

# Incremental Construction of Minimal Finite State Automata

Jan Daciuk

Gdańsk University of Technology

e-mail: [jandac@eti.pg.gda.pl](mailto:jandac@eti.pg.gda.pl)

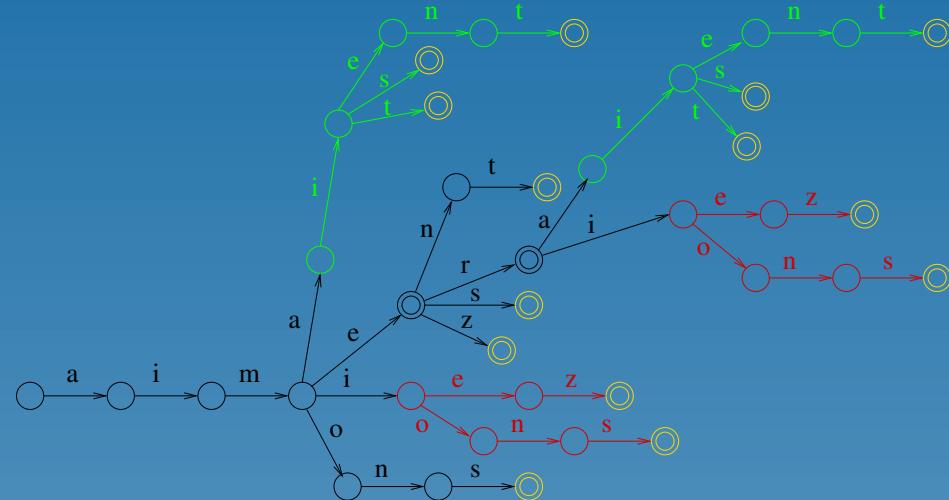
# Overview

- Incrementality and semi-incrementality
- Trie construction, minimization, synchronization
- Incremental algorithm for sorted data
- Unsorted data and confluence states
- Incremental algorithm for unsorted data
- Performance

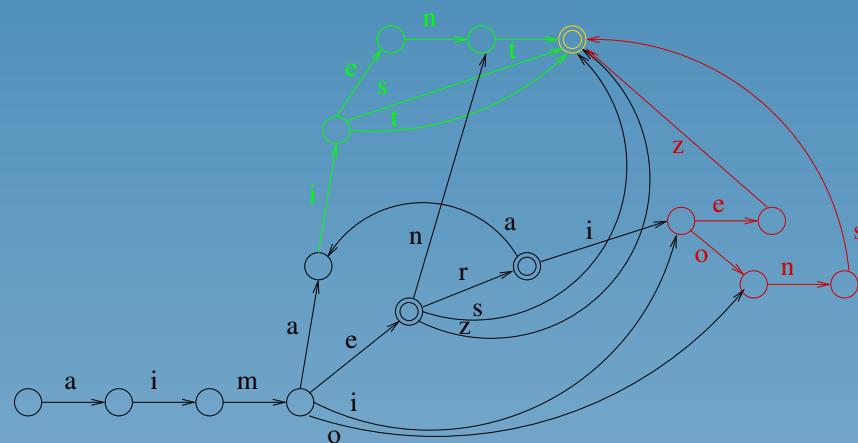
# Incrementality and automata

- Final automata
  - ◊ ideal implementation of dictionaries
  - ◊ very efficient once constructed
  - ◊ traditional construction needs much memory
- Incremental and semi-incremental construction
  - requires less memory
- Moore's law

# Traditional construction

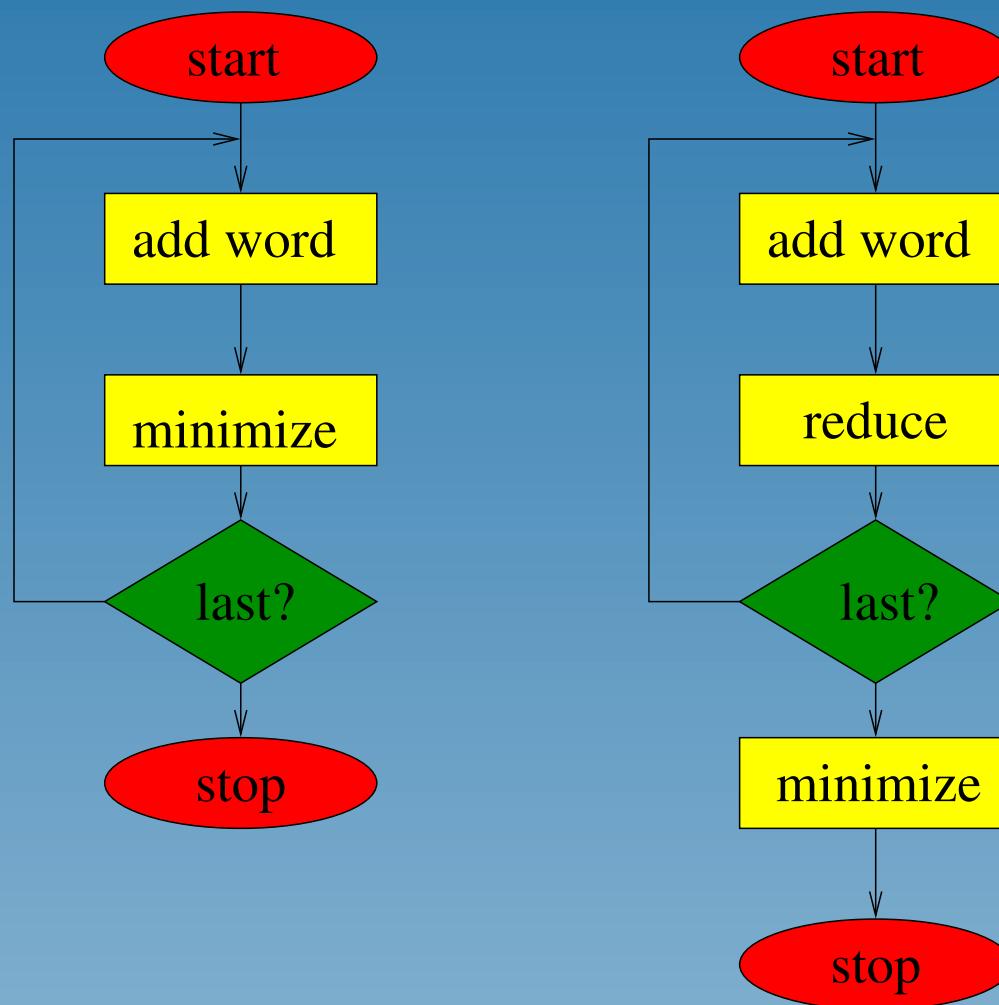


- Construct a trie



- Minimize it

# Incremental vs. semi-incremental construction algorithms



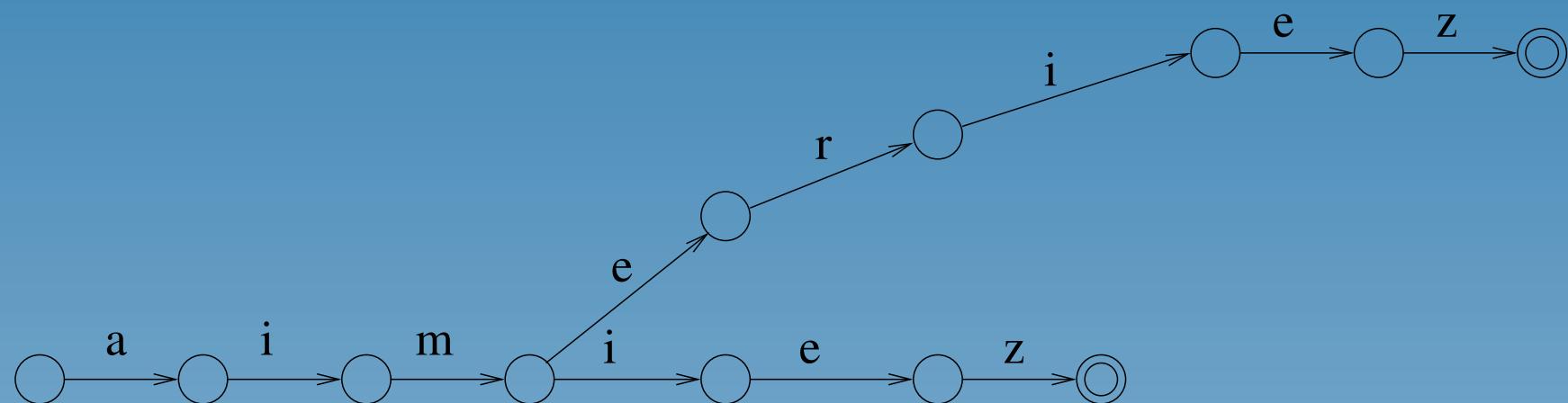
# Incremental and semi-incremental algorithms for acyclic automata

- Incremental algorithm for (lexicographically) sorted data (Daciuk, Mihov, Ciura, Deorowicz)
- Incremental algorithm for unsorted data (Aoe, Morimoto, Hase, Sgarbas, Fakotakis, Kokkinakis, Daciuk, Watson, Revuz...)
- Semi-incremental algorithm for data lexicographically sorted on reversed strings (Revuz)
- Semi-incremental algorithm for data sorted on decreasing length of strings (Watson)

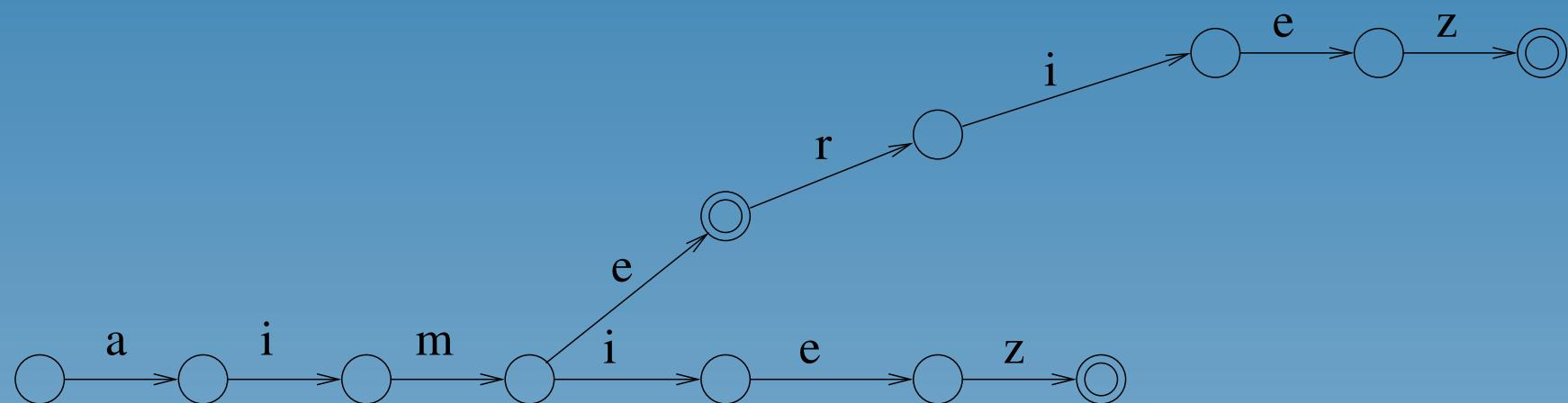
# Construction of the trie



# Construction of the trie



# Construction of the trie

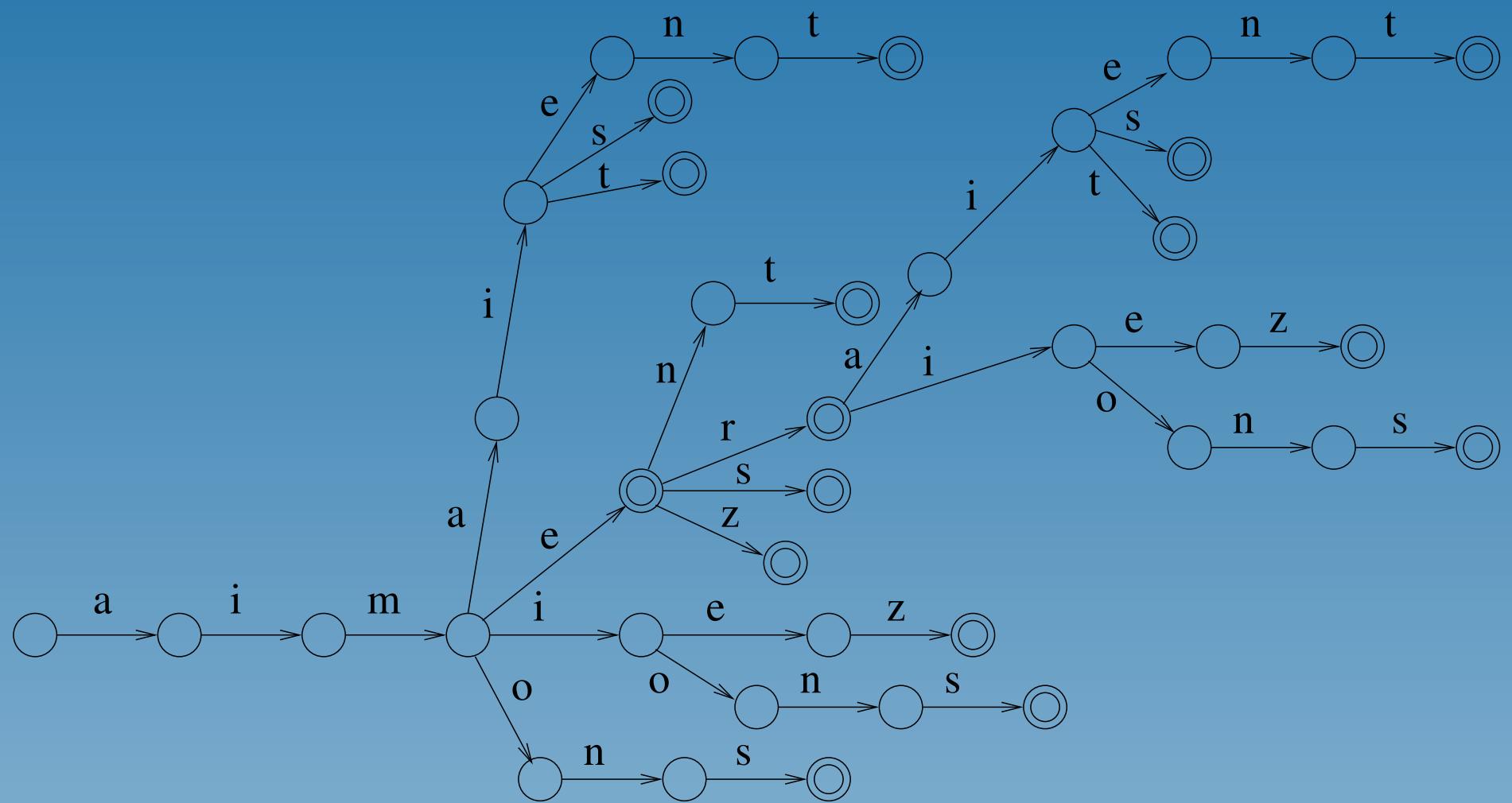


# Construction of the trie

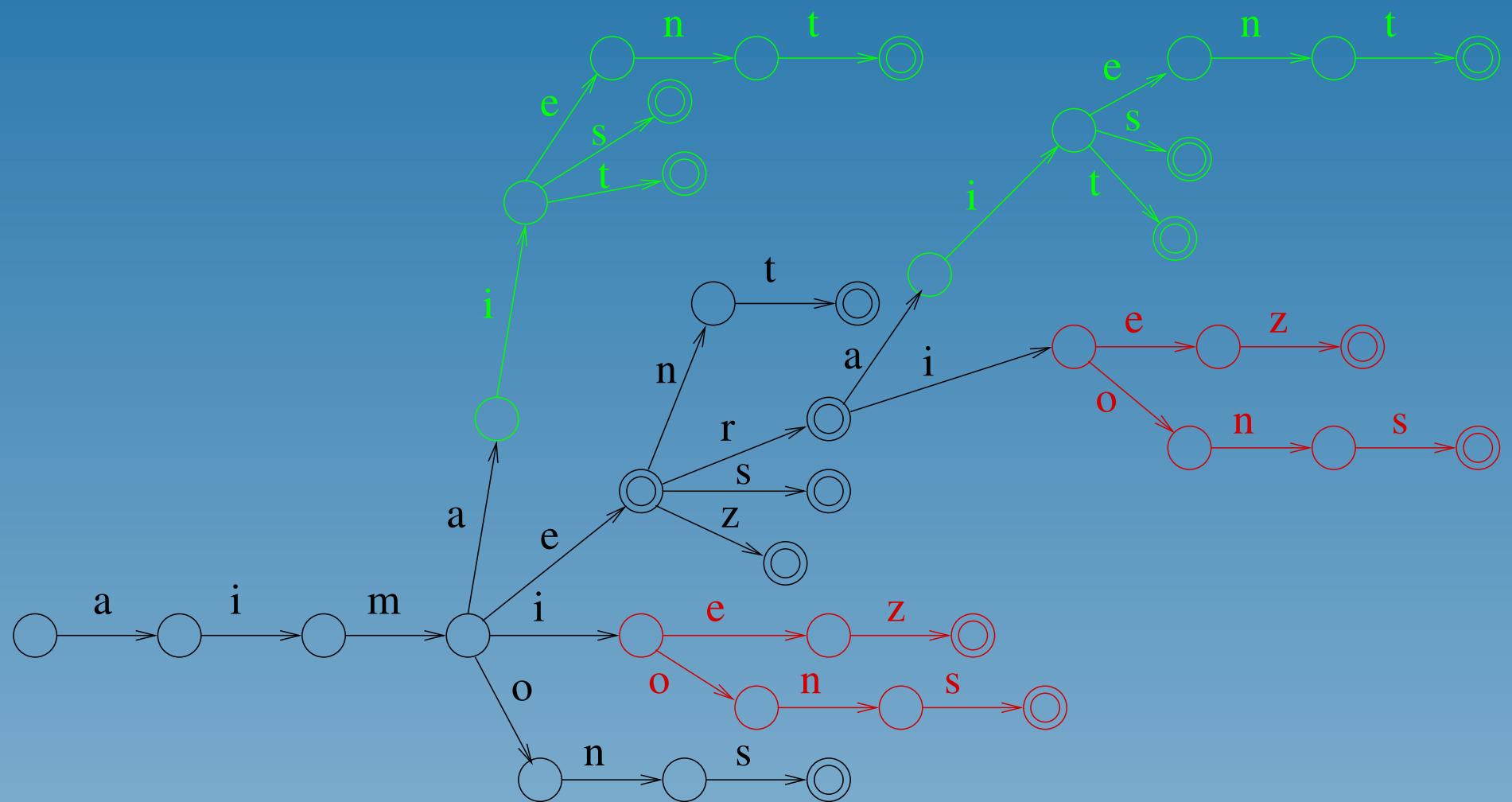
# What is actually minimization?

- $M = (Q, \Sigma, \delta, q_0, F)$ ,  $|M| = |Q|$
- $M$  is minimal iff  $\forall_{M': \mathcal{L}(M') = \mathcal{L}(M)} |M| < |M'|$
- $\vec{\mathcal{L}}(q) = \{w : \delta^*(q, w) \in F\}$ ,  $\mathcal{L}(M) = \vec{\mathcal{L}}(q_0)$
- $p \equiv q$  iff  $\vec{\mathcal{L}}(p) = \vec{\mathcal{L}}(q)$ .
- $M$  is minimal iff  $\forall_{p,q \in Q} p \equiv q \Leftrightarrow p = q$
- $\vec{\mathcal{L}}(q) = \cup_{a: \delta(q,a) \neq \perp} a \vec{\mathcal{L}}(\delta(q,a)) \cup \begin{cases} \emptyset & q \notin F \\ \epsilon & q \in F \end{cases}$

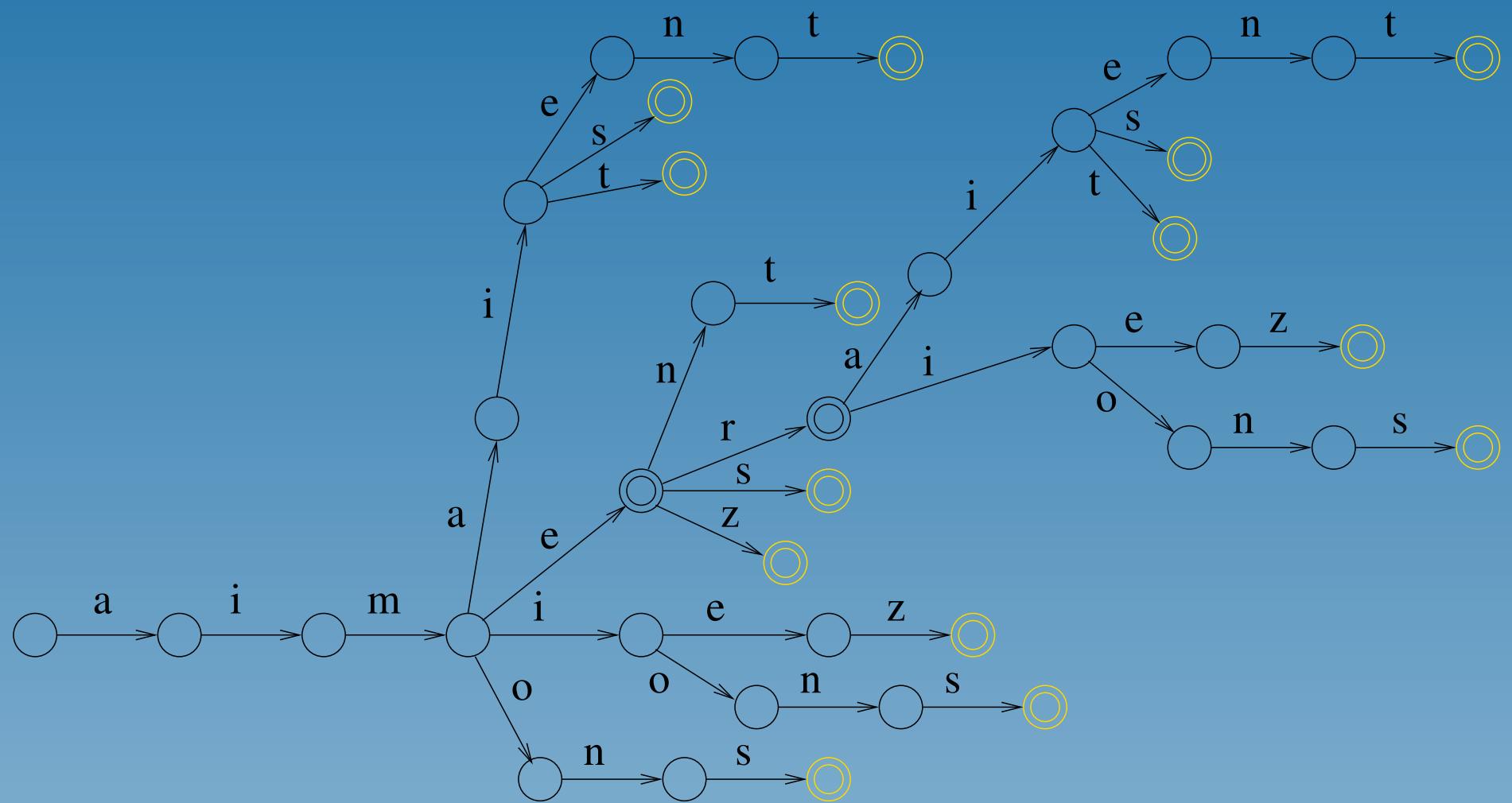
# What is minimization of a trie?



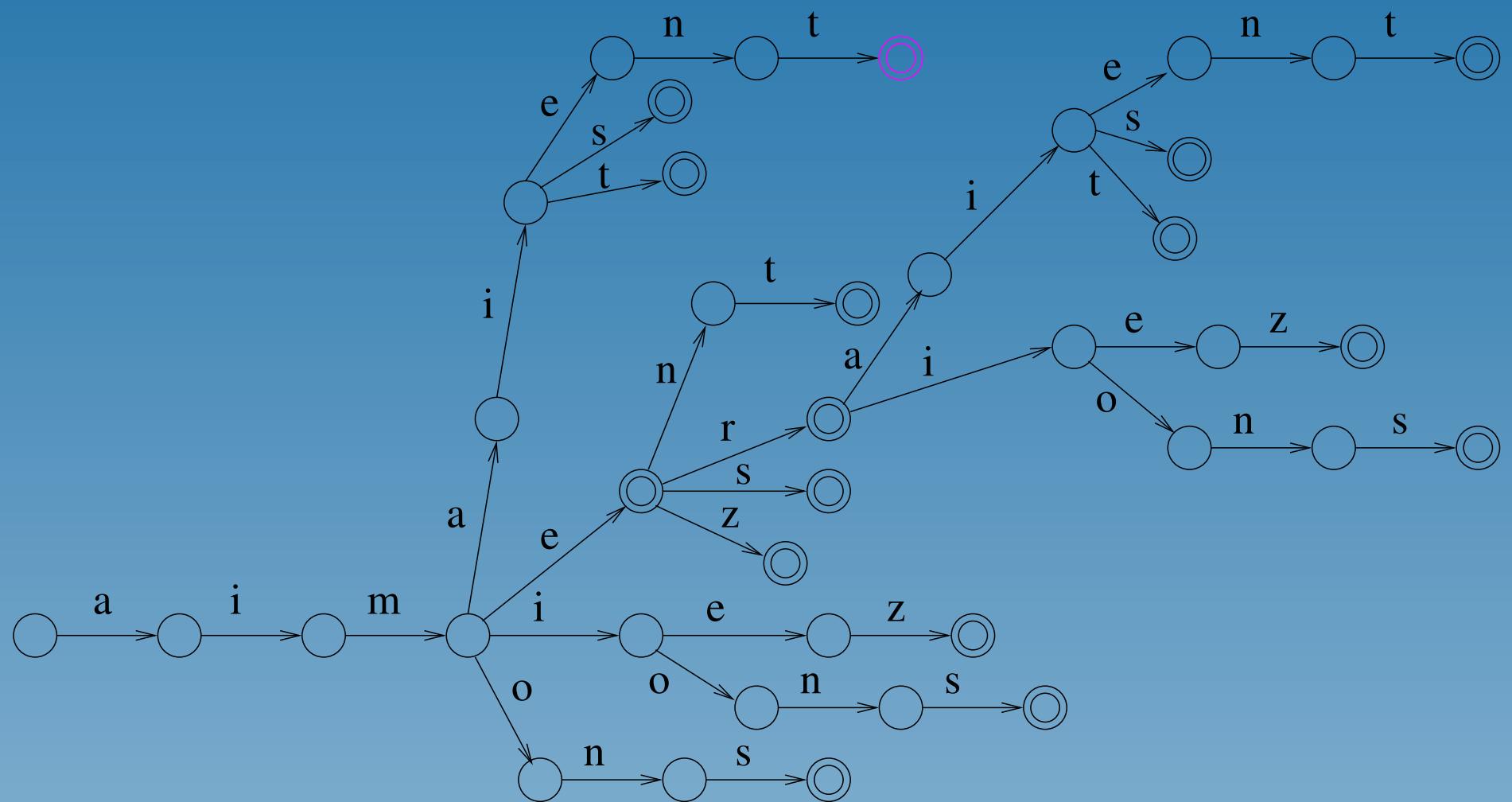
# What is minimization of a trie?



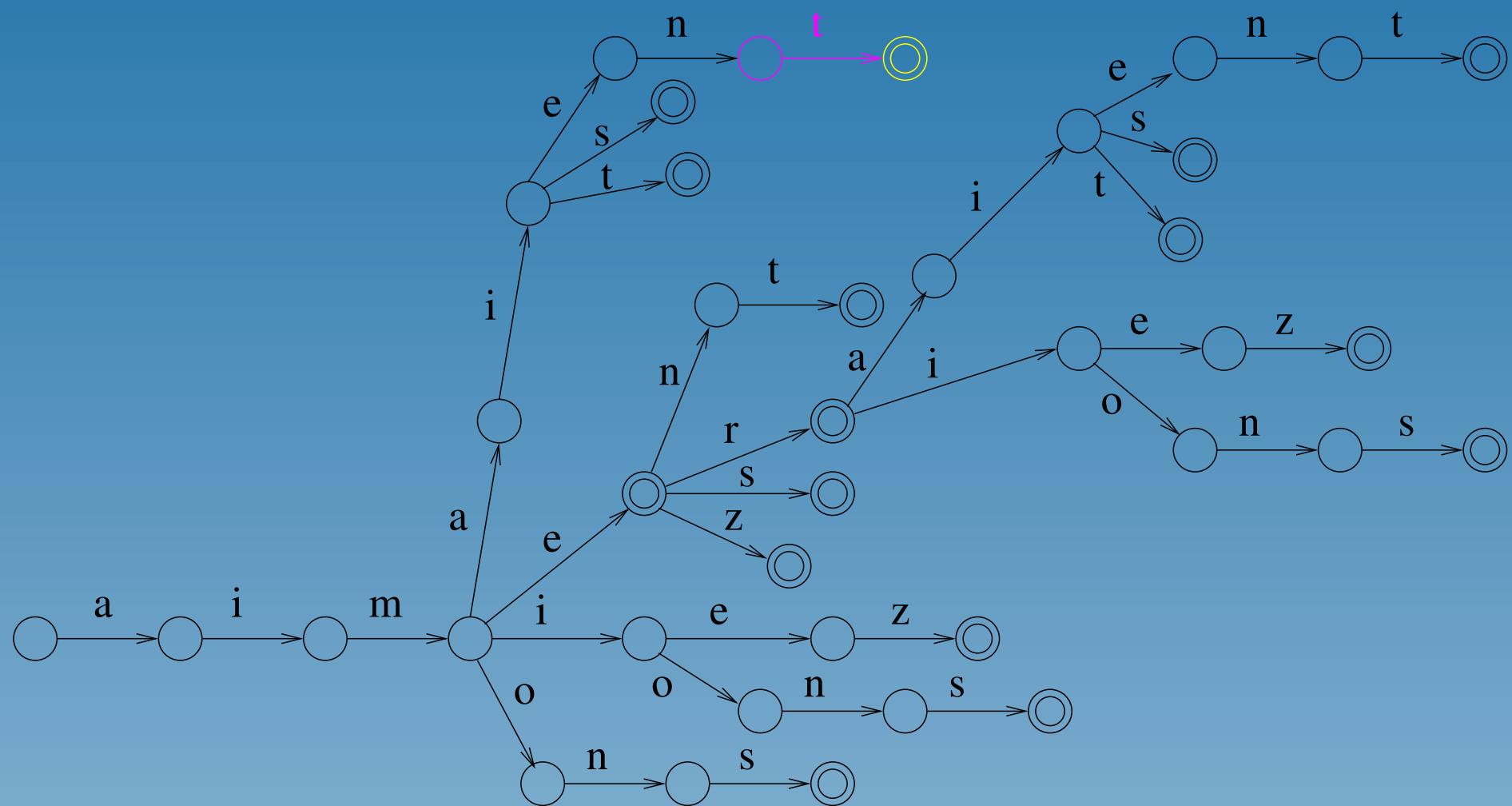
# What is minimization of a trie?



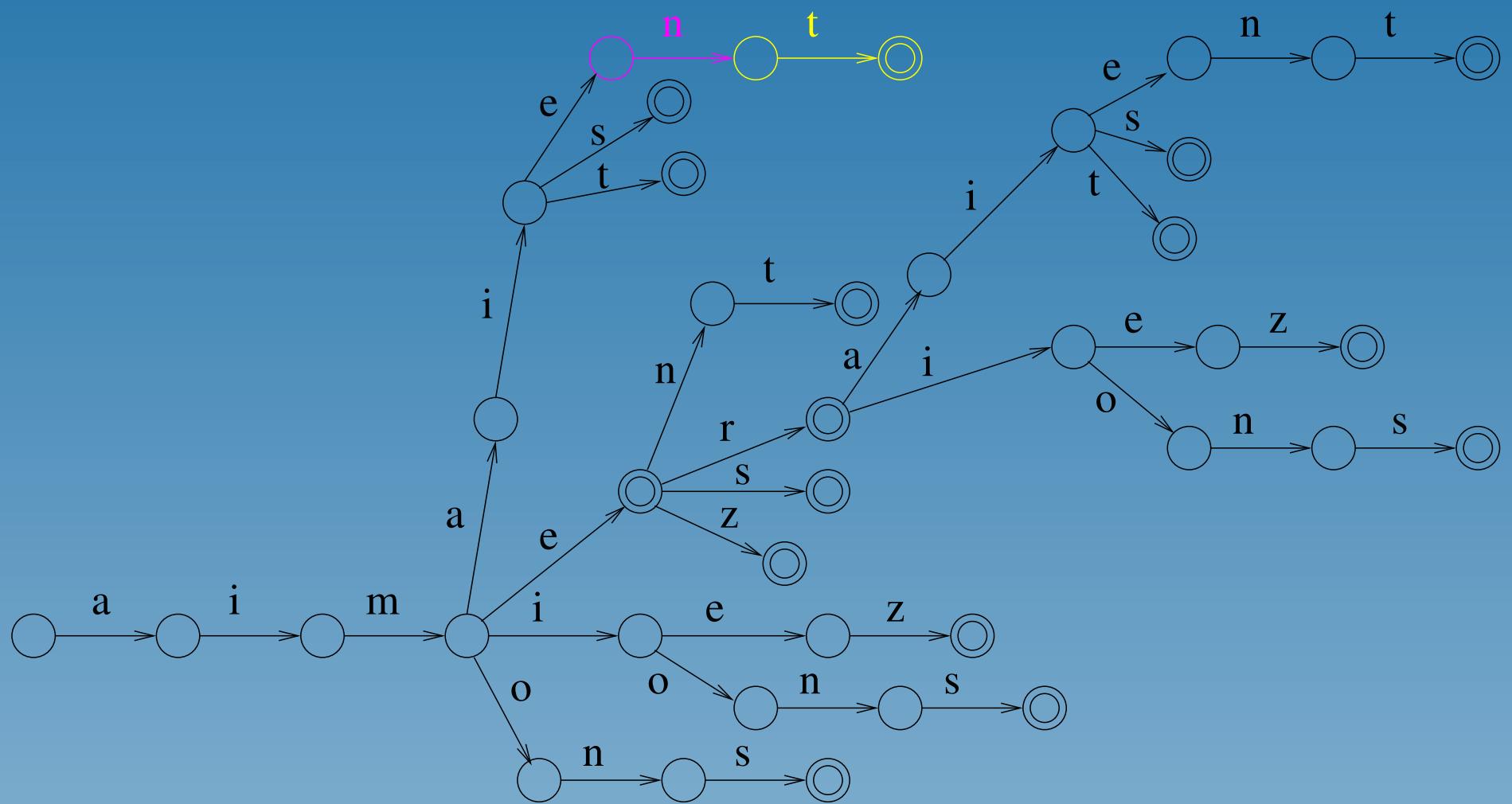
# What is minimization of a trie?



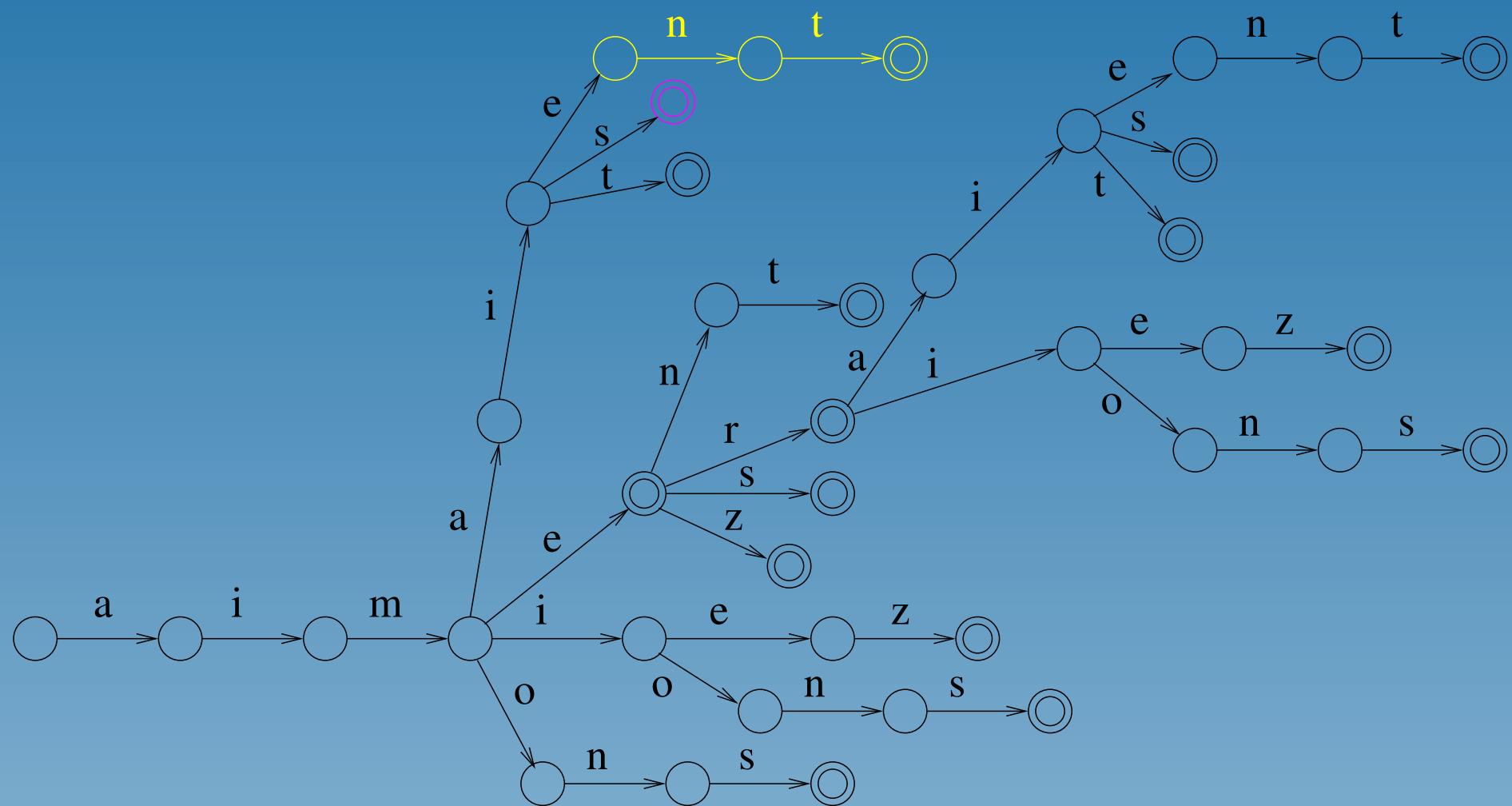
# What is minimization of a trie?



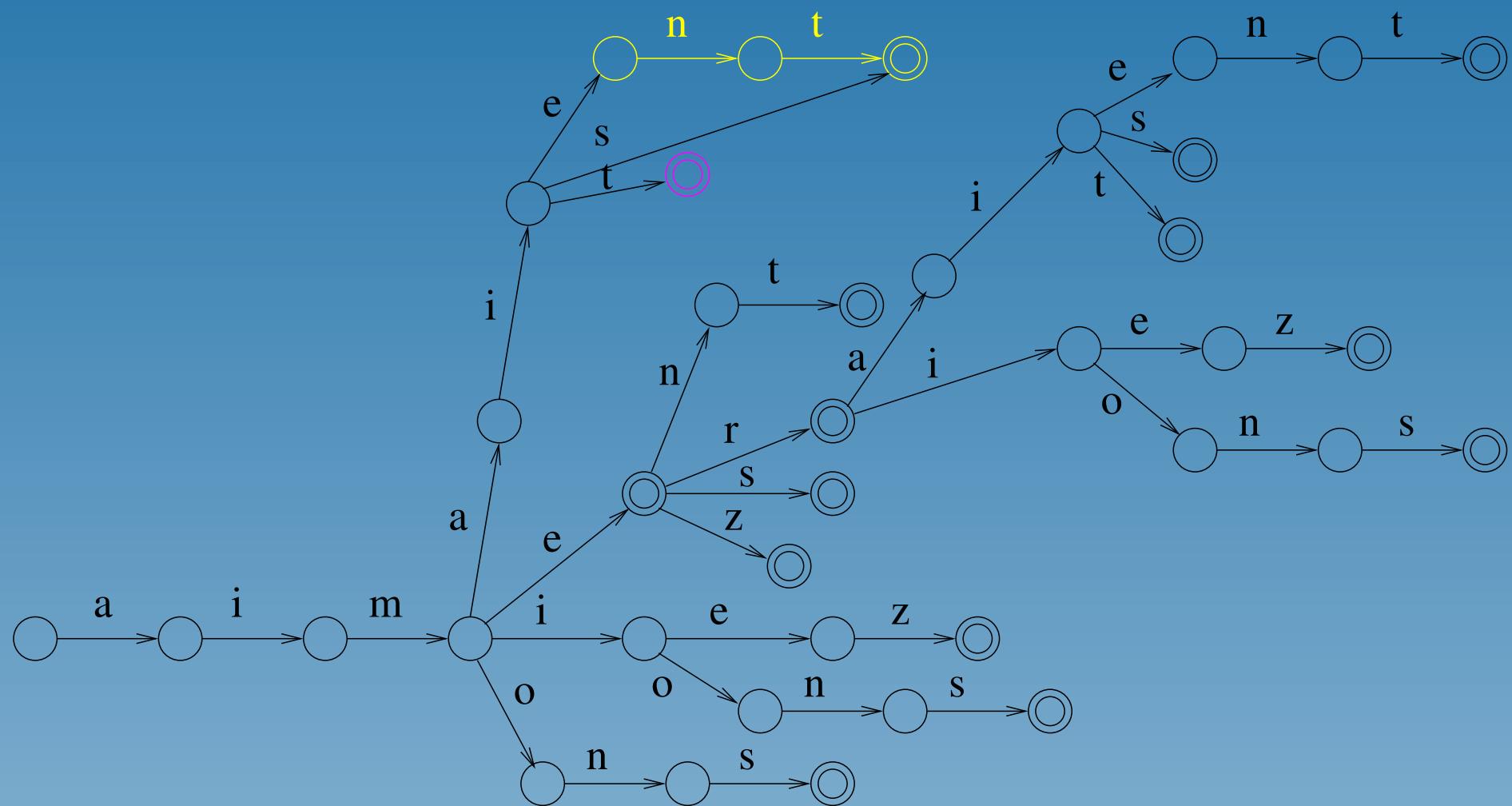
# What is minimization of a trie?



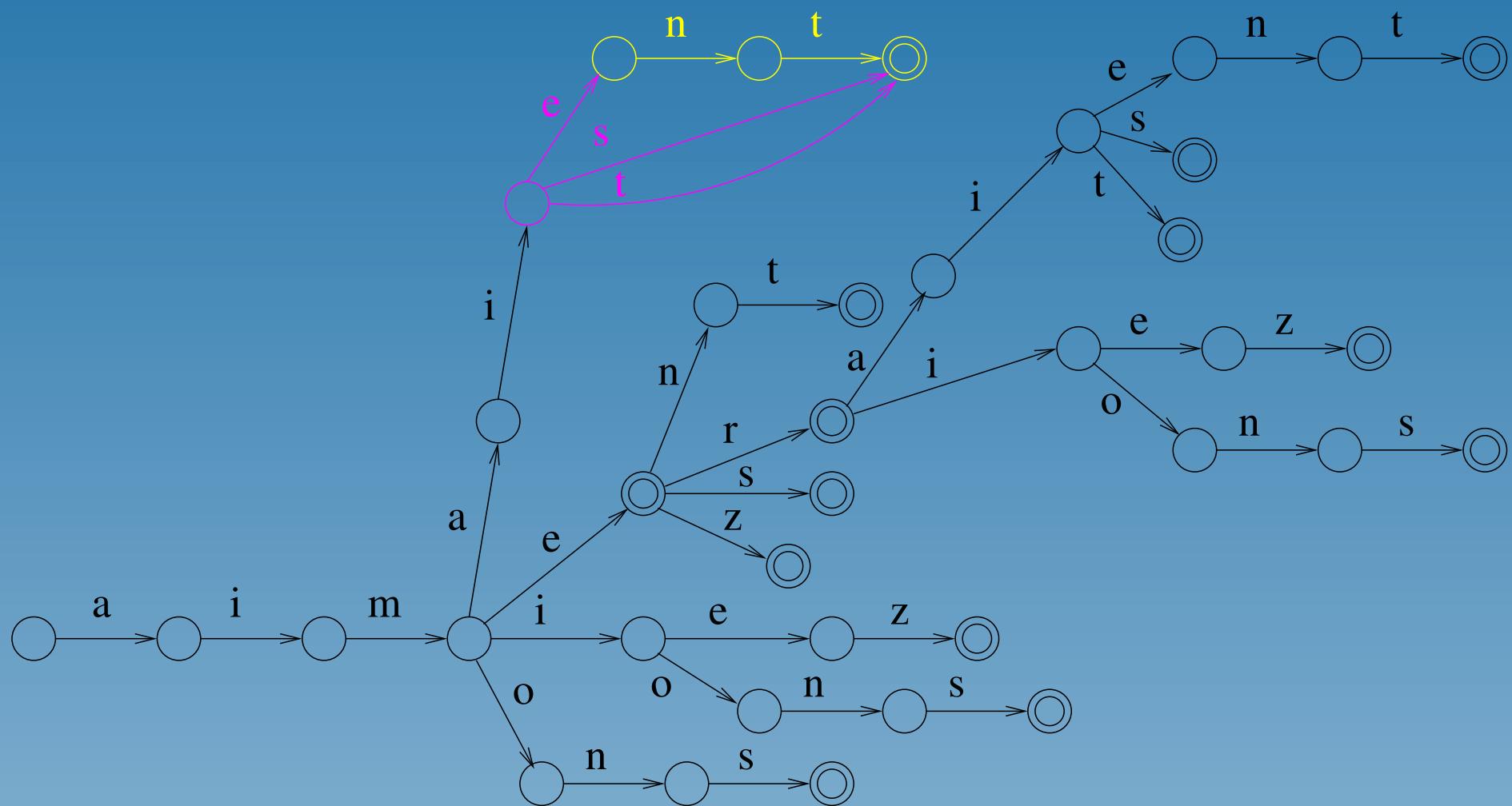
# What is minimization of a trie?



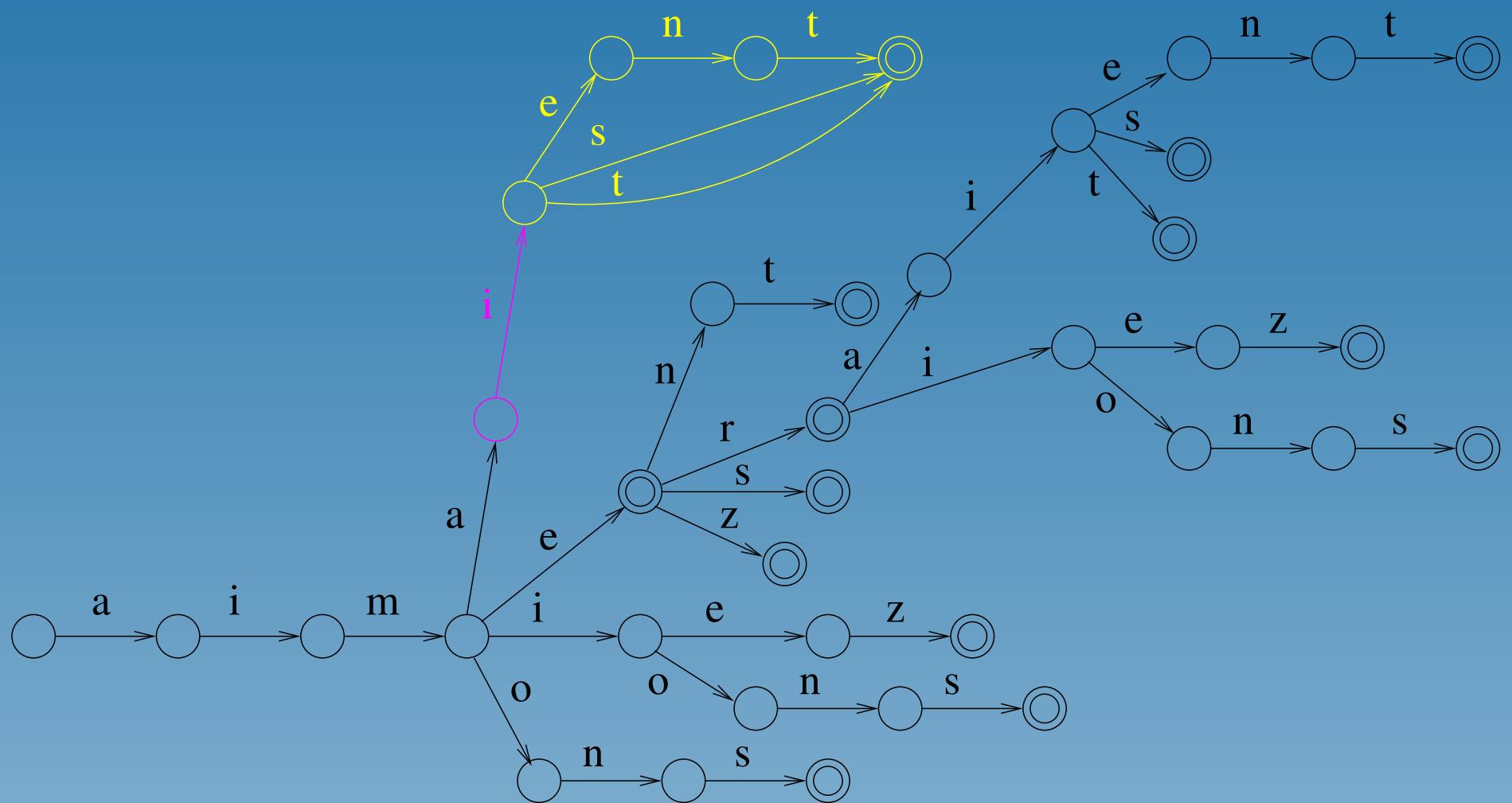
# What is minimization of a trie?



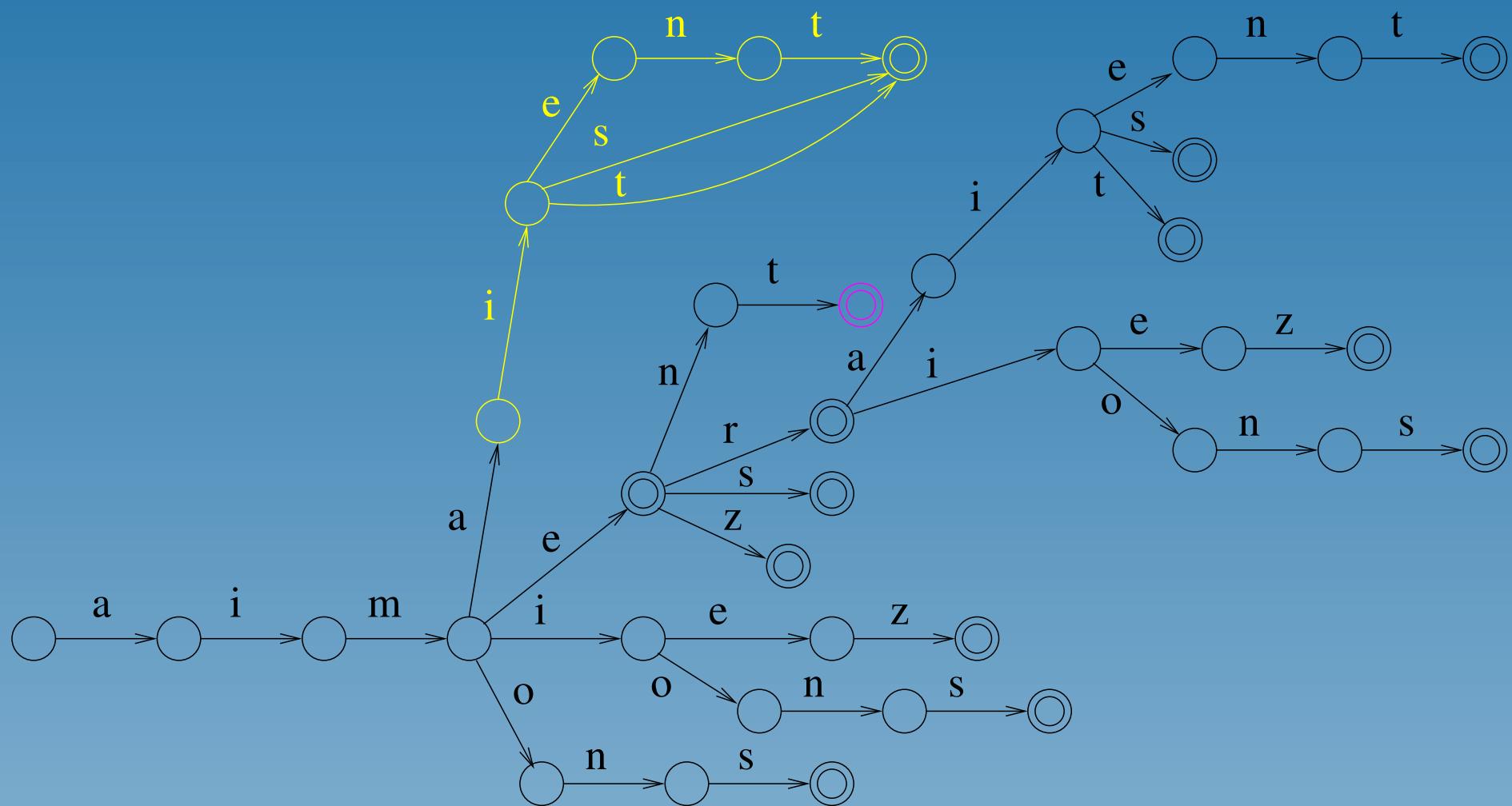
# What is minimization of a trie?



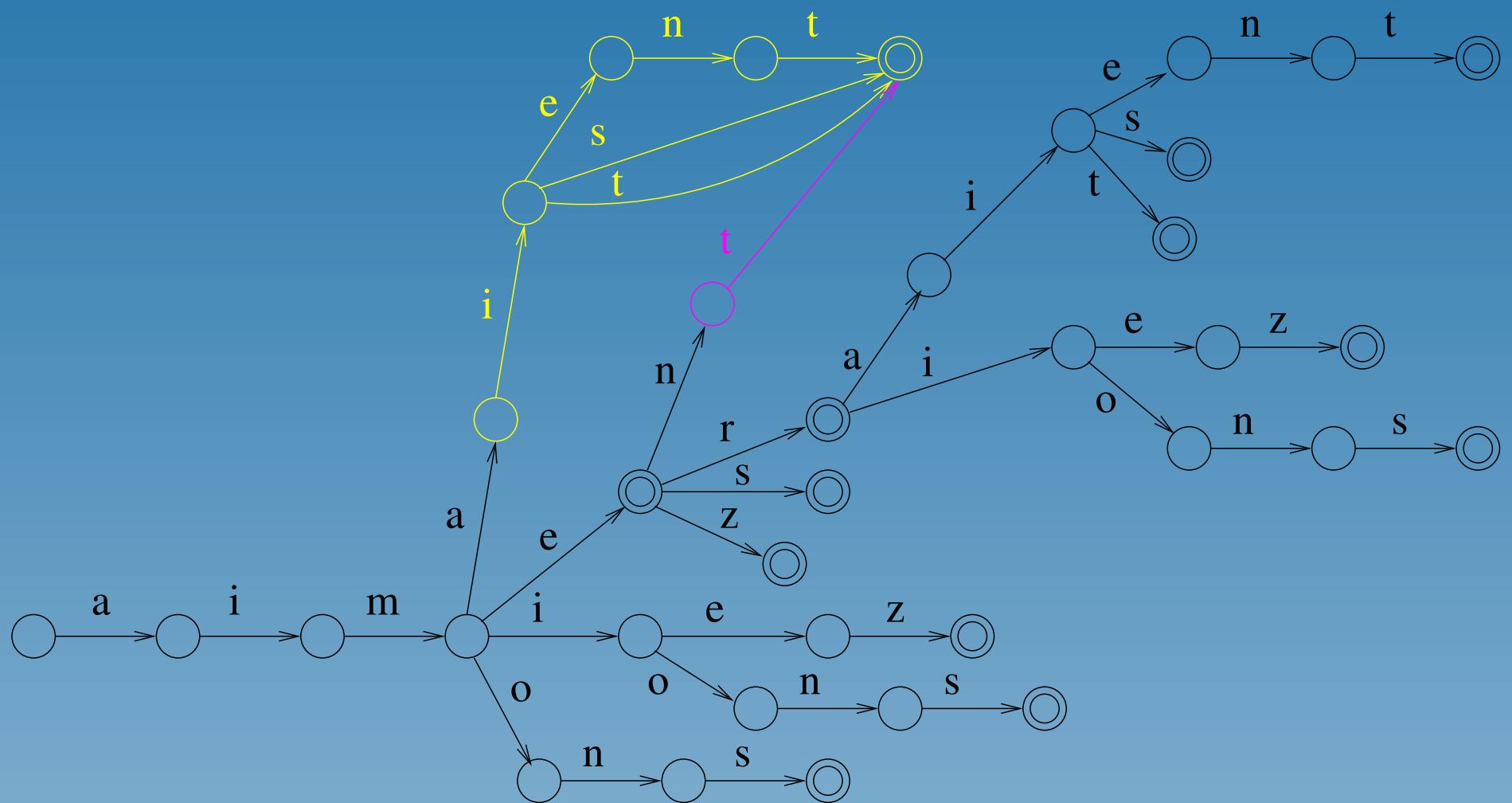
# What is minimization of a trie?



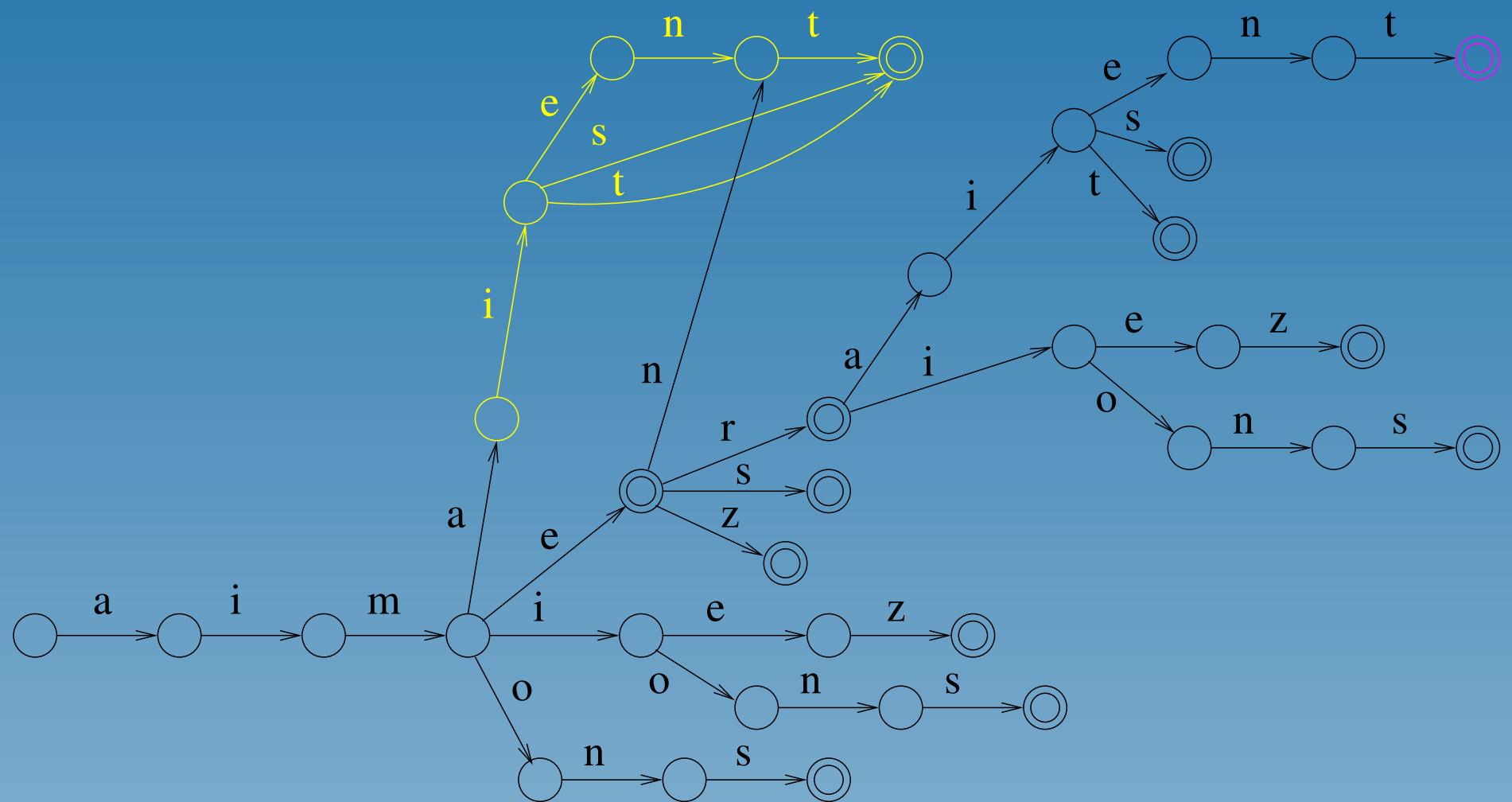
# What is minimization of a trie?



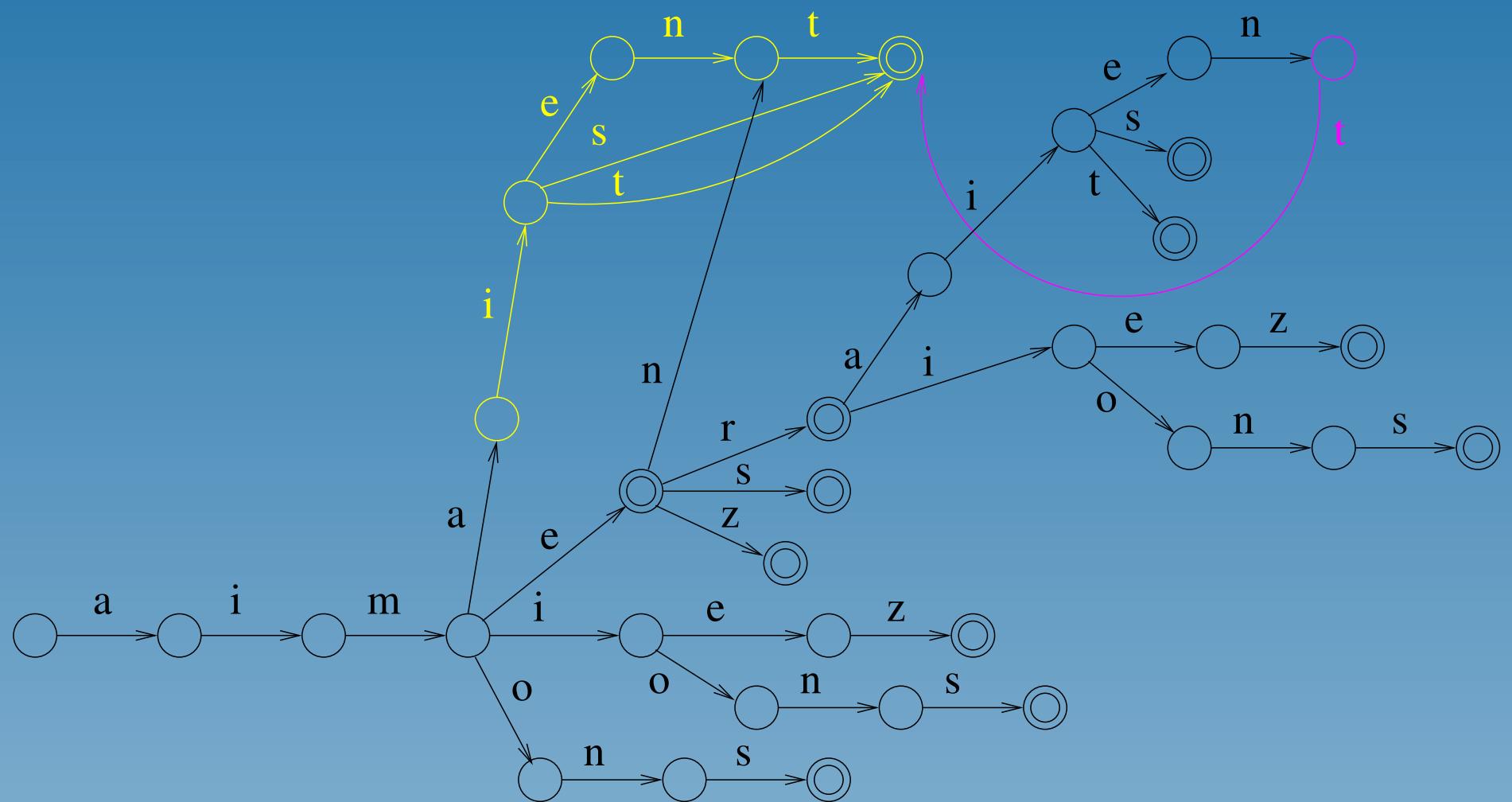
# What is minimization of a trie?



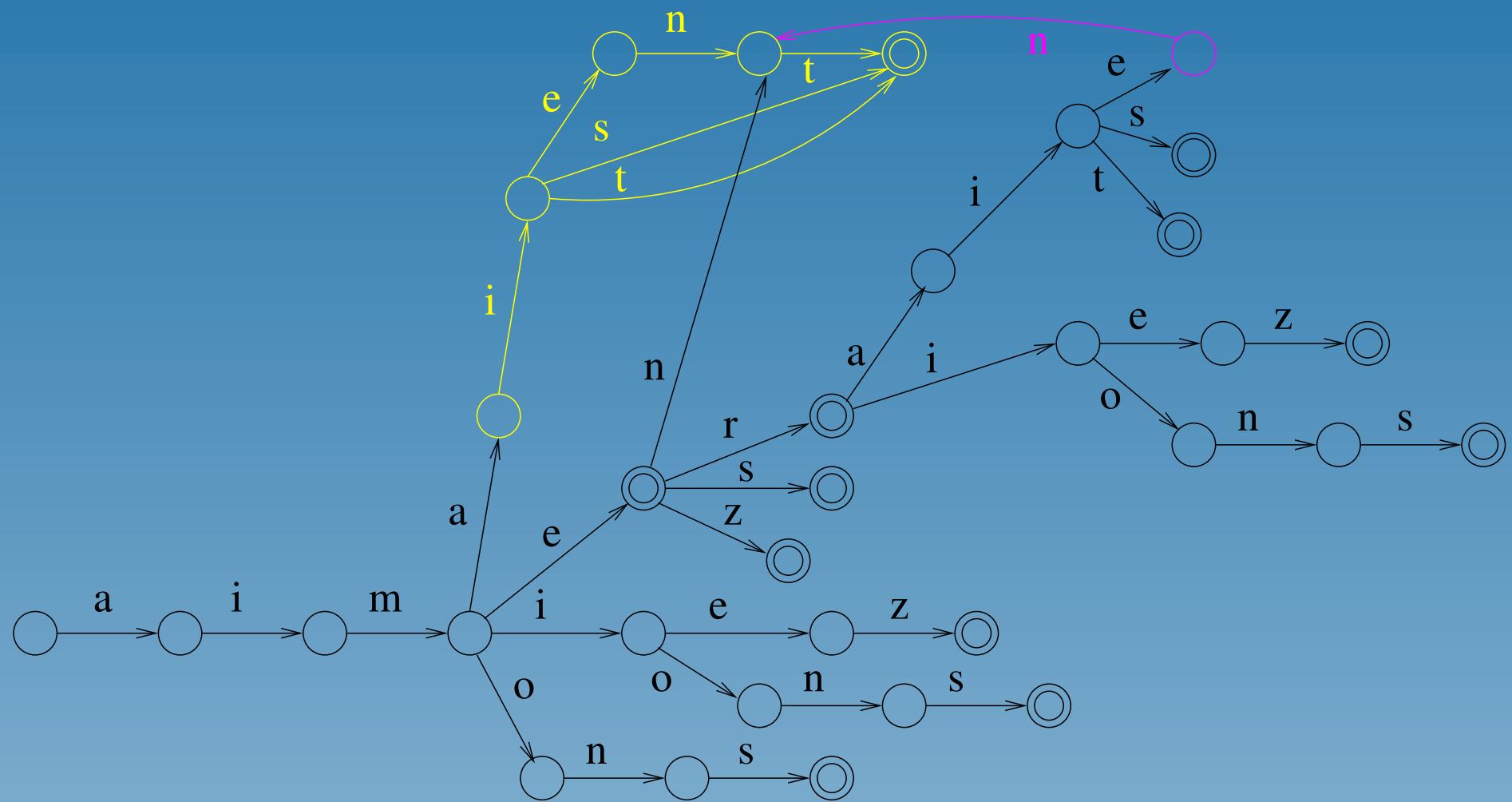
# What is minimization of a trie?



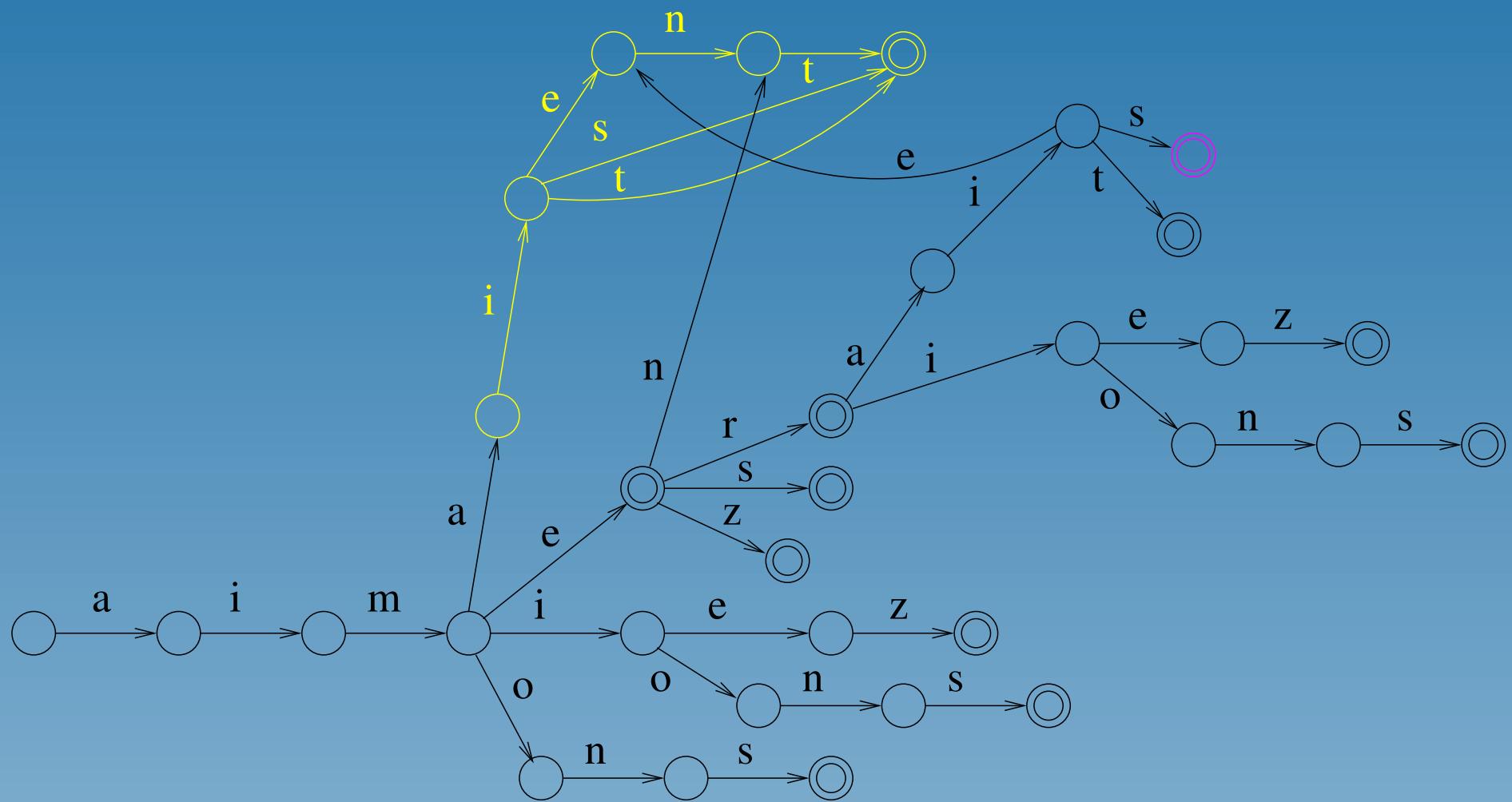
# What is minimization of a trie?



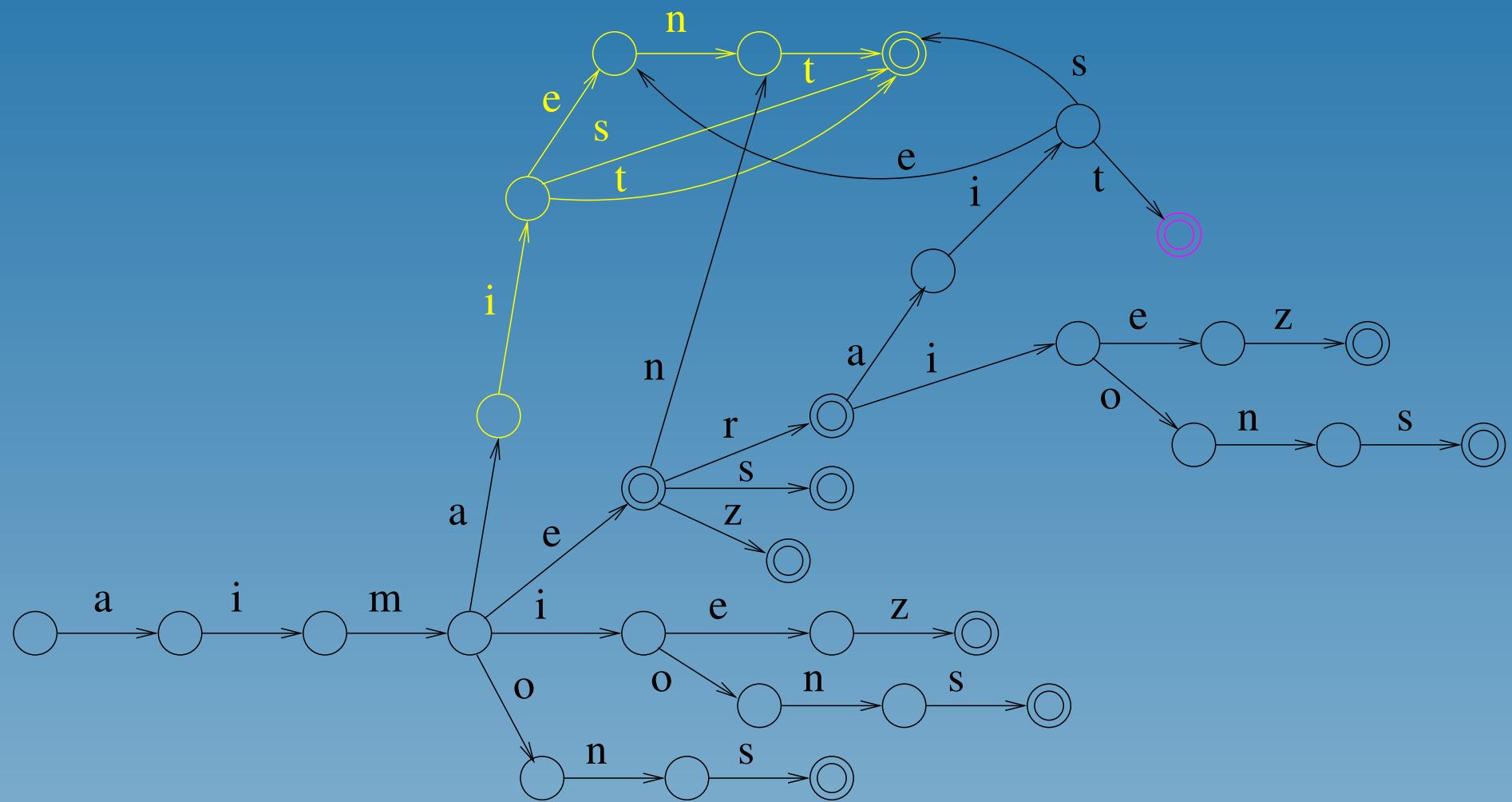
# What is minimization of a trie?



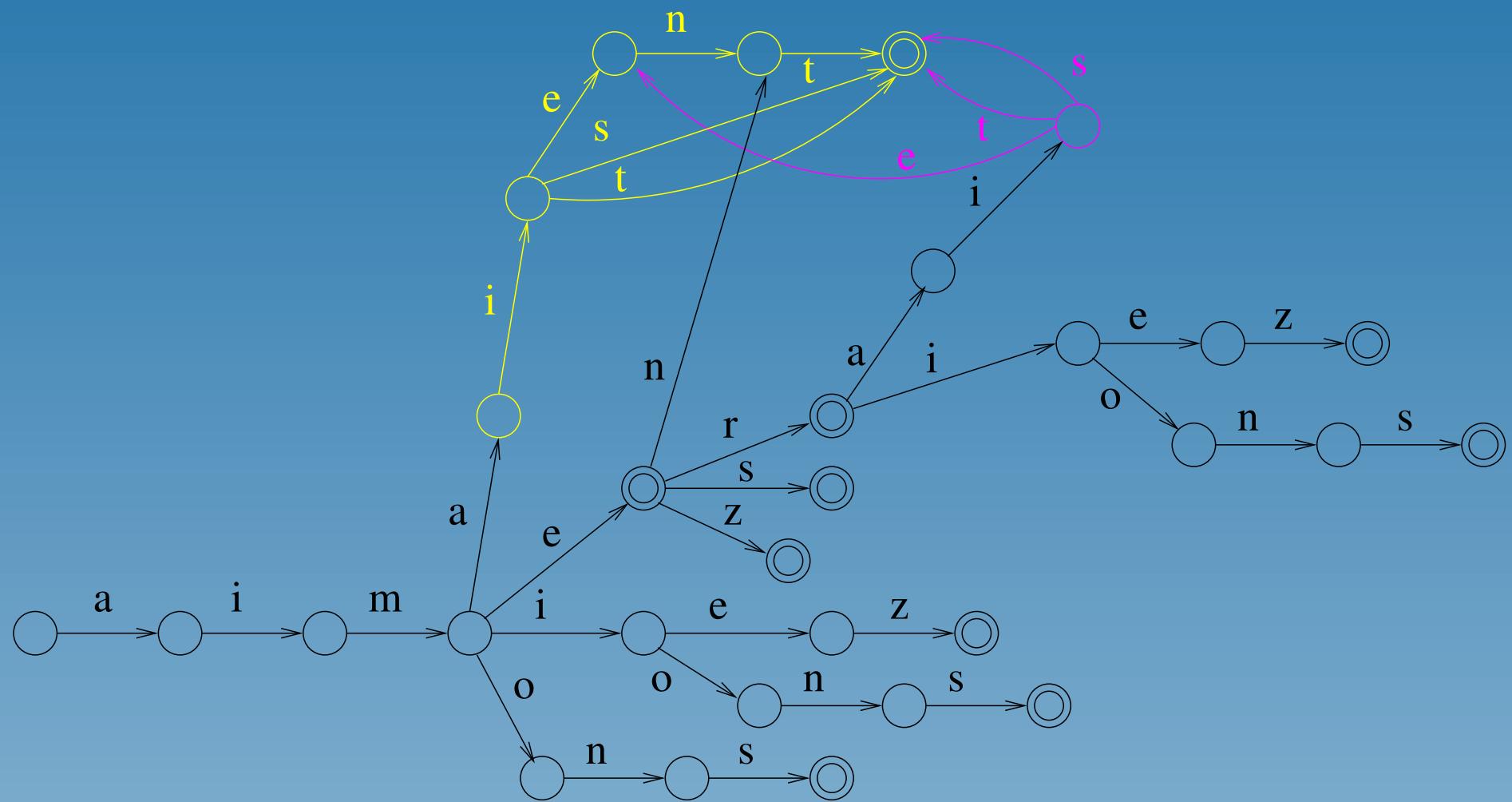
# What is minimization of a trie?



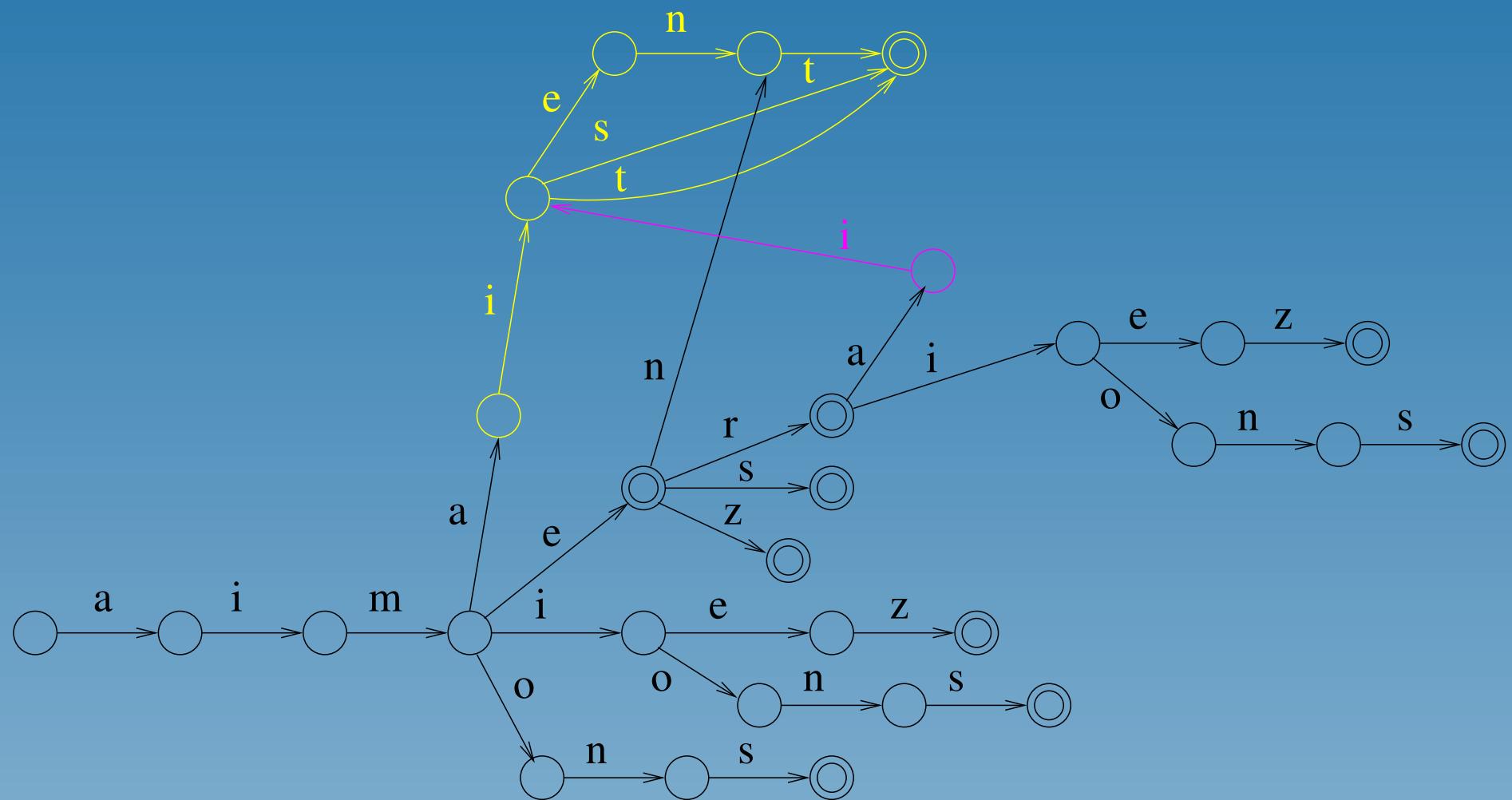
# What is minimization of a trie?



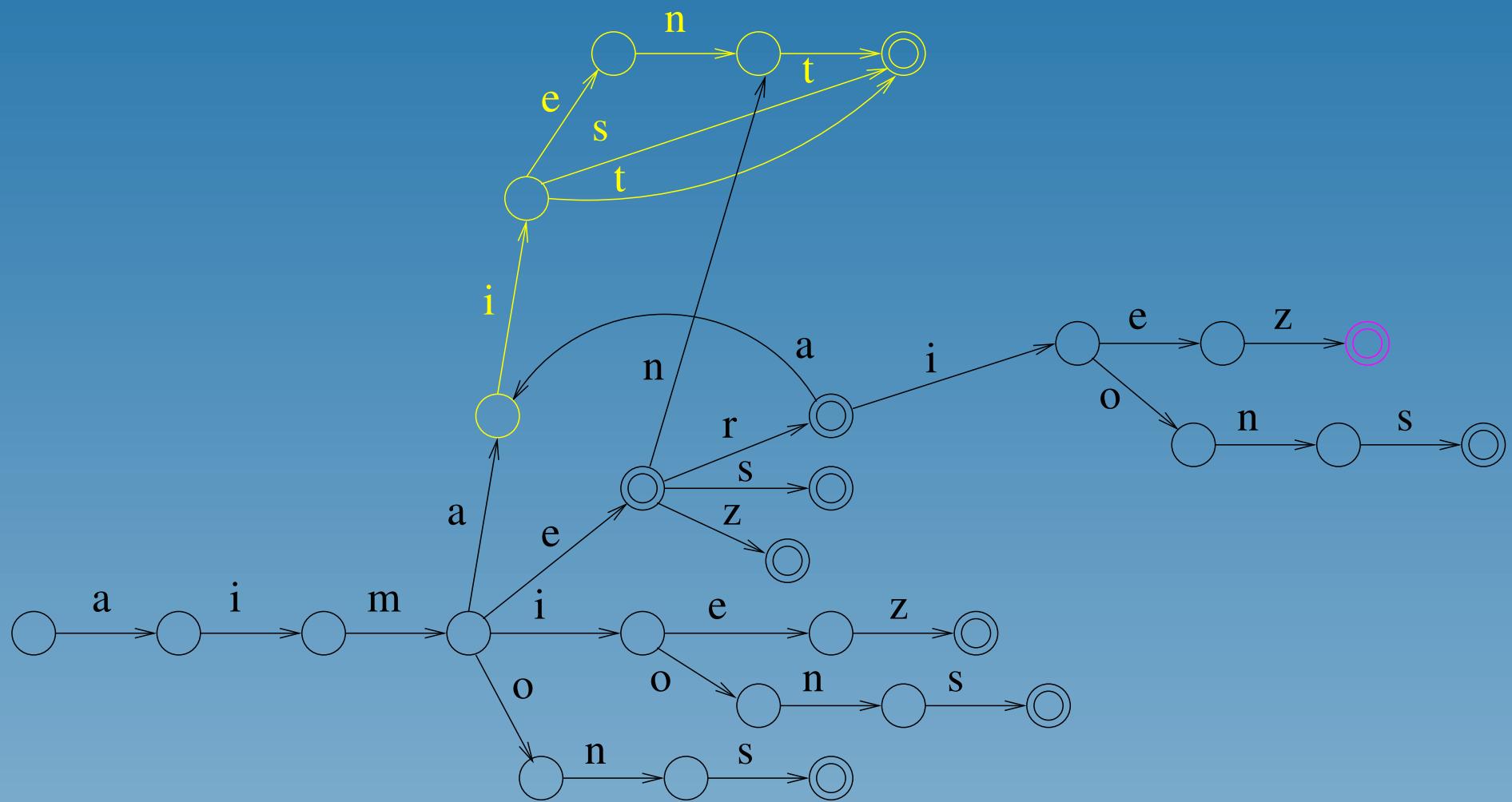
# What is minimization of a trie?



# What is minimization of a trie?



# What is minimization of a trie?



# The register

How does one check equivalence of two states?

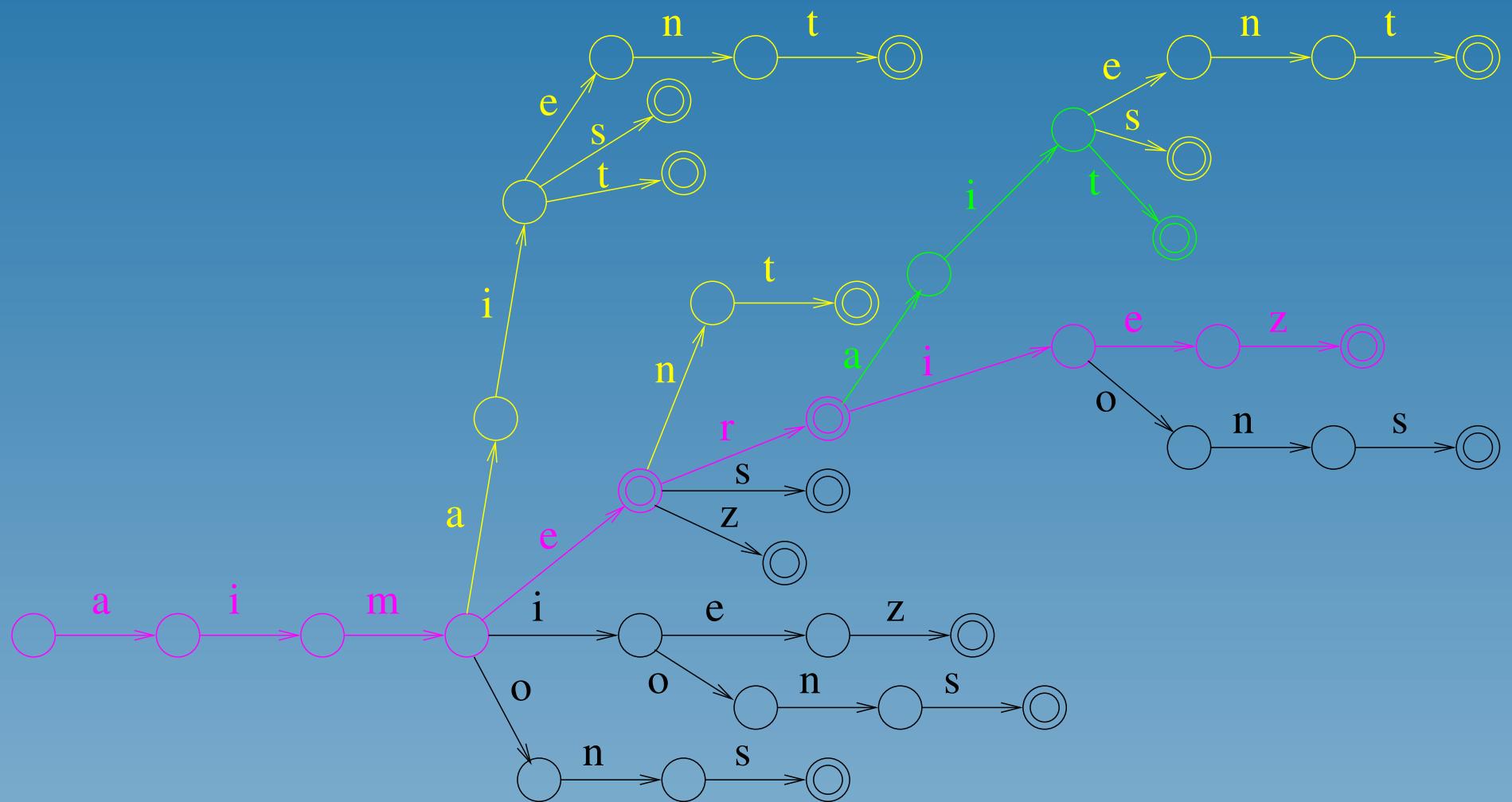
- Use the recursive definition of  $\vec{\mathcal{L}}(q)$
- Visit states using postorder, so that children have unique right languages
- Keep pointers to states with unique  $\vec{\mathcal{L}}(q)$  in a sparse table
- Use hash function on finality and transitions

Result: Operations on the register are  $\mathcal{O}(1)$

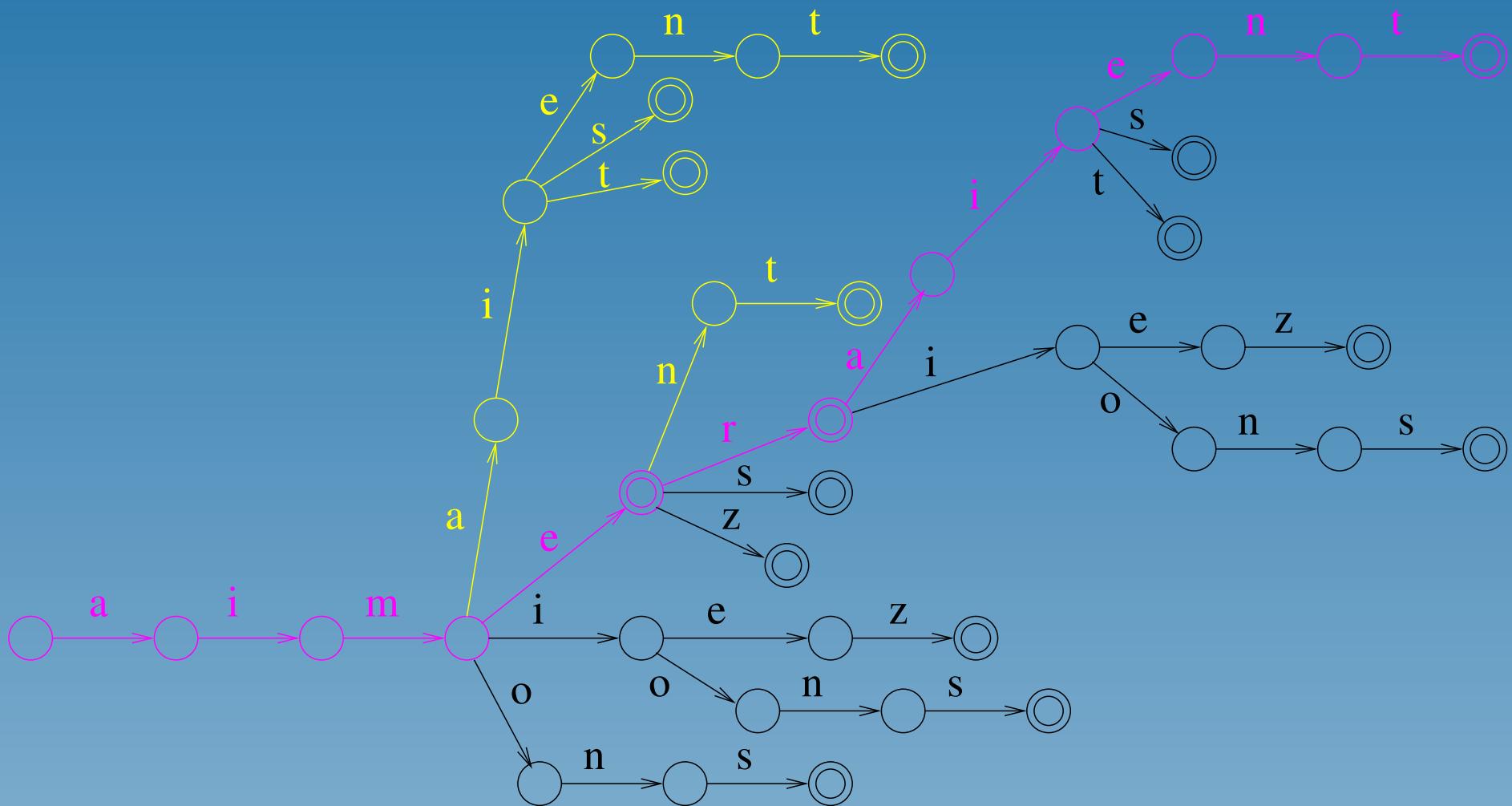
# Synchronization

- Add a word to the language of the automaton and minimize the whole automaton again
  - ◊ does not pose any restrictions on input data
  - ◊ the same states have to processed over and over again – slow
- Add a word to the language of the automaton and minimize the part that will not change in the future
  - ◊ requires data to be sorted in some way
  - ◊ faster due to processing of states only when once

# Synchronization – sorted data



# Synchronization – sorted data



# Incremental construction from sorted data

```
1: function sorted_construction;
2:    $w' \leftarrow \epsilon;$ 
3:   while input not empty do
4:      $s \leftarrow q_0; i \leftarrow 1; w \leftarrow$  next word;
5:     while  $i \leq |w|$  and  $\delta(s, w_i) != \perp$  do
6:        $s \leftarrow \delta(s, w_i); i \leftarrow i + 1;$ 
7:     end while;
8:     if  $i \leq |w'|$  then repl_or_reg( $\delta(s, w_i)$ ,  $w'_{i+1\dots|w'|}$ ); end if;
9:     while  $i \leq |w|$  do
10:       $\delta(s, w_i) \leftarrow$  new state;  $s \leftarrow \delta(s, w_i); i \leftarrow i + 1;$ 
11:    end while;
12:     $F \leftarrow F \cup \{s\}; w' \leftarrow w$ 
13:  end while;
14:  repl_or_reg( $q_0, w'$ );
15: end function;
16: function repl_or_reg( $q, v$ );
17:   if  $v \neq \epsilon$  then
18:      $\delta(q, v_1) \leftarrow$  repl_or_reg( $\delta(q, v_1)$ ,  $v_{2\dots|v|}$ );
19:   end if;
20:   if  $\exists_{r \in R} r \equiv q$  then
21:     delete  $q$ ; return  $r$ ;
22:   else
23:      $R \leftarrow R \cup \{q\}$ ; return  $q$ ;
24:   end if;
25: end function;
```

# Incremental construction from sorted data

```
1: function sorted_construction;
2:    $w' \leftarrow \epsilon;$ 
3:   while input not empty do
4:      $s \leftarrow q_0; i \leftarrow 1; w \leftarrow$  next word;
5:     while  $i \leq |w|$  and  $\delta(s, w_i) \neq \perp$  do
6:        $s \leftarrow \delta(s, w_i); i \leftarrow i + 1;$ 
7:     end while;
8:     if  $i \leq |w'|$  then repl_or_reg( $\delta(s, w_i)$ ,  $w'_{i+1\dots|w'|}$ ); end if;
9:     while  $i \leq |w|$  do
10:       $\delta(s, w_i) \leftarrow$  new state;  $s \leftarrow \delta(s, w_i); i \leftarrow i + 1;$ 
11:    end while;
12:     $F \leftarrow F \cup \{s\}; w' \leftarrow w$ 
13:  end while;
14:  repl_or_reg( $q_0, w'$ );
15: end function;
16: function repl_or_reg( $q, v$ );
17:   if  $v \neq \epsilon$  then
18:      $\delta(q, v_1) \leftarrow$  repl_or_reg( $\delta(q, v_1)$ ,  $v_{2\dots|v|}$ );
19:   end if;
20:   if  $\exists_{r \in R} r \equiv q$  then
21:     delete  $q$ ; return  $r$ ;
22:   else
23:      $R \leftarrow R \cup \{q\}$ ; return  $q$ ;
24:   end if;
25: end function;
```

# Incremental construction from sorted data

```
1: function sorted_construction;
2:    $w' \leftarrow \epsilon;$ 
3:   while input not empty do
4:      $s \leftarrow q_0; i \leftarrow 1; w \leftarrow$  next word;
5:     while  $i \leq |w|$  and  $\delta(s, w_i) = \perp$  do
6:        $s \leftarrow \delta(s, w_i); i \leftarrow i + 1;$ 
7:     end while;
8:     if  $i \leq |w'|$  then repl_or_reg( $\delta(s, w_i), w'_{i+1\dots|w'|}$ ); end if;
9:     while  $i \leq |w|$  do
10:       $\delta(s, w_i) \leftarrow$  new state;  $s \leftarrow \delta(s, w_i); i \leftarrow i + 1;$ 
11:    end while;
12:     $F \leftarrow F \cup \{s\}; w' \leftarrow w$ 
13:  end while;
14:  repl_or_reg( $q_0, w'$ );
15: end function;
16: function repl_or_reg( $q, v$ );
17:   if  $v \neq \epsilon$  then
18:      $\delta(q, v_1) \leftarrow$  repl_or_reg( $\delta(q, v_1), v_{2\dots|v|}$ );
19:   end if;
20:   if  $\exists_{r \in R} r \equiv q$  then
21:     delete  $q$ ; return  $r$ ;
22:   else
23:      $R \leftarrow R \cup \{q\}$ ; return  $q$ ;
24:   end if;
25: end function;
```

# Incremental construction from sorted data

```
1: function sorted_construction;
2:    $w' \leftarrow \epsilon$ ;
3:   while input not empty do
4:      $s \leftarrow q_0$ ;  $i \leftarrow 1$ ;  $w \leftarrow$  next word;
5:     while  $i \leq |w|$  and  $\delta(s, w_i) = \perp$  do
6:        $s \leftarrow \delta(s, w_i)$ ;  $i \leftarrow i + 1$ ;
7:     end while;
8:     if  $i \leq |w'|$  then repl_or_reg( $\delta(s, w_i)$ ,  $w'_{i+1\dots|w'|}$ ); end if;
9:     while  $i \leq |w|$  do
10:       $\delta(s, w_i) \leftarrow$  new state;  $s \leftarrow \delta(s, w_i)$ ;  $i \leftarrow i + 1$ ;
11:    end while;
12:     $F \leftarrow F \cup \{s\}$ ;  $w' \leftarrow w$ 
13:  end while;
14:  repl_or_reg( $q_0, w'$ );
15: end function;
16: function repl_or_reg( $q, v$ );
17:   if  $v \neq \epsilon$  then
18:      $\delta(q, v_1) \leftarrow$  repl_or_reg( $\delta(q, v_1)$ ,  $v_{2\dots|v|}$ );
19:   end if;
20:   if  $\exists_{r \in R} r \equiv q$  then
21:     delete  $q$ ; return  $r$ ;
22:   else
23:      $R \leftarrow R \cup \{q\}$ ; return  $q$ ;
24:   end if;
25: end function;
```

# Incremental construction from sorted data

```
1: function sorted_construction;
2:    $w' \leftarrow \epsilon$ ;
3:   while input not empty do
4:      $s \leftarrow q_0; i \leftarrow 1; w \leftarrow$  next word;
5:     while  $i \leq |w|$  and  $\delta(s, w_i) = \perp$  do
6:        $s \leftarrow \delta(s, w_i); i \leftarrow i + 1$ ;
7:     end while;
8:     if  $i \leq |w'|$  then repl_or_reg( $\delta(s, w_i)$ ,  $w'_{i+1\dots|w'|}$ ); end if;
9:     while  $i \leq |w|$  do
10:       $\delta(s, w_i) \leftarrow$  new state;  $s \leftarrow \delta(s, w_i); i \leftarrow i + 1$ ;
11:    end while;
12:     $F \leftarrow F \cup \{s\}; w' \leftarrow w$ 
13:  end while;
14:  repl_or_reg( $q_0, w'$ );
15: end function;
16: function repl_or_reg( $q, v$ );
17:   if  $v \neq \epsilon$  then
18:      $\delta(q, v_1) \leftarrow$  repl_or_reg( $\delta(q, v_1)$ ,  $v_{2\dots|v|}$ );
19:   end if;
20:   if  $\exists_{r \in R} r \equiv q$  then
21:     delete  $q$ ; return  $r$ ;
22:   else
23:      $R \leftarrow R \cup \{q\}$ ; return  $q$ ;
24:   end if;
25: end function;
```

# Incremental construction from sorted data

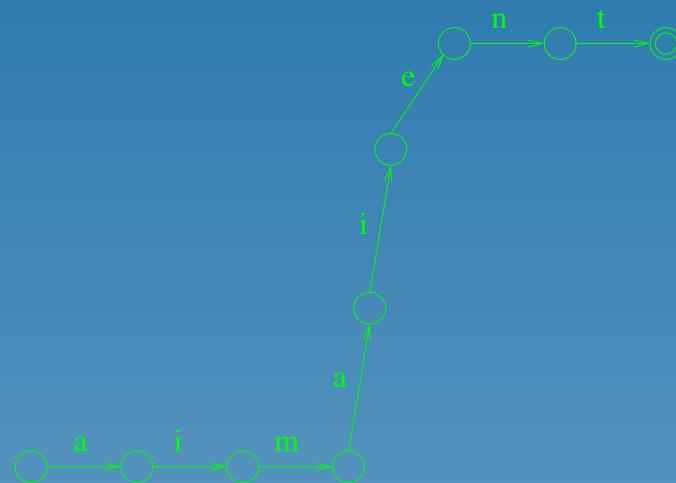
```
1: function sorted_construction;
2:    $w' \leftarrow \epsilon$ ;
3:   while input not empty do
4:      $s \leftarrow q_0; i \leftarrow 1; w \leftarrow$  next word;
5:     while  $i \leq |w|$  and  $\delta(s, w_i) != \perp$  do
6:        $s \leftarrow \delta(s, w_i); i \leftarrow i + 1$ ;
7:     end while;
8:     if  $i \leq |w'|$  then repl_or_reg( $\delta(s, w_i)$ ,  $w'_{i+1\dots|w'|}$ ); end if;
9:     while  $i \leq |w|$  do
10:       $\delta(s, w_i) \leftarrow$  new state;  $s \leftarrow \delta(s, w_i); i \leftarrow i + 1$ ;
11:    end while;
12:     $F \leftarrow F \cup \{s\}; w' \leftarrow w$ 
13:  end while;
14:  repl_or_reg( $q_0, w'$ );
15: end function;
16: function repl_or_reg( $q, v$ );
17:   if  $v \neq \epsilon$  then
18:      $\delta(q, v_1) \leftarrow$  repl_or_reg( $\delta(q, v_1)$ ,  $v_{2\dots|v|}$ );
19:   end if;
20:   if  $\exists_{r \in R} r \equiv q$  then
21:     delete  $q$ ; return  $r$ ;
22:   else
23:      $R \leftarrow R \cup \{q\}$ ; return  $q$ ;
24:   end if;
25: end function;
```

# Incremental construction from sorted data

```
1: function sorted_construction;
2:    $w' \leftarrow \epsilon$ ;
3:   while input not empty do
4:      $s \leftarrow q_0; i \leftarrow 1; w \leftarrow$  next word;
5:     while  $i \leq |w|$  and  $\delta(s, w_i) != \perp$  do
6:        $s \leftarrow \delta(s, w_i); i \leftarrow i + 1$ ;
7:     end while;
8:     if  $i \leq |w'|$  then repl_or_reg( $\delta(s, w_i)$ ,  $w'_{i+1\dots|w'|}$ ); end if;
9:     while  $i \leq |w|$  do
10:       $\delta(s, w_i) \leftarrow$  new state;  $s \leftarrow \delta(s, w_i); i \leftarrow i + 1$ ;
11:    end while;
12:     $F \leftarrow F \cup \{s\}; w' \leftarrow w$ 
13:  end while;
14:  repl_or_reg( $q_0, w'$ );
15: end function;
16: function repl_or_reg( $q, v$ );
17:   if  $v \neq \epsilon$  then
18:      $\delta(q, v_1) \leftarrow$  repl_or_reg( $\delta(q, v_1)$ ,  $v_{2\dots|v|}$ );
19:   end if;
20:   if  $\exists_{r \in R} r \equiv q$  then
21:     delete  $q$ ; return  $r$ ;
22:   else
23:      $R \leftarrow R \cup \{q\}$ ; return  $q$ ;
24:   end if;
25: end function;
```

# Incremental construction from sorted data – examples

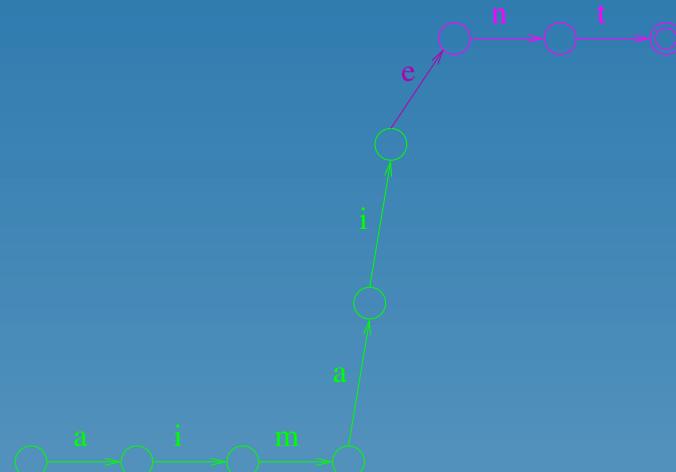
```
1: function sorted_construction;
2:    $w' \leftarrow \epsilon;$ 
3:   while input not empty do
4:      $s \leftarrow q_0; i \leftarrow 1; w \leftarrow \text{next word};$ 
5:     while  $i \leq |w|$  and  $\delta(s, w_i) = \perp$  do
6:        $s \leftarrow \delta(s, w_i); i \leftarrow i + 1;$ 
7:     end while;
8:     if  $i \leq |w'|$  then
9:       repl_or_reg( $\delta(s, w_i)$ ,  $w'_{i+1 \dots |w'|}$ );
10:      end if;
11:      while  $i \leq |w|$  do
12:         $\delta(s, w_i) \leftarrow \text{new state};$ 
13:         $s \leftarrow \delta(s, w_i); i \leftarrow i + 1;$ 
14:      end while;
15:       $F \leftarrow F \cup \{s\}; w' \leftarrow w$ 
16:    end while;
17:    repl_or_reg( $q_0, w'$ );
18:  end function;
19: function repl_or_reg( $q, v$ );
20:   if  $v \neq \epsilon$  then
21:      $\delta(q, v_1) \leftarrow \text{repl\_or\_reg}(\delta(q, v_1), v_{2 \dots |v|})$ ;
22:   end if;
23:   if  $\exists_{r \in R} r \equiv q$  then
24:     delete  $q$ ; return  $r$ ;
25:   else
26:      $R \leftarrow R \cup \{q\}$ ; return  $q$ ;
27:   end if;
28: end function;
```



aimaient

# Incremental construction from sorted data – examples

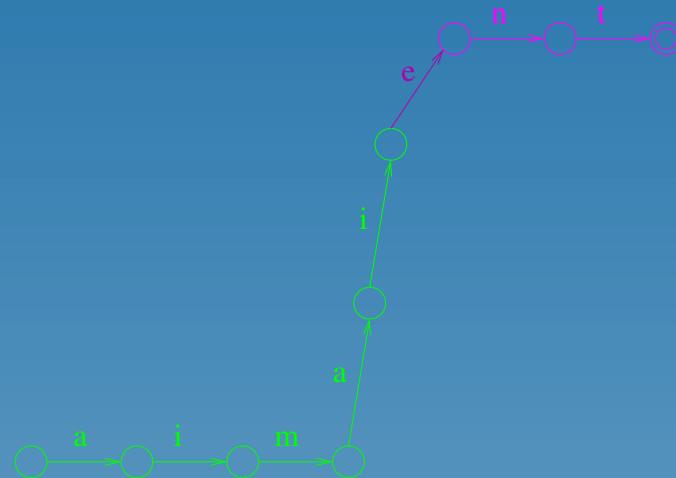
```
1: function sorted_construction;
2:    $w' \leftarrow \epsilon;$ 
3:   while input not empty do
4:      $s \leftarrow q_0; i \leftarrow 1; w \leftarrow \text{next word};$ 
5:     while  $i \leq |w|$  and  $\delta(s, w_i) = \perp$  do
6:        $s \leftarrow \delta(s, w_i); i \leftarrow i + 1;$ 
7:     end while;
8:     if  $i \leq |w'|$  then
9:       repl_or_reg( $\delta(s, w_i)$ ,  $w'_{i+1 \dots |w'|}$ );
10:      end if;
11:      while  $i \leq |w|$  do
12:         $\delta(s, w_i) \leftarrow \text{new state};$ 
13:         $s \leftarrow \delta(s, w_i); i \leftarrow i + 1;$ 
14:      end while;
15:       $F \leftarrow F \cup \{s\}; w' \leftarrow w$ 
16:    end while;
17:    repl_or_reg( $q_0, w'$ );
18:  end function;
19: function repl_or_reg( $q, v$ );
20:   if  $v \neq \epsilon$  then
21:      $\delta(q, v_1) \leftarrow \text{repl\_or\_reg}(\delta(q, v_1), v_{2 \dots |v|})$ ;
22:   end if;
23:   if  $\exists_{r \in R} r \equiv q$  then
24:     delete  $q$ ; return  $r$ ;
25:   else
26:      $R \leftarrow R \cup \{q\}$ ; return  $q$ ;
27:   end if;
28: end function;
```



aimais

# Incremental construction from sorted data – examples

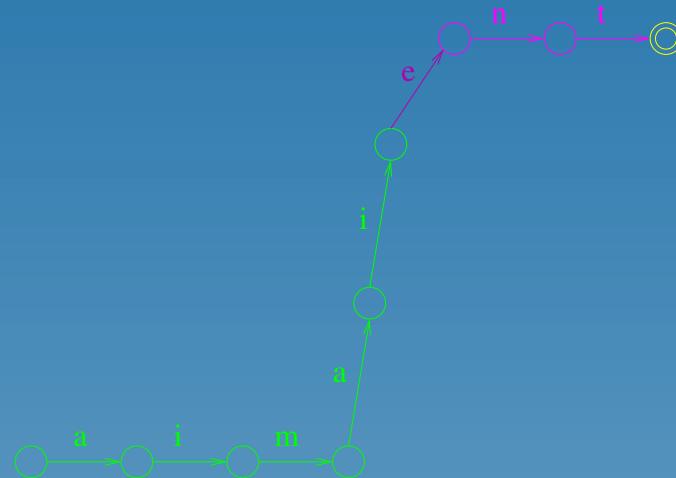
```
1: function sorted_construction;
2:    $w' \leftarrow \epsilon;$ 
3:   while input not empty do
4:      $s \leftarrow q_0; i \leftarrow 1; w \leftarrow \text{next word};$ 
5:     while  $i \leq |w|$  and  $\delta(s, w_i) = \perp$  do
6:        $s \leftarrow \delta(s, w_i); i \leftarrow i + 1;$ 
7:     end while;
8:     if  $i \leq |w'|$  then
9:       repl_or_reg( $\delta(s, w_i)$ ,  $w'_{i+1..|w'|}$ );
10:      end if;
11:      while  $i \leq |w|$  do
12:         $\delta(s, w_i) \leftarrow \text{new state};$ 
13:         $s \leftarrow \delta(s, w_i); i \leftarrow i + 1;$ 
14:      end while;
15:       $F \leftarrow F \cup \{s\}; w' \leftarrow w$ 
16:    end while;
17:    repl_or_reg( $q_0, w'$ );
18:  end function;
19: function repl_or_reg( $q, v$ );
20:   if  $v \neq \epsilon$  then
21:      $\delta(q, v_1) \leftarrow \text{repl\_or\_reg}(\delta(q, v_1), v_{2..|v|})$ ;
22:   end if;
23:   if  $\exists_{r \in R} r \equiv q$  then
24:     delete  $q$ ; return  $r$ ;
25:   else
26:      $R \leftarrow R \cup \{q\}$ ; return  $q$ ;
27:   end if;
28: end function;
```



aimais

# Incremental construction from sorted data – examples

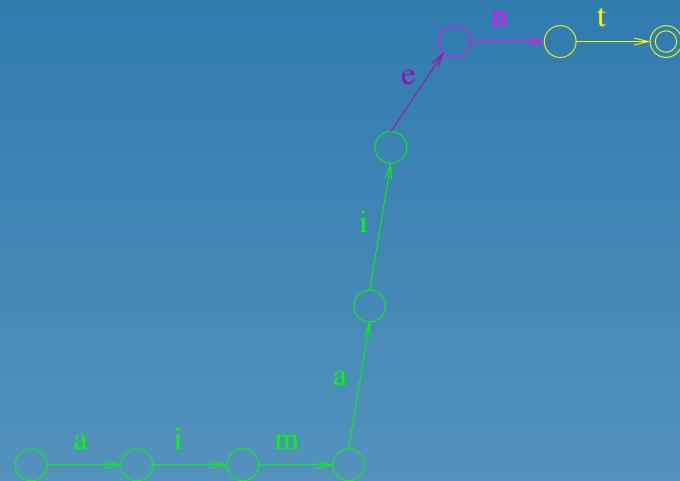
```
1: function sorted_construction;
2:    $w' \leftarrow \epsilon;$ 
3:   while input not empty do
4:      $s \leftarrow q_0; i \leftarrow 1; w \leftarrow \text{next word};$ 
5:     while  $i \leq |w|$  and  $\delta(s, w_i) = \perp$  do
6:        $s \leftarrow \delta(s, w_i); i \leftarrow i + 1;$ 
7:     end while;
8:     if  $i \leq |w'|$  then
9:       repl_or_reg( $\delta(s, w_i)$ ,  $w'_{i+1..|w'|}$ );
10:      end if;
11:      while  $i \leq |w|$  do
12:         $\delta(s, w_i) \leftarrow \text{new state};$ 
13:         $s \leftarrow \delta(s, w_i); i \leftarrow i + 1;$ 
14:      end while;
15:       $F \leftarrow F \cup \{s\}; w' \leftarrow w$ 
16:    end while;
17:    repl_or_reg( $q_0, w'$ );
18:  end function;
19: function repl_or_reg( $q, v$ );
20:   if  $v \neq \epsilon$  then
21:      $\delta(q, v_1) \leftarrow \text{repl\_or\_reg}(\delta(q, v_1), v_{2..|v|})$ ;
22:   end if;
23:   if  $\exists_{r \in R} r \equiv q$  then
24:     delete  $q$ ; return  $r$ ;
25:   else
26:      $R \leftarrow R \cup \{q\}$ ; return  $q$ ;
27:   end if;
28: end function;
```



aimais

# Incremental construction from sorted data – examples

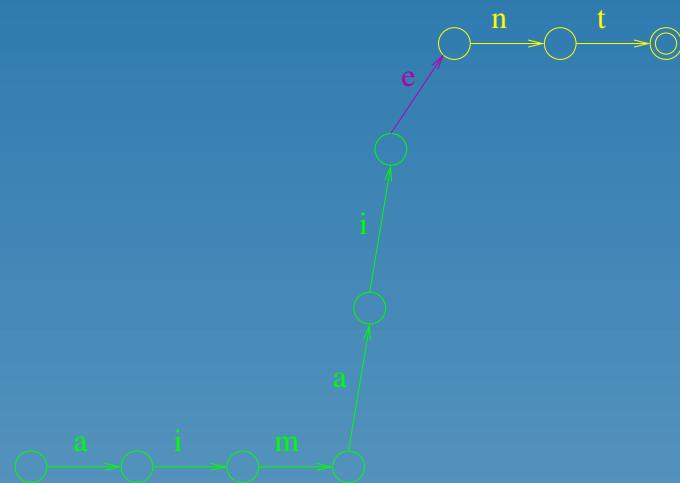
```
1: function sorted_construction;
2:    $w' \leftarrow \epsilon;$ 
3:   while input not empty do
4:      $s \leftarrow q_0; i \leftarrow 1; w \leftarrow \text{next word};$ 
5:     while  $i \leq |w|$  and  $\delta(s, w_i) = \perp$  do
6:        $s \leftarrow \delta(s, w_i); i \leftarrow i + 1;$ 
7:     end while;
8:     if  $i \leq |w'|$  then
9:       repl_or_reg( $\delta(s, w_i)$ ,  $w'_{i+1..|w'|}$ );
10:      end if;
11:      while  $i \leq |w|$  do
12:         $\delta(s, w_i) \leftarrow \text{new state};$ 
13:         $s \leftarrow \delta(s, w_i); i \leftarrow i + 1;$ 
14:      end while;
15:       $F \leftarrow F \cup \{s\}; w' \leftarrow w$ 
16:    end while;
17:    repl_or_reg( $q_0, w'$ );
18:  end function;
19: function repl_or_reg( $q, v$ );
20:   if  $v \neq \epsilon$  then
21:      $\delta(q, v_1) \leftarrow \text{repl\_or\_reg}(\delta(q, v_1), v_{2..|v|})$ ;
22:   end if;
23:   if  $\exists_{r \in R} r \equiv q$  then
24:     delete  $q$ ; return  $r$ ;
25:   else
26:      $R \leftarrow R \cup \{q\}$ ; return  $q$ ;
27:   end if;
28: end function;
```



aimais

# Incremental construction from sorted data – examples

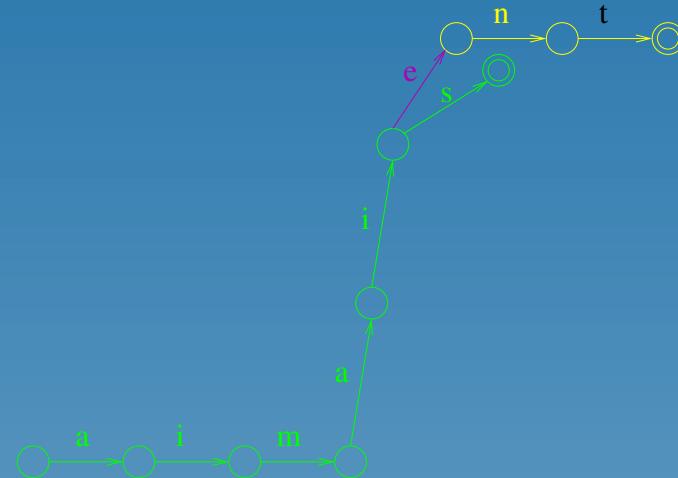
```
1: function sorted_construction;
2:    $w' \leftarrow \epsilon;$ 
3:   while input not empty do
4:      $s \leftarrow q_0; i \leftarrow 1; w \leftarrow \text{next word};$ 
5:     while  $i \leq |w|$  and  $\delta(s, w_i) = \perp$  do
6:        $s \leftarrow \delta(s, w_i); i \leftarrow i + 1;$ 
7:     end while;
8:     if  $i \leq |w'|$  then
9:       repl_or_reg( $\delta(s, w_i)$ ,  $w'_{i+1..|w'|}$ );
10:      end if;
11:      while  $i \leq |w|$  do
12:         $\delta(s, w_i) \leftarrow \text{new state};$ 
13:         $s \leftarrow \delta(s, w_i); i \leftarrow i + 1;$ 
14:      end while;
15:       $F \leftarrow F \cup \{s\}; w' \leftarrow w$ 
16:    end while;
17:    repl_or_reg( $q_0, w'$ );
18:  end function;
19: function repl_or_reg( $q, v$ );
20:   if  $v \neq \epsilon$  then
21:      $\delta(q, v_1) \leftarrow \text{repl\_or\_reg}(\delta(q, v_1), v_{2..|v|})$ ;
22:   end if;
23:   if  $\exists_{r \in R} r \equiv q$  then
24:     delete  $q$ ; return  $r$ ;
25:   else
26:      $R \leftarrow R \cup \{q\}$ ; return  $q$ ;
27:   end if;
28: end function;
```



aimais

# Incremental construction from sorted data – examples

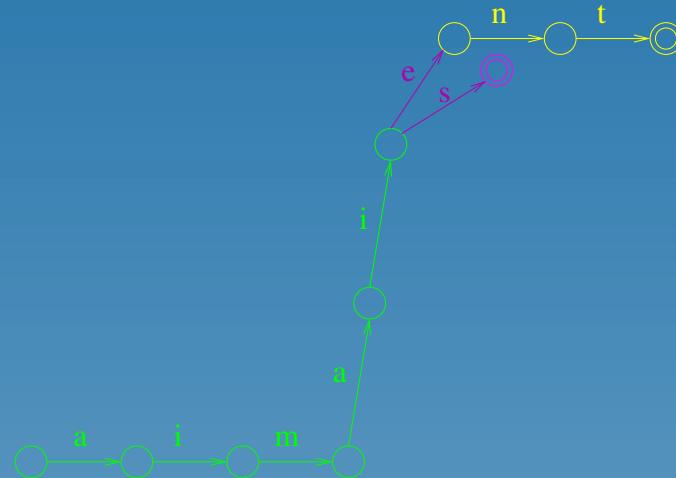
```
1: function sorted_construction;
2:    $w' \leftarrow \epsilon;$ 
3:   while input not empty do
4:      $s \leftarrow q_0; i \leftarrow 1; w \leftarrow \text{next word};$ 
5:     while  $i \leq |w|$  and  $\delta(s, w_i) = \perp$  do
6:        $s \leftarrow \delta(s, w_i); i \leftarrow i + 1;$ 
7:     end while;
8:     if  $i \leq |w'|$  then
9:       repl_or_reg( $\delta(s, w_i)$ ,  $w'_{i+1 \dots |w'|}$ );
10:      end if;
11:      while  $i \leq |w|$  do
12:         $\delta(s, w_i) \leftarrow \text{new state};$ 
13:         $s \leftarrow \delta(s, w_i); i \leftarrow i + 1;$ 
14:      end while;
15:       $F \leftarrow F \cup \{s\}; w' \leftarrow w$ 
16:    end while;
17:    repl_or_reg( $q_0, w'$ );
18:  end function;
19: function repl_or_reg( $q, v$ );
20:   if  $v \neq \epsilon$  then
21:      $\delta(q, v_1) \leftarrow \text{repl\_or\_reg}(\delta(q, v_1), v_{2 \dots |v|})$ ;
22:   end if;
23:   if  $\exists_{r \in R} r \equiv q$  then
24:     delete  $q$ ; return  $r$ ;
25:   else
26:      $R \leftarrow R \cup \{q\}$ ; return  $q$ ;
27:   end if;
28: end function;
```



aimais

# Incremental construction from sorted data – examples

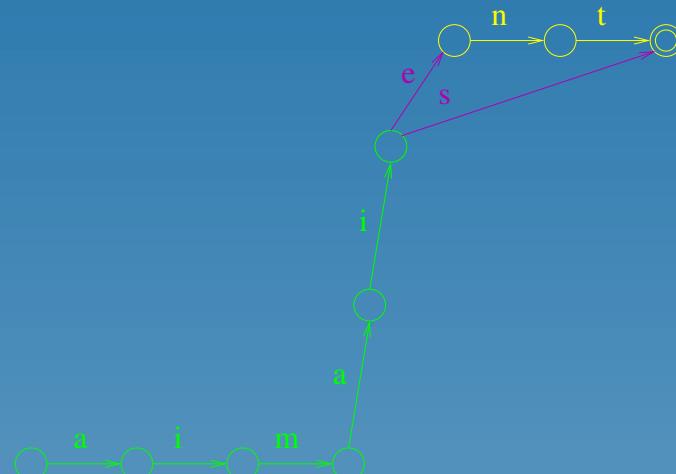
```
1: function sorted_construction;
2:    $w' \leftarrow \epsilon;$ 
3:   while input not empty do
4:      $s \leftarrow q_0; i \leftarrow 1; w \leftarrow \text{next word};$ 
5:     while  $i \leq |w|$  and  $\delta(s, w_i) = \perp$  do
6:        $s \leftarrow \delta(s, w_i); i \leftarrow i + 1;$ 
7:     end while;
8:     if  $i \leq |w'|$  then
9:       repl_or_reg( $\delta(s, w_i)$ ,  $w'_{i+1 \dots |w'|}$ );
10:      end if;
11:      while  $i \leq |w|$  do
12:         $\delta(s, w_i) \leftarrow \text{new state};$ 
13:         $s \leftarrow \delta(s, w_i); i \leftarrow i + 1;$ 
14:      end while;
15:       $F \leftarrow F \cup \{s\}; w' \leftarrow w$ 
16:    end while;
17:    repl_or_reg( $q_0, w'$ );
18:  end function;
19: function repl_or_reg( $q, v$ );
20:   if  $v \neq \epsilon$  then
21:      $\delta(q, v_1) \leftarrow \text{repl\_or\_reg}(\delta(q, v_1), v_{2 \dots |v|})$ ;
22:   end if;
23:   if  $\exists_{r \in R} r \equiv q$  then
24:     delete  $q$ ; return  $r$ ;
25:   else
26:      $R \leftarrow R \cup \{q\}$ ; return  $q$ ;
27:   end if;
28: end function;
```



aimait

# Incremental construction from sorted data – examples

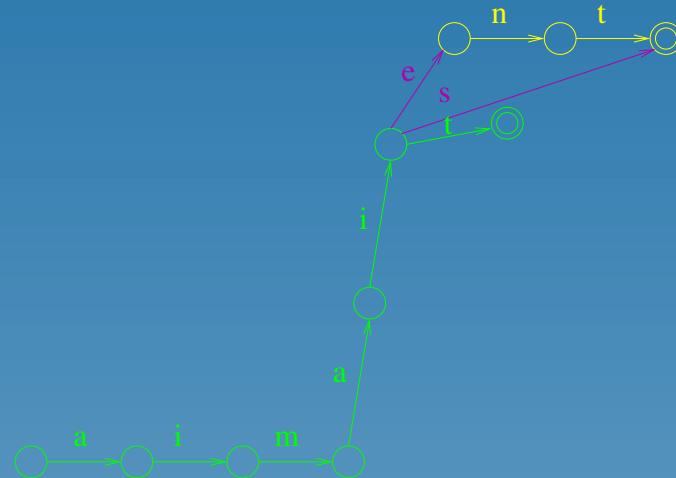
```
1: function sorted_construction;
2:    $w' \leftarrow \epsilon;$ 
3:   while input not empty do
4:      $s \leftarrow q_0; i \leftarrow 1; w \leftarrow \text{next word};$ 
5:     while  $i \leq |w|$  and  $\delta(s, w_i) = \perp$  do
6:        $s \leftarrow \delta(s, w_i); i \leftarrow i + 1;$ 
7:     end while;
8:     if  $i \leq |w'|$  then
9:       repl_or_reg( $\delta(s, w_i)$ ,  $w'_{i+1..|w'|}$ );
10:      end if;
11:      while  $i \leq |w|$  do
12:         $\delta(s, w_i) \leftarrow \text{new state};$ 
13:         $s \leftarrow \delta(s, w_i); i \leftarrow i + 1;$ 
14:      end while;
15:       $F \leftarrow F \cup \{s\}; w' \leftarrow w$ 
16:    end while;
17:    repl_or_reg( $q_0, w'$ );
18:  end function;
19: function repl_or_reg( $q, v$ );
20:   if  $v \neq \epsilon$  then
21:      $\delta(q, v_1) \leftarrow \text{repl\_or\_reg}(\delta(q, v_1), v_{2..|v|})$ ;
22:   end if;
23:   if  $\exists_{r \in R} r \equiv q$  then
24:     delete  $q$ ; return  $r$ ;
25:   else
26:      $R \leftarrow R \cup \{q\}$ ; return  $q$ ;
27:   end if;
28: end function;
```



aimait

# Incremental construction from sorted data – examples

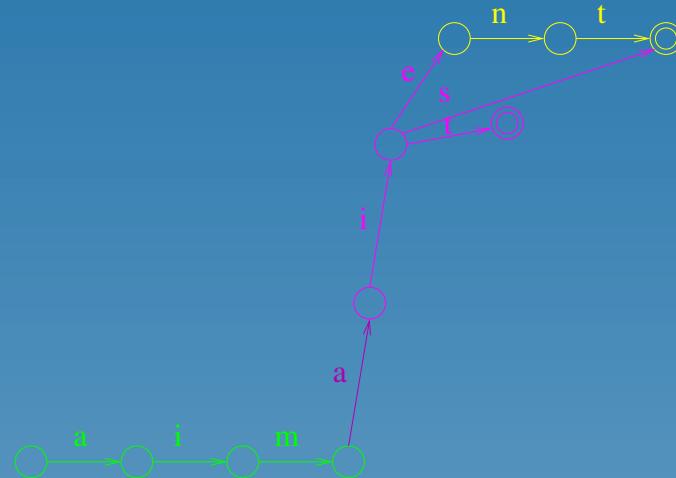
```
1: function sorted_construction;
2:    $w' \leftarrow \epsilon;$ 
3:   while input not empty do
4:      $s \leftarrow q_0; i \leftarrow 1; w \leftarrow \text{next word};$ 
5:     while  $i \leq |w|$  and  $\delta(s, w_i) = \perp$  do
6:        $s \leftarrow \delta(s, w_i); i \leftarrow i + 1;$ 
7:     end while;
8:     if  $i \leq |w'|$  then
9:       repl_or_reg( $\delta(s, w_i)$ ,  $w'_{i+1 \dots |w'|}$ );
10:      end if;
11:      while  $i \leq |w|$  do
12:         $\delta(s, w_i) \leftarrow \text{new state};$ 
13:         $s \leftarrow \delta(s, w_i); i \leftarrow i + 1;$ 
14:      end while;
15:       $F \leftarrow F \cup \{s\}; w' \leftarrow w$ 
16:    end while;
17:    repl_or_reg( $q_0, w'$ );
18:  end function;
19: function repl_or_reg( $q, v$ );
20:   if  $v \neq \epsilon$  then
21:      $\delta(q, v_1) \leftarrow \text{repl\_or\_reg}(\delta(q, v_1), v_{2 \dots |v|})$ ;
22:   end if;
23:   if  $\exists_{r \in R} r \equiv q$  then
24:     delete  $q$ ; return  $r$ ;
25:   else
26:      $R \leftarrow R \cup \{q\}$ ; return  $q$ ;
27:   end if;
28: end function;
```



aimait

# Incremental construction from sorted data – examples

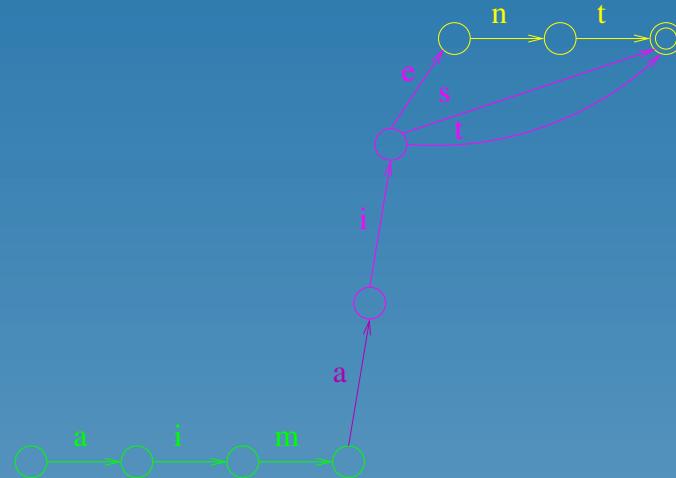
```
1: function sorted_construction;
2:    $w' \leftarrow \epsilon;$ 
3:   while input not empty do
4:      $s \leftarrow q_0; i \leftarrow 1; w \leftarrow \text{next word};$ 
5:     while  $i \leq |w|$  and  $\delta(s, w_i) = \perp$  do
6:        $s \leftarrow \delta(s, w_i); i \leftarrow i + 1;$ 
7:     end while;
8:     if  $i \leq |w'|$  then
9:       repl_or_reg( $\delta(s, w_i)$ ,  $w'_{i+1 \dots |w'|}$ );
10:      end if;
11:      while  $i \leq |w|$  do
12:         $\delta(s, w_i) \leftarrow \text{new state};$ 
13:         $s \leftarrow \delta(s, w_i); i \leftarrow i + 1;$ 
14:      end while;
15:       $F \leftarrow F \cup \{s\}; w' \leftarrow w$ 
16:    end while;
17:    repl_or_reg( $q_0, w'$ );
18:  end function;
19: function repl_or_reg( $q, v$ );
20:   if  $v \neq \epsilon$  then
21:      $\delta(q, v_1) \leftarrow \text{repl\_or\_reg}(\delta(q, v_1), v_{2 \dots |v|})$ ;
22:   end if;
23:   if  $\exists_{r \in R} r \equiv q$  then
24:     delete  $q$ ; return  $r$ ;
25:   else
26:      $R \leftarrow R \cup \{q\}$ ; return  $q$ ;
27:   end if;
28: end function;
```



aime

# Incremental construction from sorted data – examples

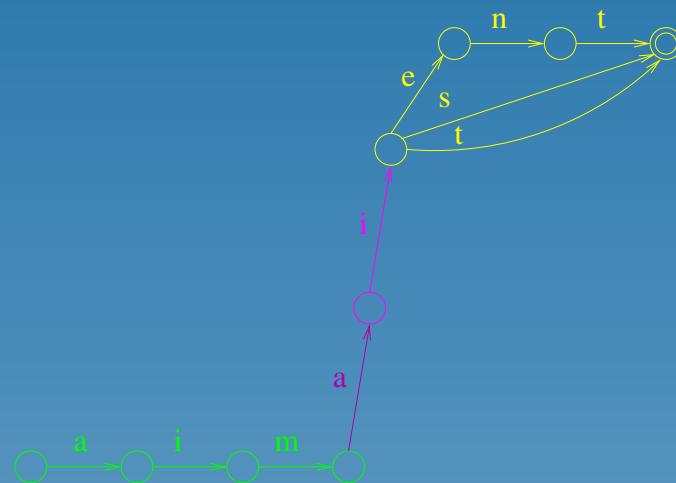
```
1: function sorted_construction;
2:    $w' \leftarrow \epsilon;$ 
3:   while input not empty do
4:      $s \leftarrow q_0; i \leftarrow 1; w \leftarrow \text{next word};$ 
5:     while  $i \leq |w|$  and  $\delta(s, w_i) = \perp$  do
6:        $s \leftarrow \delta(s, w_i); i \leftarrow i + 1;$ 
7:     end while;
8:     if  $i \leq |w'|$  then
9:       repl_or_reg( $\delta(s, w_i)$ ,  $w'_{i+1..|w'|}$ );
10:      end if;
11:      while  $i \leq |w|$  do
12:         $\delta(s, w_i) \leftarrow \text{new state};$ 
13:         $s \leftarrow \delta(s, w_i); i \leftarrow i + 1;$ 
14:      end while;
15:       $F \leftarrow F \cup \{s\}; w' \leftarrow w$ 
16:    end while;
17:    repl_or_reg( $q_0, w'$ );
18:  end function;
19: function repl_or_reg( $q, v$ );
20:   if  $v \neq \epsilon$  then
21:      $\delta(q, v_1) \leftarrow \text{repl\_or\_reg}(\delta(q, v_1), v_{2..|v|})$ ;
22:   end if;
23:   if  $\exists_{r \in R} r \equiv q$  then
24:     delete  $q$ ; return  $r$ ;
25:   else
26:      $R \leftarrow R \cup \{q\}$ ; return  $q$ ;
27:   end if;
28: end function;
```



aime

# Incremental construction from sorted data – examples

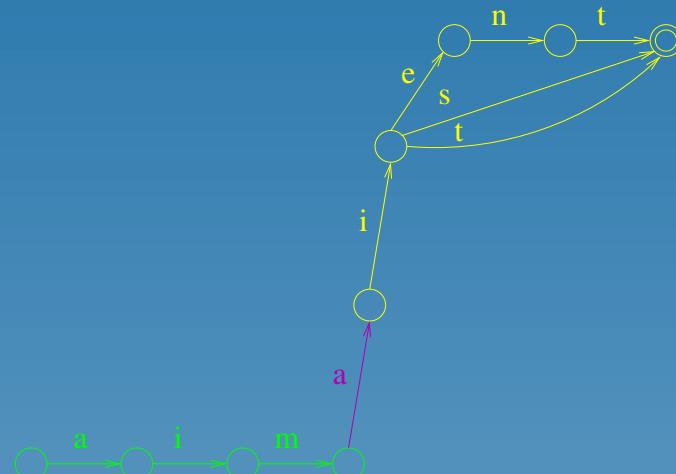
```
1: function sorted_construction;
2:    $w' \leftarrow \epsilon;$ 
3:   while input not empty do
4:      $s \leftarrow q_0; i \leftarrow 1; w \leftarrow \text{next word};$ 
5:     while  $i \leq |w|$  and  $\delta(s, w_i) = \perp$  do
6:        $s \leftarrow \delta(s, w_i); i \leftarrow i + 1;$ 
7:     end while;
8:     if  $i \leq |w'|$  then
9:       repl_or_reg( $\delta(s, w_i)$ ,  $w'_{i+1..|w'|}$ );
10:      end if;
11:      while  $i \leq |w|$  do
12:         $\delta(s, w_i) \leftarrow \text{new state};$ 
13:         $s \leftarrow \delta(s, w_i); i \leftarrow i + 1;$ 
14:      end while;
15:       $F \leftarrow F \cup \{s\}; w' \leftarrow w$ 
16:    end while;
17:    repl_or_reg( $q_0, w'$ );
18:  end function;
19: function repl_or_reg( $q, v$ );
20:   if  $v \neq \epsilon$  then
21:      $\delta(q, v_1) \leftarrow \text{repl\_or\_reg}(\delta(q, v_1), v_{2..|v|})$ ;
22:   end if;
23:   if  $\exists_{r \in R} r \equiv q$  then
24:     delete  $q$ ; return  $r$ ;
25:   else
26:      $R \leftarrow R \cup \{q\}$ ; return  $q$ ;
27:   end if;
28: end function;
```



aime

# Incremental construction from sorted data – examples

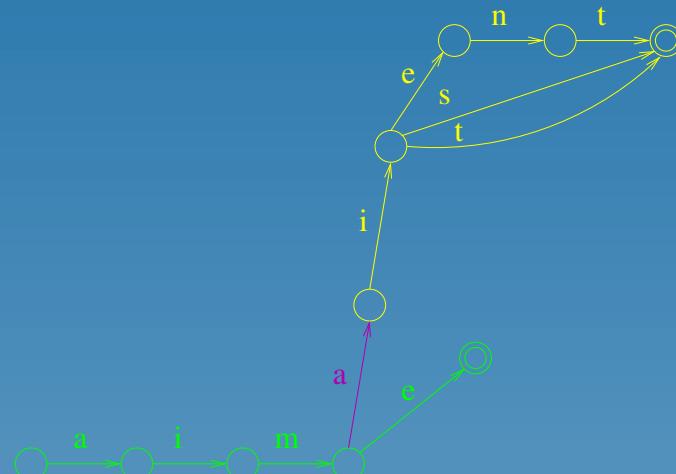
```
1: function sorted_construction;
2:    $w' \leftarrow \epsilon;$ 
3:   while input not empty do
4:      $s \leftarrow q_0; i \leftarrow 1; w \leftarrow \text{next word};$ 
5:     while  $i \leq |w|$  and  $\delta(s, w_i) = \perp$  do
6:        $s \leftarrow \delta(s, w_i); i \leftarrow i + 1;$ 
7:     end while;
8:     if  $i \leq |w'|$  then
9:       repl_or_reg( $\delta(s, w_i)$ ,  $w'_{i+1..|w'|}$ );
10:    end if;
11:    while  $i \leq |w|$  do
12:       $\delta(s, w_i) \leftarrow \text{new state};$ 
13:       $s \leftarrow \delta(s, w_i); i \leftarrow i + 1;$ 
14:    end while;
15:     $F \leftarrow F \cup \{s\}; w' \leftarrow w$ 
16:  end while;
17:  repl_or_reg( $q_0, w'$ );
18: end function;
19: function repl_or_reg( $q, v$ );
20:   if  $v \neq \epsilon$  then
21:      $\delta(q, v_1) \leftarrow \text{repl\_or\_reg}(\delta(q, v_1), v_{2..|v|})$ ;
22:   end if;
23:   if  $\exists_{r \in R} r \equiv q$  then
24:     delete  $q$ ; return  $r$ ;
25:   else
26:      $R \leftarrow R \cup \{q\}$ ; return  $q$ ;
27:   end if;
28: end function;
```



aime

# Incremental construction from sorted data – examples

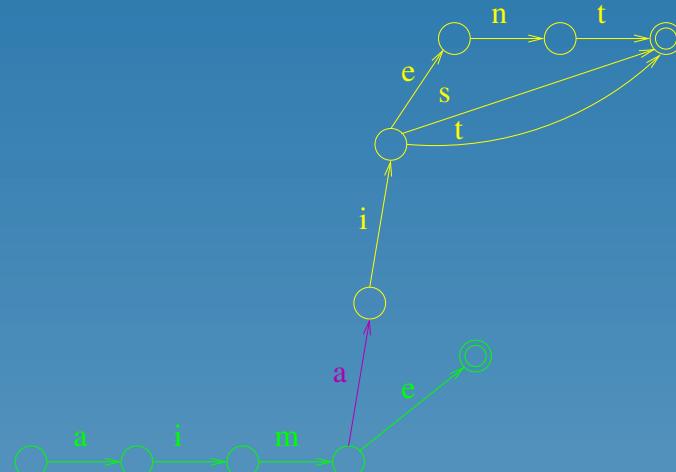
```
1: function sorted_construction;
2:    $w' \leftarrow \epsilon;$ 
3:   while input not empty do
4:      $s \leftarrow q_0; i \leftarrow 1; w \leftarrow \text{next word};$ 
5:     while  $i \leq |w|$  and  $\delta(s, w_i) = \perp$  do
6:        $s \leftarrow \delta(s, w_i); i \leftarrow i + 1;$ 
7:     end while;
8:     if  $i \leq |w'|$  then
9:       repl_or_reg( $\delta(s, w_i)$ ,  $w'_{i+1 \dots |w'|}$ );
10:      end if;
11:      while  $i \leq |w|$  do
12:         $\delta(s, w_i) \leftarrow \text{new state};$ 
13:         $s \leftarrow \delta(s, w_i); i \leftarrow i + 1;$ 
14:      end while;
15:       $F \leftarrow F \cup \{s\}; w' \leftarrow w$ 
16:    end while;
17:    repl_or_reg( $q_0, w'$ );
18:  end function;
19: function repl_or_reg( $q, v$ );
20:   if  $v \neq \epsilon$  then
21:      $\delta(q, v_1) \leftarrow \text{repl\_or\_reg}(\delta(q, v_1), v_{2 \dots |v|})$ ;
22:   end if;
23:   if  $\exists_{r \in R} r \equiv q$  then
24:     delete  $q$ ; return  $r$ ;
25:   else
26:      $R \leftarrow R \cup \{q\}$ ; return  $q$ ;
27:   end if;
28: end function;
```



aime

# Incremental construction from sorted data – examples

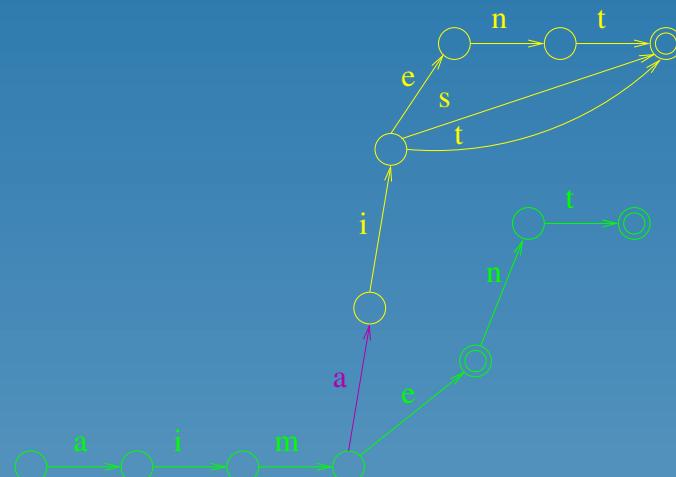
```
1: function sorted_construction;
2:    $w' \leftarrow \epsilon;$ 
3:   while input not empty do
4:      $s \leftarrow q_0; i \leftarrow 1; w \leftarrow \text{next word};$ 
5:     while  $i \leq |w|$  and  $\delta(s, w_i) = \perp$  do
6:        $s \leftarrow \delta(s, w_i); i \leftarrow i + 1;$ 
7:     end while;
8:     if  $i \leq |w'|$  then
9:       repl_or_reg( $\delta(s, w_i)$ ,  $w'_{i+1 \dots |w'|}$ );
10:      end if;
11:      while  $i \leq |w|$  do
12:         $\delta(s, w_i) \leftarrow \text{new state};$ 
13:         $s \leftarrow \delta(s, w_i); i \leftarrow i + 1;$ 
14:      end while;
15:       $F \leftarrow F \cup \{s\}; w' \leftarrow w$ 
16:    end while;
17:    repl_or_reg( $q_0, w'$ );
18:  end function;
19: function repl_or_reg( $q, v$ );
20:   if  $v \neq \epsilon$  then
21:      $\delta(q, v_1) \leftarrow \text{repl\_or\_reg}(\delta(q, v_1), v_{2 \dots |v|})$ ;
22:   end if;
23:   if  $\exists_{r \in R} r \equiv q$  then
24:     delete  $q$ ; return  $r$ ;
25:   else
26:      $R \leftarrow R \cup \{q\}$ ; return  $q$ ;
27:   end if;
28: end function;
```



aiment

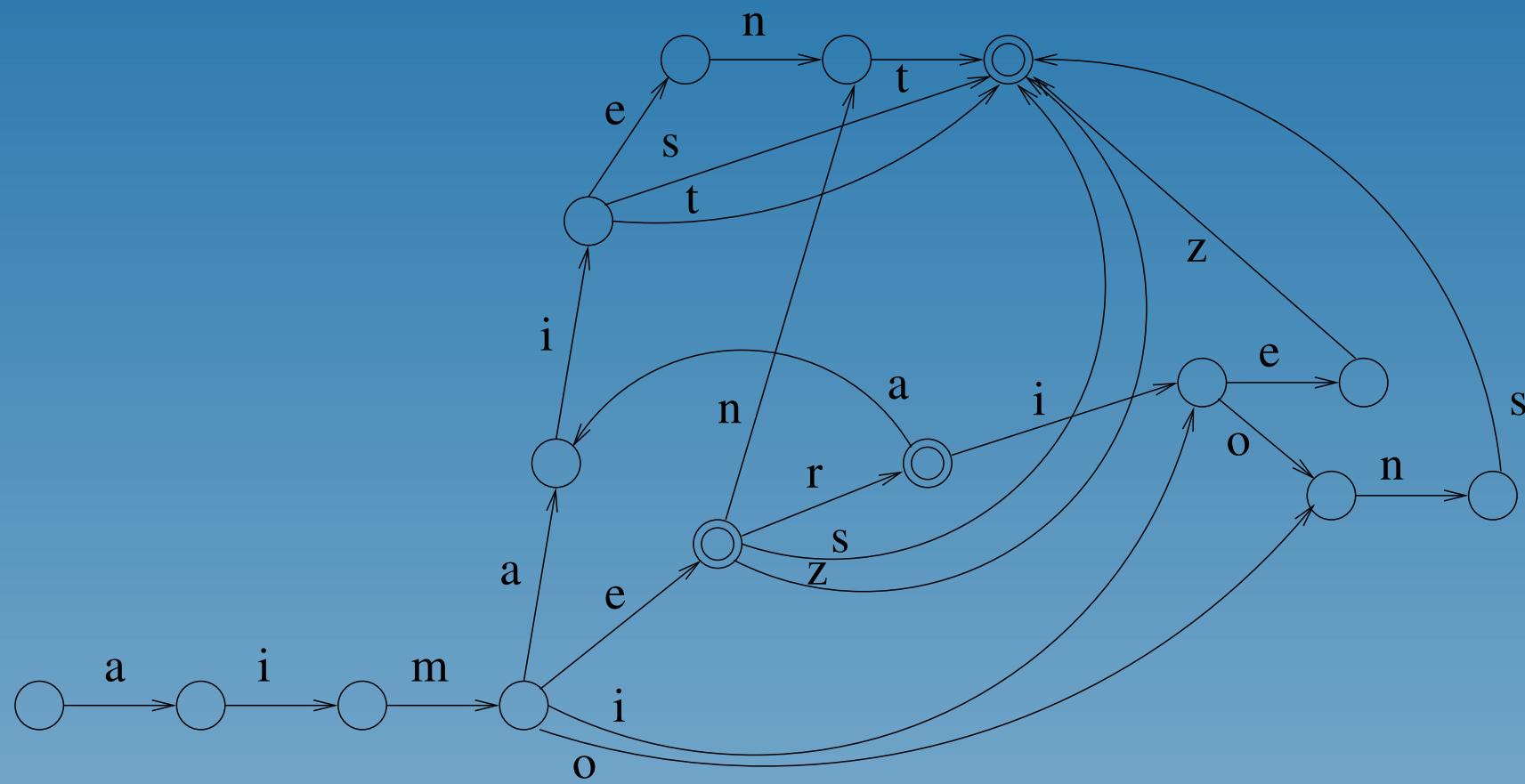
# Incremental construction from sorted data – examples

```
1: function sorted_construction;
2:    $w' \leftarrow \epsilon;$ 
3:   while input not empty do
4:      $s \leftarrow q_0; i \leftarrow 1; w \leftarrow \text{next word};$ 
5:     while  $i \leq |w|$  and  $\delta(s, w_i) = \perp$  do
6:        $s \leftarrow \delta(s, w_i); i \leftarrow i + 1;$ 
7:     end while;
8:     if  $i \leq |w'|$  then
9:       repl_or_reg( $\delta(s, w_i)$ ,  $w'_{i+1 \dots |w'|}$ );
10:      end if;
11:      while  $i \leq |w|$  do
12:         $\delta(s, w_i) \leftarrow \text{new state};$ 
13:         $s \leftarrow \delta(s, w_i); i \leftarrow i + 1;$ 
14:      end while;
15:       $F \leftarrow F \cup \{s\}; w' \leftarrow w$ 
16:    end while;
17:    repl_or_reg( $q_0, w'$ );
18:  end function;
19: function repl_or_reg( $q, v$ );
20:   if  $v \neq \epsilon$  then
21:      $\delta(q, v_1) \leftarrow \text{repl\_or\_reg}(\delta(q, v_1), v_{2 \dots |v|})$ ;
22:   end if;
23:   if  $\exists_{r \in R} r \equiv q$  then
24:     delete  $q$ ; return  $r$ ;
25:   else
26:      $R \leftarrow R \cup \{q\}$ ; return  $q$ ;
27:   end if;
28: end function;
```

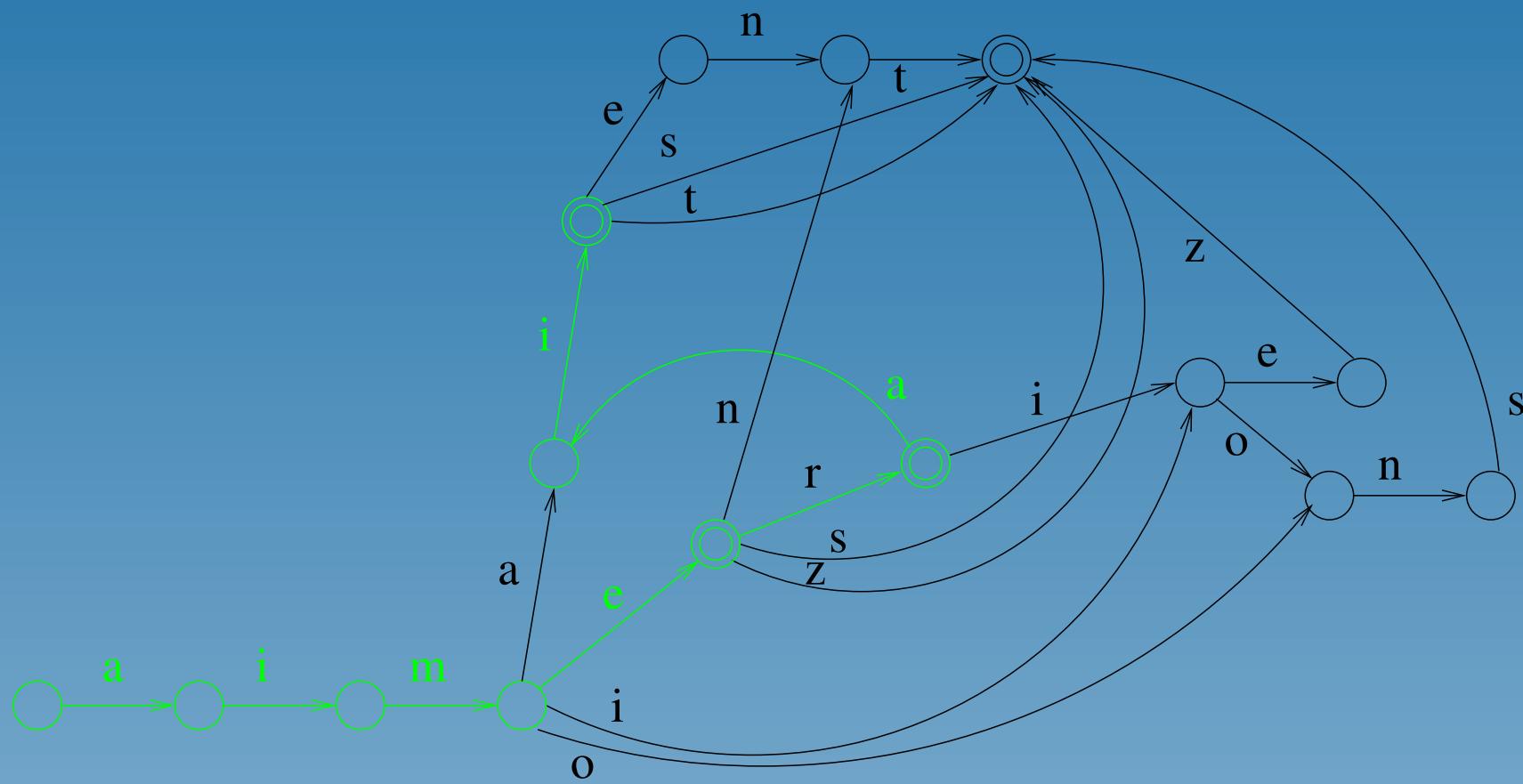


aiment

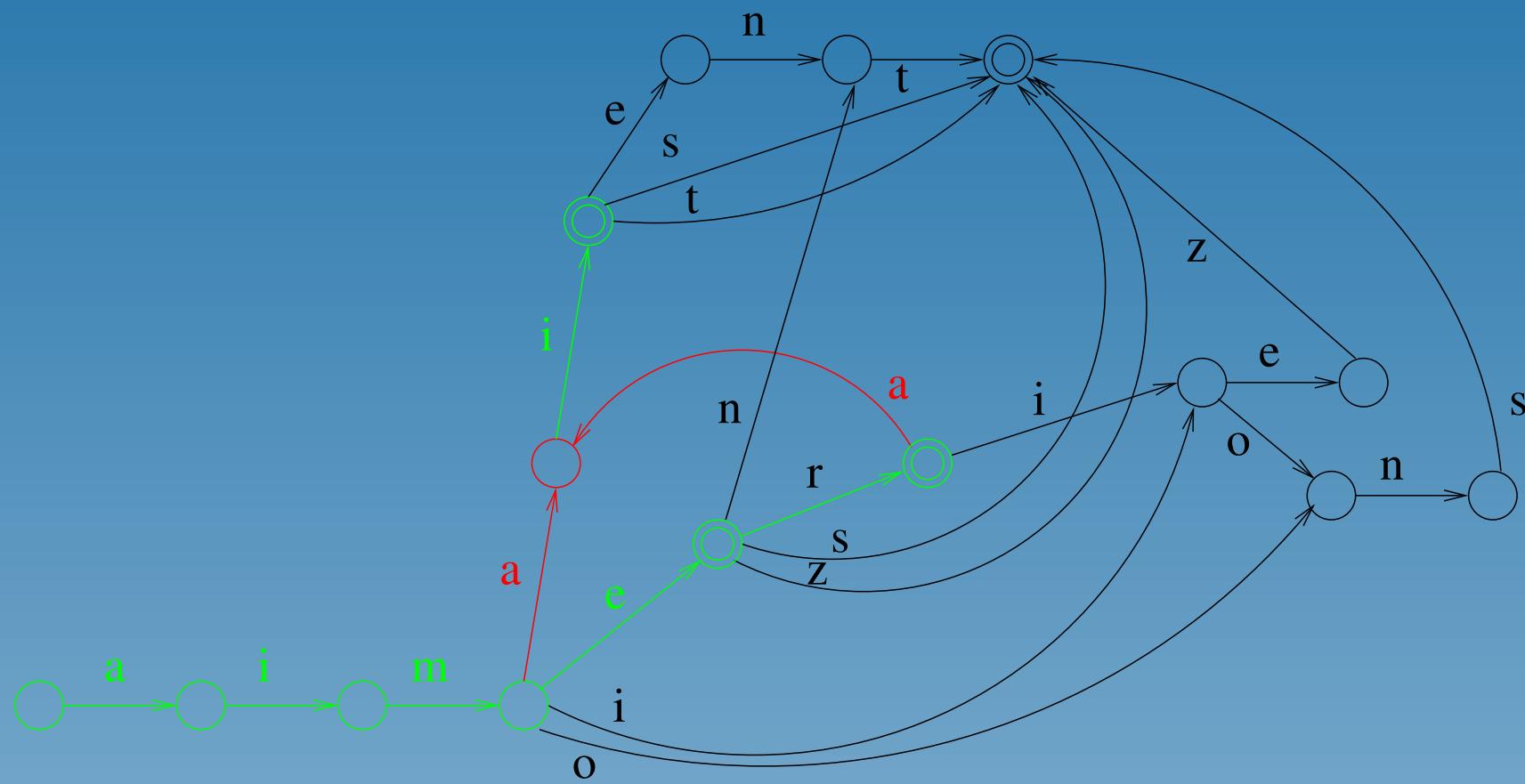
# Unsorted data – hidden dangers



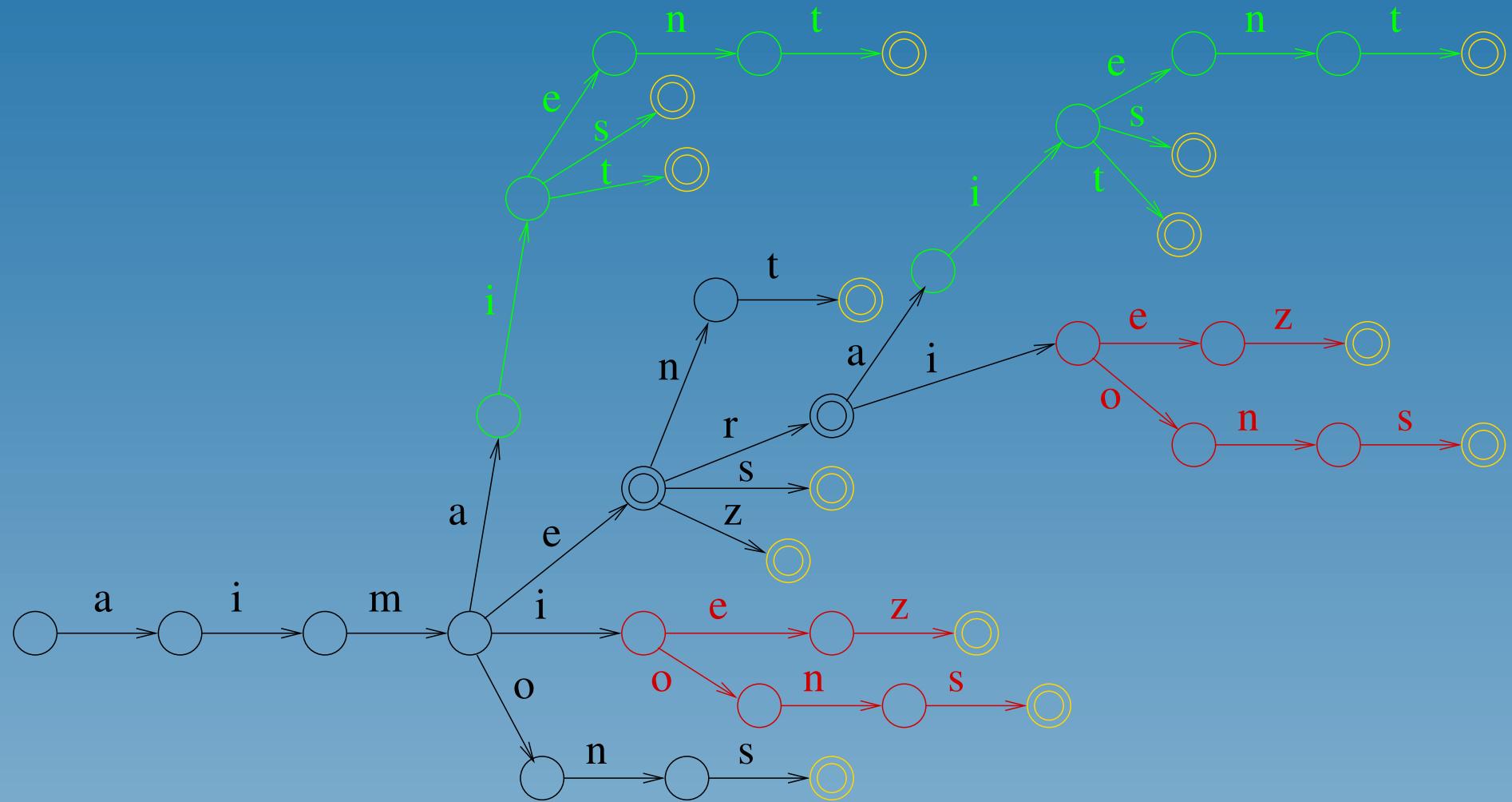
# Unsorted data – hidden dangers



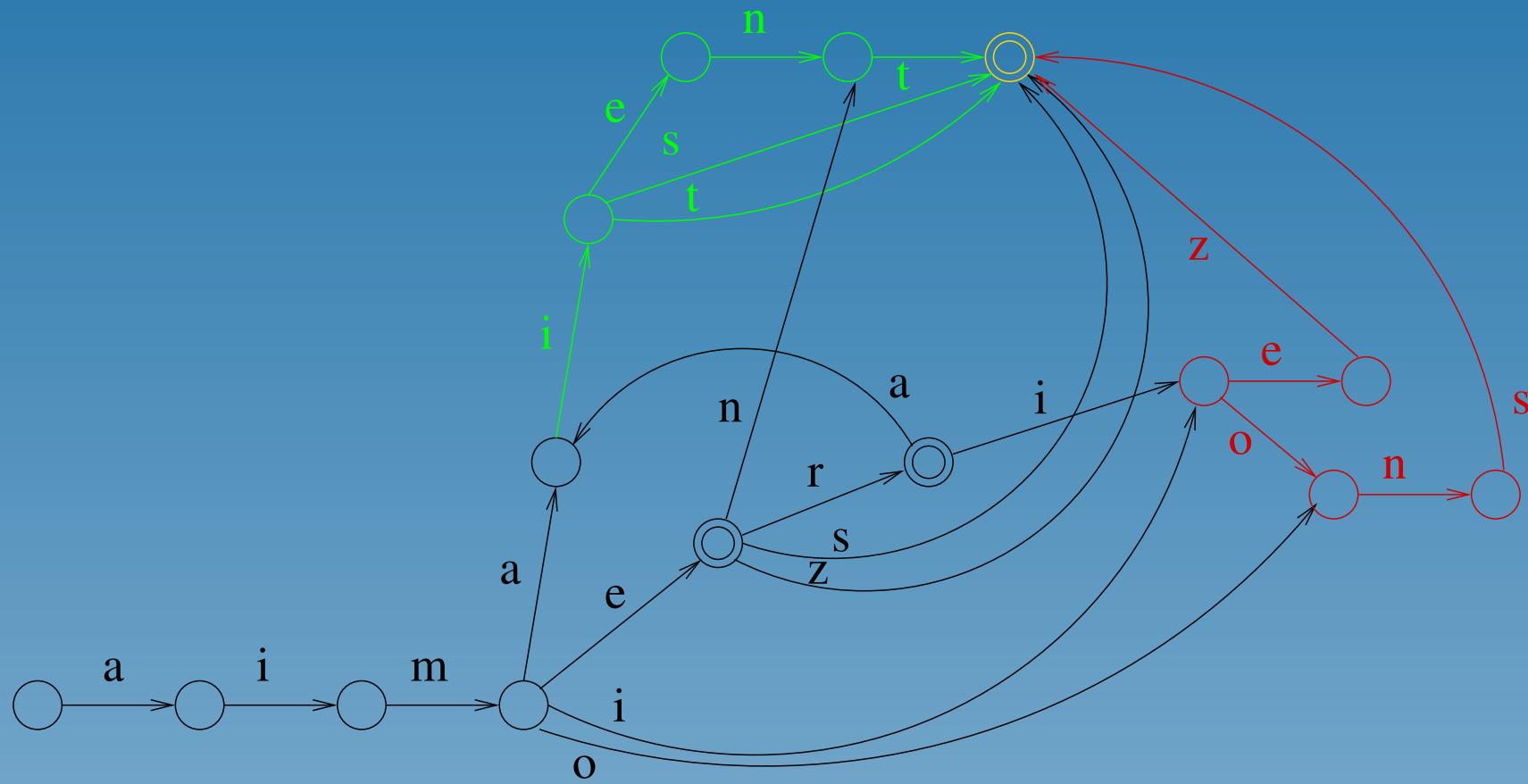
# Unsorted data – hidden dangers



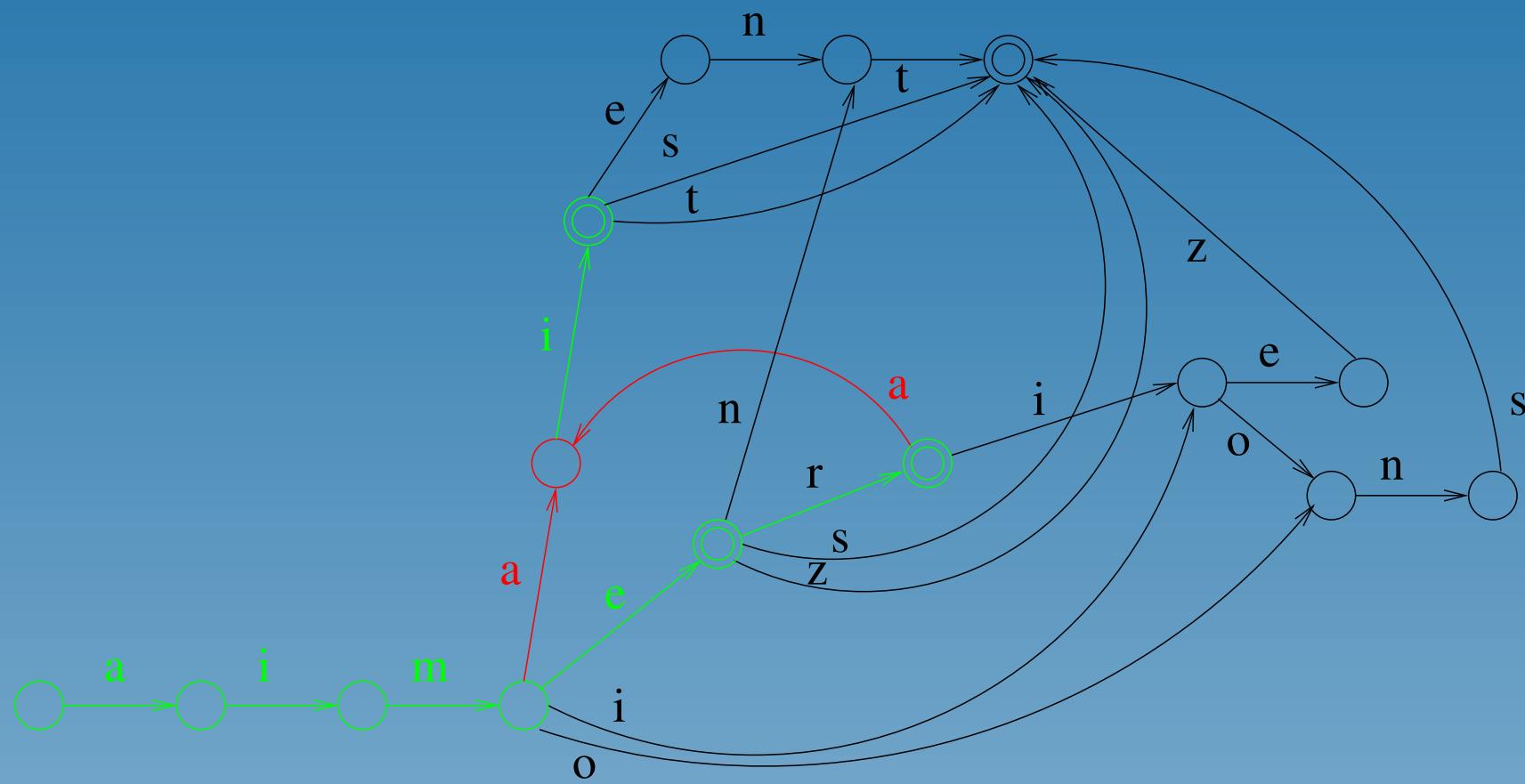
# Unsorted data – hidden dangers



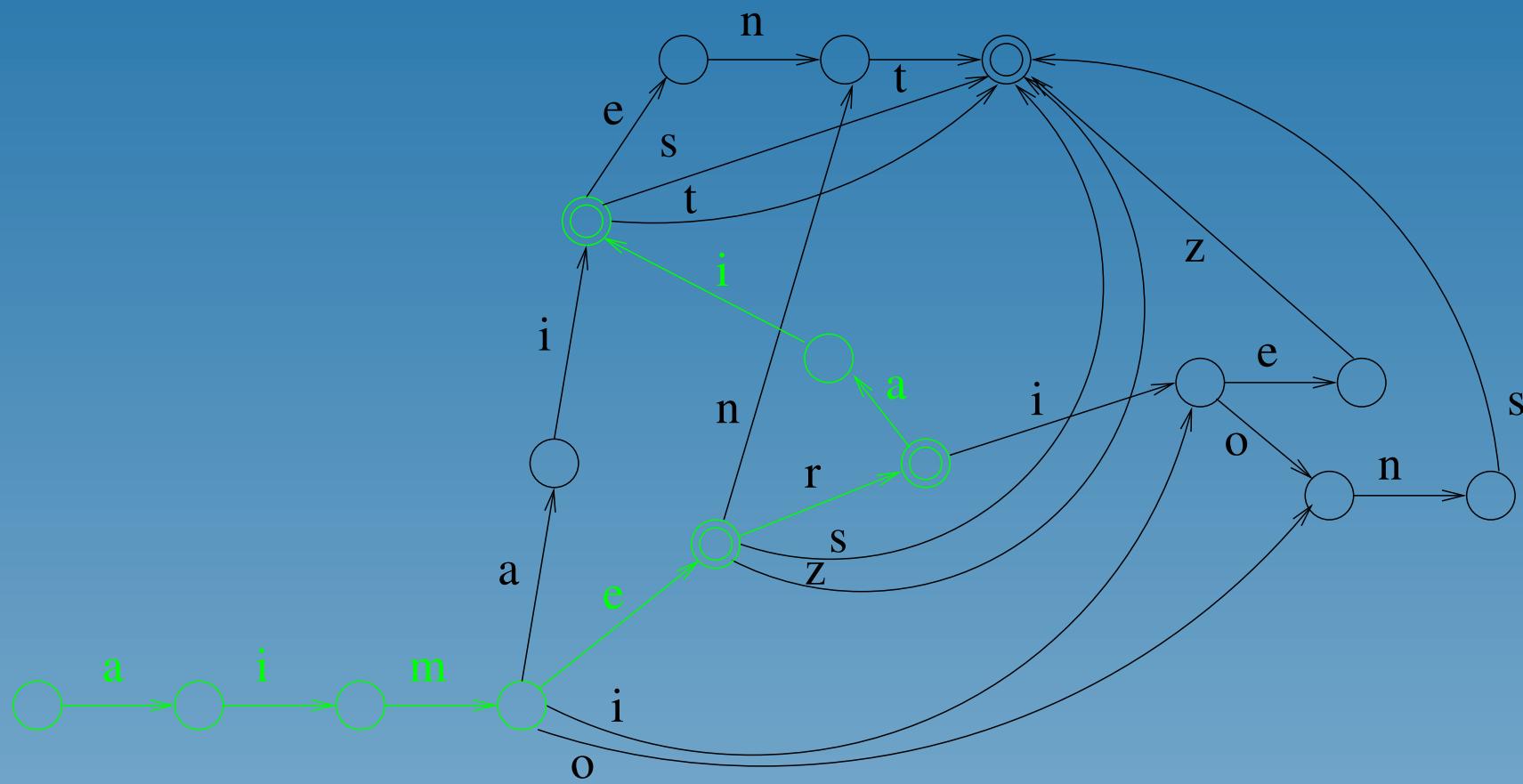
# Unsorted data – hidden dangers



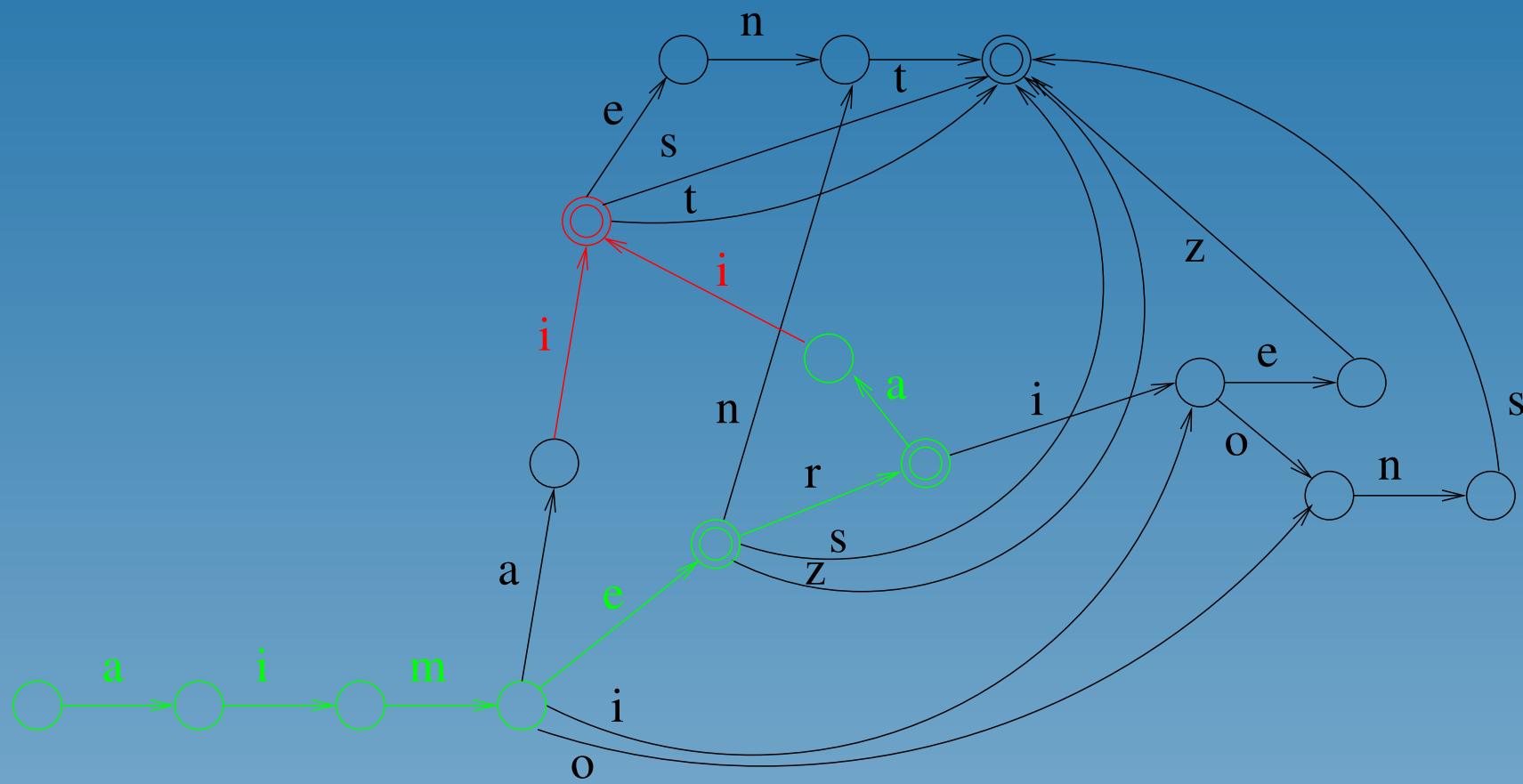
# Unsorted data – hidden dangers



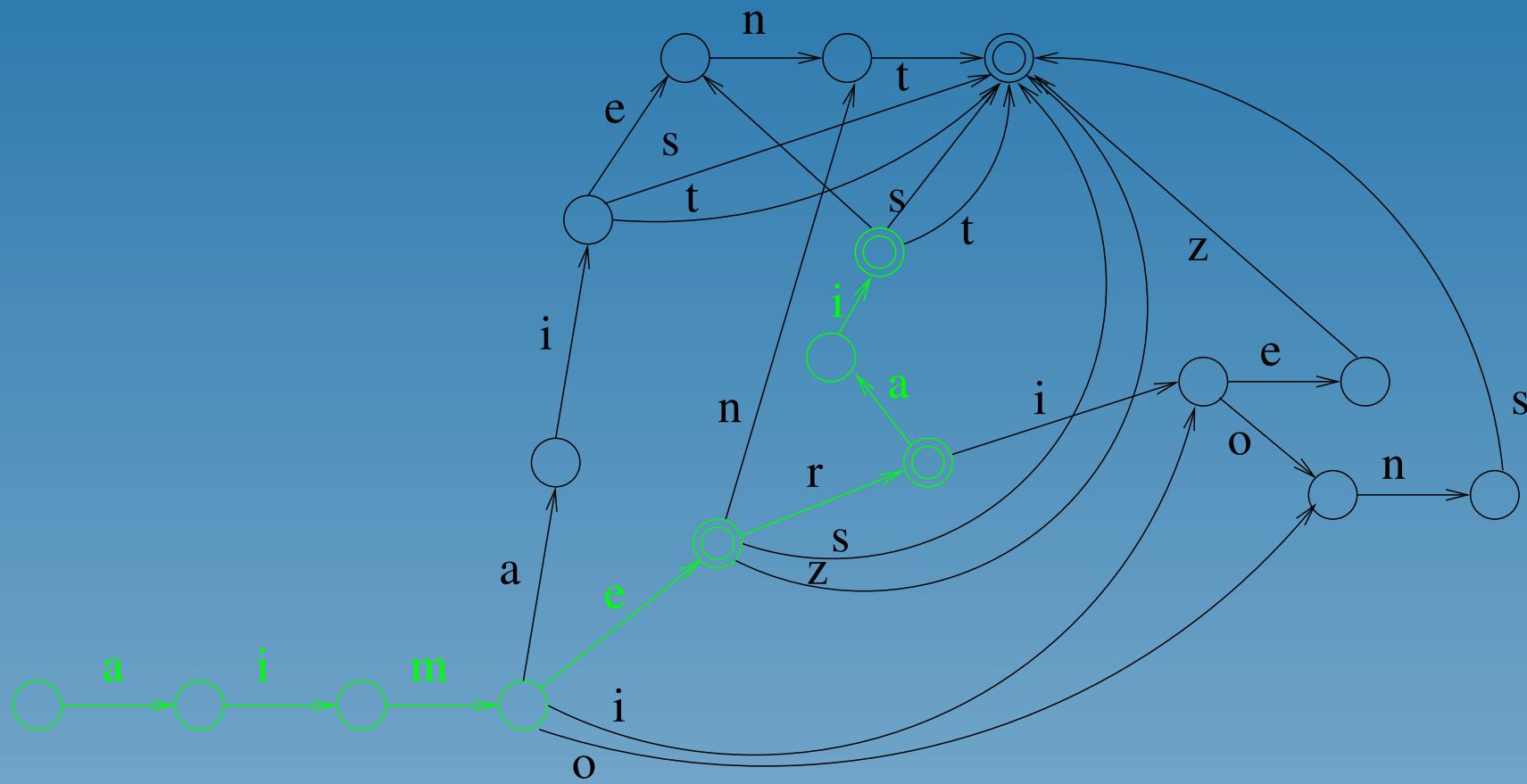
# Unsorted data – hidden dangers



# Unsorted data – hidden dangers



# Unsorted data – hidden dangers



# Incremental construction from unsorted data

```
1: function unsorted_construction;
2:   while input not empty do
3:      $s \leftarrow q_0; i \leftarrow 0; w \leftarrow \text{next word}; \text{push}(s, P);$ 
4:     while  $i \leq |w|$  and  $\delta(s, w_i) \neq \perp$  and  $\text{fanin}(\delta(s, w_i)) \leq 1$  do
5:        $s \leftarrow \delta(s, w_i); \text{push}(s, P); i \leftarrow i + 1;$ 
6:     end while;
7:      $R \leftarrow R \setminus \{s\}; u \leftarrow i;$ 
8:     while  $i \leq |w|$  and  $\delta(s, w_i) \neq \perp$  do
9:        $\delta(s, w_i) \leftarrow \text{clone}(\delta(s, w_i)); s \leftarrow \delta(s, w_i); \text{push}(s, P); i \leftarrow i + 1;$ 
10:    end while;
11:    while  $i \leq |w|$  do
12:       $s \leftarrow \text{new state}; s \leftarrow \delta(s, w_i); \text{push}(s, P); i \leftarrow i + 1;$ 
13:    end while;
14:     $F \leftarrow F \cup \{s\}; \text{pop}(P);$ 
15:    while  $P$  not empty do
16:      if  $\exists_{r \in R} r \equiv s$  then
17:        if  $i = u$  and  $i > 0$  then  $R \leftarrow R \setminus \{\text{top}(P)\}; u \leftarrow u - 1$ ; end if;
18:         $\text{delete } s; \delta(\text{top}(P), w_i) \leftarrow r;$ 
19:      else
20:         $R \leftarrow R \cup \{s\}; \text{if } i = u \text{ then break; end if};$ 
21:      end if;
22:       $i \leftarrow i - 1; s \leftarrow \text{pop}(P);$ 
23:    end while;
24:     $\text{reset } P;$ 
25:  end while;
26: end function;
```

# Incremental construction from unsorted data

```
1: function unsorted_construction;
2:   while input not empty do
3:      $s \leftarrow q_0; i \leftarrow 0; w \leftarrow \text{next word}; \text{push}(s, P);$ 
4:     while  $i \leq |w|$  and  $\delta(s, w_i) \neq \perp$  and fanin( $\delta(s, w_i)$ )  $\leq 1$  do
5:        $s \leftarrow \delta(s, w_i); \text{push}(s, P); i \leftarrow i + 1;$ 
6:     end while;
7:      $R \leftarrow R \setminus \{s\}; u \leftarrow i;$ 
8:     while  $i \leq |w|$  and  $\delta(s, w_i) \neq \perp$  do
9:        $\delta(s, w_i) \leftarrow \text{clone}(\delta(s, w_i)); s \leftarrow \delta(s, w_i); \text{push}(s, P); i \leftarrow i + 1;$ 
10:    end while;
11:    while  $i \leq |w|$  do
12:       $s \leftarrow \text{new state}; s \leftarrow \delta(s, w_i); \text{push}(s, P); i \leftarrow i + 1;$ 
13:    end while;
14:     $F \leftarrow F \cup \{s\}; \text{pop}(P);$ 
15:    while  $P$  not empty do
16:      if  $\exists_{r \in R} r \equiv s$  then
17:        if  $i = u$  and  $i > 0$  then  $R \leftarrow R \setminus \{\text{top}(P)\}; u \leftarrow u - 1$ ; end if;
18:        delete  $s$ ;  $\delta(\text{top}(P), w_i) \leftarrow r$ ;
19:      else
20:         $R \leftarrow R \cup \{s\}$ ; if  $i = u$  then break; end if;
21:      end if;
22:       $i \leftarrow i - 1; s \leftarrow \text{pop}(P);$ 
23:    end while;
24:    reset  $P$ ;
25:  end while;
26: end function;
```

# Incremental construction from unsorted data

```
1: function unsorted_construction;
2:   while input not empty do
3:      $s \leftarrow q_0; i \leftarrow 0; w \leftarrow \text{next word}; \text{push}(s, P);$ 
4:     while  $i \leq |w|$  and  $\delta(s, w_i) \neq \perp$  and  $\text{fanin}(\delta(s, w_i)) \leq 1$  do
5:        $s \leftarrow \delta(s, w_i); \text{push}(s, P); i \leftarrow i + 1;$ 
6:     end while;
7:      $R \leftarrow R \setminus \{s\}; u \leftarrow i;$ 
8:     while  $i \leq |w|$  and  $\delta(s, w_i) \neq \perp$  do
9:        $\delta(s, w_i) \leftarrow \text{clone}(\delta(s, w_i)); s \leftarrow \delta(s, w_i); \text{push}(s, P); i \leftarrow i + 1;$ 
10:    end while;
11:    while  $i \leq |w|$  do
12:       $s \leftarrow \text{new state}; s \leftarrow \delta(s, w_i); \text{push}(s, P); i \leftarrow i + 1;$ 
13:    end while;
14:     $F \leftarrow F \cup \{s\}; \text{pop}(P);$ 
15:    while  $P$  not empty do
16:      if  $\exists_{r \in R} r \equiv s$  then
17:        if  $i = u$  and  $i > 0$  then  $R \leftarrow R \setminus \{\text{top}(P)\}; u \leftarrow u - 1$ ; end if;
18:         $\text{delete } s; \delta(\text{top}(P), w_i) \leftarrow r;$ 
19:      else
20:         $R \leftarrow R \cup \{s\}; \text{if } i = u \text{ then break; end if};$ 
21:      end if;
22:       $i \leftarrow i - 1; s \leftarrow \text{pop}(P);$ 
23:    end while;
24:     $\text{reset } P;$ 
25:  end while;
26: end function;
```

# Incremental construction from unsorted data

```
1: function unsorted_construction;
2:   while input not empty do
3:      $s \leftarrow q_0; i \leftarrow 0; w \leftarrow \text{next word}; \text{push}(s, P);$ 
4:     while  $i \leq |w|$  and  $\delta(s, w_i) \neq \perp$  and  $\text{fanin}(\delta(s, w_i)) \leq 1$  do
5:        $s \leftarrow \delta(s, w_i); \text{push}(s, P); i \leftarrow i + 1;$ 
6:     end while;
7:      $R \leftarrow R \setminus \{s\}; u \leftarrow i;$ 
8:     while  $i \leq |w|$  and  $\delta(s, w_i) \neq \perp$  do
9:        $\delta(s, w_i) \leftarrow \text{clone}(\delta(s, w_i)); s \leftarrow \delta(s, w_i); \text{push}(s, P); i \leftarrow i + 1;$ 
10:    end while;
11:    while  $i \leq |w|$  do
12:       $s \leftarrow \text{new state}; s \leftarrow \delta(s, w_i); \text{push}(s, P); i \leftarrow i + 1;$ 
13:    end while;
14:     $F \leftarrow F \cup \{s\}; \text{pop}(P);$ 
15:    while  $P$  not empty do
16:      if  $\exists_{r \in R} r \equiv s$  then
17:        if  $i = u$  and  $i > 0$  then  $R \leftarrow R \setminus \{\text{top}(P)\}; u \leftarrow u - 1$ ; end if;
18:         $\text{delete } s; \delta(\text{top}(P), w_i) \leftarrow r;$ 
19:      else
20:         $R \leftarrow R \cup \{s\}; \text{if } i = u \text{ then break; end if};$ 
21:      end if;
22:       $i \leftarrow i - 1; s \leftarrow \text{pop}(P);$ 
23:    end while;
24:     $\text{reset } P;$ 
25:  end while;
26: end function;
```

# Incremental construction from unsorted data

```
1: function unsorted_construction;
2:   while input not empty do
3:      $s \leftarrow q_0; i \leftarrow 0; w \leftarrow \text{next word}; \text{push}(s, P);$ 
4:     while  $i \leq |w|$  and  $\delta(s, w_i) \neq \perp$  and  $\text{fanin}(\delta(s, w_i)) \leq 1$  do
5:        $s \leftarrow \delta(s, w_i); \text{push}(s, P); i \leftarrow i + 1;$ 
6:     end while;
7:      $R \leftarrow R \setminus \{s\}; u \leftarrow i;$ 
8:     while  $i \leq |w|$  and  $\delta(s, w_i) \neq \perp$  do
9:        $\delta(s, w_i) \leftarrow \text{clone}(\delta(s, w_i)); s \leftarrow \delta(s, w_i); \text{push}(s, P); i \leftarrow i + 1;$ 
10:    end while;
11:    while  $i \leq |w|$  do
12:       $s \leftarrow \text{new state}; s \leftarrow \delta(s, w_i); \text{push}(s, P); i \leftarrow i + 1;$ 
13:    end while;
14:     $F \leftarrow F \cup \{s\}; \text{pop}(P);$ 
15:    while  $P$  not empty do
16:      if  $\exists_{r \in R} r \equiv s$  then
17:        if  $i = u$  and  $i > 0$  then  $R \leftarrow R \setminus \{\text{top}(P)\}; u \leftarrow u - 1$ ; end if;
18:         $\text{delete } s; \delta(\text{top}(P), w_i) \leftarrow r;$ 
19:      else
20:         $R \leftarrow R \cup \{s\}; \text{if } i = u \text{ then break; end if};$ 
21:      end if;
22:       $i \leftarrow i - 1; s \leftarrow \text{pop}(P);$ 
23:    end while;
24:     $\text{reset } P;$ 
25:  end while;
26: end function;
```

# Incremental construction from unsorted data

```
1: function unsorted_construction;
2:   while input not empty do
3:      $s \leftarrow q_0; i \leftarrow 0; w \leftarrow \text{next word}; \text{push}(s, P);$ 
4:     while  $i \leq |w|$  and  $\delta(s, w_i) \neq \perp$  and fanin( $\delta(s, w_i)$ )  $\leq 1$  do
5:        $s \leftarrow \delta(s, w_i); \text{push}(s, P); i \leftarrow i + 1;$ 
6:     end while;
7:      $R \leftarrow R \setminus \{s\}; u \leftarrow i;$ 
8:     while  $i \leq |w|$  and  $\delta(s, w_i) \neq \perp$  do
9:        $\delta(s, w_i) \leftarrow \text{clone}(\delta(s, w_i)); s \leftarrow \delta(s, w_i); \text{push}(s, P); i \leftarrow i + 1;$ 
10:    end while;
11:    while  $i \leq |w|$  do
12:       $s \leftarrow \text{new state}; s \leftarrow \delta(s, w_i); \text{push}(s, P); i \leftarrow i + 1;$ 
13:    end while;
14:     $F \leftarrow F \cup \{s\}; \text{pop}(P);$ 
15:    while  $P$  not empty do
16:      if  $\exists_{r \in R} r \equiv s$  then
17:        if  $i = u$  and  $i > 0$  then  $R \leftarrow R \setminus \{\text{top}(P)\}; u \leftarrow u - 1;$  end if;
18:        delete  $s; \delta(\text{top}(P), w_i) \leftarrow r;$ 
19:      else
20:         $R \leftarrow R \cup \{s\};$  if  $i = u$  then break; end if;
21:      end if;
22:       $i \leftarrow i - 1; s \leftarrow \text{pop}(P);$ 
23:    end while;
24:    reset  $P;$ 
25:  end while;
26: end function;
```

# Incremental construction from unsorted data

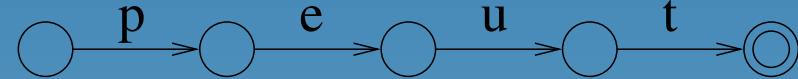
```
1: function unsorted_construction;
2:   while input not empty do
3:      $s \leftarrow q_0; i \leftarrow 0; w \leftarrow \text{next word}; \text{push}(s, P);$ 
4:     while  $i \leq |w|$  and  $\delta(s, w_i) \neq \perp$  and fanin( $\delta(s, w_i)$ )  $\leq 1$  do
5:        $s \leftarrow \delta(s, w_i); \text{push}(s, P); i \leftarrow i + 1;$ 
6:     end while;
7:      $R \leftarrow R \setminus \{s\}; u \leftarrow i;$ 
8:     while  $i \leq |w|$  and  $\delta(s, w_i) \neq \perp$  do
9:        $\delta(s, w_i) \leftarrow \text{clone}(\delta(s, w_i)); s \leftarrow \delta(s, w_i); \text{push}(s, P); i \leftarrow i + 1;$ 
10:    end while;
11:    while  $i \leq |w|$  do
12:       $s \leftarrow \text{new state}; s \leftarrow \delta(s, w_i); \text{push}(s, P); i \leftarrow i + 1;$ 
13:    end while;
14:     $F \leftarrow F \cup \{s\}; \text{pop}(P);$ 
15:    while  $P$  not empty do
16:      if  $\exists_{r \in R} r \equiv s$  then
17:        if  $i = u$  and  $i > 0$  then  $R \leftarrow R \setminus \{\text{top}(P)\}; u \leftarrow u - 1;$  end if;
18:         $\text{delete } s; \delta(\text{top}(P), w_i) \leftarrow r;$ 
19:      else
20:         $R \leftarrow R \cup \{s\}; \text{if } i = u \text{ then break; end if};$ 
21:      end if;
22:       $i \leftarrow i - 1; s \leftarrow \text{pop}(P);$ 
23:    end while;
24:     $\text{reset } P;$ 
25:  end while;
26: end function;
```

# Incremental construction from unsorted data

```
1: function unsorted_construction;
2:   while input not empty do
3:      $s \leftarrow q_0; i \leftarrow 0; w \leftarrow \text{next word}; \text{push}(s, P);$ 
4:     while  $i \leq |w|$  and  $\delta(s, w_i) \neq \perp$  and  $\text{fanin}(\delta(s, w_i)) \leq 1$  do
5:        $s \leftarrow \delta(s, w_i); \text{push}(s, P); i \leftarrow i + 1;$ 
6:     end while;
7:      $R \leftarrow R \setminus \{s\}; u \leftarrow i;$ 
8:     while  $i \leq |w|$  and  $\delta(s, w_i) \neq \perp$  do
9:        $\delta(s, w_i) \leftarrow \text{clone}(\delta(s, w_i)); s \leftarrow \delta(s, w_i); \text{push}(s, P); i \leftarrow i + 1;$ 
10:    end while;
11:    while  $i \leq |w|$  do
12:       $s \leftarrow \text{new state}; s \leftarrow \delta(s, w_i); \text{push}(s, P); i \leftarrow i + 1;$ 
13:    end while;
14:     $F \leftarrow F \cup \{s\}; \text{pop}(P);$ 
15:    while  $P$  not empty do
16:      if  $\exists_{r \in R} r \equiv s$  then
17:        if  $i = u$  and  $i > 0$  then  $R \leftarrow R \setminus \{\text{top}(P)\}; u \leftarrow u - 1$ ; end if;
18:        delete  $s$ ;  $\delta(\text{top}(P), w_i) \leftarrow r$ ;
19:      else
20:         $R \leftarrow R \cup \{s\}$ ; if  $i = u$  then break; end if;
21:      end if;
22:       $i \leftarrow i - 1; s \leftarrow \text{pop}(P);$ 
23:    end while;
24:    reset  $P$ ;
25:  end while;
26: end function;
```

# Incr. constr. from unsorted data – examples

```
1: function unsorted_construction;
2:   while input not empty do
3:      $s \leftarrow q_0; i \leftarrow 0; w \leftarrow \text{next word}; \text{push}(s, P);$ 
4:     while  $i \leq |w|$  and  $\delta(s, w_i) \neq \perp$ 
5:       and  $\text{fanin}(\delta(s, w_i)) \leq 1$  do
6:          $s \leftarrow \delta(s, w_i); \text{push}(s, P); i \leftarrow i + 1;$ 
7:       end while;
8:        $R \leftarrow R \setminus \{s\}; u \leftarrow i;$ 
9:       while  $i \leq |w|$  and  $\delta(s, w_i) \neq \perp$  do
10:         $\delta(s, w_i) \leftarrow \text{clone}(\delta(s, w_i)); s \leftarrow \delta(s, w_i);$ 
11:         $\text{push}(s, P); i \leftarrow i + 1;$ 
12:      end while;
13:      while  $i \leq |w|$  do
14:         $s \leftarrow \text{new state}; s \leftarrow \delta(s, w_i);$ 
15:         $\text{push}(s, P); i \leftarrow i + 1;$ 
16:      end while;
17:       $F \leftarrow F \cup \{s\}; \text{pop}(P);$ 
18:      if  $\exists_{r \in R} r \equiv s$  then
19:        if  $i = u$  and  $i > 0$  then
20:           $R \leftarrow R \setminus \{\text{top}(P)\}; u \leftarrow u - 1;$ 
21:        end if;           peut
22:         $\text{delete } s; \delta(\text{top}(P), w_i) \leftarrow r;$ 
23:      else
24:         $R \leftarrow R \cup \{s\};$ 
25:        if  $i = u$  then break; end if;
26:      end if;
```



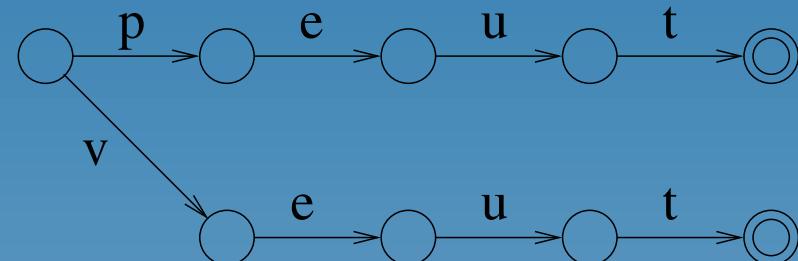
peut

# Incr. constr. from unsorted data – examples

```

1: function unsorted_construction;
2:   while input not empty do
3:      $s \leftarrow q_0; i \leftarrow 0; w \leftarrow \text{next word}; \text{push}(s, P);$ 
4:     while  $i \leq |w|$  and  $\delta(s, w_i) \neq \perp$ 
5:       and  $\text{fanin}(\delta(s, w_i)) \leq 1$  do
6:          $s \leftarrow \delta(s, w_i); \text{push}(s, P); i \leftarrow i + 1;$ 
7:       end while;
8:        $R \leftarrow R \setminus \{s\}; u \leftarrow i;$ 
9:       while  $i \leq |w|$  and  $\delta(s, w_i) \neq \perp$  do
10:         $\delta(s, w_i) \leftarrow \text{clone}(\delta(s, w_i)); s \leftarrow \delta(s, w_i);$ 
11:         $\text{push}(s, P); i \leftarrow i + 1;$ 
12:      end while;
13:      while  $i \leq |w|$  do
14:         $s \leftarrow \text{new state}; s \leftarrow \delta(s, w_i);$ 
15:         $\text{push}(s, P); i \leftarrow i + 1;$ 
16:      end while;
17:       $F \leftarrow F \cup \{s\}; \text{pop}(P);$ 
18:      if  $\exists_{r \in R} r \equiv s$  then
19:        if  $i = u$  and  $i > 0$  then
20:           $R \leftarrow R \setminus \{\text{top}(P)\}; u \leftarrow u - 1;$ 
21:        end if;
22:         $\text{delete } s; \delta(\text{top}(P), w_i) \leftarrow r;$ 
23:      else
24:         $R \leftarrow R \cup \{s\};$ 
25:        if  $i = u$  then  $\text{break}$ ; end if;
26:      end if;
27:       $i \leftarrow i - 1; s \leftarrow \text{pop}(P);$ 
28:    end while;
29:     $\text{reset } P;$ 
30:  end while;
31: end function;

```



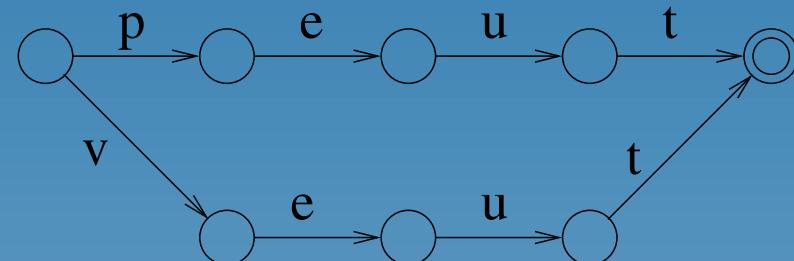
veut

# Incr. constr. from unsorted data – examples

```

1: function unsorted_construction;
2:   while input not empty do
3:      $s \leftarrow q_0; i \leftarrow 0; w \leftarrow \text{next word}; \text{push}(s, P);$ 
4:     while  $i \leq |w|$  and  $\delta(s, w_i) \neq \perp$ 
      and  $\text{fanin}(\delta(s, w_i)) \leq 1$  do
6:        $s \leftarrow \delta(s, w_i); \text{push}(s, P); i \leftarrow i + 1;$ 
7:     end while;
8:      $R \leftarrow R \setminus \{s\}; u \leftarrow i;$ 
9:     while  $i \leq |w|$  and  $\delta(s, w_i) \neq \perp$  do
10:       $\delta(s, w_i) \leftarrow \text{clone}(\delta(s, w_i)); s \leftarrow \delta(s, w_i);$ 
11:       $\text{push}(s, P); i \leftarrow i + 1;$ 
12:    end while;
13:    while  $i \leq |w|$  do
14:       $s \leftarrow \text{new state}; s \leftarrow \delta(s, w_i);$ 
15:       $\text{push}(s, P); i \leftarrow i + 1;$ 
16:    end while;
17:     $F \leftarrow F \cup \{s\}; \text{pop}(P);$ 
18:    if  $\exists_{r \in R} r \equiv s$  then
19:      if  $i = u$  and  $i > 0$  then
20:         $R \leftarrow R \setminus \{\text{top}(P)\}; u \leftarrow u - 1;$ 
21:      end if;
22:       $\text{delete } s; \delta(\text{top}(P), w_i) \leftarrow r;$ 
23:    else
24:       $R \leftarrow R \cup \{s\};$ 
25:      if  $i = u$  then break; end if;
26:    end if;
27:     $i \leftarrow i - 1; s \leftarrow \text{pop}(P);$ 
28:  end while;
29:  reset  $P;$ 
30: end while;
31: end function;

```



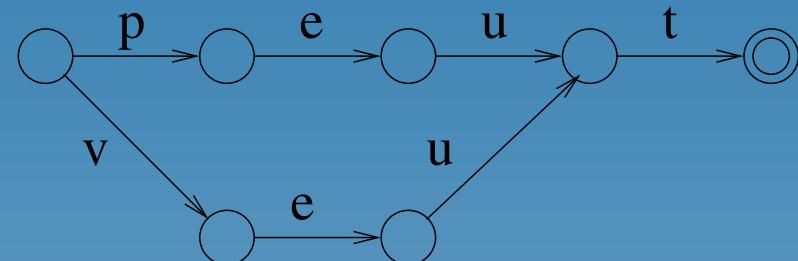
veut

# Incr. constr. from unsorted data – examples

```

1: function unsorted_construction;
2:   while input not empty do
3:      $s \leftarrow q_0; i \leftarrow 0; w \leftarrow \text{next word}; \text{push}(s, P);$ 
4:     while  $i \leq |w|$  and  $\delta(s, w_i) \neq \perp$ 
5:       and  $\text{fanin}(\delta(s, w_i)) \leq 1$  do
6:          $s \leftarrow \delta(s, w_i); \text{push}(s, P); i \leftarrow i + 1;$ 
7:       end while;
8:        $R \leftarrow R \setminus \{s\}; u \leftarrow i;$ 
9:       while  $i \leq |w|$  and  $\delta(s, w_i) \neq \perp$  do
10:         $\delta(s, w_i) \leftarrow \text{clone}(\delta(s, w_i)); s \leftarrow \delta(s, w_i);$ 
11:         $\text{push}(s, P); i \leftarrow i + 1;$ 
12:      end while;
13:      while  $i \leq |w|$  do
14:         $s \leftarrow \text{new state}; s \leftarrow \delta(s, w_i);$ 
15:         $\text{push}(s, P); i \leftarrow i + 1;$ 
16:      end while;
17:       $F \leftarrow F \cup \{s\}; \text{pop}(P);$ 
18:      while  $P$  not empty do
19:        if  $\exists_{r \in R} r \equiv s$  then
20:          if  $i = u$  and  $i > 0$  then
21:             $R \leftarrow R \setminus \{\text{top}(P)\}; u \leftarrow u - 1;$ 
22:          end if;
23:          delete  $s$ ;  $\delta(\text{top}(P), w_i) \leftarrow r;$ 
24:        else
25:           $R \leftarrow R \cup \{s\};$ 
26:          if  $i = u$  then break; end if;
27:        end if;
28:         $i \leftarrow i - 1; s \leftarrow \text{pop}(P);$ 
29:      end while;
30:      reset  $P$ ;
31:    end while;
32:  end function;

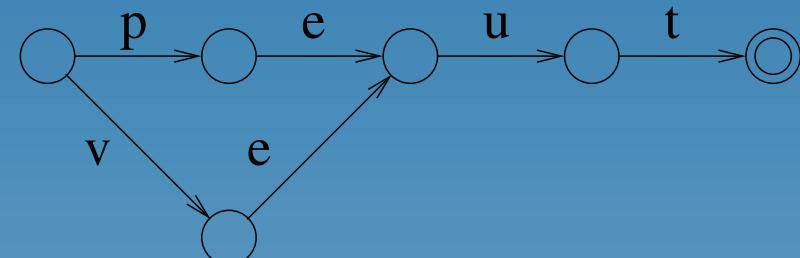
```



veut

# Incr. constr. from unsorted data – examples

```
1: function unsorted_construction;
2:   while input not empty do
3:      $s \leftarrow q_0; i \leftarrow 0; w \leftarrow \text{next word}; \text{push}(s, P);$ 
4:     while  $i \leq |w|$  and  $\delta(s, w_i) \neq \perp$ 
5:       and  $\text{fanin}(\delta(s, w_i)) \leq 1$  do
6:          $s \leftarrow \delta(s, w_i); \text{push}(s, P); i \leftarrow i + 1;$ 
7:       end while;
8:        $R \leftarrow R \setminus \{s\}; u \leftarrow i;$ 
9:       while  $i \leq |w|$  and  $\delta(s, w_i) \neq \perp$  do
10:         $\delta(s, w_i) \leftarrow \text{clone}(\delta(s, w_i)); s \leftarrow \delta(s, w_i);$ 
11:         $\text{push}(s, P); i \leftarrow i + 1;$ 
12:      end while;
13:      while  $i \leq |w|$  do
14:         $s \leftarrow \text{new state}; s \leftarrow \delta(s, w_i);$ 
15:         $\text{push}(s, P); i \leftarrow i + 1;$ 
16:      end while;
17:       $F \leftarrow F \cup \{s\}; \text{pop}(P);$ 
18:      while  $P$  not empty do
19:        if  $\exists_{r \in R} r \equiv s$  then
20:          if  $i = u$  and  $i > 0$  then
21:             $R \leftarrow R \setminus \{\text{top}(P)\}; u \leftarrow u - 1;$ 
22:          end if;
23:           $\text{delete } s; \delta(\text{top}(P), w_i) \leftarrow r;$ 
24:        else
25:           $R \leftarrow R \cup \{s\};$ 
26:          if  $i = u$  then break; end if;
```



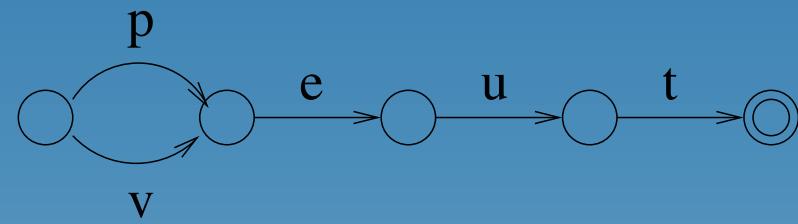
veut

# Incr. constr. from unsorted data – examples

```

1: function unsorted_construction;
2:   while input not empty do
3:      $s \leftarrow q_0; i \leftarrow 0; w \leftarrow \text{next word}; \text{push}(s, P);$ 
4:     while  $i \leq |w|$  and  $\delta(s, w_i) \neq \perp$ 
5:       and  $\text{fanin}(\delta(s, w_i)) \leq 1$  do
6:          $s \leftarrow \delta(s, w_i); \text{push}(s, P); i \leftarrow i + 1;$ 
7:       end while;
8:        $R \leftarrow R \setminus \{s\}; u \leftarrow i;$ 
9:       while  $i \leq |w|$  and  $\delta(s, w_i) \neq \perp$  do
10:         $\delta(s, w_i) \leftarrow \text{clone}(\delta(s, w_i)); s \leftarrow \delta(s, w_i);$ 
11:         $\text{push}(s, P); i \leftarrow i + 1;$ 
12:      end while;
13:      while  $i \leq |w|$  do
14:         $s \leftarrow \text{new state}; s \leftarrow \delta(s, w_i);$ 
15:         $\text{push}(s, P); i \leftarrow i + 1;$ 
16:      end while;
17:       $F \leftarrow F \cup \{s\}; \text{pop}(P);$ 
18:      while  $P$  not empty do
19:        if  $\exists_{r \in R} r \equiv s$  then
20:          if  $i = u$  and  $i > 0$  then
21:             $R \leftarrow R \setminus \{\text{top}(P)\}; u \leftarrow u - 1;$ 
22:          end if;
23:          delete  $s$ ;  $\delta(\text{top}(P), w_i) \leftarrow r;$ 
24:        else
25:           $R \leftarrow R \cup \{s\};$ 
26:          if  $i = u$  then break; end if;
27:        end if;
28:         $i \leftarrow i - 1; s \leftarrow \text{pop}(P);$ 
29:      end while;
30:      reset  $P$ ;
31:    end while;
32:  end function;

```



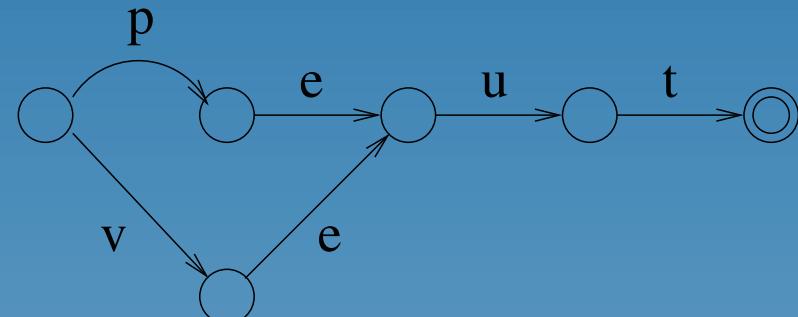
veut

# Incr. constr. from unsorted data – examples

```

1: function unsorted_construction;
2:   while input not empty do
3:      $s \leftarrow q_0; i \leftarrow 0; w \leftarrow \text{next word}; \text{push}(s, P);$ 
4:     while  $i \leq |w|$  and  $\delta(s, w_i) \neq \perp$ 
5:       and  $\text{fanin}(\delta(s, w_i)) \leq 1$  do
6:          $s \leftarrow \delta(s, w_i); \text{push}(s, P); i \leftarrow i + 1;$ 
7:       end while;
8:        $R \leftarrow R \setminus \{s\}; u \leftarrow i;$ 
9:       while  $i \leq |w|$  and  $\delta(s, w_i) \neq \perp$  do
10:         $\delta(s, w_i) \leftarrow \text{clone}(\delta(s, w_i)); s \leftarrow \delta(s, w_i);$ 
11:         $\text{push}(s, P); i \leftarrow i + 1;$ 
12:      end while;
13:      while  $i \leq |w|$  do
14:         $s \leftarrow \text{new state}; s \leftarrow \delta(s, w_i);$ 
15:         $\text{push}(s, P); i \leftarrow i + 1;$ 
16:      end while;
17:       $F \leftarrow F \cup \{s\}; \text{pop}(P);$ 
18:      while  $P$  not empty do
19:        if  $\exists_{r \in R} r \equiv s$  then
20:          if  $i = u$  and  $i > 0$  then
21:             $R \leftarrow R \setminus \{\text{top}(P)\}; u \leftarrow u - 1;$ 
22:          end if;
23:          delete  $s$ ;  $\delta(\text{top}(P), w_i) \leftarrow r;$ 
24:        else
25:           $R \leftarrow R \cup \{s\};$ 
26:          if  $i = u$  then break; end if;
27:        end if;
28:         $i \leftarrow i - 1; s \leftarrow \text{pop}(P);$ 
29:      end while;
30:      reset  $P$ ;
31:    end while;
32:  end function;

```



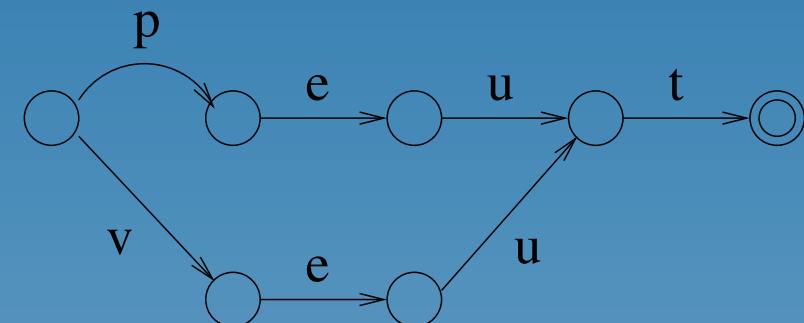
veux

# Incr. constr. from unsorted data – examples

```

1: function unsorted_construction;
2:   while input not empty do
3:      $s \leftarrow q_0; i \leftarrow 0; w \leftarrow \text{next word}; \text{push}(s, P);$ 
4:     while  $i \leq |w|$  and  $\delta(s, w_i) \neq \perp$ 
5:       and  $\text{fanin}(\delta(s, w_i)) \leq 1$  do
6:          $s \leftarrow \delta(s, w_i); \text{push}(s, P); i \leftarrow i + 1;$ 
7:       end while;
8:        $R \leftarrow R \setminus \{s\}; u \leftarrow i;$ 
9:       while  $i \leq |w|$  and  $\delta(s, w_i) \neq \perp$  do
10:         $\delta(s, w_i) \leftarrow \text{clone}(\delta(s, w_i)); s \leftarrow \delta(s, w_i);$ 
11:         $\text{push}(s, P); i \leftarrow i + 1;$ 
12:      end while;
13:      while  $i \leq |w|$  do
14:         $s \leftarrow \text{new state}; s \leftarrow \delta(s, w_i);$ 
15:         $\text{push}(s, P); i \leftarrow i + 1;$ 
16:      end while;
17:       $F \leftarrow F \cup \{s\}; \text{pop}(P);$ 
18:      while  $P$  not empty do
19:        if  $\exists_{r \in R} r \equiv s$  then
20:          if  $i = u$  and  $i > 0$  then
21:             $R \leftarrow R \setminus \{\text{top}(P)\}; u \leftarrow u - 1;$ 
22:          end if;
23:          delete  $s$ ;  $\delta(\text{top}(P), w_i) \leftarrow r;$ 
24:        else
25:           $R \leftarrow R \cup \{s\};$ 
26:          if  $i = u$  then break; end if;

```



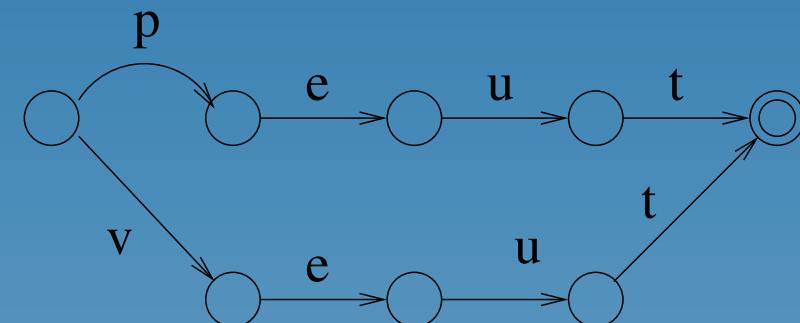
veux

# Incr. constr. from unsorted data – examples

```

1: function unsorted_construction;
2:   while input not empty do
3:      $s \leftarrow q_0; i \leftarrow 0; w \leftarrow \text{next word}; \text{push}(s, P);$ 
4:     while  $i \leq |w|$  and  $\delta(s, w_i) \neq \perp$ 
5:       and  $\text{fanin}(\delta(s, w_i)) \leq 1$  do
6:          $s \leftarrow \delta(s, w_i); \text{push}(s, P); i \leftarrow i + 1;$ 
7:       end while;
8:        $R \leftarrow R \setminus \{s\}; u \leftarrow i;$ 
9:       while  $i \leq |w|$  and  $\delta(s, w_i) \neq \perp$  do
10:         $\delta(s, w_i) \leftarrow \text{clone}(\delta(s, w_i)); s \leftarrow \delta(s, w_i);$ 
11:         $\text{push}(s, P); i \leftarrow i + 1;$ 
12:      end while;
13:      while  $i \leq |w|$  do
14:         $s \leftarrow \text{new state}; s \leftarrow \delta(s, w_i);$ 
15:         $\text{push}(s, P); i \leftarrow i + 1;$ 
16:      end while;
17:       $F \leftarrow F \cup \{s\}; \text{pop}(P);$ 
18:      while  $P$  not empty do
19:        if  $\exists_{r \in R} r \equiv s$  then
20:          if  $i = u$  and  $i > 0$  then
21:             $R \leftarrow R \setminus \{\text{top}(P)\}; u \leftarrow u - 1;$ 
22:          end if;
23:          delete  $s$ ;  $\delta(\text{top}(P), w_i) \leftarrow r;$ 
24:        else
25:           $R \leftarrow R \cup \{s\};$ 
26:          if  $i = u$  then break; end if;

```



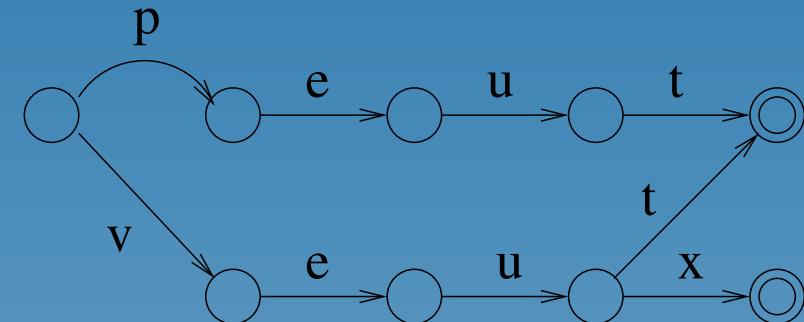
veux

# Incr. constr. from unsorted data – examples

```

1: function unsorted_construction;
2:   while input not empty do
3:      $s \leftarrow q_0; i \leftarrow 0; w \leftarrow \text{next word}; \text{push}(s, P);$ 
4:     while  $i \leq |w|$  and  $\delta(s, w_i) \neq \perp$ 
5:       and  $\text{fanin}(\delta(s, w_i)) \leq 1$  do
6:          $s \leftarrow \delta(s, w_i); \text{push}(s, P); i \leftarrow i + 1;$ 
7:       end while;
8:        $R \leftarrow R \setminus \{s\}; u \leftarrow i;$ 
9:       while  $i \leq |w|$  and  $\delta(s, w_i) \neq \perp$  do
10:         $\delta(s, w_i) \leftarrow \text{clone}(\delta(s, w_i)); s \leftarrow \delta(s, w_i);$ 
11:         $\text{push}(s, P); i \leftarrow i + 1;$ 
12:      end while;
13:      while  $i \leq |w|$  do
14:         $s \leftarrow \text{new state}; s \leftarrow \delta(s, w_i);$ 
15:         $\text{push}(s, P); i \leftarrow i + 1;$ 
16:      end while;
17:       $F \leftarrow F \cup \{s\}; \text{pop}(P);$ 
18:      while  $P$  not empty do
19:        if  $\exists_{r \in R} r \equiv s$  then
20:          if  $i = u$  and  $i > 0$  then
21:             $R \leftarrow R \setminus \{\text{top}(P)\}; u \leftarrow u - 1;$ 
22:          end if;
23:          delete  $s$ ;  $\delta(\text{top}(P), w_i) \leftarrow r;$ 
24:        else
25:           $R \leftarrow R \cup \{s\};$ 
26:          if  $i = u$  then break; end if;

```



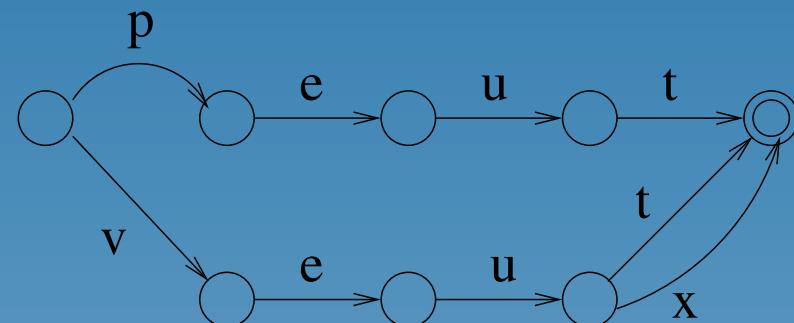
veux

# Incr. constr. from unsorted data – examples

```

1: function unsorted_construction;
2:   while input not empty do
3:      $s \leftarrow q_0; i \leftarrow 0; w \leftarrow \text{next word}; \text{push}(s, P);$ 
4:     while  $i \leq |w|$  and  $\delta(s, w_i) \neq \perp$ 
5:       and  $\text{fanin}(\delta(s, w_i)) \leq 1$  do
6:          $s \leftarrow \delta(s, w_i); \text{push}(s, P); i \leftarrow i + 1;$ 
7:       end while;
8:        $R \leftarrow R \setminus \{s\}; u \leftarrow i;$ 
9:       while  $i \leq |w|$  and  $\delta(s, w_i) \neq \perp$  do
10:         $\delta(s, w_i) \leftarrow \text{clone}(\delta(s, w_i)); s \leftarrow \delta(s, w_i);$ 
11:         $\text{push}(s, P); i \leftarrow i + 1;$ 
12:      end while;
13:      while  $i \leq |w|$  do
14:         $s \leftarrow \text{new state}; s \leftarrow \delta(s, w_i);$ 
15:         $\text{push}(s, P); i \leftarrow i + 1;$ 
16:      end while;
17:       $F \leftarrow F \cup \{s\}; \text{pop}(P);$ 
18:      while  $P$  not empty do
19:        if  $\exists_{r \in R} r \equiv s$  then
20:          if  $i = u$  and  $i > 0$  then
21:             $R \leftarrow R \setminus \{\text{top}(P)\}; u \leftarrow u - 1;$ 
22:          end if;
23:          delete  $s$ ;  $\delta(\text{top}(P), w_i) \leftarrow r;$ 
24:        else
25:           $R \leftarrow R \cup \{s\};$ 
26:          if  $i = u$  then break; end if;
27:        end if;
28:         $i \leftarrow i - 1; s \leftarrow \text{pop}(P);$ 
29:      end while;
30:      reset  $P$ ;
31:    end while;
32:  end function;

```



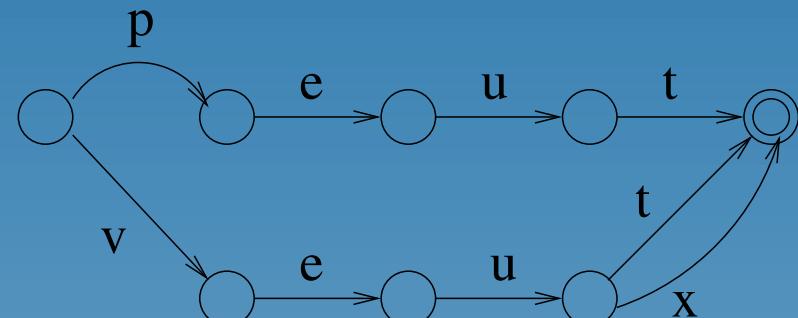
veux

# Incr. constr. from unsorted data – examples

```

1: function unsorted_construction;
2:   while input not empty do
3:      $s \leftarrow q_0; i \leftarrow 0; w \leftarrow \text{next word}; \text{push}(s, P);$ 
4:     while  $i \leq |w|$  and  $\delta(s, w_i) \neq \perp$ 
5:       and  $\text{fanin}(\delta(s, w_i)) \leq 1$  do
6:          $s \leftarrow \delta(s, w_i); \text{push}(s, P); i \leftarrow i + 1;$ 
7:       end while;
8:        $R \leftarrow R \setminus \{s\}; u \leftarrow i;$ 
9:       while  $i \leq |w|$  and  $\delta(s, w_i) \neq \perp$  do
10:         $\delta(s, w_i) \leftarrow \text{clone}(\delta(s, w_i)); s \leftarrow \delta(s, w_i);$ 
11:         $\text{push}(s, P); i \leftarrow i + 1;$ 
12:      end while;
13:      while  $i \leq |w|$  do
14:         $s \leftarrow \text{new state}; s \leftarrow \delta(s, w_i);$ 
15:         $\text{push}(s, P); i \leftarrow i + 1;$ 
16:      end while;
17:       $F \leftarrow F \cup \{s\}; \text{pop}(P);$ 
18:      if  $\exists_{r \in R} r \equiv s$  then
19:        if  $i = u$  and  $i > 0$  then
20:           $R \leftarrow R \setminus \{\text{top}(P)\}; u \leftarrow u - 1;$ 
21:        end if;
22:         $\text{delete } s; \delta(\text{top}(P), w_i) \leftarrow r;$ 
23:      else
24:         $R \leftarrow R \cup \{s\};$ 
25:        if  $i = u$  then break; end if;
26:      end if;
27:       $i \leftarrow i - 1; s \leftarrow \text{pop}(P);$ 
28:    end while;
29:     $\text{reset } P;$ 
30:  end while;
31: end function;

```



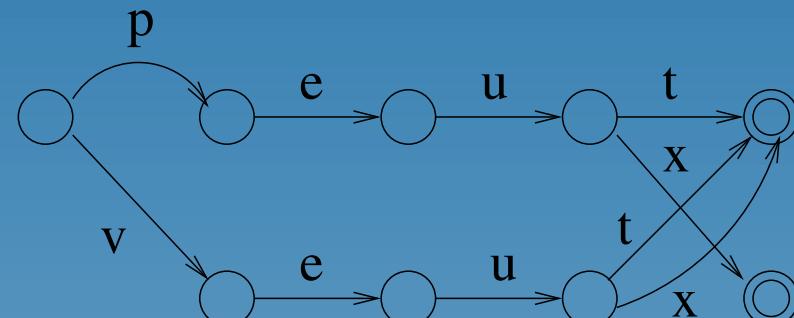
peux

# Incr. constr. from unsorted data – examples

```

1: function unsorted_construction;
2:   while input not empty do
3:      $s \leftarrow q_0; i \leftarrow 0; w \leftarrow \text{next word}; \text{push}(s, P);$ 
4:     while  $i \leq |w|$  and  $\delta(s, w_i) \neq \perp$ 
5:       and  $\text{fanin}(\delta(s, w_i)) \leq 1$  do
6:          $s \leftarrow \delta(s, w_i); \text{push}(s, P); i \leftarrow i + 1;$ 
7:       end while;
8:        $R \leftarrow R \setminus \{s\}; u \leftarrow i;$ 
9:       while  $i \leq |w|$  and  $\delta(s, w_i) \neq \perp$  do
10:         $\delta(s, w_i) \leftarrow \text{clone}(\delta(s, w_i)); s \leftarrow \delta(s, w_i);$ 
11:         $\text{push}(s, P); i \leftarrow i + 1;$ 
12:      end while;
13:      while  $i \leq |w|$  do
14:         $s \leftarrow \text{new state}; s \leftarrow \delta(s, w_i);$ 
15:         $\text{push}(s, P); i \leftarrow i + 1;$ 
16:      end while;
17:       $F \leftarrow F \cup \{s\}; \text{pop}(P);$ 
18:      if  $\exists_{r \in R} r \equiv s$  then
19:        if  $i = u$  and  $i > 0$  then
20:           $R \leftarrow R \setminus \{\text{top}(P)\}; u \leftarrow u - 1;$ 
21:        end if;
22:         $\text{delete } s; \delta(\text{top}(P), w_i) \leftarrow r;$ 
23:      else
24:         $R \leftarrow R \cup \{s\};$ 
25:        if  $i = u$  then break; end if;
26:      end if;
27:       $i \leftarrow i - 1; s \leftarrow \text{pop}(P);$ 
28:    end while;
29:     $\text{reset } P;$ 
30:  end while;
31: end function;

```



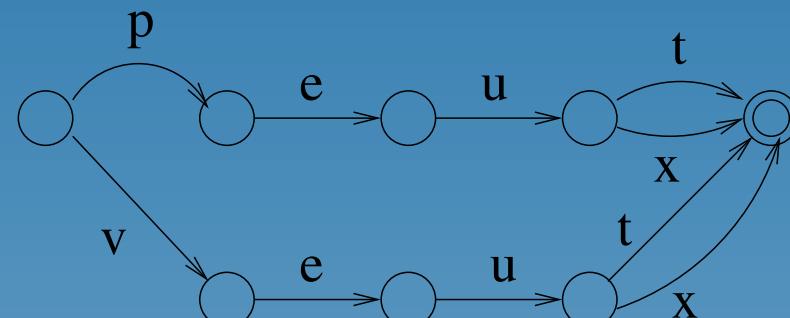
peux

# Incr. constr. from unsorted data – examples

```

1: function unsorted_construction;
2:   while input not empty do
3:      $s \leftarrow q_0; i \leftarrow 0; w \leftarrow \text{next word}; \text{push}(s, P);$ 
4:     while  $i \leq |w|$  and  $\delta(s, w_i) \neq \perp$ 
5:       and  $\text{fanin}(\delta(s, w_i)) \leq 1$  do
6:          $s \leftarrow \delta(s, w_i); \text{push}(s, P); i \leftarrow i + 1;$ 
7:       end while;
8:        $R \leftarrow R \setminus \{s\}; u \leftarrow i;$ 
9:       while  $i \leq |w|$  and  $\delta(s, w_i) \neq \perp$  do
10:         $\delta(s, w_i) \leftarrow \text{clone}(\delta(s, w_i)); s \leftarrow \delta(s, w_i);$ 
11:         $\text{push}(s, P); i \leftarrow i + 1;$ 
12:      end while;
13:      while  $i \leq |w|$  do
14:         $s \leftarrow \text{new state}; s \leftarrow \delta(s, w_i);$ 
15:         $\text{push}(s, P); i \leftarrow i + 1;$ 
16:      end while;
17:       $F \leftarrow F \cup \{s\}; \text{pop}(P);$ 
18:      while  $P$  not empty do
19:        if  $\exists_{r \in R} r \equiv s$  then
20:          if  $i = u$  and  $i > 0$  then
21:             $R \leftarrow R \setminus \{\text{top}(P)\}; u \leftarrow u - 1;$ 
22:          end if;
23:          delete  $s$ ;  $\delta(\text{top}(P), w_i) \leftarrow r;$ 
24:        else
25:           $R \leftarrow R \cup \{s\};$ 
26:          if  $i = u$  then break; end if;
27:        end if;
28:         $i \leftarrow i - 1; s \leftarrow \text{pop}(P);$ 
29:      end while;
30:      reset  $P$ ;
31:    end while;
32:  end function;

```



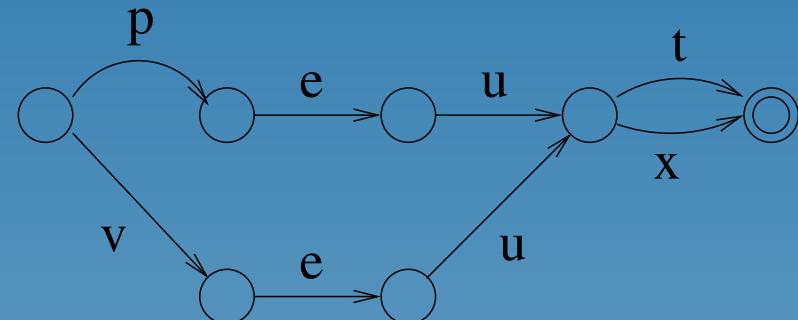
veux

# Incr. constr. from unsorted data – examples

```

1: function unsorted_construction;
2:   while input not empty do
3:      $s \leftarrow q_0; i \leftarrow 0; w \leftarrow \text{next word}; \text{push}(s, P);$ 
4:     while  $i \leq |w|$  and  $\delta(s, w_i) \neq \perp$ 
5:       and  $\text{fanin}(\delta(s, w_i)) \leq 1$  do
6:          $s \leftarrow \delta(s, w_i); \text{push}(s, P); i \leftarrow i + 1;$ 
7:       end while;
8:        $R \leftarrow R \setminus \{s\}; u \leftarrow i;$ 
9:       while  $i \leq |w|$  and  $\delta(s, w_i) \neq \perp$  do
10:         $\delta(s, w_i) \leftarrow \text{clone}(\delta(s, w_i)); s \leftarrow \delta(s, w_i);$ 
11:         $\text{push}(s, P); i \leftarrow i + 1;$ 
12:      end while;
13:      while  $i \leq |w|$  do
14:         $s \leftarrow \text{new state}; s \leftarrow \delta(s, w_i);$ 
15:         $\text{push}(s, P); i \leftarrow i + 1;$ 
16:      end while;
17:       $F \leftarrow F \cup \{s\}; \text{pop}(P);$ 
18:      while  $P$  not empty do
19:        if  $\exists_{r \in R} r \equiv s$  then
20:          if  $i = u$  and  $i > 0$  then
21:             $R \leftarrow R \setminus \{\text{top}(P)\}; u \leftarrow u - 1;$ 
22:          end if;
23:          delete  $s$ ;  $\delta(\text{top}(P), w_i) \leftarrow r;$ 
24:        else
25:           $R \leftarrow R \cup \{s\};$ 
26:          if  $i = u$  then break; end if;
27:        end if;
28:         $i \leftarrow i - 1; s \leftarrow \text{pop}(P);$ 
29:      end while;
30:      reset  $P$ ;
31:    end while;
32:  end function;

```



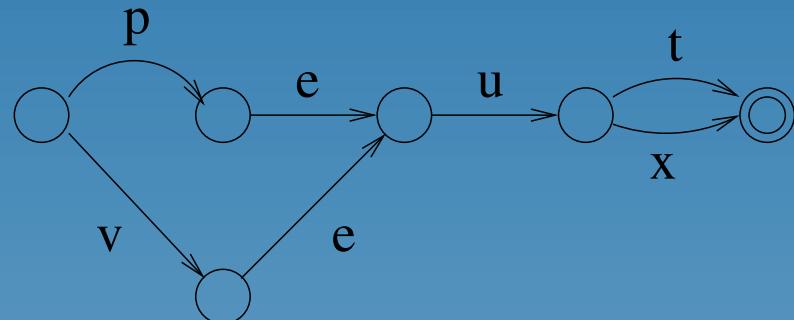
veux

# Incr. constr. from unsorted data – examples

```

1: function unsorted_construction;
2:   while input not empty do
3:      $s \leftarrow q_0; i \leftarrow 0; w \leftarrow \text{next word}; \text{push}(s, P);$ 
4:     while  $i \leq |w|$  and  $\delta(s, w_i) \neq \perp$ 
5:       and  $\text{fanin}(\delta(s, w_i)) \leq 1$  do
6:          $s \leftarrow \delta(s, w_i); \text{push}(s, P); i \leftarrow i + 1;$ 
7:       end while;
8:        $R \leftarrow R \setminus \{s\}; u \leftarrow i;$ 
9:       while  $i \leq |w|$  and  $\delta(s, w_i) \neq \perp$  do
10:         $\delta(s, w_i) \leftarrow \text{clone}(\delta(s, w_i)); s \leftarrow \delta(s, w_i);$ 
11:         $\text{push}(s, P); i \leftarrow i + 1;$ 
12:      end while;
13:      while  $i \leq |w|$  do
14:         $s \leftarrow \text{new state}; s \leftarrow \delta(s, w_i);$ 
15:         $\text{push}(s, P); i \leftarrow i + 1;$ 
16:      end while;
17:       $F \leftarrow F \cup \{s\}; \text{pop}(P);$ 
18:      while  $P$  not empty do
19:        if  $\exists_{r \in R} r \equiv s$  then
20:          if  $i = u$  and  $i > 0$  then
21:             $R \leftarrow R \setminus \{\text{top}(P)\}; u \leftarrow u - 1;$ 
22:          end if;
23:          delete  $s$ ;  $\delta(\text{top}(P), w_i) \leftarrow r;$ 
24:        else
25:           $R \leftarrow R \cup \{s\};$ 
26:          if  $i = u$  then break; end if;
27:        end if;
28:         $i \leftarrow i - 1; s \leftarrow \text{pop}(P);$ 
29:      end while;
30:      reset  $P$ ;
31:    end while;
32:  end function;

```



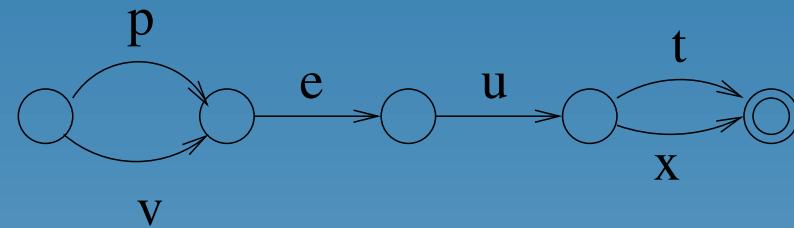
veux

# Incr. constr. from unsorted data – examples

```

1: function unsorted_construction;
2:   while input not empty do
3:      $s \leftarrow q_0; i \leftarrow 0; w \leftarrow \text{next word}; \text{push}(s, P);$ 
4:     while  $i \leq |w|$  and  $\delta(s, w_i) \neq \perp$ 
5:       and  $\text{fanin}(\delta(s, w_i)) \leq 1$  do
6:          $s \leftarrow \delta(s, w_i); \text{push}(s, P); i \leftarrow i + 1;$ 
7:       end while;
8:        $R \leftarrow R \setminus \{s\}; u \leftarrow i;$ 
9:       while  $i \leq |w|$  and  $\delta(s, w_i) \neq \perp$  do
10:         $\delta(s, w_i) \leftarrow \text{clone}(\delta(s, w_i)); s \leftarrow \delta(s, w_i);$ 
11:         $\text{push}(s, P); i \leftarrow i + 1;$ 
12:      end while;
13:      while  $i \leq |w|$  do
14:         $s \leftarrow \text{new state}; s \leftarrow \delta(s, w_i);$ 
15:         $\text{push}(s, P); i \leftarrow i + 1;$ 
16:      end while;
17:       $F \leftarrow F \cup \{s\}; \text{pop}(P);$ 
18:      while  $P$  not empty do
19:        if  $\exists_{r \in R} r \equiv s$  then
20:          if  $i = u$  and  $i > 0$  then
21:             $R \leftarrow R \setminus \{\text{top}(P)\}; u \leftarrow u - 1;$ 
22:          end if;
23:          delete  $s$ ;  $\delta(\text{top}(P), w_i) \leftarrow r;$ 
24:        else
25:           $R \leftarrow R \cup \{s\};$ 
26:          if  $i = u$  then break; end if;
27:        end if;
28:         $i \leftarrow i - 1; s \leftarrow \text{pop}(P);$ 
29:      end while;
30:      reset  $P$ ;
31:    end while;
32:  end function;

```



peux

# Complexity and performance

- Both algorithms keep intermediate automata (almost) minimal
- Both run in time proportional to input data size
- The algorithm for sorted data is faster but less flexible
- Traditional algorithms are slower and use much more memory
- There are extensions of both algorithms to the case off adding words to a cyclic automaton