Optimality is not a Race

Against a Performance-Based View of Reference-Set Computation

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

University of California, Los Angeles

Glow 34
April 29, 2011

Reference-Set Constraints

Optimality condition $\approx$ reference-set constraint
$\approx$ transderivational constraint $\approx$ global economy condition
$\approx$ interface strategy

An Informal Definition

Given some input tree t, a reference-set constraint (RC) 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)
- Shortest Derivation Principle/Fewest Steps (Chomsky 1995)
- Merge-over-Move (Chomsky 2000)
- Focus Economy (Reinhart 2006)

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 (RC) 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)
- Shortest Derivation Principle/Fewest Steps (Chomsky 1995)
- Merge-over-Move (Chomsky 2000)
- Focus Economy (Reinhart 2006)

Reference-Set Constraints

Optimality condition $\approx$ reference-set constraint
$\approx$ transderivational constraint $\approx$ global economy condition
$\approx$ interface strategy

An Informal Definition

Given some input tree t, a reference-set constraint (RC) 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)
- Shortest Derivation Principle/Fewest Steps (Chomsky 1995)
- Merge-over-Move (Chomsky 2000)
- Focus Economy (Reinhart 2006)

Deriving RCs from Performance Factors

The issue: Most linguists don't want RCs in syntax.
However, they seem to be empirically useful. Is there a solution?

Why no RCs in Syntax?

- allegedly too computationally demanding (Johnson and Lappin 1999)
- new, peculiar class of constraints ⇒ stipulative
- conceptually inconsistent (Collins 1996)
- leaves notion of simplicity unexplained (Jacobson 1997)

Why RCs in the Parser?

- RC = race
- race-effects due to parallel computation
- ample evidence for parallel computation
- simplicity = fastest to compute

Deriving RCs from Performance Factors

The issue: Most linguists don't want RCs in syntax.
However, they seem to be empirically useful. Is there a solution?

Why no RCs in Syntax?

- allegedly too computationally demanding (Johnson and Lappin 1999)
- new, peculiar class of constraints ⇒ stipulative
- conceptually inconsistent (Collins 1996)
- leaves notion of simplicity unexplained (Jacobson 1997)

Why RCs in the Parser?

- RC = race
- race-effects due to parallel computation
- ample evidence for parallel computation
- simplicity = fastest to compute

Deriving RCs from Performance Factors

The issue: Most linguists don't want RCs in syntax.
However, they seem to be empirically useful. Is there a solution?

Why no RCs in Syntax?

- allegedly too computationally demanding (Johnson and Lappin 1999)
- new, peculiar class of constraints ⇒ stipulative
- conceptually inconsistent (Collins 1996)
- leaves notion of simplicity unexplained (Jacobson 1997)

Why RCs in the Parser?

- RC = race
- race-effects due to parallel computation
- ample evidence for parallel computation
- simplicity = fastest to compute

Jacobson (1997:303f) on RCs

[. . .] I do want to suggest that the grammar indeed does not contain transderivational processes as such. Many of the cases which appear to necessitate this device seem likely to be the result of processing and/or production principles [. . .] The reasons for being suspicious of the existence of such cases is quite straight-forward: If some device computes a set of alternatives and then selects the simplest one, it is a complete accident that the selection is for the simplest. Once the full set of alternatives [. . .] has been computed, it is just as easy to locate the most complex, the third simplest, or any other possibility (including possibilities which have nothing to do with simplicity). Thus, if there are indeed cases where the simplest candidate appears to "win out" over other possibilities, then this becomes unmysterious only if we can view the system as being *driven* in some way to that alternative rather than *selecting* the simplest from among a set of fully computed alternatives. [. . .] However, as we understand formal grammatical principles, there is no way to locate this kind of process in the grammar itself. There are, however, ways to think of processing and production mechanisms in this light.

This Talk

The Basic Message

- You cannot have a grammar without RCs.
  You always get RCs for free because they are simply a different way of specifying local constraints
  $\Rightarrow$ RCs and local constraints are the same thing.
- Thus there is no need to derive RCs from something else, and hence we do not need a processing account.
- But even if you don't buy my results, a processing account is still a tough sell.

Important Caveat

I only consider a peculiar class of RCs here. This class seems to encompass all syntactic RCs proposed in the literature, but semantic RCs are a different story (due to the "identity of meaning" condition, which has little to do with their RC-hood).

This Talk

The Basic Message

- You cannot have a grammar without RCs.
  You always get RCs for free because they are simply a different way of specifying local constraints
  $\Rightarrow$ RCs and local constraints are the same thing.
- Thus there is no need to derive RCs from something else, and hence we do not need a processing account.
- But even if you don't buy my results, a processing account is still a tough sell.

Important Caveat

I only consider a peculiar class of RCs here. This class seems to encompass all syntactic RCs proposed in the literature, but semantic RCs are a different story (due to the "identity of meaning" condition, which has little to do with their RC-hood).

Outline

MGs in 3 Simple Steps

If you want to make unassailable claims, you should prove them in as rigorous a way as possible ⇒ Minimalist Grammars (MGs; Stabler 1997) as a formalization of the Minimalist Program

A Definition of MGs for Syntacticians

- As usual, Merge and (phrasal) Move are the only structure building operations.
- As usual, lexical items equipped with finitely many features, but:
  - Features must be checked in a specific order.
  - A lexical item may carry the same feature several times.
  - Features are either category or movement features, and they come in one of two polarities: $+$ ("licensor") and $-$ ("licensee")
- Move is deterministic (i.e. it is always clear which constituent has to move where).

MGs in 3 Simple Steps

If you want to make unassailable claims, you should prove them in as rigorous a way as possible ⇒ Minimalist Grammars (MGs; Stabler 1997) as a formalization of the Minimalist Program

A Definition of MGs for Syntacticians

- As usual, Merge and (phrasal) Move are the only structure building operations.
- As usual, lexical items equipped with finitely many features, but:
  - Features must be checked in a specific order.
  - A lexical item may carry the same feature several times.
  - Features are either category or movement features, and they come in one of two polarities: $+$ ("licensor") and $-$ ("licensee")
- Move is deterministic (i.e. it is always clear which constituent has to move where).

A Toy Example (yes, it's really unsophisticated)

Lexicon

men :: $-\mathrm{N}$	like :: $+\mathrm{D}\ +\mathrm{D}\ -\mathrm{V}$
the :: $+\mathrm{N}\ -\mathrm{D}$	ε :: $+\mathrm{V}\ (+\mathrm{wh})\ -\mathrm{C}$
what :: $-\mathrm{D}\ -\mathrm{wh}$	

A Toy Example (yes, it's really unsophisticated)

Lexicon

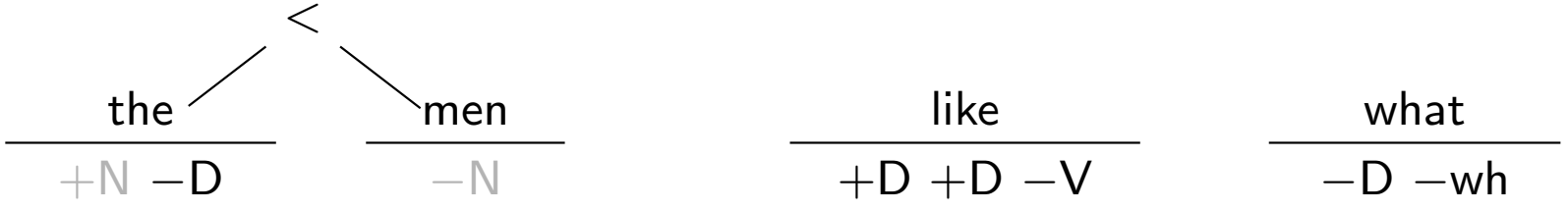
men :: $-\mathrm{N}$
the :: $+\mathrm{N}\ -\mathrm{D}$
what :: $-\mathrm{D}\ -\mathrm{wh}$
like :: $+\mathrm{D}\ +\mathrm{D}\ -\mathrm{V}$
ε :: $+\mathrm{V}\ (+\mathrm{wh})\ -\mathrm{C}$

$$\frac{\text{the}}{+\mathrm{N}\ -\mathrm{D}} \qquad \frac{\text{men}}{-\mathrm{N}} \qquad \frac{\text{like}}{+\mathrm{D}\ +\mathrm{D}\ -\mathrm{V}} \qquad \frac{\text{what}}{-\mathrm{D}\ -\mathrm{wh}}$$

A Toy Example (yes, it's really unsophisticated)

Lexicon

men :: $-\mathrm{N}$
the :: $+\mathrm{N}\ -\mathrm{D}$
what :: $-\mathrm{D}\ -\mathrm{wh}$
like :: $+\mathrm{D}\ +\mathrm{D}\ -\mathrm{V}$
ε :: $+\mathrm{V}\ (+\mathrm{wh})\ -\mathrm{C}$



A Toy Example (yes, it's really unsophisticated)

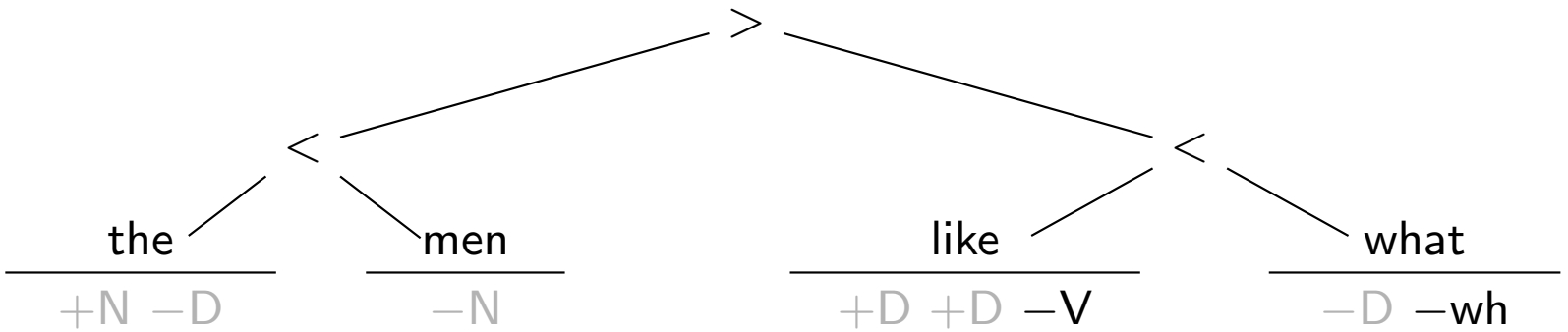
Lexicon

men :: $-\mathrm{N}$	like :: $+\mathrm{D}\ +\mathrm{D}\ -\mathrm{V}$
the :: $+\mathrm{N}\ -\mathrm{D}$	ε :: $+\mathrm{V}\ (+\mathrm{wh})\ -\mathrm{C}$
what :: $-\mathrm{D}\ -\mathrm{wh}$	

A Toy Example (yes, it's really unsophisticated)

Lexicon

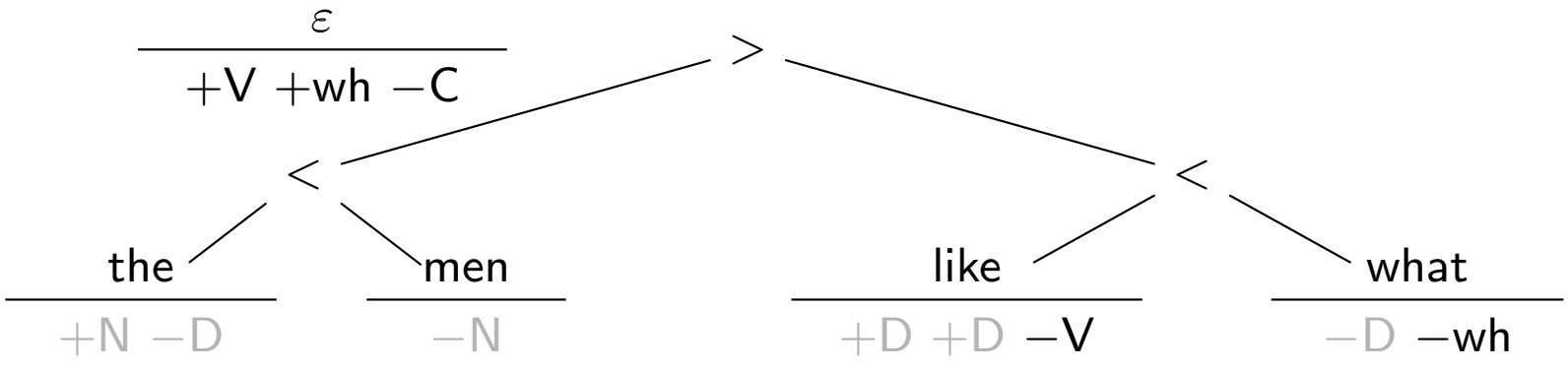
men :: $-\mathrm{N}$
the :: $+\mathrm{N}\ -\mathrm{D}$
what :: $-\mathrm{D}\ -\mathrm{wh}$
like :: $+\mathrm{D}\ +\mathrm{D}\ -\mathrm{V}$
ε :: $+\mathrm{V}\ (+\mathrm{wh})\ -\mathrm{C}$



A Toy Example (yes, it's really unsophisticated)

Lexicon

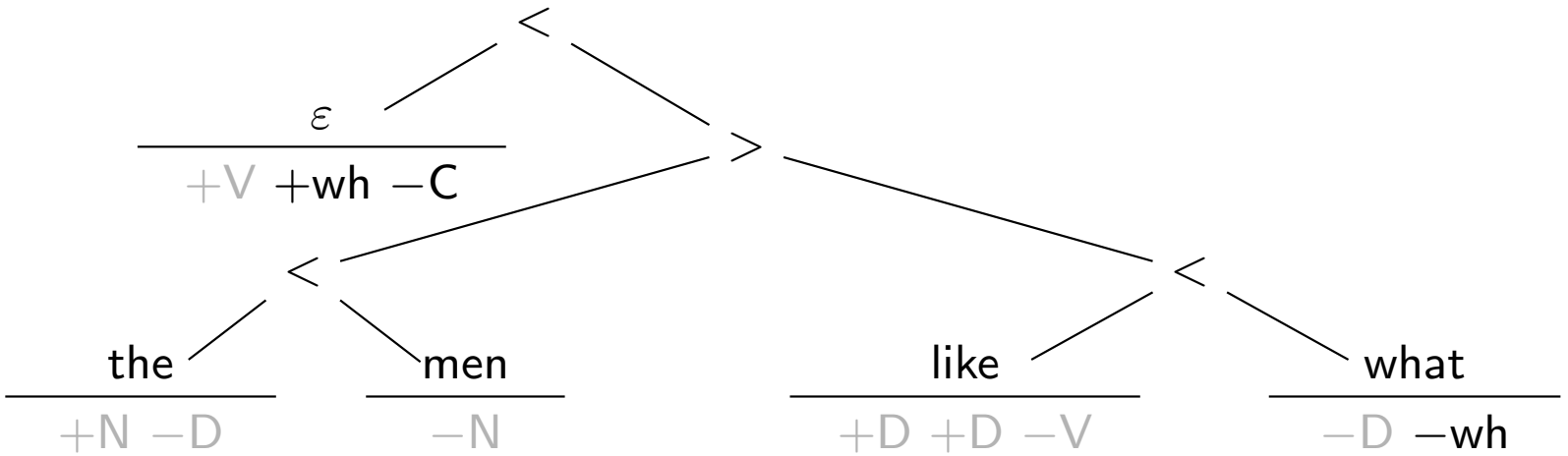
men :: $-\mathrm{N}$
the :: $+\mathrm{N}\ -\mathrm{D}$
what :: $-\mathrm{D}\ -\mathrm{wh}$
like :: $+\mathrm{D}\ +\mathrm{D}\ -\mathrm{V}$
ε :: $+\mathrm{V}\ (+\mathrm{wh})\ -\mathrm{C}$



A Toy Example (yes, it's really unsophisticated)

Lexicon

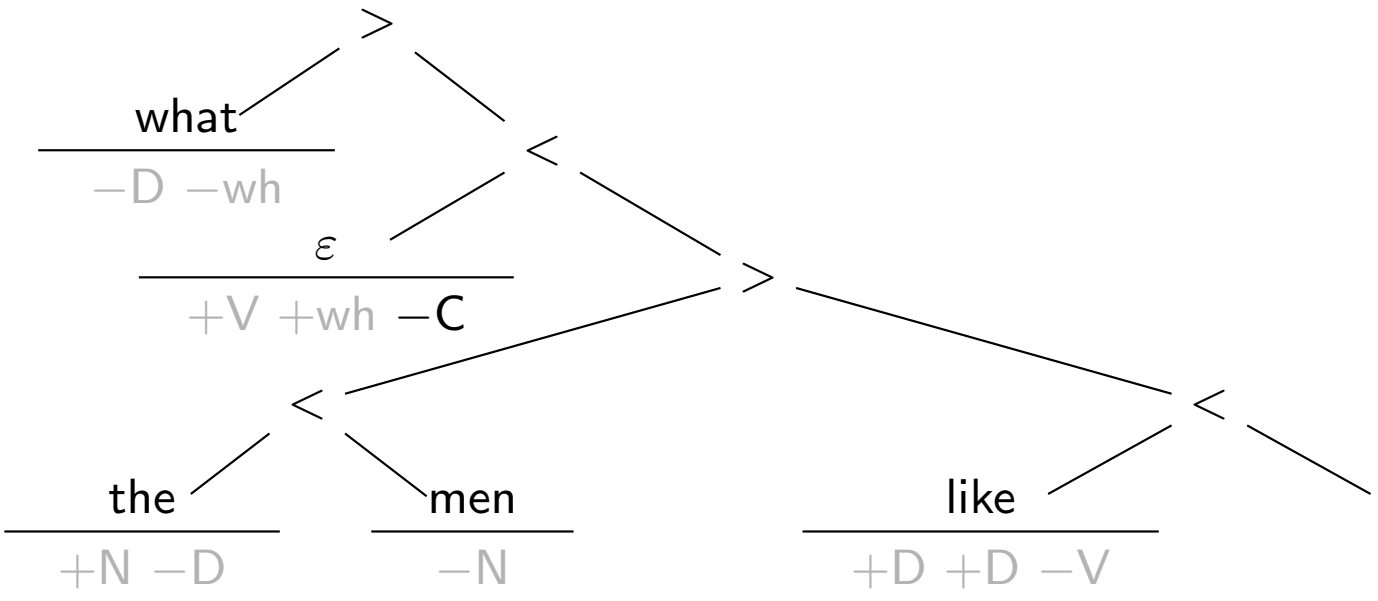
men :: $-\mathrm{N}$
the :: $+\mathrm{N}\ -\mathrm{D}$
what :: $-\mathrm{D}\ -\mathrm{wh}$
like :: $+\mathrm{D}\ +\mathrm{D}\ -\mathrm{V}$
ε :: $+\mathrm{V}\ (+\mathrm{wh})\ -\mathrm{C}$



A Toy Example (yes, it's really unsophisticated)

Lexicon

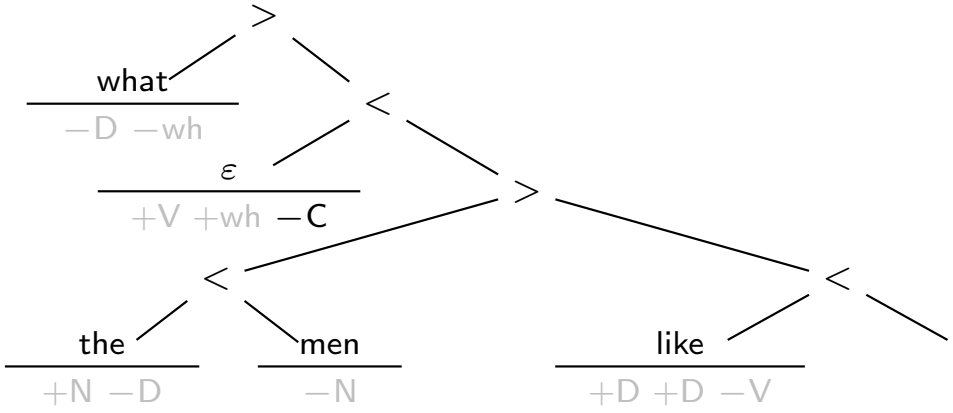
men :: $-\mathrm{N}$	like :: $+\mathrm{D}\ +\mathrm{D}\ -\mathrm{V}$
the :: $+\mathrm{N}\ -\mathrm{D}$	ε :: $+\mathrm{V}\ (+\mathrm{wh})\ -\mathrm{C}$
what :: $-\mathrm{D}\ -\mathrm{wh}$	

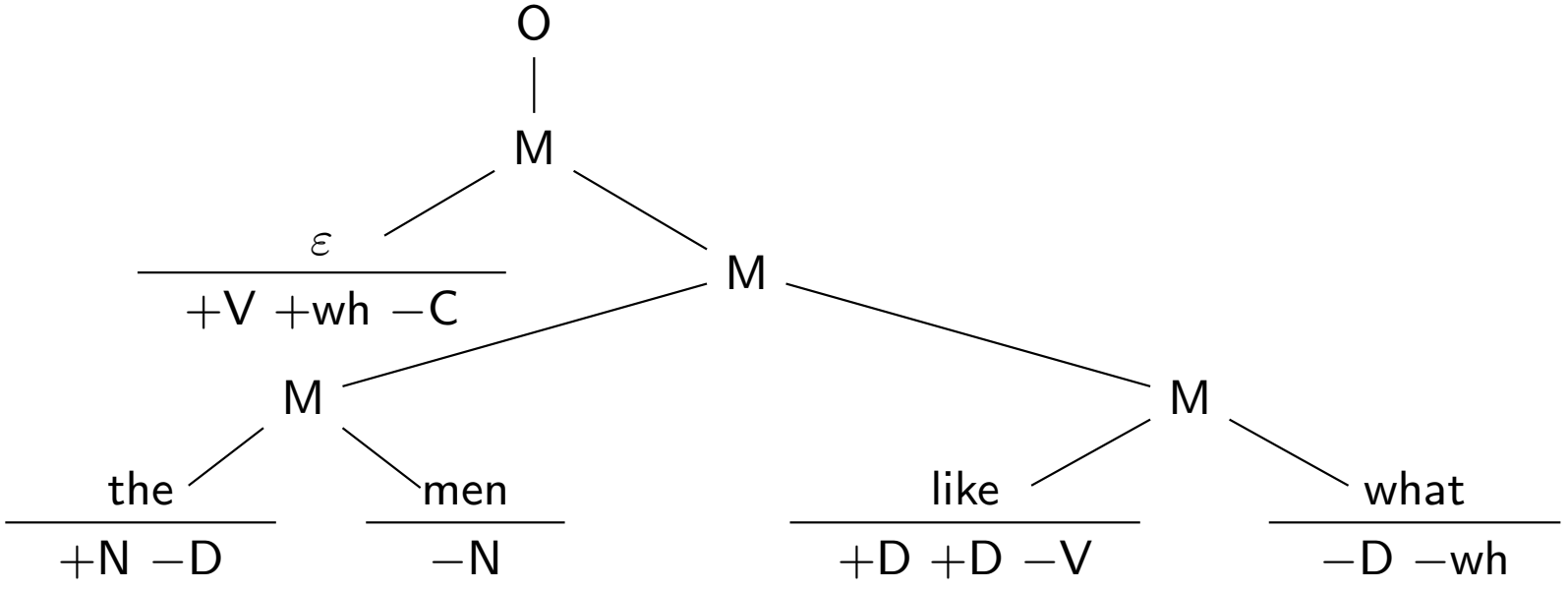


Derivation Trees

Useful Fact

Every MG is fully specified by its set of derivation trees (Kobele et al. 2007).





But this isn't Minimalism!

- Fair enough, but we can extend the simple MGs to make them more Minimalism-like:
  - feature system: $+/-$ interpretable, valued/unvalued
  - copying/traces
  - head movement, covert movement, ATB movement
  - Agree
  - phases/barriers
  - Relativized Minimality
  - islands constraints, complexity constraints
  - interface filters (type theory, linearity conditions)
  - syntactic binding conditions
  - *that*-trace filter, doubly-filled COMP filter
  - GPSG-style feature percolation
- Every extended MG can be emulated by some simple MG. (The only exception is copying/traces, but since this does not affect the derivation trees, it is irrelevant for our purposes.)

Some Neat Properties

- **MGs are a reasonable approximation of natural language**
  Just like various other linguistically motivated formalisms (Tree-Adjoining Grammar, Combinatory Categorial Grammar, Multiple Context-Free Grammars, Linear Context-Free Rewriting Systems), MGs define so-called mildly context-sensitive string languages.
  That so many different theories arrived at the same result suggests that mild context-sensitivity is indeed a non-trivial property of language. In particular, there are tree-based translations between some of these formalisms!
- **MGs can be defined by finite-state machines**
  The set of derivation trees licensed by an MG is a regular tree language. In the eyes of a formal language theorist, nothing beats being regular ;-)

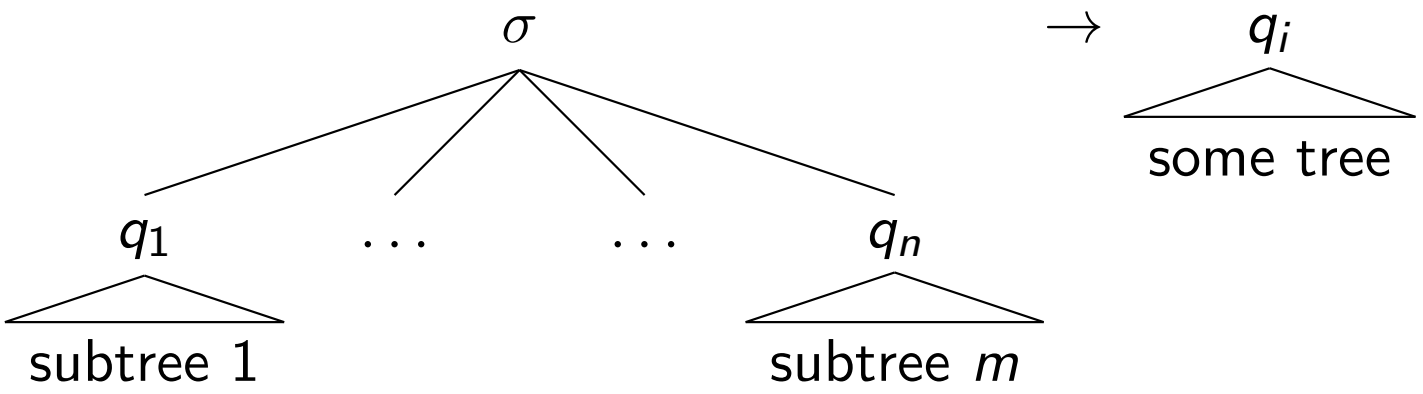
Linear Tree Transducers in Pictures

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.

$\Rightarrow$ generalization of transformational rule system of Aspects

Rules follow a specific schema:



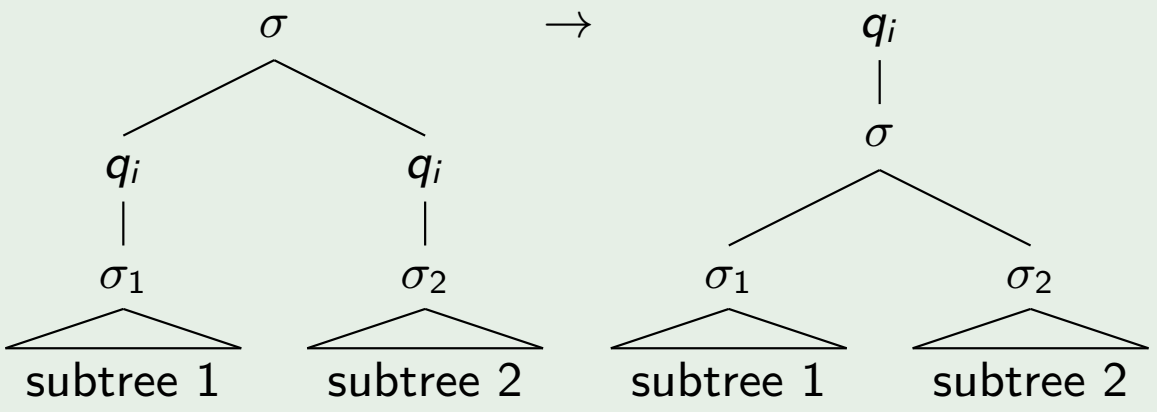
A Simple Example (Part 1)

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

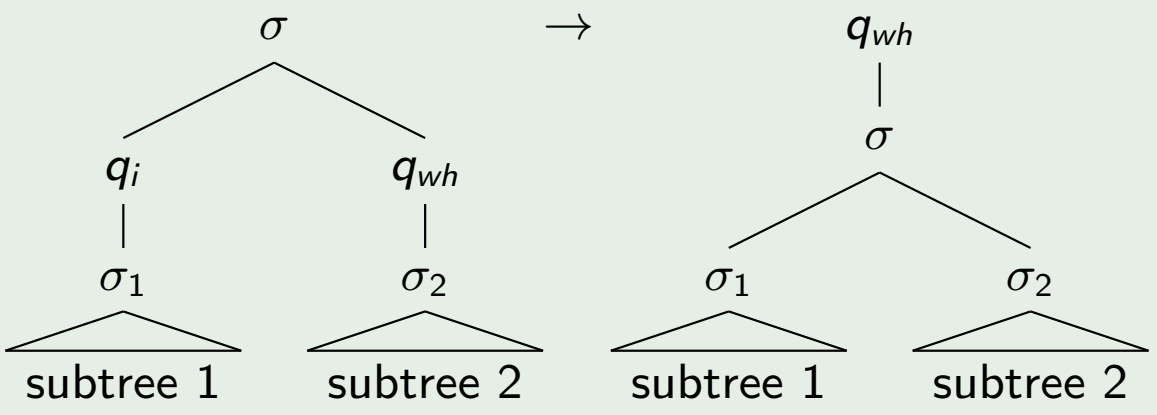
1) $\sigma \rightarrow$ q_i / σ

2) what$\rightarrow$ q_{wh} / t_{wh}

3)

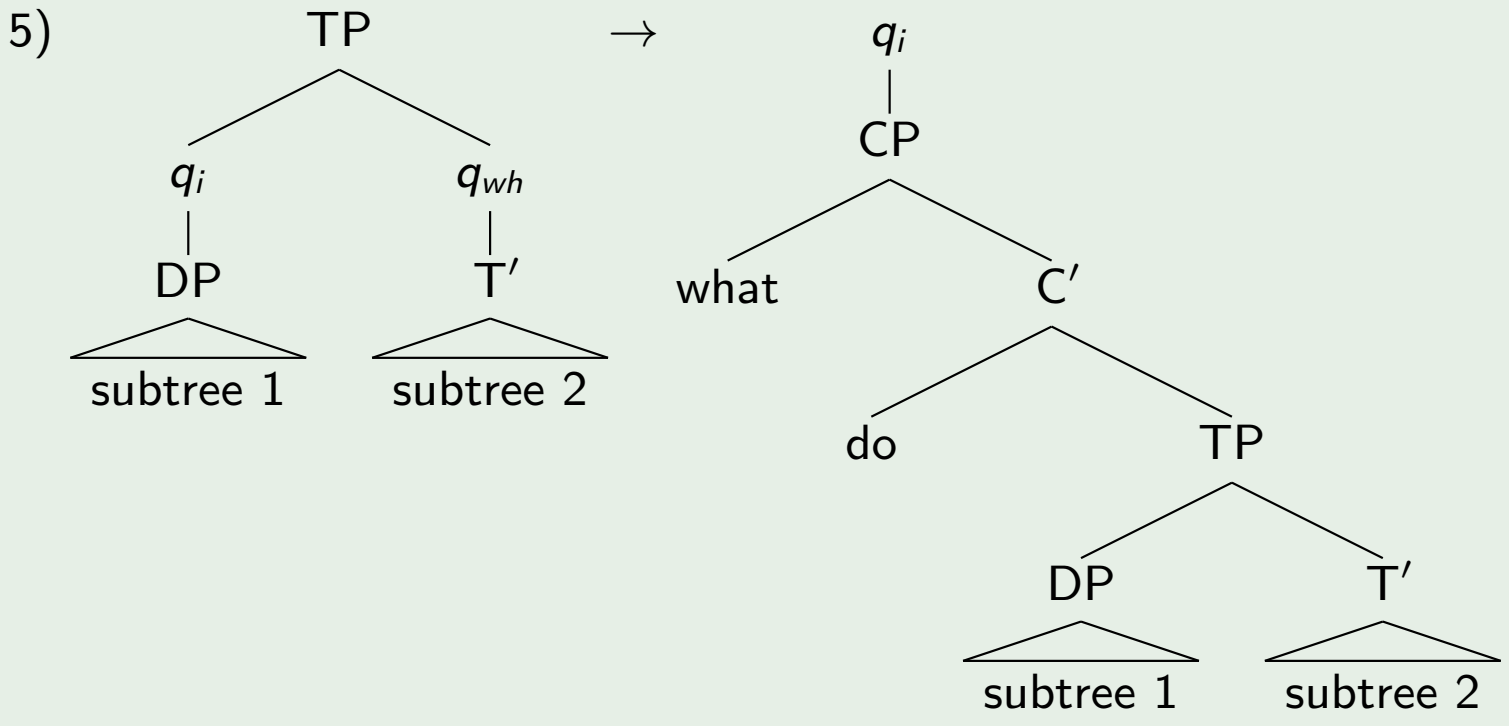


4)



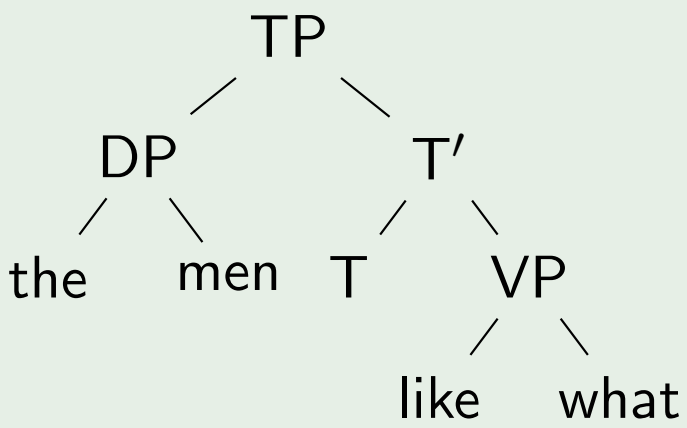
A Simple Example (Part 2)

A Transduction for Restricted wh-Movement, Rule 5



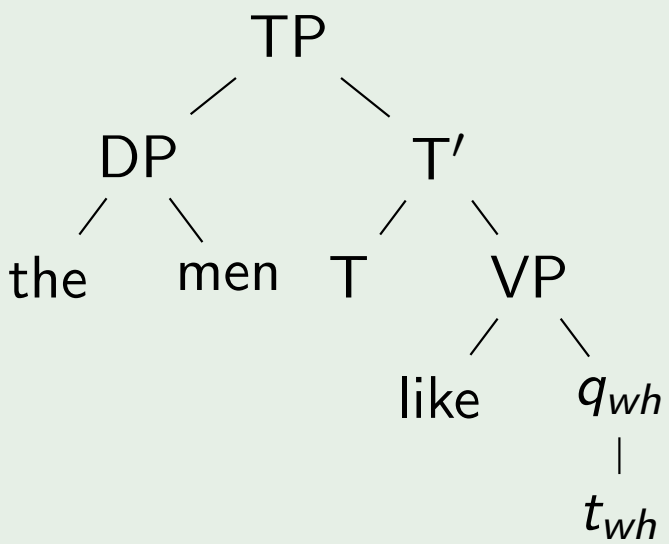
A Simple Example (Part 3)

A Transduction for Restricted wh-Movement, Application



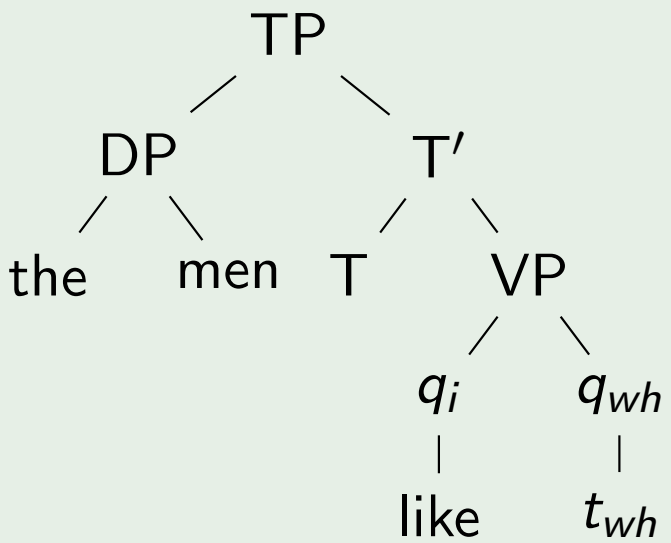
A Simple Example (Part 3)

A Transduction for Restricted wh-Movement, Application



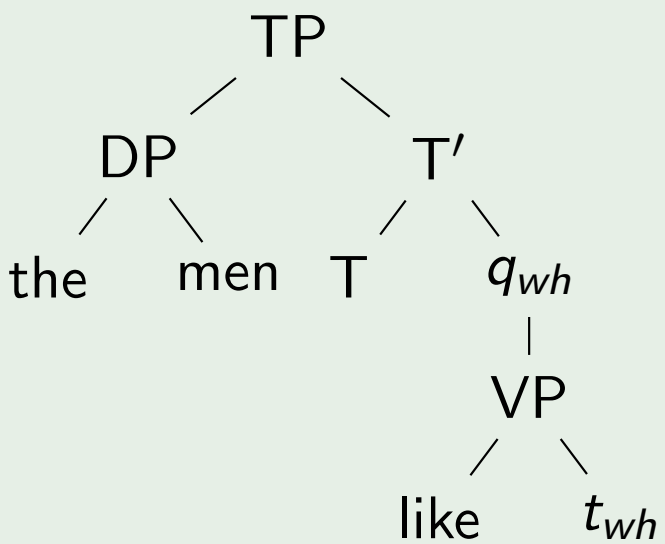
A Simple Example (Part 3)

A Transduction for Restricted wh-Movement, Application



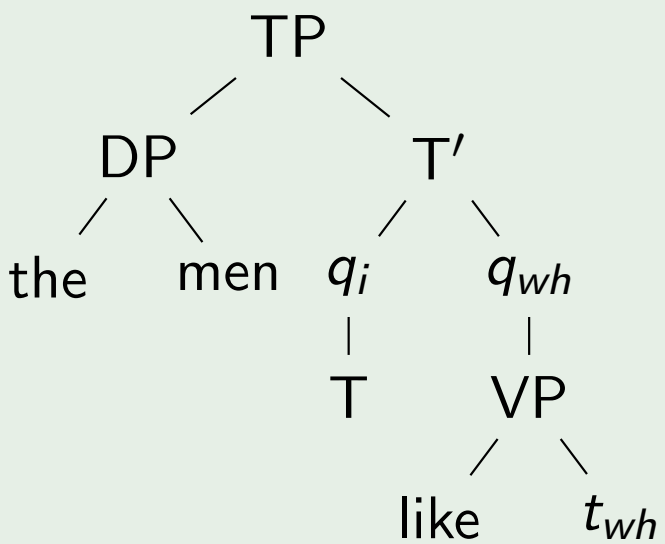
A Simple Example (Part 3)

A Transduction for Restricted wh-Movement, Application



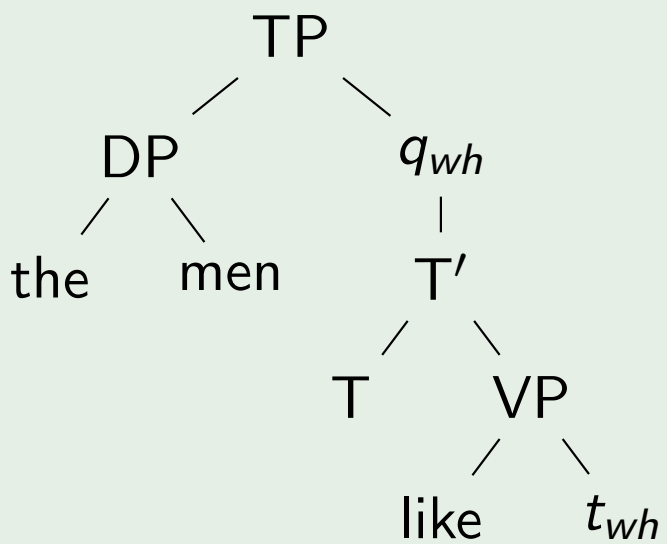
A Simple Example (Part 3)

A Transduction for Restricted wh-Movement, Application



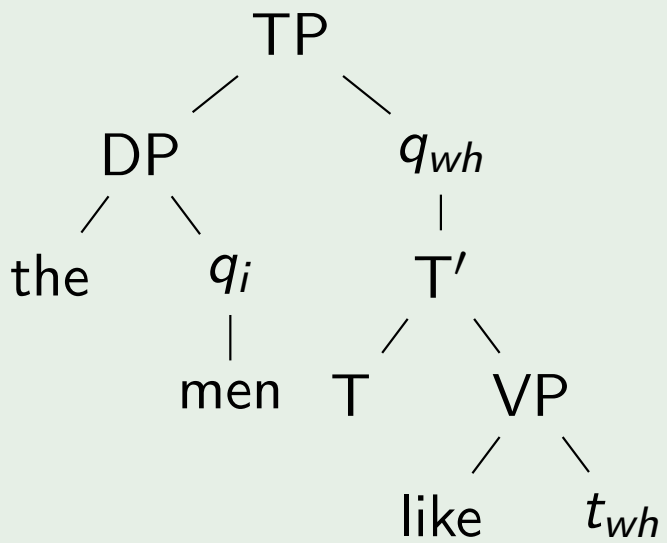
A Simple Example (Part 3)

A Transduction for Restricted wh-Movement, Application



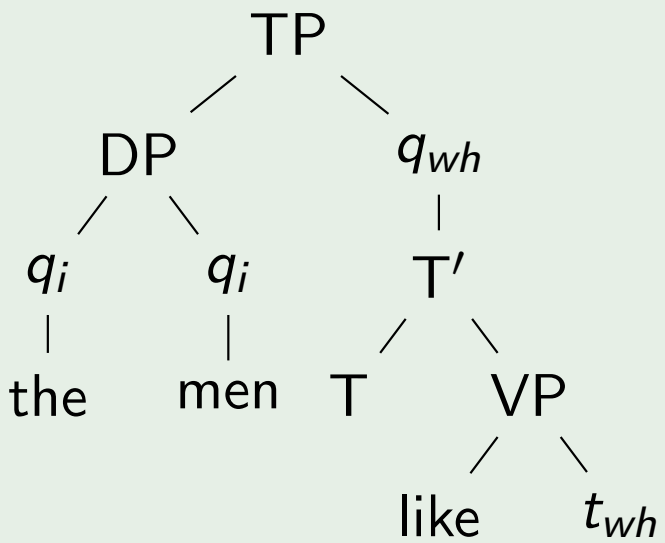
A Simple Example (Part 3)

A Transduction for Restricted wh-Movement, Application



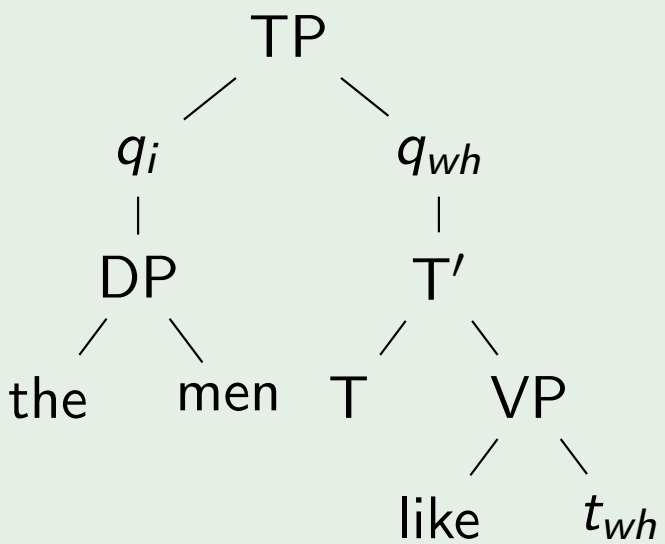
A Simple Example (Part 3)

A Transduction for Restricted wh-Movement, Application



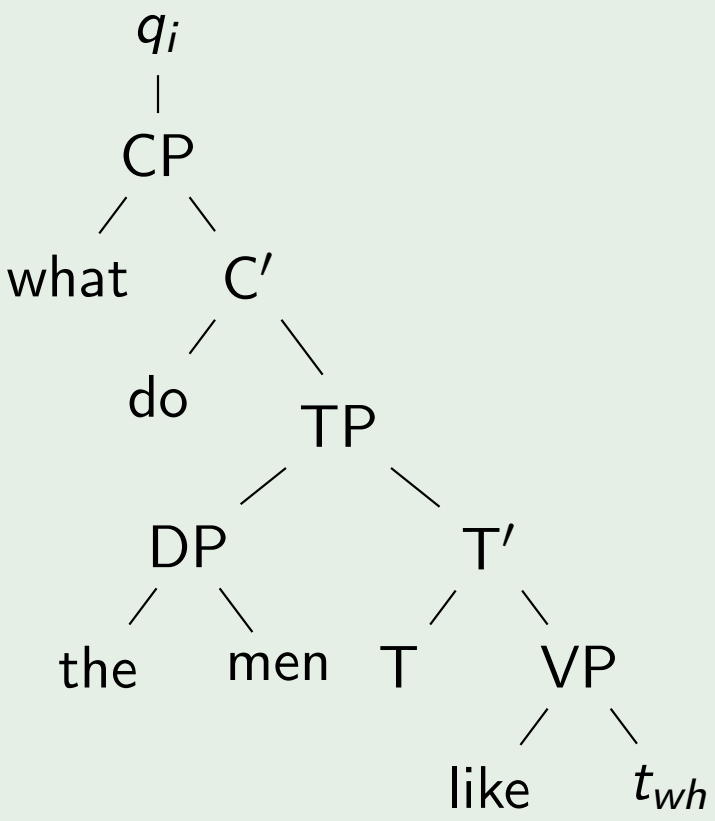
A Simple Example (Part 3)

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 output language of a transducer is intersected with some MDTL, the result is an MDTL, too. (Graf 2011; Kobele 2011)

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 output language of a transducer is intersected with some MDTL, the result is an MDTL, too. (Graf 2011; Kobele 2011)

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 output language of a transducer is intersected with some MDTL, the result is an MDTL, too. (Graf 2011; Kobele 2011)

How to Model RCs with MGs and Transducers

Strategy

For a given reference-set constraint C, exhibit

- an MG G that generates the input language, and
- a sequence τ of transducers that computes the same mapping from inputs to optimal outputs as C.

- Due to the mathematical properties of transducers, the result of feeding the MDTL of G into τ is an MDTL.
- Hence it can be generated by some MG.
- Hence C can be enforced without reference-set computation.

Every MG can trivially be decomposed into a combination of an MG plus a reference-set constraint, so we get the desired equivalence between reference-set constraints and local constraints.

How to Model RCs with MGs and Transducers

Strategy

For a given reference-set constraint C, exhibit

- an MG G that generates the input language, and
- a sequence τ of transducers that computes the same mapping from inputs to optimal outputs as C.

- Due to the mathematical properties of transducers, the result of feeding the MDTL of G into τ is an MDTL.
- Hence it can be generated by some MG.
- Hence C can be enforced without reference-set computation.

Every MG can trivially be decomposed into a combination of an MG plus a reference-set constraint, so we get the desired equivalence between reference-set constraints and local constraints.

How to Model RCs with MGs and Transducers

Strategy

For a given reference-set constraint C, exhibit

- an MG G that generates the input language, and
- a sequence τ of transducers that computes the same mapping from inputs to optimal outputs as C.

- Due to the mathematical properties of transducers, the result of feeding the MDTL of G into τ is an MDTL.
- Hence it can be generated by some MG.
- Hence C can be enforced without reference-set computation.

Every MG can trivially be decomposed into a combination of an MG plus a reference-set constraint, so we get the desired equivalence between reference-set constraints and local constraints.

Underspecification-and-Filtration

All my transducer implementations of reference-set constraints follow a strategy I call underspecification-and-filtration. Underspecification is used for computing the reference-set, and filtration for the economy metric.

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 transducer that turns optimal candidates into suboptimal ones.

Underspecification-and-Filtration

All my transducer implementations of reference-set constraints follow a strategy I call underspecification-and-filtration. Underspecification is used for computing the reference-set, and filtration for the economy metric.

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 transducer that turns optimal candidates into suboptimal ones.

Shortest Derivation Principle (SDP)

The SDP is a rather complicated constraint to implement, but it is the prototypical case of a race-like constraint and also generalizes to other reference-set constraints.

SDP

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

Empirical usage in Collins (1994): Why do we find the following contrast? In particular, what rules out (1b)?

(1) a. Who$_i$ did John take [$_{DP_j}$ a picture of t_i]?

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

Shortest Derivation Principle (SDP)

The SDP is a rather complicated constraint to implement, but it is the prototypical case of a race-like constraint and also generalizes to other reference-set constraints.

SDP

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

Empirical usage in Collins (1994): Why do we find the following contrast? In particular, what rules out (1b)?

(1) a. Who$_i$ did John take [$_{DP_j}$ a picture of t_i]?

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

Shortest Derivation Principle (SDP)

The SDP is a rather complicated constraint to implement, but it is the prototypical case of a race-like constraint and also generalizes to other reference-set constraints.

SDP

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

Empirical usage in Collins (1994): Why do we find the following contrast? In particular, what rules out (1b)?

(1) 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 (1b)

There are (supposedly) only two derivations for (1b).

- CED violation in (2c) $\Rightarrow$ ruled out independently

(2) 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]]]

- longer than (2) $\Rightarrow$ job for SDP

(3) 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]]]

Derivations for (1b)

There are (supposedly) only two derivations for (1b).

- CED violation in (2c) ⇒ ruled out independently

(2) 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]]]

- longer than (2) ⇒ job for SDP

(3) 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]]]

Derivations for (1b)

There are (supposedly) only two derivations for (1b).

- CED violation in (2c) ⇒ ruled out independently

(2) 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]]]

- longer than (2) ⇒ job for SDP

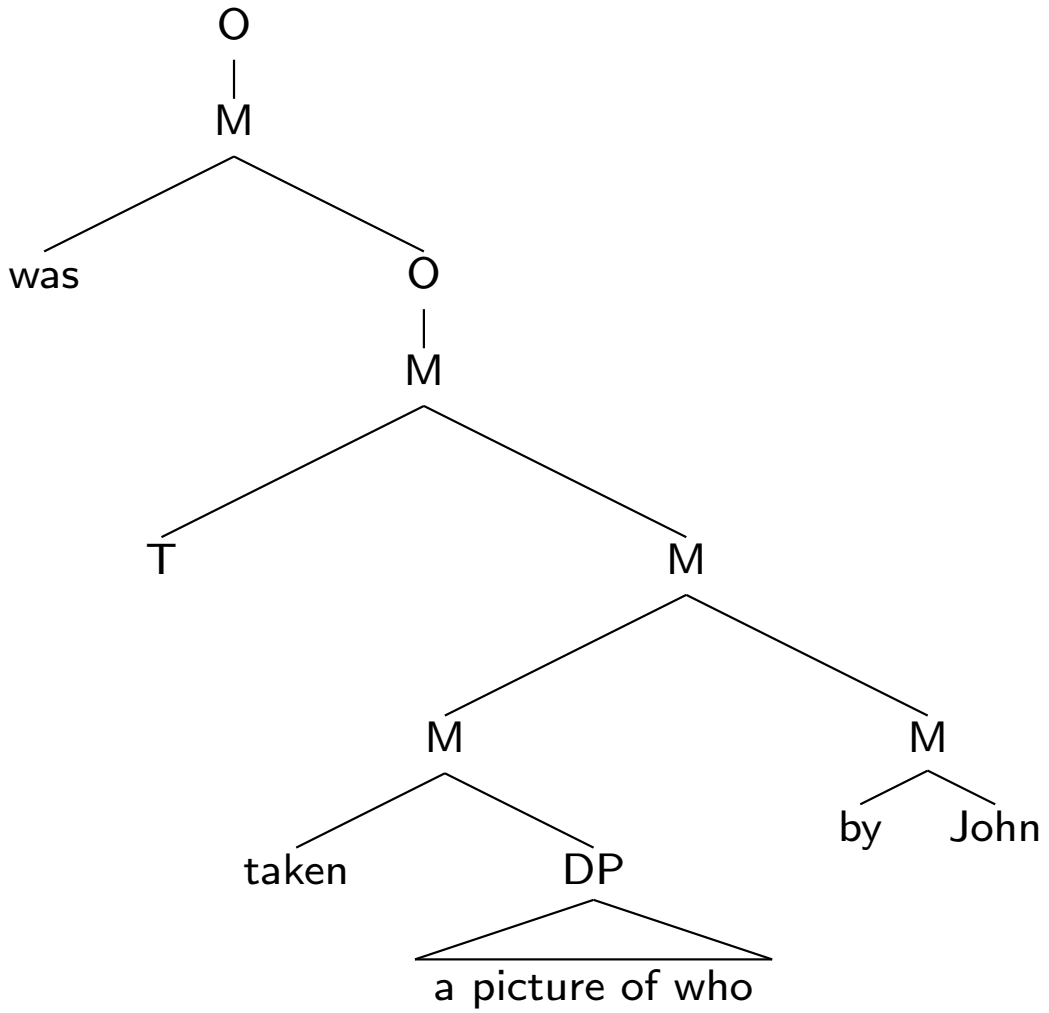
(3) a. [$_{VP}$ taken [$_{DP_j}$ a picture of who$_i$] by John]

b. [$_{VP}$ who$_i$ taken [$_{DP_j}$ a picture of t_i] by John]

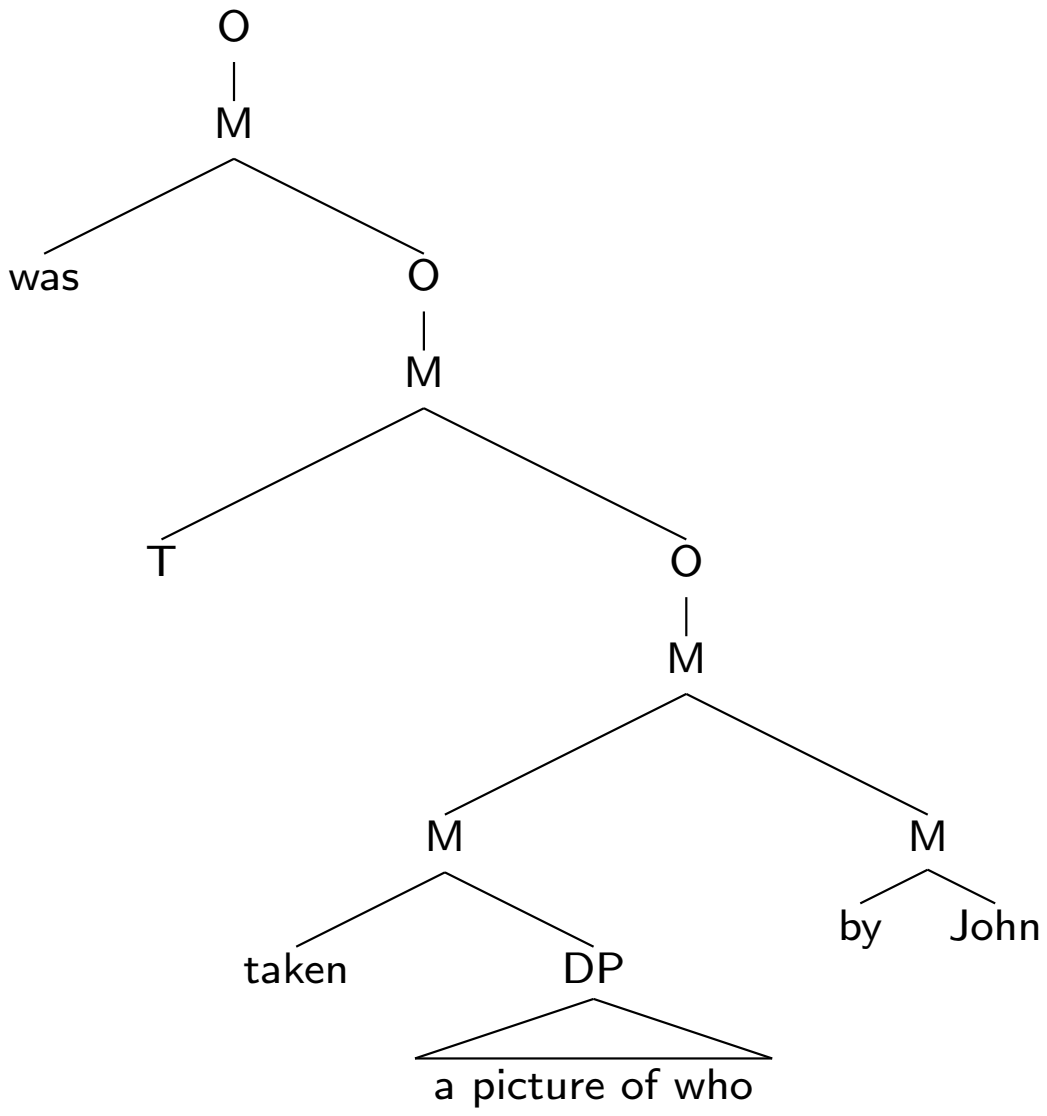
c. [$_{TP}$ [$_{DP_j}$ a picture of t_i] T [$_{VP}$ who$_i$ taken t_j by John]]

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

Derivation Tree of (2)

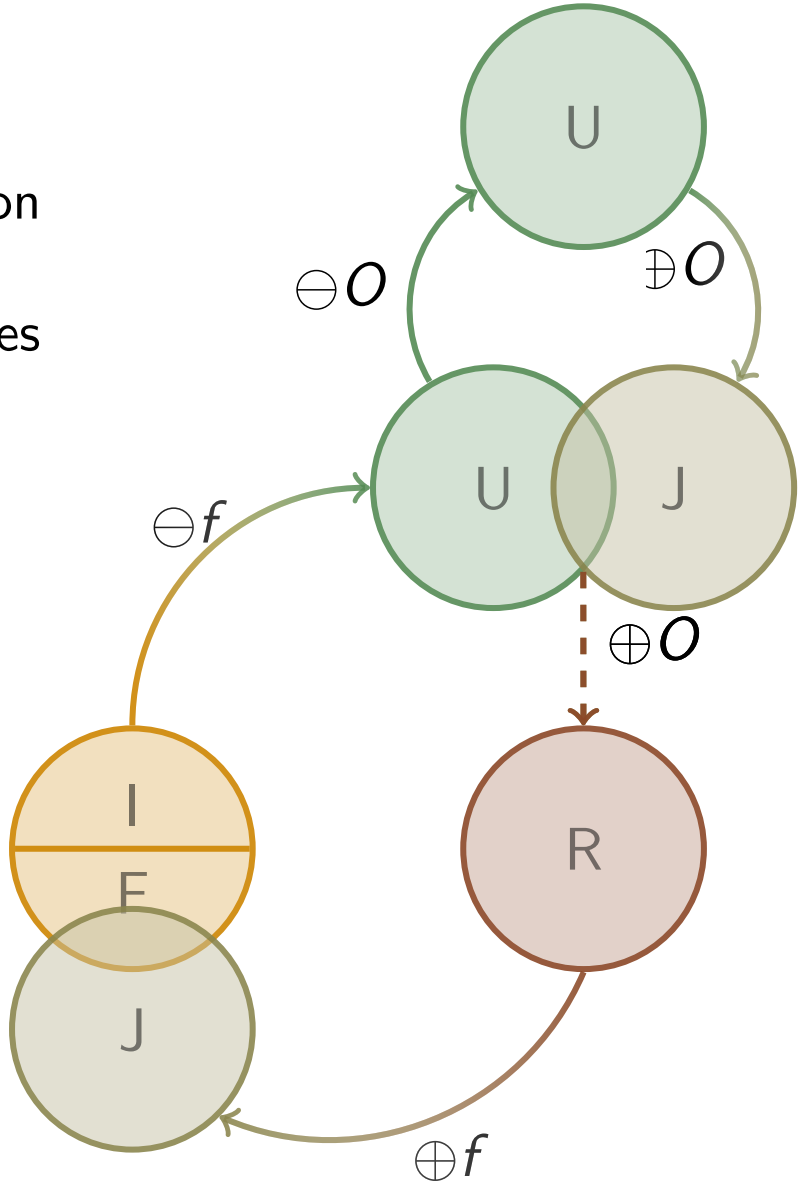


Derivation Tree of (3)



Transducer Decomposition of SDP

F	filter
I	input derivations
J	"junk" $\approx$ overgeneration
R	ranked set
U	underspecified structures
$\ominus f$	remove features
$\oplus f$	add features
$\ominus O$	remove Move nodes
$\oplus O$	add Move nodes



A Step-by-Step Example

Let's set up a scenario in the spirit of the original problem.

Step 1: Define an appropriate MG (we only need a lexicon); the new feature *s* is used to "scramble" DPs into SpecVP

Lexicon

John :: $-\mathrm{D}\ (-\mathrm{s})\ (-\mathrm{nom})$
who :: $-\mathrm{N}\ (-\mathrm{s})\ (-\mathrm{nom})\ (-\mathrm{wh})$
a picture of :: $+\mathrm{N}\ -\mathrm{D}\ (-\mathrm{s})\ (-\mathrm{nom})$

by :: $+\mathrm{D}\ -\mathrm{P}$
taken :: $+\mathrm{D}\ (+\mathrm{P})\ (+\mathrm{s})\ -\mathrm{V}$
was :: $+\mathrm{V}\ +\mathrm{nom}\ -\mathrm{T}$
ε :: $+\mathrm{T}\ (+\mathrm{wh})\ -\mathrm{C}$

A Step-by-Step Example

Let's set up a scenario in the spirit of the original problem.

Step 1: Define an appropriate MG (we only need a lexicon); the new feature *s* is used to "scramble" DPs into SpecVP

Lexicon

John :: $-\mathrm{D}\ (-\mathrm{s})\ (-\mathrm{nom})$
who :: $-\mathrm{N}\ (-\mathrm{s})\ (-\mathrm{nom})\ (-\mathrm{wh})$
a picture of :: $+\mathrm{N}\ -\mathrm{D}\ (-\mathrm{s})\ (-\mathrm{nom})$

by :: $+\mathrm{D}\ -\mathrm{P}$
taken :: $+\mathrm{D}\ (+\mathrm{P})\ (+\mathrm{s})\ -\mathrm{V}$
was :: $+\mathrm{V}\ +\mathrm{nom}\ -\mathrm{T}$
ε :: $+\mathrm{T}\ (+\mathrm{wh})\ -\mathrm{C}$

Example: Step 1 Simplification

To reduce the complexity of the example, let us assume that only the following five sentences are deemed well-formed by the grammar (so there have to be extra constraints that rule out all the other possible combinations).

(4) who$_{wh}$ C [$_{\text{DP}}$ a picture of t_{wh}] was [$_{\text{VP}}$ t_{DP} [$_{\text{VP}}$ by John taken t_{DP}]]

(5) who$_{wh}$ C [$_{\text{DP}}$ a picture of t_{wh}] was [$_{\text{VP}}$ t_{wh} [$_{\text{VP}}$ by John taken t_{DP}]]

(6) who$_{wh}$ C [$_{\text{DP}}$ a picture of t_{wh}] was [$_{\text{VP}}$ [$_{\text{PP}}$ by John] [$_{\text{VP}}$ t_{PP} taken t_{DP}]]

(7) who$_{wh}$ C [$_{\text{DP}}$ a picture of t_{wh}] was [$_{\text{VP}}$ by John taken t_{DP}]

(8) C [$_{\text{DP}}$ a picture of who] was [$_{\text{VP}}$ by John taken t_{DP}]

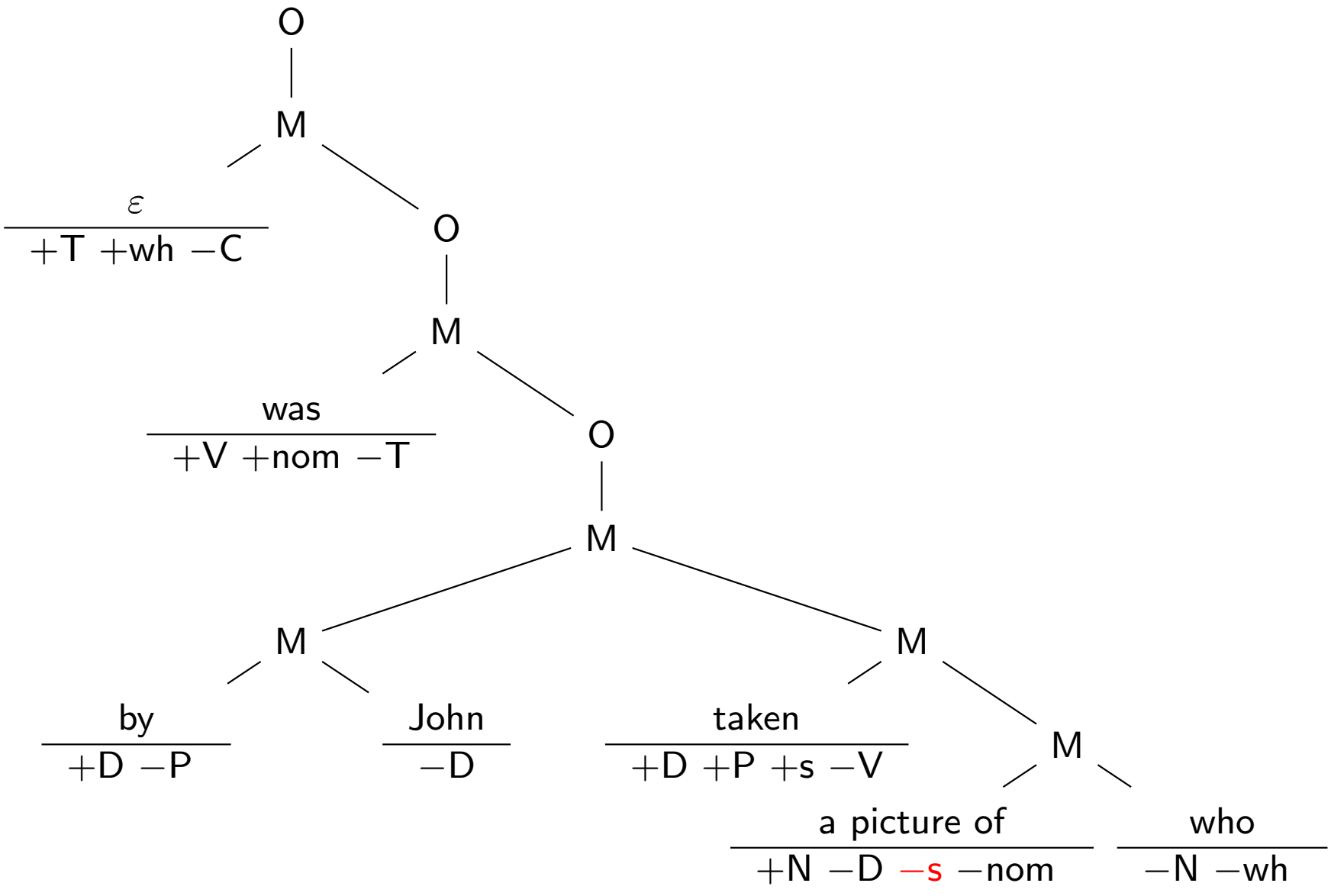
What the SDP Should Accomplish

- Desired behavior of SDP
  - (4)–(7) compete against each other, but not against (8)
  - (7) and (8) are the only licit constructions;
    (7) is ruled out later on for independent reasons
- What we have to do
  - Define a transduction that computes the correct reference-set for each sentence (composition of $\ominus f$, $\ominus O$ and $\oplus O$)
  - Define a ranking that defines the economy metric ($\oplus O$ applied to the right reference set)
  - Avoid overgeneration (several instances of filtering)

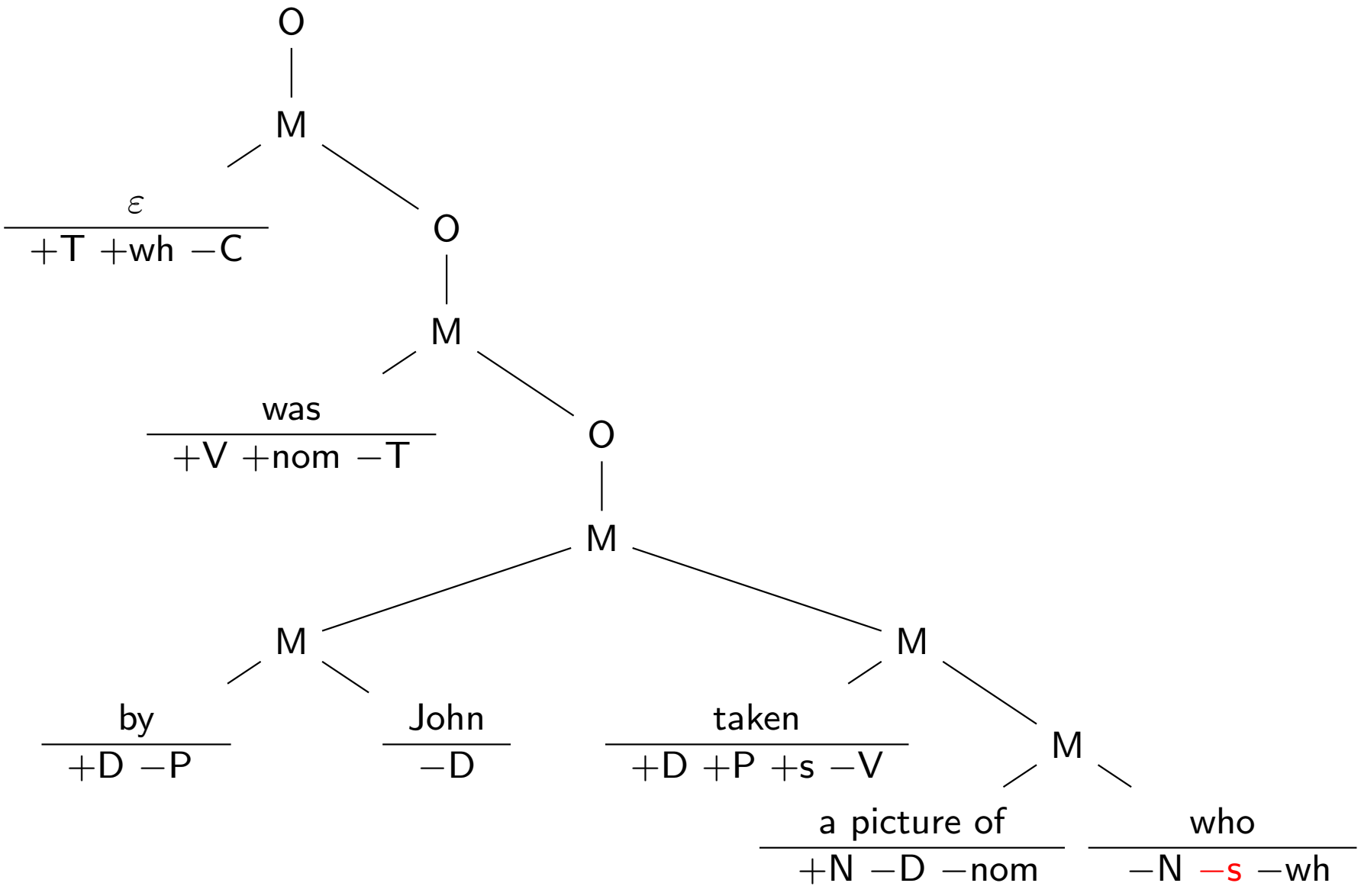
What the SDP Should Accomplish

- Desired behavior of SDP
  - (4)–(7) compete against each other, but not against (8)
  - (7) and (8) are the only licit constructions;
    (7) is ruled out later on for independent reasons
- What we have to do
  - Define a transduction that computes the correct reference-set for each sentence (composition of $\ominus f$, $\ominus O$ and $\oplus O$)
  - Define a ranking that defines the economy metric ($\oplus O$ applied to the right reference set)
  - Avoid overgeneration (several instances of filtering)

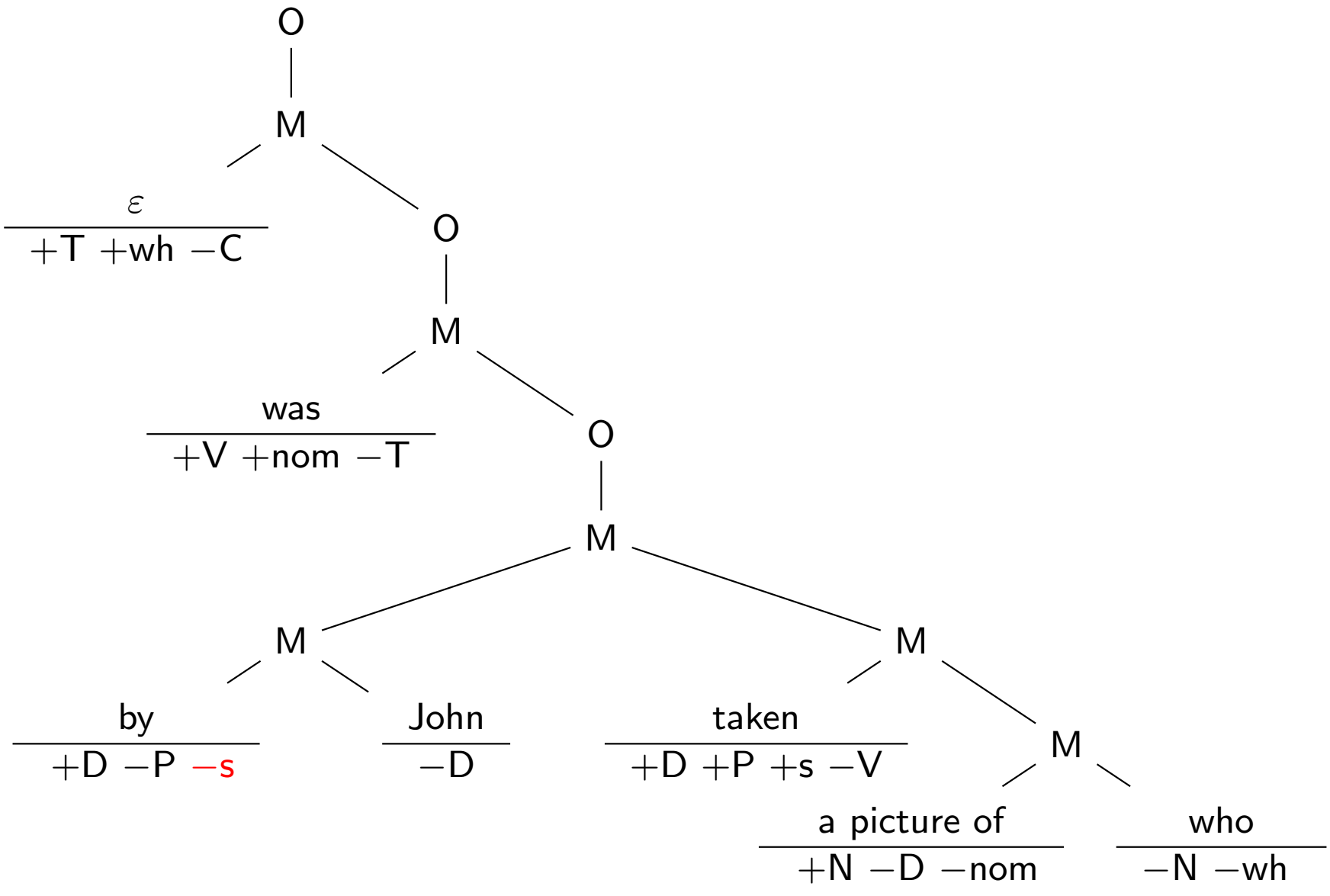
Step 2: Derivation Trees for (4)–(6)



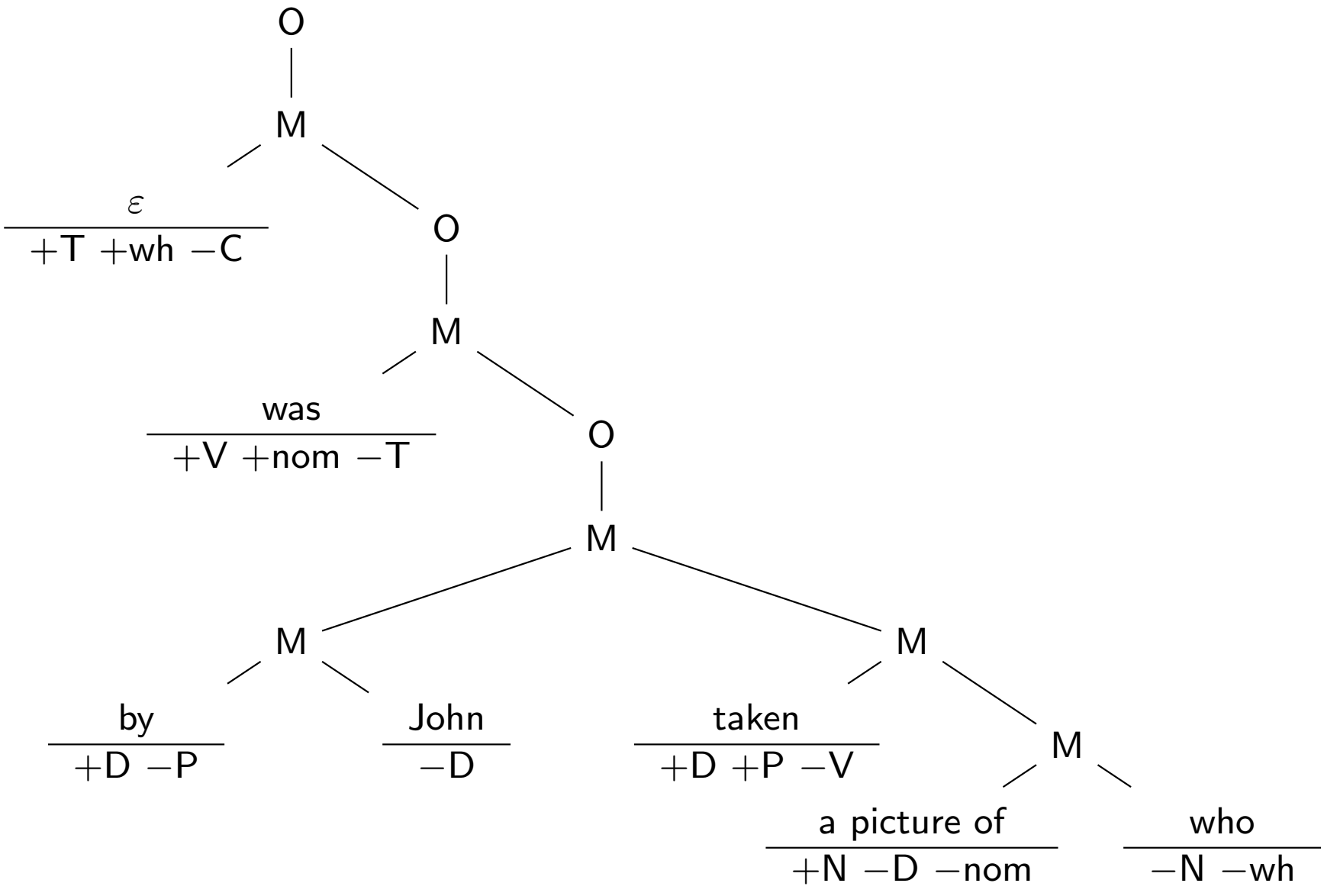
Step 2: Derivation Trees for (4)–(6)



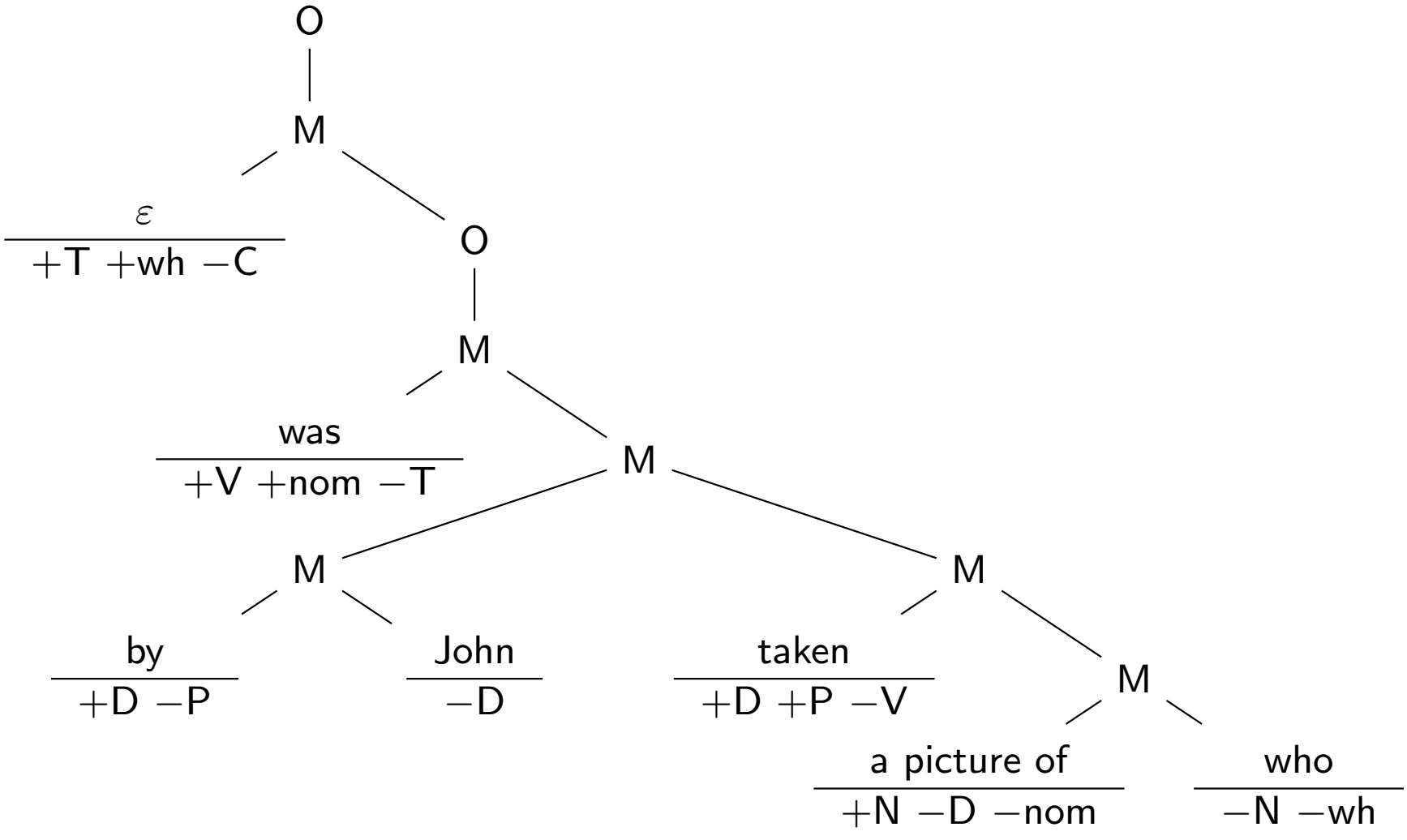
Step 2: Derivation Trees for (4)–(6)



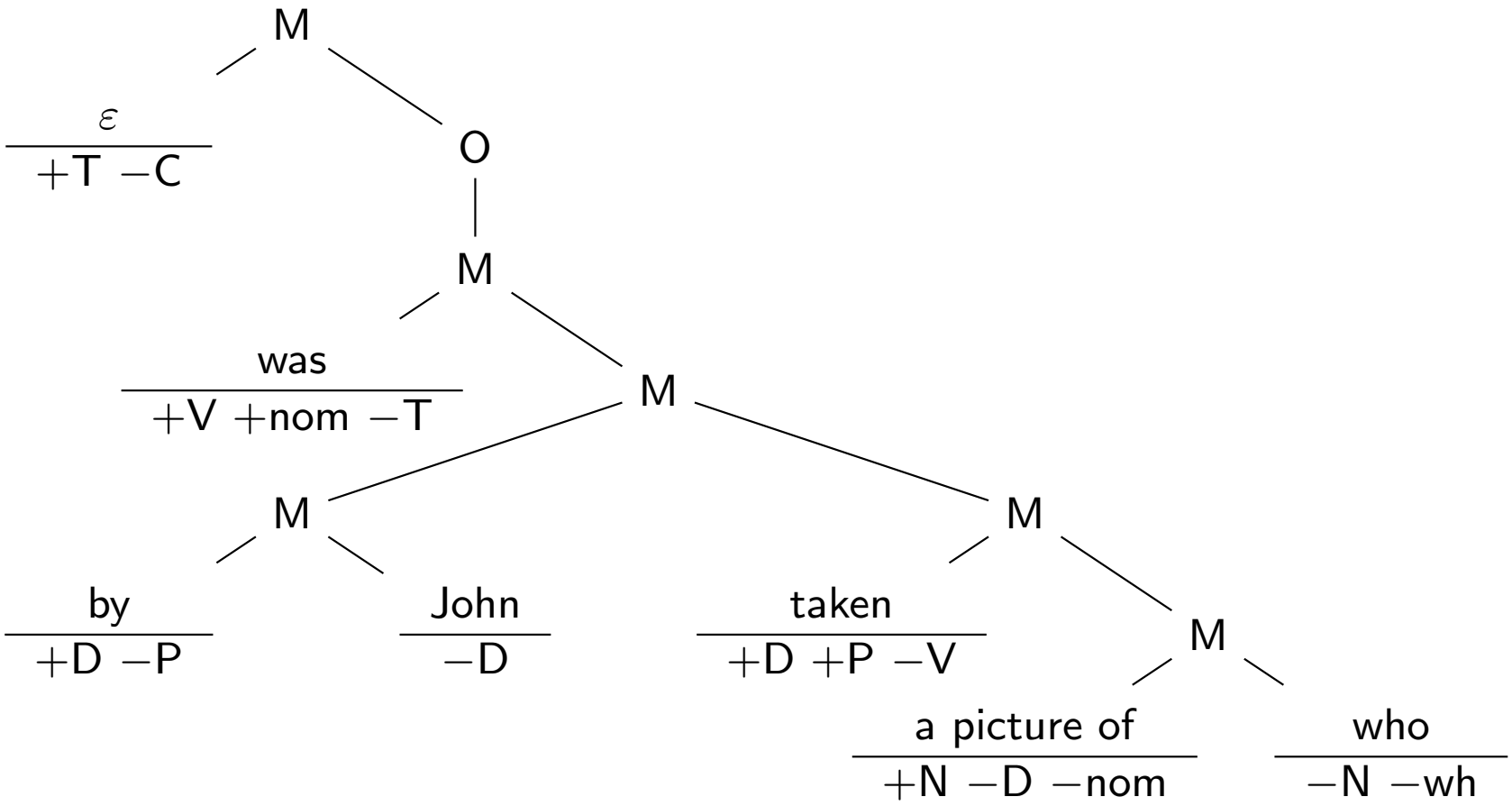
Step 2: Output of $\ominus f$ applied to (4)–(6)



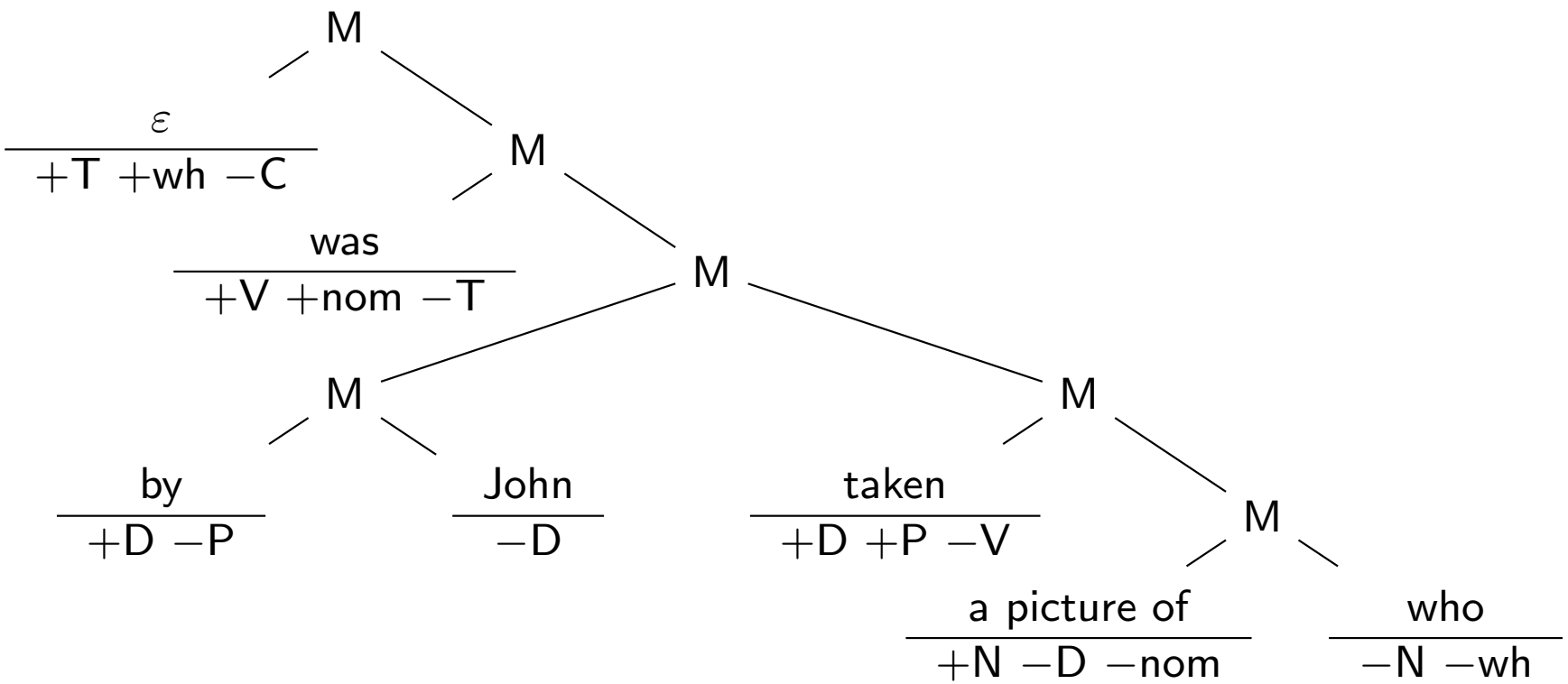
Step 2: Derivation tree of (7) = $\ominus f(7)$



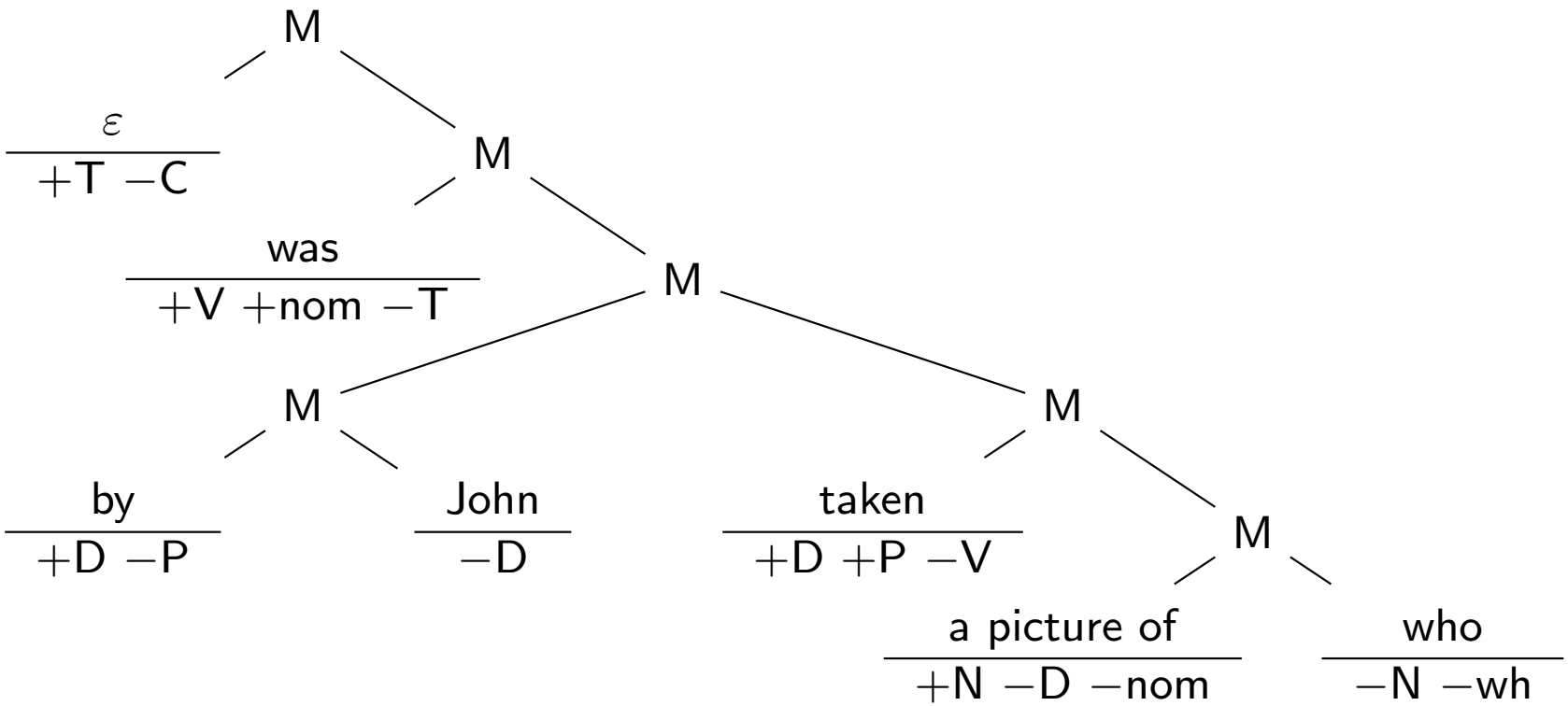
Step 2: Derivation tree of (8) = $\ominus f(8)$



Step 3: Application of $\ominus O$ to output of $\ominus f$ for (4)–(7)



Step 3: Application of $\ominus O$ to output of $\ominus f$ for (8)



Step 4: Reinsert Movement Nodes via $\oplus O$ and Filter

By reinserting movement nodes and throwing away all trees that we have not encountered so far, the Merge-only derivations are once again turned into one of the following three:

- $\ominus f(4) = \ominus f(5) = \ominus f(6)$
- $\ominus f(7)$
- $\ominus f(8)$

Crucial Fact

Removing and reinserting movement nodes has changed the reference sets for some trees. We now have:

- $\mathrm{RS}(4) = \mathrm{RS}(5) = \mathrm{RS}(6) = \mathrm{RS}(7) = \{\ominus f(4), \ominus f(7)\}$
- $\mathrm{RS}(8) = \{\ominus f(8)\}$

This is exactly the distribution of reference sets we want.

Step 4: Reinsert Movement Nodes via $\oplus O$ and Filter

By reinserting movement nodes and throwing away all trees that we have not encountered so far, the Merge-only derivations are once again turned into one of the following three:

- $\ominus f(4) = \ominus f(5) = \ominus f(6)$
- $\ominus f(7)$
- $\ominus f(8)$

Crucial Fact

Removing and reinserting movement nodes has changed the reference sets for some trees. We now have:

- $\mathrm{RS}(4) = \mathrm{RS}(5) = \mathrm{RS}(6) = \mathrm{RS}(7) = \{\ominus f(4), \ominus f(7)\}$
- $\mathrm{RS}(8) = \{\ominus f(8)\}$

This is exactly the distribution of reference sets we want.

Step 5: Ranking via $\oplus O$

The economy metric for the SDP is straight-forward: pick the candidate with the fewest Move-nodes. But how is this implemented as a transducer?

Transducers for Rankings

- We say that a derivation d is **optimal** wrt transducer τ if there is no competing derivation d' such that τ rewrites d' as d.
- A transducer τ is **well-founded** if every reference set contains at least one derivation that is optimal wrt τ.
- Given a well-founded transducer τ, we can construct a transducer τ' that rewrites no derivation as a candidate that is not optimal wrt τ (Jäger 2002).

Transducer $\oplus O$ is well-founded and gives us the right metric:
A derivation is optimal wrt $\oplus O$ if there is no competing derivation with fewer instances of Move.

Step 5: Ranking via $\oplus O$

The economy metric for the SDP is straight-forward: pick the candidate with the fewest Move-nodes. But how is this implemented as a transducer?

Transducers for Rankings

- We say that a derivation d is **optimal** wrt transducer τ if there is no competing derivation d' such that τ rewrites d' as d.
- A transducer τ is **well-founded** if every reference set contains at least one derivation that is optimal wrt τ.
- Given a well-founded transducer τ, we can construct a transducer τ' that rewrites no derivation as a candidate that is not optimal wrt τ (Jäger 2002).

Transducer $\oplus O$ is well-founded and gives us the right metric: A derivation is optimal wrt $\oplus O$ if there is no competing derivation with fewer instances of Move.

Step 5: Ranking via $\oplus O$

The economy metric for the SDP is straight-forward: pick the candidate with the fewest Move-nodes. But how is this implemented as a transducer?

Transducers for Rankings

- We say that a derivation d is **optimal** wrt transducer τ if there is no competing derivation d' such that τ rewrites d' as d.
- A transducer τ is **well-founded** if every reference set contains at least one derivation that is optimal wrt τ.
- Given a well-founded transducer τ, we can construct a transducer τ' that rewrites no derivation as a candidate that is not optimal wrt τ (Jäger 2002).

Transducer $\oplus O$ is well-founded and gives us the right metric:
A derivation is optimal wrt $\oplus O$ if there is no competing derivation with fewer instances of Move.

Step 6: Application of $\oplus f$ and Filtering

- To ensure that the end result of the transduction is a Minimalist derivation tree licensed by the original grammar, we have to reinstantiate the features that $\ominus f$ stripped away $\Rightarrow \oplus f$
- $\oplus f \approx \oplus O$ for features: Non-deterministically add random features and throw away all outputs we haven't encountered before.
- In the case at hand, no features have to be reinstantiated because we only stripped away s-features, yet no instances of s-movement occur in the trees that $\ominus f(7)$ and $\ominus f(8)$ were obtained from. So we're all done!

Uhm... What Just Happened?

- A sequence of mini-transducers computes the same mapping from inputs to optimal outputs as the SPD.
- The range of the composed transducer is a subset of the original derivation tree language.
- This subset can be computed by an MG without reference-set computation. In particular, this new MG can be obtained from the old MG by addition of a single local constraint.
- It follows that the SPD is equivalent to some local constraint.

Further Applications (cf. Graf 2010b)

- Focus Economy
- Merge-over-Move (without needing subarrays)
- a limited variant of Scope Economy (sufficient for all the data discussed in the literature)
- the Accord-Maximization-Principle

Uhm... What Just Happened?

- A sequence of mini-transducers computes the same mapping from inputs to optimal outputs as the SPD.
- The range of the composed transducer is a subset of the original derivation tree language.
- This subset can be computed by an MG without reference-set computation. In particular, this new MG can be obtained from the old MG by addition of a single local constraint.
- It follows that the SPD is equivalent to some local constraint.

Further Applications (cf. Graf 2010b)

- Focus Economy
- Merge-over-Move (without needing subarrays)
- a limited variant of Scope Economy (sufficient for all the data discussed in the literature)
- the Accord-Maximization-Principle

Arguments Against RCs in Syntax Revisited

Why no RCs in Syntax?

1. allegedly too computationally demanding (Johnson and Lappin 1999)
2. new, peculiar class of constraints ⇒ stipulative
3. conceptually inconsistent (Collins 1996)
4. leaves notion of simplicity unexplained (Jacobson 1997)

- (1)—(3) are immediately disproved by the fact that
  - linear transductions are efficiently computable,
  - every reference-set constraint can be translated into a local constraint,
  - the composed transducer never computes any of the suboptimal candidates, it directly rewrites every input as an optimal output.

Arguments Against RCs in Syntax Revisited

Why no RCs in Syntax?

1. allegedly too computationally demanding (Johnson and Lappin 1999)
2. new, peculiar class of constraints ⇒ stipulative
3. conceptually inconsistent (Collins 1996)
4. leaves notion of simplicity unexplained (Jacobson 1997)

- (1)—(3) are immediately disproved by the fact that
  - linear transductions are efficiently computable,
  - every reference-set constraint can be translated into a local constraint,
  - the composed transducer never computes any of the suboptimal candidates, it directly rewrites every input as an optimal output.

Notion of Simplicity

- Complexity of a transducer measured by its number of states.
- The fewer states we need to capture the economy metric, the simpler it is.

Example: Variants of the SDP

- Original SDP: $\oplus O$ used for ranking, uses only one state
- Accord-Maximization Principle: $\ominus O$ for ranking, only one state
- SDP that prefers second-shortest derivations: first compute $\oplus O$, then filter out optimal candidates and apply $\oplus O$; number of states might grow exponentially with complexity of set of candidates that are optimal wrt first run of $\oplus O$
- SDP that keeps only better half of all candidates: undefinable

Notion of Simplicity

- Complexity of a transducer measured by its number of states.
- The fewer states we need to capture the economy metric, the simpler it is.

Example: Variants of the SDP

- Original SDP: $\oplus O$ used for ranking, uses only one state
- Accord-Maximization Principle: $\ominus O$ for ranking, only one state
- SDP that prefers second-shortest derivations: first compute $\oplus O$, then filter out optimal candidates and apply $\oplus O$; number of states might grow exponentially with complexity of set of candidates that are optimal wrt first run of $\oplus O$
- SDP that keeps only better half of all candidates: undefinable

In Favor of Reference-Set Constraints in Syntax

Translating an MG with a reference-set constraint into a canonical MG, we might see a blowup in the size of the lexicon that is exponential in the number of states of the transducer that computes the reference-set constraint

$\Rightarrow$ reference-set constraints provide extremely succinct descriptions

Example: Blow-Up in the Lexicon

Assume MG G has a lexicon with 11 entries:

- 5 lexical items with no selection features
- 3 with exactly 1 selection feature
- 3 with exactly 2 selection features

Assume further that the transducer has 4 states.
In the worst case, the corresponding canonical MG has a lexicon of size 260 ($\approx$ 24-times the original size).

In Favor of Reference-Set Constraints in Syntax

Translating an MG with a reference-set constraint into a canonical MG, we might see a blowup in the size of the lexicon that is exponential in the number of states of the transducer that computes the reference-set constraint

$\Rightarrow$ reference-set constraints provide extremely succinct descriptions

Example: Blow-Up in the Lexicon

Assume MG G has a lexicon with 11 entries:

- 5 lexical items with no selection features
- 3 with exactly 1 selection feature
- 3 with exactly 2 selection features

Assume further that the transducer has 4 states.
In the worst case, the corresponding canonical MG has a lexicon of size 260 ($\approx$ 24-times the original size).

In Favor of Reference-Set Constraints in Syntax (Cont.)

The local correspondents of a reference-set constraint
vary depending on the input grammar
$\Rightarrow$ reference-set constraints allow us to express new generalizations

Example: Correspondents of SDP

- Original grammar: remove all lexical items with an s-feature
- Add lexical item *le* :: $-\mathrm{D} - \mathrm{s}$, but not *le* :: $-\mathrm{D}$: A verb carries $+$s only if it selects *le*.
- Does not carry over to grammars where $+$s may occur on lexical items that aren't verbs.
- Further seemingly unrelated effects of SDP: favor "chunking, interdependencies between movement-steps, etc.

In Favor of Reference-Set Constraints in Syntax (Cont.)

The local correspondents of a reference-set constraint
vary depending on the input grammar
$\Rightarrow$ reference-set constraints allow us to express new generalizations

Example: Correspondents of SDP

- Original grammar: remove all lexical items with an s-feature
- Add lexical item *le* :: $-\mathrm{D} - \mathrm{s}$, but not *le* :: $-\mathrm{D}$: A verb carries $+$s only if it selects *le*.
- Does not carry over to grammars where $+$s may occur on lexical items that aren't verbs.
- Further seemingly unrelated effects of SDP: favor "chunking, interdependencies between movement-steps, etc.

Arguments for RC-Processing Connection Revisited

Why RCs in the Parser?

1. RC = race
2. race-effects due to parallel computation
3. ample evidence for parallel computation
4. simplicity = fastest to compute

Some General Remarks Against Grammar in the Parser

- Transparent parser as null-hypothesis: abstracted away from resource-limitations, the parser successfully parses all grammatical sentences, and only those
- Parsing is about acceptability, ambiguity resolution and processing speed, but not grammaticality
- Processing literature abstracts away from specific assumptions about grammar, little interest in grammaticality

Arguments for RC-Processing Connection Revisited

Why RCs in the Parser?

1. RC = race
2. race-effects due to parallel computation
3. ample evidence for parallel computation
4. simplicity = fastest to compute

Some General Remarks Against Grammar in the Parser

- Transparent parser as null-hypothesis: abstracted away from resource-limitations, the parser successfully parses all grammatical sentences, and only those
- Parsing is about acceptability, ambiguity resolution and processing speed, but not grammaticality
- Processing literature abstracts away from specific assumptions about grammar, little interest in grammaticality

RC ≠ race

The notion of race is more complex than anticipated:

- The Accord-Maximization Principle maximizes agreement-relations, i.e. favors the longest derivation
  ⇒ inverse of the SDP
  ⇒ in how far race-like?
- Merge-over-Move has no effect on the length or complexity of a derivation
  ⇒ why should we see a difference in processing speed?
- Quite generally, since reference-set constraints are reducible to local constraints, are local constraints race-like, too?

This also cast doubt on the idea that simplicity equals speed of computation (which is vacuous anyhow in the absence of a specific model of computation)

RC ≠ race

The notion of race is more complex than anticipated:

- The Accord-Maximization Principle maximizes agreement-relations, i.e. favors the longest derivation
  ⇒ inverse of the SDP
  ⇒ in how far race-like?
- Merge-over-Move has no effect on the length or complexity of a derivation
  ⇒ why should we see a difference in processing speed?
- Quite generally, since reference-set constraints are reducible to local constraints, are local constraints race-like, too?

This also cast doubt on the idea that simplicity equals speed of computation (which is vacuous anyhow in the absence of a specific model of computation)

Status of Parallel Computation in Processing Literature

- The distinction between serial and parallel is only meaningful wrt specific parameter settings (data structure operated on by the parser, deterministic or probabilistic, +/− exhaustive, +/− competitive, +/− reanalysis, +/− predictive, mode of memory access); every piece of data I know can be explained by both serial and parallel models, depending on the choice of parameters.
- If one makes standard assumptions (data structure=tree representations, non-exhaustive, limited non-determinism for parallel, determinism + reanalysis for serial, no predictions), the evidence seems to favor parallel models, but crucially non-competitive parallel models (Pearlmutter and Mendelsohn 1999; Gibson and Pearlmutter 2000; Clifton and Staub 2008)

What About Connectionist Approaches?

- There is interesting work relating OT/HG to connectionist networks and parallel computation at the subsymbolic level (Hendriks et al. 2007; Smolensky et al. 2010)
- Reference-set constraints can be viewed as a special kind of OT-grammar (Graf 2010a,b)
- Doesn't that derive them from parallel computation?

Objections

- Even the most powerful connectionist networks compute only functions that are computable by the serial Turing machine.
  $\Rightarrow$ parallel grammar does not imply parallel computation
- Also, we can take a completely serial grammar and implement it as a neural network
  $\Rightarrow$ parallel computation does not imply parallel grammar
- So there is no causality here that would derive anything.

What About Connectionist Approaches?

- There is interesting work relating OT/HG to connectionist networks and parallel computation at the subsymbolic level (Hendriks et al. 2007; Smolensky et al. 2010)
- Reference-set constraints can be viewed as a special kind of OT-grammar (Graf 2010a,b)
- Doesn't that derive them from parallel computation?

Objections

- Even the most powerful connectionist networks compute only functions that are computable by the serial Turing machine.
  $\Rightarrow$ parallel grammar does not imply parallel computation
- Also, we can take a completely serial grammar and implement it as a neural network
  $\Rightarrow$ parallel computation does not imply parallel grammar
- So there is no causality here that would derive anything.

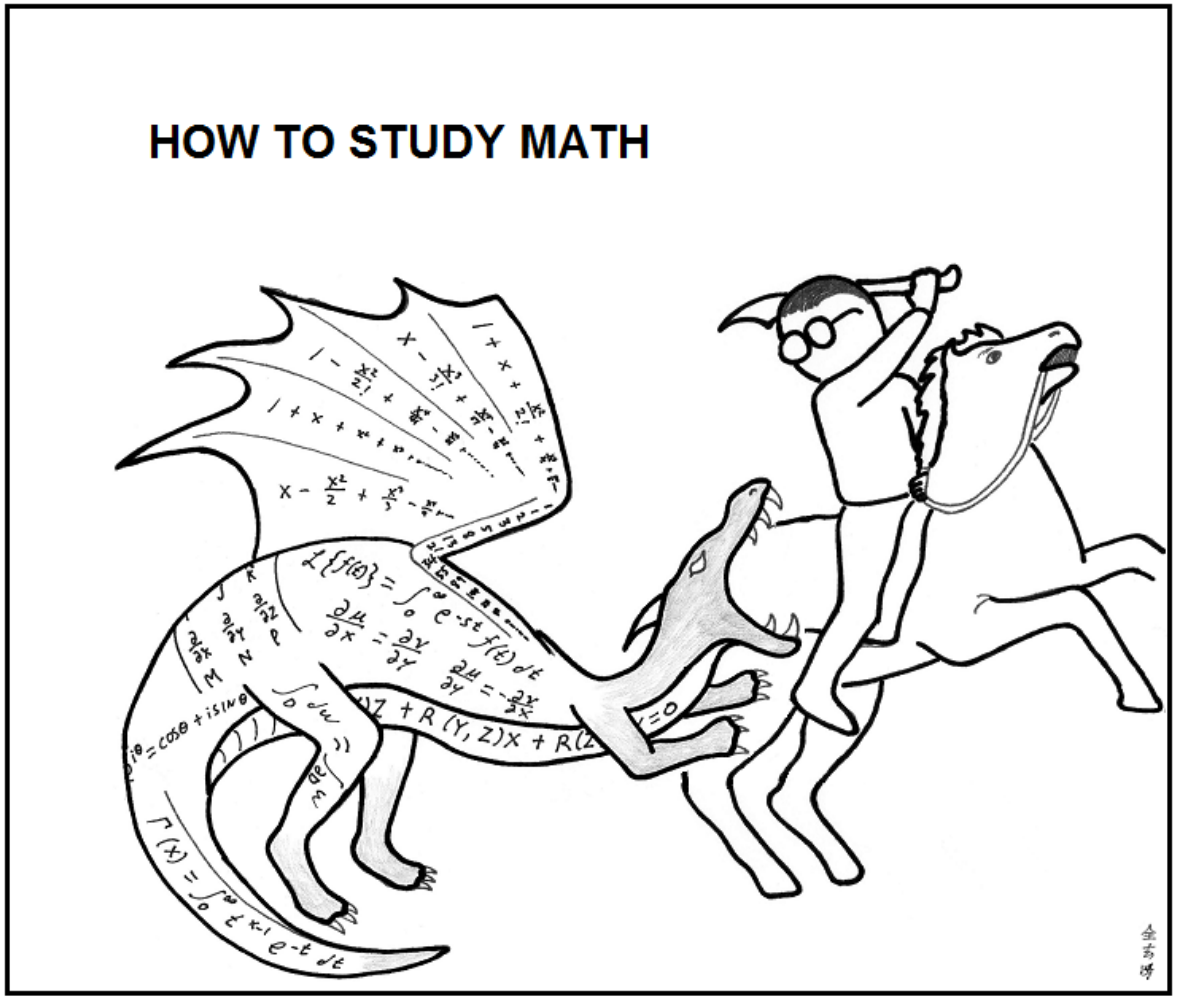
Some Loose Threads

- **Acquisition**
  If reference-set constraints are just local constraints, then what about the language acquisition phenomena that have been conjectured to be due to the complexity of reference-set computation?
- **Empirical Application**
  Since reference-set constraints open up new generalizations that hold across grammars, they should be the ideal tool for cross-linguistic comparisons. But so far they have not been used in this area.
- **Semantics**
  Many reference-set constraints require identity of meaning. But this condition isn't computable even for weak fragments of first-order logic. If context is taken into account (Rule I), the problem becomes unsolvable even for propositional logic.

Conclusion

- Reference-set constraints = different way of defining local constraints.
  - We showed this using mathematical properties of Minimalist grammars and linear tree transductions.
  - The general approach was to implement reference-set constraints as transducers using a strategy of underspecification and filtration.
- It follows that there is no reason to locate them in some syntax-external module.
- In fact, there are good reasons to keep them in syntax (succinctness, generality).
- It might be possible to derive them from parallel computation, but it will require special assumptions that are difficult to defend on independent grounds.

Thank you!



Don't just read it; fight it!

--- Paul R. Halmos

References I

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.

Clifton, Charles, and Adrian Staub. 2008. Parallelism and competition in syntactic ambiguity resolution. *Language and Linguistics Compass* 2:234–250.

Collins, Chris. 1994. Economy of derivation and the generalized proper binding condition. *Linguistic Inquiry* 25:45–61.

Collins, Chris. 1996. *Local economy*. Cambridge, Mass.: MIT Press.

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

Gibson, Edward, and Neal J. Pearlmutter. 2000. Distinguishing serial and parallel processing. *Journal of Psycholinguistic Research* 29:231–240.

References II

Graf, Thomas. 2010a. Reference-set constraints as linear tree transductions via controlled optimality systems. In *Proceedings of the 15th Conference on Formal Grammar*. To appear.

Graf, Thomas. 2010b. A tree transducer model of reference-set computation. *UCLA Working Papers in Linguistics* 15:1–53.

Graf, Thomas. 2011. Closure properties of minimalist derivation tree languages. To appear in Proceedings of LACL2011.

Hendriks, Petra, Hedderik van Rijn, and Bea Valkenier. 2007. Learning to reason about speakers' alternatives in sentence comprehension: A computational account. *Lingua* 117:1879–1896.

Jacobson, Pauline. 1997. Where (if anywhere) is transderivationality located? In *The limits of syntax*, ed. Brian D. Joseph, Carl Pollard, Peter Culicover, and Louise McNally, 303–336. Burlington, MA: Academic Press.

Johnson, David, and Shalom Lappin. 1999. *Local constraints vs. economy*. Stanford: CSLI.

References III

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.

Kobele, Gregory M. 2011. Minimalist tree languages are closed under intersection with recognizable tree languages. To appear in Proceedings of LACL2011.

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.

Pearlmutter, Neal J., and Aurora Alma Mendelsohn. 1999. Serial versus parallel sentence comprehension. Ms., Northeastern University.

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

Smolensky, Paul, Matthew Goldrick, and Donald Mathis. 2010. Optimization and quantization in gradient symbol systems: A framework for integrating the continuous and the discrete in cognition. Ms., John Hopkins and Northwestern.

References IV

Stabler, Edward P. 1997. Derivational minimalism. In *Logical aspects of computational linguistics: First international conference, LACL '96, Nancy, France, September 23-25, 1996. Selected papers*, ed. Christian Retoré, 68–95. Berlin: Springer.

Appendix

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 ⇒ Scope Economy structurally more demanding than SDP

Computing the Blow-Up

Given a lexicon Lex and $n \geq 0$, let $Lex^{(n)} := \{l \in Lex \mid l \text{ has exactly } n \text{ selector features}\}$. Now if there is no $m > k$ such that $Lex_G^{(m)} \neq \emptyset$, then in the worst case

$$|Lex_{G'}| = \sum_{i=0}^{k} \left(|Lex_G^{(i)}| \cdot |Q|^{i+1} \right)$$

Focus Economy

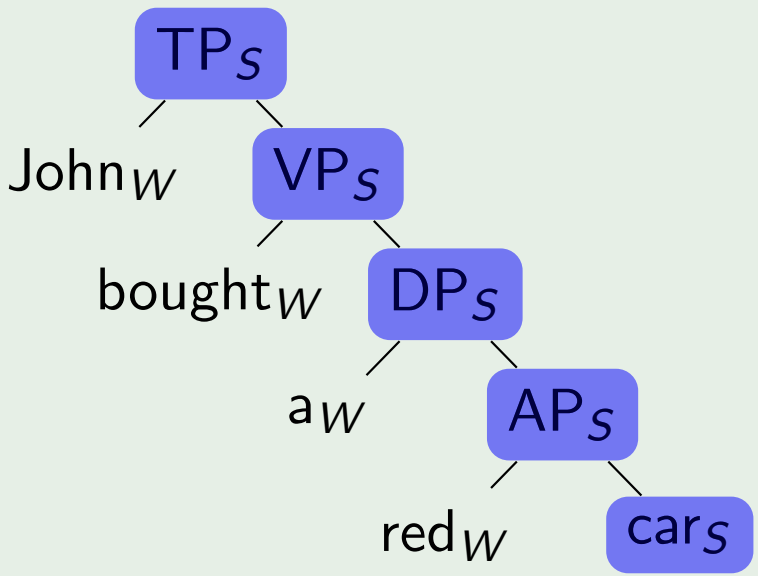
Example 1: Focus Economy

Focus Economy Rule (Reminder)

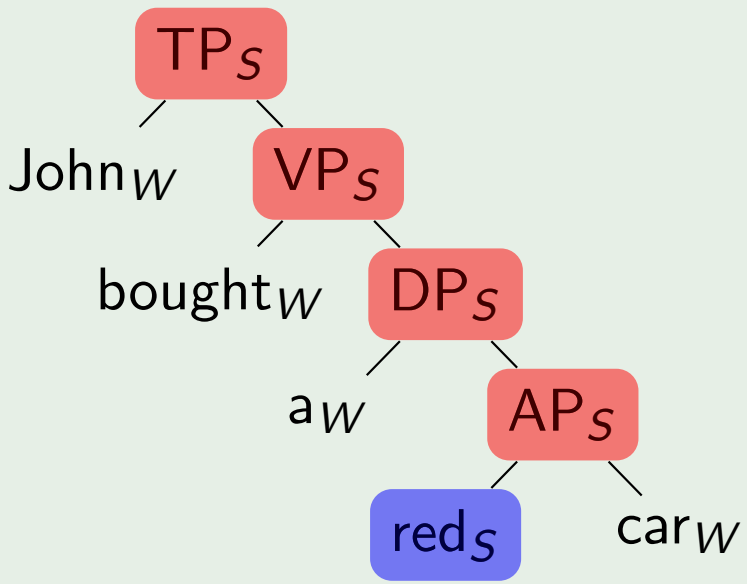
If the main stress has been shifted, a constituent containing its carrier may be focused iff it cannot be focused in the tree with unshifted stress.

Computing the Focus Sets

a) Neutral Stress



b) Shifted Stress

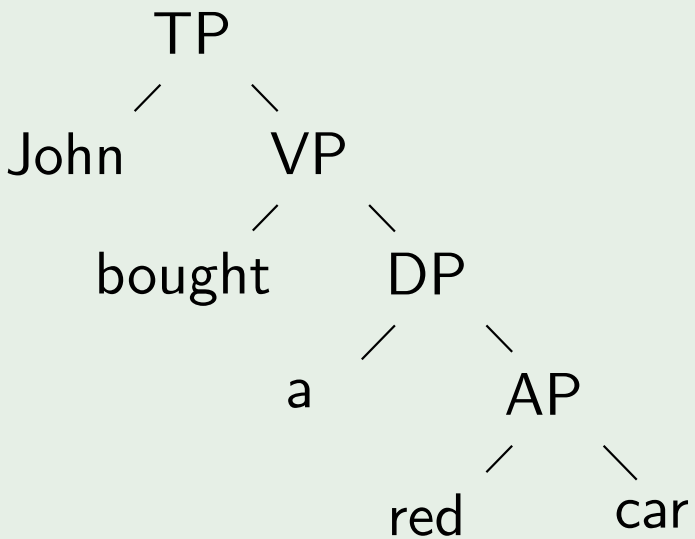


Transducer Model: GEN

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

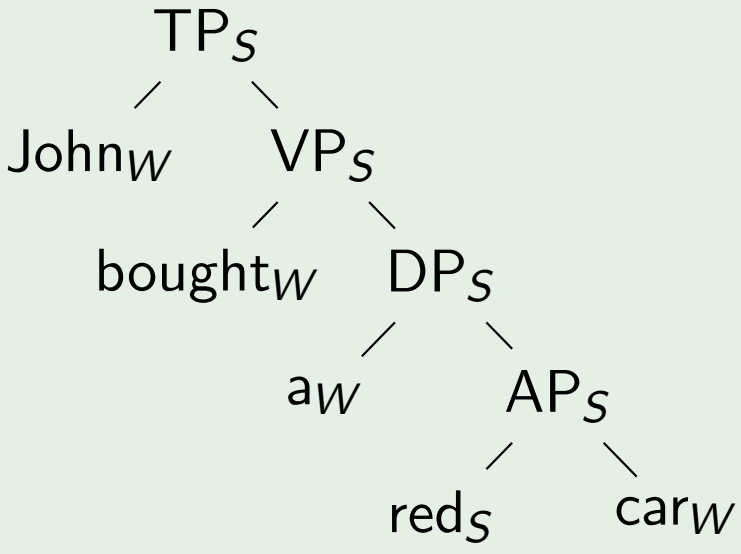


Transducer Model: GEN

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

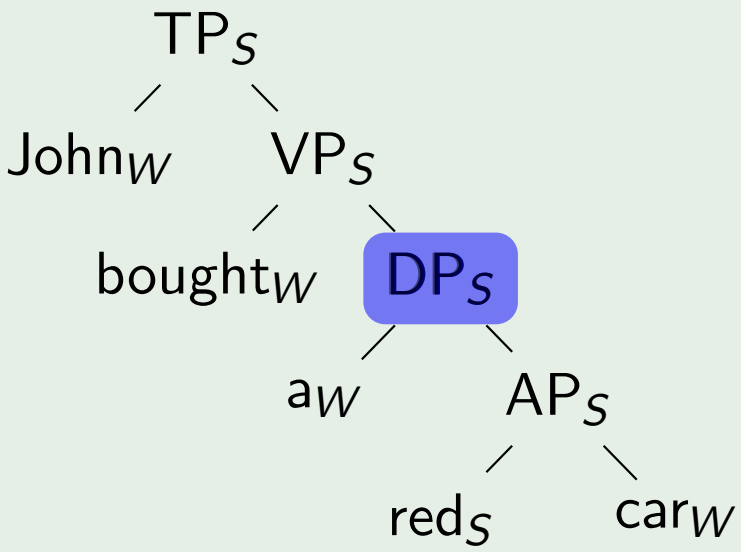


Transducer Model: Gen

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

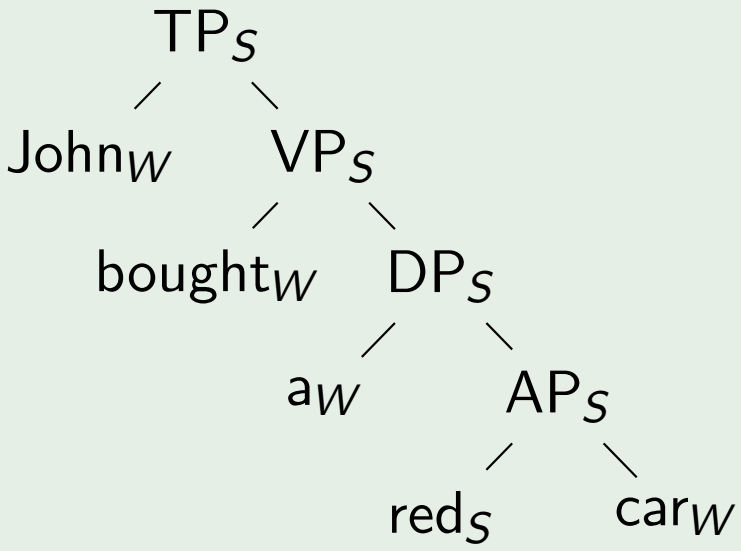
Focus Economy requires reference to the neutral stress pattern. We allow this by implicitly representing the neutral stress within the same tree!

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.

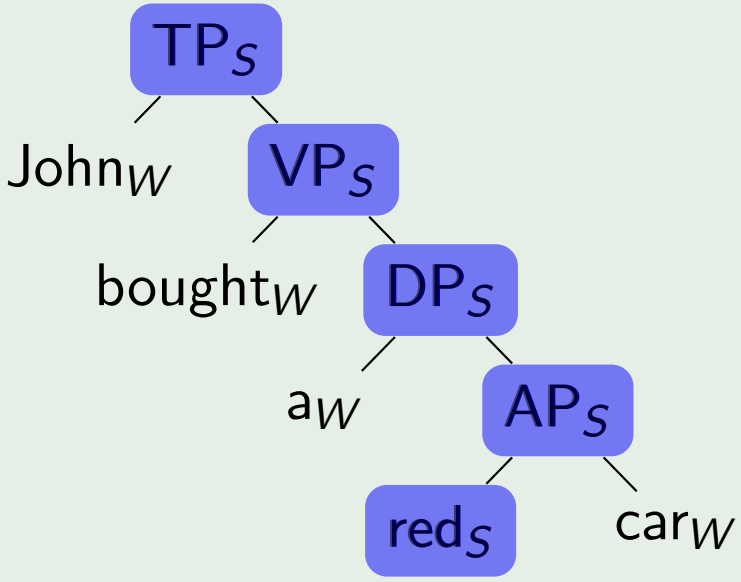
Example of ϕ

StressPath and NeutralPath



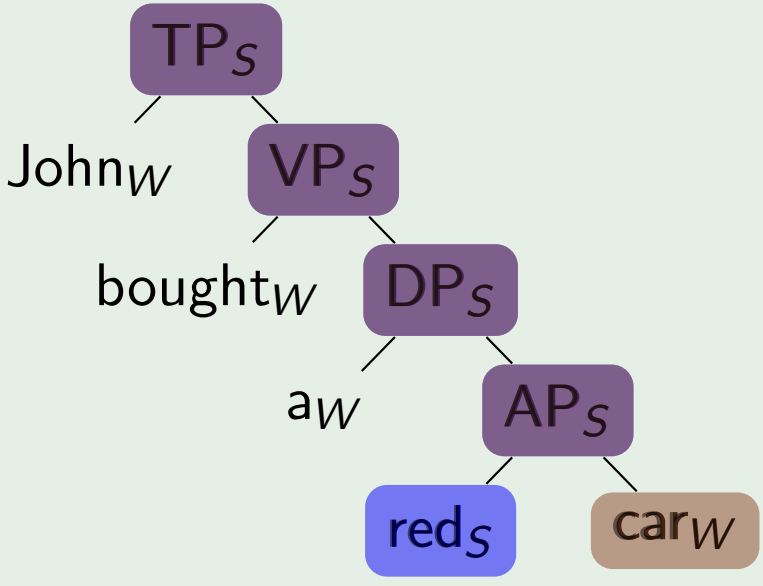
Example of ϕ

StressPath and NeutralPath



Example of ϕ

StressPath and NeutralPath



Merge-over-Move

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'.

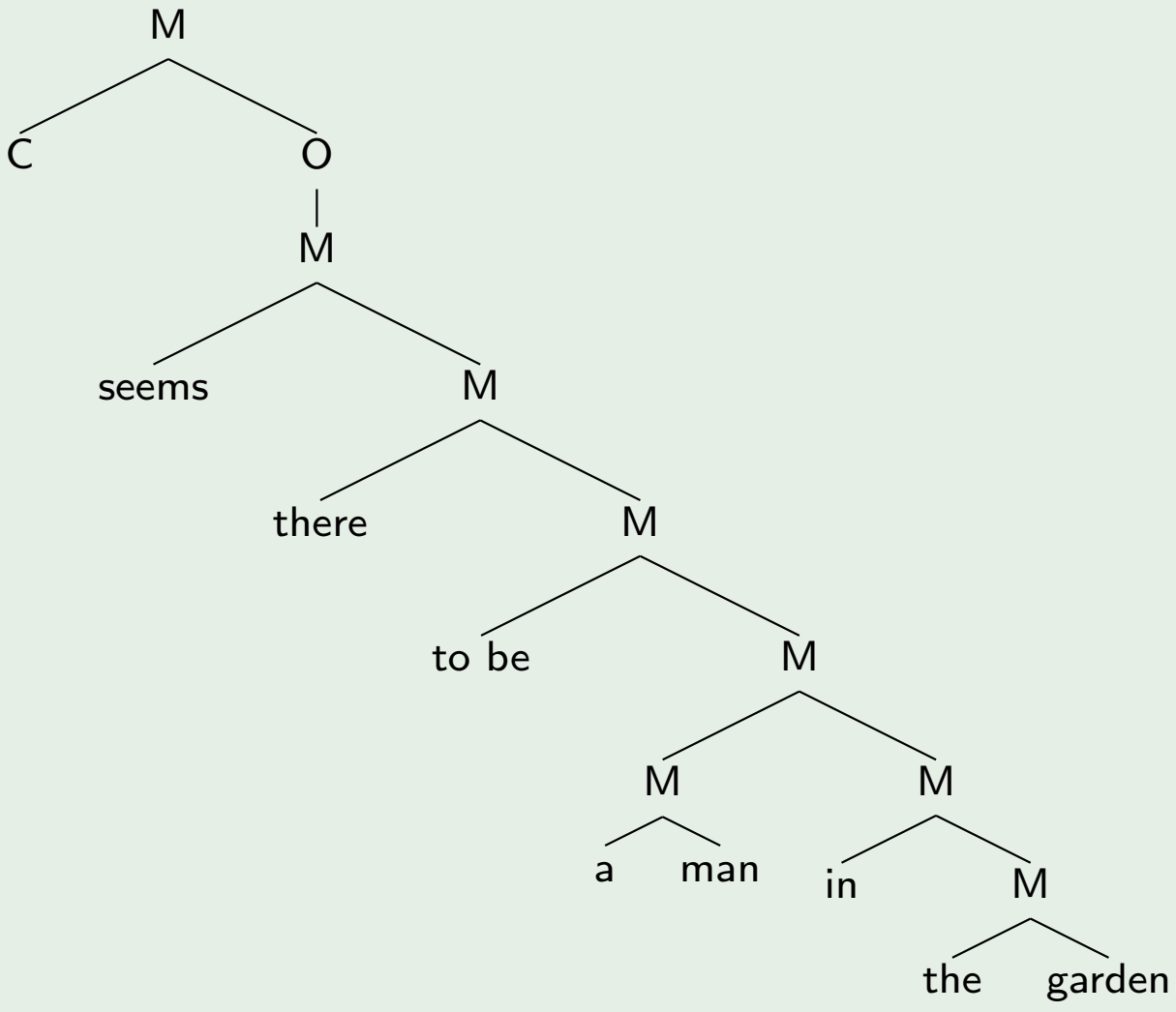
(9) a. There seems t_{there} to be a man in the garden.

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.

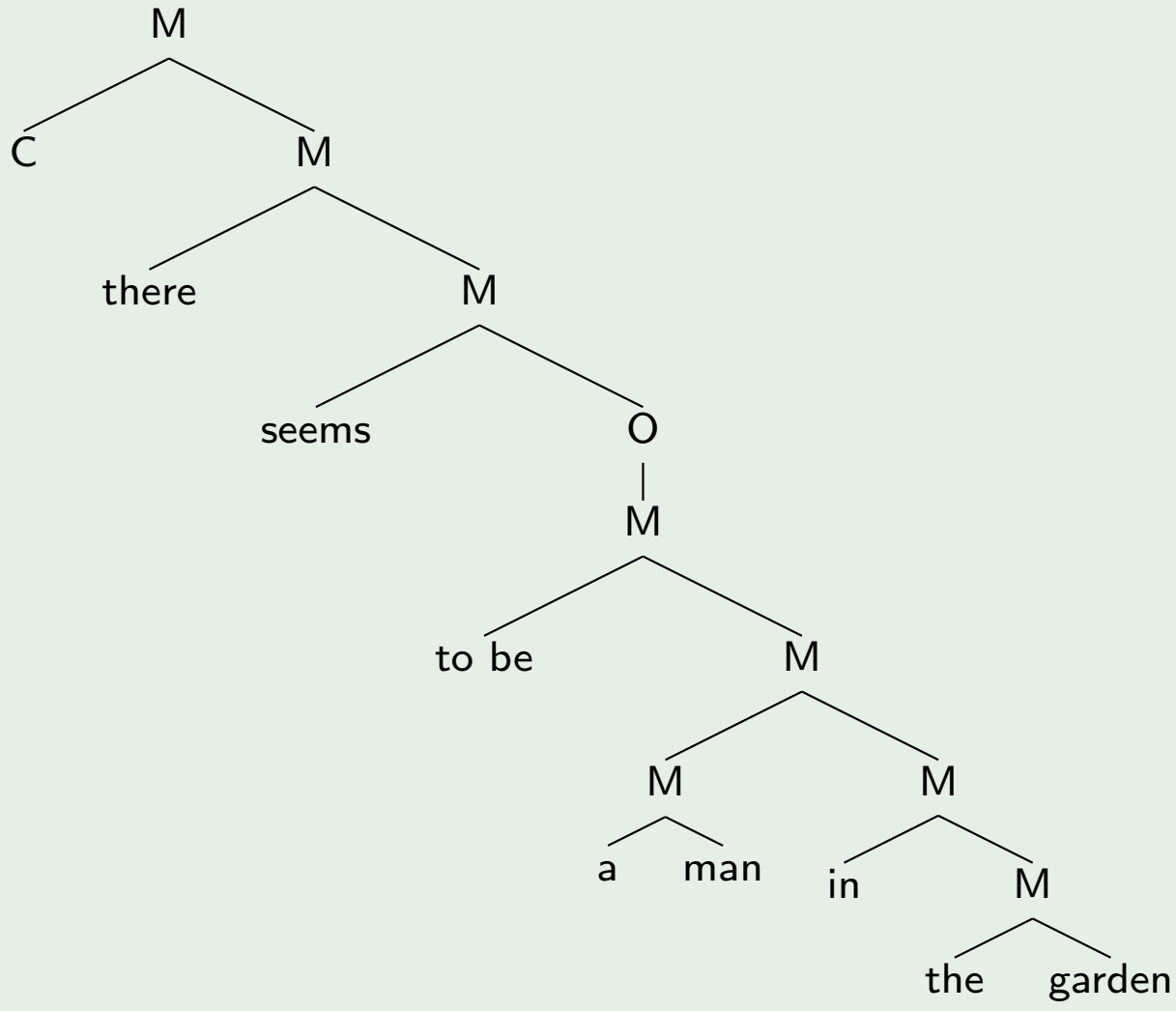
Derivation Trees of (9a) and (9b)

Example



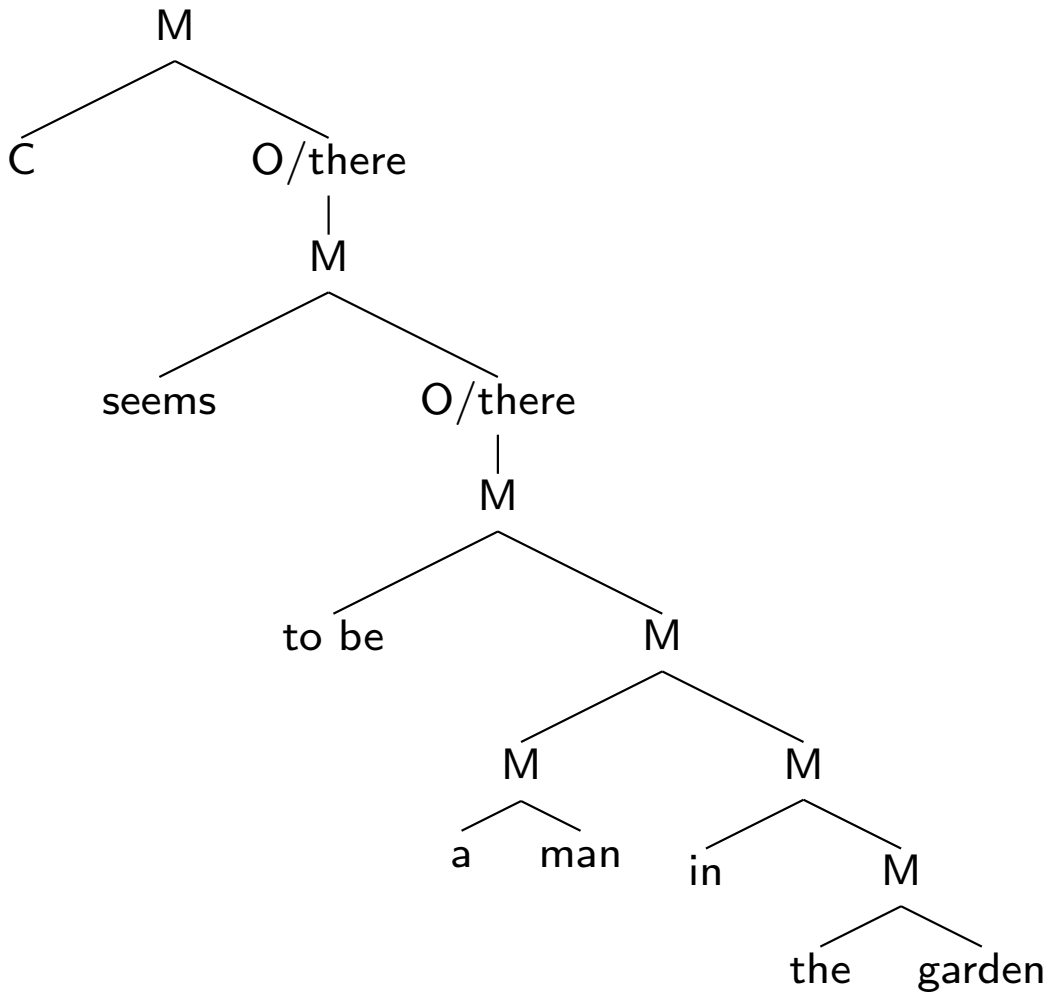
Derivation Trees of (9a) and (9b)

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.

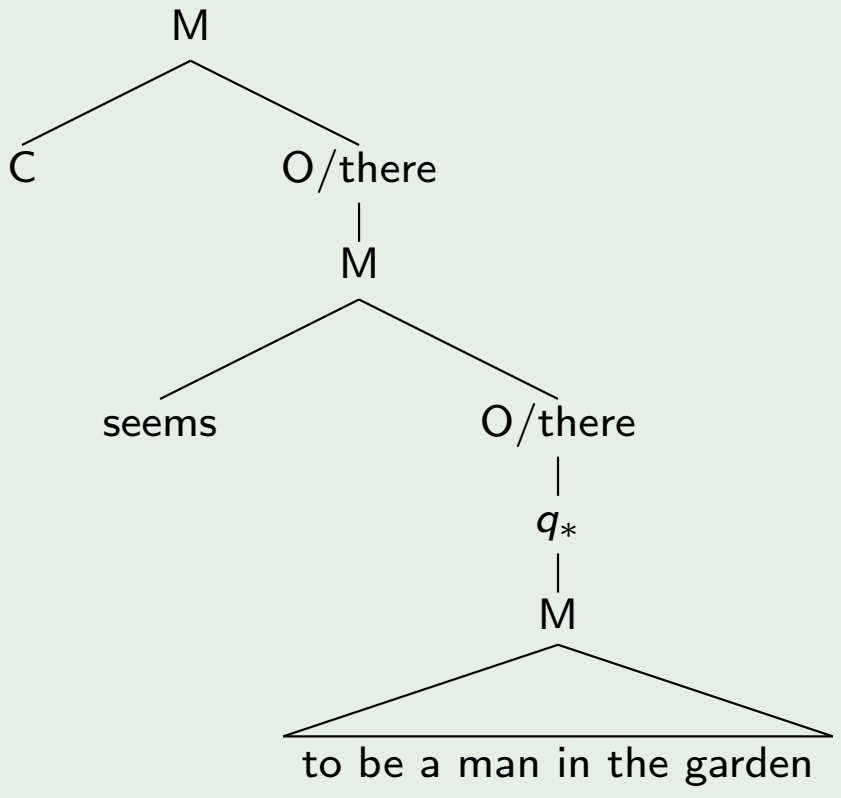


Transducer Model: GEN (Step 2)

- 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 occurrence 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.

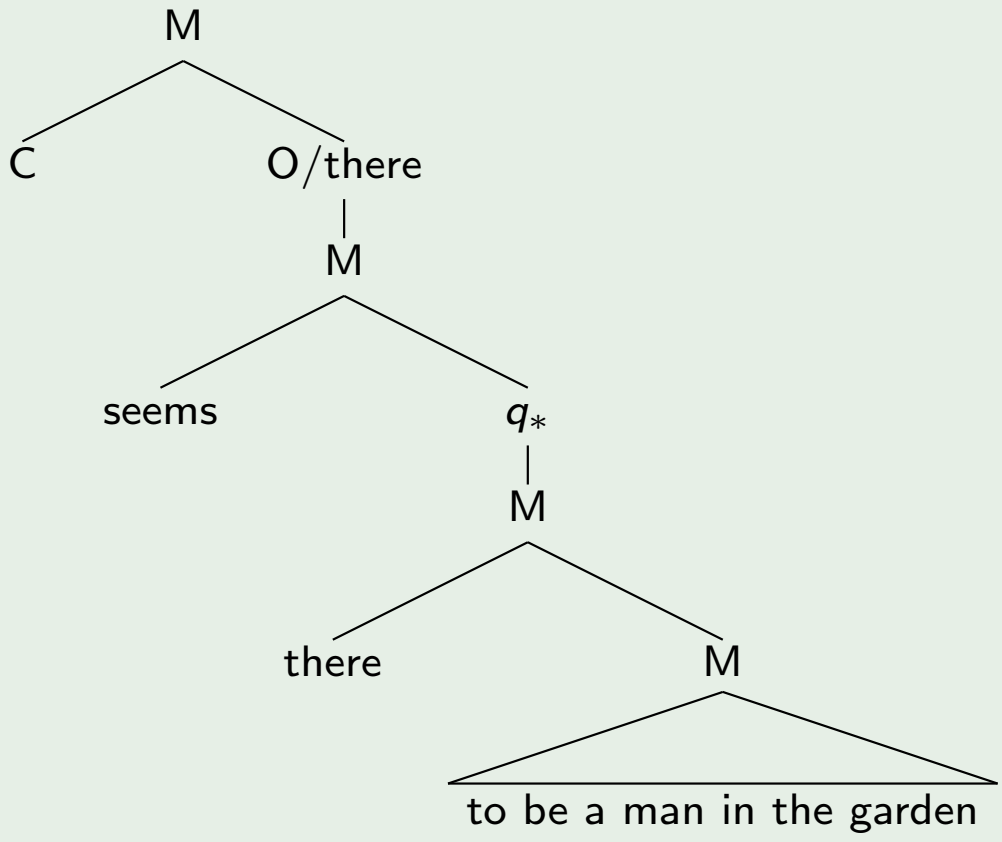
Transducer Model: Examples of Step 2

Example 1



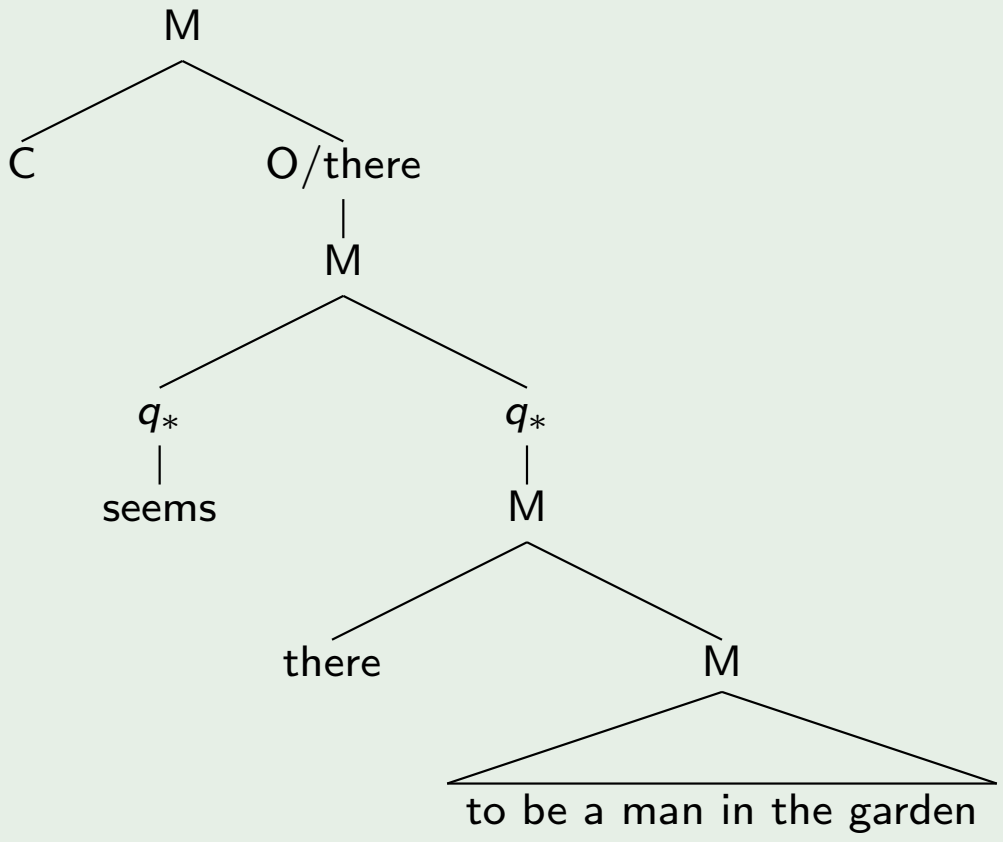
Transducer Model: Examples of Step 2

Example 1



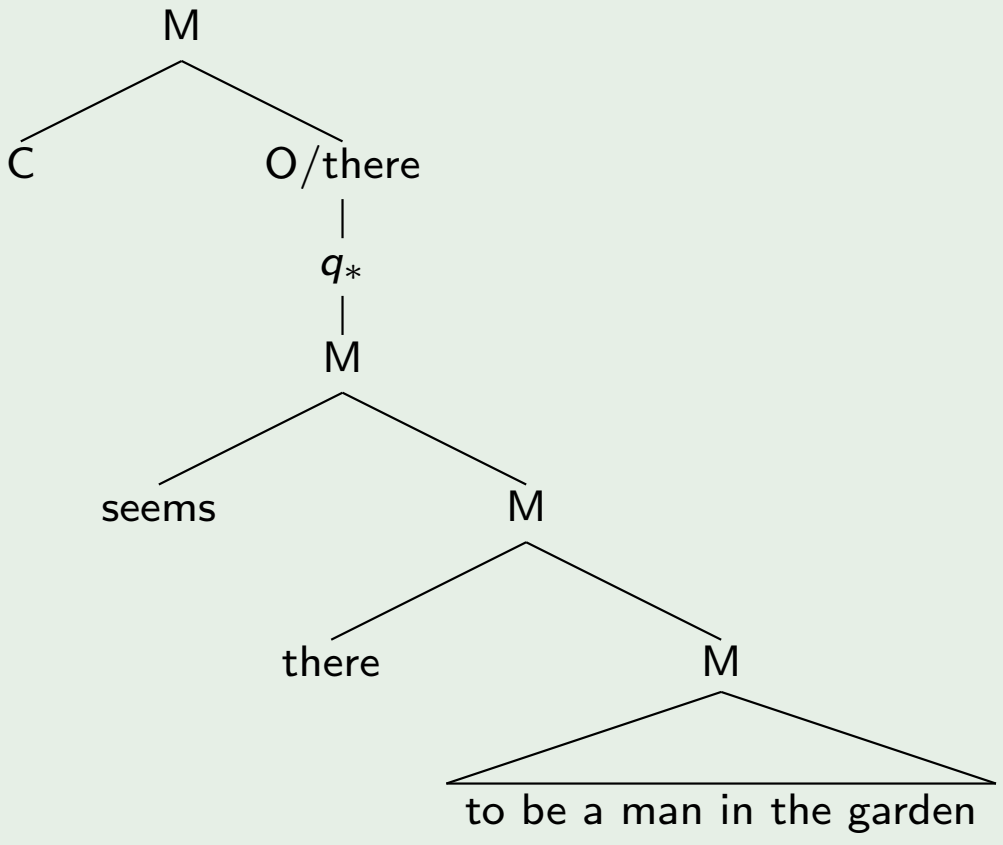
Transducer Model: Examples of Step 2

Example 1



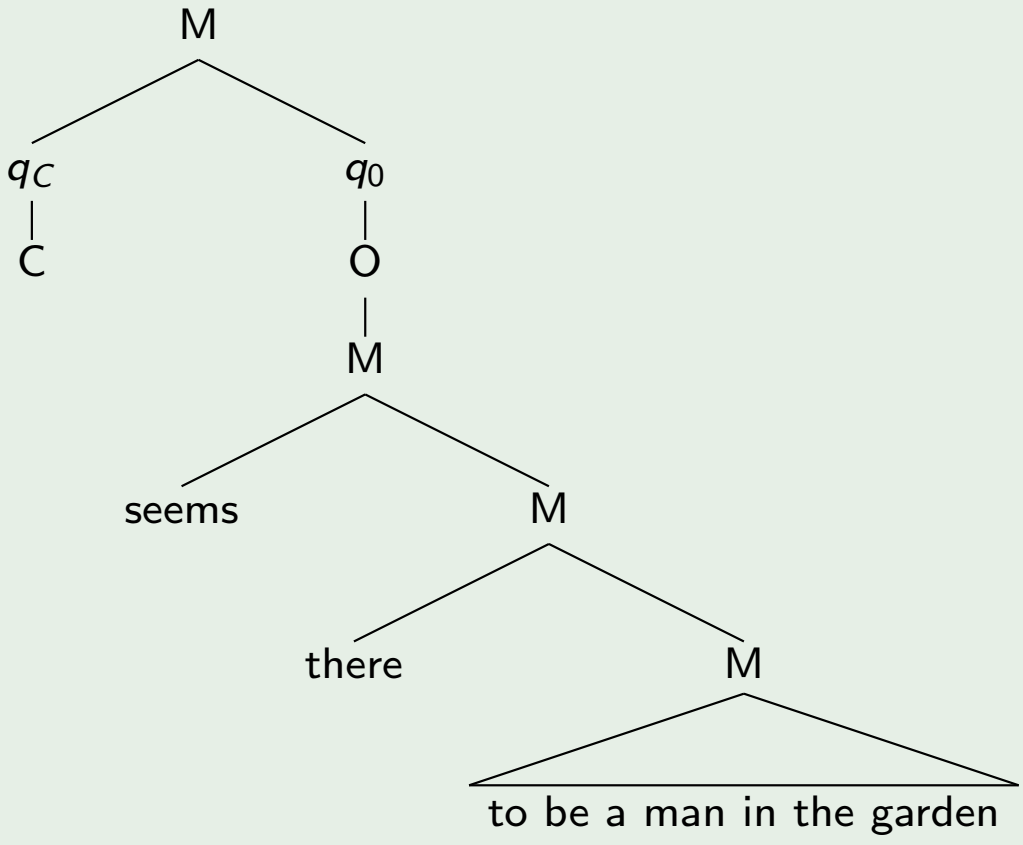
Transducer Model: Examples of Step 2

Example 1



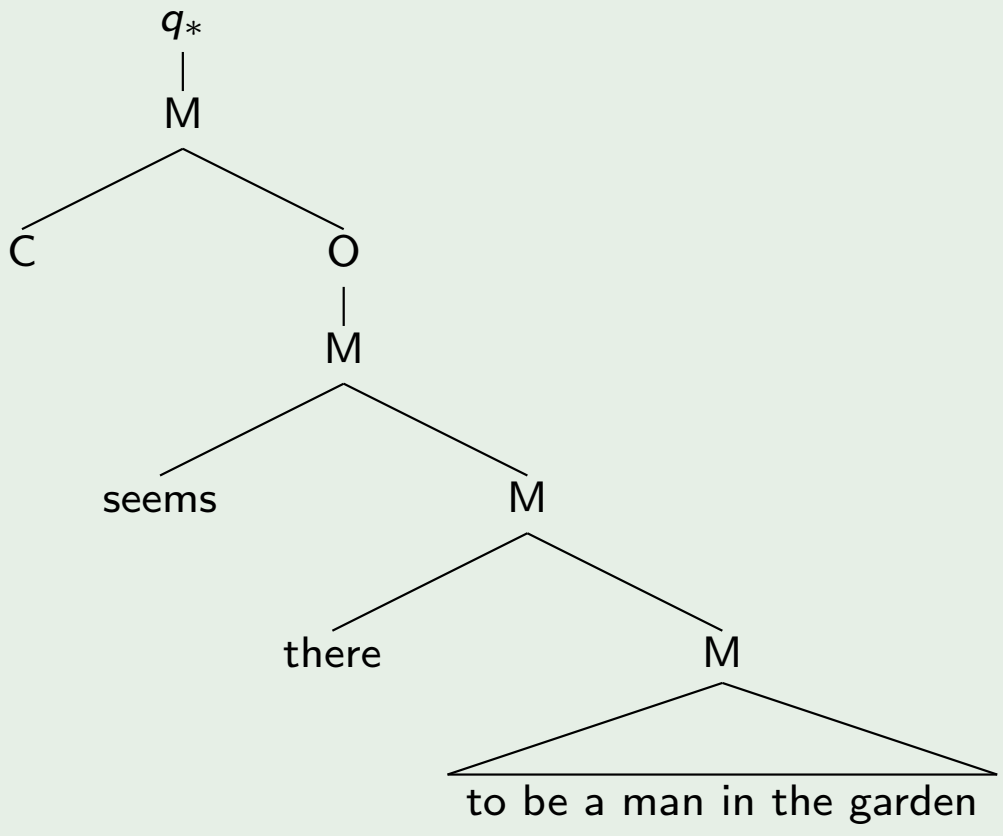
Transducer Model: Examples of Step 2

Example 1



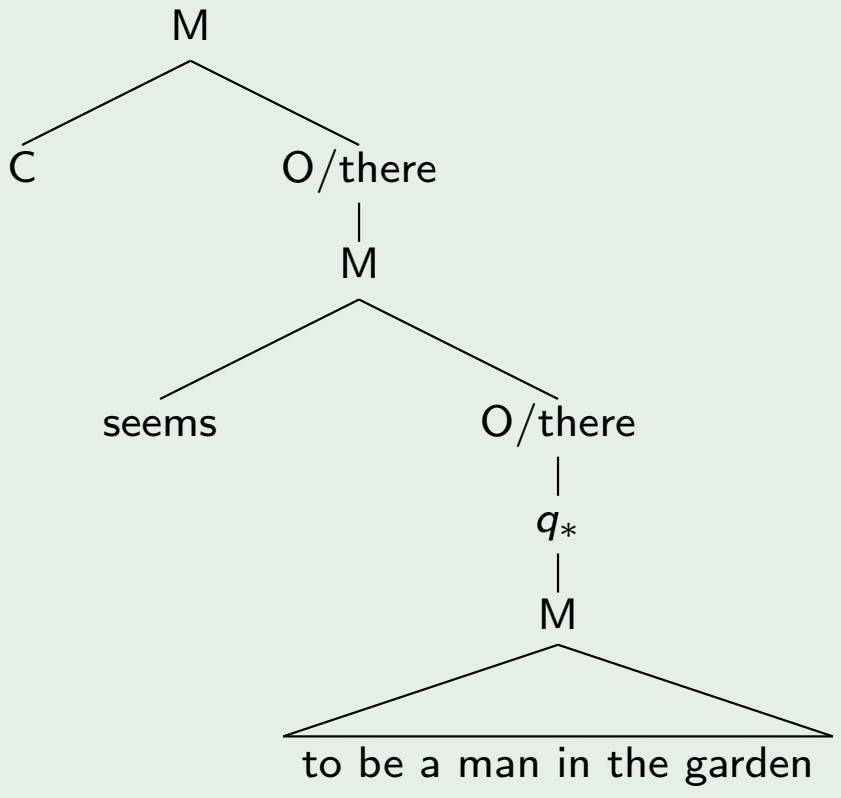
Transducer Model: Examples of Step 2

Example 1



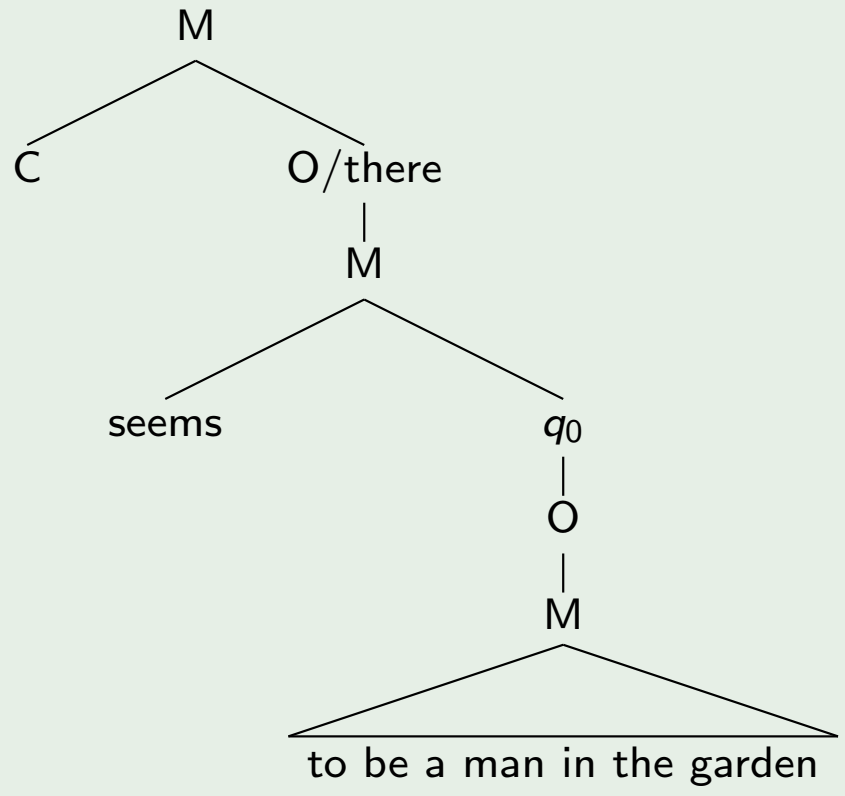
Transducer Model: Examples of Step 2

Example 2



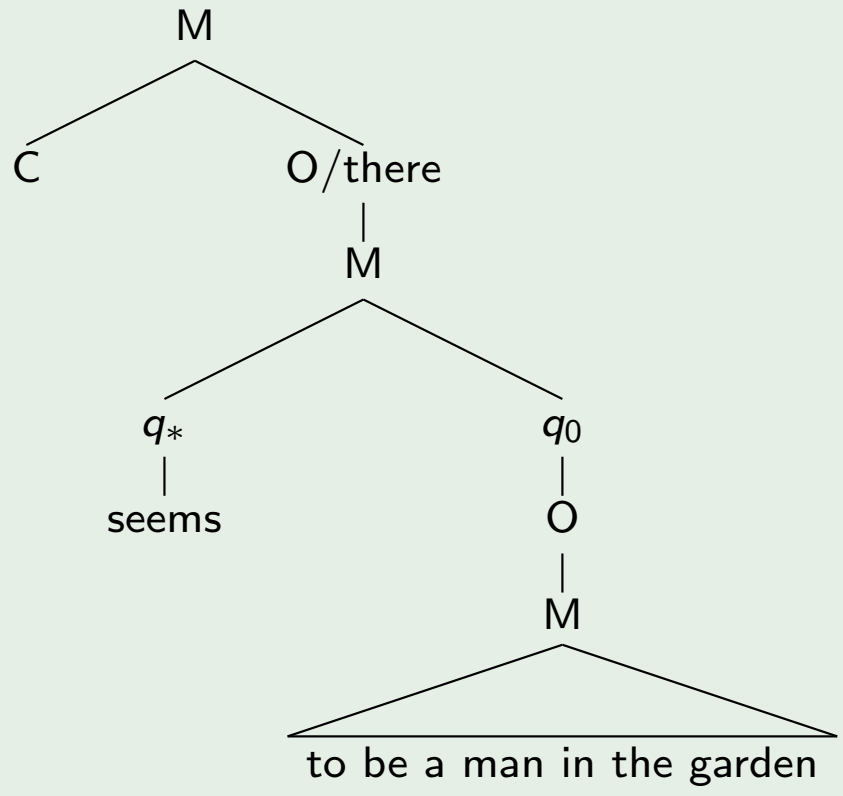
Transducer Model: Examples of Step 2

Example 2



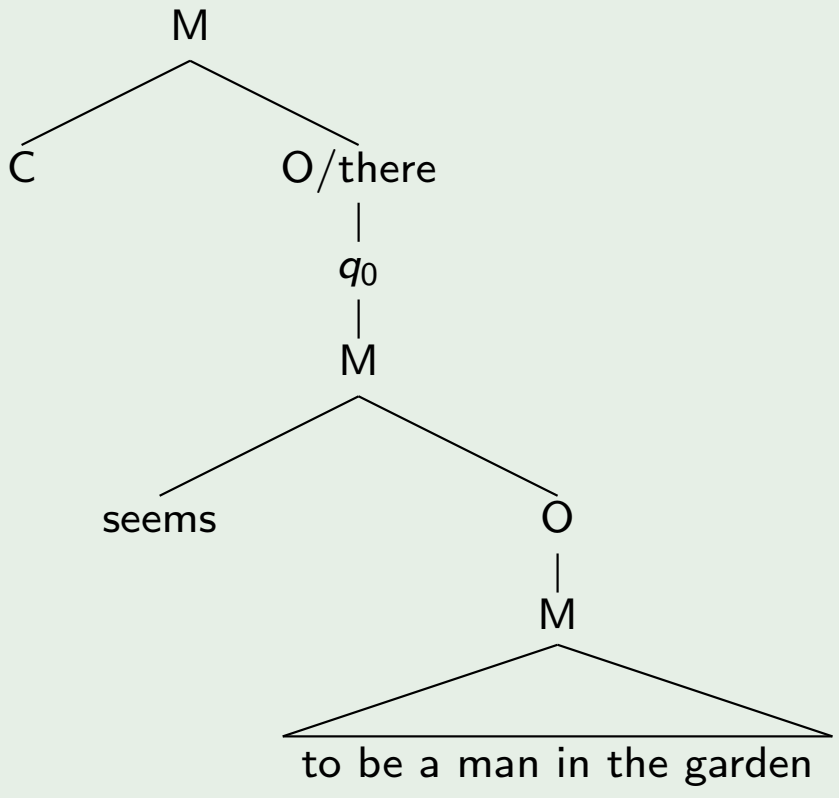
Transducer Model: Examples of Step 2

Example 2



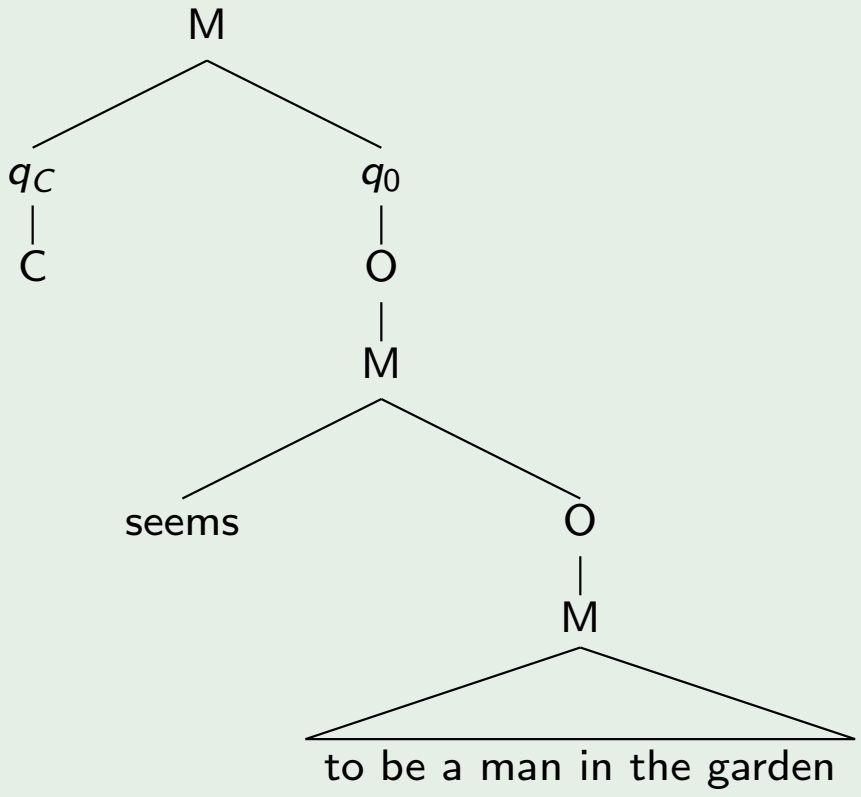
Transducer Model: Examples of Step 2

Example 2



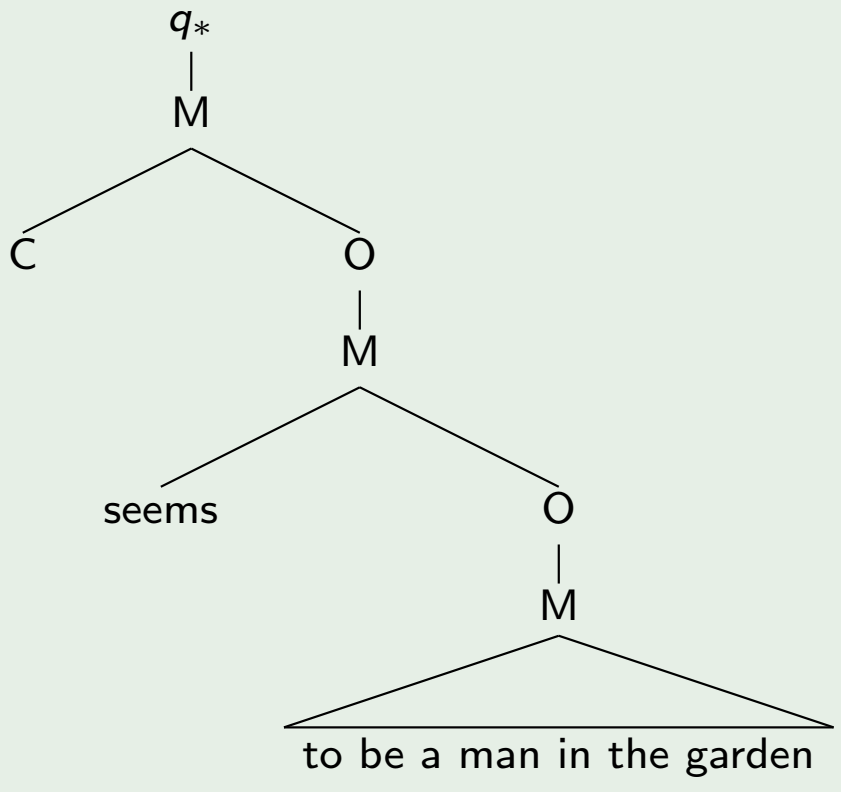
Transducer Model: Examples of Step 2

Example 2



Transducer Model: Examples of Step 2

Example 2



Transducer Model: The Induced Mapping

The output candidates for both (10a) and (10b) are now (11a)–(11b).

(10) 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.

(11) 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 (11c) is also assigned this reference set.
- (11a) 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.