NFA determinization

Data Structures and Algorithms for Computational Linguistics III
(ISCL-BA-07)

Çağrı Çöltekin
ccoltekin@sfs.uni-tuebingen.de

University of Tübingen
Seminar für Sprachwissenschaft

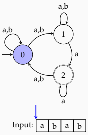
Winter Semester 2024/25

version: c8d9876 @2025-01-08

Recap

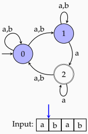
- Finite state automata come in two flavors
  - Deterministic (DFA): linear recognition time
  - Deterministic (NFA): sometimes more intuitive, easy to define, but exponential time (worst case) recognition
- The DFA and NFA are equivalent: for any language recognized by an NFA there is also a DFA recognizing the same language
- Then, the question is: how can we *determinize* an NFA to obtain an equivalent DFA

NFA recognition (again)



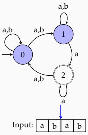
1. Start at q_0
2. Take the next input, mark all possible next states, repeat
3. If an accepting state is marked at the end of the input, accept

NFA recognition (again)



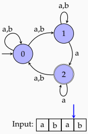
1. Start at q_0
2. Take the next input, mark all possible next states, repeat
3. If an accepting state is marked at the end of the input, accept

NFA recognition (again)



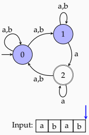
1. Start at q_0
2. Take the next input, mark all possible next states, repeat
3. If an accepting state is marked at the end of the input, accept

NFA recognition (again)



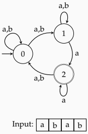
1. Start at q_0
2. Take the next input, mark all possible next states, repeat
3. If an accepting state is marked at the end of the input, accept

NFA recognition (again)



1. Start at q_0
2. Take the next input, mark all possible next states, repeat
3. If an accepting state is marked at the end of the input, accept

NFA recognition (again)



1. Start at q_0
2. Take the next input, mark all possible next states, repeat
3. If an accepting state is marked at the end of the input, accept

The process is *deterministic*, and *finite-state*.

Determinization

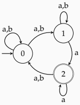
the subset construction

Intuition: remember the parallel NFA recognition. We can consider an NFA being a deterministic machine which is at a set of states at any given time.

- *Subset construction* (sometimes called power set construction) uses this intuition to convert an NFA to a DFA
- The algorithm can be modified to handle ϵ-transitions (or we can eliminate ϵ's as a preprocessing step)

The subset construction

by example



transition table with subsets

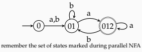
	symbol a	b
∅	∅	∅
→ {0}	{0, 1}	{0, 1}
{1}	{1, 2}	{1}
* {2}	{0, 2}	{0}
{0, 1}	{0, 1, 2}	{0, 1}
* {0, 2}	{0, 1, 2}	{0, 1}
* {1, 2}	{0, 1, 2}	{0, 1}
* {0, 1, 2}	{0, 1, 2}	{0, 1}

The subset construction

by example: the resulting DFA

transition table without useless/inaccessible states

	symbol a	b
→ {0}	{0, 1}	{0, 1}
{0, 1}	{0, 1, 2}	{0, 1}
* {0, 1, 2}	{0, 1, 2}	{0, 1}

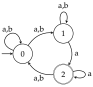


Do you remember the set of states marked during parallel NFA recognition?

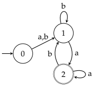
The subset construction

by example: side by side

NFA



DFA

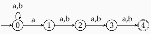


- What language do they recognize?

The subset construction

wrapping up

- In worst case, resulting DFA has 2^n nodes
- Worst case is rather rare, number of nodes in an NFA and the converted DFA are often similar
- In practice, we do not need to enumerate all 2^n subsets
- We've already seen a typical problematic case:

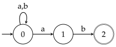


- We can also skip the unreachable states during subset construction

Yet another exercise

Determinize the following automaton

NFA:



Summary

- FSA are efficient tools with many applications
- FSA have two flavors: DFA, NFA (or maybe three: ε-NFA)
- DFA recognition is linear, recognition with NFA may require exponential time
- Reading suggestion: **hopcroft1979**, **jurafsky2009**

Next:

- Minimization

Acknowledgments, credits, references