ECS 120 Lesson 6 – Nondeterministic Finite State Machines, Pt. 2

Oliver Kreylos

Wednesday, April 11th, 2001

1 Validity of the Subset Construction

Last time, we introduced the subset construction to build a DFA $D = (Q_D, \Sigma, \delta_D, q_{D0}, F_D)$ that simulates an NFA $M = (Q, \Sigma, \delta, q_0, F)$. Today we have to prove that these two automata are in fact equivalent, i. e., that they accept the same language: $L(M) = L(D)$. We will prove this by showing that the two automata's extended transition functions are identical, using induction over the length of a word $w \in \Sigma^*$.

Claim For any subset of states $P \subset Q$ and any word $w \in \Sigma^*$, $\delta^*(P, w) = \delta_D^*(\text{ECLOSE}(P), w)$.

Induction Basis Let $P \subset Q$ be any subset of states, and let w be the empty word, $w = \epsilon$. In this case, $|w| = 0$. Then $\delta^*(P, w) = \delta^*(P, \epsilon) = \text{ECLOSE}(P)$. Starting from the other direction, $\delta_D^*(\text{ECLOSE}(P), w) = \delta_D^*(\text{ECLOSE}(P), \epsilon) = \text{ECLOSE}(P)$.

Induction Hypothesis Assume the claim is true for all words $w \in \Sigma$ with $|w| = n$, $n \geq 0$.

Induction Step Let $P \subset Q$ be any subset of states, and let $w \in \Sigma^*$ be any word of length $|w| = n + 1$. Then w can be written as $w = ax$, where $a \in \Sigma$ and $x \in \Sigma^*$ with $|x| = n$. Then $\delta^*(P, w) = \delta^*(P, ax) = \delta^*\big(\delta(\text{ECLOSE}(P), a), x\big) = \delta^*(P', x)$ with $P' := \delta(\text{ECLOSE}(P), a)$. On the other hand, $\delta_D^*(\text{ECLOSE}(P), w) = \delta_D^*(\text{ECLOSE}(P), ax) = \delta_D^*\big(\delta_D(\text{ECLOSE}(P), a), x\big) = \delta_D^*\big(\text{ECLOSE}\big(\delta(\text{ECLOSE}(P), a)\big), x\big) =$

$\delta_D^*\big(\text{ECLOSE}(P'), x\big)$. Since $|x| = n$, we can now apply the induction hypothesis $\delta^*(P', x) = \delta_D^*\big(\text{ECLOSE}(P'), x\big)$.

To finish the proof of $L(M) = L(D)$, we now have to consider $\delta^*(q_0, w)$ and $\delta_D^*(q_{D0}, w)$. Using the result from above and $q_{D0} = \text{ECLOSE}(q_0)$, we have $\delta^*(q_0, w) = \delta_D^*\big(\text{ECLOSE}(q_0), w\big) = \delta_D^*(q_{D0}, w)$. Now let $w \in \Sigma^*$ be any word. M accepts w, iff $\delta^*(q_0, w) \cap F \neq \emptyset$. D accepts w, iff $\delta_D^*(q_{D0}, w) \in F_D = \big\{\, P \in \mathcal{P}(Q) \;\big|\; P \cap F \neq \emptyset \,\big\}$. These two statements are equivalent; therefore, M accepts w exactly if D accepts w.

2 Closure of Regular Languages – Continued

Now having the necessary tool in form of nondeterministic automata, we will continue looking at the behaviour of regular languages under the language operations we defined earlier.

2.1 Union

We already proved that regular languages are closed under the union operation. To show this claim, we had to construct the product automaton. Using nondeterministic finite automata, the proof becomes much easier. Let $A, B \subset \Sigma^*$ be two regular languages, and let $M_1 = (Q_1, \Sigma, \delta_1, s_1, F_1)$ and $M_2 = (Q_2, \Sigma, \delta_2, s_2, F_2)$ be two DFAs deciding A and B, respectively. For convenience, let us assume that the sets of states are disjunct: $Q_1 \cap Q_2 = \emptyset$ (this can always be achieved by renaming states). Then we build an NFA $N := (Q, \Sigma, \delta, q_0, F)$, where:

- The set of states $Q := Q_1 \cup Q_2 \cup \{q_0\}$ is the union of M_1's and M_2's state sets and the new state q_0. We assume that q_0 is neither in Q_1 nor in Q_2.
- The transition function is defined by

$$\delta(q, a) := \begin{cases} \delta_1(q, a), & \text{if } q \in Q_1 \\ \delta_2(q, a), & \text{if } q \in Q_2 \\ \emptyset, & \text{if } q = q_0 \end{cases}$$

$$\delta(q, \epsilon) := \begin{cases} \{s_1, s_2\}, & \text{if } q = q_0 \\ \emptyset & \text{otherwise} \end{cases}$$

- $F := F_1 \cup F_2$

This construction takes the two DFAs accepting A and B almost unchanged, and adds a new start state that has ϵ-transitions to the start states of M_1 and M_2, see Figure 1. The effect is, that the computation of N branches before any input characters are read, and continues in those two branches in parallel until the input word is completely read. There are no further branches and no early terminations, because the machines M_1 and M_2 are DFAs. The resulting NFA has exactly $|Q_1| + |Q_2| + 1$ states – as opposed to the DFA accepting the same language, which has $|Q_1| \cdot |Q_2|$ states. Also, the structure of the NFA is much closer related to the structure of the two original DFAs. This example shows that NFAs are a more powerful way to describe automata.

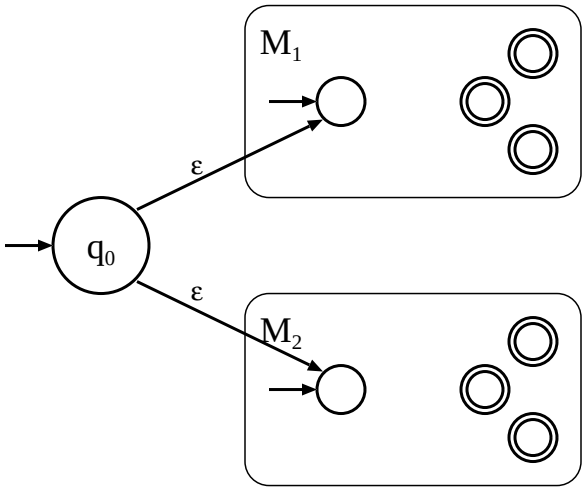

Figure 1: Union of two DFAs M_1 and M_2. The automata are depicted as "black boxes," showing only their start states and their final states.

2.2 Concatenation

Let $A, B \subset \Sigma^*$ be two regular languages, and let $M_1 = (Q_1, \Sigma, \delta_1, s_1, F_1)$ and $M_2 = (Q_2, \Sigma, \delta_2, s_2, F_2)$ be the DFAs that accept them. Again, we assume that $Q_1 \cap Q_2 = \emptyset$. We now define the NFA that will decide the language AB as $N := (Q, \Sigma, \delta, s_1, F_2)$ where:

- The set of states $Q := Q_1 \cup Q_2$ is the union of M_1's and M_2's state sets.

- The transition function δ is defined by

$$\begin{aligned} \delta(q,a) &:= \begin{cases} \delta_1(q,a), & \text{if } q \in Q_1 \\ \delta_2(q,a), & \text{if } q \in Q_2 \end{cases} \\ \delta(q,\epsilon) &:= \begin{cases} \{s_2\}, & \text{if } q \in F_1 \\ \emptyset & \text{otherwise} \end{cases} \end{aligned}$$

Informally, the new automaton N is constructed by concatenating the two DFAs M_1 and M_2 in the following way: Each final state of M_1 is connected to the start state of M_2 by an ϵ-transition. The start state of N is the start state of M_1, and the final states of N are exactly the final states of M_2, see Figure 2. The machine works in the following way: It starts reading the input word exactly as M_1 would do, and whenever the computation reaches a final state of M_1, the machine branches, and one branch continues computation in the start state of M_2, reading the rest of the input word exactly as M_2 would do. Very informally speaking, the machine "guesses" for each word $ab \in AB$ where a stops and where b begins. It then feeds a into machine M_1 and b into machine M_2. If both parts of the word are accepted, the whole word is accepted.

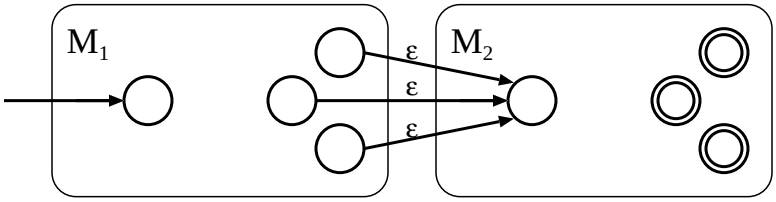

Figure 2: Concatenation of two DFAs M_1 and M_2. The complete machine's start state is the start state of M_1, and the complete machine's final states are exactly the final states of M_2.

To prove the correctness of this construction, we have to show that the language accepted by machine N is $L(N) = AB$. This is equivalent to proving that a word $w \in \Sigma^*$ is accepted by N if and only if it is in the set AB.

Let us start with showing the direction $w \in AB \Longrightarrow N$ accepts W: Take any word $w \in AB$. By definition of AB, any such word can be written as $w = ab$ where $a \in A$ and $b \in B$. Looking at the two machines M_1 and M_2 independently, we see that there must be a chain of states in M_1 leading

from its start state s_1 to some final state $f_1 \in F_1$, such that the labels along the chain spell out the word a. Simililarly, there must be a chain of states in M_2 leading from s_2 to some $f_2 \in F_2$. Since, by our construction, there is an ϵ-transition from any final state in M_1 to the start state of M_2, there must be an ϵ-transition from f_1 to s_2. This means, the concatenation of the two chains is also a valid chain from the start state to a final state in automaton N, and the characters along that chain spell out the word $ab = w$. Therefore, w is accepted by N.

To prove the other direction, N accepts $W \Longrightarrow w \in AB$, we now assume that N accepts some word $w \in \Sigma^*$. This means that $\delta^*(s_1, w) \cap F_2 \neq \emptyset$; in other words, there must be some final state $f \in F_2$ in $\delta^*(s_1, w)$. Since $F = F_2 \subset Q_2$, all of N's final states are in M_2. But N's start state is in M_1; therefore, the chain of states leading from s_1 to f must have crossed from M_1 to M_2 at some point during the computation. The only transitions from M_1 to M_2 are the ϵ-transitions we added from each final state $f_1 \in F_1$ to s_2. Also, the chain of states can only have crossed once, since there are no transitions from any state in M_2 to any state in M_1. We can break the chain of states into two parts: One part starts at s_1 and ends in some state $f_1 \in F_1$, the other part starts at s_2 and ends in state $f_2 \in F_2$. Since the transition from f_1 to s_2 was an ϵ-transition and did not read any characters, the characters along the first part of the chain spell out a word $a \in \Sigma^*$, and the characters along the second part of the chain spell out another word $b \in \Sigma^*$, with $w = ab$. Both parts of the chain start in a start state and end in a final state; therefore, a must be accepted by M_1 and b must be accepted by M_2, i. e., $w = ab \in AB$.

2.3 Kleene Star

Let $A \subset \Sigma^*$ be a regular language, and let $M = (Q, \Sigma, \delta, q_0, F)$ be the machine that accepts A. We define an NFA to accept the language A^* as $N = (Q_K, \Sigma, \delta_K, s, F_K)$, where:

- The set of states is $Q_K := Q \cup \{s\}$, where s is some new state not in Q.
- The transition function δ_K is defined by

$$\delta_K(q, a) := \begin{cases} \delta(q, a), & \text{if } q \in Q \\ \emptyset & \text{otherwise} \end{cases}$$

$$\delta_K(q, \epsilon) := \begin{cases} \emptyset, & \text{if } q \in Q \setminus F \\ \{q_0\} & \text{otherwise} \end{cases}$$

- The set of final states is $F_K := F \cup \{s\}$.

Informally, the NFA N is constructed by adding a new start state s to M, making it a final state, connecting it to M's start state by an ϵ-transition and then connecting each of M's final states to the new start state s by another ϵ-transition, see Figure 3.

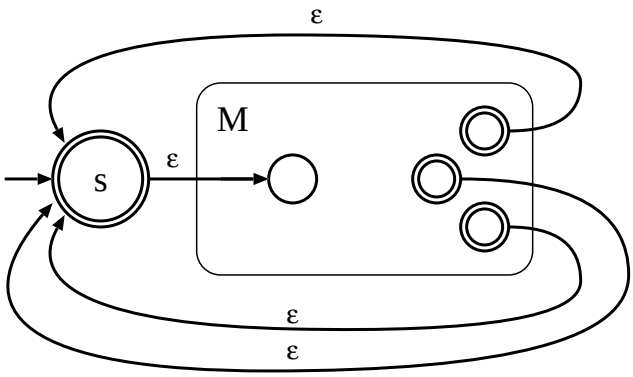

Figure 3: Kleene Star of a DFA M.

Again, let us now prove the correctness of above construction. First, we show the implication $w \in A^* \Longrightarrow N$ accepts w: Take any word $w \in A^*$. If w is the empty word, $w = \epsilon$, it is accepted by N because the start state of N is also a final state. If, on the other hand, $w \neq \epsilon$, there must be an integer $n > 0$ and words $a_1, a_2, \ldots, a_n \in A$ such that $w = a_1 a_2 \ldots a_n$. Since each of the words $a_i \in A$ is accepted by machine M, there must be a chain of states from M's start state q_0 to some final state $f_i \in F$ that spells out word a_i. Now, since the new state s is connected to M's start state q_0 by an ϵ-transition, and each state $f \in F$ is connected back to s by another ϵ-transition, all these chains together form a valid chain of states from the start state to a final state in the automaton N that spells out the word $a_1 a_2 \ldots a_n = w$. Therefore, the word w is accepted by N.

To prove the other direction, we look at any word $w \in \Sigma^*$ that is accepted by N. By definition, this means that there must be a chain of states starting at s and ending in some final state $f \in F$. If $w \neq \epsilon$, we can assume here that the chain does not end in s (which is also a final state): s can only be reached by ϵ-transitions, and by not taking the last ϵ-transition leading to s, we get another chain spelling out the same word that ends in some $f \in F$. If, on the other hand, $w = \epsilon$, then there is nothing left to prove, since $\epsilon \in A^*$

by definition. Now let us look at the end of the chain ending in some $f \in F$. The only way the computation can have reached f is by going through the start state s at some point – the computation starts in s, which is not a state of M, and the only arrow entering M is the ϵ-transition from s to q_0. By this observation, we can split the state chain into two parts: One goes from the occurrence of q_0 following the last occurrence of s in the state chain to the final state f, the other part contains all other states, see Figure 4. The transition labels along the state chain's suffix must spell out a word $a \in A$, because the suffix is a valid state chain in machine M going from its start state to one of its final states. By this observation, we can split up the input word w into a prefix $w' \in \Sigma^*$ and a suffix $a \in A$: $w = w'a$. From the recursive definition of A^* we know that $w \in A^*$ iff $w' \in A^*$. We can show that $w' \in A^*$ by repeating the last step of this proof and splitting up the prefix state chain again until it only consists of the single state s. This inductive proof shows that any word accepted by N is an element of A^*, which concludes the proof that $L(N) = AB$.

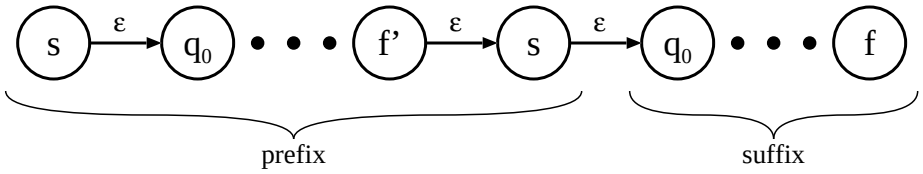

Figure 4: Splitting up a state chain in the Kleene Star automaton. The suffix spells out a word $a \in A$, the prefix spells out another word $w' \in \Sigma^*$.