# Proposals to generalise Dynamic Syntax for wider application

Julian Hough,[1] Arash Eshghi,[2]
Matthew Purver[1] and Graham White[1]
[1]Queen Mary University of London
[2]Heriot-Watt University
j.hough@qmul.ac.uk

We present proposals to generalise Dynamic Syntax (DS) from its original conception by Kempson et al. (2001). This is not only driven by intellectual curiosity, but by academic sociological need. We seek to widen the application domains of DS by generalising it, and in turn increase the number of DS practitioners, through the following:

**Proposal 1: Make it clear what DS is all about: The *dynamics* of natural language.** In the spirit of Kempson et al. (2001), DS is primarily about how a representation is built up over time, with at least a word-by-word granularity, by natural language utterances. We must be honest in cases where the utterance-final formulae of a DS variant and another formalism are similar (Hough et al., 2015)– it is not necessarily the final result, but the way we get there that we are interested in. White (2017) shows how the interesting mathematical properties of DS lie in the way the proof of grammaticality is constructed (rather than in the semantic formulae on DS tree nodes), and Sato (2011), Hough (2011) and Eshghi et al. (2015) show how the dynamics of DS represented by an action graph can model garden paths, self-repair, other-repair and backchannel processing. None of these inherently rely on a particular semantics, but more on the dynamic properties of DS. In light of this, we should define the core components of DS; we propose these should include: *typed trees* (including unfixed nodes and linked trees), basic $\lambda$-calculus composition of formulae, a notion of *subsumption* of formulae, *computational actions*, and the definition of context as a time-linear action graph. Bi-directionality of parsing and generation, and co-construction of representations naturally fall out of these properties.

**Proposal 2: Choose the composition calculus to generalise with: lambda, but not necessarily epsilon or TTR.** We suggest a move away from strict adherence to the original $\epsilon$-calculus extended $\lambda$-calculus DS or recent variants such as DS-TTR (Purver et al., 2011) which define permissible DS formulae. While the $\epsilon$-calculus and TTR are very useful, they are not the only ways to represent meaning. We argue in many cases it is useful to preserve the well-known elements of the $\lambda$-calculus that allow composition of multiple formulae such as $\alpha$-conversion (re-naming of variables with name clashes) and $\beta$-reduction (function application), as these generalise to many possible semantic systems, though we remain open to other forms of general composition calculus.

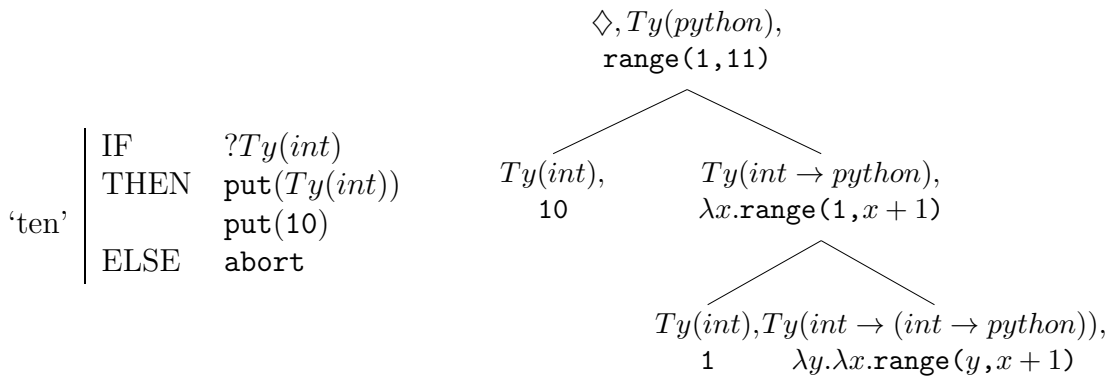


Figure 1: An example lexical action and tree in DS-Python

**Proposal 3: Liberalise permissible node types.** We advocate going beyond $t$, $e$, $cn$, $e_s$, and functions using these, to a generalisation using all and any types in a type system of the user's choosing. As formulae using the $\lambda$-calculus are already in a second order formalism, DS is Turing complete, and so complexity is not necessarily a sound argument for limiting horizons. Note that we advocate keeping DS tree logic, including unfixed nodes and linked trees, but are proposing to go beyond the standard types decorating the nodes.

**Proposal 4: Liberalise permissible semantic representation languages and interpretation functions.** Existing DS variants include DS-$\epsilon$ (Kempson et al., 2001), DS-FOL (Kempson et al., 2001, without the $\epsilon$-calculus), and DS-TTR (Purver et al., 2011). Also, currently under development is a distributional semantics variant which we will call DS-Tensor (Sadrzadeh and Purver, 2017).

However, we can go further. We claim as long as one can make $\lambda$-calculus style functional abstractions of contribution-final semantic representations, any existing syntax-semantic system can be used for DS formulae. One example is programming languages– as an illustration, DS-Python could have top-level formulae of type *python* program, and the tree nodes could contain other Python data types like *int*, and functions from one to the other like $int \rightarrow python$. See a proposed lexical action for the word 'ten' and the final tree of DS-Python for the utterance "a range of integers from one to ten" in Figure 1.

In future, we can plan for other programming and formal representations, including more embodied representations such as DS-BML (Kopp et al., 2006) which could operate a virtual human agent word-by-word, or DS-G-code which could operate a range of CNC robots such as that in Hough and Schlangen (2017). The options are limitless.

**Proposal 5: Use it!** Under this view, anything is possible in DS from an application point of view. As long as the semantic interpretation function can apply word-by-word and we can characterize $\lambda$-calculus style abstractions of formulae, the interface to many types of action execution system are possible and we will be able to use many devices much more fluidly than was previously possible, including in human-robot interaction. This also opens up the possibility of data-driven methods such as those by Eshghi et al. (2013) for inducing DS lexical actions from arbitrary ⟨utterance, formula⟩ pairs. There is much to be done.

# References


Eshghi, A., Hough, J., and Purver, M. (2013). Incremental grammar induction from child-directed dialogue utterances. In *The Fourth Annual CMCL Workshop*, pages 94–103, Sofia, Bulgaria. ACL.

Eshghi, A., Howes, C., Gregoromichelaki, E., Hough, J., and Purver, M. (2015). Feedback in conversation as incremental semantic update. In *Proceedings of the 11th International Conference on Computational Semantics*, pages 261–271, London, UK. ACL.

Hough, J. (2011). Incremental semantics driven natural language generation with self-repairing capability. In *Proceedings of the Second Student Research Workshop associated with RANLP 2011*, pages 79–84.

Hough, J., Kennington, C., Schlangen, D., and Ginzburg, J. (2015). Incremental semantics for dialogue processing: Requirements, and a comparison of two approaches. In *Proceedings of the 11th International Conference on Computational Semantics*, pages 206–216.

Hough, J. and Schlangen, D. (2017). It's not what you do, it's how you do it: Grounding uncertainty for a simple robot. In *Proceedings of the 2017 ACM/IEEE International Conference on Human-Robot Interaction*, pages 274–282. ACM.

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

Kopp, S., Krenn, B., Marsella, S., Marshall, A. N., Pelachaud, C., Pirker, H., Thórisson, K. R., and Vilhjálmsson, H. (2006). Towards a common framework for multimodal generation: The behavior markup language. In *International workshop on intelligent virtual agents*, pages 205–217. Springer.

Purver, M., Eshghi, A., and Hough, J. (2011). Incremental semantic construction in a dialogue system. In Bos, J. and Pulman, S., editors, *Proceedings of the 9th IWCS*, pages 365–369, Oxford, UK.

Sadrzadeh, M. and Purver, M. (2017). Incremental distributional semantics for Dynamic Syntax. In *1st Dynamic Syntax Conference*, London.

Sato, Y. (2011). Local ambiguity, search strategies and parsing in dynamic syntax. *The Dynamics of Lexical Interfaces. CSLI Publications*.

White, G. (2017). Dynamic Syntax and proof theory. *Theoretical Linguistics*, 43(1-2):135–140.