Converting Regular Expressions Directly to DFAs
Scheduled for 18.12.07 5th and 8th Hour

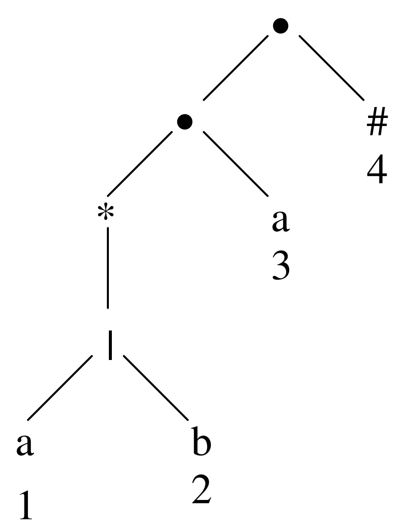
- We may convert a regular expression into a DFA (without creating a NFA first).
- First we augment the given regular expression by concatenating it with a special symbol #.

    r ➔ (r)# augmented regular expression

- Then, we create a syntax tree for this augmented regular expression.
- In this syntax tree, all alphabet symbols (including #) in the augmented regular expression will be on the leaves, and all inner nodes will be the operators in that augmented regular expression.
- Then each alphabet symbol (including #) will be numbered (position numbers).

Regular Expression ➔ DFA (cont.)

$(a|b)^* a$ ➔ $(a|b)^* a \#$ augmented regular expression



Syntax tree of $(a|b)^* a \#$

- each symbol is numbered (positions)
- each symbol is at a leaf
- inner nodes are operators

firstpos, lastpos, nullable

- To evaluate followpos, we need three more functions to be defined for the nodes (not just for leaves) of the syntax tree.

- **firstpos(n)** -- the set of the positions of the **first** symbols of strings generated by the sub-expression rooted by n.

- **lastpos(n)** -- the set of the positions of the **last** symbols of strings generated by the sub-expression rooted by n.

- **nullable(n)** -- *true* if the empty string is a member of strings generated by the sub-expression rooted by n
  false otherwise

followpos

Then we define the function **followpos** for the positions (positions assigned to leaves).

followpos(i) -- is the set of positions which can follow the position i in the strings generated by the augmented regular expression.

For example, $\underset{1}{(\,a} \mid \underset{2}{b})^{*} \underset{3}{a}\ \underset{4}{\#}$

followpos(1) = {1,2,3}
followpos(2) = {1,2,3}
followpos(3) = {4}
followpos(4) = { }

followpos is just defined for leaves, it is not defined for inner nodes.

How to evaluate firstpos, lastpos, nullable

n	nullable(n)	firstpos(n)	lastpos(n)
leaf labeled ε	true	Φ	Φ
leaf labeled with position i	false	{i}	{i}
$\vert$ c_1 c_2	nullable(c_1) or nullable(c_2)	firstpos(c_1) ∪ firstpos(c_2)	lastpos(c_1) ∪ lastpos(c_2)
• c_1 c_2	nullable(c_1) and nullable(c_2)	if (nullable(c_1)) firstpos(c_1) ∪ firstpos(c_2) else firstpos(c_1)	if (nullable(c_2)) lastpos(c_1) ∪ lastpos(c_2) else lastpos(c_2)
* c_1	true	firstpos(c_1)	lastpos(c_1)

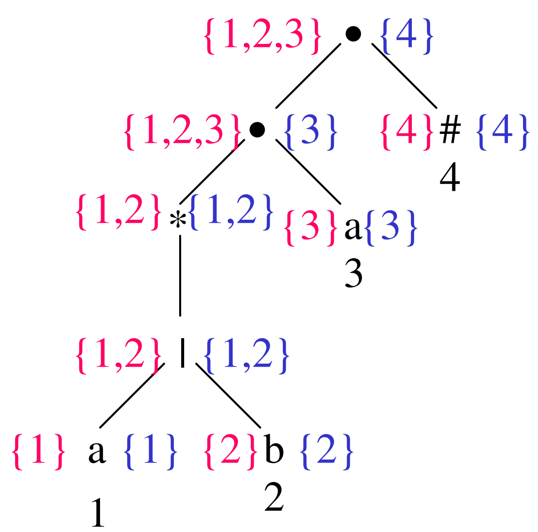
How to evaluate followpos

- Two-rules define the function followpos:

1. If **n** is concatenation-node with left child c_1 and right child c_2, and **i** is a position in **lastpos(c_1)**, then all positions in **firstpos(c_2)** are in **followpos(i)**.
2. If **n** is a star-node, and **i** is a position in **lastpos(n),** then all positions in **firstpos(n)** are in **followpos(i)**.

- If firstpos and lastpos have been computed for each node, followpos of each position can be computed by making one depth-first traversal of the syntax tree.

Example -- $(a \mid b)^* a \ \#$



pink – firstpos
blue – lastpos

Then we can calculate followpos

followpos(1) = {1,2,3}
followpos(2) = {1,2,3}
followpos(3) = {4}
followpos(4) = {}

- After we calculate follow positions, we are ready to create DFA for the regular expression.

Algorithm (RE → DFA)

- Create the syntax tree of (r) #
- Calculate the functions: followpos, firstpos, lastpos, nullable
- Put firstpos(root) into the states of DFA as an unmarked state.
- *while* (there is an unmarked state S in the states of DFA) *do*
  - mark **S**
  - *for each* input symbol **a** *do*
    - let U be the set of positions that are in followpos(p)
      for some position p in T,
      such that the symbol at position p is a;

    - if (U is not empty and not in the states of DFA)
      - add U as an unmarked state to Dstates.

    - Dtran[T,a}=U

- *the start state of DFA is firstpos(root)*
- *the accepting states of DFA are all states containing the position of #*

Example -- $(\underset{1}{a} \mid \underset{2}{b})^{*} \underset{3}{a} \underset{4}{\#}$

followpos(1)={1,2,3}   followpos(2)={1,2,3}   followpos(3)={4}   followpos(4)={}

A=firstpos(root)={1,2,3}

$\Downarrow$ mark A

A on input symbol a: followpos(1) $\cup$ followpos(3)={1,2,3,4}=B

move(A,a)=B

A on input symbol b: followpos(2)={1,2,3}=A

move(A,b)=A

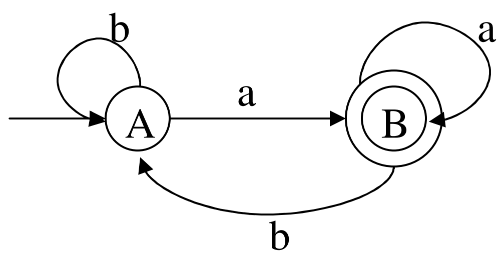
$\Downarrow$ mark B

B on input symbol a: followpos(1) $\cup$ followpos(3)={1,2,3,4}=B

move(B,a)=B

A on input symbol b: followpos(2)={1,2,3}=A

move(B,b)=A



start state: S_1

accepting states: $\{S_2\}$