

Turing Compute Model for Non-negative Binary Numbers

Chen Wenyu

School of Computer Science and Engineering, University of Electronic Science & Technology of China, Chengdu, Sichuan, China
Email: cwy@uestc.edu.cn

Wang Xiaobin

School of Computer Science and Engineering, University of Electronic Science & Technology of China, Chengdu, Sichuan, China
Email: xbwang@uestc.edu.cn

Cheng Xiaoou

School of Computer Science and Engineering, University of Electronic Science & Technology of China, Chengdu, Sichuan, China
Email: carlyhawk@gmail.com

Sun Shixin

School of Computer Science and Engineering, University of Electronic Science & Technology of China, Chengdu, Sichuan, China
Email: sxsun@uestc.edu.cn

Abstract—Turing-computable issue is important in research of Turing Machine and has significant value in both theory and practice. The paper analyzes Turing-computable issue of non-negative numbers by relational operations(includes greater than, less than and equal) and arithmetic operations(includes add operation, subtract, multiply, divide and modulo). The regulation of computing 1-bit binary number and the carry set(or borrow set) are defined to compute multi-bit binary numbers. Algorithms are described through multi-tape Turing machine. For some operations, the discussion is extended to base-N number system($3 \leq N \leq 10$). Turing regulation for add operation is used to implement the Turing counter for counting the length of the string on the input tape of the Turing machine.

Index Terms—Turing machine, Turing computable, binary, Turing counter

I. INTRODUCTION

Turing machine is a seven-tuple $(Q, \Sigma, \Gamma, \delta, q_0, q_{\text{accept}}, q_{\text{reject}})$, of which Q, Σ, Γ , are Finite set and:

Q is the set of state;

Σ is the alphabet of input, which doesn't contain a special symbol of blank $\square$.

Γ is the alphabet with the word, which $B \in \Gamma$ and $\Sigma \in \Gamma$.

$\delta: Q \times \Gamma \rightarrow Q \times \Gamma \times \{L, R, N\}$ is transfer function, which signs the L, R or N, read-write head is shifted to right, left or No-shift.

$q_0 \in Q$ is the state of initial;

$q_{\text{accept}} \in Q$ is the state of accept.

$q_{\text{reject}} \in Q$ is the state of reject, and $q_{\text{accept}} \neq q_{\text{reject}}$.

Turing machine will operate as follows:

The beginning of the input string of symbols $\omega = \omega_0 \omega_1 \dots \omega_{n-1} \in \Sigma^*$ to fill in the $0, 1 \dots n-1$ lattice from left to right, the other fills blank (that is, to fill B). M read-write head point at the No.0 grid, M is at state q_0 . When machine is running, the transfer function in accordance with the rules described in δ calculated. For example, if the current state of the machine for the q , the meaning of read-write head of the symbols for the lattice in x , set $\delta(q, x) = (q', x', L)$, the machine entered a new state q' , will read Write the first meaning of the symbols in the lattice replaced by x' , then the first left to read and write a lattice. If at some point, the meaning of read-write head is the first No.0 lattice, but the transfer function in accordance with its next step will be to continue to the left, when it is fixed in place to stop. In other words, reading and writing the first tape has not moved out of the left border. If at some point M in accordance with the transfer function into the state q_{accept} , it stands immediately and accept the input string; if at some point M in accordance with the transfer function into the state q_{reject} , it stands immediately and refused to enter the string. Note that the transfer function δ is a partial function, in other words for some q, x , $\delta(q, x)$ may not be defined, if encountered in the operation of an operation is not defined under the circumstances, the machine will be shutdown immediately[1]. Turing machine and other automata can identify languages, and they are capable to "compute".

Given designated number representation, the evaluation of functions of non-negative numbers can be achieved.

Turing-computable issue is important in research of Turing Machine and has significant value in both theory and practice[2,3]. Prof. Jiang Zongli, Prof. Chen Youqi, and Prof. Zhang Li'ang analyzed Turing computable issue of addition, proper subtraction and multiplication of non-negative numbers[2,3]. Unary method is used to represent non-negative numbers, that is, 0m represents integer m. Peter Linz ,Hopcroft, John, Michael Sipser implemented counter using two-stack machine[4,5,6,7].

The paper analyzes Turing-computable issue of non-negative numbers by relational operations(includes greater than, less than and equal) and arithmetic operations(includes add operation, subtract, multiply, divide and modulo) . Based on binary system, the discussion is extended to operations of $N(3 \leq N \leq 10)$ system. Input the operators on the first tape and second tape respectively. By storage and move functions, the Turing Machine can input symbol B on the second unit of the tapes and the operators on both tapes should be right justified, The write/read head moves to the lowest bit of the operator.

II. TURING RELATIONAL OPERATIONS OF NON-NEGATIVE NUMBERS

For binary numbers n and m, the accepting states determine the relationship between the numbers, n is greater than m or is less than m or n equals m.

2-tape Turing machine is created to compute the relational operation.

Main idea: First check the high order bits from right to left and then check the low order bits from left to right.

```

<start,[0,0],q,[0,0],L>
<start,[0,1],q,[0,1],L>
<start,[1,0],q,[1,0],L>
<start,[1,1],q,[1,1],L>
<q,[0,0],q,[0,0],L>
<q,[0,1],q,[0,1],L>
<q,[1,0],q,[1,0],L>
<q,[1,1],q,[1,1],L>
<q,[0,B],GT,[0,B],N>
<q,[1,B],GT,[1,B],N>
<q,[B,0],LT,[B,0],N>
<q,[B,1],LT,[B,1],N>
<q,[B,B],E,[B,B],R>
<E,[0,0],E,[0,0],R>
<E,[1,1],E,[1,1],R>
<E,[0,1],LT,[0,1],N>
<E,[1,0],GT,[1,0],N>
<E,[B,B],EQ,[B,B],N>

```

Accepting states GT, LT and EQ represent $n > m$, $n < m$ and $n = m$ respectively.

Example $1011011 > 1001111$?

The transforms for instantaneous description are

```

[B,B][1,1][0,0][1,0][1,1][0,1]start[1,1]
=>[B,B][1,1][0,0][1,0][1,1]q[0,1][1,1]
=>[B,B][1,1][0,0][1,0]q[1,1][0,1][1,1]
=>[B,B][1,1][0,0]q[1,0][1,1][0,1][1,1]
=>[B,B][1,1]q[0,0][1,0][1,1][0,1][1,1]

```

```

=>[B,B]q[1,1][0,0][1,0][1,1][0,1][1,1]
=>q[B,B][1,1][0,0][1,0][1,1][0,1][1,1]
=>[B,B]E[1,1][0,0][1,0][1,1][0,1][1,1]
=>[B,B][1,1]E[0,0][1,0][1,1][0,1][1,1]
=>[B,B][1,1][0,0]E[1,0][1,1][0,1][1,1]
=>[B,B][1,1][0,0]GT[1,0][1,1][0,1][1,1]

```

For base-N numbers, let $a, b = 0, 1, 2, \dots, N-1$

```

<start,[a,b],q,[a,b],L>
<q,[a,b],q,[a,b],L>
<q,[a,B],GT,[a,B],N>
<q,[B,b],LT,[B,b],N>
<q,[B,B],E,[B,B],R>
<E,[a,a],E,[a,a],R> //a=a
<E,[a,b],LT,[a,b],R> //a<b
<E,[b,a],GT,[b,a],R> //a>b
<E,[B,B],EQ,[B,B],N>

```

Accepting states GT, LT and EQ represent $n > m$, $n < m$ and $n = m$ respectively.

III. TURING ADD OPERATION OF NON-NEGATIVE BINARY NUMBERS

Create 3-tape Turing machine to compute add operation for non-negative numbers. The third tape stores computation results(input symbol B on the second unit of the first and second tape).

```

<[q,0],[0,0,B],[q,0],[0,0,0],L>
<[q,0],[0,1,B],[q,0],[0,1,1],L>
<[q,0],[1,0,B],[q,0],[1,0,1],L>
<[q,0],[1,1,B],[q,1],[1,1,0],L> //Carry set
<[q,1],[0,0,B],[q,0],[0,0,1],L>
<[q,1],[0,1,B],[q,1],[0,1,0],L>
<[q,1],[1,0,B],[q,1],[1,0,0],L>
<[q,1],[1,1,B],[q,1],[1,1,0],L>

```

//The lengths of the numbers differ

```

<[q,0],[0,B,B],[q,0],[0,B,0],L>
<[q,0],[1,B,B],[q,0],[1,B,1],L>
<[q,0],[B,0,B],[q,0],[B,0,0],L>
<[q,0],[B,1,B],[q,0],[B,1,1],L>
<[q,1],[0,B,B],[q,0],[0,B,1],L>
<[q,1],[1,B,B],[q,1],[1,B,0],L>
<[q,1],[B,0,B],[q,0],[B,0,1],L>
<[q,1],[B,1,B],[q,1],[B,1,0],L>

```

//Ending condition

```

<[q,0],[B,B,B],E,[B,B,B],L>
<[q,1],[B,B,B],E,[B,B,1],L>
<E,[⊢,⊢,⊢],END,[⊢,⊢,⊢],N>

```

Example $10110101 + 11100111 = ?$

//Initialize,

```

⊢B10110101
⊢B11100111
⊢BBBBBBBB

```

//1+1=10

```

⊢B10110101
⊢B11100111
⊢BBBBBBB0

```

//0+1+1=10

```

⊢B10110101
⊢B11100111

```

```

    ⊢BBBBBB00
//1+1+1=11
    ⊢B10110101
    ⊢B11100111
    ⊢BBBBB100
//0+0+1=1
    ⊢B10110101
    ⊢B11100111
    ⊢BBBB1100
//1+0+0=1
    ⊢B10110101
    ⊢B11100111
    ⊢BBB11100

//1+1+0=10
    ⊢B10110101
    ⊢B11100111
    ⊢BB011100
//0+1+1=10
    ⊢B10110101
    ⊢B11100111
    ⊢B0011100
//1+1+1=11
    ⊢B10110101
    ⊢B11100111
    ⊢B10011100
//0+0+1=1
    ⊢B10110101
    ⊢B11100111
    ⊢110011100

```

For base-N numbers, let $a, b = 0, 1, 2, \dots, N-1$.

```

//if  $a+b < N$ ; no carry set
    <[q,0],[a,b,B],[q,0],[a,b,a+b],L>
//if  $a+b \geq N$ ; carry set
    <[q,0],[a,b,B],[q,1],[a,b,a+b-N],L>
//if  $a+b < N-1$ ; no carry set
    <[q,1],[a,b,B],[q,0],[a,b,a+b+1],L>
//if  $a+b \geq N-1$ ; carry set continues
    <[q,1],[a,b,B],[q,1],[a,b,a+b-N+1],L>
//the length of the numbers differ
    <[q,0],[a,B,B],[q,0],[a,B,a],L>
    <[q,0],[B,b,B],[q,0],[B,b,b],L>
//if  $a < N-1$ 
    <[q,1],[a,B,B],[q,0],[a,B,a+1],L>
//if  $a = N-1$ 
    <[q,1],[a,B,B],[q,1],[a,B,0],L>
//if  $b < N-1$ 
    <[q,1],[B,b,B],[q,0],[B,b,b],L>
//if  $b = N-1$ 
    <[q,1],[B,b,B],[q,1],[B,b,0],L>
//Ending condition
    <[q,0],[B,B,B],END,[B,B,B],N>
    <[q,1],[B,B,B],END,[B,B,1],N>
For base-10

```

```

<[q,0],[0,0,B],[q,0],[0,0,0],L>
<[q,0],[0,1,B],[q,0],[0,1,1],L>
<[q,0],[0,2,B],[q,0],[0,2,2],L>
<[q,0],[0,3,B],[q,0],[0,3,3],L>
<[q,0],[0,4,B],[q,0],[0,4,4],L>
<[q,0],[0,5,B],[q,0],[0,5,5],L>
<[q,0],[0,6,B],[q,0],[0,6,6],L>
<[q,0],[0,7,B],[q,0],[0,7,7],L>
<[q,0],[0,8,B],[q,0],[0,8,8],L>
<[q,0],[0,9,B],[q,0],[0,9,9],L>

<[q,0],[1,0,B],[q,0],[1,0,1],L>
<[q,0],[1,1,B],[q,0],[1,1,1],L>
<[q,0],[1,2,B],[q,0],[1,2,2],L>
<[q,0],[1,3,B],[q,0],[1,3,3],L>
<[q,0],[1,4,B],[q,0],[1,4,5],L>
<[q,0],[1,5,B],[q,0],[1,5,6],L>
<[q,0],[1,6,B],[q,0],[1,6,7],L>
<[q,0],[1,7,B],[q,0],[1,7,8],L>
<[q,0],[1,8,B],[q,0],[1,8,9],L>
<[q,0],[1,9,B],[q,1],[1,9,0],L>
...
<[q,1],[0,0,B],[q,0],[0,0,1],L>
<[q,1],[0,1,B],[q,0],[0,1,2],L>
<[q,1],[0,2,B],[q,0],[0,2,3],L>
<[q,1],[0,3,B],[q,0],[0,3,4],L>
<[q,1],[0,4,B],[q,0],[0,4,5],L>
<[q,1],[0,5,B],[q,0],[0,5,6],L>
<[q,1],[0,6,B],[q,0],[0,6,7],L>
<[q,1],[0,7,B],[q,0],[0,7,8],L>
<[q,1],[0,8,B],[q,0],[0,8,9],L>
<[q,1],[0,9,B],[q,1],[0,9,0],L>
...
<[q,1],[1,0,B],[q,0],[1,0,2],L>
<[q,1],[1,1,B],[q,0],[1,1,3],L>
<[q,1],[1,2,B],[q,0],[1,2,4],L>
<[q,1],[1,3,B],[q,0],[1,3,5],L>
<[q,1],[1,4,B],[q,0],[1,4,6],L>
<[q,1],[1,5,B],[q,0],[1,5,7],L>
<[q,1],[1,6,B],[q,0],[1,6,8],L>
<[q,1],[1,7,B],[q,0],[1,7,9],L>
<[q,1],[1,8,B],[q,1],[1,8,0],L>
<[q,1],[1,9,B],[q,1],[1,9,1],L>
...
//The lengths of the numbers differ
    <[q,0],[0,B,B],[q,0],[0,B,0],L>
    <[q,0],[1,B,B],[q,0],[1,B,1],L>
    <[q,0],[2,B,B],[q,0],[2,B,2],L>
    <[q,0],[3,B,B],[q,0],[3,B,3],L>
    <[q,0],[4,B,B],[q,0],[4,B,4],L>
    <[q,0],[5,B,B],[q,0],[5,B,5],L>
    <[q,0],[6,B,B],[q,0],[6,B,6],L>
    <[q,0],[7,B,B],[q,0],[7,B,7],L>
    <[q,0],[8,B,B],[q,0],[8,B,8],L>
    <[q,0],[9,B,B],[q,0],[9,B,9],L>

    <[q,0],[B,0,B],[q,0],[B,0,0],L>
    <[q,0],[B,1,B],[q,0],[B,1,1],L>
    <[q,0],[B,2,B],[q,0],[B,2,2],L>
    <[q,0],[B,3,B],[q,0],[B,3,3],L>

```

```

<[q,0],[B,4,B],[q,0],[B,4,4],L>
<[q,0],[B,5,B],[q,0],[B,5,5],L>
<[q,0],[B,6,B],[q,0],[B,6,6],L>
<[q,0],[B,7,B],[q,0],[B,7,7],L>
<[q,0],[B,8,B],[q,0],[B,8,8],L>
<[q,0],[B,9,B],[q,0],[B,9,9],L>

<[q,1],[B,0,B],[q,0],[B,0,1],L>
<[q,1],[B,1,B],[q,0],[B,1,2],L>
<[q,1],[B,2,B],[q,0],[B,2,3],L>
<[q,1],[B,3,B],[q,0],[B,3,4],L>
<[q,1],[B,4,B],[q,0],[B,4,5],L>
<[q,1],[B,5,B],[q,0],[B,5,6],L>
<[q,1],[B,6,B],[q,0],[B,6,7],L>
<[q,1],[B,7,B],[q,0],[B,7,8],L>
<[q,1],[B,8,B],[q,0],[B,8,9],L>
<[q,1],[B,9,B],[q,1],[B,9,0],L>

//Ending condition
<[q,0],[B,B,B],END,[B,B,B],N>
<[q,1],[B,B,B],END,[B,B,1],N>

Example 62345 + 87971=?
//Initialize,
├B62345
├B87971
├BBBBBBB

//5+1=6
├B62345
├B87971
├BBBBBB6

//4+7+0=11
├B62345
├B87971
├BBBBB16

//3+9+1=13
├B62345
├B87971
├BBBB316

//2+7+1=10
├B62345
├B87971
├BB0316

//6+8+1=15
├B62345
├B87971
├B50316

//0+0+1=1
├B62345
├B87971
├150316

```

IV. TURING SUBTRACTION OF NON-NEGATIVE BINARY NUMBERS

Create 3-tape Turing machine to compute subtraction operation for non-negative numbers. The third tape stores computation results.

Compute the relational operation first, if the first operator is less than the second operator, then use the third tape to switch the operator to let the operator on the first tape be the greater one.

```

<[q,0],[0,0,B],[q,0],[0,0,0],L>
<[q,0],[0,1,B],[q,1],[0,1,1],L>
<[q,0],[1,0,B],[q,0],[1,0,1],L>
<[q,0],[1,1,B],[q,0],[1,1,0],L>
<[q,1],[0,0,B],[q,1],[0,0,1],L>
<[q,1],[0,1,B],[q,1],[0,1,0],L>
<[q,1],[1,0,B],[q,0],[1,0,0],L>
<[q,1],[1,1,B],[q,1],[1,1,1],L>

// the length of the numbers differ
<[q,0],[0,B,B],[q,0],[0,B,0],L>
<[q,0],[1,B,B],[q,0],[1,B,1],L>
<[q,1],[0,B,B],[q,1],[0,B,1],L>
<[q,1],[1,B,B],[q,1],[1,B,0],L>

//Ending condition
<[q,0],[B,B,B],END,[B,B,B],N>

For base-N numbers, let a=b=0,1,2, ...,N-1.
<[q,0],[a,b,B],[q,0],[a,b,a-b],L> //a>=b
<[q,0],[a,b,B],[q,1],[a,b,a+N-b],L> //a<b
<[q,1],[a,b,B],[q,0],[a,b,a-b-1],L> //a>b
<[q,1],[a,b,B],[q,1],[a,b,a+N-1-b],L> //a<=b

// the length of the numbers differ
<[q,0],[a,B,B],[q,0],[a,B,a],L>
<[q,1],[0,B,B],[q,1],[0,B,N-1],L>

//Continue borrowing
<[q,1],[a,B,B],[q,0],[a,B,a-1],L> // a≠0

//Ending condition
<[q,0],[B,B,B],END,[B,B,B],N>

Example 1101010 - 1011111=?
//Initialize,
├B1101010
├B1011111
├BBBBBBBBB

//10-1=1
├B1101010
├B1011111
├BBBBBBB1

//10-1=1
├B1101010
├B1011111
├BBBBBB11

//1-1=0
├B1101010
├B1011111
├BBBBB011

//10-1=1
├B1101010
├B1011111
├BBBB1011

//1-1=1
├B1101010
├B1011111
├BBB01011

//0-0=0
├B1101010

```

```

┌B1011111
┌BB001011
//1-1=0
┌B1101010
┌B1011111
┌B0001011
For base-10
<[q,0],[0,0,B], [q,0],[0,0,0],L>
<[q,0],[0,1,B], [q,1],[0,1,9],L>
<[q,0],[0,2,B], [q,1],[0,2,8],L>
<[q,0],[0,3,B], [q,1],[0,3,7],L>
<[q,0],[0,4,B], [q,1],[0,4,6],L>
<[q,0],[0,5,B], [q,1],[0,5,5],L>
<[q,0],[0,6,B], [q,1],[0,6,4],L>
<[q,0],[0,7,B], [q,1],[0,7,3],L>
<[q,0],[0,8,B], [q,1],[0,8,2],L>
<[q,0],[0,9,B], [q,1],[0,9,1],L>

<[q,0],[1,0,B], [q,0],[1,0,1],L>
<[q,0],[1,1,B], [q,0],[1,1,0],L>
<[q,0],[1,2,B], [q,1],[1,2,9],L>
<[q,0],[1,3,B], [q,1],[1,3,8],L>
<[q,0],[1,4,B], [q,1],[1,4,7],L>
<[q,0],[1,5,B], [q,1],[1,5,6],L>
<[q,0],[1,6,B], [q,1],[1,6,5],L>
<[q,0],[1,7,B], [q,1],[1,7,4],L>
<[q,0],[1,8,B], [q,1],[1,8,3],L>
<[q,0],[1,9,B], [q,1],[1,9,2],L>
...
<[q,1],[0,0,B], [q,1],[0,0,9],L>
<[q,1],[0,1,B], [q,1],[0,1,8],L>
<[q,1],[0,2,B], [q,1],[0,2,7],L>
<[q,1],[0,3,B], [q,1],[0,3,6],L>
<[q,1],[0,4,B], [q,1],[0,4,5],L>
<[q,1],[0,5,B], [q,1],[0,5,4],L>
<[q,1],[0,6,B], [q,1],[0,6,3],L>
<[q,1],[0,7,B], [q,1],[0,7,2],L>
<[q,1],[0,8,B], [q,1],[0,8,1],L>
<[q,1],[0,9,B], [q,1],[0,9,0],L>

<[q,1],[1,0,B], [q,0],[1,0,0],L>
<[q,1],[1,1,B], [q,1],[1,1,9],L>
<[q,1],[1,2,B], [q,1],[1,2,8],L>
<[q,1],[1,3,B], [q,1],[1,3,7],L>
<[q,1],[1,4,B], [q,1],[1,4,6],L>
<[q,1],[1,5,B], [q,1],[1,5,5],L>
<[q,1],[1,6,B], [q,1],[1,6,4],L>
<[q,1],[1,7,B], [q,1],[1,7,3],L>
<[q,1],[1,8,B], [q,1],[1,8,2],L>
<[q,1],[1,9,B], [q,1],[1,9,1],L>
...
//The lengths of the numbers differ
<[q,0],[0,B,B],[q,0],[0,B,0],L>
<[q,0],[1,B,B],[q,0],[1,B,1],L>
<[q,0],[2,B,B],[q,0],[2,B,2],L>
<[q,0],[3,B,B],[q,0],[3,B,3],L>
<[q,0],[4,B,B],[q,0],[4,B,4],L>
<[q,0],[5,B,B],[q,0],[5,B,5],L>
<[q,0],[6,B,B],[q,0],[6,B,6],L>

<[q,0],[7,B,B],[q,0],[7,B,7],L>
<[q,0],[8,B,B],[q,0],[8,B,8],L>
<[q,0],[9,B,B],[q,0],[9,B,9],L>

<[q,1],[0,B,B],[q,1],[0,B,9],L>
<[q,1],[1,B,B],[q,0],[1,B,0],L>
<[q,1],[2,B,B],[q,0],[2,B,1],L>
<[q,1],[3,B,B],[q,0],[3,B,2],L>
<[q,1],[4,B,B],[q,0],[4,B,3],L>
<[q,1],[5,B,B],[q,0],[5,B,4],L>
<[q,1],[6,B,B],[q,0],[6,B,5],L>
<[q,1],[7,B,B],[q,0],[7,B,6],L>
<[q,1],[8,B,B],[q,0],[8,B,7],L>
<[q,1],[9,B,B],[q,1],[9,B,8],L>
//Ending condition
<[q,0],[B,B,B],END,[B,B,B],N>
Example 6754345 - 5574317=?
//Initialize,
┌B8754345
┌B5574317
┌BBBBBBB
//15-7=8
┌B8754345
┌B5574317
┌BBBBBBB8
//3-1=2
┌B8754345
┌B5574317
┌BBBBBBB28
//3-3=0
┌B8754345
┌B5574317
┌BBBBB028
//4-4=0
┌B8754345
┌B5574317
┌BBBB0028
//15-7=8
┌B8754345
┌B5574317
┌BBB80028
//6-5=1
┌B8754345
┌B5574317
┌BB180028
//8-5=1
┌B8754345
┌B5574317
┌B3180028

```

V. TURING MULTIPLICATION OF NON-NEGATIVE BINARY NUMBERS

If one of the multipliers is 0, then the result is 0. If none of the multipliers is 0, then multi-tape Turing machine is used to compute the operation. If the original operators don't need to be saved, then 3-tape Turing

machine can be used to compute the multiplication of binary numbers. Symbol 2 is used as a separate label in the first tape, on the left side of the label, the number of symbol 0 is the sum of the length of the operators plus 1, on the right side of the label is the multiplicator(the second operator); the lowest bit on the second tape is the bit which is on the left to the label 2 on the first tape, the multiplicand is stored(the second operator), and the third tape stores symbol B.

Algorithm: move the read/write head to the last bit of the first tape;

1) If the last bit of the first tape is 0, then change the bit to symbol B, go to step (3); 2) If the last bit of the first tape is 1; add the numbers of the first tape and the second tape and store the result on the third tape, then copy the result to the first tape and change the last bit on the first tape to symbol B; 3) If the last bit of the first tape is 0, move the number on the second tape 1 bit leftwards and change the last bit of the first tape to symbol B, go to step (3); 4) If the last bit of the first tape is 1; move the number on the second tape 1 bit leftwards, add the number on the first tape and the second tape and store the result on the third tape, then copy the number on the third tape to the first tape and change the last bit on the first bit to symbol B, go to step (3); 5) If the last bit of the first tape is 2, the multiplication ends, the result is the number of the third tape and also the left side of label 2.

Example: 1010×101

Status of the tapes when the process starts:

```
┌000000002101
├BBBB1010BBBB
└BBBBBBBBBBBB
```

The last bit of the first tape is 1; add the numbers of the first tape and the second tape and store the result on the third tape, then copy the result to the first tape and change the last bit on the first tape to symbol B.

```
┌00001010210B
├BBBB1010BBBB
└BBBB1010BBBB
```

The last bit of the first tape is 0, move the number on the second tape 1 bit leftwards and change the last bit of the first tape to symbol B.

```
┌0000101021BB
├BBB10100BBBB
└BBBB1010BBBB
```

The last bit of the first tape is 0, move the number on the second tape 1 bit leftwards.

```
┌0000101021BB
├BB101000BBBB
└BBBB1010BBBB
```

Then, add the numbers of the first tape and the second tape and store the result on the third tape, then copy the result to the first tape and change the last bit on the first tape to symbol B

```
┌001100102BBB
├BB101000BBBB
└BB110010BBBB
```

The number on the third tape is the result.

VI. TURING DIVISION OF NON-NEGATIVE BINARY NUMBERS

Suppose the dividend is M, the divisor is N, and the quotient is y, the division is: $y = M/N$. 3-tape Turing machine is used to compute the division. The first tape stores the dividend, the second tape stores the divisor and the third tape stores the quotient. The result of the first tape is the remainder and the result of the third tape is the quotient.

If $N=0$, 0 is the divisor, then the result is nonsensical;

If $M=0$ and $N \neq 0$, then the quotient is 0;

If $M < N$, then divisor M is the remainder and the quotient is 0;

If $M \geq N$, the initial value $y=0$, then go to loop $y=y+1, M=M-N$; if $M < N$, the operation ends.

We handle the above four different situations one by one. If the length of the operator are the same, F we need to decide which operator is the greater one and the subtraction is required. If the length of M is greater than that of N, move N to let the numbers right-justified and the initial value of the quotient is 0, compute the subtraction operation; return is based on the status after the subtraction, if there is no overflow, then add 1 to the quotient; otherwise, the division ends.

When $M \neq 0, N = 0$,

```
<start,[┌,┌,┌],q0,[┌,┌,┌],R>
<q0,[a,B,B],qinf,[B,B,B],R>
<qinf,[a,B,B],qinf,[B,B,B],R>
<qinf,[B,B,B],accept,[B,B,B],N>
```

When $M=0, N \neq 0$, the quotient is 0, the remainder on the first tape is blank and change the second tape to blank, the quotient on the third tape is 0.

```
<q0,[B,1,B],qzero,[B,B,0],R>
<qzero,[B,a,B],qzero,[B,B,B],R>
<qzero,[B,B,B],accept,[B,B,B],N>
```

When $M < N$, the first tape outputs M as the remainder, change the second tape to blank and the quotient on the third tape is 0.

```
<q0,[a,b,B],q0,[a,b,B],R>
<q0,[B,b,B],qless1,[B,b,B],R>
<qless1,[B,b,B],qless1,[B,B,B],R>
<qless1,[B,B,B],qless2,[B,B,B],L>
<qless2,[B,B,B],qless2,[B,B,B],L>
<qless2,[a,b,B],qless2,[a,B,B],L>
<qless2,[S,S,S],qless3,[S,S,S],R>
```

The number on the first tape is the remainder and the number on the third tape is the result 0, change all symbols on the second tape to blank and the division ends.

```
<qless3,[a,B,c],accept,[a,B,c],N>
<qless3,[a,B,B],accept,[a,B,B],N>
```

When $M \geq N$, the result of the first tape is the remainder, and the result of the third tape is the quotient.

```
<q0,[a,b,B],q0,[a,b,B],R>
<q0,[a,B,B],qmore,[a,B,B],L>
<qmore,[a,b,B],qmove,[a,B,B],R>
<qmove,[a,B,B],qmove,[a,B,B],R>
<qmove,[a,B,B],qmodify,[B,B,B],L>
<qmodify,[a,B,B],qback,[a,b,B],L>
<qback,[a,B,B],qback,[a,B,B],L>
```

```

    <qback,[a,b,B],qmove,[a,B,B],R>
//Right-justified
    <qback,[S,S,S],qminus1,[S,S,S],R>
    <qminus1,[a,B,B],qminus1,[a,B,B],R>
    <qminus1,[a,b,B],qminus1,[a,b,B],R>
    <qminus1,[a,b,c],qminus1,[a,b,c],R>
//Read/write moves to the rightmost bit and ready for
//the subtraction
    <qminus1,[B,B,B],qminus,[B,B,B],L>
//Subtraction label when there is no borrow
    <qminus,[0,0,B],qminus,[0,0,B],L>
    <qminus,[1,0,B],qminus,[1,0,B],L>
    <qminus,[1,1,B],qminus,[0,1,B],L>
    <qminus,[0,1,B],overflow,[1,1,B],L>
    <qminus,[0,0,c],qminus,[0,0,c],L>
    <qminus,[1,0,c],qminus,[1,0,c],L>
    <qminus,[1,1,c],qminus,[0,1,c],L>
    <qminus,[0,1,c],overflow,[1,1,c],L>
//Subtraction label when there is borrow
    <overflow,[0,0,B],overflow,[1,0,B],L>
    <overflow,[0,1,B],overflow,[0,1,B],L>
    <overflow,[1,0,B],qminus,[0,0,B],L>
    <overflow,[1,1,B],overflow,[1,1,B],L>
    <overflow,[0,0,c],overflow,[1,0,c],L>
    <overflow,[0,1,c],overflow,[0,1,c],L>
    <overflow,[1,0,c],qminus,[0,0,c],L>
    <overflow,[1,1,c],overflow,[1,1,c],L>
//Ending condition of the subtraction, handle the ending
state of subtraction and
//Move the read/write head to the rightmost bit
    <qminus,[a,B,B],qmin_over,[a,B,B],R>
    <qminus,[a,B,c],qmin_over,[a,B,c],R>
    <overflow,[1,B,B],flow_over,[0,B,B],R>
    <overflow,[0,B,B],overflow,[1,B,B],L>
    <overflow,[1,B,c],overflow_over,[0,B,c],R>
    <overflow,[0,B,c],overflow,[1,B,c],L>
    <qmin_over,[a,b,c],qmin_over,[a,b,c],R>
// Move the read/write head to the rightmost bit and ready
to add 1 to the quotient
    <flow_over,[a,b,c],flow_over,[a,b,c],R>
    <qmin_over,[B,B,B],qm_return,[B,B,B],L>
    <flow_over,[B,B,B],flow_return,[B,B,B],L>
// Consider carry set issue of adding 1 to the quotient
operation
    <qm_return,[a,b,B],add_over,[a,b,1],R>
//Return directly when no carry set
    <qm_return,[a,b,0],add_over,[a,b,1],R>
    <qm_return,[a,b,1],carry,[a,b,0],L>
//Label for carry set
    <qm_return,[a,B,B], add_over,[a,b,1],R>
    <qm_return,[a,B,0],add_over,[a,b,1],R>
    <qm_return,[a,B,1],carry,[a,b,0],L>
    <flow_return,[a,b,B],add_over,[a,b,1],R>
// Return directly when no carry set
    <flow_return,[a,b,0],add_over,[a,b,1],R>
    <flow_return,[a,b,1],carry,[a,b,0],L>
// Label for carry set
    <carry,[a,b,B],add_over,[a,b,1],R>
//handle carry set status
    <carry,[a,b,0],add_over,[a,b,1],R>

```

```

    <carry,[a,b,1],carry,[a,b,0],L>
//Add 1 to the quotient and return to the rightmost bit
    <add_over,[a,b,c],add_over,[a,b,c],R>
//Return and ready for next loop
    <add_over,[B,B,B], q0,[B,B,B],L>
Use binary subtraction and adding 1 to the quotient to
compute division for binary numbers.

```

VII. TURING COUNTER FOT THE STRING LENGTHH

4-tape Turing machine is used to count the number of the symbols on the input tape.

The first tape stores the string to be counted; the second tape stores the last time counting result(decimal, the initial value is 0), the third tape stores 1(decimal) and the fourth tape stores the counting result(decimal).

From the right to left, each time scan an input symbol of the first tape, based on the rule introduced in chapter 2 and add the numbers on the second tape and the third tape and store the result on the fourth tape. Then, copy the result on the fourth tape to the second tape and continue until scanning completes.

Example, count the string length of “abcdefghijk”.

```

//Initialize,
    ⊢abcdefghijk
    ⊢BB... B0
    ⊢BB... B1 // Add 1 each time
    ⊢BB... BB
//Count for the first time
    ⊢abcdefghijB
    ⊢BB... B0
    ⊢BB... B1
    ⊢BB... B1
//Copy the number on the fourth tape to the second tape
    ⊢abcdefghijB
    ⊢BB... B1
    ⊢BB... B1
    ⊢BB... B1
//Count
    ⊢abcdefghiBB
    ⊢BB... B1
    ⊢BB... B1
    ⊢BB... B2
// Copy the number on the fourth tape to the second
tape
    ⊢abcdefghiBB
    ⊢BB... B2
    ⊢BB... B1
    ⊢BB... B2
...
//The last symbol
    ⊢aBBBBBBBBBB
    ⊢BB... B10
    ⊢BB... BB1
    ⊢BB... B10
//Count
    ⊢BBBBBBBBBBBB
    ⊢BB... B10

```

⊢BB... BB1

⊢BB... B11

// Copy the number on the fourth tape to the second tape

⊢BBBBBBBBBB

⊢BB... B11

⊢BB... BB1

⊢BB... B11

When there is only symbol on the first tape, except for the symbol B, the counting ends and the fourth tape is the result.

VIII. CONCLUSION

The paper defines regulation of computing 1-bit binary number and the carry set(or borrow) issues and analyzes Turing-computable issue of non-negative numbers by relational operations and arithmetic operations. For some operations, the discussion is extended to base-N number system($3 \leq N \leq 10$). Turing regulation for add operation is used to implement the Turing counter for counting the length of the string on the input tape of the Turing machine.

Future work will focus on the analysis of the computational complexity and the Turing computable model for base-N ($N \geq 10$) number system.

ACKNOWLEDGMENT

This paper is supported by the the Si Chuan science and technology(2006J13-068).

REFERENCES

- [1] Ming Li and Paul Vitanyi, An Introduction to Kolmogorov Complexity and Its Applications, 2nd Edition, Springer Verlag, 1997
- [2] Jiang Zongli, Theory of Formal languages and Automata [M]. 2nd ed. Beijing: Tsinghua University Press, 2008G
- [3] CHEN You-Qi, Formal Languages and Automata, Tianjing: Nankai University Press, China, 1999
- [4] Zhang Li'ang, Computability and introduction to complexity of computation [M] Beijing: Peking University Press 1996
- [5] Peter Linz .An introduction to formal languages and automata[M]. Boston :Jones and Bartlett,2001.
- [6] Michael Sipser. Introduction to the theory of computation[M]. New York: Thomson, 1996.
- [7] Hopcroft, John, Rajeev Motwani, and Jeffrey Ullman.Introduction to Automata Theory, Languages and Computation [M]. 2nd ed. Reading, MA: Addison Wesley, 2000

Chen Wenyu, male, born in 1968.

He is an associate professor of School of Computer Science and Engineering, University of Electronic Science & Technology of China. His research interest include: compiling technique, pattern recognition, formal language and automata.

He received the B.S. and M.S. degrees in computer science and engineering from University of Electronic Science and Technology of China(UESTC),Chengdu, China, in 1990 and 1993, respectively. He was a researcher at the School of Computer Science and Engineering, UESTC. From 2004, he was a associate professor at the School of Computer Science and Engineering, UESTC. His research interests include computer Language and compile, formal language and automation, and neural networks.

He is a mentor for postgraduate student;a grade college teacher (for undergraduate);a quality outstanding young teacher (for postgraduate).

He completed a number of research tasks, both through the Ministry, the provincial identification; publish over 20 papers and 6 teaching materials.

Associate prof. Chen is a senior member of China Computer Federation(E200011786S).

Wang Xiaobin, male, born in 1964.

He is an associate professor of School of Computer Science and Engineering, University of Electronic Science & Technology of China. His research interests include data structure, computer Language and compile, software engineering, neural networks, and optimization.

He received the B.S. and M.S. degrees in computer science and engineering from University of Electronic Science and Technology of China, Chengdu, China, in 1985 and 1988, respectively. He is currently working toward the Ph.D. degree in Computational Intelligence Laboratory, School of Computer Science and Engineering, University of Electronic Science and Technology of China. From 1988 to 1998, he was a researcher at the School of Computer Science and Engineering, University of Electronic Science and Technology of China. From 1998, he was a Professor at the School of Computer Science and Engineering, University of Electronic Science and Technology of China. He is currently the dean of ChengDu College, University of Electronic Science and Technology of China.

Cheng Xiaou, female, born in 1984.

She is candidate for Master's degree in Software Engineering, School of Software, University of Electronic Science & Technology of China. Her research interest include: compiling technique, theory of software.

Sun Shixin, male, born in 1940.

He is a Ph.D supervisor and professor of School of Computer Science and Engineering, University of Electronic Science & Technology of China. His research interest include: theory of Computer Science, Parallel Computer Systems, Numerical Algorithm & Non-Numerical Algorithm (including Parallel Algorithm).

Prof Sun 1966. B.A. in Mathematics,Sichuan University, Chengdu, China;1984-1987. visiting scholar at the Grenoble University, France,;1990. doing research at Roma University, Italy and Grenoble;University, France.