Deriving Reconstruction Asymmetries

Gregory M. Kobele

Humboldt Universität zu Berlin

Abstract

There appears to be a systematic difference in the reconstructability of noun phrases and predicates. In this paper I show that reconstructing the A/A-bar distinction in terms of slash-feature percolation and movement allows for a simple derivational formulation of the principles of binding and scope which derives a generalization very much along the lines of the one presented by Huang (1993).

1 Introduction

One of the important results of the study of syntactic dependencies is that different construction types (passive, raising, relative clause formation, wh-movement, topicalization, etc) have been revealed to systematically cohere in terms of the properties exhibited by their constitutive dependencies. Movement-type dependencies typically instantiate one of two attested dependency types, which are called A movement and A-bar movement. Much work has gone in to determining which properties movements of either of these two types have (see figure 1), with the ultimate goal being, of course, an explanation of why these facts obtain. Even in the absence of an explanation as to why movement dependencies divide in two, researchers can and have used the descriptive characteristics of these dependency types to classify other dependencies in other languages (such as scrambling (Mahajan, 1990; Müller and Sternefeld, 1993; Tada, 1993)).

1.1 Reconstruction

For a large number of cases, the simple principles of the binding theory and the basic scope principle (see figure 2) give correct results. There are, however, a wide variety of exceptions to these principles, as exemplified below.

	A	**A-bar**
Reconstruction is obligatory	no	yes
Licenses parasitic gaps	no	yes
Induces crossover effects	no	yes
Can escape tensed clauses	no	yes

Figure 1: Some A, A-bar Distinctions

(1) Principle A

1. Kwasi criticized himself.
2. Himself, Kwasi criticized.

(2) Principle B

1. *Kwasi$_i$ criticized him$_i$.
2. *Criticize him$_i$, Kwasi$_i$ did.

(3) Principle C

1. *He$_i$ criticized Kwasi$_i$.
2. *Criticize Kwasi$_i$, he$_i$ did.

(4) Scope

1. *He$_i$ criticized every boy$_i$.
2. *Every boy$_i$, he$_i$ criticized.

According to principle A, reflexive pronouns must be co-indexed with a c-commanding DP in the same tensed clause. Thus, we correctly predict sentence 1.1 to be well-formed. However, sentence 1.2 is mysterious, as here the reflexive c-commands its supposed antecedent, violating principle A, but the sentence is nonetheless well-formed. Principle B states that pronouns must not be coindexed with a c-commanding DP within the same clause. Thus,

Principle A

A reflexive must be c-commanded by a co-indexed expression within the same tensed clause.

Principle B

A pronoun must not be c-commanded by a co-indexed expression within the same clause.

Principle C

A proper noun must not be c-commanded by a co-indexed expression.

Scope

A quantifier can bind a pronoun only if it c-commands the pronoun.

Figure 2: Principles of Binding, and of Scope

sentence 2.1 is correctly predicted to be ill-formed. Sentence 2.2 is also ill-formed, despite the fact that the pronoun and the DP it is co-indexed with are mutually independent with respect to the c-command relation, and thus do not violate principle B. The ill-formed sentence 3.1 has a proper noun c-commanded by a co-indexed DP (*he*), thus violating principle C. Principle C is not violated in the nevertheless ill-formed 3.2, as the proper noun and the pronoun are again mutually independent with respect to the c-command relation. Finally, it is correctly predicted that the pronoun in 4.1 cannot be bound by the non-c-commanding quantified noun phrase *every boy.* However, in 4.2, the quantified noun phrase does c-command the pronoun, but is still unable to bind it.

In each of the above cases, the second sentence of the pair can be subsumed under the binding theoretic or scope principles if these are taken not to apply to the surface structure representations of the mysteriously ill-/well-formed sentences, but rather to a representation in which the topicalized constituent is 'put back' in its pre-movement position (yielding sentences identical to the first member of each of the above pairs) (Chomsky, 1976). The phenomenon above (that the second of each of the above pairs can be dealt with by a slightly more abstract version of the binding/scope theory) is called 'reconstruction', after the literal reconstruction process proposed to account for these facts by Chomsky (1976).

1.2 The Predicate/Argument Asymmetry

Huang (1993) notes that whereas moved DPs seem to be able to reconstruct in non-first-merged positions (as in sentence 5), moved predicates (as in 6) are much more restricted in their reconstruction possibilities.

(5) Which portrait of himself$_{i/j}$ does Kwasi$_i$ believe that Diego$_j$ criticized?

(6) Criticize himself$_{*i/j}$ Kwasi$_i$ believes that Diego$_j$ did.

In sentence 5, there are two different potential antecedents for the anaphor – it is semantically ambiguous. In other words, *which portrait of himself* can be reconstructed in either its θ position (sister to V), or in the intermediate SPEC-CP position. The fronted VP in sentence 6 on the other hand, can be reconstructed only in its deep structure position, as complement to Infl, as evidenced by the fact that only the lower subject (*Diego*) is acceptable as an antecedent for the anaphor.

Huang proposes to tie this contrast to the presence of an unbound trace in the fronted constituent. According to Huang (who adopts the VP-internal subject hypothesis (Koopman and Sportiche, 1991)), the structure of 6 above is as in 7 below.

(7) [t_i criticize himself]$_j$ Kwasi believes that Diego$_i$ did t_j

Huang assumes that anaphors must be bound by a binder within the minimal node containing the expression of which the anaphor is an argument, along with all other arguments of that expression. In the case of verbs, this is the vP where all arguments of the verb have been base-generated. As in 7 the fronted element contains all arguments of the verb, the anaphor must be bound by one of them (in this case, the trace t_i), regardless of whether or where the phrase is 'reconstructed.'

1.3 A and A-bar Movement in the Minimalist Program

Some of the properties distinguishing movement types, such as the inability of A movement to license parasitic gaps, or to escape from tensed clauses, can be accounted for in purely configurational terms (i.e., without recourse to the distinction between movement types) (Nunes, 2001; Kobele, 2006, 2008), and thus are better viewed as an *accidental*, rather than as an *essential* characteristic of movement types. Attempts have been made to deal with the more semantic differences configurationally as well (Sportiche, 2003), but these lead to nonstandard analyses of even the most familiar phenomena.

Perhaps the most influential account of the A/A-bar distinction in the minimalist program is propounded by Lasnik (1999) (see also (Chomsky, 1995; Fox, 2000; Boeckx, 2000)), who suggests that we attribute the different properties of these two movement types to whether or not a movement step leaves behind a copy of the moved item, or a simple trace (or nothing). His goal is to account for the observation that reconstruction into an A position is not obligatory, whereas reconstruction into an A-bar position is.

His analysis of the difference in behaviour between the two movement types is conceptually neat, as it traces the difference to a natural theoretical distinction (the difference between a full copy and an unstructured trace). However, it necessarily leaves some things to stipulation. First, no explanation seems readily available for the ban on improper movement; if movement can 'decide' to leave behind either a trace or a copy, why does the decision of a previous movement step influence the decision of the next? Second, why are the operations of grammar limited to copying and deletion/trace insertion? In Lasnik's story, this isn't derivable from anything else – it must simply be stipulated. Finally, Lasnik's idea seems committed to an 'LF' perspective on interpretation, and incompatible with a compositional, Montagovian one.

Lasnik's theory can be thought of as embodying the intuition that expressions only 'count as being there' in their A-bar positions (where they survive as copies), not in their A positions (where they are traces). Manzini and Roussou (2000) take this intuition literally, and reformulate Lasnik's ideas derivationally. They observe that the ban on improper movement structures chains in such a manner as to require that as soon as an expression counts as being there, it must be there in all of its higher chain positions as well (figure 3). This makes possible a simple 'timing' account of the A/A-bar

$$\underbrace{c_n\ c_{n-1}\ \ldots\ c_i}_{\textit{there}}\ \overbrace{c_{i-1}\ c_{i-2}\ \ldots\ c_1}^{\textit{not there}}$$

Figure 3: Chains, as per the ban on improper movement

distinction, if we make the crucial assumption that an expression may begin satisfying dependencies before it is merged. Kobele (2007) shows how this can be done. He formalizes Manzini and Roussou's 'feature attraction' operation using slash-feature percolation (as in generalized phrase-structure grammar (Gazdar et al., 1985)), and shows how slash-features and movement can naturally co-exist in the minimalist grammar formalism (Stabler, 1997), giving rise without additional stipulation to the ban on improper movement.

The remainder of this paper is structured as follows. First, we introduce minimalist grammars with slash-feature percolation. Next, we show how the particular properties of slash-feature percolation *force* 'moved' predicates to be actually moved, and not introduced via slash-feature percolation. This fact, in conjunction with a natural theory of reconstruction (presented in the section thereafter), derives Huang's generalization.

2 Slash Features and Movement

Slash-feature percolation and movement are two ways of establishing long-distance dependencies between tree positions. Though they appear to be 'inverses' of each other, they have quite different formal properties: minimalist grammars with just slash-feature percolation are context-free, whereas minimalist grammars with just movement are mildly context-sensitive. As we will soon see, the reason for this difference lies in the fact that proper binding condition violating remnant movement is possible only with movement, and is not simulable with slash-features.

2.1 Features

As the generating functions **merge** and **move** are taken to be universal and invariant, differences between languages reside in the lexicon. Lexical items have various features, which determine how they behave in the derivation. In addition to movement being feature driven, I will assume that merger is as well, and that, moreover, the kinds of features which are relevant for the **merge** and the **move** operations are distinct, and will be called **selection** and **licensing** features, respectively. Each feature has an **attractor** and an **attractee** variant (figure 4), and these must match in order for an operation to apply. Each time an operation is applied, it checks both an attractor and

	attractor	attractee
merge	=x	x
move	+y	-y

Figure 4: Features

an attractee feature, of the appropriate variety.

We will also not attempt here to model inflection or agreement, nor to make a link between morphological features and the syntactic features of expressions which we take here to drive derivations. Thus, we make no distinction between interpretable and uninterpretable features – here all features behave as though uninterpretable.

2.2 Lexical Items

Syntax relates form and meaning. Lexical items are the building blocks of this relation, atomic pairings of form and meaning, along with the syntactic information necessary to specify the distribution of these elements in more complex expressions. Here, simplifying somewhat, we take lexical items to be pairings of abstract lexemes such as dog, cat, bank_1,... with feature bundles. Feature bundles are taken to be ordered (Stabler, 1997), so that some features can be available for checking only after others have been checked. We will represent feature bundles as lists, and the currently accessible feature is at the beginning (leftmost) position of the list. An example lexical item is shown in figure 5. Its feature bundle '=d V' indicates that it first selects a DP argument, and then can be selected for as a VP. Treating feature bundles

$$\langle \mathsf{praise}, \texttt{=d V} \rangle$$

Figure 5: A lexical entry for *praise*

as ordered is convenient, as it allows us to take explicit syntactic control of which arguments get selected when. Certain lexical items, such as the Saxon genitive 's, are naturally thought of as selecting two syntactic arguments, an NP complement and a DP specifier. As the notions of 'complement' and of 'specifier' reduce to 'first-merged' and 'not-first-merged' in the context of the minimalist program, we need a way of ensuring that the first merged argument of 's is the NP, and not the DP. Here we simply structure the feature bundle for 's so as to have the noun phrase selected before the determiner phrase (the lexical entry might have the following form $\langle$'s, =n =D D$\rangle$). One question which arises at this point is whether this ordering must be taken as a primitive, or whether it is derivable from some deeper property (or a conspiracy of such) of grammar.[1] A second question is whether a substantive theory of feature bundles can be developed, to which the here hypothesized

[1]The ordering between selection features (=x) might seem relatable to the semantic type of an expression, under natural assumptions about the syntax-semantics interface. However, with no obvious reason to prefer a function f of two arguments to its permuted counterpart f', where $f'(a)(b) = f(b)(a)$, this 'reduction' would seem to put the cart before the horse. An interesting approach is suggested by Müller (2008), where the licensor and

linear ordering is perhaps only a rough approximation.[2] These questions are non-trivial, and cannot be pursued further here.

2.3 Syntactic Objects

We write lexical items using the notation $\langle\alpha, \delta\rangle$, where α is a lexeme (such as praise), and δ is a feature bundle (such as '=d V'). Complex expressions are written using a labeled bracket notation, as per the following:

$$[_{\delta}\ \alpha\ \beta]$$

The above represents an expression whose head has feature bundle δ, and consists of the two immediate sub-expressions α and β. As an example, if we assign the lexical entries $\langle$'s, =n =D D$\rangle$ and $\langle$brother, n$\rangle$ to the Saxon genitive and to brother respectively, then the complex expression 's brother, which is the result of merging brother as the complement of 's, is represented as the below.

$$[_{\texttt{=D D}}\ \text{'s [brother]}]$$

As can be seen, the above expression has feature bundle '=D D', which means after it merges with an expression of category D (such as $\langle$Kwasi, D$\rangle$), it will itself be of category D.

$$[_{\texttt{D}}\ \text{Kwasi ['s [brother]]}]$$

selector features of an expression are ordered only with respect to other features of the same type (and so licensor features are ordered with respect to other licensor features, but not with respect to selector features). He proposes a general principle which forces licensor features to be checked as soon as possible.

[2]Note first that, regardless of the shape of a feature bundle, a derivation imposes a linear order on the features in each of the feature bundles of the items occurring in it – this order is simply the order in which those features were checked in that particular derivation. (Or, if we allow multiple features to be checked *en masse*, the derivation imposes an equivalence relation over features in feature bundles, and then imposes a total order over equivalence classes of features.) A feature bundle, then, stands proxy for a set of totally ordered feature bundles (or of totally ordered feature equivalence class bundles); these are given by those total orderings which can be imposed upon it by some derivation in which it can occur. (In the system here, each feature bundle, being already totally ordered, stands for the unit set containing just it.) A natural desideratum for a lexicalized grammar formalism is that it support a type system rich enough to be able to completely account for the distribution of each expression with a single category (feature bundle). Thus, one goal of a substantive theory of feature bundles is to be able to derive all and only the necessary linearly ordered feature bundles which describe the distribution of a lexical item from a single feature bundle. Another goal is to make those feature bundles which describe the distribution of linguistically possible lexical items less marked (i.e. simpler) than those which do not. This 'markedness' can take the form either of simply ruling out linguistically impossible feature bundles, or of abbreviatory conventions (à la Chomsky (1965)) which make linguistically natural feature bundles more notationally natural.

Note that the features (`D`) of the head (`'s`) of the above expression are only represented *once*, and on the most maximal projection of that head. When a pair of brackets no longer has any unchecked features (as is the case with [brother] and with ['s [brother]] above), we no longer need to make these constituency distinctions (without features no constituent can move), and sometimes for convenience will leave those brackets out. The above expression would under this convention be rendered as per the following.

$$[_{\texttt{D}} \text{ Kwasi 's brother}]$$

Moving expressions are simply sub-expressions which have not checked all of their features. For example, the expression below is a sentence (IP) which contains a *wh*-phrase which has not yet checked its `-wh` feature.

$$[_{\texttt{i}} \text{ Kwasi [will [praise } [_{\texttt{-wh}} \text{ who]]]]}$$

We will apply our convention on leaving out brackets here as well, eliminating those brackets around constituents whose heads have no unchecked features. The expression above would be given as the below.

$$[_{\texttt{i}} \text{ Kwasi will praise } [_{\texttt{-wh}} \text{ who]]}$$

2.4 Merge

The **merge** operation applies to two arguments, A and B, resulting in the new object $A+B$, just in case the head of A has some selector feature `=x` as the first unchecked feature in its feature bundle, and the head of B has the corresponding `x` as the first unchecked feature in its feature bundle. In the resulting $A+B$, both first features used in this derivational step are checked, making available the next features of both feature bundles. There are two cases of the **merge** operation, depending on whether B will surface as a specifier or as a complement of A (figure 6). The first case of **merge** is the

$$\mathbf{merge}(A,B) = \begin{cases} [_A\ A\ B] & B \text{ is a complement} \\ [_A\ B\ A] & B \text{ is a specifier} \end{cases}$$

Figure 6: Cases of **merge**

merger of a complement, or 'first-merge'. Let us represent the active voice head in English (little-v) with the following lexical item: $\langle\texttt{-}\epsilon, \texttt{=V +k =d v}\rangle$.[3]

[3]The symbol ϵ is the empty string. The hyphen in front of it indicates that it is a suffix, and thus triggers head movement; its complement's head raises to it.

Because the first feature in the feature bundle of this lexical item is `=V`, it can be merged with the expression below (the VP *praise Kwasi*).

$$[_{\texttt{V}}\ \textsf{praise}\ [_{\texttt{-k}}\ \textsf{Kwasi}]]$$

Note that the selecting lexical item is a suffix (marked by the hyphen preceding the lexeme). This triggers head movement from its complement.[4]

$$\textbf{merge}(\langle\text{-}\epsilon, \texttt{=V +k =d v}\rangle, [_{\texttt{V}}\ \textsf{praise}\ [_{\texttt{-k}}\ \textsf{Kwasi}]]$$
$$= \Big[_{\texttt{+k =d v}}\ \textsf{praise-}\epsilon\ \Big[\ t_{praise}\ [_{\texttt{-k}}\ \textsf{Kwasi}]\Big]\Big]$$

head movement

Leaving out the traces, phonetically empty elements, and unnecessary internal structure, this expression can be abbreviated as the below.

$$[_{\texttt{+k =d v}}\ \textsf{praise}\ [_{\texttt{-k}}\ \textsf{Kwasi}]]$$

Note again that both of the matching first features of the arguments to **merge** (`=V` and `V`) have been checked in (i.e. deleted from) the result.

The second case of **merge** is merger of a specifier. This happens whenever the first argument to **merge** is not a lexical item (note that ‘first-merge’ happens when the first argument *is* a lexical item). As can be seen in figure 6, the main difference between first and later merges lies in the positioning of the merged item with respect to the head (first merged expressions come after, later merged expressions before, the head). Thus, we are essentially computing the effects of Kayne’s (1994) Linear Correspondence Axiom (LCA) incrementally during the derivation of an expression. Readers who are uncomfortable with this can view it equivalently as a convenient shorthand for first building an unordered tree, and then applying the best current linearization algorithm to this unordered tree.

2.5 Move

The **move** operation applies to a single syntactic object A just in case it contains a subexpression expression B with first unchecked feature `-y`, and the first unchecked feature of A is `+y`. In order to rule out nondeterminacy, **move** will only be defined if there is *exactly one* such subexpression beginning with a matching `-y` feature.[5] Just as with the **merge** operation, **move** checks

[4]See Stabler (2001) for more details. The basic idea is that head movement is not syntactic (i.e. feature-driven) movement (see Matushansky (2006)).

[5]This is a radical version of the shortest move constraint (Chomsky, 1995), and will be called the SMC – it requires that an expression move to the first possible landing

the matching features of the expression to which it applies. As an example, consider the expression below (which is similar to *praise Kwasi* derived above, but with who having been merged instead of Kwasi).

$$[_{\texttt{+k =d v}}\ \text{praise-}\epsilon\ [_{\texttt{-k -wh}}\ \text{who}]]$$

The **move** operation applies to this expression, as the main expression has as its first feature a licensor `+k`, and there is exactly one subexpression with first feature the matching licensee `-k`.

$$\mathbf{move}([_{\texttt{+k =d v}}\ \text{praise-}\epsilon\ [_{\texttt{-k -wh}}\ \text{who}]]) = [_{\texttt{=d v}}\ [_{\texttt{-wh}}\ \text{who}]\ \ [\text{praise-}\epsilon\ t_{who}]]$$

Again, leaving out traces and internal structure, we write the expression above as per the below. Note again that both features involved in the **move** operation (`+k` and `-k`) are deleted/checked in the result.

$$[_{\texttt{=d v}}\ [_{\texttt{-wh}}\ \text{who}]\ \text{praise}]$$

2.6 Slash-Feature Percolation

The intuition behind Manzini and Roussou's (2000) feature attraction mechanism is that features which need to be checked can be 'piled up', and checked *en masse* by a newly merged expression, provided that the source positions of the features form an appropriate movement chain with the newly merged expression. The slash-feature percolation mechanism introduced into minimalist grammars by Kobele (2007) is designed to minimize the bookkeeping necessary to ensure that the features which are piling up do indeed form a legitimate chain. As the **move** operation already builds movement chains,

site. If there is competition for that landing site, the derivation crashes (because the losing expression will have to make a longer movement than absolutely necessary). Note that the SMC plays on the fact that structuring feature bundles as lists allows features to be temporarily 'hidden'. Using the SMC as a constraint on movement has desirable computational effects (such as guaranteeing efficient recognizability – see (Harkema, 2001; Michaelis, 2001)), although other constraints have been explored in Gärtner and Michaelis (2007).

There are well-known 'counter-examples' to this restriction on movement. Notable among them are the multiple-*wh* fronting constructions familiar in the Slavic languages (Rudin, 1988). To deal with such phenomena, it seems natural to introduce a *wh*-cluster forming operation, which 'saves' otherwise SMC violating configurations by fusing together the offending expressions (Grewendorf, 2001) (see also fn. 20 in (Gärtner and Michaelis, 2005)). The *wh*-in-situ strategy of multiple questions (as in English) is dealt with by imposing a restriction on *wh*-cluster formation in such languages which requires that the phonological matrices of all fused *wh*-items but one be null – forcing spell-out of these *wh*-chains in their base (or case) positions.

Kobele introduces a 'dummy' expression (an *assumption*), with content just a list of features. The **move** operation then treats this dummy expression just like a real one, thereby ensuring that the features that are 'piling up' do indeed stand in an appropriate relation to one another. The second step involves merging in an expression with appropriate features, i.e. one that *could have* taken the place of the dummy expression (thereby *discharging* the assumption). To do all this, two new operations are introduced: **assume** and **discharge**.

2.6.1 Assume

The operation **assume** takes a single argument, the first feature of whose head is =x, and results (non-deterministically) in an expression where the =x feature of the head has been satisfied by the assumption of an expression with initial feature sequence $\texttt{x}\delta$ (for some δ), as shown in figure 7. Assumptions are

$$\textbf{assume}([_{\texttt{=x}\gamma}\ A]) \rightarrow \begin{cases} [_\gamma\ A\ [_\delta\ \texttt{x}\delta]] \\ [_\gamma\ [_\delta\ \texttt{x}\delta]\ A] \end{cases}$$

Figure 7: Cases of **assume**

written just as are normal expressions (i.e. in brackets notated with active features), however instead of containing tree structure, all that is represented is the originally assumed feature sequence. For example, the below represents an assumed expression with a single feature (-wh) left to be checked, where the original assumption was d -k -wh.

$$[_{\texttt{-wh}}\ \texttt{d -k -wh}]$$

For example, given the lexical item $\langle$praise, =d V$\rangle$, one possible output of applying the **assume** operation is given below.

$$[_{\texttt{V}}\ \text{praise}\ [_{\texttt{-k}}\ \texttt{d -k}]]$$

2.6.2 Discharge

Once an assumption is made, it needs eventually to be discharged. The **discharge** operation provides a means of doing this. The **discharge** operation takes as its two arguments an expression containing an assumption, and an expression which will replace, or 'cash out', this assumption. The assumption to be discharged must have exactly one unchecked feature remaining, and the original assumption must be an initial segment of the feature bundle of the expression replacing it.[6] As an example, consider the instance of the **discharge** operation below.

$$\begin{aligned}\textbf{discharge}([_{\mathtt{V}}\ \textsf{praise}\ [_{\mathtt{-k}}\ \mathtt{d}\ \mathtt{-k}]], [_{\mathtt{d}\ \mathtt{-k}\ \mathtt{-wh}}\ \textsf{which boy}]) = \\ [_{\mathtt{V}}\ \textsf{praise}\ [_{\mathtt{-k}\ \mathtt{-wh}}\ \textsf{which boy}]]\end{aligned}$$

Note first that all features of the discharging element are deleted up to (but not including) the remaining feature in the assumption's feature bundle (in this case, it is just the one d feature). Note also that the discharging expression actually replaces the assumption in the original expression.

2.7 On Locality

The reader will have noticed that the **move** and **discharge** operations have been presented 'by example', instead of being given a general definition as have been **merge** and **assume**. Let us denote by A_δ an expression of the form $[_\delta\ \alpha\ \beta]$, and by $A\langle B\rangle$ an expression A which contains designated occurrence of B. Then if $A\langle B\rangle$ and C are expressions, $A\langle C\rangle$ is the result of replacing the occurrence of B in A by C. We define **move** and **discharge** as per figure 8. In this figure, we see that both **move** and **discharge** have a non-local

$$\textbf{move}(A_{\mathtt{+y}\delta}\langle B_{\mathtt{-y}\gamma}\rangle) = [_\delta\ B_\gamma\ A\langle\ t_B\rangle]$$

$$\textbf{discharge}(A\langle[_{\mathtt{-y}}\ \delta\mathtt{-y}]\rangle, B_{\delta\mathtt{-y}\gamma}) = A\langle B_{\mathtt{-y}\gamma}\rangle$$

Figure 8: **move** and **discharge**

character in the following sense: in order to determine whether they can apply to an expression A, and, if so, what the result is, one must conduct a search of unbounded depth inside A (in the case of **move** for an expression B which is to be moved, and in the case of **discharge**, for a hypothesis which is featurally compatible with the second argument). However, because of the SMC, we can keep a finite list of which expressions internal to the main expression have which features. The representation of an expression *praise*

[6]That there be *at least* one feature remaining is to ensure that the assumption is still 'active' at the point it is discharged. That there be *at most* one feature remaining is to make sure that not both **move** and **discharge** can be applied to the same hypothesis at any given time.

Kwasi then looks as per the below, where the finite list of feature sequences to the right of the • indicates which moving pieces the expression contains.

$$[_{\mathtt{v}}\ \textsf{praise}\ [_{\mathtt{-k}}\ \textsf{Kwasi}]] \bullet \langle \mathtt{-k} \rangle$$

In other words, we can simply enrich our representations of expressions with the information about which moving subcomponents they contain. This representational enrichment then completely eliminates the non-locality we observed in determining whether **move** and **discharge** can apply to an expression.

The other source of non-locality (that of computing the result of **move** or **discharge**) can be similarly eliminated if, instead of representing just the features of the moving expressions internal to another after the •, we display the expressions themselves.

$$[_{\mathtt{v}}\ \textsf{praise}\ t_{Kwasi}] \bullet \langle [_{\mathtt{-k}}\ \textsf{Kwasi}] \rangle$$

We can redefine **merge** so that, if its second argument is going to move later on (i.e. if it has licensee features), a trace of it is merged in its place, and it is placed into the list of the first argument.

$$\mathbf{merge}([_{\mathtt{=d\ v}}\ \textsf{praise}] \bullet \langle\rangle, [_{\mathtt{d\ -k}}\ \textsf{Kwasi}] \bullet \langle\rangle) = [_{\mathtt{v}}\ \textsf{praise}\ t_{Kwasi}] \bullet \langle [_{\mathtt{-k}}\ \textsf{Kwasi}] \rangle$$

The same strategy of representing them external to the main expression can be applied to assumptions. The general case of merging a to be moved expression is given in figure 9. Note that the lists of moving expressions in

$$\mathbf{merge}(A_{\mathtt{=x}\gamma} \bullet \phi, B_{\mathtt{x-y}\delta} \bullet \psi) = [_{\gamma}\ A\ t_{B}] \bullet (\phi \langle B_{\mathtt{-y}\delta} \rangle \psi)$$

Figure 9: Merging an expression which will later move

the arguments to **merge** are put together (denoted by juxtaposition) in the result, which results in something like slash-feature percolation (but with moving expressions instead of (or in addition to) slash-features).

With this augmented representation, the superficial non-locality of **move** and **discharge** has been eliminated. Figure 10 shows the definitions of these operations on the augmented representations. One might wonder whether and to what extent our augmented representations are ‘the same’ as the old ones (and thereby whether we have really shown that the non-locality of **move** and **discharge** is only apparent). As this augmented representation changes neither the derivations, nor the thing we ultimately derive, and is

$$\mathbf{move}(A_{\mathtt{+y}\delta} \bullet (\phi_1\langle B_{\mathtt{-y}\gamma}\rangle\phi_2)) = \begin{cases} [_\delta\ B\ A] \bullet (\phi_1\phi_2) & \text{if } \gamma = \epsilon \\ [_\delta\ t_B\ A] \bullet (\phi_1\langle B_\gamma\rangle\phi_2) & \text{otherwise} \end{cases}$$

$$\mathbf{discharge}(A \bullet (\phi_1\langle[_{\mathtt{-y}}\ \delta\mathtt{-y}]\rangle\phi_2), B_{\delta\mathtt{-y}\gamma} \bullet \psi) = A \bullet (\phi_1\langle B_{\mathtt{-y}\gamma}\rangle\phi_2\psi)$$

Figure 10: **move** and **discharge**, locally

moreover easily obtainable from the original representation, it is not clear what kind of empirical content could be given to the claim that the augmented representations constitute a significantly different claim about the nature of our linguistic faculty. Indeed, all that we have done is identify where our operations have to do some extra work, determine that this extra work is in fact unnecessary, and change our representations so as to eliminate this extra work. It is like switching from decimal to binary representations of numbers, because we see that we have to multiply by two more often than by ten. That being said, I will continue to use the 'non-local' representation in the rest of this paper. It has the advantage of looking familiar.

2.8 Movement and Slash-Features

Adding slash-feature percolation (in the form of the operations **assume** and **discharge**) to the minimalist grammar framework results in an explosion of syntactic ambiguity. Even simple sentences like *Kofi smiled*, previously unambiguous, now have (at least) two derivations: one where the DP *Kofi* is merged with the verb *smiled*, and one where the DP *Kofi* discharges an assumption of the form $[_{\mathtt{-k}}\ \mathtt{d}\ \mathtt{-k}]$. However, although every derivation using slash-features has a corresponding derivation using movement, the reverse is not true.[7] In particular, slash-feature percolation cannot describe cases where a remnant 'moves' over something that has been extracted out of it (violating the proper binding condition (Fiengo, 1977)),[8] as in the configuration below.

$$[_{YP} \ldots t_{XP} \ldots] \ldots XP \ldots t_{YP}$$

[7]The system in Kobele (2007) uses a slightly different formulation of the **discharge** operation, which, allowing for smuggling (in the sense of Collins (2005)), can no longer be simulated in every case by movement.

[8]The slash-feature mechanism here thus implements a (restricted, as there is no downward movement) version of the trace-binding algorithm sought after by Pullum (1979).

The reason for this is that at the point in the derivation where XP is put into its surface position, its source position (inside YP) doesn't yet exist. Instead, only an assumption that we will have a YP has been made – nothing has been said about whether we will also have an XP. Contrasting this with the case of movement, in the case of movement, both XP *and* YP are present in the derivation before either of them needs to move. What this means is that if we can find examples in language of remnant movement, then we will be able to analyze them *only* in terms of actual movement, *not* using slash-feature percolation. In terms of the theory in the following section, remnants must reconstruct below the lowest expression extracted out of them.

3 A Theory of Reconstruction

Minimalist grammars with slash-feature percolation give us a natural way of dealing with reconstruction phenomena, one that takes advantage of the sudden multiplicity of derivations for sentences. What the hybrid **merge-move** and **assume-discharge** system does is to allow moving elements to be introduced into the derivation at any point between their chain-initial and chain-final positions (figure 11). It is thus a natural move to make to tie the point at which an expression is inserted into the derivation to its reconstruction possibilities. We can think of two obvious ways to do this. First, we might demand that an expression be reconstructed into the position at which it enters the derivation. This approach minimizes as much as possible the potential spurious ambiguity introduced into the minimalist grammar system by having both movement and slash-feature percolation. The other option is a relaxation of the first, requiring only that an expression be reconstructed *no lower* than the position at which it enters the derivation. Only with the first approach however will we be able to derive Huang's generalization without making any further assumptions.[9] We restate Principle A of the Binding Theory in the following terms (whether and how the other principles in figure 2 should be relativized to positions in which the talked about elements entered the derivation is orthogonal to the present issue).

Principle A (revised)

> A reflexive must be c-commanded by a co-indexed expression within the first tensed clause *above the point at which it entered the derivation.*

[9]We can think of both of these options as similar to Epstein and Seely's (2006) ideas on syntactic relations, whereby relations such as c-command are incrementally specified during the derivation, with each derivational step potentially adding new relata to the relation.

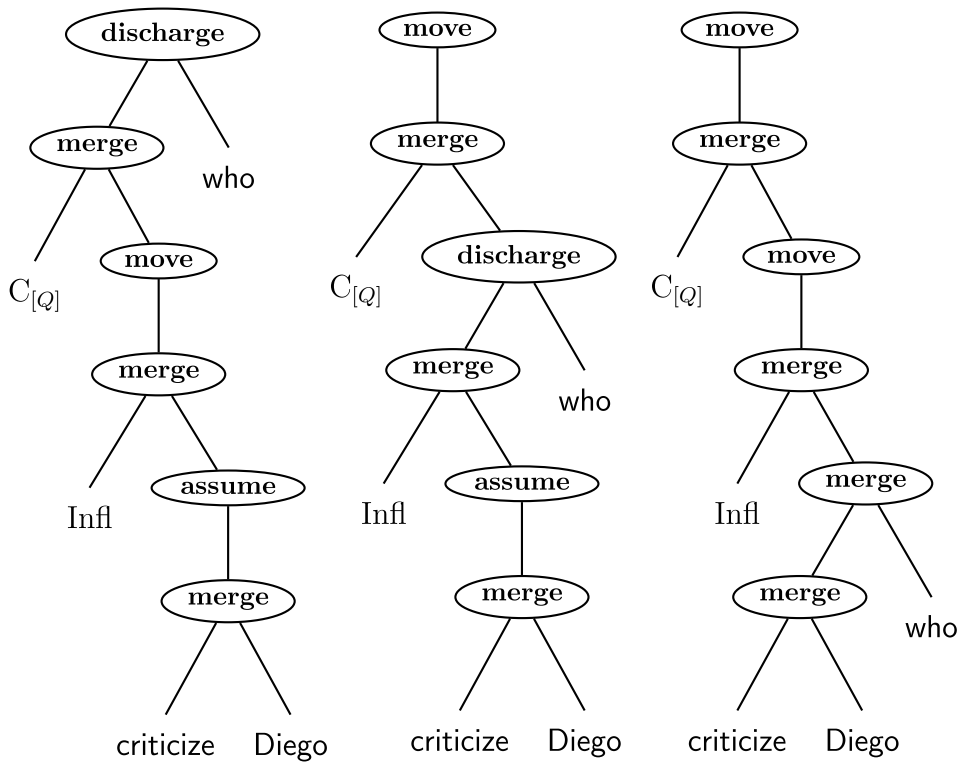

Figure 11: Three derivations for the sentence *Who criticized Diego?*

Now that we have specified both a syntactic theory (minimalist grammars with both slash-feature percolation and movement) and an interpretative theory (the revision of Principle A above), we are finally in a position to derive Huang's generalization. Let us fix our analysis of a tiny fragment of English consisting of transitive verbs like *criticize*, and of sentential complement verbs like *believe*, as in figure 12. The lexicon given in figure 12 generates many

⟨will, `=v +k S`⟩	⟨-ϵ, `=V =d v`⟩
⟨criticize, `=d +k V`⟩	⟨believe, `=S =d v`⟩
⟨Kwasi, `d -k`⟩	⟨himself, `d -k`⟩
⟨ϵ, `=v v -top`⟩	⟨ϵ, `=S +top S`⟩

Figure 12: A fragment of English

non-sentences, such as *Himself will criticize Kwasi.* These will be ruled out by principle A of the binding theory, which acts as a restriction on the distribution of the lexical item himself. With the syntactic operations our grammar formalism permits us, the only derivation of a sentence like *Criticize himself, Kwasi will* is one where the verb phrase *criticize himself* is first merged as sister to *will.*[10] First, we give a well-formed derivation of this sentence.

We begin by merging together criticize and the anaphor himself.

1. **merge**($\langle \textsf{criticize}, \texttt{=d +k V}\rangle$, $\langle \textsf{himself}, \texttt{d -k}\rangle$)

$$[_{\texttt{+k V}}\ \textsf{criticize}\ [_{\texttt{-k}}\ \textsf{himself}]]$$

Next the verb assigns case to its object. Note that both `+k` and `-k` features of the verb and its object are checked by the **move** operation.

2. **move**(1)

$$[_{\texttt{V}}\ [\ \textsf{himself}]\ [\textsf{criticize}\ t_{himself}]]$$

In the next step, little-v merges with the previous expression. Note that this triggers head movement of the V *criticize.*

3. **merge**($\langle \textsf{-}\epsilon, \texttt{=V =d v}\rangle$,2)

$$[_{\texttt{=d v}}\ \textsf{criticize-}\epsilon\ [[\ \textsf{himself}]\ [t_{criticize}\ t_{himself}]]]$$

Then the agent argument is selected.

4. **merge**(3,$\langle \textsf{Kwasi}, \texttt{d -k}\rangle$)

$$[_{\texttt{v}}\ [_{\texttt{-k}}\ \textsf{Kwasi}]\ [\textsf{criticize-}\epsilon\ [[\ \textsf{himself}]\ [t_{criticize}\ t_{himself}]]]]$$

Next, the vP is marked as requiring topicalization.

5. **merge**($\langle \epsilon, \texttt{=v v -top}\rangle$,4)

$$[_{\texttt{v -top}}\ \epsilon\ [[_{\texttt{-k}}\ \textsf{Kwasi}]\ [\textsf{criticize-}\epsilon\ [[\ \textsf{himself}]\ [t_{criticize}\ t_{himself}]]]]]$$

Then will selects the above vP.

6. **merge**($\langle \textsf{will}, \texttt{=v +k S}\rangle$,5)

$$[_{\texttt{+k S}}\ \textsf{will}\ [_{\texttt{-top}}\ \epsilon\ [[_{\texttt{-k}}\ \textsf{Kwasi}]\ [\textsf{criticize-}\epsilon\ [[\ \textsf{himself}]\ [t_{criticize}\ t_{himself}]]]]]]$$

[10]As we shall see, this is not a good description of what happens, as it is not the v' *criticize himself* which is sister to *will*, but rather the vP *Kwasi criticize himself.*

At this point, the expression derived is getting too big to be written on a single line, and so we abbreviate it using the convention discussed earlier as the below.

$[_{\texttt{+k S}}$ will $[_{\texttt{-top}}$ $[_{\texttt{-k}}$ Kwasi] criticize himself]]

The next step of the derivation is to move the subject *Kwasi* for case. Note again that both `+k` and `-k` features are checked.

7. **move**(6)

   $[_{\texttt{S}}$ [Kwasi] will $[_{\texttt{-top}}$ t_{Kwasi} criticize himself]]

Next, a head hosting a `+top` feature merges with the expression thus far derived.

8. **merge**(⟨ϵ, `=S +top S`⟩,7)

   $[_{\texttt{+top S}}$ ϵ [[Kwasi] will $[_{\texttt{-top}}$ t_{Kwasi} criticize himself]]]

Finally, the vP moves to check its `-top` feature.

9. **move**(8)

   $[_{\texttt{S}}$ [t_{Kwasi} criticize himself] [ϵ [[Kwasi] will $t_{criticize\ himself}$]]]

Abbreviating the expression derived in 9 as per our conventions, we obtain the below.

$[_{\texttt{S}}$ criticize himself Kwasi will]

As per our theory of reconstruction, because in the derivation of this sentence the fronted predicate was merged low, it behaves for the purposes of reconstruction as though it were in its base position.

A (short and unsuccessful) derivation which attempts to use slash-feature percolation to deal with the topicalized VP follows. This shows this sentence to be derivationally unambiguous (at least with respect to the fronted predicate), and thus (generalizing a little) that the only reconstructive possibilities available to sentences with fronted predicates, are those in which the predicate is interpreted in its base position.

The previous derivation introduced the vP in its lower chain position, to introduce it (via **discharge**) in its higher chain position, the to-be-moved vP is first introduced as an assumption.

1. **assume**(⟨will, `=v +k S`⟩)

   $[_{\texttt{+k S}}$ will $[_{\texttt{-top}}$ `v -top`]]

Already at this point, the derivation can continue no further, as there is no subexpression with matching feature `-k` which can be used to check the `+k` feature of will. In other words, while we have assumed that we will find some expression with feature bundle `v -top`, we do not know that it will itself contain a moving expression waiting to check its `-k` feature.[11]

We can compare the derivation of predicate fronting sentences to those with fronted DPs, like *Which portrait of himself did Kwasi believe that Diego criticized?*, which allow for the fronted DP *which portrait of himself* to be reconstructed either in its base position (thereby giving rise to the reading in which *himself* is coreferent with *Diego*) or in the intermediate SPEC-CP position (giving rise to the reading where *himself* and *Kwasi* are coreferent).

Here we will see only two derivations of the (shorter) sentence *Which portrait of himself did Kwasi criticize*, which serves to illustrate the fact that the fronted DP is not restricted in its reconstruction possibilities.[12]

We first extend our lexicon in figure 12 with the complex wh-anaphor *which portrait of himself*, to which we assign the type `d -k -wh`,[13] and a +WH Comp position.

$$\langle \textsf{which portrait of himself}, \texttt{d -k -wh}\rangle$$
$$\langle \textsf{-}\epsilon, \texttt{=S +wh S}\rangle$$

In the first derivation of interest to us, the wh-phrase is introduced in its case position via the operation **discharge**. This derivation will correspond to a reading of the sentence where the anaphor is bound by its co-argument.

We begin by assuming the existence of an appropriate DP for criticize.

[11]It is instructive to consider how to extend the system to allow this kind of derivation. In other words, why can't we simply make our hypotheses more explicit (so, not just 'we have some expression with feature bundle `v -top`', but rather 'we have some expression with feature bundle `v -top`, which itself contains an expression with feature bundle `-k`')? The reasoning is subtle, and revealing of an important but oft neglected fact. The answer is, simply put, that *there are no expressions with feature bundle* `-k` *derivable in our grammars*! The DP moving for case (with feature bundle `-k`) does not exist in isolation, but only as a part of a larger containing expression. This is true not just of minimalist grammars, but in *all* variants of minimalism. Formally speaking, we are confronted with the distinction between derivational and derived constituents. A DP with feature bundle `-k` is a derived constituent, but the discharge operation (and grammatical operations in general) are defined only over derivational constituents.

[12]The fact that the anaphor will be unbound in the second derivation is not of concern to us here.

[13]Of course, this is a complex expression composed (at least) of the lexical items which, portrait, and himself. As the internal structure of this expression isn't relevant for our purposes here (which are simply to investigate the reconstruction possibilities allowed to sentences containing it), we can safely ignore these niceties.

1. **assume**(⟨criticize, `=d +k V`⟩)

$$[_{\texttt{+k V}} \text{criticize } [_{\texttt{-k}}\ \texttt{d -k}]]$$

Next, we discharge the assumption by replacing it with which portrait of himself.

2. **discharge**(1,⟨which portrait of himself, `d -k -wh`⟩)

$$[_{\texttt{+k V}} \text{criticize } [_{\texttt{-k -wh}} \text{which portrait of himself}]]$$

Case is then assigned to the object.

3. **move**(2)

$$[_{\texttt{V}} [_{\texttt{-wh}} \text{which portrait of himself}] [\text{criticize } t_{which}]]$$

Next the little-v head selects the VP thus derived.

4. **merge**(⟨-ϵ, `=V =d v`⟩,3)

$$[_{\texttt{=d v}} \text{criticize-}\epsilon\ [[_{\texttt{-wh}} \text{which portrait of himself}]\ [t_{criticize}\ t_{which}]]]$$

We then merge the agent Kwasi.

5. **merge**(4,⟨Kwasi, `d -k`⟩)

$$[_{\texttt{v}} [_{\texttt{-k}} \text{Kwasi}] [\text{criticize-}\epsilon\ [[_{\texttt{-wh}} \text{which portrait of himself}]\ [t_{criticize}\ t_{which}]]]]$$

As our derived expression is rapidly becoming unwieldy, we abbreviate it as per our convention as the below.

$$[_{\texttt{v}} [_{\texttt{-k}} \text{Kwasi}] \text{ criticize } [_{\texttt{-wh}} \text{which portrait of himself}]]$$

In the next derivational step, will selects the expression derived thus far.

6. **merge**(⟨will, `=v +k S`⟩,5)

$$[_{\texttt{+k S}} \text{will } [[_{\texttt{-k}} \text{Kwasi}] \text{ criticize } [_{\texttt{-wh}} \text{which portrait of himself}]]]$$

Next the subject raises to check its case features.

7. **move**(6)

$$[_{\texttt{S}} [\text{Kwasi}] [\text{will } [t_{Kwasi} \text{ criticize } [_{\texttt{-wh}} \text{which portrait of himself}]]]]$$

The next derivational step is to introduce a +WH Comp, the merger of which induces head movement of the infl element will.

8. **merge**($\langle$-ϵ, `=S +wh S`$\rangle$,7)

   $[_{\texttt{+wh S}}$ will-ϵ [[Kwasi] $[t_{will}$ $[t_{Kwasi}$ criticize $[_{\texttt{-wh}}$ which portrait of himself]]]]]

Finally, the wh-phrase moves to check its wh feature.

9. **move**(8)

   $[_{\mathsf{S}}$ [which portrait of himself] [will-ϵ [[Kwasi] $[t_{will}$ $[t_{Kwasi}$ criticize t_{which}]]]]]

In this derivation, the wh-anaphor is introduced early enough to be bound by its co-argument, *Kwasi*, as depicted in the tree on the left in figure 13.

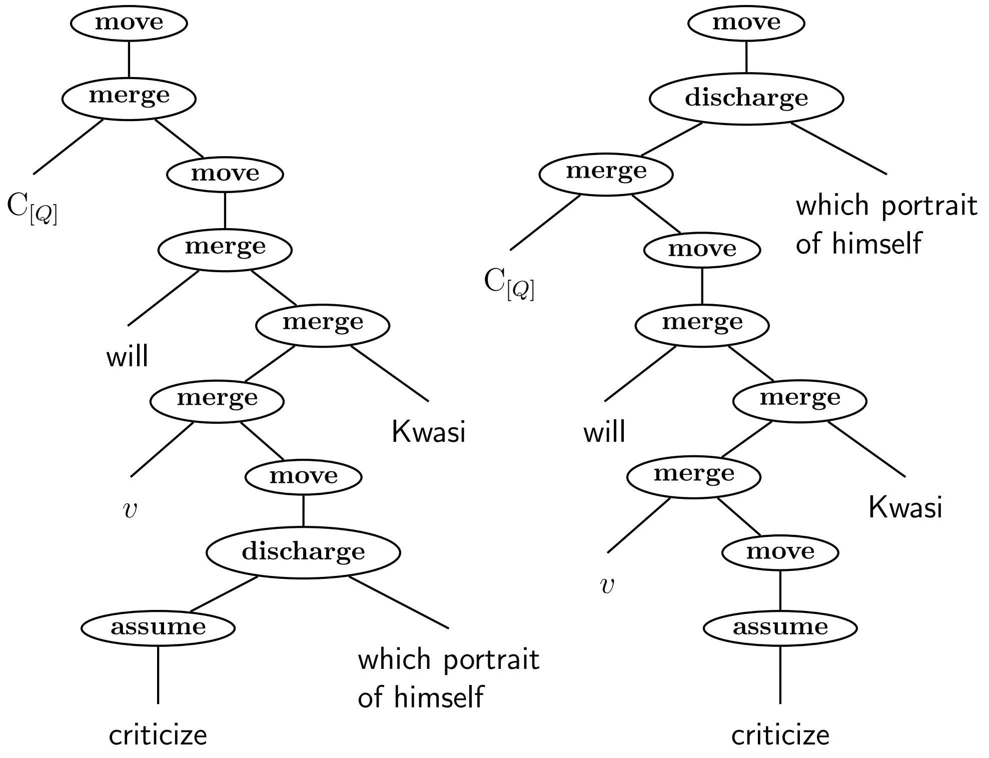

Figure 13: Two derivations of the sentence *which portrait of himself will Kwasi criticize?*

Introducing the wh-anaphor *after* the S containing its co-argument has been completed (as depicted in the tree on the right in figure 13) would force it to be bound by an argument in a higher clause. This second derivation begins with the assumption of a +wh DP as the object of criticize.

1. **assume**(⟨criticize, =d +k V⟩)

$$[_{\texttt{+k V}} \text{ criticize } [_{\texttt{-k -wh}} \texttt{ d -k -wh}]]$$

Next, case is assigned to the hypothetical object.

2. **move**(1)

$$[_{\texttt{V}} [_{\texttt{-wh}} \texttt{ d -k -wh}] [\text{criticize } t_{\texttt{d-k-wh}}]]$$

Next, little-v is merged, triggering head movement of criticize.

3. **merge**(⟨-ϵ, =V =d v⟩,2)

$$[_{\texttt{=d v}} \text{ criticize-}\epsilon \; [[_{\texttt{-wh}} \texttt{ d -k -wh}] [t_{criticize} \; t_{\texttt{d-k-wh}}]]]$$

The agent is merged.

4. **merge**(3,⟨Kwasi, d -k⟩)

$$[_{\texttt{v}} [_{\texttt{-k}} \text{ Kwasi}] [\text{criticize-}\epsilon \; [[_{\texttt{-wh}} \texttt{ d -k -wh}] [t_{criticize} \; t_{\texttt{d-k-wh}}]]]]$$

Next will is merged with the vP.

5. **merge**(⟨will, =v +k S⟩,4)

$$[_{\texttt{+k S}} \text{ will } [[_{\texttt{-k}} \text{ Kwasi}] [\text{criticize-}\epsilon \; [[_{\texttt{-wh}} \texttt{ d -k -wh}] [t_{criticize} \; t_{\texttt{d-k-wh}}]]]]]$$

Abbreviating, we obtain the below.

$$[_{\texttt{+k S}} \text{ will } [_{\texttt{-k}} \text{ Kwasi}] \text{ criticize } [_{\texttt{-wh}} \texttt{ d -k -wh}]]$$

The subject moves to receive case.

6. **move**(5)

$$[_{\texttt{S}} [\text{Kwasi}] [\text{will } t_{Kwasi} \text{ criticize } [_{\texttt{-wh}} \texttt{ d -k -wh}]]]$$

Next, a +WH position is made available, and will head-moves to the new head.

7. **merge**(⟨-ϵ, =S +wh S⟩,6)

$$[_{\texttt{+wh S}} \text{ will-}\epsilon \; [[\text{Kwasi}] [t_{will} \; t_{Kwasi} \text{ criticize } [_{\texttt{-wh}} \texttt{ d -k -wh}]]]]$$

Now, the expression which portrait of himself discharges the assumption made earlier. Note that the anaphor is outside the binding domain of the subject Kwasi.

8. **discharge**(7,⟨which portrait of himself, `d -k -wh`⟩)

   $[_{\texttt{+wh s}}$ will-ϵ [[Kwasi] [t_{will} t_{Kwasi} criticize $[_{\texttt{-wh}}$ which portrait of himself]]]]

Finally, the wh-phrase moves to check its wh feature.

9. **move**(8)

   $[_{\texttt{s}}$ [which portrait of himself] [will-ϵ [[Kwasi] [t_{will} t_{Kwasi} criticize t_{which}]]]]

4 Conclusion

We have seen that under a natural account of the syntax-semantics interface according to which the position into which elements are reconstructed depends on the point at which they are inserted into the derivation, Huang's generalization can be derived as a consequence of the architecture of the hybrid **merge-move/assume-discharge** minimalist grammar system.

Nothing has been said in this paper about *where* slash-features stop and movement begins – in other words, we have been simply looking at the architecture of the system, and have abstracted away from questions like which dependencies should be A dependencies, and which A-bar. Although it is natural to stipulate, in the context of DPs, that they be introduced via assumptions, and discharged in their case positions (recovering the traditional perspective on the A/A-bar distinction), this is not necessary to derive Huang's generalization, and has in fact been argued against by Sportiche (2003), who notes that reconstruction is sometimes possible into what are traditionally considered A positions. An interesting alternative made possible by this formal system is to view the reconstruction differences between A and A-bar movement as the result, not of a grammatical prohibition, but rather of a parsing preference. If we assume that both slash-feature percolation and movement are always available at each derivational step, but that the parser first pursues parses involving slash-feature percolation, we derive that, in traditional cases of A movement, derivations *without* reconstruction in A positions are recovered first. Note that this doesn't affect our derivation of Huang's generalization, as in cases of remnant movement, the only available derivations are those which involve reconstruction into lower chain positions.

Finally, note that Heycock (1995) has argued against the adequacy of Huang's generalization. On the basis of examples such as the below, she offers a new generalization based on referentiality, according to which "referential" phrases, but not "non-referential" ones, may be reconstructed into positions other than their base positions.

(8) Which stories about Diana$_i$ did she$_i$ most object to?

(9) *How many stories about Diana$_i$ is she$_i$ likely to invent?

As discussed above, in the present framework, there are no constraints on what can be first merged where save for those imposed by the inability of slash-feature percolation to support remnant movement. Heycock's insight surrounding the referentiality distinction can be implemented here to restrict the otherwise spurious ambiguity engendered by the addition of hypothetical reasoning to the minimalist grammar system.

References

Boeckx, C. A. (2000). A note on contraction. *Linguistic Inquiry 31*(2), 357–366.

Chomsky, N. (1965). *Aspects of the Theory of Syntax.* Cambridge, Massachusetts: MIT Press.

Chomsky, N. (1976). Conditions on rules of grammar. *Linguistic Analysis 2*, 303–351.

Chomsky, N. (1995). *The Minimalist Program.* Cambridge, Massachusetts: MIT Press.

Collins, C. (2005). A smuggling approach to the passive in English. *Syntax 8*(2), 81–120.

Epstein, S. D. and T. D. Seely (2006). *Derivations in Minimalism*, Volume 111 of *Cambridge Studies in Linguistics.* Cambridge University Press.

Fiengo, R. (1977). On trace theory. *Linguistic Inquiry 8*(1), 35–61.

Fox, D. (2000). *Economy and Semantic Interpretation.* Cambridge, Massachusetts: MIT Press.

Gärtner, H.-M. and J. Michaelis (2005). A note on the complexity of constraint interaction: Locality conditions and minimalist grammars. In P. Blache, E. Stabler, J. Busquets, and R. Moot (Eds.), *Logical Aspects of Computational Linguistics*, Volume 3492 of *Lecture Notes in Computer Science*, pp. 114–130. Berlin: Springer.

Gärtner, H.-M. and J. Michaelis (2007). Some remarks on locality conditions and minimalist grammars. In U. Sauerland and H.-M. Gärtner (Eds.), *Interfaces + Recursion = Language?*, Volume 89 of *Studies in Generative Grammar*, pp. 161–195. Berlin: Mouton de Gruyter.

Gazdar, G., E. Klein, G. Pullum, and I. Sag (1985). *Generalized Phrase Structure Grammar.* Cambridge, MA: Harvard University Press.

Grewendorf, G. (2001). Multiple *wh*-fronting. *Linguistic Inquiry 32*(1), 87–122.

Harkema, H. (2001). *Parsing Minimalist Languages.* Ph. D. thesis, University of California, Los Angeles.

Heycock, C. (1995). Asymmetries in reconstruction. *Linguistic Inquiry 26*(4), 547–570.

Huang, C.-T. J. (1993). Reconstruction and the structure of VP: Some theoretical consequences. *Linguistic Inquiry 24*(1), 103–138.

Kayne, R. (1994). *The Antisymmetry of Syntax.* Cambridge, Massachusetts: MIT Press.

Kobele, G. M. (2006). *Generating Copies: An investigation into structural identity in language and grammar.* Ph. D. thesis, University of California, Los Angeles.

Kobele, G. M. (2007). A formal foundation for A and A-bar movement in the minimalist program. In M. Kracht, G. Penn, and E. P. Stabler (Eds.), *Mathematics of Language 10.* UCLA.

Kobele, G. M. (2008). Across-the-board extraction in minimalist grammars. In *Proceedings of the Ninth International Workshop on Tree Adjoining Grammar and Related Formalisms (TAG+9)*, pp. 113–128.

Koopman, H. and D. Sportiche (1991). The position of subjects. *Lingua 85*, 211–258.

Lasnik, H. (1999). Chains of arguments. In S. D. Epstein and N. Hornstein (Eds.), *Working Minimalism*, Number 32 in Current Studies in Linguistics, Chapter 8, pp. 189–215. Cambridge, Massachusetts: MIT Press.

Mahajan, A. (1990). *The A/A-bar Distinction and Movement Theory.* Ph. D. thesis, Massachusetts Institute of Technology.

Manzini, M. R. and A. Roussou (2000). A minimalist theory of A-movement and control. *Lingua 110*(6), 409–447.

Matushansky, O. (2006). Head movement in linguistic theory. *Linguistic Inquiry 37*(1), 69–109.

Michaelis, J. (2001). *On Formal Properties of Minimalist Grammars.* Ph. D. thesis, Universität Potsdam.

Müller, G. (2008). On deriving CED effects from the PIC. ms., Universität Leipzig.

Müller, G. and W. Sternefeld (1993). Improper movement and unambiguous binding. *Linguistic Inquiry 24*(3), 461–507.

Nunes, J. (2001). Sideward movement. *Linguistic Inquiry 32*(2), 303–344.

Pullum, G. K. (1979). The nonexistence of the trace-binding algorithm. *Linguistic Inquiry 10*(2), 356–362.

Rudin, C. (1988). On multiple questions and multiple WH fronting. *Natural Language and Linguistic Theory 6*(4), 445–501.

Sportiche, D. (2003). Reconstruction, binding and scope. available at: `http://ling.auf.net/lingBuzz/000017`.

Stabler, E. P. (1997). Derivational minimalism. In C. Retoré (Ed.), *Logical Aspects of Computational Linguistics*, Volume 1328 of *Lecture Notes in Computer Science*, pp. 68–95. Berlin: Springer-Verlag.

Stabler, E. P. (2001). Recognizing head movement. In P. de Groote, G. F. Morrill, and C. Retoré (Eds.), *Logical Aspects of Computational Linguistics*, Volume 2099 of *Lecture Notes in Artificial Intelligence*, Berlin, pp. 254–260. Springer Verlag.

Tada, H. (1993). *A/A-bar Partition in Derivation.* Ph. D. thesis, Massachusetts Institute of Technology.