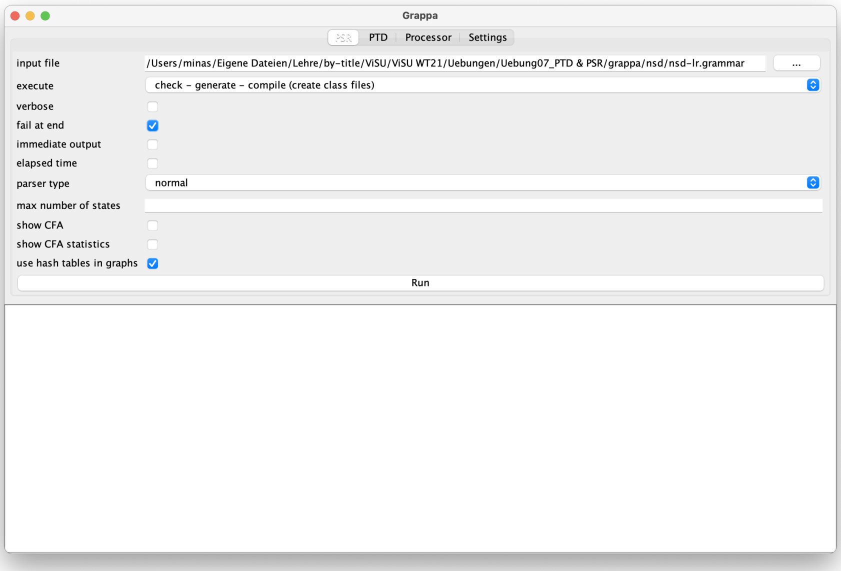
GRAPPA Mini-Tutorial

Mark Minas

February 26, 2021

Download the appropriate installer file (use `Grappa-`x`.exe` for Windows, `Grappa-`x`.dmg` for MacOS, or `grappa-`x`.deb` for Linux) and install GRAPPA. A corresponding shortcut is created on your Windows desktop and also on Linux (depending on your Linux distribution). On MacOS, you find GRAPPA in the usual Applications directory.

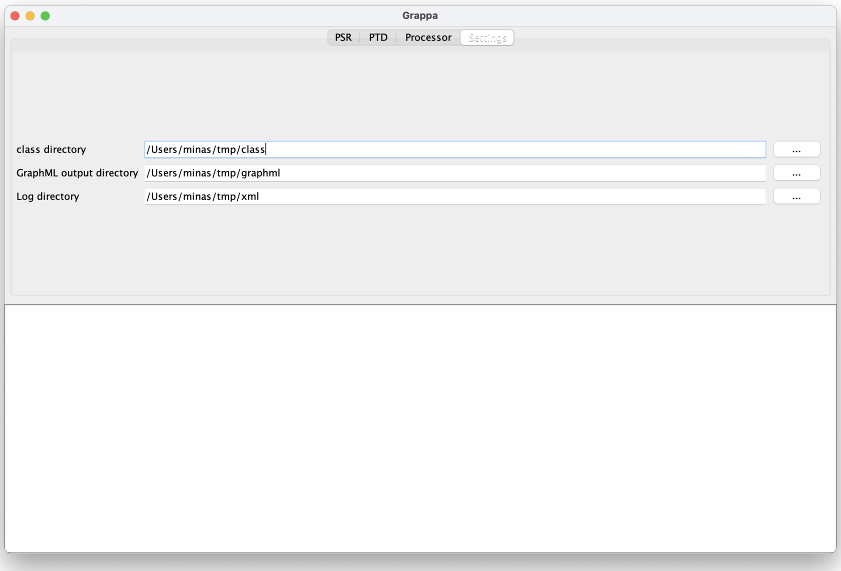


input file	/Users/minas/Eigene Dateien/Lehre/by-title/ViSU/ViSU WT21/Uebungen/Uebung07_PTD & PSR/grappa/nsd/nsd-lr.grammar
execute	check – generate – compile (create class files)
verbose	☐
fail at end	✓
immediate output	☐
elapsed time	☐
parser type	normal
max number of states	
show CFA	☐
show CFA statistics	☐
use hash tables in graphs	✓

Figure 1: GRAPPA screenshot showing the dialog of the PSR parser generator.

Start GRAPPA by double clicking on it. The program opens as shown in Fig. 1 (MacOS version). You can see the four tabs *PSR*, *PTD*, *Processor* and *Settings* at the top of the window. *PSR* and *PTD* contain the dialogs for controlling the PSR and the PTD parser generator. In Fig. 1 the dialog of the PSR Parser generator is shown. The tab *Processor* contains the dialog of the graph processor, which can analyze any input graph with a previously generated generated parser. The tab *Settings* provides access to a dialog for setting the directory where generated files are stored, etc. The lower part of the GRAPPA window shows the text output of the parser generators or the graph processor. In Fig. 1 this area is still empty.

Select the tab *Settings* and a suitable directory for *class directory* in this dialog (see Fig. 2). The button at the right end of the line opens a file selection dialog in the usual way. Generated parsers will be stored as .class files in this directory. It is recommended to create an empty directory directory for this purpose.



class directory /Users/minas/tmp/class
GraphML output directory /Users/minas/tmp/graphml
Log directory /Users/minas/tmp/xml

Figure 2: Grappa screenshot with the settings dialog.

If you also enter a directory for *GraphML output directory*, the generators and the graph processor store generated graphs in this directory. These graphs are mainly for illustration purposes. Since they are stored in GraphML format, you need the program yEd to layout and visualize them. yEd can be obtained for free from `https://www.yworks.com/products/yed/download`.

Download the the file `Grappa-mini-tutorial.zip` from the Grappa site and unpack it in a suitable directory. It contains two visual languages ($a^n b^n c^n$ as chain graphs and graphs representing Nassi-Shneiderman diagrams) with different grammars (.grammar files). Various .graph files are also included. They describe hypergraphs in textual form and can be analyzed by the graph processor using the generated parsers. An example hypergraph is represented as follows:

```
graph NSD2
   stmt(a,b,c,d)
   stmt(c,d,e,f)
end
```

in file `nsd_lr.graph` describes a graph consisting of two `stmt` edges that are attached to four nodes each, indicated by letters. In addition, chains of binary edges can also be formulated as follows:

```
n1 --a--> n2 --b--> n3 --c--> n4.
```

describes a graph with four nodes $n_1, \ldots, n_4$ and three edges between them, each marked a, b c, respectively.

The .grammar and .graph files each begin with an *options* section, in which the Java package or the qualified class name of the graph parser to be used must be specified. The corresponding parser class must have been previously generated with the PSR or PTD parser generator. Its qualified name is composed of the Java package specified in the .grammar file, the grammar name specified there, and the parser type. The latter is either PTD or PSR in its various forms (see below). The name of the generated class is reported in the textual output by the parser generator at the end of the generation process, e.g., "`Generated class nsd.NSD_lr_sPSRParser`".

Use the PTD parser generator to generate the parser for the $a^nb^nc^n$ language in the file `abc1.grammar`. You see that parser type "simple" is not sufficient, but "Parikh based" must be used. With the first variant, the parser selects a production for expanding a non-terminal based only on the next terminal edge of the input graph whereas the "Parikh based" parser potentially considers the entire remaining ("unread") input graph.

Check that the .class files for the Java class `abc.ABC1PTDParser` and others have been created in the directory you configured.

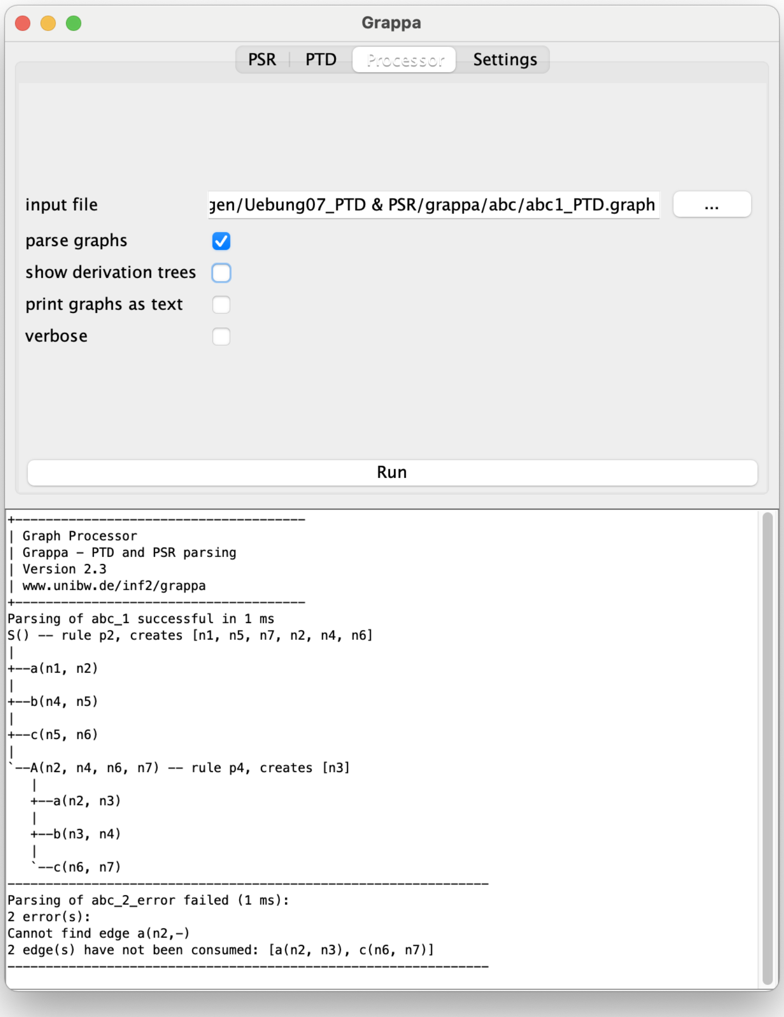
Parse the hypergraphs described in the `abc1_PTD.graph` file using the previously generated parser. The parser is applied to both of the included graphs. However, only the first of them is correct with respect to the language. For the second one, an error message is returned.

In order to visualize the derivation tree of the correct input graph in text form, you must select the option *show derivation trees*. Fig. 3 shows the corresponding output of the graph processor. If you have also configured a directory for GraphML output in the *Settings* dialog, you will find the same tree in the file `abc_1-dag.graphml` (If you have set the *verbose* option, this file name will also be shown in the text output).

The grammar specified in the file `abc1.grammar` cannot be processed by the PSR parser generator. See the output of the generator to find out why. However, the PSR generator can be applied to the file `abc2.grammar`.

Use the parser type "normal" for the PSR parser generator and then the file `abc2_sPSR.graph` to try the generated parser.

The `nsd` directory contains two hyperedge replacement grammars for Nassi-Shneiderman graphs. `nsd-lr.grammar` contains a left recursive grammar, for which no PTD parser exists. However, a PSR parser does exist. Interpret the respective text outputs of the parser generators. The problem can be solved with so-called *merging* rules, analogous to ε-rules for LL(k) grammars. A corresponding grammar can be found in the file `nsd-merging.grammar`, for which both a PTD and a PSR parser can be generated. For the PSR parser, the parser



input file: jen/Uebung07_PTD & PSR/grappa/abc/abc1_PTD.graph

parse graphs [✓]

show derivation trees [ ]

print graphs as text [ ]

verbose [ ]

```
+---------------------------------------
| Graph Processor
| Grappa - PTD and PSR parsing
| Version 2.3
| www.unibw.de/inf2/grappa
+---------------------------------------
Parsing of abc_1 successful in 1 ms
S() -- rule p2, creates [n1, n5, n7, n2, n4, n6]
|
+--a(n1, n2)
|
+--b(n4, n5)
|
+--c(n5, n6)
|
`--A(n2, n4, n6, n7) -- rule p4, creates [n3]
   |
   +--a(n2, n3)
   |
   +--b(n3, n4)
   |
   `--c(n6, n7)
----------------------------------------------------------------
Parsing of abc_2_error failed (1 ms):
2 error(s):
Cannot find edge a(n2,-)
2 edge(s) have not been consumed: [a(n2, n3), c(n6, n7)]
----------------------------------------------------------------
```

Figure 3: Grappa screenshot showing the graph processor dialog.

generator first converts the grammar into an equivalent grammar without *merging* rules ("cleaned grammar").

Apply the generated parsers to the hypergraphs in the corresponding .graph files.