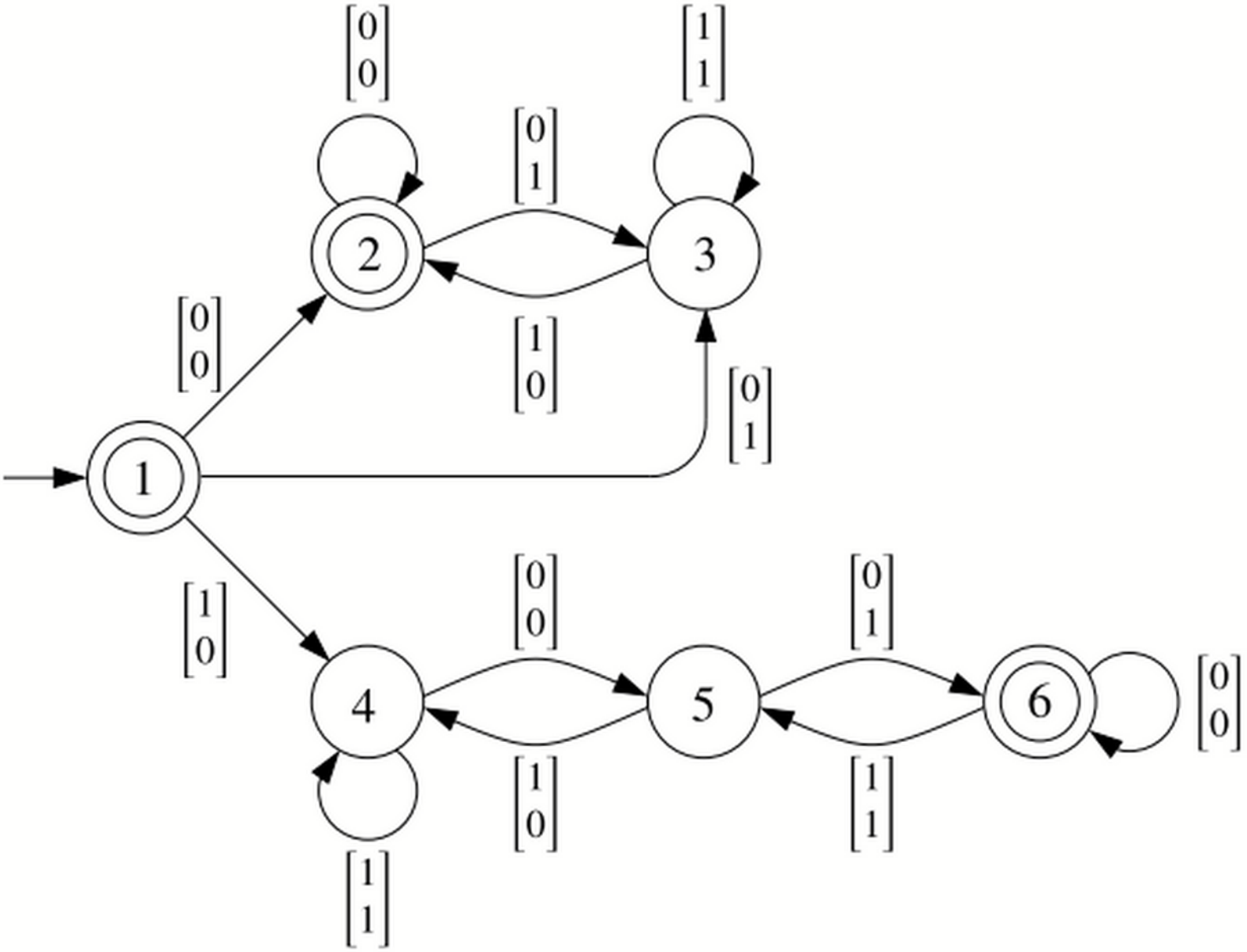
There is another possibility to define determinism of transducers, in which the letter (a, b) is interpreted as "the transducer receives the input a and produces the output b". In this view, a transducer is called deterministic if for every state q and every letter a there is exactly one transition of the form $(q, (a, b), q')$. Observe that these two definitions of determinism are *not* equivalent.

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 :

Problem: we may be accepting $s_{(x,y)}\ (\#,\#)^k\ (\#,\#)^*$
instead of $s_{(x,y)}\ (\#,\#)^*$

Solution: if a state goes with (#,#) to a final state, mark it also as final.