Set Semantics
In Davin
What is Davin?

This is an example sentence written using Davin.

wep iprİdend ejgzamp eb es ejkir davın 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

- Isolational
- Strongly head final
- Ergative
- AOV SV S=O
Davin Basics: Phonology

<p>| | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>/p/</td>
<td>/φ/</td>
<td>/t/</td>
<td>/s/</td>
<td>/θ/</td>
<td>/ʃ/</td>
</tr>
<tr>
<td>/b/</td>
<td>/β/</td>
<td>/d/</td>
<td>/z/</td>
<td>/ð/</td>
<td>/ʒ/</td>
</tr>
<tr>
<td>/r/</td>
<td>/n/</td>
<td>/m/</td>
<td>/ŋ/</td>
<td>/l/</td>
<td>/h/</td>
</tr>
</tbody>
</table>

<p>| | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>/y/</td>
<td>/ɪ/</td>
<td>/i/</td>
<td>/a/</td>
<td>/e/</td>
<td>/o/</td>
</tr>
<tr>
<td>/u/</td>
<td>/w/</td>
<td>/l/</td>
<td>/h/</td>
<td>/j/</td>
<td>/w/</td>
</tr>
</tbody>
</table>
Davin Basics: Word Classes

- **aŋtah**
  - Pronouns, articles, and proper nouns
  - “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

<table>
<thead>
<tr>
<th>Group</th>
<th>T</th>
</tr>
</thead>
<tbody>
<tr>
<td>p b f v</td>
<td>m</td>
</tr>
<tr>
<td>t d s z</td>
<td>n</td>
</tr>
<tr>
<td>ɓ ɗ ʃ ʒ</td>
<td>n</td>
</tr>
<tr>
<td>k g x ɣ</td>
<td>ɳ</td>
</tr>
</tbody>
</table>
Davin Basics: iprid

(hym ifik) (let altom) Inkej.

((hym ifik) arad) (let altom) Inkej.

((hym ifik) arad)

(hym ifik) (let altom)
Davin Basics: aʃov

There are 10 grammar words called aʃov

- e څ
- ow ḳ
- es ḵ
- ens ṩ
- eb Ṱ
- ɪlp Ḷ
- owp ṳ
- yp ḻ
- ymp ṷ
- up ṹ
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's build some sets

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

<table>
<thead>
<tr>
<th>Pronouns</th>
<th>Articles</th>
<th>Names</th>
</tr>
</thead>
<tbody>
<tr>
<td>wit</td>
<td>I</td>
<td>zakari</td>
</tr>
<tr>
<td>wep</td>
<td>this</td>
<td>Austín</td>
</tr>
<tr>
<td>hex</td>
<td>it (h+'ex')</td>
<td>baltom</td>
</tr>
<tr>
<td></td>
<td>let</td>
<td>All</td>
</tr>
<tr>
<td></td>
<td>The</td>
<td>Cat</td>
</tr>
<tr>
<td></td>
<td>A / Some</td>
<td>Zachary</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
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} \land \text{Tiger}(x)\}
  - let alter arad.
    \[= \{y : y \in \{x : x \in \text{The} \land \text{Tiger}(x)\} \land \text{Big}(y)\}\]
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 іњкеj.
  - \{a : a \in Austin \land b \in Bob \land Go(b, a)\}

- But then what's happening with intransitivity?
Contextualization

• All iprİdend 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 \in \text{The} \land c \in \text{Context} \land \text{Tiger}(x, c)\}
• We abbreviate as so:
  – \{x : x \in \text{The} \land \text{Tiger}(x, C)\}
Example

- \textbf{More definitions}
  - ifik [rock]: \textbf{he is a rock of material ji}
  - altom [cat]: \textbf{he is a cat of species ji}
- hym ifik arad let altom ینهِک.

\[
\{ c : \text{Go}(e, c) \land c \in \{ b \\
  b \in \{ a : a \in A \land \text{Rock}(a,C) \} \land \text{arad}(b,C) \} \} \\
\land e \in \{ d : d \in \text{The} \land \text{Cat}(d) \} \}
\]
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
  - \( f : (S,S) \rightarrow (S,S) \)
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 ḷηkej.
  - \{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 ɬŋkej.
- \((h, j) \rightarrow (h', j')\)

- wep joð es ɬŋkej.
- \((j, h) \rightarrow (j', h')\)

- joð wep ens ɬŋkej.
- \((h, j) \rightarrow (h', j')\)

- wep joð esens ɬŋkej.
- \((j, h) \rightarrow (j', h')\)
Swapping Things Around

• let ṭijk.
• \((h, c) \rightarrow (h', c')\)

• let ens ṭijk.
• \((h, c) \rightarrow (h', c')\)

• let es ṭijk.
• \((j, c) \rightarrow (j', c')\)

• let esens ṭijk.
• \((j, c) \rightarrow (j', c')\)
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
\( \lambda \) Functions

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

- We encapsulate phrases with “yp lab ... up”
  - yp lab alter arad up \( \rightarrow \) “big-tiger-ness”
λ Functions

• We can create these for relations too

• For a single word, use “eb” and transitive form
  – eb ᵍⁿᵏᵉʲ
  – The relation between a goer and a destination

• Phrases use “ymp ji he ... up”
  – ymp ji he ʲⁱʲᵏ ᵏᵉʲᵏᵉʲ.
  – 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 ifik ʔŋkej 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 ılnıtı
  - 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.