Subtext: A Proposed Processual Grammar for a Multi-Output Pre-Format

Subtext: návrh postupové gramatiky na předformát určený pro různá výstupní zařízení

John Haltiwanger

Abstract: Academic publishing today faces a reality in which providing multiple formats–generally HTML and PDF–is becoming a necessity. The production of multiple outputs involves a workflow of *generative typesetting*. Generative typesetting involves many constraints that resulting from edge cases between formats which must be accounted for. Against the backdrop of theory in the field of new media, a new approach towards generative typesetting is proposed. A separation of translation from effect, akin to the division of style and content in HTML/CSS, can effect a *transmutable translation* layer in which syntax, effect, and even a pre-format's reserved characters can be defined in configuration files. This transmutability is desirable because every generative typesetting workflow faces particular specificities which should be addressable without the introduction of "glue."

Keywords: typesetting, processual grammar, pre-format, multi-output.

Abstrakt: Článek přináší několik myšlenek a úvah na návrh gramatiky pro předformát u vícenásobného výstupu z jednoho zdrojového kódu.

Klíčová slova: sazba, typografie, postupová gramatika, předformát, vícenásobný výstup.

Introduction

Over the course of my recent work obtaining a master's degree in New Media at the Universiteit van Amsterdam, I had the opportunity to investigate the field of generative design–and especially generative typesetting–in the context of a thesis that interrogates existing media theories for their efficacy in describing the dynamics involved in generative workflows.

The core impulse of my thesis began with the idea of a web application which offers a simple means of 'writing once, reading everywhere.' In terms of

 doi: 10.5300/2011-2-4/140

the practical problem domain (academic publishing of humanities texts), this meant targetting two formats: HTML and PDF. There seemed to be interesting questions of 'materiality' in any discussion of the envisioned workflow:

1. A pre-format in which the source document is written. (For my thesis, I chose Markdown.) What *material specificities* does such a format exhibit? From a less directed orientation, what does it mean to write in a format whose sole purpose is to be translated into *other* formats?
2. A 'wrapper' (or translation layer) application. (For now, in terms of ConTEXt, that means Pandoc.) What does it say about the 'materiality' of the final product when, indeed, the final "product" is a multitude of output files (PDF for print, PDF for screen, HTML for browser, HTML for handheld, etc.)?
3. The glue layer. This layer was recognized as inevitable from the outset due to both personal conversations with people who have attempted cross-format publishing and the knowledge that with (translation) software, it will never Just Work Right. The 'materiality' of the outputs further spreads out across code written to satisfy edge cases and account for inadequacies in either of the other two layers.

The Interest in Materiality

Recently there has been a shift within the field of new media back towards an adherence to Marshall McLuhan's old motto: "the medium is the message." This shift has been very productive in its inspiration towards *software studies, medium-specific analysis*, and a host of other approaches of examining media as media. While quite productive in generating theory, this shift has not necessarily been so productive in terms of deciphering the (often unasked) question: What defines a medium?

The contemporary fixation with 'materiality' is a result of seeking the constitution of media from inside themselves, a fact that belies a neglect in attention to the shaping of media by processes external to themselves such as economics, political climate (including legal structures), history, and imagination.

It's All Process: A Shift to an Analytics of Becoming

There may be a better way to critically engage with media, one that can account for and respond to the fragmentation that the concept of 'medium' has experienced since the advent of the personal computer. Though relatively obscure in English language discourse, Gilbert Simondon's theory of *ontogenesis* is remarkably applicable for describing the unfolding of structures within the computer metamedium.[1]

[1] Loosely translateable as a machine in which the available potential of the machine is definable from within that machine itself, a feature that lets the computer "play host" to other media.

Simondon's ontogenesis transcends the traditional Western conundrum of existence ("what am I? what does it mean that I am?") by asking the question: "How did I become? What am I becoming?"

This is a radical shift, as the disconnect between the conditions that shape us and our current form is missing from many fields—the most obvious example lying in the forms of economic analysis that refuse to integrate so-called "externalities" such as vertical market dynamics, resource depletion, pollution, and the general turmoil associated with extravagently rich men getting richer while poor people get poorer. In the field of new media, specifically, it allows for a *process-oriented* perspective through which the conditions of a given medium can be integrated into our understanding of that medium. In a way it is not a rejection of 'looking into the medium' for understanding: rather, it implores an addtional *looking around*.

Source Code as a Site of Collective Agency

From the perspective of metamedia, where constraints are defined by programming as much as by the physical properties of the machine, source code becomes a site of agency: if one can rule the source, one can rule the metamedium. Through an examination of the discussion between Hans Magnus Enzensberger and Jean Baudrillard regarding a revolutionary theory of media, I position FLoSS as a unique and potent site of agency within metamedia (especially the still-dominant personal computer). FLoSS not only fits the seven categories of emancipatory uses of media outlined by Enzensberger, it likewise operates in a "reciprocal" (actively reflexive) way that restores the symbolic exchange relation that Baudrillard deems necessary for any subversive potential of media.

Tied to the concept of ontogenesis, FLoSS becomes a vast site of collective becoming within the computer metamedium. While many people are squeamish about granting it this mantle, the truth remains that FLoSS, and only FLoSS, carries within itself the potential for total and radical change of the metamedium. Likewise, only FLoSS provides a means for hedging against the increasing lock-in of metamedia that we find in devices such as the iPhone and the Kindle. These features are significant and the stakes are real: metamedia could cease to be metamedia if it becomes a common-place assumption that one is not intended to program them on one's own and without following strict rules of what can be programmed and how. This is a dangerous path, yet the rapid adoption of the iPhone and iPad—despite the valid criticism that the devices rigidly enforce a crippled, consumerist take on computing—proves that this direction worth examining as a very real potential road for the future of social understandings of computing.

Processual Grammars Organize Process Hybridity

A very real problem in terms of 'medium-specific analysis' that we find when dealing with computers is that there can be distinct specifities even between different versions of the *same program*. Does each new or distinct version constitute a specific medium? Just as the variety of typewriter makes and models do not constitute individual media, it makes no sense to apply the 'medium' label to every application on the computer. Yet there are distinct new capabilities and processes embodied within computer programs, and it seems just as nonsensical to render everything available on the computer as belonging to, or representing, a single medium.

Through an analytics of becoming we are made aware of the various hybridizations involved in the assembling of processes. Take, for example, the modern command-line interface (CLI). The original roots of the CLI lie in the teletype machines, which were quite literally the hybridization of typewriters and computer feedback systems such as tape or punch cards, along with telephone and modem technology. Today there are often other aspects in play: a framebuffer (perhaps even a GUI, in the case of terminal emulators), along with libraries that handle text not represented directly on the keyboard, as well as continuously evolving shells and terminal emulators.

What I argue is that these various configurations—resulting from evolutionary developments in and outside the CLI—represent distinct *processual grammars* that organize various processes into distinct hybridities. Thus the differences between the bash and csh shells (or between GIMP and Photoshop) indicate not the existence of a variety of separate media but rather divergent organizational grammars of process that engender intentional specificities that are designed to overcome or otherwise modify some real, existing constraint. In the case of bash versus csh, the constraint was ease of programming and in the case of the GIMP it was the lack of a free software alternative for hardcore raster image manipulation. The significance of FLoSS in this light is that, thanks to the availability of source code, we have the capacity to modify, extend, or replace processual grammars at will. This is something ultimately familiar to the TEX community, which has seen the steady progression of engines and patches intended to accomodate and account for overcoming constraints within the original implementation. If TEX's entire internals were proprietary and the source code withheld, these efforts would require a completely different level of work.[2]

[2]This is also assuming that Donald Knuth had not been so forthright in his writings on the topic due to proprietary concern for 'trade secrets.'

The Problem Domain

Academic papers in the humanities are still written in WYSIWYG word processors, most often the proprietary Microsoft Word. This is pervasive to the extent that editing jobs require the possession of this software. Yet the chief feature of that software in this problem domain is the "Track Changes" mode—something which version control systems handle seamlessly with regards to plain-text documents. The issue is not technical, then, but a question of interface. TEX is a great system, but it is still a visually cumbersome programming language. An abstraction into a pre-format for interfacing with typesetting could allow the adoption of TEX-based workflows without the necessity of learning TEX (or LATEX or ConTEXt). A new interface for version control based around the workflow demands of publishing and editing would likewise smooth a transition from proprietary conditions.

Requiring proprietary software for academic publishing is a strange product of history and economics. Both of these influences have been known to be overcome through FLoSS. Indeed, history and economics were both overcome when Richard Stallman made an institution and a legal hack around the considered-obsolete institution of code sharing. It is possible to imagine a new system for generative typesetting that outshines previous workflows of academic writing.[3]

Towards A New Processual Grammar for Typesetting

My thesis attempts an actively reflexive methodology: in order to investigate generative design, a workflow based on generative typesetting was employed. This workflow has already been outlined in the introduction to this piece: it is composed of the *pre-format*, the *translation layer*, and the *glue layer*.

It seems to me that with the advent of LuaTEX we now have the opportunity to integrate these three layers and, while doing so, modify them in order to fulfill our expecations in a more efficacious manner. Since LuaTEX allows for the utilization of Lua code, we can theoretically implement anything possible in Lua as an add-on for LuaTEX.

The first stage of this is to re-think the pre-format. I define a pre-format as a markup that is chiefly designed both for ease of *writing* and for *translatability*. In other words, pre-formats exist to provide a drastic reductions in learning-curve and syntatical verbosity. However, prominent pre-formats like Markdown and reStructuredText were designed primarily by programmers for programmers. In other words, the factors involved in typesetting are not always accounted for in these pre-formats. It is time to change this. We

[3] Even if the system outlined here is not ideal, I maintain that there has to be a better way of dealing with publishing in the humanities.

need a pre-format that understands how to pass environment information so that a paragraph, for example, can be marked as a member of an arbitrary environment (or class/id in HTML). Ideally all markup should be augmentable with contextual specificities that then map into the outputs. A huge element of this hybridization is delivering bibliographic/citation functionality that extends to all outputs. Such a hybridization—not currently possible in Pandoc—would elevate this processual grammar above the competition (for instance, BibLaTeX requires significant massaging just to out put the MLA format, a standard in the humanities). Since the translation layer can be implemented in Lua, it is possible to develop a library for LuaTeX that processes this pre-format and output the results in multiple other formats.[4] By hybridizing the translation layer into LuaTeX, beautiful typesetting into PDFs will become a first-order capability of the pre-format. Perhaps the best way to phrase this concept to developers is that it represents a recognition that a *pre-format can be built as a domain-specific language*. To that extent, the syntax, it's effects and even the reserved characters can be specified through configuration files and metaprogroamming. This allows for personally tailored workflows as well as defined and emergent standards of operating.

Glue-Be-Gone

This could theoretically represent a point of transcendence of that final layer of integration: the glue layer. An *transmutable* translation layer allows a new separation of translation from implementation, something akin to the attempted division of form and content vis a vis CSS and HTML. As mention, in this scheme the exact effects of the pre-format syntax (i.e, what the translator outputs upon receiving a certain markup) can be defined on a per-project (or even per-individual) basis. Pandoc, on the other hand, allows for scripting but only in a way that involves writing Haskell code. While this may be fine in some cases, clearly the necessity of learning a new programming language simply to use a translation wrapper is less-than-ideal. By loading rules in from configuration files we can separate the workflow into subsystems that align to different degrees of engagement with the workflow: users can easily interact within the specific syntax of a project, yet they can just as easily begin to modify that syntax by changing values in a configuration file. If they are interested in engaging the outputs more deeply, they can alter other configuration values in order substitute their own values as effects of translation. The point is to allow multiple levels of affecting how *this* gets changed into *that*.

The system need not necessarily be designed for inter-operability between

[4] Another option, worth investigating, is using the powerful grammar processing of Perl 6 in combination with the Parrot Virtual Machine and its native Lua implementation.

specific organizations of workflow. However, even that goal should be relatively accessible if shared code works only on an abstract level and respects arguments such as environments that have been passed from the pre-format. These arguments will have a defined structure within the configuration files, so errors in syntax can be reported when they are found during translation and even shared code libraries can be written to take these structures into account. I am not saying that edge cases will never occur, only that they should be adjustable on a granular level that allows a single workflow to be optimized and *just work*. (These adjustments will require engaging with environment setup through mechanisms such as CSS or ConTEXt macros). One could imagine a university department developing a tailored pre-format that suits their field and generates arbitrarily optimal PDFs and HTML files, something akin to the current prevalence of LaTeX document classes in mathematics and science but with the ease of use and translatibility offered by a pre-format.

This is not to say that "glue" will never be necessary, only that we should recognize its perpetual existence in generative typesetting for multiple outputs. As such, it makes sense to mitigate the source of glue, which can be found in the edge cases that must be accounted for between output formats in any given workflow. The uniqueness of every workflow means that the best solution lies in a system sufficiently flexible enough to accomodate arbitrary demands and inevitable weirdness. While adherence to a standard syntax is possible, there should ultimately be as few constraints placed upon the executiopn of that workflow as possible. As long as the configuration files are available, the system should be able to render any output formats according to the logic declared therein. There is a distinct opportunity to provide a new alternative to academic writing in the humanities. LuaTeX offers a significant site of agency for delivering on this opportunity. While it is obvious that this proposal consists only of theoretical speculation, I hope that the ideas outlined in this text provide a thought-provoking outlook on conceive of this delivery.

john (dot) haltiwanger (at) gmail (dot) com
Amsterdam, The Netherlands