Computational Lessons from and for Language

Thomas Graf

Department of Linguistics
mail@thomasgraf.net
http://thomasgraf.net

IACS Seminar
Apr 28 2016

The Gretchenfrage

The Gretchenfrage

What role can linguistics play in the computational sciences?

Three Take Home Messages

1. Language is a fundamentally computational problem.
   cognitive turn, mental grammars, structure inference
2. Linguistics is a creator of computational results.
   subregular maps, finite-state decompositions
3. Linguistics is a consumer of computational results.
   tensor spaces, game theory, . . .

Outline

1. Language as a Computational Problem

2. Results from Computational Linguistics
   - Some problems aren't as complex as you might think
   - And complex problems are simple problems in disguise

3. Computational Problems in Linguistics
   - MSO-FSA conversion
   - Parallel parsing
   - Tensor space semantics
   - Game-Theoretic Pragmatics

Language as Part of Human Cognition

There's many views on language:

- cultural artifact
- a fixed system of preordained rules ("proper grammar")
- communication system
- system of signs

⋮

Cognitive Turn (Chomsky 1957, 1965)

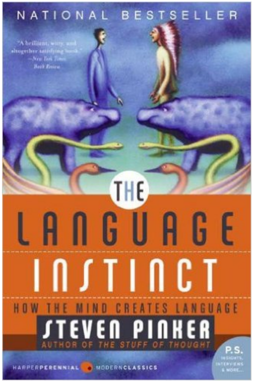
Language is an aspect of human cognition.

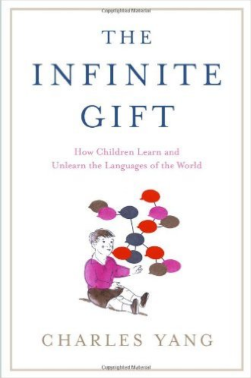
Computational Questions

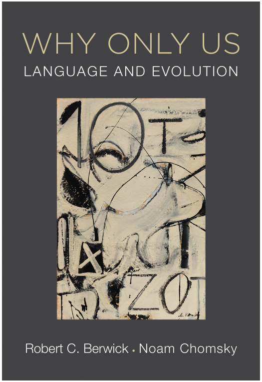
- What kind of representations are involved?
  strings, trees, graphs, hypergraphs, ...
- How can they be manipulated?
  subtree substitution, graph transductions, ...
- How can this domain knowledge be acquired from input data?
  Gold learning, PAC learning, MAT learning, ...
- How do speakers use their knowledge about a language in real-time listening and comprehension?
  recursive descent parsing, CKY, Earley, ...

⋮

If You Want to Know More. . .







Historical Role of Linguistics in Computational Sciences

- Foundations of formal language theory
  Chomsky (1956, 1959); Chomsky and Schützenberger (1963)
- Equivalence of CSGs and linear bounded automata
  Kuroda (1964)
- Initial motivation for tree transducers
  Rounds (1970)
- First parsing algorithms
  Yngve (1955)
- First string processing language (COMIT)
  Yngve (1958)

But What Have You Recently Done for Me?

Lesson 1 Some problems are hard because the model is wrong.

Lesson 2 And some complex problems are just simple problems in disguise.

Phonological Patterns

Phonology system regulating the distribution of sounds in words

Example

word-final devoicing	rat	*rad	*German*
intervocalic voicing	nevið	*nefið	*Icelandic*
vowel harmony	kauralla	*kaurella	*Finnish*
sibilant harmony	tsaanéέz	*tʃaanéέz	*Navajo*
umlaut	mömmu	*mammu	*Icelandic*
dissimilation	lunaris	*lunalis	*Latin*

Computational Status Quo

Every known phonological system defines a regular string language.

Finite-State Automata

A **finite-state automaton** (FSA) assigns every node in a string one of finitely many *states*, depending on

- the label of the node, and
- the state of the preceding node (if it exists).

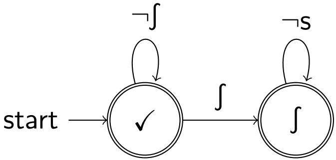
The FSA accepts the string if the last state is a *final state*.

Cognitive Intuition

- States are a metaphor for memory configurations.
- Every symbol in the input induces a change from one memory configuration into another.
- Only finitely many memory configurations are needed.
  Thus the amount of working memory used by the automaton is finitely bounded.

Example: Sibilant Harmony

Condition: ʃ cannot be followed by s
Memory: 2 distinct states ✓ and ʃ

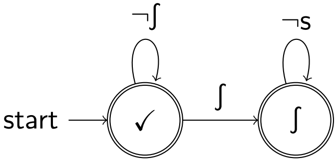


ʃ e g o ʃ a

g e ʃ o s a

Example: Sibilant Harmony

Condition: ʃ cannot be followed by s
Memory: 2 distinct states ✓ and ʃ

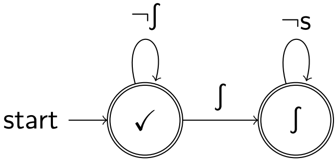


ʃ e g o ʃ a

g e ʃ o s a

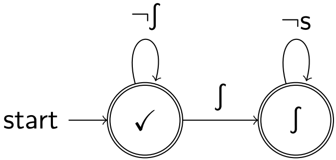
Example: Sibilant Harmony

Condition: ʃ cannot be followed by s
Memory: 2 distinct states ✓ and ʃ



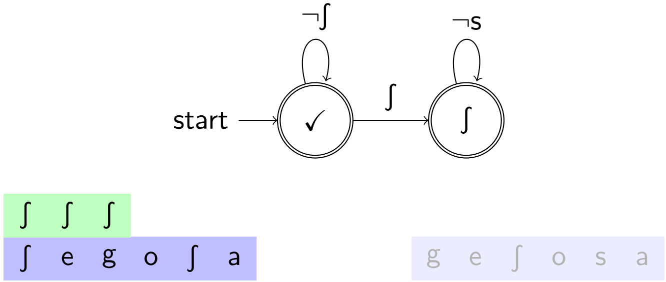
Example: Sibilant Harmony

Condition: ʃ cannot be followed by s
Memory: 2 distinct states ✓ and ʃ



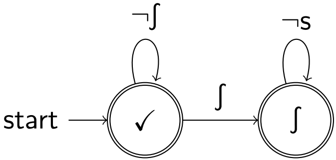
Example: Sibilant Harmony

Condition: ʃ cannot be followed by s
Memory: 2 distinct states ✓ and ʃ



Example: Sibilant Harmony

Condition: ʃ cannot be followed by s
Memory: 2 distinct states ✓ and ʃ



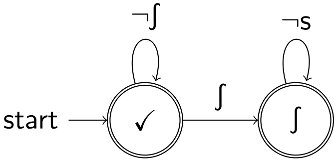
ʃ ʃ ʃ ʃ

ʃ e g o ʃ a

g e ʃ o s a

Example: Sibilant Harmony

Condition: ʃ cannot be followed by s
Memory: 2 distinct states ✓ and ʃ

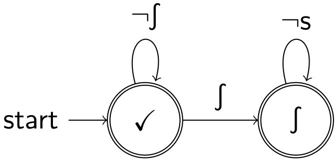


ʃ ʃ ʃ ʃ ʃ
ʃ e g o ʃ a

g e ʃ o s a

Example: Sibilant Harmony

Condition: ʃ cannot be followed by s
Memory: 2 distinct states ✓ and ʃ



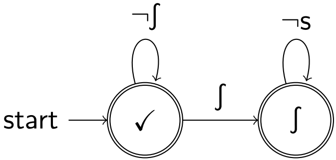
ʃ	ʃ	ʃ	ʃ	ʃ	ʃ
ʃ	e	g	o	ʃ	a

g	e	ʃ	o	s	a

Example: Sibilant Harmony

Condition: ʃ cannot be followed by s

Memory: 2 distinct states ✓ and ʃ

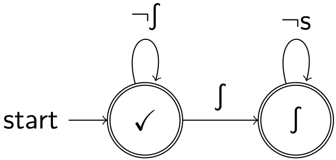


ʃ	ʃ	ʃ	ʃ	ʃ	ʃ
ʃ	e	g	o	ʃ	a

g	e	ʃ	o	s	a

Example: Sibilant Harmony

Condition: ʃ cannot be followed by s
Memory: 2 distinct states ✓ and ʃ

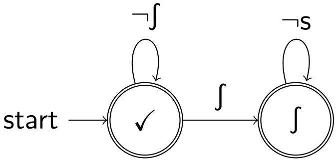


ʃ	ʃ	ʃ	ʃ	ʃ	ʃ
ʃ	e	g	o	ʃ	a

✓					
g	e	ʃ	o	s	a

Example: Sibilant Harmony

Condition: ʃ cannot be followed by s
Memory: 2 distinct states ✓ and ʃ

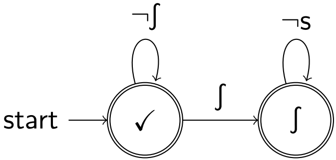


ʃ	ʃ	ʃ	ʃ	ʃ	ʃ
ʃ	e	g	o	ʃ	a

✓	✓				
g	e	ʃ	o	s	a

Example: Sibilant Harmony

Condition: ʃ cannot be followed by s
Memory: 2 distinct states ✓ and ʃ

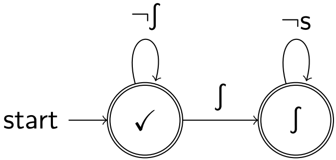


ʃ	ʃ	ʃ	ʃ	ʃ	ʃ
ʃ	e	g	o	ʃ	a

✓	✓	ʃ			
g	e	ʃ	o	s	a

Example: Sibilant Harmony

Condition: ʃ cannot be followed by s
Memory: 2 distinct states ✓ and ʃ



ʃ	ʃ	ʃ	ʃ	ʃ	ʃ
ʃ	e	g	o	ʃ	a

✓	✓	ʃ	ʃ		
g	e	ʃ	o	s	a

Subregular Phonology

Regular languages are computationally appealing, but

- closure properties do not reflect typology
  union of two phonological systems is not a phonological system
- there is no known learning algorithm for the full class of regular languages

Subregular Hypothesis (Heinz 2010)

Phonological systems define **subregular** string languages.

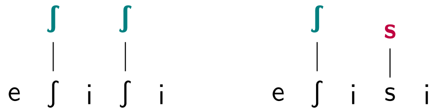
Subregular Hierarchy

- Many subregular classes were established a long time, even though they have largely been ignored. (McNaughton and Pappert 1971)
- Most of them aren't suitable for phonology, so linguists had to find **new subregular classes**:

strictly piecewise	(Rogers et al. 2010)
interval-based strictly piecewise	(Graf under review)
tier-based strictly local	(Heinz et al. 2011)

Example: Sibilant Harmony is Tier-Based Strictly Local

- For every string, induce a substructure containing only s and ʃ
- Induced substructure may not contain the bigram ʃs



Advantages of the Subregular Classes

In contrast to regular languages, these new subregular classes

- have more suitable closure properties,
- are efficiently learnable in the limit from positive text, (Heinz et al. 2012)
- share a lattice-structured grammar space.

They also have applications outside of language, e.g. robotics. (Chandlee et al. 2012)

First Take-Home Message

If you're currently working with regular languages,
one of the weaker classes we have identified may suffice.

Advantages of the Subregular Classes

In contrast to regular languages, these new subregular classes

- have more suitable closure properties,
- are efficiently learnable in the limit from positive text, (Heinz et al. 2012)
- share a lattice-structured grammar space.

They also have applications outside of language, e.g. robotics. (Chandlee et al. 2012)

First Take-Home Message

If you're currently working with regular languages,
one of the weaker classes we have identified may suffice.

Factorizing Syntax

syntax system regulating the distribution of words in sentences

(1) Who did you say John likes?

(2) Who did you say that John likes?

(3) Who did you say likes John?

(4) * Who did you say that likes John?

Syntax is Very, Very Complex

When viewed as string languages, natural languages are parallel multiple context-free languages (PMCFL; Kobele 2006).

$$\text{REG} \subset \text{DCFL} \subset \text{CFL} \subset \text{TAL} \subset \text{MCFL} \subset \text{PMCFL} \subset \text{CSL}$$

Factorizing Syntax

syntax system regulating the distribution of words in sentences

(1) Who did you say John likes?

(2) Who did you say that John likes?

(3) Who did you say likes John?

(4) * Who did you say that likes John?

Syntax is Very, Very Complex

When viewed as string languages, natural languages are parallel multiple context-free languages (PMCFL; Kobele 2006).

$$\text{REG} \subset \text{DCFL} \subset \text{CFL} \subset \text{TAL} \subset \text{MCFL} \subset \text{PMCFL} \subset \text{CSL}$$

The Trouble with PMCFLs

PMCFLs have many downsides:

- complex formalism, hard to reason about
- hard recognition problem (although PTIME)
- no useful learnability results

The Cognitive Conundrum

If PMCFLs are an accurate model of natural language syntax, then how come humans

- learn syntax easily from little data, and
- can produce and understand sentences in real-time?

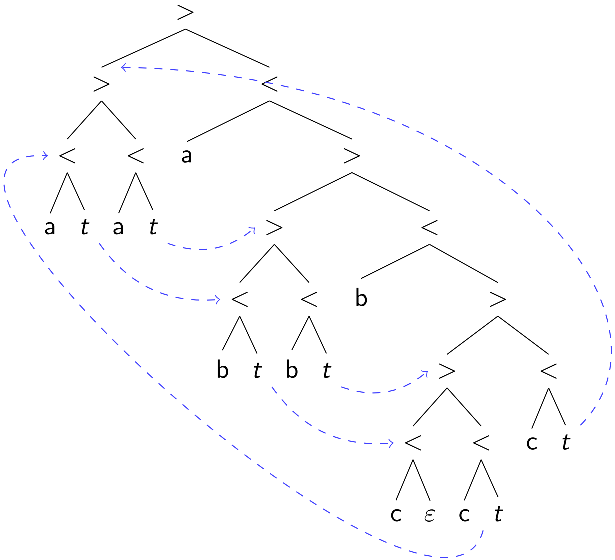
A Modular Solution

- Syntax is actually much simpler than it seems.
- The complexity arises from the interaction of **two finite-state components**:

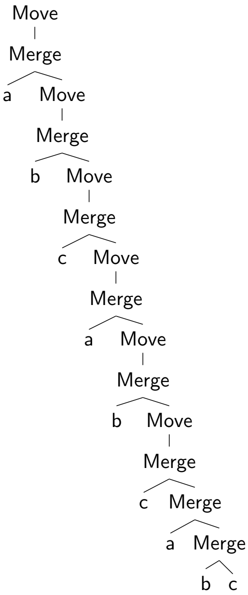
  Derivations — a set of abstract structures generated by a regular tree grammar ($\approx$ CFG)

  Interpretation — a macro tree transducer that constructs the pronounced strings from the derivations
- Since the interpretation is fixed across languages, syntax can be reduced to regular tree grammars/CFGs.

Example: Output Tree



Example: Much Simpler Derivation



One of Many Application: Transderivational Constraints

- Transderivational constraints are optimization constraints: structure X is well-formed only if there is no better choice Y.
- These were believed to be intractable for syntax.
- **But:** actually linear tree transductions on derivations $\Rightarrow$ computable in linear time! (Graf 2013)

Second Take-Home Message

- Decomposition/Modularization simplifies complex systems.
- Simpler representations allow for efficient implementations.

One of Many Application: Transderivational Constraints

- Transderivational constraints are optimization constraints: structure X is well-formed only if there is no better choice Y.
- These were believed to be intractable for syntax.
- **But:** actually linear tree transductions on derivations
  $\Rightarrow$ computable in linear time! (Graf 2013)

Second Take-Home Message

- Decomposition/Modularization simplifies complex systems.
- Simpler representations allow for efficient implementations.

Dealing With Syntactic Constraints

Theorem (Graf 2011)

Every syntactic constraint can be precompiled into the grammar.

1. Every syntactic constraint can be expressed as a formula of **monadic second-order logic** (MSO).
2. Every MSO formula can be converted into an equivalent bottom-up tree automaton.
3. Every bottom-up tree automaton can be precompiled into the grammar.

Question

Is there a significant blow-up in the size of the grammar?

Dealing With Syntactic Constraints

Theorem (Graf 2011)

Every syntactic constraint can be precompiled into the grammar.

1. Every syntactic constraint can be expressed as a formula of **monadic second-order logic** (MSO).
2. Every MSO formula can be converted into an equivalent bottom-up tree automaton.
3. Every bottom-up tree automaton can be precompiled into the grammar.

Question

Is there a significant blow-up in the size of the grammar?

What we Already Know

- The size of the new grammar is **linearly** bounded by the size of the original grammar.
- But the factor grows **polynomially** in the size of the automaton.
- Each quantifier alternation in the MSO formula may induce an **exponential blow-up** in the size of the automaton.
- **But:** syntactic constraints do not need full MSO, first-order logic suffices.

Research Project

- What is the conversion complexity for (fragments of) first-order logic?
- What is the bound on the size of the automata?

Parallel Parsing

- Humans parse sentences in real-time (= faster than linear).
- Yet our current grammar formalisms display horrible serial parsing performance:

CFG	$O(n^3)$
TAG	$O(n^6)$
k-MCFG	$O(n^{3k})$

- But **parallel parsing algorithms improve speed** significantly for CFGs:

Algorithm	Time	Processors
OCKY	$O(n)$	$O(n^4)$
Rytter	$O(\log n)$	$O(n^6)$

Research Project

- Can we reduce the number of processors and stay linear?
- Can we generalize these algorithms from CFGs?

Parallel Parsing

- Humans parse sentences in real-time (= faster than linear).
- Yet our current grammar formalisms display horrible serial parsing performance:

CFG	$O(n^3)$
TAG	$O(n^6)$
k-MCFG	$O(n^{3k})$

- But **parallel parsing algorithms improve speed** significantly for CFGs:

Algorithm	**Time**	**Processors**
OCKY	$O(n)$	$O(n^4)$
Rytter	$O(\log n)$	$O(n^6)$

Research Project

- Can we reduce the number of processors and stay linear?
- Can we generalize these algorithms from CFGs?

More Parsing

- CFG parsing can be treated as Boolean matrix multiplication.
- In theory this improves efficiency only marginally to $O(n^{2.7\cdots})$.
- But there's a practical advantage:
  you can **reuse very efficient code** for matrix multiplication!

Research Project

- Are Boolean tensor spaces the analog for MCFG?
- Can we reuse tensor space algorithms?

More Parsing

- CFG parsing can be treated as Boolean matrix multiplication.
- In theory this improves efficiency only marginally to $O(n^{2.7\cdots})$.
- But there's a practical advantage:
  you can **reuse very efficient code** for matrix multiplication!

Research Project

- Are Boolean tensor spaces the analog for MCFG?
- Can we reuse tensor space algorithms?

Tensor space semantics

- New idea: word meanings as vectors in tensor spaces
- But linguists care about how sentence meaning arises from combining word meanings.
- **Greg Kobele**: meaning composition in PTIME

Example

every	$\lambda f_{\langle e,t\rangle} \lambda g_{\langle e,t\rangle} . \forall x[f(x) \rightarrow g(x)]$
boy	$\lambda x_e.\mathrm{boy}(x)$
slept	$\lambda x_e.\mathrm{slept}(x)$

every boy slept $= \forall x[\mathrm{boy}(x) \rightarrow \mathrm{slept}(x)]$

Research Project

- What are the tensor space analogs of our logical combinators?
- Do they preserve PTIME complexity?

Tensor space semantics

- New idea: word meanings as vectors in tensor spaces
- But linguists care about how sentence meaning arises from combining word meanings.
- **Greg Kobele**: meaning composition in PTIME

Example

every	$\lambda f_{\langle e,t\rangle}\lambda g_{\langle e,t\rangle}.\forall x[f(x) \to g(x)]$
boy	$\lambda x_e.\mathrm{boy}(x)$
slept	$\lambda x_e.\mathrm{slept}(x)$

every boy slept $= \forall x[\mathrm{boy}(x) \to \mathrm{slept}(x)]$

Research Project

- What are the tensor space analogs of our logical combinators?
- Do they preserve PTIME complexity?

Game-Theoretic Pragmatics

Pragmatics study of intended (rather than literal) meaning

Example

- Can you pass the salt $\neq$ physical capability to pass the salt
- I could care less $=$ I don't care at all

One successful model of pragmatics is bidirectional OT, which is equivalent to **signaling games**.

Research Project

- How are signaling games computed?
- How can we integrate them with parsing?

Game-Theoretic Pragmatics

Pragmatics study of intended (rather than literal) meaning

Example

- Can you pass the salt $\neq$ physical capability to pass the salt
- I could care less $=$ I don't care at all

One successful model of pragmatics is bidirectional OT, which is equivalent to **signaling games**.

Research Project

- How are signaling games computed?
- How can we integrate them with parsing?

The Bigger Picture: Symbiosis

- Computational questions in linguistics give rise to general-purpose methods, techniques and results.
- Linguistics also needs to import know-how from other computational fields to solve many challenges.

References I

Chandlee, Jane, Jie Fu, Konstantinos Karydis, Cesar Koirala, Jeffrey Heinz, and Herbert G. Tanner. 2012. Integrating grammatical inference into robotic planning. In *Proceedings of the Eleventh International Conference on Grammatical Inference (ICGI 2012)*, ed. Jeffrey Heinz, Colin de la Higuera, and Tim Oates, volume 21, 69–83. JMLR Workshop and Conference Proceedings.

Chomsky, Noam. 1956. Three models for the description of language. *IRE Transactions on Information Theory* 2:113–124.

Chomsky, Noam. 1957. *Syntactic structures*. The Hague: Mouton.

Chomsky, Noam. 1959. On certain formal properties of grammars. *Information and Control* 2:137–167.

Chomsky, Noam. 1965. *Aspects of the theory of syntax*. Cambridge, MA: MIT Press.

Chomsky, Noam, and M. P. Schützenberger. 1963. The algebraic theory of context-free languages. In *Computer programming and formal systems*, ed. P. Braffort and D. Hirschberg, Studies in Logic and the Foundations of Mathematics, 118–161. Amsterdam: North-Holland.

Graf, Thomas. 2011. Closure properties of minimalist derivation tree languages. In *LACL 2011*, ed. Sylvain Pogodalla and Jean-Philippe Prost, volume 6736 of *Lecture Notes in Artificial Intelligence*, 96–111. Heidelberg: Springer.

References II

Graf, Thomas. 2013. *Local and transderivational constraints in syntax and semantics*. Doctoral Dissertation, UCLA.

Heinz, Jeffrey. 2010. Learning long-distance phonotactics. *Linguistic Inquiry* 41:623–661.

Heinz, Jeffrey, Anna Kasprzik, and Timo Kötzing. 2012. Learning with lattice-structure hypothesis spaces. *Theoretical Computer Science* 457:111–127.

Heinz, Jeffrey, Chetan Rawal, and Herbert G. Tanner. 2011. Tier-based strictly local constraints in phonology. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics*, 58–64.

Kobele, Gregory M. 2006. *Generating copies: An investigation into structural identity in language and grammar*. Doctoral Dissertation, UCLA.

Kuroda, Sige-Yuki. 1964. Classes of languages and linear-bounded automata. *Information and Control* 7:207–223.

McNaughton, Robert, and Seymour Pappert. 1971. *Counter-free automata*. Cambridge, MA: MIT Press.

Rogers, James, Jeffrey Heinz, Gil Bailey, Matt Edlefsen, Molly Vischer, David Wellcome, and Sean Wibel. 2010. On languages piecewise testable in the strict sense. In *The mathematics of language*, ed. Christan Ebert, Gerhard Jäger, and Jens Michaelis, volume 6149 of *Lecture Notes in Artificial Intelligence*, 255–265. Heidelberg: Springer.

References III

Rounds, William C. 1970. Mappings on grammars and trees. *Mathematical Systems Theory* 4:257–287.

Yngve, Victor. 1958. A programming language for mechanical translation. *Mechanical Translation* 5:25–41.

Yngve, Victor H. 1955. Syntax and the problem of multiple meaning. In *Machine translation of languages*, ed. William N. Locke and A. Donald Booth, 208–226. Cambrdige, MA: MIT Press.