The Panini Project

Sam M W

December 26, 2014

1 What is this?

The Panini Project provides:

- A method of describing the grammar of a language
- A way of matching these rules against an input text, and conversely, a way to generate new texts.
- An ontology (a way of representing the world)

By providing these things, it becomes possible to write computer programs that treat language in some way. Of course, many applications already exist that (try to) process natural language, for example:

- Databases that are interrogable in (a subset of) plain English.
- Role-playing games that are instructed in (a subset of) plain English.
- Machine translation applications that take some input in Czech, for instance, and have it translated to Swahili?
- IME: take some Japanese input in latin letters and have it fill the Japanese scripts in

There have been many ad-hoc solutions to these problems until now. The Panini Project hopes to unify these solutions (there, is after all, only the one problem — it's the problem of parsing and generating natural language) and hopefully we'll come up with a way to process language that can be used in a wide variety of applications, but with the effort having only been made once.

2 What's in a name?

The Panini Project (two capital P's; the T has no capital, except where typology norms dictate otherwise) is named after Pāṇini, the Sanskrit grammarian from ancient India. I suggest that the name of the project will usually be spelled without the diacritics, since they are awkward to type on most computers.

Originally, this project was called the Tranny Project. Among my friends at university, this was the common abbreviation of the phrases "translation", "translation classes" and the like. Originally, this project was intended as a machine translation platform, and the name stuck, despite the generalisation of the project's scope.

Since those days, whenever I have mentioned this project in a public space, many people have been taken aback, or downright offended, since this word is also a shortening of the word "transgendered". It has been such a barrier to sensible discussion that I eventually renamed the project. Beware, though, that the word "Tranny", which has nothing to do with transgenderism, pops up here and there in the source code. It is being phased out, but means exactly the same thing as "Panini".

3 Architecture

The Panini Project consists of several parts.

There is a programming language called *panini*. It is Turing complete, but is particularly geared toward describing language. Depending on how you like to think, it can be seen as a beefed-up kind of Backus-Naur notation, or as a procedural language that forks with every single function call. There should be another PDF in this folder called programming.pdf which describes it in further detail.

There is a compiler that compiles any panini source code into a lower-level language. This second language is in turn interpreted at runtime. In this way, it works similarly to the Java programming language because it too is compiled into a bytecode, which is later interpreted.

There is a run-time library (these link to a computer program when it is being built) that provides routines for using this language in your own project, and the mechanisms to parse or generate a sentence in a given language, to interrogate the meaning etc. In the demos/ folder, you may find various programs that show how this run-time library is to be used.

4 Compile-time

This section describes what happens when you build the Panini Project and install it onto your computer. The enclosed script ./build.sh takes care of all this.

First up is to compile libpanini.a. This is the run-time library for processing installed Panini files and samples of language.

Next, we compile tc, a compiler. This thing takes panini source code and compiles it into a single file that can be read in and processed by libpanini.a.

The panini source code is partially written by hand, and this is mostly found in the languages/ directory, under each language. These describe the language. To know how these work, read programming.pdf.

There are also so-called ”importers” in the imports/ directory. These are short programs that take some input and uses it to generate panini source code. At the time of writing, there are two such things:

- *ekan*: Ekan is a program that takes the excellent and freely available edict and kanjidic by Jim Breen, and converts the dictionary definitions such that they may be used in the Panini Project.
- *pp*: pp, short for principles & parameters, learns a language, including its syntax and grammar, from the examples in a text file.