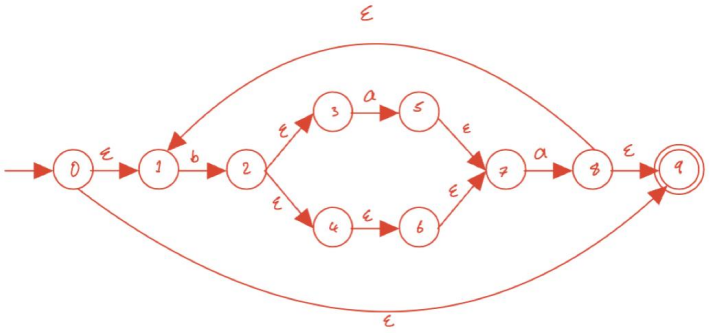
NFA to DFA Subset Construction

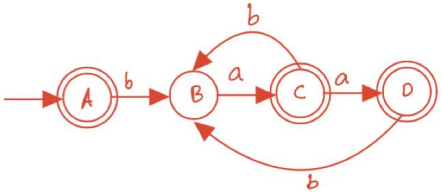
Alexis Gadonneix, Nikhil Mehta

Subset Construction Algorithm

- Means of converting NFA to DFA
- Works by considering all possible sets of states accessible in an NFA after passing all possible combinations of symbols in a language and translating those sets of states into an individual states in the DFA



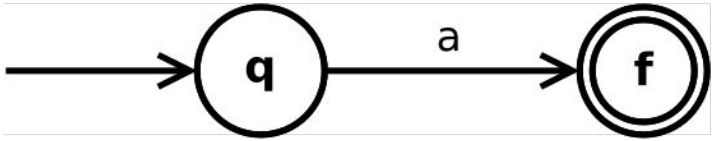
NFA state	DFA state	a	b
{0,1,9}	A	∅	B
{2,3,4,6,7}	B	C	∅
{1,5,7,8,9}	C	D	B
{1,8,9}	D	∅	B



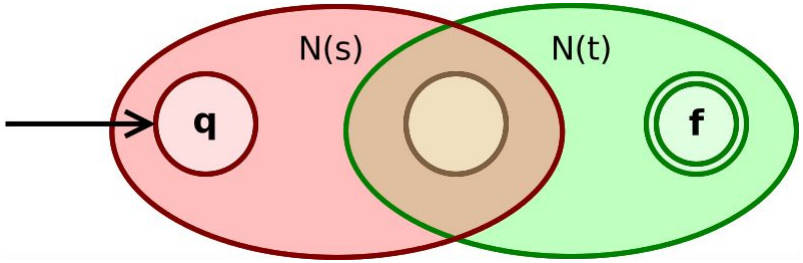
Thompson's Algorithm

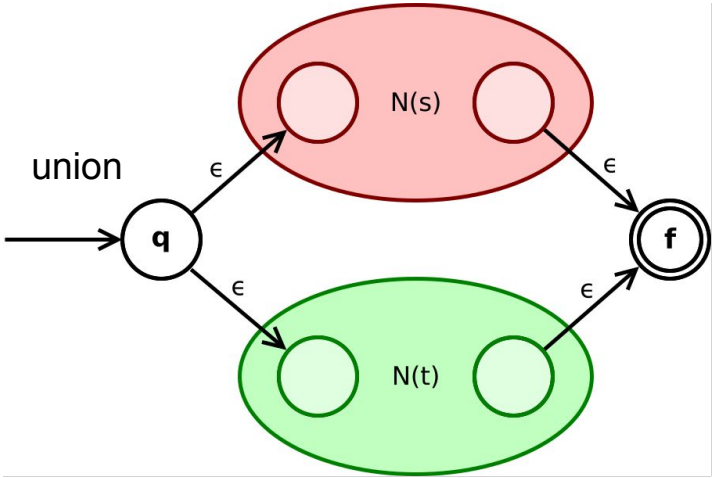
- Easy way to generate an NFA from a regex
- 4 simple rules:

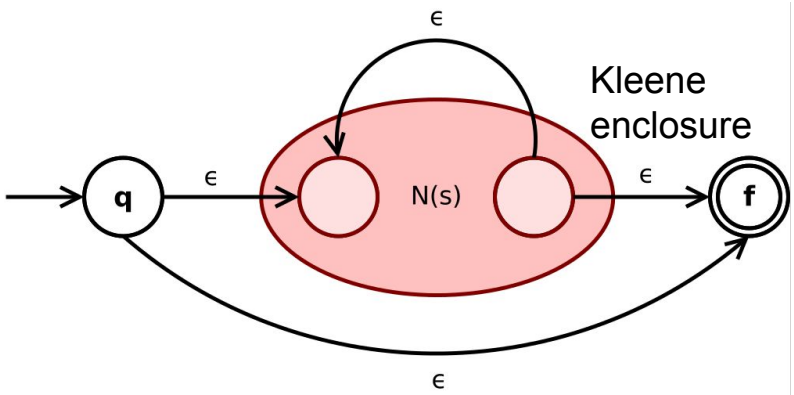
symbol



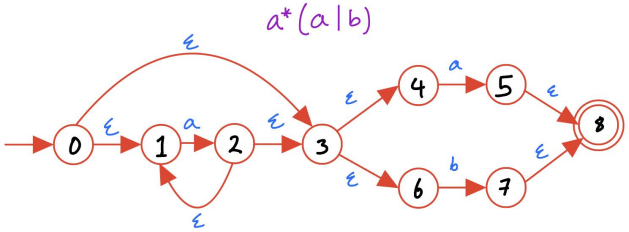
concat

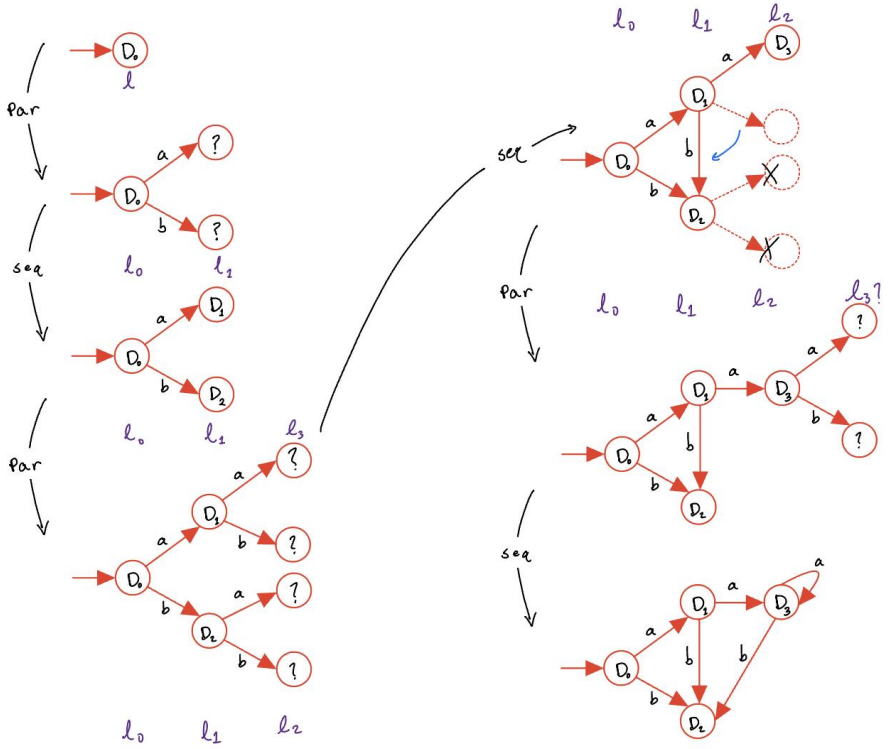






Parallelizing the algorithm





Limitations

- There is a sequential part in the algorithm → Amdahl's law

- We have to keep track of all the NFA states in each DFA state. It stored in a Map with sets of states as keys.
  → memory and garbage collection issues
  → Lookups are expensive

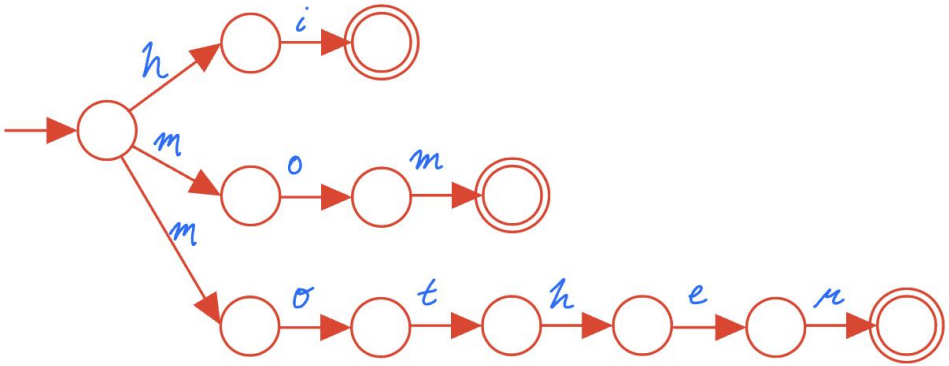
Generating data : From regexes

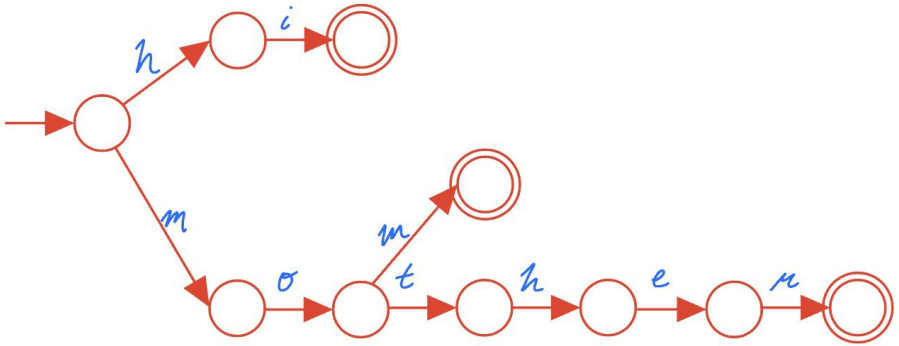
Idea : Generate random regular expressions

Problem: Not easy and not very interesting in practice

Generating data : From a list of words

Idea : Generate an NFA from a dictionary of words





Problem: A LOT of nfa states in a DFA state → Map lookups are very expensive and the algorithm spends most of the time in the sequential part

Generating data : A random NFA

Idea : Generate an NFA from a number of states, an alphabet and a probability that a given edge exists

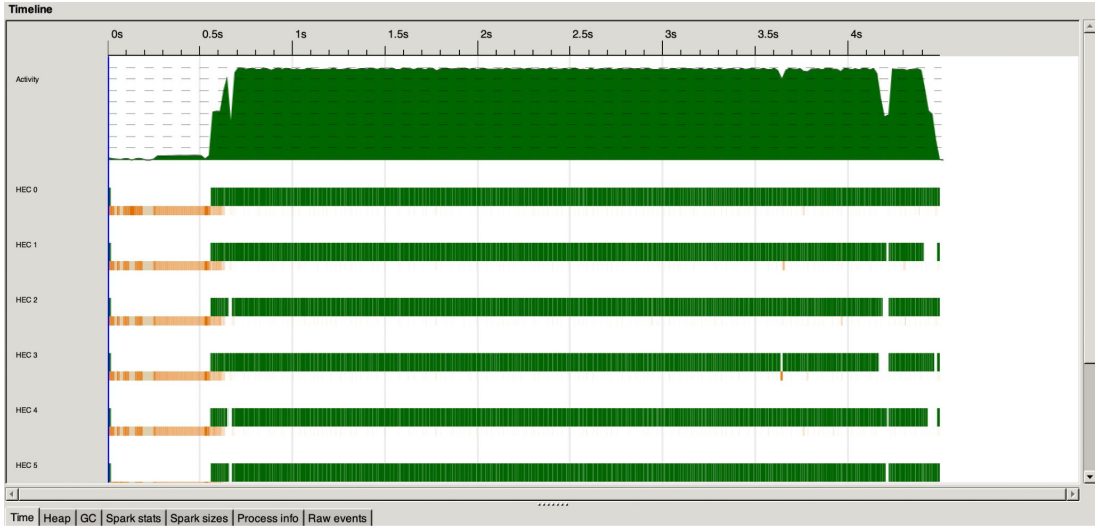
Problems:

- Is it useful?
- Keep the graph connected
- Control Epsilon transitions

Results

For random NFAs with:

- 500 states
- an alphabet of size 20
- a probability of 50%



SPARKS: 440 (437 converted, 0 overflowed, 0 dud, 0 GC'd, 3 fizzled)

Speed-up using 8 cores: 2.9