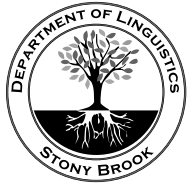
Linguistic Case and Subregular Constraints Over Trees and Paths

Kenneth Hanson
kenneth.hanson@stonybrook.edu



July 11, 2022

Overview

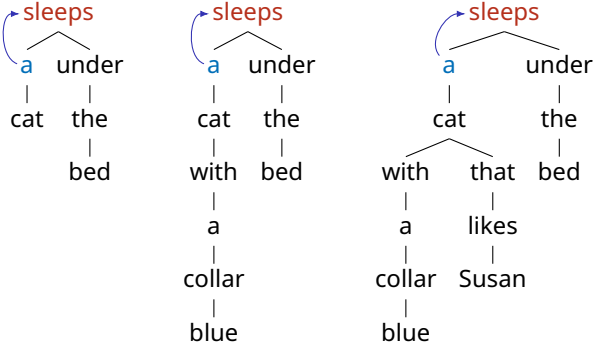
- Exploring issues of data representation and computational complexity
- Examining the syntactic distribution of linguistic case marking
  - cf. Baker 2015; Malchukov and Spencer 2008; Butt 2006; Blake 2001
  - ex. ✓ *He met her* ✗ *He met she* ✗ *Him met her* ✗ *Him met she*
  - Hypothesis: case patterns are tier-based strictly local (TSL)
    (cf. Heinz et al. 2011; Vu et al. 2019)
  - Two possible representations: trees and path strings
- This involves:
  - Systematic cross-linguistic data collection
  - Syntactic annotation based on the theoretical literature
  - Developing software to rigorously test possible analyses

Computational complexity depends on representation

[A cat] sleeps under the bed.

[A cat [with a blue collar]] sleeps under the bed.

[A cat [with a blue collar] [that Susan likes]] sleeps under the bed.



What is case?

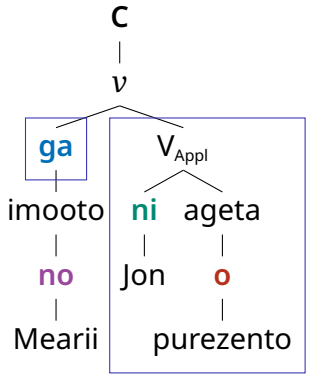
Case refers to a broad range of markings on nouns. We are interested in case markers which reflect the syntactic context in which a noun appears.

In Japanese:

- **ga** (nominative) – subjects
- **o** (accusative) – direct objects
- **ni** (dative) – indirect objects
- **no** (genitive) – modifiers of nouns

ex. *Mearii no imooto ga Jon ni purezento o ageta.*
Mary GEN sister NOM John DAT present ACC gave
'Mary's sister gave John a present.'

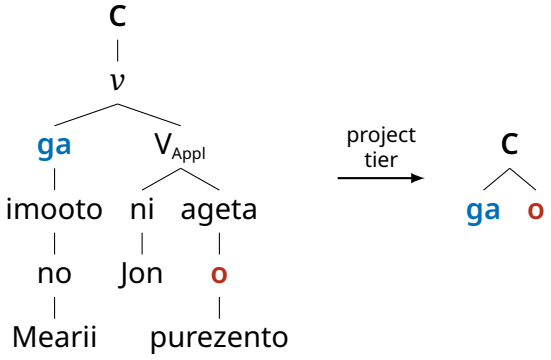
Syntactic constraints on case



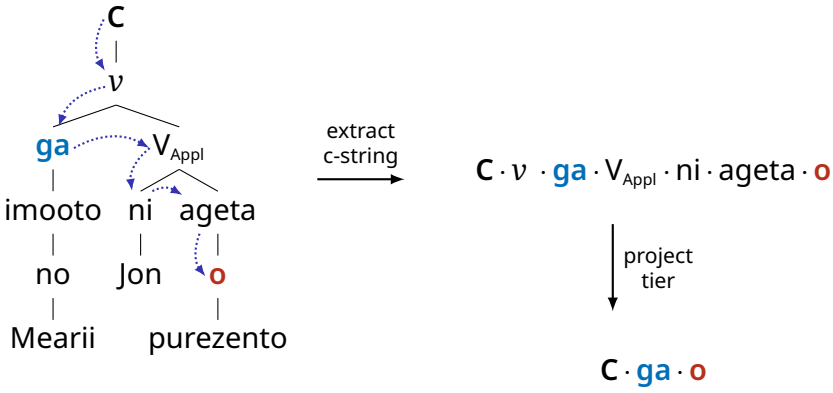
Example constraint: the case marker **o** can only occur if there is a **ga** which c-commands it in the same clause.

Hypothesis: case patterns are TSL

Basic intuition: certain long-distance dependencies become local when irrelevant material is removed.



TSL with c-strings



Trees and paths: linguistic considerations

It appears that we have two viable representations: tree tiers and c-string tiers. But is this really the case?

- Hanson (2022) provides a TSL analysis of Japanese case using tree tiers, and c-string tiers appear to work as well.
- It is an open question whether all case patterns can be analyzed as TSL over trees or c-strings (or both, or neither).
- Answering this question can help explain why natural languages look as they do (Lambert et al. 2021; Graf 2022).

Trees and paths: computational considerations

Strings are easier to work with than trees.

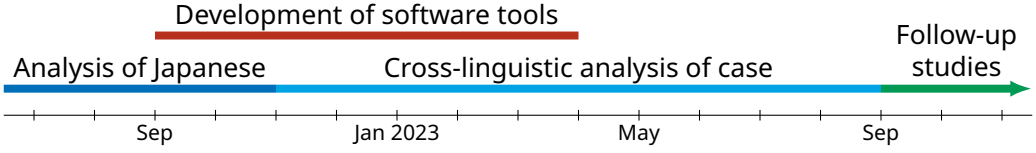
- The mathematics of strings is much better understood (cf. Lambert 2022).
- C-string dependencies can be evaluated and parsed easily (Graf and Shafiei 2019).
- TSL over strings can be learned algorithmically (Jardine and McMullin 2017).

If tree tiers turn out to be necessary for many empirical phenomena, this highlights the need for more computational research on TSL (and similar classes) over trees.

Project components

1. Build a database of case patterns with example sentences and syntactic annotation
2. Extend SigmaPie (Aksënova 2020) to handle TSL over trees and c-strings
3. Implement and test TSL grammars using (1) and (2)
4. How can insights about trees and path strings be extended to other fields?

Next steps



Planned languages to analyze include: Basque, Persian, Tamil, Warlpiri

Possible follow-up studies:

- Additional in-depth analyses like that for Japanese
- Explore alternatives to TSL (MTSL, IO-TSL)
- Conduct a similar cross-linguistic study of *agreement*

Summary

- Computational complexity depends on choice of representation.
- Linguistic case provides an excellent opportunity to explore the differences between tree and path representations.
  - Determining whether case patterns are TSL over trees and/or c-strings will improve our understanding of language.
  - The findings of this study may motivate further computational research and inform applications in other fields.

References

Aksënova, Alëna (2020). *SigmaPie*. URL: https://github.com/alenaks/SigmaPie.

Baker, Mark (2015). *Case*. Cambridge University Press. DOI: 10.1145/1463891.1464000.

Blake, Barry J. (2001). *Case*. Cambridge University Press.

Butt, Miriam (2006). *Theories of case*. Cambridge University Press.

Graf, Thomas (2022). "Typological Implications of Tier-Based Strictly Local Movement". In: *Proceedings of the Society for Computation in Linguistics 2022*, pp. 184–193.

Graf, Thomas and Nazila Shafiei (2019). "C-command dependencies as TSL string constraints". In: *Proceedings of the Society for Computation in Linguistics (SCiL) 2019*, pp. 205–215.

Hanson, Kenneth (2022). "A TSL Analysis of Japanese Case". URL: https://khanson679.github.io/files/hanson-jpn-case-draft.pdf.

Heinz, Jeffrey et al. (2011). "Tier-based strictly local constraints for phonology". In: *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human language technologies*, pp. 58–64.

Jardine, Adam and Kevin McMullin (2017). "Efficient learning of tier-based strictly k-local languages". In: *International conference on language and automata theory and applications*. Springer, pp. 64–76.

Lambert, Dakotah (2022). "Unifying Classification Schemes for Languages and Processes With Attention to Locality and Relativizations Thereof". PhD thesis. Stony Brook University.

Lambert, Dakotah et al. (2021). "Typology emerges from simplicity in representations and learning". In: *Journal of Language Modelling* 9.1, pp. 151–194.

References (2)

Malchukov, Andrej and Andrew Spencer (2008). *The handbook of case*. Oxford: Oxford University Press.

Vu, Mai Ha et al. (2019). “Case assignment in TSL syntax: A case study”. In: *Proceedings of the Society for Computation in Linguistics (SCiL) 2019*, pp. 267–276.

Case patterns in English

English examples:

(1) She arrived.
(2) He knows her.
(3) She is known by everyone.
(4) I believe [him to know her].
(5) [Him arriving late] annoyed us.
(6) [His late arrival] annoyed us.
(7) [His arriving late] annoyed us.

I/you/he/she = nominative
me/you/him/her = accusative
my/your/his/her = genitive

Case patterns in Japanese

The basic pattern:

- Sbj_{NOM} V
- Sbj_{NOM} Obj_{ACC} V
- Sbj_{NOM} $Obj1_{DAT}$ $Obj2_{ACC}$ V

Nominative objects:

- Sbj_{NOM} Obj_{NOM} V ‘He likes she.’
- Sbj_{DAT} Obj_{NOM} V

“Exceptional case marking”:

- Sbj_{ACC} V ‘We think that [them are mean].’
- Sbj_{GEN} V ‘I know the reason that [his came].’

The ergative pattern

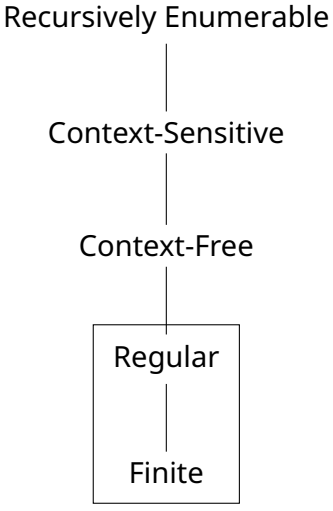
The basic pattern:

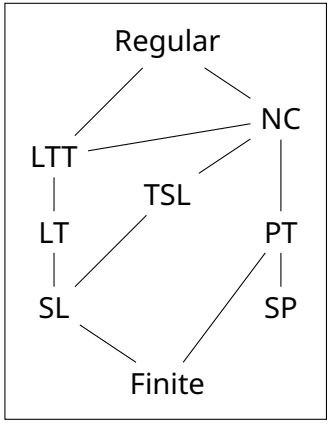
- Sbj_{Abs} V — ‘Him slept.’
- Sbj_{Erg} V Obj_{Abs} — ‘He met her.’

Split ergativity (use of ergative only in some contexts):

- Accusative system if the tense is present, ergative if the tense is past (Persian)
- Accusative system for clitic pronouns, ergative for nouns and full pronouns (Warlpiri)

The subregular hierarchy





TSL in more detail

Tree TSL

- Project a tree tier
- For each node in the tier, looking at its string of daughters:
  - Project another tier over this string (if applicable)
  - Enforce a strictly local grammar

TSL over c-strings

- For each node:
  - Extract a c-string for each node
  - Project a tier according to the label of the node
  - Enforce a strictly local grammar

The importance of hierarchical structure

Long-distance dependencies in syntax are not TSL over surface strings.

✓ [A cat [that some children like]] sleeps under the bed.

✗ [A cat [that some children likes]] sleep under the bed.

Project just singular items ⟶ don't know where the items came from

- A · sleeps
- A · likes

Project all items ⟶ the pattern on the tier is not strictly local

- A · some · like · sleeps
- A · some · likes · sleep

Evaluating/parsing TSL with c-strings

C-string dependencies can be evaluated top-down, deterministically, with finite look-ahead (Graf and Shafiei 2019).