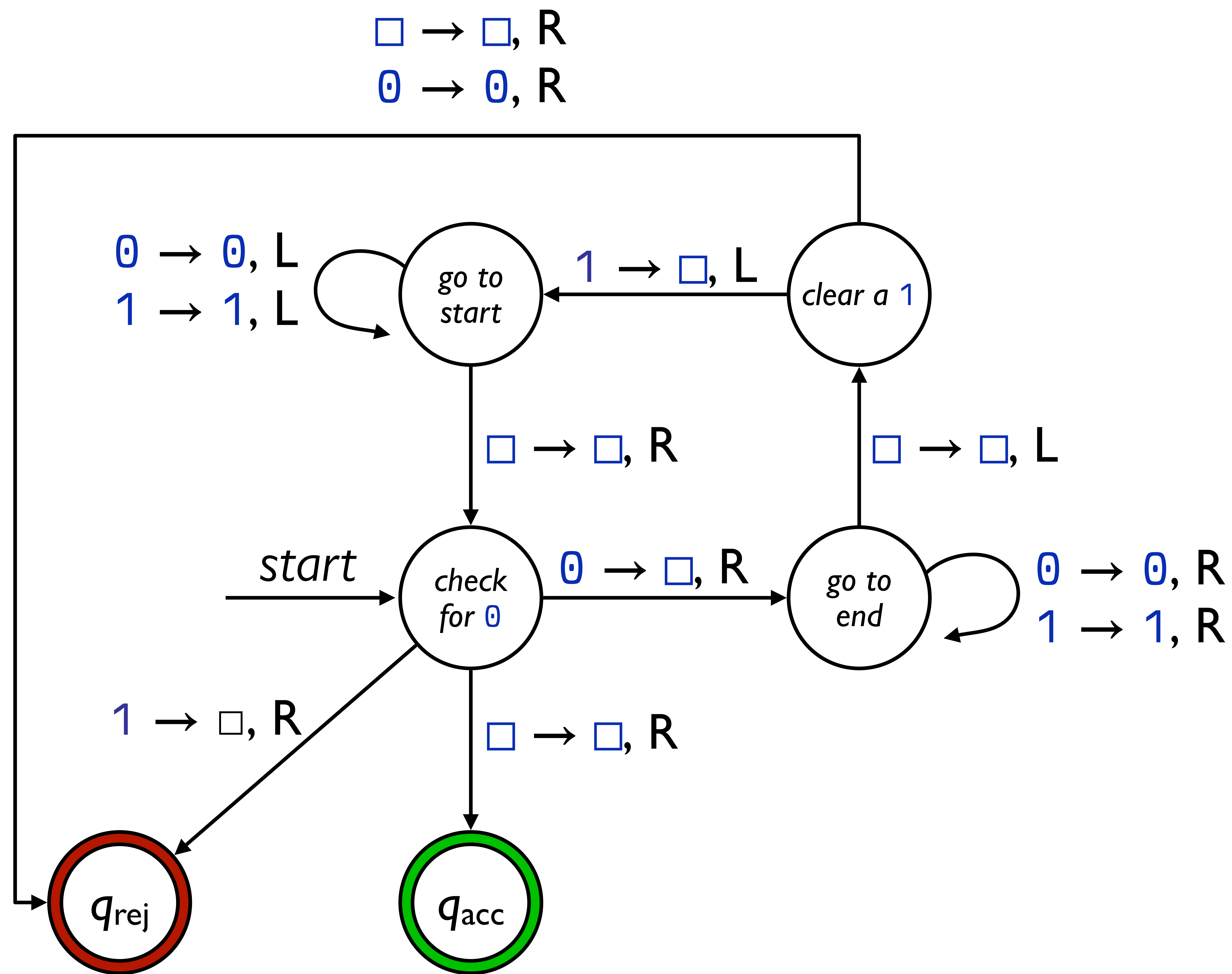
CMPU 240 · Theory of Computation

Turing Machines and Computation

21 April 2026

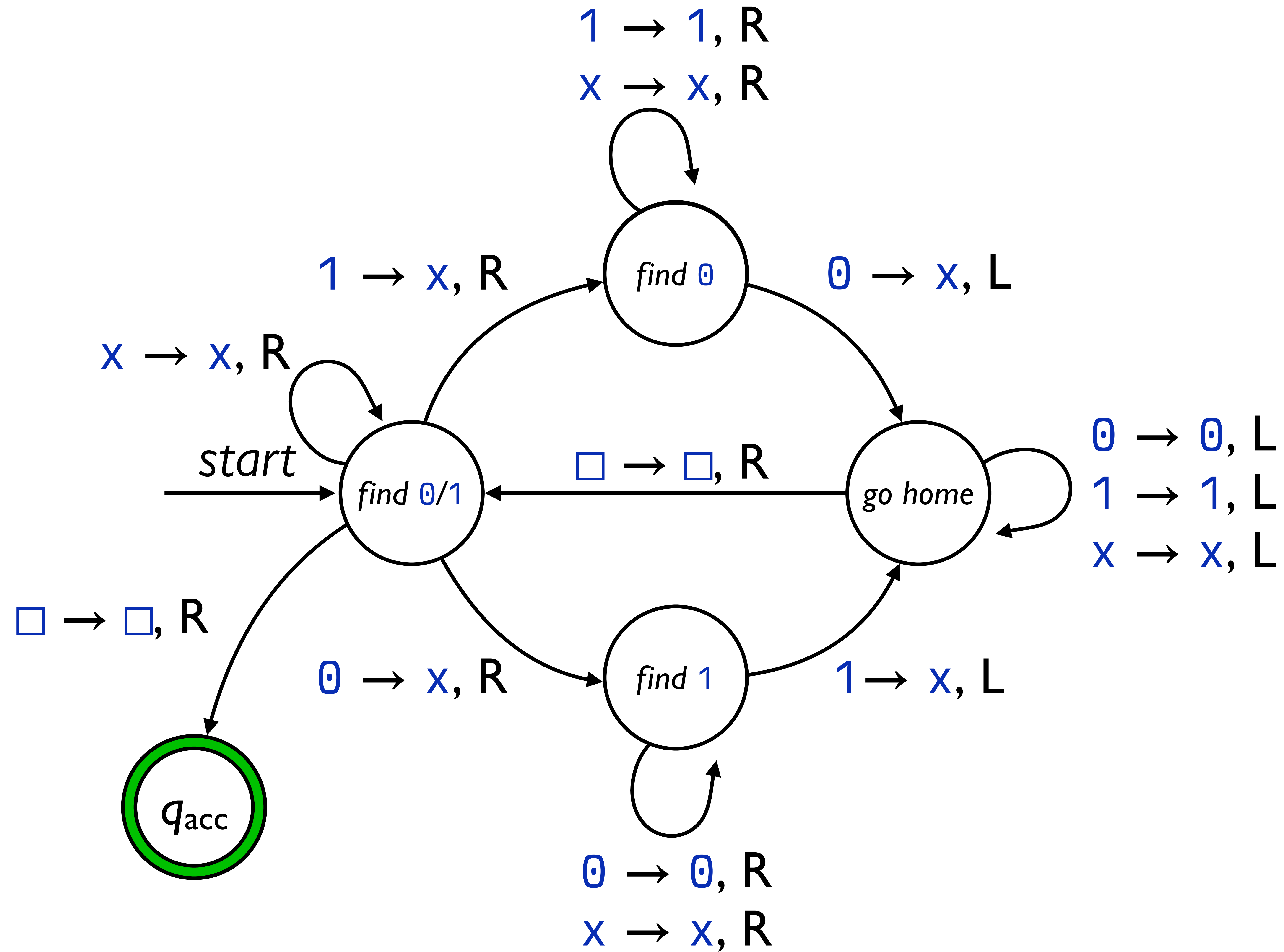
Where are we?

$$\{0^n 1^n \mid n \in \mathbb{N}_0\}$$



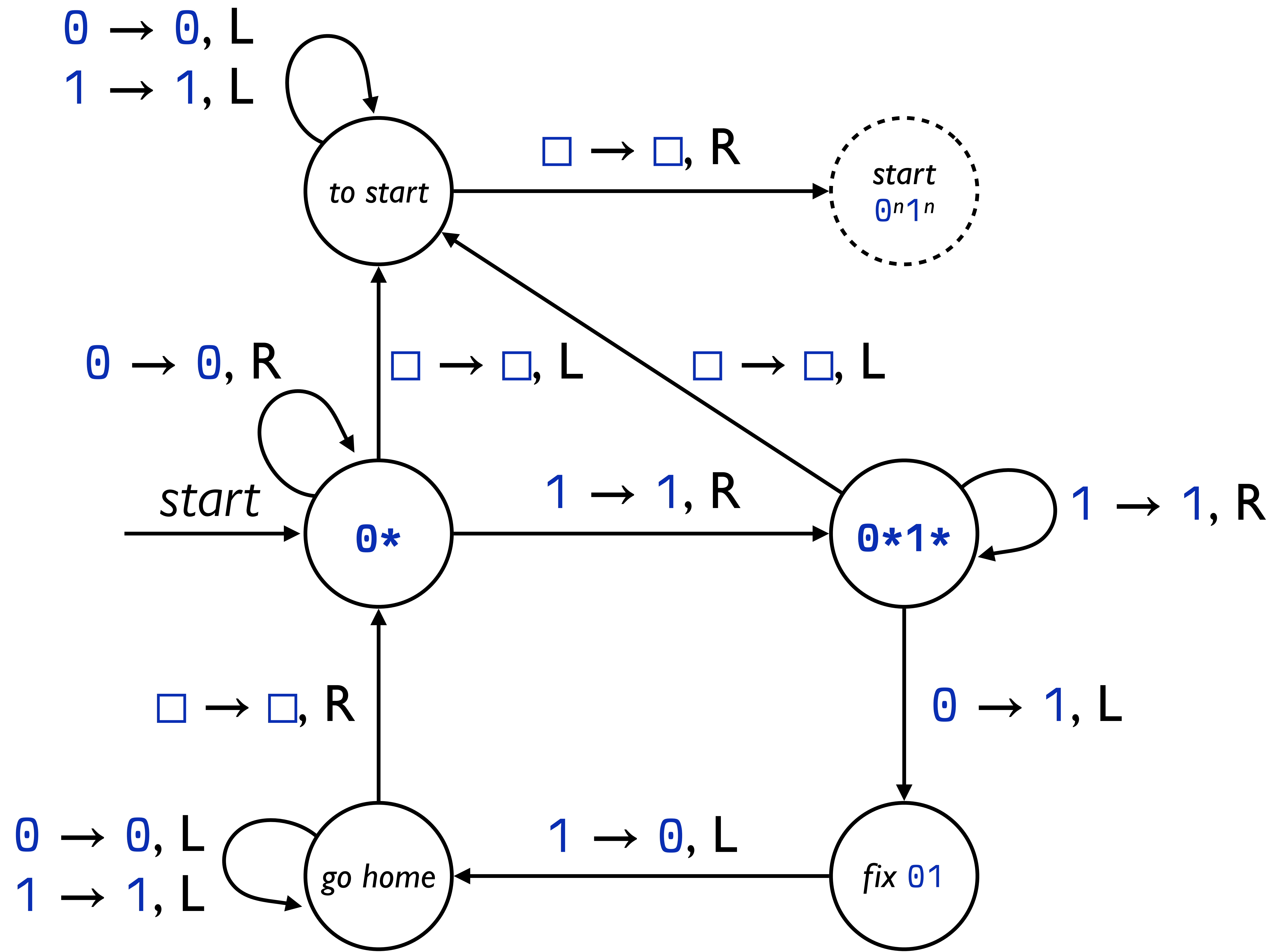
$\{0^n1^n \mid n \in \mathbb{N}_0\}$

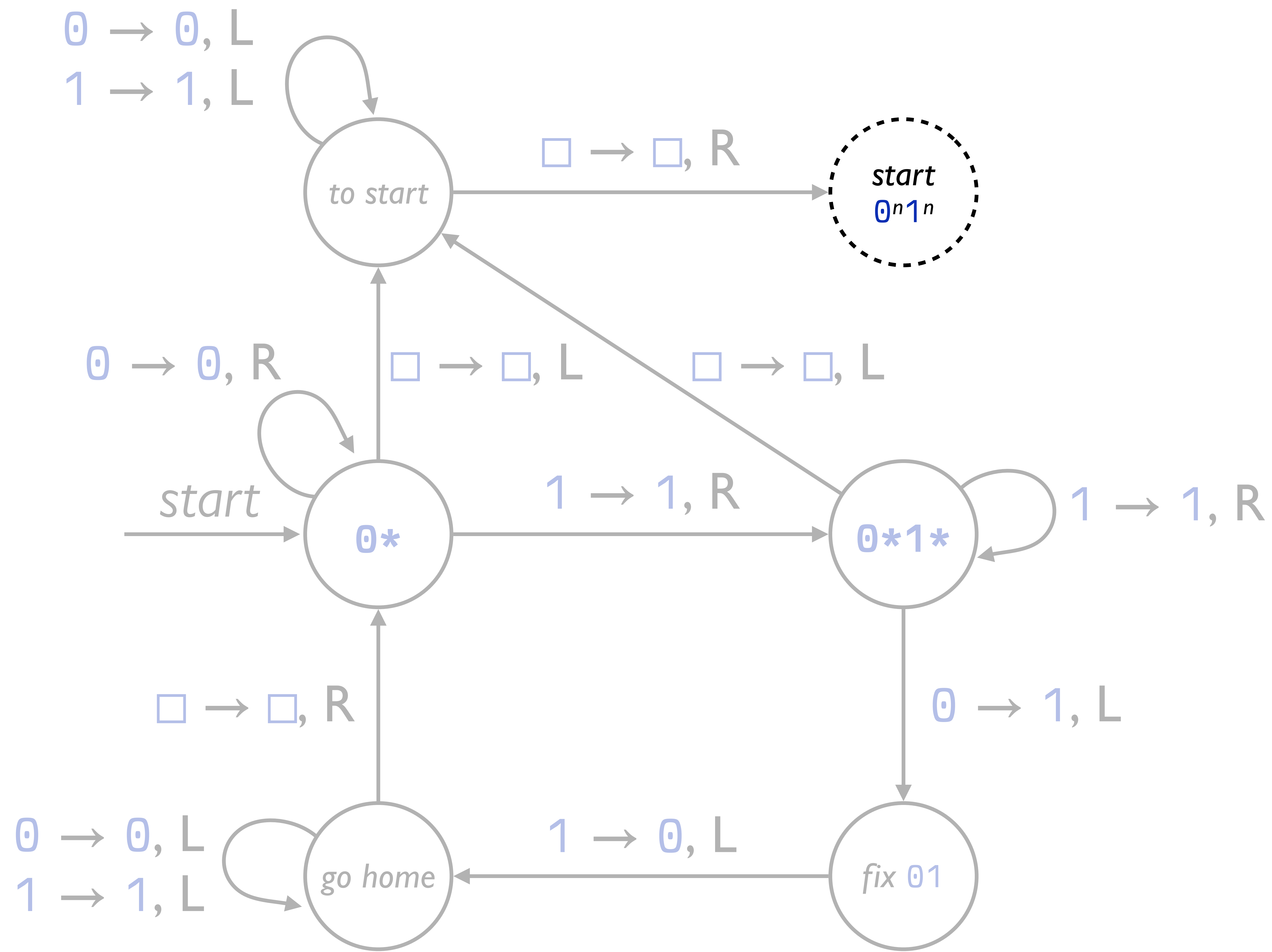
$$\{w \in \Sigma^* \mid n_0(w) = n_1(\mathrm{w})\}$$

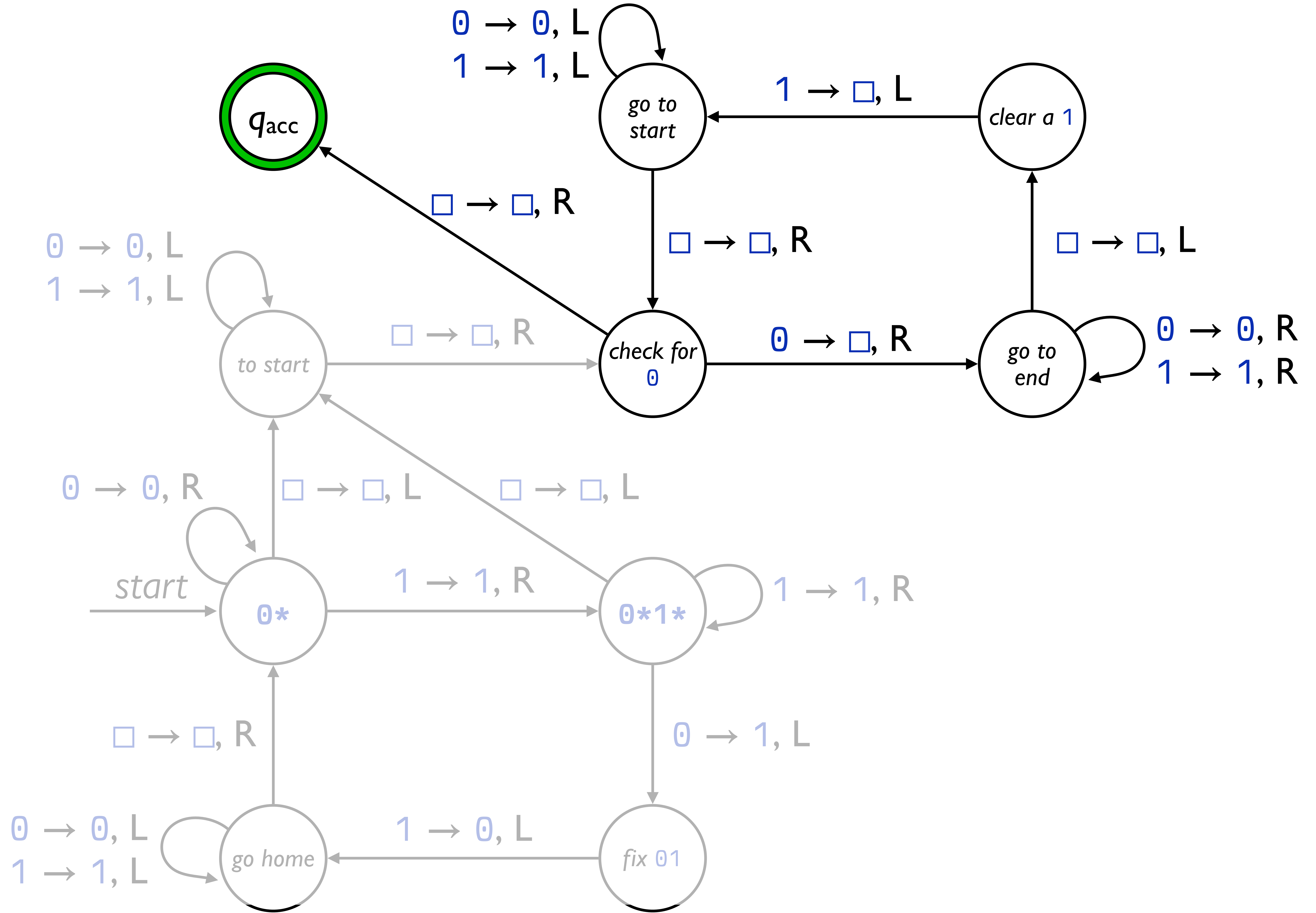


$\{w \in \Sigma^* \mid n_0(w) = n_1(w)\}$

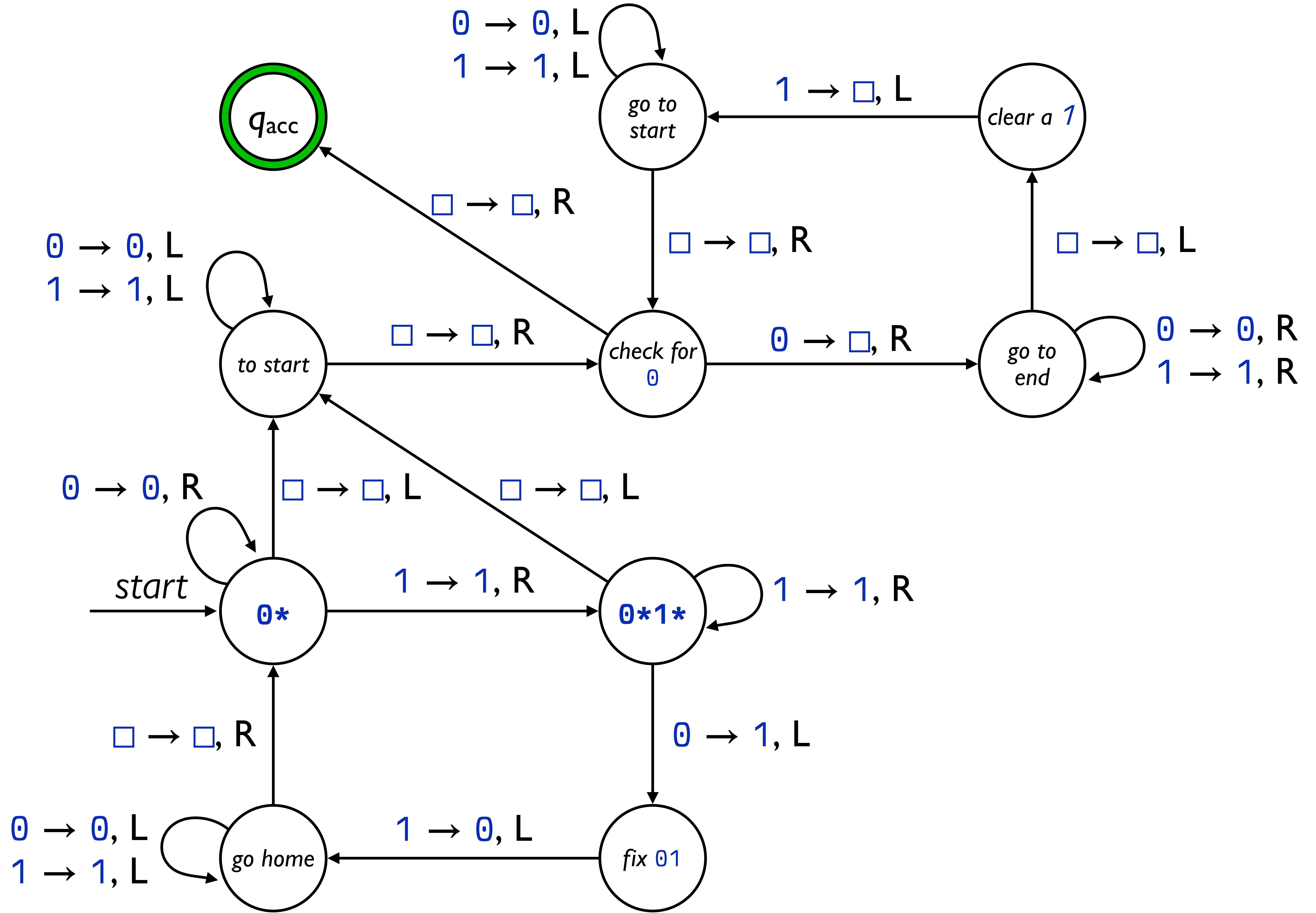
Clearly these languages are related, so we asked if we could use our Turing machine for the first language to help us recognize the second.


0 → 0, L
1 → 1, L
to start
□ → □, R
start
0ⁿ1ⁿ
0 → 0, R
□ → □, L
□ → □, L
start
0*
1 → 1, R
0*1*
1 → 1, R
□ → □, R
0 → 1, L
0 → 0, L
1 → 1, L
go home
1 → 0, L
fix 01


0 → 0, L
1 → 1, L
to start
□ → □, R
start
0ⁿ1ⁿ
0 → 0, R
□ → □, L
□ → □, L
start
0*
1 → 1, R
0*1*
1 → 1, R
□ → □, R
0 → 1, L
0 → 0, L
1 → 1, L
go home
1 → 0, L
fix 01



0 → 0, L
1 → 1, L
go to start
1 → □, L
clear a 1
q_{acc}
□ → □, R
□ → □, R
□ → □, L
0 → 0, L
1 → 1, L
to start
□ → □, R
check for 0
0 → □, R
go to end
0 → 0, R
1 → 1, R
0 → 0, R
□ → □, L
□ → □, L
start
0*
1 → 1, R
0*1*
1 → 1, R
□ → □, R
0 → 1, L
0 → 0, L
1 → 1, L
go home
1 → 0, L
fix 01


qacc
go to start
clear a 1
0 → 0, L
1 → 1, L
1 → □, L
□ → □, R
□ → □, R
□ → □, L
to start
check for 0
go to end
0 → 0, R
1 → 1, R
0 → □, R
□ → □, R
□ → □, L
□ → □, L
0 → 0, R
start
0*
0*1*
1 → 1, R
1 → 1, R
0 → 1, L
□ → □, R
go home
fix 01
1 → 0, L
0 → 0, L
1 → 1, L

A *TM subroutine* is a Turing machine that, instead of accepting or rejecting an input, does some sort of processing job.

TM subroutines let us compose larger TMs out of smaller TMs, just as you'd use a variety of helper functions to write a complicated program.

Here, we saw a TM subroutine to sort a sequence of `0`s and `1`s into ascending order.

Seeing what Turing machines are capable of

Let's design a Turing machine that, given a tape that looks like this:

ends up having the tape look like this:

In other words, a Turing machine that can add two numbers.

There are many ways we could design this Turing machine.

Here's one approach:

- Build a Turing machine to *increment* a number.
- Build a Turing machine to *decrement* a number.
- *Combine them*, repeatedly decrementing the second number and incrementing the first.

```
def add(num1, num2):
    while num2 > 0:
        decrement(num2)
        increment(num1)
```

```
def add(num1, num2):
    while num2 > 0:
        decrement(num2)
        increment(num1)
```

Let's write this subroutine first.

Incrementing numbers

Let's begin by building a TM that increments a number. We'll assume that

- the tape head points at the start of a number
- there are at least two blanks to the left of the number
- there's at least one blank at the end of the number

The tape head will end at the start of the number after incrementing it.

Incrementing numbers

Let's begin by building a TM that increments a number. We'll assume that

- the tape head points at the start of a number
- there are at least two blanks to the left of the number
- there's at least one blank at the end of the number

The tape head will end at the start of the number after incrementing it.

														9	9	8																	

Incrementing numbers

Let's begin by building a TM that increments a number. We'll assume that

- the tape head points at the start of a number
- there are at least two blanks to the left of the number
- there's at least one blank at the end of the number

The tape head will end at the start of the number after incrementing it.

Incrementing numbers

Let's begin by building a TM that increments a number. We'll assume that

- the tape head points at the start of a number
- there are at least two blanks to the left of the number
- there's at least one blank at the end of the number

The tape head will end at the start of the number after incrementing it.

														9	9	8																	

Incrementing numbers

Let's begin by building a TM that increments a number. We'll assume that

- the tape head points at the start of a number
- there are at least two blanks to the left of the number
- there's at least one blank at the end of the number

The tape head will end at the start of the number after incrementing it.

Incrementing numbers

Let's begin by building a TM that increments a number. We'll assume that

- the tape head points at the start of a number
- there are at least two blanks to the left of the number
- there's at least one blank at the end of the number

The tape head will end at the start of the number after incrementing it.

Incrementing numbers

Let's begin by building a TM that increments a number. We'll assume that

- the tape head points at the start of a number
- there are at least two blanks to the left of the number
- there's at least one blank at the end of the number

The tape head will end at the start of the number after incrementing it.

Incrementing numbers

Let's begin by building a TM that increments a number. We'll assume that

- the tape head points at the start of a number
- there are at least two blanks to the left of the number
- there's at least one blank at the end of the number

The tape head will end at the start of the number after incrementing it.

Incrementing numbers

Let's begin by building a TM that increments a number. We'll assume that

- the tape head points at the start of a number
- there are at least two blanks to the left of the number
- there's at least one blank at the end of the number

The tape head will end at the start of the number after incrementing it.

Incrementing numbers

Let's begin by building a TM that increments a number. We'll assume that

- the tape head points at the start of a number
- there are at least two blanks to the left of the number
- there's at least one blank at the end of the number

The tape head will end at the start of the number after incrementing it.

Incrementing numbers

Let's begin by building a TM that increments a number. We'll assume that

- the tape head points at the start of a number
- there are at least two blanks to the left of the number
- there's at least one blank at the end of the number

The tape head will end at the start of the number after incrementing it.

Incrementing numbers

Let's begin by building a TM that increments a number. We'll assume that

- the tape head points at the start of a number
- there are at least two blanks to the left of the number
- there's at least one blank at the end of the number

The tape head will end at the start of the number after incrementing it.

														9	9	9																	

Incrementing numbers

Let's begin by building a TM that increments a number. We'll assume that

- the tape head points at the start of a number
- there are at least two blanks to the left of the number
- there's at least one blank at the end of the number

The tape head will end at the start of the number after incrementing it.

Incrementing numbers

Let's begin by building a TM that increments a number. We'll assume that

- the tape head points at the start of a number
- there are at least two blanks to the left of the number
- there's at least one blank at the end of the number

The tape head will end at the start of the number after incrementing it.

														9	9	0																	

Incrementing numbers

Let's begin by building a TM that increments a number. We'll assume that

- the tape head points at the start of a number
- there are at least two blanks to the left of the number
- there's at least one blank at the end of the number

The tape head will end at the start of the number after incrementing it.

Incrementing numbers

Let’s begin by building a TM that increments a number. We’ll assume that

- the tape head points at the start of a number
- there are at least two blanks to the left of the number
- there’s at least one blank at the end of the number

The tape head will end at the start of the number after incrementing it.

Incrementing numbers

Let's begin by building a TM that increments a number. We'll assume that

- the tape head points at the start of a number
- there are at least two blanks to the left of the number
- there's at least one blank at the end of the number

The tape head will end at the start of the number after incrementing it.

Incrementing numbers

Let's begin by building a TM that increments a number. We'll assume that

- the tape head points at the start of a number
- there are at least two blanks to the left of the number
- there's at least one blank at the end of the number

The tape head will end at the start of the number after incrementing it.

Incrementing numbers

```
def increment(num):
    go to the end of the number
    while current_digit == 9:
        current_digit = 0
        go left one digit
    current_digit += 1
    go to the start of the number
```

														1	2	9	9																

start

1	2	9	9

start

to end

0 → 0, R
1 → 1, R
...
9 → 9, R

														1	2	9	9																

start

to end

0 → 0, R
1 → 1, R
…
9 → 9, R

1	2	9	9

start

to end

0 → 0, R
1 → 1, R
…
9 → 9, R

1	2	9	9

start

to end

0 → 0, R
1 → 1, R
...
9 → 9, R

1	2	9	9

start

to end

0 → 0, R
1 → 1, R
...
9 → 9, R

														1	2	9	9																

start

to end

□ → □, L

0 → 0, R
1 → 1, R
...
9 → 9, R

														1	2	9	9																

start

to end

□ → □, L

0 → 0, R
1 → 1, R
...
9 → 9, R

1	2	9	9

start

to end

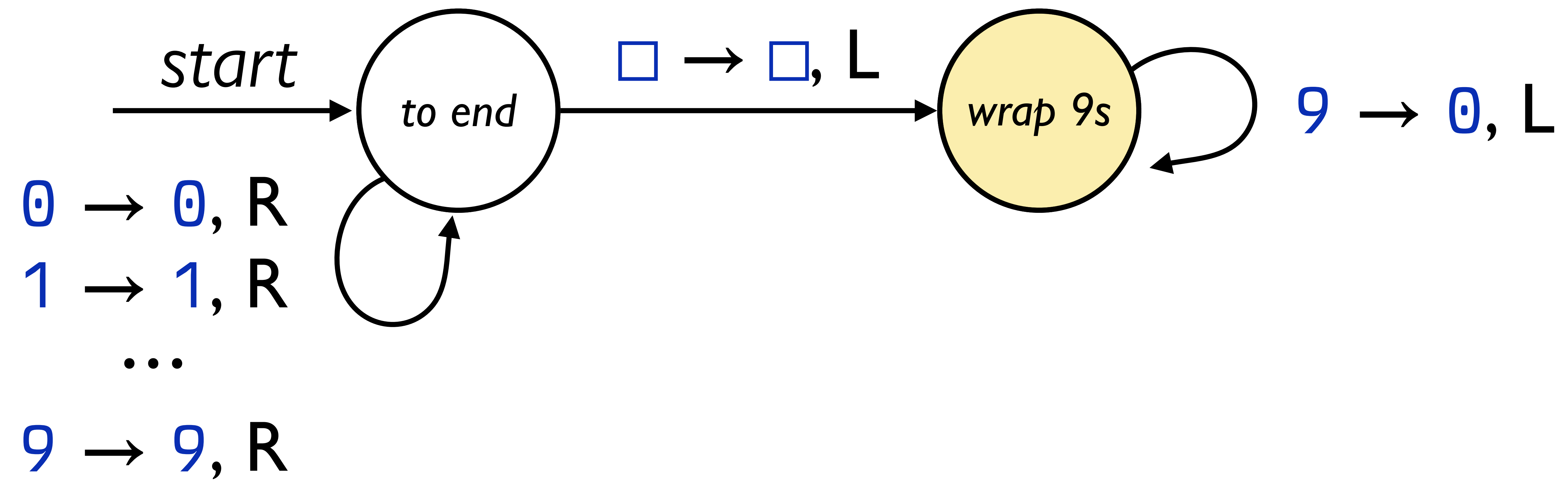
0 → 0, R
1 → 1, R
...
9 → 9, R

□ → □, L

wrap 9s

? → ?, ?

1	2	9	9


start
to end
□ → □, L
wrap 9s
9 → 0, L
0 → 0, R
1 → 1, R
…
9 → 9, R

1
2
9
9

start

to end

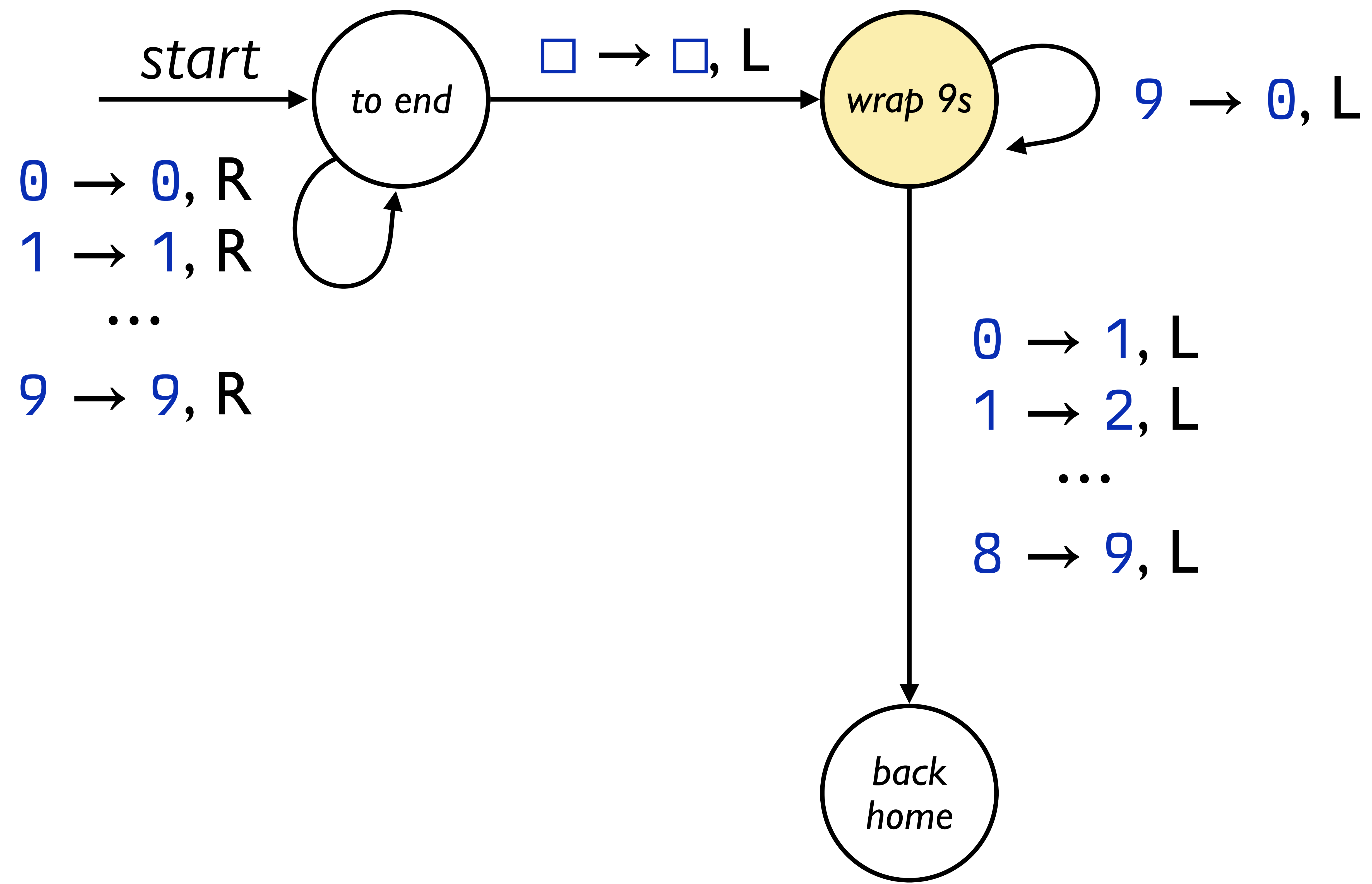
0 → 0, R
1 → 1, R
...
9 → 9, R

□ → □, L

wrap 9s

9 → 0, L

1	2	9	0


start
to end
□ → □, L
wrap 9s
9 → 0, L
0 → 0, R
1 → 1, R
...
9 → 9, R
0 → 1, L
1 → 2, L
...
8 → 9, L
back home

1 2 0 0

start

to end

0 → 0, R
1 → 1, R
...
9 → 9, R

□ → □, L

wrap 9s

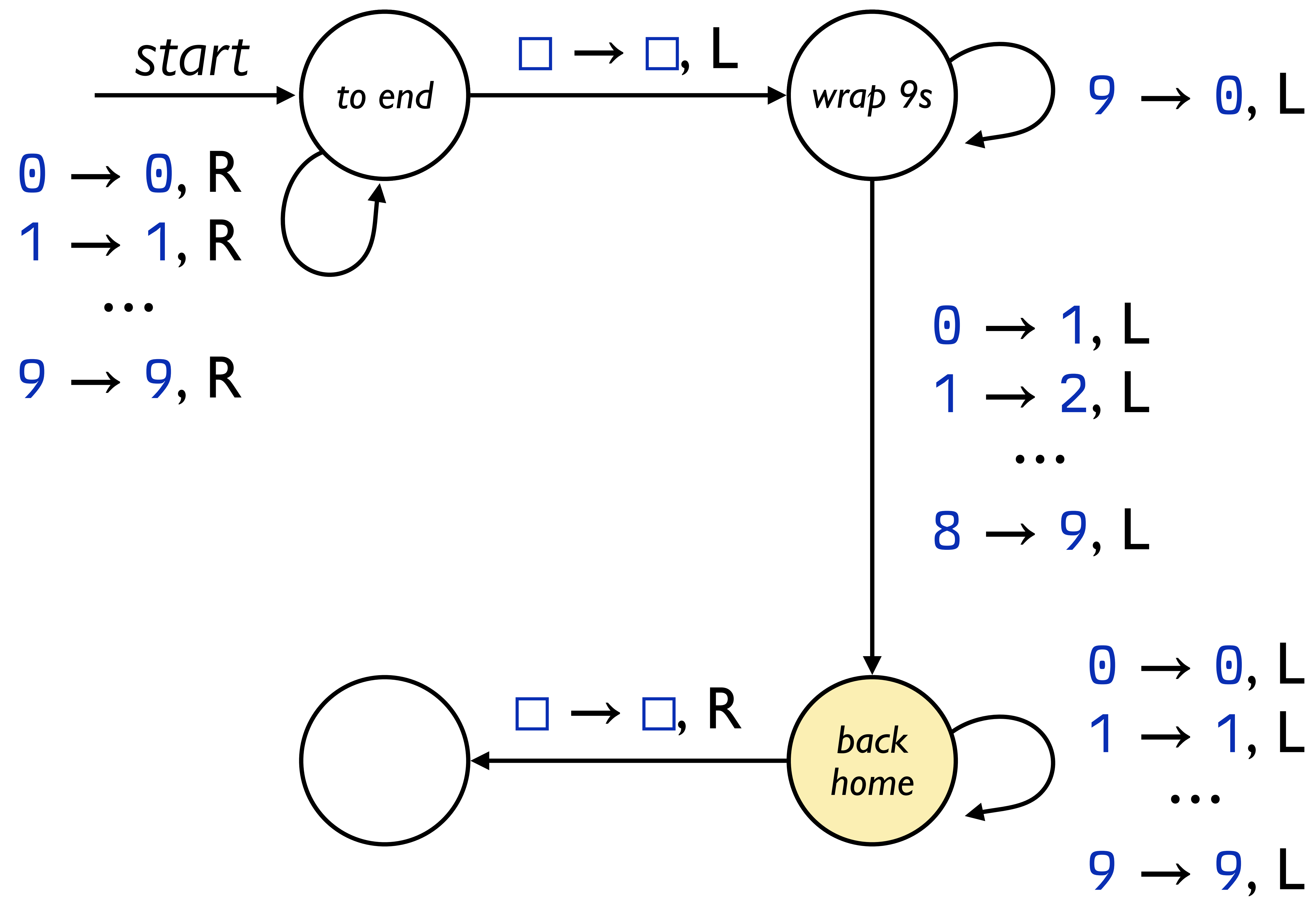
9 → 0, L

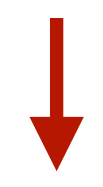
0 → 1, L
1 → 2, L
...
8 → 9, L

back home

0 → 0, L
1 → 1, L
...
9 → 9, L

														1	3	0	0																


start
to end
0 → 0, R
1 → 1, R
...
9 → 9, R
□ → □, L
wrap 9s
9 → 0, L
0 → 1, L
1 → 2, L
...
8 → 9, L
back home
0 → 0, L
1 → 1, L
...
9 → 9, L
□ → □, R



1 3 0 0

start

to end

0 → 0, R
1 → 1, R
...
9 → 9, R

□ → □, L

wrap 9s

9 → 0, L

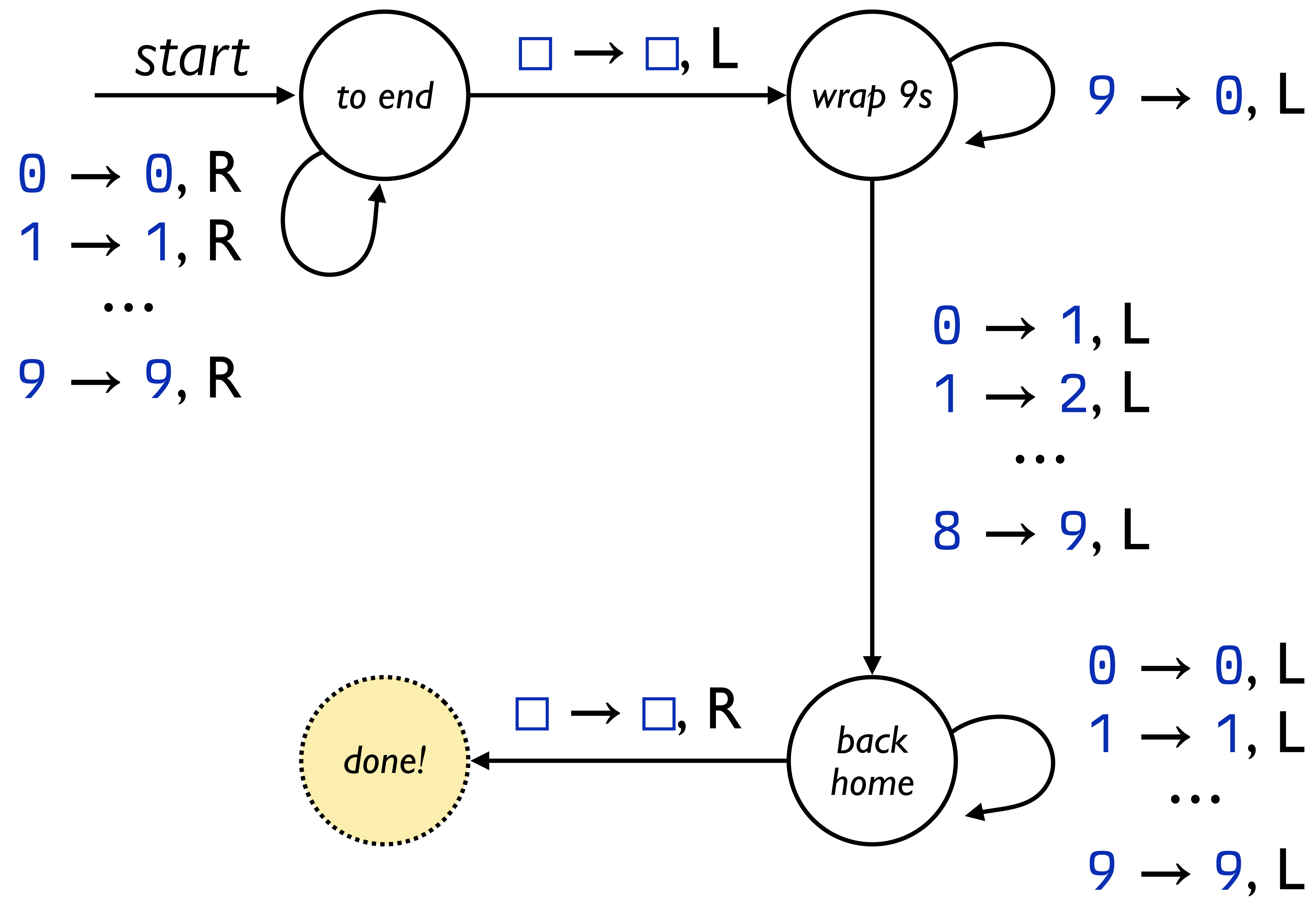
0 → 1, L
1 → 2, L
...
8 → 9, L

back home

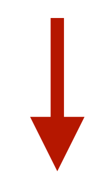
0 → 0, L
1 → 1, L
...
9 → 9, L

□ → □, R

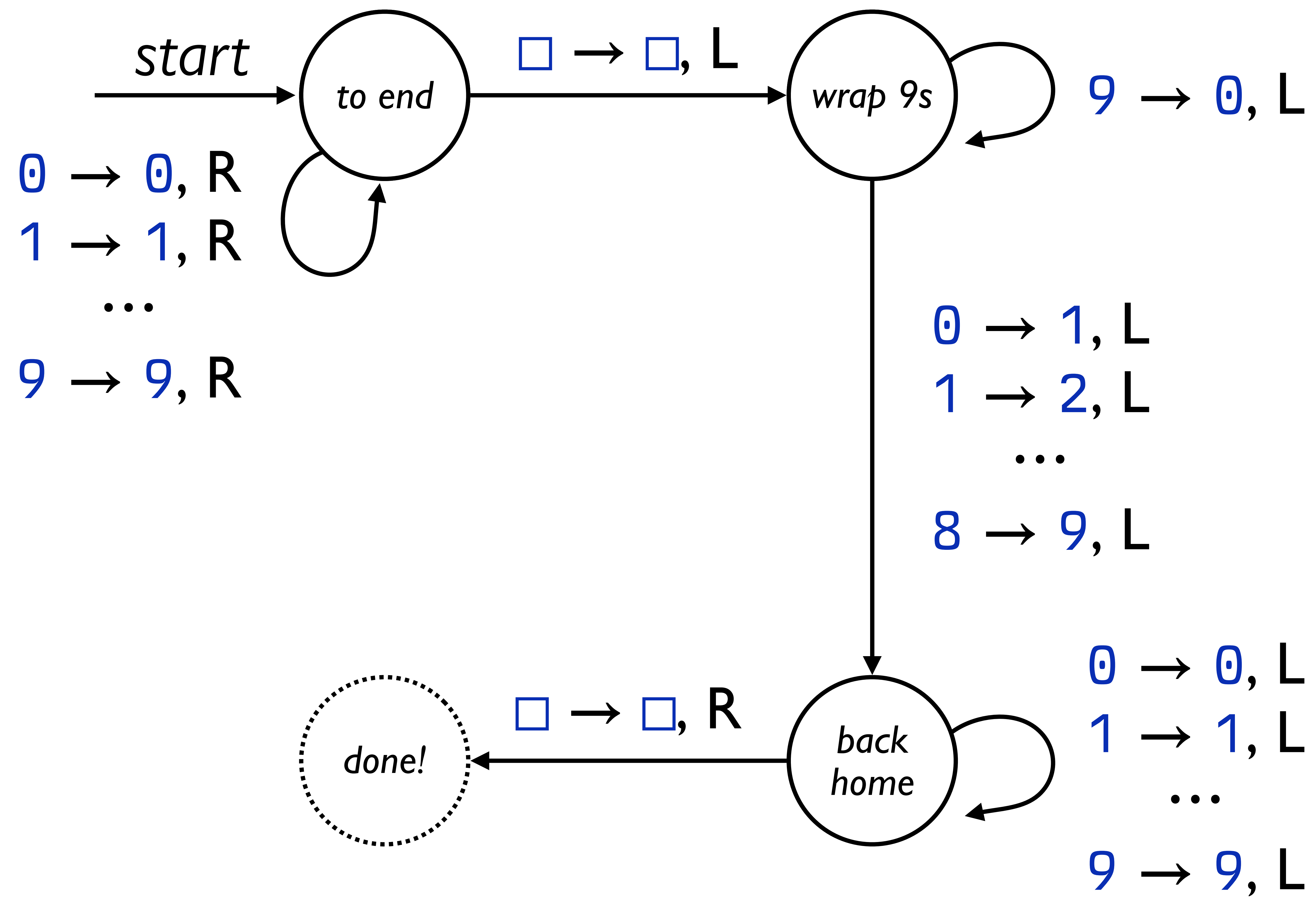
1	3	0	0

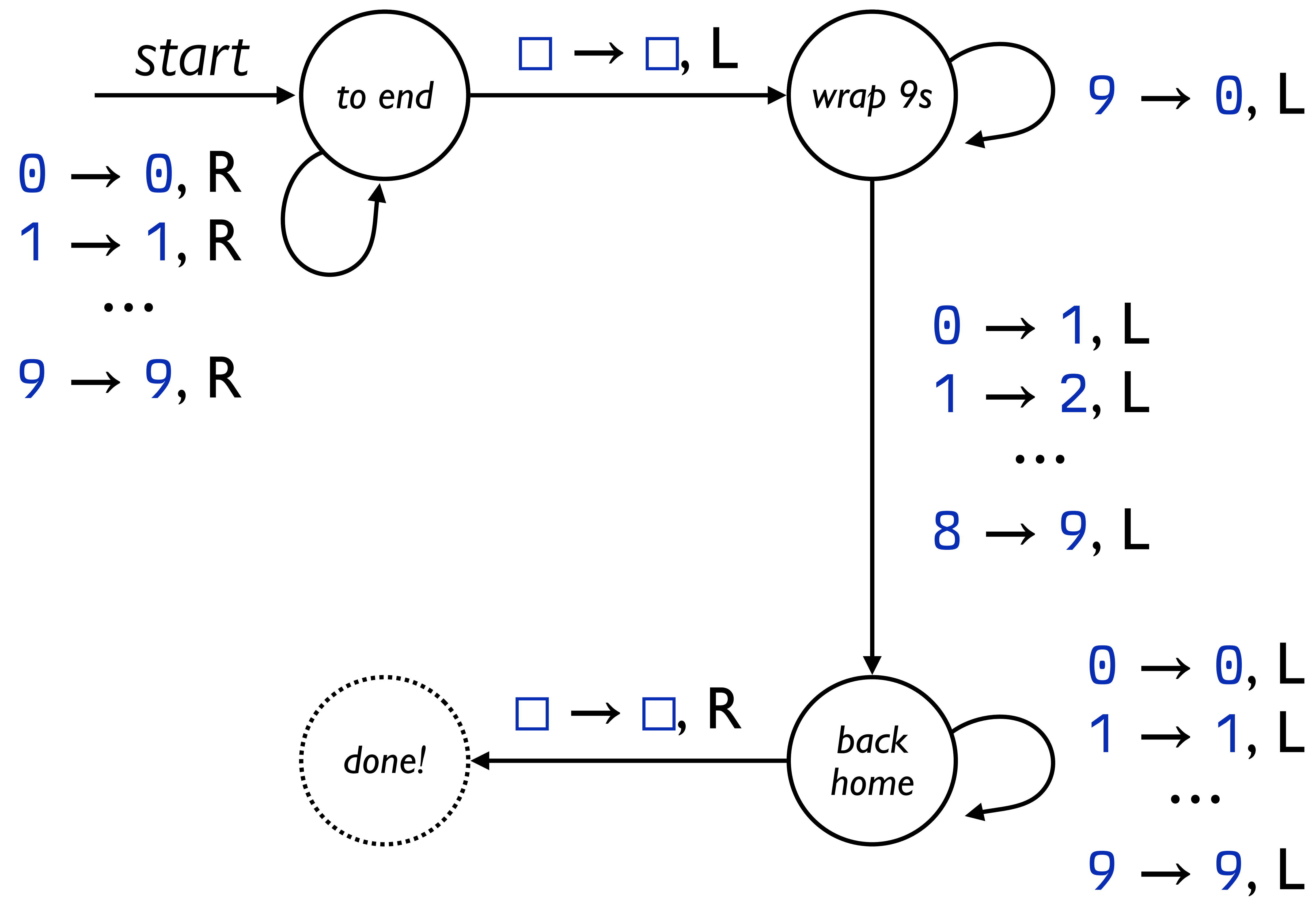


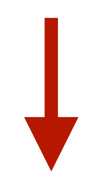
start
to end
□ → □, L
wrap 9s
9 → 0, L
0 → 0, R
1 → 1, R
…
9 → 9, R
0 → 1, L
1 → 2, L
…
8 → 9, L
0 → 0, L
1 → 1, L
…
9 → 9, L
back home
□ → □, R
done!



1 3 0 0


start
to end
□ → □, L
wrap 9s
9 → 0, L
0 → 0, R
1 → 1, R
...
9 → 9, R
0 → 1, L
1 → 2, L
...
8 → 9, L
0 → 0, L
1 → 1, L
...
9 → 9, L
back home
□ → □, R
done!


start
to end
0 → 0, R
1 → 1, R
...
9 → 9, R
□ → □, L
wrap 9s
9 → 0, L
0 → 1, L
1 → 2, L
...
8 → 9, L
back home
0 → 0, L
1 → 1, L
...
9 → 9, L
□ → □, R
done!



1 0 0 2

start

to end

0 → 0, R
1 → 1, R
...
9 → 9, R

□ → □, L

wrap 9s

9 → 0, L

0 → 1, L
1 → 2, L
...
8 → 9, L

back home

0 → 0, L
1 → 1, L
...
9 → 9, L

□ → □, R

done!

1	0	0	2

start

to end

0 → 0, R
1 → 1, R
...
9 → 9, R

□ → □, L

wrap 9s

9 → 0, L

0 → 1, L
1 → 2, L
...
8 → 9, L

back home

0 → 0, L
1 → 1, L
...
9 → 9, L

□ → □, R

done!

1	0	0	2

start

to end

□ → □, L

wrap 9s

9 → 0, L

0 → 0, R
1 → 1, R
…
9 → 9, R

0 → 1, L
1 → 2, L
…
8 → 9, L

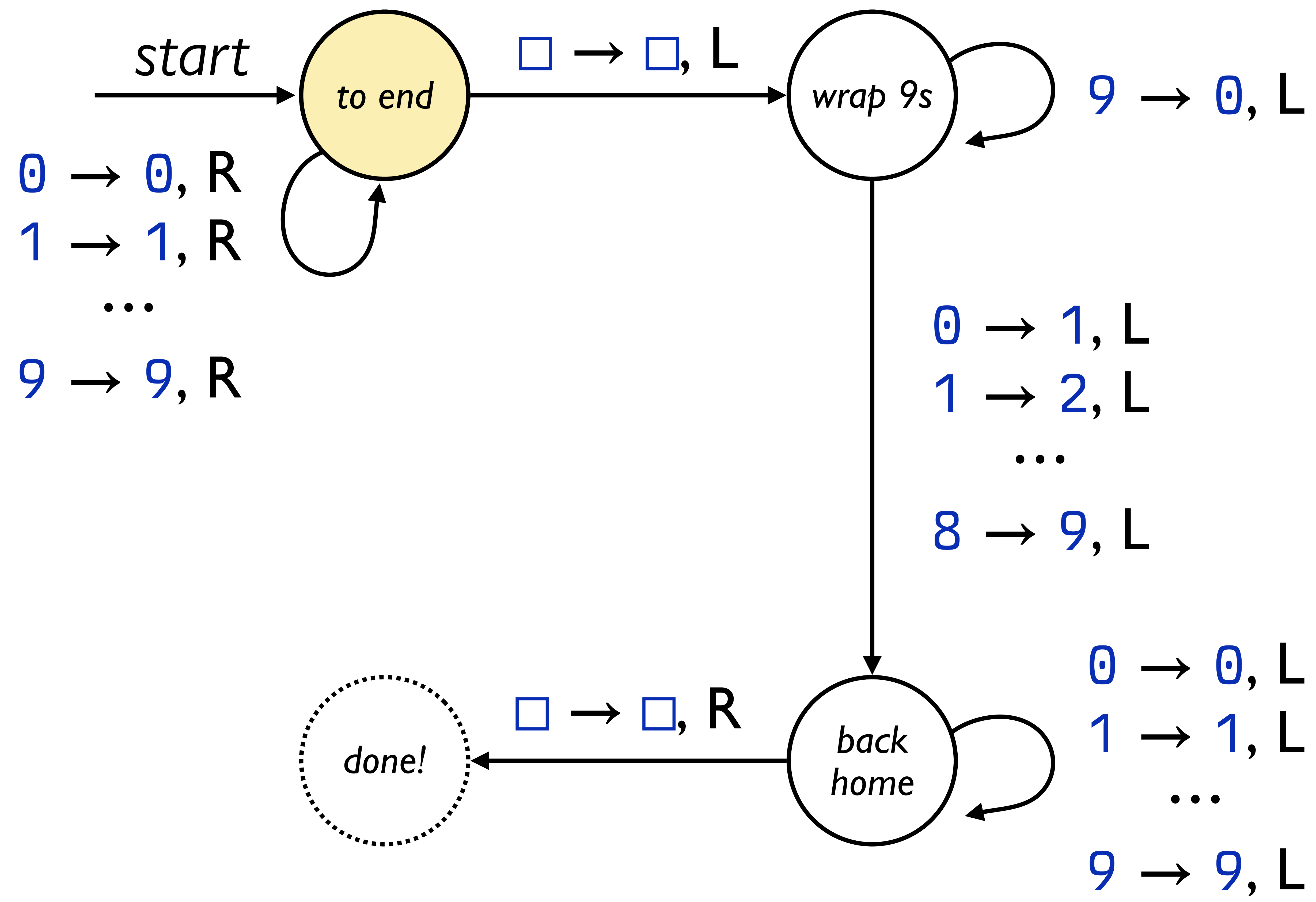
0 → 0, L
1 → 1, L
…
9 → 9, L

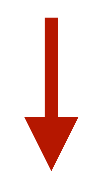
done!

□ → □, R

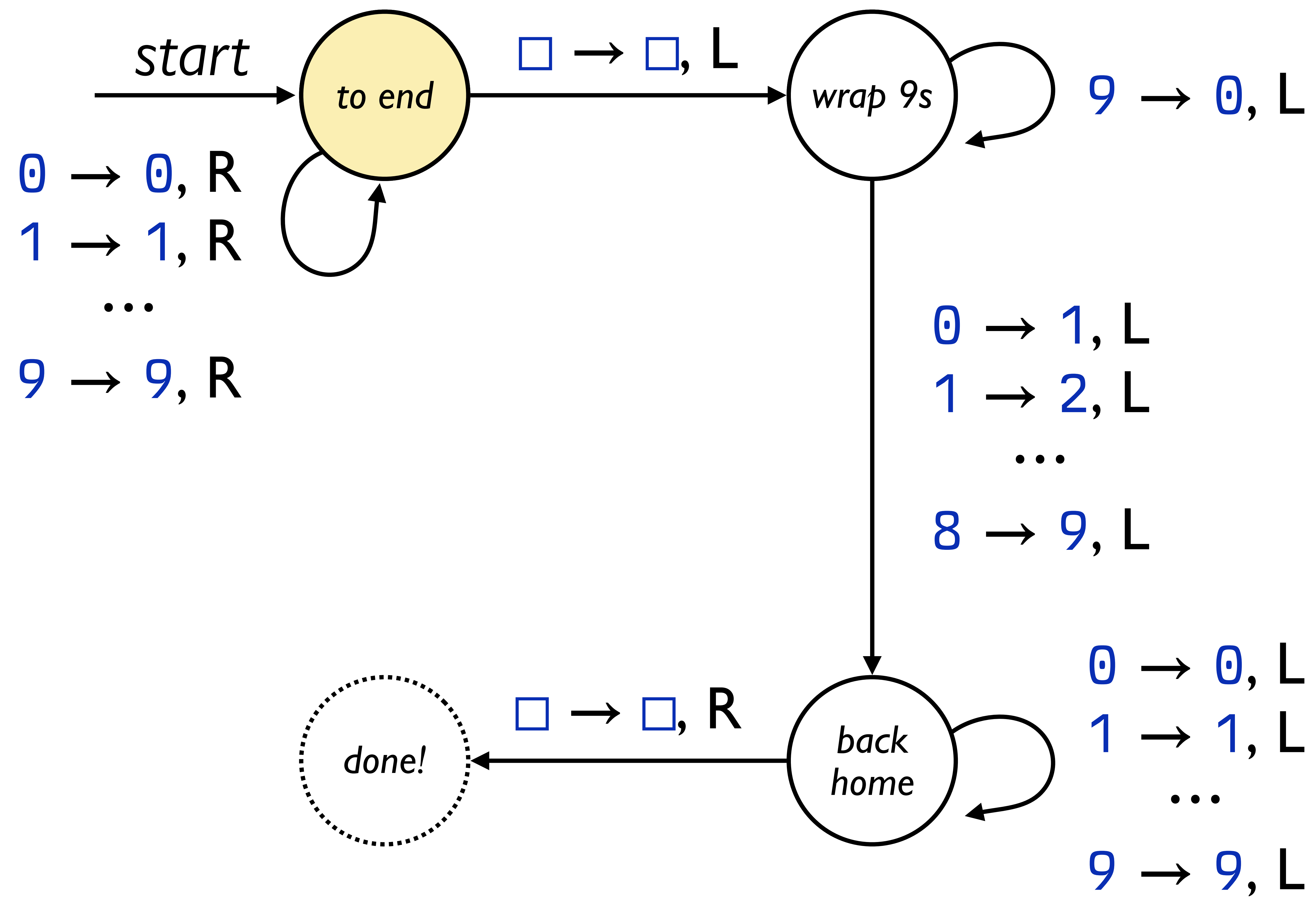
back home

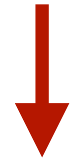
1 0 0 2


start
to end
□ → □, L
wrap 9s
9 → 0, L
0 → 0, R
1 → 1, R
...
9 → 9, R
0 → 1, L
1 → 2, L
...
8 → 9, L
0 → 0, L
1 → 1, L
...
9 → 9, L
□ → □, R
done!
back home



1 0 0 2


start
to end
0 → 0, R
1 → 1, R
...
9 → 9, R
□ → □, L
wrap 9s
9 → 0, L
0 → 1, L
1 → 2, L
...
8 → 9, L
back home
0 → 0, L
1 → 1, L
...
9 → 9, L
□ → □, R
done!



1 0 0 2

start

to end

□ → □, L

wrap 9s

9 → 0, L

0 → 0, R
1 → 1, R
…
9 → 9, R

0 → 1, L
1 → 2, L
…
8 → 9, L

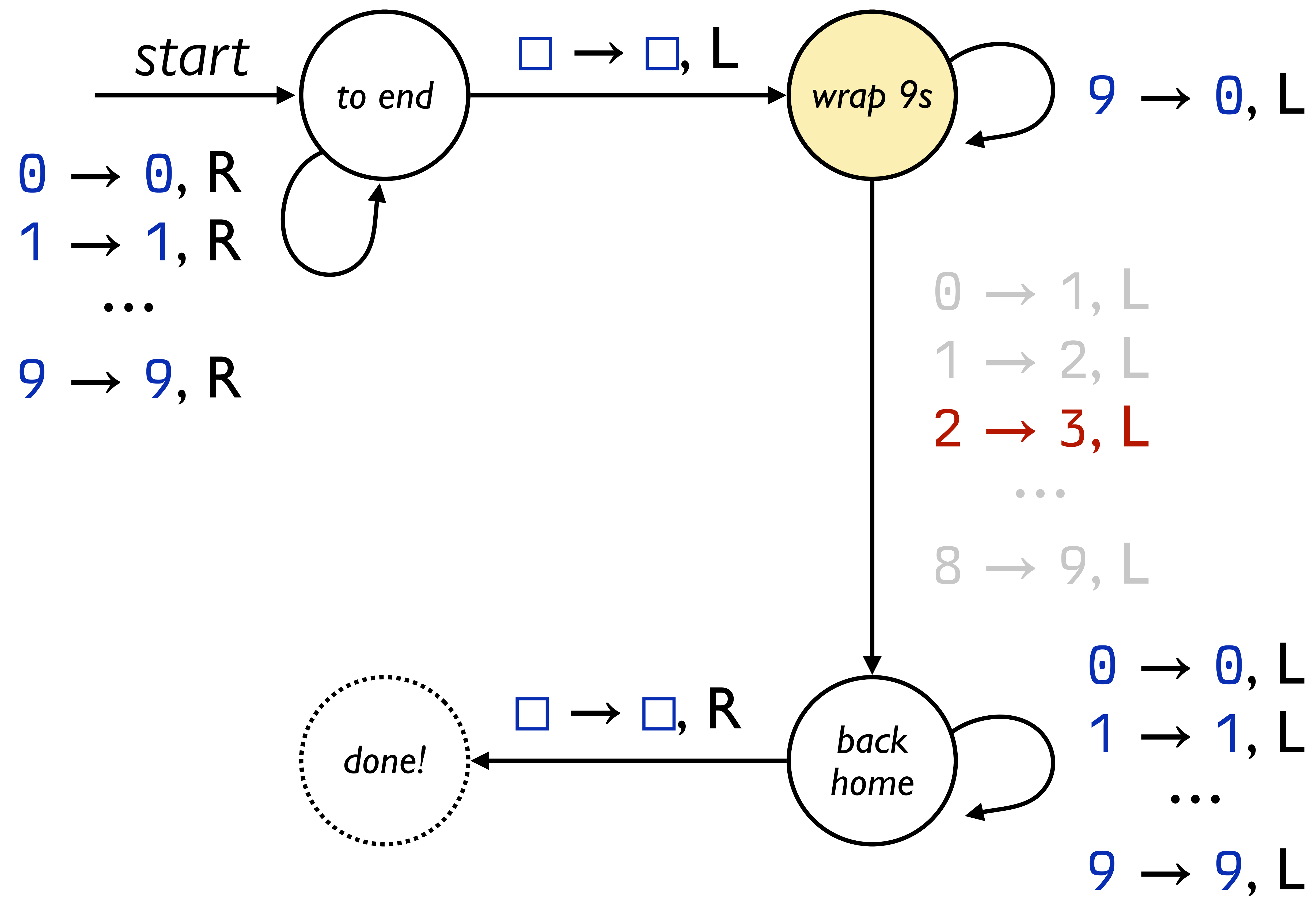
done!

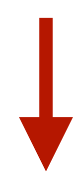
□ → □, R

back home

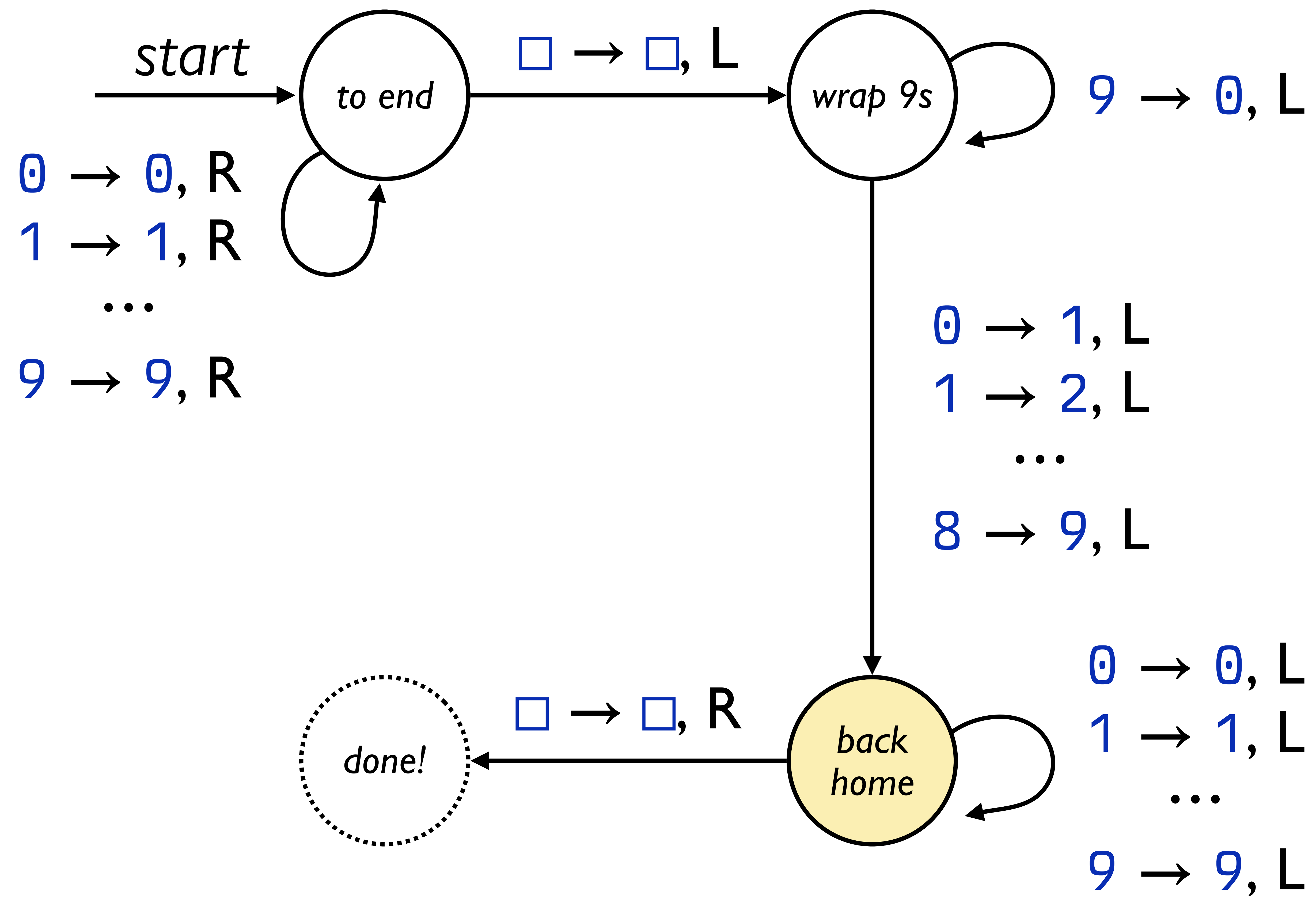
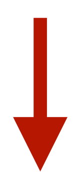
0 → 0, L
1 → 1, L
…
9 → 9, L

1	0	0	2


start
to end
□ → □, L
wrap 9s
9 → 0, L
0 → 0, R
1 → 1, R
…
9 → 9, R
0 → 1, L
1 → 2, L
2 → 3, L
…
8 → 9, L
0 → 0, L
1 → 1, L
…
9 → 9, L
□ → □, R
done!
back home



1 0 0 2


start
to end
0 → 0, R
1 → 1, R
...
9 → 9, R
□ → □, L
wrap 9s
9 → 0, L
0 → 1, L
1 → 2, L
...
8 → 9, L
back home
0 → 0, L
1 → 1, L
...
9 → 9, L
□ → □, R
done!


1 0 0 3

start

to end

0 → 0, R
1 → 1, R
...
9 → 9, R

□ → □, L

wrap 9s

9 → 0, L

0 → 1, L
1 → 2, L
...
8 → 9, L

back home

0 → 0, L
1 → 1, L
...
9 → 9, L

□ → □, R

done!

														1	0	0	3																

start
to end
0 → 0, R
1 → 1, R
...
9 → 9, R
□ → □, L
wrap 9s
9 → 0, L
0 → 1, L
1 → 2, L
...
8 → 9, L
back home
0 → 0, L
1 → 1, L
...
9 → 9, L
□ → □, R
done!
1 0 0 3

start

to end

0 → 0, R
1 → 1, R
…
9 → 9, R

□ → □, L

wrap 9s

9 → 0, L

0 → 1, L
1 → 2, L
…
8 → 9, L

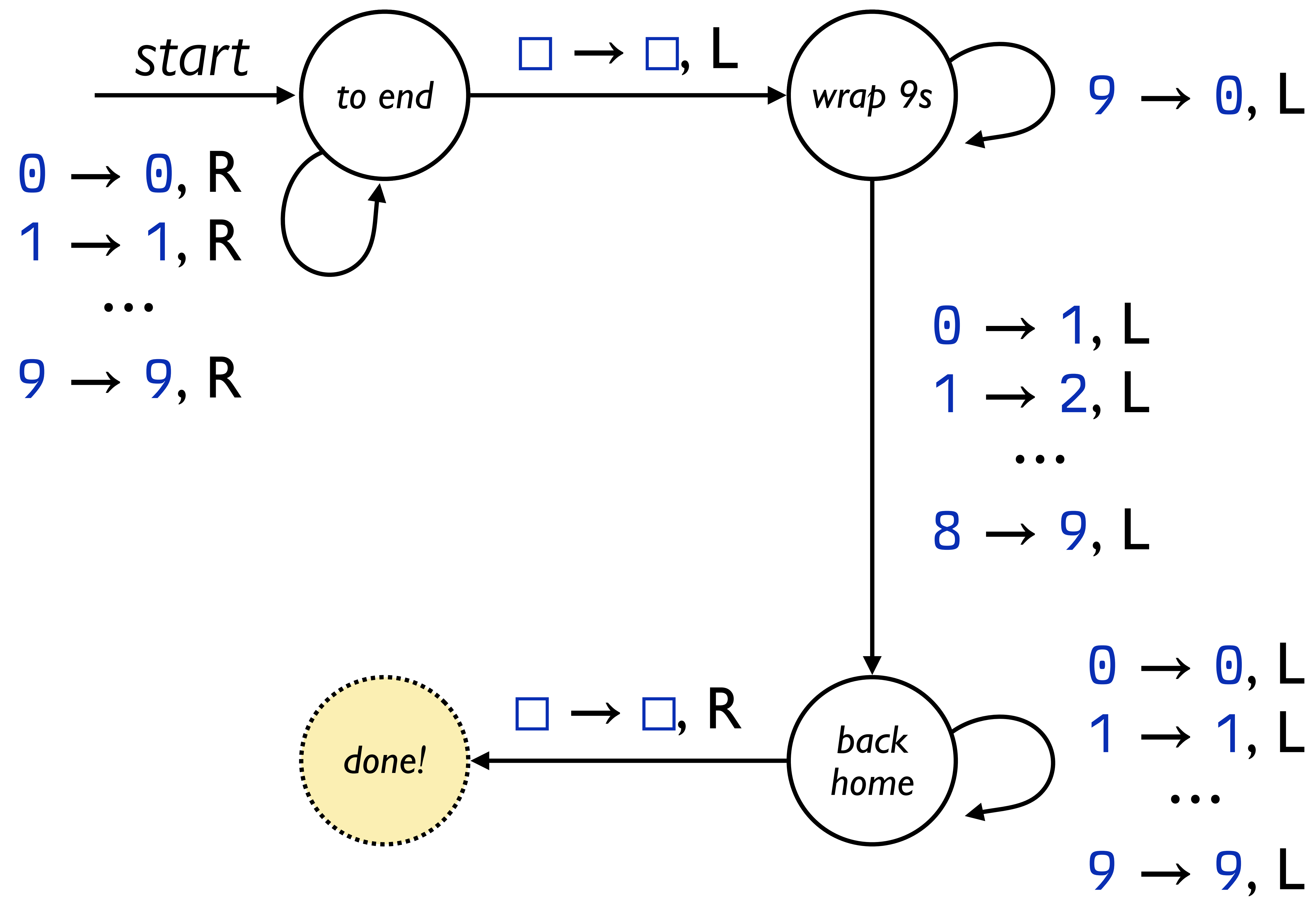
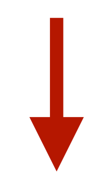
back home

0 → 0, L
1 → 1, L
…
9 → 9, L

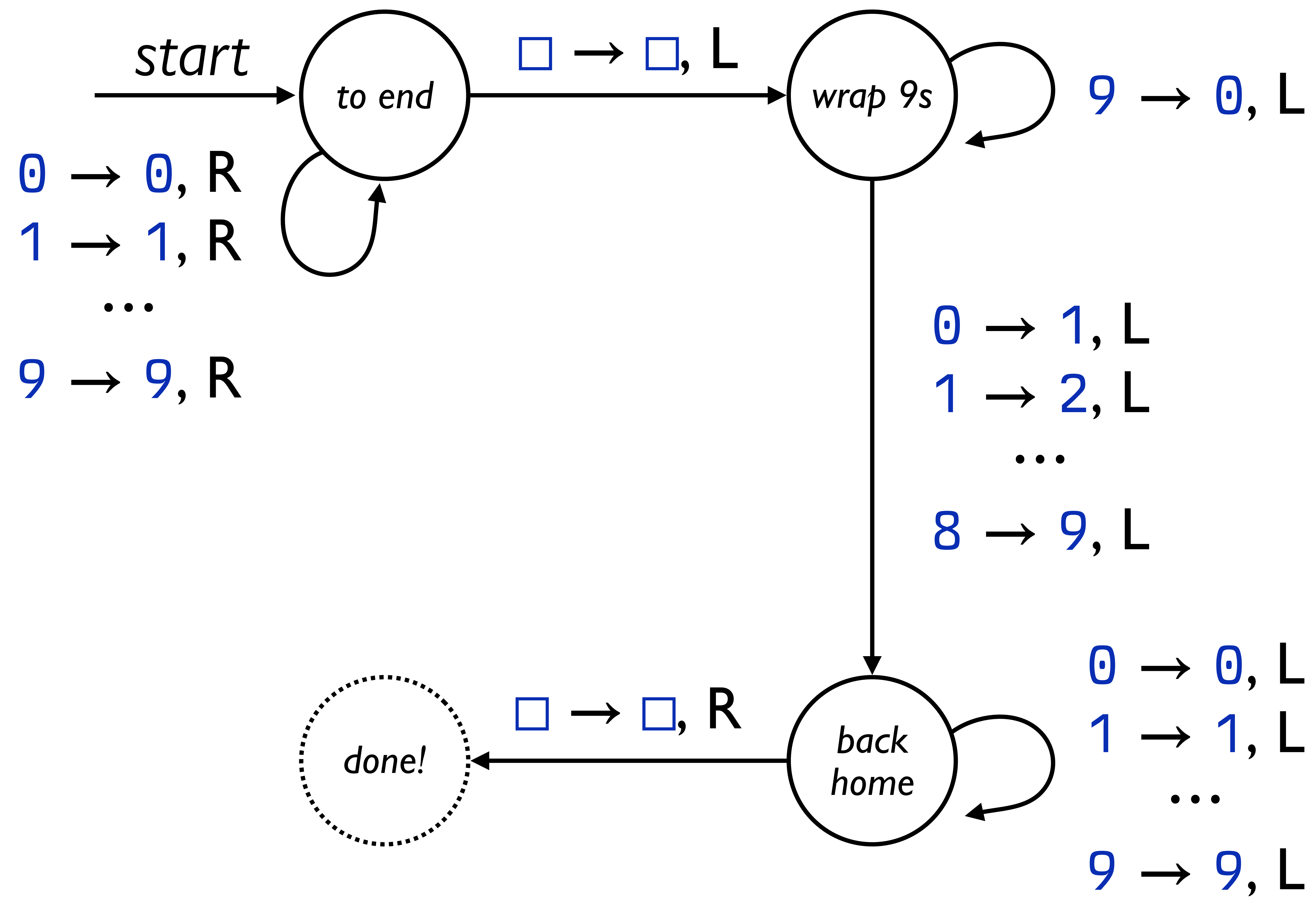
□ → □, R

done!

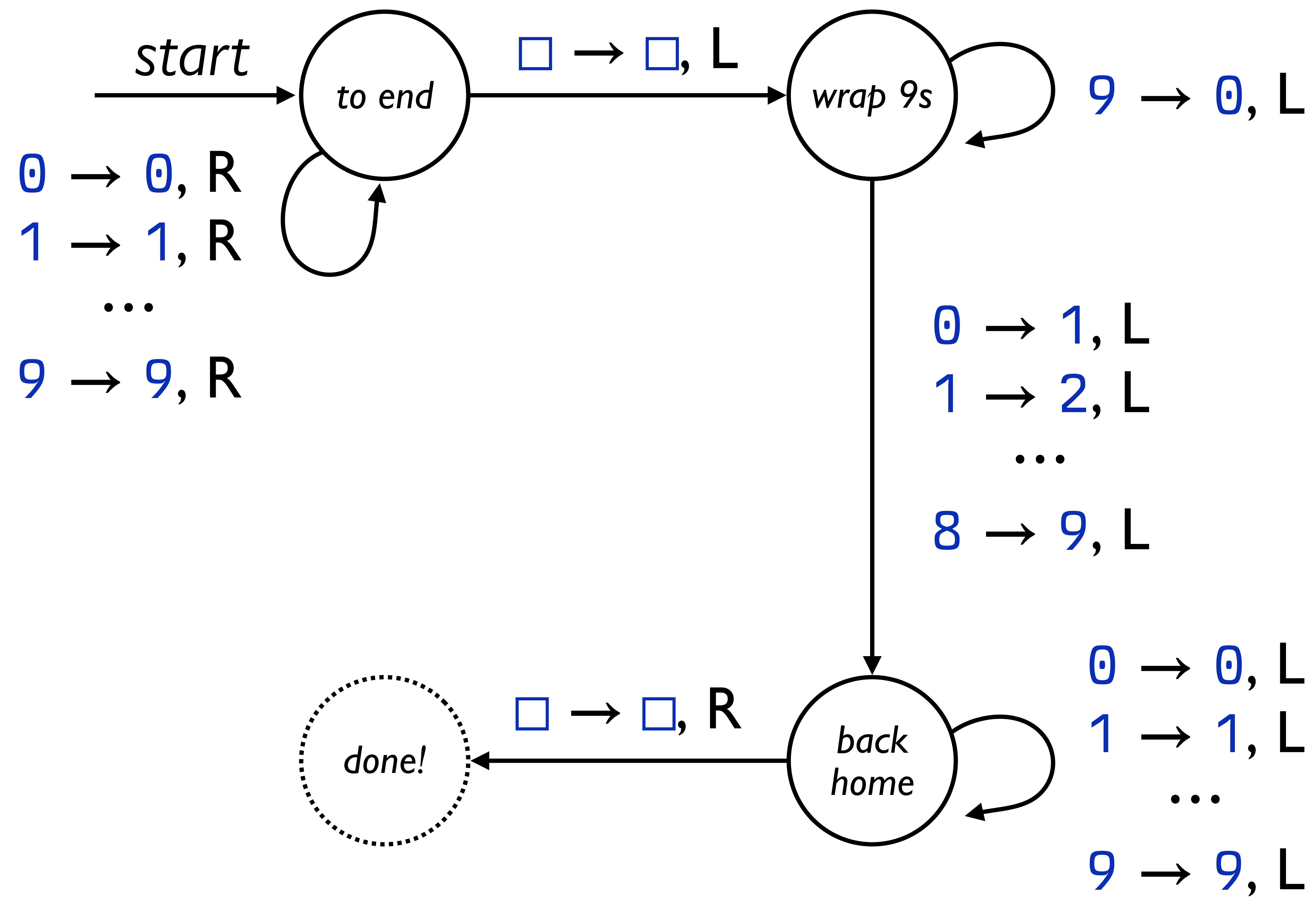
														1	0	0	3																

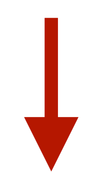

start
to end
□ → □, L
wrap 9s
9 → 0, L
0 → 0, R
1 → 1, R
...
9 → 9, R
0 → 1, L
1 → 2, L
...
8 → 9, L
0 → 0, L
1 → 1, L
...
9 → 9, L
□ → □, R
back home
done!


1 0 0 3



start
to end
□ → □, L
wrap 9s
9 → 0, L
0 → 0, R
1 → 1, R
…
9 → 9, R
0 → 1, L
1 → 2, L
…
8 → 9, L
0 → 0, L
1 → 1, L
…
9 → 9, L
done!
□ → □, R
back home


start
to end
0 → 0, R
1 → 1, R
...
9 → 9, R
□ → □, L
wrap 9s
9 → 0, L
0 → 1, L
1 → 2, L
...
8 → 9, L
back home
0 → 0, L
1 → 1, L
...
9 → 9, L
□ → □, R
done!



9 9 9

start

to end

□ → □, L

wrap 9s

9 → 0, L

0 → 0, R
1 → 1, R
…
9 → 9, R

0 → 1, L
1 → 2, L
…
8 → 9, L

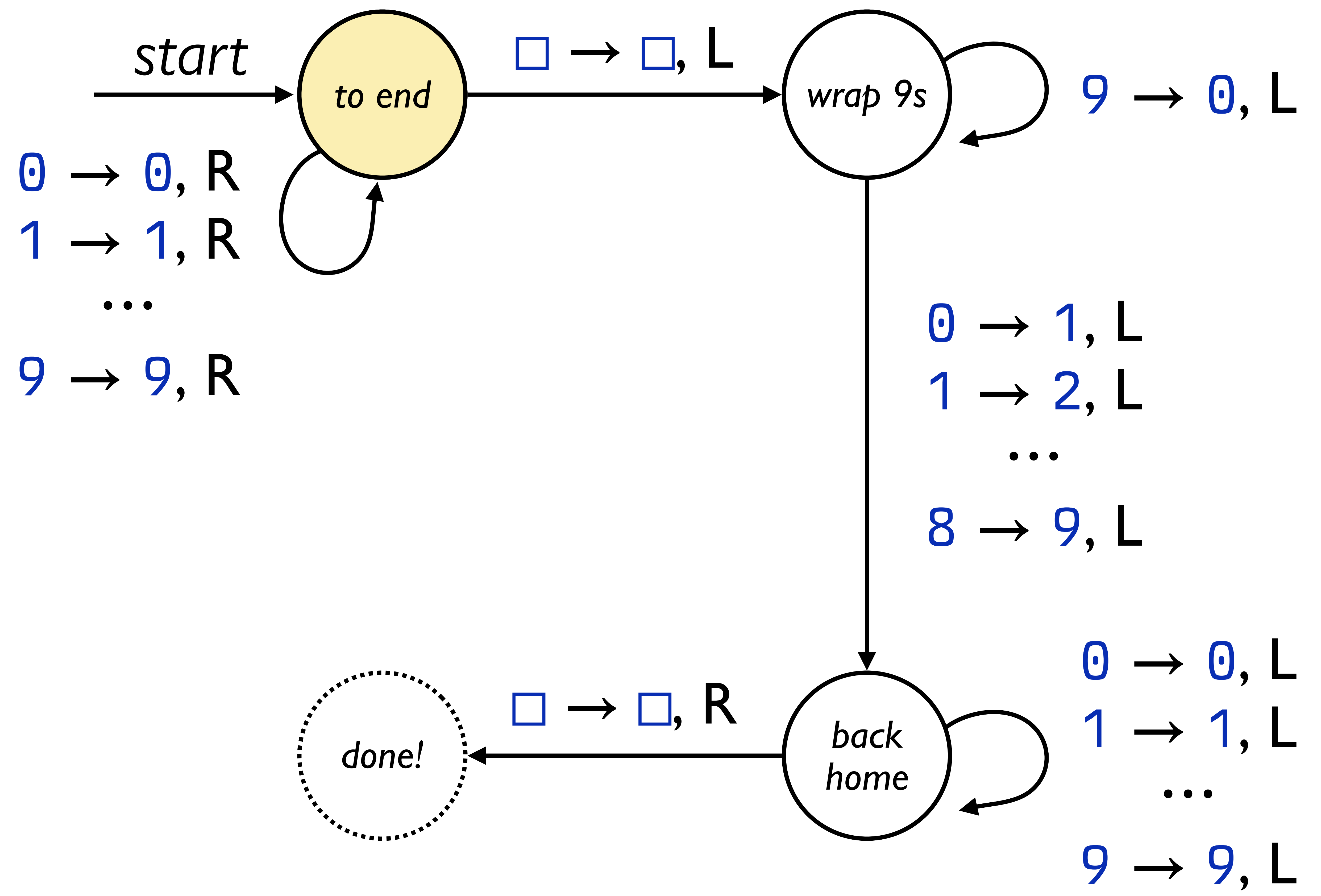
0 → 0, L
1 → 1, L
…
9 → 9, L

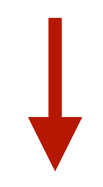
□ → □, R

done!

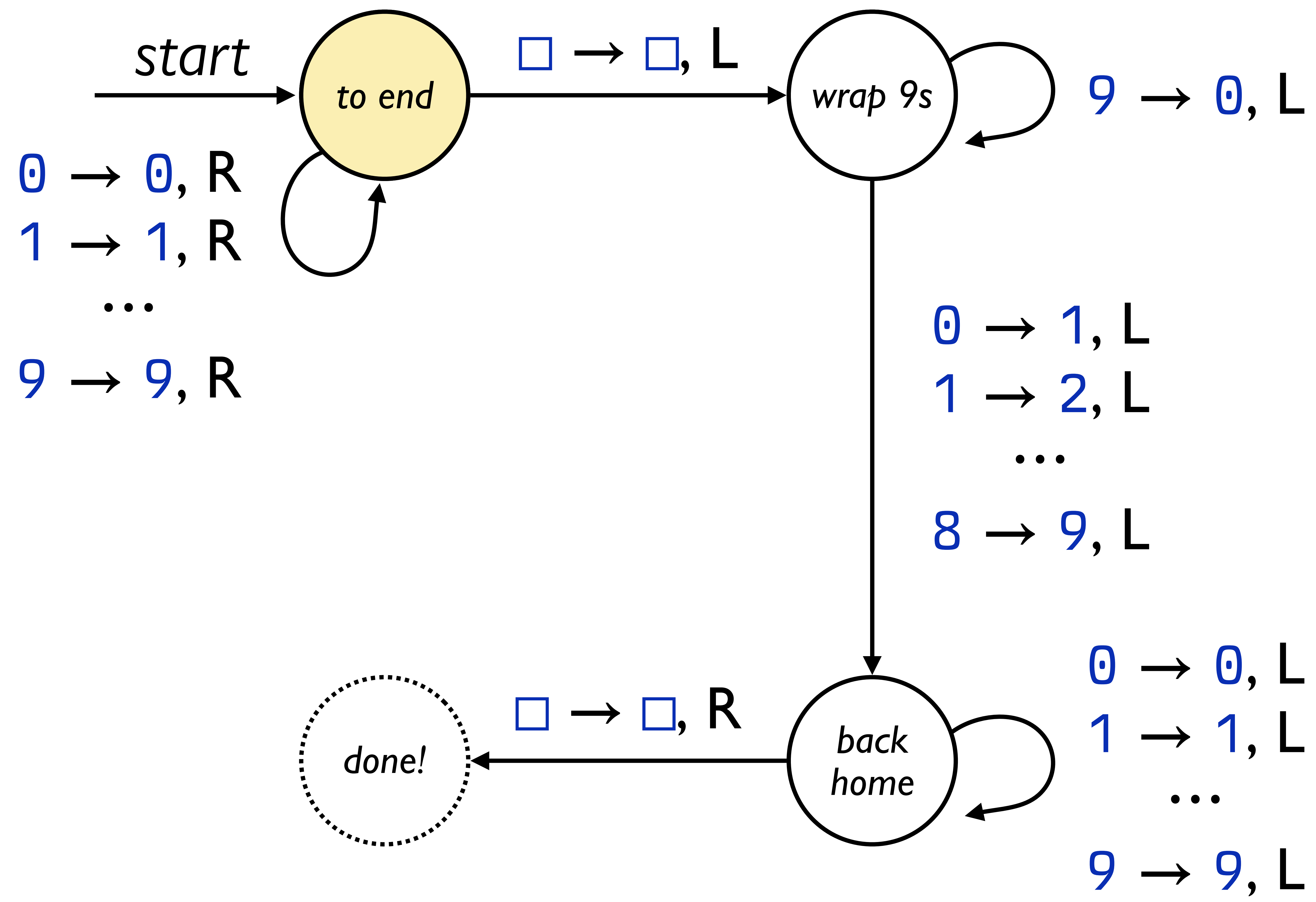
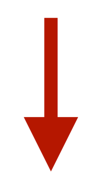
back home

9 9 9

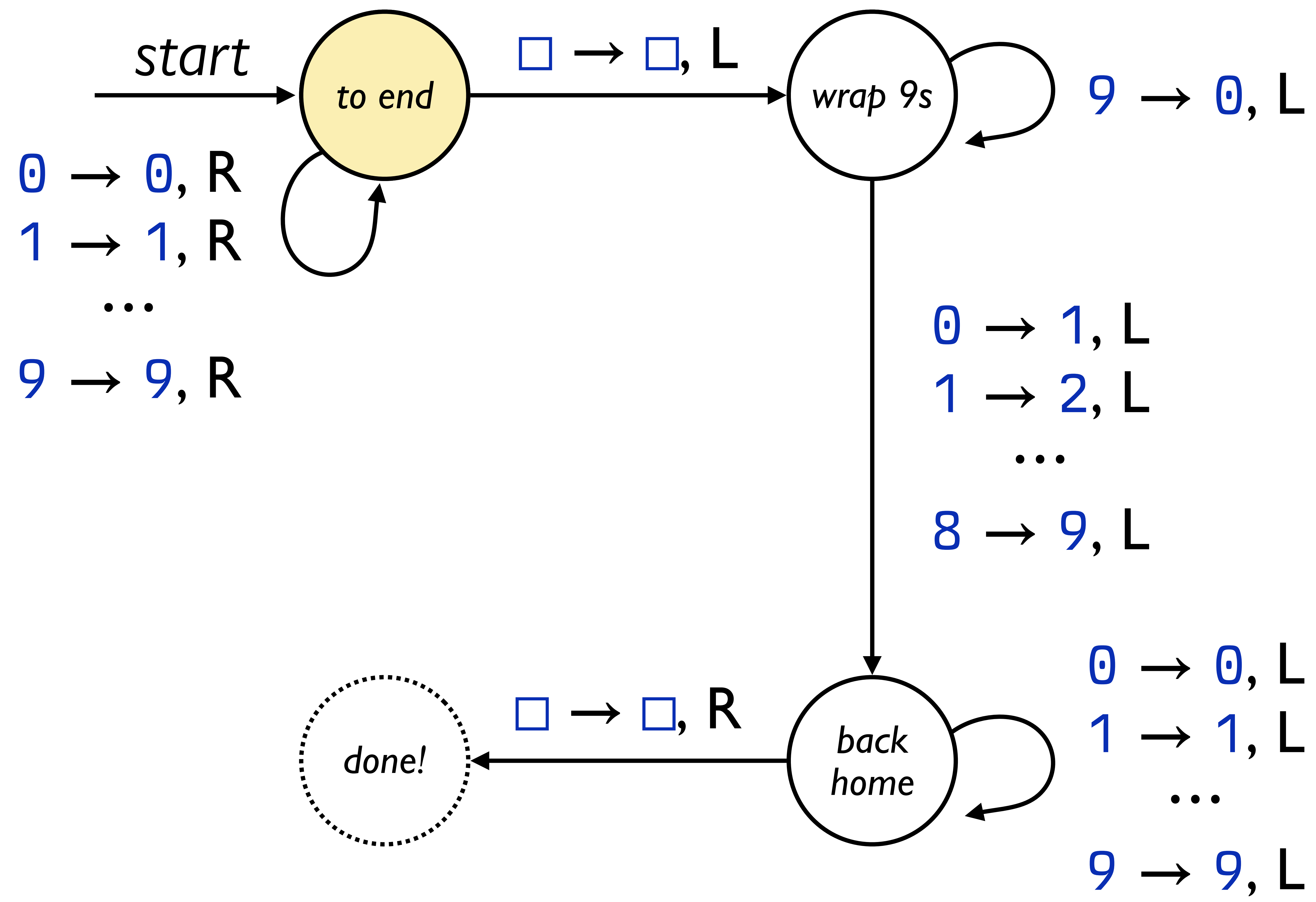

start
to end
□ → □, L
wrap 9s
9 → 0, L
0 → 0, R
1 → 1, R
...
9 → 9, R
0 → 1, L
1 → 2, L
...
8 → 9, L
0 → 0, L
1 → 1, L
...
9 → 9, L
□ → □, R
done!
back home



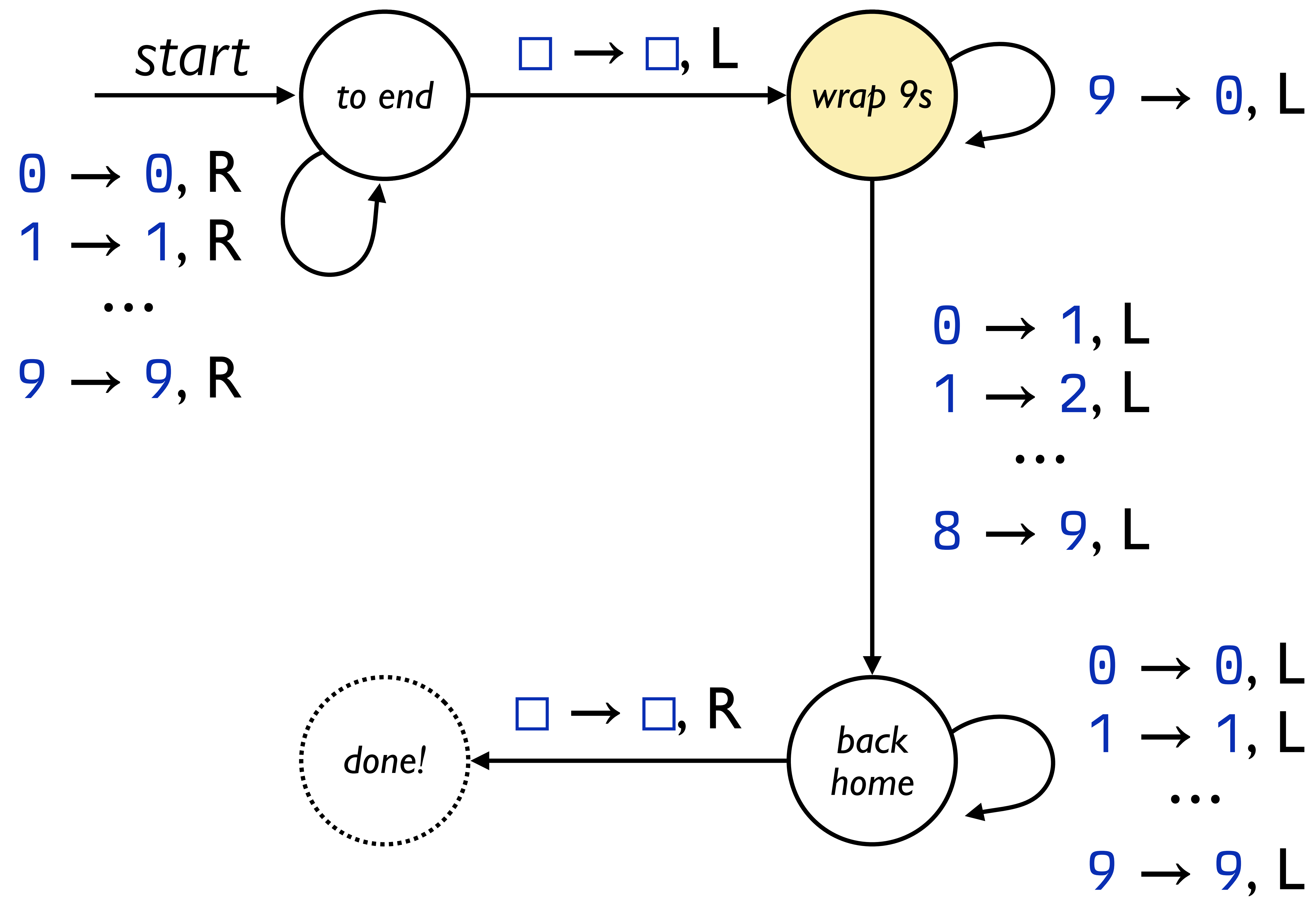
9
9
9


start
to end
□ → □, L
wrap 9s
9 → 0, L
0 → 0, R
1 → 1, R
...
9 → 9, R
0 → 1, L
1 → 2, L
...
8 → 9, L
0 → 0, L
1 → 1, L
...
9 → 9, L
□ → □, R
done!
back home


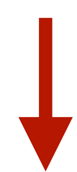
9 9 9


start
to end
0 → 0, R
1 → 1, R
...
9 → 9, R
□ → □, L
wrap 9s
9 → 0, L
0 → 1, L
1 → 2, L
...
8 → 9, L
back home
0 → 0, L
1 → 1, L
...
9 → 9, L
□ → □, R
done!

9
9
9



start
to end
0 → 0, R
1 → 1, R
...
9 → 9, R
□ → □, L
wrap 9s
9 → 0, L
0 → 1, L
1 → 2, L
...
8 → 9, L
back home
0 → 0, L
1 → 1, L
...
9 → 9, L
□ → □, R
done!



9
9
9

start

to end

0 → 0, R
1 → 1, R
…
9 → 9, R

□ → □, L

wrap 9s

9 → 0, L

0 → 1, L
1 → 2, L
…
8 → 9, L

back home

0 → 0, L
1 → 1, L
…
9 → 9, L

□ → □, R

done!

														9	9	0																

start

to end

0 → 0, R
1 → 1, R
...
9 → 9, R

□ → □, L

wrap 9s

9 → 0, L

0 → 1, L
1 → 2, L
...
8 → 9, L

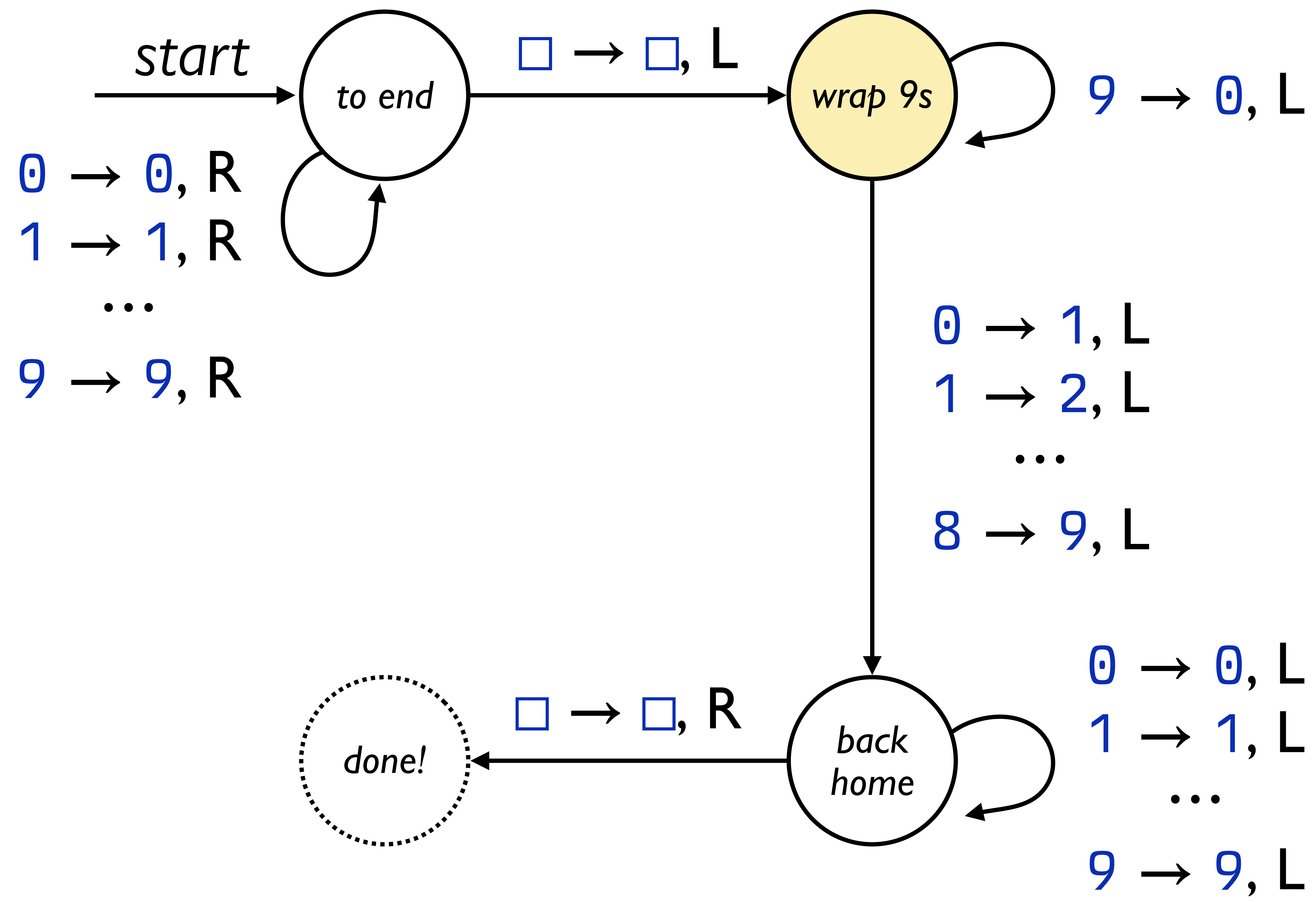
back home

0 → 0, L
1 → 1, L
...
9 → 9, L

□ → □, R

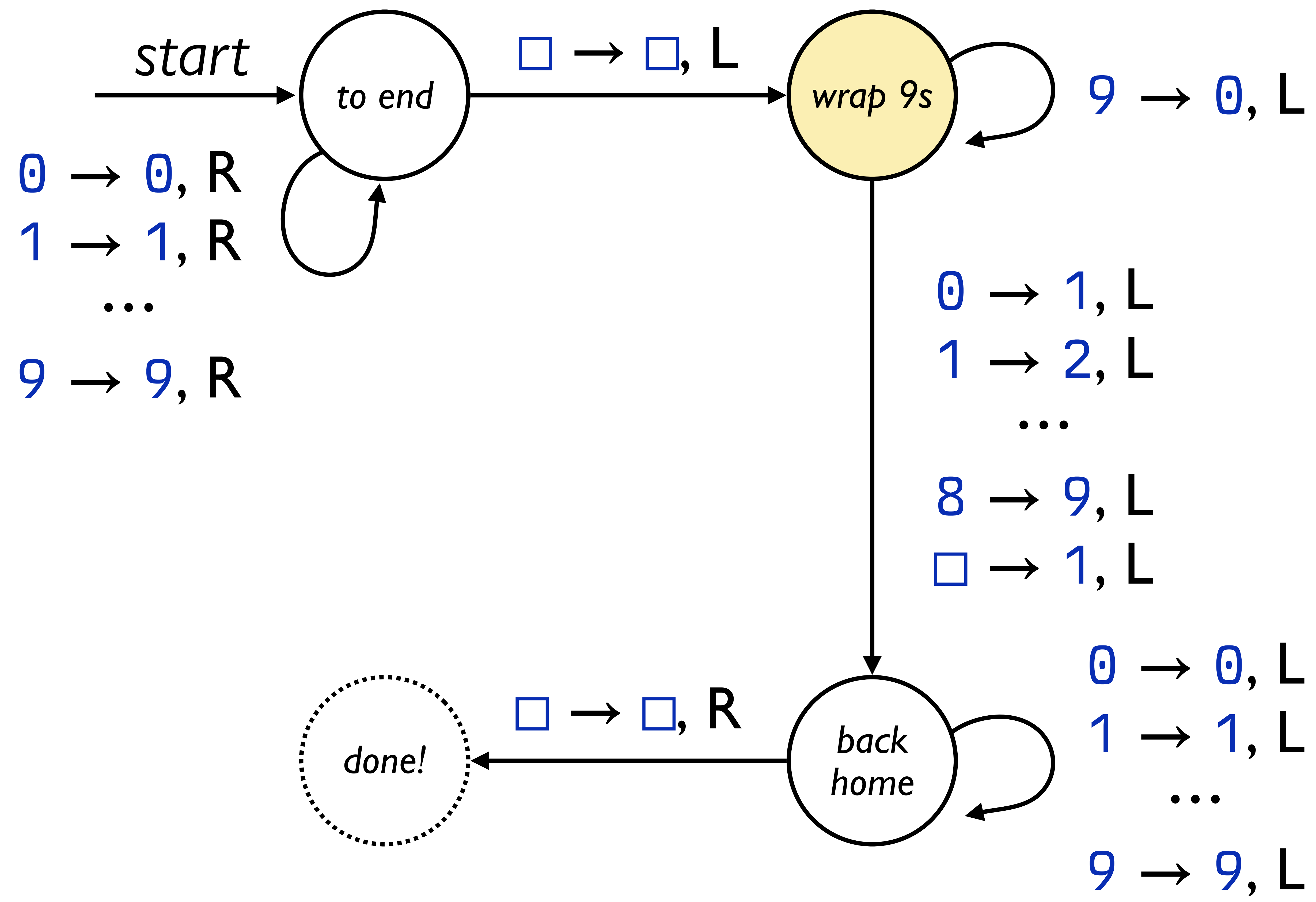
done!

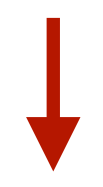
														9	0	0																	



start
to end
□ → □, L
wrap 9s
9 → 0, L
0 → 0, R
1 → 1, R
...
9 → 9, R
0 → 1, L
1 → 2, L
...
8 → 9, L
0 → 0, L
1 → 1, L
...
9 → 9, L
□ → □, R
done!
back home

0 0 0


start
to end
0 → 0, R
1 → 1, R
...
9 → 9, R
□ → □, L
wrap 9s
9 → 0, L
0 → 1, L
1 → 2, L
...
8 → 9, L
□ → 1, L
back home
0 → 0, L
1 → 1, L
...
9 → 9, L
□ → □, R
done!



0 0 0

start

to end

0 → 0, R
1 → 1, R
...
9 → 9, R

□ → □, L

wrap 9s

9 → 0, L

0 → 1, L
1 → 2, L
...
8 → 9, L
□ → 1, L

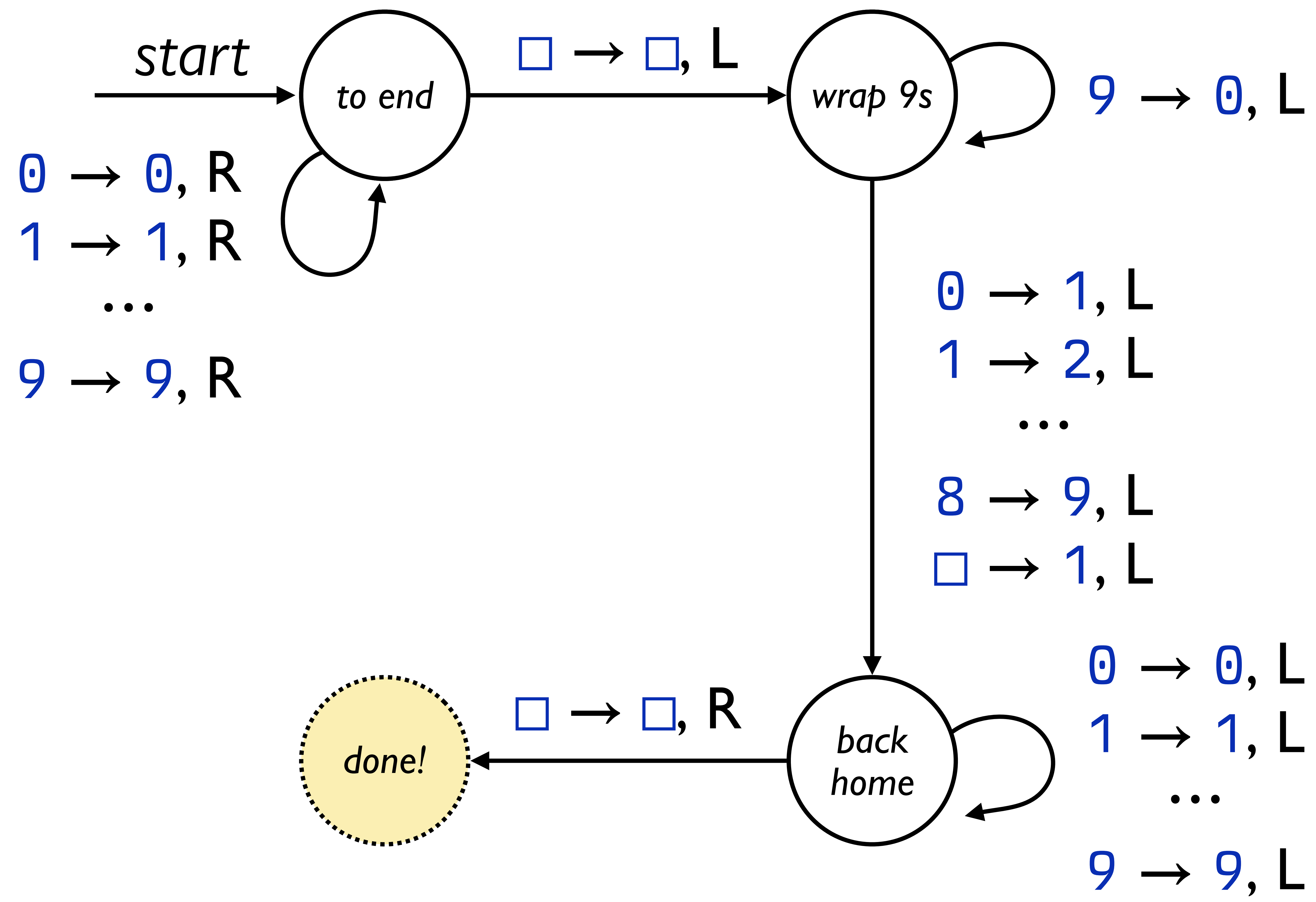
back home

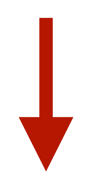
0 → 0, L
1 → 1, L
...
9 → 9, L

□ → □, R

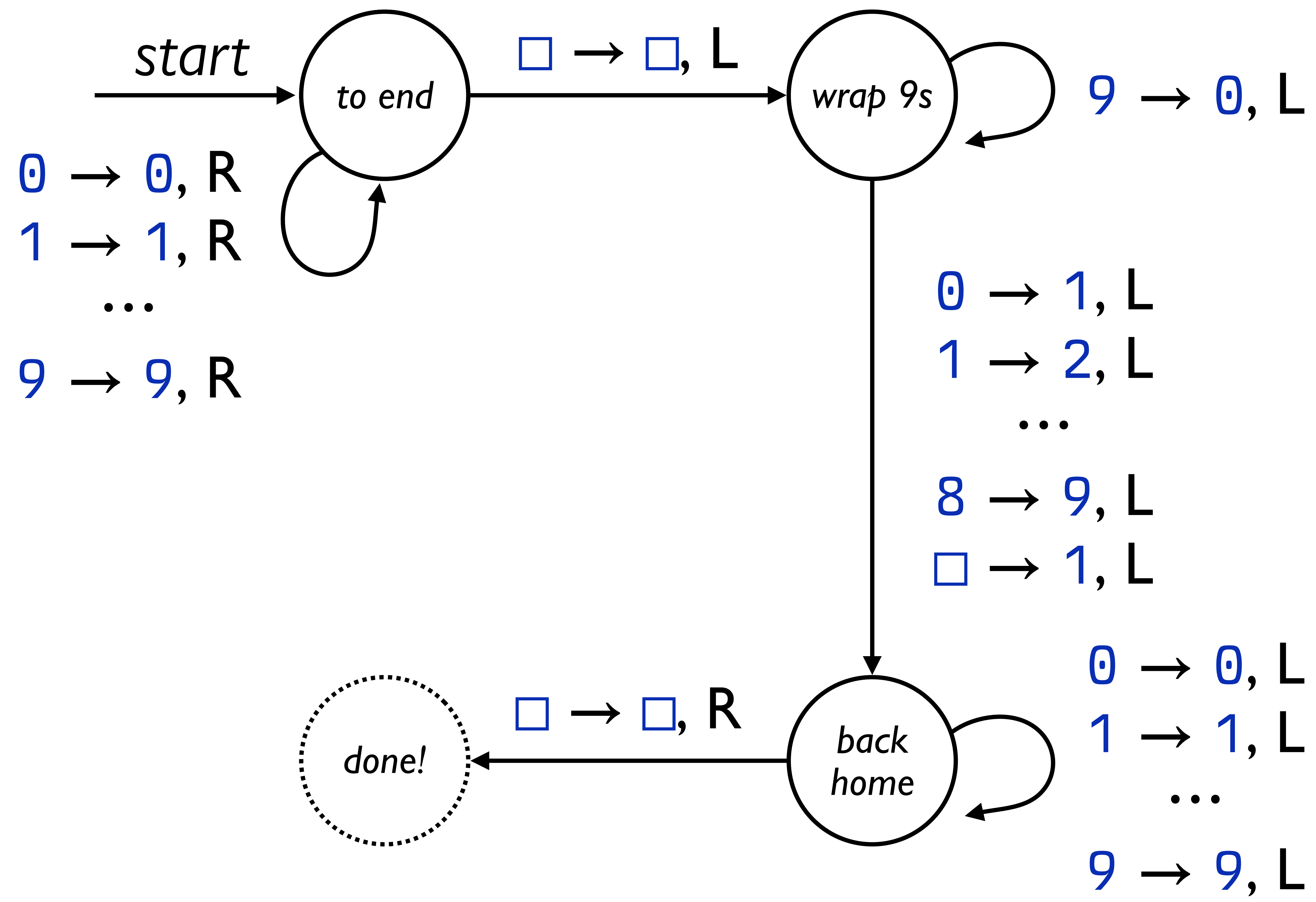
done!

1	0	0	0


start
to end
0 → 0, R
1 → 1, R
...
9 → 9, R
□ → □, L
wrap 9s
9 → 0, L
0 → 1, L
1 → 2, L
...
8 → 9, L
□ → 1, L
back home
0 → 0, L
1 → 1, L
...
9 → 9, L
□ → □, R
done!



1 0 0 0


start
to end
0 → 0, R
1 → 1, R
...
9 → 9, R
□ → □, L
wrap 9s
9 → 0, L
0 → 1, L
1 → 2, L
...
8 → 9, L
□ → 1, L
back home
0 → 0, L
1 → 1, L
...
9 → 9, L
□ → □, R
done!

```
def add(num1, num2):
    while num2 > 0:
        decrement(num2)
        increment(num1)
```

Next!

Decrementing numbers

Now let's build a TM that decrements a number. We'll assume that

- the tape head points at the start of a number
- there's at least one blank on each side of the number

The tape head will end at the start of the number.

If the input is 0, the subroutine should signal an error.

Decrementing numbers

Now let's build a TM that decrements a number. We'll assume that

- the tape head points at the start of a number
- there's at least one blank on each side of the number

The tape head will end at the start of the number.

If the input is 0, the subroutine should signal an error.

Decrementing numbers

Now let's build a TM that decrements a number. We'll assume that

- the tape head points at the start of a number
- there's at least one blank on each side of the number

The tape head will end at the start of the number.

If the input is 0, the subroutine should signal an error.

Decrementing numbers

Now let's build a TM that decrements a number. We'll assume that

- the tape head points at the start of a number
- there's at least one blank on each side of the number

The tape head will end at the start of the number.

If the input is 0, the subroutine should signal an error.

Decrementing numbers

Now let's build a TM that decrements a number. We'll assume that

- the tape head points at the start of a number
- there's at least one blank on each side of the number

The tape head will end at the start of the number.

If the input is 0, the subroutine should signal an error.

Decrementing numbers

Now let's build a TM that decrements a number. We'll assume that

- the tape head points at the start of a number
- there's at least one blank on each side of the number

The tape head will end at the start of the number.

If the input is 0, the subroutine should signal an error.

Decrementing numbers

Now let's build a TM that decrements a number. We'll assume that

- the tape head points at the start of a number
- there's at least one blank on each side of the number

The tape head will end at the start of the number.

If the input is 0, the subroutine should signal an error.

Decrementing numbers

Now let's build a TM that decrements a number. We'll assume that

- the tape head points at the start of a number
- there's at least one blank on each side of the number

The tape head will end at the start of the number.

If the input is 0, the subroutine should signal an error.

Decrementing numbers

Now let's build a TM that decrements a number. We'll assume that

- the tape head points at the start of a number
- there's at least one blank on each side of the number

The tape head will end at the start of the number.

If the input is 0, the subroutine should signal an error.

Decrementing numbers

Now let's build a TM that decrements a number. We'll assume that

- the tape head points at the start of a number
- there's at least one blank on each side of the number

The tape head will end at the start of the number.

If the input is 0, the subroutine should signal an error.

Decrementing numbers

Now let's build a TM that decrements a number. We'll assume that

- the tape head points at the start of a number
- there's at least one blank on each side of the number

The tape head will end at the start of the number.

If the input is 0, the subroutine should signal an error.

Decrementing numbers

Now let's build a TM that decrements a number. We'll assume that

- the tape head points at the start of a number
- there's at least one blank on each side of the number

The tape head will end at the start of the number.

If the input is 0, the subroutine should signal an error.

Decrementing numbers

Now let's build a TM that decrements a number. We'll assume that

- the tape head points at the start of a number
- there's at least one blank on each side of the number

The tape head will end at the start of the number.

If the input is 0, the subroutine should signal an error.

Decrementing numbers

Now let's build a TM that decrements a number. We'll assume that

- the tape head points at the start of a number
- there's at least one blank on each side of the number

The tape head will end at the start of the number.

If the input is 0, the subroutine should signal an error.

														1	0	0																	

Decrementing numbers

Now let's build a TM that decrements a number. We'll assume that

- the tape head points at the start of a number
- there's at least one blank on each side of the number

The tape head will end at the start of the number.

If the input is 0, the subroutine should signal an error.

Decrementing numbers

Now let's build a TM that decrements a number. We'll assume that

- the tape head points at the start of a number
- there's at least one blank on each side of the number

The tape head will end at the start of the number.

If the input is 0, the subroutine should signal an error.

Decrementing numbers

Now let's build a TM that decrements a number. We'll assume that

- the tape head points at the start of a number
- there's at least one blank on each side of the number

The tape head will end at the start of the number.

If the input is 0, the subroutine should signal an error.

Decrementing numbers

Now let's build a TM that decrements a number. We'll assume that

- the tape head points at the start of a number
- there's at least one blank on each side of the number

The tape head will end at the start of the number.

If the input is 0, the subroutine should signal an error.

Decrementing numbers

Now let's build a TM that decrements a number. We'll assume that

- the tape head points at the start of a number
- there's at least one blank on each side of the number

The tape head will end at the start of the number.

If the input is 0, the subroutine should signal an error.

Decrementing numbers

Now let's build a TM that decrements a number. We'll assume that

- the tape head points at the start of a number
- there's at least one blank on each side of the number

The tape head will end at the start of the number.

If the input is 0, the subroutine should signal an error.

Decrementing numbers

Now let's build a TM that decrements a number. We'll assume that

- the tape head points at the start of a number
- there's at least one blank on each side of the number

The tape head will end at the start of the number.

If the input is 0, the subroutine should signal an error.

Decrementing numbers

Now let's build a TM that decrements a number. We'll assume that

- the tape head points at the start of a number
- there's at least one blank on each side of the number

The tape head will end at the start of the number.

If the input is 0, the subroutine should signal an error.

Decrementing numbers

Now let's build a TM that decrements a number. We'll assume that

- the tape head points at the start of a number
- there's at least one blank on each side of the number

The tape head will end at the start of the number.

If the input is 0, the subroutine should signal an error.

Decrementing numbers

Now let's build a TM that decrements a number. We'll assume that

- the tape head points at the start of a number
- there's at least one blank on each side of the number

The tape head will end at the start of the number.

If the input is 0, the subroutine should signal an error.

Decrementing numbers

Now let's build a TM that decrements a number. We'll assume that

- the tape head points at the start of a number
- there's at least one blank on each side of the number

The tape head will end at the start of the number.

If the input is 0, the subroutine should signal an error.

Decrementing numbers

```
def decrement(num):
    go to the end of the number
    if every digit was 0:
        signal that we're done
    while current_digit == 0:
        current_digit = 9
        go left one digit
    current_digit -= 1
    go to the start of the number
```

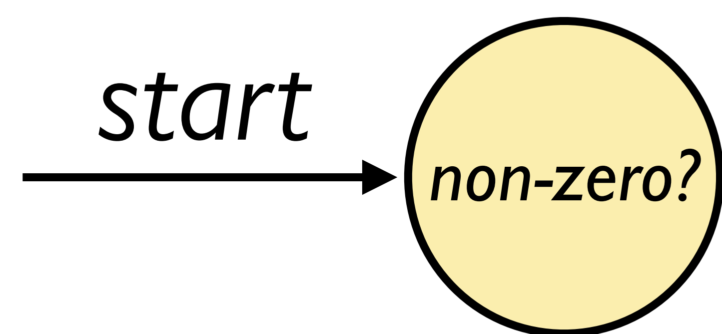
0
2
0
0

start

														0	2	0	0																

start

0	2	0	0


start
non-zero?

0 2 0 0

0 → 0, R

start

non-zero?

0	2	0	0

$0 \rightarrow 0$, R

start

non-zero?

0	2	0	0

1 → 1, R
2 → 2, R
...
9 → 9, R

0 → 0, R

start

non-zero?

0	2	0	0

1 → 1, R
2 → 2, R
...
9 → 9, R

0 → 0, R

start

non-zero?

														0	2	0	0																

to end

1 → 1, R
2 → 2, R
...
9 → 9, R

0 → 0, R

start

non-zero?

0	2	0	0

0 → 0, R
1 → 1, R
...
9 → 9, R

to end

1 → 1, R
2 → 2, R
...
9 → 9, R

0 → 0, R

start

non-zero?

0	2	0	0

0 → 0, R
1 → 1, R
...
9 → 9, R

to end

1 → 1, R
2 → 2, R
...
9 → 9, R

0 → 0, R

start

non-zero?

0	2	0	0

0 → 0, R
1 → 1, R
…
9 → 9, R

to end

1 → 1, R
2 → 2, R
…
9 → 9, R

0 → 0, R

start

non-zero?

0	2	0	0

0 → 0, R
1 → 1, R
...
9 → 9, R
to end
□ → □, L
1 → 1, R
2 → 2, R
...
9 → 9, R
0 → 0, R
start
non-zero?
0 2 0 0

0 → 0, R
1 → 1, R
…
9 → 9, R

to end

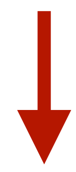
□ → □, L

1 → 1, R
2 → 2, R
…
9 → 9, R

0 → 0, R

start

non-zero?



0 → 0, R
1 → 1, R
...
9 → 9, R

to end

□ → □, L

wrap zeroes

0 → 9, L

1 → 1, R
2 → 2, R
...
9 → 9, R

0 → 0, R

start

non-zero?

0	2	0	0

0 → 0, R
1 → 1, R
...
9 → 9, R

to end

□ → □, L

0 → 9, L

wrap zeroes

1 → 1, R
2 → 2, R
...
9 → 9, R

0 → 0, R

start

non-zero?

0	2	0	9

0 → 0, R
1 → 1, R
...
9 → 9, R

to end

□ → □, L

0 → 9, L

wrap zeroes

1 → 1, R
2 → 2, R
...
9 → 9, R

0 → 0, R

start

non-zero?

0	2	9	9

0 → 0, R
1 → 1, R
...
9 → 9, R

to end

□ → □, L

0 → 9, L

wrap zeroes

1 → 1, R
2 → 2, R
...
9 → 9, R

1 → 0, L
2 → 1, L
...
9 → 8, L

0 → 0, R

start

non-zero?

														0	2	9	9																

0 → 0, R
1 → 1, R
...
9 → 9, R

to end

□ → □, L

wrap zeroes

0 → 9, L

1 → 1, R
2 → 2, R
...
9 → 9, R

1 → 0, L
2 → 1, L
...
9 → 8, L

0 → 0, R

start

non-zero?

back home

0	1	9	9

0 → 0, R
1 → 1, R
…
9 → 9, R

to end

□ → □, L

0 → 9, L

wrap zeroes

1 → 1, R
2 → 2, R
…
9 → 9, R

1 → 0, L
2 → 1, L
…
9 → 8, L

0 → 0, R

start

non-zero?

back home

0 → 0, L
1 → 1, L
…
9 → 9, L

														0	1	9	9																

0 → 0, R
1 → 1, R
…
9 → 9, R

to end

□ → □, L

0 → 9, L

wrap zeroes

1 → 1, R
2 → 2, R
…
9 → 9, R

1 → 0, L
2 → 1, L
…
9 → 8, L

0 → 0, R

start

non-zero?

back home

0 → 0, L
1 → 1, L
…
9 → 9, L

0 → 0, R
1 → 1, R
…
9 → 9, R

to end

□ → □, L

0 → 9, L

wrap zeroes

1 → 1, R
2 → 2, R
…
9 → 9, R

1 → 0, L
2 → 1, L
…
9 → 8, L

0 → 0, R

start

non-zero?

done!

□ → □, R

back home

0 → 0, L
1 → 1, L
…
9 → 9, L

0 → 0, R
1 → 1, R
...
9 → 9, R

to end

□ → □, L

wrap zeroes

0 → 9, L

1 → 1, R
2 → 2, R
...
9 → 9, R

1 → 0, L
2 → 1, L
...
9 → 8, L

0 → 0, R

start

non-zero?

done!

□ → □, R

back home

0 → 0, L
1 → 1, L
...
9 → 9, L

0	1	9	9

0 → 0, R
1 → 1, R
...
9 → 9, R

to end

□ → □, L

0 → 9, L

wrap zeroes

1 → 1, R
2 → 2, R
...
9 → 9, R

1 → 0, L
2 → 1, L
...
9 → 8, L

0 → 0, R

start

non-zero?

done!

□ → □, R

back home

0 → 0, L
1 → 1, L
...
9 → 9, L

0 → 0, R
1 → 1, R
...
9 → 9, R

to end

□ → □, L

0 → 9, L

wrap zeroes

1 → 1, R
2 → 2, R
...
9 → 9, R

1 → 0, L
2 → 1, L
...
9 → 8, L

0 → 0, R

start

non-zero?

done!

□ → □, R

back home

0 → 0, L
1 → 1, L
...
9 → 9, L

														0	1	9	9																

0 → 0, R
1 → 1, R
...
9 → 9, R

to end

□ → □, L

0 → 9, L

wrap zeroes

1 → 1, R
2 → 2, R
...
9 → 9, R

1 → 0, L
2 → 1, L
...
9 → 8, L

0 → 0, R

start

non-zero?

done!

□ → □, R

back home

0 → 0, L
1 → 1, L
...
9 → 9, L

0	1	9	9

0 → 0, R
1 → 1, R
...
9 → 9, R

to end

□ → □, L

wrap zeroes

0 → 9, L

1 → 1, R
2 → 2, R
...
9 → 9, R

1 → 0, L
2 → 1, L
...
9 → 8, L

0 → 0, R

start

non-zero?

done!

□ → □, R

back home

0 → 0, L
1 → 1, L
...
9 → 9, L

0	1	9	9

0 → 0, R
1 → 1, R
…
9 → 9, R

to end

□ → □, L

0 → 9, L

wrap zeroes

1 → 1, R
2 → 2, R
…
9 → 9, R

1 → 0, L
2 → 1, L
…
9 → 8, L

0 → 0, R

start

non-zero?

done!

□ → □, R

back home

0 → 0, L
1 → 1, L
…
9 → 9, L

0 → 0, R
1 → 1, R
...
9 → 9, R

to end

□ → □, L

0 → 9, L

wrap zeroes

1 → 1, R
2 → 2, R
...
9 → 9, R

1 → 0, L
2 → 1, L
...
9 → 8, L

0 → 0, R

start

non-zero?

done!

□ → □, R

back home

0 → 0, L
1 → 1, L
...
9 → 9, L

0 → 0, R
1 → 1, R
...
9 → 9, R

to end

□ → □, L

0 → 9, L

wrap zeroes

1 → 1, R
2 → 2, R
...
9 → 9, R

1 → 0, L
2 → 1, L
...
9 → 8, L

0 → 0, R

start

non-zero?

done!

□ → □, R

back home

0 → 0, L
1 → 1, L
...
9 → 9, L

														0	1	9	9																

0 → 0, R
1 → 1, R
…
9 → 9, R

to end

□ → □, L

0 → 9, L

wrap zeroes

1 → 1, R
2 → 2, R
…
9 → 9, R

1 → 0, L
2 → 1, L
…
9 → 8, L

0 → 0, R

start

non-zero?

done!

□ → □, R

back home

0 → 0, L
1 → 1, L
…
9 → 9, L

0	1	9	9

0 → 0, R
1 → 1, R
...
9 → 9, R

to end

□ → □, L

wrap zeroes

0 → 9, L

1 → 1, R
2 → 2, R
...
9 → 9, R

1 → 0, L
2 → 1, L
...
9 → 8, L

0 → 0, R

start

non-zero?

done!

□ → □, R

back home

0 → 0, L
1 → 1, L
...
9 → 9, L

0	1	9	8

0 → 0, R
1 → 1, R
…
9 → 9, R

to end

□ → □, L

0 → 9, L

wrap zeroes

1 → 1, R
2 → 2, R
…
9 → 9, R

1 → 0, L
2 → 1, L
…
9 → 8, L

0 → 0, R

start

non-zero?

done!

□ → □, R

back home

0 → 0, L
1 → 1, L
…
9 → 9, L

0 → 0, R
1 → 1, R
...
9 → 9, R

to end

□ → □, L

0 → 9, L

wrap zeroes

1 → 1, R
2 → 2, R
...
9 → 9, R

1 → 0, L
2 → 1, L
...
9 → 8, L

0 → 0, R

start

non-zero?

done!

□ → □, R

back home

0 → 0, L
1 → 1, L
...
9 → 9, L

0	1	9	8

0 → 0, R
1 → 1, R
...
9 → 9, R

to end

□ → □, L

wrap zeroes

0 → 9, L

1 → 1, R
2 → 2, R
...
9 → 9, R

1 → 0, L
2 → 1, L
...
9 → 8, L

0 → 0, R

start

non-zero?

done!

□ → □, R

back home

0 → 0, L
1 → 1, L
...
9 → 9, L

														0	1	9	8																

0 → 0, R
1 → 1, R
...
9 → 9, R

to end

□ → □, L

wrap zeroes

0 → 9, L

1 → 1, R
2 → 2, R
...
9 → 9, R

1 → 0, L
2 → 1, L
...
9 → 8, L

0 → 0, R

start

non-zero?

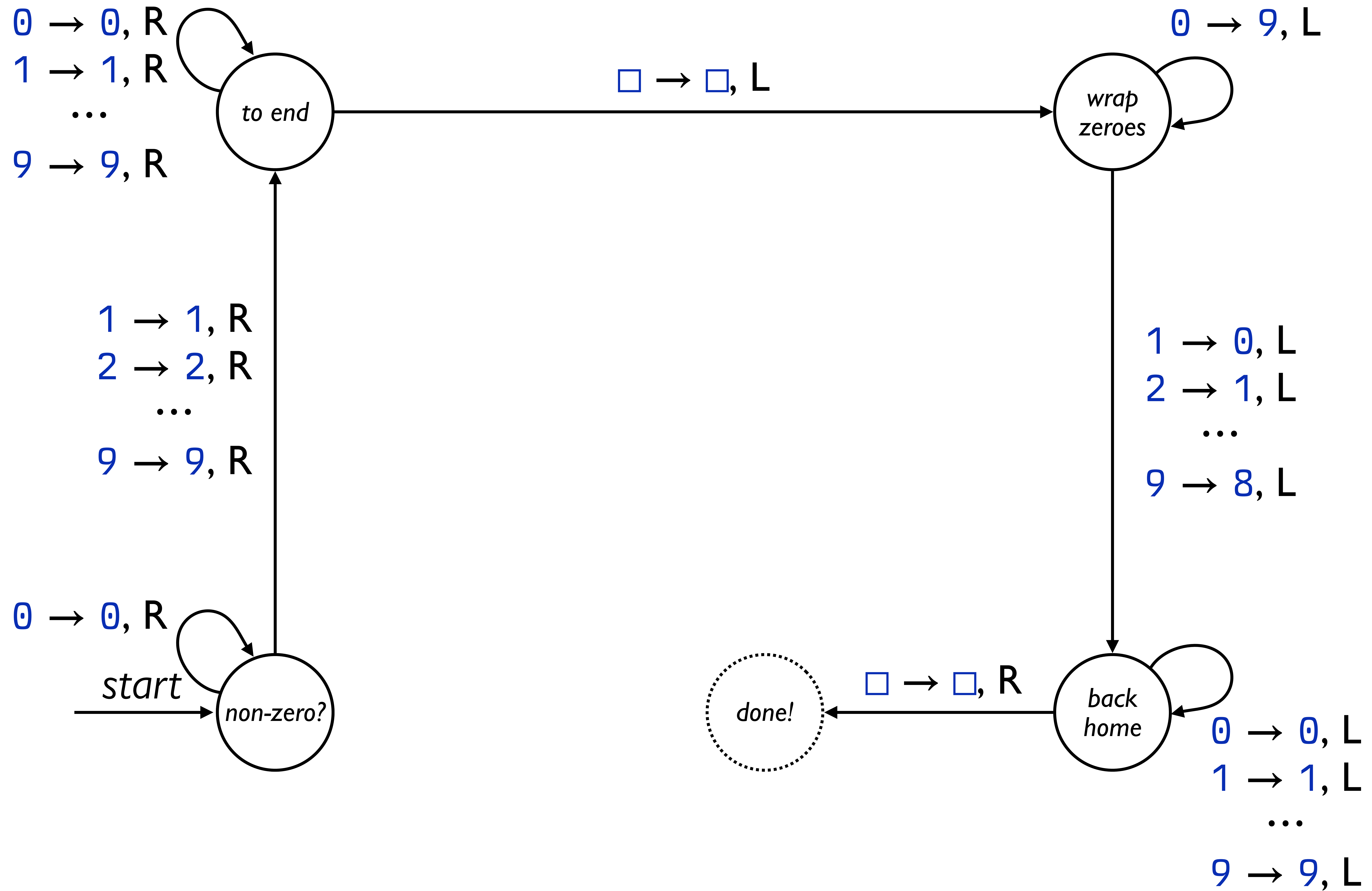
done!

□ → □, R

back home

0 → 0, L
1 → 1, L
...
9 → 9, L

0	1	9	8


0 → 0, R
1 → 1, R
...
9 → 9, R
to end
□ → □, L
wrap zeroes
0 → 9, L
1 → 1, R
2 → 2, R
...
9 → 9, R
1 → 0, L
2 → 1, L
...
9 → 8, L
0 → 0, R
start
non-zero?
done!
□ → □, R
back home
0 → 0, L
1 → 1, L
...
9 → 9, L

0 → 0, R
1 → 1, R
...
9 → 9, R

to end

□ → □, L

0 → 9, L

wrap zeroes

1 → 1, R
2 → 2, R
...
9 → 9, R

1 → 0, L
2 → 1, L
...
9 → 8, L

0 → 0, R

start

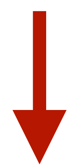
non-zero?

done!

□ → □, R

back home

0 → 0, L
1 → 1, L
...
9 → 9, L



0 → 0, R
1 → 1, R
…
9 → 9, R

to end

□ → □, L

0 → 9, L

wrap zeroes

1 → 1, R
2 → 2, R
…
9 → 9, R

1 → 0, L
2 → 1, L
…
9 → 8, L

0 → 0, R

start

non-zero?

done!

□ → □, R

back home

0 → 0, L
1 → 1, L
…
9 → 9, L

0 → 0, R
1 → 1, R
...
9 → 9, R

to end

□ → □, L

0 → 9, L

wrap zeroes

1 → 1, R
2 → 2, R
...
9 → 9, R

1 → 0, L
2 → 1, L
...
9 → 8, L

0 → 0, R

start

non-zero?

done!

□ → □, R

back home

0 → 0, L
1 → 1, L
...
9 → 9, L

0 → 0, R
1 → 1, R
…
9 → 9, R

to end

□ → □, L

0 → 9, L

wrap zeroes

1 → 0, L
2 → 1, L
…
9 → 8, L

1 → 1, R
2 → 2, R
…
9 → 9, R

0 → 0, R

start

non-zero?

done!

□ → □, R

back home

0 → 0, L
1 → 1, L
…
9 → 9, L

0 → 0, R
1 → 1, R
...
9 → 9, R

to end

□ → □, L

0 → 9, L

wrap zeroes

1 → 1, R
2 → 2, R
...
9 → 9, R

1 → 0, L
2 → 1, L
...
9 → 8, L

0 → 0, R

start

non-zero?

done!

□ → □, R

back home

0 → 0, L
1 → 1, L
...
9 → 9, L

0 → 0, R
1 → 1, R
...
9 → 9, R

to end

□ → □, L

0 → 9, L

wrap zeroes

1 → 1, R
2 → 2, R
...
9 → 9, R

1 → 0, L
2 → 1, L
...
9 → 8, L

0 → 0, R

start

non-zero?

done!

□ → □, R

back home

0 → 0, L
1 → 1, L
...
9 → 9, L

0 → 0, R
1 → 1, R
…
9 → 9, R

to end

□ → □, L

0 → 9, L

wrap zeroes

1 → 1, R
2 → 2, R
…
9 → 9, R

1 → 0, L
2 → 1, L
…
9 → 8, L

0 → 0, R

start

non-zero?

□ → □, L

done!

□ → □, R

back home

0 → 0, L
1 → 1, L
…
9 → 9, L

0 → 0, R
1 → 1, R
…
9 → 9, R

to end

□ → □, L

wrap zeroes

0 → 9, L

1 → 1, R
2 → 2, R
…
9 → 9, R

1 → 0, L
2 → 1, L
…
9 → 8, L

0 → 0, R

start

non-zero?

□ → □, L

done!

□ → □, R

back home

0 → 0, L
1 → 1, L
…
9 → 9, L

0 → 0, R
1 → 1, R
...
9 → 9, R

to end

□ → □, L

0 → 9, L

wrap zeroes

1 → 1, R
2 → 2, R
...
9 → 9, R

1 → 0, L
2 → 1, L
...
9 → 8, L

0 → 0, R

start

non-zero?

□ → □, L

back home

done!

□ → □, R

back home

0 → 0, L
1 → 1, L
...
9 → 9, L

														0	0	0	0																

0 → 0, R
1 → 1, R
…
9 → 9, R

to end

□ → □, L

0 → 9, L

wrap zeroes

1 → 1, R
2 → 2, R
…
9 → 9, R

1 → 0, L
2 → 1, L
…
9 → 8, L

0 → 0, R

0 → 0, L

start

non-zero?

□ → □, L

back home

done!

□ → □, R

back home

0 → 0, L
1 → 1, L
…
9 → 9, L

0 → 0, R
1 → 1, R
...
9 → 9, R

to end

□ → □, L

0 → 9, L

wrap zeroes

1 → 1, R
2 → 2, R
...
9 → 9, R

1 → 0, L
2 → 1, L
...
9 → 8, L

0 → 0, L

0 → 0, R

start

non-zero?

□ → □, L

back home

done!

□ → □, R

back home

0 → 0, L
1 → 1, L
...
9 → 9, L

0 → 0, R
1 → 1, R
…
9 → 9, R

to end

□ → □, L

wrap zeroes

0 → 9, L

1 → 1, R
2 → 2, R
…
9 → 9, R

1 → 0, L
2 → 1, L
…
9 → 8, L

0 → 0, R

start

non-zero?

□ → □, L

0 → 0, L

back home

done!

□ → □, R

back home

0 → 0, L
1 → 1, L
…
9 → 9, L

0 → 0, R
1 → 1, R
…
9 → 9, R

to end

□ → □, L

wrap zeroes

0 → 9, L

1 → 1, R
2 → 2, R
…
9 → 9, R

1 → 0, L
2 → 1, L
…
9 → 8, L

0 → 0, R

start

non-zero?

□ → □, L

0 → 0, L

back home

done!

□ → □, R

back home

0 → 0, L
1 → 1, L
…
9 → 9, L

0 → 0, R
1 → 1, R
…
9 → 9, R

to end

□ → □, L

0 → 9, L

wrap zeroes

1 → 1, R
2 → 2, R
…
9 → 9, R

1 → 0, L
2 → 1, L
…
9 → 8, L

0 → 0, L

0 → 0, R

start

non-zero?

□ → □, L

back home

done!

□ → □, R

back home

0 → 0, L
1 → 1, L
…
9 → 9, L

0 → 0, R
1 → 1, R
...
9 → 9, R

to end

□ → □, L

0 → 9, L

wrap zeroes

1 → 1, R
2 → 2, R
...
9 → 9, R

1 → 0, L
2 → 1, L
...
9 → 8, L

0 → 0, L

0 → 0, R

start

non-zero?

□ → □, L

back home

done!

□ → □, R

back home

0 → 0, L
1 → 1, L
...
9 → 9, L

0 → 0, R
1 → 1, R
...
9 → 9, R

to end

□ → □, L

0 → 9, L

wrap zeroes

1 → 1, R
2 → 2, R
...
9 → 9, R

1 → 0, L
2 → 1, L
...
9 → 8, L

0 → 0, L

□ → □, R

0 → 0, R

start

non-zero?

□ → □, L

back home

done!

□ → □, R

back home

0 → 0, L
1 → 1, L
...
9 → 9, L

0 → 0, R
1 → 1, R
…
9 → 9, R

to end

□ → □, L

0 → 9, L

wrap zeroes

1 → 1, R
2 → 2, R
…
9 → 9, R

1 → 0, L
2 → 1, L
…
9 → 8, L

0 → 0, L

□ → □, R

0 → 0, R

start

non-zero?

□ → □, L

back home

done!

□ → □, R

back home

0 → 0, L
1 → 1, L
…
9 → 9, L

0 → 0, R
1 → 1, R
…
9 → 9, R

to end

□ → □, L

wrap zeroes

0 → 9, L

1 → 1, R
2 → 2, R
…
9 → 9, R

1 → 0, L
2 → 1, L
…
9 → 8, L

n = 0

0 → 0, L

□ → □, R

0 → 0, R

start

non-zero?

□ → □, L

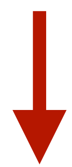
back home

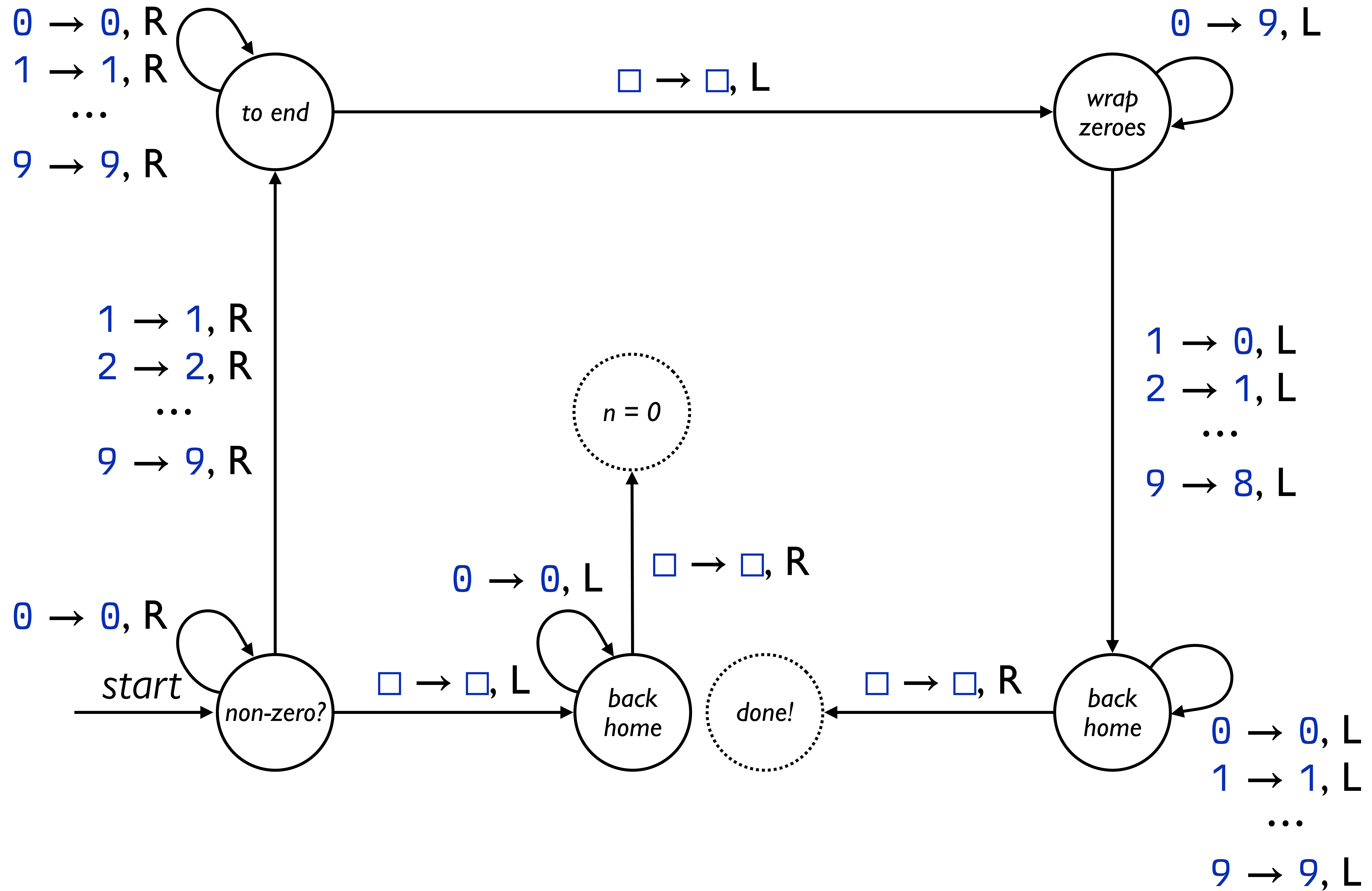
done!

□ → □, R

back home

0 → 0, L
1 → 1, L
…
9 → 9, L




0 → 0, R
1 → 1, R
...
9 → 9, R
to end
□ → □, L
0 → 9, L
wrap zeroes
1 → 1, R
2 → 2, R
...
9 → 9, R
1 → 0, L
2 → 1, L
...
9 → 8, L
n = 0
□ → □, R
0 → 0, L
0 → 0, R
start
non-zero?
□ → □, L
back home
done!
□ → □, R
back home
0 → 0, L
1 → 1, L
...
9 → 9, L

Sometimes a subroutine needs to report back some information about what happened.

Just as a function can return multiple different values, we can have subroutines to have different "done" states.

Each state can then be wired to a different state, so a Turing machine using the subroutine can control what happens next.

Putting it all together

Our goal is to build a Turing machine that, given two numbers, adds those numbers together.

1	3	7		4	2

↓

1	7	9		0	0

```python
def add(num1, num2):
    while num2 > 0:
        decrement(num2)
        increment(num1)
```

Need to make the loop

Using our subroutines

We'll build our new machine using our existing increment and decrement subroutines:

														1	3	7		4	2														

Using our subroutines

We'll build our new machine using our existing increment and decrement subroutines:

													1	3	7		4	2															

Using our subroutines

We'll build our new machine using our existing increment and decrement subroutines:

1	3	7		4	2

Using our subroutines

We'll build our new machine using our existing increment and decrement subroutines:

1	3	7		4	2

Using our subroutines

We'll build our new machine using our existing increment and decrement subroutines:

1	3	7		4	2

Using our subroutines

We'll build our new machine using our existing increment and decrement subroutines:

														1	3	7		4	2														

Using our subroutines

We'll build our new machine using our existing increment and decrement subroutines:

1	3	7		4	2	

Using our subroutines

We'll build our new machine using our existing increment and decrement subroutines:

														1	3	7		4	2														

Using our subroutines

We'll build our new machine using our existing increment and decrement subroutines:

														1	3	7		4	1														

Using our subroutines

We'll build our new machine using our existing increment and decrement subroutines:

														1	3	7		4	1															

Using our subroutines

We'll build our new machine using our existing increment and decrement subroutines:

														1	3	7		4	1														

Using our subroutines

We'll build our new machine using our existing increment and decrement subroutines:

													1	3	7		4	1															

Using our subroutines

We'll build our new machine using our existing increment and decrement subroutines:

														1	3	7		4	1														

Using our subroutines

We'll build our new machine using our existing increment and decrement subroutines:

Using our subroutines

We'll build our new machine using our existing increment and decrement subroutines:

													1	3	7		4	1															

Using our subroutines

We'll build our new machine using our existing increment and decrement subroutines:

1	3	7		4	1

Using our subroutines

We'll build our new machine using our existing increment and decrement subroutines:

														1	3	7		4	1														

Using our subroutines

We'll build our new machine using our existing increment and decrement subroutines:

														1	3	7		4	1														

Using our subroutines

We'll build our new machine using our existing increment and decrement subroutines:

													1	3	7		4	1															

Using our subroutines

We'll build our new machine using our existing increment and decrement subroutines:

													1	3	8		4	1															

Using our subroutines

We'll build our new machine using our existing increment and decrement subroutines:

1	3	8		4	1

Using our subroutines

We'll build our new machine using our existing increment and decrement subroutines:

Using our subroutines

We'll build our new machine using our existing increment and decrement subroutines:

1	3	8		4	1

1	3	7		4	2

start

														1	3	7		4	2														

start

1	3	7		4	2

start

to 2nd num.

1	3	7		4	2

start

to 2nd num.

0 → 0, R
1 → 1, R
...
9 → 9, R

														1	3	7		4	2														

start

to 2nd num.

0 → 0, R
1 → 1, R
...
9 → 9, R

1	3	7		4	2

start

to 2nd num.

0 → 0, R
1 → 1, R
...
9 → 9, R

1	3	7		4	2

start

to 2nd num.

0 → 0, R
1 → 1, R
…
9 → 9, R

1	3	7		4	2

start
to 2nd num.
□ → □, R
0 → 0, R
1 → 1, R
...
9 → 9, R
1 3 7 4 2

start
to 2nd num.
□ → □, R
0 → 0, R
1 → 1, R
...
9 → 9, R
1 3 7 4 2

start

to 2nd num.

□ → □, R

decr.

0 → 0, R
1 → 1, R
...
9 → 9, R

														1	3	7		4	2														

start

to 2nd num.

□ → □, R

decr.

0 → 0, R
1 → 1, R
...
9 → 9, R

1	3	7		4	2	

start

to 2nd num.

□ → □, R

decr.

0 → 0, R
1 → 1, R
...
9 → 9, R

1	3	7		4	2	

start

to 2nd num.

□ → □, R

decr.

0 → 0, R
1 → 1, R
...
9 → 9, R

1	3	7		4	2	

start

to 2nd num.

□ → □, R

decr.

0 → 0, R
1 → 1, R
...
9 → 9, R

1	3	7		4	2	

start
to 2nd num.
□ → □, R
decr.
0 → 0, R
1 → 1, R
...
9 → 9, R
1 3 7 4 1

start

to 2nd num.

□ → □, R

decr.

0 → 0, R
1 → 1, R
...
9 → 9, R

														1	3	7		4	1														

start
to 2nd num.
□ → □, R
decr.
0 → 0, R
1 → 1, R
...
9 → 9, R
1
3
7
4
1

start

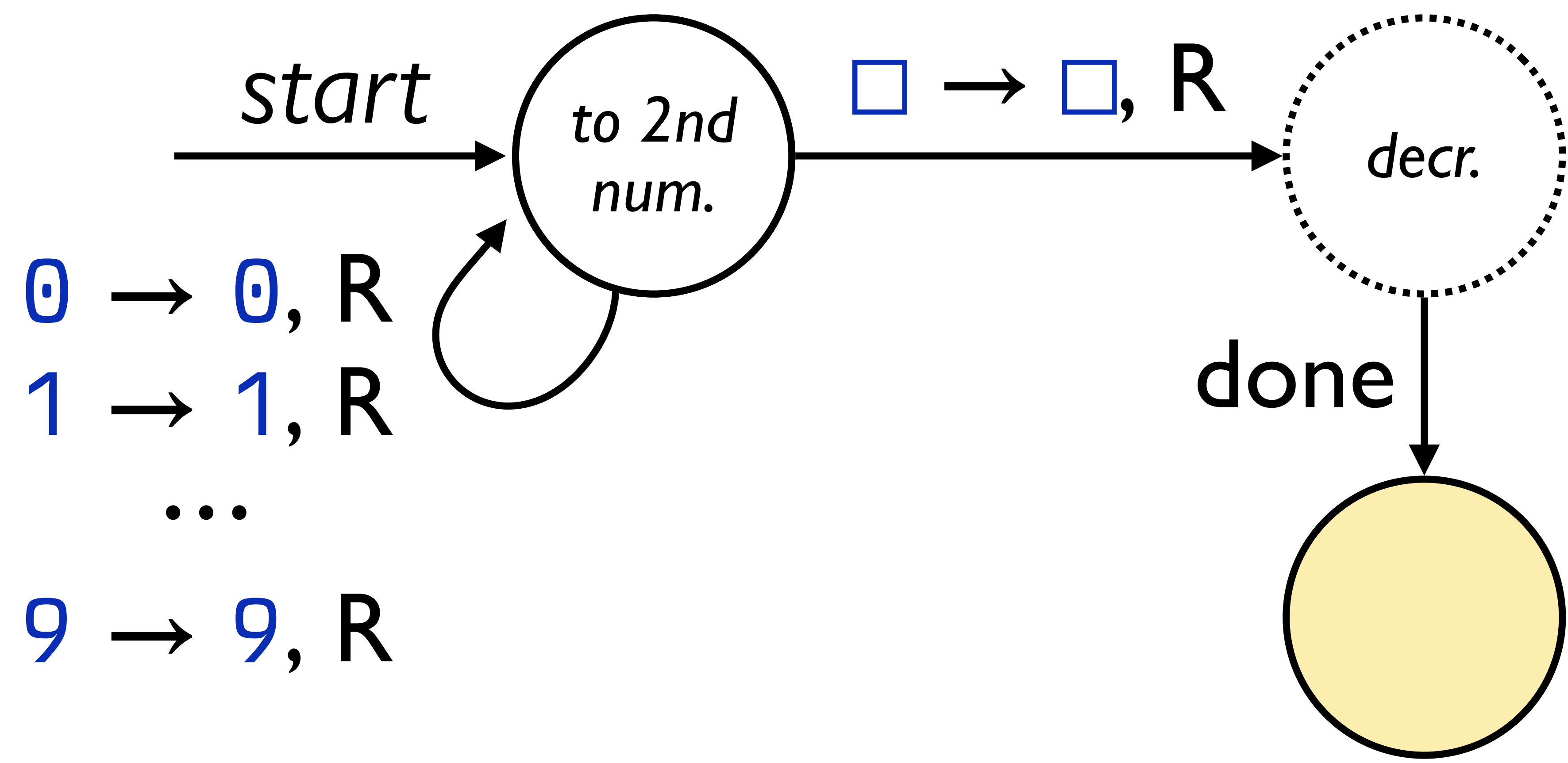
to 2nd num.

□ → □, R

decr.

0 → 0, R
1 → 1, R
...
9 → 9, R

														1	3	7		4	1														



start
to 2nd num.
□ → □, R
decr.
0 → 0, R
1 → 1, R
…
9 → 9, R
done

1 3 7 4 1

start

to 2nd num.

□ → □, R

decr.

0 → 0, R
1 → 1, R
...
9 → 9, R

done

to 1st num.

														1	3	7		4	1														

start

to 2nd num.

0 → 0, R
1 → 1, R
...
9 → 9, R

□ → □, R

decr.

done

to 1st num.

0 → 0, L
1 → 1, L
...
9 → 9, L

														1	3	7		4	1														

start

to 2nd num.

0 → 0, R
1 → 1, R
…
9 → 9, R

□ → □, R

decr.

done

to 1st num.

0 → 0, L
1 → 1, L
…
9 → 9, L

														1	3	7		4	1														

start
to 2nd num.
□ → □, R
decr.
0 → 0, R
1 → 1, R
...
9 → 9, R
done
0 → 0, L
1 → 1, L
...
9 → 9, L
to 1st num.
□ → □, L
1
3
7
4
1

start

to 2nd num.

0 → 0, R
1 → 1, R
...
9 → 9, R

□ → □, R

decr.

done

to 1st num.

0 → 0, L
1 → 1, L
...
9 → 9, L

□ → □, L

														1	3	7		4	1														

start
to 2nd num.
0 → 0, R
1 → 1, R
…
9 → 9, R
□ → □, R
decr.
done
to 1st num.
0 → 0, L
1 → 1, L
…
9 → 9, L
□ → □, L
go home
1 3 7 4 1

start

to 2nd num.

0 → 0, R
1 → 1, R
…
9 → 9, R

□ → □, R

decr.

done

to 1st num.

0 → 0, L
1 → 1, L
…
9 → 9, L

□ → □, L

go home

0 → 0, L
1 → 1, L
…
9 → 9, L

														1	3	7		4	1														

start

to 2nd num.

0 → 0, R
1 → 1, R
...
9 → 9, R

□ → □, R

decr.

done

to 1st num.

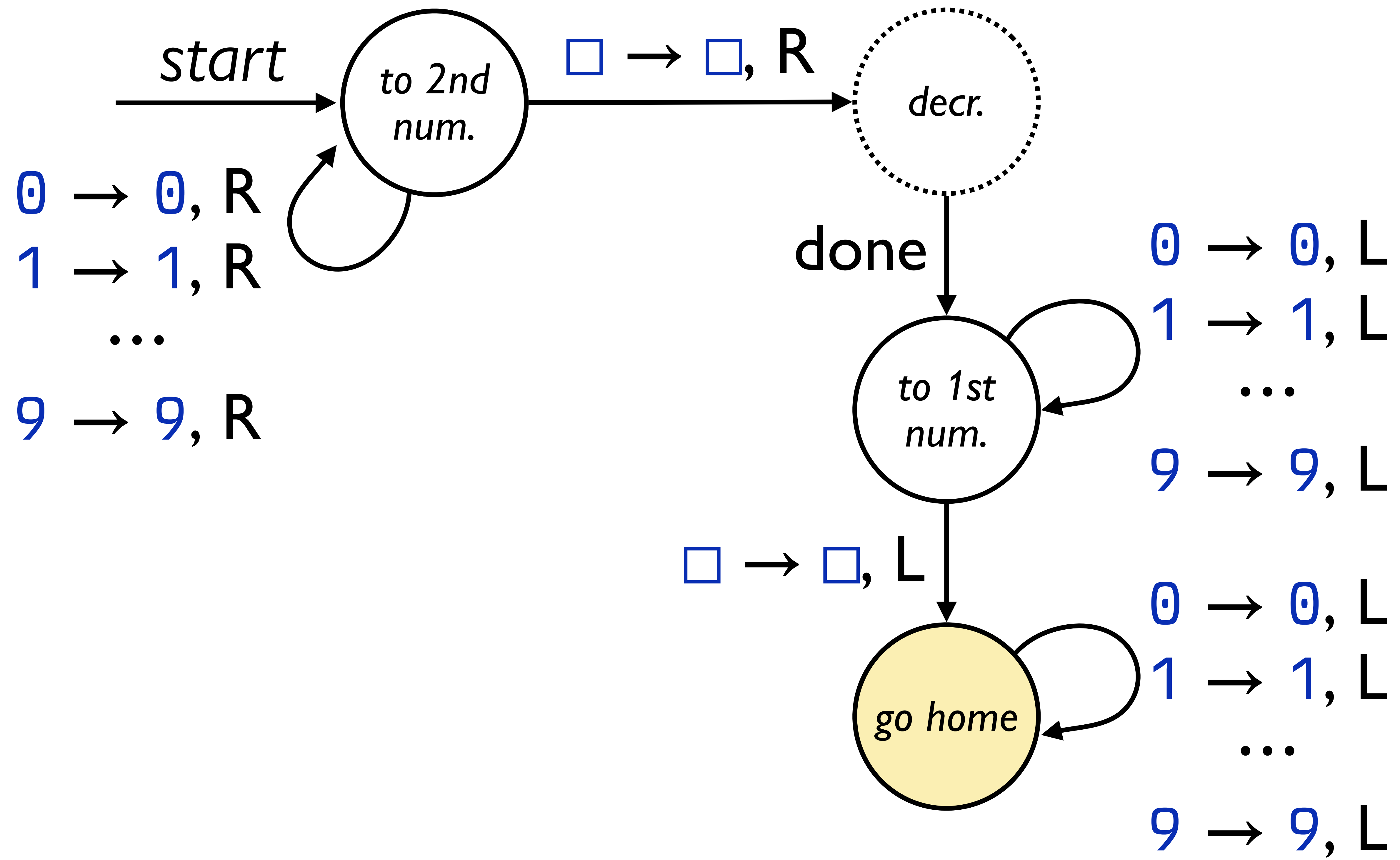
0 → 0, L
1 → 1, L
...
9 → 9, L

□ → □, L

go home

0 → 0, L
1 → 1, L
...
9 → 9, L

														1	3	7		4	1														


start
to 2nd num.
□ → □, R
decr.
0 → 0, R
1 → 1, R
...
9 → 9, R
done
to 1st num.
0 → 0, L
1 → 1, L
...
9 → 9, L
□ → □, L
go home
0 → 0, L
1 → 1, L
...
9 → 9, L

1 3 7 4 1

start

to 2nd num.

0 → 0, R
1 → 1, R
…
9 → 9, R

□ → □, R

decr.

done

to 1st num.

0 → 0, L
1 → 1, L
…
9 → 9, L

□ → □, L

go home

0 → 0, L
1 → 1, L
…
9 → 9, L

														1	3	7		4	1														

start

to 2nd num.

0 → 0, R
1 → 1, R
...
9 → 9, R

□ → □, R

decr.

done

to 1st num.

0 → 0, L
1 → 1, L
...
9 → 9, L

□ → □, L

go home

0 → 0, L
1 → 1, L
...
9 → 9, L

□ → □, R

														1	3	7		4	1														

start

to 2nd num.

0 → 0, R
1 → 1, R
…
9 → 9, R

□ → □, R

decr.

done

to 1st num.

0 → 0, L
1 → 1, L
…
9 → 9, L

□ → □, L

go home

0 → 0, L
1 → 1, L
…
9 → 9, L

□ → □, R

														1	3	7		4	1													

start

to 2nd num.

□ → □, R

decr.

0 → 0, R
1 → 1, R
...
9 → 9, R

done

to 1st num.

0 → 0, L
1 → 1, L
...
9 → 9, L

□ → □, L

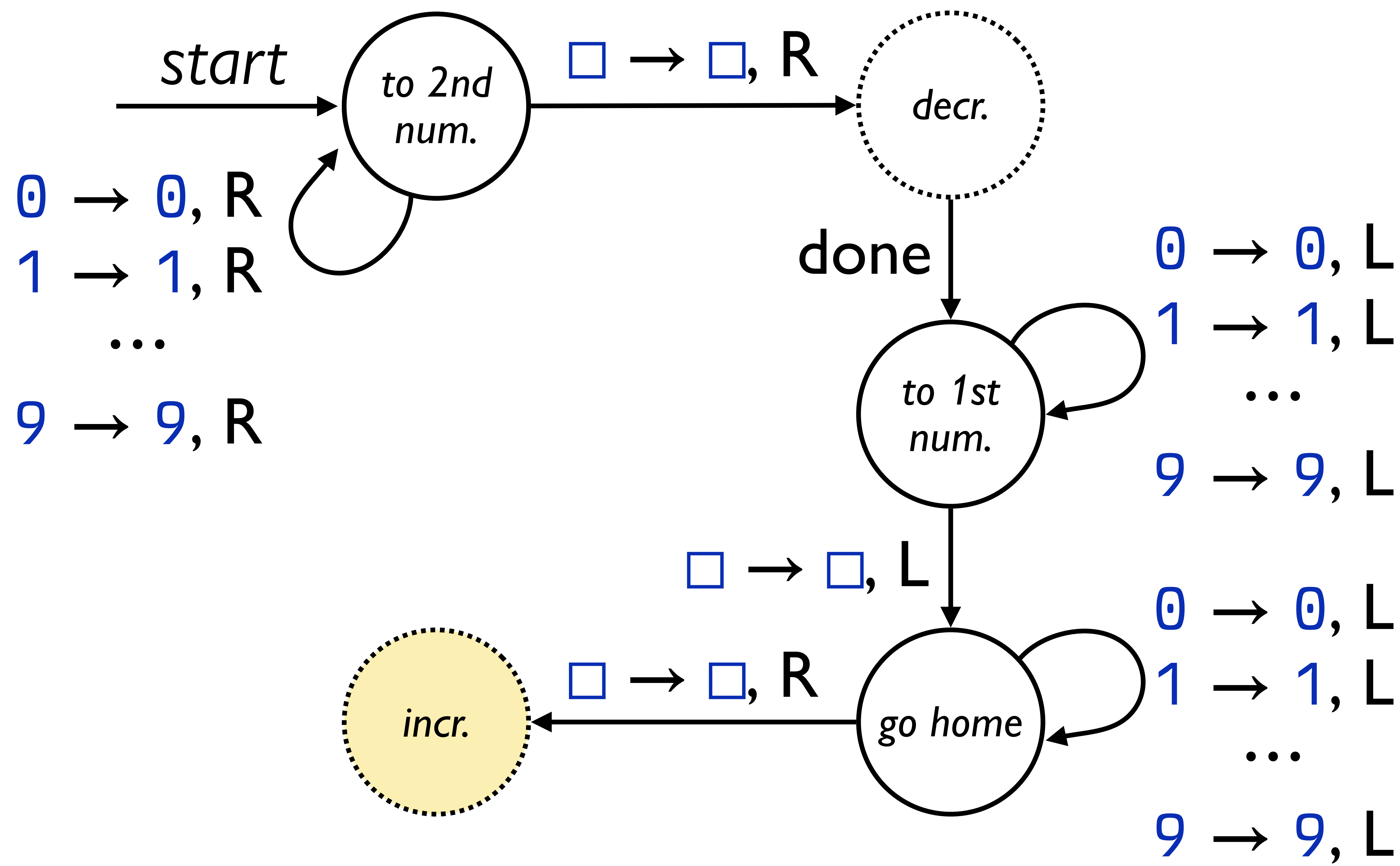
go home

0 → 0, L
1 → 1, L
...
9 → 9, L

□ → □, R

incr.

														1	3	7		4	1														


start
to 2nd num.
□ → □, R
decr.
0 → 0, R
1 → 1, R
…
9 → 9, R
done
to 1st num.
0 → 0, L
1 → 1, L
…
9 → 9, L
□ → □, L
go home
0 → 0, L
1 → 1, L
…
9 → 9, L
□ → □, R
incr.

1 3 7 4 1

start

to 2nd num.

$0 \to 0$, R
$1 \to 1$, R
...
$9 \to 9$, R

□ → □, R

decr.

done

to 1st num.

$0 \to 0$, L
$1 \to 1$, L
...
$9 \to 9$, L

□ → □, L

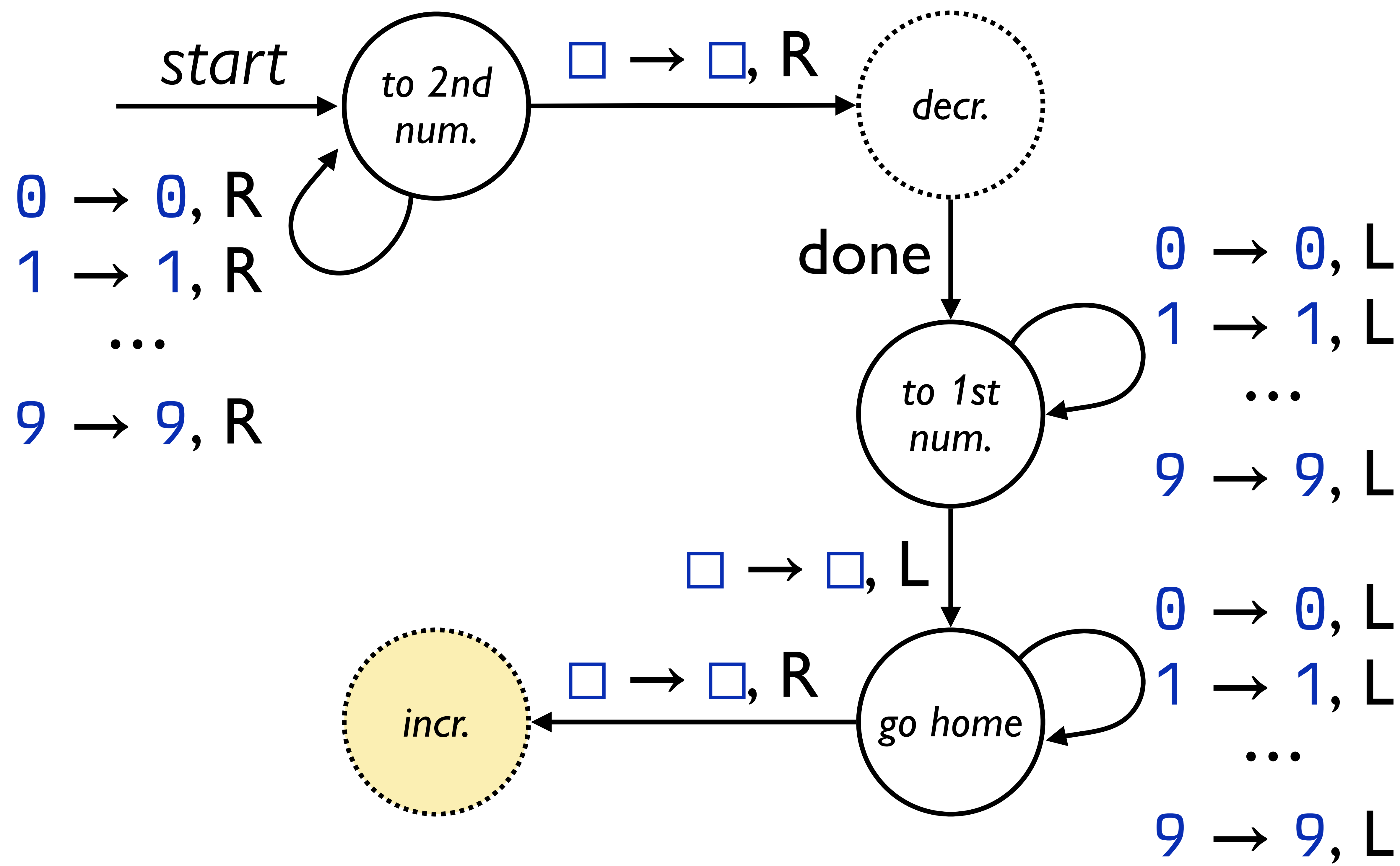
go home

$0 \to 0$, L
$1 \to 1$, L
...
$9 \to 9$, L

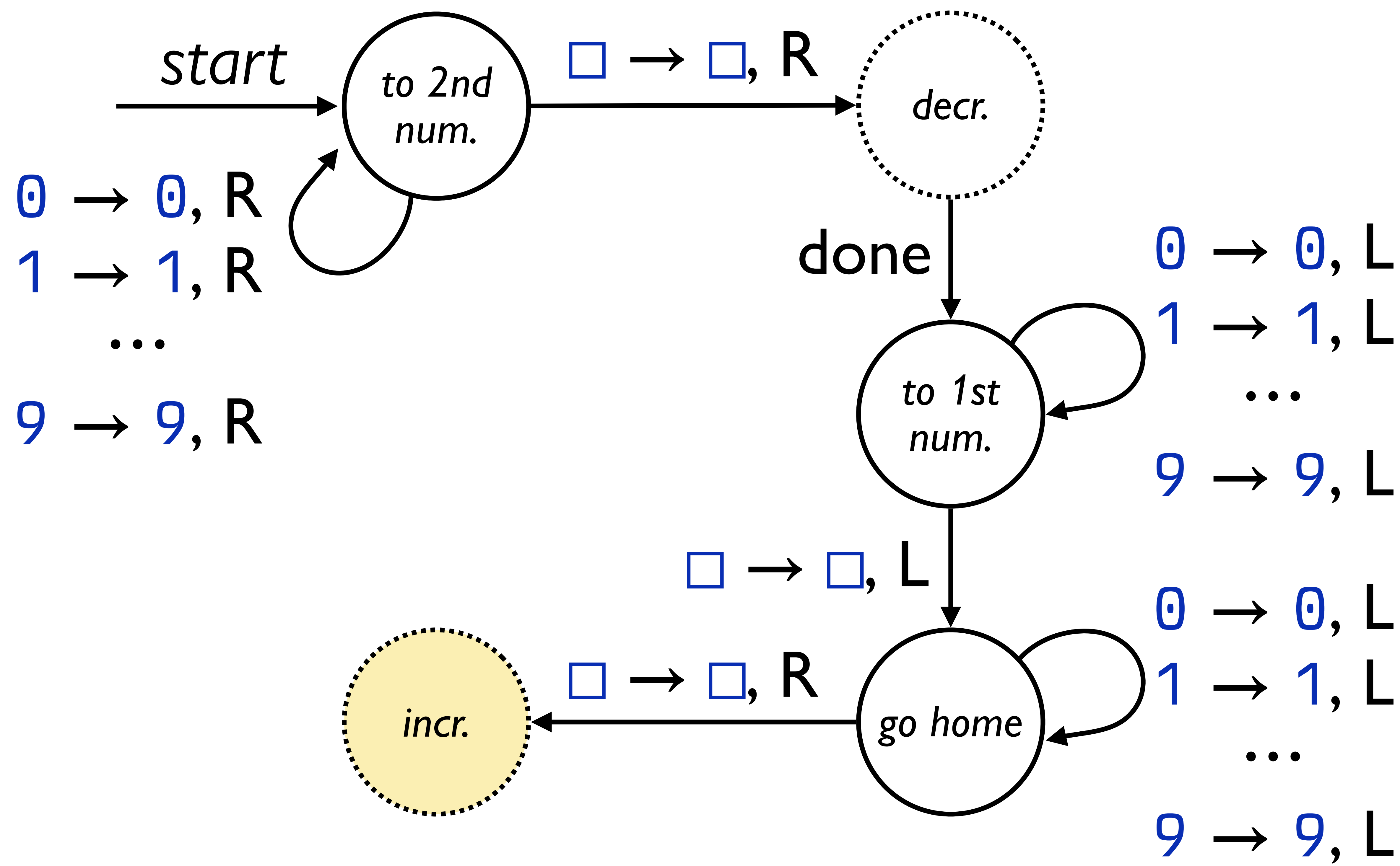
□ → □, R

incr.

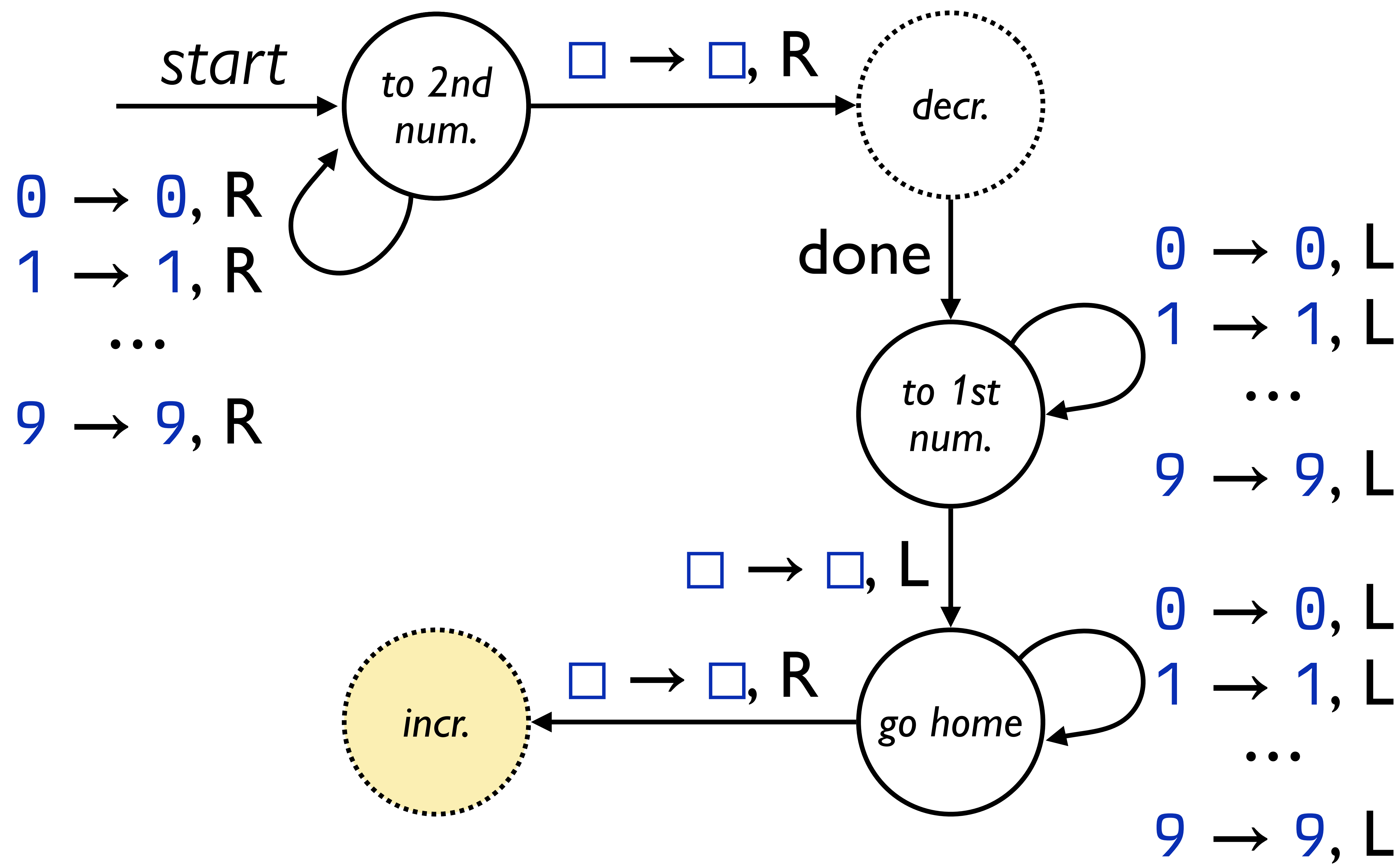
														1	3	7		4	1														


start
to 2nd num.
□ → □, R
decr.
0 → 0, R
1 → 1, R
...
9 → 9, R
done
0 → 0, L
1 → 1, L
...
9 → 9, L
to 1st num.
□ → □, L
0 → 0, L
1 → 1, L
...
9 → 9, L
incr.
□ → □, R
go home

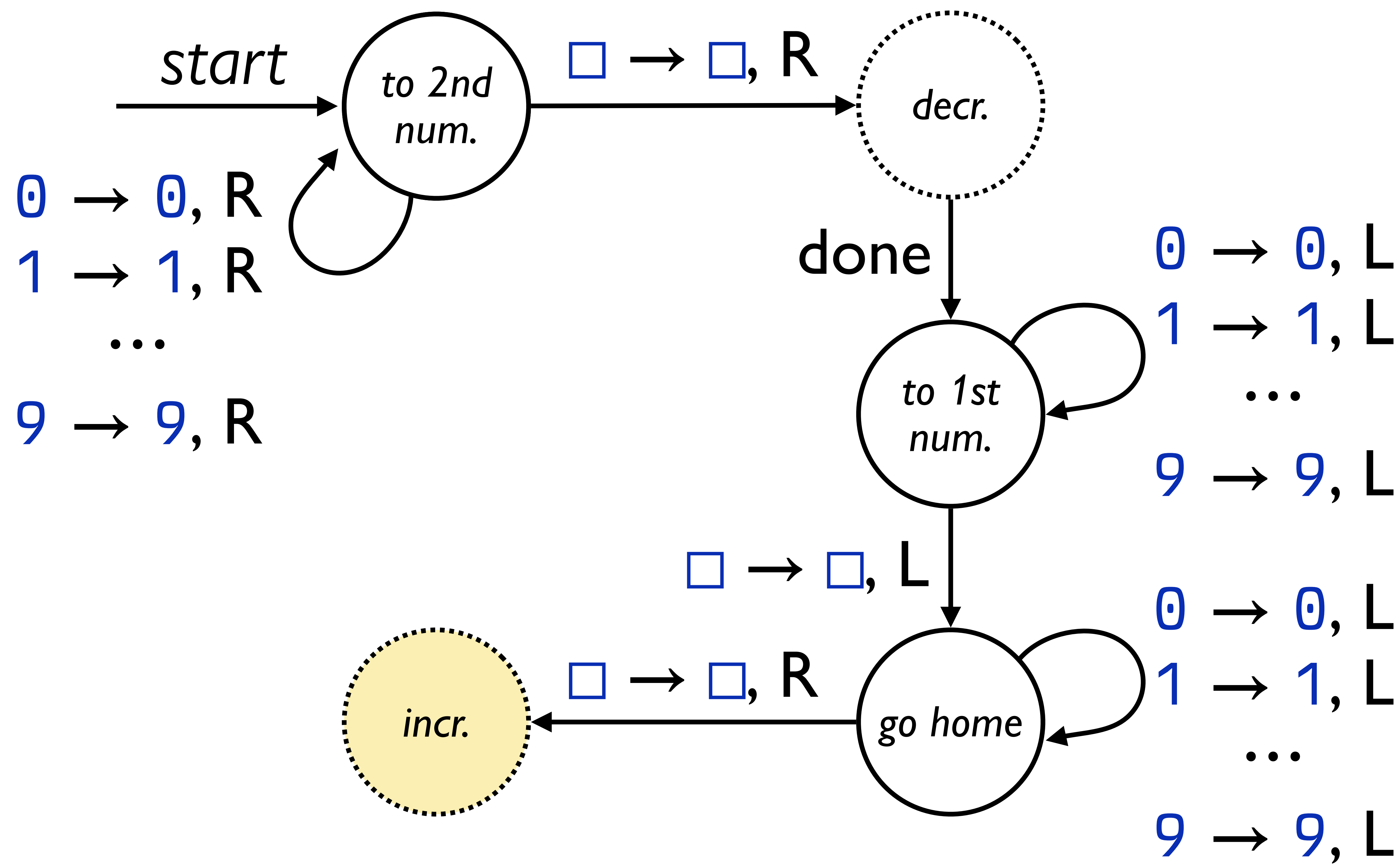
1 3 7 4 1

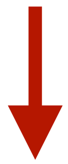

start
to 2nd num.
□ → □, R
decr.
0 → 0, R
1 → 1, R
…
9 → 9, R
done
0 → 0, L
1 → 1, L
…
9 → 9, L
to 1st num.
□ → □, L
0 → 0, L
1 → 1, L
…
9 → 9, L
□ → □, R
incr.
go home

1 3 7 4 1

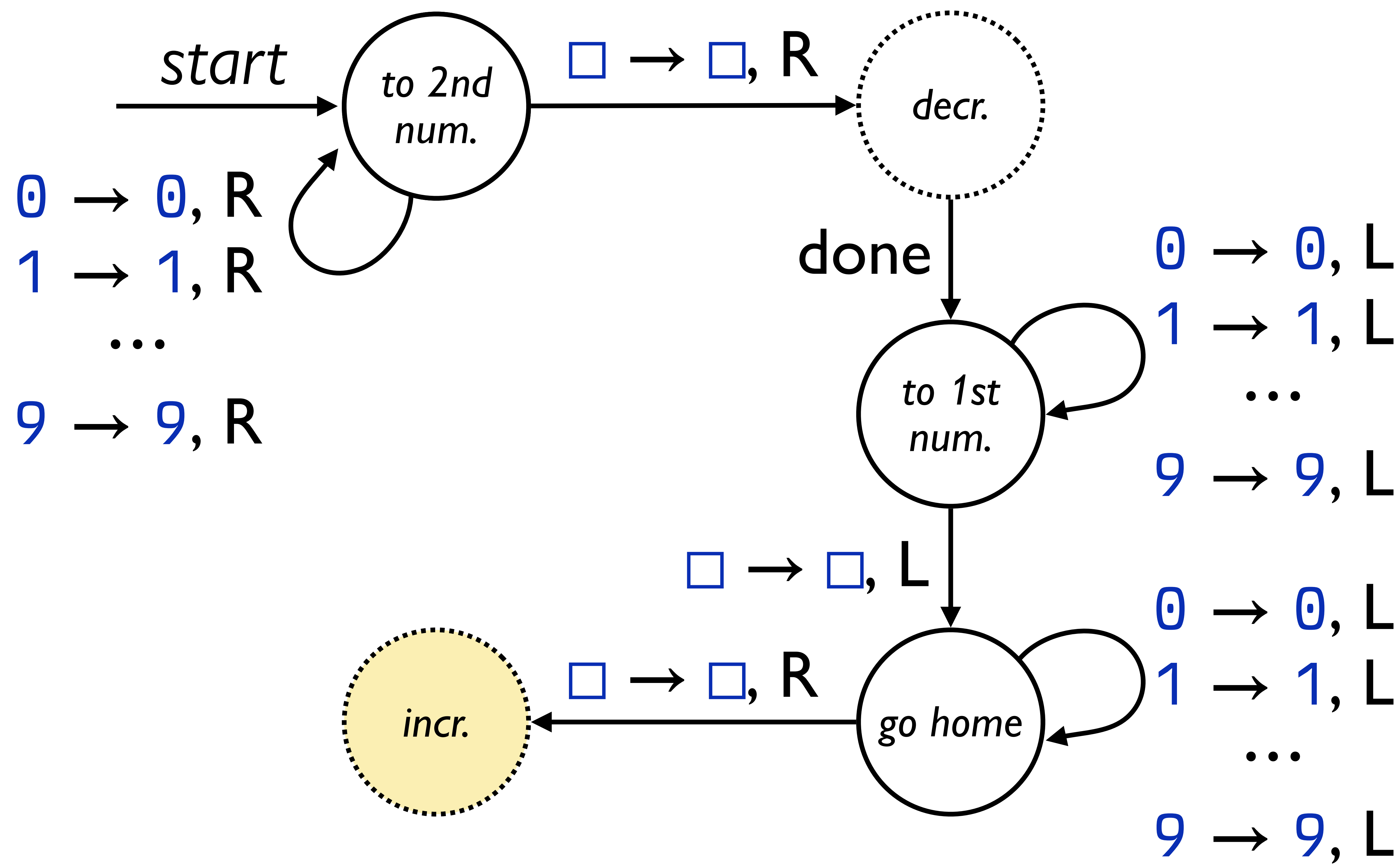

start
to 2nd num.
□ → □, R
decr.
0 → 0, R
1 → 1, R
…
9 → 9, R
done
0 → 0, L
1 → 1, L
…
9 → 9, L
to 1st num.
□ → □, L
0 → 0, L
1 → 1, L
…
9 → 9, L
□ → □, R
incr.
go home

1 3 7 4 1

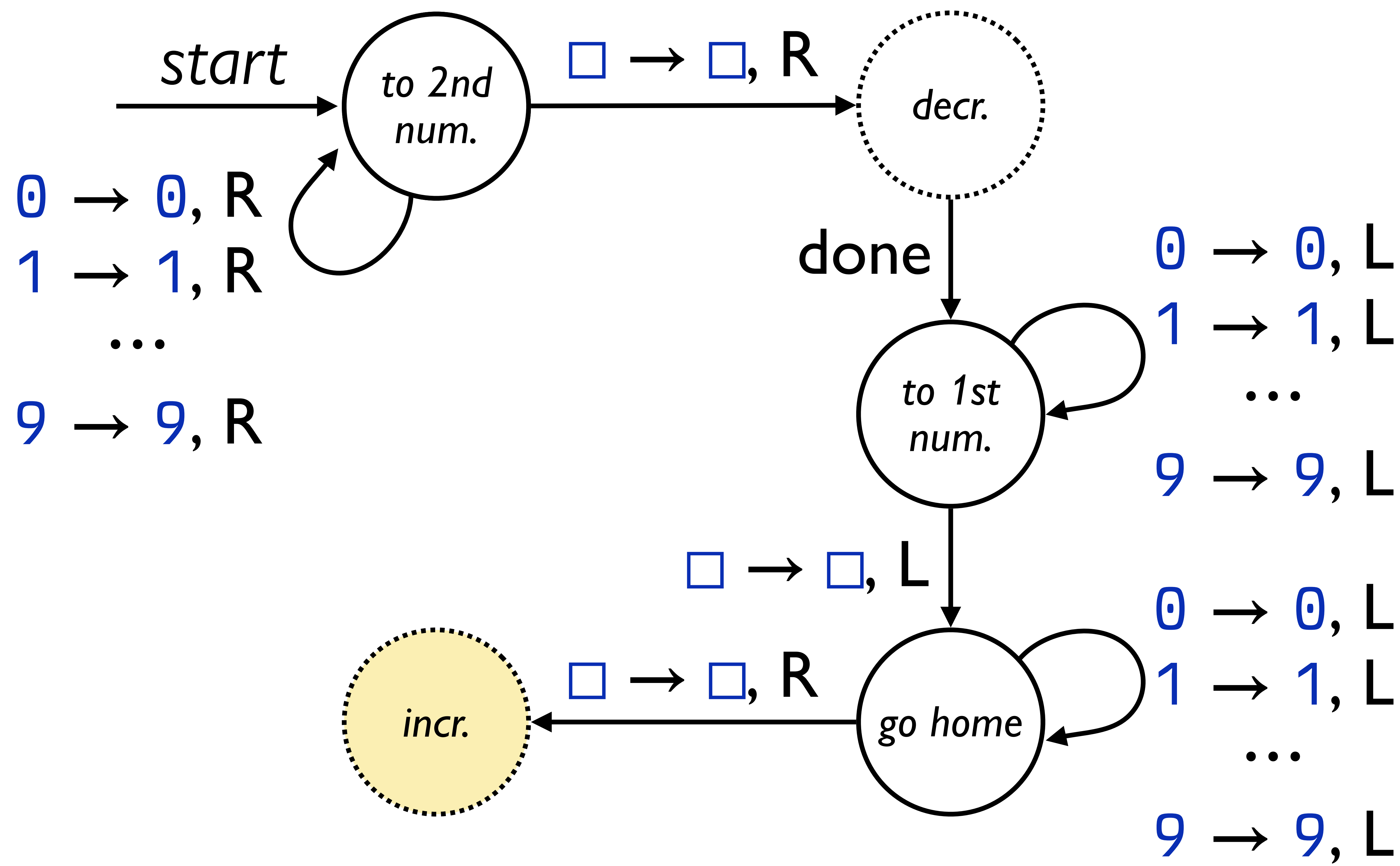

start
to 2nd num.
□ → □, R
decr.
0 → 0, R
1 → 1, R
...
9 → 9, R
done
to 1st num.
0 → 0, L
1 → 1, L
...
9 → 9, L
□ → □, L
go home
0 → 0, L
1 → 1, L
...
9 → 9, L
□ → □, R
incr.



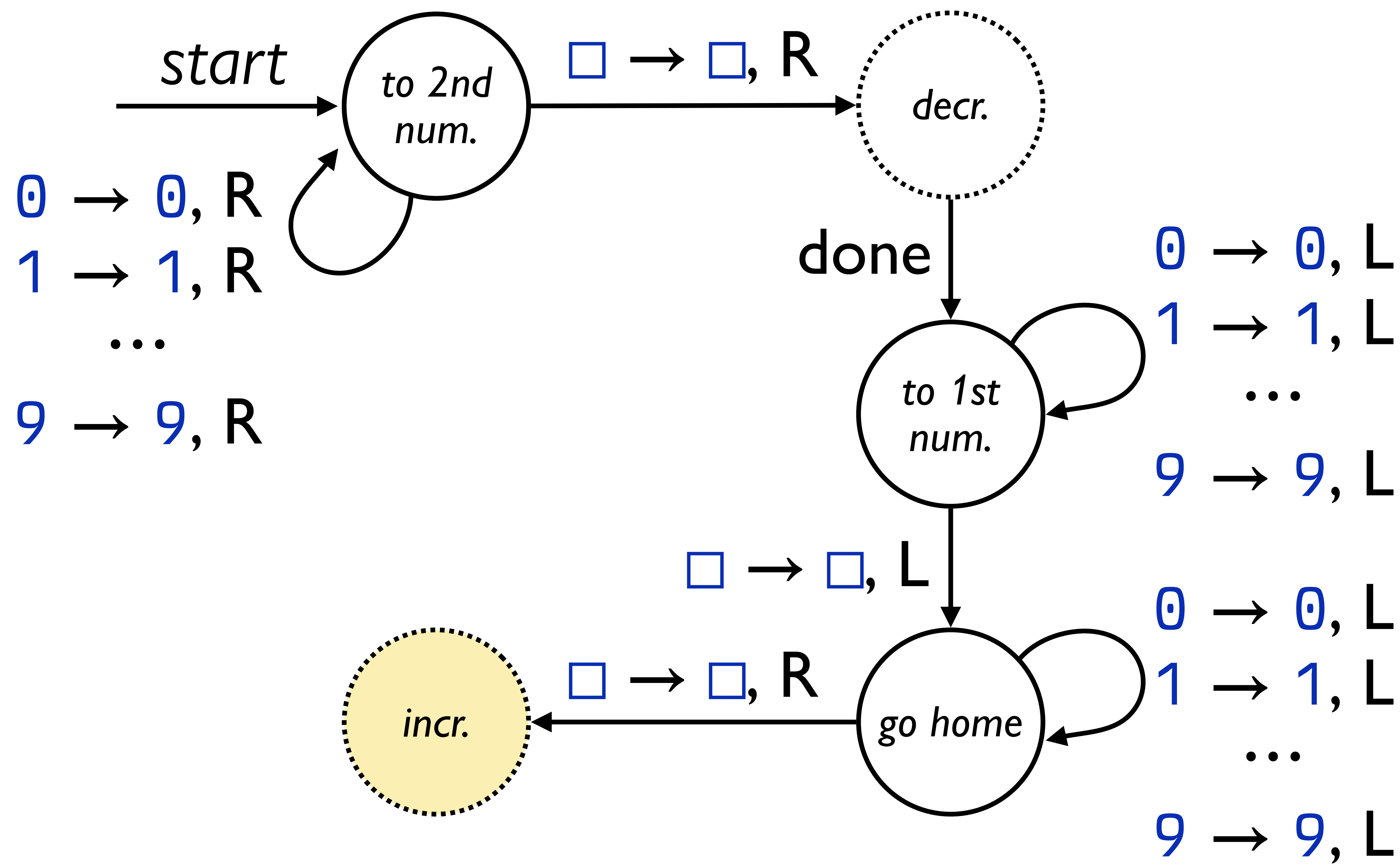
1 3 8 4 1


start
to 2nd num.
□ → □, R
decr.
0 → 0, R
1 → 1, R
…
9 → 9, R
done
0 → 0, L
1 → 1, L
…
9 → 9, L
to 1st num.
□ → □, L
0 → 0, L
1 → 1, L
…
9 → 9, L
□ → □, R
incr.
go home

1 3 8 4 1


start
to 2nd num.
□ → □, R
decr.
0 → 0, R
1 → 1, R
…
9 → 9, R
done
0 → 0, L
1 → 1, L
…
9 → 9, L
to 1st num.
□ → □, L
0 → 0, L
1 → 1, L
…
9 → 9, L
incr.
□ → □, R
go home

1 3 8 4 1


start
to 2nd num.
□ → □, R
decr.
0 → 0, R
1 → 1, R
…
9 → 9, R
done
to 1st num.
0 → 0, L
1 → 1, L
…
9 → 9, L
□ → □, L
go home
0 → 0, L
1 → 1, L
…
9 → 9, L
□ → □, R
incr.

1
3
8
4
1

start

to 2nd num.

0 → 0, R
1 → 1, R
…
9 → 9, R

□ → □, R

decr.

done

to 1st num.

0 → 0, L
1 → 1, L
…
9 → 9, L

□ → □, L

go home

0 → 0, L
1 → 1, L
…
9 → 9, L

□ → □, R

incr.

														1	3	8		4	1														

start
to 2nd num.
□ → □, R
decr.
0 → 0, R
1 → 1, R
...
9 → 9, R
done
done
to 1st num.
0 → 0, L
1 → 1, L
...
9 → 9, L
□ → □, L
go home
0 → 0, L
1 → 1, L
...
9 → 9, L
incr.
□ → □, R
1 3 8 4 1

start
to 2nd num.
□ → □, R
decr.
0 → 0, R
1 → 1, R
...
9 → 9, R
done
done
to 1st num.
0 → 0, L
1 → 1, L
...
9 → 9, L
□ → □, L
go home
0 → 0, L
1 → 1, L
...
9 → 9, L
incr.
□ → □, R
1 3 8 4 1

start

to 2nd num.

□ → □, R

decr.

0 → 0, R
1 → 1, R
…
9 → 9, R

done

to 1st num.

0 → 0, L
1 → 1, L
…
9 → 9, L

done

□ → □, L

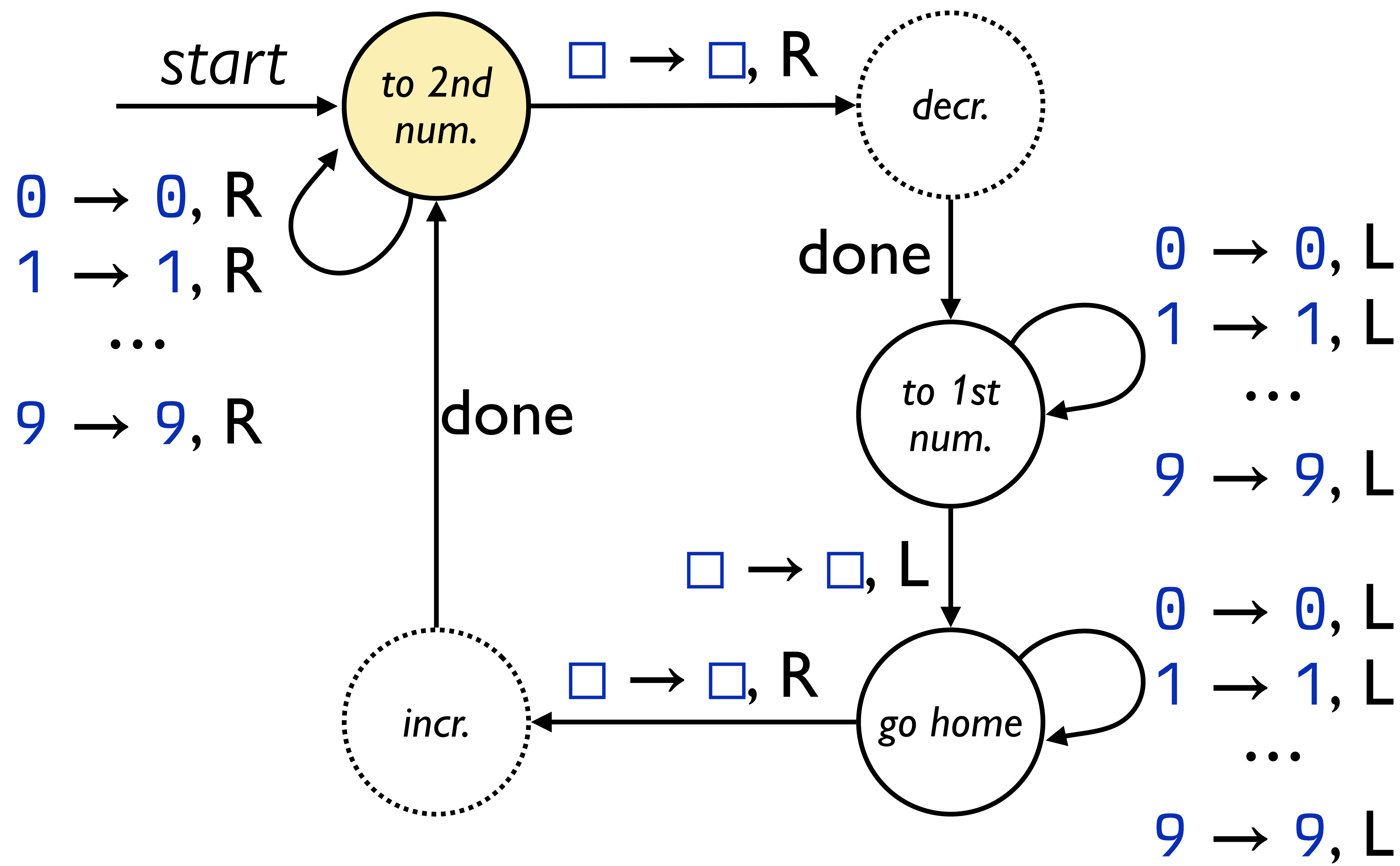
go home

0 → 0, L
1 → 1, L
…
9 → 9, L

□ → □, R

incr.

														1	3	8		4	1													


start
to 2nd num.
□ → □, R
decr.
0 → 0, R
1 → 1, R
…
9 → 9, R
done
done
to 1st num.
0 → 0, L
1 → 1, L
…
9 → 9, L
□ → □, L
incr.
□ → □, R
go home
0 → 0, L
1 → 1, L
…
9 → 9, L

1 3 8 4 1

start

to 2nd num.

□ → □, R

decr.

0 → 0, R
1 → 1, R
…
9 → 9, R

done

done

to 1st num.

0 → 0, L
1 → 1, L
…
9 → 9, L

□ → □, L

incr.

□ → □, R

go home

0 → 0, L
1 → 1, L
…
9 → 9, L

														1	3	8		4	1														

start

to 2nd num.

□ → □, R

decr.

0 → 0, R
1 → 1, R
…
9 → 9, R

done

done

to 1st num.

0 → 0, L
1 → 1, L
…
9 → 9, L

□ → □, L

go home

0 → 0, L
1 → 1, L
…
9 → 9, L

incr.

□ → □, R

1	3	8		4	1

start

to 2nd num.

□ → □, R

decr.

0 → 0, R
1 → 1, R
...
9 → 9, R

done

0 → 0, L
1 → 1, L
...
9 → 9, L

to 1st num.

done

□ → □, L

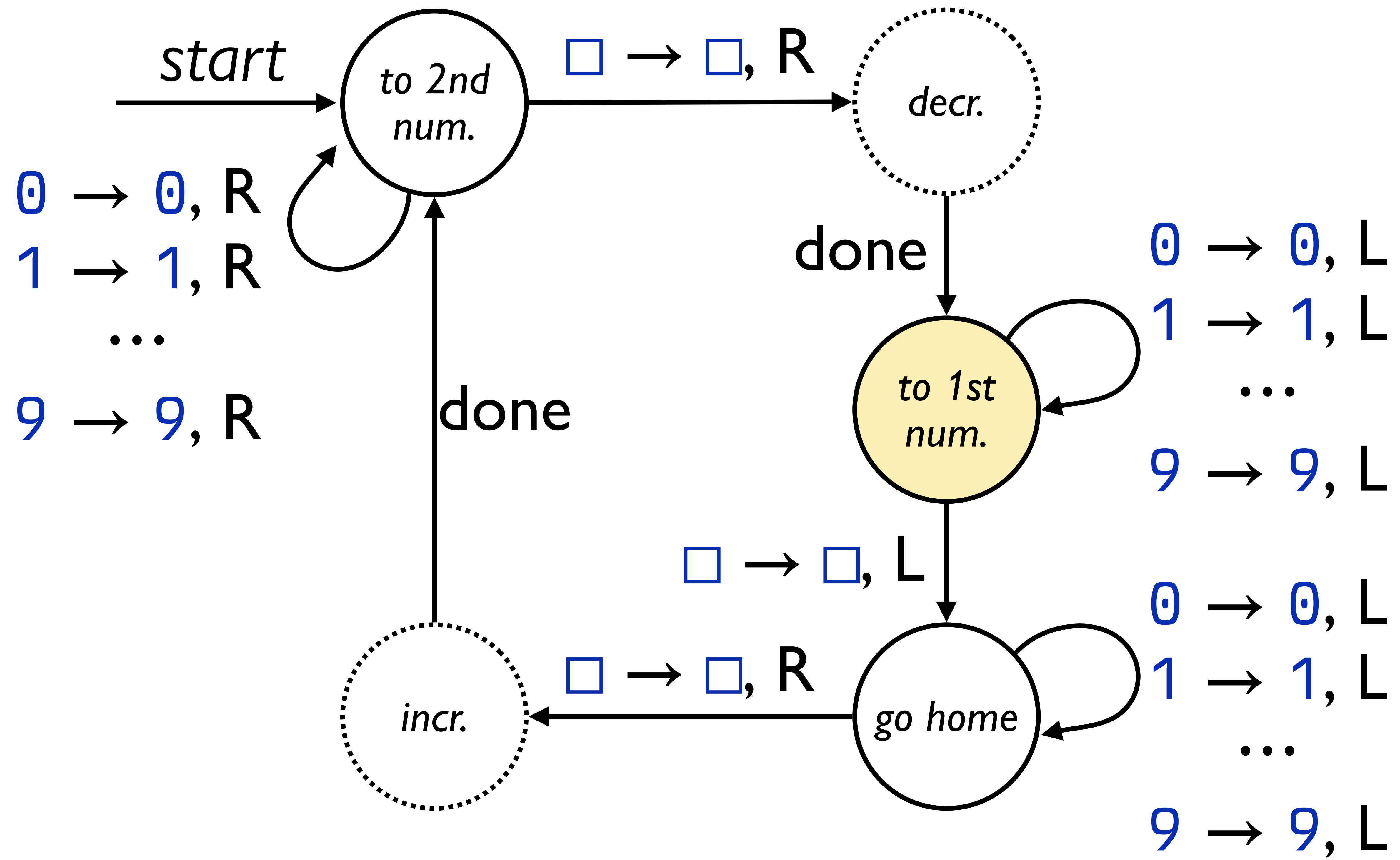
0 → 0, L
1 → 1, L
...
9 → 9, L

incr.

□ → □, R

go home

1	3	8		4	1	


start
to 2nd num.
□ → □, R
decr.
0 → 0, R
1 → 1, R
…
9 → 9, R
done
0 → 0, L
1 → 1, L
…
9 → 9, L
to 1st num.
done
□ → □, L
0 → 0, L
1 → 1, L
…
9 → 9, L
incr.
□ → □, R
go home

1 3 8 4 0

start

to 2nd num.

□ → □, R

decr.

0 → 0, R
1 → 1, R
...
9 → 9, R

done

done

to 1st num.

0 → 0, L
1 → 1, L
...
9 → 9, L

□ → □, L

incr.

□ → □, R

go home

0 → 0, L
1 → 1, L
...
9 → 9, L

1	3	8		4	0

start
to 2nd num.
□ → □, R
decr.
0 → 0, R
1 → 1, R
...
9 → 9, R
done
done
to 1st num.
0 → 0, L
1 → 1, L
...
9 → 9, L
□ → □, L
0 → 0, L
1 → 1, L
...
9 → 9, L
□ → □, R
incr.
go home
1 3 8 4 0

start
to 2nd num.
□ → □, R
decr.
0 → 0, R
1 → 1, R
...
9 → 9, R
done
done
to 1st num.
0 → 0, L
1 → 1, L
...
9 → 9, L
□ → □, L
go home
0 → 0, L
1 → 1, L
...
9 → 9, L
incr.
□ → □, R

														1	3	8		4	0														

start

to 2nd num.

□ → □, R

decr.

0 → 0, R
1 → 1, R
…
9 → 9, R

done

done

to 1st num.

0 → 0, L
1 → 1, L
…
9 → 9, L

□ → □, L

incr.

□ → □, R

go home

0 → 0, L
1 → 1, L
…
9 → 9, L

														1	3	8		4	0													

start

to 2nd num.

$\square \rightarrow \square$, R

decr.

0 → 0, R
1 → 1, R
…
9 → 9, R

done

done

to 1st num.

0 → 0, L
1 → 1, L
…
9 → 9, L

$\square \rightarrow \square$, L

go home

0 → 0, L
1 → 1, L
…
9 → 9, L

$\square \rightarrow \square$, R

incr.

														1	3	8		4	0														

start

to 2nd num.

□ → □, R

decr.

0 → 0, R
1 → 1, R
...
9 → 9, R

done

done

to 1st num.

0 → 0, L
1 → 1, L
...
9 → 9, L

□ → □, L

incr.

□ → □, R

go home

0 → 0, L
1 → 1, L
...
9 → 9, L

														1	3	8		4	0													

start

to 2nd num.

□ → □, R

decr.

0 → 0, R
1 → 1, R
...
9 → 9, R

done

done

to 1st num.

0 → 0, L
1 → 1, L
...
9 → 9, L

□ → □, L

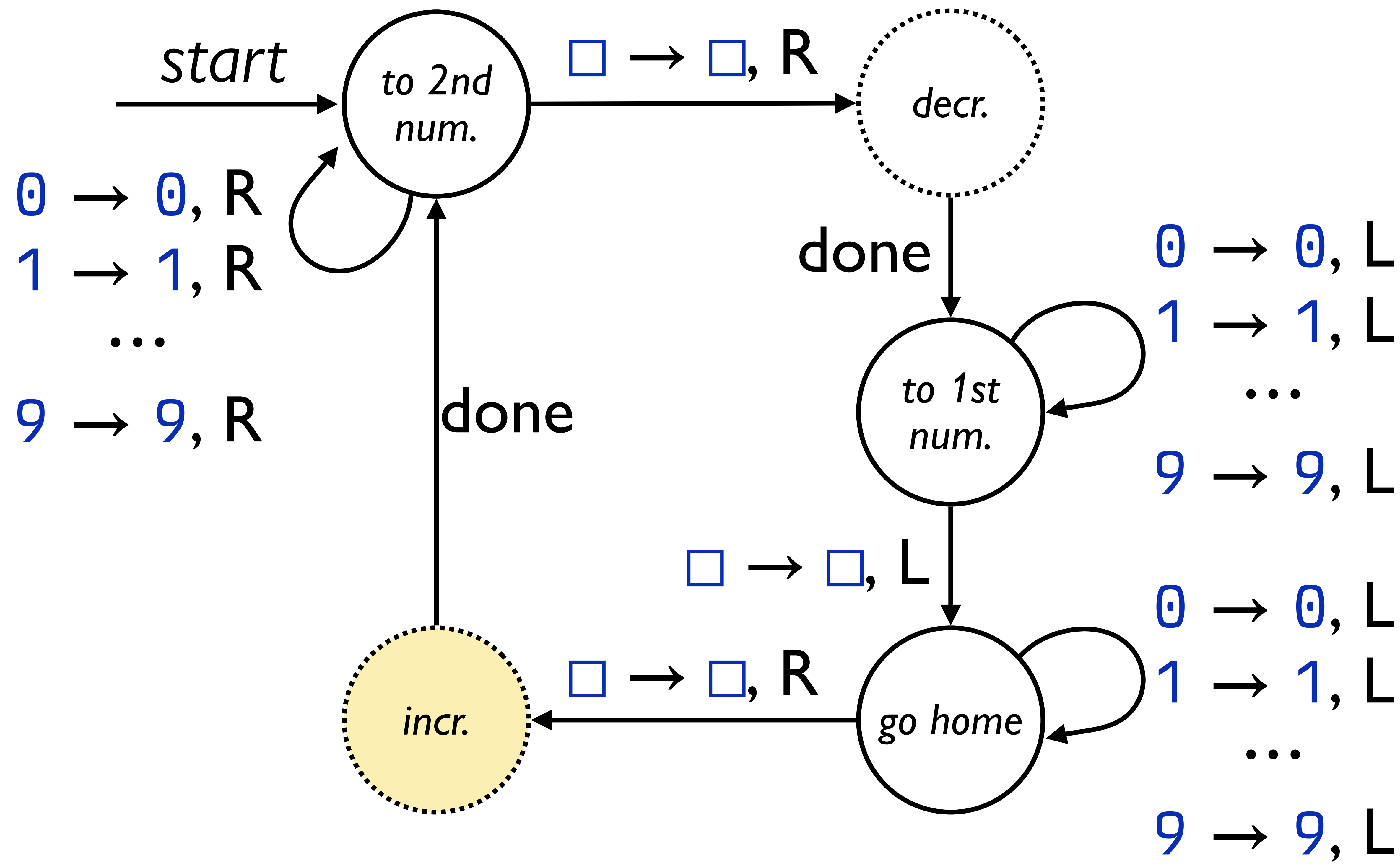
go home

0 → 0, L
1 → 1, L
...
9 → 9, L

incr.

□ → □, R

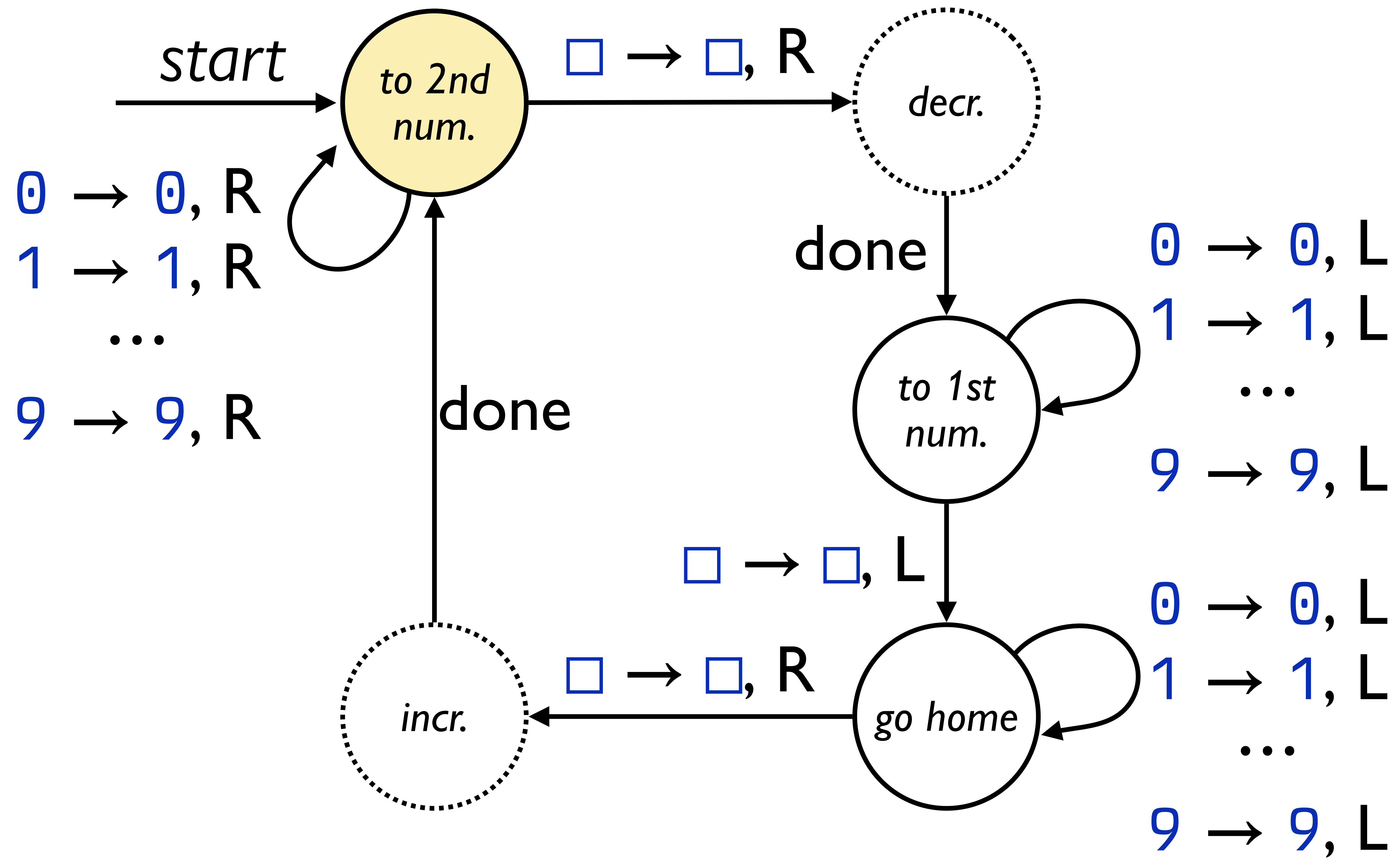
														1	3	8		4	0														


start
to 2nd num.
□ → □, R
decr.
0 → 0, R
1 → 1, R
…
9 → 9, R
done
done
to 1st num.
0 → 0, L
1 → 1, L
…
9 → 9, L
□ → □, L
incr.
□ → □, R
go home
0 → 0, L
1 → 1, L
…
9 → 9, L

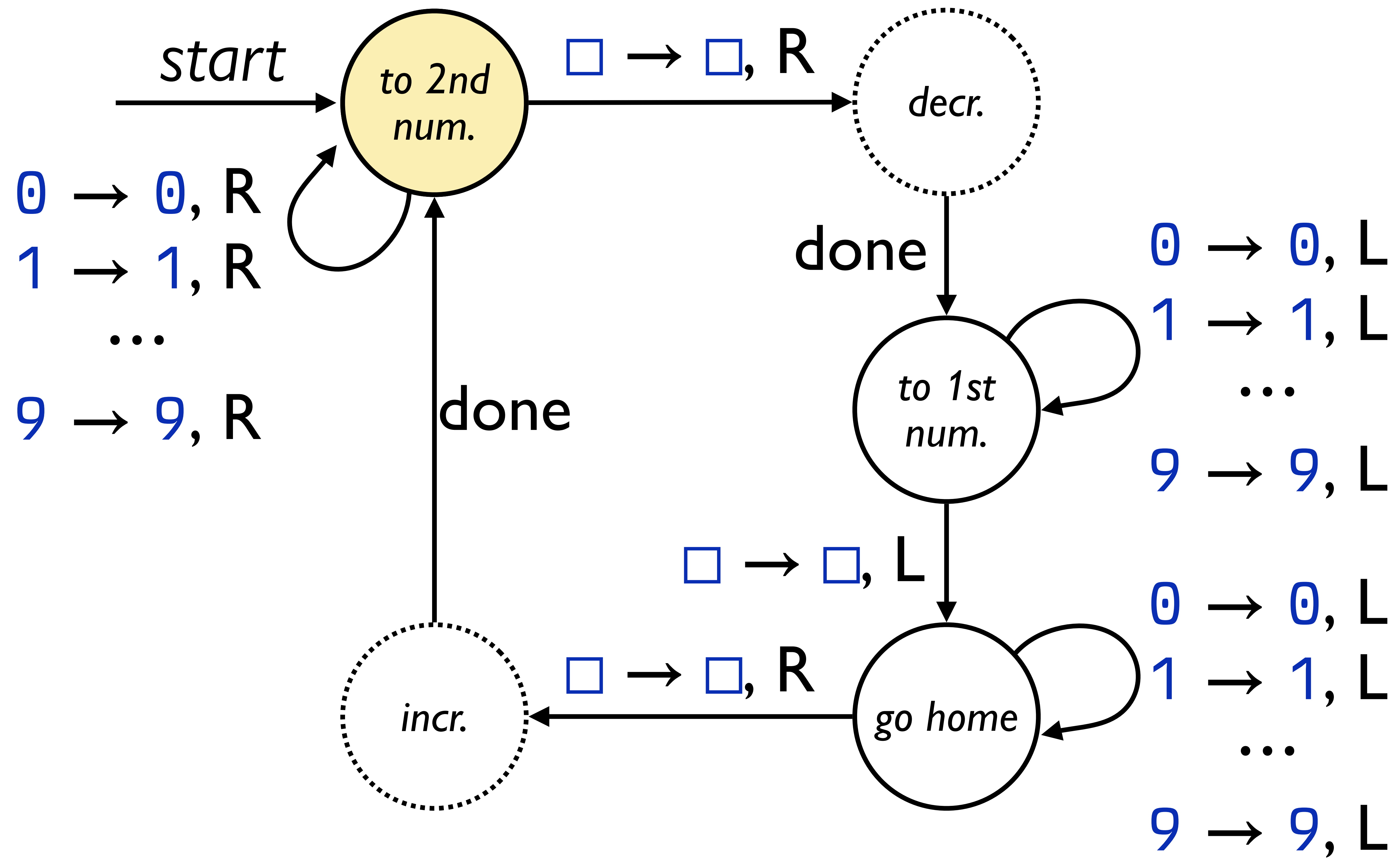
1 3 8 4 0

start
to 2nd num.
□ → □, R
decr.
0 → 0, R
1 → 1, R
…
9 → 9, R
done
done
0 → 0, L
1 → 1, L
…
9 → 9, L
to 1st num.
□ → □, L
0 → 0, L
1 → 1, L
…
9 → 9, L
□ → □, R
incr.
go home
1 3 9 4 0

start
to 2nd num.
□ → □, R
decr.
0 → 0, R
1 → 1, R
…
9 → 9, R
done
done
to 1st num.
0 → 0, L
1 → 1, L
…
9 → 9, L
□ → □, L
incr.
□ → □, R
go home
0 → 0, L
1 → 1, L
…
9 → 9, L
1 3 9 4 0


start
to 2nd num.
□ → □, R
decr.
0 → 0, R
1 → 1, R
…
9 → 9, R
done
done
0 → 0, L
1 → 1, L
…
9 → 9, L
to 1st num.
□ → □, L
0 → 0, L
1 → 1, L
…
9 → 9, L
incr.
□ → □, R
go home

1 3 9 4 0


start
to 2nd num.
□ → □, R
decr.
0 → 0, R
1 → 1, R
…
9 → 9, R
done
done
to 1st num.
0 → 0, L
1 → 1, L
…
9 → 9, L
□ → □, L
incr.
□ → □, R
go home
0 → 0, L
1 → 1, L
…
9 → 9, L

1 3 9 4 0

start

to 2nd num.

□ → □, R

decr.

0 → 0, R
1 → 1, R
…
9 → 9, R

done

done

0 → 0, L
1 → 1, L
…
9 → 9, L

to 1st num.

□ → □, L

0 → 0, L
1 → 1, L
…
9 → 9, L

incr.

□ → □, R

go home

														1	3	9		4	0														

start

to 2nd num.

□ → □, R

decr.

0 → 0, R
1 → 1, R
…
9 → 9, R

done

done

to 1st num.

0 → 0, L
1 → 1, L
…
9 → 9, L

□ → □, L

incr.

□ → □, R

go home

0 → 0, L
1 → 1, L
…
9 → 9, L

														1	3	9		4	0														

start

to 2nd num.

□ → □, R

decr.

0 → 0, R
1 → 1, R
...
9 → 9, R

done

done

to 1st num.

0 → 0, L
1 → 1, L
...
9 → 9, L

□ → □, L

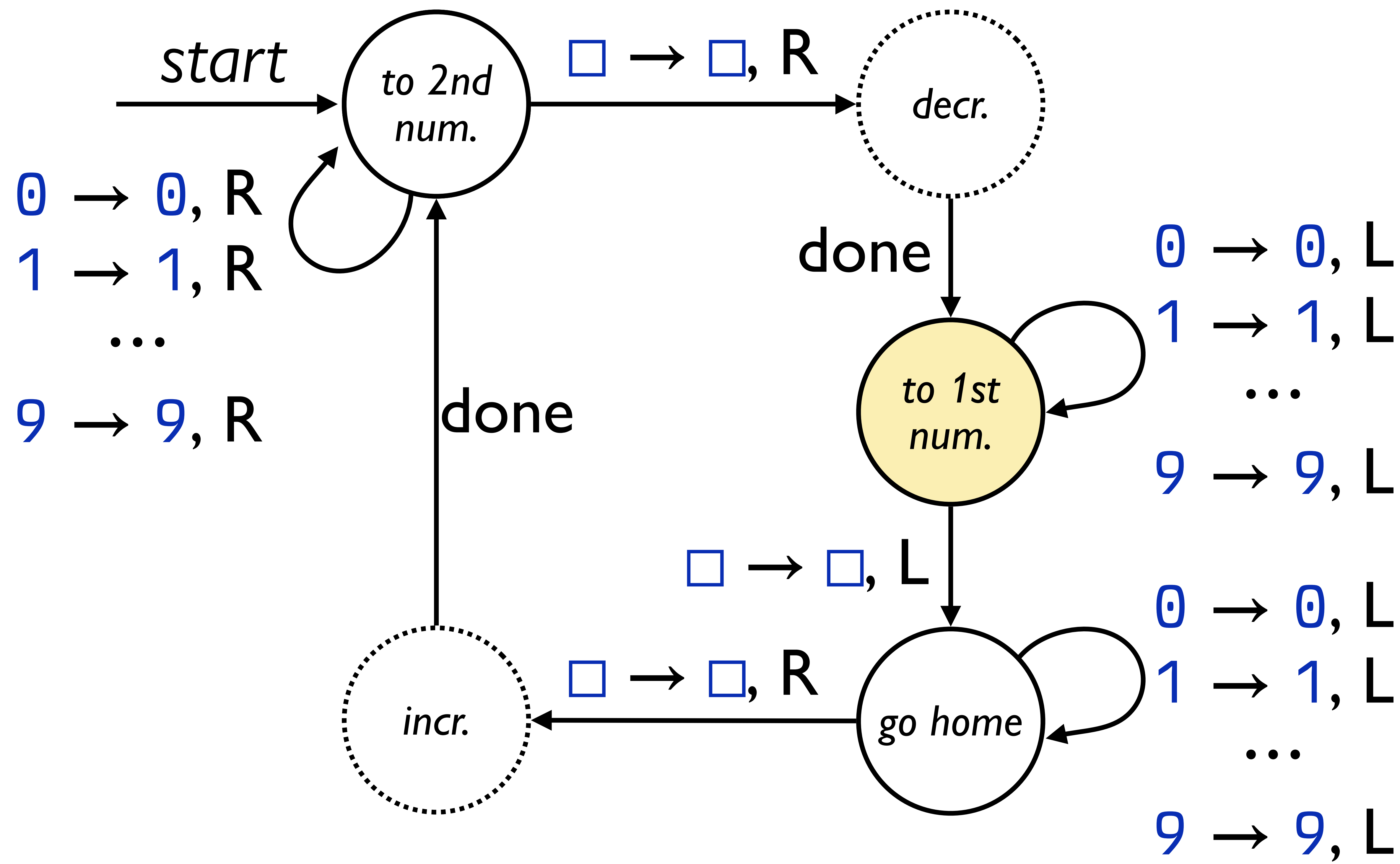
incr.

□ → □, R

go home

0 → 0, L
1 → 1, L
...
9 → 9, L

1	3	9		3	9



start
to 2nd num.
□ → □, R
decr.
0 → 0, R
1 → 1, R
…
9 → 9, R
done
done
to 1st num.
0 → 0, L
1 → 1, L
…
9 → 9, L
□ → □, L
go home
0 → 0, L
1 → 1, L
…
9 → 9, L
incr.
□ → □, R

1 3 9 3 9

start
to 2nd num.
□ → □, R
decr.
0 → 0, R
1 → 1, R
…
9 → 9, R
done
done
0 → 0, L
1 → 1, L
…
9 → 9, L
to 1st num.
□ → □, L
0 → 0, L
1 → 1, L
…
9 → 9, L
□ → □, R
incr.
go home
1 3 9 3 9

start
to 2nd num.
□ → □, R
decr.
0 → 0, R
1 → 1, R
…
9 → 9, R
done
to 1st num.
0 → 0, L
1 → 1, L
…
9 → 9, L
done
□ → □, L
incr.
□ → □, R
go home
0 → 0, L
1 → 1, L
…
9 → 9, L
1 3 9 3 9

start

to 2nd num.

□ → □, R

decr.

0 → 0, R
1 → 1, R
…
9 → 9, R

done

done

to 1st num.

0 → 0, L
1 → 1, L
…
9 → 9, L

□ → □, L

incr.

□ → □, R

go home

0 → 0, L
1 → 1, L
…
9 → 9, L

1	3	9		3	9

start
to 2nd num.
□ → □, R
decr.
0 → 0, R
1 → 1, R
...
9 → 9, R
done
done
to 1st num.
0 → 0, L
1 → 1, L
...
9 → 9, L
□ → □, L
□ → □, R
incr.
go home
0 → 0, L
1 → 1, L
...
9 → 9, L
1 3 9 3 9

start

to 2nd num.

□ → □, R

decr.

0 → 0, R
1 → 1, R
...
9 → 9, R

done

done

to 1st num.

0 → 0, L
1 → 1, L
...
9 → 9, L

□ → □, L

go home

0 → 0, L
1 → 1, L
...
9 → 9, L

□ → □, R

incr.

														1	3	9		3	9														

start

to 2nd num.

□ → □, R

decr.

0 → 0, R
1 → 1, R
...
9 → 9, R

done

to 1st num.

0 → 0, L
1 → 1, L
...
9 → 9, L

done

□ → □, L

incr.

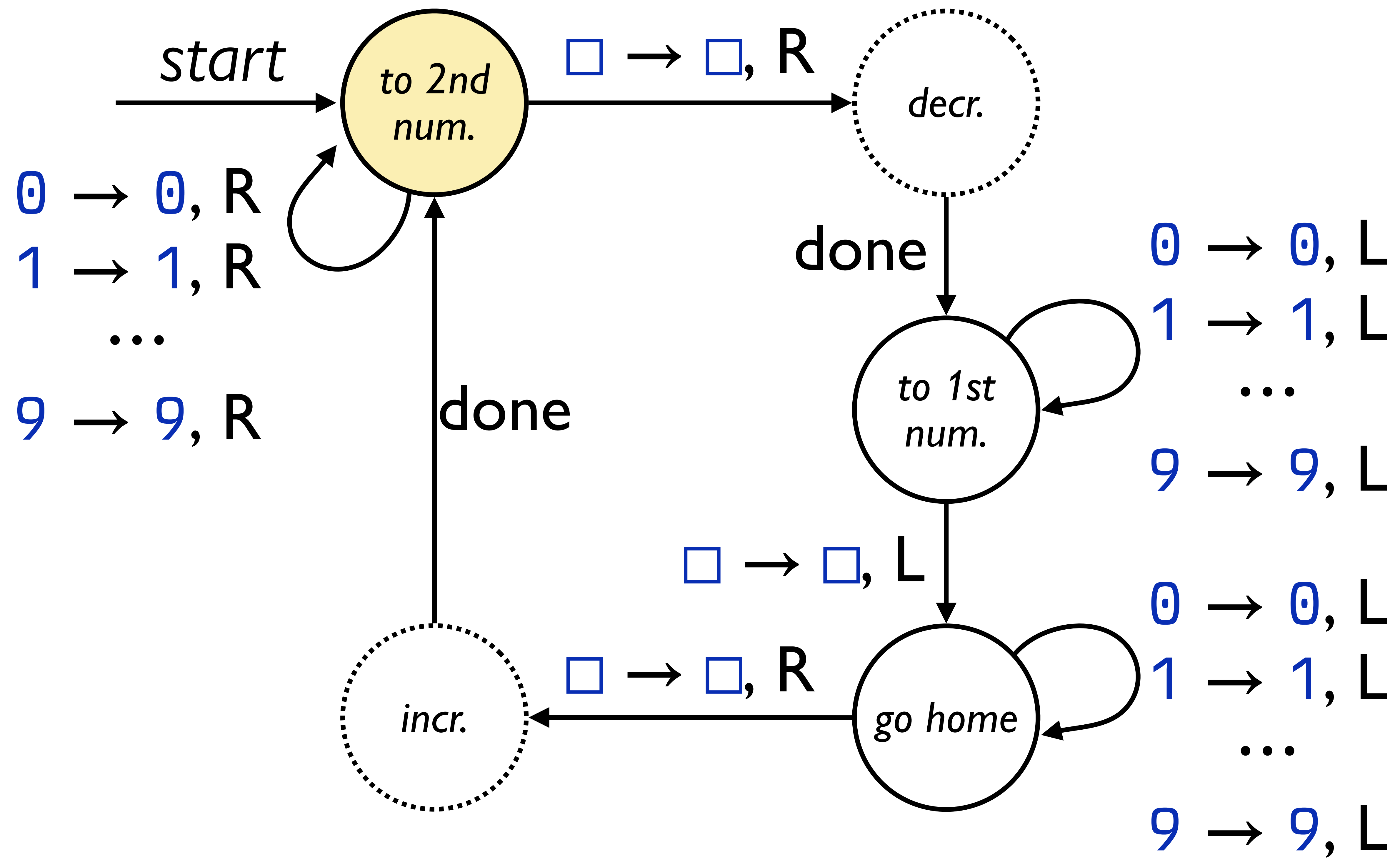
□ → □, R

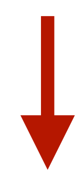
go home

0 → 0, L
1 → 1, L
...
9 → 9, L

1	4	0		3	9

Many transitions later…


start
to 2nd num.
□ → □, R
decr.
0 → 0, R
1 → 1, R
…
9 → 9, R
done
done
to 1st num.
0 → 0, L
1 → 1, L
…
9 → 9, L
□ → □, L
□ → □, R
incr.
go home
0 → 0, L
1 → 1, L
…
9 → 9, L



1 7 8 0 1

start

to 2nd num.

□ → □, R

decr.

0 → 0, R
1 → 1, R
...
9 → 9, R

done

done

to 1st num.

0 → 0, L
1 → 1, L
...
9 → 9, L

□ → □, L

incr.

□ → □, R

go home

0 → 0, L
1 → 1, L
...
9 → 9, L

														1	7	8		0	1														

start

to 2nd num.

□ → □, R

decr.

0 → 0, R
1 → 1, R
…
9 → 9, R

done

done

to 1st num.

0 → 0, L
1 → 1, L
…
9 → 9, L

□ → □, L

go home

0 → 0, L
1 → 1, L
…
9 → 9, L

□ → □, R

incr.

1	7	8		0	1

start

to 2nd num.

□ → □, R

decr.

0 → 0, R
1 → 1, R
…
9 → 9, R

done

done

to 1st num.

0 → 0, L
1 → 1, L
…
9 → 9, L

□ → □, L

go home

0 → 0, L
1 → 1, L
…
9 → 9, L

incr.

□ → □, R

														1	7	8		0	1														

start

to 2nd num.

□ → □, R

decr.

0 → 0, R
1 → 1, R
...
9 → 9, R

done

done

to 1st num.

0 → 0, L
1 → 1, L
...
9 → 9, L

□ → □, L

go home

0 → 0, L
1 → 1, L
...
9 → 9, L

□ → □, R

incr.

														1	7	8		0	1														

start

to 2nd num.

□ → □, R

decr.

0 → 0, R
1 → 1, R
...
9 → 9, R

done

done

to 1st num.

0 → 0, L
1 → 1, L
...
9 → 9, L

□ → □, L

incr.

□ → □, R

go home

0 → 0, L
1 → 1, L
...
9 → 9, L

														1	7	8		0	0														

start

to 2nd num.

0 → 0, R
1 → 1, R
…
9 → 9, R

□ → □, R

decr.

done

to 1st num.

0 → 0, L
1 → 1, L
…
9 → 9, L

□ → □, L

go home

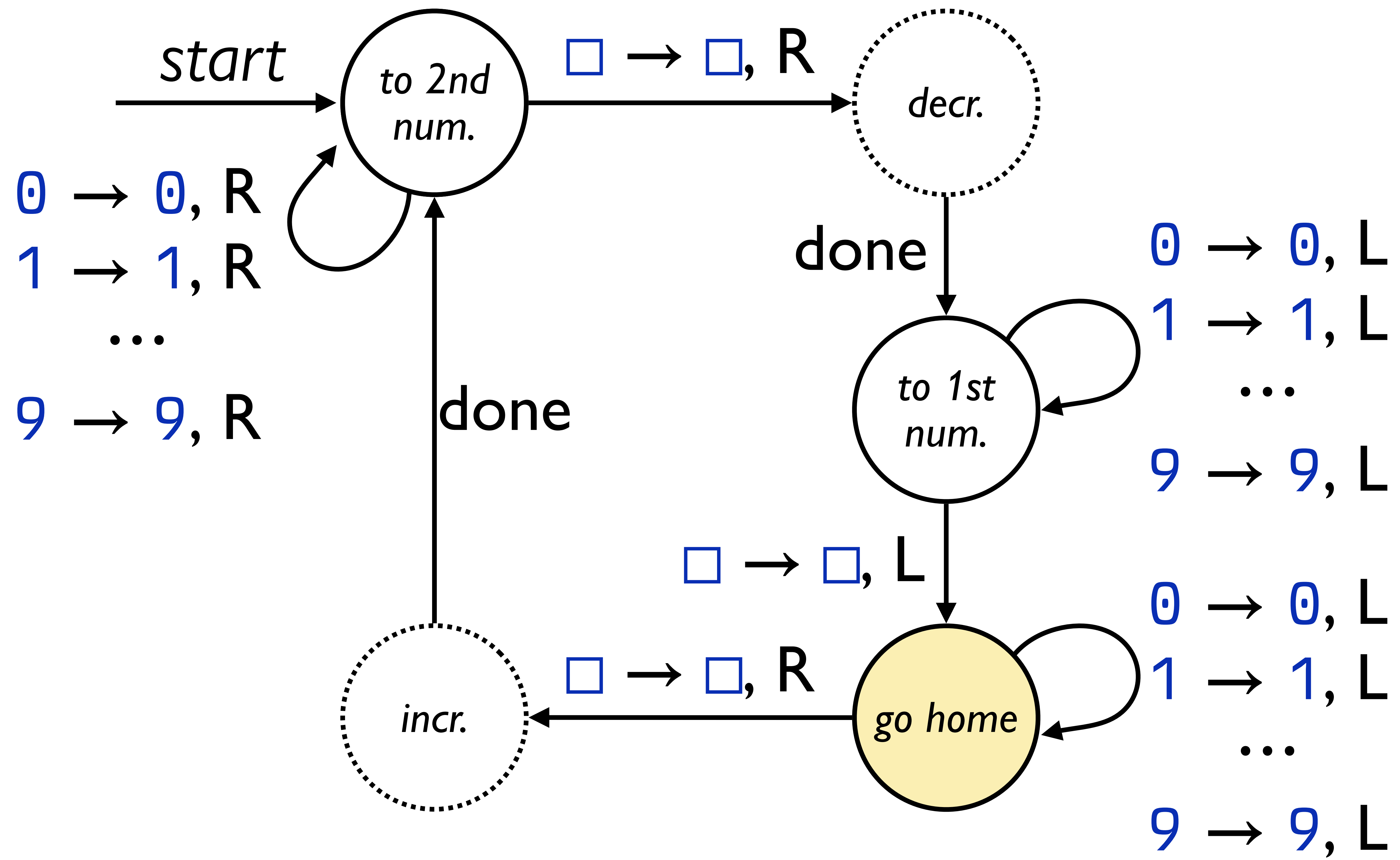
0 → 0, L
1 → 1, L
…
9 → 9, L

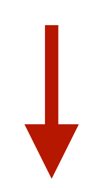
□ → □, R

incr.

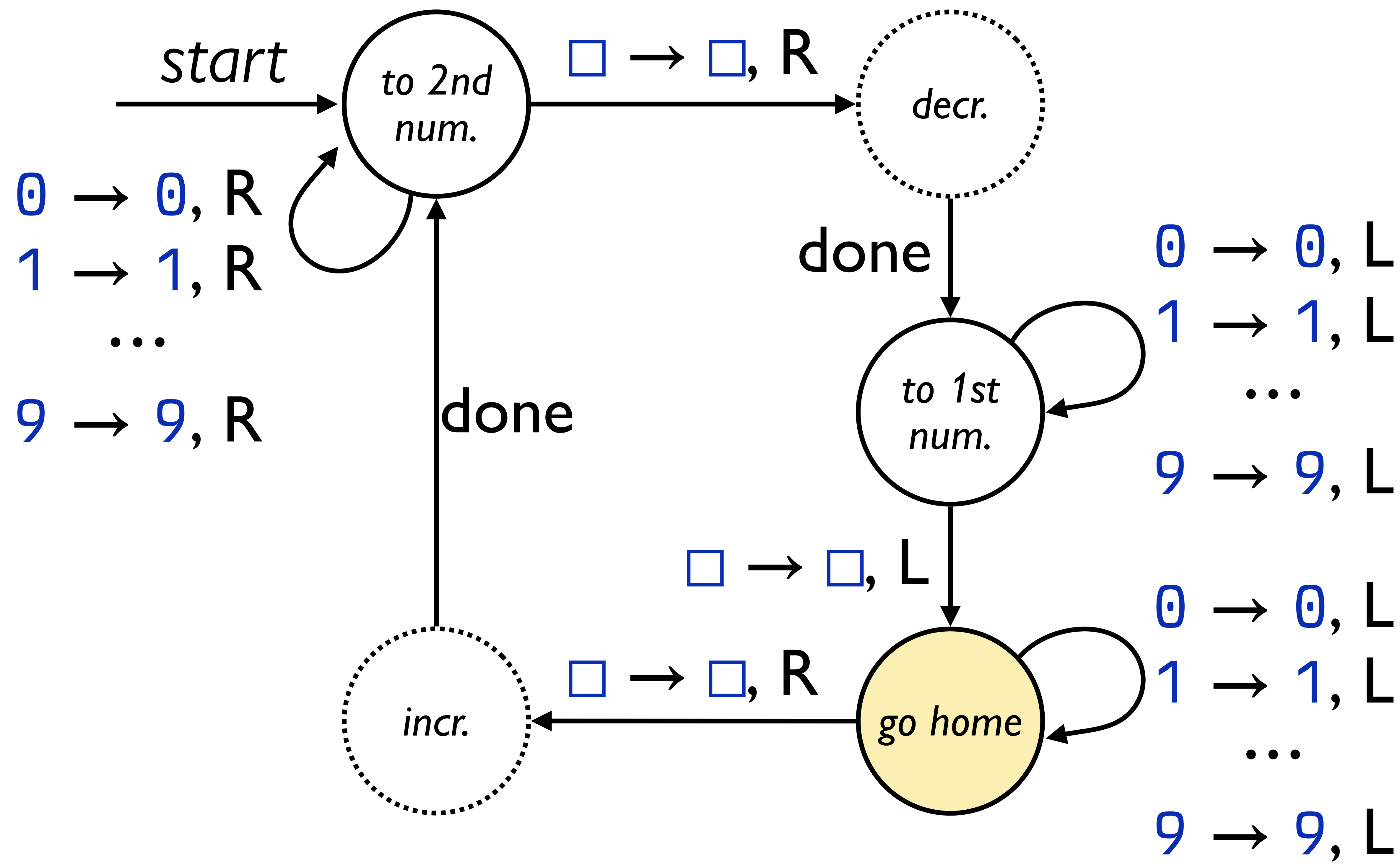
done

1	7	8		0	0

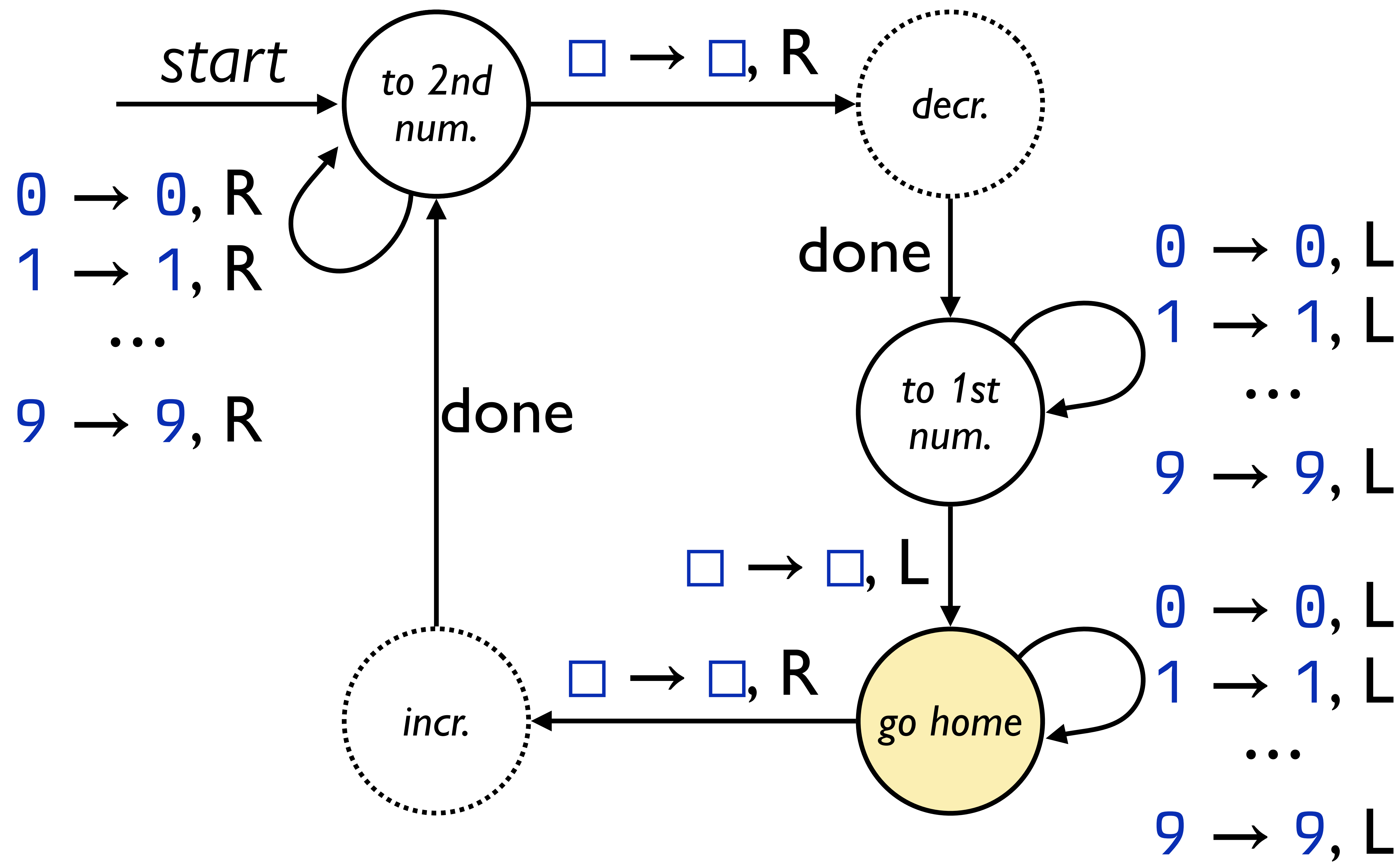

start
to 2nd num.
□ → □, R
decr.
0 → 0, R
1 → 1, R
...
9 → 9, R
done
done
to 1st num.
0 → 0, L
1 → 1, L
...
9 → 9, L
□ → □, L
incr.
□ → □, R
go home
0 → 0, L
1 → 1, L
...
9 → 9, L



1 7 8 0 0


start
to 2nd num.
□ → □, R
decr.
0 → 0, R
1 → 1, R
…
9 → 9, R
done
done
to 1st num.
0 → 0, L
1 → 1, L
…
9 → 9, L
□ → □, L
incr.
□ → □, R
go home
0 → 0, L
1 → 1, L
…
9 → 9, L

1 7 8 0 0


start
to 2nd num.
□ → □, R
decr.
0 → 0, R
1 → 1, R
…
9 → 9, R
done
done
to 1st num.
0 → 0, L
1 → 1, L
…
9 → 9, L
□ → □, L
incr.
□ → □, R
go home
0 → 0, L
1 → 1, L
…
9 → 9, L

1 7 8 0 0

start
to 2nd num.
0 → 0, R
1 → 1, R
...
9 → 9, R
□ → □, R
decr.
done
to 1st num.
0 → 0, L
1 → 1, L
...
9 → 9, L
done
□ → □, L
go home
0 → 0, L
1 → 1, L
...
9 → 9, L
□ → □, R
incr.
1 7 8 0 0

start

to 2nd num.

□ → □, R

decr.

0 → 0, R
1 → 1, R
…
9 → 9, R

done

done

to 1st num.

0 → 0, L
1 → 1, L
…
9 → 9, L

□ → □, L

go home

0 → 0, L
1 → 1, L
…
9 → 9, L

□ → □, R

incr.

														1	7	8		0	0														

start

to 2nd num.

□ → □, R

decr.

0 → 0, R
1 → 1, R
...
9 → 9, R

done

done

to 1st num.

0 → 0, L
1 → 1, L
...
9 → 9, L

□ → □, L

incr.

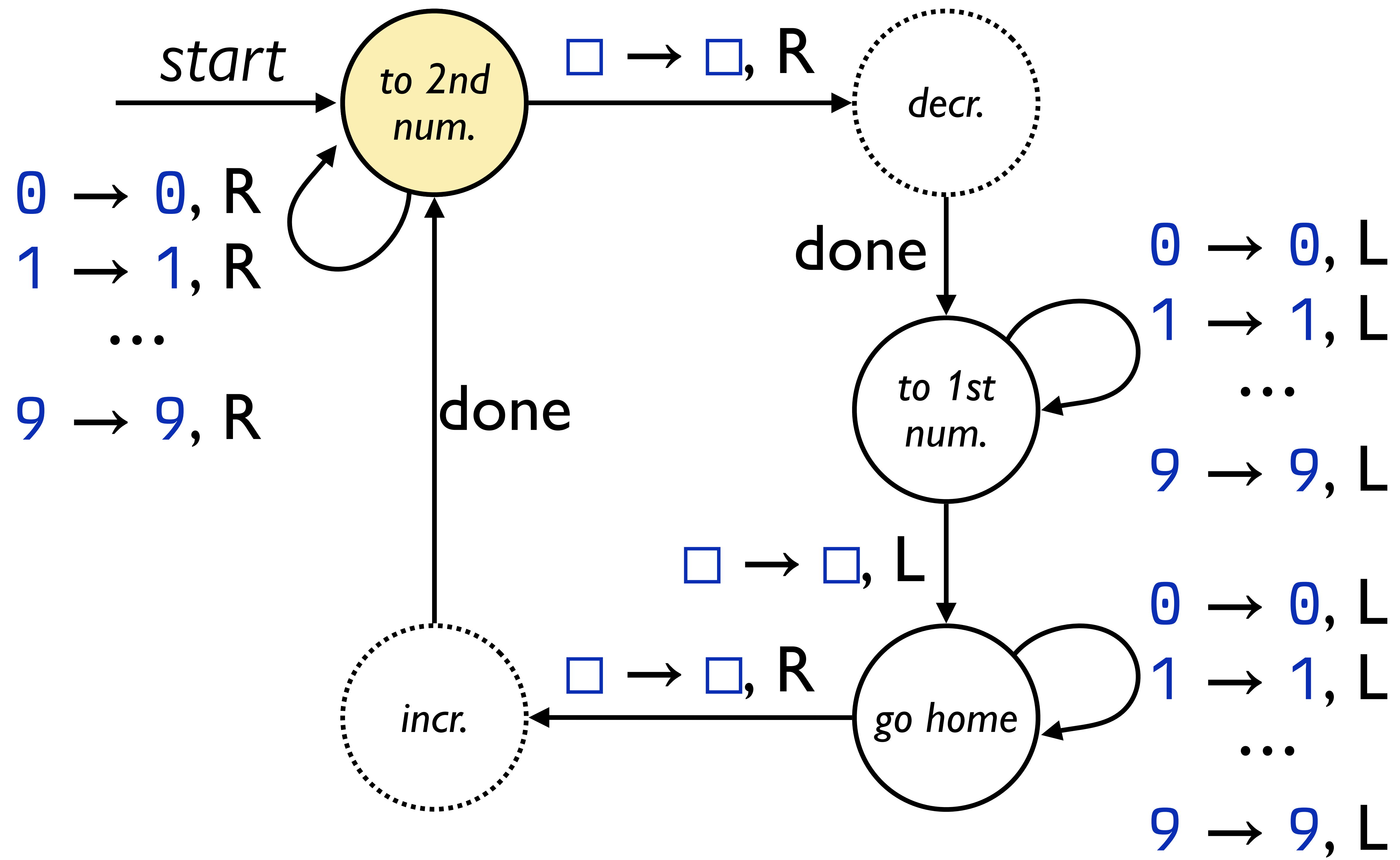
□ → □, R

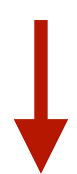
go home

0 → 0, L
1 → 1, L
...
9 → 9, L

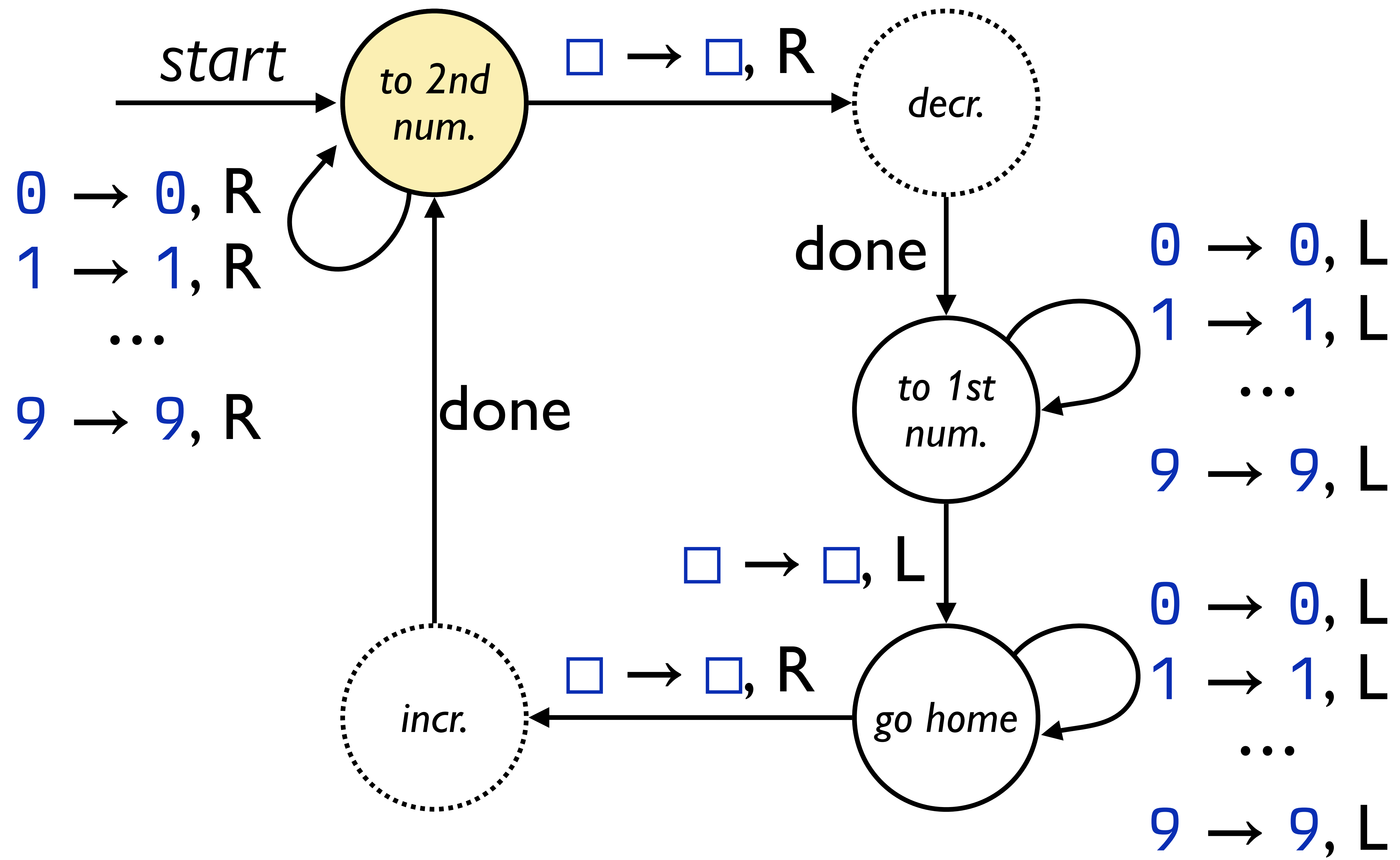
1	7	9		0	0

start
to 2nd num.
0 → 0, R
1 → 1, R
...
9 → 9, R
□ → □, R
decr.
done
to 1st num.
0 → 0, L
1 → 1, L
...
9 → 9, L
□ → □, L
go home
0 → 0, L
1 → 1, L
...
9 → 9, L
□ → □, R
incr.
done
1 7 9 0 0


start
to 2nd num.
□ → □, R
decr.
0 → 0, R
1 → 1, R
…
9 → 9, R
done
done
0 → 0, L
1 → 1, L
…
9 → 9, L
to 1st num.
□ → □, L
0 → 0, L
1 → 1, L
…
9 → 9, L
incr.
□ → □, R
go home



1 7 9 0 0


start
to 2nd num.
□ → □, R
decr.
0 → 0, R
1 → 1, R
...
9 → 9, R
done
done
to 1st num.
0 → 0, L
1 → 1, L
...
9 → 9, L
□ → □, L
incr.
□ → □, R
go home
0 → 0, L
1 → 1, L
...
9 → 9, L

1
7
9
0
0

start
to 2nd num.
□ → □, R
decr.
0 → 0, R
1 → 1, R
...
9 → 9, R
done
done
to 1st num.
0 → 0, L
1 → 1, L
...
9 → 9, L
□ → □, L
incr.
□ → □, R
go home
0 → 0, L
1 → 1, L
...
9 → 9, L
1 7 9 0 0

start

to 2nd num.

□ → □, R

decr.

n = 0

done!

0 → 0, R
1 → 1, R
…
9 → 9, R

done

done

to 1st num.

0 → 0, L
1 → 1, L
…
9 → 9, L

□ → □, L

go home

0 → 0, L
1 → 1, L
…
9 → 9, L

incr.

□ → □, R

1	7	9		0	0

start

to 2nd num.

□ → □, R

decr.

n = 0

done!

0 → 0, R
1 → 1, R
…
9 → 9, R

done

done

to 1st num.

0 → 0, L
1 → 1, L
…
9 → 9, L

□ → □, L

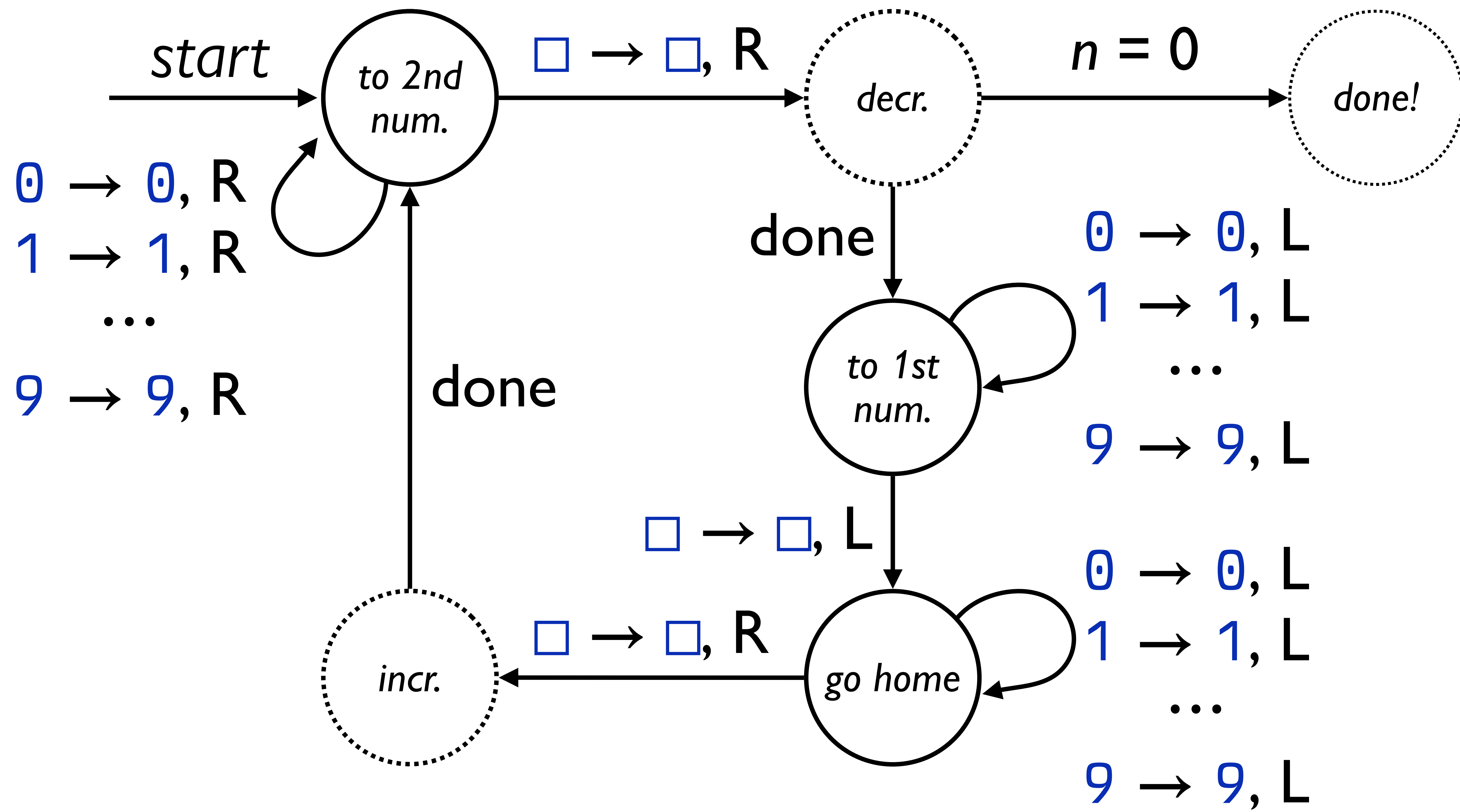
incr.

□ → □, R

go home

0 → 0, L
1 → 1, L
…
9 → 9, L

1	7	9		0	0


start
to 2nd num.
□ → □, R
decr.
n = 0
done!
0 → 0, R
1 → 1, R
…
9 → 9, R
done
done
to 1st num.
0 → 0, L
1 → 1, L
…
9 → 9, L
□ → □, L
□ → □, R
incr.
go home
0 → 0, L
1 → 1, L
…
9 → 9, L

“What problems can we solve with a computer?”

“What problems can we solve with a computer?”

What kind of computer?

Turing machines and computers

A real computer has memory limitations: You have a finite amount of RAM, a finite amount of disk space, etc.

However, as computers get more and more powerful, the amount of memory available keeps increasing.

An *idealized computer* is just like a regular computer, but with unlimited RAM and disk space.

THEOREM Turing machines are equal in power to idealized computers.

THEOREM Turing machines are equal in power to idealized computers.

CLAIM 1 Idealized computers can simulate Turing machines.

CLAIM 2 Turing machines can simulate idealized computers.

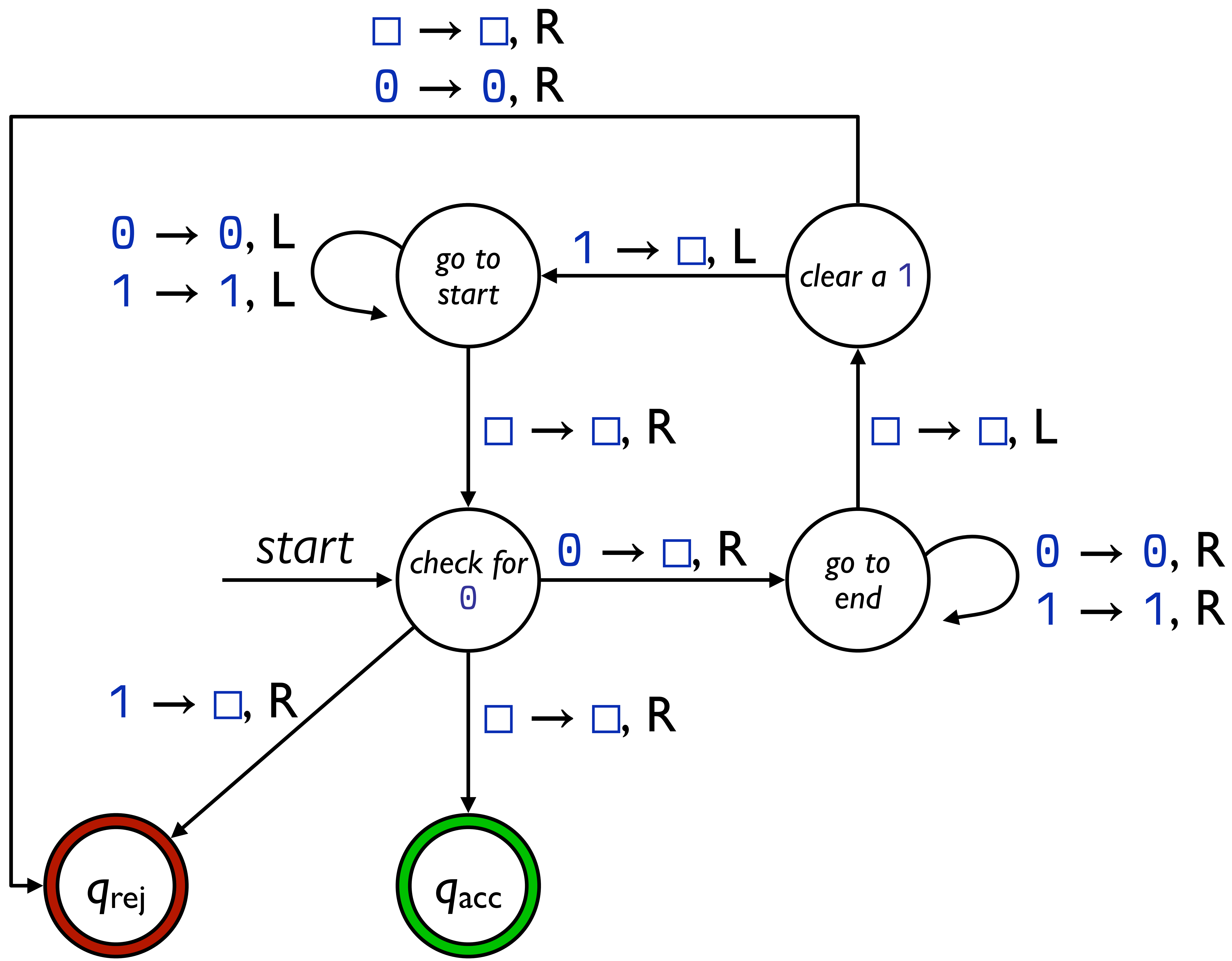
Key idea: *Two models of computation are equally powerful if they can simulate each other.*

THEOREM Turing machines are equal in power to idealized computers.

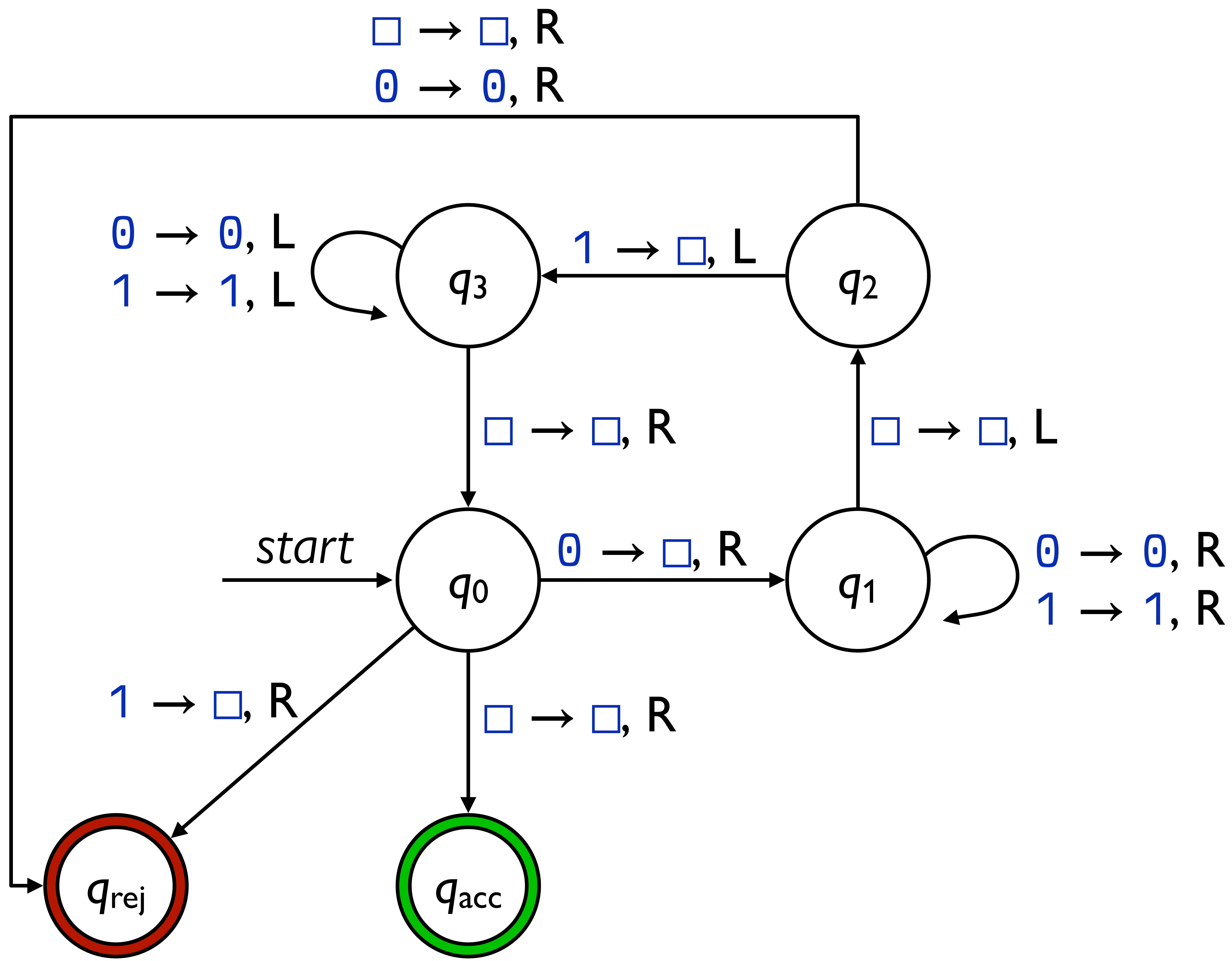
CLAIM 1 Idealized computers can simulate Turing machines.

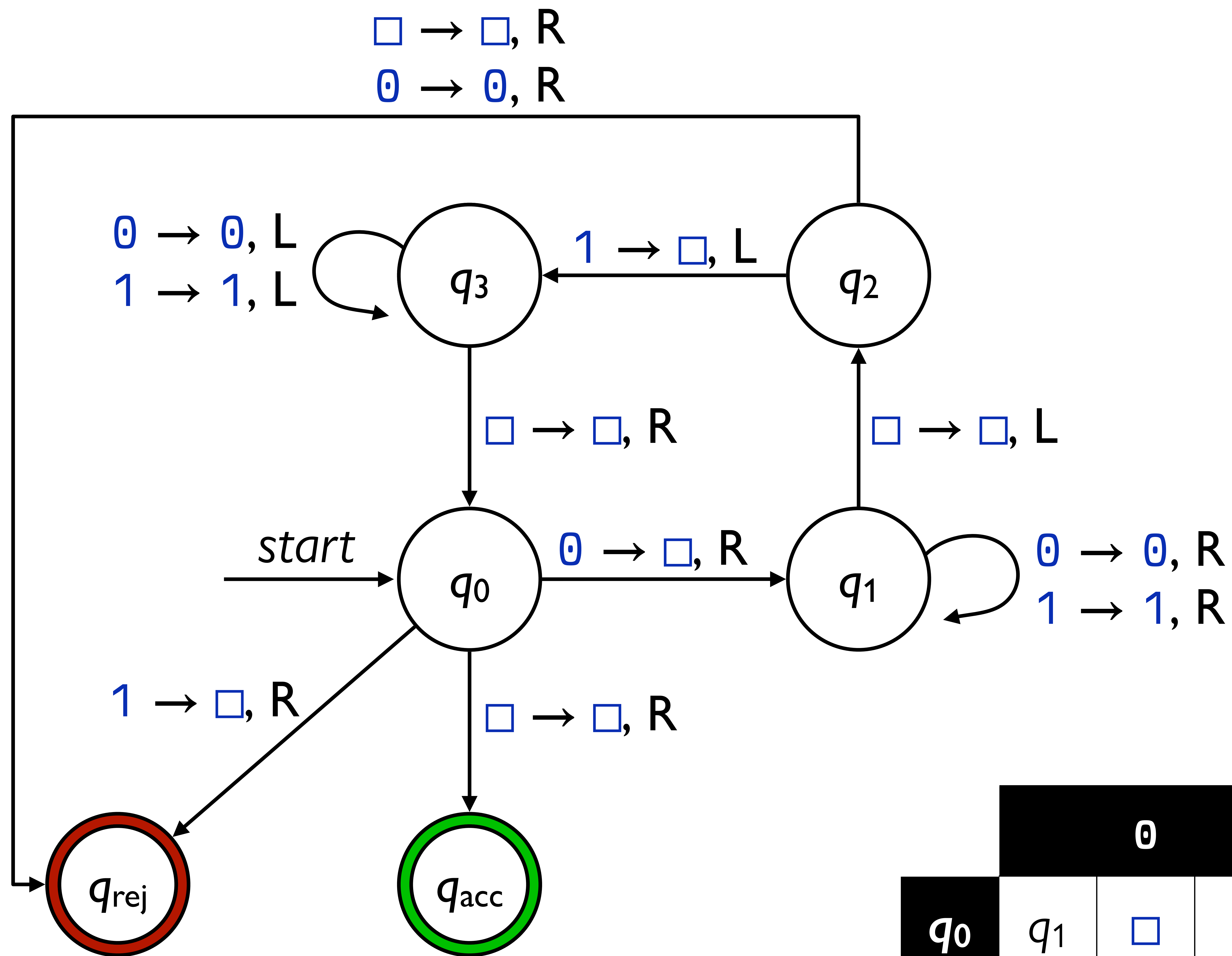
CLAIM 2 Turing machines can simulate idealized computers.

Key idea: *Two models of computation are equally powerful if they can simulate each other.*



□ → □, R
0 → 0, R
0 → 0, L
1 → 1, L
go to start
1 → □, L
clear a 1
□ → □, R
□ → □, L
start
check for 0
0 → □, R
go to end
0 → 0, R
1 → 1, R
1 → □, R
□ → □, R
qrej
qacc


□ → □, R
0 → 0, R
0 → 0, L
1 → 1, L
q3
1 → □, L
q2
□ → □, R
□ → □, L
start
q0
0 → □, R
q1
0 → 0, R
1 → 1, R
1 → □, R
□ → □, R
qrej
qacc



The Turing machine's finite-state control can be encoded as a table, making it easy for a computer to look up transitions

	0			1			□		
q_0	q_1	□	R	q_{rej}	□	R	q_{acc}	□	R
q_1	q_1	0	R	q_1	1	R	q_2	□	L
q_2	q_{rej}	0	R	q_3	□	L	q_{rej}	□	R
q_3	q_3	0	L	q_3	1	L	q_0	□	R

Simulating a Turing machine

To simulate a Turing machine, the computer would need to be able to keep track of

- the finite-state control,
- the position of the tape head, and
- the tape contents.

The tape contents are infinite, but we only need to store the part of the tape that's been read from or written to so far.

														1	7	9		0	0														

Simulating a Turing machine

To simulate a Turing machine, the computer would need to be able to keep track of

- the finite-state control,
- the position of the tape head, and
- the tape contents.

The tape contents are infinite, but we only need to store the part of the tape that's been read from or written to so far.

														1	7	9		0	0														

Simulating a Turing machine

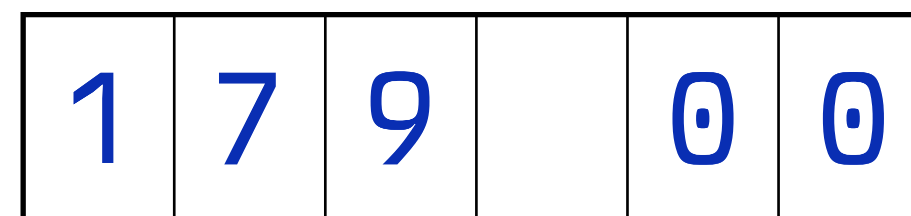
To simulate a Turing machine, the computer would need to be able to keep track of

the finite-state control,

the position of the tape head, and

the tape contents.

The tape contents are infinite, but we only need to store the part of the tape that's been read from or written to so far.



THEOREM Turing machines are equal in power to idealized computers.

CLAIM 1 Idealized computers can simulate Turing machines.

CLAIM 2 Turing machines can simulate idealized computers.

Key idea:
Two models of computation are equally powerful if they can simulate each other.

We've seen that Turing machines can

- implement loops
- make function calls (i.e., use subroutines)
- keep track of natural numbers (in unary or in decimal)
- perform elementary arithmetic
- perform `if–else` tests (i.e., take different transitions based on different cases)
- Maintain variables using different parts of the tape (e.g., the two numbers being added)

A CMPU 224 perspective

Internally, real computers execute by using basic operations like

- simple arithmetic,
- memory reads and writes,
- branches and jumps,
- register operations,
- etc.

Each of these are simple enough that they could be simulated by a Turing machine.

A Turing machine is powerful enough to simulate any computer program that gets an input, processes that input, then returns some result.

The resulting TM might be very large, very slow, or both, but it would still faithfully simulate the computer.

Just how powerful *are* Turing machines?

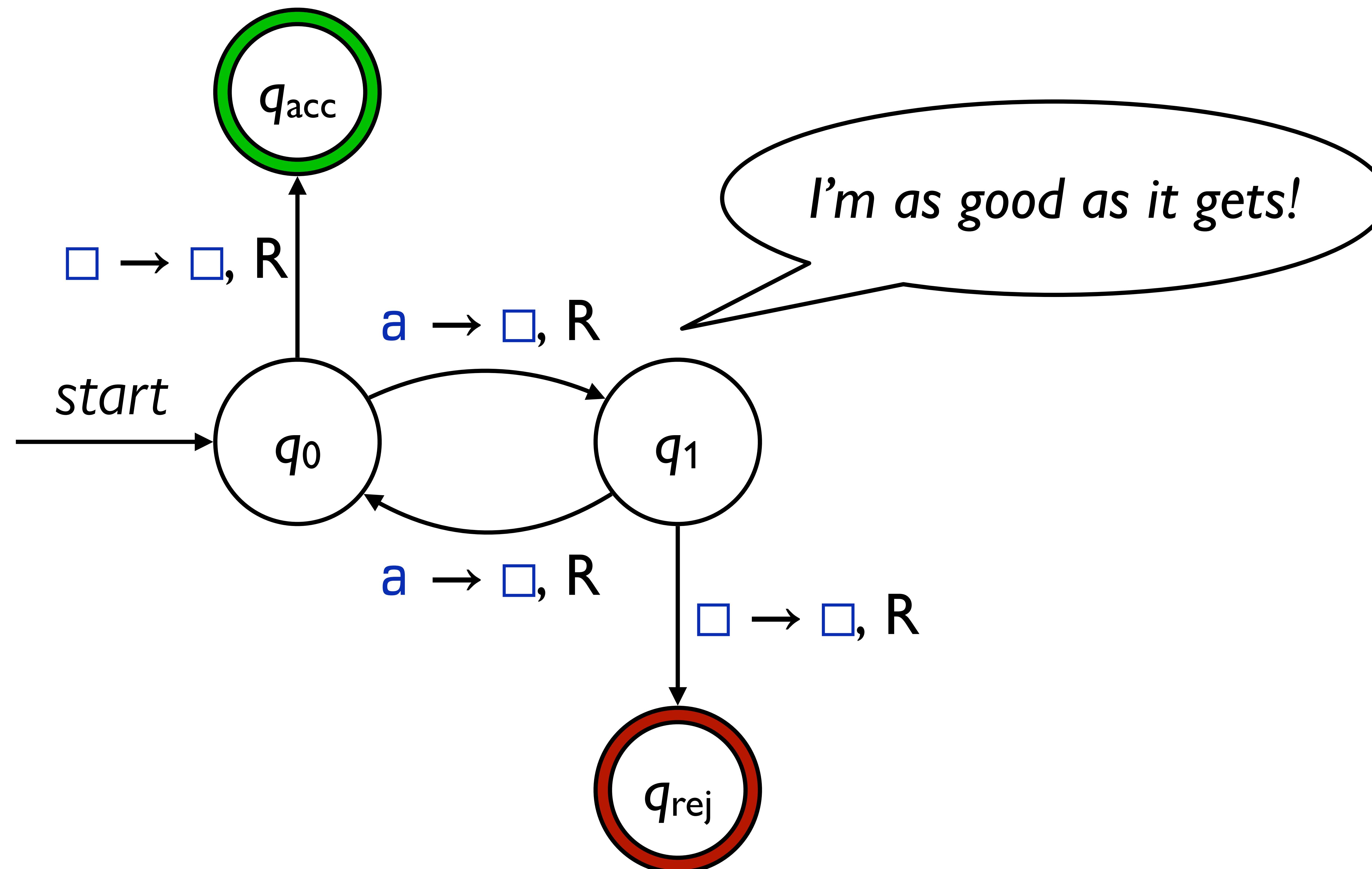
An *effective method of computation* is a form of computation with the following properties:

- The computation consists of a set of steps.
- There are fixed rules governing how one step leads to the next.
- Any computation that yields an answer does so in finitely many steps.
- Any computation that yields an answer always yields the correct answer.

This isn't a formal definition, but it's a set of properties we expect out of a computational system.

The Church–Turing Thesis

Every effective method of computation is either equivalent to or weaker than a Turing machine.



The Church–Turing Thesis

Every effective method of computation is either equivalent to or weaker than a Turing machine.

The Church–Turing Thesis

Every effective method of computation is either equivalent to or weaker than a Turing machine.

The Church–Turing Thesis

Why is this a "thesis" and not a theorem?

Every effective method of computation is either equivalent to or weaker than a Turing machine.

First justification

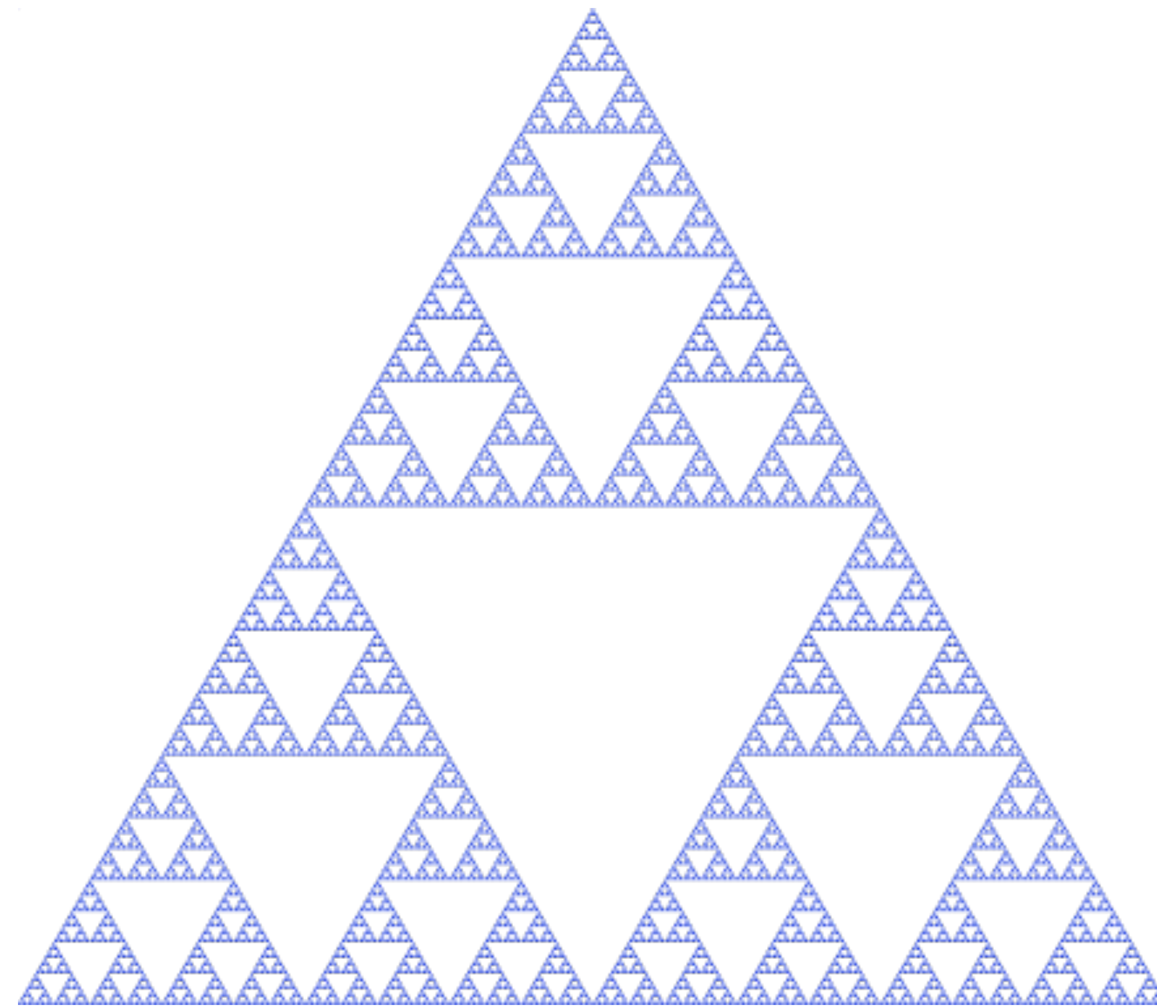
He proposed Turing machines as a model of what humans do when they compute.

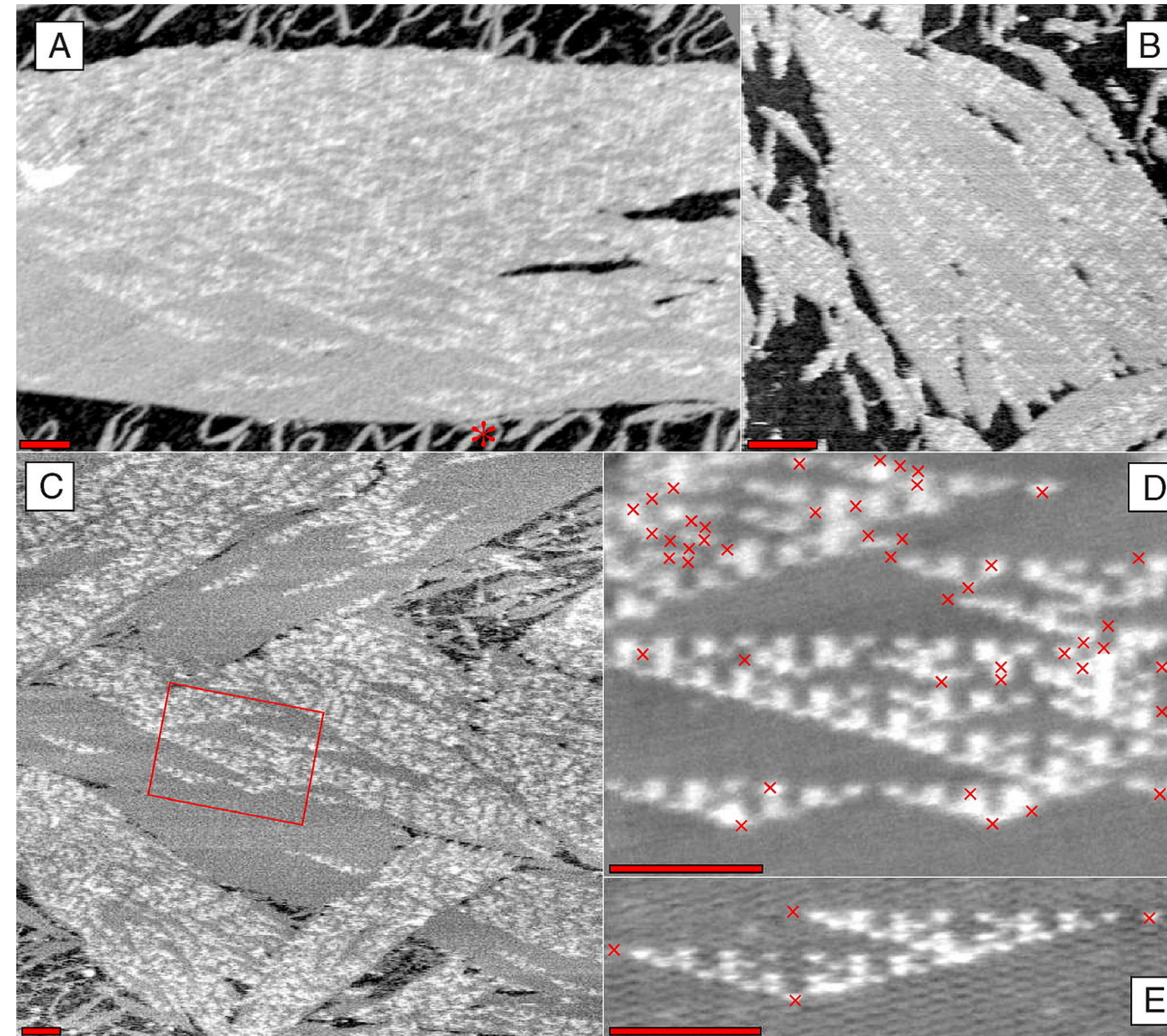
- He imagined a computing person in an idealized scenario:
  - They have an infinitely long paper tape, divided into squares.
  - They can write one symbol in each square, and the number of possible symbols is finite (e.g., 0 to 9).
  - They can only look at a finite number of squares at a time.
  - They can only move a finite distance at a time.
  - They have only a finite number of "states of mind."

Basically this is an appeal to intuition that when people compute, this is what they do.

We can consider many other reasonable models of computation:

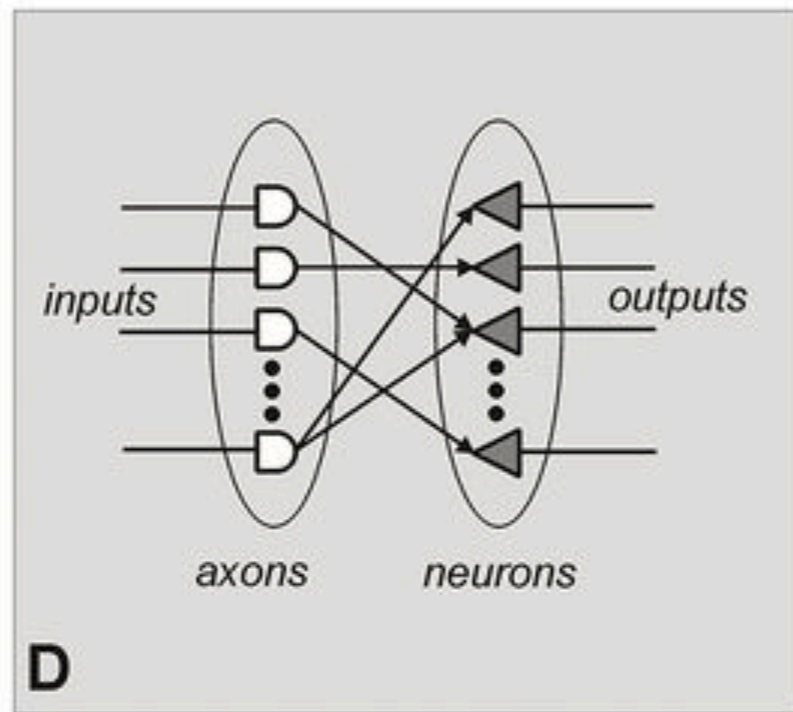
DNA computing, neural networks, quantum computing…

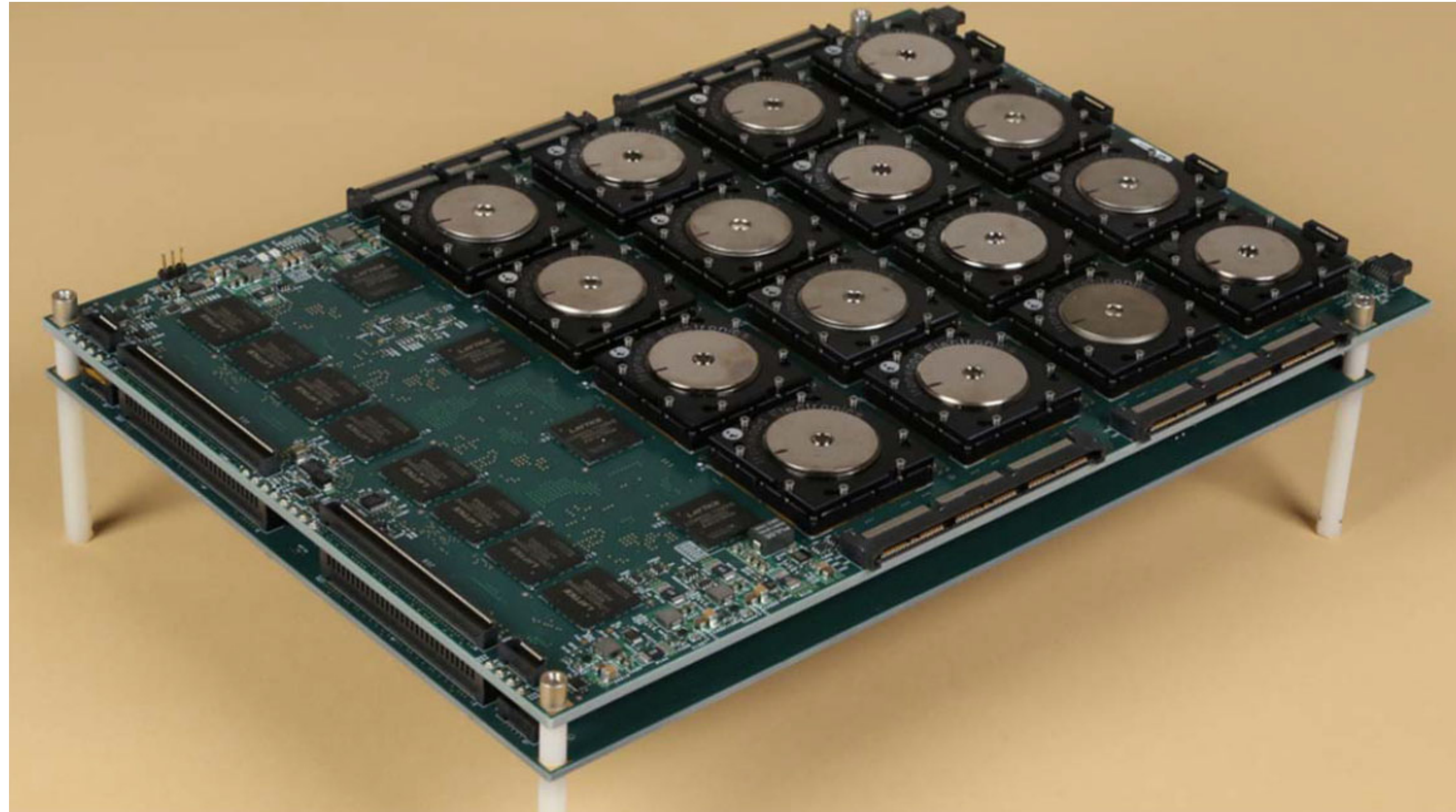




DNA arrays displaying the Sierpinski gasket, a fractal, generated by cellular automata.
Rothemund et al., 2004

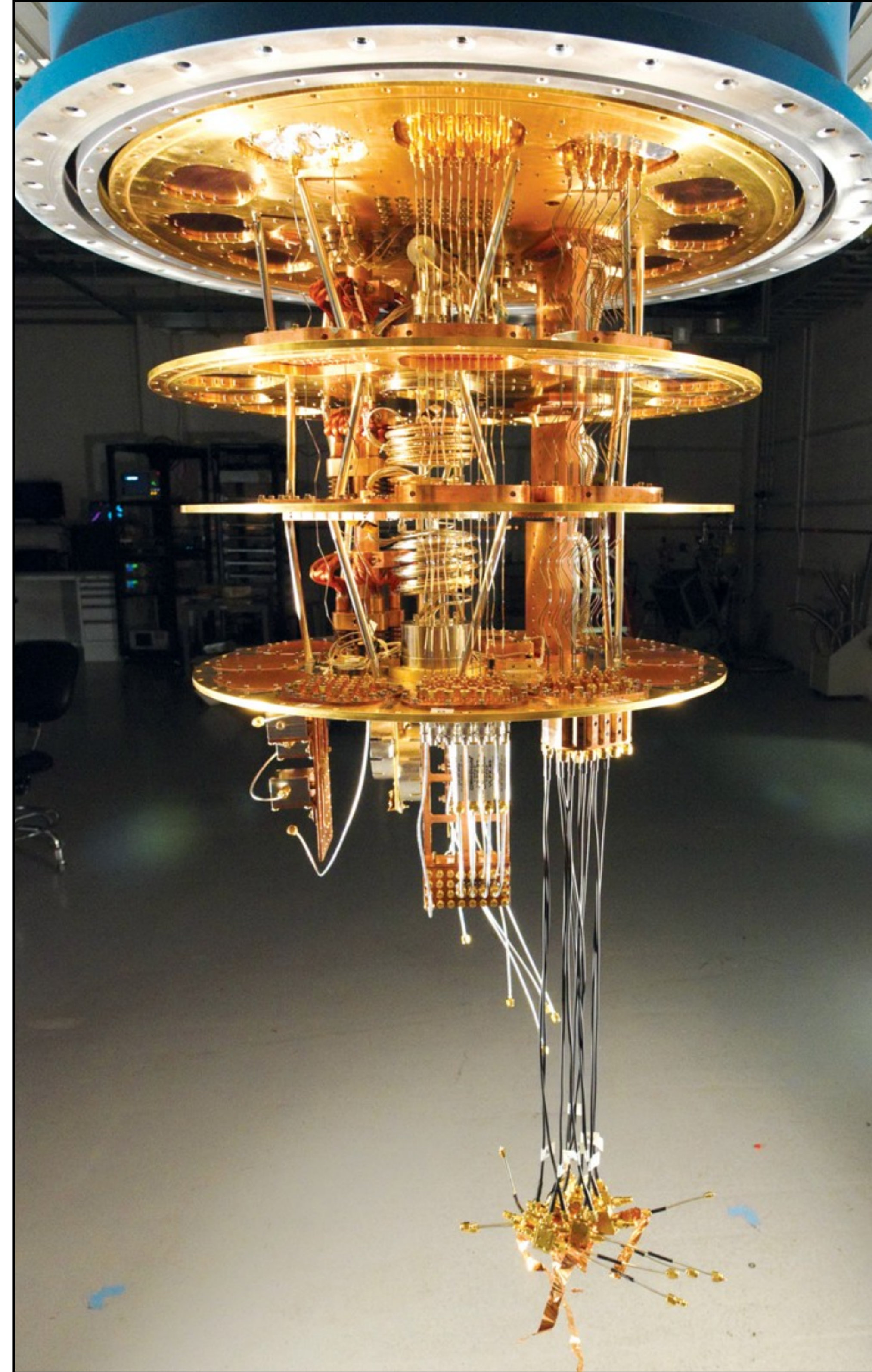
We can consider many other reasonable models of computation: *DNA computing*, neural networks, quantum computing…





Biologically inspired hardware for artificial neural networks created by IBM, 2014

We can consider many other reasonable models of computation: DNA computing, ***neural networks***, quantum computing…



The cooling system for an early-stage quantum computer at Google.
Photo by Erik Lucero

We can consider many other reasonable models of computation:
DNA computing, neural networks, *quantum computing*…

We can also consider many variations on Turing machines, with multiple tapes, nondeterminism, etc.

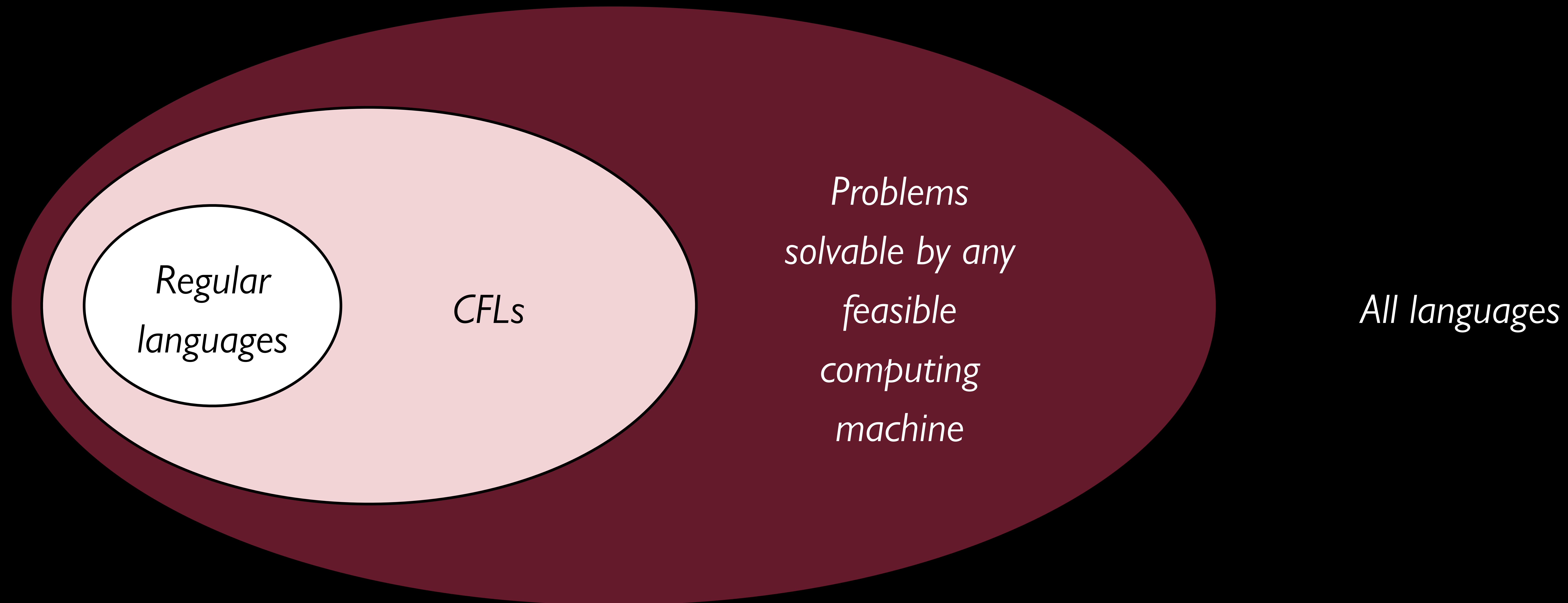
These might be (much) faster for some computations, but experience has confirmed that every such model can be simulated by a standard Turing machine.

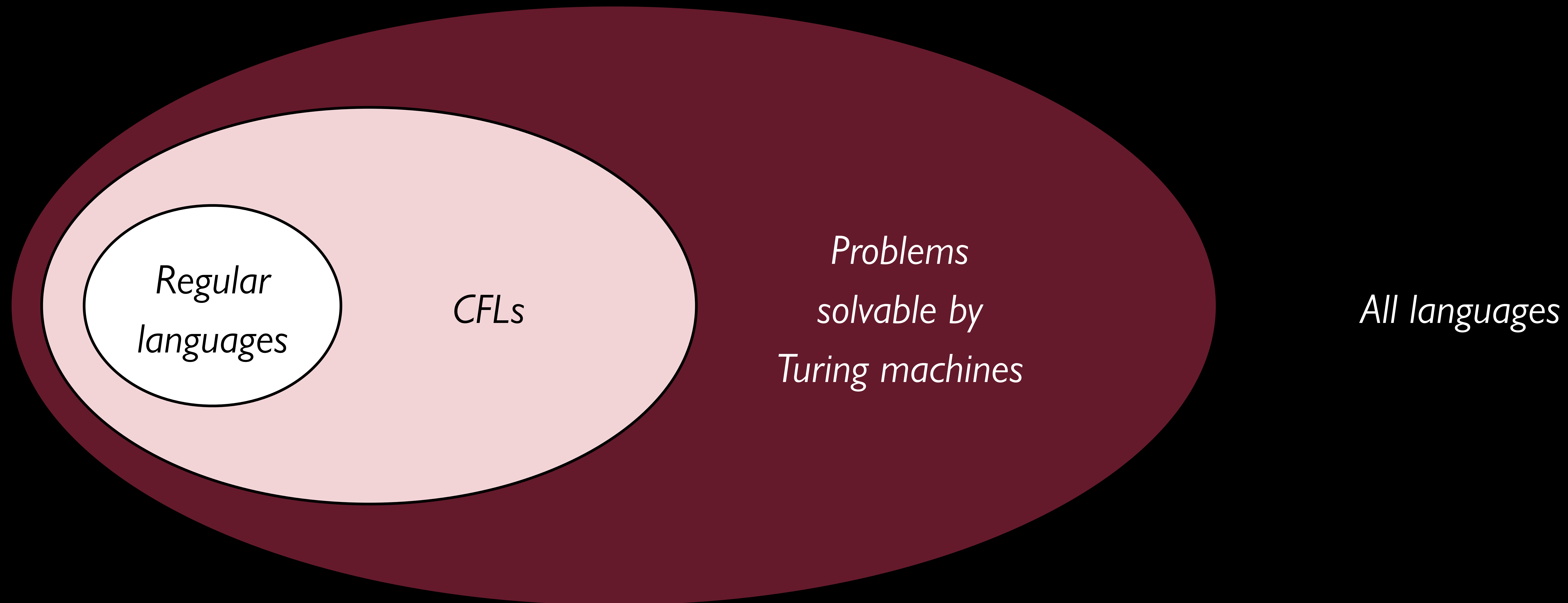
They do not change what is *computable*.

Second justification

Many other proposals have been made for models of computability, which have turned out to be equivalent to Turing machines, including

- Lambda calculus,
- Partial recursive functions,
- Unrestricted grammars, and
- Many more things that have turned out to be accidentally Turing-complete.


Regular languages
CFLs
Problems solvable by any feasible computing machine
All languages


Regular languages
CFLs
Problems solvable by Turing machines
All languages

Turing machines and computation

Because Turing machines have the same computational power as (idealized) regular computers, we can, essentially, reason about Turing machines by reasoning about actual computer programs.

Going forward, we'll switch back and forth between Turing machines and computer programs (pseudocode) based on what's most appropriate.

“What problems can we solve with a computer?”

What does it mean to “solve” a problem?

The Hailstone Sequence

Consider the following procedure, starting with some $n \in \mathbb{N}$, where $n > 0$:

```
def hailstone(n):
    while n != 1:
        if is_even(n):
            n = n / 2
        else:
            n = 3n + 1
```

The Hailstone Sequence

Consider the following procedure, starting with some $n \in \mathbb{N}$, where $n > 0$:

```
def hailstone(n):
    while n != 1:
        if is_even(n):
            n = n / 2
        else:
            n = 3n + 1
```

Does this function always terminate?

52
34
40
26
20
17
11
13
16
10
8
5
4
2
1
def hailstone(n):
while n != 1:
if is_even(n):
n = n / 2
else:
n = 3n + 1

The Hailstone Sequence

Let $\Sigma = \{1\}$ and consider the language

$$L = \{1^n \mid n > 0 \text{ and the hailstone sequence terminates for } n\}.$$

Could we build a Turing machine for L?

The Hailstone Sequence

We can build a Turing machine that works much like our pseudocode:

If the input is ε, reject.

While the string is not 1:

If the input has even length, halve the length of the string.

If the input has odd length, triple the length of the string and append a 1.

Accept

Does this Turing machine accept all nonempty strings?

It is *unknown* whether this process will terminate for all natural numbers.

In other words, no one knows whether the TM described in the previous slides will always stop running!

The conjecture (unproven claim) that the hailstone sequence always terminates is called the *Collatz Conjecture*.

“Mathematics may not be ready for such problems.”

Paul Erdős

Unlike finite automata, which automatically halt after reading the input, Turing machines keep running until they explicitly enter an accept or reject state.

As such, it's possible for a Turing machine to run forever without accepting or rejecting.

If a Turing machine might run forever, how do we formally define what it means to "build a Turing machine for a language"?

What implications does this have for problem-solving?

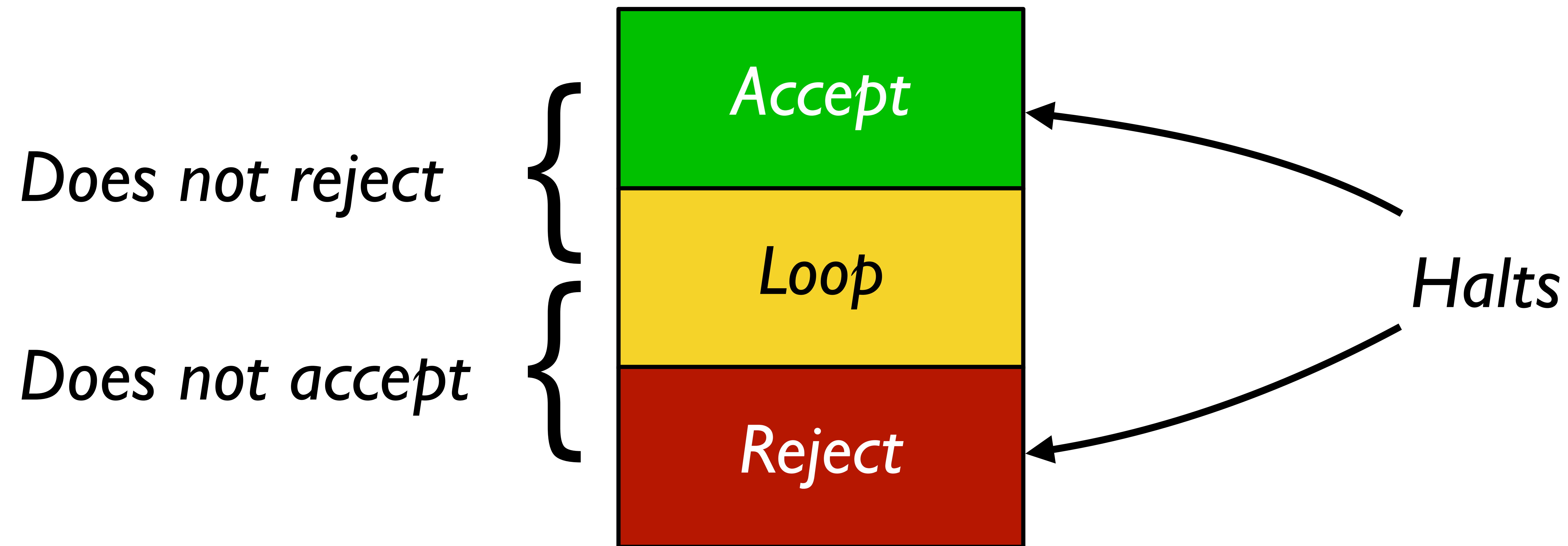
Terminology

Let M be a Turing machine.

M ***accepts*** a string w if it enters the accept state when run on w.

M ***rejects*** a string w if it enters the reject state when run on w.

M ***loops infinitely*** (or just ***loops***) on a string w if, when run on w, it never enters the accept or reject states.



M *does not accept* w if it either rejects w or loops infinitely on w.

M *does not reject* w if it either accepts w or loops on w.

M *halts on* w if it accepts w or rejects w.

The *language of a Turing machine* M is

$$L(M) = \{w \in \Sigma^* \mid M \text{ accepts } w\}.$$

If $w \in L(M)$, M accepts w.

If $w \notin L(M)$, M does not accept w.

That is, when M is run on w, either it rejects or it loops forever.

Recognizable languages

A language is called *Turing-recognizable* (or just *recognizable*) if it is the language of *some* Turing machine.

A Turing machine M where $L(M) = L$ is called a *recognizer* for L.

The set of all languages that are Turing-recognizable is called *RE*.

$$L \in \mathbf{RE} \Leftrightarrow L \text{ is recognizable}$$

Does this correspond to what you think it means to "solve a problem"?

The hailstone Turing machine M we saw earlier is a recognizer for the language

$$L = \{1^n \mid n > 0 \text{ and the hailstone sequence terminates for } n\}.$$

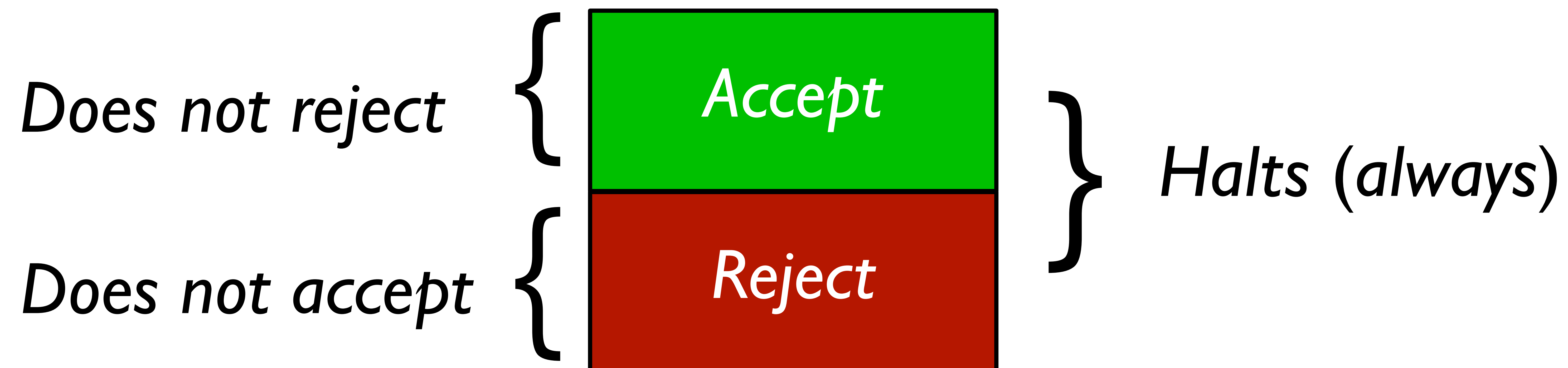
If the sequence does terminate starting at n, then M accepts 1^n.

If the sequence doesn't terminate, then M loops forever on 1^n and never gives an answer.

If you somehow knew the hailstone sequence terminated for n, this machine would (eventually) confirm this. If you didn't know, this machine might not tell you anything.

If a Turing machine M halts on every possible input – i.e., it never goes into an infinite loop – then we call M a *decider*.

For deciders, accepting is the same as not rejecting and rejecting is the same as not accepting:



Decidable languages

A language is called *Turing-decidable* (or just *decidable*) if it is the language of *some* decider.

Equivalently, a language L is Turing-decidable if there is a Turing machine M such that

If $w \in L$, then M accepts w.

If $w \notin L$, then M rejects w.

The set of all languages that are Turing-decidable is called R.

$$L \in \mathbf{R} \Leftrightarrow L \text{ is decidable}$$

The hailstone Turing machine M we saw earlier is a recognizer for the language

$$L = \{1^n \mid n > 0 \text{ and the hailstone sequence terminates for } n\}.$$

We honestly don't know if M is a decider for this language.

If the Collatz Conjecture is true, then M always halts and is a decider for L.

If the Collatz Conjecture is false, then M will loop on some inputs and isn't a decider for L.

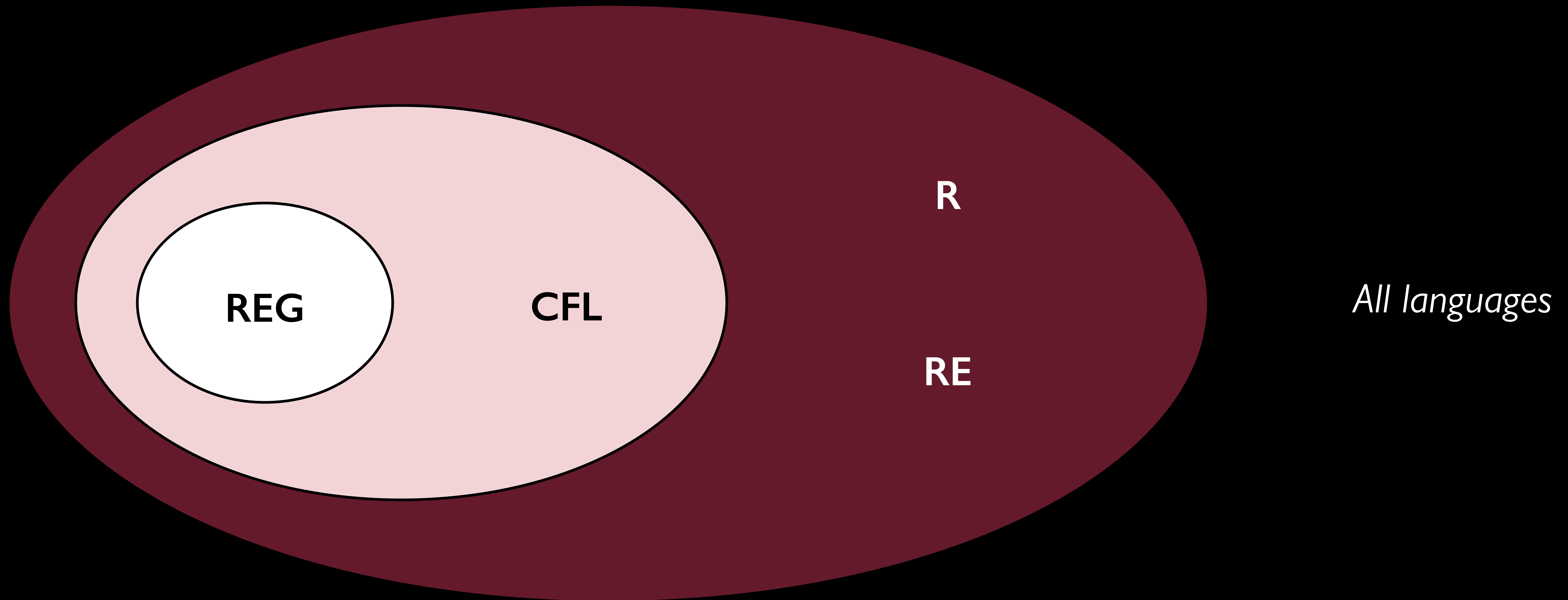
Every decider for some language L is also a recognizer for L.

So, **R** $\subseteq$ **RE**.

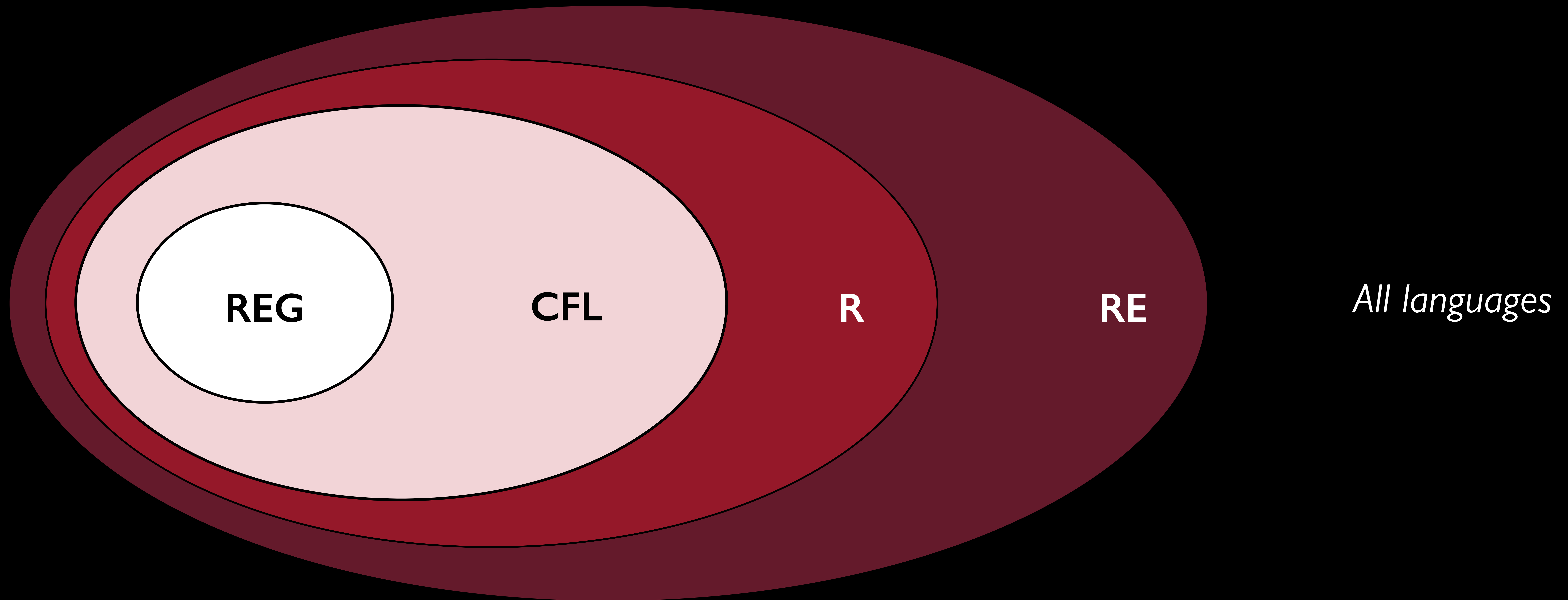
But is **R** = **RE**?

That is, if you can confirm "yes" answers to a problem, can you also *solve* that problem?

Is this right?



Or this?



“What problems can we solve with a computer?”

We haven’t answered this question yet, but we’re getting closer!

Next time

Why languages?

Why do we use languages to model problem-solving?

Emergent properties

Larger phenomena made of smaller parts

Universal machines

A single “most powerful” computer

Self-reference

Programs that ask questions about themselves

Acknowledgments

This lecture incorporates material from:

David Chiang, University of Notre Dame

Keith Schwarz, Stanford University

Michael Sipser, *Introduction to the Theory of Computation*