

Membership for Context-Free Grammars and the CYK Algorithm

1. The membership problem

Given a context-free grammar G and a string w , decide

$$w \in L(G) ?$$

We assume that G is in Chomsky Normal Form.

2. Chomsky Normal Form

We use the convention from Sipser. A CFG is in *Chomsky Normal Form (CNF)* if every production is of one of the forms

$$A \rightarrow BC$$

or

$$A \rightarrow a,$$

where A, B, C are variables and a is a terminal. In addition, the start variable does not occur on the right-hand side of any production.

If ε belongs to the language, the rule

$$S \rightarrow \varepsilon$$

may also be allowed, where S is the start variable.

Every context-free language has an equivalent grammar in CNF.

3. A naive algorithm

Let

$$w = w_1 w_2 \cdots w_n.$$

A CNF parse tree for a nonempty word of length n has:

- exactly n terminal leaves;
- exactly n variable nodes immediately above those leaves, using rules $A \rightarrow a$;
- exactly $n - 1$ binary branching variable nodes, using rules $A \rightarrow BC$.

Hence it has exactly

$$2n - 1$$

variable nodes.

A brute-force algorithm can:

1. enumerate all full binary tree shapes with n terminal leaves;
2. label the leaves, from left to right, by $w_1, w_2, \dots, w_n$;
3. guess variable labels for all variable nodes;

4. check whether every node corresponds to a production of G ;
5. accept if some valid tree has the start variable at its root.

The number of full binary trees with n leaves is the Catalan number C_{n-1} , and

$$C_{n-1} = O(4^n).$$

There are at most

$$|V|^{2n-1}$$

possible variable labelings. Therefore, for a fixed grammar, this is an exponential-time algorithm:

$$\boxed{O(c^n)}$$

for some grammar-dependent constant $c > 1$.

The inefficiency comes from repeatedly solving the same subproblems.

4. The CYK idea

Instead of enumerating complete parse trees, ask:

Which variables can generate a given substring of w ?

For $1 \leq i \leq j \leq n$, define

$$T[i, j] = \{A \in V : A \Rightarrow^* w_i w_{i+1} \cdots w_j\}.$$

Then

$$\boxed{w \in L(G) \iff S \in T[1, n].}$$

There are only $O(n^2)$ such substrings.

5. Base case

For a substring of length 1,

$$\boxed{A \in T[i, i] \iff A \rightarrow w_i \text{ is a production of } G.}$$

6. Recurrence

Let $i < j$. Since the grammar is in CNF, if A generates

$$w_i w_{i+1} \cdots w_j,$$

then the first production used must be of the form

$$A \rightarrow BC.$$

For some split position k with $i \leq k < j$, B generates

$$w_i \cdots w_k$$

and C generates

$$w_{k+1} \cdots w_j.$$

Thus

$$\boxed{A \in T[i, j]}$$

if and only if there exist k, B, C such that

$$i \leq k < j,$$

$$A \rightarrow BC \in P,$$

$$B \in T[i, k], \quad C \in T[k + 1, j].$$

The table is filled in increasing order of substring length.

7. Correctness

The key statement is

$$\boxed{A \in T[i, j] \iff A \Rightarrow^* w_i w_{i+1} \cdots w_j.}$$

Exercise. Prove this statement by induction on the substring length

$$j - i + 1.$$

Consequently,

$$w \in L(G) \iff S \Rightarrow^* w \iff S \in T[1, n].$$

Therefore CYK correctly decides the CFG membership problem.

8. Running time

There are $O(n^2)$ entries $T[i, j]$. For each entry there are at most $O(n)$ possible split points k . For each split, we inspect productions of the form $A \rightarrow BC$. Thus the straightforward running time is

$$\boxed{O(|P|n^3)}.$$

For a fixed grammar,

$$\boxed{O(n^3)}.$$

9. Example

Consider the grammar

$$S_0 \rightarrow AB \mid AC,$$

$$S \rightarrow AB \mid AC,$$

$$C \rightarrow SB,$$

$$A \rightarrow a, \quad B \rightarrow b.$$

Here S_0 is the start variable. This grammar is in CNF according to Sipser's convention: every rule has the form $A \rightarrow BC$ or $A \rightarrow a$, and the start variable S_0 does not occur on the right-hand side of any production.

It generates

$$\{a^n b^n : n \geq 1\}.$$

Apply CYK to

$$w = aabb.$$

Substrings of length 1

$$\begin{aligned} T[1, 1] &= \{A\}, & T[2, 2] &= \{A\}, \\ T[3, 3] &= \{B\}, & T[4, 4] &= \{B\}. \end{aligned}$$

Substrings of length 2

For aa ,

$$T[1, 2] = \emptyset.$$

For ab , since $A \in T[2, 2]$, $B \in T[3, 3]$, and

$$S \rightarrow AB, \quad S_0 \rightarrow AB,$$

we obtain

$$T[2, 3] = \{S, S_0\}.$$

For bb ,

$$T[3, 4] = \emptyset.$$

Substrings of length 3

For aab ,

$$T[1, 3] = \emptyset.$$

For abb , split after position 3. Since

$$S \in T[2, 3], \quad B \in T[4, 4],$$

and $C \rightarrow SB$, we obtain

$$T[2, 4] = \{C\}.$$

Substring of length 4

For $aabb$, split after the first symbol. Since

$$A \in T[1, 1], \quad C \in T[2, 4],$$

and $S_0 \rightarrow AC$, we obtain

$$S_0 \in T[1, 4].$$

Therefore

$$\boxed{aabb \in L(G)}.$$

The same computation can be displayed as an upper-triangular table, with i indexing rows and j indexing columns:

$i \backslash j$	1	2	3	4
1	$\{A\}$	$\emptyset$	$\emptyset$	$\{S_0, S\}$
2		$\{A\}$	$\{S, S_0\}$	$\{C\}$
3			$\{B\}$	$\emptyset$
4				$\{B\}$

Here the entry in row i and column j is exactly $T[i, j]$; entries with $i > j$ are unused.