ENP-Regex - a Regular Expression Matcher Prototype for the Expressive Notation Package

Mika Kuuskankare*

Sibelius Academy, Finland
mkuuskan@siba.fi

Abstract. In this paper we introduce ENP-regex, a prototype of a regular expression matcher developed for Expressive Notation Package (ENP). ENP-regex allows us to use the regular expression syntax to match against several score attributes, such as pitch and rhythm. Instead of writing the regular expression matcher from scratch we implement a scheme where a thin conversion layer is inserted between an existing Lisp-based regular expression library and ENP. The information sent from ENP to the regex matcher is transformed into a textual format. Similarly, the matches are converted into corresponding score objects. The benefit of the present implementation is that potentially the whole syntax of the regex matcher in question is at our disposal. We have implemented a prototype of the regular expression matcher. In this paper we present the current state of the system through examples.

Keywords: Regular expressions, music notation, scripting, music analysis and visualization

1 Introduction

In this paper we present an extension to Expressive Notation Package (ENP, [8]) called ENP-regex. ENP-regex allows us to use regular expressions to match against musical data, such as pitch and rhythm. Traditionally, regular expressions are used for matching characters, words, or patterns of characters in strings. Similarly, with ENP-regex, we are able to match notes, groups of notes and different patterns in an ENP score according to a given property.

Regular expressions and other string search algorithms have been widely used in the music domain. Dovey [4] reports a regular expression like search framework that uses piano-roll notations as a starting point. One of the most notable music analysis applications, Humdrum [6], uses the regular expressions extensively. The problems with representing musical attributes with text are widely discussed in [3].

Our main motivation is to study the potential of regular expressions in the context of ENP. We use symbolic music notation, not text, as a starting point

* The work of Mika Kuuskankare has been supported by the Academy of Finland (SA137619). We would also like to thank CCRMA, Stanford University, for hosting the research.

9th International Symposium on Computer Music Modelling and Retrieval (CMMR 2012)
19-22 June 2012, Queen Mary University of London

and use musical conventions, rather than textual, when describing the regular expression patterns. This should make the system more approachable for musicians. The mapping between the musical attributes and regular expressions is done on the fly without any further actions required from the user.

A new scripting language is envisioned where any Lisp function could be applied to the matching objects. Potentially, we could insert expressions, add or delete notes, transpose them, etc. For example, an intelligent find (or find and replace) extension could be implemented with the help of regular expressions.

One of the benefits of using regular expressions is that they are widely known and used. The plan is to eventually integrate ENP-regex more closely into the ENP tool-chain.

The rest of the paper is organized as follows. First, we discuss some of the implementation issues. Next, we give some examples of real-world problems where ENP-regex would prove to be useful. The paper ends with some discussion and a list of plans for further development.

2 ENP-regex

ENP-regex is based on a library called cl-ppcre [1] which is a regular expression library for Common Lisp. The ENP-regex matcher can be run in different domains, currently pitch, rhythm, interval, and harmony (pitch-class set), to match against several score properties. The user inputs the regular expression using a slightly modified syntax (this will be discussed below in more detail). The target score is encoded so that it can be processed by the cl-ppcre matcher. The results returned by cl-ppcre (indices) are translated back to score objects, and, finally, the action indicated by the user is performed. At this stage we mark the matches in several different ways, such as inserting expressions, or simply by highlighting the matches.

One of the key concepts behind the ENP-regex implementation is the idea of a translator. A translator maps the desired score objects into a representation that, in turn, can be used as an input to a conventional regex parser, which, in turn, returns indices which are mapped back to score objects. Figure 1 illustrates this process.

The regular expression syntax used in the case of ENP-regex is compatible with that of Perl but slightly extended. Although it would be convenient in our case, normally, we do not write a pattern as $\boxed{\texttt{[60-66]}}$ to match all numbers between 60 and 66. Therefore, for convenience, a small language extension is provided which allows us to use a more musically oriented syntax when defining the regular expression patterns.

For pitch, both absolute pitch and intervals, we use the MIDI note representation, i.e., middle-C is represented by the number 60, and for rhythm the fractional notation, e.g., $\boxed{1/4}$ or $\boxed{1/20}$. Note that our MIDI note representation is extended as it allows us to represent micro-intervals by adding a fractional part, such as 0.5, for example, to denote a quarter tone. For harmony, we use

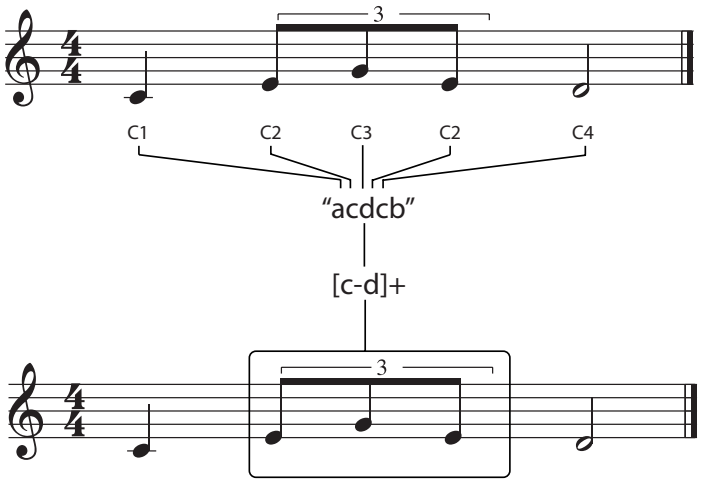

Fig. 1. The translation of score properties into a representation that can be parsed by a regular expression engine and back to score objects.

the pitch-class set notation following the conventions introduced by Allen Forte [2], where the major triad, for example, is notated with the symbol "3-11b" and the minor one with "3-11a".

To distinguish the ENP-regex notation from that of regular expressions we use a hash-mark (#) as a prefix. A pre-processor is implemented which translates our customized regex syntax into the Perl compatible syntax. Thus, it is possible to indicate, for example, a pitch range by writing it as `[#60-#72]`.

3 Examples

In this section we illustrate the potential of ENP-regex through examples.

Figure 2 shows our editor developed for testing the ENP-regex interface. The first row gives the selectors for the property domains, i.e, pitch, rhythm, intervals, and harmony. Using the next group of controllers we can select the desired side-effect. "default" indicates that we want to highlight the matches and "custom" together with the following text input field allow us to specify the class name and the attributes of an ENP-expression [7], which will be applied to the matches found in the score. The third row allows us to choose the matching direction (this will be discussed in Section **??**). Finally, in the bottom row the ENP-regex pattern is given.

3.1 Phrases

We begin with a simple example that aims to illustrate the relationship between regular expressions and ENP. The internal encoding of pitch (and other information) is arranged so that the alphanumeric characters are reserved for attributes

that are related to events, and the 'non-word characters' are reserved for rests. Currently, we do not distinguish between rests of different lengths, but treat them as non-sounding events. Therefore, the 'alphanumeric characters' symbol, `\w`, in ENP-regex is used to indicate a note, and the `\W`, in turn, indicates a rest. We provide this translation for convenience only and it does not attempt to draw any further conclusions about the relationship of music and text.

In our first example (see Figure 2) we use ENP-regex to insert phrasing slurs in the score, using the following regular expression: `\w+`. This is a straightforward way of segmenting music according to the rests. We also use a custom phrasing slur with some additional attributes (see, "SLUR :KIND :DASHED" in Figure 2) instead of simply revealing the matches. The phrasing slur is displayed in the score as a curve using a stippled pattern.

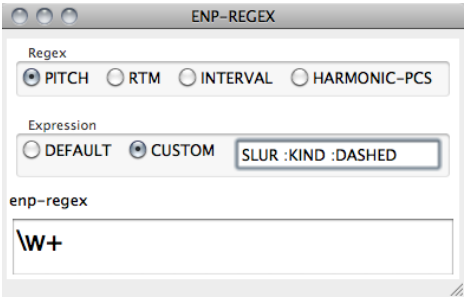

Fig. 2. The ENP-regex tool with the regex expression at the bottom.

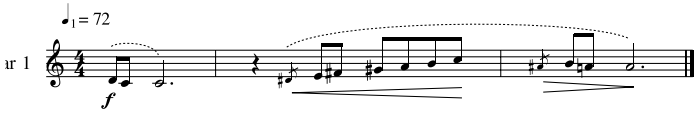

Fig. 3. Inserting phrase marking (the two dashed slurs above the score) with the help of ENP-regex using the pattern \w+. (Yesterday by The Beatles)

3.2 Pitch

In Figure 4, we give an example of ENP-regex in the pitch domain, where we aim to reveal the extreme pitches in a passage written for the flute. The flute spans form B3 to C7 and above. Here, the range considered as extreme is chosen somewhat arbitrarily. The ENP-regex pattern to find and mark the ranges is as follows: [#59-#60#90-#96].

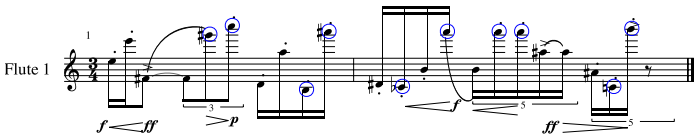

Fig. 4. Indicating extremely low and high pitches (the encircled notes) in the piece of music for the flute using a pattern with low and high ranges: [#59-#60#90-#96].

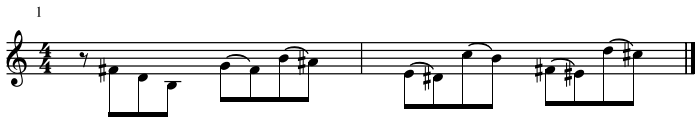

Fig. 5. Articulation slurs inserted according to the interval between two consecutive notes. (J.S.Bach)

3.3 Intervals

As an example of ENP-regex in the interval domain we attempt to add appropriate articulation slurs to a small excerpt of music by J.S. Bach (see Figure 5). We note that in the original there is a slur between two notes forming a descending minor second interval. We define the interval pattern as `#-1` and define the slur expression as in Figure 3 but without the extra attributes (:kind :dashed). Figure 5 shows the slurs inserted with the help of ENP-regex.

3.4 Harmony

The example shown in Figure 7 is a small excerpt, prepared by the Finnish composer Kimmo Kuitunen, called "6-Z47B"-blues. Here, we use ENP-regex in the harmony domain to locate certain sonorities, namely the "mother" set-class 6-Z47B and the set-class named 5-35 (a chord consisting of only perfect fifth intervals is possible to construct using the set-class 5-35). The ENP-regex is given in Figure 6. This example demonstrates the ENP-regex can also be executed in non-metrical context.

3.5 Repetitions Using Back-references

Our final example in this section deals with repetitions. Here, we use a regular expression construct, called a back-reference, which is defined as follows: `(.+)\1+`. The matching is done in the harmony domain. In Figure 8 the matching harmonies are indicated by enclosing them inside boxes. Note, that harmony here is a harmony class, thus it does not have anything to do with a particular setting or voicing. This simple pattern finds repeating series harmonies. The first of the matches is an alternating pattern between two different harmonies, and the latter two matches represent static repeating harmonies.

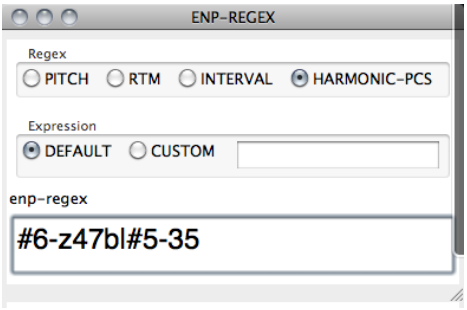

Fig. 6. ENP-regex in the harmony domain aimed at finding and marking specific harmonies in the target score.

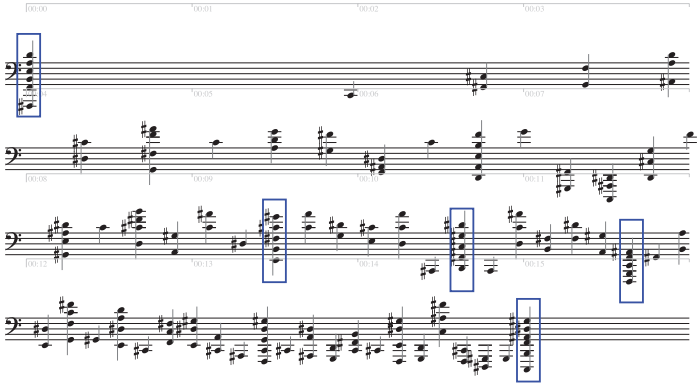

Fig. 7. The harmonies 6-Z47B and 5-35 marked in the score by Kuitunen.

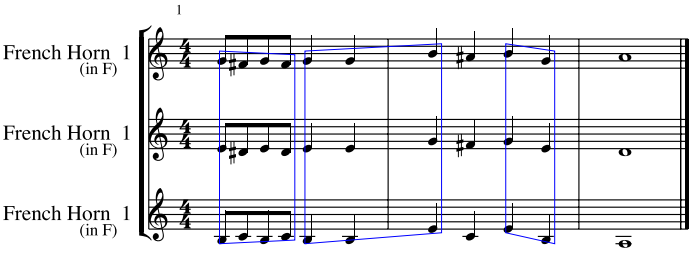

Fig. 8. Harmonic repetitions revealed with the help of ENP-regex using back-references. (Prokofiev: Peter and the Wolf)

4 Future Development

There are several improvements in the planning. First, we should develop a user API for creating custom mappings to any score property, e.g., ENP-expressions.

Second, we should support iterating over higher-level objects than notes. The user could be presented with a choice between notes, chords, and measures, for example. This way we would be free of adding any complexity in the regex pattern, in terms of dealing with the beat boundaries, for example.

Third, we should augment the regular expression specification of ENP-regex. Several additions are planned, such as specifying the matching direction, e.g., right-to-left or bi-directional matching, and incorporating loop-like constructs, such as the beginning index of the matching, step, etc. The latter would allow us to use regex matching to insert, for example, interval n-grams into the target score. n-grams have been have been widely used in text retrieval and are also proposed for MIR applications, for example, in [5].

Finally, the regex matchers could potentially also be combined. It should be investigated if there is a feasible way to logically combine the results. A simple intersection might not be enough, as, for example, a regex `[60-67]{4}` would not necessarily be true, when an intersection is taken with the results returned by the regex `1/4+` executed in the rhythm domain, etc. However, it would be interesting to provide users with the choice as it would allow us to make multi-parameter regular expression matching.

Finally, our regex implementation could also potentially be coupled with the existing pattern matching language of PWGLConstraints[9], thus allowing us to use both syntaxes interchangeably. Some patterns would be more easily expressed using the regex syntax, rather than that of our backtracking constraints system.

5 Conclusions

This paper presents ENP-regex, the prototype implementation of a regular expressions matcher for the Expressive Notation Package. Currently, ENP-regex is able to use most of the regular expression syntax and can match against different types of score information, such as pitch, rhythm, intervals, and harmony.

The most interesting applications of the present work can be found in the domains of music information retrieval, scripting, and computer assisted composition and analysis.

References

1. ppcre. http://weitz.de/cl-ppcre/
2. Allen Forte: The Structure of Atonal Music. Journal of Music Theory (1973)
3. Cambouropoulos, E., Crawford, T., Iliopoulos, C.S.: Pattern processing in melodic sequences: Challenges, caveats and prospects. In: In Proceedings of the AISB'99 Convention (Arti Intelligence and Simulation of Behaviour. pp. 42–47 (1999)

4. Dovey, M.J.: A technique for regular expression style searching in polyphonic music. In: International Symposium on Music Information Retrieval (2001)
5. Downie, J.S.: Evaluating a Simple Approach to Music Information Retrieval: Conceiving Melodic N-grams as Text. Ph.D. thesis, University of Western Ontario (1999)
6. Huron, D.: Music information processing using the humdrum toolkit: Concepts, examples, and lessons. Computer Music Journal 26(2), 15–30 (2002)
7. Kuuskankare, M., Laurson, M.: ENP-Expressions, Score-BPF as a Case Study. In: Proceedings of International Computer Music Conference. pp. 103–106. Singapore (2003)
8. Kuuskankare, M., Laurson, M.: Expressive Notation Package. Computer Music Journal 30(4), 67–79 (2006)
9. Laurson, M.: PATCHWORK: A Visual Programming Language and some Musical Applications. Studia musica no.6, doctoral dissertation, Sibelius Academy, Helsinki (1996)