ECE-374-B: Lecture 1 - Logistics and Strings/Languages

Lecturer: Nickvash Kani

January 18, 2022

University of Illinois at Urbana Champaign

Course Administration

Instructional Staff

- Instructor:
  - Section B: Nickvash Kani
- Teaching Assistants:
  - Sung Woo Jeon
  - Junyeob Lim
  - Zhongweiyang Xu
- Undergraduate Course Assistants
  - Kunal Bhargava
  - Tri Do
  - Vaibhav Gupta
  - Ayu Seiya
- Office hours: See course webpage
- Contacting us: Use private notes on Piazza to reach course staff. Direct email only for sensitive or confidential information.

- You are in Section B of 374 which is a course cross-listed in both the ECE and CS departments.
- There is a mix of CE/CS and other majors in this course and this course will count towards your 374 requirement.
- This section has 3 dedicated TAs who will handle the discussions and office hours as well as one undergraduate course assistant who will handle online questions.
- this brings us to the next major detail:

Section A vs B

This semester, the two sections will be run completely independently.

- Different lectures.
- Different homeworks, quizzes, exams.
- Different grading policies.

Section A vs B

This semester, the two sections will be run completely independently.

- Different lectures.
- Different homeworks, quizzes, exams.
- Different grading policies.

Section B will be in-person only.

- The courses will be conducted entirely separately. The only thing common between the courses is the subject matter.
- Quizzes, HWs and exams will all be different.
- **next** One major difference is that this section is goign to be in-person only. I willm try my best to post lecture recordings online and hws/quizzes need to be performed/submitted online but exams will be doen in-person only.

Online resources

- Webpage: General information, announcements, homeworks, quizzes, course policies
  `https://canvas.illinois.edu/courses/18601`
- Gradescope: Written homework submission and grading, regrade requests
- Piazza: Announcements, online questions and discussion, contacting course staff (via private notes)

See course webpage for links

Important: check Piazza/course web page at least once each day

- Here are the online resources available.
- Section B will utilize the Canvas CMS for managing course resources and administering quizzes
- Unfortunately Gradescope integration is takign longer than exected so you all will still ahve to submit Homeworks directly on gradescope. Hw grades wil then be synced manually to Canvas a couple times in the semester.

Grading Policy

Grading Policy: Overview

- Quizzes: 4%

- Two assigned on Wednesday and due the following Wednesday.
- Covers topics from the week assigned.
- Approximately 20 quizzes assigned. Lowest 8 grades are dropped.
- You'll see the correct answers after completing the quiz!

- Quizzes are intended to assist in your conceptual understanding of the material
- Will be important for doing well on exams.
- Will be graded for completeness only. As long as you attempt the quiz, you get full marks.
- The intent is to encourage you to slowly review these concepts during the course of the semester instead of cramming everything in a few days before the exam which never ends well.

Grading Policy: Overview

- Quizzes: 4%
- Homeworks: 21%

- There will be approximately 9 HWs with 3 questions each.
- Hoemworks need to be submitted on Gradescope.
- Only the top 21 question grades will be considered for final grade calculation

- Homeworks are intended to reinforce concept details.
- There are slightly fewer homeworks than normal due to scheduling issues.
- Written homeworks can be worked on in groups of up to 3 and each group submits one written solution.
- First HW will be posted Monday.

Grading Policy: Overview

- Quizzes: 4%
- Homeworks: 21%
- Midterm/Final exams: 75% ($3 \times 25\%$)

Exam dates:

- Midterm 1: Thurs, Feb 17, 3:30pm–4:45pm
- Midterm 2: Tues, Apr 4, 3:30pm–4:45pm
- Midterm 3: Thurs, Apr 28, 3:30pm–4:45pm
- Final: TBD

One exam will be dropped Drop policies should eliminate need for conflict exams.

- The biggest change compared to previous semesters is the inclusion of a extra midterms. This means you'll have 4 exams in total.
- On the plus side your lowest exam score will be dropped. This means that if you screw up on one exam it'll be ok.
- After the third midterm at the end of the semester I will do my best to give you a expected course grade. This means that if you're satisfied with this grade, you can also skip your final. Should take some pressure off the final exam period.
- This policy also means there should be low need for conflict exams since midterms are all done during the lecture period and a missed exam due to a illness or emergency will be dropped.

Discussion Sessions/Labs

- 50min problem solving session led by TAs
- Two times a week
- Go to your assigned discussion section
- Bring pen and paper!

- Discussion sections are intended to help master concept details.
- I strongly suggest you attend these sections if you want to do well on the exams.

Advice

- Attend lectures, please ask plenty of questions.
- Attend discussion sessions.
- Don't skip homework and don't copy homework solutions. Each of you should think about all the problems on the home work - do not divide and conquer.
- Start homework early! Your mind needs time to think.
- Study regularly and keep up with the course.
- This is a course on problem solving. Solve as many as you can! Books/notes have plenty.
- This is also a course on providing rigourous proofs of correctness. Refresh your 173 background on proofs.
- Ask for help promptly. Make use of office hours/Piazza.

- Do start HWs early. This is a theory oriented course and you mind needs time to chew on new ideas.
- Additionally starting HWs early can allow you to take full advantage of OHs.

Miscellaneous

Please contact instructors if you need special accommodations.

Lectures are being taped (hopefully). The issue is that these recording systems are prone to failure. While I make no promises, I will try my best to record the lectures.

See course webpage for additional information.

Over-arching course questions

High-Level Questions

This course introduces three distinct fields of computer science research:

- Computational complexity.
  - Given infinite time and a certain machine, is it possible to solve a given problem.
- Algorithms
  - Given a deterministic Turing machine, how fast can we solve certain problems.
- Limits of computation.
  - Are there tasks that our computers cannot do and how do we identify these problems?

Why not just focus on Algorithms?

Why not just focus on Algorithms?

Problem: Driving directions between two points on a map

Why not just focus on Algorithms?

Problem: Driving directions between two points on a map
Possible solution: Pathfinding algorithm (Djikstra's, Bellman-Ford, etc.) on roadmap...

Why not just focus on Algorithms?

Problem: Driving directions between two points on a map
Possible solution: Pathfinding algorithm (Djikstra's, Bellman-Ford, etc.) on roadmap...

Is this solution complete for all origin→destination pairs?

Why not just focus on Algorithms?

Problem: Driving directions between two points on a map
Possible solution: Pathfinding algorithm (Djikstra's, Bellman-Ford, etc.) on roadmap...

Is this solution complete for all origin→destination pairs?
Example: Driving directions from London to Dublin?

Why not just focus on Algorithms?

Problem: Driving directions between two points on a map
Possible solution: Pathfinding algorithm (Djikstra's, Bellman-Ford, etc.) on roadmap...

Is this solution complete for all origin→destination pairs?
Example: Driving directions from London to Dublin?

Machine you're using limits your ability to solve a problem!

Course Structure

Course divided into three parts:

- Basic automata theory: finite state machines, regular languages, hint of context free languages/grammars, Turing Machines
- Algorithms and algorithm design techniques
- Undecidability and NP-Completeness, reductions to prove intractability of problems

Week	Tuesday Lecture	Wed Lab	Thursday Lecture	Fri Lab
Jan 17-21	Admins intro and course goals Introduction and history Harini's Videos: Lec 1	Strong Induction Soft's Inductive notes Chandra's Inductive notes Solutions	Languages and regular expressions Harini's Videos: Lec 2	Regular expressions Solutions
Jan 24-28	DFAs: intuition, definitions, closure properties Harini's Videos: Lec 3	DFA construction [solutions]	Non-Determinism, NFAs Harini's Videos: Lec 4	NFAs [solutions]
Jan 31-Feb 4	Equivalence of DFAs, NFAs, and regular expressions Harini's Videos: Lec 5	Regex to NFA to DFA (to Regex) [solutions]	Fooling Sets and Proving Non-Regularity Hrishov's DFA notes Harini's Videos: Lec 6	Proving Non-Regularity [solutions]
Feb 7 - 11	Context-free languages and grammars Harini's Videos: Lec 7	Context-free grammars [solutions]	Turing machines: history, formal definitions, examples, variations Harini's Videos: Lec 8	Turing Machines [solutions]
Feb 14 - 18	Universal Turing machines Harini's Videos: Lec 9	Midterm 1 Review (effectively office hours)	**Midterm 1 – Thursday, Feb 17 14:00-15:15** build.out hoster	No Instruction
Feb 21 - 25	Reductions & Recursion Harini's Videos: Lec 10 Notes on halting Recursion	Binary search [solutions]	Divide and conquer: Selection, Karatsuba Harini's Videos: Lec 11	Divide and Conquer [solutions]
Feb 28 - Mar 4	Backtracking Harini's Videos: Lec 12	Backtracking [solutions]	Dynamic programming Harini's Videos: Lec 13	Dynamic programming [solutions]
Mar 7 - 11	More Dynamic programming Harini's Videos: Lec 14	More Dynamic programming [solutions]	Graphs: Basic Search Chandra's Graph notes Soft's Videos: Lec 15	Even more DP [solutions]
Spring Break (Mar 12-20). Have fun.				
Mar 21 - 25	Directed Graphs, DFS, DAGs and Topological Sort Chandra's Graph notes Soft's Videos: Lec 16	Graph Modeling [solutions]	Shortest Paths: BFS and Dijkstra Chandra's Graph notes Soft's Videos: Lec 17	Shortest Paths [solutions]
Mar 28 - Apr 1	Bellman-Ford, Dynamic Programming on DAGs Chandra's Graph notes Soft's Videos: Lec 18	More Shortest Paths [solutions]	MST Algorithms	MST [solutions]
Apr 4 - 8	**Midterm 2 (Recursion/DP/Graph Algorithms) – Tuesday, Apr 5 14:00-15:15** skill.set hoster	No Instruction	Reductions Soft's Videos: Lec 21	Reductions [solutions]
Apr 11 - 15	SAT, NP and NP-Hardness Harini's Videos: Lec 22-24	NP-hardness reductions [solutions]	More NP-Hardness Harini's Videos: Lec 25-26	More NP-Hardness [solutions]
Apr 18 - 22	Undecidability 1 Harini's Videos: Lec 27	Undecidability reductions [solutions]	Undecidability 2 Harini's Videos: Lec 7	No Instruction
Apr 25 - 29	Optional review for Midterm 3	Optional Review for midterm 3 (effectively office hours)	**Midterm 3 (Reductions/P-NP/Decidability) – Thursday, Apr 28 14:00-15:15** skill.set hoster	
May 2-6	Wrap-up, closing remarks Optional review for Final Exam	Optional Review for final exam	**Reading Day** **ICES Forms Due**	
Final Exam – TBD Skill.set				

- Based on those three fields of study the course has 3 distinct parts.
- Part 1 deals with language complexity
- Part 2 introduces you to besic algorithms (in particular graphing algorithms)
- Part 3 discusses computational limits and reductions.

Goals

- Algorithmic thinking
- Learn/remember some basic tricks, algorithms, problems, ideas
- Understand/appreciate limits of computation (intractability)
- Appreciate the importance of algorithms in computer science and beyond (engineering, mathematics, natural sciences, social sciences, ...)

Formal languages and complexity (The Blue Weeks!)

Why Languages?

First 5 weeks devoted to language theory.

Why Languages?

First 5 weeks devoted to language theory.

But why study languages?

Multiplying Numbers

Consider the following problem:

Problem Given two n-digit numbers x and y, compute their product.

Grade School Multiplication
Compute "partial product" by multiplying each digit of y with x and adding the partial products.

$$\begin{array}{r} 3141 \\ \times 2718 \\ \hline 25128 \\ 3141 \\ 21987 \\ 6282 \\ \hline 8537238 \end{array}$$

Time analysis of grade school multiplication

- Each partial product: $\Theta(n)$ time
- Number of partial products: $\leq n$
- Adding partial products: n additions each $\Theta(n)$ (Why?)
- Total time: $\Theta(n^2)$
- Is there a faster way?

Fast Multiplication

- $O(n^{1.58})$ time [Karatsuba 1960] disproving Kolmogorov's belief that $\Omega(n^2)$ is best possible
- $O(n \log n \log\log n)$ [Schonhage-Strassen 1971]. **Conjecture:** $O(n \log n)$ time possible
- $O(n \log n \cdot 2^{O(\log^* n)})$ time [Furer 2008]
- $O(n \log n)$ [Harvey-van der Hoeven 2019]

Can we achieve $O(n)$? No lower bound beyond trivial one!

- New question: How **hard** is multiplication.

Equivalent Complexity

Does this mean multiplication is as complex as another problem that has a $O(n \log n)$ algorithm like sorting/QuickSort?

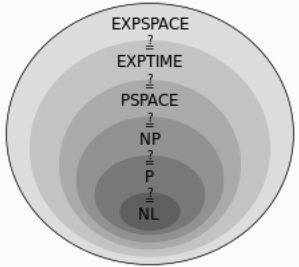
Equivalent Complexity

Does this mean multiplication is as complex as another problem that has a $O(n \log n)$ algorithm like sorting/QuickSort? How do we compare? The two problems have:

- Different inputs (two numbers vs n-element array)
- Different outputs (a number vs n-element array)
- Different entropy characteristics (from a information theory perspective)

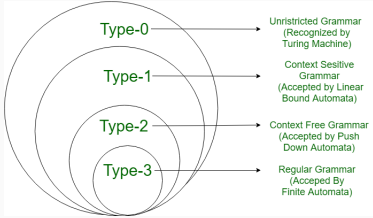
Languages, Problems and Algorithms ... oh my! II

An algorithm has a runtime complexity.



Languages, Problems and Algorithms ... oh my! III

A problem has a complexity class!



Problems do not have run-time since a problem $\neq$ the algorithm used to solve it. *Complexity classes are defined differently.*
How do we compare problems? What if we just want to know if a problem is "computable".

Algorithms, Problems and Languages ... oh my! I

Definition

1. An algorithm is a step-by-step way to solve a problem.
2. A problem is some question that we'd like answered given some input. It should be a decision problem of the form "Does a given input fulfill property X."
3. A Language is a set of strings. Given a alphabet, Σ a language is a subset of Σ^*

Algorithms, Problems and Languages ... oh my! I

Definition

1. An algorithm is a step-by-step way to solve a problem.
2. A problem is some question that we'd like answered given some input. It should be a decision problem of the form "Does a given input fulfill property X."
3. A Language is a set of strings. Given a alphabet, Σ a language is a subset of Σ^* A language is a formal realization of this problem. For problem X, the corresponding language is:

   L = {w | w is the encoding of an input y to problem X and the answer to input y for a problem X is "YES" }
   A decision problem X is "YES" is the string is in the language.

Language of multiplication

How do we define the multiplication problem as a language?

Define L as language where inputs are separated by comma and output is separated by |.

Machine accepts a x*y=z if "x*y|z" is in L. Rejects otherwise.

Language of multiplication

How do we define the multiplication problem as a language?

Define L as language where inputs are separated by comma and output is separated by |.

Machine accepts a x*y=z if "x*y|z" is in L. Rejects otherwise.

$$L_{MULT2} = \begin{Bmatrix} 1 \times 1|1, & 1 \times 2|2, & 1 \times 3|3, \dots \\ 2 \times 1|2, & 2 \times 2|4, & 2 \times 3|6, \dots \\ \vdots & \vdots & \vdots \\ n \times 1|n, & n \times 2|2n, & n \times 3|3n, \dots \end{Bmatrix} \quad (1)$$

Language of sorting

We do the same thing for sorting.

Define L as language where inputs are separated by comma and output is separated by $|$.

Machine accepts a $[i_1, i_2, \ldots] = sort(\{i_1, i_2, \ldots\})$ if "x[]|z[]" is in L. Rejects otherwise.

Language of sorting

We do the same thing for sorting.

Define L as language where inputs are separated by comma and output is separated by $|$.

Machine accepts a $[i_1, i_2, \ldots] = sort(\{i_1, i_2, \ldots\})$ if "x[]|z[]" is in L. Rejects otherwise.

$$L_{Sort2} = \begin{Bmatrix} 1,1|1,1 & 1,2|1,2 & 1,3|1,3,\ldots \\ 2,1|1,2, & 2,2|2,2, & 2,3|2,3,\ldots \\ \vdots & \vdots & \vdots \\ n,1|1,n, & n,2|2,n, & n,3|3,n,\ldots \end{Bmatrix} \tag{2}$$

Language of sorting

We do the same thing for sorting.

Define L as language where inputs are separated by comma and output is separated by $|$.

Machine accepts a $[i_1, i_2, \ldots] = sort(\{i_1, i_2, \ldots\})$ if "x[]|z[]" is in L. Rejects otherwise.

$$L_{Sort2} = \begin{Bmatrix} 1,1|1,1 & 1,2|1,2 & 1,3|1,3,\ldots \\ 2,1|1,2, & 2,2|2,2, & 2,3|2,3,\ldots \\ \vdots & \vdots & \vdots \\ n,1|1,n, & n,2|2,n, & n,3|3,n,\ldots \end{Bmatrix} \tag{2}$$

If the same type of machine can recognize both languages, then that gives us an upperbound top their hardness.

How do we formulate languages?

Strings

Alphabet

An alphabet is a **finite** set of symbols.

Examples of alphabets:

- $\Sigma = \{0, 1\}$,
- $\Sigma = \{a, b, c, \ldots, z\}$,
- ASCII.
- UTF8.
- $\Sigma = \{\langle \text{moveforward} \rangle, \langle \text{moveback} \rangle, \langle \text{moveleft} \rangle, \langle \text{moveright} \rangle\}$

String Definition

Definition

1. A string/word over Σ is a **finite sequence** of symbols over Σ. For example, '0101001', '*string*', '$\langle \text{moveback} \rangle \langle \text{rotate90} \rangle$'
2. $x \cdot y \equiv xy$ is the concatenation of two strings
3. The length of a string w (denoted by $|w|$) is the number of symbols in w. For example, $|101| = 3$, $|\epsilon| = 0$
4. For integer $n \geq 0$, Σ^n is set of all strings over Σ of length n. Σ^* is the set of all strings over Σ.
5. Σ^* set of all strings of all lengths including empty string.

Question: $\{'a', 'c'\}^* =$

- $= \{\varepsilon, a, c, aa, ac, cc, ca, aaa, \}$ Set containing one element,the element is a empty set.

Emptiness

- ϵ is a string containing no symbols. It is not a set
- $\{\epsilon\}$ is a set containing one string: the empty string. It is a set, not a string.
- $\emptyset$ is the empty set. It contains no strings.

Question: What is $\{\emptyset\}$

- $= \{\{\}, \}$ Set containing one element,the element is a empty set.

Concatenation and properties

- If x and y are strings then xy denotes their concatenation.
- Concatenation defined recursively :
  - $xy = y$ if $x = \epsilon$
  - $xy = a(wy)$ if $x = aw$
- xy sometimes written as $x \bullet y$.
- concatenation is associative: $(uv)w = u(vw)$ hence write $uvw \equiv (uv)w = u(vw)$
- **not** commutative: uv not necessarily equal to vu
- The identity element is the empty string ϵ:

$$\epsilon u = u\epsilon = u.$$

Substrings, prefixes, Suffixes

Definition
v is substring of $w \iff$ there exist strings x, y such that $w = xvy$.

- If $x = \epsilon$ then v is a prefix of w
- If $y = \epsilon$ then v is a suffix of w

Subsequence

A subsequence of a string $w[1...n]$ is either a subsequence of $w[2...n]$ or $w[1]$ followed by a subsequence of $w[2...n]$.

Example
kapa is a supsequence of *knapsack*

Subsequence

A subsequence of a string $w[1...n]$ is either a subsequence of $w[2...n]$ or $w[1]$ followed by a subsequence of $w[2...n]$.

Example
kapa is a supsequence of *knapsack*

Question: How many sub-sequences are there in a string $|w| = 5$?

- $= 2^5 = 32$ if there are no repeating characters.

String exponent

Definition
If w is a string then w^n is defined inductively as follows:
$w^n = \epsilon$ if $n = 0$
$w^n = ww^{n-1}$ if $n > 0$

Question: $(blah)^3 =$.

- $=$blahblahblah.

Rapid-fire questions -strings

Answer the following questions taking $\Sigma = \{0, 1\}$.

1. What is Σ^0?
2. How many elements are there in Σ^n?
3. If $|u| = 2$ and $|v| = 3$ then what is $|u \bullet v|$?
4. Let u be an arbitrary string in Σ^*. What is ϵu? What is $u\epsilon$?

- $= \{\epsilon\}$
- $= 2^n$
- $= 5$
- $= u$

Languages

Languages

Definition
A language L is a set of strings over Σ. In other words $L \subseteq \Sigma^*$.

Languages

Definition
A language L is a set of strings over Σ. In other words $L \subseteq \Sigma^*$.

Standard set operations apply to languages.

- For languages A, B the concatenation of A, B is $AB = \{xy \mid x \in A, y \in B\}$.
- For languages A, B, their union is $A \cup B$, intersection is $A \cap B$, and difference is $A \setminus B$ (also written as $A - B$).
- For language $A \subseteq \Sigma^*$ the complement of A is $\bar{A} = \Sigma^* \setminus A$.

Set Concatenation

Definition
Given two sets X and Y of strings (over some common alphabet Σ) the concatenation of X and Y is

$$XY = \{xy \mid x \in X, y \in Y\} \tag{3}$$

Question: $X = \{fido, rover, spot\}$, $Y = \{fluffy, tabby\} \implies XY =$.

- $=\{$fidofluffy, fidotabby, roverfluffy, rovertabby, spotfluffy, sppottabby$\}$

Σ^* and languages

Definition

1. Σ^n is the set of all strings of length n. Defined inductively:
   $\Sigma^n = \{\epsilon\}$ if $n = 0$
   $\Sigma^n = \Sigma\Sigma^{n-1}$ if $n > 0$
2. $\Sigma^* = \cup_{n \geq 0} \Sigma^n$ is the set of all finite length strings
3. $\Sigma^+ = \cup_{n \geq 1} \Sigma^n$ is the set of non-empty strings.

Definition
A language L is a set of strings over Σ. In other words $L \subseteq \Sigma^*$.

Question: Does Σ^* have strings of infinite length?

- No.

Rapid-Fire questions - Languages

Problem
Consider languages over $\Sigma = \{0, 1\}$.

1. *What is* $\emptyset^0$*?*
2. *If* $|L| = 2$*, then what is* $|L^4|$*?*
3. *What is* $\emptyset^*$, $\{\epsilon\}^*$, ϵ^**?*
4. *For what* L *is* L^* *finite?*
5. *What is* $\emptyset^+$*?*
6. *What is* $\{\epsilon\}^+$, ϵ^+*?*

- $= \{\epsilon\}$
- $= 16$
- $= \{\epsilon\}$
- $L = \emptyset$ or $\{\varepsilon\}$
- $\{\}$,
- $= \{\varepsilon\}^1 \cup \{\varepsilon\}^2 \cup \ldots = \{\varepsilon\}$, $\varepsilon^+ = \{\{\varepsilon^1\} \cup \{\varepsilon^2\} \cup \ldots\} = \{\varepsilon\}$

Terminology Review

Let's review what we learned.

- A character(a, b, c, x) is a unit of information represented by a symbol: (letters, digits, whitespace)
- A alphabet(Σ) is a set of characters
- A string(w) is a sequence of characters
- A language(A, B, C, L) is a set of strings

Terminology Review

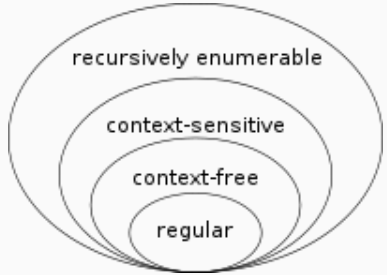
Let's review what we learned.

- A character(a, b, c, x) is a unit of information represented by a symbol: (letters, digits, whitespace)
- A alphabet(Σ) is a set of characters
- A string(w) is a sequence of characters
- A language(A, B, C, L) is a set of strings
- A grammar(G) is a set of rules that defines the strings that belong to a language

- Before we leave I want to throw one last definition at you and that is Grammar.
- It turns out groupign problems according to the grammar that generates their language is pretty useful

Languages: easiest, easy, hard, really hard, really^n hard

- Regular languages.
  - Regular expressions.
  - DFA: Deterministic finite automata.
  - NFA: Non-deterministic finite automata.
  - Languages that are not regular.
- Context free languages (stack).
- Turing machines: Decidable languages.
- TM Undecidable/unrecognizable languages (halting theorem).



Induction on strings

Inductive proofs on strings

Inductive proofs on strings and related problems follow inductive definitions.

Definition
The reverse w^R of a string w is defined as follows:

- $w^R = \epsilon$ if $w = \epsilon$
- $w^R = x^R a$ if $w = ax$ for some $a \in \Sigma$ and string x

Inductive proofs on strings

Inductive proofs on strings and related problems follow inductive definitions.

Definition
The reverse w^R of a string w is defined as follows:

- $w^R = \epsilon$ if $w = \epsilon$
- $w^R = x^R a$ if $w = ax$ for some $a \in \Sigma$ and string x

Theorem
Prove that for any strings $u, v \in \Sigma^*$, $(uv)^R = v^R u^R$.

Example: $(dog \cdot cat)^R = (cat)^R \cdot (dog)^R = tacgod$.

Principle of mathematical induction

Induction is a way to prove statements of the form $\forall n \geq 0, P(n)$ where $P(n)$ is a statement that holds for integer n.

Example: Prove that $\sum_{i=0}^{n} i = n(n+1)/2$ for all n.

Induction template:

- **Base case:** Prove $P(0)$
- **Induction hypothesis:** Let $k > 0$ be an **arbitrary** integer. Assume that $P(n)$ holds for any $n \leq k$.
- **Induction Step:** Prove that $P(n)$ holds, for $n = k + 1$.

Structured induction

- Unlike simple cases we are working with...
- ...induction proofs also work for more complicated "structures".
- Such as strings, tuples of strings, graphs etc.
- See class notes on induction for details.

Proving the theorem

Theorem
Prove that for any strings $u, v \in \Sigma^*$, $(uv)^R = v^R u^R$.

Proof: by induction.
On what?? $|uv| = |u| + |v|$?
$|u|$?
$|v|$?

What does it mean "induction on $|u|$"?

By induction on $|u|$

Theorem
Prove that for any strings $u, v \in \Sigma^*$, $(uv)^R = v^R u^R$.

Proof by induction on $|u|$ means that we are proving the following.

Base case: Let u be an arbitrary string of length 0. $u = \epsilon$ since there is only one such string. Then

$(uv)^R = (\epsilon v)^R = v^R = v^R \epsilon = v^R \epsilon^R = v^R u^R$

By induction on $|u|$

Theorem
Prove that for any strings $u, v \in \Sigma^*$, $(uv)^R = v^R u^R$.

Proof by induction on $|u|$ means that we are proving the following.

Base case: Let u be an arbitrary string of length 0. $u = \epsilon$ since there is only one such string. Then

$(uv)^R = (\epsilon v)^R = v^R = v^R \epsilon = v^R \epsilon^R = v^R u^R$

Induction hypothesis: $\forall n \geq 0$, for any string u of length n:

For all strings $v \in \Sigma^*$, $(uv)^R = v^R u^R$.

By induction on $|u|$

Theorem
Prove that for any strings $u, v \in \Sigma^*$, $(uv)^R = v^R u^R$.

Proof by induction on $|u|$ means that we are proving the following.

Base case: Let u be an arbitrary string of length 0. $u = \epsilon$ since there is only one such string. Then

$(uv)^R = (\epsilon v)^R = v^R = v^R \epsilon = v^R \epsilon^R = v^R u^R$

Induction hypothesis: $\forall n \geq 0$, for any string u of length n:

For all strings $v \in \Sigma^*$, $(uv)^R = v^R u^R$.

No assumption about v, hence statement holds for all $v \in \Sigma^*$.

Inductive step

- Let u be an arbitrary string of length $n > 0$. Assume inductive hypothesis holds for all strings w of length $< n$.
- Since $|u| = n > 0$ we have $u = ay$ for some string y with $|y| < n$ and $a \in \Sigma$.
- Then

Inductive step

- Let u be an arbitrary string of length $n > 0$. Assume inductive hypothesis holds for all strings w of length $< n$.
- Since $|u| = n > 0$ we have $u = ay$ for some string y with $|y| < n$ and $a \in \Sigma$.
- Then

$$(uv)^R \quad =$$

Inductive step

- Let u be an arbitrary string of length $n > 0$. Assume inductive hypothesis holds for all strings w of length $< n$.
- Since $|u| = n > 0$ we have $u = ay$ for some string y with $|y| < n$ and $a \in \Sigma$.
- Then

$$\begin{aligned}
(uv)^R &= ((ay)v)^R \\
&= (a(yv))^R \\
&= (yv)^R a^R \\
&= (v^R y^R) a^R \\
&= v^R (y^R a^R) \\
&= v^R (ay)^R \\
&= v^R u^R
\end{aligned}$$

Another example!

Theorem
Prove that for any strings x and y, $|xy| = |x| + |y|$

Base case: Let x be an arbitrary string of length 0. $x = \epsilon$ since there is only one such string. Then

$$|xy| = |\epsilon y| = |y| = |y| + |\epsilon| = |y| + |x| = |x| + |y|$$

Another example!

Theorem
Prove that for any strings x and y, $|xy| = |x| + |y|$

Base case: Let x be an arbitrary string of length 0. $x = \epsilon$ since there is only one such string. Then

$$|xy| = |\epsilon y| = |y| = |y| + |\epsilon| = |y| + |x| = |x| + |y|$$

Induction hypothesis: $\forall n \geq 0$, for any string x of length n:

For all strings $y \in \Sigma^*$, $|xy| = |x| + |y|$.

Another example!

Theorem
Prove that for any strings x and y, $|xy| = |x| + |y|$

Base case: Let x be an arbitrary string of length 0. $x = \epsilon$ since there is only one such string. Then

$$|xy| = |\epsilon y| = |y| = |y| + |\epsilon| = |y| + |x| = |x| + |y|$$

Induction hypothesis: $\forall n \geq 0$, for any string x of length n:

For all strings $y \in \Sigma^*$, $|xy| = |x| + |y|$.

No assumption about y, hence statement holds for all $y \in \Sigma^*$.

Another Example: Inductive step

- Let x be an arbitrary string of length $n > 0$. Assume inductive hypothesis holds for all strings w of length $< n$.
- Since $|x| = n > 0$ we have $x = az$ for some string z with $|z| < n$ and $a \in \Sigma$.
- Then

Another Example: Inductive step

- Let x be an arbitrary string of length $n > 0$. Assume inductive hypothesis holds for all strings w of length $< n$.
- Since $|x| = n > 0$ we have $x = az$ for some string z with $|z| < n$ and $a \in \Sigma$.
- Then

$$
\begin{aligned}
|xy| &= |(az)y| \\
&= |a(zy)| \\
&= 1 + |zy| && \text{recursive def of string length} \\
&= 1 + |z| + |y| && \text{inductive hypothesis} \\
&= (1 + |z|) + |y| \\
&= |az| + |y| && \text{recursive def of string length} \\
&= |x| + |y|
\end{aligned}
$$