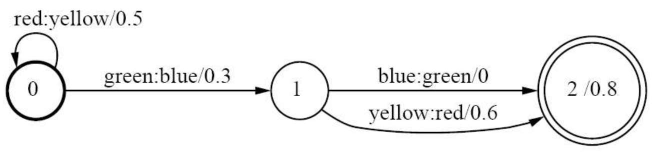
Weighted Finite State Transducer (WFST)

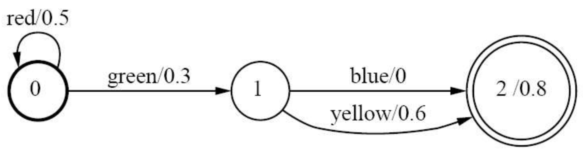
- Efficient algorithms for various operations.
- Weights
  – Handle uncertainty in text, handwritten text, speech, image, biological sequences.
- Applications:
  – Text: pattern-matching, indexation, compression.
  – Speech: speech recognition, speech synthesis.
  – Image: image compression, filters.

Weighted Finite State Transducer (WFST)

- Transducers:

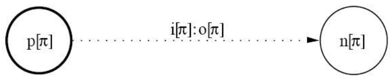


- Automata/Acceptors



WFST Definition (I)

- A path *π*: a sequence of transitions.
  - Original and destination states
  - Input and output labels



- A semiring ≡ a ring without negation
  - Number set K.
  - Sum $\oplus$ and Product $\otimes$.
- Semiring examples:
  - Probability semiring: *R, +, X.*
  - Tropical semiring: *R, min, +.*

WFST Definition (II)

- General Definitions
  - Alphabets: input *Σ*, output *Δ*
  - States: *Q*, initial *I*, final *F*.
  - Transitions: E → Q * *(Σ U ϵ)* * *(Δ U ϵ)* * *K* * Q
  - Initial/Final weights: *λ = I → K, ρ = F → K*
- WFST *T = (Σ, Q, I, F, E, λ, ρ):*

$$[T](x,y) = \bigoplus_{\pi \in P(I,x,y,F)} \lambda(p[\pi]) \otimes w[\pi] \otimes \rho(n[\pi])$$

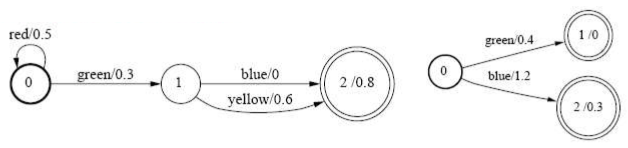
for all $x \in \Sigma^*$ and $y \in \Delta^*$.

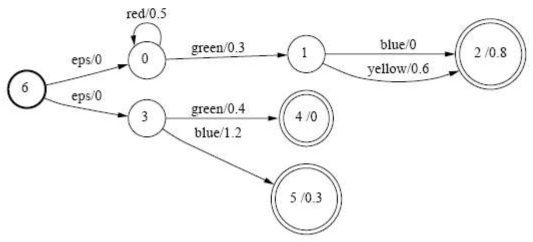
WFST Operations

- Sum
- Product
- Closure
- Reversal
- Composition
- Determinization
- Weight pushing
- Minimization

WFST Sum

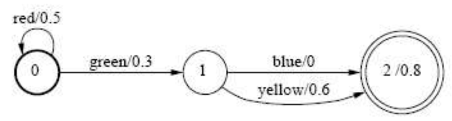
- Sum: $[T_1 \oplus T_2](x,y) = [T_1](x,y) \oplus [T_2](x,y)$



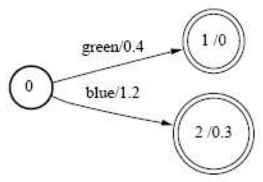


WFST Product

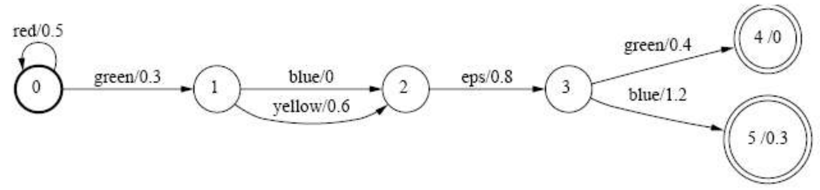
- **Product:** $[T_1 \otimes T_2](x,y) = \bigoplus_{x=x_1x_2,\, y=y_1y_2} [T_1](x_1,y_1) \otimes [T_2](x_2,y_2)$



A.fsa

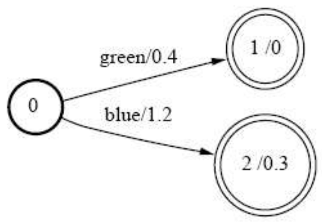


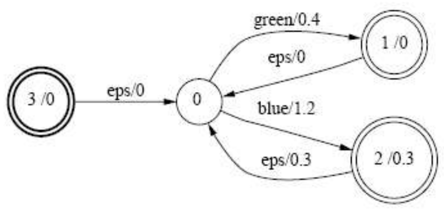
B.fsa



WFST Closure

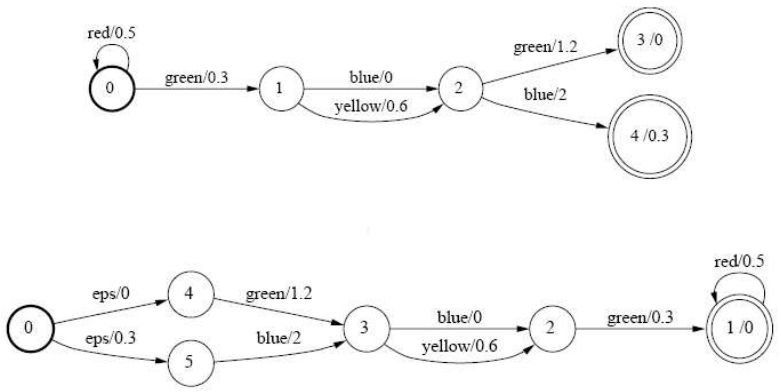
- **Closure:** $[T^*](x,y) = \bigoplus_{n=0}^{\infty} [T]^n(x,y)$





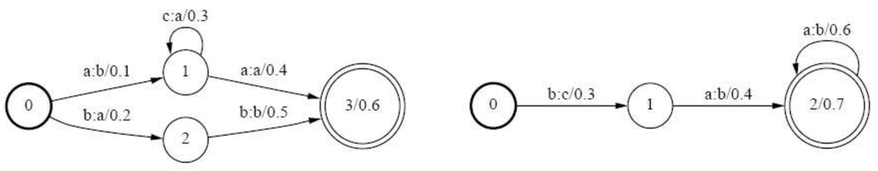
WFST Reversal

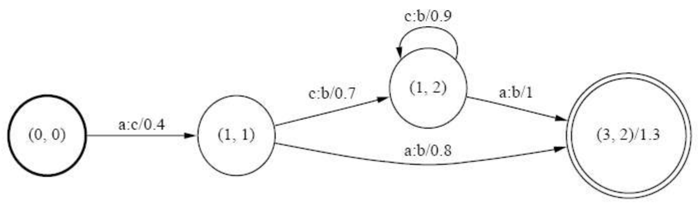
- **Reversal:** $[\widetilde{T}](x,y) = [T](\widetilde{x},\widetilde{y})$



WFST Composition

- **Composition:** $[T_1 \circ T_2](x,y) = \bigoplus_z [T_1](x,z) \otimes [T_2](z,y)$



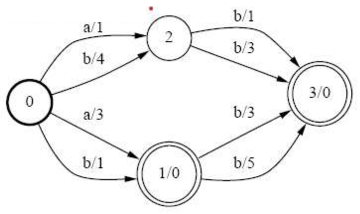


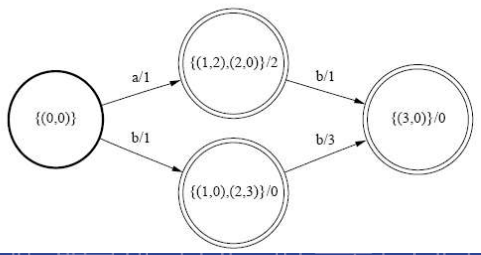
WFST Composition Algorithm

```
WEIGHTED-COMPOSITION($T_1, T_2$)
$Q \leftarrow I_1 \times I_2$
$S \leftarrow I_1 \times I_2$
while $S \neq \emptyset$ do
    $(q_1, q_2) \leftarrow$ HEAD($S$)
    DEQUEUE($S$)
    if $(q_1, q_2) \in I_1 \times I_2$ then
        $I \leftarrow I \cup \{(q_1, q_2)\}$
        $\lambda(q_1, q_2) \leftarrow \lambda_1(q_1) \otimes \lambda_2(q_2)$
    if $(q_1, q_2) \in F_1 \times F_2$ then
        $F \leftarrow F \cup \{(q_1, q_2)\}$
        $\rho(q_1, q_2) \leftarrow \rho_1(q_1) \otimes \rho_2(q_2)$
    for each $(e_1, e_2) \in E[q_1] \times E[q_2]$ such that $o[e_1] = i[e_2]$ do
        if $(n[e_1], n[e_2]) \notin Q$ then
            $Q \leftarrow Q \cup \{(n[e_1], n[e_2])\}$
            ENQUEUE($S, (n[e_1], n[e_2])$)
        $E \leftarrow E \cup \{((q_1, q_2), i[e_1], o[e_2], w[e_1] \otimes w[e_2], (n[e_1], n[e_2]))\}$
return $T$
```

WFST Determinization

- Deterministic WFST: no common input label for all outgoing transitions from any state.
- Determinimization: determinizable WFST → deterministic W.



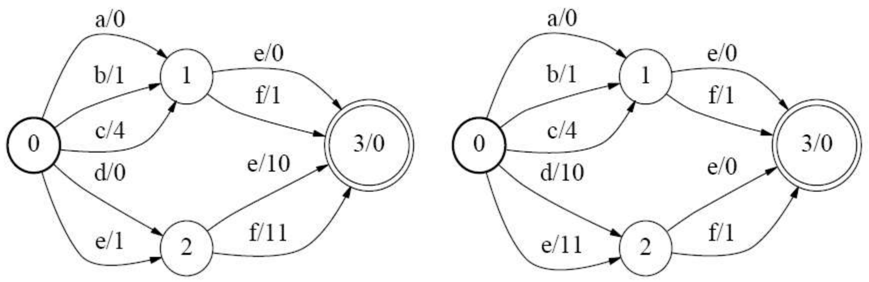


WFST Determinization Algorithm

```
WEIGHTED-DETERMINIZATION($A$)
$i' \leftarrow \{(i, \lambda(i)) : i \in I\}$
$\lambda'(i') \leftarrow \overline{1}$
$S \leftarrow \{i'\}$
while $S \neq \emptyset$ do
    $p' \leftarrow$ HEAD($S$)
    DEQUEUE($S$)
    for each $x \in i[E[Q[p']]]$ do
        $w' \leftarrow \bigoplus \{v \otimes w : (p, v) \in p', (p, x, w, q) \in E\}$
        $q' \leftarrow \{(q, \bigoplus \{w'^{-1} \otimes (v \otimes w) : (p, v) \in p', (p, x, w, q) \in E\}) :$
            $q = n[e], i[e] = x, e \in E[Q[p']]\}$
        $E' \leftarrow E' \cup \{(p', x, w', q')\}$
        if $q' \notin Q'$ then
            $Q' \leftarrow Q' \cup \{q'\}$
            if $Q[q'] \cap F \neq \emptyset$ then
                $F' \leftarrow F' \cup \{q'\}$
                $\rho'(q') \leftarrow \bigoplus \{v \otimes \rho(q) : (q, v) \in q', q \in F\}$
            ENQUEUE($S, q'$)
return $T'$
```

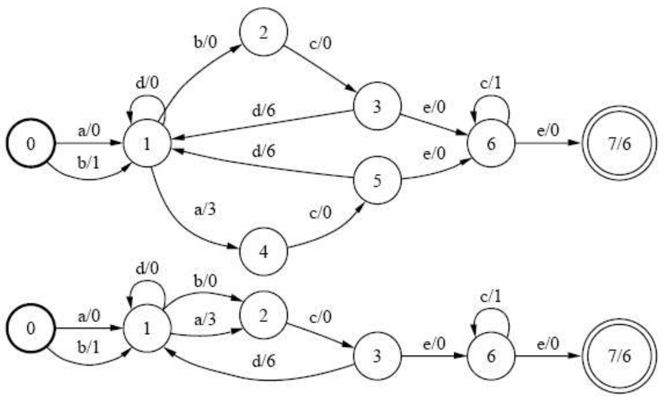
WFST Weights Pushing

- Weight pushing: re-distribute all weights along paths.



WFST Minimization

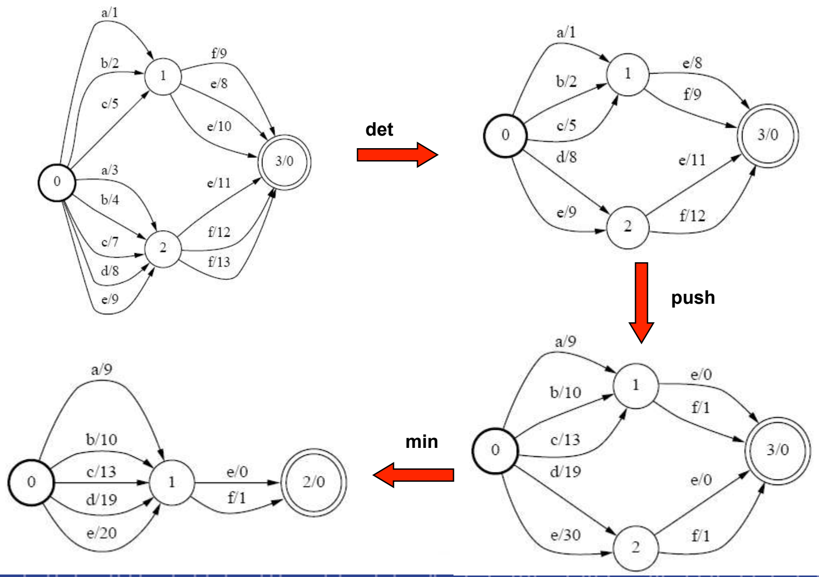
- Minimize number of states and transitions of a deterministic WFST.



WFST Operations

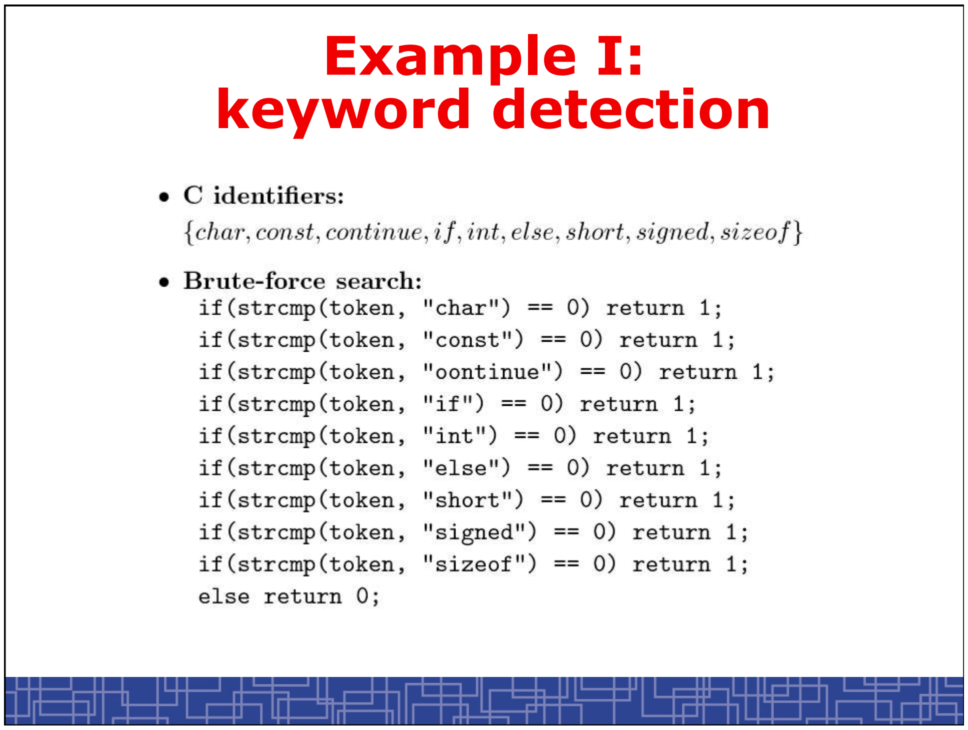
- Composition: *C = A ○ B*
- Determinization: *D = det(C)*
  – *deterministic automaton: every state has at most one out-going transition with any given label.*
- Re-weighting (Weight pushing): *E = push(D)*
- Minimization: *F = min(E)*

WFST Operations: Examples



WFST Applications

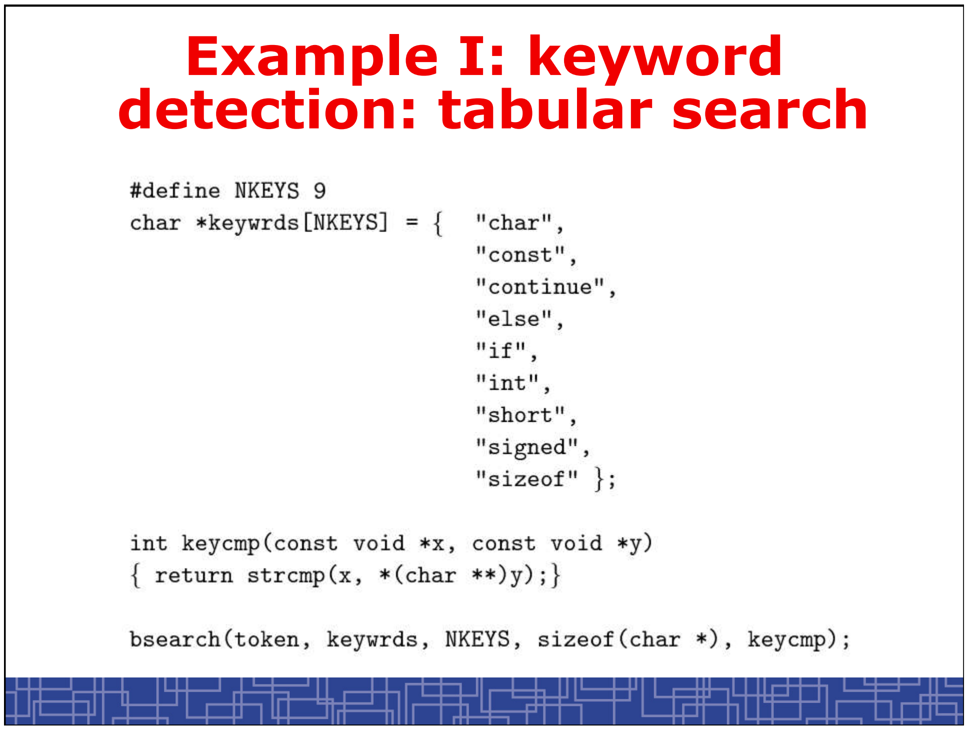
- String search/match
- String conversion/ language normalization
- Representing Language models and probabilistic grammar
- Sentence generation



Example I: keyword detection

- **C identifiers:**
  $\{char, const, continue, if, int, else, short, signed, sizeof\}$
- **Brute-force search:**

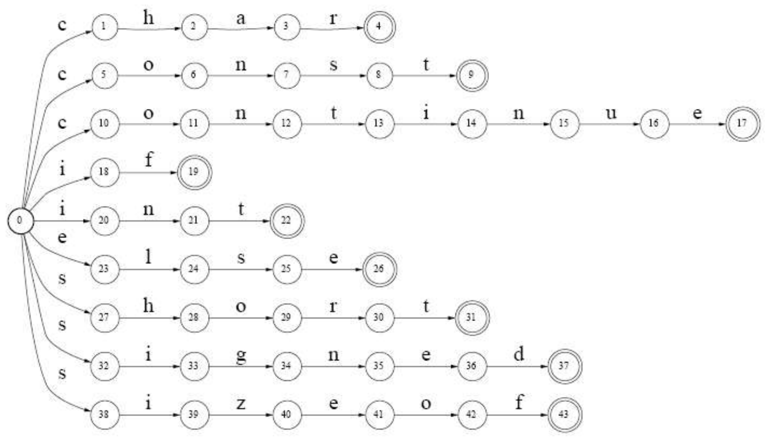
```
if(strcmp(token, "char") == 0) return 1;
if(strcmp(token, "const") == 0) return 1;
if(strcmp(token, "oontinue") == 0) return 1;
if(strcmp(token, "if") == 0) return 1;
if(strcmp(token, "int") == 0) return 1;
if(strcmp(token, "else") == 0) return 1;
if(strcmp(token, "short") == 0) return 1;
if(strcmp(token, "signed") == 0) return 1;
if(strcmp(token, "sizeof") == 0) return 1;
else return 0;
```



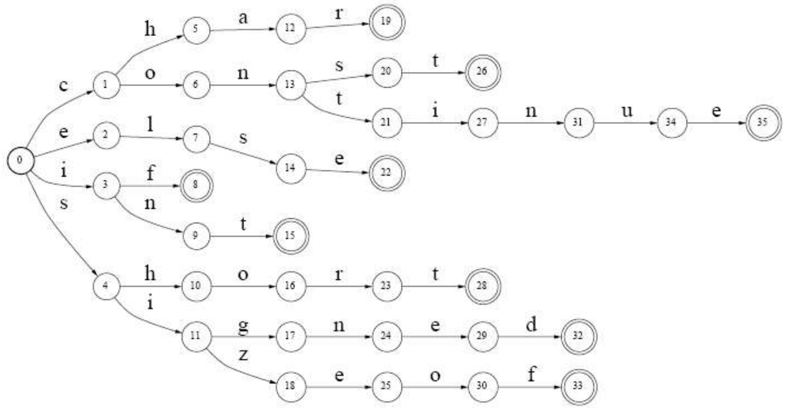
Example I: keyword detection: tabular search

```
#define NKEYS 9
char *keywrds[NKEYS] = {  "char",
                          "const",
                          "continue",
                          "else",
                          "if",
                          "int",
                          "short",
                          "signed",
                          "sizeof" };

int keycmp(const void *x, const void *y)
{ return strcmp(x, *(char **)y);}

bsearch(token, keywrds, NKEYS, sizeof(char *), keycmp);
```

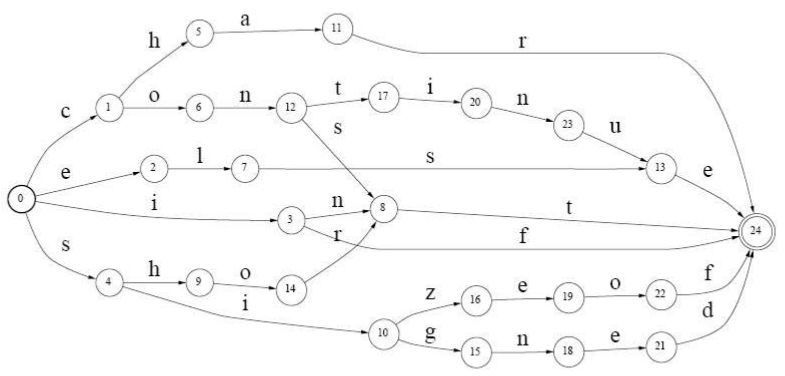
Example I: keyword detection: Automata Search



Example I: keyword detection: Deterministic Search



Example I: keyword detection: Minimal Deterministic Search



Example II: Context-dependent Phones

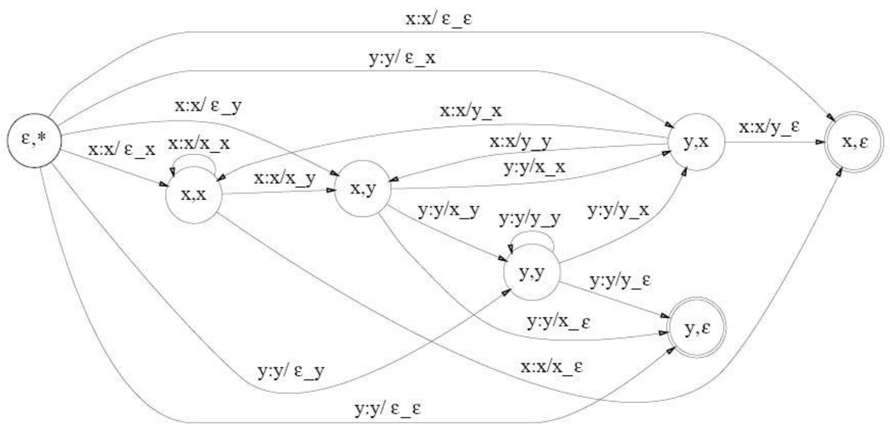
- *Monophone vs. Triphone*
- *Sentence: How do they turn out later ?*
- *Monophones: h aw d uh dh eh t er n aw t l ai t er*
- Triphones:

  <s>-h+aw h-aw+d aw-d+uh d-uh+dh uh-dh+eh …

- WFST: mapping context-independent monophones to context-dependent triphones

Example II: Context-dependent Phones

- A simple example with only two symbols x,y:



Example III: Representing Language model

- Representing language models as WFST
- Representing HMMs as WFST
- Representing overall grammar as WFST
- Come back later …