

DISCRETE MATHEMATICS

W W L CHEN

© W W L Chen, 1997, 2003.

This chapter is available free to all individuals, on the understanding that it is not to be used for financial gains, and may be downloaded and/or photocopied, with or without permission from the author.

However, this document may not be kept on any information storage and retrieval system without permission from the author, unless such system is not accessible to any individuals other than its owners.

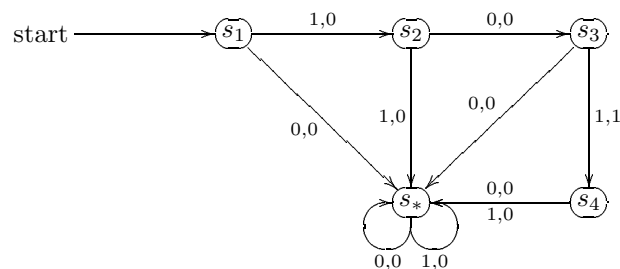
Chapter 7

FINITE STATE AUTOMATA

7.1. Deterministic Finite State Automata

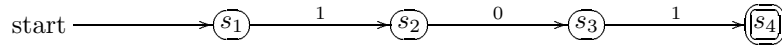
In this chapter, we discuss a slightly different version of finite state machines which is closely related to regular languages. We begin by an example which helps to illustrate the changes.

EXAMPLE 7.1.1. Consider a finite state machine which will recognize the input string 101 and nothing else. This machine can be described by the following state diagram:



We now make the following observations. At the end of a finite input string, the machine is at state s_4 if and only if the input string is 101. In other words, the input string 101 will send the machine to the state s_4 at the end of the process, while any other finite input string will send the machine to a state different from s_4 at the end of the process. The fact that the machine is at state s_4 at the end of the process is therefore confirmation that the input string has been 101. On the other hand, the fact that the machine is not at state s_4 at the end of the process is therefore confirmation that the input string has not been 101. It is therefore not necessary to use the information from the output string at all if we give state s_4 special status. The state s_* can be considered a dump. The machine will go to this state at the moment it is clear that the input string has not been 101. Once the machine reaches this state, it can never escape from this state. We can exclude the state s_* from the state diagram, and further

simplify the state diagram by stipulating that if $\nu(s_i, x)$ is not indicated, then it is understood that $\nu(s_i, x) = s_*$. If we implement all of the above, then we obtain the following simplified state diagram, with the indication that s_4 has special status and that s_1 is the starting state:



We can also describe the same information in the following transition table:

	ν	
	0	1
+s ₁ +	s _*	s ₂
s ₂	s ₃	s _*
s ₃	s _*	s ₄
-s ₄ -	s _*	s _*
s _*	s _*	s _*

We now modify our definition of a finite state machine accordingly.

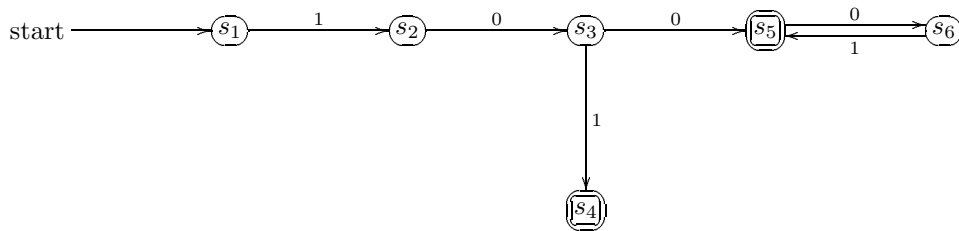
DEFINITION. A deterministic finite state automaton is a 5-tuple $A = (\mathcal{S}, \mathcal{I}, \nu, \mathcal{T}, s_1)$, where

- (a) \mathcal{S} is the finite set of states for A ;
- (b) \mathcal{I} is the finite input alphabet for A ;
- (c) $\nu : \mathcal{S} \times \mathcal{I} \rightarrow \mathcal{S}$ is the next-state function;
- (d) \mathcal{T} is a non-empty subset of \mathcal{S} ; and
- (e) $s_1 \in \mathcal{S}$ is the starting state.

REMARKS. (1) The states in \mathcal{T} are usually called the accepting states.

(2) If not indicated otherwise, we shall always take state s_1 as the starting state.

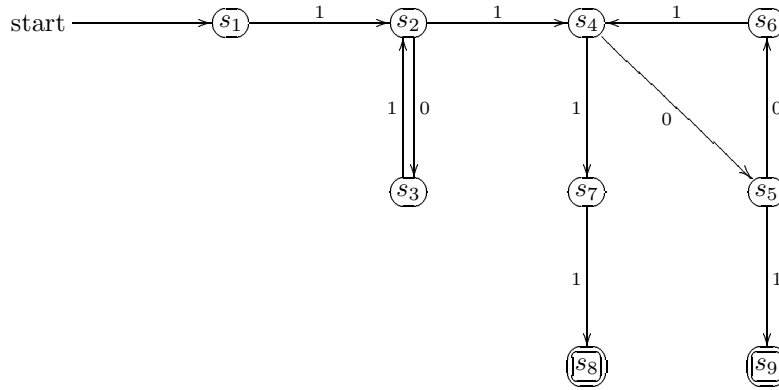
EXAMPLE 7.1.2. We shall construct a deterministic finite state automaton which will recognize the input strings 101 and 100(01)* and nothing else. This automaton can be described by the following state diagram:



We can also describe the same information in the following transition table:

	ν	
	0	1
+s ₁ +	s _*	s ₂
s ₂	s ₃	s _*
s ₃	s ₅	s ₄
-s ₄ -	s _*	s _*
-s ₅ -	s ₆	s _*
s ₆	s _*	s ₅
s _*	s _*	s _*

EXAMPLE 7.1.3. We shall construct a deterministic finite state automaton which will recognize the input strings $1(01)^*1(001)^*(0+1)1$ and nothing else. This automaton can be described by the following state diagram:



We can also describe the same information in the following transition table:

	ν	
	0	1
$+s_1+$	s_*	s_2
s_2	s_3	s_4
s_3	s_*	s_2
s_4	s_5	s_7
s_5	s_6	s_9
s_6	s_*	s_4
s_7	s_*	s_8
$-s_8-$	s_*	s_*
$-s_9-$	s_*	s_*
s_*	s_*	s_*

7.2. Equivalence of States and Minimization

Note that in Example 7.1.3, we can take $\nu(s_5, 1) = s_8$ and save a state s_9 . As is in the case of finite state machines, there is a reduction process based on the idea of equivalence of states.

Suppose that $A = (\mathcal{S}, \mathcal{I}, \nu, \mathcal{T}, s_1)$ is a deterministic finite state automaton.

DEFINITION. We say that two states $s_i, s_j \in \mathcal{S}$ are 0-equivalent, denoted by $s_i \cong_0 s_j$, if either both $s_i, s_j \in \mathcal{T}$ or both $s_i, s_j \notin \mathcal{T}$. For every $k \in \mathbb{N}$, we say that two states $s_i, s_j \in \mathcal{S}$ are k -equivalent, denoted by $s_i \cong_k s_j$, if $s_i \cong_0 s_j$ and for every $m = 1, \dots, k$ and every $x \in \mathcal{I}^m$, we have $\nu(s_i, x) \cong_0 \nu(s_j, x)$ (here $\nu(s_i, x) = \nu(s_i, x_1 \dots x_m)$ denotes the state of the automaton after the input string $x = x_1 \dots x_m$, starting at state s_i). Furthermore, we say that two states $s_i, s_j \in \mathcal{S}$ are equivalent, denoted by $s_i \cong s_j$, if $s_i \cong_k s_j$ for every $k \in \mathbb{N} \cup \{0\}$.

REMARK. Recall that for a finite state machine, two states $s_i, s_j \in \mathcal{S}$ are k -equivalent if

$$\omega(s_i, x_1 x_2 \dots x_k) = \omega(s_j, x_1 x_2 \dots x_k)$$

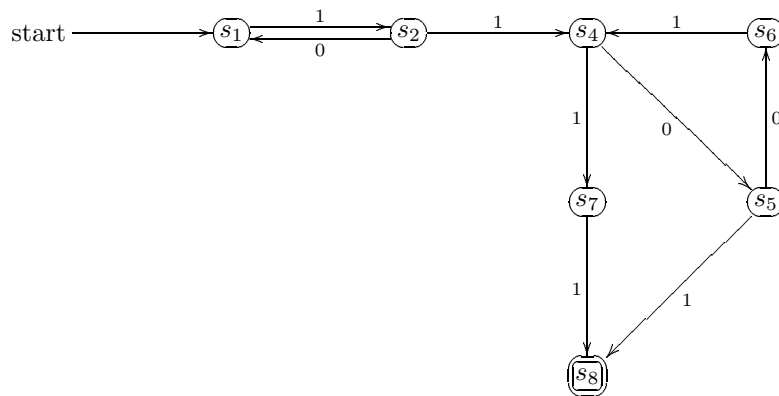
Continuing this process, we obtain the following:

	ν		\cong_3	ν		\cong_4	ν		\cong_5	ν		\cong_6
	0	1		0	1		0	1		0	1	
$+s_1+$	s_*	s_2	A	A	B	A	G	B	A	H	B	A
s_2	s_3	s_4	B	A	C	B	A	C	B	A	C	B
s_3	s_*	s_2	A	A	B	A	G	B	A	H	B	A
s_4	s_5	s_7	C	D	D	C	D	E	C	D	F	C
s_5	s_6	s_9	D	B	E	D	B	F	D	E	G	D
s_6	s_*	s_4	B	A	C	B	G	C	E	H	C	E
s_7	s_*	s_8	D	A	E	E	G	F	F	H	G	F
$-s_8-$	s_*	s_*	E	A	A	F	G	G	G	H	H	G
$-s_9-$	s_*	s_*	E	A	A	F	G	G	G	H	H	G
s_*	s_*	s_*	A	A	A	G	G	G	H	H	H	H

Choosing s_1 and s_8 and discarding s_3 and s_9 , we have the following minimized transition table:

	ν	
	0	1
$+s_1+$	s_*	s_2
s_2	s_1	s_4
s_4	s_5	s_7
s_5	s_6	s_8
s_6	s_*	s_4
s_7	s_*	s_8
$-s_8-$	s_*	s_*
s_*	s_*	s_*

Note that any reference to the removed state s_9 is now taken over by the state s_8 . We also have the following state diagram of the minimized automaton:

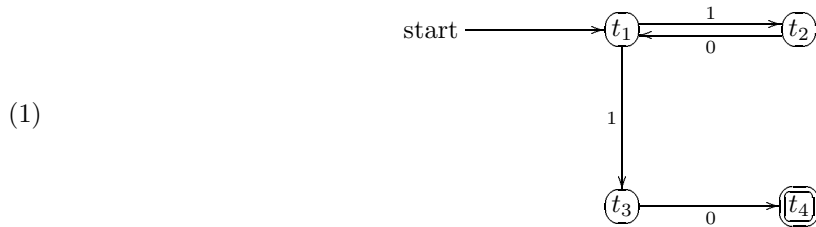


REMARK. As is in the case of finite state machines, we can further remove any state which is unreachable from the starting state s_1 if any such state exists. Indeed, such states can be removed before or after the application of the Minimization process.

7.3. Non-Deterministic Finite State Automata

To motivate our discussion in this section, we consider a very simple example.

EXAMPLE 7.3.1. Consider the following state diagram:



Suppose that at state t_1 and with input 1, the automaton has equal probability of moving to state t_2 or to state t_3 . Then with input string 10, the automaton may end up at state t_1 (via state t_2) or state t_4 (via state t_3). On the other hand, with input string 1010, the automaton may end up at state t_1 (via states t_2, t_1 and t_2), state t_4 (via states t_2, t_1 and t_3) or the dumping state t_* (via states t_3 and t_4). Note that t_4 is an accepting state while the other states are not. This is an example of a non-deterministic finite state automaton. Any string in the regular language $10(10)^*$ may send the automaton to the accepting state t_4 or to some non-accepting state. The important point is that there is a chance that the automaton may end up at an accepting state. On the other hand, this non-deterministic finite state automaton can be described by the following transition table:

	ν	
	0	1
$+t_1+$		t_2, t_3
t_2	t_1	
t_3	t_4	
$-t_4-$		

Here it is convenient not to include any reference to the dumping state t_* . Note also that we can think of $\nu(t_i, x)$ as a subset of \mathcal{S} .

Our goal in this chapter is to show that for any regular language with alphabet 0 and 1, it is possible to design a deterministic finite state automaton that will recognize precisely that language. Our technique is to do this by first constructing a non-deterministic finite state automaton and then converting it to a deterministic finite state automaton.

The reason for this approach is that non-deterministic finite state automata are easier to design; indeed, the technique involved is very systematic. On the other hand, the conversion process to deterministic finite state automata is rather easy to implement.

In this section, we shall consider non-deterministic finite state automata. In Section 7.4, we shall consider their relationship to regular languages. We shall then discuss in Section 7.5 a process which will convert non-deterministic finite state automata to deterministic finite state automata.

DEFINITION. A non-deterministic finite state automaton is a 5-tuple $A = (\mathcal{S}, \mathcal{I}, \nu, \mathcal{T}, t_1)$, where

- (a) \mathcal{S} is the finite set of states for A ;
- (b) \mathcal{I} is the finite input alphabet for A ;
- (c) $\nu : \mathcal{S} \times \mathcal{I} \rightarrow P(\mathcal{S})$ is the next-state function, where $P(\mathcal{S})$ is the collection of all subsets of \mathcal{S} ;
- (d) \mathcal{T} is a non-empty subset of \mathcal{S} ; and
- (e) $t_1 \in \mathcal{S}$ is the starting state.

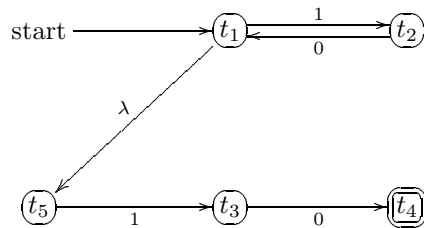
REMARKS. (1) The states in \mathcal{T} are usually called the accepting states.

(2) If not indicated otherwise, we shall always take state t_1 as the starting state.

(3) For convenience, we do not make reference to the dumping state t_* . Indeed, the conversion process in Section 7.5 will be much clearer if we do not include t_* in \mathcal{S} and simply leave entries blank in the transition table.

For practical convenience, we shall modify our approach slightly by introducing null transitions. These are useful in the early stages in the design of a non-deterministic finite state automaton and can be removed later on. To motivate this, we elaborate on our earlier example.

EXAMPLE 7.3.2. Let λ denote the null string. Then the non-deterministic finite state automaton described by the state diagram (1) can be represented by the following state diagram:



In this case, the input string 1010 can be interpreted as 10 λ 10 and so will be accepted. On the other hand, the same input string 1010 can be interpreted as 1010 and so the automaton will end up at state t_1 (via states t_2 , t_1 and t_2). However, the same input string 1010 can also be interpreted as λ 1010 and so the automaton will end up at the dumping state t_* (via states t_5 , t_3 and t_4). We have the following transition table:

	ν		
	0	1	λ
+ t_1 +		t_2	t_5
t_2	t_1		
t_3	t_4		
- t_4 -			
t_5		t_3	

REMARKS. (1) While null transitions are useful in the early stages in the design of a non-deterministic finite state automaton, we have to remove them later on.

(2) We can think of null transitions as free transitions. They can be removed provided that for every state t_i and every input $x \neq \lambda$, we take $\nu(t_i, x)$ to denote the collection of all the states that can be reached from state t_i by input x with the help of null transitions.

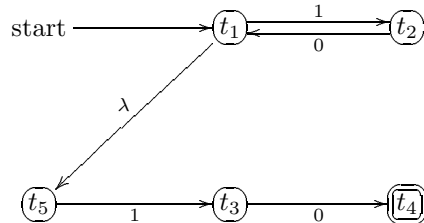
(3) If the collection $\nu(t_i, \lambda)$ contains an accepting state, then the state t_i should also be considered an accepting state, even if it is not explicitly given as such.

(4) Theoretically, $\nu(t_i, \lambda)$ contains t_i . However, this piece of useless information can be ignored in view of (2) above.

(5) Again, in view of (2) above, it is convenient not to refer to t_* or to include it in \mathcal{S} . It is very convenient to represent by a blank entry in the transition table the fact that $\nu(t_i, x)$ does not contain any state.

In practice, we may not need to use transition tables where null transitions are involved. We may design a non-deterministic finite state automaton by drawing a state diagram with null transitions. We then produce from this state diagram a transition table for the automaton without null transitions. We illustrate this technique through the use of two examples.

EXAMPLE 7.3.3. Consider the non-deterministic finite state automaton described earlier by following state diagram:



It is clear that $\nu(t_1, 0)$ is empty, while $\nu(t_1, 1)$ contains states t_2 and t_3 (the latter via state t_5 with the help of a null transition) but not states t_1, t_4 and t_5 . We therefore have the following partial transition table:

	ν	
	0	1
+t ₁ + t ₂ t ₃ -t ₄ - t ₅		t ₂ , t ₃

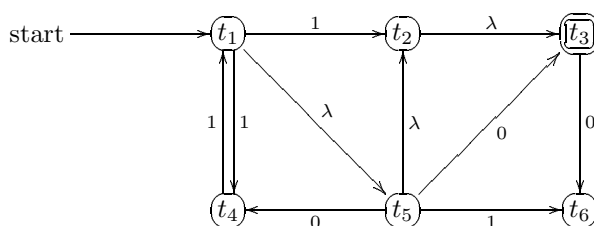
Next, it is clear that $\nu(t_2, 0)$ contains states t_1 and t_5 (the latter via state t_1 with the help of a null transition) but not states t_2, t_3 and t_4 , while $\nu(t_2, 1)$ is empty. We therefore have the following partial transition table:

	ν	
	0	1
+t ₁ + t ₂ t ₃ -t ₄ - t ₅	t ₁ , t ₅	t ₂ , t ₃

With similar arguments, we can complete the transition table as follows:

	ν	
	0	1
+t ₁ + t ₂ t ₃ -t ₄ - t ₅	t ₁ , t ₅ t ₄	t ₂ , t ₃ t ₃

EXAMPLE 7.3.4. Consider the non-deterministic finite state automaton described by following state diagram:



It is clear that $\nu(t_1, 0)$ contains states t_3 and t_4 (both via state t_5 with the help of a null transition) as well as t_6 (via states t_5, t_2 and t_3 with the help of three null transitions), but not states t_1, t_2 and t_5 . On the other hand, $\nu(t_1, 1)$ contains states t_2 and t_4 as well as t_3 (via state t_2 with the help of a null transition) and t_6 (via state t_5 with the help of a null transition), but not states t_1 and t_5 . We therefore have the following partial transition table:

	ν	
	0	1
$+t_1+$ t_2 $-t_3-$ t_4 t_5 t_6	t_3, t_4, t_6	t_2, t_3, t_4, t_6

With similar arguments, we can complete the entries of the transition table as follows:

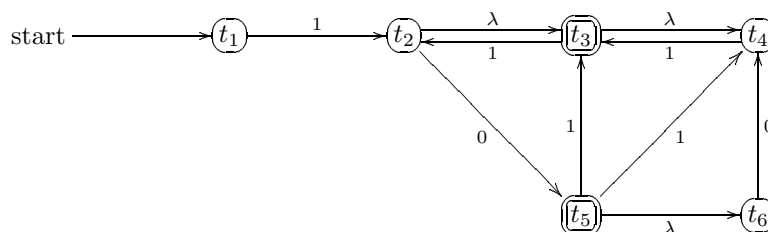
	ν	
	0	1
$+t_1+$ t_2 $-t_3-$ t_4 t_5 t_6	t_3, t_4, t_6 t_6 t_6	t_2, t_3, t_4, t_6 t_1, t_2, t_3, t_5 t_6

Finally, note that since t_3 is an accepting state, and can be reached from states t_1, t_2 and t_5 with the help of null transitions. Hence these three states should also be considered accepting states. We therefore have the following transition table:

	ν	
	0	1
$+t_1-$ $-t_2-$ $-t_3-$ t_4 $-t_5-$ t_6	t_3, t_4, t_6 t_6 t_6	t_2, t_3, t_4, t_6 t_1, t_2, t_3, t_5 t_6

It is possible to remove the null transitions in a more systematic way. We first illustrate the ideas by considering two examples.

EXAMPLE 7.3.5. Consider the non-deterministic finite state automaton described by following state diagram:



This can be represented by the following transition table with null transitions:

	ν		
	0	1	λ
$+t_1+$		t_2	
t_2	t_5		t_3
$-t_3-$		t_2	t_4
t_4		t_3	
$-t_5-$		t_3, t_4	t_6
t_6	t_4		

We shall modify this (partial) transition table step by step, removing the column of null transitions at the end.

- (1) Let us list all the null transitions described in the transition table:

$$\begin{aligned}
 t_2 &\xrightarrow{\lambda} t_3 \\
 t_3 &\xrightarrow{\lambda} t_4 \\
 t_5 &\xrightarrow{\lambda} t_6
 \end{aligned}$$

We shall refer to this list in step (2) below.

- (2) We consider attaching extra null transitions at the end of an input. For example, the inputs $0, 0\lambda, 0\lambda\lambda, \dots$ are the same. Consider now the first null transition on our list, the null transition from state t_2 to state t_3 . Clearly if we arrive at state t_2 , we can move on to state t_3 via the null transition, as illustrated below:

$$? \xrightarrow{?} t_2 \xrightarrow{\lambda} t_3$$

Consequently, whenever state t_2 is listed in the (partial) transition table, we can freely add state t_3 to it. Implementing this, we modify the (partial) transition table as follows:

	ν		
	0	1	λ
$+t_1+$		t_2, t_3	
t_2	t_5		t_3
$-t_3-$		t_2, t_3	t_4
t_4		t_3	
$-t_5-$		t_3, t_4	t_6
t_6	t_4		

Consider next the second null transition on our list, the null transition from state t_3 to state t_4 . Clearly if we arrive at state t_3 , we can move on to state t_4 via the null transition. Consequently, whenever state t_3 is listed in the (partial) transition table, we can freely add state t_4 to it. Implementing this, we modify the (partial) transition table as follows:

	ν		
	0	1	λ
$+t_1+$		t_2, t_3, t_4	
t_2	t_5		t_3, t_4
$-t_3-$		t_2, t_3, t_4	t_4
t_4		t_3, t_4	
$-t_5-$		t_3, t_4	t_6
t_6	t_4		

Consider finally the third null transition on our list, the null transition from state t_5 to state t_6 . Clearly if we arrive at state t_5 , we can move on to state t_6 via the null transition. Consequently, whenever state t_5 is listed in the (partial) transition table, we can freely add state t_6 to it. Implementing this, we modify the (partial) transition table as follows:

	0	ν 1	λ
$+t_1+$		t_2, t_3, t_4	
t_2	t_5, t_6		t_3, t_4
$-t_3-$		t_2, t_3, t_4	t_4
t_4		t_3, t_4	
$-t_5-$		t_3, t_4	t_6
t_6	t_4		

We now repeat the same process with the three null transitions on our list, and observe that there are no further modifications to the (partial) transition table. If we examine the column for null transitions, we note that it is possible to arrive at one of the accepting states t_3 or t_5 from state t_2 via null transitions. It follows that the accepting states are now states t_2, t_3 and t_5 . Implementing this, we modify the (partial) transition table as follows:

	0	ν 1	λ
$+t_1+$		t_2, t_3, t_4	
$-t_2-$	t_5, t_6		t_3, t_4
$-t_3-$		t_2, t_3, t_4	t_4
t_4		t_3, t_4	
$-t_5-$		t_3, t_4	t_6
t_6	t_4		

We may ask here why we repeat the process of going through all the null transitions on the list. We shall discuss this point in the next example.

- (3) We now update our list of null transitions in step (1) in view of extra information obtained in step (2). Using the column of null transitions in the (partial) transition table, we list all the null transitions:

$$\begin{array}{l}
 t_2 \xrightarrow{\lambda} t_3, t_4 \\
 t_3 \xrightarrow{\lambda} t_4 \\
 t_5 \xrightarrow{\lambda} t_6
 \end{array}$$

We shall refer to this list in step (4) below.

- (4) We consider attaching extra null transitions before an input. For example, the inputs $0, \lambda 0, \lambda \lambda 0, \dots$ are the same. Consider now the first null transitions on our updated list, the null transitions from state t_2 to states t_3 and t_4 . Clearly if we depart from state t_3 or t_4 , we can imagine that we have first arrived from state t_2 via a null transition, as illustrated below:

$$\begin{array}{l}
 t_2 \xrightarrow{\lambda} t_3 \xrightarrow{?} ? \\
 t_2 \xrightarrow{\lambda} t_4 \xrightarrow{?} ?
 \end{array}$$

Consequently, any destination from state t_3 or t_4 can also be considered a destination from state t_2 with the same input. It follows that we can add rows 3 and 4 to row 2. Implementing this, we

modify the (partial) transition table as follows:

	ν		
	0	1	λ
$+t_1+$		t_2, t_3, t_4	
$-t_2-$	t_5, t_6	t_2, t_3, t_4	t_3, t_4
$-t_3-$		t_2, t_3, t_4	t_4
t_4		t_3, t_4	
$-t_5-$		t_3, t_4	t_6
t_6	t_4		

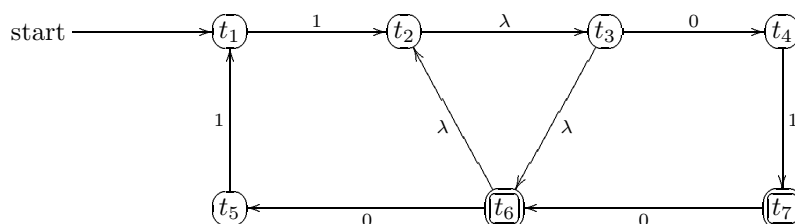
Consider next the second null transition on our updated list, the null transition from state t_3 to state t_4 . Clearly if we depart from state t_4 , we can imagine that we have first arrived from state t_3 via a null transition. Consequently, any destination from state t_4 can also be considered a destination from state t_3 with the same input. It follows that we can add row 4 to row 3. Implementing this, we realize that there is no change to the (partial) transition table. Consider finally the third null transition on our updated list, the null transition from state t_5 to state t_6 . Clearly if we depart from state t_6 , we can imagine that we have first arrived from state t_5 via a null transition. Consequently, any destination from state t_6 can also be considered a destination from state t_5 with the same input. It follows that we can add row 6 to row 5. Implementing this, we modify the (partial) transition table as follows:

	ν		
	0	1	λ
$+t_1+$		t_2, t_3, t_4	
$-t_2-$	t_5, t_6	t_2, t_3, t_4	t_3, t_4
$-t_3-$		t_2, t_3, t_4	t_4
t_4		t_3, t_4	
$-t_5-$	t_4	t_3, t_4	t_6
t_6	t_4		

(5) We can now remove the column of null transitions to obtain the transition table below:

	ν	
	0	1
$+t_1+$		t_2, t_3, t_4
$-t_2-$	t_5, t_6	t_2, t_3, t_4
$-t_3-$		t_2, t_3, t_4
t_4		t_3, t_4
$-t_5-$	t_4	t_3, t_4
t_6	t_4	

EXAMPLE 7.3.6. Consider the non-deterministic finite state automaton described by following state diagram:



This can be represented by the following transition table with null transitions:

	ν		
	0	1	λ
$+t_1+$		t_2	
t_2			t_3
t_3	t_4		t_6
t_4		t_7	
t_5		t_1	
$-t_6-$	t_5		t_2
$-t_7-$	t_6		

We shall modify this (partial) transition table step by step, removing the column of null transitions at the end.

- (1) Let us list all the null transitions described in the transition table:

$$\begin{aligned}
 t_2 &\xrightarrow{\lambda} t_3 \\
 t_3 &\xrightarrow{\lambda} t_6 \\
 t_6 &\xrightarrow{\lambda} t_2
 \end{aligned}$$

We shall refer to this list in step (2) below.

- (2) We consider attaching extra null transitions at the end of an input. In view of the first null transition from state t_2 to state t_3 , whenever state t_2 is listed in the (partial) transition table, we can freely add state t_3 to it. We now implement this. Next, in view of the second null transition from state t_3 to state t_6 , whenever state t_3 is listed in the (partial) transition table, we can freely add state t_6 to it. We now implement this. Finally, in view of the third null transition from state t_6 to state t_2 , whenever state t_6 is listed in the (partial) transition table, we can freely add state t_2 to it. We now implement this. We should obtain the following modified (partial) transition table:

	ν		
	0	1	λ
$+t_1+$		t_2, t_3, t_6	
t_2			t_2, t_3, t_6
t_3	t_4		t_2, t_6
t_4		t_7	
t_5		t_1	
$-t_6-$	t_5		t_2, t_3, t_6
$-t_7-$	t_2, t_6		

We now repeat the same process with the three null transitions on our list, and obtain the following modified (partial) transition table:

	ν		
	0	1	λ
$+t_1+$		t_2, t_3, t_6	
t_2			t_2, t_3, t_6
t_3	t_4		t_2, t_3, t_6
t_4		t_7	
t_5		t_1	
$-t_6-$	t_5		t_2, t_3, t_6
$-t_7-$	t_2, t_3, t_6		

Observe that the repetition here gives extra information like the following:

$$t_7 \xrightarrow{0} t_3$$

We see from the state diagram that this is achieved by the following:

$$t_7 \xrightarrow{0} t_6 \xrightarrow{\lambda} t_2 \xrightarrow{\lambda} t_3$$

If we do not have the repetition, then we may possibly be restricting ourselves to only one use of a null transition, and will only get as far as the following:

$$t_7 \xrightarrow{0} t_6 \xrightarrow{\lambda} t_2$$

We now repeat the same process with the three null transitions on our list one more time, and observe that there are no further modifications to the (partial) transition table. This means that we cannot attach any extra null transitions at the end. If we examine the column for null transitions, we note that it is possible to arrive at one of the accepting states t_6 or t_7 from states t_2 or t_3 via null transitions. It follows that the accepting states are now states t_2, t_3, t_6 and t_7 . Implementing this, we modify the (partial) transition table as follows:

	0	ν 1	λ
$+t_1+$		t_2, t_3, t_6	
$-t_2-$			t_2, t_3, t_6
$-t_3-$	t_4		t_2, t_3, t_6
t_4		t_7	
t_5		t_1	
$-t_6-$	t_5		t_2, t_3, t_6
$-t_7-$	t_2, t_3, t_6		

- (3) We now update our list of null transitions in step (1) in view of extra information obtained in step (2). Using the column of null transitions in the (partial) transition table, we list all the null transitions:

$$\begin{aligned}
 t_2 &\xrightarrow{\lambda} t_2, t_3, t_6 \\
 t_3 &\xrightarrow{\lambda} t_2, t_3, t_6 \\
 t_6 &\xrightarrow{\lambda} t_2, t_3, t_6
 \end{aligned}$$

We shall refer to this list in step (4) below.

- (4) We consider attaching extra null transitions before an input. In view of the first null transitions from state t_2 to states t_2, t_3 and t_6 , we can add rows 2, 3 and 6 to row 2. We now implement this. Next, in view of the second null transitions from state t_3 to states t_2, t_3 and t_6 , we can add rows 2, 3 and 6 to row 3. We now implement this. Finally, in view of the third null transitions from state t_6 to states t_2, t_3 and t_6 , we can add rows 2, 3 and 6 to row 6. We now implement this. We should obtain the following modified (partial) transition table:

	0	ν 1	λ
$+t_1+$		t_2, t_3, t_6	
$-t_2-$	t_4, t_5		t_2, t_3, t_6
$-t_3-$	t_4, t_5		t_2, t_3, t_6
t_4		t_7	
t_5		t_1	
$-t_6-$	t_4, t_5		t_2, t_3, t_6
$-t_7-$	t_2, t_3, t_6		

(5) We can now remove the column of null transitions to obtain the transition table below:

	ν	
	0	1
$+t_1+$		t_2, t_3, t_6
$-t_2-$	t_4, t_5	
$-t_3-$	t_4, t_5	
t_4		t_7
t_5		t_1
$-t_6-$	t_4, t_5	
$-t_7-$	t_2, t_3, t_6	

ALGORITHM FOR REMOVING NULL TRANSITIONS.

(1) Start with a (partial) transition table of the non-deterministic finite state automaton which includes information for every transition shown on the state diagram. Write down the list of all null transitions shown in this table.

(2) Follow the list of null transitions in step (1) one by one. For any null transition

$$t_i \xrightarrow{\lambda} t_j$$

on the list, freely add state t_j to any occurrence of state t_i in the (partial) transition table. After completing this task for the whole list of null transitions, repeat the entire process again and again, until a full repetition yields no further changes to the (partial) transition table. We then examine the column of null transitions to determine from which states it is possible to arrive at an accepting state via null transitions only. We include these extra states as accepting states.

(3) Update the list of null transitions in step (1) in view of extra information obtained in step (2). Using the column of null transitions in the (partial) transition table, we list all the null transitions originating from all states.

(4) Follow the list of null transitions in step (3) originating from each state. For any null transitions

$$t_i \xrightarrow{\lambda} t_{j_1}, \dots, t_{j_k}$$

on the list, add rows j_1, \dots, j_k to row i .

(5) Remove the column of null transitions to obtain the full transition table without null transitions.

REMARKS. (1) It is important to repeat step (2) until a full repetition yields no further modifications. Then we have analyzed the network of null transitions fully.

(2) There is no need for repetition in step (4), as we are using the full network of null transitions obtained in step (2).

(3) The network of null transitions can be analyzed by using Warshall's algorithm on directed graphs. See Section 20.1.

7.4. Regular Languages

Recall that a regular language on the alphabet \mathcal{I} is either empty or can be built up from elements of \mathcal{I} by using only concatenation and the operations $+$ (union) and $*$ (Kleene closure). It is well known that a language L with the alphabet \mathcal{I} is regular if and only if there exists a finite state automaton with inputs in \mathcal{I} that accepts precisely the strings in L .

Here we shall concentrate on the task of showing that for any regular language L on the alphabet 0 and 1, we can construct a finite state automaton that accepts precisely the strings in L . This will follow from the following result.

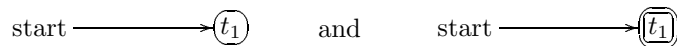
PROPOSITION 7B. Consider languages with alphabet 0 and 1.

- (a) For each of the languages $L = \emptyset, \lambda, 0, 1$, there exists a finite state automaton that accepts precisely the strings in L .
- (b) Suppose that the finite state automata A and B accept precisely the strings in the languages L and M respectively. Then there exists a finite state automaton AB that accepts precisely the strings in the language LM .
- (c) Suppose that the finite state automata A and B accept precisely the strings in the languages L and M respectively. Then there exists a finite state automaton $A + B$ that accepts precisely the strings in the language $L + M$.
- (d) Suppose that the finite state automaton A accepts precisely the strings in the language L . Then there exists a finite state automaton A^* that accepts precisely the strings in the language L^* .

Note that parts (b), (c) and (d) deal with concatenation, union and Kleene closure respectively.

For the remainder of this section, we shall use non-deterministic finite state automata with null transitions. Recall that null transitions can be removed; see Section 7.3. We shall also show in Section 7.5 how we may convert a non-deterministic finite state automaton into a deterministic one.

Part (a) is easily proved. The finite state automata



accept precisely the languages \emptyset and λ respectively. On the other hand, the finite state automata



accept precisely the languages 0 and 1 respectively.

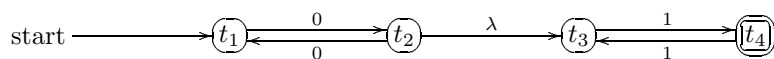
CONCATENATION. Suppose that the finite state automata A and B accept precisely the strings in the languages L and M respectively. Then the finite state automaton AB constructed as follows accepts precisely the strings in the language LM :

- (1) We label the states of A and B all differently. The states of AB are the states of A and B combined.
- (2) The starting state of AB is taken to be the starting state of A .
- (3) The accepting states of AB are taken to be the accepting states of B .
- (4) In addition to all the existing transitions in A and B , we introduce extra null transitions from each of the accepting states of A to the starting state of B .

EXAMPLE 7.4.1. The finite state automata

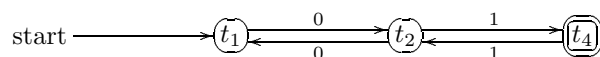


accept precisely the strings of the languages $0(00)^*$ and $1(11)^*$ respectively. The finite state automaton



accepts precisely the strings of the language $0(00)^*1(11)^*$.

REMARK. It is important to keep the two parts of the automaton AB apart by a null transition in one direction only. Suppose, instead, that we combine the accepting state t_2 of the first automaton with the starting state t_3 of the second automaton. Then the finite state automaton

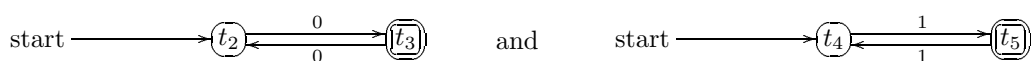


accepts the string 011001 which is not in the language $0(00)^*1(11)^*$.

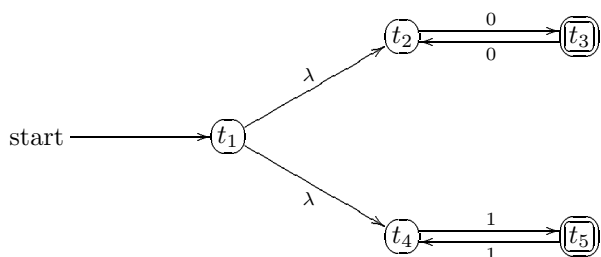
UNION. Suppose that the finite state automata A and B accept precisely the strings in the languages L and M respectively. Then the finite state automaton $A + B$ constructed as follows accepts precisely the strings in the language $L + M$:

- (1) We label the states of A and B all differently. The states of $A + B$ are the states of A and B combined plus an extra state t_1 .
- (2) The starting state of $A + B$ is taken to be the state t_1 .
- (3) The accepting states of $A + B$ are taken to be the accepting states of A and B combined.
- (4) In addition to all the existing transitions in A and B , we introduce extra null transitions from t_1 to the starting state of A and from t_1 to the starting state of B .

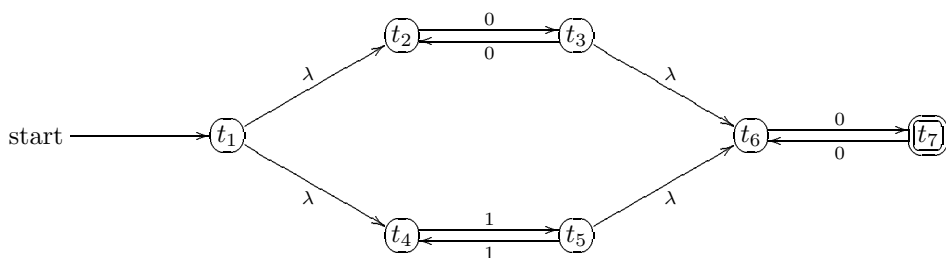
EXAMPLE 7.4.2. The finite state automata



accept precisely the strings of the languages $0(00)^*$ and $1(11)^*$ respectively. The finite state automaton



accepts precisely the strings of the language $0(00)^* + 1(11)^*$, while the finite state automaton



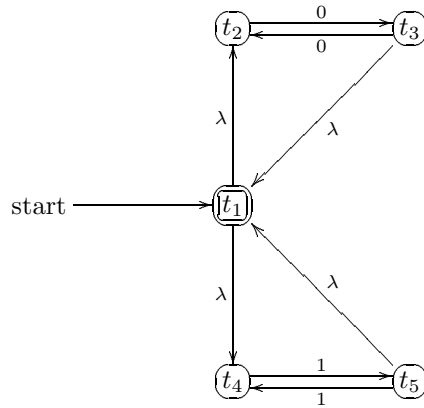
accepts precisely the strings of the language $(0(00)^* + 1(11)^*)0(00)^*$.

KLEENE CLOSURE. Suppose that the finite state automaton A accepts precisely the strings in the language L . Then the finite state automaton A^* constructed as follows accepts precisely the strings in the language L^* :

- (1) The states of A^* are the states of A .
- (2) The starting state of A^* is taken to be the starting state of A .
- (3) In addition to all the existing transitions in A , we introduce extra null transitions from each of the accepting states of A to the starting state of A .
- (4) The accepting state of A^* is taken to be the starting state of A .

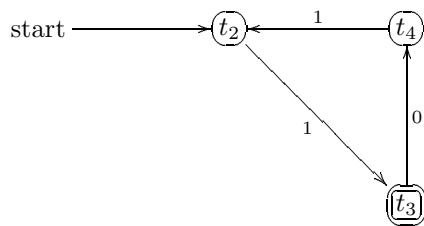
REMARK. Of course, each of the accepting states of A are also accepting states of A^* since it is possible to reach A via a null transition. However, it is not important to worry about this at this stage.

EXAMPLE 7.4.3. It follows from our last example that the finite state automaton

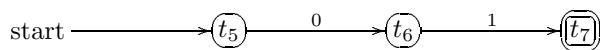


accepts precisely the strings of the language $(0(00)^* + 1(11)^*)^*$.

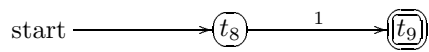
EXAMPLE 7.4.4. The finite state automaton



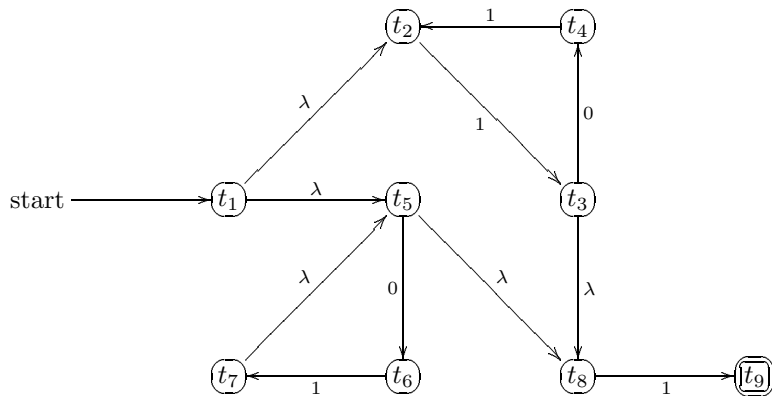
accepts precisely the strings of the language $1(011)^*$. The finite state automaton



accepts precisely the strings of the language 01 . The finite state automaton



accepts precisely the strings of the language 1 . It follows that the finite state automaton



accepts precisely the strings of the language $(1(011)^* + (01)^*)1$.

7.5. Conversion to Deterministic Finite State Automata

In this section, we describe a technique which enables us to convert a non-deterministic finite state automaton without null transitions to a deterministic finite state automaton. Recall that we have already discussed in Section 7.3 how we may remove null transitions and obtain the transition table of a non-deterministic finite state automaton. Hence our starting point in this section is such a transition table. Suppose that $A = (\mathcal{S}, \mathcal{I}, \nu, \mathcal{T}, t_1)$ is a non-deterministic finite state automaton, and that the dumping state t_* is not included in \mathcal{S} . Our idea is to consider a deterministic finite state automaton where the states are subsets of \mathcal{S} . It follows that if our non-deterministic finite state automaton has n states, then we may end up with a deterministic finite state automaton with 2^n states, where 2^n is the number of different subsets of \mathcal{S} . However, many of these states may be unreachable from the starting state, and so we have no reason to include them. We shall therefore describe an algorithm where we shall eliminate all such unreachable states in the process.

We shall illustrate the Conversion process with two examples, the first one complete with running commentary.

EXAMPLE 7.5.1. Consider the non-deterministic finite state automaton described by the following transition table:

	ν	
	0	1
+ t_1 +	t_2, t_3	t_1, t_4
t_2	t_5	t_2
- t_3 -	t_2	t_3
t_4		t_5
t_5	t_5	t_1

Here $\mathcal{S} = \{t_1, t_2, t_3, t_4, t_5\}$. We begin with a state s_1 representing the subset $\{t_1\}$ of \mathcal{S} . To calculate $\nu(s_1, 0)$, note that $\nu(t_1, 0) = \{t_2, t_3\}$. We now let s_2 denote the subset $\{t_2, t_3\}$ of \mathcal{S} , and write $\nu(s_1, 0) = s_2$. To calculate $\nu(s_1, 1)$, note that $\nu(t_1, 1) = \{t_1, t_4\}$. We now let s_3 denote the subset $\{t_1, t_4\}$ of \mathcal{S} , and write $\nu(s_1, 1) = s_3$. We have the following partial transition table:

	ν		
	0	1	
+ s_1 +	s_2	s_3	t_1
s_2			t_2, t_3
s_3			t_1, t_4

Next, note that $s_2 = \{t_2, t_3\}$. To calculate $\nu(s_2, 0)$, note that

$$\nu(t_2, 0) \cup \nu(t_3, 0) = \{t_2, t_5\}.$$

We now let s_4 denote the subset $\{t_2, t_5\}$ of \mathcal{S} , and write $\nu(s_2, 0) = s_4$. To calculate $\nu(s_2, 1)$, note that

$$\nu(t_2, 1) \cup \nu(t_3, 1) = \{t_2, t_3\} = s_2,$$

so $\nu(s_2, 1) = s_2$. We have the following partial transition table:

	ν		
	0	1	
+ s_1 +	s_2	s_3	t_1
s_2	s_4	s_2	t_2, t_3
s_3			t_1, t_4
s_4			t_2, t_5

Next, note that $s_3 = \{t_1, t_4\}$. To calculate $\nu(s_3, 0)$, note that

$$\nu(t_1, 0) \cup \nu(t_4, 0) = \{t_2, t_3\} = s_2,$$

so $\nu(s_3, 0) = s_2$. To calculate $\nu(s_3, 1)$, note that

$$\nu(t_1, 1) \cup \nu(t_4, 1) = \{t_1, t_4, t_5\}.$$

We now let s_5 denote the subset $\{t_1, t_4, t_5\}$ of \mathcal{S} , and write $\nu(s_3, 1) = s_5$. We have the following partial transition table:

	ν		
	0	1	
$+s_1+$	s_2	s_3	t_1
s_2	s_4	s_2	t_2, t_3
s_3	s_2	s_5	t_1, t_4
s_4			t_2, t_5
s_5			t_1, t_4, t_5

Next, note that $s_4 = \{t_2, t_5\}$. To calculate $\nu(s_4, 0)$, note that

$$\nu(t_2, 0) \cup \nu(t_5, 0) = \{t_5\}.$$

We now let s_6 denote the subset $\{t_5\}$ of \mathcal{S} , and write $\nu(s_4, 0) = s_6$. To calculate $\nu(s_4, 1)$, note that

$$\nu(t_2, 1) \cup \nu(t_5, 1) = \{t_1, t_2\}.$$

We now let s_7 denote the subset $\{t_1, t_2\}$ of \mathcal{S} , and write $\nu(s_4, 1) = s_7$. We have the following partial transition table:

	ν		
	0	1	
$+s_1+$	s_2	s_3	t_1
s_2	s_4	s_2	t_2, t_3
s_3	s_2	s_5	t_1, t_4
s_4	s_6	s_7	t_2, t_5
s_5			t_1, t_4, t_5
s_6			t_5
s_7			t_1, t_2

Next, note that $s_5 = \{t_1, t_4, t_5\}$. To calculate $\nu(s_5, 0)$, note that

$$\nu(t_1, 0) \cup \nu(t_4, 0) \cup \nu(t_5, 0) = \{t_2, t_3, t_5\}.$$

We now let s_8 denote the subset $\{t_2, t_3, t_5\}$ of \mathcal{S} , and write $\nu(s_5, 0) = s_8$. To calculate $\nu(s_5, 1)$, note that

$$\nu(t_1, 1) \cup \nu(t_4, 1) \cup \nu(t_5, 1) = \{t_1, t_4, t_5\} = s_5,$$

so $\nu(s_5, 1) = s_5$. We have the following partial transition table:

	ν		
	0	1	
$+s_1+$	s_2	s_3	t_1
s_2	s_4	s_2	t_2, t_3
s_3	s_2	s_5	t_1, t_4
s_4	s_6	s_7	t_2, t_5
s_5	s_8	s_5	t_1, t_4, t_5
s_6			t_5
s_7			t_1, t_2
s_8			t_2, t_3, t_5

Next, note that $s_6 = \{t_5\}$. To calculate $\nu(s_6, 0)$, note that

$$\nu(t_5, 0) = \{t_5\} = s_6,$$

so $\nu(s_6, 0) = s_6$. To calculate $\nu(s_6, 1)$, note that

$$\nu(t_5, 1) = \{t_1\} = s_1,$$

so $\nu(s_6, 1) = s_1$. We have the following partial transition table:

	ν		
	0	1	
$+s_1+$	s_2	s_3	t_1
s_2	s_4	s_2	t_2, t_3
s_3	s_2	s_5	t_1, t_4
s_4	s_6	s_7	t_2, t_5
s_5	s_8	s_5	t_1, t_4, t_5
s_6	s_6	s_1	t_5
s_7			t_1, t_2
s_8			t_2, t_3, t_5

Next, note that $s_7 = \{t_1, t_2\}$. To calculate $\nu(s_7, 0)$, note that

$$\nu(t_1, 0) \cup \nu(t_2, 0) = \{t_2, t_3, t_5\} = s_8,$$

so $\nu(s_7, 0) = s_8$. To calculate $\nu(s_7, 1)$, note that

$$\nu(t_1, 1) \cup \nu(t_2, 1) = \{t_1, t_2, t_4\}.$$

We now let s_9 denote the subset $\{t_1, t_2, t_4\}$ of \mathcal{S} , and write $\nu(s_7, 1) = s_9$. We have the following partial transition table:

	ν		
	0	1	
$+s_1+$	s_2	s_3	t_1
s_2	s_4	s_2	t_2, t_3
s_3	s_2	s_5	t_1, t_4
s_4	s_6	s_7	t_2, t_5
s_5	s_8	s_5	t_1, t_4, t_5
s_6	s_6	s_1	t_5
s_7	s_8	s_9	t_1, t_2
s_8			t_2, t_3, t_5
s_9			t_1, t_2, t_4

Next, note that $s_8 = \{t_2, t_3, t_5\}$. To calculate $\nu(s_8, 0)$, note that

$$\nu(t_2, 0) \cup \nu(t_3, 0) \cup \nu(t_5, 0) = \{t_2, t_5\} = s_4,$$

so $\nu(s_8, 0) = s_4$. To calculate $\nu(s_8, 1)$, note that

$$\nu(t_2, 1) \cup \nu(t_3, 1) \cup \nu(t_5, 1) = \{t_1, t_2, t_3\}.$$

We now let s_{10} denote the subset $\{t_1, t_2, t_3\}$ of \mathcal{S} , and write $\nu(s_8, 1) = s_{10}$. We have the following partial transition table:

	ν		
	0	1	
$+s_1+$	s_2	s_3	t_1
s_2	s_4	s_2	t_2, t_3
s_3	s_2	s_5	t_1, t_4
s_4	s_6	s_7	t_2, t_5
s_5	s_8	s_5	t_1, t_4, t_5
s_6	s_6	s_1	t_5
s_7	s_8	s_9	t_1, t_2
s_8	s_4	s_{10}	t_2, t_3, t_5
s_9			t_1, t_2, t_4
s_{10}			t_1, t_2, t_3

Next, note that $s_9 = \{t_1, t_2, t_4\}$. To calculate $\nu(s_9, 0)$, note that

$$\nu(t_1, 0) \cup \nu(t_2, 0) \cup \nu(t_4, 0) = \{t_2, t_3, t_5\} = s_8,$$

so $\nu(s_9, 0) = s_8$. To calculate $\nu(s_9, 1)$, note that

$$\nu(t_1, 1) \cup \nu(t_2, 1) \cup \nu(t_4, 1) = \{t_1, t_2, t_4, t_5\}.$$

We now let s_{11} denote the subset $\{t_1, t_2, t_4, t_5\}$ of \mathcal{S} , and write $\nu(s_9, 1) = s_{11}$. We have the following partial transition table:

	ν		
	0	1	
$+s_1+$	s_2	s_3	t_1
s_2	s_4	s_2	t_2, t_3
s_3	s_2	s_5	t_1, t_4
s_4	s_6	s_7	t_2, t_5
s_5	s_8	s_5	t_1, t_4, t_5
s_6	s_6	s_1	t_5
s_7	s_8	s_9	t_1, t_2
s_8	s_4	s_{10}	t_2, t_3, t_5
s_9	s_8	s_{11}	t_1, t_2, t_4
s_{10}			t_1, t_2, t_3
s_{11}			t_1, t_2, t_4, t_5

Next, note that $s_{10} = \{t_1, t_2, t_3\}$. To calculate $\nu(s_{10}, 0)$, note that

$$\nu(t_1, 0) \cup \nu(t_2, 0) \cup \nu(t_3, 0) = \{t_2, t_3, t_5\} = s_8,$$

so $\nu(s_{10}, 0) = s_8$. To calculate $\nu(s_{10}, 1)$, note that

$$\nu(t_1, 1) \cup \nu(t_2, 1) \cup \nu(t_3, 1) = \{t_1, t_2, t_3, t_4\}.$$

We now let s_{12} denote the subset $\{t_1, t_2, t_3, t_4\}$ of \mathcal{S} , and write $\nu(s_{10}, 1) = s_{12}$. We have the following

partial transition table:

	ν		
	0	1	
$+s_1+$	s_2	s_3	t_1
s_2	s_4	s_2	t_2, t_3
s_3	s_2	s_5	t_1, t_4
s_4	s_6	s_7	t_2, t_5
s_5	s_8	s_5	t_1, t_4, t_5
s_6	s_6	s_1	t_5
s_7	s_8	s_9	t_1, t_2
s_8	s_4	s_{10}	t_2, t_3, t_5
s_9	s_8	s_{11}	t_1, t_2, t_4
s_{10}	s_8	s_{12}	t_1, t_2, t_3
s_{11}			t_1, t_2, t_4, t_5
s_{12}			t_1, t_2, t_3, t_4

Next, note that $s_{11} = \{t_1, t_2, t_4, t_5\}$. To calculate $\nu(s_{11}, 0)$, note that

$$\nu(t_1, 0) \cup \nu(t_2, 0) \cup \nu(t_4, 0) \cup \nu(t_5, 0) = \{t_2, t_3, t_5\} = s_8,$$

so $\nu(s_{11}, 0) = s_8$. To calculate $\nu(s_{11}, 1)$, note that

$$\nu(t_1, 1) \cup \nu(t_2, 1) \cup \nu(t_4, 1) \cup \nu(t_5, 1) = \{t_1, t_2, t_4, t_5\} = s_{11},$$

so $\nu(s_{11}, 1) = s_{11}$. We have the following partial transition table:

	ν		
	0	1	
$+s_1+$	s_2	s_3	t_1
s_2	s_4	s_2	t_2, t_3
s_3	s_2	s_5	t_1, t_4
s_4	s_6	s_7	t_2, t_5
s_5	s_8	s_5	t_1, t_4, t_5
s_6	s_6	s_1	t_5
s_7	s_8	s_9	t_1, t_2
s_8	s_4	s_{10}	t_2, t_3, t_5
s_9	s_8	s_{11}	t_1, t_2, t_4
s_{10}	s_8	s_{12}	t_1, t_2, t_3
s_{11}	s_8	s_{11}	t_1, t_2, t_4, t_5
s_{12}			t_1, t_2, t_3, t_4

Next, note that $s_{12} = \{t_1, t_2, t_3, t_4\}$. To calculate $\nu(s_{12}, 0)$, note that

$$\nu(t_1, 0) \cup \nu(t_2, 0) \cup \nu(t_3, 0) \cup \nu(t_4, 0) = \{t_2, t_3, t_5\} = s_8,$$

so $\nu(s_{12}, 0) = s_8$. To calculate $\nu(s_{12}, 1)$, note that

$$\nu(t_1, 1) \cup \nu(t_2, 1) \cup \nu(t_3, 1) \cup \nu(t_4, 1) = \{t_1, t_2, t_3, t_4, t_5\}.$$

We now let s_{13} denote the subset $\{t_1, t_2, t_3, t_4, t_5\}$ of \mathcal{S} , and write $\nu(s_{12}, 1) = s_{13}$. We have the following

partial transition table:

	ν		
	0	1	
$+s_1+$	s_2	s_3	t_1
s_2	s_4	s_2	t_2, t_3
s_3	s_2	s_5	t_1, t_4
s_4	s_6	s_7	t_2, t_5
s_5	s_8	s_5	t_1, t_4, t_5
s_6	s_6	s_1	t_5
s_7	s_8	s_9	t_1, t_2
s_8	s_4	s_{10}	t_2, t_3, t_5
s_9	s_8	s_{11}	t_1, t_2, t_4
s_{10}	s_8	s_{12}	t_1, t_2, t_3
s_{11}	s_8	s_{11}	t_1, t_2, t_4, t_5
s_{12}	s_8	s_{13}	t_1, t_2, t_3, t_4
s_{13}			t_1, t_2, t_3, t_4, t_5

Next, note that $s_{13} = \{t_1, t_2, t_3, t_4, t_5\}$. To calculate $\nu(s_{13}, 0)$, note that

$$\nu(t_1, 0) \cup \nu(t_2, 0) \cup \nu(t_3, 0) \cup \nu(t_4, 0) \cup \nu(t_5, 0) = \{t_2, t_3, t_5\} = s_8,$$

so $\nu(s_{13}, 0) = s_8$. To calculate $\nu(s_{13}, 1)$, note that

$$\nu(t_1, 1) \cup \nu(t_2, 1) \cup \nu(t_3, 1) \cup \nu(t_4, 1) \cup \nu(t_5, 1) = \{t_1, t_2, t_3, t_4, t_5\} = s_{13},$$

so $\nu(s_{13}, 1) = s_{13}$. We have the following partial transition table:

	ν	
	0	1
$+s_1+$	s_2	s_3
$-s_2-$	s_4	s_2
s_3	s_2	s_5
s_4	s_6	s_7
s_5	s_8	s_5
s_6	s_6	s_1
s_7	s_8	s_9
$-s_8-$	s_4	s_{10}
s_9	s_8	s_{11}
$-s_{10}-$	s_8	s_{12}
s_{11}	s_8	s_{11}
$-s_{12}-$	s_8	s_{13}
$-s_{13}-$	s_8	s_{13}

This completes the entries of the transition table. In this final transition table, we have also indicated all the accepting states. Note that t_3 is the accepting state in the original non-deterministic finite state automaton, and that it is contained in states s_2, s_8, s_{10}, s_{12} and s_{13} . These are the accepting states of the deterministic finite state automaton.

The confident reader, out of boredom, may have read only part of the last example. However, the next example must be studied carefully.

EXAMPLE 7.5.2. Consider the non-deterministic finite state automaton described by the following transition table:

	ν	
	0	1
$+t_1+$	t_3	t_2
$-t_2-$		t_1, t_2
t_3		t_1, t_2
t_4	t_4	

Here $\mathcal{S} = \{t_1, t_2, t_3, t_4\}$. We begin with a state s_1 representing the subset $\{t_1\}$ of \mathcal{S} . The reader should check that we should arrive at the following partial transition table:

	ν		
	0	1	
$+s_1+$	s_2	s_3	t_1
s_2			t_3
s_3			t_2

Next, note that $s_2 = \{t_3\}$. To calculate $\nu(s_2, 0)$, we note that $\nu(t_3, 0) = \emptyset$. Hence we write $\nu(s_2, 0) = s_*$, the dumping state. This dumping state s_* has transitions $\nu(s_*, 0) = \nu(s_*, 1) = s_*$. To calculate $\nu(s_2, 1)$, we note that $\nu(t_3, 1) = \{t_1, t_2\}$. We now let s_4 denote the subset $\{t_1, t_2\}$ of \mathcal{S} , and write $\nu(s_2, 1) = s_4$. We have the following partial transition table:

	ν		
	0	1	
$+s_1+$	s_2	s_3	t_1
s_2	s_*	s_4	t_3
s_3			t_2
s_4			t_1, t_2
s_*	s_*	s_*	\emptyset

The reader should try to complete the entries of the transition table:

	ν		
	0	1	
$+s_1+$	s_2	s_3	t_1
s_2	s_*	s_4	t_3
s_3	s_*	s_*	t_2
s_4	s_2	s_3	t_1, t_2
s_*	s_*	s_*	\emptyset

On the other hand, note that t_2 is the accepting state in the original non-deterministic finite state automaton, and that it is contained in states s_3 and s_4 . These are the accepting states of the deterministic finite state automaton. Inserting this information and deleting the right hand column gives the following complete transition table:

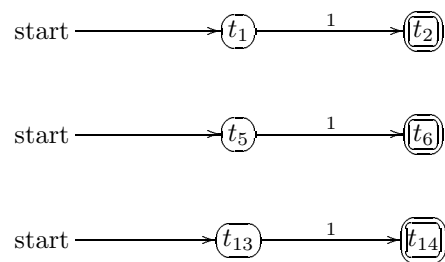
	ν	
	0	1
$+s_1+$	s_2	s_3
s_2	s_*	s_4
$-s_3-$	s_*	s_*
$-s_4-$	s_2	s_3
s_*	s_*	s_*

7.6. A Complete Example

In this last section, we shall design from scratch the deterministic finite state automaton which will accept precisely the strings in the language $1(01)^*1(001)^*(0+1)1$. Recall that this has been discussed in Examples 7.1.3 and 7.2.1. The reader is advised that it is extremely important to fill in all the missing details in the discussion here.

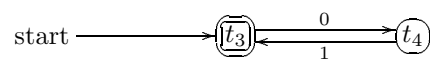
We start with non-deterministic finite state automata with null transitions, and break the language into six parts: 1, $(01)^*$, 1, $(001)^*$, $(0+1)$ and 1. These six parts are joined together by concatenation.

Clearly the three automata

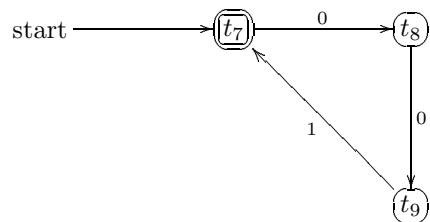


are the same and accept precisely the strings of the language 1.

On the other hand, the automaton

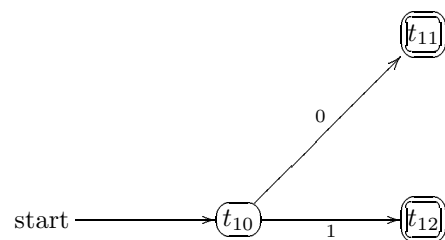


accepts precisely the strings of the language $(01)^*$, and the automaton



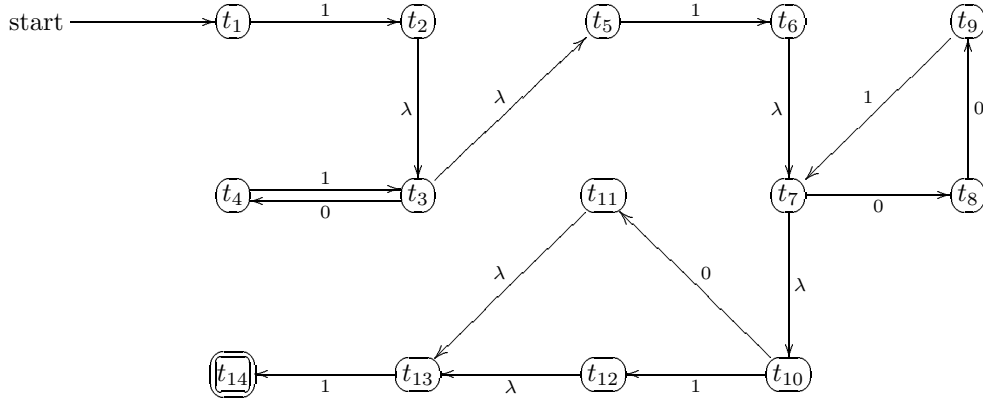
accepts precisely the strings of the language $(001)^*$. Note that we have slightly simplified the construction here by removing null transitions.

Finally, the automaton



accepts precisely the strings of the language $(0+1)$. Again, we have slightly simplified the construction here by removing null transitions.

Combining the six parts, we obtain the following state diagram of a non-deterministic finite state automaton with null transitions that accepts precisely the strings of the language $1(01)^*1(001)^*(0+1)1$:



Removing null transitions, we obtain the following transition table of a non-deterministic finite state automaton without null transitions that accepts precisely the strings of $1(01)^*1(001)^*(0+1)1$:

	ν	
	0	1
t_1		t_2, t_3, t_5
t_2	t_4	t_6, t_7, t_{10}
t_3	t_4	t_6, t_7, t_{10}
t_4		t_3, t_5
t_5		t_6, t_7, t_{10}
t_6	t_8, t_{11}, t_{13}	t_{12}, t_{13}
t_7	t_8, t_{11}, t_{13}	t_{12}, t_{13}
t_8	t_9	
t_9		t_7, t_{10}
t_{10}	t_{11}, t_{13}	t_{12}, t_{13}
t_{11}		t_{14}
t_{12}		t_{14}
t_{13}		t_{14}
t_{14}		

Applying the conversion process, we obtain the following deterministic finite state automaton corresponding to the original non-deterministic finite state automaton:

	ν		
	0	1	
$+s_1+$	s_*	s_2	t_1
s_2	s_3	s_4	t_2, t_3, t_5
s_3	s_*	s_5	t_4
s_4	s_6	s_7	t_6, t_7, t_{10}
s_5	s_3	s_4	t_3, t_5
s_6	s_8	s_9	t_8, t_{11}, t_{13}
s_7	s_*	s_9	t_{12}, t_{13}
s_8	s_*	s_{10}	t_9
$-s_9-$	s_*	s_*	t_{14}
s_{10}	s_6	s_7	t_7, t_{10}
s_*	s_*	s_*	\emptyset

Removing the last column and applying the Minimization process, we have the following table which represents the first few steps:

	ν		\cong_0	ν		\cong_1	ν		\cong_2	ν		\cong_3
	0	1		0	1		0	1		0	1	
$+s_1+$	s_*	s_2	A	A	A	A	A	A	A	A	A	
s_2	s_3	s_4	A	A	A	A	A	A	A	B	B	
s_3	s_*	s_5	A	A	A	A	A	A	A	A	A	
s_4	s_6	s_7	A	A	A	A	B	B	B	C	C	
s_5	s_3	s_4	A	A	A	A	A	A	A	A	B	
s_6	s_8	s_9	A	A	B	B	A	C	C	A	D	
s_7	s_*	s_9	A	A	B	B	A	C	C	A	D	
s_8	s_*	s_{10}	A	A	A	A	A	A	A	A	B	
$-s_9-$	s_*	s_*	B	A	A	C	A	A	D	A	A	
s_{10}	s_6	s_7	A	A	A	A	B	B	B	C	C	
s_*	s_*	s_*	A	A	A	A	A	A	A	A	A	

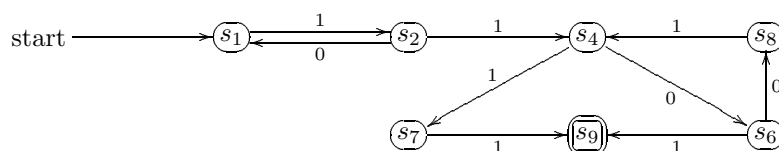
Unfortunately, the process here is going to be rather long, but let us continue nevertheless. Continuing this process, we obtain the following:

	ν		\cong_3	ν		\cong_4	ν		\cong_5	ν		\cong_6
	0	1		0	1		0	1		0	1	
$+s_1+$	s_*	s_2	A	A	B	A	G	B	A	H	B	A
s_2	s_3	s_4	B	A	C	B	A	C	B	A	C	B
s_3	s_*	s_5	A	A	B	A	G	B	A	H	B	A
s_4	s_6	s_7	C	D	D	C	D	E	C	D	E	C
s_5	s_3	s_4	B	A	C	B	A	C	B	A	C	B
s_6	s_8	s_9	D	B	E	D	B	F	D	F	G	D
s_7	s_*	s_9	D	A	E	E	G	F	E	H	G	E
s_8	s_*	s_{10}	B	A	C	B	G	C	F	H	C	F
$-s_9-$	s_*	s_*	E	A	A	F	G	G	G	H	H	G
s_{10}	s_6	s_7	C	D	D	C	D	E	C	D	E	C
s_*	s_*	s_*	A	A	A	G	G	G	H	H	H	H

Choosing s_1 , s_2 and s_4 and discarding s_3 , s_5 and s_{10} , we have the following minimized transition table:

	ν	
	0	1
$+s_1+$	s_*	s_2
s_2	s_1	s_4
s_4	s_6	s_7
s_6	s_8	s_9
s_7	s_*	s_9
s_8	s_*	s_4
$-s_9-$	s_*	s_*
s_*	s_*	s_*

This can be described by the following state diagram:



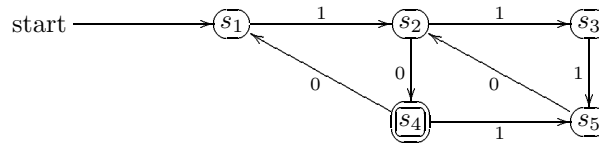
PROBLEMS FOR CHAPTER 7

1. Suppose that $\mathcal{I} = \{0, 1\}$. Consider the following deterministic finite state automaton:

	ν	
	0	1
$+s_1+$	s_2	s_3
$-s_2-$	s_3	s_2
s_3	s_1	s_3

Decide which of the following strings are accepted:

- a) 101 b) 10101 c) 11100 d) 0100010
 e) 10101111 f) 011010111 g) 00011000011 h) 1100111010011
2. Consider the deterministic finite state automaton $A = (S, \mathcal{I}, \nu, \mathcal{T}, s_1)$, where $\mathcal{I} = \{0, 1\}$, described by the following state diagram:



- a) How many states does the automaton A have?
 b) Construct the transition table for this automaton.
 c) Apply the Minimization process to this automaton and show that no state can be removed.
3. Suppose that $\mathcal{I} = \{0, 1\}$. Consider the following deterministic finite state automaton:

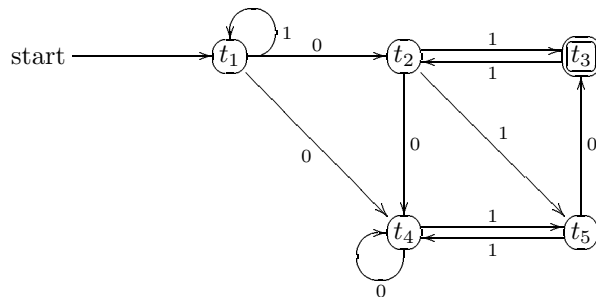
	ν	
	0	1
$+s_1-$	s_2	s_5
$-s_2-$	s_5	s_3
$-s_3-$	s_5	s_2
$-s_4-$	s_4	s_6
s_5	s_3	s_5
s_6	s_1	s_4

- a) Draw a state diagram for the automaton.
 b) Apply the Minimization process to the automaton.
4. Suppose that $\mathcal{I} = \{0, 1\}$. Consider the following deterministic finite state automaton:

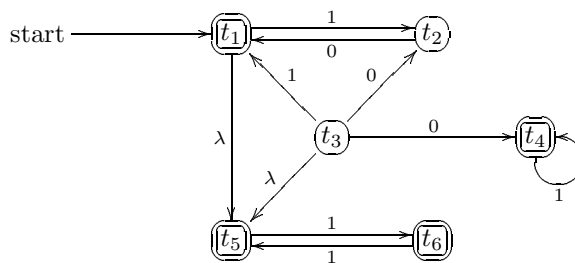
	ν	
	0	1
$+s_1+$	s_2	s_3
s_2	s_*	s_4
s_3	s_*	s_6
s_4	s_7	s_6
$-s_5-$	s_7	s_*
s_6	s_5	s_4
$-s_7-$	s_5	s_*
s_*	s_*	s_*

- a) Draw a state diagram for the automaton.
 b) Apply the Minimization process to the automaton.

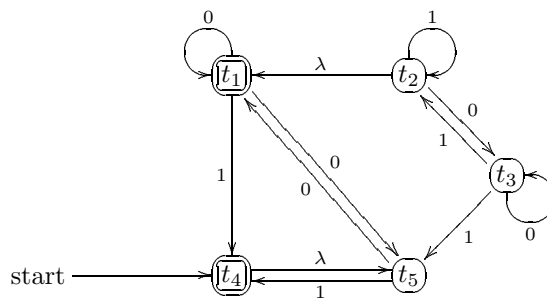
5. Let $\mathcal{I} = \{0, 1\}$.
 - a) Design a finite state automaton to accept all strings which contain exactly two 1's and which start and finish with the same symbol.
 - b) Design a finite state automaton to accept all strings where all the 0's precede all the 1's.
 - c) Design a finite state automaton to accept all strings of length at least 1 and where the sum of the first digit and the length of the string is even.
6. Consider the following non-deterministic finite state automaton $A = (\mathcal{S}, \mathcal{I}, \nu, \mathcal{T}, t_1)$ without null transitions, where $\mathcal{I} = \{0, 1\}$:



- a) Describe the automaton by a transition table.
 - b) Convert to a deterministic finite state automaton.
 - c) Minimize the number of states.
7. Consider the following non-deterministic finite state automaton $A = (\mathcal{S}, \mathcal{I}, \nu, \mathcal{T}, t_1)$ with null transitions, where $\mathcal{I} = \{0, 1\}$:

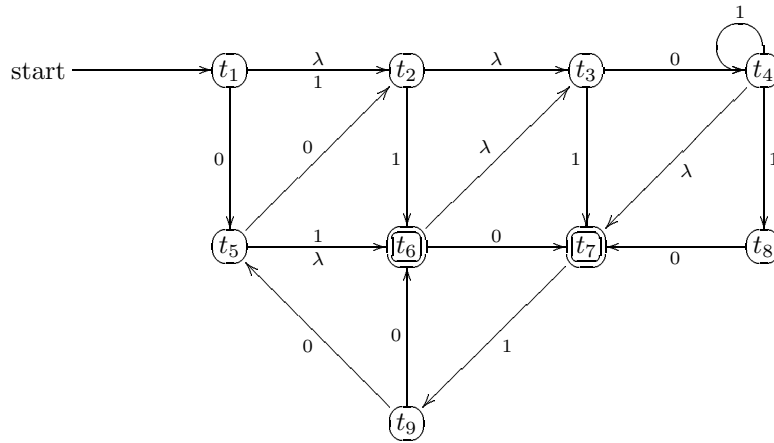


- a) Remove the null transitions, and describe the resulting automaton by a transition table.
 - b) Convert to a deterministic finite state automaton.
 - c) Minimize the number of states.
8. Consider the following non-deterministic finite state automaton $A = (\mathcal{S}, \mathcal{I}, \nu, \mathcal{T}, t_4)$ with null transitions, where $\mathcal{I} = \{0, 1\}$:

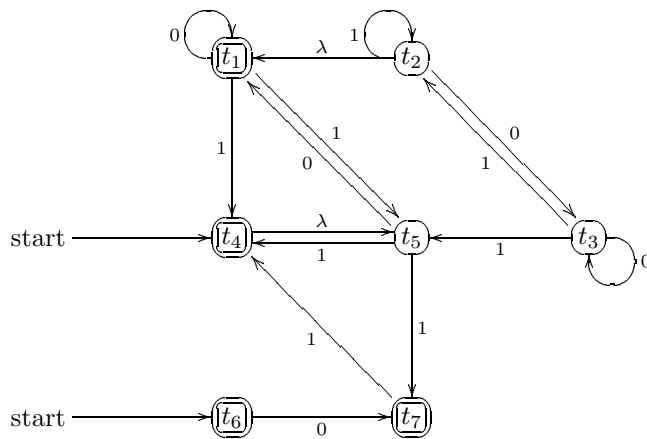


- a) Remove the null transitions, and describe the resulting automaton by a transition table.
- b) Convert to a deterministic finite state automaton.
- c) Minimize the number of states.

9. Consider the following non-deterministic finite state automaton $A = (S, \mathcal{I}, \nu, \mathcal{T}, t_1)$ with null transitions, where $\mathcal{I} = \{0, 1\}$:



- a) Remove the null transitions, and describe the resulting automaton by a transition table.
 - b) Convert to a deterministic finite state automaton.
 - c) Minimize the number of states.
10. Consider the following non-deterministic finite state automaton $A = (S, \mathcal{I}, \nu, \mathcal{T}, t_4, t_6)$ with null transitions, where $\mathcal{I} = \{0, 1\}$:



- a) Remove the null transitions, and describe the resulting automaton by a transition table.
 - b) Convert to a deterministic finite state automaton.
 - c) How many states does the deterministic finite state automaton have on minimization?
11. Prove that the set of all binary strings in which there are exactly as many 0's as 1's is not a regular language.
12. For each of the following regular languages with alphabet 0 and 1, design a non-deterministic finite state automaton with null transitions that accepts precisely all the strings of the language. Remove the null transitions and describe your result in a transition table. Convert it to a deterministic finite state automaton, apply the Minimization process and describe your result in a state diagram.
- a) $(01)^*(0 + 1)(011 + 10)^*$
 - b) $(100 + 01)(011 + 1)^*(100 + 01)^*$
 - c) $(\lambda + 01)1(010 + (01)^*)01^*$