5th lecture overview

Lecture overview

Properties of regular languages

Properties of regular languages

$N_1 = \{ wcw^R \}$

Properties of regular languages

$N_1 = \{ wcw^R \}$

$N_2 = \{ 0^{i^2} \}$

Properties of regular languages

$N_1 = \{ wcw^R \}$

$N_3 = \{ a^i b^i \}$

$N_2 = \{ 0^{i^2} \}$

Properties of regular languages

$N_1 = \{\ wcw^R\ \}$

$N_3 = \{\ a^i b^i\ \}$

$N_2 = \{\ 0^{i^2}\ \}$

$N_4 = \{\ 0^i\ \}$, *i* **prime**

Properties of regular languages

Language is not regular

$N_1 = \{ wcw^R \}$

$N_3 = \{ a^i b^i \}$

$N_2 = \{ 0^{i^2} \}$

$N_4 = \{ 0^i \}$, i **prime**

Properties of regular languages

Language is not regular

=

$N_1 = \{\ wcw^R\ \}$

$N_3 = \{\ a^i b^i\ \}$

$N_2 = \{\ 0^{i^2}\ \}$

$N_4 = \{\ 0^i\ \}$, i **prime**

Properties of regular languages

Language is not regular

=

There is no finite automaton which accepts the language

$N_1 = \{\, wcw^R \,\}$

$N_3 = \{\, a^i b^i \,\}$

$N_2 = \{\, 0^{i^2} \,\}$

$N_4 = \{\, 0^i \,\}$, i **prime**

Properties of regular languages

Language is not regular
=
There is no finite automaton which accepts the language

$N_1 = \{\ wcw^R\ \}$

$N_3 = \{\ a^i b^i\ \}$

$N_2 = \{\ 0^{i^2}\ \}$

$N_4 = \{\ 0^i\ \}$, *i* **prime**

2^{Σ^*} set of all languages over the alphabet Σ

Properties of regular languages

Language is not regular

=

There is no finite automaton which accepts the language

$N_1 = \{\, wcw^R \,\}$

$N_3 = \{\, a^i b^i \,\}$

$N_2 = \{\, 0^{i^2} \,\}$

$N_4 = \{\, 0^i \,\}$, *i* **prime**

2^{Σ^*} ***set of all languages over the alphabet*** Σ

Regular languages

$RL \subset 2^{\Sigma^*}$

Properties of regular languages

Language is not regular

=

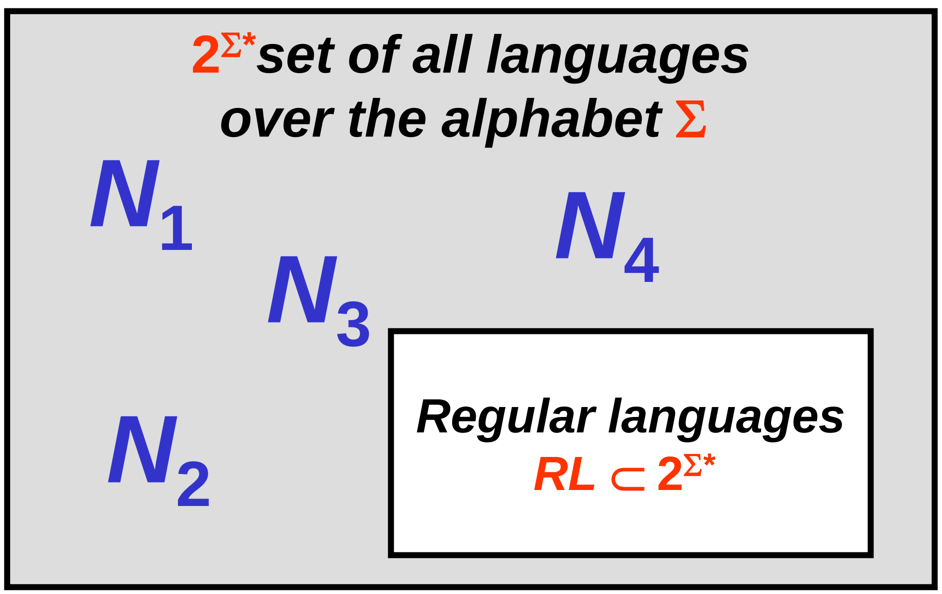
There is no finite automaton which accepts the language

$N_1 = \{ wcw^R \}$

$N_3 = \{ a^i b^i \}$

$N_2 = \{ 0^{i^2} \}$

$N_4 = \{ 0^i \}$, i **prime**



Properties of regular languages

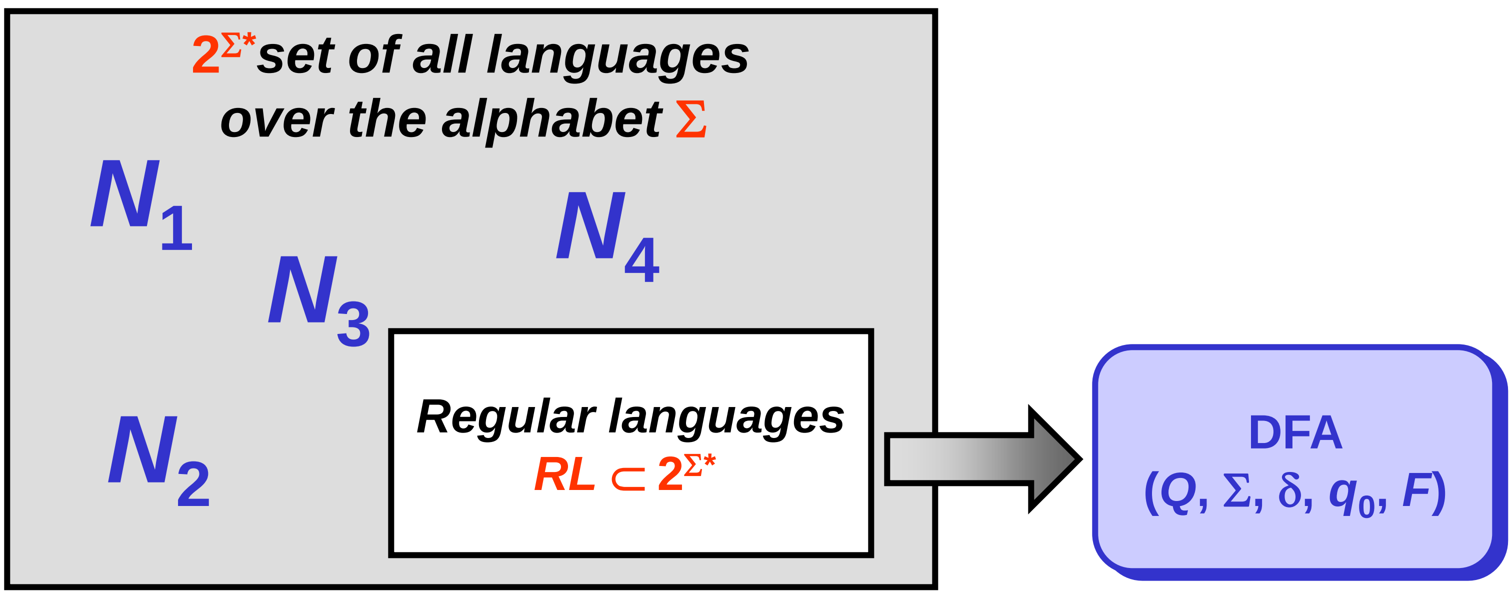
Language is not regular

=

There is no finite automaton which accepts the language

$N_1 = \{ wcw^R \}$ $\qquad$ $N_3 = \{ a^i b^i \}$

$N_2 = \{ 0^{i^2} \}$ $\qquad$ $N_4 = \{ 0^i \}$, i prime



Closure properties of regular languages

A language class is *closed under a certain operation*

if applying this operation to any language(s) in the class gives a language in the same class.

Closure properties of regular languages

Closure properties of regular languages

- **Union, concatenation and Kleene's operator**

Closure properties of regular languages

- Union, concatenation and Kleene's operator
  - Based on the definition of regular expressions

Closure properties of regular languages

- **Union, concatenation and Kleene's operator**
  - **Based on the definition of regular expressions**
  - **There is an algorithm to construct an ε-NFA from a regular expression**

Closure properties of regular languages

- **Union, concatenation and Kleene's operator**
  - **Based on the definition of regular expressions**
  - **There is an algorithm to construct an ε-NFA from a regular expression**
- **Complement**

Closure properties of regular languages

- **Union, concatenation and Kleene's operator**

  - **Based on the definition of regular expressions**
  - **There is an algorithm to construct an ε-NFA from a regular expression**

- **Complement**

  - **DFA $M=(Q, \Sigma, \delta, q_0, F)$,** $L(M)$

Closure properties of regular languages

- **Union, concatenation and Kleene's operator**

  - **Based on the definition of regular expressions**
  - **There is an algorithm to construct an ε-NFA from a regular expression**

- **Complement**

  - **DFA $M=(Q, \Sigma, \delta, q_0, F)$,** $\quad L(M)$
  - **DFA $M'=(Q, \Sigma, \delta, q_0, Q \backslash F)$,** $\quad L(M)^c$

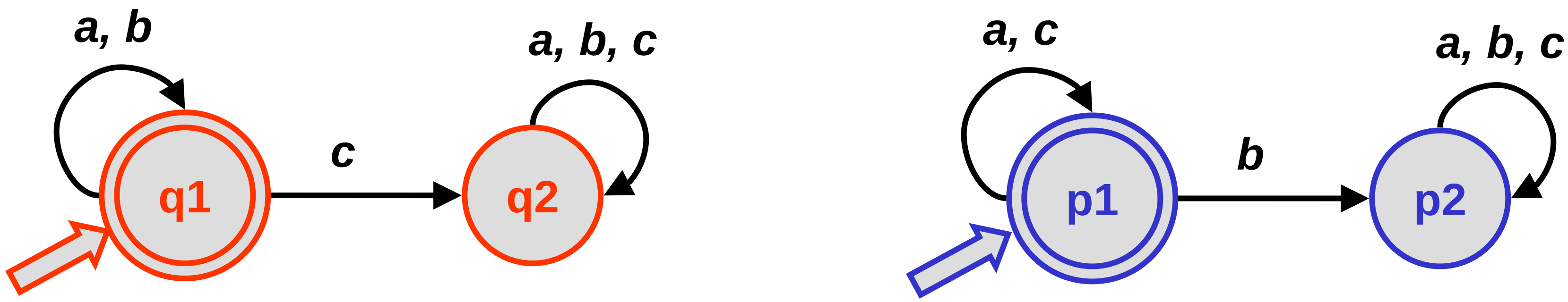
Closure properties of regular languages

- Union, concatenation and Kleene's operator
  - Based on the definition of regular expressions
  - There is an algorithm to construct an ε-NFA from a regular expression
- Complement
  - DFA $M=(Q, \Sigma, \delta, q_0, F)$, $L(M)$
  - DFA $M'=(Q, \Sigma, \delta, q_0, Q \backslash F)$, $L(M)^c$
- Intersection

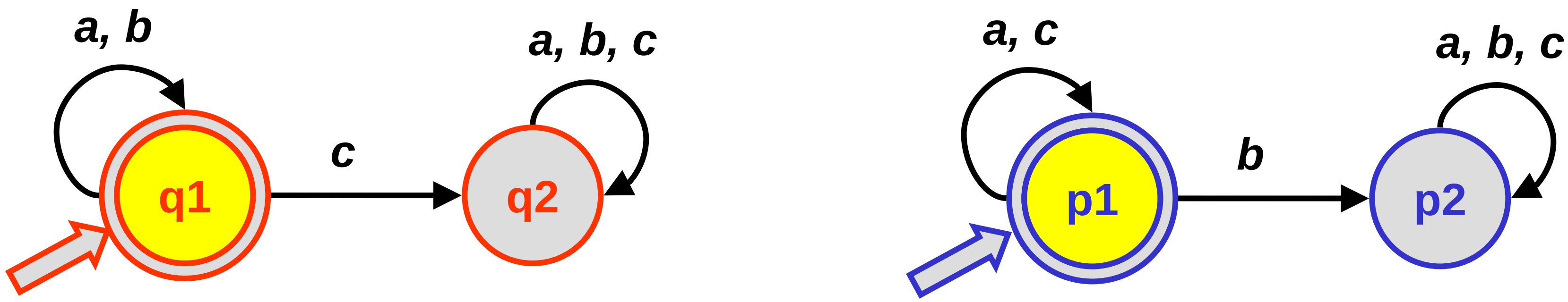
Closure properties of regular languages

- **Union, concatenation and Kleene's operator**
  - **Based on the definition of regular expressions**
  - **There is an algorithm to construct an ε-NFA from a regular expression**
- **Complement**
  - **DFA $M=(Q, \Sigma, \delta, q_0, F)$, $L(M)$**
  - **DFA $M'=(Q, \Sigma, \delta, q_0, Q \setminus F)$, $L(M)^c$**
- **Intersection**
  - **De Morgan's rule: $L \cap N = ((L \cap N)^c)^c = (L^c \cup N^c)^c$**

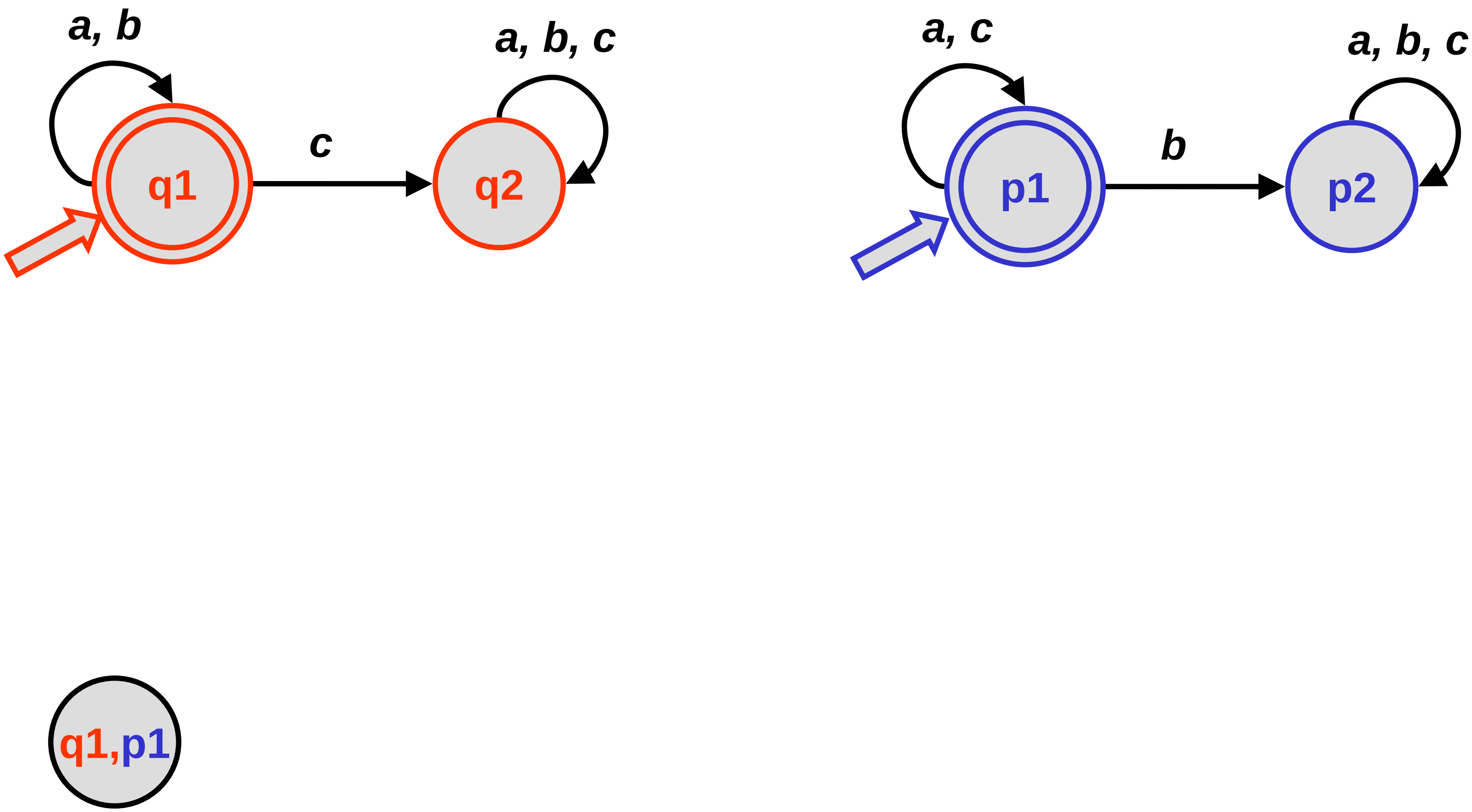
Intersection



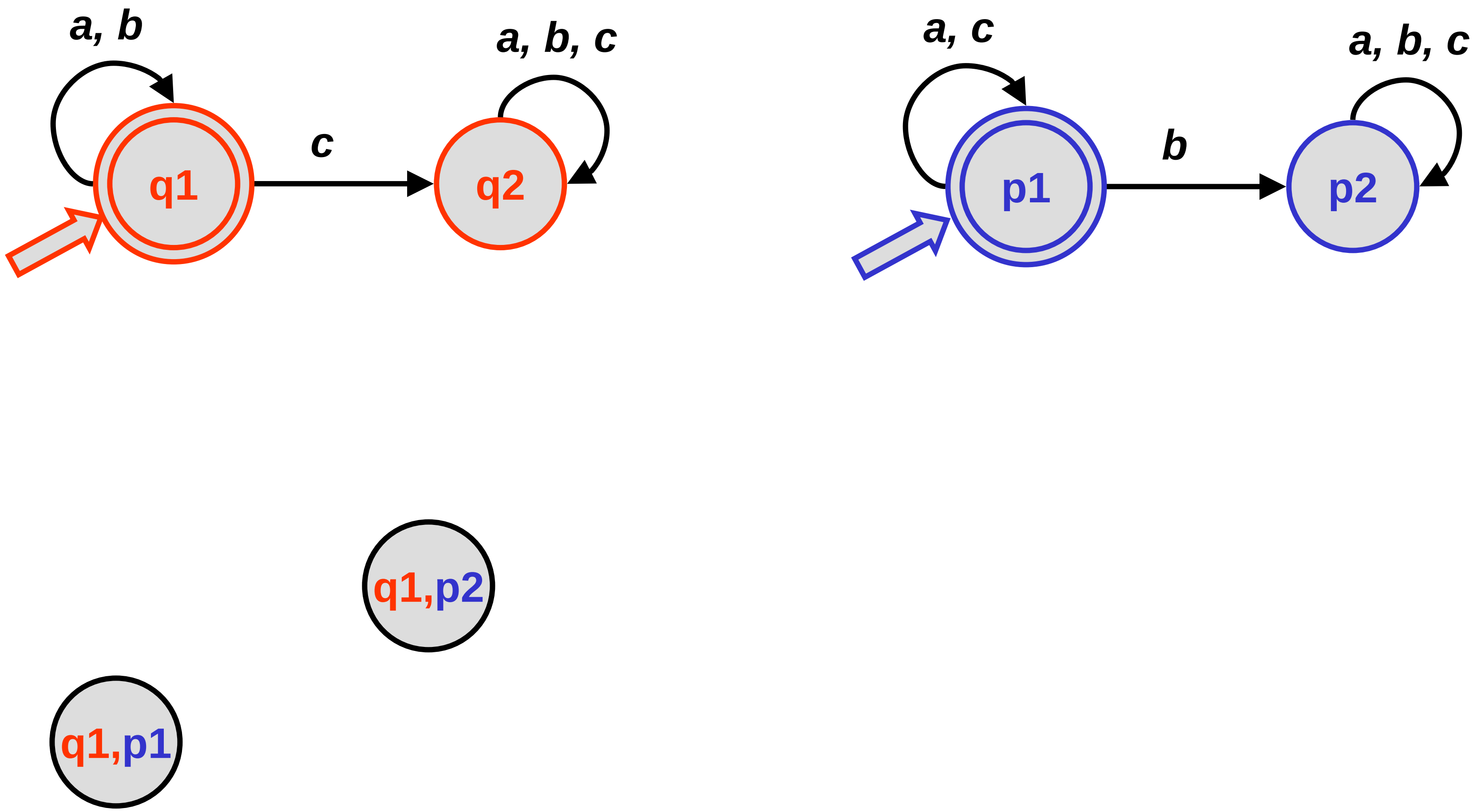
Intersection



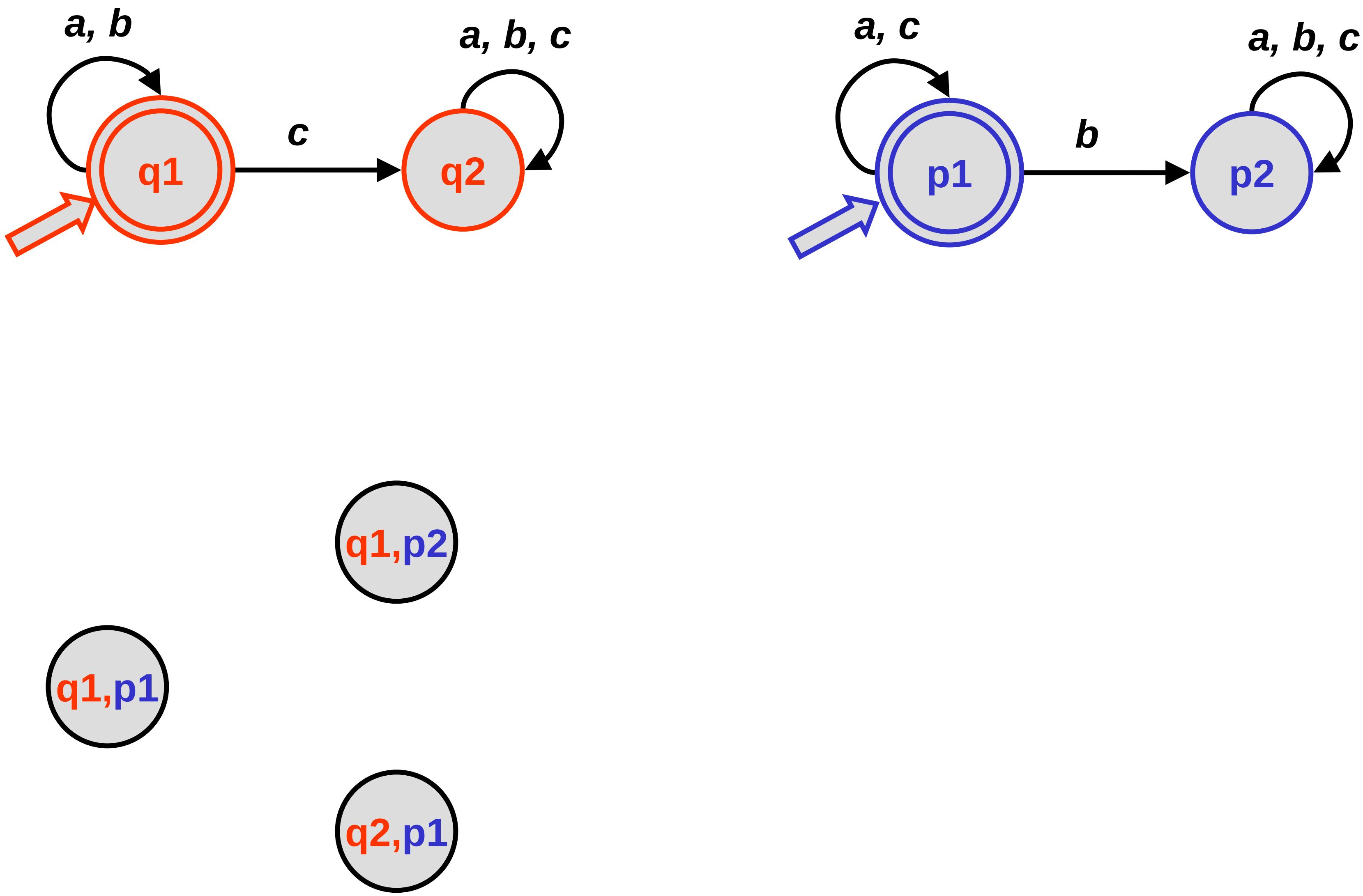
Intersection



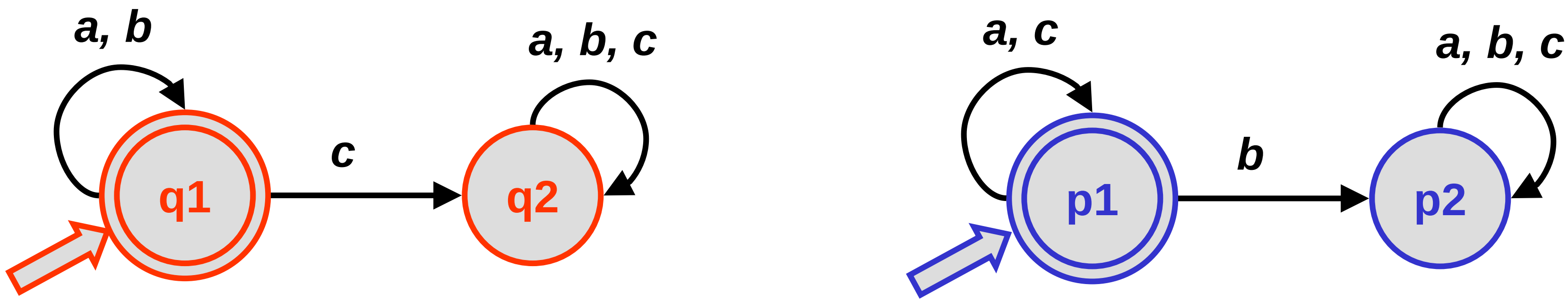
Intersection

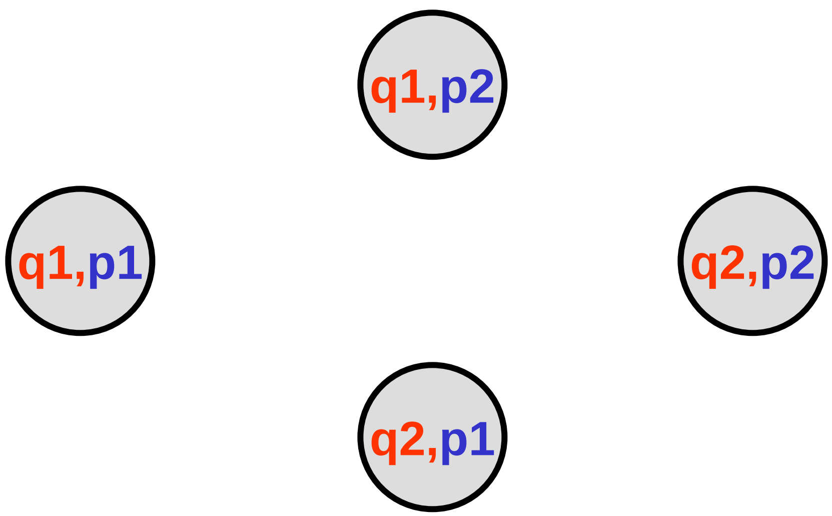


Intersection

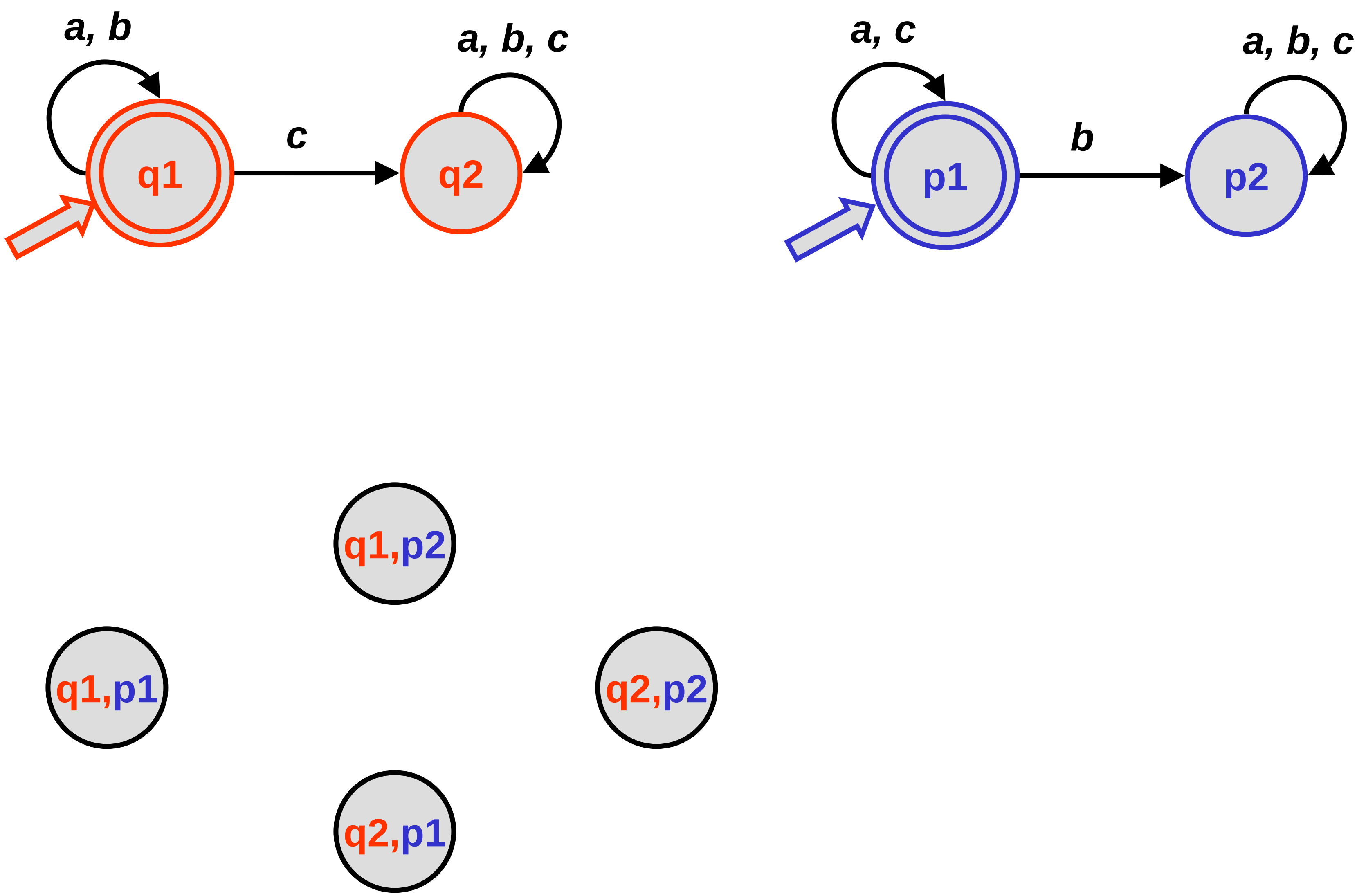


Intersection

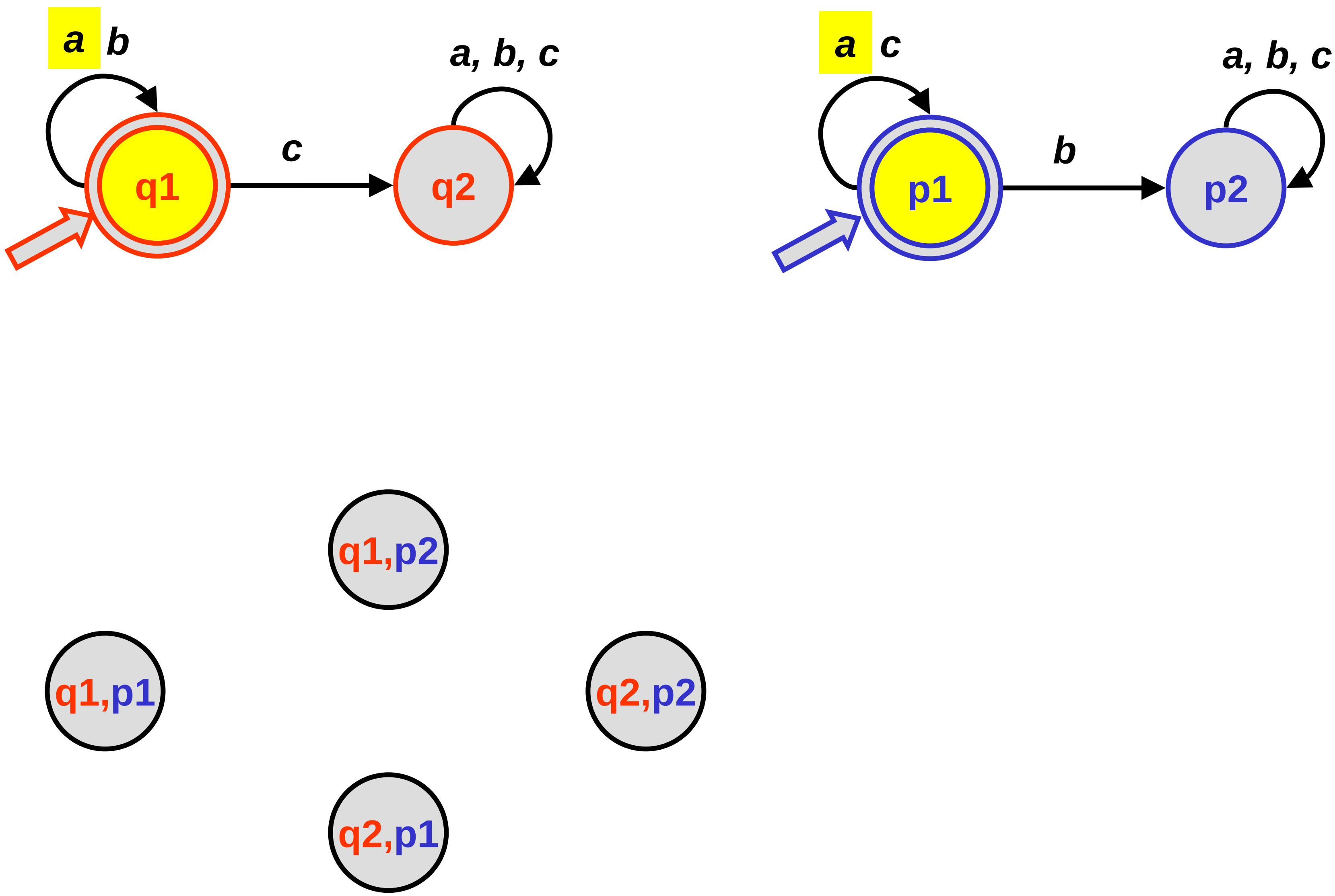




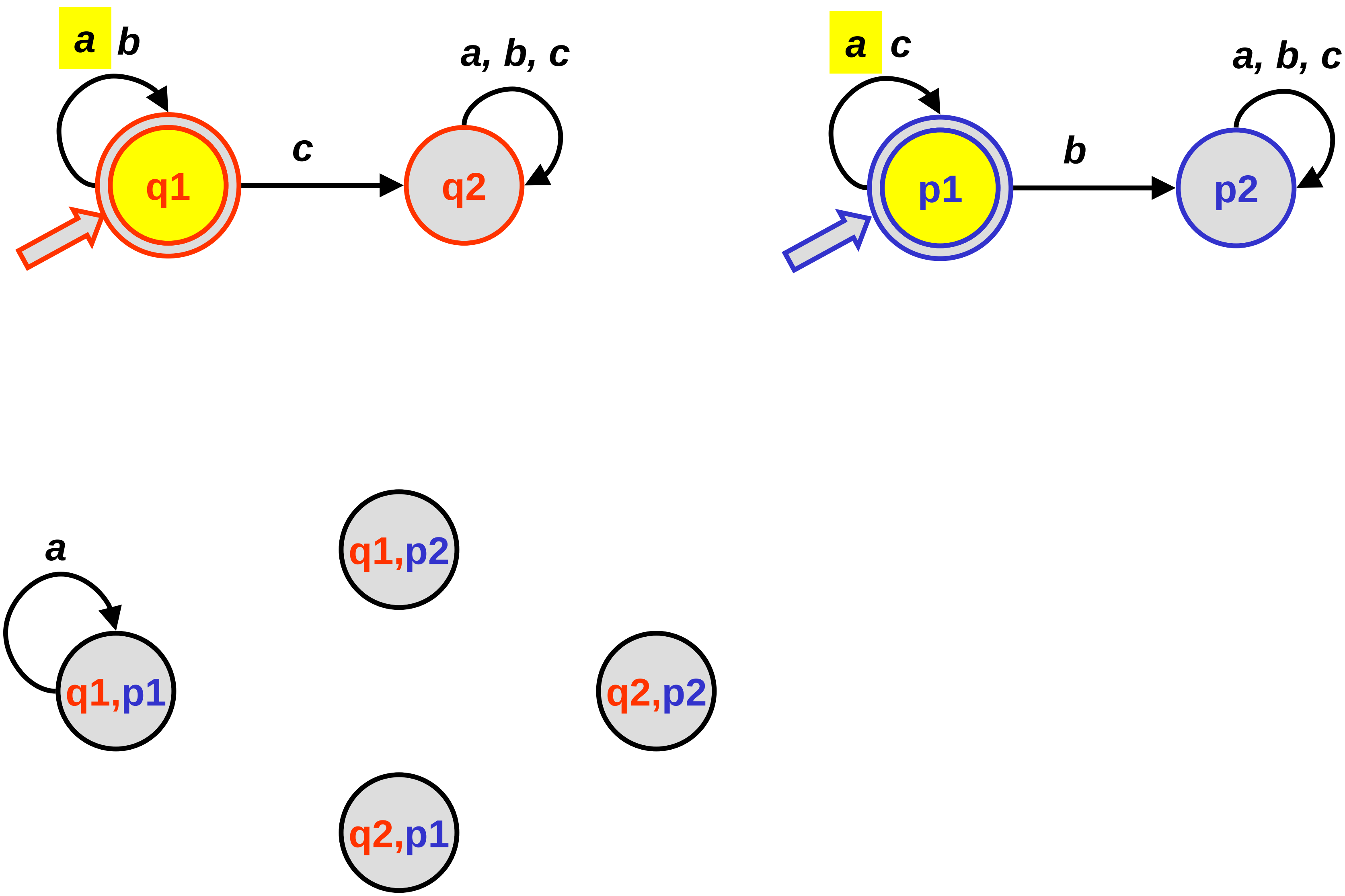
Intersection



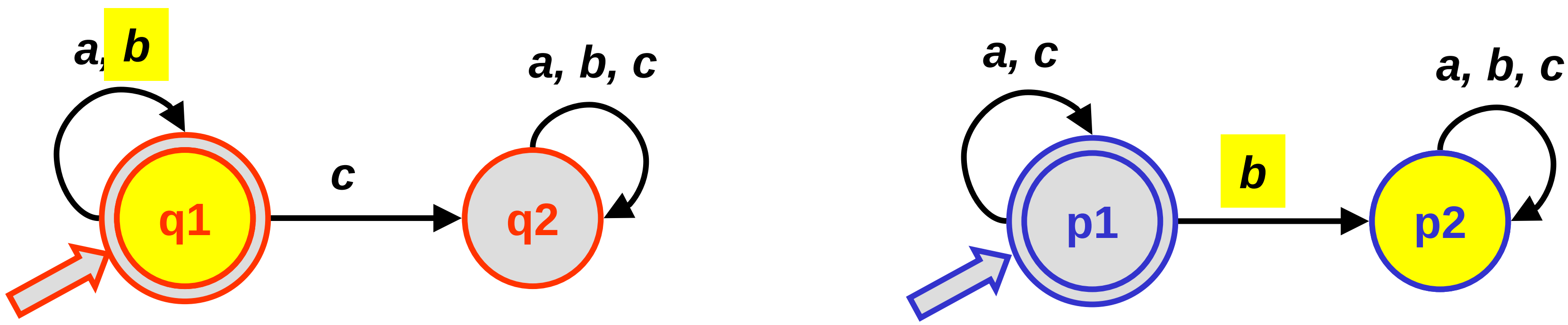
Intersection

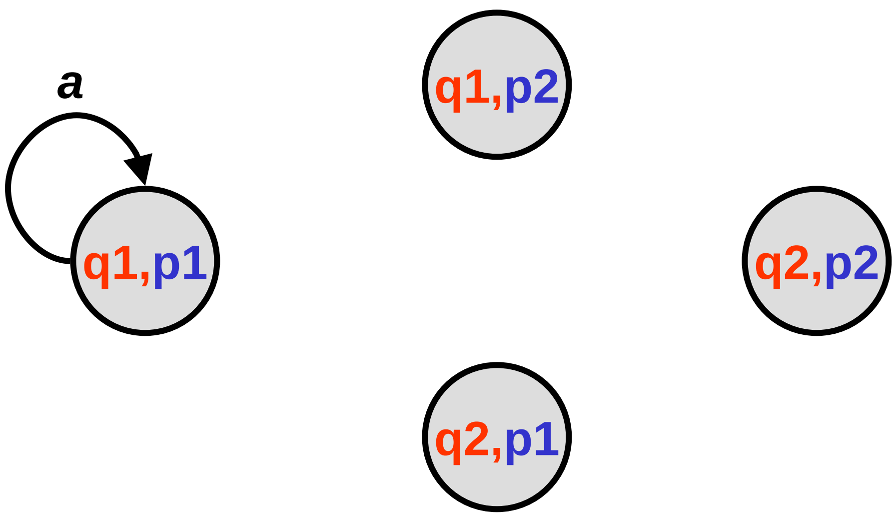


Intersection

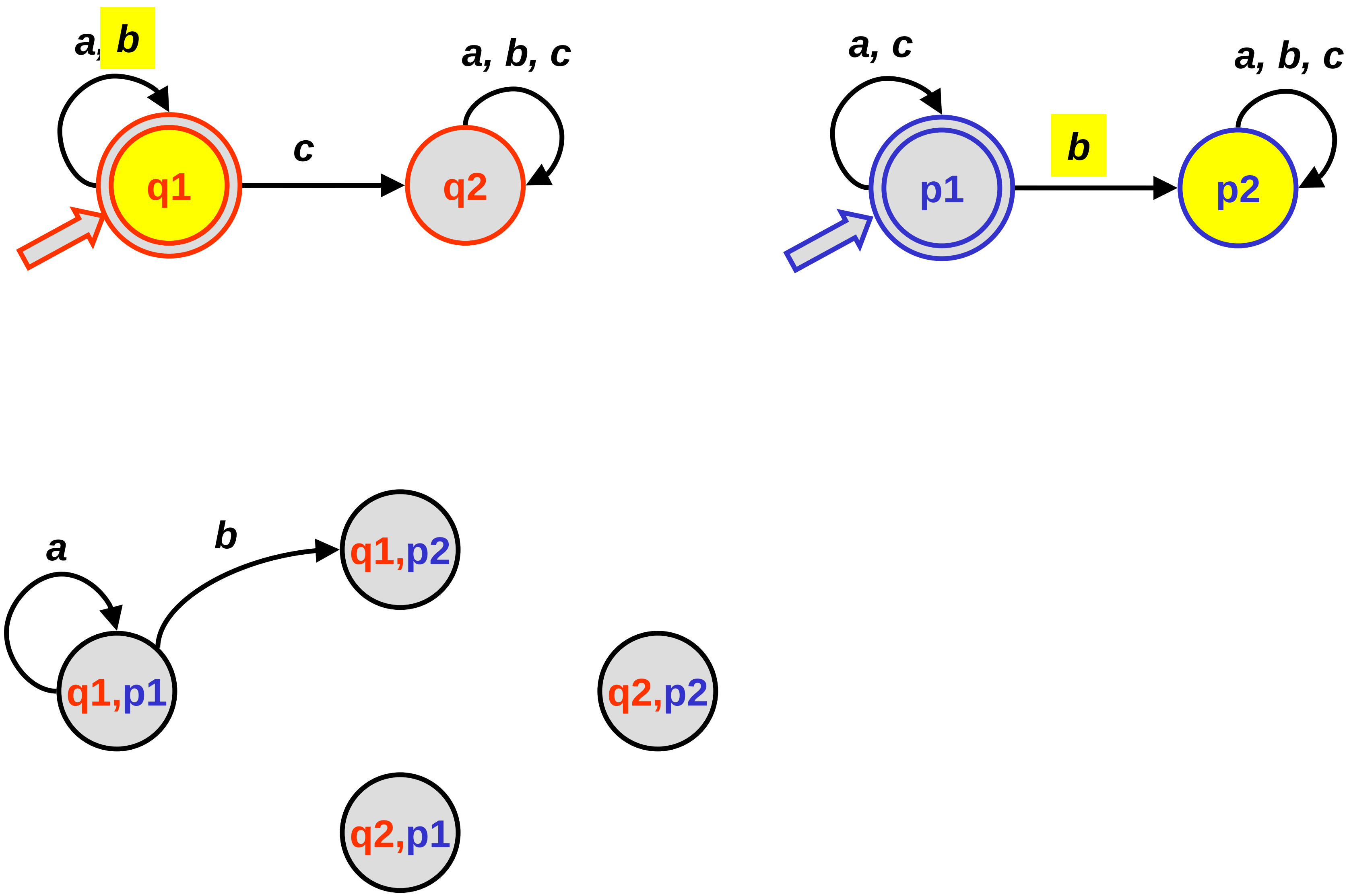


Intersection

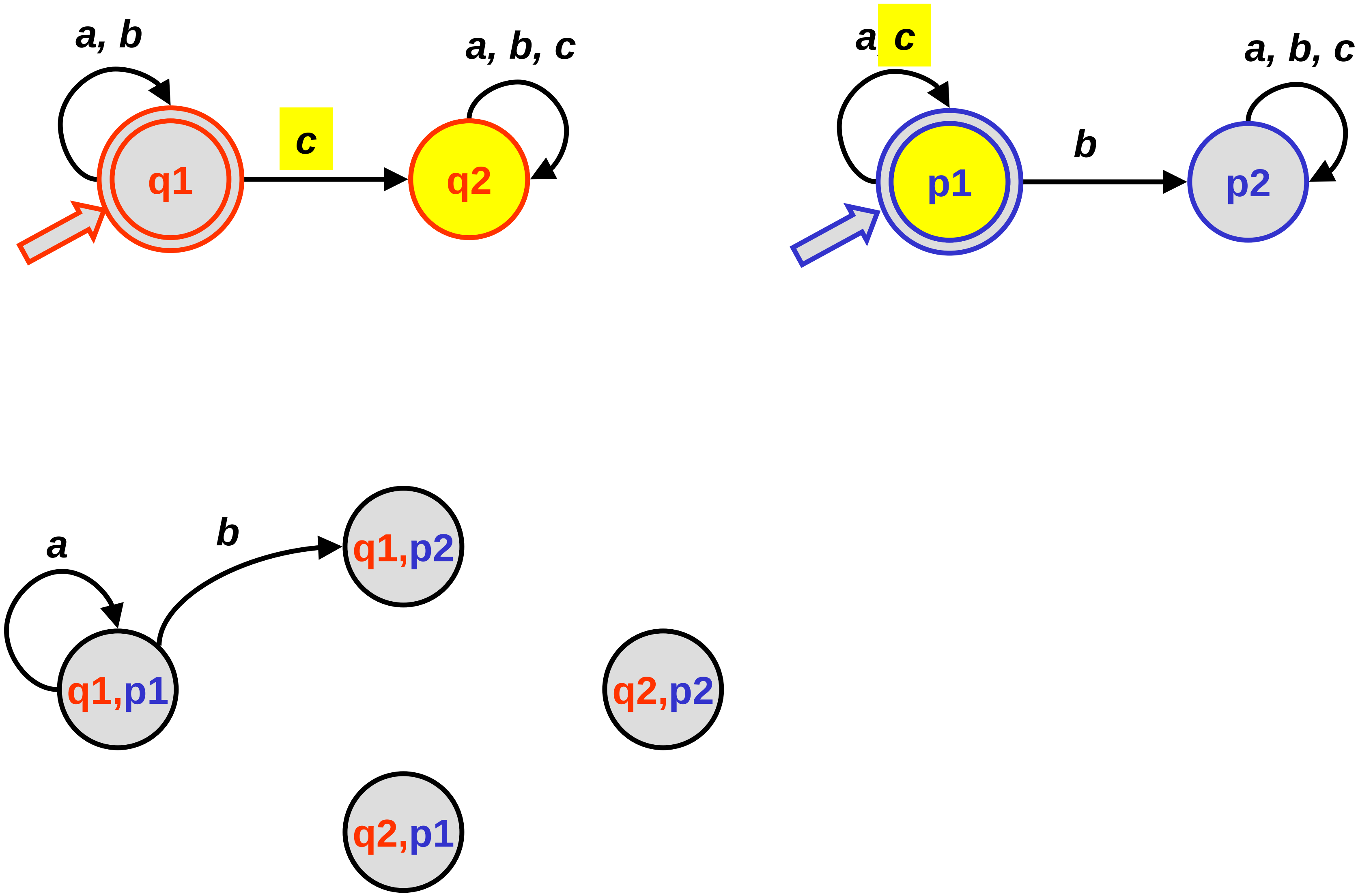




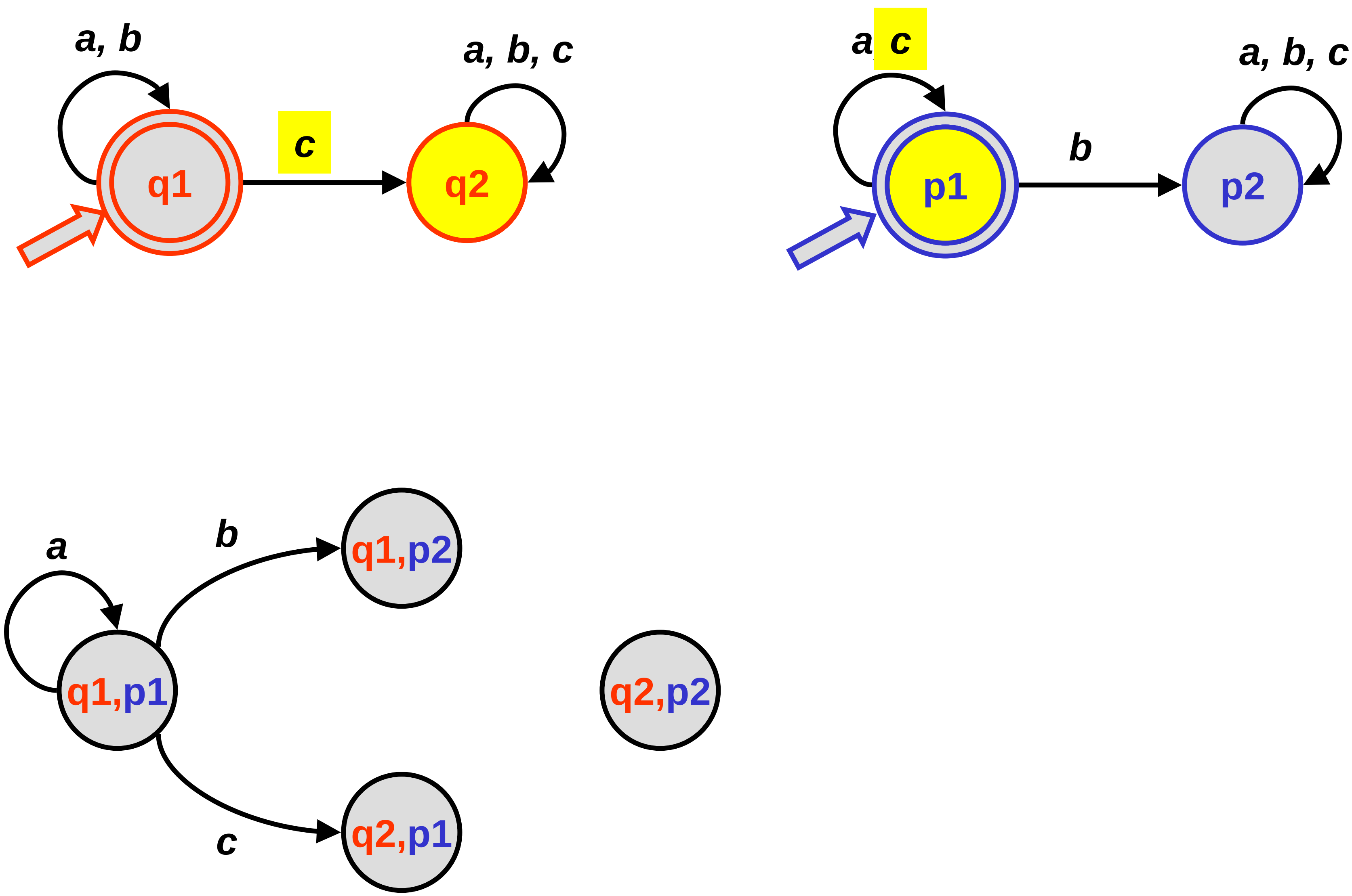
Intersection



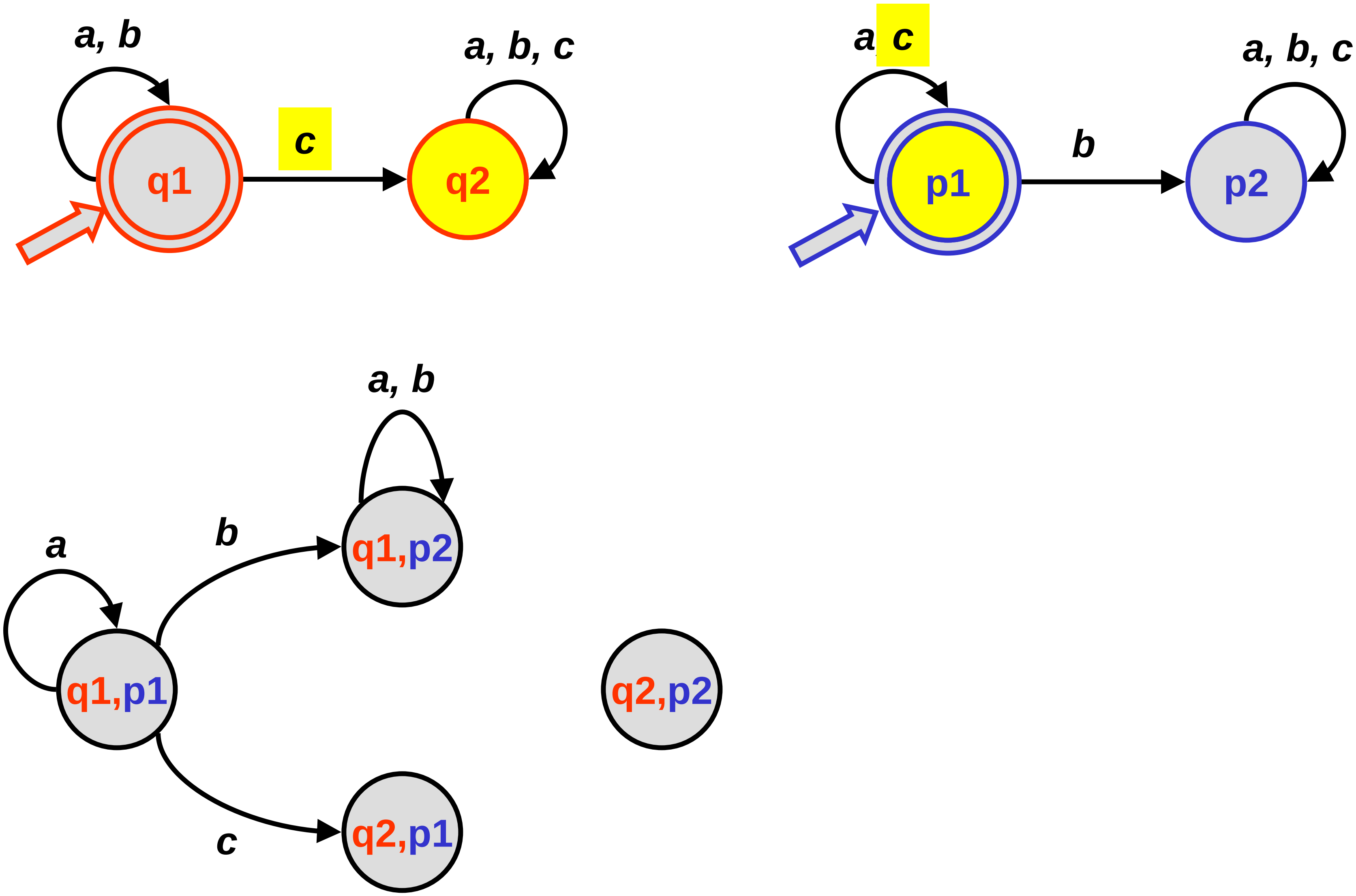
Intersection



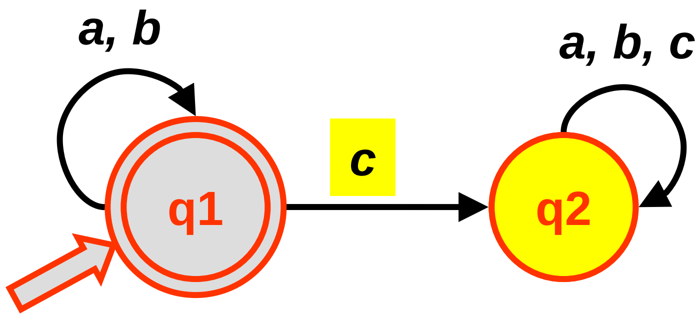
Intersection

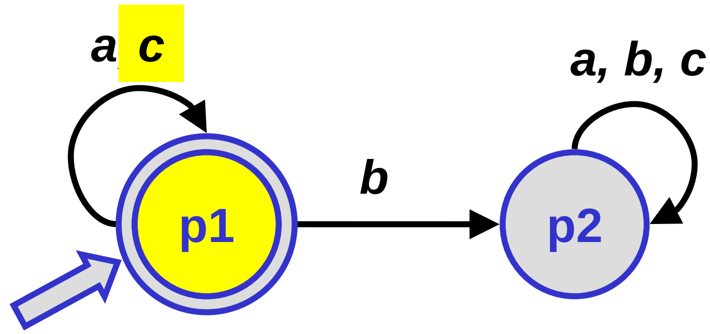


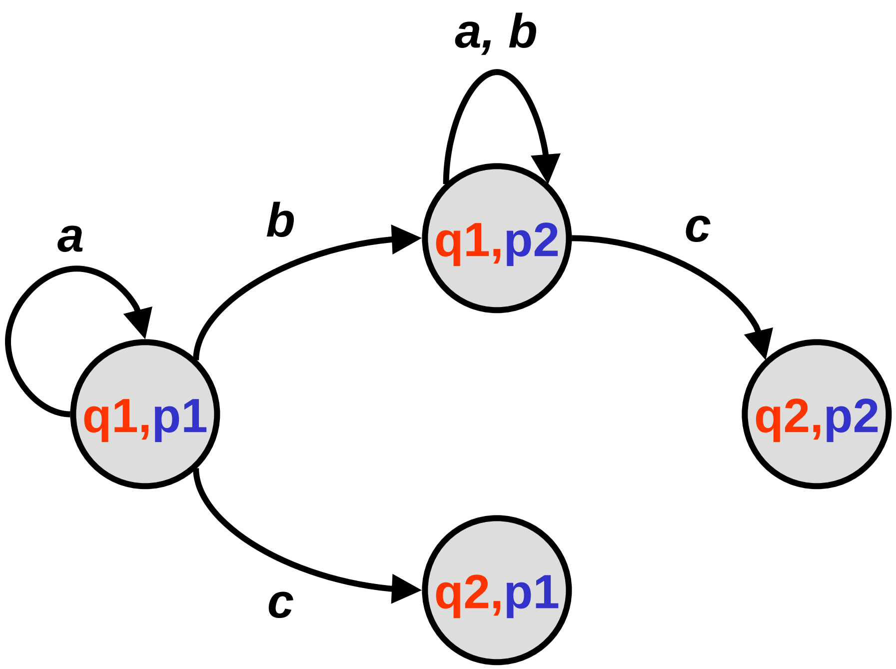
Intersection



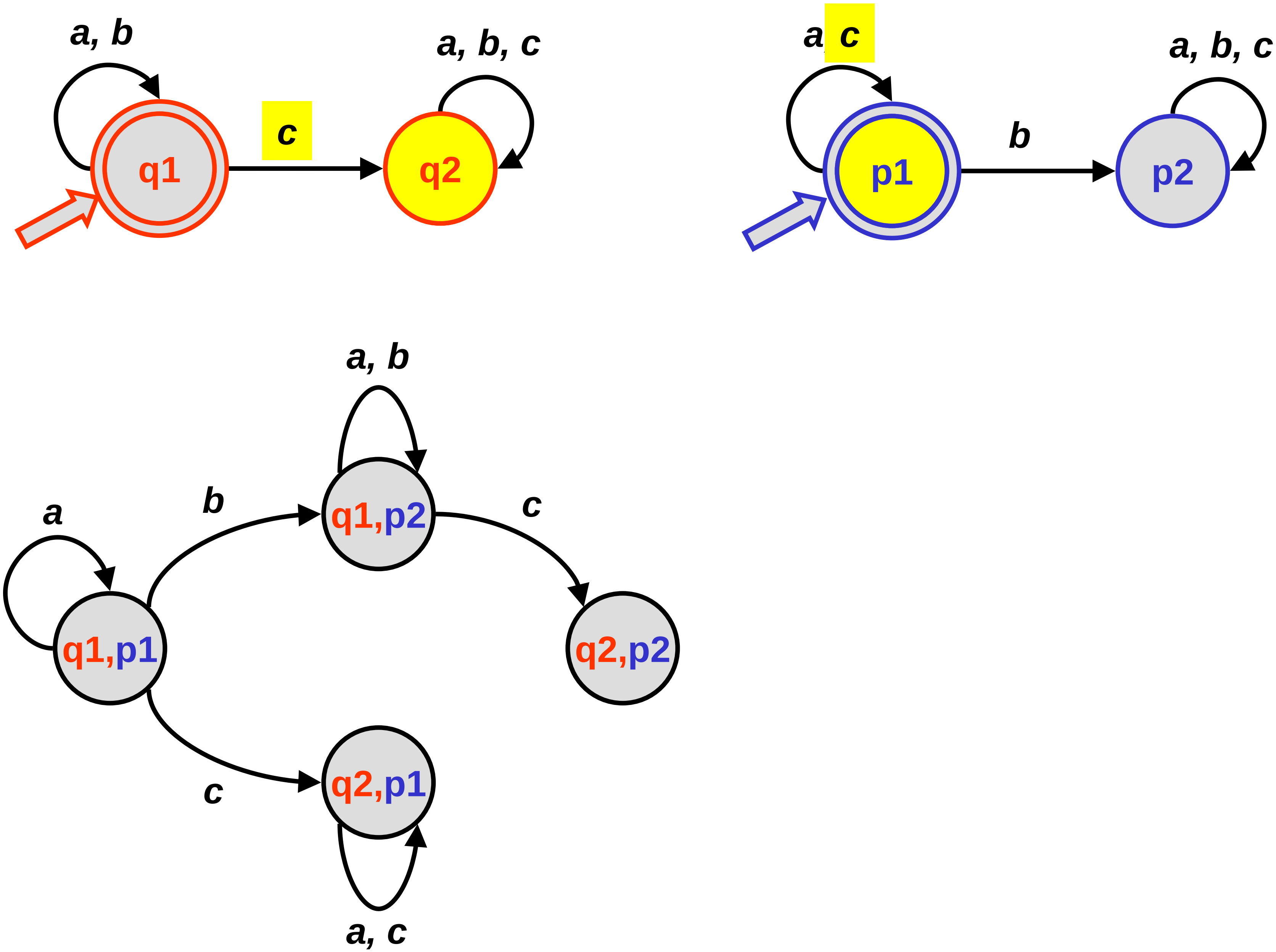
Intersection



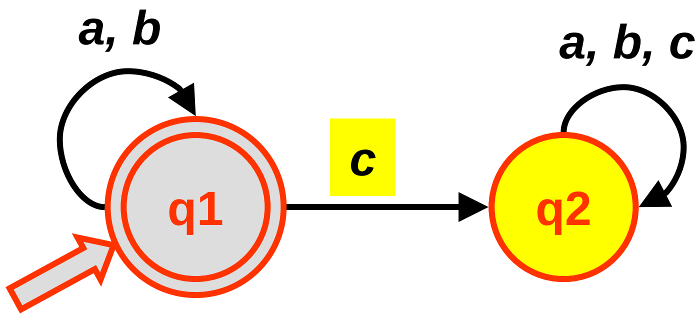


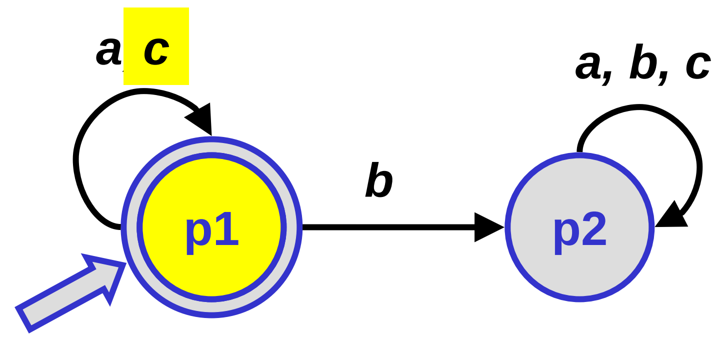


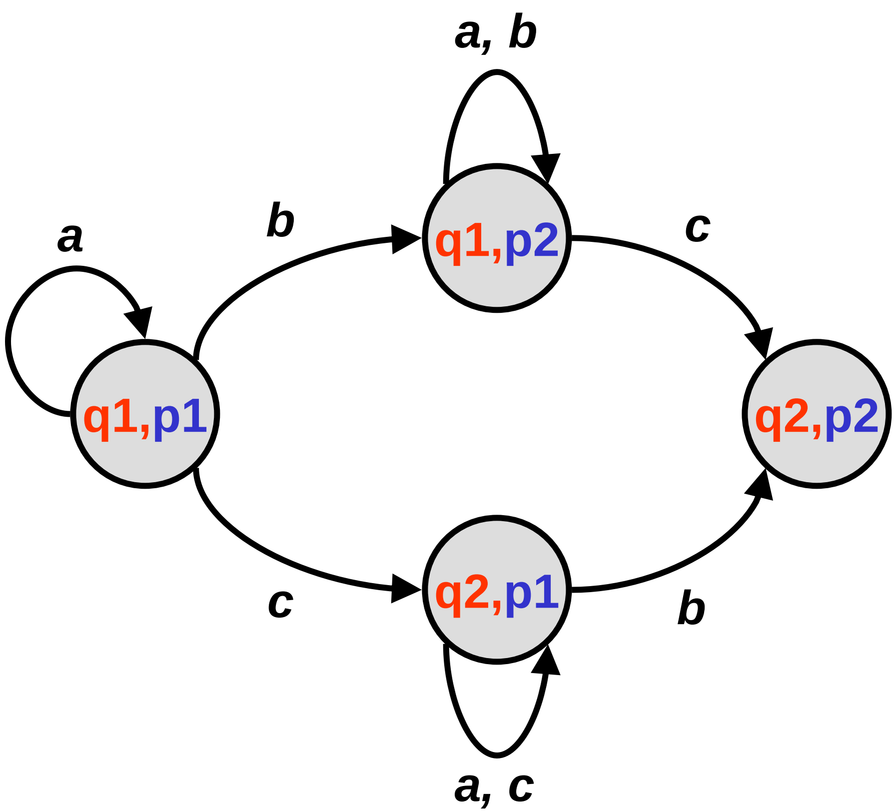
Intersection



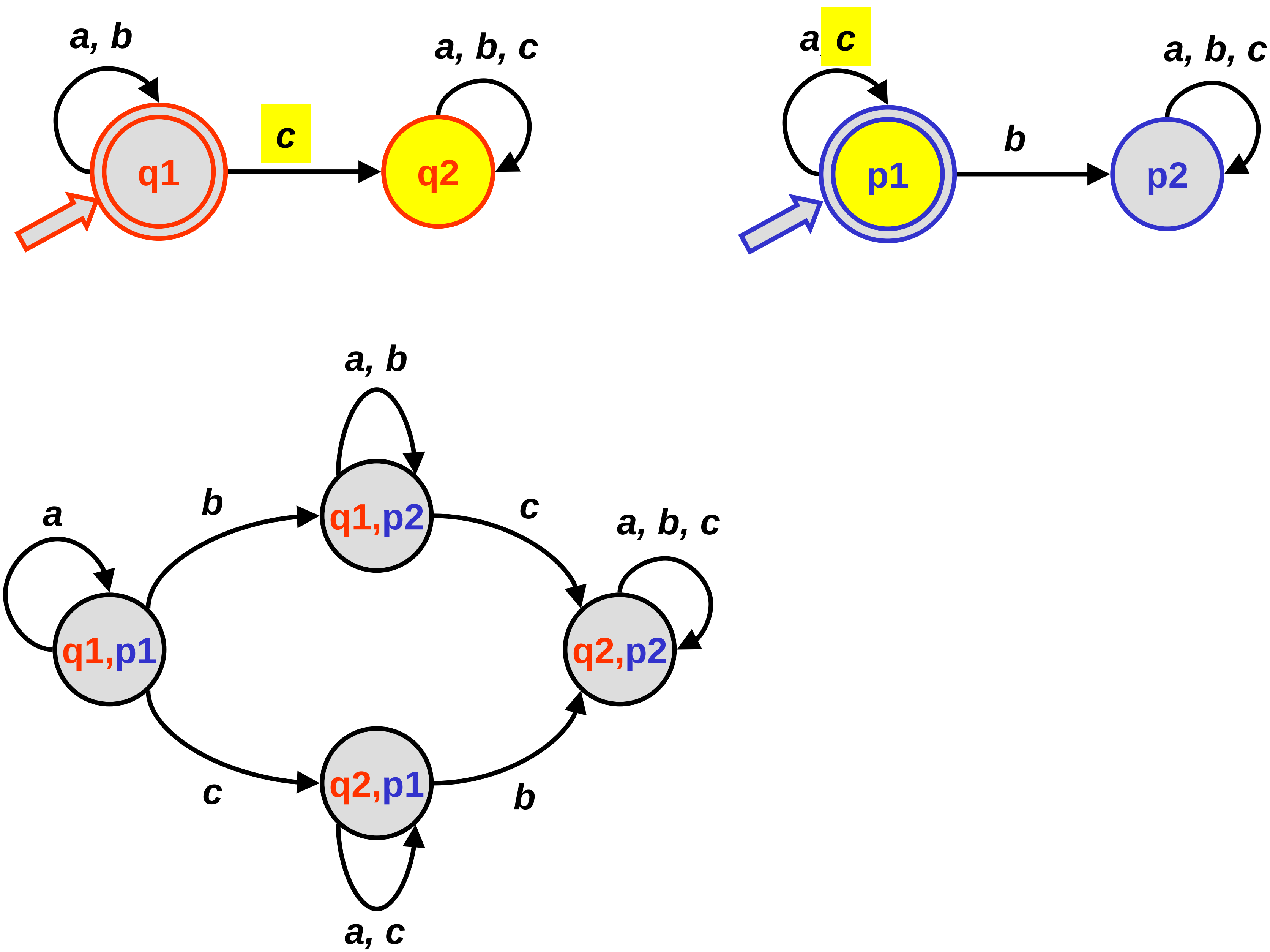
Intersection



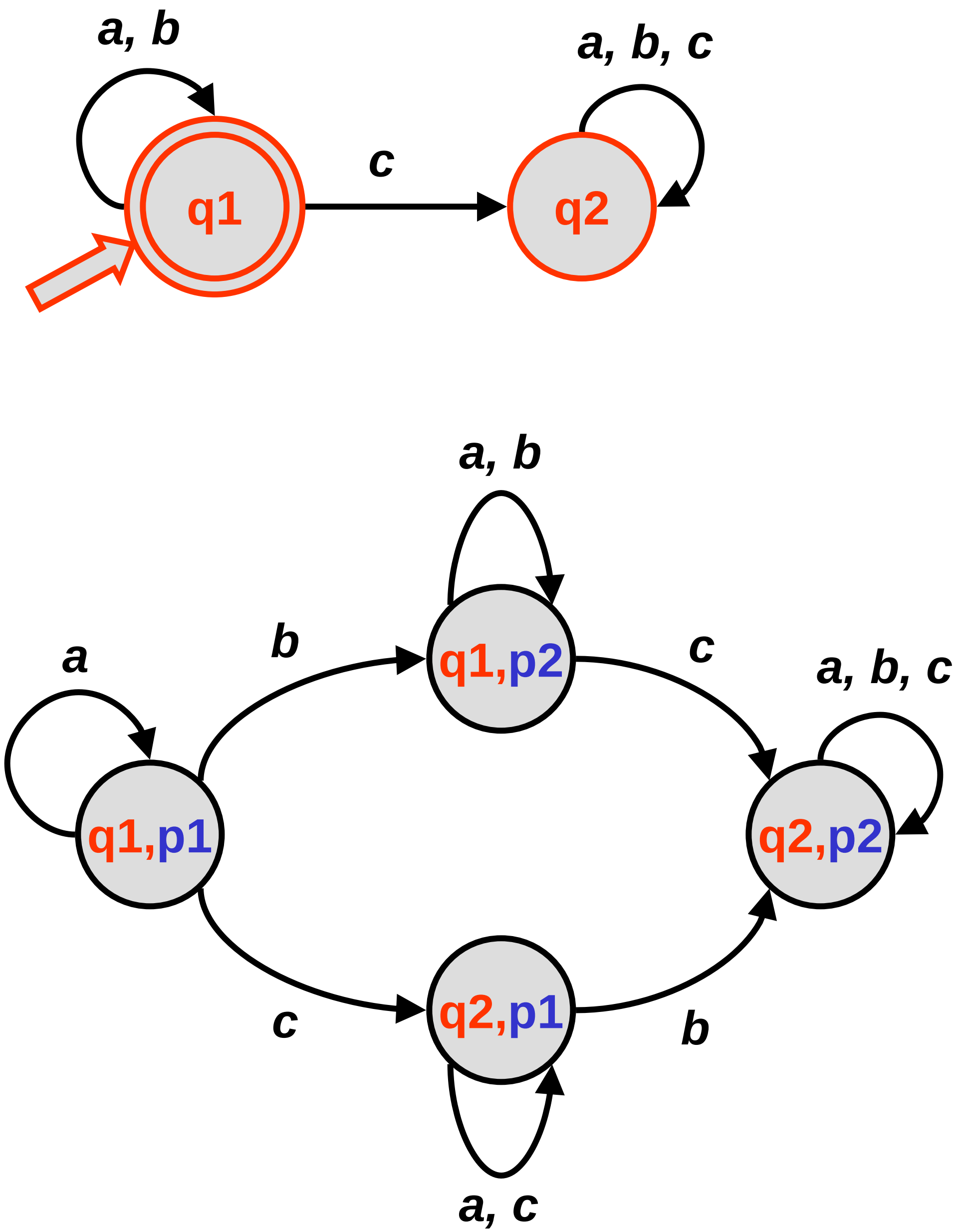




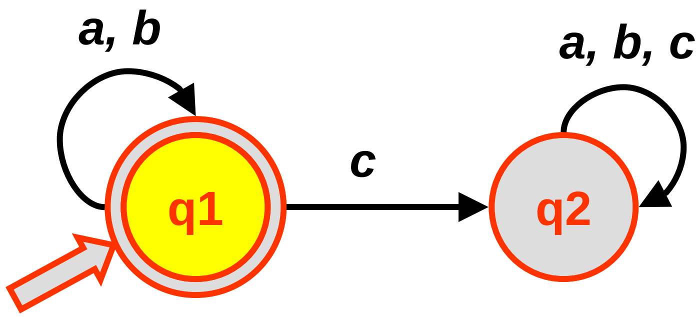
Intersection

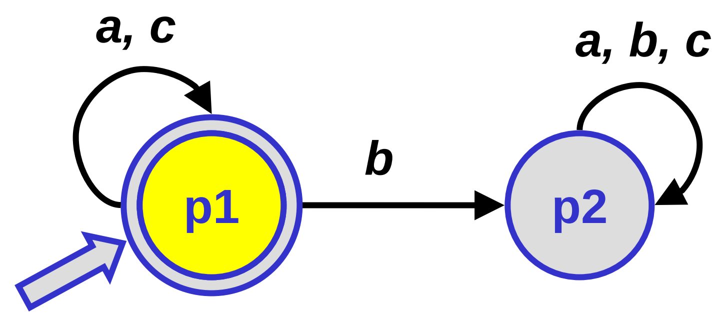


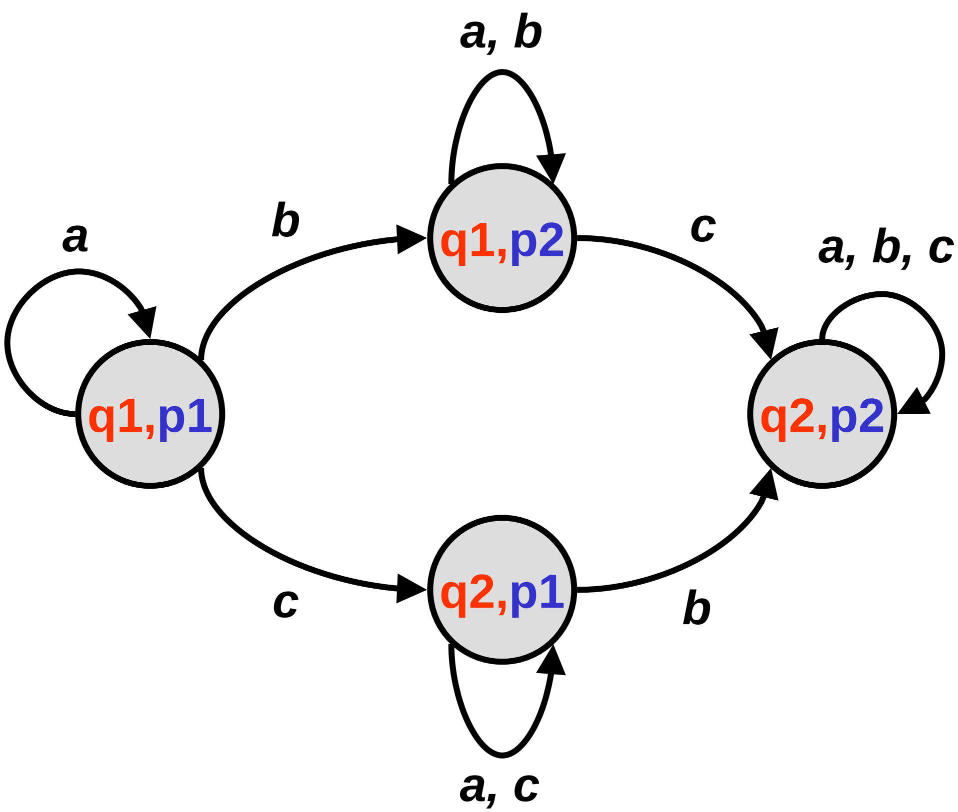
Intersection



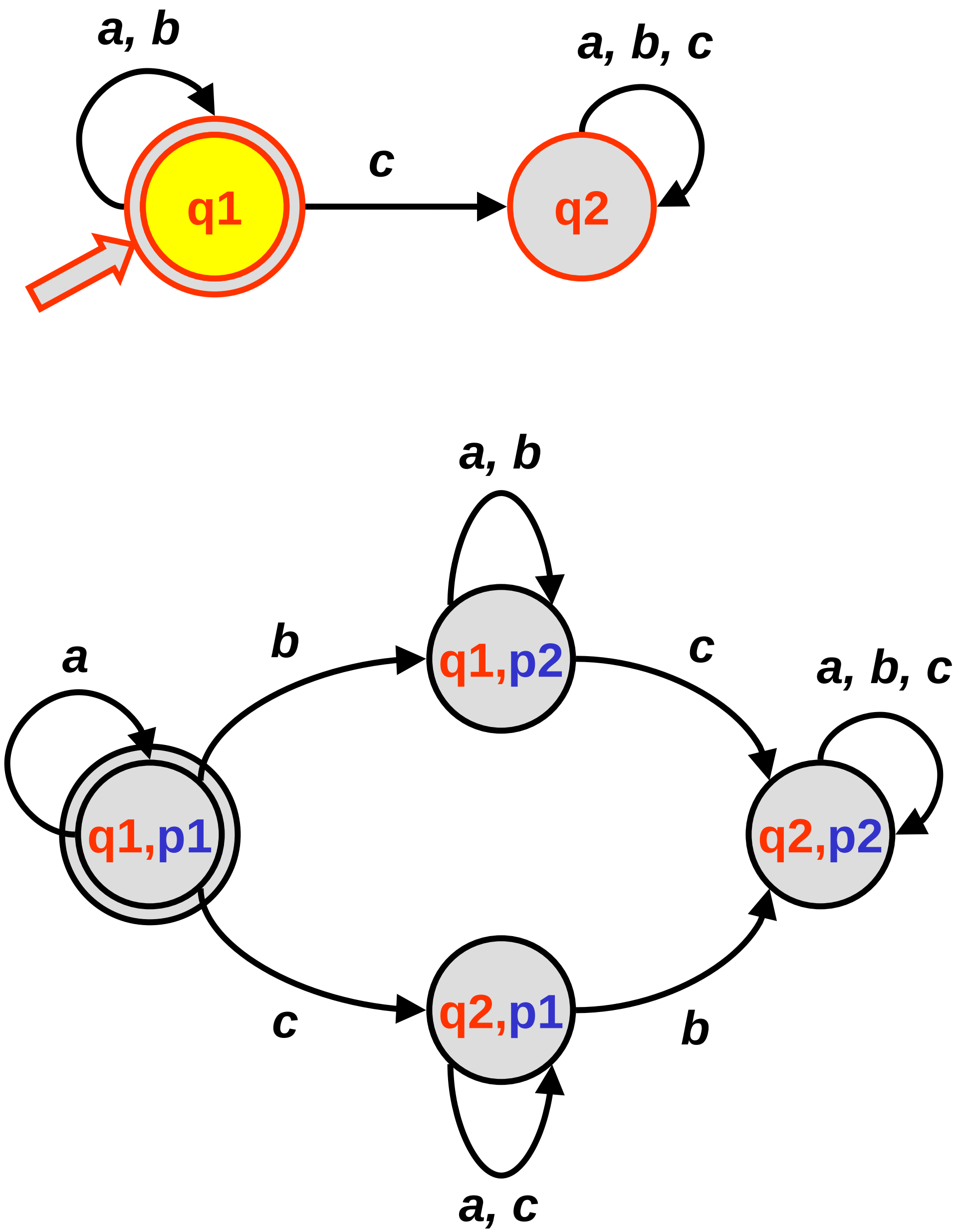
Intersection



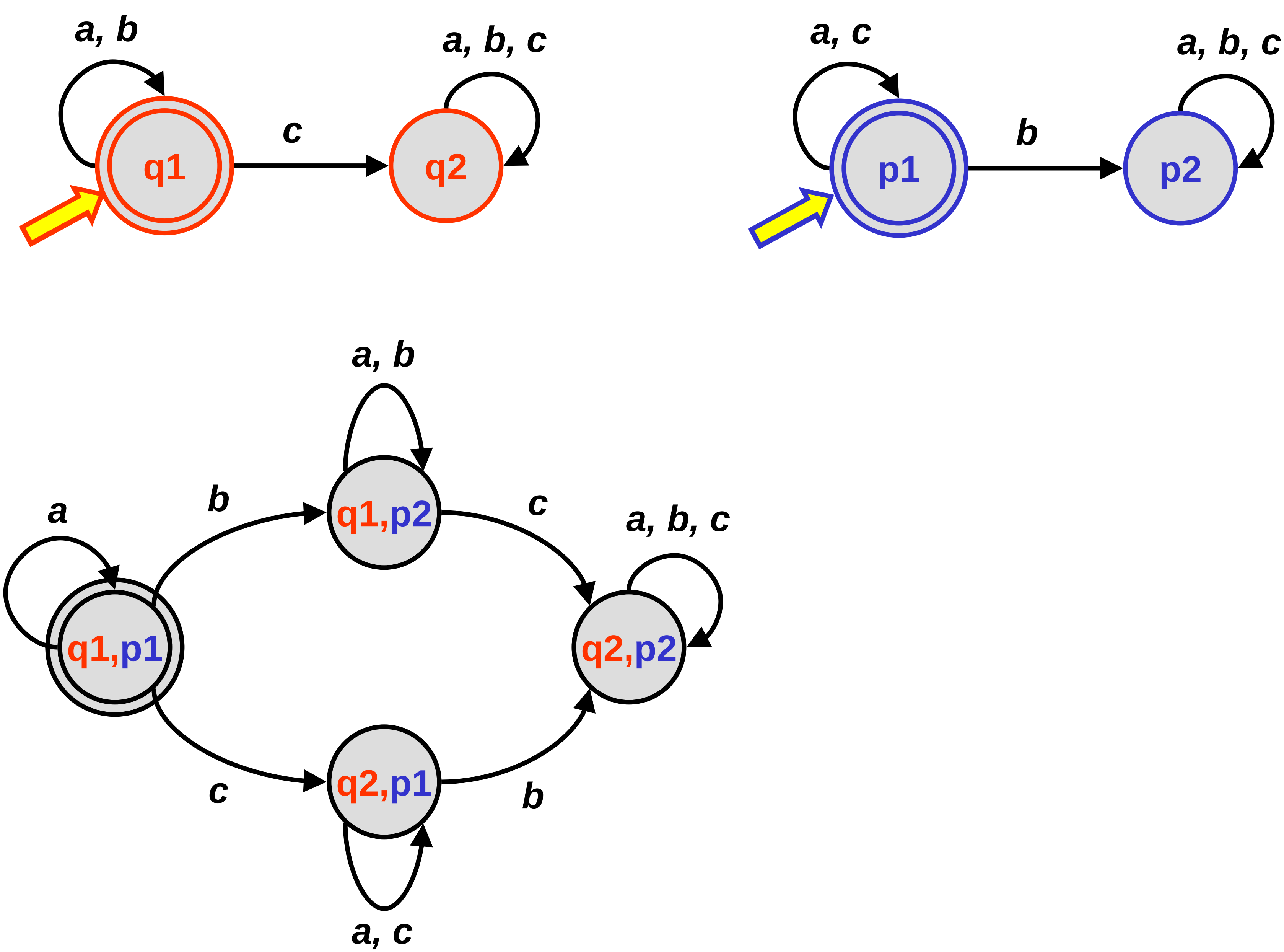




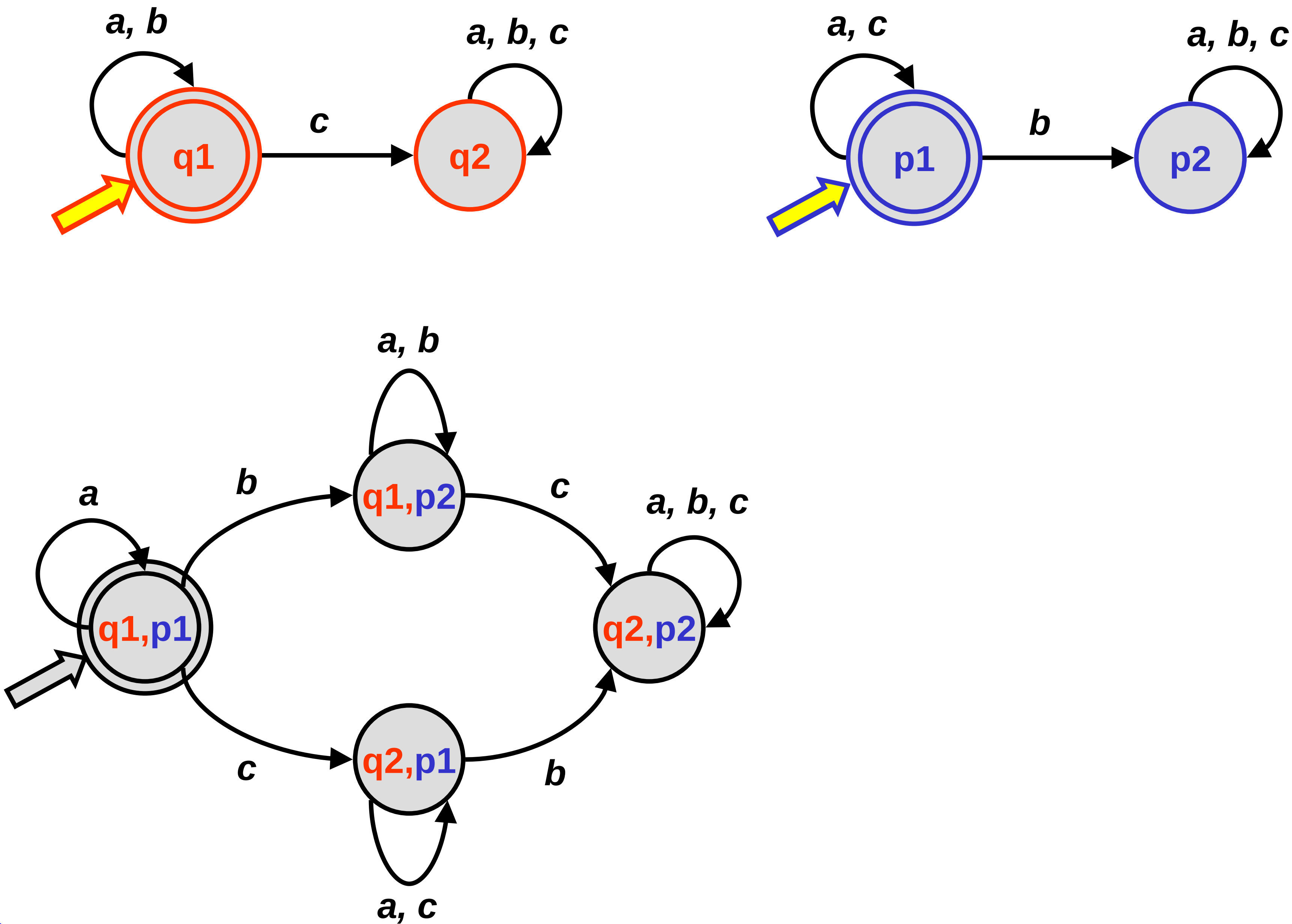
Intersection



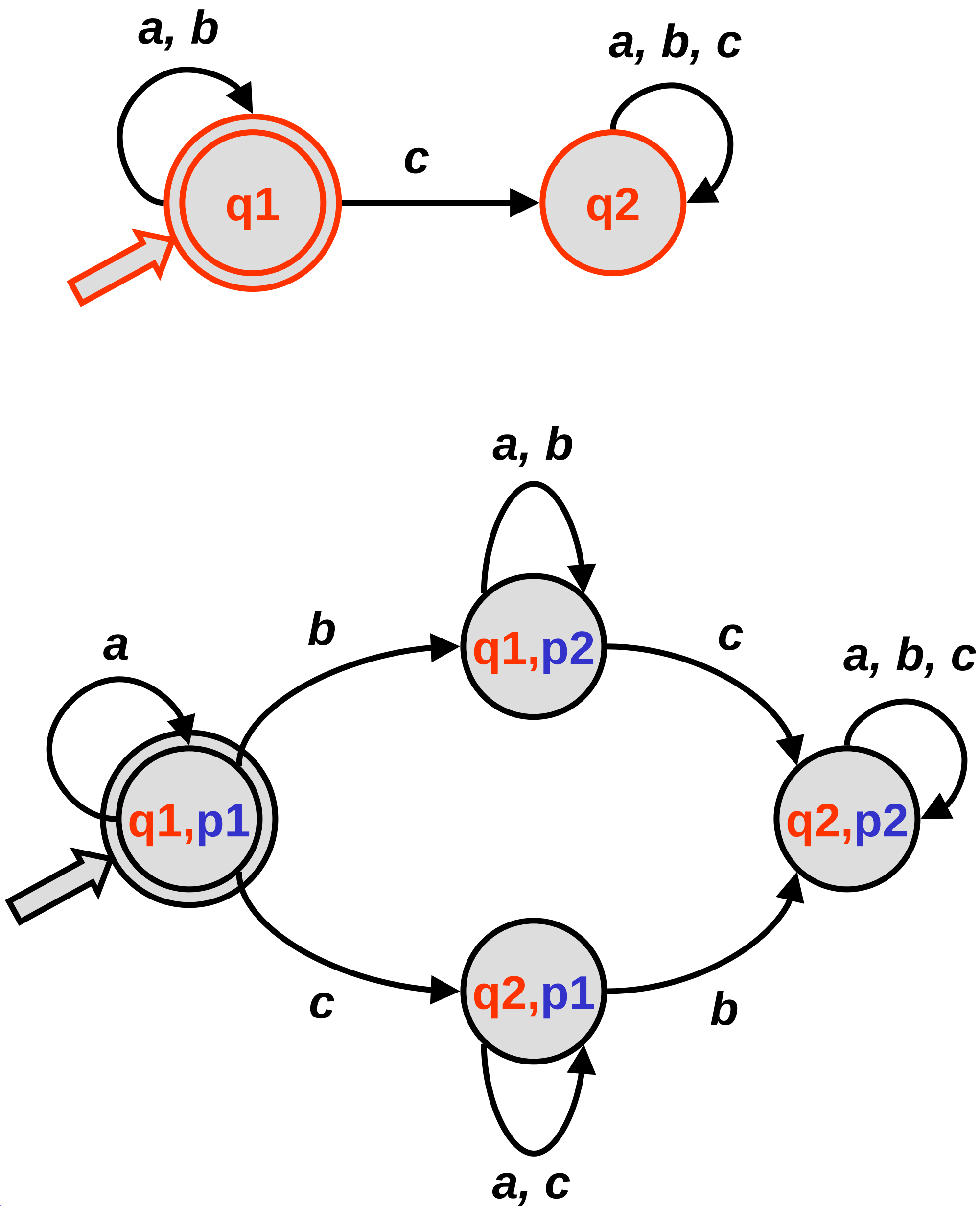
Intersection



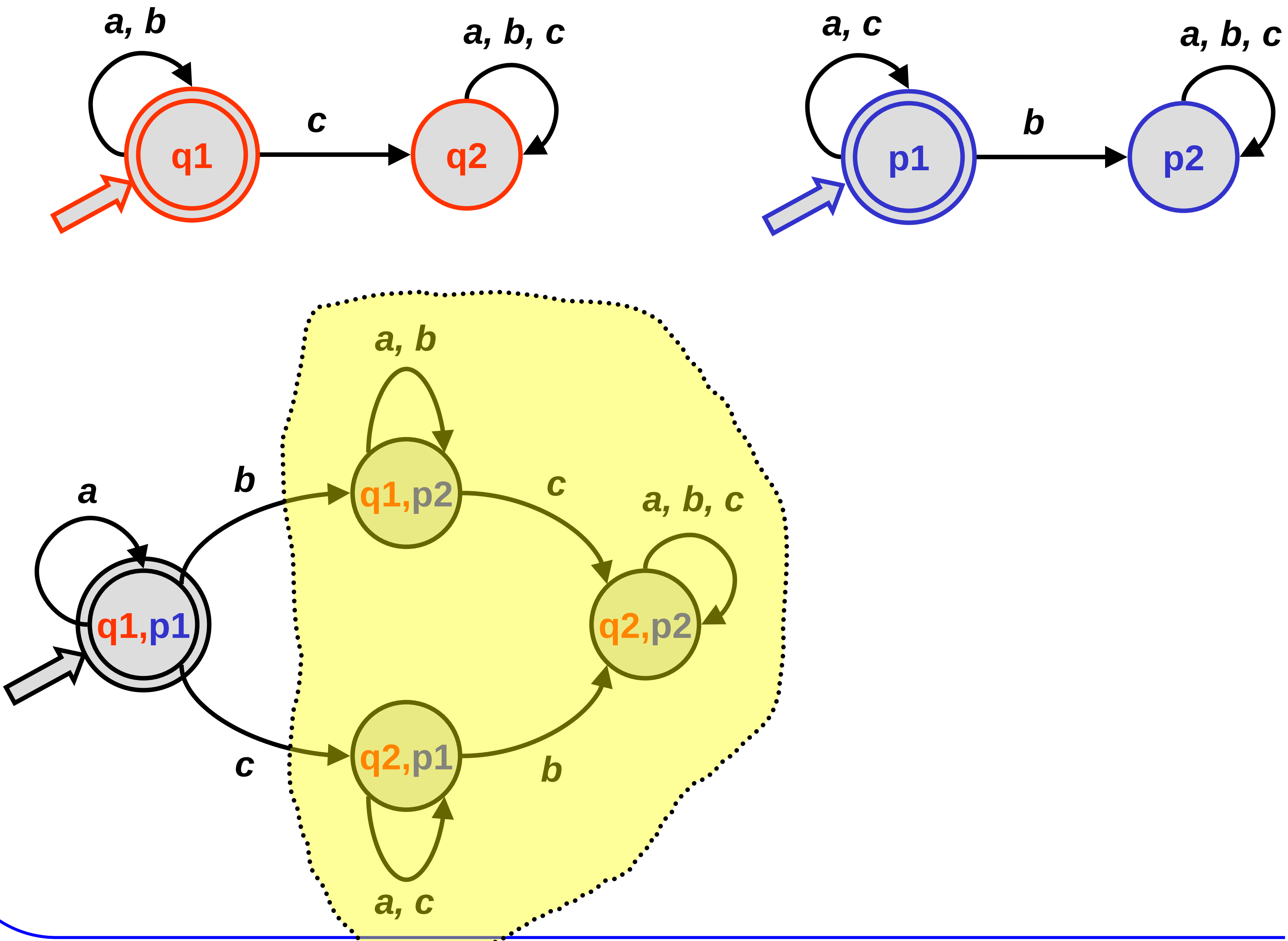
Intersection



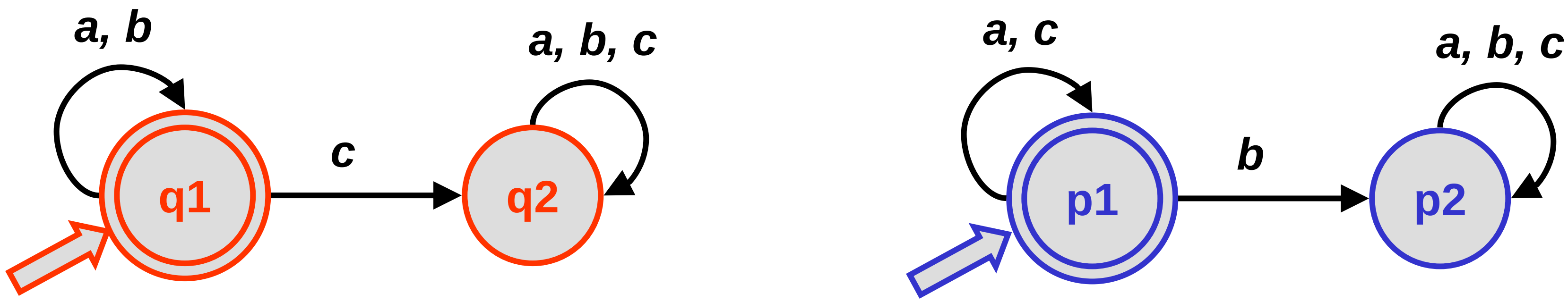
Intersection

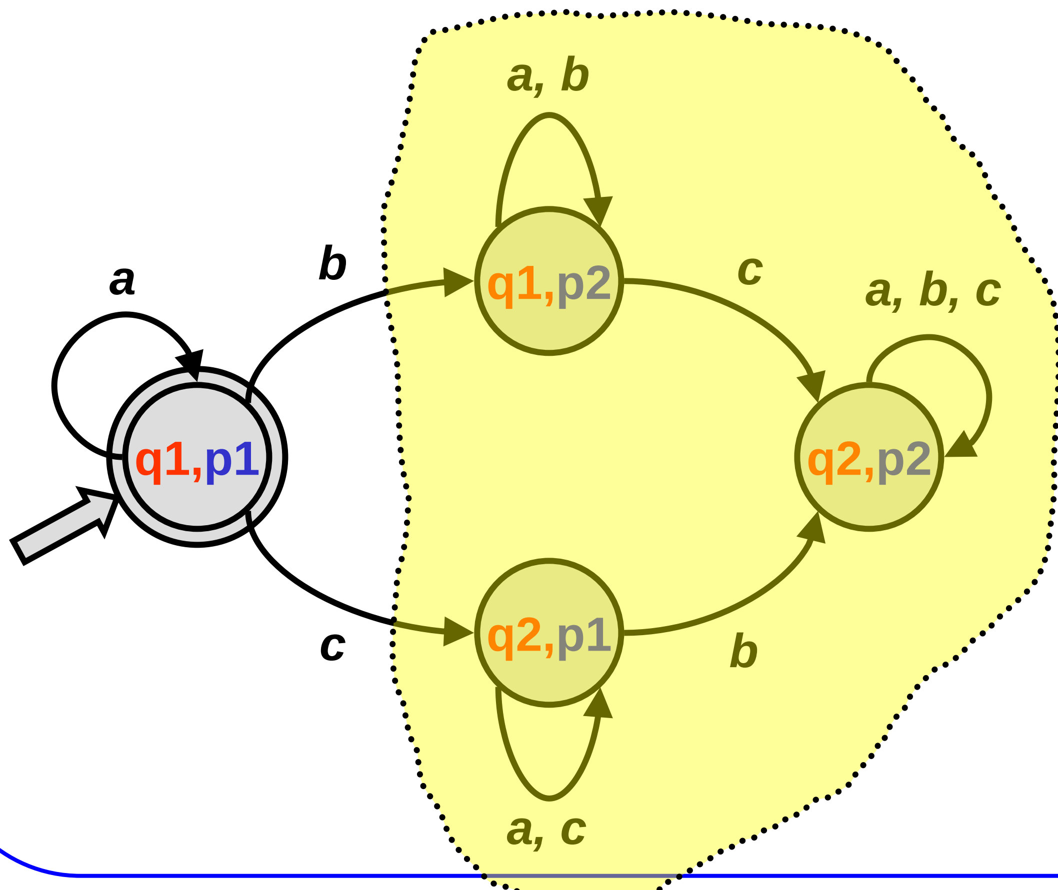


Intersection

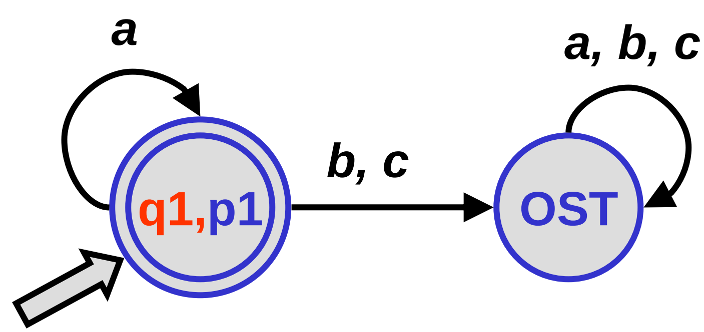


Intersection





Intersection after minimization



Intersection

Intersection

DFA $M_1=(Q_1, \Sigma, \delta_1, q_1, F_1)$

Intersection

DFA $M_1=(Q_1, \Sigma, \delta_1, q_1, F_1)$

DFA $M_2=(Q_2, \Sigma, \delta_2, p_1, F_2)$

Intersection

DFA $M_1=(Q_1, \Sigma, \delta_1, q_1, F_1)$

DFA $M_2=(Q_2, \Sigma, \delta_2, p_1, F_2)$

$$L(M) = L(M_1) \cap L(M_2)$$

Intersection

DFA $M_1=(Q_1, \Sigma, \delta_1, q_1, F_1)$ DFA $M_2=(Q_2, \Sigma, \delta_2, p_1, F_2)$

$$L(M) = L(M_1) \cap L(M_2)$$

1) $Q = Q_1 \times Q_2$

Intersection

DFA $M_1=(Q_1, \Sigma, \delta_1, q_1, F_1)$ DFA $M_2=(Q_2, \Sigma, \delta_2, p_1, F_2)$

$$L(M) = L(M_1) \cap L(M_2)$$

---

1) $Q = Q_1 \times Q_2$

2) $q_0 = [q_1, p_1]$

Intersection

DFA $M_1=(Q_1, \Sigma, \delta_1, q_1, F_1)$

DFA $M_2=(Q_2, \Sigma, \delta_2, p_1, F_2)$

$$L(M) = L(M_1) \cap L(M_2)$$

1) $Q = Q_1 \times Q_2$

2) $q_0 = [q_1, p_1]$

3) $F = F_1 \times F_2$

Intersection

DFA $M_1=(Q_1, \Sigma, \delta_1, q_1, F_1)$ DFA $M_2=(Q_2, \Sigma, \delta_2, p_1, F_2)$

$$L(M) = L(M_1) \cap L(M_2)$$

1) $Q = Q_1 \times Q_2$

2) $q_0 = [q_1, p_1]$

3) $F = F_1 \times F_2$

4) $\delta([q,p], a) = [\ \delta_1(q, a), \delta_2(p,a)\ ]$

Intersection

Intersection

	a	*b*	*c*	
q1	***q1***	***q1***	***q2***	**1**
q2	***q2***	***q2***	***q2***	**0**

Intersection

	a	b	c	
q1	q1	q1	q2	1
q2	q2	q2	q2	0

	a	b	c	
p1	p1	p2	p1	1
p2	p2	p2	p2	0

Intersection

	a	*b*	*c*	
*q*1	*q*1	*q*1	*q*2	1
*q*2	*q*2	*q*2	*q*2	0

	a	*b*	*c*	
*p*1	*p*1	*p*2	*p*1	1
*p*2	*p*2	*p*2	*p*2	0

[*q*1, *p*1]

[*q*1, *p*2]

[*q*2, *p*1]

[*q*2, *p*2]

Intersection

	a	*b*	*c*	
*q*1	*q*1	*q*1	*q*2	1
*q*2	*q*2	*q*2	*q*2	0

	a	*b*	*c*	
*p*1	*p*1	*p*2	*p*1	1
*p*2	*p*2	*p*2	*p*2	0

	a	*b*	*c*
[*q*1, *p*1]			
[*q*1, *p*2]			
[*q*2, *p*1]			
[*q*2, *p*2]			

Intersection

	a	*b*	*c*	
*q*1	*q*1	*q*1	*q*2	1
*q*2	*q*2	*q*2	*q*2	0

	a	*b*	*c*	
*p*1	*p*1	*p*2	*p*1	1
*p*2	*p*2	*p*2	*p*2	0

	a	*b*	*c*
[*q*1, *p*1]			
[*q*1, *p*2]			
[*q*2, *p*1]			
[*q*2, *p*2]			

Intersection

	a	*b*	*c*	
*q*1	*q*1	*q*1	*q*2	1
*q*2	*q*2	*q*2	*q*2	0

	a	*b*	*c*	
*p*1	*p*1	*p*2	*p*1	1
*p*2	*p*2	*p*2	*p*2	0

	a	*b*	*c*	
[*q*1, *p*1]				1
[*q*1, *p*2]				
[*q*2, *p*1]				
[*q*2, *p*2]				

Intersection

	a	b	c	
*q*1	*q*1	*q*1	*q*2	1
*q*2	*q*2	*q*2	*q*2	0

	a	b	c	
*p*1	*p*1	*p*2	*p*1	1
*p*2	*p*2	*p*2	*p*2	0

	a	b	c	
[*q*1, *p*1]				1
[*q*1, *p*2]				0
[*q*2, *p*1]				0
[*q*2, *p*2]				0

Intersection

	a	*b*	*c*	
*q*1	*q*1	*q*1	*q*2	1
*q*2	*q*2	*q*2	*q*2	0

	a	*b*	*c*	
*p*1	*p*1	*p*2	*p*1	1
*p*2	*p*2	*p*2	*p*2	0

	a	*b*	*c*	
[*q*1, *p*1]				1
[*q*1, *p*2]				0
[*q*2, *p*1]				0
[*q*2, *p*2]				0

Intersection

	a	*b*	*c*	
*q*1	*q*1	*q*1	*q*2	1
*q*2	*q*2	*q*2	*q*2	0

	a	*b*	*c*	
*p*1	*p*1	*p*2	*p*1	1
*p*2	*p*2	*p*2	*p*2	0

	a	*b*	*c*	
[*q*1, *p*1]	[*q*1, *p*1]			1
[*q*1, *p*2]				0
[*q*2, *p*1]				0
[*q*2, *p*2]				0

Intersection

	a	*b*	*c*	
*q*1	*q*1	*q*1	*q*2	1
*q*2	*q*2	*q*2	*q*2	0

	a	*b*	*c*	
*p*1	*p*1	*p*2	*p*1	1
*p*2	*p*2	*p*2	*p*2	0

	a	*b*	*c*	
[*q*1, *p*1]	[*q*1, *p*1]	[*q*1, *p*2]	[*q*2, *p*1]	1
[*q*1, *p*2]	[*q*1, *p*2]	[*q*1, *p*2]	[*q*2, *p*2]	0
[*q*2, *p*1]	[*q*2, *p*1]	[*q*2, *p*2]	[*q*2, *p*1]	0
[*q*2, *p*2]	[*q*2, *p*2]	[*q*2, *p*2]	[*q*2, *p*2]	0

Substitution

Substitution

A regular language given by the expression:

Substitution

A regular language given by the expression:

$r = 0^*(0+1)1^*$

Substitution

A regular language given by the expression:

$r = 0^*(0+1)1^*$

Substitution *R*

Substitution

A regular language given by the expression:

$r = 0^*(0+1)1^*$

Substitution *R*

Character 0

Substitution

A regular language given by the expression:

r = 0*(0+1)1*

Substitution *R*

Character 0

f*(0) = *a

Substitution

A regular language given by the expression:

$r = 0^*(0+1)1^*$

Substitution *R*

Character 0

$f(0) = a$

Character 1

Substitution

A regular language given by the expression:

$r = 0^*(0+1)1^*$

Substitution *R*

Character 0

$f(0) = a$

Character 1

$f(1) = b^*$

Substitution

A regular language given by the expression:

$r = 0^*(0+1)1^*$

Substitution R

Character 0

$f(0) = a$

Character 1

$f(1) = b^*$

We obtain a regular expression $f(R)$

Substitution

A regular language given by the expression:

$r = 0^*(0+1)1^*$

Substitution *R*

Character 0

$f(0) = a$

Character 1

$f(1) = b^*$

We obtain a regular expression *f*(*R*)

$f(\;0\;\;*\;\;(\;\;0\;\;+\;\;1\;\;)\;\;1\;\;\;*\;) =$

Substitution

A regular language given by the expression:

$r = 0^*(0+1)1^*$

Substitution R

Character 0

$f(0) = a$

Character 1

$f(1) = b^*$

We obtain a regular expression $f(R)$

$$f(\ 0\ ^*\ (\ 0\ +\ 1\ )\ 1\ ^*\ ) =$$
$$(f(0))^*\ (\ (f(0)) + (f(1))\ )\ (f(1))^* =$$

Substitution

A regular language given by the expression:

$r = 0^*(0+1)1^*$

Substitution R

Character 0

$f(0) = a$

Character 1

$f(1) = b^*$

We obtain a regular expression $f(R)$

$$f(\;0\;{}^*\;(\;0\;+\;1\;)\;1\;{}^*\;) =$$
$$(f(0))^*\;(\;(f(0)) + (f(1))\;)\;(f(1))^* =$$
$$a^*\;(\;a + b^*\;)\;(b^*)^* =$$

Substitution

A regular language given by the expression:

$r = 0^*(0+1)1^*$

Substitution R

Character 0

$f(0) = a$

Character 1

$f(1) = b^*$

We obtain a regular expression $f(R)$

$$f(\ 0\ ^*\ (\ 0\ +\ 1\ )\ 1\ ^*\ ) =$$
$$(f(0))^*\ (\ (f(0)) + (f(1))\ )\ (f(1))^* =$$
$$a^*\ (\ a + b^*\ )\ (b^*)^* =$$
$$a^*(a+b^*)(b^*)^*$$

Regular definitions

Regular definitions

$d_1 \rightarrow r_1$

Regular definitions

$d_1 \rightarrow r_1$
$d_2 \rightarrow r_2$

Regular definitions

$d_1 \rightarrow r_1$
$d_2 \rightarrow r_2$
$- - -$

Regular definitions

$d_1 \rightarrow r_1$
$d_2 \rightarrow r_2$
$---$
$d_n \rightarrow r_n$

Regular definitions

$d_1 \rightarrow r_1$
$d_2 \rightarrow r_2$
- - -
$d_n \rightarrow r_n$

r_i are regular expresions over alphabet $\Sigma \cup \{d_1, d_2, \ldots, d_{i-1}\}$

Regular definitions

$d_1 \rightarrow r_1$
$d_2 \rightarrow r_2$
- - -
$d_n \rightarrow r_n$

r_i are regular expresions over alphabet $\Sigma \cup \{d_1, d_2, \ldots, d_{i-1}\}$

---

letter $\rightarrow$ **A+B+ ... +Z+a+b+... +z**

Regular definitions

$d_1 \rightarrow r_1$
$d_2 \rightarrow r_2$
- - -
$d_n \rightarrow r_n$

r_i are regular expresions over alphabet $\Sigma \cup \{d_1, d_2, \ldots, d_{i-1}\}$

<u>*letter*</u> $\rightarrow$ **A+B+ ... +Z+a+b+... +z**

<u>*digit*</u> $\rightarrow$ **0+1+ ... +9**

Regular definitions

$d_1 \rightarrow r_1$
$d_2 \rightarrow r_2$
- - -
$d_n \rightarrow r_n$

r_i ***are regular expresions over alphabet*** $\Sigma \cup \{d_1, d_2, \ldots , d_{i-1}\}$

<u>*letter*</u> → A+B+ ... +Z+a+b+... +z

<u>*digit*</u> → 0+1+ ... +9

<u>*variable*</u> → <u>*letter*</u> (<u>*letter*</u> + <u>*digit*</u>)*

Regular definitions

$d_1 \to r_1$
$d_2 \to r_2$
- - -
$d_n \to r_n$

r_i ***are regular expresions over alphabet*** $\Sigma \cup \{d_1, d_2, \dots , d_{i\text{-}1}\}$

<u>*letter*</u> $\to$ **A+B+ ... +Z+a+b+... +z**

<u>*digit*</u> $\to$ **0+1+ ... +9**

<u>*variable*</u> $\to$ <u>*letter*</u> **(** <u>*letter*</u> **+** <u>*digit*</u> **)***

<u>*variable*</u> $\to$ **(A+B+ ... +Z+a+b+... +z)((A+B+ ... +Z+a+b+... +z)+(0+1+ ... +9))***

Regular definitions

$d_1 \rightarrow r_1$
$d_2 \rightarrow r_2$
- - -
$d_n \rightarrow r_n$

r_i are regular expresions over alphabet $\Sigma \cup \{d_1, d_2, \ldots, d_{i-1}\}$

Regular definitions

$d_1 \rightarrow r_1$
$d_2 \rightarrow r_2$
$- - -$
$d_n \rightarrow r_n$

r_i are regular expresions over alphabet $\Sigma \cup \{d_1, d_2, \ldots, d_{i-1}\}$

---

digit $\rightarrow 0 \mid 1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7 \mid 8 \mid 9$

Regular definitions

$d_1 \rightarrow r_1$
$d_2 \rightarrow r_2$
$- - -$
$d_n \rightarrow r_n$

r_i ***are regular expresions over alphabet*** $\Sigma \cup \{d_1, d_2, \ldots, d_{i-1}\}$

$\underline{digit} \rightarrow 0 \mid 1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7 \mid 8 \mid 9$

$\underline{number} \rightarrow \underline{digit}\ \underline{digit}\ ^*$

Regular definitions

$d_1 \rightarrow r_1$
$d_2 \rightarrow r_2$
- - -
$d_n \rightarrow r_n$

r_i ***are regular expresions over alphabet*** $\Sigma \cup \{d_1, d_2, \ldots, d_{i-1}\}$

---

digit $\rightarrow$ 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
number $\rightarrow$ *digit* *digit* *
decimals $\rightarrow$. *number* | ε

Regular definitions

$d_1 \rightarrow r_1$
$d_2 \rightarrow r_2$
$---$
$d_n \rightarrow r_n$

r_i are regular expresions over alphabet $\Sigma \cup \{d_1, d_2, \ldots, d_{i-1}\}$

---

digit	$\rightarrow$ 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
number	$\rightarrow$ ***digit digit****
decimals	$\rightarrow$. ***number*** | ε
exponent	$\rightarrow$ (E (+ | – | ε) ***number***) | ε

Regular definitions

$d_1 \rightarrow r_1$

$d_2 \rightarrow r_2$

- - -

$d_n \rightarrow r_n$

r_i ***are regular expresions over alphabet*** $\Sigma \cup \{d_1, d_2, \dots , d_{i-1}\}$

---

digit	$\rightarrow$ 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
number	$\rightarrow$ digit digit *
decimals	$\rightarrow$. number | ε
exponent	$\rightarrow$ (E (+ | – | ε) number) | ε
constant	$\rightarrow$ number decimals exponent

Pumping lemma

Pumping lemma

DFA *M* has *n* states

Pumping lemma

DFA *M* has *n* states

Input string $a_1 a_2$ --- a_m , $m > n$

Pumping lemma

DFA M has n states

Input string $a_1 a_2$ --- a_m , $m > n$

Sequence of states $q_0 \;\; q_1 \quad \ldots \quad q_m$

Pumping lemma

DFA M has n states

Input string $a_1 a_2$ --- a_m , $m > n$

Sequence of states $q_0 \;\; q_1 \;\; \dots \;\; q_m$

$0 \leq j < k \leq m$ *and* $q_j = q_k$

Pumping lemma

DFA M has n states

Input string $a_1 a_2$ --- a_m , $m > n$

Sequence of states $q_0 \;\; q_1 \;\; \ldots \;\; q_m$

$0 \leq j < k \leq m$ *and* $q_j = q_k$

$$q_0 \xrightarrow{a_1} q_1 \xrightarrow{a_2} q_2 \xrightarrow{a_3} \text{- - -} \; q_{j\text{-}1} \xrightarrow{a_j} q_j = q_k$$

Pumping lemma

DFA M has n states

Input string $a_1 a_2$ --- a_m , $m > n$

Sequence of states $q_0 \ \ q_1 \ \ \ldots \ \ q_m$

$0 \leq j < k \leq m$ *and* $q_j = q_k$

Substring $u = a_1 a_2$ --- a_j

$$q_0 \xrightarrow{a_1} q_1 \xrightarrow{a_2} q_2 \xrightarrow{a_3} \text{- - -} \ q_{j\text{-}1} \xrightarrow{a_j} q_j = q_k$$

Pumping lemma

DFA M has n states

Input string $a_1 a_2 \text{---} a_m$, $m > n$

Sequence of states $q_0 \;\; q_1 \;\; \ldots \;\; q_m$

$0 \leq j < k \leq m$ *and* $q_j = q_k$

Substring $u = a_1 a_2 \text{---} a_j$

$$q_0 \xrightarrow{a_1} q_1 \xrightarrow{a_2} q_2 \xrightarrow{a_3} \text{- - -} \; q_{j-1} \xrightarrow{a_j} q_j = q_k$$

$$a_{j+1} \quad q_{j+1} \xrightarrow{a_{j+2}} q_{j+2} \xrightarrow{a_{j+3}} q_{j+3} \xrightarrow{a_{j+4}} \text{- - -} \; q_{k-2} \xrightarrow{a_{k-1}} q_{k-1} \quad a_k$$

Pumping lemma

DFA M has n states

Input string $a_1\, a_2$ --- a_m , $m > n$

Sequence of states $q_0\;\; q_1\;\; \ldots\;\; q_m$

$0 \le j < k \le m$ *and* $q_j = q_k$

Substring $u = a_1\, a_2$ --- a_j

$$q_0 \xrightarrow{a_1} q_1 \xrightarrow{a_2} q_2 \xrightarrow{a_3} \cdots\; q_{j-1} \xrightarrow{a_j} q_j = q_k$$

$$q_j \xrightarrow{a_{j+1}} q_{j+1} \xrightarrow{a_{j+2}} q_{j+2} \xrightarrow{a_{j+3}} q_{j+3} \xrightarrow{a_{j+4}} \cdots\; q_{k-2} \xrightarrow{a_{k-1}} q_{k-1} \xrightarrow{a_k} q_k$$

Substring $v = a_{j+1} a_{j+2}$ --- a_k

Pumping lemma

DFA M has n states

Input string $a_1 a_2 \text{---} a_m$, $m > n$

Sequence of states $q_0 \;\; q_1 \;\; \ldots \;\; q_m$

$0 \le j < k \le m$ ***and*** $q_j = q_k$

Substring $u = a_1 a_2 \text{---} a_j$

$$q_0 \xrightarrow{a_1} q_1 \xrightarrow{a_2} q_2 \xrightarrow{a_3} \text{- - -} \; q_{j-1} \xrightarrow{a_j} q_j = q_k \xrightarrow{a_{k+1}} q_{k+1} \xrightarrow{a_{k+2}} \text{- - -} \; q_{m-1} \xrightarrow{a_m} q_m$$

$$q_j \xrightarrow{a_{j+1}} q_{j+1} \xrightarrow{a_{j+2}} q_{j+2} \xrightarrow{a_{j+3}} q_{j+3} \xrightarrow{a_{j+4}} \text{- - -} \; q_{k-2} \xrightarrow{a_{k-1}} q_{k-1} \xrightarrow{a_k} q_k$$

Substring $v = a_{j+1} a_{j+2} \text{---} a_k$

Pumping lemma

DFA *M* has *n* states

Input string $a_1 a_2 \text{---} a_m$, $m > n$

Sequence of states $q_0 \; q_1 \; \dots \; q_m$

$0 \leq j < k \leq m$ *and* $q_j = q_k$

Substring u $= a_1 a_2 \text{---} a_j$

Substring w $= a_{k+1} a_{k+2} \text{---} a_m$

$$q_0 \xrightarrow{a_1} q_1 \xrightarrow{a_2} q_2 \xrightarrow{a_3} \text{- - -} \; q_{j\text{-}1} \xrightarrow{a_j} q_j = q_k \xrightarrow{a_{k+1}} q_{k+1} \xrightarrow{a_{k+2}} \text{- - -} \; q_{m\text{-}1} \xrightarrow{a_m} q_m$$

$$q_j \xrightarrow{a_{j+1}} q_{j+1} \xrightarrow{a_{j+2}} q_{j+2} \xrightarrow{a_{j+3}} q_{j+3} \xrightarrow{a_{j+4}} \text{- - -} \; q_{k\text{-}2} \xrightarrow{a_{k\text{-}1}} q_{k\text{-}1} \xrightarrow{a_k} q_k$$

Substring v $= a_{j+1} a_{j+2} \text{---} a_k$

Pumping lemma

Substring $u = a_1\, a_2$ --- a_j

Substring $w = a_{k+1}\, a_{k+2}$ --- a_m

$$q_0 \xrightarrow{a_1} q_1 \xrightarrow{a_2} q_2 \xrightarrow{a_3} \cdots\ q_{j-1} \xrightarrow{a_j} q_j = q_k \xrightarrow{a_{k+1}} q_{k+1} \xrightarrow{a_{k+2}} \cdots\ q_{m-1} \xrightarrow{a_m} q_m$$

$$q_j \xrightarrow{a_{j+1}} q_{j+1} \xrightarrow{a_{j+2}} q_{j+2} \xrightarrow{a_{j+3}} q_{j+3} \xrightarrow{a_{j+4}} \cdots\ q_{k-2} \xrightarrow{a_{k-1}} q_{k-1} \xrightarrow{a_k} q_k$$

Substring $v = a_{j+1} a_{j+2}$ --- a_k

Pumping lemma

The following input strings are accepted:

Substring u **=** $a_1 a_2 \text{---} a_j$

Substring w **=** $a_{k+1} a_{k+2} \text{---} a_m$

$$q_0 \xrightarrow{a_1} q_1 \xrightarrow{a_2} q_2 \xrightarrow{a_3} \text{- - -}\ q_{j-1} \xrightarrow{a_j} q_j = q_k \xrightarrow{a_{k+1}} q_{k+1} \xrightarrow{a_{k+2}} \text{- - -}\ q_{m-1} \xrightarrow{a_m} q_m$$

$$q_j \xrightarrow{a_{j+1}} q_{j+1} \xrightarrow{a_{j+2}} q_{j+2} \xrightarrow{a_{j+3}} q_{j+3} \xrightarrow{a_{j+4}} \text{- - -}\ q_{k-2} \xrightarrow{a_{k-1}} q_{k-1} \xrightarrow{a_k} q_k$$

Substring v **=** $a_{j+1} a_{j+2} \text{---} a_k$

Pumping lemma

The following input strings are accepted:

$u\ w$

Substring $u = a_1\, a_2 \text{ --- } a_j$

Substring $w = a_{k+1}\, a_{k+2} \text{ --- } a_m$

$$q_0 \xrightarrow{a_1} q_1 \xrightarrow{a_2} q_2 \xrightarrow{a_3} \text{- - -}\ q_{j-1} \xrightarrow{a_j} q_j = q_k \xrightarrow{a_{k+1}} q_{k+1} \xrightarrow{a_{k+2}} \text{- - -}\ q_{m-1} \xrightarrow{a_m} q_m$$

$$q_j \xrightarrow{a_{j+1}} q_{j+1} \xrightarrow{a_{j+2}} q_{j+2} \xrightarrow{a_{j+3}} q_{j+3} \xrightarrow{a_{j+4}} \text{- - -}\ q_{k-2} \xrightarrow{a_{k-1}} q_{k-1} \xrightarrow{a_k} q_k$$

Substring $v = a_{j+1} a_{j+2} \text{ --- } a_k$

Pumping lemma

The following input strings are accepted:

u w

Substring u $= a_1\, a_2$ --- a_j

Substring w $= a_{k+1}\, a_{k+2}$ --- a_m

$$q_0 \xrightarrow{a_1} q_1 \xrightarrow{a_2} q_2 \xrightarrow{a_3} \text{- - -}\ q_{j\text{-}1} \xrightarrow{a_j} q_j = q_k \xrightarrow{a_{k+1}} q_{k+1} \xrightarrow{a_{k+2}} \text{- - -}\ q_{m\text{-}1} \xrightarrow{a_m} q_m$$

$$q_j \xrightarrow{a_{j+1}} q_{j+1} \xrightarrow{a_{j+2}} q_{j+2} \xrightarrow{a_{j+3}} q_{j+3} \xrightarrow{a_{j+4}} \text{- - -}\ q_{k\text{-}2} \xrightarrow{a_{k\text{-}1}} q_{k\text{-}1} \xrightarrow{a_k} q_k$$

Substring v $= a_{j+1}a_{j+2}$ --- a_k

Pumping lemma

The following input strings are accepted:

u w

u v w

Substring u **=** $a_1 a_2 \text{---} a_j$

Substring w **=** $a_{k+1} a_{k+2} \text{---} a_m$

$$q_0 \xrightarrow{a_1} q_1 \xrightarrow{a_2} q_2 \xrightarrow{a_3} \text{- - -}\ q_{j-1} \xrightarrow{a_j} q_j = q_k \xrightarrow{a_{k+1}} q_{k+1} \xrightarrow{a_{k+2}} \text{- - -}\ q_{m-1} \xrightarrow{a_m} q_m$$

$$q_j \xrightarrow{a_{j+1}} q_{j+1} \xrightarrow{a_{j+2}} q_{j+2} \xrightarrow{a_{j+3}} q_{j+3} \xrightarrow{a_{j+4}} \text{- - -}\ q_{k-2} \xrightarrow{a_{k-1}} q_{k-1} \xrightarrow{a_k} q_k$$

Substring v **=** $a_{j+1} a_{j+2} \text{---} a_k$

Pumping lemma

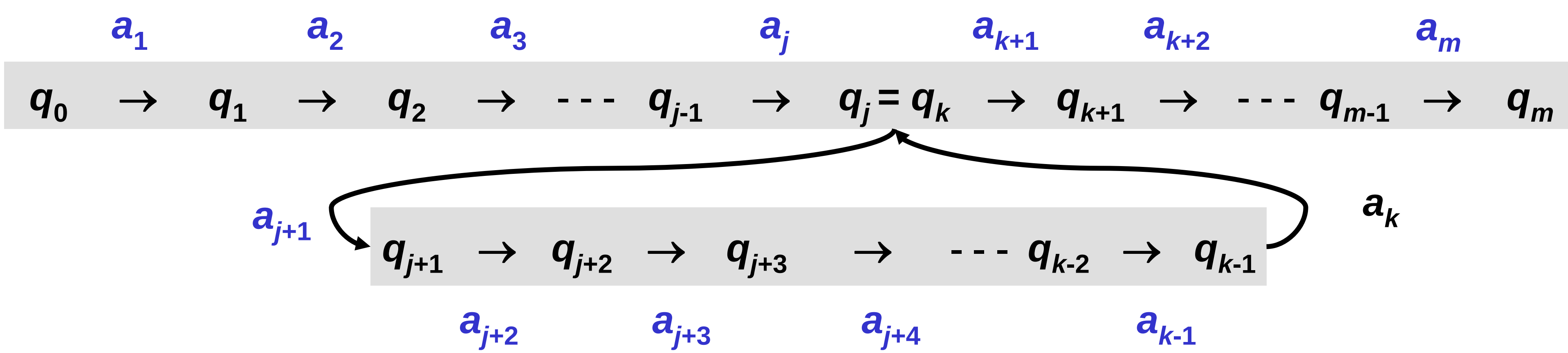
The following input strings are accepted:

$$u\ w$$

$$u\ v\ w$$

Substring $u = a_1\, a_2 \text{ --- } a_j$

Substring $w = a_{k+1}\, a_{k+2} \text{ --- } a_m$



Substring $v = a_{j+1} a_{j+2} \text{ --- } a_k$

Pumping lemma

The following input strings are accepted:

u w

u v w

Substring *u* = $a_1 a_2 \cdots a_j$

Substring *w* = $a_{k+1} a_{k+2} \cdots a_m$

$$q_0 \xrightarrow{a_1} q_1 \xrightarrow{a_2} q_2 \xrightarrow{a_3} \cdots\ q_{j-1} \xrightarrow{a_j} q_j = q_k \xrightarrow{a_{k+1}} q_{k+1} \xrightarrow{a_{k+2}} \cdots\ q_{m-1} \xrightarrow{a_m} q_m$$

$$q_j \xrightarrow{a_{j+1}} q_{j+1} \xrightarrow{a_{j+2}} q_{j+2} \xrightarrow{a_{j+3}} q_{j+3} \xrightarrow{a_{j+4}} \cdots\ q_{k-2} \xrightarrow{a_{k-1}} q_{k-1} \xrightarrow{a_k} q_k$$

Substring *v* = $a_{j+1} a_{j+2} \cdots a_k$

Pumping lemma

The following input strings are accepted:

$u\ w$

$u\ v\ w$

Substring $u = a_1\, a_2$ --- a_j

Substring $w = a_{k+1}\, a_{k+2}$ --- a_m

$$q_0 \xrightarrow{a_1} q_1 \xrightarrow{a_2} q_2 \xrightarrow{a_3} \text{- - -}\ q_{j-1} \xrightarrow{a_j} q_j = q_k \xrightarrow{a_{k+1}} q_{k+1} \xrightarrow{a_{k+2}} \text{- - -}\ q_{m-1} \xrightarrow{a_m} q_m$$

$$q_j \xrightarrow{a_{j+1}} q_{j+1} \xrightarrow{a_{j+2}} q_{j+2} \xrightarrow{a_{j+3}} q_{j+3} \xrightarrow{a_{j+4}} \text{- - -}\ q_{k-2} \xrightarrow{a_{k-1}} q_{k-1} \xrightarrow{a_k} q_k$$

Substring $v = a_{j+1} a_{j+2}$ --- a_k

Pumping lemma

The following input strings are accepted:

$$u\ w$$

$$u\ v\ w$$

$$u\ v^2\ w$$

Substring $u = a_1\, a_2 \text{ --- } a_j$

Substring $w = a_{k+1}\, a_{k+2} \text{ --- } a_m$

$$q_0 \xrightarrow{a_1} q_1 \xrightarrow{a_2} q_2 \xrightarrow{a_3} \text{- - -}\ q_{j-1} \xrightarrow{a_j} q_j = q_k \xrightarrow{a_{k+1}} q_{k+1} \xrightarrow{a_{k+2}} \text{- - -}\ q_{m-1} \xrightarrow{a_m} q_m$$

$$q_j \xrightarrow{a_{j+1}} q_{j+1} \xrightarrow{a_{j+2}} q_{j+2} \xrightarrow{a_{j+3}} q_{j+3} \xrightarrow{a_{j+4}} \text{- - -}\ q_{k-2} \xrightarrow{a_{k-1}} q_{k-1} \xrightarrow{a_k} q_k$$

Substring $v = a_{j+1} a_{j+2} \text{ --- } a_k$

Pumping lemma

The following input strings are accepted:

$$u\ w$$

$$u\ v\ w$$

$$u\ v^2\ w$$

Substring u $= a_1\, a_2 \text{ --- } a_j$

Substring w $= a_{k+1}\, a_{k+2} \text{ --- } a_m$

$$q_0 \xrightarrow{a_1} q_1 \xrightarrow{a_2} q_2 \xrightarrow{a_3} \text{- - -}\ q_{j-1} \xrightarrow{a_j} q_j = q_k \xrightarrow{a_{k+1}} q_{k+1} \xrightarrow{a_{k+2}} \text{- - -}\ q_{m-1} \xrightarrow{a_m} q_m$$

$$q_j \xrightarrow{a_{j+1}} q_{j+1} \xrightarrow{a_{j+2}} q_{j+2} \xrightarrow{a_{j+3}} q_{j+3} \xrightarrow{a_{j+4}} \text{- - -}\ q_{k-2} \xrightarrow{a_{k-1}} q_{k-1} \xrightarrow{a_k} q_k$$

Substring v $= a_{j+1}a_{j+2} \text{ --- } a_k$

Pumping lemma

The following input strings are accepted:

$$u\,w$$

$$u\,v\,w$$

$$u\,v^2\,w$$

Substring $u = a_1\,a_2$ --- a_j

Substring $w = a_{k+1}\,a_{k+2}$ --- a_m

$$q_0 \xrightarrow{a_1} q_1 \xrightarrow{a_2} q_2 \xrightarrow{a_3} \text{- - -}\; q_{j-1} \xrightarrow{a_j} q_j = q_k \xrightarrow{a_{k+1}} q_{k+1} \xrightarrow{a_{k+2}} \text{- - -}\; q_{m-1} \xrightarrow{a_m} q_m$$

$$q_j \xrightarrow{a_{j+1}} q_{j+1} \xrightarrow{a_{j+2}} q_{j+2} \xrightarrow{a_{j+3}} q_{j+3} \xrightarrow{a_{j+4}} \text{- - -}\; q_{k-2} \xrightarrow{a_{k-1}} q_{k-1} \xrightarrow{a_k} q_k$$

Substring $v = a_{j+1}a_{j+2}$ --- a_k

Pumping lemma

The following input strings are accepted:

$$u\ w$$

$$u\ v\ w$$

$$u\ v^2\ w$$

Substring $u = a_1\, a_2 \text{ --- } a_j$

Substring $w = a_{k+1}\, a_{k+2} \text{ --- } a_m$

$$q_0 \xrightarrow{a_1} q_1 \xrightarrow{a_2} q_2 \xrightarrow{a_3} \text{- - -}\ q_{j-1} \xrightarrow{a_j} q_j = q_k \xrightarrow{a_{k+1}} q_{k+1} \xrightarrow{a_{k+2}} \text{- - -}\ q_{m-1} \xrightarrow{a_m} q_m$$

$$q_j \xrightarrow{a_{j+1}} q_{j+1} \xrightarrow{a_{j+2}} q_{j+2} \xrightarrow{a_{j+3}} q_{j+3} \xrightarrow{a_{j+4}} \text{- - -}\ q_{k-2} \xrightarrow{a_{k-1}} q_{k-1} \xrightarrow{a_k} q_k$$

Substring $v = a_{j+1} a_{j+2} \text{ --- } a_k$

Pumping lemma

The following input strings are accepted:

$$u\ w$$

$$u\ v\ w$$

$$u\ v^2\ w$$

Substring $u = a_1\, a_2 \text{ --- } a_j$

Substring $w = a_{k+1}\, a_{k+2} \text{ --- } a_m$

$$q_0 \xrightarrow{a_1} q_1 \xrightarrow{a_2} q_2 \xrightarrow{a_3} \text{- - -}\ q_{j-1} \xrightarrow{a_j} q_j = q_k \xrightarrow{a_{k+1}} q_{k+1} \xrightarrow{a_{k+2}} \text{- - -}\ q_{m-1} \xrightarrow{a_m} q_m$$

$$q_j \xrightarrow{a_{j+1}} q_{j+1} \xrightarrow{a_{j+2}} q_{j+2} \xrightarrow{a_{j+3}} q_{j+3} \xrightarrow{a_{j+4}} \text{- - -}\ q_{k-2} \xrightarrow{a_{k-1}} q_{k-1} \xrightarrow{a_k} q_k$$

Substring $v = a_{j+1} a_{j+2} \text{ --- } a_k$

Pumping lemma

The following input strings are accepted:

$$u\ w$$

$$u\ v\ w$$

$$u\ v^2\ w$$

$$u\ v^i\ w$$

Substring $u = a_1 a_2 \cdots a_j$

Substring $w = a_{k+1} a_{k+2} \cdots a_m$

$$q_0 \xrightarrow{a_1} q_1 \xrightarrow{a_2} q_2 \xrightarrow{a_3} \cdots\ q_{j-1} \xrightarrow{a_j} q_j = q_k \xrightarrow{a_{k+1}} q_{k+1} \xrightarrow{a_{k+2}} \cdots\ q_{m-1} \xrightarrow{a_m} q_m$$

$$q_j \xrightarrow{a_{j+1}} q_{j+1} \xrightarrow{a_{j+2}} q_{j+2} \xrightarrow{a_{j+3}} q_{j+3} \xrightarrow{a_{j+4}} \cdots\ q_{k-2} \xrightarrow{a_{k-1}} q_{k-1} \xrightarrow{a_k} q_k$$

Substring $v = a_{j+1} a_{j+2} \cdots a_k$

Pumping lemma

Pumping lemma

***i*) *L* – regular language**

Pumping lemma

***i*) *L* – regular language**

there is a constant integer *n*

Pumping lemma

i) ***L*** **– regular language**

there is a constant integer *n*

Pumping lemma

***i*) *L* – regular language**

there is a constant integer *n*

if a string z, $|z|>n$, $z=uvw$, $|uv|\leq n$ and $1\leq|v|$

Pumping lemma

i) *L* – regular language

there is a constant integer n

if a string z, $|z|>n$, $z=uvw$, $|uv|\leq n$ and $1\leq|v|$ is an element of L

Pumping lemma

i) L – regular language

there is a constant integer n

if a string z, $|z|>n$, $z=uvw$, $|uv|\leq n$ and $1\leq|v|$
is an element of L

for any $i\geq 0$

Pumping lemma

i) L – regular language

there is a constant integer n

if a string z, $|z|>n$, $z=uvw$, $|uv|\leq n$ and $1\leq|v|$
is an element of L

for any $i\geq 0$
string uv^iw is also an element of L

Pumping lemma

$$N_3 = \{\ a^k b^k\ \}$$

Pumping lemma

$$N_3 = \{ a^k b^k \}$$

1) **Assume that N_3 is a regular language**

Pumping lemma

$$N_3 = \{ a^k b^k \}$$

1) Assume that N_3 is a regular language

2) Let $z = a^m b^m$ be a string in N_3 for which $|z| = 2m$ i $m > n$

Pumping lemma

$N_3 = \{ a^k b^k \}$

1) Assume that N_3 is a regular language

2) Let $z = a^m b^m$ be a string in N_3 for which $|z| = 2m$ i $m > n$

3) String z can be written as uvw where $1 \leq |v| \leq |uv| \leq n$

Pumping lemma

$N_3 = \{ a^k b^k \}$

1) Assume that N_3 is a regular language

2) Let $z = a^m b^m$ be a string in N_3 for which $|z| = 2m$ i $m > n$

3) String z can be written as uvw where $1 \leq |v| \leq |uv| \leq n$

4) $i=0$

$|uv| \leq n \Rightarrow$ prefix uv contains only symbols a

$\Rightarrow$ string v contains only symbols a

Pumping lemma

$N_3 = \{\ a^k b^k\ \}$

1) Assume that N_3 is a regular language

2) Let $z = a^m b^m$ be a string in N_3 for which $|z| = 2m$ i $m > n$

3) String z can be written as uvw where $1 \leq |v| \leq |uv| \leq n$

4) $i=0$

$|uv| \leq n \Rightarrow$ prefix uv contains only symbols a

$\Rightarrow$ string v contains only symbols a

1 □ $|v| \Rightarrow$ string uw has fewer a than b

Pumping lemma

$N_3 = \{\ a^k b^k\ \}$

1) **Assume that N_3 is a regular language**

2) **Let $z = a^m b^m$ be a string in N_3 for which $|z| = 2m$ i $m > n$**

3) **String z can be written as uvw where $1 \leq |v| \leq |uv| \leq n$**

4) **$i=0$**

   $|uv| \leq n \Rightarrow$ **prefix uv contains only symbols a**

   $\Rightarrow$ **string v contains only symbols a**

   1 □ $|v| \Rightarrow$ **string uw has fewer a than b**

5) **Regardless of n**

Pumping lemma

$N_3 = \{ a^k b^k \}$

1) Assume that N_3 is a regular language

2) Let $z = a^m b^m$ be a string in N_3 for which $|z| = 2m$ i $m > n$

3) String z can be written as uvw where $1 \leq |v| \leq |uv| \leq n$

4) $i=0$

$|uv| \leq n \Rightarrow$ prefix uv contains only symbols a

$\Rightarrow$ string v contains only symbols a

1 □ $|v| \Rightarrow$ string uw has fewer a than b

5) Regardless of n

Regardless of a lenght of z□N_3

Pumping lemma

$$N_3 = \{\, a^k b^k \,\}$$

1) **Assume that N_3 is a regular language**

2) **Let $z = a^m b^m$ be a string in N_3 for which $|z| = 2m$ i $m > n$**

3) **String z can be written as uvw where $1 \leq |v| \leq |uv| \leq n$**

4) **$i=0$**

$|uv| \leq n \Rightarrow$ prefix uv contains only symbols a

$\Rightarrow$ string v contains only symbols a

1 □ $|v| \Rightarrow$ string uw has fewer a than b

5) **Regardless of n**

Regardless of a lenght of z□N_3

Regardless of a decomposition into uvw

Pumping lemma

$$N_3 = \{ a^k b^k \}$$

1) Assume that N_3 is a regular language

2) Let $z = a^m b^m$ be a string in N_3 for which $|z| = 2m$ i $m > n$

3) String z can be written as uvw where $1 \leq |v| \leq |uv| \leq n$

4) $i=0$

$|uv| \leq n \Rightarrow$ prefix uv contains only symbols a

$\Rightarrow$ string v contains only symbols a

1 □ $|v| \Rightarrow$ string uw has fewer a than b

5) Regardless of n

Regardless of a lenght of z□N_3

Regardless of a decomposition into uvw

string uw is not an element of N_3

Pumping lemma

$$N_2 = \{ 0^{k^2} \}$$

Pumping lemma

$$N_2 = \{ 0^{k^2} \}$$

1) Assume that N_2 is a regular language

Pumping lemma

$$N_2 = \{ 0^{k^2} \}$$

1) Assume that N_2 is a regular language

2) Let $z=0^{n2}$ be a string in N_2 for which $|z|=n^2$ and $|z|>n$

Pumping lemma

$$N_2 = \{\ 0^{k^2}\ \}$$

1) Assume that N_2 is a regular language

2) Let $z=0^{n2}$ be a string in N_2 for which $|z|=n^2$ and $|z|>n$

3) String z can be written as uvw where $1 \leq |v| \leq |uv| \leq n$

Pumping lemma

$$N_2 = \{ 0^{k^2} \}$$

1) Assume that N_2 is a regular language

2) Let $z=0^{n^2}$ be a string in N_2 for which $|z|=n^2$ and $|z|>n$

3) String z can be written as uvw where $1\leq|v|\leq|uv|\leq n$

4) $i=2$

Pumping lemma

$$N_2 = \{ 0^{k^2} \}$$

1) Assume that N_2 is a regular language

2) Let $z=0^{n2}$ be a string in N_2 for which $|z|=n^2$ and $|z|>n$

3) String z can be written as uvw where $1 \le |v| \le |uv| \le n$

4) $i=2$

$|v| \le |uv|$ □n

Pumping lemma

$$N_2 = \{ 0^{k^2} \}$$

1) Assume that N_2 is a regular language

2) Let $z=0^{n^2}$ be a string in N_2 for which $|z|=n^2$ and $|z|>n$

3) String z can be written as uvw where $1 \leq |v| \leq |uv| \leq n$

4) $i=2$

$|v| \leq |uv| \square n$

then $|uvw| = |z| = n^2 < |uv^2w| = (n^2 + |v|) \leq (n^2+n)$

Pumping lemma

$$N_2 = \{\, 0^{k^2} \,\}$$

1) Assume that N_2 is a regular language

2) Let $z=0^{n^2}$ be a string in N_2 for which $|z|=n^2$ and $|z|>n$

3) String z can be written as uvw where $1\leq|v|\leq|uv|\leq n$

4) $i=2$

$|v|\leq|uv|$ □n

then $|uvw|=|z|=n^2 < |uv^2w| = (n^2+|v|) \leq (n^2+n)$

$(n^2+n) < (n+1)^2$

Pumping lemma

$$N_2 = \{ 0^{k^2} \}$$

1) Assume that N_2 is a regular language

2) Let $z=0^{n^2}$ be a string in N_2 for which $|z|=n^2$ and $|z|>n$

3) String z can be written as uvw where $1 \le |v| \le |uv| \le n$

4) $i=2$

$|v| \le |uv|$ □n

then $|uvw| = |z| = n^2 < |uv^2w| = (n^2 + |v|) \le (n^2+n)$

$(n^2+n) < (n+1)^2$

then $n^2 < |uv^2w| < (n+1)^2$

Pumping lemma

$$N_2 = \{\ 0^{k^2}\ \}$$

1) Assume that N_2 is a regular language

2) Let $z=0^{n^2}$ be a string in N_2 for which $|z|=n^2$ and $|z|>n$

3) String z can be written as uvw where $1\leq|v|\leq|uv|\leq n$

4) $i=2$

$|v|\leq|uv|\square n$

then $|uvw| = |z| = n^2 < |uv^2w| = (n^2+|v|) \leq (n^2+n)$

$(n^2+n) < (n+1)^2$

then $n^2 < |uv^2w| < (n+1)^2$

the length of uv^2w is not a perfect square

Pumping lemma

$$N_2 = \{ 0^{k^2} \}$$

1) **Assume that N_2 is a regular language**

2) **Let $z=0^{n^2}$ be a string in N_2 for which $|z|=n^2$ and $|z|>n$**

3) **String z can be written as uvw where $1 \leq |v| \leq |uv| \leq n$**

4) $i=2$

$|v| \leq |uv|$ □ n

then $|uvw| = |z| = n^2 < |uv^2w| = (n^2+|v|) \leq (n^2+n)$

$(n^2+n) < (n+1)^2$

then $n^2 < |uv^2w| < (n+1)^2$

the length of uv^2w is not a perfect square

5) **Regardless of n**

Pumping lemma

$$N_2 = \{\ 0^{k^2}\ \}$$

1) Assume that N_2 is a regular language

2) Let $z=0^{n^2}$ be a string in N_2 for which $|z|=n^2$ and $|z|>n$

3) String z can be written as uvw where $1 \le |v| \le |uv| \le n$

4) $i=2$

$|v| \le |uv| \Box n$

then $|uvw| = |z| = n^2 < |uv^2w| = (n^2+|v|) \le (n^2+n)$

$(n^2+n) < (n+1)^2$

then $n^2 < |uv^2w| < (n+1)^2$

the length of uv^2w is not a perfect square

5) Regardless of n

Regardless of a lenght of $z \Box N_2$

Pumping lemma

$$N_2 = \{\ 0^{k^2}\ \}$$

1) Assume that N_2 is a regular language

2) Let $z=0^{n^2}$ be a string in N_2 for which $|z|=n^2$ and $|z|>n$

3) String z can be written as uvw where $1 \leq |v| \leq |uv| \leq n$

4) $i=2$

$|v| \leq |uv| \Box n$

then $|uvw| = |z| = n^2 < |uv^2w| = (n^2+|v|) \leq (n^2+n)$

$(n^2+n) < (n+1)^2$

then $n^2 < |uv^2w| < (n+1)^2$

the length of uv^2w is not a perfect square

5) Regardless of n

Regardless of a lenght of $z \Box N_2$

Regardless of a decomposition into uvw

Pumping lemma

$$N_2 = \{\, 0^{k^2} \,\}$$

1) Assume that N_2 is a regular language

2) Let $z=0^{n^2}$ be a string in N_2 for which $|z|=n^2$ and $|z|>n$

3) String z can be written as uvw where $1 \leq |v| \leq |uv| \leq n$

4) $i=2$

$|v| \leq |uv| \square n$

then $|uvw| = |z| = n^2 < |uv^2w| = (n^2+|v|) \leq (n^2+n)$

$(n^2+n) < (n+1)^2$

then $n^2 < |uv^2w| < (n+1)^2$

the length of uv^2w is not a perfect square

5) Regardless of n

Regardless of a lenght of $z \square N_2$

Regardless of a decomposition into uvw

string uv^2w is not an element of N_2

Pumping lemma

$N_4 = \{ 0^k \}$, k prime

Pumping lemma

$N_4 = \{\ 0^k\ \}$, k prime

1) **Assume that N_4 is a regular language**

Pumping lemma

$N_4 = \{ 0^k \}$, k prime

1) **Assume that N_4 is a regular language**

2) **Let $z=0^m$ be an array in N_4 for which $|z|=m$ is prime**

Pumping lemma

$$N_4 = \{\ 0^k\ \},\ k \text{ prime}$$

1) Assume that N_4 is a regular language

2) Let $z=0^m$ be an array in N_4 for which $|z|=m$ is prime

since there are infinitely many primes, we can achieve

Pumping lemma

$N_4 = \{ 0^k \}$, k prime

1) **Assume that N_4 is a regular language**

2) **Let $z=0^m$ be an array in N_4 for which $|z|=m$ is prime**

   since there are infinitely many primes, we can achieve

   $|z| = m > n+1$

Pumping lemma

$N_4 = \{ 0^k \}$, k prime

1) **Assume that N_4 is a regular language**

2) **Let $z=0^m$ be an array in N_4 for which $|z|=m$ is prime**

   since there are infinitely many primes, we can achieve $|z| = m > n+1$

3) **String z can be written as uvw where $1 \leq |v| \leq |uv| \leq n$**

Pumping lemma

$N_4 = \{\ 0^k\ \}$, k prime

1) Assume that N_4 is a regular language

2) Let $z=0^m$ be an array in N_4 for which $|z|=m$ is prime
since there are infinitely many primes, we can achieve
$|z| = m > n+1$

3) String z can be written as uvw where $1 \le |v| \le |uv| \le n$
therefore $(\ |uvw| > n+1$ and $|uv| \le n\ ) \Rightarrow 1 < |w|$

Pumping lemma

$N_4 = \{ 0^k \}$, k prime

1) Assume that N_4 is a regular language

2) Let $z=0^m$ be an array in N_4 for which $|z|=m$ is prime
since there are infinitely many primes, we can achieve
$|z| = m > n+1$

3) String z can be written as uvw where $1 \leq |v| \leq |uv| \leq n$
therefore $(|uvw| > n+1$ and $|uv| \leq n) \Rightarrow 1 < |w|$

4) $i = |uw|$

Pumping lemma

$N_4 = \{ 0^k \}$, k prime

1) Assume that N_4 is a regular language

2) Let $z=0^m$ be an array in N_4 for which $|z|=m$ is prime
since there are infinitely many primes, we can achieve
$|z| = m > n+1$

3) String z can be written as uvw where $1 \leq |v| \leq |uv| \leq n$
therefore $(|uvw| > n+1$ and $|uv| \leq n) \Rightarrow 1 < |w|$

4) $i = |uw|$
$1 < |w| \Rightarrow 1 < |uw|$

Pumping lemma

$N_4 = \{ 0^k \}$, k prime

1) Assume that N_4 is a regular language

2) Let $z=0^m$ be an array in N_4 for which $|z|=m$ is prime

since there are infinitely many primes, we can achieve

$|z| = m > n+1$

3) String z can be written as uvw where $1 \le |v| \le |uv| \le n$

therefore $(|uvw| > n+1$ and $|uv| \le n) \Rightarrow 1 < |w|$

4) $i = |uw|$

$1 < |w| \Rightarrow 1 < |uw|$

then $|uv^iw| = |uw| + |uw|\,|v| = |uw|\,(1 + |v|)$

Pumping lemma

$N_4 = \{ 0^k \}$, k prime

1) Assume that N_4 is a regular language

2) Let $z=0^m$ be an array in N_4 for which $|z|=m$ is prime
   since there are infinitely many primes, we can achieve
   $|z| = m > n+1$

3) String z can be written as uvw where $1 \leq |v| \leq |uv| \leq n$
   therefore $(|uvw| > n+1$ and $|uv| \leq n) \Rightarrow 1 < |w|$

4) $i = |uw|$
   $1 < |w| \Rightarrow 1 < |uw|$
   then $|uv^iw| = |uw| + |uw| |v| = |uw| (1 + |v|)$
   $1 < |uw|$ i $1 < (1 + |v|) \Rightarrow |uv^iw|$ is not prime

Pumping lemma

$N_4 = \{ 0^k \}$, k prime

1) Assume that N_4 is a regular language

2) Let $z=0^m$ be an array in N_4 for which $|z|=m$ is prime
since there are infinitely many primes, we can achieve
$|z| = m > n+1$

3) String z can be written as uvw where $1 \le |v| \le |uv| \le n$
therefore $(|uvw| > n+1$ and $|uv| \le n) \Rightarrow 1 < |w|$

4) $i = |uw|$
$1 < |w| \Rightarrow 1 < |uw|$
then $|uv^iw| = |uw| + |uw|\,|v| = |uw|\,(1 + |v|)$
$1 < |uw|$ i $1 < (1 + |v|) \Rightarrow |uv^iw|$ is not prime

5) Regardless of n

Pumping lemma

$N_4 = \{ 0^k \}$, k prime

1) Assume that N_4 is a regular language

2) Let $z=0^m$ be an array in N_4 for which $|z|=m$ is prime
since there are infinitely many primes, we can achieve
$|z| = m > n+1$

3) String z can be written as uvw where $1 \leq |v| \leq |uv| \leq n$
therefore $(|uvw| > n+1$ and $|uv| \leq n) \Rightarrow 1 < |w|$

4) $i = |uw|$
$1 < |w| \Rightarrow 1 < |uw|$
then $|uv^iw| = |uw| + |uw|\,|v| = |uw|\,(1 + |v|)$
$1 < |uw|$ i $1 < (1 + |v|) \Rightarrow |uv^iw|$ is not prime

5) Regardless of n
Regardless of a lenght of $z \square N_4$

Pumping lemma

$N_4 = \{ 0^k \}$, k prime

1) Assume that N_4 is a regular language

2) Let $z=0^m$ be an array in N_4 for which $|z|=m$ is prime

since there are infinitely many primes, we can achieve $|z| = m > n+1$

3) String z can be written as uvw where $1 \le |v| \le |uv| \le n$

therefore $(|uvw| > n+1$ and $|uv| \le n) \Rightarrow 1 < |w|$

4) $i = |uw|$

$1 < |w| \Rightarrow 1 < |uw|$

then $|uv^iw| = |uw| + |uw| |v| = |uw| (1 + |v|)$

$1 < |uw|$ i $1 < (1 + |v|) \Rightarrow |uv^iw|$ is not prime

5) Regardless of n

Regardless of a lenght of z□N_4

Regardless of a decomposition into uvw

Pumping lemma

$N_4 = \{ 0^k \}$, k prime

1) Assume that N_4 is a regular language

2) Let $z=0^m$ be an array in N_4 for which $|z|=m$ is prime
since there are infinitely many primes, we can achieve
$|z| = m > n+1$

3) String z can be written as uvw where $1 \le |v| \le |uv| \le n$
therefore $(|uvw| > n+1$ and $|uv| \le n) \Rightarrow 1 < |w|$

4) $i = |uw|$
$1 < |w| \Rightarrow 1 < |uw|$
then $|uv^iw| = |uw| + |uw|\,|v| = |uw|\,(1 + |v|)$
$1 < |uw|$ i $1 < (1 + |v|) \Rightarrow |uv^iw|$ is not prime

5) Regardless of n
Regardless of a lenght of z□N_4
Regardless of a decomposition into uvw
string uv^iw is not an element of N_4

Decision algorithms

Decision algorithms

- **Non-emptiness**

Decision algorithms

- Non-emptiness
  - **A regular language $L(M)$ is non-empty if and only if DFA M accepts a string z shorter than n, i.e. $|z|<n$.**
    - If the set of reachable states contains at least one accepting state, the regular language $L(M)$ is non-empty.

Decision algorithms

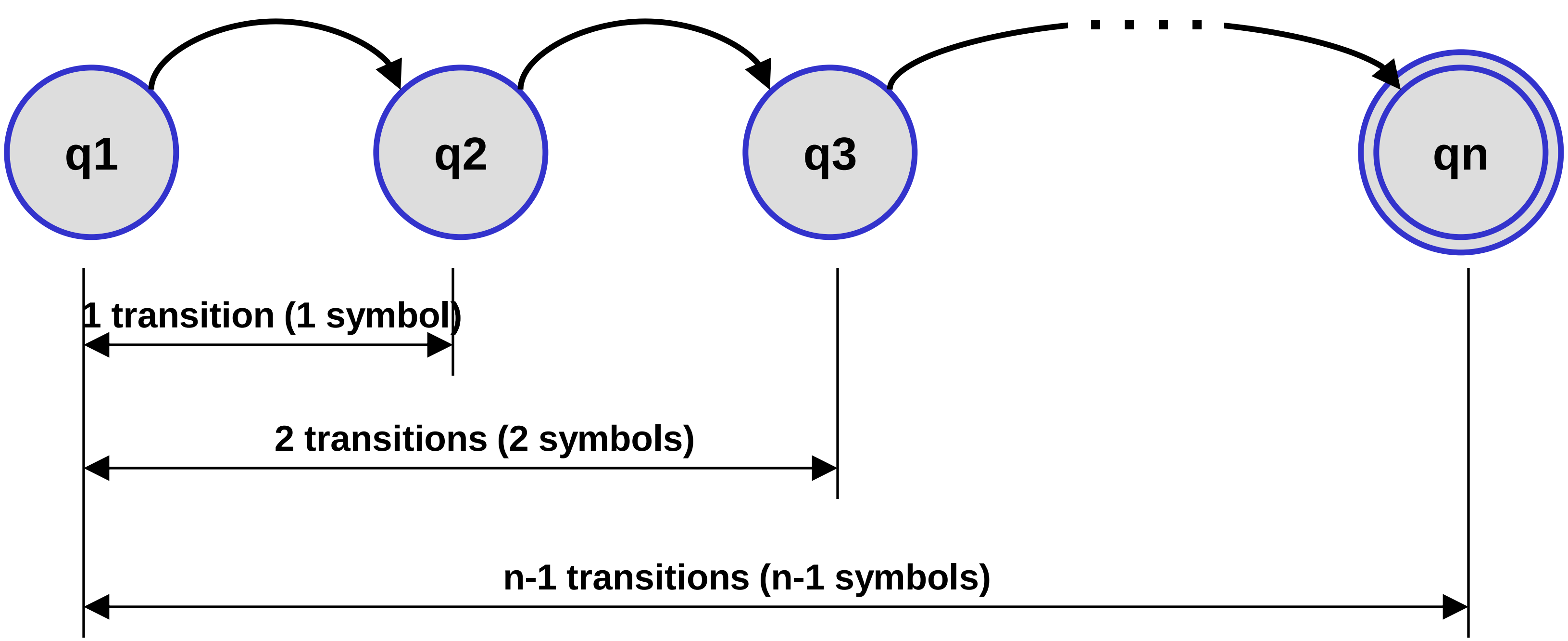
- **Non-emptiness**
  - **A regular language *L*(*M*) is non-empty if and only if DFA *M* accepts a string *z* shorter than *n*, i.e. $|z|<n$.**
    - If the set of reachable states contains at least one accepting state, the regular language *L*(*M*) is non-empty.

Lenght of the shortest path to an accepting state

Decision algorithms

- Non-emptiness
  - **A regular language *L*(*M*) is non-empty if and only if DFA *M* accepts a string *z* shorter than *n*, i.e. $|z|<n$.**
    - If the set of reachable states contains at least one accepting state, the regular language *L*(*M*) is non-empty.

Lenght of the shortest path to an accepting state



Decision algorithms

Decision algorithms

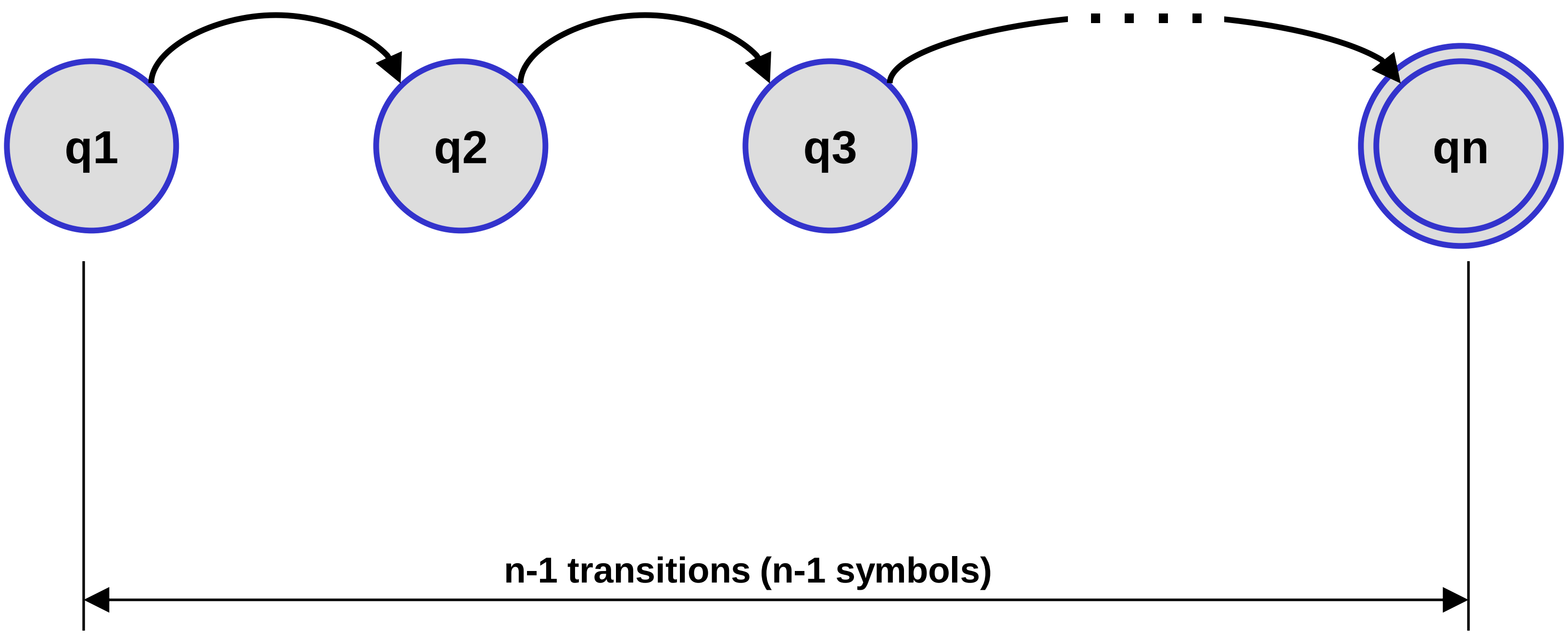
- **Infinity**

Decision algorithms

- **Infinity**
  - **A regular language $L(M)$ is infinite if and only if DFA M accepts a string of length l, where $n \leq l < 2n$.**
    - We remove all non-accepting states for which there is no sequence of transitions into an accepting state
    - If the obtained state diagram of DFA M' contains at least one closed loop, then the regular language $L(M)$ is infinite.

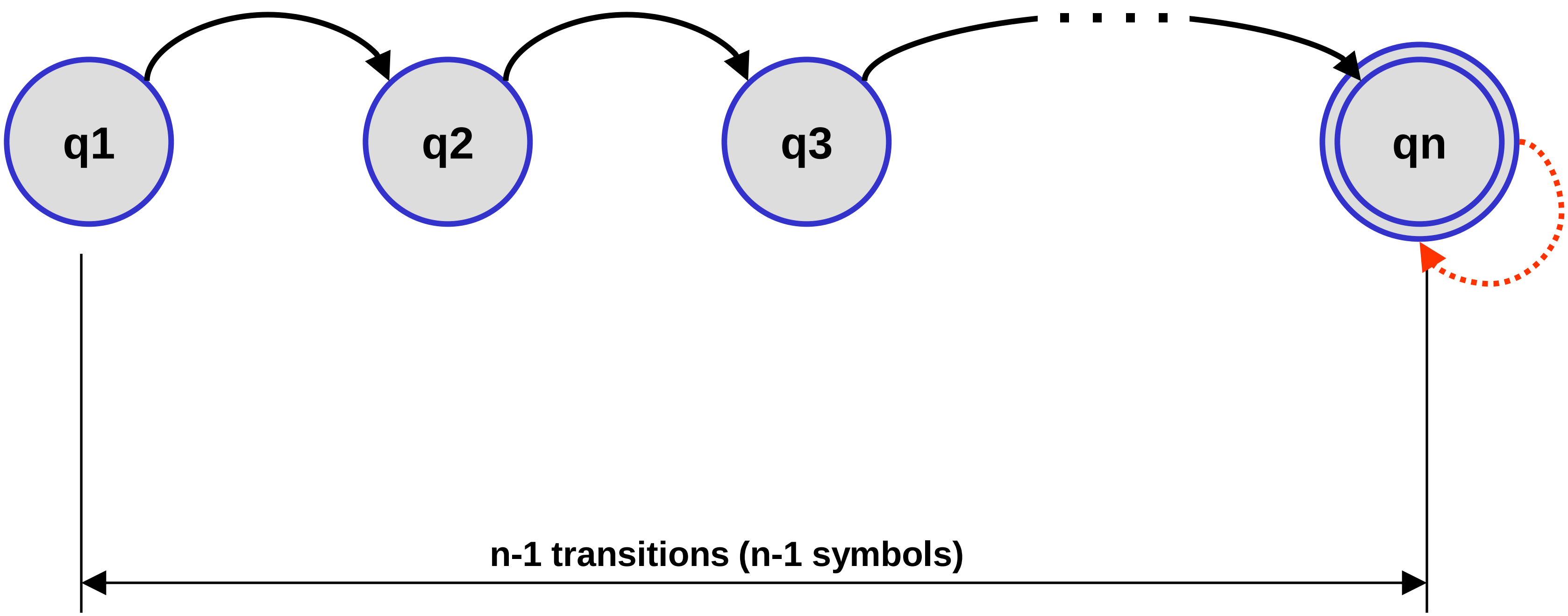
Decision algorithms

- **Infinity**
  - **A regular language *L*(*M*) is infinite if and only if DFA *M* accepts a string of length *l*, where $n \leq l < 2n$.**
    - We remove all non-accepting states for which there is no sequence of transitions into an accepting state
    - If the obtained state diagram of DFA *M'* contains at least one closed loop, then the regular language *L*(*M*) is infinite.



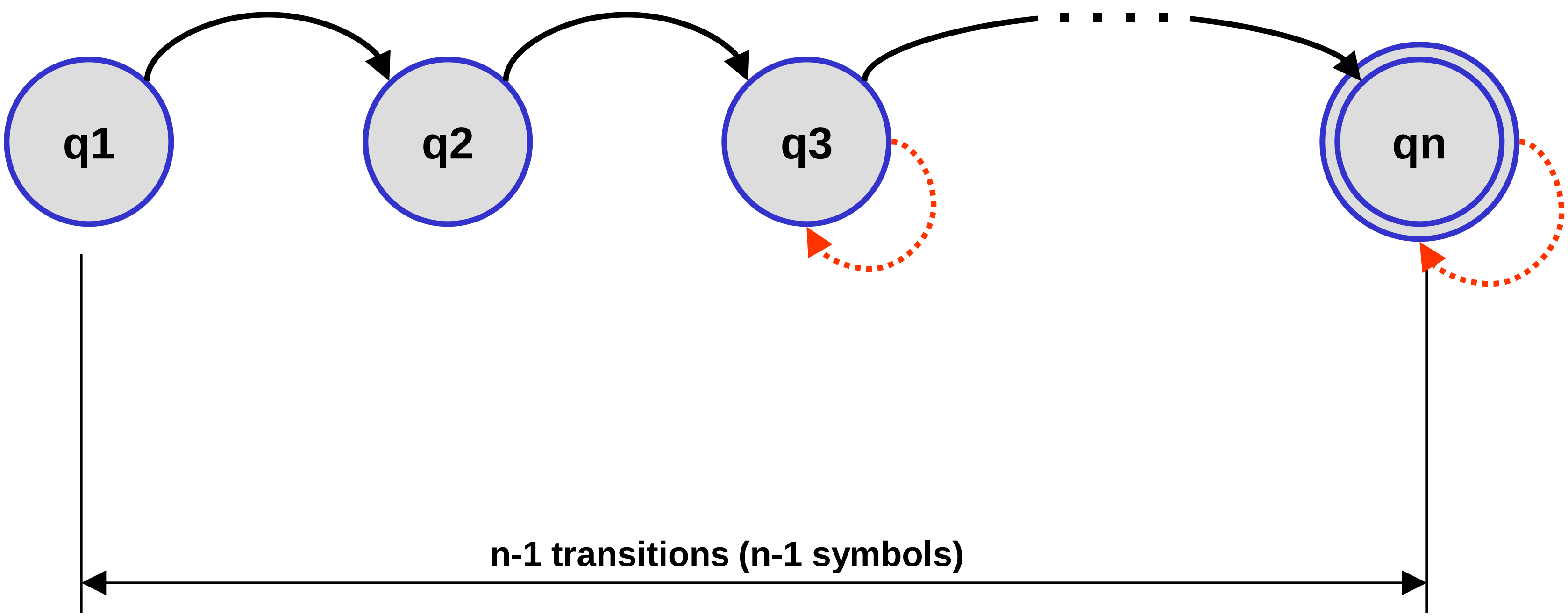
Decision algorithms

- Infinity
  - **A regular language *L*(*M*) is infinite if and only if DFA *M* accepts a string of length *l*, where $n \leq l < 2n$.**
    - We remove all non-accepting states for which there is no sequence of transitions into an accepting state
    - If the obtained state diagram of DFA *M'* contains at least one closed loop, then the regular language *L*(*M*) is infinite.



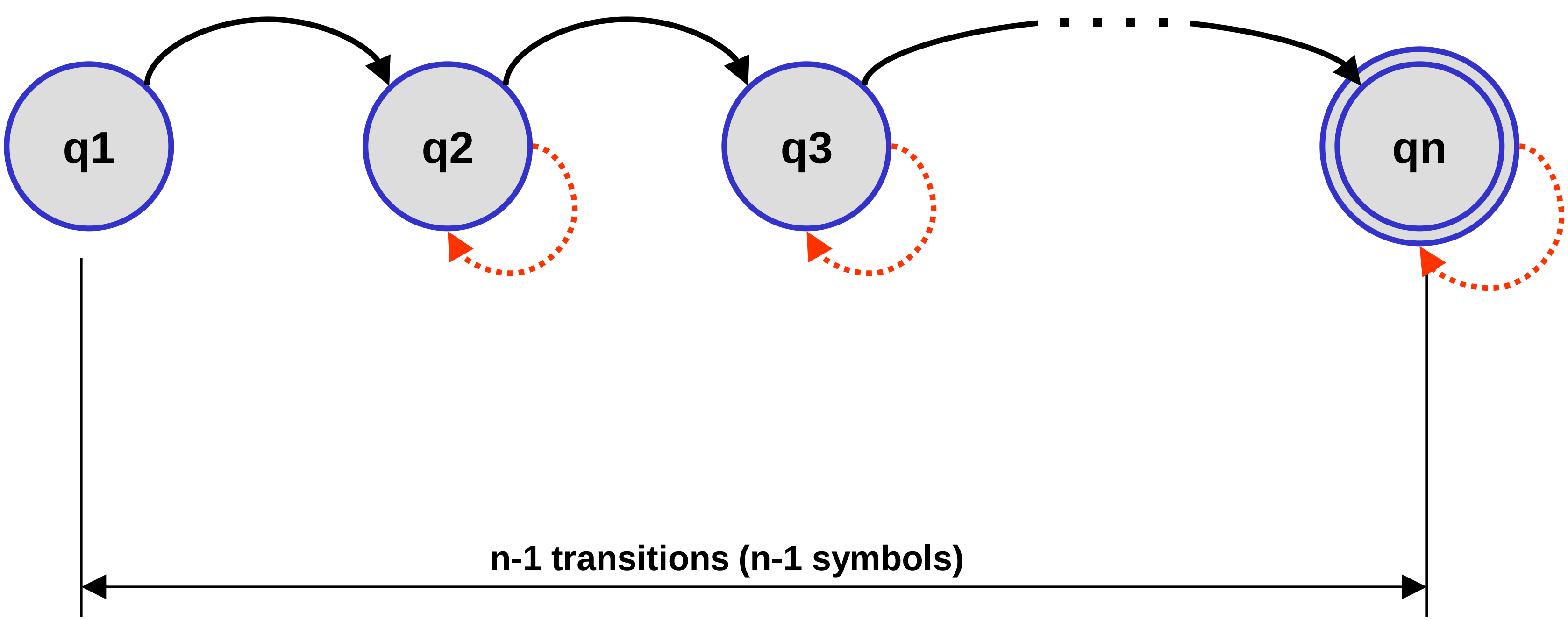
Decision algorithms

- **Infinity**
  - **A regular language *L*(*M*) is infinite if and only if DFA *M* accepts a string of length *l*, where $n \le l < 2n$.**
    - We remove all non-accepting states for which there is no sequence of transitions into an accepting state
    - If the obtained state diagram of DFA *M'* contains at least one closed loop, then the regular language *L*(*M*) is infinite.



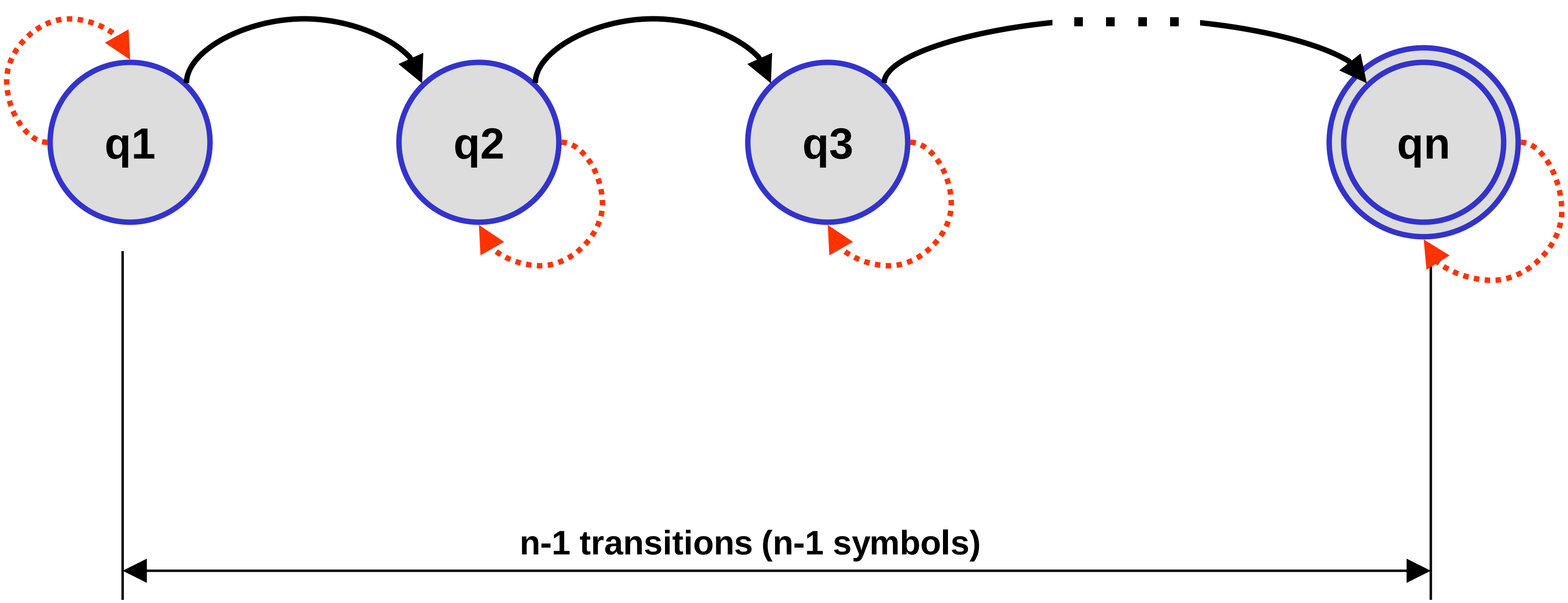
Decision algorithms

- Infinity
  - **A regular language *L*(*M*) is infinite if and only if DFA *M* accepts a string of length *l*, where $n \leq l < 2n$.**
    - We remove all non-accepting states for which there is no sequence of transitions into an accepting state
    - If the obtained state diagram of DFA *M'* contains at least one closed loop, then the regular language *L*(*M*) is infinite.



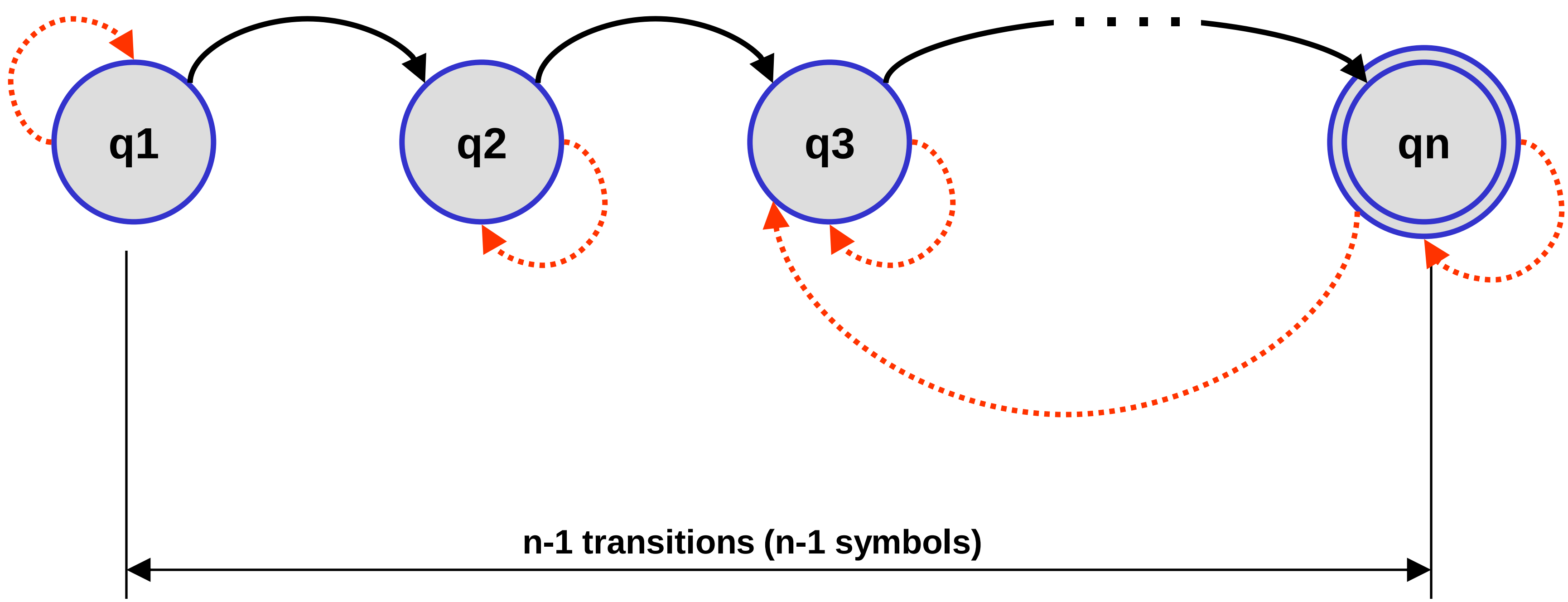
Decision algorithms

- **Infinity**
  - **A regular language *L*(*M*) is infinite if and only if DFA *M* accepts a string of length *l*, where $n \le l < 2n$.**
    - We remove all non-accepting states for which there is no sequence of transitions into an accepting state
    - If the obtained state diagram of DFA *M'* contains at least one closed loop, then the regular language *L*(*M*) is infinite.



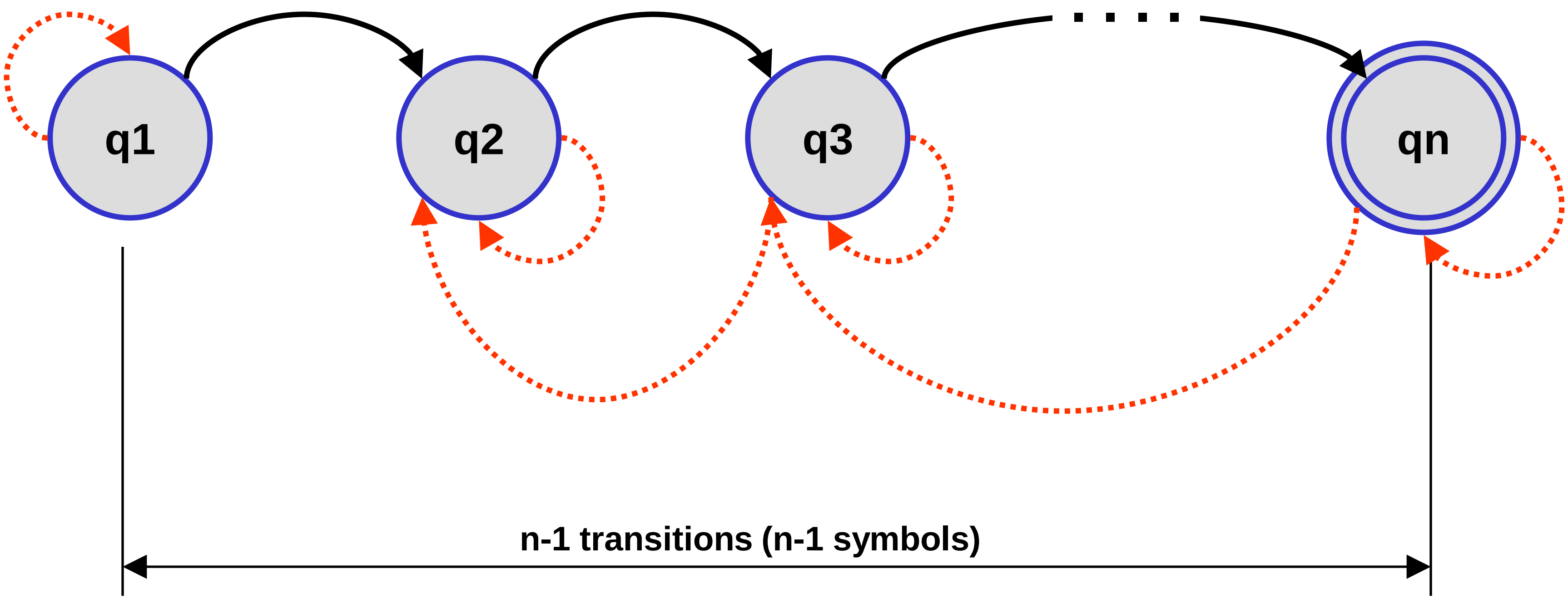
Decision algorithms

- **Infinity**
  - **A regular language *L*(*M*) is infinite if and only if DFA *M* accepts a string of length *l*, where $n \leq l < 2n$.**
    - We remove all non-accepting states for which there is no sequence of transitions into an accepting state
    - If the obtained state diagram of DFA *M'* contains at least one closed loop, then the regular language *L*(*M*) is infinite.



Decision algorithms

- Infinity
  - **A regular language *L*(*M*) is infinite if and only if DFA *M* accepts a string of length *l*, where $n \le l < 2n$.**
    - We remove all non-accepting states for which there is no sequence of transitions into an accepting state
    - If the obtained state diagram of DFA *M'* contains at least one closed loop, then the regular language *L*(*M*) is infinite.



Lecture overview

Grammars

Grammars

Elements of a sentence (variables)

Grammars

Elements of a sentence (variables)

<Sentence> , <SubjectSet>, <Verb>, <ObjectSet>, <Subject>, <Object>, <Attribute>

Grammars

Elements of a sentence (variables)

<Sentence> , <SubjectSet>, <Verb>, <ObjectSet>, <Subject>, <Object>, <Attribute>

Dictionary (terminals)

Grammars

Elements of a sentence (variables)

<Sentence> , <SubjectSet>, <Verb>, <ObjectSet>, <Subject>, <Object>, <Attribute>

Dictionary (terminals)

GIRLS, CATS, WATCH, CONFUSED, SCARED, .

Grammars

Grammars

Rules for building sentences

Grammars

Rules for building sentences

1) ***<Sentence>*** → ***<SubjectSet> <Verb> <ObjectSet>*** **.**

Grammars

Rules for building sentences

1) *<Sentence>* → *<SubjectSet> <Verb> <ObjectSet>* **.**

2) *<SubjectSet>* → *<Attribute> <Subject>*

Grammars

Rules for building sentences

1) *<Sentence>* $\rightarrow$ *<SubjectSet> <Verb> <ObjectSet>* **.**

2) *<SubjectSet>* $\rightarrow$ *<Attribute> <Subject>*

3) *<ObjectSet>* $\rightarrow$ *<Attribute> <Object>*

Grammars

Rules for building sentences

1)	***<Sentence>***	→	***<SubjectSet> <Verb> <ObjectSet>*** **.**
2)	***<SubjectSet>***	→	***<Attribute> <Subject>***
3)	***<ObjectSet>***	→	***<Attribute> <Object>***
4)	***<Verb>***	→	**WATCH**

Grammars

Rules for building sentences

1)	***<Sentence>***	$\rightarrow$ ***<SubjectSet> <Verb> <ObjectSet>*** **.**
2)	***<SubjectSet>***	$\rightarrow$ ***<Attribute> <Subject>***
3)	***<ObjectSet>***	$\rightarrow$ ***<Attribute> <Object>***
4)	***<Verb>***	$\rightarrow$ **WATCH**
5)	***<Subject>***	$\rightarrow$ **GIRLS**

Grammars

Rules for building sentences

1)	***<Sentence>***	$\rightarrow$	***<SubjectSet> <Verb> <ObjectSet>*** **.**
2)	***<SubjectSet>***	$\rightarrow$	***<Attribute> <Subject>***
3)	***<ObjectSet>***	$\rightarrow$	***<Attribute> <Object>***
4)	***<Verb>***	$\rightarrow$	**WATCH**
5)	***<Subject>***	$\rightarrow$	**GIRLS**
6)	***<Subject>***	$\rightarrow$	**CATS**

Grammars

Rules for building sentences

1)	***<Sentence>***	→ ***<SubjectSet> <Verb> <ObjectSet>*** **.**
2)	***<SubjectSet>***	→ ***<Attribute> <Subject>***
3)	***<ObjectSet>***	→ ***<Attribute> <Object>***
4)	***<Verb>***	→ **WATCH**
5)	***<Subject>***	→ **GIRLS**
6)	***<Subject>***	→ **CATS**
7)	***<Attribute>***	→ **CONFUSED**

Grammars

Rules for building sentences

1)	***<Sentence>***	→ ***<SubjectSet> <Verb> <ObjectSet>*** **.**
2)	***<SubjectSet>***	→ ***<Attribute> <Subject>***
3)	***<ObjectSet>***	→ ***<Attribute> <Object>***
4)	***<Verb>***	→ **WATCH**
5)	***<Subject>***	→ **GIRLS**
6)	***<Subject>***	→ **CATS**
7)	***<Attribute>***	→ **CONFUSED**
8)	***<Attribute>***	→ **SCARED**

Grammars

Rules for building sentences

1)	*<Sentence>*	→	*<SubjectSet> <Verb> <ObjectSet>* **.**
2)	*<SubjectSet>*	→	*<Attribute> <Subject>*
3)	*<ObjectSet>*	→	*<Attribute> <Object>*
4)	*<Verb>*	→	**WATCH**
5)	*<Subject>*	→	**GIRLS**
6)	*<Subject>*	→	**CATS**
7)	*<Attribute>*	→	**CONFUSED**
8)	*<Attribute>*	→	**SCARED**
9)	*<Object>*	→	**GIRLS**

Grammars

Rules for building sentences

1)	***<Sentence>***	→ ***<SubjectSet> <Verb> <ObjectSet>*** **.**
2)	***<SubjectSet>***	→ ***<Attribute> <Subject>***
3)	***<ObjectSet>***	→ ***<Attribute> <Object>***
4)	***<Verb>***	→ **WATCH**
5)	***<Subject>***	→ **GIRLS**
6)	***<Subject>***	→ **CATS**
7)	***<Attribute>***	→ **CONFUSED**
8)	***<Attribute>***	→ **SCARED**
9)	***<Object>***	→ **GIRLS**
10)	***<Object>***	→ **CATS**

Grammars

1) <Sentence> → <SubjectSet> <Verb> <ObjectSet> .
2) <SubjectSet> → <Attribute> <Subject>
3) <ObjectSet> → <Attribute> <Object>
4) <Verb> → WATCH
5) <Subject> → GIRLS
6) <Subject> → CATS
7) <Attribute> → CONFUSED
8) <Attribute> → SCARED
9) <Object> → GIRLS
10) <Object> → CATS

Grammars

1) *<Sentence>* → *<SubjectSet> <Verb> <ObjectSet>* **.**
2) *<SubjectSet>* → *<Attribute> <Subject>*
3) *<ObjectSet>* → *<Attribute> <Object>*
4) *<Verb>* → **WATCH**
5) *<Subject>* → **GIRLS**
6) *<Subject>* → **CATS**
7) *<Attribute>* → **CONFUSED**
8) *<Attribute>* → **SCARED**
9) *<Object>* → **GIRLS**
10) *<Object>* → **CATS**

<Sentence>

Grammars

1) *<Sentence>* → *<SubjectSet> <Verb> <ObjectSet>* .
2) *<SubjectSet>* → *<Attribute> <Subject>*
3) *<ObjectSet>* → *<Attribute> <Object>*
4) *<Verb>* → **WATCH**
5) *<Subject>* → **GIRLS**
6) *<Subject>* → **CATS**
7) *<Attribute>* → **CONFUSED**
8) *<Attribute>* → **SCARED**
9) *<Object>* → **GIRLS**
10) *<Object>* → **CATS**

<Sentence>

Grammars

1) *<Sentence>* → *<SubjectSet> <Verb> <ObjectSet>* .
2) *<SubjectSet>* → *<Attribute> <Subject>*
3) *<ObjectSet>* → *<Attribute> <Object>*
4) *<Verb>* → WATCH
5) *<Subject>* → GIRLS
6) *<Subject>* → CATS
7) *<Attribute>* → CONFUSED
8) *<Attribute>* → SCARED
9) *<Object>* → GIRLS
10) *<Object>* → CATS

<Sentence>

⇒ *<SubjectSet>* *<Verb>* *<ObjectSet>* .

Grammars

1) *<Sentence>* → *<SubjectSet> <Verb> <ObjectSet>* **.**
2) *<SubjectSet>* → *<Attribute> <Subject>*
3) *<ObjectSet>* → *<Attribute> <Object>*
4) *<Verb>* → **WATCH**
5) *<Subject>* → **GIRLS**
6) *<Subject>* → **CATS**
7) *<Attribute>* → **CONFUSED**
8) *<Attribute>* → **SCARED**
9) *<Object>* → **GIRLS**
10) *<Object>* → **CATS**

<Sentence>

⇒ *<SubjectSet>* *<Verb>* *<ObjectSet>* **.**

Grammars

1) *<Sentence>* → *<SubjectSet> <Verb> <ObjectSet>* **.**
2) *<SubjectSet>* → *<Attribute> <Subject>*
3) *<ObjectSet>* → *<Attribute> <Object>*
4) *<Verb>* → **WATCH**
5) *<Subject>* → **GIRLS**
6) *<Subject>* → **CATS**
7) *<Attribute>* → **CONFUSED**
8) *<Attribute>* → **SCARED**
9) *<Object>* → **GIRLS**
10) *<Object>* → **CATS**

<Sentence>

⇒ *<SubjectSet>* *<Verb>* *<ObjectSet>* **.**

⇒ *<Attribute>* *<Subject>*

Grammars

1) *<Sentence>* → *<SubjectSet> <Verb> <ObjectSet>* .
2) *<SubjectSet>* → *<Attribute> <Subject>*
3) *<ObjectSet>* → *<Attribute> <Object>*
4) *<Verb>* → WATCH
5) *<Subject>* → GIRLS
6) *<Subject>* → CATS
7) *<Attribute>* → CONFUSED
8) *<Attribute>* → SCARED
9) *<Object>* → GIRLS
10) *<Object>* → CATS

<Sentence>

⇒ *<SubjectSet> <Verb> <ObjectSet>* .

⇒ *<Attribute> <Subject> <Verb> <ObjectSet>* .

Grammars

1) <Sentence> → <SubjectSet> <Verb> <ObjectSet> .
2) <SubjectSet> → <Attribute> <Subject>
3) <ObjectSet> → <Attribute> <Object>
4) <Verb> → WATCH
5) <Subject> → GIRLS
6) <Subject> → CATS
7) <Attribute> → CONFUSED
8) <Attribute> → SCARED
9) <Object> → GIRLS
10) <Object> → CATS

<Sentence>

⇒ <SubjectSet> <Verb> <ObjectSet> .

⇒ <Attribute> <Subject> <Verb> <ObjectSet> .

Grammars

1) *<Sentence>* → *<SubjectSet> <Verb> <ObjectSet>* .
2) *<SubjectSet>* → *<Attribute> <Subject>*
3) *<ObjectSet>* → *<Attribute> <Object>*
4) *<Verb>* → WATCH
5) *<Subject>* → GIRLS
6) *<Subject>* → CATS
7) *<Attribute>* → CONFUSED
8) *<Attribute>* → SCARED
9) *<Object>* → GIRLS
10) *<Object>* → CATS

<Sentence>

⇒ *<SubjectSet>* *<Verb>* *<ObjectSet>* .

⇒ *<Attribute>* *<Subject>* *<Verb>* *<ObjectSet>* .

⇒ *<Attribute>* *<Object>*.

Grammars

1) *<Sentence>* → *<SubjectSet> <Verb> <ObjectSet>* .
2) *<SubjectSet>* → *<Attribute> <Subject>*
3) *<ObjectSet>* → *<Attribute> <Object>*
4) *<Verb>* → WATCH
5) *<Subject>* → GIRLS
6) *<Subject>* → CATS
7) *<Attribute>* → CONFUSED
8) *<Attribute>* → SCARED
9) *<Object>* → GIRLS
10) *<Object>* → CATS

<Sentence>

⇒ *<SubjectSet> <Verb> <ObjectSet>* .

⇒ *<Attribute> <Subject> <Verb> <ObjectSet>* .

⇒ *<Attribute> <Subject> <Verb> <Attribute> <Object>*.

Grammars

1) ***<Sentence>*** → ***<SubjectSet> <Verb> <ObjectSet>*** **.**
2) ***<SubjectSet>*** → ***<Attribute> <Subject>***
3) ***<ObjectSet>*** → ***<Attribute> <Object>***
4) ***<Verb>*** → **WATCH**
5) ***<Subject>*** → **GIRLS**
6) ***<Subject>*** → **CATS**
7) ***<Attribute>*** → **CONFUSED**
8) ***<Attribute>*** → **SCARED**
9) ***<Object>*** → **GIRLS**
10) ***<Object>*** → **CATS**

<Sentence>

⇒ ***<SubjectSet> <Verb> <ObjectSet>*** **.**

⇒ ***<Attribute> <Subject> <Verb> <ObjectSet>*** **.**

⇒ ***<Attribute> <Subject> <Verb> <Attribute> <Object>*****.**

Grammars

1) *<Sentence>* → *<SubjectSet> <Verb> <ObjectSet>* .
2) *<SubjectSet>* → *<Attribute> <Subject>*
3) *<ObjectSet>* → *<Attribute> <Object>*
4) *<Verb>* → WATCH
5) *<Subject>* → GIRLS
6) *<Subject>* → CATS
7) *<Attribute>* → CONFUSED
8) *<Attribute>* → SCARED
9) *<Object>* → GIRLS
10) *<Object>* → CATS

<Sentence>
⇒ *<SubjectSet> <Verb> <ObjectSet>* .
⇒ *<Attribute> <Subject> <Verb> <ObjectSet>* .
⇒ *<Attribute> <Subject> <Verb> <Attribute> <Object>*.
⇒ *<Attribute> <Subject>* WATCH *<Attribute> <Object>*.

Grammars

1)	*<Sentence>*	→ *<SubjectSet> <Verb> <ObjectSet>* .
2)	*<SubjectSet>*	→ *<Attribute> <Subject>*
3)	*<ObjectSet>*	→ *<Attribute> <Object>*
4)	*<Verb>*	→ WATCH
5)	*<Subject>*	→ GIRLS
6)	*<Subject>*	→ CATS
7)	*<Attribute>*	→ CONFUSED
8)	*<Attribute>*	→ SCARED
9)	*<Object>*	→ GIRLS
10)	*<Object>*	→ CATS

<Sentence>

⇒ *<SubjectSet> <Verb> <ObjectSet>* .

⇒ *<Attribute> <Subject> <Verb> <ObjectSet>* .

⇒ *<Attribute> <Subject> <Verb> <Attribute> <Object>*.

⇒ *<Attribute> <Subject>* WATCH *<Attribute> <Object>*.

Grammars

1) *<Sentence>* → *<SubjectSet> <Verb> <ObjectSet>* .
2) *<SubjectSet>* → *<Attribute> <Subject>*
3) *<ObjectSet>* → *<Attribute> <Object>*
4) *<Verb>* → WATCH
5) *<Subject>* → GIRLS
6) *<Subject>* → CATS
7) *<Attribute>* → CONFUSED
8) *<Attribute>* → SCARED
9) *<Object>* → GIRLS
10) *<Object>* → CATS

<Sentence>

⇒ *<SubjectSet> <Verb> <ObjectSet>* .
⇒ *<Attribute> <Subject> <Verb> <ObjectSet>* .
⇒ *<Attribute> <Subject> <Verb> <Attribute> <Object>*.
⇒ *<Attribute> <Subject>* WATCH *<Attribute> <Object>*.
⇒ *<Attribute>* GIRLS WATCH *<Attribute> <Object>*.

Grammars

1) *<Sentence>* → *<SubjectSet> <Verb> <ObjectSet>* **.**
2) *<SubjectSet>* → *<Attribute> <Subject>*
3) *<ObjectSet>* → *<Attribute> <Object>*
4) *<Verb>* → **WATCH**
5) *<Subject>* → **GIRLS**
6) *<Subject>* → **CATS**
7) *<Attribute>* → **CONFUSED**
8) *<Attribute>* → **SCARED**
9) *<Object>* → **GIRLS**
10) *<Object>* → **CATS**

<Sentence>
⇒ *<SubjectSet>* *<Verb>* *<ObjectSet>* **.**
⇒ *<Attribute>* *<Subject>* *<Verb>* *<ObjectSet>* **.**
⇒ *<Attribute>* *<Subject>* *<Verb>* *<Attribute>* *<Object>***.**
⇒ *<Attribute>* *<Subject>* **WATCH** *<Attribute>* *<Object>***.**
⇒ *<Attribute>* **GIRLS WATCH** *<Attribute>* *<Object>***.**
⇒ *<Attribute>* **GIRLS WATCH** *<Attribute>* **CATS.**
⇒ **CONFUSED GIRLS WATCH** *<Attribute>* **CATS.**
⇒ **CONFUSED GIRLS WATCH SCARED CATS.**

Grammars

1) *<Sentence>* → *<SubjectSet> <Verb> <ObjectSet>* **.**
2) *<SubjectSet>* → *<Attribute> <Subject>*
3) *<ObjectSet>* → *<Attribute> <Object>*
4) *<Verb>* → **WATCH**
5) *<Subject>* → **GIRLS**
6) *<Subject>* → **CATS**
7) *<Attribute>* → **CONFUSED**
8) *<Attribute>* → **SCARED**
9) *<Object>* → **GIRLS**
10) *<Object>* → **CATS**

Grammars

1) *<Sentence>* → *<SubjectSet> <Verb> <ObjectSet>* **.**
2) *<SubjectSet>* → *<Attribute> <Subject>*
3) *<ObjectSet>* → *<Attribute> <Object>*
4) *<Verb>* → **WATCH**
5) *<Subject>* → **GIRLS**
6) *<Subject>* → **CATS**
7) *<Attribute>* → **CONFUSED**
8) *<Attribute>* → **SCARED**
9) *<Object>* → **GIRLS**
10) *<Object>* → **CATS**

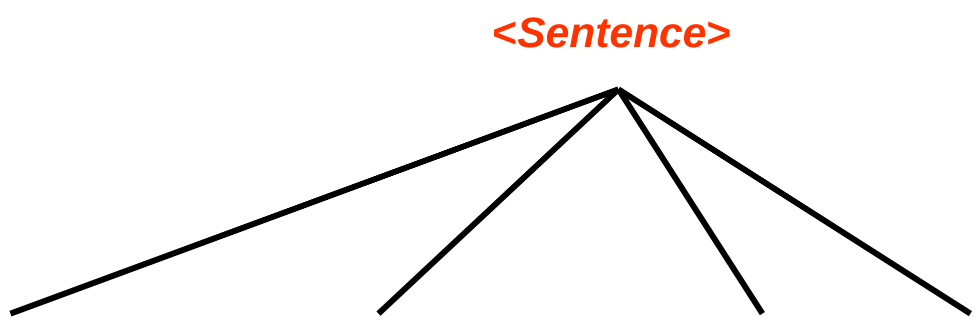
Grammars

1) *<Sentence>* → *<SubjectSet> <Verb> <ObjectSet>* **.**
2) *<SubjectSet>* → *<Attribute> <Subject>*
3) *<ObjectSet>* → *<Attribute> <Object>*
4) *<Verb>* → **WATCH**
5) *<Subject>* → **GIRLS**
6) *<Subject>* → **CATS**
7) *<Attribute>* → **CONFUSED**
8) *<Attribute>* → **SCARED**
9) *<Object>* → **GIRLS**
10) *<Object>* → **CATS**

<Sentence>

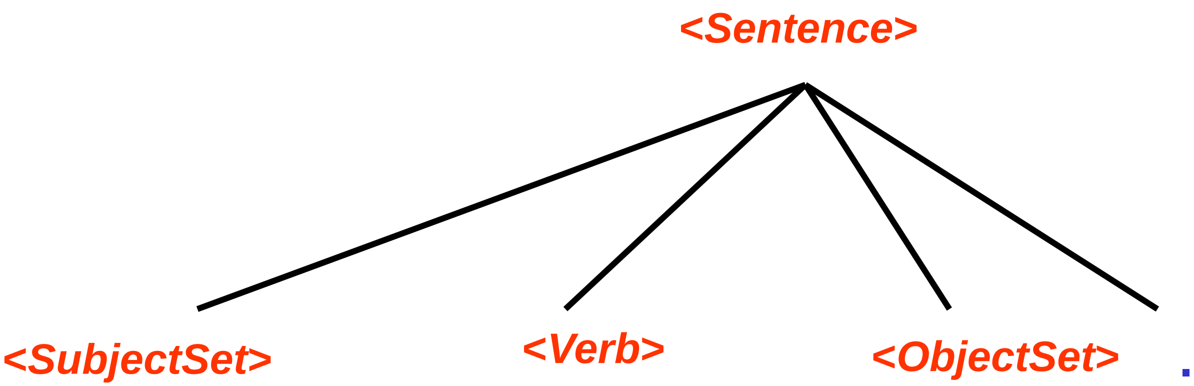
Grammars

1) *<Sentence>* → *<SubjectSet> <Verb> <ObjectSet>* **.**
2) *<SubjectSet>* → *<Attribute> <Subject>*
3) *<ObjectSet>* → *<Attribute> <Object>*
4) *<Verb>* → **WATCH**
5) *<Subject>* → **GIRLS**
6) *<Subject>* → **CATS**
7) *<Attribute>* → **CONFUSED**
8) *<Attribute>* → **SCARED**
9) *<Object>* → **GIRLS**
10) *<Object>* → **CATS**



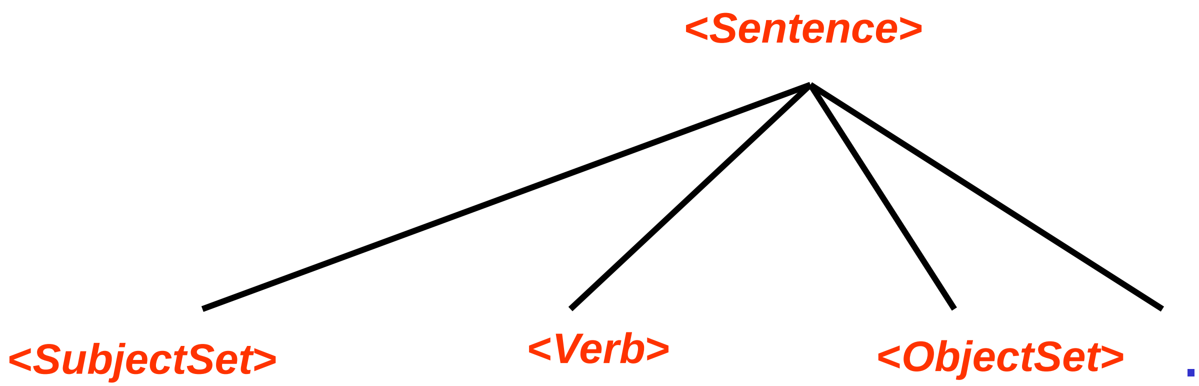
Grammars

1) *<Sentence>* → *<SubjectSet> <Verb> <ObjectSet>* **.**
2) *<SubjectSet>* → *<Attribute> <Subject>*
3) *<ObjectSet>* → *<Attribute> <Object>*
4) *<Verb>* → **WATCH**
5) *<Subject>* → **GIRLS**
6) *<Subject>* → **CATS**
7) *<Attribute>* → **CONFUSED**
8) *<Attribute>* → **SCARED**
9) *<Object>* → **GIRLS**
10) *<Object>* → **CATS**



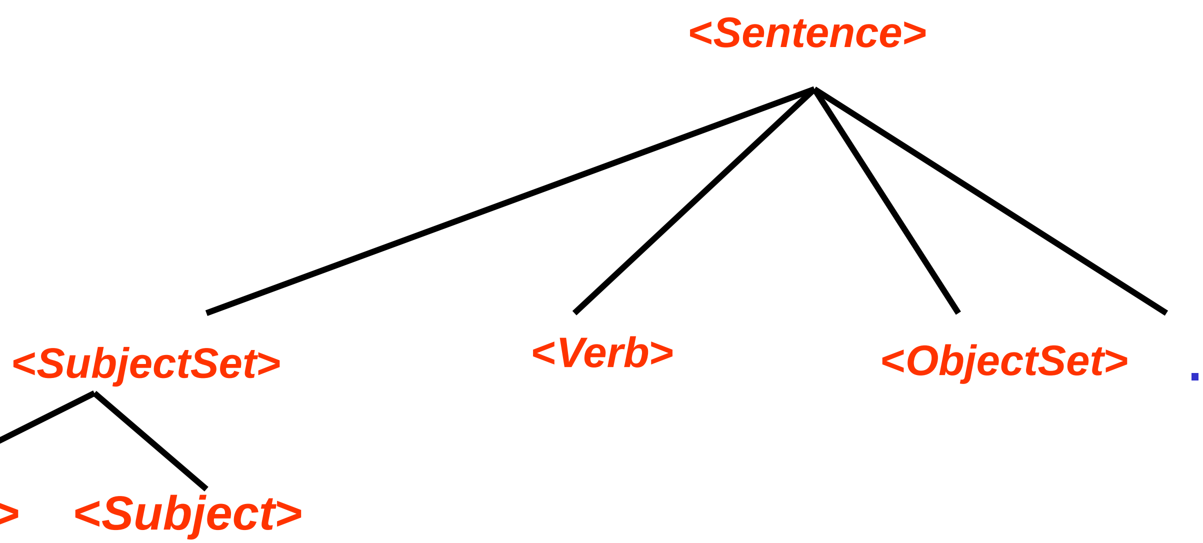
Grammars

1) *<Sentence>* → *<SubjectSet> <Verb> <ObjectSet>* **.**
2) *<SubjectSet>* → *<Attribute> <Subject>*
3) *<ObjectSet>* → *<Attribute> <Object>*
4) *<Verb>* → **WATCH**
5) *<Subject>* → **GIRLS**
6) *<Subject>* → **CATS**
7) *<Attribute>* → **CONFUSED**
8) *<Attribute>* → **SCARED**
9) *<Object>* → **GIRLS**
10) *<Object>* → **CATS**



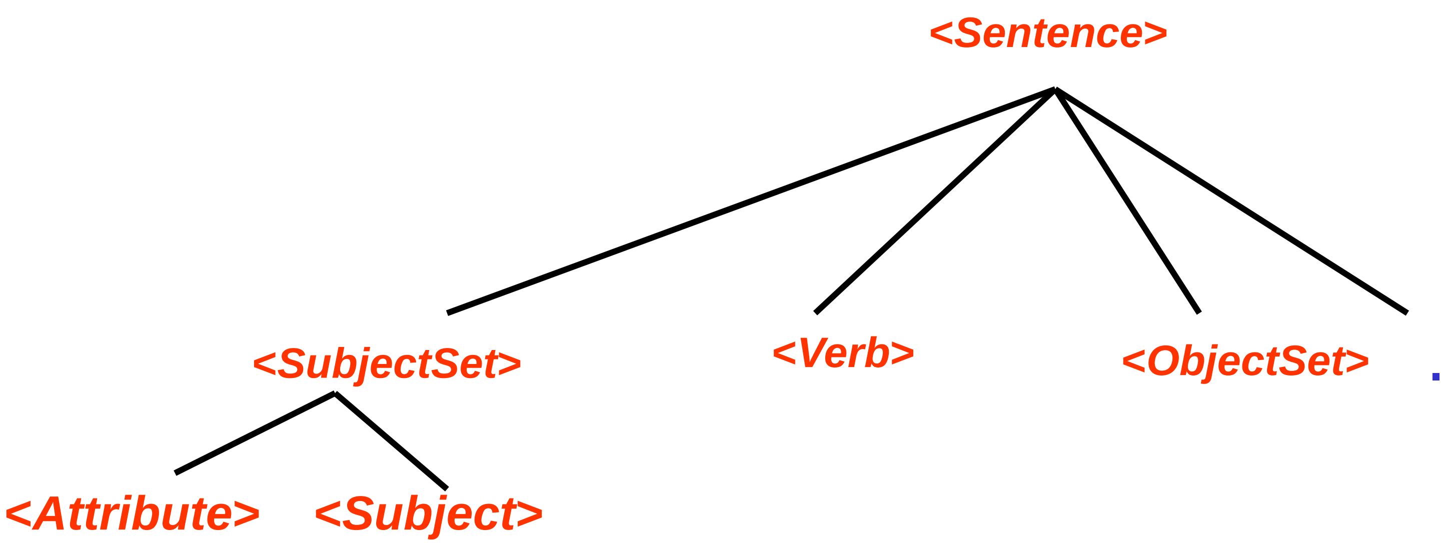
Grammars

1) <Sentence> → <SubjectSet> <Verb> <ObjectSet> .
2) <SubjectSet> → <Attribute> <Subject>
3) <ObjectSet> → <Attribute> <Object>
4) <Verb> → WATCH
5) <Subject> → GIRLS
6) <Subject> → CATS
7) <Attribute> → CONFUSED
8) <Attribute> → SCARED
9) <Object> → GIRLS
10) <Object> → CATS



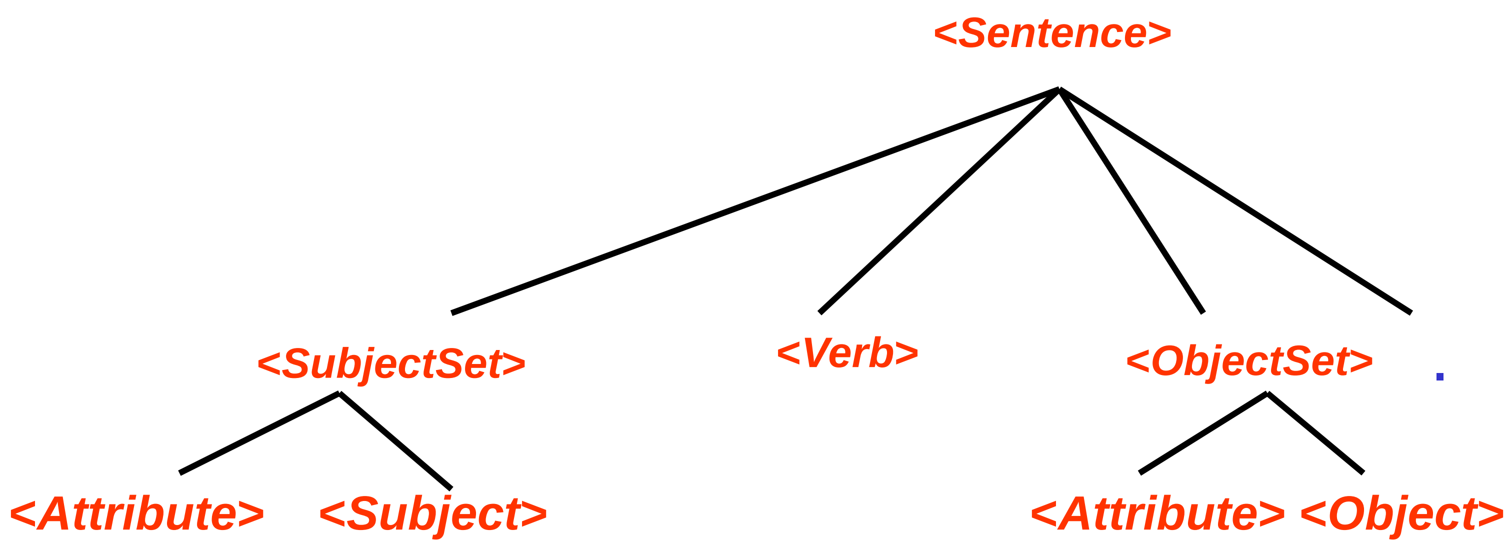
Grammars

1) *<Sentence>* → *<SubjectSet> <Verb> <ObjectSet>* **.**
2) *<SubjectSet>* → *<Attribute> <Subject>*
3) *<ObjectSet>* → *<Attribute> <Object>*
4) *<Verb>* → **WATCH**
5) *<Subject>* → **GIRLS**
6) *<Subject>* → **CATS**
7) *<Attribute>* → **CONFUSED**
8) *<Attribute>* → **SCARED**
9) *<Object>* → **GIRLS**
10) *<Object>* → **CATS**



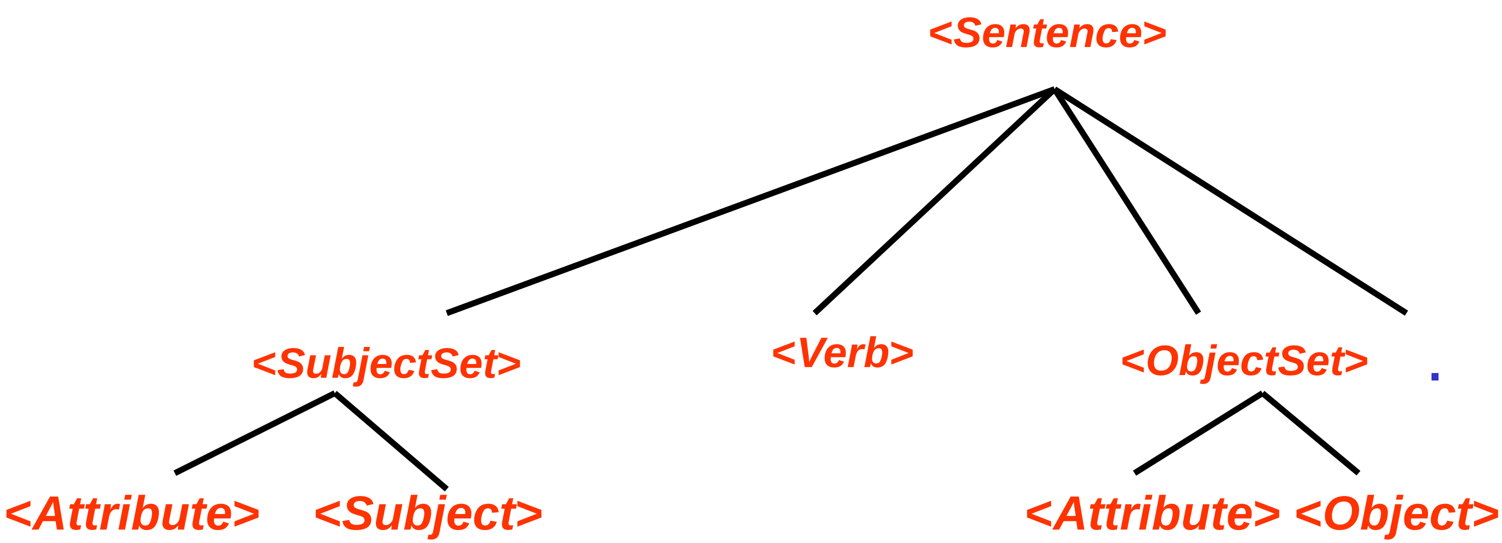
Grammars

1) *<Sentence>* → *<SubjectSet> <Verb> <ObjectSet>* **.**
2) *<SubjectSet>* → *<Attribute> <Subject>*
3) *<ObjectSet>* → *<Attribute> <Object>*
4) *<Verb>* → **WATCH**
5) *<Subject>* → **GIRLS**
6) *<Subject>* → **CATS**
7) *<Attribute>* → **CONFUSED**
8) *<Attribute>* → **SCARED**
9) *<Object>* → **GIRLS**
10) *<Object>* → **CATS**



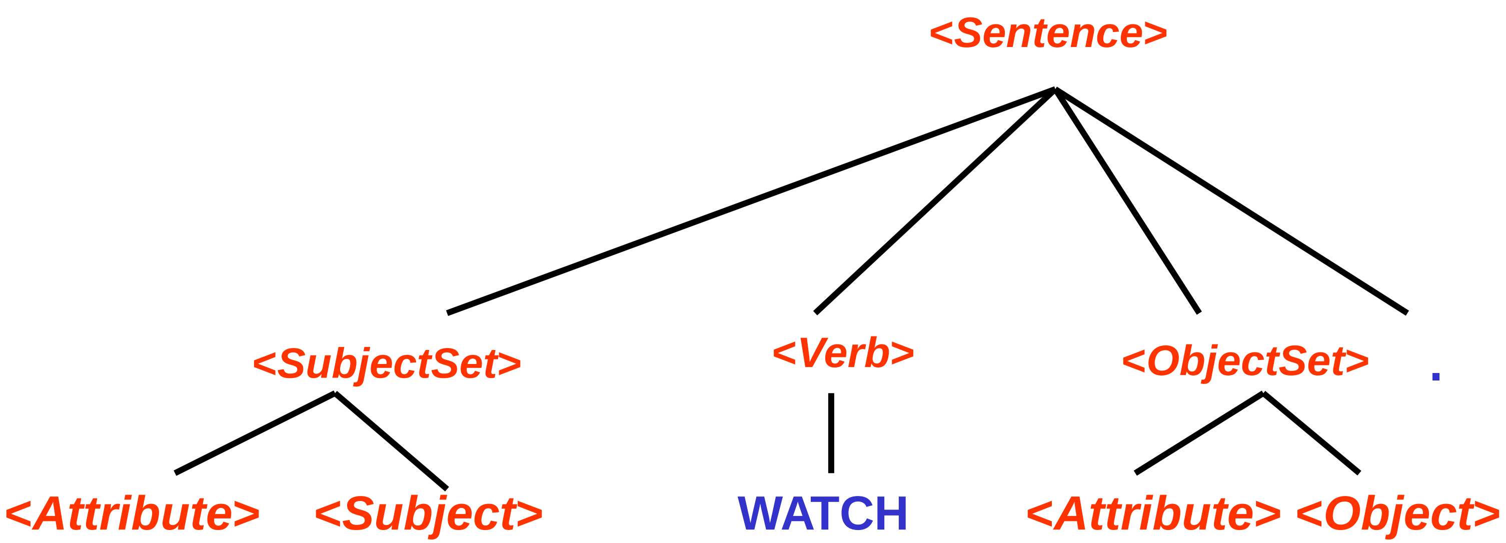
Grammars

1) *<Sentence>* → *<SubjectSet> <Verb> <ObjectSet>* **.**
2) *<SubjectSet>* → *<Attribute> <Subject>*
3) *<ObjectSet>* → *<Attribute> <Object>*
4) *<Verb>* → **WATCH**
5) *<Subject>* → **GIRLS**
6) *<Subject>* → **CATS**
7) *<Attribute>* → **CONFUSED**
8) *<Attribute>* → **SCARED**
9) *<Object>* → **GIRLS**
10) *<Object>* → **CATS**



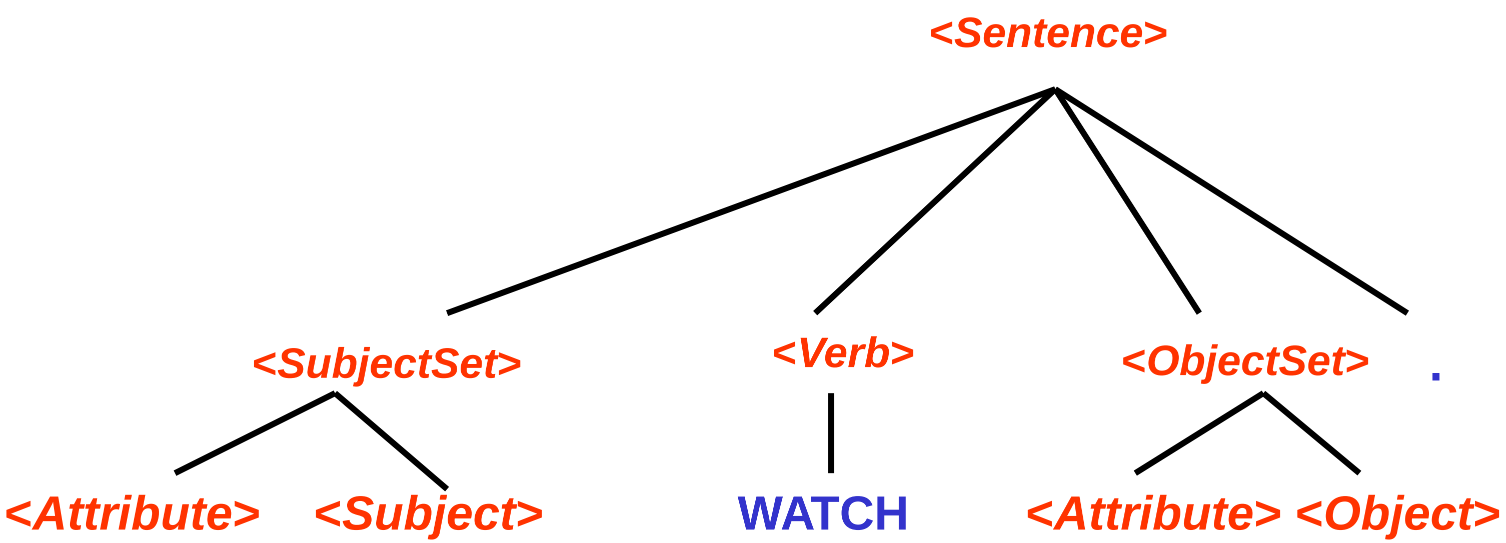
Grammars

1) <Sentence> → <SubjectSet> <Verb> <ObjectSet> .
2) <SubjectSet> → <Attribute> <Subject>
3) <ObjectSet> → <Attribute> <Object>
4) <Verb> → WATCH
5) <Subject> → GIRLS
6) <Subject> → CATS
7) <Attribute> → CONFUSED
8) <Attribute> → SCARED
9) <Object> → GIRLS
10) <Object> → CATS



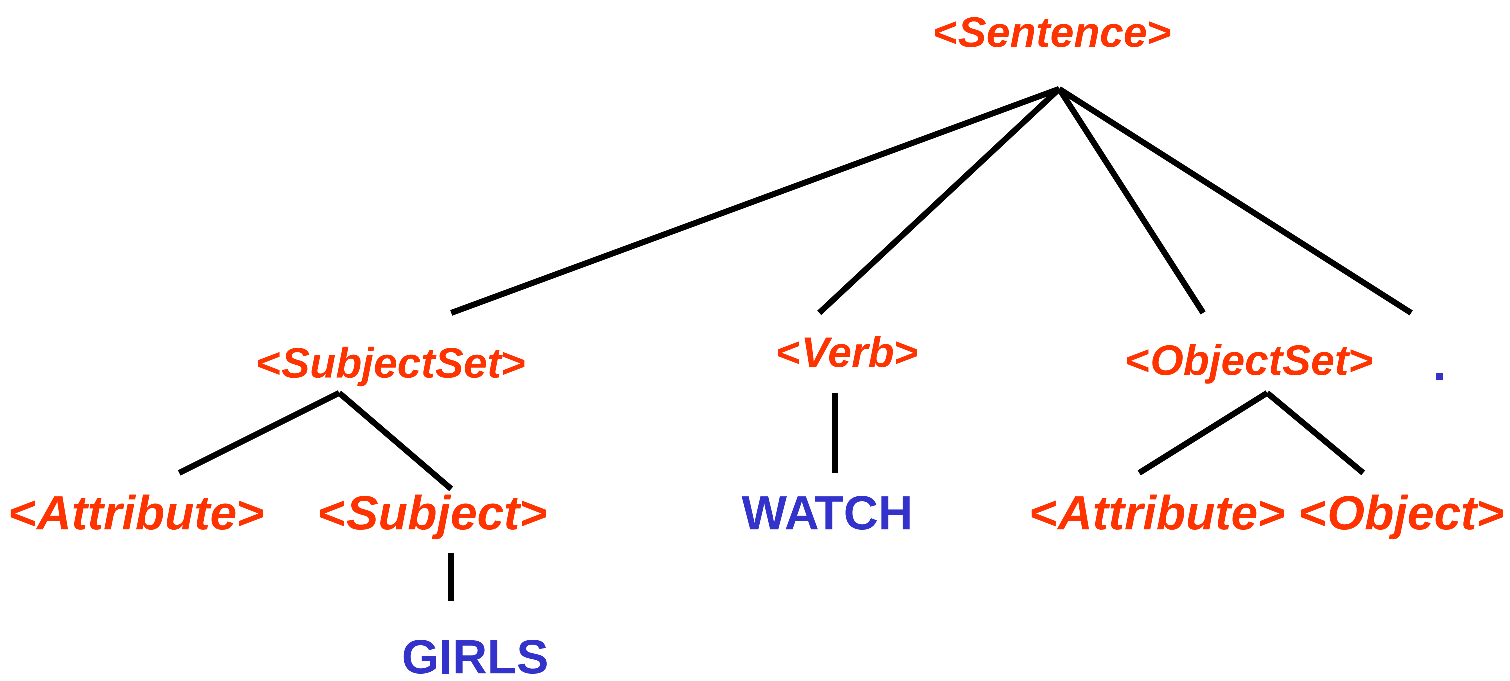
Grammars

1) <Sentence> → <SubjectSet> <Verb> <ObjectSet> .
2) <SubjectSet> → <Attribute> <Subject>
3) <ObjectSet> → <Attribute> <Object>
4) <Verb> → WATCH
5) <Subject> → GIRLS
6) <Subject> → CATS
7) <Attribute> → CONFUSED
8) <Attribute> → SCARED
9) <Object> → GIRLS
10) <Object> → CATS



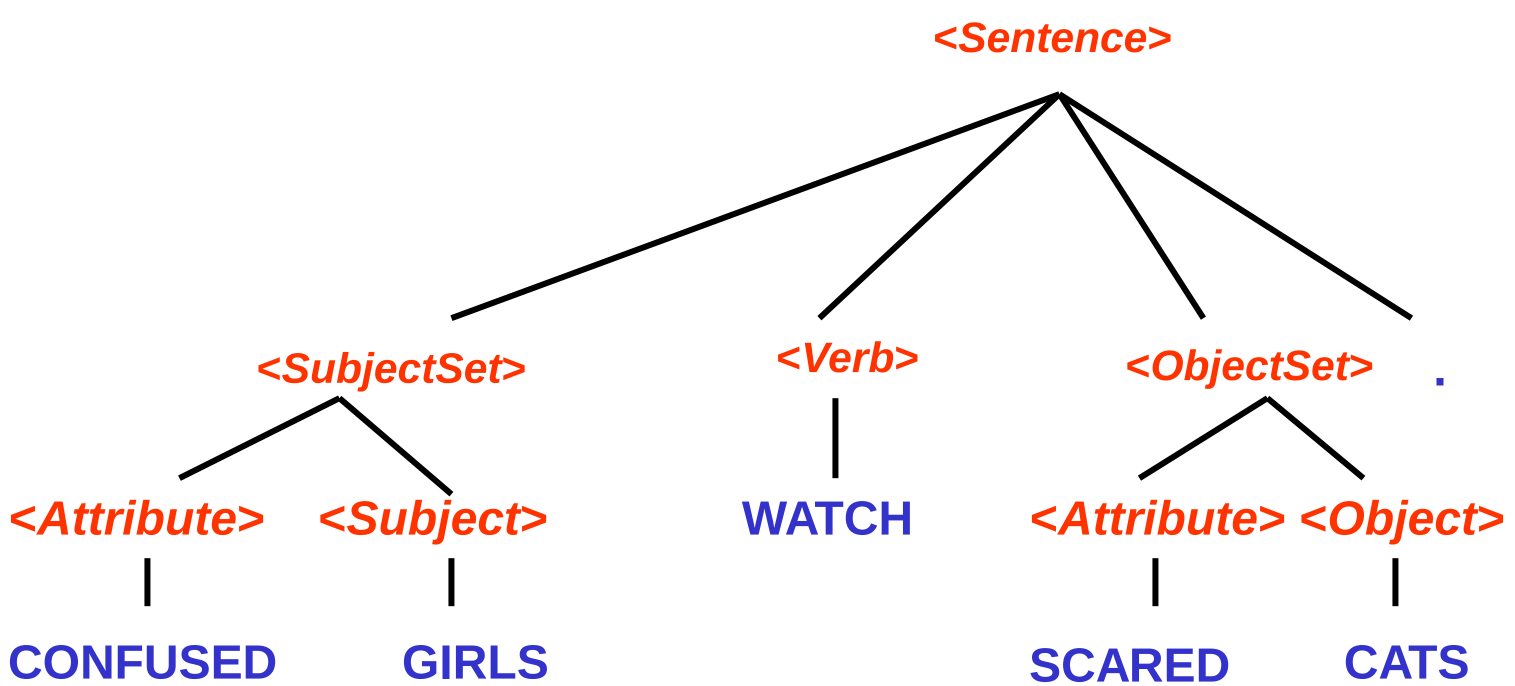
Grammars

1) *<Sentence>* → *<SubjectSet> <Verb> <ObjectSet>* .
2) *<SubjectSet>* → *<Attribute> <Subject>*
3) *<ObjectSet>* → *<Attribute> <Object>*
4) *<Verb>* → WATCH
5) *<Subject>* → GIRLS
6) *<Subject>* → CATS
7) *<Attribute>* → CONFUSED
8) *<Attribute>* → SCARED
9) *<Object>* → GIRLS
10) *<Object>* → CATS



Grammars

1) *<Sentence>* → *<SubjectSet> <Verb> <ObjectSet>* **.**
2) *<SubjectSet>* → *<Attribute> <Subject>*
3) *<ObjectSet>* → *<Attribute> <Object>*
4) *<Verb>* → **WATCH**
5) *<Subject>* → **GIRLS**
6) *<Subject>* → **CATS**
7) *<Attribute>* → **CONFUSED**
8) *<Attribute>* → **SCARED**
9) *<Object>* → **GIRLS**
10) *<Object>* → **CATS**



Grammars

Grammars

$$G = (V, T, P, S)$$

Grammars

$$G = (V, T, P, S)$$

V - finite set of variables

Grammars

$$G = (V, T, P, S)$$

V - finite set of variables

T - finite set of terminals

Grammars

$$G = (V, T, P, S)$$

V - finite set of variables

T - finite set of terminals

$$V \cap T = \varnothing$$

Grammars

$$G = (V, T, P, S)$$

V - finite set of variables

T - finite set of terminals

$$V \cap T = \varnothing$$

P - finite set of productions of the form

Grammars

$$G = (V, T, P, S)$$

V	- finite set of variables
T	- finite set of terminals
	$V \cap T = \emptyset$
P	- finite set of productions of the form
	$A \to \alpha$

Grammars

$$G = (V, T, P, S)$$

V - finite set of variables

T - finite set of terminals

$$V \cap T = \emptyset$$

P - finite set of productions of the form

$$A \rightarrow \alpha$$

A is a variable

Grammars

$$G = (V, T, P, S)$$

V - finite set of variables

T - finite set of terminals

$$V \cap T = \emptyset$$

P - finite set of productions of the form

$$A \rightarrow \alpha$$

A is a variable

α is a string in $(V \cup T)^*$ or ε

Grammars

$$G = (V, T, P, S)$$

V - finite set of variables

T - finite set of terminals

$$V \cap T = \emptyset$$

P - finite set of productions of the form

$$A \to \alpha$$

A is a variable

α is a string in $(V \cup T)^*$ or ε

S - start symbol (a variable)

Grammars

Grammars

$G = (\{ E \},$

Grammars

G = ({ E },
{ a, $*$, $+$, (,) },

Grammars

$$G = (\ \{\ E\ \},$$
$$\{\ a,\ *,\ +,\ (,\ )\ \},$$
$$\{\ E \to E{+}E \mid E{*}E \mid (E) \mid a\ \},$$

Grammars

$$G = (\{ E \},$$
$$\{ a, *, +, (,) \},$$
$$\{ E \rightarrow E{+}E \mid E{*}E \mid (E) \mid a \},$$
$$E)$$

Grammars

$G = (\{ E \},$
$\{ a, *, +, (,) \},$
$\{ E \rightarrow E+E \mid E*E \mid (E) \mid a \},$
$E)$

$E \Rightarrow a$

Grammars

$$G = (\ \{ E \},$$
$$\{ a, *, +, (,) \},$$
$$\{ E \rightarrow E+E \mid E*E \mid (E) \mid a \},$$
$$E\ )$$

$$E \Rightarrow a$$

$$E \Rightarrow E + E \Rightarrow E + a \Rightarrow a + a$$

Grammars

$$G = (\{ E \}, \\ \{ a, *, +, (,) \}, \\ \{ E \to E+E \mid E*E \mid (E) \mid a \}, \\ E)$$

$$E \Rightarrow a$$

$$E \Rightarrow E + E \Rightarrow E + a \Rightarrow a + a$$

$$E \Rightarrow E * E \Rightarrow (E) * E \Rightarrow (E + E) * E \overset{*}{\Rightarrow} (a + a) * a$$

Grammars

Grammars

$G = (V, T, P, S)$

Grammars

$G = (V, T, P, S)$

$A \to \beta$

Grammars

$G = (V, T, P, S)$

$A \rightarrow \beta$

$\alpha\ A\ \gamma$

Grammars

$G = (V, T, P, S)$

$A \to \beta$

$\alpha\ A\ \gamma$

Grammars

$G = (V, T, P, S)$

$A \to \beta$

$\alpha A \gamma$

Grammars

$G = (V, T, P, S)$

$A \rightarrow \beta$

$$\alpha A \gamma \quad \Rightarrow$$

Grammars

$G = (V, T, P, S)$

$A \rightarrow \beta$

$$\alpha A \gamma \quad \Rightarrow$$

Grammars

$G = (V, T, P, S)$

$A \rightarrow \beta$

$$\alpha A \gamma \underset{G}{\Rightarrow}$$

Grammars

$G = (V, T, P, S)$

$A \rightarrow \beta$

$$\alpha A \gamma \underset{G}{\Rightarrow} \alpha \beta \gamma$$

Grammars

$G = (V, T, P, S)$

$A \rightarrow \beta$

$$\alpha A \gamma \underset{G}{\Rightarrow} \alpha \beta \gamma$$

Grammars

$G = (V, T, P, S)$

$A \to \beta$

$$\alpha A \gamma \underset{G}{\Rightarrow} \alpha \beta \gamma$$

Grammars

$G = (V, T, P, S)$

$A \to \beta$

$$\alpha\, A\, \gamma \underset{G}{\Rightarrow} \alpha\, \beta\, \gamma$$

$$\alpha_1 \Rightarrow \alpha_2 \Rightarrow \alpha_3 \Rightarrow \alpha_{m-1} \Rightarrow \alpha_m$$

Grammars

$G = (V, T, P, S)$

$A \rightarrow \beta$

$$\alpha\, A\, \gamma \underset{G}{\Rightarrow} \alpha\, \beta\, \gamma$$

$$\alpha_1 \Rightarrow \alpha_2 \Rightarrow \alpha_3 \Rightarrow \alpha_{m\text{-}1} \Rightarrow \alpha_m$$

$$\alpha_1 \Rightarrow \alpha_m$$

Grammars

$G = (V, T, P, S)$

$A \rightarrow \beta$

$$\alpha A \gamma \underset{G}{\Rightarrow} \alpha \beta \gamma$$

$$\alpha_1 \Rightarrow \alpha_2 \Rightarrow \alpha_3 \Rightarrow \alpha_{m-1} \Rightarrow \alpha_m$$

$$\alpha_1 \overset{*}{\Rightarrow} \alpha_m$$

Grammars

$G = (V, T, P, S)$

$A \to \beta$

$$\alpha A \gamma \underset{G}{\Rightarrow} \alpha \beta \gamma$$

$$\alpha_1 \Rightarrow \alpha_2 \Rightarrow \alpha_3 \Rightarrow \alpha_{m-1} \Rightarrow \alpha_m$$

$$\alpha_1 \overset{*}{\Rightarrow} \alpha_m$$

$$\alpha \Rightarrow \beta$$

Grammars

$G = (V, T, P, S)$

$A \rightarrow \beta$

$$\alpha A \gamma \underset{G}{\Rightarrow} \alpha \beta \gamma$$

$$\alpha_1 \Rightarrow \alpha_2 \Rightarrow \alpha_3 \Rightarrow \alpha_{m-1} \Rightarrow \alpha_m$$

$$\alpha_1 \overset{*}{\Rightarrow} \alpha_m$$

$$\alpha \overset{i}{\Rightarrow} \beta$$

Grammars

$G = (V, T, P, S)$

$A \to \beta$

$$\alpha A \gamma \underset{G}{\Rightarrow} \alpha \beta \gamma$$

$$\alpha_1 \Rightarrow \alpha_2 \Rightarrow \alpha_3 \Rightarrow \alpha_{m-1} \Rightarrow \alpha_m$$

$$\alpha_1 \overset{*}{\Rightarrow} \alpha_m$$

$$\alpha \overset{i}{\Rightarrow} \beta$$

G

Grammars

$G = (V, T, P, S)$

$A \to \beta$

$$\alpha A \gamma \underset{G}{\Rightarrow} \alpha \beta \gamma$$

$$\alpha_1 \Rightarrow \alpha_2 \Rightarrow \alpha_3 \Rightarrow \alpha_{m-1} \Rightarrow \alpha_m$$

$$\alpha_1 \overset{*}{\Rightarrow} \alpha_m$$

$$\alpha \overset{i}{\Rightarrow} \beta$$

G

Grammars

$G = (V, T, P, S)$

$A \rightarrow \beta$

$$\alpha A \gamma \underset{G}{\Rightarrow} \alpha \beta \gamma$$

$$\alpha_1 \Rightarrow \alpha_2 \Rightarrow \alpha_3 \Rightarrow \alpha_{m-1} \Rightarrow \alpha_m$$

$$\alpha_1 \overset{*}{\Rightarrow} \alpha_m$$

$$\alpha \overset{i}{\Rightarrow} \beta$$

$$L(G) = \{w \mid w \in T^* \text{ for which } S \overset{*}{\Rightarrow} w\}$$

Grammars

$G = (V, T, P, S)$

$A \rightarrow \beta$

$$\alpha A \gamma \underset{G}{\Rightarrow} \alpha \beta \gamma$$

$$\alpha_1 \Rightarrow \alpha_2 \Rightarrow \alpha_3 \Rightarrow \alpha_{m-1} \Rightarrow \alpha_m$$

$$\alpha_1 \overset{*}{\Rightarrow} \alpha_m$$

$$\alpha \overset{i}{\Rightarrow} \beta$$

$$L(G) = \{w \mid w \in T^* \text{ for which } S \overset{*}{\Rightarrow} w\}$$

Grammars

$G = (V, T, P, S)$

$A \to \beta$

$$\alpha A \gamma \underset{G}{\Rightarrow} \alpha \beta \gamma$$

$$\alpha_1 \Rightarrow \alpha_2 \Rightarrow \alpha_3 \Rightarrow \alpha_{m-1} \Rightarrow \alpha_m$$

$$\alpha_1 \overset{*}{\Rightarrow} \alpha_m$$

$$\alpha \overset{i}{\Rightarrow} \beta$$

$$L(G) = \{w \mid w \in T^* \text{ for which } S \overset{*}{\Rightarrow} w\}$$

Grammars

$G = (V, T, P, S)$

$A \rightarrow \beta$

$$\alpha A \gamma \underset{G}{\Rightarrow} \alpha \beta \gamma$$

$$\alpha_1 \Rightarrow \alpha_2 \Rightarrow \alpha_3 \Rightarrow \alpha_{m-1} \Rightarrow \alpha_m$$

$$\alpha_1 \overset{*}{\Rightarrow} \alpha_m$$

$$\alpha \overset{i}{\Rightarrow} \beta$$

$$L(G) = \{w \mid w \in T^* \text{ for which } S \overset{*}{\Rightarrow} w\}$$

Grammars

$G = (V, T, P, S)$

$A \to \beta$

$$\alpha A \gamma \underset{G}{\Rightarrow} \alpha \beta \gamma$$

$$\alpha_1 \Rightarrow \alpha_2 \Rightarrow \alpha_3 \Rightarrow \alpha_{m-1} \Rightarrow \alpha_m$$

$$\alpha_1 \overset{*}{\Rightarrow} \alpha_m$$

$$\alpha \overset{i}{\Rightarrow} \beta$$

$$L(G) = \{w \mid w \in T^* \text{ for which } S \overset{*}{\Rightarrow} w\}$$

Grammars

Grammars

Context-free languages CFL

Grammars

Context-free languages CFL

Regular languages

$RL \subset CFL$

Grammars

$$N = \{\, wcw^R \,\}$$

Context-free languages CFL

Regular languages

RL ⊂ CFL

Grammars

$N = \{ \; wcw^R \; \}$

Context-free languages CFL

N

Regular languages

$RL \subset CFL$

Grammars

$$N = \{\, wcw^R \,\}$$

Context-free languages CFL

N

Regular languages

$RL \subset CFL$

$$G = (V, T, P, S)$$

Grammars

$$N = \{ wcw^R \}$$

Context-free languages CFL

N

Regular languages

$RL \subset CFL$

$$G = (V, T, P, S)$$

$$V = \{ S \}$$

Grammars

$$N = \{\, wcw^R \,\}$$

Context-free languages CFL

N

Regular languages

$RL \subset CFL$

$G = (V, T, P, S)$

$V = \{\, S \,\}$

$T = \{\, a, b, c \,\}$

Grammars

$$N = \{\, wcw^R \,\}$$

$$G = (V, T, P, S)$$

$$V = \{\, S \,\}$$

$$T = \{\, a, b, c \,\}$$

$$P = \{\, S \to aSa \mid bSb \mid c \,\}$$

Grammars

$N = \{\, wcw^R \,\}$

Context-free languages CFL

N

Regular languages

$RL \subset CFL$

$G = (V, T, P, S)$

$V = \{\, S \,\}$

$T = \{\, a, b, c \,\}$

$P = \{\, S \rightarrow aSa \mid bSb \mid c \,\}$

S

Grammars

$N = \{ wcw^R \}$

Context-free languages CFL

N

Regular languages

$RL \subset CFL$

$G = (V, T, P, S)$

$V = \{ S \}$

$T = \{ a, b, c \}$

$P = \{ S \to aSa \mid bSb \mid c \}$

$S \Rightarrow$

Grammars

Context-free languages CFL

N

Regular languages

$RL \subset CFL$

$$N = \{ wcw^R \}$$

$$G = (V, T, P, S)$$

$$V = \{ S \}$$

$$T = \{ a, b, c \}$$

$$P = \{ S \to aSa \mid bSb \mid c \}$$

$$S \Rightarrow a \quad S \quad a$$

Grammars

$$N = \{ wcw^R \}$$

Context-free languages CFL

N

Regular languages

$RL \subset CFL$

$G = (V, T, P, S)$

$V = \{ S \}$

$T = \{ a, b, c \}$

$P = \{ S \rightarrow aSa \mid bSb \mid c \}$

$$S \Rightarrow aa \; S \; aa$$

Grammars

$$N = \{\ wcw^R\ \}$$

$$G = (V, T, P, S)$$

$$V = \{\ S\ \}$$

$$T = \{\ a, b, c\ \}$$

$$P = \{\ S \rightarrow aSa \mid bSb \mid c\ \}$$

$$S \Rightarrow\ aab\ S\ baa$$

Grammars

Context-free languages CFL

N

Regular languages

RL $\subset$ *CFL*

$$N = \{\ wcw^R\ \}$$

$$G = (V, T, P, S)$$

$$V = \{\ S\ \}$$

$$T = \{\ a, b, c\ \}$$

$$P = \{\ S \rightarrow aSa \mid bSb \mid c\ \}$$

$$S \Rightarrow aabbSbbaa$$

Grammars

$N = \{\ wcw^R\ \}$

Context-free languages CFL

N

Regular languages

$RL \subset CFL$

$$G = (V, T, P, S)$$

$$V = \{\ S\ \}$$

$$T = \{\ a, b, c\ \}$$

$$P = \{\ S \rightarrow aSa \mid bSb \mid c\ \}$$

$S \Rightarrow\ aabbSbbaa$

bbaa

Grammars

Context-free languages CFL

N

Regular languages

$RL \subset CFL$

$$N = \{ wcw^R \}$$

$$G = (V, T, P, S)$$

$$V = \{ S \}$$

$$T = \{ a, b, c \}$$

$$P = \{ S \rightarrow aSa \mid bSb \mid c \}$$

$$S \Rightarrow aabbSbbaa$$

$$aabb$$

Grammars

$N = \{ wcw^R \}$

Context-free languages CFL

N

Regular languages

$RL \subset CFL$

$G = (V, T, P, S)$

$V = \{ S \}$

$T = \{ a, b, c \}$

$P = \{ S \to aSa \mid bSb \mid c \}$

$S \Rightarrow \quad w\,S\,w^R$

Grammars

$N = \{ wcw^R \}$

Context-free languages CFL

N

Regular languages

$RL \subset CFL$

$G = (V, T, P, S)$

$V = \{ S \}$

$T = \{ a, b, c \}$

$P = \{ S \rightarrow aSa \mid bSb \mid c \}$

$S \Rightarrow \quad w\,c\,w^R$

Grammars

$G = (V, T, P, S)$

$V = \{ S \}$

$T = \{ a, b, c \}$

$P = \{ S \rightarrow aSa \mid bSb \mid c \}$

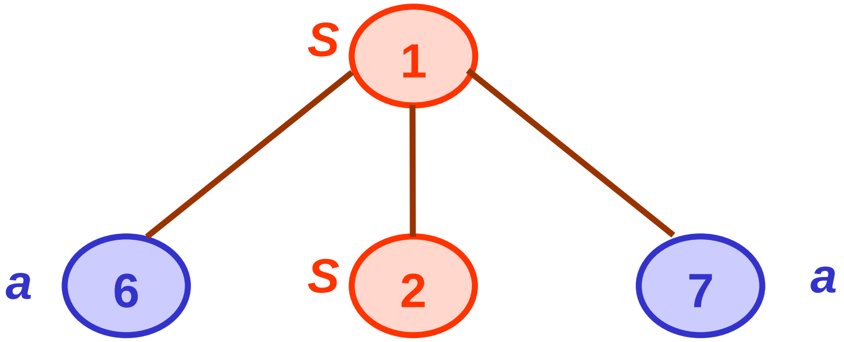
Grammars

$G = (V, T, P, S)$

$V = \{ S \}$

$T = \{ a, b, c \}$

$P = \{ S \rightarrow aSa \mid bSb \mid c \}$



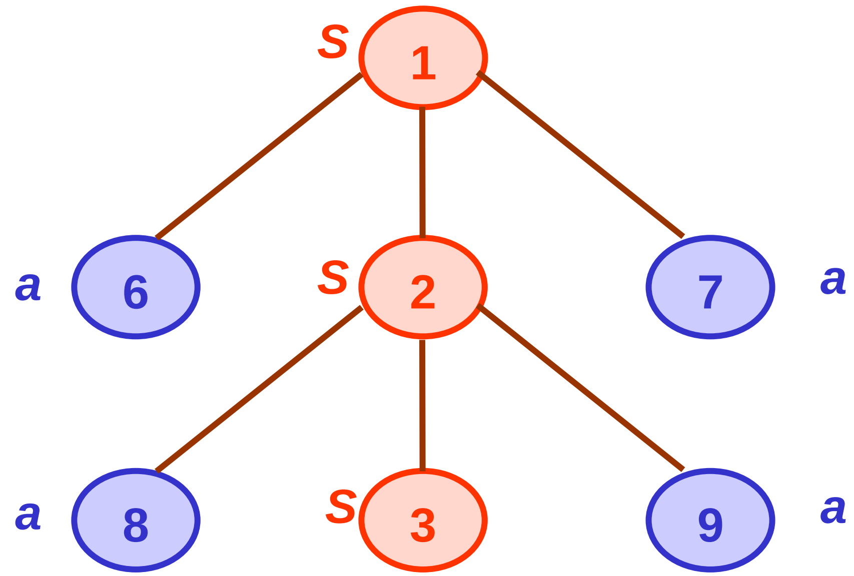
Grammars

$G = (V, T, P, S)$

$V = \{ S \}$

$T = \{ a, b, c \}$

$P = \{ S \to aSa \mid bSb \mid c \}$



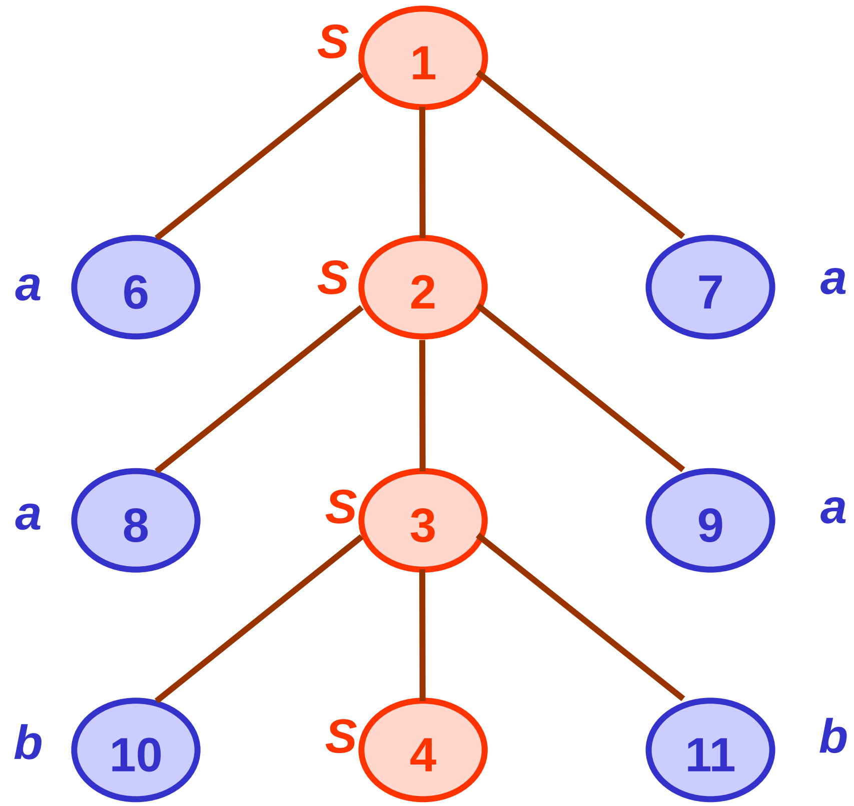
Grammars

$G = (V, T, P, S)$

$V = \{ S \}$

$T = \{ a, b, c \}$

$P = \{ S \rightarrow aSa \mid bSb \mid c \}$



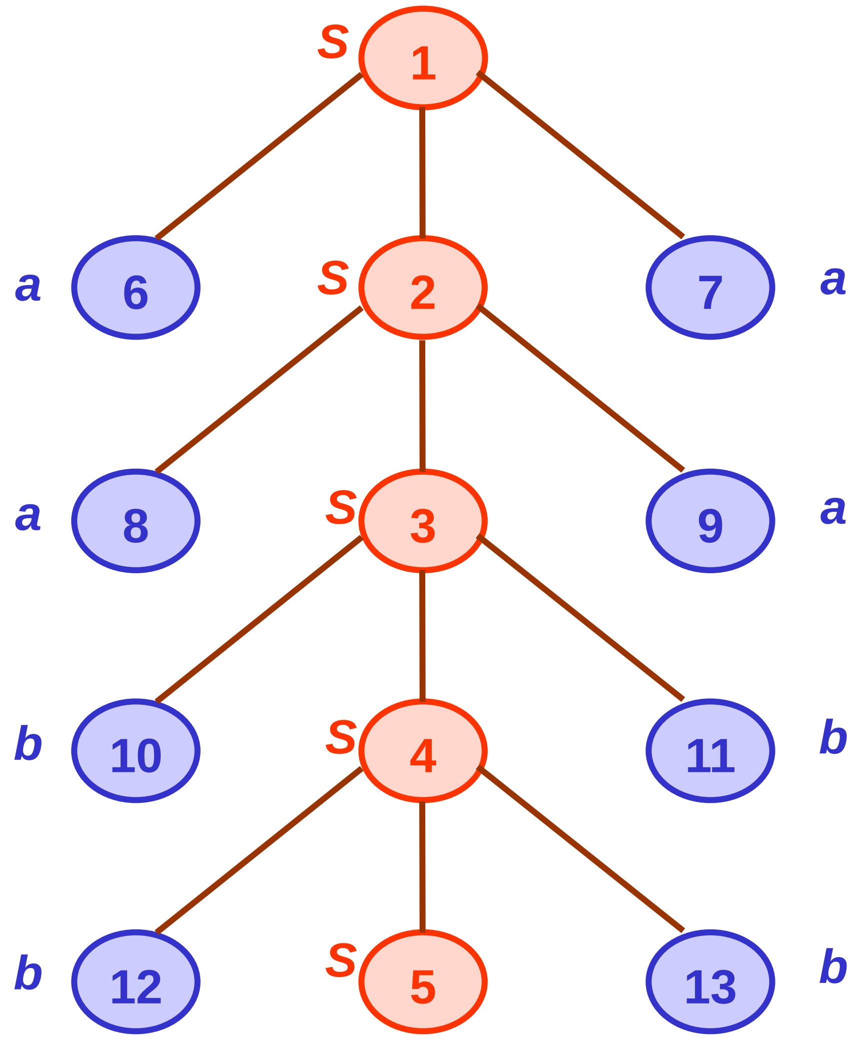
Grammars

$G = (V, T, P, S)$

$V = \{ S \}$

$T = \{ a, b, c \}$

$P = \{ S \rightarrow aSa \mid bSb \mid c \}$



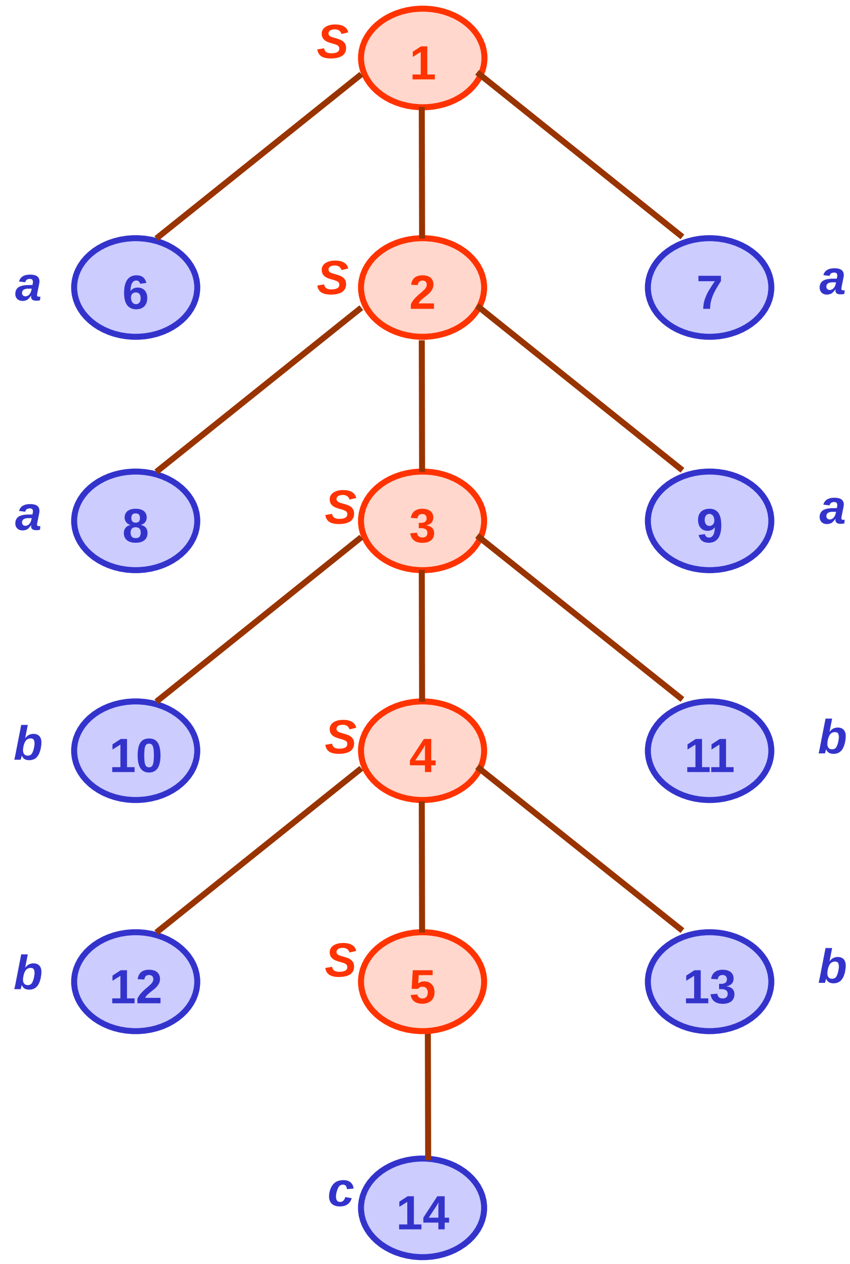
Grammars

$G = (V, T, P, S)$

$V = \{ S \}$

$T = \{ a, b, c \}$

$P = \{ S \to aSa \mid bSb \mid c \}$



Grammars

Grammars

Parse tree **for grammar** $G = (V, T, P, S)$**:**

Grammars

Parse tree **for grammar** $G = (V, T, P, S)$**:**

1) **Nodes of the tree** - $V \cup T \cup \{\varepsilon\}$

Grammars

Parse tree **for grammar** $G = (V, T, P, S)$**:**

1) **Nodes of the tree** - $V \cup T \cup \{\varepsilon\}$

2) **Root of the tree** - S

Grammars

Parse tree **for grammar** $G = (V, T, P, S)$**:**

1) **Nodes of the tree** - $V \cup T \cup \{\varepsilon\}$

2) **Root of the tree** - S

3) **Internal nodes** - $A \in V$

Grammars

Parse tree **for grammar** $G = (V, T, P, S)$**:**

1) **Nodes of the tree** - $V \cup T \cup \{\varepsilon\}$

2) **Root of the tree** - S

3) **Internal nodes** - $A \in V$

4) **Nodes** $n_1, n_2, ..., n_k$ **- child nodes of** n

Grammars

Parse tree **for grammar $G = (V, T, P, S)$:**

1)	**Nodes of the tree**	**-** $V \cup T \cup \{\varepsilon\}$
2)	**Root of the tree**	**-** S
3)	**Internal nodes**	**-** $A \in V$
4)	**Nodes** $n_1, n_2, ..., n_k$ **Node** n	**- child nodes of** n **- symbol** A

Grammars

Parse tree **for grammar $G = (V, T, P, S)$:**

1)	**Nodes of the tree**	**-** $V \cup T \cup \{\varepsilon\}$
2)	**Root of the tree**	**-** S
3)	**Internal nodes**	**-** $A \in V$
4)	**Nodes** $n_1, n_2, \ldots, n_k$ **Node** n **Nodes** $n_1, n_2, \ldots, n_k$	**- child nodes of** n **- symbol** A **- symbols** $X_1, X_2, \ldots, X_k$

Grammars

Parse tree **for grammar** $G = (V, T, P, S)$**:**

1) Nodes of the tree - $V \cup T \cup \{\varepsilon\}$

2) Root of the tree - S

3) Internal nodes - $A \in V$

4)

Nodes $n_1, n_2, \ldots, n_k$	**- child nodes of** n
Node n	**- symbol** A
Nodes $n_1, n_2, \ldots, n_k$	**- symbols** $X_1, X_2, \ldots, X_k$
$A \to X_1 X_2 \ldots X_k$	**- production from set** P

Grammars

Parse tree **for grammar** $G = (V, T, P, S)$**:**

1)	**Nodes of the tree**	**-** $V \cup T \cup \{\varepsilon\}$
2)	**Root of the tree**	**-** S
3)	**Internal nodes**	**-** $A \in V$
4)	**Nodes** $n_1, n_2, ..., n_k$ **Node** n **Nodes** $n_1, n_2, ..., n_k$ $A \rightarrow X_1 X_2 ... X_k$	**- child nodes of** n **- symbol** A **- symbols** $X_1, X_2, ..., X_k$ **- production from set** P
5)	**Character** ε	**- a tree leaf**

Grammars

***Parse tree* for grammar $G = (V, T, P, S)$:**

1)	**Nodes of the tree**	**-** $V \cup T \cup \{\varepsilon\}$
2)	**Root of the tree**	**-** S
3)	**Internal nodes**	**-** $A \in V$
4)	**Nodes** $n_1, n_2, \ldots, n_k$ **Node** n **Nodes** $n_1, n_2, \ldots, n_k$ $A \rightarrow X_1 X_2 \ldots X_k$	**- child nodes of** n **- symbol** A **- symbols** $X_1, X_2, \ldots, X_k$ **- production from set** P
5)	**Character** ε	**- a tree leaf** **- only child of its parent**

Grammars

Parse tree **for grammar $G = (V, T, P, S)$:**

1)	**Nodes of the tree**	**-** $V \cup T \cup \{\varepsilon\}$
2)	**Root of the tree**	**-** S
3)	**Internal nodes**	**-** $A \in V$
4)	**Nodes** $n_1, n_2, \ldots, n_k$	**- child nodes of** n
	Node n	**- symbol** A
	Nodes $n_1, n_2, \ldots, n_k$	**- symbols** $X_1, X_2, \ldots, X_k$
	$A \rightarrow X_1 X_2 \ldots X_k$	**- production from set** P
5)	**Character** ε	**- a tree leaf**
		- only child of its parent
6)	**Leaves of the tree**	**- symbols from** $T \cup \{\varepsilon\}$

Grammars

Parse tree **for grammar $G = (V, T, P, S)$:**

1)	**Nodes of the tree**	**-** $V \cup T \cup \{\varepsilon\}$
2)	**Root of the tree**	**-** S
3)	**Internal nodes**	**-** $A \in V$
4)	**Nodes** $n_1, n_2, \ldots, n_k$ **Node** n **Nodes** $n_1, n_2, \ldots, n_k$ $A \rightarrow X_1 X_2 \ldots X_k$	**- child nodes of** n **- symbol** A **- symbols** $X_1, X_2, \ldots, X_k$ **- production from set** P
5)	**Character** ε	**- a tree leaf** **- only child of its parent**
6)	**Leaves of the tree**	**- symbols from** $T \cup \{\varepsilon\}$ **- make up a generated string of** $L(G)$

Constructing a grammar for a regular language given by a DFA

	a	b	
S	A	B	1
A	B	A	0
B	S	A	1

Constructing a grammar for a regular language given by a DFA

	a	*b*	
S	*A*	*B*	1
A	*B*	*A*	0
B	*S*	*A*	1

Constructing a grammar for a regular language given by a DFA

	a	*b*	
S	*A*	*B*	1
A	*B*	*A*	0
B	*S*	*A*	1

$S \rightarrow aA$

Constructing a grammar for a regular language given by a DFA

	a	b	
S	A	B	1
A	B	A	0
B	S	A	1

$S \rightarrow aA$

Constructing a grammar for a regular language given by a DFA

	a	b	
S	A	B	1
A	B	A	0
B	S	A	1

$S \rightarrow aA$ $\quad$ $S \rightarrow bB$

Constructing a grammar for a regular language given by a DFA

	a	b	
S	A	B	1
A	B	A	0
B	S	A	1

$S \rightarrow aA$ $\quad$ $S \rightarrow bB$

Constructing a grammar for a regular language given by a DFA

	a	*b*	
S	*A*	*B*	1
A	*B*	*A*	0
B	*S*	*A*	1

$S \rightarrow aA \qquad S \rightarrow bB \qquad S \rightarrow \varepsilon$

Constructing a grammar for a regular language given by a DFA

	a	b	
S	A	B	1
A	B	A	0
B	S	A	1

$S \rightarrow aA$ $\quad S \rightarrow bB$ $\quad S \rightarrow \varepsilon$

Constructing a grammar for a regular language given by a DFA

	a	b	
S	A	B	1
A	B	A	0
B	S	A	1

$S \to aA \qquad S \to bB \qquad S \to \varepsilon$

$A \to aB \qquad A \to bA$

Constructing a grammar for a regular language given by a DFA

	a	b	
S	A	B	1
A	B	A	0
B	S	A	1

$S \rightarrow aA$ $\quad S \rightarrow bB$ $\quad S \rightarrow \varepsilon$

$A \rightarrow aB$ $\quad A \rightarrow bA$

Constructing a grammar for a regular language given by a DFA

	a	b	
S	A	B	1
A	B	A	0
B	S	A	1

$S \to aA \qquad S \to bB \qquad S \to \varepsilon$

$A \to aB \qquad A \to bA$

$B \to aS \qquad B \to bA \qquad B \to \varepsilon$

Constructing a grammar for a regular language given by a DFA

	a	b	
S	A	B	1
A	B	A	0
B	S	A	1

$S \rightarrow aA$ $\quad S \rightarrow bB$ $\quad S \rightarrow \varepsilon$

$A \rightarrow aB$ $\quad A \rightarrow bA$

$B \rightarrow aS$ $\quad B \rightarrow bA$ $\quad B \rightarrow \varepsilon$

String ***aba***

DFA

Grammar

Constructing a grammar for a regular language given by a DFA

	a	b	
S	A	B	1
A	B	A	0
B	S	A	1

$S \rightarrow aA$ $\quad S \rightarrow bB$ $\quad S \rightarrow \varepsilon$

$A \rightarrow aB$ $\quad A \rightarrow bA$

$B \rightarrow aS$ $\quad B \rightarrow bA$ $\quad B \rightarrow \varepsilon$

String ***aba***

DFA $\quad S$

Grammar

Constructing a grammar for a regular language given by a DFA

	a	b	
S	A	B	1
A	B	A	0
B	S	A	1

$S \to aA \quad S \to bB \quad S \to \varepsilon$

$A \to aB \quad A \to bA$

$B \to aS \quad B \to bA \quad B \to \varepsilon$

String ***aba***

DFA S

Grammar S

Constructing a grammar for a regular language given by a DFA

	a	b	
S	A	B	1
A	B	A	0
B	S	A	1

$S \to aA$ $S \to bB$ $S \to \varepsilon$

$A \to aB$ $A \to bA$

$B \to aS$ $B \to bA$ $B \to \varepsilon$

String ***aba***

DFA S

Grammar S

Constructing a grammar for a regular language given by a DFA

	a	b	
S	A	B	1
A	B	A	0
B	S	A	1

$S \to aA \qquad S \to bB \qquad S \to \varepsilon$

$A \to aB \qquad A \to bA$

$B \to aS \qquad B \to bA \qquad B \to \varepsilon$

String ***aba***

DFA $\quad S \xrightarrow{a} A$

Grammar $\quad S$

Constructing a grammar for a regular language given by a DFA

	a	b	
S	A	B	1
A	B	A	0
B	S	A	1

$S \rightarrow aA$ $\quad S \rightarrow bB$ $\quad S \rightarrow \varepsilon$

$A \rightarrow aB$ $\quad A \rightarrow bA$

$B \rightarrow aS$ $\quad B \rightarrow bA$ $\quad B \rightarrow \varepsilon$

String ***aba***

DFA $\quad S \xrightarrow{a} A$

Grammar $\quad S$

Constructing a grammar for a regular language given by a DFA

	a	b	
S	A	B	1
A	B	A	0
B	S	A	1

$S \rightarrow aA \qquad S \rightarrow bB \qquad S \rightarrow \varepsilon$

$A \rightarrow aB \qquad A \rightarrow bA$

$B \rightarrow aS \qquad B \rightarrow bA \qquad B \rightarrow \varepsilon$

String ***aba***

DFA $\quad S \xrightarrow{a} A$

Grammar $\quad S \Rightarrow aA$

Constructing a grammar for a regular language given by a DFA

	a	b	
S	A	B	1
A	B	A	0
B	S	A	1

$S \rightarrow aA$ $\quad S \rightarrow bB$ $\quad S \rightarrow \varepsilon$

$A \rightarrow aB$ $\quad A \rightarrow bA$

$B \rightarrow aS$ $\quad B \rightarrow bA$ $\quad B \rightarrow \varepsilon$

String ***aba***

DFA $\quad S \xrightarrow{a} A \xrightarrow{b} A$

Grammar $\quad S \Rightarrow aA$

Constructing a grammar for a regular language given by a DFA

	a	b	
S	A	B	1
A	B	A	0
B	S	A	1

$S \to aA \qquad S \to bB \qquad S \to \varepsilon$

$A \to aB \qquad A \to bA$

$B \to aS \qquad B \to bA \qquad B \to \varepsilon$

String aba

DFA $S \xrightarrow{a} A \xrightarrow{b} A$

Grammar $S \Rightarrow aA \Rightarrow abA$

Constructing a grammar for a regular language given by a DFA

	a	b	
S	A	B	1
A	B	A	0
B	S	A	1

$S \rightarrow aA \quad S \rightarrow bB \quad S \rightarrow \varepsilon$

$A \rightarrow aB \quad A \rightarrow bA$

$B \rightarrow aS \quad B \rightarrow bA \quad B \rightarrow \varepsilon$

String aba

DFA $\quad S \xrightarrow{a} A \xrightarrow{b} A \xrightarrow{a} B$

Grammar $\quad S \Rightarrow aA \Rightarrow abA$

Constructing a grammar for a regular language given by a DFA

	a	b	
S	A	B	1
A	B	A	0
B	S	A	1

$S \to aA \qquad S \to bB \qquad S \to \varepsilon$

$A \to aB \qquad A \to bA$

$B \to aS \qquad B \to bA \qquad B \to \varepsilon$

String *aba*

DFA $\quad S \xrightarrow{a} A \xrightarrow{b} A \xrightarrow{a} B \qquad B \in F$

Grammar $\quad S \Rightarrow aA \Rightarrow abA$

Constructing a grammar for a regular language given by a DFA

	a	b	
S	A	B	1
A	B	A	0
B	S	A	1

$S \to aA \quad S \to bB \quad S \to \varepsilon$

$A \to aB \quad A \to bA$

$B \to aS \quad B \to bA \quad B \to \varepsilon$

String ***aba***

DFA $\quad S \xrightarrow{a} A \xrightarrow{b} A \xrightarrow{a} B \quad B \in F$

Grammar $\quad S \Rightarrow aA \Rightarrow abA \Rightarrow abaB$

Constructing a grammar for a regular language given by a DFA

	a	b	
S	A	B	1
A	B	A	0
B	S	A	1

$S \to aA \qquad S \to bB \qquad S \to \varepsilon$

$A \to aB \qquad A \to bA$

$B \to aS \qquad B \to bA \qquad B \to \varepsilon$

String ***aba***

DFA $\quad S \xrightarrow{a} A \xrightarrow{b} A \xrightarrow{a} B \quad B \in F$

Grammar $\quad S \Rightarrow aA \Rightarrow abA \Rightarrow abaB \Rightarrow aba$

Constructing a grammar for a regular language given by a DFA

Constructing a grammar for a regular language given by a DFA

$G=(V, T, P, S)$

Constructing a grammar for a regular language given by a DFA

$G=(V, T, P, S)$

DFA $M=(Q, \Sigma, \delta, q_0, F)$

Constructing a grammar for a regular language given by a DFA

$G=(V, T, P, S)$

DFA $M=(Q, \Sigma, \delta, q_0, F)$

1) $T = \Sigma$

Constructing a grammar for a regular language given by a DFA

$G=(V, T, P, S)$

DFA $M=(Q, \Sigma, \delta, q_0, F)$

1) $T = \Sigma$

2) $V = Q$

Constructing a grammar for a regular language given by a DFA

$G=(V, T, P, S)$

DFA $M=(Q, \Sigma, \delta, q_0, F)$

---

1) $T = \Sigma$

2) $V = Q$

3) $S = q_0$

Constructing a grammar for a regular language given by a DFA

$G=(V, T, P, S)$

DFA $M=(Q, \Sigma, \delta, q_0, F)$

1) $T = \Sigma$

2) $V = Q$

3) $S = q_0$

4) $A \rightarrow aB$ $\delta(A, a) = B$

Constructing a grammar for a regular language given by a DFA

$G=(V, T, P, S)$

DFA $M=(Q, \Sigma, \delta, q_0, F)$

1) $T = \Sigma$

2) $V = Q$

3) $S = q_0$

4) $A \to aB$ $\quad\quad \delta(A, a) = B$

5) $A \to \varepsilon$ $\quad\quad A \in F$

Constructing a grammar for a regular language given by a DFA

Constructing a grammar for a regular language given by a DFA

DFA and grammar are equivalent

Constructing a grammar for a regular language given by a DFA

DFA and grammar are equivalent

if DFA accepts a language defined by the grammar

DFA and grammar are equivalent

if DFA accepts a language defined by the grammar

$$A \overset{*}{\Rightarrow} wB$$

Constructing a grammar for a regular language given by a DFA

DFA and grammar are equivalent

if DFA accepts a language defined by the grammar

$$A \overset{*}{\Rightarrow} wB \quad \textbf{if and only if } \delta(A, w) = B$$