langsci-avm

Felix Kopecky[*]

Version 0.4.0 – April 29, 2025

1 Introduction

langsci-avm allows typesetting of feature structures, or *attribute-value matrices* (AVM), for use in linguistics. The package provides a minimal and easy to read syntax. The package serves the same purpose as Christopher Manning's `avm` package, but shares no code base with that package. There is a conversion guide in Section 4.6.

To start using langsci-avm, place `\usepackage{langsci-avm}` in your preamble.

This documentation is structured as follows: Section 2 describes the input syntax for AVMs and their parts. Ways to customise your AVM's layout follow in Section 3, and selected usage cases are presented in Section 4. There's also an administrative and TEXnical appendix at the end of this document, in case you are interested.

1.1 Example

```
\avm{
  [ ctxt & [ max-qud \\
             sal-utt & \{ [ cat \\
                             cont <ind & i> ] \}
           ]
  ]
}
```

$$\begin{bmatrix} \text{CTXT} & \begin{bmatrix} \text{MAX-QUD} & \\ \text{SAL-UTT} & \left\{ \begin{bmatrix} \text{CAT} & \\ \text{CONT} & \langle \text{IND} \quad i \rangle \end{bmatrix} \right\} \end{bmatrix} \end{bmatrix}$$

1.2 Acknowledgements

Thanks to Phelype Oleinik for help on recursion and expansion with LATEX3. Thanks to Ahmet Bilal Özdemir and Stefan Müller for their contributions in planning and testing this package.

---

[*] Please submit bug reports and feature requests to https://github.com/langsci/langsci-avm/issues.

2 Structuring AVMs

`\avm` `\avm [⟨options⟩] {⟨structure⟩}`

This root command of the package type sets AVMs in the documen. In the `{`⟨*structure*⟩`}`, delimiter characters are processed to open and close (sub-)structures, as described in Section 2.1. Special elements (e.g. tags, operators, type descriptors) are described in Section 2.2. For a description of the layout ⟨*options*⟩, see Section 3.

A ⟨*structure*⟩ is basically the content of a stylised `tabular`: The columns are separated by `&` and a new line is entered with `\\`.

2.1 Entering (sub-)structures within `\avm`

`[...]`	`[ ⟨structure⟩ ]`
`<...>`	`< ⟨structure⟩ >`
`(...)`	`( ⟨structure⟩ )`
`\{...\}`	`\{ ⟨structure⟩ \}`
`\[...\]`	`\[ ⟨structure⟩ \]`

Updated: 2020-10-02

Within the scope of `\avm`, these delimiters create (sub-)structures that are enclosed by the respective delimiter. Due to the special meaning that curly braces have in LaTeX, these are the only ones that need to be run with an escape token (`\`). It is currently possible to mix delimiters, e.g. with `<`⟨*structure*⟩`)`, but this may change in future versions.

langsci-avm expects your (sub-)structures to have *at most two columns*, so that for every line in each (sub-)structure, there should be no more than one `&`. It is recommended to have at least some lines with a `&` in your ⟨*structure*⟩. Currently, display issues may appear in some structures if none are given – see the `align=false` option to remedy this effect.

```
\avm{
   [ < ( \{ ... \} ) > ]
}
```

$$\left[\left\langle\left(\left\{\ldots\right\}\right)\right\rangle\right]$$

```
\avm{
   [ \{ ... \} \\
  < ( ... ), ( ... ) > ]
}
```

$$\begin{bmatrix}\{\ldots\}\\ \langle(\ldots),\ (\ldots)\rangle\end{bmatrix}$$

`\[ ... \]` `\[ ⟨structure⟩ \]`

New: 2020-10-02

Add a semantic bracket ⟦⟨*structure*⟩⟧.

Warning: Semantic brackets are only available when the package option `[lfg]` is loaded (`\usepackage[lfg]{langsci-avm}`). Documents with this option can only be compiled with XeLaTeX. If the `[lfg]` option is not present, `\[ {`⟨*structure*⟩`} \]` will result in none delimiter output, but the `{`⟨*structure*⟩`}` will be printed nonetheless. (The semantic delimiters are not available in every font, and are currently not provided in standard LaTeX documents. If you load the `[lfg]` option but do not provide the symbol (e.g. by using a font such as libertinus), the package unicode-math will automatically be loaded to provide the symbol.)

`\lframe ... \rframe`

New: 2021-03-03

`\lframe` ⟨*structure*⟩ `\rframe`

Delimit a ⟨*structure*⟩ placed in a rectangular box, which is used in Fillmore & Kay's notation. It can be used like the other delimiters.

```
\avm{
  \lframe ... \rframe
}
```

$$\boxed{\dots}$$

The parameters of the frame can be adjusted with these options:

`framewidth` = ⟨*length*⟩ (initially `1pt`)
Width of the frame.

`framesep` = ⟨*length*⟩ (initially `3pt`)
Separation of the frame and its contents.

`!...!`

`!` ⟨*text*⟩ `!`

Escapes the `avm` mode so that all delimiters can be used as usual characters. If you need `!` as a regular character, see Section 3 for how to change the `switch`.

2.2 Commands for tags, types, unusal lines, and relations

`\tag`
`\0`
`\1`
`...`
`\9`

Updated: 2020-04-29

`\tag` {⟨*identifier*⟩}
`\0, \1, \2, \3, \4, \5, \6, \7, \8, \9`

`\tag` puts its {⟨*identifier*⟩} in a box, more precisely an `\fbox`. Within the box, the `tags` font is applied. `\0`, `\1`, ..., `\9` are shortcuts to `\tag` and place the respective number in the box. For example, `\4` is equivalent to `\tag{4}`. The shortcuts do not take any arguments.

If you want to use this command outside an AVM, you can obtain, for example, $\boxed{4}$, by using `\avm{\4}`, or the equivalent `{\fboxsep.25ex\fbox{\footnotesize 4}}`.

```
\avm{[ attr1 & \4\\
       attr2 & \4[attr3 & val3\\
                  attr4 & val4] ]}
```

$$\begin{bmatrix} \textsc{attr1} & \boxed{4} \\ \textsc{attr2} & \boxed{4}\begin{bmatrix} \textsc{attr3} & \textit{val3} \\ \textsc{attr4} & \textit{val4} \end{bmatrix} \end{bmatrix}$$

`\type`
`\type*`

Updated: 2020-03-30

`\type`⟨***⟩ {⟨*type*⟩}

Will output the ⟨*type*⟩ in the `types` font (serif italics by default). The starred variant `\type*` will span the complete (sub-)structure and *can only be placed in the first column* of this structure. After the starred `\type*`, a `\\` is recommended, but can usually be omitted.

```
\avm{[ \type*{A type spanning a line}
       attr & [\type{type}] ]}
```

$$\begin{bmatrix} \textit{A type spanning a line} \\ \textsc{attr}\ \ [\textit{type}] \end{bmatrix}$$

`\id`

New: 2020-10-02
Updated: 2021-12-14

`\id {⟨id⟩} {⟨structure⟩}`

A variant of `\substack` from `amsmath`, this command adds an identifier to the `{`⟨*structure*⟩`}`. The contents of `{`⟨*id*⟩`}` will be set in math mode by default, which is convenient given that they often contain variables with subscript indices. Multiple IDs should be separated by a new line, `\\`.

```
\avm{\id{n_1\\n_2}{[subj\\pred&swim]}}
```

$$\begin{array}{l}n_1\\n_2\end{array}\begin{bmatrix}\text{SUBJ} & \\ \text{PRED} & \textit{swim}\end{bmatrix}$$

The position of the `{`⟨*id*⟩`}` column relative to the `{`⟨*structure*⟩`}` and the alignment within the `{`⟨*id*⟩`}` column can be changed:

`id align = ⟨token⟩` (initially `l`)
Change the alignment of the column inserted by `\id`. Has to be a column specification. The most probable choices are `l` and `r`.

`id position = ⟨option⟩` (initially `south-west`)
Change the position of `\id`. In the standard setting `south-west`, the `\id` is placed in the lower left corner of the enclosed structure. When set to `south-east`, the contents are set to the lower right corner. Currently, only `south-west` and `south-east` are recognised inputs, and an error is raised when an unknown option is input.

`\punk`

Updated: 2021-06-08

`\punk {⟨attribute⟩}{⟨type⟩}`

Some ⟨`attributes`⟩ think that the layout of the other attributes in their community leaves no space for them to express their individuality. They desire a life outside the confines of the alignment defined by the others, while still remaining a member of the matrix.

Technically, this is a line with no snapping to the column layout, but with spacing between the ⟨`attribute`⟩ and ⟨`type`⟩. After `\punk`, a `\\` is obligatory if not in the last line.

```
\avm{[ attr1 & val1\\
      \punk{a quite long attr2}{val2}
       attr3 & val3\\
       attr4 & val4
     ]}
```

$$\begin{bmatrix}\text{ATTR1} & \textit{val1} \\ \multicolumn{2}{l}{\text{A QUITE LONG ATTR2}\ \ \textit{val2}} \\ \text{ATTR3} & \textit{val3} \\ \text{ATTR4} & \textit{val4}\end{bmatrix}$$

Hint: Also have a look at the option `align=false`.

`\+`

Updated: 2020-03-16

In the scope of `\avm`, `\+` comes out as “⊕”. “+” can be obtained normally. *In the earlier Version 0.1.0-beta,* `+` *produced “⊕”.*

`\-`

New: 2020-03-17

In the scope of `\avm`, `\-` comes out as “⊖”. To use the “optional hyphenation” meaning of `\-`, please write `!\-!`, where `!` is your current `switch` token.

`\shuffle`

New: 2020-03-17

In the scope of `\avm`, `\shuffle` is a shortcut for “○” to mark the shuffle relation.

3 AVM layout

3.1 Customisation and style definitions

You can customise many aspects of how an AVM is printed, including the fonts or spacing between delimiters and content. You can apply them locally via the `[`⟨*options*⟩`]` of `\avm` or globally by using `\avmsetup`. And you can also define your own styles and use them via the `[`⟨*style* =⟩`]` option in `\avm`.

`\avmsetup` `\avmsetup {`⟨*options*⟩`}`

`{`⟨*options*⟩`}` is a comma-separated list of `key=value` settings. See the list below for all user-configurable options. The `{`⟨*options*⟩`}` are the same as in `\avm[`⟨*options*⟩`]`. When inserted in `\avm[`⟨*options*⟩`]`, they apply locally, and globally if given to `\avmsetup`. Local settings always override global ones, and you can have any feasible number of `\avmsetups` in your document. The scope of `\avmsetup` can be restricted through grouping.

`\avmdefinestyle` `\avmdefinestyle {`⟨*name*⟩`} {`⟨*settings*⟩`}`

New: 2020-05-11

Instead of applying settings globally or per AVM, you can also define styles and assign them to AVMs, as in `\avm[style=`⟨*name*⟩`]{...}`. The ⟨*settings*⟩ are a comma-separated list of `key=value` settings, and should be a subset of the settings from `\avmsetup`. For example, the following `plain` style highlights neither attributes, values, nor types:

```
\avmdefinestyle{plain}{attributes=\normalfont,
                       values=\normalfont,
                       types=\normalfont}
```

The style is applied with `\avm[style=plain]{...}`.

Now to the list of settings:

`style =` ⟨*name*⟩ (initially empty)
In addition to any style that you possibly define yourself, a style `narrow` is predefined in the package (see Section 4.1).

`align =` ⟨*choice*⟩ (initially `true`)
Controls whether the columns in the AVM and its substructures should be aligned (snapping to the grid) or not. Aligned AVMs are separated by `columnsep`, non-aligned are separated by `vectorsep`.

`stretch =` ⟨*factor*⟩ (initially `0.9`)
Define `\arraystretch`, i.e. a factor in the determination of line height.

`columnsep =` ⟨*length*⟩ (initially `0.5ex`)
Define the `\tabcolsep`, i.e. horizontal space between columns. The first and second column will have `0\columnsep` to the left and right, respectively. Between the two the distance is `2\columnsep`. Using relative units (like `ex` or `em`) may be a good idea so that `columnsep` scales well with changes in font size.

`vectorsep =` ⟨*length*⟩ (initially `1em`)
Define the horizontal separation between columns in non-aligned matrices (see option `align`).

`delimfactor = ⟨factor⟩` (initially `1000`)

Sets `\delimiterfactor`. The calculation for the minimum height of a delimiter is $y \cdot f/1000$, where y is the height of the content and f the value of `delimfactor`. The default `1000` ensure that the delimiters' height is at least that of the structure.

`delimfall = ⟨length⟩` (initially `0pt`)

Controls `\delimitershortfall`, i.e. the maximum height that the delimiters can be shorter than the enclosed structure. The default `0pt` ensure that the delimiters are not shorter than the contents.

`extraskip` = `⟨length⟩` (initially `\smallskipamount`)

If a substructure is immediately followed by a `\\`, an extra amount of vertical skip is added so that the content of the next line, possibly another delimiter, does not clash with the delimiter in that line. This automatic skip insertion can be circumvented with placing a `\relax` before the linebreak, i.e. `\relax\\`.

`attributes = ⟨font settings⟩` (initially `\scshape`)

The font for attributes, i.e. the first column of each structure.

`values = ⟨font settings⟩` (initially `\itshape`)

The font for values, i.e. the second column of each structure.

`types = ⟨font settings⟩` (initially `\itshape`)

The font used in `\type` and `\type*`.

`tags = ⟨format settings⟩` (initially `\footnotesize`)

The font (size) used in `\tag` and the shortcuts `\1`...`\9`.

`switch = ⟨token⟩` (initially `!`)

Define the escape token. Change this if you need to use "!" as a text glyph.

`customise = ⟨settings⟩` (initially empty)

An interface to input custom commands to be run at the beginning of every `\avm`.

3.2 Local settings

Settings can be applied locally by using the `scope` environment. Local settings will take effect for the next beginning structure.

`scope` `\begin{scope} {⟨options⟩} ⟨...⟩ \end{scope}`

`New: 2024-01-30`

`{`*⟨options⟩*`}` is a comma-separated list of `key=value` settings. See the list above for all user-configurable options.

The settings will take effect for all structures that *begin* within a `scope` environment. That means that the behaviour of the currently active structure can not be changed with this environment.

```
\avm{
  [
    [Hello & World]\\
    \begin{scope}{values=\bfseries}
      [Hello & World]
    \end{scope}
  ]
}
```

$$\begin{bmatrix} [\textsc{Hello}\ \ \textit{World}] \\ [\textsc{Hello}\ \ \textbf{World}] \end{bmatrix}$$

3.3 Drawing edges between AVM contents

It is possible to make AVM contents available to `tikz`, so that they can be referenced in a `tikzpicture`. To enable this feature, langsci-avm has to be loaded with the option `[tikz]`:

```
\usepackage[tikz]{langsci-avm}
```

Additionaly, `avm` environments on which `tikz` is to be used need to have the `[pic]` option present:

```
\avm[pic] {...}
```

Only the parts of an AVM that are specifically marked will be known to `tikz`. To mark a part of an AVM to be used by Ti*k*Z, use `\node`:

`\node`

New: 2020-09-23

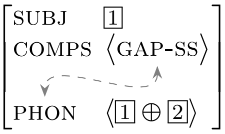
`\node {⟨id⟩} {⟨contents⟩}`

`{⟨id⟩}` serves as part of the node's identifier in a `tikzpicture`. It will be prefixed, and it's complete name will be `avm-n-⟨id⟩`, where `n` is the counter of `\avm` in your document that have the `[pic]` option enabled and that don't have a `picname` (see below). `n` starts at 1. For example, a `\node` named "pretty-node" in the fourth `[pic]`-enabled `avm` in your document will be `avm-4-pretty-node`. Note that `\node` will register the complete name globally in your document, and so can't be declared by other `tikz` nodes.

This behaviour can be adjusted by passing a `[picname = ⟨avm's name⟩]` to `\avm`. E.g., `\nodes` within `\avm[pic, picname=example1]` will have a full name pattern of `example1-⟨id⟩`. Named `\avm`s do *not* raise the `n` mentioned in the last paragraph.

Any (sub-)structure can be placed into `{⟨contents⟩}`. It could be just a value, an attribute's name, or parts thereof, but whole (sub-)structures can be part of `{⟨contents⟩}` as well.

A `tikzpicture` with options `[remember picture, overlay]` enabled can reference langsci-avm's `\node`s. This way, Ti*k*Z' extensive drawing abilities are available for the decoration of AVMs. Here's a very simple example document:



```
\documentclass{article}
\usepackage[tikz]{langsci-avm}
\usepackage{tikz} % optional, since langsci-avm will load tikz if option
                  % tikz is present
\usetikzlibrary{arrows,arrows.meta}

\avm[pic]{[ subj  & \1\\
        comps &  <\node{gap}{gap-ss}> \bigskip\\
        \node{phon}{phon} &   <\1 \+ \2>
        ]}

\begin{tikzpicture}[remember picture,overlay]
        \path[{Stealth[]}-{Stealth[]},gray,dashed,in=90,out=270]
        (avm-1-gap.south) edge (avm-1-phon.north);
\end{tikzpicture}
```

3.4 Defining input patterns

`\avmdefinecommand` `\avmdefinecommand {⟨name⟩} [⟨label⟩] {⟨settings⟩}`

New: 2020-06-29

Structures often follow specific patterns. For example, AVMs often have a PHON attribute, which is mapped to a list, the entries of which are in italics. `\avmdefinecommand` can account for this and other input patterns. For example,

```
\avmdefinecommand{custom}{...}
```

will create a command `\custom` available only in the scope of `\avm` (this means that you can have a different meaning in the rest of your document). The ⟨*`settings`*⟩ will then be applied to the scope in which `\custom` is called. If an optional ⟨*`label`*⟩ is given, the label will be printed, in the current font, before the ⟨*`settings`*⟩ are applied.

`\custom` generated in this way automatically advances to the value column after the ⟨*`label`*⟩ is printed. This means that commands generated with `\avmdefinecommand` should be called in the attribute column of an existing structure. This behaviour can be circumvented with the starred variant `\name*`, which is automatically generated by `\avmdefinecommand` as well. However, it seems advisable to use the starred variants sparingly.

Here's an example for the aforementioned phon pattern:

```
\avmdefinecommand{phon}[phon]
  {
    attributes  = \itshape,
    delimfactor = 900,
    delimfall   = 10pt
  }
```

This creates a command `\phon` (and the variant `\phon*`) within the scope of any `\avm`. It will print the label `phon` in the current font and then apply three settings locally: italics for the attribute (first) column, and two settings for very narrow delimiter fitting.

This results in:

```
\avm{
  [\type*{word}
   \phon <lin'gwistiks>\\
   synsem & [ ... ]
  ]
}
```

$$\begin{bmatrix} \textit{word} & \\ \text{PHON} & \langle \textit{lin'gwistiks} \rangle \\ \text{SYNSEM} & [\ldots] \end{bmatrix}$$

Note that any other structure type would have worked instead of ⟨⟩. But ⟨⟩ and any other markers for sub-structures are left unchanged by `\phon` and other custom commands. This is why the *attribute* font is changed by `\phon`, although *lin'gwistiks* is technically a value. Remember that `<` creates a new list sub-substructure, and the first content is printed in its attribute font.

4 Applications

4.1 Spacing and size of delimiters

langsci-avm automatically detects if the end of a sub-structure is followed by a line break. This is useful to find cases in which two sub-structures are printed immediately below each other, and to add extra spacing (the `extraskip` from the options). This automatic detection can be suppressed with `\relax`. See below for the effect of that detection:

```
\avm{[ [attr1 & val1 \\
        attr2 & val2 ] \\
       [attr1 & val1 \\
        attr2 & val2 ]
     ]}
```

```
\avm{[ [attr1 & val1 \\
        attr2 & val2 ] \relax\\
       [attr1 & val1 \\
        attr2 & val2 ]
     ]}
```

$$\begin{bmatrix}\begin{bmatrix}\textsc{attr1} & \textit{val1}\\ \textsc{attr2} & \textit{val2}\end{bmatrix}\\[4pt] \begin{bmatrix}\textsc{attr1} & \textit{val1}\\ \textsc{attr2} & \textit{val2}\end{bmatrix}\end{bmatrix} \qquad \begin{bmatrix}\begin{bmatrix}\textsc{attr1} & \textit{val1}\\ \textsc{attr2} & \textit{val2}\end{bmatrix}\\ \begin{bmatrix}\textsc{attr1} & \textit{val1}\\ \textsc{attr2} & \textit{val2}\end{bmatrix}\end{bmatrix}$$

If many delimiters are nested, this occasionally results in larger delimiter sizes. There is a pre-defined `narrow` style that resets `delimfall` (to `5pt`) and `delimfactor` (to `997`), which are the values recommended in the *TEXbook*. This results in a more compact appearance:

```
\avm{[ attr \{<\1>\}]}
```

```
\avm[style=narrow]{[ attr \{<\1>\}]}
```

$$\begin{bmatrix}\textsc{attr} & \left\langle\left\{\boxed{1}\right\}\right\rangle\end{bmatrix} \qquad \begin{bmatrix}\textsc{attr} & \langle\{\boxed{1}\}\rangle\end{bmatrix}$$

4.2 Disjunctions and other relations

Sometimes AMVs are placed beside other content to express disjunctions or other relations. In langsci-avm this is done naturally:

```
\avm{[attr1 & val1\\
      attr2 & val2\\
      attr3 & val3]} $\lor$
\avm{[attr1' & val1'\\
      attr2' & val2'\\
      attr3' & val3'\\]}
```

$$\begin{bmatrix}\textsc{attr1} & \textit{val1}\\ \textsc{attr2} & \textit{val2}\\ \textsc{attr3} & \textit{val3}\end{bmatrix} \lor \begin{bmatrix}\textsc{attr1'} & \textit{val1'}\\ \textsc{attr2'} & \textit{val2'}\\ \textsc{attr3'} & \textit{val3'}\end{bmatrix}$$

```
\textit{sign} $\to$
\avm{[attribute1 & value1\\
     attribute2 & value2\\
     attribute3 & value3]}
```

$$\textit{sign} \to \begin{bmatrix}\textsc{attribute1} & \textit{value1}\\ \textsc{attribute2} & \textit{value2}\\ \textsc{attribute3} & \textit{value3}\end{bmatrix}$$

4.3 Use as a vector

It's possible to use langsci-avm for feature vectors rather than matrices, as may be useful in generative grammar.

```
\avm[attributes=\normalfont]{[v1\\v2\\v3]}$\varphi$
```

$$\begin{bmatrix}\text{v1}\\\text{v2}\\\text{v3}\end{bmatrix}\varphi$$

4.4 Combinations with gb4e, expex, and linguex

This package works fine with gb4e and its fork langsci-gb4e. To align the example number at the top of your structure, please use `\attop` from gb4e:

```
\begin{exe}
    \ex\attop{
    \avm{[ attr1 & val1\\
           attr2 & val2\\
           attr3 & val3]}
    }
  \end{exe}
```

(1) $\begin{bmatrix}\textsc{attr1} & \textit{val1}\\\textsc{attr2} & \textit{val2}\\\textsc{attr3} & \textit{val3}\end{bmatrix}$

The same can be achieved with expex using `\envup` from lingmacros (see below) or using this *experimental* syntax:

```
\ex \vtop{\strut\vskip-\baselineskip{
  \avm{[ attr1 & val1\\
         attr2 & val2\\
         attr3 & val3]}
}}
\xe
```

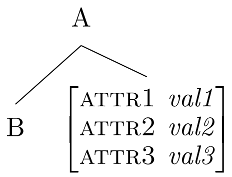
Examples typed with linguex can be combined with `\evnup` from lingmacros to align AVMs (many thanks to Jamie Findlay for pointing this out):

```
\ex. \envup{\avm{[ attr1 & val1\\
         attr2 & val2\\
         attr3 & val3]}
      }
```

4.5 Combinations with forest

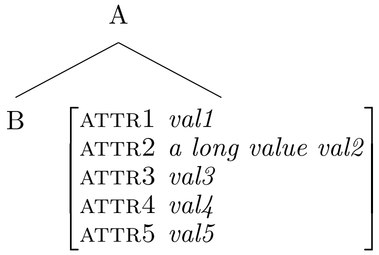
This package also works fine with forest. As per the forest documentation, it is recommended to protect any `\avm`-statements with `{}` in nodes:

```
\begin{forest}
 [A [B] [{\avm{[attr1 & val1\\
                attr2 & val2\\
                attr3 & val3]}} ] ]
\end{forest}
```



It may happen that extensive AVMs protrude into the space reserved for other forest nodes or edges. In this case, the forest setting `for children = {anchor=north}` may be useful: (If you like, try this tree without that setting.)

```
\begin{forest}
  [A, for children = {anchor=north}
     [B] [{\avm{[attr1 & val1\\
     attr2 & a long value val2\\
     attr3 & val3\\
     attr4 & val4\\
     attr5 & val5]}} ]
  ]
\end{forest}
```



4.6 Switching from Christopher Manning's **avm** package

Switching from avm to langsci-avm will require some, though hopefully minimal, changes to the code. In particular, langsci-avm doesn't distinguish between "active" and "passive" modes, there is now a single way of sorting (see `\type`, which replaces `\asort` and `\osort`), and tags are now produced without `@` (`\4` instead of `@4`, etc.).

Paths can be printed with a normal `|`, and ⊕ and other relation symbols can be input more easily (see Section 2.1), though the package will also work with `$|$` and `$\oplus$`.

5 Caveats and planned features

1. There are currently no error messages. If you do not receive the intended output, please make sure that your code fits the syntax described in this documentation. If your code is fine but the output is not, please submit a bug report or feature request at `https://github.com/langsci/langsci-avm/issues`.

   These features are planned for the future:

2. A check whether the delimiters are balanced, i.e. whether all (sub-)structures are closed by a `]`, `}`, etc.
3. Improve the appearance of (very) large angle brackets so that they vertically span the complete structure they enclose, maybe using scalerel.

6 Implementation

```
⟨*package⟩
⟨@@=avm⟩
\RequirePackage{xparse}[2022/03/26]
\RequirePackage{array}
\ProvidesExplPackage {langsci-avm}
  {2025-04-29} {0.4.0}
  {AVMs and feature structures}

\msg_new:nnnn {avm} {lfgoptionmissing}
  {  Missing~package~option~lfg~at~line~\msg_line_number:  }
  {
    You~issued~a~command~in~line~\msg_line_number:~that~is~only~available~when~
    the~lfg~package~option~is~enabled.
  }

\msg_new:nnnn {avm} {idpositionunknown}
  {  Unkown~value~for~option~`id~position`~near~line~\msg_line_number:.  }
  {  You~specified~an~unknown~value~for~option~`id~position`.~The~content~of~
     the~id~could~not~be~output.~Please~see~the~manual~for~a~list~of~valid~
     settings.
  }
```

Let's first check for package options.

```
\bool_new:N \l__avm_lfg_bool
\bool_new:N \l__avm_tikz_bool
\DeclareOption{tikz}{ \bool_set_true:N \l__avm_tikz_bool }
\DeclareOption{lfg}{ \bool_set_true:N \l__avm_lfg_bool }
\ProcessOptions\relax
```

Handling for the Ti*k*Z package option.

```
\bool_if:NT \l__avm_tikz_bool
  {
    \RequirePackage{tikz}
    \newcounter{l__avm_picture_counter}
    \tl_new:N \l__avm_picture_name_prefix_tl
  }
```

Handling for the LFG package option: If the semantic bracket is not available at the end of the preamble (i.e.) it was not loaded by another package, load unicode-math to provide the symbol.

```
\bool_if:NT \l__avm_lfg_bool
  {
    \cs_if_exist:NF \lBrack
      {
        \RequirePackage{etoolbox}
        \AtEndPreamble { \RequirePackage{unicode-math} }
      }
  }
```

`\avm` This document command initialises an AVM. The first, optional argumet is a key-value list of settings (see `\keys_define:nn` below) and the second is the AVM itself, given in the syntax described in this documentation.

`\avm` enters a group so that keys- and macro-assignemts remain local. It then initialises the commands and shortcuts and any user customisation, sets its mode to `true` and assigns the keys as given in the optional argument (if any). After the parser `\__avm_parse:n` is called, the group is closed.

```
\NewDocumentCommand{\avm}{ O{} +m }
  {
    \c_group_begin_token
    \keys_set:nn { avm } { #1 }
    \__avm_initialise_document_commands:
    \__avm_initialise_custom_commands:
    \tl_use:N \l__avm_defined_commands_tl
    \__avm_mode_switch:
    \__avm_parse:n { #2 }
    \c_group_end_token
  }
```

(*End of definition for* `\avm`. *This function is documented on page 2.*)

`\l__avm_mode_bool`
`\l__avm_parens_tracker`
`\l__avm_defined_commands_tl`
`\l__avm_fillmore_kay_box`

We need an auxiliary variable to store the current mode. `\l__avm_parens_tracker` is a stack for a future check whether the delimiters given to `\avm` are balanced. `\l__avm_defined_commands_tl` is a token list that stores any commands provided by the user via `\avmdefinecommand`. The box `\l__avm_fillmore_kay_box` is used as a temporary storage to realise Fillmore & Kay's notation.

```
\bool_new:N \l__avm_mode_bool
\seq_new:N \l__avm_parens_tracker
\tl_new:N \l__avm_defined_commands_tl
\box_new:N \l__avm_fillmore_kay_box
\tl_new:N \l__avm_parsed_tl
\int_new:N \l__avm_mode_switch_character_int
```

(*End of definition for* `\l__avm_mode_bool` *and others.*)

`\avmsetup`

Forward the key-value settings given as the optional argument to `\avm` to the keys defined in `\keys_define:nn { avm }`. For the meaning of these keys and initial values, see Section 2.

```
\NewDocumentCommand{\avmsetup}{ m }
  { \keys_set:nn { avm } { #1 } }

\keys_define:nn { avm }
  {
    align .bool_set:N      = \l__avm_align_bool,
    align .initial:n       = {true},
    stretch .tl_set:N      = \l__avm_arraystretch_tl,
    stretch .initial:n     = {0.9},
    columnsep .dim_set:N   = \l__avm_tabcolsep_dim,
    columnsep .initial:n   = {.5ex},
    vectorsep .dim_set:N   = \l__avm_singlesep_dim,
    vectorsep .initial:n   = {1em},
    delimfactor .int_set:N = \l__avm_delimfactor_int,
    delimfactor .initial:n = {1000},
    delimfall .dim_set:N   = \l__avm_delimshortfall_dim,
    delimfall .initial:n   = {0pt},
    framewidth .dim_set:N  = \l__avm_fillmore_kay_boxrule_dim,
    framewidth .initial:n  = {1pt},
    framesep .dim_set:N    = \l__avm_fillmore_kay_boxsep_dim,
    framesep .initial:n    = {3pt},
    attributes .code:n     = {\cs_set:Nn \__avm_font_attribute: {#1}},
    attributes .initial:n  = {\scshape},
    types .code:n          = {\cs_set:Nn \__avm_font_type: {#1}},
    types .initial:n       = {\itshape},
    values .code:n         = {\cs_set:Nn \__avm_font_value: {#1}},
    values .initial:n      = {\itshape},
    tags .code:n           = {\cs_set:Nn \__avm_font_tag: {#1}},
    tags .initial:n        = {\footnotesize},
    singleton .code:n      = {\cs_set:Nn \__avm_font_singleton: {#1}},
    singleton .initial:n   = {\normalfont},
    switch .code:n         =
      {
        \tl_set:Nn \l__avm_mode_switch_character {#1}
        \exp_args:NNx \int_set:Nn \l__avm_mode_switch_character_int
          {`\tl_use:N \l__avm_mode_switch_character}
      },
    switch .initial:n      = { ! },
    extraskip .dim_set:N   = \l__avm_extra_skip_dim,
    extraskip .initial:n   = {\smallskipamount},
    extraskip~in~every~row .bool_set:N = \l__avm_extraskip_bool,
    customise .code:n      = {\cs_set:Nn \__avm_initialise_custom_commands:
                                         {#1}},
    customise .initial:n   = { },
    pic .bool_set:N        = \l__avm_picture_bool,
    pic .default:n         = { true },
    picname .tl_set:N      = \l__avm_picture_name_tl,
    picname .initial:n    = {automatic},
    id~align .code:n       = { \newcolumntype{i}{#1} },
    id~align .initial:n    = {l},
    id~position .tl_set:N  = \l__avm_id_position_tl,
    id~position .initial:n = {south-west},
    style .choice:,
    style / narrow .code:n = {\int_set:Nn \l__avm_delimfactor_int {997}
                           \dim_set:Nn \l__avm_delimshortfall_dim {5pt}},
  }
```

(*End of definition for* `\avmsetup`. *This function is documented on page 5.*)

`\avmdefinestyle` Define a style to be used together with the `style` key.

```
\NewDocumentCommand{\avmdefinestyle}{ m m }
  {
    \keys_define:nn { avm }
      {
        style / #1 .code:n = { \keys_set:nn { avm } { #2 } }
      }
  }
```

(*End of definition for* `\avmdefinestyle`. *This function is documented on page 5.*)

`\avmdefinecommand` A factory function that creates commands for the layout of sub-structures and saves them to `\l__avm_defined_commands_tl`. The first argument describes the command's name, the second any (optional) label. The manufactured definitions are activated in the AVM group so that they remain local.

```
\NewDocumentCommand{\avmdefinecommand}{ m O{} m }
  {
    \tl_put_right:Nn \l__avm_defined_commands_tl
      {
        \exp_args:Nc \DeclareDocumentCommand { #1 } { s }
          {
             #2 \IfBooleanF { ##1 } { & } \avmsetup{ #3 }
          }
      }
  }
```

(*End of definition for* `\avmdefinecommand`. *This function is documented on page 8.*)

`\tl_if_eq:VnTF`

A useful variant for comparing the values of token list variables with token lists.

```
\cs_generate_variant:Nn \tl_if_eq:nnTF {VnTF}
```

(*End of definition for* `\tl_if_eq:VnTF`*.*)

`\l__avm_in_first_column`

A boolean to check whether we are in the first column (value `true`) or in the second (value `false`).

```
\bool_new:N \l__avm_in_first_column
```

(*End of definition for* `\l__avm_in_first_column`*.*)

`\__avm_init_first_column:`
`\__avm_init_second_column:`
`\__avm_init_single_column:`

These macros apply the settings for the columns in a (sub-)structure. They take care of font selection and report the currently active column back to the system. Knowing which column is active is important when closing the (sub-)structure. If the structure is closed without a second column present, we need to skip back `2\tabcolsep`. (This does not apply to the case of vector structures, which are handled without this check.)

```
\cs_new:Nn \__avm_init_first_column:
  {
    \bool_set_true:N \l__avm_in_first_column
    \normalfont\__avm_font_attribute:
  }

\cs_new:Nn \__avm_init_second_column:
  {
    \bool_set_false:N \l__avm_in_first_column
    \normalfont\__avm_font_value:
  }

\cs_new:Nn \__avm_init_single_column:
  {
    \normalfont\__avm_font_attribute:
  }

```

(*End of definition for* `\__avm_init_first_column:`*,* `\__avm_init_second_column:`*, and* `\__avm_init_single_column:`*.*)

`\__avm_deinit_first_column:`
`\__avm_deinit_second_column:`

These commands control settings that are applied after each column is exited. The single check here is whether italics is currently in use. If it is, the the italic correction is automatically applied. This replaces the user-configurable setting `apptovalues` from previous versions.

```

\tl_const:Nn \l__avm_italics_tl {it}

\cs_new:Nn \__avm_deinit_first_column:
  {
    \tl_if_eq:NNT \f@shape \l__avm_italics_tl {\/}
  }

\cs_new:Nn \__avm_deinit_second_column:
  {
    \tl_if_eq:NNT \f@shape \l__avm_italics_tl {\/}
  }

```

```
\cs_new:Nn \__avm_deinit_single_column:
  {
    \tl_if_eq:NNT \f@shape \l__avm_italics_tl {\/}
  }
```

(*End of definition for* `\__avm_deinit_first_column:` *and* `\__avm_deinit_second_column:`*.*)

`\__avm_kern_unused_columns:`

A helper macro to fill the horizontal space if a row is ended prematurely, i.e. if no `&` is present.

```
\cs_new:Nn \__avm_kern_unused_columns:
  {
    \bool_if:NTF \l__avm_in_first_column
      { \span\hspace*{-2\tabcolsep} }
      {  }
  }
```

(*End of definition for* `\__avm_kern_unused_columns:`*.*)

`\__avm_extra_skip:`

This function is used together with the delimiter replacements. It checks whether the delimiter is followed by a line break, in which case an extra skip is automatically inserted

```
\cs_new:Nn \__avm_extra_skip:
  {
    \peek_meaning_ignore_spaces:NTF \\ {\vspace*{\l__avm_extra_skip_dim}} {}
  }
```

(*End of definition for* `\__avm_extra_skip:`*.*)

`\__avm_module_begin:`
`\__avm_module_end:`
`etc.`

The replacement instructions for `\__avm_parse:n`. When option ⟨*`align = true`*⟩ (default), the structure has two columns. Vector structures are inserted if ⟨*`align = false`*⟩.

```
\cs_new:Nn \__avm_module_begin:
  {
    \bool_if:NTF \l__avm_align_bool
      {
        \begin{tabular}{@{}
                        >{\__avm_init_first_column:}l
                        <{\__avm_deinit_first_column:}
                        >{\__avm_init_second_column:}l
                        <{\__avm_deinit_second_column:}
                        @{}}
      }
      {
        \begin{tabular}{@{}
                       >{\__avm_init_single_column:}l
                       <{\__avm_deinit_single_column:}
                       @{}}
      }
  }
\cs_new:Nn \__avm_module_end:
  {
    \__avm_kern_unused_columns:
    \end{tabular}
  }

\cs_new:Nn \__avm_replace_ampersand:
  {
    \bool_if:NTF \l__avm_align_bool
      { \tl_build_put_right:Nn \l__avm_parsed_tl { & } }
      { \tl_build_put_right:Nn \l__avm_parsed_tl
        {
          \exp_not:n
            {
              \__avm_deinit_first_column:\skip_horizontal:N
              \dim_use:N \l__avm_singlesep_dim \__avm_init_second_column:
             }
         }
      }
  }
\cs_new:Nn \__avm_replace_lbrace:
  {
    \c_math_toggle_token\left\lbrace\__avm_module_begin:
  }
\cs_new:Nn \__avm_replace_rbrace:
  {
    \__avm_module_end:\right\rbrace\c_math_toggle_token\__avm_extra_skip:
  }
\cs_new:Nn \__avm_replace_lbrack:
  {
    \tl_build_put_right:Nn \l__avm_parsed_tl
      {
        \exp_not:n
          {
            \bool_if:NTF \l__avm_mode_bool
              {
                \c_math_toggle_token\left\lbrack\__avm_module_begin:
              }
              { [ }
          }
      }
  }
\cs_new:Nn \__avm_replace_rbrack:
  {
    \tl_build_put_right:Nn \l__avm_parsed_tl
      {
        \exp_not:n
          {
            \bool_if:NTF \l__avm_mode_bool
              {
                \__avm_module_end:\right\rbrack\c_math_toggle_token%
                \__avm_extra_skip:
              }
              { ] }
          }
      }
  }
\bool_if:NTF \l__avm_lfg_bool
  {
    \cs_new:Nn \__avm_replace_llbrack:
      {
        \c_math_toggle_token\left\lBrack\__avm_module_begin:
      }
    \cs_new:Nn \__avm_replace_rrbrack:
      {
        \__avm_module_end:\right\rBrack\c_math_toggle_token\__avm_extra_skip:
      }
  }
  {
    \cs_new:Nn \__avm_replace_llbrack:
      {
        \tl_build_put_right:Nn \l__avm_parsed_tl
          {
            \exp_not:n
              {
                \msg_warning:nn {avm}{lfgoptionmissing}
                \c_math_toggle_token\left.\__avm_module_begin:
              }
          }
      }
    \cs_new:Nn \__avm_replace_rrbrack:
      {
        \tl_build_put_right:Nn \l__avm_parsed_tl
          {
            \exp_not:n
            {
              \msg_warning:nn {avm}{lfgoptionmissing}
              \__avm_module_end:\right.\c_math_toggle_token\__avm_extra_skip:
            }
          }
      }
  }
\cs_new:Nn \__avm_replace_lparen:
  {
    \tl_build_put_right:Nn \l__avm_parsed_tl
      {
        \exp_not:n
          {
            \bool_if:NTF \l__avm_mode_bool
              {
                \c_math_toggle_token\left(\__avm_module_begin:
              }
              { ( }
          }
      }
  }
\cs_new:Nn \__avm_replace_rparen:
  {
    \tl_build_put_right:Nn \l__avm_parsed_tl
      {
        \exp_not:n
           {
             \bool_if:NTF \l__avm_mode_bool
               {
                 \__avm_module_end:\right)\c_math_toggle_token\__avm_extra_skip:
               }
               { ) }
           }
       }
  }
\cs_new:Nn \__avm_replace_langle:
  {
    \tl_build_put_right:Nn \l__avm_parsed_tl
      {
        \exp_not:n
          {
            \bool_if:NTF \l__avm_mode_bool
              {
                \c_math_toggle_token\left<\__avm_module_begin:
              }
              { < }
          }
      }
  }
\cs_new:Nn \__avm_replace_rangle:
  {
    \tl_build_put_right:Nn \l__avm_parsed_tl
      {
        \exp_not:n
          {
            \bool_if:NTF \l__avm_mode_bool
              {
                \__avm_module_end:\right>\c_math_toggle_token\__avm_extra_skip:
              }
              { > }
          }
      }
  }
\cs_new:Nn \__avm_replace_lframe:
  {
    \hbox_set:Nw \l__avm_fillmore_kay_box \group_begin:
    \c_math_toggle_token\__avm_module_begin:
  }
\cs_new:Nn \__avm_replace_rframe:
  {
    \__avm_module_end:\c_math_toggle_token\group_end:\hbox_set_end:
    \group_begin:
    \dim_set_eq:NN \fboxrule \l__avm_fillmore_kay_boxrule_dim
    \dim_set_eq:NN \fboxsep \l__avm_fillmore_kay_boxsep_dim
    \fbox{\box_use:N \l__avm_fillmore_kay_box}
    \group_end: \__avm_extra_skip:
  }
\cs_new:Nn \__avm_replace_plus:
  {
    \leavevmode\unskip\hbox{${}\oplus{}$}\ignorespaces
  }
\cs_new:Nn \__avm_replace_minus:
  {
    \leavevmode\unskip\hbox{${}\ominus{}$}\ignorespaces
  }
\cs_new:Nn \__avm_replace_circle:
  {
    \leavevmode\unskip\hbox{${}\bigcirc{}$}\ignorespaces
  }
```

(*End of definition for* `\__avm_module_begin:` *,* `\__avm_module_end:` *, and etc..*)

\tag
\type
\punk
\node
\id

```
\cs_new:Npn \__avm_controls_tag:n #1
  { \fboxsep.25ex\fboxrule.4pt\fbox{\normalfont\__avm_font_tag: #1} }
\cs_new:Npn \__avm_controls_type:n #1
  { \c_group_begin_token\normalfont\__avm_font_type: #1\c_group_end_token }
\cs_new_protected:Npn \__avm_controls_type_starred:n #1
  {
    \bool_set_false:N \l__avm_in_first_column
    \normalfont\__avm_font_type: #1
    \bool_if:NTF \l__avm_align_bool
      { \__avm_deinit_second_column:\span\hspace*{-2\tabcolsep} }
      { \__avm_deinit_single_column:}
    \peek_meaning_ignore_spaces:NTF \\ {} {\\}
  }
\cs_new_protected:Npn \__avm_controls_punk:nn #1 #2
  {
    \bool_set_false:N \l__avm_in_first_column
    \normalfont\c_group_begin_token\__avm_font_attribute:#1%
    \c_group_end_token\hspace{2\tabcolsep}%
    \c_group_begin_token\__avm_font_value: #2\c_group_end_token%
    \__avm_deinit_second_column:\span\hspace*{-2\tabcolsep}
    \peek_charcode_ignore_spaces:NTF \\ {} {\\}
  }

\cs_new:Nn \__avm_mode_switch:
  {
    \bool_set_inverse:N \l__avm_mode_bool
    \bool_if:NTF \l__avm_mode_bool
      {
        \DeclareDocumentCommand{\{}{}{ \__avm_replace_lbrace: }
        \DeclareDocumentCommand{\}}{}{ \__avm_replace_rbrace: }
        \DeclareDocumentCommand{\[}{}{ \__avm_replace_llbrack: }
        \DeclareDocumentCommand{\]}{}{ \__avm_replace_rrbrack: }
        \DeclareDocumentCommand{\+}{}{ \__avm_replace_plus: }
        \DeclareDocumentCommand{\-}{}{ \__avm_replace_minus: }
      }
      {
        \DeclareCommandCopy{\{}{\__avm_old_lbrace_store:}
        \DeclareCommandCopy{\}}{\__avm_old_rbrace_store:}
        \DeclareCommandCopy{\[}{\__avm_old_llbrack_store:}
        \DeclareCommandCopy{\]}{\__avm_old_rrbrack_store:}
        \DeclareCommandCopy{\+}{\__avm_old_plus_store:}
        \DeclareCommandCopy{\-}{\__avm_old_minus_store:}
      }
  }

```

```
\cs_new:Nn \__avm_initialise_document_commands:
 {
  \DeclareCommandCopy{\__avm_old_lbrace_store:}{\{}
  \DeclareCommandCopy{\__avm_old_rbrace_store:}{\}}
  \DeclareCommandCopy{\__avm_old_llbrack_store:}{\[}
  \DeclareCommandCopy{\__avm_old_rrbrack_store:}{\]}
  \DeclareCommandCopy{\__avm_old_plus_store:}{\+}
  \DeclareCommandCopy{\__avm_old_minus_store:}{\-}
  \def\arraystretch{\tl_use:N \l__avm_arraystretch_tl}
  \dim_set_eq:NN \tabcolsep \l__avm_tabcolsep_dim
  \int_set_eq:NN \delimiterfactor \l__avm_delimfactor_int
  \dim_set_eq:NN \delimitershortfall \l__avm_delimshortfall_dim
  \DeclareDocumentCommand{\shuffle}{}{ \__avm_replace_shuffle: }
  \DeclareDocumentCommand{\lframe}{}{ \__avm_replace_lframe: }
  \DeclareDocumentCommand{\rframe}{}{ \__avm_replace_rframe: }
  \DeclareDocumentCommand{\tag}{m}{ \__avm_controls_tag:n {##1} }
  \DeclareDocumentCommand{\0}{}{ \__avm_controls_tag:n {0} }
  \DeclareDocumentCommand{\1}{}{ \__avm_controls_tag:n {1} }
  \DeclareDocumentCommand{\2}{}{ \__avm_controls_tag:n {2} }
  \DeclareDocumentCommand{\3}{}{ \__avm_controls_tag:n {3} }
  \DeclareDocumentCommand{\4}{}{ \__avm_controls_tag:n {4} }
  \DeclareDocumentCommand{\5}{}{ \__avm_controls_tag:n {5} }
  \DeclareDocumentCommand{\6}{}{ \__avm_controls_tag:n {6} }
  \DeclareDocumentCommand{\7}{}{ \__avm_controls_tag:n {7} }
  \DeclareDocumentCommand{\8}{}{ \__avm_controls_tag:n {8} }
  \DeclareDocumentCommand{\9}{}{ \__avm_controls_tag:n {9} }
  \DeclareDocumentCommand{\type}{s m}
   {
    \IfBooleanTF { ##1 }
     { \__avm_controls_type_starred:n {##2} }
     { \__avm_controls_type:n {##2} }
   }
  \DeclareDocumentCommand{\punk}{m m}{ \__avm_controls_punk:nn {##1}{##2} }
  \DeclareDocumentCommand{\id}{m m}
   {%
    \hcoffin_set:Nw \l_tmpa_coffin
     \bgroup
     \def\arraystretch{.5}
     \begin{tabular}[b]{@{}>{$\scriptstyle}i<{$}@{}}
     ##1
     \end{tabular}
     \egroup
     \hcoffin_set_end:
     \hcoffin_set:Nw \l_tmpb_coffin ##2 \hcoffin_set_end:
     \tl_if_eq:VnTF \l__avm_id_position_tl {south-west}
      {%
       \coffin_join:NnnNnnnn \l_tmpb_coffin {l}{H}
       \l_tmpa_coffin {r}{H}{ 0pt }{ -\coffin_dp:N \l_tmpb_coffin }
      }
      {%
       \tl_if_eq:VnTF \l__avm_id_position_tl {south-east}
       {%
        \coffin_join:NnnNnnnn \l_tmpb_coffin {l}{H}
        \l_tmpa_coffin {l}{H}{ \coffin_wd:N \l_tmpb_coffin }
```

```
                                     { -\coffin_dp:N \l_tmpb_coffin }
              }
              {
                \msg_error:nn {avm}{idpositionunknown}
              }
            }
        \coffin_typeset:Nnnnn \l_tmpb_coffin {l}{vc}{0pt}{0pt}
      }
    \DeclareDocumentEnvironment{scope}{ +m }
      {
        \group_begin:
        \keys_set:nn { avm } { ##1 }
        \ignorespaces
      }
      {
        \group_end:
      }
```

The last of the bunch is only loaded if Ti*k*Z is loaded as well:

```
    \bool_if:NT \l__avm_tikz_bool
      {
        \tl_if_eq:VnTF \l__avm_picture_name_tl {automatic}
          {
            \stepcounter{l__avm_picture_counter}
            \tl_set:Nn \l__avm_picture_name_prefix_tl
              {avm-\tl_use:N \thel__avm_picture_counter}
          }
          {
            \tl_set_eq:NN \l__avm_picture_name_prefix_tl \l__avm_picture_name_tl
          }
        \DeclareDocumentCommand{\node}{m m}
          {
            \tikz [remember~picture,
                   baseline=(\l__avm_picture_name_prefix_tl-##1.base)]
            \node [inner~sep=0pt] (\l__avm_picture_name_prefix_tl-##1)
                                  {\strut ##2};
          }
      }
  }
```

(*End of definition for* `\tag` *and others. These functions are documented on page 3.*)

`\__avm_parse:n`

Finally, the parser. It is build on `\@@_act:NNNnn` from `l3tl` (see the sub-section *Token by token changes*). Many thanks to Phelype Oleinik for help on this, and in particular on help with expansion.

```
\cs_new:Npn \__avm_parse:n #1
  {
    \group_align_safe_begin:
    \tl_build_begin:N \l__avm_parsed_tl
    \tl_build_put_right:Nn \l__avm_parsed_tl { \exp_not:n {\ignorespaces} }
    \tl_analysis_map_inline:nn { #1 }
      {
        \int_case:nnF { ##2 }
          {
            { `&  }{ \__avm_replace_ampersand: }
          { `[  }{ \__avm_replace_lbrack: }
          { `]  }{ \__avm_replace_rbrack: }
          { `(  }{ \__avm_replace_lparen: }
          { `)  }{ \__avm_replace_rparen: }
          { `<  }{ \__avm_replace_langle: }
          { `>  }{ \__avm_replace_rangle: }
          { \l__avm_mode_switch_character_int }
            {
              \tl_build_put_right:Nn \l__avm_parsed_tl
                { \exp_not:n { \__avm_mode_switch: } }
            }
        }
        {
          \tl_build_put_right:Nn \l__avm_parsed_tl { ##1 }
        }
    }
  \tl_build_end:N \l__avm_parsed_tl
  \tl_set:Nx \l__avm_parsed_tl {\l__avm_parsed_tl}
  \tl_use:N \l__avm_parsed_tl
  \group_align_safe_end:
}

```

(*End of definition for* `\__avm_parse:n`.)

⟨/package⟩