Incremental Grammar Induction from Child-Directed Dialogue Utterances[*]

Arash Eshghi
Interaction Lab
Heriot-Watt University
Edinburgh, United Kingdom
eshghi.a@gmail.com

Julian Hough and Matthew Purver
Cognitive Science Research Group
Queen Mary University of London
London, United Kingdom
{julian.hough, mpurver}@eecs.qmul.ac.uk

Abstract

We describe a method for learning an incremental semantic grammar from data in which utterances are paired with logical forms representing their meaning. Working in an inherently incremental framework, Dynamic Syntax, we show how words can be associated with probabilistic procedures for the incremental projection of meaning, providing a grammar which can be used directly in incremental probabilistic parsing and generation. We test this on child-directed utterances from the CHILDES corpus, and show that it results in good coverage and semantic accuracy, without requiring annotation at the word level or any independent notion of syntax.

1 Introduction

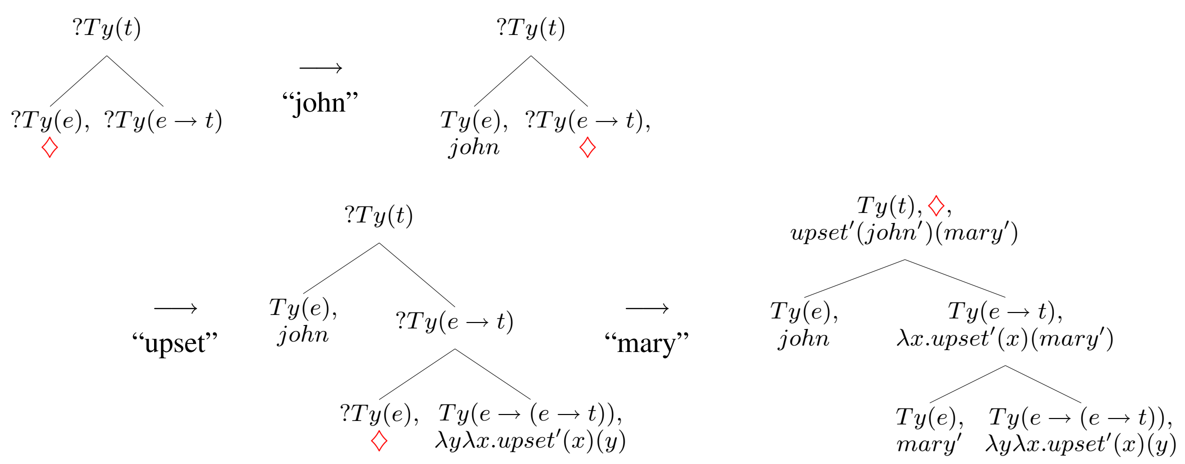
Human language processing has long been thought to function incrementally, both in parsing and production (Crocker et al., 2000; Ferreira, 1996). This incrementality gives rise to many characteristic phenomena in conversational dialogue, including unfinished utterances, interruptions and compound contributions constructed by more than one participant, which pose problems for standard grammar formalisms (Howes et al., 2012). In particular, examples such as (1) suggest that a suitable formalism would be one which defines grammaticality not in terms of licensing strings, but in terms of constraints on the *semantic* construction process, and which ensures this process is common between parsing and generation.

(1) A: I burnt the toast.
B: But did you burn ...
A: Myself? Fortunately not.
[where "did you burn myself?" if uttered by the same speaker is ungrammatical]

One such formalism is Dynamic Syntax (DS) (Kempson et al., 2001; Cann et al., 2005); it recognises no intermediate layer of syntax, but instead reflects grammatical constraints via constraints on the word-by-word incremental construction of meaning, underpinned by attendant concepts of underspecification and update.

Eshghi et al. (2013) describe a method for inducing a probabilistic DS lexicon from sentences paired with DS semantic trees (see below) representing not only their meaning, but their function-argument structure with fine-grained typing information. They apply their method only to an artificial corpus generated using a known lexicon. Here, we build on that work to induce a lexicon from real child-directed utterances paired with less structured Logical Forms in the form of TTR Record Types (Cooper, 2005), thus providing less supervision. By assuming only the availability of a small set of general compositional semantic operations, reflecting the properties of the lambda calculus and the logic of finite trees, we ensure that the lexical entries learnt include the grammatical constraints and corresponding compositional semantic structure of the language. Our method exhibits incrementality in two senses: incremental learning, with the grammar being extended and refined as each new sentence becomes available; resulting in an inherently incremental, probabilistic grammar for parsing and production, suitable for use in state-of-the-art incremental dialogue systems (Purver et al., 2011) and for modelling human-human dialogue.

[*] We are grateful to Ruth Kempson for her support and helpful discussions throughout this work. We also thank the CMCL'2013 anonymous reviewers for their constructive criticism. This work was supported by the EPSRC, RISER project (Ref: EP/J010383/1), and in part by the EU, FP7 project, SpaceBook (Grant agreement no: 270019).



?Ty(t)
?Ty(e), ?Ty(e → t)
"john"
Ty(e), john
?Ty(e → t)
"upset"
?Ty(e), Ty(e → (e → t)), λyλx.upset′(x)(y)
"mary"
Ty(t), upset′(john′)(mary′)
Ty(e → t), λx.upset′(x)(mary′)
Ty(e), mary′

Figure 1: Incremental parsing in DS producing semantic trees: *"John upset Mary"*

2 Background

2.1 Grammar Induction and Semantics

We can view existing grammar induction methods along a spectrum from supervised to unsupervised. Fully supervised methods take a parsed corpus as input, pairing sentences with syntactic trees and words with their syntactic categories, and generalise over the phrase structure rules to learn a grammar which can be applied to a new set of data. Probabilities for production rules sharing a LHS category can be estimated, producing a grammar suitable for probabilistic parsing and disambiguation e.g. a PCFG (Charniak, 1996). While such methods have shown great success, they presuppose detailed prior linguistic information and are thus inadequate as human grammar learning models. Fully unsupervised methods, on the other hand, proceed from unannotated raw data; they are thus closer to the human language acquisition setting, but have seen less success. In its pure form —positive data only, without bias— unsupervised learning is computationally too complex ('unlearnable') in the worst case (Gold, 1967). Successful approaches involve some prior learning or bias (see (Clark and Lappin, 2011)) e.g. a set of known lexical categories, a probability distribution bias (Klein and Manning, 2005) or a semi-supervised method with shallower (e.g. POS-tag) annotation (Pereira and Schabes, 1992).

Another point on the spectrum is *lightly* supervised learning: providing information which constrains learning but with little or no lexico-syntactic detail. One possibility is the use of *semantic* annotation, using sentence-level propositional Logical Forms (LF). It seems more cognitively plausible, as the learner can be said to be able to understand, at least in part, the meaning of what she hears from evidence gathered from (1) her perception of her local, immediate environment given appropriate biases on different patterns of individuation of entities and relationships between them, and (2) helpful interaction, and joint focus of attention with an adult (see e.g. (Saxton, 1997)). Given this, the problem she is faced with is one of separating out the contribution of each individual linguistic token to the overall meaning of an uttered linguistic expression (i.e. decomposition), while maintaining and generalising over several such hypotheses acquired through time as she is exposed to more utterances involving each token.

This has been successfully applied in Combinatorial Categorial Grammar (CCG) (Steedman, 2000), as it tightly couples compositional semantics with syntax (Zettlemoyer and Collins, 2007; Kwiatkowski et al., 2010; Kwiatkowski et al., 2012); as CCG is a lexicalist framework, grammar learning involves inducing a lexicon assigning to each word its syntactic and semantic contribution. Moreover, the grammar is learnt incrementally, in the sense that the learner collects data over time and does the learning sentence by sentence.

Following this approach, Eshghi et al. (2013) outline a method for inducing a DS grammar from semantic LFs. This brings an added dimension of incrementality: not only is learning sentence-by-sentence incremental, but the grammar learned is inherently word-by-word incremental (see section 2.2 below). However, their method requires a higher degree of supervision than (Kwiatkowski et al., 2012): the LFs assumed are not simply flat semantic formulae, but full DS semantic trees (see e.g. Fig. 1) containing information about the function-argument structure required for their composition, in addition to fine grained type and formula annotations. Further, they test their method only on artificial data created using a known, manually-specified DS grammar. In contrast, in this paper we provide an approach which can learn from LFs without any compositional structure information, and test it on real language data; thus providing the first practical learning system for an explicitly incremental grammar that we are aware of.

2.2 Dynamic Syntax (DS)

Dynamic Syntax (Kempson et al., 2001; Cann et al., 2005) is a parsing-directed grammar formalism, which models the word-by-word incremental processing of linguistic input. Unlike many other formalisms, DS models the incremental building up of *interpretations* without presupposing or indeed recognising an independent level of syntactic processing. Thus, the output for any given string of words is a purely *semantic* tree representing its predicate-argument structure; tree nodes correspond to terms in the lambda calculus, decorated with labels expressing their semantic type (e.g. $Ty(e)$) and formula, with beta-reduction determining the type and formula at a mother node from those at its daughters (Figure 1).

These trees can be *partial*, containing unsatisfied requirements for node labels (e.g. $?Ty(e)$ is a requirement for future development to $Ty(e)$), and contain a *pointer* $\diamondsuit$ labelling the node currently under development. Grammaticality is defined as parsability: the successful incremental construction of a tree with no outstanding requirements (a *complete* tree) using all information given by the words in a sentence. The complete sentential LF is then the formula decorating the root node – see Figure 1. Note that in these trees, leaf nodes do not necessarily correspond to words, and may not be in linear sentence order; syntactic structure is not explicitly represented, only the structure of semantic predicate-argument combination.

2.2.1 Actions in DS

The parsing process is defined in terms of conditional *actions*: procedural specifications for monotonic tree growth. These include general structure-building principles (*computational actions*), putatively independent of any particular natural language, and language-specific actions associated with particular lexical items (*lexical actions*). The latter are what we learn from data here.

Computational actions These form a small, fixed set, which we assume as given here. Some merely encode the properties of the lambda calculus and the logical tree formalism itself, LoFT (Blackburn and Meyer-Viol, 1994) – these we term *inferential* actions. Examples include THINNING (removal of satisfied requirements) and ELIMINATION (beta-reduction of daughter nodes at the mother). These actions are language-independent, cause no ambiguity, and add no new information to the tree; as such, they apply non-optionally whenever their preconditions are met.

Other computational actions reflect the fundamental predictivity and dynamics of the DS framework. For example, *-ADJUNCTION introduces a single *unfixed* node with underspecified tree position (replacing feature-passing or type-raising concepts for e.g. long-distance dependency); and LINK-ADJUNCTION builds a paired ("linked") tree corresponding to semantic conjunction (licensing relative clauses, apposition and more). These actions represent possible parsing strategies and can apply optionally whenever their preconditions are met. While largely language-independent, some are specific to language type (e.g. INTRODUCTION-PREDICTION in the form used here applies only to SVO languages).

Lexical actions The lexicon associates words with lexical actions; like computational actions, these are sequences of tree-update actions in an IF..THEN..ELSE format, and composed of explicitly procedural *atomic* tree-building actions such as `make` (creates a new daughter node), `go` (moves the pointer), and `put` (decorates the pointed node with a label). Figure 2 shows an example for a proper noun, *John*. The action checks whether the pointed node (marked as $\diamondsuit$) has a requirement for type e; if so, it decorates it with type e (thus satisfying the requirement), formula $John'$ and the bottom restriction $\langle\downarrow\rangle\bot$ (meaning that the node cannot have any daughters). Otherwise the action aborts, i.e. the word *'John'* cannot be parsed in the context of the current tree.

Graph-based Parsing & Generation These actions define the parsing process. Given a sequence of words $(w_1, w_2, ..., w_n)$, the parser starts from the *axiom* tree T_0 (a requirement to construct a complete propositional tree, $?Ty(t)$), and applies the corresponding lexical actions $(a_1, a_2, \ldots, a_n)$, optionally interspersing computational actions.

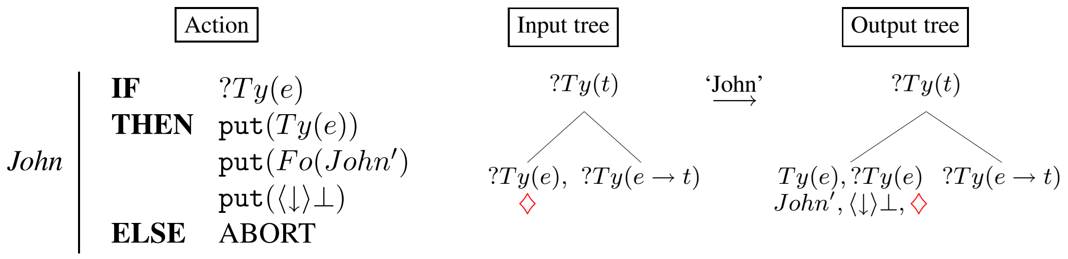


Figure 2: Lexical action for the word 'John'

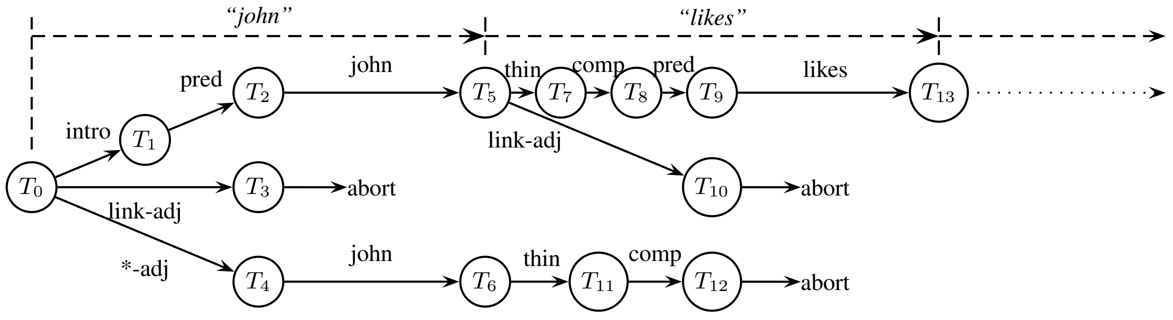


Figure 3: DS parsing as a graph: actions (edges) are transitions between partial trees (nodes).

This parsing process can be modelled as a directed acyclic graph (DAG) rooted at T_0, with partial trees as nodes, and computational and lexical actions as edges (i.e. transitions between trees) (Sato, 2011). Figure 3 shows an example: here, *intro*, *pred* and **adj* correspond to the computational actions INTRODUCTION, PREDICTION and *-ADJUNCTION respectively; and 'john' is a lexical action. Different DAG paths represent different parsing strategies, which may succeed or fail depending on how the utterance is continued. Here, the path $T_0 - T_3$ will succeed if *'John'* is the subject of an upcoming verb ("John upset Mary"); $T_0 - T_4$ will succeed if *'John'* turns out to be a left-dislocated object ("John, Mary upset").

This incrementally constructed DAG makes up the entire *parse state* at any point. The rightmost nodes (i.e. partial trees) make up the current maximal semantic information; these nodes with their paths back to the root (tree-transition actions) make up the *linguistic context* for ellipsis and pronominal construal (Purver et al., 2011). Given a conditional probability distribution $P(a|w,T)$ over possible actions a given a word w and (some set of features of) the current partial tree T, we can parse probabilistically, constructing the DAG in a best-first, breadth-first or beam parsing manner.

Generation uses exactly the same actions and structures, and can be modelled on the same DAG with the addition only of a *goal* tree; partial trees are checked for subsumption of the goal at each stage. The framework therefore inherently provides both parsing and generation that are word-by-word incremental and interchangeable, commensurate with psycholinguistic results (Lombardo and Sturt, 1997; Ferreira and Swets, 2002) and suitable for modelling dialogue (Howes et al., 2012). While standard grammar formalisms can of course also be used with incremental parsing or generation algorithms (Hale, 2001; Collins and Roark, 2004; Clark and Curran, 2007), their string-based grammaticality and lack of inherent parsing-generation interoperability means examples such as (1) remain problematic.

3 Method

Our task here is to learn an incremental DS grammar; following Kwiatkowski et al. (2012), we assume as input a set of sentences paired with their semantic LFs. Eshghi et al. (2013) outline a method for inducing DS grammars from semantic *DS trees* (e.g. Fig. 1), in which possible lexical entries are incrementally hypothesized, constrained by subsumption of the target tree for the sentence. Here, however, this structured tree information is not available to us; our method must therefore constrain hypotheses via compatibility with the sentential LF, represented as Record Types of Type Theory with Records (TTR).

3.1 Type Theory with Records (TTR)

Type Theory with Records (TTR) is an extension of standard type theory shown useful in semantics and dialogue modelling (Cooper, 2005; Ginzburg, 2012). It is also used for representing non-linguistic context such as the visual perception of objects (Dobnik et al., 2012), suggesting potential for embodied learning in future work.

Some DS variants have incorporated TTR as the semantic LF representation (Purver et al., 2011; Hough and Purver, 2012; Eshghi et al., 2012). Here, it can provide us with the mechanism we need to constrain hypotheses in induction by restricting them to those which lead to *subtypes* of the known sentential LF.

In TTR, logical forms are specified as *record types* (RTs), sequences of *fields* of the form $[\, l : T \,]$ containing a label l and a type T. RTs can be witnessed (i.e. judged true) by *records* of that type, where a record is a sequence of label-value pairs $[\, l = v \,]$, and $[\, l = v \,]$ is of type $[\, l : T \,]$ just in case v is of type T.

$$R_1 : \begin{bmatrix} l_1 & : T_1 \\ l_{2=a} & : T_2 \\ l_{3=p(l_2)} & : T_3 \end{bmatrix} \quad R_2 : \begin{bmatrix} l_1 & : T_1 \\ l_2 & : T_{2'} \end{bmatrix} \quad R_3 : []$$

Figure 4: Example TTR record types

Fields can be *manifest*, i.e. given a singleton type e.g. $[\, l : T_a \,]$ where T_a is the type of which only a is a member; here, we write this using the syntactic sugar $[\, l_{=a} : T \,]$. Fields can also be *dependent* on fields preceding them (i.e. higher) in the record type – see R_1 in Figure 4. Importantly for us here, the standard subtyping relation $\sqsubseteq$ can be defined for record types: $R_1 \sqsubseteq R_2$ if for all fields $[\, l : T_2 \,]$ in R_2, R_1 contains $[\, l : T_1 \,]$ where $T_1 \sqsubseteq T_2$. In Figure 4, $R_1 \sqsubseteq R_2$ if $T_2 \sqsubseteq T_{2'}$, and both R_1 and R_2 are subtypes of R_3.

Following Purver et al. (2011), we assume that DS tree nodes are decorated not with simple atomic formulae but with RTs, and corresponding lambda abstracts representing functions from RT to RT (e.g. $\lambda r : [\, l_1 : T_1 \,].[\, l_{2=r.l_1} : T_1 \,]$ where $r.l_1$ is a *path* expression referring to the label l_1 in r) – see Figure 5. The equivalent of conjunction for linked trees is now RT *extension* (concatenation modulo relabelling – see (Cooper, 2005; Fernández, 2006)). TTR's subtyping relation now allows a record type at the root node to be inferred for any partial tree, and incrementally further specified via subtyping as parsing proceeds (Hough and Purver, 2012).

We assume a field $head$ in all record types, with this corresponding to the DS tree node type. We also assume a neo-Davidsonian representation of

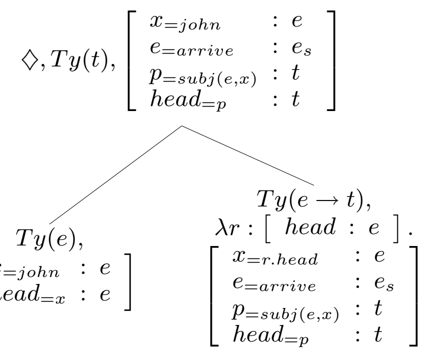

$\diamondsuit, Ty(t), \begin{bmatrix} x_{=john} & : e \\ e_{=arrive} & : e_s \\ p_{=subj(e,x)} & : t \\ head_{=p} & : t \end{bmatrix}$
$Ty(e), \begin{bmatrix} x_{=john} & : e \\ head_{=x} & : e \end{bmatrix}$
$Ty(e \rightarrow t), \lambda r : [\, head : e \,]. \begin{bmatrix} x_{=r.head} & : e \\ e_{=arrive} & : e_s \\ p_{=subj(e,x)} & : t \\ head_{=p} & : t \end{bmatrix}$

Figure 5: DS-TTR tree

predicates, with fields corresponding to the event and to each semantic role; this allows all available semantic information to be specified incrementally via strict subtyping (e.g. providing the $subj()$ field when subject but not object has been parsed) – see Figure 5 for an example.

3.2 Problem Statement

Our induction procedure now assumes as input:

- a known set of DS computational actions.
- a set of training examples of the form $\langle S_i, RT_i \rangle$, where $S_i = \langle w_1 \ldots w_n \rangle$ is a sentence of the language and RT_i – henceforth referred to as the *target RT* – is the record type representing the meaning of S_i.

The output is a grammar specifying the possible lexical actions for each word in the corpus. Given our data-driven approach, we take a probabilistic view: we take this grammar as associating each word w with a probability distribution θ_w over lexical actions. In principle, for use in parsing, this distribution should specify the posterior probability $p(a|w,T)$ of using a particular action a to parse a word w in the context of a particular partial tree T. However, here we make the simplifying assumption that actions are conditioned solely on one feature of a tree, the semantic type Ty of the currently pointed node; and that actions apply exclusively to one such type (i.e. ambiguity of type implies multiple actions). This simplifies our problem to specifying the probability $p(a|w)$.

In traditional DS terms, this is equivalent to assuming that all lexical actions have a simple IF clause of the form IF $?Ty(X)$; this is true of most lexical actions in existing DS grammars (see Fig. 2), but not all. Our assumption may therefore lead to over-generation – inducing actions which can parse some ungrammatical strings – we must rely on the probabilities learned to make such parses unlikely, and evaluate this in Section 4. Given this, our focus here is on learning the THEN clauses of lexical actions: sequences of DS atomic actions such as *go*, *make*, and *put* (Fig. 2), but now with attendant posterior probabilities. We will henceforth refer to these sequences as *lexical hypotheses*. We first describe how we construct lexical hypotheses from individual training examples; we then show how to generalise over these, while incrementally estimating corresponding probability distributions.

3.3 Hypothesis construction

DS is *strictly monotonic*: actions can only *extend* the current (partial) tree T_{cur}, deleting nothing except satisfied requirements. Thus, we can hypothesise lexical actions by incrementally exploring the space of all monotonic, well-formed extensions T of T_{cur}, whose maximal semantics R is a supertype of (extendible to) the target R_T (i.e. $R \sqsubseteq R_T$). This gives a bounded space described by a DAG equivalent to that of section 2.2.1: nodes are trees; edges are possible extensions; paths start from T_{cur} and end at any tree with LF R_T . Edges may be either known computational actions or new *lexical hypotheses*. The space is further constrained by the properties of the lambda-calculus and the modal tree logic LoFT (not all possible trees and extensions are well-formed).[1]

Hypothesising increments In purely semantic terms, the hypothesis space at any point is the possible set of TTR increments from the current LF R to the target R_T. We can efficiently compute and represent these possible increments using a *type lattice* (see Figure 6),[2] which can be constructed for the whole sentence before processing each training example. Each edge is a RT R representing an *increment* from one RT, R_j, to another, R_{j+1}, such that $R_j \wedge R_I = R_{j+1}$ (where $\wedge$ represents record type intersection (Cooper, 2005)); possible parse DAG paths must correspond to some path through this lattice.

Hypothesising tree structure These DAG paths can now be hypothesised with the lattice as a constraint: hypothesising possible sequences of ac-

$$\begin{tikzcd}
 & R_i : [\,] \arrow[dl] \arrow[d] \arrow[dr] & \\
R_{11} : \begin{bmatrix} a : b \end{bmatrix} \arrow[d] \arrow[dr] & R_{12} : \begin{bmatrix} c : d \end{bmatrix} \arrow[dl] \arrow[dr] & R_{12} : \begin{bmatrix} e : f \end{bmatrix} \arrow[dl] \arrow[d] \\
R_{21} : \begin{bmatrix} a : b \\ c : d \end{bmatrix} \arrow[dr] & R_{22} : \begin{bmatrix} a : b \\ e : f \end{bmatrix} \arrow[d] & R_{22} : \begin{bmatrix} c : d \\ e : f \end{bmatrix} \arrow[dl] \\
 & R_T : \begin{bmatrix} a : b \\ c : d \\ e : f \end{bmatrix} &
\end{tikzcd}$$

Figure 6: RT extension hypothesis lattice

tions which extend the tree to produce the required semantic increment, while the increments themselves constitute a search space of their own which we explore by traversing the lattice.

The lexical hypotheses comprising these DAG paths are divide into two general classes: (1) *tree-building* hypotheses, which hypothesise appropriately typed daughters to compose a given node; and (2) *content* hypotheses, which decorate leaf nodes with appropriate formulae from R_i (non-leaf nodes then receive their content via beta-reduction/extension of daughters).

Tree-building can be divided into two general options: functional decomposition (corresponding to the addition of daughter nodes with appropriate types and formulae which will form a suitable mother node by beta-reduction); and type extension (corresponding to the adjunction of a linked tree whose LF will extend that of the current tree, see Sec. 3.1 above). The availability of the former is constrained by the presence of suitable *dependent* types in the LF (e.g. in Fig. 5, $p = subj(e, x)$ depends on the fields with labels x and e, and could therefore be hypothesised as the body of a function with x and/or e as argument). The latter is more generally available, but constrained by sharing of a label between the resulting linked trees.

Figure 7 shows an example: a template for functional decomposition hypotheses, extending a node with some type requirement $?Ty(X)$ with daughter nodes which can combine to satisfy that requirement – here, of types Y and $Y \rightarrow X$. Specific instantiations are limited to a finite set of types: e.g. $X = e \rightarrow t$ and $Y = e$ is allowed, but higher types for Y are not. We implement these constraints by packaging together permitted sequences of tree updates as macros, and using these macros to hypothesise DAG paths commensurate with the lattice.

Finally, semantic content decorations (as se-

[1]We also prevent arbitrary type-raising by restricting the types allowed, taking the standard DS assumption that noun phrases have semantic type e (rather than a higher type as in Generalized Quantifier theory) and common nouns their own type cn, see Cann et al. (2005), chapter 3 for details.

[2]Clark (2011) similarly use a *concept* lattice relating strings to their contexts in syntactic grammar induction.

```
IF      $?Ty(X)$
THEN    $make(\langle\downarrow_0\rangle); go(\langle\downarrow_0\rangle)$
        $put(?Ty(Y)); go(\langle\uparrow\rangle)$
        $make(\langle\downarrow_1\rangle); go(\langle\downarrow_1\rangle)$
        $put(?Ty(Y \rightarrow X)); go(\uparrow)$
ELSE    ABORT
```

Figure 7: Tree-building hypothesis

quences of `put` operations) are hypothesised for the leaf nodes of the tree thus constructed; these are now determined entirely by the tree structure so far hypothesised and the target LF R_T.

3.4 Probabilistic Grammar Estimation

This procedure produces, for each training sentence $\langle w_1 \ldots w_n\rangle$, all possible sequences of actions that lead from the axiom tree T_0 to a tree with the target RT as its semantics. These must now be split into n sub-sequences, hypothesising a set of word boundaries to form discrete word hypotheses; and a probability distribution estimated over this (large) word hypothesis space to provide a grammar that can be useful in parsing. For this, we apply the procedure of Eshghi et al. (2013).

For each training sentence $S = \langle w_1 \ldots w_n\rangle$, we have a set HT of possible *Hypothesis Tuples* (sequences of word hypotheses), each of the form $HT_j = \langle h_1^j \ldots h_n^j\rangle$, where h_i^j is the word hypothesis for w_i in HT_j. We must estimate a probability distribution θ_w over hypotheses for each word w, where $\theta_w(h)$ is the posterior probability $p(h|w)$ of a given word hypothesis h being used to parse w. Eshghi et al. (2013) define an incremental version of Expectation-Maximisation (Dempster et al., 1977) for use in this setting.

Re-estimation At any point, the Expectation step assigns each hypothesis tuple HT_j a probability based on the current estimate θ'_w:

$$p(HT_j|S) = \prod_{i=1}^{n} p(h_i^j|w_i) = \prod_{i=1}^{n} \theta'_{w_i}(h_i^j) \tag{2}$$

The Maximisation step then re-estimates $p(h|w)$ as the normalised sum of the probabilities of all observed tuples HT_j which contain h, w:

$$\theta''_w(h) = \frac{1}{Z} \sum_{\{j|h,w\in HT_j\}} \prod_{i=1}^{n} \theta'_{w_i}(h_i^j) \tag{3}$$

where Z is the appropriate normalising constant summed over all the HT_j's.

Incremental update The estimate of θ_w is now updated incrementally at each training example: the new estimate θ_w^N is a weighted average of the previous estimate θ_w^{N-1} and the new value from the current example θ''_w from equation (3):

$$\theta_w^N(h) = \frac{N-1}{N}\theta_w^{N-1}(h) + \frac{1}{N}\theta''_w(h) \tag{4}$$

$$\lambda e.not(aux|do(v|have(pro|he, det|a(x, n|hat(x)), e), e), e)$$
$$\downarrow$$
$$\left[\begin{array}{ll} e_{=have} & : es \\ p3_{=not(e)} & : t \\ p2_{=do\text{-}aux(e)} & : t \\ r & : \left[\begin{array}{ll} x & : e \\ p_{=hat(x)} & : t \\ head_{=x} & : e \end{array}\right] \\ x2_{=\epsilon(r.head,r)} & : e \\ x1_{=he} & : e \\ p1_{=object(e,x2)} & : t \\ p_{=subject(e,x1)} & : t \\ head_{=e} & : es \end{array}\right]$$

Figure 8: Conversion of LFs from FOL to TTR.

For the first training example, a uniform distribution is assumed; when subsequent examples produce new previously unseen hypotheses these are assigned probabilities uniformly distributed over a held-out probability mass.

4 Experimental Setup

Corpus We tested our approach on a section of the Eve corpus within CHILDES (MacWhinney, 2000), a series of English child-directed utterances, annotated with LFs by Kwiatkowski et al. (2012) following Sagae et al. (2004)'s syntactic annotation. We convert these LFs into semantically equivalent RTs; e.g. Fig 8 shows the conversion to a record type for "He doesn't have a hat".

Importantly, our representations remove all part-of-speech or syntactic information; e.g. the *subject*, *object* and *indirect object* predicates function as purely semantic role information expressing an event's participants. This includes e.g. $do\text{-}aux(e)$ in (8), which is taken merely to represent temporal/aspectual information about the event, and could be part of any word hypothesis.

From this corpus we selected 500 short utterance-record type pairs. The minimum utterance length in this set is 1 word, maximum 7, mean 3.7; it contains 1481 word tokens of 246 types, giving a type:token ratio of 6.0). We use the first 400 for training and 100 for testing; the test set also has a mean utterance length of 3.7 words, and contains only words seen in training.

Evaluation We evaluate our learner by comparing the record type semantic LFs produced using the induced lexicon against the gold standard LFs, calculating precision, recall and f-score using a method similar to Allen et al. (2008).

	Coverage %	Precision	Recall	F-Score
Top-1	59	0.548	0.549	0.548
Top-2	85	0.786	0.782	0.782
Top-3	92	0.854	0.851	0.851

Table 1: Results: parse coverage & accuracy using the top N hypotheses induced in training.

Each field has a potential score in the range [0,1]. A method $maxMapping(R_1, R_2)$ constructs a mapping from fields in R_1 to those in R_2 to maximise alignment, with fields that map completely scoring a full 1, and partially mapped fields receiving less, depending on the proportion of the R_1 field's representation that subsumes its mapped R_2 field;e.g. a unary predicate field in $RT2$ such as $\left[\ p_{=there(e)}\ :\ t\ \right]$ could score a maximum of 3 - 1 for correct type t, 1 for correct predicate $there$ and 1 for the subsumption of its argument e; we use the total to normalise the final score. The potential maximum for any pair is therefore the number of fields in R_1 (including those in embedded record types). So, for hypothesis H and goal record type G, with N_H and N_G fields respectively:

(5) $precision = maxMapping(H, G)/N_H$
$recall = maxMapping(H, G)/N_G$

5 Results

Table 1 shows that the grammar learned achieves both good parsing coverage and semantic accuracy. Using the top 3 lexical hypotheses induced from training, 92% of test set utterances receive a parse, and average LF f-score reaches 0.851.

We manually inspected the learned lexicon for instances of ambiguous words to assess the system's ability to disambiguate (e.g. the word *''s'* (is) has three different senses in our corpus: (1) auxiliary, e.g. *"the coffee's coming"*; (2) verb predicating NP identity, e.g. *"that's a girl"*; and (3) verb predicating location, e.g. *"where's the pencil"*). From these the first two were in the top 3 hypotheses (probabilities p=0.227 and p=0.068). For example, the lexical entry learned for (2) is shown in Fig. 9.

However, less common words fared worse: e.g. the double object verb *'put'*, with only 3 tokens, had no correct hypothesis in the top 5. Given sufficient frequency and variation in the token distributions, our method appears successful in inducing the correct incremental grammar. However, the complexity of the search space also limits the possibility of learning from larger record types, as the space of possible subtypes used for hypothesising tree structure grows exponentially with the number of fields in the type. Therefore, when learning from longer, more complicated sentences, we may need to bring in further sources of bias to constrain our hypothesis process further (e.g. learning from shorter sentences first).

```
IF      $?Ty(e \rightarrow t)$
THEN    $\mathtt{make}(\langle\downarrow_0\rangle); \mathtt{go}(\langle\downarrow_0\rangle)$
        $\mathtt{put}(?Ty(e))$
        $\mathtt{go}(\langle\uparrow_0\rangle)$
        $\mathtt{make}(\langle\downarrow_1\rangle); \mathtt{go}(\langle\downarrow_1\rangle)$
        $\mathtt{put}(Ty(e \rightarrow (e \rightarrow t)))$
        $\mathtt{put}(Fo($
        $\lambda r1 : \left[\ head\ :\ e\ \right]$
        $\lambda r2 : \left[\ head\ :\ e\ \right].$
        $\left[\begin{array}{ll} x1_{=r1.head} & : e \\ x2_{=r2.head} & : e \\ e_{=eq} & : e_s \\ p1_{=subj(e,x2)} & : t \\ p2_{=obj(e,x1)} & : t \\ head_{=e} & : t \end{array}\right]))$
        $\mathtt{put}(\langle\downarrow\rangle\bot)$
ELSE    ABORT
```

Figure 9: Action learned for second sense of 'is'

6 Conclusions

We have outlined a novel method for the induction of a probabilistic grammar in an inherently incremental and semantic formalism, Dynamic Syntax, compatible with dialogue phenomena such as compound contributions and with no independent level of syntactic phrase structure. Assuming only general compositional mechanisms, our method learns from utterances paired with their logical forms represented as TTR record types. Evaluation on a portion of the CHILDES corpus of child-directed dialogue utterances shows good coverage and semantic accuracy, which lends support to viewing it as a plausible, yet idealised, language acquisition model.

Future work planned includes refining the method outlined above for learning from longer utterances, and then from larger corpora e.g. the Groningen Meaning Bank (Basile et al., 2012), which includes more complex structures. This will in turn enable progress towards large-scale incremental semantic parsers and allow further investigation into semantically driven language learning.

References

James F. Allen, Mary Swift, and Will de Beaumont. 2008. Deep Semantic Analysis of Text. In Johan Bos and Rodolfo Delmonte, editors, *Semantics in Text Processing. STEP 2008 Conference Proceedings*, volume 1 of *Research in Computational Semantics*, pages 343–354. College Publications.

Valerio Basile, Johan Bos, Kilian Evang, and Noortje Venhuizen. 2012. Developing a large semantically annotated corpus. In *Proceedings of the Eight International Conference on Language Resources and Evaluation (LREC 2012)*, pages 3196–3200, Istanbul, Turkey.

Patrick Blackburn and Wilfried Meyer-Viol. 1994. Linguistics, logic and finite trees. *Logic Journal of the Interest Group of Pure and Applied Logics*, 2(1):3–29.

Ronnie Cann, Ruth Kempson, and Lutz Marten. 2005. *The Dynamics of Language*. Elsevier, Oxford.

Eugene Charniak. 1996. *Statistical Language Learning*. MIT Press.

Stephen Clark and James Curran. 2007. Wide-coverage efficient statistical parsing with CCG and log-linear models. *Computational Linguistics*, 33(4):493–552.

Alexander Clark and Shalom Lappin. 2011. *Linguistic Nativism and the Poverty of the Stimulus*. Wiley-Blackwell.

Alexander Clark. 2011. A learnable representation for syntax using residuated lattices. In Philippe Groote, Markus Egg, and Laura Kallmeyer, editors, *Formal Grammar*, volume 5591 of *Lecture Notes in Computer Science*, pages 183–198. Springer Berlin Heidelberg.

Michael Collins and Brian Roark. 2004. Incremental parsing with the perceptron algorithm. In *Proceedings of the 42nd Meeting of the ACL*, pages 111–118, Barcelona.

Robin Cooper. 2005. Records and record types in semantic theory. *Journal of Logic and Computation*, 15(2):99–112.

Matthew Crocker, Martin Pickering, and Charles Clifton, editors. 2000. *Architectures and Mechanisms in Sentence Comprehension*. Cambridge University Press.

A.P. Dempster, N.M. Laird, and D. B. Rubin. 1977. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, 39(1):1–38.

Simon Dobnik, Robin Cooper, and Staffan Larsson. 2012. Modelling language, action, and perception in type theory with records. In *Proceedings of the 7th International Workshop on Constraint Solving and Language Processing (CSLP12)*, pages 51–63.

Arash Eshghi, Julian Hough, Matthew Purver, Ruth Kempson, and Eleni Gregoromichelaki. 2012. Conversational interactions: Capturing dialogue dynamics. In S. Larsson and L. Borin, editors, *From Quantification to Conversation: Festschrift for Robin Cooper on the occasion of his 65th birthday*, volume 19 of *Tributes*, pages 325–349. College Publications, London.

Arash Eshghi, Matthew Purver, and Julian Hough. 2013. Probabilistic induction for an incremental semantic grammar. In *Proceedings of the 10th International Conference on Computational Semantics (IWCS 2013) – Long Papers*, pages 107–118, Potsdam, Germany, March. Association for Computational Linguistics.

Raquel Fernández. 2006. *Non-Sentential Utterances in Dialogue: Classification, Resolution and Use*. Ph.D. thesis, King's College London, University of London.

Fernanda Ferreira and Benjamin Swets. 2002. How incremental is language production? evidence from the production of utterances requiring the computation of arithmetic sums. *Journal of Memory and Language*, 46:57–84.

Victor Ferreira. 1996. Is it better to give than to donate? Syntactic flexibility in language production. *Journal of Memory and Language*, 35:724–755.

Jonathan Ginzburg. 2012. *The Interactive Stance: Meaning for Conversation*. Oxford University Press.

E. Mark Gold. 1967. Language identification in the limit. *Information and Control*, 10(5):447–474.

John Hale. 2001. A probabilistic Earley parser as a psycholinguistic model. In *Proceedings of the 2nd Conference of the North American Chapter of the Association for Computational Linguistics*, Pittsburgh, PA.

Julian Hough and Matthew Purver. 2012. Processing self-repairs in an incremental type-theoretic dialogue system. In *Proceedings of the 16th SemDial Workshop on the Semantics and Pragmatics of Dialogue (SeineDial)*, pages 136–144, Paris, France, September.

Christine Howes, Matthew Purver, Rose McCabe, Patrick G. T. Healey, and Mary Lavelle. 2012. Predicting adherence to treatment for schizophrenia from dialogue transcripts. In *Proceedings of the 13th Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL 2012 Conference)*, pages 79–83, Seoul, South Korea, July. Association for Computational Linguistics.

Ruth Kempson, Wilfried Meyer-Viol, and Dov Gabbay. 2001. *Dynamic Syntax: The Flow of Language Understanding*. Blackwell.

Dan Klein and Christopher D. Manning. 2005. Natural language grammar induction with a generative constituent-context mode. *Pattern Recognition*, 38(9):1407–1419.

Tom Kwiatkowski, Luke Zettlemoyer, Sharon Goldwater, and Mark Steedman. 2010. Inducing probabilistic CCG grammars from logical form with higher-order unification. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 1223–1233, Cambridge, MA, October. Association for Computational Linguistics.

Tom Kwiatkowski, Sharon Goldwater, Luke Zettlemoyer, and Mark Steedman. 2012. A probabilistic model of syntactic and semantic acquisition from child-directed utterances and their meanings. In *Proceedings of the Conference of the European Chapter of the Association for Computational Linguistics (EACL)*.

Vincenzo Lombardo and Patrick Sturt. 1997. Incremental processing and infinite local ambiguity. In *Proceedings of the 1997 Cognitive Science Conference*.

Brian MacWhinney. 2000. *The CHILDES Project: Tools for Analyzing Talk*. Lawrence Erlbaum Associates, Mahwah, New Jersey, third edition.

Fernando Pereira and Yves Schabes. 1992. Inside-outside reestimation from partially bracketed corpora. In *Proceedings of the 30th Annual Meeting of the Association for Computational Linguistics*, pages 128–135, Newark, Delaware, USA, June. Association for Computational Linguistics.

Matthew Purver, Arash Eshghi, and Julian Hough. 2011. Incremental semantic construction in a dialogue system. In J. Bos and S. Pulman, editors, *Proceedings of the 9th International Conference on Computational Semantics*, pages 365–369, Oxford, UK, January.

Kenji Sagae, Brian MacWhinney, and Alon Lavie. 2004. Adding syntactic annotations to transcripts of parent-child dialogs. In *Proceedings of the 4th International Conference on Language Resources and Evaluation (LREC)*, pages 1815–1818, Lisbon.

Yo Sato. 2011. Local ambiguity, search strategies and parsing in Dynamic Syntax. In E. Gregoromichelaki, R. Kempson, and C. Howes, editors, *The Dynamics of Lexical Interfaces*. CSLI Publications.

Matthew Saxton. 1997. The contrast theory of negative input. *Journal of Child Language*, 24(1):139–161.

Mark Steedman. 2000. *The Syntactic Process*. MIT Press, Cambridge, MA.

Luke Zettlemoyer and Michael Collins. 2007. Online learning of relaxed CCG grammars for parsing to logical form. In *Proceedings of the Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*.