FR-0: Sets & Functions

- Sets
  - Membership:
    - \( a \in \{a, b, c\} \)
    - \( a \in \{b, c\} \)
    - \( a \in \{b, \{a, b, c\}, d\} \)
    - \( \{a, b, c\} \in \{b, \{a, b, c\}, d\} \)

FR-1: Sets & Functions

- Sets
  - Membership:
    - \( a \in \{a, b, c\} \)
    - \( a \notin \{b, c\} \)
    - \( a \notin \{b, \{a, b, c\}, d\} \)
    - \( \{a, b, c\} \in \{b, \{a, b, c\}, d\} \)

FR-2: Sets & Functions

- Sets
  - Subset:
    - \( \{a\} \subseteq \{a, b, c\} \)
    - \( \{a\} \subseteq \{b, c, \{a\}\} \)
    - \( \{a, b\} \subseteq \{a, b, c, d\} \)
    - \( \{a, b\} \subseteq \{a, b\} \)
    - \( \{\} \subseteq \{a, b, c, d\} \)

FR-3: Sets & Functions

- Sets
  - Subset:
    - \( \{a\} \subseteq \{a, b, c\} \)
    - \( \{a\} \nsubseteq \{b, c, \{a\}\} \)
    - \( \{a, b\} \subseteq \{a, b, c, d\} \)
    - \( \{a, b\} \subseteq \{a, b\} \)
    - \( \{\} \subseteq \{a, b, c, d\} \)

FR-4: Sets & Functions

- Sets
  - Cross Product:
    - \( A \times B = \{(a, b) : a \in A, b \in B\} \)
    - \( \{a, b\} \times \{a, b\} = \)
    - \( \{a, b\} \times \{\{a, b\}\} = \)
FR-5: Sets & Functions

- Sets
  - Cross Product:
    - \( A \times B = \{(a, b) : a \in A, b \in B\}\)
    - \(\{a, b\} \times \{a, b\} = \{(a, a), (a, b), (b, a), (b, b)\}\)
    - \(\{a, b\} \times \{(a, b)\} = \{(a, \{a, b\}), (b, \{a, b\})\}\)

FR-6: Sets & Functions

- Sets
  - Power Set:
    - \(2^A = \{S : S \subseteq A\}\)
    - \(2^{\{a,b\}} = \)
    - \(2^{\{a\}} = \)
    - \(2^{2^{\{a\}}} = \)

FR-7: Sets & Functions

- Sets
  - Power Set:
    - \(2^A = \{S : S \subseteq A\}\)
    - \(2^{\{a,b\}} = \{\{\}, \{a\}, \{b\}, \{a, b\}\}\)
    - \(2^{\{a\}} = \{\{\}, \{a\}\}\)
    - \(2^{2^{\{a\}}} = \{\{\}, \{\}\}, \{\{a\}\}, \{\}, \{a\}\}\)

FR-8: Sets – Partition

\(\Pi\) is a partition of \(S\) if:

- \(\Pi \subset 2^S\)
- \(\{\} \not\in \Pi\)
- \(\forall (X, Y \in \Pi), X \neq Y \implies X \cap Y = \{\}\)
- \(\bigcup \Pi = S\)

\(\{\{a, c\}, \{b, d, e\}, \{f\}\}\) is a partition of \{a,b,c,d,e,f\}
\(\{\{a, b, c, d, e, f\}\}\) is a partition of \{a,b,c,d,e,f\}
\(\{\{a, b, c\}, \{d, e, f\}\}\) is a partition of \{a,b,c,d,e,f\}

FR-9: Sets – Partition

In other words, a partition of a set \(S\) is just a division of the elements of \(S\) into 1 or more groups.

- All the partitions of the set \{a, b, c\}?

FR-10: Sets – Partition

In other words, a partition of a set \(S\) is just a division of the elements of \(S\) into 1 or more groups.

- All the partitions of the set \{a, b, c\}?
• \{\{a, b, c\}\}, \{\{a, b\}, \{c\}\}, \{\{a, c\}, \{b\}\}, \{\{a\}, \{b, c\}\}, \{\{a\}, \{b\}, \{c\}\}

FR-11: Sets & Functions

• Relation
  • A relation \( R \) is a set of ordered pairs
  • That’s all that a relation is
  • Relation Graphs

FR-12: Sets & Functions

• Properties of Relations
  • Reflexive
  • Symmetric
  • Transitive
  • Antisymmetric

• Equivalence Relation: Reflexive, Symmetric, Transitive
• Partial Order: Reflexive, Antisymmetric, Transitive
• Total Order: Partial order, for each \( a, a' \in A \), either \((a, a') \in R\) or \((a', a) \in R\)

FR-13: Sets & Functions

• What does a graph of an Equivalence relation look like?
• What does a graph of a Total Order look like?
• What does a graph of a Partial Order look like?

FR-14: Closure

• A set \( A \subseteq B \) is closed under a relation \( R \subseteq ((B \times B) \times B) \) if:
  • \( a_1, a_2 \in A \land ((a_1, a_2), c) \in R \implies c \in A \)
  • That is, if \( a_1 \) and \( a_2 \) are both in \( A \), and \(((a_1, a_2), c)\) is in the relation, then \( c \) is also in \( A \)

• \( \mathbb{N} \) is closed under addition
• \( \mathbb{N} \) is not closed under subtraction or division

FR-15: Closure

• Relations are also sets (of ordered pairs)
• We can talk about a relation \( R \) being closed over another relation \( R' \)
  • Each element of \( R' \) is an ordered triple of ordered pairs!

FR-16: Closure

• Relations are also sets (of ordered pairs)
• We can talk about a relation $R$ being closed over another relation $R'$
  • Each element of $R'$ is an ordered triple of ordered pairs!

• Example:
  • $R \subseteq A \times A$
  • $R' = \{(((a, b), (b, c)), (a, c)) : a, b, c \in A\}$
  • If $R$ is closed under $R'$, then . . .

FR-17: Closure
• Relations are also sets (of ordered pairs)
• We can talk about a relation $R$ being closed over another relation $R'$
  • Each element of $R'$ is an ordered triple of ordered pairs!

• Example:
  • $R \subseteq A \times A$
  • $R' = \{(((a, b), (b, c)), (a, c)) : a, b, c \in A\}$
  • If $R$ is closed under $R'$, then $R$ is transitive!

FR-18: Closure
• Reflexive closure of a relation $R \subseteq A \times A$ is the smallest possible superset of $R$ which is reflexive
  • Add self-loop to every node in relation
  • Add $(a,a)$ to $R$ for every $a \in A$

• Transitive Closure of a relation $R \subseteq A \times A$ is the smallest possible superset of $R$ which is transitive
  • Add direct link for every path of length 2.
  • $\forall (a, b, c \in A)$ if $(a, b) \in R \land (b, c) \in R$ add $(a, c)$ to $R$.

(examples on board) FR-19: Sets & Functions
• Functions
  • Relation $R$ over $A \times B$
  • For each $a \in A$:
    • Exactly one element $(x, y) \in R$ with $x = a$

FR-20: Sets & Functions
• For a function $f$ over $(A \times A)$, what does the graph look like?
• For a function $f$ over $(A \times B)$, what does the graph look like?

FR-21: Sets & Functions
• Functions
  • one-to-one: $f(a) \neq f(a')$ when $a \neq a'$ (nothing is mapped to twice)
onto: for each \( b \in B \), \( \exists a \) such that \( f(a) = b \) (everything is mapped to)

bijection: Both one-to-one and onto

FR-22: **Sets & Functions**

- For a function \( f \) over \( (A \times B) \)
  - What does the graph look like for a one-to-one function?
  - What does the graph look like for an onto function?
  - What does the graph look like for a bijection?

FR-23: **Sets & Functions**

- Infinite sets
  - Countable, Countably infinite
    - Bijection with the Natural Numbers
  - Uncountable, uncountable infinite
    - Infinite
    - No bijection with the Natural Numbers

FR-24: **Infinite Sets**

- We can show that a set is countable infinite by giving a bijection between that set and the natural numbers
- Same thing as as imposing an ordering on an infinite set

FR-25: **Countable Sets**

- A set is **countable infinite** (or just **countable**) if it is equinumerous with \( N \).
  - Even elements of \( N \)?

FR-26: **Countable Sets**

- A set is **countable infinite** (or just **countable**) if it is equinumerous with \( N \).
  - Even elements of \( N \)?
  - \( f(x) = 2x \)

FR-27: **Countable Sets**

- A set is **countable infinite** (or just **countable**) if it is equinumerous with \( N \).
  - Integers \( (Z) \)?

FR-28: **Countable Sets**

- A set is **countable infinite** (or just **countable**) if it is equinumerous with \( N \).
  - Integers \( (Z) \)?
  - \( f(x) = \left\lfloor \dfrac{x}{2} \right\rfloor \ast (-1)^x \)
FR-29: **Countable Sets**

- A set is **countable infinite** (or just **countable**) if it is equinumerous with $\mathbb{N}$.
  - Union of 3 (disjoint) countable sets $A$, $B$, $C$?

FR-30: **Countable Sets**

- A set is **countable infinite** (or just **countable**) if it is equinumerous with $\mathbb{N}$.
  - Union of 3 (disjoint) countable sets $A$, $B$, $C$?

$$f(x) = \begin{cases} 
  a_{x/3} & \text{if } x \mod 3 = 0 \\
  b_{x/3 - 1} & \text{if } x \mod 3 = 1 \\
  c_{x/3 - 2} & \text{if } x \mod 3 = 2 
\end{cases}$$

FR-31: **Countable Sets**

- A set is **countable infinite** (or just **countable**) if it is equinumerous with $\mathbb{N}$.
  - $\mathbb{N} \times \mathbb{N}$?

$$\begin{array}{} 
  (0,0) & (0,1) & (0,2) & (0,3) & (0,4) & \ldots \\
  (1,0) & (1,1) & (1,2) & (1,3) & (1,4) & \ldots \\
  (2,0) & (2,1) & (2,2) & (2,3) & (2,4) & \ldots \\
  (3,0) & (3,1) & (3,2) & (3,3) & (3,4) & \ldots \\
  (4,0) & (4,1) & (4,2) & (4,3) & (4,4) & \ldots \\
  \vdots & \vdots & \vdots & \vdots & \vdots & \ddots 
\end{array}$$

FR-32: **Countable Sets**

- A set is **countable infinite** (or just **countable**) if it is equinumerous with $\mathbb{N}$.
  - $\mathbb{N} \times \mathbb{N}$?
\[(0,0)\ (0,1)\ (0,2)\ (0,3)\ (0,4)\ ...
(1,0)\ (1,1)\ (1,2)\ (1,3)\ (1,4)\ ...
(2,0)\ (2,1)\ (2,2)\ (2,3)\ (2,4)\ ...
(3,0)\ (3,1)\ (3,2)\ (3,3)\ (3,4)\ ...
(4,0)\ (4,1)\ (4,2)\ (4,3)\ (4,4)\ ...
\]
\[
\cdots
\]
\[
\begin{align*}
\bullet \quad f((x, y)) &= \frac{(x+y)^{(x+y+1)}}{2} + x
\end{align*}
\]

FR-33: **Countable Sets**

- A set is **countable infinite** (or just **countable**) if it is equinumerous with \(N\).
- Real numbers between 0 and 1 (exclusive)?

FR-34: **Uncountable \(R\)**

- Proof by contradiction
  - Assume that \(R\) between 0 and 1 (exclusive) is countable
    - (that is, assume that there is some bijection from \(N\) to \(R\) between 0 and 1)
  - Show that this leads to a contradiction
    - Find some element of \(R\) between 0 and 1 that is not mapped to by any element in \(N\)

FR-35: **Uncountable \(R\)**

- Assume that there is some bijection from \(N\) to \(R\) between 0 and 1

<table>
<thead>
<tr>
<th></th>
<th>0.3412315569...</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0.0123506541...</td>
</tr>
<tr>
<td>1</td>
<td>0.1143216751...</td>
</tr>
<tr>
<td>2</td>
<td>0.2839143215...</td>
</tr>
<tr>
<td>3</td>
<td>0.2311459412...</td>
</tr>
<tr>
<td>4</td>
<td>0.8381441234...</td>
</tr>
<tr>
<td>5</td>
<td>0.7415296413...</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

FR-36: **Uncountable \(R\)**

- Assume that there is some bijection from \(N\) to \(R\) between 0 and 1

<table>
<thead>
<tr>
<th></th>
<th>0.3412315569...</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0.0123506541...</td>
</tr>
<tr>
<td>1</td>
<td>0.1143216751...</td>
</tr>
<tr>
<td>2</td>
<td>0.2839143215...</td>
</tr>
<tr>
<td>3</td>
<td>0.2311459412...</td>
</tr>
<tr>
<td>4</td>
<td>0.8381441234...</td>
</tr>
<tr>
<td>5</td>
<td>0.7415296413...</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Consider: 0.425055...

FR-37: **Formal Languages**
• **Alphabet** $\Sigma$: Set of symbols
  - $\{0, 1\}, \{a, b, c\}$, etc

• **String** $w$: Sequence of symbols
  - *cat, dog, firehouse* etc

• **Language** $L$: Set of strings
  - $\{\text{cat, dog, firehouse}\}, \{a, aa, aaa, \ldots\}$, etc

• **Language class**: Set of Languages
  - Regular languages, $P$, $NP$, etc.

**FR-38: Formal Languages**

- Language Hierarchy.

**FR-39: Regular Expressions**

- Regular expressions are a way to describe formal languages
- Regular expressions are defined recursively
  - Base case – simple regular expressions
  - Recursive case – how to build more complex regular expressions from simple regular expressions

**FR-40: Regular Expressions**

- $\epsilon$ is a regular expression, representing $\{\epsilon\}$
- $\emptyset$ is a regular expression, representing $\{\}$
- $\forall a \in \Sigma$, $a$ is a regular expression representing $\{a\}$
  - if $r_1$ and $r_2$ are regular expressions, then $(r_1r_2)$ is a regular expression
    - $L[(r_1r_2)] = L[r_1] \circ L[r_2]$
• if \( r_1 \) and \( r_2 \) are regular expressions, then \((r_1 + r_2)\) is a regular expression
  • \( L[(r_1 + r_2)] = L[r_1] \cup L[r_2] \)

• if \( r \) is regular expressions, then \((r^*)\) is a regular expression
  • \( L[(r^*)] = (L[r])^* \)

FR-41: re. Precedence
From highest to Lowest:

Kleene Closure *
Concatenation
Alternation +

\[ ab^*c+e = (a(b^*)c) + e \]

(We will still need parentheses for some regular expressions: \((a+b)(a+b)\))

FR-42: Regular Expressions

• Intuitive Reading of Regular Expressions
  • Concatenation == “is followed by”
  • + == “or”
  • * == “zero or more occurance”
• \((a+b)(a+b)(a+b)\)
• \((a+b)^*\)
• \(aab(aa)^*\)

FR-43: Regular Languages

• A language \( L \) is regular if there exists a regular expression which generates it

• Give a regular expression for:
  • All strings over \( \{a, b\} \) that have an odd # of a’s

FR-44: Regular Languages

• A language \( L \) is regular if there exists a regular expression which generates it

• Give a regular expression for:
  • All strings over \( \{a, b\} \) that have an odd # of a’s
    \[ b^*a(b^*ab^*)b^* \]
  • All strings over \( \{a, b\} \) that contain exactly two occurrences of \( bb \) (\( bbb \) counts as 2 occurrences!)

FR-45: Regular Languages

• A language \( L \) is regular if there exists a regular expression which generates it
• Give a regular expression for:
  • All strings over \{a, b\} that have an odd # of a’s
    \( b^*a(b^*ab^*a)^*b^* \)
  • All strings over \{a, b\} that contain exactly two occurrences of bb (\( bbb \) counts as 2 occurrences!)
    \( a^*(baa^*)^*bb(aa^*b)^*a^* + a^*(baa^*)^*bb(aa^*b)^*a^* \)

FR-46: **Regular Languages**

• All strings over \{0, 1\} that begin (or end) with 11
• All strings over \{0, 1\} that begin (or end) with 11, but not both

FR-47: **Regular Languages**

• All strings over \{0, 1\} that begin (or end) with 11
  • \( 11(0+1)^*11 + 11 \)
• All strings over \{0, 1\} that begin (or end) with 11, but not both
  • \( 11(0+1)^*0 + 11(0+1)^*01 + 0(0+1)^*11 + 10(0+1)^*11 \)

FR-48: **Regular Languages**

• Shortest string not described by following regular expressions?
  • \( a^*b*a*b^* \)
  • \( a^*(ab)^*(ba)^*b*a^* \)
  • \( a^*b^*(ab)^*b^*a^* \)

FR-49: **Regular Languages**

• Shortest string not described by following regular expressions?
  • \( a^*b*a*b^* \)
    • \( baba \)
  • \( a^*(ab)^*(ba)^*b*a^* \)
    • \( baab \)
  • \( a^*b^*(ab)^*b^*a^* \)
    • \( baab \)

FR-50: **Regular Languages**

• English descriptions of following regular expressions:
  • \( (aa+aaa)^* \)
  • \( b(a+b)^*b + a(a+b)^*a + a + b \)
  • \( a^*(baa^*)^*bb(aa^*b)^*a^* \)

FR-51: **Regular Languages**

• A language \( L \) is regular if there exists a DFA which accepts it
• DFA for all strings with exactly 2 occurrences of $bb$

FR-52: DFA Definition

• A DFA is a 5-tuple $M = (K, \Sigma, \delta, s, F)$
  • $K$ Set of states
  • $\Sigma$ Alphabet
  • $\delta : (K \times \Sigma) \rightarrow K$ is a Transition function
  • $s \in K$ Initial state
  • $F \subseteq K$ Final states

FR-53: Regular Languages

• A language $L$ is regular if there exists a DFA which accepts it
  • DFA for all strings with exactly 2 occurrences of $bb$

FR-54: Regular Languages

• A language $L$ is regular if there exists a DFA which accepts it
  • DFA for all strings over $\{0,1\}$ that start and end with 111

FR-55: Regular Languages

• A language $L$ is regular if there exists a DFA which accepts it
  • DFA for all strings over $\{0,1\}$ that start and end with 111
• A language $L$ is regular if there exists a DFA which accepts it
  • DFA for all strings over $\{0,1\}$ that start with 110, end with 011

**FR-57: Regular Languages**

• A language $L$ is regular if there exists a DFA which accepts it
  • DFA for all strings over $\{0,1\}$ that start with 110, end with 011

![DFA diagram](image)

**FR-58: Regular Languages**

• Give a DFA for all strings over $\{0,1\}$ that begin or end with 11
• Give a DFA for all strings over $\{0,1\}$ that begin or end with 11 (but not both)

**FR-59: Regular Languages**

• Give a DFA for all strings over $\{0,1\}$ that contain 101010
• Give a DFA for all strings over $\{0,1\}$ that contain 101 or 010
• Give a DFA for all strings over $\{0,1\}$ that contain 010 and 101

**FR-60: DFA Configuration & $\Gamma_M$**

• Way to describe the computation of a DFA
  • **Configuration**: What state the DFA is currently in, and what string is left to process
    • $\in K \times \Sigma^*$
    • $(q_2, abba)$ Machine is in state $q_2$, has $abba$ left to process
    • $(q_8, bbab)$ Machine is in state $q_8$, has $bbab$ left to process
    • $(q_4, \epsilon)$ Machine is in state $q_4$ at the end of the computation (accept iff $q_4 \in F$)

**FR-61: DFA Configuration & $\Gamma_M$**

• Way to describe the computation of a DFA
  • **Configuration**: What state the DFA is currently in, and what string is left to process
    • $\in K \times \Sigma^*$
Binary relation $\vdash_M$: What machine $M$ yields in one step

- $\vdash_M \subseteq (K \times \Sigma^*) \times (K \times \Sigma^*)$
- $\vdash_M = \{ ((q_1, aw), (q_2, w)) : q_1, q_2 \in K, w \in \Sigma^*_M, a \in \Sigma_M, ((q_1, a), q_2) \in \delta_M \}$

FR-62: DFA Configuration & $\vdash_M$

Given the following machine $M$:

![DFA Diagram]

- $((q_0, abba), (q_2, bba)) \in \vdash_M$
- can also be written $(q_0, abba) \vdash_M (q_2, bba)$

FR-63: DFA Configuration & $\vdash_M$

![DFA Diagram]

FR-64: DFA Configuration & $\vdash_M$

![DFA Diagram]

FR-65: DFA Configuration & $\vdash^*_M$

![DFA Diagram]

- $\vdash^*_M$ is the reflexive, transitive closure of $\vdash_M$
- Smallest superset of $\vdash_M$ that is both reflexive and transitive
• “yields in 0 or more steps”

• Machine $M$ accepts string $w$ if:

$$(s_M, w) \vdash_M^* (f, \epsilon) \text{ for some } f \in F_M$$

FR-66: DFA & Languages

• Language accepted by a machine $M = L[M]$

  $\{ w : (s_M, w) \vdash_M^* (f, \epsilon) \text{ for some } f \in F_M \}$

• DFA Languages, $L_{DFA}$

  • Set of all languages that can be defined by a DFA

  • $L_{DFA} = \{ L : \exists M, L[M] = L \}$

  • To think about: How does $L_{DFA} = L_{REG}$

FR-67: NFA Definition

• Difference between a DFA and an NFA

  • DFA has exactly only transition for each state/symbol pair

    • Transition function: $\delta : (K \times \Sigma) \mapsto K$

  • NFA has 0, 1 or more transitions for each state/symbol pair

    • Transition relation: $\Delta \subseteq (K \times \Sigma) \times K$

FR-68: NFA Definition

• A NFA is a 5-tuple $M = (K, \Sigma, \Delta, s, F)$

  • $K$ Set of states

  • $\Sigma$ Alphabet

  • $\Delta : (K \times \Sigma) \times K$ is a Transition relation

  • $s \in K$ Initial state

  • $F \subseteq K$ Final states

FR-69: Fun with NFA

Create an NFA for:

• All strings over $\{a, b\}$ that start with a and end with b

(Also create a DFA, and regular expression) FR-70: Fun with NFA

Create an NFA for:

• All strings over $\{a, b\}$ that contain 010 or 101

FR-71: Regular Languages

• A language $L$ is regular if there exists an NFA which accepts it

  • NFA for all strings over $\{a, b\}$ that contain $abba$
FR-72: **Regular Languages**
- A language $L$ is regular if there exists an NFA which accepts it
  - NFA for all strings over $\{a, b\}$ that contain $abba$

![NFA Diagram](image1)

FR-73: **Regular Languages**
- A language $L$ is regular if there exists an NFA which accepts it
  - NFA for all strings over $\{a, b\}$ that do not contain $abba$

FR-74: **Regular Languages**
- A language $L$ is regular if there exists an NFA which accepts it
  - NFA for all strings over $\{a, b\}$ that do not contain $abba$

![NFA Diagram](image2)

FR-75: **Regular Expression & NFA**
- Give a regular expression for all strings over $\{a, b\}$ that have an even number of a’s, and a number of b’s divisible by 3

FR-76: **Pumping Lemma**
- Not all languages are Regular
- $L = \text{all strings over } \{a, b, c\} \text{ that contain more } a \text{’s than } b \text{’s and } c \text{’s combined}$

FR-77: **Pumping Lemma**
- To show that a language $L$ is not regular, using the pumping lemma:
  - Let $n$ be the constant of the pumping lemma
  - Create a string $w \in L$, such that $|w| > n$
  - For each way of breaking $w = xyz$ such that $|xy| \leq n$, $|y| > 0$:
    - Show that there is some $i$ such that $xy^iz \notin L$
• By the pumping lemma, \( L \) is not regular

FR-78: **Pumping Lemma**

• Prove \( L = \) all strings over \( \{a, b, c\} \) that contain more \( a \)'s than \( b \)'s and \( c \)'s combined is not regular

• Let \( n \) be the constant of the pumping lemma

• Consider \( w = b^n a^{n+1} \in L \)

• If we break \( w = xyz \) such that \( |xy| \leq n \), then \( y \) must be all \( b \)'s. Let \( |y| = k \)

• Consider \( w' = xy^2x = b^{n+k}a^n \). \( w' \notin L \) for any \( k > 0 \), thus by the pumping lemma, \( L \) is not regular

FR-79: **Context-Free Languages**

• A language is context-free if a CFG generates it

  • All strings over \( \{a, b, c\} \) with same # of \( a \)'s as \( b \)'s

FR-80: **Context-Free Languages**

• A language is context-free if a CFG generates it

  • All strings over \( \{a, b, c\} \) with same # of \( a \)'s as \( b \)'s

\[
S \rightarrow aSb \\
S \rightarrow bSa \\
S \rightarrow SS \\
S \rightarrow cS \\
S \rightarrow Sc \\
S \rightarrow \epsilon
\]

FR-81: **Context-Free Languages**

• A language is context-free if a CFG generates it

  • All strings over \( \{a, b, c\} \) with more \( a \)'s than \( b \)'s

FR-82: **Context-Free Languages**

• A language is context-free if a CFG generates it

  • All strings over \( \{a, b, c\} \) with more \( a \)'s than \( b \)'s

\[
S \rightarrow cS|Sc \\
S \rightarrow aSb|bSa \\
S \rightarrow aA|Aa \\
S \rightarrow SA \\
A \rightarrow aAb \\
A \rightarrow bAa \\
A \rightarrow AA \\
A \rightarrow cA|Ac \\
A \rightarrow aA|Aa \\
A \rightarrow \epsilon
\]

FR-83: **Context-Free Languages**
A language is context-free if a PDA accepts it
- All strings over \{a, b, c\} that contain more a’s than b’s and c’s combined

**FR-84: Context-Free Languages**
- A language is context-free if a PDA accepts it
- All strings over \{a, b, c\} that contain more a’s than b’s and c’s combined

\[
\begin{align*}
(a, \epsilon, \epsilon) & \rightarrow (b, \epsilon, X) \\
(a, \epsilon, A) & \rightarrow (b, A, \epsilon) \\
(a, X, \epsilon) & \rightarrow (c, \epsilon, X) \\
(a, X, \epsilon) & \rightarrow (c, A, \epsilon)
\end{align*}
\]

**FR-85: Recursive Languages**
- A language \(L\) is recursive if an always-halting Turing Machine accepts it
  - In other words, a Turing Machine decides \(L\)
  - Create a Turing Machine for all strings over \{a, b, c\} with an equal number of a’s, b’s and c’s.

**FR-86: Recursive Languages**
- Computing functions with TMs
  - Give a TM that computes negation, for a 2’s complement binary number
  - (flip bits, add one, discard overflow)

**FR-87: Recursive Languages**
- Computing functions with TMs
  - Give a TM that computes negation, for a 2’s complement binary number

**FR-88: Recursive Languages**
- Computing functions with TMs
  - Give a TM that computes negation, for a 2’s complement binary number
  - (flip bits, add one, discard overflow)
FR-89: *r.e. Languages*

- A language $L$ is recursively enumerable if there is some Turing Machine $M$ that halts and accepts everything in $L$, and runs forever on everything not in $L$.
- Give a TM that semi-decides $L = a^n b^n$
  - Note that this language is also context-free – context-free languages are a subset of the r.e. languages.

FR-90: *r.e. Languages*

- Enumeration Machines
  - Create a Turing Machine that enumerate the language: $L = \text{all strings of the form } wcw, w \in (a + b)^*$

FR-91: *Counter Machines*

- Finite automata with a counter (never negative)
- Add one, subtract 1, check for zero
- Create a 1-counter machine for all strings over \{a,b\} that contain the same number of a’s as b’s

FR-92: *Unrestricted Grammars*

$G = (V, \Sigma, R, S)$

- $V$ = Set of symbols, both terminals & non-terminals
- $\Sigma \subseteq V$ set of terminals (alphabet for the language being described)
- $R \subseteq (V^* (V - \Sigma) V^* \times V^*)$ Set of rules
- $S \in (V - \Sigma)$ Start symbol

FR-93: *Unrestricted Grammars*

- $R \subseteq (V^* (V - \Sigma) V^* \times V^*)$ Set of rules
- In an Unrestricted Grammar, the left-hand side of a rule contains a string of terminals and non-terminals (at least one of which must be a non-terminal)
- Rules are applied just like CFGs:
• Find a substring that matches the LHS of some rule
• Replace with the RHS of the rule

FR-94: **Unrestricted Grammars**

• To generate a string with an Unrestricted Grammar:
  • Start with the initial symbol
  • While the string contains at least one non-terminal:
    • Find a substring that matches the LHS of some rule
    • Replace that substring with the RHS of the rule

FR-95: **Unrestricted Grammars**

• Example: Grammar for \( L = \{ a^n b^n c^n : n > 0 \} \)
  • First, generate \((ABC)^*\)
  • Next, non-deterministically rearrange string
  • Finally, convert to terminals \((A \rightarrow a, B \rightarrow b, \text{ etc.}),\) ensuring that string was reordered to form \(a^*b^*c^*\)

FR-96: **Unrestricted Grammars**

\[
S \rightarrow ABCS \\
S \rightarrow TC \\
CA \rightarrow AC \\
BA \rightarrow AB \\
CB \rightarrow BC \\
CTC \rightarrow TCc \\
TC \rightarrow TB \\
BTC \rightarrow TBb \\
TB \rightarrow TA \\
ATA \rightarrow TAAa \\
T \rightarrow \epsilon
\]

• Example: Grammar for \( L = \{ a^n b^n c^n : n > 0 \} \)
  
FR-97: **Unrestricted Grammars**

\( S \Rightarrow ABCS \Rightarrow AAT_A bcc \)
\( \Rightarrow ABCABCS \Rightarrow AT_A bcc \)
\( \Rightarrow ABACBCS \Rightarrow T_A aabbcc \)
\( \Rightarrow AABCBCS \Rightarrow aabbcc \)
\( \Rightarrow AABCCCS \)
\( \Rightarrow AABBCCTC \)
\( \Rightarrow AABCTCcc \)
\( \Rightarrow AAABTCccc \)
\( \Rightarrow AAABTBbacc \)
\( \Rightarrow AAABTBbacc \Rightarrow AAT_B bcc \)

FR-98: **Unrestricted Grammars**
\begin{align*}
S & \Rightarrow ABCS \Rightarrow AAABBBBCCCT_C  \\
& \Rightarrow ABCABCS \Rightarrow AAABBBBCCCT_Cc  \\
& \Rightarrow ABCABCBACS \Rightarrow AAABBBBCCT_{Ccc}  \\
& \Rightarrow ABACBCABCS \Rightarrow ABB BBCCT_{Cccc}  \\
& \Rightarrow AABBCABCS \Rightarrow AABBBBCCT_{Bccc}  \\
& \Rightarrow AABCBACBCS \Rightarrow AAABBBT_{Bcc}c  \\
& \Rightarrow AABCA B C B C S \Rightarrow AAABBT_{Bbccc}  \\
& \Rightarrow AABCA B C B C S \Rightarrow AAABTT_{Bbccc}  \\
& \Rightarrow AAACBCBCBS \Rightarrow AAABBCBBS  \\
& \Rightarrow AAABBCBCCS \Rightarrow AAABBBCCCS  \\
& \Rightarrow AAABBBCCSS \Rightarrow T_{Aaaabbbeccc} \Rightarrow aaabbbecce.
\end{align*}