Incremental Semantic Construction in a Dialogue System*

Matthew Purver, Arash Eshghi, Julian Hough
Interaction, Media and Communication
School of Electronic Engineering and Computer Science, Queen Mary University of London
{mpurver, arash, jhough}@eecs.qmul.ac.uk

Abstract

This paper describes recent work on the DynDial project* towards incremental semantic interpretation in dialogue. We outline our domain-general grammar-based approach, using a variant of Dynamic Syntax integrated with Type Theory with Records and a Davidsonian event-based semantics. We describe a Java-based implementation of the parser, used within the Jindigo framework to produce an incremental dialogue system capable of handling inherently incremental phenomena such as split utterances, adjuncts, and mid-sentence clarification requests or backchannels.

1 Introduction

Many dialogue phenomena seem to motivate an incremental view of language processing: for example, a participant's ability to change hearer/speaker role mid-sentence to produce or interpret backchannels, or complete or continue an utterance (see e.g. Yngve, 1970; Lerner, 2004, amongst many others). Much recent research in dialogue systems has pursued this line, resulting in frameworks for incremental dialogue processing (Schlangen and Skantze, 2009) and systems capable of mid-utterance backchannels (Skantze and Schlangen, 2009) or utterance completions (DeVault et al., 2009; Buß et al., 2010).

However, to date there has been little focus on semantics, with the systems produced either operating in domains in which semantic representation is not required (Skantze and Schlangen, 2009), or using variants of domain-specific canned lexical or phrasal matching (Buß et al., 2010). Our intention is to extend this work to finer-grained and more domain-general notions of grammar and semantics, by using an incremental grammatical framework, Dynamic Syntax (DS, Kempson et al., 2001) together with the structured semantic representation provided by Type Theory with Records (TTR, see e.g. Cooper, 2005).

	A: I want to go to …		A: I want to go to Paris …		A: I want to go to Paris.
(a)	B: Uh-huh	(b)	B: Uh-huh	(c)	B: OK. When do you …
	A: … Paris by train.		A: … by train.		A: By train.

Figure 1: Examples of motivating incremental dialogue phenomena

One aim is to deal with split utterances, both when the antecedent is inherently incomplete (see Figure 1(a)) and potentially complete (even if not intended as such – Figure 1(b)). This involves producing representations which are as complete as possible – i.e contain all structural and semantic information so far conveyed – on a word-by-word basis, so that in the event of an interruption or a hesitation, the system can act accordingly (by producing backchannels or contentful responses as above); but that can be further incremented in the event of a continuation by the user.

Importantly, this ability should be available not only when an initial contribution is intended and/or treated as incomplete (as in Figure 1(b)), but also when it is in fact complete, but is still subsequently extended (Figure 1(c)). Treating A's two utterances as distinct, with separate semantic representations, must require high-level processes of ellipsis reconstruction to interpret the final fragment – for example, treating it as the answer to an implicit question raised by A's initial sentence (Fernández et al., 2004). If,

*The authors were supported by the Dynamics of Conversational Dialogue project (DynDial – ESRC-RES-062-23-0962). We thank Shalom Lappin, Tim Fernando, Yo Sato, our project colleagues and the anonymous reviewers for helpful comments.

instead, we can treat such fragments as continuations which merely add directly to the existing representation, the task is made easier and the relevance of the two utterances to each other becomes explicit.

2 Dynamic Syntax (DS) and Type Theory with Records (TTR)

Our approach is a grammar-based one, as our interest is in using domain-general techniques that are capable of fine-grained semantic representation. Dynamic Syntax (DS) provides an inherently incremental grammatical framework which dispenses with an independent level of syntax, instead expressing grammaticality via constraints on the word-by-word monotonic growth of semantic structures. In DS's original form, these structures are trees with nodes corresponding to terms in the lambda calculus; nodes are decorated with labels expressing their semantic type and formula, and beta-reduction determines the type and formula at a mother node from those at its daughters (Figure 2(a)). Trees can be *partial*, with nodes decorated with requirements for future development; lexical actions (corresponding to words) and computational actions (general capabilities) are defined as operations on trees which satisfy and/or add requirements; and grammaticality of a word sequence is then defined as satisfaction of all requirements (tree *completeness*) via the application of its associated actions – see Kempson et al. (2001) for details.

Previous work in DS has shown how this allows a treatment of split utterances and non-sentential fragments (e.g. clarifications) as extensions of the semantic trees so far constructed, either directly or via the addition of "linked" trees (Purver and Kempson, 2004; Gargett et al., 2009).

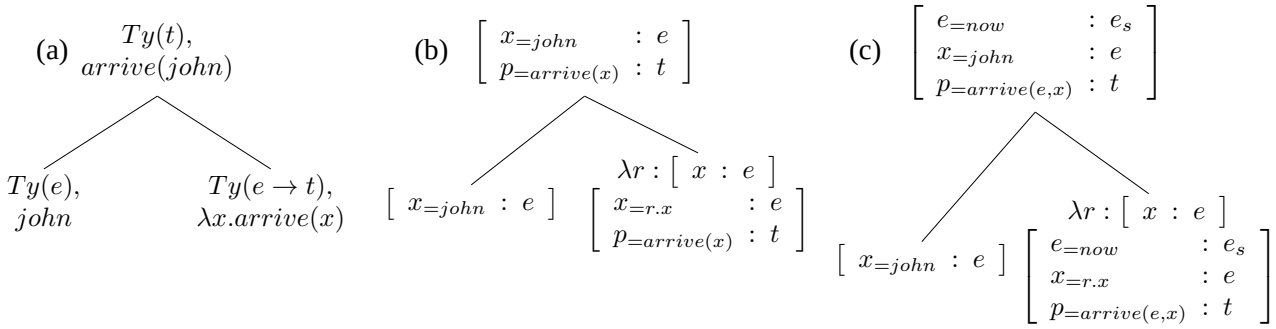

Figure 2: A simple DS tree for *"john arrives"*: (a) original DS, (b) DS+TTR, (c) event-based

2.1 Extensions

More recent work in DS has started to explore the use of TTR to extend the formalism, replacing the atomic semantic type and FOL formula node labels with more complex *record types*, and thus providing a more structured semantic representation. Purver et al. (2010) provide a sketch of one way to achieve this and explain how it can be used to incorporate pragmatic information such as participant reference and illocutionary force. As shown in Figure 2(b) above, we use a slightly different variant here: node record types are sequences of typed labels (e.g. $[x : e]$ for a label x of type e), with semantic content expressed by use of *manifest* types (e.g. $[x_{=john} : e]$ where $john$ is a singleton subtype of e).

We further adopt an event-based semantics along Davidsonian lines (Davidson, 1980). As shown in Figure 2(c), we include an event term (of type e_s) in the representation: this allows tense and aspect to be expressed (although Figure 2(c) shows only a simplified version using the current time now). It also permits a straightforward analysis of optional adjuncts as extensions of an existing semantic representation; extensions which predicate over the event term already in the representation. Adding fields to a record type results in a more fully specified record type which is still a subtype of the original:

$$\begin{bmatrix} e_{=now} & : & e_s \\ x_{=john} & : & e \\ p_{=arrive(e,x)} & : & t \end{bmatrix} \quad \mapsto \quad \begin{bmatrix} e_{=now} & : & e_s \\ x_{=john} & : & e \\ p_{=arrive(e,x)} & : & t \\ p'_{=today(e)} & : & t \end{bmatrix}$$

"john arrives" $\mapsto$ *"john arrives today"*

Figure 3: Optional adjuncts as leading to TTR subtypes

3 Implementation

The resulting framework has been implemented in Java, following the formal details of DS as per (Kempson et al., 2001; Cann et al., 2005, *inter alia*). This implementation, DyLan,[1] includes a parser and generator for English, which take as input a set of computational actions, a lexicon and a set of lexical actions (instructions for partial tree update); these are specified separately in text files in the IF-THEN-ELSE procedural (meta-)language of DS, allowing any pre-written grammar to be loaded. Widening or changing its coverage, i.e. extending the system with new analyses of various linguistic phenomena, thus do not involve modification or extension of the Java program, but only the lexicon and action specifications. The current coverage includes a small lexicon, but a broad range of structures: complementation, relative clauses, adjuncts, tense, pronominal and ellipsis construal, all in interaction with quantification.

3.1 The parsing process

Given a sequence of words $(w_1, w_2, ..., w_n)$, the parser starts from the *axiom* tree T_0 (a requirement to construct a complete tree of type t), and applies the corresponding lexical actions $(a_1, a_2, \ldots, a_n)$, optionally interspersing general computational actions (which can apply whenever their preconditions are met). More precisely: we define the parser state at step i as a set of partial trees S_i. Beginning with the singleton axiom state $S_0 = \{T_0\}$, for each word w_i:

1. Apply all lexical actions a_i corresponding to w_i to each partial tree in S_{i-1}. For each application that succeeds (i.e. the tree satisfies the action preconditions), add resulting (partial) tree to S_i.
2. For each tree in S_i, apply all possible sequences of computational actions and add the result to S_i.

If at any stage the state S_i is empty, the parse has failed and the string is deemed ungrammatical. If the final state S_n contains a complete tree (all requirements satisfied), the string is grammatical and its root node will provide the full sentence semantics; partial trees provide only partial semantic specifications.[2]

3.2 Graph representations

Sato (2010) shows how this procedure can be modelled as a *directed acyclic graph*, rooted at T_0, with individual partial trees as nodes, connected by edges representing single actions. While Sato uses this to model the search process, we exploit it (in a slightly modified form) to represent the linguistic *context* available during the parse – important in DS for ellipsis and pronominal construal. Details are given in (Cann et al., 2007; Gargett et al., 2009), but three general mechanisms are available: 1) copying formulae from some *tree* in context (used for e.g. anaphora and strict VP ellipsis); 2) rerunning *actions* in context (for e.g. sloppy VP-ellipsis and fragment corrections); and 3) directly extending/augmenting the current tree (used for most fragment types in (Fernández, 2006)). For any partial tree, then, the context available to the parser must include not only the tree itself, but the sequence of actions and previous partial trees which have gone into its construction. The parse graph (which we call the *tree* graph) provides exactly this information, via the shortest path back to the root from the current node.

However, we can also take a coarser-grained view via a graph which we term the *state* graph; here, nodes are states S_i and edges the sets of action sequences connecting them. This subsumes the tree graph, with state nodes containing possibly many tree-graph nodes; and here, nodes have multiple outgoing edges only when multiple word hypotheses are present. This corresponds directly to the input word graph (often called a word *lattice*) available from a speech recognizer, allowing close integration in a dialogue system – see below. We also see this as a suitable structure with which to begin to model phenomena such as hesitation and self-repair: as edges are linear action sequences, intended to correspond to the time-linear psycholinguistic processing steps involved, such phenomena may be analysed as building further edges from suitable departure points earlier in the graph.[3]

[1] DyLan is short for **Dy**namics of **Lan**guage. Available from `http://dylan.sourceforge.net/`.

[2] Note that only a subset of possible computational actions can apply to any given tree; together with a set of heuristics on possible application order, and the merging of identical trees produced by different sequences, this helps reduce complexity.

[3] There are similarities to chart parsing here: the tree graph edges spanning a state graph edge could be seen as corresponding to chart edges spanning a substring, with the tree nodes in the state S_i as the agenda. However, the lack of a notion of syntactic constituency means no direct equivalent for the active/passive edge distinction; a detailed comparison is still to be carried out.

4 Dialogue System

The DyLan parser has now been integrated into a working dialogue system by implementation as an `Interpreter` module in the Java-based incremental dialogue framework Jindigo (Skantze and Hjalmarsson, 2010). Jindigo follows Schlangen and Skantze (2009)'s abstract architecture specification and is specifically designed to handle units smaller than fully sentential utterances; one of its specific implementations is a travel agent system, and our module integrates semantic interpretation into this.

As set out by Schlangen and Skantze (2009)'s specification, our `Interpreter`'s essential components are a *left buffer* (LB), *processor* and *right buffer* (RB). *Incremental units* (IUs) of various types are posted from the RB of one module to the LB of another; for our module, the LB-IUs are ASR word hypotheses, and after processing, domain-level concept frames are posted as RB-IUs for further processing by a downstream dialogue manager. The input IUs are provided as updates to a word lattice, and new edges are passed to the DyLan parser which produces a state graph as described above in 3.1 and 3.2: new nodes are new possible parse states, with new edges the sets of DS actions which have created them. These state nodes are then used to create Jindigo domain concept frames by matching against the TTR record types available (see below), and these are posted to the RB as updates to the state graph (*lattice updates* in Jindigo's terminology).

Crucial in Schlangen and Skantze (2009)'s model is the notion of *commitment*: IUs are hypotheses which can be revoked at any time until they are *committed* by the module which produces them. Our module hypothesizes both parse states and associated domain concepts (although only the latter are outputs); these are committed when their originating word hypotheses are committed (by ASR) and a type-complete subtree is available; other strategies are possible and are being investigated.

4.1 Mapping TTR record types to domain concepts incrementally

Our `Interpreter` module matches TTR record types to domain concept frames via a simple XML matching specification; TTR fields map to particular concepts in the domain depending on their semantic type (e.g. go events map to `Trip` concepts, and the entity of manifest type $paris$ maps to the `City[paris]` concept). As the tree and parse state graphs are maintained, incremental sub-sentential extensions can produce TTR subtypes and lead to enrichment of the associated domain concept.

User: I want to go to Paris ...

$$\left[\begin{array}{ll} e_{=\epsilon,e17,PresentState} & : e_s \\ e1_{=\epsilon,e19,FutureAccomp} & : e_s \\ x1_{=Paris} & : e \\ p2_{=to(e1,x1)} & : t \\ x_{=speaker} & : e \\ p1_{=go(e1,x)} & : t \\ p*_{=want(e,x,p1)} & : t \end{array}\right] \quad Trip(to : City[paris])$$

User: ... from London

$$\left[\begin{array}{ll} e_{=\epsilon,e17,PresentState} & : e_s \\ e1_{=\epsilon,e19,FutureAccomp} & : e_s \\ x1_{=Paris} & : e \\ p2_{=to(e1,x1)} & : t \\ x2_{=London} & : e \\ p3_{=from(e1,x2)} & : t \\ x_{=speaker} & : e \\ p1_{=go(e1,x)} & : t \\ p*_{=want(e,x,p1)} & : t \end{array}\right] \quad \begin{array}{c} Trip(from : City[london], \\ to : City[paris]) \end{array}$$

Figure 4: Incremental construction of a TTR record type over two turns

Figure 4 illustrates this process for a user continuation; the initial user utterance is parsed to produce a TTR record type, with a corresponding domain concept – a valid incremental unit to post in the RB. The subsequent user continuation *"from London"* extends the parser state graph, producing a new TTR subtype (in this case via the DS apparatus of an adjoining *linked* tree (Cann et al., 2005)), and a more

fully specified concept (with a further argument slot filled) as output.

System behaviour between these two user contributions will depend on the committed status of the input, and perhaps some independent prosody-based judgement of whether a turn is finished (Skantze and Schlangen, 2009). An uncommitted input might be responded to with a backchannel (Yngve, 1970); commitment might lead to the system beginning processing and starting to respond more substantively. However, in either case, the maintenance of the parse state graph allows the user continuation to be treated as extending a parse tree, subtyping the TTR record type, and finally mapping to a fully satisfied domain concept frame that can be committed.

5 Conclusions

We have implemented an extension of the Dynamic Syntax framework, integrated with Type Theory with Records, which provides structured semantic representations suitable for use in a dialogue system, and which does so incrementally, producing well-defined partial representations on a word-by-word basis. This has been integrated into a working Jindigo dialogue system, capable of incremental behaviour such as mid-sentence backchannels and utterance continuations, which will be demonstrated at the conference. The coverage of the parser is currently limited, but work is in progress to widen it; the possibility of using grammar induction to learn lexical actions from real corpora is also being considered for future projects. We are also actively pursuing possbilities for tighter integration of TTR and DS, with the aim of unifying syntactic and semantic incremental construction.

References

Buß, O., T. Baumann, and D. Schlangen (2010). Collaborating on utterances with a spoken dialogue system using an ISU-based approach to incremental dialogue management. In *Proceedings of the SIGDIAL 2010 Conference*.

Cann, R., R. Kempson, and L. Marten (2005). *The Dynamics of Language*. Oxford: Elsevier.

Cann, R., R. Kempson, and M. Purver (2007). Context and well-formedness: the dynamics of ellipsis. *Research on Language and Computation 5*(3), 333–358.

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

Davidson, D. (1980). *Essays on Actions and Events*. Oxford, UK: Clarendon Press.

DeVault, D., K. Sagae, and D. Traum (2009). Can I finish? learning when to respond to incremental interpretation results in interactive dialogue. In *Proceedings of the SIGDIAL 2009 Conference*.

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

Fernández, R., J. Ginzburg, H. Gregory, and S. Lappin (2004). SHARDS: Fragment resolution in dialogue. In H. Bunt and R. Muskens (Eds.), *Computing Meaning*, Volume 3. Kluwer Academic Publishers. To appear.

Gargett, A., E. Gregoromichelaki, R. Kempson, M. Purver, and Y. Sato (2009). Grammar resources for modelling dialogue dynamically. *Cognitive Neurodynamics 3*(4), 347–363.

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

Lerner, G. H. (2004). Collaborative turn sequences. In *Conversation analysis: Studies from the first generation*, pp. 225–256. John Benjamins.

Purver, M., E. Gregoromichelaki, W. Meyer-Viol, and R. Cann (2010). Splitting the 'I's and crossing the 'You's: Context, speech acts and grammar. In *Aspects of Semantics and Pragmatics of Dialogue. SemDial 2010, 14th Workshop on the Semantics and Pragmatics of Dialogue*.

Purver, M. and R. Kempson (2004). Incremental context-based generation for dialogue. In *Proceedings of the 3rd International Conference on Natural Language Generation (INLG04)*.

Sato, Y. (2010). Local ambiguity, search strategies and parsing in Dynamic Syntax. In E. Gregoromichelaki, R. Kempson, and C. Howes (Eds.), *The Dynamics of Lexical Interfaces*. CSLI. to appear.

Schlangen, D. 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)*.

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

Skantze, G. and D. Schlangen (2009). Incremental dialogue processing in a micro-domain. In *Proceedings of the 12th Conference of the European Chapter of the ACL (EACL 2009)*.

Yngve, V. H. (1970). On getting a word in edgewise. In *Papers from the 6th regional meeting of the Chicago Linguistic Society*.