

Converting a λ -NDFFA to a DFA

Handout written by David Underhill.

This document outlines a three-step process for converting an λ -NDFFA to a DFA. Another approach called *subset construction* is presented in the Dragon book in section 3.7.1.

Step 1: Removing λ -Cycles from an NDFFA

A λ -cycle occurs whenever there is a set of two or more states in which you can get from any state in the set to any other state in the set without reading any input. A simple example of an λ -NDFFA which contains a λ -cycle can be seen in Figure 1. The states q_1 , q_2 , and q_3 are part of the λ -cycle.

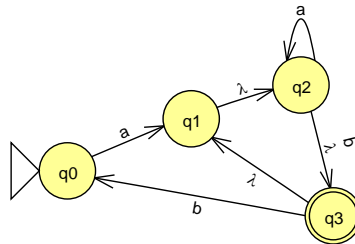


Figure 1: A simple λ -NDFFA with a λ -cycle.

Now that we have identified a cycle, we would like to remove it. The states in a λ -cycle are really like one big state, so we can remove by simply combining all of its states into a single state which we will designate Q_{big} for this example. If any of the original states was a start state, then Q_{big} is also a start state. Likewise, if any of the original states was an accept state, then Q_{big} is an accept state. For example, when removing the λ -cycle from the λ -NDFFA in Figure 1, we would combine q_1 , q_2 , and q_3 into Q_{big} .

Now that we've combined λ -cycle's states into Q_{big} , we need to specify the transitions for Q_{big} . A process for doing so follows:

1. Any transition which was *out of* one of the original states is now out of Q_{big} . For example, the transition on b from q_3 to q_0 would now be from Q_{big} instead.
2. Any transition which was *into* one of the original states is now into Q_{big} . For example, the transition from q_0 to q_1 now goes to Q_{big} .

These two rules also cover transitions which were out of and into one of the original states. For example, the transition from q_2 to q_3 on a b is now from and to Q_{big} .

Once we apply this process to the λ -cycle from Figure 1, we end up with the λ -NDFFA shown in Figure 2. In this case, there are no more λ -cycles, though if there were we would just repeatedly apply this process until each one had been eliminated.

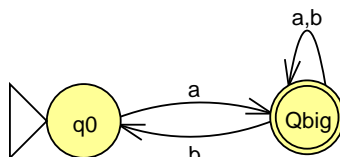


Figure 2: The λ -cycle has been removed from the λ -NDFFA in Figure 1.

Step 2: Removing λ -Transitions from an NDFFA without λ -Cycles

A λ -transition essentially allows the automaton to behave as if it is in multiple states at once. It follows from this that you can remove λ -transitions from some state q by essentially copying the transitions from each successor state which can be reached from q via a λ -transition. The process described here only works if the successor state has no λ -transitions out of it. This means the algorithm presented in this step is incapable of removing λ -cycles - hence the reason for the previous step!

To clarify what copying transitions from a successor state entails, let's work through removing the λ -transitions from Figure 3.

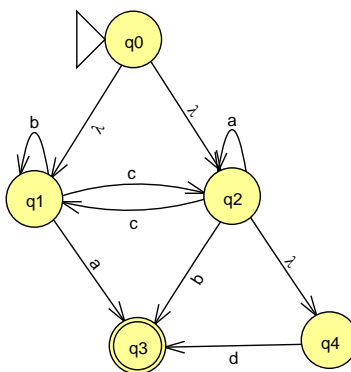


Figure 3: This is a simple λ -NDFFA with λ -transitions (but not λ -cycles).

First consider the λ -transition from q_0 to q_1 . q_1 has no λ -transitions out of it, so we can copy its transitions to q_0 and then eliminate the λ -transition from q_0 to q_1 and still end up with an equivalent machine. Let's walk through copying each of the transitions from q_1 to be from q_0 . First, we notice that q_1 transitions to q_3 on a . So we add a transition from q_0 to q_3 on a . We finish eliminating this λ -transition by adding a transition from q_0 to q_1 on b , and from q_0 to q_2 on c . The resulting machine is shown in Figure 4.

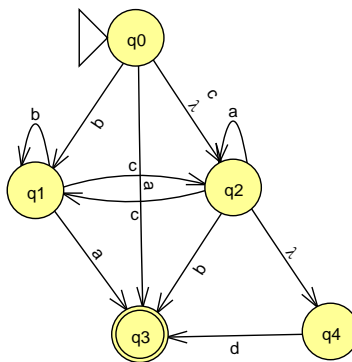


Figure 4: This the λ -NDFFA from Figure 3 with the λ -transition from q_0 to q_1 eliminated.

Now consider the λ -transition from q_0 to q_2 . Notice that q_2 has a λ -transition to q_4 - therefore we cannot remove this λ -transition from q_0 to q_2 just yet.

Instead, lets remove the λ -transition from q_2 to q_4 . As usual we copy the transitions. In particular, we add a transition from q_2 to q_3 on d . In this case, when we remove the λ -transition, q_4 is orphaned and has no transitions into it which means we can simply delete it. The resulting machine is shown in Figure 5.

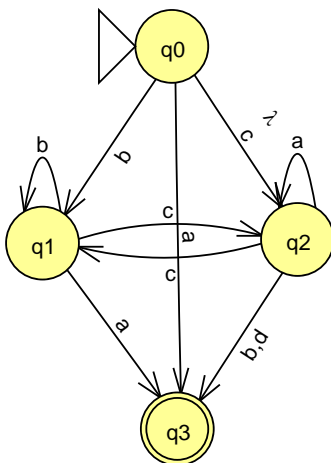


Figure 5: This the λ -NDFFA from Figure 4 with the λ -transition from q_2 to q_4 eliminated.

Finally, we can remove the λ -transition from q_0 to q_2 . The transitions from q_2 on a , b , and c can be handled in the same way we handled the transitions from q_1 earlier. The resulting machine is shown in Figure 6.

Now that all the λ -transitions have been removed, we can move on to the final step in the conversion process!

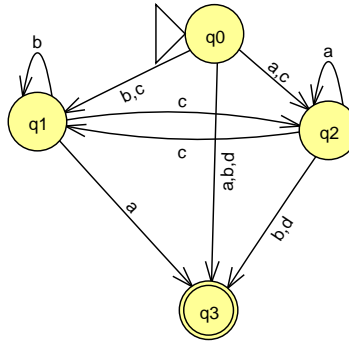


Figure 6: This the λ -NFA from Figure 5 with the λ -transition from q_0 to q_2 eliminated.

Step 3: Removing Non-Determinism from an NFA without λ -Transitions

3.1 Overview

Since an NFA is non-deterministic, it can be in multiple states at once. This final step in converting an NFA to a DFA removes any non-determinism from an NFA with no λ -transitions. Such a DFA will have one state for every *set of states* that the NFA can be in. In general, since an NFA can be in any number of its states at once, the number of states in the corresponding DFA can be as large as the power set of all the NFA (e.g. if there are n states in the NFA, then there may be as many as $2^n + 1$ states in the corresponding DFA including the reject state!). Since the DFA's states are based on sets of the NFA's states, we name our DFA's states as sets which contain the NFA states it represents. In other words, if a state in the DFA corresponds to q_0 and q_1 in the NFA, we would name the DFA state $\{q_0, q_1\}$. If a state in the DFA corresponds to just q_2 in the NFA, then we would name the DFA state $\{q_2\}$.

This final step is encoded as a simple table-driven process described here. The table is composed of one row per state in the DFA being generated. There is a column to note which state the row belongs to, a column which tells us whether or not the state is an accept state, and one column per symbol in the alphabet. Cells in each of these symbol columns tell us where the state described by the row transitions to on a given symbol. Table 1 shows the format just described.

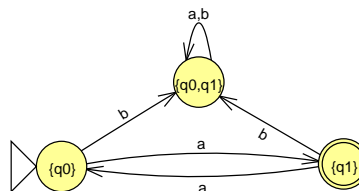


Figure 7: The DFA described by Table 1.

State	Symbols		Accept State?
	a	b	
{q0}	{q1}	{q0, q1}	No
{q1}	{q0}	{q0, q1}	Yes
{q0, q1}	{q0, q1}	{q0, q1}	No

Table 1: This is an example of a table which might be built by the algorithm described in step 3. The corresponding DFA is shown in Figure 7. It describes a DFA with the states {q0}, {q1}, and {q0, q1}. The symbol columns specify transitions from each of these states. For instance {q0} transitions to {q1} on a and to {q0, q1} on b.

3.2 Rules

Now that we know where we are going, lets look at the rules for building the table:

1. Create a state in your DFA which corresponds to the start state in the N DFA. This state is the first row in your table. Fill in the transitions for this state as described in rule 2.
2. Fill in the transitions for each symbol in the alphabet. For each symbol x , the DFA should transition to the set of all states the N DFA could transition to on x from all N DFA states represented by the current DFA state. If there is no transition for a symbol in the N DFA, then add a transition to the reject state which is named $\{\emptyset\}$.
3. If any transition goes to a state not yet in the table, add a new row for the new state and fill in its transitions as described in rule 2.
4. If *any* of the N DFA states which make up a DFA state was an accept state, then the new DFA state is also an accept state.

3.3 Example

To illustrate this step, we will convert the N DFA from Figure 6 to a DFA. To start, we add the start state from the N DFA, or q0, to the table as our first DFA state {q0}. For the symbol a, the N DFA in q0 transitions to q2 and q3. Therefore the DFA will transition from {q0} to the new DFA state {q2, q3} on a. Since the {q2, q3} state is new, we add a new row in the table for it which we will fill in after we finish with {q0}. The transitions on the symbols b and c are very similar to a. Finally, for d we add a transition to the new DFA state {q3}. q0 was not an accept state in the N DFA, so {q0} will not be an accept state in the DFA either. We are now done with the first row in the table. Our progress so far is shown in Table 2.

Now we can move on to the second row and fill in the transitions for {q2, q3}. When determining the transitions out of this state, we consider the the transitions out of N DFA states q2 *and* q3 since both of those states jointly make up {q2, q3}. Since q2 transitions to itself on a and q3 does not have a transition for a, {q2, q3} transitions to the aggregate

State	Symbols				Accept State?
	a	b	c	d	
{q0}	{q2, q3}	{q1, q3}	{q1, q2}	{ \emptyset }	No
{q2, q3}					
{q1, q3}					
{q1, q2}					
{ \emptyset }					

Table 2: This is the state of the conversion table after filling in the first row in the table for the conversion of the NDFFA in Figure 6 to a DFA.

of these two, or simply {q2}. This is another new state which we add to our table. The other transitions are computed in the same manner. Finally, we note that {q2, q3} is an accept state because one of its component states from the NDFFA, q3, was an accept state in the NDFFA. We are now done with the second row in the table. Our progress so far is shown in Table 3.

State	Symbols				Accept State?	
	a	b	c	d		
{q0}	{q2, q3}	{q1, q3}	{q1, q2}	{ \emptyset }	No	
{q2, q3}	{q2}	{q3}	{q1}	{q3}		Yes
{q1, q3}						
{q1, q2}						
{ \emptyset }						
{q2}						
{q1}						
{q3}						

Table 3: This is the state of the conversion table after filling in the second row in the table for the conversion of the NDFFA in Figure 6 to a DFA.

Filling in the remainder of the rows simply involves applying rules we discussed above. The final table is shown in Table 4. This table contains all the information needed to draw the final DFA. A diagram of this final DFA is shown in Figure 8.

State	Symbols				Accept State?
	a	b	c	d	
{q0}	{q2, q3}	{q1, q3}	{q1, q2}	{ \emptyset }	No
{q2, q3}	{q2}	{q3}	{q1}	{q3}	Yes
{q1, q3}	{q3}	{q1}	{q2}	{ \emptyset }	Yes
{q1, q2}	{q2, q3}	{q1, q3}	{q1, q2}	{q3}	No
{ \emptyset }	{ \emptyset }	{ \emptyset }	{ \emptyset }	{ \emptyset }	No
{q2}	{q2}	{q3}	{q1}	{q3}	No
{q1}	{q3}	{q1}	{q2}	{ \emptyset }	No
{q3}	{ \emptyset }	{ \emptyset }	{ \emptyset }	{ \emptyset }	Yes

Table 4: This is the completed conversion table for the conversion of the NDFSA in Figure 6 to a DFA. The corresponding DFA is shown in Figure 8.

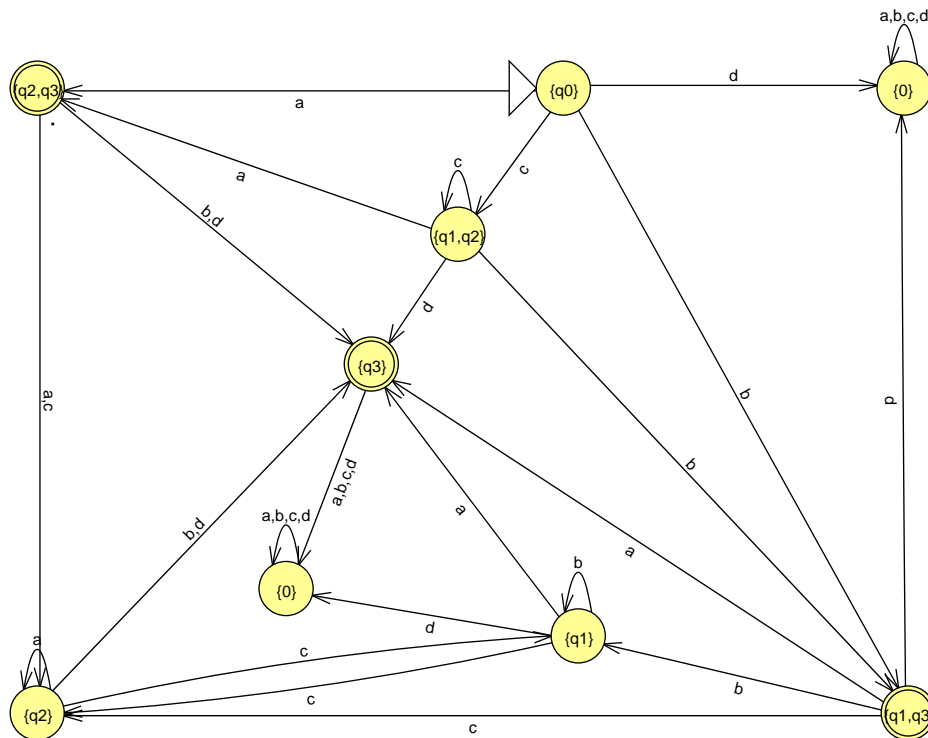


Figure 8: This the DFA described by Table 4. It may look bad, but while the number of states did double, it could have been even worse!