

# ppp Documentation

Daniel Bruder

Version 0.6.0

## Contents

<b>1</b>	<b>Abstract</b>	<b>3</b>
<b>2</b>	<b>General usage</b>	<b>3</b>
2.1	General Renderers . . . . .	3
2.2	General Options . . . . .	3
<b>3</b>	<b>ditaa Diagrams</b>	<b>4</b>
3.1	ditaa Options . . . . .	4
3.2	ditaa Examples . . . . .	4
<b>4</b>	<b>dot Diagrams</b>	<b>6</b>
4.1	dot Options . . . . .	6
4.2	dot Examples . . . . .	6
<b>5</b>	<b>neato Diagrams</b>	<b>7</b>
5.1	neato Options . . . . .	7
5.2	neato Examples . . . . .	7
<b>6</b>	<b>yUML</b>	<b>8</b>
6.1	yUML Options . . . . .	8
6.2	yUML Examples . . . . .	8
6.2.1	yUML Class diagrams . . . . .	8
6.2.2	yuml Usecase diagrams . . . . .	10
6.2.3	yuml Activity diagrams . . . . .	11

<b>7</b>	<b>rdfdot Diagrams</b>	<b>12</b>
7.1	rdfdot Options . . . . .	12
7.2	rdfdot Examples . . . . .	12
<b>8</b>	<b>List of options</b>	<b>13</b>
<b>9</b>	<b>List of homepages and documentation to renderers</b>	<b>14</b>
<b>10</b>	<b>Credits and further references</b>	<b>14</b>

## 1 Abstract

*ppp* allows you to use pandoc in new ways by rendering ASCII-markup to beautiful pictures right from within pandoc's verbatim environments.

See below for illustrative examples and detailed usage instructions.

Bonus on top: Leaving out *ppp* from the typesetting pipeline will still render your document and the verbatims with the ASCII-markup will still stay readable!

---

## 2 General usage

In each case, you will use pandoc's verbatim environment, set the rendering engine and additional options:

```
~~~~~ {.renderer .option1 .option2=value2}
--- RENDERER-SPECIFIC MARKUP GOES HERE ---
~~~~~
```

### 2.1 General Renderers

The renderers available to *ppp* are:

- ditaa
- yuml diagrams:
  - class diagrams (cf. Figure 5)
  - usecase diagrams (cf. Figure 6)
  - activity diagrams (cf. Figure 7)
- dot
- neato
- rfdot

### 2.2 General Options

This is a list of the general options, compatible with any type of renderer:

- `.scale=90%`
  - `.label=fig:my-figure`
  - `.title="Some label for the figure"`
-

## 3 ditaa Diagrams

In order to generate ditaa-diagrams, ditaa needs to be installed.

For an exhaustive list of options and possibilities, please check the [ditaa home-page](#).

### 3.1 ditaa Options

Apart from the [General Options](#), the possible options specific to ditaa are:

- `.rounded-corners`
- `.no-shadows`
- `.no-antialias`
- `.no-separation`

### 3.2 ditaa Examples

Using ditaa, the following markup will produce Figure 1.

```
~~~~~ {.ditaa .rounded-corners .no-shadows
      .scale=90% .title="The ppp and pandoc pipeline"
      .label=fig:pipeline-overview .no-antialias .no-separation
      } # Caution! These lines actually would have to be on *one* line only!
+-----+ +-----+ +-----+
| markdown source |----->| ppp |----->| processed markdown |
+-----+ +-----+ +-----+
                        |
                        | \--->| image files |
                        +-----+
                        |
                        | diagram creation |
                        +-----+
                        | ditaa/dot/rdfdot |
                        +-----+
~~~~~
```

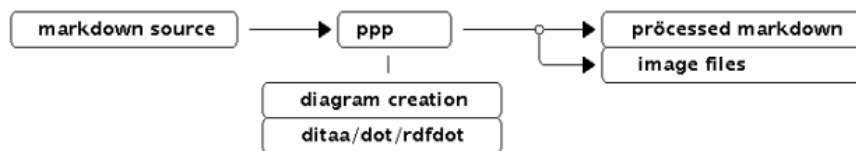


Figure 1: “The ppp and pandoc pipeline”

As a contrast, turning off several options, ditaa will produce an output as in Figure 2:

```

~~~~~ {.ditaa .scale=90% .title="The ppp and pandoc pipeline #2" .label=fig:pipeline-overview-2}
+-----+ +-----+ +-----+
| markdown source |----->| ppp |-----*-->| processed markdown |
+-----+ +-----+ +-----+
|                                     | \--->| image files |
+-----+ +-----+
| diagram creation |
+-----+
| ditaa/dot/rdfdot |
+-----+
~~~~~

```

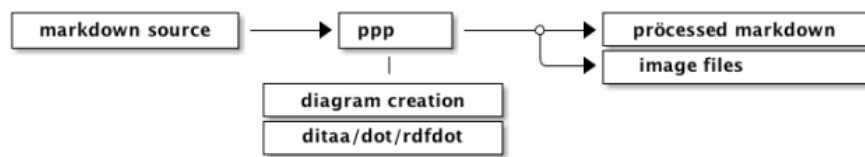


Figure 2: “The ppp and pandoc pipeline #2”

## 4 dot Diagrams

dot rendering is done through [GraphViz's](#) engine. Please cf. [Graphviz's Documentation](#) for exact usage specifics on the usage of dot.

### 4.1 dot Options

- currently none apart from the [General Options](#)

### 4.2 dot Examples

With dot as the *renderer*, the following markup produces the figure as seen in [Figure 3](#).

```
~~~~~ {.dot .scale=50% .title=dot Finite State Automaton .label=fig:dot-fsa}
digraph finite_state_machine {
    rankdir=LR;
    size="8,5"
    node [shape = doublecircle]; LR_0 LR_3 LR_4 LR_8;
    node [shape = circle];
    LR_0 -> LR_2 [ label = "SS(B)" ];
    LR_0 -> LR_1 [ label = "SS(S)" ];
    LR_1 -> LR_3 [ label = "S($end)" ];
    LR_2 -> LR_6 [ label = "SS(b)" ];
    LR_2 -> LR_5 [ label = "SS(a)" ];
    LR_2 -> LR_4 [ label = "S(A)" ];
    LR_5 -> LR_7 [ label = "S(b)" ];
    LR_5 -> LR_5 [ label = "S(a)" ];
    LR_6 -> LR_6 [ label = "S(b)" ];
    LR_6 -> LR_5 [ label = "S(a)" ];
    LR_7 -> LR_8 [ label = "S(b)" ];
    LR_7 -> LR_5 [ label = "S(a)" ];
    LR_8 -> LR_6 [ label = "S(b)" ];
    LR_8 -> LR_5 [ label = "S(a)" ];
}
~~~~~
```

---

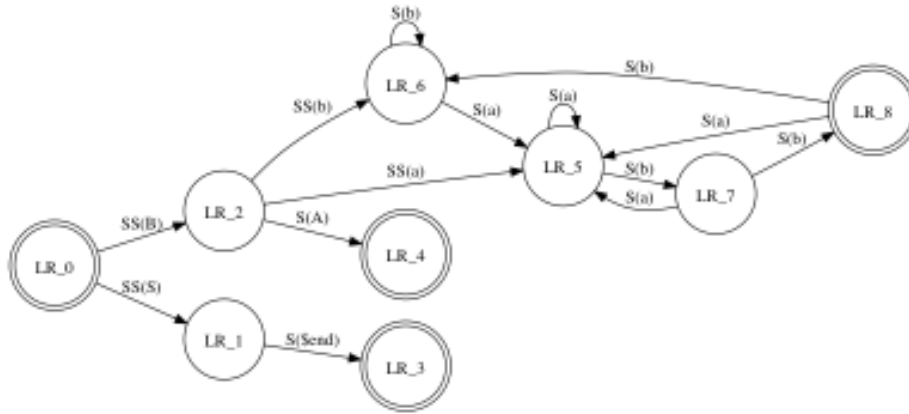


Figure 3: dot Finite State Automaton

## 5 neato Diagrams

neato diagrams behave very similarly to [dot Diagrams](#). Please cf [dot Diagrams](#) for more information

### 5.1 neato Options

- same as [dot Options](#)

### 5.2 neato Examples

The following example produces [Figure 4](#).

```
~~~~~ {.neato .scale=50% .title=neato diagram .label=fig:neato-diagram}
graph G {
  n0 -- n1 -- n2 -- n3 -- n0;
}
~~~~~
```

---

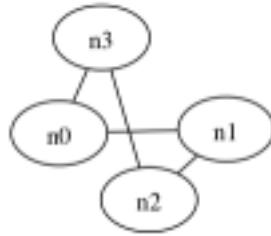


Figure 4: neato diagram

## 6 yUML

yUML needs a network connection and uses <http://yuml.me> as the rendering service.

### 6.1 yUML Options

Options specific to yUML can be:

- `.type=:` any of [ class, activity, usecase ]
- `.style=:` any of [ scruffy, boring, plain ]
- `.direction=:` any of [ LR, RL, TD, ]

### 6.2 yUML Examples

#### 6.2.1 yUML Class diagrams

With *yUML* as the renderer, setting `.type=class` and using the style `.style=boring`, the following markup produces Figure 5.

```

~~~~ { .yuml .style=boring .type=class .direction=TD .title=yUML class diagram .label=fig:yuml-class-diagram }

[Customer] +1 ->          * [Order]
[Order]    ++1 -items>    * [LineItem]
[Order]    -0..1>         [PaymentMethod]
~~~~

```



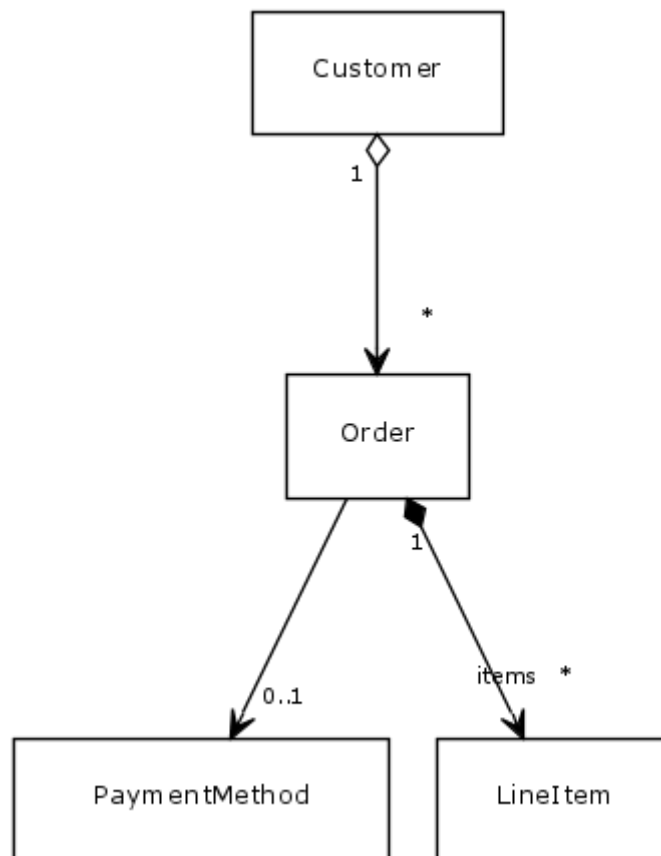


Figure 5: yUML class diagram

### 6.2.2 yuml Usecase diagrams

With `scruffy` style and `.type=usecase`, the following example produces Figure 6.

```
~~~~ {yuml .style=scruffy .type=usecase .title=yUML usecase diagram .label=fig:yuml-usecase-diagram}  
// Cool Use Case Diagram  
[Customer]-(Make Cup of Tea)  
(Make Cup of Tea)<(Add Milk)  
(Make Cup of Tea)>(Add Tea Bag)  
~~~~
```

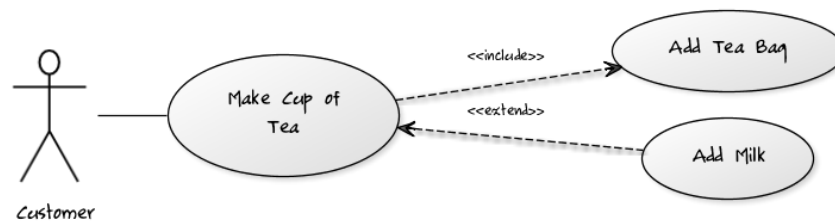


Figure 6: yUML usecase diagram

### 6.2.3 yuml Activity diagrams

Lastly, using `.type=activity` and `.style=plain` the following example produces Figure 7.

```
~~~~ {.yuml .style=plain .type=activity .title=yUML activity Diagram .label=fig:yuml-activity-diagram}
(start)->|a|,|a|->(Make Coffee)->|b|,|a|->(Make Breakfast)->|b|,|b|-><c>[want more coffee]->(Make Coffee),<c>[satisfied]->(end)
~~~~
```

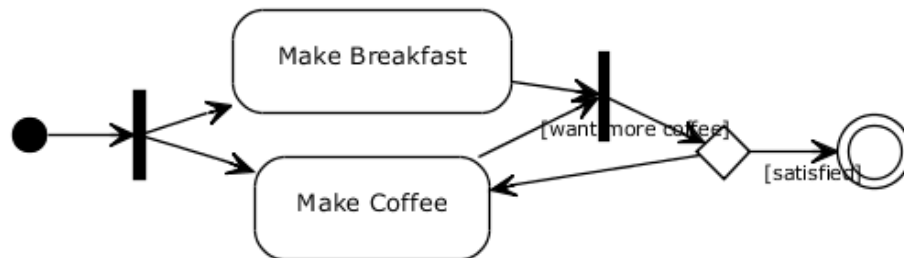


Figure 7: yUML activity Diagram

## 7 rdfdot Diagrams

### 7.1 rdfdot Options

- currently none apart from the [General Options](#)

### 7.2 rdfdot Examples

The following example produces Figure 8 on page 12.

```
~~~~~ {.rdfdot .scale=65% .title=rdfdot Diagram .label="fig:rdfdot-diagram"}  
@prefix foaf: <http://xmlns.com/foaf/0.1/> .  
@base <http://example.com/> .  
<alice> foaf:name "Alice" ;  
        foaf:knows [ foaf:name "Bob" ] .  
~~~~~
```

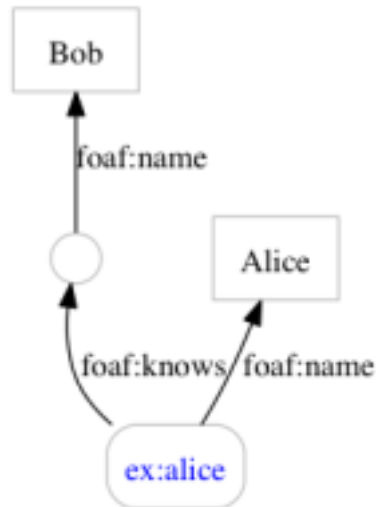


Figure 8: rdfdot Diagram

## 8 List of options

Renderer	Option	possible values
<i>General</i>	<code>.scale</code>	1%-99%
	<code>.label</code>	<code>fig:my-figure</code>
	<code>.title</code>	"Some label for the figure"
ditaa	<code>.rounded-corners</code>	
	<code>.no-shadows</code>	
	<code>.no-antialias</code>	
	<code>.no-separation</code>	
dot	N/A	
neato	N/A	
yUML	<code>.type=</code>	any of [ class, activity, usecase ]
	<code>.style=</code>	any of [ scruffy, boring, plain ]
	<code>.direction=</code>	any of [ LR, RL, TD, ]
rdfdot	N/A	

Table 1: List of options

## 9 List of homepages and documentation to renderers

Renderer	Links
ppp	(this document) <a href="https://metacpan.org/release/App-pandoc-preprocess">https://metacpan.org/release/App-pandoc-preprocess</a> <a href="https://github.com/xdbr/p5-App-pandoc-preprocess">https://github.com/xdbr/p5-App-pandoc-preprocess</a>
ditaa	<a href="http://ditaa.sourceforge.net/">http://ditaa.sourceforge.net/</a>
dot	<a href="http://www.graphviz.org/">http://www.graphviz.org/</a>
neato	<a href="http://www.graphviz.org/">http://www.graphviz.org/</a>
yUML	<a href="http://yuml.me/">http://yuml.me/</a> <a href="https://github.com/wandernauta/yuml">https://github.com/wandernauta/yuml</a>
rdfdot	<a href="https://metacpan.org/pod/RDF::Trine::Exporter::GraphViz">https://metacpan.org/pod/RDF::Trine::Exporter::GraphViz</a>

Table 2: List of options

## 10 Credits and further references

- <http://www.asciiflow.com/#Draw>: an excellent helper for all things diagram
- <http://randomdeterminism.wordpress.com/2012/06/01/how-i-stopped-worrying-and-started-using-markdown/>: general introduction to another approach to typesetting and using `gpp`
- <https://github.com/nichtich/ditaa-markdown>: This is where the original idea came from
- `gpp`: <http://files.nothingisreal.com/software/gpp/gpp.html>

## List of Figures

1	“The ppp and pandoc pipeline” . . . . .	4
2	“The ppp and pandoc pipeline #2” . . . . .	5
3	dot Finite State Automaton . . . . .	7
4	neato diagram . . . . .	8
5	yUML class diagram . . . . .	9

6	yUML usecase diagram . . . . .	10
7	yUML activity Diagram . . . . .	11
8	rdfdot Diagram . . . . .	12

---