COMP2111 Week 8
Term 1, 2019
Regular languages and beyond

Summary

- Regular expressions
- Myhill-Nerode theorem
- Context-free languages
- Mealy machines
- LTL: Logic for transition systems

Summary

- Regular expressions
- Myhill-Nerode theorem
- Context-free languages
- Mealy machines
- LTL: Logic for transition systems

Regular expressions

Regular expressions are a way of describing "finite automaton" patterns:

- Second-last letter is b
- Every odd symbol is b

Many applications in CS:

- Lexical analysis in compiler construction
- Search facilities provided by text editors and databases; utilities such as `grep` and `awk`
- Programming languages such as Perl and XML

Regular expressions

Given a finite set Σ, a **regular expression (RE) over Σ** is defined recursively as follows:

- $\emptyset$ is a regular expression
- ϵ is a regular expression
- a is a regular expression for all $a \in \Sigma$
- If E_1 and E_2 are regular expressions, then $E_1 E_2$ is a regular expression
- If E_1 and E_2 are regular expressions, then $E_1 + E_2$ is a regular expression
- If E is a regular expression, then E^* is a regular expression

We use parentheses to disambiguate REs, though $*$ binds tighter than concatenation, which binds tighter than $+$.

Examples

Example

The following are regular expressions over $\Sigma = \{0, 1\}$:

- $\emptyset$
- $101 + 010$
- $(\epsilon + 10)^*01$

Language of a Regular expression

A RE defines a language over Σ: the set of words which "match" the expression:

- Concatenation = sequences of expressions
- Union = choice of expressions
- Star = 0 or more occurrences of an expression

Example

The following words match $(000 + 10)^{*}01$:

- 01
- 101001
- 000101000001

Language of a Regular Expression

Formally, given an RE, E, over Σ, we define $L(E) \subseteq \Sigma^*$ recursively as follows:

- If $E = \emptyset$ then $L(E) = \emptyset$
- If $E = \epsilon$ then $L(E) = \{\lambda\}$
- If $E = a$ where $a \in \Sigma$ then $L(E) = \{a\}$
- If $E = E_1 E_2$, then $L(E) = L(E_1) \cdot L(E_2)$
- If $E = E_1 + E_2$, then $L(E) = L(E_1) \cup L(E_2)$
- If $E = E_1^*$ then $L(E) = (L(E_1))^*$

Example

$$L(010 + 101) = ?$$

$$L((\epsilon + 10)^* 01) = ?$$

Language of a Regular Expression

Formally, given an RE, E, over Σ, we define $L(E) \subseteq \Sigma^*$ recursively as follows:

- If $E = \emptyset$ then $L(E) = \emptyset$
- If $E = \epsilon$ then $L(E) = \{\lambda\}$
- If $E = a$ where $a \in \Sigma$ then $L(E) = \{a\}$
- If $E = E_1 E_2$, then $L(E) = L(E_1) \cdot L(E_2)$
- If $E = E_1 + E_2$, then $L(E) = L(E_1) \cup L(E_2)$
- If $E = E_1^*$ then $L(E) = (L(E_1))^*$

Example

$$L(010 + 101) = \{010, 101\}$$

$$L((\epsilon + 10)^* 01) = ?$$

Language of a Regular Expression

Formally, given an RE, E, over Σ, we define $L(E) \subseteq \Sigma^*$ recursively as follows:

- If $E = \emptyset$ then $L(E) = \emptyset$
- If $E = \epsilon$ then $L(E) = \{\lambda\}$
- If $E = a$ where $a \in \Sigma$ then $L(E) = \{a\}$
- If $E = E_1 E_2$, then $L(E) = L(E_1) \cdot L(E_2)$
- If $E = E_1 + E_2$, then $L(E) = L(E_1) \cup L(E_2)$
- If $E = E_1^*$ then $L(E) = (L(E_1))^*$

Example

$$L(010 + 101) = \{010, 101\}$$

$$L((\epsilon + 10)^* 01) = \{01, 1001, 101001, \ldots\}$$

Regular expressionss vs NfAs

Theorem (Kleene's theorem)

- *For any regular expression* E, $L(E)$ *is a regular language.*
- *For any regular language* L, *there is a regular expression* E *such that* $L = L(E)$

Proof of Kleene's theorem

Given E, $L(E)$ is a regular language. Proof by induction on E.

Given L, find E such that $L = L(E)$

Proof of Kleene's theorem

Given E, $L(E)$ is a regular language. Proof by induction on E.

Given L, find E such that $L = L(E)$

- Let $L^X_{q,q'} = \{w \in \Sigma^* \ : \ q \xrightarrow{w}{}^{*} q' \text{ with all intermediate states in } X\}$
- Define $E^X_{q,q'}$ such that $L(E^X_{q,q'}) = L^X_{q,q'}$:
  - When $q = q'$: $E^{\emptyset}_{q,q'} = \epsilon + a_1 + a_2 + \ldots + a_k$ where $q \xrightarrow{a_i} q$
  - When $q \neq q'$: $E^{\emptyset}_{q,q'} = \emptyset + a_1 + a_2 + \ldots + a_k$ where $q \xrightarrow{a_i} q'$
  - For $X \neq \emptyset$:

$$E^X_{q,q'} = \underbrace{E^{X-\{r\}}_{q,q'}}_{(1)} + \underbrace{E^{X-\{r\}}_{q,r} \cdot (E^{X-\{r\}}_{r,r})^* \cdot E^{X-\{r\}}_{r,q'}}_{(2)}$$

- The required expression is then $E = \sum_{q \in F} E^Q_{q_0,q}$

Proof of Kleene's theorem

Given E, $L(E)$ is a regular language. Proof by induction on E.

Given L, find E such that $L = L(E)$

- Let
  $L^X_{q,q'} = \{w \in \Sigma^* \ : \ q \xrightarrow{w}{}^* \ q' \text{ with all intermediate states in } X\}$
- Define $E^X_{q,q'}$ such that $L(E^X_{q,q'}) = L^X_{q,q'}$:
  - When $q = q'$: $E^{\emptyset}_{q,q'} = \epsilon + a_1 + a_2 + \ldots + a_k$ where $q \xrightarrow{a_i} q$
  - When $q \neq q'$: $E^{\emptyset}_{q,q'} = \emptyset + a_1 + a_2 + \ldots + a_k$ where $q \xrightarrow{a_i} q'$
  - For $X \neq \emptyset$:

$$E^X_{q,q'} = \underbrace{E^{X-\{r\}}_{q,q'}}_{(1)} + \underbrace{E^{X-\{r\}}_{q,r} \cdot (E^{X-\{r\}}_{r,r})^* \cdot E^{X-\{r\}}_{r,q'}}_{(2)}$$

- The required expression is then $E = \sum_{q \in F} E^{Q}_{q_0,q}$

Proof of Kleene's theorem

Given E, $L(E)$ is a regular language. Proof by induction on E.

Given L, find E such that $L = L(E)$

- Let $L^X_{q,q'} = \{w \in \Sigma^* \ : \ q \xrightarrow{w}{}^* q' \text{ with all intermediate states in } X\}$
- Define $E^X_{q,q'}$ such that $L(E^X_{q,q'}) = L^X_{q,q'}$:
  - When $q = q'$: $E^{\emptyset}_{q,q'} = \epsilon + a_1 + a_2 + \ldots + a_k$ where $q \xrightarrow{a_i} q$
  - When $q \neq q'$: $E^{\emptyset}_{q,q'} = \emptyset + a_1 + a_2 + \ldots + a_k$ where $q \xrightarrow{a_i} q'$
  - For $X \neq \emptyset$:

$$E^X_{q,q'} = \underbrace{E^{X-\{r\}}_{q,q'}}_{(1)} + \underbrace{E^{X-\{r\}}_{q,r} \cdot (E^{X-\{r\}}_{r,r})^* \cdot E^{X-\{r\}}_{r,q'}}_{(2)}$$

- The required expression is then $E = \sum_{q \in F} E^Q_{q_0,q}$

Proof of Kleene's theorem

Given E, $L(E)$ is a regular language. Proof by induction on E.

Given L, find E such that $L = L(E)$

- Let
  $L^X_{q,q'} = \{w \in \Sigma^* \ : \ q \xrightarrow{w}{}^* q' \text{ with all intermediate states in } X\}$
- Define $E^X_{q,q'}$ such that $L(E^X_{q,q'}) = L^X_{q,q'}$:
  - When $q = q'$: $E^{\emptyset}_{q,q'} = \epsilon + a_1 + a_2 + \ldots + a_k$ where $q \xrightarrow{a_i} q$
  - When $q \neq q'$: $E^{\emptyset}_{q,q'} = \emptyset + a_1 + a_2 + \ldots + a_k$ where $q \xrightarrow{a_i} q'$
  - For $X \neq \emptyset$:

$$E^X_{q,q'} = \underbrace{E^{X-\{r\}}_{q,q'}}_{(1)} + \underbrace{E^{X-\{r\}}_{q,r} \cdot (E^{X-\{r\}}_{r,r})^* \cdot E^{X-\{r\}}_{r,q'}}_{(2)}$$

- The required expression is then $E = \sum_{q \in F} E^Q_{q_0,q}$

Example

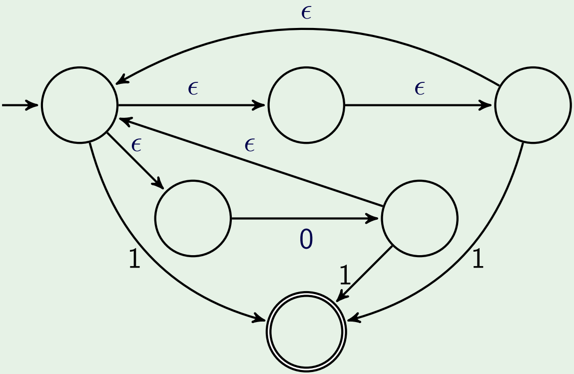
Example

Construct an NFA for $(\epsilon + 0)^*1$

Example

Example

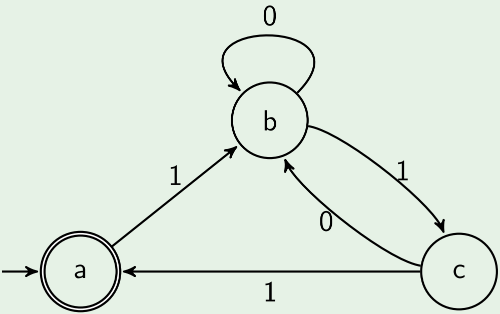
Construct an NFA for $(\epsilon + 0)^*1$



Example

Example

Find a regular expression for this NFA:



Example (ctd)

Example

Picking c as the separating state:

$$E_{a,a}^{\{a,b,c\}} = E_{a,a}^{\{a,b\}} + E_{a,c}^{\{a,b\}} \cdot (E_{c,c}^{\{a,b\}})^* \cdot E_{c,a}^{\{a,b\}}.$$

By inspection $E_{a,a}^{\{a,b\}} = \epsilon$, $E_{a,c}^{\{a,b\}} = 10^*1$ and $E_{c,a}^{\{a,b\}} = 1$.

Now picking b as the separating state:

$$E_{c,c}^{\{a,b\}} = E_{c,c}^{\{a\}} + E_{c,b}^{\{a\}} \cdot (E_{b,b}^{\{a\}})^* \cdot E_{b,c}^{\{a\}}$$

where $E_{c,c}^{\{a\}} = \epsilon$, $E_{c,b}^{\{a\}} = 0 + 11$, $E_{b,b}^{\{a\}} = \epsilon + 0$ and $E_{b,c}^{\{a\}} = 1$.

Putting it all together we have

$$E_{a,a}^{\{a,b,c\}} = \epsilon + 10^*1(\epsilon + (0+11)(\epsilon+0)^*1)^*1$$

Summary

- Regular expressions
- Myhill-Nerode theorem
- Context-free languages
- Mealy machines
- LTL: Logic for transition systems

L-indistinguishability

Let $x, y \in \Sigma^*$ and let $L \subseteq \Sigma^*$.
We say that x and y are L-**indistinguishable**, written $x \equiv_L y$, if for every $z \in \Sigma^*$,

$$xz \in L \quad \text{if and only if} \quad yz \in L.$$

Fact

$\equiv_L$ is an equivalence relation.

We define the **index** of L to be the number of equivalence classes of $\equiv_L$.

NB

The index of L may be finite or infinite.

L-indistinguishability

Let $x, y \in \Sigma^*$ and let $L \subseteq \Sigma^*$.
We say that x and y are L-**indistinguishable**, written $x \equiv_L y$, if for every $z \in \Sigma^*$,

$$xz \in L \quad \text{if and only if} \quad yz \in L.$$

Fact

$\equiv_L$ is an equivalence relation.

We define the **index** of L to be the number of equivalence classes of $\equiv_L$.

NB

The index of L may be finite or infinite.

Examples

Example

Take $\Sigma = \{0, 1\}$.

$$L_1 = \{w : w \text{ has even length}\}.$$

$$u \equiv_{L_1} v \text{ iff } \text{length}(u) \equiv \text{length}(v) (\text{mod } 2).$$

Now $\equiv_{L_1}$ has two equivalence classes:
$[\epsilon] = [00] = [10] = \cdots = \{w : \text{length}(w) \text{ even}\}$ and
$[0] = [1] = [010] = [110] = \cdots = \{w : \text{length}(w) \text{ odd}\}$.

Examples

Example

Take $\Sigma = \{0, 1\}$

$$L_2 = \{w : w \text{ has equal numbers of 0s and 1s}\}.$$

For any $i, j \geq 0$, if $i \neq j$ then $0^i \not\equiv_{L_2} 0^j$ (because $0^i 1^i \in L_2$ but $0^j 1^i \notin L_2$).

Therefore the index of L_2 is infinite.

Myhill-Nerode theorem

Theorem (Myhill-Nerode theorem)

L is regular if and only if L has finite index.

Moreover, the index is the size ($=$ number of states) of the smallest DFA accepting L.

Example

Example

Take $\Sigma = \{a, b\}$.

$$L_n = \{w : \text{ the } n\text{-th last symbol of } w \text{ is } b\}$$

What is the index of L_n?

Example

Example

Take $\Sigma = \{a, b\}$.

$$L_n = \{w : \text{ the } n\text{-th last symbol of } w \text{ is } b\}$$

What is the index of L_n?

- An NFA with n states can accept L_n
- So there is a DFA with 2^n states that accepts L_n
- So the index is at most 2^n

Example

Example

Take $\Sigma = \{a, b\}$.

$$L_n = \{w : \text{ the } n\text{-th last symbol of } w \text{ is } b\}$$

What is the index of L_n?

- An NFA with n states can accept L_n
- So there is a DFA with 2^n states that accepts L_n
- So the index is at most 2^n

Example

Example

Take $\Sigma = \{a, b\}$.

$$L_n = \{w : \text{ the } n\text{-th last symbol of } w \text{ is } b\}$$

What is the index of L_n?

- An NFA with n states can accept L_n
- So there is a DFA with 2^n states that accepts L_n
- So the index is at most 2^n

Example

Example

Take $\Sigma = \{a, b\}$.

$$L_n = \{w : \text{ the } n\text{-th last symbol of } w \text{ is } b\}$$

What is the index of L_n?

Take $w, v \in \Sigma^n$ with $w \neq v$. Suppose w and v differ in the i-th symbol, $0 \leq i \leq n-1$. Let $z = a^{n-i}$.

- Then only one of wz, vz is in L_n
- So w and v are L_n-distinguishable ($w \not\equiv_{L_n} v$)
- So the index is at least 2^n

Example

Example

Take $\Sigma = \{a, b\}$.

$$L_n = \{w : \text{ the } n\text{-th last symbol of } w \text{ is } b\}$$

What is the index of L_n?
Take $w, v \in \Sigma^n$ with $w \neq v$. Suppose w and v differ in the i-th symbol, $0 \leq i \leq n-1$. Let $z = a^{n-i}$.

- Then only one of wz, vz is in L_n
- So w and v are L_n-distinguishable ($w \not\equiv_{L_n} v$)
- So the index is at least 2^n

Example

Example

Take $\Sigma = \{a, b\}$.

$$L_n = \{w : \text{ the } n\text{-th last symbol of } w \text{ is } b\}$$

What is the index of L_n?
Take $w, v \in \Sigma^n$ with $w \neq v$. Suppose w and v differ in the i-th symbol, $0 \leq i \leq n-1$. Let $z = a^{n-i}$.

- Then only one of wz, vz is in L_n
- So w and v are L_n-distinguishable ($w \not\equiv_{L_n} v$)
- So the index is at least 2^n

Example

Example

Take $\Sigma = \{a, b\}$.

$$L_n = \{w : \text{ the } n\text{-th last symbol of } w \text{ is } b\}$$

What is the index of L_n?
Take $w, v \in \Sigma^n$ with $w \neq v$. Suppose w and v differ in the i-th symbol, $0 \leq i \leq n-1$. Let $z = a^{n-i}$.

- Then only one of wz, vz is in L_n
- So w and v are L_n-distinguishable ($w \not\equiv_{L_n} v$)
- So the index is at least 2^n

Summary

- Regular expressions
- Myhill-Nerode theorem
- Context-free languages
- Mealy machines
- LTL: Logic for transition systems

Context-free grammars

Regular languages can be specified in terms of finite automata that accept or reject strings, equivalently, in terms of regular expressions, which strings are to match.

Grammars are a *generative* means of specifying sets of strings.

Context-free grammars (CFG): A way of generating words

Ingredients of a CFG:

$$(\{\text{variables}\}, \{\text{terminals}\}, \{\text{productions (or rules)}\}, \text{start symbol})$$

The start symbol is a special variable.
A CFG generates strings over the alphabet $\Sigma = \{\text{terminals}\}$.

Example

$G = (\{A, B\}, \{0, 1\}, \mathcal{R}, A)$ where $\mathcal{R}$ consists of three rules:

$$\begin{cases} A & \rightarrow & 0\,A\,1 \\ A & \rightarrow & B \\ B & \rightarrow & \epsilon \end{cases}$$

How to generate strings using a CFG

1. Set w to be the start symbol.
2. Choose an occurrence of a variable X in w if any, otherwise STOP.
3. Pick a production whose lhs is X, replace the chosen occurrence of X in w by the rhs.
4. GOTO 2.

Example

$G = (\{A, B\}, \{0, 1\}, \{A \to 0\,A\,1 \mid B, \quad B \to \epsilon\}, A)$ generates $\{0^i\,1^i : i \geq 0\}$.

$$\begin{aligned} A &\Rightarrow 0\,A\,1 \\ &\Rightarrow 0\,0\,A\,1\,1 \\ &\Rightarrow 0\,0\,B\,1\,1 \\ &\Rightarrow 0\,0\,\epsilon\,1\,1 = 0^2\,1^2 \end{aligned}$$

Such sequences are called **derivations**.

Formal definition

A **context-free grammar** is a 4-tuple $G = (V, \Sigma, \mathcal{R}, S)$ where

- V is a finite set of *variables* (or *non-terminals*)
- Σ (the alphabet) is a finite set of *terminals*
- $\mathcal{R}$ is a finite set of *productions*. A *production* (or *rule*) is an element of $V \times (V \cup \Sigma)^*$, written $A \to w$.
- $S \in V$ is the *start symbol*.

We define a binary relation $\Rightarrow$ over $(\{V \cup \Sigma\})^*$ by: for each $u, v \in (\{V \cup \Sigma\})^*$, for each $A \to w$ in $\mathcal{R}$

$$u\,A\,v \Rightarrow u\,w\,v$$

The **language generated by the grammar**, $L(G)$, is $\{w \in \Sigma^* : S \Rightarrow^* w\}$.

A language is **context-free** if it can be generated by a CFG.

Examples

Example

Well-balanced parentheses: generated by $(\{S\}, \{\,(\,,\,)\,\}, \mathcal{R}, S)$ where $\mathcal{R}$ consists of

$$S \;\rightarrow\; (\;S\;) \mid S\,S \mid \epsilon$$

E.g. $(\;\;(\;\;)\;\;(\;\;(\;\;)\;\;)\;\;)\;\;(\;\;)$

Examples

Example

Inductively defined syntax:

- Well-formed formulas
- $\mathcal{L}$
- Regular expressions

WFFs: Generated by $(\{\varphi\}, \Sigma, \mathcal{R}, \varphi)$ where $\Sigma = \mathrm{P{\scriptsize ROP}} \cup \{\top, \bot, (,), \neg, \wedge, \vee, \rightarrow, \leftrightarrow\}$ and $\mathcal{R}$ consists of

$$\varphi \rightarrow \top \mid \bot \mid P \mid \neg\varphi \mid (\varphi \wedge \varphi) \mid (\varphi \vee \varphi) \mid (\varphi \rightarrow \varphi) \mid (\varphi \leftrightarrow \varphi)$$

Examples

Example

Inductively defined syntax:

- Well-formed formulas
- $\mathcal{L}$
- Regular expressions

WFFs: Generated by $(\{\varphi\}, \Sigma, \mathcal{R}, \varphi)$ where $\Sigma = \textsc{Prop} \cup \{\top, \bot, (,), \neg, \wedge, \vee, \rightarrow, \leftrightarrow\}$ and $\mathcal{R}$ consists of

$$\varphi \;\rightarrow\; \top \mid \bot \mid P \mid \neg\varphi \mid (\varphi \wedge \varphi) \mid (\varphi \vee \varphi) \mid (\varphi \rightarrow \varphi) \mid (\varphi \leftrightarrow \varphi)$$

Examples

Example

A small English language

$$
\begin{array}{rcl}
\langle \text{SENTENCE} \rangle & \rightarrow & \langle \text{NOUN-PHRASE} \rangle \ \langle \text{VERB-PHRASE} \rangle \\
\langle \text{NOUN-PHRASE} \rangle & \rightarrow & \langle \text{CMPLX-NOUN} \rangle \quad | \quad \langle \text{CMPLX-NOUN} \rangle \ \langle \text{PREP-PHRASE} \\
\langle \text{VERB-PHRASE} \rangle & \rightarrow & \langle \text{CMPLX-VERB} \rangle \quad | \quad \langle \text{CMPLX-VERB} \rangle \ \langle \text{PREP-PHRASE} \rangle \\
\langle \text{PREP-PHRASE} \rangle & \rightarrow & \langle \text{PREP} \rangle \ \langle \text{CMPLX-NOUN} \rangle \\
\langle \text{CMPLX-NOUN} \rangle & \rightarrow & \langle \text{ARTICLE} \rangle \ \langle \text{NOUN} \rangle \\
\langle \text{CMPLX-VERB} \rangle & \rightarrow & \langle \text{VERB} \rangle \quad | \quad \langle \text{VERB} \rangle \ \langle \text{NOUN-PHRASE} \rangle \\
\langle \text{ARTICLE} \rangle & \rightarrow & \mathsf{a} \quad | \quad \mathsf{the} \\
\langle \text{NOUN} \rangle & \rightarrow & \mathsf{boy} \quad | \quad \mathsf{girl} \quad | \quad \mathsf{flower} \\
\langle \text{VERB} \rangle & \rightarrow & \mathsf{touches} \quad | \quad \mathsf{like} \quad | \quad \mathsf{see} \\
\langle \text{PREP} \rangle & \rightarrow & \mathsf{with}
\end{array}
$$

Examples

Example

A small English language

$$
\begin{aligned}
\langle\text{SENTENCE}\rangle &\Rightarrow \langle\text{NOUN-PHRASE}\rangle\ \langle\text{VERB-PHRASE}\rangle\\
&\Rightarrow \langle\text{CMPLX-NOUN}\rangle\ \langle\text{PREP-PHRASE}\rangle\ \langle\text{VERB-PHRASE}\rangle\\
&\Rightarrow \langle\text{ARTICLE}\rangle\ \langle\text{NOUN}\rangle\ \langle\text{PREP-PHRASE}\rangle\ \langle\text{VERB-PHRASE}\rangle\\
&\Rightarrow \text{a girl}\ \langle\text{PREP}\rangle\ \langle\text{CMPLX-NOUN}\rangle\ \langle\text{VERB-PHRASE}\rangle\\
&\Rightarrow \text{a girl with}\ \langle\text{CMPLX-NOUN}\rangle\ \langle\text{VERB-PHRASE}\rangle\\
&\Rightarrow \text{a girl with}\ \langle\text{ARTICLE}\rangle\ \langle\text{NOUN}\rangle\ \langle\text{VERB-PHRASE}\rangle\\
&\Rightarrow \text{a girl with a flower}\ \langle\text{VERB-PHRASE}\rangle\\
&\Rightarrow \text{a girl with a flower}\ \langle\text{CMPLX-VERB}\rangle\\
&\Rightarrow \text{a girl with a flower}\ \langle\text{VERB}\rangle\ \langle\text{NOUN-PHRASE}\rangle\\
&\Rightarrow \text{a girl with a flower likes}\ \langle\text{CMPLX-NOUN}\rangle\\
&\Rightarrow \text{a girl with a flower likes}\ \langle\text{ARTICLE}\rangle\ \langle\text{NOUN}\rangle\\
&\Rightarrow \text{a girl with a flower likes the boy}
\end{aligned}
$$

Regular languages vs Context-free languages

A CFG is **right-linear** if every rule is either of the form $R \rightarrow wT$ or of the form $R \rightarrow w$ where w ranges over strings of terminals, and R and T over variables.

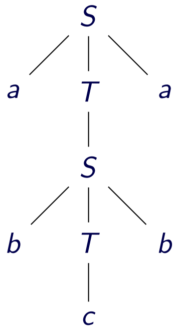
Theorem

A language is regular if and only if it is generated by a right-linear CFG.

Parse trees

Each derivation determines a **parse tree**.
Parse trees are *ordered* trees: the children at each node are ordered.
The parse tree of a derivation abstracts away from the order in which variables are replaced in the sequence.

$$
\begin{aligned}
S &\Rightarrow a\,T\,a \\
&\Rightarrow a\,S\,a \\
&\Rightarrow a\,b\,T\,b\,a \\
&\Rightarrow a\,b\,c\,b\,a
\end{aligned}
$$



Properties of CFLs

Context-free languages are closed under union

Context-free languages are **not** closed under complement nor intersection

Properties of CFLs

Context-free languages are closed under union

Context-free languages are **not** closed under complement nor intersection

Pushdown automata

CFLs can be recognized by **Pushdown automata**:

- Non-deterministic finite automaton, PLUS
- Stack memory:
  - Infinite capacity for storing inputs
  - Can recover top-most memory item to influence transitions