Types for Ancient Greek

Experiments in Modeling Greek Grammar

Outline

1. Problem—What are we doing?

2. Dream—Where would we like to be?

3. Reality—Where are we now?

Texts

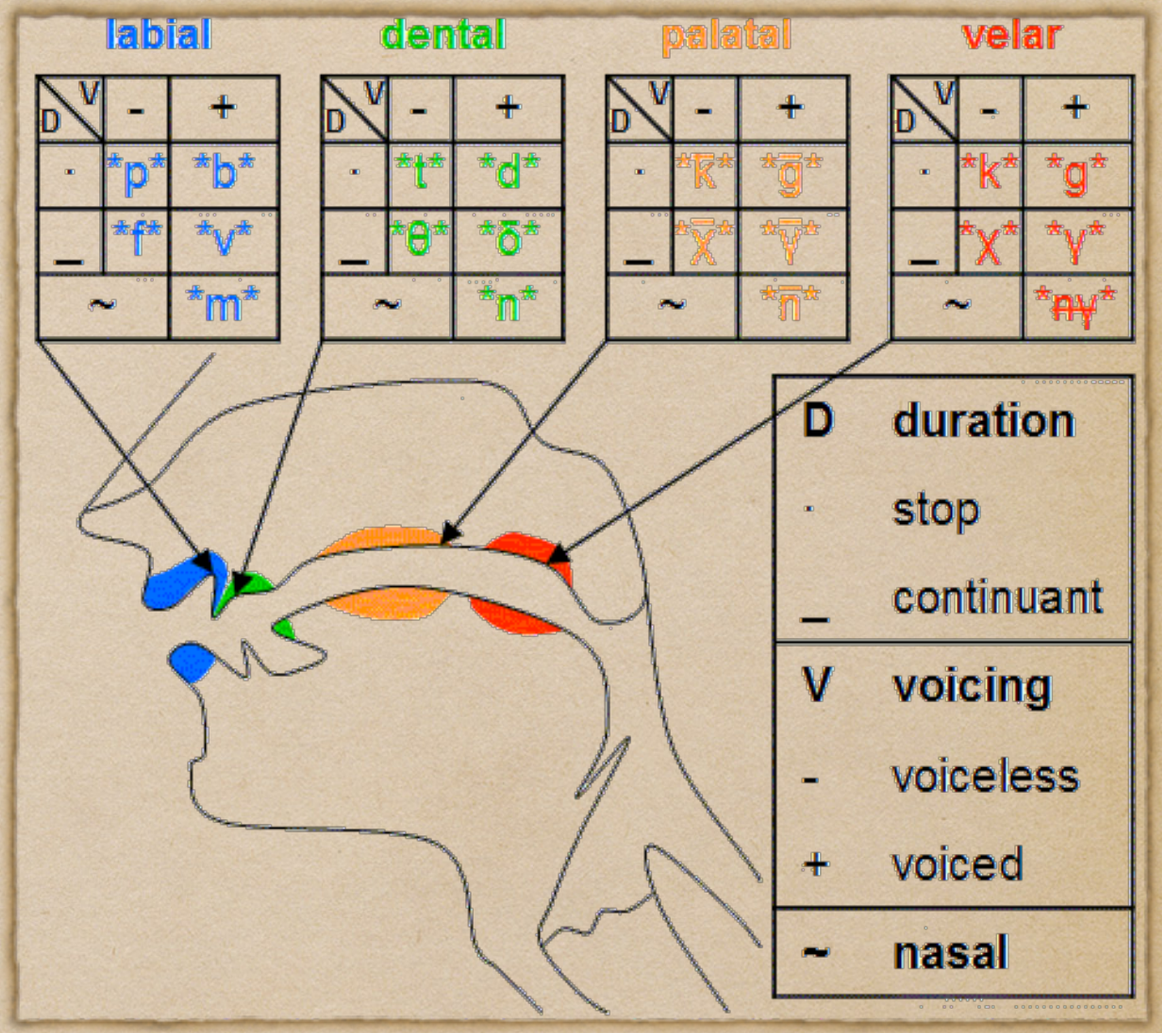
- Corpus linguistics
- Digital Texts (XML)
- Edited
- Transliterated

Analysis—Script

- Letters/Symbols
- Marks
  - Accents
  - Breathing Marks
- Even marks have meaning

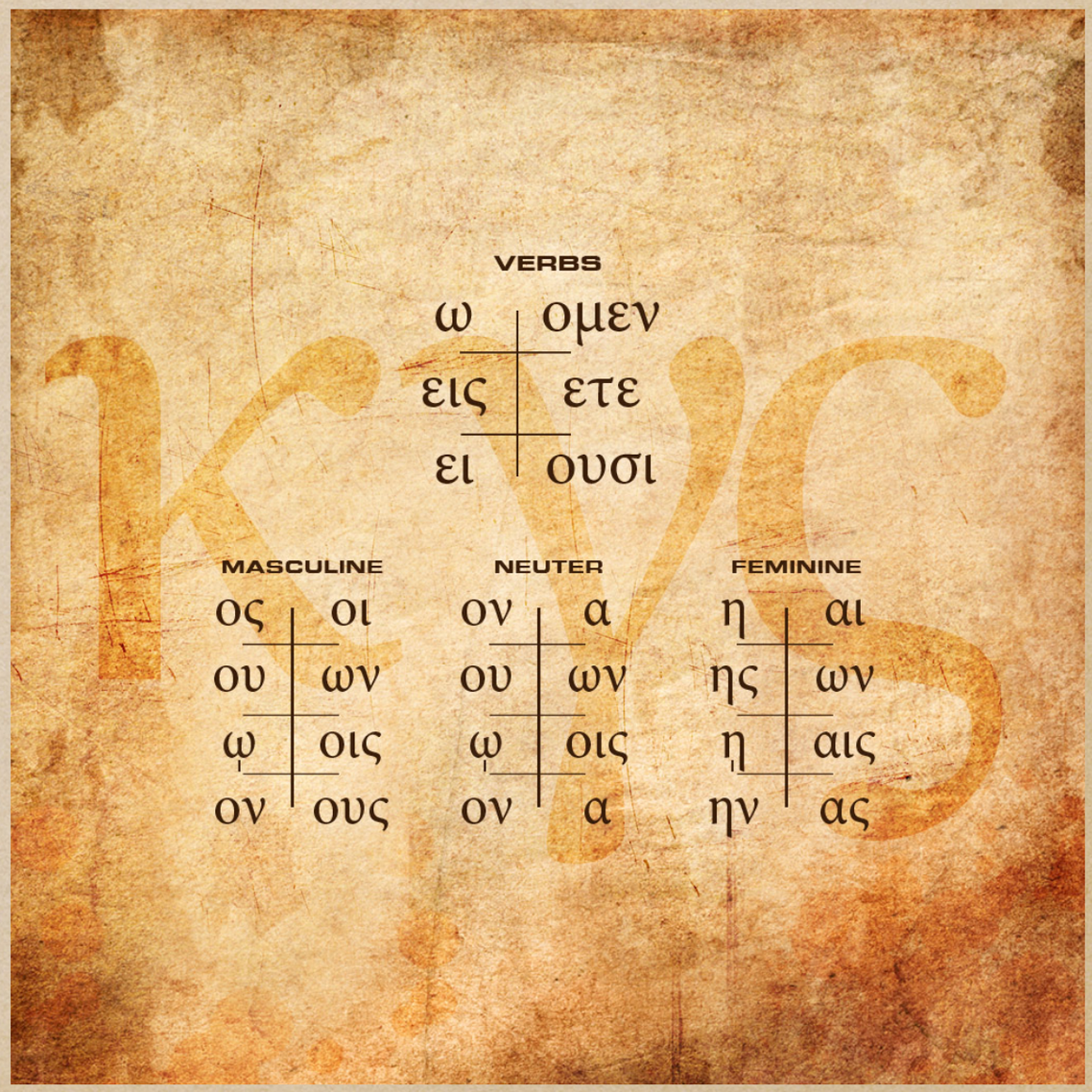
Analysis—Sounds

- Sounds
- Parts of mouth
- Syllables
- Change of sounds
- "Phonology"



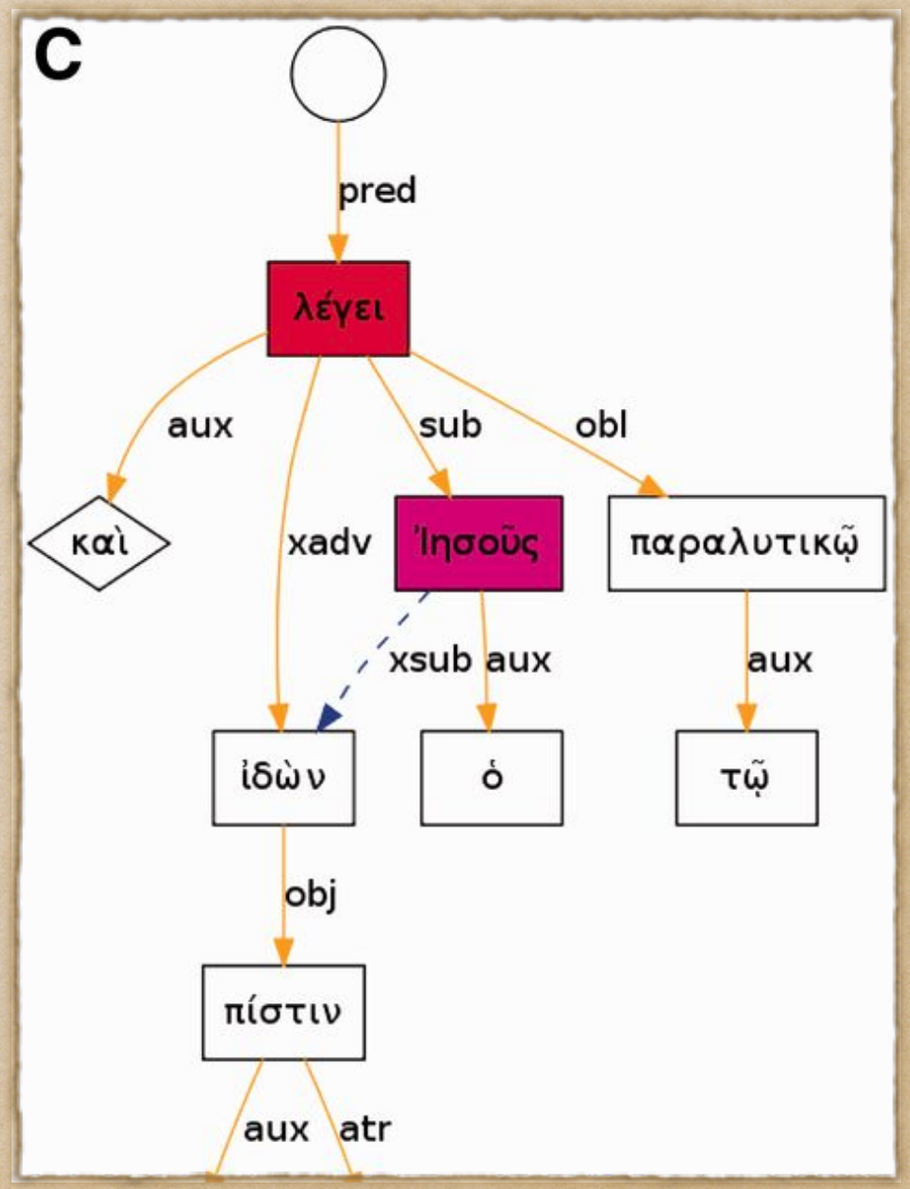
Analysis—Words

- Different forms of the same word
- Suffix/prefix patterns
- Spelling changes
- "Morphology"



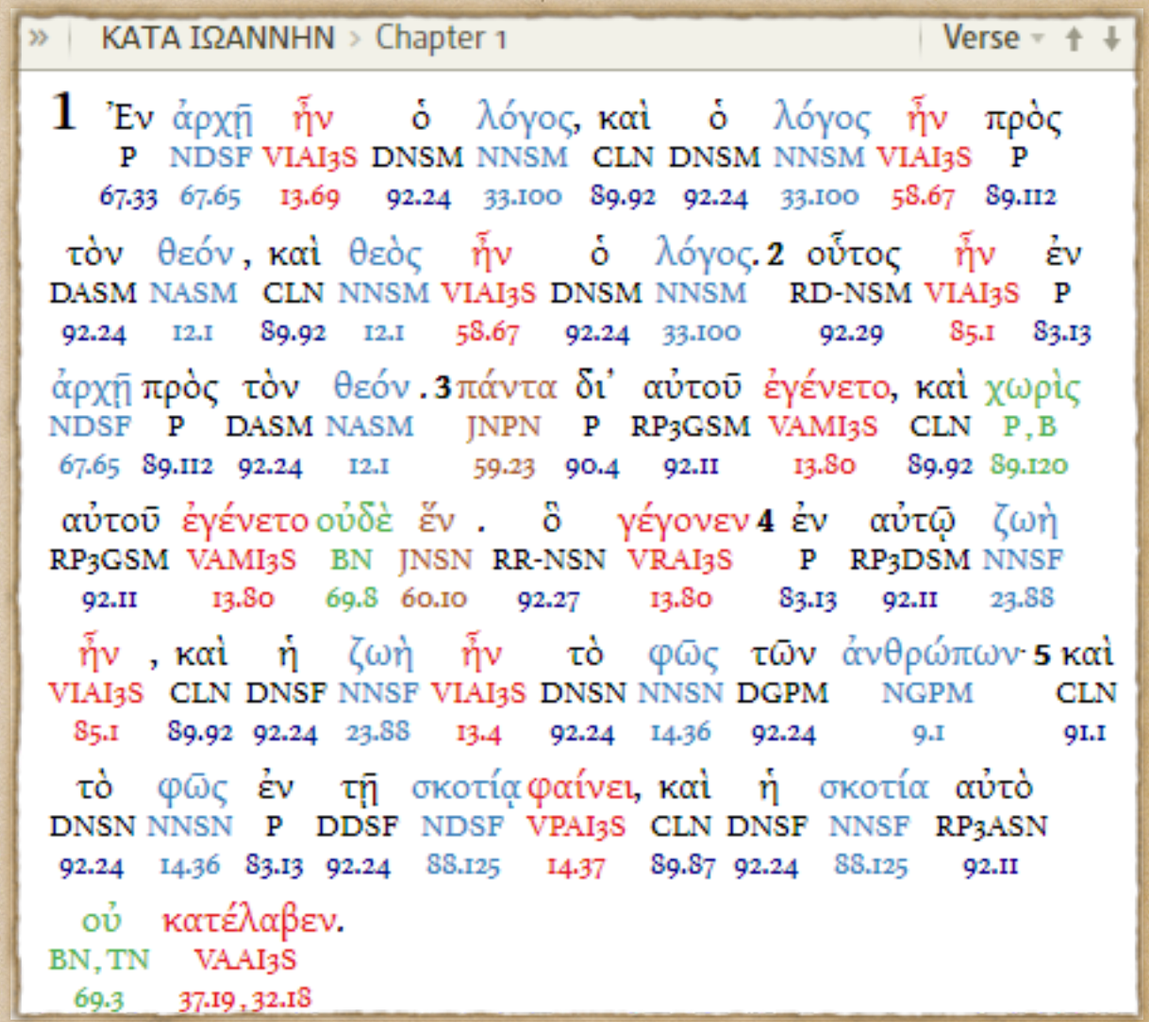
Analysis—Structure

- Syntax
- Semantics
- Discourse
- Pragmatics



Manual Annotation

- Word analysis—Months
- Structural analysis—Years
- Other Problems:
  - Consistency
  - Preconceived model
  - Expensive to change



» | ΚΑΤΑ ΙΩΑΝΝΗΝ › Chapter 1 | Verse

1 Ἐν ἀρχῇ ἦν ὁ λόγος, καὶ ὁ λόγος ἦν πρὸς
P NDSF VIAI3S DNSM NNSM CLN DNSM NNSM VIAI3S P
67.33 67.65 13.69 92.24 33.100 89.92 92.24 33.100 58.67 89.112

τὸν θεόν, καὶ θεὸς ἦν ὁ λόγος. **2** οὗτος ἦν ἐν
DASM NASM CLN NNSM VIAI3S DNSM NNSM RD-NSM VIAI3S P
92.24 12.1 89.92 12.1 58.67 92.24 33.100 92.29 85.1 83.13

ἀρχῇ πρὸς τὸν θεόν. **3** πάντα δι' αὐτοῦ ἐγένετο, καὶ χωρὶς
NDSF P DASM NASM JNPN P RP3GSM VAMI3S CLN P,B
67.65 89.112 92.24 12.1 59.23 90.4 92.11 13.80 89.92 89.120

αὐτοῦ ἐγένετο οὐδὲ ἕν. ὃ γέγονεν **4** ἐν αὐτῷ ζωὴ
RP3GSM VAMI3S BN JNSN RR-NSN VRAI3S P RP3DSM NNSF
92.11 13.80 69.8 60.10 92.27 13.80 83.13 92.11 23.88

ἦν, καὶ ἡ ζωὴ ἦν τὸ φῶς τῶν ἀνθρώπων· **5** καὶ
VIAI3S CLN DNSF NNSF VIAI3S DNSN NNSN DGPM NGPM CLN
85.1 89.92 92.24 23.88 13.4 92.24 14.36 92.24 9.1 91.1

τὸ φῶς ἐν τῇ σκοτίᾳ φαίνει, καὶ ἡ σκοτία αὐτὸ
DNSN NNSN P DDSF NDSF VPAI3S CLN DNSF NNSF RP3ASN
92.24 14.36 83.13 92.24 88.125 14.37 89.87 92.24 88.125 92.11

οὐ κατέλαβεν.
BN,TN VAAI3S
69.3 37.19,32.18

Patterns

- Express patterns in a precise language
- Computer finds where those patterns occur

Human-Directed

- A human selects the pattern
- Goals
  - More precision and consistency than manual annotation
  - Annotation takes less time

Experiment—2013 Hackathon

- Use C# to create Role and Reference Grammar Constituency Trees
- Use existing morphology, syntax databases
- Problems:
  - C#—Not a good fit for tree transformations
  - Undoing choices of existing data—e.g. adverb bucket

Experiment—
Learn Functional Programming

- Standard ML, F#, Haskell
- Coq, Idris, Agda
- Result—Fun & Profit

Existing Grammars

- Work done by hand over many years
- Wealth of information—letters, sounds, words, syntax, etc.
- Word & structure depend on sounds/script
- What appears irregular may have a clearer pattern at the sound/script level

Grammars vs Data

• Script	✓	✗
• Sounds	✓	✗
• Words	✓	✓
• Structure	✓	✓

Experiment—Greek Script in Agda

- Goal—precisely encode Greek script
- Dependent types—high degree of precision
- Steep learning curve
- Result—Successful but slow to type check

Agda Code as Data

- Generate Agda files from XML
- Dependently typed data!
- Data available for type checking
- Can write a proof of completeness for a text

```
module Text.Greek.SBLGNT.John where

open import Data.List
open import Text.Greek.Bible
open import Text.Greek.Script
open import Text.Greek.Script.Unicode

KATA-ΙΩΑΝΝΗΝ : List (Word)
KATA-ΙΩΑΝΝΗΝ =
    word (Ἐ :: ν :: []) "John.1.1"
  :: word (ά :: ρ :: χ :: ῆ :: []) "John.1.
  :: word (ῆ̃ :: ν :: []) "John.1.1"
  :: word (ὁ :: []) "John.1.1"
  :: word (•λ :: ό :: γ :: ο :: ς :: []) "Jo
```

Experiment—2015 Hackathon

- Use Haskell to encode sound transformations
- Types forced us to resolve confusing aspects of the grammatical rules—debugging our brains

Experiment—SBL 2015

- Presented at Society of Biblical Literature (SBL)
- Start with Greek texts, apply each level of analysis—script, sounds, etc.
- Haskell for data transformation
- Serialize data to JSON
- View results in a web browser (ES6/React)

Tame the Complexity

- Instead of one gigantic, complex parser…
- Use a series of minute parsers

Lenses, Prisms, Traversals

- Compose to focus on structure to change
- Each transformation:
  - Changes the types
  - Encodes information
  - Reduces cardinality

Letter to Vowel/Consonant

listWorks : [WorkProps × [WordProps × [Letter]]]

f : Letter -> Vowel + Consonant

path = traverse . _2 . traverse . _2 . traverse

over path f listWorks :

[WorkProps × [WordProps × [Vowel + Consonant]]]

Letter to Vowel/Consonant

listWorks : [WorkProps × [WordProps × [Letter]]]

f : Letter -> Vowel + Consonant

path = traverse . _2 . traverse . _2 . traverse

over path f listWorks :

[WorkProps × [WordProps × [Vowel + Consonant]]]

Letter to Vowel/Consonant

listWorks : [WorkProps × [WordProps × [Letter]]]

f : Letter -> Vowel + Consonant

path = traverse . _2 . traverse . _2 . traverse

over path f listWorks :

[WorkProps × [WordProps × [Vowel + Consonant]]]

Letter to Vowel/Consonant

listWorks : [WorkProps × [WordProps × [A]]]

f : A -> Maybe B

path = traverse . _2 . traverse . _2 . traverse

path f listWorks :

Maybe [WorkProps × [WordProps × [B]]]

Script Analysis

- Unicode (composed, decomposed)
- Letter case, final form sigma
- Marks:
  - Letter/mark combinations
  - Accents, breathing and syllabic identifiers
- Vowels and consonants
- Syllables

Changing Types

[WorkProps × [WordProps × [Char]]]

→

Maybe

[WorkProps × [WordProps × [ConcreteLetter + Mark]]]

f : Char -> Maybe (ConcreteLetter + Mark)

Changing Types

[WorkProps × [WordProps × [ConcreteLetter + Mark]]]

→

Maybe

[WorkProps × [WordProps × [ConcreteLetter × [Mark]]]]

f : [ConcreteLetter + Mark] -> Maybe [ConcreteLetter × [Mark]]

Changing Types

[WorkProps × [WordProps × [ConcreteLetter × [Mark]]]

→

[WorkProps × [WordProps × [AbstractLetter × Case × Final × [Mark]]]

f : ConcreteLetter -> AbstractLetter × Case × Final

Changing Types

[WorkProps × [WordProps × [AbstractLetter × Case × Final × [Mark]]]

→

Maybe

[WorkProps × [WordProps × Caps × [AbstractLetter × Final × [Mark]]]

f : a × [Case × b] -> Maybe (a × Caps × [b])

Changing Types

[WorkProps × [WordProps × Caps × [AbstractLetter × Final × [Mark]]]]

→

Maybe

[WorkProps × [WordProps × Caps × [AbstractLetter × [Mark]]]]

f : [Final × b] -> Maybe [b]

Changing Types

[WorkProps × [WordProps × Caps × [AbstractLetter × [Mark]]]

→

[WorkProps × [WordProps × Caps × [(Vowel + Consonant) × [Mark]]]

f : AbstractLetter -> Vowel + Consonant

Viewing the Analyses

- Works—View the words of the text in order and the analyses attached to each word
- Types—View the analyses and all the places where each analysis occurs
- https://scott-fleischman.github.io/greek-grammar/

Desirable System Features

- Discovery—Ask questions about data
- Exhaustive—Account for all data
- Precision—Model omits nonsense
- Browsability
  - View the analyses in detail
  - Ask questions about results

Desirable System Features

- Discovery—Ask questions about data
- Exhaustive—Account for all data
- Precision—Model omits nonsense
- Browsability
  - View the analyses in detail
  - Ask questions about results

Where to go from here?

- Agda: Prove equivalences between transitions

$$(f : A \to B)\ (f^{-1} : B \to A)\ (x : B) \to x \equiv f\ (f^{-1}\ x)$$

- Include more data—Perseus Digital Library
- Deepen analysis—Sounds, Words, Structure
- Improve serialization

Related Work

- Functional Morphology
  Forsberg, Ranta—ICFP 2004
  http://dl.acm.org/citation.cfm?id=1016879

- Grammatical Framework
  http://www.grammaticalframework.org/

https://github.com/scott-fleischman/greek-grammar