Modelling Parsing and Unparsing

Vadim Zaytsev[1] and Anya Helene Bagge[2]

[1] Universiteit van Amsterdam, The Netherlands, vadim@grammarware.net
[2] Universitetet i Bergen, Norway, anya@ii.uib.no

Abstract

Any expert researcher in the field of parsing knows exactly what parsing is, and does not need any definition for it. With some expendable mental effort, we can all read one another's papers on "parsing" that do not explicitly state whether the focus is on the process of syntactic analysis of tokenised sequences, or on the generalised scannerless parsing, or on extracting linked structured information from flat textual data, or on converting concrete syntax trees to abstract ones — all these activities, and many others, at some point have been referred to as "parsing", with details emerging from the context. Yet, some of these processes are inherently different than others, and that distinction is not apparent in any way to outsiders and to students studying compiler construction or software language engineering, in particular.

Recently we have proposed a unified model of parsing in a broad sense [6] — a lattice-like diagram with nodes/cells corresponding to different kinds of software artefacts such as parse trees or token sequences and edges representing existing mappings between them which are (or can be) referred to as "parsing" and "unparsing". It builds on top of our previous work on understanding (un)parsing [1,4, etc] and refers to many well-known (un)parsing activities. The model can be "renarrated" in order to explain a particular parsing technology [5], which we already successfully use in teaching [2].

The twelve kinds of artefacts commonly found in software language engineering can be seen on Figure 1:

- ◊ Str — a string, a file, a byte stream.
- ◊ Tok — a finite sequence of tokens which, when concatenated, yield Str. Includes spaces, line breaks, comments, etc — collectively, *layout*.
- ◊ TTk — a finite sequence of typed tokens, possibly with layout removed, some classified as numbers, strings, etc.
- ◊ Lex — a lexical source model (ILA, TEBA, srcML, etc) that adds grouping to typing; a possibly incomplete tree connecting most tokens within one structure.
- ◊ For — a forest of parse trees, a parse graph or an ambiguous parse tree.
- ◊ Ptr — an unambiguous parse tree, the leaves can be concatenated to form Str.
- ◊ Cst — a parse tree with concrete syntax information. Structurally similar to Ptr, but abstracted from layout and other minor details. Comments could still be a part of the Cst model, depending on the use case.
- ◊ Ast — a tree which contains only abstract syntax information.
- ◊ Pic — a picture, which can be an ad hoc model, a natural model [3] or a rendering of a formal model.

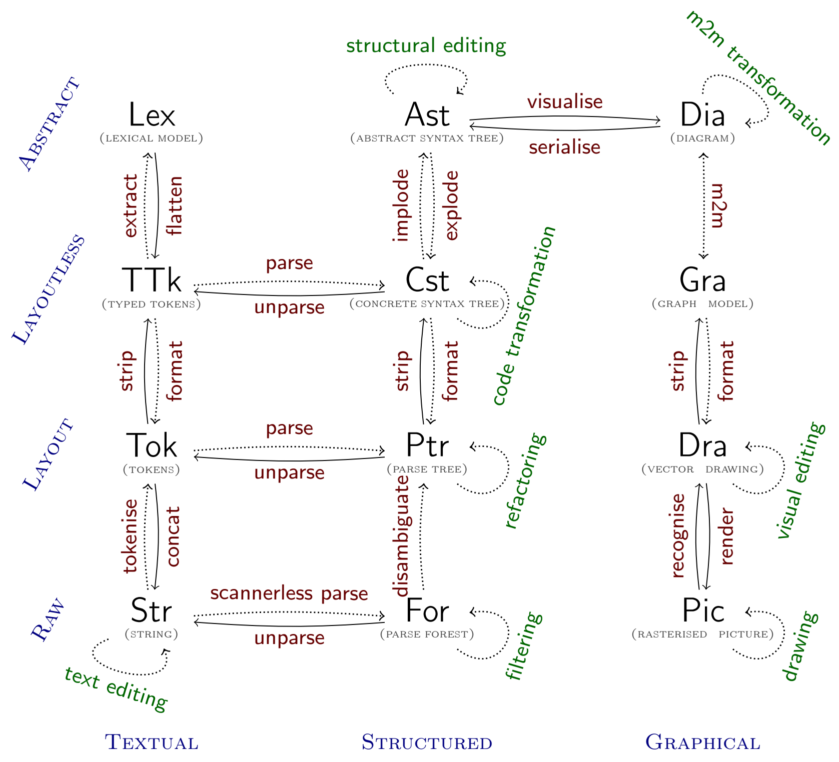

Fig. 1. A model of parsing in a broad sense [6].

◊ Dra — a graphical representation, a drawing in the sense of GraphML or SVG.

◊ Gra — an entity-relationship graph, a categorical diagram or any other primitive "boxes and arrows" level model.

◊ Dia — a diagram, a graphical model in the sense of EMF or UML, a model with an explicit advanced metamodel.

References

1. A. H. Bagge and T. Hasu. A Pretty Good Formatting Pipeline. In M. Erwig, R. F. Paige, and E. V. Wyk, editors, *SLE'13, LNCS 8225*, pages 177–196, 2013.
2. A. H. Bagge, R. Lämmel, and V. Zaytsev. Reflections on Courses for Software Language Engineering. Submitted to EduSymp'14. Pending reviews, July 2014.
3. Z. Zarwin, M. Bjekovic, J.-M. Favre, J.-S. Sottet, and H. A. Proper. Natural Modelling. *Journal of Object Technology*, 13(3):4:1–36, 2014.
4. V. Zaytsev. Formal Foundations for Semi-parsing. In S. Demeyer, D. Binkley, and F. Ricca, editors, *CSMR-WCRE'14*, pages 313–317. IEEE, Feb. 2014.
5. V. Zaytsev. Understanding Metalanguage Integration by Renarrating a Technical Space Megamodel. Submitted to GEMOC'14. Pending reviews, July 2014.
6. V. Zaytsev and A. H. Bagge. Parsing in a Broad Sense. In *MoDELS'14*, Oct. 2014.