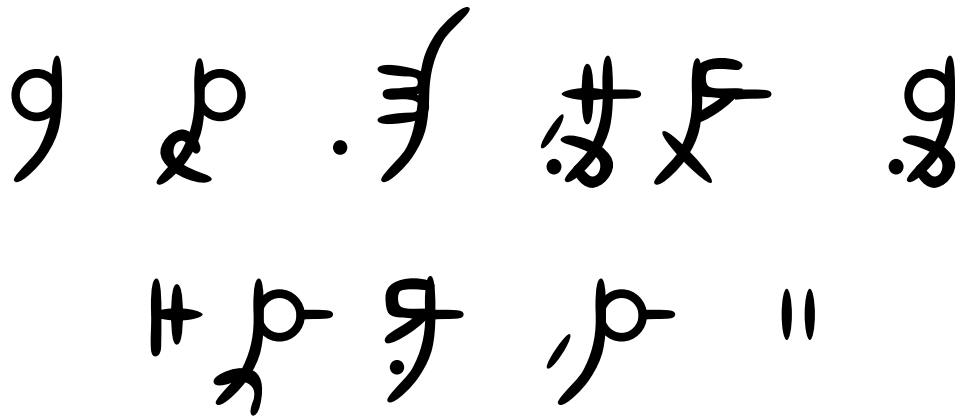
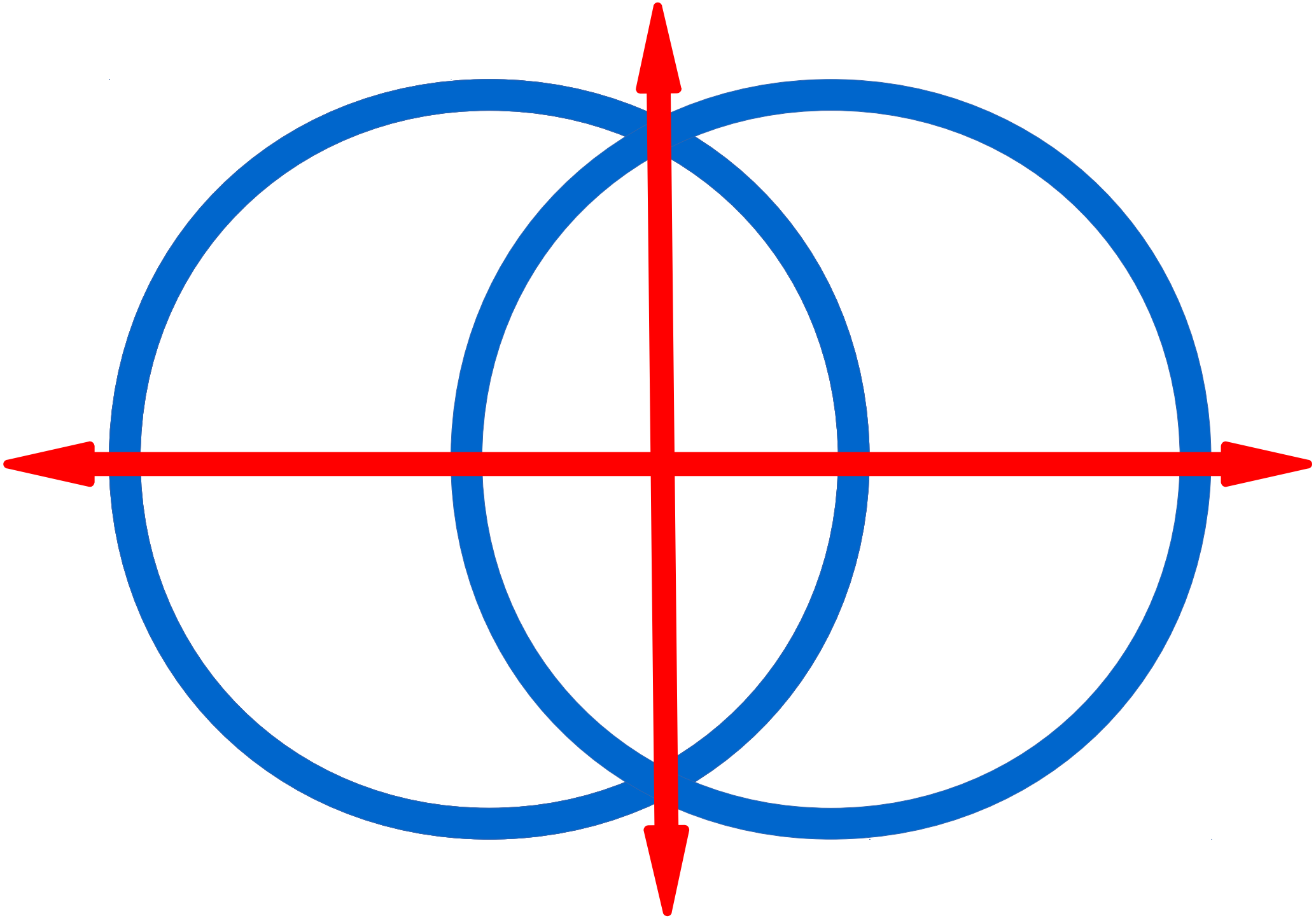
Set Semantics
In Davin

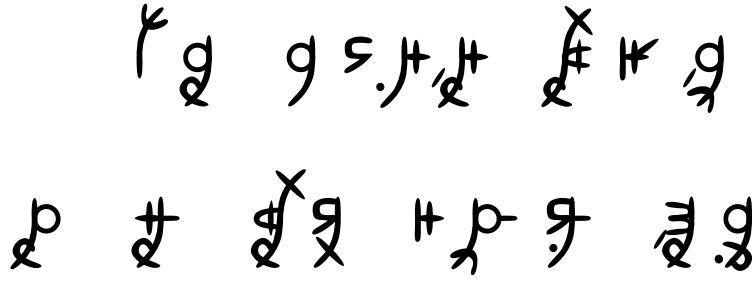




What is Davin ?

This is an example sentence

written using Davin.

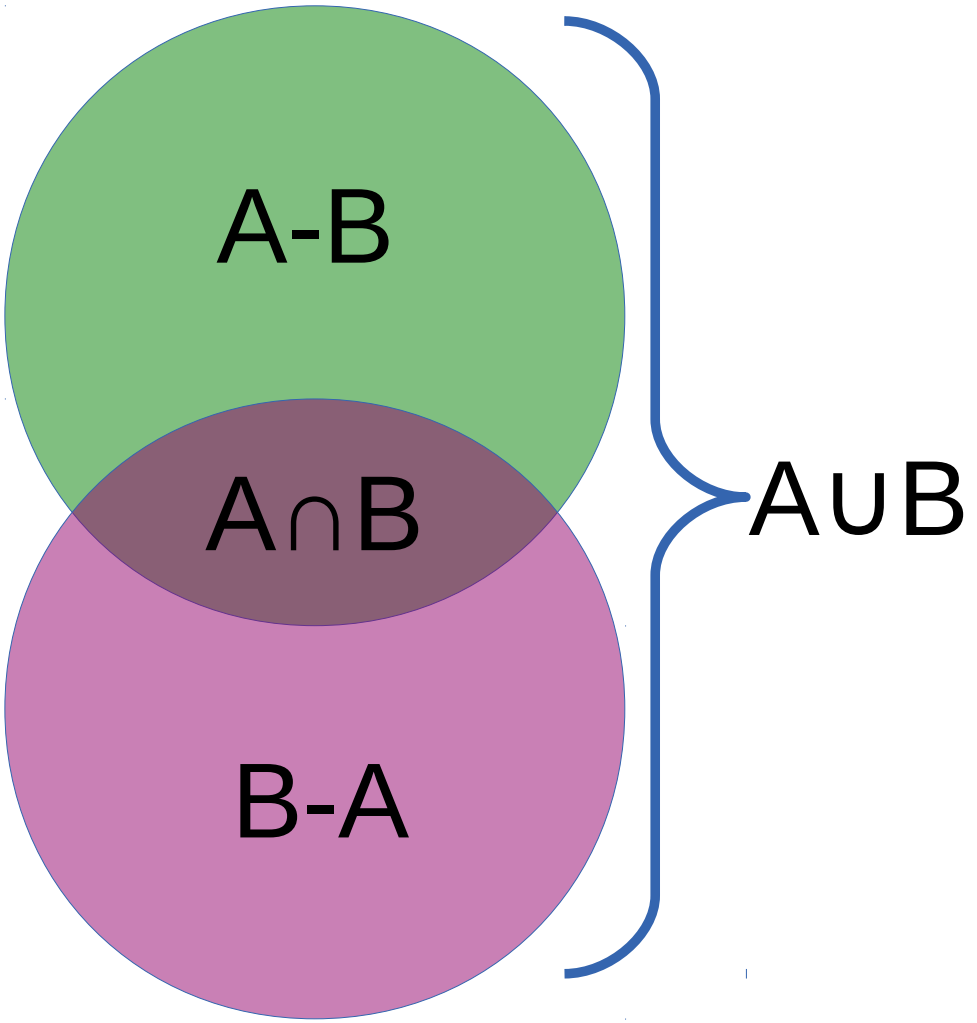


wep iprɪdend ejgzamp

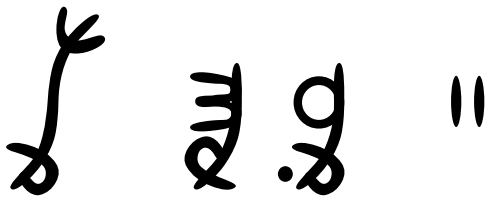
eb es ejkir davin enþup ilnt.

Set theory

- Sets are groups of things
- Any element of a set occurs exactly once
- Recall intersection, union and others



We use Davin as a tool



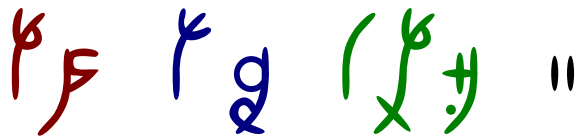
- This is only one syntactic formulation over a portable logical core
- It's the one we will use, however

Davin Basics: Linguistically

S V

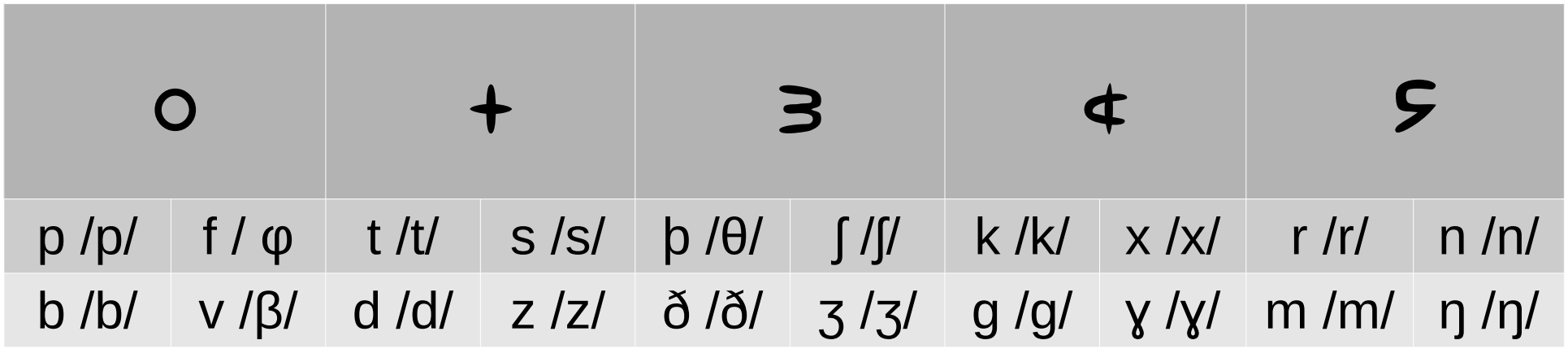
- Isolational
- Strongly head final

A O V



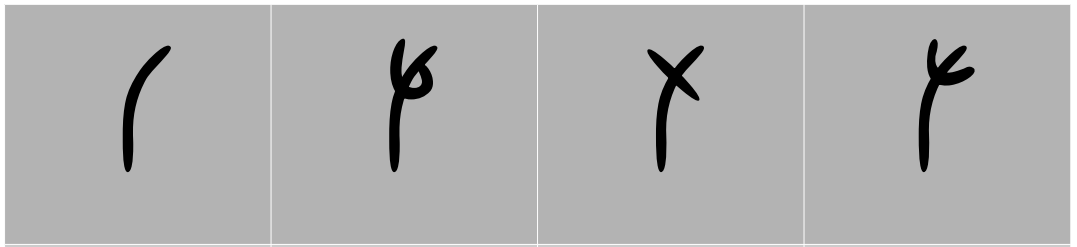
- Ergative
- AOV SV S=O

Davin Basics: Phonology



p /p/	f / φ	t /t/	s /s/	þ /θ/	ʃ /ʃ/	k /k/	x /x/	r /r/	n /n/
b /b/	v /β/	d /d/	z /z/	ð /ð/	ʒ /ʒ/	g /g/	ɣ /ɣ/	m /m/	ŋ /ŋ/

y /ʌ/	ɪ /ɪ/	i /i/	a /a/	e /ɛ/	o /o/	u /ɯ/



l /l/	h /h/	j /j/	w /w/

Davin Basics: Word Classes

- aŋtah
  - Pronouns, articles, and proper nouns
  - Start with consonants
  - “Pushed”
- owpys
  - Nouns, adjectives, verbs, postpositions
  - Start with vowels
  - Operators

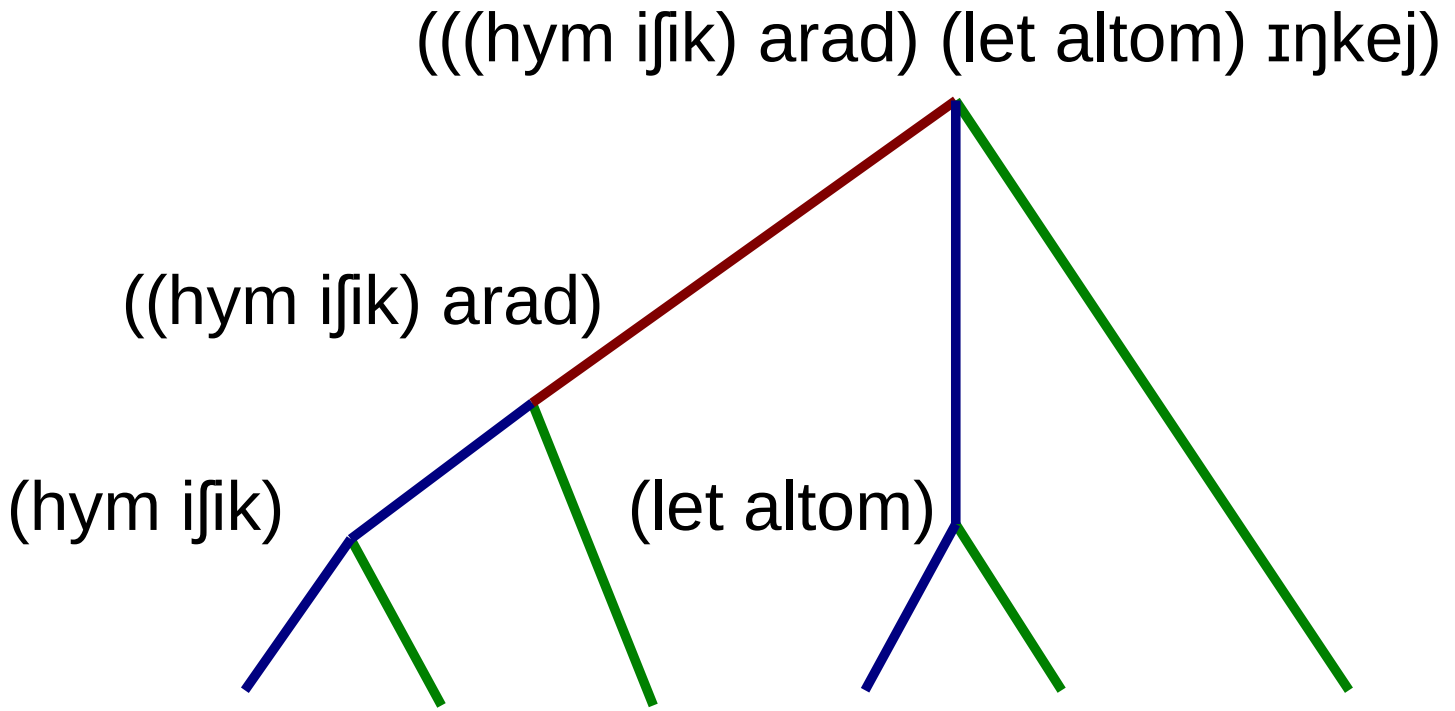
    “Pop then push”

Davin Basics: conjugation

- owpys are by default intransitive
- Conjugate first syllable to make transitive
- glossed with -TRAN or simply -T

Group	T
p b f v	m
t d s z	n
þ ð ʃ ʒ	n
k g x ɣ	ŋ

Davin Basics: iprid

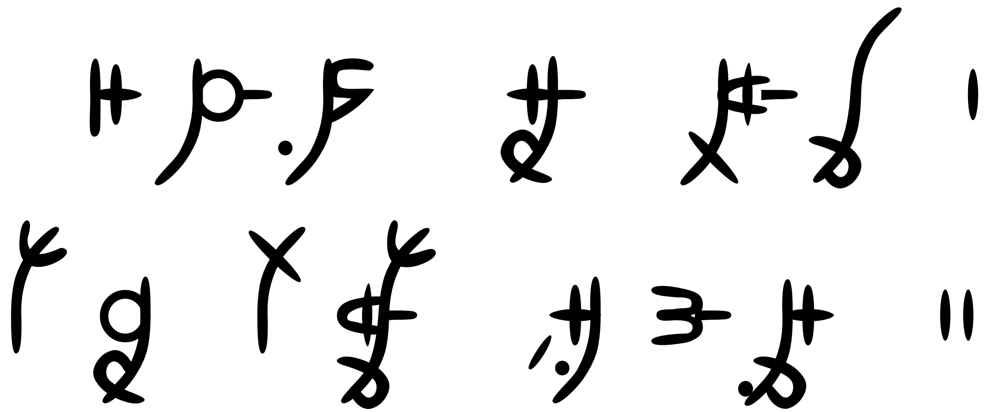


hym iʃik arad let altom ɪŋkej.

Davin Basics: aʃov

There are 10 grammar words called aʃov

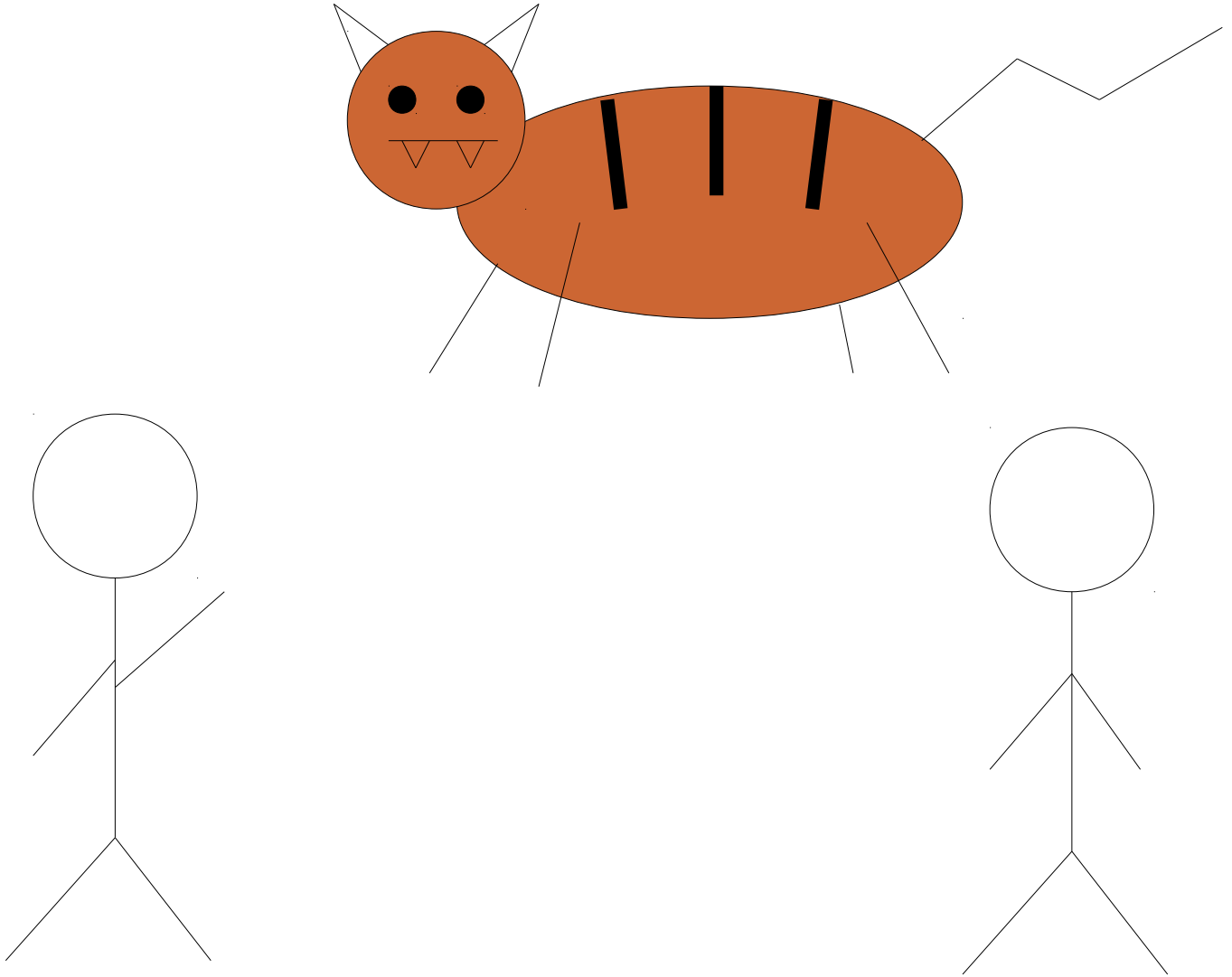
- e 
- ow 
- es 
- ens 
- eb 
- ɪlp
- owp
- yp
- ymp
- up

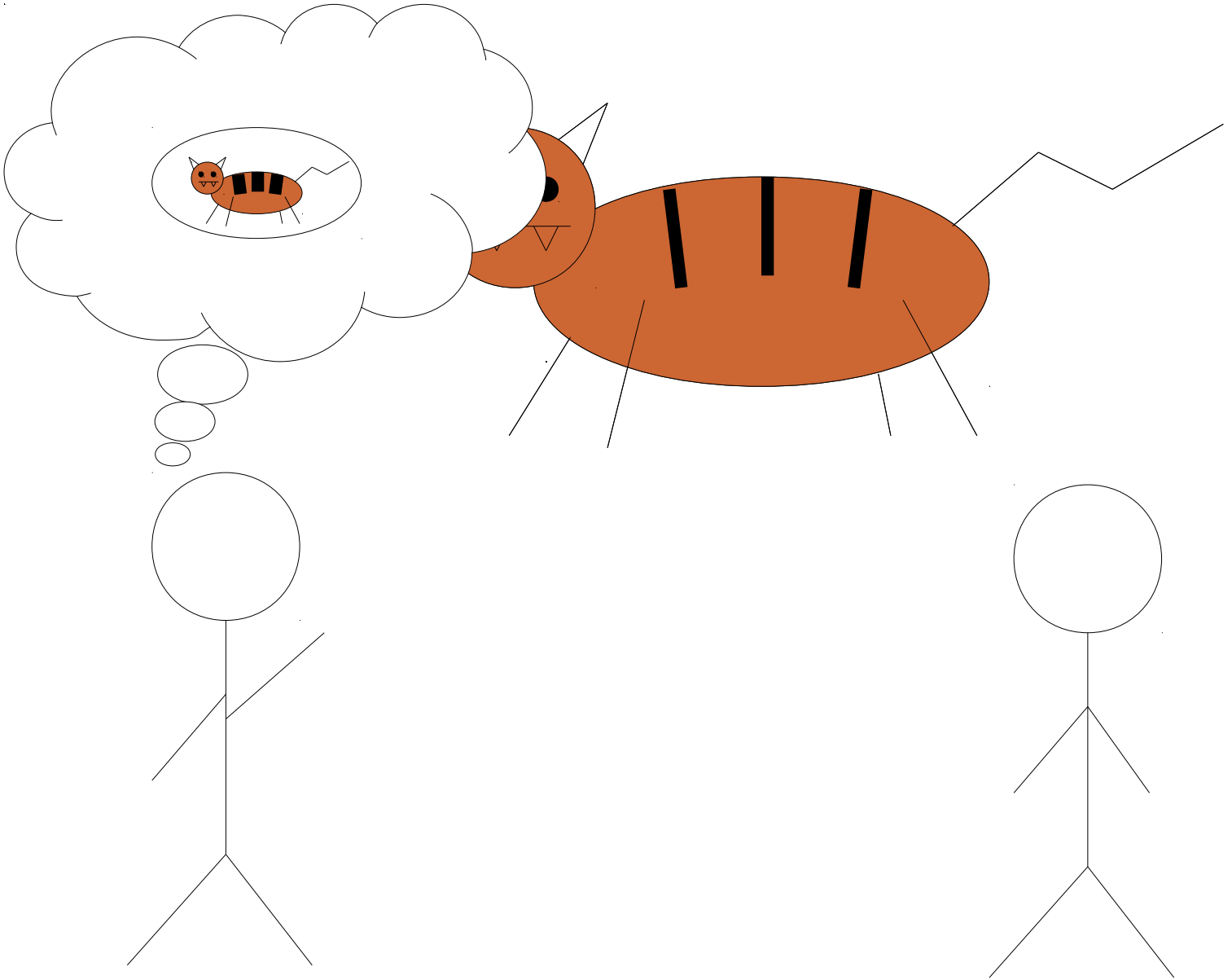


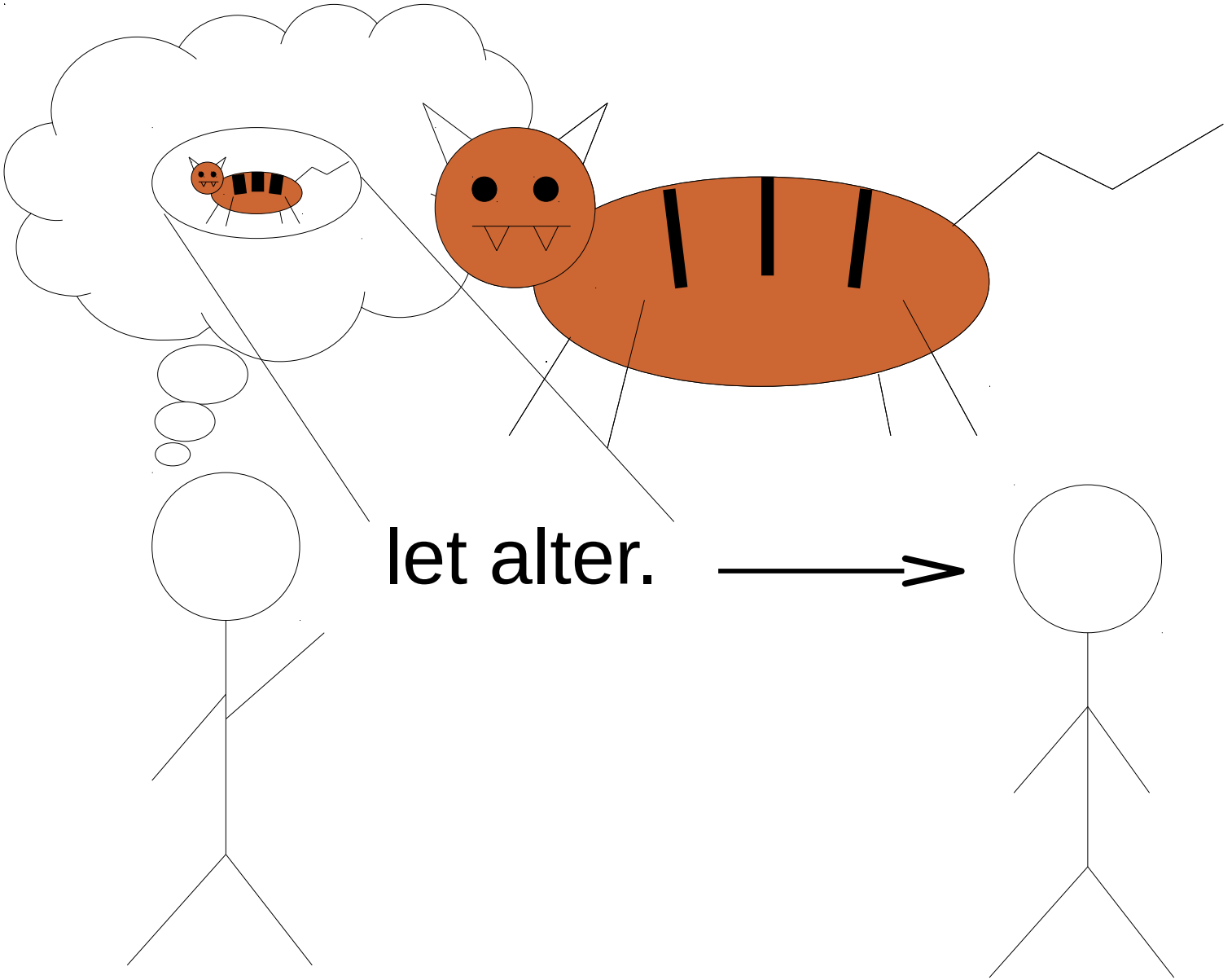
Set Semantics

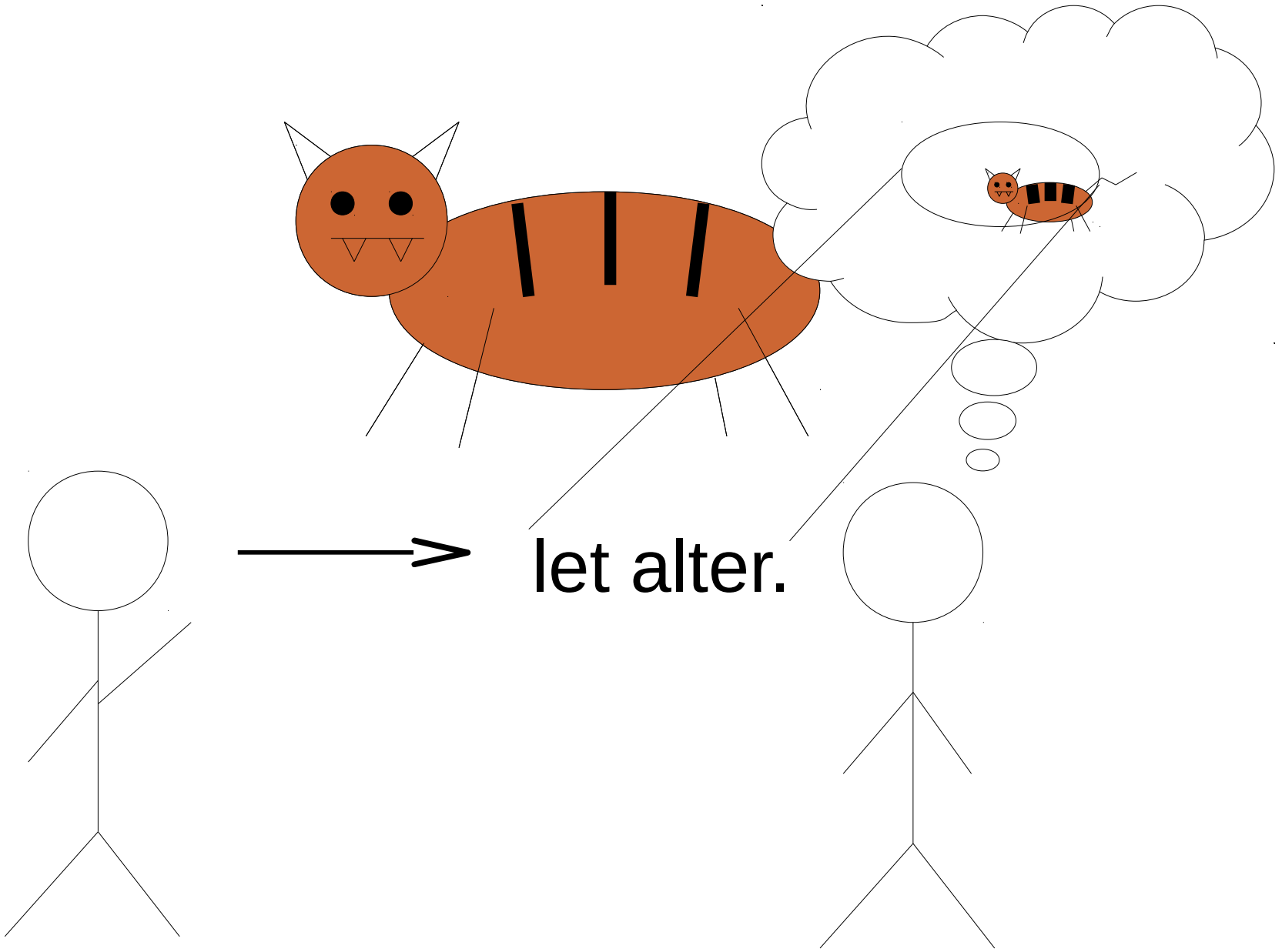
- Humans like to tell stories
- Davin is declarative (CS sense)
- *What* instead of *how*
- We do this by specifying sets

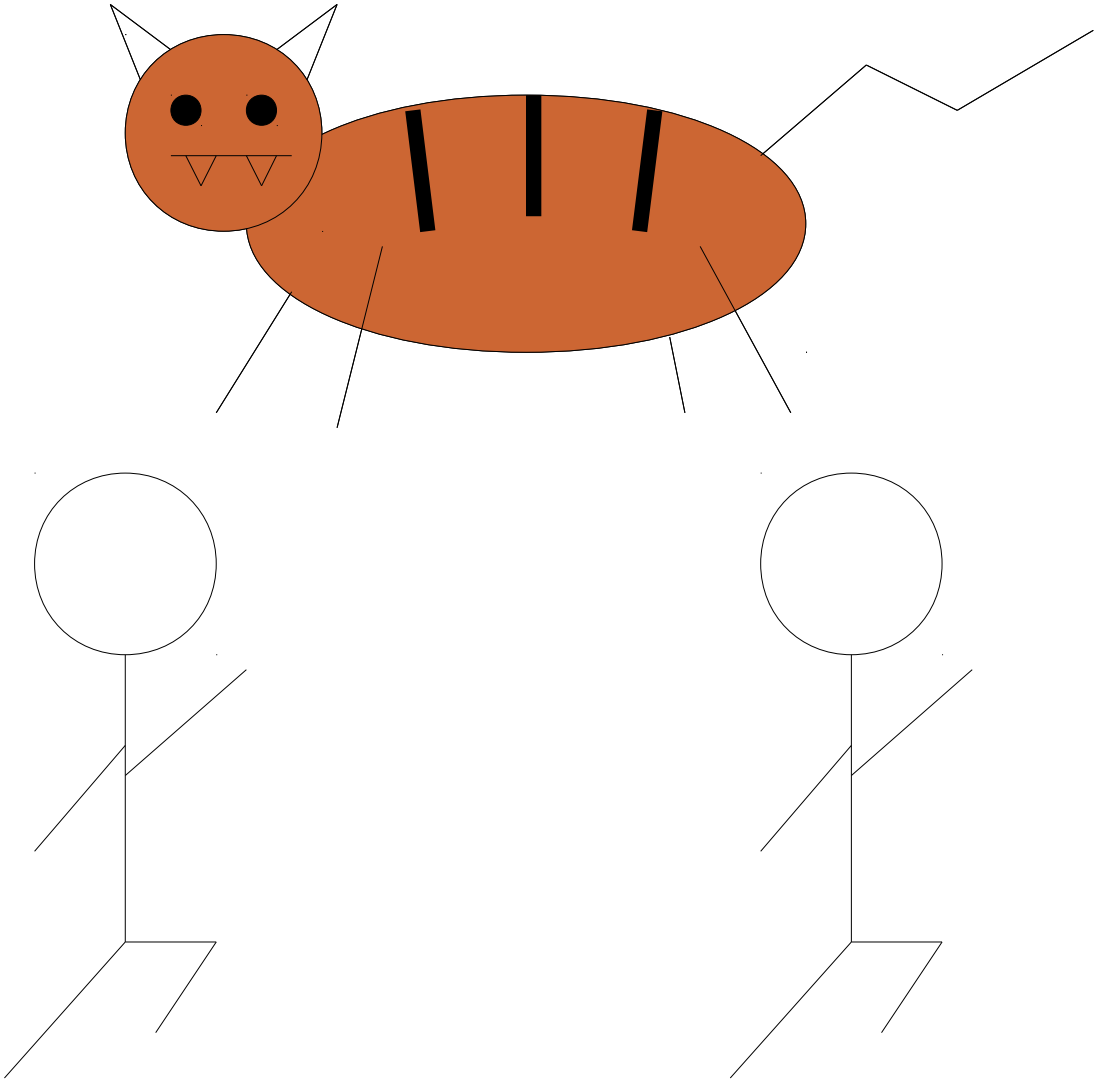
let alter.

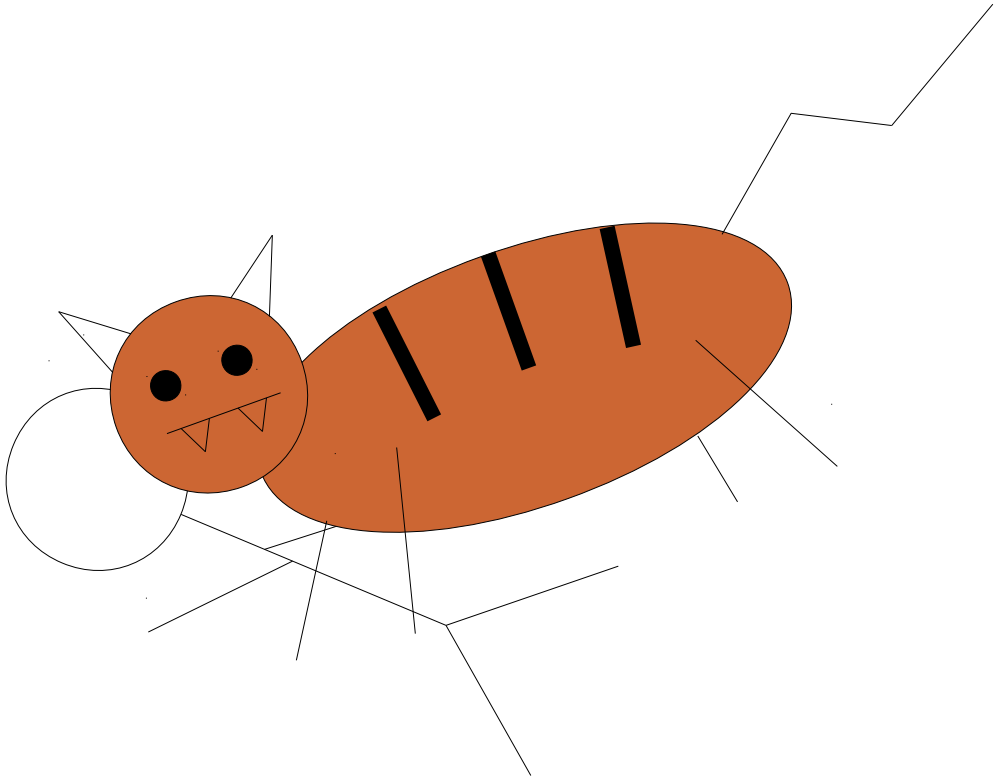





let alter.


let alter.





Let's build some sets

- We need to start somewhere
- aŋtah are our basic building blocks
- They are pre-existing specific sets

Pronouns		Articles		Names	
wit	I	let	The	zakari	Zachary
wep	this	hym	A / Some	tostɪn	Austin
hex	it (h+'ex')	wor	All	baltom	Cat

Let's build some sets

- Remember that most nouns are owpys
- We use them by *comprehending* over a set
  - We go through each thing and decide if it “fits”
  - I.e. satisfies the predicate
- Examples
  - let alter. = $\{x : x \in \text{The} \wedge \text{Tiger}(x)\}$
  - let alter arad.

    $= \{y : y \in \{x : x \in \text{The} \wedge \text{Tiger}(x)\} \wedge \text{Big}(y)\}$

Relationships

- We need to talk about two things
  - “I go. Austin is a destination”
  - This implies no relationship.
- All owpys are defined as a transitively
  - alter [tiger]: **he** is a tiger of breed **ji**
  - arad [big]: **he** is bigger than **ji**
  - ɪkej [go]: **he** goes to **ji**

Transitive Comprension

- We now iterate over both iprid
- Only the elements of **ji** escape
- Example
  - tostɪn bab ɪŋkej.
  - {a : a ∈ Austin ∧ b ∈ Bob ∧ Go(b, a)}

- But then what's happening with intransitivity?

Contextualization

- All iprɪdend are considered in context
- That is, all resultant sets are intersected with the context of the listeners thoughts
- The context is also used to “fill-in-the-blanks”
  - let alter.
  - {x : x ∈ The ∧ c ∈ *Context* ∧ Tiger(x, c)}
- We abbreviate as so:
  - {x : x ∈ The ∧ Tiger(x, C)}

Example

- More definitions
  - iʃik [rock]: **he** is a rock of material **ji**
  - altom [cat]: **he** is a cat of species **ji**
- hym iʃik arad let altom ɪnkej.

$$\{ c : Go(e, c)\ m \wedge c \in \{b : b \in \{ a : a \in A \wedge Rock(a,C)\} \wedge arad(b,C) \}\} \wedge e \in \{ d : d \in The \wedge Cat(d) \}\}$$

Full Model

- Most owpys are just filters
- However, some more advance owpys are a functional relationship
- Example
  - ab [or]: The union of he and ji
- We map pairs of sets to pairs of sets
  - f : (S,S) → (S,S)

Full Model: Comprension

- We then note that owpys are then sets of tuples of tuples of sets
  - { ((S, S), (S, S)) }
- The actual application of an owpys looks like this:
  - joð ju ɪŋkej.
  - {x : t ∈ There ∧ u ∈ You ∧

    ∃U∃T∃Y∃X[((u, t), (Y, x)) ∈ Go ∧

    u ∈ U ∧ t ∈ T ∧ x∈X]}

Swapping Things Around

- joð wep ɪŋkej.
- (h, j) → (h', j')

- joð wep **ens** ɪŋkej.
- (h, j) → (h', j')

- wep joð **es** ɪŋkej.
- (j, h) → (j', h')

- wep joð **esens** ɪŋkej.
- (j, h) → (j', h')

Swapping Things Around

- let iʃik.
- (h, c) → (h', c')

- let **ens** iʃik.
- (h, c) → (h', c')

- let **es** iʃik.
- (j, c) → (j', c')

- let **esens** iʃik.
- (j, c) → (j', c')

Raised Phrases

- How do we refer to relationships themselves ?
- This is different from using the word or phrase
  - The idea of something; things that fit
- Similar to the “-ness” suffix in English

λ Functions

- Allows us to specify a single property
- These become aŋtah
- For a single word, prefix “eb”
  - eb altom → “catness”

- We encapsulate phrases with “yp lab ... up”
  - yp lab alter arad up → “big-tiger-ness”

λ Functions

- We can create these for relations too
- For a single word, use “eb” and transitive form
  - eb ɪnkej
  - The relation between a goer and a destination
- Phrases use “ymp ji he ... up”
  - ymp ji he iʃik ɪŋkej.
  - The relation between rock goers and the rock

λ Functions: aŋtah

- he, ji, and lab are actually special aŋtah
  - Hence can be used anywhere in a raising
  - Hence can be used in a principal sentence

- ji and lab can be omitted if the first word
- he if ji is omitted and is now first
  - yp alter arad up
  - ymp iʃik ɪŋkej up

λ Functions: Interpretation

- What are these as sets then ?
  - Exactly the same things as the functional interpretation of owpys
  - But now reified so we can manipulate them

- Note this is true of unary raisings too
  - Functions of the form f(x,x)=(y,y)

Adverbs With ilt and ejʒ

- We can then filter these like any other set
- We fill he using
  - ilt [apply]: apply raising **he** to **ji**
- We fill ji using
  - ejʒ [parameter]: **he** parameterizes **ji**
- Example:
  - joð eb ɪkej efɪs ilnt.
  - He quickly goes.

Defining Words

- We use the aʃov “ow” to quote words
- Then with the Davin definitions
  - aŋtah: **he** is a set word representing **ji**
  - owpys: **he** is a relation word **ji**
- We can natively define new words
  - ymp he elʃiɣ ji imfɪnt up ow elʃɪɣɪmf ens owpys.
  - [λ1 language λ2 make-T] 'conlang' is verb.

And Much More

- Lists
- Numbers
- Tense
- Questions
- Negation
- Logic

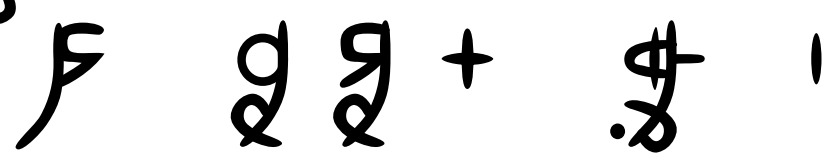
Intuition

- This knowledge is not necessary to speak
- It does explain some oddities
- There was effort to increase co-location.

Basic Steps

- We start by understanding set declarative semantics.
  - What, not how
- We filter this to specific meanings
- We filter more precisely with relations
- Noting that functions are sets
- We create a way to reify relations

Questions ?



hym epert ux.