Grounding As Learning

Gregory M. Kobele, Jason Riggle, Travis Collier, Yoosook Lee,
Ying Lin, Yuan Yao, Charles Taylor, Edward P. Stabler
University of California, Los Angeles

http://taylor0.biology.ucla.edu/al/

1 Grounding

Communication among agents requires (among many other things) that each agent be able to identify the semantic values of the generators of the language. This is the ”grounding” problem: how do agents with different cognitive and perceptual experiences successfully converge on common (or at least sufficiently similar) meanings for the language? There are many linguistic studies of how human learners do this, and also studies of how this could be achieved in robotic contexts (e.g., (Steels, 1996; Kirby, 1999)). These studies provide insight, but few of them characterize the problem precisely. In what range of environments can which range of languages be properly grounded by distributed agents? This paper takes a first step toward bringing the tools of formal language theory to bear on this problem. In the first place, these tools easily reveal a number of grounding problems which are simply unsolvable with reasonable assumptions about the evidence available, and some problems that can be solved. In the second place, these tools provide a framework for exploring more sophisticated grounding strategies (Stabler et al., 2003). We explore here some preliminary ideas about how hypotheses about syntactic structure can interact with hypotheses about grounding in a fruitful way to provide a new perspective on the emergence of recursion in language. Simpler grounding methods look for some kind of correlation between the mere occurrence of particular basic generators and semantic elements, but richer hypotheses about relations among the generators themselves can provide valuable additional constraints on the problem.

2 Learning Grounding

A first useful perspective on learning can be gained from the “identification in the limit” paradigm (Gold, 1967), a framework that is useful for identifying learning problems that are solvable (perfectly) when one makes very generous assumptions about the data potentially available to the learner. In this framework, the learner is successively presented with positive examples of a language, making a (possibly new) hypothesis after each example. Each possible order of presentation of every sentence of the language (repetitions allowed) is called a text (for that language). (Formally, a text is an infinite sequence $t \in L^\infty$ such that for every $s \in L$, there is some i such that $t_i = s$.) The learner learns the language if on each text there is a point after which the learner’s hypothesis never changes, and the hypothesis is correct.

We capitalize on the insight in (Siskind, 1996) that keeping track of the cooccurrance of morphemes and meaning atoms (sememes) can allow us to learn the morpheme-sememe association by a process of elimination. Extending this approach to morpheme to morpheme cooccurrence will allow the learner to extract coherent hypotheses from incomplete information and will allow us a foothold into bootstrapping syntax.

We take a language to be a set of sentence-meaning pairs. For our purposes, a sentence is a finite sequence of morphemes (i.e. $s \in \Sigma^*$), and a meaning is a multi-set of sememes ($m \subseteq M^{\mathbb{N}}$). We say a meaning map $\mu^* : \Sigma^* \rightarrow M^{\mathbb{N}}$ is compositional iff there is a map $\mu : \Sigma \rightarrow M^{\mathbb{N}}$ such that for $w \in \Sigma$, $\mu^*(w) = \mu(w)$, and for $s \in \Sigma^*$, $\mu^*(s) = \biguplus_{1 \leq i \leq |s|} \mu(s_i)$ (i.e. μ^* is the homomorphic extension of some μ). If such a map exists it is unique, and we will identify μ and μ^* when no confusion will arise. For $S \subseteq \Sigma^*$, maps μ^*, ν^* are S-equivalent ($\mu \approx_S \nu$) iff for all $s \in S, \mu^*(s) = \nu^*(s)$. We define $\mathcal{L}_S$ to be the set of all pairs $\langle S, \mu \rangle$, where μ is a compositional meaning map. We write $\mu \in \mathcal{L}_S$ for $\langle S, \mu \rangle \in \mathcal{L}_S$. Note that there might be many S-equivalent maps in $\mathcal{L}_S$. We say that $\mathcal{L}_S$ is exactly identifiable in the limit iff there is an algorithm A such that for any $\mu \in \mathcal{L}_S$, A converges to μ on any text from $L = \{\langle s, \mu^*(s) \rangle | s \in S\}$.

A straightforward adaptation of Siskind's cross-situational grounding algorithm to our setting is as follows. To each morpheme w in our lexicon is associated a set $P(w)$ (the *possible* meanings of w). Upon hearing a sentence meaning pair $\langle s, m_s \rangle$, for each morpheme $w = s_i$, if w is not already in the lexicon, it is added and its possible meanings are bounded only by m_s. Otherwise, if w is already in the lexicon, then its possible meanings are reduced to those which occur also in m_s.

Algorithm 1 On input $\langle s, m_s \rangle$

for each $w \in s$

if $w \in Lex$

$P(w) \leftarrow P(w) \cap m_s$

else

$P(w) \leftarrow m_s$

The first question we investigate is this: *if the criterion of learning is exact identification of a particular word to meaning mapping, under which circumstances is successful learning possible?*

Algorithm 1 works by eliminating a sememe from the possible meaning of a morpheme whenever a datum is presented that contains the morpheme without the sememe. This allows for a simple characterization of the language classes it identifies: they are those in which the meaning of each morpheme is exactly the set of sememes that constantly cooccur with the morpheme in the text.

Theorem 1 Algorithm 1 exactly identifies a class of languages $\mathcal{L}_S$ iff for all $\mu \in \mathcal{L}_S$, every $w \in \Sigma$ is such that $\mu(w) = \bigcap \{\mu^*(s) | s \in S \wedge w \in s\}$.

Proof: The **only if** direction follows immediately from the definition of the algorithm above. For the **if** direction, assume that for each w, $\mu(w) = \bigcap \{\mu^*(s) | s \in S \wedge w \in s\}$. Then as there are at most finitely many elements in any $\mu^*(s)$, there is a finite subset

$S_w \subseteq \{\mu^*(s) | s \in S \wedge w \in s\}$ such that $\bigcap S_w = \mu(w)$, and thus a finite point after which all elements of S_w have appeared in the text. As there are only finitely many w, there is a point in the text after which all elements of every S_w have been seen, and thus at this point Algorithm 1 will have converged on μ. □

Now that we have an exact characterization of the languages learnable by this algorithm, we can ask what kinds of languages these are. The next theorem provides a necessary syntactic condition on languages learnable by Algorithm 1; no morpheme may constantly co-occur with another. That is, there could be no morpheme *-ing* whose presence in a sentence entails the presence of another morpheme *be*.

Theorem 2 If every $\mu \in \mathcal{L}_S$ is such that for each $w \in \Sigma$, $\mu(w) = \bigcap\{\mu^*(s) | s \in S \wedge w \in s\}$, then for all w, $\bigcap_{s \in S}\{w' \in s | w \in s\} \subseteq \{w\}$.

Proof: Let $\mu \in \mathcal{L}_S$ be as in the statement of the theorem. Toward a contradiction, let w be such that $\bigcap_{s \in S}\{w' \in s | w \in s\} \supset \{w\}$. Then there is some z such that for any $s \in S$, $w \in s$ entails $z \in s$. Then by assumption, $\mu(w) \supseteq \mu(w) \uplus \mu(z)$, and so $\mu(z) = \emptyset$. However, as μ was arbitrary, we have shown that all $\nu \in \mathcal{L}_S$ map z to the empty set, which is a contradiction. □

Example 1 The following set of sentences gives rise to a set $\mathcal{L}_S$ of languages which are not exactly learnable using Algorithm 1 (by Theorem 2, as $\bigcap_{s \in S}\{w' \in s | w_2 \in s\} = \{w_1, w_2\}$):

$$S = \{w_1w_2, w_1w_3, w_1w_4, w_3w_4\}$$

Note that no two meaning maps in $\mathcal{L}_S$ are S-equivalent - because each of w_1, w_3 and w_4 occurs with the other, S-equivalent maps will agree on their meanings. But then there is no choice for the meaning of w_2.

Theorem 1 gives a precise characterization of the classes of languages which are exactly identifiable by Algorithm 1. However, Example 1 exhibited a simple, and not obviously unreasonable language which was unlearnable by Algorithm 1. Because Algorithm 1 does not keep track of when certain morphemes cooccur, it cannot use the successful resolution of the meaning of one morpheme to assist in resolving the meaning of another. We present below an algorithm which does exactly this. This will allow us to exactly identify any class $\mathcal{L}_S$ of languages with the property that no two meaning maps are S-equivalent.

A (partial) hypothesis is a (partial) function $h : \Sigma \rightarrow M^{\mathbb{N}}$. Given partial functions h, g, they are consistent (hRg) iff whenever both are defined, they agree (for all w, if $\downarrow h(w)$ and $\downarrow g(w)$, then $h(w) = g(w)$). We define a partial operation $\vee$ ('join') over partial functions such that $h \vee g$ is defined just in case hRg and, if defined, is their set theoretic union.

Given a multi-set M, $\Pi(M)$ is the set of partitions of M.

Algorithm 2 first constructs the set of partial hypotheses (defined only on morphemes present in the current sentence) which are consistent with the presented datum. Then each hypothesis already in the lexicon[1]is successively paired with each hypothesis in the newly constructed set. If this pairing of hypotheses is consistent, then their join is added to the lexicon.

[1]We are using the word 'lexicon' here to denote our set of working hypotheses. Once the learner converges on a language the lexicon will contain all and only S-equivalent hypotheses (Theorem 3). Of course, if no two maps are S-equivalent, then the learning will be exact.

Algorithm 2 On input $\langle s, m_s \rangle$

$T \leftarrow \emptyset$

$H \leftarrow \bigcup_{\pi \in \Pi(m_s)} \{h : \{s_1, \ldots, s_{|s|}\} \rightarrow \pi | m_s = h^*(s)\}$

if $Lex = \emptyset$

$Lex \leftarrow H$

else

for each $h \in Lex$

for each $g \in H$

if hRg

$T \leftarrow \{h \vee g\} \cup T$

$Lex \leftarrow T$

The following examples illustrate the behaviour of Algorithm 2.

Example 2 Imagine that the first piece of data a learner saw was $\langle w_1 w_1 w_2, \{0, 0, 2, 2\} \rangle$. Then every partial hypothesis which is consistent with this datum (there are exactly four) is in $H = \bigcup_{\pi \in \Pi(m_s)} \{h : \{s_1, \ldots, s_{|s|}\} \rightarrow \pi | m_s = \biguplus_{1 \leq i \leq |s|} h(s_i)\} = \{h_1, h_2, h_3, h_4\}$, where

$$\begin{array}{rcccl} h_1 & : & w_1 & \rightarrow & \{0, 2\} \\ & & w_2 & \rightarrow & \emptyset \end{array}$$

$$\begin{array}{rcccl} h_2 & : & w_1 & \rightarrow & \{0\} \\ & & w_2 & \rightarrow & \{2, 2\} \end{array}$$

$$\begin{array}{rcccl} h_3 & : & w_1 & \rightarrow & \{2\} \\ & & w_2 & \rightarrow & \{0, 0\} \end{array}$$

$$\begin{array}{rcccl} h_4 & : & w_1 & \rightarrow & \emptyset \\ & & w_2 & \rightarrow & \{0, 0, 2, 2\} \end{array}$$

As this is the first datum presented to the learner, $Lex = H = \{h_1, h_2, h_3, h_4\}$.

Example 3 Continuing from Example 2, imagine that the next datum presented to our learner was $\langle w_2 w_2 w_3, \{0, 0, 0, 0, 1\} \rangle$. Computing H, we find the only consistent maps to be h_5 and h_6:

$$\begin{array}{rcccl} h_5 & : & w_2 & \rightarrow & \{0, 0\} \\ & & w_3 & \rightarrow & \{1\} \end{array}$$

$$\begin{array}{rcccl} h_6 & : & w_2 & \rightarrow & \emptyset \\ & & w_3 & \rightarrow & \{0, 0, 0, 0, 1\} \end{array}$$

Now, as $Lex \neq \emptyset$, we proceed into the **for each** loop in Algorithm 2.

We begin by evaluating h_1 and h_5. As $\neg(h_1 R h_5)$ (because $h_1(w_2) = \emptyset \neq \{0,0\} = h_5(w_2)$), we go on to the next map, h_6. Since $h_1 R h_6$, we set $T = \{(h_1 \vee h_6)\}$, where

$$\begin{array}{rcccl} (h_1 \vee h_6) & : & w_1 & \rightarrow & \{0,2\} \\ & & w_2 & \rightarrow & \emptyset \\ & & w_3 & \rightarrow & \{0,0,0,0,1\} \end{array}$$

Continuing on, we find that the only other pair of maps to bear R to one other are h_3 and h_5. We set $T = \{(h_3 \vee h_5), (h_1 \vee h_6)\}$, where

$$\begin{array}{rcccl} (h_3 \vee h_5) & : & w_1 & \rightarrow & \{2\} \\ & & w_2 & \rightarrow & \{0,0\} \\ & & w_3 & \rightarrow & \{1\} \end{array}$$

Exiting the **for each** loops, we set $Lex = T = \{(h_3 \vee h_5), (h_1 \vee h_6)\}$.

Note that the size of Lex decreases monotonically throughout the course of grounding. Once Lex is initialized (upon seeing the first datum), each successive iteration reduces the number of distinct hypotheses stored. Thus the main computational cost of Algorithm 2 lies in computing H.

We are now ready to prove the main result of this paper:

Theorem 3 For any $S \subseteq \Sigma^*$, $\mathcal{L}_S$ is identifiable in the limit.

Proof: Let $S \subseteq \Sigma^*$, and $\mu \in \mathcal{L}_S$ be arbitrary. Let t be a text for $L = \{\langle s, \mu^*(s)\rangle | s \in S\}$. We show that at some point in the text t_n, every $h \in Lex_n$ is such that $h \approx_S \mu$, and that at every subsequent point, $Lex_n = Lex_{n+i}$.

Note that for every datum $\langle s, \mu^*(s)\rangle \in L$, there is some hypothesis $h \in H$ such that $\mu R h$. Note also that if $\mu R h$ and $\mu R g$, then $h R g$. This is due to the fact that the domain of μ includes the domains of h and of g. From this we conclude that at every step there is a hypothesis $h \in Lex$ such that $\mu R h$.

Now, after seeing a datum $\langle s, \mu^*(s)\rangle$, there will be a hypothesis $h \in Lex$ such that for every $1 \leq i \leq |s|$, $h(s_i) = \mu(s_i)$. Thus, after seeing a finite number of sentences (at most one for each $w \in \Sigma$)[2], $\mu \in Lex$. Similar reasoning tells us that at that point $\{\nu | \mu \approx_S \nu\} \subseteq Lex$.

Now, let $h \in Lex$ be such that $h \not\approx_S \mu$. Then there is some $s \in S$ such that $h^*(s) \neq \mu^*(s)$. $h \vee g$ will not be defined on any g which is such that $g^*(s) = \mu^*(s)$, and so once $\langle s, \mu^*(s)\rangle$ is seen, Lex will contain only hypotheses consistent with it (as H will, and at the next iteration Lex contains only those hypotheses which were the join of some hypothesis in H with some hypothesis already in Lex), and thus at no later point will inconsistent hypotheses enter into Lex. As there are finitely many meaning maps (each is uniquely defined by its behaviour on the finite set Σ), there are thus a finite number of sentences that need to be seen to eliminate them. At that point, $Lex = \{\nu | \mu \approx_S \nu\}$.

After $Lex = \{\nu | \mu \approx_S \nu\}$, there will be no sentence which will change Lex, as for any hypothesis $h \in H$, for any $\nu \in Lex$ such that $h R \nu$, $\nu \vee h = \nu$. □

Returning to the question we began with, namely, when it is possible to exactly identify a class of languages, Theorem 3 provides us immediately with the following answer:

Corollary 1 For any $S \subseteq \Sigma^*$, $\mathcal{L}_S$ is exactly identifiable in the limit iff for any $\mu, \mu' \in \mathcal{L}_S$, $\mu \approx_S \mu' \rightarrow \mu = \mu'$.

[2]This is assuming that for every $w \in \Sigma$, there is some $s \in S$ such that $w \in s$.

3 Grounding Syntax

The semantic representations used above are almost completely disconnected from the syntax of the language - they are attuned only to which morphemes occur and how many times they do so. This notion of the syntax-semantics interface does not allow semantic representations to provide any information about the syntax of the language beyond simple numerical relations between elements in sentences. Our definition of compositionality is a naïve approximation of the more common usage, whereby the mode of combination of the meainings of the parts of expressions is related to the abstract syntactic structure of those expressions.

Luckily for language learners, the situation in natural language is (perhaps) less difficult. If we were to assume (see e.g. (Fulop, 1999) and related work) that the semantics of expressions very closely mirrors their syntactic structure, then the syntactic learner would have an easier job of things once the grounding had taken place. Note, however, that this relies on an aspect of the grounding problem that we have been able to ignore up to this point due to our overly simple semantics. Namely, that the more structured the meanings, the more work the grounding algorithm need do. (Kanazawa, 2001) is an investigation of this problem in a type-logical setting. We will continue to ignore it here.

In minimalist grammars (Stabler, 1997), there are two basic dependencies between morphemes given by the two structure building operations, MERGE and MOVE. In the Chomskyian tradition (Chomsky, 1965; Chomsky, 1986), the dependencies given by merger are those which correspond most closely to predicate argument structures (movement is usually taken to have scopal effects). Enriching our semantic representations to reflect the predicate argument structures corresponding to merger dependencies[3] would result in a much more robust syntax-semantics interface. This in turn will allow the learner to use the results of grounding to tightly constrain its initial hypotheses about the syntactic structure of its language. If in addition we require that movement dependencies are evidenced by string displacement from merged positions, this would enable the learner to reconstruct the dependency structures from which syntactic learning can be shown to successfully take place (Kanazawa, 1998; Stabler, 2001; Kobele et al., 2002).

4 Extending Grounding

The previous sections detailed a very idealized perspective on the grounding problem. One assumption we made was that the learner was able to determine exactly the intended meaning of the utterance. We can relax this assumption by redefining a text for a language. A referentially uncertain text is one in which each sentence of the language is paired not only with its meaning, but also with other possible meanings. The algorithms of §2 may be extended to referentially uncertain texts with varying degrees of success, depending in part upon how we choose the incorrect meanings. Note that every text in the sense of last section can be viewed as a (degenerate) referentially uncertain text. For a trivial case, if every possible meaning always accompanies each sentence of the language, there is no way to determine the 'correct' (even up to S-equivalence) meaning map for the language. On the other hand, if the incorrect meanings are chosen in such a manner so as to preserve the fact that the only sememes that constantly cooccur with a morpheme are exactly the meaning of

[3] In minimalist grammars this is simply the yield of the derivation tree for a sentence (Hale and Stabler, 2001).

that morpheme, even Algorithm 1 will (exactly) identify this class of languages. Another assumption was that the level of analysis of the sentence was abstract enough so as to filter out any ambiguity in the language (i.e. instead of the ambiguous *bank*, there are $bank_1$ and $bank_2$). Of course, this does not seem to be the case in (the early stages of) natural language learning.

5 Grounding and Language Change

In the previous sections we were unconcerned with the efficiency of the learning algorithms. In particular, Algorithm 2 might require huge amounts of computational resources to compute all partial hypotheses consistent with a particular datum - this cost will only increase with referential uncertainty. We can bound the computational resources required by Algorithm 2 by, for example, limiting the size of H - data for which there are more than a certain number of consistent hypotheses might be ignored.[4] This restriction, while possibly comprimising the learnability theoretic properties of the system, introduces a new possibility for language change. This doesn't represent a selectional pressure, but does introduce a perturbation in the linguistic system which other selectional pressures might interact with to give rise to changes over time.

However, even without modifying the learning algorithms themselves, the bare fact that the learner is *not* given arbitrarily much time to identify the text it is faced with introduces the possibility of imperfect (incomplete) learning. This is exploited in the system for linguistic emergence and transmission described in (Stabler et al., 2003).

References

Chomsky, Noam. 1965. *Aspects of the Theory of Syntax*. MIT Press, Cambridge, Massachusetts.

Chomsky, Noam. 1986. *Knowledge of Language*. Praeger, NY.

Fulop, Sean. 1999. *On the Logic and Learning of Language*. Ph.D. thesis, University of California, Los Angeles.

Gold, E. Mark. 1967. Language identification in the limit. *Information and Control*, 10:447–474.

Hale, John and Edward P. Stabler. 2001. Notes on unique readability. ms. UCLA.

Kanazawa, Makoto. 1998. *Learnable Classes of Categorial Grammars*. CSLI Publications, Stanford University.

Kanazawa, Makoto. 2001. Learning word-to-meaning mappings in logical semantics. In R. van Rooy and M. Stokhof, editors, *Proceedings of the Thirteenth Amsterdam Colloquium*, pages 126–131. University of Amsterdam.

Kirby, Simon. 1999. Syntax of learning: The cultural evolution of structured communication in a population of induction algorithms. In D. Floreano, J-D. Nicoud, and F. Mondada, editors, *Advances in Artificial Life: 5th European Conference, ECAL'99*, Berlin. Springer-Verlag.

Kobele, Gregory M., Travis Collier, Charles Taylor, and Edward P. Stabler. 2002. Learning mirror theory. In *Proceedings of the Sixth International Workshop on Tree Adjoining Grammars and Related Frameworks (TAG+6)*, Venezia.

[4] There are other means of bounding resources - perhaps only the first n hypotheses formed for a sentence are used. Or perhaps if there are more than n unknown morphemes in a data point it is ignored, etc.

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.

Stabler, Edward P. 2001. Recognizing head movement. In Philippe de Groote, Glyn Morrill, and Christian Retoré, editors, *Logical Aspects of Computational Linguistics*, Lecture Notes in Artificial Intelligence, No. 2099. Springer, NY, pages 254–260.

Stabler, Edward P., Travis Collier, Gregory M. Kobele, Yoosook Lee, Ying Lin, Jason Riggle, Yuan Yao, and Charles Taylor. 2003. The learning and evolution of mildly context sensitive languages. In *European Conference on Artificial Life, ECAL'03*.

Steels, Luc. 1996. Synthesizing the origins of language and meaning using co-evolution, self-organisation and level formation. In J. Hurford, C. Knight, and M. Studdert-Kennedy, editors, *Evolution of Human Language*. Edinburgh University Press, Edinburgh, pages 161–165.