1. Give context-free grammars that generate the following languages.
   1. { *w* ∈ {0*,* 1}∗ | *w* contains at least three 1s }

Answer: *G* = (*V,* Σ*, R, S*) with set of variables *V* = {*S, X*}, where *S* is the start variable; set of terminals Σ = {0*,* 1}; and rules

*S* → *X*1*X*1*X*1*X*

*X* → 0*X* | 1*X* | *ε*

* 1. { *w* ∈ {0*,* 1}∗ | *w* = *w*R and |*w*| is even }

Answer: *G* = (*V,* Σ*, R, S*) with set of variables *V* = {*S*}, where *S* is the start variable; set of terminals Σ = {0*,* 1}; and rules

*S* → 0*S*0 | 1*S*1 | *ε*

* 1. { *w* ∈ {0*,* 1}∗ | the length of *w* is odd and the middle symbol is 0 }

Answer: *G* = (*V,* Σ*, R, S*) with set of variables *V* = {*S*}, where *S* is the start variable; set of terminals Σ = {0*,* 1}; and rules

*S* → 0*S*0 | 0*S*1 | 1*S*0 | 1*S*1 | 0

* 1. { *ai bj ck* | *i, j, k* ≥ 0, and *i* = *j* or *i* = *k* }

Answer: *G* = (*V,* Σ*, R, S*) with set of variables *V* = {*S, W, X, Y, Z*}, where

*S* is the start variable; set of terminals Σ = {*a, b, c*}; and rules

|  |  |  |
| --- | --- | --- |
| *S* | → | *XY* | *W* |
| *X* | → | *aXb* | *ε* |
| *Y* | → | *cY* | *ε* |
| *W* | → | *aWc* | *Z* |
| *Z* | → | *bZ* | *ε* |

* 1. { *ai bj ck* | *i, j, k* ≥ 0 and *i* + *j* = *k* }

Answer: *G* = (*V,* Σ*, R, S*) with set of variables *V* = {*S, X*}, where *S* is the start variable; set of terminals Σ = {*a, b, c*}; and rules

*S* → *aSc* | *X X* → *bXc* | *ε*

* 1. { *ai bj ck* | *i, j, k* ≥ 0 and *i* + *k* = *j* }

Answer: Let *L* = { *ai bj ck* | *i, j, k* ≥ 0 and *i* + *k* = *j* } be the language given in the problem, and define other languages

*L*1 = { *ai bi* | *i* ≥ 0 }*, L*2 = { *bk ck* | *k* ≥ 0 }*.*

Note that *L* = *L*1 ◦ *L*2 because concatenating any string *aibi* ∈ *L*1 with any string *bkck* ∈ *L*2 results in a string *aibibkck* = *aibi*+*kck* ∈ *L*. Thus, if *L*1 has a CFG *G*1 = (*V*1*,* Σ*, R*1*, S*1), and *L*2 has a CFG *G*2 = (*V*2*,* Σ*, R*2*, S*2), we

can construct a CFG for *L* = *L*1 ◦ *L*2 by using the approach in problem 3b, as suggested in the hint. Specifically,

* + - *L*1 has a CFG *G*1 = (*V*1*,* Σ*, R*1*, S*1), with *V*1 = {*S*1}, Σ = {*a, b, c*}, *S*1

as the starting variable, and rules *S*1 → *aS*1*b* | *ε* in *R*1;

* + - *L*2 has a CFG *G*2 = (*V*2*,* Σ*, R*2*, S*2), with *V*2 = {*S*2}, Σ = {*a, b, c*}, *S*2

as the starting variable, and rules *S*2 → *bS*2*c* | *ε* in *R*2.

Even though Σ = {*a, b, c*} for both CFGs *G*1 and *G*2, CFG *G*1 never generates a string with *c*, and CFG *G*2 never generates a string with *a*. Then from problem 3b, a CFG *G*3 = (*V*3*,* Σ*, R*3*, S*3) for *L* has *V*3 = *V*1 ∪ *V*2 ∪ {*S*3} = {*S*1*, S*2*, S*3}

with *S*3 the starting variable, Σ = {*a, b, c*}, and rules

*S*3 → *S*1*S*2

*S*1 → *aS*1*b* | *ε S*2 → *bS*2*c* | *ε*

* 1. ∅

Answer: *G* = (*V,* Σ*, R, S*) with set of variables *V* = {*S*}, where *S* is the start variable; set of terminals Σ = {0*,* 1}; and rules

*S* → *S*

Note that if we start a derivation, it never finishes, i.e., *S* ⇒ *S* ⇒ *S* ⇒ · · · , so no string is ever produced. Thus, *L*(*G*) = ∅.

* 1. The language *A* of strings of properly balanced left and right brackets: every left bracket can be paired with a unique subsequent right bracket, and every right

bracket can be paired with a unique preceding left bracket. Moreover, the string between any such pair has the same property. For example, [ ] [ [ [ ] [ ] ] [ ] ] ∈ *A*.

Answer: *G* = (*V,* Σ*, R, S*) with set of variables *V* = {*S*}, where *S* is the start variable; set of terminals Σ = {[*,* ]}; and rules

*S* → *ε* | *SS* | [*S*]

1. Let *T* = { 0*,* 1*,* (*,* )*,* ∪*,* ∗*,* ∅*, e* }. We may think of *T* as the set of symbols used by regular expressions over the alphabet {0*,* 1}; the only difference is that we use *e* for symbol *ε*, to avoid potential confusion in what follows.
   1. Your task is to design a CFG *G* with set of terminals *T* that generates exactly the regular expressions with alphabet {0*,* 1}.

Answer: *G* = (*V,* Σ*, R, S*) with set of variables *V* = {*S*}, where *S* is the start variable; set of terminals Σ = *T* ; and rules

*S* → *S* ∪ *S* | *SS* | *S*∗ | (*S*) | 0 | 1 | ∅ | *e*

* 1. Using your CFG *G*, give a derivation and the corresponding parse tree for the string (0 ∪ (10)∗1)∗.

Answer: A derivation for (0 ∪ (10)∗1)∗ is

*S* ⇒ *S*∗ ⇒ (*S*)∗ ⇒ (*S* ∪ *S*)∗ ⇒ (0 ∪ *S*)∗ ⇒ (0 ∪ *SS*)∗ ⇒ (0 ∪ *S*∗*S*)∗

⇒ (0 ∪ (*S*)∗*S*)∗ ⇒ (0 ∪ (*SS*)∗*S*)∗ ⇒ (0 ∪ (1*S*)∗*S*)∗

⇒ (0 ∪ (10)∗*S*)∗ ⇒ (0 ∪ (10)∗1)∗

and the corresponding parse tree is

*S*

*S* ∗

( *S* )

*S* ∪ *S*

0 *S S*

*S* ∗ 1

( *S* )

*S S*

1 0

1. (a) Suppose that language *A*1 has a context-free grammar *G*1 = (*V*1*,* Σ*, R*1*, S*1), and language *A*2 has a context-free grammar *G*2 = (*V*2*,* Σ*, R*2*, S*2), where, for *i* = 1*,* 2, *Vi* is the set of variables, *Ri* is the set of rules, and *Si* is the start variable for CFG *Gi*. The CFGs have the same set of terminals Σ. Assume that *V*1 ∩*V*2 =

∅. Define another CFG *G*3 = (*V*3*,* Σ*, R*3*, S*3) with *V*3 = *V*1 ∪ *V*2 ∪ {*S*3}, where

*S*3 /∈ *V*1 ∪ *V*2, and *R*3 = *R*1 ∪ *R*2 ∪ { *S*3 → *S*1*, S*3 → *S*2 }. Argue that *G*3 generates the language *A*1 ∪ *A*2. Thus, conclude that the class of context-free languages is closed under union.

Answer: Let *A*3 = *A*1 ∪ *A*2, and we need to show that *L*(*G*3) = *A*3. To do this, we need to prove that *L*(*G*3) ⊆ *A*3 and *A*3 ⊆ *L*(*G*3). To show that *L*(*G*3) ⊆ *A*3, first consider any string *w* ∈ *L*(*G*3). Since *w* ∈ *L*(*G*3), we

∗

have that *S*3 ⇒ *w*. Since the only rules in *R*3 with *S*3 on the left side are

*S*3 → *S*1 | *S*2, we must have that *S*3 ⇒ *S*1

∗

∗

⇒ *w* or *S*3 ⇒ *S*2

∗

⇒ *w*. Suppose

first that *S*3 ⇒ *S*1 ⇒ *w*. Since *S*1 ∈ *V*1 and we assumed that *V*1 ∩ *V*2 = ∅, the

∗

derivation *S*1 ⇒ *w* must only use variables in *V*1 and rules in *R*1, which implies

∗

that *w* ∈ *A*1. Similarly, if *S*3 ⇒ *S*2 ⇒ *w*, then we must have that *w* ∈ *A*2.

Thus, *w* ∈ *A*3 = *A*1 ∪ *A*2, so *L*(*G*3) ⊆ *A*3.

To show that *A*3 ⊆ *L*(*G*3), first suppose that *w* ∈ *A*3. This implies *w* ∈ *A*1 or

∗ ∗

*w* ∈ *A*2. If *w* ∈ *A*1, then *S*1 ⇒ *w*. But then *S*3 ⇒ *S*1 ⇒ *w*, so *w* ∈ *L*(*G*3).

∗ ∗

Similarly, if *w* ∈ *A*2, then *S*2 ⇒ *w*. But then *S*3 ⇒ *S*2 ⇒ *w*, so *w* ∈ *L*(*G*3).

Thus, *A*3 ⊆ *L*(*G*3), and since we previously showed that *L*(*G*3) ⊆ *A*3, it follows that *L*(*G*3) = *A*3; i.e., the CFG *G*3 generates the language *A*1 ∪ *A*2.

1. Prove that the class of context-free languages is closed under concatenation.

Answer: Suppose that language *A*1 has a context-free grammar *G*1 = (*V*1*,* Σ*, R*1*, S*1), and language *A*2 has a context-free grammar *G*2 = (*V*2*,* Σ*, R*2*, S*2), where, for

*i* = 1*,* 2, *Vi* is the set of variables, *Ri* is the set of rules, and *Si* is the start variable for CFG *Gi*. The CFGs have the same set of terminals Σ. Assume that *V*1 ∩ *V*2 = ∅. Then a CFG for *A*1 ◦ *A*2 is *G*3 = (*V*3*,* Σ*, R*3*, S*3) with *V*3 = *V*1 ∪ *V*2 ∪ {*S*3}, where *S*3 /∈ *V*1 ∪ *V*2, and *R*3 = *R*1 ∪ *R*2 ∪ { *S*3 → *S*1*S*2 }.

To understand why *L*(*G*3) = *A*1 ◦ *A*2, note that any string *w* ∈ *A*1 ◦ *A*2 can

∗

be written as *w* = *uv*, where *u* ∈ *A*1 and *v* ∈ *A*2. It follows that *S*1 ⇒ *u* and

∗ ∗ ∗

*S*2 ⇒ *v*, so *S*3 ⇒ *S*1*S*2 ⇒ *uS*2 ⇒ *uv*, so *w* = *uv* ∈ *L*(*G*3). This proves that

*A*1 ◦ *A*2 ⊆ *L*(*G*3).

To prove that *L*(*G*3) ⊆ *A*1 ◦ *A*2, consider any string *w* ∈ *L*(*G*3). Since *w* ∈

∗

*L*(*G*3), it follows that *S*3 ⇒ *w*. The only rule in *R*3 with *S*3 on the left side is

∗

*S*3 → *S*1*S*2, so *S*3 ⇒ *S*1*S*2 ⇒ *w*. Since *V*1 ∩ *V*2 = ∅, any derivation starting

from *S*1 can only generate a string in *A*1, and any derivation starting from *S*2

∗

can only generate a string in *A*2. Thus, since *S*3 ⇒ *S*1*S*2 ⇒ *w*, it must be that

*w* is the concatenation of a string from *A*1 with a string from *A*2. Therefore,

*w* ∈ *A*1 ◦ *A*2, which establishes that *L*(*G*3) ⊆ *A*1 ◦ *A*2.

1. Prove that the class of context-free languages is closed under Kleene-star.

Answer: Suppose that language *A* has a context-free grammar *G*1 = (*V*1*,* Σ*, R*1*, S*1). Then a CFG for *A*∗ is *G*2 = (*V*2*,* Σ*, R*2*, S*2) with *V*2 = *V*1 ∪ {*S*2}, where

*S*2 /∈ *V*1, and *R*2 = *R*1 ∪ { *S*2 → *S*1*S*2*, S*2 → *ε* }.

To show that *L*(*G*2) = *A*∗, we first prove that *A*∗ ⊆ *L*(*G*2). Consider any string *w* ∈ *A*∗. We can write *w* = *w*1*w*2 · · · *wn* for some *n* ≥ 0, where each *wi* ∈ *A*. (Here, we interpret *w* = *w*1*w*2 · · · *wn* for *n* = 0 to be *w* = *ε*.) Since

∗

each *wi* ∈ *A*, we have that *S*1 ⇒ *wi*. To derive the string *w* using CFG *G*2, we

first apply the rule *S*2 → *S*1*S*2 a total of *n* times, followed by one application of

∗

the rule *S*2 → *ε*. Then for the *i*th *S*1, we use *S*1 ⇒ *wi*. Thus, we get

∗ ∗

*S*2 ⇒ *S*`1*S*1˛·¸· · *S*x1 *S*2 ⇒ `*S*1*S*1˛·¸· · *S*x1 ⇒ *w*1*w*2 · · · *wn* = *w*

*n* times *n* times

Therefore, *w* ∈ *L*(*G*2), so *A*∗ ⊆ *L*(*G*2).

To show that *L*(*G*2) ⊆ *A*∗, suppose we apply the rule *S* → *S*1*S*2 a total of

*n* ≥ 0 times, followed by an application of the rule *S*2 → *ε*. This gives

∗

*S*2 ⇒ `*S*1*S*1˛·¸· · *S*x1 *S*2 ⇒ `*S*1*S*1˛·¸· · *S*x1 *.*

*n* times *n* times

Now each of the variables *S*1 can be used to derive a string *wi* ∈ *A*, i.e., from the

∗

*i*th *S*1, we get *S*1 ⇒ *wi*. Thus,

*S* ∗ ∗ ∗

2 ⇒ `*S*1*S*1˛·¸· · *S*x1 ⇒ *w*1*w*2 · · · *wn* ∈ *A*

*n* times

since each *wi* ∈ *A*. Therefore, we end up with a string in *A*∗. To convince ourselves that the productions applied to the various separate *S*1 terms do not interfere in undesired ways, we need only think of the parse tree. Each *S*1 is the root of a distinct branch, and the rules along one branch do not affect those on another. Here, we assumed that we first applied the rule *S*2 → *S*1*S*2 a total of

*n* times, then applied the rule *S*2 → *ε*, and then applied rules to change each

*S*1 into strings. However, we could have applied the rules in a different order, as long as the rule *S*2 → *ε* is applied only after the *n* applications of *S*2 → *S*1*S*2. By examining the parse tree, we can argue as before that the order in which we applied the rules doesn’t matter.

1. Convert the following CFG into an equivalent CFG in Chomsky normal form, using the procedure given in Theorem 2.9.

*S* → *BSB* | *B* | *ε B* → 00 | *ε*

Answer: First introduce new start variable *S*0 and the new rule *S*0 → *S*, which gives

*S*0 → *S*

*S* → *BSB* | *B* | *ε B* → 00 | *ε*

Then we remove *ε* rules:

* Removing *B* → *ε* yields

*S*0 → *S*

*S* → *BSB* | *BS* | *SB* | *S* | *B* | *ε B* → 00

* Removing *S* → *ε* yields

|  |  |  |
| --- | --- | --- |
| *S*0 | → | *S* | *ε* |
| *S* | → | *BSB* | *BS* | *SB* | *S* | *B* | *BB* |
| *B* | → | 00 |

* We don’t need to remove the *ε*-rule *S*0 → *ε* since *S*0 is the start variable and that is allowed in Chomsky normal form.

Then we remove unit rules:

* Removing *S* → *S* yields

*S*0 → *S* | *ε*

*S* → *BSB* | *BS* | *SB* | *B* | *BB B* → 00

* Removing *S* → *B* yields

*S*0 → *S* | *ε*

*S* → *BSB* | *BS* | *SB* | 00 | *BB B* → 00

* Removing *S*0 → *S* gives

*S*0 → *BSB* | *BS* | *SB* | 00 | *BB* | *ε S* → *BSB* | *BS* | *SB* | 00 | *BB*

*B* → 00

Then we replaced ill-placed terminals 0 by variable *U* with new rule *U* → 0, which gives

*S*0 → *BSB* | *BS* | *SB* | *UU* | *BB* | *ε S* → *BSB* | *BS* | *SB* | *UU* | *BB*

*B* → *UU*

*U* → 0

Then we shorten rules with a long RHS to a sequence of RHS’s with only 2 variables each. So the rule *S*0 → *BSB* is replaced by the 2 rules *S*0 → *BA*1 and *A*1 → *SB*. Also the rule *S* → *BSB* is replaced by the 2 rules *S* → *BA*2 and *A*2 → *SB*. Thus, our final CFG in Chomsky normal form is

|  |  |  |
| --- | --- | --- |
| *S*0 | → | *BA*1 | *BS* | *SB* | *UU* | *BB* | *ε* |
| *S* | → | *BA*2 | *BS* | *SB* | *UU* | *BB* |
| *B* | → | *UU* |
| *U* | → | 0 |
| *A*1 | → | *SB* |
| *A*2 | → | *SB* |

To be precise, the CFG in Chomsky normal form is *G* = (*V,* Σ*, R, S*0), where the set of variables is *V* = {*S*0*, S, B, U, A*1*, A*2}, the start variable is *S*0, the set of terminals is Σ = {0}, and the rules *R* are given above.

1. (a) The CFG *G* derives the string − − −5 as

*S* ⇒ −*S* ⇒ − − *S* ⇒ − − −*S* ⇒ − − −5

so − − −5 ∈ *L*(*G*). The CFG *G* derives the string 2 + − − 4 as

*S* ⇒ *S* + *S* ⇒ *S* + −*S* ⇒ *S* + − − *S* ⇒ 2 + − − *S* ⇒ 2 + − − 4

so 2 + − − 4 ∈ *L*(*G*).

(b) To prevent generating strings such as in part (a), we can use the CFG *G*′ = (*V* ′*,* Σ*, R*′*, S*), where *V* ′ = { *S, N* } is the set of variables with *S* as the starting variable, alphabet Σ = { +*,* −*,* ×*, /,* (*,* )*,* 0*,* 1*,* 2*, . . . ,* 9 }, and rules *R*′ as

*S* → *S* + *S* | *S* − *S* | *S* × *S* | *S/S* | (*S*) | *N* | −*N N* → 0 | 1 | · · · | 9