Journal of Logic, Language and Information
https://doi.org/10.1007/s10849-021-09334-x

Dynamic Syntax

The Dynamics of Incremental Processing: Constraints on Underspecification

Christine Howes[1] · Hannah Gibson[2]

Accepted: 28 April 2021
© The Author(s) 2021

Abstract
Dynamic Syntax (DS: Kempson et al. 2001; Cann et al. 2005) is an action-based grammar formalism which models the process of natural language understanding as monotonic tree growth. This paper presents an introduction to the notions of incrementality and underspecification and update, drawing on the assumptions made by DS. It lays out the tools of the theoretical framework that are necessary to understand the accounts developed in the other contributions to the Special Issue. It also represents an up-to-date account of the framework, combining the developments that have previously remained distributed in a diverse body of literature.

Keywords Dynamic Syntax · Incrementality · Underspecification

1 Introduction

Dynamic Syntax (DS: Kempson et al. 2001; Cann et al. 2005) is an action-based grammar formalism which models the process of natural language understanding as monotonic tree growth. The foundations of DS are based in the recognition of the fact that what are usually considered independent features of language; syntax, semantics and pragmatics, are in fact mutually dependent features of human communication.

✉ Christine Howes
christine.howes@gu.se

Hannah Gibson
h.gibson@essex.ac.uk

[1] Department of Philosophy, Linguistics and Theory of Science, University of Gothenburg, Gothenburg, Sweden

[2] Department of Language and Linguistics, University of Essex, Colchester, UK

Published online: 28 May 2021

Words are analysed in the order in which they are uttered in a string in a context, thus taking how an interpretation is built up to have a central role, including factors usually considered to be outside the remit of a grammar, such as pragmatic enrichment from context. Complete trees in DS have no representation of word order, instead they are binary branching semantic trees representing the argument structure of the utterance being parsed or produced. DS postulates that phenomena usually described as syntactic can be explained and described by the dynamics of the growth of a (potentially partial) parse tree.

This paper constitutes an overview of the Dynamic Syntax framework and an introduction to the current Special Issue which explores the notion of parsing/production incrementality, drawing particularly on the concepts of underspecification and update. The theoretical notions upon which the framework depends are outlined below, with the formal tools laid out in more detail in Sect. 2. Section 3 provides an introduction to the papers of the Special Issue.

1.1 Incrementality

In DS, trees are built up incrementally on a word-by-word basis via a combination of generally applicable computational rules, specific lexical actions and pragmatic actions. Any string encountered is grammatical if there is a licit sequence of steps which can lead to a completed tree (with no unresolved requirements—see Sect. 2.1) by the time all the words in the string have been parsed. Importantly (particularly for analyses of dialogue) at any point in an ongoing parse we can also distinguish potentially grammatical sequences from ungrammatical ones (Cann et al. 2007). In the case of potentially grammatical sequences, there is a successful sequence of steps up to a given point, although the tree is not yet complete and there may be outstanding requirements. In contrast, if there is no possible sequence of steps which allows development up to a given point, the sequence is ungrammatical and the process aborted.

Crucially then, a grammatical string from the DS perspective depends on the ability for one partial tree to unfold into another partial tree. This contrasts with other theoretical approaches in which the 'final' tree state is the determining factor in grammaticality. Another key feature of the Dynamic Syntax approach is that alternative parsing routes may be possible for any given sentence, and this will not be recoverable from the final tree, only by looking at the actions taken to reach it.

1.2 Underspecification

Key to the way that interpretations are built up in DS is the notion of underspecification. This can be structural, in the form of unfixed nodes (see Sect. 2.4.1), semantic e.g. in anaphora where pronouns are taken to project place-holding metavariables (see Sect. 2.4.3) or pragmatic, where e.g. interpretations may need to be resolved by access to non-linguistic context. Underspecification is a key concept in the DS formalism and is the feature which unites the contributions in this Special Issue.

2 Mechanisms

Many grammar formalisms characterise properties of strings of words. In DS, what is ultimately characterised is propositional structure. This propositional structure is represented using binary logical trees. The establishment of propositional structure is further built up step-by-step on a left-to-right basis and is modelled as tree growth, and this is taken to constitute syntax. Knowledge of syntax is therefore characterised as the knowledge to construct semantic representations from words encountered in context. DS is crucially concerned with characterising the growth of such representations through strings uttered in context.

We next describe the formal tools used in DS, then take the reader step-by-step through a simple parse.

2.1 Trees

Decorations Propositions are ultimately modelled using (binary) semantic trees. Each tree node is annotated with decorations which provide different types of information. Tree nodes must contain information in terms of both type and formula values.[1]

1. The type of node, e.g. propositions $Ty(t)$ and entities $Ty(e)$—with complex types built up from the basic types, e.g., intransitive verb phrases are functions from entities to truth values $Ty(e \rightarrow t)$ etc;
2. A formula value (the semantic content)—these are traditionally expressed in terms such as $Fo(John')$, which, by the rules of the grammar can be combined to form complex expressions such as $Fo(Love'(Mary')(John'))$, where X' corresponds to the concept which X expresses. Current formulations of DS (see e.g. Eshghi et al. 2015, a.o.) use records and record types from type theory with records (TTR; Cooper 2005) or distributional semantic vectors for the semantic content (Sadrzadeh et al. 2018). While there is a stipulation that each node has a semantic value, DS itself is agnostic as to the precise semantic representation.
3. The tree node address—using the logic of finite trees (LOFT; Blackburn and Meyer-Viol 1994) this is either based on the root node of the tree under construction being $Tn(0)$, with each daughter node being assigned an additional one for a functor daughter or zero for an argument daughter, or by its relation to any other tree node. For example, $\langle\downarrow_0\rangle X$ means that X holds at the current node's argument daughter, whilst $\langle\downarrow_*\rangle X$ means X holds somewhere below or at the current node in the local tree. By convention predicates appear on the right-branching nodes and arguments appear on left-branching nodes.
4. The pointer ($\diamond$) marks the node currently under development.

Requirements All these labels (except the pointer) can show information about what has already been parsed, or be requirements, indicating what else is needed to complete the current partial tree. Unlike complete descriptive decorations, requirements are preceded by a question mark(?); $?Ty(t)$ is a requirement for a proposition, $?\exists x.Tn(x)$ is a requirement for a fixed tree node address. Crucially, this is only one way in

[1] Note that decorations are often suppressed in trees in the interests of readability.

which underspecification is modelled in DS. For a successful parse, there must be no outstanding requirements on the tree by the time the parse is complete.[2]

2.2 Tree Update

There are a finite set of tree actions that can be applied in DS. These include actions for inducing tree growth (`make()`); decorating tree nodes (`put()`); and moving the pointer within an existing tree structure (`go()`). These procedures are applied in the parse in generally applicable computational actions, or triggered by words.

Lexical actions Like Head-driven Phrase Structure Grammar (HPSG; see e.g. Sag et al. 2003) and Combinatory Categorial Grammar (CCG; see e.g. Steedman 2000), DS is a lexicalised grammatical framework, acknowledging the fact that complexity in language relies to a large extent on information stored in the lexicon, and is therefore language specific. In DS, what is stored in the lexicon is a set of procedures, known as lexical actions.

The lexical action for 'John', shown in Example 1, will be accessed when the word John is encountered, and states: if there is a requirement for $Ty(e)$ at the current node (determined by the position of the pointer $\Diamond$), then put the decorations for the type $Ty(e)$, and formula $Fo(John')$ at the current node, otherwise abort the parsing process.[3]

Example 1

$$\textit{John} \left| \begin{array}{ll} \text{IF} & ?Ty(e) \\ \text{THEN} & \texttt{put}(Ty(e), Fo(John'), [\downarrow]\bot) \\ \text{ELSE} & \text{ABORT} \end{array} \right.$$

Lexical actions can also account for some aspects of syntactic predictability. For example, in the lexical entry for the transitive verb loves, nodes for the verb and its object are created with the pointer left at the object node with the expectation that the next lexical item will be of $Ty(e)$.[4]

Example 2

$$\textit{loves} \left| \begin{array}{ll} \text{IF} & ?Ty(e \rightarrow t) \\ \text{THEN} & \texttt{go}(\langle\uparrow_1\rangle); \texttt{put}(PRES); \texttt{go}(\langle\downarrow_1\rangle); \texttt{make}(\langle\downarrow_1\rangle); \texttt{go}(\langle\downarrow_1\rangle); \\ & \texttt{put}(Ty(e \rightarrow (e \rightarrow t)), Fo(Love'), [\downarrow]\bot); \\ & \texttt{go}(\langle\uparrow_1\rangle); \texttt{make}(\langle\downarrow_0\rangle); \texttt{go}(\langle\downarrow_0\rangle); \\ & \texttt{put}(?Ty(e)) \\ \text{ELSE} & \text{ABORT} \end{array} \right.$$

[2] Though recall the distinction between potential grammaticality versus ungrammaticality discussed in Sect. 1.1.

[3] The last decoration in this lexical entry $[\downarrow]\bot$, is the bottom restriction, which means that the node may not have daughter nodes.

[4] Note that the decoration `put`(PRES) is a shorthand notation for tense information. This has been more formally represented using the notion of "situation nodes" and "event variables" (see Gregoromichelaki 2006, a.o.). We do not go into the details here.

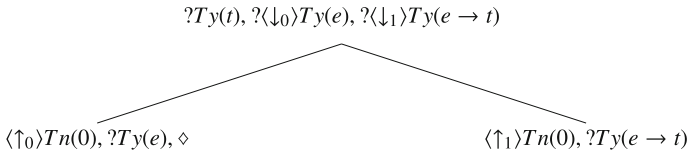
$?Ty(t), ?\langle\downarrow_0\rangle Ty(e), ?\langle\downarrow_1\rangle Ty(e \rightarrow t)$
$\langle\uparrow_0\rangle Tn(0), ?Ty(e), \diamond$
$\langle\uparrow_1\rangle Tn(0), ?Ty(e \rightarrow t)$

Fig. 1 Tree following Introduction and Prediction

Computational actions Computational rules apply optionally whenever their preconditions are met (DS does not give us a strategy for choosing a rule at any given point, although of course there are restrictions on when they can apply). These form a small fixed set of macros, which can be expressed in the same way as lexical actions. Some encode the properties of LOFT, the Language of Finite Trees (such as using Anticipation to move the pointer to a node with unsatisfied requirements, or tree growth using e.g. *Adjunction as in Fig. 5); others encode semantic actions (e.g. β-reduction; see Example 3). Broadly speaking, these computational rules are considered to be available cross-linguistically and can apply whenever the necessary triggering conditions are present. This contrasts with lexical input, for example, which is language-specific.

Pragmatic actions In addition to the mechanisms of computational and lexical actions, in order to complete a parse it may be necessary to perform pragmatic actions, such as finding an appropriate antecedent for a metavariable from context (see the case of pronouns discussed in Sect. 2.4.3, where successful processing involves a pragmatic process of Substitution). This may mean incorporating non-linguistic information, such as when he refers to someone being gestured towards, and the machinery of DS itself does not provide constraints on this process. Crucially then, context from the DS perspective can be seen to refer to both the broader discourse context and the context provided by the tree at a given point in the parsing process, as well as the sequence of actions used thus far (see also Sect. 2.7).

2.3 Step-through 'John Loves Mary'

The Axiom states that we begin a parse with a single node consisting of a requirement for a proposition, and with the pointer at that node. This reflects the expectation on the part of the parser that the speaker will provide some propositional content—namely, an utterance that is truth-evaluable. Using the computational actions Introduction and Prediction creates a partial tree waiting for a subject and predicate,[5] as shown in Fig. 1.

[5] Note that these particular computational rules are considered to be routinised instantiations of common parsing actions. As such, they are language specific, and are appropriate for English as it has subject-verb-object (SVO) word order. As discussed in Cann et al. (2005), these rules do not generalise to free word order and prodrop languages (e.g. Japanese, Spanish). For more discussion of routinisation in DS, see Bouzouita (2009); Bouzouita and Chatzikyriakidis (2009). Bouzouita (2009) approaches language change through adopting the notion of routinisation in the sense of Pickering and Garrod (2004) under which language change over time is explained through progressive changes in lexical specifications, with each state being a reflex of general properties of tree growth which standardly dictate the limits on permissible variation (Bouzouita 2009, p20).

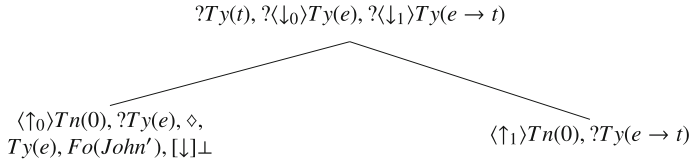
$?Ty(t), ?\langle\downarrow_0\rangle Ty(e), ?\langle\downarrow_1\rangle Ty(e \rightarrow t)$
$\langle\uparrow_0\rangle Tn(0), ?Ty(e), \diamond,$
$Ty(e), Fo(John'), [\downarrow]\bot$
$\langle\uparrow_1\rangle Tn(0), ?Ty(e \rightarrow t)$

Fig. 2 Parsing John

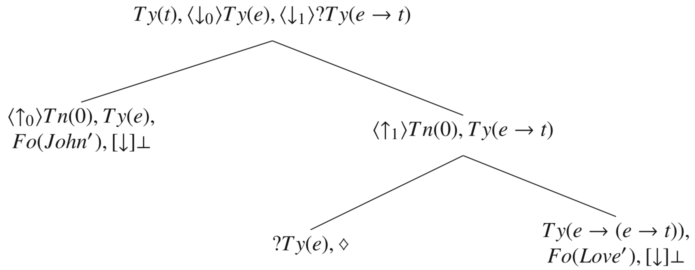
$Ty(t), \langle\downarrow_0\rangle Ty(e), \langle\downarrow_1\rangle ?Ty(e \rightarrow t)$
$\langle\uparrow_0\rangle Tn(0), Ty(e),$
$Fo(John'), [\downarrow]\bot$
$\langle\uparrow_1\rangle Tn(0), Ty(e \rightarrow t)$
$?Ty(e), \diamond$
$Ty(e \rightarrow (e \rightarrow t)),$
$Fo(Love'), [\downarrow]\bot$

Fig. 3 Parsing …loves…

Given a simple example John loves Mary using the lexical entries shown in Examples 1 and 2, the next step would be to parse the word John, as per the lexical actions discussed above, leaving the resulting tree, as in Fig. 2.

Further rules allow us to update the tree such that the pointer is at the node with a requirement $?Ty(e \rightarrow t)$, where we can parse the word loves. They are THINNING, which removes a requirement from a node if the completed form is also present, COMPLETION, which moves the pointer to a mother node if no requirements are outstanding at the current node, and ANTICIPATION, which moves the pointer to any daughter node with outstanding requirements.

The application of the lexical actions for loves in Example 2 results in the tree shown in Fig. 3. This leaves the pointer at the object node, where the trigger condition for Mary ($?Ty(e)$) is met, and the tree can then be completed using THINNING, COMPLETION and ELIMINATION (Example 3), which derives the value of a mother node's content and type from those of its daughters (using β-reduction), resulting in the tree shown in Fig. 4.

Example 3

	IF	$?Ty(X), \langle\downarrow_0\rangle(Fo(\alpha), Ty(Y)), \langle\downarrow_1\rangle(Fo(\beta), Ty(Y \rightarrow X)$
ELIMINATION	THEN	$\texttt{put}(Fo(\beta(\alpha)), Ty(X))$;
	ELSE	ABORT

2.4 Underspecification

Central to DS is the notion of underspecification. At every non-final point in the parse, the partial tree may be underspecified, with each type of tree decoration ($Fo()$, $Ty()$, $Tn()$) a potential source of underspecification.

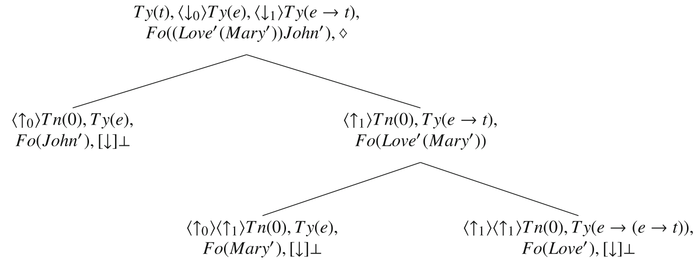
$Ty(t), \langle\downarrow_0\rangle Ty(e), \langle\downarrow_1\rangle Ty(e \rightarrow t),$
$Fo((Love'(Mary'))John'), \diamond$
$\langle\uparrow_0\rangle Tn(0), Ty(e),$
$Fo(John'), [\downarrow]\bot$
$\langle\uparrow_1\rangle Tn(0), Ty(e \rightarrow t),$
$Fo(Love'(Mary'))$
$\langle\uparrow_0\rangle\langle\uparrow_1\rangle Tn(0), Ty(e),$
$Fo(Mary'), [\downarrow]\bot$
$\langle\uparrow_1\rangle\langle\uparrow_1\rangle Tn(0), Ty(e \rightarrow (e \rightarrow t)),$
$Fo(Love'), [\downarrow]\bot$

Fig. 4 Parsing …Mary

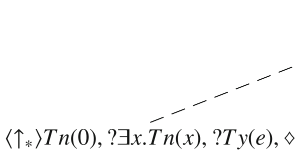
$?Ty(t)$
$\langle\uparrow_*\rangle Tn(0), ?\exists x.Tn(x), ?Ty(e), \diamond$

Fig. 5 Using *ADJUNCTION

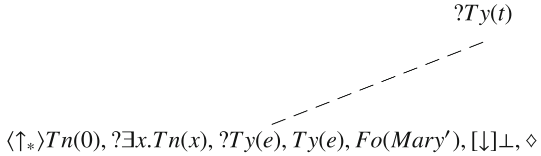
$?Ty(t)$
$\langle\uparrow_*\rangle Tn(0), ?\exists x.Tn(x), ?Ty(e), Ty(e), Fo(Mary'), [\downarrow]\bot, \diamond$

Fig. 6 Parsing Mary, …, after *ADJUNCTION

2.4.1 Unfixed Nodes

Unfixed nodes are nodes whose position ($Tn(x)$) in a tree is initially underspecified, with a requirement to be fixed at a later point in the parse. This means that a parse tree can unfold with certain elements not yet in the positions they will occupy in the final tree, without having to resort to the notion of movement of already fixed elements in a syntactic tree. A canonical example of this is left topic dislocation, as in Mary, John loves, in which, although it is the first item encountered in the string, Mary is the object of loves, not the subject. In a transformational account, it is assumed that Mary is moved from its usual object position, as a focus effect, but, in DS, a parse may proceed using the weak structural relation rule of *ADJUNCTION (Fig. 5) from where the trigger requirement for parsing Mary is met, as in Fig. 6.[6]

THINNING and COMPLETION leave the pointer at the $?Ty(t)$ node, from where John and loves can be parsed as before, with the results shown in Fig. 7.

However, the parse is not yet complete, as the pointer is at a node with an outstanding requirement ($?Ty(e)$) and there is a requirement for a fixed position in the tree on the unfixed node. These can both be satisfactorily resolved by merging the two nodes

[6] For an account of GENERALISED ADJUNCTION as an available strategy whereby a node can be introduced from any node to another of the same type across any arbitrary relation, see Cann et al. (2005, p. 206).

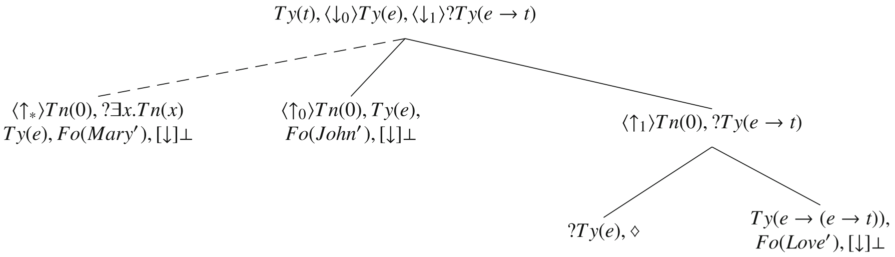
$Ty(t), \langle\downarrow_0\rangle Ty(e), \langle\downarrow_1\rangle ?Ty(e \rightarrow t)$
$\langle\uparrow_*\rangle Tn(0), ?\exists x.Tn(x)$ $Ty(e), Fo(Mary'), [\downarrow]\bot$
$\langle\uparrow_0\rangle Tn(0), Ty(e),$ $Fo(John'), [\downarrow]\bot$
$\langle\uparrow_1\rangle Tn(0), ?Ty(e \rightarrow t)$
$?Ty(e), \diamond$
$Ty(e \rightarrow (e \rightarrow t)),$ $Fo(Love'), [\downarrow]\bot$

Fig. 7 Parsing Mary, John loves …

(using MERGE), resulting in the same tree shown in Fig. 4, with the only difference being in the steps used to get there.

2.4.2 Multiple Unfixed Nodes

An additional constraint is operative on underspecification: No two unfixed nodes of the same modality can co-exist. Rather than being an arbitrary stipulation within DS, this follows as a natural result of the tree logic. Two unfixed nodes of the same modality (i.e. two locally unfixed nodes or two generally unfixed nodes) have the same tree node address and as a result can not be kept distinct. Therefore, any point in the parse in which two putative unfixed nodes co-exist will automatically result in the collapse of these nodes on to each other. Rather than being a challenge for the theoretical approach, this observation has been used to model a number of unrelated phenomena in unrelated languages (i.e. clitic placement in Romance and Greek (Bouzouita and Chatzikyriakidis 2009), focus constructions in Japanese (Seraku and Gibson 2016), scrambling in Korean and Japanese (Kempson and Kiaer 2010), and word order alternations in the Bantu language Rangi (Gibson 2016)).

2.4.3 Metavariables

Underspecification of content ($Fo(\mathbf{U})$) is achieved in DS through metavariables, which formalise one way in which we use context to interpret strings in, for example, pronouns, anaphora and ellipsis.

In DS, a sentence such as He loves cakes would not be well-formed if there were no contextual indication of how to interpret he. This underspecification is important in that it assumes that, as processors, we constantly update our interpretations of utterances based on what we know about the world, previous discourse, or other perceptual indicators (e.g. pointing). Parsing a string with a pronoun in it involves a pragmatic process of SUBSTITUTION; for example, if the string He loves cakes follows John ate all the meringues, we would be able to substitute the formula value $Fo(John')$ for he in the second string, resulting in the parse leading to the complete formula $Fo(Love'(Cake')(John'))$.

The lexical entry for he (see Example 4) contains a formula value that is underspecified; this is in the form of a *metavariable*, $Fo(\mathbf{U}_{Male'})$, with a requirement for a fixed formula value ($?\exists x.Fo(x)$), which must be filled for the parse to be complete.[7]

Example 4 *he*

$$\begin{array}{|ll} \text{IF} & ?Ty(e) \\ \text{THEN} & \texttt{put}(Ty(e), Fo(\mathbf{U}_{Male'})), \\ & ?\langle\uparrow_0\rangle(Ty(t) \wedge \exists x.Tns(x)) \\ & ?\exists x.Fo(x), [\downarrow]\bot) \\ \text{ELSE} & \text{ABORT} \end{array}$$

Pronouns can thus be seen as place-holders for some other information to be assigned from context. First (and second) person pronouns would have similar lexical entries, except that the metavariable is further restricted as to what value it can take according to who is the currently speaker (or hearer) $Fo(\mathbf{U}_{Spkr'})$, which enables DS to account easily for mid-utterance speaker switches in dialogue which involve a change of pronoun (Purver et al. 2010; Kempson et al. 2016, a.o.).

The same is true of other information that might be conveyed by pronouns and pronominal elements such as gender. In languages with grammatical gender, metavariables serve to further restrict possible substituents while fully-specified information is not provided by the form itself. In the Bantu languages for example, subject (and object) markers convey information pertaining to noun classes (which function as grammatical genders). Within the DS approach, these markers can be analysed as introducing a metavariable which is restricted in terms of noun class (which also encodes a singular versus plural distinction). Thus, the class 10 subject marker *zi-* in Swahili conveys that the possible interpretation and annotation of the node can only be an nominal of class 10. As was seen with the English pronoun *he*, this is encoded in the lexical entry of the marker as can be seen in (5).

Example 5 *zi-*

$$\begin{array}{|ll} \text{IF} & ?Ty(e) \\ \text{THEN} & \texttt{put}(Ty(e), Fo(\mathbf{U}_{Class10})), \\ & ?\exists x.Fo(x), [\downarrow]\bot) \\ \text{ELSE} & \text{ABORT} \end{array}$$

2.5 Linked Trees

As can be seen in earlier examples in which a formula value from one tree can be pragmatically substituted into another, trees are not constructed in isolation. This has implications for many different types of construction, including coordination and relative clauses and adjunction in general. The way these are dealt with in the DS framework is through the building of separate, but linked semantic trees, in tandem. The rule of Link Adjunction allows us to construct a new tree, linked to the tree currently under construction, and carrying the requirement for a copy of the formula value from the node at which Link Adjunction is applied. The shared term ensures the flow of information between the two trees.

[7] Note that "$?\langle\uparrow_0\rangle(Ty(t) \wedge \exists x.Tns(x))$" is a case constraint to prevent strings such as Mary liked he being licensed.

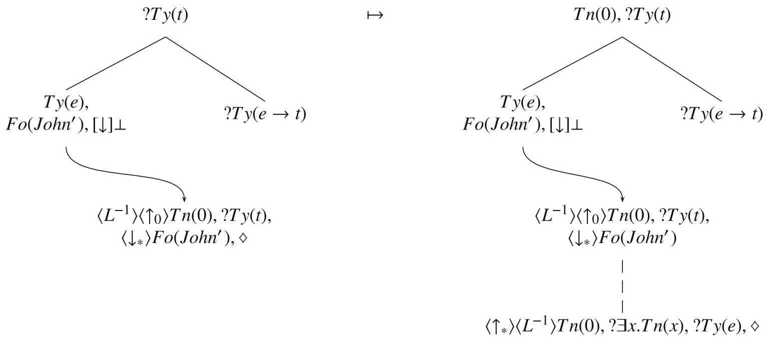
$?Ty(t)$
$\mapsto$
$Tn(0), ?Ty(t)$
$Ty(e)$, $Fo(John'), [\downarrow]\bot$
$?Ty(e \rightarrow t)$
$\langle L^{-1}\rangle\langle\uparrow_0\rangle Tn(0), ?Ty(t)$, $\langle\downarrow_*\rangle Fo(John'), \diamond$
$Ty(e)$, $Fo(John'), [\downarrow]\bot$
$?Ty(e \rightarrow t)$
$\langle L^{-1}\rangle\langle\uparrow_0\rangle Tn(0), ?Ty(t)$, $\langle\downarrow_*\rangle Fo(John')$
$\langle\uparrow_*\rangle\langle L^{-1}\rangle Tn(0), ?\exists x.Tn(x), ?Ty(e), \diamond$

Fig. 8 Parsing John …(ready to parse who), using Link Adjunction and *Adjunction

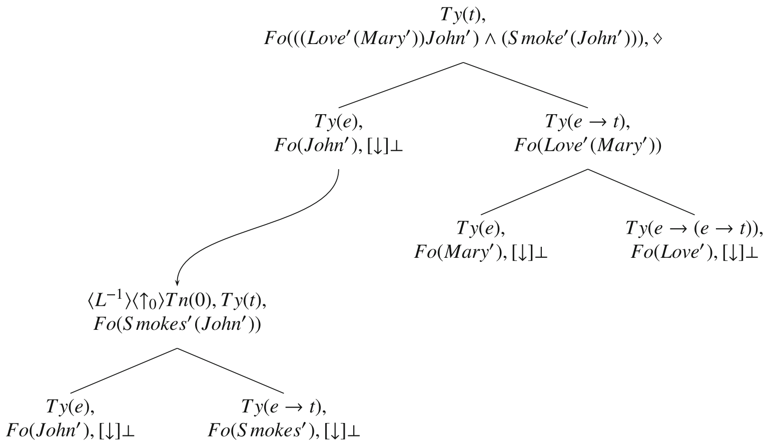
$Ty(t)$, $Fo(((Love'(Mary'))John') \wedge (Smoke'(John'))), \diamond$
$Ty(e)$, $Fo(John'), [\downarrow]\bot$
$Ty(e \rightarrow t)$, $Fo(Love'(Mary'))$
$Ty(e)$, $Fo(Mary'), [\downarrow]\bot$
$Ty(e \rightarrow (e \rightarrow t))$, $Fo(Love'), [\downarrow]\bot$
$\langle L^{-1}\rangle\langle\uparrow_0\rangle Tn(0), Ty(t)$, $Fo(Smokes'(John'))$
$Ty(e)$, $Fo(John'), [\downarrow]\bot$
$Ty(e \rightarrow t)$, $Fo(Smokes'), [\downarrow]\bot$

Fig. 9 Parsing John, who smokes, loves Mary—Completed parse

2.5.1 Relative Clauses

In combination, the rules of Link Adjunction and *Adjunction allow us to project a linked tree (using the modal operators $\langle L\rangle$ and $\langle L^{-1}\rangle$) which must contain a copy of the node it is linked to, so that sentences such as John, who smokes, loves Mary can be parsed as shown in Fig. 8. Later in the parse, shown in Fig. 9, the unfixed node is merged to a fixed node position, carrying the copy of the node the linked tree is from. This can then be evaluated using the rule of Link Evaluation, giving the correct interpretation that it is John who both smokes and loves Mary. Notice that, as with the pronoun he, above, there is no trace of the word who on the final tree; the lexical actions merely provide a metavariable which is then updated by the copy of $Fo(John')$.

2.6 Dialogue

Due to the (inter)action-based nature of the machinery of DS, it is uniquely placed such that it can also be viewed as a grammar of dialogue rather than of sentences in isolation. In DS, generation (production) uses the same tree representations and actions as parsing (comprehension), with the addition of a goal tree which is subject to a subsumption check of the partial tree under construction at every step. Note that the goal tree can itself be partial; the only stipulation is that it is more advanced than the current parse state. And that the current parse state is more enriched than the preceding tree state (if there was one). This means that DS directly models dialogue phenomena such as backchannels and clarification requests (Eshghi et al. 2015; Howes and Eshghi 2017), self-repair (Hough 2014) and split utterances (Kempson et al. 2016, a.o.), without recourse to a competence/performance distinction and without requiring any additional level of representation. In the case of co-construction of utterances, for example, a hearer can simply take over from the initial speaker on the basis of the partial tree they themselves have so far been constructing as a parse tree.

2.7 Context

Our discussion so far has largely been on the basis of taking uttered sentences as though in isolation, but, as the modelling of pronoun resolution and the interactive dynamics of dialogue make clear, processing of language, whether production or parsing, invariably requires incremental access to previous states—node contents, partial trees, and even sequences of actions—so the process of tree growth is never in isolation but always against context of some sort, here modelled, just like the building up of content, in tree growth terms.

2.8 Summary

This introductory paper has provided an overview of Dynamic Syntax and the tools and mechanisms adopted by the framework. After introducing the main conceptual foundations upon which the approach is based (Sect. 1), it introduced the mechanisms which constitute the formal means of representation —semantic trees, tree update (or tree growth), underspecification and link structures (Sect. 2).

For a more in-depth presentation of the formal tools of the framework, including the specific details of the lexical rules see Cann et al. (2005). Kempson et al. (2011) provides an up-to-date account of a range of phenomena modelled from the DS perspective.

Additional discussion of the constraint which allows only one fixed node at a time can be found in Bouzouita (2009), Bouzouita and Chatzikyriakidis (2009), Chatzikyriakidis (2010), Chatzikyriakidis and Kempson (2011), Gibson (2012, 2016), Seraku (2013) and Seraku and Gibson (2016).

3 The Papers in this Collection

The papers in this collection are all unified by an exploration of the concepts of parsing incrementality and underspecification, and the adoption of a DS-inspired approach to modelling natural language. Five of the papers present analyses of specific phenomena in specific languages, all from the perspective of a incremental parsing/production approach (Seraku, Kiaer, Chatzikyriakidis, Yang and Wu, Christopher). In addition to this, the papers by Purver et al. and Howes and Eshghi couple DS with different semantic models (Vector Space Semantics; Purver et al. and Type Theory with Records in Howes and Eshghi) to explore how far DS can account for the incremental compositionality of (shared) semantic representations.

Chatzikyriakidis investigates the historical development of three different clitic systems of Standard Modern, Cypriot and Pontic Greek from a common linguistic ancestor. He argues that the transition from Koine Greek to the Medieval varieties and from the Medieval varieties to the respective modern ones can be explained by processes of routinization and parsing/hearer asymmetries as two of the driving factors behind syntactic change.

The paper by Christopher presents a Dynamic Syntax analysis of the phenomenon of differential object marking in Kazakh. The difference in the pragmatics associated with marked and unmarked direct objects, as well as the syntactic restrictions on the positioning of unmarked direct objects, are explained through the differential application of the processing options which can use either fixed or unfixed nodes.

Howes and Eshghi present a corpus study of feedback in dialogues and describe how a low-level, semantic processing model in Dynamic Syntax, using the predictive, incremental and interactive nature of the formalism, accounts for this feedback, and where it occurs. This model shows how feedback serves to continually realign processing contexts and thus manage the characteristic divergence and convergence that is key to moving dialogue forward.

The contribution by Kiaer aims to explain Korean speakers' strong preference for incremental, left-to-right structure building, showing how a range of phenomena reflect the dynamics of structural growth in Korean. She argues that these phenomena necessitate adopting a grammar formalism with left-to-right incrementality as a core property of the syntactic architecture.

The Purver et al paper presents a version of DS which, rather than relying on symbolic representations of meaning, is assigned a compositional distributional semantics which enables incremental judgements of similarity or disambiguation. The model is implemented and evaluated on real data.

Seraku examines case marking in Japanese and adopts a DS-style approach to the notion of grammatical relations. While the nominative marker *-ga* is most commonly thought of as a subject marker, Seraku analyses data where *-ga* marks an object or a single clause exhibits a co-occurence of *ga*. He accounts for these data by assuming that a case marker maximally excludes potential landing sites of an unfixed node (rather than uniquely identifying a landing site).

The paper by Yang and Wu examines minimisers in Mandarin Chinese, with a focus on the *lian...dou* construction. They argue that, contrary to previous accounts, the distribution and interpretation of the *lian…dou* construction can only accurately be captured through the development of an account which incorporates semantic, syntactic and pragmatic elements—in line with the DS approach to natural language.

We believe that together, this Special Issue presents an up-to-date account of the Dynamic Syntax framework, including the tools and machinery made available by the formalism alongside the latest application of this approach to a range of phenomena across unrelated languages. The Special Issue both contributes to the development of the framework, as well as presenting cutting-edge research in a number of distinct fields of linguistics and the study of language. The papers are unified by the emphasis placed on the dynamic nature of the parsing/production process, thereby foregrounding the incremental nature of structure building and the establishment of meaning in context.

Acknowledgements We would like to thank the attendees of the First Dynamic Syntax conference (held at SOAS University of London) where a number of the papers presented in this Special Issue were initially presented. Thanks go also to the broader Dynamic Syntax community who have contributed to this volume, either directly or through ongoing thought-provoking and inspiring conversations. Finally, thanks to the anonymous reviewers who read and commented on the submissions, as well as the JLLI editorial team whose invitation and subsequent support led to this Special Issue. Howes was supported by two grants from the Swedish Research Council (VR); 2016-0116—Incremental Reasoning in Dialogue (IncReD) and 2014-39 for the establishment of the Centre for Linguistic Theory and Studies in Probability (CLASP) at the University of Gothenburg.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit http://creativecommons.org/licenses/by/4.0/.

References

Blackburn, P., & Meyer-Viol, W. (1994). Linguistics, logic and finite trees. *Logic Journal of the Interest Group of Pure and Applied Logics, 2*(1), 3–29.

Bouzouita, M. (2009). Modelling syntactic variation. *Diálogo de la Lengua, 1,* 1–25.

Bouzouita, M., & Chatzikyriakidis, S. (2009). Clitics as calcified processing strategies. In *14th International Lexical Functional Grammar (LFG-2009)* (pp. 188–207). CSLI Publications.

Cann, R., Kempson, R., & Marten, L. (2005). *The dynamics of language*. Oxford: Elsevier.

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

Chatzikyriakidis, S. (2010). Clitics in four dialects of Modern Greek: A dynamic account. Ph.D. thesis, King's College London

Chatzikyriakidis, S., & Kempson, R. (2011). Standard Modern and Pontic Greek person restrictions: A feature-free dynamic account. *Journal of Greek Linguistics, 11,* 127–166.

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

Eshghi, A., Howes, C., Hough, J., Gregoromichelaki, E., & Purver, M. (2015). Feedback in conversation as incremental semantic update. In *Proceedings of the 11th international conference on computational semantics (IWCS).*

Gibson, H. (2016). A unified dynamic account of auxiliary placement in Rangi. *Lingua, 184,* 79–103.

Gibson, H. C. (2012). Auxiliary placement in Rangi: A Dynamic Syntax perspective. Ph.D. thesis, SOAS, University of London.

Gregoromichelaki, E. (2006). Conditionals: A Dynamic Syntax account. Ph.D. thesis, King's College London.

Hough, J. (2014). Modelling incremental self-repair processing in dialogue. Ph.D. thesis, Queen Mary University of London.

Howes, C., & Eshghi, A. (2017). Feedback Relevance Spaces: The organisation of increments in conversation. In *Proceedings of the 12th international conference on computational semantics (IWCS)*.

Kempson, R., Cann, R., Gregoromichelaki, E., & Chatzikiriakidis, S. (2016). Language as mechanisms for interaction. *Theoretical Linguistics, 42*(3–4), 203–275.

Kempson, R., Gregoromichelaki, E., Meyer-Viol, W., Purver, M., White, G., & Cann, R. (2011). Natural-language syntax as procedures for interpretation: The dynamics of ellipsis construal. In A. Lecomte & S. Tronçon (Eds.), *Ludics, Dialogue and Interaction, no. 6505 in Lecture Notes in Computer Science* (pp. 114–133). Berlin, Heidelberg: Springer.

Kempson, R., & Kiaer, J. (2010). Multiple long-distance scrambling: Syntax as reflections of processing. *Journal of Linguistics, 46*(01), 127–192.

Kempson, R., Meyer-Viol, W., & Gabbay, D. (2001). *Dynamic Syntax: The flow of language understanding*. Oxford: Blackwell.

Pickering, M., & Garrod, S. (2004). Toward a mechanistic psychology of dialogue. *Behavioral and Brain Sciences, 27,* 169–226.

Purver, M., Gregoromichelaki, E., Meyer-Viol, W., & Cann, R. (2010). Splitting the 'I's and crossing the 'you's: Context, speech acts and grammar. In *Proceedings of the 14th SemDial Workshop on the Semantics and Pragmatics of Dialogue*, Poznań (pp. 43–50).

Sadrzadeh, M., Purver, M., Hough, J., & Kempson, R. (2018). Exploring semantic incrementality with Dynamic Syntax and Vector Space Semantics. In *Proceedings of the 22nd SemDial workshop on the Semantics and Pragmatics of Dialogue* (pp. 122–132).

Sag, I. A., Wasow, T., & Bender, E. M. (2003). *Syntactic theory: A formal introduction*. Stanford: CSLI Publications.

Seraku, T. (2013). Multiple foci in Japanese clefts revisited: A semantic incrementality account. *Lingua, 137,* 145–171.

Seraku, T., & Gibson, H. (2016). A Dynamic Syntax modelling of Japanese and Rangi clefts: Parsing incrementality and the growth of interpretation. *Language Sciences, 56,* 45–67.

Steedman, M. (2000). *The syntactic process*. Cambridge, MA: MIT Press.

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.