



Figure 8. A transcription example

The trees of Section 3.3 are abstract descriptions of music scores. The elementary score elements (symbols) correspond to the names of token types (notes, rests, chords *etc* in K) labelling leaves. The MIDI pitch of every note can be extracted from the input MIDI sequence using the definition of tokens in Section 3.4. A pitch-spelling algorithm [20] is then necessary to cast these MIDI key values to note names. Moreover, the durations are encoded in t by the symbols of F labeling inner nodes. Additionally to the operations on time intervals, some info about the output score can be attached to the symbols of F . For instance, we can express in a symbol div_n whether we want the notes below this symbol to be beamed or not.

4.4 Algorithm and Implementation

We designed and implemented an algorithm for the parsing problem defined in Section 4 based on ordinary tabulation technique.

Given in input a MIDI sequence E and a weighted tree grammar, it returns a tree t minimizing (1) in Section 4.3. Figure 9 presents the parsing algorithm designed based on ordinary Dynamic Programming. It assumes that the input

midi-sequence E is played with a constant tempo and converted to musical time so that one bar is identical to one second.

The algorithm deals with a set of tabulated items, which is kept by the variable C , as candidates of a parsing result. each item contains

1. a current parse tree t (or a sequence of parse trees),
2. the sum $w = ca_E(t, I) + cr_G(t)$ of the parse-tree weight and its alignment cost (or the weight sum of the parse trees and their alignment costs),
3. the set $E_{\text{snd}(I)}$ of unprocessed events in the interval I of the parse tree(s), and
4. the assigned time prev of the last token in the parse tree(s).

See appendix B for the detail.

The source code of this implementation is found in URL ¹, and produces the command line utilities *monopase*

¹ <https://gitlab.inria.fr/qparse/qparselib>

