CALCULUS EDUCATIONAL CONSULTS

MOTTO: BRINGING KNOWLEDGE TO YOUR DOORSTEP
TEL: 07037507487, 09094642770,07037302585
MAINLAND LECTURE CENTER

CIT342: FORMAL LANGUAGES AND AUTOMATA THEORY

CALCULUS EDUCATIONAL CONSULT 2019_2 SECOND SEMESTER USE ONLY

What is an Alphabets

In computer science and formal language, an alphabet or vocabulary is a finite set of symbols or letters, e.g. characters or digits. The most common alphabet is {0,1}, the binary alphabet. A finite string is a finite sequence of letters from an alphabet; for instance a binary string is a string drawn from the alphabet {0,1}. An infinite sequence of letters may be constructed from elements of an alphabet as well. For example, if we use the binary alphabet {0,1}, the strings (ε, 0, 1, 00, 01, 10, 11, 000, etc.) would all be in the Kleene closure of the alphabet (where ε represents the empty string)

What is a string

In formal languages, which are used in mathematical logic and theoretical computer science, a string is a finite sequence of symbols that are chosen from a set or alphabet. In computer programming, a string is, essentially, a sequence of characters. A string is generally understood as a data type storing a sequence of data values, usually bytes, in which elements usually stand for characters according to a character encoding, which differentiates it from the more general array data type

What is formal theory

Let Σ be an alphabet, a non-empty finite set. Elements of Σ are called symbols or characters. A string (or word) over Σ is any finite sequence of characters from Σ. For example, if Σ = {0, 1}, then 0101 is a string over Σ.
The length of a string is the number of characters in the string (the length of the sequence) and can be any non-negative integer. The empty string is the unique string over Σ of length 0, and is denoted ε or λ.
The set of all strings over Σ of length n is denoted Σ n
. For example, if Σ = {0, 1}, then
Σ 2 = {00, 01, 10, 11}. Note that Σ 0 = {ε} for any alphabet Σ.

Alphabet of a string

The alphabet of a string is a list of all of the letters that occur in a particular string. If s is a string, its alphabet is denoted by Alph(s)

What is String substitution
A string substitution or simply a substitution is a mapping f that maps letters in Σ to languages (possibly in a different alphabet). Thus, for example, given a letter a ∈ Σ , one has f(a) = La where La ⊂ Δ* is some language whose alphabet is Δ. This mapping may be extended to strings as f(ε) = ε for the empty string ε, and f(sa) = f(s)f(a)
An example of string substitution occurs in regular languages, which are closed under string substitution. That is, if the letters of a regular language are substituted by other regular languages, the result is still a regular language. Another example is the conversion of an EBCDIC-encoded string to ASCII.

Concatenation and Substrings

Concatenation is an important binary operation on Σ*. For any two strings s and t in Σ*, their concatenation is defined as the sequence of characters in s followed by the sequence of characters in t, and is denoted st. For example, if Σ = {a, b, …, z}, s = bear, and t = hug, then st = bearhug and ts = hugbear. String concatenation is an associative, but non-commutative operation. The empty string serves as the identity element; for any string s, εs = sε = s. Therefore, the set Σ* and the concatenation operation form a monoid, the free monoid generated by Σ. In addition, the length function defines a monoid homomorphism from Σ* to the nonnegative integers. A string s is said to be a substring or factor of t if there exist (possibly empty) strings u and v such that t = usv. The relation "is a substring of" defines a

partial order on Σ*, the least element of which is the empty string.

types of string length

fixed length strings which have a fixed maximum length and which use the same amount of memory whether this maximum is reached or not

variable length strings whose length is not arbitrarily fixed and which use varying amounts of memory depending on their actual size.

Briefly describe the following: i) unary representation ii) binary representation

If Σ is a unary alphabet, then the representation is said to be a unary representation

If Σ is a binary alphabet, then the representation is said to be a binary representation.

Formal Grammar

A formal grammar (sometimes simply called a grammar) is a set of rules for forming strings in a formal language. The rules describe how to form strings from the language's alphabet that are valid according to the language's syntax. A grammar does not describe the meaning of the strings or what can be done with them in whatever context – only their form

What is formal language theory

Formal language theory, the discipline which studies formal grammars and languages, is a branch of applied mathematics. Its applications are found in theoretical computer science, theoretical linguistics, formal semantics, mathematical logic, and other areas

Parsing

Parsing is the process of recognizing an utterance (a string in natural languages) by breaking it down to a set of symbols and analyzing each one against the grammar of the language.

The Syntax of Grammars

a grammar G consists of the following components:

- A finite set N of nonterminal symbols, none of which appear in strings formed from G.
- A finite set Σ of terminal symbols that is disjoint from N.
- A finite set P of production rules, each rule of the form

$$(\Sigma \cup N)^* N (\Sigma \cup N)^* \rightarrow (\Sigma \cup N)^*$$

where * is the Kleene star operator and denotes set union.

- A distinguished symbol that is the start symbol.

The Semantics of Grammars

The operation of a grammar can be defined in terms of relations on strings:

CALCULUS EDUCATIONAL CONSULT 2019_2 SECOND SEMESTER USE ONLY

- Given a grammar $G = (N,\Sigma,P,S)$, the binary relation $\Rightarrow_G$(pronounced as "G derives in one step") on strings in $(\Sigma \cup N)^*$ is defined by:

$$x \Rightarrow_G y \text{ iff } \exists u, v, p, q \in (\Sigma \cup N)^* : (x = upv) \wedge (p \to q \in P) \wedge (y = uqv)$$

- the relation $\Rightarrow_G^*$ (pronounced as *G derives in zero or more steps*) is defined as the reflexive transitive closure of $\Rightarrow_G$
- a *sentential form* is a member of $(\Sigma \cup N)^*$ that can be derived in a finite number of steps from the start symbol S; that is, a sentential form is a member of $\{w \in (\Sigma \cup N)^* \mid S \Rightarrow_G^* w\}$. A sentential form that contains no nonterminal symbols (i.e. is a member of Σ^*) is called a *sentence*.
- the *language* of G, denoted as $L(G)$, is defined as all those sentences that can be derived in a finite number of steps from the start symbol S; that is, the set $\{w \in \Sigma^* \mid S \Rightarrow_G^* w\}$.

The Chomsky Hierarchy

When Noam Chomsky first formalized generative grammars in 1956, he classified them into types now known as the Chomsky hierarchy. The difference between these types is that they have increasingly strict production rules and can express fewer formal languages. Two important types are context-free grammars (Type 2) and regular grammars (Type 3). The languages that can be described with such a grammar are called context-free languages and regular languages, respectively. Although much less powerful than unrestricted grammars (Type 0), which can in fact express any language that can be accepted by a Turing machine, these two restricted types of grammars are most often used because parsers for them can be efficiently implemented. For example, all regular languages can be recognized by a finite state machine, and for useful subsets of context-free grammars there are well-known algorithms to generate efficient LL parsers and LR parsers to recognize the corresponding languages those grammars generate.

What is Context-Free Grammars

A context-free grammar is a grammar in which the left-hand side of each production rule consists of only a single nonterminal symbol. This restriction is non-trivial; not all languages can be generated by context-free grammars. Those that can are called context-free languages. A context-free language can be recognized in O(n 3) time (see Big O notation) by an algorithm such as Earley's algorithm. That is, for every context-free language, a machine can be built that takes a string as input and determines in O(n 3) time whether the string is a member of the language, where n is the length of the string. Further, some important subsets of the context-free languages can be recognized in linear time using other algorithms.

What is Regular Grammars

In regular grammars, the left hand side is again only a single nonterminal symbol, but now the right-hand side is also restricted. The right side may be the empty string, or a single terminal symbol, or a single terminal symbol followed by a nonterminal symbol, but nothing else.

Other Forms of Generative Grammars

• **Tree-adjoining grammars** increase the expressiveness of conventional generative grammars by allowing rewrite rules to operate on parse trees instead of just strings.

• **Affix grammars and attribute grammars** allow rewrite rules to be augmented with semantic attributes and operations, useful both for increasing grammar expressiveness and for constructing practical language translation tools.

Analytic Grammars

Though there is a tremendous body of literature on parsing algorithms, most of these algorithms assume that the language to be parsed is initially described by means of a generative formal grammar, and that the goal is to

CALCOLI EDUCATIONAL CONSULTANTS 2020_2 SECOND SEM... ONLY

transform this generative grammar into a working parser. Strictly speaking, a generative grammar does not in any way correspond to the algorithm used to parse a language, and various algorithms have different restrictions on the form of production rules that are considered well-formed.

Examples of analytic grammar formalisms

• **The Language Machine** directly implements unrestricted analytic grammars. Substitution rules are used to transform an input to produce outputs and behaviour. The system can also produce the lm-diagram which shows what happens when the rules of an unrestricted analytic grammar are being applied.
• **Top-down parsing language (TDPL**): a highly minimalist analytic grammar formalism developed in the early 1970s to study the behavior of top-down parsers.
• **Link grammars**: a form of analytic grammar designed for linguistics, which derives syntactic structure by examining the positional relationships between pairs of words.
• **Parsing expression grammars** (**PEGs**): a more recent generalization of TDPL designed around the practical expressiveness needs of programming language and compiler writers.

What is Formal Language

A formal language is a set of words, i.e. finite strings of letters, or symbols. The inventory from which these letters are taken is called the alphabet over which the language is defined. A formal language is often defined by means of a formal grammar. Formal languages are a purely syntactical notion, so there is not necessarily any meaning associated with them.

What is formal language theory

The branch of mathematics and computer science that is concerned only with the purely syntactical aspects of such languages, i.e. their internal structural patterns

What is an alphabet

An alphabet, in the context of formal languages can be any set, although it often makes sense to use an alphabet in the usual sense of the word, or more generally a character set such as ASCII. Alphabets can also be infinite; e.g. first-order logic is often expressed using an alphabet which, besides symbols such as ∧, ¬, ∀ and parentheses, contains infinitely many elements x0, x1, x2, … that play the role of variables. The elements of an alphabet are called its letters.

What is a word

A word over an alphabet can be any finite sequence, or string, of letters. The set of all words over an alphabet Σ is usually denoted by Σ * (using the Kleene star).

examples of formal languages

• L = Σ *, the set of all words over Σ;
• L = {a}* = {an}, where n ranges over the natural numbers and an means "a" repeated n times (this is the set of words consisting only of the symbol "a");
• the set of syntactically correct programs in a given programming language (the syntax of which is usually defined by a context-free grammar;
• the set of inputs upon which a certain Turing machine halts; or
• the set of maximal strings of alphanumeric ASCII characters on this line, (i.e., the set {"the", "set", "of", "maximal", "strings", "alphanumeric", "ASCII", "characters", "on", "this", "line", "i", "e"}).

Language-Specification Formalisms

various types of formalisms to describe languages. For instance, a language can be given as
• those strings generated by some formal grammar (see Chomsky hierarchy);
• those strings described or matched by a particular regular expression;
• those strings accepted by some automaton, such as a Turing machine or finitestate automaton;
• those strings for which some decision procedure (an algorithm that asks a sequence of related YES/NO questions) produces the answer YES.

CAMBRIDGE EDUCATIONAL CONSULT 2019 2 SECOND SEMESTER USE ONLY

Operations on Languages

Suppose L1 and L2 are languages over some common alphabet.

- The concatenation L1L2 consists of all strings of the form vw where v is a string from L1 and w is a string from L2.
- The intersection L1 ∩ L2 of L1 and L2 consists of all strings which are contained in both languages
- The complement ¬L of a language with respect to a given alphabet consists of all strings over the alphabet that are not in the language.

Other Operations on Languages

Some other operations frequently used in the study of formal languages are the following:

- **The Kleene star**: the language consisting of all words that are concatenations of 0 or more words in the original language;
- **Reversal**: o Let e be the empty word, then e R= e, and o for each non-empty word w = x1…xn over some alphabet, let w R = xn…x1,o then for a formal language L, LR = {wR | w ∈ L}.
- **String homomorphism**.

What is Automata Theory

In theoretical computer science, automata theory is the study of abstract machines (or more appropriately, abstract 'mathematical' machines or systems) and the computational problems that can be solved using these machines. These abstract machines are called automata.

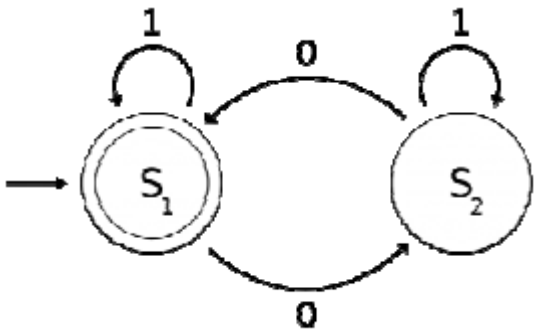


Diagram of automata

automata theory is a subject matter which studies properties of various types of automata. For example, following questions are studied about a given type of automata.

- Which class of formal languages is recognizable by some type of automata? (Recognizable languages)
- Is certain automata closed under union, intersection, or complementation of formal languages? (Closure properties)
- How much is a type of automata expressive in terms of recognizing class of formal languages? And, their relative expressive power? (Language Hierarchy)

Informal Description of Automaton

An automaton is supposed to run on some given sequence or string of inputs in discrete time steps. At each time step, an automaton gets one input that is picked up from a set of symbols or letters, which is called an alphabet. At any time, the symbols so far fed to the automaton as input form a finite sequence of symbols, which is called a word. An automaton contains a finite set of states. At each instance in time of some run, automaton is in one of its states. At each time step when the automaton reads a symbol, it jumps or transits to next state depending on its current state and on the symbol currently read. This function in terms of the current state and input symbol is called transition function. The automaton reads the input word one symbol after another in the sequence and transits from state to state according to the transition function, until the word is read completely. Once the input word has been read, the automaton is said to have been stopped and the state at which automaton has stopped is called final state. Depending on the final state, it is said that the automaton either accepts or rejects an input word. There is a subset of states of the automaton, which is defined as the set of accepting states. If the final state is an accepting state, then

CALCULUS EDUCATIONAL ... JLT 2019_2 SECOND SEMESTER USE ONLY

the automaton accepts the word. Otherwise, the word is rejected.

Formal Definitions of automaton

An automaton is represented formally by the 5-tuple $\langle Q, \Sigma, \delta, q0, F \rangle$, where:

- Q is a finite set of states.
- Σ is a finite set of symbols, called the alphabet of the automaton.
- δ is the transition function, that is, $\delta: Q \times \Sigma \rightarrow Q$.
- q0 is the start state, that is, the state which the automaton is in when no input has been processed yet, where $q0 \in Q$.
- F is a set of states of Q (i.e. $F \subseteq Q$) called accept states.

Variations in Definition of Automata

The following are some popular variations in the definition of different components of automata.

Input

- **Finite input**: An automaton that accepts only finite sequence of words. The above introductory definition only accepts finite words.
- **Infinite input**: An automaton that accepts infinite words (ω-words). Such automata are called ω-automata.
- **Tree word input**: The input may be a tree of symbols instead of sequence of symbols.

States

- **Finite states**: An automaton that contains only a finite number of states. The above introductory definition describes automata with finite numbers of states.
- **Infinite states**: An automaton that may not have a finite number of states, or even a countable number of states. For example, the quantum finite automaton or topological automaton has uncountable infinity of states.
- **Stack memory**: An automaton may also contain some extra memory in the form of a stack in which symbols can be pushed and popped. This kind of automaton is called a pushdown automaton

Transition Function

- **Deterministic**: For a given current state and an input symbol, if an automaton can only jump to one and only one state then it is a deterministic automaton.
- **Nondeterministic**: An automaton that, after reading an input symbol, may jump into any of a number of states, as licensed by its transition relation. Notice that the term transition function is replaced by transition relation:
- **Alternation**: This idea is quite similar to tree automaton, but orthogonal. The automaton may run its multiple copies on the same next read symbol.

Acceptance Condition

- **Acceptance of finite words**: Same as described in the informal definition above.
- **Acceptance of infinite words**: an omega automaton cannot have final states, as infinite words never terminate. Rather, acceptance of the word is decided by looking at the infinite sequence of visited states during the run.
- **Probabilistic acceptance**: An automaton need not strictly accept or reject an Input

Classes of automata

Types of Automata

CALCULUS EDUCATIONAL CONSULT 2019_2 SECOND SEMESTER USE ONLY

Automata	Recognizable language
Deterministic finite automata (DFA)	regular languages
Nondeterministic finite automata (NFA)	regular languages
Nondeterministic finite automata with ε transitions (FND-ε or ε-NFA)	regular languages
Pushdown automata (PDA)	context-free languages
Linear bounded automata (LBA)	context-sensitive language
Turing machines	Recursively enumerable languages
Timed automata	
Deterministic Büchi automata	omega limit languages
Nondeterministic Büchi automata	omega regular languages
Nondeterministic/Deterministic Rabin automata	omega regular languages
Nondeterministic/Deterministic Streett automata	omega regular languages
Nondeterministic/Deterministic parity automata	omega regular languages
Nondeterministic/Deterministic Muller automata	omega regular languages

CALCULUS EDUCATION CONSULT 2019_2 SECOND ... ER USE ONLY

Applications of Automata Theory

Each model in automata theory play varied roles in several applied areas. Finite automata are used in text processing, compilers, and hardware design. Context-free grammar (CFG) is used in programming languages and artificial intelligence. Originally, CFG were used in the study of the human languages. Cellular automata are used in the field of biology, the most common example being John Conway's Game of Life. Some other examples which could be explained using automata theory in biology include mollusk and pine cones growth and pigmentation patterns. Going further, Stephen Wolfram claims that the entire universe could be explained by machines with a finite set of states and rules with a single initial condition. Other areas of interest which he has related to automata theory include: fluid flow, snowflake and crystal formation, chaos theory, cosmology, and financial analysis.

What is Finite State Automata

An FSA is a virtual device that manipulates a candidate string, one character at a time, and determines whether that string is in the language implemented by the machine. The simplest state machine reads the string exactly once, and has no memory, only registers. It is therefore a finite state automaton. An FSA is defined by a set of states and a transition function that maps state/input pairs into states. In this state, reading this character, move to that state and advance to the next character

Deterministic Finite Acceptors/Automata (DFA)

DFAs are:

- Deterministic i.e. there is no element of choice
- Finite i.e. only a finite number of states and arcs
- Acceptors i.e. produce only a yes/no answer

Algorithm for the Operation of a DFA

- Start with the "current state" set to the start state and a "read head" at the beginning of the input string;
- while there are still characters in the string:

o Read the next character and advance the read head;

o From the current state, follow the arc that is labelled with the characterjust read; the state that the arc points to becomes the next current state;

- When all characters have been read, accept the string if the current state is a final state, otherwise reject the string.

Ways of Implementing a DFA

Using a GO TO Statement
Using a CASE Statement

Formal Definition of a DFA

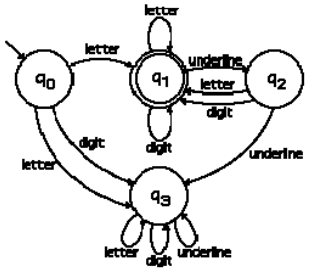
A deterministic finite acceptor/automaton or DFA is a quintuple:
$M = (Q, \Sigma, \delta, q0, F)$

Where

- Q is a finite set of *states,*
- Σ is a finite set of symbols, the *input alphabet*
- δ: $Q \times \Sigma \rightarrow Q$ is a *transition function,*
- $q0 \in Q$ is the *initial state,*
- $F \subseteq Q$ is a set of *final states.*

Acceptor for Ada identifiers

In Ada, an identifier consists of a letter followed by any number of letters, digits, and underlines. However, the identifier may not end in an underline or have two underlines in a row
an automaton to recognize Ada identifiers



Nondeterministic Finite Automata/Acceptors (NFA)

An FSA is nondeterministic if it is confronted with several choices when processing each character. Thus the transition function of a nondeterministic FSA maps state/input pairs into sets of states. The machine somehow traverses all possible paths in parallel. In addition, E transitions are permitted, allowing the machine to change states without reading an input character. A string is accepted by an NFA if one of its parallel transition sequences leads to a final state.

If you think of an automaton as a computer, how does it handle nondeterminism?

1. When the automaton is faced with a choice, it always (magically) chooses correctly. We sometimes think of the automaton as consulting an oracle which advises it as to the correct choice.
2. When the automaton is faced with a choice, it spawns a new process, so that all possible paths are followed simultaneously.

Ways of Implementing an NFA

There are three ways, two feasible and one not yet feasible, to simulate the second alternative:
1. Use a recursive backtracking algorithm. Whenever the automaton has to make a choice, cycle through all the alternatives and make a recursive call to determine whether any of the alternatives leads to a solution (final state).
2. Maintain a state set or a state vector, keeping track of all the states that the NFA could be in at any given point in the string.
3. Use a quantum computer. Quantum computers explore literally all possibilities simultaneously. They are theoretically possible, but are at the cutting edge of physics. It may (or may not) be feasible to build such a device.

State-Set Implementation of NFA

Another way to implement an NFA is to keep either a state set or a bit vector of all the states that the NFA could be in

CALCULUS EDUCATIONAL CONSULT 2019_2 SECOND SEMESTER USE ONLY

at any given time. Implementation is easier if you use a bit-vector approach (v[i] is True iff state i is a possible state), since most languages provide vectors, but not sets, as a built-in datatype

Formal Definition of NFAs

A nondeterministic finite acceptor/automaton or NFA is defined by the quintuple: $M = (Q, \Sigma, \delta, q0, F)$

Where

- Q is a finite set of *states,*
- Σ is a finite set of symbols, the *input alphabet,*
- δ: $Q \times (\Sigma \cup \{\lambda\}) \rightarrow 2^Q$ is a *transition function,*
- $q0 \in Q$ is the *initial state,*
- $F \subseteq Q$ is a set of *final states.*

Is an NFA more powerful than a DFA? Discuss.

or any NFA, we can construct an equivalent DFA So NFAs are not more powerful than DFAs. DFAs and NFAs define the same class of languages – the regular languages.

How is a DFA different from an NFA? How are they similar?

A DFA is just a special case of an NFA that happens not to have any null transitions or multiple transitions on the same symbol. So DFAs are not more powerful than NFAs.

What is a regular expression

A regular expression can be used to define a language. A regular expression represents a "pattern;" strings that match the pattern that are in the language, strings that do not match the pattern are not in the language.

the languages defined by the primitive regular expressions:

- For each $x \in \Sigma$, the primitive regular expression x denotes the language {x}. That is, the only string in the language is the string "x".
- The primitive regular expression λ denotes the language $\{\lambda\}$. The only string in this language is the empty string.
- The primitive regular expression $\varnothing$ denotes the language {}. There are *no* strings in this language.

Regular Expressions

We can compose additional regular expressions by applying the following rules a finite number of times:

• If r1 is a regular expression, then so is (r1).
• If r1 is a regular expression, then so is r1*.
• If r1 and r2 are regular expressions, then so is r1r2 (Concatenation)
• If r1 and r2 are regular expressions, then so is r1+r2 or r1/r2 (Union)

Building Regular Expressions

some hints on building regular expressions.

- Zero or more.
- One or more.
- Zero or one.
- Any string at all.
- Any nonempty string.
- Any string not containing..
- Any string containing exactly one...

CALCULUS EDUCATION 2019_2 SECOND SEMESTER USE ONLY

Three Ways of Defining a Language

Definition by Grammar
Definition by NFA
Definition by Regular Expression

three ways of characterising regular languages

regular expressions
finite automata and
construction from simple languages using simple operations

Distinguish between right-linear grammar and left-linear grammar

To be a right-linear grammar, *every* production of the grammar must have one of the two forms $V \rightarrow T^*V$ or $V \rightarrow T^*$.

To be a left-linear grammar, *every* production of the grammar must have one of the two forms $V \rightarrow VT^*$ or $V \rightarrow T^*$.

---

closure properties of regular languages

Closure I
Closure II: Union, Concatenation, Negation, Kleene Star, Reverse
Closure III: Intersection and Set Difference
Closure IV: Homomorphism
Closure V: Right Quotient

What is a pigeonhole

A pigeonhole is a hole or small recess for pigeons to nest.it is a small open compartment (as in a desk or cabinet) for keeping letters or documents.it is a neat category which usually fails to reflect actual complexities.

Pigeonhole Principle

If n objects are put into m containers, where n > m, then at least one container must hold more than one object. The pigeonhole can be used to prove that certain infinite languages are not regular. (Remember, any finite language is regular.)

The Pumping Lemma

Here is what the pumping lemma says:

- If an infinite language is regular, it can be defined by a DFA.
- The DFA has some finite number of states (say, n).
- Since the language is infinite, some strings of the language must have length >n.
- For a string of length > n accepted by the DFA, the walk through the DFA must contain a cycle.
- Repeating the cycle an arbitrary number of times must yield another string accepted by the DFA.

what is a Context-Free Grammars (CFG)

A context-free grammar is a grammar G= (X, T, S, P) for which all rules, or productions, in P have the special form $A \Rightarrow \alpha$, for $A \in X - T$ and $\alpha \in X^*$.

Syntax Trees

Tree representations of derivations, also known as syntax trees, were briefly introduced in the preceding unit to promote intuition of derivations. Since these are such important tools for the investigation of context-free languages, they will be dealt with a little more systematically here

Definition of Syntax Tree

Let G = (X, T, P, S) be a context-free grammar. A syntax tree for this grammar consists of one of the following:
1) A single node x for an $x \in T$. This x is both root and leaf node.
2) An edge

CALCULUS EDUCATIONAL CONSULT 2019_2 SECOND SEMESTER USE ONLY

3) A tree

Ambiguous Grammars

A grammar G is ambiguous if there exists some string w ∈L(G) for which
- there are two or more distinct derivation trees, or
- there are two or more distinct leftmost derivations, or
- there are two or more distinct rightmost derivations.

Ambuity of grammars

Ambiguity is a property of a grammar, and it is usually (but not always) possible to find an equivalent unambiguous grammar

What is Ambiguous Languages

An inherently ambiguous language is a language for which no unambiguous grammar exists.

Definition of Chomsky Normal Form

The context-free Grammar G = (X, T, P, S) is said to be in Chomsky normal form if all grammar rules have the form

$$A \Rightarrow a \mid BC, \qquad (1)$$

for $a \in T$ and $B, C \in X - T$. There is one exception. If $\lambda \in L(G)$, then the single extra rule

$$S \Rightarrow \lambda \qquad (2)$$

is permitted. If $\lambda \notin L(G)$ then production rule 2 is not allowed.

ways to use a grammar

- Use the grammar to generate strings of the language. This is easy -- start with the start symbol, and apply derivation steps until you get a string composed entirely of terminals.
- Use the grammar to recognize strings; that is, test whether they belong to the language. For CFGs, this is usually much harder.

What is a Pushdown Automata(PDAs)

This machine is fed input just as a finite automaton is. Some texts speak of the input coming in on a read-only tape with a tape head that moves left to right until it comes to the end of the input. At that point it reads a special character that marks a blank tape cell. We will use the character or λ as a "blank". When the tape head reads a blank the machine halts, or begins the process of halting (which will be explained later.) The tape head may not reverse directions, nor may it be used to write to the tape. The input tape is infinitely long in the rightward direction, which allows a PDA to accept a finite input of any length. The cells of the tape are numbered, with the first input character occurring in cell 1 where the tape head is set initially

Types of PDAs(Push down automata)

- A deterministic PDA (DPDA) is one in which every input string has a unique path through the machine.
- A nondeterministic PDA (NPDA) is one in which we may have to choose among several paths for an input string.

Formal definition of PDA

Formally, a PDA is defined as a collection of seven things:
- an alphabet of input letters
- an input tape containing an input string followed by
- an alphabet of stack letters
- a pushdown stack, initially empty
- one start state
- a set of accept states
- a transition function

CALOOUS EDUCATIONAL CONSULT 2019_2 SECOND SEMESTER USE ONLY

What is a Nondeterministic Pushdown Automata (NPDA)

a nondeterministic pushdown automaton (NPDA) is basically an NFA with a stack added to it.

Formal definition of NPDA

A nondeterministic pushdown automaton or NPDA is, therefore, a 7-tuple $M = (Q, \Sigma, \Gamma, \delta, q_0, z, F)$
Where

- Q is a finite set of *states,*
- Σ is a the *input alphabet,*
- Γ is the *stack alphabet,*
- δ is a *transition function,*
- $q_0 \in Q$ is the *initial state,*
- $z \in \Gamma$ is the *stack start symbol,* and
- $F \subseteq Q$ is a set of *final states.*

A nondeterministic finite acceptor differs from a deterministic finite acceptor in two ways:

• The transition function is single-valued for a DFA, multi-valued for an NFA.
• An NFA may have -transitions.

A nondeterministic pushdown automaton differs from a deterministic pushdown automaton (DPDA) in almost the same ways:

• The transition function is at most single-valued for a DPDA, multi-valued foran NPDA.
• Both NPDAs and DPDAs may have -transitions; but a DPDA may have a -transition only if no other transition is possible.

UNIT 2019_2 SECOND SEMESTER USE ONLY

Algorithm to construct a PDA for a CFG

Output: a PDA P such that $N(P) = L(G)$.

Method: Let $P = (\{q\}, T, V \cup T, \delta, q, S)$ where:

1. $\delta(q, \varepsilon, A) = \{(q, \beta) \mid A \rightarrow \beta \text{ is in Q } \}$ for each nonterminal A in V.
2. $\delta(q, a, a) = \{(q, \varepsilon)\}$ for each terminal a in T.

For a given input string w, the PDA simulates a leftmost derivation for w in G.

We can prove that $N(P) = L(G)$ by showing that w is in $N(P)$ iff w is in $L(G)$:

- If part: If w is in $L(G)$, then there is a leftmost derivation
- $S = \gamma_1 \Rightarrow \gamma_2 \Rightarrow \ldots \Rightarrow \gamma_n = w$
  We show by induction on i that P simulates this leftmost derivation by the sequence of moves
  $(q, w, S) \vdash^* (q, y_i, \alpha_i)$
  such that if $\gamma_i = x_i\alpha_i$, then $x_iy_i = w$.

- Only-if part: If $(q, x, A) \vdash^* (q, \varepsilon, \varepsilon)$, then $A \Rightarrow^* x$.

  We can prove this statement by induction on the number of moves made by P.

Algorithm to construct a CFG for a PDA

Input: a PDA P = (Q, Σ, Γ, δ, q0, Z0, F).

Output: a CFG G = (V, Σ, R, S) such that L(G) = N(P).
Method:

1. Let the nonterminal S be the start symbol of G. The other nonterminals in V will be symbols of the form $[pXq]$ where p and q are states in Q, and X is a stack symbol in Γ.
2. The set of productions R is constructed as follows:
   - For all states p, R has the production $S \rightarrow [q_0 Z_0 p]$.
   - If $\delta(q, a, X)$ contains $(r, Y_1 Y_2 \ldots Y_k)$, then R has the productions

     $[qXr_k] \rightarrow a[rY_1r_1] [r_1Y_2r_2] \ldots [r_{k-1}Y_kr_k]$
     for all lists of states $r_1, r_2, \ldots, r_k$.

We can prove that $[qXp] \Rightarrow^* w$ iff $(q, w, X) \vdash^* (p, \varepsilon, \varepsilon)$.

From this, we have $[q_0 Z_0 p] \Rightarrow^* w$ iff $(q_0, w, Z_0) \vdash^* (p, \varepsilon, \varepsilon)$, so we can conclude $L(G) = N(P)$.

What is aTuring Machines

A Turing machine (TM) is a generalization of a PDA which uses a tape instead of a stack. Turing machines are an abstract version of a computer - they have been used to define formally what is computable. There are a number of alternative approaches to formalize the concept of computability (e.g. called the λ-calculus, or μ-recursive functions, ...) but they can all be shown to be equivalent. That this is the case for any reasonable notion of computation is called the Church-Turing Thesis.

Formal definition of Turing machine?

A Turing machine M = (Q, Σ, Γ, δ, q0, B, F) is given by the following data
• A finite set Q of states,
• A finite set Σ of symbols (the alphabet),
• A finite set Γ of tape symbols s.t. Σ ⊆ Γ. This is the case because we use the tape also for the input.
• A transition function: δ ∈ Q× Γ → {stop**} U Q × Γ × {L,R}**

What is The Halting Problem

The Halting Problem says that no computer program or Turing machine can determine if ALL computer programs or Turing machines will halt or not halt on ALL inputs. To prove the Halting Problem is unsolvable we will construct one program and one input for which there is no computer program or Turing machine.

The powerful mathematical concepts that do the proofs for both a computer program and a Turing machine. The mathematical concepts we need are:
Proof by contradiction: Assume a statement is true, show that the assumption leads to a contradiction. Thus the statement is proven false.
Self referral: Have a computer program or a Turing machine operate on itself, well, a copy of itself, as input data.
Logical negation: Take a black box that accepts input and outputs true or false, put that black box in a bigger black box that switches the output so it is false or true respectively.

Godel Incompleteness Theorem

Any formal system powerful enough to express arithmetic must have true theorems that cannot be proven within the formal system.
Basically Godel proved that when trying to add axioms to a formal system in order to prove all true theorems within the formal system, eventually the system will become inconsistent before it becomes complete.
A complete formal system is a formal system where all true theorems can be proved.
An inconsistent formal system is a formal system where at least one false statement can be proved within the formal system.

CALCULUS EDUCATIONAL CONSULT 2019 2 SECOND SEMESTER USE ONLY

Unrestricted Grammars

The productions of an unrestricted grammar have the form:

$$(V \cup T)+ \rightarrow (V \cup T)^*$$

The other grammar types we have considered (left linear, right linear, linear, context free) restrict the form of productions in one way or another. An unrestricted grammar does not.

Define the following terms:

· Strings

In formal languages, which are used in mathematical logic and theoretical computer science, a string is a finite sequence of symbols that are chosen from a set or alphabet.

In computer programming, a string is, essentially, a sequence of characters. A string is generally understood as a data type storing a sequence of data values, usually bytes, in which elements usually stand for characters according to a character encoding, which differentiates it from the more general array data type. In this context, the terms binary string and byte string are used to suggest strings in which the stored data does not (necessarily) represent text.

· Alphabets

In computer science and formal language, an alphabet or vocabulary is a finite set of symbols or letters, e.g. characters or digits. The most common alphabet is {0,1}, the binary alphabet. A finite string is a finite sequence of letters from an alphabet; for instance a binary string is a string drawn from the alphabet {0,1}. An infinite sequence of letters may be constructed from elements of an alphabet as well.

Given an alphabet Σ, we write Σ* to denote the set of all finite strings over the alphabet Σ. Here, the * denotes the Kleene star operator. We write (or occasionally, or Σω) to denote the set of all infinite sequences over the alphabet Σ.

For example, if we use the binary alphabet {0,1}, the strings (ε, 0, 1, 00, 01, 10, 11, 000, etc.) would all be in the Kleene closure of the alphabet (where ε represents the empty string)

Briefly describe the following:

i) unary representation

If Σ is a unary alphabet, then the representation is said to be a unary representation

ii) binary representation

Σ is a binary alphabet, then the representation is said to be a binary representation.

What you understand by grammars.

A grammar mainly consists of a set of rules for transforming strings. (If it only consisted of these rules, it would be a semi-Thue system.) To generate a string in the language, one begins with a string consisting of only a single start symbol. The production rules are then applied in any order, until a string that contains neither the start symbol nor designated nonterminal symbols is produced. The language formed by the grammar consists of all distinct strings that can be generated in this manner. Any particular sequence of production rules on the start symbol yields a distinct string in the language. If there are multiple ways of generating the same single string, the grammar is said to be ambiguous.

Distinguish among the following grammar types:

· Regular grammars

In regular grammars, the left hand side is again only a single nonterminal symbol, but now the right-hand side is also restricted. The right side may be the empty string, or a single terminal symbol, or a single terminal symbol followed by a nonterminal symbol, but nothing else. (Sometimes a broader definition is used: one can allow longer strings of terminals or single nonterminals without anything else, making languages easier to denote while still defining the same class of languages.)

The language defined above is not regular, but the language {anbm | m, n ≥ 1} (at least 1 a followed by at least 1 b, where the numbers may be different) is, as it can be defined by the grammar G3 with N = {S, A, B}, Σ = {a, b}, S the start symbol

CALCUTTA EDUCATIONAL CONSULT 2019 - SECOND SEMESTER USE ONLY

· Context-free grammar

A context-free grammar is a grammar in which the left-hand side of each productionrule consists of only a single nonterminal symbol. This restriction is non-trivial; notall languages can be generated by context-free grammars. Those that can are calledcontext-free languages.

The language defined above is not a context-free language, and this can be strictly proven using the pumping lemma for context-free languages, but for example the language L(G) = {anbn|n ≥ 1} (at least 1 a followed by the same number of b's) is context-free, as it can be defined by the grammar G2 with N = {S}, Σ = {a, b}, S the start symbol

· Analytical grammars

Though there is a tremendous body of literature on parsing algorithms, most of these algorithms assume that the language to be parsed is initially described by means of a generative formal grammar, and that the goal is to transform this generative grammar into a working parser. Strictly speaking, a generative grammar does not in any way correspond to the algorithm used to parse a language, and various algorithms have different restrictions on the form of production rules that are considered well-formed.

An alternative approach is to formalize the language in terms of an analytic grammar in the first place, which more directly corresponds to the structure and semantics of a parser for the language.

Briefly discuss the Chomsky hierarchy. What is the relationship among the various types of grammars described in the Chomsky hierarchy?

When Noam Chomsky first formalized generative grammars in 1956, he classified them into types now known as the Chomsky hierarchy. The difference between these types is that they have increasingly strict production rules and can express fewer formal languages. Two important types are context-free grammars (Type 2) and regular grammars (Type 3). The languages that can be described with such a grammar are called context-free languages and regular languages, respectively. Although much less powerful than unrestricted grammars (Type 0), which can in fact express any language that can be accepted by a Turing machine, these two restricted types ofgrammars are most often used because parsers for them can be efficientlyimplemented. For example, all regular languages can be recognized by a finite state machine, and for useful subsets of context-free grammars there are well-known algorithms to generate efficient LL parsers and LR parsers to recognize the corresponding languages those grammars generate.

What you understand by formal languages.

A formal language is a set of words, i.e. finite strings of letters, or symbols. The inventory from which these letters are taken is called the alphabet over which the language is defined. A formal language is often defined by means of a formal grammar. Formal languages are a purely syntactical notion, so there is not necessarily any meaning associated with them. To distinguish the words that belong to a language from arbitrary words over its alphabet, the former are sometimes called well-formed words (or, in their application in logic, well-formed formulas).

Formal languages are studied in the fields of logic, computer science and linguistics. Their most important practical application is for the precise definition of syntactically correct programs for a programming language. The branch of mathematics and computer science that is concerned only with the purely syntactical aspects of such languages, i.e. their internal structural patterns, is known as formal language theory.

What do you understand by automata theory?

In theoretical computer science, automata theory is the study of abstract machines (or more appropriately, abstract 'mathematical' machines or systems) and the computational problems that can be solved using these machines. These abstract machines are called automata.

Automata theory is also closely related to formal language theory, as the automata are often classified by the class of formal languages they are able to recognize. An automaton can be a finite representation of a formal language that may be an infinite set.

In the context of automata theory, briefly describe the following terms:

· Recognised language

An automaton can recognize a formal language. The recognized language L ⊂ Σ* by an automaton is the set of all the words that are accepted by the automaton.

PAKULUS EDUCATIONAL CONSULT 2019 - SECOND SEMESTER USE ONLY

· Recognizable languages

The recognizable languages is the set of languages that are recognized by some automaton. For above definition of automata, the recognizable languages are regular languages. For different definitions of automata, the recognizable languages are different.

· Run

A run of the automaton on an input word w = a1, a2,...., an ∈ Σ*, is a sequenceof states q0, q1, q2,...., qn, where qi ∈ Q such that q0 is a start state and qi = δ(qi- 1,ai) for 0 <i ≤ n. In words, at first the automaton is at the start state q0 and then automaton reads symbols of the input word in sequence. When automaton reads symbol ai then it jumps to state qi = δ(qi-1,ai). qn said to be the final state of the run.

Define context-free grammars

A context-free grammar is a grammar G= (X, T, S, P) for which all rules, or productions, in P have the special form A ⇒ a, for A ∈ X - T and a∈X*. Additionally, for any two strings u, v ∈X* write u ⇒v (u directly produces v) if and only if

(1) u = u1Au2 for u1, u2 ∈X* and A ∈X - T and

(2) v = v1av2 and A ⇒a, a∈X*,

is a production from P.

The reduction u ⇒v is also called a direct production. Finally, write u ⇒v for two strings u, v ∈X* (u derives v) if there is a sequence u = u0 ⇒ u1 ⇒ u2 ⇒... ⇒un = v of direct productions ui⇒ui+1 from R. The length of the derivation is n. The language generated by G is {x ∈T*| S ⇒*x}.

Thus, the definition just articulates the reduction of A to a in any context in which A occurs. The productions for a context-free grammar are a restricted form of the productions allowed for a general grammar. Thus, a context-free grammar is a grammar.

Briefly explain the concept of ambiguity in grammars

Since derivation trees, leftmost derivations, and rightmost derivations are equivalentnotations, the following definitions are equivalent:

A grammar G is ambiguous if there exists some string w ∈L(G) for which

· there are two or more distinct derivation trees, or

· there are two or more distinct leftmost derivations, or

· there are two or more distinct rightmost derivations.

Grammars are used in compiler construction. Ambiguous grammars are undesirable because the derivation tree provides considerable information about the semantics of a program; conflicting derivation trees provide conflicting information.

Ambiguity is a property of a grammar, and it is usually (but not always) possible to find an equivalent unambiguous grammar.

An inherently ambiguous language is a language for which no unambiguous grammar exists.

From a PDA to an equivalent CFG

Given a PDA P, we can construct a CFG G such that L(G) = N(P). The basic idea of the proof is to generate the strings that cause P to go from state q to state p, popping a symbol X off the stack, by a nonterminal of the form [qXp].

Algorithm to construct a CFG for a PDA

Input: a PDA P = (Q, Σ, Γ, δ, q0, Z0, F).

Output: a CFG G = (V, Σ, R, S) such that L(G) = N(P).

Method:

1. Let the nonterminal S be the start symbol of G. The other nonterminals in V will be symbols of the form [pXq] where p and q are states in Q, and X is a stack symbol in Γ.

2. The set of productions R is constructed as follows:

For all states p, R has the production S → [q0Z0p]. If δ(q, a, X) contains (r, Y1Y2 … Yk), then R has the productions [qXrk] → a[rY1r1] [r1Y2r2] … [rk-1Ykrk] for all lists of states r1, r2, … , rk. We can prove that [qXp] ⇒* w iff (q, w, X) |–* (p, ε, ε). From this, we have [q0Z0p] ⇒* w iff (q0, w, Z0) |–* (p, ε, ε), so we can conclude L(G) = N(P).

COLLEGES EDUCATIONAL CONSULT 2019_2 SECOND SEMESTER USE ONLY

From CFG to NPDA

For any context-free grammar in Greibach Normal Form we can build an equivalent nondeterministic pushdown automaton. This establishes that an NPDA is at least as powerful as a CFG.

Key idea: Any string of a context-free language has a leftmost derivation. We set up the NPDA so that the stack contents "correspond" to this sentential form; every move of the NPDA represents one derivation step.

The sentential form is:

· the characters already read,

· PLUS the symbols on the stack

· MINUS the final z (the initial stack symbol).

Define Turing machine

A Turing machine (TM) is a generalization of a PDA which uses a tape instead of a stack. Turing machines are an abstract version of a computer - they have been used todefine formally what is computable. There are a number of alternative approaches toformalize the concept of computability (e.g. called the l-calculus, or m-recursive functions, ...) but they can all be shown to be equivalent.

A Turing machine M = (Q, S, G, d, q0, B, F) is given by the following data

· A finite set Q of states,

· A finite set S of symbols (the alphabet),

· A finite set G of tape symbols s.t. S Í G. This is the case because we use the tape also for the input.

· A transition function: d ∈ Q× G → {stop} ∪ Q × G × {L,R}

Define unrestricted grammars

The productions of an unrestricted grammar have the form:

(V T)+ (V T) *

The other grammar types we have considered (left linear, right linear, linear, context free) restrict the form of productions in one way or another. An unrestricted grammar does not.

In what follows, we will attempt to show that unrestricted grammars are equivalent to Turing machines. Bear in mind that:

· A language is recursively enumerable if there exists a Turing machine that accepts every string of the language, and does not accept strings that are not in the language.

· "Does not accept" is not the same as "reject" – the Turing machine could go into an infinite loop instead, and never get around to either accepting or rejecting the string.

Our plan of attack is to show that the languages generated by unrestricted grammars are precisely the recursively enumerable languages.

Prove that any language generated by an unrestricted grammar is recursively enumerable.

This can be proven as follows:

1. If a procedure exists for enumerating the strings of a language, then the language is recursively enumerable.
2. There exists a procedure for enumerating all the strings in any language generated by an unrestricted grammar.
3. Therefore, any language generated by an unrestricted grammar is recursively enumerable.

Here is a review of the argument for (1) above. We prove the language is recursively enumerable by constructing a Turing machine to accept any string w of the language.

· Build one Turing machine that generates the strings of the language in some systematic order.

· Build a second Turing machine that compares its input to w and accepts its input if the two strings are identical.

· Build a composite Turing machine that incorporates the two machines above, using the output of the first as input to the second.

State Godel incompleteness theorem

Any formal system powerful enough to express arithmetic must have true theorems that cannot be proven within the formal system.Basically Godel proved that when trying to add axioms to a formal system in order to prove all true

CAMBRIDGE EDUCATIONAL CONSULT 2019_2 SECOND SEMESTER USE ONLY

theorems within the formal system, eventually the system will become inconsistent before it becomes complete.
A complete formal system is a formal system where all true theorems can be proved.
An inconsistent formal system is a formal system where at least one false statement can be proved within the formal system.
Due to the computational equivalence of formal systems to other computational capability, we get the Halting problem, the uncomputable numbers and other unsolvable problems.

What do you understand by halting problem?

Assumption: There exists a Turing machine, TMh, such that: When the input tape contains the encoding of a Turing machine, TMj followed by input data k, TMh accepts if TMj halts with input k and TMh rejects ifTMj is not a Turing machine or TMj does not halt with input k.

What do you understand by decision problems?

Decision problems are stated as questions where the answer is binary, 0 or 1, False or True, No or yes, Reject or Accept and so forth.
Generally a decision problem states a problem and gives a candidate solution, asking if the solution solves the problem.
Examples:
Given the math expression 2+2 is the answer 4?
Given a formal language and a string, is the string in the language?
Given a grammar and a string, is the string accepted by the grammar?

What does it mean to say a formally stated problem is

a) Unsolvable

A formally stated problem is Unsolvable if no Turing machine exists to compute the solution. A formally stated problem is provably unsolvable if it can be proved no Turing machine exists to compute the solution.

b) undecidable

A formally stated problem is Undecidable if no total recursive function and thus, no Turing machine that always halts can be constructed to decide the problem, usually true or false.
Let us fix a simple alphabet S = {0,1}. As computer scientists we are well aware that everything can be coded up in bits and hence we accept that there is an encoding of TMs in binary. i.e. given a TM M we write ⌈M⌉∈ {0, 1}* for its binary encoding. We assume that the encoding contains its length such that we know when subsequent input on the tape starts.

Give the formal definition of the following:
FSA
DFA
NFA

Answer

FSA

An FSA is a virtual device that manipulates a candidate string, one character at a time, and determines whether that string is in the language implemented by the machine. The simplest state machine reads the string exactly once, and has no memory, only registers. It is therefore a finite state automaton. An FSA is defined by a set of states and a transition function that maps state/input pairs into states. In this state, reading this character, move to that state and advance to the next character.

DFA

A deterministic finite acceptor/automaton or DFA is a quintuple: $M = (Q, \Sigma, \delta, q0, F)$
Where

CALCULUS EDUCATIONAL CONSULT 2019_2 SECOND SEMESTER USE ONLY

- Q is a finite set of *states,*
- Σis a finite set of symbols, the *input alphabet,*
- δ: Q ×Σ→Q is a *transition function,*
- q0 ∈Q is the *initial state,*
- F ⊆Q is a set of *final states.*

NFA

A nondeterministic finite acceptor/automaton or NFA is defined by the quintuple

M = (Q, Σ, δ, q0, F)

Where

M = (Q, Σ, δ, q0, F)
where

- Q is a finite set of *states,*
- Σis a finite set of symbols, the *input alphabet,*
- δ: Q ×(Σ∪{ λ}) →2^Q is a *transition function,*
- q0 ∈Q is the *initial state,*
- F ⊆Q is a set of *final states.*

What you understand by unrestricted grammars in Formal Language

Answer

The productions of an unrestricted grammar have the form:

(V ∪T)+ →(V ∪T) *

The other grammar types we have considered (left linear, right linear, linear, context free) restrict the form of productions in one way or another. An unrestricted grammar does not.

What is a pigeonhole?

Answer

Its is a hole or small recess for pigeons to nest.its a small open compartment (as in a desk or cabinet) for keeping letters or documents.its a neat category which usually fails to reflect actual complexities

Describe an algorithm for the Operation of a DFA

Answer

• Start with the "current state" set to the start state and a "read head" at the beginning of the input string;
• while there are still characters in the string:
o Read the next character and advance the read head;
o From the current state, follow the arc that is labelled with the character just read; the state that the arc points to becomes the next current state;
• When all characters have been read, accept the string if the current state is a final state, otherwise reject the string.

Briefly describe the pigeonhole principle. How is it related to them pumping lemma for regular languages?

Answer

If n objects are put into m containers, where n > m, then at least one container must hold more than one object. The pigeonhole can be used to prove that certain infinite languages are not regular.
Suppose L is regular. There are an infinite number of values of n but M(L) has only a finite number of states. By the pigeonhole principle, there must be distinct values of i and j such that ai and aj end in the same state.

What do you understand by formal languages and Give any three examples of languages

Answer

A formal language is a set of words, i.e. finite strings of letters, or symbols. The inventory from which these letters are taken is called the alphabet over which the language is defined. A formal language is often defined by means of a

CAC COLLEGE EDUCATIONAL CONSULT 2019_2 SECOND SEMESTER USE ONLY

formal grammar. Formal languages are a purely syntactical notion, so there is not necessarily any meaning associated with them. To distinguish the words that belong to a language from arbitrary words over its alphabet, the former are sometimes called well-formed words.
Examples of formal languages
• L = Σ *, the set of all words over Σ;
• L = {a}* = {an}, where n ranges over the natural numbers and an means "a" repeated n times (this is the set of words consisting only of the symbol "a");
• the set of syntactically correct programs in a given programming language (the syntax of which is usually defined by a context-free grammar;
• the set of inputs upon which a certain Turing machine halts; or
• the set of maximal strings of alphanumeric ASCII characters on this line, (i.e., the set {"the", "set", "of", "maximal", "strings", "alphanumeric", "ASCII", "characters", "on", "this", "line", "i", "e"}).

Is formal language finite or infinite? Discuss

Answer

For finite languages one can simply enumerate all well-formed words. For example, we can define a language L as just L = {"a", "b", "ab", "cba"}. However, even over a finite (non-empty) alphabet such as Σ = {a, b} there are infinitely many words: "a", "abb", "ababba", "aaababbbbaab", …. Therefore formal languages are typically infinite, and defining an infinite formal language is not as simple as writing L = {"a", "b", "ab", "cba"}.

State five what pumping lemma say.

answer

• If an infinite language is regular, it can be defined by a DFA.
• The DFA has some finite number of states (say, n).
• Since the language is infinite, some strings of the language must have length >n.
• For a string of length > n accepted by the DFA, the walk through the DFA must contain a cycle.
• Repeating the cycle an arbitrary number of times must yield another string accepted by the DFA.

What do you understand by halting problem?

Answer

The "Halting Problem" is a very strong, provably correct, statement that no one will ever be able to write a computer program or design a Turing machine that can determine if a arbitrary program will halt (stop, exit) for a given input. The Halting Problem says that no computer program or Turing machine can determine if ALL computer programs or Turing machines will halt or not halt on ALL inputs. To prove the Halting Problem is unsolvable we will construct one program and one input for which there is no computer program or Turing machine.

Define context-sensitive grammars

Answer

a grammar is context-sensitive (or type 1) if the left hand side of a production is at least as long as the right hand side. That is for each $\alpha \rightarrow \beta \in P$ we have $|\alpha| \leq |\beta|$
Here is an example of a context-sensitive grammar G = (V, Σ, S, P): with L(G) = {{a n b n c n | n ∈ ℕ ∧ n ≥ 1}. Where V = {S, B, C}
Σ = {a, b, c}
P = { S → aSBC
S → aBC
aB → ab
bB → bB
bC → bc
cC → cc}

What do you understand by decision problems?.

Answer

Decision problems are stated as questions where the answer is binary, 0 or 1, False or True, No or yes, Reject or

CAPULOS EDUCATIONAL CONSULT 2019_2 SECOND SEMESTER USE ONLY

Accept and so forth. Generally a decision problem states a problem and gives a candidate solution, asking if the solution solves the problem.
Examples: Given the math expression 2+2 is the answer 4?
Given a formal language and a string, is the string in the language?
Given a grammar and a string, is the string accepted by the grammar?

PDA is defined as a collection of seven things mention
Answer

- an alphabet of input letters
- an input tape containing an input string followed by
- an alphabet of stack letters
- a pushdown stack, initially empty
- one start state
- a set of accept states
- a transition function

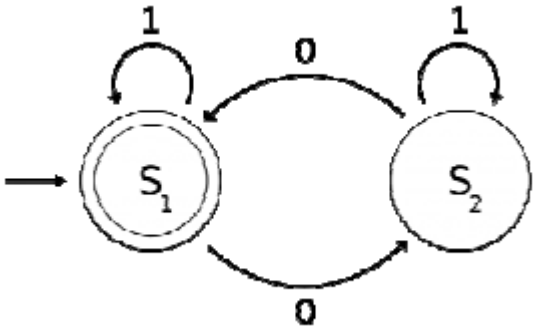
The context-free languages are closed under the formation of?
Answer

- Union
- Concatenation
- Kleene star.

State the two type of PDAs
Answer

Nondeterministic Pushdown Automata (NPDA)
Deterministic Pushdown Automata

With the aid of diagram, briefly describe the automata theory
Answer



Clearly state the Godel incompleteness theorem
Answer

it states that ;
-Any formal system powerful enough to express arithmetic must have true theorems that cannot be proven within the formal system
- Basically Godel proved that when trying to add axioms to a formal system in order to prove all true theorems within the formal system, eventually the system will become inconsistent before it becomes complete.
- A complete formal system is a formal system where all true theorems can be proved.
- An inconsistent formal system is a formal system where at least one false statement can be proved within the formal system.

List the three ways of defining a language
Answer

Definition by Regular Expression
Definition by NFA
Definition by Grammar

In a formal way, describe an automaton

Answer

An automaton is represented formally by the 5-tuple ⟨Q, Σ, δ, q0, F⟩, where:
• Q is a finite set of states.
• Σ is a finite set of symbols, called the alphabet of the automaton.
• δ is the transition function, that is, δ: Q × Σ → Q.
• q0 is the start state, that is, the state which the automaton is in when no input has been processed yet, where q0 ∈ Q.
• F is a set of states of Q (i.e. F⊆Q) called accept states.

Explain the Regular Expressions

Answer

A regular expression can be used to define a language. A regular expression represents a "pattern;" strings that match the pattern that are in the language, strings that do not match the pattern are not in the language.
The following are primitive regular expressions:
• x, for each x ,
• , the empty string, and
• , indicating no strings at all.

Define context-sensitive grammars

Answer

a grammar is context-sensitive (or type 1) if the left hand side of a production is at least as long as the right hand side. That is for each α → β ∈ P we have |α| ≤ |β |
Here is an example of a context-sensitive grammar G = (V, Σ, S, P): with L(G) = {{a n b n c n | n ∈ ℕ ∧ n ≥ 1}. Where V = {S, B, C}
Σ = {a, b, c}
P = { S → aSBC
S → aBC
aB → ab
bB → bB
bC → bc
cC → cc}

List 6 types of Automata

Answer

Deterministic finite automata (DFA)
Nondeterministic finite automata (NFA)
Nondeterministic finite automata with ε transitions (FND-ε or ε-NFA)
Pushdown automata (PDA)
Linear bounded automata (LBA)
Turing machines
Timed automata
Deterministic Büchi automata
Nondeterministic/Deterministic Rabin automata
Nondeterministic/Deterministic Streett automata
Nondeterministic/Deterministic parity automata
Nondeterministic/Deterministic Muller automata

As touching regular expressions, state the precedence of the following operations relative to one another (3

CALCULUS EDUCATIONAL CONSULT 2019_2 SECOND SEMESTER USE ONLY

marks)

1. **Kleene Star**
2. **Concatenation**
3. **Union**

Answer

1) The unary operator * (kleene closure) has the highest precedence and is left associative. For example, a+bc* or a|bc* denotes the language {a, b, bc, bcc, bccc, bcccc, ...}.
2) Concatenation has a second highest precedence and is left associative.
3) Union has lowest precedence and is left associative

When is a grammar recursively enumerable?

Answer

A language is recursively enumerable if there exists a Turing machine that accepts every string of the language, and does not accept strings that are not in the language.

List and define the two types of Push Down Automata

Answer

- A deterministic PDA (DPDA) is one in which every input string has a unique path through the machine.
- A nondeterministic PDA (NPDA) is one in which we may have to choose among several paths for an input string.

A formal system can be regarded (i) complete; or (ii) incomplete. Discuss

Answer

Basically Godel proved that when trying to add axioms to a formal system in order to prove all true theorems within the formal system, eventually the system will become inconsistent before it becomes complete. A complete formal system is a formal system where all true theorems can be proved. An inconsistent formal system is a formal system where at least one false statement can be proved within the formal system. Due to the computational equivalence of formal systems to other computational capability, we get the Halting problem, the uncomputable numbers and other unsolvable problems.

Considering that an automaton as a computer, state the way(s) it can handle non-determinism

1. Use a recursive backtracking algorithm. Whenever the automaton has to make a choice, cycle through all the alternatives and make a recursive call to determine whether any of the alternatives leads to a solution (final state).
2. Maintain a state set or a state vector, keeping track of all the states that the NFA could be in at any given point in the string.
3. Use a quantum computer. Quantum computers explore literally all possibilities simultaneously. They are theoretically possible, but are at the cutting edge of physics. It may (or may not) be feasible to build such a device.

Distinguish between regular grammar and context-free grammar

Answer

Regular grammar	Context free grammer
In regular grammars, the left hand side is again only a single nonterminal symbol, but now the right-hand side is also restricted	A context-free grammar is a grammar in which the left-hand side of each production rule consists of only a single nonterminal symbol.
The right side may be the empty string, or a single terminal symbol, or a single terminal symbol followed by a nonterminal symbol, but nothing else	Their restriction is non-trivial; not all languages can be generated by context-free grammars. Those that can are called context-free languages.

State two of the ways of implementing a DFA.

Answer

Using a GO TO Statement
Using a CASE Statement

CARLILLE EDUCATIONAL CONSULT 2019_2 SECOND SEMESTER USE ONLY

Explain the two types of Push DownAutomata

Answer

• A deterministic PDA (DPDA) is one in which every input string has a unique path through the machine.
• A nondeterministic PDA (NPDA) is one in which we may have to choose among several paths for an input string

Distinguish between a word and a vocabulary in formal language. Use examples to illustrate your answer

Answer

A word over an alphabet can be any finite sequence, or string, of letters. The set of all words over an alphabet Σ is usually denoted by Σ * (using the Kleene star).it is also known as formulas or sentences
A vocabulary (or alphabet or character set or word list) is a finite nonempty set of indivisible symbols (letters, digits, punctuation marks, operators, etc.). an alphabet or vocabulary is a finite set of symbols or letters, e.g. characters or digits. The most common alphabet is {0,1}, the binary alphabet

Let V be a set of strings. Is $V^+ = V^*$? Justify your answer.

Answer

If V is a set, then V* denotes the set of all finite strings of elements of V including the empty string which will be denoted by ε. e.g. {0,1}* = {ε, 0, 1, 00, 01, 10, 11, 000, 001,... }
The set of all non empty strings of elements of V is denoted by V+
Usually, V+ = V* \ {ε}, but when ε ∈ V, V+ = V*. e.g.
{0,1}+ = {0, 1, 00, 01, 10, 11, 000, 001,... }
but {ε, 0, 1}+ = {0,1}* = {ε, 0, 1, 00, 01, 10, 11, 000, 001,... }

Enumerate the components of a formal grammar.

Answer

• A finite set N of nonterminal symbols, none of which appear in strings formed from G.
• A finite set Σ of terminal symbols that is disjoint from N.
• A finite set P of production rules, each rule of the form
$(\Sigma \cup N)^* N (\Sigma \cup N)^* \to (\Sigma \cup N)^*$
where * is the Kleene star operator and denotes set union. That is, each production rule maps from one string of symbols to another, where the first string (the "head") contains an arbitrary number of symbols provided at least one of them is a nonterminal.
• A distinguished symbol that is the start symbol.

Differentiate between an alphabet and a language

answer

A **vocabulary** (or alphabet or character set or word list) is a finite nonempty set of indivisible symbols (letters, digits, punctuation marks, operators, etc.).
A l**anguage** over a vocabulary V is any subset L of V* which has a finite description. There are two approaches for making this mathematically precise. One is to use a grammar – a form of inductive definition of L.

Enumerate any two of the typical questions being asked about formalism in formal language theory.

Answer

• What is their expressive power? (Can formalism X describe every language that formalism Y can describe? Can it describe other languages?)
• What is their recognizability? (How difficult is it to decide whether a given word belongs to a language described by formalism X?)
• What is their comparability? (How difficult is it to decide whether two languages, one described in formalism X and one in formalism Y, or in X again, are actually the same language?).

Define automata theory.

Answer

automata theory is the study of abstract machines (or more appropriately, abstract 'mathematical' machines or systems) and the computational problems that can be solved using these machines. These abstract machines are

CALCULUS EDUCATIONAL CONSULT 2019_2 SECOND SEMESTER USE ONLY

called automata.

Formally define an automaton

Answer

An automaton is represented formally by the 5-tuple ⟨Q, Σ, δ, q0, F⟩, where:
• Q is a finite set of states.
• Σ is a finite set of symbols, called the alphabet of the automaton.
• δ is the transition function, that is, δ: Q × Σ → Q.
• q0 is the start state, that is, the state which the automaton is in when no input has been processed yet, where q0 ∈ Q.
• F is a set of states of Q (i.e. F⊆Q) called accept states.

Briefly describe any two of the popular variations in the definition of different components of automata.

Answer

Input
• **Finite input**: An automaton that accepts only finite sequence of words. The above introductory definition only accepts finite words.
• **Infinite input**: An automaton that accepts infinite words (ω-words). Such automata are called ω-automata.
• **Tree word input**: The input may be a tree of symbols instead of sequence of symbols.

States
• **Finite states**: An automaton that contains only a finite number of states. The above introductory definition describes automata with finite numbers of states.
• **Infinite states**: An automaton that may not have a finite number of states, or even a countable number of states. For example, the quantum finite automaton or topological automaton has uncountable infinity of states.
• **Stack memory**: An automaton may also contain some extra memory in the form of a stack in which symbols can be pushed and popped. This kind of automaton is called a pushdown automaton

Transition Function
• **Deterministic**: For a given current state and an input symbol, if an automatoncan only jump to one and only one state then it is a deterministic automaton.
• **Nondeterministic**: An automaton that, after reading an input symbol, may jump into any of a number of states, as licensed by its transition relation.
• **Alternation**: This idea is quite similar to tree automaton, but orthogonal. The automaton may run its multiple copies on the same next read symbol.

Acceptance Condition
• **Acceptance of finite words**: Same as described in the informal definition above.
• **Acceptance of infinite words**: an omega automaton cannot have final states, as infinite words never terminate. Rather, acceptance of the word is decided by looking at the infinite sequence of visited states during the run.
• **Probabilistic acceptance**: An automaton need not strictly accept or reject an input.

List any four types of automata and state their respective recognizable language.

Answer

CALCULUS EDUCATIONAL CONSULT 2019 2 SECOND SEMESTER USE ONLY

Table 1: Types of Automata

Automata	Recognizable language
Deterministic finite automata (DFA)	regular languages
Nondeterministic finite automata (NFA)	regular languages
Nondeterministic finite automata with ε transitions (FND-ε or ε-NFA)	regular languages
Pushdown automata (PDA)	context-free languages
Linear bounded automata (LBA)	context-sensitive language
Turing machines	Recursively enumerable languages
Timed automata	
Deterministic Büchi automata	omega limit languages
Nondeterministic Büchi automata	omega regular languages
Nondeterministic/Deterministic Rabin automata	omega regular languages
Nondeterministic/Deterministic Streett automata	omega regular languages
Nondeterministic/Deterministic parity automata	omega regular languages
Nondeterministic/Deterministic Muller automata	omega regular languages

In the context of automata theory, briefly describe the following terms:

i. **Recognised language**
ii. **Run**
iii. **Transducer**

Answer

Recognizable languages

The recognizable languages is the set of languages that are recognized by some automaton. For above definition of automata, the recognizable languages are regular languages. For different definitions of automata, the recognizable languages are different.

run

A run of the automaton on an input word w = a1, a2,...., an ∈ Σ*, is a sequence
of states q0, q1, q2,...., qn, where qi ∈ Q such that q0 is a start state and qi = δ(qi1,ai) for 0 < i ≤ n. In words, at first the automaton is at the start state q0 and then automaton reads symbols of the input word in sequence. When automaton reads symbol ai
then it jumps to state qi
= δ(qi-1,ai). qn said to be the final state
of the run

transducer

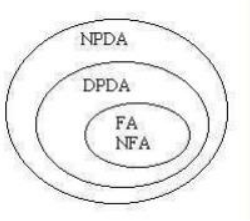
An automaton that produces more general output (typically a string) is called a transducer.

Is a Non-Deterministic Finite Automaton (NFA) more powerful than a Deterministic Finite Automaton (DFA)? Explain.

Answer

a nondeterministic PDA is more powerful than a deterministic one, unlike the situation with DFAs and NFAs. In other words, there are languages that can be accepted by an NPDA that cannot be accepted by a DPDA.

CAMBRIDGE EDUCATIONAL CONSULTANTS



State the precedence of the following with respect to regular expressions:
Kleene Star
Concatenation
Union
Parentheses

Answer

1) The unary operator *** (kleene closure**) has the highest precedence and is left associative. For example, a+bc* or a|bc* denotes the language {a, b, bc, bcc, bccc, bcccc, ...}.
2) Concatenation has a second highest precedence and is left associative.
3) Union has lowest precedence and is left associative
4) Parentheses override operator precedence as usual. For example, (0|1)* stands for all possible binary strings, 0|1* stands for either a 0 or an arbitrarily long string of 1's, and 01* stands for 0 followed by an arbitrarily long string of 1's.

Briefly explain the concept of ambiguity in grammars.

Answer

A grammar G is ambiguous if there exists some string w ∈L(G) for which
• there are two or more distinct derivation trees, or
• there are two or more distinct leftmost derivations, or
• there are two or more distinct rightmost derivations.
Grammars are used in compiler construction. Ambiguous grammars are undesirable because the derivation tree provides considerable information about the semantics of a program; conflicting derivation trees provide conflicting information.
Ambiguity is a property of a grammar, and it is usually (but not always) possible to find an equivalent unambiguous grammar.
An inherently ambiguous language is a language for which no unambiguous grammarexists.

Describe the types of Push Down Automata (PDAs).

answer

• A deterministic PDA (DPDA) is one in which every input string has a unique path through the machine.
• A nondeterministic PDA (NPDA) is one in which we may have to choose among several paths for an input string

Give the formal definition of a PDA

Formally, a PDA is defined as a collection of seven things:
• an alphabet of input letters
• an input tape containing an input string followed by
• an alphabet of stack letters
• a pushdown stack, initially empty
• one start state
• a set of accept states
• a transition function

List any two ways of defining a language

Answer

Definition by Grammar
Definition by NFA

CAVOLUS EDUCATIONAL CONSULT 2019_2 SECOND SEMESTER USE ONLY

Definition by Regular Expression

Enumerate any two of the mathematical concepts needed to proof the Halting Problem.

Answer

1. Proof by contradiction: Assume a statement is true, show that the assumption leads to a contradiction. Thus the statement is proven false.
2. Self referral: Have a computer program or a Turing machine operate on itself, well, a copy of itself, as input data.
3. Logical negation: Take a black box that accepts input and outputs true or false, put that black box in a bigger black box that switches the output so it is false or true respectively.

What does it mean to say a formally stated problem is:
Unsolvable?
Provably unsolvable?
Undecidable?

Answer

A formally stated problem is **Unsolvable** if no Turing machine exists to compute the solution.
A formally stated problem is provably **unsolvable** if it can be proved no Turing machine exists to compute the solution.
A formally stated problem is **Undecidable** if no total recursive function and thus, no Turing machine that always halts can be constructed to decide the problem, usually
true or false.

State the Halting Problem.

Answer

The Halting Problem says that no computer program or Turing machine can determine if ALL computer programs or Turing machines will halt or not halt on ALL inputs. To prove the Halting Problem is unsolvable we will construct one program and one input for which there is no computer program or Turing machine.

State Godel incompleteness theorem.

Answer

Any formal system powerful enough to express arithmetic must have true theorems that cannot be proven within the formal system.

Basically, what did Godel prove?

Basically Godel proved that when trying to add axioms to a formal system in order to prove all true theorems within the formal system, eventually the system will become inconsistent before it becomes complete.

When is formal system said to be:
i) Complete?
ii) Inconsistent?

Answer

A **complete formal system** is a formal system where all true theorems can be proved.
An **inconsistent formal system** is a formal system where at least one false statement can be proved within the formal system.

Distinguish between the strings and the words of a language

A string is generally understood as a data type storing a sequence of data values, usually bytes, in which elements usually stand for characters according to a character encoding, which differentiates it from the more general array data type while A word over an alphabet can be any finite sequence, or string, of letters.

Compare Non-Determinism to Determinism. Which of them is ideal for use in a computer system?

Nondeterministic: An automaton that, after reading an input symbol, may jump into any of a number of states, as licensed by its transition relation. Notice that the term transition function is replaced by transition relation: The automaton non-deterministically decides to jump into one of the allowed choices. Such automaton are called

CALCULUS EDUCATIONAL CONSULT 2019_2 SECOND SEMESTER USE ONLY

nondeterministic automaton.
Non deterministic is ideal

Explain the Applications of Finite Automata

Finite automata are used in text processing, compilers, and hardware design. Context-free grammar (CFG) is used in programming languages and artificial intelligence. Originally, CFG were used in the study of the human languages. Cellular automata are used in the field of biology, the most common example being John Conway's Game of Life. Some other examples which could be explained using automata theory in biology include mollusk and pine cones growth and pigmentation patterns. Going further, Stephen Wolfram claims that the entire universe could be explained by machines with a finite set of states and rules with a single initial condition. Other areas of interest which he has related to automata theory include: fluid flow, snowflake and crystal formation, chaos theory, cosmology, and financial analysis

Using the Set language, define the following terms and concepts: (i) Union (ii) Intersection (iii) Set Concatenation (iv) Kleene-star,(v) Set Subtraction (vi) Complement

Answer

• The **concatenation** L1L2 consists of all strings of the form vw where v is a string from L1 and w is a string from L2.
• The i**ntersection** L1 ∩ L2 of L1 and L2 consists of all strings which are contained in both languages
• The **complement** ¬L of a language with respect to a given alphabet consists of all strings over the alphabet that are not in the language.
• The **Kleene star**: the language consisting of all words that are concatenations of 0 or more words in the original language

Set difference

Mark a state (A, B) as final if A is a final state in L1, but B is not a final state in L2.

Union of L1 and L2

1) Create a new start state.
2) Make a transition from the new start state to each of the origina

Briefly explain the following:

- **Turing-decidable language**
- **Turing-recognizable language**

Turing-recognizable language

The recognizable languages is the set of languages that are recognized by some automaton. For above definition of automata, the recognizable languages are regular languages. For different definitions of automata, the recognizable languages are different.

Turing-decidable language

We say a language is decidable, if there is a TM which will always stop. There are type-0-languages which are not decidable – the most famous one is the halting problem

State the characteristics of grammars in Greibach Normal Form

A grammar is in Greibach Normal Form if all productions are of the form:
$A \rightarrow ax$
where a is a terminal and x V*.
Grammars in Greibach Normal Form are typically ugly and much longer than the CFG from which they were derived..

State the use(s) of Greibach Normal Form

Greibach Normal Form is useful for proving the equivalence of CFGs and NPDAs

Formally define Type 1 grammar

We say that a grammar is context-sensitive (or type 1) if the left hand side of a production is at least as long as the right hand side. That is for each $\alpha \rightarrow \beta \in P$ we have $|\alpha| \leq |\beta|$

CALCULUS EDUCATIONAL CONSULT 2019_2 SECOND SEMESTER USE ONLY

Define context-sensitive grammars

Grammars G = (V, Σ, S, P) are defined as context-free grammars before with the only difference that there may be several symbols on the left-hand side of a production

What do you understand by decision problems?

Decision problems are stated as questions where the answer is binary, 0 or 1, False or True, No or yes, Reject or Accept and so forth.

Generally a decision problem states a problem and gives a candidate solution, asking if the solution solves the problem.

Examples:

Given the math expression 2+2 is the answer 4?

Given a formal language and a string, is the string in the language?

Given a grammar and a string, is the string accepted by the grammar?

What do you understand by the term automata theory?

In theoretical computer science, automata theory is the study of abstract machines (or more appropriately, abstract 'mathematical' machines or systems) and the computational problems that can be solved using these machines. These abstract machines are called automata.

Define leftmost and rightmost derivation of a grammar. State the distinction between the two.

A **leftmost derivation** is obtained by applying production to the **leftmost** variable in each step.

A **rightmost derivation** is obtained by applying production to the **rightmost** variable in each step.

Consider the grammar: G = ({S, A, B, C}, {a, b, c}, S, P) Where, P = {S→ABC, A→aA, A→□, B→bB, B→□, C→cC, C→□}, derive the string

abbc in a:

i) leftmost derivation

ii) rightmost derivation

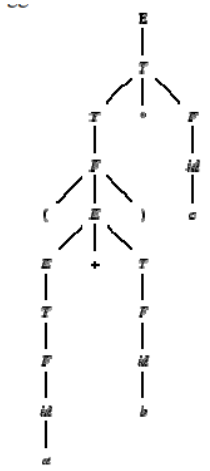
If we always expanded the leftmost variable first, we would have a leftmost derivation:

S ⇒ABC ⇒aABC ⇒aBC ⇒abBC ⇒abbBC ⇒abbC ⇒abbcC ⇒abbc

Conversely, if we always expanded the rightmost variable first, we would have a rightmost derivation:

S ⇒ABC ⇒ABcC ⇒ABc ⇒AbBc ⇒AbbBc ⇒Abbc ⇒aAbbc ⇒abbc

Draw the derivation tree for the leftmost derivation



When is a grammar said to be in Chomsky normal form?

The context-free Grammar G = (X, T, P, S) is said to be in Chomsky normal form if all grammar rules have the form

NATIONAL CONSULT 2019 2 SECOND SEMESTER USE ONLY

$$A \Rightarrow a \mid BC, \qquad (1)$$

for $a \in T$ and $B, C \in X - T$. There is one exception. If $\lambda \in L(G)$, then the single extra rule

$$S \Rightarrow \lambda \qquad (2)$$

is permitted. If $\lambda \notin L(G)$ then production rule 2 is not allowed.

Prove that context-free languages are closed under the formation of union.

Let $G_1 = (X_1, T_1, P_1, S_1)$ and $G_2 = (X_2, T_2, P_2, S_2)$ be context-free grammars. Without loss of generality, it can be supposed that $(X_1 - T_1) \cap (X_2 - T_2) = \phi$. If not, then rewrite the grammar rules of one with new nonterminal symbols. Let S be another nonterminal symbol. Then set $P = P_1 \cup P_2 \cup \{S \Rightarrow S_1, S \Rightarrow S_2\}$ and $G = (X_1 \cup X_2 \cup \{S\}, T_1 \cup T_2, P, S)$. It then follows that $L(G) = L(G_1) \cup L(G_2)$. Verifying the closure under concatenation and Kleene star in a similar manner is left to you as an exercise.

The closure under intersection property is not quite exact.

In the context of automata theory, briefly describe the following terms:
i. Recognised language
ii. Run
iii. Transducer

- **Recognised language**

An automaton can recognize a formal language. The recognized language $L \subset \Sigma^*$ by an automaton is the set of all the words that are accepted by the automaton.

- **Run**

run of the automaton on an input word $w = a1, a2, ..., an \in \Sigma^*$, is a sequence of states q0, q1, q2,...., qn, where $qi \in Q$ such that q0 is a start state and $qi = \delta(qi1,ai)$ for $0 < i \leq n$. In words, at first the automaton is at the start state q0 and then automaton reads symbols of the input word in sequence

- **transducer**

An automaton that produces more general output (typically a string) is called a transducer.

Give the state-set implementation of NFAs

Another way to implement an NFA is to keep either a state set or a bit vector of all the states that the NFA could be in at any given time. Implementation is easier if you use a bit-vector approach (v[i] is True iff state i is a possible state), since most languages provide vectors, but not sets, as a built-in datatype.

```
function NFA (state set A) returns Boolean:
    local state set B, state a, state b, state c, symbol
x;
  for each a in A do
     for each transition from a
        to some state b do
           add b to B;
   while there is a next symbol do
```

CALCULUS EDUCATIONAL CONSULT 2019_2 SECOND SEMESTER USE ONLY

```
{ read next symbol (x);
B := ∅;
for each a in A do
{ for each transition from a to some state b
Do
add b to B
for each x transition from a to some state b
Do
add b to B;
}
for each λ transition from
some state b in B to some state c not in B do
add c to B;
A := B;
}
if any element of A is a final state then
return True;
else
return False;
```

explain the Chomsky normal form

Work with a given context-free grammar is greatly facilitated by putting it into a socalled normal form. This provides some kind of regularity in the appearance of the right-hand sides of grammar rules.

The context-free Grammar G = (X, T, P, S) is said to be in Chomsky normal form if all grammar rules have the form

$A \Rightarrow a \mid BC$,

for $a \in T$ and $B, C \in X - T$. There is one exception. If $\lambda \in L(G)$, then the single extra rule

$S \Rightarrow \lambda$

is permitted. If $\lambda \notin L(G)$ then production rule 2 is not allowed.

LET G be a context free grammar with productions

S → TU | V

T → aTb | λ

U→ cU| λ

W → bW| λ

- **Identify the nullable variable in the grammar**
- **Elimate the λ-production**

CALCULUS EDUCATIONAL CONSULT 2019_2 SECOND SEMESTER USE ONLY

solution

1.the nullable variables are S,T,U,W

2.For the production S → TU | V we add the productions S → A | B
For the production A → aAA we add the productions A → aA | a
For the production B → bBB we add the productions B → bB | b

What is DFA

A deterministic finite acceptor/automaton or DFA is a quintuple:

M1 = $(Q, \Sigma, \delta, q0, F)$ where

- Q is {, q1, q2, q3},
- Σ is {a,b,ab}
- δ_i is {a,b,ab}

CALCULUS EDUCATIONAL CONSULT 2019_2 SECOND SEMESTER USE ONLY