TableLayout KnownProblems

At the end of TableLayout KnuthElementsForTables there is a notice about inaccurate steps when the table is broken over more than 2 pages. In this document another flaw is described which does not involve penalties at all.

Sample FO File

We will take the following FO file as an example:

```
<fo:root xmlns:fo="http://www.w3.org/1999/XSL/Format">
  <fo:layout-master-set>
    <fo:simple-page-master master-name="page" page-height="4cm" page-width="15cm" margin="12pt">
      <fo:region-body/>
    </fo:simple-page-master>
  </fo:layout-master-set>
  <fo:page-sequence master-reference="page">
    <fo:flow flow-name="xsl-region-body">
      <fo:block
        space-after.minimum="2pt"
        space-after.optimum="4pt"
        space-after.maximum="6pt">Before the table</fo:block>
      <fo:table table-layout="fixed" width="100%" border-collapse="separate"
        border-left="2pt solid black" border-right="2pt solid black">
        <fo:table-column number-columns-repeated="2" column-width="proportional-column-width(1)"/>
        <fo:table-body>
          <fo:table-row>
            <fo:table-cell>
              <fo:block keep-together="always">
                <fo:block>Cell 1 Line 1</fo:block>
                <fo:block>Cell 1 Line 2</fo:block>
              </fo:block>
              <fo:block keep-together="always" color="blue">
                <fo:block>Cell 1 Line 3</fo:block>
                <fo:block>Cell 1 Line 4</fo:block>
                <fo:block>Cell 1 Line 5</fo:block>
                <fo:block>Cell 1 Line 6</fo:block>
              </fo:block>
              <fo:block keep-together="always">
                <fo:block>Cell 1 Line 7</fo:block>
                <fo:block>Cell 1 Line 8</fo:block>
              </fo:block>
              <fo:block keep-together="always" color="blue">
                <fo:block>Cell 1 Line 9</fo:block>
                <fo:block>Cell 1 Line 10</fo:block>
                <fo:block>Cell 1 Line 11</fo:block>
              </fo:block>
              <fo:block keep-together="always">
                <fo:block>Cell 1 Line 12</fo:block>
              </fo:block>
            </fo:table-cell>
            <fo:table-cell>
              <fo:block keep-together="always" color="green">
                <fo:block>Cell 2 Line 1</fo:block>
                <fo:block>Cell 2 Line 2</fo:block>
              </fo:block>
              <fo:block keep-together="always" color="purple">
                <fo:block>Cell 2 Line 3</fo:block>
                <fo:block>Cell 2 Line 4</fo:block>
                <fo:block>Cell 2 Line 5</fo:block>
              </fo:block>
              <fo:block keep-together="always" color="green">
                <fo:block>Cell 2 Line 6</fo:block>
                <fo:block>Cell 2 Line 7</fo:block>
                <fo:block>Cell 2 Line 8</fo:block>
              </fo:block>
              <fo:block keep-together="always" color="purple">
                <fo:block>Cell 2 Line 9</fo:block>
                <fo:block>Cell 2 Line 10</fo:block>
```

```
              <fo:block>Cell 2 Line 11</fo:block>
            </fo:block>
            <fo:block keep-together="always" color="green">
              <fo:block>Cell 2 Line 12</fo:block>
              <fo:block>Cell 2 Line 13</fo:block>
              <fo:block>Cell 2 Line 14</fo:block>
              <fo:block>Cell 2 Line 15</fo:block>
            </fo:block>
          </fo:table-cell>
        </fo:table-row>
      </fo:table-body>
    </fo:table>
    <fo:block>After the table</fo:block>
  </fo:flow>
 </fo:page-sequence>
</fo:root>
```

The resulting pdf is interesting to see as well. Colours are used to illustrate unbreakable blocks of text. On the second page the line 9 of the first cell is laying inside the margin, and lines 10 and 11 are outside the page.

The list of Knuth elements for each column is as follows (the unit is a line):

```
    box(2)                  box(2)
    penalty(0)              penalty(0)
    box(4)                  box(3)
    penalty(0)              penalty(0)
    box(2)                  box(3)
    penalty(0)              penalty(0)
    box(3)                  box(3)
    penalty(0)              penalty(0)
    box(1)                  box(4)
    penalty(0)              penalty(0)
```

The merging algorithm gives the following sequence:

```
    box(2)
    penalty(0)
    box(3)
    penalty(0)  <--
    box(0)
    penalty(1)
    box(3)
    penalty(0)
    box(3)
    penalty(0)  <--
    box(0)
    penalty(1)
    box(4)
    penalty(0)
```

The two chosen breakpoints are marked with arrows. What's going on? Let's use some pictures to ease the explanations.

Illustrating the Process

We will use the following graphical notation to represent chunks of text:
http://people.apache.org/~vhennebert/wiki/TableLayout/KnownProblems/notations.png

According to that notation we have the following graphics for the two columns of the table and the Knuth sequence that results from the merge:
http://people.apache.org/~vhennebert/wiki/TableLayout/KnownProblems/mergedKnuthSequence.png

The merging algorithm works well when the table is broken over only 2 pages. In such a case, each time a break is considered, the reasoning is done in terms of what can be put on the current page and what will be left over to the next page. And this reasoning is the same for *every* break in the table. So in our case, the first chosen break corresponds to the following expected situation:
http://people.apache.org/~vhennebert/wiki/TableLayout/KnownProblems/firstBreak.png
whereas the second chosen break corresponds to the following:
http://people.apache.org/~vhennebert/wiki/TableLayout/KnownProblems/secondBreak.png

But the actual situation does not correspond to the expected one, since the second break is chosen to break the table *a second time*, and not for the first time like is 'coded' in the merged Knuth sequence. So at the chosen breakpoint, the algorithm believes that all the blocks but the last can fit on the page. That gives the following result:
http://people.apache.org/~vhennebert/wiki/TableLayout/KnownProblems/result.png

What's worrying here is that the algorithm really believes that everything is normal, and won't even complain about some content overflowing the page. That can lead to lost text without the user being warned. This is all the more frustrating than a much more desirable result is perfectly achievable:
http://people.apache.org/~vhennebert/wiki/TableLayout/KnownProblems/desirableResult.png

Towards a Solution

The problem in the previous situation is that the two Knuth sequences get desynchronized: the blank space due to shorter content in the first column is placed in the middle of the sequence instead of at the end of the table.

Like said earlier, the Knuth sequence resulting from the merge is valid only for the first break; after that, a new sequence needs to be generated to accurately represent the bits of columns that get placed on the second page. Depending on where the table is broken, the situation can vary a lot. On the following drawing some cases have been represented. Obviously computing all of the possible layouts leads to a combinatorial explosion (approximately O(
!), where *n* is the number of legal breakpoints in the merged Knuth sequence):
http://people.apache.org/~vhennebert/wiki/TableLayout/KnownProblems/combinatorialExplosion.png

But do we really need to compute *all* of the possibilities? No, of course! We could compute just the useful ones.

We would start with the sequence computed for the first break, like is currently done. Among all those breaks, the total-fit algorithm will select just a few, those which are likely to result into acceptable results; that is, the breaks which make *feasible* breaks and not too short or too long nodes. This will depend on the set of nodes that are 'active' when the legal breaks of the table are considered.

Each of the feasible table breaks would be flagged as special, that is, only a special sequence of Knuth elements would be allowed to follow them. That sequence would result from the merge of the parts of the table left over to the following page. It would let compute the feasible second breaks in the table, if any. And so on...

Once the end of the table is reached, we would return to a usual situation with normal Knuth elements for content following the table.

What amount of overhead would that add? In most situations, none, since tables will entirely fit on one page, or broken just once. For the rest, that would require to run the merging algorithm for each recorded feasible break inside the table. That should remain reasonable with carefully optimized code.

This situation seems to be similar to the changing IPD issue, and combining line- and page-level breaking. So this will probably make sense to study both issues at the same time.