Tik-79.148 **Spring 2001**
Introduction to Theoretical Computer Science
Tutorial 9
Solutions to Demonstration Exercises

4. Constructing a complex Turing machine can be very laborious. With the help of machine schemas it is possible to combine simple Turing machines to obtain more complex ones.

   The simple basic machines are:

   - For every symbol in Σ there is a machine that writes the symbol in the tape and then stops with out moving its read/write head.
   - Machines R and L that move the head one step to right or left, respectively, and then stop.

   The machines in the exercise work as follows:

   (a) The machine moves the head to the right in an eternal loop.

   (b) The machine first moves its head one step to the right. If it now scans an a, it will replace it by a b. If it scans a b, it will replace by an a.

   (c) The machine moves its head two steps to the right. Note that if the head already is in the left end of the tape, the behaviour of the machine is not well defined. (The starting mark $\triangleright$ always causes the head to move one step to the right.)

   (d) The machine first moves its head one step to the left, and if it now scans a non-empty symbol it moves the head back to the right.

5. The three most fundamental terms related to Turing machines are:

   - *Turing decidable*
   - *Turing acceptable, semi-decidable*
   - *Turing computable*

   A language L is Turing decidable if there exists a Turing machine M so that

   $$(s, \triangleright \sqcup w\underline{\sqcup}) \vdash_M^* \begin{cases} (h, \triangleright \sqcup Y\underline{\sqcup}), & \text{if } w \in L \\ (h, \triangleright \sqcup N\underline{\sqcup}), & \text{if } w \notin L \end{cases}$$

   For all strings $w \in \Sigma^*$ it is possible to determine whether they belong to the language or not.

   A language L is Turing acceptable if there exists a Turing machine M so that

   $$M \text{ stops with the input } w \Leftrightarrow w \in L$$

   Turing machine that accepts (or semidecides) a language stops if and only if the input string belongs to the language. In other cases the machine continues its computation forever. There are languages that are Turing acceptable but not Turing decidable. For example:

$$L = \{M \mid M \text{ is a Turing machine that halts on the input } e\}$$

is such a language.

You can think that a Turing machine that decides a language L actually computes the function $f : \Sigma^* \to \{y, n\}$. This can be generalized to arbitrary functions on strings $g : \Sigma^* \to \Sigma^*$. We get the definition: a Turing machine computes a function g, if

$$f(w) = u \Leftrightarrow (s, \triangleright \sqcup w\underline{\sqcup}) \vdash_M^* (h, \triangleright \sqcup u\underline{\sqcup})$$

The machine has a string x as input, an in the end of the computation there is the string $f(x)$ in the tape. Function with several arguments can be defined in a similar way. In this case, the arguments are separated by one blank from each other and written one after another in the beginning of the tape. This is the initial configuration of the machine M. (Example: $\triangleright \sqcup w_1 \sqcup w_2 \sqcup \cdots \sqcup w_n\underline{\sqcup}$).

(a) Turing machine that accepts the language a^*ba^*b:

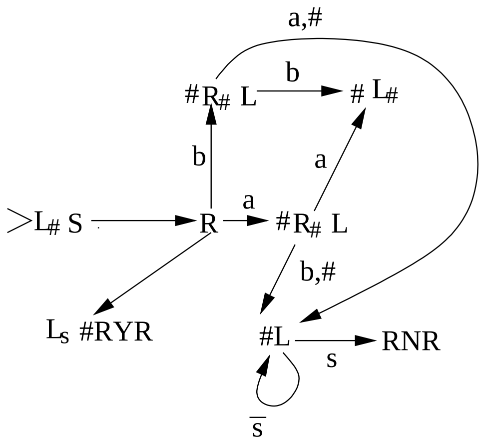
$$\begin{aligned} M &= (K, \Sigma, \delta, s) \\ K &= \{q_0, q_1, q_2, q_3\} \\ \Sigma &= \{a, b, \sqcup\} \\ s &= q_0 \end{aligned}$$

q	σ	$\delta(q, \sigma)$
q_0	$\sqcup$	(q_1, L)
q_1	a	(q_1, a)
q_1	b	(q_2, L)
q_1	$\sqcup$	$(q_1, \sqcup)$
q_2	a	(q_2, L)
q_2	b	(q_3, L)
q_2	$\sqcup$	$(q_2, \sqcup)$
q_3	a	(q_3, L)
q_3	b	(q_3, b)
q_3	$\sqcup$	$(h, \sqcup)$

The starting point of the construction of the machine was a corresponding state machine. The machine can stop only when the input string belongs to the language (by the definition of semideciding), so the machine is left to an eternal loop every time a wrong type of string is found. In fact, the language given in the exercise is decidable. That is, it is possible to construct a machine that always halts and expresses whether the input string belonged to the language or not (compare to the next exercise).

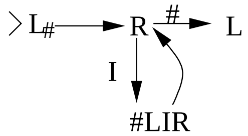
(b) You are asked to construct a machine that gets as input $(\sqcup w\underline{\sqcup})$, and after the computing the tape contains $(\sqcup Y\underline{\sqcup})$, if $w \in L$ or $(\sqcup N\underline{\sqcup})$, if $w \notin L$.

In the construction, the beginning of the tape is first marked with a symbol S. After that the machines compares the first letter of the word to the last letter, erases them and repeats.

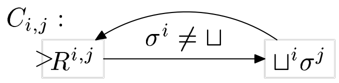


A machine L_σ scans the tape to the left until it finds the first occurrence of the symbol σ.

(c) We construct a machine that computes the function $f(m,n) = m+n$. The input of the machine is of the form $(\sqcup I^m \sqcup I^n \underline{\sqcup})$, and after the computation the tape contains $(\sqcup I^{m+n}\underline{\sqcup})$. The easiest way to accomplish this is to move the string I^n one step to the left:



6. First we define machine $C_{i,j}$ that copies the symbols on the right side of the head in the tape i to the tape j, until the first blank is scanned.



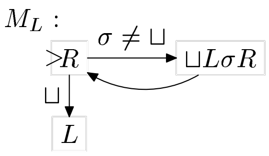
For example: The configuration

▷	$\underline{\sqcup}$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$
▷	$\sqcup$	1	$\underline{0}$	1	0	$\sqcup$

is transformed to

▷	$\sqcup$	1	0	$\underline{\sqcup}$	$\sqcup$	$\sqcup$
▷	$\sqcup$	1	0	1	0	$\underline{\sqcup}$

Next we define another auxiliary machine M_L that moves the string on the right side of the head one step to the left, and leaves the head to the right end of the moved string.



For example, the configuration

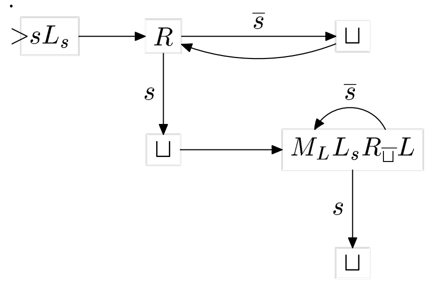
$\triangleright$	$\underline{\sqcup}$	0	1	0

is transformed to

$\triangleright$	0	1	0	$\underline{\sqcup}$

An auxiliary machine E moves string on the right side of the head to the beginning of the tape.

E :



For example, the configuration

$\triangleright$	$\sqcup$	0	1	$\underline{\sqcup}$	1	0	$\sqcup$

is transformed to

$\triangleright$	$\underline{\sqcup}$	1	0	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$

With the help of the auxiliary machines we construct a 2-tape machine $\sum_{1,2}$ that sums two binary numbers. The machine is based on the text book example 4.3.2 and it is relatively complex[1] The machine in the example of the book only works correctly, if the two numbers have equal number of bits. Additionally, the machine leaves the tape in an erroneous configuration. Correcting these mistakes takes quite an effort.

Example: Starting from the configuration

$\triangleright$	$\underline{\sqcup}$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$
$\triangleright$	$\sqcup$	1	0	$\sqcup$	1	0	1	$\underline{\sqcup}$

The machine $\sum_{1,2}$ marks the beginnings of the tapes by symbol s, copies the first word of the input to the second tape, and moves the heads to the end of the tapes:

$\triangleright$	s	1	0	$\underline{\sqcup}$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$
$\triangleright$	s	1	0	$\sqcup$	1	0	1	$\underline{\sqcup}$

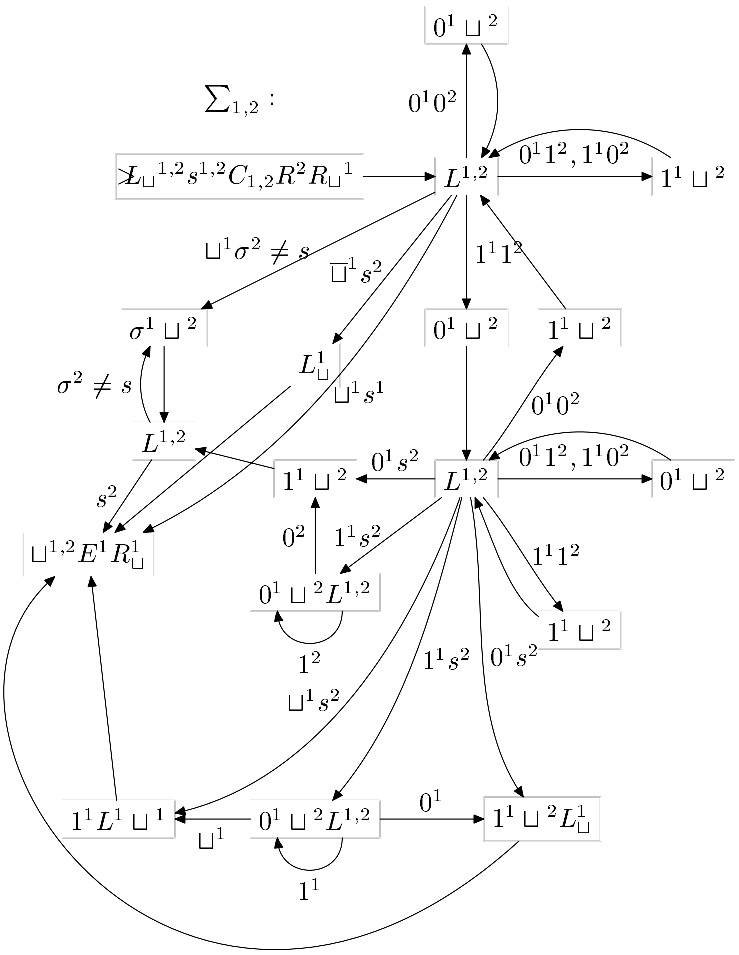
[1]If any of the readers find a simpler solution, the assistants of the course would be very grateful if it were not told to them.

After that the machines scans both numbers from their end to the beginning, and calculates their sum bit by bit. At the same time, the second tape is being cleaned.

$\triangleright$	$\underline{\text{s}}$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$
$\triangleright$	s	1	0	$\sqcup$	$\underline{1}$	1	1	$\sqcup$

Calculation ends when either of the numbers ends. In this example, the upper number ended first. The answer, 111, is moved to the beginning of the first tape, and symbols s are removed.

$\triangleright$	$\underline{\sqcup}$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$
$\triangleright$	$\sqcup$	1	1	0	$\underline{\sqcup}$

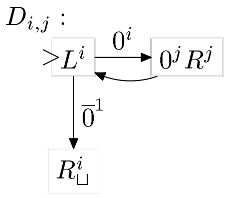


Before implementing multiplication, two more auxiliary machines are needed: $D_{i,j}$ and W_L. The machine $D_{i,j}$ computes the number of zeros in the end of the number in the first tape, and multiplies the number in the second tape by 2 for each zero. For example

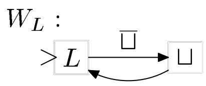
$$\left\{\begin{array}{l}\begin{array}{|c|c|c|c|c|c|c|c|c|}\hline \triangleright & \sqcup & 1 & 0 & \underline{\sqcup} & \sqcup & \sqcup & \sqcup & \sqcup \\ \hline\end{array}\\ \begin{array}{|c|c|c|c|c|c|c|c|c|}\hline \triangleright & \sqcup & 1 & 0 & \sqcup & 1 & 0 & 0 & \underline{\sqcup} \\ \hline\end{array}\end{array}\right.$$

is transformed to

$$\left\{\begin{array}{l}\begin{array}{|c|c|c|c|c|c|c|c|c|}\hline \triangleright & \sqcup & 1 & 0 & 0 & 0 & \underline{\sqcup} & \sqcup & \sqcup \\ \hline\end{array}\\ \begin{array}{|c|c|c|c|c|c|c|c|c|}\hline \triangleright & \sqcup & 1 & 0 & \sqcup & \underline{1} & 0 & 0 & \sqcup \\ \hline\end{array}\end{array}\right.$$



The machine W_L cleans the tape by removing the string on the left side of the head.



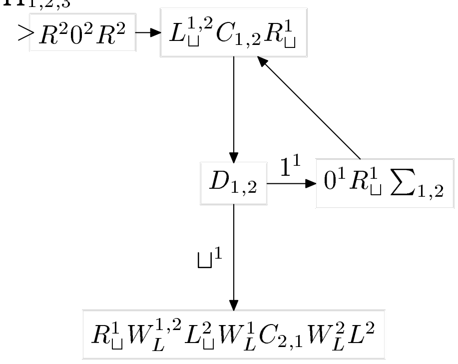
Multiplication of binary numbers can be accomplished by series of summations and multiplications by two. For example:

$$101_2 \cdot 110_2 = 101_2 \cdot 2^2 + 101_2 \cdot 2^1$$

Thus, multiplication can be accomplished by the next algorithm and a 3-tape Turing machine:

1. Write 0 in the beginning of the second tape.
2. Copy the first factor to the second tape.
3. Go through the second factor from the end to the beginning. For each zero in the end of the number multiply the factor in the second tape by two.
4. If there have been found a 1-bit in the second factor in the end of the third step, replace the 1-bit by a 0-bit. If the number only contained zeros, multiplication is ready and the result can be found in the beginning of the second tape.
5. Compute the sum of the numbers in the second tape with the help of the third tape, and return to the step 2.

$\prod_{1,2,3}$:



Even if the machine $\prod_{1,2,3}$ has three tapes, it only uses two tapes explicitly, and the third tape serves only as an auxiliary tape for the summation.

Finally, let's study how the machine calculates $2 \cdot 3$ (that is, $10_2 \cdot 11_2$).

In the beginning the first two tapes contain:

$\triangleright$	$\underline{\sqcup}$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$
$\triangleright$	$\sqcup$	1	0	$\sqcup$	1	1	$\underline{\sqcup}$

The product is initialized to zero, and the first factor is copied to the second tape.

$\triangleright$	$\sqcup$	0	$\sqcup$	1	0	$\underline{\sqcup}$	$\sqcup$
$\triangleright$	$\sqcup$	1	0	$\sqcup$	1	1	$\underline{\sqcup}$

Because the last bit of the second factor is not zero, there is no need for multiplication by two but the summation can be done directly.

$\triangleright$	$\sqcup$	1	0	$\underline{\sqcup}$	$\sqcup$	$\sqcup$	$\sqcup$
$\triangleright$	$\sqcup$	1	0	$\sqcup$	1	0	$\underline{\sqcup}$

The first factor is copied to the second tape again.

$\triangleright$	$\sqcup$	1	0	$\sqcup$	1	0	$\underline{\sqcup}$
$\triangleright$	$\sqcup$	1	0	$\sqcup$	1	0	$\underline{\sqcup}$

This time, the first factor end with a 0-bit, so the other factor will be multiplied by two.

$\triangleright$	$\sqcup$	1	0	$\sqcup$	1	0	0	$\underline{\sqcup}$
$\triangleright$	$\sqcup$	1	0	$\sqcup$	$\underline{1}$	0	$\sqcup$	$\sqcup$

Next, the last 1-bit in the first tape is replaced by a zero, and the numbers in the second tape will be summed.

$\triangleright$	$\sqcup$	1	1	0	$\underline{\sqcup}$	$\sqcup$	$\sqcup$
$\triangleright$	$\sqcup$	1	0	$\sqcup$	0	0	$\underline{\sqcup}$

During the next iteration the product is ready and it will be copied to the beginning of the first tape.

$$\left\{\begin{array}{l}
\begin{array}{|c|c|c|c|c|c|}\hline \triangleright & \underline{\sqcup} & \sqcup & \sqcup & \sqcup & \sqcup \\ \hline \end{array} \\
\begin{array}{|c|c|c|c|c|c|}\hline \triangleright & \sqcup & 1 & 1 & 0 & \underline{\sqcup} \\ \hline \end{array}
\end{array}\right.$$