“Guide” to the CS103 Final

Let’s do this!

benson97, amauro

Table of Contents

Studying for the exam

We can do this!

- If you have seen the lectures, you'll know how to solve the problems
- If you know how to do the problem sets, you'll be prepared

You may wonder... OK, Benson, Annika, that's nice and all, **but**

What does "know" even mean?

... Fair question. For the next hour-and-a-half or so, we'll do our best to answer.

Studying for the exam

What's the best way to study? There's one answer[1]:

Practice the material you are least comfortable with!

Warning

Don't read solutions before you write your own answer.

You would be better off re-doing problems we've seen before.

Warning

Don't try to solve dozens of dozens of problems.

It's not efficient. Theory courses do not award rote memorization.

[1] More or less.

Studying for the exam

In CS103, **there are no tricks.** We expect you to be familiar[2] with

- 8 to 10 topics (depends on how you count[3])
- "Mathematical thinking" fundamentals

Reminder

The best way to prepare is to **practice the material** you are **least comfortable with!!!**

So this is what we'll be emphasizing today.

[2] This goes without saying, but also, the practice exams are representative of our expectations
[3] I would say 5, but I think most people would disagree. — Benson

EXTREMELY IMPORTANT WARNING!

For this review session, we will “work through” problems.

Instead of presenting solutions, we will instead demonstrate our thought process while working through problems.

EXTREMELY IMPORTANT WARNING!

This means that slides may be incorrect and/or incomplete!

To our live audience, that means that if you see something that’s incorrect — please point it out! The earlier we can catch an error, the better!

To students reviewing these slides, this means that the correct solution may not be presented until the very end of a section.

Graph Homomorphisms

Tag(s): Functions, Graphs, Injections

Graph Homomorphisms

Let's begin with a new definition. Suppose that $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ are graphs. We'll say that a **homomorphism** from G_1 to G_2 is a function $h : V_1 \to V_2$ with the following property:

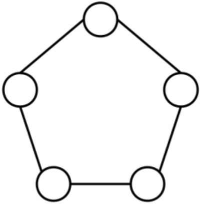
$$\forall u \in V_1. \forall v \in V_1. (\{u, v\} \in E_1 \to \{h(u), h(v)\} \in E_2).$$

Graph Homomorphisms

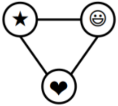
Let's begin with a new definition. Suppose that $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ are graphs. We'll say that a **homomorphism** from G_1 to G_2 is a function $h : V_1 \to V_2$ with the following property:

$$\forall u \in V_1 . \forall v \in V_1 . (\{u, v\} \in E_1 \to \{h(u), h(v)\} \in E_2).$$

(i) Below are pictures of two graphs G_1 and G_2. Find a homomorphism from G_1 to G_2 To give your answer, label each node of with the corresponding node in that the homomorphism maps it to.



The graph G_1



The graph G_2

First steps:

1. Try and unpack and intuitively understand this definition
2. Figure out how to check if something satisfies this definition

First steps:

1. Try and unpack and intuitively understand this definition
2. Figure out how to check if something satisfies this definition

We'll say that a **homomorphism** from G_1 to G_2 is a function $h : V_1 \to V_2$ with the following property:

$$\forall u \in V_1. \forall v \in V_1. (\{u, v\} \in E_1 \to \{h(u), h(v)\} \in E_2).$$

First steps:

1. Try and unpack and intuitively understand this definition
2. Figure out how to check if something satisfies this definition

We'll say that a **homomorphism** from G_1 to G_2 is a function $h : V_1 \to V_2$ with the following property:

$$\forall u \in V_1. \forall v \in V_1. (\{u, v\} \in E_1 \to \{h(u), h(v)\} \in E_2).$$

To do this: walk through the first-order logic carefully with these goals in mind. It may be less dense than it seems at first!

First steps:

1. Try and unpack and intuitively understand this definition
2. Figure out how to check if something satisfies this definition

We'll say that a **homomorphism** from G_1 to G_2 is a function $h : V_1 \to V_2$ with the following property:

$$\forall u \in V_1 . \forall v \in V_1 . (\{u, v\} \in E_1 \to \{h(u), h(v)\} \in E_2).$$

To do this: walk through the first-order logic carefully with these goals in mind. It may be less dense than it seems at first!

1. Intuition: Whenever two vertices are neighbors in G_1, their images must still be neighbors in G_2

First steps:

1. Try and unpack and intuitively understand this definition
2. Figure out how to check if something satisfies this definition

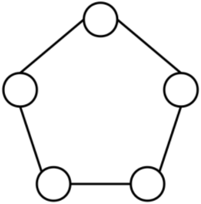
We'll say that a **homomorphism** from G_1 to G_2 is a function $h : V_1 \to V_2$ with the following property:

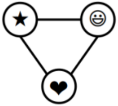
$$\forall u \in V_1. \forall v \in V_1. (\{u, v\} \in E_1 \to \{h(u), h(v)\} \in E_2).$$

To do this: walk through the first-order logic carefully with these goals in mind. It may be less dense than it seems at first!

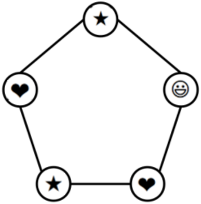
1. Intuition: Whenever two vertices are neighbors in G_1, their images must still be neighbors in G_2
2. How to check: For each edge $\{u, v\}$, make sure that $h(v)$ and $h(u)$ are connected by an edge.

Back to our problem:
(i) Below are pictures of two graphs G_1 and G_2. Find a homomorphism from G_1 to G_2 To give your answer, label each node of with the corresponding node in that the homomorphism maps it to.

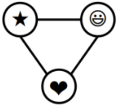

The graph G_1


The graph G_2

Method 1: start sending adjacent pairs of vertices to adjacent pairs of vertices and see what happens!

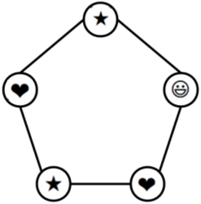


The graph G_1

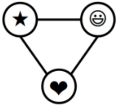


The graph G_2

Method 1: start sending adjacent pairs of vertices to adjacent pairs of vertices and see what happens!



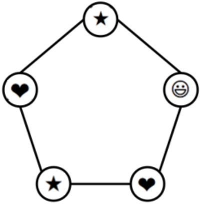
The graph G_1



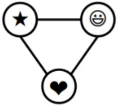
The graph G_2

Method 2: Notice that in G_2 the only pairs vertices not connected are if they are identical - all we need is to send adjacent nodes to distinct nodes (same symbol not adjacent in G_1)

Method 1: start sending adjacent pairs of vertices to adjacent pairs of vertices and see what happens!



The graph G_1

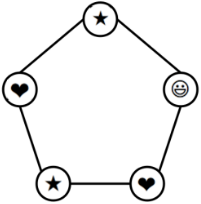


The graph G_2

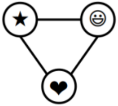
Method 2: Notice that in G_2 the only pairs vertices not connected are if they are identical - all we need is to send adjacent nodes to distinct nodes (same symbol not adjacent in G_1)

Last step: Double-check that our construction works using the "how to check" we found earlier

Method 1: start sending adjacent pairs of vertices to adjacent pairs of vertices and see what happens!



The graph G_1



The graph G_2

Method 2: Notice that in G_2 the only pairs vertices not connected are if they are identical - all we need is to send adjacent nodes to distinct nodes (same symbol not adjacent in G_1)

Last step: Double-check that our construction works using the "how to check" we found earlier
(For each edge $\{u, v\}$, make sure that $h(v)$ and $h(u)$ are connected by an edge.)

A graph is called **complete** if:

$$\forall u \in V. \forall v \in V. (u \neq v \leftrightarrow \{u, v\} \in E).$$

(ii) Let $G = (V, E)$ be a complete graph and let $h : V \to V$ be an arbitrary function. Prove that h is injective if and only if it's a homomorphism from to itself.

1. Unpack definition of complete: The graph contains all edges
2. Recall definition of injective: A function is injective if whenever $h(x_1) = h(x_2)$ then $x_1 = x_2$.

Here, we will just try and prove one direction.

Want to show: if $h : V \to V$ is a homomorphism and V is complete, then h is injective.

Want to show: if $h : V \to V$ is a homomorphism and V is complete, then h is injective. Try and "follow your nose" by unpacking definitions:

Want to show: if $h : V \to V$ is a homomorphism and V is complete, then h is injective. Try and "follow your nose" by unpacking definitions:

- Suppose $h(x_1) = h(x_2)$

Want to show: if $h : V \to V$ is a homomorphism and V is complete, then h is injective. Try and "follow your nose" by unpacking definitions:

- Suppose $h(x_1) = h(x_2)$
- If $x_1 = x_2$ we would be done, so suppose $x_1 \neq x_2$.

Want to show: if $h : V \to V$ is a homomorphism and V is complete, then h is injective. Try and "follow your nose" by unpacking definitions:

- Suppose $h(x_1) = h(x_2)$
- If $x_1 = x_2$ we would be done, so suppose $x_1 \neq x_2$.
- What does V being complete imply here?

Want to show: if $h : V \to V$ is a homomorphism and V is complete, then h is injective. Try and "follow your nose" by unpacking definitions:

- Suppose $h(x_1) = h(x_2)$
- If $x_1 = x_2$ we would be done, so suppose $x_1 \neq x_2$.
- What does V being complete imply here?
- Then x_1 and x_2 neighbors

Want to show: if $h : V \to V$ is a homomorphism and V is complete, then h is injective. Try and "follow your nose" by unpacking definitions:

- Suppose $h(x_1) = h(x_2)$
- If $x_1 = x_2$ we would be done, so suppose $x_1 \neq x_2$.
- What does V being complete imply here?
- Then x_1 and x_2 neighbors
- What does h being a homomorphism imply here?

Want to show: if $h : V \to V$ is a homomorphism and V is complete, then h is injective. Try and "follow your nose" by unpacking definitions:

- Suppose $h(x_1) = h(x_2)$
- If $x_1 = x_2$ we would be done, so suppose $x_1 \neq x_2$.
- What does V being complete imply here?
- Then x_1 and x_2 neighbors
- What does h being a homomorphism imply here?
- $h(x_1)$ and $h(x_2)$ must be neighbors as well

Want to show: if $h : V \to V$ is a homomorphism and V is complete, then h is injective. Try and "follow your nose" by unpacking definitions:

- Suppose $h(x_1) = h(x_2)$
- If $x_1 = x_2$ we would be done, so suppose $x_1 \neq x_2$.
- What does V being complete imply here?
- Then x_1 and x_2 neighbors
- What does h being a homomorphism imply here?
- $h(x_1)$ and $h(x_2)$ must be neighbors as well
- Is this possible?

Want to show: if $h : V \to V$ is a homomorphism and V is complete, then h is injective. Try and "follow your nose" by unpacking definitions:

- Suppose $h(x_1) = h(x_2)$
- If $x_1 = x_2$ we would be done, so suppose $x_1 \neq x_2$.
- What does V being complete imply here?
- Then x_1 and x_2 neighbors
- What does h being a homomorphism imply here?
- $h(x_1)$ and $h(x_2)$ must be neighbors as well
- Is this possible?
- No! $h(x_1) = h(x_2)$

Want to show: if $h : V \to V$ is a homomorphism and V is complete, then h is injective. Try and "follow your nose" by unpacking definitions:

- Suppose $h(x_1) = h(x_2)$
- If $x_1 = x_2$ we would be done, so suppose $x_1 \neq x_2$.
- What does V being complete imply here?
- Then x_1 and x_2 neighbors
- What does h being a homomorphism imply here?
- $h(x_1)$ and $h(x_2)$ must be neighbors as well
- Is this possible?
- No! $h(x_1) = h(x_2)$
- Therefore $x_1 = x_2$.

Want to show: if $h : V \to V$ is a homomorphism and V is complete, then h is injective. Try and "follow your nose" by unpacking definitions:

- Suppose $h(x_1) = h(x_2)$
- If $x_1 = x_2$ we would be done, so suppose $x_1 \neq x_2$.
- What does V being complete imply here?
- Then x_1 and x_2 neighbors
- What does h being a homomorphism imply here?
- $h(x_1)$ and $h(x_2)$ must be neighbors as well
- Is this possible?
- No! $h(x_1) = h(x_2)$
- Therefore $x_1 = x_2$.

Then you would write this down in the form of a proof. This is an example of a thought process to come up with a solution!

Domatic Partitions

Tag(s): Graphs, Pigeonhole Principle

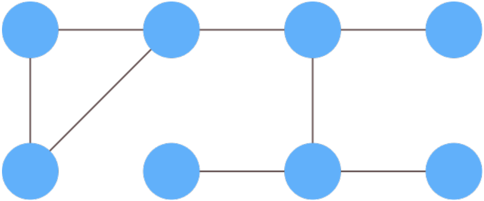
Domatic Partitions

A **domatic partition** of the nodes of a graph $G(V, E)$ is a set

$$\{V_1, ..., V_n\}$$

such that each V_i is a dominating set of G, and every node $v \in V$ belongs to exactly one of the V_i's. The **domatic number** of a graph, denoted $d(G)$, is the maximum number of dominating sets in a domatic partition of V.

(i) The graph shown below has domatic number two. Find two examples of domatic partitions of that graph into two dominating sets. No justification is necessary.



Domatic Partitions

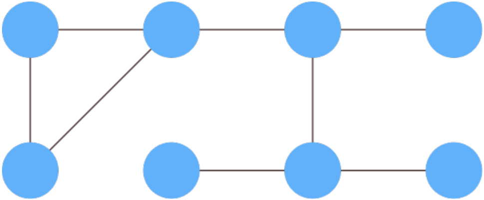
Uh oh.

This is a dense definition. We can't just write a proof.

That's why examples are important! And part (i) is meant to guide us through this process. So while we work on (i), let's process the definition of domatic partitions.

You may have noticed that, in CS103, this is the first part of many problems. In future proof-based courses, if you're ever stuck (we often are), we should start with examples.

Domatic Partitions



Ummm.. uhhh.. what's a dominating set?

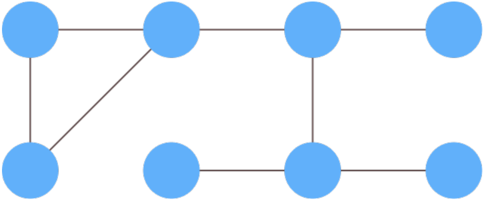
Definition

A **dominating set** in $G(V, E)$ is a set $D \subseteq V$ with with the following property:

$$\forall v \in V. (v \notin D \to \exists u \in D. \{u, v\} \in E)$$

OK! Wow! Now we understand dominating sets! Right?

Domatic Partitions



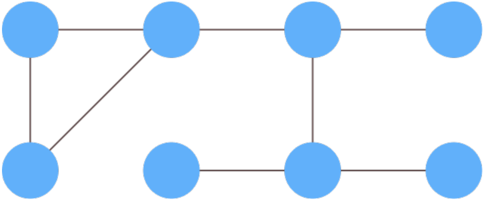
Definition, informal edition

A **dominating set** is a set of vertices that are connected to every other vertex.

Much better! (: and (informally, again) domatic partitions are sets of dominating sets, with some extra stuff.

So... back to our problem I guess...

Domatic Partitions



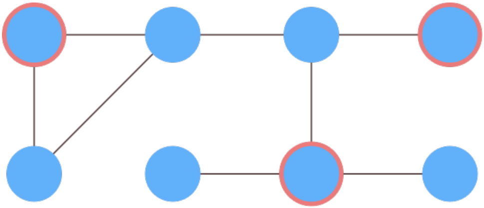
(i) Find two examples of domatic partitions of that graph into two dominating sets

Using our intuition

We want vertices that are connected to a lot of other vertices!

Let's just find those.

Domatic Partitions

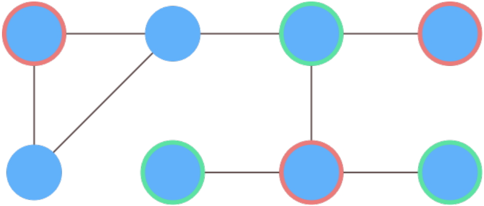


That looks pretty good. Let's quickly look back our problem...

i) Find two examples of domatic partitions of that graph into **two dominating sets**

OK! Round 2!

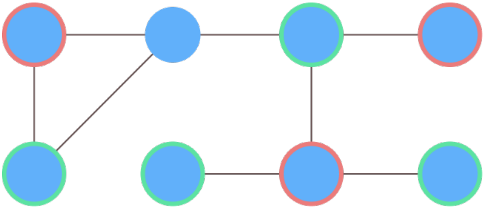
Domatic Partitions



Me: Great, that's correct, right?

Narrator: This was not correct.

Domatic Partitions



Me: Great, that has to be correct, right?

Narrator: Nope.

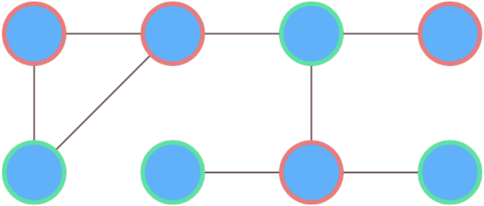
Definition

A **domatic partition** of the nodes of a graph $G(V, E)$ is a set

$$\{V_1, ..., V_n\}$$

such that... **every node** $v \in V$ **belongs to exactly one of the** V_i**'s**.

Domatic Partitions



Me: Surely, this is correct.

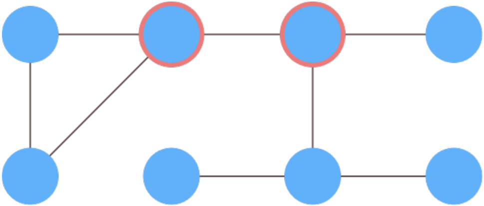
Narrator: ...

i) Find **two examples** of domatic partitions of that graph into two dominating sets

Domatic Partitions

Me: AsdkjlkqwjLQJOIQjqwegotthisoiwjdlkadwjl!!

OK, round 3, but this time, let's spice things up.



Note that we explicitly created a different set!

And now it's your turn (:

Domatic Partitions

(ii) As a refresher, the degree of a node in a graph G, denoted $d(G)$, is the number of nodes that is adjacent to. Equivalently, it's the number of edges touching. Prove that if is a graph $G(V, E)$, then $d(G) \leq deg(v) + 1$ for each node $v \in V$.

Domatic Partitions

(ii) Prove that if is a graph $G(V, E)$, then $d(G) \leq deg(v) + 1$ for each node $v \in V$.

Note

If you have attended our OH (and thank you for all who did!) you know that we like to write down whatever information we have, preferably in an easy to reference place.

This is what we did for this problem! But it's on some scratch paper, and not on these slides.

Domatic Partitions

(ii) Prove that if is a graph $G(V, E)$, then $d(G) \leq deg(v) + 1$ for each node $v \in V$.

Thought 1

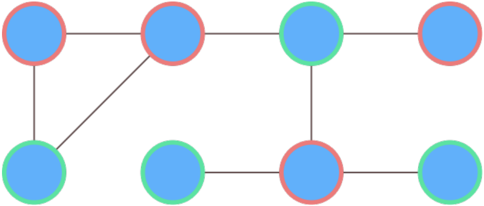
For each $v \in V$? Why? Isn't $d(G)$ constant?

$d(G)$ must be restricted by the "smallest" v. (The v with the smallest degree.)

Let's look at our graph again.

Domatic Partitions

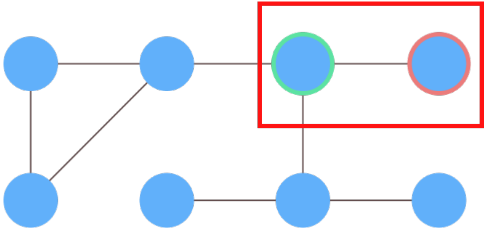
(ii) Prove that if is a graph $G(V, E)$, then $d(G) \leq deg(v) + 1$ for each node $v \in V$.



Why can't we have $d(G) > 2$? What would happen if we tried to create 3 partitions?

Domatic Partitions

(ii) Prove that if is a graph $G(V, E)$, then $d(G) \leq deg(v) + 1$ for each node $v \in V$.



We immediately run into a problem!!

Thought 2

If we have too many $V_1, ..., V_n$, our smallest vertex doesn't have enough neighbors!

Domatic Partitions

(ii) Prove that if is a graph $G(V, E)$, then $d(G) \leq deg(v) + 1$ for each node $v \in V$.

Now, let's try to describe what we just did. We...

- Found the smallest node
- Added it to V_1
- Added a neighbor to V_2

Then we ran out of space.

Thought 3

Remember when we forgot to add a vertex to a V_i in (i)?

We should try to add a vertex to each V_i, and see if we run into problems.

Domatic Partitions

(ii) Prove that if is a graph $G(V, E)$, then $d(G) \leq deg(v) + 1$ for each node $v \in V$.

For the sake of contradiction, let's say we have $d(G) > \deg(v) + 1$

Let's think of v as the node with the fewest edges.

Add a neighbor of v to each V_i in our domatic partition.

... we will run out of neighbors! Some V_i will not have a neighbor.

Thought 4

Me: Does this matter?

Narrator: It does. He won't find out for another 10 minutes though.

Domatic Partitions

(ii) Prove that if is a graph $G(V, E)$, then $d(G) \leq deg(v) + 1$ for each node $v \in V$.

Uh oh.

We are officially stuck. What should we do?

This is when it's helpful to look back on our definitions. Which is what we did here. What definition relates to neighbors?

If we calmly look through our notes...

Definition

A **dominating set** in $G(V, E)$ is a set $D \subseteq V$ with with the following property:

$$\forall v \in V. (v \notin D \to \exists u \in D. \{u, v\} \in E)$$

Domatic Partitions

(ii) Prove that if is a graph $G(V, E)$, then $d(G) \leq deg(v) + 1$ for each node $v \in V$.

V_i must have a vertex **that's a neighbor to** v **OR** v **itself!!**

But we're out of neighbors, so we must have reached some sort of contradiction.

Now it's your turn!

The logic we have above is OK, but it's not complete.

As we write our proof, we may notice a few pitfalls. You'll have to fix them.

If it's helpful, you may recall that this was filed under "Pigeonhole Principle". We don't necessarily have to frame our solution as such (I did not) but... could be relevant.

Avoid the Triple B

Tag(s): DFAs, Regexes

Avoid the Triple B

Let $\Sigma = \{a, b\}$ and $L = \{w \in \Sigma^* \mid w \text{ does not contain } www \text{ as a substring}\}$

- Design a DFA for L
- Write a regular expression for L

Avoid the Triple B

Let $\Sigma = \{a, b\}$ and $L = \{w \in \Sigma^* \mid w \text{ does not contain } www \text{ as a substring}\}$

- Design a DFA for L
- Write a regular expression for L

For the sake of time, we'll skip the DFA. But it's not too bad and if you have any questions, feel free to ask on Ed.

Avoid the Triple B

Let $\Sigma = \{a, b\}$ and $L = \{w \in \Sigma^* \mid w \text{ does not contain } www \text{ as a substring}\}$. Write a regular expression for L.

How should we approach this problem?

Like a DFA or NFA, it is nice to consider cases.

A Promising Start?

Because bbb cannot be in w, $\{bbbb, bbbbb, \ldots\}$ can't be in w either.

Therefore, we can break this into three cases! Each segment of w has 0, 1, or 2 b's.

This is not too different from a DFA — we create our regex based upon the number of b's allowed at any given time.

Avoid the Triple B

Let $\Sigma = \{a, b\}$ and $L = \{w \in \Sigma^* \mid w \text{ does not contain } www \text{ as a substring}\}$.
Write a regular expression for L.

How can we implement this? In short, our current plan to union the languages of 3 regexes:

$$\{w \text{ has no b's}\} \cup \{w \text{ has b's as a substring}\} \cup \{w \text{ has no bb's as a substring}\}$$

Avoid the Triple B

Let $\Sigma = \{a, b\}$ and $L = \{w \in \Sigma^* \mid w$ does not contain www as a substring$\}$. Write a regular expression for L.

Case 1: w has no b. Then there are only a's. Our regex is then $\mathbf{a}^*$.

Avoid the Triple B

Let $\Sigma = \{a, b\}$ and $L = \{w \in \Sigma^* \mid w \text{ does not contain } www \text{ as a substring}\}$. Write a regular expression for L.

Case 2: w has the substring b. What does this look like?

- b
- ab
- ba

Our first attempt: $\mathbf{b}^*(\mathbf{ab})^*(\mathbf{ba})^*$

When does this work?

Works!	b, abab, babab, ba
Does not work!	bbb, aab, abba

Avoid the Triple B

Let $\Sigma = \{a, b\}$ and $L = \{w \in \Sigma^* \mid w$ does not contain www as a substring$\}$. Write a regular expression for L.

Case 2: w has the substring b. What does this look like?

- b
- $abab$

Let's take a step back...

Based upon our examples, what problems do you spot?

When does this work?

Works!	b, abab, babab, ba
Does not work!	bbb, aab, abba

Avoid the Triple B

Let $\Sigma = \{a, b\}$ and $L = \{w \in \Sigma^* \mid w \text{ does not contain } www \text{ as a substring}\}$. Write a regular expression for L.

Case 2: w has the substring b. What does this look like?

Our first attempt: $\mathbf{b}^*(\mathbf{ab})^*(\mathbf{ba})^*$

- Cannot handle multiple a's
- Can use b^* to generate bbb
- *abba* is valid, but not what we intended

Idea: Prevent b's from being next to each other!

Our second attempt: $\mathbf{b}(\mathbf{aba})^*$

Avoid the Triple B

Let $\Sigma = \{a, b\}$ and $L = \{w \in \Sigma^* \mid w \text{ does not contain } www \text{ as a substring}\}$. Write a regular expression for L.

Case 2: w has the substring b. What does this look like?

Our second attempt: $\mathbf{b}(\mathbf{aba})^*$

- *baaa* is still a problem
- Cannot end with b e.g. *babab*

Let's just add them.

Our third attempt: $\mathbf{b}(\mathbf{aba} \cup \mathbf{a} \cup \mathbf{ab})^*$

Looks good... except we don't need to start with a b!

Our fourth attempt: $(\mathbf{b} \cup \mathbf{a})(\mathbf{aba} \cup \mathbf{a} \cup \mathbf{ab})^*$

Avoid the Triple B

Let $\Sigma = \{a, b\}$ and $L = \{w \in \Sigma^* \mid w \text{ does not contain } www \text{ as a substring}\}$. Write a regular expression for L.

Case 1: $\mathbf{a}^*$

Case 2: $(\mathbf{b} \cup \mathbf{a})(\mathbf{aba} \cup \mathbf{a} \cup \mathbf{ab})^*$

Now we need to handle Case 3.

... Well, we'll let you do that. We can proceed in a similar fashion. But there is a problem.

Uh oh.

What if w has b and bb as substrings?

We didn't cover this case!!

Avoid the Triple B

Let $\Sigma = \{a, b\}$ and $L = \{w \in \Sigma^* \mid w \text{ does not contain } www \text{ as a substring}\}$. Write a regular expression for L.

Don't panic! We're not in trouble.

An Alternate Perspective

We can actually divide our string (Case 2) into three parts.

- $b \cup a$ (start)
- $aba \cup a \cup ab$ (middle)
- ab (end)

Note: Case 2 did not explicitly handle ab (end), because it was already in $aba \cup a \cup ab$.

We're actually really close! With a few adjustments to Case 2, we (i.e. you, at home, as practice) can obtain the correct answer.

Avoid the Triple B

From the get-go, we could have started from "An Alternative Perspective" i.e. dividing our string into 3 parts.

Many of you may have jumped to that intuition! And that is a perfectly acceptable approach. In fact, on an exam, wouldn't that be ideal?

Unfortunately, in our experience, the correct intuition for a regex, DFA, or NFA does not always immediately occur to us.

If this is the case, then is beneficial to break down the problem in any way you can, create examples, and iteratively build your regex/DFA/NFA ("guess-and-check").

Remark

What we just saw is actually how we (your CAs) solved this problem. Notice that mistakes were made! Generally, when we do math, the solution often comes to us in bits and pieces.

Even Sums

Tag(s): CFGs

Even Sums

Our problem:

- $\Sigma = \{3, 8, +, (\,,\,)\,\}$
- $L = \{w \in \Sigma^* \mid w$ is a syntactically correct mathematical expression for an even number$\}$

Examples	Not Examples
8	ε
38	3
$8 + 8$	$8 + 8 + 3$
$33 + 38 + 83 + 88$	33388833
(8)	$(((8 + 3$
$(8 + 3 + 3 + (3 + 3))$	$++3$
((((88333388))))	8(3)

Even Sums

Our problem:

- $\Sigma = \{3, 8, +, (\,,\,)\}$

Hmmm..

This problem is complicated. How can we make it easier for ourselves? What sticks out to you?

This is not a rhetorical question. If you're watching a recording, pause the video. If you're with us, live, say anything that comes to mind!

$8 + 8$	$8 + 8 + 3$
$33 + 38 + 83 + 88$	33388833
(8)	$(((8 + 3$
$(8 + 3 + 3 + (3 + 3))$	$++3$
$((((88333388))))$	$8(3)$

Even Sums

Examples	Not Examples
8	ε
38	3
$8 + 8$	$8 + 8 + 3$
33 + 38 + 83 + 88	33388833
(8)	(((8 + 3
$(8 + 3 + 3 + (3 + 3))$	++3
((((88333388))))	8(3)

You may have noticed:

- Odd numbers can be split
- A 3 does not mean a number is odd!
- Parentheses are recursive

Also, addition is the only valid operation. The problem specifically states this.

Even Sums

Examples	Not Examples
8	ε
38	3
8 + 8	8 + 8 + 3
33 + 38 + 83 + 88	33388833
(8)	(((8 + 3
(8 + 3 + 3 + (3 + 3))	++3
((((88333388))))	8(3)

Let's think about a) **base cases** and b) **even/odd numbers**

We would say that this is the essential "insight" to solve this problem.

Even Sums

Rule	Why?
$A \to ...$	Our "base case"
$B \to ...$	Even numbers
$C \to ...$	Odd numbers

We love base cases! So let's handle those first.

Even Sums

Practice, practice, practice

If you're watching a recording, really, pause this video and try to fill out A on your own. This is one of the few times where we're rewarded for not watching live.

As an example, here's how we could handle odd numbers: $A \to B + B$

Can you think of more rules?

Even Sums

Rule	Why?
$A \to \ldots$	Our "base case"
$B \to \ldots$	Even numbers
$C \to \ldots$	Odd numbers

We love base cases! So let's handle those first.

- We need parentheses! $\mathbf{A} \to (\mathbf{A})$
- We need to add even numbers! $\mathbf{A} \to \mathbf{B} + \mathbf{A}$
- We need to add odd numbers! $\mathbf{A} \to \mathbf{C} + \mathbf{C} + \mathbf{A}$

Why did we end each with rule with A?

Even Sums

Rule	Why?
$A \to (A) \mid A \to B + A \mid A \to C + C + A$	Our "base case"
$B \to \ldots$	Even numbers
$C \to \ldots$	Odd numbers

How can we create even numbers?

- B ends in 8 (but can have other stuff, too)

That's pretty much it, actually. Let's create a rule.

- We need to end in 8! $\mathbf{B} \to \mathbf{8}$
- We need other stuff! $\mathbf{B} \to \mathbf{D8}$
  Let D be other stuff.
- After 8, the number doesn't matter. $\mathbf{D} \to \varepsilon \mid \mathbf{3D} \mid \mathbf{8D}$

Even Sums

Rule	Why?
$A \to (A) \mid A \to B + A \mid A \to C + C + A$	Our "base case"
$B \to D8$	Even numbers
$D \to \varepsilon \mid 3D \mid 8D$	Terminal for evens
$C \to \ldots$	Odd numbers

How can we create odd numbers?

- C ends in 3! $\mathbf{C} \to \mathbf{E3}$
  We'll need to do something similar to D with E
- An odd number and an even number! $\mathbf{C} \to \mathbf{C} + \mathbf{B}$

Even Sums

Rule	Why?
$A \to (A) \mid A \to B + A \mid A \to C + C + A$	Our "base case"
$B \to D8$	Even numbers
$D \to \varepsilon \mid 3D \mid 8D$	Terminal for evens
$C \to E3 \mid C + B$	Odd numbers
$E \to \varepsilon \mid 3E \mid 8E$	Terminal for odds

Me: This has to be correct! I totally do not need to check my work, even if I'll have accidentally misled dozens of people on record!

Narrator: This was not correct.

Even Sums

Rule	Why?
$A \to (A) \mid A \to B + A \mid A \to C + C + A$	Our "base case"
$B \to D8$	Even numbers

Remark

It's your turn again!

To check our work efficiently, it's easiest to return to examples. Fortunately, the problem gave us plenty earlier.

M

a

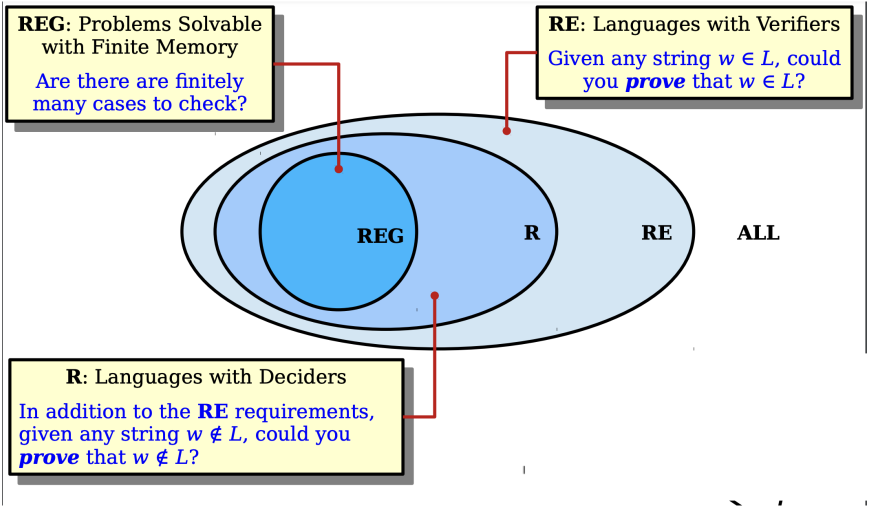
Narrator: This was not correct.

Lava Diagrams

Tag(s): Lava Diagram

Lava Diagrams

Let's do a lava diagram question! First, let's recall the lava diagram guide.



Let's first look at an example where our language is comprised of encodings of Turing machines. Lava diagram 6 part 6:

$\{\langle M, w\rangle \mid M \text{ is a TM, } w \text{ is a string, } M \text{ halts on } w \text{ in at most } |w|^{137} \text{ steps}\}$

Let's first look at an example where our language is comprised of encodings of Turing machines. Lava diagram 6 part 6:

$\{\langle M, w\rangle \mid M$ is a TM, w is a string, M halts on w in at most $|w|^{137}$ steps$\}$

1. Q: Is it recognizable? (Given $\langle M, w\rangle \in L$, can we prove that $\langle M, w\rangle \in L$)

Let's first look at an example where our language is comprised of encodings of Turing machines. Lava diagram 6 part 6:

$\{\langle M, w\rangle \mid M$ is a TM, w is a string, M halts on w in at most $|w|^{137}$ steps$\}$

1. Q: Is it recognizable? (Given $\langle M, w\rangle \in L$, can we prove that $\langle M, w\rangle \in L$)
   A: Yes, we can run the Turing machine M for $|w|^{137}$ steps on w to confirm it halts.

Let's first look at an example where our language is comprised of encodings of Turing machines. Lava diagram 6 part 6:

$\{\langle M, w\rangle \mid M$ is a TM, w is a string, M halts on w in at most $|w|^{137}$ steps$\}$

1. Q: Is it recognizable? (Given $\langle M, w\rangle \in L$, can we prove that $\langle M, w\rangle \in L$)
   A: Yes, we can run the Turing machine M for $|w|^{137}$ steps on w to confirm it halts.
2. Q: Is it decidable? (Given $\langle M, w\rangle \notin L$, can we prove that $\langle M, w\rangle \notin L$)

Let's first look at an example where our language is comprised of encodings of Turing machines. Lava diagram 6 part 6:

$\{\langle M, w\rangle \mid M$ is a TM, w is a string, M halts on w in at most $|w|^{137}$ steps$\}$

1. Q: Is it recognizable? (Given $\langle M, w\rangle \in L$, can we prove that $\langle M, w\rangle \in L$)
   A: Yes, we can run the Turing machine M for $|w|^{137}$ steps on w to confirm it halts.
2. Q: Is it decidable? (Given $\langle M, w\rangle \notin L$, can we prove that $\langle M, w\rangle \notin L$)
   A: Yes, we can run M on w for $|w|^{137}$ steps and if it does not halt, it is not in L.

Let's first look at an example where our language is comprised of encodings of Turing machines. Lava diagram 6 part 6:

$$\{\langle M, w\rangle \mid M \text{ is a TM, } w \text{ is a string, } M \text{ halts on } w \text{ in at most } |w|^{137} \text{ steps}\}$$

1. Q: Is it recognizable? (Given $\langle M, w\rangle \in L$, can we prove that $\langle M, w\rangle \in L$)
   A: Yes, we can run the Turing machine M for $|w|^{137}$ steps on w to confirm it halts.
2. Q: Is it decidable? (Given $\langle M, w\rangle \notin L$, can we prove that $\langle M, w\rangle \notin L$)
   A: Yes, we can run M on w for $|w|^{137}$ steps and if it does not halt, it is not in L.
3. Q: Is it regular? Are there finitely many cases to check?

Let's first look at an example where our language is comprised of encodings of Turing machines. Lava diagram 6 part 6:

$\{\langle M, w\rangle \mid M$ is a TM, w is a string, M halts on w in at most $|w|^{137}$ steps$\}$

1. Q: Is it recognizable? (Given $\langle M, w\rangle \in L$, can we prove that $\langle M, w\rangle \in L$)
   A: Yes, we can run the Turing machine M for $|w|^{137}$ steps on w to confirm it halts.
2. Q: Is it decidable? (Given $\langle M, w\rangle \notin L$, can we prove that $\langle M, w\rangle \notin L$)
   A: Yes, we can run M on w for $|w|^{137}$ steps and if it does not halt, it is not in L.
3. Q: Is it regular? Are there finitely many cases to check?
   A: No, since Turing machines are not limited to finite memory.

Let's do one more example with a language of Turing machine encodings!
Lava diagram 2, part 5:

$$L = \{\langle M, w \rangle \mid M \text{ is a TM and } M \text{ does not accept } w\}$$

Let's do one more example with a language of Turing machine encodings!
Lava diagram 2, part 5:

$$L = \{\langle M, w\rangle \mid M \text{ is a TM and } M \text{ does not accept } w\}$$

1. Q: Is it recognizable? (Given $\langle M, w\rangle \in L$, can we prove that $\langle M, w\rangle \in L$)

Let's do one more example with a language of Turing machine encodings! Lava diagram 2, part 5:

$$L = \{\langle M, w\rangle \mid M \text{ is a TM and } M \text{ does not accept } w\}$$

1. Q: Is it recognizable? (Given $\langle M, w\rangle \in L$, can we prove that $\langle M, w\rangle \in L$)
   A: No, if we run M on w, it may loop on w. There is no way to rule out the chance that M may accept w eventually.

Takeaways:

- The only way to learn information about what a Turing machine will do on a certain input is to run it on that input!

Takeaways:

- The only way to learn information about what a Turing machine will do on a certain input is to run it on that input!
- You cannot check infinitely many strings to prove something

Takeaways:

- The only way to learn information about what a Turing machine will do on a certain input is to run it on that input!
- You cannot check infinitely many strings to prove something
- If there is a chance a Turing machine could loop on a certain input, there is no way to prove it does (it might halt at some far later time, and no matter how long you wait you can't rule this out).

Now let's do an example that does not involve Turing machines. Lava diagram 6 part 4:

$$L = \{1^m + 1^n = 1^{m+n} | m, n \in \mathbb{N} \text{ and } m \leq 10^{137}\}$$

1. Q: Is it recognizable? (Given $w \in L$, can we prove that $w \in L$?)

Now let's do an example that does not involve Turing machines. Lava diagram 6 part 4:

$$L = \{1^m + 1^n = 1^{m+n} | m, n \in \mathbb{N} \text{ and } m \leq 10^{137}\}$$

1. Q: Is it recognizable? (Given $w \in L$, can we prove that $w \in L$?)
   A: Yes, we can count the number of 1s appearing on the left-hand side and check if it is the same number on the right-hand side. We can also count and confirm that $m \leq 10^{137}$

Now let's do an example that does not involve Turing machines. Lava diagram 6 part 4:

$$L = \{1^m + 1^n = 1^{m+n} | m, n \in \mathbb{N} \text{ and } m \leq 10^{137}\}$$

1. Q: Is it recognizable? (Given $w \in L$, can we prove that $w \in L$?)
   A: Yes, we can count the number of 1s appearing on the left-hand side and check if it is the same number on the right-hand side. We can also count and confirm that $m \leq 10^{137}$
2. Q: Is it decidable? (Given $w \notin L$, can we prove that $w \notin L$?)

Now let's do an example that does not involve Turing machines. Lava diagram 6 part 4:

$$L = \{1^m + 1^n = 1^{m+n} | m, n \in \mathbb{N} \text{ and } m \leq 10^{137}\}$$

1. Q: Is it recognizable? (Given $w \in L$, can we prove that $w \in L$?)
   A: Yes, we can count the number of 1s appearing on the left-hand side and check if it is the same number on the right-hand side. We can also count and confirm that $m \leq 10^{137}$
2. Q: Is it decidable? (Given $w \notin L$, can we prove that $w \notin L$?)
   A: Yes, we can count and see that either $m > 10^{137}$ or that the number of 1s on the left hand side is different than the number on the right.

Now let's do an example that does not involve Turing machines. Lava diagram 6 part 4:

$$L = \{1^m + 1^n = 1^{m+n} | m, n \in \mathbb{N} \text{ and } m \leq 10^{137}\}$$

1. Q: Is it recognizable? (Given $w \in L$, can we prove that $w \in L$?)
   A: Yes, we can count the number of 1s appearing on the left-hand side and check if it is the same number on the right-hand side. We can also count and confirm that $m \leq 10^{137}$
2. Q: Is it decidable? (Given $w \notin L$, can we prove that $w \notin L$?)
   A: Yes, we can count and see that either $m > 10^{137}$ or that the number of 1s on the left hand side is different than the number on the right.
3. Q: Is it regular?

Now let's do an example that does not involve Turing machines. Lava diagram 6 part 4:

$$L = \{1^m + 1^n = 1^{m+n} | m, n \in \mathbb{N} \text{ and } m \leq 10^{137}\}$$

1. Q: Is it recognizable? (Given $w \in L$, can we prove that $w \in L$?)
   A: Yes, we can count the number of 1s appearing on the left-hand side and check if it is the same number on the right-hand side. We can also count and confirm that $m \leq 10^{137}$
2. Q: Is it decidable? (Given $w \notin L$, can we prove that $w \notin L$?)
   A: Yes, we can count and see that either $m > 10^{137}$ or that the number of 1s on the left hand side is different than the number on the right.
3. Q: Is it regular?
   A: No, for example $S = \{1 + 1^n | n \in \mathbb{N}\}$ is an infinite distinguishing set. Choosing $1 + 1^n$ and $1 + 1^m$ from S, with $w == 1^{n+1}$ we see that $1 + 1^m = 1^{n+1}$ is not in L while $1 + 1^n = 1^{n+1}$ is in L.

Closing thoughts