# Lost in Translation: A Formal Model of Merge-Over-Move and its Implications for the Language Faculty

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

University of California, Los Angeles

ConSOLE XIX, Groningen, Netherlands
January 7, 2010

# Two Stories

## Story 1: Syntax and Interface Conditions

In Minimalism, it is assumed that syntax is restricted by interface conditions. But do those conditions uniquely determine it?

Result: No, once in a while syntax can trick the interfaces and thus flout some of their demands.

## Story 2: The Dual Nature of Reference-Set Constraints

Reference-set constraints are argued to be too computationally demanding for the parser, whence they must not be part of syntax.

Result: Many reference-set constraints can be replaced by standard well-formedness conditions that are efficiently computable.

# Two Stories

## Story 1: Syntax and Interface Conditions

In Minimalism, it is assumed that syntax is restricted by interface conditions. But do those conditions uniquely determine it?

Result: No, once in a while syntax can trick the interfaces and thus flout some of their demands.

## Story 2: The Dual Nature of Reference-Set Constraints

Reference-set constraints are argued to be too computationally demanding for the parser, whence they must not be part of syntax.

Result: Many reference-set constraints can be replaced by standard well-formedness conditions that are efficiently computable.

# Two Stories

## Story 1: Syntax and Interface Conditions

In Minimalism, it is assumed that syntax is restricted by interface conditions. But do those conditions uniquely determine it?

Result: No, once in a while syntax can trick the interfaces and thus flout some of their demands.

## Story 2: The Dual Nature of Reference-Set Constraints

Reference-set constraints are argued to be too computationally demanding for the parser, whence they must not be part of syntax.

Result: Many reference-set constraints can be replaced by standard well-formedness conditions that are efficiently computable.

# Two Stories

## Story 1: Syntax and Interface Conditions

In Minimalism, it is assumed that syntax is restricted by interface conditions. But do those conditions uniquely determine it?

Result: No, once in a while syntax can trick the interfaces and thus flout some of their demands.

## Story 2: The Dual Nature of Reference-Set Constraints

Reference-set constraints are argued to be too computationally demanding for the parser, whence they must not be part of syntax.

Result: Many reference-set constraints can be replaced by standard well-formedness conditions that are efficiently computable.

# Syntax & the Interfaces

**Minimalist Dictum**

Everything in syntax beyond Merge has to obey and/or follow from interface requirements.

- PF: interface to phonology/articulatory systems
  - linearization requirements
  - locality/islands?
- LF: interface to semantics/conceptual-interpretative systems
  - full interpretation
  - $\theta$-criterion?

But computability is also an issue:

- Phases
- Shortest Move/Closeness condition

Hence the parser, too, restricts syntax.

# Syntax & the Interfaces

**Minimalist Dictum**

Everything in syntax beyond Merge has to obey and/or follow from interface requirements.

- PF: interface to phonology/articulatory systems
  - linearization requirements
  - locality/islands?
- LF: interface to semantics/conceptual-interpretative systems
  - full interpretation
  - $\theta$-criterion?

But computability is also an issue:

- Phases
- Shortest Move/Closeness condition

Hence the parser, too, restricts syntax.

# A (Wrong) Conjecture

**Strong Minimalist Hypothesis (Chomsky 2000)**

Narrow Syntax is determined by interface conditions and nothing else.

**Strongest Minimalist Hypothesis**

Narrow Syntax is uniquely determined by interface conditions and nothing else.

I show that the Strongest Minimalist Hypothesis is wrong: Syntax is underdetermined by the interfaces. Syntax may violate an interface condition if it can "hide the violation".

# Linking Story 1 & 2

## Logic of the Argument

1. Reference-set constraints are argued to be too computationally demanding for the parser, so according to the Strongest Minimalist Hypothesis they must not be part of Narrow Syntax.
2. But many reference-set constraints are equivalent to constraints that involve no reference-set computation.
3. Narrow Syntax may use reference-set constraints, while the parser is fed the corresponding constraints without reference-set computation. That way, Narrow Syntax evades the computability requirement imposed by the parser, contra the Strongest Minimalist Hypothesis.

# 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 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)

# 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 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)

# 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 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. [$_{\text{TP}}$ 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

(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.

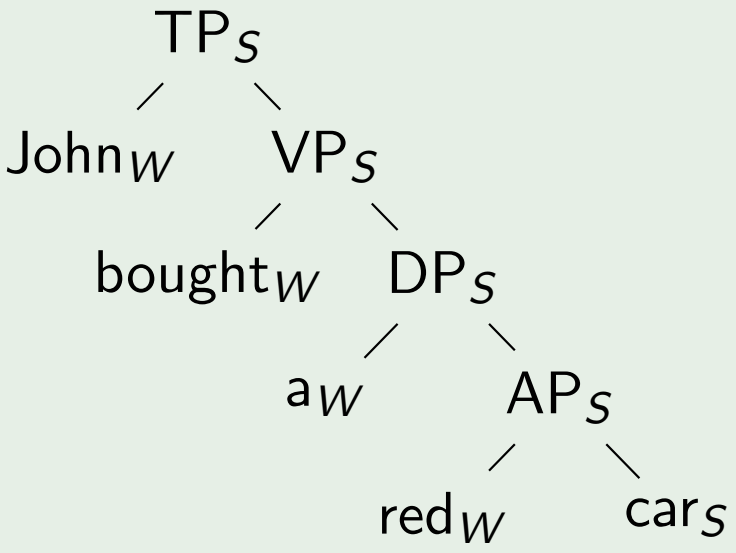
## Focus Economy Rule

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.

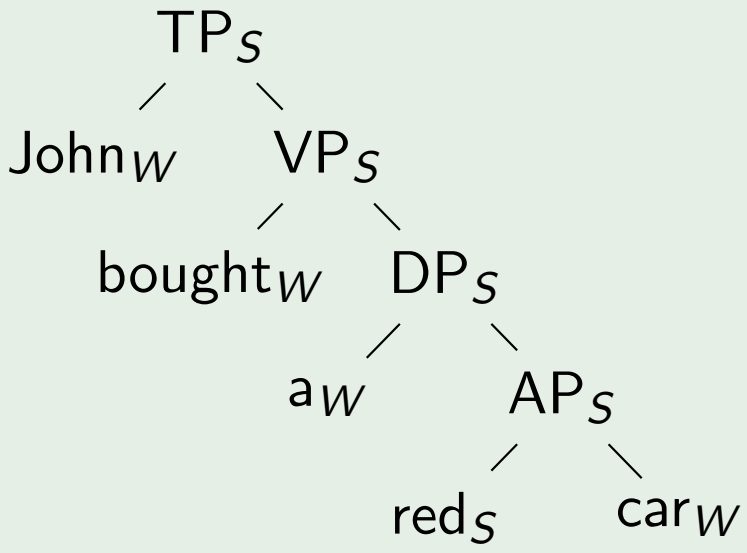
# Example: Focus Economy, Cont.

## Computing the Focus Sets

**a) Neutral Stress**



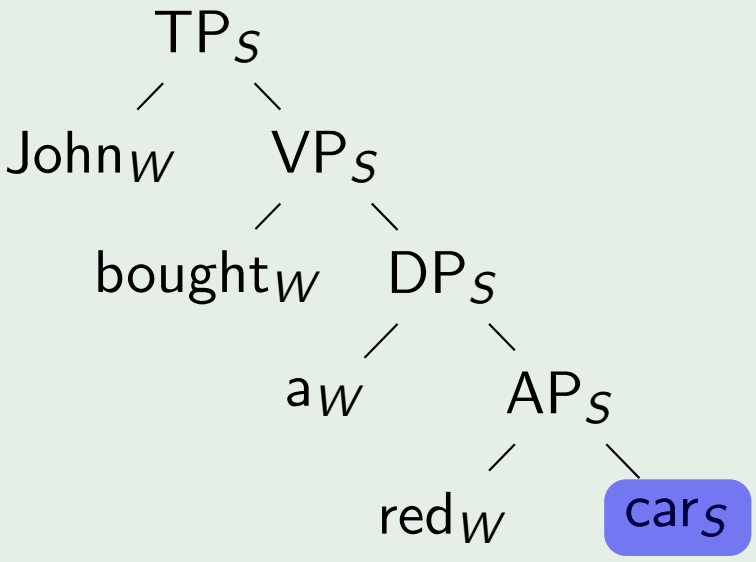

**b) Shifted Stress**



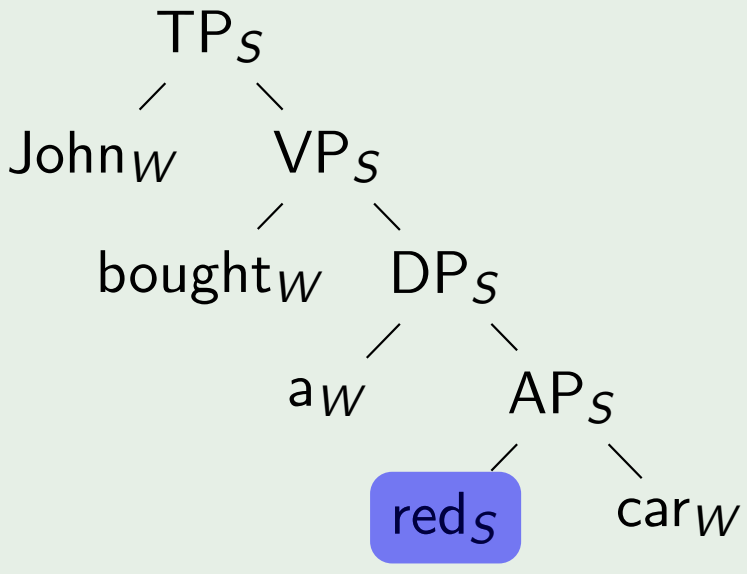
# Example: Focus Economy, Cont.

## Computing the Focus Sets

**a) Neutral Stress**



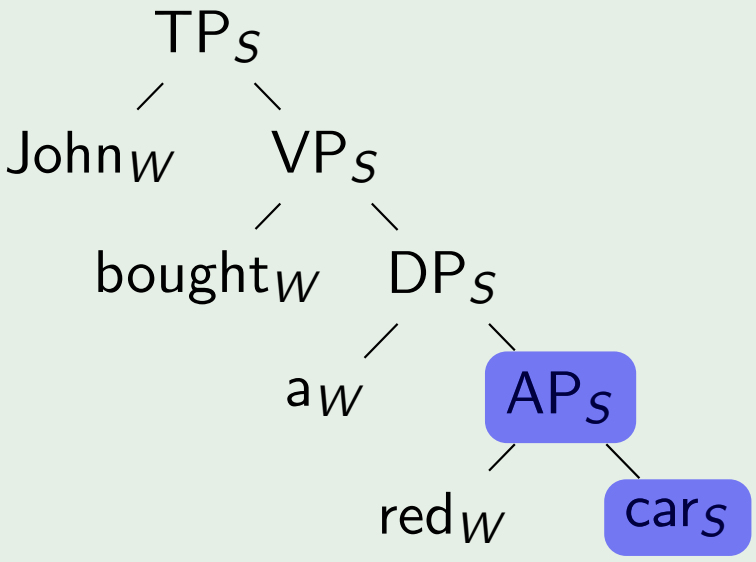

**b) Shifted Stress**



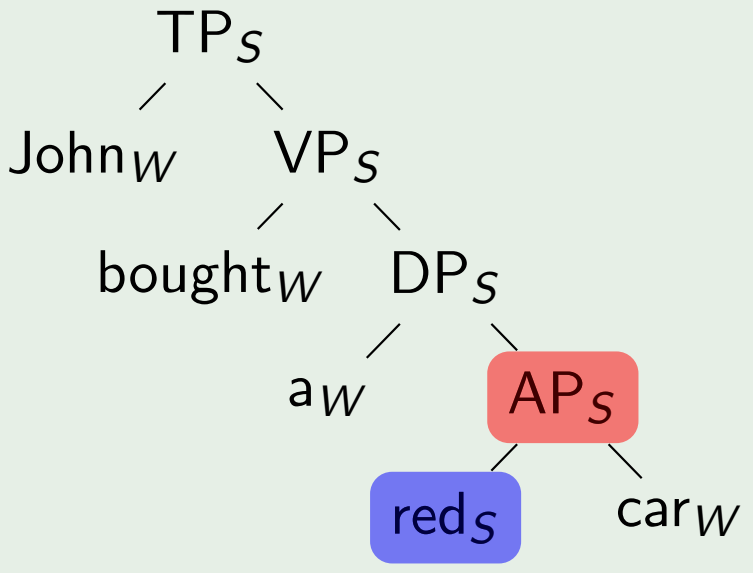
# Example: Focus Economy, Cont.

## Computing the Focus Sets

**a) Neutral Stress**



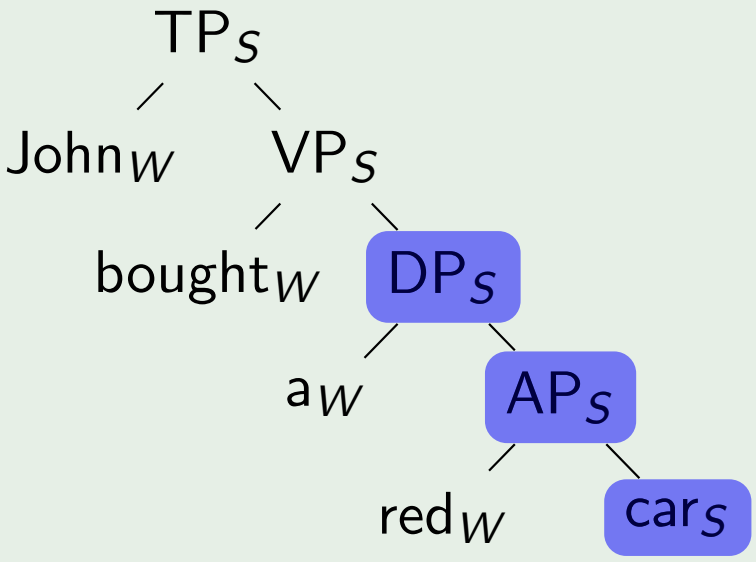

**b) Shifted Stress**



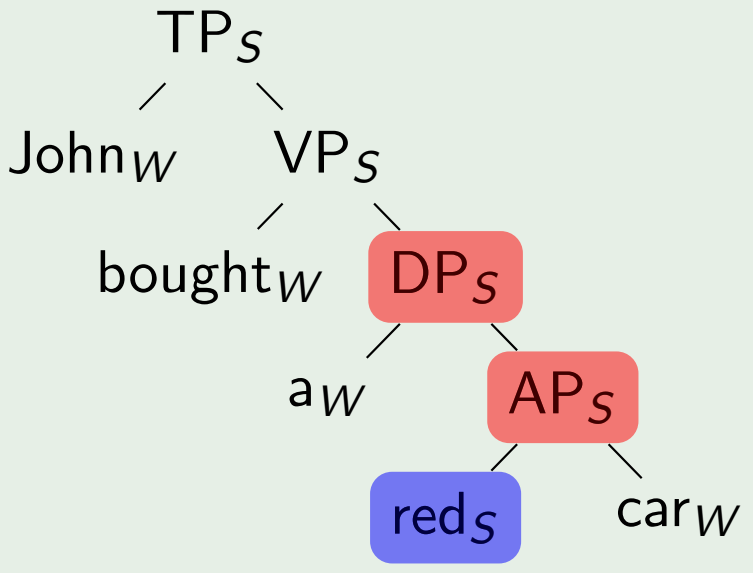
# Example: Focus Economy, Cont.

## Computing the Focus Sets

**a) Neutral Stress**



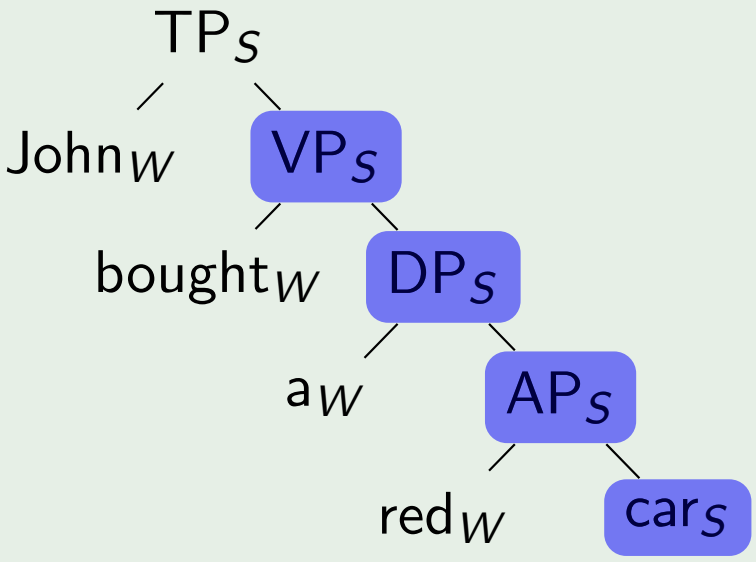

**b) Shifted Stress**



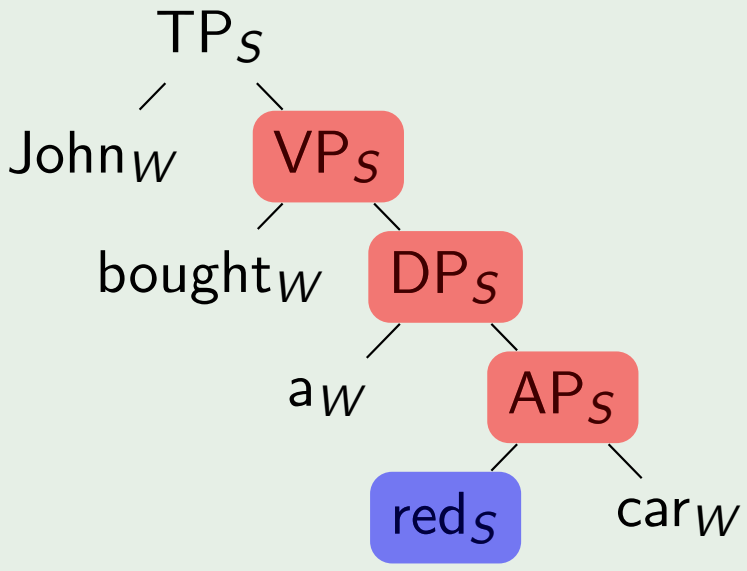
# Example: Focus Economy, Cont.

## Computing the Focus Sets

**a) Neutral Stress**



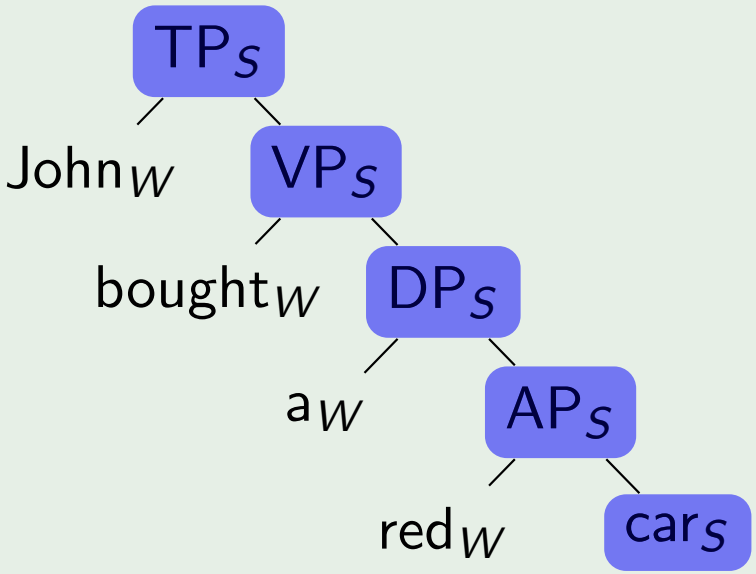

**b) Shifted Stress**



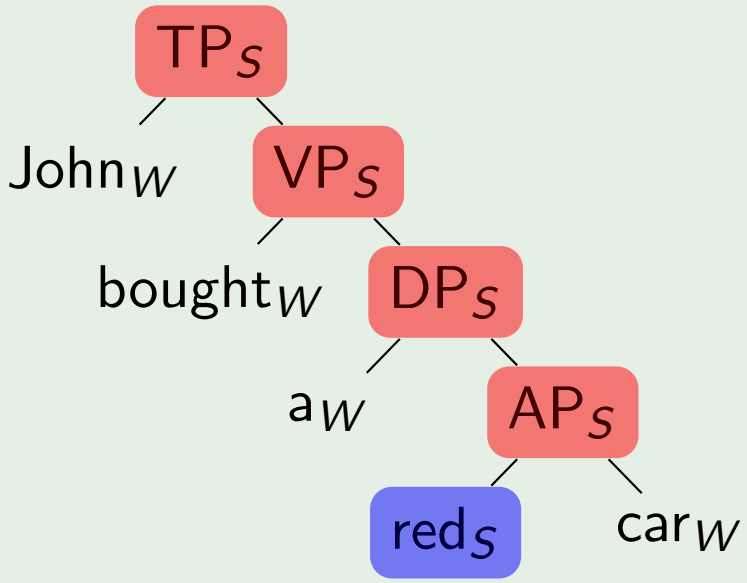
# Example: Focus Economy, Cont.

## Computing the Focus Sets

**a) Neutral Stress**




**b) Shifted Stress**



# The Computability Issue

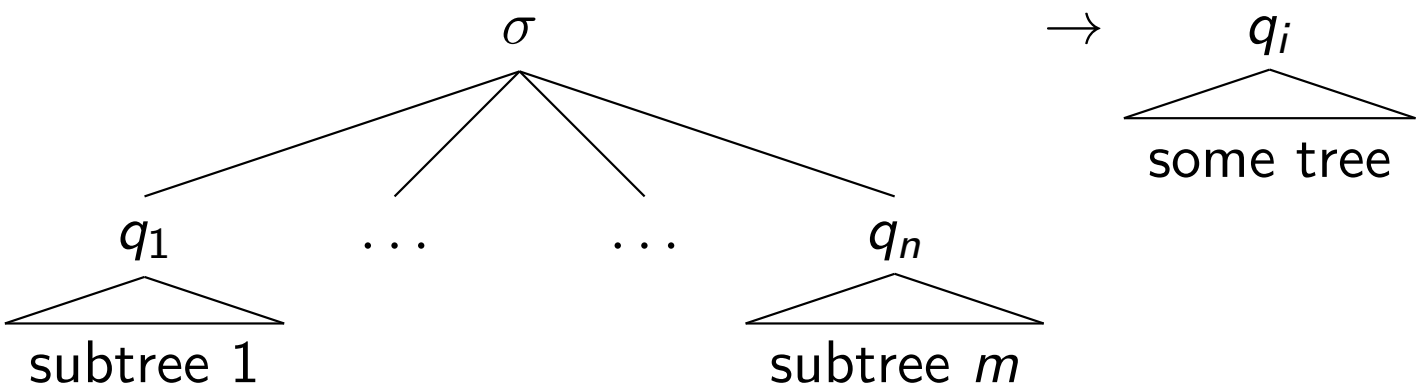
- As they involve comparisons of multiple trees, all of which have to be computed first, reference-set constraints are believed to be too computationally demanding.
  (Collins 1996; Johnson and Lappin 1999).
- But if we use linear tree transducers as a model, it turns out that this concern is unwarranted.
  Rather, many reference-set constraints have fully equivalent local constraints that operate within a single tree and do not involve any comparisons between trees.

# 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:



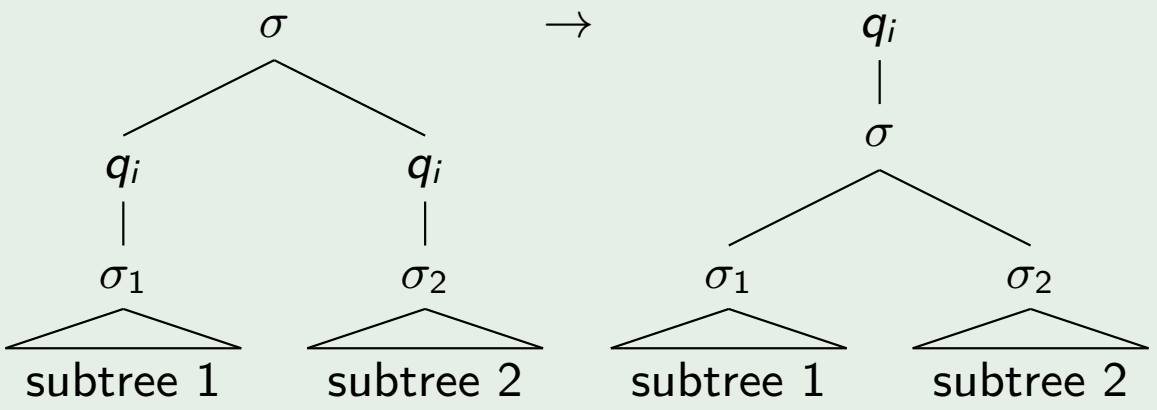
# A Simple Example (Part 1)

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

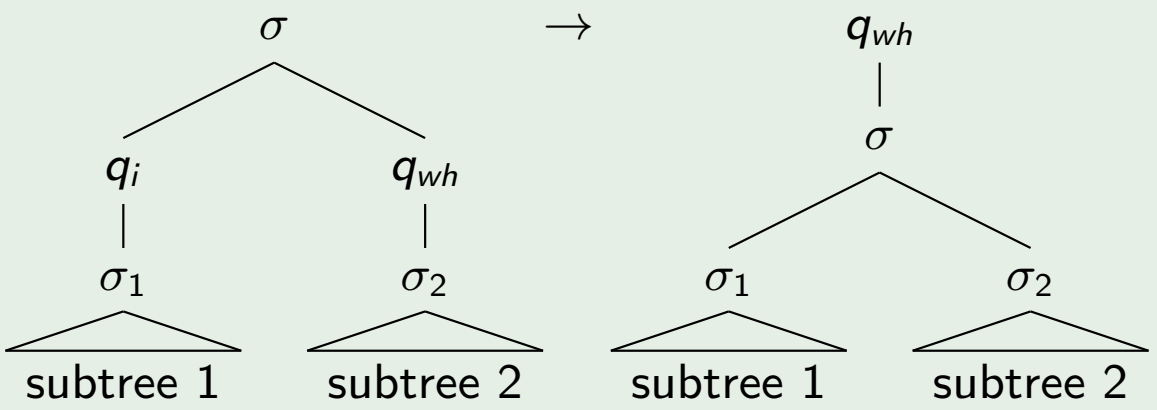
1) $\sigma \rightarrow$ $q_i$ / $\sigma$

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

3)

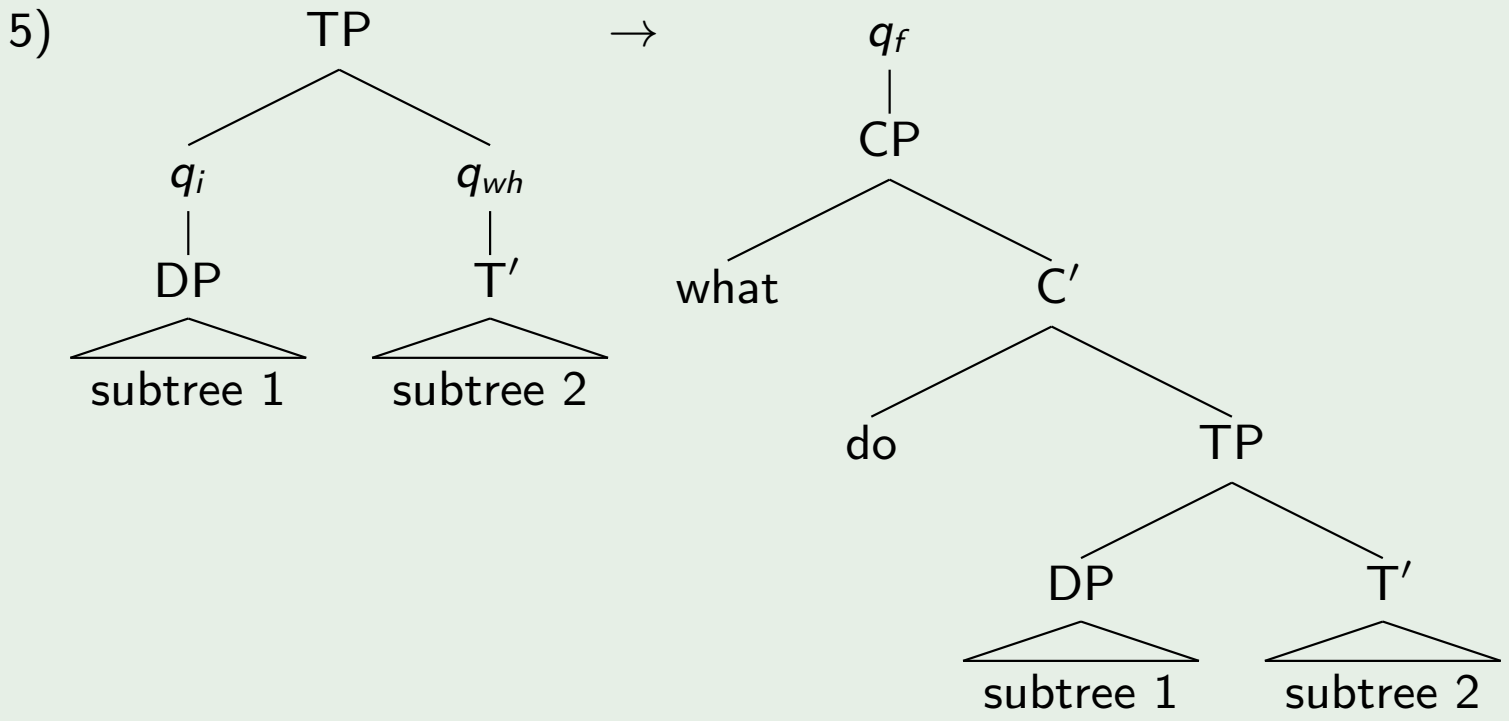



4)



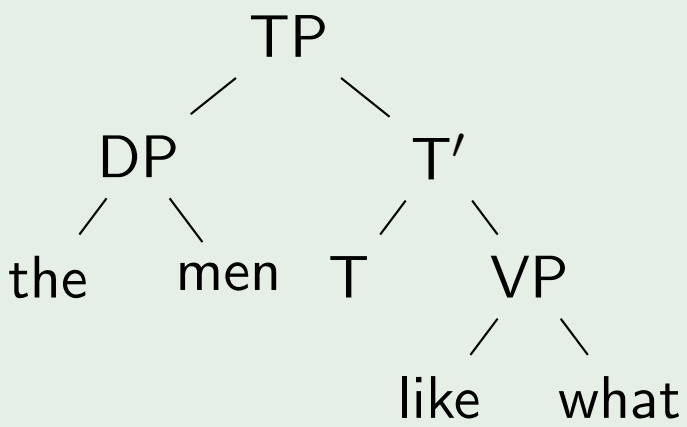
# A Simple Example (Part 2)

## A Transduction for Restricted wh-Movement, Rule 5



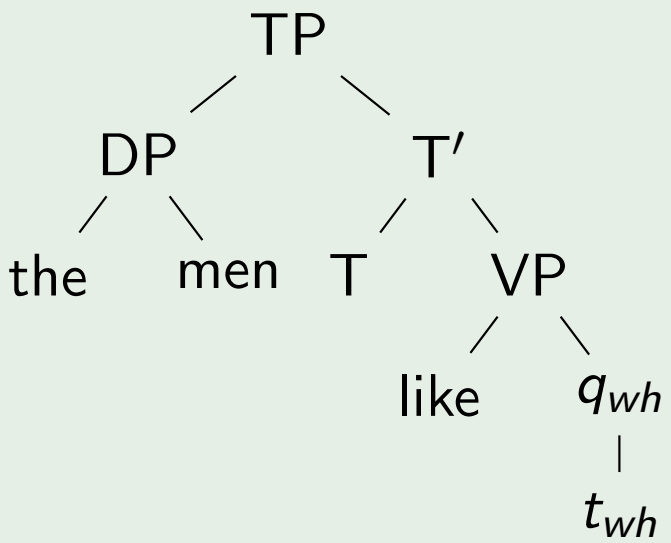
# A Simple Example (Part 3)

## A Transduction for Restricted wh-Movement, Application



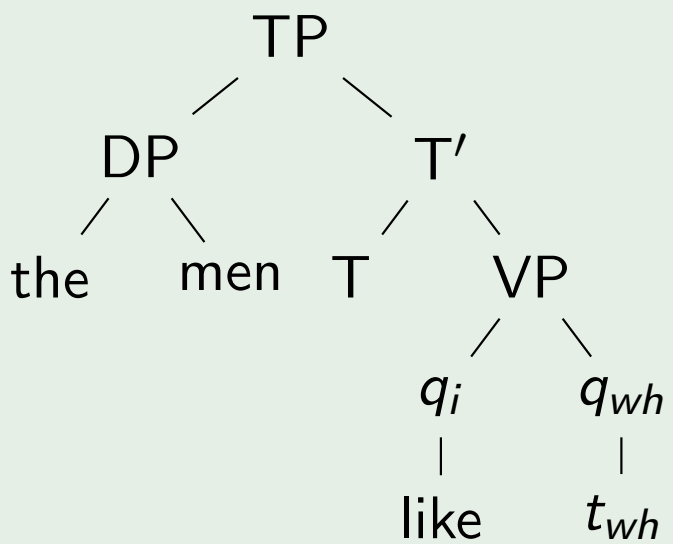
# A Simple Example (Part 3)

## A Transduction for Restricted wh-Movement, Application



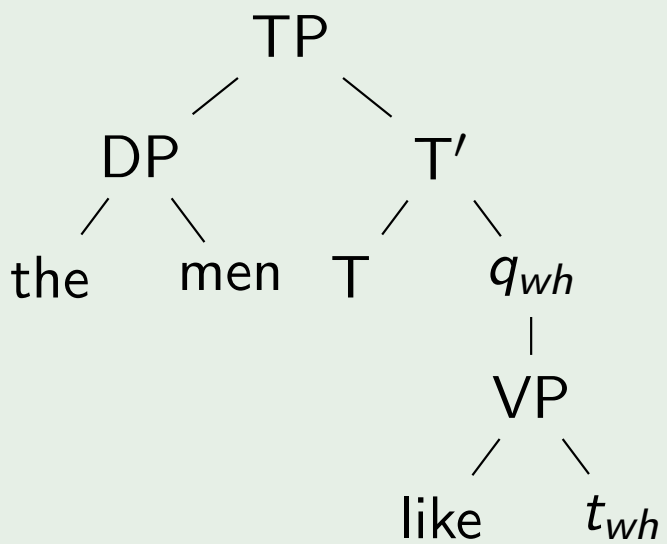
# A Simple Example (Part 3)

## A Transduction for Restricted wh-Movement, Application



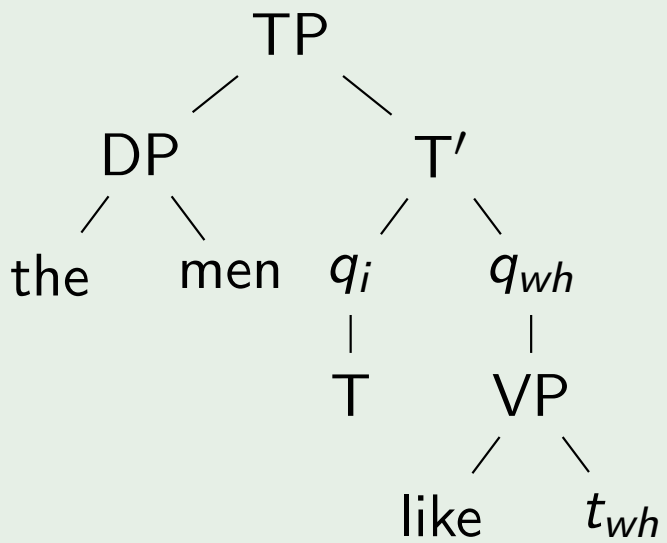
# A Simple Example (Part 3)

## A Transduction for Restricted wh-Movement, Application



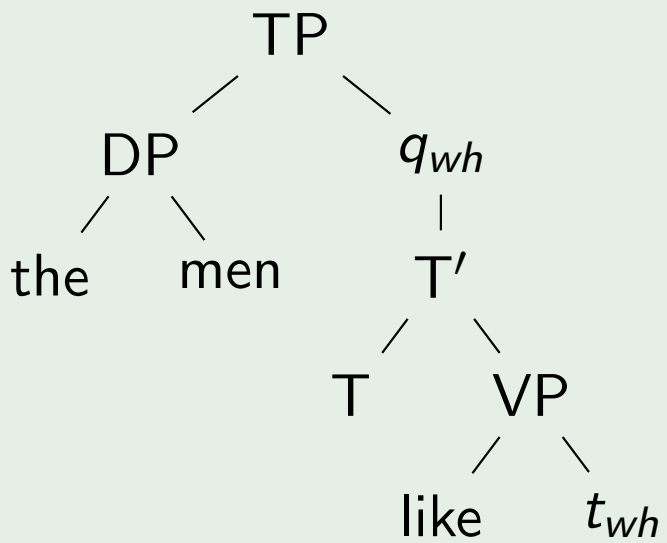
# A Simple Example (Part 3)

## A Transduction for Restricted wh-Movement, Application



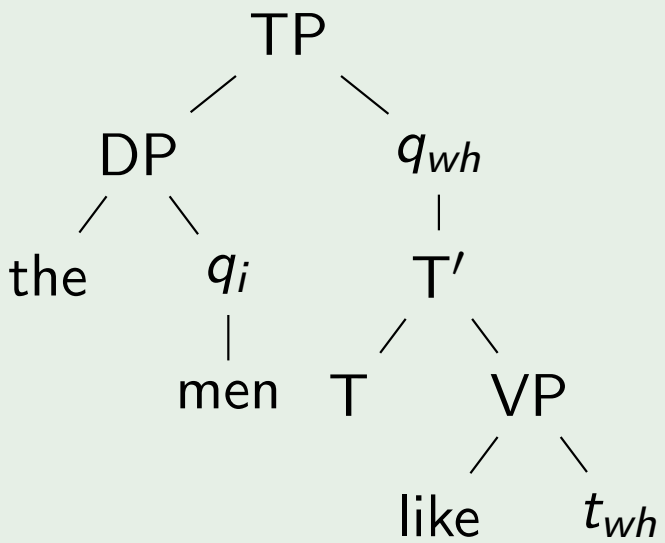
# A Simple Example (Part 3)

## A Transduction for Restricted wh-Movement, Application



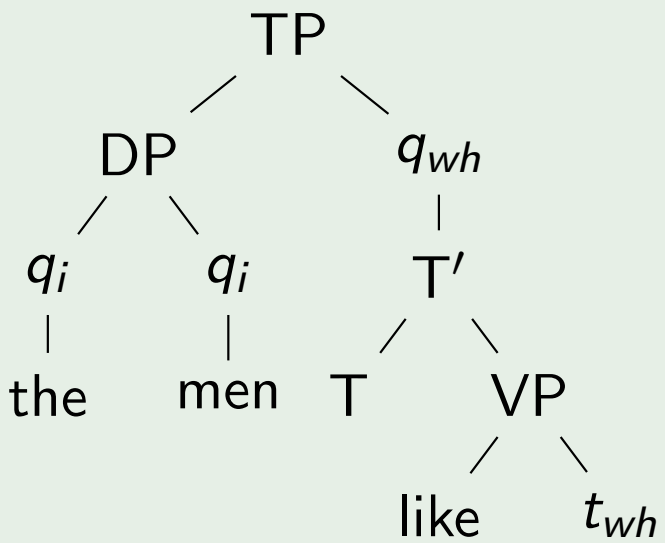
# A Simple Example (Part 3)

## A Transduction for Restricted wh-Movement, Application



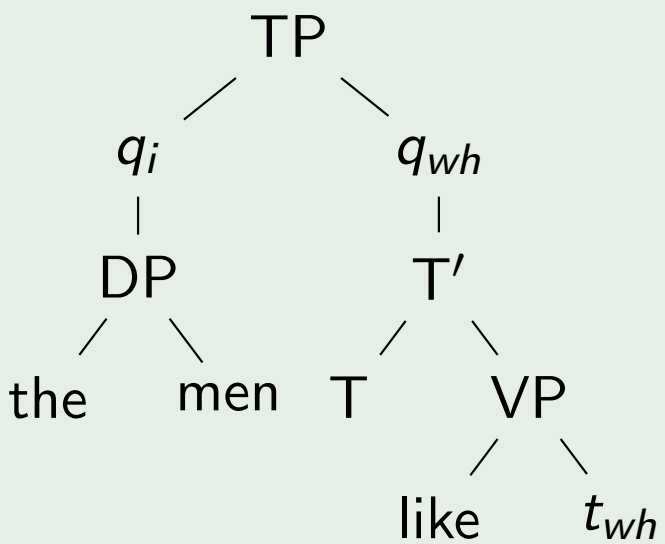
# A Simple Example (Part 3)

## A Transduction for Restricted wh-Movement, Application



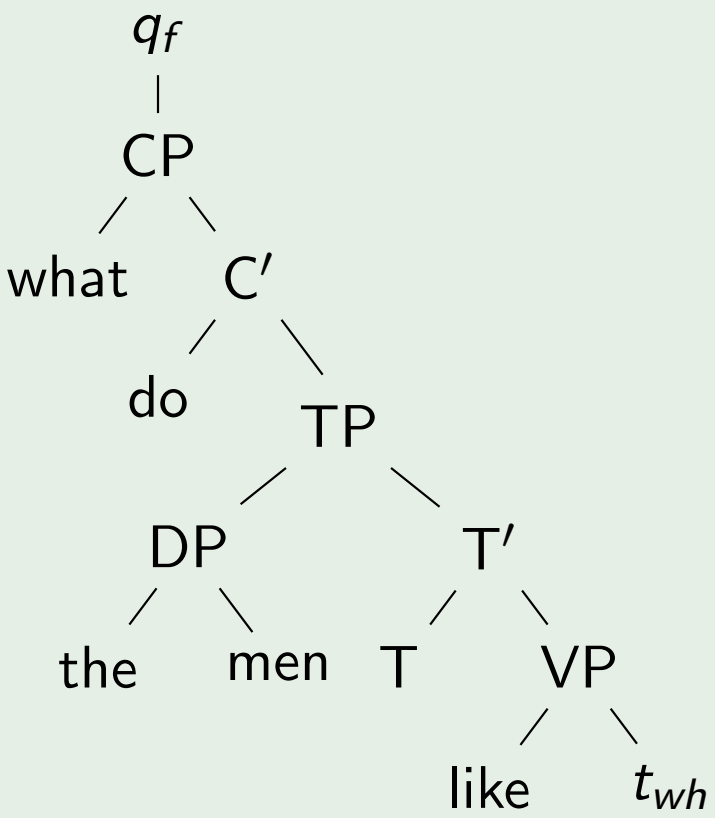
# A Simple Example (Part 3)

## 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 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.

# 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 $d'$.

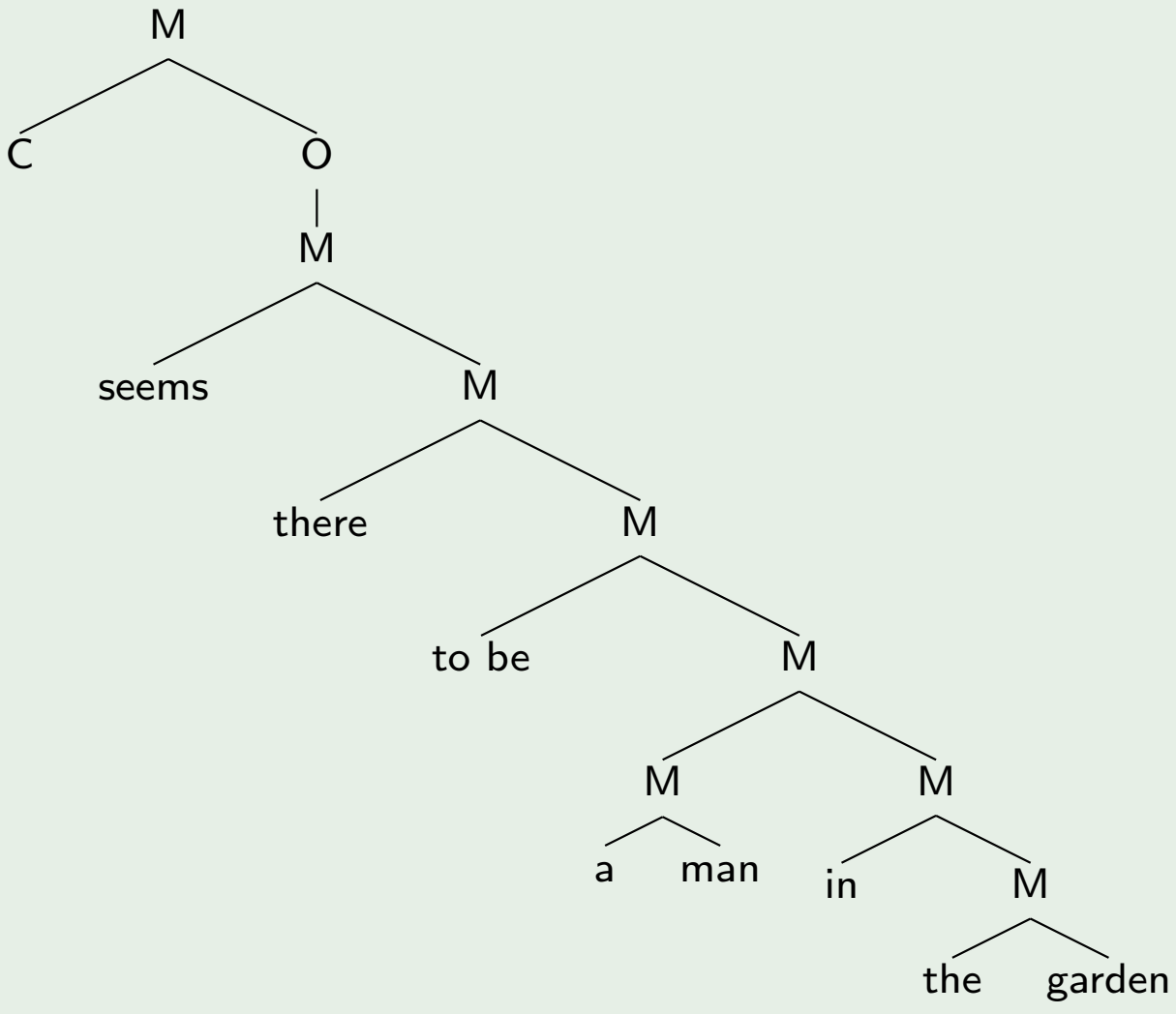
(3) a. There seems $t_{\text{there}}$ to be a man in the garden.

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.

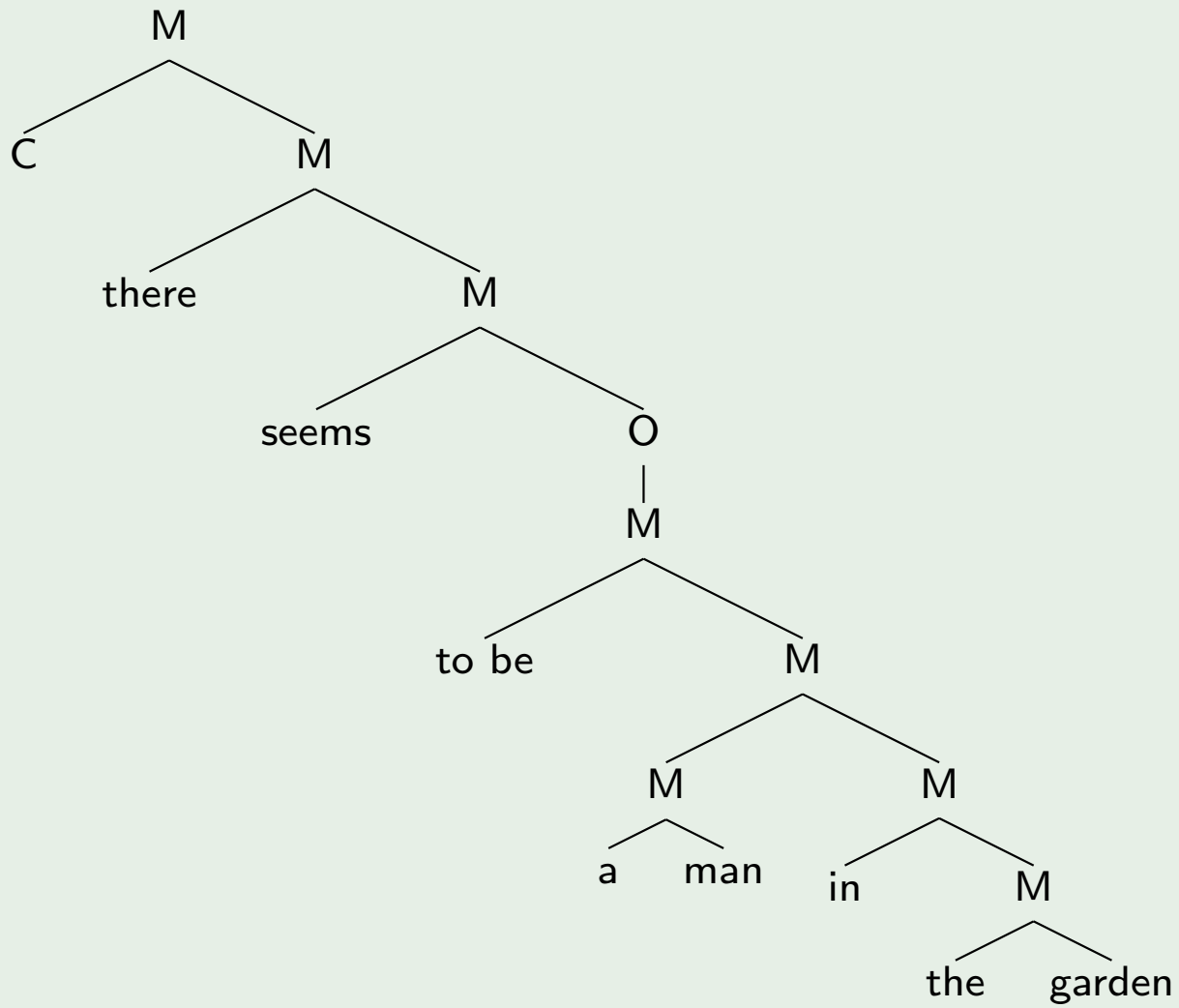
# Derivation Trees of (3a) and (3b)

## Example



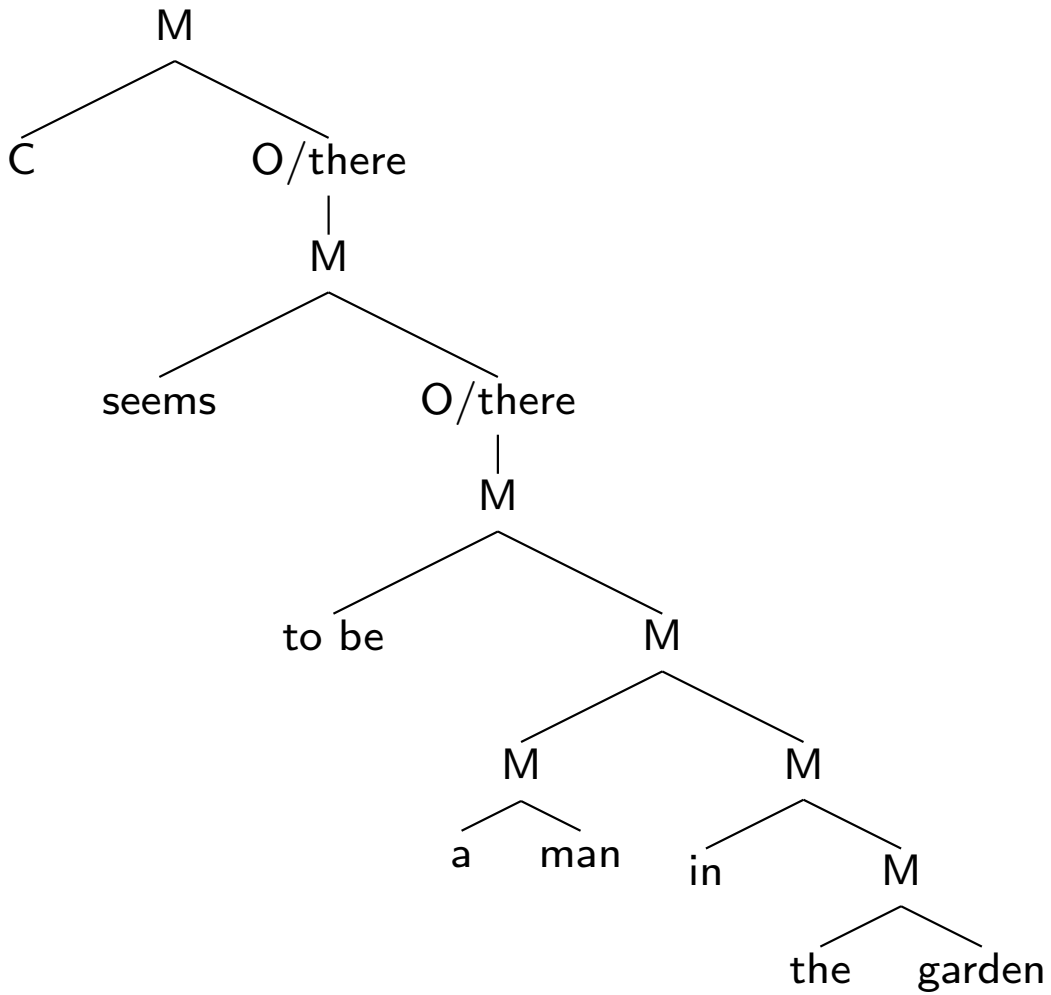
# Derivation Trees of (3a) and (3b)

## 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.

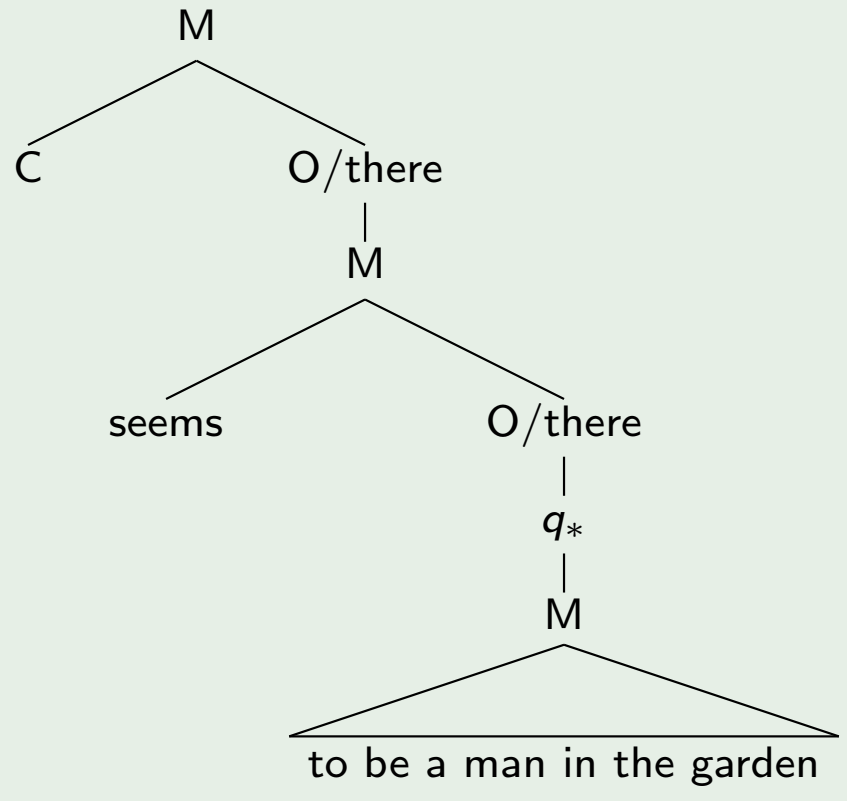


# Transducer Model: GEN (Step 2)

- 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.

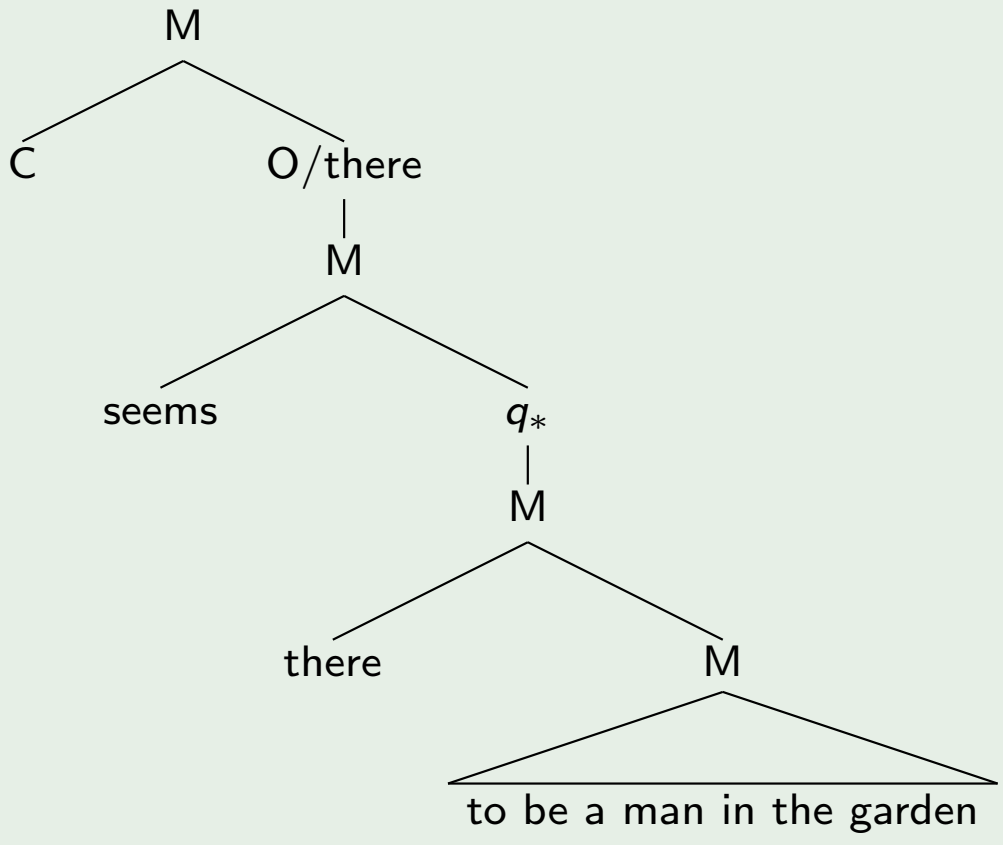
# Transducer Model: Examples of Step 2

## Example 1



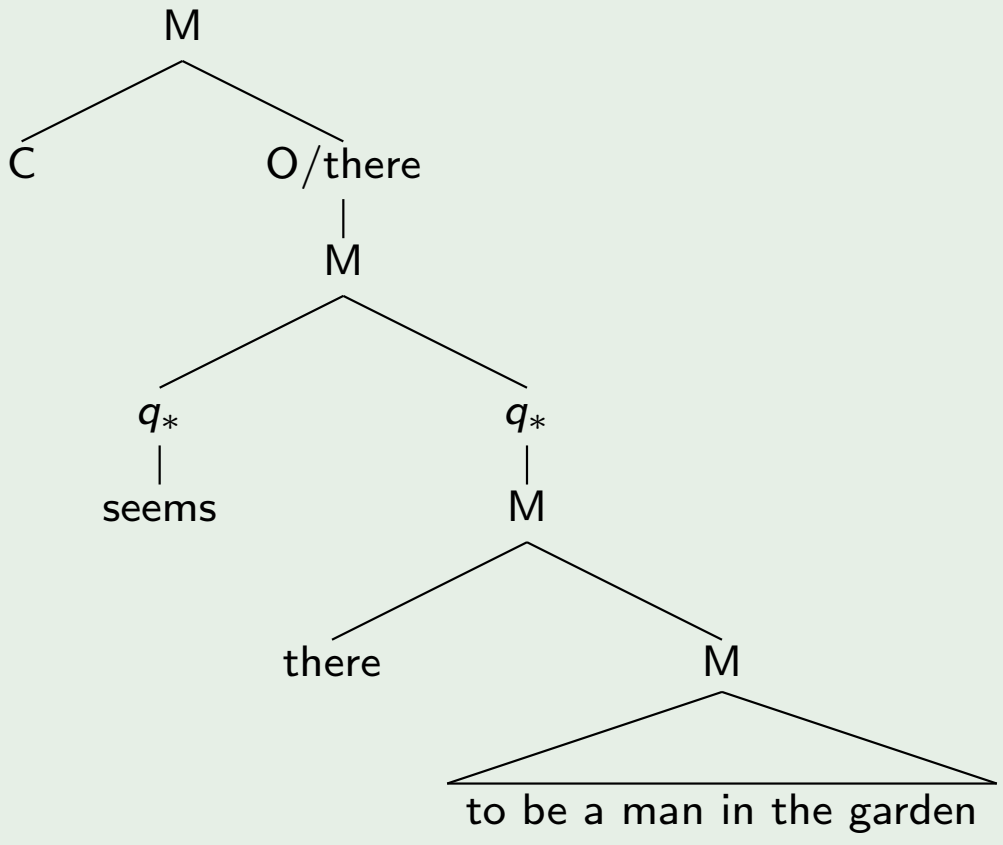
# Transducer Model: Examples of Step 2

## Example 1



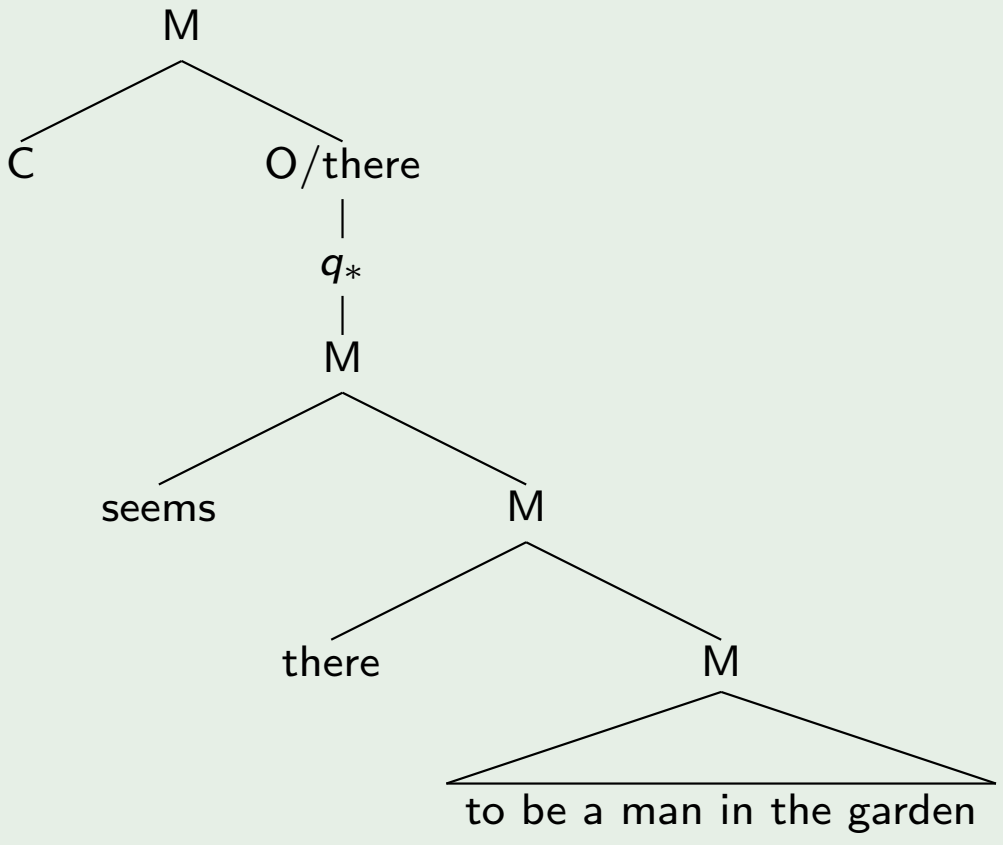
# Transducer Model: Examples of Step 2

## Example 1



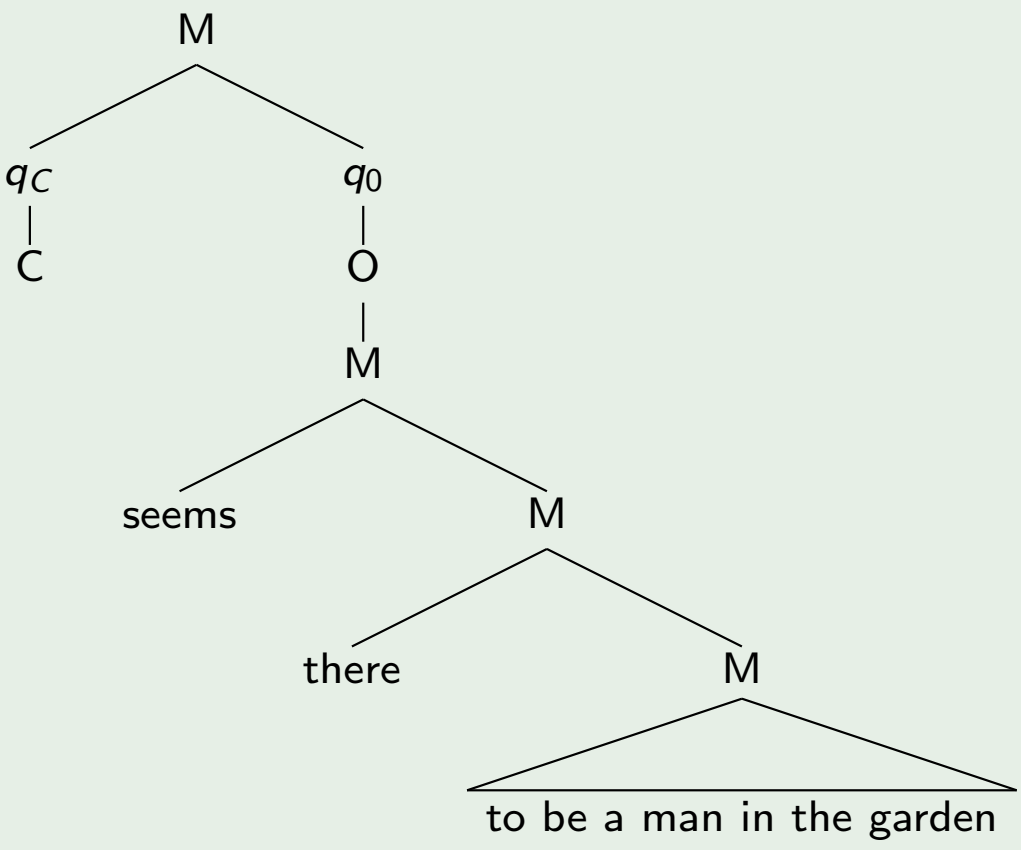
# Transducer Model: Examples of Step 2

## Example 1



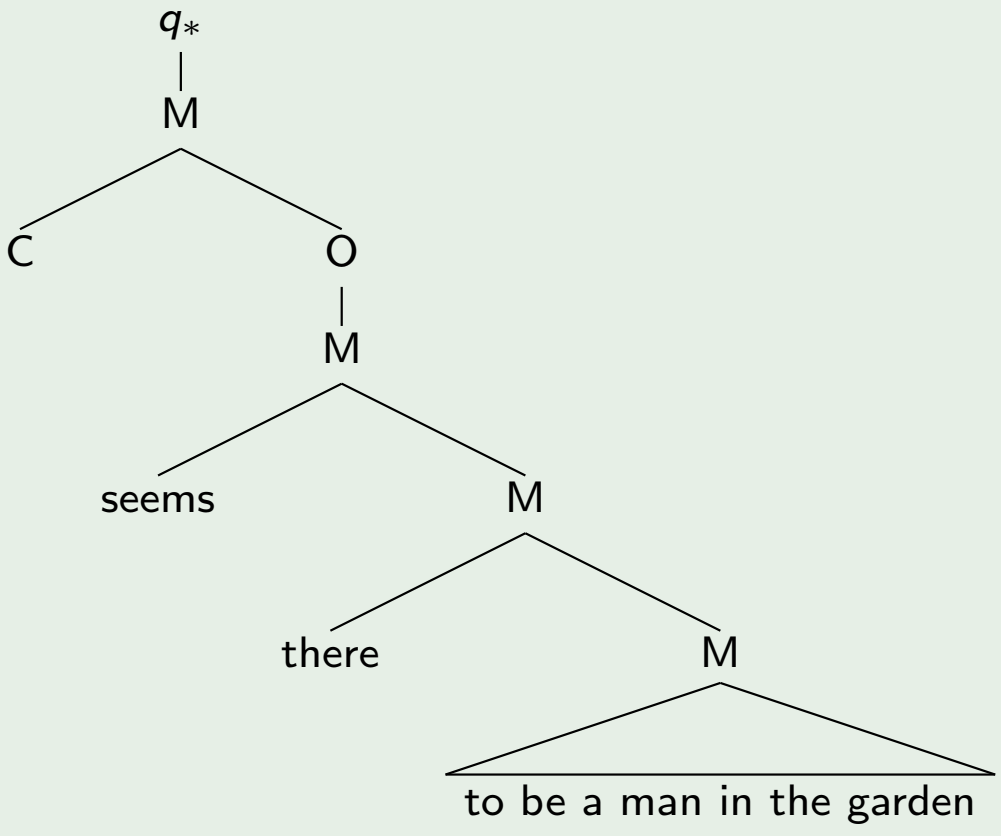
# Transducer Model: Examples of Step 2

## Example 1



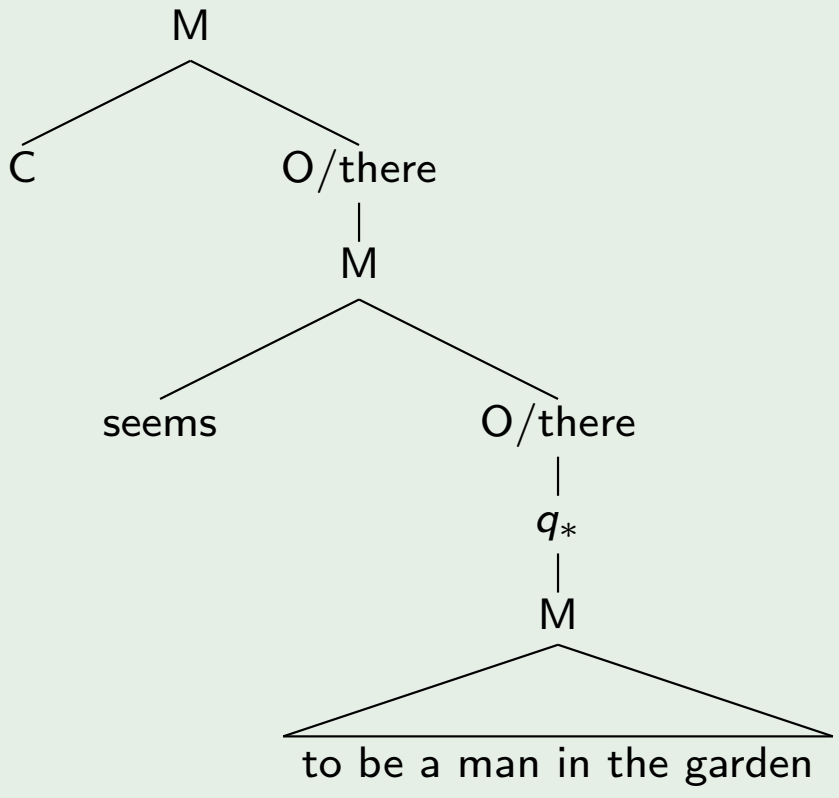
# Transducer Model: Examples of Step 2

## Example 1



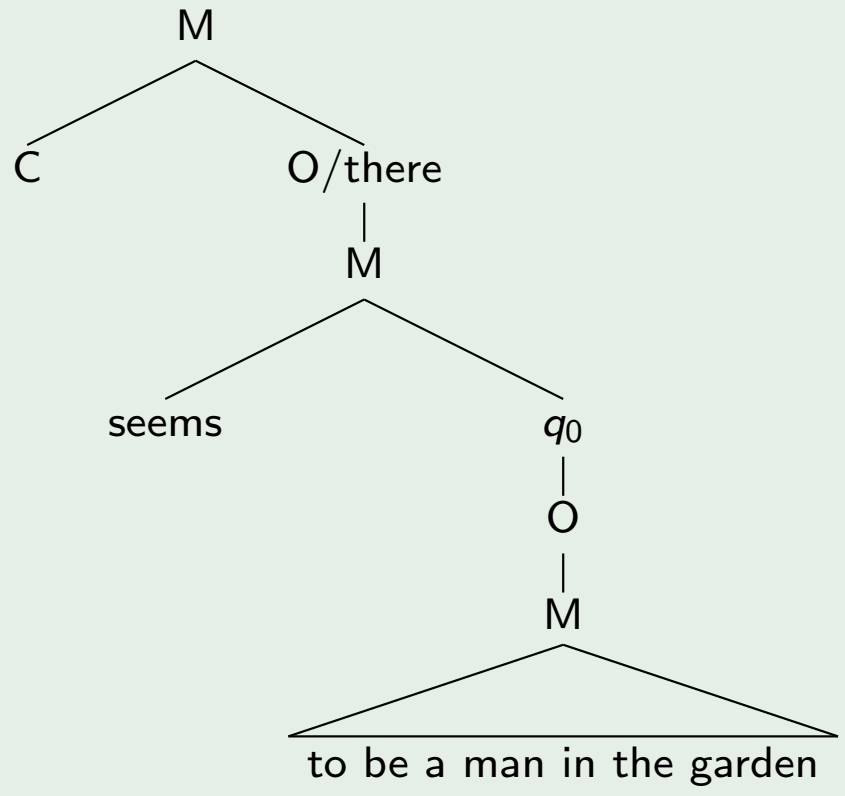
# Transducer Model: Examples of Step 2

## Example 2



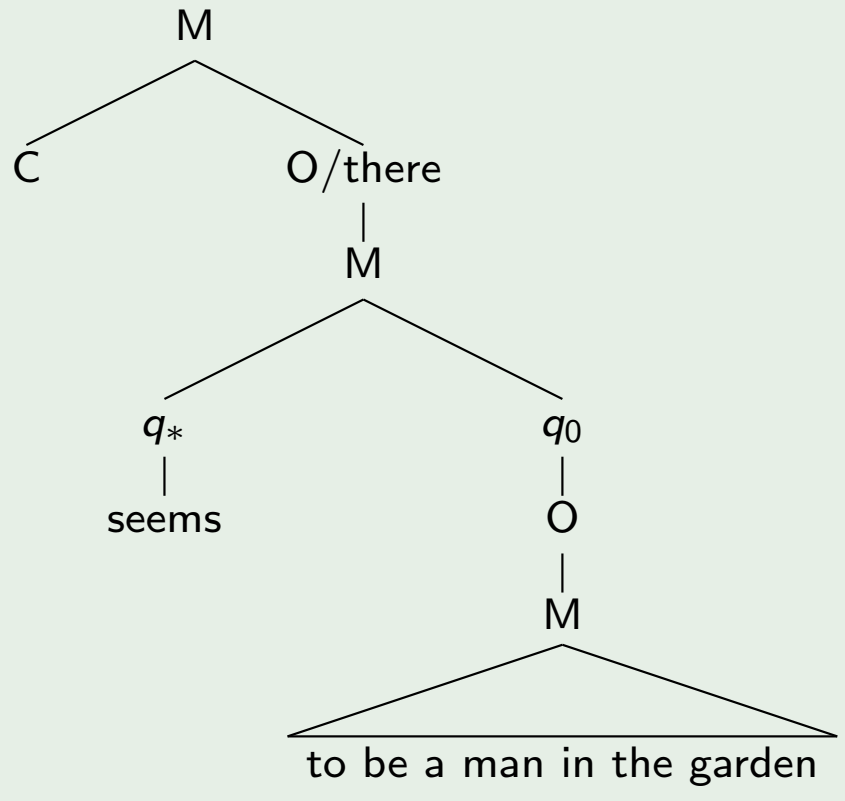
# Transducer Model: Examples of Step 2

## Example 2



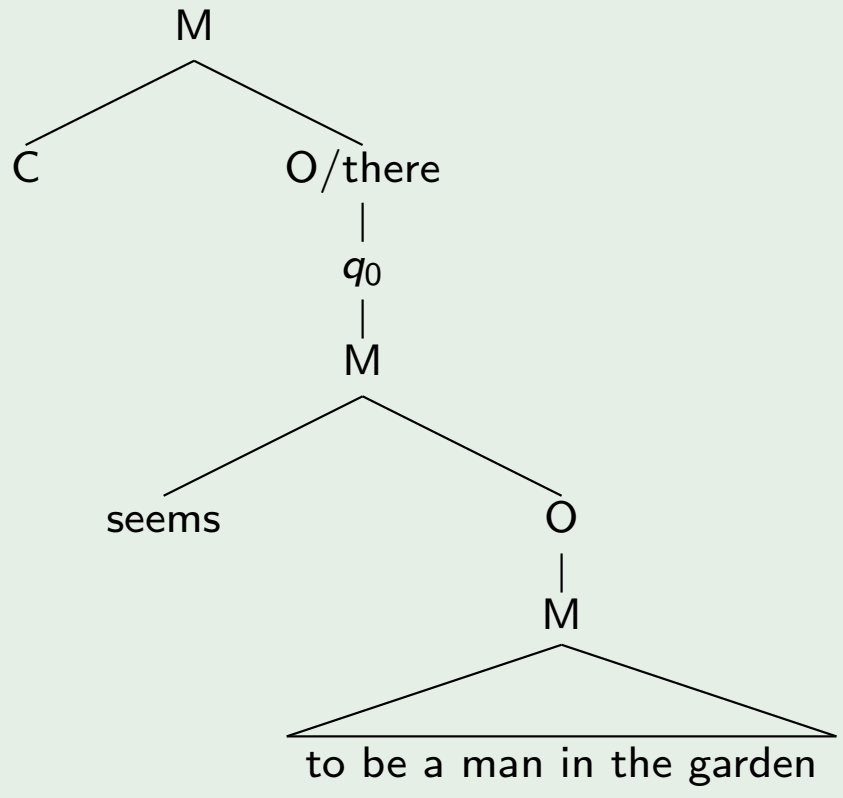
# Transducer Model: Examples of Step 2

## Example 2



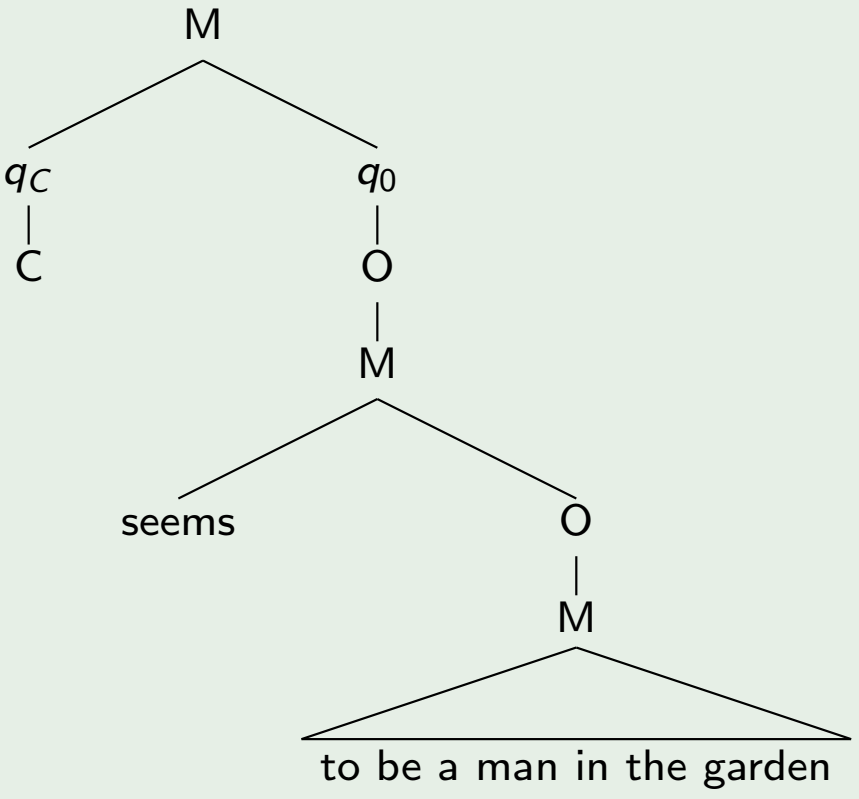
# Transducer Model: Examples of Step 2

## Example 2



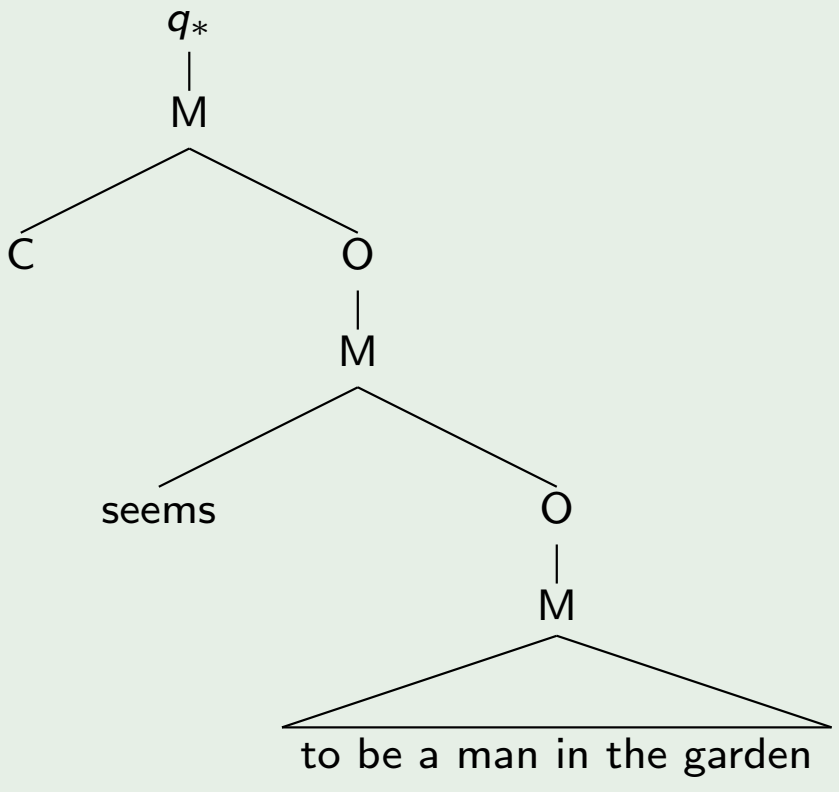
# Transducer Model: Examples of Step 2

## 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 $t_{\text{there}}$ 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.

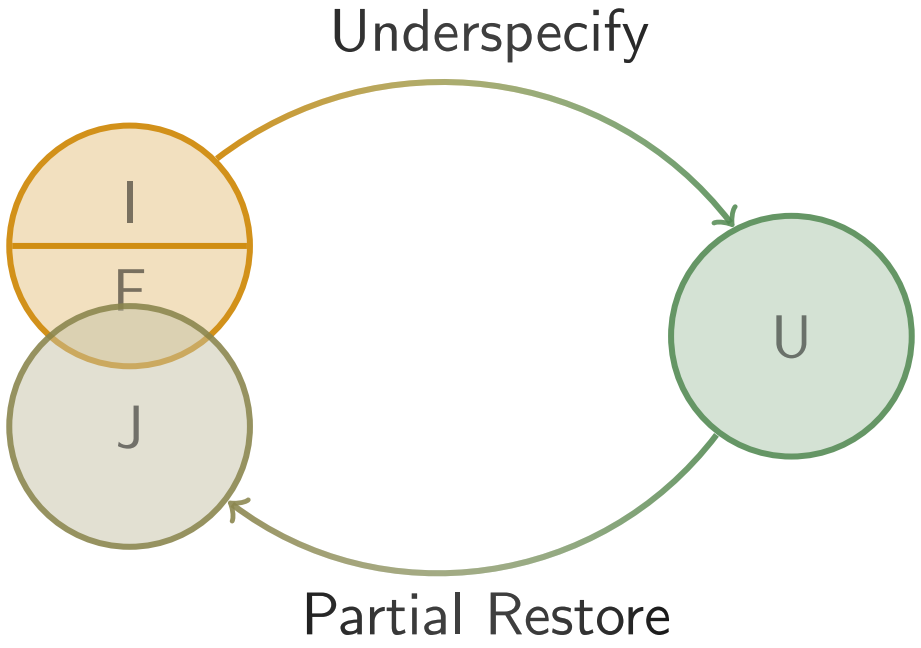
b. There seems $t_{\text{there}}$ to be a man in the garden.

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.



# Why Does Intersection Prevent Overgeneration?

- Intersecting the output language with the input language is tantamount to throwing away all trees that weren't generated by the original grammar
- If (5a) isn't in the input language, then it will be thrown away and thus the transducer does not overgenerate anymore.
- If (5a) is in the input language, then it will not be thrown away. But in this case the transducer didn't overgenerate in the first place, since (5a) was already in the input language, so it is supposedly grammatical and it isn't the job of MOM to rule it out.

# Upshot of Story 2 (Reference-Set Computation)

## 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.

If a reference-set constraint can be computed by a transducer, there is an equivalent local constraint.

# Upshot of Story 1 (Architecture of Language Faculty)

- The local correspondent of a reference-set constraint is efficiently computable.
- This allows syntax to trick the parser and use reference-set constraints after all.

## (Semi-)Open Questions

- Why would Narrow Syntax prefer the reference-set constraint over the local correspondent? Succinctness!
- Are there any other instances of syntax tricking the interfaces?

# 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.

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

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

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

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.