

Only you can see this message



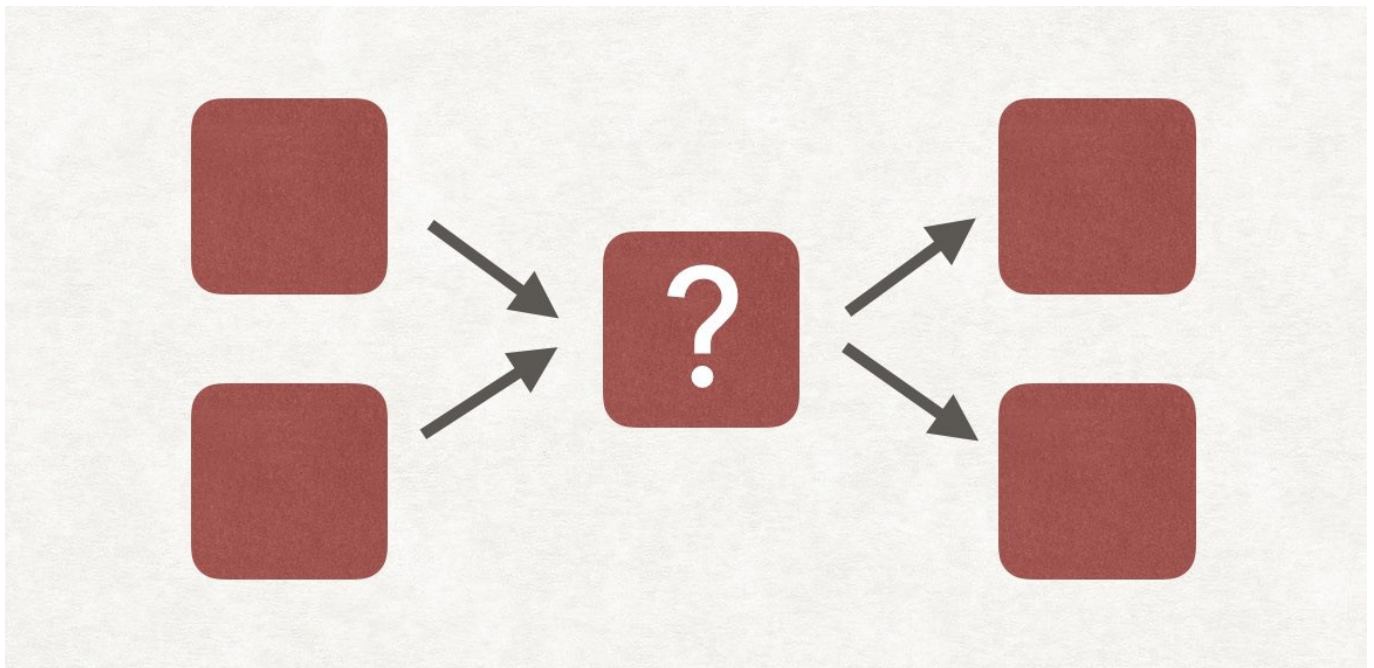
This story's distribution setting is off. [Learn more](#)

The Trouble With Transitions



Aaron A. Reed

Oct 15, 2019 · 7 min read



In my last post I described the .quant format, a minimalist syntax I created for writing *Subcutaneous* (a novel that changes each time it's printed). For example:

```
[DEFINE verbose]
```

```
I'm having a [verbose>really quite wonderful|great] day  
[verbose>, actually, thank you for asking].
```

The `DEFINE` command sets the variable *verbose* to either true or false at random, producing one of two possible outputs when the text is rendered:

I'm having a really quite wonderful day, actually, thank you for asking.
I'm having a great day.

I actually use variables like this a lot in *Subcutanean* to ensure that even though decisions are made at random, the output has a degree of consistency: one narrator might tend to use bigger words, another might consistently be less honest about their emotions, and so on. In fact I'm doing this even without explicit variables — but more about that in a later post. (*Update: this one!*)

The `.quant` syntax was able to remain straightforward in part because of my design decision not to allow any nesting or conditional logic. But almost as soon as I started authoring, and despite my purity of intentions, I stumbled across a situation that was almost impossible to handle without boolean logic: transitions.

Subcutanean contains a lot of sequences — ranging in length from sentences to whole scenes — that might randomly be present or absent, or might at random appear in one of several versions. Sometimes these sequences would abut one other, and when this happened, it became very difficult to write natural-sounding text that could transition between any possible preceding and following text.

For instance, say in one possible lead-in two characters get drunk, while in another they don't. The next sequence could be either a midnight walk or sleeping the evening off. The natural transitions in all four cases — the way a human would write them — are probably unique:

[drunk story] Heads spinning, we spilled outside, raucous. ***[midnight walk]***
[drunk story] We knew our limits; it was time to call it a night. ***[sleep it off]***
[not drunk] Wanting some fresh air, we stepped outside. ***[midnight walk]***
[not drunk] It was getting late. ***[sleep it off]***

You don't *always* need unique text for each possible combination, but clearly a natural transition references both what came before and what's coming next. It's often very hard to write these in such a way that you don't need to know something about both states to do so. Even single-word transitions have implications about what's coming before and

after that might not always hold true: this is of course the entire reason languages have different transition words in the first place, like *and*, *but*, *still*, *anyway*, *therefore*, *nevertheless*, and so on. Each implies something different about why these two thoughts are connected.

I tried to write my way around this for a while, but eventually decided I did need some form of combinatorial conditions, after all. But I was wary about most ways I could think to add them. Adding boolean logic, for instance, such as with a syntax like **[drunk AND MidnightWalk>...]**, could be a slippery slope: if I supported AND, shouldn't I also support OR? What about either of those in combination with NOT? If I supported two terms, shouldn't I also allow three or four or however many I wanted? And once I had more than two, wouldn't I also want to control order of operations with parentheses? And then nested parentheses? As the complexity ramped up, it would become increasingly difficult to protect against both logic and parsing errors. Recall that for *Subcutanean*, the stakes are high: any error in text rendering gets shipped off to a print-on-demand service and enshrined in a printed book. I could be as careful as I wanted, but inevitably, increasing complexity like this compounds the risk of mistakes slipping through.

Another option would be to allow nesting, maybe with a syntax like **[verbose> [angry>...]]**. But nesting without indentation, I've decided after trying this in multiple past projects, is just inevitably a bad idea. It's hard for both authors and code to parse and error-correct, and it allows all kinds of subtle errors to creep in where the syntax looks correct at a glance but certain possibility states are unaccounted for, or text is getting printed in situations it shouldn't. For my use case, with these random bits sprinkled in and among paragraphs of prose, using indentation instead would feel awkward, breaking up the flow of those paragraphs and adding concerns over when line breaks and spacing were code constructs versus literal output. I didn't want to go down this route, either.

Ultimately I went for a third option. Remember in my last post when I said I didn't want to add GOTOs or routines? Well... I added routines. I called them macros, but basically they jump to a label, print the text found there, and then return.

```
[MidnightWalk>{subject change} The walk down the hill was brisk...]
```

```
[MACRO subject change][drunk>Heads spinning, we spilled outside, raucous. |  
Wanting some fresh air, we stepped outside.]
```

This was definitely ugly in certain ways, but solved some problems. It let me reuse chunks of text in multiple variants, if for instance I had a moment in the middle of two different scenes that would be the same regardless of which one was chosen. It also allowed for elegant nesting: the contents of a macro can contain another macro with additional conditions, without an individual list of variants ever getting overly complicated. Without any further changes or additions, this syntax could also support most conditional logic I could think of: you can use it to emulate AND, OR, or NOT without much trouble, and nest things arbitrarily deeply. The flow of the prose is broken, of course — during editing, you might have to jump around the document to follow it, especially in longer variants where the macro definition might be pages away — but I still felt this was a better approach than the alternatives.

The biggest danger this change introduces is removing the context from the seams of the inserted text. If, for instance, I'd forgotten to include periods at the ends of each subject change expansion, I'd get malformed output like *Heads spinning, we spilled outside, raucous The walk down the hill...* . I'll talk in a future post about a separate tool I wrote to help with this problem. (*Update: and that one's here!*)

One final twist in the .quant syntax, also brought about from my unique use case: I wanted a way to preserve one particular rendering of the entire text, which was the first one I wrote.

Subcutanean actually began life as a traditional novel, and in fact had been through a whole gamut of drafts and revisions before I ever started thinking about quantum-izing it. As I began this new and more significant round of revisions, adding variant possibilities throughout, I wanted to reserve the right to change my mind and get back the “original” linear version at any point, even while continuing to edit and improve it alongside the alternatives: an escape hatch, if you will, in case the project turned out to be ill-conceived.

To achieve this, I adopted the convention that the first option in any list of variants would always be the original, “safe” version. I could then add a toggle for my renderer to consistently choose the first variant, and thus pull the original text back out of the

quantum soup any time I wanted. This convention was unambiguous except in the situation of single-option text which would either appear or not: here I needed a new symbol to indicate whether the original version said something, or nothing:

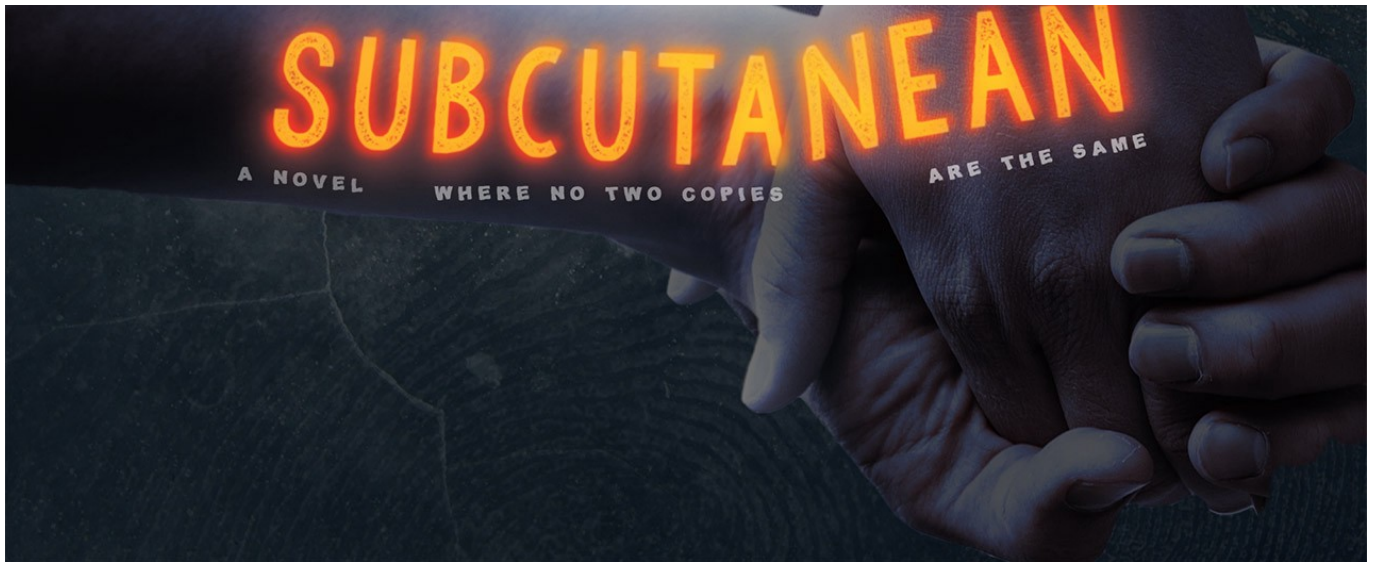
```
[DEFINE wordy]      # if "--original", never choose this  
[DEFINE wordy]      # if "--original", always choose this
```

The caret could also be used to mark a variant other than the first one as the new preferred version, in case I changed my mind about which one I liked better. In fact as the writing continued I found myself doing this more and more often. This soon evolved into a personal quality metric for the quantum version of *Subcutanean*: when revising and rewriting a list of alternatives, I knew I was done when I could no longer decide which one to mark as “best”: when I’d be equally proud to have any of them carry the story. As I reached this state with more and more of the writing, I began to believe my new vision of a multilinear *Subcutanean* — where I’d be completely happy sending off a random rendering, sight unseen, to a reader — could really work. The notion of needing to mark the “original” version became, in the end, fully vestigial.

I still had the problem of ensuring that every one of those millions of possible combinations of text would have perfect transitions, no misplaced punctuation, and no awkward joins between static and variant text. In a future post I’ll talk about the tool I wrote specifically to help solve that problem.

Subcutanean is an upcoming horror novel that changes with each new copy. Pre-order it now, or follow the project on Twitter, Facebook, or Goodreads.





[Interactive Fiction](#)

[Procedural Text](#)

[Programming](#)

[Subcutaneous](#)

[Indie Books](#)

[About](#) [Help](#) [Legal](#)