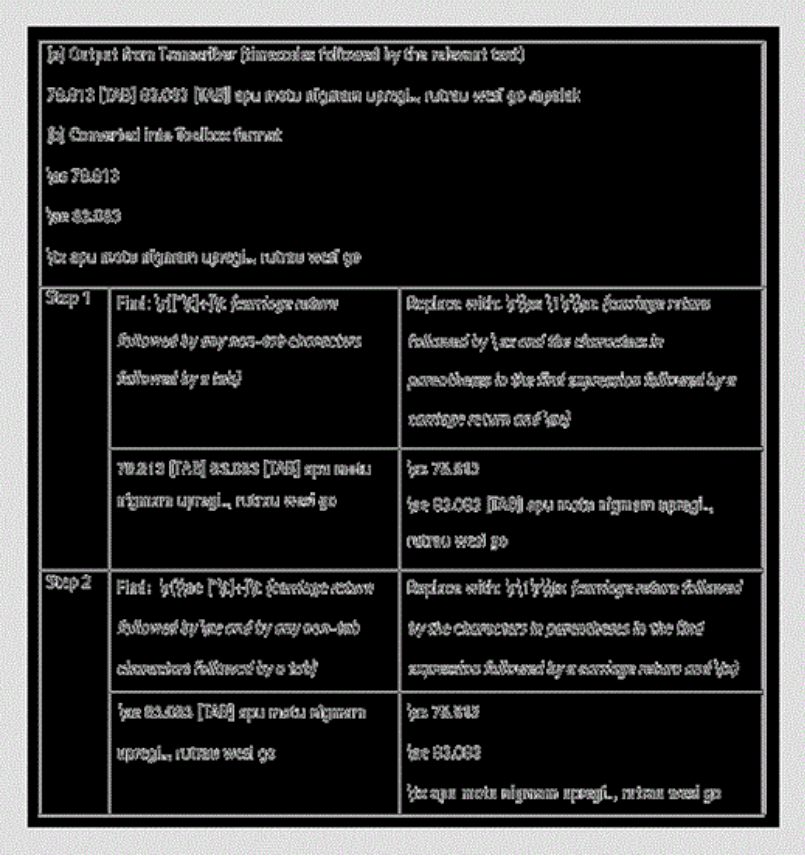
4.3.4 Regular expressions

A consequence of a workflow using different software tools is that data needs to be converted from the output format of one tool to the input format of another tool without losing any of the data in the process. Textual conversions can be achieved by using *regular expressions* (also known as 'regex'), which are sophisticated search-and-replace routines. Regular expressions can save you a great deal of time, and it is well worth at least learning what they can do, even if you don't want to learn how to use them yourself. Basic information is easily found on the web,[27] Friedl (2006) is a useful reference, and Gries (2009: 68–99, 105–72) provides good instruction on using regular expressions in linguistic contexts. You can always find someone to help you with regular expressions if you do not feel confident creating them yourself.

Three examples of the power of regular expressions are given below. Fig. 4.7 shows how regular expressions can be used to convert the tab-delimited output from Transcriber software in (a) into the structured text that Toolbox software requires, shown in (b). While the change could be done with a single regular expression, we have split it into two steps for illustrative purposes.

Step 1 finds the first tab stop, then inserts '\as' (the MDF field marker indicating the time code for start of the audio clip) at the beginning of the line and replaces the first tab stop with a carriage return and '\ae' (for
p. 110 the time code of the end of the ↳ audio clip). The second step finds the next tab stop and replaces it with a carriage return and the MDF text field marker '\tx'.

Figure 4.7.



Example of a text (a) and its derived form (b), arrived at by use of regular expression search-and-replace routines.

A second example of the use of regular expressions is the conversion of a dictionary made in a word processor into computationally tractable files, ready to be used in specialized lexicographic software. Fig.