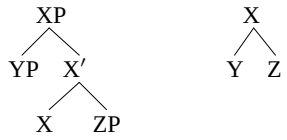
Learning Mirror Theory

REVISED 29.4.02

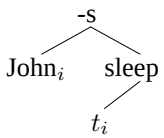
Gregory M. Kobele, Travis Collier, Charles Taylor, and Edward P. Stabler
University of California, Los Angeles

1. Mirror Theory

Mirror Theory is a syntactic framework developed in (Brody, 1997), where it is offered as a consequence of eliminating purported redundancies in Chomsky's minimalism (Chomsky, 1995). A fundamental feature of Mirror Theory is its requirement that the syntactic head-complement relation mirror certain morphological relations (such as constituency). This requirement constrains the types of syntactic structures that can express a given phrase; the morphological constituency of the phrase determines part of the syntactic constituency, thereby ruling out other, weakly equivalent, alternatives. A less fundamental, but superficially very noticeable feature is the elimination of phrasal projection. Thus the X-bar structure on the left becomes the mirror theoretic structure on the right:



(Brody, 1997) calls this systematic collapse of X, X' and XP nodes 'telescope'. Every node may now have phonetic content, and children are identified as specifiers or complements depending on their direction of branching; left-daughters are specifiers and right-daughters are complements (previously, specifiers were children of XP, and complements were children of X'). Furthermore, the complement relation is a "word-forming" relation, where according to the "mirroring" relation, the phonetic content of each head follows the phonetic content of its complement. For example, MTGs can generate trees like the following, which given the "mirror" relation between morphology and syntax, is pronounced *John sleep -s*:



1.1. Trees

A mirror theoretic tree (MTT) can be viewed as a standard binary branching tree together with two functions; one, a function f from branches to a two element set $\{right, left\}$, the other, a function g from nodes to a two element set $\{strong, weak\}$. If a is the parent of a', then a' is a specifier (or left child) of a if $f(\langle a, a'\rangle) = left$, and a complement (or right child) of a otherwise. Formally, we represent a MTT as a particular kind of tree domain:

Definition 1
A MTT $\tau = \langle T, S\rangle$ where $T, S \subseteq \{0,1\}^*$ such that

1. $S \subseteq T$
2. T is prefix closed (if $x \in T$ and $x = yz$ then $y \in T$)

Domination corresponds to the initial substring relation with x dominating y iff x is an initial substring of y. The greatest node dominating both x and y, $x \wedge y$, is their longest common initial substring. The function g from nodes to $\{strong, weak\}$ is the characteristic function of the set S:

$$g(t) = \begin{cases} strong, \text{ if } t \in S \\ weak, \text{ if } t \notin S \end{cases}$$

 Proceedings of the Sixth International Workshop on Tree Adjoining Grammar and Related Frameworks (TAG+6), pp. 101–108. Universitá di Venezia.

From Definition 1 we define f from branches to $\{right, left\}$ as follows. A child is a left child if it ends in a '1', and it is a right child if it ends in a '0'.

$$f(\langle x, xn\rangle) = \begin{cases} left, \text{ if } n = 1 \\ right, \text{ if } n = 0 \end{cases}$$

As even internal nodes may have phonetic content, the standard definitions of the yield of a tree will not suffice. We want a node to be spellt out after its left subtree, and before its right subtree. We define a total order $\prec\!\!\prec$ on the nodes of T, such that $x \prec\!\!\prec y$ whenever x is visited before y in an in-order tree traversal of T. Thus $x \prec\!\!\prec y$ holds between x and y just in case one of the following is true:

1. $y \lhd^* x$ and x is in the left subtree of y
2. $x \lhd^* y$ and y is in the right subtree of x
3. $x \prec\!\!\prec x \wedge y$ and $x \wedge y \prec\!\!\prec y$

This gives us a strict SPEC - HEAD - COMP ordering. But wait. There's more. The partitioning of branches into left branches and right branches is not just to determine a reletive precedence with respect to a parent. Right branches are different from left branches in kind; a maximal sequence of right branches is what Brody (1997) calls a morphological word (MW), and a morphological word has a special status with respect to spellout - all the nodes in a MW are spellt out as a single unit. The relation $\prec\!\!\prec$ determines the relative ordering of MWs at spellout.

We define a morphological word to be a block in (the partition induced by) the equivalence relation $\approx$ defined on T in the following manner:

$$x \approx y \text{ iff } y \in x0^* \vee x \in y0^*$$

Two nodes are equated by this relation just in case one is the complement of (...the complement of) the other. As trees are binary branching, the immediate domination relation totally orders each MW. With each MW B, we associate an element $p(B) \in B$ and call $p(B)$ the spellout position of B.[1] Given two MWs B, B', every $y \in B$ is spellt out before any $z \in B'$ iff $p(B) \prec\!\!\prec p(B')$. At this point the nodes in each MW must be spellt out in a contiguous manner (as $\prec\!\!\prec$ totally orders T), but nothing has been said about the relative order in which they are spellt out. In keeping with (Brody, 1997) (but see (Kobele, forthcoming) for alternatives), we adapt Brody's mirror principle (whence '*Mirror Theory*') to our terminology:

(1) *The Mirror Principle*
if x is the complement of y then y is spellt out before x.

Thus, each MW is spellt out in 'reverse domination order' (x is spellt out before y iff $y \lhd^+ x$).

1.2. Mirror Theoretic Grammars

A formal treatment of mirror theory inspired by Minimalist Grammars (Stabler, 1997) is given in (Kobele, forthcoming), where an empirically grounded restriction of the formalism therein is shown to be weakly equivalent to MCTAGs (Joshi, 1987).[2]A mirror theoretic expression is defined to be a mirror theoretic tree along with a labeling function from the nodes of the tree to a set of labels. A label consists of a phonetic part (which is opaque to the syntax) and a finite sequence of syntactic features. A mirror theoretic grammar consists of a finite lexicon of 'basic' expressions, together with two structure building operations, $merge$ and $move$, which build expressions from others either by adjoining structures, or by displacing sub-parts of structures. Each operation is feature driven, and 'checks' features (and thus a derived expression will have fewer features than the sum total of the features of the expressions (tokens) used to derive it). The expressions generated by the grammar are those in the closure of the lexicon under the structure building functions. A complete expression is one all of whose features have been checked, save for the category feature of the root, and the string language at a particular category is simply the yields of the complete expressions of that category.

1. The element so picked out, $p(B)$, is defined in (Brody, 1997) to be the 'deepest' node, if no nodes in B are strong. If some nodes in B are strong, then $p(B)$ is the 'highest' one of the strong nodes. In other words, if $S \cap B \neq \emptyset$, then $p(B) = x$, where $\forall y \in S \cap B\ x \lhd^* y$. If $S \cap B = \emptyset$, then $p(B) = x$ such that $\forall y \in B\ y \lhd^* x$
2. Because of the "mirroring" and the relative flexibility in where MWs get spellt out in relation to the other material, even the movement-free subset of the framework defines a proper superset of the context free languages. See (Michaelis, this volume) for a discussion closely related to this issue.

Definition 2
A MTG $G = \langle \Sigma, Syn, Lex, \{merge, move\}\rangle$, where

1. Σ is a non-empty set (the pronounced elements)

2. Syn is the disjoint union of the following sets (the syntactic features):

   (a) $base$, a non-empty finite set.
   (b) $cselect = \{=b | b \in base\}$
   (c) $sselect = \{b= | b \in base\}$
   (d) $licensees = \{-b | b \in base\}$
   (e) $licensors = \{+b | b \in base\}$

   An expression is a pair $\langle\langle T, S\rangle, \mu\rangle$, where $\langle T, S\rangle$ is a MTT, and $\mu : T \to \Sigma^* \times Syn^*$ is the labelling function.

3. Lex is a finite set of expressions $\langle\langle T, S\rangle, \mu\rangle$, such that[3]

   (a) $|T| = 1$, and
   (b) $\mu : T \to \Sigma^* \times cselect^? \ (sselect + licensors)^? \ base \ licensees^*$

   The shape of the lexical labels is partly determined by the nature of MTTs (and the particular generating functions we have).[4]The precategory sequence (the features before the $base$ category) allows for up to one complement (cselection features) and up to one specifier (sselection or licensor features). Each lexical item has a category (a $base$ feature), and no more than one, as nodes in any tree have only at most one parent. There are no restrictions as to the number of licensee features - movement is ad libitum.

merge $merge$ is a function from pairs of expressions to single expressions. We divide the presentation of the function definition into two cases according to whether the merged item is merged into the specifier ($smerge$) or the complement ($cmerge$) position.

SMERGE $smerge$ is defined on the pair of expressions $\langle\langle T_1, S_1\rangle, \mu_1\rangle, \langle\langle T_2, S_2\rangle, \mu_2\rangle$ iff all of the following obtain:

- the root of T_1 has an available specifier position ($1 \notin T_1$)
- the first syntactic feature of the root of T_1 is b= , and
- the first syntactic feature of the root of T_2 is b

In this case, $smerge$ is defined on the pair, and it maps to the expression $\langle\langle T, S\rangle, \mu\rangle$, where

- $T = T_1 \cup 1T_2$
- $S = S_1 \cup 1S_2$
- the label of the root of T is gotten from the label of the root of T_1 by deleting the first syntactic feature
- the label of the left child of T is gotten from the label of the root of T_2 by deleting the first syntactic feature
- otherwise, for $x \in T_1$, $\mu(x) = \mu_1(x)$, and for $x \in T_2$, $\mu(1x) = \mu_2(x)$

CMERGE $cmerge$ is defined on the pair of expressions $\langle\langle T_1, S_1\rangle, \mu_1\rangle, \langle\langle T_2, S_2\rangle, \mu_2\rangle$ iff all of the following obtain:

- the root of T_1 has an available complement position ($0 \notin T_1$)
- the first syntactic feature of the root of T_1 is =b, and
- the first syntactic feature of the root of T_2 is b

In this case, $cmerge$ is defined on the pair, and it maps to the expression $\langle\langle T, S\rangle, \mu\rangle$, where

- $T = T_1 \cup 0T_2$
- $S = S_1 \cup 0S_2$
- the label of the root of T is gotten from the label of the root of T_1 by deleting the first syntactic feature
- the label of the right child of T is gotten from the label of the root of T_2 by deleting the first syntactic feature
- otherwise, for $x \in T_1$, $\mu(x) = \mu_1(x)$, and for $x \in T_2$, $\mu(0x) = \mu_2(x)$

3. $\Phi^?$ should be read as 'one or zero tokens of Φ'.
4. Only partly, as there is no functional reason that sselection features cannot precede cselection features. Doing so makes no difference (other than further complicating the description of a lexical label).

move $move$ is a function from expressions to expressions. $move(\langle\langle T_1, S_1\rangle, \mu_1\rangle)$ is defined whenever the following conditions obtain:

- the root of T_1 has an available specifier position
- the first syntactic feature of the root of T_1 is +b, and
- there is *exactly one* node $n \in T_1$ such that the first syntactic feature in n is -b, and, moreover, n cannot be in the same MW as the root of T_1

If the above conditions obtain, then $move(\langle\langle T_1, S_1\rangle, \mu_1\rangle)$ is defined, and is equal to $\langle\langle T, S\rangle, \mu\rangle$, which is the result of moving the subtree rooted in the least node in the MW containing n, to the specifier position of the root. Note that since n is not in the same MW as the root, we must have that $n = x10^i$ for some $i \in \mathbb{N}$. The subtree we are to move is $\langle\langle T_2, S_2\rangle, \mu_1 \restriction T_2\rangle$, where $T_2 = \{y | x1y \in T_1\}$, and $S_2 = \{y | x1y \in S_1\}$. Then

- $T = 1T_2 \cup (T_1 - x1T_2)$
- $S = 1S_2 \cup (S_1 - x1S_2)$
- the label of the root of T is gotten from the label of the root of T_1 by removing the first syntactic feature
- the label of $10^i \in T$ is gotten from the label of $n = x10^i \in T_1$ by removing the first syntactic feature
- otherwise, for $z \in 1T_2$, $\mu(z) = \mu_1(xz)$, and for $z \in T - (1T_2)$, $\mu(z) = \mu_1(z)$

Given a MTG $G = \langle \Sigma, Syn, Lex, \{merge, move\}\rangle$, $L(G)$ denotes the closure of Lex under the structure building functions $merge$ and $move$. An expression is complete just in case the only node that has syntactic features is the root, and it has only a base feature. The string language of G at a category $b \in base$ ($L_b(G)$) is the set of the yields of the complete expressions whose root's unique syntactic feature is b. The mirror theoretic languages (MTLs) are the string languages of an MTG G at a category b, for some $G \in MTG$ and $b \in base_G$.

1.3. Derivations

An expression $e \in L(G)$ might have been built up in several ways from the generating functions. A derivation tree for an expression is a record of one possible sequence of steps taken to derive the expression in question from lexical items. Given a MTG G, we denote by $\Gamma(G)$ the set of all derivation trees for each expression in the language of G. $eval : \Gamma(G) \rightarrow L(G)$ is the map which takes each derivation of an expression to the expression it derives. We define $\Gamma(G)$ and $eval : \Gamma(G) \rightarrow L(G)$ by mutual recursion:

1. for each $\ell \in Lex_G$, $\ell \in \Gamma(G)$, and $eval(\ell) = \ell$
2. for $\gamma, \gamma' \in \Gamma(G)$, if $merge(eval(\gamma), eval(\gamma'))$ is defined, then $\gamma'' = \langle \gamma, \gamma' \rangle \in \Gamma(G)$ and $eval(\gamma'') = merge(eval(\gamma), eval(\gamma'))$
3. for $\gamma \in \Gamma(G)$, if $move(eval(\gamma))$ is defined, then $\gamma' = \langle \gamma \rangle \in \Gamma(G)$ and $eval(\gamma') = move(eval(\gamma))$

These structures are called derivation *trees* because it is simple to give them a (standard) tree-interpretation:

1. $\ell \in Lex_G$ denotes the tree with one node, labelled ℓ, and no branches.
2. $\langle \gamma, \gamma' \rangle \in \Gamma(G)$ denotes the tree with root labelled $\langle \gamma, \gamma' \rangle$, whose left child is the tree denoted by γ, and whose right child is the tree denoted by γ'
3. $\langle \gamma \rangle \in \Gamma(G)$ denotes the tree with root labelled $\langle \gamma \rangle$, and whose only child is the tree denoted by γ

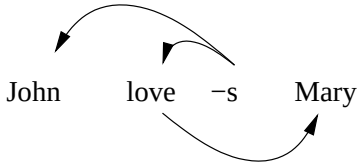
The sequence of lexical items used in a derivation γ is the yield of the tree denoted by γ, and, as shown in (Hale and Stabler, 2001), no two distinct derivations use the same sequence of lexical items.

2. Learning

Adapting a technique familiar from (Kanazawa, 1998) and others, we show that if the lexical ambiguity in target grammars is restricted, this can provide a basis for generalization from a finite sample. We describe an algorithm that identifies the class of languages generated by the rigid mirror theoretic grammars (rMTG) (grammars in which every lexical item has a unique string component) in the limit from any text of "dependency structures."

2.1. Dependancy Structures

Dependency structures show relations among the lexical items in a sentence. Information about these relations is, at least in many cases, plausibly available to the language learner (surface order, morphological decomposition and affixation (Baroni, 2000; Goldsmith, 2001) and selection relations (Siskind, 1996)). For example, imagine that upon hearing "John loves Mary" the language learner is able to infer these relations (Here, 's' is marked as a suffix (by the dash preceding it), and the arcs indicate that the source selected the target at some point in the derivation):



A dependency structure (henceforth: 'd-structure') is a tuple $\langle V, E, S, \mu, < \rangle$, where $\langle V, E \rangle$ is a directed multi-graph (i.e. $E \subseteq V \times V$ is a multi-set), $\mu : V \to \Sigma^*$ is a labeling function from vertices to phonetic strings, $S \subseteq V$ is a distinguished subset (of suffixes), and $<$ is a total ordering on V (the surface order). Intuitively, the vertices correspond to the lexical items used in a derivation, and there is one edge between two vertices for every pair of features, one from each of the two lexical items, such that the one checks the other in the course of the derivation. Formally, a d-structure d is 'for' a derivation γ just in case:[5]

1. for $s = \langle s_1, \ldots, s_n \rangle$ the sequence of lexical items used in γ, there is a sequence $v = \langle v_1, \ldots, v_n \rangle$ which enumerates without repetition the elements of V, and for $1 \leq i \leq n$, $\mu(v_i)$ is the string component of (the label of the lexical expression) s_i

2. there is a bijection from edges in E to non-leaf nodes in the derivation tree denoted by γ (or equivalently, to occurances of left brackets in $\gamma \ldots$) such that if edge $\langle v_i, v_j \rangle$ is mapped to γ', then γ' checks a syntactic feature in s_i against a syntactic feature in s_j

3. $v_i < v_j$ iff the phonetic features from s_i precede the phonetic features from s_j at spellout

$d(G) = \{d | \exists \gamma \in \Gamma(G)\ d \text{ is for } \gamma\}$ is the d-structure language of the MTG G.

Given a d-structure $\langle V, E, S, \mu, < \rangle$, we define the following notions which we will use in the description of the learning algorithm:

- vEv' just in case there is an edge from v to v'. We write vE^+v' in case there is a finite sequence of vertices $v_1, \ldots, v_{n+1}$ such that v_iEv_{i+1}, $v = v_1$, and $v' = v_{n+1}$.

- $<_E$ is a partial ordering of E such that $a <_E a'$ iff $a = \langle v, v' \rangle$, $a' = \langle v'', v' \rangle$ and $v''E^+v$

- $v <^1 v'$ (v immediately precedes v') iff $v < v'$ and no vertex follows v and precedes v'

- an arc $\langle v, v' \rangle$ is a $cmerge$ arc iff vEv', $v \in S$, and $v' <^1 v$

- an arc $\langle v, v' \rangle$ is a $move$ arc iff vEv', and $\exists v''\ vE^+v'' \& v''Ev'$

- an arc $\langle v, v' \rangle$ is a $smerge$ arc iff vEv', and it is neither a $cmerge$ arc nor a $move$ arc

- a vertex v is the surface specifier of a vertex v' iff $v'Ev$, $\neg\exists v''\ v''E^+v' \& v''Ev$, and $\langle v', v \rangle$ is not a $cmerge$ arc

- a vertex v has been shown weak iff there is a sequence $a_1, \ldots, a_n$ of $cmerge$ arcs such that $a_1 = \langle v, t_1 \rangle$, $a_i = \langle t_{i-1}, t_i \rangle$, and there is some v' such that $v' < v$ and v' is the surface specifier of t_n. Intuitively, v has been shown weak just in case it is pronounced after some specifier it mediately dominates.

5. There could be more than one derivation a d-structure is 'for' in any given MTG.

2.2. Learning

We work within the learning paradigm established in (Gold, 1967). There, a learner is a function from finite sequences of sentences to grammars for languages. When the learner is presented with a sequence of sentences, she makes a guess as to the language that these sentences are from (in the form of a grammar). A learner *converges* on an infinite sequence of sentences s just in case there is some finite i such that for all $j > i$, $\phi(s_1, \ldots, s_i)$ is (some variant of) the learner's guess on the sequence of the first i sentences in s is the same as her guess on the sequence of the first j sentences in s, namely, G. She *identifies* a language L (a set of sentences) *in the limit* iff on every infinite sequence enumerating the sentences of L she converges to some grammar G for L (possibly different grammars for different sequences). A learner ϕ identifies a class of languages $\mathcal{L}$ in the limit iff ϕ identifies every $L \in \mathcal{L}$ in the limit. The question we address here is whether the class of rigid d-structure languages ($d(rMTG) = \{d(G)|G \in rMTG\}$) is identifiable in the limit. Our result that this class is indeed indentifiable in the limit relies on a result by Angluin (1980) which shows that a class of languages $\mathcal{L}$ is identifiable in the limit iff for every $L \in \mathcal{L}$ there is a finite subset $D_L \subseteq L$ such that no other $L' \in \mathcal{L}$ can both contain D_L and be properly contained by L. We describe the construction of such a set for each $L \in d(rMTG)$, and show that it has these properties.

First we introduce some concepts that will help us in this section. A substitution is any total function θ over the set of base features that fixes the start category. A grammar G is an instance of a grammar G' ($G' \sqsubseteq G$) iff there is some substitution θ such that for each lexical expression ℓ in G', the result of applying the substitution to every feature in the label of ℓ (where θ 'commutes' with the complex features (e.g. $\theta(+b) = +(\theta(b))$) is some lexical item $\theta(\ell)$ in G. G and G' are alphabetic variants of one another ($G \square G'$) iff they are variants of each other. A grammar G is *reduced* in the sense of (Kanazawa, 1998) iff there is no G' such that $d(G') = d(G)$ and $G' \sqsubset G$. One way to think of this is to read the $\sqsubseteq$ relation as 'makes more category distinctions than' (in the sense of (the feature sequence)'$=a\ a$' makes fewer category distinctions than '$=a\ b$'). Then a grammar is reduced iff you can't make more category distinctions and still derive the same (d-structure) language. For example, in a reduced grammar no element $b \in base$ occurs as both a selector/base feature ($=b$, $b=$, b) and as a licensee/or feature ($+b$, $-b$). This is because movement and selection features never select the same feature. Thus, one can 'rename' all occurances of the selection features with distinct names without changing the expresivity of the grammar. In the remainder of this paper we will be focussing on reduced rigid mirror theoretic grammars (rrMTGs). This change of perspective serves to simplify discussion, and does not alter the class of languages to be learned.

The idea behind the construction of the sets D_L is to constrain as much as possible the grammars capable of generating supersets of D_L. As MTGs differ only in their lexical inventories, we do this by putting information about the lexicon of a grammar that generates L into D_L. Given a dependency structure d of a derivation γ in which lexical item ℓ occurs, we can reconstruct not only which types of features ℓ has ($sselect$, $cselect$, $licensor$, ...), but also the order in which they occur (so if the syntactic features of ℓ are $b=\ c\ -d\ -a$, we can determine that ℓ begins with a sselect feature of some sort, followed by some base feature, followed by two licensee features of some kind). To be able to determine *which* features of the given type ℓ has, we need to also add information about what other features each feature of ℓ can check/be checked by.

We associate with each rrMTG G a finite set $D_G \subseteq d(G)$, such that

1. for each lexical item ℓ in G, D_G contains a d-structure containing ℓ, if one exists
2. for each weak lexical item ℓ in G, D_G contains a d-structure which witnesses ℓ's weakness (a d-structure in which ℓ is shown weak), if one exists
3. for each selector or licensor feature $x(f)$ on every lexical item ℓ in G, for every feature f on each additional lexical item ℓ', D_G contains a d-structure of a derivation in which the feature $x(f)$ on ℓ checks f on ℓ', if one exists

Now we quickly outline a proof that the rigid MTGs are in fact learnable.

Lemma 1 Let $G, G' \in rrMTG$ such that $D_G \subseteq d(G') \subseteq d(G)$. Then the lexicons of G and G' are identical up to renaming of the syntactic features modulo the strength of their lexical items.

Proof Sketch: By the first clause in the definition of D_G, $Lex_{G'}$ and Lex_G have the same lexical items with respect to the string component, and the sequence of syntactic types, as $d(G') \supseteq D_G$, and $d(G) \supseteq d(G') \supseteq D_{G'}$. By the third clause in the definition of D_G, every feature of a lexical item has an example in D_G of every feature

it can combine with in the course of a derivation. As $D_G \subseteq d(G')$, G' must at least give the same feature to those elements which can combine with one another, and as $d(G') \subset d(G)$, G' must not unify category features more than $d(G)$. As G is reduced, it does not unify syntactic categories beyond what is recorded in D_G. □

Lemma 2 Given $G, G' \in rrMTG$ such that the lexicons of G and G' differ only in the strength they assign to the lexical item ℓ, if ℓ is not shown to be weak, then $d(G) = d(G')$.

Proof Sketch: By changing the strength of a node to strong, one ensures only that it is not pronounced after any surface specifiers of nodes further down in its morphological word, in any derivation. But as ℓ is never shown to be weak, there is no derivation γ where the surface specifier of a node in ℓ's MW which ℓ properly dominates, precedes ℓ at spellout. □

Theorem 1 Let $G \in rMTG$. For any $G' \in rMTG$, if $D_G \subseteq d(G')$, then $d(G')$ is not a proper subset of $d(G)$.

Proof Sketch: Let $D_G \subseteq d(G')$, and assume $d(G') \subseteq d(G)$. We show $d(G') = d(G)$. As $D_G \subseteq d(G') \subseteq d(G)$, the lexicons of G and G' are identical save possibly for the strength of their lexical items (Lemma 1). Let $\ell \in Lex_G$ and $\ell' \in Lex_{G'}$ be identical except perhaps for their strength. We show that neither ℓ nor ℓ' can be shown weak independantly of the other, and thus the lexicons of G and G' agree on any lexical items that are shown weak. The conclusion then follows from Lemma 2. If ℓ were shown weak in G, then some d-structure would be a witness to it in $D_G \subseteq d(G')$, whereby ℓ' would be shown weak as well in $d(G')$. If ℓ' were shown weak in G', then, as by hypothesis $d(G') \subseteq d(G)$, ℓ would also be shown weak in G. □

Corollary 1 The class of rigid mirror theoretic languages is identifiable in the limit from texts of dependency structures.

On a finite sequence $t = \langle d_1, \ldots, d_i \rangle$ of d-structures, our algorithm first constructs a 'general form' grammar, $GF(t)$, assigning to each phonetic string in each dependency structure a unique syntactic category sequence. This grammar is not normally rigid, and does not generalize (i.e. the language it guesses contains exactly the sentences it has seen). We then unify lexical items in $GF(t)$ to get a reduced rigid grammar, $RG(t)$.

The idea behind the learning algorithm is that, given a d-structure d, we can almost exactly reconstruct the derivation γ that d is of - we can not normally determine the strength of lexical items used in γ. Our learner, when determining the strength of a lexical expression, assumes it to be strong unless there is evidence to the contrary. During the unification process, if we have two lexical items which differ only in whether they are strong, we unify them as though they were both weak (as all weak features are data-driven).

Input: a sequence of d-structures $t = \langle d_1, \ldots, d_i \rangle$ of some rigid MTG G

1. let T be the base feature of the root of each of $d_1, \ldots, d_i$

2. for each non-root node we construct the base + post-base feature sequences, with one feature per incoming arc, as follows:

(a) with the head of the least incoming arc in we associate the feature f, where f is a new, unique base feature

(b) with the head of each subsequent incoming arc, in turn, we associate $-f$, where f is a new, unique base feature

3. we then construct the pre-base sequence, associating with the tail of each outgoing arc one feature as follows:

(a) if the head of the arc is associated with a feature $-f$, we associate the feature $+f$ with the tail

(b) if the head of the arc is associated with a feature f, then

- if the tail of the arc is a suffix, and if the head of the arc is ordered immediately before the tail by $<$, the arc is a $cmerge$ arc, and we associate the feature $=f$ with its tail
- otherwise, the arc is an $smerge$ arc, and associate the feature $f=$ with its tail

2. Collect the lexical categories from $d_1, \ldots, d_i$, making a lexical item weak if it has been shown so, and strong otherwise to get an MTG $GF(t)$

3. Unify the categories assigned to each vocabulary element to get a reduced rigid MTG, $RG(t)$, resolving strength conflicts in favour of weak features

References

Angluin, Dana. 1980. Inductive inference of formal languages from positive data. *Information and Control*, 45:117–135.
Baroni, Marco. 2000. Distributional Cues in Morpheme Discovery: A Computational Model and Empirical Evidence. UCLA, dissertation.
Brody, Michael. 1997. Mirror Theory. ms. University College London.
Chomsky, Noam. 1995. *The Minimalist Program*. Cambridge, Massachusetts: MIT Press.
Gold, E. Mark. 1967. Language identification in the limit. *Information and Control*, 10:447–474.
Goldsmith, John. 2001. Unsupervised Learning of the Morphology of a Natural Language. *Computational Linguistics*, 27:153–198.
Hale, John and Edward P. Stabler. 2001. Notes on Unique Readability. ms. UCLA.
Joshi, Aravind K. 1987. An Introduction to Tree Adjoining Grammars. In A. Manaster-Ramer, editor, *Mathematics of Language*. John Benjamins, Amsterdam.
Kanazawa, Makoto. 1998. *Learnable Classes of Categorial Grammars*. Stanford University.: CSLI Publications.
Kobele, Gregory M. forthcoming. Formalizing Mirror Theory. UCLA.
Michaelis, Jens. 2002. Notes on the complexity of complex heads in a minimalist grammar. In *Proceedings of the Sixth International Workshop on Tree Adjoining Grammars and Related Formalisms (TAG+6)*, Venezia.
Siskind, Jeffrey M. 1996. A Computational Study of Cross-Situational Techniques for Learning Word-to-Meaning Mappings. *Cognition*, 61:39–91.
Stabler, Edward P. 1997. Derivational minimalism. In Christian Retoré, editor, *Logical Aspects of Computational Linguistics*. Springer-Verlag (Lecture Notes in Computer Science 1328), NY, pages 68–95.