DFA revisit (pg 19; figure 2.7)

Design a DFA that accepts the language of all strings with an even number of 'a's and odd number of 'b's. (This is a classic problem):

Write $i$ in binary, where:

- 0 - 00
- 1 - 01
- 2 - 10
- 3 - 11

We use 0 to indicate even number (for any even number mod 2 is 0)
We use 1 to indicate odd number (for any odd number mod 2 is 1)

We will use the first index to indicate the parity of 'a', the second index the parity of 'b'. Hence a “10” means "odd number of 'a's and even number of 'b's".
Q: Suppose we want to design an FA to accept the language of all strings over \{a, b\} with 'aa' as a substring or 'bb' as a substring.

A: DFA for strings with 'aa' as a substring:

```
DFA for strings with 'bb' as a substring:

We could combine these two DFA’s to construct an FA to accept the union of the two languages:

- Add a new initial state
```
This FA is not deterministic:

1. We have two choices to move to the next state from the initial state.
2. e-moves (e-links) are introduced.

---

2.2: Nondeterministic Finite Automata (NFA)

**Definition:** An NFA M is a tuple \((Q, \Sigma, \Delta, s, F)\), where

- \(Q, \Delta, s, F\) are the same as in DFA and
- \(\Delta\) is a transition function: \(Q \times (\Sigma \cup \{e\}) \rightarrow 2^Q\) (\(2^Q\) represents the set of all subsets of \(Q\))

(We will show that NFA does not accept more languages then DFA.)

**Acceptance Notion (NFA)**

- For any given input \(x\) there may be more than one computation path.
- As long as there is one computation path that ends at an accepting state then we say that \(x\) is accepted.

A DFA is a special case of NFA.

Let \(L(M)\) be the set of all strings accepted by \(M\).

- The computation of a NFA \(M\) on input \(x\) can be viewed as a tree:

  ![Tree Diagram](image)

  - 'H' is the height of the tree (which equals the length of the input)
  - As long as one leaf is in the final state, accept.
Advantage of NFA: It is easier to construct an NFA to accept a regular language than a DFA.

For example, let us define a DFA to accept the set of all binary strings with 101 as a substring:

We can construct a NFA for any regular language using an algorithm:

- To construct a DFA is often highly combinatorial.
- To construct a NFA is mechanical (algorithmic).

Q: Do NFA accept more languages than DFA? In other words, is there a language L that is accepted by an NFA but not by any DFA?
A: No.

=> There must be a way to convert NFA to DFA.

Q: How do we convert NFA to DFA to accept the same language?

- To do so, we want to "compress" ("collect") a set of states into one state:

- We also need to handle e-moves:
  
  If the state 'q' can be reached from state 'p' via a sequence of e-links, then state 'q’ cannot be separated from state ‘p’.

- Concept called "E-closure":

  E-closure of a state 'p': E[p] = \{q | q ∈ Q and q can be reached from p via a sequence of e-links\}

End of lecture.