The sdapsarray package*

Benjamin Berg
benjamin@sipsolutions.net

February 15, 2020

1 Documentation

Please refer to https://sdaps.org/class-doc for documentation.

2 Implementation

This package uses the LaTeX3 language internally, so we need to enable it.

```
% We need at least 2011-08-23 for \keys_set_known:nnN
\RequirePackage{expl3}[2011/08/23]
%\RequirePackage{xparse}
\ExplSyntaxOn
```

And we need a number of other packages.

```
\ExplSyntaxOff

\RequirePackage{xparse}
\RequirePackage{sdapsbase}


\ExplSyntaxOn

```

2.1 Tempfile handling

This is a bit weird, but under some conditions we need extended information about each row in the document (for page break detection). As it makes little to no sense to load all this information into memory at start we use two temporary files instead. As these files should not change their content for reruns (that would break e.g. latexmk) we copy the "tic" file into "toc" so that "toc" can be read while "tic" is being re-written. We then go on to define a macro which will read a single line and return true if it is different from the previous line. This is the

*This document corresponds to sdapsarray v0.1, dated 2015/04/10.

indicator that a page/column break has happened and the row header needs to be inserted.

```
13
14 % This will change in early 2018, but the new code apparently does not
15 % provide the old name of the definition.
16 % i.e. this is a bad hack and should be removed latest in 2019 or so
17 \cs_if_exist:NF \ior_str_get:NN { \cs_set_eq:Nc \ior_str_get:NN { ior_get_str:NN } }
18
19 \bool_new:N \g__sdaps_array_info_open
20 \bool_gset_false:N \g__sdaps_array_info_open
21 \iow_new:N \g__sdaps_array_info_iow
22 \ior_new:N \g__sdaps_array_info_ior
23
24 \cs_generate_variant:Nn \int_set:Nn { Nf }
25 \cs_generate_variant:Nn \coffin_join:NnnNnnnn { NnVNnVnn }
26
27 \cs_new_protected_nopar:Nn \_sdaps_array_open_tmpfiles:
28 {
29   % Guard against being executed multiple times
30   \bool_if:NF \g__sdaps_array_info_open {
31     \bool_gset_true:N \g__sdaps_array_info_open
32
33     % Also ensures toc file exists (i.e. is readable)
34     \iow_open:Nn \g__sdaps_array_info_iow { \c_sys_jobname_str .sdapsarraytoc }
35     \file_if_exist:nTF { \c_sys_jobname_str .sdapsarraytic } {
36       % Copy into toc file, then open that.
37       \ior_open:Nn \g__sdaps_array_info_ior { \c_sys_jobname_str .sdapsarraytic }
38
39       \ior_str_map_inline:Nn \g__sdaps_array_info_ior { \iow_now:Nn \g__sdaps_array_info_iow { #
40       \ior_close:N \g__sdaps_array_info_ior
41
42       \ior_open:Nn \g__sdaps_array_info_ior { \c_sys_jobname_str .sdapsarraytoc }
43     } {
44     }
45     \iow_close:N \g__sdaps_array_info_iow
46
47     \iow_open:Nn \g__sdaps_array_info_iow { \c_sys_jobname_str .sdapsarraytic }
48   }
49 }
50
51 \tl_new:N \g__sdaps_array_last_row_tl
52 \tl_gset_eq:NN \g__sdaps_array_last_row_tl \c_empty_tl
53
54 \cs_new_protected_nopar:Nn \_sdaps_array_check_insert_header:N
55 {
56   \bool_gset_eq:NN #1 \c_false_bool
57   \ior_if_eof:NF \g__sdaps_array_info_ior {
58     \ior_str_get:NN \g__sdaps_array_info_ior \l_tmpa_tl
59
```

```
    \tl_if_eq:NNF \g__sdaps_array_last_row_tl \c_empty_tl {
      \tl_if_eq:NNF \g__sdaps_array_last_row_tl \l_tmpa_tl {
        \bool_gset_eq:NN #1 \c_true_bool
      }
    }
    \tl_gset_eq:NN \g__sdaps_array_last_row_tl \l_tmpa_tl
  }
}

```

2.2 Initialization

Global definitions and penalty constant.

```

\prop_new:N \g__sdaps_array_layouter_prop

% XXX: Penalty in between rows. After the header a nobreak is inserted, but
%      do we want special penalties elsewhere (preventing orphans/widows?)
\int_new:N \g_sdaps_array_row_penalty_tl
\int_gset:Nn \g_sdaps_array_row_penalty_tl { 10 }

```

2.3 Initialization

Define some routines to store width information for columns (across builds).

```

\tl_new:N \g_sdaps_array_shared_data_tl
\tl_new:N \g_sdaps_array_stored_data_tl
\tl_new:N \g_sdaps_array_local_data_tl
\tl_new:N \g_sdaps_array_local_data_new_tl
\prop_new:N \g__sdaps_array_stored_data_prop
\prop_new:N \g__sdaps_array_shared_data_prop
\prop_new:N \g__sdaps_array_local_data_prop

\cs_generate_variant:Nn \prop_item:Nn { NV }

\cs_new_protected_nopar:Nn \_sdaps_array_load_data:
{
  \tl_gset:Nx \g_sdaps_array_stored_data_tl { \prop_item:NV \g__sdaps_array_stored_data_prop \l
  \tl_gset:Nx \g_sdaps_array_shared_data_tl { \prop_item:NV \g__sdaps_array_shared_data_prop \l
  \tl_gset:Nx \g_sdaps_array_local_data_tl { \prop_item:NV \g__sdaps_array_local_data_prop \l__
}

\cs_new_protected_nopar:Nn \_sdaps_array_store_data:
{
  % Do not overwrite the "stored" data that we have right now.
  \prop_gput:NVV \g__sdaps_array_shared_data_prop \l__sdaps_array_global_name_tl \g_sdaps_array

```

```
  \immediate\write\@auxout{\exp_not:n{\sdapsarrayloadstoreddata}{\l__sdaps_array_global_name_tl
  \tl_if_empty:NF \g_sdaps_array_local_data_new_tl {
    \immediate\write\@auxout{\exp_not:n{\sdapsarrayloadlocaldata}{\l__sdaps_array_local_name_tl
  }
}

% Define for loading sdaps code in aux file
\cs_new_protected_nopar:Nn \sdaps_array_load_stored_data:nn {
  \prop_gput:Nnn \g__sdaps_array_stored_data_prop { #1 } { #2 }
}
\cs_new_eq:NN \sdapsarrayloadstoreddata \sdaps_array_load_stored_data:nn

% Define for loading sdaps code in aux file
\cs_new_protected_nopar:Nn \sdaps_array_load_local_data:nn {
  \prop_gput:Nnn \g__sdaps_array_local_data_prop { #1 } { #2 }
}
\cs_new_eq:NN \sdapsarrayloadlocaldata \sdaps_array_load_local_data:nn

```

2.4 Array Layouter

2.4.1 User facing macros

```

\int_new:N \g__sdaps_array_current_id_int
\tl_new:N \l__sdaps_array_global_name_tl
\tl_new:N \l__sdaps_array_local_name_tl

\cs_generate_variant:Nn \keys_set:nn { nf }
\cs_new_protected_nopar:Nn \sdaps_array_begin:nn
{
 \tl_set:Nx \l__sdaps_array_local_name_tl { sdapsarray \int_use:N\g__sdaps_array_current_id_int
  \int_gincr:N \g__sdaps_array_current_id_int
  \tl_if_empty:nTF { #2 } {
    \tl_set:NV \l__sdaps_array_global_name_tl \l__sdaps_array_local_name_tl
  } {
    \tl_set:Nx \l__sdaps_array_global_name_tl { #2 }
  }

  \_sdaps_array_load_data:

  \keys_set:nf { sdaps / array } { \prop_item:Nn \g__sdaps_array_layouter_prop { #1 } }

  % Force cell layouter instead of column header layouter
  \bool_if:NT \l_sdaps_sdapsarray_no_header_bool {
    \tl_set:NV \l__sdaps_array_colhead_tl \l__sdaps_array_cell_tl
  }

  \_sdaps_array_open_tmpfiles:

```

```
  \tl_gset_eq:NN \g__sdaps_array_last_row_tl \c_empty_tl
  \bool_gset_true:N \g_sdaps_array_first_row_bool

  \l__sdaps_array_begin_tl
}
\cs_generate_variant:Nn \sdaps_array_begin:nn { Vn }
\cs_generate_variant:Nn \sdaps_array_begin:nn { VV }
\cs_generate_variant:Nn \sdaps_array_begin:nn { nV }

\cs_new_protected_nopar:Nn \sdaps_array_begin:n
{
  \sdaps_array_begin:nn { #1 } { }
}

\cs_new_protected_nopar:Nn \sdaps_array_row_start:
{
  \l__sdaps_array_row_start_tl
}

\cs_new_protected_nopar:Nn \_sdaps_array_assign_use: {
    \box_use:N \l_tmpa_box
  \egroup
}

\cs_new_protected_nopar:Npn \sdaps_array_assign_colhead:Nw #1
{
  \l__sdaps_array_colhead_tl #1
}

\cs_new_protected_nopar:Npn \sdaps_array_colhead:w
{
  \bgroup
    \l__sdaps_array_colhead_tl \l_tmpa_box \bgroup
    \group_insert_after:N\_sdaps_array_assign_use:
    % swallow group opening token
    \tex_let:D\next=
}

\cs_new_protected:Npn \sdaps_array_assign_rowhead:Nw #1
{
  \l__sdaps_array_rowhead_tl #1
}

\cs_new_protected:Npn \sdaps_array_rowhead:w
{
  \bgroup
    \l__sdaps_array_rowhead_tl \l_tmpa_box \bgroup
    \group_insert_after:N\_sdaps_array_assign_use:
    % swallow group opening token
    \tex_let:D\next=
```

```
}

\cs_new_protected_nopar:Npn \sdaps_array_assign_cell:Nw #1
{
  \l__sdaps_array_cell_tl #1
}

\cs_new_protected_nopar:Npn \sdaps_array_cell:w
{
  \bgroup
    \l__sdaps_array_cell_tl \l_tmpa_box \bgroup
    \group_insert_after:N\_sdaps_array_assign_use:
    % swallow group opening token
    \tex_let:D\next=
}

% XXX: Could this live in local scope instead?
\box_new:N \g__sdaps_array_header_box
\dim_new:N \g__sdaps_array_header_dim

\cs_new_protected_nopar:Nn \_sdaps_array_calc_interlineskip:nnN
{
  \dim_compare:nNnTF { #1 } > { -1000pt } {
    \skip_set:Nn #3 { \baselineskip - #1 - #2 }
    \dim_compare:nNnF { #3 } > { \lineskiplimit } {
      \skip_set:Nn #3 { \lineskip }
    }
  } {
    \skip_set:Nn #3 { 0pt }
  }
  \skip_set:Nn #3 { #3 + \l_sdaps_sdapsarray_rowsep_dim }
}

\cs_new_protected_nopar:Nn \sdaps_array_row:NN
{
  \if_mode_vertical:
  \else:
    \msg_error:nn { sdapsarray } { wrong_mode }
  \fi

  % XXX: \l_tmpa_dim is the height to the first baseline in the box. Note that
  %      we use the real baseline in the case of the header row!
  \l__sdaps_array_row_tl #1 #2 \l_tmpb_box \l_tmpa_dim

  \bool_if:nTF { \g_sdaps_array_first_row_bool && !\l_sdaps_sdapsarray_no_header_bool } {
    % Stow away the box for later use
    \box_gset_eq:NN \g__sdaps_array_header_box \l_tmpb_box
    \dim_gset:Nn \g__sdaps_array_header_dim { \box_ht:N \g__sdaps_array_header_box + \box_dp:N
  } {
    % Pagebreak detection (not needed for header row)
```

```
  \_sdaps_array_check_insert_header:N \g_tmpa_bool

  \hbox_set:Nn \l_tmpb_box {
    \pdfsavepos
    \iow_shipout_x:Nn \g__sdaps_array_info_iow {
      \thepage,
      \the\pdflastxpos
    }
    \box_use:N \l_tmpb_box
  }
}

\bool_if:nTF { \g_sdaps_array_first_row_bool || \l_sdaps_sdapsarray_no_header_bool } {
  \_sdaps_array_calc_interlineskip:nnN { \prevdepth } { \l_tmpa_dim } \l_tmpa_skip
  \nointerlineskip
  \skip_vertical:n { \l_tmpa_skip }

  % However do not insert the rowskip in the case of the first line.
  % We rely on surrounding code to insert proper spacing before/after
  % the environment.
  \bool_if:NT \g_sdaps_array_first_row_bool {
    \skip_vertical:n { - \l_sdaps_sdapsarray_rowsep_dim }
  }

  \box_use:N \l_tmpb_box
  % For the header, insert a \nobreak, otherwise the normal inter-row penalty
  \bool_if:NTF \l_sdaps_sdapsarray_no_header_bool {
    \penalty\int_use:N\g_sdaps_array_row_penalty_tl
  } {
    \nobreak
  }

  \bool_gset_false:N \g_sdaps_array_first_row_bool
} {
  % The idea is simple. Before every line the header is re-inserted (either
  % the real one or an empty box with the same dimensions). In the case that
  % there is *no* page break we insert a corresponding negative skip so that
  % the resulting skip (including interline skip) is exactly the normal
  % interline skip between the rows.
  % In the case that the skip *is* discarded we end up with the header box
  % and the normal interline skip between the new row and the header. i.e.:
  %   skip = 1 * ( interlineskip_for_row - interlineskip_for_header - header_height - header_
  % or
  %   skip = 0 * ( interlineskip_for_row - interlineskip_for_header - header_height - header_

  % Calculate interlineksip_for_row and interlineskip_for_header
  \_sdaps_array_calc_interlineskip:nnN { \prevdepth } { \l_tmpa_dim } \l_tmpa_skip
  \_sdaps_array_calc_interlineskip:nnN { \box_dp:N \g__sdaps_array_header_box } { \l_tmpa_dim
  \nointerlineskip
  \skip_vertical:n { \l_tmpa_skip - \l_tmpb_skip }
```

```
    \kern -\g__sdaps_array_header_dim

    % Inser the real or fake box
    \bool_if:NTF \g_tmpa_bool {
      \box_use:N \g__sdaps_array_header_box
    } {
     \hrule height \box_ht:N \g__sdaps_array_header_box depth \box_dp:N \g__sdaps_array_header_
    }
    \nobreak
    % Insert the calculated interline skip (in the same way the TeX would do it.
    \nointerlineskip
    \skip_vertical:N \l_tmpb_skip

    \box_use:N \l_tmpb_box

    % And insert the sdapsarray sepecific inter row penalty.
    \penalty\int_use:N\g_sdaps_array_row_penalty_tl
  }
}

\cs_new_protected_nopar:Nn \sdaps_array_end:
{
  \l__sdaps_array_end_tl
  \box_gclear:N \g__sdaps_array_header_box
}
```

2.4.2 Common Layouter Macros

```
\tl_new:N \l__sdaps_array_begin_tl
\tl_new:N \l__sdaps_array_row_start_tl
\tl_new:N \l__sdaps_array_colhead_tl
\tl_new:N \l__sdaps_array_rowhead_tl
\tl_new:N \l__sdaps_array_cell_tl
\tl_new:N \l__sdaps_array_row_tl
\tl_new:N \l__sdaps_array_end_tl

\keys_define:nn { sdaps / array }
{
  begin        .tl_set:N   = \l__sdaps_array_begin_tl,
  row_start    .tl_set:N   = \l__sdaps_array_row_start_tl,
  colhead      .tl_set:N   = \l__sdaps_array_colhead_tl,
  rowhead      .tl_set:N   = \l__sdaps_array_rowhead_tl,
  cell         .tl_set:N   = \l__sdaps_array_cell_tl,
  row          .tl_set:N   = \l__sdaps_array_row_tl,
  end          .tl_set:N   = \l__sdaps_array_end_tl,
}


\seq_new:N \g_sdaps_array_overhangs_left_seq
```

```
\seq_new:N \g_sdaps_array_overhangs_right_seq
\seq_new:N \g_sdaps_array_shared_colwidths_seq
\seq_new:N \g_sdaps_array_stored_colwidths_seq

\cs_new_protected_nopar:Npn \_sdaps_array_rowhead_default:Nw #1
{
  \tl_if_empty:NTF \g_sdaps_array_local_data_tl {
      \tl_if_empty:NTF \g_sdaps_array_local_data_new_tl {
        \dim_set:Nn \l_tmpa_dim { 0.5 \hsize }
      } {
        \dim_set:Nn \l_tmpa_dim { \g_sdaps_array_local_data_new_tl }
      }
  } {
    \dim_set:Nn \l_tmpa_dim { \g_sdaps_array_local_data_tl }
  }
  % \vbox_set_top:Nw is still missing as of 2017-08-11
  \tex_setbox:D #1 \tex_vtop:D \bgroup
    \sdaps_if_rtl:TF {
      \raggedright
    } {
      \raggedleft
    }
    \hsize=\dim_use:N\l_tmpa_dim
    \group_begin:\bgroup
    \group_insert_after:N \group_end:
    \group_insert_after:N \egroup
    % swallow group opening token
    \tex_let:D\next=
}

\cs_new_protected_nopar:Npn \_sdaps_array_cell_default:Nw #1
{
  \hbox_set:Nw #1 \bgroup
    % swallow group opening token
    \group_insert_after:N \hbox_set_end:
    \tex_let:D\next=
}

\cs_new:Nn \_sdaps_array_cell_rotated_end: {
    \hbox_set_end:
    \dim_set:Nn \l_tmpa_dim { \box_ht:N \l_tmpa_box }
    \dim_set:Nn \l_tmpb_dim { \box_dp:N \l_tmpa_box }

    \dim_set:Nn \l_tmpa_dim { \l_sdaps_sdapsarray_angle_sine_tl \l_tmpa_dim }
    \dim_set:Nn \l_tmpb_dim { \l_sdaps_sdapsarray_angle_sine_tl \l_tmpb_dim }

    \box_rotate:Nn \l_tmpa_box { \l_sdaps_sdapsarray_angle_int }

    % We want the baseline of the box to be centered, that only works if we
    % leave the same space both ways.
```

```
    % That is not ideal, but we cannot move the cell content accordingly.
    \dim_set:Nn \l_tmpb_dim { \dim_max:nn { \l_tmpa_dim } { \l_tmpb_dim } }
    \skip_horizontal:n { \l_tmpb_dim }
    \rlap{
      \skip_horizontal:n { -\l_tmpa_dim }
      \box_use:N \l_tmpa_box
    }
    \skip_horizontal:n { \l_tmpb_dim }
    % dummy skip that will be removed again by other code
    \skip_horizontal:n { 0pt }

    \dim_set:Nn \l_tmpa_dim { \l_tmpa_dim + \l_tmpb_dim }

    \dim_set:Nn \l_tmpb_dim { \box_wd:N \l_tmpa_box }
    \dim_set:Nn \l_tmpa_dim { \dim_max:nn { 0pt } { \l_tmpb_dim - \l_tmpa_dim } }

    \seq_gpush:Nn \g_sdaps_array_overhangs_left_seq { 0pt }
    \seq_gpush:Nx \g_sdaps_array_overhangs_right_seq { \dim_use:N \l_tmpa_dim }
  \egroup
  \hbox_set_end:
}

% Only sane for header row
\cs_new_protected_nopar:Npn \_sdaps_array_cell_rotated:Nw #1
{
  \hbox_set:Nw #1 \bgroup
    \hbox_set:Nw \l_tmpa_box
    \bgroup
    \group_insert_after:N \_sdaps_array_cell_rotated_end:
    % swallow group opening token
    \tex_let:D\next=
}

% XXX: A parbox layouter with fixed width would be nice
%\cs_new_protected_nopar:Nn \sdaps_array_cell_fixed:n {}


\cs_new_protected_nopar:Nn \_sdaps_array_begin_default:
{
  \tl_if_empty:NTF \g_sdaps_array_shared_data_tl {
    \seq_clear:N \g_sdaps_array_shared_colwidths_seq
  } {
    \seq_gset_split:NnV \g_sdaps_array_shared_colwidths_seq { ~ } \g_sdaps_array_shared_data_tl
  }
  \tl_if_empty:NTF \g_sdaps_array_stored_data_tl {
    \seq_clear:N \g_sdaps_array_stored_colwidths_seq
  } {
    \seq_gset_split:NnV \g_sdaps_array_stored_colwidths_seq { ~ } \g_sdaps_array_stored_data_tl
  }
}
```

```
443
444 \cs_new_protected_nopar:Nn \_sdaps_array_end_default:
445 {
446   \tl_gset:Nx \g_sdaps_array_shared_data_tl { \seq_use:Nn \g_sdaps_array_shared_colwidths_seq {
447   \tl_gset:Nx \g_sdaps_array_stored_data_tl { \seq_use:Nn \g_sdaps_array_stored_colwidths_seq {
448
449   % Clear the global sequences, to save memory
450   \seq_gclear:N \g_sdaps_array_overhangs_left_seq
451   \seq_gclear:N \g_sdaps_array_overhangs_right_seq
452   \seq_gclear:N \g_sdaps_array_shared_colwidths_seq
453   \seq_gclear:N \g_sdaps_array_stored_colwidths_seq
454
455   \_sdaps_array_store_data:
456 }
457
458 \cs_new_protected_nopar:Nn \_sdaps_array_row_start_default:
459 {
460   \seq_gclear:N \g_sdaps_array_overhangs_left_seq
461   \seq_gclear:N \g_sdaps_array_overhangs_right_seq
462 }
463
464 \cs_new_protected_nopar:Nn \_sdaps_array_row:NNNN
465 {
466   % #1: A vbox with baseline on the *first* item containing the row header
467   %     (\vtop in plain TeX).
468   % #2: Data cells packed into an hbox. Each of these needs to be set to the
469   %     correct width and inserted.
470   % #3: The box register to store the resulting hbox in. The depth of this box
471   %     needs to be correct to calculate the interline glue to the following
472   %     row.
473   % #4: A dim register to store the height of the box in for the purpose of
474   %     calculating the interline glue in front of the produced row.
475   %
476   % { \dim_use:N\@totalleftmargin } { \dim_use:N\linewidth }
477   % The macro should create an hbox which is exactly \linewidth wide and also
478   % contains internal indentation by \@totalleftmargin into the box register #3.
479   % The box will be used while in vertical mode and may% be inserted multiple
480   % times in the case of the header row.
481   % To simplify the iteration it is guaranteed that the data cell boxes are not
482   % completely empty. This means the code can simply unbox until it sees a box
483   % that is void.
484
485   \seq_gclear:N \g_tmpa_seq
486
487   % Insert the boxes into a local hbox to work with them
488   \hbox_set:Nn #2 {
489     \hbox_unpack:N #2
490
491     % Handle the overhang, note that we modify the \g_sdaps_array_overhangs_right_seq locally o
492     \seq_pop:NNTF \g_sdaps_array_overhangs_right_seq \l_tmpa_tl {
```

```
    \dim_set:Nn \l_tmpb_dim { \l_tmpa_tl }
  } {
    \dim_set:Nn \l_tmpb_dim { 0pt }
  }
  % Implicit "last" column that contains the overhang
  \seq_gpop:NNTF \g_sdaps_array_shared_colwidths_seq \l_tmpa_tl {
    \dim_set:Nn \l_tmpa_dim { \l_tmpa_tl }
  } {
    \dim_set:Nn \l_tmpa_dim { 0pt }
  }
  \dim_set:Nn \l_tmpa_dim { \dim_max:nn { \l_tmpa_dim } { \l_tmpb_dim } }


  % MAX with stored values (NOTE: sequence only modified in local scope)
  \seq_pop:NNTF \g_sdaps_array_stored_colwidths_seq \l_tmpa_tl {
    \dim_set:Nn \l_tmpb_dim { \l_tmpa_tl }
  } {
    \dim_set:Nn \l_tmpb_dim { 0pt }
  }
  % Store value from this run, and then calculate max with previous run
  \seq_gput_right:Nx \g_tmpa_seq { \dim_use:N \l_tmpa_dim }
  \dim_set:Nn \l_tmpa_dim { \dim_max:nn { \l_tmpa_dim } { \l_tmpb_dim } }


  % Insert the overhang space
  \hbox_set:Nn \l_tmpa_box { \skip_horizontal:n { \l_tmpa_dim } }

  % Now grab the first of the cells, and then loop over the rest
  \box_set_to_last:N #2
  \bool_do_while:nn { ! \box_if_empty_p:N #2 } {
    % Strip any trailing glue (i.e. space) coming from the user (for the
    % leading side we ensure that \tex_ignorespaces:D is called).
    % Note that this may remove e.g. a trailing \hfill from the user, the
    % user needs to work around that (in the same way as is required in e.g.
    % tabular).
    \hbox_set:Nn #2 { \hbox_unpack:N #2 \unskip }

    % Pop the target width for the current box (i.e. we don't globally
    % modify the clist here).
    \seq_gpop:NNTF \g_sdaps_array_shared_colwidths_seq \l_tmpa_tl {
      \dim_set:Nn \l_tmpa_dim { \l_tmpa_tl }
    } {
      \dim_set:Nn \l_tmpa_dim { 0pt }
    }
    % Calculate the maximum width of current and previous items
    \dim_set:Nn \l_tmpa_dim { \dim_max:nn { \box_wd:N #2 } { \l_tmpa_dim } }


    % MAX with stored values (NOTE: sequence only modified in local scope)
    \seq_pop:NNTF \g_sdaps_array_stored_colwidths_seq \l_tmpa_tl {
```

```
      \dim_set:Nn \l_tmpb_dim { \l_tmpa_tl }
    } {
      \dim_set:Nn \l_tmpb_dim { 0pt }
    }
    % Store value from this run, and then calculate max with previous run
    \seq_gput_right:Nx \g_tmpa_seq { \dim_use:N \l_tmpa_dim }
    \dim_set:Nn \l_tmpa_dim { \dim_max:nn { \l_tmpa_dim } { \l_tmpb_dim } }

    % Set the box into a new box with the correct width which contains fil
    % to center it.
    \hbox_set_to_wd:Nnn \l_tmpb_box \l_tmpa_dim { \hfil \hbox_unpack:N #2 \hfil }

    % This loops works backward, so attach the cell on the right side.
    % We used to make sure that it is layed out in order, but that is now
    % obsolete and doing it out of order is simpler in RTL mode.
    \sdaps_if_rtl:TF {
      % The boxes are shown on the page from LTR
      \hbox_set:Nn \l_tmpa_box { \box_use:N \l_tmpa_box \skip_horizontal:n { \l_sdaps_sdapsar
    } {
      \hbox_set:Nn \l_tmpa_box { \skip_horizontal:n { \l_sdaps_sdapsarray_colsep_dim } \box_u
    }

    % Grab next cell
    \box_set_to_last:N #2
  }

  % Get the coffin out of the nested scope by placing it into the box and
  % placing that into it again ...
  \box_use:N \l_tmpa_box
}
\hcoffin_set:Nn \l_tmpa_coffin { \box_use_drop:N #2 }

\seq_gconcat:NNN \g_sdaps_array_shared_colwidths_seq \g_tmpa_seq \g_sdaps_array_shared_colwid
\seq_gclear:N \g_tmpa_seq

% Calculate the space that is left for the header column
\dim_set:Nn \l_tmpa_dim { \linewidth - \coffin_wd:N \l_tmpa_coffin - 2\l_sdaps_sdapsarray_col
\tl_gset:Nx \g_sdaps_array_local_data_new_tl { \dim_use:N \l_tmpa_dim }

% TODO: The \hfil here is a hack to prevent a warning if the vbox is empty.
%       Unfortunately checking for an empty box does not work for some reason.
\dim_set:Nn \l_tmpb_dim { \box_ht:N #1 }
\sdaps_if_rtl:TF {
  \hcoffin_set:Nn \l_tmpb_coffin { \hbox_to_wd:nn \l_tmpa_dim { \hfil \vbox:n { \vbox_unpack_
  \tl_set:Nn \l_tmpa_tl { l }
  \tl_set:Nn \l_tmpb_tl { r }
} {
  \hcoffin_set:Nn \l_tmpb_coffin { \skip_horizontal:n { \l_sdaps_sdapsarray_colsep_dim } \hbo
  \tl_set:Nn \l_tmpa_tl { r }
  \tl_set:Nn \l_tmpb_tl { l }
```

```
  }
  \dim_set:Nn \l_tmpa_dim { \coffin_ht:N \l_tmpb_coffin }

  % If the first/last baseline differ then center the vbox, otherwise align the
  % baseline with the cells
  \dim_compare:nNnTF { \l_tmpa_dim } = { \l_tmpb_dim } {
    \coffin_join:NnVNnVnn \l_tmpb_coffin { H } \l_tmpa_tl \l_tmpa_coffin { H } \l_tmpb_tl { \l_
    \dim_set:Nn #4 { \coffin_ht:N \l_tmpb_coffin }
  } {
    \coffin_join:NnVNnVnn \l_tmpb_coffin { vc } \l_tmpa_tl \l_tmpa_coffin { vc } \l_tmpb_tl { \
    % XXX: Assume that the header is higher than the content cells
    \dim_set:Nn #4 { \l_tmpb_dim }
  }

  \hbox_set:Nn #3 { \skip_horizontal:N \@totalleftmargin \coffin_typeset:Nnnnn \l_tmpb_coffin {
}


\prop_gput:Nnn \g__sdaps_array_layouter_prop { default } {
  begin = { \_sdaps_array_begin_default: },
  row_start = { \_sdaps_array_row_start_default: },
  rowhead = { \_sdaps_array_rowhead_default:Nw },
  colhead = { \_sdaps_array_cell_default:Nw },
  cell = { \_sdaps_array_cell_default:Nw },
  row = { \_sdaps_array_row:NNNN },
  end = { \_sdaps_array_end_default: },
}


\prop_gput:Nnn \g__sdaps_array_layouter_prop { rotated } {
  begin = { \_sdaps_array_begin_default: },
  row_start = { \_sdaps_array_row_start_default: },
  rowhead = { \_sdaps_array_rowhead_default:Nw },
  colhead = { \_sdaps_array_cell_rotated:Nw },
  cell = { \_sdaps_array_cell_default:Nw },
  row = { \_sdaps_array_row:NNNN },
  end = { \_sdaps_array_end_default: },
}
```

2.5 Exporting a tabular/array like environment

2.5.1 Helper required for the environment

```
\bool_new:N \g_sdaps_array_first_row_bool
\bool_new:N \l__sdaps_sdapsarray_in_top_group_bool
\bool_new:N \l__sdaps_sdapsarray_have_content_bool
\bool_new:N \l_sdaps_sdapsarray_flip_bool
\tl_new:N \l_sdaps_sdapsarray_layouter_tl
\tl_new:N \l_sdaps_sdapsarray_align_tl
```

```
\bool_new:N \l_sdaps_sdapsarray_keepenv_bool
\int_new:N \l_sdaps_sdapsarray_angle_int
\tl_new:N \l_sdaps_sdapsarray_angle_sine_tl
\dim_new:N \l_sdaps_sdapsarray_colsep_dim
\dim_new:N \l_sdaps_sdapsarray_rowsep_dim
\bool_new:N \l_sdaps_sdapsarray_no_header_bool

\keys_define:nn { sdaps / sdapsarray }
{
  flip       .bool_set:N   = \l_sdaps_sdapsarray_flip_bool,
  flip       .initial:n  = false,
  flip       .default:n  = true,
  layouter   .tl_set:N   = \l_sdaps_sdapsarray_layouter_tl,
  layouter   .initial:n  = default,
  align      .tl_set:N   = \l_sdaps_sdapsarray_align_tl,
  align      .initial:n  = { },
  keepenv    .bool_set:N   = \l_sdaps_sdapsarray_keepenv_bool,
  keepenv    .initial:n  = false,
  keepenv    .default:n  = true,
  no_header  .bool_set:N = \l_sdaps_sdapsarray_no_header_bool,
  no_header  .initial:n  = false,
  no_header  .default:n  = true,

  angle          .code:n     = {
    \int_set:Nn \l_sdaps_sdapsarray_angle_int {#1}
    \tl_set:Nx \l_sdaps_sdapsarray_angle_sine_tl { \fp_to_decimal:n {sind(#1)}}
  },
  angle          .initial:n  = 70,
  colsep         .dim_set:N  = \l_sdaps_sdapsarray_colsep_dim,
  colsep         .initial:n  = 6pt,
  rowsep         .dim_set:N  = \l_sdaps_sdapsarray_rowsep_dim,
  rowsep         .initial:n  = 0pt,
}


```

2.5.2 Environment definition

```

\cs_new_nopar:Nn \l_sdaps_sdapsarray_alignment_set_have_content:
{
  \bool_set_true:N\l__sdaps_sdapsarray_have_content_bool
}

\cs_new_nopar:Nn \_sdaps_sdapsarray_alignment: {
    % End the last group, which will be the group that was begun earlier.
    % If the earlier cell was the first one, then this egroup also starts the
    % hbox to temporarily store the cells.
  \egroup
  \bool_if:NF \l__sdaps_sdapsarray_in_top_group_bool {
    \msg_error:nnn { sdapsarray } { unmatched_grouping_level } { an~alignment~tab }
```

```
  }

  % We need to notify the outside scope that there are items, will be inserted
  % multiple times, but that does not matter.
  \group_insert_after:N\l_sdaps_sdapsarray_alignment_set_have_content:

  % Just in case someone leaked a change into our scope
  \bool_if:NF \l_sdaps_sdapsarray_keepenv_bool {
    \cs_set_eq:NN \cr \_sdaps_sdapsarray_newline:
    \cs_set_eq:NN \\ \cr
  }

  % Define a cell now, we can just put everything into a new cell, and that
  % should work fine.
  % Note that cells are not safe for fragile commands at the moment.
  \_sdaps_sdapsarray_cell:w \bgroup
    \bool_set_false:N \l__sdaps_sdapsarray_in_top_group_bool
    \cs_set_eq:NN \\ \cr
    \tex_ignorespaces:D
}

\msg_new:nnn { sdapsarray } { unmatched_grouping_level } { The~grouping~level~of~a~cell~was~not
\msg_new:nnn { sdapsarray } { unequal_cols } { The~number~of~columns~is~not~equal~for~all~rows.
\msg_new:nnn { sdapsarray } { no_new_line_at_end } { You~have~terminated~the~last~line~with~\te
\msg_new:nnn { sdapsarray } { wrong_mode } { The~sdapsarray~environment~can~only~function~in~ve

\cs_new:Nn \_sdaps_sdapsarray_start_cells: {
    \bool_if:NF \l__sdaps_sdapsarray_in_top_group_bool {
      \msg_error:nnn { sdapsarray } { unmatched_grouping_level } { the~end~of~a~row~header }
    }
  \egroup

  % We are in the environment scope again here

  % Notify code that we are going to generate cells for a new row.
  \sdaps_array_row_start:

  % Start an hbox to stow away the cells.
  % The rest of the setup happens in the alignment handler
  \hbox_set:Nw \l_tmpb_box \bgroup
    \group_insert_after:N \hbox_set_end:
    \bool_set_true:N \l__sdaps_sdapsarray_in_top_group_bool
}

\cs_new_nopar:Nn \_sdaps_sdapsarray_linestart: {
  \sdaps_array_assign_rowhead:Nw \l_tmpa_box
    \bgroup
      \cs_set_eq:NN \\ \cr

      \bool_set_true:N \l__sdaps_sdapsarray_in_top_group_bool
```

```
      \bgroup
        \group_insert_after:N \_sdaps_sdapsarray_start_cells:
        \bool_set_false:N \l__sdaps_sdapsarray_in_top_group_bool
        % Ignore following spaces by the user
        \tex_ignorespaces:D
}

\cs_new_nopar:Nn \_sdaps_sdapsarray_newline: {
    \egroup
    \bool_if:NF \l__sdaps_sdapsarray_in_top_group_bool {
      \msg_error:nnn { sdapsarray } { unmatched_grouping_level } { the~end~of~a~row }
    }
  \egroup

  % We are in the environment scope again here
  % Output the last line if the cells were non-empty.
  \bool_if:NT \l__sdaps_sdapsarray_have_content_bool {
    \sdaps_array_row:NN \l_tmpa_box \l_tmpb_box
  }
  \bool_set_false:N \l__sdaps_sdapsarray_have_content_bool

  \cs_set_eq:NN \_sdaps_sdapsarray_cell:w \sdaps_array_cell:w
  \_sdaps_sdapsarray_linestart:
}

\cs_new:Npn\sdaps_array_nested_alignenv: {
  \char_set_catcode_alignment:N &
  \cs_set_eq:NN \cr \sdaps_orig_cr
  \cs_set_eq:NN \\ \sdaps_orig_backslash
}

\cs_new:Npn\sdaps_array_nested_alignenv:w {
  \bgroup
    \sdaps_array_nested_alignenv:
    % swallow group opening token
    \tex_let:D\next=
}
\cs_new_eq:NN \sdapsnested \sdaps_array_nested_alignenv:w

\group_begin:
\char_set_catcode_active:N &
\cs_new:Nn \_sdaps_sdapsarray_defines: {
  \cs_set_eq:NN \sdaps_orig_cr \cr
  \cs_set_eq:NN \sdaps_orig_backslash \\
  \bool_if:NF \l_sdaps_sdapsarray_keepenv_bool {
    \char_set_catcode_active:N &
    \cs_set_eq:NN \cr \sdaps_array_newline:
    \cs_set_eq:NN \\ \cr
    \cs_set_eq:NN & \sdaps_array_alignment:
  }
```

```
}
\group_end:

%%%%%%
% Flipping environment
%%%%%%

% First some helpers

\box_new:N \l_sdaps_sdapsarray_headers_box
\box_new:N \l_sdaps_sdapsarray_boxlist_head_box
\box_new:N \l_sdaps_sdapsarray_boxlist_tail_box

\cs_new_protected:Nn \_sdaps_sdapsarray_prepend_box:NN {
  \hbox_set:Nn #2 {
    \box_use:N #1
    \hbox_unpack:N #2
  }
  \box_clear:N #1
}

\cs_new_protected:Nn \_sdaps_sdapsarray_append_box:NN {
  \hbox_set:Nn #2 {
    \hbox_unpack:N #2
    \box_use:N #1
  }
  \box_clear:N #1
}

\cs_new_protected:Nn \_sdaps_sdapsarray_pop_last_box:NN {
  \hbox_set:Nn #2 {
    \hbox_unpack:N #2
    \box_gset_to_last:N \g_tmpa_box
  }
  \box_set_eq:NN #1 \g_tmpa_box
  \box_gclear:N \g_tmpa_box
}

\cs_new_protected:Nn \_sdaps_sdapsarray_pop_last_hbox_unpack:NN {
  \_sdaps_sdapsarray_pop_last_box:NN #1 #2
  \hbox_set:Nn #1 {
    \hbox_unpack:N #1
    \box_gset_to_last:N \g_tmpa_box
  }
  \box_set_eq:NN #1 \g_tmpa_box
  \box_gclear:N \g_tmpa_box
}

\cs_new_protected:Nn \_sdaps_sdapsarray_boxlist_void_if_empty:N {
  \hbox_set:Nn #1 {
```

```
    \hbox_unpack:N #1
    \box_set_to_last:N #1
    \box_if_empty:NTF #1 {
      \bool_gset_true:N \g_tmpa_bool
    } {
      \box_use:N #1
      \bool_gset_false:N \g_tmpa_bool
    }
  }
  \bool_if:NT \g_tmpa_bool {
    \box_clear:N #1
  }
}

% Lets say we are in row 4, cell 2 as defined in the environment, so the actual
% position is 2, 4. Minus the row headers, this gives us 2, 3.
% This means we need to append the cell to the box 3rd last box.
% In that case we have to append the cell

\cs_new_protected_nopar:Nn \_sdaps_sdapsarray_alignment_flip:
{
  \_sdaps_sdapsarray_end_cell_flip:

  % Just in case someone leaked a change into our scope
  \bool_if:NF \l_sdaps_sdapsarray_keepenv_bool {
    \cs_set_eq:NN \cr \_sdaps_sdapsarray_newline_flip:
    \cs_set_eq:NN \\ \cr
  }

  % Next up is either a cell or a row header. We can figure that out by checking
  % that the row headings box is void
  \box_if_empty:NTF \l_sdaps_sdapsarray_headers_box {
    \sdaps_array_assign_rowhead:Nw \l_tmpa_box \bgroup
      \bool_set_false:N \l__sdaps_sdapsarray_in_top_group_bool
      \cs_set_eq:NN \\ \cr
      % Ignore following spaces by the user
      \tex_ignorespaces:D
  } {
    \sdaps_array_assign_cell:Nw \l_tmpa_box \bgroup
      \bool_set_false:N \l__sdaps_sdapsarray_in_top_group_bool
      \cs_set_eq:NN \\ \cr
      \tex_ignorespaces:D
  }
}

\cs_new_nopar:Nn \_sdaps_sdapsarray_end_cell_flip: {
  \egroup % Finish of the cell
  \bool_if:NF \l__sdaps_sdapsarray_in_top_group_bool {
    \msg_error:nnn { sdapsarray } { unmatched_grouping_level } { end~of~cell~or~row }
  }
```

```
887
888   % Get last box from head
889   \_sdaps_sdapsarray_pop_last_box:NN \l_tmpb_box \l_sdaps_sdapsarray_boxlist_head_box
890   % Append the new box to the list of boxes for this row
891   \_sdaps_sdapsarray_append_box:NN \l_tmpa_box \l_tmpb_box
892   % Prepend the new box to the tail
893   \_sdaps_sdapsarray_prepend_box:NN \l_tmpb_box \l_sdaps_sdapsarray_boxlist_tail_box
894 }
895
896 \cs_new_nopar:Nn \_sdaps_sdapsarray_end_line_flip: {
897   % At the end of the line, move tail into head.
898   % First check that head is empty.
899   \_sdaps_sdapsarray_boxlist_void_if_empty:N \l_sdaps_sdapsarray_boxlist_head_box
900   \box_if_empty:NF \l_sdaps_sdapsarray_boxlist_head_box {
901     \msg_error:nn { sdapsarray } { unequal_cols }
902   }
903   \box_set_eq:NN \l_sdaps_sdapsarray_boxlist_head_box \l_sdaps_sdapsarray_boxlist_tail_box
904   \box_clear:N \l_sdaps_sdapsarray_boxlist_tail_box
905
906   % If this was the first row, store it away, these are the headings.
907   \box_if_empty:NT \l_sdaps_sdapsarray_headers_box {
908     \box_set_eq:NN \l_sdaps_sdapsarray_headers_box \l_sdaps_sdapsarray_boxlist_head_box
909     \box_clear:N \l_sdaps_sdapsarray_boxlist_head_box
910   }
911 }
912
913 \cs_new_nopar:Nn \_sdaps_sdapsarray_newline_flip: {
914   \_sdaps_sdapsarray_end_cell_flip:
915   \_sdaps_sdapsarray_end_line_flip:
916
917   % Create next box to store away, this has to be a colhead at this point
918   \sdaps_array_assign_colhead:Nw \l_tmpa_box \bgroup
919     \bool_set_false:N \l__sdaps_sdapsarray_in_top_group_bool
920     \cs_set_eq:NN \\ \cr
921     \tex_ignorespaces:D
922 }
923
924
925 \NewDocumentEnvironment { sdapsarray } { o }
926 {
927   \bool_set_false:N \l__sdaps_sdapsarray_in_top_group_bool
928   \group_begin:
929
930     \tl_set:Nn \l_tmpa_tl { }
931     \IfNoValueF { #1 } {
932       \tl_set:Nn \l_tmpa_tl { #1 }
933     }
934
935     \box_clear:N \l_tmpa_box
936     \box_clear:N \l_tmpb_box
```

```
937
938     % Ensure vertical mode.
939     \tex_par:D
940     \if_mode_vertical:
941     \else:
942       \msg_error:nn { sdapsarray } { wrong_mode }
943     \fi
944
945     % This needs to be initialized here as otherwise the values would be
946     % expanded at import time.
947     \keys_set:nV { sdaps / sdapsarray } \l_tmpa_tl
948
949     \sdaps_array_begin:VV \l_sdaps_sdapsarray_layouter_tl \l_sdaps_sdapsarray_align_tl
950
951     % Note, this environment is fragile; we redefine & to be active.
952     % One can go back into normal mode by using \sdapsnested{} though.
953
954     \bool_if:NTF \l_sdaps_sdapsarray_flip_bool {
955       \cs_set_eq:NN \sdaps_array_newline: \_sdaps_sdapsarray_newline_flip:
956       \cs_set_eq:NN \sdaps_array_alignment: \_sdaps_sdapsarray_alignment_flip:
957
958       \_sdaps_sdapsarray_defines:
959
960       % Two hboxes to hold the content, note that
961       % a: row headers, b: cells/col headers
962       \box_clear:N \l_sdaps_sdapsarray_headers_box
963       \box_clear:N \l_sdaps_sdapsarray_boxlist_head_box
964       \box_clear:N \l_sdaps_sdapsarray_boxlist_tail_box
965
966       % not sure why we need this group, but nothing works without it
967       \bgroup
968         % This is a bit creative to say the least
969         \sdaps_array_row_start:
970
971         \bool_set_true:N \l__sdaps_sdapsarray_in_top_group_bool
972         \sdaps_array_assign_rowhead:Nw \l_tmpa_box \bgroup
973           \bool_set_false:N \l__sdaps_sdapsarray_in_top_group_bool
974           \cs_set_eq:NN \\ \cr
975           % Ignore following spaces by the user
976           \tex_ignorespaces:D
977     } {
978       \cs_set_eq:NN \sdaps_array_newline: \_sdaps_sdapsarray_newline:
979       \cs_set_eq:NN \sdaps_array_alignment: \_sdaps_sdapsarray_alignment:
980       \_sdaps_sdapsarray_defines:
981
982       \cs_set_eq:NN \_sdaps_sdapsarray_cell:w \sdaps_array_colhead:w
983       \_sdaps_sdapsarray_linestart:
984     }
985
986   % If we redefine &, then the next character might have the wrong catcode
```

```
  % (i.e. it could still be an alignment character). Execute the alignment
  % code directly if the next character is &.
  \bool_if:NF \l_sdaps_sdapsarray_keepenv_bool {
    \peek_charcode_remove_ignore_spaces:NT & { \sdaps_array_alignment: }
  }
}
{
    \bool_if:NTF \l_sdaps_sdapsarray_flip_bool {
        \_sdaps_sdapsarray_end_cell_flip:

        % At this point we should have swallowed all items from the head list.
        % If not, then someone likely add a stray \\ command or similar
        \_sdaps_sdapsarray_boxlist_void_if_empty:N \l_sdaps_sdapsarray_boxlist_head_box
        \box_if_empty:NTF \l_sdaps_sdapsarray_boxlist_head_box {
          \_sdaps_sdapsarray_end_line_flip:
        } {
          \msg_error:nn { sdapsarray } { no_new_line_at_end }
        }

        % Now we can have fun!
        % Pop cells and heading, until we cannot find any new ones.
        \_sdaps_sdapsarray_pop_last_hbox_unpack:NN \l_tmpa_box \l_sdaps_sdapsarray_headers_box
        \_sdaps_sdapsarray_pop_last_box:NN \l_tmpb_box \l_sdaps_sdapsarray_boxlist_head_box
        \bool_do_while:nn { ! \box_if_empty_p:N \l_tmpa_box || ! \box_if_empty_p:N \l_tmpb_box
          \sdaps_array_row:NN \l_tmpa_box \l_tmpb_box

          \sdaps_array_row_start:

          \_sdaps_sdapsarray_pop_last_hbox_unpack:NN \l_tmpa_box \l_sdaps_sdapsarray_headers_bo
          \_sdaps_sdapsarray_pop_last_box:NN \l_tmpb_box \l_sdaps_sdapsarray_boxlist_head_box
        }

      \egroup
    } {
        \egroup

        \bool_if:NF \l__sdaps_sdapsarray_in_top_group_bool {
          \msg_error:nnn { sdapsarray } { unmatched_grouping_level } { the~end~of~the~environme
        }
      \egroup

      % We are in the environment scope again here
      % Output the last line if the cells were non-empty.
      \bool_if:NT \l__sdaps_sdapsarray_have_content_bool {
        \sdaps_array_row:NN \l_tmpa_box \l_tmpb_box
      }
    }

    \sdaps_array_end:

```

```
1037   \group_end:
1038 }
1039
1040
1041
1042
1043 \ExplSyntaxOff
1044
1045 %
```

Change History