Universality of Turing Machine and Its Limitations 7

The universal Turing machine is the one that, when given a description of Turing machine M and input w, can simulate the behavior of M with input w. In this chapter we construct the universal Turing machine. This means that the universal Turing machine can behave like any Turing machine with any input if the description of the Turing machine to be simulated is given together with its input. On the other hand, there exists a limit to the computational power of Turing machines. This limitation is shown by giving the halting problem, which any Turing machine can never solve. The halting problem asks whether, when a Turing machine M and an input w are given, M with input w will eventually halt or continue to run forever. As another example of a problem that any Turing machine cannot solve, we give the Post correspondence problem. The Post correspondence problem asks whether or not, when a collection of pairs of strings is given, there exists a sequence of the pairs (repetitions permitted) that has certain properties of a match.

7.1 Universal Turing Machine

When given a description of a Turing machine (TM) and its input, it can be mechanically performed to simulate its moves in steps. Therefore, according to the Church–Turing thesis, there *exists* a Turing machine that performs so. But how can we construct such a Turing machine? That's another story. The Turing machine that does this task is called the *universal Turing machine*. As will be shown, the universal Turing machine behaves essentially the same way as we trace a program of a Turing machine written in terms of 5-tuples. Let U denote the universal Turing machine and let M denote any Turing machine with input w. When U is given as input M, or more precisely a description of M, and w, U simulates the behavior of M with input w, by examining descriptions of M and its input w. Shortly we explain how to describe a Turing machine as a string.

The notion of the universal Turing machine reminds us of the fact that a modern computer has its hardware fixed and what we want to perform is stored as a program in its memory; this is called a stored-program computer. You can easily see that

DOI 10.1007/978-0-85729-535-4_7, © Springer-Verlag London Limited 2011

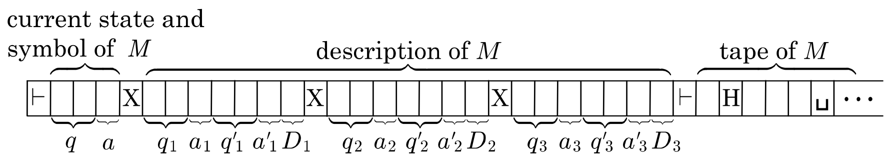
current state and symbol of M
description of M
tape of M
q
a
q_1 a_1 q'_1 a'_1 D_1
q_2 a_2 q'_2 a'_2 D_2
q_3 a_3 q'_3 a'_3 D_3
X
H

Fig. 7.1 Tape of universal TM U

there exists a correspondence between the stored-program computer and the universal Turing machine: the hardware corresponds to the universal Turing machine, while the stored program itself corresponds to a description of a Turing machine to be simulated which is contained in the tape of the universal Turing machine initially. Von Neumann wrote a report on the EDVAC (Electronic Discrete Variable Automatic Computer)—one of the earliest electronic computers—in which he introduced the idea of a stored-program computer, which owes to the concept of the universal Turing machine due to Alan M. Turing. In fact, following the behavior of the universal Turing machine described later, the readers are expected to grasp intuitively the core idea of the stored-program computers.

Let us proceed to the details of the universal Turing machine. Throughout this chapter we assume that a Turing machine M that we want to simulate is deterministic unless otherwise stated. Initially, the universal Turing machine has on its tape the description of a Turing machine M, denoted by $\langle M \rangle$, and an input to M, denoted by w, where $\langle M \rangle$ is the string that somehow encodes a Turing machine M. More specifically, as illustrated in Fig. 7.1, the contents of the tape of the universal Turing machine is stored in three regions, each containing the following information: the current state and the symbol under the head of M; the description of M; the tape contents of M. As illustrated in Fig. 7.1, in the machine description region the sequence of 5-tuples separated by X's is placed, where each 5-tuple is expressed as the binary sequence representing (q, a, q', a', D) such that $\delta(q, a) = (q', a', D)$, where δ denotes M's transition function. To the right of the machine description region there exists M's tape region, where the current content of M's tape is placed except for the square on which the head is located. On that square the marker H is placed instead of the symbol under the head. Finally, to the left of the machine description region, there exists the machine condition region, where we find M's current state q and the current symbol a under the head of M, namely, the symbol that is supposed to be placed on the square of H. So the head position and the symbol under the head are stored separately. It will be explained shortly that the current symbol square in the machine condition region contains the symbol that M has just read or is just about to write, depending on the phase of the simulation.

It is easy to figure out how the universal Turing machine U behaves by consulting the contents of the tape of the universal Turing machine just described. That is, the Turing machine U continues simulating the behavior of M one step at a time: by remembering M's current state and the current symbol in the machine condition region and by searching for the 5-tuple that is compatible with the current state and the current symbol, U accordingly finds out what state to go into, what symbol to write, which way to move the head, and how to update the tape content. More specifically, U takes up and memorizes (q, a) in the machine condition region, finds out the 5-tuple (q, a, q', a', D) whose first and second components coincide with (q, a), and updates the machine condition region and M's tape region according to (q', a', D).

Before proceeding to more details of the simulation, we should explain about the tape alphabet of the universal Turing machine U. First of all, U is just one fixed machine; hence, it has a fixed tape alphabet. On the other hand, the machines that are simulated by U are arbitrary and hence may have arbitrarily large numbers of states, although each of them is finite. So U cannot use a different symbol to represent each state of the machine to be simulated. How can we designate an arbitrarily large number of states by a fixed tape alphabet? The trick is as follows. When the machine to be simulated has m states and l is the least integer such that $m \leq 2^l$, we use binary sequences of length l to represent the states. In Fig. 7.1 this l is assumed to be 2. As for a tape alphabet of a machine to be simulated we can argue similarly. Namely, we can use binary sequences to represent an arbitrary large number of the tape symbols of the machines to be simulated. But to make matters simple we assume instead that the tape alphabet of any machine to be simulated is fixed as binary symbols. Furthermore, the directions L and R are represented by 0 and 1, respectively. Then, by adding the symbols $A, B, X, H, S, \vdash$ to the binary symbols 0 and 1, we set the tape alphabet of the U as follows:

$$\Gamma = \{0, 1, A, B, X, H, S, \vdash, \sqcup\}.$$

Before proceeding to explain the state diagram of the universal Turing machine U in Fig. 7.3, it would be helpful to trace a higher level description of U, which is given as follows.

The behavior of the universal Turing machine U:

1. Search for a 5-tuple (q, a, q', a', D) in the machine description region such that its first and second elements coincide with the current state and the current symbol in the machine condition region.

2. Write (q', a') of the 5-tuple (q, a, q', a', D) found in **1** into the machine condition region. Furthermore, memorize as a state the direction D of that 5-tuple.

3. Memorize as a state the a' written on the current symbol square in **2** and write symbol S on that square. Write the a' memorized on the square on which H is placed, and move U's head one square to the direction indicated by D that is memorized in **2**.

4. Memorize as a state the symbol a'' that the head reads, and write symbol H on the square under the head. Then search for the square with symbol S and write on that square the a'' memorized. Go to **1**.

Figure 7.2 illustrates how the contents of the tape will change when U simulates M. The four tapes in the figure represent some parts of the contents just at the end of the four stages of the behavior of U. Among the three 5-tuples $(q_1, a_1, q'_1, a'_1, D_1), \ldots, (q_3, a_3, q'_3, a'_3, D_3)$ in the machine description region, only

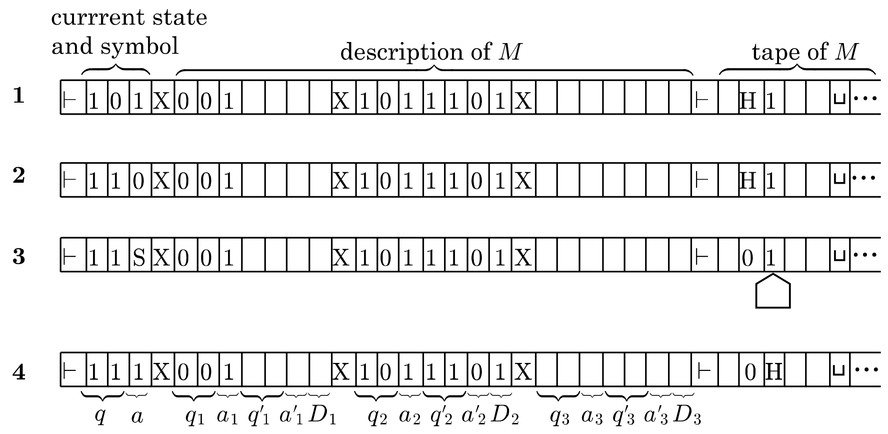
currrent state
and symbol
description of M
tape of M
1
2
3
4
q a q_1 a_1 q'_1 a'_1 D_1 q_2 a_2 q'_2 a'_2 D_2 q_3 a_3 q'_3 a'_3 D_3

Fig. 7.2 Transitions of the contents of the tape of the universal TM U

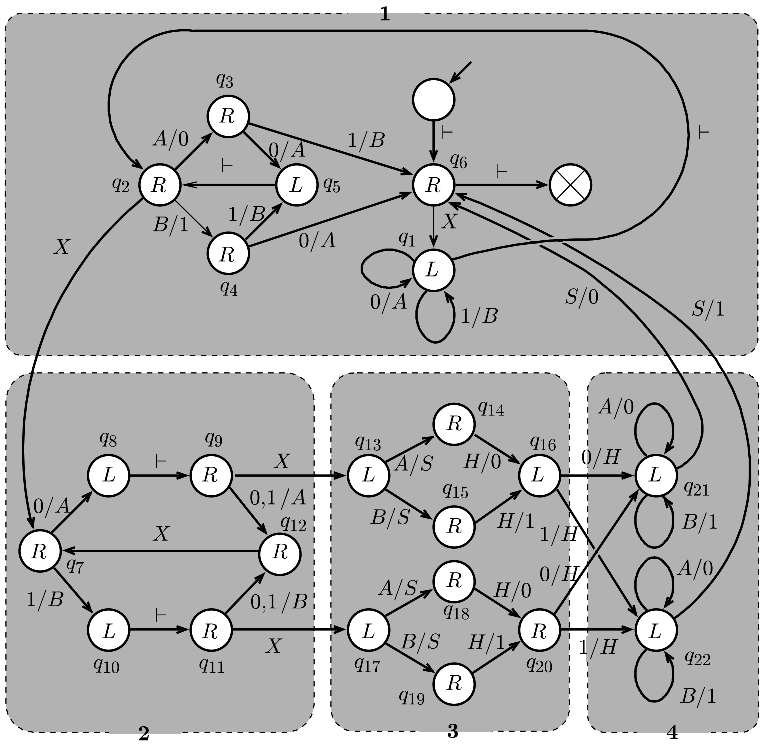
1
2
3
4
q_1 q_2 q_3 q_4 q_5 q_6 q_7 q_8 q_9 q_{10} q_{11} q_{12} q_{13} q_{14} q_{15} q_{16} q_{17} q_{18} q_{19} q_{20} q_{21} q_{22}
$A/0$ $B/1$ $0/A$ $1/B$ X $S/0$ $S/1$
$0,1/A$ $0,1/B$ A/S B/S $H/0$ $H/1$ $0/H$ $1/H$

Fig. 7.3 State diagram of the universal TM U

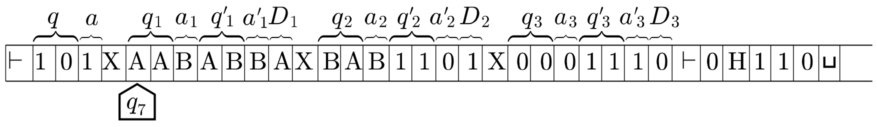
q
a
q1
a1
q'1
a'1
D1
q2
a2
q'2
a'2
D2
q3
a3
q'3
a'3
D3
⊢ 1 0 1 X A A B A B B A X B A B 1 1 0 1 X 0 0 0 1 1 1 0 ⊢ 0 H 1 1 0 ⊔
q7

Fig. 7.4 Configuration at the moment when a transition from q_2 to q_7 occurs

the second is shown in the figure. The states are represented by strings in $\{0, 1\}^2$ and both the directions and the tape symbols are represented by binary symbols in $\{0, 1\}$. As mentioned above, in **1**, the 5-tuple $(q_2, a_2, q'_2, a'_2, D_2)$ is searched for because the first and the second components coincide with the pair of the current state and symbol $(10, 1)$. In **2**, the $q'_2 = 11$ and $a'_2 = 0$ of that 5-tuple are written in the machine condition region. The direction D_2 (in this case, 1 meaning R) is memorized as a state. In **3**, $a'_2 = 0$ is memorized and S is written in that square on which the a'_2 was temporarily written. The memorized symbol a'_2 is written on the square on which symbol H is placed. Then the head of U is moved one square to the direction of D_2. In **4**, the symbol under the current head is memorized as a state and H is written on that square which is the new head position. The memorized symbol is written on the square on which symbol S is placed as the new current symbol.

Figure 7.3 gives the state diagram of the universal Turing machine U. The diagram is divided into four parts attached with **1** through **4**, each corresponding to one of the four stages of the behavior of the U. The readers who have grasped intuitively how the universal Turing machine works so far may skip the rest of this section. The readers who are not yet convinced may figure it out by tracing the more detailed behavior of U, which will be given in what follows.

We begin with the behavior of stages **1** and **2**. These parts of the diagram of Fig. 7.3 coincide with the diagram of Fig. 6.6, which stores and retrieves information. In the case of U, the three bits of (q_i, a_i) correspond to a name, whereas the four bits of (q'_i, a'_i, D_i) correspond to an item. So, as opposed to the former case of the diagram of Fig. 6.6, the length of a name is not equal to that of an item. The point is that U makes use of the difference of the lengths as follows. Figure 7.4 shows the moment when U has found out (q_2, a_2) of the second 5-tuple that coincides with the current state and symbol in the machine condition region. Notice that, to keep the symbols used simple, the same symbol q_i is used for the states of both M and U. The machine in the configuration shown in Fig. 7.4 is about to copy the part (q'_2, a'_2, D_2) into the machine condition region. In doing so the squares containing $(q_1, a_1, q'_1, a'_1, D_1)$ and (q_2, a_2) will be skipped because the symbols 0 and 1 on these squares are changed to the symbols A and B, respectively. In general, the symbols 0 and 1 that are already scanned will be changed to A and B so that they will be skipped in later searching and storing. In this figure note that 0 and 1 correspond to A and B as well as to L and R, respectively.

Let's see more precisely how to copy the first three bits of $q'_2 a'_2 D_2 (= 1101)$, namely 110, into the machine condition region. Starting on the leftmost X with state q_7, the machine copies 110 one symbol at a time repeatedly as follows: shift to the right until U reads for the first time some 0 or 1; memorize the symbol just

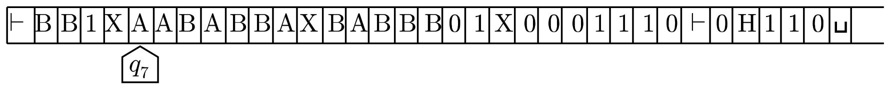
⊢ B B 1 X A A B A B B A X B A B B B 0 1 X 0 0 0 1 1 1 0 ⊢ 0 H 1 1 0 ⊔
q_7

Fig. 7.5 Configuration at the moment when the first two bits of 110 are copied

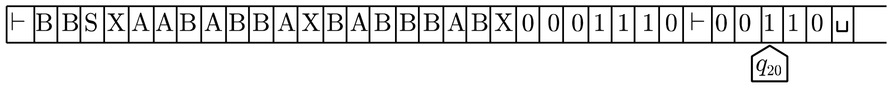
⊢ B B S X A A B A B B A X B A B B B A B X 0 0 0 1 1 1 0 ⊢ 0 0 1 1 0 ⊔
q_{20}

Fig. 7.6 Configuration at the moment when U is in q_{20}

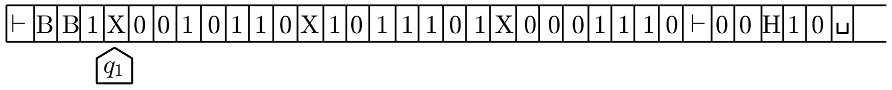
⊢ B B 1 X 0 0 1 0 1 1 0 X 1 0 1 1 1 0 1 X 0 0 0 1 1 1 0 ⊢ 0 0 H 1 0 ⊔
q_1

Fig. 7.7 Configuration at the moment when U is in q_1

read and replace the symbol with A or B ($q_7 \xrightarrow{0/A} q_8$ or $q_7 \xrightarrow{1/B} q_{10}$); shift to the left until U reads the leftmost $\vdash$ ($q_8 \xrightarrow{\vdash} q_9$ or $q_{10} \xrightarrow{\vdash} q_{11}$); shift to the right until U reads for the first time some 0 or 1 and write on the square the symbol memorized with A or B ($q_9 \xrightarrow{0,1/A} q_{12}$ or $q_{11} \xrightarrow{0,1/B} q_{12}$); shift to the right until U reads the first X to prepare for the next cycle ($q_{12} \xrightarrow{X} q_7$). Figure 7.5 shows the tape contents at the time when the first two bits of $q_2'a_2' = 110$ have been copied as BB on the leftmost squares between $\vdash$ and X. When starting at the configuration of Fig. 7.4 with $q_2'a_2'D_2 = 1101$, the machine repeats the cycle three times, and the content on the leftmost five squares becomes $\vdash BBAX$. Then the machine remembers the fourth bit 1 of $q_2'a_2'D_2 = 1101$ by going from q_7 to q_{10} and moves to the leftmost $\vdash$ and then searches for 0 or 1 encountered for the first time going rightward. But this time before the machine encounters 0 or 1 it reads X and does transition $q_{11} \xrightarrow{X} q_{10}$, thereby proceeding to **3**.

In stage **3** the machine writes the symbol a_i' written on the current symbol square onto the square on which symbol H is placed and moves the head one square to the direction of D_i (in our case to the right) that is memorized as a state in **2**. That head position is the new head position for the next step, whose configuration is shown in Fig. 7.6. While doing what has been described so far, the machine writes S on the current symbol square. Then the machine goes to **4**.

Finally in **4**, the machine memorizes the symbol on the new head position as a state (state q_{21} or q_{22}) and moves its head to the left, restoring 0 and 1 until it encounters S, which will be replaced with the new memorized symbol a_i' (in our case $a_i' = 1$). Figure 7.7 shows the configuration at this moment of the simulation. Then the machine goes to **1**. We omit the accept and reject states in Fig. 7.3. To complete the diagram, we must modify it by adding, for example, the transitions to the accept state of U when in the course of the simulation an accept state of M is encountered. But the details of the modification are omitted.