Idioms and extended transducers

Gregory M. Kobele
University of Chicago
kobele@uchicago.edu

Abstract

There is a tension between the idea that idioms can be both listed in the lexicon, and the idea that they are themselves composed of the lexical items which seem to inhabit them in the standard way. In other words, in order to maintain the insight that idioms actually contain the words they look like they contain, we need to derive them syntactically from these words. However, the entity that should be assigned a special meaning is then a derivation, which is not the kind of object that can occur in a lexicon (which is, by definition, the atoms of which derivations are built), and thus not the kind of thing that we are able to assign meanings directly to. Here I will show how to resolve this tension in an elegant way, one which bears striking similarities to those proposed by psychologists and psycholinguists working on idioms.

1 Introduction

One standard conception of the lexicon is that it is a set of form-meaning pairs, usually just the bare minimum needed to derive in a systematic way all the other form-meaning pairings constitutive of the language (Di Sciullo and Williams, 1987). Under this conception of the lexicon, an idiom, as its meaning is by definition not predictable given its form, must be a lexical item. Given that idioms are able to occur in syntactic environments where their parts are separated by arbitrarily large amounts of material, as in 1, it then follows, under this conception of the lexicon, that lexical items must (be able to) have complex internal structure, of the sort that is amenable to syntactic manipulation.

(1) *The cat* seems to have been *let out of the bag*.

This treatment of idioms is unsatisfying in the following way. It is natural to think that the word '*cat*' occurs in the idiom '*let the cat out of the bag*', and not just a synchronically unrelated string of phonemes /c/, /a/, /t/. However, as idioms are simply entered *en masse* into the lexicon, and are therefore not derived objects, they cannot be said to contain the words they look like they contain. In other words, the word '*cat*' occurs in '*let the cat out of the bag*' under this conception of the lexicon to the very same degree it occurs in '*catastrophe*' (i.e. not at all).

In order for the word '*cat*' to actually occur in the idiomatic expression '*let the cat out of the bag*', the idiomatic expression must have a derivation that uses the word '*cat*'.[1] But then it would seem that we must assign a non-predictable meaning to a non-lexical item.

There are thus two (mutually incompatible) roles that a lexicon plays. It is on one hand the set of syntactic building blocks, and on the other the repository of form-meaning pairings. In this paper I will show how the transductive ('two-step') approach to grammar (Morawietz, 2003) disentangles these two notions. The transductive approach to grammar defines the expressions generated in two steps: first, in terms of a set of derivation trees, and second, in terms of operations turning these derivation trees into the objects (sounds, and meanings) that they are the derivations of. We propose that the mapping between derivation tree and derived object be not a simple transduction, but rather an *extended* one in the sense of Graehl et al. (2008). These transducers implement exactly the kind of 'bounded subderivation' translation suggested by Shieber (1994). This ennables us to dissolve the tension between listing idioms on the one hand,

[1] This is true by definition: an expression A contains all and only those expressions that its immediate constituents contain, along with its immediate constituents. (Note that this implicitly involves identifying an expression with its derivation; we don't say of ambiguous expressions that they contain all the expressions that occur in any of their derivations, but rather that under one reading, the expression contains one set of expressions, and under another, another.)

and allowing them to be derived, on the other – they are derivationally complex, but interpretatively atomic.

Compare this approach with the general picture painted by Chomsky (1995):

> A language, in turn, determines an infinite set of linguistic expressions (SDs), each a pair $\langle \pi, \lambda \rangle$ drawn from the interface levels (PF,LF), respectively.

According to the two-step approach, each language determines an infinite set of linguistic expressions d, which are then mapped to interface interpretable elements π and λ by operations Π and Λ, respectively. Λ will be viewed as the repository of atomic syntax-meaning pairings, and Π as the repository of atomic syntax-form pairings, while these may make reference to syntactic atoms, they are defined over entire derivation trees, and thus can make reference to larger chunks than single nodes.

The remainder of this paper is structured as follows. In section 2 we present some formal preliminaries. Our proposal is presented in more detail in §3. Section 5 explores linguistic aspects of idioms from this perspective. Section 6 is the conclusion.

2 Background

Lambda terms provide a simple and uniform presentation of structured objects (trees, strings) and of mappings between them. Given a denumerably infinite set X of variables, and a set C of constants, the set of lambda terms over C is defined by the following grammar:

$$M ::= X \mid C \mid (MM) \mid \lambda X.M$$

The usual notions of β, η, and α reduction apply (Barendregt, 1984), and we do not distinguish between terms which are β, η, or α equivalent.

We are interested in lambda terms which can be (simply) typed. Given a finite set A (of atomic types), we define the set $\mathcal{T}(A)$ of types as the closure of A under pairing. When writing elements of $\mathcal{T}(A)$, we treat pairing as associating to the right; $(a,(b,c))$ will be written simply as abc. We write $t^0 := t$ and $t^{n+1} := tt^n$. The order of an atomic type is 1, and of a function type ab is the greater of $\mathsf{ord}(a)+1$ and $\mathsf{ord}(b)$. A linear Higher Order Signature (HOS) is a triple $\Sigma = \langle C, A, \tau \rangle$ where C and A are finite sets (of constants and atomic types, respectively), and $\tau : C \to \mathcal{T}(A)$ is a function assigning types over A to each $c \in C$. We adopt the notational convention that a HOS Σ_i consists of C_i, A_i, and τ_i. A HOS is of order n just in case the highest order type assigned to any constant is n.

$$\frac{}{x:\alpha \vdash_\Sigma x:\alpha} \qquad \frac{\Gamma \vdash_\Sigma M:\alpha\beta \quad \Delta \vdash_\Sigma N:\alpha}{\Gamma,\Delta \vdash_\Sigma (MN):\beta}$$

$$\frac{\Gamma,x:\alpha \vdash_\Sigma M:\beta}{\Gamma \vdash_\Sigma \lambda x.M:\alpha\beta} \qquad \frac{}{\vdash_\Sigma c:\tau(c)}$$

Figure 1: Deriving typing judgments

A typing context Γ is a finite map from variables to types. We write $x:\alpha$ to indicate the typing context defined only on x, mapping x to α. Given typing contexts Γ, Δ, their union Γ, Δ is defined iff they have disjoint domains (i.e. no variable is assigned a type in both Γ and Δ). A typing judgment $\Gamma \vdash_\Sigma M:\alpha$ indicates that M has type α in context Γ, with respect to HOS Σ. A typing judgment is derivable just in case it is licensed by the inference system in figure 1.[2] The language $\Lambda_\alpha(\Sigma)$ at type α of HOS Σ is defined to be the set of lambda terms which have the type α in the empty context ($\Lambda_\alpha(\Sigma) := \{M : \vdash_\Sigma M:\alpha\}$). We write $\Lambda(\Sigma)$ to denote the set of all well-typed lambda terms over Σ.

A tree is a first order term (i.e. a term of atomic type) over a second order HOS (which we will call a *tree signature*). Here a constant corresponds to a node of a tree, and its arguments (which are themselves trees) to its daughters. A tree context is a second order term over a second order HOS. A set of trees is regular iff it is the language $\Lambda_\alpha(\Sigma)$ at

[2] Because the rules for variables and constants require particular typing contexts, lambda bindings are necessarily non-vacuous. Furthermore, because contexts are finite maps, and context union is only defined over disjoint contexts, lambda bindings are necessarily linear.

some atomic type α of some tree signature Σ. The atomic types in this case correspond to states of a tree automaton, and a constant c of type $a_1 a_2 \cdots a_n a$ corresponds to a production $c(a_1, \ldots, a_n) \rightarrow a$. A string is a second order term over a second order HOS all of whose constants have type $\alpha\beta$ for some atomic types α, β (which we will call a *string signature*). As an example, $\lambda x.x$ represents the empty string, and $\lambda x.a(b(x))$ the string "ab". As before, the types correspond to states of an NFA, and the function type $\alpha\beta$ assigned to a constant c to a transition from α to β reading c (without ϵ transitions). A set of strings is regular iff it is the language $\Lambda_{\alpha\beta}(\Sigma)$ of some string signature Σ, for α, β atomic types (here α is the start state of the NFA, and β the (unique) final state).

A linear homomorphism $\mathcal{L}$ from HOS Σ_1 to HOS Σ_2 is a pair of maps $F : A_1 \rightarrow \mathcal{T}(A_2)$ and $G : C_1 \rightarrow \Lambda(\Sigma_2)$ such that for any $c \in C_1$, $\vdash_{\Sigma_2} G(c) : \hat{F}(\tau_1(c))$, where $\hat{F}$ is F extended pointwise over $\mathcal{T}(A_1)$; this simply demands that the lambda term a constant is mapped to has a type which is appropriately related to the type of the original constant. The order of a linear homomorphism $\mathcal{L}$ is the maximum order of the image under it of an atomic type.[3]

Our proposal is formalized using abstract categorial grammars (ACGs). An ACG (de Groote, 2001) $G = \langle \Sigma_1, \Sigma_2, \mathcal{L}, \alpha \rangle$ consists of a pair of higher order signatures Σ_1 (called its *abstract* vocabulary) and Σ_2 (its *concrete* vocabulary), a linear homomorphism $\mathcal{L}$ (called a *lexicon*) between them, and a designated type α in A_1. The *abstract language* of an ACG G is the set $\Lambda_\alpha(\Sigma_1)$, and its *concrete language* is the set $\mathcal{L}(\Lambda_\alpha(\Sigma_1)) := \{\mathcal{L}(M) : M \in \Lambda_\alpha(\Sigma_1)\}$. It will be convenient to depict ACGs graphically, as in figure 2.

We write $\mathcal{G}(m, n)$ for the set of ACGs G with m the order of its abstract vocabulary, and n the order of its lexicon. Thinking of the abstract vocabulary as the 'derivation structures', ACGs with a second order abstract vocabulary (i.e. those in $\mathcal{G}(2) := \bigcup_{n \in \mathbb{N}} \mathcal{G}(2, n)$) represent grammar formalisms with regular derivation tree languages (de Groote and Pogodalla, 2004). The set of concrete tree languages defined by $\mathcal{G}(2)$ is exactly the set of tree languages generated by hyperedge replacement grammars (Kanazawa, 2010), and the concrete string languages defined by $\mathcal{G}(2)$ is exactly the set of multiple context free languages (Salvati, 2007).

[3] $\texttt{ord}(\mathcal{L}) := \texttt{max}(\{\texttt{ord}(\mathcal{L}(a)) : a \in A_1\})$

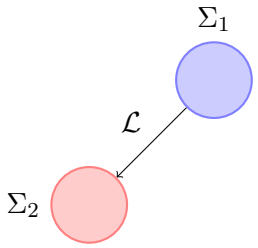

Figure 2: ACGs, graphically

3 Proposal

We focus on grammars with regular derivation tree languages; in particular tree adjoining grammars (Joshi, 1987) and minimalist grammars (Stabler, 1997).

The derived structures generated by a grammar in both of these grammar formalisms can be given in terms of a transducer of a particular sort acting on a regular set of derivation trees – in the case of tree adjoining grammars the transducer is a simple macro tree transducer (Shieber, 2006), and in the case of minimalist grammars it is a linear deterministic multi-bottom up tree transducer (Kobele et al., 2007; Mönnich, 2007).

Importantly, natural semantic analyses for both grammar formalisms can be given in terms of a similar tree transduction over the derivation (Nesson, 2009; Kobele, 2006), allowing a 'synchronous' representation whereby lexical items ℓ are represented as triples of the form $\langle h_1(\ell), \texttt{cat}(\ell), h_2(\ell) \rangle$, where h_1, h_2 are the derived structure and semantic structure transductions respectively, and $\texttt{cat}(\ell)$ is the relevant categorical information (the state the regular tree automaton recognizing the well-formed derivation trees is in upon scanning ℓ).[4] The 'standard' approach

[4] The second component, $\texttt{cat}(\ell)$, is only implicit in the standard presentations of synchronous TAGs,

to idioms in (synchronous) TAG is to enter them directly into the lexicon (Shieber and Schabes, 1990) – this amounts to introducing a new atomic lexical item ℓ_{idiom}, with (possibly complex) images under h_1 and h_2. While idiomatic expressions have not been handled in published work on MGs, this is also the obvious approach here too. While judicious choice of the derived structures associated with ℓ_{idiom} may allow for the ability of idiomatic material to be affected by syntactic operations, it does not capture the intuition that idioms actually contain the words it seems they do. To do this, we move to extended transductions (we discuss for simplicity homomorphisms: transductions without state). The kernel of an extended homomorphism is a finite relation between tree contexts: $H \subset T_\Sigma(X) \times T_\Delta(X)$. This kernel is thus the repository of non-predictable interpretations of derivation tree contexts. Intuitively, H will map $\mathsf{kick}(\mathsf{the}(\mathsf{bucket}))$ directly to `die`, as well as, indirectly via its components, to $\mathtt{kick}(\iota x.\mathtt{bucket}(x))$. Note that, in the case we are primarily interested in here in which only the semantic map is 'non-compositional', we have two objects which can be weighted – the set of lexical items, and the kernel of the semantic homomorphism. This is in line with the psycholinguistic findings (more on which in §5.2) (Titone and Connine, 1999) that (1) idioms seem to behave as though they are syntactically complex (weights over the lexicon) and (2) idioms themselves have different frequencies which subjects are sensitive to (weights over the extended transducers). In the remainder of this paper, we use abstract categorial grammars to work out this approach to idioms.

4 An ACG perspective

Viewing extended transducers from the perspective of abstract categorial grammars (ACGs) (de Groote, 2001) gives us a uniform way of visualizing this approach to idioms, one which explains the attraction of treating idioms as complex lexical items. Simply put, applying an extended transducer to a term t is the same as applying a non-extended transducer to the inverse homomorphic image of t under the map $h_{ex} : \Sigma \cup \Delta \to T_\Sigma(X)$, which maps elements of Σ to themselves, and elements of Δ to contexts over Σ. (Δ represents the idioms as atomic objects.) (Second order) ACGs provide a uniform notation for the macro and the multi bottom-up transducers mentioned above – both are defined in terms of a common set A of 'abstract λ-terms' which are mapped via homomorphism $\mathcal{L}$ to a set of 'concrete λ-terms' C. The differences between macro and multi bottom-up transductions are cashed out in terms of the homomorphisms and the nature of the concrete terms (de Groote and Pogodalla, 2004). Synchronous grammars are given in terms of two ACGs sharing the same abstract language A, but with different homomorphisms $\mathcal{L}_\Pi$ and $\mathcal{L}_\Lambda$ to different concrete languages C_1 (derived syntax) and C_2 (semantics) respectively as illustrated in figure 3 (cf. (Pogodalla, 2007)).

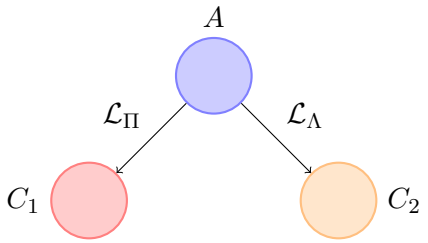

Figure 3: An ACG Perspective on Synchronous Grammars

An ACG representation of an extended transduction τ from the terms of HOS A to those of HOS C is obtained as follows. First, we create a HOS X, whose constants represent the left hand sides of the extended transducer rules, and a homomorphism $\mathcal{L}_X$ expanding constants in X to the terms over A which they represent. Then the right hand sides of the extended transducer rules are implemented via a homomorphism from X to C. In the degenerate case where τ is a non-extended transduction, A and X are the same and the map $\mathcal{L}_X$ is the identity function. This is shown in figure 4. As a concrete example, consider an extended (linear) bottom-up transducer t

e.g. (Shieber and Schabes, 1990) (but is partially indicated by the links).

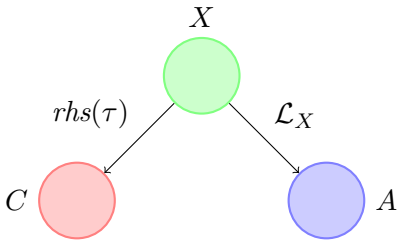

Figure 4: Representing an extended transducer τ

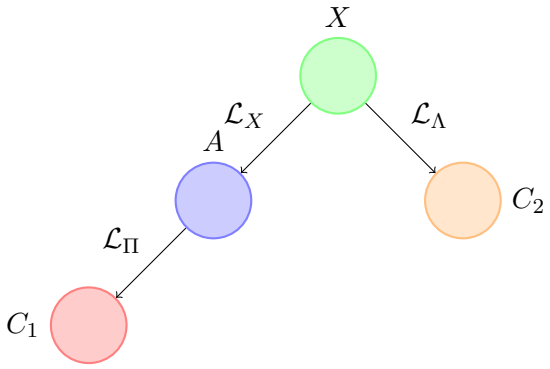

Figure 5: Semantically Extended Synchronous Grammars

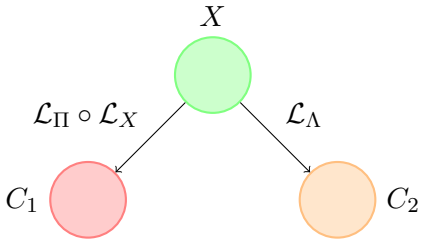

Figure 6: An ACG Perspective on Idioms as Lexical Items

from T_Σ to T_Δ. The HOS A has a single atomic type o, and constants $\sigma \in \Sigma$ of type $o^{\mathsf{rank}(\sigma)}$, similarly for C and Δ. The atomic types of the HOS X are the states of the transducer. For each rule $\rho = q(D[x_1, \ldots, x_n]) \to E[q_1(x_1), \ldots, q_n(x_n)]$, we have in X a constant c_ρ of type $q_1 \cdots q_n q$, whose image under $\mathcal{L}_X : X \to A$ is $\lambda x_1, \ldots, x_n.E(x_1) \cdots (x_n)$, and whose image under $\mathcal{L} : X \to C$ is $\lambda x_1, \ldots, x_n.D(x_1) \cdots (x_n)$.

From the synchronous perspective, it is natural to begin with the non-extended case, as shown in figure 3. Adding 'extension' on the semantic side, so as to describe idioms, we must introduce a new abstract language X, which is simply the original abstract language of derivation terms A with an extra atomic constant for each idiom. Then the semantic homomorphism $\mathcal{L}_\Lambda$ is defined from X to C_2, and the derived syntactic homomorphism is the composition of the mapping $\mathcal{L}_X$ from X to A, which maps elements of A to themselves, and idiomatic constants to the complex (derivational) objects they appear to be, and the mapping $\mathcal{L}_\Pi$ from A to C_1. Here we see that idioms are both lexical items (elements of X) and derived (complex objects in A). This is shown in figure 5.

Indeed, the standard approach to idioms, whereby they are simply lexical items, is obtained from ours by eliminating the HOS A by composing the maps $\mathcal{L}_X$ and $\mathcal{L}_\Pi$, as shown in figure 6. Distinguishing between A and X, as we propose here, does however have two (potential) benefits, as discussed in sections 4.1, where we propose a similar approach to morphological suppletion, and 5.2, where we suggest that our approach has some psychological plausibility.

4.1 Interface Uniformity

In decompositional (i.e. non-lexicalist) approaches to syntax, the notion of word-hood is divorced from the notion of syntactic terminal – words are conceived of as syntactically complex objects. In this sort of approach, an additional interface is needed to mediate between syntax and morphology, so as to permit suppletive forms (`go` + `Past` $\rightsquigarrow$ `went`). A natural idea is to take both the mapping from syntax to semantics and the one from syntax to morphology to be of the same kind; an extended transduction.

In our ACG setting, this amounts to introducing a new abstract language Y, and a new lexicon $\mathcal{L}_Y$ mapping terms in Y to terms in A. The lexicon Π from A to C_1, the concrete language of strings, is replaced by one from Y to C_1. This gives rise to a new 'w-shaped' way of combining ACGs, shown in figure 7. In the figure, we see that the derivation tree language is no longer directly mapped to

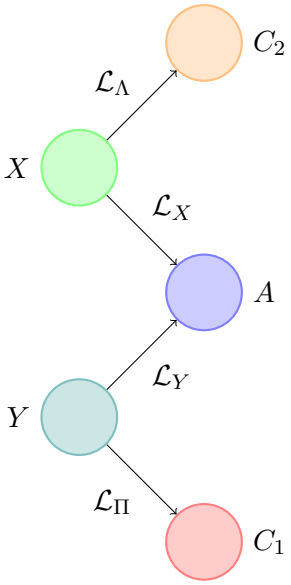

Figure 7: A w-shaped ACG

any derived structure language (and has become therefore a concrete language). Instead, it serves only to coordinate the abstract languages X and Y, making sure that the strings and meanings they produce are 'of the same derivation'. Note also that we are no longer able to eliminate the HOS A (and recover the standard approach to idioms outlined in figure 6) as the relation between X and C_1 (or between Y and C_2) is no longer functional.

Parsing in a w-shaped ACG is a matter of computing the inverse image of $\mathcal{L}_\Pi$, then its image under $\mathcal{L}_Y$, then its inverse image under $\mathcal{L}_X$, and then finally its image under $\mathcal{L}_\Lambda$; in other words, the composition $\mathcal{L}_\Lambda \circ \mathcal{L}_X^{-1} \circ \mathcal{L}_Y \circ \mathcal{L}_\Pi^{-1}$. Whereas application of a lambda homomorphism to a recognizable set of terms does not usually preserve recognizability, here the mappings $\mathcal{L}_X$ and $\mathcal{L}_Y$ are both first order (as they map atomic types to atomic types) and lexicalized (abstract constants are mapped to non-combinators), and thus do indeed preserve recognizability. Hence a regular set of strings (over the concrete HOS C_1) is mapped to a regular set of trees over the abstract HOS X.

The fact that the maps $\mathcal{L}_X$ and $\mathcal{L}_Y$ are first order distinguishes our proposal formally from the related one proposed by Dras (1999) in the context of textual paraphrase, which, from our perspective, introduces a sequence of ever more abstract languages, which are related to each other by maps of the same complexity as the maps $\mathcal{L}_\Pi$ and $\mathcal{L}_\Lambda$. (There it is described as deriving tree languages which are then interpreted as derivation tree languages and so on.)

5 Linguistics

5.1 Constraints on Idiomatic Structure

Linguists have formulated various constraints on possible idioms, in particular that dependents of a head can belong to an idiom only if the head belongs as well (Koopman and Sportiche, 1991; O'Grady, 1998). This constraint is naturally implemented in the present context by adopting a TAG-like perspective on derivations, whereby the operations of the grammar are left implicit, and the lexical items are treated as constants of higher rank. Then any first order subterm of a derivation tree satisfies this constraint. This move additionally rules out completely unlexicalized idioms and constructions.[5] This perspective on derivation trees can be easily adapted into the minimalist grammar framework, where the rank of each lexical item is the number of positive selector features[6] it has.[7] Although the set of well-formed minimalist derivation trees is not the algebra freely generated over this signature, it is a regular subset thereof.

5.1.1 Syntactic Permeability

Empirical 'puzzles' about idioms, such as their variable permeability to syntactic operations (Nunberg et al., 1994), must be dealt with by fine-tuning the syntactic anal-

[5] For example, $\lambda x, y.\mathbf{merge}(\mathbf{move}(y), z)$ is a first order term over a HOS for (standard presentations of) minimalist derivations. It could be the image of some idiom under the map $\mathcal{L}_S$.

[6] Recall that minimalist grammar categories are finite lists of syntactic features, which are either positive or negative versions of either licensing (for the **move** operation) or selection (for **merge**) feature types.

[7] It is also not necessary in the TAG framework, where one could treat elementary trees as nullary function symbols, and **adjoin** and **substitute** as binary ones.

ysis; canonical analyses of voice phenomena in MGs (as in (Kobele, 2006)) treat the voice head as taking a verb phrase as an argument – an idiom which is not passivizable must include the voice head, whereas one which is must not.

As an example, consider the idiom "*kick the bucket*" ('die'), which cannot be passivized.[8] Adopting the naïve formalization of standard transformational analyses from Kobele (2006), we could represent the idiomatic interpretation of this phrase by means of a unary abstract constant $c_{\textit{kick the bucket}}$, which is interpreted semantically as die, and derivationally as $\lambda x.\mathtt{act}(\mathtt{kick}(\mathtt{the}(\mathtt{bucket})))(x)$.[9] On the other hand, idioms like "*let the cat out of the bag*" ('reveal a secret') can be passivized (and then subject to raising, and other transformations). While there is no analysis of prepositions in Kobele (2006), we can treat *let* as a raising to object verb, and adapt a small clause analysis of the following form:[10]

$$[_V \textit{let}\ [_{sc}[_D \textit{the cat}]\ \textit{out}\ [_P \textit{of the bag}]]$$

We represent this in terms of a nullary abstract constant $c_{\textit{let cat out of bag}}$, which is semantically interpreted as $\lambda x.\mathsf{some}(\mathsf{secret})(\lambda y.\mathsf{reveal}(y)(x))$, and which is derivationally interpreted as $\mathtt{let}(\mathtt{out}(\mathtt{P}(\mathtt{of}(\mathtt{the}(\mathtt{bag})))))(\mathtt{the}(\mathtt{cat})))$

Finally, an idiom like "throw the book at" ('$\mathsf{punish\ severely}$') illustrates the need for constants corresponding not to subtrees, but to derivation tree contexts. Adopting again an overly simplistic approach to prepositions, we represent this passivizible idiom using the unary abstract constant $c_{\textit{throw book at}}$, which is semantically interpreted as $\lambda x.\mathsf{severely}(\mathsf{punish}(x))$, and is interpreted as the derivational term $\lambda x.\mathtt{Adj}(\mathtt{throw}(\mathtt{the}(\mathtt{book})))(\mathtt{P}(\mathtt{at}(x)))$.[11]

[8] This means that sentences like "*The bucket was kicked*" can have only a literal interpretation.

[9] Lexical entries:
⟨act, =V +k =d v⟩ ⟨kick, =d V⟩
⟨the, =n d -k -q⟩ ⟨bucket, n⟩

[10] Lexical entries:
⟨let, =sc V⟩ ⟨out, =p =d sc⟩
⟨of, =d +k +q P⟩ ⟨P, =P p⟩
⟨bag, n⟩ ⟨cat, n⟩

[11] Lexical entries:
⟨throw, =d V⟩ ⟨Adj, =V =p V⟩
⟨at, =d +k +q P⟩ ⟨P, =P p⟩
⟨book, n⟩ ⟨cat, n⟩

5.1.2 Transformationalism

In general, treating idioms via extended transducers (as done here) puts great pressure on the type system of the grammar formalism (if it is desired to have a single abstract constant representing the idiom in all of its glory). As one of the arguments for transformational analysis continues to be the variety of syntactic contexts which permits idiomatic interpretation, it is perhaps no surprise that the minimalist analyses sketched here go some ways in achieving the ideal of assigning a single type to the idiomatic abstract constant. However, the particular constraints on idiom shape in place here rule out certain otherwise reasonable transformational analyses, such as the implementation of the raising analysis of relative clauses in Kobele (2006).

5.2 Psycholinguistics

Adopting a 'levels' perspective on the relation between grammar and parser (Marr, 1982), a 'search' perspective on parsing still needs a way to order nodes in the agenda. A natural strategy here is to use weights assigned to lexical entries. Perhaps the main substantive difference between our proposal and 'standard' treatments of idioms is the availability of *two* 'lexica' – the HOSs A (for derivations) and X (for idioms) – for differential weighting (see figure 5). The standard treatment, according to which idioms are lexical items (as in figure 6), does not have access to the HOS A. We will need to make a linking theory precise, but, generally speaking, this setup will predict that idioms behave in certain respects as though they were indeed composed of the syntactic atoms it appears they contain. We claim that the literature on priming is consistent with this prediction (Cutting and Bock, 1997; Titone and Connine, 1999; Peterson et al., 2001; Sprenger et al., 2006; Konopka and Bock, 2009). Priming is the phenomenon whereby a prior exposure to some linguistic object token facilitates the comprehension of another token of that type, and increases the likelihood of producing a token of that type.

In order to account for the fact that words

can prime idioms which contain them, and vice versa, Sprenger et al. (2006) (following Cutting and Bock (1997)) propose that the mental lexicon is structured as a graph; the lexicon contains idioms, along with links to their constituent parts. Priming works by increasing the weights attached not only to a single node, but to all of its immediate neighbors. (And thus *kick* will prime the node kick as well as its neighbors kick the bucket, kick the can, ...) One defect of this account is that it does not explain *why* certain nodes are linked to others. Still, it bears an obvious resemblance to the ACG account presented here – its nodes are the elements of HOS X (which is the HOS A with extra constants for idioms), and the links are given by the lexicon $\mathcal{L}_X$.

First, a (sketch of a) naïve linking theory. We assume that priming effects are to be captured by updating weights on lexical items, and that the weights of all and only the lexical items in a successful parse are increased.[12] Under this view, priming effects come about due to an increased weight of the primed lexical item as a result of having used it previously in a successful parse. For our preliminary purposes here, we will say that a derivation d' primes derivation d (of sentence s) given grammar G just in case $\texttt{Weight}_{G_{d'}}(d) > \texttt{Weight}_G(d)$, where G_x is the result of updating the weights of lexical items in parse x (i.e. the weight of d is higher in the reweighted grammar than in the original grammar).

Now, taking the unique derivation d' of the sentence "John kicked Mary" as our primer, and the idiomatic derivation d of the sentence "Susan will kick the bucket" as our primee, we have that the weight of d in the 'idiomatic' HOS X is the same as in $X_{d'}$, but that the weight in the 'derivational' HOS A is less than in $A_{d'}$ (as the weight of the constant c_{kick} was increased after parsing d'). Taking the overall weight of a term t in G to be a monotonic function f of its weight in X and its weight in A, we have that d is primed.

[12] This is an 'off-line' implementation of priming effects, which neglects the role played by alternative parses of the sentence; Slevc and Ferreira (to appear) demonstrate that priming may occur from *failed parses*.

6 Conclusion

The noncompositional aspect of idiomatic expressions is most naturally dealt with in terms of extended transducers. The ACG perspective allows for a generalization to a wide variety of transducer types, and language families. In addition to being a principled formal account of idioms, the present story allows for a natural connection to the psycholinguistic data.

What we have not explained is the intuition that, in some idioms (such as *let the cat out of the bag*), certain parts of the idioms (e.g. *the cat*) seem to be associated with certain parts of the idiomatic meaning (e.g. *the secret*). That this intuition may in fact be deserving of an explanation is suggested by Horn (2003) (building on work by Nunberg et al. (1994)), who argues that this sort of 'semantic transparency' is a good predictor of whether the internal structure of an idiom can be subject to syntactic manipulation.

References

Hendrik Barendregt. 1984. *The Lambda Calculus: Its syntax and semantics*, volume 103 of *Studies in Logic and the Foundations of Mathematics*. Elsevier, Amsterdam.

Noam Chomsky. 1995. *The Minimalist Program*. MIT Press, Cambridge, Massachusetts.

J. Cooper Cutting and Kathryn Bock. 1997. That's the way the cookie bounces: Syntactic and semantic components of experimentally elicited idiom blends. *Memory and Cognition*, 25(1):57–71.

Philippe de Groote and Sylvain Pogodalla. 2004. On the expressive power of Abstract Categorial Grammars: Representing Context-Free formalisms. *Journal of Logic, Language and Information*, 13(4):421–438.

Philippe de Groote. 2001. Towards abstract categorial grammars. In *Association for Computational Linguistics, 39th Annual Meeting and 10th Conference of the European Chapter, Proceedings of the Conference*, pages 148–155.

Anna Maria Di Sciullo and Edwin Williams. 1987. *On the definition of word*, volume 14 of *Linguistic Inquiry Monographs*. MIT Press.

Mark Dras. 1999. *Tree Adjoining Grammar and the Reluctant Paraphrasing of Text*. Ph.D. thesis, Macquarie University.

Jonathan Graehl, Kevin Knight, and Jonathan May. 2008. Training tree transducers. *Computational Linguistics*, 34(3):391–427.

George M. Horn. 2003. Idioms, metaphors and syntactic mobility. *Journal of Linguistics*, 39:245–273.

Aravind K. Joshi. 1987. An introduction to tree adjoining grammars. In A. Manaster-Ramer, editor, *Mathematics of Language*. John Benjamins, Amsterdam.

Makoto Kanazawa. 2010. Second-order abstract categorial grammars as hyperedge replacement grammars. *Journal of Logic, Language and Information*, 19:137–161.

Gregory M. Kobele, Christian Retoré, and Sylvain Salvati. 2007. An automata theoretic approach to minimalism. In James Rogers and Stephan Kepser, editors, *Proceedings of the Workshop Model-Theoretic Syntax at 10; ESSLLI '07*, Dublin.

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

Agnieszka E. Konopka and Kathryn Bock. 2009. Lexical of syntactic control of sentence formulation? Structural generalizations from idiom production. *Cognitive Psychology*, 58:68–101.

Hilda Koopman and Dominique Sportiche. 1991. The position of subjects. *Lingua*, 85:211–258.

David Marr. 1982. *Vision*. W. H. Freeman and Company, New York.

Uwe Mönnich. 2007. Minimalist syntax, multiple regular tree grammars and direction preserving tree transductions. In James Rogers and Stephan Kepser, editors, *Proceedings of the Workshop Model-Theoretic Syntax at 10; ESSLLI '07*, Dublin.

Frank Morawietz. 2003. *Two-Step Approaches to Natural Language Formalisms*, volume 64 of *Studies in Generative Grammar*. Mouton de Gruyter.

Rebecca Nesson. 2009. *Synchronous and Multicomponenet Tree-Adjoining Grammars: Complexity, Algorithms and Linguistic Applications*. Ph.D. thesis, Harvard University.

Geoffrey Nunberg, Ivan A. Sag, and Thomas Wasow. 1994. Idioms. *Language*, 70(3):491–538.

William O'Grady. 1998. The syntax of idioms. *Natural Language and Linguistic Theory*, 16:279–312.

Robert R. Peterson, Curt Burgess, Gary S. Dell, and Kathleen M. Eberhard. 2001. Dissociation between syntactic and semantic processing during idiom comprehension. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, 27(5):1223–1237.

Sylvain Pogodalla. 2007. Generalizing a proof-theoretic account of scope ambiguity. In J. Geertzen, E. Thijsse, H. Bunt, and A. Schiffrin, editors, *Proceedings of the 7th International Workshop on Computational Semantics (IWCS)*, pages 154–165.

Sylvain Salvati. 2007. Encoding second order string acgs with deterministic tree walking transducers. In Shuly Wintner, editor, *Proceedings of FG 2006: the 11th conference on Formal Grammar*, pages 143–156. CSLI Publications.

Stuart M. Shieber and Yves Schabes. 1990. Synchronous tree-adjoining grammars. In *Proceedings of the 13th International Conference on Computational Linguistics (COLING)*, volume 3, pages 253–258, Helsinki,Finland.

Stuart M. Shieber. 1994. Restricting the weak-generative capacity of synchronous tree-adjoining grammars. *Computational Intelligence*, 10(4):371–385.

Stuart M. Shieber. 2006. Unifying synchronous tree-adjoining grammars and tree transducers via bimorphisms. In *Proceedings of the 11th Conference of the European Chapter of the Association for Computational Linguistics (EACL-2006)*, pages 377–384, Trento.

L. Robert Slevc and Victor S. Ferreira. to appear. To err is human; to structurally prime from errors is also human. *Journal of Experimental Psychology: Learning, Memory, and Cognition*. doi:10.1037/a0029525.

Simone A. Sprenger, Willem J. M. Levelt, and Gerard Kempen. 2006. Lexical access during the production of idiomatic phrases. *Journal of Memory and Language*, 54:161–184.

Edward P. Stabler. 1997. Derivational minimalism. In Christian Retoré, editor, *Logical Aspects of Computational Linguistics*, volume 1328 of *Lecture Notes in Computer Science*, pages 68–95. Springer-Verlag, Berlin.

Debra A. Titone and Cynthia M. Connine. 1999. On the compositional and noncompositional nature of idiomatic expressions. *Journal of Pragmatics*, 31:1655–1674.