Probabilistic Type Theory for Incremental Dialogue Processing

Julian Hough and Matthew Purver
Cognitive Science Research Group
School of Electronic Engineering and Computer Science
Queen Mary University of London
{j.hough,m.purver}@qmul.ac.uk

Abstract

We present an adaptation of recent work on probabilistic Type Theory with Records (Cooper et al., 2014) for the purposes of modelling the incremental semantic processing of dialogue participants. After presenting the formalism and dialogue framework, we show how probabilistic TTR type judgements can be integrated into the inference system of an incremental dialogue system, and discuss how this could be used to guide parsing and dialogue management decisions.

1 Introduction

While classical type theory has been the predominant mathematical framework in natural language semantics for many years (Montague, 1974, inter alia), it is only recently that probabilistic type theory has been discussed for this purpose. Similarly, type-theoretic representations have been used within dialogue models (Ginzburg, 2012); and probabilistic modelling is common in dialogue systems (Williams and Young, 2007, inter alia), but combinations of the two remain scarce. Here, we attempt to make this connection, taking (Cooper et al., 2014)'s probabilistic Type Theory with Records (TTR) as our principal point of departure, with the aim of modelling incremental inference in dialogue.

To our knowledge there has been no practical integration of probabilistic type-theoretic inference into a dialogue system so far; here we discuss computationally efficient methods for implementation in an extant incremental dialogue system. This paper demonstrates their efficacy in simple referential communication domains, but we argue the methods could be extended to larger domains and additionally used for on-line learning in future work.

2 Previous Work

Type Theory with Records (TTR) (Betarte and Tasistro, 1998; Cooper, 2005) is a rich type theory which has become widely used in dialogue models, including information state models for a variety of phenomena such as clarification requests (Ginzburg, 2012; Cooper, 2012) and non-sentential fragments (Fernández, 2006). It has also been shown to be useful for incremental semantic parsing (Purver et al., 2011), incremental generation (Hough and Purver, 2012), and recently for grammar induction (Eshghi et al., 2013).

While the technical details will be given in section 3, the central judgement in type theory $s : T$ (that a given object s is of type T) is extended in TTR so that s can be a (potentially complex) *record* and T can be a *record type* – e.g. s could represent a dialogue gameboard state and T could be a dialogue gameboard state type (Ginzburg, 2012; Cooper, 2012). As TTR is highly flexible with a rich type system, variants have been considered with types corresponding to real-number-valued perceptual judgements used in conjunction with linguistic context, such as visual perceptual information (Larsson, 2011; Dobnik et al., 2012), demonstrating its potential for embodied learning systems. The possibility of integration of perceptron learning (Larsson, 2011) and naive Bayes classifiers (Cooper et al., 2014) into TTR show how linguistic processing and probabilistic conceptual inference can be treated in a uniform way within the same representation system.

Probabilistic TTR as described by Cooper et al. (2014) replaces the categorical $s : T$ judgement with the real number valued $p(s : T) = v$ where $v \in [0,1]$. The authors show how standard probability theoretic and Bayesian equations can be applied to TTR judgements and how an agent might learn from experience in a simple classification game. The agent is presented with instances of

a situation and it learns with each round by updating its set of probabilistic type judgements to best predict the type of object in focus — in this case updating the probability judgement that something is an apple given its observed colour and shape $p(s : T_{apple} \mid s : T_{Shp}, s : T_{Col})$ where $Shp \in \{shp1, shp2\}$ and $Col \in \{col1, col2\}$. From a cognitive modelling perspective, these judgements can be viewed as probabilistic perceptual information derived from learning. We use similar methods in our toy domain, but show how prior judgements could be constructed efficiently, and how classifications can be made without exhaustive iteration through individual type classifiers.

There has also been significant experimental work on simple referential communication games in psycholinguistics, computational and formal modelling. In terms of production and generation, Levelt (1989) discusses speaker strategies for generating referring expressions in a simple object naming game. He showed how speakers use informationally redundant features of the objects, violating Grice's Maxim of Quantity. In natural language generation (NLG), referring expression generation (REG) has been widely studied (see (Krahmer and Van Deemter, 2012) for a comprehensive survey). The incremental algorithm (IA) (Dale and Reiter, 1995) is an iterative feature selection procedure for descriptions of objects based on computing the distractor set of referents that each adjective in a referring expression could cause to be inferred. More recently Frank and Goodman (2012) present a Bayesian model of optimising referring expressions based on surprisal, the information-theoretic measure of how much descriptions reduce uncertainty about their intended referent, a measure which they claim correlates strongly to human judgements.

The element of the referring expression domain we discuss here is incremental processing. There is evidence from (Brennan and Schober, 2001)'s experiments that people reason at an incredibly time-critical level from linguistic information. They demonstrated *self-repair* can speed up semantic processing (or at least object reference) in such games, where an incorrect object being partly vocalized and then repaired in the instructions (e.g. "the yell-, uh, purple square") yields quicker response times from the onset of the target ("purple") than in the case of the fluent instructions ("the purple square"). This example will be addressed in section 5. First we will set out the framework in which we want to model such processing.

3 Probabilistic TTR in an incremental dialogue framework

In TTR (Cooper, 2005; Cooper, 2012), the principal logical form of interest is the *record type* ('RT' from here), consisting of sequences of *fields* of the form $[\, l : T \,]$ containing a label l and a type T.[1] RTs can be witnessed (i.e. judged as inhabited) by *records* of that type, where a record is a set of label-value pairs $[\, l = v \,]$. The central type judgement in TTR that a record s is of (record) type R, i.e. $s : R$, can be made from the component type judgements of individual fields; e.g. the one-field record $[\, l = v \,]$ is of type $[\, l : T \,]$ just in case v is of type T. This is generalisable to records and RTs with multiple fields: a record s is of RT R if s includes fields with labels matching those occurring in the fields of R, such that all fields in R are matched, and all matched fields in s must have a value belonging to the type of the corresponding field in R. Thus it is possible for s to have more fields than R and for $s : R$ to still hold, but not vice-versa: $s : R$ cannot hold if R has more fields than s.

$$R_1 : \left[\begin{array}{l} l_1 : T_1 \\ l_2 : T_2 \\ l_3 : T_3(l_1) \end{array}\right] \quad R_2 : \left[\begin{array}{l} l_1 : T_1 \\ l_2 : T_{2'} \end{array}\right] \quad R_3 : [\,]$$

Figure 1: Example TTR record types

Fields can have values representing predicate types (*ptypes*), such as T_3 in Figure 1, and consequently fields can be *dependent* on fields preceding them (i.e. higher) in the RT, e.g. l_1 is bound in the predicate type field l_3, so l_3 depends on l_1.

Subtypes, meets and joins A relation between RTs we wish to explore is $\sqsubseteq$ ('is a subtype of'), which can be defined for RTs in terms of fields as simply: $R_1 \sqsubseteq R_2$ if for all fields $[\, l : T_2 \,]$ in R_2, R_1 contains $[\, l : T_1 \,]$ where $T_1 \sqsubseteq T_2$. In Figure 1, both $R_1 \sqsubseteq R_3$ and $R_2 \sqsubseteq R_3$; and $R_1 \sqsubseteq R_2$ iff $T_2 \sqsubseteq T_{2'}$. The transitive nature of this relation (if $R_1 \sqsubseteq R_2$ and $R_2 \sqsubseteq R_3$ then $R_1 \sqsubseteq R_3$) can be used effectively for type-theoretic inference.

[1] We only introduce the elements of TTR relevant to the phenomena discussed below. See (Cooper, 2012) for a detailed formal description.

We also assume the existence of *manifest* (singleton) types, e.g. T_a, the type of which only a is a member. Here, we write manifest RT fields such as $[\, l : T_a \,]$ where $T_a \sqsubseteq T$ using the syntactic sugar $[\, l_{=a} : T \,]$. The subtype relation effectively allows progressive instantiation of fields (as addition of fields to R leads to R' where $R' \sqsubseteq R$), which is practically useful for an incremental dialogue system as we will explain.

We can also define *meet* and *join* types of two or more RTs. The representation of the meet type of two RTs R_1 and R_2 is the result of a merge operation (Larsson, 2010), which in simple terms here can be seen as union of fields. A meet type is also equivalent to the extraction of a maximal common subtype, an operation we will call $MaxSub(R_i..R_n)$:[2]

$$\text{if } R_1 = \begin{bmatrix} l_1 : T_1 \\ l_2 : T_2 \end{bmatrix} \text{and } R_2 = \begin{bmatrix} l_2 : T_2 \\ l_3 : T_3 \end{bmatrix}$$
$$R_1 \wedge R_2 = \begin{bmatrix} l_1 : T_1 \\ l_2 : T_2 \\ l_3 : T_3 \end{bmatrix}$$
$$= MaxSub(R_1, R_2)$$

R_1 and R_2 here are common $supertypes$ of the resulting $R_1 \wedge R_2$. On the other hand, the join of two RTs R_1 and R_2, the type $R_1 \vee R_2$ cannot be represented by field intersection. It is defined in terms of type checking, in that $s : R_1 \vee R_2$ iff $s : R_1$ or $s : R_2$. It follows that if $R_1 \sqsubseteq R_2$ then $s : R_1 \wedge R_2$ iff $s : R_1$, and $s : R_1 \vee R_2$ iff $s : R_2$.

While technically the maximally common supertype of R_1 and R_2 is the join type $R_1 \vee R_2$, here we introduce the maximally common *simple* (non disjunctive) supertype of two RTs R_1 and R_2 as field intersection:

$$\text{if } R_1 = \begin{bmatrix} l_1 : T_1 \\ l_2 : T_2 \end{bmatrix} \text{and } R_2 = \begin{bmatrix} l_2 : T_2 \\ l_3 : T_3 \end{bmatrix}$$
$$MaxSuper(R_1, R_2) = \begin{bmatrix} l_2 : T_2 \end{bmatrix}$$

We will explore the usefulness of this new operation in terms of RT lattices in sec. 4.

3.1 Probabilistic TTR

We follow Cooper et al. (2014)'s recent extension of TTR to include probabilistic type judgements of the form $p(s : R) = v$ where $v \in [0,1]$, i.e. the real valued judgement that a record s is of RT R. Here we use probabilistic TTR to model a common psycholinguistic experimental set up in section 5. We repeat some of Cooper et al.'s calculations here for exposition, but demonstrate efficient graphical methods for generating and incrementally retrieving probabilities in section 4.

Cooper et al. (2014) define the probability of the meet and join types of two RTs as follows:

$$\begin{aligned} p(s : R_1 \wedge R_2) &= p(s : R_1)p(s : R_2 \mid s : R_1) \\ p(s : R_1 \vee R_2) &= p(s : R_1) + p(s : R_2) - p(s : R_1 \wedge R_2) \end{aligned} \tag{1}$$

It is practically useful, as we will describe below, that the join probability can be computed in terms of the meet. Also, there are equivalences between meets, joins and subtypes in terms of type judgements as described above, in that assuming if $R_1 \sqsubseteq R_2$ then $p(s : R_2 \mid s : R_1) = 1$, we have:

$$\begin{aligned} &\text{if } R_1 \sqsubseteq R_2 \\ p(s : R_1 \wedge R_2) &= p(s : R_1) \\ p(s : R_1 \vee R_2) &= p(s : R_2) \\ p(s : R_1) &\leq p(s : R_2) \end{aligned} \tag{2}$$

The conditional probability of a record being of type R_2 given it is of type R_1 is:

$$p(s : R_2 \mid s : R_1) = \frac{p(s : R_1 \wedge s : R_2)}{p(s : R_1)} \tag{3}$$

We return to an explanation for these classical probability equations holding within probabilistic TTR in section 4.

Learning and storing probabilistic judgements When dealing with referring expression games, or indeed any language game, we need a way of storing perceptual experience. In probabilistic TTR this can be achieved by positing a judgement set J in which an agent stores probabilistic type judgements.[3] We refer to the sum of the value of probabilistic judgements that a situation has been judged to be of type R_i within J as $\|R_i\|_J$ and the sum of all probabilistic judgements in J simply as $P(J)$; thus the prior probability that anything is of type R_i under the set of judgements J is $\frac{\|R_i\|_J}{P(J)}$. The conditional probability $p(s : R_1 \mid s : R_2)$ under J can be reformulated in terms of these sets of judgements:

$$p_J(s : R_1 \mid s : R_2) = \begin{cases} \frac{\|R_1 \wedge R_2\|_J}{\|R_2\|_J} & \text{iff } \|R_2\|_J \neq 0 \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

[2] Here we concern ourselves with simple examples that avoid label-type clashes between two RTs (i.e. where R_1 contains $l_1 : T1$ and R_2 contains $l_1 : T2$); in these cases the operations are more complex than field concatenation/sharing.

[3] (Cooper et al., 2014) characterise a type judgement as an Austinian proposition that a situation is of a given type with a given probability, encoded in a TTR record.

where the sample spaces $\|R_1 \wedge R_2\|_J$ and $\|R_2\|_J$ constitute the observations of the agent so far. J can have new judgements added to it during learning. We return to this after introducing the incremental semantics needed to interface therewith.

3.2 DS-TTR and the DyLan dialogue system

In order to permit type-theoretic inference in a dialogue system, we need to provide suitable TTR representations for utterances and the current pragmatic situation from a parser, dialogue manager and generator as instantaneously and accurately as possible. For this purpose we use an incremental framework DS-TTR (Eshghi et al., 2013; Purver et al., 2011) which integrates TTR representations with the inherently incremental grammar formalism Dynamic Syntax (DS) (Kempson et al., 2001).

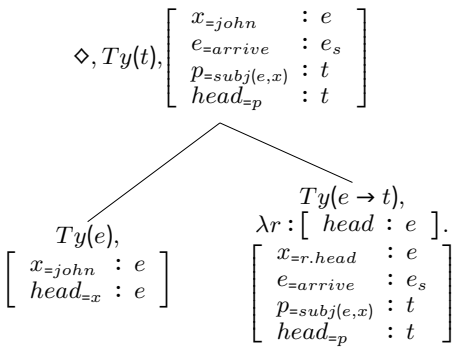

Figure 2: DS-TTR tree

DS produces an incrementally specified, partial logical tree as words are parsed/generated; following Purver et al. (2011), DS tree nodes are decorated not with simple atomic formulae but with RTs, and corresponding lambda abstracts representing functions of type $RT \rightarrow RT$ (e.g. $\lambda r : [\, l_1 : T_1 \,].[\, l_{2=r.l_1} : T_1 \,]$ where $r.l_1$ is a *path* expression referring to the label l_1 in r) – see Figure 2. Using the idea of manifestness of fields as mentioned above, we have a natural representation for underspecification of leaf node content, e.g. $[\, x : e \,]$ is unmanifest whereas $[\, x_{=john} : e \,]$[4] is manifest and the latter is a subtype of the former. Functional application can apply incrementally, allowing a RT at the root node to be compiled for any partial tree, which is incrementally further specified as parsing proceeds (Hough and Purver, 2012). Within a given parse path, due to DS-TTR's monotonicity, each maximal RT of the tree's root node is a subtype of the parser's previous maximal output.

Following (Eshghi et al., 2013), DS-TTR tree nodes include a field $head$ in all RTs which corresponds to the DS tree node type. We also assume a neo-Davidsonian representation of predicates, with fields corresponding to an event term and to each semantic role; this allows all available semantic information to be specified incrementally in a strict subtyping relation (e.g. providing the $subj()$ field when subject but not object has been parsed) – see Figure 2.

We implement DS-TTR parsing and generation mechanisms in the `DyLan` dialogue system[5] within Jindigo (Skantze and Hjalmarsson, 2010), a Java-based implementation of the incremental unit (IU) framework of (Schlangen and Skantze, 2009). In this framework, each module has input and output IUs which can be `added` as edges between vertices in module buffer graphs, and become `committed` should the appropriate conditions be fulfilled, a notion which becomes important in light of hypothesis change and repair situations. Dependency relations between different graphs within and between modules can be specified by *groundedIn* links (see (Schlangen and Skantze, 2009) for details).

The `DyLan` interpreter module (Purver et al., 2011) uses Sato (2011)'s insight that the context of DS parsing can be characterized in terms of a Directed Acyclic Graph (DAG) with trees for nodes and DS actions for edges. The module's state is characterized by three linked graphs as shown in Figure 3:

- *input:* a time-linear word graph posted by the ASR module, consisting of word hypothesis edge IUs between vertices W_n
- *processing:* the internal DS parsing DAG, which adds parse state edge IUs between vertices S_n *groundedIn* the corresponding word hypothesis edge IU
- *output:* a concept graph consisting of domain concept IUs (RTs) as edges between vertices C_n, *groundedIn* the corresponding path in the DS parsing DAG

Here, our interest is principally in the parser output, to support incremental inference; a DS-TTR generator is also included which uses RTs as goal concepts (Hough and Purver, 2012) and uses the

[4] This is syntactic sugar for $[\, x : e_{john} \,]$ and the = sign is not the same semantically as that in a record.

[5] Available from http://dylan.sourceforge.net/

same parse graph as the interpreter to allow self-monitoring and compound contributions, but we omit the details here.

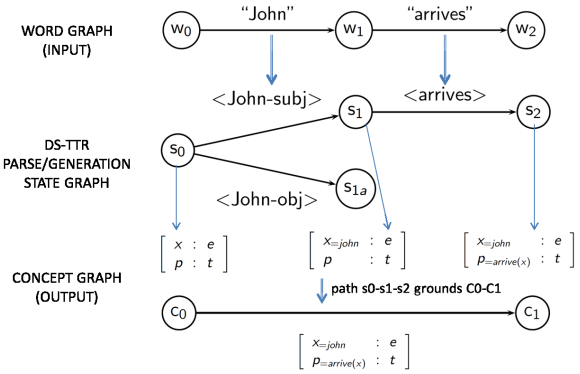

Figure 3: Normal incremental parsing in Dylan

4 Order theoretic and graphical methods for probabilistic TTR

RT lattices to encode domain knowledge To support efficient inference in DyLan, we represent dialogue domain concepts via partially ordered sets (*posets*) of RT judgements, following similar insights used in inducing DS-TTR actions (Eshghi et al., 2013). A poset has several advantages over an unordered list of un-decomposed record types: the possibility of incremental type-checking; increased speed of type-checking, particularly for pairs of/multiple type judgements; immediate use of type judgements to guide system decisions; inference from negation; and the inclusion of learning within a domain. We leave the final challenge for future work, but discuss the others here.

We can construct a poset of type judgements for any single RT by decomposing it into its constituent supertype judgements in a *record type lattice*. Representationally, as per set-theoretic lattices, this can be visualised as a Hasse diagram such as Figure 4, however here the ordering arrows show $\sqsubseteq$ ('subtype of') relations from descendant to ancestor nodes.

To characterize an RT lattice G ordered by $\sqsubseteq$, we adapt Knuth (2005)'s description of lattices in line with standard order theory: for a pair of RT elements R_x and R_y, their lower bound is the set of all $R_z \in G$ such that $R_z \sqsubseteq R_x$ and $R_z \sqsubseteq R_y$. In the event that a unique greatest lower bound exists, this is their meet, which in G happily corresponds to the TTR meet type $R_x \wedge R_y$. Dually, if their unique least upper bound exists, this is their

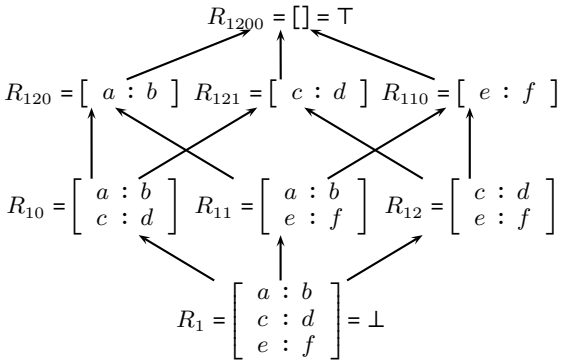

Figure 4: Record Type lattice ordered by the subtype relation

join and in TTR terms is $MaxSuper(R_x, R_y)$ but not necessarily their join type $R_x \vee R_y$ as here we concern ourselves with simple RTs. One element *covers* another if it is a direct successor to it in the subtype ordering relation hierarchy. G has a greatest element ($\top$) and least element ($\bot$), with the *atoms* being the elements that cover $\bot$; in Figure 4 if R_1 is viewed as $\bot$, the atoms are $R_{\{10,11,12\}}$. An RT element R_x has a *complement* if there is a unique element $\neg R_x$ such that $MaxSuper(R_x, \neg R_x) = \top$ and $R_x \wedge \neg R_x = \bot$ (the lattice in Figure 4 is *complemented* as this holds for every element).

Graphically, the join of two elements can be found by following the connecting edges upward until they first converge on a single RT, giving us $MaxSuper(R_{10}, R_{12}) = R_{121}$ in Figure 4, and the meet can be found by following the lines downward until they connect to give their meet type, i.e. $R_{10} \wedge R_{12} = R_1$.

If we consider R_1 to be a domain concept in a dialogue system, we can see how its RT lattice G can be used for incremental inference. As incrementally specified RTs become available from the interpreter they are matched to those in G to determine how far down towards the final domain concept R_1 our current state allows us to be. Different sequences of words/utterances lead to different paths. However, any practical dialogue system must entertain more than one possible domain concept as an outcome; G must therefore contain multiple possible final concepts, constituting its atoms, each with several possible dialogue move sequences, which correspond to possible downward paths – e.g. see the structure of Figure 5. Our aim here is to associate each RT in G with a probabilistic judgement.

Initial lattice construction We define a simple bottom-up procedure in Algorithm 1 to build a RT

lattice G of all possible simple domain RTs and their prior probabilistic judgements, initialised by the disjunction of possible final state judgements (the priors),[6] along with the absurdity $\bot$, stipulated a priori as the least element with probability 0 and the meet type of the atomic priors. The algorithm recursively removes one field from the RT being processed at a time (except fields referenced in a remaining dependent *ptype* field), then orders the new supertype RT in G appropriately.

Each node in G contains its RT R_i and a sum of probability judgements $\{\|R_k\|_J + .. + \|R_n\|_J\}$ corresponding to the probabilities of the priors it stands in a supertype relation to. These sums are propagated up from child to parent node as it is constructed. It terminates when all simple maximal supertypes[7] have been processed, leaving the maximally common supertype as $\top$ (possibly the empty type [ ]), associated with the entire probability mass $P(J)$, which constitutes the denominator to all judgements- given this, only the numerator of equation $\frac{\|R_i\|_J}{P(J)}$ needs to be stored at each node.

Algorithm 1 Probabilistic TTR record type lattice construction algorithm

```
INPUT: priors                          ▷ use the initial prior judgements for G's atoms
OUTPUT: G
G = newGraph(priors)                   ▷ P(J) set to equal sum of prior probs
agenda = priors                        ▷ Initialise agenda
while not agenda is empty do
    RT = agenda.pop()
    for field ∈ RT do
        if field ∈ RT.paths then       ▷ Do not remove bound fields
            continue
        superRT = RT - field
        if superRT ∈ G then            ▷ not new? order w.r.t. RT and inherit RT's priors
            G.order(RT.address,G.getNode(superRT),⊑)
        else                           ▷ new?
            superNode = G.newNode(superRT)   ▷ create new node w. empty priors
            for node ∈ G do            ▷ order superNode w.r.t. other nodes in G
                if superRT.fields ⊂ node.fields then
                    G.order(node,superNode,⊑)   ▷ superNode inherits node's priors
            agenda.append(superRT)     ▷ add to agenda for further supertyping
```

Direct inference from the lattice To explain how our approach models incremental inference, we assume Brennan and Schober (2001)'s experimental referring game domain described in section 2: three distinct domain situation RTs R_1, R_2 and R_3 correspond to a purple square, a yellow square and a yellow circle, respectively.

The RT lattice G constructed initially upon observation of the game (by instructor or instructee) shown in Figure 5 uses a uniform distribution for the three disjunctive final situations. Each node shows an RT R_i on the left and the derivation of its prior probability $p_J(R_i)$ that any game situation record will be of type R_i on the right, purely in terms of the relevant priors and the global denominator $P(J)$.

G can be searched to make inferences in light of partial information from an ongoing utterance. We model inference as predicting the likelihood of relevant type judgements $R_y \in G$ of a situation s, given the judgement $s : R_x$ we have so far. To do this we use conditional probability judgements following Knuth's work on distributive lattices, using the $\sqsubseteq$ relation to give a choice function:

$$p_J(s : R_y \mid s : R_x) = \begin{cases} 1 & \text{if } R_x \sqsubseteq R_y \\ 0 & \text{if } R_x \wedge R_y = \bot \\ p & \text{otherwise, where } 0 \le p \le 1 \end{cases} \quad (5)$$

The third case is the degree of inclusion of R_y in R_x, and can be calculated using the conditional probability calculation (4) in sec. 3. For negative RTs, a lattice generated from Algorithm 1 will be distributive but not guaranteed to be complemented, however we can still derive $p_J(s : R_y \mid s : \neg R_x)$ by obtaining $p_J(s : R_y)$ in G modulo the probability mass of R_x and that of its subtypes:

$$p_J(s : R_y \mid s : \neg R_x) = \begin{cases} 0 & \text{if } R_y \sqsubseteq R_x \\ \frac{p_J(s:R_y) - p_J(s:R_x \wedge R_y)}{p_J(s:\top) - p_J(s:R_x)} & \text{otherwise} \end{cases} \quad (6)$$

The subtype relations and atomic, join and meet types' probabilities required for (1) - (6) can be calculated efficiently through graphical search algorithms by characterising G as a DAG: the reverse direction of the subtype ordering edges can be viewed as reachability edges, making $\top$ the source and $\bot$ the sink. With this characterisation, if R_x is reachable from R_y then $R_x \sqsubseteq R_y$.

In DAG terms, the probability of the meet of two RTs R_x and R_y can be found at their highest common descendant node – e.g. $p_J(R_4 \wedge R_5)$ in Figure 5 can be found as $\frac{1}{3}$ directly at R_1. Note if R_x is reachable from R_y, i.e. $R_x \sqsubseteq R_y$, then due to the equivalences listed in (2), $p_J(R_x \wedge R_y)$ can be found directly at R_x. If the meet of two nodes is $\bot$ (e.g. R_4 and R_3 in Figure 5), then their meet probability is 0 as $p_J(\bot)$=0.

While the lattice does not have direct access to the join types of its elements, a join type probability $p_J(R_x \vee R_y)$ can be calculated in terms of $p_J(R_x \wedge R_y)$ by the join equation in (1), which holds for all probabilistic distributive lat-

[6] Although the priors' disjunctive probability sums to 1 after G is constructed, i.e. in Figure 5 $\frac{\|R_1\|_J + \|R_2\|_J + \|R_3\|_J}{P(J)} = 1$, the real values initially assigned to them need not sum to unity, as they form the atoms of G (see (Knuth, 2005)).

[7] Note that it does not generate the join types but maximal common supertypes defined by field intersection.

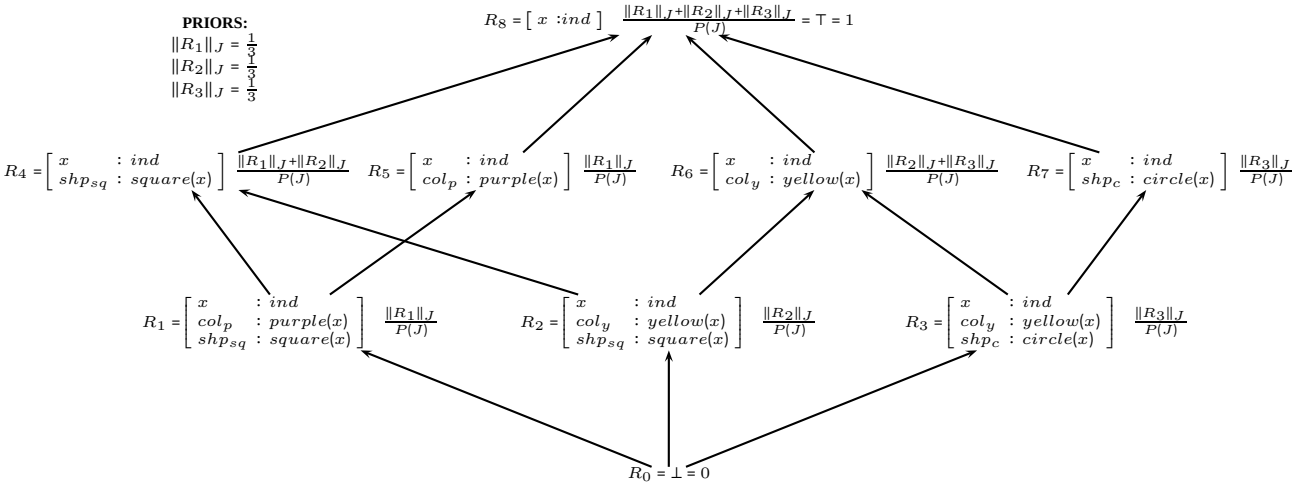

Figure 5: Record type lattice with initial uniform prior probablities

tices (Knuth, 2005).[8] As regards efficiency, worst case complexity for finding the meet probability at the common descendant of R_x and R_y is a linear $O(m+n)$ where m and n are the number of edges in the downward (possibly forked) paths $R_x \rightarrow \bot$ and $R_y \rightarrow \bot$.[9]

5 Simulating incremental inference and self-repair processing

Interpretation in DyLan and its interface to the RT lattice G follows evidence that dialogue agents parse self-repairs efficiently and that repaired dialogue content (reparanda) is given special status but not removed from the discourse context. To model Brennan and Schober (2001)'s findings of disfluent spoken instructions speeding up object recognition (see section 2), we demonstrate a self-repair parse in Figure 6 for "The yell-, uh, purple square" in the simple game of predicting the final situation from $\{R_1, R_2, R_3\}$ continuously given the type judgements made so far. We describe the stages T1-T4 in terms of the current word being processed- see Figure 6:

At **T1:'the'** the interpreter will not yield a subtype checkable in G so we can only condition on R_8 ($\top$), giving us $p_J(s : R_i \mid s : R_8) = \frac{1}{3}$ for $i \in \{1, 2, 3\}$, equivalent to the priors. At **T2: 'yell-'**, the best partial word hypothesis is now "yellow";[10] the interpreter therefore outputs an RT which matches the type judgement $s : R_6$ (i.e. that the object is a yellow object). Taking this judgement as the conditioning evidence using function (5) we get $p_J(s : R_1 \mid s : R_6) = 0$ and using (4) we get $p_J(s : R_2 \mid s : R_6) = 0.5$ and $p_J(s : R_3 \mid s : R_6) = 0.5$ (see the schematic probability distribution at stage T2 in Figure 6 for the three objects). The meet type probabilities required for the conditional probabilities can be found graphically as described above.

At **T3:'uh purple'**, low probability in the interpreter output causes a self-repair to be recognised, enforcing backtracking on the parse graph which informally operates as follows (see Hough and Purver (2012)) :

> **Self-repair:**
> IF from parsing word W the edge SE_n is insufficiently likely to be constructed from vertex S_n OR IF there is no sufficiently likely judgement $p(s : R_x)$ for $R_x \in G$
> THEN parse word W from vertex S_{n-1}. IF successful `add` a new edge to the top path, without removing any `commit`ted edges beginning at S_{n-1}; ELSE set $n = n$–1 and repeat.

This algorithm is consistent with a local model for self-repair backtracking found in corpora (Shriberg and Stolcke, 1998; Hough and Purver, 2013). As regards inference in G, upon detection of a self-repair that revokes $s : R_6$, the type judgement $s : \neg R_6$, i.e. that this is not a yellow object,

[8] The search for the meet probability is generalisable to conjunctive types by searching for the conjuncts' highest common descendant. The join probability is generalisable to the disjunctive probability of multiple types, used, albeit programatically, in Algorithm 1 for calculating a node's probability from its child nodes.

[9] While we do not give details here, simple graphical search algorithms for conjunctive and disjunctive multiple types are linear in the number of conjuncts and disjuncts, saving considerable time in comparison to the algebraic calculations of the sum and product rules for distributive lattices.

[10] In practice, ASR modules yielding partial results are less reliable than their non-incremental counterparts, but progress is being made here (Schlangen and Skantze, 2009).

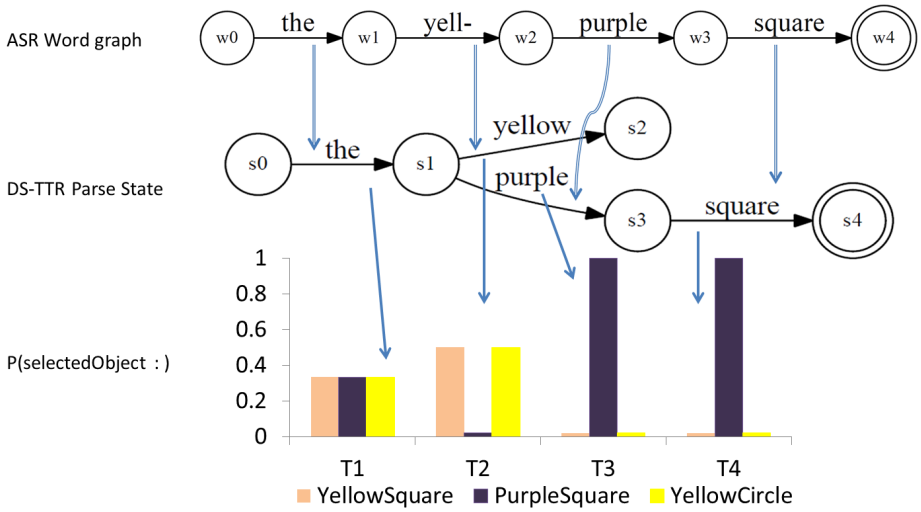

Figure 6: Incremental DS-TTR self-repair parsing. Inter-graph *groundedIn* links go top to bottom.

is immediately available as conditioning evidence. Using (6) our distribution of RT judgements now shifts: $p_J(s : R_1 \mid s : \neg R_6) = 1$, $p_J(s : R_2 \mid s : \neg R_6) = 0$ and $p_J(s : R_3 \mid s : \neg R_6) = 0$ before "purple" has been parsed – thus providing a probabilistic explanation for increased subsequent processing speed. Finally at **T4: 'square'** given $p_J(s : R_1 \mid s : R_1) = 1$ and $R_1 \wedge R_2 = R_1 \wedge R_3 = \bot$, the distribution remains unchanged.

The system's processing models how listeners reason about the revocation itself rather than predicting the outcome through positive evidence alone, in line with (Brennan and Schober, 2001)'s results.

6 Extensions

Dialogue and self-repair in the wild To move towards domain-generality, generating the lattice of all possible dialogue situations for interesting domains is computationally intractable. We intend instead to consider incrementally occurring *issues* that can be modelled as questions (Larsson, 2002). Given one or more issues manifest in the dialogue at any time, it is plausible to generate small lattices dynamically to estimate possible answers, and also assign a real-valued relevance measure to questions that can be asked to resolve the issues. We are exploring how this could be implemented using the inquiry calculus (Knuth, 2005), which defines information theoretic relevance in terms of a probabilistic question lattice, and furthermore how this could be used to model the cause of self-repair as a time critical trade-off between relevance and accuracy.

Learning in a dialogue While not our focus here, lattice G's probabilities can be updated through observations after its initial construction. If a reference game is played over several rounds, the choice of referring expression can change based on mutually salient functions from words to situations- see e.g. (DeVault and Stone, 2009). Our currently frequentist approach to learning is: given an observation of an existing RT R_i is made with probability v, then $\|R_i\|_J$, the overall denominator $P(J)$, and the nodes in the upward path from R_i to $\top$ are incremented by v. The approach could be converted to Bayesian update learning by using the prior probabilities in G for calculating v before it is added. Furthermore, observations can be added to G that include novel RTs: due to the DAG structure of G, their subtype ordering and probability effects can be integrated efficiently.

7 Conclusion

We have discussed efficient methods for constructing probabilistic TTR domain concept lattices ordered by the subtype relation and their use in incremental dialogue frameworks, demonstrating their efficacy for realistic self-repair processing. We wish to explore inclusion of join types, the scalability of RT lattices to other domains and their learning capacity in future work.

Acknowledgements

We thank the two TTNLS reviewers for their comments. Purver is supported in part by the European Community's Seventh Framework Programme under grant agreement no 611733 (ConCreTe).

References

G. Betarte and A. Tasistro. 1998. Extension of Martin-Löf type theory with record types and subtyping. In G. Sambin and J. Smith, editors, *25 Years of Constructive Type Theory*. Oxford University Press.

S. Brennan and M. Schober. 2001. How listeners compensate for disfluencies in spontaneous speech. *Journal of Memory and Language*, 44(2):274–296.

R. Cooper, S. Dobnik, S. Lappin, and S. Larsson. 2014. A probabilistic rich type theory for semantic interpretation. In *Proceedings of the EACL Workshop on Type Theory and Natural Language Semantics (TTNLS)*.

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

R. Cooper. 2012. Type theory and semantics in flux. In R. Kempson, N. Asher, and T. Fernando, editors, *Handbook of the Philosophy of Science*, volume 14: Philosophy of Linguistics, pages 271–323. North Holland.

R. Dale and E. Reiter. 1995. Computational interpretations of the gricean maxims in the generation of referring expressions. *Cognitive Science*, 19(2):233–263.

D. DeVault and M. Stone. 2009. Learning to interpret utterances using dialogue history. In *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*.

S. Dobnik, R. Cooper, and S. Larsson. 2012. Modelling language, action, and perception in type theory with records. In *Proceedings of the 7th International Workshop on Constraint Solving and Language Processing (CSLP12)*.

A. Eshghi, J. Hough, and M. Purver. 2013. Incremental grammar induction from child-directed dialogue utterances. In *Proceedings of the 4th Annual Workshop on Cognitive Modeling and Computational Linguistics (CMCL)*.

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

M. C. Frank and N. D. Goodman. 2012. Predicting pragmatic reasoning in language games. *Science*, 336(6084):998–998.

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

J. Hough and M. Purver. 2012. Processing self-repairs in an incremental type-theoretic dialogue system. In *Proceedings of the 16th SemDial Workshop on the Semantics and Pragmatics of Dialogue (SeineDial)*.

J. Hough and M. Purver. 2013. Modelling expectation in the self-repair processing of annotat-, um, listeners. In *Proceedings of the 17th SemDial Workshop on the Semantics and Pragmatics of Dialogue (DialDam)*.

R. Kempson, W. Meyer-Viol, and D. Gabbay. 2001. *Dynamic Syntax: The Flow of Language Understanding*. Blackwell.

K. H. Knuth. 2005. Lattice duality: The origin of probability and entropy. *Neurocomputing*, 67:245–274.

E. Krahmer and K. Van Deemter. 2012. Computational generation of referring expressions: A survey. *Computational Linguistics*, 38(1):173–218.

S. Larsson. 2002. *Issue-based Dialogue Management*. Ph.D. thesis, Göteborg University. Also published as Gothenburg Monographs in Linguistics 21.

S. Larsson. 2010. Accommodating innovative meaning in dialogue. *Proc. of Londial, SemDial Workshop*, pages 83–90.

S. Larsson. 2011. The TTR perceptron: Dynamic perceptual meanings and semantic coordination. In *Proceedings of the 15th Workshop on the Semantics and Pragmatics of Dialogue (SemDial 2011 - Los Angelogue)*.

W. Levelt. 1989. *Speaking: From Intention to Articulation*. MIT Press.

R. Montague. 1974. *Formal Philosophy: Selected Papers of Richard Montague*. Yale University Press.

M. Purver, A. Eshghi, and J. Hough. 2011. Incremental semantic construction in a dialogue system. In J. Bos and S. Pulman, editors, *Proceedings of the 9th International Conference on Computational Semantics*.

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

D. Schlangen and G. Skantze. 2009. A general, abstract model of incremental dialogue processing. In *Proceedings of the 12th Conference of the European Chapter of the ACL (EACL 2009)*.

E. Shriberg and A. Stolcke. 1998. How far do speakers back up in repairs? A quantitative model. In *Proceedings of the International Conference on Spoken Language Processing*.

G. Skantze and A. Hjalmarsson. 2010. Towards incremental speech generation in dialogue systems. In *Proceedings of the SIGDIAL 2010 Conference*.

J. Williams and S. Young. 2007. Scaling POMDPs for spoken dialog management. *IEEE Transactions on Audio, Speech, and Language Processing*, 15(7):2116–2129.