Reference-Set Constraints as Linear Tree Transductions via Controlled Optimality Systems

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

University of California, Los Angeles

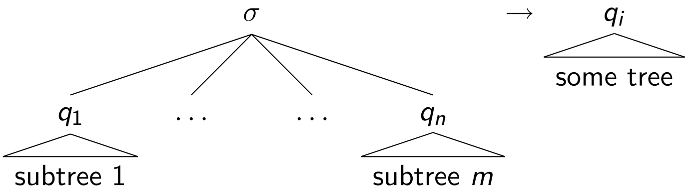
Formal Grammar 2010
Copenhagen, Denmark

Tree Transducers in Pictures

A finite-state bottom-up tree transducer

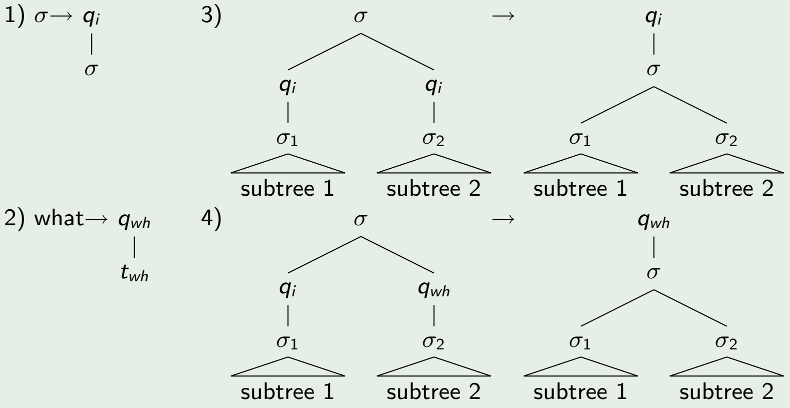
- traverses an input-tree from the leaves towards the root,
- labels it with states q_i, and
- transforms it into an output-tree.

It does so using rules of the following kind:



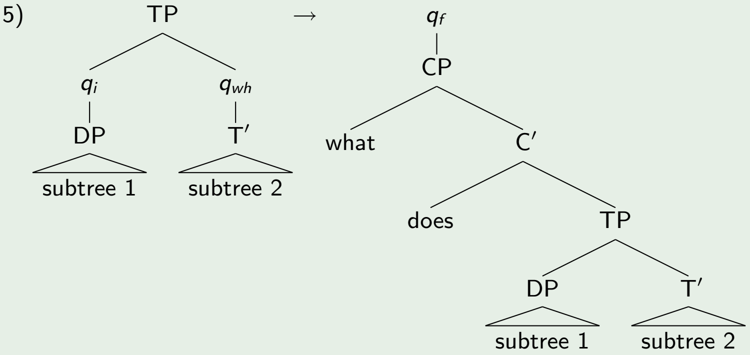
A Simple Example (Part 1)

A Transduction for wh-Movement, Rules 1–4



A Simple Example (Part 2)

A Transduction for wh-Movement, Rule 5



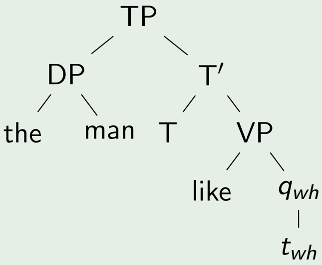
A Simple Example (Part 3)

A Transduction for wh-Movement, Application

[TP [DP the man] [T′ T [VP like what]]]

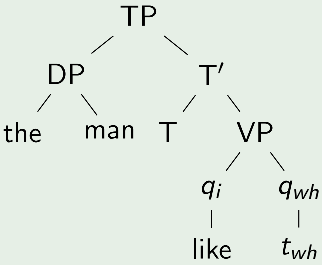
A Simple Example (Part 3)

A Transduction for wh-Movement, Application



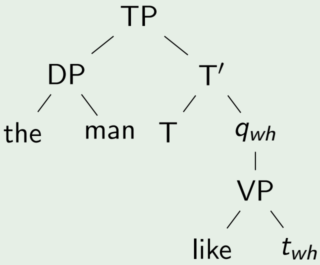
A Simple Example (Part 3)

A Transduction for wh-Movement, Application



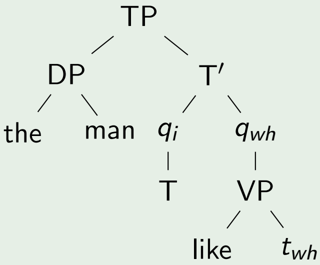
A Simple Example (Part 3)

A Transduction for wh-Movement, Application



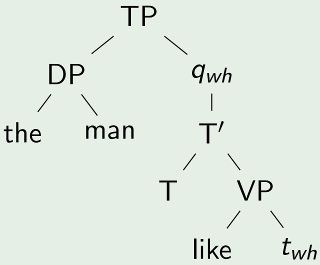
A Simple Example (Part 3)

A Transduction for wh-Movement, Application



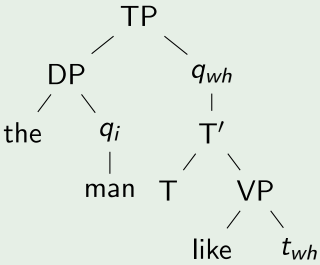
A Simple Example (Part 3)

A Transduction for wh-Movement, Application



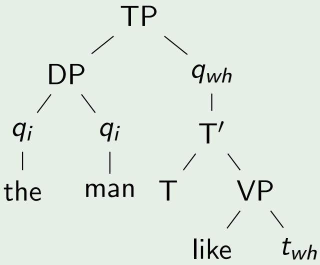
A Simple Example (Part 3)

A Transduction for wh-Movement, Application



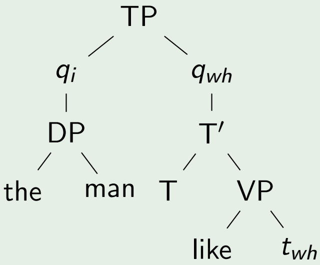
A Simple Example (Part 3)

A Transduction for wh-Movement, Application



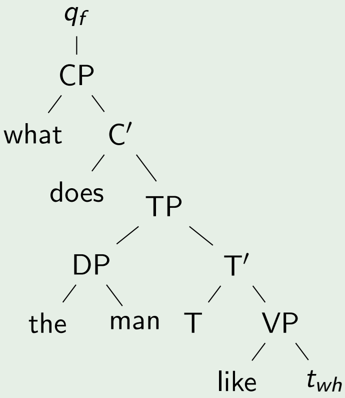
A Simple Example (Part 3)

A Transduction for wh-Movement, Application



A Simple Example (Part 3)

A Transduction for wh-Movement, Application



Some Important Facts

- There are also transducers that traverse a tree top-down.
- A transducer is linear iff it does not copy an subtrees, i.e. iff it has no rules where a subtree of the input occurs more than once on the right hand side of the rule.
- Every linear top-down transducer can be emulated by a linear bottom-up transducer.
- The class of linear bottom-up transducers is closed under composition.
- The class of regular tree languages is closed under linear transductions.
- The diagonal of a regular tree language is a linear transduction.

Outline

Reference-Set Constraints

An Informal Definition

Given some 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.

Examples in the literature

- Rule I (Reinhart 2006)
- Scope Economy (Fox 2000)
- Fewest Steps (Chomsky 1995)
- Merge-over-Move (Chomsky 2000)
- Resumption (Aoun et al. 2001)

⋮

Example: Focus Economy (Reinhart 2006)

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

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

Focus Projection

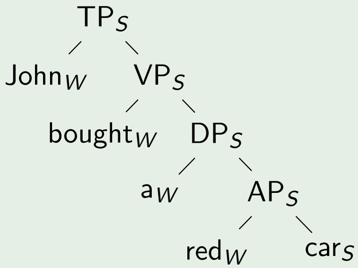
Any constituent containing the carrier of sentential main stress may be focused.

Focus Economy Rule

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

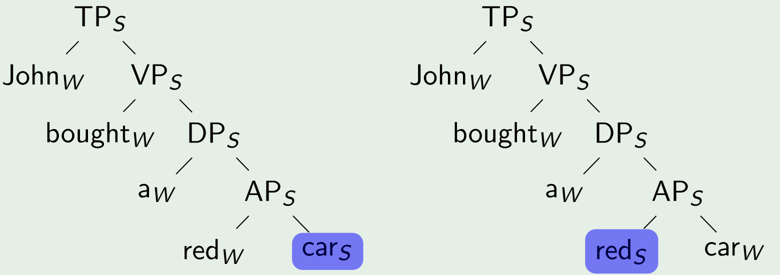
Example: Focus Economy, Cont.

Computing the Focus Sets



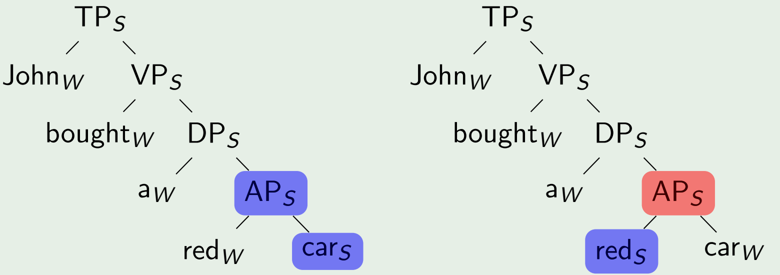
Example: Focus Economy, Cont.

Computing the Focus Sets



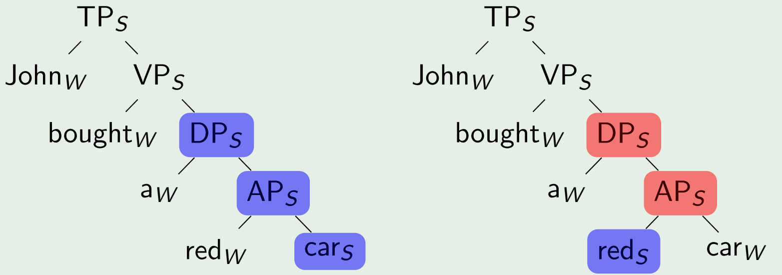
Example: Focus Economy, Cont.

Computing the Focus Sets



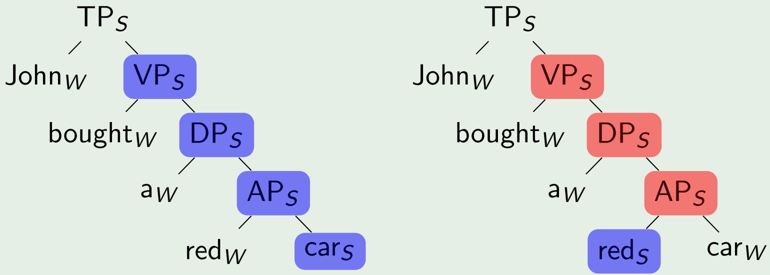
Example: Focus Economy, Cont.

Computing the Focus Sets



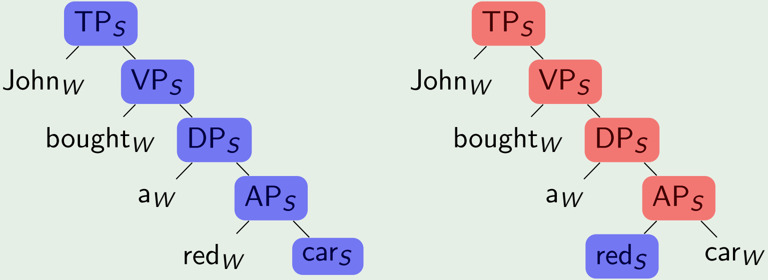
Example: Focus Economy, Cont.

Computing the Focus Sets



Example: Focus Economy, Cont.

Computing the Focus Sets



So What's the Deal?

- Reference-set constraints are believed to be too computationally demanding, for real-world applications as well as human cognition in general (Collins 1996; Johnson and Lappin 1999).
- Reasoning in the literature: Not only do we have to compute a (possibly infinite) reference-set, picking the optimal ouput also requires comparing distinct trees, in contrast to standard well-formedness condition.
- However, similar things were once said about a different piece of linguistic machinery that is now known to be efficiently computable: Optimality Theory. . .

Definition of Optimality Systems

Optimality Systems (Frank and Satta 1998)

An *optimality system* over input language L and output candidate language L' is a pair $\mathcal{O} := \langle \textsc{Gen}, C \rangle$ with

- $\textsc{Gen} \subseteq L \times L'$ a relation from inputs to output candidates,
- $C := \langle c_1, \ldots, c_n \rangle$ a linearly ordered sequence of functions $c_i \colon \textsc{Gen} \to \mathbb{N}$ that assign each input-output pair the number of violations it incurs with respect to the i^{th} constraint.

For pairs $p, q \in \textsc{Gen}$, p is more optimal than q iff there is an $1 \leq k \leq n$ such that $c_k(a) < c_k(b)$ and for all $j < k$, $c_j(a) = c_j(b)$.

The output language of $\mathcal{O}$ is the smallest set containing all $\langle i, o \rangle \in \textsc{Gen}$ for which it holds that there is no o' such that $\langle i, o' \rangle$ is more optimal than $\langle i, o \rangle$.

Example (Optimality System)

$$L := \{a\} \quad L' := \{a, aa, ab, b, bb, c\} \quad C := \langle {}^*c,\ \text{save } a,\ {}^*a,\ {}^*\#a \rangle$$

$$\textsc{Gen} := \{\langle a, a\rangle, \langle a, aa\rangle, \langle a, ab\rangle, \langle a, b\rangle, \langle a, bb\rangle, \langle a, c\rangle\}$$

Input a	$^*c$	save a	$^*a$	$^*\#a$
Output a				
Output aa				
Output ab				
Output b				
Output bb				
Output c				

Example (Optimality System)

$$L := \{a\} \quad L' := \{a, aa, ab, b, bb, c\} \quad C := \langle {}^{*}c,\ \text{save } a,\ {}^{*}a,\ {}^{*}\#a \rangle$$

$$\textsc{Gen} := \{\langle a, a\rangle, \langle a, aa\rangle, \langle a, ab\rangle, \langle a, b\rangle, \langle a, bb\rangle, \langle a, c\rangle\}$$

Input a	$^{*}c$	save a	$^{*}a$	$^{*}\#a$
Output a				
Output aa				
Output ab				
Output b				
Output bb				
Output c	1			

Example (Optimality System)

$$L := \{a\} \quad L' := \{a, aa, ab, b, bb, c\} \quad C := \langle {}^*c,\ \text{save } a,\ {}^*a,\ {}^*\#a \rangle$$

$$\textsc{Gen} := \{\langle a, a\rangle, \langle a, aa\rangle, \langle a, ab\rangle, \langle a, b\rangle, \langle a, bb\rangle, \langle a, c\rangle\}$$

Input a	${}^*c$	save a	${}^*a$	${}^*\#a$
Output a				
Output aa				
Output ab				
Output b				
Output bb				

Example (Optimality System)

$$L := \{a\} \quad L' := \{a, aa, ab, b, bb, c\} \quad C := \langle {}^*c,\ \text{save } a,\ {}^*a,\ {}^*\#a \rangle$$

$$\textsc{Gen} := \{\langle a, a\rangle, \langle a, aa\rangle, \langle a, ab\rangle, \langle a, b\rangle, \langle a, bb\rangle, \langle a, c\rangle\}$$

Input a	${}^*c$	save a	${}^*a$	${}^*\#a$
Output a				
Output aa				
Output ab				
Output b		1		
Output bb		1		

Example (Optimality System)

$$L := \{a\} \quad L' := \{a, aa, ab, b, bb, c\} \quad C := \langle {}^*c,\ \text{save } a,\ {}^*a,\ {}^*\#a \rangle$$

$$\textsc{Gen} := \{\langle a, a\rangle, \langle a, aa\rangle, \langle a, ab\rangle, \langle a, b\rangle, \langle a, bb\rangle, \langle a, c\rangle\}$$

Input a	$^*c$	save a	$^*a$	$^*\#a$
Output a				
Output aa				
Output ab				

Example (Optimality System)

$$L := \{a\} \quad L' := \{a, aa, ab, b, bb, c\} \quad C := \langle {}^*c,\ \text{save } a,\ {}^*a,\ {}^*\#a \rangle$$

$$\textsc{Gen} := \{\langle a, a\rangle, \langle a, aa\rangle, \langle a, ab\rangle, \langle a, b\rangle, \langle a, bb\rangle, \langle a, c\rangle\}$$

Input a	$^*c$	save a	$^*a$	$^*\#a$
Output a			1	
Output aa			2	
Output ab			1	

Example (Optimality System)

$$L := \{a\} \quad L' := \{a, aa, ab, b, bb, c\} \quad C := \langle {}^*c,\ \text{save } a,\ {}^*a,\ {}^*\#a \rangle$$

$$\textsc{Gen} := \{\langle a, a\rangle, \langle a, aa\rangle, \langle a, ab\rangle, \langle a, b\rangle, \langle a, bb\rangle, \langle a, c\rangle\}$$

Input a	${}^*c$	save a	${}^*a$	${}^*\#a$
Output a			1	
Output ab			1	

Example (Optimality System)

$$L := \{a\} \quad L' := \{a, aa, ab, b, bb, c\} \quad C := \langle {}^*c,\ \text{save } a,\ {}^*a,\ {}^*\#a \rangle$$

$$\textsc{Gen} := \{\langle a, a\rangle, \langle a, aa\rangle, \langle a, ab\rangle, \langle a, b\rangle, \langle a, bb\rangle, \langle a, c\rangle\}$$

Input a	$^*c$	save a	$^*a$	$^*\#a$
Output a			1	1
Output ab			1	1

Example (Optimality System)

$L := \{a\} \quad L' := \{a, aa, ab, b, bb, c\} \quad C := \langle {}^*c,\ \text{save } a,\ {}^*a,\ {}^*\#a \rangle$

$$\textsc{Gen} := \{\langle a, a\rangle, \langle a, aa\rangle, \langle a, ab\rangle, \langle a, b\rangle, \langle a, bb\rangle, \langle a, c\rangle\}$$

Input a	${}^*c$	save a	${}^*a$	${}^*\#a$
Output a			1	1
Output ab			1	1

transduction $\tau := \{\langle a, a\rangle, \langle a, ab\rangle\}$ output language $:= \{a, ab\}$

OS as Linear Tree Transducers

Without further restrictions, OSs can generate any kind of string and tree language. However, a few conditions suffice to restrict them to the power of linear tree transducers.

Finite-State OSs (Wartena 2000; Jäger 2002)

Let $\mathcal{O} := \langle \textsc{Gen}, C \rangle$ be an OS such that

- the domain of $\textsc{Gen}$ is a regular tree language,
- $\textsc{Gen}$ is a linear tree transduction,
- all constraints are insensitive to the input,
- each constraint defines a linear tree transduction on the range of $\textsc{Gen}$,
- $\mathcal{O}$ is globally optimal.

Then the transduction τ induced by $\mathcal{O}$ is a linear tree transduction and its range is a regular tree language.

Global Optimality

Global Optimality

An OS is globally optimal iff for every output candidate o that is optimal for some input i it holds that there is no input i' such that o is an output candidate for i but not an optimal one.

Examples

- An OS with $\textsc{Gen} := \{i, i'\} \times \{o, o'\}$ and only $\langle i, o \rangle$ and $\langle i', o' \rangle$ as the optimal pairings is not globally optimal
  $\Rightarrow$ if constraints may take the input into account, global optimally is the exception
- An OS with $\textsc{Gen} := \{\langle i, o \rangle, \langle i, o' \rangle, \langle i', o' \rangle\}$ and o a universally better candidate than o' is not globally optimal
  $\Rightarrow$ input-insensitivity is no guarantee for global optimality

Reference-Set Constraints as OSs

General Strategy

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

But Gen is a "flat" relation, is does not directly represent reference-sets and their algebraic properties.

Maybe we can enrich OSs appropriately?

Outline

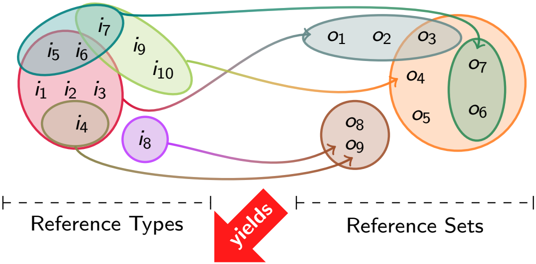
Definition of Controlled OSs

Controlled OSs

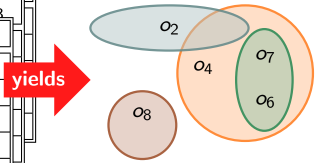
An $\mathcal{F}$*-controlled optimality system* over languages L, L' is a 4-tuple $\mathcal{O}[\mathcal{F}] := \langle \textsc{Gen}, C, \mathcal{F}, \gamma \rangle$, where

- $\textsc{Gen}$ and C are defined as usual,
- $\mathcal{F}$ is a family of non-empty subsets of L, each of which we call a *reference type*,
- the *control map* $\gamma : \mathcal{F} \to \wp(L')$ associates every reference type with a non-empty set of output candidates, the *reference set*,
- the following conditions are satisfied
    - *exhaustivity*: every member of the input language belongs to at least one reference type
    - *bootstrapping*: if input i belongs to the reference types $X_1, \ldots, X_n$, then $\langle i, o \rangle \in \textsc{Gen}$ iff $o \in \bigcup \{X_1\gamma, \ldots, X_n\gamma\}$

Depiction of a Controlled OS



i_4	c_1	c_2	c_3
o_1	1	2	0
☞ o_2	0	0	4
o_3	2	2	1
☞ o_8	0	0	4
o_9	0	1	0

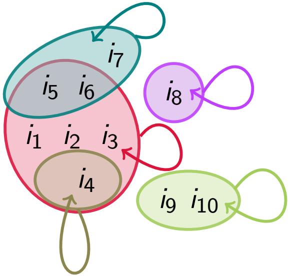


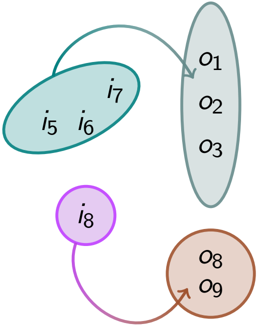
Evaluation

Output Language

Reference-set constraints as controlled OSs

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





Two Common Properties of Reference-Set Constraints

Output Joint Preservation (restricts $\textsc{Gen}$)

An $\mathcal{F}$-controlled OS is *output joint preserving* iff it holds for all reference types X and Y in $\mathcal{F}$ that whenever their reference sets $X\gamma$ and $Y\gamma$ overlap, so do X and Y themselves.

Output joint preservation is a strong restriction on $\textsc{Gen}$.
When it comes to C, many reference set constraints do not use the full range of options either:

Type-Level Optimality (restricts C)

An $\mathcal{F}$-controlled OS is *type-level optimal* iff it holds for all reference types $X \in \mathcal{F}$ and output candidates o in the reference set of X that if o is optimal for some input in X, it is optimal for all inputs that belong to X.

Global Optimality Implies Type-Level Optimality

Lemma

An OS is type-level optimal if it is globally optimal.
If all constraints of an OS are insensitive to the input, it is type-level optimal.

Intuitively, the first statement is obvious because type-level optimality is "global optimality restricted to single reference-sets".

Proof.

Prove the contrapositive. If the OS is not type-level optimal, then there is some reference type X whose reference set contains an output candidate z on which at least two inputs that are contained by X disagree with respect to optimality.
This is an unequivocal violation of global optimality (optimal for some input $\rightarrow$ optimal for every input). $\square$

Global Optimality Implies Type-Level Optimality

Lemma

An OS is type-level optimal if it is globally optimal.
If all constraints of an OS are insensitive to the input, it is type-level optimal.

Intuitively, the first statement is obvious because type-level optimality is "global optimality restricted to single reference-sets".

Proof.

Prove the contrapositive. If the OS is not type-level optimal, then there is some reference type X whose reference set contains an output candidate z on which at least two inputs that are contained by X disagree with respect to optimality.
This is an unequivocal violation of global optimality (optimal for some input $\rightarrow$ optimal for every input). $\square$

Global Optimality for Reference-Set Constraints

Theorem (Characterization of Global Optimality)

Every output joint preserving OS is type-level optimal iff it is globally optimal.

Proof.

$\Leftarrow$: Follows from the previous lemma.
$\Rightarrow$: An indirect proof of the contrapositive is given on the next slide (in pictures!). $\square$

Global Optimality for Reference-Set Constraints

Theorem (Characterization of Global Optimality)

Every output joint preserving OS is type-level optimal iff it is globally optimal.

Proof.

$\Leftarrow$: Follows from the previous lemma.
$\Rightarrow$: An indirect proof of the contrapositive is given on the next slide (in pictures!). $\square$

Pictorial Proof

To Show

If an output joint preserving OS fails global optimality, it also fails type-level optimality.

i_5 x p y

o_1 o_2 z o_8 o_9

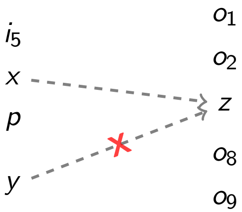
Assumptions

- ¬global optimality
- controlled OS
- type-level optimality
- output joint preservation

Pictorial Proof

To Show

If an output joint preserving OS fails global optimality,
it also fails type-level optimality.



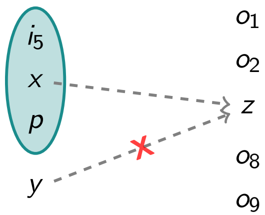
Assumptions

- $\neg$global optimality
- controlled OS
- type-level optimality
- output joint preservation

Pictorial Proof

To Show

If an output joint preserving OS fails global optimality, it also fails type-level optimality.



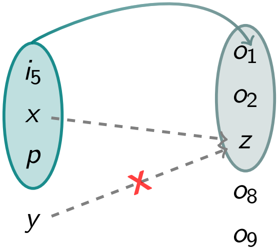
Assumptions

- ¬global optimality
- controlled OS
- type-level optimality
- output joint preservation

Pictorial Proof

To Show

If an output joint preserving OS fails global optimality,
it also fails type-level optimality.



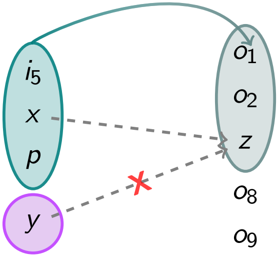
Assumptions

- ¬global optimality
- controlled OS
- type-level optimality
- output joint preservation

Pictorial Proof

To Show

If an output joint preserving OS fails global optimality,
it also fails type-level optimality.



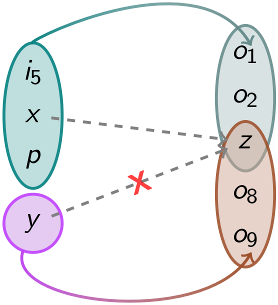
Assumptions

- ¬global optimality
- controlled OS
- type-level optimality
- output joint preservation

Pictorial Proof

To Show

If an output joint preserving OS fails global optimality, it also fails type-level optimality.



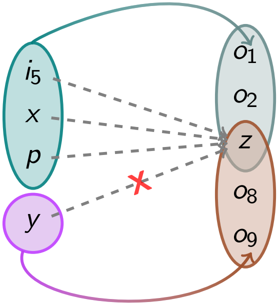
Assumptions

- ¬global optimality
- controlled OS
- type-level optimality
- output joint preservation

Pictorial Proof

To Show

If an output joint preserving OS fails global optimality, it also fails type-level optimality.



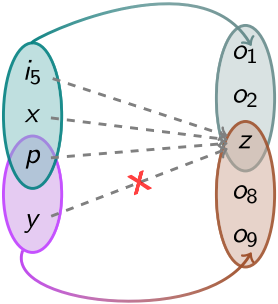
Assumptions

- ¬global optimality
- controlled OS
- type-level optimality
- output joint preservation

Pictorial Proof

To Show

If an output joint preserving OS fails global optimality, it also fails type-level optimality.



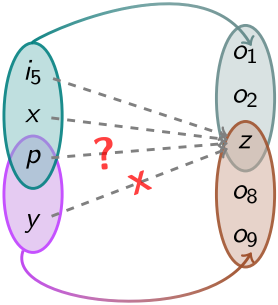
Assumptions

- ¬global optimality
- controlled OS
- type-level optimality
- output joint preservation

Pictorial Proof

To Show

If an output joint preserving OS fails global optimality,
it also fails type-level optimality.



Assumptions

- $\neg$global optimality
- controlled OS
- type-level optimality
- output joint preservation

Reference-Set Constraints as Linear Tree Transductions

These new results allow us to state when a controlled OS can be realized by a linear tree transducer. Note that the elusive property of global optimality has been replaced by output joint preservation, which is satisfied by all reference-set constraints.

Finite-State Controlled OSs

Let $\mathcal{O}[\mathcal{F}] := \langle \textsc{Gen}, C, \mathcal{F}, \gamma \rangle$ an $\mathcal{F}$-controlled OS such that

- the domain of $\textsc{Gen}$ is a regular tree language,
- $\textsc{Gen}$ is a linear tree transduction,
- all constraints are insensitive to the input,
- each constraint defines a linear tree transduction on the range of $\textsc{Gen}$,
- $\mathcal{O}[\mathcal{F}]$ is output joint preserving.

Then the transduction τ induced by $\mathcal{O}$ is a linear tree transduction and its range is a regular tree language.

Reference-Set Constraints as Linear Tree Transductions

These new results allow us to state when a controlled OS can be realized by a linear tree transducer. Note that the elusive property of global optimality has been replaced by output joint preservation, which is satisfied by all reference-set constraints.

Finite-State Controlled OSs

Let $\mathcal{O}[\mathcal{F}] := \langle \textsc{Gen}, C, \mathcal{F}, \gamma \rangle$ an $\mathcal{F}$-controlled OS such that

- the domain of $\textsc{Gen}$ is a regular tree language,
- $\textsc{Gen}$ is a linear tree transduction,
- all constraints are insensitive to the input,
- each constraint defines a linear tree transduction on the range of $\textsc{Gen}$,
- $\mathcal{O}[\mathcal{F}]$ is output joint preserving.

Then the transduction τ induced by $\mathcal{O}$ is a linear tree transduction and its range is a regular tree language.

Outline

Focus Economy Revisited

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.

Informal Derivational Order

- Compute output tree with neutral stress.
- Project focus according to Focus Projection.
- Optionally: Shift stress and recompute focus according to Focus Economy.

Formal Derivational Order

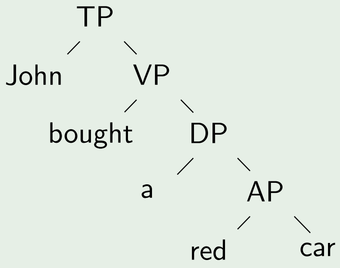
- Compute all stress patterns ($\rightarrow$ multiple output trees).
- Project focus according to Focus Projection.
- Filter out illicit focus projections according to Focus Economy.

Formal 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

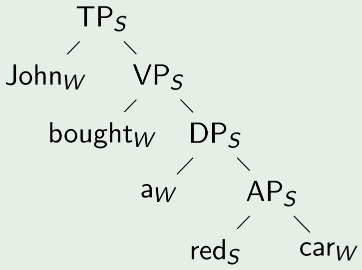


Formal Model: Gen

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

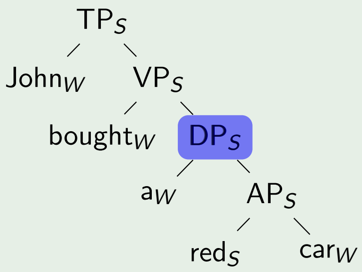


Formal Model: Gen

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



Formal Model: C

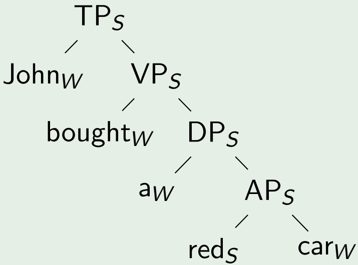
Focus Economy requires reference to the neutral stress pattern. We allow this by implicitly representing the neutral stress within the same tree!

Strategy

- Define monadic second-order predicates FocusPath and StressPath.
- FocusPath represents the path of the current stress.
- StressPath represents the path of the neutral stress.
- Write a formula ϕ that requires focus to be in the focus path, but unless FocusPath and StressPath pick out the same set, focus may not be in the stress path.

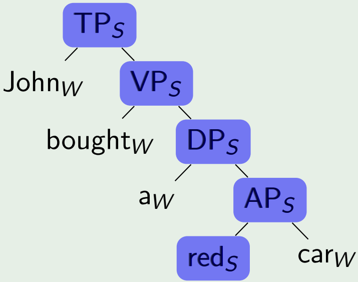
Example of ϕ

FocusPath and StressPath



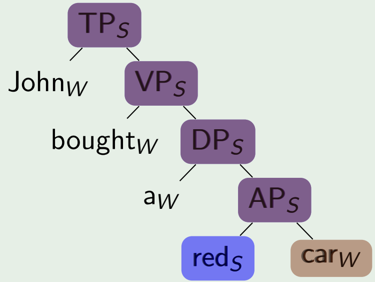
Example of ϕ

FocusPath and StressPath



Example of ϕ

FocusPath and StressPath



Formal Model: Putting it Together

- We already know that output joint preservation is satisfied.
- GEN is a linear transduction.
- C consists of only one constraint c, a linear transduction.
  - The tree models of the formula ϕ form a regular language $L(\phi)$.
  - The diagonal of $L(\phi)$ is a linear transduction (representing C).
- The composition of GEN with c is a linear transduction and yields the intended output language.
- This shows that Focus Economy is efficiently computable.

Conclusion

- Controlled OS were introduced as a model for reference-set constraints.
- Most requirements for a controlled OS to be efficiently computable are fulfilled by reference-set constraints; in particular, their corresponding OSs are globally optimal.
- The only problematic areas are $\textsc{Gen}$ and the OS constraints.
- The formalization of Focus Economy indicates that these do not pose an insurmountable challenge either.

References I

Aoun, Joseph, Lina Choueiri, and Norbert Hornstein. 2001. Resumption, movement and derivational economy. *Linguistic Inquiry* 32:371–403.

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.

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

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

References II

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.

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

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