CSE 311 Section 07

Structural Induction, Functions, Regex, CFGs

Administrivia

Announcements & Reminders

- Midterm grades released
  - See Parker's Ed post for more info
  - Exams page on website has solutions and logistics for midterm re-take
  - Fill out Midterm retake scheduling form - posted by Evan on Ed (due tonight!)

- HW5 Grades Released
  - Regrade request window open as usual
  - If something was graded incorrectly, submit a regrade request

- HW6
  - Due **Friday** 8/8 @ 11:59pm (note the unusual day!)
  - Same late due date! Saturday 8/9 @ 11:59 PM

Structural Induction

Idea of Structural Induction

Every element is built up recursively…

So to show $P(s)$ for all $s \in S$…

Show $P(b)$ for all base case elements b.

Show for an arbitrary element of the set, if $P()$ holds for that element, then $P()$ holds for everything you can make out of that element.

Structural Induction Template

Let $P(x)$ be “<predicate>”. We show $P(x)$ holds for all $x \in S$ by structural induction.

Base Case: Show $P(x)$
[Do that for every x in the basis step of defining S.]

Inductive Hypothesis: Suppose $P(x)$ for an arbitrary x
[Do that for every x listed as already in S in the recursive rules.]

Inductive Step: Show $P()$ holds for y.
[You will need a separate case/step for every recursive rule.]

Therefore $P(x)$ holds for all $x \in S$ by the principle of induction.

Problem 2b – Structural Induction on Trees

Definition of Tree:
Basis Step: • is a Tree.
Recursive Step: If L is a Tree and R is a Tree then Tree(•, L, R) is a Tree

Definition of leaves():
leaves(•) = 1
leaves(Tree(•, L, R)) = leaves(L) + leaves(R)

Definition of size():
size(•) = 1
size(Tree(•, L, R)) =1 + size(L) + size(R)

Prove that leaves(T) ≥ size(T)/2 + 1/2 for all Trees T

Work on this problem with the people around you!

Problem 2b – Structural Induction on Trees

Prove that leaves(T) ≥ size(T)/2 + 1/2 for all Trees T

For $x \in S$, let P(x) be "".
We show P(x) holds for all $x \in S$ by structural induction on x.

Base Case: Show P(x) (for all x in the basis rules)

Inductive Hypothesis: Suppose P(x) (for all x in the recursive rules), i.e. (IH in terms of P(x))

Inductive Step: Goal: Show that P(?) holds. (IS goal in terms of P(?))

Conclusion: Therefore P(x) holds for all $x \in S$ by the principle of induction.

Problem 2b – Structural Induction on Trees

Prove that leaves(T) ≥ size(T)/2 + 1/2 for all Trees T

For a tree T, let P(T) be “leaves(T) ≥ size(T)/2 + 1/2”.
We show P(x) holds for all x ∈ S by structural induction on x.

Base Case: Show P(x) (for all x in the basis rules)

Inductive Hypothesis: Suppose P(x) (for all x in the recursive rules), i.e. (IH in terms of P(x))

Inductive Step: Goal: Show that P(?) holds. (IS goal in terms of P(?))

Conclusion: Therefore P(x) holds for all x ∈ S by the principle of induction.

Problem 2b – Structural Induction on Trees

Prove that leaves(T) ≥ size(T)/2 + 1/2 for all Trees T

For a tree T, let P(T) be “leaves(T) ≥ size(T)/2 + 1/2”.
We show P(T) holds for all trees T by structural induction on T.

Base Case: Show P(x) (for all x in the basis rules)

Inductive Hypothesis: Suppose P(x) (for all x in the recursive rules),
i.e. (IH in terms of P(x))

Inductive Step: Goal: Show that P(?) holds. (IS goal in terms of P(?))

Conclusion: Therefore P(T) holds for all trees T by the principle of induction.

Problem 2b - Structural Induction on Trees

Prove that leaves(T) ≥ size(T)/2 + 1/2 for all Trees T

For a tree T, let P(T) be “leaves(T) ≥ size(T)/2 + 1/2”.
We show P(T) holds for all trees T by structural induction on T.

Base Case: P(•): By definition of leaves(•), leaves(•) = 1 and size(•) = 1.
So, leaves(•) = 1 ≥ 1/2 + 1/2 = size(•)/2 + 1/2, so P(•) holds.

Inductive Hypothesis: Suppose P(x) (for all x in the recursive rules),
i.e. (IH in terms of P(x))

Inductive Step: Goal: Show that P(?) holds. (IS goal in terms of P(?))

Conclusion: Therefore P(T) holds for all trees T by the principle of induction.

Problem 2b - Structural Induction on Trees

Prove that leaves(T) ≥ size(T)/2 + 1/2 for all Trees T

For a tree T, let P(T) be “leaves(T) ≥ size(T)/2 + 1/2”.
We show P(T) holds for all trees T by structural induction on T.

Base Case: P(•): By definition of leaves(•), leaves(•) = 1 and size(•) = 1.
So, leaves(•) = 1 ≥ 1/2 + 1/2 = size(•)/2 + 1/2, so P(•) holds.

Inductive Hypothesis: Suppose P(L) and P(R) hold for some arbitrary trees L and R, i.e. (IH in terms of P(x))

Inductive Step: Goal: Show that P(?) holds. (IS goal in terms of P(?))

Conclusion: Therefore P(T) holds for all trees T by the principle of induction.

Problem 2b – Structural Induction on Trees

Prove that leaves(T) ≥ size(T)/2 + 1/2 for all Trees T

For a tree T, let P(T) be “leaves(T) ≥ size(T)/2 + 1/2”.
We show P(T) holds for all trees T by structural induction on T.

Base Case: P(•): By definition of leaves(•), leaves(•) = 1 and size(•) = 1.
So, leaves(•) = 1 ≥ 1/2 + 1/2 = size(•)/2 + 1/2, so P(•) holds.

Inductive Hypothesis: Suppose P(L) and P(R) hold for some arbitrary trees L and R, i.e., leaves(L) ≥ size(L)/2 + 1/2, leaves(R) ≥ size(R)/2 + 1/2

Inductive Step: Goal: Show that P(?) holds. (IS goal in terms of P(?))

Conclusion: Therefore P(T) holds for all trees T by the principle of induction.

Problem 2b – Structural Induction on Trees

Prove that leaves(T) ≥ size(T)/2 + 1/2 for all Trees T

For a tree T, let P(T) be “leaves(T) ≥ size(T)/2 + 1/2”.
We show P(T) holds for all trees T by structural induction on T.

Base Case: P(•): By definition of leaves(•), leaves(•) = 1 and size(•) = 1.
So, leaves(•) = 1 ≥ 1/2 + 1/2 = size(•)/2 + 1/2, so P(•) holds.

Inductive Hypothesis: Suppose P(L) and P(R) hold for some arbitrary trees L and R, i.e., leaves(L) ≥ size(L)/2 + 1/2, leaves(R) ≥ size(R)/2 + 1/2

Inductive Step: Goal: Show P(Tree(•, L, R)): leaves(Tree(•, L, R)) ≥ size(Tree(•, L, R))/2 + 1/2

Conclusion: Therefore P(T) holds for all trees T by the principle of induction.

Problem 2b – Structural Induction on Trees

Prove that leaves(T) ≥ size(T)/2 + 1/2 for all Trees T

For a tree T, let P(T) be "leaves(T) ≥ size(T)/2 + 1/2".
We show P(T) holds for all trees T by structural induction on T.

Base Case: P(•): By definition of leaves(•), leaves(•) = 1 and size(•) = 1.
So, leaves(•) = 1 ≥ 1/2 + 1/2 = size(•)/2 + 1/2, so P(•) holds.

Inductive Hypothesis: Suppose P(L) and P(R) hold for some arbitrary trees L and R, i.e., leaves(L) ≥ size(L)/2 + 1/2, leaves(R) ≥ size(R)/2 + 1/2

Inductive Step: Goal: Show P(Tree(•, L, R)): leaves(Tree(•, L, R)) ≥ size(Tree(•, L, R))/2 + 1/2

Just filling out this gets the majority of the proof done! Go for this skeleton first, and then think about what you need to do to complete the proof.

Conclusion: Therefore P(T) holds for all trees T by the principle of induction.

Problem 2b – Structural Induction on Trees

Prove that leaves(T) ≥ size(T)/2 + 1/2 for all Trees T

For a tree T, let P(T) be "leaves(T) ≥ size(T)/2 + 1/2".
We show P(T) holds for all trees T by structural induction on T.

Base Case: P(•): By definition of leaves(•), leaves(•) = 1 and size(•) = 1.
So, leaves(•) = 1 ≥ 1/2 + 1/2 = size(•)/2 + 1/2, so P(•) holds.

Inductive Hypothesis: Suppose P(L) and P(R) hold for some arbitrary trees L and R,
i.e., leaves(L) ≥ size(L)/2 + 1/2, leaves(R) ≥ size(R)/2 + 1/2

Inductive Step: Goal: Show P(Tree(•, L, R)): leaves(Tree(•, L, R)) ≥ size(Tree(•, L, R))/2 + 1/2
leaves(Tree(•, L, R)) =

Conclusion: Therefore P(T) holds for all trees T by the principle of induction.

Problem 2b – Structural Induction on Trees

Prove that leaves(T) ≥ size(T)/2 + 1/2 for all Trees T

For a tree T, let P(T) be "leaves(T) ≥ size(T)/2 + 1/2".
We show P(T) holds for all trees T by structural induction on T.

Base Case: P(•): By definition of leaves(•), leaves(•) = 1 and size(•) = 1.
So, leaves(•) = 1 ≥ 1/2 + 1/2 = size(•)/2 + 1/2, so P(•) holds.

Inductive Hypothesis: Suppose P(L) and P(R) hold for some arbitrary trees L and R,
i.e., leaves(L) ≥ size(L)/2 + 1/2, leaves(R) ≥ size(R)/2 + 1/2

Inductive Step: Goal: Show P(Tree(•, L, R)): leaves(Tree(•, L, R)) ≥ size(Tree(•, L, R))/2 + 1/2

leaves(Tree(•, L, R)) = leaves(L) + leaves(R) definition of leaves

Conclusion: Therefore P(T) holds for all trees T by the principle of induction.

Problem 2b – Structural Induction on Trees

Prove that leaves(T) ≥ size(T)/2 + 1/2 for all Trees T

For a tree T, let P(T) be "leaves(T) ≥ size(T)/2 + 1/2".
We show P(T) holds for all trees T by structural induction on T.

Base Case: P(•): By definition of leaves(•), leaves(•) = 1 and size(•) = 1.
So, leaves(•) = 1 ≥ 1/2 + 1/2 = size(•)/2 + 1/2, so P(•) holds.

Inductive Hypothesis: Suppose P(L) and P(R) hold for some arbitrary trees L and R,
i.e., leaves(L) ≥ size(L)/2 + 1/2, leaves(R) ≥ size(R)/2 + 1/2

Inductive Step: Goal: Show P(Tree(•, L, R)): leaves(Tree(•, L, R)) ≥ size(Tree(•, L, R))/2 + 1/2

leaves(Tree(•, L, R)) = leaves(L) + leaves(R) — definition of leaves

≥ (size(L)/2 + 1/2) + (size(R)/2 + 1/2) — by Inductive Hypothesis

Conclusion: Therefore P(T) holds for all trees T by the principle of induction.

Problem 2b – Structural Induction on Trees

Prove that leaves(T) ≥ size(T)/2 + 1/2 for all Trees T

For a tree T, let P(T) be "leaves(T) ≥ size(T)/2 + 1/2".
We show P(T) holds for all trees T by structural induction on T.

Base Case: P(•): By definition of leaves(•), leaves(•) = 1 and size(•) = 1.
So, leaves(•) = 1 ≥ 1/2 + 1/2 = size(•)/2 + 1/2, so P(•) holds.

Inductive Hypothesis: Suppose P(L) and P(R) hold for some arbitrary trees L and R,
i.e., leaves(L) ≥ size(L)/2 + 1/2, leaves(R) ≥ size(R)/2 + 1/2

Inductive Step: Goal: Show P(Tree(•, L, R)): leaves(Tree(•, L, R)) ≥ size(Tree(•, L, R))/2 + 1/2

leaves(Tree(•, L, R)) = leaves(L) + leaves(R) — definition of leaves
≥ (size(L)/2 + 1/2) + (size(R)/2 + 1/2) — by Inductive Hypothesis
= (1/2 + size(L)/2 + size(R)/2) + 1/2

Conclusion: Therefore P(T) holds for all trees T by the principle of induction.

Problem 2b – Structural Induction on Trees

Prove that leaves(T) ≥ size(T)/2 + 1/2 for all Trees T

For a tree T, let P(T) be "leaves(T) ≥ size(T)/2 + 1/2".
We show P(T) holds for all trees T by structural induction on T.

Base Case: P(•): By definition of leaves(•), leaves(•) = 1 and size(•) = 1.
So, leaves(•) = 1 ≥ 1/2 + 1/2 = size(•)/2 + 1/2, so P(•) holds.

Inductive Hypothesis: Suppose P(L) and P(R) hold for some arbitrary trees L and R,
i.e., leaves(L) ≥ size(L)/2 + 1/2, leaves(R) ≥ size(R)/2 + 1/2

Inductive Step: Goal: Show P(Tree(•, L, R)): leaves(Tree(•, L, R)) ≥ size(Tree(•, L, R))/2 + 1/2

leaves(Tree(•, L, R)) = leaves(L) + leaves(R) — definition of leaves
≥ (size(L)/2 + 1/2) + (size(R)/2 + 1/2) — by Inductive Hypothesis
= (1/2 + size(L)/2 + size(R)/2) + 1/2
= (1 + size(L) + size(R)) / 2 + 1/2

Conclusion: Therefore P(T) holds for all trees T by the principle of induction.

Problem 2b – Structural Induction on Trees

Prove that leaves(T) ≥ size(T)/2 + 1/2 for all Trees T

For a tree T, let P(T) be "leaves(T) ≥ size(T)/2 + 1/2".
We show P(T) holds for all trees T by structural induction on T.

Base Case: P(•): By definition of leaves(•), leaves(•) = 1 and size(•) = 1.
So, leaves(•) = 1 ≥ 1/2 + 1/2 = size(•)/2 + 1/2, so P(•) holds.

Inductive Hypothesis: Suppose P(L) and P(R) hold for some arbitrary trees L and R,
i.e., leaves(L) ≥ size(L)/2 + 1/2, leaves(R) ≥ size(R)/2 + 1/2

Inductive Step: Goal: Show P(Tree(•, L, R)): leaves(Tree(•, L, R)) ≥ size(Tree(•, L, R))/2 + 1/2

leaves(Tree(•, L, R)) = leaves(L) + leaves(R)	definition of leaves
≥ (size(L)/2 + 1/2) + (size(R)/2 + 1/2)	by Inductive Hypothesis
= (1/2 + size(L)/2 + size(R)/2) + 1/2	
= (1 + size(L) + size(R)) / 2 + 1/2	
= size(T)/2 + 1/2	definition of size

Conclusion: Therefore P(T) holds for all trees T by the principle of induction.

Problem 2b – Structural Induction on Trees

Prove that leaves(T) ≥ size(T)/2 + 1/2 for all Trees T

For a tree T, let P(T) be "leaves(T) ≥ size(T)/2 + 1/2".
We show P(T) holds for all trees T by structural induction on T.

Base Case: P(•): By definition of leaves(•), leaves(•) = 1 and size(•) = 1.
So, leaves(•) = 1 ≥ 1/2 + 1/2 = size(•)/2 + 1/2, so P(•) holds.

Inductive Hypothesis: Suppose P(L) and P(R) hold for some arbitrary trees L and R,
i.e., leaves(L) ≥ size(L)/2 + 1/2, leaves(R) ≥ size(R)/2 + 1/2

Inductive Step: Goal: Show P(Tree(•, L, R)): leaves(Tree(•, L, R)) ≥ size(Tree(•, L, R))/2 + 1/2

leaves(Tree(•, L, R)) = leaves(L) + leaves(R)	definition of leaves
≥ (size(L)/2 + 1/2) + (size(R)/2 + 1/2)	by Inductive Hypothesis
= (1/2 + size(L)/2 + size(R)/2) + 1/2	
= (1 + size(L) + size(R)) / 2 + 1/2	
= size(T)/2 + 1/2	definition of size

So, P(Tree(•, L, R)) holds!
Conclusion: Therefore P(T) holds for all trees T by the principle of induction.

Functions

One-To-One Proofs

One-to-one (aka injection)

A function f is one-to-one iff

$$\forall a \forall b (f(a) = f(b) \rightarrow a = b)$$

This is a for-all statement! We know how to prove this.

1. Let a and b be arbitrary elements in your **domain**.
2. Suppose that f(a) = f(b); i.e. both a and b output the same value when passed into f
3. Show that a = b; i.e. if two inputs map to the same input, the two inputs must be the same value.

One-To-One Proofs

Onto (aka surjection)

A function $f: A \to B$ is onto iff

$\forall b \in B \exists a \in A (b = f(a))$

This is a for-all statement with an exists! We know how to prove this.

1. Let b be an arbitrary element in your **co-domain**.
2. Show that for some a, f(a) = b; i.e. for an arbitrary output, there is some input mapping to that output
3. Since b was arbitrary, conclude that all elements of the **co-domain** have at least one element in the domain that map to it.

Problem 4 - One-to-One and Onto

Determine if the following functions are (1) one-to-one and (2) onto. For each claim: provide a proof if true, otherwise give a counterexample (you must also explain why the counterexample works).

(i) $f : \mathbb{N} \to \mathbb{N}$, $f(x) = x^2$

(ii) $f : \mathbb{R} \to \mathbb{R}$, $f(x) = x^2$

(iii) $f : \mathbb{R}^+ \to \mathbb{R}^+$, $f(x) = x^2$, where $\mathbb{R}^+ = \{x : x \in \mathbb{R} \wedge x \geq 0\}$, i.e., the set of non-negative real numbers.

Notice that the domain and co-domain matter! You have to know both to tell whether the function is one-to-one or onto.

Work on this problem with the people around you!

Problem 4 - One-to-One and Onto

Determine if the following functions are (1) one-to-one and (2) onto. For each claim: provide a proof if true, otherwise give a counterexample (you must also explain why the counterexample works).

(i) $f : \mathbb{N} \to \mathbb{N}$, $f(x) = x^2$

Problem 4 - One-to-One and Onto

Determine if the following functions are (1) one-to-one and (2) onto. For each claim: provide a proof if true, otherwise give a counterexample (you must also explain why the counterexample works).

(i) $f : \mathbb{N} \to \mathbb{N}$, $f(x) = x^2$

One-to-One (injective): Let $x, y \in \mathbb{N}$ be arbitrary. Suppose $f(x) = f(y)$.

Problem 4 - One-to-One and Onto

Determine if the following functions are (1) one-to-one and (2) onto. For each claim: provide a proof if true, otherwise give a counterexample (you must also explain why the counterexample works).

(i) $f : \mathbb{N} \to \mathbb{N}$, $f(x) = x^2$

One-to-One (injective): Let $x, y \in \mathbb{N}$ be arbitrary. Suppose $f(x) = f(y)$. Then $x^2 = y^2$, so either $x = y$ or $x = -y$. Since $x, y \in \mathbb{N}$, either x and y are both 0 or they're both positive integers.

Problem 4 - One-to-One and Onto

Determine if the following functions are (1) one-to-one and (2) onto. For each claim: provide a proof if true, otherwise give a counterexample (you must also explain why the counterexample works).

(i) $f : \mathbb{N} \to \mathbb{N}$, $f(x) = x^2$

One-to-One (injective): Let $x, y \in \mathbb{N}$ be arbitrary. Suppose $f(x) = f(y)$. Then $x^2 = y^2$, so either $x = y$ or $x = -y$. Since $x, y \in \mathbb{N}$, either x and y are both 0 or they're both positive integers.

- If $x = 0$ then $y = 0 = -y$, and clearly $x = y$.

Problem 4 - One-to-One and Onto

Determine if the following functions are (1) one-to-one and (2) onto. For each claim: provide a proof if true, otherwise give a counterexample (you must also explain why the counterexample works).

(i) $f : \mathbb{N} \to \mathbb{N}$, $f(x) = x^2$

One-to-One (injective): Let $x, y \in \mathbb{N}$ be arbitrary. Suppose $f(x) = f(y)$. Then $x^2 = y^2$, so either $x = y$ or $x = -y$. Since $x, y \in \mathbb{N}$, either x and y are both 0 or they're both positive integers.

- If $x = 0$ then $y = 0 = -y$, and clearly $x = y$.
- If both are positive, then $x = -y$ is impossible since x is positive and $-y$ is negative. So we must have $x = y$.

Problem 4 - One-to-One and Onto

Determine if the following functions are (1) one-to-one and (2) onto. For each claim: provide a proof if true, otherwise give a counterexample (you must also explain why the counterexample works).

(i) $f : \mathbb{N} \to \mathbb{N}$, $f(x) = x^2$

One-to-One (injective): Let $x, y \in \mathbb{N}$ be arbitrary. Suppose $f(x) = f(y)$. Then $x^2 = y^2$, so either $x = y$ or $x = -y$. Since $x, y \in \mathbb{N}$, either x and y are both 0 or they're both positive integers.

- If $x = 0$ then $y = 0 = -y$, and clearly $x = y$.
- If both are positive, then $x = -y$ is impossible since x is positive and $-y$ is negative. So we must have $x = y$.

In both cases, we have $x = y$. Since x and y were arbitrary, f is one-to-one.

Problem 4 - One-to-One and Onto

Determine if the following functions are (1) one-to-one and (2) onto. For each claim: provide a proof if true, otherwise give a counterexample (you must also explain why the counterexample works).

(i) $f : \mathbb{N} \to \mathbb{N}$, $f(x) = x^2$

One-to-One (injective): Let $x, y \in \mathbb{N}$ be arbitrary. Suppose $f(x) = f(y)$. Then $x^2 = y^2$, so either $x = y$ or $x = -y$. Since $x, y \in \mathbb{N}$, either x and y are both 0 or they're both positive integers.

- If $x = 0$ then $y = 0 = -y$, and clearly $x = y$.
- If both are positive, then $x = -y$ is impossible since x is positive and $-y$ is negative. So we must have $x = y$.

In both cases, we have $x = y$. Since x and y were arbitrary, f is one-to-one.

Not onto (not surjective): $5 \in \mathbb{N}$, our codomain, but there is no natural number x such that $f(x) = x^2 = 5$.

Problem 4 - One-to-One and Onto

(ii) $f : \mathbb{R} \to \mathbb{R}$, $f(x) = x^2$

Problem 4 - One-to-One and Onto

(ii) $f : \mathbb{R} \to \mathbb{R}$, $f(x) = x^2$

Not one-to-one (not injective): $f(-2) = 4 = f(2)$**, but** $-2 \neq 2$**.**

Problem 4 - One-to-One and Onto

(ii) $f : \mathbb{R} \to \mathbb{R}$, $f(x) = x^2$

Not one-to-one (not injective): $f(-2) = 4 = f(2)$**, but** $-2 \neq 2$**.**

Not onto (not surjective): $-5 \in \mathbb{R}$, our codomain, but there is no real number x such that $f(x) = x^2 = -5$ since all real numbers are non-negative when squared.

Problem 4 - One-to-One and Onto

(iii) $f : \mathbb{R}^+ \to \mathbb{R}^+$, $f(x) = x^2$, where $\mathbb{R}^+ = \{x : x \in \mathbb{R} \wedge x \geq 0\}$, i.e., the set of non-negative real numbers.

Problem 4 - One-to-One and Onto

(iii) $f : \mathbb{R}^+ \to \mathbb{R}^+$, $f(x) = x^2$, where $\mathbb{R}^+ = \{x : x \in \mathbb{R} \wedge x \geq 0\}$, i.e., the set of non-negative real numbers.

One-to-one (injective): Let $x, y \in \mathbb{R}^+$ be arbitrary. Suppose $f(x) = f(y)$.

Problem 4 - One-to-One and Onto

(iii) $f : \mathbb{R}^+ \to \mathbb{R}^+$, $f(x) = x^2$, where $\mathbb{R}^+ = \{x : x \in \mathbb{R} \wedge x \geq 0\}$, i.e., the set of non-negative real numbers.

One-to-one (injective): Let $x, y \in \mathbb{R}^+$ be arbitrary. Suppose $f(x) = f(y)$. Then $x^2 = y^2$, so either $x = y$ or $x = -y$.

Problem 4 - One-to-One and Onto

(iii) $f : \mathbb{R}^+ \to \mathbb{R}^+$, $f(x) = x^2$, where $\mathbb{R}^+ = \{x : x \in \mathbb{R} \wedge x \geq 0\}$, i.e., the set of non-negative real numbers.

One-to-one (injective): Let $x, y \in \mathbb{R}^+$ be arbitrary. Suppose $f(x) = f(y)$. Then $x^2 = y^2$, so either $x = y$ or $x = -y$. Since $x, y \in \mathbb{R}^+$, either x and y are both 0 or they're both positive real numbers.

Problem 4 - One-to-One and Onto

(iii) $f : \mathbb{R}^+ \to \mathbb{R}^+$, $f(x) = x^2$, where $\mathbb{R}^+ = \{x : x \in \mathbb{R} \wedge x \geq 0\}$, i.e., the set of non-negative real numbers.

One-to-one (injective): Let $x, y \in \mathbb{R}^+$ be arbitrary. Suppose $f(x) = f(y)$. Then $x^2 = y^2$, so either $x = y$ or $x = -y$. Since $x, y \in \mathbb{R}^+$, either x and y are both 0 or they're both positive real numbers.

- If $x = 0$ then $y = 0 = -y$, and clearly $x = y$.

Problem 4 - One-to-One and Onto

(iii) $f : \mathbb{R}^+ \to \mathbb{R}^+$, $f(x) = x^2$, where $\mathbb{R}^+ = \{x : x \in \mathbb{R} \wedge x \geq 0\}$, i.e., the set of non-negative real numbers.

One-to-one (injective): Let $x, y \in \mathbb{R}^+$ be arbitrary. Suppose $f(x) = f(y)$. Then $x^2 = y^2$, so either $x = y$ or $x = -y$. Since $x, y \in \mathbb{R}^+$, either x and y are both 0 or they're both positive real numbers.

- If $x = 0$ then $y = 0 = -y$, and clearly $x = y$.
- If both are positive, then $x = -y$ is impossible since x is positive and $-y$ is negative. So we must have $x = y$.

Problem 4 - One-to-One and Onto

(iii) $f : \mathbb{R}^+ \to \mathbb{R}^+$, $f(x) = x^2$, where $\mathbb{R}^+ = \{x : x \in \mathbb{R} \wedge x \geq 0\}$, i.e., the set of non-negative real numbers.

One-to-one (injective): Let $x, y \in \mathbb{R}^+$ be arbitrary. Suppose $f(x) = f(y)$. Then $x^2 = y^2$, so either $x = y$ or $x = -y$. Since $x, y \in \mathbb{R}^+$, either x and y are both 0 or they're both positive real numbers.

- If $x = 0$ then $y = 0 = -y$, and clearly $x = y$.
- If both are positive, then $x = -y$ is impossible since x is positive and $-y$ is negative. So we must have $x = y$.

In both cases, we have $x = y$. Since x and y were arbitrary, f is one-to-one.

Problem 4 - One-to-One and Onto

(iii) $f : \mathbb{R}^+ \to \mathbb{R}^+$, $f(x) = x^2$, where $\mathbb{R}^+ = \{x : x \in \mathbb{R} \wedge x \geq 0\}$, i.e., the set of non-negative real numbers.

One-to-one (injective): Let $x, y \in \mathbb{R}^+$ be arbitrary. Suppose $f(x) = f(y)$. Then $x^2 = y^2$, so either $x = y$ or $x = -y$. Since $x, y \in \mathbb{R}^+$, either x and y are both 0 or they're both positive real numbers.

- If $x = 0$ then $y = 0 = -y$, and clearly $x = y$.
- If both are positive, then $x = -y$ is impossible since x is positive and $-y$ is negative. So we must have $x = y$.

In both cases, we have $x = y$. Since x and y were arbitrary, f is one-to-one.

Onto (surjective): Let $x \in \mathbb{R}^+$ be arbitrary. Since x is a positive real number, $\sqrt{x}$ is also a positive real number (i.e., is in our domain).

Problem 4 - One-to-One and Onto

(iii) $f : \mathbb{R}^+ \to \mathbb{R}^+$, $f(x) = x^2$, where $\mathbb{R}^+ = \{x : x \in \mathbb{R} \wedge x \geq 0\}$, i.e., the set of non-negative real numbers.

One-to-one (injective): Let $x, y \in \mathbb{R}^+$ be arbitrary. Suppose $f(x) = f(y)$. Then $x^2 = y^2$, so either $x = y$ or $x = -y$. Since $x, y \in \mathbb{R}^+$, either x and y are both 0 or they're both positive real numbers.

- If $x = 0$ then $y = 0 = -y$, and clearly $x = y$.
- If both are positive, then $x = -y$ is impossible since x is positive and $-y$ is negative. So we must have $x = y$.

In both cases, we have $x = y$. Since x and y were arbitrary, f is one-to-one.

Onto (surjective): Let $x \in \mathbb{R}^+$ be arbitrary. Since x is a positive real number, $\sqrt{x}$ is also a positive real number (i.e., is in our domain). Then $f(\sqrt(x)) = (\sqrt(x))^2 = x$.

Problem 4 - One-to-One and Onto

(iii) $f : \mathbb{R}^+ \to \mathbb{R}^+$, $f(x) = x^2$, where $\mathbb{R}^+ = \{x : x \in \mathbb{R} \wedge x \geq 0\}$, i.e., the set of non-negative real numbers.

One-to-one (injective): Let $x, y \in \mathbb{R}^+$ be arbitrary. Suppose $f(x) = f(y)$. Then $x^2 = y^2$, so either $x = y$ or $x = -y$. Since $x, y \in \mathbb{R}^+$, either x and y are both 0 or they're both positive real numbers.

- If $x = 0$ then $y = 0 = -y$, and clearly $x = y$.
- If both are positive, then $x = -y$ is impossible since x is positive and $-y$ is negative. So we must have $x = y$.

In both cases, we have $x = y$. Since x and y were arbitrary, f is one-to-one.

Onto (surjective): Let $x \in \mathbb{R}^+$ be arbitrary. Since x is a positive real number, $\sqrt{x}$ is also a positive real number (i.e., is in our domain). Then $f(\sqrt(x)) = (\sqrt(x))^2 = x$. Therefore, x is the image of some input under f. Since x was arbitrary, f is surjective.

Regular Expressions

Regular Expressions

Basis:

- ε : The empty string itself matches the pattern (and nothing else does).
- $\varnothing$: No strings match this pattern
- a for any $a \in \Sigma$: The character itself matching this pattern

Recursive:

- If A, B are regular expressions then $(A \cup B)$ is a regular expression
  - matched by any string that matches A or that matches B [or both]
- If A, B are regular expressions then AB is a regular expression
  - matched by any string x such that $x = yz$, y matches A and z matches B
- If A is a regular expression, then A^* is a regular expression
  - matched by any string that can be divided into 0 or more strings that match A

Regular Expressions

A regular expression is a recursively defined set of strings that form a language.

A regular expression will generate all strings in a language, and won't generate any strings that ARE NOT in the language

Hints:

- If writing a regex for the language is tricky, target one requirement of the language and then build the regex from there
- Come up with a few examples of strings that ARE and ARE NOT in your language. After writing your regex, check that it CAN generate all of your examples in the language, and it CAN'T generate those that are not in the language.

Problem 6 – Regular Expressions

a) Write a regular expression that matches base 10 numbers (e.g., there should be no leading zeroes).

b) Write a regular expression that matches all base-3 numbers that are divisible by 3.

c) Write a regular expression that matches all binary strings that contain the substring "111", but not the substring "000".

d) Write a regular expression that matches all binary strings that do not have any consecutive 0's or 1's.

e) Write a regular expression that matches all binary strings of the form $1^k y$, where $k \geq 1$ and $y \in \{0,1\}^*$ has at least k 1's.

Work on this a,b,d,e with the people around you!

Problem 6 – Regular Expressions

a) Write a regular expression that matches base 10 numbers (e.g., there should be no leading zeroes).

base-10 numbers:

Our everyday numbers! Notice we have 10 symbols (0-9) to represent numbers.

256: $(\mathbf{2} * 10^2) + (\mathbf{5} * 10^1) + (\mathbf{6} * 10^0)$

base-2 numbers: (binary)

10: $(\mathbf{1} * 2^1) + (\mathbf{0} * 2^0)$

Problem 6 – Regular Expressions

a) Write a regular expression that matches base 10 numbers (e.g., there should be no leading zeroes).

Representing numbers all possible *strings* using numbers 0-9:

Problem 6 – Regular Expressions

a) Write a regular expression that matches base 10 numbers (e.g., there should be no leading zeroes).

Representing numbers all possible *strings* using numbers 0-9:

$$(0 \cup 1 \cup 2 \cup 3 \cup 4 \cup 5 \cup 6 \cup 7 \cup 8 \cup 9)*$$

Problem 6 – Regular Expressions

a) Write a regular expression that matches base 10 numbers (e.g., there should be no leading zeroes).

Representing numbers all possible *strings* using numbers 0-9:

(0 U 1 U 2 U 3 U 4 U 5 U 6 U 7 U 8 U 9)*

⚠️ “0101” or “091” **are not** Base-10 numbers

Problem 6 – Regular Expressions

a) Write a regular expression that matches base 10 numbers (e.g., there should be no leading zeroes).

Representing numbers all possible *strings* using numbers 0-9:

(0 ∪ 1 ∪ 2 ∪ 3 ∪ 4 ∪ 5 ∪ 6 ∪ 7 ∪ 8 ∪ 9)*

⚠️ “0101” or “091” **are not** Base-10 numbers

All possible *strings* using numbers 0-9 that never start with 0

Problem 6 – Regular Expressions

a) Write a regular expression that matches base 10 numbers (e.g., there should be no leading zeroes).

Representing numbers all possible *strings* using numbers 0-9:

$$(0 \cup 1 \cup 2 \cup 3 \cup 4 \cup 5 \cup 6 \cup 7 \cup 8 \cup 9)*$$

⚠️ “0101” or “091” **are not** Base-10 numbers

All possible *strings* using numbers 0-9 that never start with 0

$$(1 \cup 2 \cup 3 \cup 4 \cup 5 \cup 6 \cup 7 \cup 8 \cup 9)(0 \cup 1 \cup 2 \cup 3 \cup 4 \cup 5 \cup 6 \cup 7 \cup 8 \cup 9)*$$

Problem 6 – Regular Expressions

a) Write a regular expression that matches base 10 numbers (e.g., there should be no leading zeroes).

Representing numbers all possible *strings* using numbers 0-9:

$$(0 \cup 1 \cup 2 \cup 3 \cup 4 \cup 5 \cup 6 \cup 7 \cup 8 \cup 9)*$$

⚠️ “0101” or “091” **are not** Base-10 numbers

All possible *strings* using numbers 0-9 that never start with 0

$$(1 \cup 2 \cup 3 \cup 4 \cup 5 \cup 6 \cup 7 \cup 8 \cup 9)(0 \cup 1 \cup 2 \cup 3 \cup 4 \cup 5 \cup 6 \cup 7 \cup 8 \cup 9)*$$

⚠️ “0” **is** a Base-10 number not considered

Problem 6 – Regular Expressions

a) Write a regular expression that matches base 10 numbers (e.g., there should be no leading zeroes).

Representing numbers all possible *strings* using numbers 0-9:

$$(0 \cup 1 \cup 2 \cup 3 \cup 4 \cup 5 \cup 6 \cup 7 \cup 8 \cup 9)*$$

⚠️ “0101” or “091” **are not** Base-10 numbers

All possible *strings* using numbers 0-9 that never start with 0

$$(1 \cup 2 \cup 3 \cup 4 \cup 5 \cup 6 \cup 7 \cup 8 \cup 9)(0 \cup 1 \cup 2 \cup 3 \cup 4 \cup 5 \cup 6 \cup 7 \cup 8 \cup 9)*$$

⚠️ “0” **is** a Base-10 number not considered

All possible *strings* using numbers 0-9 that never start with 0 *or* is 0

Problem 6 – Regular Expressions

a) Write a regular expression that matches base 10 numbers (e.g., there should be no leading zeroes).

Representing numbers all possible *strings* using numbers 0-9:

$(0 \cup 1 \cup 2 \cup 3 \cup 4 \cup 5 \cup 6 \cup 7 \cup 8 \cup 9)*$

⚠️ “0101” or “091” **are not** Base-10 numbers

All possible *strings* using numbers 0-9 that never start with 0

$(1 \cup 2 \cup 3 \cup 4 \cup 5 \cup 6 \cup 7 \cup 8 \cup 9)(0 \cup 1 \cup 2 \cup 3 \cup 4 \cup 5 \cup 6 \cup 7 \cup 8 \cup 9)*$

⚠️ “0” **is** a Base-10 number not considered

All possible *strings* using numbers 0-9 that never start with 0 *or* is 0

$0 \cup ((1 \cup 2 \cup 3 \cup 4 \cup 5 \cup 6 \cup 7 \cup 8 \cup 9)(0 \cup 1 \cup 2 \cup 3 \cup 4 \cup 5 \cup 6 \cup 7 \cup 8 \cup 9)*)$

Problem 6 – Regular Expressions

a) Write a regular expression that matches base 10 numbers (e.g., there should be no leading zeroes).

Representing numbers all possible *strings* using numbers 0-9:

$$(0 \cup 1 \cup 2 \cup 3 \cup 4 \cup 5 \cup 6 \cup 7 \cup 8 \cup 9)*$$

⚠️ “0101” or “091” **are not** Base-10 numbers

All possible *strings* using numbers 0-9 that never start with 0

$$(1 \cup 2 \cup 3 \cup 4 \cup 5 \cup 6 \cup 7 \cup 8 \cup 9)(0 \cup 1 \cup 2 \cup 3 \cup 4 \cup 5 \cup 6 \cup 7 \cup 8 \cup 9)*$$

⚠️ “0” **is** a Base-10 number not considered

All possible *strings* using numbers 0-9 that never start with 0 *or* is 0

$$0 \cup ((1 \cup 2 \cup 3 \cup 4 \cup 5 \cup 6 \cup 7 \cup 8 \cup 9)(0 \cup 1 \cup 2 \cup 3 \cup 4 \cup 5 \cup 6 \cup 7 \cup 8 \cup 9)*)$$

✅Generates only all possible Base-10 numbers

Problem 6 – Regular Expressions

b) Write a regular expression that matches all base-3 numbers that are divisible by 3.

Problem 6 – Regular Expressions

b) Write a regular expression that matches all base-3 numbers that are divisible by 3.

Write a regular expression that matches all base-3 numbers

Problem 6 – Regular Expressions

b) Write a regular expression that matches all base-3 numbers that are divisible by 3.

Write a regular expression that matches all base-3 numbers

$0 \cup ((1 \cup 2)(0 \cup 1 \cup 2)*)$

✅Generates only all possible Base-3 numbers

Problem 6 – Regular Expressions

b) Write a regular expression that matches all base-3 numbers that are divisible by 3.

Write a regular expression that matches all base-3 numbers

$$0 \cup ((1 \cup 2)(0 \cup 1 \cup 2)*)$$

Generates only all possible Base-3 numbers

…divisible by 3

Problem 6 – Regular Expressions

b) Write a regular expression that matches all base-3 numbers that are divisible by 3.

Write a regular expression that matches all base-3 numbers

$0 \cup ((1 \cup 2)(0 \cup 1 \cup 2)*)$

Generates only all possible Base-3 numbers

…divisible by 3

Hint: you know that Base-10 numbers are divisible by 10 when they end in 0 (10, 20, 30, 40…)

Problem 6 – Regular Expressions

b) Write a regular expression that matches all base-3 numbers that are divisible by 3.

Write a regular expression that matches all base-3 numbers

$0 \cup ((1 \cup 2)(0 \cup 1 \cup 2)*)$

Generates only all possible Base-3 numbers

…divisible by 3

Hint: you know that Base-10 numbers are divisible by 10 when they end in 0 (10, 20, 30, 40…)

$0 \cup ((1 \cup 2)(0 \cup 1 \cup 2)*\mathbf{0})$

✅all possible Base-3 numbers divisible by 3

Problem 6 – Regular Expressions

c) Write a regular expression that matches all binary strings that contain the substring “111”, but not the substring “000”.

Problem 6 – Regular Expressions

c) Write a regular expression that matches all binary strings that contain the substring “111”, but not the substring “000”.

all binary strings that contain the substring “111”

Problem 6 – Regular Expressions

c) Write a regular expression that matches all binary strings that contain the substring “111”, but not the substring “000”.

all binary strings that contain the substring “111”

$(0 \cup 1)^* \, 111 \, (0 \cup 1)^*$

⚠️ The Kleene-star has us generating any number of 0’s

Problem 6 – Regular Expressions

c) Write a regular expression that matches all binary strings that contain the substring “111”, but not the substring “000”.

all binary strings that contain the substring “111”

$(0 \cup 1)^* 111 (0 \cup 1)^*$

⚠️ The Kleene-star has us generating any number of 0’s

…without the substring “000”

Use careful case-work to modify this and produce only 0,1,or two 0’s

Problem 6 – Regular Expressions

c) Write a regular expression that matches all binary strings that contain the substring “111”, but not the substring “000”.

all binary strings that contain the substring “111”

(0 ∪ 1)* 111 (0 ∪ 1)*

⚠️ The Kleene-star has us generating any number of 0’s

…without the substring “000”

Use careful case-work to modify this and produce only 0,1,or two 0’s

(0 ∪ 00 ∪ ε) (1)* 111 **(0 ∪ 00 ∪ ε)** (1)*

Problem 6 – Regular Expressions

c) Write a regular expression that matches all binary strings that contain the substring “111”, but not the substring “000”.

all binary strings that contain the substring “111”

(0 ∪ 1)* 111 (0 ∪ 1)*

⚠️ The Kleene-star has us generating any number of 0’s

…without the substring “000”

Use careful case-work to modify this and produce only 0,1,or two 0’s

(0 ∪ 00 ∪ ε) (1)* 111 **(0 ∪ 00 ∪ ε)** (1)*

⚠️ Cannot produce 1’s with “0” or “00” like “1011101”

Problem 6 – Regular Expressions

c) Write a regular expression that matches all binary strings that contain the substring “111”, but not the substring “000”.

all binary strings that contain the substring “111”

(0 ∪ 1)* 111 (0 ∪ 1)*

⚠️ The Kleene-star has us generating any number of 0’s

…without the substring “000”

Use careful case-work to modify this and produce only 0,1,or two 0’s

(0 ∪ 00 ∪ ε) (1)* 111 **(0 ∪ 00 ∪ ε)** (1)*

⚠️ Cannot produce 1’s with “0” or “00” like “1011101”

(0 ∪ 00 ∪ ε) **(01 ∪ 001 ∪ 1)*** 111 (0 ∪ 00 ∪ ε) **(01 ∪ 001 ∪ 1)***

Problem 6 – Regular Expressions

c) Write a regular expression that matches all binary strings that contain the substring “111”, but not the substring “000”.

all binary strings that contain the substring “111”

(0 ∪ 1)* 111 (0 ∪ 1)*

⚠ The Kleene-star has us generating any number of 0’s

…without the substring “000”

Use careful case-work to modify this and produce only 0,1,or two 0’s

(0 ∪ 00 ∪ ε) (1)* 111 **(0 ∪ 00 ∪ ε)** (1)*

⚠ Cannot produce 1’s with “0” or “00” like “1011101”

(0 ∪ 00 ∪ ε) **(01 ∪ 001 ∪ 1)*** 111 (0 ∪ 00 ∪ ε) **(01 ∪ 001 ∪ 1)***

⚠ Generates “000” like “00 01 111”

Problem 6 – Regular Expressions

c) Write a regular expression that matches all binary strings that contain the substring “111”, but not the substring “000”.

all binary strings that contain the substring “111”

(0 ∪ 1)* 111 (0 ∪ 1)*

⚠️ The Kleene-star has us generating any number of 0’s

…without the substring “000”

Use careful case-work to modify this and produce only 0,1,or two 0’s

(0 ∪ 00 ∪ ε) (1)* 111 **(0 ∪ 00 ∪ ε)** (1)*

⚠️ Cannot produce 1’s with “0” or “00” like “1011101”

(0 ∪ 00 ∪ ε) **(01 ∪ 001 ∪ 1)*** 111 (0 ∪ 00 ∪ ε) **(01 ∪ 001 ∪ 1)***

⚠️ Generates “000” like “00 01 111”

(01 ∪ 001 ∪ 1)* (0 ∪ 00 ∪ ε) 111 (01 ∪ 001 ∪ 1)* (0 ∪ 00 ∪ ε)

✅ all binary strings with “111” and without “000”

Problem 6 – Regular Expressions

c) Write a regular expression that matches all binary strings that contain the substring “111”, but not the substring “000”.

all binary strings that contain the substring “111”

(0 ∪ 1)* 111 (0 ∪ 1)*

⚠️ The Kleene-star has us generating any number of 0’s

…without the substring “000”

Use careful case-work to modify this and produce only 0,1,or two 0’s

(0 ∪ 00 ∪ ε) (1)* 111 **(0 ∪ 00 ∪ ε)** (1)*

⚠️ Cannot produce 1’s with “0” or “00” like “1011101”

(0 ∪ 00 ∪ ε) **(01 ∪ 001 ∪ 1)*** 111 (0 ∪ 00 ∪ ε) **(01 ∪ 001 ∪ 1)***

⚠️ Generates “000” like “00 01 111”

(01 ∪ 001 ∪ 1)* (0 ∪ 00 ∪ ε) 111 (01 ∪ 001 ∪ 1)* (0 ∪ 00 ✅ all binary strings with “111” and without “000”

(01 ∪ 001 ∪ 1)* (0 ∪ 00 ∪ ε) 111 (01 ∪ 001 ∪ 1)* (0 ∪ 00 ∪ ε)

Problem 6 – Regular Expressions

d) Write a regular expression that matches all binary strings that do not have any consecutive 0's or 1's.

Problem 6 – Regular Expressions

d) Write a regular expression that matches all binary strings that do not have any consecutive 0's or 1's.

Step 1: Write out basic and more intricate cases

Problem 6 – Regular Expressions

d) Write a regular expression that matches all binary strings that do not have any consecutive 0's or 1's.

Step 1: Write out basic and more intricate cases

Accepted Strings	Rejected Strings
ε	00
1	11
10101	101011
0101	0100

Problem 6 – Regular Expressions

d) Write a regular expression that matches all binary strings that do not have any consecutive 0's or 1's.

Step 1: Write out basic and more intricate cases

Accepted Strings	Rejected Strings
ε	00
1	11
10101	101011
0101	0100

Step 2: Find a pattern!

Problem 6 – Regular Expressions

d) Write a regular expression that matches all binary strings that do not have any consecutive 0's or 1's.

Step 1: Write out basic and more intricate cases

Accepted Strings	Rejected Strings
ε	00
1	11
10101	101011
0101	0100

Step 2: Find a pattern!

strings can be generated from **either a series** of **"01" or "10" substrings**

(1) Using the "01" substring, one additional 0 can be added

(1) Using the "10" substring, one additional 1 can be added

Problem 6 – Regular Expressions

d) Write a regular expression that matches all binary strings that do not have any consecutive 0's or 1's.

Step 1: Write out basic and more intricate cases

Accepted Strings	Rejected Strings
ε	00
1	11
10101	101011
0101	0100

Step 2: Find a pattern!

Strings that start with 0 can be generated from a series of "01" substrings, but we might need an extra 0 at the end.

Strings that start with 1 can be generated from a series of "10" substrings, but we might need an extra 1 at the end.

Problem 6 – Regular Expressions

d) Write a regular expression that matches all binary strings that do not have any consecutive 0's or 1's.

Step 3: Write out the expression with the two cases we found

Problem 6 – Regular Expressions

d) Write a regular expression that matches all binary strings that do not have any consecutive 0's or 1's.

Step 3: Write out the expression with the two cases we found

$$((01)* (0 \cup \varepsilon)) \cup ((10)* (1 \cup \varepsilon))$$

Problem 6 – Regular Expressions

e) Write a regular expression that matches all binary strings of the form $1^k y$ where $k \geq 1$ and $y \in \{0, 1\}^*$ has at least k 1's.

Problem 6 – Regular Expressions

e) Write a regular expression that matches all binary strings of the form $1^k y$ where $k \geq 1$ and $y \in \{0, 1\}^*$ has at least k 1's.

$1(0 \cup 1)^* 1(0 \cup 1)^*$

Explanation: While it may seem like we need to keep track of how many 1's there are, it turns out that we don't. If k is 1, then y just needs to match any binary string with **at least one** 1. If k is greater than 1, we can treat the extra 1's at the start of the string as being part of y. This means y still only needs to match binary strings with **at least one** 1. We are essentially always treating k as 1, because this makes it much easier to match the rest of the string.

Since y just needs to have one 1, we can write y as $(0 \cup 1)^* 1(0 \cup 1)^*$. Then, we can add our leading 1 to get the final solution.

Context-Free Grammars

Context-Free Grammars

A context free grammar (CFG) is a finite set of production rules over:

- An alphabet Σ of “terminal symbols”
- A finite set V of “nonterminal symbols”
- A start symbol (one of the elements of V) usually denoted S

A production rule for a nonterminal $A \in V$ takes the form

- $A \rightarrow w1 \mid w2 \mid \ldots \mid wk$

Where each $wi \in V \cup \Sigma^*$ is a string of nonterminals and terminals.

Problem 7 – CFGs

Write a context-free grammar to match each of these languages.

a) All binary strings that start with 11.

b) All binary strings that contain at most one 1.

c) All strings over 0, 1, 2 with the same number of 1s and 0s and exactly one 2.

Work on this problem with the people around you!

Problem 7 – CFGs

a) All binary strings that start with 11.

Problem 7 – CFGs

a) All binary strings that start with 11.

Thinking back to regular expressions…

Problem 7 – CFGs

a) All binary strings that start with 11.

Thinking back to regular expressions…

11 (0 ∪ 1)*

Problem 7 – CFGs

a) All binary strings that start with 11.

Thinking back to regular expressions…

11 (0 ∪ 1)*

Now generate the CFG…

Problem 7 – CFGs

a) All binary strings that start with 11.

Thinking back to regular expressions…

11 (0 ∪ 1)*

Now generate the CFG…

S → 11**T**
T → 1**T** | 0**T** | **ε**

Problem 7 – CFGs

b) All binary strings that contain at most one 1.

Problem 7 – CFGs

b) All binary strings that contain at most one 1.

Thinking back to Regular expressions…

Problem 7 – CFGs

b) All binary strings that contain at most one 1.

Thinking back to Regular expressions…

$0^* (1 \cup \varepsilon) 0^*$

Problem 7 – CFGs

b) All binary strings that contain at most one 1.

Thinking back to Regular expressions…

$0^* (1 \cup \varepsilon) 0^*$

Now generate the CFG…

Problem 7 – CFGs

b) All binary strings that contain at most one 1.

Thinking back to Regular expressions…

$0^* (1 \cup \varepsilon) 0^*$

Now generate the CFG…

$S \rightarrow ABA$
$A \rightarrow 0A \mid \varepsilon$
$B \rightarrow 1 \mid \varepsilon$

Problem 7 – CFGs

b) All binary strings that contain at most one 1.

Thinking back to Regular expressions…

$0^* (1 \cup \varepsilon) 0^*$

Now generate the CFG…

$S \rightarrow ABA$
$A \rightarrow 0A \mid \varepsilon$
$B \rightarrow 1 \mid \varepsilon$

Alternate solution:

$S \rightarrow 0S \mid S0 \mid 1 \mid \varepsilon$

Problem 7 – CFGs

c) All strings over 0, 1, 2 with the same number of 1s and 0s and exactly one 2.

Problem 7 – CFGs

c) All strings over 0, 1, 2 with the same number of 1s and 0s and exactly one 2.

First, let's match strings with the same number of 1s and 0s.

Problem 7 – CFGs

c) All strings over 0, 1, 2 with the same number of 1s and 0s and exactly one 2.

First, let's match strings with the same number of 1s and 0s.
S → ε | 1S0 | 0S1

Problem 7 – CFGs

c) All strings over 0, 1, 2 with the same number of 1s and 0s and exactly one 2.

First, let's match strings with the same number of 1s and 0s.
$S \rightarrow \varepsilon \mid 1S0 \mid 0S1$

What about 1001? We need a way to make our string symmetric.

Problem 7 – CFGs

c) All strings over 0, 1, 2 with the same number of 1s and 0s and exactly one 2.

First, let's match strings with the same number of 1s and 0s.
S → ε | 1S0 | 0S1

What about 1001? We need a way to make our string symmetric.

S → ε | 1S0 | 0S1 | SS

Problem 7 – CFGs

c) All strings over 0, 1, 2 with the same number of 1s and 0s and exactly one 2.

First, let's match strings with the same number of 1s and 0s.
S → ε | 1S0 | 0S1

What about 1001? We need a way to make our string symmetric.

S → ε | 1S0 | 0S1 | SS

Now, we need to add the "exactly one 2" constraint. Let's define our previous work as **T.**

Problem 7 – CFGs

c) All strings over 0, 1, 2 with the same number of 1s and 0s and exactly one 2.

T → ε | 1T0 | 0T1 | TT

How do we add the “exactly one 2” constraint?

Problem 7 – CFGs

c) All strings over 0, 1, 2 with the same number of 1s and 0s and exactly one 2.

T → ε | 1T0 | 0T1 | TT

How do we add the “exactly one 2” constraint?

S → T2T

The T on the left is a string with the same number of 1s and 0s. The T on the right is a string with the same number of 1s and 0s. So the entire string has the same number of 1s and 0s, and a 2 somewhere.

Problem 7 – CFGs

c) All strings over 0, 1, 2 with the same number of 1s and 0s and exactly one 2.

S → T2T
T → ε | 1T0 | 0T1 | TT

This doesn't match strings like 021 or 120. Let's expand S to include 1s and 0s.

Problem 7 – CFGs

c) All strings over 0, 1, 2 with the same number of 1s and 0s and exactly one 2.

S → T2T | 0S1 | 1S0
T → ε | 1T0 | 0T1 | TT

This seems better…but what about a string like 01120? We can get 01 and 120, but we can't concatenate them together. Let's add that logic to S.

Problem 7 – CFGs

c) All strings over 0, 1, 2 with the same number of 1s and 0s and exactly one 2.

S → T2T | 0S1 | 1S0 | ST | TS
T → ε | 1T0 | 0T1 | TT

This is our final CFG!

That's All, Folks!

Thanks for coming to section this week!
Any questions?