

Reversing the perspective on Quantifier Raising

Valentina Bianchi & Cristiano Chesi, May 2010

The controversial rule of Quantifier Raising cannot be fully subsumed under the standard definition of the Move operation. We argue that the differences between overt movement and QR can be accounted for if the syntactic computation proceeds top-down (and left-right), rather than bottom-up.

In a bottom-up derivation, a phrase must be first merged in argument position, and the remerge position has different properties in the two cases: in particular, the target position of QR is covert, it is not featurally licensed, and it can not freely host a successive cyclic step (whence the clause-boundedness of QR).

If we assume instead a top-down + left-right derivation, a radical asymmetry emerges. Overtly moved phrases are first merged in a featurally licensed clausal position, and covertly remerged in argument position. In the case of QR, instead, the computation of the QP in argument position precedes the computation of the remerge position where the QP attaches to its nuclear scope, after the latter has been completely built: thus, QR is necessarily «rightward». The covertness of QR follows from the fact that the system invariably spells out the first merge position; the clause boundedness of QR is shown to be a right roof effect (due to a final phase boundary). The left-right orientation also immediately derives the Leftness Condition on quantifier binding. This rather conservative approach is briefly compared with the continuation-based analysis of quantification outlined in Barker (2002).

NB: This paper is part of a larger project on a top-down, phase-based minimalist grammar, which is described at: <http://www.ciscl.unisi.it/top-down/>

1. Introduction

The scope-assigning movement operation of Quantifier Raising (May 1977) is at the heart of the debate between different approaches to natural language semantics (for general discussion, see Jacobson 2002, Barker & Jacobson 2007). Even within the LF-based approach, QR is problematic, because in various respects, it does not fit the minimalist definition of the Move operation, and it is sensitive to interface economy (Fox 2000) in a peculiar way. Early minimalist attempts to reduce QR to independently motivated A-movement have been abandoned since Kennedy (1997), and the difficulty still persists.

In this paper we argue that the peculiar properties of QR can be derived on a principled basis in a phase-based minimalist grammar by simply reversing the direction of the computation, from a bottom-to-top to a top-down (left-to-right) orientation. Our argument will be based on the model first outlined in Chesi (2004), which we briefly summarize here.

The paper is organized as follows: in § 2, we list various properties that set QR apart from standard overt movement, and we briefly review various attempts to cope with them in the previous literature. In § 3, we outline the treatment of syntactic dependencies in a top-down

grammar by means of a general storage and retrieval mechanism. From a top-down perspective, the fundamental asymmetry between overt A'-movement and QR is that in the former, the computation of the (overt) A' position precedes the computation of the (re-merge) argument position of the dependency, whereas in the case of QR, the computation of the (overt) argument position must precede the computation of the (re-merge) scope position: put differently, in a top-down computation QR is necessarily «rightward». In § 4 we show how this different perspective can explain the fact that QR is necessarily covert, and the fact that it is clause-bound. In § 5 we extend our discussion to scope ambiguities and pronominal binding, subsuming the Leftness Condition under the general left-to-right architecture of the system. In § 6 we compare our proposal with other approaches, and in § 7 we offer some concluding remarks.

2. The initial assimilation, and its problems

May (1977) realized that the quantifying in rule could be recast as a movement transformation which happened to be invisible to the phonological component.¹

- (1) a. Mary see who → Who did Mary see t?
 b. Mary saw nobody → <Nobody> Mary saw t.

This proposal had dramatic consequences for the architecture of the grammar: the assumption of a level of representation (LF) distinct from surface structure as the input for semantic interpretation implied the abandonment of direct compositionality.

The empirical arguments that supported the reduction of scope assignment to syntactic movement have by now become a matter for textbooks. One well known argument claims that QR obeys the same locality constraints as overt movement, e.g. the adjunct island in (2) and the Coordinate Structure Constraint in (3).

- (2) a. A technician will complain [if you damage every plane]. (* $\forall > \exists$)
 b. LF: * <Every plane> a technician will complain [if you damage t]
 c. *What will a technician complain [if you damage t]?
 (Cecchetto 2004, (5)-(6))

- (3) a. A (#different) student [[likes every professor] and [hates the dean]]. (* $\forall > \exists$)
 b. LF: * <Every professor> a student [[likes t] and [hates the dean]].
 c. *Which professor does John [[like t] and [hate the dean]]?
 (Fox 2003, (12)-(13))

But this optimistic assimilation left various problems open (see Reinhart 1997 and Szabolsci 1997, 2008 for general discussion). Furthermore, with the advent of the minimalist framework,

¹ Following the current practice, unpronounced material is enclosed between angled brackets.

the status of QR as a movement operation became suspicious. With regard to the current definition of Move, QR remains exceptional in at least three important respects:

1. QR is not a feature-driven movement;
2. QR is not cyclic;
3. QR is covert.

2.1 QR is not feature-driven

First, Move (internal Merge/Remerge) is triggered by an Agree relation between a probe (an «attracting» head, to adopt an outdated but illuminating metaphor) and a goal (an «attracted» category), whereby the respective unvalued features of the probe and the goal get valued. This implies that Move is a feature-driven operation. QR, as a free adjunction operation with no designated landing site, is exceptional in this respect, as Hornstein pointed out. In his very early and thorough discussion of this problem, Hornstein attempted at reducing QR to independently motivated A-movement.² This move was criticized by Kennedy (1997) and starting from Fox (2000), the earlier view of QR prevailed again in the minimalist literature. Beghelli & Stowell (1997), on the other hand, proposed that QR is a feature driven movement, since every type of quantifier (e.g. Distributive Universal QPs, Counting QPs, Group Denoting QPs etc.) has its own landing site identified by a specific projection in the CP-IP functional fields. We will return to this proposal later.³

2.2 QR is not cyclic

Move has always been conceived of as successive cyclic, from the earliest formulation of the Subjacency condition up to Chomsky's (2001) Phase Impenetrability Condition: the persisting insight is that apparently long-distance movement is actually composed of a «chain» of local steps.⁴ In this respect, the behaviour of QR is peculiar: different classes of quantifiers can take scope over different syntactic domains, but none of them allows for scope extension in the way that is predicted by successive cyclic movement. The problem is extensively discussed in Reinhart (1997) and Szabolsci (2008), a.o.; here we mention two subcases. Universal/distributive quantifiers cannot extend their scope past a tensed clause boundary; this suggests that they cannot access the edge of a tensed clause, contrary to successive cyclic wh-movement.⁵

² A different line of attack can be found in Kayne (1998), which reduces covert movement to overt movement whose effect on linear order is «masked» by subsequent remnant movement.

³ In this discussion we will leave aside negative quantifiers and focussed phrases, since these elements are commonly assumed to be endowed with dedicated features ([neg] and [focus] respectively) and to target dedicated landing sites.

⁴ McCloskey (1979) was the first to provide empirical evidence in support of successive cyclic movement.

⁵ There are exceptions to this generalization, e.g. (i) (from Reinhart 1997, 336):

(i) A doctor will make sure that we give every new patient a tranquilizer. ($\sqrt{\forall} > \exists$)

Reinhart notices that not every *that*-clause allows for such scope extension. One possibly relevant factor is that in (i) the *that*-clause has a future-oriented reading. In French, this clause type also facilitates matrix scope for negative quantifiers (Kayne 1981):

- (4) a. A technician said [that John inspected every plane] (Cecchetto 2004, (11)-(12))
 b. LF: * <every plane> [a technician said [that John inspected t]] (* $\forall > \exists$)
 c. What did a technician say [that John inspected t]?

Indefinites, instead, can extend their scope past a tensed clause boundary, but they are apparently not sensitive to the island effects that constrain overt successive cyclic movement:

- (5) a. Most guests will be offended if we don't invite *some philosopher*. ($\sqrt{\text{some}} > \text{most}$)
 b. Everyone reported that [Max and *some lady*] disappeared. ($\sqrt{\text{some}} > \text{every}$)
 (Reinhart 1997, 339)

As for the second problem, Reinhart argued that QR is not involved in the exceptionally wide scope of indefinites, and suggested long-distance existential closure over choice function variables.⁶ A different solution was proposed in Schwarzschild (2002), who analysed such «wide scope» indefinites as existential quantifiers ranging over a singleton set: this makes the narrow scope reading truth-conditionally indistinguishable from the wide scope reading. Either solution will avoid the unwanted conclusion that in (5), QR can violate island constraints (contrary to (2)-(3)). Anyway, the first problem still remains: the clause-boundedness of universal/distributive scope in bridge (non-island) contexts like (4) cannot be explained if QR is successive cyclic.

A related property that sets QR apart from overt movement is the Scope Economy condition proposed by Fox (2000 and subsequent work) and adopted by Cecchetto (2004). According to this view, QR can be successive cyclic, but any cyclic step must be semantically motivated by the need to allow the moved QP to take scope over some scopally non-commutative element (in the sense of Fox 2000, 26), yielding different truth-conditions.

This interface economy condition is supported by the contrast in (6). Assume that VP ellipsis is subject to a parallelism requirement which, in particular, forces the object quantifier in the elliptical VP to take exactly the same scope as the parallel quantifier in the antecedent VP. In (6a), the object quantifier can undergo «long distance» QR by first adjoining to vP and then to IP: the second step complies with Scope Economy, because in both clauses, the subject is a scopally non-commutative element (an existential quantifier). In (6b), instead, «long distance» QR is impossible: this is because in the elliptical clause, the subject is a scopally inert element (a name), and hence adjunction of the object quantifier to IP violates Scope Economy; by the parallelism requirement, the object quantifier cannot undergo long distance QR in the first clause either.

- (6) a. A boy admires every teacher. A girl does, too <admire every teacher>. ($\sqrt{\forall} > \exists$)
 b. A boy admires every teacher. Mary does, too <admire every teacher>. (* $\forall > \exists$)

(ii) Je n'exige que tu arrêtes *personne*.
 I not-demand that you arrest.SUBJ nobody

Notice that, in all the examples, the subordinate verb is subjunctive. This could be a key factor w.r.t. the possibility of scope extension.

⁶ Reinhart also extends this approach to *wh-in-situ*. As a matter of fact, Huang (1982) was the first to notice that covert movement of *in situ* wh-phrases did not obey Subjacency.

The relevance of interface economy seems to correlate with the assumption that QR does not have a designated final landing site (contra Beghelli & Stowell 1997). On the contrary, overt movement does have a unique final landing site in the vicinity of the probe/attracting head, and interface economy is irrelevant (cf. Szabolsci 2008, § 10).⁷

2.3 QR is covert

The third difference between Movement and QR concerns the overt vs. covert nature of the operation. In wh-movement, the last occurrence of the moved element is spelled out: by the Extension Condition (or earlier, by the Proper Binding Condition), this occurrence is syntactically higher than the external merge position; hence, feature-driven movement is overt and generally leftward. In the T-model, the overt/covert opposition was determined by the stage of the derivation at which movement applied: only pre-S-structure movement had an effect on the phonological representation. However, the reason why quantifier movement applied post-S-structure remained elusive. In the minimalist framework, with a single cycle and a single level of representation, the covert nature of QR is even more puzzling. In the earlier minimalist framework, the driving force of overt movement was thought of as a strong/EPP feature, which had to be checked and eliminated before Spell-Out. It was soon clear that this solution could hardly go beyond this stipulative step, and strong features were abandoned. Given the copy theory of traces, the overt/covert opposition can now be reduced to the choice of spelling out the highest or the lowest link of a movement chain; but once again, it is unclear what determines either option.⁸

The problem is even more acute in Beghelli & Stowell's (1997) approach, in which QR is not a free adjunction operation, but targets specific functional positions in the clausal structure, dedicated to various classes of quantifiers. This approach can elegantly capture the non-uniform scopal behaviour of different quantifiers, and with minor changes, it even makes it possible to reduce QR to the standard probe-goal format. However, if QR is feature-driven, it is quite mysterious why it is covert (with the only possible exception of Hungarian, Szabolsci 1997,

⁷ Of course, *syntactic* economy conditions are relevant, cf. Relativized Minimality-Minimal Link Condition effects.

⁸ Yet another aspect of the problem, mentioned in von Stechow (2000, 219), is that QR, contrary to overt wh-movement, does not alter the c-command relations that are relevant for binding:

- (i) Which book that John_i read did he_j like t? (Obviation of Principle C)
- (ii) * He_i liked every book that John_i read. (No obviation of Principle C)
- (iii) John said that every picture of himself_i, Mary liked t. (Extended binding domain for Principle A)
- (iv) * John_i said that Mary liked every picture of himself_i. (No extension of binding domain for Principle A)

The contrast could be explained if in (ii) and (iv) QR is forced to target vP, since by Scope Economy it cannot target IP (the subject *Mary* is a scopally commutative element). Notice however that in (v), where the subject is scopally non-commutative (*some girl*), the object quantifier would be allowed to move past it by Scope Economy, yet the Principle A violation is not alleviated:

- (v) * John_i said that some girl liked every picture of himself_i.

A more systematic investigation is beyond our current purposes.

2008). Why would the quantificational features that trigger QR not be «strong» or EPP-like, as opposed to, e.g., the *wh*-feature?⁹

In sum, we can conclude that QR is not a well-behaved instance of Move. In the standard view, QR is not feature-driven, it cannot cyclically cross a CP boundary, and it never affects the phonological representation. According to Cecchetto (2004), QR complies with the Phase Impenetrability Condition and it can be successive cyclic, but it is not feature-driven, and the cyclic steps require an extra semantic motivation. According to Beghelli & Stowell (1997), QR is feature-driven and targets IP-internal positions: this may explain why QR cannot cross a CP phase boundary, but it is unclear why it is generally covert.

3. Reversing the perspective

The conclusion just reached seems discouraging. QR looks like an instance of Move, but it is something different, at least in part. It can be implemented as a movement operation, but only by bestowing a number of special properties, which cannot be derived in a principled way.

It is the last point that we will tackle in this paper. We squarely admit that QR, as a movement operation, differs systematically from overt movement: our aim is to try to explain this difference, while maintaining that one and the same fundamental mechanism is involved in both. In a nutshell, this is our main claim:

- (7) The specific properties of QR, as opposed to overt movement, follow if we assume a top-down, left-to-right computation divided in phases (Phillips 1996, 2003; Chesi 2004; Bianchi & Chesi 2006).

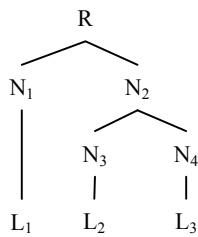
Before proceeding to the core of our proposal, it is necessary to summarize the relevant aspects of the top-down model on which our discussion will be based.

3.1. Sketch of a Top-Down minimalist grammar

In order to explore a tree structure, we need to define a direction both in the linear dimension and in the hierarchical one. As for the hierarchical dimension, there are in principle two main ways to traverse/build the tree: from the leaves to the top (i.e. bottom-up), or from the root to the leaves (i.e. top-down). To illustrate, consider the tree in (8):

⁹ In current phase theory, it is in principle possible to assume that QR targets the highest position in the phase edge, and the latter fails to be spelled out (Kayne 2005); the external merge position would have to be spelled out in order to comply with the recoverability condition. However, QR is commonly assumed to target the IP category, rather than the edge of CP (cf. e.g. Cecchetto 2004, 354). If QR could target the phase edge, we would expect it to be successive cyclic, contrary to fact (§ 2.2).

(8)



The bottom-up algorithm proceeds from L_{1-3} to R: lexical items are generated first (L_1, L_2, L_3), then non-terminal nodes are generated to collect the terminal nodes under the relevant constituent (N_1, N_3, N_4), and so on, up to the uppermost, unique, node which is the root one (R). In the other direction (top-down), the exploration proceeds from R to L_{1-3} : the root node (R) is generated first, then its children (N_1, N_2) are expanded in order to create the wanted subcomponents, in the end lexicalizing every terminal element (i.e. any node must terminate with a leaf/lexical item: in (8), N_1 terminates with L_1 , N_3 with L_2 , and N_4 with L_3).

These two strategies provide instructions to completely explore the tree in its hierarchical dimension. The instructions are however non-deterministic if we do not decide how to proceed also in the linear dimension: in order to create a complete tree-traversal procedure, we should also decide whether to proceed, ply by ply, from left to right (e.g. N_1 before N_2), or from right to left (e.g. N_2 before N_1).

Notice also that if we assume hybrid and/or parallel algorithms, the logical options are more than four (Top-Down + Left-Right, Top-Down + Right-Left, Bottom-Up + Left-Right, Bottom-Up + Right-Left). For instance, the standard «bottom-up» Minimalist procedure does not fully specify the order of inspection/generation of a subject constituent in a standard SVO language like English; there are two alternatives:

1. we stop building the main branch and start building the subject on a parallel «workspace» (Nunes and Uriagereka 2000); assuming that, in (8), N_1 is the subject, we create the nodes in this order: «main workspace» = $L_3, L_2, N_4, N_3, N_2, N_1, R$; «independent/parallel workspace» = L_1, N_1 ;
2. alternatively, we create the Subject node first, and then expand its children ($L_3, L_2, N_4, N_3, N_2, N_1, L_1, R$).

The first solution has attractive consequences. Nunes and Uriagereka (2000) analyze subject and adjunct islands as separate workspaces, which are spelled out independently of the main phase: in this way, under standard conditions, the computational system no longer has access to the sub-constituents of a subject or adjunct, since these are already spelt out when they are merged in the main workspace; this is why no item can be extracted from them. On the other hand, the second solution would determine a hybrid bottom-up / top-down strategy, and it would lose the account of subject and adjunct islands.

Thus, in the bottom-up strategy, the assumption of parallel workspaces seems preferable. Notice however that this assumption leads to a system which is not crash-proof; for instance, it is

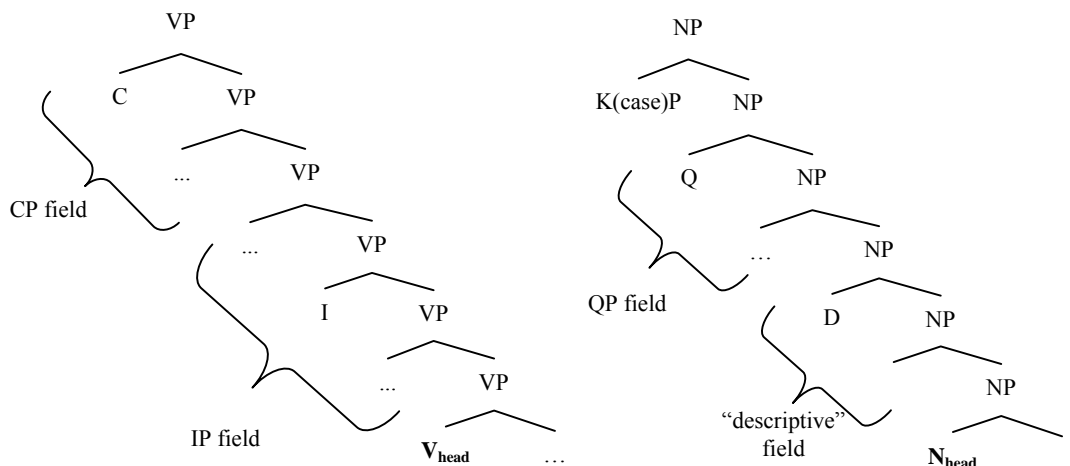
unclear how to guarantee tense agreement between the matrix clause and the adjunct clause if the two are generated in independent workspaces (9a,b):

- (9) a. He will arrive [before you leave].
 b. He arrived [before you left].

The option that we evaluate in this paper is a pure Top-Down + Left-Right strategy (for short, «Top-Down», since we are not aware of any Top-Down phrase building algorithm which is Right-Left oriented). Following this procedure, we generate the nodes in (8) in this order: R, N₁, N₂, L₁, N₃, N₄, L₂, L₃.

Because of complexity reasons (Chesi 2005) we adopt a version of the phase idea (Chomsky 1998-2005) and we chunk the derivation in pieces that roughly correspond to CPs and DPs.¹⁰ We assume moreover that these two kinds of phase have a unique head each, which is a main verb (V) for the CP phase, and a nominal (N) for the DP phase. Following the extended projection idea (Grimshaw 1991), functional specifications both in the VP and the NP domain are dominated by the phase head projection as depicted below:

(10)



This Top-Down + Left-Right orientation of the tree-traversal/building procedure is motivated by reasons of computational efficiency and cognitive plausibility (Chesi 2004, 70-81)¹¹ and is justified on empirical ground (it allows us to capture strong island effects, Chesi 2004, the connectedness condition, Bianchi & Chesi 2006, A-binding, Bianchi 2007, extraposition and Heavy NP-shift, Chesi 2009).

From a formal point of view, we obtain the directionality shift by defining:

¹⁰ We will not discuss here the possibility of having also AP and/or other phases.

¹¹ The grammar should have the property of flexibility: it should be directly usable both in a parsing and in a generation context.

1. Structural constraints (such as the constituent structures proposed in (10)), which are formulated in terms of universal principles.
2. Structure Building Operations (Merge, Move and Phase Projection) that operate Top-Down and from Left to Right, and are constrained by economy conditions.

The last fundamental component, crucial to fully specify our grammar, is the lexicon:

3. A list of language-specific feature structures, each one including semantic, syntactic and (possibly also) phonetic features.

In this sense, the lexicon is the locus of cross-linguistic parameterization, reducing language variation to differences in the featural inventory and/or feature clustering/combinations.

This «computationally minimalist» grammar can be explicitly described along the lines of Stabler's (1997) formalism.¹² The core idea of the structure building procedure is that any element be licensed within a Structural Description (SD) if and only if it is selected¹³ by a lexical head (a main Verb or a Nominal) or it is a possible functional specification of it. Accordingly, a lexical head is specified for two types of features:

- a. the SELECT features (prefixed with the equal sign: e.g. “=NP”) express its argumental valency, and license the head's arguments;¹⁴
- b. the LICENSOR features (prefixed with the plus sign: e.g. “+T”) instead encode the possible functional specifications that can be associated to the head: these correspond to the standard functional heads in the lexical head's extended projection. Importantly, the LICENSOR features associated to a given lexical head are limited in number and are linearly ordered, much as in the «cartographic» approach (Cinque 1999, Rizzi 1997 and related work).

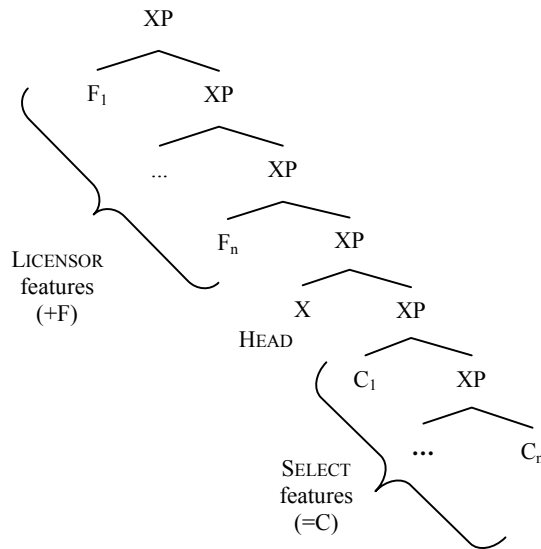
Assuming a top-down and left-right oriented algorithm, the functional features are processed before the phase head and the selected arguments after the licensing head; by expanding these features, we get the following prototypic phase structure:

¹² The result is the Phase-based Minimalist Grammar (PMG) discussed in Chesi (2008).

¹³ Here selection means both C(ategorial)-selection and S(emantic)-selection (Pesetsky 1982).

¹⁴ The select features correspond to the GB theta-grid, and to the SUBCAT (or SUBJ and COMPS) list in HPSG.

(11)



We fix the order of the functional and selected constituents w.r.t. the phase head by means of a Linearization Principle (inspired by Kayne's (1994) LCA):

(12) LINEARIZATION PRINCIPLE

- a. $\langle A, B \rangle$ if A (is a lexical head and) selects B as an argument
- b. $\langle B, A \rangle$ if B is a functional specification of A.

We emphasize here that any linear order imposed in this grammar (e.g. the order of the functional heads and their relative position w.r.t. heads and arguments) directly yields the processing order. The dynamic part of the construction of a SD is driven by three structure building operations that strictly operate Top-Down and Left-Right:¹⁵

1. MERGE is a binary function sensitive to temporal order, which takes two feature structures and unifies them (in the sense of unification grammars, Shieber 1983);
2. PHASE PROJECTION is triggered by the SELECT features of the currently processed lexical head and adds to the SD the minimal set of dominance relations to satisfy them;
3. MOVE is a top-down oriented function which stores an *unselected element* in a memory buffer¹⁶ and re-merges it at the point of the computation where the element is selected by a lexical head.

¹⁵ This general top-down algorithm can be exploited both in the generation task, converting a set of dominance relations among semantic/formal structures into a set of precedence relations among lexicalized phonological structures, and in the parsing task, converting a set of precedence relations among phonological structures into a set of dominance relations among lexicalized semantic/formal structures: it thus complies with the «flexibility» requirement (Chesi 2004; cf. note 11).

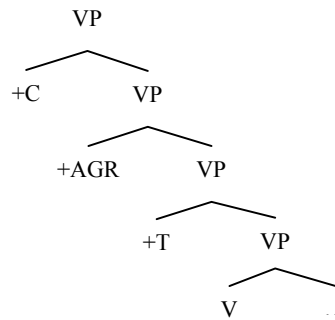
¹⁶ Minimality effects can be captured by assuming a Last In First Out memory, so that at a given point of the computation only the last element that was inserted in the buffer can be retrieved, and the previously inserted ones

An *unselected element* is any phase (e.g. [NP [+D the] [N boy]]) that is processed before the selecting lexical head (e.g. [+D NP =NP v kissed]), and hence temporally and linearly precedes the head itself, according to the Linearization Principle in (12).

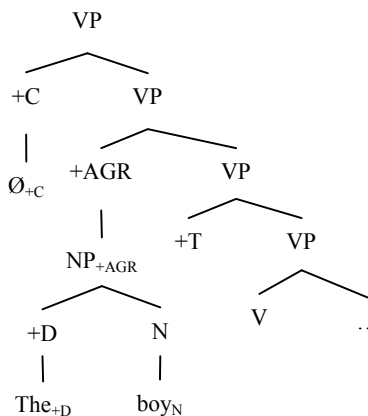
To illustrate the algorithm, consider the generation of the simple sentence in (13):

(13) The boy kissed the girl.

i. As the initial step, by Phase Projection the system projects a top-down expectation of a tensed verbal phase,¹⁷ whose lexical head will have to be a verb.



ii. Each functional projection will be expanded in sequence from left to right, and lexicalized by means of Merge; while an empty element (which should be present in the English lexicon) can satisfy the +C LICENSOR feature¹⁸, we need a full DP (i.e. an NP phase, with at least a +D LICENSOR feature, matching the projected +AGR features) to satisfy the [+AGR] LICENSOR feature; this can be satisfied by Merge of an empty [_{NP} +D N] constituent¹⁹, lexicalized next by merging some relevant item like [+D the] [N boy].



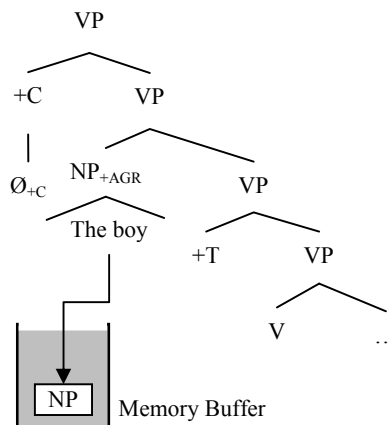
cannot. Given the selectivity of intervention (Relativized Minimality) effects (Rizzi 2001), the memory buffer must be multidimensional, i.e. different kinds of elements are stored in separate stacks.

¹⁷ This root application of Phase Projection is obviously not triggered by any SELECT feature.

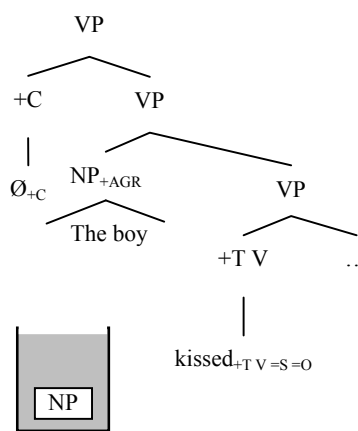
¹⁸ The unification algorithm implemented in Merge fully specifies this: a compatibility list among (clusters of) features.

¹⁹ As in (Non-Lexicalized) Tree Adjoining Grammar (Joshi et al 1975), we assume that non-lexicalized elementary trees are present in our lexicon: the unification function specifies that the licenser +AGR features and an NP constituent are compatible and can be merged only if they don't have any incompatible feature values.

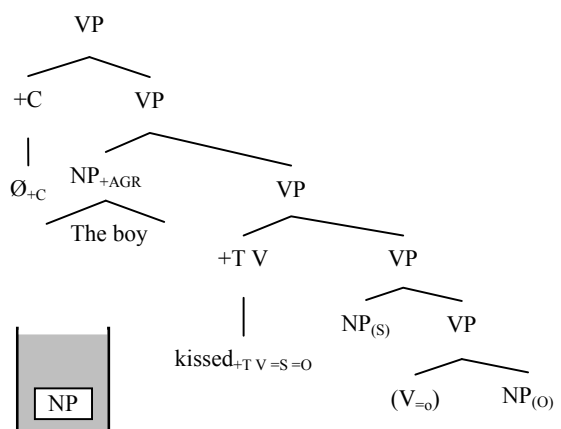
- iii. Since the NP phase just lexicalized is not selected in this position, it is also stored, by Move, in the memory buffer.



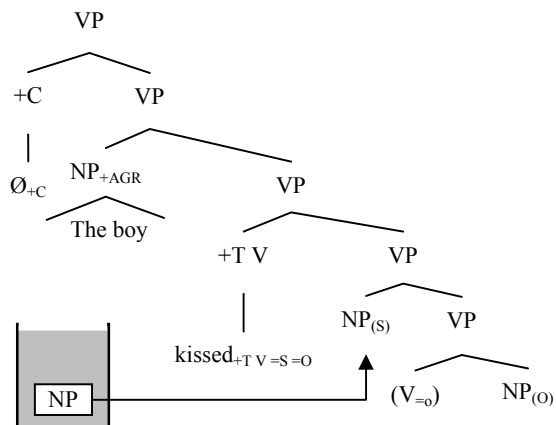
- iv. The lexical item *kissed* (analyzed as *kiss* + T) is processed; this introduces in the derivation the verb's SELECT features, here abbreviated as $\langle =S, =O \rangle$:



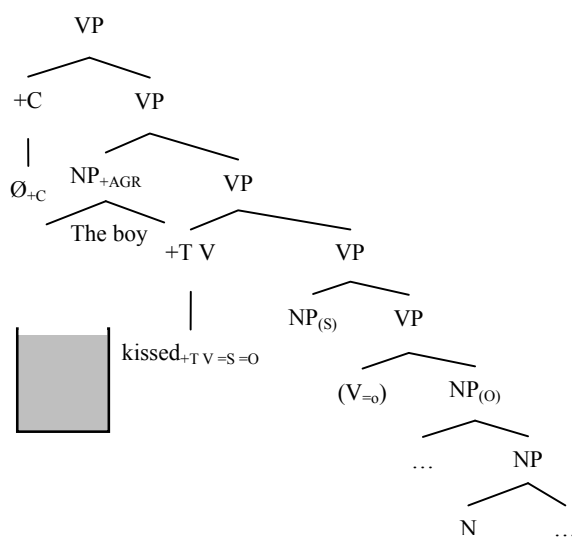
- v. The SELECT features in *kiss* $\langle =S, =O \rangle$ are expanded, in this order, by Phase Projection, and create two Nominal phases.



- vi. The constituent [_{NP} the boy] previously stored in the memory buffer is re-merged²⁰ in the (empty) nominal phase projected to satisfy the verb's =S feature.



- vii. As a final step, the skeleton structure created by Phase Projection to satisfy the =O feature will be processed as a new phase (the algorithm restarts from ii.).



We return immediately to the special status of the lowest selected complement and to the novel definition of «phase».

3.2. Phases

As mentioned, Chesi (2004, 2005) argues that in order to gain computational tractability, the computation must be broken up into phases, i.e. subparts of the computational process with a fixed upper bound in complexity:

- (14) A phase is the minimal part of a top-down computational process in which all the functional and selectional specifications associated to a given lexical head are satisfied.

Intuitively, each phase corresponds to the computation of a minimal chunk of syntactic structure like (11) above. Importantly, each phase will have a fixed upper bound in depth,

²⁰ In this system MOVE preempts MERGE; this seems to be empirically correct (Richards 1999).

determined by a limited number of possible functional specifications (Grimshaw 1991, Cinque 1999, 2002) and of selected arguments (Pesetsky 1982). Note however that, contrary to the standard bottom-to-top derivation, here a phase does not correspond to a complete subtree. In fact, when Phase Projection is triggered by the last SELECT feature of the lexical head, the phase gets closed, and the computation of the complement constitutes the next phase.²¹ Thus, a phase corresponds to a subtree whose lowest s-selected element is not yet expanded. As we mentioned before, for the sake of simplicity, we assume that only V and N can head a phase, and accordingly, phases correspond to the computation of standard CP or DP chunk.²²

The top-down perspective allows us to draw an important distinction between sequential and nested phases.²³ As we have just said, when a phase reaches the lowest position s-selected by the lexical head, it is closed off: the expansion of the complement constitutes the next sequential phase. A sequential phase thus follows the phase of the selecting head, and is separated from it.

On the other hand, any DP or CP within a phase P_n that does not occur in the lowest position s-selected by the lexical head of P_n constitutes a nested phase, which must be processed while P_n is still incomplete. Hence, all unselected DPs or CPs preceding the lexical head are necessarily nested phases: a preverbal subject, a fronted *wh*- or topical phrase can only be a nested phase. In (13), for instance, the subject [the boy] constitutes a nested phase within the matrix CP phase.²⁴

Since phases are the minimal chunks of the syntactic computation, it is natural to assume that each phase has its own «local» memory buffer for Move.²⁵ However, since long-distance movement can cross phase boundaries, it is necessary to devise a way to transmit the content of a phase's memory buffer to that of another phase. We propose the following Success Condition:

- (15) Success Condition: At the end of each phase the local buffer is empty, or else its content is inherited by the memory buffer of the next sequential (selected) phase.

Obviously, at the end of the last phase of all the local buffer will have to be empty, if not the elements in the memory buffer will not be selected in the structure and this would result in a violation of the principle of Full Interpretation (Chomsky 1986).

Crucially, this condition only allows for a communication between the memory buffers of two adjacent sequential phases. This accounts for the transparency of the lowest recursive branch of

²¹ Alternatively, the last SELECT feature can be satisfied by discharging an already processed constituent stored in the memory buffer by a previous application of Move.

²² From our perspective, vP is not a separate phase. For the sake of simplicity, we will retain the usual labels DP and CP for nominal and verbal phases.

²³ The distinction between sequential and nested phases is independently justified by their different effects on the computational complexity function (Chesi 2004, 2006), and it is also relevant to the analysis of left branch islands and connectedness effects (Bianchi & Chesi 2006).

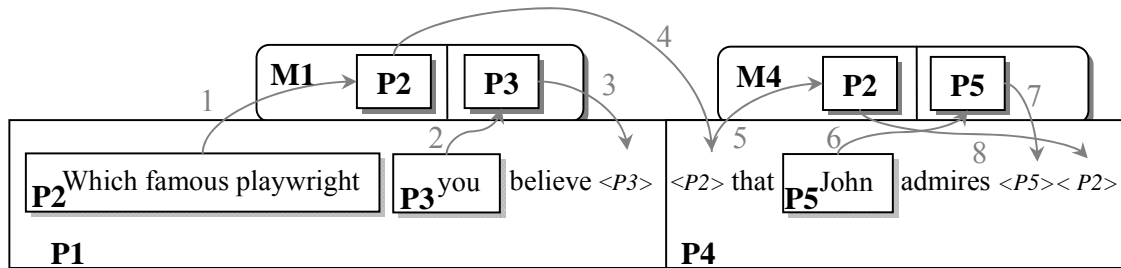
²⁴ Additionally, by (12), a non-selected nested phase is stored in the memory buffer of the matrix phase when its computation is completed.

²⁵ This is our way to reconstruct Chomsky's Phase Impenetrability Condition.

the tree. To see this, consider for instance a computation for (16), as schematically represented in (17) (where the boxes identify phase boundaries):²⁶

(16) Which famous playwright do you believe that John admires?

(17)



The algorithm initializes a CP phase 1 (P1). Then it computes the *wh*-phrase, which constitutes a separate nominal phase 2 (P2). Since the *wh*-phrase is not selected, it is stored in the local memory buffer of phase 1 (M1) by Move (step 1). The computation of phase 1 proceeds, by processing first the subject phase P3: this is again a nested phase, and it therefore introduced in the memory buffer M1 (step 2) and re-merged²⁷ (step 3) in the subject thematic position projected by the first SELECT feature of the main verb. The second SELECT feature of the main verb also triggers an application of Phase Projection for the object clause. At this point phase 1 is closed, and the *wh*-phrase in its memory buffer M1 is discharged into the local buffer of the complement CP phase 4 (P4), since the latter is sequential and selected. We propose that this takes place by re-merging the content of the memory buffer of Phase 1 in the left periphery of the complement CP phase 4 (step 4); since this position is unselected, by (12) the *wh*-phrase is re-stored (step 5) in the local memory buffer of phase 4 (M4). As a result, the inheritance mechanism leaves an intermediate copy/trace in the edge of the complement CP phase.²⁸ The computation proceeds (the embedded clause subject behaves as the main subject, steps 6 and 7) down to the object position of the verb *admires*, where the *wh*-phrase is discharged from the local memory buffer of phase 4 and re-merged (step 8). As the local memory buffer remains empty, the success condition (15) is satisfied at the end of the computation.

²⁶ Admittedly, the graphic representation (17) and the following ones are not very perspicuous, but this is the best possible representation we could figure out on a bi-dimensional sheet. Intuitively, the left-to-right orientation of the written line corresponds to the progress of the computation in time; the boxes represent phase boundaries; the arrows represent storage into/retrieval from a memory buffer.

²⁷ By a LIFO retrieval strategy, since the last inserted element is the most prominent one in the buffer (cf. note 16).

²⁸ Although this assumption is not strictly necessary for the algorithm to work, it seems fairly natural and it allows us to capture various successive cyclicity effects, like e.g. Irish complementizer alternations (McCloskey 1979, 1990) or French stylistic inversion (Kayne & Pollock 1978). We thank Luigi Rizzi for discussion of this point.

3.3. Summary

We summarize in four points the aspects of this top-down grammar that are immediately relevant for the following discussion:

1. Every computation is a top-down process divided in phases of fixed maximal size. A phase is the minimal part of a top-down computational process in which all the functional and selectional specifications associated to a given lexical head are satisfied.
2. A phase gets closed when the lowest selected position of its head is processed; the lowest selected complement constitutes the next sequential phase.
3. All unselected constituents are instead nested phases: they are processed while the superordinate phase has not been closed yet.
4. The Move operation stores an unselected element found before (i.e. to the left of) the head position in the local memory buffer of the current phase, and discharges it in a selected position if possible; if not, when the phase is closed the content of the memory buffer is inherited by that of the next sequential phase. The memory buffer of the last phase must be empty at the end of the computation.

4. Overt movement vs. QR

Going back to the main line of the argument, we saw that in the minimalist framework it is difficult to draw a principled distinction between overt and covert instances of movement. From the present top-down perspective, overt movement corresponds to first merge of the element in a peripheral, unselected position, which is licensed as a functional specification of the phase head.

On the other hand, the adjunction analysis of QR expresses the view that the quantifier / nuclear scope partition does *not* correspond to a functional specification of the clause (contra Beghelli & Stowell 1997): accordingly, the quantifier is first merged in a selected (argument) position.²⁹ How can the remerge scope position be structurally licensed, then?

We suggest that this is made possible by a mechanism of «inverse» selection. Generalized quantifiers are functions that take a nuclear scope as their argument and return a truth value (extensional type $\langle\langle e,t \rangle, t \rangle$): therefore, the quantifier is removed from the original argument position and is remerged in a scope position, from which it selects the remnant structure as its nuclear scope. This implies that the quantifier's SELECT feature temporarily remains unsatisfied in the first merge (argument) position: in other terms, QPs are phases which get closed with one SELECT feature still unsatisfied, for the obvious reason that this SELECT feature cannot be satisfied by projecting a sequential phase. We call such SELECT features, whose satisfaction must be

²⁹ Of course, quantifiers can appear overtly in a displaced position when they satisfy some independent LICENSOR feature, e.g. [topic]:

i. [Every one of these entries], we should thoroughly revise _.

delayed, INVERSE SELECT features: in this case the remerged element selects its sister, instead of satisfying a selectional requirement of its sister.

This hypothesis explains the fact that quantifiers are computed and linearized (i.e. first merged) in a non-displaced position. We will now argue that all the special properties of QR follow from the top-down architecture of the system.

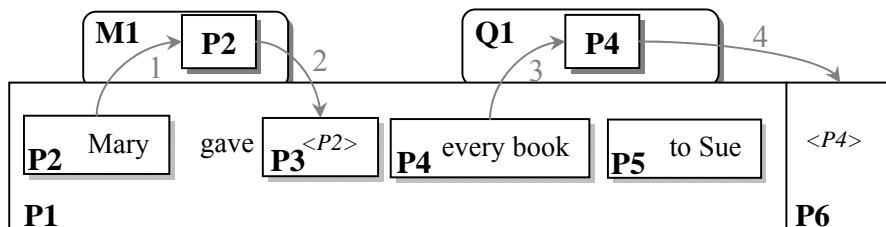
4.1. An implementation of Quantifier Raising

Here is a conservative top-down implementation of QR. The movement step (b) stores the QP in a specialized memory buffer (Q-buffer). In order to properly identify the variable to be abstracted over in the nuclear scope, the operation must also insert two occurrences of the same index: one attached to the stored QP, and one in the argument position.

- (18) Quantifier Raising
- i. Compute a QP and linearize it in a selected position³⁰ within phase n .
 - ii. Insert the QP in the Q-buffer of phase n with index i (QP _{i}).
 - iii. Insert a variable with index i in the selected position.³¹
 - iv. At the end of the computation of phase n , retrieve QP _{i} from the Q-buffer of n and re-Merge it with the structure built in phase n , satisfying the QPs' INVERSE SELECT feature.³²

(19) is a sample derivation of the sentence *Mary gave every book to Sue*.³³

(19)



After A-movement of the subject phase (P2, steps 1-2), the quantifier *every book* is computed and inserted in the selected direct object position; given its unsatisfied INVERSE SELECT feature, it is stored in the local Q-buffer of the matrix phase 1 (Q1, step 3), leaving a coindexed variable in

³⁰ Or in a functionally licensed position (see the preceding note).

³¹ This may be a plain indexed variable, or an indexed definite description à la Fox; the choice is orthogonal to our present concerns.

³² This implementation of Quantifier Raising is obviously reminiscent of Cooper storage: see § 6.1.

³³ For sake of simplicity the steps ii. and iii. in (18) are not represented in (19); accordingly, the index i on the QP is not indicated.

the selected position. After the computation of the second complement (P5)³⁴, the quantifier is retrieved from the Q-buffer and attached to the whole structure (step 4) so as to satisfy its INVERSE SELECT feature.

We can see that overt movement and QR exploit the same general storage and retrieval mechanism: the overt occurrence of an element is computed, assigning it an internal structure as well as a position in the clausal structure; the computed element is then stored in a memory buffer, and it is later retrieved and remerged at an appropriate point.

However, the top-down perspective reveals a fundamental asymmetry between the two cases. In the case of overt movement, the moved element is computed in a non-selected position satisfying a LICENSOR feature; as it also needs licensing by selection, it is stored in the local M-buffer, and it is remerged at the point where the computation of the selected position is reached. Given the top-down left-right orientation, the unselected position is computed *before* the selected position.

In the case of QR, instead, the quantifier is computed directly in the selected position, but it cannot be interpreted there, as it needs to combine with its nuclear scope; therefore, it is removed and stored in the Q-buffer, and it is remerged at a later point where an adequate nuclear scope has been built. Thus, the computation of the scope position *follows* the computation of the selected position: in other terms, QR is necessarily «rightward».

4.2. Covertness and rightward orientation

The covert nature of QR follows directly from a general property of the storage and retrieval mechanism: any element is linearized in its first merge position, whereas remerge positions are generally unpronounced (Chesi 2004, 148-151). In the case of *wh*-movement, remerge targets the selected position, which fails to be spelled out; in the case of QR, remerge targets an adjoined scope position, which also fails to be spelled out. The standard bottom-up architecture cannot express this difference, as the first merge position must be the selected one in both cases.

Interestingly, the rightward orientation of QR that is forced by the top-down computation has been independently stipulated by Fox & Nissenbaum (2000) and Fox (2002) in their analysis of adjunct extraposition:

(20) We saw a painting yesterday *by John*.

A crucial observation is that a QP from which adjunct extraposition takes place must have its scope as high as the extraposition site. For instance, in (21)a ‘free choice’ *any* is licensed in the scope of the intensional verb *look for*. In (21)b, relative clause extraposition interferes with the licensing of *any*, apparently forcing the direct object to take higher scope than the verb:

³⁴ Notice that the lowest complement (P5) becomes a nested phase by Inverse Selection here: the matrix phase P1 cannot be closed before the computation of P5, as its Q-buffer is not empty yet (by the Success Condition (14)). See § 6.1 for discussion.

- (21) a. I looked very intensely for [anything that would help me with my thesis].
 b. * I looked for [anything] very intensely [that would/will help me with my thesis].
 (Fox & Nissenbaum 1999, (5))

For this reason, Fox & Nissenbaum propose a late merge derivation of adjunct extraposition along the lines of (22). As a first step, the QP (without the adjunct) is merged in the direct object position, and the adverb *yesterday* attaches to VP (22)a. Then, the QP undergoes covert QR and is right-adjoined to the right of the adverb (22)b. Finally, the modifier *by John* is late merged with the unpronounced copy of the QP in the QR-ed position (22)c: this is why the adjunct's position marks the scope of the QP.

- (22) a. We [[saw *a painting*] yesterday]
 b. We [[[saw *a painting*] yesterday] <*a painting*>]
 c. We [[[saw *a painting*] yesterday] <*a painting*> by John]

Notice that two ingredients of the late merge analysis have to be stipulated by the authors: the fact that the QR-ed position is phonologically covert, and the fact that it is syntactically attached to the right. Both these properties follow from our approach to QR. This is, in our view, a suggestive result. A full discussion of the late merge analysis, however, is beyond the scope of this paper (see Chesi 2009).

4.3. Clause Boundedness

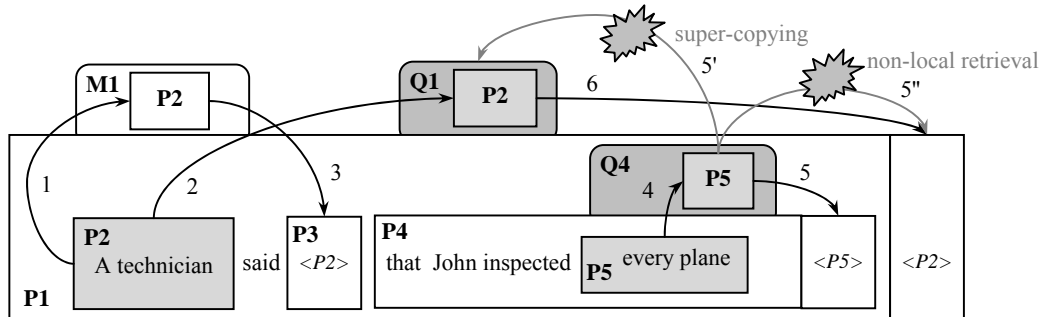
The most interesting consequence of our proposal concerns the clause-boundedness of QR. In a nutshell:

- (23) The clause-boundedness of QR is a *right roof effect*, corresponding to a final phase boundary.³⁵

Recall that a QP is stored in the local Q-buffer of the containing phase n , and by rightward attachment, it takes scope over any phase $n+i$ nested in n . On the other hand, the QP cannot take scope over any superordinate phase $n-i$, because this would require either (a) non-local retrieval of the QP from the Q-buffer of phase n , which is distinct from that of $n-i$, or (b) super-copying of the QP from the local Q-buffer of phase n into the Q-buffer of the superordinate $n-i$. Both these moves are disallowed: retrieval must always be local, and inheritance relations between the local phase buffers can only proceed «downward» to the next sequential phase (cf. (15)).

³⁵ See Chesi (2009) for general discussion of «rightward movement» and the Right Roof Constraint.

(24) *A technician* said [that John inspected *every plane*]. (* $\forall > \exists$)



Notice also that in the case of QR, the content of a Q-buffer cannot be discharged into that of the immediately following sequential phase: e.g., in (24) the matrix *QP a technician* cannot be discharged from the Q-buffer of phase 1 (Q1) into the Q-buffer of the subordinate clause (Q4). If this happened, then the *QP* would be locally retrieved and attached to the subordinate clause (phase P4), and it would fail to take scope over the matrix phase P1, which contains the variable to be bound (in the matrix clause subject position). In other terms, the semantic scope requirement³⁶ prevents downward inheritance. Therefore, in the case of QR the Success Condition can be simplified to (25):

(25) Success Condition for QR : at the end of any phase *n*, the Q-buffer of *n* must be empty.

In conclusion, we have seen that the top-down perspective allows us to derive the exceptional properties of QR, while maintaining that it is a syntactic operation exploiting the same storage and retrieval mechanism that implements overt movement.

We now turn to two crucial problems for any theory of QR: the treatment of scope ambiguities and of pronominal binding.

5. Further consequences

5.1. Scope reversals

One of the initial motivations for Quantifier Raising was the fact that it could easily derive scope reversals, i.e. scope relations that do not correspond to the surface relative prominence of the quantifiers. In (26), for instance, the direct object universal quantifier can take scope over the subject existential quantifier:

³⁶ That is, the requirement that the bound variable be in the semantic scope of the quantifier.

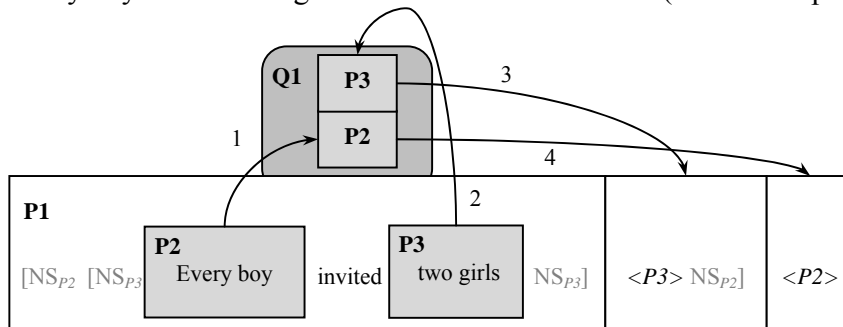
(26) A boy climbed every tree.

This can be easily obtained by allowing QR to attach the direct object quantifier to a position c-commanding the subject.³⁷

But scope reversals are not *that* easy, after all. A well known observation is that surface scope is consistently preferred by the speakers, and in certain languages, e.g. Chinese, surface scope seems to be quite rigid (cf. Huang 1982, Aoun & Li 1993).

In our approach, the preference for surface scope reading could be analyzed in terms of a Last-In-First-Out retrieval strategy, whereby the first stored element is remerged last and takes wider scope.³⁸ This is illustrated by the schematic derivation of (27) (we ignore for simplicity A-movement from the AGR-licensed subject position; NS_{P_n} stands for the Nuclear Scope of Phase n):

(27) Every boy invited two girls. (surface scope: $\forall > \exists$)



The subject quantifier is stored first in the Q-buffer; the object quantifier is stored on top of it, and hence it is retrieved first and attached to the structure. Then subject quantifier is then retrieved and attached to a higher position, taking wider scope.

Assuming a LIFO order of retrieval, of course, implies that scope reversal must come about through some extra mechanism. One possible solution is to allow for a rearrangement of the quantifiers stored in the Q-buffer: this will allow the QPs to be retrieved in a different order from the one symmetric to the order in which they were stored. The option of rearrangement could also be constrained by a condition of scopal non-commutativity of the QPs involved, thus recasting Fox's Scope Economy Condition (cf. the discussion around (6) in § 2 above).³⁹

³⁷ This long QR complies with Scope Economy, cf. the discussion in § 2 above. For general discussion, see Fox (2003).

³⁸ Cf. note 16 for another application of the LIFO strategy in the domain of Relativized Minimality effects.

³⁹ The idea of rearrangement in the Q-buffer would also allow us to implement the Condition on Scope Shift proposed by Neeleman & van de Koot (2009), according to which QR cannot move a QP across another QP that has already raised (though QR can move a QP across another QP that stays in situ; the authors crucially assume that QPs can be interpreted in situ). Assume a structure where, in their respective argument positions, QP1 c-commands QP2, which in turn c-commands QP3. The Condition on Scope Shift allows for the scope rearrangements in (b)-(d), which are derived by just one application of QR, and disallows those in (e)-(f), which require two applications of QR:

(a) QP1 > QP2 > QP3 (surface scope)
 (b) QP1 > QP3 > QP2 (movement of QP3 across QP2)

However, an unselective scope reversal mechanism rearranging the QPs in a Q-store is likely to overgenerate much as the standard QR. Importantly, Beghelli & Stowell (1997), May & Bale (2007) and Szabolsci (1997, 2008) argue at length that not every non-commutative pair of QPs allows for scope reversal:

- | | | |
|------|---|--|
| (28) | a. Everyone joined no committee. | (* scope reversal: no > \forall)
(May & Bale (2007), (5a)) |
| | b. Most of the girls admire every teacher. | (* scope reversal: \forall > most) |
| | c. A boy admires every teacher. | (\sqrt scope reversal: \forall > \exists) |
| (29) | a. More than one soprano sings in every show. | (\sqrt scope reversal: \forall > \exists) |
| | b. Every soprano sings in more than one show. | (*scope reversal: \exists > \forall)
(Szabolsci 2008, (73)-(74)) |

Real cases of scope reversal involve a universal/distributive quantifier taking scope over a syntactically higher weak (existential) quantifier, as in (28)c and (29)a. Beghelli & Stowell (1997) analyze this kind of scope reversal by assuming that the narrow scope weak quantifier moves to a functional projection ShareP which is in the immediate scope of the functional projection DistP targeted by the distributive quantifier. (On the other hand, wide scope existential QPs target a distinct projection RefP, syntactically higher than DistP.)

This analysis can be implemented in our approach by assuming a multi-layered Q-buffer,⁴⁰ with a separate stack for Beghelli & Stowell's «Share QPs», i.e. weak quantifiers that are distributed over: crucially, the «Share» stack can be accessed for retrieval only after one distributive QP has been retrieved from the general stack of the Q-buffer.

This second solution is empirically more restrictive, but contrary to the first one, it does not allow us to immediately capture Fox's Scope Economy Condition. Consider again the crucial example in (6)b above, repeated here:

- (6) b. A boy admires every teacher. Mary does, too <admire every teacher>. (* \forall > \exists)

Note that in both the antecedent and the elliptical clause, the universal quantifier is stored in the general stack of the Q-buffer, but only in the antecedent clause is an existential quantifier (the subject) stored in the «ShareP» stack; in the elliptical clause, instead, the universal quantifier cannot distribute over the subject denotation: the only possibility for it is to distribute over the existentially quantified eventuality expressed in the clause (Beghelli & Stowell 1997, § 3.2). It is

-
- (c) QP3 > QP1 > QP2 (QP3 across QP1 and QP2)
 (d) QP2 > QP1 > QP3 (QP2 across QP1)
 (e) * QP2 > QP3 > QP1 (QP3 across QP1 and QP2 across both)
 (f) * QP3 > QP2 > QP1 (QP2 across QP1 and QP3 across both)

In terms of our storage mechanism, the condition can be expressed by allowing at most one QP to be «reordered» in the Q-buffer.

⁴⁰ As mentioned in note 14, the Move-buffers must be multi-layered as well, given the selectivity of Relativized Minimality effects.

reasonable to assume that these different distributive configurations violate the parallelism requirement for ellipsis.

In sum, the selectivity of scope interactions seems to defy any unconstrained scope reversal mechanism. In the present storage-and-retrieval approach, this selectivity can be best captured by assuming multilayered storage buffers.

5.2. Pronominal binding and the Leftness Condition

Let us now consider quantifier-bound pronouns:

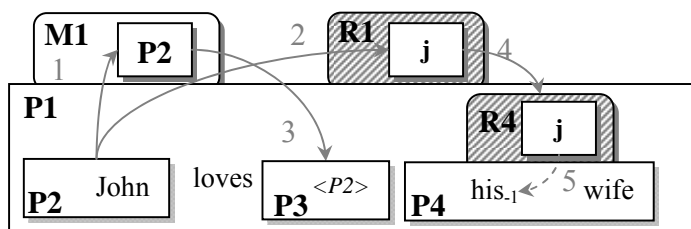
(30) Every man loves his wife.

Clearly, the system must be extended so as to capture pronominal binding.

Fortunately, the left-to-right interpretation for A-binding and Q-binding proposed by Schlenker (2005) is compatible with our approach, and it can be modified so as to incorporate the notion of phase proposed here (see Bianchi 2007 for details). Schlenker's approach shares with ours the use of an ordered memory register, which incrementally keeps track of noun phrase referents: the referents are introduced by R-expressions, and are retrieved by anaphoric pronouns. We will call this memory register (Schlenker's sequence of evaluation) the R(eferential)-buffer. This is distinct from both the M(ove)-buffer and the Q(uantifier)-buffer: in particular, the stored referents need not be discharged from the R-buffer by the end of the phase, as they are used for truth-conditional interpretation.

For reasons of space, here we will just illustrate the analysis of A-binding with a simple example (see Bianchi 2007 for detailed discussion):

(31) John loves his wife.



As shown in the figure above, A-binding of the pronoun *his* involves three steps:

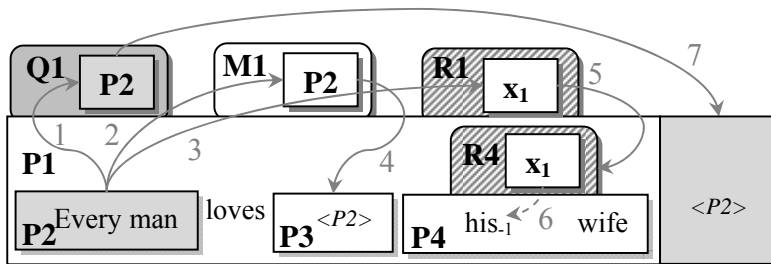
- i. when the subject R-expression is processed, its referent (j) is stored (step 2) in the R(eferential)-buffer R1 of the matrix phase 1 (step 2). (We disregard here A-movement of the subject, steps 1 and 3.)
- ii. The R-buffer of the direct object phase (P4) inherits the content of the R-buffer R1 of the containing phase (step 4).⁴¹

⁴¹ Both nested and selected phases (in the sense of § 3.2) inherit the R-buffer of the containing phase. This inheritance is not equivalent to discharge, in the sense of the Success Condition (14) for the M-buffers.

- iii. The bound pronoun *his* retrieves the referent j (via a negative index)⁴² from within the local R-buffer (step 5). The referent is then used to obtain the denotation of the direct object phase.

When the pronoun is Q-bound, its A-binder is the QP's trace (in our system, the indexed variable introduced in the argument position by the QR operation). Following Schlenker (2005), the variable bound by the QP is stored in the R-buffer and is then retrieved by the negative index of the anaphoric pronoun. This is illustrated by the derivation of (33), schematically represented in the following figure:

(32)



After the computation of the subject QP, QR stores the QP in the local Q-buffer Q1 (step 1), whereas the indexed variable inserted by QR is stored in the local R-buffer R1 (step 3). (As usual, steps 2 and 4 correspond to A-movement of the subject, which we can disregard.) Upon opening the direct object phase P4, the local R-buffer R4 inherits from the matrix R-buffer R1 the indexed variable: the latter is then retrieved by the pronoun in the usual way.

As Schlenker emphasizes, this mechanism can immediately derive the Leftness Condition on Q-binding:

(33) **His* wife loves *every* man.

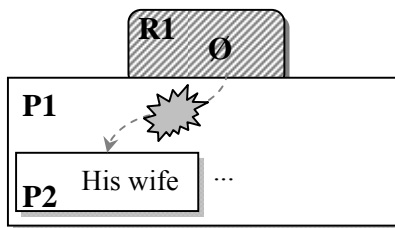
Starting from Reinhart (1983), the configuration in (33) (generally known as Weak Crossover configuration) has been analysed as a violation of a c-command requirement: the pronoun cannot be properly Q-bound because it is not c-commanded by the QP's trace. However, the c-command requirement appears to be too demanding in a number of configurations (see Hornstein 1995, Bianchi 2001 for discussion). An alternative to Reinhart's c-command requirement is the Leftness Condition, first formulated by Chomsky (1976), according to which a bound pronoun cannot linearly precede the binding quantifier.

The Leftness Condition never gained popularity, because it was a purely linear constraint and it disregarded hierarchical structure in a suspicious way. As independently argued by Schlenker

⁴² In Schlenker's system, the negative index indicates the position in the memory register where the referent is to be found. The element is then moved to the last position of the register, which is used for the computation of the denotation.

(2005) and Shan & Barker (2006), a left-to-right interpretation process derives the Leftness Condition for free. In our terms, the pronoun can retrieve the Q-bound variable from the R-buffer only after the QP has been processed and the variable has been inserted by the QR operation. Therefore, the processing of the QP must precede the processing of the bound pronoun.

As shown in the figure below, the attempted derivation of Q-binding in (33) fails because at the point of the derivation when the subject phase P2 is initialized, the R-buffer of the matrix phase is empty: therefore, the anaphoric pronoun *his* cannot retrieve from the local R-buffer a variable bound by the QP *every man*.



We consider this a major advantage of the left-to-right perspective.⁴³ Notice that in the «static» view, it was never really clear why a Q-bound pronoun had to be subject to a stricter requirement than merely that of being in the semantic scope of the binding quantifier. The stricter requirement (leftness) can be seen to follow from the temporal, incremental nature of the interpretation process.

6. Comparison with alternatives

6.1. Semantic vs. syntactic storage

The treatment of QR outlined in § 4.1 is obviously modeled on Cooper's (1983, chapter 3) Quantifier Storage: there too, the argument position is filled by an indexed variable, and the quantifier is stored *along with a binding index* (what Cooper calls a «binding operator»).

One obvious difference is that Cooper's system works bottom-up, so that the Quantifier store is inherited upward by the dominating nodes; the present system, instead, is top-down oriented: the Q-store is defined at the phase level, and its content is preserved as the computation proceeds downward, until the point when the QP can be remerged with its nuclear scope.

Another obvious difference is that Cooper storage is a purely semantic mechanism, whereas our QR is still a transformation of the syntactic structure.

This syntactic view of QR actually has one serious drawback for our system: it forces phase nesting. To see this, recall the distinction between sequential and nested phases (§ 3.2): a sequential phase constitutes the expansion of the lowest selected position of the lexical head of

⁴³ Shan & Barker (2006) conceive of the left-to-right orientation as a processing default. The view endorsed here is much stronger: the left-to-right orientation is intrinsic to the grammatical system.

the immediately preceding phase; all other phases which are unselected for, are computationally nested, in that their computation is included within the computation of the superordinate phase.

It is easy to see that when a superordinate phase contains a quantifier, the lowest selected complement of its lexical head must be nested, rather than sequential: the reason is that the Q-store of the superordinate phase cannot be emptied until the point where an appropriate nuclear scope has been built; however, the lowest selected complement must be included in the nuclear scope as well: thus, it has to remain «nested» in the superordinate phase. This complements our previous observation that the semantic scope of a quantifier cannot be inherited by a sequential phase (cf. § 4.3).

If QR were a semantic operation, like the original Cooper storage, the result would be semantic nesting. If, however, the combination of a stored QP with its nuclear scope is an instance of syntactic remerge, as assumed here, then we expect syntactic nesting as well.

As argued in Chesi (2005), nested phases induce an exponential increase of the complexity function for any long distance dependency that crosses the phase boundary: in essence, this is why nested phases are strong islands. However, notice that even if remerge of a QP turns the lowest complement into a nested phase, the latter does not constitute a syntactic island. In (34), the matrix quantifier *everyone* takes scope over the complement clause, yet the latter is transparent for extraction:

(34) the girl that everyone believes [that you are dating t]

It seems that the top-down computation of the Move-dependency relying on the M-buffers is insensitive to the fact that, at the point where the computation of the complement clause begins, the Q-buffer of the matrix clause still contains the quantifier *everyone*. This suggests that the management of the M-buffers does not «see» the content of the Q-buffers, so that the computation of the two types of dependencies is not parallel: the complement clause is sequential for the purposes of movement, but it is nested for the purposes of QR.

6.2. Storage vs. continuations

Before closing this discussion, it is important to take into account an alternative view of quantifier scope that is compatible with the general left-to-right orientation of our system, namely the continuation-based account of quantification proposed by Barker (2002, 2004 and subsequent work).

Barker starts by observing two quirks about quantification. First, quantifiers form a natural syntactic class with non-quantificational noun phrases, yet the semantic types of their denotations appear to be quite different. As is well known, Montague's PTQ achieved unification by assigning all noun phrases a denotation of type $\langle\langle e,t \rangle, t \rangle$, but this unification was stipulative, in that it merely constituted a generalization to the worst case.

The second quirk is the property of scope displacement, whereby the semantic scope of a quantifier does not seem to generally coincide with its syntactic scope: for instance, a quantifier

sitting in direct object position only c-commands its sister node V, and yet takes the entire clause in its semantic scope.⁴⁴

Barker's answer to both questions is based on the idea that quantification relies on the possibility for the quantifier to manipulate its own continuations. Summarizing very roughly, in a context like $[_B \dots A \dots]$ the *continuation* of the expression A (type α) relative to context B (type β), is a function that, when combined with the denotation of A, yields the denotation of the larger expression B (type $\langle \alpha, \beta \rangle$). Thus, the continuation of a noun phrase of type e relative to a sentence of type t is a function of type $\langle e, t \rangle$: e.g., the continuation of (the denotation of) *Mary* relative to the sentence *John saw Mary* is the function $\lambda x. \text{saw } x \text{ j}$.

It is then possible to define the *continuized denotation* of an expression as a function which takes as an input a continuation of that expression and yields an output of the appropriate type (in the relevant cases, of type t). Therefore, the continuized denotation of a noun phrases will be a function from continuations of type $\langle e, t \rangle$ to outputs of type t: this is exactly the type $\langle \langle e, t \rangle, t \rangle$ that Montague stipulated as the uniform type of NP denotations.

Since any expression has a continuized denotation that takes as an input a continuation of the expression itself, the semantic composition of continuized denotations is more complex than direct functional application. To illustrate, consider the derivation of a simple sentence like *everyone left*. The continuized denotation $\{\{QP\}\}$ of the QP *everyone* is the $\langle \langle e, t \rangle, t \rangle$ function $\lambda \bar{x} \forall x. \bar{x}(x)$, where \bar{x} is a variable over continuations of type $\langle e, t \rangle$. The intransitive VP [*left*] (direct type $\langle e, t \rangle$), in turn, has continuations of type $\langle \langle e, t \rangle, t \rangle$ and a continuized denotation $\{\{VP\}\}$ of type $\langle \langle \langle e, t \rangle, t \rangle, t \rangle$: $\lambda \bar{p}. \bar{p}(\text{left})$. The composition of QP and VP requires several instances of type lifting:

- (35) a. $\lambda \bar{p}. \{\{VP\}\}(\lambda P. \{\{QP\}\}(\lambda x. \bar{p}(Px)))$ (Barker 2002, (10a))
 b. $\lambda \bar{p}. \bar{p}(\lambda \bar{P}. \bar{P}(\text{left}))(\lambda P. (\lambda \bar{x} \forall x. \bar{x}(x))(\lambda x. \bar{p}(Px)))$
 $\lambda \bar{p}. (\lambda \bar{P}. \bar{P}(\text{left}))(\lambda P. \forall x. \bar{p}(Px))$
 $\lambda \bar{p}. \forall x. \bar{p}(\text{left } x)$

(The output is a function of type $\langle \langle t, t \rangle, t \rangle$, taking as input sentence continuations of type $\langle t, t \rangle$. This is applied to the trivial continuation, the identity function $\lambda p. p$, and yields the truth conditions $\forall x(\text{left } x)$.)

Crucially, this way of composing the continuized denotations ensures that whatever the syntactic position of the QP, the value returned by its composition with the sister node always contains the variables for all the direct types of the constituents involved (in the example above, P and x):⁴⁵ this implies that the QP always takes semantic scope over the whole clause, accounting for scope displacement.

⁴⁴ Notice that QR gives us a way to syntactically derive the appropriate Q-scope, but it does not account for the more fundamental question of why things are the way they are.

⁴⁵ Actually, a QP does not have a direct denotation of type e, but this is the type of the variable it binds.

Thus, the continuation-based analysis accounts for the fundamental «type mechanics» of quantification without relying on any special rule of covert displacement or storage. This result is impressive indeed and, as the analysis also allows for direct left-to-right interpretation, it seems to be preferable also from a general conceptual viewpoint. It goes without saying that this solution also wipes away the problems that we raised above about the covert and non-feature-driven nature of scope displacement *qua* QR.

Indeed, continuations represent “the entire (default) future for the computation”, and in this respect they intuitively correspond to a fundamental operation of the top-down syntactic algorithm, namely the projection of top-down expectations for structures that are yet to be built. We thus see a significant consonance between Barker’s proposal and the general top-down system outlined in Chesi (2004). In perspective, our present proposal concerning QR can be seen as a more conservative solution than Barker’s for the problem of quantification.

There are two remaining problems with respect to which Barker’s approach strikes us as not restrictive enough, at least at our present stage of understanding. The first one is the clause-boundedness of distributive quantifier scope, here analyzed (§ 4.3) as a consequence of a (rightward) phase boundary. Barker (2002, (20b)) shows that it is possible to implement clause boundedness in his continuized grammar of quantification by redefining the composition rule (35)a in such a way that the continuation variable \bar{p} fall outside the scope of the quantifier, as shown in (36): in this way, any element that is composed later will be outside the QP’s scope.

(35) a. $\lambda \bar{p}. \{\{\text{VP}\}\}(\lambda P. \{\{\text{QP}\}\}(\lambda x. \bar{p}(Px)))$

(36) $\lambda \bar{p}. \bar{p}\{\{\text{VP}\}\}(\lambda P. \{\{\text{QP}\}\}(\lambda x. (Px)))$

It is fair to say that, although this adjustment is perfectly licit, it does not seem to follow from anything in the system.

Furthermore, the solution Barker puts forth for scope ambiguities appears to be too unconstrained, much like free adjunction or free rearrangement in the Q-store (recall the discussion in § 5.1). Once again, the solution comes from a reformulation of the composition rule (35).a: by this rule, the (continuized denotation of) the VP applies to a continuation that contains the (continuized denotation of) the subject QP. As a result, the VP takes priority, and any quantificational element contained in the VP takes scope over the subject QP. However, it is perfectly possible to derive an alternative composition rule in which the subject takes priority over the VP:

(37) $\lambda \bar{p}. \{\{\text{QP}\}\}(\lambda x. \{\{\text{VP}\}\}(\lambda P. \bar{p}(Px)))$ (cf. Barker 2002, (18b))

As far as we can see, these rules are so general that they predict a complete freedom of scope reversal between the subject and the verbal complements: but as discussed above, this freedom is not empirically supported. At first blush, it is unclear how the two composition rules could be relativized so as to be triggered by specific subtypes of quantifiers (à la Beghelli & Stowell). We leave the question open for further investigation.

7. Concluding remarks

In this paper we argued that by reversing the orientation of the derivation from Bottom-to-Top to Top-Down + Left-Right, we can derive a principled explanation for some puzzling properties of QR, without giving up the assumption that QR is, in its core, a movement operation.

Both Move and QR are triggered by the satisfaction of some non-local requirement: in the case of standard Movement, the requirement of thematic selection; in the case of QR, the requirement for the quantifier to apply to an appropriate nuclear scope.

In the Movement case, the element to be moved is merged in the structure to satisfy a functional feature of the processed phase (e.g. a *wh*-element, the subject of the clause in its preverbal position etc.), but in this functional position its thematic role cannot be licensed (since thematic selection must be local w.r.t. the selecting head). The required long-distance dependency is realized by means of phase-local memory buffers which behave like a stack (the last element inserted is first retrieved and re-merged in the structure) and keep track of every unsatisfied selectional dependency. Memory buffers must be empty at the end of the computation; if not, the undischarged elements will be unselected and will violate the full interpretation hypothesis. With QR, the quantified phase is moved in a similar memory buffer that keeps the QP up to the point when the nuclear scope is built, and allow us to retrieve it and attach it to the phrase structure by means of an inverse selection mechanism.

This parallelism allows us to derive:

- i. the covertness of QR (the re-merge position is not spelled out) and its rightward orientation (Fox & Nissebaum);
- ii. the Right-roof constraint (i.e. clause boundedness): a Q-buffer is phase-local and cannot be inherited by any containing phase;
- iii. the Leftness Condition on Q-binding of pronouns.

Of course, this restatement of QR still implies a significant weakening of direct compositionality: interpretation must be delayed to the end of the phase. However, this solution maintains a clear parallelism between the syntactic and semantic computation: both proceed top-down and left-to-right, and both are divided in phases. The system thus captures the fundamental insight that the temporal dimension is not just an aspect of processing, but it is a constitutive constraint on the structure of natural language grammars.⁴⁶

References

- Aoun, J. & . Li. 1993. *The Syntax of Scope*. Cambridge, Mass., The MIT Press.
Barker, C. 2002. Continuations and the nature of quantification. *Natural Language Semantics*. 10:3, 211-242.

⁴⁶ This insight is at the core of Dynamic Syntax (Kempson et al. 2001 and related work) and it also constitutes the fundamental insight of Kayne (1994).

- Barker, C. 2004. Continuations in natural language. Fourth ACM-SIGPLAN Continuation Workshop, Venice, Italy.
- Barker, C. & P. Jacobson. 2007. *Direct Compositionality*. New York, Oxford University Press.
- Beghelli, F. & Stowell T. 1997. Distributivity and negation. In *Ways of scope taking*, ed. Anna Szabolcsi, 77-109. Dordrecht, Kluwer.
- Bianchi, V. 2001. Antisymmetry and the Leftness Condition: Leftness as anti-c-command. *Studia Linguistica* 55, 1-38.
- Bianchi, V. 2007. A note on backward anaphora. <http://www.ciscl.unisi.it/persona/bianchi.htm> (Revised version of the talk given at the XXX Glow Colloquium, Tromsø.)
- Bianchi, V. & Cristiano C. 2006. Phases, left branch islands, and computational nesting. *U.Penn Working Papers in Linguistics* 12.1, 15-28.
- Büring, D. 2004. Crossover situations. *Natural Language Semantics*. 12, 23-62.
- Cecchetto, C. 2004. Explaining the locality conditions of QR: Consequences for the theory of phases. *Natural Language Semantics* 12, 345-397.
- Chesi, C. 2004. *Phases and Cartography in Linguistic Computation*. Ph.D. Thesis. University of Siena.
- Chesi, C. 2005. *Phases and Complexity in Phrase Structure Building*. Selected Papers from The 15th Meeting of Computational Linguistics in the Netherlands.
- Chesi, C. 2009. *Rightward Movement from A Different Perspective*. Proceedings of the "Rightward Movement in a Comparative Perspective", Workshop at the DGfS Meeting, February 27-29, 2008, Bamberg.
- Chomsky, N. 1976. Conditions on the rules of grammar. *Linguistic Analysis* 2, 303-351.
- Chomsky, N. 1986. *Knowledge of Language: Its Nature, Origin and Use*. New York, Praeger.
- Chomsky, N. 2001. Derivation by phase. In M. Kenstowicz (ed.), *Ken Hale: A Life in Language*. Cambridge, Mass., The MIT Press.
- Cinque, G. 1999. *Adverbs and Functional Heads*. New York, Oxford University Press.
- Cinque, G. (ed.). 2002. *Functional Structure in DP and IP. The Cartography of Syntactic Structures, Vol. 1*. New York, Oxford University Press.
- Cooper, R. 1983. *Quantification and Syntactic Theory*. Dordrecht, Reidel.
- Fox, D. & Nissenbaum, J. 1999. Extraposition and scope: A case for overt QR. *Proceedings of WCCFL* 18, 132-144.
- Fox, D. 2000. *Economy and semantic interpretation*. Cambridge, Mass., The MIT Press.
- Fox, D. 2002. Antecedent Contained Deletion and the copy theory of movement. *Linguistic Inquiry* 33: 63-96.
- Fox, D. 2003. On Logical Form. In R. Hendrick (ed.), *Minimalist Syntax*. Oxford, Blackwell.
- Grimshaw J. 1991. Extended projection. In P. Coopmans, M. Everaert & J. Grimshaw ed. *Lexical specification and insertion*. The Hague, Holland Academic Graphics, 115-134.
- Hornstein, N. 1995. *Logical Form: From GB to Minimalism*. Oxford, Blackwell.
- Huang, 1982. *Logical Relations in Chinese and the Theory of Grammar*. PhD thesis, MIT.
- Jacobson, P. 2002. The (dis)organization of the grammar: 25 Years. *Linguistics and Philosophy* 25, 601-626.
- Joshi, A. K., L. S. Levy, and M. Takahashi. 1975. Tree adjunct grammars. *Journal Computer Systems Science*, 10(1).
- Kayne, R. & Pollock J.Y. 1978. Stylistic inversion, successive cyclicity, and Move NP in French. *Linguistic Inquiry* 9, 595-621.
- Kayne, R. 1981. ECP extensions. *Linguistic Inquiry* 12, 93-133.
- Kayne, R. 1994. *The Antisymmetry of Syntax*. Cambridge, Mass., The MIT Press.

- Kayne, R. 1998. Overt vs. covert movement. *Syntax* 1: 128-191.
- Kayne, R. 2005. *Movement and Silence*. New York, Oxford University Press.
- Kempson, R, W. Meyer-Viol, D. Gabbay. 2001. *Dynamic syntax: the flow of language understanding*. Oxford, UK, Blackwell Publisher.
- Kennedy, C. 1997. Antecedent contained deletion and the syntax of quantification. *Linguistic Inquiry* 28, 662-688.
- Longobardi, G. 1991. In Defense of the Correspondence Hypothesis: Island effects and parasitic constructions in LF. In: C.-T.J. Huang & R. May (eds.), *Logical structure and Linguistic Structure*. Dordrecht, Kluwer, 149-196.
- May, R. 1977. *The Grammar of Quantification*. PhD Thesis, MIT. (Printed by Garland, 1991).
- May, R. 1985. *Logical Form*. Cambridge, Mass., The MIT Press.
- May, R. & A. Bale. 2007. Inverse linking. In M. Everaert & H. van Riemsdijk (eds.), *Blackwell Companion to Syntax*. Oxford, Blackwell.
- Neeleman, A. & H. van de Koot. 2009. Scope inversion. Ms., University College London. (Available at: <http://ling.auf.net/lingBuzz/000962>).
- Nunes, J. & J. Uriagereka. 2000. Cyclicity and extraction domains. *Syntax* 3:1, 20–43.
- Pesetsky, D. 1982. *Paths and Categories*. Doctoral diss., MIT, Cambridge, Mass.
- Phillips, C. 1996. *Order and structure*. Ph.D. thesis, MIT.
- Phillips, C. 2003. Linear order and constituency. *Linguistic Inquiry*, 34:1, 37–90.
- Reinhart, T. 1983. *Anaphora and Semantic Interpretation*. Chicago, The University of Chicago Press.
- Reinhart, T. 1997. Quantifier scope: how labor is divided between QR and choice functions. *Linguistics and Philosophy* 20, 335-397.
- Richards, N. 1999. Dependency formation and directionality of tree construction. *MIT Working Papers in Linguistics* 34, 67-105.
- Rizzi, L. 1997. The fine structure of the left periphery. In L. Haegeman (ed.), *Elements of Grammar*. Dordrecht, Kluwer.
- Rizzi, L. 2002. *Locality and Left Periphery*. In Belletti, A. (ed). *Structures and Beyond. The Cartography of Syntactic Structures, vol. 3*. New York, Oxford University Press.
- Schlenker, P. 2005. Non-redundancy: towards a semantic reinterpretation of binding theory. *Natural Language Semantics* 13, 1-92.
- Schwarzschild, R. 2002. Singleton indefinites. *Journal of Semantics* 19, 289-314.
- Shan, C. & C. Barker. 2006. Explaining crossover and superiority as left-to-right evaluation. *Linguistics and Philosophy* 29, 91-134.
- Stabler, E. 1997. Derivational minimalism. In C. Retoré (ed.), *Logical Aspects of Computational Linguistics*. New York, Springer, 68-95.
- Szabolci, A. (ed.) 1997. *Ways of Scope Taking*. Dordrecht, Kluwer.
- Szabolci, A. 2008. Scope and binding. To appear in Maienborn, von Stechow & Portner (eds.), *Semantics: an International Handbook of Natural Language Meaning*. Berlin, Mouton de Gruyter.
- Von Stechow, A. 2000. Some remarks on choice functions and LF movement. In K. von Stechow & U. Egli (eds.), *Reference and Anaphoric Relations*. Dordrecht, Kluwer, 193-228.

Valentina Bianchi, bianchi10 AT unisi.it
 Cristiano Chesi, chesi AT media.unisi.it
 University of Siena, Italy
<http://www.ciscl.unisi.it/>