ADT Style File Documentation

v1.5

Patrick Schweitzer*

9. July, 2013

Contents

*Developer: Patrick.Schweitzer@uni.lu
(c) 2012–2013 University of Luxembourg — Interdisciplinary Centre for Security Reliability and Trust (SnT)

Abstract

This is the documentation of the ADT style file. It illustrates the available commands of the ADT style file with the help of numerous examples.

Summary:

- The ADT style file can be used to draw visually appealing attack–defense trees.
- It is a collection of commands for the PSTricks pstree package.
- The layout and the coloring is configured with the help of node commands.
- Using the command

  `\pstree[OPTIONS]{ROOT}{CHILD 1 CHILD 2 ... CHILD N}`

  a tree structure is set up.
  - `ROOT` has to be one of the node commands explained in Section 3.
  - `CHILD i` can either be an instance of the `\pstree` command or one of the node commands explained in Section 3.
- The size of the nodes and the length of the arcs are customizable either with a global or a local option.

1 Introduction to ADTrees

This package is intended to help typeset visually appealing attack–defense trees (ADTrees) for attack–defense scenarios. We first illustrate the usage of ADTrees on a toy example before elaborating on the commands that are available for typesetting them. For a general introduction to ADTrees, please refer to one of the papers available at `http://satoss.uni.lu/members/barbara/publications.php`.

Imagine the following (incomplete) scenario. In order to protect a building, a security guard is employed. In order to do his job, the guard has to protect against possible intruders. We exemplify three situations: An attacker that want to access a building can bribe the guard, subdued the guard or steal the guard's keys. To subdue the guard, the attacker could hire a colleague such that the guard is outnumbered. This allows one of the attackers to use a weapon to keep the guard from interfering. To protect against these three attacks, video cameras with remote surveillance capabilities could be employed. These, in turn, could be sabotaged by an attacker.

Based on the scenario, we setup a tree that helps us answer the question "How can a security guard be defeated?" The ADTrees is illustrated in Figure 1. The figure is produced by the following LaTeX code.

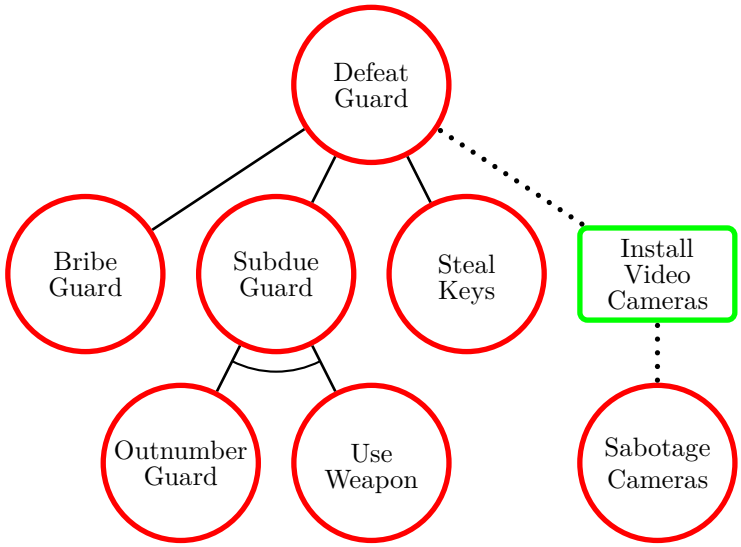


Figure 1: An ADTree for defeating a guard.

```
\settowidth{\pbs}{Outnumber}
\pstree[levelsep=2.5cm,treesep=0.4cm]{\NodeAC{Defeat\\Guard}}{
        \NodeA{Bribe\\Guard}
        \pstree{\NNodeA{OV}{Subdue\\Guard}}{
                \NNodeA{OV2}{Outnumber\\Guard}
                \NNodeA{OV3}{Use\\Weapon}
        }
        \NodeA{Steal\\Keys}
        \pstree{\NodeBC{Install\\Video\\Cameras}}{
                \NodeAC{Sabotage\\Cameras}
        }
}
\Arc{OV}{OV2}{OV3}
```

2 Usage of the ADT Style File

The ADT style file provides predefined node commands that guarantee the compliance with the layout for ADTrees. It is built upon the pstree package, which is part of PSTricks . Any pstree command can be used to further modify the layout of the tree. Pstree commands are (with two exceptions) not detailed in this documentation.[1]

Every ADTree is build up using the following structure:

```
PREAMBLE
\pstree[OPTIONS]{ROOT}{
        CHILD 1
        CHILD 2
        ...
        CHILD N
}
```

[1] A PSTricks documentation is available at `http://www.tug.org/PSTricks/main.cgi/`.

2.1 PREAMBLE

The preamble consists of two commands that govern the layout of a tree:

- The command `\pbs` is used to adjust the size of a node.
- The command `\corner` is used to adjust the radius of the rounded corner for defensive rectangles.

Both, `\pbs` and `\corner` are two LaTeX length variables. When used in the preamble, they globally define the size of the nodes and the radius of the corners of the rectangular defense nodes. They can either be set by providing a specific length (`\setlength`) or by providing a string of characters (`\settowidth`). In the latter case, LaTeX computes the length of the string and uses the corresponding length.

The default values are 1.5 cm for the size of a node and 0.1 cm for the radius of the corners of the defensive rectangles. The following commands illustrate how to change the sizes:

```
\setlength{\pbs}{2.5cm}
\settowidth{\pbs}{Something}

\setlength{\corner}{0.2cm}
\settowidth{\corner}{.}
```

Note that only normal and small nodes are affected by the length variable `\pbs`. Adjustable nodes are not affected.

2.2 OPTIONS

All options available to the pstree package can be used to configure a tree. There are two main parameters to change the layout of the trees. These are:

- `levelsep`
- `treesep`

They are used as LaTeX options with a specified length. The option `levelsep` is used to configure the vertical distance between tree levels and the option `treesep` is the horizontal distance between two children.

For example: `\pstree[levelsep=2.5cm,treesep=0.4cm]`

2.3 ROOT

The root has to be a node command. These commands are detailed in Section 3.

2.4 CHILD

A child can either be an instance of the `\pstree` command or a node command, detailed in Section 3.

3 The Node Commands

This section details the available node commands. All node commands follow a similar structure.

3.1 The Node Commands Templates

To construct a node command, one of the following three expressions is used: `\*Node+`, `\*Nodesmall+` or `\*Node+a`, where (`*`) denotes an optional prefix and (`+`) one or more suffixes. The three commands are used to distinguish between different sizes: a regular size, a small size and an adjustable size. To complete a command, the following prefix and suffix options are available:

- The suffix `A` is used to denote attack nodes, drawn as red circles, see Section 5.1.
- The suffix `B` is used to denote defense nodes, drawn as green rectangles, see Section 5.1.
- The suffix `C` is used to denote countermeasure nodes, drawn as dotted line, see Section 5.1.
- The suffix `T` is used to draw a node with a subtree below it, see Section 5.1.
- The prefix `N` gives the node an identifier. This identifier is only needed when drawing arcs to symbolize a conjunctive refinement, see Section 5.2 and 5.3.

One of the two suffixes `A` or `B` is mandatory. It indicates that the node is either an attack node (`A`) or a defense node (`B`). The suffixes `C` and `T` as well as the prefix `N` are optional.

The number of arguments of a node command depends on the type of the node and the added pre- and suffixes. The two commands `\*Node+` and `\*Nodesmall+` have one argument that represents the label of the corresponding node. The command `\*Node+a` has two arguments, representing the size and the label of the node. The prefix `N` adds another argument as identifier. Moreover, the suffix `T` adds an argument as label for the subtree. In case `T` is added to a `\*Node+a` command, it adds two additional arguments denoting the size and the label of the subtree.

The argument order is the following. Arguments are dropped if the option is not included in the command.

1. Identifier (`N`)
2. Size of node (`*Node+a`)
3. Size of subtree (`T`)
4. Label
5. Label label of subtree (`*Node+a` together with `T`)

The basic commands that do not contain an identifier are illustrated in Section 5.1, ones with an identifier are illustrated in Section 5.2 and 5.3.

3.2 Layout of Small Nodes

Note that small size nodes are meant to be used in conjunction with regular nodes and are intended to only contain only one symbol. If the node is too small, some text will not be contained inside the node. For defensive nodes this text is not centered. This is due to the LaTeX command `\parbox`. To circumvent this problem, the size of the node should be enlarged via the length `\pbs` or adjustable nodes should be used.

3.3 Identifiers and Arcs

Identifiers (IDs) are used to draw arcs, indicating conjunctive refinements. There are three `\Arc` commands available:

```
\Arc{Parent ID}{Child 1 ID}{Child 2 ID}
\Arclhs{Parent ID}{Child 1 ID}{Child 2 ID}
\Arcrhs{Parent ID}{Child 1 ID}{Child 2 ID}
```

Every argument of an `\Arc` command is an identifier of a node. `\Arc` is used to draw an arc in quadrant III and IV if a Cartesian coordinate system. `\Arclhs` is used to draw an arc in quadrant III and `\Arcrhs` is used to draw an arc in quadrant IV. The arcs are visualized in Section 5.3.

In other words, the three arguments of an `\Arc` command are meant to be a parent node and two child nodes. An arc is supposed to connect the edge between the parent and the first child node and the edge between the parent and the second child node. The regular version (`\Arc`) has to be used if the center of the first child node is left of the center of the parent and the center of the second child of the center of the parent node. If the centers of both children are left of the center of the parent, then `\Arclhs` version has to be used, if the centers of both children are to the right of the center of the parent node, the command `\Arcrhs` has to be user.

Arcs also exist in a small version (`\Arcsmall`) and in an adjustable version (`\Arca`). In the latter case an additional (first) argument specifies the length of the arcs.

3.4 Legend

To draw a legend for ADTrees, the following symbols can be used:

- ◯ or ◯, typed as `$\A$` or `$\ACC$`
- □ or □, typed as `$\D$` or `$\DCC$`
- △ or △, typed as `$\T$` or `$\TCC$`

A full legend is provided as the following example. Unwanted lines can be removed and the placement can be adjusted using the `\rput` command.

```
\rput(-1,-1.5){
\begin{tabular}{|cl|}
\hline
\hspace{1ex}$\A$ & attack node\\
\hspace{1ex}$\ACC$ & attack node, thicker lines\\
\hspace{1ex}$\D$ & defense node\\
\hspace{1ex}$\DCC$ & defense node, thicker lines\\
\hspace{1ex}$\T$ & subtree\\
\hspace{1ex}$\TCC$ & subtree, thicker lines\\
\hspace{2ex}\psline[linewidth=0.5pt](0,.3)(-.3,0)
        \psline[linewidth=0.5pt](0,.3)(-.1,0)
        \psline[linewidth=0.5pt](0,.3)(.1,0)
        & disjunctive refinement\\
\hspace{2ex}\psline[linewidth=0.5pt](0,.3)(-.3,0)
        \psline[linewidth=0.5pt](0,.3)(-.1,0)
        \psline[linewidth=0.5pt](0,.3)(.1,0)
        \psarc[linewidth=0.5pt](0,.3){0.2cm}{225}{288}
```

```
        & conjunctive refinement\\
\hspace{1ex}\psline[linewidth=1pt,linestyle=dotted](0,.25)(0,-.05)
        & countermeasure\\
\hline
\end{tabular}
}
```

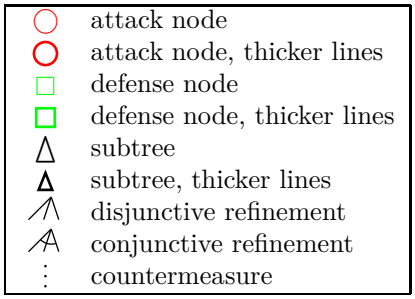


Figure 2: An ADTree for RFID DoS

3.5 Miscellaneous Nodes

Undocumented examples of several other nodes types are provided in Sections 5.4 and 5.5.

4 Textual Syntax

The ADT style files also comes equipped with a visually more appealing version of the textual syntax for ADTrees. We first explain what the textual syntax is.

4.1 Introduction to the Textual Syntax

The textual syntax is a way to represent ADTrees as a linear ASCII description. It does not require LaTeX. It can be employed for quick and easy display of ADTrees, for instance in an email exchange. For this purpose the following ASCII characters are used:

```
( ) [ ] | - = #.
```

To depict the type of a node we use the brackets. More specifically, the label of an attack node is enclosed between round brackets "`( )`", the label of a defense node in square brackets "`[ ]`". The "`|`" symbol helps to depict the structure of the tree and the remaining symbols are used to express the connection between a parent node and its children. Each node is written on its own line adhering to the following structure.

The first line corresponds to the root node and contains only the label and the type of the root. All other lines correspond to intermediate nodes or leaves. They start with a number of "`|`" symbols, followed by "`-`", "`=`" or "`#`" and end with the label of the node enclosed in brackets. On each such line the depth of the node in the tree is reflected by the amount of "`|`" symbols. The symbols "`-`", "`=`" and "`#`" illustrate that this node is linked to its parent via a disjunctive refinement, a conjunctive refinement or a countermeasure, respectively.

The vertical structure of the ASCII description illustrates the parent–child relation between the nodes. The tree is set up from top to bottom and from left to right.

Children of nodes are considered before siblings. The structure is created recursively. First, the line corresponding to the root is created. Then, for every parent node, lines corresponding to its children are inserted underneath the line corresponding to the parent. This construction is repeated until we reach lines corresponding to leaves.

The linear ASCII syntax for the ADTree from Example 1 is given in Figure 3.

```
(Defeat Guard)
|-(Bribe Guard)
|-(Subdue Guard)
||=(Outnumber Guard)
||=(Use Weapon)
|-(Steal Key)
|#[Install Video Camera]
||#(Sabotage Camera)
```

Figure 3: An example of the linear ASCII syntax.

Since white space should be insignificant in the textual ADTree syntax, there are representations of ADTrees that do not look intuitive. We, therefore, give some conventions for a proper layout:

- Every node is followed by a new line.
- The bar "`|`" and the connector symbols "`-`", "`=`" and "`#`" are aligned vertically.
- Two vertically aligned bars may be connected to form a continuous figure, see Figure 4.
- The symbols denoting the connector type may also be connected to the vertically aligned bars.
- Brackets may be replaced with a red oval (in case of attack node) or green rectangles (in case of defense nodes) surrounding the node label.

4.2 A Fancy Textual Syntax

To typeset a fancier textual syntax in LaTeX, we use the following commands:

Node	`\attack{Node}`	Attack nodes
Node	`\defense{Node}`	Defense nodes
—	`\mydash`	Disjunctive refinement
=	`\myeq`	Conjunctive refinement
#	`\myhash`	Countermeasure
	`\onebar`	One bar
	`\twobar`	Two bars
	`\threebar`	Three bars
	`\fourbar`	Four bars
	`\fivebar`	Five bars

They are used in a tabbing environment that looks like the following example:

```
\begin{tabbing}
\settabs
\raisebox{0.13cm}{\attack{Root node}}\+\\
More lines representing a tree
\end{minipage}
```

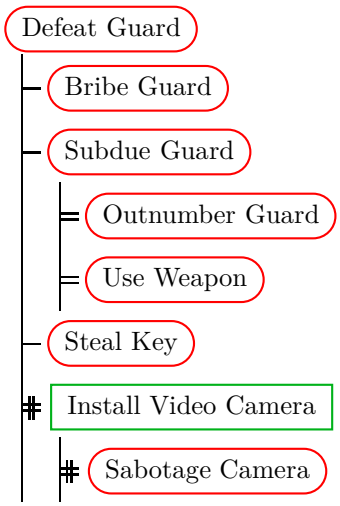


Figure 4: A colored textual ADTree for defeating a guard.

Figure 4 depicts a fancier version of Figure 3 that respects the conventions outlined in the previous section. It is generated using the following code:

```
\begin{figure}[tbh]
\begin{tabbing}
\settabs
\raisebox{0.13cm}{\attack{Defeat Guard}}\+\\
\onebar \mydash \attack{Bribe Guard}\\
\onebar \mydash \attack{Subdue Guard}\\
\twobar \myeq \attack{Outnumber Guard}\\
\twobar \myeq \attack{Use Weapon}\\
\onebar \mydash \attack{Steal Key}\\
\onebar \myhash \defense{Install Video Camera}\\
\twobar \myhash \attack{Sabotage Camera}
\end{tabbing}
\caption{A colored textual ADTree for defeating a guard.}
\end{figure}
```

5 Examples

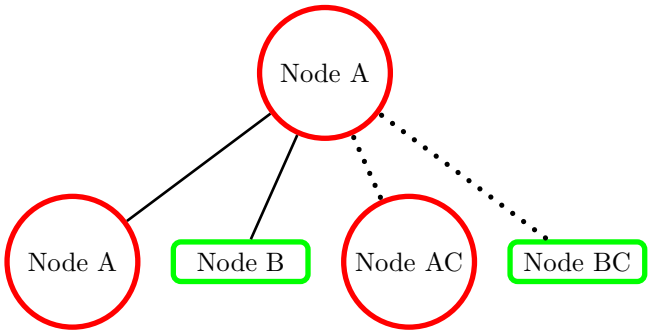
5.1 Nodes without IDs

Examples in this section are created with the following template:

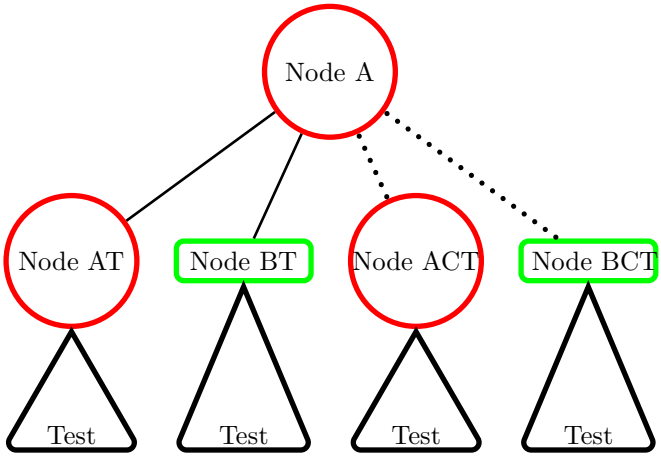
```
\begin{figure}[H]
\centering
\psscalebox{1}{
\pstree[levelsep=2.5cm,treesep=0.4cm]{\NodeA{Node A}}{
        INSERT CODE SNIPPET HERE
}
}
\end{figure}
```

5.1.1 Regular Nodes

```
\NodeA{Node A}
\NodeB{Node B}
\NodeAC{Node AC}
\NodeBC{Node BC}
```

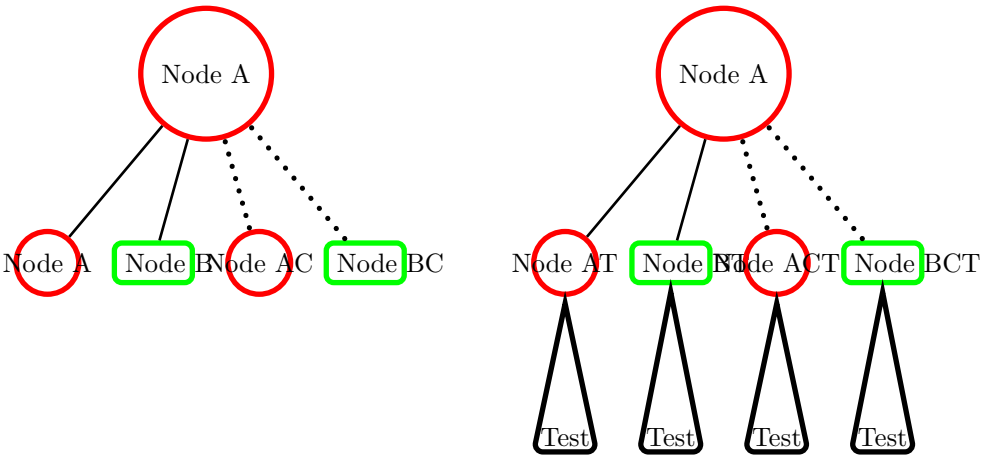



```
\NodeAT{Node AT}{Test}
\NodeBT{Node BT}{Test}
\NodeACT{Node ACT}{Test}
\NodeBCT{Node BCT}{Test}
```



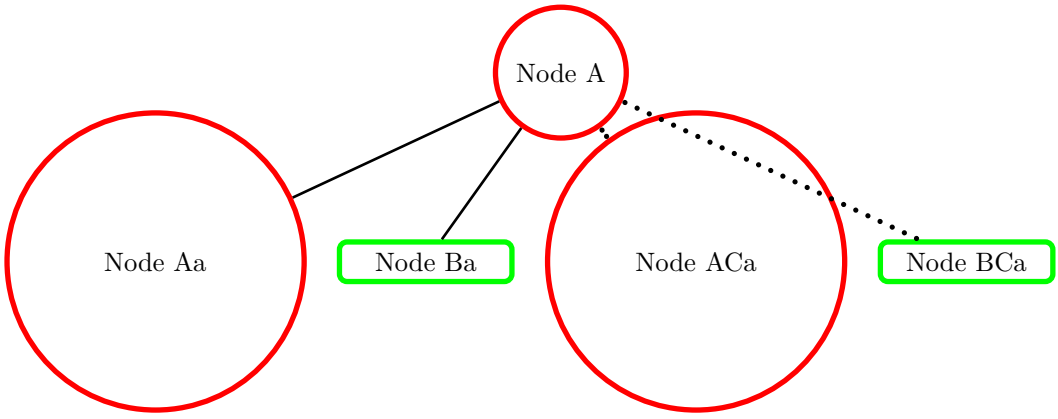
5.1.2 Small Nodes

```
\NodeAsmall{Node A}
\NodeBsmall{Node B}
\NodeACsmall{Node AC}
\NodeBCsmall{Node BC}
\NodeATsmall{Node AT}{Test}
\NodeBTsmall{Node BT}{Test}
\NodeACTsmall{Node ACT}{Test}
\NodeBCTsmall{Node BCT}{Test}
```

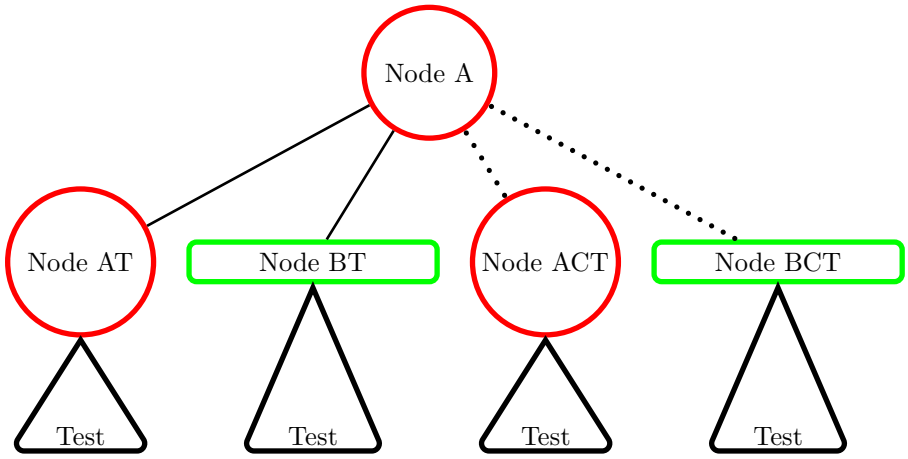


5.1.3 Adjustable Nodes

```
\NodeAa{2cm}{Node Aa}
\NodeBa{2cm}{Node Ba}
\NodeACa{2cm}{Node ACa}
\NodeBCa{2cm}{Node BCa}
```




```
\NodeATa{1cm}{2cm}{Node AT}{Test}
\NodeBTa{3cm}{2cm}{Node BT}{Test}
\NodeACTa{1cm}{2cm}{Node ACT}{Test}
\NodeBCTa{3cm}{2cm}{Node BCT}{Test}
```



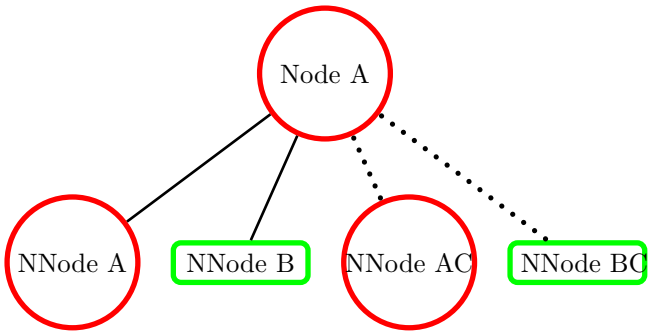
5.2 Nodes with IDs

Examples in this section are created with the following template:

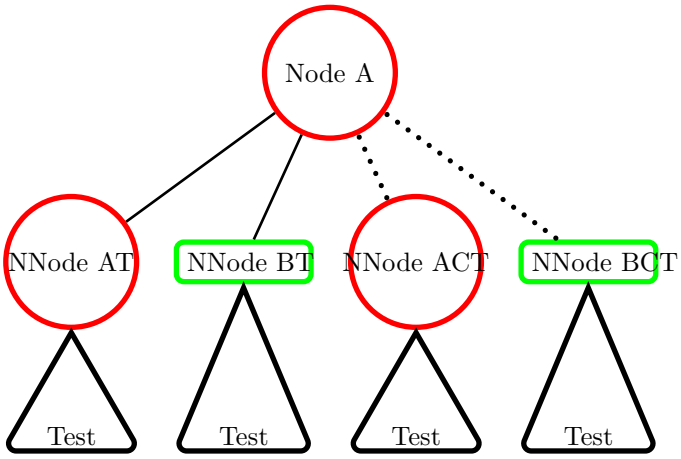
```
\begin{figure}[H]
\centering
\psscalebox{1}{
\pstree[levelsep=2.5cm,treesep=0.4cm]{\NodeA{Node A}}{
        INSERT CODE SNIPPET HERE
}
}
\end{figure}
```

5.2.1 Regular Nodes with IDs

```
\NNodeA{ID1}{NNode A}
\NNodeB{ID2}{NNode B}
\NNodeAC{ID3}{NNode AC}
\NNodeBC{ID4}{NNode BC}
```

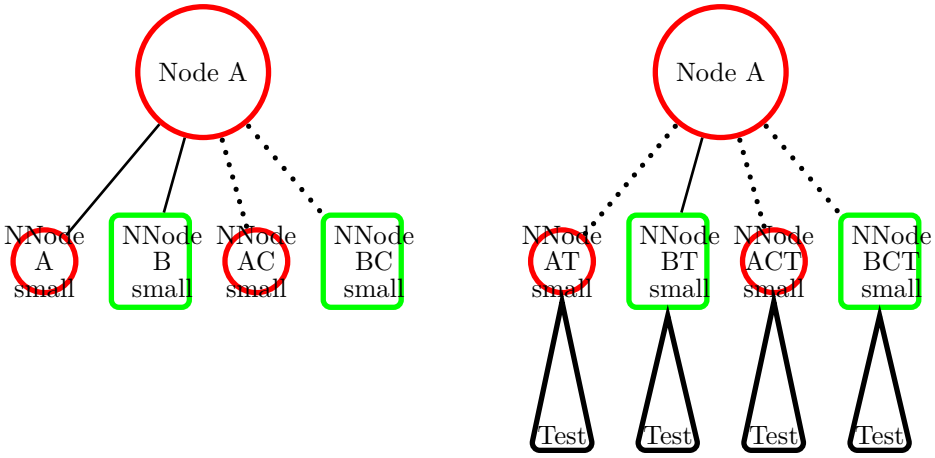



```
\NNodeAT{ID5}{NNode AT}{Test}
\NNodeBT{ID6}{NNode BT}{Test}
\NNodeACT{ID7}{NNode ACT}{Test}
\NNodeBCT{ID8}{NNode BCT}{Test}
```



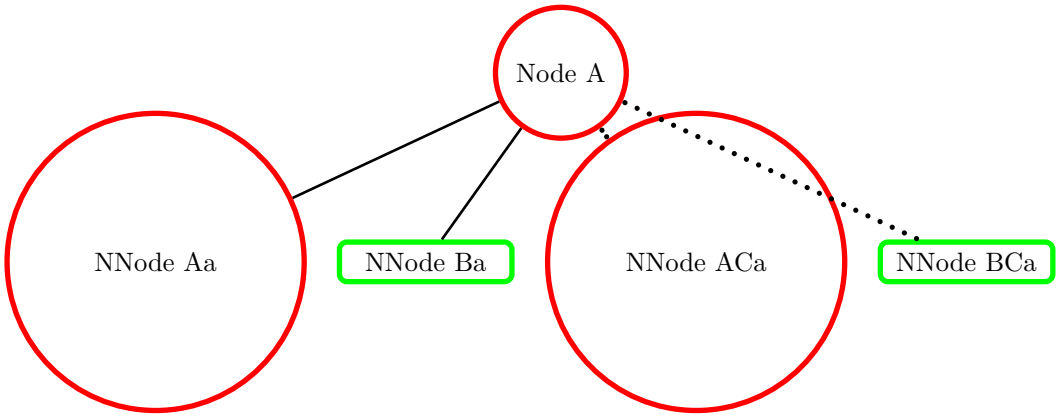
5.2.2 Small Nodes with IDs

```
\NNodeAsmall{ID1}{NNode\\A\\small}
\NNodeBsmall{ID2}{NNode\\B\\small}
\NNodeACsmall{ID3}{NNode\\AC\\small}
\NNodeBCsmall{ID4}{NNode\\BC\\small}
\NNodeATsmall{ID5}{NNode\\AT\\small}{Test}
\NNodeBTsmall{ID6}{NNode\\BT\\small}{Test}
\NNodeACTsmall{ID7}{NNode\\ACT\\small}{Test}
\NNodeBCTsmall{ID8}{NNode\\BCT\\small}{Test}
```

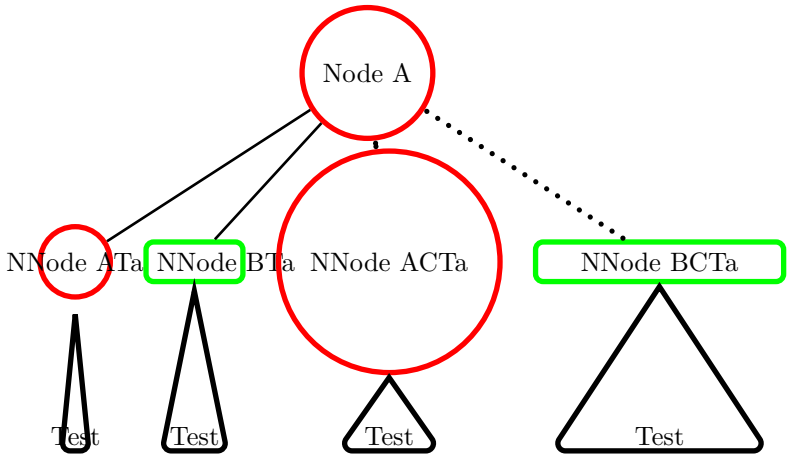


5.2.3 Adjustable Nodes with IDs

```
\NNodeAa{ID1}{2cm}{NNode Aa}
\NNodeBa{ID2}{2cm}{NNode Ba}
\NNodeACa{ID3}{2cm}{NNode ACa}
\NNodeBCa{ID4}{2cm}{NNode BCa}
```




```
\NNodeATa{ID5}{0.5cm}{0.5cm}{Node AT}{Test}
\NNodeBTa{ID6}{1cm}{1cm}{Node BT}{Test}
\NNodeACTa{ID7}{1.5cm}{1.5cm}{Node ACT}{Test}
\NNodeBCTa{ID8}{3cm}{3cm}{Node BCT}{Test}
```

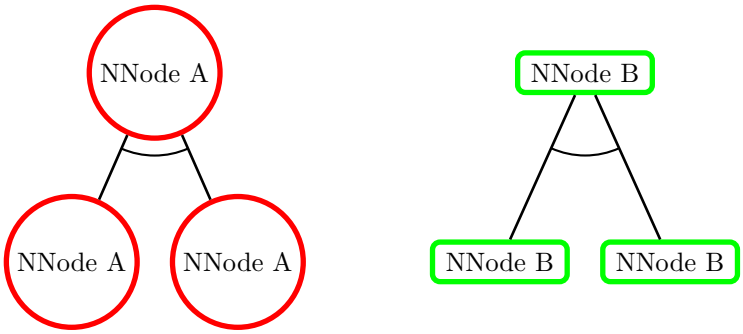


5.3 Arcs

Examples in this section contain the full template.

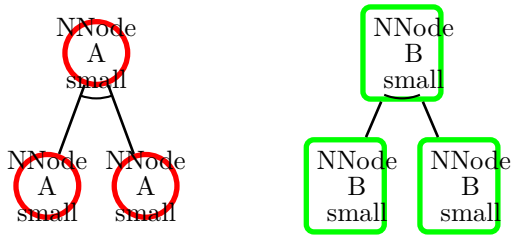
5.3.1 The Arc Command

```
\pstree[levelsep=2.5cm,treesep=0.4cm]{\NNodeA{ID1}{NNode A}}{
        \NNodeA{ID2}{NNode A}
        \NNodeA{ID3}{NNode A}
}
\Arc{ID1}{ID2}{ID3}

\pstree[levelsep=2.5cm,treesep=0.4cm]{\NNodeB{ID1}{NNode B}}{
        \NNodeB{ID2}{NNode B}
        \NNodeB{ID3}{NNode B}
}
\Arc{ID1}{ID2}{ID3}
```



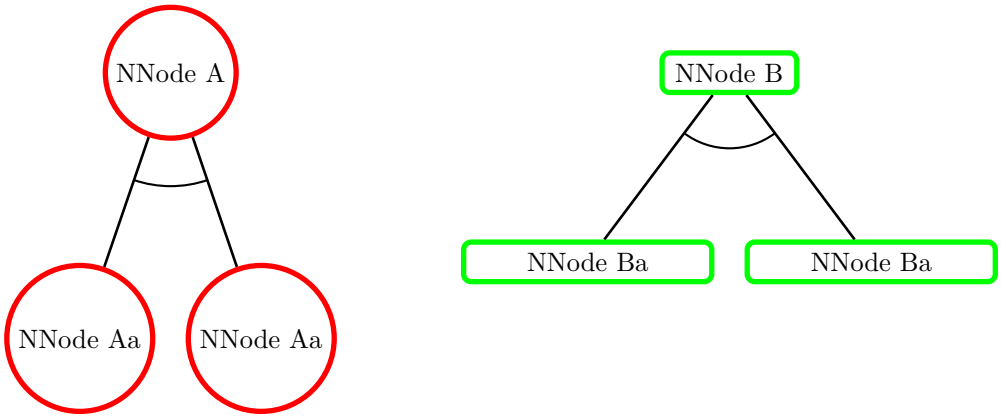
5.3.2 The Small Arc Command

```
\pstree[levelsep=50pt,treesep=0.4cm]{\NNodeAsmall{ID1}{NNode\\A\\small}}{
        \NNodeAsmall{ID2}{NNode\\A\\small}
        \NNodeAsmall{ID3}{NNode\\A\\small}
}
\Arcsmall{ID1}{ID2}{ID3}

\pstree[levelsep=50pt,treesep=0.4cm]{\NNodeBsmall{ID1}{NNode\\B\\small}}{
        \NNodeBsmall{ID2}{NNode\\B\\small}
        \NNodeBsmall{ID3}{NNode\\B\\small}
}
\Arcsmall{ID1}{ID2}{ID3}
```



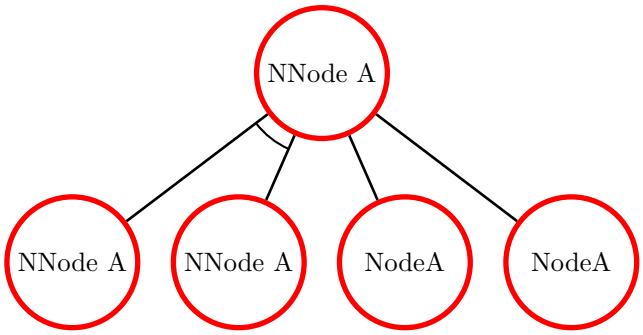
5.3.3 The Adjustable Arc Command

```
\pstree[levelsep=100pt,treesep=0.4cm]{\NNodeA{ID1}{NNode A}}{
        \NNodeAa{ID2}{1cm}{NNode Aa}
        \NNodeAa{ID3}{1cm}{NNode Aa}
}
\Arca{1.5cm}{ID1}{ID2}{ID3}

\pstree[levelsep=2.5cm,treesep=0.4cm]{\NNodeB{ID1}{NNode B}}{
        \NNodeBa{ID2}{3cm}{NNode Ba}
\NNodeBa{ID3}{3cm}{NNode Ba}
}
\Arca{1cm}{ID1}{ID2}{ID3}
```

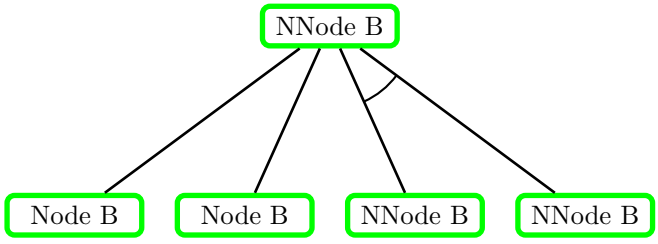


5.3.4 Left-hand and Right-hand Side Arc Commands

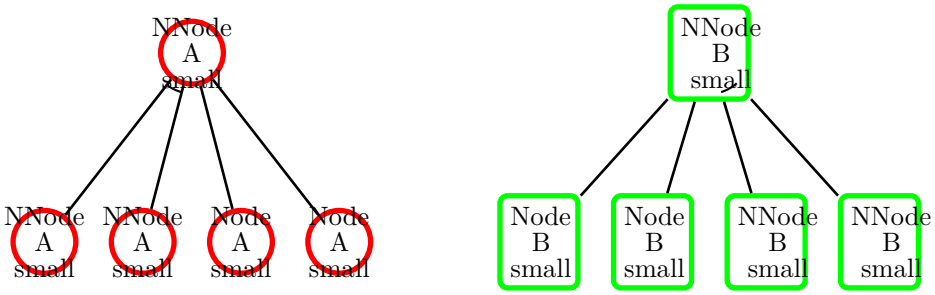
```
\pstree[levelsep=2.5cm,treesep=0.4cm]{\NNodeA{ID1}{NNode A}}{
\NNodeA{ID2}{NNode A}
        \NNodeA{ID3}{NNode A}
        \NodeA{NodeA}
        \NodeA{NodeA}
}
\Arclhs{ID1}{ID2}{ID3}
```



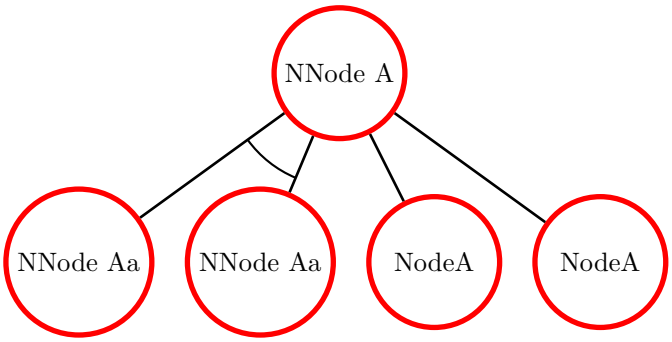

```
\pstree[levelsep=2.5cm,treesep=0.4cm]{\NNodeB{ID1}{NNode B}}{
        \NodeB{Node B}
        \NodeB{Node B}
        \NNodeB{ID2}{NNode B}
        \NNodeB{ID3}{NNode B}
}
\Arcrhs{ID1}{ID2}{ID3}
```



```
\pstree[levelsep=2.5cm,treesep=0.4cm]{\NNodeAsmall{ID1}{NNode\\A\\small}}{
        \NNodeAsmall{ID2}{NNode\\A\\small}
        \NNodeAsmall{ID3}{NNode\\A\\small}
        \NodeAsmall{Node\\A\\small}
        \NodeAsmall{Node\\A\\small}
}
\Arclhssmall{ID1}{ID2}{ID3}

\pstree[levelsep=2.5cm,treesep=0.4cm]{\NNodeBsmall{ID1}{NNode\\B\\small}}{
        \NodeBsmall{Node\\B\\small}
        \NodeBsmall{Node\\B\\small}
        \NNodeBsmall{ID2}{NNode\\B\\small}
        \NNodeBsmall{ID3}{NNode\\B\\small}
}
\Arcrhssmall{ID1}{ID2}{ID3}
```

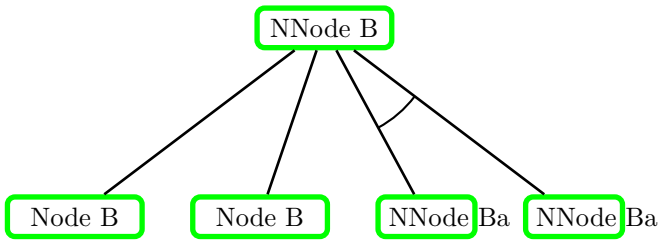



```
\pstree[levelsep=2.5cm,treesep=0.4cm]{\NNodeA{ID1}{NNode A}}{
        \NNodeAa{ID2}{1cm}{NNode Aa}
        \NNodeAa{ID3}{1cm}{NNode Aa}
        \NodeA{NodeA}
        \NodeA{NodeA}
}
\Arclhsa{1.5cm}{ID1}{ID2}{ID3}
```




```
\pstree[levelsep=2.5cm,treesep=0.6cm]{\NNodeB{ID1}{NNode B}}{
        \NodeB{Node B}
        \NodeB{Node B}
        \NNodeBa{ID2}{1cm}{NNode Ba}
```

```
        \NNodeBa{ID3}{1cm}{NNode Ba}
}
\Arcrhsa{1.5cm}{ID1}{ID2}{ID3}
```



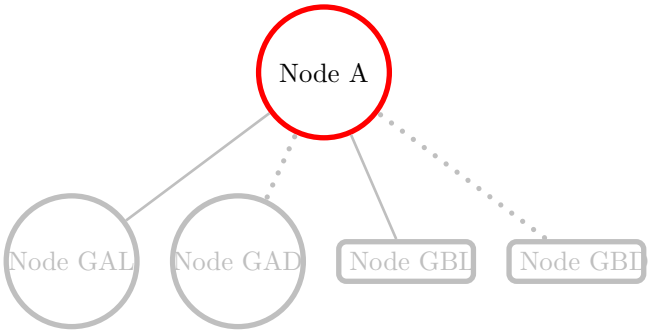
5.4 Miscellaneous Nodes

Examples in this section are created with the following template:

```
\begin{figure}[H]
\centering
\psscalebox{1}{
\pstree[levelsep=2.5cm,treesep=0.4cm]{\NodeA{Node A}}{
        INSERT CODE SNIPPET HERE
}
}
\end{figure}
```

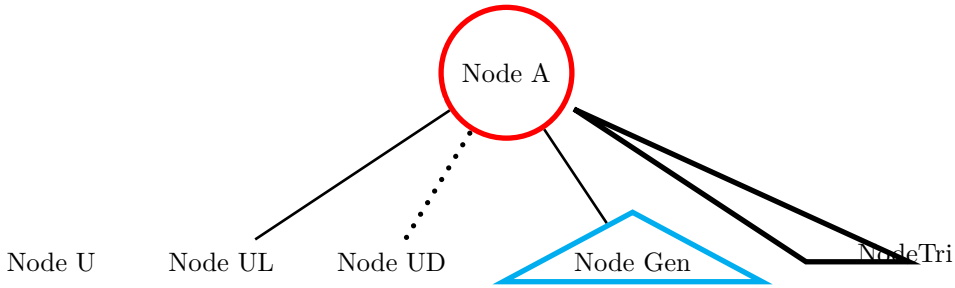
5.4.1 Grayed-out Nodes

```
\NodeGAL{Node GAL}
\NodeGAD{Node GAD}
\NodeGBL{Node GBL}
\NodeGBD{Node GBD}
```



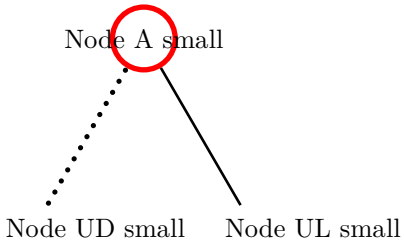
5.4.2 Other Nodes

```
\NodeU{Node U}
\NodeUL{Node UL}
\NodeUD{Node UD}
\NodeGen{Node Gen}
\NodeTri NodeTri
```



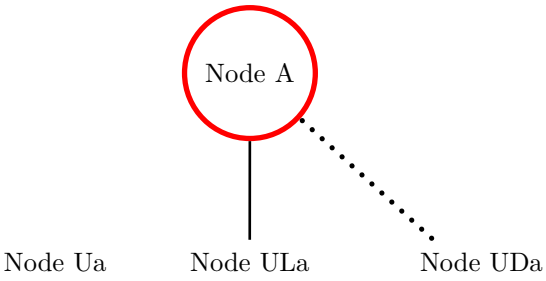
5.4.3 Other Small Nodes

```
\pstree[levelsep=2.5cm,treesep=1.8cm]{\NodeA{Node A}}{
\NodeUDsmall{Node UD small}
\NodeULsmall{Node UL small}
}
```



5.4.4 Other Adjustable Nodes

```
\NodeUa{1cm}{Node Ua}
\NodeULa{3cm}{Node ULa}
\NodeUDa{1cm}{Node UDa}
```



5.5 Miscellaneous Nodes with IDs

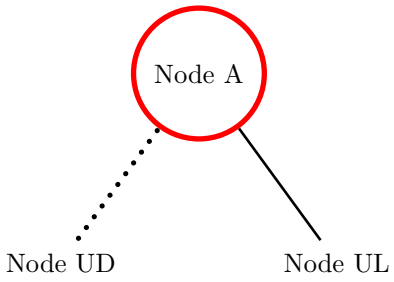
Examples in this section are created with the following template:

```
\begin{figure}[H]
\centering
\psscalebox{1}{
\pstree[levelsep=2.5cm,treesep=1.8cm]{\NodeA{Node A}}{
        INSERT CODE SNIPPET HERE
```

```
}
}
\end{figure}
```

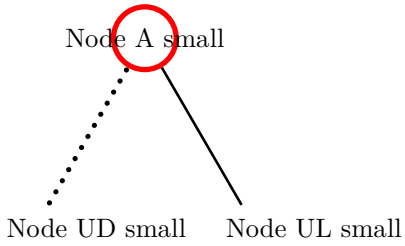
5.5.1 Other Nodes with IDs

```
\NNodeUD{ID1}{Node UD}
\NNodeUL{ID2}{Node UL}
```



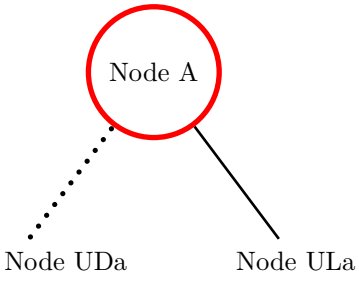
5.5.2 Other Small Nodes with IDs

```
\NNodeUDsmall{ID1}{Node UD small}
\NNodeULsmall{ID2}{Node UL small}
```



5.5.3 Other Adjustable Nodes with IDs

```
\NNodeUDa{ID1}{1cm}{Node UD}
\NNodeULa{ID2}{2cm}{Node UL}
```



6 Outdated Commands

The following operators are kept for backwards compatibility and are no longer documented:

```
\DeclareMathOperator{\Child}{Children}
\DeclareMathOperator{\TYPE}{TYPE}
\DeclareMathOperator{\CONN}{CONN}
\DeclareMathOperator{\VAR}{VAR}
\DeclareMathOperator{\Cu}{Cu}
\DeclareMathOperator{\Tq}{Tq}
\DeclareMathOperator{\Af}{Af}
\DeclareMathOperator{\Tk}{Tk}
\DeclareMathOperator{\Ca}{Ca}
\DeclareMathOperator{\Gu}{Gu}
\DeclareMathOperator{\Pa}{Pa}
\DeclareMathOperator{\Br}{Br}
\DeclareMathOperator{\Zo}{Zo}
\CircleVee
\CircleWedge
\CircleX
\CircleEmpty
\BoxVee
\BoxWedge
\BoxX
\BoxEmpty
\CircleBox
\BoxCircle
\Circlephi
\Boxphi
\mathphi
\CP
\BP
\every
\one
\Bogen{ID1}{ID2}{ID3}
\makearc
```