

Arborly

A library for producing beautiful syntax tree graphs.

Version 0.2.0

2025-03-13

MIT

Table of Contents

I Usage	2
I.1 Importing the Package	2
I.2 Building a Syntax Tree	2
II Arguments	3
III Examples	5
III.1 The Quick Brown Fox	6
III.2 Sphinx of Black Quartz	7
III.3 Math Nodes	8
III.4 Content Nodes	9
III.5 Dense	9
III.6 Long Sentence	10

Part I Usage

I.1 Importing the Package

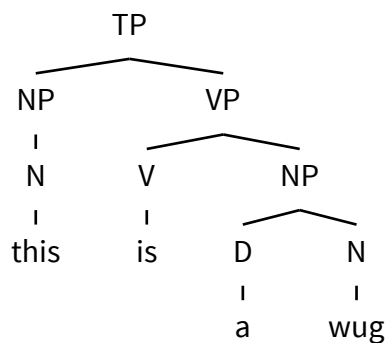
```
#import "@preview/arborly:0.2.0"
```

I.2 Building a Syntax Tree

Each node in the tree is an array of a label and a child node array, with the exception of terminal nodes, which contain content instead.

Be aware that the trailing commas after lone children are *not* optional, as otherwise `("NP", (("N", "this")))` parses as `("NP", ("N", "this"))`. See examples for more supported syntax.

```
#let tree = ("TP", (  
  ("NP", (  
    ("N", "this"),  
  )),  
  ("VP", (  
    ("V", "is"),  
    ("NP", (  
      ("D", "a"),  
      ("N", "wug"),  
    )),  
  )),  
))  
  
#arborly.tree(tree, min-gap-x: 1)
```



Part II Arguments

```
#tree(  
  {node},  
  {hug-bottom}: false,  
  {min-space-y}: 1.0,  
  {min-gap-x}: 0.3,  
  {min-slope}: 0.0,  
  {full-step-y}: false,  
  {label-alignment}: "middle"
```

) → **content**

A function which takes a hierarchy of syntax elements and displays a tree of them.

Argument

{node}

array

The root node of the syntax tree. See the examples for its structure.

Argument

{hug-bottom}: false

boolean

Draws all words at the bottom in a line, with stems extending down from their parent. It does not function correctly with a non-default min-slope.

Argument

{min-space-y}: 1.0

float

The vertical separation of levels of the tree. It may be higher than this if min-slope is set.

Argument

{min-gap-x}: 0.3

float

The minimum horizontal gap between terminal nodes, which applies even if they are not vertically aligned.

Argument

{min-slope}: 0.0

float

Sets an approximate lower bound for the slope of the connecting lines. Setting it to a non-zero value will change the vertical spacing of the graph. 0.2 - 0.5 has worked well for me, but this can be any number.

Argument

{full-step-y}: false

boolean

To space tree layers by increments of min-space-y when using a min-slope. This prevents layers from getting “out of sync” and having words that are slightly out of alignment with each other.

Argument

`{label-alignment}: "middle"`

string

This accepts one of three strings: “middle”, “average”, or “smart”.

Middle places a parent label exactly between its outer children’s labels (the ones furthest to the left and right)

Average places it at the average of its children’s horizontal positions, which can look better when there are many clumped close together and only one off to the side.

Smart acts like average, but if there are three children and the second label is sufficiently close to the middle, the parent will “snap” to it.

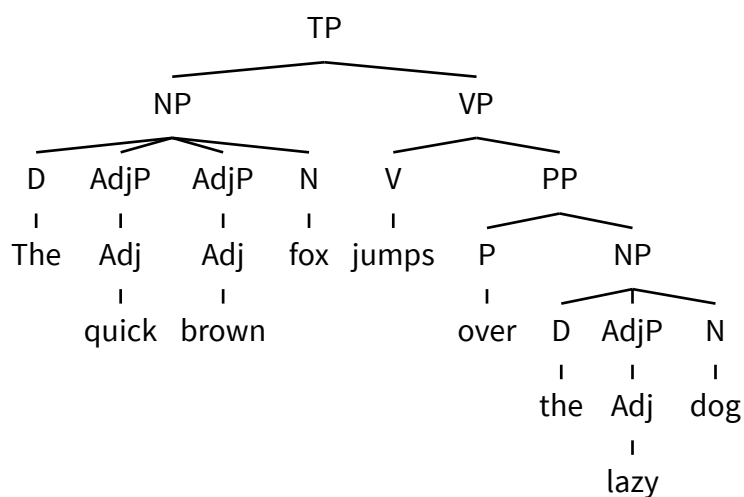
Part III Examples

Note that I am not a linguist, so these analyses may be wrong.

III.1 The Quick Brown Fox

```
#let tree = ("TP", (
  ("NP", (
    ("D", "The"),
    ("AdjP", (
      ("Adj", ("quick")),
    )),
    ("AdjP", (
      ("Adj", ("brown")),
    )),
    ("N", "fox"),
  )),
  ("VP", (
    ("V", "jumps"),
    ("PP", (
      ("P", "over"),
      ("NP", (
        ("D", "the"),
        ("AdjP", (
          ("Adj", "lazy"),
        )),
        ("N", "dog"),
      )),
    )),
  )),
))

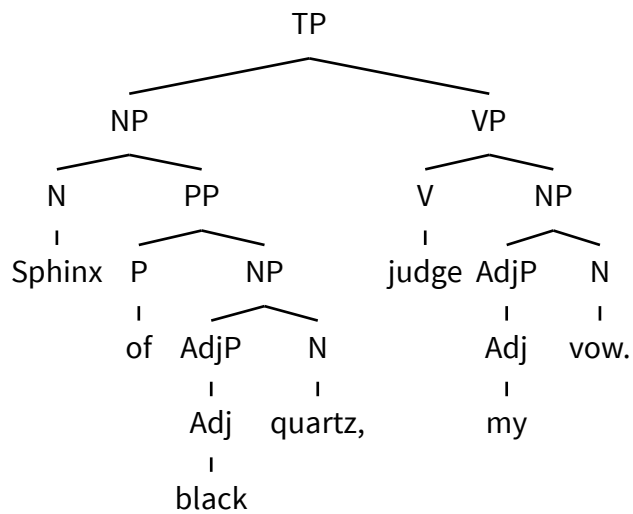
#arborly.tree(tree, label-alignment: "smart")
```



III.2 Sphinx of Black Quartz

```
#let tree = ("TP", (
  ("NP", (
    ("N", "Sphinx"),
    ("PP", (
      ("P", "of"),
      ("NP", (
        ("AdjP", (
          ("Adj", "black"),
        )),
        ("N", "quartz,")
      )),
    )),
  )),
  ("VP", (
    ("V", "judge"),
    ("NP", (
      ("AdjP", (
        ("Adj", "my"),
      )),
      ("N", "vow."),
    )),
  )),
))

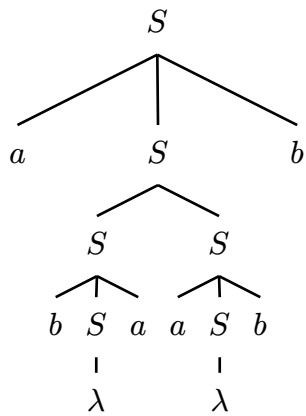
#arborly.tree(tree, min-slope: 0.2)
```



III.3 Math Nodes

```
#let tree = ($S$, (
  $a$,
  ($S$, (
    ($S$, (
      $b$,
      ($S$, $lambda$),
      $a$,
    )),
    ($S$, (
      $a$,
      ($S$, $lambda$),
      $b$,
    )),
  )),
  $b$,
))

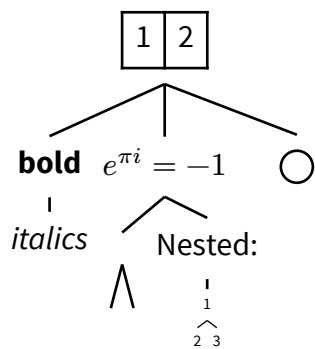
#arborly.tree(tree, min-slope: 0.5)
```



III.4 Content Nodes

```
#let tree = (table(columns: 2, [1],[2]), (
  ([*bold*], [_italics_]),
  ($e^{pi i} = -1$, (
    ([], ({}, none)),
    ([Nested:], pad(-40%, scale(
      50%,
      reflow: true,
      arborly.tree([1], ([2], [3]))
    ))),
  )),
  circle(radius: 0.5em),
))

#arborly.tree(tree, min-slope: 0.5, label-alignment: "smart")
```



III.5 Dense

```
#let tree = ("1234", (
  ("12", ("1", "2")),
  ("34", ("3", "4")),
))

#arborly.tree(tree, min-gap-x: 0)
```

```
1234
 ^
1234
 ^ ^
1234
```

III.6 Long Sentence

[illegible]