



中国科学院大学
University of Chinese Academy of Sciences

CS101

Turing Adder

zxu@ict.ac.cn
zhangjialin@ict.ac.cn

Outline

- Week 1: Lab highlights, design Q&A
 - Submit the 1st version design
- Week 2: Team discussion, peer review
 - Submit the 2nd version design
- Week 3: Team presentation to the class
 - Submit project report

First week

Week1 Outline

- Introduction of Turing Machine
- Experiment Platform
- Examples
- Questions

Objective

- Design three Turing machines to perform,
respectively
 - unary addition
 - 4-bit binary addition
 - arbitrary-bit binary addition

1	1	1	+	1	1	1	=	1	1	1	1	1	1	B	B	B	B	B	B
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19

Unary addition

0	1	1	1	+	0	1	0	1	=	0	1	1	0	0	B	B	B	B	B
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19

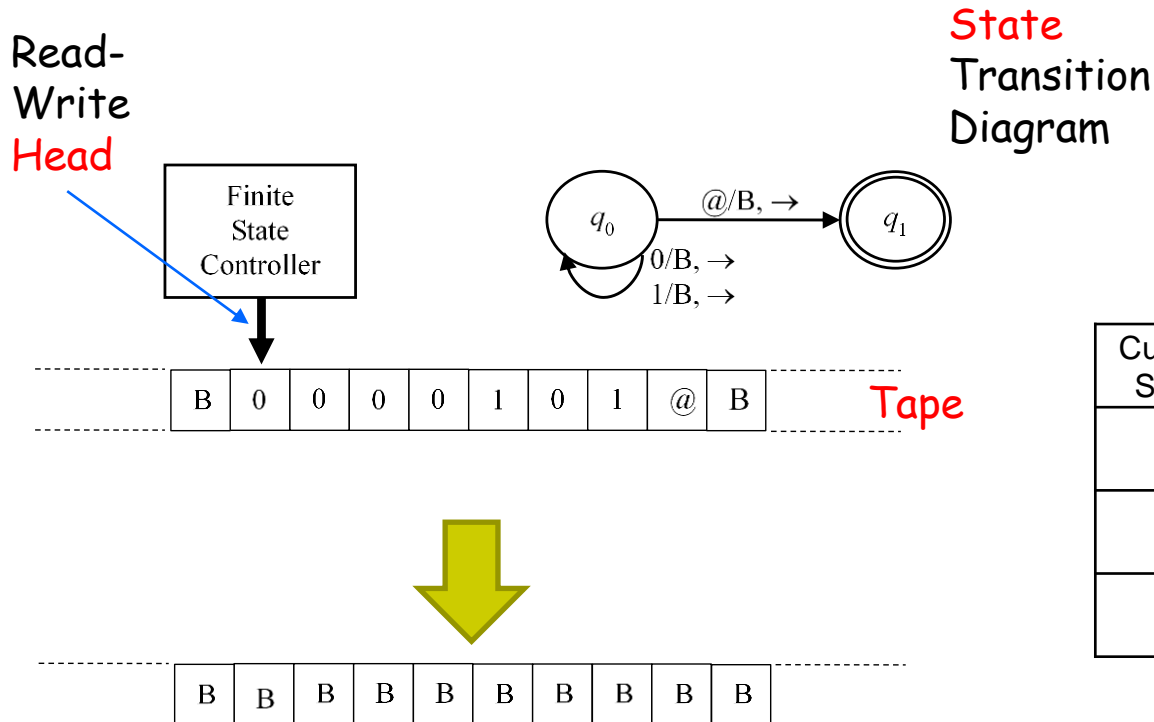
4-bit binary addition

1	1	1	+	1	1	0	1	=	1	0	1	0	0	B	B	B	B	B	B
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19

Arbitrary-bit binary addition

Turing machine

- In addition to the state-transition diagram stored in the Finite State Controller, there is an infinite tape in a Turing machine
- Cleanup: Replace every symbol by a blank (B)
 - The input data string is initially placed in the tape between two blanks



State-Transition Table

Current State	Symbol Read	Symbol to Write	Head Move	Next State
q_0	0	B	→	q_0
q_0	1	B	→	q_0
q_0	@	B	→	q_1 (Halt)

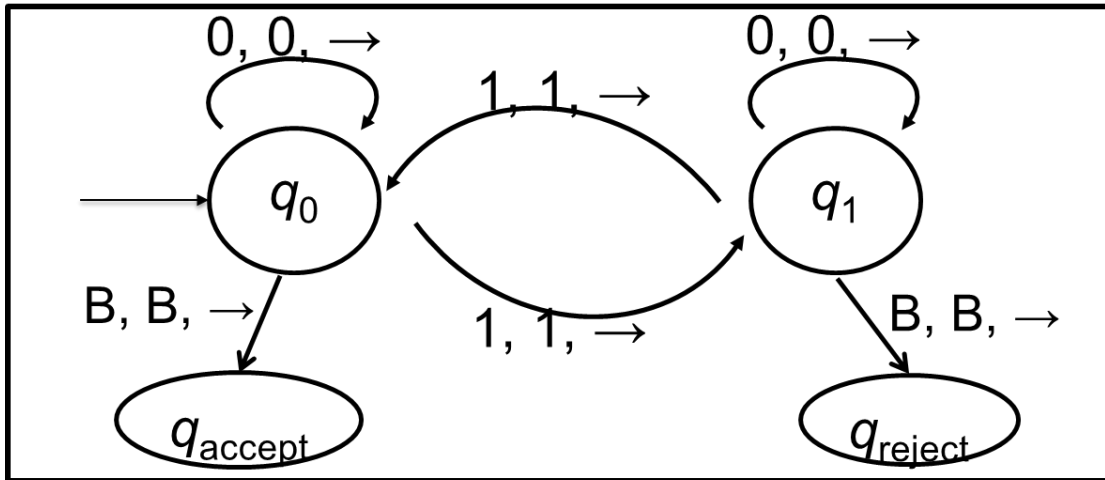
Definition of Turing Machine

- A Turing machine is a 7-tuple $M = \{Q, \Sigma, \Gamma, \delta, q_0, q_{\text{Accept}}, q_{\text{Reject}}\}$.
- Q is a finite, non-empty set of states.
- Σ is a finite, non-empty set of input symbols.
- Γ is a finite, non-empty set of tape symbols. There is a special character $B \in \Gamma$ for the blank symbol. We require $B \notin \Sigma$ and $\Sigma \subset \Gamma$.
- $\delta: (Q - \{q_{\text{Accept}}, q_{\text{Reject}}\}) \times \Gamma \rightarrow Q \times \Gamma \times \{\rightarrow, \leftarrow\}$ is the transition function.
- $q_0 \in Q$ is the initial state.
- $q_{\text{Accept}} \in Q$ is the accept state.
- $q_{\text{Reject}} \in Q$ is the reject state.

Simple Example

- Goal: Judge parity of the number of “0” in a “01” string
- State Set $Q = \{q_0, q_1, q_{\text{accept}}, q_{\text{reject}}\}$
- Input Symbols Set: $\Sigma = \{0, 1\}$
- Tape Symbols Set: $\Gamma = \{0, 1, B\}$
- Transition function: δ (next slide)
- Initial state: q_0
- Accept state: q_{accept}
- Reject state: q_{reject}

Simple Example -- Transition Function



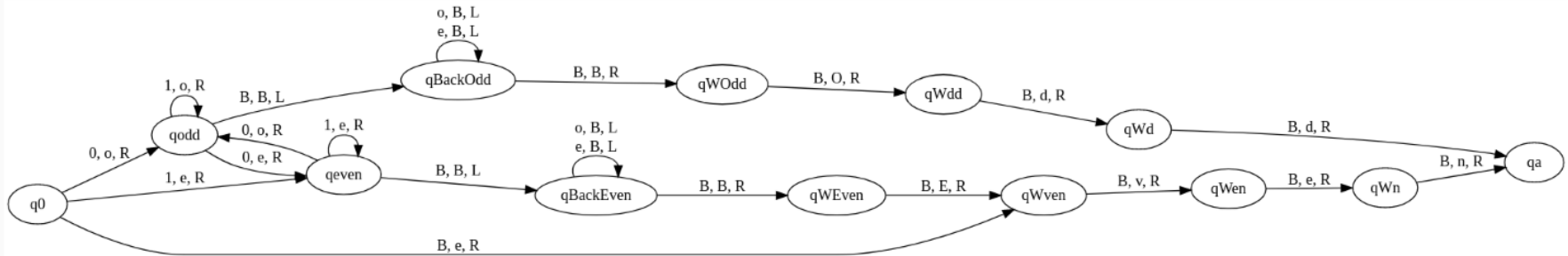
Current State	Symbol Read	Symbol to Write	Head Move	Next State
q_0	0	0	\rightarrow	q_0
q_0	1	1	\rightarrow	q_1
q_0	B	B	\rightarrow	q_{accept}
q_1	0	0	\rightarrow	q_1
q_1	1	1	\rightarrow	q_0
q_1	B	B	\rightarrow	q_{reject}

Week1 Outline

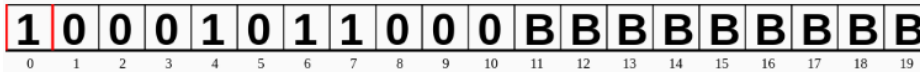
- Introduction of Turing Machine
- Experiment Platform
- Examples
- Questions

Experiment Platform

State Transition Diagram



Execution Animation



Current Step: 0 Current Rule: Next Rule:

```
1 q0,0,o,R,qodd
2 q0,1,e,R,qeven
3 q0,B,e,R,qWven
4 qodd,0,e,R,qeven
5 qodd,1,o,R,qodd
6 qodd,B,B,L,qBackOdd
7 qeven,0,o,R,qodd
8 qeven,1,e,R,qeven
9 qeven,B,B,L,qBackEven
10 qWven,B,v,R,qWen
11 qWen,B,e,R,qWn
12 qWn,B,n,R,qa
13 qBackOdd,o,B,L,qBackOdd
14 qBackOdd,e,B,L,qBackOdd
15 qBackOdd,B,B,R,qWOdd
16 qBackEven,o,B,L,qBackEven
17 qBackEven,e,B,L,qBackEven
18 qBackEven,B,B,R,qWEven
19 qWOdd,B,O,R,qWdd
20 qWdd,B,d,R,qWd
21 qWd,B,d,R,qa
22 qWEven,B,E,R,qWven
```

Submit Recover

10001011000

Set Continue Debug Finish << Speed: 1.00 >> Get State Transition Diagram Test

Experiment Platform -- Overview

- Property
 - The input is put on the tape, and other square are letter B
 - Infinite for both side of tape
 - Header is initially on the far-left side of the input string
- Buttons
 - “Set”: Initialize Turing machine, including import rules and tape data initialization
 - “Pause”/“Continue”: Pause or continue simulation
 - “Debug”: Invalid when in continue state, otherwise execute one step
 - “Finish”: Skip Simulation and display the result of the Turing machine
 - “<< ” : Slow down the simulation
 - “>> ”: Speed up the simulation
 - “Get State Transition Diagram”: As the name suggests
 - “Test”: See if you can pass a test point
- Note
 - If the simulation exceeds 10,000 steps on a certain input, we believe that the Turing machine will not terminate on this input and throw a “no terminate” error.

Experiment Platform -- How to Use

- Only state-transition table is needed to submit
 - Multiple lines
 - Each line only include one state-transition rule, in the form of:
State, Symbol Read, Symbol to Write, Head Move, Next State
 - Note
 - State / Next State: a string. Specify **q0** as the starting state and **qa** as the stop state.
 - Symbol Read: a visible ASCII character, represent the letter that head read.
 - Symbol to Write: a visible ASCII character, represent the letter that head write.
 - Head Move: One of {L, R}, where L means the head moves one space to the left, R means the head moves one space to the right.
 - **Case sensitive.**

How to Use (2)

- Automatic generate and specified tuples
 - State Set: Q is generated according to the state transition table, including all the states that have appeared in the "state"/"next state" column of the state transition table.
 - Tape Symbols Set: Γ is specified as all the visible ASCII codes.
 - Input Symbol Set: Σ is a subset $\Gamma \setminus \{B\}$
 - Initial state: q_0
 - Stop state: q_a

Week1 Outline

- Introduction of Turing Machine
- Experiment Platform
- Examples
- Questions

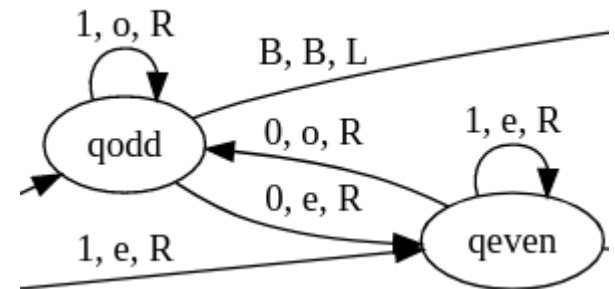
Examples

- Turing machine of parity recognition
 - Judge parity of the number of “0” in a “01” string
 - If the number of “0” is even, then write “Even” on the tape
 - If the number of “0” is odd, then write “Odd” on the tape
- Turing machine of palindrome recognition
 - Judge whether a “01” string is palindrome or not
 - If yes, then write “Y” on the tape
 - Otherwise, write “N”

Parity Recognition -- Key Idea

- Key Idea

- qodd: the number of “0” in the string ever read is odd
- qeven: the number of “0” in the string ever read is even
- Read 0: qodd \rightarrow qeven, qeven \rightarrow qodd
- Read 1: qodd \rightarrow qodd, qeven \rightarrow qeven

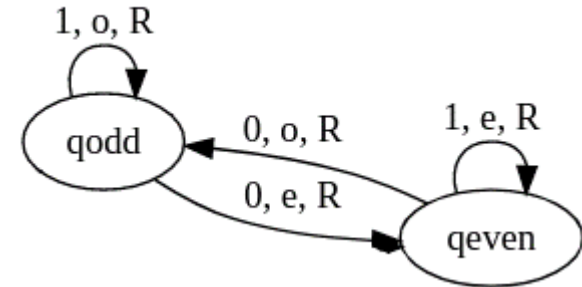


Parity Recognition -- Key Principle

- Key Principle:

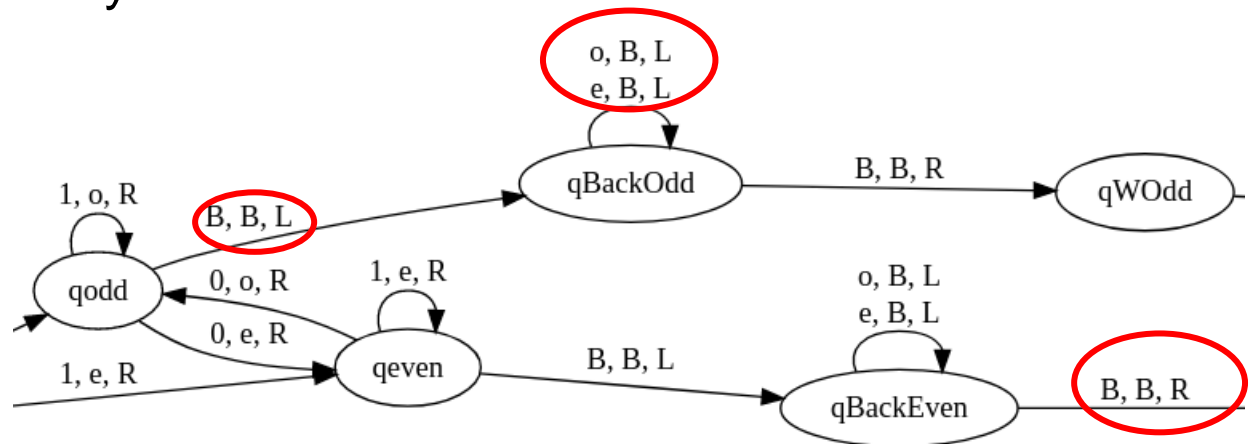
- Loop

- Use four rule to handle infinite input
- NOT infinite rule
 - $q_{\text{odd}}1 \rightarrow q_{\text{even}}1, q_{\text{even}}2 \rightarrow q_{\text{odd}}3, \dots$

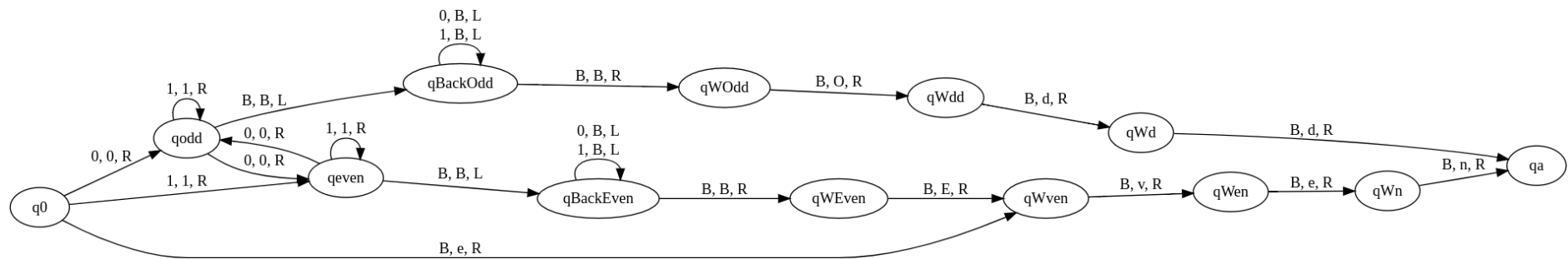


- Able to Move Back

- Read the tape Many Times



Parity Recognition state-transition diagram



Parity Recognition -- state-transition rules

q0,0,o,R,qodd

q0,1,e,R,qeven

q0,B,e,R,qWven

qWven,B,v,R,qWen

qWen,B,e,R,qWn

qWn,B,n,R,qa

qWOdd,B,O,R,qWdd

qWdd,B,d,R,qWd

qWd,B,d,R,qa

qodd,0,e,R,qeven

qodd,1,o,R,qodd

qodd,B,B,L,qBackOdd

qBackOdd,o,B,L,qBackOdd

qBackOdd,e,B,L,qBackOdd

qBackOdd,B,B,R,qWOdd

qWEven,B,E,R,qWven

qeven,0,o,R,qodd

qeven,1,e,R,qeven

qeven,B,B,L,qBackEven

qBackEven,o,B,L,qBackEven

qBackEven,e,B,L,qBackEven

qBackEven,B,B,R,qWEven

Parity Recognition -- Execution Animation



1	0	0	0	1	0	1	1	0	0	0	B	B	B	B	B	B	B	B	B
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19

Current Step: 0 Current Rule: Next Rule:

```
1  q0,0,o,R,qodd
2  q0,1,e,R,qeven
3  q0,B,e,R,qWven
4  qodd,0,e,R,qeven
5  qodd,1,o,R,qodd
6  qodd,B,B,L,qBackOdd
7  qeven,0,o,R,qodd
8  qeven,1,e,R,qeven
9  qeven,B,B,L,qBackEven
10 qWven,B,v,R,qWen
11 qWen,B,e,R,qWn
12 qWn,B,n,R,qa
13 qBackOdd,o,B,L,qBackOdd
14 qBackOdd,e,B,L,qBackOdd
15 qBackOdd,B,B,R,qWOdd
16 qBackEven,o,B,L,qBackEven
17 qBackEven,e,B,L,qBackEven
18 qBackEven,B,B,R,qWEven
19 qWOdd,B,O,R,qWdd
20 qWdd,B,d,R,qWd
21 qWd,B,d,R,qa
22 qWEven,B,E,R,qWven
```

Submit

Recover

10001011000

Set

Continue

Debug

Finish

<<

Speed: 1.00

>>

Get State-Transition Diagram

Test

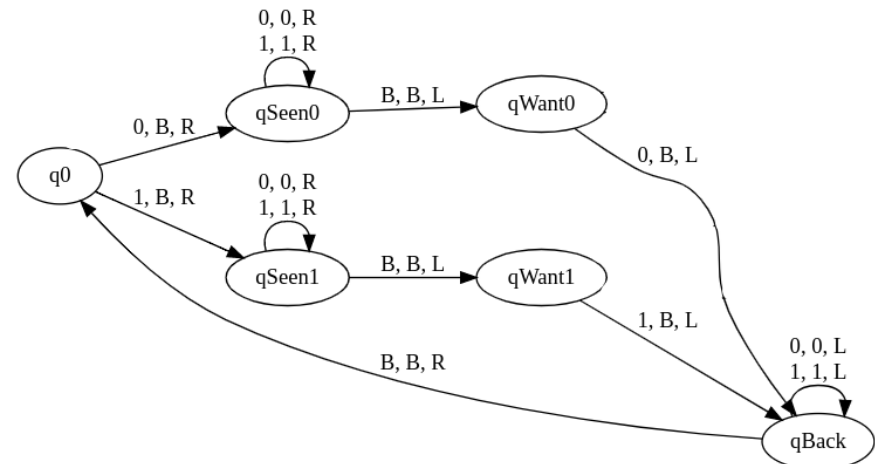
Examples

- Turing machine of parity recognition
 - Judge parity of the number of “0” in a “01” string
 - If the number of “0” is even, then write Even on the tape
 - If the number of “0” is odd, then write Odd on the tape
- Turing machine of palindrome recognition
 - Judge whether a “01” string is palindrome or not
 - If yes, then write “Y” on the tape
 - Otherwise, write “N”

Palindrome Recognition -- Key Idea

● Key Idea

- qSeen0: Found 0 on the far-left of the remaining string
- qSeen1: Found 1 on the far-left of the remaining string
- qWant0: Move left to seek 0 on the far-right of the remaining string
- qWant1: Move left to seek 0 on the far-right of the remaining string
- qBack: find wanted letter on the far-right of the remaining string



Palindrome Recognition -- Key Principle

- Key Principle

- Loop

- Use finite rule to handle infinite input

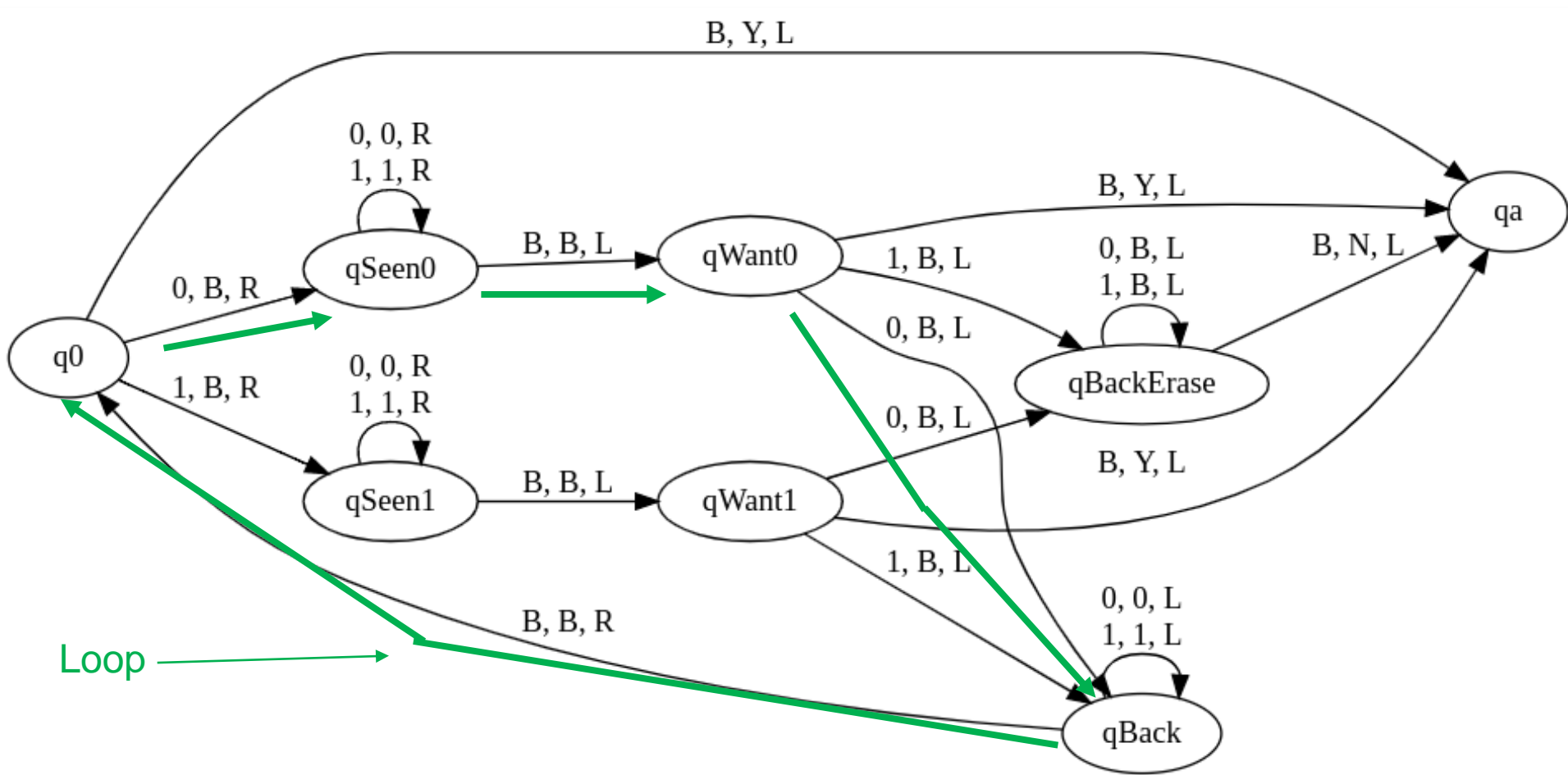
- Move back and Mark

- The tape can be read infinite times
 - Write letters to the tape, and assign some meaning

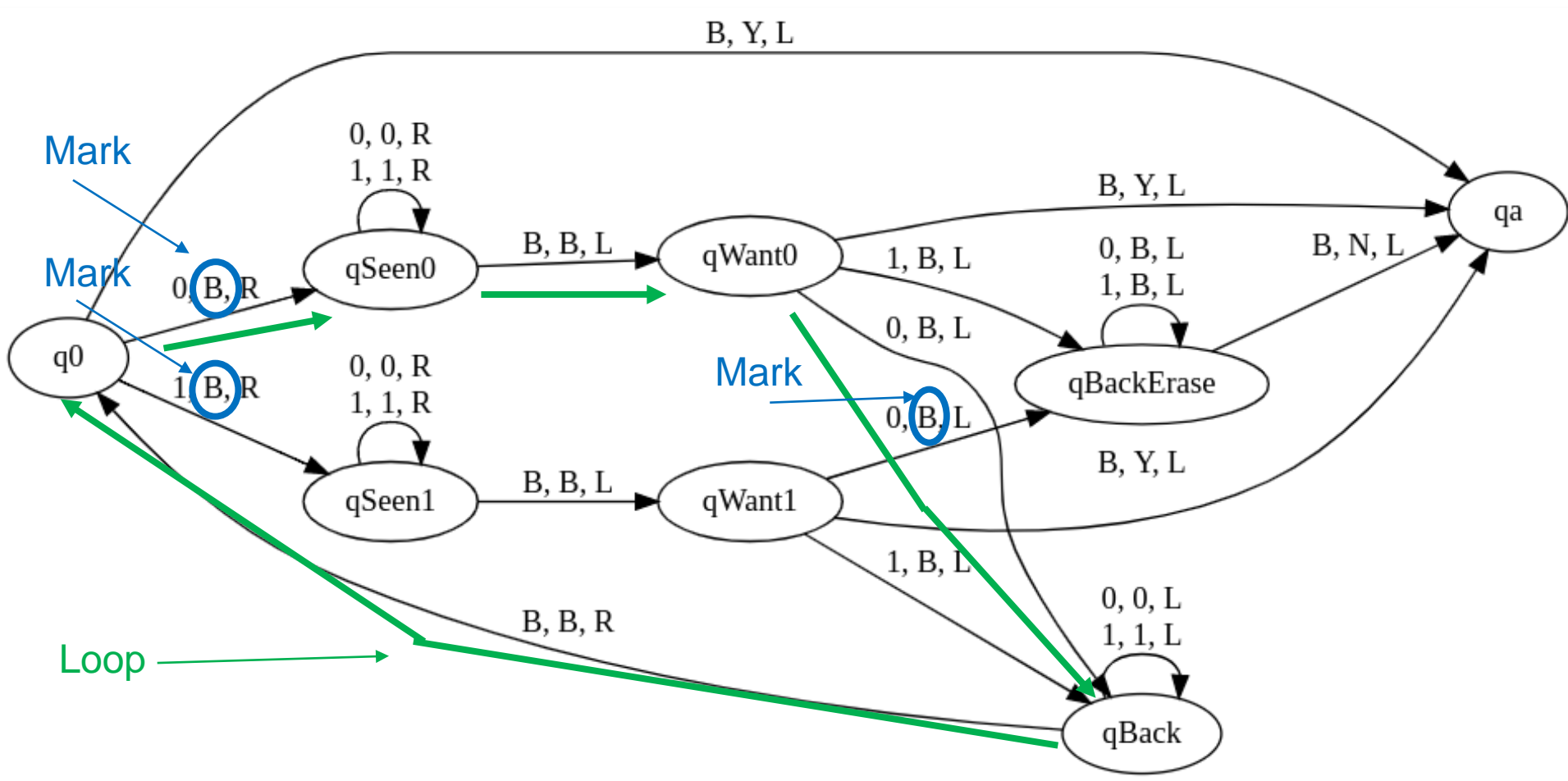
- Implicit conditional judgements

- Use current state and symbol read as a unique condition
 - Result of condition holds:
 - write a symbol and change to the next state

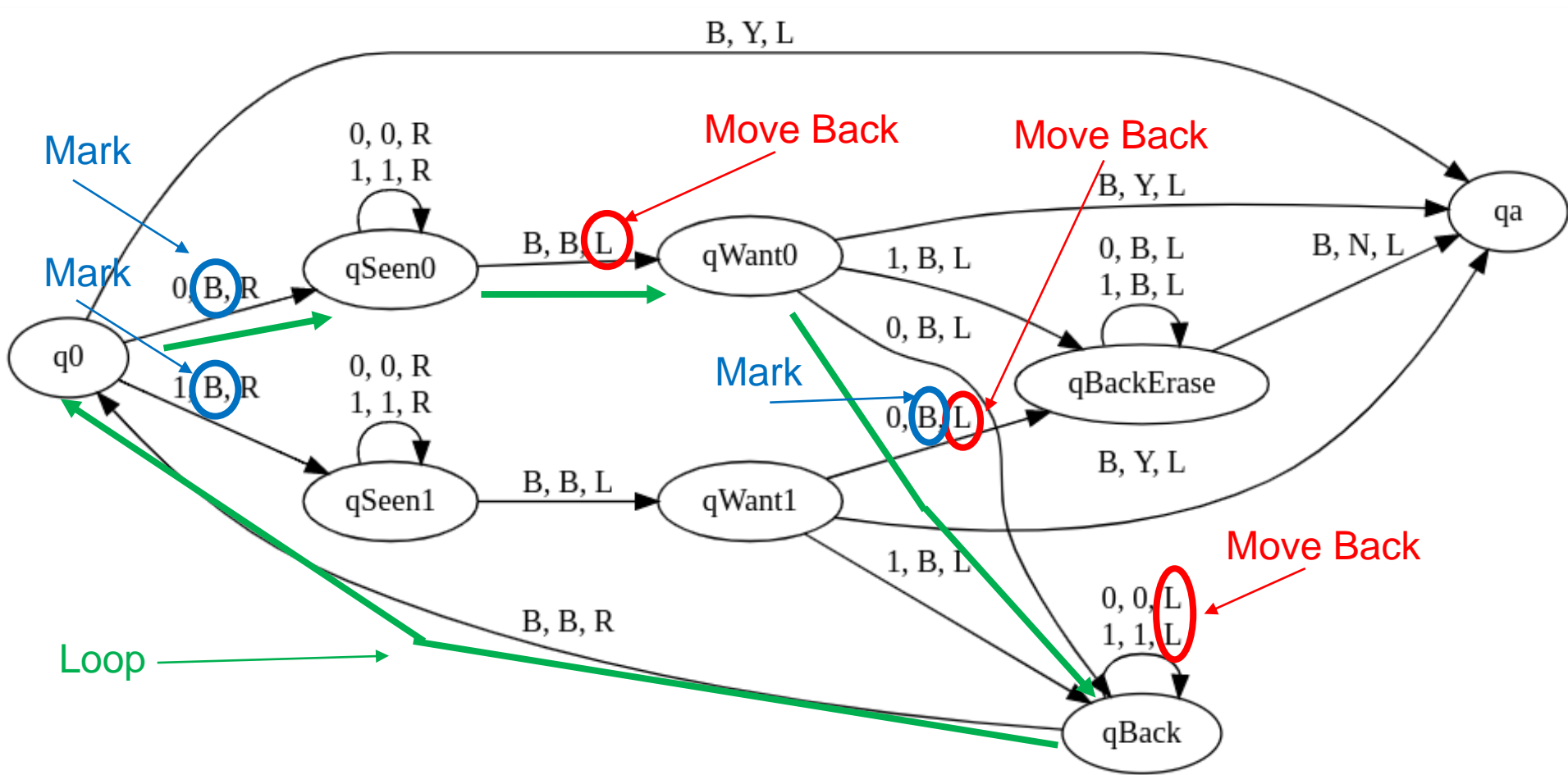
Palindrome Recognition State-Transition Diagram



Palindrome Recognition State-Transition Diagram



Palindrome Recognition State-Transition Diagram



Palindrome Recognition State-Transition Rules

q0,0,B,R,qSeen0	qWant0,0,B,L,qBack	qBackErase,0,B,L,qBackErase
q0,1,B,R,qSeen1	qWant0,1,B,L,qBackErase	qBackErase,1,B,L,qBackErase
q0,B,Y,L,qa	qWant0,B,Y,L,qa	qBackErase,B,N,L,qa
qSeen0,0,0,R,qSeen0	qWant1,0,B,L,qBackErase	
qSeen0,1,1,R,qSeen0	qWant1,1,B,L,qBack	
qSeen0,B,B,L,qWant0	qWant1,B,Y,L,qa	
qSeen1,0,0,R,qSeen1	qBack,0,0,L,qBack	
qSeen1,1,1,R,qSeen1	qBack,1,1,L,qBack	
qSeen1,B,B,L,qWant1	qBack,B,B,R,q0	

Palindrome Recognition Execution Animation



0	0	1	0	1	0	0	B	B	B	B	B	B	B	B	B	B	B	B	
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19

Current Step: 0 Current Rule: qWant0, B, Y, L, qa Next Rule:

```
1  q0, 0, B, R, qSeen0
2  q0, 1, B, R, qSeen1
3  q0, B, Y, L, qa
4  qSeen0, 0, 0, R, qSeen0
5  qSeen0, 1, 1, R, qSeen0
6  qSeen0, B, B, L, qWant0
7  qSeen1, 0, 0, R, qSeen1
8  qSeen1, 1, 1, R, qSeen1
9  qSeen1, B, B, L, qWant1
10 qWant0, 0, B, L, qBack
11 qWant0, 1, B, L, qBackErase
12 qWant0, B, Y, L, qa
13 qWant1, 0, B, L, qBackErase
14 qWant1, 1, B, L, qBack
15 qWant1, B, Y, L, qa
16 qBack, 0, 0, L, qBack
17 qBack, 1, 1, L, qBack
18 qBack, B, B, R, q0
19 qBackErase, 0, B, L, qBackErase
20 qBackErase, 1, B, L, qBackErase
21 qBackErase, B, N, L, qa
```

Submit

Recover

0010100

Set

Continue

Debug

Finish

<<

Speed: 1.00

>>

Get State-Transition Diagram

Test

Week1 Outline

- Introduction of Turing Machine
- Experiment Platform
- Examples
- Questions

This lab includes three questions

- Q1: Unary addition of two unary numbers
- Q2: Binary addition of two 4-bit numbers
- Q3: Binary addition of two arbitrary numbers

Examples of Unary and Binary Addition

- Unary Addition
 - $11 + 11 = ?$
 - $111 + 1111 = ?$
- Binary Addition
 - $1010 + 1 = ?$
 - $111 + 1111 = ?$

Examples of Unary and Binary Addition

- Unary Addition

- $11 + 11 = ?$ 1111

- $111 + 1111 = ?$ 11111

- Binary Addition

- $1010 + 1 = ?$ 1011

- $111 + 1111 = ?$ 10110

Question 1

- Design a Turing machine that can perform **arbitrary bit unary addition**
- Format requirement:
 - Input: (some 1)+(some 1)=
 - Output on tape: ...###(some 1)B###...
 - The input "a number of 1s" represents the unary number to be added, which may be zero ones; the "a number of 1s" on the paper tape when the machine stops.
 - As the answer, the starting position should be **one square to the right of the = position in the input, and end with B**; # is any letter in Γ .
 - Samples:
 - Input: 11+1=
 - on tape: ...11+1=**111B**...
 - Input: 11+1=
 - Output on tape: ...FBc0**111B**cWW...
 - Note: the position of blue letter 1 in output is same as the position of letter = in input

Question 2

- Design a Turing machine that can perform binary addition of **4-bit numbers**
- Format requirement:
 - Input: (4-bit 0 or 1)+(4-bit 0 or 1)=
 - Output on tape: ...###(5-bit 0 or 1)B###...
 - The input “4-bit 0 or 1” represents the binary number to be added; the “5-bit 0 or 1” on the tape when the machine stops (pay attention to the **leading 0** when the result is only 4 digits).
 - As the answer, the starting position should be **one square to the right of the = position in the input, and end with B**; # is any letter in Γ .
 - Samples:
 - Input: 1101+0011=
 - Output on tape: ...1101+0011=**10000B**...
 - Input: 0001+0011=
 - Output on tape: ...001101101**000100B**000...
 - Note: the position of blue letter 0 in output is same as the position of letter = in input

Question 3

- Design a Turing machine that can perform **arbitrary bit binary addition**
- Format requirement:
 - Input: (some 0 or 1)+(some 0 or 1)=
 - Output on tape: ...####(some 0 or 1)B####...
 - The input “some 0 or 1” **must not contain leading 0s; the length of each operand is greater than or equal to 1**. The “some 0 or 1” on the tape when the machine stops.
 - As the answer **should not contain leading 0s**. The starting position should be **one square to the right of the = position in the input, and end with B**; # is any letter in Γ .
 - Samples:
 - Input: 11+1001=
 - Output on tape: ...11+1001=**1100B**...
 - Input: 11+1001=
 - Output on tape: ...wRQTqLaA**1100B**ssR...
 - Note: the position of letter A in output is same as the position of letter = in input

Grading Policy

- Questions (50%)
 - 50% for the unary adder
 - 20% for the 4-bit binary adder
 - 30% for the arbitrary-bit binary adder
- Team Presentation (20%)
- Project Report (30%)

Criteria of team presentation

- The design of the state transition table
 - summarize the design methods of each group member
- Experiment procedure
 - When, Where, What, How
- Experimental difficulties
- Proof of the correctness and completeness of the state transition table
 - Mathematical or experimental proof
- Understanding of Turing Machine
 - Why Turing machine can handle unlimited input
 - Why Turing machine is so powerful

Suggested structure of Lab report

- The Turing adder lab report needs to include the following as well as explanatory text
 - ① The design of the state transition table
 - ② Experiment procedure
 - ③ Experimental difficulties
 - ④ Proof of the correctness and completeness of the state transition table
 - ⑤ Thinking and perception of the Lab

Second week

Team discussion

- Peer review, revise design and discuss understanding of Turing Machine.
 - At the end of this lab session, each team member should be able to retell the work of other members.
- Suggested steps
 - Each team member shares his/her design with the team
 - Record bugs, difficulties, key ideas and differences
 - Talking about understanding of Turing Machine
 - Answer questions teacher asked in the last session
 - Prepare for team presentation to the class
- Team leader needs to play a leadership role