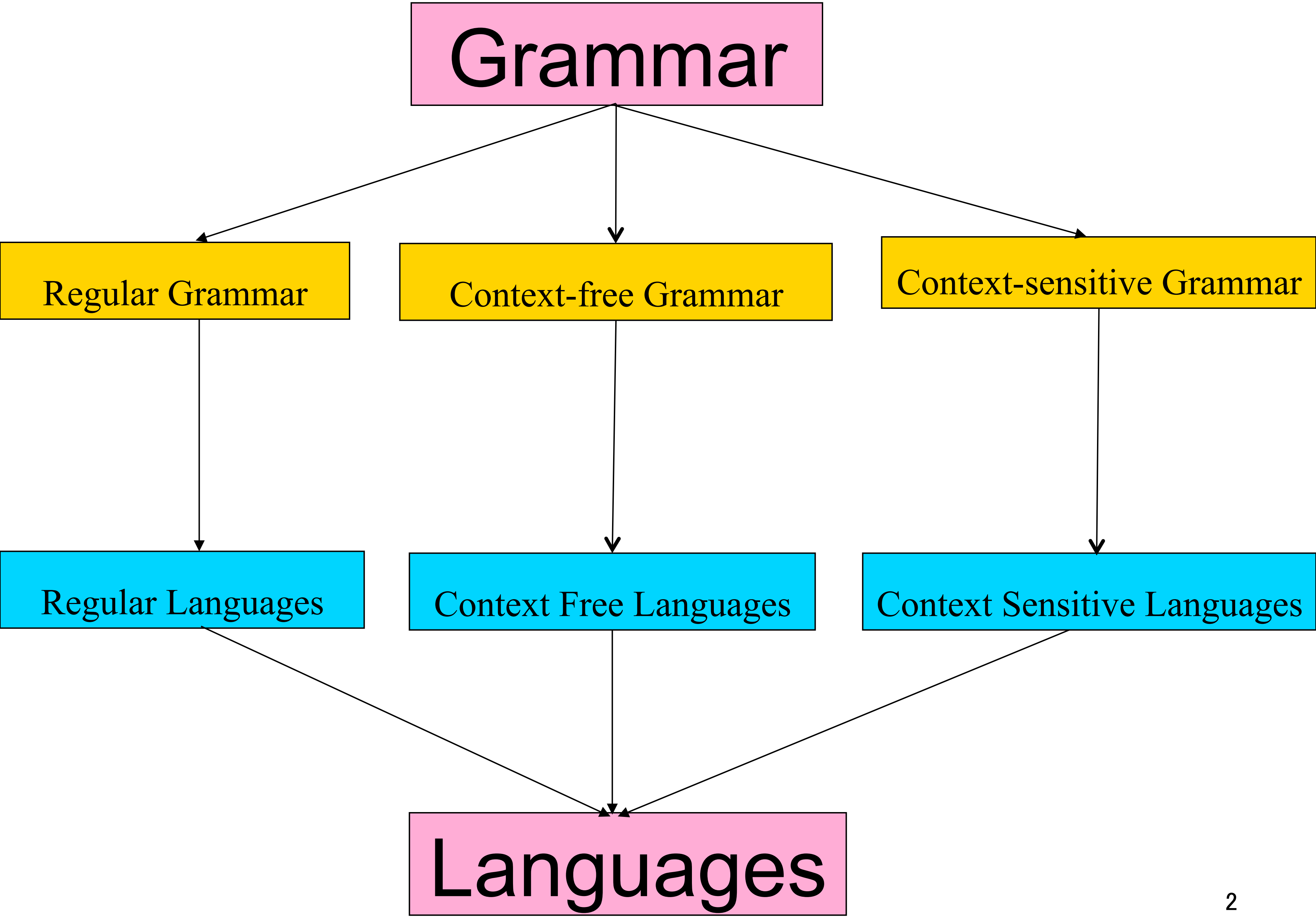
Automata and Languages

Prof. Mohamed Hamada

Software Engineering Lab.
The University of Aizu
Japan


Grammar
Regular Grammar
Context-free Grammar
Context-sensitive Grammar
Regular Languages
Context Free Languages
Context Sensitive Languages
Languages

Content

- Regular Languages
- Equivalence between Regular Grammars and Regular Languages
- Pumping Lemma (PL)
- Examples

Regular Languages

- L is a **Regular Language** if and only if there exist a finite automaton

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

such that:

$$L = L(M) = \{ w \in \Sigma^* : \delta(q_0, w) \in F \}$$

Equivalence between Regular Grammars and Regular Languages

Theorem 1

If L is a regular language then there is a right-linear grammar G =(V,T,S,P) such that L=L(G).

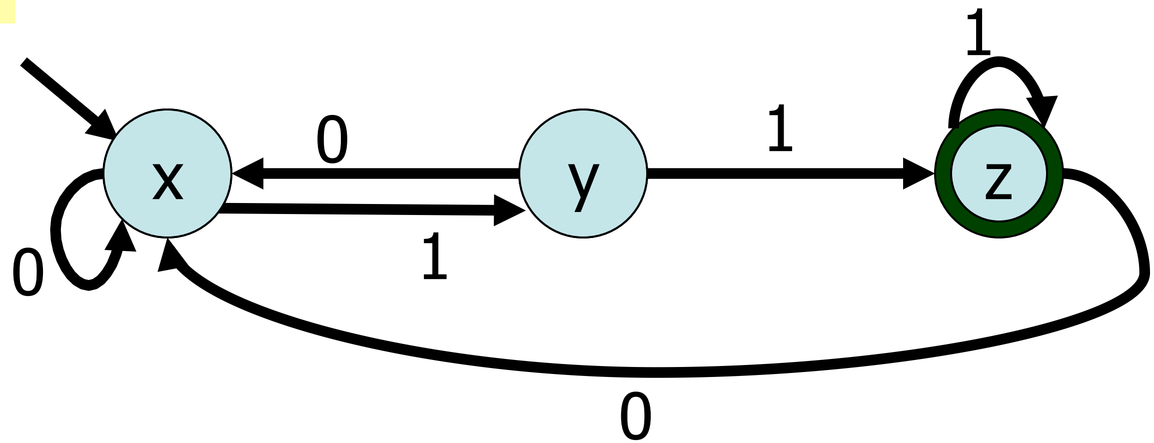
Proof. L is a regular implies (by def.) there exist a finite automaton $M = (Q, \Sigma, \delta, q_0, F)$ such that L(M)=L. Now we construct the equivalent grammar G as follows:

- Variables are the states: $V = Q$
- Start symbol is start state: $S = q_0$
- Same alphabet of terminals $T=\Sigma$
- A transition $\delta(q_1,a)=q_2$ becomes the rule $q_1 \rightarrow aq_2$
- Accept states $q \in F$ define the λ-productions $q \rightarrow \lambda$

Accepted paths give rise to terminating derivations and vice versa. L(G)=L(M).

Equivalence between Regular Grammars and Regular Languages

Example 1



The DFA above can be simulated by the grammar

$x \rightarrow 0x \mid 1y$

$y \rightarrow 0x \mid 1z$

$z \rightarrow 0x \mid 1z \mid \lambda$

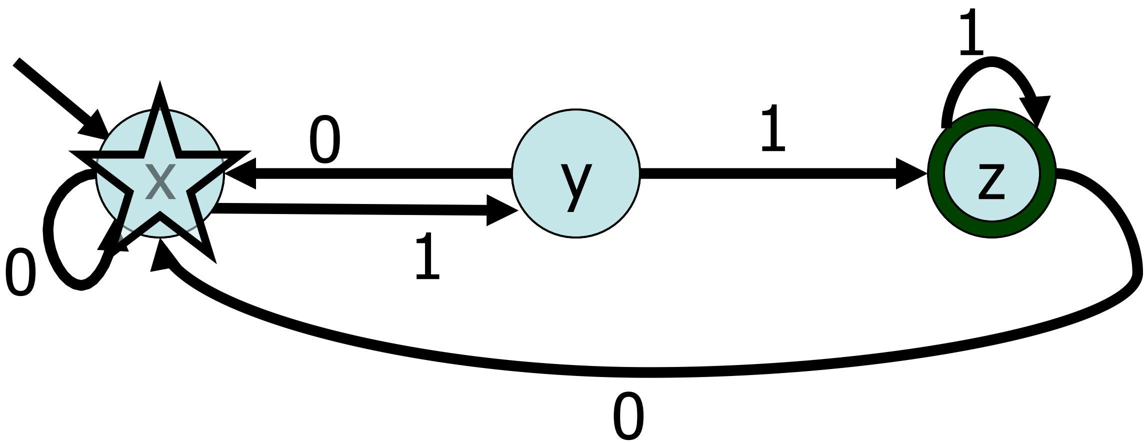
Equivalence between Regular Grammars and Regular Languages

Example 1

$x \to 0x \mid 1y$

$y \to 0x \mid 1z$

$z \to 0x \mid 1z \mid \lambda$



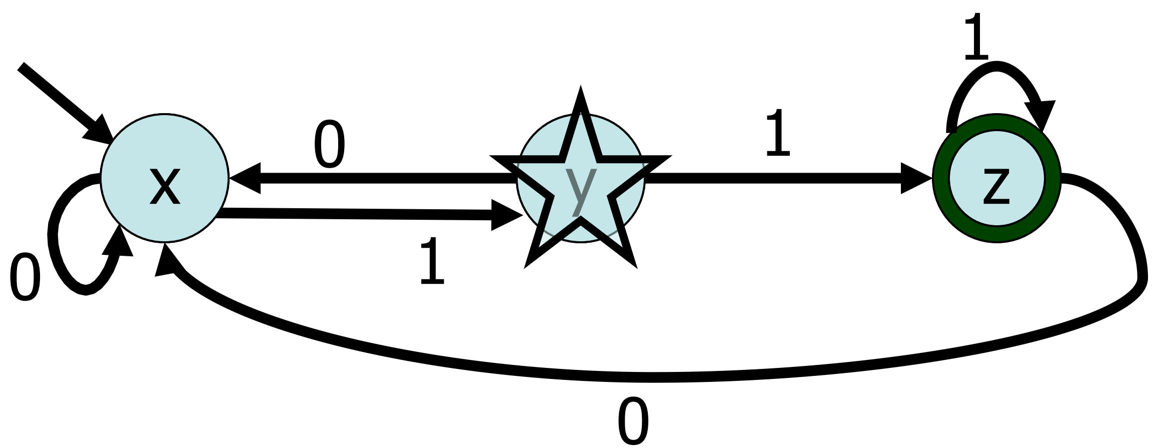
x

10011

Equivalence between Regular Grammars and Regular Languages

Example 1

$x \rightarrow 0x \mid 1y$
$y \rightarrow 0x \mid 1z$
$z \rightarrow 0x \mid 1z \mid \lambda$



$$x \Rightarrow 1y$$

10011

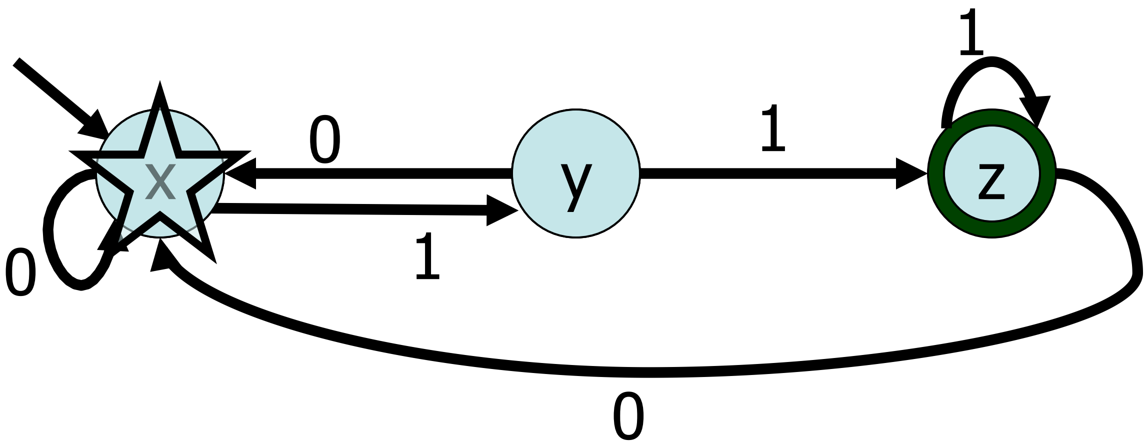
Equivalence between Regular Grammars and Regular Languages

Example 1

$x \rightarrow 0x \mid 1y$

$y \rightarrow 0x \mid 1z$

$z \rightarrow 0x \mid 1z \mid \lambda$



$$x \Rightarrow 1y \Rightarrow 10x$$

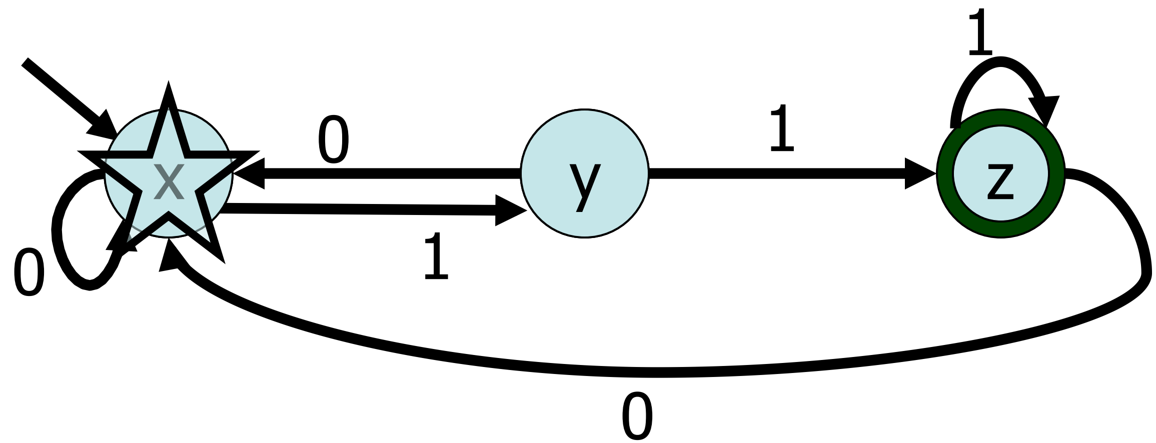
10011

Example 1

$x \to 0x \mid 1y$

$y \to 0x \mid 1z$

$z \to 0x \mid 1z \mid \lambda$



$$x \Rightarrow 1y \Rightarrow 10x \Rightarrow 100x$$

10011

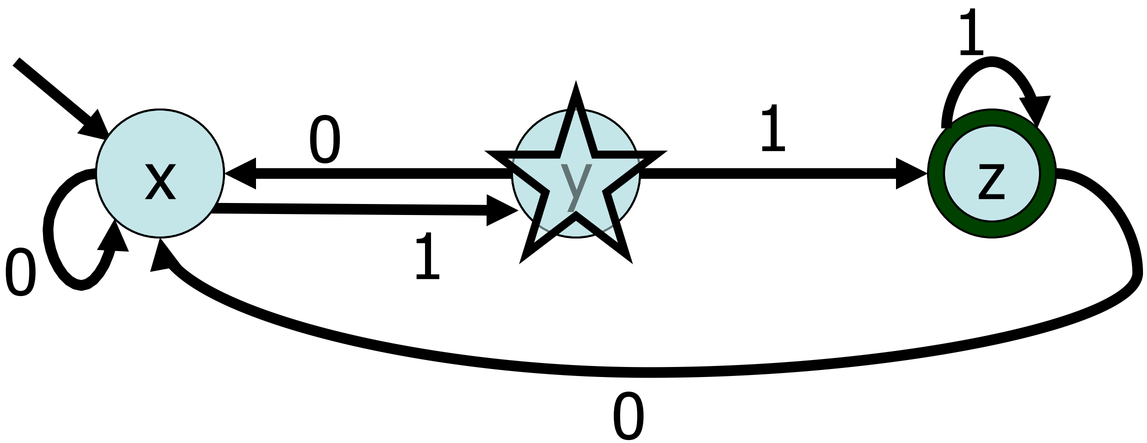
Equivalence between Regular Grammars and Regular Languages

Example 1

$x \rightarrow 0x \mid 1y$

$y \rightarrow 0x \mid 1z$

$z \rightarrow 0x \mid 1z \mid \lambda$



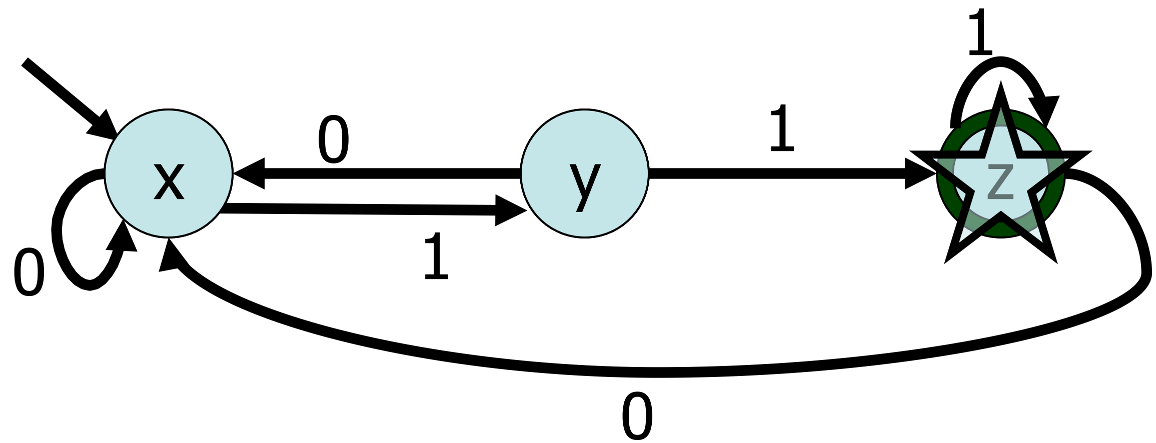
$$x \Rightarrow 1y \Rightarrow 10x \Rightarrow 100x \Rightarrow 1001y$$

10011

Equivalence between Regular Grammars and Regular Languages

Example 1

$x \rightarrow 0x \mid 1y$
$y \rightarrow 0x \mid 1z$
$z \rightarrow 0x \mid 1z \mid \lambda$



$$x \Rightarrow 1y \Rightarrow 10x \Rightarrow 100x \Rightarrow 1001y$$
$$\Rightarrow 10011z$$

10011

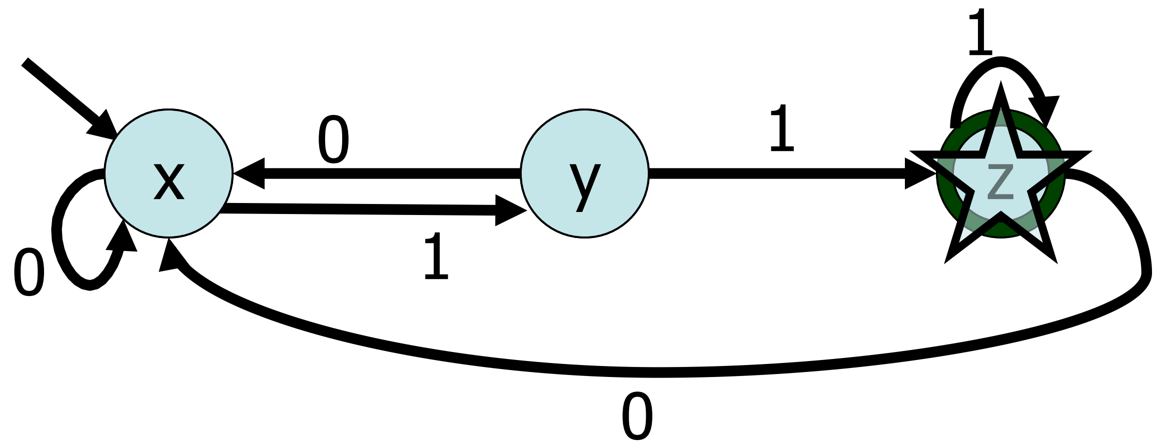
Equivalence between Regular Grammars and Regular Languages

Example 1

$x \rightarrow 0x \mid 1y$

$y \rightarrow 0x \mid 1z$

$z \rightarrow 0x \mid 1z \mid \lambda$



$$x \Rightarrow 1y \Rightarrow 10x \Rightarrow 100x \Rightarrow 1001y$$
$$\Rightarrow 10011z \Rightarrow 10011$$

10011

ACCEPT!

Theorem 2

If G =(V,T,S,P) is a right-linear grammar then L(G) is a regular language.

Proof. : Define a FA $M = (Q, \Sigma, \delta, q_0, F)$ as follows

- Start state q_0 correspond to start symbol S
- A non-final state q_i corresponds to a variable symbol V_i
- Same alphabet of terminals $\Sigma = \mathrm{T}$
- For every rule $V_i \rightarrow a_1 \ldots a_m V_j$, define a transition $\delta(q_i, a_1 \ldots a_m) = q_j$
- For every rule $V_i \rightarrow a_1 \ldots a_m$, define a transition $\delta(q_i, a_1 \ldots a_m) = q_f$ final state

Terminating derivations give rise to accepted paths and vice versa. So L(M)=L(G). Hence (by def.) L(G) is a regular language.

Theorem 2

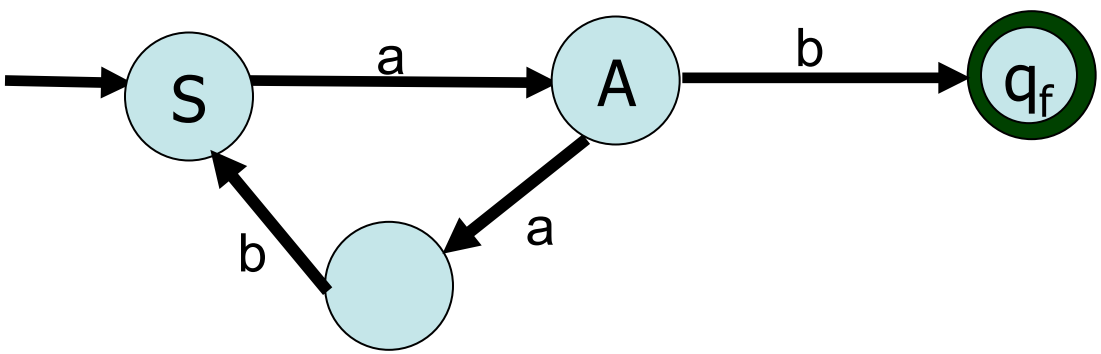
Construct an FA that is equivalent to the right-linear grammar:

$S \to aA$

$A \to abS$

$A \to b$

Answer:



Comments

- THEOREM 1 and THEOREM 2 show that right-linear grammars and regular languages are equivalent.
- Similarly we can show that left-linear grammars and regular languages are equivalent.
- Hence we conclude that Regular Grammars and Regular Languages are equivalent.

Regular Languages

Q:

Can every CFG be converted into a right linear grammar?

A:

NO! This would mean that all context free languages are regular.

For example:

$$S \rightarrow \lambda \mid aSb$$

cannot be converted because $\{a^n b^n\}$ is not regular.

Regular Languages

Q:

How we can identify non-regular languages?

A:

By using a technique called "Pumping Lemma"

Pumping Lemma (PL)

Motivation

Consider the language

$$L_1 = 01^* = \{0, 01, 011, 0111, \ldots\}$$

The string $0\underline{1}1$ is said to be ***pumpable*** in L_1

because can take the underlined portion, and pump it up (i.e. repeat) as much as desired while *always* getting elements in L_1.

Pumping Lemma (PL)

Motivation

Consider the language

$$L_1 = 01^* = \{0, 01, 011, 0111, \dots\}$$

Q:

Which of the following are pumpable?

1. 01111
2. 01
3. 0

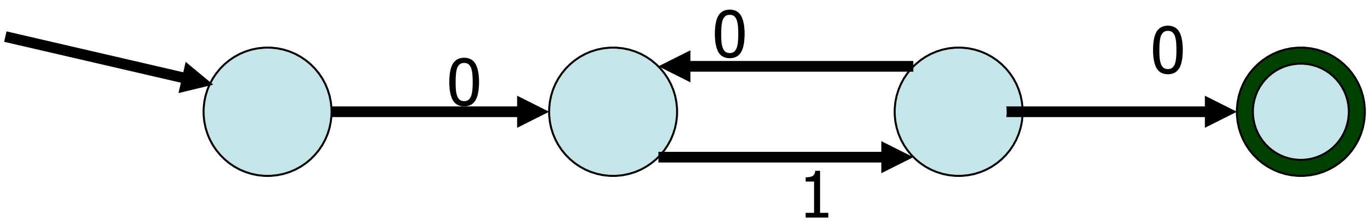
A:

1. Pumpable: $011\underline{1}1$, $0\underline{1}111$, $0\underline{111}1$, $0\underline{1111}$, etc.
2. Pumpable: $0\underline{1}$
3. 0 *not* pumpable because most of 0^* not in L_1

Pumping Lemma (PL)

Motivation

Define L_2 by the following automaton:



Q: Is 01010 pumpable?

A: Pumpable: 0$\underline{10}$10, 01$\underline{01}$0. Underlined substrings correspond to cycles in the FA!

Cycles in the FA can be repeated arbitrarily often, hence pumpable.

Pumping Lemma (PL)

Motivation

Let $L3 = \{011,11010,000, \lambda\}$

Q:

Which strings are pumpable?

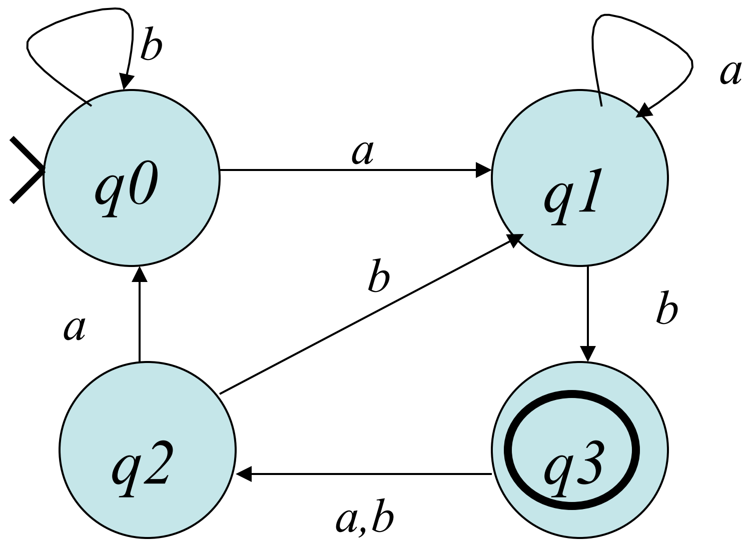
A:

None! When pumping any string non-trivially, always result in infinitely many possible strings. So no pumping can go on inside a finite set.

Pumping Lemma give a criterion for when strings can be pumped.

Pumping Lemma (PL)

Motivation



We have: $$\boxed{ababbaaab \in L(M)}$$

Because:

$$\boxed{q0 \xrightarrow{a} q1 \xrightarrow{b} q3 \xrightarrow{a} q2 \xrightarrow{b} q1 \xrightarrow{b} q3 \xrightarrow{a} q2 \xrightarrow{a} q0 \xrightarrow{a} q1 \xrightarrow{b} q3}$$

Pumping Lemma (PL)

Motivation

Note, $q0 \xrightarrow{a} q1 \xrightarrow{b} q3 \xrightarrow{a} q2 \xrightarrow{b} q1 \xrightarrow{b} q3 \xrightarrow{a} q2 \xrightarrow{a} q0 \xrightarrow{a} q1 \xrightarrow{b} q3$

So, $\boxed{ababb \in L(M)}$

Also, $q0 \xrightarrow{a} q1 \xrightarrow{b} q3 \xrightarrow{a} q2 \xrightarrow{b} q1 \xrightarrow{b} q3 \xrightarrow{a} q2 \xrightarrow{a} q0 \xrightarrow{a} q1 \xrightarrow{b} q3$

So, $\boxed{abaaab \in L(M)}$

We note that:

$$\forall i, j \in \mathbb{N}: \quad ab(abb)^i(aaab)^j \in L(M)$$

Pumping Lemma (PL)

Theorem

- Given an (infinite) regular language L, there is a number p (called the ***pumping number***) such that any string in L of length $\geq p$ is pumpable within its first p letters.
- In other words, for all $u \in L$ with

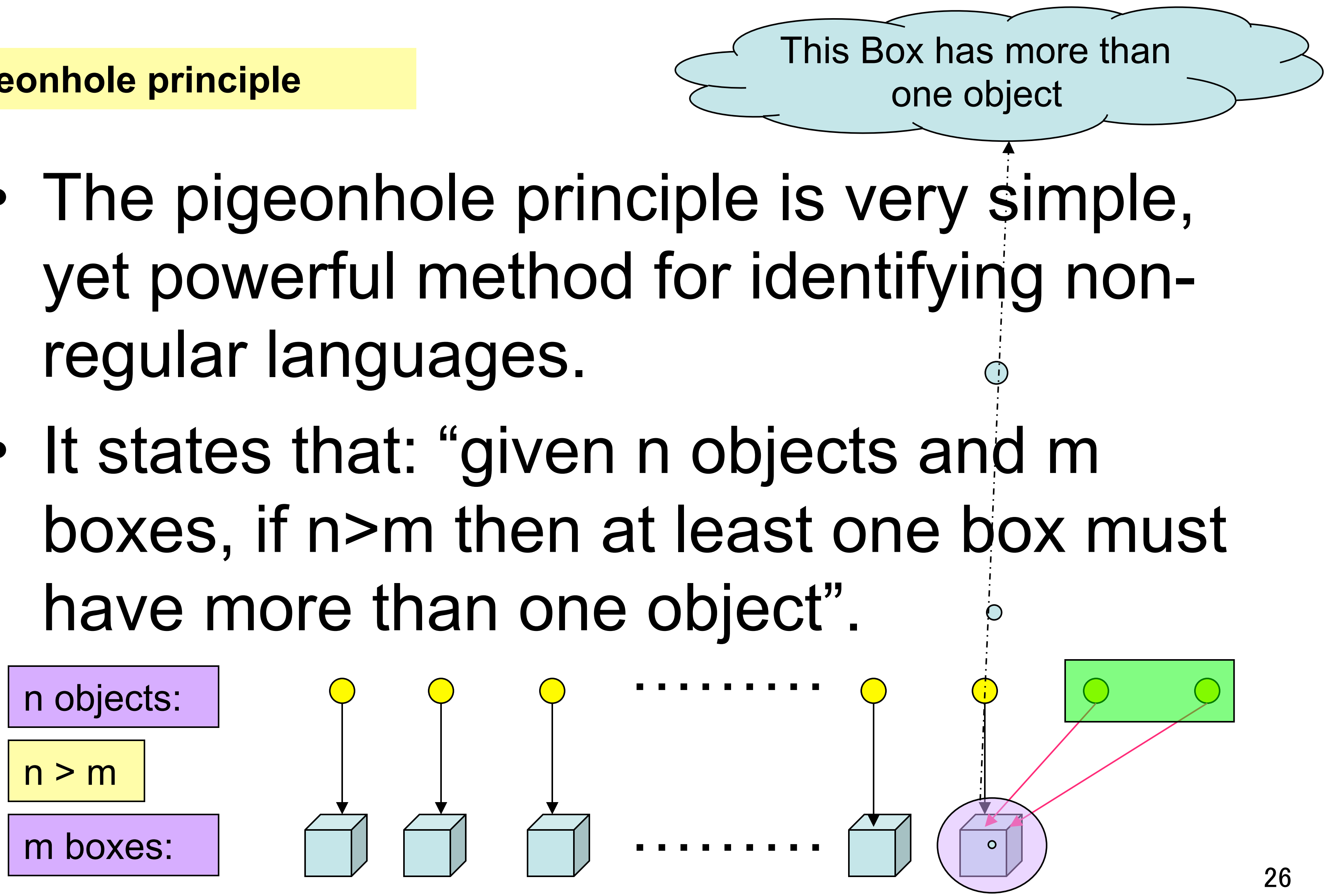
  $|u| \geq p$ we can write:
  - $u = xyz$ (x is a prefix, z is a suffix)
  - $|y| \geq 1$ (mid-portion y is non-empty)
  - $|xy| \leq p$ (pumping occurs in first p letters)
  - $xy^iz \in L$ for all $i \geq 0$ (can pump y-portion)

To prove the Pumping Lemma we need to know the *Pigeonhole Principle*

Pumping Lemma (PL)

Pigeonhole principle

- The pigeonhole principle is very simple, yet powerful method for identifying non-regular languages.
- It states that: "given n objects and m boxes, if n>m then at least one box must have more than one object".



Pumping Lemma (PL)

Pigeonhole principle fundamental observation

- Given a “sufficiently” long string, the states of a DFA must repeat in an accepting computation. These cycles can then be used to predict (generate) infinitely many other strings in (of) the language.

Pigeon-Hole Principle

Pumping Lemma (PL)

Proof

Now consider an accepted string *u*.
By assumption *L* is regular so let *M* be the FA accepting it.

Let $p = |Q| =$ no. of states in *M*.
Suppose $|u| \geq p$.

The path labeled by *u* visits $p+1$ states in its first *p* letters.

Thus (by pigeonhole principle) *u* must visit some state twice.

The sub-path of *u* connecting the first and second visit of the vertex is a loop, and gives the claimed string *y* that can be pumped within the first *p* letters.

Pumping Lemma (PL)

Notes:

- It is a necessary condition.
  - Every regular language satisfies it.
  - If a language violates it, it is not regular.
    - RL => PL   not PL => not RL
- It is *not* a sufficient condition.
  - Not every non-regular language violates it.
    - not RL =>? PL or not PL (no conclusion)

Pumping Lemma (PL)

Notes:

For all sufficiently long strings (u)
There exists non-null prefix (xy)
and substring (y)
For all repetitions of the substring (y),
we get strings in the language.

$$\forall u \in L: \ |u| \geq k \ \Rightarrow$$
$$\exists x, y, z: \ (xyz = u)$$
$$\wedge \ (|xy| \leq p) \ \wedge \ (|y| \geq 1)$$
$$\wedge \ (\forall i : i \geq 0 \Rightarrow xy^i z \in L)$$

Pumping Lemma (PL)

Proving non-regularity

- If there exists an *arbitrarily* long string $u \in L$, and for each decomposition $u = xyz$, there exists an i such that $xy^i z \notin L$, then L is non-regular.

Negation of the necessary condition:

$$\exists u \in L : \ |u| \geq p \ \wedge$$
$$\forall x, y, z : \ (xyz = u) \ \wedge \ (|xy| \leq p) \ \wedge \ (|y| \geq 1)$$
$$\Rightarrow (\exists i : i \geq 0 \wedge xy^i z \notin L)$$

Pumping Lemma (PL)

Proving non-regularity

In general, to prove that *L* isn't regular:

1. Assume *L* were regular
2. Therefore it has a pumping no. *p*
3. *Find a string pattern involving the length p in some clever way, and which cannot be pumped.* ***This is the hard part*****.**
4. (2)→←(3) <contradiction> Therefore our assumption (1) was wrong and conclude that *L* is *not* a regular language

Explanation of Step 3: How to get a contradiction

1. Let m be the pumping number

2. Choose a particular string $w \in L$ which satisfies the length condition $|w| \geq m$

3. Write $w = xyz$

4. Show that $w' = xy^i z \notin L$ for some $i \neq 1$

5. This gives a contradiction, since from pumping lemma $w' = xy^i z \in L$

Example

Show that the language $L = \{a^n b^n : n \geq 0\}$

is not regular

Answer: **Use the Pumping Lemma**

Example

$$L = \{a^n b^n : n \geq 0\}$$

Assume for contradiction
that L is a regular language

Since L is infinite
we can apply the Pumping Lemma

Example $L = \{a^n b^n : n \geq 0\}$

Let m **be the Pumping number**

Pick a string w **such that:** $w \in L$

and length $|w| \geq m$

We pick $\boxed{w = a^m b^m}$

Example **From the Pumping Lemma:**

we can write $w = a^m b^m = x\ y\ z$

with lengths $|\,x\,y\,| \le m, \quad |\,y\,| \ge 1$

$$w = xyz = a^m b^m = \overbrace{\underbrace{a\ldots a}_{x}\underbrace{a\ldots a}_{y}a\ldots a}^{m}\overbrace{b\ldots b}^{m}$$

$$\hphantom{w = xyz = a^m b^m = a\ldots a}\underbrace{\hphantom{a\ldots a}a\ldots ab\ldots b}_{z}$$

Thus: $y = a^k, \quad 1 \le k \le m$

Example

$$x\ y\ z = a^m b^m \qquad y = a^k, \quad 1 \le k \le m$$

From the Pumping Lemma:

$$x\ y^i\ z \in L$$

$$i = 0, 1, 2, \ldots$$

Thus: $x\ y^2\ z \in L$

Example

$$x\ y\ z = a^m b^m \qquad y = a^k, \quad 1 \le k \le m$$

From the Pumping Lemma: $x\ y^2\ z \in L$

$$xy^2z = \overbrace{\underbrace{a...a}_{x}\underbrace{a...a}_{y}\underbrace{a...a}_{y}a...a}^{m+k}\overbrace{b...b}^{m} \in L$$

where $z = a...ab...b$

Thus: $a^{m+k}b^m \in L$

Example

$$a^{m+k}b^m \in L \qquad k \geq 1$$

BUT: $L = \{a^n b^n : n \geq 0\}$

$$\Downarrow$$

$$a^{m+k}b^m \notin L$$

CONTRADICTION!!!

Example

Therefore: **Our assumption that *L* is a regular language is not true**

Conclusion: L **is not a regular language**

Pumping Lemma (PL)

Exercise

Show that the following languages are not regular:

$$L_p = \{a^p \mid p \text{ is a prime number}\}$$

$$L_c = \{a^c \mid c \text{ is a composite number}\}$$

$$L = \{\omega \in \{a,b\}^* \mid \#a\text{'s in } \omega = \#b\text{'s in } \omega\}$$

$$\boldsymbol{L_{\mathbf{pal}}} = \{x \in \Sigma^* \mid x = x^R\}$$