Algebra for N00bs

By Max

Table of Contents

1. First Order Logic // you already know this
2. Words and Languages // aka easy algebra
3. Finite Deterministic Automata
4. Regular Expressions & Languages
5. Monoids
6. Metric Spaces // the secret ingredient to ML

// Plus a short note at the end about how Pin denotes functions …

1st Order Logic

Variables: x_0, x_1, x_2, …

max-is-smart,max-is-dumb

1st Order Logic

Variables: x_0, x_1, x_2, …	There are countably many of these.
max-is-smart,max-is-dumb	

1st Order Logic

Variables: x_0, x_1, x_2, …	There are countably many of these.
max-is-smart,max-is-dumb	

Operators: ~, ^, V, =	Input: Properties Output: Properties
max-is-dumb v max-is-smart	

1st Order Logic

Variables: x_0, x_1, x_2, …

max-is-smart,max-is-dumb

There are countably many of these.

Operators: ~, ^, V, =

max-is-dumb v max-is-smart

Input: Properties
Output: Properties

Parentheses: (…)

1st Order Logic

Variables: x_0, x_1, x_2, …	There are countably many of these.
max-is-smart,max-is-dumb	
Operators: ~, ^, V, =	Input: Properties Output: Properties
max-is-dumb v max-is-smart	
Parentheses: (…)	
Predicates: $P_i(y_0, …, y_n)$	Input: Variables Output: Properties
are-siblings(Max, Sam)	

1st Order Logic

Variables: x_0, x_1, x_2, …	There are countably many of these.
max-is-smart,max-is-dumb	

Operators: ~, ^, V, =	Input: Properties Output: Properties
max-is-dumb v max-is-smart	

Parentheses: (…)

Quantifier: ∃	∀xP(x) means ~∃x~P(x)
∃ x jacob-thinks(x)	

Predicates: $P_i(y_0,…,y_n)$	Input: Variables Output: Properties
are-siblings(Max, Sam)	

1st Order Logic

Variables: x_0, x_1, x_2, …

max-is-smart,max-is-dumb

There are countably many of these.

Operators: ~, ^, V, =

max-is-dumb v max-is-smart

Input: Properties
Output: Properties

Parentheses: (…)

Quantifier: ∃

∀xP(x) means ~∃x~P(x)

∃ x jacob-thinks(x)

Predicates: $P_i(y_0, \ldots, y_n)$

are-siblings(Max, Sam)

Input: Variables
Output: Properties

Equality: “=”

This ingredient is different in HoTT

1st Order Logic

Variables: x_0, x_1, x_2, …

max-is-smart,max-is-dumb

There are countably many of these.

Operators: ~, ^, V, =

max-is-dumb v max-is-smart

Input: Properties
Output: Properties

Parentheses: (…)

Quantifier: ∃

∀xP(x) means ~∃x~P(x)

∃ x jacob-thinks(x)

Predicates: $P_i(y_0, \dots, y_n)$

are-siblings(Max, Sam)

Input: Variables
Output: Properties

Functions: $f_i(y_0, \dots, y_n) \mapsto y_k$

father-of(Max) = Frank

Input: Variables, Output: Variables

Equality: "="

This ingredient is different in HoTT

1st Order Logic

Variables: x_0, x_1, x_2, …

max-is-smart,max-is-dumb

There are countably many of these.

Operators: ~, ^, V, =

max-is-dumb v max-is-smart

Input: Properties
Output: Properties

Parentheses: (…)

Quantifier: ∃

∀xP(x) means ~∃x~P(x)

∃ x jacob-thinks(x)

Predicates: $P_i(y_0, …, y_n)$

are-siblings(Max, Sam)

Input: Variables
Output: Properties

Theory: a finite set Γ of variables, assumed to be *true*. ⊢ is entailment.

… more on this in a moment.

Functions: $f_i(y_0, …, y_n) \mapsto y_k$

father-of(Max) = Frank

Input: Variables, Output: Variables

Equality: “=”

This ingredient is different in HoTT

1st Order Logic <<Surprise Gödel>>

Theory: a finite set Γ of predicates, assumed to be *true*.

```
    // For example, Peano Arithmetic:
    /* { nat(0),
         ∀x(x=x),
         ∀x,y(x=y ⇒ y=x),
         ∀x,y,z (x=y ^ y=z) ⇒ x=z,
         ∀x,y (nat(x) ^ x=y ) ⇒ nat(y),
         ∀x nat(x) ⇒ nat(S(x)),
         ∀x,y (x = y) ⇔ (S(x) = S(y)),
         ∀x ~(S(x)=0) } */
```

1st Order Logic <<Surprise Gödel>>

Theory: a finite set Γ of predicates, assumed to be *true*.

Syntactic Entailment: Γ ⊢ φ.
    // Does there exist a proof, which assumes only Γ, and concludes φ?

1st Order Logic <<Surprise Gödel>>

Theory: a finite set Γ of predicates, assumed to be *true*.

Syntactic Entailment: $\Gamma \vdash \varphi$.

- If x in Γ then Γ⊢x // Is it an axiom?

1st Order Logic <<Surprise Gödel>>

Theory: a finite set Γ of predicates, assumed to be *true*.

Syntactic Entailment: $\Gamma \vdash \varphi$.

- If x in Γ then $\Gamma \vdash x$ // Is it an axiom?
  - If x is in Γ it's called a non-logical axiom.

1st Order Logic <<Surprise Gödel>>

Theory: a finite set Γ of predicates, assumed to be *true*.

Syntactic Entailment: $\Gamma \vdash \varphi$.

- If x in Γ then $\Gamma \vdash x$ // Is it an axiom?
  - If x is in Γ it's called a non-logical axiom.
  - If $\{\ \} \vdash x$ then x is a “tautology”, or a logical axiom.

1st Order Logic <<Surprise Gödel>>

Theory: a finite set Γ of predicates, assumed to be *true*.

Syntactic Entailment: $\Gamma \vdash \varphi$.

- If x in Γ then $\Gamma \vdash x$ // Is it an axiom?
  - If x is in Γ it's called a non-logical axiom.
  - If $\{\ \} \vdash x$ then x is a "tautology", or a logical axiom.
    - I'm not getting into logical axioms right now because I don't want to talk about modus ponens …

1st Order Logic <<Surprise Gödel>>

Theory: a finite set Γ of predicates, assumed to be *true*.

Syntactic Entailment: $\Gamma \vdash \varphi$.

- If x in Γ then $\Gamma \vdash x$
- If $\Gamma \vdash x$ and $\Gamma \vdash y$ then $\Gamma \vdash x \wedge y$ // Can we prove this and that?

1st Order Logic <<Surprise Gödel>>

Theory: a finite set Γ of predicates, assumed to be *true*.

Syntactic Entailment: $\Gamma \vdash \varphi$.

- If x in Γ then $\Gamma \vdash x$
- If $\Gamma \vdash x$ and $\Gamma \vdash y$ then $\Gamma \vdash x \wedge y$
- If $\Gamma \vdash x$ or $\Gamma \vdash y$ then $\Gamma \vdash x \vee y$ // Can we prove this or that?

1st Order Logic <<Surprise Gödel>>

Theory: a finite set Γ of predicates, assumed to be *true*.

Syntactic Entailment: $\Gamma \vdash \varphi$.

- If x in Γ then $\Gamma \vdash x$
- If $\Gamma \vdash x$ and $\Gamma \vdash y$ then $\Gamma \vdash x \wedge y$
- If $\Gamma \vdash x$ or $\Gamma \vdash y$ then $\Gamma \vdash x \vee y$
- If $\Gamma, y \vdash z$ and $\Gamma, y \vdash \sim z$ then $\Gamma \vdash \sim y$ // Proof by way of contradiction

1st Order Logic <<Surprise Gödel>>

Theory: a finite set Γ of predicates, assumed to be *true*.

Syntactic Entailment: $\Gamma \vdash \varphi$.

- If x in Γ then $\Gamma \vdash x$
- If $\Gamma \vdash x$ and $\Gamma \vdash y$ then $\Gamma \vdash x \wedge y$
- If $\Gamma \vdash x$ or $\Gamma \vdash y$ then $\Gamma \vdash x \vee y$
- If $\Gamma, y \vdash z$ and $\Gamma, y \vdash \sim z$ then $\Gamma \vdash \sim y$

Semantic Entailment: $\Gamma \vDash \varphi$ if, for every theory Π containing Γ, $\Pi \nvdash \sim\varphi$.
// The property is compatible with all similar theories.

1st Order Logic <<Surprise Gödel>>

Theory: a finite set Γ of predicates, assumed to be *true*.

Syntactic Entailment: $\Gamma \vdash \varphi$.

- If x in Γ then $\Gamma \vdash x$
- If $\Gamma \vdash x$ and $\Gamma \vdash y$ then $\Gamma \vdash x \wedge y$
- If $\Gamma \vdash x$ or $\Gamma \vdash y$ then $\Gamma \vdash x \vee y$
- If $\Gamma, y \vdash z$ and $\Gamma, y \vdash \sim z$ then $\Gamma \vdash \sim y$

Semantic Entailment: $\Gamma \vDash \varphi$ if, for every theory Π containing Γ, $\Pi \nvdash \sim\varphi$.

Consistency: $\Gamma \vdash \varphi$ if and only if $\Gamma \nvdash \sim\varphi$. // Don't prove a contradiction.

1st Order Logic <<Surprise Gödel>>

Theory: a finite set Γ of predicates, assumed to be *true*.

Syntactic Entailment: $\Gamma \vdash \varphi$.

- If x in Γ then $\Gamma \vdash x$
- If $\Gamma \vdash x$ and $\Gamma \vdash y$ then $\Gamma \vdash x \wedge y$
- If $\Gamma \vdash x$ or $\Gamma \vdash y$ then $\Gamma \vdash x \vee y$
- If $\Gamma, y \vdash z$ and $\Gamma, y \vdash \sim z$ then $\Gamma \vdash \sim y$

Semantic Entailment: $\Gamma \vDash \varphi$ if, for every theory Π containing Γ, $\Pi \nvdash \sim\varphi$.

Consistency: $\Gamma \vdash \varphi$ if and only if $\Gamma \nvdash \sim\varphi$.

Soundness: If $\Gamma \vdash \varphi$ then $\Gamma \vDash \varphi$. // Don't prove anything that can be disproven by a compatible theory.

1st Order Logic <<Surprise Gödel>>

Theory: a finite set Γ of predicates, assumed to be *true*.

Syntactic Entailment: $\Gamma \vdash \varphi$.

- If x in Γ then $\Gamma \vdash x$
- If $\Gamma \vdash x$ and $\Gamma \vdash y$ then $\Gamma \vdash x \wedge y$
- If $\Gamma \vdash x$ or $\Gamma \vdash y$ then $\Gamma \vdash x \vee y$
- If $\Gamma, y \vdash z$ and $\Gamma, y \vdash \sim z$ then $\Gamma \vdash \sim y$

Semantic Entailment: $\Gamma \vDash \varphi$ if, for every theory Π containing Γ, $\Pi \nvdash \sim\varphi$.

Consistency: $\Gamma \vdash \varphi$ if and only if $\Gamma \nvdash \sim\varphi$.

Soundness: If $\Gamma \vdash \varphi$ then $\Gamma \vDash \varphi$.

Completeness: If $\Gamma \vDash \varphi$ then $\Gamma \vdash \varphi$. // All theories w/ these axioms allow φ

1st Order Logic <<Surprise Gödel>>

Theory: a finite set Γ of predicates, assumed to be *true*.

Syntactic Entailment: $\Gamma \vdash \varphi$.

- If x in Γ then $\Gamma \vdash x$
- If $\Gamma \vdash x$ and $\Gamma \vdash y$ then $\Gamma \vdash x \wedge y$
- If $\Gamma \vdash x$ or $\Gamma \vdash y$ then $\Gamma \vdash x \vee y$
- If $\Gamma, y \vdash z$ and $\Gamma, y \vdash \sim z$ then $\Gamma \vdash \sim y$

Semantic Entailment: $\Gamma \vDash \varphi$ if, for every theory Π containing Γ, $\Pi \nvdash \sim\varphi$.

Consistency: $\Gamma \vdash \varphi$ if and only if $\Gamma \nvdash \sim\varphi$.

Soundness: If $\Gamma \vdash \varphi$ then $\Gamma \vDash \varphi$.

Completeness: If $\Gamma \vDash \varphi$ then $\Gamma \vdash \varphi$.

Thm 1: You can't have both consistency & completeness.

1st Order Logic <<Surprise Gödel>>

Theory: a finite set Γ of predicates, assumed to be *true*.

Syntactic Entailment: $\Gamma \vdash \varphi$.

- If x in Γ then $\Gamma \vdash x$
- If $\Gamma \vdash x$ and $\Gamma \vdash y$ then $\Gamma \vdash x \wedge y$
- If $\Gamma \vdash x$ or $\Gamma \vdash y$ then $\Gamma \vdash x \vee y$
- If $\Gamma, y \vdash z$ and $\Gamma, y \vdash \sim z$ then $\Gamma \vdash \sim y$

Semantic Entailment: $\Gamma \vDash \varphi$ if, for every theory Π containing Γ, $\Pi \nvdash \sim\varphi$.

Consistency: $\Gamma \vdash \varphi$ if and only if $\Gamma \nvdash \sim\varphi$.

Soundness: If $\Gamma \vdash \varphi$ then $\Gamma \vDash \varphi$.

Completeness: If $\Gamma \vDash \varphi$ then $\Gamma \vdash \varphi$.

Thm 1: You can't have both consistency & completeness.

Thm 2: Γ can't prove its own consistency.

Table of Contents

1. ~~First Order Logic // you already know this~~
2. Words and Languages // aka easy algebra
3. Finite Deterministic Automata
4. Regular Expressions & Languages
5. Monoids
6. Metric Spaces // the secret ingredient to ML

// Plus a short note at the end about how Pin denotes functions …

Words and Languages

Alphabet: a non-empty finite set.

A = { 0, 1, 2, …, 9 }

Words and Languages

Alphabet: a non-empty finite set.	Word: a finite sequence from the alphabet A.
A = { 0, 1, 2, …, 9 }	69420

Words and Languages

Alphabet: a non-empty finite set.

A = { 0, 1, 2, …, 9 }

Word: a finite sequence from the alphabet A.

69420

Free Monoid: The set of all words over A.

A* = ℕ (with leading 0s)

Words and Languages

Alphabet: a non-empty finite set.

A = { 0, 1, 2, …, 9 }

Word: a finite sequence from the alphabet A.

69420

Alternatively, any element of A*.

1 denotes “”.

Free Monoid: The set of all words over A.

A* = ℕ (with leading 0s)

Words and Languages

Alphabet: a non-empty finite set.

$A = \{ 0, 1, 2, \ldots, 9 \}$

Word: a finite sequence from the alphabet A.

69420

Alternatively, any element of A*.

1 denotes “”.

Free Monoid: The set of all words over A.

$A^* = \mathbb{N}$ (with leading 0s)

Language over A: any subset of A*

Words and Languages

Alphabet: a non-empty finite set.

$A = \{ 0, 1, 2, \ldots, 9 \}$

Word: a finite sequence from the alphabet A.

69420

Alternatively, any element of A*.

1 denotes “”.

Free Monoid: The set of all words over A.

$A^* = \mathbb{N}$ (with leading 0s)

Language over A: any subset of A*

$\mathbb{N}$, positives, even numbers, odd numbers, binary, binary programs that terminate, binary programs that were written by Microsoft engineers, prime numbers, real-world RSA moduli, ternary encodings of Max’s passwords …

Words and Languages

69420

Initial

Words and Languages

69420

Ending

Words and Languages

69420

(Proper) prefix or left factor

Words and Languages

69420

(Proper) postfix or right factor

Words and Languages

69420

(Proper) subword

Words and Languages

$|69420| = 5$

Words and Languages

$$|69420| = 5$$

$$|69420|_6 = 1$$

Words and Languages

$$|69420| = 5$$

$$|69420|_6 = 1$$

$$|69666|_6 = 4$$

Words and Languages

$|69420| = 5$

$|69420|_6 = 1$

$|69666|_6 = 4$

W is called **multilinear** if $|w|_x \leq 1$ for all x in A

Words and Languages

69 * 420 = 69420

* is associative, non-commutative

*'s identity is 1

420^3 = 420420420

Words and Languages

$\{69\} * \{666, 314, 420, 000\} = \{69666, 69314, 69420, 69000\}$

* is associative, non-commutative

*'s identity is $\mathbb{1} = \{ 1 \} = \{ \text{“”} \}$

$\{ 22, 33 \}^2 = \{ 2222, 2233, 3322, 3333 \}$

Words and Languages

$\{6,9\} \setminus \{6,300\} = \{9\}$

Words and Languages

$\{6,9\} \setminus \{6,300\} = \{9\}$

$\{6,9\} \cup \{6,300\} = \{6,9,300\}$

Words and Languages

$\{6,9\} \setminus \{6,300\} = \{9\}$

$\{6,9\} \cup \{6,300\} = \{6,9,300\}$

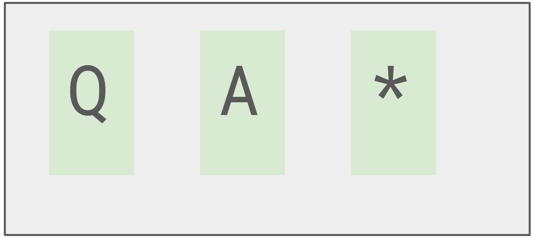
$\{6,9\}^c$ = Everything in A^* except for 6 and 9

Table of Contents

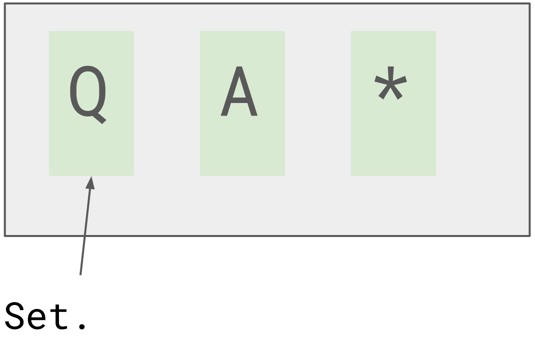
1. ~~First Order Logic // you already know this~~
2. ~~Words and Languages // aka easy algebra~~
3. Finite Deterministic Automata
4. Regular Expressions & Languages
5. Monoids
6. Metric Spaces // the secret ingredient to ML

// Plus a short note at the end about how Pin denotes functions …

Finite Deterministic Automata



Finite Deterministic Automata



Finite Deterministic Automata

Q A *

… how Pin defines things

Set.

S Σ δ

… how CS people usually define things

Finite Deterministic Automata

Q A *

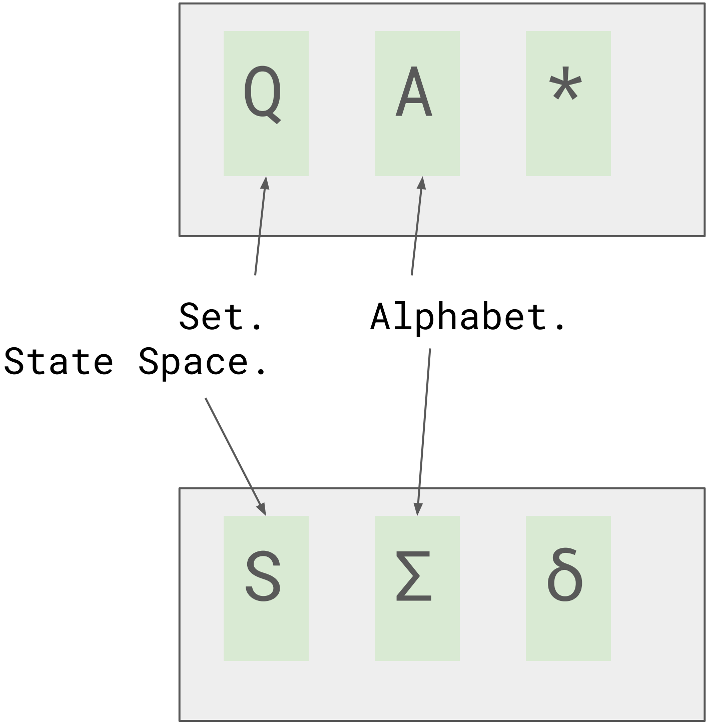
… how Pin defines things

Set.
State Space.

S Σ δ

… how CS people usually define things

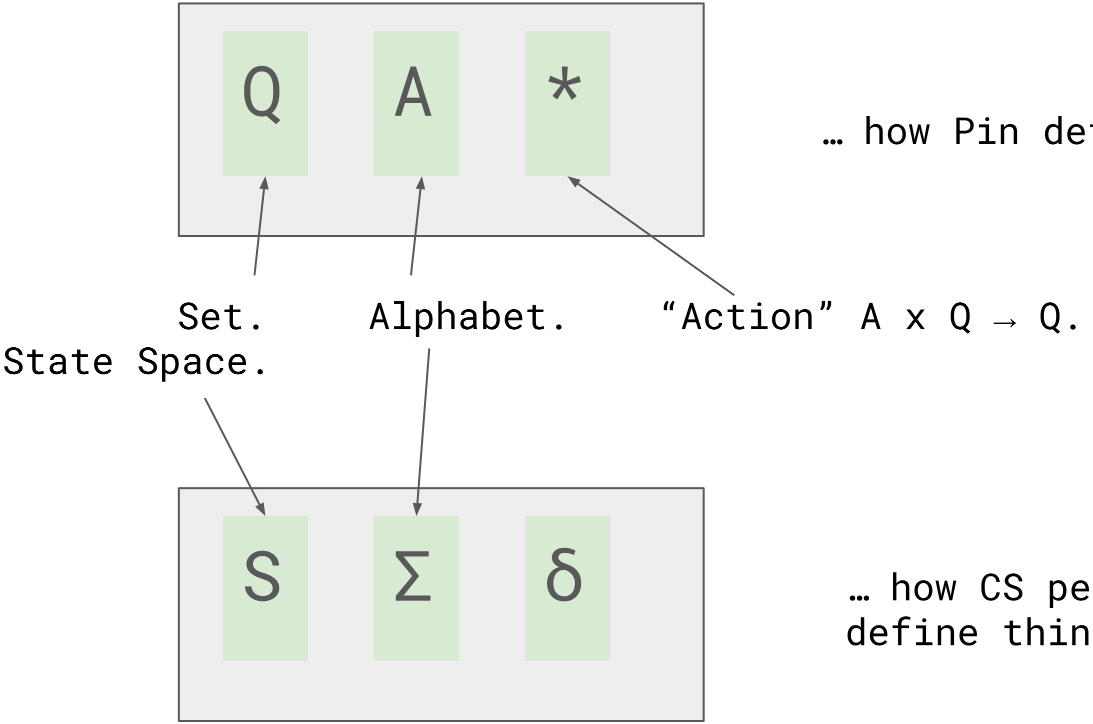
Finite Deterministic Automata



… how Pin defines things

… how CS people usually define things

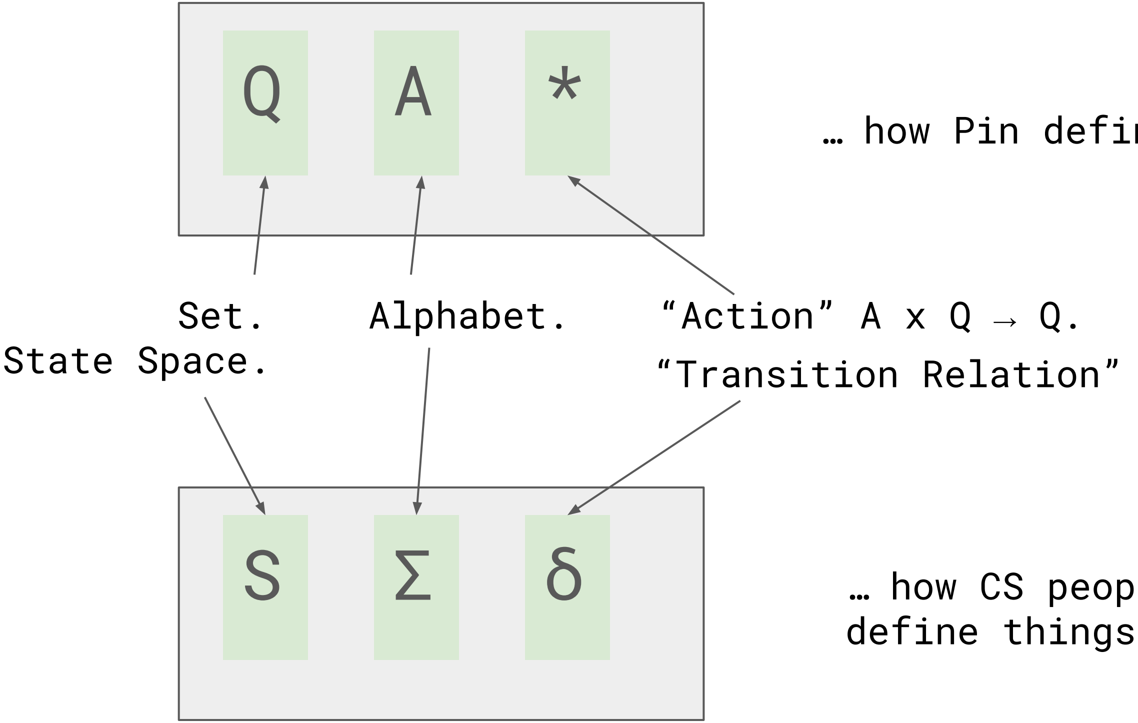
Finite Deterministic Automata



… how Pin defines things

… how CS people usually define things

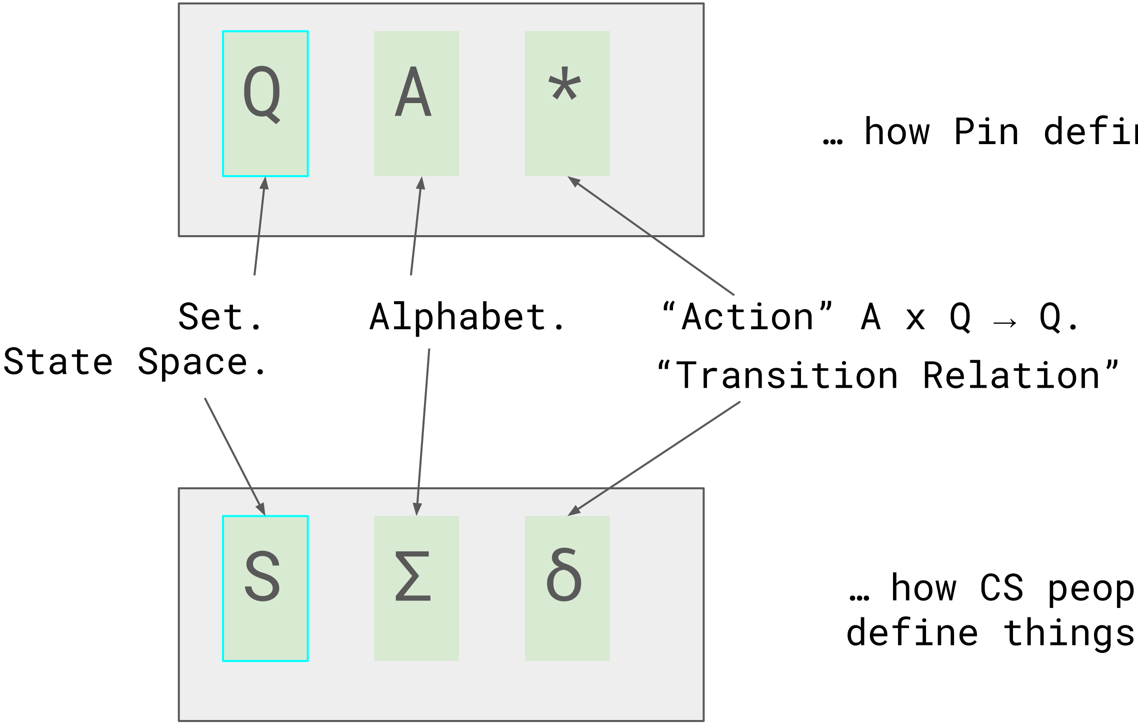
Finite Deterministic Automata



… how Pin defines things

… how CS people usually define things

Finite Deterministic Automata

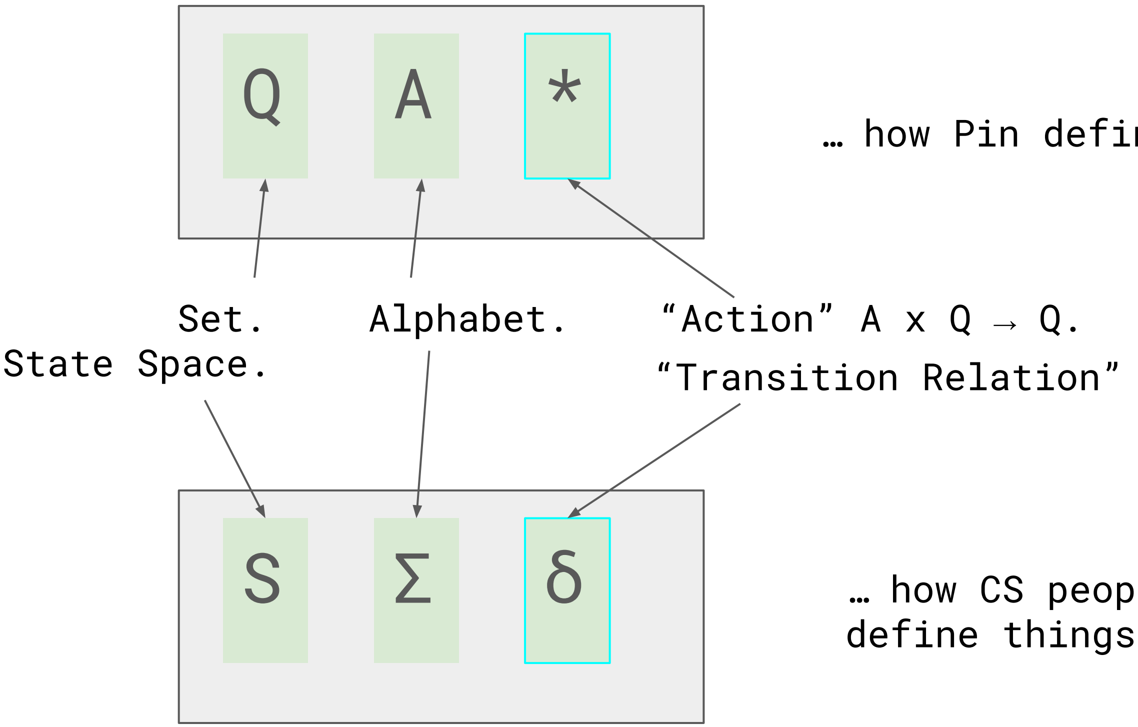


… how Pin defines things

… how CS people usually define things

Finite if Q (aka Σ) is finite.

Finite Deterministic Automata



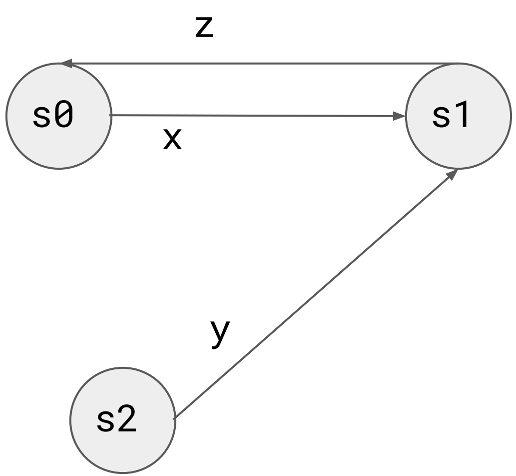
… how Pin defines things

… how CS people usually define things

Finite if Q (aka Σ) is finite.

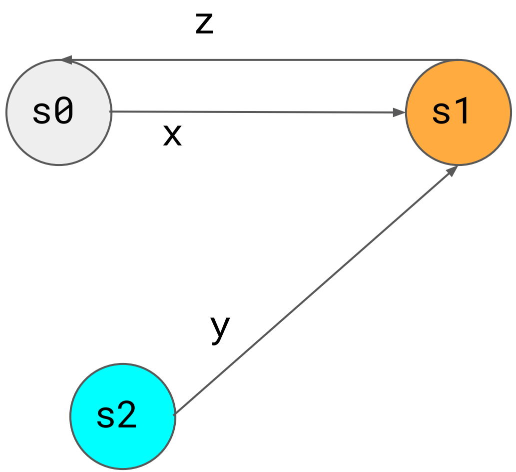
Deterministic if * (aka δ) is a function.

Finite Deterministic Automata



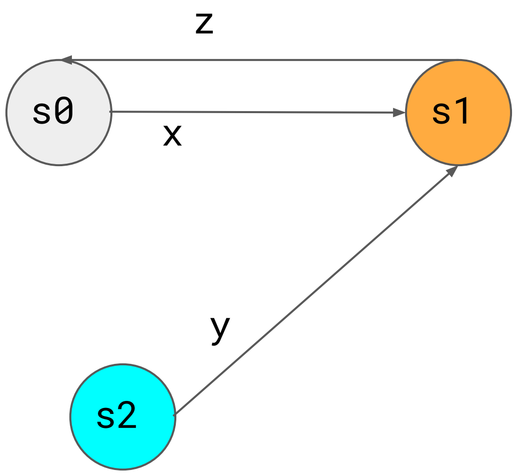

```
Q = { s0, s1, s2 } // aka S
A = { x, y, z }    // aka Σ

s0 * x = s1 // δ(s0,x)=s1
s1 * z = s0 // δ(s1,z)=s0
s2 * y = s1 // δ(s2,y)=s1
```

Finite Deterministic Automata



We can extend our definition to include one or more **initial states** and **final states**.

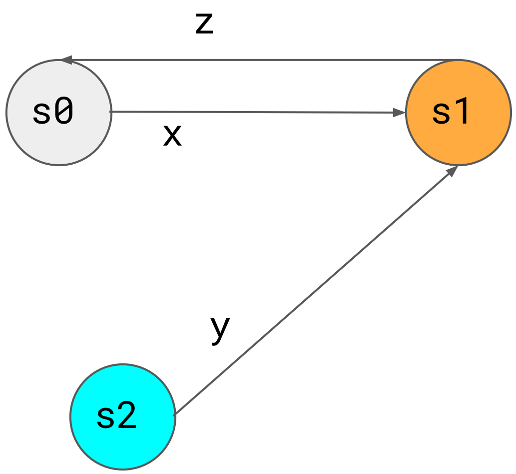
Finite Deterministic Automata



We can extend our definition to include one or more **initial states** and **final states**.

A **path** = a sequence of consecutive transitions.

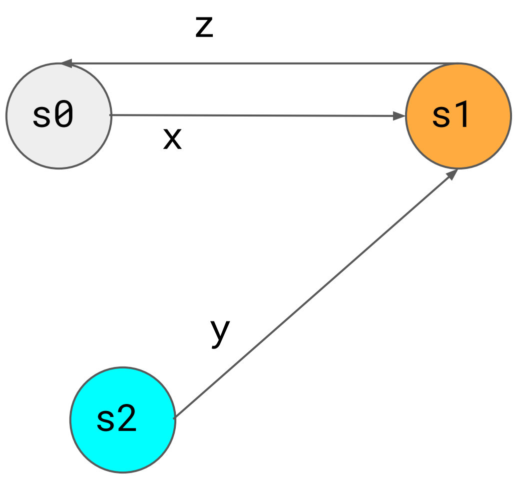
Finite Deterministic Automata



We can extend our definition to include one or more **initial states** and **final states**.

A **path** = a sequence of consecutive transitions.

Finite Deterministic Automata

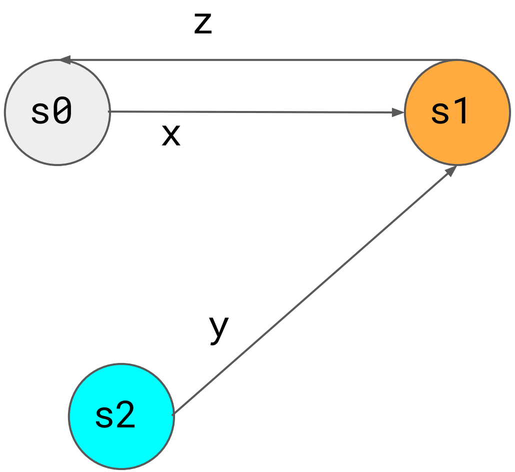


We can extend our definition to include one or more **initial states** and **final states**.

A **path** = a sequence of consecutive transitions.

- "Final" if ends in a final state.

Finite Deterministic Automata

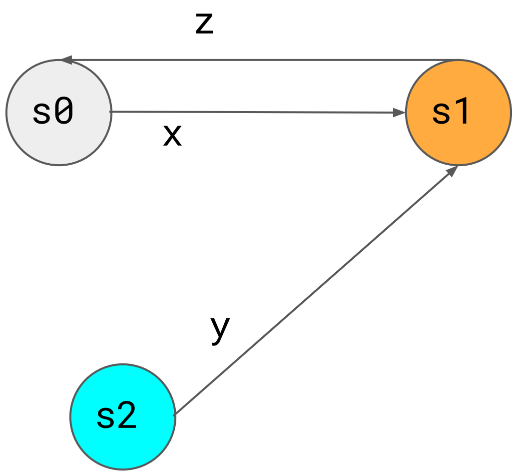


We can extend our definition to include one or more **initial states** and **final states**.

A **path** = a sequence of consecutive transitions.

- “Final” if ends in a final state.
- “Initial” if starts in an initial state.

Finite Deterministic Automata

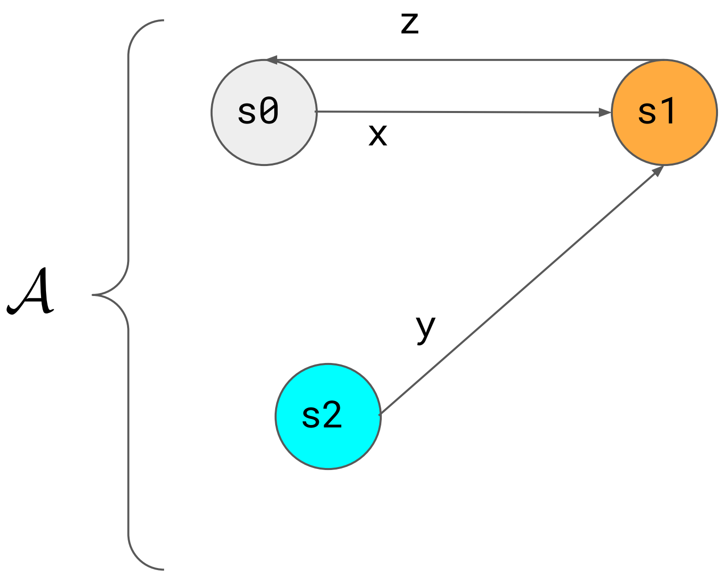


We can extend our definition to include one or more **initial states** and **final states**.

A **path** = a sequence of consecutive transitions.

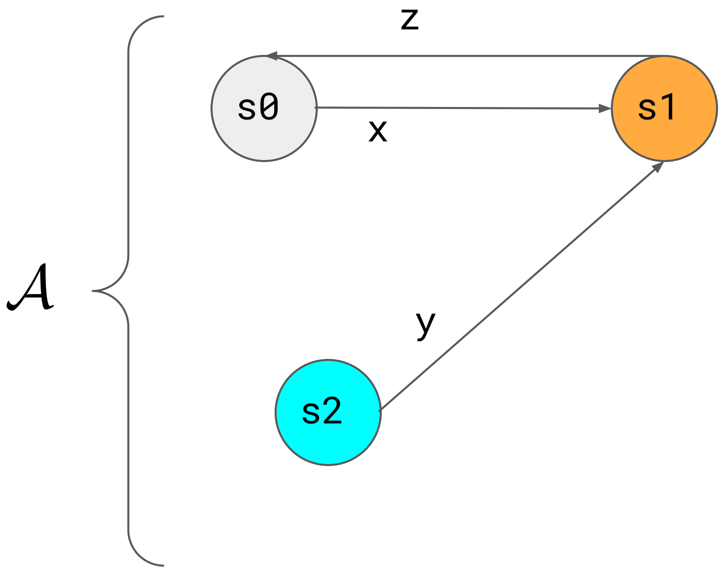
- "Final" if ends in a final state.
- "Initial" if starts in an initial state.
- "Successful" if both Final and Initial.

Finite Deterministic Automata



We can extend our definition to include one or more **initial states** and **final states**.

Finite Deterministic Automata



We can extend our definition to include one or more **initial states** and **final states**.

$\mathscr{A}$ accepts the language L iff there exists initial states I and final states F s.t. L equals all of the successful paths from I to F.

{ y, yzx, yzxzx, yzxzxzx, … }

Table of Contents

1. ~~First Order Logic // you already know this~~
2. ~~Words and Languages // aka easy algebra~~
3. ~~Finite Deterministic Automata~~
4. Regular Expressions & Languages
5. Monoids
6. Metric Spaces // the secret ingredient to ML

// Plus a short note at the end about how Pin denotes functions …

Regular Expressions & Languages

Let A be a finite alphabet. Rat(A*), is the smallest subset of A* satisfying all the following:

…

Regular Expressions & Languages

Let A be a finite alphabet. Rat(A*), is the smallest subset of A* satisfying all the following:

- Rat(A*) contains { }

Regular Expressions & Languages

Let A be a finite alphabet. Rat(A*), is the smallest subset of A* satisfying all the following:

- Rat(A*) contains { }
- If a is in A then { a } is in Rat(A*)

Regular Expressions & Languages

Let A be a finite alphabet. Rat(A*), is the smallest subset of A* satisfying all the following:

- Rat(A*) contains { }
- If a is in A then { a } is in Rat(A*)
- If X and Y are in Rat(A*) then so are X Y, X U Y, and X*.

Regular Expressions & Languages

Let A be a finite alphabet. Rat(A*), is the smallest subset of A* satisfying all the following:

- Rat(A*) contains { }
- If a is in A then { a } is in Rat(A*)
- If X and Y are in Rat(A*) then so are X Y, X U Y, and X*.

Aka “Rational”.

Regular Expressions & Languages

Let A be a finite alphabet. Rat(A*), is the smallest subset of A* satisfying all the following:

- Rat(A*) contains { }
- If a is in A then { a } is in Rat(A*)
- If X and Y are in Rat(A*) then so are X Y, X U Y, and X*.

Aka “Rational”.

Theorem: Every finite language is regular.

Regular Expressions & Languages

Let A be a finite alphabet. Rat(A*), is the smallest subset of A* satisfying all the following:

- Rat(A*) contains { }
- If a is in A then { a } is in Rat(A*)
- If X and Y are in Rat(A*) then so are X Y, X U Y, and X*.

Aka “Rational”.

Theorem: Every finite language is regular.

{ a , ab, ba }*, A*abaA*, abA*, set of words of even length

Regular Expressions & Languages

Let A be a finite alphabet. Rat(A*), is the smallest subset of A* satisfying all the following:

- Rat(A*) contains { }
- If a is in A then { a } is in Rat(A*)
- If X and Y are in Rat(A*) then so are X Y, X U Y, and X*.

Aka “Rational”.

Theorem: Every finite language is regular.

{ a , ab, ba }*, A*abaA*, abA*, set of words of even length

A*abaA* is the “regular expression” defining the “regular language” { $a_0 \dots a_k$ aba $a_{k+1} \dots a_{k+t}$ | a_i in A and t, k naturals }

Table of Contents

1. ~~First Order Logic // you already know this~~
2. ~~Words and Languages // aka easy algebra~~
3. ~~Finite Deterministic Automata~~
4. ~~Regular Expressions & Languages~~
5. Monoids
6. Metric Spaces // the secret ingredient to ML

// Plus a short note at the end about how Pin denotes functions …

Monoids

M * 1

Monoids

M   *   1

Set.

Monoids

M * 1

Set.

Operator M x M → M satisfying
(associativity) (n * m) * k = n * (m * k)

Monoids

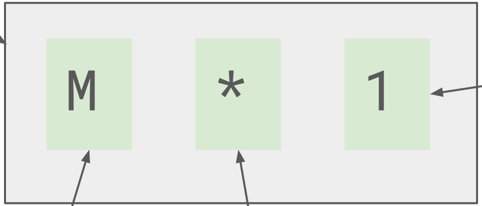
M * 1

Identity: $m * 1 = 1 * m = m$

Set.

Operator $M \times M \rightarrow M$ satisfying
(associativity) $(n * m) * k = n * (m * k)$

Monoids

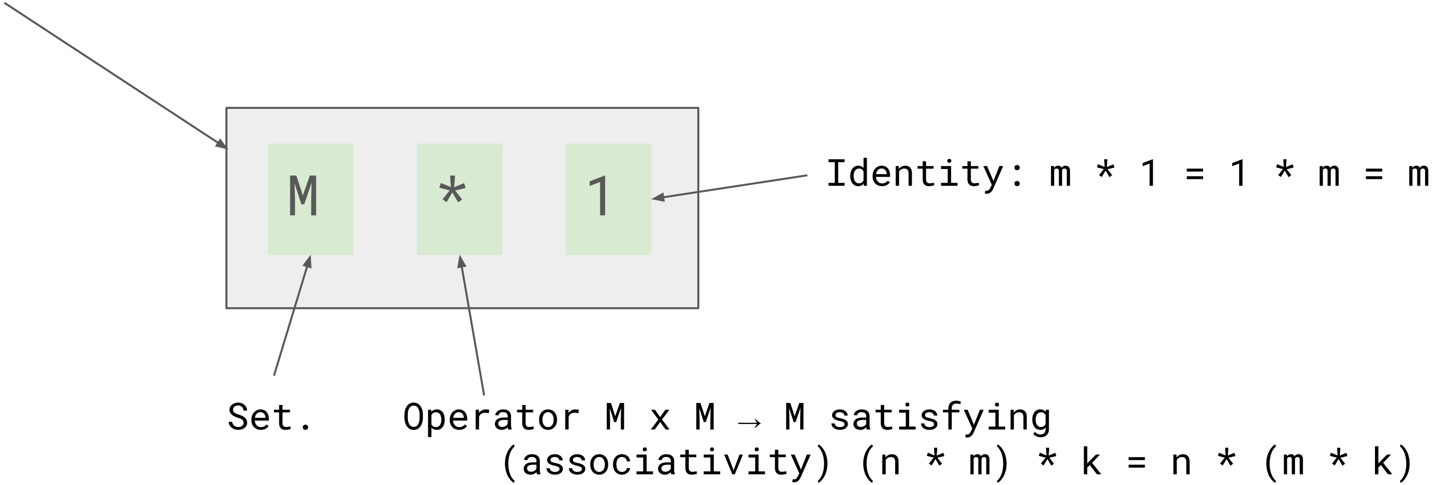


Identity: m * 1 = 1 * m = m

Set.

Operator M x M → M satisfying
(associativity) (n * m) * k = n * (m * k)

Monoids



A* is the “free monoid” over A, with * = concatenation and 1 = “”.

Monoids

Suppose that φ is a function …

$$\varphi : \boxed{M \quad * \quad 1} \longrightarrow \boxed{N \quad @ \quad e}$$

… s.t. $\varphi(1)=e$, and $\varphi(mk)=\varphi(m)\varphi(k)$.

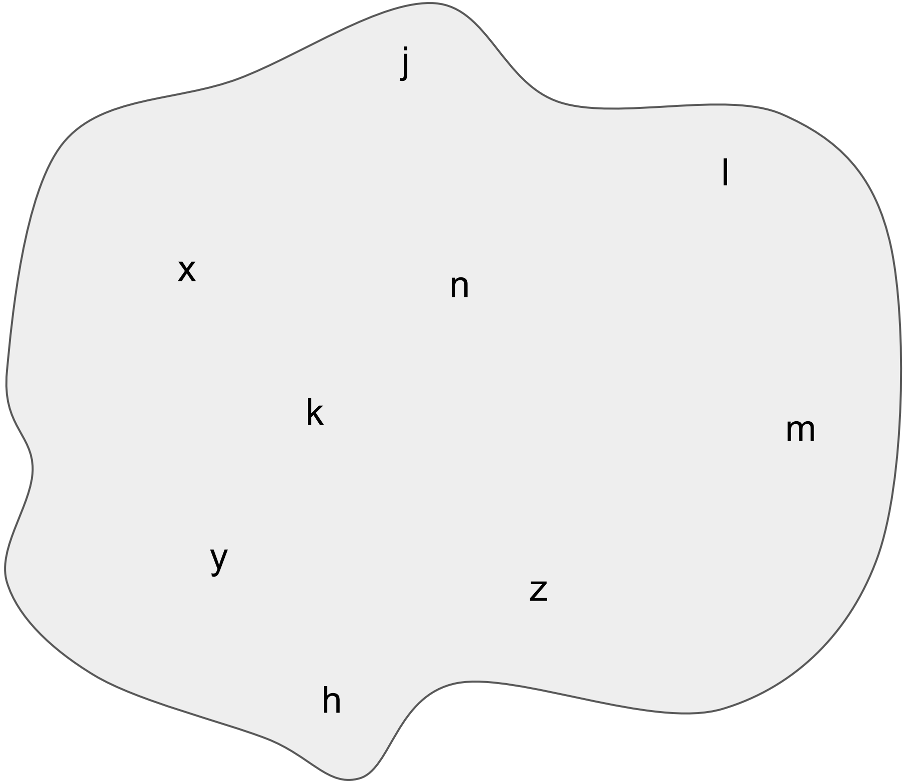
Then we say φ is a **morphism**.

Table of Contents

1. ~~First Order Logic // you already know this~~
2. ~~Words and Languages // aka easy algebra~~
3. ~~Finite Deterministic Automata~~
4. ~~Regular Expressions & Languages~~
5. ~~Monoids~~
6. Metric Spaces // the secret ingredient to ML

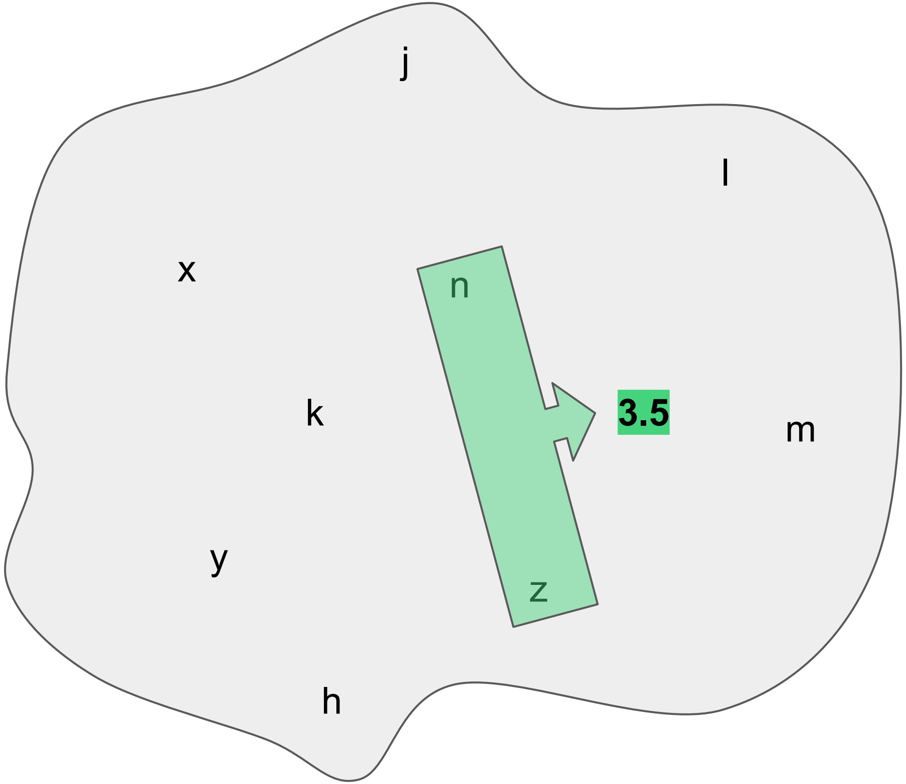
// Plus a short note at the end about how Pin denotes functions …

Metric Spaces



X is a Set.

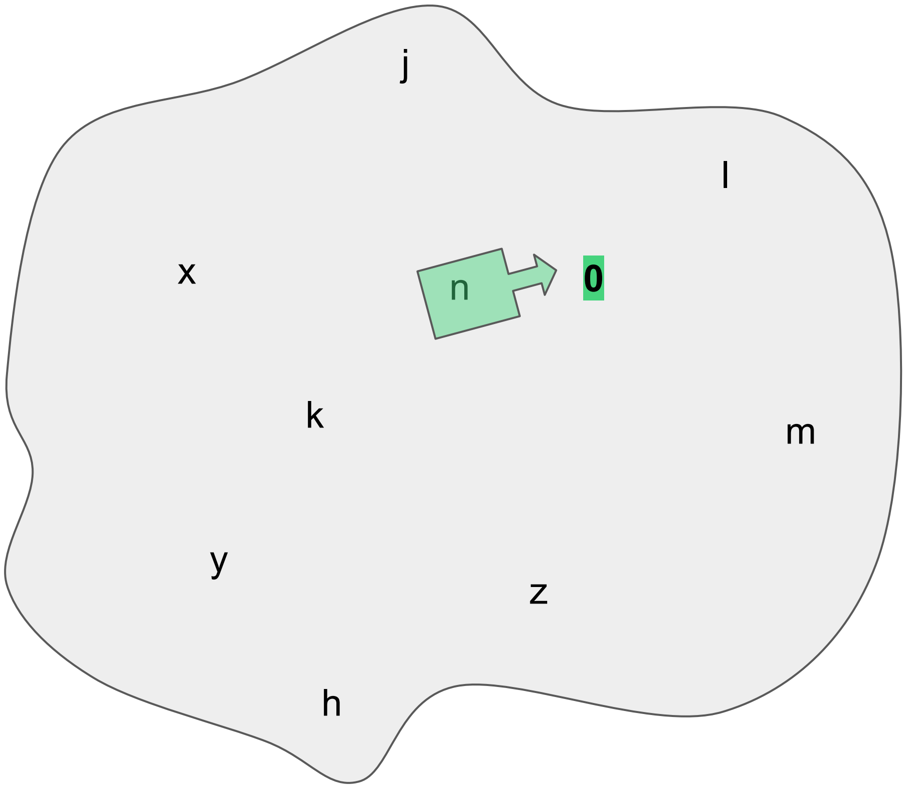
Metric Spaces



X is a Set.

d : X x X → ℝ is a function,
Satisfying:

Metric Spaces

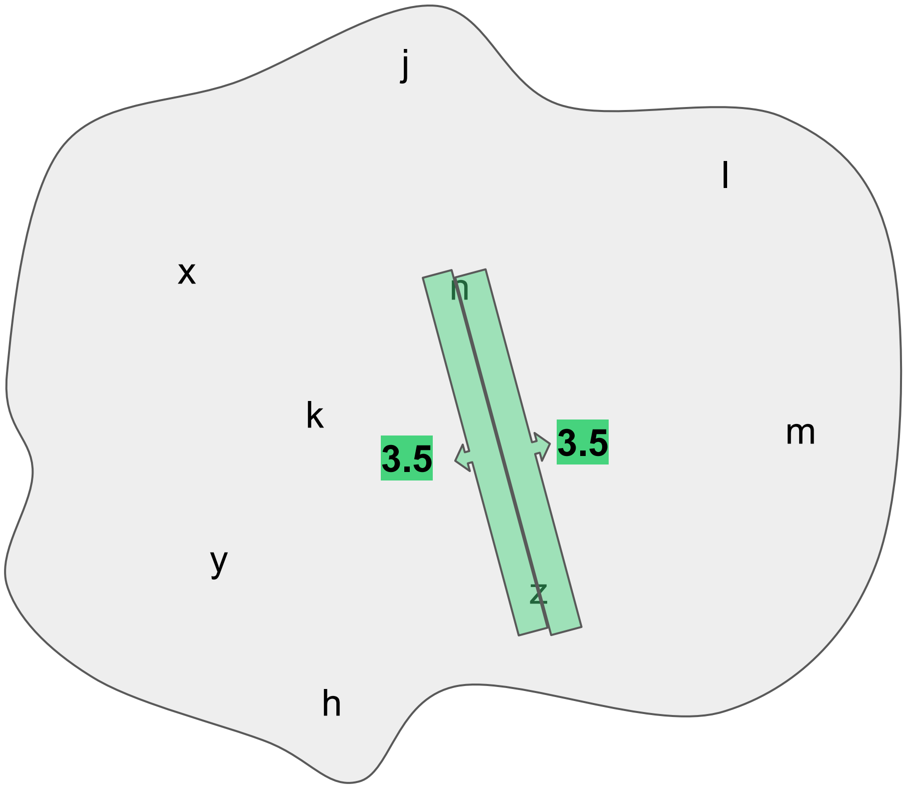


X is a Set.

d : X x X → ℝ is a function,
Satisfying:

- **d(x, y) = 0 iff x = y**

Metric Spaces

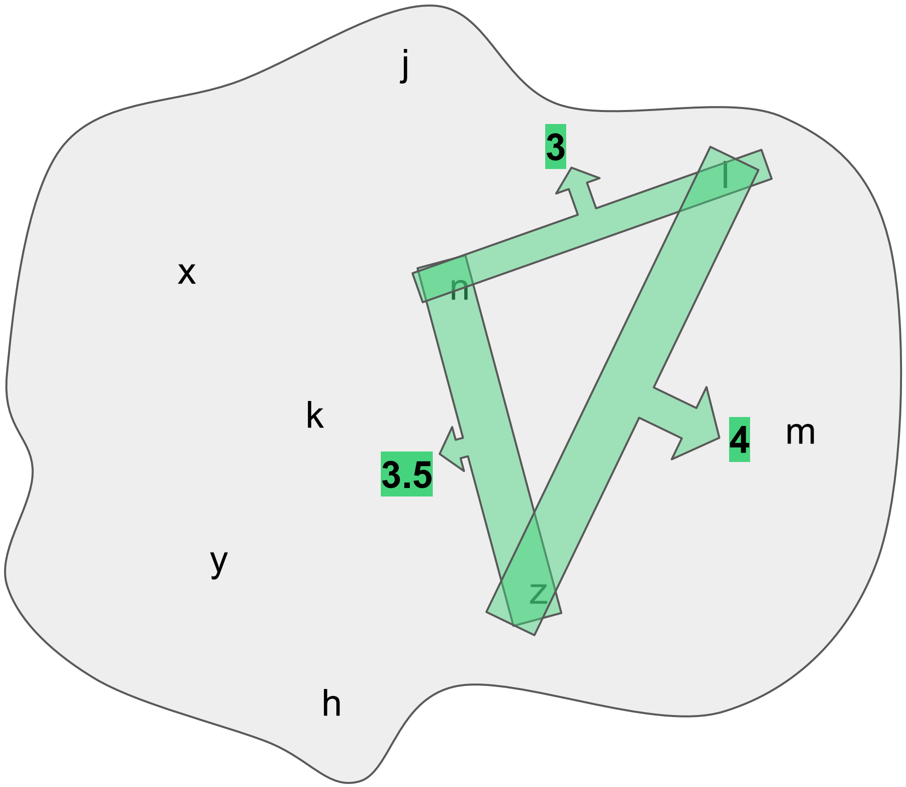


X is a Set.

d : X x X → ℝ is a function,
Satisfying:

- d(x, y) = 0 iff x = y
- **d(x, y) = d(y, x)**

Metric Spaces

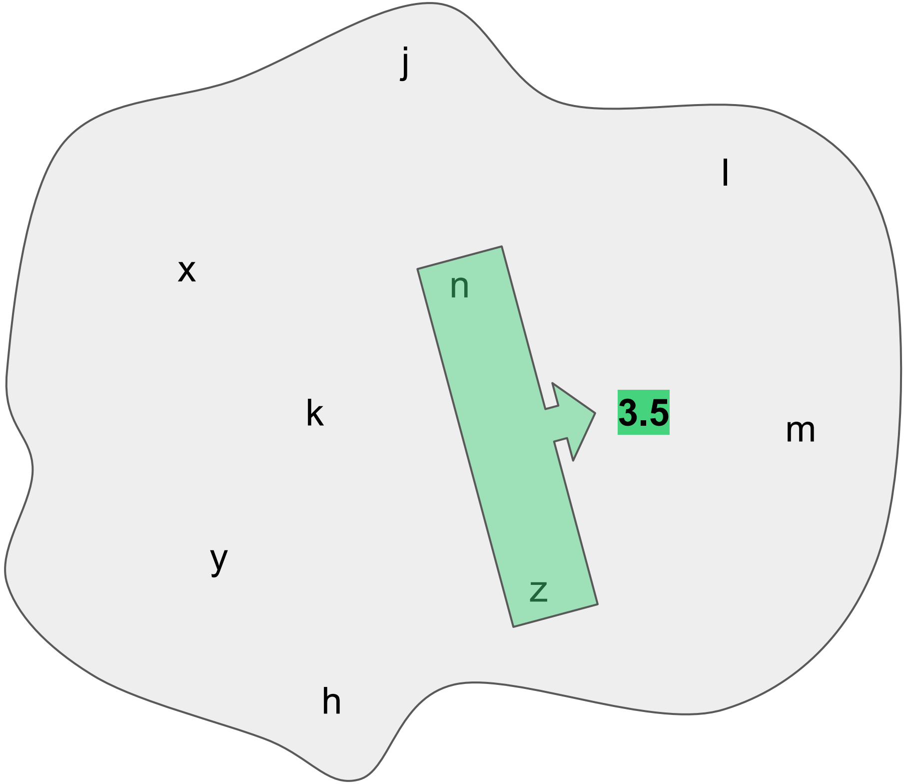


X is a Set.

d : X x X → ℝ is a function,
Satisfying:

- $d(x, y) = 0$ iff $x = y$
- $d(x, y) = d(y, x)$
- **$d(x, y) + d(y, z) \geq d(x, z)$**

Metric Spaces



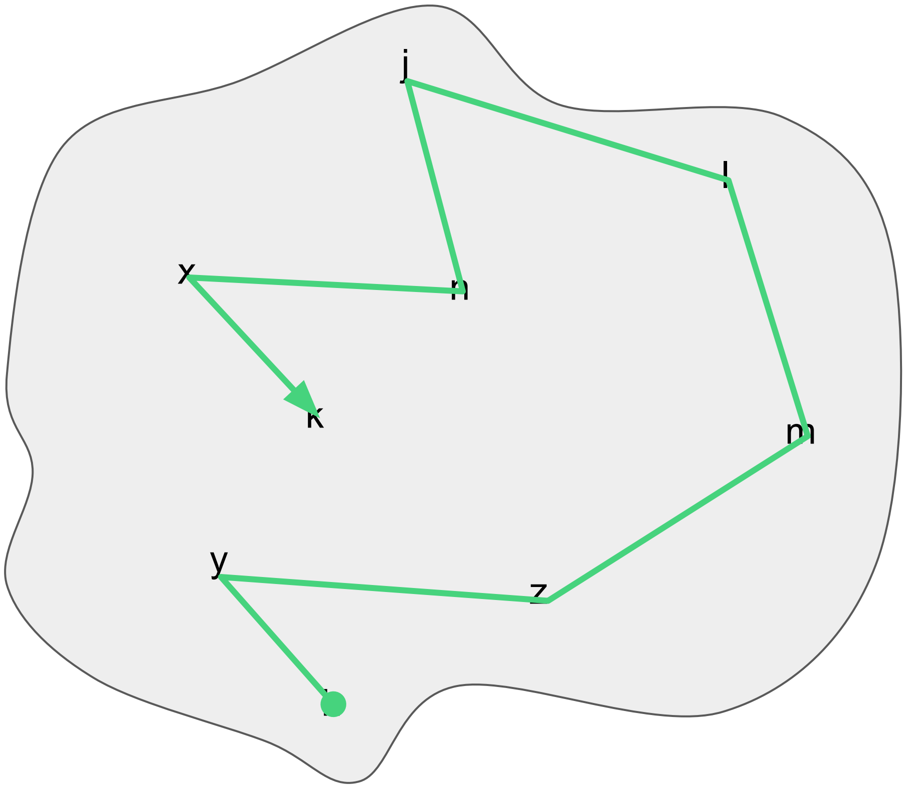
X is a Set.

d : X x X → ℝ is a function,
Satisfying:

- d(x, y) = 0 iff x = y
- d(x, y) = d(y, x)
- d(x, y) + d(y, z) ≥ d(x, z)

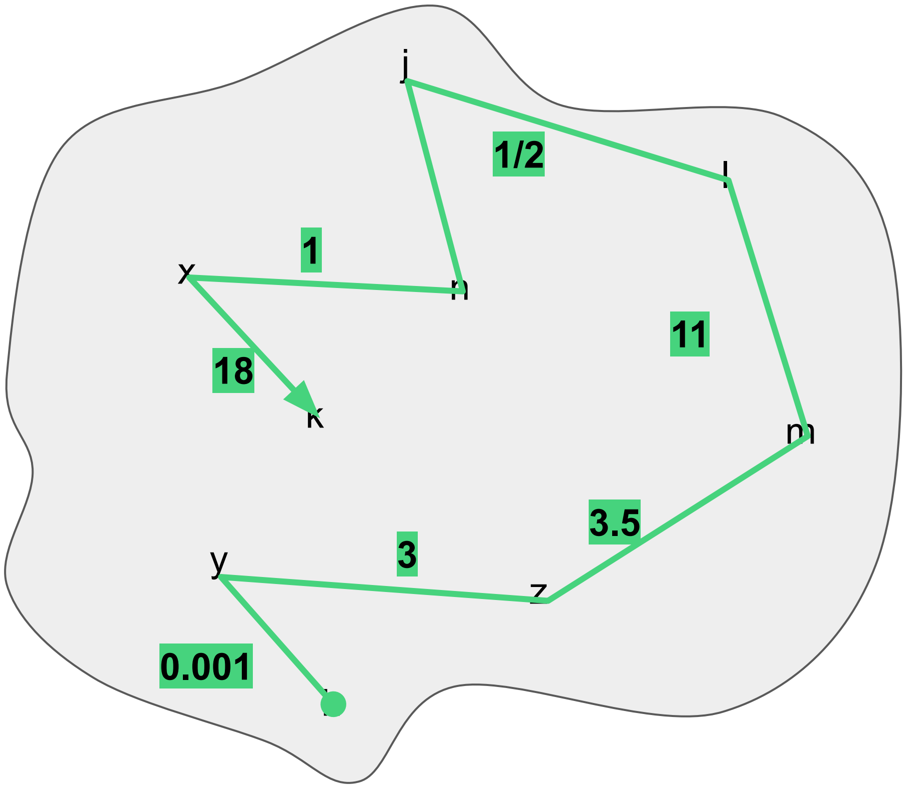
Then d is a “metric” on X, and (X, d) is a “metric space”.

Metric Spaces



Consider a sequence through the space.

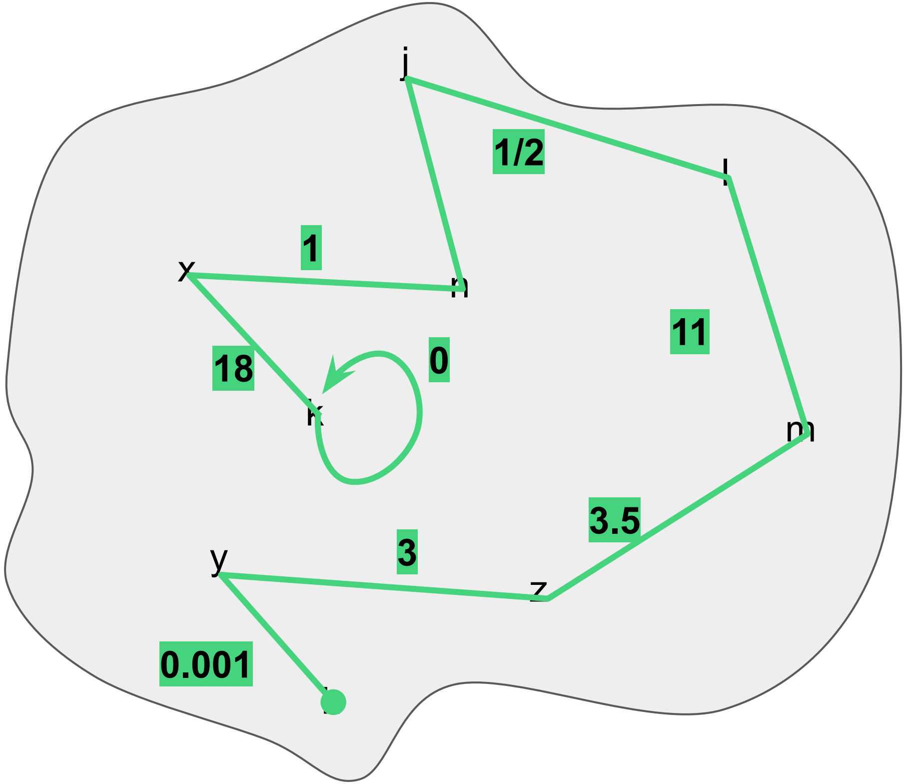
Metric Spaces



Consider a sequence through the space.

We can use our metric to measure the distance of every segment in the sequence.

Metric Spaces

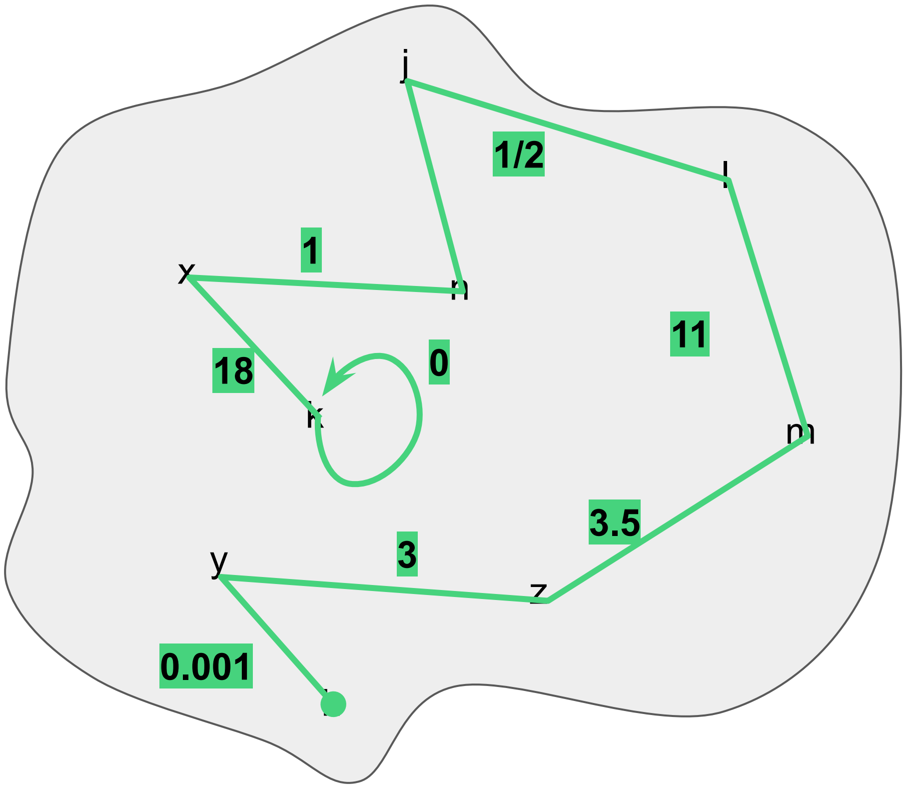


Consider a sequence through the space.

We can use our metric to measure the distance of every segment in the sequence.

Maybe the sequence is infinite.

Metric Spaces



Consider a sequence through the space.

We can use our metric to measure the distance of every segment in the sequence.

Maybe the sequence is infinite.

If the sequence is infinite, and the distances converge to zero, then the sequence is “cauchy”.

Metric Spaces

1, ½, ⅓, ¼, ⅕, ⅙, …

(10, 10, 10), (10, 1, 1), (10, 0.1, 0.1), (10, 0.001, 0.001), …

3, 3.1, 3.14, 3.141, 3.1415, 3.14159, …

Consider a sequence through the space.

We can use our metric to measure the distance of every segment in the sequence.

Maybe the sequence is infinite.

If the sequence is infinite, and the distances converge to zero, then the sequence is “cauchy”.

Metric Spaces

1, ½, ⅓, ¼, ⅕, ⅙, …

(10, 10, 10), (10, 1, 1), (10, 0.1, 0.1),
(10, 0.001, 0.001), …

3, 3.1, 3.14, 3.141, 3.1415, 3.14159, … → π

OBSERVE: A Cauchy sequence might *not* converge in the space.

Metric Spaces

$1, \frac{1}{2}, \frac{1}{3}, \frac{1}{4}, \frac{1}{5}, \frac{1}{6}, \ldots$

(10, 10, 10), (10, 1, 1), (10, 0.1, 0.1), (10, 0.001, 0.001), …

3, 3.1, 3.14, 3.141, 3.1415, 3.14159, …

OBSERVE: A Cauchy sequence might *not* converge in the space.

If every Cauchy sequence converges inside the space, then we say the metric space is “Complete”.

Metric Spaces

1, ½, ⅓, ¼, ⅕, ⅙, …

(10, 10, 10), (10, 1, 1), (10, 0.1, 0.1),
(10, 0.001, 0.001), …

3, 3.1, 3.14, 3.141, 3.1415, 3.14159, …

OBSERVE: A Cauchy sequence might *not* converge in the space.

If every Cauchy sequence converges inside the space, then we say the metric space is “Complete”.

Suppose (X, d) is a metric space, X is a subset of Z.

Metric Spaces

1, ½, ⅓, ¼, ⅕, ⅙, …

(10, 10, 10), (10, 1, 1), (10, 0.1, 0.1), (10, 0.001, 0.001), …

3, 3.1, 3.14, 3.141, 3.1415, 3.14159, …

OBSERVE: A Cauchy sequence might *not* converge in the space.

If every Cauchy sequence converges inside the space, then we say the metric space is "Complete".

Suppose (X, d) is a metric space, X is a subset of Z.

We say X is *dense* in Z if for all z in Z, either z is in X, or there exists a Cauchy sequence in X which converges to z.

Metric Spaces

1, ½, ⅓, ¼, ⅕, ⅙, …

(10, 10, 1
(10, 0.00

3, 3.1, 3.

OBSERVE: A
converge i

If every Cauchy sequence
converges inside the space,
space is

ric
Z.

if for
in X,
y
sequence in X which converges
to z.

QUESTION If (X, d) is *not* complete, can I “fill in the holes” to make it complete?

Metric Spaces

Let (X, d) be a metric space.

Metric Spaces

Let (X, d) be a metric space.

For each pair of Cauchy sequences **z**, **v** in the space, let

$$D(\mathbf{z}, \mathbf{v}) = \lim_{n \to \infty} d(\mathbf{z}[n], \mathbf{v}[n])$$

Metric Spaces

Let (X, d) be a metric space.

For each pair of Cauchy sequences **z**, **v** in the space, let

$$D(\mathbf{z}, \mathbf{v}) = \lim_{n \to \infty} d(\mathbf{z}[n], \mathbf{v}[n])$$

Let C(X) = the Cauchy sequences in (X, d), such as **z** and **v**.

Metric Spaces

Let (X, d) be a metric space.

For each pair of Cauchy sequences **z**, **v** in the space, let

$$D(\mathbf{z}, \mathbf{v}) = \lim_{n \to \infty} d(\mathbf{z}[n], \mathbf{v}[n])$$

Let C(X) = the Cauchy sequences in (X, d), such as **z** and **v**.

Let **z** ~ **v** iff D(**z**, **v**) = 0.

Metric Spaces

Let (X, d) be a metric space.

For each pair of Cauchy sequences **z**, **v** in the space, let

$$D(\mathbf{z}, \mathbf{v}) = \lim_{n \to \infty} d(\mathbf{z}[n], \mathbf{v}[n])$$

Let C(X) = the Cauchy sequences in (X, d), such as **z** and **v**.

Let **z** ~ **v** iff D(**z**, **v**) = 0. Let [**z**]=[**v**]={ **j** in C(x) | **j** ~ **z** }.

Metric Spaces

Let (X, d) be a metric space.

For each pair of Cauchy sequences **z**, **v** in the space, let

$$D(\mathbf{z}, \mathbf{v}) = \lim_{n \to \infty} d(\mathbf{z}[n], \mathbf{v}[n])$$

Let C(X) = the Cauchy sequences in (X, d), such as **z** and **v**.

Let **z** ~ **v** iff D(**z**, **v**) = 0. Let [**z**]=[**v**]={ **j** in C(x) | **j** ~ **z** }.

Let [C(X)] = { [**x**] | **x** in C(x) }.

Metric Spaces

Let (X, d) be a metric space.

For each pair of Cauchy sequences **z**, **v** in the space, let

$$D(\mathbf{z}, \mathbf{v}) = \lim_{n \to \infty} d(\mathbf{z}[n], \mathbf{v}[n])$$

Let C(X) = the Cauchy sequences in (X, d), such as **z** and **v**.

Let **z** ~ **v** iff D(**z**, **v**) = 0. Let [**z**]=[**v**]={ **j** in C(x) | **j** ~ **z** }.

Let [C(X)] = { [**x**] | **x** in C(x) }. Let [D]([**z**], [**v**]) = D(**z**, **v**).

Metric Spaces

Let (X, d) be a metric space.

For each pair of Cauchy sequences **z**, **v** in the space, let

$$D(\mathbf{z}, \mathbf{v}) = \lim_{n \to \infty} d(\mathbf{z}[n], \mathbf{v}[n])$$

Let C(X) = the Cauchy sequences in (X, d), such as **z** and **v**.

Let **z** ~ **v** iff D(**z**, **v**) = 0. Let [**z**]=[**v**]={ **j** in C(x) | **j** ~ **z** }.

Let [C(X)] = { [**x**] | **x** in C(x) }. Let [D]([**z**], [**v**]) = D(**z**, **v**).

Then ([C(x)], [D]) is a metric space called the “completion” of (X, d).

Metric Spaces

Let (X

For ea space, let

D(**z**

Let C(h as **z** and **v**.

Let **z** C(x) | **j** ~ **z** }.

Let [C) = D(**z**, **v**).

Then (mpletion" of
(X, d)

Table of Contents

1. ~~First Order Logic // you already know this~~
2. ~~Words and Languages // aka easy algebra~~
3. ~~Finite Deterministic Automata~~
4. ~~Regular Expressions & Languages~~
5. ~~Monoids~~
6. ~~Metric Spaces // the secret ingredient to ML~~

// Plus a short note at the end about how Pin denotes functions …

Finally, how Pin denotes functions.

Dr. Pin uses postfix notation …

Finally, how Pin denotes functions.

Dr. Pin uses postfix notation …

Suppose $f : X \to Y$, and $g : Y \to Z$.

Finally, how Pin denotes functions.

Dr. Pin uses postfix notation …

Suppose $f : X \to Y$, and $g : Y \to Z$.

Then $g \circ f : X \to Z$

Finally, how Pin denotes functions.

Dr. Pin uses postfix notation …

Suppose $f : X \to Y$, and $g : Y \to Z$.

Then $g \circ f : X \to Z$

$Y \to Z$ $X \to Y$

Finally, how Pin denotes functions.

Dr. Pin uses postfix notation …

Suppose $f : X \to Y$, and $g : Y \to Z$.

Then $g \circ f : X \to Z$

$Y \to Z$ $X \to Y$

Kinda lame because the signatures don't line up.

Finally, how Pin denotes functions.

Dr. Pin uses postfix notation …

Suppose $f : X \to Y$, and $g : Y \to Z$.

Then $g \circ f : X \to Z$

$Y \to Z$ $X \to Y$

Kinda lame because the signatures don't line up.

What if instead we wrote x f g?

$x\ (X \to Y)\ (Y \to Z)$

Finally, how Pin denotes functions.

Dr. Pin uses postfix notation …

Suppose $f : X \to Y$, and $g : Y \to Z$.

Then $g \circ f : X \to Z$

$Y \to Z$ $X \to Y$

Kinda lame because the signatures don’t line up.

What if instead we wrote x f g?

```
x ( X → Y ) (Y → Z )
    X -----------> Z
```

Finally, how Pin denotes functions.

Dr. Pin uses postfix notation …

Suppose f : X → Y, and g : Y → Z.

Then g ∘ f : X → Z

Y → Z X → Y

Kinda lame because the signatures don’t line up.

What if instead we wrote x f g?

```
x ( X → Y ) (Y → Z )
    X -----------> Z
```

Much prettier!

The End.

Questions?