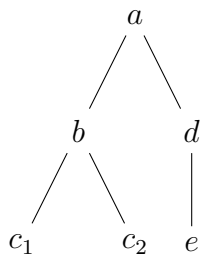
Homework 2 . Set Theory. Languages.

Scott Weeden - Distance Learner

1. (10) (Tuples). Represent the following tree using a tuple. The goal of this question is to show that tuple is a construct you can use to represent very complex objects.



Tuples are mathematically precise constructs that serve as a **"handy way"** to represent objects with numerous, often recursively defined components. By using ordered nesting, the tuple captures the hierarchical structure of the tree.

The representation starts with the root node (a) followed by the tuples representing its children subtrees:

(a) The root node is a.

(b) The first child sub-tree is rooted at b, with children c_1 and c_2: (b, c_1, c_2).

(c) The second child sub-tree is rooted at d, with child e: (d, e).

The formal tuple representation for the entire tree is:

$$(\mathbf{a}, (\mathbf{b}, \mathbf{c_1}, \mathbf{c_2}), (\mathbf{d}, \mathbf{e}))$$

The components of complex systems, like the elements of a Finite Automaton ($(\Sigma, K, s, F, \delta)$), are themselves represented efficiently using tuples.

2. (5) (Write precise definition). Let $A = \{1, 2, 3\}, B = \{1, 2\}, C = \{1, 2, 3\}, D = \{1, 2, 3, 4\}$. Is $A = B$? $A = C$? $A = D$? Write a mathematical definition of = between two sets. (Study how subset definition is written - around P12 of L3. You may directly use concepts such as *subset* defined in the slides.)

   Set equality requires two sets to contain **exactly the same elements**.

   (a) Is **A = B**? **No.** A contains the element 3, which is not in B.

   (b) Is **A = C**? **Yes.** A and C contain the same elements $\{1, 2, 3\}$.

   (c) Is **A = D**? **No.** D contains the element 4, which is not in A.

   Precise Mathematical Definition of Set Equality (=):

   The definition of a concept must use only standard concepts (like logical symbols or membership) or concepts that have been previously defined. Since the query allows using the concept of **subset** ($\subseteq$), set equality is defined based on mutual subset containment.

   Definition(equal): For every set A and B, $\quad A = B \quad$ if and only if $\quad A \subseteq B$ and $B \subseteq A$

   This definition states that two sets A and B are equal if A is a subset of B ($\mathbf{A} \subseteq \mathbf{B}$) and simultaneously B is a subset of A ($\mathbf{B} \subseteq \mathbf{A}$).

   Discussion: In mathematical language, defining equality requires leveraging previously defined concepts, in this case, the subset relation ($\subseteq$). The definition relies on the definition of the subset concept itself, which uses the universal quantifier ($\forall$) and implication ($\rightarrow$): $A \subseteq B$ if and only if for every element x, if $x \in A$ then $x \in B$.

   Why Mutual Containment?: Simply checking cardinality ("same number of elements") is insufficient, especially when dealing with infinite sets. The formal definition of $A = B \iff (A \subseteq B \land B \subseteq A)$ is required because it uses established logical and set theoretical definitions to ensure absolute equivalence of elements.

3. (5) (Write precise definition). Let $A = \{1, 2, 3\}, B = \{1, 2\}, C = \{a\}$. What is $A \cup B$? $A \cup C$? Using *set builder notation*, write a precise definition of $\cup$ between two sets.

   The set union operation ($\cup$) is a function over sets. It yields a new set containing all elements present in either operand set.

   Calculations:

   (a) $A \cup B = \{1, 2, 3\} \cup \{1, 2\} = \{\mathbf{1}, \mathbf{2}, \mathbf{3}\}$

   (b) $A \cup C = \{1, 2, 3\} \cup \{a\} = \{\mathbf{1}, \mathbf{2}, \mathbf{3}, \mathbf{a}\}$

   Precise Definition of Set Union ($\cup$): A precise definition must rely only on standard concepts (like set, membership ($\in$), and logical connectives). We use set builder notation, which is read as "a set consisting of every x which makes the statement to be true".

   Given the definition of intersection $A \cap B = \{xx \in A \text{ and } x \in B\}$, we define union by replacing the conjunction ("and") with the disjunction ("or"):

   $$\textbf{Definition(union):} \quad A \cup B = \{\mathbf{x} \mid \mathbf{x} \in \mathbf{A} \text{ or } \mathbf{x} \in \mathbf{B}\}$$

   The resulting set consists of all elements x such that the condition ($x \in A$ or $x \in B$) is true.

4. (10) (Concepts and Statements). Consider the statement:

   for any two sets A and B, $A \cap B \subseteq B \cap A$.

   (a) Write each concept in the statement in the format of name with arguments. No need to write logical connectives or quantifiers.

   (b) Draw a tree for the statement $A \cap B \subseteq B \cap A$.

   (c) Apply the definition of $\subseteq$ to its use in statement $A \cap B \subseteq B \cap A$. (Recall how a definition of a concept is applied to its use around Page 25 of L2-foundation.)

   (d) If we use the working backward approach to proving the statement $A \cap B \subseteq B \cap A$, what (sub-statement) should we prove by working one step backward based on the definition application in (c)?

Part (a): Concepts and Arguments

A concept is defined mathematically as a function with a name and parameters. Concepts used in the statement, excluding the universal quantifiers ($\forall A, \forall B$) are:

(a) **Intersection** ($\cap$): This is a standard function over sets.

$$\text{Intersection}(A, B)$$

(b) **Intersection** ($\cap$):

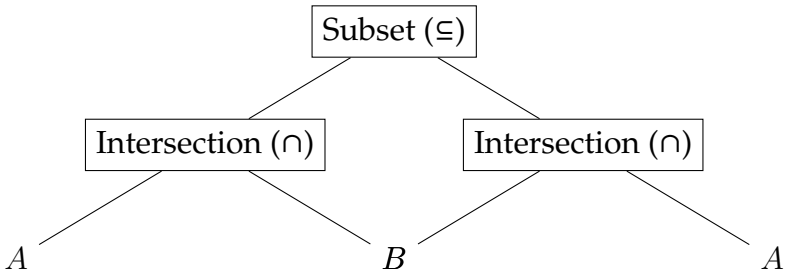
$$\text{Intersection}(B, A)$$

(c) **Subset** ($\subseteq$): This is a relation/concept between two sets. The arguments here are the resulting sets from the intersection operations.

$$\text{Subset}(A \cap B, B \cap A)$$

Part (b): Structure Tree for $A \cap B \subseteq B \cap A$

The structure tree (or parse tree) illustrates the nested use of concepts. The **main concept** (topmost node) governs the entire structure. In this case, the main concept is the subset relation.



Part (c): Application of the Definition of $\subseteq$

Applying the definition of a concept involves rewriting the general definition by replacing the parameters of the concept with the specific arguments used in the instance across the whole definition.

The precise definition of subset states:

Definition(subset): For every A' and B', $A' \subseteq B'$ if for every element x, if $x \in A'$ then $x \in B'$.

For the statement $A \cap B \subseteq B \cap A$:

- Arguments in use: $A_{arg} = A \cap B$, $B_{arg} = B \cap A$.
- Parameters in definition: A', B'.

Replacing the parameters with the arguments yields the instance of the definition:

$$\mathbf{A \cap B \subseteq B \cap A} \text{ if for every element } \mathbf{x}, \text{ if } \mathbf{x \in (A \cap B)} \text{ then } \mathbf{x \in (B \cap A)}$$

Part (d): Sub-statement to Prove (Working Backward)

The **working backward** methodology dictates that we follow the structure tree from top to bottom, applying logical rules at each step to determine the prerequisite statement that needs to be proven.

The result from (c) shows that proving $A \cap B \subseteq B \cap A$ (let this be statement P) is equivalent to proving the consequence of the definition (let this be statement Q). Since the definition statement is assumed to be true, proving P requires proving Q.

Working backward one step from the relation $A \cap B \subseteq B \cap A$ based on the application of the subset definition, we must prove the core quantified implication:

$$\text{For every element } \mathbf{x}, \text{ if } \mathbf{x \in (A \cap B)} \text{ then } \mathbf{x \in (B \cap A)}$$

(Or using symbols: $\forall \mathbf{x}, (\mathbf{x \in (A \cap B)} \implies \mathbf{x \in (B \cap A)})$). This decomposition transforms the problem from proving a set relation to proving a statement about arbitrary element membership.

5. (10) (Concepts and Statements). For the definition of *functions* (around P28 in L3), answer the following questions (refer to pages around 12 and 13 in the slides for the answer of some similar question):

   (a) Write each concept, in the format of name and parameters, that is defined in that definition.

   (b) Write each concept, in the format of name and arguments, that is used to define the concepts being defined in the definition.

   (c) Write logical symbols (connectives and quantifiers) used in the definition.

   (d) Write variables used in the definition.

   The concept of a function is defined as a specialized type of relation, requiring a comprehensive analysis of the concepts used and logical symbols required to establish its rigor.

   (a) **Concepts defined (Name and Parameters):** The primary concept being defined is the property that a given relation constitutes a function from one set to another. We present the concept name followed by its parameters:

   $$\textbf{function}(f, A, B)$$

   where f is the relation being defined, A is the domain set, and B is the co-domain set. The use of bold font in the source material indicates the concept being defined.

   (b) **Concepts used to define the concepts being defined (Name and Arguments):** The definition relies on fundamental concepts established earlier in set theory:

      i. **Relation:** The definition begins by asserting that f must first be a relation on sets A and B.
      $$\text{relation}(f, A, B)$$

      ii. **Membership ($\in$):** Used to express that elements belong to sets or that a tuple belongs to the relation.
      $$\text{membership}(a, A)$$
      $$\text{membership}((a, b), f)$$

      iii. **Tuple:** The relation f consists of ordered pairs (a, b), requiring the concept of a tuple.
      $$\text{tuple}(a, b)$$

      iv. **Unique Value:** Although often implied by logical structure, the crucial requirement that the output b must be unique for every input a is a core concept used.
      $$\text{unique}(b)$$

(c) **Logical symbols (connectives and quantifiers) used in the definition:** The constraints defining a function are expressed using standard logical symbols:

i. **Universal Quantifier (∀):** Used in the phrase "for every $a \in A$".

ii. **Existential Quantifier (∃):** Used in the phrase "there is a unique value b of B".

iii. **Implication (→ or if/then):** Implicit in the structure "A relation f is a function... **if** [conditions apply]".

(d) **Variables used in the definition:** Variables are names used to represent general objects (or parameters/arguments) within the definition statement. The variables used here are:

$$\mathbf{f}, \mathbf{A}, \mathbf{B}, \mathbf{a}, \mathbf{b}$$

where f, A, and B represent the defining sets/relation, and a and b represent arbitrary elements from those sets subject to the function's constraints.

(10) (Proof and Reasoning). Prove: the set of natural numbers $\mathbb{N}$ is countable. (This question is for you to practice the following skills: recognizing concepts in a statement, working backwards, application of definition, proper labels and indentation of statements, and basic logical reasoning.)

The statement to prove is $\mathbb{N}$ is countable. We follow the methodology of working backwards and applying precise definitions .

1. Understand the Statement and Apply Definition (Working Backwards)

The primal concept is **countable** .

1. We note that the set of natural numbers $\mathbb{N} = \{1, 2, 3, \dots\}$ is infinite .
2. We apply the definition of a **countable set** (Definition): A set A is countable if it is finite or **countably infinite** .
3. We apply the definition of a **countably infinite set** (Definition): A set is countably infinite if there is a **bijective function** from $\mathbb{N}$ to A .

Working backwards one step (from point 3), the required goal is to demonstrate:

$$\text{Goal: There exists a bijective function } f : \mathbb{N} \to \mathbb{N}$$

2. Construct the Bijective Function

We construct the simplest possible function f that maps natural numbers to themselves :

$$\text{Construct: Let } f(n) = n \quad \text{for all } n \in \mathbb{N}$$

This function is a relation defined as $\{(n, n) \mid n \in \mathbb{N}\}$.

3. Prove the Function is Bijective

We must prove that $f(n) = n$ satisfies the definition of a **bijective function**, meaning it must be both **one-one** and **onto** .

1. **Proof that f is One-One (Injective):** By definition, a function f is one-one if for every n_1 and n_2, $f(n_1) = f(n_2)$ implies $n_1 = n_2$.
   (a) Assume $f(n_1) = f(n_2)$.
   (b) By the construction of f, this means $n_1 = n_2$.
   (c) Therefore, f is one-one.
2. **Proof that f is Onto (Surjective):** By definition, a function $f : \mathbb{N} \to \mathbb{N}$ is onto if for every $m \in \mathbb{N}$ (the co-domain), there exists an $n \in \mathbb{N}$ (the domain) such that $f(n) = m$.

(a) Let m be an arbitrary element in the co-domain $\mathbb{N}$.

(b) We must find $n \in \mathbb{N}$ such that $f(n) = m$.

(c) By setting $n = m$, we satisfy $f(n) = n = m$.

(d) Since m is a natural number, n is a natural number.

(e) Therefore, f is onto.

4. Conclusion

Since the function $f(n) = n$ is both one-one and onto, it is a bijective function from $\mathbb{N}$ to $\mathbb{N}$. By the definition of countably infinite, $\mathbb{N}$ is countably infinite, and therefore, $\mathbb{N}$ is countable .

(9) (Create example). Study the one-one function definition (around Page 29 of L3).

1. Construct a one-one function f from $A = \{1, 2, 3, 4\}$ to 2^A. You can present your function by listing the value of $f(i)$ for each $i \in A$.
2. How to read $D = \{x | x \in A \text{ and } x \notin f(x)\}$?
3. What is D given your function f in (a)?

Part (a): Constructing a One-One Function $f : A \to 2^A$

A function f is defined as **one-one (injective)** if for every x_1 and x_2, $f(x_1) = f(x_2)$ implies $x_1 = x_2$. Since 2^A (the power set of A) has $2^4 = 16$ elements and A has 4 elements, a one-one function exists. We must ensure each element of A maps to a distinct subset of A.

Let $A = \{1, 2, 3, 4\}$. We define f:

$$f(1) = \{1, 2\}$$

$$f(2) = \{1, 3\}$$

$$f(3) = \{3, 4\}$$

$$f(4) = \{1, 2, 3\}$$

(Each output subset is distinct, so f is one-one.)

Part (b): Reading the Set D

The notation $D = \{x \mid x \in A \text{ and } x \notin f(x)\}$ uses **set builder notation** .

The set D is read as:

D is the set consisting of every element x such that x is a member of A **and** x is not a member of the specific set $f(x)$ (the image of x under f).

This construct is the mathematical basis for the Diagonalization Principle used to prove that $2^{\mathbb{N}}$ is uncountable .

Part (c): Determining D

We use the constructed function f from part (a) to determine which elements $x \in A$ satisfy the condition $x \notin f(x)$:

$$D = \{x \in \{1, 2, 3, 4\} \mid x \notin f(x)\}$$

1. For $x = 1$: $f(1) = \{1, 2\}$. Since $1 \in f(1)$, the condition $1 \notin f(1)$ is **False**. $1 \notin D$.
2. For $x = 2$: $f(2) = \{1, 3\}$. Since $2 \notin f(2)$, the condition $2 \notin f(2)$ is **True**. $2 \in D$.

3. For $x = 3$: $f(3) = \{3, 4\}$. Since $3 \in f(3)$, the condition $3 \notin f(3)$ is **False**. $3 \notin D$.
4. For $x = 4$: $f(4) = \{1, 2, 3\}$. Since $4 \notin f(4)$, the condition $4 \notin f(4)$ is **True**. $4 \in D$.

Therefore, the resulting set D is:

$$\mathbf{D = \{2, 4\}}$$

(10) (Application of skills). Find a piece of text from a textbook or slides of a course you are taking now. It should be the hardest for your to understand (using your own way).

1. Literally copy the text here.
2. List each concept in the format of concept name and arguments. List all logical symbols (connectives or quantifiers).
3. Does the identification of concepts from this text help you to locate difficulties you might have otherwise and find directions for a further study to understand the text better (e.g., by tracing back to find definition/explanations of all concepts used in this text)? Why?

Part (a): Copied Text

I have chosen the formal definition of a Non-Deterministic Finite Automaton (NFA) because it introduces several complex, previously defined concepts (tuple, relation, sets) and uses mathematical syntax ($\in, \subseteq, \cup$) to define a crucial computational model.

Definition (NFA). An NFA is a tuple $(\Sigma, K, s, F, \Delta)$ where Σ is an alphabet; K and F are sets, $s \in K$, a

Part (b): Concepts and Logical Symbols

The analysis requires treating the text as a formal statement, identifying the concepts (functions) and the binding logical structure .

1. **Concept Being Defined (using bold font as a defined concept):**

$$\textbf{NFA}(\Sigma, K, s, F, \Delta)$$

2. **Concepts Used (Name and Arguments):**
   - tuple$(\Sigma, K, s, F, \Delta)$
   - alphabet(Σ)
   - set(K)
   - set(F)
   - membership$(\mathbf{s}, K)$: Used in the assertion $s \in K$.
   - subset$(\mathbf{F}, K)$: Used in the assertion $F \subseteq K$.
   - union$(\Sigma, \{\epsilon\})$: Used to form the extended input symbols for the transition relation .
   - relation$(\Delta, K \times (\Sigma \cup \{\epsilon\}), K)$: The most complex component, stating that Δ is a mathematical relation over the Cartesian product of the state set, the extended alphabet, and the state set .

3. **Logical Symbols:** The structure uses explicit set relations ($\in, \subseteq, \cup$) and implicitly uses **conjunction** ($\wedge$) because all the conditions must hold simultaneously for the tuple to constitute an NFA (e.g., K is a set $\wedge$ F is a set $\wedge$ $s \in K \wedge \ldots$) . No explicit quantifiers ($\forall, \exists$) are present in the provided definition block .

Part (c): Locating Difficulties and Directions for Study

Yes, the rigorous identification of concepts immediately helps to locate the source of difficulty, aligning with the principles of precise thinking and problem decomposition taught in the course .

- **Difficulty 1: Transition Relation (Δ):** The most difficult conceptual element is understanding $\boldsymbol{\Delta}$ as a **relation** rather than a **function** (like the δ in a DFA) . The breakdown highlights the need to trace back and compare the formal definitions of a function (which requires a unique output for every input) versus a relation (which is merely a subset of the Cartesian product). This pinpoints the source of non-determinism: Δ allows an input state/symbol pair to map to zero, one, or multiple next states .
- **Difficulty 2: The Epsilon (ϵ) Symbol:** The appearance of $\Sigma \cup \{\epsilon\}$ clarifies that ϵ is treated formally as an input symbol within the relation domain. This requires studying the precise semantics of ϵ-transitions, which is necessary for defining the NFA computation formally .
- **Direction for Study:** The methodology confirms that understanding this complex statement requires mastering the foundational definitions upon which it is built . I must specifically review: 1. The definition of **relation** in set theory . 2. The definition of **tuple** for representing complex, ordered data structures . 3. The recursive definition of **set union** ($\cup$) and **Cartesian product** ($\times$) as applied to the NFA domain definition .

This exercise validates the course's emphasis on breaking down complex statements using concepts and logical symbols to eliminate ambiguity and provide clear directions for mastering the material .

(10) (Write precise definitions). Given strings $x = abc$ over alphabet $\{a, b, c\}$, is a a prefix of x? ab a prefix of x? b? Write a precise definition for **prefix**.

A string v is a **prefix** of string w if w can be formed by concatenating v with some subsequent string x .

Analysis of Examples: Given the string $x = abc$ over the alphabet $\Sigma = \{a, b, c\}$:

1. Is a a prefix of x? **Yes.** x can be written as $a \circ bc$. We find an existing string (bc) such that $x = a \circ (bc)$.
2. Is ab a prefix of x? **Yes.** x can be written as $ab \circ c$. We find an existing string (c) such that $x = ab \circ c$.
3. Is b a prefix of x? **No.** x cannot be decomposed such that b is the first component followed by any string. b is a substring, but not a prefix.

Precise Definition of Prefix: The definition uses the concept of concatenation ($\circ$) and requires the existence of a subsequent string, thus employing the existential quantifier ($\exists$) .

Definition(prefix): For every string v and w, v is a prefix of w if and only if $\exists$ string x such tha

(Note: $v \circ x$ represents the concatenation of string v and string x . The string x may be the empty string ϵ.)

(6) (Create examples from definitions). Create an example for the concatenation of two languages. (refer to "operations over languages" in L3). Do not use the examples in the slides.

The definition of language concatenation (denoted $\circ$) is given as: For every language L_1 and L_2 over an alphabet Σ, $L_1 \circ L_2 = \{w_1 \circ w_2 \mid w_1 \in L_1 \text{ and } w_2 \in L_2\}$. This means the resulting language is the set of all possible strings formed by concatenating a string from L_1 with a string from L_2 .

Let $\Sigma = \{x, y, 0, 1, \_\}$.

1. **Define Language L_1 (Prefixes):**

$$L_1 = \{\text{x0}, \text{y1}\}$$

2. **Define Language L_2 (Suffixes):**

$$L_2 = \{\_\text{data}, \_\text{config}\}$$

3. **Calculate Concatenation $L_1 \circ L_2$:** We form the new strings by concatenating every string in L_1 with every string in L_2:

- $x0 \circ \_data = x0\_data$
- $x0 \circ \_config = x0\_config$
- $y1 \circ \_data = y1\_data$
- $y1 \circ \_config = y1\_config$

The resulting concatenated language is:

$$\mathbf{L_1 \circ L_2 = \{x0\_data, x0\_config, y1\_data, y1\_config\}}$$

(10) (Representing a language using sets and functions/concepts (over languages)). What is the language represented by the regular expression: $(\!|a(\!|a \Cup b|\!)^{\circledast}|\!)$? When you present your language, you can use set notation and operations over languages. For example the language for a is a, and that for $(\!|a \Cup b|\!)$ is a, b. (For regular expressions (syntax and semantics) see pages around 58 and 59 of L3.)

The language represented by the regular expression (RE) $(\!|a(\!|a \Cup b|\!)^{\circledast}|\!)$ is determined by recursively applying the semantic rules for regular expressions, which define the language represented by a given RE structure. We assume the underlying alphabet is $\Sigma = a, b$.

0.1 Language Representation Analysis

The process follows the recursive definition of RE semantics, moving from the innermost parts outward:

0.1.1 1. Base Symbols

The base symbols a and b are themselves considered regular expressions, and by definition, they represent the language consisting only of that single string:

$$L(a) = a$$

$$L(b) = b$$

0.1.2 2. Innermost Union

The term $(\!|a \Cup b|\!)$ involves the union operation ($\Cup$), which corresponds to the set union operator ($\cup$) on the languages represented by the sub-expressions:

$$L(a \Cup b) = L(a) \cup L(b) = a \cup b = a, b$$

0.1.3 3. Kleene Star Operation

The expression $(\!|a \Cup b|\!)^{\circledast}$ uses the Kleene star operation ($\circledast$), which corresponds to the Kleene star operator ($*$) applied to the language defined in the previous step. The Kleene star operation applied to a language A, denoted A, yields the set of all strings formed by concatenating zero or more strings from A:

$$L((\!|a \Cup b|\!)^{\circledast}) = (L(a \Cup b))^{=} a, b^{*}$$

The language $\mathbf{a}, \mathbf{b}^{*}$ is the set of all possible strings (including the empty string ϵ) consisting of a's and b's.

0.1.4 4. Concatenation

The expression contains an implied concatenation (juxtaposition) between the base symbol a and the result of the Kleene star operation, corresponding to Rule 3 of the RE semantics. Language concatenation ($\circ$) combines every string from the first language with every string from the second language:

$$L(a(\!|a \Cup b|\!)^*) = L(a) \circ L((\!|a \Cup b|\!)^*)$$

$$L_{intermediate} = a \circ a, b^*$$

0.1.5 5. Final Language

The outermost grouping symbols $(\!|\ldots|\!)$ serve as syntax markers and do not alter the calculated language set.

0.2 Final Language Definition

The language $L(\gamma)$ represented by the regular expression $(\!|a(\!|a \Cup b|\!)^*|\!)$ is the result of the concatenation $a \circ a, b^*$:

$$\mathbf{L}(\gamma) = \mathbf{w} \mid \mathbf{w} = \mathbf{a} \circ \mathbf{x}, \text{ where } \mathbf{x} \in \mathbf{a}, \mathbf{b}^*$$

In set notation and using operations over languages:

$$\mathbf{L}(\gamma) = \mathbf{a} \circ \mathbf{a}, \mathbf{b}^*$$

Description of the Language: This language is the set of all non-empty strings over the alphabet a, b that begin with the symbol a.