How domain-general can we be? Learning incremental Dialogue Systems without Dialogue Acts

Arash Eshghi
Interaction Lab
Heriot-Watt University
Edinburgh EH14 4AS
a.eshghi@hw.ac.uk

Oliver Lemon
Interaction Lab
Heriot-Watt University
Edinburgh EH14 4AS
o.lemon@hw.ac.uk

Abstract

Dialogue is domain-specific, in that the communicative import of utterances is severely underdetermined in the absence of a specific domain of language use. This has lead dialogue system developers to use various techniques to map dialogue utterances onto hand-crafted, highly domain-specific Dialogue Act (DA) representations, leading to systems which lack generality and do not easily scale or transfer to new domains. Here we first propose a new method which avoids the use of DAs altogether by combining an open-domain, incremental, semantic NL grammar for dialogue - Dynamic Syntax - with machine learning techniques for optimisation of dialogue management and utterance generation. We then focus on a key sub-problem associated with this vision: automatically *grounding* domain-general semantic representations in the non-linguistic actions used in specific dialogue domains. Similar to some recent work on open-domain question answering, we present an algorithm that clusters domain-general semantic representations of dialogue utterances based on computing *pragmatic synonymy*, in effect automatically inducing a more coarse-grained domain-specific semantic ontology than that encoded by open-domain semantic grammars.

1 Introduction

"How many kinds of sentence are there? Say assertion, question, command? – there are countless kinds: countless different kinds of use of what we call "symbols", "words", "sentences". And this multiplicity is not something fixed, given once and for all; but new types of language, new language-games, as we may say, come into existence, and others become obsolete and get forgotten." (Wittgenstein, 1953)

Perhaps the most unyielding obstacle in the working out of sufficiently general models of meaning in dialogue is the astonishingly wide and open-ended range of communicative effects that people can achieve with language in different contexts of use. This is not just a matter of structural context-dependence of fragments, ellipsis and anaphora for which there are increasingly general accounts (see e.g. Ginzburg (2012); Kempson et al. (forthcoming); Kamp&Reyle (1993)). Even when a fully specified semantic representation in some logical language is derived for an utterance, the communicative import of the representation is severely underdetermined in the absence of a known activity, a 'language-game', that the representation is deployed in. Conversely, even within a simple domain, there's a lot of variation in language use that does not ultimately affect the overall communicative goal of the dialogue. For example, in the travel domain, the following dialogues all lead to a context in which A is committed to booking a ticket for B from London to Paris: (a) *A: Where would you like to go? B: Paris, from London*; (b) *A: Where is your destination? B: Paris, A: And your port of departure? B: London.* (c) *B: I need to get to Paris from London, A: Sure*. These dialogues can be said to be *pragmatically synonymous modulo the travel domain*. What is striking about these simple examples is that much of this synonymy breaks down if one moves to another domain (e.g. example (b) where A is an immigration officer): pragmatic synonymy relations are domain-specific.

To bypass this difficulty, Spoken Dialogue Systems (SDS) designers/researchers have used hand-crafted representations of the communicative content of utterances in specific domains, in the form of Dialogue Acts (DA)[1], designed to capture the

[1]Here we use the term "dialogue act" to encompass the whole semantic representation used, ie. standard dialogue acts such as "inform" together with content such as "desti-

specific information needed to complete specific tasks. DAs operate at the interfaces between the core system components in a SDS - Dialogue Management (DM), Natural Language Generation (NLG), and Spoken Language Understanding (SLU) - and have thus lead to systems that lack generality, and are difficult or impossible to transfer to new domains. DAs form a bottleneck representation between SLU and DM, and between DM and NLG. In addition, from a machine-learning point of view DA representations may either under- or over-estimate the features required for learning good DM and/or NLG policies for a domain.

1.1 Structure of the paper

In this paper, we first propose a novel architecture for data-driven learning of fully incremental dialogue systems with little supervision beyond raw dialogue transcripts, which avoids the use of DAs altogether. DAs are instead generated as emergent properties of semantic representations of utterances in specific domains, formed by combining basic semantic units which are delivered by open-domain incremental, semantic grammars[2].

While we do not dispute people's sensitivity to DAs as more coarse-grained units of meaning, here we operate under the assumption that, given a set, stable domain of language use - such as buying a drink at a bar, ordering food in a restaurant, booking a flight, etc. - to which interlocutors are already attuned, the low-level semantic features of utterances are sufficient to encode their pragmatic force, and therefore, that Dialogue Acts need not be explicitly represented[3].

Instead, the appropriate level of meaning representation for a domain will be learned - rather than hand-crafted/designed - from a set of successful in-domain dialogues with no DA annotations. These dialogues are first parsed using Dynamic Syntax (Kempson et al., 2001; Cann et al., 2005), which maps them to open-domain semantic representations of the final contexts reached by the interlocutors, i.e. the semantic content that they jointly commit to. In order to capture the domain-specific pragmatic synonymy relations described above, we will assume a weak form of supervision: that the dialogues are annotated with representations of the non-linguistic actions taken and when, e.g. a data-base query, a flight booking, serving a drink, etc. A function is then learnt which maps these contexts to the non-linguistic action representations. Effectively, this function maps the very fine-grained semantic ontology encoded by the open-domain DS grammars (or any open-domain semantic parser), onto a more coarse-grained ontology with fewer semantic distinctions, based on pragmatic synonymy. It is an algorithm for learning this function that we then focus on in this paper.

First we review some recent related work, in section 2. Then we present the overall model and framework that we are developing fror this problem, in section 3. In section 4 we present the algorithm we have developed for computing the pragmatic synonymy function.

2 Related work

There has been a recent surge of interest in domain-general or "open-domain" semantic parsing. Most similar to our work is perhaps that of (Allen et al., 2007; Dzikovska et al., 2008) who devise a system for mapping open-domain logical forms in a formalism that is similar to Minimal Recursion Semantics (the LF representation), onto domain-specific representations suitable for reasoning and planning within a specific dialogue domain (the KR representation). However, unlike the architecture proposed here, the ontology mappings are defined by hand, rather than learned from data, and the grammar employed is not incremental.

There's also the work of (Kwiatkowski et al., 2013), who map open-domain CCG semantic parses to Freebase for question-answering. Here, an open-domain Question-Answering system (note: not a full dialogue system) is learned by using a wide-coverage CGG parser over questions. Kwiatkowski et al. (2013) develop a method for automatically mapping CCG semantic LFs onto the Freebase ontology, which is similar in spirit to the algorithm we present in section 4. In our case, the ontology is not that provided by Freebase (although nothing prohibits this), but instead the ontology of back-end application actions used in spe-

nation=Dublin").

[2]Note that these grammars will also deliver generic speech act representations such as "question" and "acknowledgement" which we will learn the import of in specific domains of usage.

[3]The question of how interlocutors come to coordinate on the structure of an activity, i.e. how language-games emerge in the first place, is a challenging one. We put this problem on one side here, but see e.g. Healey (2008); Mills (2013 in press); Mills & Gregoromichelaki (2010).

cific dialogue systems (e.g. searching for a flight from X to Y, paying a bill, etc). At a high level, the problem is similar: mapping domain-general semantic representations onto an ontology, though Kwiatkowski et al. (2013) do not need to consider sequences of sentences / utterances, or dialogue acts. Similar work is presented by (Cai and Yates, 2013b; Cai and Yates, 2013a), who also work using Freebase and do not consider dialogues. Their system maps English words onto individual Freebase symbols, and does not handle conjunctions and disjunctions of ontology symbols, as our approach and that of Kwiatkowski et al. (2013) do.

3 Overall model

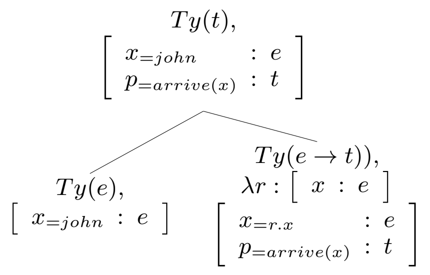
Before presenting our main algorithm, we first outline the overall method we propose of combining (1) Dynamic Syntax (DS), a domain-general incremental, semantic grammar framework, shown to be uniquely well-placed in capturing the fragmentary and context-dependent nature of spontaneous dialogue (Gargett et al., 2009; Gregoromichelaki et al., 2009); and (2) statistical machine learning with data-driven optimisation methods which are known to robustly handle noise and uncertainty in spoken language. DS will provide the domain-general semantic parsing (i.e. SLU) and surface realisation (i.e. low-level language generation) components, and machine learning for DM will provide the crucial bridge between them and higher-level action and content selection processes. In order to integrate these components, and to use dialogue data for training, we require a 'pragmatic synonymy" function mapping semantic representations provided by DS into specific dialogue system domain ontologies. We present this is section 4.

We first introduce and motivate the particular open-domain semantic parsing formalism that we will use in this work, and then explain the the proposed overall method (see section 3.2).

3.1 Dynamic Syntax and TTR (DS-TTR)

For the required semantic parser, we use a well-established semantic parsing framework, Dynamic Syntax (DS, (Kempson et al., 2001)), which models dialogue as a word-by-word *incremental, interactive* process of constructing meaning representations, with no intermediary layer of syntactic structure over words. We choose this rather than other possible semantic formalisms (e.g. CCG) because it has been shown to be uniquely well-placed in capturing the inherent fragmentary and context-dependent nature of spontaneous dialogue (Eshghi et al., 2012; Gregoromichelaki et al., 2013 in press; Gargett et al., 2009). Since DS is inherently incremental, and not sentence-based, it enables the word-by-word exploration – babbling – of the space of possible grammatical dialogues and their corresponding contexts within a given domain (see e.g. Fig. 3).

In DS, grammaticality is defined as parsability in context; words are associated with conditional Lexical Actions that monotonically update (partial) *semantic trees*, representing predicate argument structure with new semantic information and/or requirements for information to come; there are also Computational Actions, specifying general logical tree operations (e.g. beta-reduction of daughters), and strategies to adjust context for parsing of subsequent words. DS is bidirectional with generation defined in terms of parsing, and operating over the same meaning representations: a dialogue agent can switch from parser to generator (and vice versa) at any point (subsententially, as well as at sentence boundaries), where the generator starts where the parser finished, i.e. the context for generation will be the (partial) semantic tree derived by the parser so far. Dialogue fragments, including corrections, clarification ellipsis, short answers, adjuncts and continued utterances are all modelled grammar- internally in this way (Gregoromichelaki et al., 2009; Gargett et al., 2009).



$Ty(t),$
$\left[\begin{array}{ll} x_{=john} & : e \\ p_{=arrive(x)} & : t \end{array}\right]$
$Ty(e),$
$\left[\begin{array}{ll} x_{=john} & : e \end{array}\right]$
$Ty(e \rightarrow t)),$
$\lambda r : \left[\begin{array}{ll} x & : e \end{array}\right]$
$\left[\begin{array}{ll} x_{=r.x} & : e \\ p_{=arrive(x)} & : t \end{array}\right]$

Figure 1: Complete semantic tree for "John arrives". Nodes are decorated with semantic type and formulae.

Type Theory with Records (TTR) Type Theory with Records (TTR) is an extension of standard type theory shown useful in semantics and dialogue modelling (Cooper, 2005; Ginzburg, 2012).

To accommodate dialogue processing, and allow for richer representations of the dialogue context recent work has integrated DS and the TTR framework to replace the logical formalism in which meanings are expressed (Purver et al., 2010; Purver et al., 2011; Eshghi et al., 2012).

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

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

Figure 2: Example TTR record types

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

3.2 Proposed Overall Method: BABBLE

We start with two resources: a) a wide-coverage Dynamic Syntax parser L (either learned from data (Eshghi et al., 2013), or constructed by hand), for incremental spoken language understanding; b) a set D of transcribed successful example dialogues in the target application domain. Overall, we then need to perform 2 main steps: 1) extract the dialogue goal states from D using L, and 2) automatically generate jointly optimised Dialogue Manager and NLG components.

We then carry out the following steps, explained in greater detail below) to achieve steps 1 and 2:

Step 1.1 Parse all $d \in D$ using L, generating a set of final dialogue contexts, C, each a TTR Record Type representing the grounded semantic content for d; see Fig. 3[4] Collect the successful dialogues in D and extract the set of goal states A, represented as record types;

Step 1.2 Construct the Generalized Goal Context, *GGC*: the *maximally specific super-type* (the largest common denominator) of A;

Step 2.1 Automatically construct a Markov Decision Process (MDP) for D (see Fig. 3). Generate the state space S using feature function F defined to extract the semantic features (Record Types) in the *GGC* (i.e. the state space tracks all and only the semantic types present in the *GGC*), and compute the transition function T via the set of parsed dialogues, use L as the MDP action set, and define Reward function R as reaching the *GGC* state while minimising time penalties;

Step 2.2 Solve the generated MDP using Reinforcement Learning methods: train an action selection mechanism, where actions are system utterances of the lexical items $a \in L$, optimised via R. This process has a large action set, but action selection will be bounded via a measure of distance from *GGC* (see below) and is also constrained by the DS grammar.

The result will be the combined DM and NLG components of a dialogue system for D: i.e. a jointly optimised action selection mechanism for DM and NLG, with L providing the SLU component. Domain extension would then be a matter of adding new data and retraining the system. We now describe each of these steps in further detail.

Inducing the dialogue goal (Step 1). Recall the examples of pragmatic synonymy in dialogue given in the introduction, for example

(a) A: Where would you like to go? B: Paris, from London; (b) B: I would like to go to Paris; A: Sure, where from? B: London; (c) A: Where is your destination? B: Paris A: And your port of departure? B: London. (d) B: I need to get to Paris from London A: Sure. These dialogues can be said to be *Pragmatically Synonymous modulo the travel domain*. The source of this variation is twofold: structural, i.e. syntactic and interactional variation; and lexical-semantic, i.e. variation in the basic semantic ontology employed. While (a) and (b) differ only structurally, and not semantically, they differ from (c) and (d) on both levels.

The aim of this step is to extract automatically from D, a compact, tractable representation of a Generalised Goal Dialogue Context (GGC) that captures – abstracts over – both kinds of variation, and which the RL agent will later be trained to track and achieve in the MDP state space. The GGC thus constructed will allow the RL agent not

[4]In all our example context representations in TTR, information about commitment to content, and who said what is suppressed, but see (Purver et al., 2010) for how they are encoded in TTR.

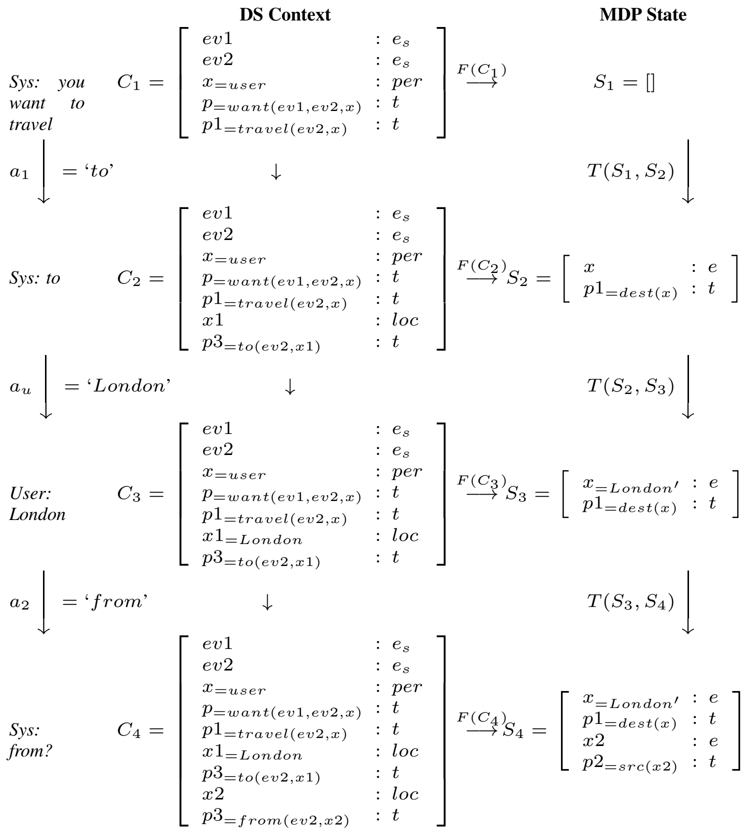


Figure 3: Example incremental action (word) selection via the BABBLE method. See Section 3.2.

only to reproduce the same diversity in its generated utterances, but understand and respond to the diverse language employed by its users, as exemplified in D, without recourse to hand-crafted dialogue act representations. Importantly, the GGC will also serve to constrain the very large space of dialogue policies that the RL agent would otherwise have to search/explore. The construction of the GGC will proceed in two generalisation stages: (1) structural: parsing all the dialogues in D with L producing a set of all the final contexts, C, reached by the dialogues in D; and (2) semantic: partitioning of the set of all semantic features of the C s into a set of equivalence classes, modulo pragmatic synonymy relations, forming, in effect, a domain-specific ontology. We explain these steps below:

1.1. Parsing dialogues with a DS grammar allows us to abstract away from the syntactic and interactional particularities of specific dialogues in D: dialogues are mapped onto domain-general semantic representations of the final contexts jointly established by the interlocutors, in effect allowing us to organise the dialogues in D into a set of structural equivalence classes. For example, dialogues (a) and (b) above will be grouped into the same class in virtue of giving rise to the same final context.

1.2. However, the DS grammar is domain-general, encoding a very fine-grained ontology of semantic types, i.e. lexical variation in the dialogues will always lead to semantic variation in the C's. But much of this variation is pragmatically inconsequential for task success within a given domain: for example modulo the travel domain, dialogues (b), (c) and (d) are pragmatically synonymous (c.f. in the question-answering case, (Kwiatkowski et al., 2013)).

Therefore, our goal here is to *create equivalence classes of the semantic features* (TTR record types) of the C, such that two features are placed in the same equivalence class if they make the same pragmatic contribution to in-domain task success. To achieve this, we can use a weak form of supervision: we can assume that the datasets D contain, in addition to raw dialogue transcripts, representations of the non-linguistic actions taken, e.g. data-base queries, flight bookings, serving a drink; depending on the domain. The semantic features of the C will then be grouped into equivalence classes in virtue of giving rise to the same non-linguistic actions, i.e. in virtue of being pragmatically equivalent. For example, dialogues (b) and (c) above will give rise to different final contexts, but both lead to the same non-linguistic action `book(Source=London, Dest=Paris)`. These action representations encode a domain-specific ontology and provide an interface between the domain-general semantic representations delivered by L and the extralinguistic context of the dialogue task. This process can thus be described as mapping a fine-grained, open domain, semantic ontology onto a more coarse-grained domain-specific one with fewer semantic distinctions, based on pragmatic synonymy relations. The task of finding this mapping is akin to that of (Kwiatkowski et al., 2013) who present a method for doing this, in order to produce an open-domain Question Answering system that uses an open domain CCG semantic parser. This is the main algorithm that we present in section 4.

Other steps are needed, in particular the reinforcement learning of incrementally generating lexical actions so as to achieve the GGC. We leave presentation of this method to future work, and

Mapping Type	Example mapping in FOL (Kwiatkowski et. al)	Example mapping in TTR (this paper)
Collapse (type e)	$\iota.x\ Public(x) \wedge Library(x) \rightarrow PL$	$\left[\begin{array}{lll} r & : & \left[\begin{array}{lll} x & : & e \\ p_{=Public(x)} & : & t \\ p1_{=Library(x)} & : & t \end{array}\right] \\ x_{=\iota(r.x,r)} & : & e \end{array}\right] \rightarrow \left[\begin{array}{lll} x_{=PL} & : & e \end{array}\right]$
Collapse (type t)	$capital(y) \wedge in(y,x) \rightarrow capitalof(x,y)$	$\left[\begin{array}{lll} x & : & e \\ y & : & e \\ p_{=capital(y)} & : & t \\ p1_{=in(y,x)} & : & t \end{array}\right] \rightarrow \left[\begin{array}{lll} x & : & e \\ y & : & e \\ p_{=capitalof(x,y)} & : & t \end{array}\right]$
Splitting	$capitalof(x,y) \rightarrow capital(y) \wedge in(y,x)$	$\left[\begin{array}{lll} x & : & e \\ y & : & e \\ p_{=capitalof(x,y)} & : & t \end{array}\right] \rightarrow \left[\begin{array}{lll} x & : & e \\ y & : & e \\ p_{=capital(y)} & : & t \\ p1_{=in(x,y)} & : & t \end{array}\right]$

Table 1: Examples of the different types of ontology mapping in FOL and TTR

here focus on step 1.2 above.

4 Pragmatic Synonymy: grounding semantic ontologies in action

In this section we describe an algorithm for learning a mapping F from semantic contexts derived from parsing in-domain dialogues with wide-coverage DS grammars, onto representations of the back-end, non-linguistic actions of the system, whose parameters together constitute the MDP state space (see above).

4.1 Types of synonymy mappings

Our aim here can be seen as somewhat similar to the work of Kwiatkowski et al. (2013), where an open-domain Question-Answering system (note: not a full dialogue system) is learned by using a wide-coverage CGG parser over questions. Kwiatkowski et al. (2013) develop a method for automatically mapping CCG semantic parses (of questions, not dialogues) onto a particular knowledge base ontology (in our case, the application back-end actions, such as database searches, flight bookings, etc). Overall, two types of mappings between meaning representations are discussed, *collapsing* and *splitting* ontology constants of different types(e.g. type e or t). Table 1 shows examples of these in First-Order Logic (FOL) as per Kwaitkowski et al. and Record Types (RT) of the Type Theory with Records used in this paper:

As noted by Kwiatkowski et al. (2013), the full set of possible collapses of an input meaning representation MR is limited by its number of constants, since each collapse removes at east one constant. The number of possible collapses is therefore polynomial in the number of constants in MR and exponential in the arity of the most complex type in the ontology. For typical dialogue system domains this arity is only 2 or 3. The splitting operation covers cases where multiple constants in the ontology represent the meaning of a single word. To constrain complexity, we can limit the splitting operation to apply only once for each underspecified constant in MR.

$$\left\langle \left[\begin{array}{lll} ev1 & : & e_s \\ ev2 & : & e_s \\ x_{=user} & : & per \\ p_{=want(ev1,ev2,x)} & : & t \\ p1_{=travel(ev2,x)} & : & t \\ x1_{=London} & : & loc \\ p3_{=to(ev2,x1)} & : & t \end{array}\right], \left[\begin{array}{lll} x_{=London'} & : & e \\ p1_{=dest(x)} & : & t \\ act_{=book(x)} & : & e \end{array}\right] \right\rangle$$

Figure 4: Example $\langle C, A\rangle$ pairing. C represents the context reached in: "A: I want to travel to London B: Sure", and A represents a booking action with London as destination

4.2 Problem Statement

Input A set, T, of training examples of the form $\langle C, A\rangle$ where each C is a domain-general record-type (RT) representation of the final semantic context reached by parsing an in-domain dialogue with DS; and A, also a RT, representing the non-linguistic, back-end action taken by the system at the point where C was reached. As such, the A encodes the domain-relevant information required by a dialogue system to complete its tasks. Figure 4 shows one training example in the travel domain.

Output A function $DCont : RecType \rightarrow RecType$ ($DCont$ stands for domain content, and is a function from TTR record types to TTR record types, see section 3.1), determined by a set of ordered pairs, $F = \{\langle c_1, a_1\rangle, \ldots, \langle c_n, a_n\rangle\}$, which, given new, unseen contexts - but *in part* similar to the training instances - extracts the domain-relevant information from them: F specifies which parts of the semantic information in the contexts - i.e. which supertypes of the context RTs - go on to make up which parts of the target action representations. F determines $DCont$ as follows:
$DCont(x) = \bigwedge_{\langle c,a\rangle \in S} a$, where, $S = \{\langle c,a\rangle \in F | x \sqsubseteq c\}$ ($\wedge$ represents the intersection of one or more types

(Cooper, 2005). The intersection is formed by the union of the fields in the record types, with fields that have the same label collapsing into one) $DCont$ has the following properties:

1. *Many-to-one:* Distinct semantic information in the Cs could, in the general case, be mapped onto the same action representation or parts thereof. This property ensures pragmatic synonymy relations among the supertypes of the Cs. For example, the semantics of "my destination is Paris" and that of "I want to travel to Paris", while being for the most part distinct, will be mapped onto the same booking action in the travel domain.

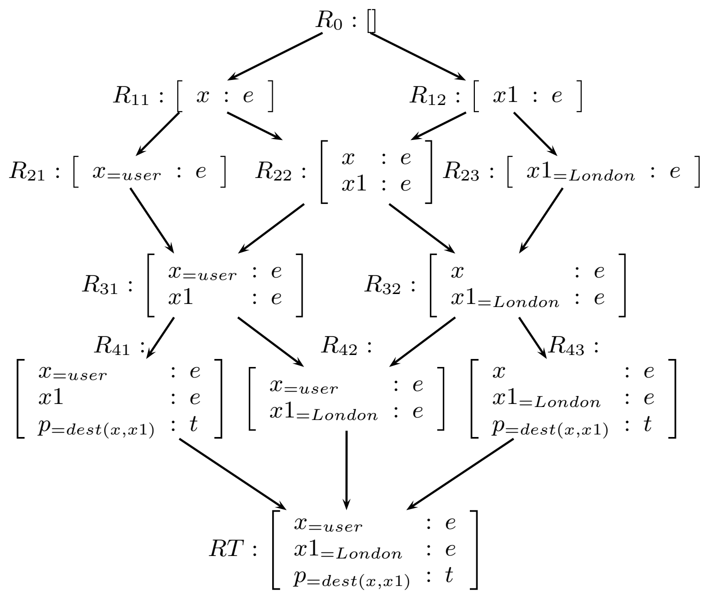
2. *Surjective over T:* The space of possible target action representations, i.e. the space of the supertypes of the As is fully covered by the mapping. Formally: $\forall\,(\langle C,A\rangle \in T)\,\exists\,(S \subseteq F)\,\left[\bigwedge_{\langle c,a\rangle \in S} a = A\right]$

3. *Maximally general over T:*

   (a) $\forall\,(\langle c_j,a_j\rangle \in F)\,\forall\,(\langle C,A\rangle \in T)\,[C \sqsubseteq c_j \rightarrow A \sqsubseteq a_j]$ i.e. that F generalises to - is correct for - T;
   (b) that anything less specific would not generalise to T:
   $\forall\,(\langle c_j,a_j\rangle \in F)\,\neg\exists c_k$
   $[c_j \sqsubset c_k \wedge \forall\,(\langle C,A\rangle \in T)\,[C \sqsubseteq c_k \rightarrow A \sqsubseteq a_j]]$,
   ensuring that F determines *the minimal* amount of semantic information needed in the contexts to determine some part of an action representation, i.e. that the domain of F remains most general (least specific).
   (c) similarly to (b), that the mappings determine *the maximal* amount of semantic information in the target action representations - the range of F - i.e. that for any $\langle c,a\rangle \in F$ anything *more specific* than a would not be sufficiently encoded by c.

4.3 Learning F

Hypothesising individual mappings using type lattices In processing each training pair $\langle C_i, A_i\rangle$, and enumerating mappings from C_i to A_i, the algorithm makes use of *type lattices*, constructed in advance for all the C_i and A_i. These encode the space of possible super-types of a record type RT - see Fig. 5 - with RT appearing at the bottom node, the empty type [] at the top node, and all super-types of RT in between getting progressively more specified as we move down



R_0 : []
R_{11} : [x : e]
R_{12} : [$x1$: e]
R_{21} : [$x_{=user}$: e]
R_{22} : [x : e / $x1$: e]
R_{23} : [$x1_{=London}$: e]
R_{31} : [$x_{=user}$: e / $x1$: e]
R_{32} : [x : e / $x1_{=London}$: e]
R_{41} : [$x_{=user}$: e / $x1$: e / $p_{=dest(x,x1)}$: t]
R_{42} : [$x_{=user}$: e / $x1_{=London}$: e]
R_{43} : [x : e / $x1_{=London}$: e / $p_{=dest(x,x1)}$: t]
RT : [$x_{=user}$: e / $x1_{=London}$: e / $p_{=dest(x,x1)}$: t]

Figure 5: RT hypothesis lattice

the lattice: the lattice is a partial order with $\sqsubseteq$ (is subtype of) being the order relation. Importantly, each edge is also a record type R_I representing the possible *minimal increments* from one RT, R_j, to another, R_{j+1}, such that $R_j \wedge R_I = R_{j+1}$ (see Eshghi et al. (2013) where lattices are similarly used to hypothesise semantic increments in a grammar induction task).

A pair of such lattices for each training example $\langle C_i, A_i\rangle$ (henceforth context lattice and action lattice) thus specifies a partial order on individual mappings $\langle c,a\rangle$ from supertypes of C_i onto supertypes of A_i: we can therefore explore the space of such mappings, in an order that guarantees that the first $\langle c,a\rangle$ encountered, that generalises to other training examples - that satisfies property 3(a) above - also satisfies 3(b) and 3(c), i.e. that c encodes the minimal amount of semantic information needed to determine a: the maximally specific supertype of A_i that generalises. Once any such $\langle c,a\rangle$ is found, we can mark c and a with *pointers* on the lattices, thus partitioning C_i and A_i into what has already been processed/consumed successfully (intersection of RT edges/increments leading to the root above the pointer), and what remains to be processed (intersection of RT edges/increments below the pointers). What remains of the exploration of the space of mappings can now take place, in a recursive fashion, on the sub-lattices whose roots the pointers now mark, with whatever falls outside these sub-lattices ignored in subsequent processing. Furthermore, in the processing of a subsequent training example, $\langle C_j, A_j\rangle$, the mappings already found for previous training examples and stored in F, if applicable (i.e. if for a $\langle c, a\rangle \in F$, $C_j \sqsubseteq c$ and $A_j \sqsubseteq a$) can be ‘applied’ immediately to $\langle C_j, A_j\rangle$, by moving the pointers on the corresponding lattices to c and a, thus precluding any repetitive processing across the training examples. In fact, given bounded semantic variability within a dialogue domain, if the first few training examples are varied enough, not much will remain to be done for later examples. This process is, in effect, a dynamic programming solution to the problem and thus gives us a handle on its exponential computational complexity.

```
input : A list $T$ of training pairs $[\langle C_1, A_1\rangle, \ldots, \langle C_n, A_n\rangle]$
output: The mapping $F$, a set of ordered pairs
Initialise $F = \{\}$;
Construct/Initialise Lattices for $T$;
    $lattices \leftarrow [\langle LC_1, LA_1\rangle, \ldots, \langle LC_n, LA_n\rangle]$;
for $i \leftarrow 1$ to $n$ do
    $\langle LC, LA\rangle \leftarrow lattices[i]$;
    $\langle LC, LA\rangle$.MovePointersTo($F$);
    while $\neg LA.pointerAtBottom()$ do
        CONTEXTINC: while HasMoreIncrements($LC$) do
            $c \leftarrow$ NextSmallestIncrement($LC$);
            ACTIONINC: while HasMoreIncrements($LA$)
            do
                $a \leftarrow$ NextLargestIncrement($LA$);
                for $j \leftarrow i+1$ to $n$ do
                    $\langle LC_j, LA_j\rangle \leftarrow lattices[j]$;
                    if $C_j \sqsubseteq c \wedge A_j \not\sqsubseteq a$ then
                        continue ACTIONINC;
                    end
                end
                $F.add(\langle c, a\rangle)$;
                $\langle LC, LA\rangle$.MovePointersTo($\langle c, a\rangle$)
            end
        end
    end
end
```

Algorithm 1: Learning F

Details of Algorithm 1 Algorithm 1 details the above process. Given current pointer positions on lattice pairs, $\langle LC, LA\rangle$, the functions, `NextSmallestIncrement(LC)` and `NextLargestIncrement(LA)` return the next least specific, and next most specific increments respectively. These are formed by intersecting the record types corresponding to edges on paths of increasing or decreasing length respectively, downwards through the lattice, from the current pointer position. The implementations of these functions are both in terms of a simple breadth first traversal of the sub-lattices whose roots are marked by the pointers - we suppress any detail here. The `HasMoreIncrements()` function is boolean valued, and determines whether the current sub-lattice is exhausted, i.e. whether all possible increments have already been returned. The function `MovePointersTo(`$\langle c, a\rangle$`)`, applied to a lattice pair, moves the pointers down to c and a on the context and action lattices, as described above. Finally, the inner most `for` loop, checks to see if the current mapping hypothesis generalises to the rest of the training examples, i.e. whether it has the property 3(a) above.

This algorithm covers the mapping types discussed in section 4.1: collapsing and splitting of ontology constants. To further constrain the search, we can incorporate the constraints discussed briefly in that section. Finally, we have not covered functional types here, but TTR affords the full power of the lambda calculus (Cooper, 2005), and these can be incorporated within the algorithm. We leave the details on one side here.

5 Summary and Future Work

We proposed a novel architecture for learning fully incremental dialogue systems with little supervision beyond raw dialogue transcripts and without recourse to dialogue act representations, by combining open-domain, incremental semantic grammars with state-of-the-art machine learning methods for learning NLG/DM policies. We argued that dialogue acts can instead be seen as emergent from learning, and that they need not be explicitly represented. We then focused on a key sub-problem associated with this vision: automatically *grounding* domain-general semantic representations in the non-linguistic actions used in specific dialogue domains. We presented an algorithm for learning such a mapping, which, in effect, clusters parts of domain-general semantic representations of dialogue contexts based on *pragmatic synonymy*, thus inducing a more coarse-grained domain-specific semantic ontology than that encoded by open-domain semantic grammars.

A major part of this paper is a proposal for a programme of research, and hence the most immediate future work consists in carrying out this research and implementing/evaluating the algorithms proposed.

6 Acknowledgments

The research leading to this work was funded by the EC FP7 programme FP7/2011-14 under grant agreement no. 287615 (PARLANCE). We also thank Matthew Purver and Robin Cooper for valuable discussions around the ideas presented here.

References

James Allen, Myroslava Dzikovska, Mehdi Manshadi, and Mary Swift. 2007. Deep linguistic processing for spoken dialogue systems. In *Proceedings of the the ACL workshop on Deep Linguistic Processing*.

Qingqing Cai and Alexander Yates. 2013a. Large-scale semantic parsing via schema matching and lexicon extension. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*.

Qingqing Cai and Alexander Yates. 2013b. Semantic parsing freebase: Towards open-domain semantic parsing. In *Proceedings of the Second Joint Conference on Lexical and Computational Semantics*.

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

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

Myroslava O. Dzikovska, James F. Allen, and Mary D. Swift. 2008. Linking semantic and knowledge representations in a multi-domain dialogue system. *Journal of Logic and Computation*, 18:405–430.

Arash Eshghi, Julian Hough, Matthew Purver, Ruth Kempson, and Eleni Gregoromichelaki. 2012. Conversational interactions: Capturing dialogue dynamics. In S. Larsson and L. Borin, editors, *From Quantification to Conversation: Festschrift for Robin Cooper*, pages 325–349.

Arash Eshghi, Julian Hough, and Matthew Purver. 2013. Incremental Grammar Inducion from Child-directed Dialogue Utterances. In *Proc. of ACL Workshop on CMCL*.

Andrew Gargett, Eleni Gregoromichelaki, Ruth Kempson, Matthew Purver, and Yo Sato. 2009. Grammar resources for modelling dialogue dynamically. *Cognitive Neurodynamics*, 3(4):347–363.

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

Eleni Gregoromichelaki, Yo Sato, Ruth Kempson, Andrew Gargett, and Christine Howes. 2009. Dialogue modelling and the remit of core grammar. In *Proceedings of IWCS*.

Eleni Gregoromichelaki, Ruth Kempson, Christine Howes, and Arash Eshghi. 2013, in press. On making syntax dynamic: the challenge of compound utterances and the architecture of the grammar. In Ipke Wachsmuth, Jan de Ruiter, Petra Jaecks, and Stefan Kopp, editors, *Alignment in Communication: Towards a New Theory of Communication*. Series: Advances in Interaction Studies (series editors: Kerstin Dautenhahn, Angelo Cangelosi).

Patrick G. T. Healey. 2008. Interactive misalignment: The role of repair in the development of group sub-languages. In R. Cooper and R. Kempson, editors, *Language in Flux*. College Publications.

Hans Kamp and Uwe Reyle. 1993. *From Discourse To Logic*. Kluwer Academic Publishers.

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

Ruth Kempson, Ronnie Cann, Arash Eshghi, Eleni Gregoromichelaki, and Matthew Purver. forthcoming. Ellipsis. In S. Lappin and C. Fox, editors, *Handbook of Contemporary Semantics*. Oxford University Press.

Tom Kwiatkowski, Eunsol Choi, Yoav Artzi, and Luke Zettlemoyer. 2013. Scaling semantic parsers with on-the-fly ontology matching. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

G. Mills and E. Gregoromichelaki. 2010. Establishing coherence in dialogue: sequentiality, intentions and negotiation. In *Proceedings of SemDial (PozDial)*.

Greg J. Mills. 2013, in press. Dialogue in joint activity: Complementarity, convergence and conventionalization. *New Ideas in Psychology*.

Matthew Purver, Eleni Gregoromichelaki, Wilfried Meyer-Viol, and Ronnie Cann. 2010. Splitting the 'I's and crossing the 'You's: Context, speech acts and grammar. In P. Łupkowski and M. Purver, editors, *Aspects of Semantics and Pragmatics of Dialogue. SemDial 2010, 14th Workshop on the Semantics and Pragmatics of Dialogue*, pages 43–50, Poznań, June. Polish Society for Cognitive Science.

Matthew Purver, Arash Eshghi, and Julian Hough. 2011. Incremental semantic construction in a dialogue system. In *Proc. Int. Conference on Computational Semantics*, pages 365–369.

Ludwig Wittgenstein. 1953. *Philosophical Investigations*. Blackwell Publishing.