Syntax, Semantics, Pragmatics: Where do we Find Optimality?

Thomas Graf
tgraf@ucla.edu
tgraf.bol.ucla.edu

University of California, Los Angeles

Universität Göttingen, Göttingen, Germany
December 14, 2010

Optimality in Pragmatics

Many pragmatic phenomena are analyzed as requiring comparisons of several alternatives and picking the best one (examples from Blutner and Zeevat 2008)

- Implicatures
    - Conventional implicatures
    - Conversational implicatures
    - Scalar implicatures
    - Exhaustivity implicatures
    - Implicature projection
- Presupposition projection
- Distribution of discourse particles

  ⋮

Optimality in Syntax/Semantics: Reference-Set Constraints

Optimality condition ≈ reference-set constraint ≈ transderivational constraint ≈ global economy condition ≈ interface strategy

An Informal Definition

Given some input tree t, a reference-set constraint computes a set of possible output trees for t — called the reference set of t — and picks from said set the optimal output tree according to some economy metric.

Some examples from the literature:

- Rule I (Reinhart 2006)
- Scope Economy (Fox 2000)
- Fewest Steps (Chomsky 1995)
- Merge-over-Move (Chomsky 2000)
- Focus Economy (Reinhart 2006)

Optimality in Syntax/Semantics: Reference-Set Constraints

Optimality condition ≈ reference-set constraint
≈ transderivational constraint ≈ global economy condition ≈ interface strategy

An Informal Definition

Given some input tree t, a reference-set constraint computes a set of possible output trees for t — called the reference set of t — and picks from said set the optimal output tree according to some economy metric.

Some examples from the literature:

- Rule I (Reinhart 2006)
- Scope Economy (Fox 2000)
- Fewest Steps (Chomsky 1995)
- Merge-over-Move (Chomsky 2000)
- Focus Economy (Reinhart 2006)

Optimality in Syntax/Semantics: Reference-Set Constraints

Optimality condition ≈ reference-set constraint ≈ transderivational constraint ≈ global economy condition ≈ interface strategy

An Informal Definition

Given some input tree t, a reference-set constraint computes a set of possible output trees for t — called the reference set of t — and picks from said set the optimal output tree according to some economy metric.

Some examples from the literature:

- Rule I (Reinhart 2006)
- Scope Economy (Fox 2000)
- Fewest Steps (Chomsky 1995)
- Merge-over-Move (Chomsky 2000)
- Focus Economy (Reinhart 2006)

Example: Focus Economy

(1) a. [$_{\mathrm{TP}}$ John [$_{\mathrm{VP}}$ bought [$_{\mathrm{DP}}$ a red **car**]]].
Focus set: {TP, VP, DP, red car, car}

b. [$_{\mathrm{TP}}$ John [$_{\mathrm{VP}}$ bought [$_{\mathrm{DP}}$ a **red** car]]].
Focus set: {red}

Focus Projection

Any constituent containing the carrier of sentential main stress may be focused.

Focus Economy Rule

If the main stress has been shifted, a constituent containing its carrier may be focused iff it cannot be focused in the tree with unshifted stress.

Example: Focus Economy

(2) a. [$_{\text{TP}}$ John [$_{\text{VP}}$ bought [$_{\text{DP}}$ a red **car**]]].
Focus set: {TP, VP, DP, red car, car}

b. [$_{\text{TP}}$ John [$_{\text{VP}}$ bought [$_{\text{DP}}$ a **red** car]]].
Focus set: {red}

Focus Projection

Any constituent containing the carrier of sentential main stress may be focused.

Focus Economy Rule

If the main stress has been shifted, a constituent containing its carrier may be focused iff it cannot be focused in the tree with unshifted stress.

Example: Focus Economy, Cont.

Computing the Focus Sets

a) Neutral Stress

- TP_S
  - John_W
  - VP_S
    - bought_W
    - DP_S
      - a_W
      - AP_S
        - red_W
        - car_S

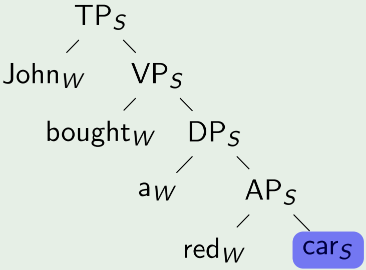
b) Shifted Stress

- TP_S
  - John_W
  - VP_S
    - bought_W
    - DP_S
      - a_W
      - AP_S
        - red_S
        - car_W

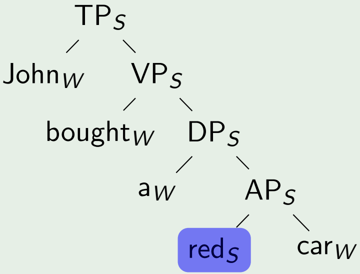
Example: Focus Economy, Cont.

Computing the Focus Sets

a) Neutral Stress



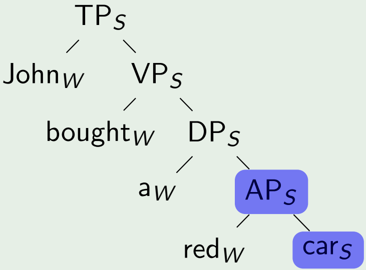
b) Shifted Stress



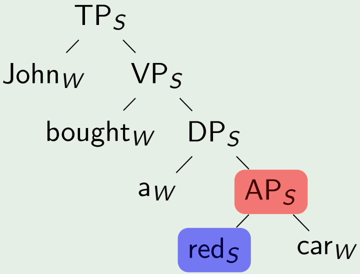
Example: Focus Economy, Cont.

Computing the Focus Sets

a) Neutral Stress



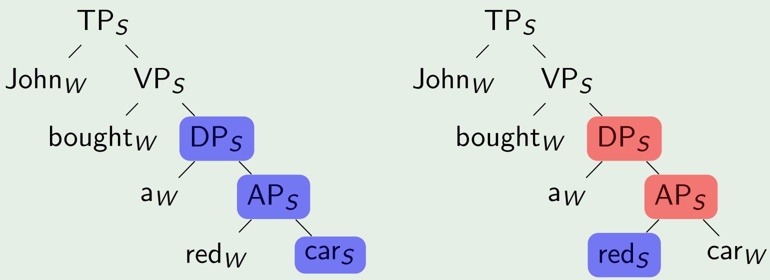
b) Shifted Stress



Example: Focus Economy, Cont.

Computing the Focus Sets

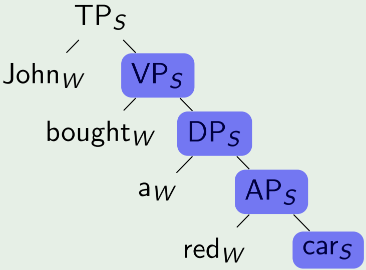
a) Neutral Stress **b) Shifted Stress**



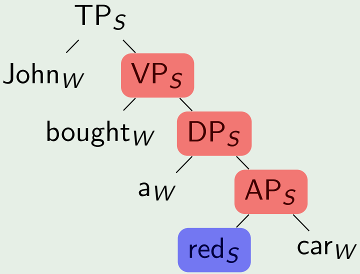
Example: Focus Economy, Cont.

Computing the Focus Sets

a) Neutral Stress



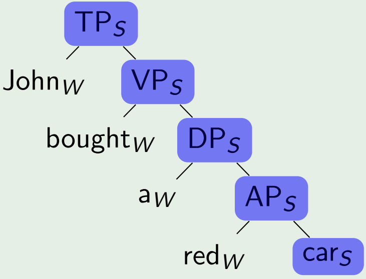
b) Shifted Stress



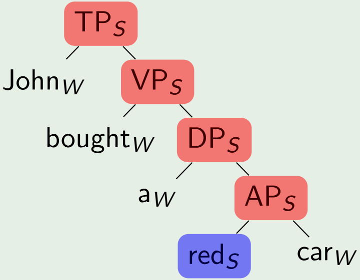
Example: Focus Economy, Cont.

Computing the Focus Sets

a) Neutral Stress



b) Shifted Stress



Optimality in Syntax, Semantics, Pragmatics

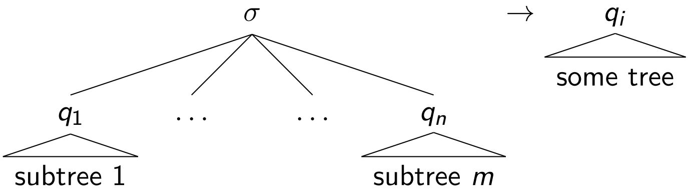
- It seems that the same kind of optimality conditions can be found in all three modules:
  1. compute set of alternatives
  2. pick best option
- But if we use linear tree transducers as a model, it turns out that reference-set constraints involve no comparisons. Rather, they are...
  - **Insight 1 (theory-internal)**
    a different way of specifying standard well-formedness constraints ⇒ involve no tangible notion of optimality
  - **Insight 2 (across theories)**
    connected to unidirectional OT.
- Pragmatic optimality conditions, on the other hand, are usually modelled with bidirectional OT
  ⇒ different from reference-set constraints.

Linear Tree Transducers in Pictures

A linear finite-state bottom-up tree transducer

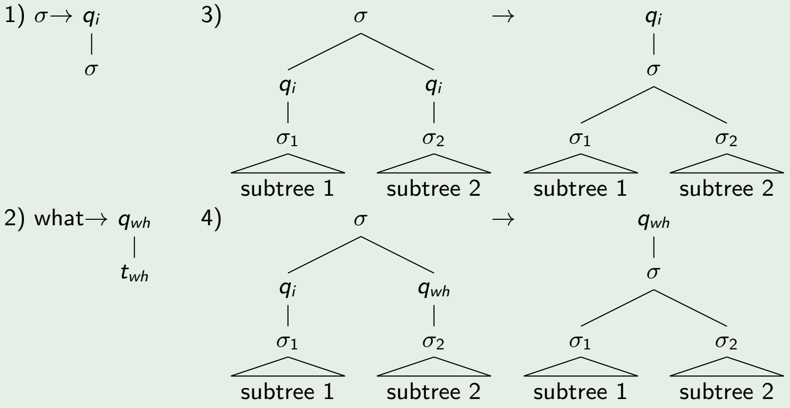
- traverses an input-tree from the leaves towards the root,
- labels it with states q_i, and
- transforms it into an output-tree.

It does so using rules of the following kind:



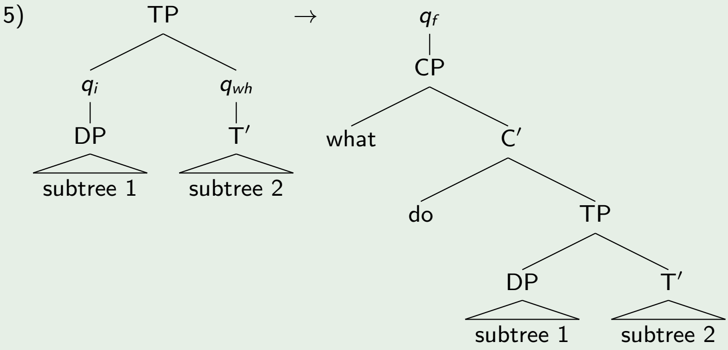
A Simple Example (Part 1)

A Transduction for Restricted wh-Movement, Rules 1–4



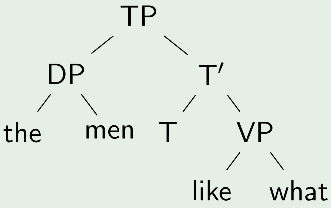
A Simple Example (Part 2)

A Transduction for Restricted wh-Movement, Rule 5



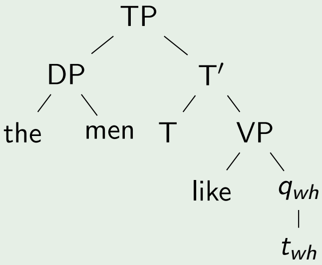
A Simple Example (Part 3)

A Transduction for Restricted wh-Movement, Application



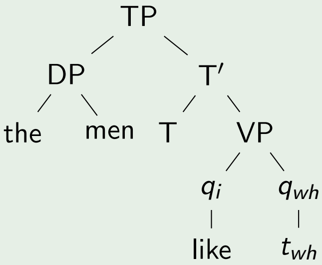
A Simple Example (Part 3)

A Transduction for Restricted wh-Movement, Application



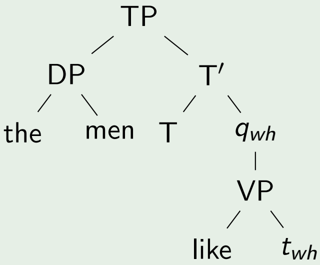
A Simple Example (Part 3)

A Transduction for Restricted wh-Movement, Application



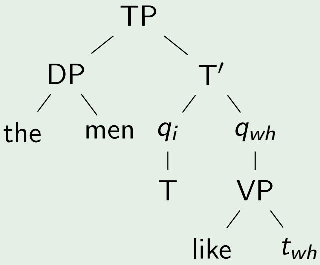
A Simple Example (Part 3)

A Transduction for Restricted wh-Movement, Application



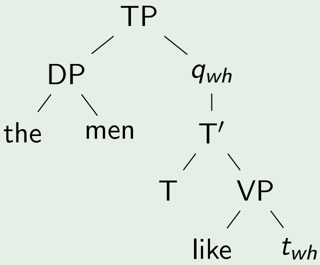
A Simple Example (Part 3)

A Transduction for Restricted wh-Movement, Application



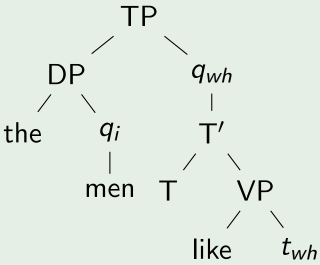
A Simple Example (Part 3)

A Transduction for Restricted wh-Movement, Application



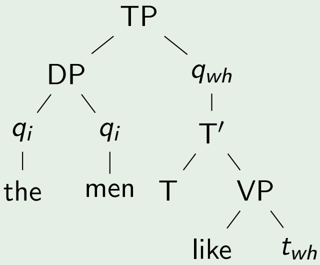
A Simple Example (Part 3)

A Transduction for Restricted wh-Movement, Application



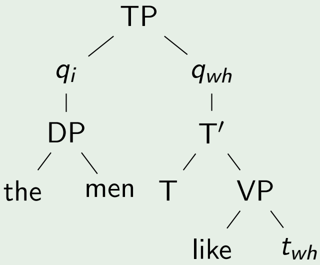
A Simple Example (Part 3)

A Transduction for Restricted wh-Movement, Application



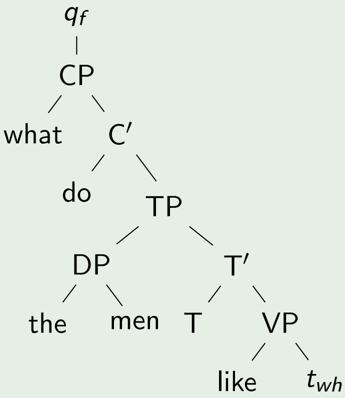
A Simple Example (Part 3)

A Transduction for Restricted wh-Movement, Application



A Simple Example (Part 3)

A Transduction for Restricted wh-Movement, Application



Some Important Facts

What is Possible?

- Relabeling nodes
- Deleting subtrees
- Inserting subtrees of bounded size
- Enforcing constraints that define regular tree languages

What is Impossible?

- Copying of arbitrary subtrees
- Switching positions of non-siblings (e.g. specifier and complement)
- Counting past some threshold

Mathematical Properties

- A transducer can be decomposed into a sequence of smaller transducers, *et vice versa*.
- If the input tree language of a transducer is regular, then so is its output language. Regular tree languages are sufficiently powerful for Minimalism (Kobele et al. 2007).

Some Important Facts

What is Possible?

- Relabeling nodes
- Deleting subtrees
- Inserting subtrees of bounded size
- Enforcing constraints that define regular tree languages

What is Impossible?

- Copying of arbitrary subtrees
- Switching positions of non-siblings (e.g. specifier and complement)
- Counting past some threshold

Mathematical Properties

- A transducer can be decomposed into a sequence of smaller transducers, *et vice versa*.
- If the input tree language of a transducer is regular, then so is its output language. Regular tree languages are sufficiently powerful for Minimalism (Kobele et al. 2007).

Some Important Facts

What is Possible?

- Relabeling nodes
- Deleting subtrees
- Inserting subtrees of bounded size
- Enforcing constraints that define regular tree languages

What is Impossible?

- Copying of arbitrary subtrees
- Switching positions of non-siblings (e.g. specifier and complement)
- Counting past some threshold

Mathematical Properties

- A transducer can be decomposed into a sequence of smaller transducers, *et vice versa*.
- If the input tree language of a transducer is regular, then so is its output language. Regular tree languages are sufficiently powerful for Minimalism (Kobele et al. 2007).

Overall Reasoning

Strategy

For a given reference-set constraint C, exhibit

- a Minimalist grammar that generates the input language, and
- a sequence of transducers that computes the same mapping from inputs to optimal outputs.

- Due to the mathematical properties of transducers, the output language is no more complex than the input language
- Hence it can be generated by some Minimalist grammar
- Hence C is equivalent to some “constraint” that does not involve reference-set computation.

Overall Reasoning

Strategy

For a given reference-set constraint C, exhibit

- a Minimalist grammar that generates the input language, and
- a sequence of transducers that computes the same mapping from inputs to optimal outputs.

- Due to the mathematical properties of transducers, the output language is no more complex than the input language
- Hence it can be generated by some Minimalist grammar
- Hence C is equivalent to some "constraint" that does not involve reference-set computation.

But why should this work for arbitrary reference-set constraints?

OT: A Bird's Eye Perspective

It seems natural to model reference-set constraints via OT.

Reference-Set Constraints as OT Grammars

- Use Gen to compute the reference-sets.
- Use a sequence of constraints to filter out the suboptimal candidates.

A Major Complaint

Without further restrictions, OT grammars can generate any kind of (tree) language
$\Rightarrow$ they don't tell us anything about reference-set constraints.

Fortunately, there is a weaker alternative...

OT: A Bird's Eye Perspective

It seems natural to model reference-set constraints via OT.

Reference-Set Constraints as OT Grammars

- Use GEN to compute the reference-sets.
- Use a sequence of constraints to filter out the suboptimal candidates.

A Major Complaint

Without further restrictions, OT grammars can generate any kind of (tree) language
$\Rightarrow$ they don't tell us anything about reference-set constraints.

Fortunately, there is a weaker alternative...

OT: A Bird's Eye Perspective

It seems natural to model reference-set constraints via OT.

Reference-Set Constraints as OT Grammars

- Use GEN to compute the reference-sets.
- Use a sequence of constraints to filter out the suboptimal candidates.

A Major Complaint

Without further restrictions, OT grammars can generate any kind of (tree) language
$\Rightarrow$ they don't tell us anything about reference-set constraints.

Fortunately, there is a weaker alternative...

Optimality Systems: A Restricted Version of OT

Optimality Systems (OSs; Frank and Satta 1998)

A variant of OT that keeps just the bare skeleton.

- All constraints only consider the output, never the input.
- No correspondence theory
- No output-output correspondence
- No sympathy constraints

There are mathematical conditions that ensure that an OS can be implemented by a tree transducer.

A Minor Quibble

GEN is too "flat" for faithful models of reference-set computation, it does not directly represent reference-sets and their algebraic properties.

Optimality Systems: A Restricted Version of OT

Optimality Systems (OSs; Frank and Satta 1998)

A variant of OT that keeps just the bare skeleton.

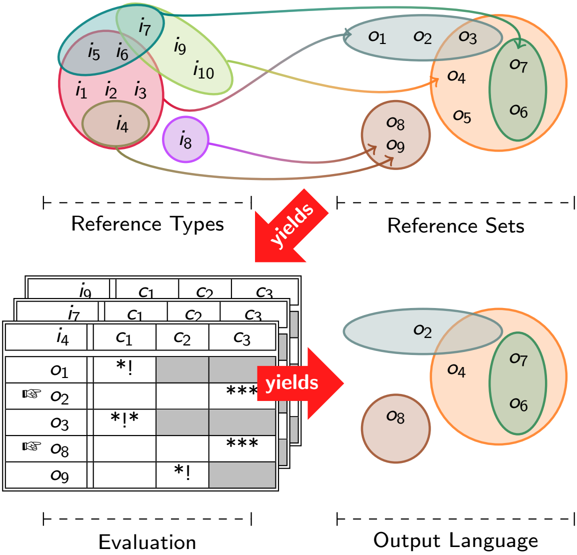
- All constraints only consider the output, never the input.
- No correspondence theory
- No output-output correspondence
- No sympathy constraints

There are mathematical conditions that ensure that an OS can be implemented by a tree transducer.

A Minor Quibble

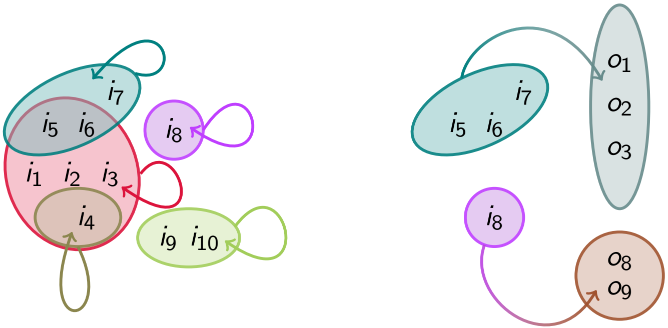
GEN is too "flat" for faithful models of reference-set computation, it does not directly represent reference-sets and their algebraic properties.

Depiction of a Controlled OS



Reference-set Constraints as Controlled OSs

- Almost all constraints in the literature exhibit one of the two configurations below.
- What do the two have in common?



Output Joint Preservation

Output Joint Preservation

If two reference sets overlap, then so do the reference types that are mapped to them.

Theorem (Frank and Satta 1998; Wartena 2000; Jäger 2002)

A controlled OS can be implemented as a transducer if

- the OS is output-joint preserving, and
- the input language is regular, and
- GEN and all constraints can be implemented as transducers.

Time to check this for specific reference-set constraints!

Output Joint Preservation

Output Joint Preservation

If two reference sets overlap, then so do the reference types that are mapped to them.

Theorem (Frank and Satta 1998; Wartena 2000; Jäger 2002)

A controlled OS can be implemented as a transducer if

- the OS is output-joint preserving, and
- the input language is regular, and
- GEN and all constraints can be implemented as transducers.

Time to check this for specific reference-set constraints!

Output Joint Preservation

Output Joint Preservation

If two reference sets overlap, then so do the reference types that are mapped to them.

Theorem (Frank and Satta 1998; Wartena 2000; Jäger 2002)

A controlled OS can be implemented as a transducer if

- the OS is output-joint preserving, and
- the input language is regular, and
- GEN and all constraints can be implemented as transducers.

Time to check this for specific reference-set constraints!

Output Joint Preservation

Output Joint Preservation

If two reference sets overlap, then so do the reference types that are mapped to them.

Theorem (Frank and Satta 1998; Wartena 2000; Jäger 2002)

A controlled OS can be implemented as a transducer if

- the OS is output-joint preserving, and
- the input language is regular, and
- Gen and all constraints can be implemented as transducers.

Time to check this for specific reference-set constraints!

Output Joint Preservation

Output Joint Preservation

If two reference sets overlap, then so do the reference types that are mapped to them.

Theorem (Frank and Satta 1998; Wartena 2000; Jäger 2002)

A controlled OS can be implemented as a transducer if

- the OS is output-joint preserving, and
- the input language is regular, and
- Gen and all constraints can be implemented as transducers.

Time to check this for specific reference-set constraints!

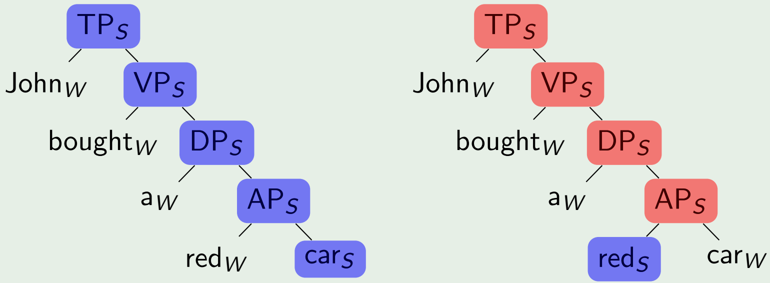
Example 1: Focus Economy

Focus Economy Rule (Reminder)

If the main stress has been shifted, a constituent containing its carrier may be focused iff it cannot be focused in the tree with unshifted stress.

Computing the Focus Sets

a) Neutral Stress **b) Shifted Stress**

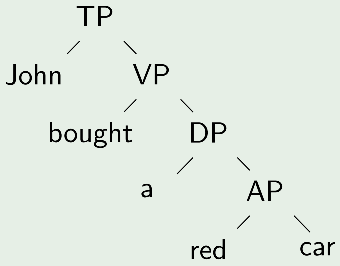


Transducer Model: Gen

Step 1 & 2: Gen

- Non-deterministically relabel input with S/W-subscripts.
- Non-deterministically focus some node along the "stress path".

Transducing an Input into a Stress-Annotated Output with Focus

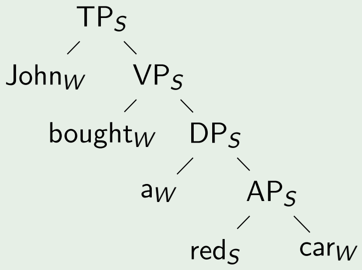


Transducer Model: Gen

Step 1 & 2: Gen

- Non-deterministically relabel input with S/W-subscripts.
- Non-deterministically focus some node along the "stress path".

Transducing an Input into a Stress-Annotated Output with Focus

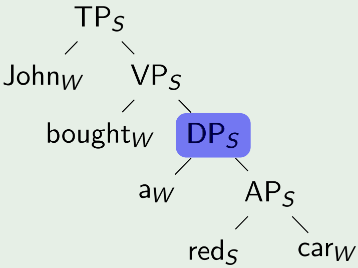


Transducer Model: Gen

Step 1 & 2: Gen

- Non-deterministically relabel input with S/W-subscripts.
- Non-deterministically focus some node along the "stress path".

Transducing an Input into a Stress-Annotated Output with Focus



Transducer Model: The Constraint

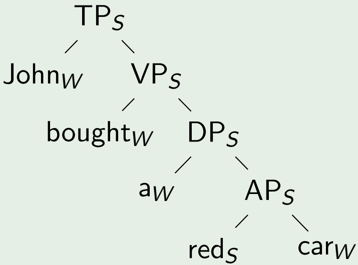
Focus Economy requires reference to the neutral stress pattern. We allow this by implicitly representing the neutral stress within the same tree!

Strategy

- Define two paths StressPath and NeutralPath.
- StressPath represents the path of the current stress.
- NeutralPath represents the path of the neutral stress.
- Add a constraint that requires focus to be in the stress path, but unless StressPath and NeutralPath pick out the same nodes, focus may not be in NeutralPath.

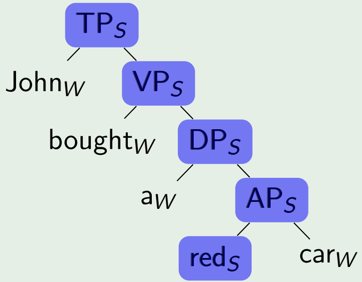
Example of ϕ

StressPath and NeutralPath



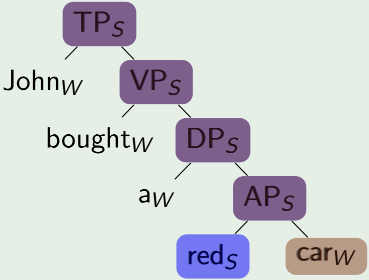
Example of ϕ

StressPath and NeutralPath



Example of ϕ

StressPath and NeutralPath



Merge-over-Move (MOM)

Merge-over-Move (MOM)

If two convergent derivations d and d' are built from the same lexical items and identical up to step n, at which point d continues with Merge and d' with Move, filter out d'.

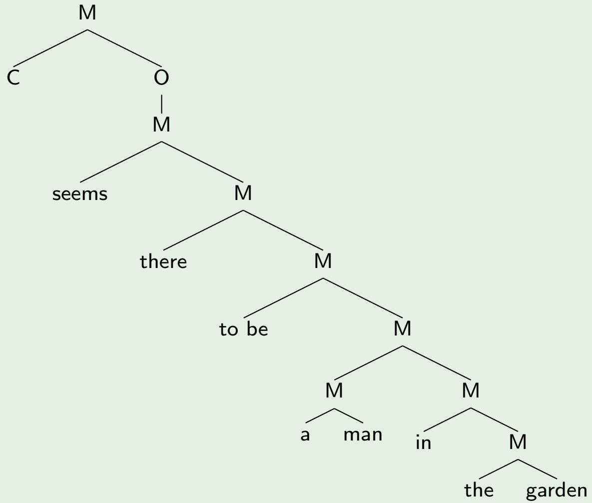
(3) a. There seems t_{there} to be a man in the garden.

b. * There seems a man to be $t_{\text{a man}}$ in the garden.

c. A man seems $t_{\text{a man}}$ to be $t_{\text{a man}}$ in the garden.

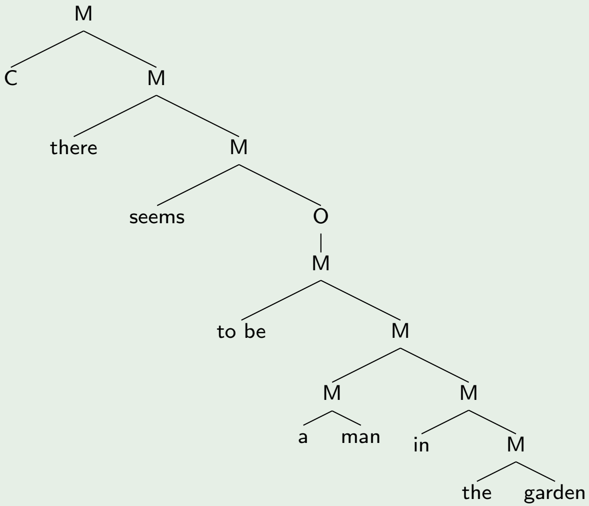
Derivation Trees of (3a) and (3b)

Example



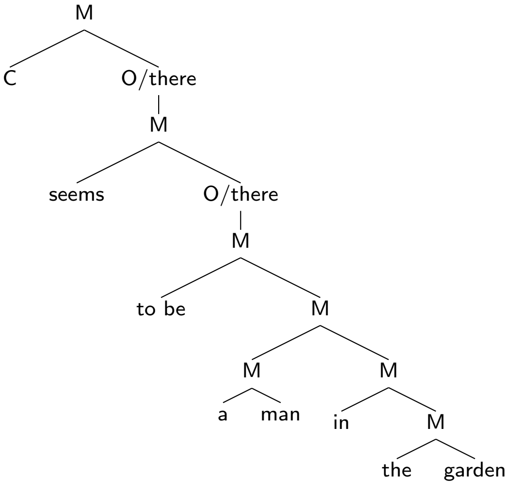
Derivation Trees of (3a) and (3b)

Example



Transducer Model: GEN (Step 1)

- Fuse the two derivations into one underspecified derivation.
  - Remove all features but the category feature.
  - Inside TP: Replace O or Merger of *there* by new label O/there.

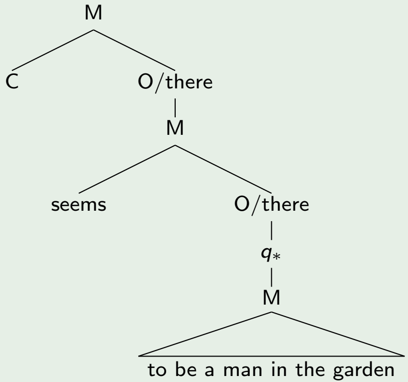


Transducer Model: GEN (Step 2)

- Turn O/there back into O or Merge of *there*.
  - Use a transducer with states q_*, q_O and q_C.
  - In state q_*, the transducer non-deterministically rewrites O/there as O or Merge of *there*.
  - If the transducer rewrites O/there as O, it switches into state q_0.
  - In state q_0, every occurence of O/there is rewritten just as O.
  - The transducer switches out of q_0 only if it encounters a CP (indicated by state q_C; cf. structured numerations).
- Reinstantiate the features.

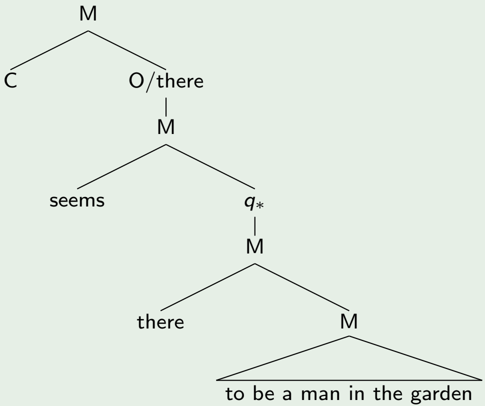
Transducer Model: Examples of Step 2

Example 1



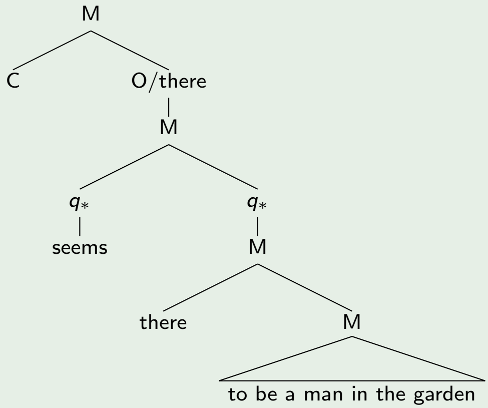
Transducer Model: Examples of Step 2

Example 1



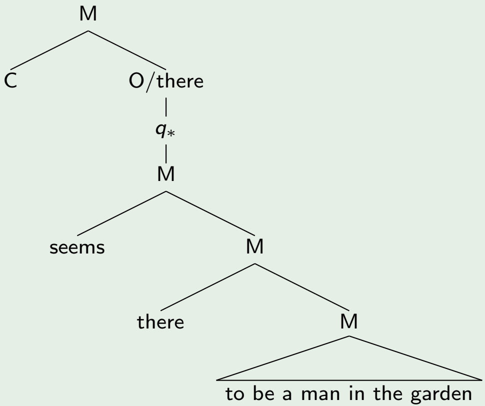
Transducer Model: Examples of Step 2

Example 1



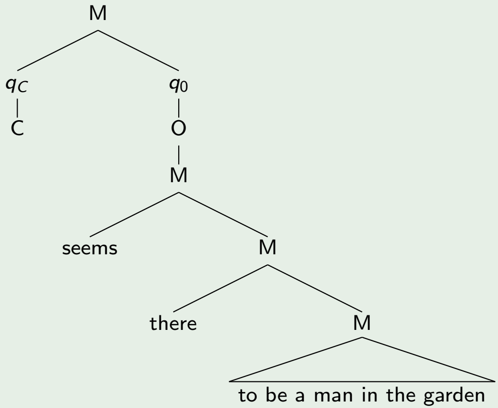
Transducer Model: Examples of Step 2

Example 1



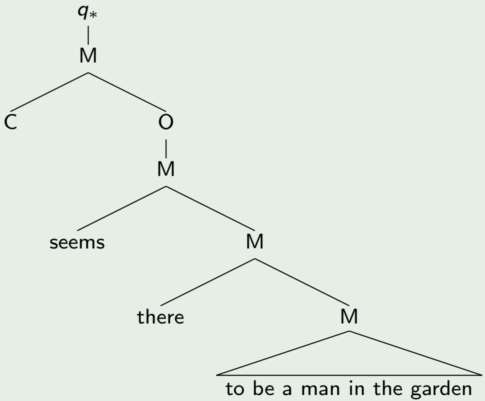
Transducer Model: Examples of Step 2

Example 1



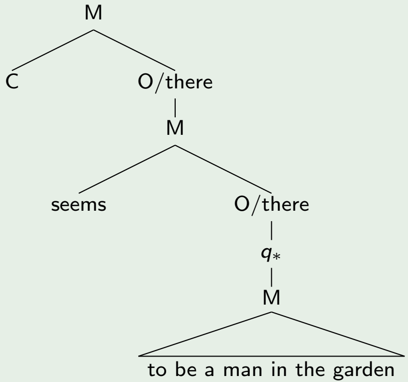
Transducer Model: Examples of Step 2

Example 1



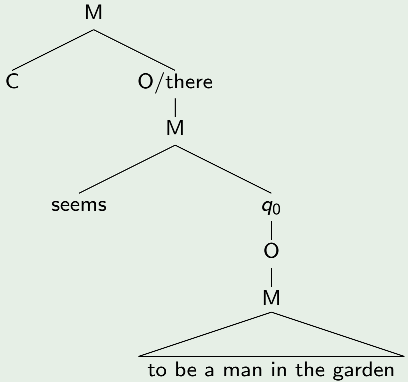
Transducer Model: Examples of Step 2

Example 2



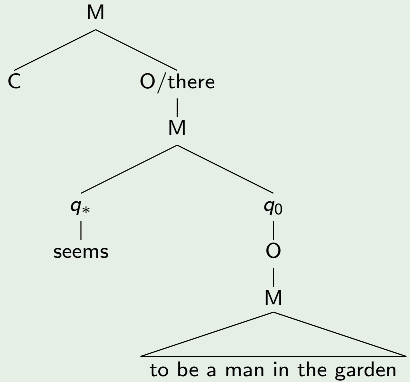
Transducer Model: Examples of Step 2

Example 2



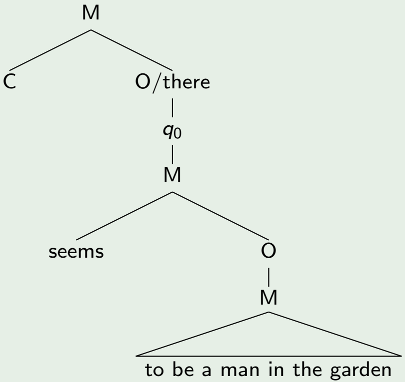
Transducer Model: Examples of Step 2

Example 2



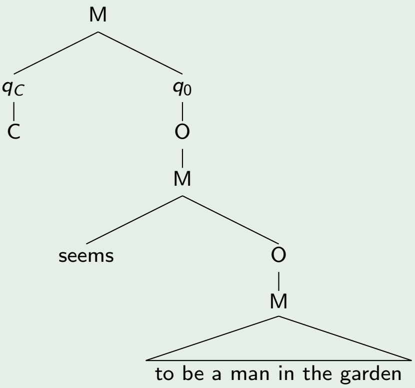
Transducer Model: Examples of Step 2

Example 2



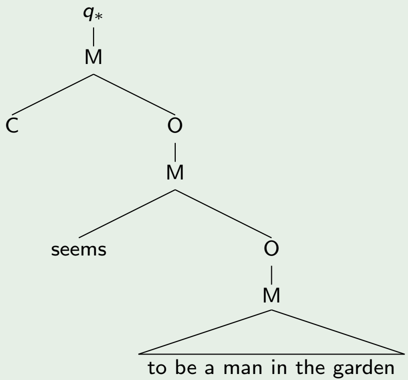
Transducer Model: Examples of Step 2

Example 2



Transducer Model: Examples of Step 2

Example 2



Transducer Model: The Induced Mapping

The output candidates for both (4a) and (4b) are now (5a)–(5b).

(4) a. There seems t_{there} to be a man in the garden.

b. * There seems a man to be $t_{\text{a man}}$ in the garden.

(5) a. * There seems there to be a man in the garden.

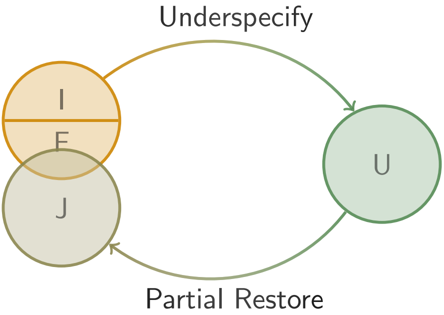
b. There seems t_{there} to be a man in the garden.

c. A man seems $t_{\text{a man}}$ to be $t_{\text{a man}}$ in the garden.

- We may extend the mapping such that (5c) is also assigned this reference set.
- (5a) still has to be ruled out.

Transducer Model: The Constraint

The only constraint is the input language itself!
By turning it into a transducer and composing it with GEN,
we remove all instances of overgeneration and filter out the illicit MOM violators.



Shortest Derivation Principle (SDP)

SDP

Given convergent derivations $d_1, \ldots, d_n$ over the same lexical items, pick the one(s) with the fewest instances of Move.

Why do we find the following contrast?

(6) a. Who$_i$ did John take [$_{\text{DP}_j}$ a picture of t_i]?

b. * Who$_i$ was [$_{\text{DP}_j}$ a picture of t_i] taken t_j by John?

Derivations for (6b)

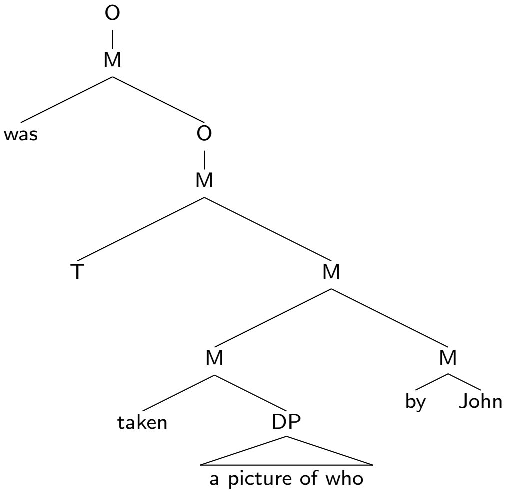
Two derivations are possible for (6b).
CED violation in (7c)

(7) a. $[_{VP}$ taken $[_{DP_j}$ a picture of who$_i]$ by John$]$
b. $[_{TP}$ $[_{DP_j}$ a picture of who$_i$ $]$ T $[_{VP}$ taken t_j by John$]]$
c. $[_{CP}$ who$_i$ was $[_{TP}$ $[_{DP_j}$ a picture of t_i $]$ T $[_{VP}$ taken t_j by John$]]]$

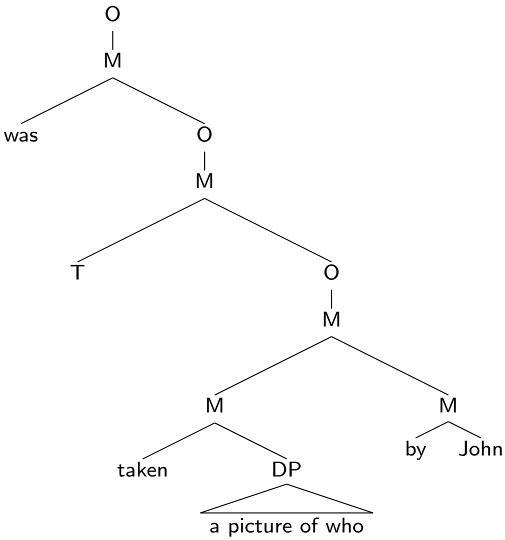
Derivation (8) is longer than (7)!

(8) a. $[_{VP}$ taken $[_{DP_j}$ a picture of who$_i]$ by John$]$
b. $[_{VP}$ who$_i$ taken $[_{DP_j}$ a picture of $t_i]$ by John$]$
c. $[_{TP}$ $[_{DP_j}$ a picture of t_i $]$ T $[_{VP}$ who$_i$ taken t_j by John$]]$
d. $[_{CP}$ who$_i$ was $[_{TP}$ $[_{DP_j}$ a picture of t_i $]$ T $[_{VP}$ taken t_j by John$]]]$

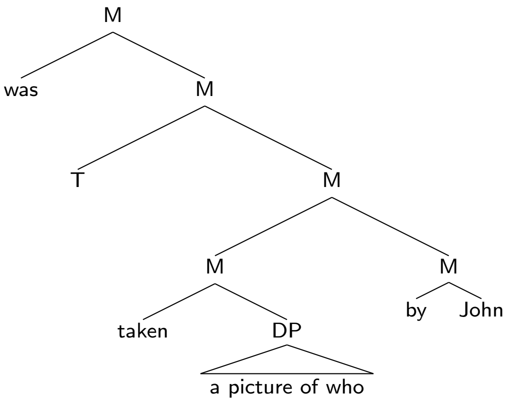
Derivation Tree of (7)



Derivation Tree of (8)



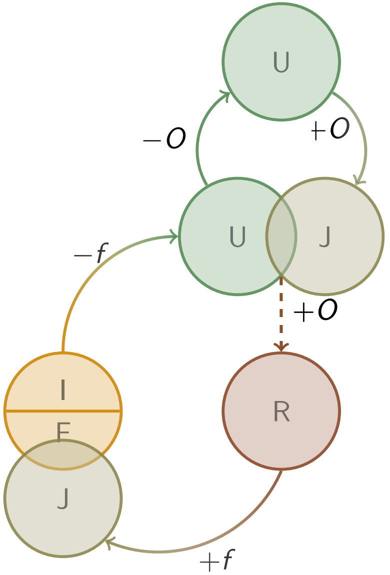
Underspecified Derivation Tree of (7) and (8)



Strategy

- Compute reference-set by
  1. mapping to underspecified derivation (i.e. remove Move-nodes)
  2. arbitrarily adding Move-nodes to underspecified derivation
  3. discarding all derivation trees that aren't in the input language (i.e. the junk)
- Filter out the suboptimal derivation trees (those that can be obtained from others by adding Move-nodes)
  1. Let R be the transduction that maps a derivation tree to the trees in its reference-set and $+O$ the transduction defined by adding Move-nodes
  2. The range of the composition of R with $+O$ is the set of derivation trees that can be obtained from some tree in the range of R by adding Move-nodes, i.e. the suboptimal outputs.
  3. Thus, the relative complement of the range of R and the range of the composition of R with $+O$ is the set S of optimal outputs. Composing R with the diagonal over S maps every tree to its optimal outputs.

Architecture of SDP



Scope Economy ≠ Semantic SDP

Scope Economy

QR is licit only if it induces a change in meaning.

Scope Economy (Rephrased)

Given convergent derivations $d_1, \ldots, d_n$ that are identical modulo QR and have identical meaning, prefer the one with the fewest instances of Move.

- Checking semantic identity is hard.
- Even if we ignore semantics, Scope Economy needs more power than the SDP because the number of QR-able phrases per CP is not finitely bounded!
- We can move to a more powerful type of transducer that still preserves regularity, but we lose closure under composition $\Rightarrow$ Scope Economy structurally more demanding than SDP

Underspecification-and-Filtration

A Rule of Thumb

A reference-set constraint is likely to be computable by a transducer if

- one can find a structure that encodes the commonalities of all the competitors, and
- neither the underspecification step nor the recovery step require insertion of material of unbounded size, and
- the economy metric can be implemented as
  - a well-formedness constraint on underspecified structures, or
  - a specific restriction on the recovery step, or
  - a transducer that turns optimal candidates into suboptimal ones.

Advantages of Reference-Set Constraints

- **Modularity**
  Constraint only depends on input language,
  not on mechanisms that generate it
- **Succinctness**
  Non-reference-set correspondent may fail to make
  the restriction explicit or be much more complicated;
  reference-set constraint may subsume
  very different constraints, depending on input grammar
- **More Tweakable Parameters**
  Reference-set constraint gives us at least four parametrizable
  components: reference types, reference sets, the map between
  the two, and the economy metric.
- **Reaching out**
  Connections to OT, sTAG and others may allow us
  to incorporate results from these frameworks

Conclusion (Part 1)

- Tree transducers were proposed as a model for reference-set constraints.
- OSs offer a bird's eye view on them (**Insight 2**).
- Most requirements for an OS to be efficiently computable are fulfilled by reference-set constraints; in particular, their corresponding OSs are output joint preserving.
- The only problematic areas are GEN and the OS constraints.
- The underspecification-and-filtration strategy offers a general solution.

Conclusion (Part 2)

- **Syntax**
  optimality conditions can be modelled by transducers
  $\Rightarrow$ no optimality considerations involved
- **Semantics**
  - Incorporating semantic information is difficult.
  - Even on a purely structural level, more powerful transducers are necessary (cf. Scope Economy).
- **Pragmatics**
  assumed to require at least bidirectional optimization, whereas transducers correspond to unidirectional optimality $\Rightarrow$ optimality in pragmatics fundamentally different

References I

Blutner, Reinhard, and Henk Zeevat. 2008. Optimality-theoretic pragmatics. In *Semantics: An international handbook of natural language meaning*, ed. Claudia Maienborn, Klaus von Heusinger, and Paul Portner, pp–pp. Berlin: Mouton de Gruyter. To appear.

Chomsky, Noam. 1995. *The minimalist program*. Cambridge, Mass.: MIT Press.

Chomsky, Noam. 2000. Minimalist inquiries: The framework. In *Step by step: Essays on minimalist syntax in honor of Howard Lasnik*, ed. Roger Martin, David Michaels, and Juan Uriagereka, 89–156. Cambridge, Mass.: MIT Press.

Fox, Danny. 2000. *Economy and semantic interpretation*. Cambridge, Mass.: MIT Press.

Frank, Robert, and Giorgio Satta. 1998. Optimality theory and the generative complexity of constraint violability. *Computational Linguistics* 24:307–315.

Jäger, Gerhard. 2002. Gradient constraints in finite state OT: The unidirectional and the bidirectional case. In *More than words. A festschrift for Dieter Wunderlich*, ed. I. Kaufmann and B. Stiebels, 299–325. Berlin: Akademie Verlag.

References II

Kobele, Gregory M., Christian Retoré, and Sylvain Salvati. 2007. An automata-theoretic approach to minimalism. In *Model Theoretic Syntax at 10*, ed. James Rogers and Stephan Kepser, 71–80. Workshop organized as part of the Europen Summer School on Logic, Language and Information, ESSLLI 2007, 6-17 August 2007, Dublin, Ireland.

Reinhart, Tanya. 2006. *Interface strategies: Optimal and costly computations*. Cambridge, Mass.: MIT Press.

Wartena, Christian. 2000. A note on the complexity of optimality systems. In *Studies in optimality theory*, ed. Reinhard Blutner and Gerhard Jäger, 64–72. Potsdam, Germany: University of Potsdam.