Computational Models - Lecture 3

- Non Regular Languages: Two Approaches
- (1) The Pumping Lemma
- (2) Myhill-Nerode Theorem (not in Sipser's book)
- Closure properties
- Algorithmic questions for NFAs

- Sipser's book, 1.4, 2.1, 2.2
- Hopcroft and Ullman, 3.4

Proved Last Time

Thm.: A language, L, is described by a regular expression, R, if and only if L is regular.

$\Longrightarrow$ construct an NFA accepting R.

$\Longleftarrow$ Given a regular language, L, construct an equivalent regular expression

We have made a lot of progress understanding what finite automata can do. But what can't they do?

Negative Results

Is there a DFA that accepts

- $B = \{0^n 1^n | n \geq 0\}$
- $C = \{w | w$ has an equal number of 0's and 1's$\}$
- $D = \{w | w$ has an equal number of occurrences of 01 and 10 substrings$\}$

Consider B:

- DFA must "remember" how many 0's it has seen
- impossible with finite state.

The others are exactly the same...

Question: Is this a proof?
Answer: No, D is regular!???

Pumping Lemma

Pumping Lemma

We will show that all regular languages have a special property.

- Suppose L is regular.
- If a string in L is longer than a certain critical length ℓ (the pumping length),
- then it can be "pumped" to a longer string by repeating an internal substring any number of times.
- The longer string must be in L too.
- This is a powerful technique for showing that a language is not regular.

Pumping Lemma

Theorem: If L is a regular language, then there is an $\ell > 0$ (the pumping length), where if s is any string in L of length $|s| \geq \ell$, then s may be divided into three pieces $s = xyz$ such that

- for every $i \geq 0$, $xy^i z \in L$,
- $|y| > 0$, and
- $|xy| \leq \ell$.

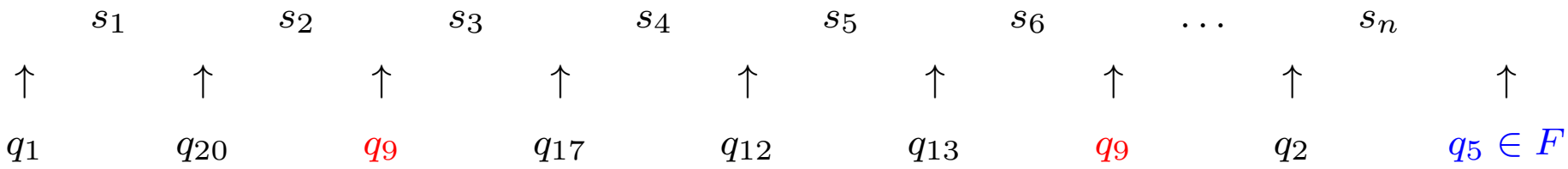
Remarks: Without the second condition, the theorem would be trivial.
The third condition is technical and sometimes useful.

Pumping Lemma – Proof

Let $M = (Q, \Sigma, \delta, q_1, F)$ be a DFA that accepts L.

Let ℓ be $|Q|$, the number of states of M.

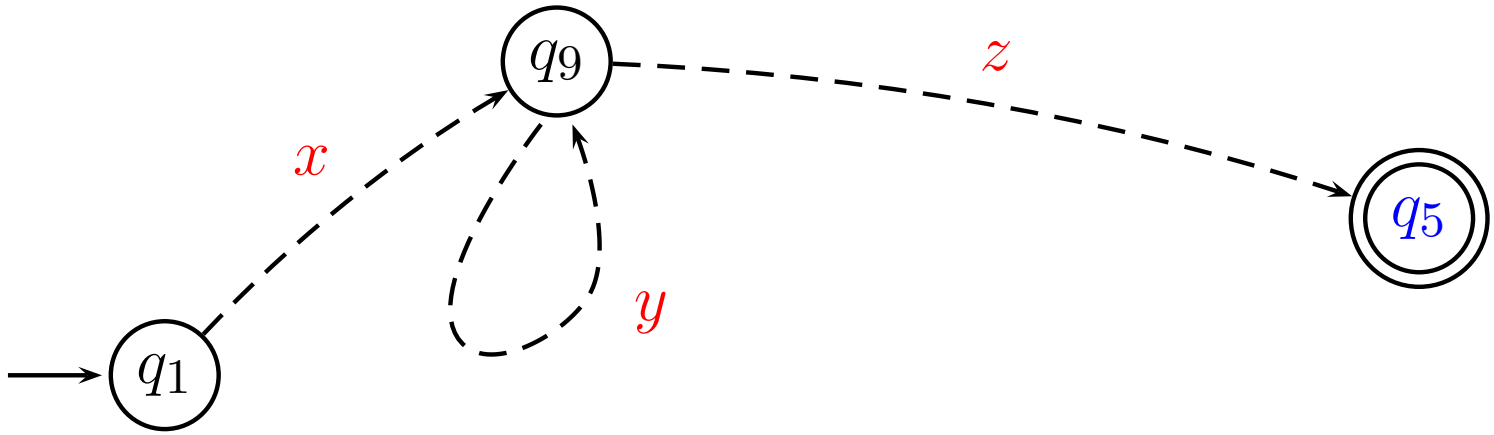
If $s \in L$ has length at least ℓ, consider the sequence of states M goes through as it reads s:



Since the sequence of states is of length $|s| + 1 > \ell$, and there are only ℓ different states in Q, at least one state is repeated (by the pigeonhole principle).

Pumping Lemma – Proof (cont.)

Write down $s = xyz$



- By inspection, M accepts $xy^k z$ for every $k \geq 0$.
- $|y| > 0$ because the state (q_9 in figure) is repeated.
- To ensure that $|xy| \leq \ell$, pick first state repetition, which must occure no later than $\ell + 1$ states in sequence.

Application # 1

Corollary: The language $B = \{0^n 1^n | n > 0\}$ is not regular.

Proof: By contradiction. Suppose B is regular, accepted by DFA M. Let 2ℓ be the pumping length.

- Consider the string $s = 0^\ell 1^\ell$.
- By pumping lemma $s = xyz$, where $xy^k z \in B$ for every $k \geq 0$.
- If y is all 0, then $xy^k z$ has too many 0's, for $k \geq 2$.
- If y is all 1, then $xy^k z$ has too many 1's, for $k \geq 2$.
- If y is mixed, then $xy^k z$ is not of right form. ♣

Using the Pumping Lemma

How to prove that a language is not regular

- Select a language L
- An adversary sets the parameter ℓ.
- Select a word $s \in L$.
- The word s would (usually) depend on ℓ.
- The adversary selects a partition $s = xyz$, such that $|y| \geq 1$ and $|xy| \leq \ell$.
- Show an index k, such that $xy^k z \notin L$.
- Need to prove for any parameter ℓ and any partition xyz.

Application # 2

Corollary: The language
$C = \{w | w$ has an equal number of 0's and 1's$\}$
is not regular.

Proof: By contradiction. Suppose C is regular, accepted by DFA M. Let ℓ be the pumping length.

- Consider the string $s = 0^\ell 1^\ell$.
- By pumping lemma $s = xyz$, where $xy^k z \in C$ for every $k \geq 0$.
- Since $|xy| \leq \ell$ then y is all 0, and $xy^k z$ has too many 0's. ♣

What about $D = \{w | w$ has an equal number of occurrences of 01 and 10 substrings$\}$?

Application # 3

Corollary: The language $E = \left\{0^i 1^j | i > j\right\}$ is not regular.

Proof: By contradiction. Suppose E is regular, accepted by DFA M. Let ℓ be its pumping length.

- Consider the string $s = 0^\ell 1^{\ell-1}$.
- By pumping lemma $s = xyz$, where $xy^k z \in E$ for every $k \geq 0$, $|y| > 0$ and $|xy| \leq \ell$
- But for $k = 0$ we have $xz \notin E$, contradiction.

Application # 4

Corollary: The language $Primes \subset \{1\}^*$, which contains all strings whose length is a prime number, is not regular.

Proof: By contradiction. Suppose $Primes$ is regular, accepted by DFA M. Let ℓ be the pumping length.

- Let $s = 1^p \in Primes$, where $p \geq \ell$ is a prime.
- By pumping lemma $s = xyz$, where $xy^k z \in Primes$ for every k.
- For $k = p + 1$ we have $xy^k z = 1^{p+mp}$, where $|y| = m$.
- since $p(m + 1)$ is not prime, we have a contradiction. ♣

Another Example

Consider the language

$L = \{a^i b^n c^n | n \geq 0, i \geq 1\} \cup \{b^n c^m | n, m \geq 0\}$,

For any word $s \in L$ we can apply the pumping lemma:

- If $s = a^i b^n c^n$, then set $x = \varepsilon$ and $y = a$.
- If $s = b^n c^m$, then set $x = \varepsilon$ and $y = b$.
- Is L regular?!
- How can we prove it?!

Characterization of Regular Languages

The Equivalence Relation $\sim_L$

Let $L \subseteq \Sigma^*$ be a language.

Define an equivalence relation $\sim_L$ on pairs of strings:

Let $x, y \in \Sigma^*$. We say that $x \sim_L y$ if for every string $z \in \Sigma^*$, $xz \in L$ if and only if $yz \in L$.

It is easy to see that $\sim_L$ is indeed an equivalence relation (reflexive, symmetric, transitive) on Σ^*.

In addition, if $x \sim_L y$ then for every string $z \in \Sigma^*$, $xz \sim_L yz$ as well (this is called right invariance).

The Equivalence Relation $\sim_L$ cont.

Like every equivalence relation, $\sim_L$ partitions Σ^* to (disjoint) equivalence classes. For every string x, let $[x] \subseteq \Sigma^*$ denote its equivalence class w.r.t. $\sim_L$ (if $x \sim_L y$ then $[x] = [y]$ – equality of sets).

Question is, how many equivalence classes does $\sim_L$ induce?

In particular, is the number of equivalence classes of $\sim_L$ finite or infinite?

Well, it could be either finite or infinite. This depends on the language L.

Three Examples

- Let $L_1 \subset \{0,1\}^*$ contain all strings where the number of 1s is divisible by 4. Then $\sim_{L_1}$ has finitely many equivalence classes.
- Let $L_2 \subset \{0,1\}^*$ contain all strings of the form $0^n 1^n$. Then $\sim_{L_2}$ has infinitely many equivalence classes.
- Let $L_3 = \{a^i b^n c^n | n \geq 0, i \geq 1\} \cup \{b^n c^m | n, m \geq 0\}$. Then $\sim_{L_3}$ has infinitely many equivalence classes.

(Proof on the board)

Myhill-Nerode Theorem

Theorem: Let $L \subseteq \Sigma^*$ be a language. Then

L is regular $\Longleftrightarrow$ $\sim_L$ has finitely many equivalence classes.

- Three specific consequences:
- $L_1 \subset \{0,1\}^*$ contains all strings where the number of 1s is divisible by 4. Then L_1 is regular.
- $L_2 \subset \{0,1\}^*$ contains all strings of the form $0^n 1^n$. Then L_2 is not regular.
- Let $L_3 = \{a^i b^n c^n | n \geq 0, i \geq 1\} \cup \{b^n c^m | n, m \geq 0\}$. Then L_3 is not regular.

Myhill-Nerode Theorem: Proof

$\Longrightarrow$

Suppose L is regular. Let $M = (Q, \Sigma, \delta, q_0, F)$ be a DFA accepting it. The relation $\sim_M$ on pairs of strings is defined

as follows: $x \sim_M y$ if $\delta(q_0, x) = \delta(q_0, y)$.
Clearly, $\sim_M$ is an equivalence relation.

Furthermore, if $x \sim_M y$, then $xz \sim_M yz$ for every $z \in \Sigma^*$.
Therefore, $xz \in L$ if and only if $yz \in L$.

This means that $x \sim_M y \Longrightarrow x \sim_L y$ (i.e., $\sim_M$ is a refinement of $\sim_L$).

Myhill-Nerode Theorem: Proof cont.

$\Longrightarrow$

- The equivalence relation $\sim_M$ has finitely many equivalence classes (at most the number of states in M).
- We saw that $x \sim_M y \Longrightarrow x \sim_L y$, so the number of equivalence classes of $\sim_L$ is less or equal than the number of equivalence classes of $\sim_M$.
- Therefore, $\sim_L$ has finitely many equivalence classes. ♠

Myhill-Nerode Theorem: Proof cont.

$\Longleftarrow$

- Suppose $\sim_L$ has finitely many equivalence classes. We'll construct a DFA $M = (Q, \Sigma, \delta, q_0, F)$ that accepts L.
- Let $x_1, \ldots, x_n \in \Sigma^*$ be representatives for the finitely many equivalence classes of $\sim_L$.
- $Q = \{[x_1], \ldots, [x_n]\}$.
- $\delta([x_i], a) = [x_i a]$ for all $a \in \Sigma$.
- $q_0 = [\varepsilon]$.
- $F = \{[x_i] \colon x_i \in L\}$.

Myhill-Nerode Theorem: Proof cont.

$\Longleftarrow$

- $\delta([\varepsilon], x) = [x]$
- proof: Assume $\delta([\varepsilon], x) = [x] = [x_i]$. By right invariance $\delta([\varepsilon], xa) = \delta([x_i], a) = [x_i a] = [xa]$
- Therefore, M accepts x iff $x \in L$
- So L is accepted by DFA, hence L is regular. ♠

Example

Construct DFA (via the above method) for $L_1 \subset \{0,1\}^*$ contains all strings where the number of 1s is divisible by 5.

Applications of the Proof

Let L be a regular language and let M be a DFA accepting it

- The number of equivalence classes of $\sim_L$ *lowerbounds* the number of equivalence classes of $\sim_M$, which equals the number of states in M.
- The equivalence relation $\sim_M$, is a refinement of $\sim_L$ (each equivalence class of $\sim_L$ correspond to a union of states).
- There is an automata whose number of states equals the number of equivalence classes of $\sim_L$

Minimizing Automata

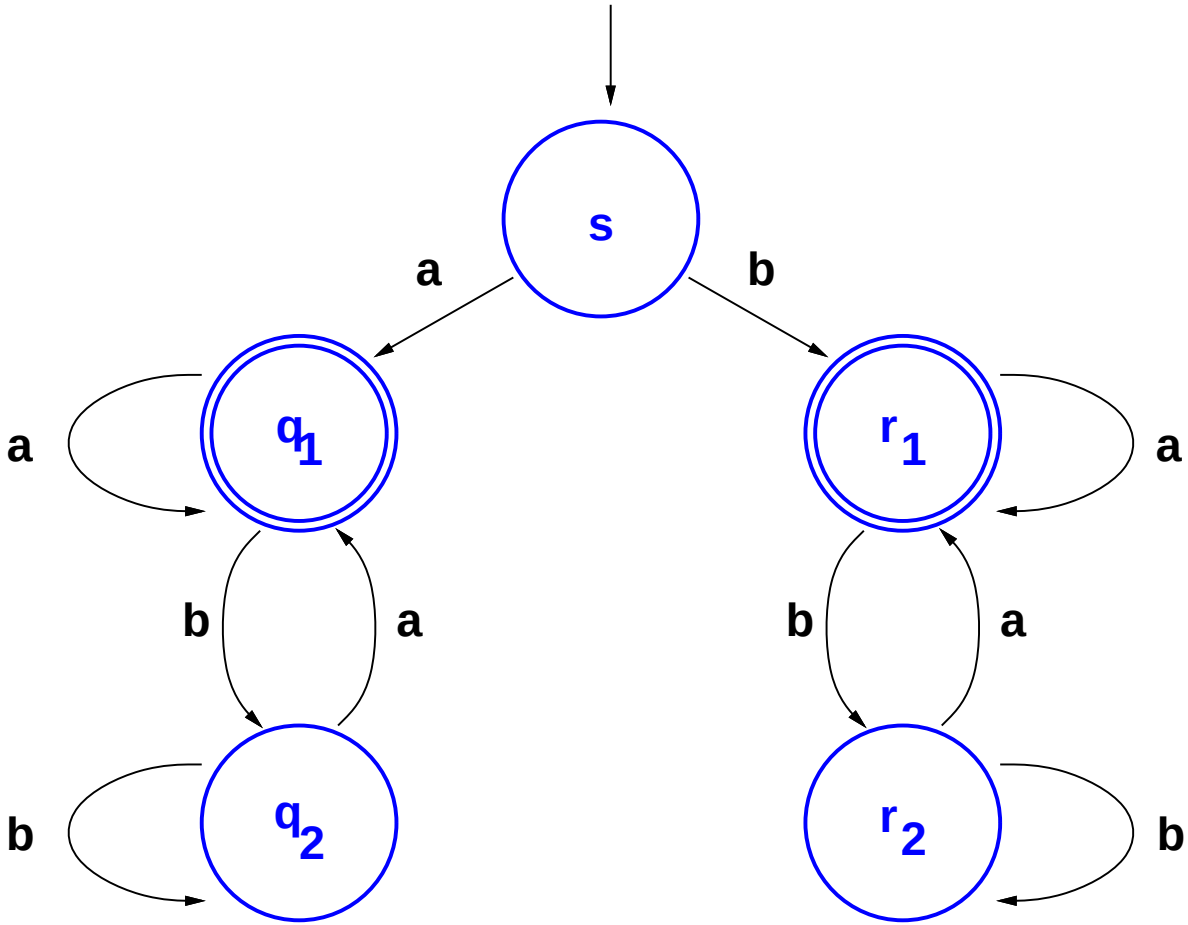
Input: An automata M
Output: An automata M', such that $L(M) = L(M')$ and M' has a minimal number of states.

- Let $S_1 = F$ and $S_2 = Q - F$. Set $\mathcal{S} = \{S_1, S_2\}$.
- While exists equivalent class $S_i \in \mathcal{S}$, $q_1, q_2 \in S_i$ and $\sigma \in \Sigma$ such that,
- $\delta(q_1, \sigma) \in S_{j_1}$ and $\delta(q_2, \sigma) \in S_{j_2}$, $j_1 \neq j_2$, then
- let $S_{i,1} = \{q \in S_i : \delta(q, \sigma) \in S_{j_1}\}$, $j_1 \neq i$.
- $\mathcal{S} = \mathcal{S} - S_i \cup S_{i,1} \cup (S_i - S_{i,1})$

Minimizing Automata

- Output $M = (Q', \delta', q_0', F')$: where
- $Q' = \mathcal{S}$,
- $q_0' = S_0 \in \mathcal{S}$ such that $q_0 \in S_0$
- $F' = \{S_1, \ldots, S_k\} \subset \mathcal{S}$, such that $S_i \subset F$.
- $\delta'(S_i, \sigma) = S_j$ if for $q \in S_i$ then $\delta(q, \sigma) \in S_j$.
- Claim: The algorithm terminates, and outputs a (in fact, the) minimal automata.

Example



Closure Properties of Regular Languages

Simple Closure Properties

- Regular languages are closed under complement.
- Let $M = (Q, \Sigma, \delta, q_0, F)$ be a DFA that accepts L.
- Then $M' = (Q, \Sigma, \delta, q_0, Q - F)$ is a DFA that accepts $\bar{L} = \Sigma^* \setminus L$.
- NFA ?!
- Regular languages are closed under intersection.
- $L_1 \cap L_2 = \overline{\overline{L_1} \cup \overline{L_2}}$.
- Proof with automata ?!

Division

$$L_1/L_2 = \{x | \exists y \in L_2, xy \in L_1\}$$

Examples:

- $L_1 = (01 + 1)^*$ and $L_2 = 00$, $L_1/L_2 = ?$
- $L_1/L_2 = \emptyset$.
- $L_3 = a^*b^*c^*$ and $L_4 = b$, $L_3/L_4 = ?$.
- $L_3/L_4 = a^*b^*$.

Division cont.

Theorem: Regular languages are closed under division with any language.
Proof:

- L_1 is a regular language, so it has a DFA $M = (Q, \Sigma, \delta, q_0, F)$.
- L_2 is an arbitrary language.
- For L_1/L_2 we build $M' = (Q, \Sigma, \delta, q_0, F')$.
- $F' = \{q | \exists y \in L_2, \delta(q, y) \in F\}$.
- F' is well defined, but might be hard to compute – “non constructive proof".

Assignments

An *assignment* substitutes each letter with a language.
Example: $f(0) = \{b\}$, $f(1) = \{a, bb\}$
$L = \{010, 10\} \implies f(L) = \{bab, bbbb, ab, bbb\}$
Theorem: Regular languages are closed under regular assignment (i.e., assignments to *regular languages*).
Proof: Let f be a regular assignment over Σ. Let $\mathcal{R}(\Sigma)$ denote all RE's over Σ, and $R(L)$ be an *arbitrary* RE for L

- Define $g : \mathcal{R}(\Sigma) \mapsto \mathcal{R}$ as
- $g(r_1 \cup r_2) = g(r_1) \cup g(r_2)$.
- $g(r_1 r_2) = g(r_1)g(r_2)$.
- $g((r_1)^*) = g(r_1)^*$.
- $g(a) = R(f(a))$, for $a \in \Sigma$
- Claim: $g(R) \in \mathcal{R}$ and $L(g(R)) = f(L(R))$, $\forall R \in \mathcal{R}(\Sigma)$

Homomorphism

- *Homomorphism:* an assignment that replaces each letter with a word
- Example: $h(1) = aba$, $h(0) = aa$
  $h(010) = aa\,aba\,aa$
  $L_1 = (01)^*$, $h(L_1) = (aaaba)^*$.
- *Inverse homomorphism:* $h^{-1}(w) = \{x | h(x) = w\}$,
  $h^{-1}(L) = \{x | h(x) \in L\}$
- Example: $L_2 = (ab + ba)^* a$, $h^{-1}(L_2) = \{1\}$.
- Claim: $h(h^{-1}(L)) \subseteq L \subseteq h^{-1}(h(L))$.

Homomorphism cont.

Theorem: Regular languages are closed under homomorphism and inverse homomorphism.
Proof:

- Homomorphism: special case of assignment.
- Inverse Homomorphism: Let $M = (Q, \Sigma, \delta, q_0, F)$ the automata for L, and $h : \Delta \to \Sigma^*$.
- Proof idea: for each letter $a \in \Delta$ we advance in M using $h(a)$.
- Formally, we define $M' = (Q, \Delta, \delta', q_0, F)$, where $\delta'(q, a) = \delta(q, h(a))$.
- Hence, $\delta'(q, w) = \delta(q, h(w))$
- $w \in L(M') \longleftrightarrow h(w) \in L(M)$

Using Homomorphism

- We know that $L_1 = \{0^n 1^n | n \geq 1\}$ is not regular.
- Show that $L_2 = \{a^n b a^n | n \geq 1\}$ is not regular
- We will prove using homomorphism and inverse homomorphism
- $h_1(a) = a$, $h_1(b) = b$, $h_1(c) = a$.
- $h_2(a) = 0$, $h_2(b) = \epsilon$, $h_2(c) = 1$.
- $h_2(h_1^{-1}(L_2) \cap a^* b^* c^*) = L_1$
- $h_1^{-1}(L_2) = (a \cup c)^k b (a \cup c)^k$
- $h_1^{-1}(L_2) \cap a^* b c^* = \{a^n b c^n | n \geq 1\}$
- $h_2(h_1^{-1}(L_2) \cap a^* b c^*) = \{0^n 1^n | n \geq 1\}$

Algorithmic Questions for NFAs

Algorithmic Questions for NFAs

Q.: Given an NFA, N, and a string w, is $w \in L(N)$?

Answer: Construct the DFA equivalent to N and run it on w.

Q.: Is $L(N) = \emptyset$?
Answer: This is a reachability question in graphs: Is there a path in the states' graph of N from the start state to some accepting state. There are simple, efficient algorithms for this task.

More Questions

Q.: Is $L(N) = \Sigma^*$?

Answer: Check if $\overline{L(N)} = \emptyset$.

Q.: Given N_1 and N_2, is $L(N_1) \subseteq L(N_2)$?

Answer: Check if $\overline{L(N_2)} \cap L(N_1) = \emptyset$.

Q.: Given N_1 and N_2, is $L(N_1) = L(N_2)$?

Answer: Check if $L(N_1) \subseteq L(N_2)$ and $L(N_2) \subseteq L(N_1)$.

In the future, we will see that for stronger models of computations, many of these problems cannot be solved by any algorithm.

Summary – Regular Languages

Summary - Regular Languages

So far we saw

- finite automata,
- regular languages,
- regular expressions,
- Myhill-Nerode theorem and pumping lemma for regular languages.

Next class we introduce stronger machines and languages with more expressive power:

- pushdown automata,
- context-free languages,
- context-free grammars