Tombstone package

v. 0.1

Mathias Jakobsen

February 11, 2017

The tombstone package is a LaTeX package for drawing tombstone diagrams (or T-diagrams).

Contents

1 Introduction

Figure 1 shows the four shapes that are available for use in tombstone diagrams. In order, the four different shapes represent programs, compilers, interpreters and machines. These shapes can be combined into 'puzzles' that represent the transformation of a program written in a source language into the same program written in a target language.

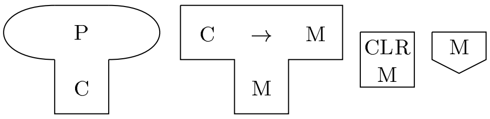

Figure 1: Shows the possible shapes for use in a tombstone diagram.

1.1 Installation

Installation of this package is very simple. Simply move the `tombstone.sty` file to the root folder of your project, and include the package using the command `\usepackage{tombstone}`.

2 Creating diagrams

This section will describe the commands necessary to create diagrams.

2.1 Shapes

The four shapes are created using matrices, which defines points that can be used to attach different shapes to each other. The matrix used to create the program shape can be seen below on figure 2. The red disk on the figure shows the default anchor of the shape. The location of these anchors differs from shape to shape, according to how they are commonly placed in a diagram. The default anchor location can be overwritten by specifying a new anchor as a named argument to the shape.

Each corner in the matrix can be used as a anchor point by following the convention `<shapeName>-<row>-<column>[.north/south west/east]`. For example, to draw the red disc in the figure, the point `prg-2-2.north west` is used to denote the north western corner of the middle column of the bottom row.

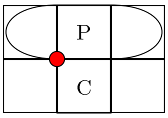

Figure 2:

```
\begin{tikzpicture}
    \begin{tombstonediagram}
        \program[drawmatrix]{prg}{P}{C}{}
        \draw [fill=red] (prg-2-2.north west) circle (1mm);
    \end{tombstonediagram}
\end{tikzpicture}
```

Listing 1: Code for generating figure 2

The four different shapes can be created using the following commands. The placement specifier for the first shape in a diagram can be omitted, as the shape will be placed in the upper left corner by default. Otherwise the placement specifier is used for choosing what other shape to place the shape unto.

```
\begin{tikzpicture}
    \begin{tombstonediagram}
        \program{<shape name>}{<program name>}{<language>}
            {<placement>}
        \compiler{<shape name>}{<source language>}
            {<target language>}{<placement>}
        \interpreter{<shape name>}{<language for interpreter>}
            {<placement>}
        \machine{<shape name>}{machine language}{<placement>}
    \end{tombstonediagram}
\end{tikzpicture}
```

Listing 2: Code for different shapes

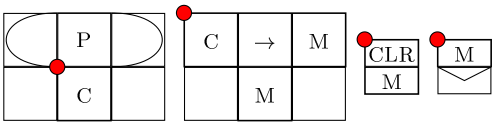

Figure 3: Shows the matrices and anchors of the different possible shapes.

2.2 Connecting shapes

Showing how shapes are connected is best done through examples. The following listing generates the diagram shown on figure 4.

```
\begin{tikzpicture}
    \begin{tombstonediagram}
        \compiler{cmp}{Java}{JVM}{JVM}{}
        \program[anchor=prg-2-2.north east]{prg}{Tetris}{Java}
            {at (cmp-1-1.north west)}
        \interpreter{int}{JVM}{x86}{at (cmp-2-2.south west)}
        \program{prgo}{Tetris}{JVM}{at (cmp-1-3.north east)}
        \machine{mac}{x86}{at (int-2-1.south west)}
    \end{tombstonediagram}
\end{tikzpicture}
```

Listing 3: Code for generating figure 4

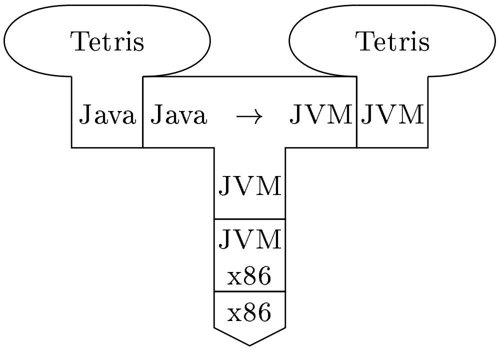

Figure 4: Connected shapes

2.2.1 Highlighting shapes

When creating diagrams, it may be necessary to highlight shapes in the diagram. Using the commands `\hightlight<shape type>{<shape name>}` it is possible to do so.

Appending the following 4 lines to the former example, generates figure 5.

```
\highlightprogram{prg}
\highlightcompiler{cmp}
\highlightinterpreter{int}
\highlightmachine{mac}
```

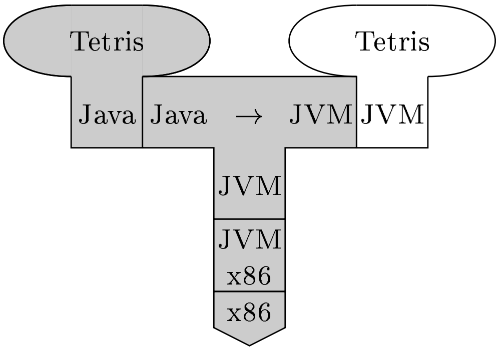

Figure 5: Connected shapes with highlighting