
CHAPTER 17

Theory of Computation

Review Questions

1. We need the theory of computation in order for computer scientists to answer some intrinsic questions.
2. The increment statement adds 1 to a variable. The decrement statement subtracts 1 from a variable. The loop statement repeats an action while the value of a variable is not zero.
3. The macro $Y \Leftarrow X$ in the Simple Language that assigns the value of one variable to another is as follows:

```
Y  $\Leftarrow$  0
TEMP  $\Leftarrow$  0
while X
{
  incr Y
  decr X
  incr TEMP
}
while TEMP
{
  decr TEMP
  incr X
}
```

We can see that this macro uses the three basic statements.

4. Any program that can be written in our Simple Language can also be implemented on the Turing machine.

5. The tape holds a sequence of characters from an acceptable character set. The read/write head points to a character on the tape and is used to read and write characters. The controller controls the read/write head and is the theoretical counterpart of the CPU in today's computers.
6. One way is to assign specific characters to denote the beginning and the end of the data on the tape.
7. It can move forward, move backward, or stay in the same place.
8. A transition diagram shows the state of the controller at any given time.
9. The transition diagram is a pictorial representation of the possible states of the controller. The transition table shows the current state, the character read, the character written, where the read/write head should move, and the new state. The transition table has much more information.
10. A Godel number is an unsigned number that is assigned to every program that can be written in a specific language.
11. We use the Godel number to specify the program to be tested by the theoretical program used to determine whether the specified program will terminate.
12. We use big-O notation to specify the complexity of a program as a function of the number of instructions executed for a given number of inputs.
13. Non-polynomial problems usually take longer to solve than polynomial problems if the number of inputs is very large.

Multiple-Choice Questions

14. c
15. a
16. c
17. b
18. d
19. a
20. d
21. c
22. c
23. b
24. d
25. c
26. c
27. a
28. c
29. d
30. d
31. c
32. b

- 33. b
- 34. d
- 35. d

Exercises

36.

```
Z <==== X - Y
Z <==== X
TEMP <==== Y
while TEMP
{
  decr Z
  decr TEMP
}
```

37.

```
if X < Y then A1 else A2
TEMP1 <==== X
TEMP2 <==== Y
while TEMP1
{
  if TEMP2
  {
    decr TEMP1
    decr TEMP2
  }
  else
  {
    TEMP1 <==== 0
  }
}
if TEMP2 then A1 else A2
```

38.

```
if X > Y then A1 else A2
TEMP1 <==== X
TEMP2 <==== Y
while TEMP2
{
```

```

    if TEMP1
    {
        decr TEMP1
        decr TEMP2
    }
    else
    {
        TEMP2 <==== 0
    }
}
if TEMP1 then A1 else A2

```

39.

```

while X > Y { actions }
if X > Y then TRUE <==== 1 else TRUE <==== 0
while TRUE
{
    actions
    if X > Y then { } else TRUE <==== 0
}

```

40.

```

while X < Y { actions }
if X < Y then TRUE <==== 1 else TRUE <==== 0
while TRUE
{
    actions
    if X < Y then { } else TRUE <==== 0
}

```

41.

```

while X == Y { actions }
if X < Y then TRUE <==== 0 else if X > Y then TRUE <==== 0 else TRUE <==== 1
while TRUE
{
    actions
    if X < Y then TRUE <==== 0 else if X > Y then TRUE <==== 0 else TRUE <==== 1
}

```

42. Table 17.1 shows the transition table for $X \lll 0$

Table 17.1 Exercise 42

Current State	Read	Write	Move	Next State
Start Zero	#	#	====>	ToEnd
ToEnd	1	1	====>	ToEnd
ToEnd	&	blank	<====	Delete
Delete	1	blank	<====	Delete
Delete	#	#	====>	Zero
Zero	blank	&	<====	StopZero

43. The following is the transition table for $X \lll n$

Table 17.2 Exercise 43

Current State	Read	Write	Move	Next State
StartSetToN	#	#	none	StartZero
<i>Insert Table 17.1 of Exercise 42 Here.</i>				
StopZero	#	#	====>	Set0
Set0	any	1	====>	Set1
Set1	any	1	====>	Set2
		.		
		.		
		.		
Setn	any	&	<====	Backward
Backward	not a #	same as read	<====	Backward
Backward	#	#	none	StopSetToN

44. For this macro, X is first on the tape and Y is second. It's much easier to change a value if there is nothing following that value on the tape. First, we define a macro to add the current value to the one following it. This has applications beyond simple assignment (addition and multiplication, for instance). We walk through X and increment Y for each '1' we find, using a blank character as a place holder in X. Note that this code repeats the code for Incr X with one difference: it skips blank

characters on its way through the second number. This is necessary for the Power function in Exercise 47 below. Table 17.3 is the transition table for $Y += X$

Table 17.3 Exercise 44: Part I

Current State	Read	Write	Move	Next State
StartAddThisToNext	#	#	====>	ToStart
ToStart	&	&	<====	ToStart
ToStart	1	1	<====	ToStart
ToStart	blank	blank	<====	ToStart
ToStart	#	#	====>	Advance
Advance	blank	blank	====>	Advance
Advance	1	blank	====>	IncrNext
IncrNext	1	1	====>	IncrNext
IncrNext	&	&	====>	IncrThis
IncrThis	#	#	====>	Forward
Forward	1	1	====>	Forward
Forward	blank	blank	====>	Forward
Forward	&	1	====>	Added
Added	any	&	<====	Backward
Backward	not a #	same as read	<====	Backward
Backward	#	#	<====	Advance
Advance	&	&	<====	RestoreOnes
RestoreOnes	blank	1	<====	RestoreOnes
RestoreOnes	#	#	none	StopAddThisTo- Next

Now for the problem at hand. We first zero-out the second variable (Y), then we add X to it. Table 17.4 is the transition table for $Y <==== X$.

Table 17.4 Exercise 44: Part II

Current State	Read	Write	Move	Next State
StartSetY	#	#	====>	SetNextToZero
SetNextToZero	1	1	====>	SetNextToZero
SetNextToZero	&	&	====>	StartZero
<i>Insert Table 17.1 from Exercise 42 here.</i>				
StartZero	#	#	<====	StartAddThisToNext

Table 17.4 Exercise 44: Part II

Current State	Read	Write	Move	Next State
<i>Insert macro from Table 17.3 here.</i>				
StopAddThisToNext	#	#	none	StopSetY

45. For this macro, X and Y are first on the tape and Z is last. It's much easier to change a value if there is nothing following that value on the tape. We set the value of Z to the value of X and then add Y using the macro defined in #44 above. Table 17.5 shows the transition table for $Z \leq X + Y$.

Table 17.5 Exercise 45

Current State	Read	Write	Move	Next State
StartAdd	#	#	====>	SetThirdToZero
SetThirdToZero	1	1	====>	SetThirdToZero
SetThirdToZeroo	&	&	====>	SetSecondToZero
SetSecondToZero	#	#	====>	SetSecondToZero
SetSecondToZero	1	1	====>	SetSecondToZero
SetSecondToZero	&	&	====>	StartZero
<i>Insert Table 17.1 of Exercise 42 Here.</i>				
StopZero	#	#	<====	Back2
Back2	&	&	<====	Back2
Back2	1	1	<====	Back2
Back2	#	#	<====	Back1
Back1	&	&	<====	Back1
Back1	1	1	<====	Back1
Back1	blank	blank	<====	Back1
Back1	#	#	====>	Advance2
Advance2	blank	blank	====>	Advance2
Advance2	1	blank	====>	IncrSecond
IncrSecond	1	1	====>	IncrSecond
IncrSecond	&	&	====>	IncrNext
IncrNext	#	#	====>	IncrNext
IncrNext	1	1	====>	IncrNext
IncrNext	&	&	====>	StartIncr
<i>Insert Table 17.2 from text (page 323) here.</i>				

Table 17.5 Exercise 45

Current State	Read	Write	Move	Next State
StopIncr	#	#	<===	Advance2
Advance2	&	&	===>	StartAddThisToNext
<i>Insert macro defined in Table 17.3 (solution to Exercise 44) here.</i>				
StopAddThisToNext	#	#	<===	RestoreOnes
RestoreOnes	blank	1	<===	RestoreOnes
RestoreOnes	#	#	none	StopAdd

46. For this macro, X and Y are first on the tape and Z is last. It's much easier to change a value if there is nothing following that value on the tape. First, we set Z to zero and then we add the value of Y X times using the macro defined in #44 above. Table 17.6 shows the transition table for $Z \leftarrow X * Y$.

Table 17.6 Exercise 46

Current State	Read	Write	Move	Next State
StartMultiply	#	#	===>	SetThirdToZero
SetThirdToZero	1	1	===>	SetThirdToZero
SetThirdToZero	&	&	===>	SetSecondToZero
SetSecondToZero	#	#	===>	SetSecondToZero
SetSecondToZero	1	1	===>	SetSecondToZero
SetSecondToZero	&	&	===>	StartZero
<i>Insert Table 17.1 of Exercise 42 here.</i>				
StopZero	#	#	<===	Back2
Back2	&	&	<===	Back2
Back2	1	1	<===	Back2
Back2	#	#	<===	Back1
Back1	&	&	<===	Back1
Back1	1	1	<===	Back1
Back1	blank	blank	<===	Back1
Back1	#	#	===>	Advance3
Advance3	blank	blank	===>	Advance3
Advance3	1	blank	===>	AddNextToLast
AddNextToLast	1	1	===>	AddNextToLast
AddNextToLast	&	&	===>	StartAddThisToNext

Table 17.6 Exercise 46

<i>Current State</i>	<i>Read</i>	<i>Write</i>	<i>Move</i>	<i>Next State</i>
<i>Insert macro defined in Table 17.3 of Exercise 44 here.</i>				
StopAddThisToNext	#	#	<===	Back3
Back3	&	&	<===	RestoreOnes
RestoreOnes	blank	1	<===	RestoreOnes
RestoreOnes	#	#	none	StopMultiply

47. Table 17.7 shows the transition table for comp(X).

Table 17.7 Exercise 47

<i>Current State</i>	<i>Read</i>	<i>Write</i>	<i>Move</i>	<i>Next State</i>
StartCompX	#	#	====>	Check
Check	&	1	====>	WasZero
WasZero	Any	&	<===	Done
Done	1	1	<===	Done
Done	#	#	none	StopComp
StopComp	1	1	====>	SetToZero
SetToZero	not a &	what was read	====>	SetToZero
SetToZero	&	blank	<===	Delete
Delete	1	blank	<===	Delete
Delete	#	#	====>	Zero
Zero	blank	&	<===	StopCompX

48. For this macro, X is first on the tape and the data that gets processed by A1 and A2 follows X. Table 17.8 shows the transition table for if X then A1 else A2.

Table 17.8 Exercise 48

<i>Current State</i>	<i>Read</i>	<i>Write</i>	<i>Move</i>	<i>Next State</i>
StartIf	#	#	====>	CheckX
CheckX	1	1	====>	XIsTrue
XIsTrue	1	1	====>	XIsTrue
XIsTrue	&	&	====>	StartA1
<i>INSERT CODE FOR A1 HERE.</i>				
StopA1	#	#	<===	Done
Done	1	1	<===	Done
Done	#	#	none	StopIf

Table 17.8 Exercise 48

<i>Current State</i>	<i>Read</i>	<i>Write</i>	<i>Move</i>	<i>Next State</i>
StopIf	&	&	====>	StartA2
INSERT CODE FOR A2 HERE.				
StopA2	#	#	<====	Done

49. The macro $X1 \lll 0$ is written as:

```
while X1
{
  decr X1
}
```

Therefore the Godel number is CF1DBF1E

50. The macro $X1 \lll n$ is written as:

```
while X1
{
  decr X1
}
incr X1
incr X1
incr X1
...
incr X1
```

Therefore, the Godel number is CF1DBF1EAF1AF1AF1...AF1 with AF1 repeated n times.

51. The macro $X3 \lll X1 + X2$ is written as:

```
while X3
{
  decr X3
}
while X4
{
  decr X4
}
while X1
{
```

```
incr X3
incr X4
decr X1
}
while X4
{
incr X1
decr X4
}
while X2
{
incr X3
incr X4
decr X2
}
while X4
{
incr X2
decr X4
}
```

Therefore the Godel number is:

CF3DBF3ECF4DBF4ECF1DAF3AF4BF1ECF4DAF1BF4ECF2DAF3AF4BF2E
CF4DAF2BF4E

