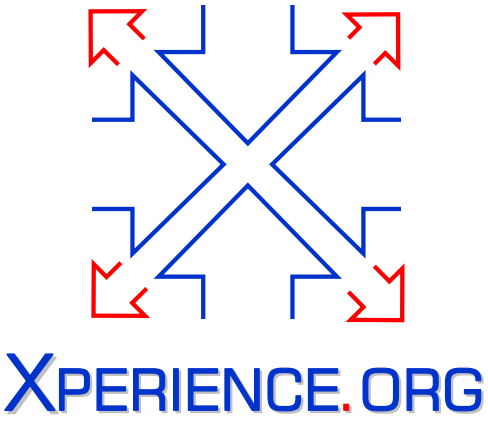
Project Acronym:	Xperience
Project Type:	IP
Project Title:	Robots Bootstrapped through Learning from Experience
Contract Number:	215821
Starting Date:	01-01-2011
Ending Date:	31-12-2015



Deliverable Number:	D4.2.2
Deliverable Title :	Structural BootStrapping for Syntax and Semantics
Type (Internal, Restricted, Public):	PU
Authors	M. Steedman
Contributing Partners	UEDIN

Contractual Date of Delivery to the EC:	31-01-2013
Actual Date of Delivery to the EC:	06-02-2013

Contents

Structurally Bootstrapping Grounded Language for Robots *

Mark Steedman
University of Edinburgh
steedman@inf.ed.ac.uk

February 11, 2013

ABSTRACT

This paper constitutes D.4.2.2 for the Xperience project, consisting of an executive summary of work over the period M13-M24, and of the attached publications.

1 INTRODUCTION

The supposed demise in the late 1980s of "Good Old-Fashioned Artificial Intelligence" (GOFAI), due to its admitted failure to come up with systems worthy of the name, deployable in the real-world for practical tasks, was widely agreed to have been due to hubristic overreliance on the notion of "representation", and insufficient attention to the problem of "grounding" representations in the world, or rather, the sensory-motor manifolds that transduce the world into the machine. Brooks (1991) argued that it was better to use the world in this sense "as its own model" when it came to tasks like acting in real time in real dynamic situations. Brooks pointed out that the time it took evolution on Earth to progress from bacteria to arthropod intelligence was on the order of 3Bn years, as compared to only around 0.5 Bn from there to primate intelligence, plausibly suggesting that the higher level problems would be relatively easy once the problem of grounded insect-level intelligence was solved, a problem he proposed to solve using a layered "subsumption architecture".

A number of successful reactive and adaptive robots at this level of sensory-motor control were built according to this philosophy, which played an important though not exclusive role in applications like the Mars rover and other autonomous vehicles. There has subsequently been rather less success at building on this foundation towards the level of higher intelligence, suggesting that the problem of insect level cognition has not entirely been solved. Nevertheless, a lesson has

*

been learned.

The Xperience project seeks to extend the capabilities of reactive planning by inducing event representations in the form of PDDL STRIPS-style rules from observed changes in a state representation transduced from sensory-motor level representations on the KIT ARMAR platform. PDDL rules constitute the top level of the OACS architecture for action represntations described by Krüger et al. (2011), while sensory motor representations for ARMAR form the lowest level.

One of the problems in achieving this goal is that the world is not in fact its own best representation for this purpose: ARMAR is not grounded in anything like the sense that a child or a kitten is grounded. Since we don't have 500M years and the resources of a planet available to allow such grounding to evolve, we have to construct a representation.

Our work on learning event representations under conditions of noise, error and partial observation arising under real world conditions is reported elsewhere under Xperience deliverables D3.2.1 and D3.2.2. However, we return to the details of the representation itself below

Natural language understanding might be seen as having undergone a rather similar progression to robotics over the same period. Along with a general shift away from concerns with expressive linguistic representation towards lower-level context-free or even finite-state representations in order to scale to systems of the size needed for real-world applications of the kind newly made possible by the effects of Moore's Law on computing power, there has been a similar shift towards machine-learning and optimization of statistical models.

Nowhere has this change been more obvious that in the subfield of natural language semantic interpretation and inference, where the emphasis has shifted from representations closely resembling linguistic forms, and entirely parallel to logicist AI knowledge representations as far as inference goes, to distributed representations based on collocation statistics or paraphrase relations, induced by machine-learning methods such as clustering over large volumes of text. Again, there has been considerable improvement over earlier methods on certain low-level tasks NLP tasks such as information retrieval and extraction of the kind exploited by Google. However, there has been the same disappointingly limited progress in lifting these low-level successes to the higher level tasks such as open-domain question-answering.

Alongside these two parallel developments, there has been a lot of interest in the idea that the access of new-generation robots to grounded representations could be leveraged to induce a similarly grounded semantics for natural language (Steels 1998). Such grounded language semantics might be expected to support more efficient commonsense reasoning and inference. Discovering a semantics of this kind and using it to induce "semantic parsers" for multiple human languages is one of the goals of Xperience under the present workpackage that we report on here.

Since we have noted that our robots are not grounded in any relevant sense, and that a state representation from which we can induce grounded event representations has to be constructed by hand, It might appear that we are back in the

trap of old-fashioned AI, dooomed to construct knowledge representations that will be special case and non-generalizable to other robot tasks and domains.

However, we claim that the task of making the state and event representations support higher level processes including both deliberative planning and a specifically natural language semantics that can support human-like language learning and grammar induction for any language of the world constrains the state space in ways that make it widely applicable and general-purpose, and allow us to escape the logicist trap. The rationale behind the claim is that nobody knows any way in which children could learn the languages of the world except by having access to a conceptual representation that is closely related to the non-linguistic conceptual representation with which our closest animal relatives act purposefully in the world, and to which the languages of the world, despite their diversity, are transparent enough for the child to see the relation. If so, then we should be able to work out from observation of the ways languages structure their semantics what the hidden common conceptual representation must be. In this connection, it is important to realize that we know of some strong commonalities across the linguistic categories that represent events and their relations, namely tense, mood, aspect, and adverbial modification.

This work requires achieving two subgoals. The first is the development of a general purpose mechanism for inducing "semantic parsers"—that is, grammars and parsing models that simultaneously parse and build interpretations—from exposure to paired sentences and meanings (Thompson and Mooney 2003; Zettlemoyer and Collins 2005). This process is often referred to in the context of child language acquisition as "Semantic Bootstrapping" of grammatical knowledge (Pinker 1979, 1984). In the later stages of child language acquisition, Gleitman (1990) and Papafragou, Cassidy and Gleitman (2007) have described a related process of "Syntactic Bootstrapping", according to which earlier semantic bootstrapping draws the childas attention via learned grammatical rules to new lexical entries in a process of one-trial learning.

The sentences in such a procedure may be in any language and the meaning representations my be in an arbitrary knowledge representation (so long as the latter is compositional). First results were reported last year and published as Kwiatkowski et al. (2010). This phase of the work is largely complete, pursued as anticipated in the Xperience proposal in conjunction with Dr. Luke Zettlemoyer, now at University of Washington, Seattle. Noteworthy results of this work discussed in section 2 include: state-of-the-art semantic parser induction algorithms; the first psychologically realistic and computationally effective model of child language acquisition by semantic bootstrapping; and a demonstration that syntactic bootstrapping effects are predicted by the same Bayesian model that mediates semantic bootstrapping, rather than stemming from a separate process.

The second is the application of such a learner to data pairing sentences with a meaning representation for natural language temporal expressions including tense, mood, aspect and adverbial modification, following Reichenbach (1947), Vendler (1967), and Davidson (1967), as outlined in Steedman (1997, 2012a,b). The meaning representation is grounded in a STRIPS-like event representation

that constitutes the top level of the Object-Action Complex architecture (OACs, Krüger et al. 2011), developed for the robot domain under WP3, as reported in D3.2.2. This phase of the work is ongoing. Noteworthy results of this work discussed in section 3 include a specification of the first fully formal semantics of tense, mood, and aspect in natural language integrating a truly grounded action representation into linguistic semantics, using the linear dynamic event calculus (LDEC) logical formalism first outlined in Steedman 2002.

The next two sections of this report summarize progress on and notable results from these two problems, referring to the published papers in the appendix.

REFERENCES

Brooks, Rodney. 1991. "Intelligence without Representation." *Artificial Intelligence*, 47, 139–159.

Davidson, Donald. 1967. "The Logical Form of Action Sentences." In Nicholas Rescher, ed., *The Logic of Decision and Action*, 81–95. Pittsburgh, PA: University of Pittsburgh Press.

Gleitman, Lila. 1990. "The Structural Sources of Verb Meanings." *Language Acquisition*, 1, 1–55.

Krüger, Norbert, Christopher Geib, Justus Piater, Ronald Petrick, Mark Steedman, Florentin Wörgötter, Aleš Ude, Tamim Asfour, Dirk Kraft, Damir Omrčen, Alejandro Agostini, and Rüdiger Dillmann. 2011. "Object-Action Complexes: Grounded Abstractions of Sensorimotor Processes." *Robotics and Autonomous Systems*, 59, 740–757.

Kwiatkowski, Tom, Luke Zettlemoyer, Sharon Goldwater, and Mark Steedman. 2010. "Inducing Probabilistic CCG Grammars from Logical Form with Higher-Order Unification." In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, 1223–1233. Cambridge, MA: ACL.

Papafragou, Anna, Kimberly Cassidy, and Lila Gleitman. 2007. "When We Think about Thinking: The Acquisition of Belief Verbs." *Cognition*, 105, 125–165.

Pinker, Steven. 1979. "Formal Models of Language Learning." *Cognition*, 7, 217–283.

Pinker, Steven. 1984. *Language Learnability and Language Development*. Cambridge, MA: Harvard University Press.

Reichenbach, Hans. 1947. *Elements of Symbolic Logic*. Berkeley: University of California Press.

Steedman, Mark. 1997. "Temporality." In Johan van Benthem and Alice ter Meulen, eds., *Handbook of Logic and Language*, 895–938. Amsterdam: North Holland/Elsevier.

Steedman, Mark. 2002. "Plans, Affordances, and Combinatory Grammar." *Linguistics and Philosophy*, 25, 723–753.

Steedman, Mark. 2012a. "Computational Linguistics." In Robert Binnick, ed., *Handbook of Tense and Aspect*, 102–120. Oxford: Oxford University Press.

Steedman, Mark. 2012b. *Taking Scope: The Natural Semantics of Quantifiers*. Cambridge, MA: MIT Press.

Steels, Luc. 1998. "The Origins of Syntax in Visually Grounded Robotic Agents." *Artificial Intelligence*, 103, 133–156.

Thompson, Cynthia, and Raymond Mooney. 2003. "Acquiring Word-Meaning Mappings for Natural Language Interfaces." *Journal of Artificial Intelligence Research*, 18, 1–44.

Vendler, Zeno. 1967. *Linguistics in Philosophy*. Ithaca, NY: Cornell University Press.

Zettlemoyer, Luke, and Michael Collins. 2005. "Learning to Map Sentences to Logical Form: Structured Classification with Probabilistic Categorial Grammars." In *Proceedings of the 21st Conference on Uncertainty in AI (UAI)*, 658–666. Edinburgh: AAAI.

2 SEMANTIC PARSERS

The task of semantic parser induction is to automatically learn a grammar and a parsing model for a parser that builds semantically interpretable structures from data pairing sentences in natural language with their meanings expressed in some meaning representation language. The sentences can be in any human language (English, Japanese, Navajo, etc.), and the meaning representations can be in any form for which a compositional semantics, pairing meaning-building operations with syntactic operations and lexical items, can be devised. Examples of the latter are: database query languages such as SQL or the SABRE air travel booking system; tactical plays in RoboCup soccer games or Dungeons and Dragons-style video games; LISP S-expressions; terms in the λ-calculus; full Montague semantics for natural languages.

Semantic parsers are a relevant technology for systems involving large grammars and/or "productive" grammars generating unboundedly large stringsets. Embedding constructions like relative clauses and complement clauses are productive in this sense. An example of the latter construction arises from verbs like "help" in English, which gives rise to an unboundedly large family of sentences like the following:

(1) a. Help me to wipe the table clean.
    b. I persuaded Frank to help me to wipe the table clean.
    c. I told Betty to help Frank to help me wipe the table clean.
    d. etc.

The GeoQueries database (Thompson and Mooney, 2003) pairs a productive set of English queries with logical forms denoting queries concerning geographical information, such as the following:

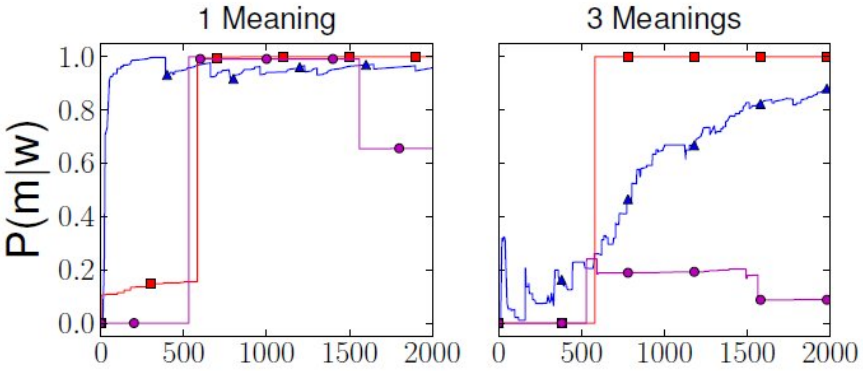
(2) a. Which states border Texas?
    b. Which states border states which border Texas?
    c. etc.

Zettlemoyer and Collins (2005) showed that a small, relatively unambiguous, but unboundedly productive parser for a Combinatory Categorial Grammar (CCG, Steedman, 2000) can be induced from such data. (CCG is particularly well suited to this task, because it is fully lexicalized, with all language-specific information being specified in the lexicon, and because the syntactic operations that project lexical items onto the sentences of the language are monotononic and fully transparent to a compositional semantics.

Zettlemoyer and Collins' method, like that of Thompson and Mooney, is to some extent English-specific, and does not include a fully general parsing model, making it difficult to generalize and scale to other languages and larger grammars. However, Kwiatkowski et al. 2010 showed that a fully general method amounting to learning a parsing model for the entire space of grammars permitted by CCG could be applied to the GeoQueries data to induce semantic parsers not only for English. but also for corresponding fragments of Italian, Japanese, and Turkish, with performance at a new state of the art.

Kwiatkowski et al. (2011) (see appendix) show that this method scales to the much larger and more complex ATIS dataset of English sentences and database queries and answers for a travel bookings domain, with state-of-the-art performance.

Kwiatkowski et al. (2012) (see appendix), using a related algorithm using a generative Bayesian model in application to a semantically annotated portion of the CHILDES database of real child-directed adult utterance, show that semantic parser induction provides a psychologically realistic model of child language acquisition exhibiting the structural bootstrapping effects that are characteristic of child language acquisition (Gleitman, 1999). For example, the following pair of graphs shows the strength as learning proceeds of the hypothesis that the category for the highly frequent determiner *a*, the less frequent determiner *another*, and the very infrequent determiner *any* is to the left of nouns like *biscuit* in English. (The first graph shows learning from the correct logical form alone for each utterance, and the second shows learning for the correct form plus two plausible but irrelevant logical forms.) The frequent determiner *a* is learned incrementally and continuously, as one would expect. (Of course, learning is slower in the second graph with the distractors.) The infrequent determiners are learned later, again as one would expect. However, the fact that they have the same semantic type as the frequent determiner means that by the time they are encountered at all, the prior probability on the correct hypothesis has increased by learning the frequent one. Learning is therefore step-like and all-or-none, rather than incremental. Thus, the graphs demonstrate a clear case of structural bootstrapping, of the kind targeted in Xperience.



More generally, we claim that syntactic bootstrapping in language acquisition is an emergent late effect of shifting prior probabilities over the entire space of possibilities allowed by CCG in a uniform Bayesian model of semantic structural bootstrapping. A journal paper presenting this result for psycholinguistic and cognitive science audiences is in preparation.

Further work on this problem includes extending it to grounded action representations for the ARMAR robot platform, as expressed in the temporal semantic language developed in the next section, and further broadening the semantics to

include dialog planning, as in the original proposal, Annex 1.

REFERENCES

Kwiatkowski, Tom, Sharon Goldwater, Luke Zettlemoyer, and Mark Steedman. 2012. "A Probabilistic Model of Language Acquisition from Utterances and Meanings." In *Proceedings of the 13th Conference of the European Chapter of the ACL (EACL 2012)*, 234–244. Avignon: ACL.

Kwiatkowski, Tom, Luke Zettlemoyer, Sharon Goldwater, and Mark Steedman. 2011. "Lexical Generalization in CCG Grammar Induction for Semantic Parsing." In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, 1512–1523. Edinburgh: ACL.

3 GROUNDED SEMANTICS OF TEMPORALITY

3.1 The Task

The task is to take the PDDL/STRIPS action representations (the topmost level in the OAC architecture of Krüger et al. 2011), acquired by observing change in the dynamic sensory motor state representation, and to build plans using the PKS planner developed under WP3 for compound actions such as *wiping the table clean* that can be executed by the robot, and monitored by plan monitoring during execution. A semantic parser of the kind developed in the previous section can then be induced from such representations to obey commands such as "wipe the table clean", "Help me to wipe the table clean" and "Help Tamim to help me to wipe the table clean", and to deliver fault reports arising from errors observed in plan monitoring, as in the following dialog:

(3)USER : What happened?
ARMAR : I was wiping the table clean when the cloth disappeared.
USER : Is the table clean?
ARMAR : No.

The following is a preliminary sketch of a knowledge representation supporting STRIPS-style planning and a comositional semantics for natural language, for discussion in the Xperience project in ongoing work under WP3.1 in developing a version grounded in the ARMAR sensory-motor representation at the lower end of the OACs architecture. (Certain liberties are taken with the type-system in the interests of conveying the intuition—*caveat lector*.)

3.2 The Representation

We seek a knowledge representation that will support both STRIPS-style planning and a compositonal semantics for tense mood and aspect in natural language. We use the LDEC notation described by Steedman 2002, in which the implication sign $\multimap$ denotes linear implication, according to which antecedents on the left are consumed or deleted. Thus, rules of the following form mean that if P and Q hold, then if you take action α followed by actiion β, then Q no longer holds, R does hold, and whether P holds or not is undefined:

(4) $\{P\} \wedge Q \multimap [\alpha;\beta]R$

This notation can be regarded as a (sugared, readable) version of the PDDL notation used by the planner (McDermott et al 1998), defined in terms of state fluents grounded in robot sensory motorics.

3.2.1 Knowledge of wiping

In order to preempt well-known "ramification problems" arising from interval-based temporal knowledge representation systems, we represent durative events by an instantaneous event initiating the process in question and introducing a fact that the process in question is *ongoing*, followed by an event concluding the pro-

cess and consuming or deleting that fact:

(5) a. $\neg ongoing(p) \multimap [commence(p)]\ ongoing(p)$
b. $ongoing(p) \multimap [conclude(p)]\neg ongoing(p)$

The distinction of Vendler 1967 between atelic events of type "activity" like *running* (which can be modified by "for ten minutes" but not by "in ten minutes") and telic events of type "accomplishment" (for which the reverse holds) is expressed in LDEC as the distinction between an atomic event of running (which may be iterated) and a Piagetian "circular reaction" or conditionally iterating "TOTE unit" or operant (Piaget, 1936; Miller, Galanter, and Pribram 1960), of the form $(\neg g?;p)^+$, meaning "p until g". The general form of such atelic and telic units is as follows:[1]

(6) a. $\{afford(p^+)\} \multimap [commence(p^+(E));p^+;conclude(p^+(E))]$
b. $\{afford(p^+)\} \wedge cause(p^+,g) \wedge \neg g$
$\multimap [commence(p^+(g,E));(\neg g?;p)^+;conclude(p^+(g,E))]g$

Events are defined in terms both of the class of situations that afford their occurrence, and in terms of a linear rule describing their effects:

(7) a. $on(X,cloth) \wedge grasp(me,cloth) \Rightarrow afford(wipe(me,X)^+))$
b. $\{afford(wipe(me,X)^+))\}$
$\multimap [commence(wipe(me,X)^+(E));wipe(me,X)^+;conclude(wipe(me,X)^+(E))]$
c. $\{afford(wipe(me,X)^+))\} \wedge \neg clean(table)$
$\multimap [commence(wipe(me,table)^+(clean(table)),E));(\neg clean(table)?;wipe(me,table))^+;$
$conclude(wipe(me,table)^+(clean(table)),E))]clean(table)$

3.2.2 The goal

The goal is to find a plan α that gets from the present situation to one in which the table is clean:

(8) $afford(\alpha) \wedge [\alpha]clean(table)$

3.2.3 The Plan

The plan is the accomplishment of *wiping the table clean*, by iterating *wiping* until *clean*:

(9) $commence(wipe(me,table)^+(clean(table)),E));(\neg clean(table)?;wipe(me,table))^+;$
$conclude(wipe(me,table)^+(clean(table)),E))$

3.2.4 The Model

The following is a trace of the state defined as a set of ground fluents: (1) when the accomplishment of *wiping the table clean* is successfully ongoing; (2) when plan monir-toring reveals that the cloth has mysteriously vanished, due to sensor error; (3) after the plan as a consequence fails.

[1] If a situation affords p^+ (as opposed to merely affording p) p must be an elementary action whose consequent state still affords p.

(10) 1. $on(table,cloth) \wedge grasp(me,cloth) \wedge \neg clean(table) \wedge ongoing([(\neg clean(table)?;wipe(x,y))^{+}]) \wedge goal(clean(table))$

2. $\neg clean(table) \wedge ongoing([(\neg clean(table)?;wipe(x,y))^{+}]) \wedge goal(clean(table))$

3. $\neg clean(table) \wedge goal(clean(table))$

3.2.5 The Appropriate Utterance

(11) I was wiping the table clean when the cloth disappeared.

3.2.6 Semantics of Tense, Mood, and Aspect

The bare infinitival denotes the bare activity of wiping or the bare accomplishment of wiping until clean:

(12) a. wipe := $(S\backslash NP)/NP : \lambda x\lambda y.[wipe(x,y)^{+}]$
b. wipe := $((S\backslash NP)/NP)/AP : \lambda w\lambda x\lambda y.[(\neg w(x)?;wipe(me,x))^{+}]$

The gerund represents the activity or accomplishment as ongoing:

(13) a. wiping := $(Sin\backslash NP)/NP : \lambda x\lambda y.ongoing([wipe(x,y)])$
b. wiping := $((S\backslash NP)/NP)/AP : \lambda w\lambda x\lambda y.ongoing([(\neg w(x)?;wipe(x,y))^{+}])$

Tense defines the relation of Reichenbachian reference time R to the commencement and conclusion of activity or accomplishment E.

(14) a. wipes := $(S\backslash NP)/NP : \lambda x\lambda y\lambda E\lambda R.[commence(wipe(x,y)(E);wipe(x,y);$
$conclude(wipe(x,y)(E))] \wedge E = R$
b. wipes := $((S\backslash NP)/NP)/AP : \lambda w\lambda x\lambda y\lambda E\lambda R.[commence(wipe(y,y)(w(x),E));$
$(\neg w(x)?;wipe(me,x))^{+};conclude(wipe(y,y)(w(x),E))]$
$\wedge E = R$

A preliminary paper surveying current approaches to temporal semantics for natural language based on Reichenbach, Vendler, and Davidson appears as Steedman (2012a) (see appendix).

This work constitutes a component of a wider project to deliver an ambitious computationally manageable full semantics for unrestricted natural language text, which is under development in the form of a modification of existing wide-coverage CCG parsers for natural language. The first component of this semantics, concerning quantified NPs and negation, has appeared this year as a book with MIT Press (Steedman 2012b). The implementation of this theory, together with an extension representing the meaning of content words as clusters based on paraphrase detection in large amounts of unlabeled text, forms the basis of Mike Lewis' ongoing Phd under Xperience funding, and of a paper in submission to a journal (Lewis and Steedman 2012)

Work in progress reported under D3.1.2 includes the grounding of the event semantics in ARMAR sensory-motor level representations under the multi-level OAC architecture (Krüger et al., 2011), and its incorporation in the latter wide coverage parser.

References

Lewis, Michael, and Mark Steedman. 2012. "Combining Distributional and Logical Semantics." *Submitted*.

Steedman, Mark. 2012a. "Computational Linguistics." In Robert Binnick, ed., *Handbook of Tense and Aspect*, 102–120. Oxford: Oxford University Press.

Steedman, Mark. 2012b. *Taking Scope: The Natural Semantics of Quantifiers*. Cambridge, MA: MIT Press.

APPENDIX

A Probabilistic Model of Syntactic and Semantic Acquisition from Child-Directed Utterances and their Meanings

Tom Kwiatkowski[*,†] tomk@cs.washington.edu
Sharon Goldwater[*] sgwater@inf.ed.ac.uk
Luke Zettlemoyer[†] lsz@cs.washington.edu
Mark Steedman[*] steedman@inf.ed.ac.uk

[*] ILCC, School of Informatics
University of Edinburgh
Edinburgh, EH8 9AB, UK

[†]Computer Science & Engineering
University of Washington
Seattle, WA, 98195, USA

Abstract

This paper presents an incremental probabilistic learner that models the acquisition of syntax and semantics from a corpus of child-directed utterances paired with possible representations of their meanings. These meaning representations approximate the contextual input available to the child; they do not specify the meanings of individual words or syntactic derivations. The learner then has to infer the meanings and syntactic properties of the words in the input along with a parsing model. We use the CCG grammatical framework and train a non-parametric Bayesian model of parse structure with online variational Bayesian expectation maximization. When tested on utterances from the CHILDES corpus, our learner outperforms a state-of-the-art semantic parser. In addition, it models such aspects of child acquisition as "fast mapping," while also countering previous criticisms of statistical syntactic learners.

1 Introduction

Children learn language by mapping the utterances they hear onto what they believe those utterances mean. The precise nature of the child's prelinguistic representation of meaning is not known. We assume for present purposes that it can be approximated by compositional logical representations such as (1), where the meaning is a logical expression that describes a relationship $have$ between the person you refers to and the object $another(x, cookie(x))$:

$$\begin{aligned}&\text{Utterance}: \text{you have another cookie} \quad (1)\\ &\text{Meaning}: have(you, another(x, cookie(x)))\end{aligned}$$

Most situations will support a number of plausible meanings, so the child has to learn in the face of *propositional uncertainty*[1], from a set of contextually afforded meaning candidates, as here:

$$\begin{aligned}&\text{Utterance}: \text{you have another cookie}\\ &\text{Candidate Meanings}: \begin{cases} have(you, another(x, cookie(x))) \\ eat(you, your(x, cake(x))) \\ want(i, another(x, cookie(x))) \end{cases}\end{aligned}$$

The task is then to learn, from a sequence of such (utterance, meaning-candidates) pairs, the correct lexicon and parsing model. Here we present a probabilistic account of this task with an emphasis on cognitive plausibility.

Our criteria for plausibility are that the learner must not require any language-specific information prior to learning and that the learning algorithm must be strictly *incremental*: it sees each training instance sequentially and exactly once. We define a Bayesian model of parse structure with Dirichlet process priors and train this on a set of (utterance, meaning-candidates) pairs derived from the CHILDES corpus (MacWhinney, 2000) using online variational Bayesian EM.

We evaluate the learnt grammar in three ways. First, we test the accuracy of the trained model in parsing unseen utterances onto gold standard annotations of their meaning. We show that it outperforms a state-of-the-art semantic parser (Kwiatkowski et al., 2010) when run with similar training conditions (i.e., neither system is given the corpus based initialization originally used by Kwiatkowski et al.). We then examine the learning curves of some individual words, showing that the model can learn word meanings on the basis of a single exposure, similar to the *fast mapping* phenomenon observed in children (Carey and Bartlett, 1978). Finally, we show that our

[1]Similar to *referential uncertainty* but relating to propositions rather than referents.

Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics, pages 234–244,
Avignon, France, April 23 - 27 2012.

learner captures the step-like learning curves for word order regularities that Thornton and Tesan (2007) claim children show. This result counters Thornton and Tesan's criticism of statistical grammar learners—that they tend to exhibit gradual learning curves rather than the abrupt changes in linguistic competence observed in children.

1.1 Related Work

Models of syntactic acquisition, whether they have addressed the task of learning both syntax and semantics (Siskind, 1992; Villavicencio, 2002; Buttery, 2006) or syntax alone (Gibson and Wexler, 1994; Sakas and Fodor, 2001; Yang, 2002) have aimed to learn a single, correct, deterministic grammar. With the exception of Buttery (2006) they also adopt the Principles and Parameters grammatical framework, which assumes detailed knowledge of linguistic regularities[2]. Our approach contrasts with all previous models in assuming a very general kind of linguistic knowledge and a probabilistic grammar. Specifically, we use the probabilistic Combinatory Categorial Grammar (CCG) framework, and assume only that the learner has access to a small set of general combinatory schemata and a functional mapping from semantic type to syntactic category. Furthermore, this paper is the first to evaluate a model of child syntactic-semantic acquisition by parsing unseen data.

Models of child word learning have focused on semantics only, learning word meanings from utterances paired with either sets of concept symbols (Yu and Ballard, 2007; Frank et al., 2008; Fazly et al., 2010) or a compositional meaning representation of the type used here (Siskind, 1996). The models of Alishahi and Stevenson (2008) and Maurits et al. (2009) learn, as well as word-meanings, orderings for verb-argument structures but not the full parsing model that we learn here.

Semantic parser induction as addressed by Zettlemoyer and Collins (2005, 2007, 2009), Kate and Mooney (2007), Wong and Mooney (2006, 2007), Lu et al. (2008), Chen et al. (2010), Kwiatkowski et al. (2010, 2011) and Börschinger et al. (2011) has the same task definition as the one addressed by this paper. However, the learning approaches presented in those previous papers are not designed to be cognitively plausible, using batch training algorithms, multiple passes over the data, and language specific initialisations (lists of noun phrases and additional corpus statistics), all of which we dispense with here. In particular, our approach is closely related that of Kwiatkowski et al. (2010) but, whereas that work required careful initialisation and multiple passes over the training data to learn a discriminative parsing model, here we learn a generative parsing model without either.

[2]This linguistic use of the term "parameter" is distinct from the statistical use found elsewhere in this paper.

1.2 Overview of the approach

Our approach takes, as input, a corpus of (utterance, meaning-candidates) pairs $\{(s_i, \{m\}_i) : i = 1, \ldots, N\}$, and learns a CCG lexicon Λ and the probability of each *production* $a \rightarrow b$ that could be used in a parse. Together, these define a probabilistic parser that can be used to find the most probable meaning for any new sentence.

We learn both the lexicon and production probabilities from allowable parses of the training pairs. The set of allowable parses $\{t\}$ for a single (utterance, meaning-candidates) pair consists of those parses that map the utterance onto one of the meanings. This set is generated with the functional mapping $\mathcal{T}$:

$$\{t\} = \mathcal{T}(s, m), \tag{2}$$

which is defined, following Kwiatkowski et al. (2010), using only the CCG combinators and a mapping from semantic type to syntactic category (presented in in Section 4).

The CCG lexicon Λ is learnt by reading off the lexical items used in all parses of all training pairs. Production probabilities are learnt in conjunction with Λ through the use of an incremental parameter estimation algorithm, online Variational Bayesian EM, as described in Section 5.

Before presenting the probabilistic model, the mapping $\mathcal{T}$, and the parameter training algorithm, we first provide some background on the meaning representations we use and on CCG.

2 Background

2.1 Meaning Representations

We represent the meanings of utterances in first-order predicate logic using the lambda-calculus. An example logical expression (henceforth also referred to as a lambda expression) is:

$$like(eve, mummy) \tag{3}$$

which expresses a logical relationship $like$ between the entity eve and the entity $mummy$. In Section 6.1 we will see how logical expressions like this are created for a set of child-directed utterances (to use in training our model).

The lambda-calculus uses λ operators to define functions. These may be used to represent functional meanings of utterances but they may also be used as a 'glue language', to compose elements of first order logical expressions. For example, the function $\lambda x \lambda y.like(y, x)$ can be combined with the object $mummy$ to give the phrasal meaning $\lambda y.like(y, mummy)$ through the lambda-calculus operation of *function application.*

2.2 CCG

Combinatory Categorial Grammar (CCG; Steedman 2000) is a strongly lexicalised linguistic formalism that tightly couples syntax and semantics. Each CCG lexical item in the lexicon Λ is a triple, written as $\text{word} \vdash \mathsf{syntactic\ category} : \mathit{logical\ expression}$. Examples are:

$$\begin{aligned} \text{You} &\vdash \mathsf{NP} : you \\ \text{read} &\vdash \mathsf{S\backslash NP/NP} : \lambda x \lambda y.read(y, x) \\ \text{the} &\vdash \mathsf{NP/N} : \lambda f.the(x, f(x)) \\ \text{book} &\vdash \mathsf{N} : \lambda x.book(x) \end{aligned}$$

A full CCG category $\mathsf{X} : h$ has syntactic category X and logical expression h. Syntactic categories may be atomic (e.g., S or NP) or complex (e.g., $(\mathsf{S\backslash NP})/\mathsf{NP}$). Slash operators in complex categories define functions from the range on the right of the slash to the result on the left in much the same way as lambda operators do in the lambda-calculus. The direction of the slash defines the linear order of function and argument.

CCG uses a small set of *combinatory rules* to concurrently build syntactic parses and semantic representations. Two example combinatory rules are forward ($>$) and backward ($<$) *application*:

$$\begin{aligned} X/Y : f \quad Y : g \quad &\Rightarrow \quad X : f(g) \qquad (>) \\ Y : g \quad X\backslash Y : f \quad &\Rightarrow \quad X : f(g) \qquad (<) \end{aligned}$$

Given the lexicon above, the phrase "You read the book" can be parsed using these rules, as illustrated in Figure 1 (with additional notation discussed in the following section)..

CCG also includes combinatory rules of forward ($> \mathbf{B}$) and backward ($< \mathbf{B}$) *composition*:

$$\begin{aligned} X/Y : f \quad Y/Z : g \Rightarrow X/Z : \lambda x.f(g(x)) \quad (> \mathbf{B}) \\ Y\backslash Z : g \quad X\backslash Y : f \Rightarrow X\backslash Z : \lambda x.f(g(x)) \quad (< \mathbf{B}) \end{aligned}$$

3 Modelling Derivations

The objective of our learning algorithm is to learn the correct parameterisation of a probabilistic model $P(s, m, t)$ over (utterance, meaning, derivation) triples. This model assigns a probability to each of the *grammar productions* $a \rightarrow b$ used to build the derivation tree t. The probability of any given CCG derivation t with sentence s and semantics m is calculated as the product of all of its production probabilities.

$$P(s, m, t) = \prod_{a \rightarrow b \in t} P(b|a) \tag{4}$$

For example, the derivation in Figure 1 contains 13 productions, and its probability is the product of the 13 production probabilities. Grammar productions may be either *syntactic*—used to build a syntactic derivation tree, or *lexical*—used to generate logical expressions and words at the leaves of this tree.

A syntactic production $\mathsf{C_h} \rightarrow \mathcal{R}$ expands a head node $\mathsf{C_h}$ into a result $\mathcal{R}$ that is either an ordered pair of syntactic parse nodes $\langle \mathsf{C_l}, \mathsf{C_r} \rangle$ (for a binary production) or a single parse node (for a unary production). Only two unary syntactic productions are allowed in the grammar: $\mathsf{START} \rightarrow \mathsf{A}$ to generate A as the top syntactic node of a parse tree and $\mathsf{A} \rightarrow [\mathsf{A}]_{\mathsf{lex}}$ to indicate that A is a leaf node in the syntactic derivation and should be used to generate a logical expression and word. Syntactic derivations are built by recursively applying syntactic productions to non-leaf nodes in the derivation tree. Each syntactic production $\mathsf{C_h} \rightarrow \mathcal{R}$ has conditional probability $P(\mathcal{R}|\mathsf{C_h})$. There are 3 binary and 5 unary syntactic productions in Figure 1.

Lexical productions have two forms. *Logical expressions* are produced from leaf nodes in the syntactic derivation tree $\mathsf{A_{lex}} \rightarrow m$ with conditional probability $P(m|\mathsf{A_{lex}})$. *Words* are then produced from these logical expressions with conditional probability $P(w|m)$. An example logical production from Figure 1 is $[\mathsf{NP}]_{\mathsf{lex}} \rightarrow you$. An example word production is $you \rightarrow \text{You}$.

Every production $a \rightarrow b$ used in a parse tree t is chosen from the set of productions that could be used to expand a head node a. If there are a finite K productions that could expand a then a K-dimensional *Multinomial distribution* parameterised by θ_a can be used to model the categorical

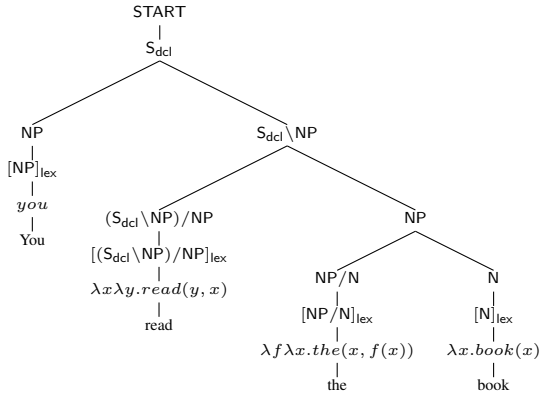


Figure 1: Derivation of sentence `You read the book` with meaning $read(you, the(x, book(x)))$.

choice of production:

$$b \sim \text{Multinomial}(\theta_a) \quad (5)$$

However, before training a model of language acquisition the dimensionality and contents of both the syntactic grammar and lexicon are unknown. In order to maintain a probability model with cover over the countably infinite number of possible productions, we define a Dirichlet Process (DP) prior for each possible production head a. For the production head a, $DP(\alpha_a, H_a)$ assigns some probability mass to all possible production targets $\{b\}$ covered by the base distribution H_a.

It is possible to use the DP as an infinite prior from which the parameter set of a finite dimensional Multinomial may be drawn provided that we can choose a suitable partition of $\{b\}$. When calculating the probability of an (s, m, t) triple, the choice of this partition is easy. For any given production head a there is a finite set of usable production targets $\{b_1, \ldots, b_{k-1}\}$ in t. We create a partition that includes one entry for each of these along with a final entry $\{b_k, \ldots\}$ that includes all other ways in which a could be expanded in different contexts. Then, by applying the distribution G_a drawn from the DP to this partition, we get a parameter vector θ_a that is equivalent to a draw from a k dimensional Dirichlet distribution:

$$G_a \sim DP(\alpha_a, H_a) \quad (6)$$

$$\begin{aligned}\theta_a &= (G_a(b_1), \ldots, G_a(b_{k-1}), G_a(\{b_k, \ldots\}) \\ &\sim \text{Dir}(\alpha_a H(b_1), \ldots, \alpha_a H_a(b_{k-1}), \\ &\qquad \alpha_a H_a(\{b_k, \ldots\}))\end{aligned} \quad (7)$$

Together, Equations 4-7 describe the joint distribution $P(\mathbf{X}, \mathbf{S}, \theta)$ over the observed training data $\mathbf{X} = \{(s_i, \{m\}_i) : i = 1, \ldots, N\}$, the latent variables $\mathbf{S}$ (containing the productions used in each parse t) and the parsing parameters θ.

4 Generating Parses

The previous section defined a parameterisation over parses assuming that the CCG lexicon Λ was known. In practice Λ is empty prior to training and must be populated with the lexical items from parses t consistent with training pairs $(s, \{m\})$.

The set of allowed parses $\{t\}$ is defined by the function $\mathcal{T}$ from Equation 2. Here we review the *splitting procedure* of Kwiatkowski et al. (2010) that is used to generate CCG lexical items and describe how it is used by $\mathcal{T}$ to create a packed chart representation of all parses $\{t\}$ that are consistent with s and at least one of the meaning representations in $\{m\}$. In this section we assume that s is paired at each point with only a single meaning m. Later we will show how $\mathcal{T}$ is used multiple times to create the set of parses consistent with s and a set of candidate meanings $\{m\}$.

The splitting procedure takes as input a CCG category $\mathsf{X}:h$, such as $\mathsf{NP}:a(x, cookie(x))$, and returns a set of *category splits*. Each category split is a pair of CCG categories $(\mathsf{C_l}:m_l, \mathsf{C_r}:m_r)$ that can be recombined to give $\mathsf{X}:h$ using one of the CCG combinators in Section 2.2. The CCG category splitting procedure has two parts: *logical splitting* of the category semantics h; and *syntactic splitting* of the syntactic category X. Each logical split of h is a pair of lambda expressions (f, g) in the following set:

$$\{(f, g) \mid h = f(g) \ \vee \ h = \lambda x.f(g(x))\}, \quad (8)$$

which means that f and g can be recombined using either *function application* or *function composition* to give the original lambda expression h. An example split of the lambda expression $h = a(x, cookie(x))$ is the pair

$$(\lambda y.a(x, y(x)), \lambda x.cookie(x)), \quad (9)$$

where $\lambda y.a(x, y(x))$ applied to $\lambda x.cookie(x)$ returns the original expression $a(x, cookie(x))$.

Syntactic splitting assigns linear order and syntactic categories to the two lambda expressions f and g. The initial syntactic category X is split by a reversal of the CCG application combinators in Section 2.2 if f and g can be recombined to give

Syntactic Category	Semantic Type	Example Phrase
$\mathsf{S_{dcl}}$	$\langle ev, t\rangle$	I took it $\vdash \mathsf{S_{dcl}}{:}\lambda e.took(i, it, e)$
$\mathsf{S_t}$	t	I'm angry $\vdash \mathsf{S_t}{:}angry(i)$
$\mathsf{S_{wh}}$	$\langle e, \langle ev, t\rangle\rangle$	Who took it? $\vdash \mathsf{S_{wh}}{:}\lambda x\lambda e.took(x, it, e)$
$\mathsf{S_q}$	$\langle ev, t\rangle$	Did you take it? $\vdash \mathsf{S_q}{:}\lambda e.Q(take(you, it, e))$
N	$\langle e, t\rangle$	cookie $\vdash \mathsf{N}{:}\lambda x.cookie(x)$
NP	e	John $\vdash \mathsf{NP}{:}john$
PP	$\langle ev, t\rangle$	on John $\vdash \mathsf{PP}{:}\lambda e.on(john, e)$

Figure 2: Atomic Syntactic Categories.

h with function application:

$$\{(X/Y : f \;\; Y : g), \quad (Y : g \;\; : X\backslash Y : f) | h = f(g)\} \tag{10}$$

or by a reversal of the CCG composition combinators if f and g can be recombined to give h with function composition:

$$\{(X/Z : f \;\; Z/Y : g), \quad (Z\backslash Y : g \;\; : X\backslash Z : f) | h = \lambda x.f(g(x))\} \tag{11}$$

Unknown category names in the result of a split (Y in (10) and Z in (11)) are labelled via a functional mapping `cat` from semantic type T to syntactic category:

$$\texttt{cat}(T) = \left\{ \begin{array}{ll} \texttt{Atomic}(T) & \text{if } T \in \text{Figure 2} \\ \texttt{cat}(T_1)/\texttt{cat}(T_2) & \text{if } T = \langle T_1, T_2\rangle \\ \texttt{cat}(T_1)\backslash\texttt{cat}(T_2) & \text{if } T = \langle T_1, T_2\rangle \end{array} \right\}$$

which uses the `Atomic` function illustrated in Figure 2 to map semantic-type to basic CCG syntactic category. As an example, the logical split in (9) supports two CCG category splits, one for each of the CCG application rules.

$$(\mathsf{NP/N}{:}\lambda y.a(x, y(x)),\ \mathsf{N}{:}\lambda x.cookie(x)) \tag{12}$$

$$(\mathsf{N}{:}\lambda x.cookie(x),\ \mathsf{NP\backslash N}{:}\lambda y.a(x, y(x))) \tag{13}$$

The parse generation algorithm $\mathcal{T}$ uses the function `split` to generate all CCG category pairs that are an allowed split of an input category $\mathsf{X}{:}h$:

$$\{(\mathsf{C_l}{:}m_l, \mathsf{C_r}{:}m_r)\} = \texttt{split}(\mathsf{X}{:}h),$$

and then packs a chart representation of $\{t\}$ in a top-down fashion starting with a single cell entry $\mathsf{C_m} : m$ for the top node shared by all parses $\{t\}$. For the utterance and meaning in (1) the top parse node, spanning the entire word-string, is

$$\mathsf{S}{:}have(you, another(x, cookie(x))).$$

$\mathcal{T}$ cycles over all cell entries in increasingly small spans and populates the chart with their splits. For any cell entry $\mathsf{X}{:}h$ spanning more than one word $\mathcal{T}$ generates a set of pairs representing the splits of $\mathsf{X}{:}h$. For each split $(\mathsf{C_l}{:}m_l, \mathsf{C_r}{:}m_r)$ and every binary partition $(w_{i:k}, w_{k:j})$ of the word-span $\mathcal{T}$ creates two new cell entries in the chart: $(\mathsf{C_l}{:}m_l)_{i:k}$ and $(\mathsf{C_r}{:}m_r)_{k:j}$.

Input : Sentence $[w_1, \ldots, w_n]$, top node $\mathsf{C_m}{:}m$
Output: Packed parse chart `Ch` containing $\{t\}$
`Ch` $= [\,[\{\}_1, \ldots, \{\}_n]_1,\ \ldots\ , [\{\}_1, \ldots, \{\}_n]_n\,]$
`Ch`$[1][n-1] = \mathsf{C_m}{:}m$
for $i = n, \ldots, 2;\ j = 1 \ldots (n-i)+1$ **do**
  for $\mathsf{X}{:}h \in$ `Ch`$[j][i]$ **do**
    for $(\mathsf{C_l}{:}m_l, \mathsf{C_r}{:}m_r) \in \texttt{split}(\mathsf{X}{:}h)$ **do**
      for $k = 1, \ldots, i-1$ **do**
        `Ch`$[j][k] \leftarrow \mathsf{C_l}{:}m_l$
        `Ch`$[j+k][i-k] \leftarrow \mathsf{C_r}{:}m_r$

Algorithm 1: Generating $\{t\}$ with $\mathcal{T}$.

Algorithm 1 shows how the learner uses $\mathcal{T}$ to generate a packed chart representation of $\{t\}$ in the chart `Ch`. The function $\mathcal{T}$ massively overgenerates parses for any given natural language. The probabilistic parsing model introduced in Section 3 is used to choose the best parse from the overgenerated set.

5 Training

5.1 Parameter Estimation

The probabilistic model of the grammar describes a distribution over the observed training data $\mathbf{X}$, latent variables $\mathbf{S}$, and parameters θ. The goal of training is to estimate the posterior distribution:

$$p(\mathbf{S}, \theta|\mathbf{X}) = \frac{p(\mathbf{S}, \mathbf{X}|\theta)p(\theta)}{p(\mathbf{X})} \tag{14}$$

which we do with online Variational Bayesian Expectation Maximisation (oVBEM; Sato (2001), Hoffman et al. (2010)). oVBEM is an online

Bayesian extension of the EM algorithm that accumulates observation pseudocounts $n_{a\to b}$ for each of the productions $a \to b$ in the grammar. These pseudocounts define the posterior over production probabilities as follows:

$$(\theta_{a\to b_1}, \ldots, \theta_{a\to b_{\{k,\ldots\}}})) \mid \mathbf{X}, \mathbf{S} \sim \quad (15)$$
$$\mathrm{Dir}(\alpha H(b_1) + n_{a\to b_1}, \ldots, \sum_{j=k}^{\infty} \alpha H(b_j) + n_{a\to b_j})$$

These pseudocounts are computed in two steps:

oVBE-step For the training pair $(s_i, \{m\}_i)$ which supports the set of parses $\{t\}$, the expectation $E_{\{t\}}[a \to b]$ of each production $a \to b$ is calculated by creating a packed chart representation of $\{t\}$ and running the inside-outside algorithm. This is similar to the E-step in standard EM apart from the fact that each production is scored with the current *expectation* of its parameter weight $\hat{\theta}^{i-1}_{a\to b}$, where:

$$\hat{\theta}^{i-1}_{a\to b} = \frac{e^{\Psi\left(\alpha_a H_a(a\to b) + n^{i-1}_{a\to b}\right)}}{e^{\Psi\left(\sum_{\{b'\}}^{K} \alpha_a H_a(a\to b') + n^{i-1}_{a\to b'}\right)}} \quad (16)$$

and Ψ is the digamma function (Beal, 2003).

oVBM-step The expectations from the oVBE step are used to update the pseudocounts in Equation 15 as follows,

$$n^i_{a\to b} = n^{i-1}_{a\to b} + \eta_i(N \times E_{\{t\}}[a \to b] - n^{i-1}_{a\to b}) \quad (17)$$

where η_i is the learning rate and N is the size of the dataset.

5.2 The Training Algorithm

Now the training algorithm used to learn the lexicon Λ and pseudocounts $\{n_{a\to b}\}$ can be defined. The algorithm, shown in Algorithm 2, passes over the training data only once and one training instance at a time. For each $(s_i, \{m\}_i)$ it uses the function $\mathcal{T}$ $|\{m\}_i|$ times to generate a set of consistent parses $\{t\}'$. The lexicon is populated by using the `lex` function to read all of the lexical items off from the derivations in each $\{t\}'$. In the parameter update step, the training algorithm updates the pseudocounts associated with each of the productions $a \to b$ that have ever been seen during training according to Equation (17).

Only non-zero pseudocounts are stored in our model. The count vector is expanded with a new entry every time a new production is used. While the parameter update step cycles over all productions in $\{t\}$ it is not neccessary to store $\{t\}$, just the set of productions that it uses.

Input : Corpus $D = \{(s_i, \{m\}_i) | i = 1, \ldots, N\}$, Function $\mathcal{T}$, Semantics to syntactic category mapping `cat`, function `lex` to read lexical items off derivations.
Output: Lexicon Λ, Pseudocounts $\{n_{a\to b}\}$.
$\Lambda = \{\}$, $\{t\} = \{\}$
for $i = 1, \ldots, N$ **do**
  $\{t\}_i = \{\}$
  for $m' \in \{m\}_i$ **do**
    $\mathsf{C}_{\mathsf{m}'} = \mathtt{cat}(m')$
    $\{t\}' = \mathcal{T}(s_i, \mathsf{C}_{\mathsf{m}'} : m')$
    $\{t\}_i = \{t\}_i \cup \{t\}'$, $\{t\} = \{t\} \cup \{t\}'$
    $\Lambda = \Lambda \cup \mathtt{lex}(\{t\}')$
  for $a \to b \in \{t\}$ **do**
    $n^i_{a\to b} = n^{i-1}_{a\to b} + \eta_i(N \times E_{\{t\}_i}[a \to b] - n^{i-1}_{a\to b})$

Algorithm 2: Learning Λ and $\{n_{a\to b}\}$

6 Experimental Setup

6.1 Data

The Eve corpus, collected by Brown (1973), contains 14,124 English utterances spoken to a single child between the ages of 18 and 27 months. These have been hand annotated by Sagae et al. (2004) with labelled syntactic dependency graphs. An example annotation is shown in Figure 3.

While these annotations are designed to represent syntactic information, the parent-child relationships in the parse can also be viewed as a proxy for the predicate-argument structure of the semantics. We developed a template based deterministic procedure for mapping this predicate-argument structure onto logical expressions of the type discussed in Section 2.1. For example, the dependency graph in Figure 3 is automatically transformed into the logical expression

$$\lambda e.have(you, another(y, cookie(y)), e) \quad (18)$$
$$\wedge\, on(the(z, table(z)), e),$$

where e is a Davidsonian event variable used to deal with adverbial and prepositional attachments. The deterministic mapping to logical expressions uses 19 templates, three of which are used in this example: one for the verb and its arguments, one for the prepositional attachment and one (used twice) for the quantifier-noun constructions.

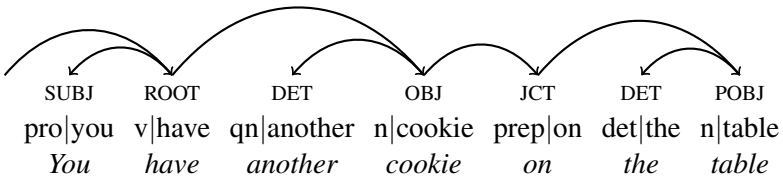


Figure 3: Syntactic dependency graph from Eve corpus.

This mapping from graph to logical expression makes use of a predefined dictionary of allowed, typed, logical constants. The mapping is successful for 31% of the child-directed utterances in the Eve corpus[3]. The remaining data is mostly accounted for by one-word utterances that have no straightforward interpretation in our typed logical language (e.g. *what; okay; alright; no; yeah; hmm; yes; uhhuh; mhm; thankyou*), missing verbal arguments that cannot be properly guessed from the context (largely in imperative sentences such as *drink the water*), and complex noun constructions that are hard to match with a small set of templates (e.g. as *top to a jar*). We also remove the small number of utterances containing more than 10 words for reasons of computational efficiency (see discussion in Section 8).

Following Alishahi and Stevenson (2010), we generate a context set $\{m\}_i$ for each utterance s_i by pairing that utterance with its correct logical expression along with the logical expressions of the preceding and following $(|\{m\}_i| - 1)/2$ utterances.

6.2 Base Distributions and Learning Rate

Each of the production heads a in the grammar requires a base distribution H_a and concentration parameter α_a. For word-productions the base distribution is a geometric distribution over character strings and spaces. For syntactic-productions the base distribution is defined in terms of the new category to be named by `cat` and the probability of splitting the rule by reversing either the *application* or *composition* combinators.

Semantic-productions' base distributions are defined by a probabilistic branching process conditioned on the type of the syntactic category. This distribution prefers less complex logical expressions. All concentration parameters are set to 1.0. The learning rate for parameter updates is $\eta_i = (0.8 + i)^{-0.5}$.

[3]Data available at www.tomkwiat.com/resources.html

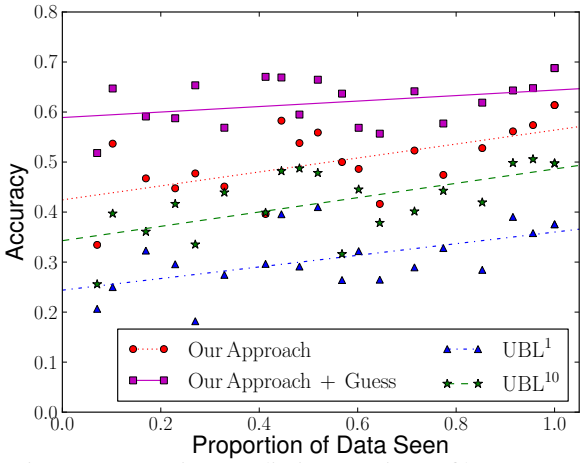


Figure 4: Meaning Prediction: Train on files $1, \ldots, n$ test on file $n + 1$.

7 Experiments

7.1 Parsing Unseen Sentences

We test the parsing model that is learnt by training on the first i files of the longitudinally ordered Eve corpus and testing on file $i + 1$, for $i = 1 \ldots 19$. For each utterance s' in the test file we use the parsing model to predict a meaning m^* and compare this to the target meaning m'. We report the proportion of utterances for which the prediction m^* is returned correctly both with and without word-meaning guessing. When a word has never been seen at training time our parser has the ability to 'guess' a typed logical meaning with placeholders for constant and predicate names.

For comparison we use the UBL semantic parser of Kwiatkowski et al. (2010) trained in a similar setting—i.e., with no language specific initialisation[4]. Figure 4 shows accuracy for our approach with and without guessing, for UBL

[4]Kwiatkowski et al. (2010) initialise lexical weights in their learning algorithm using corpus-wide alignment statistics across words and meaning elements. Instead we run UBL with small positive weight for all lexical items. When run with Giza++ parameter initialisations, UBL^{10} achieves 48.1% across folds compared to 49.2% for our approach.

when run over the training data once (UBL[1]) and for UBL when run over the training data 10 times (UBL[10]) as in Kwiatkowski et al. (2010). Each of the points represents accuracy on one of the 19 test files. All of these results are from parsers trained on utterances paired with a single candidate meaning. The lines of best fit show the upward trend in parser performance over time.

Despite only seeing each training instance once, our approach, due to its broader lexical search strategy, outperforms both versions of UBL which performs a greedy search in the space of lexicons and requires initialisation with co-occurence statistics between words and logical constants to guide this search. These statistics are not justified in a model of language acquisition and so they are not used here. The low performance of all systems is due largely to the sparsity of the data with 32.9% of all sentences containing a previously unseen word.

7.2 Word Learning

Due to the sparsity of the data, the training algorithm needs to be able to learn word-meanings on the basis of very few exposures. This is also a desirable feature from the perspective of modelling language acquisition as Carey and Bartlett (1978) have shown that children have the ability to learn word meanings on the basis of one, or very few, exposures through the process of *fast mapping*.

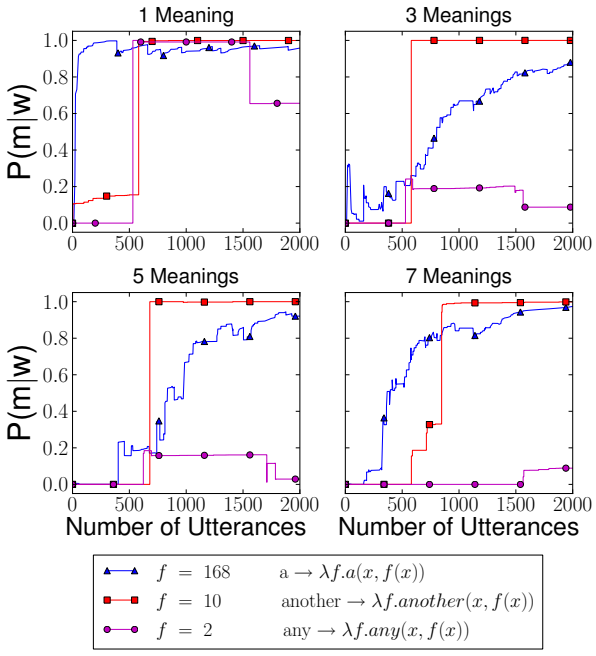


Figure 5: Learning quantifiers with frequency *f*.

Figure 5 shows the posterior probability of the correct meanings for the quantifiers 'a', 'another' and 'any' over the course of training with 1, 3, 5 and 7 candidate meanings for each utterance[5]. These three words are all of the same class but have very different frequencies in the training subset shown (168, 10 and 2 respectively). In all training settings, the word 'a' is learnt gradually from many observations but the rarer words 'another' and 'any' are learnt (when they are learnt) through large updates to the posterior on the basis of few observations. These large updates result from a *syntactic bootstrapping* effect (Gleitman, 1990). When the model has great confidence about the derivation in which an unseen lexical item occurs, the pseudocounts for that lexical item get a large update under Equation 17. This large update has a greater effect on rare words which are associated with small amounts of probability mass than it does on common ones that have already accumulated large pseudocounts. The fast learning of rare words later in learning correlates with observations of word learning in children.

7.3 Word Order Learning

Figure 6 shows the posterior probability of the correct SVO word order learnt from increasing amounts of training data. This is calculated by summing over all lexical items containing transitive verb semantics and sampling in the space of parse trees that could have generated them. With no propositional uncertainty in the training data the correct word order is learnt very quickly and stabilises. As the amount of propositional uncertainty increases, the rate at which this rule is learnt decreases. However, even in the face of ambiguous training data, the model can learn the correct word-order rule. The distribution over word orders also exhibits initial uncertainty, followed by a sharp convergence to the correct analysis. This ability to learn syntactic regularities abruptly means that our system is not subject to the criticisms that Thornton and Tesan (2007) levelled at statistical models of language acquisition—that their learning rates are too gradual.

[5]The term 'fast mapping' is generally used to refer to noun learning. We chose to examine quantifier learning here as there is a greater variation in quantifier frequencies. Fast mapping of nouns is also achieved.

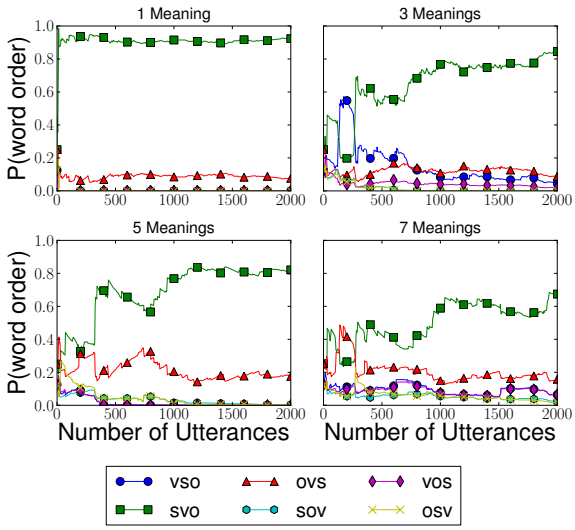


Figure 6: Learning SVO word order.

8 Discussion

We have presented an incremental model of language acquisition that learns a probabilistic CCG grammar from utterances paired with one or more potential meanings. The model assumes no language-specific knowledge, but does assume that the learner has access to language-universal correspondences between syntactic and semantic types, as well as a Bayesian prior encouraging grammars with heavy reuse of existing rules and lexical items. We have shown that this model not only outperforms a state-of-the-art semantic parser, but also exhibits learning curves similar to children's: lexical items can be acquired on a single exposure and word order is learnt suddenly rather than gradually.

Although we use a Bayesian model, our approach is different from many of the Bayesian models proposed in cognitive science and language acquisition (Xu and Tenenbaum, 2007; Goldwater et al., 2009; Frank et al., 2009; Griffiths and Tenenbaum, 2006; Griffiths, 2005; Perfors et al., 2011). These models are intended as *ideal observer* analyses, demonstrating what would be learned by a probabilistically optimal learner. Our learner uses a more cognitively plausible but approximate online learning algorithm. In this way, it is similar to other cognitively plausible approximate Bayesian learners (Pearl et al., 2010; Sanborn et al., 2010; Shi et al., 2010).

Of course, despite the incremental nature of our learning algorithm, there are still many aspects that could be criticized as cognitively implausible. In particular, it generates all parses consistent with each training instance, which can be both memory- and processor-intensive. It is unlikely that children do this once they have learnt at least some of the target language. In future, we plan to investigate more efficient parameter estimation methods. One possibility would be an approximate oVBEM algorithm in which the expectations in Equation 17 are calculated according to a high probability subset of the parses $\{t\}$. Another option would be particle filtering, which has been investigated as a cognitively plausible method for approximate Bayesian inference (Shi et al., 2010; Levy et al., 2009; Sanborn et al., 2010).

As a crude approximation to the context in which an utterance is heard, the logical representations of meaning that we present to the learner are also open to criticism. However, Steedman (2002) argues that children do have access to structured meaning representations from a much older apparatus used for planning actions and we wish to eventually ground these in sensory input.

Despite the limitations listed above, our approach makes several important contributions to the computational study of language acquisition. It is the first model to learn syntax and semantics concurrently; previous systems (Villavicencio, 2002; Buttery, 2006) learnt categorial grammars from sentences where all word meanings were known. Our model is also the first to be evaluated by parsing sentences onto their meanings, in contrast to the work mentioned above and that of Gibson and Wexler (1994), Siskind (1992) Sakas and Fodor (2001), and Yang (2002). These all evaluate their learners on the basis of a small number of predefined syntactic parameters.

Finally, our work addresses a misunderstanding about statistical learners—that their learning curves must be gradual (Thornton and Tesan, 2007). By demonstrating sudden learning of word order and fast mapping, our model shows that statistical learners can account for sudden changes in children's grammars. In future, we hope to extend these results by examining other learning behaviors and testing the model on other languages.

9 Acknowledgements

We thank Mark Johnson for suggesting an analysis of learning rates. This work was funded by the ERC Advanced Fellowship 24952 GramPlus and EU IP grant EC-FP7-270273 Xperience.

References

Alishahi and Stevenson, S. (2008). A computational model for early argument structure acquisition. *Cognitive Science*, 32:5:789–834.

Alishahi, A. and Stevenson, S. (2010). Learning general properties of semantic roles from usage data: a computational model. *Language and Cognitive Processes*, 25:1.

Beal, M. J. (2003). Variational algorithms for approximate Bayesian inference. Technical report, Gatsby Institute, UCL.

Börschinger, B., Jones, B. K., and Johnson, M. (2011). Reducing grounded learning tasks to grammatical inference. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 1416–1425, Edinburgh, Scotland, UK. Association for Computational Linguistics.

Brown, R. (1973). *A First Language: the Early Stages*. Harvard University Press, Cambridge MA.

Buttery, P. J. (2006). Computational models for first language acquisition. Technical Report UCAM-CL-TR-675, University of Cambridge, Computer Laboratory.

Carey, S. and Bartlett, E. (1978). Acquring a single new word. *Papers and Reports on Child Language Development*, 15.

Chen, D. L., Kim, J., and Mooney, R. J. (2010). Training a multilingual sportscaster: Using perceptual context to learn language. *J. Artif. Intell. Res. (JAIR)*, 37:397–435.

Fazly, A., Alishahi, A., and Stevenson, S. (2010). A probabilistic computational model of cross-situational word learning. *Cognitive Science*, 34(6):1017–1063.

Frank, M., Goodman, S., and Tenenbaum, J. (2009). Using speakers referential intentions to model early cross-situational word learning. *Psychological Science*, 20(5):578–585.

Frank, M. C., Goodman, N. D., and Tenenbaum, J. B. (2008). A bayesian framework for cross-situational word-learning. *Advances in Neural Information Processing Systems 20*.

Gibson, E. and Wexler, K. (1994). Triggers. *Linguistic Inquiry*, 25:355–407.

Gleitman, L. (1990). The structural sources of verb meanings. *Language Acquisition*, 1:1–55.

Goldwater, S., Griffiths, T. L., and Johnson, M. (2009). A Bayesian framework for word segmentation: Exploring the effects of context. *Cognition*, 112(1):21–54.

Griffiths, T. L., . T. J. B. (2005). Structure and strength in causal induction. *Cognitive Psychology*, 51:354–384.

Griffiths, T. L. and Tenenbaum, J. B. (2006). Optimal predictions in everyday cognition. *Psychological Science*.

Hoffman, M., Blei, D. M., and Bach, F. (2010). Online learning for latent dirichlet allocation. In *NIPS*.

Kate, R. J. and Mooney, R. J. (2007). Learning language semantics from ambiguous supervision. In *Proceedings of the 22nd Conference on Artificial Intelligence (AAAI-07)*.

Kwiatkowski, T., Zettlemoyer, L., Goldwater, S., and Steedman, M. (2010). Inducing probabilistic CCG grammars from logical form with higher-order unification. In *Proceedings of the Conference on Emperical Methods in Natural Language Processing*.

Kwiatkowski, T., Zettlemoyer, L., Goldwater, S., and Steedman, M. (2011). Lexical generalization in ccg grammar induction for semantic parsing. In *Proceedings of the Conference on Emperical Methods in Natural Language Processing*.

Levy, R., Reali, F., and Griffiths, T. (2009). Modeling the effects of memory on human online sentence processing with particle filters. In *Advances in Neural Information Processing Systems 21*.

Lu, W., Ng, H. T., Lee, W. S., and Zettlemoyer, L. S. (2008). A generative model for parsing natural language to meaning representations. In *Proceedings of The Conference on Empirical Methods in Natural Language Processing*.

MacWhinney, B. (2000). *The CHILDES project: tools for analyzing talk*. Lawrence Erlbaum, Mahwah, NJ u.a. EN.

Maurits, L., Perfors, A., and Navarro, D. (2009). Joint acquisition of word order and word reference. In *Proceedings of the 31th Annual Conference of the Cognitive Science Society*.

Pearl, L., Goldwater, S., and Steyvers, M. (2010). How ideal are we? Incorporating human limi-

tations into Bayesian models of word segmentation. pages 315–326, Somerville, MA. Cascadilla Press.

Perfors, A., Tenenbaum, J. B., and Regier, T. (2011). The learnability of abstract syntactic principles. *Cognition*, 118(3):306 – 338.

Sagae, K., MacWhinney, B., and Lavie, A. (2004). Adding syntactic annotations to transcripts of parent-child dialogs. In *Proceedings of the 4th International Conference on Language Resources and Evaluation*. Lisbon, LREC.

Sakas, W. and Fodor, J. D. (2001). The structural triggers learner. In Bertolo, S., editor, *Language Acquisition and Learnability*, pages 172–233. Cambridge University Press, Cambridge.

Sanborn, A. N., Griffiths, T. L., and Navarro, D. J. (2010). Rational approximations to rational models: Alternative algorithms for category learning. *Psychological Review*.

Sato, M. (2001). Online model selection based on the variational bayes. *Neural Computation*, 13(7):1649–1681.

Shi, L., Griffiths, T. L., Feldman, N. H., and Sanborn, A. N. (2010). Exemplar models as a mechanism for performing bayesian inference. *Psychonomic Bulletin & Review*, 17(4):443–464.

Siskind, J. M. (1992). *Naive Physics, Event Perception, Lexical Semantics, and Language Acquisition*. PhD thesis, Massachusetts Institute of Technology.

Siskind, J. M. (1996). A computational study of cross-situational techniques for learning word-to-meaning mappings. *Cognition*, 61(1-2):1–38.

Steedman, M. (2000). *The Syntactic Process*. MIT Press, Cambridge, MA.

Steedman, M. (2002). Plans, affordances, and combinatory grammar. *Linguistics and Philosophy*, 25.

Thornton, R. and Tesan, G. (2007). Categorical acquisition: Parameter setting in universal grammar. *Biolinguistics*, 1.

Villavicencio, A. (2002). The acquisition of a unification-based generalised categorial grammar. Technical Report UCAM-CL-TR-533, University of Cambridge, Computer Laboratory.

Wong, Y. W. and Mooney, R. (2006). Learning for semantic parsing with statistical machine translation. In *Proceedings of the Human Language Technology Conference of the NAACL*.

Wong, Y. W. and Mooney, R. (2007). Learning synchronous grammars for semantic parsing with lambda calculus. In *Proceedings of the Association for Computational Linguistics*.

Xu, F. and Tenenbaum, J. B. (2007). Word learning as Bayesian inference. *Psychological Review*, 114:245–272.

Yang, C. (2002). *Knowledge and Learning in Natural Language*. Oxford University Press, Oxford.

Yu, C. and Ballard, D. H. (2007). A unified model of early word learning: Integrating statistical and social cues. *Neurocomputing*, 70(13-15):2149 – 2165.

Zettlemoyer, L. S. and Collins, M. (2005). Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence*.

Zettlemoyer, L. S. and Collins, M. (2007). Online learning of relaxed CCG grammars for parsing to logical form. In *Proc. of the Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*.

Zettlemoyer, L. S. and Collins, M. (2009). Learning context-dependent mappings from sentences to logical form. In *Proceedings of The Joint Conference of the Association for Computational Linguistics and International Joint Conference on Natural Language Processing*.

Lexical Generalization in CCG Grammar Induction for Semantic Parsing

Tom Kwiatkowski[*] t.m.kwiatkowksi@sms.ed.ac.uk
Luke Zettlemoyer[†] lsz@cs.washington.edu
Sharon Goldwater[*] sgwater@inf.ed.ac.uk
Mark Steedman[*] steedman@inf.ed.ac.uk

[*]School of Informatics
University of Edinburgh
Edinburgh, EH8 9AB, UK

[†]Computer Science & Engineering
University of Washington
Seattle, WA 98195

Abstract

We consider the problem of learning factored probabilistic CCG grammars for semantic parsing from data containing sentences paired with logical-form meaning representations. Traditional CCG lexicons list lexical items that pair words and phrases with syntactic and semantic content. Such lexicons can be inefficient when words appear repeatedly with closely related lexical content. In this paper, we introduce factored lexicons, which include both *lexemes* to model word meaning and *templates* to model systematic variation in word usage. We also present an algorithm for learning factored CCG lexicons, along with a probabilistic parse-selection model. Evaluations on benchmark datasets demonstrate that the approach learns highly accurate parsers, whose generalization performance benefits greatly from the lexical factoring.

1 Introduction

Semantic parsers automatically recover representations of meaning from natural language sentences. Recent work has focused on learning such parsers directly from corpora made up of sentences paired with logical meaning representations (Kate et al., 2005; Kate and Mooney, 2006; Wong and Mooney, 2006, 2007; Zettlemoyer and Collins, 2005, 2007; Lu et al., 2008; Kwiatkowski et al., 2010).

For example, in a flight booking domain we might have access to training examples such as:

Sentence: I want flights from Boston
Meaning: $\lambda x.flight(x) \wedge from(x, bos)$

and the goal is to learn a grammar that can map new, unseen, sentences onto their corresponding meanings, or logical forms.

One approach to this problem has developed algorithms for leaning probabilistic CCG grammars (Zettlemoyer and Collins, 2005, 2007; Kwiatkowski et al., 2010). These grammars are well-suited to the task of semantic parsing, as they closely link syntax and semantics. They can be used to model a wide range of complex linguistic phenomena and are strongly lexicalized, storing all language-specific grammatical information directly with the words in the lexicon. For example, a typical learned lexicon might include entries such as:

(1) $flight \vdash N : \lambda x.flight(x)$
(2) $flight \vdash N/(S|NP) : \lambda f \lambda x.flight(x) \wedge f(x)$
(3) $flight \vdash N\backslash N : \lambda f \lambda x.flight(x) \wedge f(x)$
(4) $fare \vdash N : \lambda x.cost(x)$
(5) $fare \vdash N/(S|NP) : \lambda f \lambda x.cost(x) \wedge f(x)$
(6) $fare \vdash N\backslash N : \lambda f \lambda x.cost(x) \wedge f(x)$
(7) $Boston \vdash NP : bos$
(8) $Boston \vdash N\backslash N : \lambda f \lambda x.from(x, bos) \wedge f(x)$
(9) $New\ York \vdash NP : nyc$
(10) $New\ York \vdash N\backslash N : \lambda f \lambda x.from(x, nyc) \wedge f(x)$

Although lexicalization of this kind is useful for learning, as we will see, these grammars can also suffer from sparsity in the training data, since closely related entries must be repeatedly learned for all members of a certain class of words. For example, the list above shows a selection of lexical items that would have to be learned separately.

In this list, the word "flight" is paired with the predicate *flight* in three separate lexical items which are required for different syntactic contexts. Item

Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing, pages 1512–1523,
Edinburgh, Scotland, UK, July 27–31, 2011.

(1) has the standard *N* category for entries of this type, item (2) allows the use of the word "flight" with that-less relative clauses such as "flight departing Boston", and item (3) is useful for phrases with unconventional word order such as "from Boston flight to New York". Representing these three lexical items separately is inefficient, since each word of this class (such as "fare") will require three similarly structured lexical entries differing only in predicate name. There may also be systemtatic semantic variation between entries for a certain class of words. For example, in (6) "Boston" is paired with the constant *bos* that represents its meaning. However, item (7) also adds the predicate *from* to the logical form. This might be used to analyse somewhat elliptical, unedited sentences such as "Show me flights Boston to New York," which can be challenging for semantic parsers (Zettlemoyer and Collins, 2007).

This paper builds upon the insight that a large proportion of the variation between lexical items for a given class of words is systematic. Therefore it should be represented once and applied to a small set of basic lexical units. [1] We develop a *factored lexicon* that captures this insight by distinguishing *lexemes*, which pair words with logical constants, from *lexical templates*, which map lexemes to full lexical items. As we will see, this can lead to a significantly more compact lexicon that can be learned from less data. Each word or phrase will be associated with a few lexemes that can be combined with a shared set of general templates.

We develop an approach to learning factored, probabilistic CCG grammars for semantic parsing. Following previous work (Kwiatkowski et al., 2010), we make use of a higher-order unification learning scheme that defines a space of CCG grammars consistent with the (sentence, logical form) training pairs. However, instead of constructing fully specified lexical items for the learned grammar, we automatically generate sets of lexemes and lexical templates to model each example. This is a difficult learning problem, since the CCG analyses that are required to construct the final meaning representations are not explicitly labeled in the training data. Instead, we model them with hidden variables and develop an online learning approach that simultaneously estimates the parameters of a log-linear parsing model, while inducing the factored lexicon.

We evaluate the approach on the benchmark Atis and GeoQuery domains. This is a challenging setup, since the GeoQuery data has complex meaning representations and sentences in multiple languages, while the Atis data contains spontaneous, unedited text that can be difficult to analyze with a formal grammar representation. Our approach achieves at or near state-of-the-art recall across all conditions, despite having no English or domain-specific information built in. We believe that ours is the only system of sufficient generality to run with this degree of success on all of these datasets.

2 Related work

There has been significant previous work on learning semantic parsers from training sentences labelled with logical form meaning representations.

We extend a line of research that has addressed this problem by developing CCG grammar induction techniques. Zettlemoyer and Collins (2005, 2007) presented approaches that use hand generated, English-language specific rules to generate lexical items from logical forms as well as English specific type-shifting rules and relaxations of the CCG combinators to model spontaneous, unedited sentences. Zettlemoyer and Collins (2009) extends this work to the case of learning in context dependent environments. Kwiatkowski et al. (2010) described an approach for language-independent learning that replaces the hand-specified templates with a higher-order-unification-based lexical induction method, but their approach does not scale well to challenging, unedited sentences. The learning approach we develop for inducing factored lexicons is also language independent, but scales well to these challenging sentences.

There have been a number of other approaches for learning semantic parsers, including ones based on machine translation techniques (Papineni et al., 1997; Ramaswamy and Kleindienst, 2000; Wong and Mooney, 2006), parsing models (Miller et al., 1996; Ge and Mooney, 2006; Lu et al., 2008), in-

[1] A related tactic is commonly used in wide-coverage CCG parsers derived from treebanks, such as work by Hockenmaier and Steedman (2002) and Clark and Curran (2007). These parsers make extensive use of category-changing unary rules, to avoid data sparsity for systematically related categories (such as those related by type-raising). We will automatically learn to represent these types of generalizations in the factored lexicon.

ductive logic programming algorithms (Zelle and Mooney, 1996; Thompson and Mooney, 2002; Tang and Mooney, 2000), probabilistic automata (He and Young, 2005, 2006), and ideas from string kernels and support vector machines (Kate and Mooney, 2006; Nguyen et al., 2006).

More recent work has focused on training semantic parsers without supervision in the form of logical-form annotations. Clarke et al. (2010) and Liang et al. (2011) replace semantic annotations in the training set with target answers which are more easily available. Goldwasser et al. (2011) present work on unsupervised learning of logical form structure. However, all of these systems require significantly more domain and language specific initialization than the approach presented here.

Other work has learnt semantic analyses from text in the context of interactions in computational environments (Branavan et al. (2010), Vogel and Jurafsky (2010)); text grounded in partial observations of a world state (Liang et al., 2009); and from raw text alone (Poon and Domingos, 2009, 2010).

There is also related work that uses the CCG grammar formalism. Clark and Curran (2003) present a method for learning the parameters of a log-linear CCG parsing model from fully annotated normal–form parse trees. Watkinson and Manandhar (1999) describe an unsupervised approach for learning syntactic CCG lexicons. Bos et al. (2004) present an algorithm for building semantic representations from CCG parses but requires fully–specified CCG derivations in the training data.

3 Overview of the Approach

Here we give a formal definition of the problem and an overview of the learning approach.

Problem We will learn a semantic parser that takes a sentences x and returns a logical form z representing its underlying meaning. We assume we have input data $\{(x_i, z_i) | i = 1 \ldots n\}$ containing sentences x_i and logical forms z_i, for example $x_i =$"Show me flights to Boston" and $z_i = \lambda x.flight(x) \wedge to(x, bos)$.

Model We will represent the parser as a factored, probabilistic CCG (PCCG) grammar. A traditional CCG lexical item would fully specify the syntax and semantics for a word (reviewed in Section 4). For example, Boston $\vdash NP : bos$ represents the entry for the word "Boston" with syntactic category NP and meaning represented by the constant bos. Where a lexicon would usually list lexical items such as this, we instead use a factored lexicon (L, T) containing:

- A list of lexemes L. Each lexeme pairs a word or phrase with a list of logical constants that can be used to construct its meaning. For example, one lexeme might be $(\text{Boston}, [bos])$.
- A list of lexical templates T. Each template takes a lexeme and maps it on to a full lexical item. For example, there is a single template that can map the lexeme above to the final lexical entry Boston $\vdash NP : bos$.

We will make central use of this factored representation to provide a more compact representation of the lexicon that can be learned efficiently.

The factored PCCG will also contain a parameter vector, θ, that defines a log-linear distribution over the possible parses y, conditioned on the sentence x.

Learning Our approach for learning factored PCCGs extends the work of Kwiatkowski et al. (2010), as reviewed in Section 7. Specifically, we modify the lexical learning, to produce lexemes and templates, as well as the feature space of the model, but reuse the existing parameter estimation techniques and overall learning cycle, as described in Section 7.

We present the complete approach in three parts by describing the factored representation of the lexicon (Section 5), techniques for proposing potential new lexemes and templates (Section 6), and finally a complete learning algorithm (Section 7). However, the next section first reviews the required background on semantic parsing with CCG.

4 Background

4.1 Lambda Calculus

We represent the meanings of sentences, words and phrases with logical expressions that can contain constants, quantifiers, logical connectors and lambda abstractions. We construct the meanings of sentences from the meanings of words and phrases using lambda-calculus operations. We use a version of the typed lambda calculus (Carpenter, 1997), in which the basic types include e, for entities; t, for truth values; and i for numbers. We also have function types that are assigned to lambda expressions.

The expression $\lambda x.flight(x)$ takes an entity and returns a truth value, and has the function type $\langle e,t \rangle$.

4.2 Combinatory Categorial Grammar

CCG (Steedman, 1996, 2000) is a linguistic formalism that tightly couples syntax and semantics, and can be used to model a wide range of language phenomena. A traditional CCG grammar includes a lexicon Λ with entries like the following:

$$\begin{aligned} &flights \vdash N : \lambda x.flight(x) \\ &\quad to \vdash (N\backslash N)/NP : \lambda y.\lambda f.\lambda x.f(x) \wedge to(x,y) \\ &Boston \vdash NP : bos \end{aligned}$$

where each lexical item w$\vdash X : h$ has words w, a syntactic category X, and a logical form h. For the first example, these are "flights," N, and $\lambda x.flight(x)$. In this paper, we introduce a new way of representing lexical items as (*lexeme*, *template*) pairs, as described in section 5.

CCG syntactic categories may be atomic (such as S or NP) or complex (such as $(N\backslash N)/NP$) where the slash combinators encode word order information. CCG uses a small set of *combinatory rules* to build syntactic parses and semantic representations *concurrently*. Two example combinatory rules are forward ($>$) and backward ($<$) *application*:

$$\begin{aligned} X/Y : f \quad Y : g \quad &\Rightarrow \quad X : f(g) \qquad (>) \\ Y : g \quad X\backslash Y : f \quad &\Rightarrow \quad X : f(g) \qquad (<) \end{aligned}$$

These rules apply to build syntactic and semantic derivations under the control of the word order information encoded in the slash directions of the lexical entries. For example, given the lexicon above, the phrase "flights to Boston" can be parsed to produce:

$$\dfrac{\dfrac{\begin{array}{c}\text{flights}\\ \hline N\\ \lambda x.flight(x)\end{array} \quad \dfrac{\begin{array}{c}\text{to}\\ \hline (N\backslash N)/NP\\ \lambda y\lambda f\lambda x.f(x)\wedge to(x,y)\end{array} \quad \begin{array}{c}\text{Boston}\\ \hline NP\\ bos\end{array}}{(N\backslash N)} >}{\begin{array}{c} \lambda f\lambda x.f(x)\wedge to(x,bos)\end{array}}}{\begin{array}{c}N\\ \lambda x.flight(x)\wedge to(x,bos)\end{array}} <$$

where each step in the parse is labeled with the combinatory rule ($->$ or $-<$) that was used.

CCG also includes combinatory rules of forward ($>\mathbf{B}$) and backward ($<\mathbf{B}$) *composition*:

$$\begin{aligned} X/Y : f \quad Y/Z : g &\Rightarrow X/Z : \lambda x.f(g(x)) \quad (>\mathbf{B}) \\ Y\backslash Z : g \quad X\backslash Y : f &\Rightarrow X\backslash Z : \lambda x.f(g(x)) \quad (<\mathbf{B}) \end{aligned}$$

These rules allow a relaxed notion of constituency which helps limit the number of distinct CCG lexical items required.

To the standard forward and backward slashes of CCG we also add a vertical slash for which the direction of application is underspecified. We shall see examples of this in Section 10.

4.3 Probabilistic CCGs

Due to ambiguity in both the CCG lexicon and the order in which combinators are applied, there will be many parses for each sentence. We discriminate between competing parses using a log-linear model which has a feature vector ϕ and a parameter vector θ. The probability of a parse y that returns logical form z, given a sentence x is defined as:

$$P(y,z|x;\theta,\Lambda) = \frac{e^{\theta\cdot\phi(x,y,z)}}{\sum_{(y',z')} e^{\theta\cdot\phi(x,y',z')}} \qquad (1)$$

Section 8 fully defines the set of features used in the system presented. The most important of these control the generation of lexical items from (lexeme, template) pairs. Each (lexeme, template) pair used in a parse fires three features as we will see in more detail later.

The parsing, or inference, problem done at test time requires us to find the most likely logical form z given a sentence x, assuming the parameters θ and lexicon Λ are known:

$$f(x) = \arg\max_z p(z|x;\theta,\Lambda) \qquad (2)$$

where the probability of the logical form is found by summing over all parses that produce it:

$$p(z|x;\theta,\Lambda) = \sum_y p(y,z|x;\theta,\Lambda) \qquad (3)$$

In this approach the distribution over parse trees y is modeled as a hidden variable. The sum over parses in Eq. 3 can be calculated efficiently using the inside-outside algorithm with a CKY-style parsing algorithm.

To estimate the parameters themselves, we use stochastic gradient updates (LeCun et al., 1998). Given a set of n sentence-meaning pairs $\{(x_i,z_i) : i = 1...n\}$, we update the parameters θ iteratively, for each example i, by following the local gradient of the conditional log-likelihood objective

$O_i = \log P(z_i|x_i;\theta,\Lambda)$. The local gradient of the individual parameter θ_j associated with feature ϕ_j and training instance (x_i, z_i) is given by:

$$\frac{\partial O_i}{\partial \theta_j} = E_{p(y|x_i,z_i;\theta,\Lambda)}[\phi_j(x_i,y,z_i)] - E_{p(y,z|x_i;\theta,\Lambda)}[\phi_j(x_i,y,z)] \quad (4)$$

As with Eq. 3, all of the expectations in Eq. 4 are calculated through the use of the inside-outside algorithm on a pruned parse chart. For a sentence of length m, each parse chart span is pruned using a beam width proportional to $m^{\frac{2}{3}}$, to allow larger beams for shorter sentences.

5 Factored Lexicons

A factored lexicon includes a set L of lexemes and a set T of lexical templates. In this section, we formally define these sets, and describe how they are used to build CCG parses. We will use a set of lexical items from our running example to discuss the details of how the following lexical items:

(1) $flight \vdash N : \lambda x.flight(x)$
(2) $flight \vdash N/(S|NP) : \lambda f \lambda x.flight(x) \wedge f(x)$
...
(6) $Boston \vdash NP : bos$
(7) $Boston \vdash N\backslash N : \lambda f \lambda x.from(x, bos) \wedge f(x)$

are constructed from specific lexemes and templates.

5.1 Lexemes

A lexeme $(\mathrm{w}, \vec{c})$ pairs a word sequence w with an ordered list of logical constants $\vec{c} = [c_1 \ldots c_m]$. For example, item (1) and (2) above would come from a single lexeme $(\text{flight}, [flight])$. Similar lexemes would be represented for other predicates, for example $(\text{fare}, [cost])$. Lexemes also can contain multiple constants, for example $(\text{cheapest}, [argmin, cost])$, which we will see more examples of later.

5.2 Lexical Templates

A lexical template takes a lexeme and produces a lexical item. Templates have the general form

$$\lambda(\omega, \vec{v}).[\omega \vdash X : h_{\vec{v}}]$$

where $h_{\vec{v}}$ is a logical expression that contains variables from the list $\vec{v}$. Applying this template to the input lexeme $(\mathrm{w}, \vec{c})$ gives the full lexical item $\mathrm{w} \vdash X : h$ where the variable ω has been replaced with the wordspan w and the logical form h has been created by replacing each of the variables in $\vec{v}$ with the counterpart constant from $\vec{c}$. For example, the lexical item (6) above would be constructed from the lexeme $(\text{Boston}, [bos])$ using the template $\lambda(\omega, \vec{v}).[\omega \vdash NP : v_1]$. Items (1) and (2) would both be constructed from the single lexeme $(\text{flight}, [flight])$ with the two different templates $\lambda(\omega, \vec{v}).[\omega \vdash N : \lambda x.v_1(x)]$ and $\lambda(\omega, \vec{v}).[\omega \vdash N/(S|NP) : \lambda f \lambda x.v_1(x) \wedge f(x)]$

5.3 Parsing with a Factored Lexicon

In general, there can by many different (lexeme, template) pairs that produce the same lexical item. For example, lexical item (7) in our running example above can be constructed from the lexemes $(\text{Boston}, [bos])$ and $(\text{Boston}, [from, bos])$, given appropriate templates.

To model this ambiguity, we include the selection of a (lexeme, template) pair as a decision to be made while constructing a CCG parse tree. Given the lexical item produced by the chosen lexeme and template, parsing continues with the traditional combinators, as reviewed in Section 4.2. This direct integration allows for features that signal which lexemes and templates have been used while also allowing for well defined marginal probabilities, by summing over all ways of deriving a specific lexical item.

6 Learning Factored Lexicons

To induce factored lexicons, we will make use of two procedures, presented in this section, that factor lexical items into lexemes and templates. Section 7 will describe how this factoring operation is integrated into the complete learning algorithm.

6.1 Maximal Factorings

Given a lexical item l of the form $w \vdash X : h$ with words w, a syntactic category X, and a logical form h, we define the *maximal factoring* to be the unique (lexeme, template) pair that can be used to reconstruct l and includes all of the constants of h in the lexeme (listed in a fixed order based on an ordered traversal of h). For example, the maximal factoring for the lexical item $Boston \vdash NP : bos$ is the pair we saw before: $(\text{Boston}, [bos])$ and $\lambda(\omega, \vec{v}).[\omega \vdash NP : v_1]$. Similarly, the lexical item $Boston \vdash N\backslash N : \lambda f.\lambda x.f(x) \wedge from(x, bos)$ would be factored to produce $(\text{Boston}, [from, bos])$ and $\lambda(\omega, \vec{v}).[\omega \vdash N\backslash N : \lambda f.\lambda x.f(x) \wedge v_1(x, v_2)]$.

As we will see in Section 7, this notion of factor-

ing can be directly incorporated into existing algorithms that learn CCG lexicons. When the original algorithm would have added an entry l to the lexicon, we can instead compute the factoring of l and add the corresponding lexeme and template to the factored lexicon.

6.2 Introducing Templates with Content

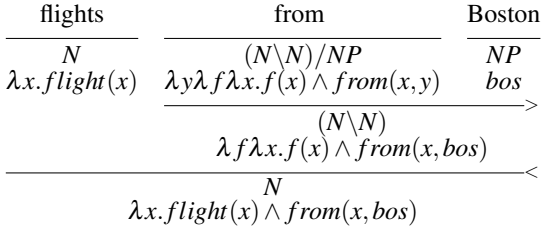
Maximal factorings, as just described, provide for significant lexical generalization but do not handle all of the cases needed to learn effectively. For instance, the maximal split for the item $Boston \vdash N\backslash N : \lambda f.\lambda x.f(x) \wedge from(x,bos)$ would introduce the lexeme $(\text{Boston}, [from, bos])$, which is suboptimal since each possible city would need a lexeme of this type, with the additional *from* constant included. Instead, we would ideally like to learn the lexeme $(\text{Boston}, [bos])$ and have a template that introduces the *from* constant. This would model the desired generalization with a single lexeme per city.

In order to permit the introduction of extra constants into lexical items, we allow the creation of templates that contain logical constants through *partial factorings*. For instance, the template below can introduce the predicate *from*

$$\lambda(\omega,\vec{v}).[\omega \vdash N\backslash N : \lambda f.\lambda x.f(x) \wedge from(x,v_1)]$$

The use of templates to introduce extra semantic constants into a lexical item is similar to, but more general than, the English-specific *type-shifting* rules used in Zettlemoyer and Collins (2007), which were introduced to model spontaneous, unedited text. They are useful, as we will see, in learning to recover semantic content that is implied, but not explicitly stated, such as our original motivating phrase "flights Boston to New York."

To propose templates which introduce semantic content, during learning, we build on the intuition that we need to recover from missing words, such as in the example above. In this scenario, there should also be other sentences that actually include the word, in our example this would be something like "flights from Boston." We will also assume that we have learned a good factored lexicon for the complete example that could produce the parse:



Given analyses of this form, we introduce new templates that will allow us to recover from missing words, for example if "from" was dropped. We identify commonly occurring nodes in the best parse trees found during training, in this case the nonterminal spanning "from Boston," and introduce templates that can produce the nonterminal, even if one of the words is missing. Here, this approach would introduce the desired template $\lambda(\omega,\vec{v}).[\omega \vdash N\backslash N : \lambda f.\lambda x.f(x) \wedge from(x,v_1)]$ for mapping the lexeme $(\text{Boston}, [bos])$ directly to the intermediate structure.

Not all templates introduced this way will model valid generalizations. However, we will incorporate them into a learning algorithm with indicator features that can be weighted to control their use. The next section presents the complete approach.

7 Learning Factored PCCGs

Our Factored Unification Based Learning (FUBL) method extends the UBL algorithm (Kwiatkowski et al., 2010) to induce factored lexicons, while also simultanously estimating the parameters of a log-linear CCG parsing model. In this section, we first review the NEW-LEX lexical induction procedure from UBL, and then present the FUBL algorithm.

7.1 Background: NEW-LEX

NEW-LEX generates lexical items by *splitting* and *merging* nodes in the best parse tree of each training example. Each parse node has a CCG category $X : h$ and a sequence of words w that it spans. We will present an overview of the approach using the running example with the phrase $w =$"in Boston" and the category $X : h = S\backslash NP : \lambda x.loc(x,bos)$, which is of the type commonly seen during learning. The splitting procedure is a two step process that first splits the logical form h, then splits the CCG syntactic category X and finally splits the string w.

The first step enumerates all possible splits of the logical form h into a pair of new expressions

(f, g) that can be used to reconstruct h by either function application ($h = f(g)$) or composition ($h = \lambda x.f(g(x))$). For example, one possible split is:

$$(f = \lambda y.\lambda x.loc(x,y)\ ,\ g = bos)$$

which corresponds to the function application case.

The next two steps enumerate all ways of splitting the syntactic category X and words w to introduce two new lexical items which can be recombined with CCG combinators (application or composition) to recreate the original parse node $X : h$ spanning w. In our example, one possibility would be:

$$(\text{in} \vdash (S\backslash NP)/NP : \lambda y.\lambda x.loc(x,y)\ ,\ \text{Boston} \vdash NP : bos)$$

which could be recombined with the forward application combinator from Section 4.2.

To assign categories while splitting, the grammar used by NEW-LEX only uses two atomic syntactic categories S and NP. This allows NEW-LEX to make use of a direct mapping from semantic type to syntactic category when proposing syntactic categories. In this schema, the standard syntactic category N is replaced by the category $S|NP$ which matches the type $\langle e,t \rangle$ and uses the vertical slash introduced in Section 4.2. We will see categories such as this in the evaluation.

7.2 The FUBL Algorithm

Figure 1 shows the FUBL learning algorithm. We assume training data $\{(x_i, z_i) : i = 1 \ldots n\}$ where each example is a sentence x_i paired with a logical form z_i. The algorithm induces a factored PCCG, including the lexemes L, templates T, and parameters θ.

The algorithm is online, repeatedly performing both lexical expansion (Step 1) and a parameter update (Step 2) for each training example. The overall approach is closely related to the UBL algorithm (Kwiatkowski et al., 2010), but includes extensions for updating the factored lexicon, as motivated in Section 6.

Initialization The model is initialized with a factored lexicon as follows. MAX-FAC is a function that takes a lexical item l and returns the maximal factoring of it, that is the unique, maximal (lexeme, template) pair that can be combined to construct l, as described in Section 6.1. We apply MAX-FAC to each of the training examples (x_i, z_i), creating a single way of producing the desired meaning z_i from a lexeme containing all of the words in x_i. The lexemes and templates created in this way provide the initial factored lexicon.

Inputs: Training set $\{(x_i, z_i) : i = 1 \ldots n\}$ where each example is a sentence x_i paired with a logical form z_i. Set of entity name lexemes L_e. Number of iterations J. Learning rate parameter α_0 and cooling rate parameter c. Empty lexeme set L. Empty template set T.

Definitions: NEW-LEX(y) returns a set of new lexical items from a parse y as described in Section 7.1. MAX-FAC(l) generates a (lexeme, template) pair from a lexical item l. PART-FAC(y) generates a set of templates from parse y. Both of these are described in Section 7.2. The distributions $p(y|x,z;\theta,(L,T))$ and $p(y,z|x;\theta,(L,T))$ are defined by the log-linear model described in Section 4.3.

Initialization:

- For $i = 1 \ldots n$
  - $(\psi, \pi) = \text{MAX-FAC}(x_i \vdash S : z_i)$
  - $L = L \cup \psi\ ,\ T = T \cup \pi$
- Set $L = L \cup L_e$.
- Initialize θ using coocurrence statistics, as described in Section 8.

Algorithm:

For $t = 1 \ldots J, i = 1 \ldots n$:

Step 1: (Add Lexemes and Templates)

- Let $y^* = \arg\max_y p(y|x_i, z_i; \theta, (L,T))$
- For $l \in \text{NEW-LEX}(y^*)$
  - $(\psi, \pi) = \text{MAX-FAC}(l)$
  - $L = L \cup \psi\ ,\ T = T \cup \pi$
- $\Pi = \text{PART-FAC}(y^*)\ ,\ T = T \cup \Pi$

Step 2: (Update Parameters)

- Let $\gamma = \frac{\alpha_0}{1 + c \times k}$ where $k = i + t \times n$.
- Let $\Delta = E_{p(y|x_i,z_i;\theta,(L,T))}[\phi(x_i, y, z_i)] - E_{p(y,z|x_i;\theta,(L,T))}[\phi(x_i, y, z)]$
- Set $\theta = \theta + \gamma\Delta$

Output: Lexemes L, templates T, and parameters θ.

Figure 1: The FUBL learning algorithm.

Step 1 The first step of the learning algorithm in Figure 1 adds lexemes and templates to the factored model given by performing manipulations on the highest scoring correct parse y^* of the current training example (x_i, z_i). First the NEW-LEX procedure is run on y^* as described in Section 6.1 to

generate new lexical items. We then use the function MAX-FAC to create the maximal factorings of each of these new lexical items as described in Section 6 and these are added to the factored representation of the lexicon. New templates can also be introduced through partial factorings of internal parse nodes as described in Section 6.2. These templates are generated by using the function PART-FAC to abstract over the wordspan and a subset of the constants contained in the internal parse nodes of y^*. This step allows for templates that introduce new semantic content to model elliptical language, as described in Section 6.2.

Step 2 The second step does a stochastic gradient descent update on the parameters θ used in the parsing model. This update is described in Section 4.3

Discussion The FUBL algorithm makes use of a direct online approach, where lexemes and templates are introduced in place while analyzing specific sentences. In general, this will overgeneralize; not all ways of combining lexemes and templates will produce high quality lexical items. However, the overall approach includes features, presented in Section 8, that can be used to learn which ones are best in practice. The complete algorithm iterates between adding new lexical content and updating the parameters of the parsing model with each procedure guiding the other.

8 Experimental setup

Data Sets We evaluate on two benchmark semantic parsing datasets: GeoQuery, which is made up of natural language queries to a database of geographical information; and Atis, which contains natural language queries to a flight booking system. The Geo880 dataset has 880 (English-sentence, logical-form) pairs split into a training set of 600 pairs and a test set of 280. The Geo250 data is a subset of the Geo880 sentences that have been translated into Japanese, Spanish and Turkish as well as the original English. We follow the standard evaluation procedure for Geo250, using 10-fold cross validation experiments with the same splits of the data as Wong and Mooney (2007). The Atis dataset contains 5410 (sentence, logical-form) pairs split into a 4480 example training set, a 480 example development set and a 450 example test set.

Evaluation Metrics We report exact match *Recall* (percentage of sentences for which the correct logical-form was returned), *Precision* (percentage of returned logical-forms that are correct) and *F1* (harmonic mean of Precision and Recall). For Atis we also report partial match Recall (percentage of correct literals returned), Precision (percentage of returned literals that are correct) and F1, computed as described by Zettlemoyer and Collins (2007).

Features We introduce two types of features to discriminate between parses: *lexical features* and *logical-form features*.

Lexical features fire on the lexemes and templates used to build the lexical items used in a parse. For each (lexeme,template) pair used to create a lexical item we have indicator features ϕ_l for the lexeme used, ϕ_t for the template used, and $\phi_{(l,t)}$ for the pair that was used. We assign the features on lexical templates a weight of 0.1 to prevent them from swamping the far less frequent but equally informative lexeme features.

Logical-form features are computed on the lambda-calculus expression z returned at the root of the parse. Each time a predicate p in z takes an argument a with type $Ty(a)$ in position i, it triggers two binary indicator features: $\phi_{(p,a,i)}$ for the predicate-argument relation; and $\phi_{(p,Ty(a),i)}$ for the predicate argument-type relation. Boolean operator features look at predicates that occurr together in conjunctions and disjunctions. For each variable v_i that fills argument slot i in two conjoined predicates p_1 and p_2 we introduce a binary indicator feature $\phi_{conj(i,p_1,p_2)}$. We introduce similar features $\phi_{disj(i,p_1,p_2)}$ for variables v_i that are shared by predicates in a disjunction.

Initialization The weights for lexeme features are initialized according to coocurrance statistics between words and logical constants. These are estimated with the Giza++ (Och and Ney, 2003) implementation of IBM Model 1. The initial weights for templates are set by adding -0.1 for each slash in the syntactic category and -2 if the template contains logical constants. Features on lexeme-template pairs and all parse features are initialized to zero.

Systems We compare performance to all recently-published, directly-comparable results. For GeoQuery, this includes the ZC05, ZC07 (Zettlemoyer

System	Exact Match		
	Rec.	Pre.	F1
ZC07	74.4	**87.3**	80.4
UBL	65.6	67.1	66.3
FUBL	**81.9**	82.1	**82.0**

Table 1: Performance on the Atis development set.

System	Exact Match			Partial Match		
	Rec.	Pre.	F1.	Rec.	Pre.	F1
ZC07	**84.6**	**85.8**	**85.2**	**96.7**	**95.1**	**95.9**
HY06	-	-	-	-	-	90.3
UBL	71.4	72.1	71.7	78.2	98.2	87.1
FUBL	82.8	82.8	82.8	95.2	93.6	94.6

Table 2: Performance on the Atis test set.

and Collins, 2005, 2007), λ-WASP (Wong and Mooney, 2007), UBL (Kwiatkowski et al., 2010) systems and DCS (Liang et al., 2011). For Atis, we report results from HY06 (He and Young, 2006), ZC07, and UBL.

9 Results

Tables 1-4 present the results on the Atis and Geoquery domains. In all cases, FUBL achieves at or near state-of-the-art recall (overall number of correct parses) when compared to directly comparable systems and it significantly outperforms UBL on Atis.

On Geo880 the only higher recall is achieved by DCS with prototypes - which uses significant English-specific resources, including manually specified lexical content, but does not require training sentences annotated with logical-forms. On Geo250, FUBL achieves the highest recall across languages. Each individual result should be interpreted with care, as a single percentage point corresponds to 2-3 sentences, but the overall trend is encouraging.

On the Atis development set, FUBL outperforms ZC07 by 7.5% of recall but on the Atis test set FUBL lags ZC07 by 2%. The reasons for this discrepancy are not clear, however, it is possible that the syntactic constructions found in the Atis test set do not exhibit the same degree of variation as those seen in the development set. This would negate the need for the very general lexicon learnt by FUBL.

Across the evaluations, despite achieving high recall, FUBL achieves significantly lower precision than ZC07 and λ-WASP. This illustrates the trade-off from having a very general model of proposing lexical structure. With the ability to skip unseen

System	Rec.	Pre.	F1
Labelled Logical Forms			
ZC05	79.3	**96.3**	87.0
ZC07	86.1	91.6	**88.8**
UBL	87.9	88.5	88.2
FUBL	**88.6**	88.6	88.6
Labelled Question Answers			
DCS	**91.1**	-	-

Table 3: Exact match accuracy on the Geo880 test set.

System	English			Spanish		
	Rec.	Pre.	F1	Rec.	Pre.	F1
λ-WASP	75.6	**91.8**	82.9	80.0	**92.5**	**85.8**
UBL	81.8	83.5	82.6	81.4	83.4	82.4
FUBL	**83.7**	83.7	**83.7**	**85.6**	85.8	85.7
System	Japanese			Turkish		
	Rec.	Pre.	F1	Rec.	Pre.	F1
λ-WASP	81.2	**90.1**	**85.8**	68.8	**90.4**	**78.1**
UBL	83.0	83.2	83.1	71.8	77.8	74.6
FUBL	**83.2**	83.8	83.5	**72.5**	73.7	73.1

Table 4: Exact-match accuracy on the Geo250 data set.

words, FUBL returns a parse for all of the Atis test sentences, since the factored lexicons we are learning can produce a very large number of lexical items. These parses are, however, not always correct.

10 Analysis

The Atis results in Tables 1 and 2 highlight the advantages of factored lexicons. FUBL outperforms the UBL baseline by 16 and 11 points respectively in exact-match recall. Without making any modification to the CCG grammars or parsing combinators, we are able to induce a lexicon that is general enough model the natural occurring variations in the data, for example due to sloppy, unedited sentences.

Figure 2 shows a parse returned by FUBL for a sentence on which UBL failed. While the word "cheapest" is seen 208 times in the training data, in only a handful of these instances is it seen in the middle of an utterance. For this reason, UBL never proposes the lexical item, cheapest $\vdash NP\backslash(S|NP)/(S|NP)$: $\lambda f \lambda g.argmin(\lambda x.f(x) \wedge g(x), \lambda y.cost(y))$, which is used to parse the sentence in Figure 2. In contrast, FUBL uses a lexeme learned from the same word in different contexts, along with a template learnt from similar words in a similar context, to learn to per-

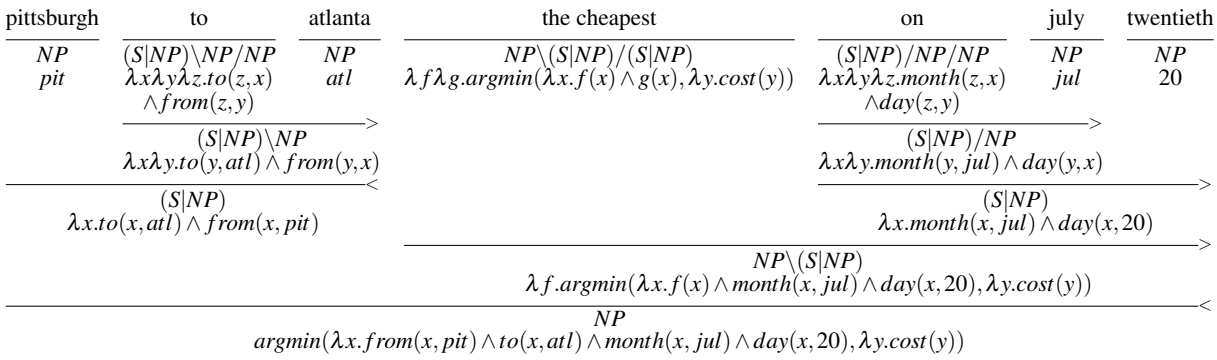


Figure 2: An example learned parse. FUBL can learn this type of analysis with novel combinations of lexemes and templates at test time, even if the individual words, like "cheapest," were never seen in similar syntactic constructions during training, as described in Section 10.

form the desired analysis.

As well as providing a new way to search the lexicon during training, the factored lexicon provides a way of proposing new, unseen, lexical items at test time. We find that new, non-*NP*, lexical items are used in 6% of the development set parses.

Interestingly, the addition of templates that introduce semantic content (as described in Section 6.2) account for only 1.2% of recall on the Atis development set. This is suprising as elliptical constructions are found in a much larger proportion of the sentences than this. In practice, FUBL learns to model many elliptical constructions with lexemes and templates introduced through maximal factorings. For example, the lexeme $(\text{to}, [from, to])$ can be used with the correct lexical template to deal with our motivating example "flights Boston to New York". Templates that introduce content are therefore only used in truly novel elliptical constructions for which an alternative analysis could not be learned.

Table 5 shows a selection of lexemes and templates learned for Atis. Examples 2 and 3 show that morphological variants of the same word must still be stored in separate lexemes. However, as these lexemes now share templates, the total number of lexical variants that must be learned is reduced.

Most common lexemes by type of constants in $\vec{c}$.			
1	e	(Boston, $[bos]$)	(Denver, $[den]$)
2	$\langle e,t \rangle$	(flight, $[flight]$)	(flights, $[flight]$)
3	$\langle e,i \rangle$	(fare, $[cost]$)	(fares, $[cost]$)
4	$\langle e,\langle e,t \rangle \rangle$	(from, $[from]$)	(to, $[to]$)
5	$\langle e,i \rangle, \langle e,t \rangle$	(cheapest, $[argmin, cost]$)	(earliest, $[argmin, dep\_time]$)
6	$\langle i,\langle i,t \rangle \rangle, \langle e,i \rangle$	(after, $[>, dep\_time]$)	(before, $[<, dep\_time]$)
Most common templates matching lexemes above.			
1	$\lambda(\omega,\vec{v}).\omega \vdash NP : v_1$		
2	$\lambda(\omega,\vec{v}).\omega \vdash S\|NP : \lambda x.v_1(x)$		
3	$\lambda(\omega,\vec{v}).\omega \vdash NP\|NP : \lambda x.v_1(x)$		
4	$\lambda(\omega,\vec{v}).\omega \vdash S\|NP/NP\backslash(S\|NP) : \lambda x\lambda y.v_1(x,y)$		
5	$\lambda(\omega,\vec{v}).\omega \vdash NP/(S\|NP) : \lambda f.v_1(\lambda x.f(x), \lambda y, v_2(y))$		
6	$\lambda(\omega,\vec{v}).\omega \vdash S\|NP\backslash(S\|NP)/NP : \lambda x\lambda y\lambda z.v_1(v_2(z),x) \wedge y(x)$		

Table 5: Example lexemes and templates learned from the Atis development set.

11 Discussion

We argued that factored CCG lexicons, which include both lexemes and lexical templates, provide a compact representation of lexical knowledge that can have advantages for learning. We also described a complete approach for inducing factored, probabilistic CCGs for semantic parsing, and demonstrated strong performance across a wider range of benchmark datasets that any previous approach.

In the future, it will also be important to explore morphological models, to better model variation within the existing lexemes. The factored lexical representation also has significant potential for lexical transfer learning, where we would need to learn new lexemes for each target application, but much of the information in the templates could, potentially, be ported across domains.

Acknowledgements

The work was supported in part by EU ERC Advanced Fellowship 249520 GRAMPLUS, and an ESPRC PhD studentship. We would like to thank Yoav Artzi for helpful discussions.

References

Bos, Johan, Stephen Clark, Mark Steedman, James R. Curran, and Julia Hockenmaier. 2004. Wide-coverage semantic representations from a CCG parser. In *Proceedings of the International Conference on Computational Linguistics*.

Branavan, S.R.K., Luke Zettlemoyer, and Regina Barzilay. 2010. Reading between the lines: Learning to map high-level instructions to commands. In *Association for Computational Linguistics (ACL)*.

Carpenter, Bob. 1997. *Type-Logical Semantics*. The MIT Press.

Clark, Stephen and James R. Curran. 2003. Log-linear models for wide-coverage CCG parsing. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

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

Clarke, James, Dan Goldwasser, Ming-Wei Chang, and Dan Roth. 2010. Driving semantic parsing from the world's response. In *Proceedings of the Fourteenth Conference on Computational Natural Language Learning (CoNLL-2010)*. Uppsala, Sweden, pages 18–27.

Ge, Ruifang and Raymond J. Mooney. 2006. Discriminative reranking for semantic parsing. In *Proceedings of the COLING/ACL 2006 Main Conference Poster Sessions*.

Goldwasser, Dan, Roi Reichart, James Clarke, and Dan Roth. 2011. Confidence driven unsupervised semantic parsing. In *Association for Computational Linguistics (ACL)*.

He, Yulan and Steve Young. 2005. Semantic processing using the hidden vector state model. *Computer Speech and Language* .

He, Yulan and Steve Young. 2006. Spoken language understanding using the hidden vector state model. *Speech Communication* 48(3-4).

Hockenmaier, Julia and Mark Steedman. 2002. Generative models for statistical parsing with Combinatory Categorial Grammar. In *Proceedings of the 40th Meeting of the ACL*. Philadelphia, PA, pages 335–342.

Kate, Rohit J. and Raymond J. Mooney. 2006. Using string-kernels for learning semantic parsers. In *Proceedings of the 44th Annual Meeting of the Association for Computational Linguistics*.

Kate, Rohit J., Yuk Wah Wong, and Raymond J. Mooney. 2005. Learning to transform natural to formal languages. In *Proceedings of the National Conference on Artificial Intelligence*.

Kwiatkowski, Tom, Luke Zettlemoyer, Sharon Goldwater, and Mark Steedman. 2010. Inducing probabilistic CCG grammars from logical form with higher-order unification. In *Proceedings of the Conference on Emperical Methods in Natural Language Processing*.

LeCun, Y., L. Bottou, Y. Bengio, and P. Haffner. 1998. Gradient-based learning applied to document recognition. *Proceedings of the IEEE* 86(11):2278–2324.

Liang, P., M. I. Jordan, and D. Klein. 2009. Learning semantic correspondences with less supervision. In *Association for Computational Linguistics and International Joint Conference on Natural Language Processing (ACL-IJCNLP)*.

Liang, P., M. I. Jordan, and D. Klein. 2011. Learning dependency-based compositional semantics. In *Association for Computational Linguistics (ACL)*.

Lu, Wei, Hwee Tou Ng, Wee Sun Lee, and Luke S. Zettlemoyer. 2008. A generative model for parsing natural language to meaning representations. In *Proceedings of The Conference on Empirical Methods in Natural Language Processing*.

Miller, Scott, David Stallard, Robert J. Bobrow, and Richard L. Schwartz. 1996. A fully statistical approach to natural language interfaces. In *Proc. of the Association for Computational Linguistics*.

Nguyen, Le-Minh, Akira Shimazu, and Xuan-Hieu Phan. 2006. Semantic parsing with structured SVM ensemble classification models. In *Proceedings of the COLING/ACL 2006 Main Conference Poster Sessions*.

Och, Franz Josef and Hermann Ney. 2003. A systematic comparison of various statistical alignment models. *Computational Linguistics* 29(1):19–51.

Papineni, K. A., S. Roukos, and T. R. Ward. 1997. Feature-based language understanding. In *Proceedings of European Conference on Speech Communication and Technology*.

Poon, Hoifung and Pedro Domingos. 2009. Unsupervised semantic parsing. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Poon, Hoifung and Pedro Domingos. 2010. Unsupervised ontology induction from text. In *Association for Computational Linguistics (ACL)*.

Ramaswamy, Ganesh N. and Jan Kleindienst. 2000. Hierarchical feature-based translation for scalable natural language understanding. In *Proceedings of International Conference on Spoken Language Processing*.

Steedman, Mark. 1996. *Surface Structure and Interpretation*. The MIT Press.

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

Tang, Lappoon R. and Raymond J. Mooney. 2000. Automated construction of database interfaces: Integrating statistical and relational learning for semantic parsing. In *Proceedings of the Joint Conference on Empirical Methods in Natural Language Processing and Very Large Corpora*.

Thompson, Cynthia A. and Raymond J. Mooney. 2002. Acquiring word-meaning mappings for natural language interfaces. *Journal of Artificial Intelligence Research* 18.

Vogel, Adam and Dan Jurafsky. 2010. Learning to follow navigational directions. In *Association for Computational Linguistics (ACL)*.

Watkinson, Stephen and Suresh Manandhar. 1999. Unsupervised lexical learning with categorial grammars using the LLL corpus. In *Proceedings of the 1st Workshop on Learning Language in Logic*.

Wong, Yuk Wah and Raymond Mooney. 2006. Learning for semantic parsing with statistical machine translation. In *Proceedings of the Human Language Technology Conference of the NAACL*.

Wong, Yuk Wah and Raymond Mooney. 2007. Learning synchronous grammars for semantic parsing with lambda calculus. In *Proceedings of the Association for Computational Linguistics*.

Zelle, John M. and Raymond J. Mooney. 1996. Learning to parse database queries using inductive logic programming. In *Proceedings of the National Conference on Artificial Intelligence*.

Zettlemoyer, Luke S. and Michael Collins. 2005. Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence*.

Zettlemoyer, Luke S. and Michael Collins. 2007. Online learning of relaxed CCG grammars for parsing to logical form. In *Proc. of the Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*.

Zettlemoyer, Luke S. and Michael Collins. 2009. Learning context-dependent mappings from sentences to logical form. In *Proceedings of The Joint Conference of the Association for Computational Linguistics and International Joint Conference on Natural Language Processing*.

COMPUTATIONAL LINGUISTIC APPROACHES TO TEMPORALITY

(Draft 2.2, 4 April 2011

Mark Steedman

1 INTRODUCTION

Temporality in computational linguistics and natural language processing can be considered from two aspects. One concerns the use of linguistic and philosophical theories of temporality in computational applications. The other concerns use of computational theory in its own right to define new kinds of theories of dynamical systems including natural language and its temporal semantics. The latter influence is at least as important as the former.

2 LINGUISTIC CONTRIBUTIONS TO COMPUTATIONAL LINGUISTICS

As in the case of nominal expressions in natural language, we should be careful to distinguish temporal *semantics*, or the question of what kinds of objects and relations temporal categories denote, from the question of temporal *reference* to particular times or events that the discourse context affords.

It is useful to draw a further distinction within the semantics between temporal *ontology*, or the types of temporal entity that the theory entertains, such as instants, intervals, events, states, or whatever, temporal *quantification* over such entities, and the temporal *relations* over them that it countenances, such as priority or posteriority, causal dependence, and the like.

2.1 Temporal Semantics: Ontologies, Quantifiers, and Relations

Applications such as information retrieval (IR), and question-answering (QA), have made surprisingly little use of the riches offered by linguistic temporal semantics and temporal logic. The reason is the inextricable entanglement of the temporal categories with everyday knowledge. As other chapters of this handbook show, categories like tense, mood, and aspect are confounded with non-temporal relations such as causality, teleology, counterfactuality, evidentiality, and the like, to an extent that makes firm boundaries to temporal semantics hard to draw.

For example, the question

(1) Have you met Miss Jones?

does not define a temporal relation between the present time and an event of meeting Miss Jones, as theories of temporal reference founded on Reichenbach (1947) might seem to suggest, but rather asks whether the state of affairs that is *consequent upon* such an event—roughly speaking, *knowing Miss Jones*—is in force (Moens and Steedman, 1988). One may be able to answer such questions in the affirmative even if one has no recollection of the event in question, nor any idea when it might have been, or even lacks the capacity for such recollection, as in the case of certain agnosias.

Of course, one may infer that meeting Miss Jones must have preceded the present, for this state of affairs to hold—but that is an entailment of the relation between cause and effect, rather than temporal sequence as such.

Similarly, if a search engine offers (2A) in answer to a query (2Q), in order to answer the question correctly, a question answerer must understand the textual entailment of A that, although one might have expected Swatman to win, in the event he did not:

(2) Q: Did John Swatman win the British Open Gold Medal?

A: In 1980 at 16 years of age he fought his way to the final in the under 60 Kg category, and *was winning the contest when he was forced to withdraw through injury*.

That is, the progressive denotes a state of affairs that would normally bring about a win, rather than a temporal relation to an actual event of winning.

Such inferences are extremely specific to the particular content that is involved. Thus, the temporal extent of the state of *having met Miss Jones* is generally (as the song says) only bounded by the lifetime of the participants. However, if the following question is asked, the relevant consequent state is bounded by our knowledge of the digestive process to a few hours:

(3) Has the patient had anything to eat?

Similarly, if the question is:

(4) Has the patient had a tetanus shot recently?

—then the answer depends on specific knowledge of the length of time the

consequent protection typically lasts, which is a few years.

Human beings are remarkably good at such associative inference, which they seem to achieve quite effortlessly. However, the problem of formulating such knowledge in computational terms, and carrying out similar inference by machine, using the standard logics that have been designed for the purpose (Prior, 1967; McDermott, 1982; Allen, 1983), is very hard, although Prior's work provided one of the foundations for the computational dynamic logics discussed in section 3. Pratt and Francez (2001) formulate temporal generalized quantifiers for such a framework, and Pratt-Hartmann (2005) proves complexity properties of this system.

There have been attempts to design limited logics with better search properties, which have tended to trade under the name of "temporal database query languages", and there have been attempts to design natural language user-interfaces or "front ends" for such systems, drawing on linguistically-informed semantics, notably the ontologies of Vendler (1967) and followers (e.g. Bruce, 1972; Ritchie, 1979; Moens, 1987; Hinrichs, 1988; Palmer et al., 1993; Crouch and Pulman, 1993; White, 1994; Dorr and Olsen, 1997; Androutsopoulos et al., 1998; Dorr, 2007).

Such ontologies typically distinguish event-types according to a number of dimensions including $\pm$durativity and $\pm$telicity, and distinguish states as of type progressive, consequent, iterative, habitual, and so on. Several of these systems constitute recursive mereologies, or part-whole hierarchies, sometimes embodying a notion of type-*coercion* or *overloading*, whereby aspects and adverbial modifiers compositionally add layers of temporal predication such as preparation, initiation, iteration, and culmination, and the like, without any limit on depth of embedding, as in:

(5) It took me two years to be able to play "Young and Foolish" in less than thirty seconds for up to an hour at a time.

Further examples of work of this linguistically informed kind are Pustejovsky 1991a; Kameyama et al. 1993; Hitzeman 1997; Narayanan 1997. However, such attempts at general-purpose solutions to the problem of temporal query have not been widely used, and typically involve serious investments of time in knowledge-engineering by hand.

Instead, early natural language query systems for temporally rich domains, such as Sager et al. 1994 tended to hard-wire the requisite knowledge into

collections of very specific hand-built inference rules representing actions directly, in ad-hoc (although nonetheless revealing) ways, in the tradition of AI action representations systems, to which we will return in section 3.

The CYC project (Lenat, 1995) is an attempt in the same tradition to build a knowledge representation of this kind by hand on a very large scale, and with general applicability including temporal reasoning, based on a rather general ontology and a number in the millions of specific rules of inference, many of which represent relations among events and states. Specialized versions of CYC have been developed for supporting effective inference in several narrower "closed" domains, many of them industrially significant, and a research version is freely available.

However, common sense reasoning about the everyday world, of the kind that is needed to capture the simple natural language examples with which this piece began, remains a very difficult task. Attempts to apply the publicly available version in tasks like text-based inference have not proved very successful (Mahesh et al., 1996), The general suspicion is that this is because such hand-built resources are both too high-level in terms of ontology, and too small-scale in comparison to what a mixture of animal evolution and social learning has put in our own heads.

In reaction to this realization concerning the limits of hand-built knowledge resources of many kinds, including ontologies such as WordNet (Fellbaum, 1998b), FrameNet (Baker et al., 1998), and VerbNet (Kipper et al., 2008), there is renewed interest recently in building larger resources of this kind by clustering on collocations in large volumes of text (Lin and Pantel, 2001), or by searching such text corpora for string-based proxies for ontological relations (Webber et al., 2002). The proposals of Banko and Etzioni (2007); Etzioni et al. (2007) and Mitchell et al. (2009) for "machine reading" to create similar knowledge resources are related.

A more radical proposal of this kind is to use more directly associative knowledge representations such as associative memory models and semantic networks whose nodes directly correspond to individuals of various types, and whose arcs represent relations between them. Harrington and Clark (2009) have proposed a method using a wide coverage parser to parse unlabeled text and constructing a semantic network an orders of magnitude larger than CYC, using spreading network activation to bound the complexity of network access

and update.

The biggest obstacle to such ambitious plans is the low reliability of wide-coverage parsers. Because of the large search spaces involved, the best-performing parsers use parsing models (and often grammars) acquired by "supervised" machine learning from human-annotated corpora such as the one million word Penn *Wall Street Journal* Treebank. The models are acquired by deriving probabilities (or feature weights) from the frequency with which components of derivations are found in the corpus, in order to choose the derivation that most closely resembles the training material.

The most successful parsers frequently exploit "head dependency" models, in which probabilities or weights are computed on the basis of frequency of co-occurrence of relations between particular words, such as the noun "days" acting as the subject of verb "elapsed". Such parsers, while performing better than the hand-built alternative, still have dependency recovery rates of only around 90% (Collins, 2003). Since word dependencies are closely related to semantic predicate-argument relations, there is a danger that the structures the parsers deliver will be too errorful for this process to deliver useful semantic networks.

Error analysis suggests that the reason the parsers are so weak is that 1M words of fairly arbitrarily selected annotated newspaper text is not enough to give us a grammar or a parsing model comparable to what we have in our own heads. There is considerable work going on to develop unsupervised methods for parser induction from unannotated text, and semisupervised methods for using unlabeled text to generalize the treebank parsers.

The relative success of supervised-learned parsers using head dependency models trained on human-labeled data might seem to suggest a quite different and much bolder solution to the common-sense reasoning bottleneck in temporal semantics. The content-dependency of the extent of the consequent state denoted by the perfect on the nature of the core event—*meeting Miss Jones* versus *eating something* versus *having a tetanus shot*—is reminiscent of the way in which those parsing models rank parses by assigning higher probability to a head-word dependency that occurs frequently in the training data than one that appears rarely or not at all. Head-word dependency parsing models work because they approximate a mixture of semantics and real world knowledge that underlies frequent collocations.

One can therefore consider as a thought experiment the idea of approximating a similar mixture underlying the interpretation of tenses, moods, and aspects, by having human annotators annotate texts about events like *meeting Miss Jones* with the implicit consequent states like *knowing Miss Jones* and preparatory processes like *traveling for the purpose of fulfilling an appointment with Miss Jones*, together with their temporal extents, and learning a model that would allow a machine to answer questions like *Was Mr. Smith meeting Miss Jones when he had the accident?* and *had he met Miss Jones?*.

Of course, such an experiment is completely unrealistic, both in terms of the possibility of obtaining reliable annotations, and in terms of the amount of annotated data that would be required for effective machine learning, let alone in terms of the limitations of the learning techniques themselves when faced with an essentially AI-complete problem. However, a scaled-down version of this idea is being attempted in the related area of temporal reference, to which we now turn.

2.2 Temporal Reference

By temporal reference is meant the anchoring of temporal descriptions to specific clock-times, or to other events in an established narrative. The simple tenses—the past, present, and in English the bare infinitival future—are temporally referential, in the sense that their underlying Reichenbachian reference time R must stand in an inferable temporal/causal relation to some time or event that is either already give discourse-given, as in (6a), or provided by a modifier that is itself temporal/causally anchored, as in (6b,c);

(6) a. It was the night they raided Minsky's. I met Miss Jones.
    b. I met Miss Jones the night they raided Minsky's.
    c. I met Miss Jones when they raided Minsky's.
    d. I met Miss Jones soon after they raided Minsky's.

Webber (1988) points out that such temporal anchoring processes resemble definite noun-phrase reference in allowing "bridging" reference to inferred referents, as when the mention of a car supports reference to "the driver" (Clark and Marshall, 1981). Such inferences are knowledge-dependent in the same way as the temporal semantic interpretations considered in the last section. Thus, in the following example, the fact that we know that throwing us in jail

followed the raid, and that coming with a warrant *preceded* it, is a matter of world-knowledge about preparations for, and consequences of, such events:

(7) The night they raided Minsky's,
    a. they threw us all in jail.
    b. they came with a warrant

Such discourse is also characterized by shifting temporal focus: once we have decided the relation of throwing in jail to the raid, it may in turn act as anchor for further events such as phone calls, which may act as anchors for events of obtaining bail, and so on, under similar conditions of script-like general knowledge about goal-directed activities (Schank, 1975).

As in the case of early work on temporal semantics, for suitably closed domains we may be able to finesse explicit reference. Wiebe et al. (1998) describe a domain-specific rule-based approach to temporal reference resolution in the sense of time-stamping for a corpus of scheduling dialogs consisting of exchanges like the following:

(8) a. Would you like to meet Wednesday, August 2nd?
    b. No, how about Friday at 2?

This work uses a graph-structured stack as a focus model that allows non-adjacent antecedent anchors in complex dialogs. Filatova and Hovy (2001) offer a related approach to time-stamping event clauses in the more open domain of newspaper stories, including relations of anteriority and priority. Both papers evaluate on held-out data—that is, additional human-labeled data that have not been used for training.

Developed as part of the ACE (automatic content extraction) initiative hosted at the Linguistic Data Consortium at the University of Pennsylvania, the TIMEX2 annotation scheme (Ferro et al., 2005) has been used to annotate corpora such as the ACE 2005 corpus (around 600 documents), which has been used for training and evaluating automatic temporal expression recognition and normalization (TERN) programs using a mixture of small numbers of hand-written rules and machine learning (e.g. Ahn et al. 2007)

The TimeML temporal mark-up language (Pustejovsky et al., 2003a; Verhagen et al., 2009) is a reformulation of TIMEX2 that has been extended to cover events, temporal relations, and certain kinds of state, and used for annotation of the Timebank corpus of 186 news reports (Pustejovsky et al., 2003b). Pan et al.

(2006) have extended the Timebank annotation to include estimated upper and lower bounds on the temporal extents of TimeML temporal expressions, with reasonable inter-annotator agreement. Chambers et al. (2007) present a temporal relation classifier for six relations trained on the original Timebank corpus, reporting 72% accuracy when these relations are collapsed to the simplest binary classification *before/after*.

Mazur and Dale (2010) criticize both ACE and Timebank for the brevity of the documents that they include, and the limitations on the complexity of the temporal reference that they support. They point out that most temporal expressions in these corpora can be interpreted relative to a single temporal focus or "anchor", defined as the dateline of the report, rather than involving the kinds of shifting focus characteristic of extended discourse and narrative. They offer an alternative WikiWars corpus comprising 22 much more extended Wikipedia articles on the major wars of human history, containing around 2700 TIMEX2annotated temporal expressions.

It is possible in principle that the typical extent of events could be learned from such data, and used to improve TERN-style temporal reference programs of the kind discussed earlier. While TimeML does not mark consequent states (of the kind crucial to the interpretation of (1), *Have you met Miss Jones?*) as such, it does mark "signal words" such as modals and auxiliary verbs, so it is even possible in principle that the typical temporal extent that should be considered in answering questions like (3) (*Has the patient eaten anything?*) and (4) (*Has the patient had a tetanus shot recently?*) could be learned.

Not surprisingly, nothing as ambitious as this has been attempted so far. As Lapata and Lascarides (2006) point out, these corpora are quite small in comparison with the Penn *Wall Street Journal* treebank (the TIMEX2-annotated English portion of ACE 2005 is around 26K words, while TimeBank is around 69K.) Given the sparse nature of these data, and the sheer difficulty in many cases of annotating temporal relations reliably, it is unclear whether supervised learning with human-labeled data can succeed practically on this problem, although, as Lapata and Lascarides point out, the labeled corpora remain valuable as gold-standards against which unsupervised methods can be evaluated.

In reaction to these resource limitations for supervised learning, there has been considerable research into unsupervised methods for training such classifiers using unsupervised methods based on wide-coverage parsing of unlabeled

text. Chklovski and Pantel (2004) learn verb subcategorization frames and semantic relations between them including temporal relations, the latter chosen on the model of Fellbaum 1998a. Lapata and Lascarides (2006) have used such methods successfully to automatically extract a restricted class of specifically temporal relations, by parsing unlabeled text using a wide-coverage statistical parser trained on the Penn treebank, in search of main and subordinate clauses linked by temporal connectives such as "after", "while", and "until", evaluating against the human-labeled Timebank corpus (see above). Chambers and Jurafsky (2009) show how script-like narrative chains involving shared participants can be mined using similar unsupervised methods, evaluating in comparison to the related but non-narrative relations in the hand-built FrameNet corpus (Baker et al., 1998), as well as by a novel "narrative Cloze" procedure.

Automatically identifying temporal semantics and temporal reference remains an extremely hard problem, to which linguistic semantics provides only part of a solution which we do not seem very close to attaining. The Recognizing Text Entailment (RTE) task (Dagan et al., 2006) attempts to provide a standard test-set of pairs of text passages of the kind delivered by real information retrieval and machine translation programs, and questions or "hypotheses" which the text may or may not answer in the positive or negative. Many of the examples involve temporal reference, such as the following:

(9) T: Bush returned to the White House late Saturday while his running mate was off campaigning in the West.
H: Bush left the White House. (RTE example no. 1960:PP)

(10) T: De La Cruz's family said he had gone to Saudi Arabia a year ago to work as a driver after a long period of unemployment.
H: De la Cruz was unemployed. (RTE example no. 1030:RC)

The question of whether **T** entails **H** is in both cases dependent upon the temporal referent of the latter. If in (9), it is taken as Saturday relative to the dateline of **T** then the latter entails that **H** is false. If it is taken as sometime prior to that Saturday, then the entailment is true. Similarly, in (10), the text **T** says that at the time the family spoke, the time De la Cruz went to employment in SA was *after* being unemployed. If the reference time of the hypothesis **H** is the time the family spoke, then either the entailment is false, or there is no entailment (because we are not actually told how long the employment

lasted). Thus it seems that the RTE task examples considerably underspecify the task of temporal reference (Beigman-Klebanov and Beigman 2010). Linguistic semantics will certainly continue to be crucial to solving these hard computational problems, but it is not in itself a sufficient solution.

3 Computational Contributions to Linguistic Temporal Semantics

The temporal semantics of both human languages and programming languages can be thought of as logical languages predicating relations over a model (in the logicians' narrow sense of the term) that can be visualized as in figure 1.

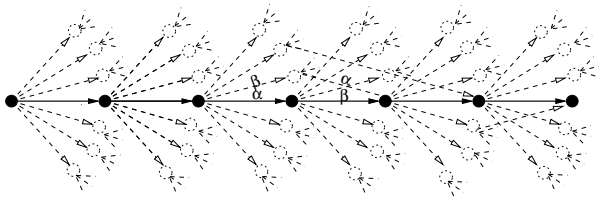


Figure 1: The S4 model

The figure depicts a Kripke or S4 model, in which nodes represent possible states of the world (only a few of which are depicted, and which should be thought of as complex structures, consisting of a number of propositional "fluents", or facts subject to change), and directed arcs represent events α, β, etc. that transform one state into another (of which few are depicted also). We may want to distinguish some particular sequence of states and events as actual or historical: those might be the ones in solid black.

This structure is not "there," in the mind or the computer. It is not something that can ever be built—for one thing, it is infinitely extending. Rather, it describes the space of possibilities that we or a machine inhabit, and to a very limited extent can think about by searching it to some limited depth.

What we and other animals *do* have in our heads (as do machines, if we program them with that capability, or allow them to acquire it for themselves) is a finite but extendable set of rules that describe the events that change one state to another, some of which are probabilistically under our control.

These rules (together with some computational resources that must include

a (possibly simulated) push-down automaton) are what allows us and some other animals to see small portions of the eternal world, and to construct *plans* or sequences of actions that (with any luck) will take us to the more desirable possible worlds (or at least to the ones that we can find by searching the forest of crossing destinies to some very limited depth).

Most of the semantic theories discussed in the present volume assume such a model, implicitly or explicitly, and can be seen as addressing the question of the precise content of the states, and the nature of the events that take us from one state to another.

For example, the theories differ as to whether they take intervals as the basic temporal primitive, and regard events as durative, or whether they take instants as primitive and intervals as composite. Under the first view, a Vendlerian Activity like *running* would be represented as a transition, with a temporal and spatial extent. Under the second view, an Activity would be regarded as a progressive fluent, or property of a state, with the states that it characterizes being accessed via instantaneous incipitative events of *beginning running* and abandoned via terminative events of *stopping running*. (Vendler and his followers seem equivocal between these two interpretations.) Under the latter interpretation, the instantaneous incipitative and terminative events themselves correspond to Vendlerian Achievements, associated with further changes in fluents corresponding to consequent states, such as *running* and *having stopped running*. Vendlerian Accomplishments like *running to the bus-stop* are then the composition of an Activity of *running with the goal of being at the bus-stop*, the terminative Achievement of *stopping running* and the culminative achievement of *reaching the bus-stop*, which in turn initiates its own consequent state of *being at the bus-stop*.

When scaled to practical problems of planning in realistic worlds, such models are clearly going to be very complex. In deciding which of the many theories they allow we should adopt, we will be guided not only by the usual questions of soundness in representing temporal knowledge, but also by questions of efficiency for the purpose of searching for plans, which we can think of as *proofs* in a logic of change.

It is in this connection that theoretical computer science can be of help to linguistics. Computation can also be modeled as a space of states and operators that change state, using logics of change, as Prior (1967) noted himself.

In computer science, issues of constructivity (that proofs of attainability of a state are always accompanied by an algorithm for actually getting there) and efficiency (that useful proofs can actually be found with affordable resources) are always paramount. As a result, theoretical computer science has been the main engine driving progress in the use of temporal logics since the time of Prior.

With these ends in mind, computer scientists have made a very important observation about logics of change as they apply to programs and human reasoning. That is that the kind of changes that we are interested in are *localized*, affecting only a very few among the vast number of facts or fluents that define the current state in the model. Thus, *assigning a value to a register* affects that register and no other aspect of the state of the machine. Similarly, *drinking an ice-cold beer* affects the beer, and the consumer, in predictable ways, but leaves unaffected a miriad other facts that hold in the situation of the action, such as the weather, the color of the walls, and the current popularity ratings of the president of the United States. This suggests that events are to be defined in terms of *partial* descriptions of situations. (There are some events, like detonating H-bombs, that change practically everything. However, these too are only useful to the extent that they can be defined in terms of simple partial descriptions—for example, using a universal quantifier.)

This insight has been captured in a number of ways, both informal and formal. In linguistic theory, an early version of the idea surfaced in Lewis's (1973) idea of "inertia worlds," which he defined in terms of similarity between actual and hypothetical worlds, in order to provide a semantics for counterfactuals, which Dowty (1991) used explain the imperfective paradox. (Fine, 1975, criticized this definition using examples involving H-Bombs and the like, for which similarity between the inertia world and the actual world doesn't seem to work. Lascarides (1991) showed that inertia worlds cannot sensibly be defined other than in terms of the progressive itself.)

In Artificial Intelligence, the idea is usually identified with the STRIPS representation for actions for the purpose of automatically constructing plans (Fikes and Nilsson, 1971), although the idea seems to have been arisen more than once (cf. PLANNER, Hewitt, 1969). STRIPS actions are represented in terms of three elements: a list of *preconditions*, that is, facts or "fluents" which must hold for the action to be possible; a list of *deletions*, or fluents that

cease to be true when the action occurs, and a list of *additions*, or fluents that become true in the aftermath of the action. Additions and deletions are modeled by database update, so the property that every fact that is not explicitly mentioned in the rule remains as it is, holds by default.

For example, the earlier example of *my drinking an ice-cold beer* might be represented by the following triple, in which the variable x is implicitly existentially quantified:

(11) preconditions: *beer(x), ice-cold(x), here(x), here(me), thirsty(me)*
delete: *beer(x), here(x), ice-cold(x), thirsty(me)*
add: *high(me)*

This rule says that when I drink an ice-cold beer, it ceases to be, while I, although I still exist, stop being thirsty and start being high. Whatever else holds in the current state remains unchanged. If we want to model a counterfactual situation such as *If I had not drunk an ice-cold beer* we can reverse the rule and work out that it would be pretty much the same apart from my state and that of the beer. Since our representation is in terms of actions rather than possible worlds, notions of similarity between worlds don't come into it: if we want to include actions like detonating an H-bomb in our plans we can do so. (Simplifying a bit, the latter action deletes everything, so it is easy to work out that counterfactuals like *If someone had detonated an H-Bomb, you wouldn't be here* are true.)

A number of important lessons were learned from work using STRIPS-like action representations. First, it isn't at all easy to represent even the simplest temporal knowledge domains consistently, especially if you want to be able to extend the domains by freely adding new actions. For example, if you represent the fact that some boxes and a truck are *in Edinburgh* as ground facts in a database, then your action of loading boxes *on trucks* should delete the ground fact that the boxes are *in Edinburgh*, and add a ground fact that they are *on the truck*. If the *move* action for trucks is defined in the obvious way, as deleting the ground fact of the truck being *in Edinburgh* and adding one of it being *in London* (say), this stratagem will ensure consistency in reasoning about where the boxes are when the truck moves. (Alternatively, you could define the move action as deleting the location at the origin of any objects that are *on* the moving object and asserting their location at the destination.) It is easy to make mistakes defining domain knowledge like this.

The other lesson learned from STRIPS is that, if you want to do any kind of temporal reasoning over the representation, or more generally need to represent co-occurrent actions, then you have to represent durative events like trucks moving as composed of an instantaneous incipitative event that introduces a progressive fluent, and a terminative event removing it (cf. Kowalski and Sergot, 1986). This ensures that if the truck is *dry* at the start of its journey, and it starts to rains while the truck is moving, the database will not end up saying inconsistently that the truck is both *wet* and *dry* at its destination. (This observation seems to suggest that instants and not intervals should be taken as the primitive elements in any model theory for systems of this kind—see Allen and Hayes, 1989 for a dissenting opinion.)

Event calculi of this kind underlie some very powerful planning programs, which compete at an annual competition on shared tasks of considerable complexity. The STRIPS idea was enshrined in a standard notation by McDermott et al. (1998), and extended in a series of papers culminating in Fox and Long (2006) to cover a rich ontology of time-stamped event types for purposes of the competition.

STRIPS representations were initially derided by logicians for their nonmonotonicity. However, they provide a very natural expression for change of state, particularly when events are instantaneous and discrete, as they are in digital computers. Hoare Logic (Hoare, 1969) is founded on a very similar idea of "triples" $P\{S\}Q$, where P is a set of preconditions for a program statement S and Q describes the resulting state.

Pratt (1976) and Harel (1984) extend Hoare Logic to Dynamic Logic (DL) for the purpose of proving correctness of programs. DL combines a modal logic with the algebra of regular events or finite-state machines. DL is a multimodal one, in which the $\Box$ and $\Diamond$ modalities are relativized to particular event-types. For example, if a program or command α computes a function F over the integers, then we may write the following:

(12) a. $n \geq 0 \Rightarrow [\alpha](y = F(n))$
b. $n \geq 0 \Rightarrow \langle\alpha\rangle(y = F(n))$

The meaning of (12a) is that in any state in which $n \geq 0$, executing α always results in a state where $y = F(n)$. The meaning of (12b) is that in any state in which $n \geq 0$, executing α sometimes results in a state where $y = F(n)$. Although our knowledge of action is inherently nondeterministic, as far as

reasoning about the world goes, we usually reason as if we could predict outcomes, even if we attach a probability of success, so we will mostly be dealing with the $[\alpha]$ modalities.

The α may be sequences $\alpha;\beta;\ldots$. They may also include loops or iteration, as in the following representation for Piaget's "primary circular reaction" of sucking in infants (1936):

(13) [**while** *hungry* **do** *suck*]

Dynamic Logic is usually applied to pure functional programming languages, without update. If we want to apply it for representing actions and change we may want to extend to make it "resource sensitive", by extending it to include Girard's (1995) linear implication operator, written $-\!\circ$ (pronounced "lolly"), to yield "linear dynamic" versions of event calculi such as STRIPS and its descendants.

For example, we might represent the earlier naive version of the *move* action as follows:

(14) $\mathit{affords}(\mathit{move}(x,\mathit{loc}_2)) \wedge \mathit{at}(x,\mathit{loc}_1) -\!\circ [\mathit{move}(x,\mathit{loc}_2)]\mathit{at}(x,\mathit{loc}_2)$

This formula means that if you can move and are at a location, and you move to another location, you stop being *at* the first place and start being *at* the other place. We adopt a convention that only *ground* fact (that is, the ones actually explicit in the data-base) are deleted or added, so the rule doesn't define whether the new situation supports inferrable facts like $\mathit{affords}(\mathit{move}(x,\mathit{loc}_2)$. If we define the latter in terms of a ground facts, using standard implicature as follows, to say that you can't move to a place if you are at that place, then it will not:

(15) $\neg \mathit{at}(x,\mathit{loc}) \Rightarrow \mathit{affords}(\mathit{move}(x,\mathit{loc}))$

(The predicate *affords* for preconditions is used in homage to Gibson's (1979) notion of affordance of actions by situations, which lies at the heart of effective action representation.)

If we want to avoid ramification problems arising from unexpected events like rain, as in the earlier example, then we need to recast the representation in terms of the instantaneous and stative components described there. For example:

(16) a. $\neg at(x, loc) \Rightarrow \mathit{affords}(\mathit{start}(\mathit{move}(x, loc)))$

b. $\mathit{affords}(\mathit{start}(\mathit{move}(x, loc_2))) \wedge at(x, loc_1)$
$\quad -\!\circ[\mathit{start}(\mathit{move}(x, loc_2))]\mathit{moving}(x, loc_1, loc_2)$

(17) a. $\mathit{moving}(x, loc_1, loc_2) \Rightarrow \mathit{affords}(\mathit{stop}(\mathit{move}(x, loc_2)))$

b. $\mathit{affords}(\mathit{stop}(\mathit{move}(x, loc_2)))) \wedge at(x, loc_3) \wedge \mathit{moving}(x, loc_1, loc_2)$
$\quad -\!\circ[\mathit{stop}(\mathit{move}(x, loc_2))]at(x, loc_3)$

Equipped with such rules, practical planning programs can search possible futures by progressing the database breadth-first to some limited depth (say, by iterative deepening, Korf, 1985), and build and execute plans to reach desirable states by search and composing actions in very complicated domains involving multiple actions and objects.

At some point, it may be thought desirable to timestamp everything in such representations. However, the causal structure implicit in the representation will often define the simplest relations of temporal antecedence and aspectual state without explicit indexing to clock-times. For example a simple history of starting to move from loc_1 to loc_2 followed by stopping doing so at a different place loc_3 will contain the information necessary to answer the question "Was x moving to loc_2 when she stopped at loc_3?" Such calculi therefore appear to offer a transparent and efficient representation for the concepts implicit in most current linguistic theories of temporal semantics for natural language, and have obvious relevance for purposes of linguists and philosophers of language.

Systems related to dynamic and nonmonotonic logics in application to natural language semantics are described by van Benthem (1991); Blackburn et al. (1994); Barwise and Seligman (1997)and Fernando (2011). Dynamic and nonmonotonic formalisms invoking or capturing real-world knowledge and related to the computational calculi outlined in this section have been applied to elegant effect in linguistic theories of temporality by Dowty (1986); Sperber and Wilson (1986); Pustejovsky (1991b); Moltmann (1991); Lascarides (1991); Asher (1992); Kamp and Reyle (1993); ter Meulen (1995); Glasbey (2004); Ramchand (1997); Piñon (1997); Pianesi and Varzi (1999); Stone and Hardt (1999); van Lambalgen and Hamm (2005); Bittner (2007); Truswell (2007), among others.

4 FURTHER READING

The computational literature on temporality and representation of causal action and its applications is overwhelming, and I am painfully aware of having been forced to pass over entirely or, worse, to treat very superficially, a great deal of important and relevant work. The following sources are offered as a means of entry to a more extensive literature.

Vardi 2008 provides an exceptionally readable summary of temporal and dynamic logic from Prior to the present from a computer science perspective. Mani et al. 2005 is an indispensable collection of mainly computational readings including several of the papers discussed above, and much other work that deserves attention. Virtually all of the more recent literature on computational approaches is accessible from the web, and in particular from the similarly indispensable ACL Computational Linguistics Anthology, at `http://www.aclweb.org/anthology-new/`. References to the broader literature on tense and aspect in natural language can be found in the Annotated Bibliography of Contemporary Research in Tense, Grammatical Aspect, Aktionsart, and Related Areas at `http://www.scar.utoronto.ca/~binnick/TENSE/`

ACKNOWLEDGMENTS

I am grateful to Robert Binnick and Alex Lascarides for commenting on the draft. The work was supported by EU ERC Advanced Fellowship 249520 GRAMPLUS and EU FP7 IP Xperience.

REFERENCES

Ahn, D., van Rantwijk, J., and de Rijke, M. (2007). A cascaded machine learning approach to interpreting temporal expressions. In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Proceedings of the Main Conference*, pages 420–427, Rochester, NY. ACL.

Allen, J. (1983). Maintaining knowledge about temporal intervals. *Communications of the Association for Computing Machinery*, 26:832–843.

Allen, J. and Hayes, P. (1989). Moments and points in an interval-based temporal logic. *Computational Intelligence*, 5:225–238.

Androutsopoulos, I., Ritchie, G., and Thanisch, P. (1998). Time, tense and aspect in natural language database interfaces. *Natural Language Engineering*, 4:211–228.

Asher, N. (1992). A default truth conditional semantics for the progressive. *Linguistics and Philosophy*, 15:469–598.

Baker, C. F., Fillmore, C. J., and Lowe, J. B. (1998). The berkeley framenet project. In *Proceedings of the 17th international conference on Computational linguistics*, pages 86–90, Montreal, Quebec, Canada. ACL.

Banko, M. and Etzioni, O. (2007). Strategies for lifelong knowledge extraction from the web. In *Proceedings of the 4th International Conference on Knowledge Capture*, pages 95–102, New York. SIGART, ACM.

Barwise, J. and Seligman, J. (1997). *Information Flow*. Cambridge Tracts in Theoretical Computer Science 44. Cambridge University Press, Cambridge.

Beigman-Klebanov, B. and Beigman, E. (2010). Some empirical evidence for annotation noise in a benchmarked dataset. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 438–446, Los Angeles, CA. ACL.

Bittner, M. (2007). Aspectual universals of temporal anaphora. In Rothstein, S., editor, *Theoretical and Crosslinguistic Approaches to the Semantics of Aspect*, pages 348–385. John Benjamins, Amsterdam.

Blackburn, P., Gardent, C., and de Rijke, M. (1994). Back and forth through time and events. In Gabbay, D. and Ohlbach, H.-J., editors, *Temporal Logic*, pages 225–237. Springer, Berlin.

Bruce, B. (1972). A model for temporal references and its application in a question answering program. *Artificial Intelligence*, 3:1–25.

Chambers, N. and Jurafsky, D. (2009). Unsupervised learning of narrative schemas and their participants. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2*, pages 602–610, Suntec, Singapore. ACL.

Chambers, N., Wang, S., and Jurafsky, D. (2007). Classifying temporal relations between events. In *Proceedings of the 45th Annual Meeting of the ACL: Interactive Poster and Demonstration Sessions*, pages 173–176, Prague. ACL.

Chklovski, T. and Pantel, P. (2004). Verbocean: Mining the web for fine-grained semantic verb relations. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 33–40, Barcelona. ACL.

Clark, H. and Marshall, C. (1981). Definite reference and mutual knowledge. In Joshi, A., Webber, B., and Sag, I., editors, *Elements of Discourse Understanding*, pages 10–63. Cambridge University Press, Cambridge.

Collins, M. (2003). Head-driven statistical models for natural language parsing. *Computational Linguistics*, 29:589–637.

Crouch, R. and Pulman, S. (1993). Time and modality in a natural language interface. *Artificial Intelligence*, 63:265–304.

Dagan, I., Glickman, O., and Magnini, B. (2006). The PASCAL recognising text entailment challenge. In Joaquin Quiñonero Candela, Ido Dagan, B. M. and dAlché Buc, F., editors, *Machine Learning Challenges*, volume 3944 of *Lecture Notes in Computer Science*, pages 177–190. Springer, Berlin.

Dorr, B. (2007). Exploiting aspectual features and connecting words for summarization-inspired temporal relation extraction. *Information Processing and Management*, 43:1681–1704.

Dorr, B. J. and Olsen, M. (1997). Deriving verbal and compositional lexical aspect for NLP applications. In *Proceedings of the 8th Conference of the European Chapter of the Association for Computational Linguistics*, pages 151–158, Madrid. ACL.

Dowty, D. (1986). The effects of aspectual class on the temporal structure of discourse: Semantics or pragmatics? *Linguistics and Philosophy*, 9:37–62.

Dowty, D. (1997/1991). *Word Meaning in Montague Grammar*. Reidel, Dordrecht, 2nd edition.

Etzioni, O., Banko, M., and Cafarella, M. (2007). Machine reading. In *Proceedings of AAAI Spring Symposium on Machine Reading*, Menlo Park, CA. AAAI Press.

Fellbaum, C. (1998a). Semantic network of English verbs. In Fellbaum, C., editor, *WordNet: an Electronic Lexical Database*, pages 69–104. MIT Press, Cambridge, MA.

Fellbaum, C., editor (1998b). *WordNet: An Electronic Lexical Database*. MIT Press, Cambridge, MA.

Fernando, T. (2011). Constructing situations and time. *Journal of Philosophical Logic*, 40:371–396.

Ferro, L., Gerber, L., Mani, I., Sundheim, B., and Wilson, G. (2005). Tides 2005 standard for the annotation of temporal expressions. Technical report, MITRE Corporation, Boston, MA.

Fikes, R. and Nilsson, N. (1971). STRIPS: A new approach to the application of theorem proving to problem solving. *Artificial Intelligence*, 2:189–208.

Filatova, E. and Hovy, E. (2001). Assigning time-stamps to event-clauses. In *Proceedings of the Workshop on Temporal and Spatial Information Processing, held at ACL 2001*, pages 104–111, Toulouse. ACL.

Fine, K. (1975). Critical notice of lewis' "counterfactuals". *Mind*, 84:451–458.

Fox, M. and Long, D. (2006). Modelling mixed discrete-continuous domains for planning. *Journal of Artificial Intelligence Research*, 27:235–297.

Gibson, J. (1979). *The Ecological Approach to Visual Perception*. Houghton-Mifflin Co., Boston, MA.

Girard, J.-Y. (1995). Linear logic: Its syntax and semantics. In Girard, J.-Y., Lafont, Y., and Regnier, L., editors, *Advances in Linear Logic*, London Mathematical Society Lecture Notes 222, pages 1–42. Cambridge University Press, Cambridge.

Glasbey, S. (2004). Event structure, punctuality, and 'when'. *Natural Language Semantics*, 12:191–211.

Harel, D. (1984). Dynamic logic. In Gabbay, D. and Guenthner, F., editors, *Handbook of Philosophical Logic*, volume 2, pages 497–604. Reidel, Dordrecht.

Harrington, B. and Clark, S. (2009). ASKNet: Creating and evaluating large scale integrated semantic networks. *International Journal of Semantic Computing*, 2:343–364.

Hewitt, C. (1969). Planner: a language for proving theorems in robots. In *Proceedings of the 1st International Joint Conference on Artificial Intelligence (IJCAI'69)*, pages 295–301, Washington, DC. AAAI.

Hinrichs, E. (1988). Tense, quantifiers, and contexts. *Computational Linguistics*, 14:3–14.

Hitzeman, J. (1997). Semantic partition and temporal adverbials. *Natural Language Semantics*, 5:87–100.

Hoare, C. A. (1969). An axiomatic basis for computer programming. *Communications of the Association for Computing Machinery*, 30:576–583.

Kameyama, M., Passonneau, R., and Poesio, M. (1993). Temporal centering. In *Proceedings of the 31st Annual Meeting of the Association for Computational Linguistics*, pages 70–77, Columbus, OH. ACL.

Kamp, H. and Reyle, U. (1993). *From Discourse to Logic*. Kluwer, Dordrecht.

Kipper, K., Korhonen, A., Ryant, N., and Palmer, M. (2008). A large-scale classification of English verbs. *Language Resources and Evaluation*, 42:21–40.

Korf, R. (1985). Depth-first iterative deepening: An optimal admissible tree search. *Artificial Intelligence*, 27:97–109.

Kowalski, R. and Sergot, M. (1986). A logic-based calculus of events. *New Generation Computing*, 4:67–95.

Lapata, M. and Lascarides, A. (2006). Learning sentence-internal temporal relations. *Journal of Artificial Intelligence Research*, 27:85–117.

Lascarides, A. (1991). The progressive and the imperfective paradox. *Synthèse*, 87(6):401–447.

Lenat, D. (1995). Cyc: a large-scale investment in knowledge infrastructure. *Communications of the ACM*, 38(11):33–38.

Lewis, D. (1973). *Counterfactuals*. Harvard University Press, Cambridge, MA.

Lin, D. and Pantel, P. (2001). Induction of semantic classes from natural language text. In *Proceedings of ACM SIGKDD Conference on Knowledge Discovery and Data-Mining (KDD-01)*, pages 317–322, San Francisco.

Mahesh, K., Nirenburg, S., Cowie, J., and Farwell, D. (1996). An assessment of Cyc for natural language processing. Technical Report MCCS-96-302, New Mexico State University.

Mani, I., Pustejovsky, J., and Gaizauskas, R., editors (2005). *The Language of Time*. Oxford University Press, Oxford.

Mazur, P. and Dale, R. (2010). WikiWars: A new corpus for research on temporal expressions. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 913–922, Cambridge, MA. ACL.

McDermott, D. (1982). A temporal logic for reasoning about processes and actions. *Cognitive Science*, 6:101–155.

McDermott, D., Ghallab, M., Howe, A., Knoblock, C., Ram, A., Veloso, M., Weld, D., and Wilkins, D. (1998). PDDL the planning domain definition language. DCS TR-1165, Yale Center for Computational Vision and Control, New Haven, CT.

Mitchell, T., Betteridge, J., Carlson, A., Hruschka, E., and Wang, R. (2009). Populating the semantic web by macro-reading Internet text. In *Proceedings of the 8th International Semantic Web Conference (ISWC 2009)*, Karlsuhe. Semantic Web Science Association.

Moens, M. (1987). *Tense, Aspect, and Time Reference*. PhD thesis, Edinburgh.

Moens, M. and Steedman, M. (1988). Temporal ontology and temporal reference. *Computational Linguistics*, 14:15–28.

Moltmann, F. (1991). Measure adverbials. *Linguistics and Philosophy*, 14:629–660.

Narayanan, S. (1997). Talking the talk *is* like walking the walk: A computational model of verbal aspect. In *Proceedings of the Nineteenth Conference of the Cognitive Science Society, Stanford, August*, Mahwah, NJ. Lawrence Erlbaum Associates.

Palmer, M., Passonneau, R., Weir, C., and Finin, T. (1993). The kernel text understanding system. *Artificial Intelligence*, 63:17–68.

Pan, F., Rutu, M., and Hobbs, J. (2006). Extending timeml with typical durations of events. In *Proceedings of the Worshop on Annotating and Reasoning about Time and Events, held at the 44th Annual Meeting of the Association for Computational Linguistics*, pages 38–45, Sydney. ACL.

Piñon, C. (1997). Achievements in an event semantics. In *Proceedings of the 7th Conference on Semantics and Linguistic Theory*, pages 276–293, Ithaca, NY. Cornell University.

Piaget, J. (1936). *La naissance de l'intelligence chez l'enfant*. Delachaux et Niestle, Paris. translated 1953 as *The Origin of Intelligence in the Child*, Routledge and Kegan Paul.

Pianesi, F. and Varzi, A. (1999). The context-dependency of temporal reference in event semantics. In Bouquet, P., Brezillon, P., and Serafini, L., editors, *Modeling and Using Context: Proceedings of the 2nd International and Interdiciplinary Conference*, pages 507–510. Springer, Berlin.

Pratt, I. and Francez, N. (2001). Temporal prepositions and temporal generalized quantifiers. *Linguistics and Philosophy*, 24:187–222.

Pratt, V. (1976). Semantical considerations on floyd-hoare logic. In *Proceedings of the 17th Symposium on Foundations of Computer Science*, pages 109–121.

Pratt-Hartmann, I. (2005). Temporal prepositions and their logic. *Artificial Intelligence*, 166:1–36.

Prior, A. (1967). *Past, Present and Future*. Clarendon Press, Oxford.

Pustejovsky, J. (1991a). The generative lexicon. *Computational Linguistics*, 17:409–441.

Pustejovsky, J. (1991b). The syntax of event structure. *Cognition*, 41:47–81.

Pustejovsky, J., Casteño, J., Ingria, R., Saurí, R., Gaizauiuskas, R., Setzer, A., Katz, G., and Radev, D. (2003a). Timeml: Robust specification of event and temporal expressions in text. In *Proceedings of the 5th International Workshop on Computational Semantics*, Tilburg. AAAI. as TR-SS-03-07.

Pustejovsky, J., Hanks, P., Saurí, R., See, A., Day, D., Ferro, L., Gaizauiuskas, R., Lazo, M., Setzer, A., and Sundheim, B. (2003b). The timebank corpus. In *Proceedings of the Conference on Corpus Linguistics*, pages 647–656. University of Lancaster.

Ramchand, G. (1997). *Aspect and Predication*. Oxford University Press, Oxford.

Reichenbach, H. (1947). *Elements of Symbolic Logic*. University of California Press, Berkeley.

Ritchie, G. (1979). Temporal clauses in English. *Theoretical Linguistics*, 6:87–115.

Sager, N., Lyman, M., Bucknall, C., Nhan, N., and Tick, L. (1994). Natural language processing and the processing of clinical data. *Journal of the American Medical Informatics Association*, 1:142–160.

Schank, R. (1975). The structure of episodes in memory. In Bobrow, D. and Collins, A., editors, *Representation and Understanding*, pages 237–272. Academic Press, New York.

Sperber, D. and Wilson, D. (1986). *Relevance*. Harvard, Cambridge, MA.

Stone, M. and Hardt, D. (1999). Dynamic discourse referents for tense and modals. In Bunt, H. and Muskens, R., editors, *Computing Meaning*, volume 1, pages 301–319. Springer, Berlin.

ter Meulen, A. (1995). *Representing Time in Natural Language: the Dynamic Interpretation of Tense and Aspect*. MIT Press, Cambridge, MA.

Truswell, R. (2007). *Locality of Wh-Movement and the Individuation of Events*. PhD thesis, University College London.

van Benthem, J. (1991). *Language in Action*. North Holland, Amsterdam.

van Lambalgen, M. and Hamm, F. (2005). *The Proper Treatment of Events*. Blackwell, Oxford.

Vardi, M. (2008). From Church and Prior to PSL. In Grumberg, O. and Veith, H., editors, *25 Years of Model Checking*, volume 5000 of *LNCS*, pages 150–171, Berlin. Springer.

Vendler, Z. (1967). *Linguistics in Philosophy*. Cornell University Press, Ithaca, NY.

Verhagen, M., Gaizauskas, R., Schilder, F., Hepple, M., Moszkowitcz, J., and Pustejovsky, J. (2009). The tempeval challenge: Identifying temporal relations in text. *Language Resources and Evaluation*, 43:161–179.

Webber, B. (1988). Tense as discourse anaphor. *Computational Linguistics*, 14:61–73.

Webber, B., Gardent, C., and Bos, J. (2002). Position statement: Inference in question answering. In *Proceedings of the International Conference on Language Resources and Evaluation*, Las Palmas. LREC.

White, M. (1994). *A Computational Approach to Aspectual Composition*. PhD thesis, University of Pennsylvania.

Wiebe, J., O'Hara, T., Öhrström, T., and McKeever, K. (1998). An empirical approach to temporal reference resolution. *Journal of Artificial Intelligence Research*, 9:247–293.