Processing Self-Repairs in an Incremental Type-Theoretic Dialogue System*

Julian Hough **Matthew Purver**
Interaction, Media and Communication group
School of Electronic Engineering and Computer Science
Queen Mary University of London
{julian.hough, matthew.purver}@eecs.qmul.ac.uk

Abstract

We present a novel incremental approach to modelling self-repair phenomena in dialogue, using the grammar and parsing mechanism of Dynamic Syntax (DS) to construct Type Theory with Records (TTR) record type representations incrementally in both parsing and generation. We demonstrate how a DS-TTR hybrid implementation when integrated into an incremental dialogue system can be exploited to account for the semantic processing of self-repair phenomena in a unified way and in line with psycholinguistic evidence.

1 Introduction

Self-repairs are too pervasive in human dialogue to be considered exceptional and they should be integral to any dialogue model, an insight from early Conversational Analysis work which revealed them to be systematic (Schegloff et al., 1977). The following are typical of the within-turn (first position), self-initiated type of self-repair often found in natural dialogue:

(1) Our situation is just [a little bit, + kind of the opposite] of that *(Switchboard)*

(2) [the interview was {...} + it was] alright *(Clark, 1996, p.266)*

(3) John goes to Paris [{uhh}+ from London] *(constructed example)*

*This work was partly supported by the RISER project EPSRC reference EP-J010383-1 and an EPSRC Doctoral Training Account scholarship for the first author. Thanks to the Semdial reviewers for their helpful comments.

For terminological and annotation purposes, following the disfluency-tagged Switchboard corpus, first position self-repairs will be discussed with reference to a division into a *reparandum* (the speech that is repaired, up to the repair point +), a possibly null *interregnum* (the filler words or pause between {}) and the following *repair* (the strings after the repair point + up to the closing square bracket). We also consider *extensions*- also called 'covert' (Levelt, 1989) or 'forward-looking' (Ginzburg et al., 2007) repairs- such as (3) which may not in fact function to alter the previous part of the utterance, but to extend it.

The formal model we describe here attempts to address two principal aspects of self-repair phenomena: firstly, in terms of cognitive processing, the semantic parsing and generation of self-repaired utterances is just as straightforward as for fluent utterances in dialogue (and in fact, in some domains semantic processing is aided (Brennan and Schober, 2001)); secondly, that the repaired material (reparandum) can be referred to in context, as in (2) above where the reparandum still needs to be accessed for the anaphoric use of 'it' to succeed, "leaving the incriminated material with a special status, but within the discourse context" (Ginzburg et al., 2007, p. 59).

2 Related work

Work on the processing of self-repair phenomena has not generally focused on the semantics and pragmatics of ongoing dialogue. Parsing approaches have tended to implement a *parse→string-edit→reparse* pipeline, which which takes disfluent inputs

and returns 'cleaned-up' grammatical strings relative to a given grammar- this was done with a TAG transducer in the case of (Johnson and Charniak, 2004). In terms of psychological validity for dialogue the approach is questionable, as parts of an utterance cannot be removed from the hearer's perceptual record, discounting the possibility of properly processing reparanda, as in example (2) above. McKelvie (1998) introduces a more explicit disfluency rule-based syntactic account, which instead of expunging 'junk' material, exploits *aborted* syntactic categories and provides optional rules for producing cleaned-up parses. However, again under the assumption that self-repair operates as a module outside the principal grammar, no method for obtaining the semantics of a self-repair is suggested.

Self-repair has received more attention from the generation (NLG) community, particularly as incremental NLG models were initially motivated by psycholinguistics, most notably Levelt (1989)'s influential modularization of speech production into distinct *conceptualization*, *formulation* and *articulation* phases. Following this, De Smedt showed how developing the syntactic component of the formulation phase in detail could give models of lexical selection and memory limitations (De Smedt, 1991) which could trigger syntactic revision and Neumann (1998) introduced reversible incremental parsing-generation processes to implement ambiguity detection and paraphrasing corrections. In conceptualization, Guhe (2007) modelled online modifications to pre-verbal messages that cause self-repair surface forms to be realized.

Albeit less psychologically motivated, Buß and Schlangen (2011) and Skantze and Hjalmarsson (2010) introduce self-repair generation strategies in incremental dialogue systems. Both systems make use of the Incremental Unit (IU) dialogue framework model (Schlangen and Skantze, 2009), which allows online revision of input and outputs between modules. Skantze and Hjalmarsson (2010) use string-based speech plans which may change dynamically during interaction with a user, allowing for changing ASR hypotheses, which could lead to the generation of a limited set of 'covert' (non-replacement extensions) and 'overt' self-repairs. The interactional benefits of the approach are clear, however the lack of incremental semantics and domain-general grammar makes scalability to more complex domains and integration with a parsing module difficult.

In terms of the dialogue semantics of self-repair, Ginzburg and colleagues (Ginzburg et al., 2007; Ginzburg, 2012) working within the KoS framework (Ginzburg, 2012) with Dialogue Gameboard (DGB) update mechanisms at its core, attempt to unify an account of self-repair and other-initiated repair by drawing the parallels between self-initiated editing phrases (interregna) and clarification requests (CRs) as cues for repair. They make an adjustment to KoS in allowing CRs and editing signals and their following corrections to occur mid-utterance, accommodating incrementality by allowing the DGB word-by-word updates to its PENDING component. They also suggest that Type Theory with Records (TTR) could be instrumental in enabling appropriate types for word-by-word semantic updates in their future work. However, while this provides a general dialogue model, the relationship of these updates to incremental parsing and generation processes is not made explicit.

3 Criteria for a unified account

The parsing, generation and dialogue semantics implementations of self-repairs have been slightly orthogonal, so a grammar which can provide a suitable semantic representation to capture the phenomena in both modalities within a dialogue context is lacking. We suggest that two requirements of a grammar to remedy this are *strong incremental interpretation* and *incremental representation* (Milward, 1991). Strong incremental interpretation is the ability to make available the maximal amount of information possible from an unfinished utterance as it is being processed, particularly semantic dependencies (e.g. a representation such as $\lambda x.like'(john', x)$ should be available after processing "John likes"). Incremental representation, on the other hand, is defined as a representation being available for each substring of an utterance, but not necessarily including the dependencies (e.g. having a representation such as $john'$ attributed to "John" and $\lambda y.\lambda x.like'(y, x)$ attributed to "likes" after processing "John likes"). These representations should become available immediately to connected modules, therefore requiring

seamless integration with other dialogue semantics.[1]

Furthermore a record of *processing context* is required to be sufficiently detailed, and suitably structured, so that parsing, generation and dialogue management algorithms can access the material in the reparandum straightforwardly, as shown by example (2). This context must extend from the phonetic level to the conceptual level: Brennan and Schober (2001)'s experiments demonstrated self-repair can speed up semantic processing (or at least object reference) in a small visual domain of shape selection, where an incorrect object being partly vocalized and then repaired in the instructions (e.g. "the yell-purple square") yielded quicker response times from the onset of the target ("purple") than in the case of the equivalent fluent instructions (e.g. "the purple square"). This example will be addressed in section 6.

Given these requirements and the lacunae from previous work, in the remainder of the paper we present a type-theoretic incremental model of parsing, generation and context that addresses them, showing how a suitable grammar formalism and semantic representation (section 4) integrated into an incremental dialogue system (section 5) can account for parsing (section 6) and generating (section 7) self-repairs in a psycholinguistically plausible way.

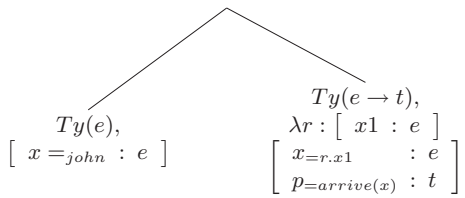
4 Dynamic Syntax and TTR

Dynamic Syntax (DS) (Kempson et al., 2001) is an action-based and semantically oriented incremental grammar framework that defines grammaticality as parsability. The DS lexicon consists of *lexical actions* keyed to words, and also a set of globally applicable *computational actions*, both of which constitute packages of monotonic update operations on semantic trees, and take the form of IF-THEN action-like structures. For example, in DS notation, the lexical action corresponding to the word *john* has the preconditions and update operations in example (4): if the pointer object ($\Diamond$), which indicates the node being checked on the tree, is currently positioned at a node that satisfies the properties of the precondition then all the actions in the post-condition can be completed, these being simple monotonic tree operations.

(4)

john:	
IF	$?Ty(e)$
THEN	put($Ty(e)$)
	put($[\ x_{=john} : e\ ]$)
ELSE	abort

(5) *"John arrived"* $\longmapsto$



In DS, the trees upon which actions operate represent terms in the typed lambda calculus, with mother-daughter node relations corresponding to semantic predicate-argument structure, with no independent layer of syntax represented. Tree nodes are typed, and can be either type-complete (e.g. $Ty(e)$) and decorated with a semantic formula, or have a requirement for a type (e.g. $?Ty(e)$). As can be seen in (5) above, recent DS variants (Purver et al., 2010) incorporate Type Theory with Records (TTR) (Cooper, 2005), with TTR *record types* decorating tree nodes, rather than simple atomic formulae.

Following Cooper (2005), each field in a record type is of the form $[\ l : T\ ]$, containing a unique label l in the record type and a type T. Fields can be *manifest*, i.e. have a singleton type such as $[\ l : T_a\ ]$ 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 be *dependent* on fields preceding them (i.e. higher up in the graphical representation), e.g. the predicate type $[\ p_{=like(x,y)} : t\ ]$, where x and y are labels in preceding fields. DS node semantic formulae are now taken to be record types, with the type of the final (i.e. lowest down) field corresponding to the $Ty()$ node type. Functions from record type to record type in the variant of TTR we use here employ paths, and are of the form $\lambda r : [\ l1 : T1\ ][\ l2_{=r.l1} : T1\ ]$, an example being the formula at the type $Ty(e \rightarrow t)$ node in tree (5) above, giving DS-TTR the required functional application capability: functor node functions are applied to their sister argument node's

[1] Recently, (Peldszus et al., 2012) show how incrementally integrating incremental syntactic and pragmatic processing can improve an interpreter module's performance.

formula, with the resulting β-reduced record type added to their mother node.[2]

In DS parsing, beginning with an axiom tree with a single node of requirement type $?Ty(t)$, parsing intersperses the testing and application of both lexical actions triggered by input words such as 4 and the execution of permissible (Kleene* iterated) sequences of computational actions, with their updates monotonically constructing the tree. Successful parses are sequences of action applications that lead to a tree which is complete (i.e. has no outstanding requirements on any node, and has type $Ty(t)$ at its root node as in (5)). The DS notion of incrementality is two-fold, in that action sequences monotonically extend the trees, and that these sequences are maximally applied on a word-by-word basis.

Here we modify the traditional DS parsing and generation model by allowing the compilation of TTR formulae for *partial* trees in addition to complete ones. This is achieved through a simple tree-compiling algorithm which decorates terminal nodes with record types containing underspecified variables of the appropriate type, then applies functional application between sister nodes to compile a β-reduced record type at their mother node, continuing in bottom-up fashion until a record type is compiled at the root (see (Hough, 2011) for details). The modification means the DS-TTR model now meets the criteria of strong incremental interpretation, as maximal record types represent all possible dependencies made available as each word is processed.

4.1 DS-TTR generation as parsing

As Purver and Kempson (2004) demonstrate, an incremental DS model of surface realization can be neatly defined in terms of the DS parsing process and a *subsumption check* against a *goal tree*. The goal tree input is a complete and fully specified DS tree such as (5), and the generation of each word consists of attempting to parse each word in the lexicon to extend the trees under construction in the parse state. Partial trees are checked for suitability via goal tree subsumption, with unsuitable trees and their parse paths removed from the generator state. The DS generation process is word-by-word incremental with maximal tree representations continually available, and it effectively combines lexical selection and linearization into a single action due to the word-by-word iteration through the lexicon. Also, self-monitoring is inherently part of the generation process, as each word generated is parsed. However, this model requires fully structured *trees* as input, problematic for a dialogue manager.

Here, though, with incremental representations now available through the tree compiling mechanism as described above, a modification can be made by replacing the goal tree with a *TTR goal concept*, which can take the form of a record type such as:

(6) $$\begin{bmatrix} x1_{=Paris} & : & e \\ x_{=john} & : & e \\ p1_{=to(x1)} & : & t \\ p_{=go(x)} & : & t \end{bmatrix}$$

Consequently, the tree subsumption check in the original DS generation model can now be characterized as a TTR subtype relation check between the goal tree and the compiled TTR formulae of the trees in the parse state. A definition for the check, adapted from Fernández (2006, p.96), is defined in (7).

(7) *Subtype relation check*:
For record types $p1$ and $p2$, $p1 \sqsubseteq p2$ holds just in case for each field $\begin{bmatrix} l & : & T2 \end{bmatrix}$ in $p2$ there is a field $\begin{bmatrix} l & : & T1 \end{bmatrix}$ in $p1$ such that $T1 \sqsubseteq T2$, i.e. iff any object of type $T1$ is also of type $T2$. This relation is reflexive and transitive.

The advantage of this move is that for the logical input to generation a goal tree no longer needs to be constructed from the grammar's actions, so the dialogue management module need not have full knowledge of the DS parsing mechanism and lexicon. An example successful generation path can be seen in Figure 1,[3] showing how the maximal TTR record type for each tree is continually available.

[2] For functional application and record type extension (concatenation), which is required in DS grammar for merging the formulae at the top of LINKed tree structures, *relabelling* is carried out when necessary in the record types in the way described by Cooper (2005) and Fernández (2006).

[3] The incremental generation of "john arrives" succeeds as the successful lexical action applications at transitions 1 ↦ 2 and 3 ↦ 4 are interspersed with applicable computational action sequences at transitions 0 ↦ 1 and 2 ↦ 3, at each stage passing the subtype relation check with the goal (i.e. the goal is a subtype of the top node's compiled record type), until arriving at a tree that *type matches* in 4.

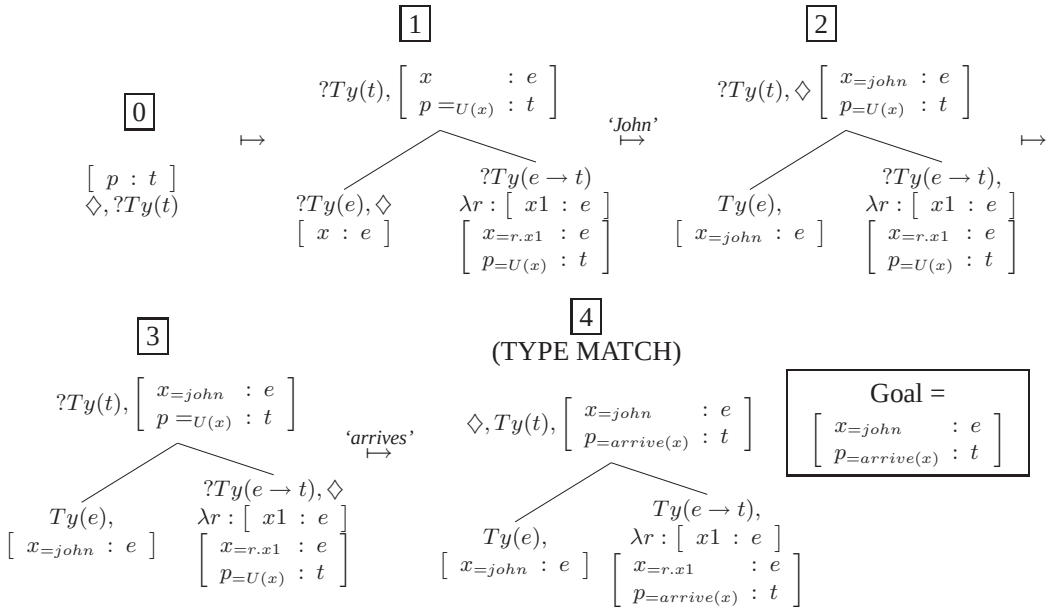


Figure 1: Successful generation path in DS-TTR

Another efficiency advantage is that subtype checking can also reduce the computational complexity of lexicalisation through pre-verbal lexical action selection, removing the need to iterate through the entire lexicon on a word-by-word basis. A sublexicon $SubLex$ can be created when a goal concept $GoalTTR$ is inputted to the generator by searching the lexicon to select lexical actions whose TTR record type is a valid supertype of $GoalTTR$.

5 Incremental DS-TTR parsing and generation in DyLan

In order to meet the criteria of a continuously updating contextual record, we implement DS-TTR parsing and generation mechanisms in the prototype `DyLan` dialogue system[4] within Jindigo (Skantze and Hjalmarsson, 2010), a Java-based implementation of the incremental unit (IU) dialogue system framework (Schlangen and Skantze, 2009). As per Schlangen and Skantze (2009)'s model, there are input and output IUs to each module, 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:

- *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 (TTR record types) constructed between vertices C_n, *groundedIn* the corresponding path of edges in the DS parsing DAG

In the generation module, the architecture is the inverse of interpretation given the input of TTR goal concepts:

- *input:* the concept graph has goal concept IU edges (TTR record types) between vertices GC_n posted by the dialogue manager

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

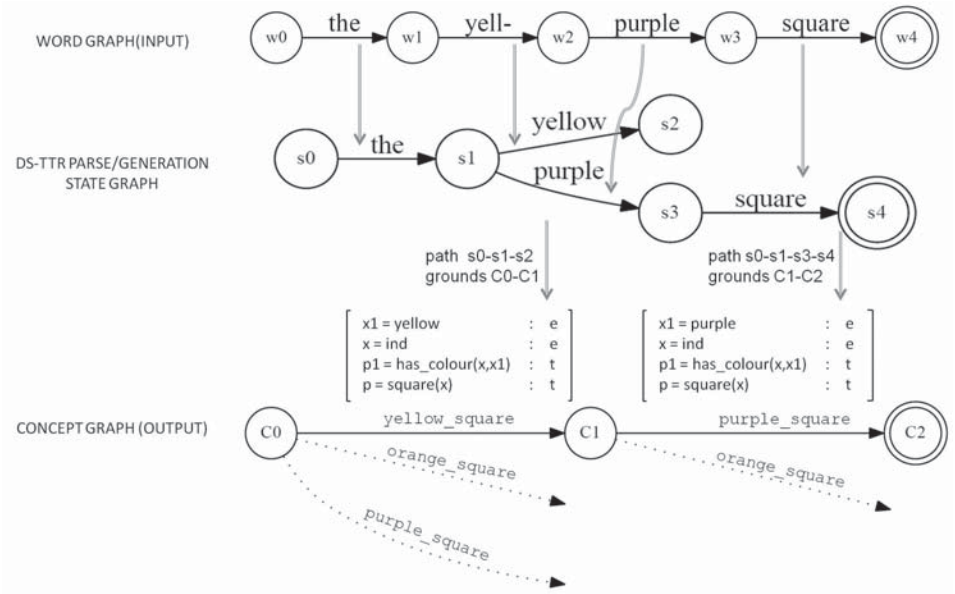


Figure 2: Incremental DS-TTR parsing of a self-repair. Revoked edges indicating failed search paths are dotted. Inter-graph *groundedIn* links go from top to bottom.

- *processing:* the DS parsing graph (shared with the interpreter module's graph) is incrementally constructed word-by-word by parsing the lexical actions in the sublexicon and subtype checking the result against the current goal concept (see section 4.1)
- *output:* the word graph's edges are `added` to the output buffer during word-by-word generation, and `committed` when they are *groundedIn* DS parsing graph paths that form part of a valid generation path *type matched* with the goal concept (as in Figure 1).

6 Parsing self-repairs

Interpretation in `DyLan` follows evidence that dialogue agents parse self-repairs efficiently and that repaired material is given special status but not removed from the discourse context. To simulate Brennan and Schober (2001)'s experimental findings described in section 3, we demonstrate a self-repair parse in Figure 2 using a domain of three domain concepts, `yellow_square`, `purple_square` and `orange_square`, each with a distinct record type. When "yell-" is processed, the word hypothesizer adds the edge 'yellow', which in turn is parsed, returning a TTR record type. Search is initiated for domain concepts in a subtype relation to it, in this case finding a valid subtype in the concept `yellow_square`- when matched it is moved from the domain concepts to the concept graph's active edge. The following failure to interpret 'purple' forces a repair under the definition in 8 below:

(8) **Repair** IF from parsing word W there is no edge SE_n able to be constructed from vertex S_n (no parse) or if no domain concept hypothesis can be made through subtype relation checking, `repair`: parse word W from vertex S_{n-1}. Should that parse be successful `add` a new edge to the top path, without removing any `committed` edges beginning at S_{n-1}.

This does not remove the initially matched concept IU at edge $C0$-$C1$, but forces another matching process to add a successor edge. The consequent subtype-checking operation is then limited to just the concepts `purple_square` and `orange_square`, finding a type match in the former. While this trivially reduces the subtype checking iteration process here for illustrative purposes, with a bigger domain this could remove many concepts (i.e. all of those that are subtypes of the incriminated parse path's current record type).

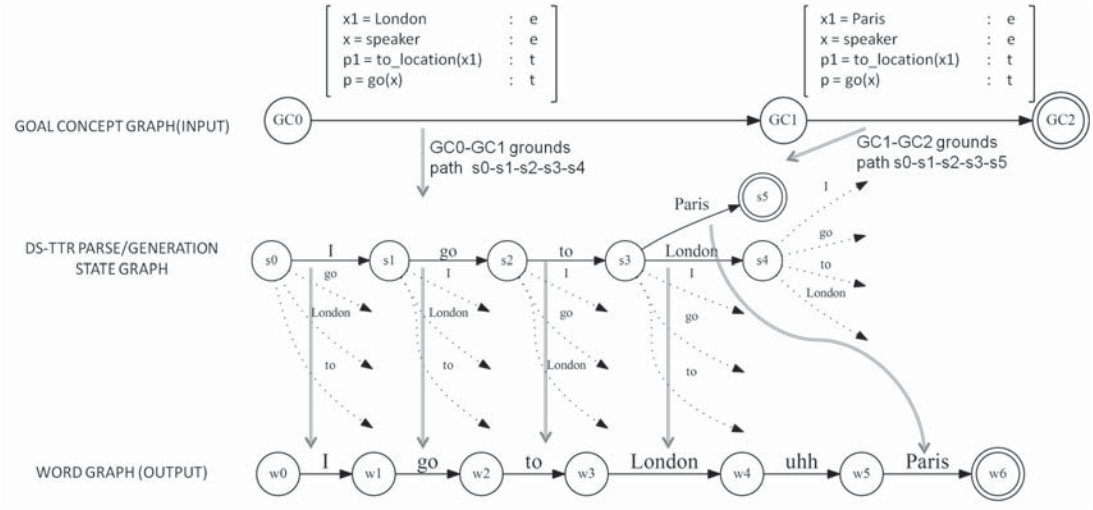


Figure 3: Incremental DS-TTR generation of a self-repair upon goal concept change. Type-matched record types are double-circled nodes. Inter-graph *groundedIn* links go from top to bottom.

This strategy will also allow the parsing of (2) "the interview was.. it was alright", with the correct reference resolution of 'it': any committed preceding edge on the word hypothesis graph can be accessed (i.e. any word/partial word heard in the user's speech stream), as can its corresponding *groundedIn* DS-TTR parse graph edge IU, so the TTR formula for 'the interview' is accessible and DS anaphora mechanisms using context may run as normal.

While the rule in (8) will only allow the parsing of replacement type self-repairs, in our prototype dialogue system this can be triggered not only by syntactic disfluency but also by pragmatic infelicity. For example, if the user were to say "I pick the yellow square or rather the blue square", which may be parsable in the DS grammar without backtracking, the mechanism will still work in the same way because in our micro-domain there is no available concept that represents the user selecting both the yellow and blue squares simultaneously in one turn. Work is also under way to lexicalise editing signals in terms of their effect on DS parsing context.

7 Generating self-repairs

In `DyLan`'s generation module, whose processing is driven by parsing as described in section 4.1, the parsing `repair` function defined in (8) will operate if there is no resulting word edge output after a generation cycle to produce the next word. This will be triggered by a change in goal concept during generation. As per parsing, in repair the generation algorithm continues backtracking by one vertex at a time in an attempt to extend the DS DAG until successful, as can be seen in Figure 3 with the successful backtrack and parse of 'Paris' resulting in successful subsumption to the new goal concept. The time-linear word graph continues to extend but with the repair's edges *groundedIn* different paths of the parse DAG to those which ground the reparandum.[5] Our protocol is consistent with Shriberg and Stolcke (1998)'s empirical observation that the probability of retracing N words back in an utterance is more likely than retracing from N+1 words back, making the repair as local as possible.

Another type of self-repair, *extension*, such as example (3) above, is dealt with straightforwardly in our generation module. For these covert repairs, the incoming goal concept must be a subtype of the one it replaces, and so the DS parser can induce monotonic growth of the matrix tree through LINK adjunction (Kempson et al., 2001), resulting in subtype extension of the root TTR record type. Thus, a change in goal concept during generation will not always put demands on the system to backtrack, such as in generating the fragment after the pause in "John goes to Paris…from London". It is only

[5]The previously committed word graph edge for 'London' is not revoked nor is its *groundedIn* parse graph edge, following our parsing algorithm and the principle that has been in the public record and hence should still be accessible.

at a semantics-syntax mismatch, where the revised goal TTR record type does not correspond to a permissible extension of a DS tree in the parsing DAG where overt repair will occur. In contrast to Skantze and Hjalmarsson (2010)'s string-based *speech plan* comparison approach, there is no need to regenerate a fully-formed string from a revised goal concept and compare it with the string generated thus far. Far from a phonetic form deletion account, self-repair in DyLan is driven by attempting to extend existing parse paths to construct the new target record type, *retaining* all the semantic representation and the procedural context of actions already built up in the generation process to avoid the computational demand of constructing semantic representations from afresh. This way a unified mechanism for modification and extension repairs is possible.

8 Conclusion and Future Work

We have presented a framework for parsing and generating word-by-word incrementally using a hybrid grammar of Dynamic Syntax and Type Theory with Records (DS-TTR) which has been implemented in the DyLan dialogue system, utilising the mechanisms of the Incremental Unit framework. DyLan provides a preliminary model of the parsing and generation of self-repair in line with psycholinguistic evidence of preference for locality and the availability of access to the semantics of repaired material.

In terms of development, while our model currently covers replacement type repairs and extensions, there is potential for expansion to insertion type repairs such as "Peter went swimming with Susan... or rather surfing, yesterday".[6] The use of a DS-TTR parsing context DAG constructed by the utterance so far could again be used to resolve these repairs, in this case by reusing preceding action edges in the spirit of the recent DS account of verb phrase ellipsis (VPE) (Kempson et al., forthcoming). Schematically, the repair mechanism would work in a similar way to the resolution of the VPE in "[Peter went] swimming [with Susan] and $_{<Ai-Aj>}$ surfing $_{<Ak-Al>}$ yesterday", where $_{<Ai-Aj>}$ is the sequence of action edges used in the construction of the DAG triggered by the words "Peter went" and $_{<Ak-Al>}$ is the sequence for "with Susan", both of which are re-used either side of the actions triggered by "surfing". The regeneration (Kempson et al., forthcoming) rule operating over DS context makes this possible here, enabling the parser (and generator) to take a sequence of actions from context and re-use them, provided that they were triggered by the same type-requirement as is imposed on the node currently under development in the tree being constructed in the DAG. The difference between the VPE and repair mechanisms would lie in the commitment status of the TTR formulae constructed by these action paths: in VPE the record types yielded from both construction paths would be committed to the dialogue manager, whilst in repair, although the reparandum's path remains accessible to the parsing and generation modules, only the repair's resultant TTR record type would be committed. This account is yet to be fully fledged out for these repairs, particularly the consideration of how editing terms would be interpreted appropriately, and the computational demands for the on-line resolution of the repair, however the potential for extending the self-repair model provided by the DS characterization of context is clear.

Other future work planned includes investigating the lexical semantic structure that TTR record types may offer for modelling type dependencies between reparanda and repairs.

[6] Many thanks to the anonymous reviewer who provided the example and highlighted the importance of this.

References

S.E. Brennan and M.F. Schober. 2001. How listeners compensate for disfluencies in spontaneous speech* 1. *Journal of Memory and Language*, 44(2):274–296.

O. Buß and D. Schlangen. 2011. Dium–an incremental dialogue manager that can produce self-corrections. In *Proceedings of the 15th Workshop on the Semantics and Pragmatics of Dialogue (SEMDIAL)*, pages 47–54, Los Angeles, California, September.

Herbert H. Clark. 1996. *Using Language*. Cambridge University Press.

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

Koenraad De Smedt. 1991. Revisions during generation using non-destructive unification. In *Proceedings of the Third European Workshop on Natural Language Generation*, pages 63–70.

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

Jonathan Ginzburg, Raquel Fernández, and David Schlangen. 2007. Unifying self- and other-repair. In *Proceedings of the 11th Workshop on the Semantics and Pragmatics of Dialogue (DECALOG)*.

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

Markus Guhe. 2007. *Incremental Conceptualization for Language Production*. NJ: Lawrence Erlbaum Associates.

Julian Hough. 2011. Incremental semantics driven natural language generation with self-repairing capability. In *RANLP 2011 Student Research Workshop*, pages 79–84, Hissar, Bulgaria.

Mark Johnson and Eugene Charniak. 2004. A tag-based noisy channel model of speech repairs. In *Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics*, ACL '04, Stroudsburg, PA, USA. Association for Computational Linguistics.

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 Semantic Theory*. 2nd edition.

W.J.M. Levelt. 1989. *Speaking: From intention to articulation*. MIT Press.

David McKelvie. 1998. The syntax of disfluency in spontaneous spoken language. In *HCRC Research Paper, HCRC/RP-95*.

David Milward. 1991. *Axiomatic Grammar, Non-Constituent Coordination and Incremental Interpretation*. Ph.D. thesis, University of Cambridge.

Günter Neumann. 1998. Interleaving natural language parsing and generation through uniform processing. *Artificial Intelligence*, 99:121–163.

Andreas Peldszus, Okko Buß, Timo Baumann, and David Schlangen. 2012. Joint satisfaction of syntactic and pragmatic constraints improves incremental spoken language understanding. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*, pages 514–523, Avignon, France, April. Association for Computational Linguistics.

Matthew Purver and Ruth Kempson. 2004. Incremental context-based generation for dialogue. In *Proceedings of the 3rd International Conference on Natural Language Generation (INLG04)*, number 3123 in Lecture Notes in Artifical Intelligence, pages 151–160, Brockenhurst, UK, July. Springer.

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 *Proceedings of the 14th Workshop on the Semantics and Pragmatics of Dialogue (SEMDIAL)*, 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 *Proceedings of the 9th International Conference on Computational Semantics*, pages 365–369, Oxford, UK, January.

Yo Sato. 2011. Local ambiguity, search strategies and parsing in Dynamic Syntax. In E. Gregoromichelaki, R. Kempson, and C. Howes, editors, *The Dynamics of Lexical Interfaces*, pages 205–233. CSLI.

Emanuel A. Schegloff, Gail Jefferson, and Harvey Sacks. 1977. The preference for self-correction in the organization of repair in conversation. *Language*, 53(2):361–382.

David Schlangen and Gabriel 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)*, pages 710–718, Athens, Greece, March. Association for Computational Linguistics.

Elizabeth Shriberg and Andreas Stolcke. 1998. How far do speakers back up in repairs? A quantitative model. In *Proceedings of the International Conference on Spoken Language Processing*, pages 2183–2186.

Gabriel Skantze and Anna Hjalmarsson. 2010. Towards incremental speech generation in dialogue systems. In *Proceedings of the SIGDIAL 2010 Conference*, pages 1–8, Tokyo, Japan, September. Association for Computational Linguistics.