Tom Collins

MCStylistic: a Lisp package for research into music theory, music cognition, and stylistic composition

Version: June 2014

ACKNOWLEDGEMENTS

My thanks to the following for various help with MCStylistic: Darrell Conklin, Dave Cope, Joe Cornelli, Peter Elsea, Jamie Forth, Chris Fox, Paul Garthwaite, Soren Goodman, Robin Laney, Kjell Lemström, Veli Mäkinen, Dave Meredith, Paul Pelton, Richard Sutcliffe, Jeremy Thurlow, Esko Ukkonen, Geraint Wiggins, and Alistair Willis.

The MCStylistic package includes CL-FAD (http://weitz.de/cl-fad/), a portable pathname library for Common Lisp, and the Humdrum extras function xml2krn (http://extras.humdrum.org/). Edited versions of data from KernScores (http://kern.ccarh.org/) are included also. I am grateful to the Center for Computer Assisted Research in the Humanities at Stanford University for curating this large, high-quality collection of symbolic music representations.

Tom Collins,
Leicester, UK,
Home of Gary Lineker and Richard III,
June 2014.

RELATED PUBLICATIONS

If making use of MCStylistic code in a publication, please include a citation such as 'we made use of MCStylistic-Jun2014 (Collins, 2011), available from http://www.tomcollinsresearch.net.'

For more specific use of code in Secs. 3.1, 3.2, or 4.8, please cite the following conference paper.

> Tom Collins. Stravinsqi/De Montfort University at the MediaEval 2014 C@merata task. In *Proceedings of the MediaEval 2014 Workshop*, 6 pages, Barcelona, 2014.

For Secs. 3.4, 3.5, or 4.6, please cite the following journal paper.

> Tom Collins, Robin Laney, Alistair Willis, and Paul H. Garthwaite. Developing and evaluating computational models of musical style. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, in press.

For Secs. 3.3 or 4.3, please cite the following journal paper.

> Tom Collins, Robin Laney, Alistair Willis, and Paul H. Garthwaite. Modelling pattern importance in Chopin's mazurkas. *Music Perception*, 28(4):387-414, 2011.

For code in Secs. 3.3 or 4.4, please cite the following conference paper.

> Tom Collins, Jeremy Thurlow, Robin Laney, Alistair Willis, and Paul H. Garthwaite. A comparative evaluation of algorithms for discovering translational patterns in Baroque keyboard works. In J. Stephen Downie and Remco Veltkamp, editors, *Proceedings of the International Symposium on Music Information Retrieval*, pages 3-8, Utrecht, 2010. International Society for Music Information Retrieval.

If making use of any other code, please cite my thesis.

> Tom Collins. *Improved methods for pattern discovery in music, with applications in automated stylistic composition.* PhD thesis, Faculty of Mathematics, Computing and Technology, The Open University, 2011.

CONTENTS

PREFACE

MCStylistic supports research in **M**usic theory, music **C**ognition, and **Stylistic** composition. It is free, cross-platform, and written in Common Lisp. In terms of algorithmic highlights, MCStylistic contains the following.

- The Stravinsqi algorithm takes as input a piece of digital staff notation and a natural language query, such as 'submediant triad followed five bars later by a perfect cadence'. It returns the time window(s) at which events corresponding to the query occur in the music (Collins, 2014). This work represents a step towards an easy-to-use, high-level music-theoretic search engine.

- The HarmAn algorithm performs automatic chord labelling (Pardo and Birmingham, 2002). I have extended its capability to automatic functional harmonic analysis.

- The Inner Metric Analysis algorithm analyses the time-varying prevalence of different metric levels (Volk, 2008).

- The Keyscape algorithm visualises local and global key regions (Sapp, 2005), which in turn builds on probe-tone profiles (Krumhansl, 1990; Aarden, 2003);

- The Structure Induction Algorithm (SIA, Meredith, Lemström, and Wiggins, 2002, 2003; Meredith, 2006) and its extensions (Collins et al., 2010, 2011; Collins, 2011) attempt to discover perceptually salient or musically important repetitions within a given piece of music. (Parallelised and more up-to-date versions of these algorithms are available via PattDisc, a Matlab package I maintain.)

- In the spirit of Experiments in Musical Intelligence (EMI, Cope, 1996, 2001, 2005), two of my algorithms for generating stylistic compositions (Racchman-Oct2010, Sec. 3.4, and Racchmaninof-Oct2010, see Collins et al., in press, or Sec. 3.5) are included also.

Chapter 1 gets you setup with Common Lisp, but if you are already running a version, please jump straight to loading MCStylistic (Sec. 1.2).

Chapter 2 gives examples for importing and exporting files of various formats. (It is difficult to do much of any substance until you can import a file, work on it in the Lisp environment, and then export it to another file as saved work.)

Chapter 3 is devoted to coding examples, and splits into six sections. Section 3.1 gives an overview of the capabilites of the Stravinsqi algorithm, which stands for STaff Representation Analysed VIa Natural language String Query Input. Section 3.2 provides examples of functional harmonic analysis with the HarmAn->roman algorithm. Section 3.3 contains code for discovering and rating musical patterns, along with an explanation. Sections 3.4 and 3.5 exemplify two models for generating stylistic compositions. The models are called Racchman-Oct2010 and Racchmaninof-Oct2010, standing for RAndom Constrained CHain of Markovian Nodes with INheritance Of Form. Section 3.6 is a guide to using the evaluation functions for the MIREX pattern discovery task:

```
http://www.music-ir.org/mirex/wiki/2013:
  Discovery\_of\_Repeated\_Themes\_\%26\_Sections
```

The longest chapter by far is Chapter 4. It contains documentation for each individual function that is included in the package MCStylistic-Jun2014.

New highlights in MCStylistic-Jun2014

1. Anacrusis handling in kern import.
2. Import of lyric, articulation, rest, and tie data from kern files.
3. The Stravinsqi algorithm, for taking a natural language query and searching digital staff notation for musical events that correspond to the query.
4. Texutre identification.
5. Extension of HarmAn-> to functional harmonic analysis.
6. Re-checked Chopin mazurka data for errors.

CHAPTER 1

SETUP

1.1 Installing Clozure Common Lisp

There are many versions of Common Lisp, one of which is Clozure Common Lisp (CCL). Information about CCL can be found at

http://ccl.clozure.com/

For download information, please see:

http://ccl.clozure.com/download.html.

This page recommends getting CCL via Subversion. I have found the archive files (link below the comments on Subversion) and **this CCL 1.7 disk image** (ftp://ftp.clozure.com/pub/release/1.7/ccl-1.7-darwinx86.dmg) for Mac to work just as well. More recent updates of CCL for Mac do not seem so stable.

If you require more information on installing CCL, please consult the online documentation:

http://ccl.clozure.com/manual/index.html.

1.2 Loading MCStylistic

The default path for MCStylistic-Aug2013 is

/Users/Shared/

If you wish to move MCStylistic-Aug2013, or any of the other data, example files, or Lisp functions to other locations, then please do so, carry on reading, and see the next footnote.

The folder MCStylistic-Aug2013 contains a file called *setup.lisp*. To load the MCStylistic package, open up CCL (or your preferred Lisp implementation) and open *setup.lisp*.[1] Now execute all of the text in *setup.lisp*: from the program menu select Lisp → Execute All, or use the keyboard shortcut Command-Shift-E. A Listener window should open up, and messages will begin to appear as paths, variables, and functions are defined. Finally, you should see a message in the Listener that reads 'Welcome to MCStylistic-Aug2013.' This means you are ready to begin.

Each time you wish to use the MCStylistic package, the ritual of executing all of the text in *setup.lisp* must be observed. Before moving on, a couple of tips for using Lisp:

- Peter Seibel's Practical Common Lisp, The Common Lisp Cookbook, and Stack Overflow address %95 of questions/problems I encounter with Common Lisp. Paul Graham's Lisp site is also great, and contains some articles about the history of Lisp.

- in CCL it is possible to highlight code between parentheses by double-clickling either the opening or closing parenthesis;

- a semicolon can be used to comment-out the remainder of a line of code;

- the symbol combination '#|' begins a commented paragraph, and '|#' ends a commented paragraph.

[1] To use the MCStylistic-Aug2013 package from a location other than '/Users/Shared/', go to line 13 of *setup.lisp* (four lines down from `*MCStylistic-Aug2013-path*`) and change the strings to specify your preferred location. Following the definition of `*MCStylistic-Aug2013-path*` are definitions for the locations of some sample music data, example files, and not least the Lisp functions, all of which can be altered if you wish. When finished making changes, please double-check your pathnames and save *setup.lisp*.

CHAPTER 2

IMPORTING AND EXPORTING FILES

The examples in this chapter assume that the Lisp package MCStylistic has been loaded (cf. Sec. 1.2).

2.1 Lists stored as text

The function read-from-file can be used to import lists that are stored as text files (.txt) into the Lisp environment. For instance, the folder called *Example files* that comes with MCStylistic contains a subfolder called *Example data*, which contains a text file called *short-list.txt*, and this can be imported into the Lisp environment using the following code.

```
1 (read-from-file
2  (merge-pathnames
3   (make-pathname
4    :name "short-list" :type "txt")
5   *MCStylistic-Aug2013-example-files-data-path*))
6 --> ((9 23 1 19) (14 9 14 5 20 25) (16 5 18 3 5 14 20)
7      ("sure" 9 4) (13 9 19 8 5 1 18 4) (8 5 18))
```

Not setting the path in line 4 correctly (cf. Sec. 1.2) may result in an error message.

To be able to make use of the above list, it is necessary to give it a name as it is imported. For instance the following code names an imported list, and then accesses its second element.

```
(setq
 *little-list*
 (read-from-file
  (merge-pathnames
```

```
  (make-pathname
   :name "short-list" :type "txt")
  *MCStylistic-Aug2013-example-files-data-path*)))
--> ((9 23 1 19) (14 9 14 5 20 25) (16 5 18 3 5 14 20)
     ("sure" 9 4) (13 9 19 8 5 1 18 4) (8 5 18))
(nth 1 *little-list*)
--> (14 9 14 5 20 25)
```

Sometimes we want to import and name a list (or other data types for that matter), but do not want to see the entire list output (which is what happened in both of the above examples), perhaps because the list is very long. One way of suppressing output is as follows.

```
1  (progn
2    (setq
3     *little-list*
4     (read-from-file
5      (merge-pathnames
6       (make-pathname
7        :name "short-list" :type "txt")
8       *MCStylistic-Aug2013-example-files-data-path*)))
9     "*little-list* imported")
10 --> "*little-list* imported"
```

The function write-to-file enables lists that are defined within the Lisp environment to be stored as text files. For instance, at present the *Example results* folder in *Example files* does not contain any file called *another-list.txt*, but we are going to create one. The following code assumes that the variable `*little-list*` has been defined as per lines 1-10 above. Elements of this list are combined with other variables and values to form a new list, which is stored as a text file using the function write-to-file.

```
(setq *A* (list 0 60 60 1 0))
--> (0 60 60 1 0)
(setq *B* "middle C")
--> "middle C"
(setq
 *new-list*
 (list
  (nth 3 *little-list*) *A* (nth 4 *little-list*) *B*
  4 2))
```

```
10 --> (("sure" 9 4) (0 60 60 1 0) (13 9 19 8 5 1 18 4)
11      "middle C" 4 2)
12 (write-to-file
13  *new-list*
14   (merge-pathnames
15     (make-pathname
16      :name "another-list" :type "txt")
17     *MCStylistic-Aug2013-example-files-results-path*))
18 --> T
```

The *Example results* folder should now contain a file called *another-list.txt*. If you open up this text file in a text editor, it should contain elements of the list shown in lines 10-11 above.[1] If you are using a different version of Lisp to CCL, it is worth checking two things. First, importing a file and then exporting it with a new name results in two text files with exactly the same format. This should be the case as `*print-length*` and `*print-pretty*` are both set to `nil` (cf. Sec. 1.2). Second, if the location specified in the call to write-to-file does not already exist, Lisp creates the location and writes the file.

2.2 Musical Instrument Digital Interface (MIDI)

The function load-midi-file can be used to import a Standard MIDI File into the Lisp environment. Musical instrument digital interface (MIDI) is a means by which an electronic instrument (such as a synthesiser, electronic piano, drum kit, even a customised guitar, etc.) can connect to a computer and hence communicate with music software (Roads, 1996). When a MIDI file is imported into the Lisp environment using the function load-midi-file, a list of five-element lists is defined, where the first element is the *ontime* of a MIDI event, the second element is the *MIDI note number* (MNN, with 'middle C' = C4 = 60, C♯4 = 61, etc.), the third element is the *duration* (arbitrary timescale), the fourth element is the *channel* (between 1 and 16), and the fifth element is the *velocity* (between 0 for silence and 127 for maximal loudness). These five-element lists are shown in lines 12-15 of the following code, which imports a MIDI representation of the second movement of the Concerto in G minor op.6 no.3 by Antonio Vivaldi (1678-1741).

[1] The function write-to-file and other export functions exemplified in this chapter *overwrite* existing files, as opposed to using *supersede* or *append* (for details, see Seibel, 2005).

```
1 (progn
2   (setq
3    *vivaldi-movement*
4    (load-midi-file
5     (merge-pathnames
6      (make-pathname
7       :name "vivaldi-op6-no3-2" :type "mid")
8      *MCStylistic-Aug2013-example-files-data-path*)))
9   "*vivaldi-movement* imported")
10 --> "*vivaldi-movement* imported"
11 (firstn 10 *vivaldi-movement*)
12 --> ((0 79 1 6 64) (0 69 1 4 64) (0 49 1 1 64)
13      (0 49 1 3 64) (0 76 1 5 64) (1 79 1 6 64)
14      (1 69 1 4 64) (1 76 1 5 64) (1 49 1 1 64)
15      (1 49 1 3 64))
```

The function saveit can be used to export an appropriate list defined in the Lisp environment to a Standard MIDI File. The term *appropriate* means that each element of the list is a five-element list (as in lines 12-15 above), with permissible values for ontime (positive float), MNN (integer between 0 and 127), duration (positive float), channel (integer between 1 and 16), and velocity (integer between 0 and 127). Trying to export an inappropriate list as a MIDI file may result in an error message. As an example, the folder called *Example results* that comes with MCStylistic does not contain any file called *two-arpeggios.mid*, but we can create one using the following code.

```
(progn
  (setq
   *arpeggios*
   '((0 49 8000 1 64) (1000 69 7000 1 69)
     (2000 76 6000 1 74) (3000 79 5000 7 79)
     (8000 50 8000 1 84) (9000 69 7000 1 89)
     (10000 74 6000 1 94) (11000 77 5000 7 99)))
  (saveit
   (merge-pathnames
    (make-pathname
     :name "two-arpeggios" :type "mid")
    *MCStylistic-Aug2013-example-files-results-path*)
   *arpeggios*))
--> Returns pathname of the file created.
```

MIDI files can be opened and played using programs such as QuickTime Player, RealPlayer, and Windows Media Player. They can also be embedded in web pages. Playing the file created above, you may notice that ontimes and durations are specified in milliseconds. If you have used MCStylistic to discover some repeated patterns (cf. Sec. 3.3) or to generate a dataset representation of music (cf. Sec. 3.4), then the function saveit is a convenient way to audition (hear) the results.

2.3 Kern

At http://kern.ccarh.org/ there are a large number of symbolic music encodings in so-called kern format. A short kern file can be found in the folder *Example data* that comes with MCStylistic, called *C-6-1-small.txt*, and this can be imported into the Lisp environment using the following code.

```
(setq
 *C-6-1-small*
 (kern-file2dataset-by-col
  (merge-pathnames
   (make-pathname
    :name "C-6-1-small" :type "krn")
   *MCStylistic-Jun2014-example-files-data-path*)))
--> ((-1 66 63 5/3 0) (0 37 46 1 1) (2/3 66 63 1/3 0)
     (1 49 53 1 1) (1 56 57 1 1) (1 59 59 1 1)
     (1 65 62 1/2 0) (3/2 66 63 1/2 0) (2 49 53 1 1)
     (2 53 55 1 1) (2 59 59 1 1) (2 68 64 7/8 0)
     (23/8 62 61 1/8 0) (3 42 49 1 1) (3 61 60 1/2 0)
     (15/4 66 63 1/4 0) (4 54 56 1 1) (4 61 60 1 1)
     (4 69 65 1 0) (5 54 56 1 1) (5 61 60 1 1))
```

In the above, the first element of each list is the ontime of a note counting in crotchet beats from 0 for bar 1 beat 1, the second element is its MIDI note number, the third element is its morphetic pitch number (see the function pitch&octave2MIDI-morphetic-pair for more details), the fourth element is its duration in crotchet beats, and the fifth element is its staff number (counting from 0 for the top staff). The structure of an input kern file will not be explained in more detail here, as there is an explanation given in Collins (2011, Appendix B.2.3). Suffice it to say that this piece has an anacrusis of one crotchet beat, and the function kern-file2dataset-by-col has recognised this, and set the ontime of the first note to -1, so that bar 1 beat 1 has ontime 0. The functions bar&beat-number-of-ontime and ontime-of-bar&beat-number are useful for converting between ontime/bar and beat number representations. If you want to load lyric, articulation, rest, or tie information, then please see the functions kern-file2points-artic-dynam-lyrics, kern-file2rest-set-by-col, and kern-file2tie-set-by-col for more details.

2.4 MusicXML

MusicXML is a widely used (if verbose) format for digital staff notation. While MCStylistic does not import from MusicXML, it does include a Humdrum extras function that will convert a piece from MusicXML to kern, and kern files can be parsed by MCStylistic (see Sec. 2.3). The function can be found in

MCStylistic/Functions/Third party/humdrum-extras

Mac users should use *xml2hum-mac*, Linux users *xml2hum-linux*, and Windows users *xml2hum-win.exe*. Use of this function is demonstrated now, and occurs external to the Lisp environment.

1. Open up Terminal (Mac or Linux users) or cmd (Windows users).
2. Change directory to the *humdrum-extras* folder mentioned above. In Terminal this can be achieved by issuing the cd (change directory) command.
3. Construct a valid path to an existing MusicXML file, and, after a > symbol, define the path for an output kern file. For example, executing

   ```
   ./xml2hum-mac
    ../../../Example\ files/Example\ data/tallis-love.xml
    > ../../../Example\ files/Example\ data/tallis-love.krn
   ```

   will create a kern version of Thomas Tallis' 'If ye love me' in the specified location. This command can (and probably should) be all on one line: I have broken it on to several lines for ease of reading. If using Linux or Windows, then be sure to replace xml2hum-mac above with the operating-system-appropriate function name.

Please refer to Sec. 2.3 for further instructions on how to import a kern file into the Lisp environment.

If you are having trouble with converting an MusicXML file to a kern file, here are some things to try: make sure the xml2hum function is executable on your machine. Its permissions can be changed with the command

```
chmod 0755 xml2hum
```

Text added above or below a system can sometimes cause errors. A quick fix for this is to open the MusicXML file in a score editor (e.g., MuseScore), delete any such text, resave the file with the appropriate XML extension, and try again. Systems containing braces (e.g., piano right hand braced with piano left hand) sometimes cause issues too. A somewhat tedious fix for this is to open the MusicXML file in a score editor, create a new piece with the same meta information, instruments, etc., copy-paste the old piece one staff at a time, resave the file with the appropriate XML extension, and try again.

2.5 See also

Section 2.1 contained some information for importing/exporting text files. The functions csv2dataset and dataset2csv perform analogous tasks (analogous to read-from-file and write-to-file) for comma-separated variable (csv) files. The function read-from-file-arbitrary is useful for parsing less structued text data.

CHAPTER 3

EXAMPLE CODE

3.1 Music-theoretic analysis with Stravinsqi

This code demonstrates the use of an algorithm called Stravinsqi-Jun2014 (Collins, 2014), which stands for STaff Representation Analysed VIa Natural language Query Input. The algorithm parses a symbolic representation of a piece of music as well as a query string consisting of a natural language expression, and identifies where event(s) specified by the query occur in the music.

Step 1. Specify a piece. In the last chapter we saw several methods for importing a piece of music into the Lisp environment. On this occasion, we do not need to import the piece explicitly, but merely provide the Stravinsqi algorithm with its path, and it will do the rest. The piece will need to be in kern format though. Let us suppose we have a MusicXML excerpt of 'If ye love me' by Thomas Tallis (as shown in Fig. 3.2). So before we can call Stravinsqi, the MusicXML file needs converting to a kern file.

To do this, open up Terminal (separate from Lisp) and navigate using cd (change directory) to

MCStylistic/Functions/Third party/humdrum-extras

Then execute

```
./xml2hum-mac
 ../../../Example\ files/Example\ data/tallis-love.xml
 > ../../../Example\ files/Example\ data/tallis-love.krn
```

This should create a kern file in the same location as the original MusicXML file. This command can (and probably should) be all on one line: I have broken it on to several lines for ease of reading. If using Linux or Windows, then be sure to replace xml2hum-mac above with the operating-system-appropriate function name.

Figure 3.1: Bars 1-13 from 'If ye love me' by Thomas Tallis.

Now, back in Lisp, we can specify the kern file's location:

```
(setq
 notation-path
 *MCStylistic-MonthYear-example-files-data-path*)
(setq notation-name "tallis-love")
(setq
 notation-path&name
 (merge-pathnames
  (make-pathname
   :name notation-name :type "krn") notation-path))
```

Step 2. Ask some questions. In the same location as the kern file there is an (ordinary) XML file called *natural-language-queries.xml*. The first query it contains is 'perfect cadence'. That is, we want to know the bar and beat numbers of the perfect cadence(s) in the music of Figure 3.2. We can pass this query to Stravinsqi by specifying the path of *natural-language-queries.xml*, and the question number '001'. Stravinsqi will parse the query, identify any events that correspond to the query, and write the corresponding bar and beat numbers to a text file. By default, it creates a text file called *dmun01.txt* in the same location as the query file. Try executing the following code in the Lisp Listener:

```
(setq
 question-path&name
 (merge-pathnames
  (make-pathname
   :name "natural-language-queries" :type "xml")
  *MCStylistic-MonthYear-example-files-data-path*))
(setq question-number "001")
(Stravinsqi-Jun2014
 question-number question-path&name notation-path
 notation-name)
```

Checking the text file *dmun01.txt*, it can be confirmed that Stravinsqi's answers to question 1 are:

- bar 2 beat 1 to bar 3 beat 2;
- bar 12 beat 3 to bar 13 beat 4.

Question 2 shows that one can use British or American terminology interchangeably (‘authentic cadence’).

```
(setq question-number "002")
(Stravinsqi-Jun2014
 question-number question-path&name notation-path
 notation-name)
```

Stravinsqi’s answers are the same.

Question 3 asks Stravinsqi to identify a homophonic texture. Its answer is:

- bar 1 beat 1 to bar 5 beat 4.

```
(setq question-number "003")
(Stravinsqi-Jun2014
 question-number question-path&name notation-path
 notation-name)
```

Question 4 shows that Stravinsqi can also answer more straightforward questions, such as ‘dotted crotchet G’. Its answers are:

- bar 8 quaver beats 5 to 7;
- bar 12 quaver beats 5 to 7.

```
(setq question-number "004")
(Stravinsqi-Jun2014
 question-number question-path&name notation-path
 notation-name)
```

Question 5 demonstrates that Stravinsqi can answer compound questions, such as ‘two melodic unisons then a melodic rising third’. This musical event occurs set to ‘and I will pray’, in the soprano and alto voices. Accordingly, Stravinsqi’s answers are:

- bar 5 beat 2 to bar 6 beat 2;
- bar 7 beat 2 to bar 8 beat 2.

```
(setq question-number "005")
(Stravinsqi-Jun2014
 question-number question-path&name notation-path
 notation-name)
```

Question 6 indicates that Stravinsqi can answer queries about text, articulation marks, ties, and rests. For example, for the query ‘word love’, Stravinsqi’s answer is:

- bar 2 beats 1 to 2.

```
(setq question-number "006")
(Stravinsqi-Jun2014
 question-number question-path&name notation-path
 notation-name)
```

Users are encouraged to open up the file *natural-language-queries.xml*, and use copy-paste to add a few questions of their own. The divisions value specifies granularity for Stravinsqi: that is, if you want time windows up to crotchet-beat granularity, then set divisions to 1; for quaver-beat granularity, set divisions to 2, etc. If Stravinsqi cannot answer what seems an entirely reasonable query, or gets the answer wrong, you are welcome to get in touch to request an improvement.

3.2 Functional-harmonic analysis with HarmAn->roman

This code demonstrates the use of an algorithm called HarmAn->roman for segmenting and labelling chords in some input piece of music. The algorithm is based on an implementation of HarmAn by Pardo and Birmingham (2002). HarmAn compares input triples of ontimes, MIDI note numbers, and durations to predefined chord templates, and performs segmentation and segment labelling on this basis. The labels are absolute, for instance (15 2 1 1 8) means that a chord begins on ontime 15, has root 2 modulo 12 (i.e., D), is of type 1 (dom7 chord), lasts 1 beat, and was assigned to this chord template with strength 8.

While useful, this output does not provide a functional-harmonic analysis. I programmed some extra steps to estimate the overall key of the input piece, using the Krumhansl-Schmuckler key-finding algorithm (Krumhansl, 1990), and then to caluclate relative (or functional) harmonic labels by combining the estimate of overall key with the absolute labels output by HarmAn->. For instance, if the overall key is G major, and HarmAn-> output the label D dom7, then my code would convert this to V7. I have taken care to make sure the labelling of diminished 7th chords is correct. The overall program is referred to as HarmAn->roman. It does not handle secondary keys, but might be adapted to do so using a slice through a keyscape (Sapp, 2005).

The input piece should be in kern or point-set format. If your piece is in MusicXML format instead, then it can be converted according to the description in Sec. 2.4. Before proceeding, please note: it is assumed that the package MCStylistic has been loaded, and that the variable *MCStylistic-MonthYear-data-path* has been defined appropriately. Files are imported from the location specified by this variable.

```
#| Step 1 - Load a piece by Chopin. |#
(setq *piece-name* "C-6-1-ed")
(setq
 *path&name*
 (merge-pathnames
  (make-pathname
   :directory
   '(:relative "Dataset")
   :name *piece-name* :type "txt")
  *MCStylistic-MonthYear-data-path*))
(progn
  (setq
   point-set
   (restrict-dataset-in-nth-to-tests
    (read-from-file *path&name*) 0 (list #'<)
    (list 47)))
  "Piece loaded.")

#| Step 2 - Run HarmAn->roman. To see these results
and compare with a ground truth, please see
Fig. 3.2. |#
(HarmAn->roman
 point-set *chord-templates-p&b&min7ths*)
--> (("Ic" (-1 1/3)) ("V" (1/3 2/3)) ("Ic" (2/3 1))
     ("V7" (1 3)) ("i7" (3 7)) ("VII7" (7 39/4))
     ("III" (39/4 12)) ("#vih7c" (12 13))
     ("II7" (13 14)) ("V7c" (14 15))
```

```
     ("vh7c" (15 16)) ("I7" (16 17))
     ("#vio7b" (17 18)) ("ivh7c" (18 19))
     ("VII7" (19 20)) ("vo7b" (20 21))
     ("iiih7c" (21 22)) ("VI7" (22 23))
     ("iih7c" (23 24)) ("Ic" (24 73/3))
     ("V" (73/3 74/3)) ("Ic" (74/3 25))
     ("V7" (25 27)) ("i7" (27 31))
     ("VII7" (31 135/4)) ("III" (135/4 36))
     ("ivh7b" (36 38)) ("III" (38 39))
     ("ivh7b" (39 41)) ("i7b" (41 85/2))
     ("VI" (85/2 43)) ("iih7c" (43 44))
     ("V" (44 45)) ("ivc" (45 136/3))
     ("i" (136/3 137/3)) ("iih7d" (137/3 46))
     ("VIb" (46 47)))
```

3.3 Discovering and rating musical patterns

The folder called *Example files* that comes with MCStylistic contains a Lisp file called *Discovering and rating musical patterns.lisp*. This section will reproduce and discuss chunks of code from this file. The idea is to discover and rate some repeated patterns occurring in bars 1-19 of the Sonata in C minor L10 by Domenico Scarlatti (1685-1757). There is a MIDI file of this excerpt in the subfolder of *Example files* called *Example data*. The algorithm is the Structure Induction Algorithm with Compactness Trawling (SIACT), as defined by Collins, Thurlow, Laney, Willis, and Garthwaite (2010). It combines the Strucutre Induction Algorithm (SIA) with the concept of compactness (Meredith, Lemström, and Wiggins, 2002, 2003). The formula for rating discovered patterns in terms of musical importance was developed by Collins, Laney, Willis, and Garthwaite (2011).

Figure 3.3 shows the Scarlatti excerpt annotated with four patterns *A*-*D* discovered by a music analyst, Dr Jeremy Thurlow.[1] As an example, we will determine which of the four patterns are discovered by SIACT. It is assumed that the package MCStylistic has been loaded, and that the variable `*MCStylistic-Aug2013-example-files-path*` has been defined appropriately. Files are imported from the *Example data* folder mentioned, and several new files exported to the *Example results* folder in the same location.

[1]The analyst's complete annotations and a parallel commentary can be found at http://www.tomcollinsresearch.net

Figure 3.2: Bars 1-16 from Mazurka in F♯ minor op.6 no.1 by Frédéric Chopin, annotated with the output of HarmAn->roman (AO) and a ground truth labelling (GT).

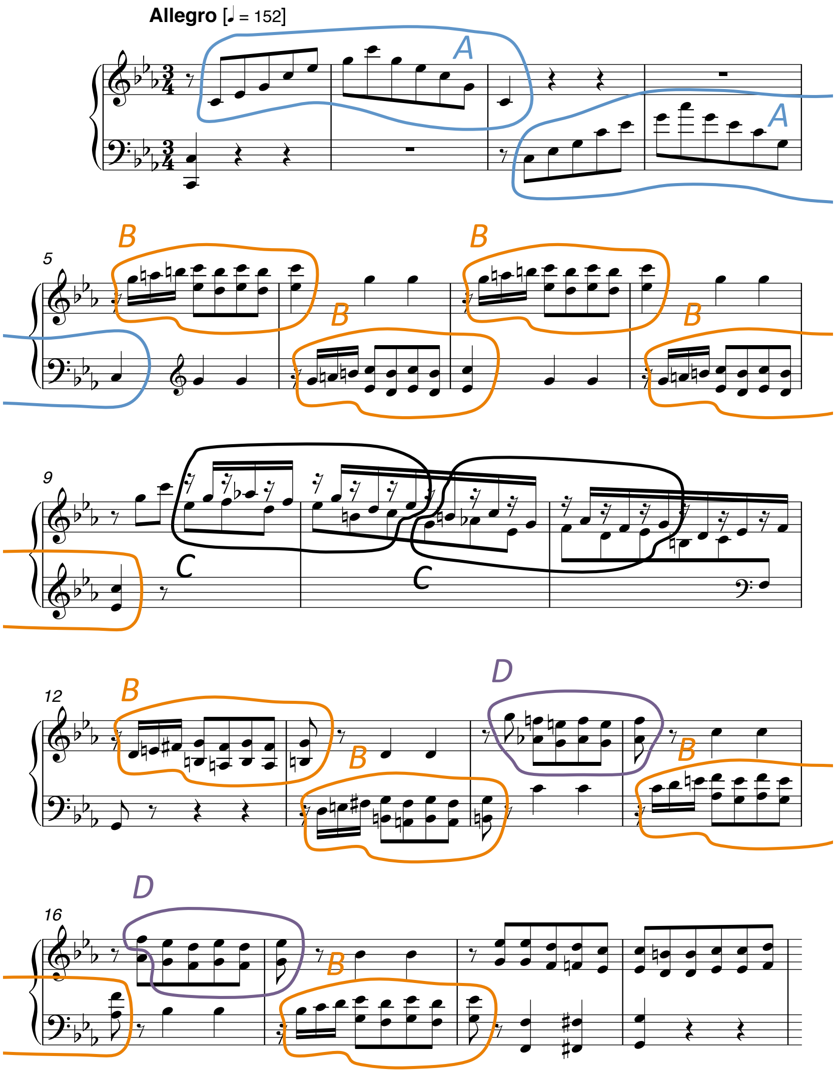

Figure 3.3: Bars 1-19 from the Sonata in C minor L10 by D. Scarlatti. Bounding lines indicate some of the analyst's annotations for this excerpt.

```
#| Step 1 - Set the parameters. |#
(setq *compact-thresh* 2/3)
(setq *cardina-thresh* 3)
(setq *region-type* "lexicographic")
```

The Structure Induction Algorithm with Compactness Trawling (SIACT) has two parameters; a compactness threshold and a points (or cardinality) threshold (Collins et al., 2010). The default version of *compactness* uses a lexicographic region type.

```
#| Step 2 - Load dataset and create projections. |#
(progn
  (setq
   *dataset*
   (read-from-file
    (merge-pathnames
     (make-pathname
      :name "scarlatti-L10-bars1-19" :type "txt")
     *MCStylistic-Aug2013-example-files-data-path*)))
  (setq
   *dataset-1-1-0-1-0*
   (orthogonal-projection-unique-equalp
    *dataset* '(1 1 0 1 0)))
  (setq
   *dataset-1-0-1-0-0*
   (orthogonal-projection-unique-equalp
    *dataset* '(1 0 1 0 0)))
  "Dataset loaded and projections created")
```

The full dataset representation of the excerpt by D. Scarlatti contains dimensions for ontime, MIDI note number (MNN), morphetic pitch number (MPN, for details see Sec. 2.25 of Collins, 2011), duration, and staff. In general, we will want to look at lots of different projections for this dataset, but for this example we consider just two: first, the projection on to ontime, MNN, and duration; second, the projection on to ontime and MPN. These projections are defined using the function orthogonal-projection-unique-equalp.

```
#| Step 3 - Run SIA on projection (1 1 0 1 0). |#
(time
```

```
83  (SIA-reflected-merge-sort
84   *dataset-1-1-0-1-0*
85   (merge-pathnames
86    (make-pathname
87     :name "L 10 (1 1 0 1 0) SIA" :type "txt")
88    *MCStylistic-Aug2013-example-files-results-path*)))
89 ; 0.585303 seconds.
```

This code runs an implementation of the Structure Induction Algorithm (SIA) on the dataset projection for ontime, MNN, and duration, and exports the output to a text file called *L 10 (1 1 0 1 0) SIA.txt* in the *Example results* folder. On a 2.33 GHz machine with 3 GB RAM, this code takes just over half a second to run (line 89).

```
91 #| Step 4 - Run SIACT on projection (1 1 0 1 0). |#
92 (progn
93   (setq
94    *SIA-1-1-0-1-0*
95    (read-from-file
96     (merge-pathnames
97      (make-pathname
98       :name "L 10 (1 1 0 1 0) SIA" :type "txt")
99      *MCStylistic-Aug2013-example-files-results-path*)
100      ))
101   (time
102    (compactness-trawler
103     *SIA-1-1-0-1-0* *dataset-1-1-0-1-0*
104     (merge-pathnames
105      (make-pathname
106       :name "L 10 (1 1 0 1 0) SIACT" :type "txt")
107      *MCStylistic-Aug2013-example-files-results-path*)
108     *compact-thresh* *cardina-thresh* *region-type*)))
109 ; 0.559605 seconds.
```

Lines 93-100 of the above code import the output of SIA from the file *L 10 (1 1 0 1 0) SIA.txt* just created in the *Example results* folder. Then, in lines 102-108, the compactness trawler (CT) is run, using the parameter values defined earlier. The output of the function compactness-trawler is exported to a text file called *L 10 (1 1 0 1 0) SIACT.txt* in the *Example results* folder. Taking steps 3 (SIA) and 4 (CT) together, SIACT has been applied to the excerpt by D. Scarlatti.

```
111 #| Step 5 - Rate discovered patterns for projection
112 (1 1 0 1 0). |#
113 (progn
114   (setq
115    *SIACT-1-1-0-1-0*
116    (read-from-file
117     (merge-pathnames
118      (make-pathname
119       :name "L 10 (1 1 0 1 0) SIACT" :type "txt")
120      *MCStylistic-Aug2013-example-files-results-path*)
121      ))
122   (time
123    (setq
124     *hash-1-1-0-1-0*
125     (evaluate-variables-of-patterns2hash
126      *SIACT-1-1-0-1-0* *dataset-1-1-0-1-0*)))
127   (write-to-file-balanced-hash-table
128    *hash-1-1-0-1-0*
129    (merge-pathnames
130     (make-pathname
131      :name "L 10 (1 1 0 1 0) hash" :type "txt")
132     *MCStylistic-Aug2013-example-files-results-path*))
133   (concatenate
134    'string
135    "Discovered patterns have been rated and placed in"
136    " in the hash table *hash-1-1-0-1-0*. They have"
137    " also been written to a text file for future"
138    " reference."))
139 ; 2.633121 seconds.
```

Lines 114-121 of the above code import the output of SIACT from the file just created in the *Example results* folder, *L 10 (1 1 0 1 0) SIACT.txt*. Each discovered pattern is given a rating for musical importance in lines 123-126, using the function evaluate-variables-of-patterns2hash. The hash table (cf. Sec. 2.5) created by this function is exported to a text file called *L 10 (1 1 0 1 0) hash.txt* in the *Example results* folder.

```
#| Here are the details for pattern A, as annotated in
the Documentation, Fig. 3.1. |#
(disp-ht-el (nth 13 *hash-1-1-0-1-0*))
```

```
144 --> (("name" . "pattern 24") ("compactness" . 1)
145      ("expected occurrences" . 35.375904)
146      ("rating" . 7.539052)
147      ("pattern"
148       (1/2 60 1/2) (1 63 1/2) (3/2 67 1/2) (2 72 1/2)
149       (5/2 75 1/2) (3 79 1/2) (7/2 84 1/2) (4 79 1/2)
150       (9/2 75 1/2) (5 72 1/2) (11/2 67 1/2) (6 60 1))
151      ("translators" (0 0 0) (6 -12 0)) ("index" . 24)
152      ("cardinality" . 12) ("MTP vectors" (6 -12 0))
153      ("compression ratio" . 24/13)
154      ("region"
155       (1/2 60 1/2) (1 63 1/2) (3/2 67 1/2) (2 72 1/2)
156       (5/2 75 1/2) (3 79 1/2) (7/2 84 1/2) (4 79 1/2)
157       (9/2 75 1/2) (5 72 1/2) (11/2 67 1/2) (6 60 1))
158      ("occurrences" . 2))
```

Above we see details for pattern A, as annotated in Fig. 3.3. It has been rated as approximately 7.5 out of 10, using a weighted combination of *compactness*, *expected occurrences*, and *compression ratio*.

Lines 160-218 of the Lisp file *Discovering and rating musical patterns.lisp* will not be reproduced here. Steps 6, 7, and 8 are analogous to steps 3, 4, and 5, applying SIACT and the rating formula to the projection for ontime and MPN. Among the output, we pick out the following result.

```
220 #| Here are the details for pattern B, as annotated in
221 the Documentation, Fig. 3.1. |#
222 (disp-ht-el (nth 2 *hash-1-0-1-0-0*))
223 --> (("name" . "pattern 40") ("compactness" . 13/15)
224      ("expected occurrences" . 44.16958)
225      ("rating" . 8.927441)
226      ("pattern"
227       (73/4 71) (37/2 72) (75/4 73) (19 69) (19 74)
228       (39/2 68) (39/2 73) (20 69) (20 74) (41/2 68)
229       (41/2 73) (21 69) (21 74))
230      ("translators"
231       (-6 0) (-3 -7) (0 0) (3 -7) (15 -10) (18 -17)
232       (24 -11) (30 -12))
233      ("index" . 40) ("cardinality" . 13)
234      ("MTP vectors"
235       (36 -12) (30 -11) (30 -12) (24 -11) (24 -17)
236       (18 -17) (15 -2) (12 -3) (9 -1) (9 -7) (3 -7)
```

```
   (3 -7))
  ("compression ratio" . 26/5)
  ("region"
   (73/4 71) (37/2 72) (75/4 73) (19 64) (19 69)
   (19 74) (39/2 68) (39/2 73) (20 64) (20 69)
   (20 74) (41/2 68) (41/2 73) (21 69) (21 74))
  ("occurrences" . 8))
```

Above we see details for pattern B, as annotated in Fig. 3.3. It has been rated as approximately 8.9 out of 10. Pattern C, as annotated in the Fig. 3.3, is not discovered by SIACT. Close inspection of the music reveals that C is not a translational pattern for either of the projections considered above. Pattern D is not discovered by SIACT either, but a pattern that contains D is discovered. The containing pattern begins where D does, in bar 14, and continues into bar 16. We are prompted to consider why the first occurrence of D is annotated as finishing earlier, in bar 15. It may be that the rests in bar 15 suggest a boundary to the analyst; an appropriate point for demarcating the first occurrence of pattern D. When the sonata is played at full pace, however, these rests can be difficult to perceive.

More information about functions for pattern discovery and rating can be found in Secs. 4.4 and 4.3.

3.4 Stylistic composition with Racchman-Oct2010

The folder called *Example files* that comes with MCStylistic contains a Lisp file called *Stylistic composition with Racchman-Oct2010.lisp.* This section will reproduce and discuss chunks of code from this file. The idea is to demonstrate Racchman-Oct2010 (standing for RAndom Constrained CHain of MArkovian Nodes), which is a model for automated stylistic composition. A date stamp is added to Racchman in case it is superseded by future work. Chapters 8 and 9 of Collins (2011) provide a full explanation of the model. It is similar in spirit to the databases and programs referred to collectively as Experiments in Musical Intelligence (EMI), as outlined by Cope (1996, 2001, 2005).

Here I will exemplify the building of initial states lists and transition lists, and the use of these lists by Racchman-Oct2010 to generate an opening passage of a mazurka in the style of Frédéric Chopin (1810-1849). The user may wish to experiment with different random seeds, resulting in different generated passages. The building of the initial states lists and transition lists takes about two hours on a 2.33 GHz machine with 3 GB RAM, so for users not wishing to wait that long, the resulting files have been placed in the *Example files* → *Example results* → *Racchman-Oct2010 example*. The generation of a new passage takes about 7 minutes in total.

Step 1 of the code involves creating one list called `*variable-names*`, creating another list called `*catalogue*`, and then importing dataset representations for thirty-nine mazurkas into the Lisp environment. The code is not reproduced here; it is verbose but relatively straightforward to understand.

```
#| Step 2 - Create lists of initial/final state-
context pairs, and transition lists. |#
(progn
  (setq
   *initial-states*
   (construct-initial-states
    *variable-names* *catalogue* "beat-spacing-states"
    10 3 3 1))
  (write-to-file
   *initial-states*
   (merge-pathnames
    (make-pathname
     :directory
     '(:relative
       "Racchman-Oct2010 example")
     :name "initial-states" :type "txt")
    *MCStylistic-MonthYear-example-files-results-path*))
  (setq
   *final-states*
   (construct-final-states
    *variable-names* *catalogue* "beat-spacing-states"
    10 3 3 1))
  (write-to-file
   *final-states*
   (merge-pathnames
    (make-pathname
     :directory
     '(:relative
       "Racchman-Oct2010 example")
     :name "final-states" :type "txt")
```

```
381     *MCStylistic-MonthYear-example-files-results-path*))
382   "Initial/final state-context pairs exported.")
383
384 (progn
385   (setq *transition-matrix* nil)
386   (construct-stm
387    *variable-names* *catalogue* "beat-spacing-states"
388    3 3 1)
389   (write-to-file
390    *transition-matrix*
391    (merge-pathnames
392     (make-pathname
393      :directory
394      '(:relative
395        "Racchman-Oct2010 example")
396      :name "transition-matrix" :type "txt")
397     *MCStylistic-MonthYear-example-files-results-path*))
398   (setq *transition-matrix* nil)
399   (construct-stm<-
400    *variable-names* *catalogue* "beat-spacing-states"
401    3 3 1)
402   (write-to-file
403    *transition-matrix*
404    (merge-pathnames
405     (make-pathname
406      :directory
407      '(:relative
408        "Racchman-Oct2010 example")
409      :name "transition-matrix<-" :type "txt")
410     *MCStylistic-MonthYear-example-files-results-path*))
411   "Transition lists exported.")
```

The function construct-initial-states is called in line 356 to construct a list of initial state-context pairs, which are then exported to a text file (lines 359-366). In lines 367-407 there are analogous calls to (and file exports for) the functions construct-final-states, construct-stm, and construct-stm<-. As mentioned, users not wishing to create these lists themselves will find them in the folder called *Racchman-Oct2010 example.*[2]

[2]It should be noted that the internal initial states and internal final states are defined by hand (but completely algorithmically). The list of internal initial states, for instance, con-

```
#| Step 3 - Define parameter values for
Racchman-Oct2010. |#
(progn
  (setq *beats-in-bar* 3) (setq *c-absrb* 10)
  (setq *c-src* 4) (setq *c-bar* 19) (setq *c-min* 19)
  (setq *c-max* 19) (setq *c-beat* 12)
  (setq *c-prob* 0.2) (setq *c-for* 3)
  (setq *c-back* 3)
  (setq
   *checklist*
   (list "originalp" "mean&rangep" "likelihoodp"))
  "Racchman-Oct2010 parameters defined.")
```

The above parameter values for Racchman-Oct2010 are explained fully in Chapters 8 and 9 of Collins (2011). In brief, they control the number of absorptions permitted at each stage of the generating process ($c_{\text{absrb}} = 10$), the number of consecutive states heralding from the same source ($c_{\text{src}} = 4$), the range ($c_{\min} = c_{\max} = \bar{c} = 19$), low-likelihood chords ($c_{\text{prob}} = .2$ and $c_{\text{beat}} = 12$), and a sense of departure/arrival ($c_{\text{for}} = c_{\text{back}} = 3$).

```
#| Step 4 - Import lists of initial/final state-
context pairs, and transition lists. It should be
noted that some variables, such as
*internal-initial-states*, are not required for this
example, so their import code is commented out. |#
(progn
  (setq
   *initial-states*
   (read-from-file
    (merge-pathnames
     (make-pathname
      :directory
      '(:relative
       "Racchman-Oct2010 example")
      :name "initial-states" :type "txt")
     *MCStylistic-MonthYear-example-files-results-path*)
```

tains three beat-spacing states (where these exist) from each of the thirty-nine mazurkas, taken from the time points at which the first three phrases are marked as ending in the score, according to Paderewski's (1953) edition.

```
  ))
#|
(setq
 *internal-initial-states*
 (read-from-file
  (merge-pathnames
   (make-pathname
    :directory
    '(:relative "Racchman-Oct2010 example")
    :name "internal-initial-states" :type "txt")
   *MCStylistic-MonthYear-example-files-path*)
  ))
|#
(setq
 *internal-final-states*
 (read-from-file
  (merge-pathnames
   (make-pathname
    :directory
    '(:relative
     "Racchman-Oct2010 example")
    :name "internal-final-states" :type "txt")
   *MCStylistic-MonthYear-example-files-results-path*)
  ))
#|
(setq
 *final-states*
 (read-from-file
  (merge-pathnames
   (make-pathname
    :directory
    '(:relative "Racchman-Oct2010 example")
    :name "final-states" :type "txt")
   *MCStylistic-MonthYear-example-files-path*)
  ))
|#
(setq
 *stm->*
 (read-from-file
  (merge-pathnames
   (make-pathname
```

```
479       :directory
480       '(:relative
481        "Racchman-Oct2010 example")
482       :name "transition-matrix" :type "txt")
483      *MCStylistic-MonthYear-example-files-results-path*)
484     ))
485   (setq
486    *stm<-*
487    (read-from-file
488     (merge-pathnames
489      (make-pathname
490       :directory
491       '(:relative
492        "Racchman-Oct2010 example")
493       :name "transition-matrix<-" :type "txt")
494      *MCStylistic-MonthYear-example-files-results-path*)
495     ))
496   (concatenate
497    'string
498    "Initial state/context pairs and transition lists"
499    " imported."))
```

The above code imports initial states lists and transition lists, some of which were created in step 1. As the idea is to generate the opening passage of a mazurka, it is not necessary to import the external final states list, nor the internal initial states list, which is why lines 439-448 and 461-470 are commented out.

```
497 #| Step 5 - Import the dataset of an existing Chopin
498 mazurka to be used as a template (op.56 no.2). A
499 template (cf. Def. 9.1 in Collins, 2011) consists of
500 basic information, such as tempo and the pitch of the
501 lowest-sounding note of the first chord, which is
502 transferred to the generated passage. |#
503 (progn
504   (setq
505    *dataset-all*
506    (read-from-file
507     (merge-pathnames
508      (make-pathname
509       :name "C-56-2-ed" :type "txt")
```

```
510      *chopin-mazurka-datasets*)))
511   (setq
512    *dataset-template*
513    (subseq *dataset-all* 0 135))
514   "Template imported.")
```

The above code imports an existing Chopin mazurka to be used as a template. The opening section of this mazurka is defined as the `*dataset-template*`, as the idea is to generate an opening passage.

```
516 #| Step 6 - Generate candidate passages using
517 Racchman-Oct2010 and select one. |#
518 (progn
519   (setq
520    *rs*
521    #.(CCL::INITIALIZE-MRG31K3P-STATE 1912893808
522       1292746109 1626081729 1084533696 825207402
523       71914375))
524   (setq time-a (get-internal-real-time))
525   (setq
526    *output*
527    (generate-beat-MNN-spacing<->
528     *initial-states* *stm->* *internal-final-states*
529     *stm<-* *dataset-template* *checklist*
530     *beats-in-bar* *c-absrb* *c-src* *c-min* *c-max*
531     *c-bar* *c-beat* *c-prob* *c-for* *c-back*))
532   (setq time-b (get-internal-real-time))
533   (float
534    (/
535     (- time-b time-a)
536     internal-time-units-per-second)))
537 ; 196.77089 seconds.
538 (most-plausible-join
539  (third *output*) 23 *dataset-template* *stm->* 3 3 1
540  *c-beat*)
541 ; "united,1,1,superimpose"
```

The above code is at the heart of the Racchman-Oct2010 model, as it is responsible for generating a passage. In lines 519-523 a random seed is defined called `*rs*`. If users wish to experiment with different random seeds, they can alter the numbers in lines 522-523 manually, or use the built-in function make-random-state. Lines 525-531 of the code generate several candidate passages, which takes about 7 minutes. It is worth pointing out that different random seeds will result in different passage generation times. Of all the generated candidates, the output of Racchman-Oct2010 is the passage whose states are all members of the transition list and whose likelihood profile is, on average, closest to that of the template piece. This is determined in lines 538-540.

```
543 #| Step 7 - Export the generated passage to MIDI and
544 text files. |#
545 (progn
546   (setq
547    *output-datapoints*
548    (gethash
549     '"united,1,1,superimpose"
550     (third *output*)))
551   (saveit
552    (merge-pathnames
553     (make-pathname
554      :name "generated-passage1" :type "mid")
555     *MCStylistic-MonthYear-example-files-results-path*)
556    (modify-to-check-dataset
557     (translation
558      *output-datapoints*
559      (list
560       (- 0 (first (first *output-datapoints*)))
561       0 0 0 0)) 950))
562   (write-to-file
563    *output-datapoints*
564    (merge-pathnames
565     (make-pathname
566      :name "generated-passage1" :type "txt")
567     *MCStylistic-MonthYear-example-files-results-path*))
568   (concatenate
569    'string
570    "Generated passage exported to MIDI and text"
571    " files."))
```

The generated passage is exported to a MIDI file, as well as to a text file. When this code is executed, the files *generated-passage1.mid* and *generated-passage1.txt* should appear in *Example files* → *Example results*.

The above code and functions invoked (cf. Sec. 4.6 for additional documentation) represent something of an achievement: this code accompanies the first full description (in Chapters 8 and 9 of Collins, 2011) of a model for generating passages in the style of Chopin mazurkas. There is still much to be achieved, however. Models for automated stylistic composition ought to be evaluated thoroughly in order to gauge stylistic success and identify weaknesses. Evalutation of the passages generated by Racchman-Oct2010 suggests that there are stylistically successful aspects, with room for future improvements (see Chapter 11 of Collins, 2011 for more details).

3.5 Stylistic composition with Racchmaninof-Oct2010

The folder called *Example files* that comes with MCStylistic contains a Lisp file called *Stylistic composition with Racchmaninof-Oct2010.lisp*. This section will reproduce and discuss chunks of code from this file. The code is very similar to that discussed in Sec. 3.4, so in the following we will focus on the parts that differ. The idea is to demonstrate Racchmaninof-Oct2010 (standing for RAndom Constrained CHain of MArkovian Nodes with INheritance Of Form), which is a model for automated stylistic composition. A date stamp is added to Racchmaninof in case it is superseded by future work. Chapters 8 and 9 of Collins (2011) provide a full explanation of the model. The main difference between Racchman-Oct2010 and Racchmaninof-Oct2010 is that the latter includes pattern inheritance. This means the temporal and registral positions of discovered repeated patterns from an existing piece are used as a template to guide the generation of a new passage of music. Both models are similar in spirit to the databases and programs referred to collectively as Experiments in Musical Intelligence (EMI), as outlined by Cope (1996, 2001, 2005).

Here I will exemplify the building of initial states lists and transition lists, and the use of these lists by Racchmaninof-Oct2010 to generate an opening passage of a mazurka in the style of Chopin. The user may wish to experiment with different random seeds, resulting in different generated passages. The building of the initial states lists and transition lists takes about two hours on a 2.9 GHz machine with 8 GB RAM, so for users not wishing to wait that long, the resulting files have been placed in the *Example files* folder, in a folder called *Racchman-Oct2010 example*. The passage generation takes about 5 seconds. Steps 1-4 of the code will not reproduced; it is analogous to the code in Sec. 3.4.

```
#| Step 5 - Import the dataset of an existing Chopin
mazurka (op.68 no.1) to be used as a template with
patterns. A template with patterns (cf. Def. 9.3 in
Collins, 2011) consists of basic information, such as
tempo and the pitch of the lowest-sounding note of the
first chord, but also information to do with
discovered patterns, such as the ontimes of first and
last datapoints, translators, and subset scores. The
second chunk of code here runs SIACT, rates the
discovered patterns, and performs some filtering as
described in Sec. 7.3.1 of Collins (2011). |#
(progn
  (setq
   *dataset-all*
   (read-from-file
    (merge-pathnames
     (make-pathname
      :name "C-68-1-ed" :type "txt")
     *chopin-mazurka-datasets*)))
  (setq
   *dataset-template*
   (subseq *dataset-all* 0 231))
  (setq
   *dataset-projected*
   (orthogonal-projection-unique-equalp
    *dataset-template* '(1 1 1 0 0)))
  "Template imported and projected.")

(progn
  (setq time-a (get-internal-real-time))
  (setq *compact-thresh* 4/5)
  (setq *cardina-thresh* 3)
  (setq *region-type* "lexicographic")
  (setq *duration-thresh* 3)
  (SIA-reflected-merge-sort
   *dataset-projected*
   (merge-pathnames
    (make-pathname
     :directory
     '(:relative
```

```
      "Racchmaninof-Oct2010 example")
    :name "C-68-1 (1 1 1 0 0) SIA" :type "txt")
   *MCStylistic-MonthYear-example-files-results-path*))
 (setq
  *SIA-output*
  (read-from-file
   (merge-pathnames
    (make-pathname
     :directory
     '(:relative
       "Racchmaninof-Oct2010 example")
     :name "C-68-1 (1 1 1 0 0) SIA" :type "txt")
    *MCStylistic-MonthYear-example-files-results-path*)
   ))
 (compactness-trawler
  *SIA-output* *dataset-projected*
  (merge-pathnames
   (make-pathname
    :directory
    '(:relative
      "Racchmaninof-Oct2010 example")
    :name "C-68-1 (1 1 1 0 0) CT" :type "txt")
   *MCStylistic-MonthYear-example-files-results-path*)
  *compact-thresh* *cardina-thresh*
  *region-type*)
 (setq
  *SIACT-output*
  (read-from-file
   (merge-pathnames
    (make-pathname
     :directory
     '(:relative
       "Racchmaninof-Oct2010 example")
     :name "C-68-1 (1 1 1 0 0) CT" :type "txt")
    *MCStylistic-MonthYear-example-files-results-path*)
   ))
 (setq
  *patterns-hash*
  (prepare-for-pattern-inheritance
   *SIACT-output* *dataset-projected*
   *duration-thresh*))
```

```
581   (write-to-file-balanced-hash-table
582    *patterns-hash*
583    (merge-pathnames
584     (make-pathname
585      :directory
586      '(:relative
587        "Racchmaninof-Oct2010 example")
588      :name "C-68-1 (1 1 1 0 0) PH" :type "txt")
589     *MCStylistic-MonthYear-example-files-results-path*))
590   (setq time-b (get-internal-real-time))
591   (float
592    (/
593     (- time-b time-a)
594     internal-time-units-per-second)))
595 ; 0.979296 sec. Patterns discovered, rated, filtered.
```

In lines 512-518 a dataset representation of Chopin's Mazurka in C major op.68 no.1 is imported, and attention restricted to the first 231 datatpoints (line 521). A projection on to the dimensions of ontime, MNN, and MPN is defined in lines 522-525. The second chunk of code (lines 528-589) runs SIACT (an example run of the pattern discovery algorithm SIACT was discussed in Sec. 3.3). Parameters are set in lines 530-533. The parameters `*compact-thresh*`, `*cardina-thresh*`, and `*region-type*` were met before (cf. Sec. 3.3). The parameter `*duration-thresh*` ensures that only discovered patterns of at least this duration (last ontime minus first ontime) are inherited. SIA is run and the results imported in lines 534-551. The compactness trawler (CT) is run and the results imported in lines 552-571. The patterns are rated, filtered, and exported in lines 572-584. The filters include the duration threshold mentioned above, as well as removal of overlapping occurrences of the same pattern, and removal of a pattern Q and its occurrences when Q is the lower-rated of two patterns P and Q, they have the same translators, and Q is a subset of P. For further discussion of these filters, see Sec. 7.3.1 in Collins (2011).

```
562 #| Step 6 - Generate candidate passages using
563 Racchmaninof-Oct2010 and select one. |#
564 (progn
565   (setq
566    *rs*
567    #.(CCL::INITIALIZE-MRG31K3P-STATE
568       1480006552 490947557 697061576 1760965485
```

```
569       2015184206 904512324))
570    (setq
571     *whole-piece-interval*
572     (list
573      (floor (first (first *dataset-all*)))
574      (ceiling (first (my-last *dataset-all*)))))
575    (setq time-a (get-internal-real-time))
576    (setq
577     *interval-output-pairs*
578     (generate-beat-spacing<->pattern-inheritance
579      *external-initial-states*
580      *internal-initial-states* *stm->*
581      *external-final-states* *internal-final-states*
582      *stm<-* *dataset-template* *patterns-hash*
583      *whole-piece-interval* *checklist* *beats-in-bar*
584      *c-absrb* *c-src* *c-bar* *c-min* *c-max* *c-beat*
585      *c-prob* *c-for* *c-back*))
586    (setq time-b (get-internal-real-time))
587    (float
588     (/
589      (- time-b time-a)
590      internal-time-units-per-second)))
591  ; 1.326446 seconds.
```

The above code is at the heart of the Racchmaninof-Oct2010 model, as it is responsible for generating a passage. In lines 595-599 a random seed is defined called *rs*. If users wish to experiment with different random seeds, they can alter the numbers in lines 598-599 manually, or use the built-in function make-random-state. The time interval for which a new passage will be generated is defined in lines 600-604 of the codet. The order in which different portions of this time interval are addressed depends on the subset scores and ratings of discovered patterns. Lines 606-615 generate and select from several candidate passages, which takes about 25 seconds. It is worth pointing out that different random seeds will result in different passage generation times. The way in which discovered patterns are inherited by the generated passage is exemplified in Sec. 9.6 of Collins (2011), but not discussed further here.

```
590  #| Step 7 - Export the generated passage to MIDI and
591  text files. |#
592  (progn
```

```
(setq
 *output-datapoints*
 (interval-output-pairs2dataset
  *interval-output-pairs*))
(saveit
 (merge-pathnames
  (make-pathname
   :name "generated-passage2" :type "mid")
  *MCStylistic-Aug2013-example-files-results-path*)
 (modify-to-check-dataset
  (translation
   *output-datapoints*
   (list
    (- 0 (first (first *output-datapoints*)))
    0 0 0 0)) 850))
(write-to-file
 *output-datapoints*
 (merge-pathnames
  (make-pathname
   :name "generated-passage2" :type "txt")
  *MCStylistic-Aug2013-example-files-results-path*))
(concatenate
 'string
 "Generated passage exported to MIDI and text"
 " files."))
```

The generated passage is exported to a MIDI file, as well as to a text file. When this code is executed, the files *generated-passage2.mid* and *generated-passage2.txt* should appear in *Example files* → *Example results*.

As with Sec. 3.4, the above code and functions invoked (cf. Sec. 4.6 for additional documentation) represent something of an achievement: this code accompanies the first full description (in Chapters 8 and 9 of Collins, 2011) of a model for pattern inheritance. There is still much to be achieved, however. Models for automated stylistic composition ought to be evaluated thoroughly in order to gauge stylistic success and identify weaknesses. Evalutation of the passages generated by Racchmaninof-Oct2010 suggests that there are stylistically successful aspects, with room for future improvements (Chapter 11 of Collins, 2011). In particular, the prize remains unclaimed for demonstrating experimentally that pattern inheritance alone can lead to improved ratings of stylistic success.

3.6 Evaluating pattern discovery algorithms for MIREX 2013

The folder called *Example files* that comes with MCStylistic contains a Lisp file called *Evaluating pattern discovery algorithms for MIREX 2013.lisp*. This section will reproduce and discuss chunks of code from this file. The functions are intended to help participants in the 2013 MIREX Pattern Discovery Task to evaluate their algorithms.

```
17 #| Step 1 - Set the paths for the locations of output
18 patterns and ground truth patterns. |#
19 (setq
20  *algorithms-output-root*
21  (merge-pathnames
22   (make-pathname
23    :directory
24    '(:relative "MIREX 2013 pattern discovery task"))
25   *MCStylistic-Aug2013-example-files-path*))
```

The code begins (lines 19-25) by setting the location of algorithm output. The folder called *Example files* that comes with MCStylistic contains a subfolder called *MIREX 2013 pattern discovery task*. Here you will see example output for four algorithms. In the folder *algorithm1output*, for instance, there are two subfolders (*beethovenOp2No1Mvt3* and *gibbonsSilverSwan1612*) that contain algorithmically discovered patterns within each piece. In order for the evaluation functions to work, please adhere to the folder structure when saving the results of your own algorithm(s):

$$algorithmX \rightarrow pieceY \rightarrow patternZ \rightarrow occurrences \rightarrow csv$$

Each occurrence of each discovered pattern should be saved as a separate csv file (*occ1.csv*, *occ2.csv*,..., *occm.csv*) with one ontime-pitch pair per row. You can either replace the example algorithm output with your own results, or point `*algorithms-output-root*` to the appropriate location. More than one algorithm can be evaluated simultaneously (see below).

```
26 (setq *task-version* "polyphonic")
27 (setq
28  *annotations-poly*
29  (list
30   "bruhn" "barlowAndMorgensternRevised"
```

```
31   "sectionalRepetitions" "schoenberg" "tomcollins"))
32 (setq
33  *annotations-mono*
34  (list
35   "bruhn" "barlowAndMorgenstern"
36   "barlowAndMorgensternRevised" "sectionalRepetitions"
37   "schoenberg" "tomcollins"))
```

If your algorithm works on the polyphonic version of each ground truth piece, then leave `*task-version*` in line 26 as ‘polyphonic’. Otherwise, please switch to ‘monophonic’. The variables `*annotations-poly*` and `*annotations-mono*` defined in lines 39-49 tell the evaluation functions which annotations should be included as ground truth. It is possible to alter these lists to begin investigating the strengths and weaknesses of a pattern discovery algorithm, but please note that some annotations are empty for some pieces (for example, there is no ‘bruhn’ annotation for ‘gibbonsSilverSwan1612’).

```
38 (setq
39  *ground-truth-paths*
40  (list
41   (merge-pathnames
42    (make-pathname
43     :directory
44     '(:relative "beethovenOp2No1Mvt3"))
45   *jkuPattsDevDB-Aug2013-gtr-path*)
46   (merge-pathnames
47    (make-pathname
48     :directory
49     '(:relative "gibbonsSilverSwan1612"))
50   *jkuPattsDevDB-Aug2013-gtr-path*)))
51 #|
52 (setq
53  *ground-truth-paths*
54  (cl-fad:list-directory
55   *jkuPattsDevDB-Aug2013-gtr-path*))
56 |#
57 (setq
58  *algorithm-output-paths*
59  (cl-fad:list-directory *algorithms-output-root*))
60 ; Save the calculated metrics to this csv file.
```

```
61 (setq
62  *csv-save-path&name*
63  (merge-pathnames
64   (make-pathname
65    :name "calculated-metrics" :type "csv")
66   *MCStylistic-Aug2013-example-files-path*))
```

Lines 38-50 define the locations of two ground truth annotations (‘beethoven Op2No1Mvt3’ and ‘gibbonsSilverSwan1612’). The code makes use of the variable `*jkuPattsDevDB-Aug2013-gtr-path*`, which is defined in *setup.lisp* (see Sec. 1.2). Commented-out in lines 51-56 (by ‘#|’ and ‘|#’) is an alternative definition of `*ground-truth-paths*`, which uses the function cl-fad:list-directory to create a list of all available ground truth annotations. I offer the first alternative, in case in the beginning you want to restrict evaluation to a couple of pieces.

Lines 57-59 constitute analogous code for creating a list of algorithm outputs. The function cl-fad:list-directory is used to create a list of all algorithm output located at `*algorithms-output-root*`. So if the output of more than one algorithm is located here, then all these algorithms will be evaluated simultaneously. If you want to restrict evaluation to a couple of algorithms located at `*algorithms-output-root*`, then the code in lines 38-50 could be copied, pasted, and adapted easily:

- replace `*ground-truth-paths*` with `*algorithm-output-paths*`;
- replace ‘beethovenOp2No1Mvt3’ with the folder name of the first algorithm’s output; replace ‘gibbonsSilverSwan1612’ with the folder name of the second algorithm’s output, etc.;
- finally, each `*jkuPattsDevDB-Aug2013-gtr-path*` should be replaced with `*algorithms-output-root*`.

It should be noted that the evaluation functions will still work if an algorithm is missing output for one or more ground truth pieces (the line in the results file will be blank). Lines 61-66 specify the location to which the results file will be saved. A version of the results is given in Table 3.1, and soon it will be discussed in more detail, after we have looked over code for specifying metrics and metric parameters.

```
68 #| Step 2 - List metrics to calculate and any
69 parameters. |#
70 (setq
```

Table 3.1: Evaluation metrics of some pattern discovery algorithms for pieces by Beethoven and Gibbons.

algorithm1output

Piece	P	R	P_{est}	R_{est}	P_{occ}	R_{occ}
beethovenOp2No1Mvt3	1	1	1	1	1	1
gibbonsSilverSwann1612	1	1	1	1	1	1

algorithm2output

Piece	P	R	P_{est}	R_{est}	P_{occ}	R_{occ}
beethovenOp2No1Mvt3	0	0	.03	.02	0	0
gibbonsSilverSwann1612	0	0	.02	.03	0	0

algorithm3output

Piece	P	R	P_{est}	R_{est}	P_{occ}	R_{occ}
beethovenOp2No1Mvt3	.36	.57	.73	.97	.95	.97
gibbonsSilverSwann1612	0	0	.98	.27	.98	.98

algorithm4output

Piece	P	R	P_{est}	R_{est}	P_{occ}	R_{occ}
beethovenOp2No1Mvt3	0	0	.26	.43	.97	.97
gibbonsSilverSwann1612	.02	.25	.34	.86	.84	.84

```
71  *metrics-to-calculate*
72  (list
73   "precision" "recall" "precision-est-card"
74   "recall-est-card" "precision-occ-card"
75   "recall-occ-card"))
76 #|
77 (setq
78  *metrics-to-calculate*
79  (list
80   "precision-est-card" "recall-est-card"
81   "precision-occ-card" "recall-occ-card"))
82 (setq
83  *metrics-to-calculate*
84  (list
85   "precision-est-card" "recall-est-card"
86   "precision-occ-card" "recall-occ-card"
87   "precision-est-match" "recall-est-match"))
88 |#
89 (setq
90  *metric-parameters*
91  (list
92   (list "score-thresh" .75) (list "tolp" t)
93   (list "translationp" nil) (list "card-limit" 150)))
94 (setq *file-type* "csv")
```

Lines 70-75 set the metrics that will be calculated for each algorithm on each ground truth piece. We do not discuss the definitions of the metrics here, as they are given on the MIREX page for the Pattern Discovery Task. Some alternative versions of metric lists are commented out in lines 76-88.[3]

Metric parameters are set in lines 89-94. The 'score-thresh' parameter (line 92) is used by the *occurrence precision* and *occurrence recall* metrics. A value of .75 indicates that an output pattern must score at least .75 in terms of symbolic music similarity to a ground truth pattern, in order for it to be considered *discovered*. The 'tolp' parameter (line 92) is set to `t` (true), and allows for equality up to a tolerance of 10^{-5}. This can be useful for handling rounding errors in csv files. The 'translationp' parameter (line 93) is set to `nil` (false). As such, time- or pitch-shifted discoveries of a ground

[3] Use of the 'match' metrics is not recommended in this Lisp implementation: the implementation is slow for verbose algorithm output, and unstable for sectional repetitions. An optimised version will be employed for the evaluation proper.

truth pattern do not count as successes. If at least one occurrence of an output pattern is very similar to at least one occurrence of a ground truth pattern, however, then the *establishment precision* (P_{est}) and *establishment recall* (R_{est}) metrics will reward the discovery. The 'card-limit' parameter (line 93) is an attempt to improve the stability of the 'match' metrics, causing the function matching-score to use an estimate if a pattern contains more than 150 ontime-pitch pairs. Finally, the variable `*file-type*` controls whether output and ground truth patterns are imported from csv or text files. I recommend the former, as many other programs have the ability to read/write csv files, and, as mentioned above, functionality has been included in MCStylistic to handle rounding errors.

```
96  #| Step 3 - Calculate the metrics. |#
97  (setq
98   *ans*
99   (pattern-discovery-metrics
100   *algorithm-output-paths* *ground-truth-paths*
101   *csv-save-path&name* *task-version*
102   *annotations-poly* *metrics-to-calculate*
103   *metric-parameters* *file-type*))
```

The function pattern-discovery-metrics is called in lines 99-103. It calculates the specified metrics for the algorithm output and ground truth pieces, and writes the results to a csv file. Table 3.1 shows the results for the current example. Algorithms 1 and 2 are sanity checks. For algorithm 1, I defined the output to be the ground truth patterns. As expected, all metrics for algorithm 1 are at ceiling. Conversely for algorithm 2, I defined the output by swapping the ground truths for 'beethovenOp2No1Mvt3' and 'gibbonsSilverSwan1612'. As expected, the metrics for algorithm 2 are at or very near the floor of zero. Rows for algorithms 3 and 4 represent the evaluation of real output. Algorithm 4 is more verbose than algorithm 3, which is reflected partly in algorithm 4's lower values for precision. It can be seen that the *establishment precision* (P_{est}) and *establishment recall* (R_{est}) metrics are more robust to slight differences between output and ground truth patterns than are standard *precision* (P) and *recall* (R). Compare, say, $P = 0$ with $P_{\text{est}} = .98$ for algorithm 3 on 'gibbonsSilverSwan1612'. The *occurrence precision* (P_{occ}) and *occurrence recall* (R_{occ}) metrics assess how well an algorithm discovers *all occurrences* of a pattern, given that it has discovered at least one occurrence (controlled by the 'score-thresh' parameter). For example, although algorithm 4 does not retrieve many of the ground truth patterns in 'beethovenOp2No1Mvt3' ($R_{\text{est}} = .43$), for those it does retrieve it does a good job of identifying all occurrences ($R_{\text{occ}} = .97$).

A more general introduction to the Pattern Discovery Task, including metric definitions, can be found on the corresponding MIREX page.

CHAPTER 4

DOCUMENTATION FOR INDIVIDUAL FUNCTIONS

4.1 Maths foundation

4.1.1 List processing

These functions do simple but important things with lists. For example, the function add-to-list adds the first argument (a number) to each element of the second argument (a list). A slightly more complicated function called remove-nth removes the nth element from a given list.

add-to-list

Started, last checked	19/1/2010, 19/1/2010
Location	List processing
Calls	
Called by	
Comments/see also	

Example:

```
(add-to-list 5 '(1 2 3))
--> (6 7 8)
```

This function adds a constant to each element of a list.

add-to-nth

Started, last checked	19/1/2010, 19/1/2010
Location	List processing
Calls	firstn
Called by	
Comments/see also	

Example:

```
(add-to-nth 1 3 '(1 2 3 5 9))
--> (1 2 4 5 9)
```

This function adds a constant to the nth element of a list.

choose

Started, last checked	19/1/2010, 19/1/2010
Location	List processing
Calls	factorial, factorial-j
Called by	
Comments/see also	

Example:

```
(choose 9 5)
--> 126
```

This function returns 'n choose r', that is $n!/(r!(n-r)!)$, where n and r are natural numbers or zero.

constant-vector

Started, last checked	19/1/2010, 19/1/2010
Location	List processing
Calls	
Called by	
Comments/see also	

Example:

```
(constant-vector 2.4 6)
--> (2.4 2.4 2.4 2.4 2.4 2.4)
```

This function gives a constant vector of prescribed length.

cyclically-permute-list-by

Started, last checked	9/3/2013, 9/3/2013
Location	List processing
Calls	
Called by	
Comments/see also	

Example:

```
(cyclically-permute-list-by
 '(17.77 0.15 14.93 0.16 4.95) 2)
--> (14.93 0.16 4.95 17.77 0.15)
```

This function moves the ith item of a list to the first item in the output list, where $i - 1$ is the second argument. The $i - 1$th item is moved to the last item in the output list, etc.

factorial

Started, last checked	19/1/2010, 19/1/2010
Location	List processing
Calls	
Called by	choose
Comments/see also	

Example:

```
(factorial 5)
--> 120
```

This function returns $n(n-1)(n-2)\cdots 3 \cdot 2 \cdot 1$, where n is a natural number or zero.

factorial-j

Started, last checked	19/1/2010, 19/1/2010
Location	List processing
Calls	
Called by	choose
Comments/see also	

Example:

```
(factorial-j 9 3)
--> 3024
```

The arguments of this function are $n > j$, both natural numbers or zero. The answer $n(n-1)(n-2)\cdots(n-j)$ is returned. If $j \geq n$ or $j < 0$, 1 is returned. This function makes the function choose more efficient by avoiding direct calculation of $n!/r!$.

first-n-naturals

Started, last checked	19/1/2010, 19/1/2010
Location	List processing
Calls	
Called by	
Comments/see also	

Example:

```
(first-n-naturals 5)
--> (5 4 3 2 1)
```

This function returns the first n natural numbers as a list.

firstn

Started, last checked	19/1/2010, 19/1/2010
Location	List processing
Calls	
Called by	add-to-nth, remove-nth
Comments/see also	

Example:

```
(firstn 3 '(3 4 (5 2) 2 0))
--> (3 4 (5 2))
```

This function returns the first n items of a list.

index-item-1st-occurs

Started, last checked	19/1/2010, 19/1/2010
Location	List processing
Calls	
Called by	
Comments/see also	

Example:

```
(index-item-1st-occurs 2 '(1 0 0 2 4 2))
--> 3
```

Taking an item and a list of items as its arguments, this function returns the index at which the given item first occurs, counting from zero. If the item does not occur at all then the function returns NIL.

last-first

Started, last checked	19/1/2010, 19/1/2010
Location	List processing
Calls	
Called by	lastn
Comments/see also	

Example:

```
(last-first 3 '(3 4 (5 2) 2 0))
--> (0 2 (5 2))
```

This function returns the last n items of a list, but in reverse order. NB the function last returns a list rather than a list element.

lastn

Started, last checked	19/1/2010, 19/1/2010
Location	List processing
Calls	last-first
Called by	add-to-nth, remove-nth
Comments/see also	

Example:

```
(lastn 3 '(3 4 (5 2) 2 0))
--> ((5 2) 2 0)
```

This function returns the last n items of a list.

multiply-list-by-constant

Started, last checked	19/1/2010, 19/1/2010
Location	List processing
Calls	
Called by	
Comments/see also	

Example:

```
(multiply-list-by-constant '(2 0) 5)
--> (10 0)
```

Two arguments are supplied to this function: a list and a constant. A list is returned, containing the result of multiplying each element of the list by the constant.

my-last

Started, last checked	19/1/2010, 19/1/2010
Location	List processing
Calls	
Called by	
Comments/see also	my-last-string

Example:

```
(my-last '(1 3 6 7))
--> 7
```

Returns the last element of a list as an element, not as a list.

nth-list

Started, last checked	19/1/2010, 19/1/2010
Location	List processing
Calls	
Called by	
Comments/see also	

Example:

```
(nth-list '(1 3 0) '(6 -3 -88 0 4 44))
--> (-3 0 6)
```

This function applies the function nth recursively to the second list argument, according to the items of the first list argument.

nth-list-of-lists

Started, last checked	19/1/2010, 19/1/2010
Location	List processing
Calls	
Called by	
Comments/see also	

Example:

```
(nth-list-of-lists 0 '((48 2) (-50 0) (-5 5)))
--> (48 -50 5)
```

This function takes two arguments; an item n and a list of sub-lists. It returns the nth item of each sub-list as a list.

positions

Started, last checked	19/1/2010, 16/5/2014
Location	List processing
Calls	
Called by	
Comments/see also	Amended 16/5/2014 to allow passing an equality-checking function handle.

Example:

```
(positions
 '(4 0) '((0 1) (3 2) (4 0) (2 2) (-4 4) (4 0) (5 6)))
--> (2 5)
```

This code returns the positions of a query in a list.

remove-nth

Started, last checked	19/1/2010, 19/1/2010
Location	List processing
Calls	
Called by	remove-nth-list, sort-by-col-asc, sort-by-col-desc
Comments/see also	

Example:

```
(remove-nth 4 '(6 4 5 5 2 3 1))
--> (6 4 5 5 3 1)
```

This code removes the nth item of a list, counting from zero.

remove-nth-list

Started, last checked	19/1/2010, 19/1/2010
Location	List processing
Calls	remove-nth
Called by	
Comments/see also	

Example:

```
(remove-nth-list '(3 5 0) '(1 2 3 4 5 6 7))
--> (2 3 5 7)
```

The function remove-nth-list applies the function remove-nth recursively to the second argument, according to the indices in the first argument, which do not have to be ordered or distinct.

test-equalp-nth-to-x

Started, last checked	19/1/2010, 19/1/2010
Location	List processing
Calls	
Called by	
Comments/see also	Deprecated.

Example:

```
(test-equalp-nth-to-x '(3 5 0) 1 5)
--> T
```

The first argument to this function is a list of numbers, the second argument is an index that refers to one of these numbers. If this number is equalp to the third argument, T is returned, and nil otherwise.

test-equalp-nth-to-xs

Started, last checked	19/1/2010, 19/1/2010
Location	List processing
Calls	
Called by	
Comments/see also	Deprecated.

Example:

```
(test-equalp-nth-to-xs '(3 5 0) 1 '(2 4 5 6))
--> T
```

The first argument to this function is a list of numbers, the second argument is an index that refers to one of these numbers. This number is tested for membership in the third argument, and the output is the result of this test. Note it will not recognise 1.0 as 1.

4.1.2 Set operations

These functions enable the formation of unions and intersections over lists that represent finite sets in n-dimensional space. It is also possible to find translators of a pattern in a dataset.

add-two-lists

Started, last checked	13/1/2010, 13/1/2010
Location	Set operations
Calls	
Called by	test-equal<potential-translator, translation
Comments/see also	add-two-lists-mod-2nd-n

Example:

```
(add-two-lists '(4 7 -3) '(8 -2 -3))
--> (12 5 -6)
```

Adds two lists element-by-element. It is assumed that elements of list arguments are numbers, and the list arguments are of the same length. An empty first (but not second) argument will be tolerated.

check-potential-translators

Started, last checked	13/1/2010, 13/1/2010
Location	Set operations
Calls	test-equal<potential-translator
Called by	translators-of-pattern-in-dataset
Comments/see also	check-potential-translators-mod-2nd-n

Example:

```
(check-potential-translators
 '(3 52) '((0 0) (1 2) (1 5) (2 7))
 '((0 60) (3 52) (4 57) (5 59)))
--> ((0 0) (1 5) (2 7))
```

Checks whether the first argument, when translated by each member of the second argument, is a member of the third argument. Members of the second argument that satisfy this property are returned.

equal-up-to-tol

Started, last checked	9/3/2013, 9/3/2013
Location	Set operations
Calls	
Called by	cardinality-score, equalp-score, frequency-count, most-frequent-difference-vector
Comments/see also	

Example:

```
(equal-up-to-tol '(2 2 4 5) '(2 2 4 4.501) 1/2)
--> T
(equal-up-to-tol '(2 2.5 4 5) '(2 2 4 5) 1/2)
--> T
(equal-up-to-tol '(2 2 4.5 5) '(2 2 4 5) 1/3)
--> NIL
```

This function compares two lists for equality, up to a given tolerance.

insert-retaining-sorted-asc

Started, last checked	13/1/2010, 13/1/2010
Location	Set operations
Calls	vector<vector
Called by	union-multidimensional-sorted-asc
Comments/see also	

Example:

```
(insert-retaining-sorted-asc
 '(5 0) '((-6 2) (-4 1) (8 0)))
--> ((-6 2) (-4 1) (5 0) (8 0))
```

Two arguments are supplied to this function: a (real) vector and a strictly-ascending list of (real) vectors (of the same dimension). The first argument is included in the second and output, so that it remains a strictly-ascending list of vectors. (Note this means that if the first argument is already in the list, then this list is output unchanged.)

intersection-multidimensional

Started, last checked	13/1/2010, 13/1/2010
Location	Set operations
Calls	test-equal<list-elements
Called by	intersections-multidimensional
Comments/see also	

Example:

```
(intersection-multidimensional
 '((4 8 8) (4 7 6) (5 -1 0) (2 0 0))
 '((4 6 7) (2 0 0) (4 7 6)))
--> ((4 7 6) (2 0 0))
```

Like the built-in Lisp function intersection, this function returns the intersection of two lists. Unlike the built-in Lisp function, this function handles lists of lists.

intersections-multidimensional

Started, last checked	13/1/2010, 13/1/2010
Location	Set operations
Calls	intersection-multidimensional, null-list-of-lists
Called by	
Comments/see also	

Example:

```
(intersections-multidimensional
 '(((4 8 8) (4 7 6) (5 -1 0) (2 0 0))
   ((4 6 7) (2 0 0) (4 7 6))
   ((4 7 6) (2 1 0) (5 -1 0) (5 0 5))))
--> ((4 7 6))
```

The single argument to this function consists of n lists of lists (of varying length). Their intersection is calculated and returned.

null-list-of-lists

Started, last checked	13/1/2010, 13/1/2010
Location	Set operations
Calls	
Called by	intersections-multidimensional
Comments/see also	

Example:

```
(null-list-of-lists
 '(((4 8 8) (4 7 6) (5 -1 0) (2 0 0))
   ()
   ((4 7 6) (2 1 0) (5 -1 0) (5 0 5))))
--> T
```

The single argument to this function consists of n lists of lists (of varying length). If any one of these lists is empty then the value T is returned. Otherwise the value NIL is returned. Note that a null argument gives the output NIL.

set-difference-multidimensional-sorted-asc

Started, last checked	13/1/2010, 13/1/2010
Location	Set operations
Calls	test-equal<list-elements
Called by	
Comments/see also	

Example:

```
(set-difference-multidimensional-sorted-asc
 '((-1 1) (0 1) (1 1) (2 3) (4 -4) (4 3))
 '((-1 1) (0 1) (2 3) (3 2) (4 3)))
--> ((1 1) (4 -4))
```

This function computes the set difference $A \backslash B = \{a \in A \mid a \notin B\}$ for point sets.

sort-dataset-asc

Started, last checked	13/1/2010, 16/6/2014
Location	Set operations
Calls	
Called by	union-multidimensional-sorted-asc, unions-multidimensional-sorted-asc
Comments/see also	16/6/2014, introduced an optional function argument.

Example:

```
(sort-dataset-asc
 '((1 1) (0 1) (4 4) (0 1) (1 1) (-2 3) (4 4) (4 3)))
--> ((-2 3) (0 1) (0 1) (1 1)
     (1 1) (4 3) (4 4) (4 4))
```

This function takes one argument: a dataset. It sorts the dataset ascending by each dimension in turn. By the definition of *dataset*, the dataset should not contain repeated values. If it does these will be removed.

subset-multidimensional

Started, last checked	13/1/2010, 13/1/2010
Location	Set operations
Calls	test-equal<list-elements
Called by	subset-score-of-pattern
Comments/see also	

Example:

```
(subset-multidimensional
 '((2 56) (6 60)) '((0 62) (2 56) (6 60) (6 72)))
--> T
```

This function returns T if and only if the first argument is a subset of the second, and it is assumed that the second list is sorted ascending.

subtract-list-from-each-list

Started, last checked	13/1/2010, 13/1/2010
Location	Set operations
Calls	subtract-two-lists
Called by	translators-of-pattern-in-dataset
Comments/see also	subtract-list-from-each-list-mod-2nd-n

Example:

```
(subtract-list-from-each-list
 '((8 -2 -3) (4 6 6) (0 0 0) (4 7 -3)) '(4 7 -3))
--> ((4 -9 0) (0 -1 9) (-4 -7 3) (0 0 0))
```

The function subtract-two-lists is applied recursively to each sublist in the first list argument, and the second argument.

subtract-two-lists

Started, last checked	13/1/2010, 13/1/2010
Location	Set operations
Calls	
Called by	subtract-list-from-each-list, test-translation-no-length-check
Comments/see also	subtract-two-lists-mod-2nd-n

Example:

```
(subtract-two-lists '(4 7 -3) '(8 -2 -3))
--> (-4 9 0)
```

Subtracts the second list from the first, element-by-element. It is assumed that elements of list arguments are numbers, and the list arguments are of the same length. An empty first (but not second) argument will be tolerated.

test-equal<list-elements

Started, last checked	13/1/2010, 13/1/2010
Location	Set operations
Calls	
Called by	intersection-multidimensional, set-difference-multidimensional-sorted-asc
Comments/see also	

Example:

```
(test-equal<list-elements
 '((0 1) (0 2) (1 1) (3 1/4)) '(1 1))
--> T
```

The first argument is a list of sublists, assumed to be sorted ascending by each of its elements in turn. We imagine it as a set of vectors (all members of the same n-dimensional vector space). The second argument $\mathbf{v}$ (another list) is also an n-dimensional vector. If v_1 is less than v_2, the first element of the first element of the first argument then NIL is returned, since we know the list is sorted ascending. Otherwise each item is checked for equality.

test-equal<potential-translator

Started, last checked	13/1/2010, 13/1/2010
Location	Set operations
Calls	add-two-lists
Called by	check-potential-translators
Comments/see also	test-equal<potential-translator-mod-2nd-n

Example:

```
(test-equal<potential-translator
 '((0 1) (0 2) (1 2) (3 1/4)) '(0 1) '(1 1))
--> ((1 1))
```

This function is very similar in spirit to test-equal<list-elements. The first argument here is a dataset, the second is a member of some pattern (so also a member of the dataset), and the third is a potential translator of the patternpoint. If the potential translator is really a translator, it is returned, and NIL otherwise.

test-translation

Started, last checked	13/1/2010, 13/1/2010
Location	Set operations
Calls	test-translation-no-length-check
Called by	check-potential-translators
Comments/see also	test-translation-mod-2nd-n

Example:

```
(test-translation
 '((2 2) (4 5)) '((11 6) (13 9)))
--> T
```

If the first argument to this function, a list, consists of vectors of uniform dimension that are the pairwise translation of vectors in another list (the function's second argument), then T is returned, and nil otherwise. The length of the vectors is checked first for equality, then passed to the function test-translation-no-length-check if equal.

test-translation-no-length-check

Started, last checked	13/1/2010, 13/1/2010
Location	Set operations
Calls	subtract-two-lists
Called by	test-translation
Comments/see also	test-translation-mod-2nd-n-no-length-check

Example:

```
(test-translation-no-length-check
 '((2 2) (4 5)) '((11 6) (13 9)))
--> T
```

If the first argument to this function, a list, consists of vectors of uniform dimension that are the pairwise translation of vectors in another list (the function's second argument), then T is returned, and nil otherwise. The length of the vectors is not checked for equality. (At present the function returns T if two empty lists are provided as arguments.)

translation

Started, last checked	13/1/2010, 13/1/2010
Location	Set operations
Calls	add-two-lists
Called by	translational-equivalence-class
Comments/see also	translation-mod-2nd-n

Example:

```
(translation '((8 -2 -3) (4 6 6) (4 7 -3)) '(3 1 0))
--> ((11 -1 -3) (7 7 6) (7 8 -3))
```

The first argument is a list of sublists, but we imagine it as a set of vectors (all members of the same n-dimensional vector space). The second argument—another list—is also an n-dimensional vector, and this is added to each of the members of the first argument. 'Added' means vector addition, that is element-wise, so the resulting set is a translation of the first argument by the second.

translational-equivalence-class

Started, last checked	13/1/2010, 13/1/2010
Location	Set operations
Calls	translation
Called by	
Comments/see also	

Example:

```
(translational-equivalence-class
 '((6 2) (7 1/2) (15/2 1/4) (31/4 1/4) (8 1) (9 1))
 '((0 1) (0 4/3) (0 2) (1 1) (4/3 1/3) (5/3 1/3)
   (2 1/2) (2 1) (5/2 1/2) (3 1/2) (3 2) (7/2 1/2)
   (4 1/2) (4 1) (9/2 1/2) (5 1) (6 1) (6 2)
   (7 1/2) (7 1) (15/2 1/4) (31/4 1/4) (8 1) (9 1)
   (9 2) (10 1/2) (10 1) (21/2 1/4) (43/4 1/4)
   (11 1) (12 1) (12 2) (13 1/2) (13 2) (27/2 1/4)
   (55/4 1/4) (14 1) (14 2) (15 1) (16 1/3) (16 2)
   (49/3 1/3) (50/3 1/3) (17 1)))
--> (((6 2) (7 1/2) (15/2 1/4)
      (31/4 1/4) (8 1) (9 1))
     ((9 2) (10 1/2) (21/2 1/4)
      (43/4 1/4) (11 1) (12 1))
     ((12 2) (13 1/2) (27/2 1/4)
      (55/4 1/4) (14 1) (15 1)))
```

The function takes two arguments: a pattern P and a dataset D. It returns the translational equivalence class of P in D.

translations

Started, last checked	13/1/2010, 13/1/2010
Location	Set operations
Calls	translation
Called by	
Comments/see also	translations-mod-2nd-n

Example:

```
(translations
 '((1 2) (2 4)) '((0 0) (1 2)))
--> (((1 2) (2 4)) ((2 4) (3 6)))
```

There are two arguments to this function, a pattern and some translators. The pattern is translated by each translator and the results returned.

translators-of-pattern-in-dataset

Started, last checked	13/1/2010, 13/1/2010
Location	Set operations
Calls	check-potential-translators, subtract-list-from-each-list
Called by	translational-equivalence-class
Comments/see also	Should be deprecated by implementing the version in Ukkonen, Lemström, and Mäkinen (2003). See also translators-of-pattern-in-dataset-mod-2nd-n.

Example:

```
(translators-of-pattern-in-dataset
 '((8 3) (8 7))
 '((4 7) (8 -3) (8 3) (8 7) (9 -3) (10 7) (11 -3)
   (13 -3) (13 1)))
--> ((0 0) (5 -6))
```

A pattern and dataset are provided. The transaltors of the pattern in the dataset are returned.

union-multidimensional-sorted-asc

Started, last checked	13/1/2010, 13/1/2010
Location	Set operations
Calls	insert-retaining-sorted-asc, sort-dataset-asc
Called by	union-multidimensional-sorted-asc
Comments/see also	

Example:

```
(union-multidimensional-sorted-asc
 '((-5 0 4) (-4 3 1) (8 5 3) (8 6 0))
 '((-4 3 1) (-6 2 2) (8 5 0) (8 6 0))
 T)
--> ((-6 2 2) (-5 0 4) (-4 3 1) (8 5 0)
     (8 5 3) (8 6 0))
```

Two lists of (real) vectors of the same dimension are supplied to this function. If the first is sorted strictly ascending already, a third argument of T should be supplied to prevent it being sorted so. The union of these lists is output and remains strictly ascending.

unions-multidimensional-sorted-asc

Started, last checked	13/1/2010, 13/1/2010
Location	Set operations
Calls	sort-dataset-asc, union-multidimensional-sorted-asc
Called by	
Comments/see also	

Example:

```
(unions-multidimensional-sorted-asc
 '(((12 10) (0 0) (1 2)) ((0 0) (1 5)) ((6 6))))
--> ((0 0) (1 2) (1 5) (6 6) (12 10))
```

The function union-multidimensional-sorted- asc is applied recursively to a list of k-dimensional vector sets.

vector<vector

Started, last checked	13/1/2010, 13/1/2010
Location	Set operations
Calls	
Called by	insert-retaining-sorted-asc
Comments/see also	

Example:

```
(vector<vector '(4 6 7) '(4 6 7.1))
--> T
```

For $\mathbf{d} = (d_1, d_2, \ldots, d_k)$, $\mathbf{e} = (e_1, e_2, \ldots, e_k)$, we say that $\mathbf{d}$ is less than $\mathbf{e}$, denoted $\mathbf{d} \prec \mathbf{e}$, if and only if there exists an integer $1 \leq j \leq k$ such that $d_j < e_j$, and $d_i = e_i$ for $1 \leq i < j$. This function returns true if its first argument is 'less than' its second argument, "equal" if the two arguments are equal, and nil otherwise.

vector<vector-t-or-nil

Started, last checked	13/1/2010, 13/1/2010
Location	Set operations
Calls	
Called by	difference-list-sorted-asc, translate-pattern-to-1st-occurrence
Comments/see also	

Example:

```
(vector<vector '(4 6 7) '(4 6 7.1))
--> T
```

The function vector¡vector returned 'equal' if the arguments were equal. This function returns nil in such a scenario.

4.1.3 Sort by

These functions culminate in the function sort-by, and were written when I was new to Lisp. As such, there are faster, more robust, in-built alternatives for some of the functions.

break-fixed-with-sort-by-col

Started, last checked	8/8/2008, 8/8/2008
Location	Sort by
Calls	lastn, rows-with-fixed-same-as-1st-row, sort-by-col
Called by	sort-by
Comments/see also	

Example:

```
(break-fixed-with-sort-by-col '(1 2)
  '((3 4 0 0) (3 4 5 2) (0 4 5 -3)
    (-1 4 5 9) (1 3 6 1) (-1 2 7 0))
  3 "desc")
--> '((3 4 0 0) (-1 4 5 9) (3 4 5 2)
      (0 4 5 -3) (1 3 6 1) (-1 2 7 0))
```

This function has as its second argument a list whose items are themselves lists of m items, and it is assumed that this list has already been sorted by certain 'columns' (ascending or descending) specified in the first argument; a list called fixed. In these specified 'columns', any ties which persist between consecutive 'rows' are (potentially) broken, with a sort by the argument col (default direction is ascending).

We see in the example above that a sort has already been conducted by 'columns' one and two, hence the argument fixed is '(1 2). The function breaks the specified ties in the given list with a descending sort by column three.

index-item-1st-doesnt-occur

Started, last checked	8/8/2008, 8/8/2008
Location	Sort by
Calls	
Called by	
Comments/see also	index-item-1st-occurs

Example:

```
(index-item-1st-doesnt-occur 0 '(0 0 0 -2 4 2))
--> 3
```

Taking an item and a list of items as its arguments, this function returns the index at which the given item first does not occur, counting from zero. If the list is constant and equal to the item then the function returns NIL.

index-equalps-for-pair-list

Started, last checked	8/8/2008, 8/8/2008
Location	Sort by
Calls	
Called by	nos-consecutives-with-nonempty-fixed
Comments/see also	

Example:

```
(index-equalps-for-pair-list
 '(1 3 4) '((1 7 9 2 1 1) (0 7 9 9 1 0)))
--> (T NIL T)
```

This function looks for equality (using the function equalp) in a pair of lists at those indices specified by a second variable index.

max-argmax

Started, last checked	8/8/2008, 8/8/2008
Location	Sort by
Calls	
Called by	sort-by-col-desc, sort-items-desc
Comments/see also	min-argmin

Example:

```
(max-argmax '(2 4 -2 7/2 4))
--> (4 1)
```

This function returns the maximum item in a nonempty list (assuming all items are of the same type), as well as the index of that maximum item, counting from zero.

max-item

Started, last checked	8/8/2008, 8/8/2008
Location	Sort by
Calls	
Called by	
Comments/see also	min-item

Example:

```
(max-item '(2/3 -3 15 2))
--> 15
```

This function finds the maximum item in a nonempty list. It assumes all items are of the same type; otherwise nonsense output can be produced.

max-nth

Started, last checked	8/8/2008, 8/8/2008
Location	Sort by
Calls	
Called by	max-nth-argmax
Comments/see also	min-nth

Example:

```
(max-nth 0 '((0 3 10) (1 5 2) (0 4 9)))
--> (1 5 2)
```

This function returns the maximum item in a list where all items are themselves lists of m items. In order to find the maximum therefore, it is necessary to specify the 'column' (counting from zero) by which the search ought to be conducted.

max-nth-argmax

Started, last checked	8/8/2008, 8/8/2008
Location	Sort by
Calls	index-item-1st-occurs, max-nth
Called by	
Comments/see also	min-nth-argmin

Example:

```
(max-nth-argmax 2 '((0 3 2) (0 4 8) (0 5 -2) (9 9 2)))
--> ((0 4 8) 1)
```

Here we return the maximum item in a list where all items are themselves lists of m items, searching by the nth 'column' counting from zero. Also returned is the index of the maximum item.

min-argmin

Started, last checked	8/8/2008, 8/8/2008
Location	Sort by
Calls	index-item-1st-occurs, min-item
Called by	sort-by-col-asc, sort-items-asc
Comments/see also	max-argmax

Example:

```
(min-argmin '(0 4 -2 7/2 -2))
--> (-2 2)
```

This function returns the minimum item in a nonempty list (assuming all items are of the same type), as well as the index of that minimum item, counting from zero.

min-item

Started, last checked	8/8/2008, 8/8/2008
Location	Sort by
Calls	
Called by	min-argmin
Comments/see also	max-item

Example:

```
(min-item '(2/3 -3 15 2))
--> -3
```

This function finds the minimum item in a nonempty list. It assumes all items are of the same type; otherwise nonsense output can be produced.

min-nth

Started, last checked	8/8/2008, 8/8/2008
Location	Sort by
Calls	
Called by	min-nth-argmin
Comments/see also	max-nth

Example:

```
(min-nth 2 '((0 3 10) (1 5 7) (0 4 9)))
--> (1 5 7)
```

This function returns the minimum item in a list where all items are themselves lists of m items. In order to find the minimum therefore, it is necessary to specify the 'column' (counting from zero) by which the search ought to be conducted.

min-nth-argmin

Started, last checked	8/8/2008, 8/8/2008
Location	Sort by
Calls	index-item-1st-occurs, min-nth
Called by	
Comments/see also	max-nth-argmax

Example:

```
(min-nth-argmin 1 '((0 3 2) (0 5 -2) (0 0 8) (9 9 2)))
--> ((0 0 8) 2)
```

Here we return the minimum item in a list where all items are themselves lists of m items, searching by the nth 'column' counting from zero. Also returned is the index of the minimum item.

nos-consecutives-with-fixed

Started, last checked	8/8/2008, 8/8/2008
Location	Sort by
Calls	nos-consecutives-with-nonempty-fixed
Called by	rows-with-fixed-same-as-1st-row
Comments/see also	

Example:

```
(nos-consecutives-with-fixed
 '(1 3)
 '((7 4 0 2 1) (1 4 -1 2 -9) (3 4 4 1 1)
   (2 -5 0 3 -9) (2 4 5 2 -2) (3 4 4 2 1)
   (1 1 1 1 1)))
--> 2
```

Suppose that the items indexed by $I = (i_1, i_2, \ldots, i_m)$ in a list are x_1, $x_2, \ldots$, x_m, and that this is the case for several such lists, appearing as the first n entries in some list of lists. Then this function will return the value n. If index is empty, then the length of the list is returned. This has a bearing on higher-level functions.

nos-consecutives-with-nonempty-fixed

Started, last checked	8/8/2008, 8/8/2008
Location	Sort by
Calls	firstn, index-equalps-for-pair-list, test-all-true
Called by	nos-consecutives-with-fixed
Comments/see also	

Example:

```
(nos-consecutives-with-nonempty-fixed
 '(1 3)
 '((7 4 0 2 1) (1 4 -1 2 -9) (3 4 4 2 1)
   (2 -5 0 2 -9) (2 4 5 2 -2) (3 4 4 2 1)
   (1 1 1 1 1)))
--> 2
```

Here the function assumes a nonempty $I = (i_1, i_2, \ldots, i_m)$, indexing items $x_1, x_2, \ldots, x_m$ in some list. This is supposed to be the case for several such lists, appearing as the first n entries in some list of lists. This function will return the value n. It may seem unnecessary to have both the functions nos-consecutives-with-fixed and nos-consecutives-with-nonempty-fixed, but writing the two as a single function results in a less efficient algorithm.

rows-with-fixed-same-as-1st-row

Started, last checked	8/8/2008, 8/8/2008
Location	Sort by
Calls	firstn, nos-consecutives-with-fixed
Called by	break-fixed-with-sort-by-col
Comments/see also	

Example:

```
(rows-with-fixed-same-as-1st-row
 '(1 3)
 '((7 4 0 2 1) (1 4 -1 2 -9) (3 4 4 1 1)
   (2 -5 0 3 -9) (2 4 5 2 -2) (3 4 4 2 1)
   (1 1 1 1 1)))
--> ((7 4 0 2 1) (1 4 -1 2 -9))
```

Suppose that the i_1th, i_2th,..., i_mth items in a list are $x_1, x_2, \ldots, x_m$, and that this is the case for several such lists appearing as the first n entries in some list of lists. Then this function will return those first n entries.

sort-by

Started, last checked	8/8/2008, 8/8/2008
Location	Sort by
Calls	break-fixed-with-sort-by-col
Called by	
Comments/see also	

Example:

```
(sort-by
 '((5 "asc") (0 "asc") (1 "asc") (3 "desc"))
 '((1000 41 500 1 84 1500) (1000 36 500 1 84 1500)
   (1000 41 500 2 84 1500) (0 60 1000 1 84 1000)
   (2500 61 500 1 84 3000) (3000 62 1000 1 84 4000)
   (1500 60 1500 1 84 3000)))
--> ((0 60 1000 1 84 1000) (1000 36 500 1 84 1500)
     (1000 41 500 2 84 1500) (1000 41 500 1 84 1500)
     (1500 60 1500 1 84 3000) (2500 61 500 1 84 3000)
     (3000 62 1000 1 84 4000))
```

This code sorts a list of items (where each item is itself a list of m items). It does so according to an index (of arbitrary length) consisting of 'column' numbers and the direction in which each column ought to be sorted. If for example, `(0 "asc")` appears before `(3 "desc")` in the index, then the list is sorted first by 'column' 0 ascending. And then any ties which emerge might be broken by sorting among tied sets according to 'column' 3 descending.

sort-by-col

Started, last checked	8/8/2008, 8/8/2008
Location	Sort by
Calls	sort-by-col-asc, sort-by-col-desc
Called by	break-fixed-with-sort-by-col
Comments/see also	

Example:

```
(sort-by-col
 2 '((3 -2 5 0) (4 1 -8 1) (4 1 0 -2) (3 0 0 -1))
 "desc")
--> ((3 -2 5 0) (4 1 0 -2) (3 0 0 -1) (4 1 -8 1))
(sort-by-col
 2 '((3 -2 5 0) (4 1 -8 1) (4 1 0 -2) (3 0 0 -1)))
--> ((4 1 -8 1) (4 1 0 -2) (3 0 0 -1) (3 -2 5 0))
```

This code sorts a list of items (where each item is itself a list of m items) in a specified direction (e.g. 'desc'). If this direction is not provided, the function sorts in an ascending order. It assumes all items are of the same type; otherwise nonsense output can be produced.

sort-by-col-asc

Started, last checked	8/8/2008, 8/8/2008
Location	Sort by
Calls	min-nth-argmin, remove-nth
Called by	sort-by-col
Comments/see also	

Example:

```
(sort-by-col-asc
```

```
 2 '((3 -2 5 0) (4 1 -8 1) (4 1 0 -2) (3 0 0 -1)))
--> ((4 1 -8 1) (4 1 0 -2) (3 0 0 -1) (3 -2 5 0))
```

This function returns a list (where each item is itself a list of m items) which is ordered ascending by a particular 'column'. It assumes all items are of the same type; otherwise nonsense output can be produced.

sort-by-col-desc

Started, last checked	8/8/2008, 8/8/2008
Location	Sort by
Calls	max-nth-argmax, remove-nth
Called by	sort-by-col
Comments/see also	

Example:

```
(sort-by-col-desc
 2 '((3 -2 -5 0) (4 1 8 1) (4 1 0 -2) (3 0 0 -1)))
--> ((4 1 8 1) (4 1 0 -2) (3 0 0 -1) (3 -2 -5 0))
```

This function returns a list (where each item is itself a list of m items) which is ordered descending by a particular 'column'. It assumes all items are of the same type; otherwise nonsense output can be produced.

sort-items

Started, last checked	8/8/2008, 8/8/2008
Location	Sort by
Calls	
Called by	
Comments/see also	Non-destructive use of built-in function called sort.

Example:

```
(sort-items '(8 2 5 0 -6 2) "desc")
--> (8 5 2 2 0 -6)
(sort-items '(8 2 5 0 -6 2))
--> (-6 0 2 2 5 8)
```

This code sorts a list of items (non- destructively) in a specified direction (e.g. 'desc'). If this direction is not provided, the function sorts in an ascending order. It assumes all items are of the same type; otherwise nonsense output can be produced.

sort-items-asc

Started, last checked	8/8/2008, 8/8/2008
Location	Sort by
Calls	
Called by	
Comments/see also	Deprecated.

Example:

```
(sort-items-asc '(0 2 4 6 -1 2))
--> (-1 0 2 2 4 6)
```

This code sorts a list of items (non- destructively) in ascending order. It assumes all items are of the same type; otherwise nonsense output can be produced.

sort-items-desc

Started, last checked	8/8/2008, 8/8/2008
Location	Sort by
Calls	
Called by	
Comments/see also	Deprecated.

Example:

```
(sort-items-desc '(0 2 4 6 -1 2))
--> (6 4 2 2 0 -1)
```

This code sorts a list of items (non- destructively) in descending order. It assumes all items are of the same type; otherwise nonsense output can be produced.

test-all-true

Started, last checked	8/8/2008, 8/8/2008
Location	Sort by
Calls	
Called by	nos-consecutives-with-nonempty-fixed
Comments/see also	

Example:

```
(test-all-true '(T NIL T T NIL))
--> NIL
```

This code tests whether all of the items in a list (of Ts and NILs) are in fact Ts.

4.1.4 Vector operations

These functions allow common vector operations, such as taking norms, calculating dot products and distance functions.

dot-product

Started, last checked	9/1/2015, 9/1/2015
Location	Vector operations
Calls	4.1.4, 4.1.4
Called by	
Comments/see also	

Example:

```
(dot-product '(0 1 2) '(3 -1 4))
--> 7.
```

The dot product of two lists is returned. If $x = (x_1, x_2, \ldots, x_n)$ and $y = (y_1, y_2, \ldots, y_n)$, then the output is $\sum_{i=1}^{n} x_i y_i$. Passing lists of different lengths may lead to errors.

fibonacci-list

Started, last checked	29/1/2010, 29/1/2010
Location	Vector operations
Calls	
Called by	
Comments/see also	

Example:

```
(fibonacci-list '(0 1 2 4 8))
-->(0 1 3 7 15)
```

The nth element of the list returned is the sum of the previous $n-1$ elements, with the convention that a sum over an empty set is zero.

max-matrix

Started, last checked	29/1/2010, 29/1/2010
Location	Vector operations
Calls	
Called by	establishment-matrix
Comments/see also	

Example:

```
(max-matrix '((4 0 -3) (-2 3 5) (0 0 0) (0 -1 3)))
--> 5
```

This function returns the maximum element in a matrix represented as a list of lists.

min-matrix

Started, last checked	29/1/2010, 29/1/2010
Location	Vector operations
Calls	
Called by	establishment-matrix
Comments/see also	

Example:

```
(max-matrix '((4 0 -3) (-2 3 5) (0 0 0) (0 -1 3)))
--> 5
```

This function returns the maximum element in a matrix represented as a list of lists.

multiply-two-lists

Started, last checked	29/1/2010, 29/1/2010
Location	Vector operations
Calls	
Called by	
Comments/see also	

Example:

```
(multiply-two-lists '(4 7 -3) '(8 -2 -3))
--> (32 -14 9)
```

Multiplies two lists element-by-element. It is assumed that elements of list arguments are numbers, and the list arguments are of the same length. An empty first (but not second) argument will be tolerated.

normalise-0-1

Started, last checked	29/1/2010, 29/1/2010
Location	Vector operations
Calls	normalise-0-1-checks-done
Called by	
Comments/see also	

Example:

```
(normalise-0-1 '(4 7 -3 2))
--> (7/10 1 0 1/2)
```

Normalises data (linearly) to $[0, 1]$.

normalise-0-1-checks-done

Started, last checked	29/1/2010, 29/1/2010
Location	Vector operations
Calls	
Called by	normalise-0-1
Comments/see also	

Example:

```
(normalise-0-1-checks-done '(4 7 -3 2))
--> (7/10 1 0 1/2)
```

Normalises data (linearly) to $[0, 1]$, assuming that the data is not constant and that the min and max are not already 0, 1 respectively.

replace-nth-in-list-with-x

Started, last checked	23/6/2013, 23/6/2013
Location	Vector operations
Calls	
Called by	sky-line-clipped
Comments/see also	substitute-index-by-index-abs-x

Example:

```
(replace-nth-in-list-with-x 3 '(0 52 55 0.5 1) 5.4)
--> (0 52 55 5.4 1)
```

This function replaces the nth item of a list with whatever is supplied as the third variable. Passing a value for n less than zero or greater than m, where m is one less than the length of the list, will result in an error.

4.1.5 Stats sampling

The functions below are for finding summary statistics, and for taking random samples from data.

choose-one

Started, last checked	6/10/2010, 6/10/2010
Location	Stats sampling
Calls	
Called by	
Comments/see also	

Example:

```
(choose-one '(1 2 4))
--> 4
```

A random, equiprobable choice is made between elements of a list.

cor

Started, last checked	6/10/2010, 6/10/2010
Location	Stats sampling
Calls	fibonacci-list, my-last
Called by	key-correlations
Comments/see also	

Example:

```
(cor '(6 7 4) '(6 7 4))
--> 1.0
(cor '(6 7 4) '(-6 -7 -4))
--> -1.0
(cor '(6 7 4) '(0 2 1.5))
--> 0.05
```

The sample Pearson correlation coefficient is returned for two input lists, which are assumed to be of equal length.

frequency-count

Started, last checked	8/3/2013, 8/3/2013
Location	Stats sampling
Calls	sort-dataset-asc
Called by	most-frequent-difference-vector, structure-induction-algorithm-r
Comments/see also	

Example:

```
(frequency-count
 '((5 4) (3 2) (3 2.000001) (0 1)) t)
--> (((0 1) 1) ((3 2.000001) 2) ((5 4) 1))
```

The frequency of occurrence of a list member in a list of lists is returned. It is possible to specify use of equality up to an error tolerance (given by the variable *equality-tolerance*).

histogram

Started, last checked	6/10/2010, 6/10/2010
Location	Stats sampling
Calls	add-to-nth, constant-vector, index-1st-sublist-item>=
Called by	matching-score-histogram
Comments/see also	

Example:

```
(setq a-list '(2 4 -1 6 9 12 0 -7 5 3 1 2 3 8 3 1 -5))
(setq edges '(-7.5 -3.5 0.5 4.5 8.5 12.5))
(histogram a-list edges)
--> (2 2 8 3 2)
```

A list of scalar data is the input to this function, along with a list of edges, assumed to be in ascending order. The output is a list of length one less than the number of edges. Item i of the output list gives the number of data d that satisfy $e(i-1) < d \leq e(i)$.

mean

Started, last checked	6/10/2010, 6/10/2010
Location	Stats sampling
Calls	fibonacci-list, my-last
Called by	median
Comments/see also	

Example:

```
(mean '(6 7 4))
--> 17/3
```

The mean of a list of numbers is returned.

median

Started, last checked	6/10/2010, 6/10/2010
Location	Stats sampling
Calls	mean
Called by	quartiles
Comments/see also	

Example:

```
(setq
 a-list
 '(0 9 0 4 0 29 82 21 28 4 17 78 33 8 8 8 17 20 4 12))
(median a-list)
--> 21/2
```

The median of a list of numbers is returned.

quartiles

Started, last checked	6/10/2010, 6/10/2010
Location	Stats sampling
Calls	median
Called by	
Comments/see also	

Example:

```
(setq
 a-list
 '(0 9 0 4 0 29 82 21 28 4 17 78 33 8 8 8 17 20 4))
(quartiles a-list)
--> (4 9 28)
```

The lower, median, and upper quartiles of a list of numbers are returned.

random-permutation

Started, last checked	6/10/2010, 6/10/2010
Location	Stats sampling
Calls	nth-list, sample-integers-no-replacement
Called by	
Comments/see also	

Example:

```
(random-permutation '("A" "B" "C" "D" "E"))
--> ("C" "A" "E" "D" "B")
```

The output of this function is a random permutation of an input list.

range

Started, last checked	6/10/2010, 6/10/2010
Location	Stats sampling
Calls	max-item, min-item
Called by	pitch-range
Comments/see also	

Example:

```
(range '(60 61 62))
--> 2
```

Range is the maximum member of a list, minus the minimum member.

sample-integers-no-replacement

Started, last checked	6/10/2010, 6/10/2010
Location	Stats sampling
Calls	add-to-list, choose-one first-n-naturals
Called by	random-permutation
Comments/see also	

Example:

```
(sample-integers-no-replacement 10 7)
--> (5 4 8 2 0 3 9)
```

The first argument to this function, n, is an integer, as is the second $m \leq n$. The output is a random sample (without replacement) from the integers $0, \ldots, n-1$ of size m. If $m > n$, we set $m = n$.

sd

Started, last checked	6/10/2010, 6/10/2010
Location	Stats sampling
Calls	fibonacci-list, mean, my-last
Called by	rhythmic-variability
Comments/see also	

Example:

```
(sd '(64 55 65 55 72 55 55 55 60 59 67))
--> 5.7178855
```

The standard deviation of the sample (using a denominator of n, where n is the sample size).

4.1.6 Geometric operations

The functions below are for finding summary statistics, and for taking random samples from data.

convex-hull

Started, last checked	11/5/2010, 11/5/2010
Location	Geometric operations
Calls	
Called by	
Comments/see also	

Example:

```
(setq a-list '((-4 4) (-2 -2) (-2 2) (0 0) (1 1)
               (2 -2) (2 4) (6 2)))
(convex-hull a-list)
--> ((-2 -2) (2 -2) (6 2) (2 4) (-4 4))
```

For a set of points in the plane, this function returns those points that lie on the convex hull, using the Graham scan algorithm. Passing an empty set of points to this function will result in an error.

counter-clockwisep

Started, last checked	11/5/2010, 11/5/2010
Location	Geometric operations
Calls	
Called by	convex-hull
Comments/see also	

Example:

```
(counter-clockwisep '((-2 -2) (2 -2) (1 1)))
--> -1
```

This function takes three points in the plane as its argument, p_1, p_2, and p_3, arranged in a single list. If travelling along the line from p_1 to p_2, turning next to p_3 means turning counter-clockwise, then 1 is the value returned. If clockwise then -1 is returned, and if collinear then 0 is returned.

dot-adjacent-points

Started, last checked	11/5/2010, 11/5/2010
Location	Geometric operations
Calls	
Called by	in-polygonp
Comments/see also	

Example:

```
(dot-adjacent-points
 '((-1 -3) (1 1) (-2 -1) (-1 -3)))
--> (-4 -3 5)
```

This function takes adjacent pairs from the argument list and computes their dot product.

in-polygonp

Started, last checked	11/5/2010, 11/5/2010
Location	Geometric operations
Calls	dot-adjacent-points, fibonacci-list, multiply-list-by-constant, multiply-two-lists, my-last, quadrant-number, signum-adjacent-determinants, spacing-items, substitute-index-by-index-abs-x, translation
Called by	points-in-convex-hull
Comments/see also	

Example:

```
(setq closed-vertices
      '((-2 -2) (2 -2) (6 2) (2 4) (-4 4) (-2 -2)))
(in-polygonp '(1 1) closed-vertices)
--> T
```

A point in the plane and a list of closed, adjacent vertices are supplied as arguments. T is returned if the point is inside or on the polygon, and nil otherwise.

min-y-coord

Started, last checked	11/5/2010, 11/5/2010
Location	Geometric operations
Calls	
Called by	convex-hull
Comments/see also	

Example:

```
(min-y-coord '((-4 4) (-2 -2) (-2 2) (2 -2) (6 2)))
--> (-2 -2)
```

This function returns the point with the minimum y-coordinate, where the argument is assumed to be in the form $((x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n))$. Ties are broken using the x-coordinate.

points-in-convex-hull

Started, last checked	11/5/2010, 11/5/2010
Location	Geometric operations
Calls	
Called by	
Comments/see also	

Example:

```
(points-in-convex-hull
 '((-1.71 -1.13) (1.27 -3.95) (3.66 -2.05)
   (-2.65 -3.48) (1.4 -2.94) (1.53 0.51) (-2.67 0.32))
 '((-1.33 0.3) (-1.3 -4.0) (0.83 1.41) (1.83 2.89)
   (1.85 -0.94) (2.22 -2.93) (2.34 2.81) (2.4 -0.15)
   (2.49 -2.71)))
--> ((-1.33 0.3) (1.85 -0.94) (2.22 -2.93)
     (2.49 -2.71))
```

This function takes two sets of points in the plane as its arguments. The convex hull is found for the first set. It is then determined for each member of the second set whether or not that member is inside (or on) the convex hull or not. The points in the convex hull are returned. There is a plot for the above example in the *Example files* folder, entitled *convex-hull.pdf*.

quadrant-number

Started, last checked	11/5/2010, 11/5/2010
Location	Geometric operations
Calls	
Called by	in-polygonp
Comments/see also	

Example:

```
(quadrant-number '(-4 4))
--> 2
```

This function returns the quadrant number of the plane point (x, y) supplied as argument.

signum-adjacent-determinants

Started, last checked	11/5/2010, 11/5/2010
Location	Geometric operations
Calls	
Called by	in-polygonp
Comments/see also	

Example:

```
(signum-adjacent-determinants
 '((-1 -3) (1 1) (-2 -1) (-1 -3)))
--> (1 1 1)
```

This function takes adjacent pairs from the argument list and computes the sign of the determinant, as though the pairs were in a 2×2 matrix.

spacing-items

Started, last checked	11/5/2010, 11/5/2010
Location	Geometric operations
Calls	
Called by	in-polygonp
Comments/see also	

Example:

```
(spacing-items '(0 12 1 7 4))
--> '(12 -11 6 -3)
```

A list of numbers is the only argument. The intervals between adjacent numbers are returned. It is possible to produce nonsense output if null values are interspersed with non-null values.

substitute-index-by-index-abs-x

Started, last checked	11/5/2010, 11/5/2010
Location	Geometric operations
Calls	
Called by	in-polygonp
Comments/see also	replace-nth-in-list-with-x

Example:

```
(substitute-index-by-index-abs-x
 '(-4 4 -2 2 6 2) '(3 5 10 12 7 13) 2)
--> (-4 4 10 12 6 13)
```

This function is very specific. When the absolute value of the ith item of the first argument is equal to the third argument, that item is replaced in the output with the ith item of the second argument.

4.1.7 Interpolation

These functions are for interpolating step functions given by pairs of x- and y-values at specified values.

abs-differences-for-curves-at-points

Started, last checked	1/10/2010, 1/10/2010
Location	Interpolation
Calls	linearly-interpolate
Called by	
Comments/see also	

Example:

```
(setq knot-value-pairs1 '((0 0) (1.5 1) (2 3) (4 2)))
(setq knot-value-pairs2 '((0 0) (1 1) (2 3) (4 3)))
(abs-differences-for-curves-at-points
 knot-value-pairs1 knot-value-pairs2)
--> (0 1.0 0 1)
```

Two lists of knot-value pairs are provided as arguments. The x-values of the first argument are interpolated using the second argument. The absolute difference between these interpolated values and the actual y-values of the first argument is returned.

index-1st-sublist-item<

Started, last checked	1/10/2010, 1/10/2010
Location	Interpolation
Calls	
Called by	
Comments/see also	index-nth-sublist-item<, index-1st-sublist-item<=

Example:

```
(index-1st-sublist-item<
 0 '(14 14 14 11 0 0 -1 -2 -2))
--> 6
```

This function takes two arguments: a real number x and a list L of real numbers. It returns the index of the first element of L which is less than x.

index-1st-sublist-item>

Started, last checked	1/10/2010, 1/10/2010
Location	Interpolation
Calls	
Called by	linearly-interpolate
Comments/see also	index-nth-sublist-item>, index-1st-sublist-item>=

Example:

```
(index-1st-sublist-item>
 0 '(-2 -2 -1 0 0 11 14 14 14))
--> 5
```

This function takes two arguments: a real number x and a list L of real numbers. It returns the index of the first element of L which is greater than x.

linearly-interpolate

Started, last checked	1/10/2010, 1/10/2010
Location	Interpolation
Calls	index-1st-sublist-item>, my-last, nth-list-of-lists
Called by	abs-differences-for-curves-at-points, linearly-interpolate-x-values
Comments/see also	

Example:

```
(setq knot-value-pairs '((0 0) (1 1) (2 3) (4 3)))
(linearly-interpolate 1.5 knot-value-pairs)
--> 2.0
```

The second argument is a list of knot-value pairs. The x-value of the first argument is interpolated to give a y-value using the knot-value pairs. If the first argument exceeds the endpoints, the appropriate endpoint value is returned.

linearly-interpolate-x-values

Started, last checked	1/10/2010, 1/10/2010
Location	Interpolation
Calls	linearly-interpolate
Called by	
Comments/see also	

Example:

```
(setq knot-value-pairs '((0 0) (1 1) (2 3) (4 3)))
(linearly-interpolate-x-values
 '(1.5 2 1.75) knot-value-pairs)
--> (2.0 3 2.5)
```

The second argument is a list of knot-value pairs. The first argument is a list of x-values that are interpolated to give y-values using the knot- value pairs. If any members of the first argument exceeds the endpoints, the appropriate endpoint value is returned.

4.1.8 Merge sort operations

These functions implement various merge sorts and related operations.

vector<vector-different-lengths

Started, last checked	19/1/2010, 19/1/2010
Location	List processing
Calls	vector<vector-t-or-nil
Called by	
Comments/see also	

Example:

```
(vector<vector-different-lengths '(0 -2) '(0 -2 3))
--> T
```

If the vectors are of the same length, the function vector<vector-t-or-nil is applied. If the second vector $\mathbf{v}_2 = (v_{21}, \ldots, v_{2n})$ is longer than the first $\mathbf{v}_1 = (v_{11}, \ldots, v_{1m})$, we must allow for the possibility of equality $\mathbf{v}_1 = \mathbf{v}_2$, before applying the function vector<vector. When equal as described, $\mathbf{v}_1 < \mathbf{v}_2$.

4.1.9 Locating indices

Functions here are for finding indices of lists whose members satisfy certain requirements.

Functions such as index-1st-sublist-item<= ought to be moved here eventually as well.

index-nth-sublist-item<

Started, last checked	14/9/2013, 14/9/2013
Location	Locating indices
Calls	
Called by	
Comments/see also	index-1st-sublist-item<, index-nth-sublist-item<=

Example:

```
(index-nth-sublist-item<
 0 3 '(14 14 14 11 0 0 -1 -2 -2))
```

```
--> 8
```

This function takes three arguments: a real number x, an integer counter n, and a list L of real numbers. It returns the index of the nth element of L which is less than x, where $n = 1$ refers to the first element.

index-nth-sublist-item>

Started, last checked	14/9/2013, 14/9/2013
Location	Locating indices
Calls	
Called by	
Comments/see also	index-1st-sublist-item>, index-nth-sublist-item>=

Example:

```
(index-nth-sublist-item>
 4 2 '(0 0 0 1 1 4 6 6 7 7 11 14 14 14))
--> 7
```

This function takes three arguments: a real number x, an integer counter n, and a list L of real numbers. It returns the index of the nth element of L which is greater than x, where $n = 1$ refers to the first element.

index-nth-sublist-item<=

Started, last checked	14/9/2013, 14/9/2013
Location	Locating indices
Calls	
Called by	
Comments/see also	index-1st-sublist-item<=, index-nth-sublist-item<

Example:

```
(index-nth-sublist-item<=
 6 3 '(14 14 14 11 7 7 6 6 4 1 1 0 0))
--> 8
```

This function takes three arguments: a real number x, an integer counter n, and a list L of real numbers. It returns the index of the nth element of L which is less than or equal to x, where $n = 1$ refers to the first element.

index-nth-sublist-item>=

Started, last checked	14/9/2013, 14/9/2013
Location	Locating indices
Calls	
Called by	
Comments/see also	index-1st-sublist-item>=, index-nth-sublist-item>

Example:

```
(index-nth-sublist-item>=
 4 2 '(0 0 0 1 1 4 6 6 7 7 11 14 14 14))
--> 6
```

This function takes three arguments: a real number x, an integer counter n, and a list L of real numbers. It returns the index of the nth element of L which is greater than or equal to x, where $n = 1$ refers to the first element.

4.2 File conversion

4.2.1 Text files

The functions below will export a list to a text file, and import such text into the Lisp environment as lists.

frac2dec

Started, last checked	15/1/2013, 15/1/2013
Location	Text files
Calls	
Called by	
Comments/see also	

Example:

```
(frac2dec '(1 3.4 ("no" 4/3 "yeah")))
--> (1 3.4 ("no" 1.3333334 "yeah")).
```

This function converts fractions occurring in a list (of arbitrary depth) into floats.

pathname-typesp

Started, last checked	15/1/2013, 15/1/2013
Location	Text files
Calls	
Called by	
Comments/see also	

Example:

```
(pathname-typesp
 #P"/Users/hello.txt" (list "csv" "txt"))
--> T
```

This function checks whether the path (including file name and type) supplied as the first argument is of one of the types specified by the second argument.

positions-char

Started, last checked	15/6/2014, 15/6/2014
Location	Text files
Calls	
Called by	pitch-class-sequential-expression2list
Comments/see also	

Example:

```
(positions-char #\_ "ascending _ _ _")
--> (10 12 14)
```

This function returns the indices in a string where instances of the character argument occur.

read-from-file

Started, last checked	15/1/2010, 15/1/2010
Location	Text files
Calls	
Called by	
Comments/see also	

Example:

```
(read-from-file
 (concatenate
  'string
  *MCStylistic-Oct2010-example-files-path*
  "/short-list.txt"))
--> ((9 23 1 19) (14 9 14 5 20 25) (16 5 18 3 5 14 20)
     ("sure" 9 4) (13 9 19 8 5 1 18 4) (8 5 18))
```

This function returns the contents of a file specified by the variable path&name. It returns each row of the file as a list, in a list of lists.

read-from-file-arbitrary

Started, last checked	15/1/2010, 15/1/2010
Location	Text files
Calls	
Called by	
Comments/see also	

Example:

```
(read-from-file-arbitrary
 (concatenate
  'string
  *MCStylistic-Oct2010-example-files-path*
  "/short-list 2.txt"))
--> ("first line consisting of anything"
     "sure 9 4" "second line consisting of &^%$")
```

This function is similar to the function read-from-file. The difference is that read-from-file- arbitrary will parse any file, converting each line to a string for further processing.

replace-all

Started, last checked	1/5/2014, 1/5/2014
Location	Text files
Calls	
Called by	harmonic-interval-of-a, pitch-class-time-intervals, word&event-time-intervals
Comments/see also	

Example:

```
(replace-all
 "all the occurences of the part" "the" "THE")
--> "all THE occurences of THE part"
```

This function, from the Common Lisp Cookbook, returns a new string in which all the occurences of the part is replaced with replacement.

replace-once

Started, last checked	10/6/2015, 10/6/2015
Location	Text files
Calls	
Called by	number-string2numberless-string
Comments/see also	

Example:

```
(replace-once "16 16th notes" "16" "17")
--> "17 16th notes"
(replace-once "16 16th notes" "16" "")
--> "16th notes"
```

This function replaces the first instance (from the left) of its second argument in the first argument by the third argument.

update-written-file

Started, last checked	13/3/2013, 13/3/2013
Location	Text files
Calls	read-from-file, write-to-file
Called by	SIA-reflected
Comments/see also	

Example:

```
(update-written-file
 (merge-pathnames
  (make-pathname
   :name "list-to-update" :type "txt")
  *MCStylistic-Aug2013-example-files-data-path*)
```

```
 1 '(6 60) '((2 2) . ((2 72))))
--> (((0 7) . ((0 60) (2 63)))
     ((2 2) . ((2 72) (6 60)))
     ((3 -1) . ((0 60) (3 67)))
     ((6 0) . ((3 67) (5 66))))
whereas originally file read
    (((0 7) . ((0 60) (2 63)))
     ((2 2) . ((2 72)))
     ((3 -1) . ((0 60) (3 67)))
     ((6 0) . ((3 67) (5 66))))
```

This function updates the contents of a specifed file by removing the row associated with the variable updatee, and replacing it with updater appended within updatee. It should overwrite the existing file. The position of the row is preserved.

write-to-file

Started, last checked	15/1/2010, 2/1/2015
Location	Text files
Calls	
Called by	update-written-file
Comments/see also	

Example:

```
(write-to-file
 '(5 7 8 "hello" 9)
 (concatenate
  'string
  *MCStylistic-Oct2010-example-files-path*
  "/short-list 4.txt"))
--> T
```

This function writes the data provided in the first list to a file with the path and name provided in the second list. The s in the format argument is essential for retaining strings as they appear in the data.

2/1/2015. Added an optional argument to prevent closing the file after the end of writing.

write-to-file-append

Started, last checked	15/1/2010, 15/1/2010
Location	Text files
Calls	
Called by	
Comments/see also	

Example:

```
(write-to-file-append
 '(10 "goodbye")
 (concatenate
  'string
  *MCStylistic-Oct2010-example-files-path*
  "/short-list 4.txt"))
--> T
```

The only difference between this and the function write-to-file is that an existing file will be opened and new data appended, rather than overwritten.

write-to-file-supersede

Started, last checked	15/1/2010, 15/1/2010
Location	Text files
Calls	write-to-file
Called by	
Comments/see also	

Example:

```
(write-to-file
 '(5 7 8 "hello" 9)
 (concatenate
  'string
  *MCStylistic-Oct2010-example-files-path*
  "/short-list 5.txt"))
--> T
(write-to-file-supersede
 '(10 "goodbye")
 (concatenate
```

```
  'string
  *MCStylistic-Oct2010-example-files-path*
  "/short-list 5.txt"))
--> T
```

The only difference between this and the function write-to-file is that an existing file will be superseded, rather than overwritten.

write-to-file-with-open-file

Started, last checked	15/1/2010, 15/1/2010
Location	Text files
Calls	
Called by	
Comments/see also	

Example:

```
(write-to-file-with-open-file
 "hello"
 (concatenate
  'string
  *MCStylistic-Oct2010-example-files-path*
  "/New folder/short-list.txt"))
--> T
```

There was a problem with the function write- to-file in Emacs, because it would not export to a directory that did not already exist. This was remedied using the functions with-open-file and ensure-directories-exist. However, this function only works with a single (i.e. non-list) variable. Once you have used it to create the directory, use the function write-to-file as per usual.

4.2.2 MIDI import

The main function here is load-midi-file, for importing a MIDI (Musical Instrument Digital Interface) file into the Lisp environment as a list of lists.

add-tempo

Started, last checked	26/1/2009, 26/1/2009
Location	MIDI import
Calls	
Called by	
Comments/see also	Deprecated.

Example:

```
(add-tempo '(5012 5012 5012))
--> 2
```

As tempo and granularity are needed to convert ticks to ms, this function is invoked when the number 81 is parsed (which indicates a tempo change). The format of each entry here is (time-in-ticks time-in-ms usec/qn). Storing the time of the tempo change in both formats simplifies the calculations.

convert-granularity

Started, last checked	26/1/2009, 26/1/2009
Location	MIDI import
Calls	
Called by	get-header
Comments/see also	

Example: within get-header example,

```
(convert-granularity (get-short input-stream))
--> 120
```

The treatment of division is unusual. Granularity is in ticks per beat (crotchet).

convert-vlq

Started, last checked	26/1/2009, 26/1/2009
Location	MIDI import
Calls	
Called by	set-track-time
Comments/see also	get-vlq

Example:

```
(setq
 fstring
 (concatenate
  'string *MCStylistic-Oct2010-example-files-path*
  "/vivaldi-op6-no3-2.mid"))
(with-open-file
    (input-stream
     fstring :element-type '(unsigned-byte 8)
     :if-does-not-exist nil)
  (convert-vlq (get-vlq input-stream)))
--> 77
```

This function converts an integer represented using variable-length quantity (VLQ) into the digit representation. In MIDI files, time is listed as ticks in VLQs.

earlier

Started, last checked	26/1/2009, 26/1/2009
Location	MIDI import
Calls	
Called by	load-midi-file
Comments/see also	

Example:

```
(earlier '(5 60) '(6.5 61))
--> T
```

This function returns T if the first element of the first argument is less than the first element of the second argument, and NIL otherwise.

first>=

Started, last checked	26/1/2009, 26/1/2009
Location	MIDI import
Calls	
Called by	ticks-ms
Comments/see also	

Example:

```
(first>= 60 '(59 64 67))
--> T
```

This is a test function for tempo searches.

gather-bytes

Started, last checked	26/1/2009, 26/1/2009
Location	MIDI import
Calls	
Called by	get-metadata, read-track
Comments/see also	

Example:

```
(with-open-file
    (s
     (concatenate
      'string
      *MCStylistic-Oct2010-example-files-path*
      "/temp-bytes")
     :direction :output :element-type 'unsigned-byte)
  (write-byte 101 s) (write-byte 111 s))
--> 111
(with-open-file
    (s
     (concatenate
      'string
      *MCStylistic-Oct2010-example-files-path*
      "/temp-bytes") :element-type 'unsigned-byte)
  (gather-bytes s 2))
--> (101 111)
```

This function reads arbitrary bytes into a list.

get-header

Started, last checked	26/1/2009, 26/1/2009
Location	MIDI import
Calls	
Called by	get-metadata, read-track
Comments/see also	

Example:

```
(setq
 fstring
 (concatenate
  'string *MCStylistic-Oct2010-example-files-path*
  "/vivaldi-op6-no3-2.mid"))
(with-open-file
    (input-stream
     fstring :element-type '(unsigned-byte 8)
     :if-does-not-exist nil)
  (setup)
  (get-header input-stream))
--> 120
```

This function reads the file header.

get-metadata

Started, last checked	26/1/2009, 26/1/2009
Location	MIDI import
Calls	gather-bytes
Called by	parse-events, read-track
Comments/see also	

Example:

```
(setq
 fstring
 (concatenate
  'string *MCStylistic-Oct2010-example-files-path*
  "/vivaldi-op6-no3-2.mid"))
(with-open-file
    (input-stream
     fstring :element-type '(unsigned-byte 8)
     :if-does-not-exist nil)
  (get-metadata input-stream))
--> (84 104 100 0 0 0 6 0 1 0 12 0 120 77 84 114 107 0
     0 0 12 0 255 81 3 15 66 64 196 56 255 47 0 77 84
     114 107 0 0 1 196 0 192 6 0 144 49 64 0 255 3 11
     104 97 114 112 115 105 99 104 111 114 100 0 255 4
     11 104 97 114 112 115 105 99 104 111 114)
```

This function reads a length, then gathers that many bytes together (representing metadata).

get-short

Started, last checked	26/1/2009, 26/1/2009
Location	MIDI import
Calls	
Called by	get-header
Comments/see also	get-word

Example: within get-header example,

```
(get-short input-stream)
--> 1
```

This function is a 16-bit retriever.

get-track-header

Started, last checked	26/1/2009, 26/1/2009
Location	MIDI import
Calls	get-type, get-word
Called by	read-track
Comments/see also	

Example:

```
(setq
 fstring
 (concatenate
  'string *MCStylistic-Oct2010-example-files-path*
  "/vivaldi-op6-no3-2.mid"))
(with-open-file
    (input-stream
     fstring :element-type '(unsigned-byte 8)
     :if-does-not-exist nil)
  (setup)
  (get-header input-stream)
  (get-track-header input-stream))
--> 12
```

This function reads a track header.

get-type

Started, last checked	26/1/2009, 26/1/2009
Location	MIDI import
Calls	
Called by	get-header, get-track-header
Comments/see also	

Example: within get-header example,

```
(get-type input-stream)
--> "MThd"
```

Helps to read header.

get-word

Started, last checked	26/1/2009, 26/1/2009
Location	MIDI import
Calls	
Called by	get-header, get-track-header
Comments/see also	get-short

Example: within get-header example,

```
(get-word input-stream)
--> 6
```

This function is a 32-bit retriever.

get-vlq

Started, last checked	26/1/2009, 26/1/2009
Location	MIDI import
Calls	
Called by	set-track-time
Comments/see also	convert-vlq

Example:

```
(setq
 fstring
```

```
 (concatenate
  'string *MCStylistic-Oct2010-example-files-path*
  "/vivaldi-op6-no3-2.mid"))
(with-open-file
    (input-stream
     fstring :element-type '(unsigned-byte 8)
     :if-does-not-exist nil)
  (get-vlq input-stream))
--> (77)
```

All events are seperated by a delta time, so this function gets the VLQ.

handle-bend

Started, last checked	26/1/2009, 26/1/2009
Location	MIDI import
Calls	
Called by	parse-events
Comments/see also	

Example:

```
(handle-bend #XA0 60 3)
--> (160 60 3)
```

This function discards bend data.

handle-control

Started, last checked	26/1/2009, 26/1/2009
Location	MIDI import
Calls	
Called by	parse-events
Comments/see also	

Example:

```
(handle-control #XA0 60 84)
--> (160 60 84)
```

This function discards control data.

handle-note

Started, last checked	26/1/2009, 26/1/2009
Location	MIDI import
Calls	ticks-ms
Called by	parse-events
Comments/see also	

Example:

```
(handle-note #XA0 60 84)
--> 0
```

This function parses a note-on event.

handle-off

Started, last checked	26/1/2009, 26/1/2009
Location	MIDI import
Calls	match-note, ticks-ms
Called by	parse-events
Comments/see also	

Example:

```
(handle-off #XA0 60 84)
--> 0
```

This function searches for the note-on event to which a note-off event belongs and sets the duration accordingly. This does not handle overlapping notes of the same pitch.

handle-pressure

Started, last checked	26/1/2009, 26/1/2009
Location	MIDI import
Calls	
Called by	parse-events
Comments/see also	

Example:

```
(handle-pressure #XA0 60)
--> (160 60)
```

This function discards pressure data.

handle-program

Started, last checked	26/1/2009, 26/1/2009
Location	MIDI import
Calls	
Called by	parse-events
Comments/see also	

Example:

```
(handle-program #XA0 60)
--> (160 60)
```

This function discards program data.

handle-touch

Started, last checked	26/1/2009, 26/1/2009
Location	MIDI import
Calls	
Called by	parse-events
Comments/see also	

Example:

```
(handle-touch #XA0 60 84)
--> (160 60 84)
```

This function discards touch data.

list-to-string

Started, last checked	26/1/2009, 26/1/2009
Location	MIDI import
Calls	
Called by	parse-metadata
Comments/see also	

Example:

```
(list-to-string '(84 104 111 109 97 115))
--> "Thomas."
```

This function converts ASCII to strings. Note that most metadata is text.

load-midi-file

Started, last checked	26/1/2009, 26/1/2009
Location	MIDI import
Calls	earlier, get-header, read-track, setup
Called by	
Comments/see also	

Example:

```
(setq
 imported-midi
 (load-midi-file
  (concatenate
   'string *MCStylistic-Oct2010-example-files-path*
   "/vivaldi-op6-no3-2.mid")))
(subseq imported-midi 0 10)
--> ((0 79 1 6 64) (0 69 1 4 64) (0 49 1 1 64)
     (0 49 1 3 64) (0 76 1 5 64) (1 79 1 6 64)
     (1 69 1 4 64) (1 76 1 5 64) (1 49 1 1 64)
     (1 49 1 3 64))
```

This main function imports a MIDI file into the Lisp environment.

match-note

Started, last checked	26/1/2009, 26/1/2009
Location	MIDI import
Calls	
Called by	handle-off
Comments/see also	

Example:

```
(match-note '(#XA0 62) '(4 62 0 #XA0))
--> T
```

This function tests for a note-off event.

parse-events

Started, last checked	26/1/2009, 26/1/2009
Location	MIDI import
Calls	handle-bend, handle-control, handle-note, handle-off, handle-pressure, handle-program, handle-touch, parse-metadata, strip-sysex
Called by	read-and-parse-event
Comments/see also	

Example:

```
(setq
 fstring
 (concatenate
  'string *MCStylistic-Oct2010-example-files-path*
  "/vivaldi-op6-no3-2.mid"))
(with-open-file
    (input-stream
     fstring :element-type '(unsigned-byte 8)
     :if-does-not-exist nil)
  (parse-events #XA0 60 input-stream))
--> (160 60 77)
```

This function handles track data.

parse-metadata

Started, last checked	26/1/2009, 26/1/2009
Location	MIDI import
Calls	list-to-string
Called by	parse-events
Comments/see also	

Example:

```
(parse-metadata '(9 104))
--> 1
```

This function extracts tempo and end-of- track information from the metadata.

re-time

Started, last checked	26/1/2009, 26/1/2009
Location	MIDI import
Calls	my-last
Called by	
Comments/see also	Deprecated.

Example:

```
(re-time
 '((1 69 1 4 64) (0 79 1 6 64) (2 49 1 3 64)
   (0 69 1 4 64))
 '((1 69 1 4 64) (0 79 1 6 64) (2 49 1 3 64)
   (0 69 1 4 64)))
--> ((1 69 1 4 64) (0 79 1 6 64) (2 49 1 3 64)
     (0 69 1 4 64) (2 69 1 4 64) (1 79 1 6 64)
     (3 49 1 3 64) (1 69 1 4 64))
```

This function appends some events (the second argument) to other events (the first argument) by translating them by the ontime of the last event from the first argument.

read-and-parse-event

Started, last checked	26/1/2009, 26/1/2009
Location	MIDI import
Calls	parse-events
Called by	read-track
Comments/see also	

Example:

```
(setq
 fstring
 (concatenate
  'string *MCStylistic-Oct2010-example-files-path*
  "/vivaldi-op6-no3-2.mid"))
(with-open-file
    (input-stream
     fstring :element-type '(unsigned-byte 8)
     :if-does-not-exist nil)
```

```
  (read-and-parse-event input-stream))
--> NIL
```

This function deals with running status.

read-track

Started, last checked	26/1/2009, 26/1/2009
Location	MIDI import
Calls	gather-bytes, get-track-header, read-and-parse-event, set-track-time
Called by	load-midi-file
Comments/see also	

Example:

```
(setq
 fstring
 (concatenate
  'string *MCStylistic-Oct2010-example-files-path*
  "/vivaldi-op6-no3-2.mid"))
(with-open-file
    (input-stream
     fstring :element-type '(unsigned-byte 8)
     :if-does-not-exist nil)
  (setup)
  (get-header input-stream)
  (read-track input-stream))
--> NIL
```

This function is called once per track, and reads track data.

set-track-time

Started, last checked	26/1/2009, 26/1/2009
Location	MIDI import
Calls	convert-vlq, get-vlq
Called by	read-track
Comments/see also	

Example:

```
(setq
 fstring
 (concatenate
  'string *MCStylistic-Oct2010-example-files-path*
  "/vivaldi-op6-no3-2.mid"))
(with-open-file
    (input-stream
     fstring :element-type '(unsigned-byte 8)
     :if-does-not-exist nil)
  (set-track-time input-stream))
--> 77
```

There are times between events, so *track-time* must be accumulated across each track.

setup

Started, last checked	26/1/2009, 26/1/2009
Location	MIDI import
Calls	
Called by	load-midi-file
Comments/see also	Consider giving more specific name.

Example:

```
(setup)
--> #()
```

This function sets the values of three variables.

strip-sysex

Started, last checked	26/1/2009, 26/1/2009
Location	MIDI import
Calls	
Called by	parse-events
Comments/see also	

Example:

```
(setq
```

```
 fstring
 (concatenate
  'string *MCStylistic-Oct2010-example-files-path*
  "/vivaldi-op6-no3-2.mid"))
(with-open-file
    (input-stream
     fstring :element-type '(unsigned-byte 8)
     :if-does-not-exist nil)
  (strip-sysex input-stream))
--> "Error: Unexpected end of file"
```

This function deletes sysex data. The example gives an error because the imported MIDI file does not contain any sysex.

ticks-ms

Started, last checked	26/1/2009, 26/1/2009
Location	MIDI import
Calls	
Called by	handle-note, handle-off
Comments/see also	

Example:

```
(ticks-ms 50)
--> 5/12
```

The time conversion function searches the tempo map from the end to find tempo in effect at the time.

4.2.3 MIDI export

The functions below are for exporting datapoints (dimensions for ontime in milliseconds, MIDI note number, duration in milliseconds, channel, and velocity) to a MIDI file. The main function is saveit.

convert-ontime-to-deltatime

Started, last checked	27/1/2009, 27/1/2009
Location	MIDI export
Calls	
Called by	create-midi-events
Comments/see also	

Example:

```
(convert-ontime-to-deltatime 500)
--> 24
    0
```

This function converts an ontime to a deltatime.

create-midi-events

Started, last checked	27/1/2009, 15/7/2013
Location	MIDI export
Calls	convert-ontime-to-deltatime, make-midi-note-msg, make-midi-pc-msg
Called by	create-MTrk
Comments/see also	

Example:

```
(create-midi-events
 '((0 36 1000 1 35) (0 48 1000 1 35) (500 60 500 7 40)
   (0 1 0 1 255)))
--> ((0 (144 36 35)) (48 (128 36 35)) (0 (144 48 35))
     (48 (128 48 35)) (24 (150 60 40))
     (48 (134 60 40)) (0 (192 0)))
```

This function converts datapoints into the format required for the function create-midi-track-data.

create-midi-file

Started, last checked	27/1/2009, 27/1/2009
Location	MIDI export
Calls	
Called by	save-as-midi, saveit
Comments/see also	

Example:

```
(setq
 outfilename
 (concatenate
```

```
  'string *MCStylistic-Oct2010-example-files-path*
  "/midi-export-test.mid"))
(create-midi-file outfilename)
--> #<BASIC-FILE-BINARY-OUTPUT-STREAM
    ((concatenate
      'string *MCStylistic-Oct2010-example-files-path*
      "/midi-export-test.mid")/13 ISO-8859-1)
    #x3000419BC3FD>
```

This function creates an output stream.

create-midi-track-data

Started, last checked	27/1/2009, 15/7/2013
Location	MIDI export
Calls	make-var-len
Called by	create-MTrk
Comments/see also	

Example:

```
(create-midi-track-data '((0 (193 1)) (2 (193 1))))
--> (0 193 1 2 193 1)
```

Each element of the variable midiEvents is of the format ((deltaTime (byte3 byte2 byte1)). They should be sorted in the order for the file and their deltaTimes are relative to each other (each relative to the previous). This function creates an integer-stream representation.

create-midi-tracks

Started, last checked	27/1/2009, 27/1/2009
Location	MIDI export
Calls	create-tempo-track, create-MTrk
Called by	save-as-midi, saveit
Comments/see also	

Example:

```
(create-midi-tracks
 '((0 1 0 1 255) (0 2 0 2 255) (0 3 0 3 255)
   (0 4 0 4 255) (0 5 0 5 255) (0 6 0 6 255)
```

```
   (0 7 0 7 255) (0 8 0 8 255) (0 9 0 9 255)
   (0 10 0 10 255) (0 11 0 11 255) (0 12 0 12 255)
   (0 13 0 13 255) (0 14 0 14 255) (0 15 0 15 255)
   (0 16 0 16 255) (0 60 1000 1 127)))
--> ((77 84 114 107 0 0 0 4 0 255 47 0)
     (77 84 114 107 0 0 0 15 0 192 0 0 144 60 127 48
      128 60 127 0 255 47 0)
     (77 84 114 107 0 0 0 7 0 193 1 0 255 47 0)
     (77 84 114 107 0 0 0 7 0 194 2 0 255 47 0)
     (77 84 114 107 0 0 0 7 0 195 3 0 255 47 0)
     (77 84 114 107 0 0 0 7 0 196 4 0 255 47 0)
     (77 84 114 107 0 0 0 7 0 197 5 0 255 47 0)
     (77 84 114 107 0 0 0 7 0 198 6 0 255 47 0)
     (77 84 114 107 0 0 0 7 0 199 7 0 255 47 0)
     (77 84 114 107 0 0 0 7 0 200 8 0 255 47 0)
     (77 84 114 107 0 0 0 7 0 201 9 0 255 47 0)
     (77 84 114 107 0 0 0 7 0 202 10 0 255 47 0)
     (77 84 114 107 0 0 0 7 0 203 11 0 255 47 0)
     (77 84 114 107 0 0 0 7 0 204 12 0 255 47 0)
     (77 84 114 107 0 0 0 7 0 205 13 0 255 47 0)
     (77 84 114 107 0 0 0 7 0 206 14 0 255 47 0)
     (77 84 114 107 0 0 0 7 0 207 15 0 255 47 0))
```

This function takes datapoints and lists representing the ends of tracks, and converts them into the integer streams in preparation for the function write-to-midi-file.

create-MThd

Started, last checked	27/1/2009, 27/1/2009
Location	MIDI export
Calls	
Called by	save-as-midi, saveit
Comments/see also	

Example:

```
(create-MThd 17)
--> (77 84 104 100 0 0 0 6 0 1 0 17 0 48)
```

This function creates the integer stream representing a MIDI file's header data.

create-MTrk

Started, last checked	27/1/2009, 27/1/2009
Location	MIDI export
Calls	create-midi-events, create-midi-track-data, fix-deltatime, sort-by-deltatime, split-bytes
Called by	create-midi-tracks
Comments/see also	

Example:

```
(create-MTrk '((0 1 0 1 255) (0 60 1000 1 127)))
--> (77 84 104 100 0 0 0 6 0 1 0 17 0 48)
```

This function creates the integer stream representing one track in a MIDI file.

create-tempo-track

Started, last checked	27/1/2009, 27/1/2009
Location	MIDI export
Calls	split-bytes
Called by	create-midi-tracks
Comments/see also	

Example:

```
(create-tempo-track)
--> ((77 84 114 107 0 0 0 4 0 255 47 0))
```

This function creates the integer representation for a MIDI file's tempo track.

fix-deltatime

Started, last checked	27/1/2009, 15/7/2013
Location	MIDI export
Calls	
Called by	create-MTrk
Comments/see also	

Example:

```
(fix-deltatime 40 '((0 (207 15))))
--> ((-40 (207 15)))
```

This function shifts all of the deltatimes back by the first argument.

get-channel-events

Started, last checked	27/1/2009, 27/1/2009
Location	MIDI export
Calls	
Called by	create-midi-tracks
Comments/see also	

Example:

```
(get-channel-events
 1
 '((0 1 0 1 255) (0 2 0 2 255) (0 3 0 3 255)
   (0 4 0 4 255) (0 5 0 5 255) (0 6 0 6 255)
   (0 7 0 7 255) (0 8 0 8 255) (0 9 0 9 255)
   (0 10 0 10 255) (0 11 0 11 255) (0 12 0 12 255)
   (0 13 0 13 255) (0 14 0 14 255) (0 15 0 15 255)
   (0 16 0 16 255) (0 60 1000 1 127)))
--> ((0 1 0 1 255) (0 60 1000 1 127))
```

This function takes an integer between 1 and 16 (inclusive) as its first argument, and a list of datapoints as its second argument. It returns elements of this list whose fourth element is equal to the first argument.

get-byte

Started, last checked	27/1/2009, 27/1/2009
Location	MIDI export
Calls	
Called by	split-bytes
Comments/see also	

Example:

```
(get-byte 3 7)
--> 0
```

This function converts an integer to bytes, starting at the rightmost index.

insert-program-changes

Started, last checked	27/1/2009, 27/1/2009
Location	MIDI export
Calls	
Called by	save-as-midi, saveit
Comments/see also	

Example:

```
(insert-program-changes '((0 60 1000 1 127)))
--> ((0 1 0 1 255) (0 2 0 2 255) (0 3 0 3 255)
     (0 4 0 4 255) (0 5 0 5 255) (0 6 0 6 255)
     (0 7 0 7 255) (0 8 0 8 255) (0 9 0 9 255)
     (0 10 0 10 255) (0 11 0 11 255) (0 12 0 12 255)
     (0 13 0 13 255) (0 14 0 14 255) (0 15 0 15 255)
     (0 16 0 16 255) (0 60 1000 1 127))
```

This function inserts MIDI track headers as datapoints (signified by 255).

make-midi-note-msg

Started, last checked	27/1/2009, 27/1/2009
Location	MIDI export
Calls	
Called by	create-midi-events
Comments/see also	

Example:

```
(make-midi-note-msg '(0 60 1000 1 127) 144)
--> (144 60 127)
```

This function creates MIDI note messages of the type processed by the function create-midi-track-data. It should be pointed out that #x90 is for a note-on and #x80 for a note-off.

make-midi-pc-msg

Started, last checked	27/1/2009, 27/1/2009
Location	MIDI export
Calls	
Called by	create-midi-events
Comments/see also	

Example:

```
(make-midi-pc-msg '(0 16 0 16 255))
--> (207 15)
```

This function creates MIDI PC messages of the type processed by the function create-midi-track-data. It should be pointed out that #xC0 is for a program change.

make-var-len

Started, last checked	27/1/2009, 15/7/2013
Location	MIDI export
Calls	
Called by	create-midi-track-data
Comments/see also	

Example:

```
(make-var-len 1241)
--> (137 89)
```

This function converts integers to a binary- like representation. Adapted from http://www.blitter.com/∼russtopia/MIDI/∼jglatt/.

save-as-midi

Started, last checked	27/1/2009, 27/1/2009
Location	MIDI export
Calls	create-midi-file, create-midi-tracks, create-MThd, insert-program-changes, write-to-midi-file
Called by	
Comments/see also	saveit

Example:

```
(save-as-midi
 "midi-export-test.mid"
 '((0 36 1000 1 35) (0 48 1000 1 35) (500 60 500 7 40)
   (1000 63 500 7 45) (1500 67 500 7 50)
   (2000 72 500 7 55) (2500 75 500 7 60)
   (3000 79 500 12 65) (3500 84 500 12 70)
   (4000 79 500 12 75) (4500 75 500 12 80)
   (5000 72 500 12 84) (5500 67 500 12 84)))
--> (concatenate
     'string
     *MCStylistic-Oct2010-example-files-path*
     "/midi-export-test.mid")
```

This function exports datapoints (dimensions for ontime in milliseconds, MIDI note number, duration in milliseconds, channel, and velocity) to a MIDI file. It can clip the end of the file, so to avoid this, use the function saveit.

saveit

Started, last checked	27/1/2009, 27/1/2009
Location	MIDI export
Calls	create-midi-file, create-midi-tracks, create-MThd, insert-program-changes, write-to-midi-file
Called by	
Comments/see also	save-as-midi

Example:

```
(saveit
 (concatenate
  'string
  *MCStylistic-Oct2010-example-files-path*
  "/midi-export-test.mid")
 '((0 36 1000 1 35) (0 48 1000 1 35) (500 60 500 7 40)
   (1000 63 500 7 45) (1500 67 500 7 50)
   (2000 72 500 7 55) (2500 75 500 7 60)
   (3000 79 500 12 65) (3500 84 500 12 70)
   (4000 79 500 12 75) (4500 75 500 12 80)
```

```
   (5000 72 500 12 84) (5500 67 500 12 84)))
--> (concatenate
     'string
     *MCStylistic-Oct2010-example-files-path*
     "/midi-export-test.mid")
```

This function is very similar to the function save-as-midi (exporting datapoints with dimensions for ontime in milliseconds, MIDI note number, duration in milliseconds, channel, and velocity, to a MIDI file). The difference is that this function does not clip the end of the file.

sort-by-deltatime

Started, last checked	27/1/2009, 27/1/2009
Location	MIDI export
Calls	
Called by	create-MTrk
Comments/see also	

Example:

```
(sort-by-deltatime '((24 (207 15)) (0 (207 15))))
--> ((0 (207 15)) (24 (207 15)))
```

This function sorts a list of lists ascending by the car of each list.

split-bytes

Started, last checked	27/1/2009, 27/1/2009
Location	MIDI export
Calls	get-byte
Called by	create-MTrk, create-tempo-track
Comments/see also	

Example:

```
(split-bytes 7 4)
--> (0 0 0 7)
```

This function splits a long integer into its high byte and low byte.

write-to-midi-file

Started, last checked	27/1/2009, 27/1/2009
Location	MIDI export
Calls	
Called by	save-as-midi, saveit
Comments/see also	

Example:

```
(setq
 outfilename
 (concatenate
  'string *MCStylistic-Oct2010-example-files-path*
  "/midi-export-test.mid"))
(write-to-midi-file
 (create-midi-file outfilename)
 '((77 84 104 100 0 0 0 6 0 1 0 17 0 48)
   (77 84 114 107 0 0 0 4 0 255 47 0)
   (77 84 114 107 0 0 0 15 0 192 0 0 144 60 127 48 128
    60 127 0 255 47 0)
   (77 84 114 107 0 0 0 7 0 193 1 0 255 47 0)
   (77 84 114 107 0 0 0 7 0 194 2 0 255 47 0)))
--> NIL
```

This function will convert MIDI track events to bytes and write them to a specified file.

4.2.4 Hash tables

The functions below are for saving, reading, displaying and querying hash-tables. It can be convenient to work with lists, each of whose elements is a hash table.

copy-hash-table

Started, last checked	25/1/2010, 25/1/2010
Location	Hash tables
Calls	
Called by	disp-ht-el, disp-ht-key
Comments/see also	

Example:

```
(setq A (make-hash-table :test #'equal))
(setf (gethash '"hair colour" A) "brown")
(setq B (copy-hash-table A))
(list A B)
--> (#<HASH-TABLE
        :TEST EQUAL size 1/60 #x300041A694FD>
     #<HASH-TABLE
        :TEST EQUAL size 1/60 #x300041A683DD>)
```

This function returns a copy of hash table, with the same keys and values. The copy has the same properties as the original, unless overridden by the keyword arguments.

Before each of the original values is set into the new hash-table, key is invoked on the value. As key defaults to cl:identity, a shallow copy is returned by default. Adapted from http://common-lisp.net/project/alexandria/darcs /alexandria/hash-tables.lisp.

disp-ht-el

Started, last checked	25/1/2010, 25/1/2010
Location	Hash tables
Calls	
Called by	write-to-file-balanced-hash-table
Comments/see also	

Example:

```
(setq A (make-list-of-hash-tables 2))
(setf (gethash '"hair colour" (first A)) "brown")
(setf (gethash '"eye colour" (first A)) "brown")
(setf (gethash '"hair colour" (second A)) "blond")
(setf (gethash '"gender" (second A)) "male")
(disp-ht-el (first A))
--> (("hair colour" . "brown")
     ("eye colour" . "brown"))
```

This function displays the contents of a hash table.

disp-ht-key

Started, last checked	25/1/2010, 25/1/2010
Location	Hash tables
Calls	
Called by	write-to-file-balanced-hash-table
Comments/see also	

Example:

```
(setq A (make-hash-table :test #'equalp))
(setf (gethash '"hair colour" A) "brown")
(setf (gethash '"eye colour" A) "brown")
(setf (gethash '"gender" A) "male")
(disp-ht-key A)
--> ("hair colour" "eye colour" "gender")
```

This function displays the keys of a hash table.

hash-tables-with-key-value-pairs

Started, last checked	25/1/2010, 25/1/2010
Location	Hash tables
Calls	constant-vector
Called by	
Comments/see also	

Example:

```
(setq A (make-list-of-hash-tables 2))
(setf (gethash '"hair colour" (first A)) "brown")
(setf (gethash '"eye colour" (first A)) "brown")
(setf (gethash '"name" (first A)) "Chris")
(setf (gethash '"hair colour" (second A)) "blond")
(setf (gethash '"gender" (second A)) "male")
(setf (gethash '"name" (second A)) "Chris")
(setq
 B
 (hash-tables-with-key-value-pairs
  A '(("name" . "Chris") ("hair colour" . "brown"))))
--> (#<HASH-TABLE
     :TEST EQUAL size 3/60 #x3000417DF6FD>)
```

This function returns those hash tables in a list that have key-value pairs equalp to those specified in the second argument. The example returns a list of one hash table because only the first hash table contains information for somebody called Chris with brown hair.

index-target-translation-in-hash-tables

Started, last checked	25/1/2010, 25/1/2010
Location	Hash tables
Calls	test-translation
Called by	
Comments/see also	

Example:

```
(setq
 A
 (read-from-file-balanced-hash-table
  (concatenate
   'string
   *MCStylistic-Oct2010-example-files-path*
   "/patterns-hash.txt")))
(disp-ht-el (fourth A))
(index-target-translation-in-hash-tables
 '((4 38 47) (9/2 38 47) (5 38 47)) A "pattern")
--> 3
```

The hash tables each contain a value specified by the third argument (a key). We think of these as patterns, and want to know if any of the patterns are translations of the first argument, the target. The index of the first extant translation is returned, and nil otherwise.

index-target-translation-mod-in-hash-tables

Started, last checked	25/1/2010, 25/1/2010
Location	Hash tables
Calls	test-translation-mod-2nd-n
Called by	number-of-targets-trans-mod-in-hash-tables
Comments/see also	

Example:

```
(setq
 A
 (read-from-file-balanced-hash-table
  (concatenate
   'string
   *MCStylistic-Oct2010-example-files-path*
   "/patterns-hash.txt")))
(disp-ht-el (fourth A))
(index-target-translation-mod-in-hash-tables
 '((0 36 46) (1/2 48 46) (1 36 46)) A 12 "pattern")
--> 3
```

This function is very similar to the function index-target-translation-in-hash-tables, except that in the second dimension translations are carried out modulo the third argument.

make-list-of-hash-tables

Started, last checked	25/1/2010, 25/1/2010
Location	Hash tables
Calls	
Called by	read-from-file-balanced-hash-table
Comments/see also	

Example:

```
(make-list-of-hash-tables 3)
--> (#<HASH-TABLE
        :TEST EQUAL size 0/60 #x300041BB8A5D>
     #<HASH-TABLE
        :TEST EQUAL size 0/60 #x300041BB84AD>
     #<HASH-TABLE
        :TEST EQUAL size 0/60 #x300041BB7EFD>)
```

This function returns a list, each of whose elements is an empty hash table of type ‘equal’.

number-of-targets-trans-mod-in-hash-tables

Started, last checked	25/1/2010, 25/1/2010
Location	Hash tables
Calls	index-target-translation-mod-in-hash-tables
Called by	
Comments/see also	

Example:

```
(setq
 A
 (read-from-file-balanced-hash-table
  (concatenate
   'string
   *MCStylistic-Oct2010-example-files-path*
   "/patterns-hash.txt")))
(number-of-targets-trans-mod-in-hash-tables
 '(((0 36 46) (1/2 60 46) (1 36 46))
   ((0 60 46) (0 48 53) (0 55 57) (0 60 60) (0 64 62)
    (1/2 36 46) (1/2 48 53) (1/2 55 57) (1/2 62 61)
    (1/2 65 63) (1 36 46) (1 48 53) (1 55 57)
    (1 64 62) (1 67 64) (2 48 53) (2 65 63) (2 69 65)
    (3 36 46) (3 48 53) (3 55 57) (3 64 62) (3 67 64)
    (7/2 36 46) (7/2 48 53) (7/2 55 57) (7/2 60 60)
    (7/2 64 62) (4 36 46) (4 48 53) (4 55 57)))
 A 12 "pattern")
--> 2
```

This function is very similar to the function number-of-targets-translation-in-hash-tables, except that in the second dimension translation is performed modulo the third argument.

number-of-targets-translation-in-hash-tables

Started, last checked	25/1/2010, 25/1/2010
Location	Hash tables
Calls	test-target-translation-in-hash-tables
Called by	
Comments/see also	

Example:

```
(setq
 A
 (read-from-file-balanced-hash-table
  (concatenate
   'string
   *MCStylistic-Oct2010-example-files-path*
   "/patterns-hash.txt")))
(number-of-targets-translation-in-hash-tables
 '(((6 48 53) (13/2 48 53) (7 48 53))
   ((24 29 42) (24 41 49) (24 48 53) (24 53 56)
    (24 57 58) (49/2 29 42) (49/2 41 49) (49/2 48 53)
    (49/2 55 57) (49/2 58 59) (25 29 42) (25 41 49)
    (25 48 53) (25 57 58) (25 60 60) (26 41 49)
    (26 58 59) (26 62 61) (27 29 42) (27 41 49)
    (27 48 53) (27 57 58) (27 60 60) (55/2 29 42)
    (55/2 41 49) (55/2 48 53) (55/2 53 56)
    (55/2 57 58) (28 29 42) (28 41 49) (28 48 53)))
 A "pattern")
--> 2
```

The function test-target-translation-in- hash-tables is applied recursively to each member of the first argument of this function. This argument is a list of targets. Each time a translation of a target is detected, the output (initially set to zero) is incremented by one.

re-index-list-of-hash-tables

Started, last checked	25/1/2010, 25/1/2010
Location	Hash tables
Calls	
Called by	
Comments/see also	

Example:

```
(setq
 A
 (read-from-file-balanced-hash-table
  (concatenate
   'string
   *MCStylistic-Oct2010-example-files-path*
```

```
   "/small-hash-table.txt")))
(setq A (re-index-list-of-hash-tables A 780))
(gethash '"index" (first A))
--> 780
    T
```

This function re-indexes a list of hash tables beginning from an optional second argument.

read-from-file-balanced-hash-table

Started, last checked	25/1/2010, 25/1/2010
Location	Hash tables
Calls	make-list-of-hash-tables, read-from-file
Called by	
Comments/see also	

Example:

```
(setq
 A
 (read-from-file-balanced-hash-table
  (concatenate
   'string
   *MCStylistic-Oct2010-example-files-path*
   "/small-hash-table.txt")))
(disp-ht-el (second A))
--> (("height" . "6ft") ("name" . "Justin")
     ("index" . 77))
```

This function reads a balanced list of hash tables that have been written to a file, by the function write-to-file-balanced-hash-table. It is assumed that the hash tables are homogeneous or balanced in the sense that they contain exactly the same keys.

set-each-hash-table-element

Started, last checked	25/1/2010, 25/1/2010
Location	Hash tables
Calls	
Called by	
Comments/see also	

Example:

```
(setq A (make-list-of-hash-tables 2))
(setf (gethash '"hair colour" (first A)) "brown")
(setf (gethash '"eye colour" (first A)) "brown")
(setf (gethash '"hair colour" (second A)) "blond")
(setf (gethash '"gender" (second A)) "male")
(set-each-hash-table-element A "height" "tall")
(list
 (gethash '"height" (first A))
 (gethash '"height" (second A)))
--> ("tall" "tall")
```

This function is useful if you have a list of hash tables and you want to set each hash table to have an identical key-value pair.

test-target-translation-in-hash-tables

Started, last checked	25/1/2010, 25/1/2010
Location	Hash tables
Calls	test-translation
Called by	number-of-targets-translation-in-hash-tables
Comments/see also	Implementation could call index-target-translation-in-hash-tables instead.

Example:

```
(setq
 A
 (read-from-file-balanced-hash-table
  (concatenate
   'string
   *MCStylistic-Oct2010-example-files-path*
   "/patterns-hash.txt")))
(disp-ht-el (fourth A))
(test-target-translation-in-hash-tables
 '((6 48 53) (13/2 48 53) (7 48 53)) A "pattern")
--> T
```

The hash tables each contain a value specified by the third argument (a key). We think of these as patterns, and want to know if any of the patterns are translations of the first argument, the target. T is returned if a translation does exist among the hash tables, and nil otherwise.

write-to-file-balanced-hash-table

Started, last checked	25/1/2010, 25/1/2010
Location	Hash tables
Calls	disp-ht-el, write-to-file
Called by	
Comments/see also	

Example:

```
(setq A (make-list-of-hash-tables 2))
(setf (gethash '"hair colour" (first A)) "brown")
(setf (gethash '"eye colour" (first A)) "brown")
(setf (gethash '"height" (first A)) 187)
(setf (gethash '"hair colour" (second A)) "blond")
(setf (gethash '"gender" (second A)) "male")
(write-to-file-balanced-hash-table
 A
 (concatenate
  'string
  *MCStylistic-Oct2010-example-files-path*
  "/small-hash-table 2.txt"))
--> T
```

This function takes as its first argument a list, each of whose elements is a hash table. It applies the function disp-ht-el to each element, collects the output and writes it to a text file. At the top of the file are two integers n and m, referring to the length of the list and the number of elements in a hash table respectively. It is assumed that the hash tables are homogeneous or balanced in the sense that they contain exactly the same keys, but as the example demostrates, this does not have to be the case. (Balanced hash tables are easier to read back in.)

4.2.5 CSV files

These functions enable the conversion of csv files into lists of lists, and vice versa. Despite using terms like dataset below, neither representation has to be balanced, that is, rows/lists can contain different numbers of elements.

comma-positions

Started, last checked	30/7/2010, 30/7/2010
Location	CSV files
Calls	
Called by	comma-separated-string2list
Comments/see also	space-bar-positions, tab-positions

Example:

```
(comma-positions "uh,ktr,3,4")
--> (2 6 8)
```

This function returns the positions at which commas occur in a string.

comma-separated-integers2list

Started, last checked	30/7/2010, 30/7/2010
Location	CSV files
Calls	comma-separated-string2list
Called by	
Comments/see also	

Example:

```
(comma-separated-integers2list "1.00, 3.00")
--> (1 3)
```

This function applies the function parse-integer recursively, once the string supplied as an argument has had the function comma-separated-string2list applied.

comma-separated-reals2list

Started, last checked	30/7/2010, 30/7/2010
Location	CSV files
Calls	comma-separated-string2list
Called by	csv2dataset
Comments/see also	space-bar-separated-string2list, tab-separated-string2list, tab-separated-reals2list

Example:

```
(comma-separated-reals2list "1.50, 3/4, -5.6")
--> (1.5 3/4 -5.6)
```

This function applies the function read- from-string recursively, once the string supplied as an argument has had the function comma-separated- string2list applied.

comma-separated-string2list

Started, last checked	30/7/2010, 30/7/2010
Location	CSV files
Calls	comma-positions
Called by	comma-separated-integers2list
Comments/see also	space-bar-separated-string2list, tab-separated-string2list

Example:

```
(comma-separated-string2list "uh,ktr,3,4")
--> ("uh" "ktr" "3" "4")
```

This function turns a comma-separated string into a list, where formerly each item was preceded or proceeded by a comma.

csv2dataset

Started, last checked	30/7/2010, 30/7/2010
Location	CSV files
Calls	comma-separated-integers2list, read-from-file-arbitrary
Called by	comma-separated-integers2list
Comments/see also	tab2dataset

Example:

```
(csv2dataset
 (merge-pathnames
  (make-pathname
   :name "short-list" :type "csv")
  *MCStylistic-Aug2013-example-files-data-path*))
--> ((1 3) (2 6.1) (5 2 2) (6 2))
```

This function converts a file in comma- separated-value (csv) format to a dataset. It will handle reals, strings, and characters.

dataset2csv

Started, last checked	30/7/2010, 30/7/2010
Location	CSV files
Calls	read-from-file
Called by	
Comments/see also	list-of-lists2csv

Example:

```
(dataset2csv
 (merge-pathnames
  (make-pathname
   :name "scarlatti-L10-bars1-19" :type "txt")
  *MCStylistic-Aug2013-example-files-data-path*)
 (merge-pathnames
  (make-pathname
   :name "scarlatti-L10-bars1-19" :type "csv")
  *MCStylistic-Aug2013-example-files-data-path*))
-->  0.00, 36.00, 46.00, 1.00, 1.00
     0.00, 48.00, 53.00, 1.00, 1.00
     0.50, 60.00, 60.00, 0.50, 0.00
    ...
    56.50, 74.00, 68.00, 0.50, 0.00
```

This function converts a dataset (a list of lists of equal length) to a csv file. The first argument is either the path where the dataset resides or the dataset itself.

list-of-lists2csv

Started, last checked	30/7/2010, 2/1/2015
Location	CSV files
Calls	read-from-file
Called by	comma-separated-integers2list
Comments/see also	dataset2csv

Example:

```
(setq
 fpath&name
 (merge-pathnames
  (make-pathname
   :name "list2csv-test" :type "csv")
  *MCStylistic-Aug2013-example-files-results-path*))
(list-of-lists2csv
 '((4 2.0) (0 -2 1) ("A" "B") (4 3) ("W") (0 0 0))
 fpath&name)
--> T
```

This function converts a (possibly unbalanced) list of lists to a csv file. An unbalanced list will also work. The first argument is either the path where the dataset resides or the dataset itself.

2/1/2015. Altered ∼a to ∼s in this function, to aid lossless conversion when re-importing.

string-positions

Started, last checked	4/6/2014, 4/6/2014
Location	CSV files
Calls	
Called by	string-separated-string2list
Comments/see also	space-bar-positions, tab-positions

Example:

```
(string-positions "and" "yes and maybe no and May")
--> (4 17)
```

This function returns the positions at which the first specified substring occurs in the second (longer) string.

string-separated-string2list

Started, last checked	4/6/2014, 4/6/2014
Location	CSV files
Calls	string-positions
Called by	followed-by-splitter
Comments/see also	**??**, tab-separated-string2list

Example:

```
(string-separated-string2list
 "and" "yes and maybe no and May")
--> ("yes" "maybe no" "May")
```

This function turns a tab-separated string into a list, where formerly each item was preceded or proceeded by a tab.

tab-separated-reals2list

Started, last checked	4/9/2013, 4/9/2013
Location	CSV files
Calls	tab-separated-string2list
Called by	tab2dataset
Comments/see also	comma-separated-reals2list

Example:

```
(tab-separated-reals2list "1.50 3/4 -5.6")
--> (1.5 3/4 -5.6)
```

This function applies the function read- from-string recursively, once the string supplied as an argument has had the function tab-separated- string2list applied. It did have problems parsing elements consisting of a dot and no other alpha-numeric characters. A fix was found using string-trim, removing white space and new-line commands before parsing.

tab2dataset

Started, last checked	4/9/2013, 3/1/2014
Location	CSV files
Calls	tab-separated-reals2list
Called by	
Comments/see also	csv2dataset

Example:

```
(tab2dataset
 (merge-pathnames
  (make-pathname
   :name "short-tab-list" :type "txt")
  *MCStylistic-MonthYear-example-files-data-path*))
--> ((0.74 64) (-0.74 76) (1.44 -60))
```

This function converts a file in tab- separated format to a dataset. It will handle reals, strings, and characters. It did have problems parsing elements consisting of a dot and no other alpha-numeric characters. A fix was found using string-trim, removing white space and new-line commands before parsing.

4.2.6 Director musices

Director musices is a music file format. It is not as common as kern or musicXML, but it uses a list format, which makes it amenable to Lisp import using only a few functions. As far as I can tell, the director musices format does not handle multiple voices on one stave. Some of the functions here, like MIDI-morphetic-pair2pitch&octave, are called by music-import and export functions for different formats.

check-pitch&octave

Started, last checked	17/11/2009, 17/11/2009
Location	Director musices
Calls	
Called by	
Comments/see also	

Example:

```
(check-pitch&octave "C3")
--> "C3"
```

This function tests whether a supplied pitch&octave is in an acceptable format and range. I was intending to allow pitches from C0 to C8 (MNNs 12 to 108) but this function will not allow C8, so could be adjusted in future. If acceptable the pitch&octave is returned, and nil otherwise.

director-musice2datapoint

Started, last checked	17/11/2009, 17/11/2009
Location	Director musices
Calls	pitch&octave2MIDI-morphetic-pair
Called by	director-musices2dataset-chunked
Comments/see also	

Example:

```
(director-musice2datapoint
 7 1
 '(bar 1 n ("C3" 1/2) key "C" modus "maj" mm 192
   meter (2 2))
 3 "C3" 1/2)
--> (7 48 53 1/2 1 nil)
```

This function converts one line of a director musices file into a datapoint consisting of ontime, MIDI note number, morphetic pitch number, duration, stave, and T if the note is tied over.

director-musices2dataset

Started, last checked	17/11/2009, 17/11/2009
Location	Director musices
Calls	director-musices2dataset-chunked, resolve-ties
Called by	
Comments/see also	

Example:

```
(director-musices2dataset
 (concatenate
  'string
  *MCStylistic-Oct2010-example-files-path*
  "/scarlatti-L1-bars11-13.dm"))
--> ((0 79 71 1 1) (0 86 75 7/3 0) (1 43 50 1 1)
     (2 38 47 2 1) (7/3 84 74 1/3 0) (8/3 83 73 1/3 0)
     (3 81 72 1/3 0) (10/3 83 73 1/3 0)
     (11/3 84 74 1/3 0) (4 79 71 1 1) (4 83 73 1/3 0)
     (13/3 84 74 1/3 0) (14/3 86 75 1/3 0)
     (5 43 50 1 1) (5 86 75 4/3 0) (6 38 47 2 1)
     (19/3 84 74 1/3 0) (20/3 83 73 1/3 0)
     (7 81 72 1/3 0) (22/3 83 73 1/3 0)
     (23/3 84 74 1/3 0) (8 79 71 1 1) (8 83 73 1/3 0))
```

This function converts a piece of music represented in the director-musices format into a dataset where each datapoint consists of an ontime, MIDI note number, morphetic pitch number, duration and stave.

director-musices2dataset-chunked

Started, last checked	17/11/2009, 17/11/2009
Location	Director musices
Calls	director-musice2datapoint, read-from-file
Called by	director-musices2dataset
Comments/see also	

Example:

```
(director-musices2dataset-chunked
 (concatenate
  'string
  *MCStylistic-Oct2010-example-files-path*
  "/scarlatti-L1-bars11-13.dm"))
--> ((0 86 75 2 0 T) (2 86 75 1/3 0 NIL)
     (7/3 84 74 1/3 0 NIL) (8/3 83 73 1/3 0 NIL)
     (3 81 72 1/3 0 NIL) (10/3 83 73 1/3 0 NIL)
     (11/3 84 74 1/3 0 NIL) (4 83 73 1/3 0 NIL)
     (13/3 84 74 1/3 0 NIL) (14/3 86 75 1/3 0 NIL)
     (5 86 75 1 0 T) (6 86 75 1/3 0 NIL)
     (19/3 84 74 1/3 0 NIL) (20/3 83 73 1/3 0 NIL)
     (7 81 72 1/3 0 NIL) (22/3 83 73 1/3 0 NIL)
     (23/3 84 74 1/3 0 NIL) (8 83 73 1/3 0 NIL)
     (0 79 71 1 1 NIL) (1 43 50 1 1 NIL)
     (2 38 47 2 1 NIL) (4 79 71 1 1 NIL)
     (5 43 50 1 1 NIL) (6 38 47 2 1 NIL)
     (8 79 71 1 1 NIL))
```

This function converts a piece of music represented in the director-musices format into a chunked dataset, chunked in the sense that ties still have to be resolved.

guess-morphetic

Started, last checked	5/7/2013, 3/10/2013
Location	Director musices
Calls	guess-morphetic-in-C-major
Called by	
Comments/see also	

Example:

```
(guess-morphetic 63 '(4 0))
--> 61.
(guess-morphetic 63 '(-2 0))
--> 62.
(guess-morphetic 70 '(5 5))
--> 65.
(guess-morphetic 70 '(1 5))
--> 66.
```

This function takes a MIDI note number and a key (represented by steps on the cycle of fiths, and mode). It attempts to guess the corresponding morphetic pitch number, given the key.

guess-morphetic-in-C-major

Started, last checked	17/11/2009, 17/11/2009
Location	Director musices
Calls	
Called by	guess-morphetic
Comments/see also	

Example:

```
(guess-morphetic-in-C-major 68)
--> 65
```

This function takes a MIDI note number as its only argument. It attempts to guess (very naively) the corresponding morphetic pitch number, assuming a key of or close to C major.

index-of-1st-tie

Started, last checked	17/11/2009, 17/11/2009
Location	Director musices
Calls	my-last
Called by	resolve-tie
Comments/see also	

Example:

```
(index-of-1st-tie
 '((4 62 61 1 0 T) (5 62 61 1/4 0 NIL)
```

```
   (21/4 64 62 1/8 0 NIL) (43/8 66 63 1/8 0 NIL)
   (11/2 67 64 1/8 0 NIL) (45/8 69 65 1/8 0 NIL)
   (23/4 71 66 1/8 0 NIL) (47/8 73 67 1/8 0 NIL)))
--> 0
```

This function returns the index of the first element of a list of lists whose last value (indicating a tie) is T.

indices-of-ties

Started, last checked	17/11/2009, 17/11/2009
Location	Director musices
Calls	
Called by	resolve-tie
Comments/see also	

Example:

```
(indices-of-ties
 '((4 62 61 1 0 T) (5 62 61 1 0 T)
   (21/4 64 62 1/8 0 NIL) (43/8 66 63 1/8 0 NIL)
   (11/2 67 64 1/8 0 NIL) (45/8 69 65 1/8 0 NIL)
   (23/4 71 66 1/8 0 NIL) (47/8 73 67 1/8 0 NIL)
   (6 62 61 1/4 0 NIL)) 0)
--> (1 8)
```

This function returns the indices of elements that have the same MIDI-morphetic pairs as the element indicated by the second argument, so long as these elements continue to be tied. This function may not be robust enough: replacing the last MNN by 63 results in an infinite loop.

MIDI-morphetic-pair2pitch&octave

Started, last checked	17/11/2009, 17/11/2009
Location	Director musices
Calls	
Called by	
Comments/see also	

Example:

```
(MIDI-morphetic-pair2pitch&octave '(70 65))
--> "A#4"
```

This function returns the pitch and octave of an input MIDI note number and morphetic pitch number.

modify-to-check-dataset

Started, last checked	17/11/2009, 15/7/2013
Location	Director musices
Calls	
Called by	
Comments/see also	

Example:

```
(modify-to-check-dataset
 '((0 50 54 5/4 1) (5/4 52 55 1/8 1)
   (11/8 54 56 1/8 1) (3/2 55 57 1/8 1)
   (13/8 57 58 1/8 1) (7/4 59 59 1/8 1)))
--> ((0 50 1250 1 90) (1250 52 125 1 90)
     (1375 54 125 1 90) (1500 55 125 1 90)
     (1625 57 125 1 90) (1750 59 125 1 90))
```

This function converts standard vector representation to events for saving as a MIDI file. Channel can be set to a default value of 1 (piano sound) or channel values can be maintained from the input variable.

pitch&octave2MIDI-morphetic-pair

Started, last checked	17/11/2009, 17/11/2009
Location	Director musices
Calls	
Called by	director-musice2datapoint
Comments/see also	

Example:

```
(pitch&octave2MIDI-morphetic-pair "A#4")
--> (70 65)
```

This function returns the MIDI note number and morphetic pitch number of an input pitch and octave.

resolve-tie

Started, last checked	17/11/2009, 17/11/2009
Location	Director musices
Calls	firstn, index-of-1st-tie, indices-of-ties, my-last, remove-nth-list
Called by	resolve-ties
Comments/see also	resolve-ties-kern

Example:

```
(resolve-tie
 '((4 62 61 1 0 T) (5 62 61 1/4 0 NIL)
   (21/4 64 62 1/8 0 NIL) (43/8 66 63 1/8 0 NIL)
   (11/2 67 64 1/8 0 NIL) (45/8 69 65 1/8 0 NIL)
   (23/4 71 66 1/8 0 NIL) (47/8 73 67 1/8 0 NIL)))
--> ((4 62 61 5/4 0 NIL) (21/4 64 62 1/8 0 NIL)
     (43/8 66 63 1/8 0 NIL) (11/2 67 64 1/8 0 NIL)
     (45/8 69 65 1/8 0 NIL) (23/4 71 66 1/8 0 NIL)
     (47/8 73 67 1/8 0 NIL))
```

This function locates notes relevant to a tie, creates a single appropriately defined note, and removes the redundant notes.

resolve-ties

Started, last checked	17/11/2009, 17/11/2009
Location	Director musices
Calls	orthogonal-projection-unique-equalp, resolve-tie
Called by	resolve-ties
Comments/see also	

Example:

```
(resolve-ties
 '((4 62 61 1 0 T) (5 62 61 1/4 0 NIL)
   (5 74 68 1 0 T)
   (21/4 64 62 1/8 0 NIL) (43/8 66 63 1/8 0 NIL)
   (11/2 67 64 1/8 0 NIL) (45/8 69 65 1/8 0 NIL)
   (23/4 71 66 1/8 0 NIL) (47/8 73 67 1/8 0 NIL)
   (6 74 68 1 0 T) (7 74 68 1/4 0 NIL)))
```

```
--> ((4 62 61 5/4 0) (5 74 68 9/4 0)
     (21/4 64 62 1/8 0) (43/8 66 63 1/8 0)
     (11/2 67 64 1/8 0) (45/8 69 65 1/8 0)
     (23/4 71 66 1/8 0) (47/8 73 67 1/8 0))
```

This function applies the function resolve-tie recursively until all ties have been resolved. At this point the input dataset is projected to remove the tie dimension.

4.2.7 Kern articulation

The functions below will parse a kern file (http://kern.ccarh.org/) by column and convert it to a list representation in which notes appear as points in pitch-time space, and performance directions such as articulation, dynamic markings, and lyrics appear as sublists of strings in later elements. The main function is kern-file2points-artic-dynam-lyrics. The functions were coded hastily and require further testing.

articulation-string2list

Started, last checked	16/6/2014, 16/6/2014
Location	Kern articulation
Calls	
Called by	parse-kern-notes-artic-dynam-lyrics
Comments/see also	

Example:

```
(setq a-string "(''^")
(articulation-string2list a-string)
--> ("(" "''" "^")
```

This function splits up a string of concatenated articulation markings into a list of articulation markings, taking care over elements such as marcato markings (′′).

dynamics-string2list

Started, last checked	16/6/2014, 16/6/2014
Location	Kern articulation
Calls	
Called by	parse-kern-notes-artic-dynam-lyrics
Comments/see also	

Example:

```
(dynamics-string2list "p")
--> ("p")
(dynamics-string2list "pp<")
--> ("pp" "<")
```

This function splits up a string of concatenated dynamic markings into a list of dynamic markings, taking care over elements such as pianissimo markings (*pp*).

kern-cols2points-artic-dynam-lyrics

Started, last checked	16/6/2014, 16/6/2014
Location	Kern articulation
Calls	append-list, constant-vector, parse-kern-notes-artic-dynam-lyrics, resolve-ties-kern, return-lists-of-length-n
Called by	kern-file2points-artic-dynam-lyrics
Comments/see also	

Example:

```
(setq
 note-list
 '((("[4f#" "4e#")) NIL (("12f#]")) (("12g#"))
   (("12f#")) (("8e#")) (("8f#")) (("8.g#")) (("16d"))
   NIL (("")) (("8c#") ("4r"))))
(setq
 artic-list
 '((("^" "(^")) NIL (("")) ((""))
   (("")) (("")) (("")) (("")) ((""))
   NIL (("^")) ((")") (""))))
```

```
(setq
 dynam-list
 '((("p")) NIL ((""))  ((""))
   ((""))  (("<")) ((""))  ((""))  ((""))
   NIL ((""))  ((""))))
(setq lyrics-list nil)
(kern-cols2points-artic-dynam-lyrics
 note-list artic-list dynam-list lyrics-list 0 '(0))
--> ((0 65 62 1 0      ("(" "^") ("p") NIL)
     (0 66 63 4/3 0    ("^")     ("p") NIL)
     (4/3 68 64 1/3 0  NIL       NIL   NIL)
     (5/3 66 63 1/3 0  NIL       NIL   NIL)
     (2 65 62 1/2 0    NIL       ("<") NIL)
     (5/2 66 63 1/2 0  NIL       NIL   NIL)
     (3 68 64 3/4 0    NIL       NIL   NIL)
     (15/4 62 61 1/4 0 NIL       NIL   NIL)
     (4 61 60 1/2 0    (")")     NIL   NIL))
```

This function combines a column of notes from a kern file with corresponding columns of articulation marks, dynamics, and lyrics. It is called by the function kern-file2points-artic-dynam-lyrics and performs a similar role to the function kern-col2dataset in the function kern-file2dataset-by-col.

kern-file2dynamics-tf

Started, last checked	16/6/2014, 19/1/2015
Location	Kern articulation
Calls	read-from-file-arbitrary, tab-separated-string2list
Called by	kern-file2points-artic-dynam-lyrics
Comments/see also	

Example:

```
(setq
 path&name
 (merge-pathnames
  (make-pathname
   :directory '(:relative "Kern")
   :name "C-6-1-ed" :type "txt")
  *MCStylistic-MonthYear-data-path*))
```

```
(kern-file2dynamics-tf path&name)
--> 9
(setq
 path&name
 (merge-pathnames
  (make-pathname
   :directory
   '(:relative
    "C@merata2014" "training_v1")
   :name "f7" :type "krn")
  *MCStylistic-MonthYear-data-path*))
(kern-file2dynamics-tf path&name)
--> NIL
```

This function determines whether the string "**dynam" appears in the kern file for a piece of music. If yes the function returns the index of the kern row where this string appears, and NIL otherwise.

19/1/2015. Introduced a check to avoid trying to parse root and harm spines in kern files with this function.

kern-file2lyrics-tf

Started, last checked	16/6/2014, 16/6/2014
Location	Kern articulation
Calls	read-from-file-arbitrary, tab-separated-string2list
Called by	kern-file2points-artic-dynam-lyrics
Comments/see also	

Example:

```
(setq
 path&name
 (merge-pathnames
  (make-pathname
   :directory
   '(:relative "C@merata2014" "training_v1")
   :name "f6" :type "krn")
  *MCStylistic-MonthYear-data-path*))
(kern-file2lyrics-tf path&name)
--> 0
```

```
(setq
 path&name
 (merge-pathnames
  (make-pathname
   :directory
   '(:relative "C@merata2014" "training_v1")
   :name "f7" :type "krn")
  *MCStylistic-MonthYear-data-path*))
(kern-file2lyrics-tf path&name)
--> NIL
```

This function determines whether the string "**text" appears in the kern file for a piece of music. If yes the function returns the index of the kern row where this string appears, and NIL otherwise.

kern-file2points-artic-dynam-lyrics

Started, last checked	16/6/2014, 16/6/2014
Location	Kern articulation
Calls	kern-anacrusis-correction, kern-cols2points-artic-dynam-lyrics, kern-file2dynamics-tf, kern-file2lyrics-tf, kern-rows2col, read-from-file-arbitrary, sort-dataset-asc, staves-info2staves-variable-robust, tab-separated-string2list, test-all-true
Called by	
Comments/see also	

Example:

```
(setq
 path&name
 (merge-pathnames
  (make-pathname
   :directory '(:relative "Kern")
   :name "C-6-1-ed" :type "txt")
  *MCStylistic-MonthYear-data-path*))
(firstn
 10 (kern-file2points-artic-dynam-lyrics path&name))
--> ((0 66 63 4/3 0 ("(" "^") ("p") NIL)
```

```
     (1 37 46 1 1 NIL NIL NIL)
     (4/3 68 64 1/3 0 NIL NIL NIL)
     (5/3 66 63 1/3 0 NIL NIL NIL)
     (2 49 53 1 1 NIL ("<") NIL)
     (2 56 57 1 1 NIL ("<") NIL)
     (2 59 59 1 1 NIL ("<") NIL)
     (2 65 62 1/2 0 NIL ("<") NIL)
     (5/2 66 63 1/2 0 NIL NIL NIL)
     (3 49 53 1 1 NIL NIL NIL))
(setq
 path&name
 (merge-pathnames
  (make-pathname
   :directory
   '(:relative
    "C@merata2014" "training_v1")
   :name "f1" :type "krn")
  *MCStylistic-MonthYear-data-path*))
(firstn
 10 (kern-file2points-artic-dynam-lyrics path&name))
--> ((0 46 52 4 3 NIL NIL ("place."))
     (0 58 59 2 2 NIL NIL ("place."))
     (0 65 63 2 1 NIL NIL ("place."))
     (0 70 66 2 0 NIL NIL ("place"))
     (3 58 59 1 2 NIL NIL ("With-"))
     (3 70 66 1 1 NIL NIL ("With-"))
     (3 74 68 1 0 NIL NIL ("With-"))
     (4 65 63 1 2 NIL NIL ("-in"))
     (4 69 65 1 1 NIL NIL ("-in"))
     (4 72 67 1 0 NIL NIL ("-in")))
```

This function is similar to kern-file2dataset-by-col. As well as converting a kern file to a point set, it includes articulation, dynamics, and lyrics information with each note/point to which they apply.

parse-kern-notes-artic-dynam-lyrics

Started, last checked	16/6/2014, 16/6/2014
Location	Kern articulation
Calls	articulation-string2list, dynamics-string2list, kern-tie-dur-pitch2list, pitch&octave2MIDI-morphetic-pair
Called by	kern-cols2points-artic-dynam-lyrics
Comments/see also	

Example:

```
(parse-kern-notes-artic-dynam-lyrics
 '("[4f#" "4e#") '("^" "(^") '("p") '("-fraid" "laid")
 0 3 0)
--> (1
     ((3 66 63 1 0 ("^") ("p") ("-fraid" "laid") "[")
      (3 65 62 1 0 ("(" "^") ("p") ("-fraid" "laid")))
```

This function converts a kern row consisting of spaced notes into a list of points, and also returns the minimum duration of those notes. It is assumed that any irrelevant symbols have already been removed via the code

```
(remove-if #'not-tie-dur-pitch-char-p *kern-note*)
```

Non-notes/rests should then result in '(0 NIL) being returned. A lone crotchet rest should result in '(1 NIL) being returned, etc.

4.2.8 Kern by col

The functions below will parse a kern file (http://kern.ccarh.org/) by column and convert it to a dataset. The main function is kern-file2dataset-by-col. Conflicts between kern's relative encoding and the absolute parsing (which affected the function kern-file2dataset) have been resolved.

append-list

Started, last checked	17/3/2012, 17/3/2012
Location	Kern by col
Calls	
Called by	kern-col2dataset
Comments/see also	append-list-of-lists

Example:

```
(append-list '(("4cc" "4dd") (7.2 -5 6) (".")))
--> ("4cc" "4dd" 7.2 -5 6 ".")
```

Removes one structural level from a list.

header2trans-vec

Started, last checked	17/3/2012, 15/7/2013
Location	Kern by col
Calls	firstn, not-tie-dur-pitch-char-p, space-bar-separated-string2list, tab-separated-string2list
Called by	kern-transp-file2dataset-by-col
Comments/see also	

Example:

```
(setq
 kern-rows
 '("!!!COM: Beethoven, Ludwig van"
   "!!!OPR: Symphony No. 3 in E-flat Major"
   "!!!ONM: Opus 55" "!!!OTL:" "!!!OMV: 1"
   "**kern **dynam"
   "*I:Corno 1, 2 in Eb *I:Corno 1, 2 in Eb"
   "*trvc 3 2 *" "*>[A,B,B,C] *" "*clefG2 *"
   "*k[] *" "*C: *" "*M3/4 *"
   "*tb24 *" "=1 =1" "(2d/ pp"
   "*>2nd ending *"))
(setq staves-variable '(((0 1) (-1/2 1)) 5))
(header2trans-vec kern-rows staves-variable)
--> ((3 2) NIL)
```

This function looks for a line in the kern file containing the string trvc, short for transposition vector. The format `*trvc 3 2` means that this instrument sounds three semitones and two stave steps higher than it is notated. The corresponding points in the point set could be translated by this amount, otherwise the MIDI and point-set encodings will contain unintended bitonalities, which is problematic for probabilistic calculations on the notes as heard, and for checking data by auditioning MIDI files.

kern-anacrusis-correction

Started, last checked	16/6/2014, 11/12/2014
Location	Kern by col
Calls	kern-col2dataset-no-tie-resolution, kern-col2rest-set, read-from-file-arbitrary, tab-separated-string2list
Called by	kern-file2dataset-by-col, kern-transp-file2dataset-by-col, kern-file2points-artic-dynam-lyrics
Comments/see also	

Example:

```
(setq
 path&name
 (merge-pathnames
  (make-pathname
   :directory '(:relative "Kern")
   :name "C-6-1-ed" :type "txt")
  *MCStylistic-MonthYear-data-path*))
(kern-anacrusis-correction path&name)
--> 1
```

This function begins parsing a kern file up to bar one (usually indicated by “=1” or “=1-”). Any notes appearing before bar one are considered to be an anacrusis. The duration of the anacrusis is calculated (using the ontime of the first note(s) in bar one) and returned. Zero is returned otherwise.

On 11 December 2014 I was converting some kern files that did not contain “=1” or “=1-”; nor did they contain anacruses. The absence of these strings caused an error, because `index-bar1` was null. To remedy this I put in some extra logic when defining `mini-dataset`.

kern-col2dataset

Started, last checked	17/3/2012, 17/3/2012
Location	Kern by col
Calls	append-list, constant-vector, parse-kern-spaced-notes, resolve-ties-kern, return-lists-of-length-n, sort-dataset-asc
Called by	kern-file2dataset-by-col
Comments/see also	

Example:

```
(kern-col2dataset
 '((("31")) (("8E")) ((".")) (("8G")) ((".")) (("8F"))
   ((".")) (("8A-")) ((".")) (("8BB")) ((".")) 
   (("8D")) ((".")) (("8C")) ((".")) (("8E-")) ((".")) 
   (("32")) (("")) (("2r") ("4DD")) NIL ((".") ("."))
   ((".") (".")) NIL ((".") ("4EE-")) NIL
   ((".") (".")) ((".") (".")) NIL (("12d") ("4FF"))
   (("12A-") (".")) (("12F") (".")) (("12c") ("4FF#"))
   (("12A") (".")) (("12E-") ("."))
   (("33") ("33")) (("12c") ("8GGG")) (("12G") ("."))
   ((".") ("8GG")) (("12E-") (".")) (("12c") ("8GGG"))
   (("12G") (".")) ((".") ("8GG")) (("12E-") ("."))
   (("12B") ("8GGG")) (("12G") (".")) ((".") ("8GG"))
   (("12D") (".")) (("12B") ("8GGG")) (("12G") ("."))
   ((".") ("8GG")) (("12D") (".")))
  0 (list 0))
--> ((0 52 55 1/2 0) (1/2 55 57 1/2 0) (1 53 56 1/2 0)
     (3/2 56 58 1/2 0) (2 47 52 1/2 0)
     (5/2 50 54 1/2 0) (3 48 53 1/2 0)
     (7/2 51 55 1/2 0) (4 38 47 1 0) (5 39 48 1 0)
     (6 41 49 1 0) (6 62 61 1/3 0) (19/3 56 58 1/3 0)
     (20/3 53 56 1/3 0) (7 42 49 1 0) (7 60 60 1/3 0)
     (22/3 57 58 1/3 0) (23/3 51 55 1/3 0)
     (8 31 43 1/2 0) (8 60 60 1/3 0)
     (25/3 55 57 1/3 0) (17/2 43 50 1/2 0)
     (26/3 51 55 1/3 0) (9 31 43 1/2 0)
     (9 60 60 1/3 0) (28/3 55 57 1/3 0)
     (19/2 43 50 1/2 0) (29/3 51 55 1/3 0)
     (10 31 43 1/2 0) (10 59 59 1/3 0)
     (31/3 55 57 1/3 0) (21/2 43 50 1/2 0)
     (32/3 50 54 1/3 0) (11 31 43 1/2 0)
     (11 59 59 1/3 0) (34/3 55 57 1/3 0)
     (23/2 43 50 1/2 0) (35/3 50 54 1/3 0))
```

This function converts a kern column consisting of spaced notes into a dataset, and also returns the minimum duration of those notes. It is assumed that any irrelevant symbols have already been removed via the code

```
(remove-if #'not-tie-dur-pitch-char-p *kern-note*)
```

Non-notes/rests should then result in '(0 NIL) being returned. A lone crotchet rest should result in '(1 NIL) being returned, etc.

kern-col2dataset-no-tie-resolution

Started, last checked	21/8/2014, 21/8/2014
Location	Kern by col
Calls	append-list, constant-vector, firstn, parse-kern-spaced-notes, sort-dataset-asc
Called by	kern-anacrusis-correction
Comments/see also	

Example:

```
(kern-col2dataset-no-tie-resolution
 '(NIL (("[4f#"))) 0 (list 0))
--> ((0 66 63 1 0))
```

This function is very similar to the function kern-col2dataset. The difference is that no attempt is made to resolve ties. This is useful for calculating the length of any anacrusis: if every note in the anacrusis is tied into the first full bar of the piece, then resolving ties can lead to incorrect calculation of the anacrusis length.

kern-col2rest-set

Started, last checked	13/6/2014, 13/6/2014
Location	Kern by col
Calls	append-list, constant-vector, parse-kern-spaced-rests, return-lists-of-length-n
Called by	kern-anacrusis-correction, kern-file2rest-set-by-col
Comments/see also	

Example:

```
(kern-col2rest-set '((("4r")) (("8r"))) 0 (list 0))
--> ((0 "rest" "rest" 1 0) (1 "rest" "rest" 1/2 0))
```

This function is similar to the function kern-col2dataset. Rather than converting written notes in a particular voice to points, it converts written rests in a particular voice to points. The output is a point set, where each point consists of an ontime, two 'rest' strings (placeholders for MIDI note and morphetic pitch numbers), duration, and staff number.

kern-file2dataset-by-col

Started, last checked	17/3/2012, 17/3/2012
Location	Kern by col
Calls	kern-anacrusis-correction, kern-col2dataset, kern-rows2col, read-from-file-arbitrary, sort-dataset-asc, staves-info2staves-variable-robust
Called by	
Comments/see also	kern-transp-file2dataset-by-col. Introduced anacrusis handling on 16/6/2014.

Example:

```
(kern-file2dataset-by-col
  (merge-pathnames
   (make-pathname
    :name "C-6-1-small" :type "krn")
   *MCStylistic-Aug2013-example-files-data-path*))
--> ((0 66 63 4/3 0) (1 37 46 1 1) (4/3 68 64 1/3 0)
     (5/3 66 63 1/3 0) (2 49 53 1 1) (2 56 57 1 1)
     (2 59 59 1 1) (2 65 62 1/2 0) (5/2 66 63 1/2 0)
     (3 49 53 1 1) (3 53 55 1 1) (3 59 59 1 1)
     (3 68 64 3/4 0) (15/4 62 61 1/4 0) (4 42 49 1 1)
     (4 61 60 1/2 0) (19/4 66 63 1/4 0) (5 54 56 1 1)
     (5 61 60 1 1) (5 69 65 1 0) (6 54 56 1 1)
     (6 61 60 1 1))
```

This function is a more robust version of the function kern-file2dataset. It converts a text file in the kern format into a dataset, where each datapoint consists of an ontime, MIDI note number, morphetic pitch number, duration, and staff number.

It is more robust because kern-file2dataset parsed a kern score by row only, so sometimes whitespace in a score was misinterpreted. For example, rows such as

```
*staff2 *staff1
```

```
4.c 4g
. 4a
8b. .
```

would be interpreted as

```
((0 60 3/2) (0 67 1) (1 69 1) (2 59 1/2))
```

rather than

```
((0 60 3/2) (0 67 1) (1 69 1) (3/2 59 1/2))
```

The example at the top of this function's documentation is a case in point.

kern-file2tie-set-by-col

Started, last checked	17/6/2014, 17/6/2014
Location	Kern by col
Calls	kern-anacrusis-correction, kern-col2dataset-no-tie-resolution, kern-rows2col, read-from-file-arbitrary, sort-dataset-asc, staves-info2staves-variable-robust, tab-separated-string2list
Called by	
Comments/see also	

Example:

```
(kern-file2tie-set-by-col
  (merge-pathnames
   (make-pathname
    :name "C-6-1-small" :type "krn")
   *MCStylistic-MonthYear-example-files-data-path*))
--> ((-1 66 63 1 0 "[") (0 66 63 1/3 0 "][")
     (1/3 66 63 1/3 0 "]") (5 66 63 1 0 "[")
     (5 69 65 1 0 "["))
```

This function converts any notes that are tied in a score into points, using '[' to mean tied forward, ']' to mean tied back, and '[]' to mean tied both.

kern-rows2col

Started, last checked	17/3/2012, 18/7/2013
Location	Kern by col
Calls	fibonacci-list, not-tie-dur-pitch-char-p, nth-list-of-lists, space-bar-separated-string2list, recognised-spine-commandp, tab-separated-string2list, update-staves-variable
Called by	kern-file2dataset-by-col
Comments/see also	

Example:

```
(setq
 rows
 '("(4b-/ 2.f\ [4.gg\ ."
   "4cc/ 4dd/ . . ."
   ". . (16.ggS\LL] ."
   ". . 32aa\JJk ."
   "4dd-/) . 16ccc\LL ."
   ". . 16bb-\ ."
   ". . 16gg\ ."
   ". . 16ee\JJ) ."
   "=60 =60 =60 =60"
   "(4b-/ 4dd-/ 4.f\ (4ee\ 4gg\ pp"
   "8a/ 8cc/) . 8ff\) ."))
(setq staves-variable '((1 2) (0 1) (-1/2 1)))
(setq i 0)
(kern-rows2col rows i staves-variable)
--> ((("4b-") ("2.f")) (("4cc" "4dd") ("."))
((".") (".")) ((".") (".")) (("4dd-") ("."))
((".") (".")) ((".") (".")) ((".") ("."))
(("60") ("60")) (("4b-" "4dd-") ("4.f"))
(("8a" "8cc") (".")))

(setq
 rows
 (list
  "*staff2 *staff1 *staff1/2"
```

```
  "=58 =58 =58" "8f/ 8a/ 8ff\\ ."
  "8r 8r ." "4r 16ccS\\LL pp"
  ". (32dd'\\L ." ". 32ee'\\ ."
  ". 32ff'\\ ." ". 32gg'\\ ." ". 32aa'\\ ."
  ". 32bb-'\\JJJ) ."
  "4c/ 4e/ 4g/ 4b-/ (32ccc\\LLL ."
  ". 32bb\\ ." ". 32ccc\\ ." ". 32ddd\\ ."
  ". 32ccc\\ ." ". 32bb-\\ ." ". 32aa\\ ."
  ". 32gg\\JJJ) ." "=59 =59 =59"
  "*^ * *" "(4b-/ 2.f\\ [4.gg\\ ."
  "4cc/ . . ."
  ". . (16.ggS\\LL] ."
  ". . 32aa\\JJk ."
  "4dd-/) . 16ccc\\LL ."
  ". . 16bb-\\ ." ". . 16gg\\ ."
  ". . 16ee\\JJ) ."
  "=60 =60 =60 =60"
  "(4b-/ 4dd-/ 4.f\\ (4ee\\ 4gg\\ pp"
  "8a/ 8cc/) . 8ff\\) ."
  "8r 4.ry 8r ." "4r . 4r ."
  "*v *v * *" "*clefF4 * *"
  "=61 =61 =61"
  (concatenate
   'string
   "8C'\\ 8E'\\ 8G'\\ 8c'\\ "
   "8g'\\ 8b-'\\ 8cc'\\ 8ee'\\ 8gg'\\ pp")
  "8r 8r ."
  (concatenate
   'string
   "8FF'/ 8AA'/ 8C'/ 8F'/ "
   "8a'\\ 8cc'\\ 8ff'\\ .")
  "8r; 8r; ." "== == ==")
(setq staves-variable '((1 1) (0 1) (-1/2 1)))
(setq i 0)
(kern-rows2col rows i staves-variable)
--> ((("aff2")) (("58")) (("8f" "8a")) (("8r"))
(("4r")) ((".")) ((".")) ((".")) ((".")) (("."))
((".")) (("4c" "4e" "4g" "4b-")) ((".")) (("."))
((".")) ((".")) ((".")) ((".")) ((".")) (("59"))
(("")) (("4b-") ("2.f")) (("4cc") (".")) ((".") ("."))
((".") (".")) (("4dd-") (".")) ((".") ("."))
```

```
((".") (".")) ((".") (".")) (("60") ("60"))
(("4b-" "4dd-") ("4.f")) (("8a" "8cc") ("."))
(("8r") ("4.r")) (("4r") (".")) (("") (""))
(("cefF4")) (("61")) (("8C" "8E" "8G" "8c")) (("8r"))
(("8FF" "8AA" "8C" "8F")) (("8r")) ((""")))
```

This function focuses on events in kern rows that occur on the ith stave, and returns only those events as datapoints.

Encountered some kern files that used spine commands (beginning $*$) to encode information other than splitting ($*\wedge$) or collapsing ($*\vee$). Introduced a third test to the and condition, checking whether the first character in a kern row is $*$. If so the function recognised-spine-commandp checks the row for recognised spine commands, and the row is ignored if none are found.

kern-transp-file2dataset-by-col

Started, last checked	15/7/2013, 15/7/2013
Location	Kern by col
Calls	header2trans-vec, kern-anacrusis-correction, kern-col2dataset, kern-rows2col, read-from-file-arbitrary, sort-dataset-asc, staves-info2staves-variable-robust
Called by	
Comments/see also	kern-file2dataset-by-col. Introduced anacrusis handling on 16/6/2014.

Example:

```
\noindent Example:
\begin{verbatim}
(kern-transp-file2dataset-by-col
  (merge-pathnames
   (make-pathname
    :name "B-55-1-small" :type "krn")
   *MCStylistic-Mar2013-example-files-data-path*)
  t)
--> ((0 65 63 2 0))
```

This function is very similar to kern-file2dataset-by-col. The difference is that this function is intended to be run on kern files that contain transposing instruments in them. It is assumed that the kern file will contain a string such as `"*trvc 3 2 *"` soon after the announcement of staves `"**kern **dynam"`, to indicate for instance that notes on the first stave will sound three semitones and two staff steps higher than written.

parse-kern-spaced-rests

Started, last checked	13/6/2014, 13/6/2014
Location	Kern by col
Calls	kern-tie-dur-pitch2list, pitch&octave2MIDI-morphetic-pair
Called by	kern-col2rest-set
Comments/see also	parse-kern-spaced-notes

Example:

```
(setq a-list '("4C#" "4r" "4B" "8g#"))
(setq staff-index 0)
(setq ontime 3)
(setq minimum-duration 0)
(parse-kern-spaced-rests
 a-list staff-index ontime minimum-duration)
--> (1/2
     ((3 "rest" "rest" 1 0) (3 59 59 1 0)
      (3 68 64 1/2 0)))
```

This function is the rests equivalent of the function parse-kern-spaced-notes.

4.2.9 Kern

The functions below will parse a kern file (http://kern.ccarh.org) and convert it to a dataset. The main function is kern-file2dataset. Occasionally there are conflicts between kern's relative encoding and the timewise parsing function. These have been resolved by the function kern-file2dataset-by-col.

accidental-char-p

Started, last checked	30/6/2010, 30/6/2010
Location	Kern
Calls	
Called by	kern-pitch-chars2pitch&octave
Comments/see also	

Example:

```
(accidental-char-p #\a)
--> nil
```

This function returns true if the input character is associated with kern's representation of accidentals.

always-nil

Started, last checked	16/6/2014, 16/6/2014
Location	Kern
Calls	
Called by	kern-file2points-artic-dynam-lyrics
Comments/see also	

Example:

```
(always-nil #\e)
--> nil
```

This function always returns nil. It is useful for passing as a compiled function to the function kern-rows2col.

concat-strings

Started, last checked	16/6/2014, 16/6/2014
Location	Kern
Calls	
Called by	followed-by-splitter
Comments/see also	space-bar-separated-string2list

Example:

```
(concat-strings '("put " 7 "us " "together"))
--> "put us together"
```

This function by Svante concatenates a list of strings, ignoring elements of the list that are not strings.

index-of-backward-tie

Started, last checked	30/6/2010, 30/6/2010
Location	Kern
Calls	
Called by	resolve-ties-kern
Comments/see also	

Example:

```
(index-of-backward-tie
 '((4 62 61 1 0 "[") (5 63 61 1 0 "[")
   (21/4 64 62 1/8 0 "[") (43/8 63 61 1/8 0 "]")
   (45/8 64 62 1/8 0 "][") (23/4 62 61 1/8 0 "]")
   (47/8 64 62 1/8 0 "]") (6 63 61 1/4 0 "]")) 2)
--> 5
```

This function returns the index of the element that has the same MIDI-morphetic pairs as the element indicated by the second argument, so long as this element is tied backward.

kern-dur-pitch2pitch&octave-dur

Started, last checked	30/6/2010, 13/7/2013
Location	Kern
Calls	kern-pitch-chars2pitch&octave
Called by	kern-tie-dur-pitch2list
Comments/see also	

Example:

```
(kern-dur-pitch2pitch&octave-dur "8e#")
--> ("E#4" 1/2)
```

This function converts a kern note into pitch-and-octave-number and a duration. It is assumed that any irrelevant symbols have already been removed via the function remove-if in combination with the test function not-tie-dur-pitch-char-p as applied to `*kern-note*`. Non-notes should then result in nil being returned.

kern-file2dataset

Started, last checked	30/6/2010, 30/6/2010
Location	Kern
Calls	parse-kern-row, read-from-file-arbitrary, resolve-ties-kern, staves-info2staves-variable
Called by	
Comments/see also	

Example:

```
(firstn
 10
 (kern-file2dataset
  (concatenate
   'string
   *MCStylistic-Oct2010-example-files-path*
   "/vivaldi-op6-no3-2.txt")))
--> ((0 49 53 1 3) (0 49 53 1 5) (0 69 65 1 2)
     (0 76 69 1 1) (0 79 71 1 0) (1 49 53 1 3)
     (1 49 53 1 5) (1 69 65 1 2) (1 76 69 1 1)
     (1 79 71 1 0))
```

This function converts a text file in the kern format into a dataset, where each datapoint consists of an ontime, MIDI note number, morphetic pitch number, duration, and staff number.

kern-pitch-chars2pitch&octave

Started, last checked	30/6/2010, 30/6/2010
Location	Kern
Calls	accidental-char-p, upcase-p
Called by	kern-dur-pitch2pitch&octave-dur
Comments/see also	

Example:

```
(kern-pitch-chars2pitch&octave "e#")
--> "E#4"
```

This function converts kern pitch characters into the pitch-and-octave-number representation. It can accept junk input, but may produce junk output. For example, try '.' or '∗' as input.

kern-tie-dur-pitch2list

Started, last checked	30/6/2010, 18/7/2013
Location	Kern
Calls	kern-dur-pitch2pitch&octave-dur, number-chars-p, upcase-p
Called by	parse-kern-spaced-notes
Comments/see also	

Example:

```
(kern-tie-dur-pitch2list "[8e#]")
--> ("E#4" 1/2 "][")
```

This function converts a kern note into a list consisting of pitch-and-octave, duration, and tie type. It is assumed that any irrelevant symbols have already been removed via the function remove-if in combination with the test function not-tie-dur-pitch-char-p as applied to `*kern-note*`. Non-notes should then result in nil being returned.

not-articulation-char-p

Started, last checked	16/6/2014, 16/6/2014
Location	Kern
Calls	
Called by	kern-file2points-artic-dynam-lyrics
Comments/see also	

Example:

```
(not-articulation-char-p #\e)
--> T
```

This function returns true if the input character is not associated with kern's representation of articulation.

not-dynamics-char-p

Started, last checked	16/6/2014, 16/6/2014
Location	Kern
Calls	
Called by	kern-file2points-artic-dynam-lyrics
Comments/see also	

Example:

```
(not-articulation-char-p #\e)
--> T
```

This function returns true if the input character is not associated with kern's representation of dynamics.

not-tie-dur-pitch-char-p

Started, last checked	30/6/2010, 18/7/2013
Location	Kern
Calls	
Called by	parse-kern-row
Comments/see also	

Example:

```
(not-tie-dur-pitch-char-p #\h)
--> T
```

This function returns true if the input character is not associated with kern's representation of pitch.

number-chars-p

Started, last checked	30/6/2010, 30/6/2010
Location	Kern
Calls	
Called by	kern-tie-dur-pitch2list
Comments/see also	

Example:

```
(number-chars-p #\2)
--> nil
```

This function returns true if the input character is 0-9.

parse-kern-row

Started, last checked	30/6/2010, 18/7/2013
Location	Kern
Calls	not-tie-dur-pitch-char-p, parse-kern-row-as-notes, space-bar-separated-string2list, tab-separated-string2list, update-staves-variable
Called by	kern-file2dataset
Comments/see also	

Example:

```
(parse-kern-row
 "2d#/ 2f#/      ⇥4A\       ⇥12cc#\L]       ⇥."
 '((1 2) (0 1) (1/2 1)) 15)
--> (((1 2) (0 1) (1/2 1))
     46/3
     ((15 63 61 2 1) (15 66 63 2 1) (15 57 58 1 1))
     ((15 73 67 1/3 0 "]")))
(parse-kern-row
 "*       ⇥*v       ⇥*v       ⇥*"
 '((1 1) (0 2) (1/2 1)) 15)
--> (((1 1) (0 1) (1/2 1)) 15)
(parse-kern-row
 ".       ⇥.       ⇥16r       ⇥."
 '((1 2) (0 1) (1/2 1)) 15)
--> (((1 2) (0 1) (1/2 1)) 61/4)
```

This function parses a kern row, consisting of notes/rests, changes to the staves variable, or irrelevant information for our purposes. The ouptut is the staves variable, the new ontime, new datapoints, and new tied datapoints.

parse-kern-row-as-notes

Started, last checked	30/6/2010, 30/6/2010
Location	Kern
Calls	parse-kern-spaced-notes
Called by	parse-kern-row
Comments/see also	

Example:

```
(parse-kern-row-as-notes
 '(("2d#" "2f#") ("4A") ("12cc#]") (""))
 '((1 2) (0 1) (1/2 1)) 15)
--> (1/3
     ((15 63 61 2 1) (15 66 63 2 1) (15 57 58 1 1)
      (15 73 67 1/3 0 "]")))
```

This function converts a kern row consisting of tabbed notes into a list of datapoints, and also returns the minimum duration of those notes. It recurses over the staves-variable to ensure that each note is labelled correctly according to staff. It is assumed that any irrelevant symbols have already been removed via the the function remove-if in combination with the test function not-tie-dur-pitch-char-p as applied to `*kern-note*`. Non-notes/rests should then result in '(0 NIL) being returned. A lone crotchet rest should result in '(1 NIL) being returned, etc.

parse-kern-spaced-notes

Started, last checked	30/6/2010, 30/6/2010
Location	Kern
Calls	kern-tie-dur-pitch2list, pitch&octave2MIDI-morphetic-pair
Called by	parse-kern-row-as-notes
Comments/see also	parse-kern-spaced-rests

Example:

```
(parse-kern-spaced-notes
 '("[4C#" "4E#" "4B" "8g#") 0 3 0)
--> (1/2
     ((3 49 53 1 0 "[") (3 53 55 1 0) (3 59 59 1 0)
      (3 68 64 1/2 0)))
```

This function converts a kern row consisting of spaced notes into a list of datapoints, and also returns the minimum duration of those notes. It is assumed that any irrelevant symbols have already been removed via the the function remove-if in combination with the test function not-tie-dur-pitch-char-p as applied to `*kern-note*`. Non-notes/rests should then result in '(0 NIL) being returned. A lone crotchet rest should result in '(1 NIL) being returned, etc.

recognised-spine-commandp

Started, last checked	15/7/2013, 15/7/2013
Location	Kern
Calls	
Called by	kern-rows2col
Comments/see also	

Example:

```
(recognised-spine-commandp "*>2nd ending *")
--> NIL
(recognised-spine-commandp "* *")
--> NIL
(recognised-spine-commandp "*^ *")
--> T
```

Some kern files that used spine commands (beginning $*$) to encode information other than splitting ($*\wedge$) or collapsing ($*\vee$). This function checks whether the input kern row contains any recognised spine commands, outputting T if this is the case, and NIL otherwise.

resolve-ties-kern

Started, last checked	30/6/2010, 30/6/2010
Location	Kern
Calls	index-of-backward-tie
Called by	kern-file2dataset
Comments/see also	resolve-ties

Example:

```
(resolve-ties-kern
 '((4 62 61 1 0 "[") (5 63 61 1 0 "[")
   (21/4 64 62 1/8 0 "[") (43/8 63 61 1/8 0 "]")
   (45/8 64 62 1/8 0 "][") (23/4 62 61 1/8 0 "]")
   (47/8 64 62 1/8 0 "]") (6 63 61 1/4 0 "]"))
 '((0 60 60 1 0)))
--> ((0 60 60 1 0) (4 62 61 15/8 0) (5 63 61 1/2 0)
     (21/4 64 62 3/4 0))
```

This function resolves tied datapoints by applying the function index-of-backward-tie recursively. It is quite similar to the function resolve-ties, which was defined for reading director- musices files.

return-lists-of-length-n

Started, last checked	30/6/2010, 30/6/2010
Location	Kern
Calls	
Called by	parse-kern-row
Comments/see also	Consider changing location.

Example:

```
(return-lists-of-length-n
 '((1 0) (0) (2 -1) nil (1 2 3) (7 -2)) 2)
--> ((1 0) (2 -1) (7 -2))
```

Returns all lists in a list of lists that are of length n.

space-bar-positions

Started, last checked	30/6/2010, 30/6/2010
Location	Kern
Calls	
Called by	space-bar-separated-string2list
Comments/see also	comma-positions, tab-positions

Example:

```
(space-bar-positions
 "4C#\ 4G#\ 4B\        ⇥8e#/L        ⇥<")
--> (3 7)
```

This function returns the positions at which space-bar symbols occur in a string.

space-bar-separated-string2list

Started, last checked	30/6/2010, 30/6/2010
Location	Kern
Calls	space-bar-positions
Called by	parse-kern-row
Comments/see also	comma-separated-string2list, tab-separated-string2list, concat-strings

Example:

```
(space-bar-separated-string2list
 "4C#\ 4G#\ 4B\        ⇥8e#/L        ⇥<")
--> ("4C#" "4G#" "4B        ⇥8e#/L        ⇥<")
```

This function turns a space-bar-separated string into a list, where formerly each item was preceded or proceeded by a space.

split-or-collapse-index

Started, last checked	30/6/2010, 30/6/2010
Location	Kern
Calls	
Called by	update-staves-variable
Comments/see also	

Example:

```
(split-or-collapse-index 6 '(2 3 5 7 8))
--> 3
(split-or-collapse-index nil '(2 3 5 7 8))
--> nil
(split-or-collapse-index 8 '(2 3 5 7 8))
--> nil
```

Returns the index of the second argument at which the first argument is exceeded. Deals with degenerate cases as indicated.

staff-char-p

Started, last checked	30/6/2010, 30/6/2010
Location	Kern
Calls	
Called by	staves-info2staves-variable
Comments/see also	

Example:

```
(staff-char-p #\2)
--> nil
```

This function returns true if the input character is '$*$', 's', 't', 'a', or 'f'.

staves-info2staves-variable

Started, last checked	30/6/2010, 30/6/2010
Location	Kern
Calls	staff-char-p, tab-separated-string2list
Called by	kern-file2dataset
Comments/see also	

Example:

```
(staves-info2staves-variable
 '("!!!COM: Chopin, Frederic"
   "!!!CDT: 1810///-1849///"
   "!!!OTL: Mazurka in F-sharp Minor, Op. 6, No. 1"
   "!!!OPS: Op. 6" "!!!ONM: No. 1"
   "!!!ODT: 1830///-1832///"
   "!!!PDT: 1832///-1833///"
   "!!!PPP: Leipzig (1832); Paris (1833) and London"
   "!!!ODE: Pauline Plater"
   "**kern       ⇥**kern       ⇥**dynam"
   "*thru       ⇥*thru       ⇥*thru"
   "*staff2       ⇥*staff1       ⇥*staff1/2"
   "*Ipiano       ⇥*Ipiano       ⇥*Ipiano"
   "*>A       ⇥*>A       ⇥*>A"))
--> ((1 1) (0 1) (-1/2 1))
```

This function looks through the first few rows of a parsed kern file and determines how many staves there are, leading to the definition of the staves variable.

staves-info2staves-variable-robust

Started, last checked	30/6/2010, 30/6/2010
Location	Kern
Calls	staves-info2staves-variable, staves-variable-index, tab-separated-string2list
Called by	kern-file2dataset-by-col
Comments/see also	

Example:

```
(staves-info2staves-variable-robust
 '("!!!COM: Chopin, Frederic"
   "**kern **kern **dynam"
   "*thru *thru *thru"
   "*>A *>A *>A"))
--> (((1 1) (0 1) (-1/2 1)) 1)

(staves-info2staves-variable-robust
 '("!!!COM: Beethoven, Ludwig van"
   "!!!CDT: 1770///-1827///"
   "**kern **dynam" "*Ipiano *Ipiano"
   "*clefG2 *clefG2" "*k[b-] *k[b-]"
   "*F: *F:" "*M3/4 *M3/4" "*MM40 *MM40"
   "8.c/L ." "16c/Jk ." "=1 =1" "* *"
   "(4aS/ p" "ccq/ ." "8b-/L ."
   "8a/ ." "8g/ ." "8f/J) ." "=2 =2"
   "(4f/ ." "8e/) ." "(8c/L ."
   "8d/ ." "8e/J ." "* *" "=3 =3" "* *"
   "(8f/L ." "16cc/Jk) ." "16r ."
   "(8cc/L ." "16b-/Jk) ." "16r ."
   "(8b-/L ." "16a/Jk) ." "16r ." "* *"
   "=4 =4" "4a/ ." "16g/ ."))
--> (((0 1) (-1/2 1)) 2).
```

This function looks through the first few rows of a parsed kern file and determines how many staves there are, leading to the definition of the staves variable. The index of the row of the staves variable is also returned.

The function is more robust than a similar function called staves-info2staves-variable, because it can determine the number of staves without the presence of a line containing "*staff" strings.

staves-variable-index

Started, last checked	30/6/2010, 30/6/2010
Location	Kern
Calls	add-to-list, first-n-naturals
Called by	staves-info2staves-variable-robust
Comments/see also	

Example:

```
(staves-variable-index
 '("**kern" "**dynam" "**kern" "**dynam") 2)
--> ((1 1) (1/2 1) (0 1) (-1/2 1))
```

This function converts a string containing staff information into a list of staff numbers. Columns for dynamics are given fractional values.

tab-positions

Started, last checked	30/6/2010, 30/6/2010
Location	Kern
Calls	
Called by	tab-separated-string2list
Comments/see also	comma-positions, space-bar-positions

Example:

```
(tab-positions "4C#\ 4G#\ 4B\       ⇥8e#/L       ⇥<")
--> (10 16)
```

This function returns the positions at which tabs occur in a string.

tab-separated-string2list

Started, last checked	30/6/2010, 30/6/2010
Location	Kern
Calls	tab-positions
Called by	parse-kern-row, staves-info2staves-variable, update-staves-variable
Comments/see also	comma-separated-string2list, space-bar-separated-string2list

Example:

```
(tab-separated-string2list
 "4C#\ 4G#\ 4B\       ⇥8e#/L       ⇥<")
--> ("4C#\ 4G#\ 4B\" "8e#/L" "<")
```

This function turns a tab-separated string into a list, where formerly each item was preceded or proceeded by a tab.

tied-kern-note-p

Started, last checked	30/6/2010, 18/7/2013
Location	Kern
Calls	
Called by	
Comments/see also	

Example:

```
(tied-kern-note-p "12f#/L]")
--> T
```

This function returns true if the input kern note is tied over, from or both.

upcase-p

Started, last checked	30/6/2010, 30/6/2010
Location	Kern
Calls	
Called by	kern-pitch-chars2pitch&octave
Comments/see also	

Example:

```
(upcase-p #\a)
--> nil
```

This function returns true if the input character is upper case, and nil otherwise.

update-staves-variable

Started, last checked	30/6/2010, 30/6/2010
Location	Kern
Calls	fibonacci-list, index-item-1st-occurs, split-or-collapse-index, tab-separated-string2list
Called by	parse-kern-row
Comments/see also	

Example:

```
(update-staves-variable
 '((1 1) (0 1) (1/2 1)) "*       ⇥*^       ⇥*")
--> ((1 1) (0 2) (1/2 1))
(update-staves-variable
 '((1 1) (0 2) (1/2 1)) "*       ⇥*v       ⇥*v       ⇥*")
--> ((1 1) (0 1) (1/2 1))
(update-staves-variable
 '((1 2) (0 1) (1/2 1)) "*       ⇥*       ⇥*^       ⇥*")
--> ((1 2) (0 2) (1/2 1))
```

The staves-variable is a list of pairs. The first of each pair gives the staff to which a note belongs. The second of each pair indicates whether that stave is split into multiple voices. The symbol '∗' means leave this staff as it is, the symbol '^' means this staff is splitting into an extra voice, and the symbol 'v' means this staff is collapsing into one less voice.

4.2.10 Kern rests

The functions below will parse a kern file (http://kern.ccarh.org/) by column and convert the rests therein to a point set. The main function is kern-file2rest-set-by-col.

kern-file2rest-set-by-col

Started, last checked	13/6/2014, 13/6/2014
Location	Kern rests
Calls	kern-anacrusis-correction, kern-col2rest-set, read-from-file-arbitrary, sort-dataset-asc, staves-info2staves-variable-robust, tab-separated-string2list
Called by	
Comments/see also	

Example:

```
(kern-file2rest-set-by-col
  (merge-pathnames
   (make-pathname
    :name "C-6-1-small" :type "krn")
   *MCStylistic-MonthYear-example-files-data-path*))
--> ((-1 1 1) (3 1 0) (7/2 1/4 0) (4 1 0))
```

This function is similar to the function kern-file2dataset-by-col. Rather than converting written notes to points, it converts written rests to points. The output is a point set, where each point consists of an ontime, two 'rest' strings (placeholders for MIDI note and morphetic pitch numbers), duration, and staff number. The function was written for retrieving rests, which was part of the requirements for the MediaEval 2014 C@merata task.

rest-duration-time-intervals

Started, last checked	13/6/2014, 13/6/2014
Location	Kern rests
Calls	dataset-restricted-to-m-in-nth, duration-string2numeric, modify-question-by-staff-restriction, pitch&octave2MIDI-morphetic-pair, restrict-dataset-in-nth-to-xs
Called by	**??**
Comments/see also	duration-time-intervals

Example:

```
(rest-duration-time-intervals
 "sixteenth note rest"
```

```
 '((-1 1 1) (3 1 0) (7/2 1/4 0) (4 1 0))
 '(("piano left hand" "bass clef")
   ("piano right hand" "treble clef")))
--> ((7/2 15/4))
(rest-duration-time-intervals
 "crotchet rest in the piano left hand"
 '((-1 1 1) (3 1 0) (7/2 1/4 0) (4 1 0))
 '(("piano left hand" "bass clef")
   ("piano right hand" "treble clef")))
--> ((-1 0))
```

This function returns (ontime, offtime) pairs of points (rests) that have the duration specified by the first string argument. It can be in the format 'dotted minim rest' or 'dotted half note rest', for instance. The function does not look for dotted rests in the case of the word dotted, but adds one half of the value to the corresponding rest type and looks for the numeric value.

4.3 Pattern rating

4.3.1 Projection

The main functions of use here are for creating projections of datasets (Meredith et al., 2002), which is a precursor to pattern discovery.

difference-list

Started, last checked	23/7/2009, 23/7/2009
Location	Projection
Calls	remove-nth, subtract-list-from-each-list
Called by	
Comments/see also	Deprecated.

Example:

```
(difference-list '((8 -2 -3) (4 6 6) (4 7 -3)))
--> ((-4 8 9) (-4 9 0) (4 -8 -9) (0 1 -9) (4 -9 0)
     (0 -1 9))
```

The argument to this function is a list consisting of sublists of equal lengths. For $i = 1, 2, \ldots, n$, the ith sublist S is removed from the argument to give a list L, and the function subtract-list-from-each-list is applied.

difference-lists

Started, last checked	8/3/2013, 8/3/2013
Location	Projection
Calls	subtract-two-lists
Called by	most-frequent-difference-vector
Comments/see also	

Example:

```
(difference-lists
 '((2 1) (2 2)) '((8 -2) (4 6) (4 7)))
--> ((6 -3) (2 5) (2 6) (6 -4) (2 4) (2 5))
```

The arguments to this function are two lists, assumed to be of the same dimension (that is, each sublist has the same length), but possibly of different lengths. The difference between each pair of sublist items is computed and output as a single list.

index-1st-sublist-item<=

Started, last checked	23/7/2009, 23/7/2009
Location	Projection
Calls	
Called by	test-equal<subset
Comments/see also	index-nth-sublist-item<=, index-1st-sublist-item<

Example:

```
(index-1st-sublist-item<=
 6 '(14 14 14 11 7 7 6 6 4 1 1 0 0))
--> 6
```

This function takes two arguments: a real number x and a list L of real numbers. It returns the index of the first element of L which is less than or equal to x.

index-1st-sublist-item>=

Started, last checked	23/7/2009, 23/7/2009
Location	Projection
Calls	
Called by	test-equal<subset
Comments/see also	index-nth-sublist-item>=, index-1st-sublist-item>

Example:

```
(index-1st-sublist-item>=
 4 '(0 0 0 1 1 4 6 6 7 7 11 14 14 14))
--> 5
```

This function takes two arguments: a real number x and a list L of real numbers. It returns the index of the first element of L which is greater than or equal to x.

is-maximal-translatable-pattern

Started, last checked	23/7/2009, 23/7/2009
Location	Projection
Calls	difference-list, maximal-translatable-pattern
Called by	
Comments/see also	A more efficient implementation is required.

Example:

```
(is-maximal-translatable-pattern
 '((0 1/2) (1/2 1/2))
 '((0 1/2) (1/2 1/2) (1 1/2) (1 1) (2 3) (5 1/2)
   (11/2 1/2)))
--> (5 0)
```

Two arguments are supplied to this function: a pattern P and a dataset D. If P is a maximal translatable pattern in D for some vector $\mathbf{u}$, then $\mathbf{u}$ is returned. NIL is returned otherwise.

maximal-translatable-pattern

Started, last checked	23/7/2009, 23/7/2009
Location	Projection
Calls	add-two-lists, subtract-two-lists, vector<vector
Called by	is-maximal-translatable-pattern
Comments/see also	Should be deprecated by implementing a version analogous to translators-of-pattern-in-dataset from Ukkonen et al. (2003). See also maximal-translatable-pattern-mod-2nd-n.

Example:

```
(maximal-translatable-pattern
 '(2 0)
 '((0 1/2) (0 1) (1 1) (2 1/2) (2 1) (3 2)))
--> ((0 1) (0 1/2))
```

This function assumes that the dataset is sorted ascending. This enables a more efficient search for the maximal translatable pattern of an arbitrary vector u, searching in some dataset D, defined by $\mathrm{MTP}(\mathbf{u}, D) = \{\mathbf{d} \in D : \mathbf{d} + \mathbf{u} \in D\}$.

nth-list-index

Started, last checked	23/7/2009, 23/7/2009
Location	Projection
Calls	
Called by	orthogonal-projection-not-unique-equalp
Comments/see also	

Example:

```
(nth-list-index '(1 1 0 1 0))
--> (0 1 3)
```

This function returns the value of the increment i if the ith element of the input list is equal to 1.

orthogonal-projection-not-unique-equalp

Started, last checked	23/7/2009, 23/7/2009
Location	Projection
Calls	nth-list, nth-list-index
Called by	orthogonal-projection-unique-equalp
Comments/see also	

Example:

```
(orthogonal-projection-not-unique-equalp
 '((2 4 -1 6 9) (0 0 4 2 -7) (-3 -2 -1 -1 1)
   (12 0 -7 5 3) (1 2 3 4 3) (1 2 5 4 5))
 '(1 1 0 1 0))
--> ((2 4 6) (0 0 2) (-3 -2 -1) (12 0 5) (1 2 4)
     (1 2 4))
```

Given a set of vectors (all members of the same n-dimensional vector space), and an n-tuple of zeros and ones indicating a particular orthogonal projection, this function returns the projected set of vectors.

orthogonal-projection-unique-equalp

Started, last checked	23/7/2009, 23/7/2009
Location	Projection
Calls	orthogonal-projection-not-unique-equalp, sort-dataset-asc
Called by	
Comments/see also	

Example:

```
(orthogonal-projection-unique-equalp
 '((2 4 -1 6 9) (0 0 4 2 -7) (-3 -2 -1 -1 1)
   (12 0 -7 5 3) (1 2 3 4 3) (1 2 5 4 5)
   (12 0 -6 5 4) (-3 -2 1 -1 0) (12 0 -7 5 4))
 '(1 1 0 1 0))
--> ((2 4 6) (0 0 2) (-3 -2 -1) (12 0 5) (1 2 4))
```

Given a set of vectors (all members of the same n-dimensional vector space), and an n-tuple of zeros and ones indicating a particular orthogonal projection, this function returns the projected set of vectors. Coincidences are reduced to single vectors.

pair-off-lists

Started, last checked	23/7/2009, 23/7/2009
Location	Projection
Calls	
Called by	
Comments/see also	

Example:

```
(pair-off-lists '("asc" "asc" "asc") '(0 1 2))
--> (("asc" 0) ("asc" 1) ("asc" 2))
```

Two lists A and B of equal length are provided as arguments to this function. The first element a_1 of A is paired off with the first element b_1 of B to become the first sublist of a new list, and so on for a_2 and b_2, a_3 and b_3.

test-equal<subset

Started, last checked	23/7/2009, 23/7/2009
Location	Projection
Calls	index-1st-sublist-item<=, index-1st-sublist-item>=, my-last, nth-list-of-lists, test-equal<list-elements
Called by	
Comments/see also	

Example:

```
(test-equal<subset '((4 6) (6 5) (6 5) (6 7))
 '((0 1) (0 2) (1 3) (1 4) (4 6) (6 5) (6 7)
   (7 9) (7 10) (11 11) (14 1) (14 3) (14 14)))
--> T
```

There are two arguments to this function, both lists of n-tuples. If when written as sets, the first argument is a subset of the second, then T is returned. Otherwise NIL is returned (and an empty first argument is permissible). The $<$ in the function name indicates that a subfunction, test-equal<list-elements, assumes an argument has been sorted ascending by each of its elements.

4.3.2 Musical properties

These functions aid the calculation musical attributes, such as the number of intervallic leaps in a melody. Some of the attributes are implementations of definitions from Pearce and Wiggins (2007); von Hippel (2000); Eerola and North (2000).

cons-ith-while-floor-jth-constantp

Started, last checked	19/10/2009, 19/10/2009
Location	Musical properties
Calls	
Called by	density
Comments/see also	cons-ith-while-jth-constantp

Example:

```
(cons-ith-while-floor-jth-constantp
 '((13 55) (13 60) (13 64) (27/2 63) (14 55) (15 55)
   (15 59) (15 65) (16 55) (17 72) (18 55) (19 55)
   (22 55) (23 60) (24 55) (24 59) (25 55)) 1 0)
--> (55 60 64 63)
```

This function makes a list from the ith item of each list in a list of lists, so long as the floor of the jth item is constant.

cons-ith-while-jth-constantp

Started, last checked	19/10/2009, 19/10/2009
Location	Musical properties
Calls	
Called by	top-line
Comments/see also	cons-ith-while-floor-jth-constantp

Example:

```
(cons-ith-while-jth-constantp
 '((13 55) (13 60) (13 64) (14 55) (15 55) (15 59)
   (15 65) (16 55) (17 72) (18 55) (19 55) (22 55)
   (23 60) (24 55) (24 59) (25 55) (25 67)) 1 0)
--> (55 60 64)
```

This function makes a list from the ith item of each list in a list of lists, so long as the jth item is constant.

cons-ith-while-jth-constantp

Started, last checked	19/10/2009, 19/10/2009
Location	Musical properties
Calls	
Called by	top-line
Comments/see also	cons-ith-while-floor-jth-constantp

Example:

```
(cons-ith-while-jth-constantp
 '((13 55) (13 60) (13 64) (14 55) (15 55) (15 59)
   (15 65) (16 55) (17 72) (18 55) (19 55) (22 55)
   (23 60) (24 55) (24 59) (25 55) (25 67)) 1 0)
--> (55 60 64)
```

This function makes a list from the ith item of each list in a list of lists, so long as the jth item is constant.

density

Started, last checked	19/10/2009, 19/10/2009
Location	Musical properties
Calls	cons-ith-while-floor-jth-constantp
Called by	rhythmic-density
Comments/see also	

Example:

```
(density
 '((13 55) (13 60) (13 64) (27/2 63) (14 55)) 13)
--> 4
```

In a pattern $P = \{\mathbf{p}_1, \mathbf{p}_2, \ldots, \mathbf{p}_l\}$, let $\mathbf{p}_i$ have ontime x_i, $i = 1, 2, \ldots, l$. The tactus beats are then the integers from $a = \lfloor x_1 \rfloor$ to $b = \lfloor x_l \rfloor$, assuming that beats coincide with integer ontimes and that the bottom number in the time signature does not change over the course of the pattern. The rhythmic density of the pattern at beat $c \in [a, b]$, denoted $\rho(P, c)$, is given by the cardinality of the set of all pattern points such that $\lfloor x_i \rfloor = c$.

intervallic-leaps

Started, last checked	19/10/2009, 19/10/2009
Location	Musical properties
Calls	spacing-items, top-line
Called by	
Comments/see also	small-intervals

Example:

```
(intervallic-leaps
 '((13 57) (13 60) (13 62) (14 57) (15 57) (15 59)
   (15 63) (16 57) (17 67) (18 57) (19 57) (22 57)
   (23 60) (24 57) (24 59) (25 57) (25 64)))
--> 7
```

This variable counts the number of intervallic leaps present in the melody line of a pattern, the intuition being that leaping melodies may be rated as more noticeable or important. Any interval larger than a major third counts, and the same 'top- line' rule as in the function small-intervals is observed.

max-pitch-centre

Started, last checked	19/10/2009, 19/10/2009
Location	Musical properties
Calls	mean, nth-list-of-lists
Called by	
Comments/see also	

Example:

```
(max-pitch-centre
 '(((0 60) (1 61)) ((3 48) (4 49))) 1
 '((0 60) (1 61) (2 62) (3 48) (3 57) (4 49)))
--> 23/3
```

Pitch centre is defined as 'the absolute distance, in semitones, of the mean pitch of a [pattern]...from the mean pitch of the dataset' (Pearce and Wiggins, 2007, p. 78). By taking the maximum pitch centre over all occurrences of a pattern, I hope to isolate either unusually high, or unusually low occurrences.

monophonise

Started, last checked	24/6/2013, 24/6/2013
Location	Musical properties
Calls	add-to-nth, constant-vector, dataset-restricted-to-m-in-nth, max-item, nth-list-of-lists, sky-line-clipped, top-line-verbose, translation
Called by	
Comments/see also	

Example:

```
(monophonise
 '((13 55 3 1) (13 60 2 0) (13 64 1 0) (14 55 2 0)
   (15 55 1/2 1) (15 59 1/2 1) (15 65 1/2 0)
   (15 55 1/2 0))
   4 0 1 2 3 "top-line-verbose")
--> ((13 64 1 0) (14 55 2 0) (15 65 1/2 0) (17 55 3 1)
     (19 59 1/2 1))
(monophonise
 '((13 55 3 1) (13 60 2 0) (13 64 1 0) (14 55 2 0)
   (15 55 1/2 1) (15 59 1/2 1) (15 65 1/2 0)
   (15 55 1/2 0))
   4 0 1 2 3 "sky-line-clipped")
--> ((13 64 1 0) (14 55 1 0) (15 55 1/2 0) (17 55 2 1)
     (19 59 1/2 1))
```

This function segments the input dataset into different datasets depending on the value in the staff index. For each distinct ontime in each dataset, it returns the datapoint (all provided dimensions returned) with maximum pitch as a member of a list. It translates (or 'unfolds') datapoints belonging to successive staves, so that for instance none are overlapping in generated MIDI files.

The mapping to maximum pitch is done in one of two ways, depending on the variable monophonise-fn. If set to sky-line-clipped, it applies this function, clipping any within-voice overlapping notes so that each line is strictly monophonic. If set to top-line-verbose, it applies this function, where within-voice overlapping notes are still permitted.

pitch-centre

Started, last checked	19/10/2009, 19/10/2009
Location	Musical properties
Calls	mean, nth-list-of-lists
Called by	
Comments/see also	

Example:

```
(pitch-centre
 '(60 61 62) '((0 60) (1 61) (2 62) (3 48) (3 57)))
--> 17/5
```

Pitch centre is defined as 'the absolute distance, in semitones, of the mean pitch of a [pattern]...from the mean pitch of the dataset' (Pearce and Wiggins, 2007, p. 78). By taking the maximum pitch centre over all occurrences of a pattern, I hope to isolate either unusually high, or unusually low occurrences.

pitch-range

Started, last checked	19/10/2009, 19/10/2009
Location	Musical properties
Calls	nth-list-of-lists, range
Called by	
Comments/see also	

Example:

```
(pitch-range '((0 60) (1 61) (3 62)) 1)
--> 2
```

Pitch range is the range in semitones of a pattern.

restn

Started, last checked	19/10/2009, 19/10/2009
Location	Musical properties
Calls	
Called by	rhythmic-density, top-line, top-line-verbose
Comments/see also	

Example:

```
(restn '((13 55) (13 60) (13 64) (14 55) (15 55)) 3)
--> ((14 55) (15 55))
```

Applies the function rest n times.

rhythmic-density

Started, last checked	19/10/2009, 19/10/2009
Location	Musical properties
Calls	density, my-last, restn
Called by	
Comments/see also	

Example:

```
(rhythmic-density
 '((13 55) (13 60) (13 64) (27/2 63) (14 55) (17 48)))
--> 6/5
```

The rhythmic density of a pattern is defined as 'the mean number of events per tactus beat' (Pearce and Wiggins, 2007, p. 78). See the function density for further definitions.

rhythmic-variability

Started, last checked	19/10/2009, 19/10/2009
Location	Musical properties
Calls	nth-list-of-lists, sd
Called by	
Comments/see also	

Example:

```
(rhythmic-variability
 '((0 64 1) (1 55 1/2) (1 65 1) (2 55 1/2) (2 72 1/3)
   (3 55 1) (4 55 2) (5 55 1/2) (5 60 1)
   (6 59 1/3) (6 67 1/2)) 2)
--> 0.5354223
```

The rhythmic variability of a pattern is defined as 'the degree of change in note duration (i.e., the standard deviation of the log of the event durations)' (Pearce and Wiggins, 2007, p. 78). The intuition is that patterns with much rhythmic variation are likely to be noticeable.

sky-line-clipped

Started, last checked	23/6/2013, 23/6/2013
Location	Musical properties
Calls	nth-list-of-lists, replace-nth-in-list-with-x
Called by	monophonise
Comments/see also	top-line, top-line-verbose

Example:

```
(sky-line-clipped
 '((3 50 53 2 1) (5 44 55 5 1) (5 52 55 5 1)
   (7 45 49 2 1) (9 50 53 2 1) (9 54 56 2 1)
   (10.5 40 50 1 1) (10.5 50 52 1 1)))
--> ((3 50 53 2 1) (5 52 55 4 1) (9 54 56 1 1))
(sky-line-clipped
 '((0 52 55 0.5 1) (0.25 76 69 0.5 0)
   (0.5 54 56 0.5 1) (0.75 75 68 0.5 0)
   (1 56 57 0.5 1) (1.25 74 68 0.5 0)))
--> ((0 52 55 0.25 1) (0.25 76 69 0.5 0)
     (0.75 75 68 0.5 0) (1.25 74 68 0.5 0))
```

This function returns the clipped skyline of an input point set. Generally this is the highest note at each unique onset, unless the current highest note is still sounding when a new lower note begins (in which case the new lower note is ignored), or the current highest note is still sounding when a new higher note begins (in which case the new higher note is included in the output, and the previous note's duration is clipped to this ontime). It is assumed that the input point set is in lexicographic order.

small-intervals

Started, last checked	19/10/2009, 19/10/2009
Location	Musical properties
Calls	spacing-items, top-line
Called by	
Comments/see also	intervallic-leaps

Example:

```
(small-intervals
 '((13 57) (13 60) (13 62) (14 57) (15 57) (15 59)
   (15 63) (16 57) (17 67) (18 57) (19 57) (22 57)
   (23 60) (24 57) (24 59) (25 57) (25 64)))
--> 3
```

The small intervals variable counts the number of such intervals present in the melody line of a pattern, the intuition being that scalic, static or stepwise melodies may be rated as more noticeable or important. As sometimes the melody is not obvious in polyphonic music, I use a 'top-line' rule: at each of the pattern's distinct ontimes there will be at least one datapoint present. At this ontime the melody takes the value of the maximum morphetic pitch number present.

top-line

Started, last checked	19/10/2009, 19/10/2009
Location	Musical properties
Calls	cons-ith-while-jth-constantp, max-item, restn
Called by	intervallic-leaps, small-intervals
Comments/see also	sky-line-clipped, top-line-verbose

Example:

```
(top-line
 '((13 55) (13 60) (13 64) (14 55) (15 55) (15 59)
   (15 65) (16 55) (17 72) (18 55) (19 55) (22 55)
   (23 60) (24 55) (24 59) (25 55) (25 67)) 1)
--> (64 55 65 55 72 55 55 55 60 59 67)
```

For each distinct ontime, this function returns the maximum pitch as a member of a list.

top-line-verbose

Started, last checked	19/10/2009, 19/10/2009
Location	Musical properties
Calls	cons-ith-while-jth-constantp, max-nth-argmax, restn
Called by	intervallic-leaps, small-intervals
Comments/see also	sky-line-clipped, top-line

Example:

```
(top-line-verbose
 '((13 55) (13 60) (13 64) (14 55) (15 55) (15 59)
   (15 65) (16 55) (17 72) (18 55) (19 55) (22 55)
   (23 60) (24 55) (24 59) (25 55) (25 67)) 1)
--> ((13 64) (14 55) (15 65) (16 55) (17 72) (18 55)
     (19 55) (22 55) (23 60) (24 59) (25 67))
```

For each distinct ontime, this function returns the datapoint (all provided dimensions returned) with maximum pitch as a member of a list.

top-line-verbose-top-staff

Started, last checked	19/10/2009, 19/10/2009
Location	Musical properties
Calls	dataset-restricted-to-m-in-nth, nth-list-of-lists, restn
Called by	intervallic-leaps, small-intervals
Comments/see also	

Example:

```
(top-line-verbose-top-staff
 '((13 55 2 1) (13 60 2 0) (13 64 1 0) (14 55 1 0)
   (15 55 1/2 1) (15 59 1/2 1) (15 65 1/2 0)
   (15 55 1/2 0))
   1 3)
--> ((13 64 1 0) (14 55 1 0) (15 65 1/2 0))
```

This function is very similar to the function monophonise. It extracts datapoints occurring in the lowest-numbered staff, and applies the function top-line-verbose.

4.3.3 Empirical preliminaries

These functions make it possible to form empirical n-dimensional distributions. One of the applications of these empirical distributions is to adapt pattern interest (Conklin and Bergeron, 2008) for polyphonic music.

accumulate-to-mass

Started, last checked	20/10/2009, 20/10/2009
Location	Empirical preliminaries
Calls	
Called by	present-to-mass
Comments/see also	add-to-mass

Example:

```
(accumulate-to-mass
 '(6 72) '((6 72) 1/4)
 '(((6 72) 1/4) ((4 0.1) 1/2)) 1/4)
--> (((6 72) 1/2) ((4 0.1) 1/2))
```

This function takes four arguments: a datapoint **d**; an element (to be updated) of the emerging empirical probability mass function L; L itself is the third argument; and the fourth argument is μ, the reciprocal of the number of datapoints that have been observed. This function has been called because **d** is new to the empirical mass—it is added with mass μ.

add-to-mass

Started, last checked	20/10/2009, 20/10/2009
Location	Empirical preliminaries
Calls	
Called by	present-to-mass
Comments/see also	accumulate-to-mass

Example:

```
(add-to-mass '(6 72) '(((4 0.1) 2/3)) 1/3)
--> (((6 72) 1/3) ((4 0.1) 2/3))
```

This function takes three arguments: a datapoint **d**; an emerging empirical probability mass function L; and the third argument is μ, the reciprocal of the number of datapoints that have been observed. This function has been called because **d** already forms part λ of the mass. This element is increased to $\lambda + \mu$.

direct-product-of-n-sets

Started, last checked	20/10/2009, 20/10/2009
Location	Empirical preliminaries
Calls	direct-product-of-two-sets
Called by	likelihood-of-pattern-or-translation, likelihood-of-translations-geometric-mean
Comments/see also	

Example:

```
(direct-product-of-n-sets
 '((1 2) ((59) (60)) (-4 -2)))
--> ((1 59 -4) (1 59 -2) (1 60 -4) (1 60 -2) (2 59 -4)
     (2 59 -2) (2 60 -4) (2 60 -2)).
```

This function takes a single argument (assumed to be a list of sets of numbers or sets of sets), and returns the direct product of these sets.

direct-product-of-two-sets

Started, last checked	20/10/2009, 20/10/2009
Location	Empirical preliminaries
Calls	
Called by	direct-product-of-n-sets
Comments/see also	

Example:

```
(direct-product-of-two-sets '(1/3 1 2) '(59 60))
--> ((1/3 59) (1/3 60) (1 59) (1 60) (2 59) (2 60))
```

This function takes two arguments (assumed to be sets of numbers or sets of sets), and returns the direct product of these sets.

empirical-mass

Started, last checked	20/10/2009, 20/10/2009
Location	Empirical preliminaries
Calls	present-to-mass
Called by	likelihood-of-pattern-or-translation, likelihood-of-translations-geometric-mean
Comments/see also	

Example:

```
(empirical-mass '((4 0) (4 0) (0 4)) '())
--> (((0 4) 1/3) ((4 0) 2/3))
```

This function returns the empirical probability mass function L for a dataset listed $\mathbf{d}_1^*, \mathbf{d}_2^*, \ldots, \mathbf{d}_n^*$.

events-with-these-ontime-others

Started, last checked	20/10/2009, 20/10/2009
Location	Empirical preliminaries
Calls	events-with-this-ontime-other, index-1st-sublist-item>=, nth-list-of-lists, my-last
Called by	
Comments/see also	

Example:

```
(events-with-these-ontime-others
 '((6 63) (7 96) (9 112))
 '((23/4 86 1/4 2 46) (6 55 1/2 1 37)
   (6 63 1/3 1 37) (6 63 1/2 2 34) (7 91 1 1 56)
   (7 96 1/2 1 73) (7 96 1 1 95) (7 108 3/2 2 50)
   (17/2 109 1/2 2 49) (9 95 1 1 71)
   (9 98 1 1 71) (9 102 1 1 71) (9 112 3/4 2 73)) 1 2)
--> ((6 1/3) (6 1/2) (7 1/2) (7 1) (9 3/4))
```

The first argument to this function is a pattern, under the projection of ontime and MIDI note number (in which case the variable other-index is 1) or morphetic pitch (in which case other-index is 2). The corresponding members of the full dataset are sought out and returned as ontime-other pairs.

events-with-this-ontime-other

Started, last checked	20/10/2009, 20/10/2009
Location	Empirical preliminaries
Calls	
Called by	events-with-these-ontime-others
Comments/see also	

Example:

```
(events-with-this-ontime-other
 '(7 96)
 '((23/4 86 1/4 2 46) (6 55 1/2 1 37)
   (6 63 1/3 1 37) (6 63 1/2 2 34) (7 91 1 1 56)
   (7 96 1/2 1 73) (7 96 1 1 95) (7 108 3/2 2 50)
   (17/2 109 1/2 2 49) (9 95 1 1 71)
   (9 98 1 1 71) (9 102 1 1 71) (9 112 3/4 2 73)) 1 2)
--> ((7 1/2) (7 1))
```

The first argument to this function is a datapoint, under the projection of ontime and MIDI note number (in which case the variable other-index is 1) or morphetic pitch (in which case other-index is 2). The corresponding members of the full dataset are sought out and returned as ontime-other pairs.

likelihood-of-pattern-or-translation

Started, last checked	20/10/2009, 20/10/2009
Location	Empirical preliminaries
Calls	constant-vector, direct-product-of-n-sets, empirical-mass, likelihood-of-subset, orthogonal-projection-not-unique-equalp, potential-n-dim-translations
Called by	
Comments/see also	likelihood-of-translations-geometric-mean

Example:

```
(likelihood-of-pattern-or-translation
 '((0 60 60 1) (1 62 61 1/2) (2 64 62 1/3)
   (3 60 60 1))
```

```
 '((0 60 60 1) (1 62 61 1/2) (2 64 62 1/3) (3 60 60 1)
   (4 62 61 1) (5 64 62 1/2) (6 66 63 1/3) (7 62 61 1)
   (8 69 65 3) (11 59 59 1) (12 60 60 1)))
--> 9/14641 + 4/14641 = 13/14641

(likelihood-of-pattern-or-translation
 '((0 60 1) (1 61 1) (2 62 1) (3 60 1))
 '((0 60 1) (1 61 1) (1 66 1/2) (3/2 67 1/2) (2 62 1)
   (2 68 1) (5/2 66 1/2) (3 60 1)))
--> 1/4*1/8*1/8*1/4 = 1/1024

(likelihood-of-pattern-or-translation
 '((0 60) (1 61) (2 62) (3 60))
 '((0 60) (1 61) (1 66) (3/2 67) (2 62)
   (2 68) (5/2 66) (3 60)))
--> 1/4*1/8*1/8*1/4 + 1/4*1/8*1/8*1/4 = 1/512

(likelihood-of-pattern-or-translation
 '((0 1) (1 1) (2 1) (3 1))
 '((0 1) (1 1) (1 1/2) (3/2 1/2) (2 1)
   (2 1/2) (5/2 1/2) (3 1)))
--> 1/16 + 1/16 = 1/8
```

This function takes a pattern and the dataset in which the pattern occurs. It calculates the potential translations of the pattern in the dataset and returns the sum of their likelihoods.

likelihood-of-subset

Started, last checked	20/10/2009, 20/10/2009
Location	Empirical preliminaries
Calls	
Called by	likelihood-of-pattern-or-translation
Comments/see also	likelihood-of-subset-geometric-mean

Example:

```
(likelihood-of-subset
 '((60 60 1) (62 61 1/2) (64 62 1/3) (60 60 1))
 '(((60 60 1) 3/11) ((62 61 1/2) 1/11)
   ((64 62 1/3) 1/11) ((62 61 1) 2/11)
```

```
   ((64 62 1/2) 1/11) ((66 63 1/3) 1/11)
   ((69 65 3) 1/11) ((59 59 1) 1/11)))
--> 9/14641
```

This function takes a pattern-palette and the empirical mass for the dataset-palette in which the pattern occurs. The product of the individual masses is returned, and reverts to zero if any pattern points do not occur in the empirical mass.

likelihood-of-subset-geometric-mean

Started, last checked	20/10/2009, 20/10/2009
Location	Empirical preliminaries
Calls	
Called by	likelihood-of-translations-geometric-mean
Comments/see also	likelihood-of-subset

Example:

```
(likelihood-of-subset-geometric-mean
 '((60 60 1) (62 61 1/2) (64 62 1/3) (60 60 1)) 1/4
 '(((60 60 1) 3/11) ((62 61 1/2) 1/11)
   ((64 62 1/3) 1/11) ((62 61 1) 2/11)
   ((64 62 1/2) 1/11) ((66 63 1/3) 1/11)
   ((69 65 3) 1/11) ((59 59 1) 1/11)))
--> 0.1574592
```

This function takes a pattern-palette, the reciprocal length of that pattern, and the empirical mass for the dataset-palette in which the pattern occurs. The geometric mean of the individual masses is returned, and reverts to zero if any pattern points do not occur in the empirical mass.

likelihood-of-translations-geometric-mean

Started, last checked	20/10/2009, 20/10/2009
Location	Empirical preliminaries
Calls	constant-vector, direct-product-of-n-sets, empirical-mass, likelihood-of-subset-geometric-mean, orthogonal-projection-not-unique-equalp, potential-n-dim-translations, translation
Called by	
Comments/see also	

Example:

```
(likelihood-of-translations-geometric-mean
 '((0 60 60 1) (1 62 61 1/2) (2 64 62 1/3)
   (3 60 60 1))
 '((0 60 60 1) (1 62 61 1/2) (2 64 62 1/3) (3 60 60 1)
   (4 62 61 1) (5 64 62 1/2) (6 66 63 1/3) (7 62 61 1)
   (8 69 65 3) (11 59 59 1) (12 60 60 1)))
--> (9/14641)^(1/4) + (4/14641)^(1/4) = 0.2860241

(likelihood-of-translations-geometric-mean
 '((0 60 1) (1 61 1) (2 62 1) (3 60 1))
 '((0 60 1) (1 61 1) (1 66 1/2) (3/2 67 1/2) (2 62 1)
   (2 68 1) (5/2 66 1/2) (3 60 1)))
--> (1/4*1/8*1/8*1/4)^(1/4) = 0.17677668

(likelihood-of-translations-geometric-mean
 '((0 60) (1 61) (2 62) (3 60))
 '((0 60) (1 61) (1 66) (3/2 67) (2 62)
   (2 68) (5/2 66) (3 60)))
--> (1/4*1/8*1/8*1/4)^(1/4) + (1/4*1/8*1/8*1/4)^(1/4)
 = 0.35355335

(likelihood-of-translations-geometric-mean
 '((0 1) (1 1) (2 1) (3 1))
 '((0 1) (1 1) (1 1/2) (3/2 1/2) (2 1)
   (2 1/2) (5/2 1/2) (3 1)))
--> (1/16)^(1/4) + (1/16)^(1/4) = 1.
```

This function takes a pattern and the dataset in which the pattern occurs. It calculates the potential translations of the pattern in the dataset and returns the sum of the geometric means of their likelihoods.

Note that this is not really a likelihood, as it is possible for probabilities to be greater than 1.

likelihood-of-translations-reordered

Started, last checked	20/10/2009, 20/10/2009
Location	Empirical preliminaries
Calls	constant-vector, direct-product-of-n-sets, likelihood-of-subset, orthogonal-projection-not-unique-equalp, potential-n-dim-translations, translation
Called by	evaluate-variables-of-pattern2hash
Comments/see also	

Example:

```
(likelihood-of-translations-reordered
 '((0 60 60 1) (1 62 61 1/2) (2 64 62 1/3)
   (3 60 60 1))
 '((60 60 1) (62 61 1/2) (64 62 1/3) (60 60 1)
   (62 61 1) (64 62 1/2) (66 63 1/3) (62 61 1)
   (69 65 3) (59 59 1) (60 60 1))
 '(((60 60 1) 3/11) ((59 59 1) 1/11) ((69 65 3) 1/11)
   ((62 61 1) 2/11) ((66 63 1/3) 1/11)
   ((64 62 1/2) 1/11) ((64 62 1/3) 1/11)
   ((62 61 1/2) 1/11)))
--> 9/14641 + 4/14641 = 13/14641
```

This function takes a pattern and the dataset in which the pattern occurs. It calculates the potential translations of the pattern in the dataset and returns the sum of their likelihoods. Note the order (and mandate) of the arguments is different to the original version of this function, which is called likelihood-of-pattern-or-translation.

potential-1-dim-translations

Started, last checked	20/10/2009, 20/10/2009
Location	Empirical preliminaries
Calls	constant-vector, nth-list-of-lists, orthogonal-projection-unique-equalp
Called by	potential-n-dim-translations
Comments/see also	

Example:

```
(potential-1-dim-translations
 '(60 60 1)
 '((60 60 1) (62 61 1/2) (64 62 1/3) (60 60 1)
   (62 61 1) (64 62 1/2) (66 63 1/3) (62 61 1)
   (69 65 3) (59 59 1) (60 60 1)) 0)
--> (-1 0 2 4 6 9)
```

This function takes three arguments, the first member of a pattern palette, the dataset palette and an index i. First of all, the dataset is projected uniquely along the dimension of index, creating a vector $\mathbf{u}$. Then the ith member of the first-pattern-palette is subtracted from each member of $\mathbf{u}$, giving a list of potential translations along this dimension.

potential-n-dim-translations

Started, last checked	20/10/2009, 20/10/2009
Location	Empirical preliminaries
Calls	add-to-list, first-n-naturals, potential-1-dim-translations
Called by	likelihood-of-pattern-or-translation, likelihood-of-translations-geometric-mean
Comments/see also	

Example:

```
(potential-n-dim-translations
 '(60 60 1)
 '((60 60 1) (62 61 1/2) (64 62 1/3) (60 60 1)
   (62 61 1) (64 62 1/2) (66 63 1/3) (62 61 1)
   (69 65 3) (59 59 1) (60 60 1)))
--> ((-1 0 2 4 6 9) (-1 0 1 2 3 5) (-2/3 -1/2 0 2))
```

This function takes two arguments, the first member of a pattern palette, the dataset palette and an index. The function potential-n-dim-translations is applied recursively to an increment.

present-to-mass

Started, last checked	20/10/2009, 20/10/2009
Location	Empirical preliminaries
Calls	accumulate-to-mass, add-to-mass
Called by	empirical-mass
Comments/see also	

Example:

```
(present-to-mass '(0 4) '(((4 0) 2/3)) 1/3)
--> (((0 4) 1/3) ((4 0) 2/3))
```

This function takes three arguments: a datapoint $\mathbf{d}$, an empirical probability mass function L which is in the process of being calculated, and μ, the reciprocal of the number of datapoints that have been observed. If $\mathbf{d}$ is new to the empirical mass, it is added with mass μ, and if it already forms part λ of the mass, then this component is increased to $\lambda + \mu$.

4.3.4 Evaluation heuristics

These functions implement definitions of coverage, compactness, and compression ratio (Meredith et al., 2003; Forth and Wiggins, 2009).

compactness

Started, last checked	13/1/2010, 13/1/2010
Location	Evaluation heuristics
Calls	index-item-1st-occurs, my-last
Called by	
Comments/see also	compact-subpatterns-more-output

Example:

```
(compactness
 '((1 2) (2 4)) '((1 2) (2 -1) (2 4) (3 6) (5 2))
 0.2 1 "straight down")
--> 2/3
```

The ratio of the number of points in the pattern to the number of points in the region spanned by the pattern. Both pattern and dataset are assumed to be sorted ascending. At present the only admissible definition of region is 'straight down' (which means 'lexicographic', cf. Def. 2.10 in Collins, 2011).

compactness-max

Started, last checked	13/1/2010, 13/1/2010
Location	Evaluation heuristics
Calls	compactness-min-max, translation
Called by	heuristics-pattern-translators-pair
Comments/see also	

Example:

```
(compactness-max
 '((1 2) (2 4)) '((0 0) (1 2) (3 -2))
 '((1 2) (2 -1) (2 0) (2 4) (3 0) (3 1) (3 3) (3 6)
   (4 0) (5 1) (5 2))
 0.2 1 "straight down" 2)
--> 2/3
```

The function compactness is applied to each occurrence of a pattern and the maximum compactness returned.

compactness-min-max

Started, last checked	13/1/2010, 13/1/2010
Location	Evaluation heuristics
Calls	index-item-1st-occurs, my-last
Called by	compactness-max
Comments/see also	

Example:

```
(compactness-max
 '((1 2) (2 4)) '((0 0) (1 2) (3 -2))
 '((1 2) (2 -1) (2 0) (2 4) (3 0) (3 1) (3 3) (3 6)
   (4 0) (5 1) (5 2))
 0.2 1 "straight down" 2)
--> 2/3
```

The function compactness is applied to each occurrence of a pattern and the maximum compactness returned.

compression-ratio

Started, last checked	13/1/2010, 13/1/2010
Location	Evaluation heuristics
Calls	coverage
Called by	heuristics-pattern-translators-pair
Comments/see also	

```
(compactness-min-max
 '((1 2) (2 4)) '((1 2) (2 -1) (2 4) (3 6) (5 2))
 0.2 1 "straight down")
--> 2/3
```

The ratio of the number of points in the pattern to the number of points in the region spanned by the pattern. Both pattern and dataset are assumed to be sorted ascending. At present the only admissible definition of region is 'straight down' (which means 'lexicographic', cf. Def. 2.10 in Collins, 2011).

cover-ratio

Started, last checked	13/1/2010, 13/1/2010
Location	Evaluation heuristics
Calls	coverage
Called by	heuristics-pattern-translators-pair
Comments/see also	

Example:

```
(cover-ratio
 '((1 2) (2 4)) '((0 0) (1 2))
 '((1 2) (2 4) (3 6) (5 2) (6 1)) 0.2 t t)
--> 3/5
```

The ratio between the number of uncovered datapoints in the dataset that are members of occurrences of the pattern, to the total number of uncovered datapoints in the dataset.

coverage

Started, last checked	13/1/2010, 13/1/2010
Location	Evaluation heuristics
Calls	intersection-multidimensional, translations, unions-multidimensional-sorted-asc
Called by	compression-ratio, cover-ratio, heuristics-pattern-translators-pair
Comments/see also	coverage-mod-2nd-n

Example:

```
(coverage
 '((1 2) (2 4)) '((0 0) (1 2))
 '((1 2) (2 4) (3 6) (5 2) (6 1)) t t)
--> 3
```

The number of datapoints in the dataset that are members of occurrences of the pattern.

coverage-mod-2nd-n

Started, last checked	13/1/2010, 13/1/2010
Location	Evaluation heuristics
Calls	intersection-multidimensional, translations-mod-2nd-n, unions-multidimensional-sorted-asc
Called by	compression-ratio, cover-ratio, heuristics-pattern-translators-pair
Comments/see also	coverage

Example:

```
(coverage-mod-2nd-n
 '((1 2) (2 4)) '((0 0) (1 2))
 '((1 2) (2 4) (3 6) (5 2) (6 1)) 12 t t)
--> 3
```

The number of datapoints in the dataset that are members of occurrences of the pattern. Translations are carried out modulo the fourth argument.

heuristics-pattern-translators-pair

Started, last checked	13/1/2010, 13/1/2010
Location	Evaluation heuristics
Calls	compactness-max, compression-ratio, cover-ratio, coverage
Called by	heuristics-pattern-translators-pairs
Comments/see also	

Example:

```
(heuristics-pattern-translators-pair
 '((1 2) (2 4)) '((0 0) (1 2) (3 -2))
 '((1 2) (2 -1) (2 0) (2 4) (3 0) (3 1) (3 3) (3 6)
   (4 0) (5 1) (5 2)) '(t t t t t t)
   0.2 0.25 1 0.25 1 "straight down" 11)
--> (5 5/11 1 2/3 2 3)
```

A pattern and its translators in a projected dataset are supplied as arguments to this function, along with an indicator vector that indicates which heuristics out of coverage, cover ratio, compression ratio, compactness, $|P|$ and $|T(P, D)|$ should be calculated.

heuristics-pattern-translators-pairs

Started, last checked	13/1/2010, 13/1/2010
Location	Evaluation heuristics
Calls	heuristics-pattern-translators-pair
Called by	musicological-heuristics
Comments/see also	

Example:

```
(heuristics-pattern-translators-pairs
 '((((1 2) (2 4)) ((0 0) (1 2) (3 -2)))
   (((1 2) (2 0)) ((0 0) (2 0))))
 '((1 2) (2 -1) (2 0) (2 4) (3 0) (3 1) (3 2) (3 6)
   (4 0) (5 1) (5 2)) '(t t t t t t)
   0.2 0.25 1 0.25 1 "straight down" 11)
--> ((5 5/11 1 2/3 2 3) (4 4/11 1 2/3 2 2))
```

The function heuristics-pattern-translators-pair is applied recursively to pairs of pattern-translators.

musicological-heuristics

Started, last checked	13/1/2010, 13/1/2010
Location	Evaluation heuristics
Calls	heuristics-pattern-translators-pairs, normalise-0-1
Called by	
Comments/see also	

Example:

```
(musicological-heuristics
 '((((1 2) (2 4)) ((0 0) (1 2) (3 -2)))
   (((1 2) (2 0)) ((0 0) (2 0)))
   (((1 2) (2 4) (4 0)) ((0 0) (1 2) (2 -4))))
 '((1 2) (2 -1) (2 0) (2 4) (3 -2) (3 0) (3 1)
   (3 2) (3 6) (4 0) (5 1) (5 2) (6 -4))
 0.25 1 0.25 1 "straight down" 11)
--> ((1 1 1) (1 1 0))
```

The function heuristics-pattern-translators-pairs is applied to pattern-translator pairs with the heuristics indicator set to compression ratio and compactness (max). The values are normalised (linearly) to $[0, 1]$ and returned as two lists.

4.4 Pattern discovery

4.4.1 Structural induction mod

The functions below include two early implementations of SIA (Structure induction algorithm, Meredith et al., 2002), one version working modulo n.

assoc-files

Started, last checked	13/3/2013, 13/3/2013
Location	Structural induction mod
Calls	read-from-file
Called by	SIA-reflected, SIA-reflected-mod-2nd-n
Comments/see also	

Example:

```
(assoc-files
 '(2 (2 6)) nil 1 1
 (merge-pathnames
  (make-pathname
   :directory '(:relative "Racchman-Oct2010 example")
   :name "initial-states" :type "txt")
  *MCStylistic-Aug2013-example-files-results-path*))
--> (1
     ((2 (2 6))
      (NIL NIL "C-17-4"
       ((1 57 58 1 1 2 0) (1 59 59 1 1 2 1)
        (1 65 63 1 1 2 2)))))
```

The arguments to this function are a probe, a path&name, and a positive integer. The integer indicates how many files there are with the specified path&name. Each one, assumed to contain an assoc- list, is read in turn and probed for the presence of the argument in probe. If it is present the relevant row is returned.

add-two-lists-mod-2nd-n

Started, last checked	15/1/2010, 15/1/2010
Location	Structural induction mod
Calls	add-two-lists
Called by	maximal-translatable-pattern-mod-2nd-n, test-equal<potential-translator-mod-2nd-n, translation-mod-2nd-n
Comments/see also	

Example:

```
(add-two-lists-mod-2nd-n '(4 2 -3) '(8 60 -3) 12)
```

```
--> (12 2 -6)
```

Adds two lists element-by-element, treating the second elements of each list modulo n.

check-potential-translators-mod-2nd-n

Started, last checked	15/1/2010, 15/1/2010
Location	Structural induction mod
Calls	read-from-file
Called by	translators-of-pattern-in-dataset-mod-2nd-n
Comments/see also	check-potential-translators

Example:

```
(check-potential-translators-mod-2nd-n
 '(3 4) '((0 0) (1 2) (1 5) (2 9))
 '((0 0) (3 4) (4 9) (5 1)) 12)
--> ((0 0) (1 5) (2 9))
```

This function is very similar to the function check-potential-translators. The difference is that the translation of the 2nd element is being carried out modulo n.

dataset-restricted-to-m-in-nth

Started, last checked	15/1/2010, 15/1/2010
Location	Structural induction mod
Calls	
Called by	
Comments/see also	

Example:

```
(dataset-restricted-to-m-in-nth
 '((12 41 49 1 1) (12 81 72 2 0) (13 53 56 1 1)
   (14 55 57 1 1) (14 74 68 2 0) (15 43 50 1 1)
   (16 36 46 2 1) (16 72 67 1/2 0)) 1 4)
--> ((12 41 49 1 1) (13 53 56 1 1) (14 55 57 1 1)
     (15 43 50 1 1) (16 36 46 2 1))
```

This function acts on a list of sublists. The nth item of each sublist is tested for equality (equalp) with the second argument. If it is equal it is retained, otherwise it is not included in the output.

indices-of-matrix-passing-tests

Started, last checked	15/1/2010, 15/1/2010
Location	Structural induction mod
Calls	test-all-true
Called by	occurrence-matrix&rel-idx
Comments/see also	

Example:

```
(indices-of-matrix-passing-tests
 '((2 4 -1 6 9) (0 0 4 2 -7) (-3 -2 -1 4 1)
   (2 4 -1 3.3 9) (0 0 4 6.8 -7)) (list #'>= #'<)
   '(3.9 6.8))
--> ((0 1) (0 3) (1 2) (2 3) (3 1) (4 2))
```

The first argument to this function is a matrix in list representation (a list of sublists, where each sublist correspond to a row, and the jth item of each sublist to the jth column). The tests specified in the second argument are applied for constants specified in the third argument. If an element of the matrix passes all tests, its index in (i, j) form is appended to the output list.

maximal-translatable-pattern-mod-2nd-n

Started, last checked	15/1/2010, 15/1/2010
Location	Structural induction mod
Calls	add-two-lists-mod-2nd-n, test-equal<list-elements
Called by	
Comments/see also	As with maximal-translatable-pattern, the implementation could be improved.

Example:

```
(maximal-translatable-pattern-mod-2nd-n
 '(2 0) '((0 0) (1 1) (1 2) (2 0) (2 5) (3 1)) 12)
--> ((0 0) (1 1))
```

This function computes the maximal translatable pattern of an arbitrary vector $\mathbf{u}$, searching in some dataset D, and treating the second element of each datapoint modulo n.

mod-column

Started, last checked	15/1/2010, 15/1/2010
Location	Structural induction mod
Calls	firstn, lastn
Called by	test-translation-mod-2nd-n-no-length-check
Comments/see also	

Example:

```
(mod-column
 '((1 2 12 4) (1 2 16 -1) (2 4 32 6) (5 2 50 6)) 7 2)
--> ((1 2 5 4) (1 2 2 -1) (2 4 4 6) (5 2 1 6))
```

The first argument to this function is a list, assumed to contain sublists of equal length. The second argument specifies what modulo will be calculated for the nth item of each sublist, where n is given by the third argument.

mod-list

Started, last checked	13/3/2013, 13/3/2013
Location	Structural induction mod
Calls	
Called by	segments2MNNs-mod12
Comments/see also	

Example:

```
(mod-list '(1 2 3 4 5 7) 3)
--> (1 2 0 1 2 1)
```

This function gives the value of each item of a list, modulo b.

restrict-dataset-in-nth-to-xs

Started, last checked	15/1/2010, 15/1/2010
Location	Structural induction mod
Calls	
Called by	
Comments/see also	

Example:

```
(restrict-dataset-in-nth-to-xs
 '((2 4 -1 6 9) (0 0 4 2 -7) (-3 -2 -1 -1 1)
   (12 0 -7 5 3) (1 2 3 4 3) (1 2 5 4 5)
   (12 0 -6 5 4) (-3 -2 1 -1 0) (12 0 -7 5 4))
 3 '(4 5 6))
--> ((2 4 -1 6 9) (12 0 -7 5 3) (1 2 3 4 3)
     (1 2 5 4 5) (12 0 -6 5 4) (12 0 -7 5 4))
```

The first argument to this function is a dataset. We are interested in the nth dimension of each vector, where n is the second argument. A datapoint is retained in the output if its nth value is a member of the list specified by the third argument. Note it will not recognise 1.0 as 1.

restrict-dataset-in-nth-to-tests

Started, last checked	13/3/2013, 13/3/2013
Location	Structural induction mod
Calls	
Called by	
Comments/see also	

Example:

```
(restrict-dataset-in-nth-to-tests
 '((2 4 -1 6 9) (0 0 4 2 -7) (-3 -2 -1 4 1)
   (2 4 -1 3.3 9) (0 0 4 6.8 -7))
 3 (list #'>= #'<) '(3.9 6.8))
--> ((2 4 -1 6 9) (12 0 -7 5 3) (1 2 3 4 3)
     (1 2 5 4 5) (12 0 -6 5 4) (12 0 -7 5 4))
```

The first argument to this function is a dataset. We are interested in the nth dimension of each vector, where n is the second argument. A datapoint is retained in the output if its nth value is true compared with the each element of the third argument using the test supplied as each element of the fourth argument.

SIA-reflected

Started, last checked	13/3/2013, 13/3/2013
Location	Structural induction mod
Calls	assoc-files, subtract-two-lists, update-written-file, write-to-file
Called by	
Comments/see also	SIA-reflected-merge-sort for a more efficient implementation.

Example:

```
(SIA-reflected
 '((0 61) (0 65) (1 64) (4 62) (4 66) (5 65) (8 60)
   (8 64) (9 63) (12 56) (13 69) (15 65) (16 57)
   (16 59) (17 64) (19 63))
 (concatenate
  'string
  *MCStylistic-Oct2010-example-files-path*
  "/SIA output") 50)
--> 2
```

This function is a version of the SIA algorithm. It is called 'SIA-reflected' because the results (pairs of vectors and the corresponding MTPs) are the other way round to the algorithm specified by Meredith et al. (2002). The example causes two files to be created in the specified location.

SIA-reflected-mod-2nd-n

Started, last checked	13/3/2013, 13/3/2013
Location	Structural induction mod
Calls	assoc-files, subtract-two-lists-mod-2nd-n, update-written-file, write-to-file
Called by	
Comments/see also	SIA-reflected

Example:

```
(SIA-reflected-mod-2nd-n
 '((0 61) (0 65) (1 64) (4 62) (4 66) (5 65) (8 60)
   (8 64) (9 63) (12 56) (13 69) (15 65) (16 57)
   (16 59) (17 64) (19 63))
```

```
 12
 (merge-pathnames
  (make-pathname
   :name "SIA mod 2nd n output" :type "txt")
  *MCStylistic-Aug2013-example-files-results-path*)
 50)
--> 2
```

This function is a version of the SIA algorithm that works with a pitch representation modulo n. The example causes two files to be created in the specified location.

split-point-set-by-staff

Started, last checked	17/6/2014, 17/6/2014
Location	Structural induction mod
Calls	dataset-restricted-to-m-in-nth
Called by	melodic-interval-of-a
Comments/see also	

Example:

```
(split-point-set-by-staff
 '((13 55 3 1) (13 60 2 0) (13 64 1 0) (14 55 2 0)
   (15 55 1/2 1) (15 59 1/2 1) (15 65 1/2 0)
   (15 55 1/2 0)) 3)
--> (((13 60 2 0) (13 64 1 0) (14 55 2 0)
     (15 65 1/2 0) (15 55 1/2 0))
    ((13 55 3 1) (15 55 1/2 1) (15 59 1/2 1)))
```

This function splits the input point set into different point sets depending on the value in the staff index.

Unlike the function monophonise, this function does not create monophonic lines within staves, or translate (or 'unfold') point sets belonging to successive staves.

subtract-list-from-each-list-mod-2nd-n

Started, last checked	15/1/2010, 15/1/2010
Location	Structural induction mod
Calls	subtract-two-lists-mod-2nd-n
Called by	translators-of-pattern-in-dataset-mod-2nd-n
Comments/see also	subtract-list-from-each-list

Example:

```
(subtract-list-from-each-list-mod-2nd-n
 '((8 -2 -3) (4 6 6) (0 0 0) (4 7 -3)) '(4 7 -3) 12)
--> ((4 3 0) (0 11 9) (-4 5 3) (0 0 0))
```

The function subtract-two-lists-mod-2nd-n is applied recursively to each sub-list in the first list argument, and the second argument.

subtract-two-lists-mod-2nd-n

Started, last checked	15/1/2010, 15/1/2010
Location	Structural induction mod
Calls	subtract-two-lists
Called by	subtract-list-from-each-list-mod-2nd-n
Comments/see also	subtract-two-lists

Example:

```
(subtract-two-lists-mod-2nd-n '(8 60 1) '(4 67 2) 12)
--> (4 5 -1)
```

Subtracts the second list from the first, element-by-element. The subtraction of the second elements is performed modulo n, where n is the third argument to the function. It is assumed that the list is at least of length 2.

test-equal<potential-translator-mod-2nd-n

Started, last checked	15/1/2010, 15/1/2010
Location	Structural induction mod
Calls	add-two-lists-mod-2nd-n
Called by	check-potential-translators-mod-2nd-n
Comments/see also	test-equal<potential-translator

Example:

```
(test-equal<potential-translator-mod-2nd-n
 '((0 0) (3 4) (4 9) (5 1)) '(2 9) '(3 4) 12)
--> ((3 4))
```

This function is very similar to the function test-equal<potential-translator. The difference is the call to the function add-two-lists- mod-2nd-n (as opp- posed to calling add-two-lists), and this requires the inclusion of an extra argument.

test-translation-mod-2nd-n

Started, last checked	15/1/2010, 15/1/2010
Location	Structural induction mod
Calls	test-translation-mod-2nd-n-no-length-check
Called by	
Comments/see also	test-translation

Example:

```
(test-translation-mod-2nd-n
 '((2 2) (4 5)) '((11 9) (13 0)) 12)
--> T
```

This function is very similar to the function test-translation, except that here the translation in the second dimension is performed modulo the third argument.

test-translation-mod-2nd-n-no-length-check

Started, last checked	15/1/2010, 15/1/2010
Location	Structural induction mod
Calls	mod-column, sort-dataset-asc, subtract-two-lists-mod-2nd-n, translation-mod-2nd-n
Called by	test-translation-mod-2nd-n
Comments/see also	test-translation-no-length-check

Example:

```
(test-translation-mod-2nd-n-no-length-check
 '((40 0) (40 10) (43 7)) '((44 7) (44 9) (47 4)) 12)
--> T
```

This function ought to be very similar to the function test-translation-no-length-check. However simply altering the translation in the second dimension to modulo n (the third argument) can be problematic: In the above example, the pitch classes B♭, C, G are a translation of G, A, E, but when these are ordered modulo 12, the C and the B♭ swap positions. The function below accounts for this but will generally take longer to return an answer than test-translation-no-length-check.

translation-mod-2nd-n

Started, last checked	15/1/2010, 15/1/2010
Location	Structural induction mod
Calls	add-two-lists-mod-2nd-n
Called by	test-translation-mod-2nd-n-no-length-check, translations-mod-2nd-n
Comments/see also	translation

Example:

```
(translation-mod-2nd-n
 '((8 0 3) (9 11 1) (9 4 2)) '(3 3 0) 12)
--> ((11 3 3) (12 2 1) (12 7 2))
```

The first argument is a list of sublists, but we imagine it as a set of vectors (all members of the same n-dimensional vector space). The second argument—another list—is also an n-dimensional vector, and this is added to each of the members of the first argument. 'Added' means vector addition, that is element-wise, and addition in the second dimension is performed modulo the third argument. The resulting set is a translation of the first argument by the second.

translations-mod-2nd-n

Started, last checked	15/1/2010, 15/1/2010
Location	Structural induction mod
Calls	translation-mod-2nd-n
Called by	
Comments/see also	translations

Example:

```
(translations-mod-2nd-n
 '((8 0 3) (9 11 1) (9 4 2)) '((0 0 0) (3 3 0)) 12)
--> (((8 0 3) (9 4 1) (9 11 2))
     ((11 3 3) (12 2 1) (12 7 2)))
```

There are three arguments to this function, a pattern, some translators and a modulo argument. The pattern is translated by each translator, modulo n in the second dimension, and the results returned.

translators-of-pattern-in-dataset-mod-2nd-n

Started, last checked	15/1/2010, 15/1/2010
Location	Structural induction mod
Calls	check-potential-translators-mod-2nd-n, subtract-list-from-each-list-mod-2nd-n
Called by	
Comments/see also	As with translators-of-pattern-in-dataset, the implementation could be improved.

Example:

```
(translators-of-pattern-in-dataset-mod-2nd-n
 '((8 3) (8 7))
 '((4 7) (8 3) (8 4) (8 7) (9 3) (10 7)
   (11 3) (13 0) (13 4)) 12)
--> ((0 0) (5 9))
```

A pattern and dataset are provided. The transaltors of the pattern in the dataset are returned.

4.4.2 Structural induction merge

These functions implement SIA (Structural Induction Algorithm, Meredith et al., 2002) using a merge sort.

collect-by-car

Started, last checked	6/9/2010, 6/9/2010
Location	Structural induction merge
Calls	
Called by	collect-by-cars, collect-by-cars-partition
Comments/see also	

Example:

```
(collect-by-car
 '(((1 -14) 7/2 60) ((1 -14) 2 74) ((1 -2) 5/2 64)))
--> ((2 74))
```

A list is the only argument to this function. The car of the first element is compared with the cars of proceeding elements, and these proceeding elements are returned so long as there is equality.

collect-by-cars

Started, last checked	6/9/2010, 6/9/2010
Location	Structural induction merge
Calls	collect-by-car, restn
Called by	collect-by-cars-partition, SIA-reflected-merge-sort
Comments/see also	

Example:

```
(collect-by-cars
 '(((1/2 -14) 7/2 60) ((1/2 -10) 2 74)
   ((1/2 -2) 5/2 64) ((1/2 -2) 3 62) ((1/2 2) 1/2 67)
   ((1/2 2) 1 69) ((1/2 3) 3/2 71) ((1/2 14) 0 53)
   ((1 21) 2 74) ((1 21) 3 62) ((1 21) 4 46)
   ((1 -7) 3/2 71) ((1 -4) 5/2 64) ((1 4) 1/2 67)))
--> (((1/2 -14) (7/2 60)) ((1/2 -10) (2 74))
     ((1/2 -2) (5/2 64) (3 62))
     ((1/2 2) (1/2 67) (1 69))
     ((1/2 3) (3/2 71)) ((1/2 14) (0 53))
     ((1 21) (2 74) (3 62) (4 46))
     ((1 -7) (3/2 71)) ((1 -4) (5/2 64))
     ((1 4) (1/2 67)))
```

A list is the only argument to this function. The function collect-by-car is applied to each new vector appearing as the car of each element of the list.

collect-by-cars-partition

Started, last checked	6/9/2010, 6/9/2010
Location	Structural induction merge
Calls	collect-by-cars, write-to-file-append
Called by	SIA-reflected-merge-sort
Comments/see also	

Example:

```
(collect-by-cars-partition
 '((((1/2 -14) 7/2 60) ((1/2 -10) 2 74)
   ((1/2 -2) 5/2 64) ((1/2 -2) 3 62) ((1/2 2) 1/2 67)
   ((1/2 2) 1 69) ((1/2 3) 3/2 71) ((1/2 14) 0 53)
   ((1 21) 2 74) ((1 21) 3 62) ((1 21) 4 46)
   ((1 -7) 3/2 71) ((1 -4) 5/2 64) ((1 4) 1/2 67))
 (concatenate
  'string
  *MCStylistic-Oct2010-example-files-path*
  "/collected-by-cars.txt") 5)
--> NIL
```

The function collect-by-cars can cause a stack overflow for moderately sized lists. This function writes the output of collect-by-cars to a text file (using write-to-file-append) every so often to prevent stack overflow. The example causes a file to be created in the specified location.

SIA-reflected-merge-sort

Started, last checked	6/9/2010, 6/9/2010
Location	Structural induction merge
Calls	collect-by-cars-partition, subtract-two-lists, vector<vector-car
Called by	
Comments/see also	

Example: see Discovering and rating musical patterns (Sec. 3.3), especially lines 83-88).

This function is a faster version of the function SIA-reflected. The improved runtime is due to the use of merge-sort.

vector<vector-car

Started, last checked	6/9/2010, 6/9/2010
Location	Structural induction merge
Calls	vector<vector-t-or-nil
Called by	SIA-reflected-merge-sort
Comments/see also	vector<vector

Example:

```
(vector<vector-car '((1 1) . (1 3)) '((2 2) . (1 3)))
--> T
```

Applies the function vector<vector-t-or-nil to the car of each list (the two arguments).

vector<vector-t-or-nil

Started, last checked	6/9/2010, 6/9/2010
Location	Structural induction merge
Calls	
Called by	vector<vector-car
Comments/see also	vector<vector

Example:

```
(vector<vector-t-or-nil '(4 6 7) '(4 6 7.1))
--> T
```

The function vector<vector returns "equal" if the arguments were equal. This function returns nil in such a scenario.

4.4.3 Further structural induction algorithms

The functions below implement SIATEC (Structural Induction Algorithm for Transational Equivalence Classes) as described by Meredith et al. (2002), and COSIATEC (COvering Strucutral Induction Algorithm for Translational Equivalence Classes) as described by Forth and Wiggins (2009); Meredith et al. (2003).

COSIATEC

Started, last checked	25/1/2010, 25/1/2010
Location	Further structural induction algorithms
Calls	argmax-of-threeCs, read-from-file, remove-pattern-occurrences-from-dataset, SIA-reflected-for-COSIATEC, SIATEC, threeCs-pattern-translators-pairs, write-to-file
Called by	
Comments/see also	A more efficient implementation is required. See also COSIATEC-mod-2nd-n

Example:

```
(COSIATEC
 '((0 61) (0 65) (1 64) (4 62) (4 66) (5 65) (8 60)
   (8 64) (9 63) (12 56) (13 69) (15 65) (16 57)
   (16 59) (17 64) (19 63))
 (concatenate
  'string
  *MCStylistic-Oct2010-example-files-path*
  "/COSIATEC output"))
--> 1 2 T
```

Implementation of the COSIATEC algorithm. It can be verified (by checking the files created in the specified location) that the output (pattern- translators pairs) constitutes a cover of the input dataset.

COSIATEC-mod-2nd-n

Started, last checked	25/1/2010, 25/1/2010
Location	Further structural induction algorithms
Calls	argmax-of-threeCs, read-from-file, remove-pattern-occs-from-dataset-mod-2nd-n, SIA-reflected-for-COSIATEC-mod-2nd-n, SIATEC-mod-2nd-n, threeCs-pattern-trans-pairs-mod-2nd-n, write-to-file
Called by	
Comments/see also	A more efficient implementation is required. See also COSIATEC

Example:

```
(COSIATEC-mod-2nd-n
 '((0 1) (0 5) (1 4) (4 2) (4 6) (5 5) (8 0)
   (8 4) (9 3) (12 8) (13 9) (15 5) (16 9)
   (16 11) (17 4) (19 3))
 12
 (concatenate
  'string
  *MCStylistic-Oct2010-example-files-path*
  "/COSIATEC mod 2nd output"))
--> 1 2 T
```

Implementation of the COSIATEC algorithm, where translations in the second dimension are performed modulo the second argument. It can be verified (by checking the files created in the specified location) that the output above (pattern-translators pairs) constitutes a cover of the input dataset.

remove-pattern-occurrences-from-dataset

Started, last checked	25/1/2010, 25/1/2010
Location	Further structural induction algorithms
Calls	set-difference-multidimensional-sorted-asc, translations, unions-multidimensional-sorted-asc
Called by	COSIATEC
Comments/see also	remove-pattern-occs-from-dataset-mod-2nd-n

Example:

```
(remove-pattern-occurrences-from-dataset
 '(((0 60) (1 61)) (0 0) (1 1) (4 -1))
 '((0 60) (1 61) (2 62) (3 60) (4 59) (5 60) (6 57)))
--> ((3 60) (6 57))
```

All of the datapoints that are members of occurrences of a given pattern are calculated. These datapoints are then removed from the dataset (calling the function set-difference-multidimensional-sorted- asc), and the new dataset returned.

remove-pattern-occs-from-dataset-mod-2nd-n

Started, last checked	25/1/2010, 25/1/2010
Location	Further structural induction algorithms
Calls	set-difference-multidimensional-sorted-asc, translations-mod-2nd-n, unions-multidimensional-sorted-asc
Called by	COSIATEC-mod-2nd-n
Comments/see also	remove-pattern-occurrences-from-dataset

Example:

```
(remove-pattern-occs-from-dataset-mod-2nd-n
 '(((0 0) (1 1)) (0 0) (1 1) (4 11))
 '((0 0) (1 1) (2 2) (3 0) (4 11) (5 0) (6 9))
 12)
--> ((3 0) (6 9))
```

All of the datapoints that are members of occurrences of a given pattern are calculated, where translations in the second dimension are carried out modulo the third argument. These datapoints are then removed from the dataset (calling the function set- difference-multidimensional-sorted-asc), and the new dataset returned.

SIA-reflected-for-COSIATEC

Started, last checked	25/1/2010, 25/1/2010
Location	Further structural induction algorithms
Calls	subtract-two-lists, write-to-file
Called by	COSIATEC
Comments/see also	Deprecated, contingent on a rewrite of COSIATEC.

Example:

```
(SIA-reflected-for-COSIATEC
 '((0 61) (0 65) (1 64) (4 62) (4 66) (5 65) (8 60)
   (8 64) (9 63) (12 56) (13 69) (15 65) (16 57)
   (16 59) (17 64) (19 63))
 (concatenate
  'string
  *MCStylistic-Oct2010-example-files-path*
```

```
  "/SIA4COSIATEC output.txt"))
--> T
```

This function is a version of the SIA algorithm. It is called 'SIA-reflected-for-COSIATEC' because it is a slight variant on SIA-reflected. In particular it does not allow a partition size to be set.

SIA-reflected-for-COSIATEC-mod-2nd-n

Started, last checked	25/1/2010, 25/1/2010
Location	Further structural induction algorithms
Calls	subtract-two-lists-mod-2nd-n, write-to-file
Called by	COSIATEC-mod-2nd-n
Comments/see also	Deprecated, contingent on a rewrite of COSIATEC-mod-2nd-n.

Example:

```
(SIA-reflected-for-COSIATEC-mod-2nd-n
 '((0 1) (0 5) (1 4) (4 2) (4 6) (5 5) (8 0)
   (8 4) (9 3) (12 8) (13 9) (15 5) (16 9)
   (16 11) (17 4) (19 3))
 12
 (concatenate
  'string
  *MCStylistic-Oct2010-example-files-path*
  "/SIA4COSIATEC mod 2nd output.txt"))
--> T
```

This function is a version of the SIA algorithm. It is called 'SIA-reflected-for-COSIATEC' because it is a slight variant on SIA-reflected. In particular it does not allow a partition size to be set. Also in the mod-2nd-n version, translations in the second dimension are made modulo the second argument.

SIATEC

Started, last checked	25/1/2010, 25/1/2010
Location	Further structural induction algorithms
Calls	translators-of-pattern-in-dataset, write-to-file
Called by	COSIATEC
Comments/see also	SIATEC-mod-2nd-n

Example:

```
(progn
  (setq
   SIA-output
   '((((1/2 60) (1 62) (3/2 64) (2 67) (5/2 69) (3 71)
       (7/2 74) (4 71) (9/2 69) (5 67) (11/2 64)
       (6 60))
      ((49/4 71) (25/2 72) (51/4 73) (13 69) (13 74)
       (27/2 68) (27/2 73) (14 69) (14 74) (29/2 68)
       (29/2 73) (15 69) (15 74))))
  (setq
   dataset
   (read-from-file
    (concatenate
     'string
     *MCStylistic-Oct2010-example-files-path*
     "/scarlatti-L10-bars1-19.txt")))
  (setq
   projected-dataset
   (orthogonal-projection-unique-equalp
    dataset '(1 0 1 0 0)))
  (setq
   path&name
   (concatenate
    'string
    *MCStylistic-Oct2010-example-files-path*
    "/SIATEC output.txt"))
  (SIATEC SIA-output projected-dataset path&name))
--> T
```

This function applies the SIATEC algorithm to the output of the function SIA-reflected. The example causes a file to be created in the specified location.

SIATEC-mod-2nd-n

Started, last checked	25/1/2010, 25/1/2010
Location	Further structural induction algorithms
Calls	translators-of-pattern-in-dataset-mod-2nd-n, write-to-file
Called by	COSIATEC-mod-2nd-n
Comments/see also	SIATEC

Example:

```
(progn
  (setq
   SIA-mod-2nd-n-output
   '((((1/2 4) (1 6) (3/2 1) (2 4) (5/2 6) (3 1)
      (7/2 4) (4 1) (9/2 6) (5 4) (11/2 1) (6 4))
     ((49/4 1) (25/2 2) (51/4 3) (13 6) (13 4)
      (27/2 5) (27/2 3) (14 6) (14 4) (29/2 5)
      (29/2 3) (15 6) (15 4))))
  (setq
   dataset
   (read-from-file
    (concatenate
     'string
     *MCStylistic-Oct2010-example-files-path*
     "/scarlatti-L10-bars1-19.txt")))
  (setq
   dataset-1-0-1-0-0
   (orthogonal-projection-unique-equalp
    dataset '(1 0 1 0 0)))
  (setq
   dataset-1-0-1*-1-0
   (sort-dataset-asc
    (mod-column
     dataset-1-0-1-0-0 7 1)))
  (setq
   path&name
   (concatenate
    'string
    *MCStylistic-Oct2010-example-files-path*
    "/SIATEC mod 2nd output.txt"))
```

```
  (SIATEC-mod-2nd-n
   SIA-mod-2nd-n-output dataset-1-0-1*-1-0 7
   path&name))
--> T
```

This function applies the SIATEC algorithm to the output of the function SIA-reflected, where translations in the second dimension are carried out modulo the third argument. The example causes a file to be created in the specified location.

4.4.4 Evaluation for SIA+

The aim of these functions is to provide support for the implementation of COSIATEC (Meredith et al., 2003; Forth and Wiggins, 2009).

argmax-of-threeCs

Started, last checked	10/8/2009, 10/8/2009
Location	Evaluation for SIA+
Calls	multiply-two-lists, max-argmax, normalise-0-1
Called by	
Comments/see also	

Example:

```
(argmax-of-threeCs
 '((1 5 5/4) (1/10 4 4/3) (1 6 1) (4/5 12 2)))
--> 3
```

The argument to this function is a list of sublists, each containing three elements. Lists are constructed from these elements along dimensions one to three, and normalised linearly to $[0, 1]$. Then the lists are multiplied element-wise, and the resulting list is searched for the argument of the maximum element.

index-target-translation-in-list-assoc

Started, last checked	10/8/2009, 10/8/2009
Location	Evaluation for SIA+
Calls	test-translation
Called by	number-of-targets-translation-in-list-assoc
Comments/see also	

Example:

```
(setq
 a-list
 '((((151/3 84 1/3) (152/3 83 1/3) (51 81 1/3))
    (-8 0 0) (-4 0 0) (0 0 0) (11/3 7 0) (19/3 0 0)
    (23/3 0 0) (26/3 -5 0))
   (((143/3 84 1/3) (48 83 1/3) (146/3 86 1/3))
    (0 0 0) (23/3 0 0))
   (((52 43 2) (54 31 2) (56 36 2))
    (0 0 0) (8 7 0))
   (((5 76 1/2) (11/2 79 1/2))
    (0 0 0) (225/4 0 -1/4))))
(index-target-translation-in-list-assoc
 '((62 44 2) (64 32 2) (66 37 2)) a-list)
--> 2
```

The sublists of the list each contain a pattern and its translators. We want to know if any of the patterns are translations of the first argument, the target. The index of the first extant translation is returned, and nil otherwise. This function is used for checking the output of COSIATEC, as it uses assoc.

index-target-translation-in-list-rassoc

Started, last checked	10/8/2009, 10/8/2009
Location	Evaluation for SIA+
Calls	test-translation
Called by	number-of-targets-translation-in-list-rassoc
Comments/see also	

Example:

```
(setq
```

```
 a-list
 '(((1 0 0)
    (66 55 1) (66 65 1) (67 55 1) (68 55 1) (68 64 1)
    (69 55 1))
   ((11/3 -42 5/3)
    (163/3 90 1/3))
   ((10/3 -11 0)
    (163/3 90 1/3))
   ((3 -9 0)
    (163/3 90 1/3) (164/3 88 1/3) (56 88 1/3))))
(index-target-translation-in-list-rassoc
 '((166/3 60 1/3) (167/3 58 1/3) (57 58 1/3)) a-list)
--> 3
```

The sublists of the list each contain a pattern and its translators. We want to know if any of the patterns are translations of the first argument, the target. The index of the first extant translation is returned, and nil otherwise. This function is used for checking the output of SIA, as it uses rassoc.

index-target-translation-mod-in-list-assoc

Started, last checked	10/8/2009, 10/8/2009
Location	Evaluation for SIA+
Calls	test-translation-mod-2nd-n
Called by	number-of-targets-trans-mod-in-list-assoc
Comments/see also	

Example:

```
(setq
 a-list
 '((((151/3 0 1/3) (152/3 11 1/3) (51 10 1/3))
    (-8 0 0) (-4 0 0) (0 0 0) (11/3 7 0) (19/3 0 0)
    (23/3 0 0) (26/3 -5 0))
   (((143/3 0 1/3) (48 11 1/3) (146/3 10 1/3))
    (0 0 0) (23/3 0 0))
   (((52 7 2) (54 7 2) (56 0 2))
    (0 0 0) (8 7 0))
   (((5 4 1/2) (11/2 7 1/2))
    (0 0 0) (225/4 0 -1/4))))
(index-target-translation-mod-in-list-assoc
```

```
 '((62 0 2) (64 0 2) (66 5 2)) a-list 12)
--> 2
```

This function is very similar to the function index-target-translation-in-list-assoc, except that in the second dimension translations are carried out modulo the third argument. The sublists of the list each contain a pattern and its translators. We want to know if any of the patterns are translations of the first argument, the target. The index of the first extant translation is returned, and nil otherwise. This function is used for checking the output of the function COSIATEC-mod-2nd-n, as it uses assoc (when the dataset has been projected modulo n).

index-target-translation-mod-in-list-rassoc

Started, last checked	10/8/2009, 10/8/2009
Location	Evaluation for SIA+
Calls	test-translation-mod-2nd-n
Called by	number-of-targets-trans-mod-in-list-rassoc
Comments/see also	

Example:

```
(setq
 a-list
 '(((1 0 0)
    (66 55 1) (66 65 1) (67 55 1) (68 55 1) (68 64 1)
    (69 55 1))
   ((11/3 -42 5/3)
    (163/3 90 1/3))
   ((10/3 -11 0)
    (163/3 90 1/3))
   ((3 -9 0)
    (163/3 90 1/3) (164/3 88 1/3) (56 88 1/3))))
(index-target-translation-mod-in-list-rassoc
 '((166/3 0 1/3) (167/3 10 1/3) (57 10 1/3))
 a-list 12)
--> 3
```

This function is very similar to the function index-target-translation-in-list-rassoc, except that in the second dimension translations are carried out modulo the third argument. The sublists of the list each contain a pattern and its translators. We want to know if any of the patterns are translations of the first argument, the target. The index of the first extant translation is returned, and nil otherwise. This function is used for checking the output of the function SIA-mod-2nd-n, as it uses rassoc (when the dataset has been projected modulo n).

number-of-targets-translation-in-list-assoc

Started, last checked	10/8/2009, 10/8/2009
Location	Evaluation for SIA+
Calls	index-target-translation-in-list-assoc
Called by	
Comments/see also	

Example:

```
(setq
 a-list
 '((((151/3 84 1/3) (152/3 83 1/3) (51 81 1/3))
    (-8 0 0) (-4 0 0) (0 0 0) (11/3 7 0) (19/3 0 0)
    (23/3 0 0) (26/3 -5 0))
   (((143/3 84 1/3) (48 83 1/3) (146/3 86 1/3))
    (0 0 0) (23/3 0 0))
   (((52 43 2) (54 31 2) (56 36 2))
    (0 0 0) (8 7 0))
   (((5 76 1/2) (11/2 79 1/2))
    (0 0 0) (225/4 0 -1/4))))
(number-of-targets-translation-in-list-assoc
 '(((62 44 2) (64 32 2) (66 37 2))
   ((5 76 1/2) (11/2 79 1/2))
   ((5 76 1/2) (6 79 1/2)))
 a-list)
--> 2
```

The function index-target-translation-in- list-assoc is applied recursively to each member of the first argument of this function. This argument is a list of targets. Each time a translation of a target is detected, the output (initially set to zero) is incremented by one.

number-of-targets-translation-in-list-rassoc

Started, last checked	10/8/2009, 10/8/2009
Location	Evaluation for SIA+
Calls	index-target-translation-in-list-rassoc
Called by	
Comments/see also	

Example:

```
(setq
 a-list
 '(((1 0 0)
    (66 55 1) (66 65 1) (67 55 1) (68 55 1) (68 64 1)
    (69 55 1))
   ((11/3 -42 5/3)
    (163/3 90 1/3))
   ((10/3 -11 0)
    (163/3 90 1/3))
   ((3 -9 0)
    (163/3 90 1/3) (164/3 88 1/3) (56 88 1/3))))
(number-of-targets-translation-in-list-rassoc
 '(((166/3 60 1/3) (167/3 58 1/3) (57 58 1/3))
   ((66 55 1) (66 65 1) (67 55 1) (68 55 1) (68 64 1)
    (69 55 1))
   ((163/3 90 1/3)))
 a-list)
--> 3
```

The function index-target-translation-in- list-rassoc is applied recursively to each member of the first argument of this function. This argument is a list of targets. Each time a translation of a target is detected, the output (initially set to zero) is incremented by one.

number-of-targets-trans-mod-in-list-assoc

Started, last checked	10/8/2009, 10/8/2009
Location	Evaluation for SIA+
Calls	index-target-translation-mod-in-list-assoc
Called by	
Comments/see also	

Example:

```
(setq
 a-list
 '((((151/3 84 1/3) (152/3 83 1/3) (51 81 1/3))
    (-8 0 0) (-4 0 0) (0 0 0) (11/3 7 0) (19/3 0 0)
    (23/3 0 0) (26/3 -5 0))
   (((143/3 84 1/3) (48 83 1/3) (146/3 86 1/3))
    (0 0 0) (23/3 0 0))
   (((52 43 2) (54 31 2) (56 36 2))
    (0 0 0) (8 7 0))
   (((5 76 1/2) (11/2 79 1/2))
    (0 0 0) (225/4 0 -1/4))))
(number-of-targets-trans-mod-in-list-assoc
 '(((62 8 2) (64 8 2) (66 1 2))
   ((5 4 1/2) (11/2 7 1/2))
   ((5 4 1/2) (6 7 1/2)))
 a-list 12)
--> 2
```

The function index-target-translation-mod- in-list-assoc is applied recursively to each member of the first argument of this function. This argument is a list of targets. Each time a translation (modulo the third argument) of a target is detected, the output (initially set to zero) is incremented by one.

number-of-targets-trans-mod-in-list-rassoc

Started, last checked	10/8/2009, 10/8/2009
Location	Evaluation for SIA+
Calls	index-target-translation-mod-in-list-rassoc
Called by	
Comments/see also	

Example:

```
(setq
 a-list
 '(((1 0 0)
    (66 55 1) (66 65 1) (67 55 1) (68 55 1) (68 64 1)
    (69 55 1))
   ((11/3 -42 5/3)
```

```
    (163/3 90 1/3))
   ((10/3 -11 0)
    (163/3 90 1/3))
   ((3 -9 0)
    (163/3 90 1/3) (164/3 88 1/3) (56 88 1/3))))
(number-of-targets-trans-mod-in-list-rassoc
 '((((166/3 1 1/3) (167/3 11 1/3) (57 11 1/3))
   ((66 8 1) (66 6 1) (67 8 1) (68 8 1) (68 11 1)
    (69 8 1))
   ((163/3 6 1/3)))
 a-list 12)
--> 2
```

The function index-target-translation-mod- in-list-rassoc is applied recursively to each member of the first argument of this function. This argument is a list of targets. Each time a translation (modulo the third argument) of a target is detected, the output (initially set to zero) is incremented by one.

threeCs-pattern-trans-pair-mod-2nd-n

Started, last checked	10/8/2009, 10/8/2009
Location	Evaluation for SIA+
Calls	coverage-mod-2nd-n, index-item-1st-occurs, my-last
Called by	threeCs-pattern-trans-pairs-mod-2nd-n
Comments/see also	

Example:

```
(threeCs-pattern-trans-pair-mod-2nd-n
 '((0 0) (1 1)) '((0 0) (1 1) (3 0))
 '((0 0) (1 1) (2 2) (3 0) (5/2 7) (4 1)) 12)
--> (1 5 5/4)
```

A pattern and its translators in a projected dataset are supplied as arguments to this function. The output is the compactness, coverage and compression ratio, with translations in the second dimension being carried out modulo the fourth argument.

threeCs-pattern-trans-pairs-mod-2nd-n

Started, last checked	10/8/2009, 10/8/2009
Location	Evaluation for SIA+
Calls	threeCs-pattern-trans-pair-mod-2nd-n
Called by	
Comments/see also	

Example:

```
(threeCs-pattern-trans-pairs-mod-2nd-n
 '((((0 11) (1 0) (3 11)) (0 0) (1 1))
   (((0 11)) (0 0) (1 1) (2 2) (3 0) (5/2 8) (4 1)))
 '((0 11) (1 0) (2 1) (3 11) (5/2 7) (4 0))
 12)
--> ((3/4 5 5/4) (1 6 1))
```

Pairs (consisting of patterns and their translators and sometimes referred to as SIATEC- output) in a projected dataset are supplied as arguments to this function. The output is a list of lists, each of which contains the compactness, coverage and compression ratio of the corresponding pattern. Translations in the second dimension are carried out modulo the third argument.

threeCs-pattern-translators-pair

Started, last checked	10/8/2009, 10/8/2009
Location	Evaluation for SIA+
Calls	coverage, index-item-1st-occurs, my-last
Called by	threeCs-pattern-translators-pairs
Comments/see also	

Example:

```
(threeCs-pattern-translators-pair
 '((0 60) (1 61)) '((0 0) (1 1) (3 0))
 '((0 60) (1 61) (2 62) (3 60) (5/2 67) (4 61)))
--> (1 5 5/4)
```

A pattern and its translators in a projected dataset are supplied as arguments to this function. The output is the compactness, coverage and compression ratio.

threeCs-pattern-translators-pairs

Started, last checked	10/8/2009, 10/8/2009
Location	Evaluation for SIA+
Calls	threeCs-pattern-translators-pair
Called by	
Comments/see also	

Example:

```
(threeCs-pattern-translators-pairs
 '((((0 60) (1 61)) (0 0) (1 1) (3 0))
   (((0 60)) (0 0) (1 1) (2 2) (3 0) (5/2 7) (4 1)))
 '((0 60) (1 61) (2 62) (3 60) (5/2 67) (4 61)))
--> ((1 5 5/4) (1 6 1))
```

Pairs (consisting of patterns and their translators and sometimes referred to as SIATEC- output) in a projected dataset are supplied as arguments to this function. The output is a list of lists, each of which contains the compactness, coverage and compression ratio of the corresponding pattern.

4.4.5 Compactness trawl

These functions are designed to trawl through already- discovered patterns (usually MTPs) from beginning to end. They return subpatterns that have compactness (Meredith et al., 2003) and cardinality greater than thresholds that can be specified (Collins et al., 2010).

compact-subpatterns

Started, last checked	12/1/2010, 30/1/2010
Location	Compactness trawl
Calls	compactness, my-last
Called by	
Comments/see also	A more efficient implementation could be achieved by retaining the index of datapoints. See also compact-subpatterns-more-output

Example:

```
(compact-subpatterns
 '((1/2 72 67 1/2) (1 76 69 1/2) (3/2 79 71 1/2)
   (2 84 74 2) (7/2 60 60 1/2) (5 76 69 1/2)
   (11/2 79 71 1/2) (6 84 74 2) (13/2 67 64 1/2)
   (15/2 60 60 1/2))
 '((0 48 53 2) (1/2 72 67 1/2) (1 76 69 1/2)
   (3/2 79 71 1/2) (2 84 74 2) (5/2 67 64 1/2)
   (3 64 62 1/2) (7/2 60 60 1/2) (4 36 46 2)
   (9/2 72 67 1/2) (5 76 69 1/2) (11/2 79 71 1/2)
   (6 84 74 2) (13/2 67 64 1/2) (7 64 62 1/2)
   (15/2 60 60 1/2) (8 36 46 2) (17/2 72 67 1/2))
 2/3 3)
--> (((1/2 72 67 1/2) (1 76 69 1/2) (3/2 79 71 1/2)
      (2 84 74 2) (7/2 60 60 1/2))
     ((5 76 69 1/2) (11/2 79 71 1/2) (6 84 74 2)
      (13/2 67 64 1/2) (15/2 60 60 1/2)))
```

This function takes a pattern and looks within that pattern for subpatterns that have compactness and cardinality greater than certain thresholds, which are optional arguments. In this version, just the subpatterns are returned.

compact-subpatterns-more-output

Started, last checked	12/1/2010, 30/1/2010
Location	Compactness trawl
Calls	compactness, my-last
Called by	compactness-trawler
Comments/see also	A more efficient implementation could be achieved by retaining the index of datapoints. See also compact-subpatterns

Example:

```
(compact-subpatterns-more-output
 '((1/2 72 67 1/2) (1 76 69 1/2) (3/2 79 71 1/2)
   (2 84 74 2) (7/2 60 60 1/2) (5 76 69 1/2)
   (11/2 79 71 1/2) (6 84 74 2) (13/2 67 64 1/2)
   (15/2 60 60 1/2))
 '((0 48 53 2) (1/2 72 67 1/2) (1 76 69 1/2)
   (3/2 79 71 1/2) (2 84 74 2) (5/2 67 64 1/2)
   (3 64 62 1/2) (7/2 60 60 1/2) (4 36 46 2)
```

```
   (9/2 72 67 1/2) (5 76 69 1/2) (11/2 79 71 1/2)
   (6 84 74 2) (13/2 67 64 1/2) (7 64 62 1/2)
   (15/2 60 60 1/2) (8 36 46 2) (17/2 72 67 1/2))
 2/3 3)
--> ((((1/2 72 67 1/2) (1 76 69 1/2) (3/2 79 71 1/2)
       (2 84 74 2) (7/2 60 60 1/2))
      ((5 76 69 1/2) (11/2 79 71 1/2) (6 84 74 2)
       (13/2 67 64 1/2) (15/2 60 60 1/2)))
     (5/7 5/6))
```

This function takes a pattern and looks within that pattern for subpatterns that have compactness and cardinality greater than certain thresholds, which are optional arguments. In this version, the subpatterns and corresponding compactness values are returned.

compactness-trawler

Started, last checked	12/1/2010, 30/1/2010
Location	Compactness trawl
Calls	compact-subpatterns-more-output, test-translation, write-to-file
Called by	
Comments/see also	

Example: see Discovering and rating musical patterns (Sec. 3.3), especially lines 101-107).

The compactness trawler (Collins et al., 2010) applies the function compact-subpatterns-more-output recursively to the output of SIA (Structure Induction Algorithm, Meredith et al., 2002), or any output with an analogous list format.

4.4.6 Evaluation for SIACT

The purpose of these functions is to rate the trawled patterns, according to the formula for perceived pattern importance (Collins et al., 2011).

collect-indices&ratings

Started, last checked	19/1/2010, 30/1/2010
Location	Evaluation for SIACT
Calls	
Called by	evaluate-variables-of-patterns2hash
Comments/see also	

Example:

```
(setq
 patterns-hash
 (list
  (make-hash-table :test #'equal)
  (make-hash-table :test #'equal)
  (make-hash-table :test #'equal)))
(setf (gethash '"index" (first patterns-hash)) 0)
(setf (gethash '"rating" (first patterns-hash)) 3.3)
(setf (gethash '"index" (second patterns-hash)) 1)
(setf (gethash '"rating" (second patterns-hash)) 8.0)
(setf (gethash '"index" (third patterns-hash)) 2)
(setf (gethash '"rating" (third patterns-hash)) 2.1)
(collect-indices&ratings patterns-hash)
--> ((0 3.3) (1 8.0) (2 2.1))
```

This function collects the index and rating from each sublist of a list, where the sublist is a hash table consisting of information about a pattern.

evaluate-variables-of-pattern2hash

Started, last checked	19/1/2010, 30/1/2010
Location	Evaluation for SIACT
Calls	add-to-list, choose, coverage, first-n-naturals, index-item-1st-occurs, likelihood-of-translations-reordered, multiply-list-by-constant, my-last, nth-list, translators-of-pattern-in-dataset
Called by	evaluate-variables-of-patterns2hash
Comments/see also	

Example:

```
(setq
 pattern&source
 '((((1/2 72 67 1/2) (1 76 69 1/2) (3/2 79 71 1/2)
    (2 84 74 2) (5/2 67 64 1/2) (3 64 62 1/2)
    (7/2 60 60 1/2))
   16/23 (140 5 0) 1 (104 -5 0) 4/5 (96 -5 0)))
(setq
 dataset
 '((0 48 53 2) (1/2 72 67 1/2) (1 76 69 1/2)
   (3/2 79 71 1/2) (2 84 74 2) (5/2 67 64 1/2)
   (3 64 62 1/2) (7/2 60 60 1/2) (4 36 46 2)
   (9/2 72 67 1/2) (5 76 69 1/2) (11/2 79 71 1/2)
   (6 84 74 2) (13/2 67 64 1/2) (7 64 62 1/2)
   (15/2 60 60 1/2) (8 36 46 2) (17/2 72 67 1/2)
   (9 76 69 1/2) (19/2 79 71 1/2)))
(setq
 dataset-palette
 (orthogonal-projection-not-unique-equalp
  dataset
  (append
   (list 0)
   (constant-vector
    1
    (- (length
(first (first pattern&source))) 1)))))
(setq
 empirical-mass
 (empirical-mass dataset-palette))
(setq
 pattern-hash
 (evaluate-variables-of-pattern2hash
  pattern&source dataset 20 dataset-palette
  empirical-mass
  '(4.277867 3.422478734 -0.038536808 0.651073171)
  '(73.5383283152 0.02114878519) 1))
--> #<HASH-TABLE
    :TEST EQUAL size 12/60 #x3000418C079D>
(disp-ht-el pattern-hash)
--> (("name" . "pattern 1") ("compactness" . 16/23)
     ("expected occurrences" . 62.352943)
     ("rating" . 5.3952165)
```

```
("pattern"
 (1/2 72 67 1/2) (1 76 69 1/2) (3/2 79 71 1/2)
 (2 84 74 2) (5/2 67 64 1/2) (3 64 62 1/2)
 (7/2 60 60 1/2))
("translators" (0 0 0 0) (4 0 0 0)) ("index" . 1)
("cardinality" . 7)
("MTP vectors" (96 -5 0) (104 -5 0) (140 5 0))
("compression ratio" . 7/4)
("region"
 (1/2 72 67 1/2) (1 76 69 1/2) (3/2 79 71 1/2)
 (2 84 74 2) (5/2 67 64 1/2) (3 64 62 1/2)
 (7/2 60 60 1/2)) ("occurrences" . 2))
```

This function evaluates variables of the supplied pattern, such as cardinality and expected occurrences.

evaluate-variables-of-patterns2hash

Started, last checked	19/1/2010, 30/1/2010
Location	Evaluation for SIACT
Calls	collect-indices&ratings, constant-vector, empirical-mass, evaluate-variables-of-pattern2hash, nth-list, nth-list-of-lists, orthogonal-projection-not-unique-equalp, sort-by
Called by	
Comments/see also	

Example:

```
(setq
 pattern&sources
 '((((1/2 72 67 1/2) (1 76 69 1/2) (3/2 79 71 1/2)
     (2 84 74 2) (5/2 67 64 1/2) (3 64 62 1/2)
     (7/2 60 60 1/2))
    16/23 (140 5 0) 1 (104 -5 0) 4/5 (96 -5 0))
   (((1/2 72 67 1/2) (3/2 79 71 1/2))
    1 (130 0 0) 2/3 (100 0 1/2))))
(setq
 dataset
```

```
 '((0 48 53 2) (1/2 72 67 1/2) (1 76 69 1/2)
   (3/2 79 71 1/2) (2 84 74 2) (5/2 67 64 1/2)
   (3 64 62 1/2) (7/2 60 60 1/2) (4 36 46 2)
   (9/2 72 67 1/2) (5 76 69 1/2) (11/2 79 71 1/2)
   (6 84 74 2) (13/2 67 64 1/2) (7 64 62 1/2)
   (15/2 60 60 1/2) (8 36 46 2) (17/2 72 67 1/2)
   (9 76 69 1/2) (19/2 79 71 1/2)))
(setq
 patterns-hash
 (evaluate-variables-of-patterns2hash
  pattern&sources dataset
  '(4.277867 3.422478734 -0.038536808 0.651073171)
  '(73.5383283152 0.02114878519)))
--> (#<HASH-TABLE
     :TEST EQUAL size 12/60 #x300041916ACD>
     #<HASH-TABLE
     :TEST EQUAL size 12/60 #x30004188107D>)
(disp-ht-el (first patterns-hash))
--> (("name" . "pattern 1") ("compactness" . 1)
     ("expected occurrences" . 72.79239)
     ("rating" . 5.8717685)
     ("pattern" (1/2 72 67 1/2) (3/2 79 71 1/2))
     ("translators" (0 0 0 0) (4 0 0 0) (8 0 0 0))
     ("index" . 1) ("cardinality" . 2)
     ("MTP vectors" (100 0 1/2) (130 0 0))
     ("compression ratio" . 3/2)
     ("region"
      (1/2 72 67 1/2) (1 76 69 1/2) (3/2 79 71 1/2))
     ("occurrences" . 3))
```

This function applies the function evaluate-variables-of-pattern2hash recursively.

4.4.7 Superdiagonals

These functions implement a structural induction algorithm using merge sort, and with calculation of difference vectors restricted to the first r superdiagonals of the dataset's similarity matrix.

difference-list-sorted-asc

Started, last checked	6/9/2010, 6/9/2010
Location	Superdiagonals
Calls	subtract-two-lists, vector<vector-t-or-nil
Called by	merge-sort-diff-sets-of-datapoints
Comments/see also	

Example:

```
(difference-list-sorted-asc
 '((71 1/2) (143/2 1/2) (72 1/2) (145/2 1/2)))
--> ((1/2 0) (1/2 0) (1/2 0) (1 0) (1 0) (3/2 0))
```

For a dataset $D = \{\mathbf{d}_1, \ldots, \mathbf{d}_n\}$, this function calculates $(\mathbf{d}_j - \mathbf{d}_i)_{i<j}$ and sorts the output by lexicographic order.

merge-sort-by

Started, last checked	6/9/2010, 6/9/2010
Location	Superdiagonals
Calls	vector<vector-car-cdr
Called by	structure-induction-algorithm-r
Comments/see also	

Example:

```
(merge-sort-by
 '(((1 -1) 0 60) ((2 5) 1 59) ((1 -3) 3 64)
   ((3 4) 4 61) ((3 4) 0 60) ((3 2) 1 59)
   ((4 1) 3 64)))
--> (((1 -3) 3 64) ((1 -1) 0 60) ((2 5) 1 59)
     ((3 2) 1 59) ((3 4) 0 60) ((3 4) 4 61)
     ((4 1) 3 64))
```

As a default, this function applies the predicate vector<vector-car-cdr to merge-sort the list provided as an argument.

merge-sort-diff-sets-of-datapoints

Started, last checked	6/9/2010, 6/9/2010
Location	Superdiagonals
Calls	difference-list-sorted-asc, remove-duplicates-sorted-asc, vector<vector-t-or-nil
Called by	structure-induction-algorithm-r
Comments/see also	

Example:

```
(merge-sort-diff-sets-of-datapoints
 '(((1 0) (0 60) (0 64) (4 59)) ((4 7) (0 60) (0 64))
   ((0 16) (0 60) (0 62) (0 64))))
--> ((0 2) (0 4) (0 4) (0 4) (4 -5) (4 -1))
```

The argument to this function is a list of vector-datapoints pairs. The elements of the upper triangle of the similarity matrix of each set of datapoints are merge sorted. Duplicates are removed within similarity matrices but may still occur between similarity matrices.

remove-duplicates-sorted-asc

Started, last checked	6/9/2010, 6/9/2010
Location	Superdiagonals
Calls	
Called by	merge-sort-diff-sets-of-datapoints
Comments/see also	Should test against built-in Lisp functions.

Example:

```
(remove-duplicates-sorted-asc '(0 0 6 7 8 8 8))
--> (0 6 7 8)
```

Consequent elements of a list are checked for equality to remove duplicates from an already- sorted list.

structure-induction-algorithm-r

Started, last checked	6/9/2010, 6/9/2010
Location	Superdiagonals
Calls	collect-by-cars, frequency-count, maximal-translatable-pattern, merge-sort-by, merge-sort-diff-sets-of-datapoints, sort-dataset-asc, subtract&retain-at-fixed-distances
Called by	
Comments/see also	

Example:

```
(setq
 dataset
 '((0 60) (1 61) (2 62) (3 60) (5 60) (5 61) (6 59)
   (6 62) (7 60) (7 63) (8 61)))
(structure-induction-algorithm-r
 dataset 1
 (merge-pathnames
  (make-pathname
   :name "SIA_r example"
   :type "txt")
  *MCStylistic-Aug2013-example-files-results-path*))
--> Writes file to the specified location.
```

The first r superdiagonals of the similarity matrix for the dataset are treated in a SIA-like fashion to form patterns. For each pattern $P_i = \{\mathbf{p}_{i,1}, \ldots, \mathbf{p}_{i,l_i}\}$, we calculate the vector $\mathbf{v} = \mathbf{p}_{i,j} - \mathbf{p}_{i,1}, 2 \leq j \leq l_i$. If this vector is not in a growing list, then its MTP is computed and appended to the output.

subtract&retain-at-fixed-distance

Started, last checked	6/9/2010, 6/9/2010
Location	Superdiagonals
Calls	subtract-two-lists
Called by	subtract&retain-at-fixed-distances
Comments/see also	

Example:

```
(subtract&retain-at-fixed-distance
 '((0 60) (0 67) (0 72) (1 62) (2 66) (3 59) (3 62)
   (3 67) (4 69) (11/2 71) (6 60) (6 67) (6 72)) 1)
--> (((0 7) 0 60) ((0 5) 0 67) ((1 -10) 0 72)
     ((1 4) 1 62) ((1 -7) 2 66) ((0 3) 3 59)
     ((0 5) 3 62) ((1 2) 3 67) ((3/2 2) 4 69)
     ((1/2 -11) 11/2 71) ((0 7) 6 60) ((0 5) 6 67))
```

A list $(\mathbf{e}_1, \mathbf{e}_2, \ldots, \mathbf{e}_n)$ is the argument to this function, and an optional fixed distance k. The pairs $(\mathbf{e}_{k+i} - \mathbf{e}_i, \mathbf{e}_i)$ are output for $i = 1, 2, \ldots, n-k$.

subtract&retain-at-fixed-distances

Started, last checked	6/9/2010, 6/9/2010
Location	Superdiagonals
Calls	subtract&retain-at-fixed-distance
Called by	structure-induction-algorithm-r
Comments/see also	

Example:

```
(subtract&retain-at-fixed-distances
 '((0 60) (0 67) (0 72) (1 62)) 2)
--> (((0 7) 0 60) ((0 5) 0 67) ((1 -10) 0 72)
     ((0 12) 0 60) ((1 -5) 0 67))
```

The function subtract&retain-at-fixed- distance is applied for $i = 1, 2, \ldots, k$, where k is the second argument. The output of each application is appended.

vector<vector-car-cdr

Started, last checked	6/9/2010, 6/9/2010
Location	Superdiagonals
Calls	vector<vector, vector<vector-t-or-nil
Called by	merge-sort-by
Comments/see also	

Example:

```
(vector<vector-car-cdr
 '((2 2) . (1 4)) '((2 2) . (1 3)))
--> NIL
```

Applies the function vector<vector to the car of each list (the two arguments), and if equal, applies the function vector<vector-t-or-nil to the cdr of each list.

4.5 Pattern metrics

4.5.1 Robust metrics

The functions below are metrics for evaluating the extent to which different ontime-pitch pairs have been output by an algorithm.

cardinality-score

Started, last checked	5/2/2013, 5/2/2013
Location	Robust metrics
Calls	constant-vector, most-frequent-difference-vector, max-item
Called by	establishment-matrix, establishment-metric, matching-score, occurrence-matrix&rel-idx, occurrence-metric
Comments/see also	

Example:

```
(setq
 a-dataset
 '((25 53) (25 69) (53/2 53) (53/2 69) (27 48) (27 52)
   (27 60) (27 67) (28 50) (28 53) (28 65)))
(setq
 b-dataset
 '((25 53) (25 60) (25 69) (53/2 53) (53/2 69) (27 40)
   (27 48) (27 51) (27 60) (27 67) (28 50) (28 53)))
--> 9/12
(cardinality-score a-dataset b-dataset)
(setq
 b-dataset (translation b-dataset '(40 6)))
(cardinality-score a-dataset b-dataset)
--> 0
(cardinality-score a-dataset b-dataset nil t)
--> 9/12
```

```
(setq
 b-dataset (translation a-dataset '(1e-5 0)))
(cardinality-score a-dataset b-dataset t)
--> 12/12
```

Returns the cardinality score for two point sets. This is the number of occurrences of the most frequent difference vector between two point sets (possibly including an error tolerance), divided by the maximum cardinality of the two point sets. If the optional argument translationp is set to false, the occurrences of the zero vector in the differences is used as the numerator, i.e., not considering translations to be equivalent.

equalp-score

Started, last checked	5/2/2013, 5/2/2013
Location	Robust metrics
Calls	
Called by	score-matrix
Comments/see also	

Example:

```
(setq
 a-dataset
 '((12 45 51 1/2 0) (12 48 53 1/2 0) (25/2 45 51 3/2)
   (25/2 52 55 1/2 0) (13 36 46 1 1) (13 41 49 1)
   (13 53 56 1 0) (14 29 42 1 1) (14 36 46 1 1)))
(setq
 b-dataset
 '((12 45 51 1/2 0) (12 48 53 1/2 0) (25/2 45 51 3/2)
   (25/2 52.0001 55 1/2 0) (13 36 46 1 1) (13 41 49 1)
   (13 53 56 1 0) (14 29 42 1 1) (14 36 46 1 1)))
(setq *equality-tolerance* 1e-2)
(equalp-score a-dataset b-dataset t)
--> T
```

This function returns t if the two datasets provided as arguments are equal. An optional argument for equality up to tolerance specified by the variable *equality-tolerance* allows for different degrees of exactness in testing equality.

establishment-matrix

Started, last checked	5/2/2013, 5/2/2013
Location	Robust metrics
Calls	
Called by	establishment-metric, occurrence-matrix&rel-idx
Comments/see also	

Example:

```
(setq P11 '((1 53) (5/2 53) (3 40) (3 52) (4 50)))
(setq
 P12 '((21 60) (45/2 60) (23 47) (23 59) (25 57)))
(setq P13 '((41 53) (85/2 53) (43 52) (44 50)))
(setq P1 (list P11 P12 P13))

(setq P21 '((5/2 55) (3 57) (7/2 58) (4 60) (5 48)))
(setq
 P22 '((21/2 55) (43/4 57) (23/2 58) (12 60) (13 48)))
(setq
 P23 '((23/2 58) (12 60) (25/2 62) (13 64) (14 52)))
(setq
 P24 '((27/2 50) (14 52) (29/2 53) (15 55) (16 42)))
(setq P2 (list P21 P22 P23 P24))
(setq P-patt&occ (list P1 P2))

(setq Q11 '((1 53) (5/2 53) (3 40) (3 52) (4 50)))
(setq Q12 (translation Q11 '(5 2)))
(setq Q13 '((0 59) (1/2 60) (1 60) (2 62)))
(setq Q14 (translation Q11 '(40 0)))
(setq Q1 (list Q11 Q12 Q13 Q14))

(setq Q21 (butlast (translation Q11 '(8 5))))
(setq Q22 (rest (translation Q12 '(8 5))))
(setq Q23 (rest (translation Q13 '(8 5))))
(setq Q24 (butlast (translation Q14 '(8 5))))
(setq Q2 (list Q21 Q22 Q23 Q24))

(setq
 Q31
```

```
 '((67/2 60) (67/2 64) (34 62) (34 65) (69/2 64)
   (69/2 67) (35 57) (35 65) (35 69) (35 72)))
(setq
 Q32
 '((97/2 60) (97/2 64) (49 62) (49 65) (99/2 64)
   (99/2 67) (50 58) (50 66) (50 70) (50 72)))
(setq Q3 (list Q31 Q32))
(setq Q-patt&occ (list Q1 Q2 Q3))
(establishment-matrix P-patt&occ Q-patt&occ)
--> ((1 0 0) (0 0 0))
```

The establishment matrix indicates whether an algorithm is strong at *establishing* that a pattern P is repeated at least once during a piece. It does not indicate whether the algorithm is strong at retrieving all occurrences of P (exact and inexact). For such a matrix, please see the function occurrence-matrix.

The first argument to the function establishment- matrix is a list of patterns, and within each list, all occurrences of that pattern. These are assumed to be the ground truth. The second argument has the same structure as the first, but for the output of some algorithm. The number of each element in the output establishment matrix indicates the extent to which output pattern Qj (and all its occurrences) constitutes the discovery of ground truth pattern Pi (and all its occurrences). The are options for comparison function, error tolerance, and accepting discoveries up to translation.

For each pair of point sets in the first and second arguments, this function computes the cardinality score or matching score (or some other function for computing symbolic musical similarity). These scores are returned in a list of lists. Each lists contains a row of scores for a point set from P-occurrences, and so the nth element of each list corresponds to a column of scores for a point set from Q-occurrences.

establishment-metric

Started, last checked	5/2/2013, 5/2/2013
Location	Robust metrics
Calls	establishment-matrix, precision-matrix, recall-matrix
Called by	
Comments/see also	

Example:

```
P-patt&occ and Q-patt&occ defined as in the example
for the function establishment-matrix.
(establishment-metric
 P-patt&occ Q-patt&occ #'precision-matrix)
--> 11/15
(establishment-metric
 P-patt&occ Q-patt&occ #'recall-matrix)
--> 7/10
```

This function is a wrapper for calculating the precision or recall of an establishment matrix. The establishment matrix indicates whether an algorithm is strong at *establishing* that a pattern P is repeated at least once during a piece. The entry element (i, j) is a scalar between 0 and 1 indicating the performance of algorith-output pattern j for ground truth pattern i. This metric does not indicate whether the algorithm is strong at retrieving all occurrences of P (exact and inexact). For such a metric, please see the function occurrence-metric.

index-lookup

Started, last checked	5/2/2013, 5/2/2013
Location	Robust metrics
Calls	
Called by	indices-lookup
Comments/see also	

Example:

```
(index-lookup
 '(1/2 60) '((0 59) (1/2 60) (1/2 60) (2 62)))
--> (1 2).
```

Returns indices of the second list that are equal to the first list.

indices-lookup

Started, last checked	5/2/2013, 5/2/2013
Location	Robust metrics
Calls	index-lookup
Called by	indices-lookup
Comments/see also	

Example:

```
(indices-lookup
 '((1/2 60) (2 63))
 '((0 59 59) (1/2 60 60) (1/2 61 60) (2 62 63))
 '(1 0 1))
--> ((1 2) (3))
```

Returns indices of the second list that are equal to members of the first list. If there is more than one, both matches are returned. It is possible to focus on certain dimensions by specifying a projection vector.

matching-score

Started, last checked	5/2/2013, 5/2/2013
Location	Robust metrics
Calls	cardinality-score, matching-score-histogram, max-item, symbolic-fingerprint
Called by	score-matrix
Comments/see also	

Example:

```
(setq
 a-dataset
 '((-1 81) (-3/4 76) (-1/2 85) (-1/4 81) (0 88) (1 57)
   (1 61) (1 64) (2 73) (9/4 69) (5/2 76) (11/4 73)
   (3 81) (4 45) (4 49) (4 52) (4 57) (5 61) (21/4 57)
   (11/2 64) (23/4 61) (6 57) (6 69) (7 54) (7 59)
   (7 63) (7 69) (8 51) (8 59) (8 66) (8 69) (9 52)
   (9 59) (9 66) (9 69) (10 40) (10 64) (10 68)))
(setq
 b-dataset
 '((21 56) (21 62) (21 71) (22 57) (22 61) (22 69)
   (23 81) (93/4 76) (47/2 85) (95/4 81) (24 88)
   (25 57) (25 58) (25 64) (26 73) (105/4 69)
   (53/2 76) (107/4 73) (27 81) (28 45) (28 49)
   (28 52) (28 57) (29 61) (117/4 57) (59/2 64)
   (119/4 61) (30 57) (30 69) (31 54) (31 59) (31 63)
   (31 69) (32 51) (32 59) (32 66) (32 69) (33 52)
   (33 59) (33 66) (33 69) (34 40) (34 64) (34 68)
```

```
   (35 80) (141/4 76) (71/2 83) (143/4 80) (36 86)
   (37 52) (37 56) (37 59) (37 62) (38 68) (153/4 64)
   (77/2 71) (155/4 68) (39 74) (40 40) (40 44)
   (40 47) (40 52) (41 62) (165/4 59) (83/2 64)
   (167/4 62) (42 68) (43 52) (43 59) (43 62) (43 68)
   (44 52) (44 59) (44 62) (44 68) (45 45) (45 56)
   (45 62) (45 71) (46 57)))
(matching-score a-dataset b-dataset)
--> 0.6097795
```

This function calculates the matching score histogram defined by Arzt et al. (2012). En route it calculates two collections of symbolic fingerprints for the two input datasets. The maximum value in the histogram corresponds to the time point at which the two datasets are most similar. Divided by the maximum possible number of matches (maximum number of symbolic fingerprints), and taking the square root, this number can be used to measure the symbolic musical similarity of the two datasets.

most-frequent-difference-vector

Started, last checked	5/2/2013, 5/2/2013
Location	Robust metrics
Calls	constant-vector, difference-lists, frequency-count, nth-list-of-lists, sort-dataset-asc
Called by	cardinality-score
Comments/see also	

Example:

```
(setq
 a-list
 '((25 53) (25 69) (53/2 53) (53/2 69) (27 48) (27 52)
   (27 60) (27 67) (28 50) (28 53) (28 65.01)))
(setq
 b-list
 '((25 53) (25 69) (53/2 53) (53/2 69) (27 48) (27 52)
   (27 60) (27 67) (28 50) (28 53) (28 65)))
(most-frequent-difference-vector a-list b-list nil)
--> ((0 0) 10)
(setq
```

```
 a-list
 '((25 53) (25 69) (53/2 53) (53/2 69) (27 48) (27 52)
   (27 60) (27 67) (28 50) (28 53) (28 65.00001)))
(most-frequent-difference-vector a-list b-list t)
--> ((0 0) 11)
(setq a-list (translation a-list '(4 5)))
(most-frequent-difference-vector a-list b-list t t)
((-4 -5) 11)
```

Returns the most frequent difference vector (and its total number of occurrences) between two lists of lists. There are options for allowing equality up to a given tolerance (specified by the variable *equality-tolerance*), and for whether the sets are allowed to be translations of one another. If not, the count of the zero vector is returned.

occurrence-matrix&rel-idx

Started, last checked	5/2/2013, 5/2/2013
Location	Robust metrics
Calls	add-to-list, cardinality-score, establishment-matrix, first-n-naturals, indices-of-matrix-passing-tests, precision-matrix, score-matrix
Called by	occurrence-metric
Comments/see also	

Example:

```
(setq P11 '((1 53) (5/2 53) (3 40) (3 52) (4 50)))
(setq
 P12 '((21 60) (45/2 60) (23 47) (23 59) (25 57)))
(setq P13 '((41 53) (85/2 53) (43 52) (44 50)))
(setq P1 (list P11 P12 P13))

(setq P21 '((5/2 55) (3 57) (7/2 58) (4 60) (5 48)))
(setq
 P22 '((21/2 55) (43/4 57) (23/2 58) (12 60) (13 48)))
(setq
 P23 '((23/2 58) (12 60) (25/2 62) (13 64) (14 52)))
(setq
 P24 '((27/2 50) (14 52) (29/2 53) (15 55) (16 42)))
```

```
(setq P2 (list P21 P22 P23 P24))
(setq P-patt&occ (list P1 P2))

(setq Q11 '((1 53) (5/2 53) (3 40) (3 52) (4 50)))
(setq Q12 (translation Q11 '(5 2)))
(setq Q13 '((0 59) (1/2 60) (1 60) (2 62)))
(setq Q14 (translation Q11 '(40 0)))
(setq Q1 (list Q11 Q12 Q13 Q14))

(setq Q21 (butlast (translation Q11 '(8 5))))
(setq Q22 (rest (translation Q12 '(8 5))))
(setq Q23 (rest (translation Q13 '(8 5))))
(setq Q24 (butlast (translation Q14 '(8 5))))
(setq Q2 (list Q21 Q22 Q23 Q24))

(setq
 Q31
 '((67/2 60) (67/2 64) (34 62) (34 65) (69/2 64)
   (69/2 67) (35 57) (35 65) (35 69) (35 72)))
(setq
 Q32
 '((97/2 60) (97/2 64) (49 62) (49 65) (99/2 64)
   (99/2 67) (50 58) (50 66) (50 70) (50 72)))
(setq Q3 (list Q31 Q32))
(setq Q-patt&occ (list Q1 Q2 Q3))

(occurrence-matrix&rel-idx
 P-patt&occ Q-patt&occ #'precision-matrix)
--> (((9/20 0 0) (0 0 0)) ((0 0) (0 1)))
(occurrence-matrix&rel-idx
 P-patt&occ Q-patt&occ #'recall-matrix)
--> (((3/5 0 0) (0 0 0)) ((0 0) (0 1)))
```

The occurrence matrix indicates whether an algorithm is strong at retrieving *all occurrences* of a pattern P, given that the existence of at least one repetition of P has been established. So even if an algorithm fails to discover many noticeable/important patterns in a piece, it can still score well on the precision or recall of its occurrence matrix.

The first argument to the function occurrence-matrix is a list of patterns, and within each list, all occurrences of that pattern. These are assumed to be the ground truth. The second argument has the same structure as the first, but for the output of some algorithm. The first calculation is the establishment matrix, indicating the extent to which output pattern Qj (and all its occurrences) constitutes the discovery of ground truth pattern Pi (and all its occurrences). The are options for comparison function, error tolerance, and accepting discoveries up to translation.

The second calculation is restricted to elements of the establishment matrix greater than a score threshold (specifiable by the variable score-thresh), thus putting failed discoveries to one side and focusing on the retrieved occurrences of successful discoveries. A score matrix is computed for each Pi and Qj above the score threshold, which contains information about the retrieval of occurrences. Either the precision or recall of each matrix is returned and forms a new sparser matrix called the occurrence matrix. Also output are the indices of the establishment matrix greater than the score threshold.

occurrence-metric

Started, last checked	5/2/2013, 5/2/2013
Location	Robust metrics
Calls	cardinality-score, nth-list-of-lists, precision-matrix, occurrence-matrix&rel-idx
Called by	occurrence-matrix&rel-idx
Comments/see also	

Example:

```
P-patt&occ and Q-patt&occ defined as in the example
for the function occurrence-matrix&rel-idx.
(occurrence-metric
 P-patt&occ Q-patt&occ #'precision-matrix)
--> 9/40
(occurrence-metric
 P-patt&occ Q-patt&occ #'recall-matrix)
--> 3/5
```

The occurrence metrics indicate whether an algorithm has strong precision/recall for retrieving *all occurrences* of a pattern P, given that the existence of at least one repetition of P has been established. So even if an algorithm fails to discover many noticeable/important patterns in a piece, it can still score well on the precision or recall occurrence metrics.

First this function calculates either the precision or recall occurrence matrix. This is calculated by taking the indices of those elements of the establishment matrix that are greater than some score threshlold (default .75), and calculating the precision or recall of the score matrix for $\mathcal{P}_i$ (all occurrences) and $\mathcal{Q}_j$ (all occurrences), for each pair (i, j) of indices. Then the precision or recall of the occurrence matrix is calculated (over this subset of columns/rows), to return a scalar value.

precision-matrix

Started, last checked	5/2/2013, 5/2/2013
Location	Robust metrics
Calls	add-to-list, first-n-naturals, fibonacci-list, max-item, my-last
Called by	establishment-metric, occurrence-matrix&rel-idx, occurrence-metric
Comments/see also	

Example:

```
(precision-matrix '((1 4/5 1/5) (2/5 1/5 2/5)))
--> 11/15
```

The precision of a matrix where rows represent ground truth items and columns represent retrieved items is defined as the mean of the column maxima.

recall-matrix

Started, last checked	5/2/2013, 5/2/2013
Location	Robust metrics
Calls	fibonacci-list, max-item, my-last
Called by	establishment-metric
Comments/see also	

Example:

```
(recall-matrix '((1 4/5 1/5) (2/5 1/5 2/5)))
--> 7/10
```

The recall of a matrix where rows represent ground truth items and columns represent retrieved items is defined as the mean of the row maxima.

score-matrix

Started, last checked	5/2/2013, 5/2/2013
Location	Robust metrics
Calls	cardinality-score, equalp-score, matching-score
Called by	establishment-metric, occurrence-matrix&rel-idx
Comments/see also	

Example:

```
(setq P1 '((1 53) (5/2 53) (3 40) (3 52) (4 50)))
(setq P2 '((21 60) (45/2 60) (23 47) (23 59) (25 57)))
(setq P3 '((41 53) (85/2 53) (43 52) (44 50)))
(setq P-occurrences (list P1 P2 P3))
(setq Q1 '((1 53) (5/2 53) (3 40) (3 52) (4 50)))
(setq Q2 (translation Q1 '(5 2)))
(setq Q3 '((0 59) (1/2 60) (1 60) (2 62)))
(setq Q4 (translation Q1 '(40 0)))
(setq Q-occurrences (list Q1 Q2 Q3 Q4))
(score-matrix P-occurrences Q-occurrences)
--> ((1 0 0 0) (0 0 0 0) (0 0 0 4/5))
(score-matrix
 P-occurrences Q-occurrences #'cardinality-score nil
 t)
--> ((1 1 1/5 1) (4/5 4/5 1/5 4/5) (4/5 4/5 1/4 4/5))
```

For each pair of point sets in the first and second arguments, this function computes the cardinality score or matching score (or some other function for computing symbolic musical similarity). These scores are returned in a list of lists. Each lists contains a row of scores for a point set from P-occurrences, and so the nth element of each list corresponds to a column of scores for a point set from Q-occurrences.

4.5.2 Evaluate discovered versus annotated patterns

The functions below are metrics for evaluating the extent to which different ontime-pitch pairs have been output by an algorithm.

metrics-for-algorithm&piece

Started, last checked	10/3/2013, 10/3/2013
Location	Evaluate discovered versus annotated patterns
Calls	establishment-metric, occurrence-metric, nth-list-of-lists, max-item, read-ground-truth-for-piece, read-patts&occs
Called by	pattern-discovery-metrics
Comments/see also	metrics-for-algorithm&piece-all-patt-all-occ

Example:

```
(setq
 *example-files-path*
 (merge-pathnames
  (make-pathname
   :directory
   '(:relative "MIREX 2013 pattern discovery task"))
  *MCStylistic-Aug2013-example-files-path*))
(setq
 *algorithm-path*
 (merge-pathnames
  (make-pathname
   :directory
   '(:relative
     "MIREX 2013 pattern discovery task"
     "algorithm3output"))
  *MCStylistic-Aug2013-example-files-path*))
(setq
 *piece-path*
 (merge-pathnames
  (make-pathname
   :directory
   '(:relative "groundTruth" "beethovenOp2No1Mvt3"))
  *jkuPattsDevDB-Aug2013-path*))
(metrics-for-algorithm&piece
 *algorithm-path* *piece-path*)
--> (.364 .571 .729 .967 .947 0.967)
```

This function lists a pattern collection (for example the patterns annotated in a fugue by Bruhn, or the patterns output by an algorithm), and loads all occurrences of all these patterns as nested lists.

metrics-for-algorithm&piece-all-patt-all-occ

Started, last checked	4/7/2013, 4/7/2013
Location	Evaluate discovered versus annotated patterns
Calls	establishment-metric, occurrence-metric, nth-list-of-lists, max-item, read-ground-truth-for-piece, read-patts&occs
Called by	pattern-discovery-metrics
Comments/see also	metrics-for-algorithm&piece

Example:

```
(setq
 *algorithm-path*
 (merge-pathnames
  (make-pathname
   :directory
   (list
    :relative "Example files"
    "MIREX 2013 pattern discovery task"
    "algorithm5output" "a_beethoven_piece"))
  *MCStylistic-Aug2013-path*))
(setq
 *piece-path*
 (merge-pathnames
  (make-pathname
   :directory
   (list
    :relative "Example files"
    "MIREX 2013 pattern discovery task"
    "exampleGroundTruth" "a_beethoven_piece"))
  *MCStylistic-Aug2013-path*))
(metrics-for-algorithm&piece-all-patt-all-occ
 *algorithm-path* *piece-path*)
--> (.364 .571 .729 .967 .947 0.967)
```

This function is a slight variant on metrics-for-algorithm&piece. It assumes that the entire algorithm output and for one piece (movement) is contained in one text file, in a nested list. Similarly for the ground truth.

pattern-discovery-metrics

Started, last checked	10/3/2013, 10/3/2013
Location	Evaluate discovered versus annotated patterns
Calls	metrics-for-algorithm&piece, my-last
Called by	
Comments/see also	

Example:

```
(setq
 *algorithms-output-root*
 (merge-pathnames
  (make-pathname
   :directory
   '(:relative "MIREX 2013 pattern discovery task"))
  *MCStylistic-Aug2013-example-files-path*))
(setq *task-version* "polyphonic")
(setq
 *annotations-poly*
 (list
  "bruhn" "barlowAndMorgensternRevised"
  "sectionalRepetitions" "schoenberg" "tomcollins"))
(setq
 *ground-truth-paths*
 (list
  (merge-pathnames
   (make-pathname
    :directory
    '(:relative "groundTruth" "beethovenOp2No1Mvt3"))
  *jkuPattsDevDB-Aug2013-path*)
  (merge-pathnames
   (make-pathname
    :directory
    '(:relative
      "groundTruth" "gibbonsSilverSwan1612"))
```

```
  *jkuPattsDevDB-Aug2013-path*)))
(setq
 *algorithm-output-paths*
 (firstn
  2
  (cl-fad:list-directory *algorithms-output-root*)))
; Save the calculated metrics to this csv file.
(setq
 *csv-save-path&name*
 (merge-pathnames
  (make-pathname
   :name "calculated-metrics" :type "csv")
  *MCStylistic-Aug2013-example-files-path*))
(setq
 *metrics-to-calculate*
 (list
  "precision" "recall" "precision-est-card"
  "recall-est-card" "precision-occ-card"
  "recall-occ-card"))
(setq
 *metric-parameters*
 (list
  (list "score-thresh" .75) (list "tolp" t)
  (list "translationp" nil) (list "card-limit" 150)))
(setq *file-type* "csv")
(setq
 *ans*
 (pattern-discovery-metrics
  *algorithm-output-paths* *ground-truth-paths*
  *csv-save-path&name* *task-version*
  *annotations-poly* *metrics-to-calculate*
  *metric-parameters* *file-type*))
--> Writes a file to the specified location.
```

This function loops over algorithm output and annotated ground truth patterns. It computes a list of metrics for each (algorithm output, annotation ground truth) pair, and according to an optional argument, writes the results to a csv table. If there is no algorithm output for a particular piece, it will be skipped and an empty row will appear in the table.

read-patts&occs

Started, last checked	10/3/2013, 10/3/2013
Location	Evaluate discovered versus annotated patterns
Calls	csv2dataset, read-from-file
Called by	read-ground-truth-for-piece, metrics-for-algorithm&piece
Comments/see also	

Example:

```
(setq
 *annotation-collection*
 (merge-pathnames
  (make-pathname
   :directory
   (list
    :relative "groundTruth" "bachBWV889Fg"
    *task-version* "repeatedPatterns" "bruhn"))
  *jkuPattsDevDB-Aug2013-path*))
(setq
 *pattern-paths*
 (remove-if-not
  #'directory-pathname-p
  (cl-fad:list-directory *annotation-collection*)))
(read-patts&occs *pattern-paths*)
--> ((((1 64) (2 60) (3 65) (4 56) (6 62) (7 59)
       (7 64) (8 60))
      ((21 76) (22 72) (23 77) (24 68) (26 74) (27 71)
       (27 76) (28 72))...
      ((66 60) (67 65) (68 56) (70 62) (71 59) (71 64)
       (72 60)))
     (((10 64) (10 62) (10 60) (10 59) (11 57) (11 60)
       (11 59) (11 57) (11 55) (12 54) (12 55) (12 57)
       (12 55) (12 54) (12 52) (13 51) (13 52) (13 54)
       (13 52) (13 51) (13 49) (14 47) (14 49) (14 51)
       (15 51) (15 49) (15 51) (16 52))
      ((22 69) (22 67) (22 65) (22 64) (23 62) (23 65)
       (23 64) (23 62) (23 60) (24 59) (24 60) (24 62)
       (24 60) (24 59) (24 57) (25 56) (25 57) (25 59)
```

```
       (25 57) (25 56) (25 54) (26 52) (26 54) (26 56)
       (27 56) (27 54) (27 56) (28 57))...
      ((88 50) (88 52) (88 53) (88 52) (88 50) (88 48)
       (89 47) (89 48) (89 50) (89 48) (89 47) (89 45)
       (90 43) (90 45) (90 47) (91 47) (91 45) (91 47)
       (92 48)))
     (((24 53) (25 47) (25 50) (26 44) (26 47) (27 52)
       (28 45))
      ((52 60) (53 54) (53 57) (54 51) (54 54) (55 59)
       (56 52))...
      ((88 69) (89 62) (89 65) (90 59) (90 74)
       (91 67))))
```

This function lists a pattern collection (for example the patterns annotated in a fugue by Bruhn, or the patterns output by an algorithm), and loads all occurrences of all these patterns as nested lists.

read-ground-truth-for-piece

Started, last checked	10/3/2013, 10/3/2013
Location	Evaluate discovered versus annotated patterns
Calls	read-patts&occs
Called by	metrics-for-algorithm&piece
Comments/see also	

Example:

```
(setq
 *annotations-poly*
 (list
  "bruhn" "barlowAndMorgensternRevised"
  "sectionalRepetitions" "schoenberg" "tomCollins"))
(setq
 *annotation-paths*
 (mapcar
  #'(lambda (x)
      (merge-pathnames
       (make-pathname
        :directory
        (list
```

```
        :relative "groundTruth" "beethovenOp2No1Mvt3"
        *task-version* "repeatedPatterns" x))
      *jkuPattsDevDB-Aug2013-path*))
  *annotations-poly*))
(read-ground-truth-for-piece *annotation-paths*)
--> (;pattern 1
     (
      ;occurrence 1
      ((-1 60) (-1 68)...(10 68))
      ...
      ;occurrence m1
      ((449 63) (449 72)..(460 72)))
     ;pattern 2
     (
      ;occurrence 1
      ((239 60) (240 53)...(251 65))
      ...
      ;occurrence m2
      ((414 53) (414 69)...(425 65)))
     ...
     ;pattern n
     (
      ;occurrence 1
      ((41 60) (41 68)...(159 53))
      ...
      ;occurrence mn
      ((437 60) (437 68)...(555 53))))
```

This function lists all annotated patterns for a piece (for example the patterns annotated in a sonata by Barlow and Morgenstern, and sectional repetitions), and loads all occurrences of all these patterns as nested lists.

4.5.3 Matching score

The functions below implement the symbolic fingerprinting process described by Arzt et al. (2012).

matching-lists-indices

Started, last checked	10/3/2013, 10/3/2013
Location	Matching score
Calls	constant-vector, positions, positions-last-within-c
Called by	matching-score-histogram
Comments/see also	

Example:

```
(setq X-tok '((4 0) (3 -1) (2 0)))
(setq
 Y-tok
 '((6 5) (2 2) (2 0) (2 1) (1 2) (3 -1) (2 0) (0 -1)
   (3 -1) (2 2) (2 0) (4 -1)))
(matching-lists-indices X-tok Y-tok)
--> ((1 1 2 2 2) (5 8 2 6 10))
```

This function takes two lists A and B as input. It returns two lists a and b of equal length as output. The output lists contain all indices such that $A(a) = B(b)$. There may be some lenience in checking the last dimension of each sublist for equality, depending on the variable flextime.

matching-score-histogram

Started, last checked	10/3/2013, 10/3/2013
Location	Matching score
Calls	histogram, matching-lists-indices, max-item, nth-list-of-lists
Called by	matching-score
Comments/see also	

Example:

```
(progn
  (setq
   X
   '((-1 81) (-3/4 76) (-1/2 85) (-1/4 81)
     (0 88) (1 57) (1 61) (1 64) (2 73)
     (9/4 69) (5/2 76) (11/4 73) (3 81) (4 45)
```

```
     (4 49) (4 52) (4 57) (5 61) (21/4 57)
     (11/2 64) (23/4 61) (6 57) (6 69) (7 54)
     (7 59) (7 63) (7 69) (8 51) (8 59) (8 66)
     (8 69) (9 52) (9 59) (9 66) (9 69)
     (10 40) (10 64) (10 68)))
  (setq
   Y-all
   (read-from-file
    (merge-pathnames
     (make-pathname
      :name "mutantBeethovenOp2No2Mvt3"
      :type "txt")
     *MCStylistic-Aug2013-example-files-data-path*)))
  (setq
   Y
   (orthogonal-projection-unique-equalp
    Y-all '(1 1 0 0 0)))
  (setq Y (firstn 250 Y))
  (setq
   X-fgp (symbolic-fingerprint X "query"))
  (setq
   Y-fgp (symbolic-fingerprint Y "mutant"))
  (matching-score-histogram X-fgp Y-fgp))
--> (0 6 715 7 0 0 0 0 0 0 0 0 0 0 2 0 0 0 0 0 0 0 0 1
     0 6 660 7 0 0 0 0 0 0 0 0 0 0 2 0 0 0 0 0 0 0 0 0
     0 8 158 4 1 0 0 0 0 0 0 0 0 0 2 0 0 0 0 0 0 0 0 1
     0 0 70)
```

This function takes two lists A and B as input. It returns two lists a and b of equal length as output. The output lists contain all indices such that $A(a) = B(b)$. There may be some lenience in checking the last dimension of each sublist for equality, depending on the variable flextime.

positions-last-within-c

Started, last checked	10/3/2013, 10/3/2013
Location	Matching score
Calls	my-last
Called by	matching-score-histogram
Comments/see also	

Example:

```
(positions-last-within-c
 '(4 0) '((0 1) (3 2) (4 0.1) (2 2) (-4 4) (4 0)) .25)
--> (2 5)
```

This is a very specific function. It checks for instances of a query in a longer list. All but the last elements are tested for equality, but the last element is allowed to be within an amount c of the query.

symbolic-fingerprint

Started, last checked	10/3/2013, 10/3/2013
Location	Matching score
Calls	nth-list-of-lists
Called by	matching-score
Comments/see also	

Example:

```
(setq
 dataset
 '((-1 81) (-3/4 76) (-1/2 85) (-1/4 81) (0 88)
   (1 57) (1 61) (1 64) (2 73) (9/4 69) (5/2 76)
   (11/4 73) (3 81) (4 45) (4 49) (4 52)))
(setq ID "beethovenOp2No2Mvt3")
(symbolic-fingerprint dataset ID)
--> (((81 76 85 1) ("beethovenOp2No2Mvt3" -1 1/4))
     ((81 76 81 2) ("beethovenOp2No2Mvt3" -1 1/4))
     ((81 76 88 3) ("beethovenOp2No2Mvt3" -1 1/4))
     ((81 76 57 7) ("beethovenOp2No2Mvt3" -1 1/4))
     ((81 76 61 7) ("beethovenOp2No2Mvt3" -1 1/4))
     ((81 85 81 1/2) ("beethovenOp2No2Mvt3" -1 1/2))
     ((81 85 88 1) ("beethovenOp2No2Mvt3" -1 1/2))
     ((81 85 57 3) ("beethovenOp2No2Mvt3" -1 1/2))
     ((81 85 61 3) ("beethovenOp2No2Mvt3" -1 1/2))
     ((81 85 64 3) ("beethovenOp2No2Mvt3" -1 1/2))
     ((81 81 88 1/3) ("beethovenOp2No2Mvt3" -1 3/4))
     ((81 81 57 5/3) ("beethovenOp2No2Mvt3" -1 3/4))
     ((81 81 61 5/3) ("beethovenOp2No2Mvt3" -1 3/4))
     ((81 81 64 5/3) ("beethovenOp2No2Mvt3" -1 3/4))
```

```
((81 81 73 3) ("beethovenOp2No2Mvt3" -1 3/4))
...
((76 73 52 5) ("beethovenOp2No2Mvt3" 5/2 1/4))
((76 81 45 2) ("beethovenOp2No2Mvt3" 5/2 1/2))
((76 81 49 2) ("beethovenOp2No2Mvt3" 5/2 1/2))
((76 81 52 2) ("beethovenOp2No2Mvt3" 5/2 1/2))
((73 81 45 4) ("beethovenOp2No2Mvt3" 11/4 1/4))
((73 81 49 4) ("beethovenOp2No2Mvt3" 11/4 1/4))
((73 81 52 4) ("beethovenOp2No2Mvt3" 11/4 1/4)))
```

Given a two-dimensional dataset consisting of ontimes and MIDI note numbers (or some other numeric representation of pitch), this function returns symbolic fingerprints as described by Arzt et al. (2012). For transposition-variant fingerprints, the format is

$$[m_1 : m_2 : m_3 : r] : \mathrm{ID} : t : d_{1,2}$$

for locally constrained combinations (controlled by n1, n2, and d) of successive MIDI notes, where m_1, m_2, and m_3 are MIDI note numbers, $d_{i,j}$ is the difference between the onsets of MIDI notes i and j, r is the fraction $d_{2,3}/d_{1,2}$, and t is a time stamp. For the transposition-invariant version, replace $[m_1 : m_2 : m_3]$ by $[m_2 - m_1 : m_3 - m_2]$. The tokens $[m_1 : m_2 : m_3 : r]$ are stored as the first of a pair and the rest of the fingerprint as the second of a pair, in the list of lists that is returned.

4.6 Markov models

4.6.1 Segmentation

The fundamental functions here are used to segment datapoints based on ontime and offtime. Subsequent functions do things like computing chord spacing and holding types.

append-offtimes

Started, last checked	16/1/2009, 24/1/2009
Location	Segmentation
Calls	
Called by	segments, segments-strict
Comments/see also	

Example:

```
(append-offtimes '((0 48 2) (1 60 1) (1 57 1/2)))
--> '((0 48 2 2) (1 60 1 2) (1 57 1/2 3/2))
```

This function takes a list, assumed to be datapoints, and appends the offset of each datapoint as the final item.

chord-candidates-offtimes

Started, last checked	16/1/2009, 24/1/2009
Location	Segmentation
Calls	my-last
Called by	segment
Comments/see also	Possibly obsolete.

Example:

```
(chord-candidates-offtimes
 '((1579 66 191 2 49 1770 5) (1974 64 191 3 49 2165 2)
   (1974 67 191 2 49 2165 3) (2368 66 191 2 49 2559 0)
   (2368 62 191 3 49 2559 4) (2763 64 191 2 49 2954 6)
   (2763 57 191 3 49 2954 7) (2800 72 191 1 49 2991 8)
   (3158 38 191 4 49 3349 9)
   (1579 62 1920 3 49 3499 1)
   (3158 54 385 3 49 3543 10)
   (3158 62 385 2 49 3543 11)
   (3553 42 191 4 49 3744 12)
   (3947 45 191 4 49 4138 13)
   (4342 50 191 4 49 4533 14)) 15 2368 0)
--> (5 2 3)
```

There are four arguments to this function: a list of datapoints (ordered by offtimes ascending and appended with an enumeration), the length l of the list, a point in time x, and an index s from which to begin searching. When the nth offtime equals or exceeds x, the search stops. As subsequent calls to this function use larger values of x, the search can begin at the sth offtime.

chord-candidates-offtimes-strict

Started, last checked	16/1/2009, 24/1/2009
Location	Segmentation
Calls	my-last
Called by	segment-strict
Comments/see also	Possibly obsolete.

Example:

```
(chord-candidates-offtimes-strict
 '((3 55 57 3/4 1 15/4 1) (3 60 60 3/4 1 15/4 2)
   (3 67 64 3/4 0 15/4 3) (3 76 69 3/4 0 15/4 4)
   (15/4 59 59 1/4 1 4 5) (15/4 65 63 1/4 0 4 6)
   (15/4 74 68 1/4 0 4 7) (3 48 53 3 1 6 0)
   (4 60 60 2 1 6 8) (4 64 62 2 0 6 9)
   (4 72 67 2 0 6 10)) 11 15/4 0)
--> (1 2 3 4)
```

Contrast the output of this function with the function chord-candidate-offtimes. The difference is that the present function will not return indices of datapoints whose offtimes coincide with the provided time x. There are four arguments to this function: a list of datapoints (ordered by offtimes ascending and appended with an enumeration), the length l of the list, a point in time x, and an index s from which to begin searching. When the nth offtime equals or exceeds x, the search stops. As subsequent calls to this function use larger values of x, the search can begin at the sth offtime.

chord-candidates-ontimes

Started, last checked	16/1/2009, 24/1/2009
Location	Segmentation
Calls	my-last
Called by	segment, segment-strict
Comments/see also	Possibly obsolete.

Example:

```
(chord-candidates-ontimes
 '((1579 62 1920 3 49 3499 1)
   (1579 66 191 2 49 1770 5) (1974 64 191 3 49 2165 2)
   (1974 67 191 2 49 2165 3) (2368 66 191 2 49 2559 0)
```

```
   (2368 62 191 3 49 2559 4) (2763 64 191 2 49 2954 6)
   (2763 57 191 3 49 2820 7) (2800 72 191 1 49 2991 8)
   (3158 38 191 4 49 3349 9)
   (3158 54 385 3 49 3543 10)
   (3158 62 385 2 49 3543 11)
   (3553 42 191 4 49 3744 12)
   (3947 45 191 4 49 4138 13)
   (4342 50 191 4 49 4533 14)) 15 2368 0)
--> (1 5 2 3 0 4)
```

There are four arguments to this function: a list of datapoints (ordered by ontimes and appended with offtimes and an enumeration), the length l of the list, a point in time x, and an index s from which to begin searching. When the nth ontime exceeds x, the search stops. As subsequent calls to this function use larger values of x, the search can begin at the sth ontime.

enumerate-append

Started, last checked	16/1/2009, 24/1/2009
Location	Segmentation
Calls	
Called by	segments, segments-strict
Comments/see also	

Example:

```
(enumerate-append '((3 53) (6 0) (42 42)))
--> ((3 53 0) (6 0 1) (42 42 2))
```

This function enumerates a list by appending the next natural number, counting from 0, to the end of each list.

points-belonging-to-time-interval

Started, last checked	10/9/2014, 10/9/2014
Location	Segmentation
Calls	
Called by	segments-strict
Comments/see also	

Example:

```
(setq time-interval '(1 2))
(points-belonging-to-time-interval
 '((0 53 56 1 "h" 1 0) (0 60 60 3/2 "h" 3/2 1)
   (1/2 72 67 5/2 "h" 3 2) (5/4 53 56 1/2 "h" 7/4 3)
   (3/2 60 60 1 "h" 5/2 4) (3 60 60 1 "h" 4 5))
   time-interval)
-->((0 60 60 3/2 "h" 3/2 1) (1/2 72 67 5/2 "h" 3 2)
    (5/4 53 56 1/2 "h" 7/4 3) (3/2 60 60 1 "h" 5/2 4))
```

This function returns points with (ontime, offtime) pairs (x_i, y_i) such that for a given time interval $[a, b)$, we have $x_i < b$ and $y_i > a$.

prepare-for-segments

Started, last checked	16/1/2009, 24/1/2009
Location	Segmentation
Calls	sort-by
Called by	segments
Comments/see also	Possibly obsolete.

Example:

```
(prepare-for-segments
 '((2368 66 191 2 49 2559 0)
   (1579 62 1920 3 49 3499 1)
   (1974 64 191 3 49 2165 2) (1974 67 191 2 49 2165 3)
   (2368 62 191 3 49 2559 4) (1579 66 191 2 49 1770 5)
   (2763 64 191 2 49 2954 6) (2763 57 191 3 49 2954 7)
   (2800 72 191 1 49 2991 8) (3158 38 191 4 49 3349 9)
   (3158 54 385 3 49 3543 10)
   (3158 62 385 2 49 3543 11)
   (3553 42 191 4 49 3744 12)
   (3947 45 191 4 49 4138 13)
   (4342 50 191 4 49 4533 14)))
--> (((1579 62 1920 3 49 3499 1)
      (1579 66 191 2 49 1770 5)
      (1974 64 191 3 49 2165 2)
      (1974 67 191 2 49 2165 3)
      (2368 66 191 2 49 2559 0)
      (2368 62 191 3 49 2559 4)
      (2763 64 191 2 49 2954 6)
```

```
 (2763 57 191 3 49 2820 7)
 (2800 72 191 1 49 2991 8)
 (3158 38 191 4 49 3349 9)
 (3158 54 385 3 49 3543 10)
 (3158 62 385 2 49 3543 11)
 (3553 42 191 4 49 3744 12)
 (3947 45 191 4 49 4138 13)
 (4342 50 191 4 49 4533 14))
((1579 66 191 2 49 1770 5)
 (1974 64 191 3 49 2165 2)
 (1974 67 191 2 49 2165 3)
 (2368 66 191 2 49 2559 0)
 (2368 62 191 3 49 2559 4)
 (2763 64 191 2 49 2954 6)
 (2763 57 191 3 49 2954 7)
 (2800 72 191 1 49 2991 8)
 (3158 38 191 4 49 3349 9)
 (1579 62 1920 3 49 3499 1)
 (3158 54 385 3 49 3543 10)
 (3158 62 385 2 49 3543 11)
 (3553 42 191 4 49 3744 12)
 (3947 45 191 4 49 4138 13)
 (4342 50 191 4 49 4533 14)))
```

The datapoints already have offtimes appended and are enumerated. They are sent to two lists; one ordered by ontime, the other by offtime.

segment

Started, last checked	16/1/2009, 24/1/2009
Location	Segmentation
Calls	add-to-list, chord-candidates-offtimes, chord-candidates-ontimes, first-n-naturals, nth-list
Called by	segments
Comments/see also	Possibly obsolete.

Example:

```
(segment
 1579
```

```
 '((2368 66 191 2 49) (1579 62 1920 3 49)
   (1974 64 191 3 49) (1974 67 191 2 49)
   (2368 62 191 3 49) (1579 66 191 2 49)
   (2763 64 191 2 49) (2763 57 191 3 49)
   (2800 72 191 1 49) (3158 38 191 4 49)
   (3158 54 385 3 49) (3158 62 385 2 49)
   (3553 42 191 4 49) (3947 45 191 4 49)
   (4342 50 191 4 49))
 15
 '(((1579 62 1920 3 49 3499 1)
    (1579 66 191 2 49 1770 5)
    (1974 64 191 3 49 2165 2)
    (1974 67 191 2 49 2165 3)
    (2368 66 191 2 49 2559 0)
    (2368 62 191 3 49 2559 4)
    (2763 64 191 2 49 2954 6)
    (2763 57 191 3 49 2820 7)
    (2800 72 191 1 49 2991 8)
    (3158 38 191 4 49 3349 9)
    (3158 54 385 3 49 3543 10)
    (3158 62 385 2 49 3543 11)
    (3553 42 191 4 49 3744 12)
    (3947 45 191 4 49 4138 13)
    (4342 50 191 4 49 4533 14))
   ((1579 66 191 2 49 1770 5)
    (1974 64 191 3 49 2165 2)
    (1974 67 191 2 49 2165 3)
    (2368 66 191 2 49 2559 0)
    (2368 62 191 3 49 2559 4)
    (2763 64 191 2 49 2954 6)
    (2763 57 191 3 49 2954 7)
    (2800 72 191 1 49 2991 8)
    (3158 38 191 4 49 3349 9)
    (1579 62 1920 3 49 3499 1)
    (3158 54 385 3 49 3543 10)
    (3158 62 385 2 49 3543 11)
    (3553 42 191 4 49 3744 12)
    (3947 45 191 4 49 4138 13)
    (4342 50 191 4 49 4533 14))))
--> (((1579 66 191 2 49) (1579 62 1920 3 49))
     (1 5) (14 13 12 11 10 9 8 7 6 5 4 3 2 1 0))
```

This function takes an ontime t, a list of datapoints of length N, with offsets and enumeration appended, but in original order, as well as the datapoints having had the function prepared-for- segments applied. It returns any datapoints which exist at the point t, as well as lists which help speed up any subsequent searches.

segment-strict

Started, last checked	16/1/2009, 24/1/2009
Location	Segmentation
Calls	add-to-list, chord-candidates-offtimes-strict, chord-candidates-ontimes, first-n-naturals, nth-list
Called by	
Comments/see also	Possibly obsolete.

Example:

```
(segment-strict
 15/4
 '((3 48 53 3 1 6 0) (3 55 57 3/4 1 15/4 1)
   (3 60 60 3/4 1 15/4 2) (3 67 64 3/4 0 15/4 3)
   (3 76 69 3/4 0 15/4 4) (15/4 59 59 1/4 1 4 5)
   (15/4 65 63 1/4 0 4 6) (15/4 74 68 1/4 0 4 7)
   (4 60 60 2 1 6 8) (4 64 62 2 0 6 9)
   (4 72 67 2 0 6 10))
 11
 '(((3 48 53 3 1 6 0) (3 55 57 3/4 1 15/4 1)
    (3 60 60 3/4 1 15/4 2) (3 67 64 3/4 0 15/4 3)
    (3 76 69 3/4 0 15/4 4) (15/4 59 59 1/4 1 4 5)
    (15/4 65 63 1/4 0 4 6) (15/4 74 68 1/4 0 4 7)
    (4 60 60 2 1 6 8) (4 64 62 2 0 6 9)
    (4 72 67 2 0 6 10))
   ((3 55 57 3/4 1 15/4 1) (3 60 60 3/4 1 15/4 2)
    (3 67 64 3/4 0 15/4 3) (3 76 69 3/4 0 15/4 4)
    (15/4 59 59 1/4 1 4 5) (15/4 65 63 1/4 0 4 6)
    (15/4 74 68 1/4 0 4 7) (3 48 53 3 1 6 0)
    (4 60 60 2 1 6 8) (4 64 62 2 0 6 9)
    (4 72 67 2 0 6 10))))
--> (((15/4 74 68 1/4 0 4 7) (15/4 65 63 1/4 0 4 6)
      (15/4 59 59 1/4 1 4 5) (3 48 53 3 1 6 0))
```

```
    (0 1 2 3 4 5 6 7) (10 9 8 7 6 5 0))
```

This function uses the function chord-candidate-offtimes-strict instead of chord-candidate-offtimes, and performs a sort on the output, according to the optional argument sort-index. The function takes an ontime t, a list of datapoints of length N, with offsets and enumeration appended, but in original order, as well as the datapoints having had the function prepared-for-segments applied. It returns any datapoints which exist at the point t, as well as lists which help speed up any subsequent searches.

segments

Started, last checked	16/1/2009, 24/1/2009
Location	Segmentation
Calls	add-to-list, append-offtimes, enumerate-append, first-n-naturals, nth-list-of-lists, prepare-for-segments
Called by	
Comments/see also	Possibly obsolete.

Example:

```
(segments
 '((2368 66 191 2 49) (1579 62 1920 3 49)
   (1974 64 191 3 49) (1974 67 191 2 49)
   (2368 62 191 3 49) (1579 66 191 2 49)
   (2763 64 191 2 49) (2763 57 191 3 49)
   (2800 72 191 1 49) (3158 38 191 4 49)
   (3158 54 385 3 49) (3158 62 385 2 49)
   (3553 42 191 4 49) (3947 45 191 4 49)
   (4342 50 191 4 49)))
--> ((1579 ((1579 66 191 2 49 1770 5)
    (1579 62 1920 3 49 3499 1)))
     (1770 ((1579 66 191 2 49 1770 5)
    (1579 62 1920 3 49 3499 1)))
     (1974 ((1974 67 191 2 49 2165 3)
    (1974 64 191 3 49 2165 2)
    (1579 62 1920 3 49 3499 1)))
     (2165 ((1974 67 191 2 49 2165 3)
    (1974 64 191 3 49 2165 2)
    (1579 62 1920 3 49 3499 1)))
```

```
 (2368 ((2368 62 191 3 49 2559 4)
(2368 66 191 2 49 2559 0)
(1579 62 1920 3 49 3499 1)))
 (2559 ((2368 62 191 3 49 2559 4)
(2368 66 191 2 49 2559 0)
(1579 62 1920 3 49 3499 1)))
 (2763 ((2763 57 191 3 49 2954 7)
(2763 64 191 2 49 2954 6)
(1579 62 1920 3 49 3499 1)))
 (2800 ((2800 72 191 1 49 2991 8)
(2763 57 191 3 49 2954 7)
(2763 64 191 2 49 2954 6)
(1579 62 1920 3 49 3499 1)))
 (2954 ((2800 72 191 1 49 2991 8)
(2763 57 191 3 49 2954 7)
(2763 64 191 2 49 2954 6)
(1579 62 1920 3 49 3499 1)))
 (2991 ((2800 72 191 1 49 2991 8)
(1579 62 1920 3 49 3499 1)))
 (3158 ((3158 62 385 2 49 3543 11)
(3158 54 385 3 49 3543 10)
(3158 38 191 4 49 3349 9)
(1579 62 1920 3 49 3499 1)))
 (3349 ((3158 62 385 2 49 3543 11)
(3158 54 385 3 49 3543 10)
(3158 38 191 4 49 3349 9)
(1579 62 1920 3 49 3499 1)))
 (3499 ((3158 62 385 2 49 3543 11)
(3158 54 385 3 49 3543 10)
(1579 62 1920 3 49 3499 1)))
 (3543 ((3158 62 385 2 49 3543 11)
(3158 54 385 3 49 3543 10)))
 (3553 ((3553 42 191 4 49 3744 12)))
 (3744 ((3553 42 191 4 49 3744 12)))
 (3947 ((3947 45 191 4 49 4138 13)))
 (4138 ((3947 45 191 4 49 4138 13)))
 (4342 ((4342 50 191 4 49 4533 14)))
 (4533 ((4342 50 191 4 49 4533 14))))
```

This function takes a list of datapoints as its argument. First it creates a variable containing the distinct times (on and off) of the datapoints. Then it returns the segment for each of these times.

segments-strict

Started, last checked	16/1/2009, 10/9/2014
Location	Segmentation
Calls	append-offtimes, enumerate-append, my-last, nth-list-of-lists, points-belonging-to-time-interval
Called by	
Comments/see also	More efficient implementations welcome.

Example:

```
(segments-strict
 '((3 48 53 3 1) (3 67 64 3/4 0) (3 76 69 3/4 0)
   (15/4 65 63 1/4 0) (15/4 74 68 1/4 0) (4 64 62 2 0)
   (4 72 67 2 0) (13/2 61 60 1/2 0) (7 62 61 1/2 0)
   (15/2 64 62 1/2 0) (8 50 54 1 1) (8 65 63 1 0))
 1 3)
--> ((3 ((3 48 53 3 1 6 0) (3 67 64 3/4 0 15/4 1)
         (3 76 69 3/4 0 15/4 2)))
     (15/4 ((3 48 53 3 1 6 0) (15/4 65 63 1/4 0 4 3)
            (15/4 74 68 1/4 0 4 4)))
     (4 ((3 48 53 3 1 6 0) (4 64 62 2 0 6 5)
         (4 72 67 2 0 6 6)))
     (6 NIL)
     (13/2 ((13/2 61 60 1/2 0 7 7)))
     (7 ((7 62 61 1/2 0 15/2 8)))
     (15/2 ((15/2 64 62 1/2 0 8 9)))
     (8 ((8 50 54 1 1 9 10) (8 65 63 1 0 9 11)))
     (9 NIL))
```

This function takes a list of points (assumed to be in lexicographic order) as its only mandatory argument. It returns $(\text{timepoint}_i, \text{point set}_i)$ pairs such that the points belonging to point set_i sound during the time interval $[\text{timepoint}_i, \text{timepoint}_{i+1})$.

Originally this function was based on the function segments, but it was very slow and so has been rewritten. The earlier function did not re-sort points in the segment point sets, meaning point set_i might not be in lexicographic order. In the most recent version these point sets are lexicographic. It is not clear whether this will have any knock-on effects.

4.6.2 Markov analyse

The functions here are designed to analyse data according to a Markov-chain model. At present the code handles a first-order analysis. The aim is to build a state transition matrix for some variables (referenced by variable-names and catalogue). Hence, the variable variable-names points to some actual data (note the use of the function symbol-value) which is indexed by the variable catalogue. Using the function write-to-file, the information can be sent to a text file, to avoid having to display it.

accumulate-to-stm

Started, last checked	27/1/2009, 24/8/2010
Location	Markov analyse
Calls	
Called by	present-to-stm
Comments/see also	accumulate-to-stm<-, add-to-stm

Example:

```
(accumulate-to-stm
 '(((4) ("Piece A")) ((2) ("Piece A")))
 '((4) (((1) ("Piece B")) ((2) ("Piece C"))))
 '(((1) (((5) ("Piece A")) ((4) ("Piece B"))))
   ((2) (((5) ("Piece A"))))
   ((4) (((1) ("Piece B"))
         ((2) ("Piece C"))))
   ((5) (((4) ("Piece B"))))))
--> '(((4) (((2) ("Piece A")) ((1) ("Piece B"))
    ((2) ("Piece C"))))
      ((1) (((5) ("Piece A")) ((4) ("Piece B"))))
      ((2) (((5) ("Piece A"))))
      ((5) (((4) ("Piece B")))))
```

The first argument is a listed state; the second is the relevant row of the state transition matrix; the third is the state transition matrix itself. This function is called when the state of the first item of the listed state has appeared in the state transition matrix before. The references of the event are included.

add-to-stm

Started, last checked	27/1/2009, 24/8/2010
Location	Markov analyse
Calls	
Called by	present-to-stm
Comments/see also	accumulate-to-stm, add-to-stm<-

Example:

```
(add-to-stm
 '(((3) ("Piece A")) ((4) ("Piece A")))
 '(((1) (((5) ("Piece A")) ((4) ("Piece B"))))
   ((2) (((5) ("Piece A"))))
   ((4) (((1) ("Piece B")) ((2) ("Piece C"))))
   ((5) (((4) ("Piece B"))))))
--> '(((3) (((4) ("Piece A"))))
      ((1) (((5) ("Piece A")) ((4) ("Piece B"))))
      ((2) (((5) ("Piece A"))))
      ((4) (((1) ("Piece B")) ((2) ("Piece C"))))
      ((5) (((4) ("Piece B")))))
```

The first argument is a listed state; the second is the state transition matrix. This function is called when the state of the first item of the listed state has not appeared in the state transition matrix before. It is added.

construct-initial-states

Started, last checked	27/1/2009, 24/8/2010
Location	Markov analyse
Calls	beat-spacing-states, firstn, spacing-holding-states
Called by	
Comments/see also	construct-final-states, construct-stm, construct-stm<-

Example:

```
(progn
  (setq
   variable-1
   '((0 30 1 1 84) (0 33 1 1 84) (1 40 1 1 84)
```

```
      (1 41 1 1 84)))
   (setq
    variable-2
    '((0 60 1 1 84) (0 63 1 1 84) (1 62 1 1 84)
      (1 63 1 1 84)))
   (setq *variable-names* '(variable-1 variable-2))
   (setq *catalogue* '("variable-1" "variable-2"))
   (construct-initial-states
    *variable-names* *catalogue*))
--> '((((3) (0 0))
        (NIL 1 "variable-1"
     ((0 30 1 1 84 1 0) (0 33 1 1 84 1 1))))
       (((3) (0 0))
        (NIL 1 "variable-2"
     ((0 60 1 1 84 1 0) (0 63 1 1 84 1 1)))))
```

This recursion analyses one variable name at a time, taking a catalogue name from the variable catalogue, and outputs initial states accordingly.

construct-internal-states

Started, last checked	7/1/2015, 28/5/2015
Location	Markov analyse
Calls	kern-file2phrase-boundary-states
Called by	
Comments/see also	construct-initial-states, construct-final-states

Example:

```
(setq
 *catalogue*
 (list "bachChoraleBWV411R246" "bachChoraleBWV4p8R184"))
(setq
 *kern-path&names*
 (loop for s in *catalogue* collect
   (merge-pathnames
    (make-pathname
     :directory
     (list :relative "bachChorales" s "kern")
     :name s :type "krn")
```

```
    *MCStylistic-MonthYear-data-path*)))
(setq *too-close* 1)
(setq
 internal-initial-states
 (construct-internal-states
  *kern-path&names* *catalogue* "fermata ending"
  *too-close* "beat-rel-MNN-states" 4))
--> (((3 (-12 7 12 16))
      ("bachChoraleBWV411R246"
       ((6 43 50 2 3) (6 62 61 2 2) (6 67 64 2 1)
        (6 71 66 2 0)) (55 57) (1 0)))
     ((3 (-5 2 11 19))
      ("bachChoraleBWV411R246"
       ((14 50 54 2 3) (14 57 58 2 2) (14 66 63 2 1)
        (14 74 68 2 0)) (55 57) (1 0)))
     ...
     ((3 (-7 0 9))
      ("bachChoraleBWV4p8R184"
       ((54 46 52 2 3) (54 53 56 2 2) (54 62 61 2 0)
        (54 62 61 2 1)) (53 56) (2 5)))).
```

This function identifies events in kern files where phrases begin/end or where there are fermata. It calculates the state representations for these events and outputs them in a list.

construct-stm

Started, last checked	27/1/2009, 24/8/2010
Location	Markov analyse
Calls	beat-spacing-states, markov-analyse, spacing-holding-states
Called by	
Comments/see also	construct-final-states, construct-initial-states, construct-stm<-

Example:

```
(progn
  (setq
   variable-1
   '((0 30 1 1 84) (0 33 1 1 84) (1 40 1 1 84)
```

```
    (1 41 1 1 84)))
  (setq
   variable-2
   '((0 60 1 1 84) (0 63 1 1 84) (1 62 1 1 84)
     (1 63 1 1 84)))
  (setq *variable-names* '(variable-1 variable-2))
  (setq *catalogue* '("variable-1" "variable-2"))
  (construct-stm
   *variable-names* *catalogue* "beat-spacing-states"
   2 4 1)
--> "Finished"
*transition-matrix*
--> (((1 (3))
      (((2 (1))
        (2 0 "variable-2"
         ((1 62 1 1 84 2 2) (1 63 1 1 84 2 3))))
       ((2 (1))
        (10 0 "variable-1"
         ((1 40 1 1 84 2 2) (1 41 1 1 84 2 3)))))))
```

This recursion analyses one variable name at a time, taking a catalogue name from the variable catalogue, and updates the transition matrix accordingly. The output "Finished" is preferable to the transition matrix, which is large enough that it can cause the Listener to crash.

firstn-list

Started, last checked	27/1/2009, 24/8/2010
Location	Markov analyse
Calls	firstn
Called by	markov-analyse
Comments/see also	

Example:

```
(firstn-list 3 '(1 2 3 4 5)
--> '((1 2 3) (2 3 4) (3 4 5))
```

This function applies the function firstn recursively to a list. It is like producing an n- gram, and is useful for building a first-order Markov model. I call the output 'listed states'.

kern-file2phrase-boundary-states

Started, last checked	7/1/2015, 28/5/2015
Location	Markov analyse
Calls	articulation-points, beat-rel-MNN-states, kern-file2points-artic-dynam-lyrics, remove-duplicates&values-too-close, segments-strict
Called by	construct-internal-states
Comments/see also	

Example:

```
(setq
 kern-path&name
 (merge-pathnames
  (make-pathname
   :name "C-6-1-small" :type "krn")
  *MCStylistic-MonthYear-example-files-data-path*))
(kern-file2phrase-boundary-states kern-path&name)
--> (((4 (9))
      ("C-6-1-small" ((-1 66 63 5/3 0)) (57 58) (6 5)))
     ((19/4 (-15 9))
      ("C-6-1-small" ((3 42 49 1 1) (15/4 66 63 1/4 0))
       (57 58) (6 5)))).
```

This function imports a kern file and searches for notes that have articulation associated with phrase beginnings or endings, with the aim of returning states that will be appropriate for using as initial or final internal states. The first optional argument phrase-str can be called with 'phrase beginning', 'phrase ending', 'fermata beginning', or 'fermata ending'. Phrase beginnings are indicated by `(` in a kern file, endings by `)`, and fermata by `;`. Fermata indicate the end of a phrase, and therefore the next state will be the beginning of the next phrase. So if this function is called with phrase-str equal to 'fermata beginning', the next state(s) following fermata will be returned.

Sometimes fermata signs appear in close succession (for instance imagine the tenor sings A3 held over from the fourth beat of the bar to the first beat of the next bar, resolving to G♯3 on beat two, whilst the bass, alto, and soprano sing E3, E4, B4 resepctively on beat one, and suppose further that fermata are written on these three notes and the resolving G♯3. Then there will be three fermata at ontime x and one at ontime $x+1$. The end of the phrase is not at ontime x but at ontime $x+1$. This function will remove fermata ontimes that are too close together—in our example removing the fermata ontime x. This functionality is controlled by the optional argument too-close.

markov-analyse

Started, last checked	27/1/2009, 24/8/2010
Location	Markov analyse
Calls	update-stm
Called by	construct-stm
Comments/see also	markov-analyse<-

Example:

```
(markov-analyse
 '(((3) ("Piece A")) ((6) ("Piece A"))
   ((4) ("Piece A")) ((4) ("Piece A"))
   ((3) ("Piece A")) ((2) ("Piece A"))))
--> "Finished"
*transition-matrix*
--> '(((3) (((2) ("Piece A")) ((6) ("Piece A"))))
      ((4) (((3) ("Piece A")) ((4) ("Piece A"))))
      ((6) (((4) ("Piece A")))))
```

This function has one argument - some states which are to be analysed according to a first-order Markov model. Note the need to define a variable here, `*transition-matrix*`. The output "Finished" is preferable to the transition matrix, which is large enough that it can cause the Listener to crash.

present-to-stm

Started, last checked	27/1/2009, 24/8/2010
Location	Markov analyse
Calls	add-to-stm, accumulate-to-stm
Called by	update-stm
Comments/see also	present-to-stm<-

Example:

```
(present-to-stm
 '(((4) ("Piece A")) ((2) ("Piece A")))
```

```
 '(((1) (((5) ("Piece A")) ((4) ("Piece B"))))
   ((2) (((5) ("Piece A"))))
   ((4) (((1) ("Piece B")) ((2) ("Piece C"))))
   ((5) (((4) ("Piece B"))))))
--> '(((4) (((2) ("Piece A")) ((1) ("Piece B"))
    ((2) ("Piece C"))))
      ((1) (((5) ("Piece A")) ((4) ("Piece B"))))
      ((2) (((5) ("Piece A"))))
      ((5) (((4) ("Piece B"))))).
```

This function calls either the function accumulate-to-stm, or add-to-stm, depending on whether the first argument, a listed-state, has appeared in the second argument, a state-transition matrix, before. The example above results in accumulate-to-stm being called, as the state of (4) has occurred before. However, changing this state to (3) in the argument would result in add-to-stm being called.

remove-duplicates&values-too-close

Started, last checked	7/1/2015, 28/5/2015
Location	Markov analyse
Calls	
Called by	kern-file2points-artic-dynam-lyrics
Comments/see also	

Example:

```
(setq times '(2 2 4 7 7.5 7.75 8))
(remove-duplicates&values-too-close times)
--> (2 4 8).
```

This function takes a list of floats as input, assumed to be ordered ascending. It will remove any duplicates from that list, as well as removing any elements that are too close in value (controlled by the second optional argument), leaving the largest.

update-stm

Started, last checked	27/1/2009, 24/8/2010
Location	Markov analyse
Calls	present-to-stm
Called by	markov-analyse
Comments/see also	update-stm<-

Example:

```
(update-stm
 '((((3) ("Piece A")) ((6) ("Piece A")))
   (((6) ("Piece A")) ((4) ("Piece A")))
   (((4) ("Piece A")) ((4) ("Piece A")))
   (((4) ("Piece A")) ((3) ("Piece A")))
   (((3) ("Piece A")) ((2) ("Piece A")))))
--> '(((3) (((2) ("Piece A")) ((6) ("Piece A"))))
      ((4) (((3) ("Piece A")) ((4) ("Piece A"))))
      ((6) (((4) ("Piece A")))))
```

This function has as its argument listed states, and it applies the function present-to-stm recursively to these listed states. The variable `*transition-matrix*` is updated as it proceeds.

4.6.3 Markov analyse backwards

The functions here are very similar to those contained in the file markov-analyse.lisp. Whereas those functions are designed to analyse data according to a Markov-chain model that runs forward in time, these functions do the same for going backwards in time.

accumulate-to-stm<-

Started, last checked	4/10/2010, 4/10/2010
Location	Markov analyse backwards
Calls	
Called by	present-to-stm<-
Comments/see also	accumulate-to-stm, add-to-stm<-

Example:

```
(accumulate-to-stm<-
 '(((4) ("Piece A")) ((2) ("Piece A")))
 '((2) (((5) ("Piece A"))))
 '(((1) (((5) ("Piece A")) ((4) ("Piece B"))))
   ((2) (((5) ("Piece A"))))
   ((4) (((1) ("Piece B")) ((2) ("Piece C"))))
   ((5) (((4) ("Piece B"))))))
--> '(((2) (((4) ("Piece A")) ((5) ("Piece A"))))
      ((1) (((5) ("Piece A")) ((4) ("Piece B"))))
      ((4) (((1) ("Piece B")) ((2) ("Piece C"))))
      ((5) (((4) ("Piece B")))))
```

This function is similar to the function accumulate-to-stm, the difference being X_n is looked up and X_{n-1} accumulated to the state- transition matrix. The first argument is a listed state; the second is the relevant row of the state transition matrix; the third is the state transition matrix itself. This function is called when the state of the second item of the listed state has appeared in the state transition matrix before. The references of the event are included.

add-to-stm<-

Started, last checked	4/10/2010, 4/10/2010
Location	Markov analyse backwards
Calls	
Called by	present-to-stm<-
Comments/see also	accumulate-to-stm<-, add-to-stm

Example:

```
(add-to-stm<-
 '(((4) ("Piece A")) ((3) ("Piece A")))
 '(((1) (((5) ("Piece A")) ((4) ("Piece B"))))
   ((2) (((5) ("Piece A"))))
   ((4) (((1) ("Piece B")) ((2) ("Piece C"))))
   ((5) (((4) ("Piece B"))))))
--> '(((3) (((4) ("Piece A"))))
      ((1) (((5) ("Piece A")) ((4) ("Piece B"))))
      ((2) (((5) ("Piece A"))))
      ((4) (((1) ("Piece B")) ((2) ("Piece C"))))
      ((5) (((4) ("Piece B")))))
```

This function is similar to the function add-to-stm, the difference being X_n is looked up and X_{n-1} added to the state-transition matrix. The first argument is a listed state; the second is the state transition matrix. This function is called when the state of the first item of the listed state has not appeared in the state transition matrix before. It is added.

construct-final-states

Started, last checked	4/10/2010, 4/10/2010
Location	Markov analyse backwards
Calls	beat-spacing-states, lastn, spacing-holding-states
Called by	
Comments/see also	construct-initial-states, construct-stm, construct-stm<-

Example:

```
(progn
  (setq
   variable-1
   '((0 60 60 2 0) (0 63 62 1 0) (1 67 64 2 0)
     (2 59 59 1 0)))
  (setq
   variable-2
   '((0 64 62 1 0) (1 62 61 1 0) (1 69 65 2 0)
     (2 66 63 1 0)))
  (setq *variable-names* '(variable-1 variable-2))
  (setq *catalogue* '("variable-1" "variable-2"))
  (construct-final-states
   *variable-names* *catalogue* "beat-spacing-states"
   10 3 3 1))
--> '(((3 (8))
       (NIL NIL "variable-1"
        ((2 59 59 1 0 3 3) (1 67 64 2 0 3 2))))
      ((3 (3))
       (NIL NIL "variable-2"
        ((2 66 63 1 0 3 3) (1 69 65 2 0 3 2)))))
```

This function is similar to the function construct-stm, the difference being X_n is looked up and X_{n-1} is accumulated or added to a state- transition matrix.

This recursion analyses one variable name at a time, taking a catalogue name from the variable catalogue, and outputs final states accordingly.

construct-stm<-

Started, last checked	4/10/2010, 4/10/2010
Location	Markov analyse backwards
Calls	beat-spacing-states, markov-analyse<-, spacing-holding-states
Called by	
Comments/see also	construct-final-states, construct-initial-states, construct-stm

Example:

```
(progn
  (setq
   variable-1
   '((0 60 60 2 0) (0 63 62 1 0) (1 67 64 2 0)
     (2 59 59 1 0)))
  (setq
   variable-2
   '((0 64 62 1 0) (1 62 61 1 0) (1 69 65 2 0)
     (2 66 63 1 0)))
  (setq *variable-names* '(variable-1 variable-2))
  (setq *catalogue* '("variable-1" "variable-2"))
  (construct-stm<-
   *variable-names* *catalogue* "beat-spacing-states"
   3 3 1))
--> "Finished"
*transition-matrix*
--> (((3 (3))
      (((2 (7))
        (4 2 "variable-2"
         ((1 62 61 1 0 2 1) (1 69 65 2 0 3 2))))))
     ((2 (7))
      (((1 NIL)
        (-2 -1 "variable-2"
         ((0 64 62 1 0 1 0))))
       ((1 (3))
        (0 0 "variable-1"
```

```
          ((0 60 60 2 0 2 0) (0 63 62 1 0 1 1))))))
      ((3 (8))
       (((2 (7))
         (-1 -1 "variable-1"
          ((0 60 60 2 0 2 0) (1 67 64 2 0 3 2)))))))
```

This function is similar to the function construct-stm, the difference being X_n is looked up and X_{n-1} is accumulated or added to a state- transition matrix. This recursion analyses one variable name at a time, taking a catalogue name from the variable catalogue, and updates the transition matrix accordingly. The output "Finished" is preferable to the transition matrix, which is large enough that it can cause the Listener to crash.

markov-analyse<-

Started, last checked	4/10/2010, 4/10/2010
Location	Markov analyse backwards
Calls	update-stm<-
Called by	construct-stm<-
Comments/see also	markov-analyse

Example:

```
(markov-analyse<-
 '(((3) ("Piece A")) ((6) ("Piece A"))
   ((4) ("Piece A")) ((4) ("Piece A"))
   ((3) ("Piece A")) ((2) ("Piece A"))))
--> "Finished"
*transition-matrix*
--> '(((2) (((3) ("Piece A"))))
      ((3) (((4) ("Piece A"))))
      ((4) (((4) ("Piece A")) ((6) ("Piece A"))))
      ((6) (((3) ("Piece A")))))
```

This function is similar to the function markov-analyse, the difference being X_n is looked up and X_{n-1} is accumulated or added to a state- transition matrix. This function has one argument - some states which are to be analysed according to a backwards first-order Markov model. Note the need to define a variable here, *transition-matrix*. The output "Finished" is preferable to the transition matrix, which is large enough that it can cause the Listener to crash.

present-to-stm<-

Started, last checked	4/10/2010, 4/10/2010
Location	Markov analyse backwards
Calls	add-to-stm<-, accumulate-to-stm<-
Called by	update-stm<-
Comments/see also	present-to-stm

Example:

```
(present-to-stm<-
 '(((4) ("Piece A")) ((2) ("Piece A")))
 '(((1) (((5) ("Piece A")) ((4) ("Piece B"))))
   ((2) (((5) ("Piece A"))))
   ((4) (((1) ("Piece B")) ((2) ("Piece C"))))
   ((5) (((4) ("Piece B"))))))
--> '(((2) (((4) ("Piece A")) ((5) ("Piece A"))))
      ((1) (((5) ("Piece A")) ((4) ("Piece B"))))
      ((4) (((1) ("Piece B")) ((2) ("Piece C"))))
      ((5) (((4) ("Piece B")))))
```

This function is similar to the function present-to-stm, the difference being X_n is looked up and X_{n-1} is accumulated or added to a state- transition matrix. The function calls either the function accumulate-to-stm¡-, or add-to-stm¡-, depending on whether the first argument, a listed- state, has appeared in the second argument, a state- transition matrix, before. The example above results in accumulate-to-stm¡- being called, as the state of (2) has occurred before. However, changing this state to (3) in the argument would result in add-to-stm¡- being called.

update-stm<-

Started, last checked	4/10/2010, 4/10/2010
Location	Markov analyse backwards
Calls	present-to-stm<-
Called by	markov-analyse<-
Comments/see also	update-stm

Example:

```
(update-stm<-
```

```
 '((((3) ("Piece A")) ((6) ("Piece A")))
   (((6) ("Piece A")) ((4) ("Piece A")))
   (((4) ("Piece A")) ((4) ("Piece A")))
   (((4) ("Piece A")) ((3) ("Piece A")))
   (((3) ("Piece A")) ((2) ("Piece A"))))
 nil)
--> '(((2) (((3) ("Piece A"))))
      ((3) (((4) ("Piece A"))))
      ((4) (((4) ("Piece A")) ((6) ("Piece A"))))
      ((6) (((3) ("Piece A")))))
```

This function is similar to the function update-stm, the difference being X_n is looked up and X_{n-1} is accumulated or added to a state- transition matrix. This function has as its argument listed states, and it applies the function present-to-stm¡- recursively to these listed states. The variable *transition-matrix* is updated as it proceeds.

4.6.4 Markov compose

These functions realise a sequence of states in some given transition matrix. The states are converted to datapoints so that they can be written to a MIDI file. A pivotal function is states2datapoints.

create-MIDI-note-numbers

Started, last checked	27/1/2009, 10/2/2009
Location	Markov compose
Calls	spacing2note-numbers
Called by	states2datapoints
Comments/see also	create-MIDI&morphetic-numbers

Example:

```
(create-MIDI-note-numbers
 '((((12 3 4) (1 0 1 1))
    (NIL 500 "b707b"
     ((3000 57 1000 4 96 4000 0)
      (3000 69 500 3 96 3500 1)
      (3000 72 1000 2 96 4000 2)
      (3000 76 1000 1 96 4000 3))))
   (((10 5 4) (2 0 3 2))
```

```
    (0 500 "b41500b"
     ((43000 52 1000 4 96 44000 54)
      (43500 62 500 3 96 44000 134)
      (43000 67 1500 2 96 44500 201)
      (43000 71 1000 1 96 44000 256)))))
--> ((((48 60 63 67) (1 0 1 1))
      (NIL 500 "b707b"
           ((3000 57 1000 4 96 4000 0)
            (3000 69 500 3 96 3500 1)
            (3000 72 1000 2 96 4000 2)
            (3000 76 1000 1 96 4000 3))))
     (((48 58 63 67) (2 0 3 2))
      (0 500 "b41500b"
         ((43000 52 1000 4 96 44000 54)
          (43500 62 500 3 96 44000 134)
          (43000 67 1500 2 96 44500 201)
          (43000 71 1000 1 96 44000 256)))))
```

A list of realised states is provided, and this function returns so-called 'half-states': MIDI note numbers have been created from the chord spacings. To do this, we take the MIDI note number of the previous bass note, and the variable bass-step, which is the interval in semitones between the bass note of the current state and previous state as they appeared in the original data. If the current state is an initial state in some original data, then this will be empty, and so it is set to zero.

half-state2datapoints

Started, last checked	27/1/2009, 10/2/2009
Location	Markov compose
Calls	state-note2datapoint
Called by	states2datapoints
Comments/see also	half-state2datapoints-by-lookup

Example:

```
(half-state2datapoints 0
 '((((48 60 63 67) (1 0 1 1))
    (NIL 500 "b707b"
     ((3000 57 1000 4 96 4000 0)
      (3000 69 500 3 96 3500 1)
```

```
      (3000 72 1000 2 96 4000 2)
      (3000 76 1000 1 96 4000 3))))
   (((48 58 63 67) (2 0 3 2))
    (0 500 "b41500b"
     ((43000 52 1000 4 96 44000 54)
      (43500 62 500 3 96 44000 134)
      (43000 67 1500 2 96 44500 201)
      (43000 71 1000 1 96 44000 256))))
   (((41 56 63 68) (1 1 2 1))
    (-7 500 "b37800n"
     ((62000 50 1000 4 96 63000 62)
      (62000 65 1000 3 96 63000 127)
      (61000 72 1500 2 96 62500 189)
      (62000 77 1000 1 96 63000 244)))))
 '(500 500 500 500 500) '(0 500 1000 1500 2000 2500))
--> ((0 48 1000 4 96) (0 60 500 3 96)
     (0 63 1500 2 96) (0 67 1000 1 96))
```

This function increments over the ith note of a half-state, $i = 0, 1, \ldots, n-1$. If the ith note in this state is of tie-type 0 or 1 (corresponding to 'untied' or 'tied-forward') then the function state-note2datapoint is applied.

index-of-offtime

Started, last checked	27/1/2009, 10/2/2009
Location	Markov compose
Calls	index-item-1st-occurs
Called by	state-note2datapoint
Comments/see also	index-of-offtime-by-lookup

Example:

```
(index-of-offtime 0 63
  '((((48 60 63 67) (1 0 1 1))
    (NIL 500 "b707b"
     ((3000 57 1000 4 96 4000 0)
      (3000 69 500 3 96 3500 1)
      (3000 72 1000 2 96 4000 2)
      (3000 76 1000 1 96 4000 3))))
   (((48 58 63 67) (2 0 3 2))
    (0 500 "b41500b"
```

```
     ((43000 52 1000 4 96 44000 54)
      (43500 62 500 3 96 44000 134)
      (43000 67 1500 2 96 44500 201)
      (43000 71 1000 1 96 44000 256))))
   (((41 56 63 68) (1 1 2 1))
    (-7 500 "b37800n"
     ((62000 50 1000 4 96 63000 62)
      (62000 65 1000 3 96 63000 127)
      (61000 72 1500 2 96 62500 189)
      (62000 77 1000 1 96 63000 244))))))
--> 2
```

Given a starting index, a note-number and some half-states to search through, this function returns the index of the half-state where the note-number in question comes to an end. This will be where its tie type is first equal to 0 or 2, indicating 'untied' and 'tied-back' respectively.

realise-states

Started, last checked	27/1/2009, 10/2/2009
Location	Markov compose
Calls	choose-one
Called by	
Comments/see also	

Example:

```
(realise-states *initial-states* *stm* 2)
--> ((((4 3 5) (0 0 1 1))
      (NIL 500 "b43800b"
           ((3000 48 500 4 96 3500 0)
            (3000 52 500 3 96 3500 1)
            (3000 55 1000 2 96 4000 2)
            (3000 60 1000 1 96 4000 3))))
     (((4 5 5) (0 0 2 2))
      (-2 500 "b42600b"
          ((46500 60 500 4 96 47000 56)
           (46500 64 500 3 96 47000 126)
           (46000 69 1000 2 96 47000 191)
           (46000 74 1000 1 96 47000 253)))))
```

Given some initial states and a transition matrix, and an optional argument called count, this function realises a total of count states in the transition matrix. If a closed state is reached, then the process is terminated, and however many states have been generated by this stage are returned.

scale-datapoints-by-factor

Started, last checked	27/1/2009, 10/2/2009
Location	Markov compose
Calls	choose-one
Called by	
Comments/see also	

Example:

```
(scale-datapoints-by-factor 2
 '((0 48 1000 4 96) (0 60 500 3 96) (0 63 1500 2 96)
   (0 67 1000 1 96) (500 58 500 3 96)
   (1000 41 1000 4 96) (1000 56 1000 3 96)))
--> ((0 48 2000 4 96) (0 60 1000 3 96)
     (0 63 3000 2 96) (0 67 2000 1 96)
     (1000 58 1000 3 96) (2000 41 2000 4 96)
     (2000 56 2000 3 96))
```

The ontimes and durations of datapoints are scaled up or down by a constant factor.

spacing2note-numbers

Started, last checked	27/1/2009, 10/2/2009
Location	Markov compose
Calls	add-to-list, fibonacci-list
Called by	create-MIDI-note-numbers
Comments/see also	

Example:

```
(spacing2note-numbers '(3 12) 64)
--> (64 67 79)
```

A chord spacing and note-number are inputs to this function. Returned are the MIDI note numbers of the chord whose lowest note is given by note-number, and whose spacing is as provided.

state-durations

Started, last checked	27/1/2009, 10/2/2009
Location	Markov compose
Calls	
Called by	states2datapoints
Comments/see also	state-durations-by-beat

Example:

```
(state-durations
 '((((10 5 4) (2 0 3 2))
    (0 500 "b41500b"
     ((43000 52 1000 4 96 44000 54)
      (43500 62 500 3 96 44000 134)
      (43000 67 1500 2 96 44500 201)
      (43000 71 1000 1 96 44000 256))))
   (((15 7 5) (1 1 2 1))
    (-7 500 "b37800n"
     ((62000 50 1000 4 96 63000 62)
      (62000 65 1000 3 96 63000 127)
      (61000 72 1500 2 96 62500 189)
      (62000 77 1000 1 96 63000 244))))
   (((15 6 6) (2 2 0 2))
    (0 500 "b42600b"
     ((39000 45 1000 4 96 40000 47)
      (39000 60 1000 3 96 40000 119)
      (39500 66 500 2 96 40000 183)
      (39000 72 1000 1 96 40000 247))))
   (((12 9 7) (1 0 0 1))
    (1 500 "b39100b"
     ((35000 53 2000 4 96 37000 31)
      (35000 65 500 3 96 35500 95)
      (35000 74 500 2 96 35500 162)
      (35000 81 1000 1 96 36000 225))))))
--> (500 500 500 500)
```

This function takes a list of states as its argument, and returns a list containing the duration of each state in a list. The set of so-called ‘partition points’ can be generated easily by applying the function fibonacci-list.

state-note2datapoint

Started, last checked	27/1/2009, 10/2/2009
Location	Markov compose
Calls	index-item-1st-occurs, index-of-offtime, min-item
Called by	half-state2datapoints
Comments/see also	state-note2datapoint-by-lookup

Example:

```
(state-note2datapoint 2 0
 '((((48 60 63 67) (1 0 1 1))
    (NIL 500 "b707b"
     ((3000 57 1000 4 96 4000 0)
      (3000 69 500 3 96 3500 1)
      (3000 72 1000 2 96 4000 2)
      (3000 76 1000 1 96 4000 3))))
   (((48 58 63 67) (2 0 3 2))
    (0 500 "b41500b"
     ((43000 52 1000 4 96 44000 54)
      (43500 62 500 3 96 44000 134)
      (43000 67 1500 2 96 44500 201)
      (43000 71 1000 1 96 44000 256))))
   (((41 56 63 68) (1 1 2 1))
    (-7 500 "b37800n"
     ((62000 50 1000 4 96 63000 62)
      (62000 65 1000 3 96 63000 127)
      (61000 72 1500 2 96 62500 189)
      (62000 77 1000 1 96 63000 244)))))
 '(500 500 500) '(0 500 1000 1500))
--> (0 63 1500 2 96)
```

The ith note of the jth half-state is transformed into a so-called 'datapoint', meaning we find its ontime (the jth element of the partition points), its MIDI note number, its offtime, which is trickier. Other information, such as voicing and relative dynamics, are drawn from the initial occurence of the half-state in question.

Returning to the calculation of offtime, we find the half-state k where the note ends and use this to calculate the duration of the note up until the kth state. Whichever is less—the kth state duration or the duration of this note within the kth state—is added to give the total duration. This encapsulates the implicit encoding of rests.

states2datapoints

Started, last checked	27/1/2009, 10/2/2009
Location	Markov compose
Calls	create-MIDI-note-numbers, fibonacci-list, half-state2datapoints, state-durations
Called by	
Comments/see also	states2datapoints-by-lookup, states2datapoints-by-lookup<-

Example:

```
(states2datapoints
 '(((((12 3 4) (1 0 1 1))
    (NIL 500 "b707b"
     ((3000 57 1000 4 96 4000 0)
      (3000 69 500 3 96 3500 1)
      (3000 72 1000 2 96 4000 2)
      (3000 76 1000 1 96 4000 3))))
   (((10 5 4) (2 0 3 2))
    (0 500 "b41500b"
     ((43000 52 1000 4 96 44000 54)
      (43500 62 500 3 96 44000 134)
      (43000 67 1500 2 96 44500 201)
      (43000 71 1000 1 96 44000 256))))) 48)
--> ((0 48 1000 4 96) (0 60 500 3 96) (0 63 1000 2 96)
     (0 67 1000 1 96) (500 58 500 3 96))
```

This function applies the function half-state2datapoint recursively to a list of states. Some initial caluclations are performed to obtain half-states, state durations and partition points.

4.6.5 Beat rel MNN states

The aim of these functions is to convert a dataset representing a melody or polyphonic piece to beat-MNN states, where MNN (MIDI note number) is relative to the tonic note closest to the mean MNN of the melody. The key is either required as an argument, or estimated using the Krumhansl-Schmuckler key-finding algorithm (Krumhansl, 1990). The function 4.6.5 is the most robust here. The functions 4.6.5 and 4.6.5 contribute towards Collins and Coulon (2012).

beat-MNN-states

Started, last checked	2/1/2013, 2/1/2013
Location	Beat rel MNN states
Calls	centre-dataset, **??**, segments-strict
Called by	
Comments/see also	Possibly obsolete.

Example:

```
(beat-MNN-states 4 '(-3 0)
 '((0 63 62 3/4 0) (3/4 63 62 1/4 0) (1 65 63 1/2 0)
   (2 66 64 1 0) (3 65 63 3/4 0) (15/4 63 62 1/4 0)
   (4 66 64 1 0) (5 65 63 3/4 0) (23/4 63 62 1/4 0)
   (6 66 64 1 0) (7 65 63 1 0) (35/4 63 62 1/4 0)
   (9 65 63 3/4 0) (39/4 63 62 1/4 0)
   (10 66 64 3/4 0) (43/4 65 63 1/4 0)
   (11 63 62 1 0)) "gersh06")
--> (((1 0)
      ("gersh06" (0 63 62 3/4 0) (63 62) (-3 0)))
     ((7/4 0)
      ("gersh06" (3/4 63 62 1/4 0) (63 62) (-3 0)))
     ((2 2)
      ("gersh06" (1 65 63 1/2 0) (63 62) (-3 0)))
     ((5/2 NIL)
      ("gersh06" NIL (63 62) (-3 0)))
     ((3 3)
      ("gersh06" (2 66 64 1 0) (63 62) (-3 0)))
     ((4 2)
      ("gersh06" (3 65 63 3/4 0) (63 62) (-3 0)))
     ((19/4 0)
      ("gersh06" (15/4 63 62 1/4 0) (63 62) (-3 0)))
     ((1 3)
      ("gersh06" (4 66 64 1 0) (63 62) (-3 0)))
     ((2 2)
      ("gersh06" (5 65 63 3/4 0) (63 62) (-3 0)))
```

```
((11/4 0)
 ("gersh06" (23/4 63 62 1/4 0) (63 62) (-3 0)))
((3 3)
 ("gersh06" (6 66 64 1 0) (63 62) (-3 0)))
((4 2)
 ("gersh06" (7 65 63 1 0) (63 62) (-3 0)))
((1 NIL)
 ("gersh06" NIL (63 62) (-3 0)))
((7/4 0)
 ("gersh06" (35/4 63 62 1/4 0) (63 62) (-3 0)))
((2 2)
 ("gersh06" (9 65 63 3/4 0) (63 62) (-3 0)))
((11/4 0)
 ("gersh06" (39/4 63 62 1/4 0) (63 62) (-3 0)))
((3 3)
 ("gersh06" (10 66 64 3/4 0) (63 62) (-3 0)))
((15/4 2)
 ("gersh06" (43/4 65 63 1/4 0) (63 62) (-3 0)))
((4 0)
 ("gersh06" (11 63 62 1 0) (63 62) (-3 0))))
```

The function contributes towards Collins and Coulon (2012). It converts the dataset into beat-MNN states. The dataset is assumed to represent a melody. MNN is relative to the tonic note closest to the mean MNN, which can be worked out from the second argument.

beat-MNNs-states

Started, last checked	2/1/2013, 2/1/2013
Location	Beat rel MNN states
Calls	centre-dataset, **??** , nth-list-of-lists, segments-strict
Called by	
Comments/see also	Possibly obsolete.

Example:

```
\noindent Example:
\begin{verbatim}
(beat-MNNs-states 4 '(1 0)
 '((-1 55 57 1 3) (-1 59 59 1 2) (-1 62 61 1 1)
```

```
   (-1 67 64 1 0) (0 54 56 1 3) (0 57 58 1 2)
   (0 62 61 1/2 1) (0 74 68 1 0) (1/2 64 62 1/2 1)
   (1 50 54 1 3) (1 62 61 1/2 2) (1 66 63 1 1)
   (1 74 68 1 0) (3/2 60 60 1/2 2) (2 55 57 1/2 3)
   (2 59 59 1/2 2) (2 67 64 1 1) (2 74 68 1 0)
   (5/2 57 58 1/2 3) (5/2 60 60 1/2 2) (3 59 59 1 3)
   (3 62 61 1 2) (3 67 64 1 1) (3 74 68 1 0))
 "chorale-bwv-151-ed")
--> (((4 (-12 -8 -5 0))
      ("chorale-bwv-151-ed"
       ((-1 55 57 1 3) (-1 59 59 1 2) (-1 62 61 1 1)
        (-1 67 64 1 0)) (67 64) (1 0)))
     ((1 (-13 -10 -5 7))
      ("chorale-bwv-151-ed"
       ((0 54 56 1 3) (0 57 58 1 2) (0 62 61 1/2 1)
        (0 74 68 1 0)) (67 64) (1 0)))
     ((3/2 (-13 -10 -3 7))
      ("chorale-bwv-151-ed"
       ((0 54 56 1 3) (0 57 58 1 2)
        (1/2 64 62 1/2 1) (0 74 68 1 0)) (67 64)
       (1 0)))
     ((2 (-17 -5 -1 7))
      ("chorale-bwv-151-ed"
       ((1 50 54 1 3) (1 62 61 1/2 2) (1 66 63 1 1)
        (1 74 68 1 0)) (67 64) (1 0)))
     ((5/2 (-17 -7 -1 7))
      ("chorale-bwv-151-ed"
       ((1 50 54 1 3) (3/2 60 60 1/2 2)
        (1 66 63 1 1) (1 74 68 1 0)) (67 64) (1 0)))
     ((3 (-12 -8 0 7))
      ("chorale-bwv-151-ed"
       ((2 55 57 1/2 3) (2 59 59 1/2 2)
        (2 67 64 1 1) (2 74 68 1 0)) (67 64) (1 0)))
     ((7/2 (-10 -7 0 7))
      ("chorale-bwv-151-ed"
       ((5/2 57 58 1/2 3) (5/2 60 60 1/2 2)
        (2 67 64 1 1) (2 74 68 1 0)) (67 64) (1 0)))
     ((4 (-8 -5 0 7))
      ("chorale-bwv-151-ed"
       ((3 59 59 1 3) (3 62 61 1 2) (3 67 64 1 1)
        (3 74 68 1 0)) (67 64) (1 0)))).
```

The function contributes towards Collins and Coulon (2012). It converts the dataset into beat-MNNs states. The dataset can represent polyphonic or melodic material. MNN is relative to the tonic note closest to the mean MNN, which can be worked out from the second argument.

beat-rel-MNN-states

Started, last checked	2/1/2013, 14/1/2015
Location	Beat rel MNN states
Calls	centre-dataset, **??** , nth-list-of-lists, segments-strict
Called by	
Comments/see also	

Example:

```
(setq
 dataset
 '((-1 72 67 7/4 0) (0 55 57 1 1) (0 61 61 1 1)
   (0 64 63 1 1) (3/4 70 66 1/4 0) (1 56 58 1 1)
   (1 60 60 2 1) (1 63 62 2 1) (1 68 65 1/2 0)
   (3/2 70 66 1/2 0) (2 51 55 1 1) (2 72 67 7/4 0)
   (3 55 57 1 1) (3 61 61 1 1) (3 64 63 1 1)
   (15/4 70 66 1/4 0) (4 56 58 1 1) (4 60 60 2 1)
   (4 63 62 2 1) (4 68 65 1/2 0) (9/2 77 70 1/2 0)
   (5 51 55 1 1) (5 75 69 1 0) (6 55 57 1 1)
   (6 61 61 1 1) (6 64 63 1 1) (6 72 67 3/4 0)
   (27/4 70 66 1/4 0) (7 56 58 1 1) (7 60 60 2 1)
   (7 63 62 2 1) (7 68 65 1/2 0) (15/2 70 66 1/2 0)
   (8 51 55 1 1) (8 72 67 1 0) (9 56 58 3/4 1)
   (9 61 61 3 1) (9 63 62 3 0) (39/4 55 57 1/4 1)
   (10 53 56 1/2 1) (21/2 55 57 1/2 1) (11 51 55 1 1)
   (12 55 57 1 1) (12 61 61 1 1) (12 64 63 1 1)
   (12 72 67 3/4 0)))
(beat-rel-MNN-states
 dataset "C-17-4-mini" 3 1 2 3)
--> (((3 (4))
      ("C-17-4-mini" ((-1 72 67 7/4 0)) (68 65)
       (-4 0)))
     ((1 (-13 -7 -4 4))
      ("C-17-4-mini"
```

```
  ((0 55 57 1 1) (0 61 61 1 1) (0 64 63 1 1)
   (-1 72 67 7/4 0)) (68 65) (-4 0)))
((7/4 (-13 -7 -4 2))
 ("C-17-4-mini"
  ((0 55 57 1 1) (0 61 61 1 1) (0 64 63 1 1)
   (3/4 70 66 1/4 0)) (68 65) (-4 0)))
((2 (-12 -8 -5 0))
 ("C-17-4-mini"
  ((1 56 58 1 1) (1 60 60 2 1) (1 63 62 2 1)
   (1 68 65 1/2 0)) (68 65) (-4 0)))
((5/2 (-12 -8 -5 2))
 ("C-17-4-mini"
  ((1 56 58 1 1) (1 60 60 2 1) (1 63 62 2 1)
   (3/2 70 66 1/2 0)) (68 65) (-4 0)))
((3 (-17 -8 -5 4))
 ("C-17-4-mini"
  ((2 51 55 1 1) (1 60 60 2 1) (1 63 62 2 1)
   (2 72 67 7/4 0)) (68 65) (-4 0)))
((1 (-13 -7 -4 4))
 ("C-17-4-mini"
  ((3 55 57 1 1) (3 61 61 1 1) (3 64 63 1 1)
   (2 72 67 7/4 0)) (68 65) (-4 0)))
((7/4 (-13 -7 -4 2))
 ("C-17-4-mini"
  ((3 55 57 1 1) (3 61 61 1 1) (3 64 63 1 1)
   (15/4 70 66 1/4 0)) (68 65) (-4 0)))
((2 (-12 -8 -5 0))
 ("C-17-4-mini"
  ((4 56 58 1 1) (4 60 60 2 1) (4 63 62 2 1)
   (4 68 65 1/2 0)) (68 65) (-4 0)))
((5/2 (-12 -8 -5 9))
 ("C-17-4-mini"
  ((4 56 58 1 1) (4 60 60 2 1) (4 63 62 2 1)
   (9/2 77 70 1/2 0)) (68 65) (-4 0)))
((3 (-17 -8 -5 7))
 ("C-17-4-mini"
  ((5 51 55 1 1) (4 60 60 2 1) (4 63 62 2 1)
   (5 75 69 1 0)) (68 65) (-4 0)))
((1 (-13 -7 -4 4))
 ("C-17-4-mini"
  ((6 55 57 1 1) (6 61 61 1 1) (6 64 63 1 1)
```

```
    (6 72 67 3/4 0)) (68 65) (-4 0)))
((7/4 (-13 -7 -4 2))
 ("C-17-4-mini"
  ((6 55 57 1 1) (6 61 61 1 1) (6 64 63 1 1)
   (27/4 70 66 1/4 0)) (68 65) (-4 0)))
((2 (-12 -8 -5 0))
 ("C-17-4-mini"
  ((7 56 58 1 1) (7 60 60 2 1) (7 63 62 2 1)
   (7 68 65 1/2 0)) (68 65) (-4 0)))
((5/2 (-12 -8 -5 2))
 ("C-17-4-mini"
  ((7 56 58 1 1) (7 60 60 2 1) (7 63 62 2 1)
   (15/2 70 66 1/2 0)) (68 65) (-4 0)))
((3 (-17 -8 -5 4))
 ("C-17-4-mini"
  ((8 51 55 1 1) (7 60 60 2 1) (7 63 62 2 1)
   (8 72 67 1 0)) (68 65) (-4 0)))
((1 (-12 -7 -5))
 ("C-17-4-mini"
  ((9 56 58 3/4 1) (9 61 61 3 1) (9 63 62 3 0))
  (68 65) (-4 0)))
((7/4 (-13 -7 -5))
 ("C-17-4-mini"
  ((39/4 55 57 1/4 1) (9 61 61 3 1)
   (9 63 62 3 0)) (68 65) (-4 0)))
((2 (-15 -7 -5))
 ("C-17-4-mini"
  ((10 53 56 1/2 1) (9 61 61 3 1)
   (9 63 62 3 0)) (68 65) (-4 0)))
((5/2 (-13 -7 -5))
 ("C-17-4-mini"
  ((21/2 55 57 1/2 1) (9 61 61 3 1)
   (9 63 62 3 0)) (68 65) (-4 0)))
((3 (-17 -7 -5))
 ("C-17-4-mini"
  ((11 51 55 1 1) (9 61 61 3 1) (9 63 62 3 0))
  (68 65) (-4 0)))
((1 (-13 -7 -4 4))
 ("C-17-4-mini"
  ((12 55 57 1 1) (12 61 61 1 1) (12 64 63 1 1)
   (12 72 67 3/4 0)) (68 65) (-4 0)))
```

```
((7/4 (-13 -7 -4))
 ("C-17-4-mini"
  ((12 55 57 1 1) (12 61 61 1 1)
   (12 64 63 1 1)) (68 65) (-4 0))))
```

Suppose you have three states X_{n-1}, X_n, X_{n+1}. The function beat-rel-MNN-states looks at the beat and MIDI note numbers of X_n, the latter being centred relative to an estimated tonic.

14/1/2015. It was noticed that the first attempt at this function did not sort the relative MIDI note numbers ascending, nor did it remove duplicates. This is likely to exacerbate problems with dead ends, so the function was altered to do both sorting and removing of duplicates, and comparative generation tests were performed.

centre-dataset

Started, last checked	2/1/2013, 2/1/2013
Location	Beat rel MNN states
Calls	add-to-list, fifth-steps-mode2MNN-MPN, min-argmin
Called by	beat-MNN-states, beat-MNNs-states, beat-rel-MNN-states, HarmAn->roman
Comments/see also	

Example:

```
(centre-dataset '(-3 0)
 '((0 63 62 3/4 0) (3/4 63 62 1/4 0) (1 65 63 3/4 0)
   (7/4 63 62 1/4 0) (2 66 64 1 0) (3 65 63 3/4 0)
   (15/4 63 62 1/4 0) (4 66 64 1 0) (5 65 63 3/4 0)
   (23/4 63 62 1/4 0) (6 66 64 1 0) (7 65 63 1 0)
   (35/4 63 62 1/4 0) (9 65 63 3/4 0)
   (39/4 63 62 1/4 0) (10 66 64 3/4 0)
   (43/4 65 63 1/4 0) (11 63 62 1 0)))
--> ((63 62)
     ((0 0 0 3/4 0) (3/4 0 0 1/4 0) (1 2 1 3/4 0)
      (7/4 0 0 1/4 0) (2 3 2 1 0) (3 2 1 3/4 0)
      (15/4 0 0 1/4 0) (4 3 2 1 0) (5 2 1 3/4 0)
      (23/4 0 0 1/4 0) (6 3 2 1 0) (7 2 1 1 0)
      (35/4 0 0 1/4 0) (9 2 1 3/4 0) (39/4 0 0 1/4 0)
      (10 3 2 3/4 0) (43/4 2 1 1/4 0) (11 0 0 1 0)))
```

Translates the dataset so that the tonic note closest to the mean MNN is represented by the pair (0 0).

fifth-steps-mode2MNN-MPN

Started, last checked	2/1/2013, 2/1/2013
Location	Beat rel MNN states
Calls	
Called by	centre-dataset
Comments/see also	

Example:

```
(fifth-steps-mode2MNN-MPN '(-5 0))
--> (61 61)
```

A pair consisting of position on the cycle of fifths and mode (0 for Ionian, 1 for Dorian, etc.) is converted to a pair consisting of a MIDI note number and morphetic pitch number for the tonic. This was called by an older version of 4.6.5 but now it may be obsolete.

4.6.6 Generating beat relative MNN

The main function here is called generate-beat-rel-MNN->. It is embedded in Racchman-Jun2014, and is similar to generate-beat-MNN-spacing-> from Racchman-Oct2010. The difference is that the MIDI note numbers in generate-beat-rel-MNN-> (and Racchman-Jun2014) are assumed to be relative to a global tonic. Given initial states, a state transition matrix, an upper limit for ontime, a template point set, and the tonic pitch closest (tpc) to its mean MIDI note number, this function generates points (notes, among other output) that conform to various criteria, which can be specified using the optional arguments. The main criterion that has been tested for generate-beat-rel-MNN-> so far is not too many consecutive states coming from the same source. It is not clear whether having to control for range or expectancy (as in Racchman-Oct2010) is necessary here. Some unusual pauses have been noted in the output already, however, so this will need checking.

checklistp-rel

Started, last checked	1/6/2014, 17/8/2014
Location	Generating beat relative MNN
Calls	index-item-1st-doesnt-occur, lastn, my-last
Called by	generate-beat-rel-MNN->, ??
Comments/see also	??

Example:

```
(setq
 states
 '((((1 (-15 4))
    ("Complicated"
     ((0 50 54 3/2 1) (0 69 65 1 0)) (65 63) (-1 0)))
   ((5/4 (-15 2))
    ("Complicated"
     ((72 50 54 1 1) (289/4 67 64 1/4 0))
     (65 63) (-1 0)))
   ((3/2 (-8 0))
    ("Am_I_wrong"
     ((449/2 55 57 1/2 1) (449/2 63 62 1/2 0))
     (63 62) (-3 0)))
   ((2 (-3))
    ("Am_I_wrong" ((185 60 60 1/2 1))
     (63 62) (-3 0)))))
(setq point-set nil)
(setq template-segments nil)
(setq checklist '("originalp"))
(setq c-sources 3)
(checklistp-rel
 states point-set template-segments checklist
 c-sources)
--> T.
```

A simpler version of the function checklistp that just checks whether the sources are all the same. Note the location of the sources within the context have changed for Racchman-Jun2014 compared to Racchman-Oct2010, from third to first.

generate-beat-rel-MNN->

Started, last checked	1/6/2014, 17/8/2014
Location	Generating beat relative MNN
Calls	checklistp-rel, choose-one, choose-one-with-beat, index-1st-sublist-item>=, my-last, segments-strict, sort-dataset-asc, states2datapoints-by-rel, translate-datapoints-to-first-ontime
Called by	
Comments/see also	generate-beat-MNN-spacing->

Example:

```
(progn
  (setq
   temp-path
   (merge-pathnames
    (make-pathname
     :directory
     '(:relative "Racchman-Jun2014 example"))
  *MCStylistic-MonthYear-example-files-results-path*))
  (setq
   initial-states
   (read-from-file
    (merge-pathnames
     (make-pathname
      :directory
      '(:relative "Racchman-Jun2014 example")
      :name "initial-states" :type "txt")
 *MCStylistic-MonthYear-example-files-results-path*)))
  (setq
   stm
   (read-from-file
    (merge-pathnames
     (make-pathname
      :directory
      '(:relative "Racchman-Jun2014 example")
      :name "transition-matrix" :type "txt")
 *MCStylistic-MonthYear-example-files-results-path*)))
```

```
  (setq no-ontimes> 12)
  (setq
   dataset-all
   (read-from-file
    (merge-pathnames
     (make-pathname
      :directory
      '(:relative "softEmotivePop" "lisp")
      :name "Young_and_beautiful" :type "txt")
     *MCStylistic-MonthYear-data-path*)))
  (setq
   template-fsm (fifth-steps-mode dataset-all))
  (setq
   trans-pair&c-dataset
   (centre-dataset template-fsm dataset-all))
  (setq template-tpc (first trans-pair&c-dataset))
  (setq beats-in-bar 4)
  (setq checklist (list "originalp"))
  (setq scale 1000)
  "Yes!")

(setq
 *rs*
 #.(CCL::INITIALIZE-MRG31K3P-STATE 119640237
    1896132409 1283053466 2078949444 1948704030
    110577318))
(setq time-a (get-internal-real-time))
(setq
 output
 (generate-beat-rel-MNN->
  initial-states stm no-ontimes> dataset-all
  template-tpc checklist beats-in-bar))
(setq time-b (get-internal-real-time))
(setq
 time-taken
 (float
  (/
   (- time-b time-a)
   internal-time-units-per-second)))
--> 81.822464
(write-to-file
```

```
 (cons time-taken output)
 (merge-pathnames
  (make-pathname
   :name "Racchman-Jun2014-sample-output" :type "txt")
  temp-path))
(saveit
 (merge-pathnames
  (make-pathname
   :name "Racchman-Jun2014-sample-output" :type "mid")
  temp-path)
 (modify-to-check-dataset (second output) scale))
(firstn 11 (second output))
--> ((0 50 54 3/2 1) (0 69 65 3/2 0) (0 74 68 3/2 0)
     (0 78 70 3/2 0) (3/2 50 54 1/2 1) (3/2 69 65 2 0)
     (3/2 74 68 2 0) (3/2 78 70 2 0) (2 50 54 3/2 1)
     (7/2 50 54 1/2 1) (7/2 69 65 1/2 0))
```

This function is embedded in Racchman-Jun2014, and is similar to generate-beat-MNN-spacing-> from Racchman-Oct2010. The difference is that the MIDI note numbers in generate-beat-rel-MNN-> (and Racchman-Jun2014) are assumed to be relative to a global tonic. Given initial states, a state-transition matrix, an upper limit for ontime, a template point set, and the tonic pitch closest (tpc) to its mean MIDI note number, this function generates points (notes, among output) that conform to various criteria, which can be specified using the optional arguments. At present, the criteria are things like: number of failures tolerated at any point before process terminated (c-failures); not too many consecutive states from the same source (c-sources). A record, named index-failures, is kept of how many times a generated state fails to meet the criteria. When a threshold c-failures is exceeded at any state index, the penultimate state is removed and generation continues. If the value of c-failures is exceeded for the first state, a message is returned. This function returns the states as well as the points, and also index-failures and i, the total number of iterations.

states2datapoints-by-rel

Started, last checked	1/6/2014, 17/8/2014
Location	Generating beat relative MNN
Calls	add-to-list, fibonacci-list, half-state2datapoints-by-lookup, nth-list-of-lists, state-durations-by-beat
Called by	generate-beat-rel-MNN->
Comments/see also	states2datapoints-by-lookup

Example:

```
(setq
 states
 '(((1 (-12 -5))
    ("C-6-3" ((0 52 55 1 1) (0 59 59 1 1))
     (64 62) (4 0)))
   ((2 (0))
    ("C-7-2" ((4 62 61 3/2 0)) (62 61) (2 0)))
   ((7/2 (-1))
    ("C-6-4" ((125/2 59 59 1/2 0)) (60 60) (0 0)))
   ((1 (-12 -5 4))
    ("C-6-3"
     ((0 52 55 1 1) (0 59 59 1 1) (0 66 63 2 0))
     (64 62) (4 0)))
   ((2 (4))
    ("C-6-4" ((7 64 62 1 0)) (60 60) (0 0)))
   ((3 (2))
    ("C-6-4" ((8 64 62 1 0)) (60 60) (0 0)))
   ((1 "rest")
    ("C-6-3" nil (64 62) (4 0)))
   ((3/2 (2))
    ("C-7-2" ((13/2 64 62 1/2 0)) (62 61) (2 0)))
   ((2 (4))
    ("C-7-2" ((7 66 63 1/2 0)) (62 61) (2 0)))
   ((3 (5))
    ("C-7-2" ((8 67 64 1 0)) (62 61) (2 0)))
   ((1 (-8 -1 7))
    ("C-6-3"
     ((72 56 57 3 1) (72 63 61 3 1) (72 71 66 3 1))
     (64 62) (4 0)))))
```

```
(states2datapoints-by-rel states 3 60 60)
--> ((0 48 53 1 1) (0 55 57 1 1) (1 60 60 3/2 0)
     (5/2 59 59 1/2 0) (3 48 53 1 1) (3 55 57 1 1)
     (3 62 61 1 0) (4 64 62 1 0) (5 64 62 1 0)
     (13/2 62 61 1/2 0) (7 64 62 1 0) (8 65 63 1 0)
     (9 52 55 3 1) (9 59 59 3 1) (9 67 64 3 1))
```

This function applies the function half-state2datapoints-by-lookup recursively to a list of states. It is very similar to the function states2datapoints-by-lookup.

4.6.7 Spacing states

The functions here use segmentations to build different types of states. One, output by the function spacing-holding-states, consists of chord spacing and holding types. Another, output by the function beat-spacing-states, consists of beat-of-bar and chord spacing.

bass-steps

Started, last checked	24/8/2010, 24/8/2010
Location	Spacing states
Calls	
Called by	spacing-holding-states
Comments/see also	bass-steps-with-rests

Example:

```
(bass-steps
 '(((56 0 0) (60 1 1) (72 1 2))
   ((58 1 3) (60 2 1) (72 3 2))
   ((58 2 3) (65 1 4) (72 3 2))
   ((56 0 5) (65 2 4) (72 2 2))
   ((55 0 6) (64 0 7) (73 1 8)) ((NIL NIL NIL))
   ((54 2 9) (70 2 10) (74 0 11))
   ((59 0 12) (63 1 13) (75 1 14))))
--> '(2 0 -2 -1 NIL NIL 5)
```

This function takes a list of sorted holding types and returns the intervals between the bass notes of adjacent segments. It handles null entries, but these will have been removed if it is being called by the function spacing-holding-states.

bass-steps-with-rests

Started, last checked	24/8/2010, 24/8/2010
Location	Spacing states
Calls	
Called by	beat-spacing-states, beat-spacing-states<-
Comments/see also	bass-steps

Example:

```
(bass-steps-with-rests
 '((3 ((3 48 53 3 1 6 0) (3 67 64 3/4 0 15/4 1)
       (3 76 69 3/4 0 15/4 2)))
   (15/4 ((3 48 53 3 1 6 0) (15/4 65 63 1/4 0 4 3)
  (15/4 74 68 1/4 0 4 4)))
   (4 ((3 48 53 3 1 6 0) (4 64 62 2 0 6 5)
       (4 72 67 2 0 6 6)))
   (6 NIL)
   (13/2 ((13/2 61 60 1/2 0 7 7)))
   (7 ((7 62 61 1/2 0 15/2 8)))
   (15/2 ((15/2 64 62 1/2 0 8 9)))
   (8 ((8 50 54 1 1 9 10) (8 65 63 1 0 9 11)))
   (9 NIL)))
--> '(0 0 NIL 13 1 2 -14)
```

This function is similar to the function bass-steps. Rather than waiting for the next two consecutive non-nil states to calculate step sizes, it calculates step sizes across a rest state.

beat-spacing-states

Started, last checked	24/8/2010, 24/8/2010
Location	Spacing states
Calls	append-offtimes, bass-steps-with-rests, enumerate-append, nth-list-of-lists, segments-strict, spacing
Called by	
Comments/see also	beat-spacing-states<-, spacing-holding-states

Example:

```
(beat-spacing-states
 '((3 48 53 3 1) (3 67 64 3/4 0) (3 76 69 3/4 0)
   (15/4 65 63 1/4 0) (15/4 74 68 1/4 0) (4 64 62 2 0)
   (4 72 67 2 0) (13/2 61 60 1/2 0) (7 62 61 1/2 0)
   (15/2 64 62 1/2 0) (8 50 54 1 1) (8 65 63 1 0))
 "C-68-3-mini" 3 1 3)
--> (((1 (19 9))
      (NIL NIL "C-68-3-mini"
           ((3 48 53 3 1 6 0) (3 67 64 3/4 0 15/4 1)
            (3 76 69 3/4 0 15/4 2))))
     ((7/4 (17 9))
      (0 0 "C-68-3-mini"
         ((3 48 53 3 1 6 0) (15/4 65 63 1/4 0 4 3)
          (15/4 74 68 1/4 0 4 4))))
     ((2 (16 8))
      (0 0 "C-68-3-mini"
         ((3 48 53 3 1 6 0) (4 64 62 2 0 6 5)
          (4 72 67 2 0 6 6))))
     ((1 "rest")
      (NIL NIL "C-68-3-mini" NIL))
     ((3/2 NIL)
      (13 7 "C-68-3-mini"
          ((13/2 61 60 1/2 0 7 7))))
     ((2 NIL)
      (1 1 "C-68-3-mini"
         ((7 62 61 1/2 0 15/2 8))))
     ((5/2 NIL)
      (2 1 "C-68-3-mini"
         ((15/2 64 62 1/2 0 8 9))))
     ((3 (15))
      (-14 -8 "C-68-3-mini"
           ((8 50 54 1 1 9 10) (8 65 63 1 0 9 11)))))
```

Suppose you have three states X_{n-1}, X_n, X_{n+1}. The function beat-spacing-states looks at the beat and spacing of X_n, and also records the difference between the bass notes of X_n and X_{n-1}.

beat-spacing-states<-

Started, last checked	24/8/2010, 24/8/2010
Location	Spacing states
Calls	append-offtimes, bass-steps-with-rests, enumerate-append, nth-list-of-lists, segments-strict, spacing
Called by	
Comments/see also	beat-spacing-states, spacing-holding-states

Example:

```
(beat-spacing-states<-
 '((3 48 53 3 1) (3 67 64 3/4 0) (3 76 69 3/4 0)
   (15/4 65 63 1/4 0) (15/4 74 68 1/4 0) (4 64 62 2 0)
   (4 72 67 2 0) (13/2 61 60 1/2 0) (7 62 61 1/2 0)
   (15/2 64 62 1/2 0) (8 50 54 1 1) (8 65 63 1 0))
 "C-68-3-mini" 3 1 3)
--> (((1 (19 9))
      (NIL NIL "C-68-3-mini"
           ((3 48 53 3 1 6 0) (3 67 64 3/4 0 15/4 1)
            (3 76 69 3/4 0 15/4 2))))
     ((7/4 (17 9))
      (0 0 "C-68-3-mini"
         ((3 48 53 3 1 6 0) (15/4 65 63 1/4 0 4 3)
          (15/4 74 68 1/4 0 4 4))))
     ((2 (16 8))
      (0 0 "C-68-3-mini"
         ((3 48 53 3 1 6 0) (4 64 62 2 0 6 5)
          (4 72 67 2 0 6 6))))
     ((1 "rest")
      (NIL NIL "C-68-3-mini" NIL))
     ((3/2 NIL)
      (13 7 "C-68-3-mini"
          ((13/2 61 60 1/2 0 7 7))))
     ((2 NIL)
      (1 1 "C-68-3-mini"
         ((7 62 61 1/2 0 15/2 8))))
     ((5/2 NIL)
      (2 1 "C-68-3-mini"
         ((15/2 64 62 1/2 0 8 9))))
```

```
     ((3 (15))
      (-14 -8 "C-68-3-mini"
           ((8 50 54 1 1 9 10) (8 65 63 1 0 9 11)))))
```

This function is very similar to the function beat-spacing-states. Suppose you have three states X_{n-1}, X_n, X_{n+1}. The function beat-spacing-states looks at the beat and spacing of X_n, and also records the difference between the bass notes of X_n and X_{n-1}. The function beat-spacing-states<- looks at the beat and spacing of X_n, and also records the difference between the bass notes of X_{n+1} and X_n.

holding-type

Started, last checked	24/8/2010, 24/8/2010
Location	Spacing states
Calls	holding-type-finish, holding-type-normal, holding-type-start
Called by	holding-types
Comments/see also	

Example:

```
(holding-type
  '(1/2
    ((1/2 65 1/2 1 58 1 4) (1/3 58 1/3 1 69 2/3 3)
     (0 72 1 1 71 1 2) (0 60 1/2 1 66 1/2 1)))
  '(2/3
    ((2/3 56 1/3 1 60 1 5) (1/2 65 1/2 1 58 1 4)
     (1/3 58 1/3 1 69 2/3 3) (0 72 1 1 71 1 2)))
  '(1
    ((1 73 3/2 1 69 5/2 8) (1 64 1 1 69 2 7)
     (1 55 1 1 66 2 6) (2/3 56 1/3 1 60 1 5)
     (1/2 65 1/2 1 58 1 4) (0 72 1 1 71 1 2))))
--> ((56 0 5) (65 2 4) (72 2 2))
```

The function holding-type calls to one of holding-type-start, holding-type-normal or holding- type-finish, according to the emptiness of its arguments.

holding-type-finish

Started, last checked	24/8/2010, 24/8/2010
Location	Spacing states
Calls	index-item-1st-occurs, nth-list-of-lists
Called by	holding-type
Comments/see also	holding-type-normal, holding-type-start

Example:

```
(holding-type-finish
 '(11/2 ((4 67 2 1 55 6 20) (4 76 3/2 1 69 11/2 18)))
 '(6 ((4 67 2 1 55 6 20))))
--> (NIL NIL NIL)
```

The function holding-type-finish is called by the function holding-type in the event that the variable next-segment is empty. This only happens at the end of a list of segments. I am yet to think of an example where something other than an empty list should be returned.

holding-type-normal

Started, last checked	24/8/2010, 24/8/2010
Location	Spacing states
Calls	index-item-1st-occurs, my-last, nth-list-of-lists
Called by	holding-type
Comments/see also	holding-type-finish, holding-type-start

Example:

```
(holding-type-normal
 '(0 ((0 72 1 1 71 1 2) (0 60 1/2 1 66 1/2 1)
      (0 56 1/3 1 47 1/3 0)))
 '(1/3 ((1/3 58 1/3 1 69 2/3 3) (0 72 1 1 71 1 2)
(0 60 1/2 1 66 1/2 1) (0 56 1/3 1 47 1/3 0)))
 '(1/2 ((1/2 65 1/2 1 58 1 4) (1/3 58 1/3 1 69 2/3 3)
(0 72 1 1 71 1 2) (0 60 1/2 1 66 1/2 1))))
--> ((58 1 3) (72 3 2) (60 2 1))
```

The function holding-type-normal is called by the function holding-type, in ‘non-boundary circumstances’. Holding types are assigned appropriately, and returned as the second item in a sublist of lists, along with MIDI note numbers and identifiers.

holding-type-start

Started, last checked	24/8/2010, 24/8/2010
Location	Spacing states
Calls	index-item-1st-occurs, my-last, nth-list-of-lists
Called by	holding-type
Comments/see also	holding-type-finish, holding-type-normal

Example:

```
(holding-type-start
 '(0 ((0 72 1 1 71 1 2) (0 60 1/2 1 66 1/2 1)
      (0 56 1/3 1 47 1/3 0)))
 '(1/3 ((1/3 58 1/3 1 69 2/3 3) (0 72 1 1 71 1 2)
(0 60 1/2 1 66 1/2 1) (0 56 1/3 1 47 1/3 0))))
--> ((72 1) (60 1) (56 0))
```

The function holding-type-start is called by the function holding-type in the datapoint that the variable previous-segment is empty. This only happens at the beginning of a list of segments. Holding types are assigned appropriately, and returned as the second item in a sublist of lists, along with MIDI note numbers and identifiers. It is possible to generate an error using this function, if there are ontimes less than the initial ontime present.

holding-types

Started, last checked	24/8/2010, 24/8/2010
Location	Spacing states
Calls	holding-type
Called by	spacing-holding-states
Comments/see also	

Example:

```
(holding-types
 '((0 ((0 72 1 1 71 1 2) (0 60 1/2 1 66 1/2 1)
       (0 56 1/3 1 47 1/3 0)))
```

```
   (1/3 ((1/3 58 1/3 1 69 2/3 3) (0 72 1 1 71 1 2)
 (0 60 1/2 1 66 1/2 1) (0 56 1/3 1 47 1/3 0)))
   (1/2 ((1/2 65 1/2 1 58 1 4) (1/3 58 1/3 1 69 2/3 3)
 (0 72 1 1 71 1 2) (0 60 1/2 1 66 1/2 1)))
   (2/3 ((2/3 56 1/3 1 60 1 5) (1/2 65 1/2 1 58 1 4)
 (1/3 58 1/3 1 69 2/3 3) (0 72 1 1 71 1 2)))
   (1 ((2/3 56 1/3 1 60 1 5) (1/2 65 1/2 1 58 1 4)
       (0 72 1 1 71 1 2)))))
--> (((72 1 2) (60 1 1) (56 0 0))
     ((58 1 3) (72 3 2) (60 2 1))
     ((65 1 4) (58 2 3) (72 3 2))
     ((56 0 5) (65 2 4) (72 2 2))
     ((NIL NIL NIL) (NIL NIL NIL) (NIL NIL NIL)))
```

This function assigns holding-types to the datapoints at each segment: 0 for unheld; 1 for held- forward; 2 for held-backward; 3 for held-both. This information is returned, along with the MIDI note numbers and identifiers.

index-rests

Started, last checked	24/8/2010, 24/8/2010
Location	Spacing states
Calls	
Called by	spacing-holding-states
Comments/see also	

Example:

```
(index-rests
 '(((59 0 12) (63 0 13) (75 0 14)) NIL
   ((60 1 15) (63 0 16)) ((60 2 15)) (NIL NIL NIL)))
--> (1)
```

A list of sorted holdings is the only argument to this function. The output is the indices of those sub-lists which are empty (excluding the last sub-list) and therefore harbour rests.

intervals-above-bass

Started, last checked	24/8/2010, 24/8/2010
Location	Spacing states
Calls	
Called by	
Comments/see also	Deprecated.

Example:

```
(intervals-above-bass
 0 '((59 0 12) (63 1 13) (75 1 14)))
--> (0 4 16)
```

An index n is provided as first argument; a list of lists is the second argument. The nth item of each sub-list is a MIDI note number, and these sub-lists are in order of ascending MIDI note number. The intervals above the bass are returned. It is possible to produce nonsense output if null values are interspersed with non-null values. I use the function chord-spacing in preference to the function intervals-above-bass.

sort-holding-types

Started, last checked	24/8/2010, 24/8/2010
Location	Spacing states
Calls	sort-by
Called by	spacing-holding-states
Comments/see also	

Example:

```
(sort-holding-types
 '(((72 1 2) (60 1 1) (56 0 0)) ((NIL NIL NIL))
   ((58 1 3) (72 3 2) (58 3 1))))
--> (((56 0 0) (60 1 1) (72 1 2)) ((NIL NIL NIL))
     ((58 1 3) (58 3 1) (72 3 2)))
```

The sub-lists are returned, ordered by MIDI note number and then holding-type (both ascending). The function checks for empty chords to avoid errors occurring in the sort function.

spacing

Started, last checked	24/8/2010, 24/8/2010
Location	Spacing states
Calls	
Called by	beat-spacing-states, beat-spacing-states<-, spacing-holding-states
Comments/see also	intervals-above-bass

Example:

```
(spacing 0 '((59 0 12) (63 1 13) (75 1 14)))
--> (4 12)
```

An index n is provided as first argument; a list of lists is the second argument. The nth item of each sub-list is a MIDI note number, and these sub-lists are in order of ascending MIDI note number. The intervals between adjacent notes (chord spacing) are returned. It is possible to produce nonsense output if null values are interspersed with non-null values.

spacing-holding-states

Started, last checked	24/8/2010, 24/8/2010
Location	Spacing states
Calls	append-offtimes, bass-steps, enumerate-append, index-rests, nth-list, nth-list-of-lists, remove-nth-list, sort-holding-types, spacing
Called by	
Comments/see also	beat-spacing-states, beat-spacing-states<-

Example:

```
(spacing-holding-states
 '((0 74 1/2 1 84) (0 52 1 2 84) (1/2 76 1/4 1 84)
   (3/4 78 1/4 1 84) (1 80 1/4 1 84) (5/4 81 1/4 1 84)
   (3/2 83 1/2 1 84) (4 67 1 2 84) (4 64 1 2 84)
   (4 79 1/2 1 84) (9/2 78 1/2 1 84) (5 67 1 2 84)
   (5 64 1 2 84) (5 76 2 1 84)) "D Scarlatti L484" 2)
--> ((((22) (1 0))
      (NIL 1/2 "D Scarlatti L484"
```

```
  ((0 52 1 2 84 1 1) (0 74 1/2 1 84 1/2 0))))
(((24) (3 0))
 (0 1/4 "D Scarlatti L484"
  ((0 52 1 2 84 1 1) (1/2 76 1/4 1 84 3/4 2))))
(((26) (2 0))
 (0 1/4 "D Scarlatti L484"
  ((0 52 1 2 84 1 1) (3/4 78 1/4 1 84 1 3))))
((NIL (0))
 (28 1/4 "D Scarlatti L484"
  ((1 80 1/4 1 84 5/4 4))))
((NIL (0))
 (1 1/4 "D Scarlatti L484"
  ((5/4 81 1/4 1 84 3/2 5))))
((NIL (0))
 (2 5/2 "D Scarlatti L484"
  ((3/2 83 1/2 1 84 2 6))))
(((3 12) (1 1 0))
 (-19 1/2 "D Scarlatti L484"
  ((4 64 1 2 84 5 8) (4 67 1 2 84 5 7)
   (4 79 1/2 1 84 9/2 9))))
(((3 11) (2 2 0))
 (0 1/2 "D Scarlatti L484"
  ((4 64 1 2 84 5 8) (4 67 1 2 84 5 7)
   (9/2 78 1/2 1 84 5 10))))
(((3 9) (0 0 1))
 (0 1 "D Scarlatti L484"
  ((5 64 1 2 84 6 12) (5 67 1 2 84 6 11)
   (5 76 2 1 84 7 13))))
((NIL (2))
 (12 1 "D Scarlatti L484"
  ((5 76 2 1 84 7 13)))))
```

This function takes datapoints as its argument, and some optional catalogue information about those datapoints. It converts the input into a list of sub-lists, with each sub-list consisting of a pair of lists. The first of the pair contains a chord spacing, followed by holding types relating to the notes of the chord. The second of the pair retains the following information: the step (in semitones) between the bass note of the previous chord and the current state; the duration of the state (which can exceed the minimum duration of the constituent datapoints if rests are present); the catalogue information; the relevant original datapoints, with offtimes and enumeration appended.

4.6.8 Generating beat MNN spacing forwards

The main function here is called generate-beat-MNN-spacing->. Given initial states, a state-transition matrix, an upper limit for ontime, and a template dataset, it generates datapoints (among other output) that conform to various criteria, which can be specified using the optional arguments. The criteria are things like: not too many consecutive states from the same source, the range is comparable with that of the template, and the likelihood of the states is comparable with those of the template.

checklistp

Started, last checked	28/9/2010, 28/9/2010
Location	Generating beat MNN spacing forwards
Calls	comparable-likelihood-profilep, index-item-1st-doesnt-occur, lastn, mean&rangep, my-last, pitch&octave-spellingp, segments-strict
Called by	generate-beat-MNN-spacing->, generate-beat-spacing-forcing->
Comments/see also	checklist<-p

Example:

```
(setq
 states
 '(((1 (60 72 79 84 88) (60 67 71 74 76))
      (NIL NIL "C-63-1"
           ((0 35 45 1/2 1 1/2 0)
            (0 47 52 1/2 1 1/2 1)
            (0 54 56 1/2 0 1/2 2)
            (0 59 59 1/2 0 1/2 3)
            (0 63 61 1/2 0 1/2 4))))
   ((3/2 NIL NIL)
      (NIL NIL "C-63-1" NIL))
   ((7/4 (54) (57))
      (-6 -3 "C-63-1"
          ((831/4 56 57 1/4 0 208 701))))
   ((2 (28 40 54 60 63) (42 49 57 60 62))
      (-26 -15 "C-63-1"
           ((208 30 42 1 1 209 702)
```

```
            (208 42 49 1 1 209 703)
            (208 56 57 1 0 209 704)
            (208 62 60 1 0 209 705)
            (208 65 62 1 0 209 706))))))
(setq datapoints nil)
(setq checklist '("originalp"))
(setq c-sources 3)
(setq c-bar 12)
(setq c-min 7)
(setq c-max 7)
(setq c-beats 12)
(setq c-prob 0.1)
(checklistp
 states datapoints dataset-template
 template-segments checklist c-sources c-bar c-min
 c-max c-beats c-prob)
--> NIL
```

This function checks that the previous c-sources sources are not all the same, it checks whether the range is acceptable (using c-bar, c-min, and c-max), and whether the chord likelihoods are acceptable (using c-beats and c-prob). If all three of these aspects are acceptable, the function returns T, and NIL otherwise.

choose-one-with-beat

Started, last checked	28/9/2010, 28/9/2010
Location	Generating beat MNN spacing forwards
Calls	
Called by	generate-beat-MNN-spacing->, generate-beat-spacing-forcing->
Comments/see also	

Example:

```
(setq
 mini-initial-states
 '(((3 NIL)
    (NIL NIL "C-6-1" ((-1 66 63 4/3 0 1/3 0))))
   ((1 (7 9 8))
    (NIL NIL "C-6-2"
```

```
     ((0 44 50 1 1 1 0) (0 51 54 1 1 1 1)
      (0 60 59 1 1 1 2) (0 68 64 1 0 1 3))))
   ((1 (7))
    (NIL NIL "C-6-3"
     ((0 52 55 1 1 1 0) (0 59 59 1 1 1 1))))
   ((3 NIL)
    (NIL NIL "C-6-4" ((-1 70 66 3/2 0 1/2 0))))
   ((1 (24))
    (NIL NIL "C-7-1"
     ((0 41 49 1 1 1 0) (0 65 63 1/2 0 1/2 1))))))
(choose-one-with-beat 1 mini-initial-states)
--> ((1 (7))
     (NIL NIL "C-6-3"
      ((0 52 55 1 1 1 0) (0 59 59 1 1 1 1))))
```

This function takes a beat of a bar and a list of initial states as its arguments. These states may be genuinely initial, or constructed from appropriate beginnings. A search of the states is made beginning at a random index, and the first state whose beat matches that of the first argument is returned. In the event that all the states are searched, the output reverts to the original random index, so something is always output.

comparable-likelihood-profilep

Started, last checked	28/9/2010, 28/9/2010
Location	Generating beat MNN spacing forwards
Calls	geom-mean-likelihood-of-subset, linearly-interpolate, max-item, min-item, nth-list-of-lists, orthogonal-projection-not-unique-equalp
Called by	checklistp
Comments/see also	comparable-likelihood-profile<-p

Example:

```
(setq
 ontime-state-points-pair
 '(7/2
   ((3 45 51 1 0) (7/2 71 66 1/2 0))))
(setq
 datapoints
```

```
 '((0 52 55 1 1) (0 59 59 1 1) (0 64 62 1 0)
   (0 68 64 1 0) (1 52 55 2 1) (1 59 59 1 1)
   (1 64 62 1 0) (1 68 64 1/2 0) (3/2 69 65 1/2 0)
   (2 62 61 1 0) (2 71 66 1/2 0) (5/2 66 63 1/2 0)
   (3 45 51 1 0) (3 64 62 1/2 0) (7/2 71 66 1/2 0)
   (4 52 55 2 1) (4 57 58 2 1) (4 61 60 2 1)
   (4 69 65 1/2 0) (9/2 73 67 1/2 0) (5 76 69 1 0)
   (6 38 47 1 1) (6 71 66 1/2 0) (13/2 69 65 1/2 0)
   (7 50 54 1 1) (7 54 56 1 1) (7 57 58 2 1)
   (7 66 63 1/2 0) (15/2 67 64 1/2 0) (8 49 53 1 1)
   (8 52 55 1 1) (8 69 65 1 0) (9 33 44 1 1)
   (9 64 62 1/2 0) (19/2 61 60 1/2 0) (10 45 51 1 1)
   (10 52 55 1 1) (10 57 58 1 0) (11 45 51 1 1)
   (11 52 55 1 1) (11 64 62 2 0) (12 45 51 1 1)
   (12 57 58 1 1) (12 60 60 1 0)))
(setq c-beats 12)
(setq
 template-likelihood-profile
 '((0 0.19999999) (1 0.19999999) (2 0.16054831)
   (11/4 0.1875) (3 0.10939984) (4 0.07914095)
   (19/4 0.09988681) (5 0.08333333) (6 0.03448276)))
(setq c-prob 0.1)
(comparable-likelihood-profilep
 ontime-state-points-pair datapoints c-beats
 template-likelihood-profile c-prob)
--> T
```

This function takes a pair consisting of an ontime and points that sound at the ontime. Based on an empirical distribution over the argument named datapoints (with a context governed by c-beats), it calculates the geometric mean of the likelihood of these points. This likelihood is then compared with that at the same ontime in the template. If the absolute difference between the likelihoods is less than the threshold c-prob, T is returned, and NIL otherwise. T will also be returned if there are no points.

full-segment-nearest¡ontime

Started, last checked	28/9/2010, 28/9/2010
Location	Generating beat MNN spacing forwards
Calls	index-1st-sublist-item>, nth-list-of-lists
Called by	mean&rangep
Comments/see also	

Example:

```
(setq
 mini-template-segments
 '((0
    ((0 52 55 1 1 1 0) (0 62 61 1 0 1 1)
     (0 64 62 1 0 1 2) (0 68 64 1 0 1 3)
     (0 71 66 1 0 1 4)))
   (1
    ((1 52 55 1 1 2 5) (1 62 61 1 0 2 6)
     (1 64 62 1 0 2 7) (1 68 64 1 0 2 8)
     (1 71 66 1 0 2 9)))
   (2
    ((2 52 55 1 1 3 10) (2 62 61 1 0 3 11)
     (2 64 62 1 0 3 12) (2 68 64 1 0 3 13)
     (2 72 67 3/4 0 11/4 14)))
   (11/4
    ((2 52 55 1 1 3 10) (2 62 61 1 0 3 11)
     (2 64 62 1 0 3 12) (2 68 64 1 0 3 13)
     (11/4 71 66 1/4 0 3 15)))
   (3
    ((3 45 51 3 1 6 16) (3 52 55 3 1 6 17)
     (3 60 60 3 0 6 18) (3 64 62 3 0 6 19)
     (3 71 66 1 0 4 20)))))
(full-segment-nearest<ontime 1 mini-template-segments)
--> (1
     ((1 52 55 1 1 2 5) (1 62 61 1 0 2 6)
      (1 64 62 1 0 2 7) (1 68 64 1 0 2 8)
      (1 71 66 1 0 2 9)))
```

This function takes an ontime and a list of segments as its arguments. It returns the full (that is, non-null) segment whose ontime is closest to and less than the first argument. There should always be such a segment, but if there is not, NIL is returned.

generate-beat-MNN-spacing->

Started, last checked	28/9/2010, 28/9/2010
Location	Generating beat MNN spacing forwards
Calls	checklistp, choose-one, choose-one-with-beat, geom-mean-likelihood-of-states, index-1st-sublist-item>=, my-last, segments-strict, sort-dataset-asc, states2datapoints-by-lookup, translate-datapoints-to-first-ontime
Called by	
Comments/see also	Consider subdividing into several functions. See also generate-beat-MNN-spacing<-, generate-beat-spacing-forcing->.

Example:

```
(progn
  (setq
   initial-states
   (read-from-file
    (concatenate
     'string
     *MCStylistic-Oct2010-example-files-path*
     "/Racchman-Oct2010 example"
     "/initial-states.txt")))
  (setq
   stm
   (read-from-file
    (concatenate
     'string
     *MCStylistic-Oct2010-example-files-path*
     "/Racchman-Oct2010 example"
     "/transition-matrix.txt")))
  (setq no-ontimes> 6)
  (setq
   dataset-all
   (read-from-file
    (concatenate
     'string
```

```
     *MCStylistic-Oct2010-data-path*
     "/Dataset/C-41-2-ed.txt")))
  (setq
   dataset-template
   (subseq dataset-all 0 184))
  (setq
   *rs* #.(CCL::INITIALIZE-RANDOM-STATE 477 10894))
  "Data loaded and *rs* set.")
(generate-beat-MNN-spacing->
 initial-states stm no-ontimes> dataset-template)
--> ((((1 NIL)
       (NIL NIL "C-59-1" ((0 76 69 1/2 0 1/2 0))))
      ((3/2 NIL)
       (2 1 "C-56-1" ((289/2 72 67 1/2 0 145 495))))
      ((7/4 (26 12))
       (0 0 "C-30-2"
        ((39 38 47 1 1 40 134)
         (159/4 64 62 1/4 0 40 136)
         (159/4 76 69 1/4 0 40 137))))
      ((2 (7 5 12))
       (12 7 "C-30-2"
        ((40 50 54 1 1 41 138) (40 57 58 1 1 41 139)
         (40 62 61 2 0 42 140)
         (40 74 68 2 0 42 141))))
      ((5/2 (7 9 8))
       (0 0 "C-63-1"
        ((46 54 56 1 1 47 209) (46 61 60 1 1 47 210)
         (93/2 70 65 1/2 0 47 213)
         (46 78 70 5/2 0 97/2 212))))
      ((3 (7 5 7))
       (0 0 "C-63-2"
        ((92 43 50 1 1 93 302) (92 50 54 1 1 93 303)
         (92 55 57 1 0 93 304)
         (92 62 61 1 0 93 305))))
      ((1 (5 2))
       (13 8 "C-63-2"
        ((72 56 58 1/2 1 145/2 232)
         (72 61 61 2 0 74 233)
         (72 63 62 2 0 74 234))))
      ((3/2 (6 2))
       (-1 -1 "C-50-3"
```

```
    ((277/2 67 63 1/2 1 139 448)
     (138 73 67 1 0 139 447)
     (137 75 68 2 0 139 446))))
  ((2 NIL)
   (-16 -9 "C-50-3"
    ((415 51 54 1/2 1 831/2 1350))))
  ((3 (9))
   (7 4 "C-17-1"
    ((131 46 52 1 1 132 647)
     (131 55 57 1 1 132 648)))))
 ((0 52 55 1/2 0) (1/2 54 56 1/2 0)
  (3/4 80 71 1/4 0) (3/4 92 78 1/4 0)
  (1 66 63 3 1) (1 73 67 3 1) (1 78 70 1/2 0)
  (1 90 77 1 0) (3/2 82 72 1/2 0) (2 78 70 2 0)
  (2 85 74 2 0) (4 79 71 1/2 1) (4 84 74 1 0)
  (4 86 75 1 0) (9/2 78 70 1/2 1) (5 62 61 1 1)
  (6 69 65 1 1) (6 78 70 1 1))
 376 (0 0 0 0 0 0 3 0 0 0))
```

Given initial states, a state-transition matrix, an upper limit for ontime, and a template dataset, this function generates datapoints (among other output) that conform to various criteria, which can be specified using the optional arguments. The criteria are things like: not too many consecutive states from the same source (c-sources), the range is comparable with that of the template (c-bar, c-min, and c-max), and the likelihood of the states is comparable with that of the template (c-beats and c-prob). A record, named index-failures, is kept of how many times a generated state fails to meet the criteria. When a threshold c-failures is exceeded at any state index, the penultimate state is removed and generation continues. If the value of c-failures is exceeded for the first state, a message is returned.

This function returns the states as well as the datapoints, and also index-failures and i, the total number of iterations.

generate-beat-spacing-forcing->

Started, last checked	28/9/2010, 28/9/2010
Location	Generating beat MNN spacing forwards
Calls	beat-spacing-states, checklistp, choose-one, choose-one-with-beat, geom-mean-likelihood-of-states, index-1st-sublist-item>=, my-last, segments-strict, sort-dataset-asc, states2datapoints-by-lookup, translate-datapoints-to-first-ontime
Called by	
Comments/see also	Consider subdividing into several functions. See also generate-beat-MNN-spacing->, generate-beat-spacing-forcing<-.

Example:

```
(progn
  (setq
   initial-states
   (read-from-file
    (concatenate
     'string
     *MCStylistic-Oct2010-example-files-path*
     "/Racchman-Oct2010 example"
     "/initial-states.txt")))
  (setq
   stm
   (read-from-file
    (concatenate
     'string
     *MCStylistic-Oct2010-example-files-path*
     "/Racchman-Oct2010 example"
     "/transition-matrix.txt")))
  (setq no-ontimes> 14)
  (setq
   dataset-all
   (read-from-file
    (concatenate
     'string
```

```
     *MCStylistic-Oct2010-data-path*
     "/Dataset/C-41-2-ed.txt")))
  (setq
   dataset-template
   (subseq dataset-all 0 184))
  (setq
   *rs* #.(CCL::INITIALIZE-RANDOM-STATE 477 10894))
  (setq
   datapoints-from-previous-interval
   '((17/2 63 62 1/2 0) (17/2 72 67 1/2 0)
     (9 34 45 1 1) (9 46 52 1 1) (9 65 63 1/2 0)
     (9 74 68 1/2 0) (39/4 65 63 1/4 0)))
  (setq
   previous-state-context-pair
   (my-last
    (beat-spacing-states
     datapoints-from-previous-interval
     "No information" 3 1 3)))
  (setq generation-interval '(13 15))
  (setq
   pattern-region
   '((27/2 66 63) (27/2 69 65) (14 55 57) (14 62 61)
     (14 67 64) (14 71 66) (29/2 69 65) (29/2 72 67)))
  "Data loaded and variables set.")
(generate-beat-spacing-forcing->
 initial-states stm no-ontimes> dataset-template
 generation-interval pattern-region
 previous-state-context-pair (list "originalp")
 3 10 4 19 12 12 12 .15)
--> ((((7/4 (12 19))
       (0 0 "No information"
        ((9 34 45 1 1 10 2) (9 46 52 1 1 10 3)
         (39/4 65 63 1/4 0 10 6))))
      ((2 (7 5 4 3 5))
       (12 7 "C-17-1"
        ((202 46 52 1 1 203 908)
         (202 53 56 1 1 203 909)
         (202 58 59 1 1 203 910)
         (202 62 61 1 0 203 911)
         (202 65 63 1 0 203 912)
         (202 70 66 1 0 203 913))))
```

```
     ((3 (7 5 4 3 9))
      (0 0 "C-17-1"
       ((203 46 52 1 1 204 914)
        (203 53 56 1 1 204 915)
        (203 58 59 1 1 204 916)
        (203 62 61 1 0 204 917)
        (203 65 63 1 0 204 918)
        (203 74 68 1 0 204 919))))
     ((15/4 (7 5 4 3 9))
      (0 0 "C-41-2"
       ((137 47 52 1 1 138 539)
        (137 54 56 1 1 138 540)
        (137 59 59 1 1 138 541)
        (137 63 61 1 0 138 542)
        (137 66 63 1 0 138 543)
        (551/4 75 68 1/4 0 138 544))))
     ((1 NIL)
      (28 16 "C-41-2" ((66 75 68 2 0 68 261))))
     ((3/2 NIL)
      (1 1 "C-30-1" ((61/2 75 69 1/2 0 31 110))))
     ((2 NIL)
      (-5 -3 "C-56-1"
       ((409 68 65 1/2 0 819/2 1334))))
     ((3 NIL)
      (0 0 "C-56-3" ((143 62 61 1 0 144 423)))))
    ((9 34 45 1 1) (9 46 52 2 1) (39/4 65 63 1/4 0)
     (10 53 56 1 1) (10 58 59 1 1) (10 62 61 1 0)
     (10 65 63 1 0) (10 70 66 1 0) (11 46 52 1 1)
     (11 53 56 1 1) (11 58 59 1 1) (11 62 61 1 0)
     (11 65 63 1 0) (11 74 68 1 0) (12 74 68 1/2 0)
     (25/2 75 69 1/2 0) (13 70 66 1 0)
     (14 70 66 1 0))
    20 (0 0 0 1 0 0 0 0))
```

This function appears to be very simiar to the function generate-beat-MNN-spacing->. The difference is that there are some extra arguments here, which allow for using either external or internal initial/final states, and for using information from a discovered pattern or previous/next state to further guide the generation, hence ‘forcing’.

geom-mean-likelihood-of-states

Started, last checked	28/9/2010, 28/9/2010
Location	Generating beat MNN spacing forwards
Calls	geom-mean-likelihood-of-subset, max-item, min-item, nth-list-of-lists, orthogonal-projection-not-unique-equalp
Called by	generate-beat-MNN-spacing->, generate-beat-spacing-forcing->
Comments/see also	

Example:

```
(progn
  (setq
   ontime-state-points-pairs
   '((0 ((0 38 47 1/2 1 1/2 0) (0 62 61 1/2 0 1/2 1)))
     (1/2 NIL)
     (1
      ((1 50 54 2 1 3 2) (1 57 58 1/2 1 3/2 3)
       (1 60 60 3 0 4 4) (1 66 63 1/2 0 3/2 5)))
     (3/2
      ((1 50 54 2 1 3 2) (3/2 55 57 1/2 1 2 6)
       (1 60 60 3 0 4 4) (3/2 64 62 1/2 0 2 7)))
     (2
      ((1 50 54 2 1 3 2) (2 54 56 1/2 1 5/2 8)
       (1 60 60 3 0 4 4) (2 62 61 1/2 0 5/2 9)))
     (5/2
      ((1 50 54 2 1 3 2) (5/2 57 58 1/2 1 3 10)
       (1 60 60 3 0 4 4) (5/2 66 63 1/2 0 3 11)))
     (3
      ((3 50 54 15/4 1 27/4 12) (1 60 60 3 0 4 4)
       (3 69 65 1/2 0 7/2 13)))
     (7/2
      ((3 50 54 15/4 1 27/4 12) (1 60 60 3 0 4 4)))
     (15/4
      ((3 50 54 15/4 1 27/4 12) (1 60 60 3 0 4 4)
       (15/4 71 66 1/4 0 4 14)))
     (4
      ((3 50 54 15/4 1 27/4 12) (4 60 60 2 0 6 15)
       (4 62 61 2 1 6 16) (4 66 63 2 0 6 17)
```

```
      (4 72 67 2 0 6 18)))
    (6
     ((6 43 50 2 1 8 19) (3 50 54 15/4 1 27/4 12)
      (6 62 61 1/2 0 13/2 20) (6 67 64 1/2 0 13/2 21)
      (6 71 66 1/2 0 13/2 22)))
    (13/2
     ((6 43 50 2 1 8 19) (3 50 54 15/4 1 27/4 12)))
    (27/4
     ((6 43 50 2 1 8 19) (27/4 50 54 1/4 1 7 23)
      (27/4 60 60 1/4 0 7 24)
      (27/4 69 65 1/4 0 7 25)))
    (7
     ((6 43 50 2 1 8 19) (7 55 57 2 1 9 26)
      (7 59 59 1 0 8 27) (7 67 64 1 0 8 28)))
    (8
     ((8 43 50 2 1 10 29) (7 55 57 2 1 9 26)
      (8 59 59 1 0 9 30) (8 62 61 1 0 9 31)
      (8 71 66 1 0 9 32)))
    (9
     ((8 43 50 2 1 10 29) (9 49 53 1 1 10 33)
      (9 58 58 1 1 10 34) (9 64 62 1 0 10 35)
      (9 67 64 1 0 10 36) (9 76 69 1 0 10 37)))
    (10 NIL)))
 (setq
  dataset
  '((0 38 47 1/2 1) (0 62 61 1/2 0) (1 50 54 2 1)
    (1 57 58 1/2 1) (1 60 60 3 0) (1 66 63 1/2 0)
    (3/2 55 57 1/2 1) (3/2 64 62 1/2 0)
    (2 54 56 1/2 1) (2 62 61 1/2 0) (5/2 57 58 1/2 1)
    (5/2 66 63 1/2 0) (3 50 54 15/4 1)
    (3 69 65 1/2 0) (15/4 71 66 1/4 0) (4 60 60 2 0)
    (4 62 61 2 1) (4 66 63 2 0) (4 72 67 2 0)
    (6 43 50 2 1) (6 62 61 1/2 0) (6 67 64 1/2 0)
    (6 71 66 1/2 0) (27/4 50 54 1/4 1)
    (27/4 60 60 1/4 0) (27/4 69 65 1/4 0)
    (7 55 57 2 1) (7 59 59 1 0) (7 67 64 1 0)
    (8 43 50 2 1) (8 59 59 1 0) (8 62 61 1 0)
    (8 71 66 1 0) (9 49 53 1 1) (9 58 58 1 1)
    (9 64 62 1 0) (9 67 64 1 0) (9 76 69 1 0)))
 (setq c-beats 4)
 "Variables set.")
```

```
(geom-mean-likelihood-of-states
 ontime-state-points-pairs dataset c-beats)
--> ((0 0.5) (1 0.16666667) (3/2 0.125) (2 0.11892071)
     (5/2 0.11785113) (3 0.089994356)
     (7/2 0.101015255) (15/4 0.08399473)
     (4 0.10777223) (6 0.075700045) (13/2 0.061487548)
     (27/4 0.09343293) (7 0.05662891) (8 0.09750821)
     (9 0.058608957))
```

This function applies the function geom-mean-likelihood-of-subset recursively, having extracted a subset from each ontime-state-points pair.

geom-mean-likelihood-of-subset

Started, last checked	28/9/2010, 28/9/2010
Location	Generating beat MNN spacing forwards
Calls	empirical-mass, index-1st-sublist-item>=, index-1st-sublist-item>, likelihood-of-subset
Called by	comparable-likelihood-profilep, geom-mean-likelihood-of-states
Comments/see also	

Example:

```
(progn
  (setq
   subset
   '((8 43 50 2 1 10 29) (7 55 57 2 1 9 26)
     (8 59 59 1 0 9 30) (8 62 61 1 0 9 31)
     (8 71 66 1 0 9 32)))
  (setq
   subset-palette
   (orthogonal-projection-not-unique-equalp
    subset '(0 1)))
  (setq first-subset-ontime 7)
  (setq last-subset-ontime 8)
  (setq
   dataset
   '((0 38 47 1/2 1) (0 62 61 1/2 0) (1 50 54 2 1)
     (1 57 58 1/2 1) (1 60 60 3 0) (1 66 63 1/2 0)
     (3/2 55 57 1/2 1) (3/2 64 62 1/2 0)
```

```
     (2 54 56 1/2 1) (2 62 61 1/2 0) (5/2 57 58 1/2 1)
     (5/2 66 63 1/2 0) (3 50 54 15/4 1)
     (3 69 65 1/2 0) (15/4 71 66 1/4 0) (4 60 60 2 0)
     (4 62 61 2 1) (4 66 63 2 0) (4 72 67 2 0)
     (6 43 50 2 1) (6 62 61 1/2 0) (6 67 64 1/2 0)
     (6 71 66 1/2 0) (27/4 50 54 1/4 1)
     (27/4 60 60 1/4 0) (27/4 69 65 1/4 0)
     (7 55 57 2 1) (7 59 59 1 0) (7 67 64 1 0)
     (8 43 50 2 1) (8 59 59 1 0) (8 62 61 1 0)
     (8 71 66 1 0) (9 49 53 1 1) (9 58 58 1 1)
     (9 64 62 1 0) (9 67 64 1 0) (9 76 69 1 0)))
  (setq
   dataset-palette
   (orthogonal-projection-not-unique-equalp
    dataset '(0 1)))
  (setq
   ontimes-list (nth-list-of-lists 0 dataset))
  (setq c-beats 4)
  "Variables set.")
(geom-mean-likelihood-of-subset
 subset subset-palette first-subset-ontime
 last-subset-ontime dataset-palette
 ontimes-list c-beats)
--> 0.09750821
```

The first argument to this function, called subset, is a point set. Both in the scenario of likelihood calculation for an original excerpt and for a generated passage, the point set is a segment of the music. The argument subset-palette consists of a (listed) list of MIDI note numbers from the subset. Note: first-subset-ontime is not necessarily the ontime of the first datapoint, as they will have been sorted by MIDI note number. The variables dataset and dataset-palette are analogous, ontimes- list is a list of ontimes from the dataset. The threshold c-beats determines how far back we look to form the empirical distribution. The output of this function is the geometric mean of the likelihood of the subset (that is, a product of the individual empirical probabilities of the constituent MIDI note numbers.

mean&rangep

Started, last checked	28/9/2010, 28/9/2010
Location	Generating beat MNN spacing forwards
Calls	full-segment-nearest¡ontime, max-item, min-item, nth-list-of-lists
Called by	checklistp
Comments/see also	

Example:

```
(setq
 mini-template-segments
 '((0
    ((0 52 55 1 1 1 0) (0 62 61 1 0 1 1)
     (0 64 62 1 0 1 2) (0 68 64 1 0 1 3)
     (0 71 66 1 0 1 4)))
   (1
    ((1 52 55 1 1 2 5) (1 62 61 1 0 2 6)
     (1 64 62 1 0 2 7) (1 68 64 1 0 2 8)
     (1 71 66 1 0 2 9)))
   (2
    ((2 52 55 1 1 3 10) (2 62 61 1 0 3 11)
     (2 64 62 1 0 3 12) (2 68 64 1 0 3 13)
     (2 72 67 3/4 0 11/4 14)))
   (11/4
    ((2 52 55 1 1 3 10) (2 62 61 1 0 3 11)
     (2 64 62 1 0 3 12) (2 68 64 1 0 3 13)
     (11/4 71 66 1/4 0 3 15)))
   (3
    ((3 45 51 3 1 6 16) (3 52 55 3 1 6 17)
     (3 60 60 3 0 6 18) (3 64 62 3 0 6 19)
     (3 71 66 1 0 4 20)))))
(setq
 datapoints-segment
 '(3/2 ((1 64 62 1 0 2 7) (1 68 64 1 0 2 8))))
(mean&rangep
 datapoints-segment mini-template-segments 4 3 3)
--> NIL
```

This function takes five arguments: a pair consisting of an ontime and a list of datapoints, a list of pairs as above, and three thresholds. It uses the ontime in the first argument to determine which list is relevant in the second argument. Then the two sets of MIDI note numbers are compared, in terms of their mean, min, and max values. If, for each of these summary statistics, the absolute difference between each set of MNNs is smaller than the threshold, then T is returned, and NIL otherwise.

pitch&octave-spellingp

Started, last checked	28/9/2010, 28/9/2010
Location	Generating beat MNN spacing forwards
Calls	MIDI-morphetic-pair2pitch&octave
Called by	checklistp
Comments/see also	

Example:

```
(pitch&octave-spellingp
 '((12 50 54 1 1 13 42) (12 62 61 1 0 13 43)
   (12 65 63 1 0 13 44) (12 69 65 1 0 13 45)))
--> T
```

This function converts each MIDI-morphetic pair (assumed to be second and third entries of each list) to pitch and octave number. If the spelling requires more than two flats or sharps, then NIL will be returned, and T otherwise.

translate-datapoints-to-first-ontime

Started, last checked	28/9/2010, 28/9/2010
Location	Generating beat MNN spacing forwards
Calls	constant-vector, translation
Called by	generate-beat-MNN-spacing->, generate-beat-spacing-forcing->
Comments/see also	translate-datapoints-to-last-ontime

Example:

```
(translate-datapoints-to-first-ontime
 4 0
 '((28 44 51 1 1) (28 48 53 1 1) (28 56 58 1 0)
   (30 44 51 1 1) (31 48 53 1 1) (34 56 58 1 0)))
--> ((4 44 51 1 1) (4 48 53 1 1) (4 56 58 1 0)
     (6 44 51 1 1) (7 48 53 1 1) (10 56 58 1 0))
```

This function takes three arguments: an ontime, an ontime index and a list of datapoints (assumed to be sorted in lexicographical order). It translates these datapoints such that the ontime of the first datapoint equals the first argument.

4.6.9 Generating beat MNN spacing backwards

The main function here is called generate-beat-MNN-spacing<-. Given final states, a state-transition matrix, a lower limit for the ontime of the first state, and a template dataset, it generates datapoints (among other output) that conform to various criteria, which can be specified using the optional arguments. The criteria are things like: not too many consecutive states from the same source, the range is comparable with that of the template, and the likelihood of the states is comparable with that of the template.

checklist<-p

Started, last checked	12/10/2010, 12/10/2010
Location	Generating beat MNN spacing backwards
Calls	comparable-likelihood-profile<-p, index-item-1st-doesnt-occur, lastn, mean&rangep, pitch&octave-spellingp, segments-strict
Called by	generate-beat-MNN-spacing<-, generate-beat-spacing-forcing<-
Comments/see also	checklistp

Example:

```
(setq
 states<-
 '(((2 (7 5 4))
    (NIL NIL "C-24-2"
     ((358 48 53 2 1 360 6) (358 55 57 2 1 360 7)
      (358 60 60 2 0 360 8) (358 64 62 2 0 360 9))))
   ((3/2 (19 7))
    (12 7 "C-30-2"
     ((153 45 51 1 1 154 599) (153 64 62 1 0 154 600)
      (307/2 71 66 1/2 0 154 602))))))
(setq
 datapoints
```

```
 '((67/2 32 44 1/2 1) (67/2 51 55 5/2 0)
   (67/2 58 59 1/2 0) (34 44 51 2 1) (34 56 58 2 0)
   (34 60 60 2 0)))
(setq
 template-segments
 '((27
    ((27 39 48 1 1 28 91) (27 63 62 1/2 0 55/2 92)))
   (55/2 ((27 39 48 1 1 28 91)))
   (111/4
    ((27 39 48 1 1 28 91) (111/4 72 67 1/4 0 28 93)))
   (28
    ((28 51 55 1 1 29 94) (28 60 60 1 1 29 95)
     (28 68 65 1 1 29 96) (28 84 74 1 0 29 97)))
   (29 ((29 82 73 1/3 0 88/3 98)))
   (88/3 ((88/3 80 72 1/3 0 89/3 99)))
   (89/3 ((89/3 77 70 1/3 0 30 100)))
   (30
    ((30 39 48 1 1 31 101) (30 79 71 1/2 0 61/2 102)))
   (61/2
    ((30 39 48 1 1 31 101) (61/2 77 70 1/2 0 31 103)))
   (31
    ((31 51 55 1 1 32 104) (31 55 57 1 1 32 105)
     (31 61 61 1 1 32 106) (31 75 69 1/2 0 63/2 107)))
   (63/2
    ((31 51 55 1 1 32 104) (31 55 57 1 1 32 105)
     (31 61 61 1 1 32 106) (63/2 73 68 1/2 0 32 108)))
   (32
    ((32 51 55 1 1 33 109) (32 55 57 1 1 33 110)
     (32 61 61 1 1 33 111) (32 70 66 1/2 0 65/2 112)))
   (65/2
    ((32 51 55 1 1 33 109) (32 55 57 1 1 33 110)
     (32 61 61 1 1 33 111) (65/2 65 63 1/2 0 33 113)))
   (33
    ((33 44 51 2 1 35 114) (33 63 62 1/2 0 67/2 115)))
   (67/2 ((33 44 51 2 1 35 114)))
   (135/4
    ((33 44 51 2 1 35 114)
     (135/4 72 67 1/4 0 34 116)))
   (34
    ((33 44 51 2 1 35 114) (34 51 55 1 1 35 117)
     (34 60 60 1 1 35 118) (34 68 65 1 0 35 119)))))
```

```
(setq
 template-likelihood-profile
 '((27 0.11785113) (55/2 1/6) (111/4 0.10878566)
   (28 0.08494337) (29 1/15) (88/3 1/14) (89/3 1/7)
   (30 0.06666667) (61/2 0.06666667) (31 0.11632561)
   (63/2 0.11632561) (32 0.108109735)
   (65/2 0.108109735) (33 0.16666667) (67/2 1/6)
   (135/4 0.16666667) (34 0.16666667)))
(setq
 checklist '("originalp" "range&meanp" "likelihoodp"))
(setq c-sources 3)
(setq c-bar 12)
(setq c-min 7)
(setq c-max 7)
(setq c-beats 3)
(setq c-prob 0.1)
(checklist<-p
 states<- datapoints template-segments
 template-likelihood-profile checklist c-sources c-bar
 c-min c-max c-beats c-prob)
--> T
```

Checks are made of sources, of the range and mean of the notes supplied in the last element of the backwards-generated states, and their likelihoods.

comparable-likelihood-profile<-p

Started, last checked	12/10/2010, 12/10/2010
Location	Generating beat MNN spacing backwards
Calls	geom-mean-likelihood-of-subset<-, linearly-interpolate, max-item, min-item, nth-list-of-lists, orthogonal-projection-not-unique-equalp
Called by	checklist<-p
Comments/see also	comparable-likelihood-profilep

Example:

```
(setq
 ontime-state-points-pair
 '(111/4
```

```
   ((27 39 48 1 1) (111/4 72 67 1/4 0))))
(setq
 datapoints
 '((27 39 48 1 1) (27 63 62 1/2 0) (111/4 72 67 1/4 0)
   (28 51 55 1 1) (28 60 60 1 1) (28 68 65 1 1)
   (28 84 74 1 0) (29 82 73 1/3 0) (88/3 80 72 1/3 0)
   (89/3 77 70 1/3 0) (30 39 48 1 1) (30 79 71 1/2 0)
   (61/2 77 70 1/2 0) (31 51 55 1 1) (31 55 57 1 1)
   (31 61 61 1 1) (31 75 69 1/2 0) (63/2 73 68 1/2 0)
   (32 51 55 1 1) (32 55 57 1 1) (32 61 61 1 1)
   (32 70 66 1/2 0) (65/2 65 63 1/2 0) (33 44 51 2 1)
   (33 63 62 1/2 0) (135/4 72 67 1/4 0) (34 51 55 1 1)
   (34 60 60 1 1) (34 68 65 1 0)))
(setq c-beats 3)
(setq
 template-likelihood-profile
 '((27 0.07142857) (55/2 1/14) (111/4 0.062499996)
   (28 0.10958345) (29 0.09672784) (30 0.11785113)
   (91/3 0.083333336) (92/3 0.11785113) (31 0.1514267)
   (32 0.13999122) (33 0.16666667) (67/2 1/6)
   (135/4 0.16666667) (34 0.3333333)))
(setq c-prob 0.1)
(comparable-likelihood-profile<-p
 ontime-state-points-pair datapoints c-beats
 template-likelihood-profile c-prob)
--> T
```

This function is similar to the function comparable-likelihood-profilep, the difference being it calls the function geom-mean-likelihood-of- subset¡- (forwards looking for the empirical distribution) rather than geom-mean-likelihood-of- subset. It takes a pair consisting of an ontime and points that sound at the ontime. Based on an empirical distribution over the argument named datapoints (with a context governed by c-beats), it calculates the geometric mean of the likelihood of these points. This likelihood is then compared with that at the same ontime in the template. If the absolute difference between the likelihoods is less than the threshold c-prob, T is returned, and NIL otherwise. T will also be returned if there are no points.

generate-beat-MNN-spacing<-

Started, last checked	12/10/2010, 12/10/2010
Location	Generating beat MNN spacing backwards
Calls	checklist<-p, choose-one, choose-one-with-beat, geom-mean-likelihood-of-states<-, index-1st-sublist-item>=, my-last, segments-strict, sort-dataset-asc, states2datapoints-by-lookup<-, translate-datapoints-to-last-ontime
Called by	
Comments/see also	Consider subdividing into several functions. See also generate-beat-MNN-spacing->, generate-beat-spacing-forcing<-.

Example:

```
(progn
  (setq
   final-states
   (read-from-file
    (concatenate
     'string
     *MCStylistic-Oct2010-example-files-path*
     "/Racchman-Oct2010 example"
     "/final-states.txt")))
  (setq
   stm<-
   (read-from-file
    (concatenate
     'string
     *MCStylistic-Oct2010-example-files-path*
     "/Racchman-Oct2010 example"
     "/transition-matrix<-.txt")))
  (setq no-ontimes< 111/4)
  (setq
   dataset-all
   (read-from-file
    (concatenate
     'string
```

```
     *MCStylistic-Oct2010-data-path*
     "/Dataset/C-24-3-ed.txt")))
  (setq
   dataset-template
   (subseq dataset-all 0 120))
  (setq
   *rs* #.(CCL::INITIALIZE-RANDOM-STATE 477 10894))
  "Data loaded and *rs* set.")
(generate-beat-MNN-spacing<-
 final-states stm<- no-ontimes< dataset-template)
--> ((((3 NIL)
       (NIL NIL "C-68-4" ((184 65 63 2 0 186 9))))
      ((2 (7 5 5))
       (16 9 "C-17-2"
        ((10 55 57 1 1 11 34) (10 62 61 1 1 11 35)
         (9 67 64 2 0 11 32) (9 72 67 2 0 11 33))))
      ((1 (24 5))
       (12 7 "C-17-2"
        ((165 43 50 1 1 166 608)
         (165 67 64 2 0 167 609)
         (165 72 67 2 0 167 610))))
      ((3 (7 3 6 6))
       (-7 -4 "C-33-2"
        ((212 52 55 1 1 213 956)
         (212 59 59 1 1 213 957)
         (212 62 61 1 0 213 958)
         (211 68 64 2 0 213 955)
         (212 74 68 1 0 213 959))))
      ((2 (7 3 6 6))
       (0 0 "C-33-2"
        ((235 52 55 1 1 236 1057)
         (235 59 59 1 1 236 1058)
         (235 62 61 1 0 236 1059)
         (235 68 64 2 0 237 1060)
         (704/3 74 68 4/3 0 236 1056))))
      ((1 (12 12))
       (24 14 "C-17-1"
        ((204 36 46 1 1 205 920)
         (204 48 53 1 1 205 921)
         (204 60 60 1 0 205 922))))
      ((3 (5 4 3 5 4))
```

```
  (-19 -11 "C-50-3"
   ((101 52 55 1 1 102 289)
    (101 57 58 1 1 102 290)
    (101 61 60 1 0 102 291)
    (101 64 62 1 1 102 292)
    (101 69 65 1 0 102 293)
    (101 73 67 1 0 102 294)))))
((26 18 36 2 1) (26 23 39 2 1) (26 27 41 2 0)
 (26 30 43 2 1) (26 35 46 2 0) (26 39 48 2 0)
 (28 -1 25 1 1) (28 11 32 1 1) (28 23 39 1 0)
 (29 23 39 1 1) (29 30 43 1 1) (29 33 45 1 0)
 (29 39 48 3 0) (29 45 52 3 0) (30 23 39 2 1)
 (30 30 43 2 1) (30 33 45 2 0) (32 16 35 1 1)
 (32 40 49 2 0) (32 45 52 2 0) (33 28 42 1 1)
 (33 35 46 1 1) (34 44 51 2 0))
8 (0 0 0 1 0 0 0))
```

This function is similar to the function generate-beat-MNN-spacing-¿, the difference is that the former generates a passage backwards one state at a time. The checking process is analogous. Given final states, a state-transition matrix, a lower limit for ontime, and a template dataset, this function generates datapoints (among other output) that conform to various criteria, which can be specified using the optional arguments. The criteria are things like: not too many consecutive states from the same source (c- sources), the range is comparable with that of the template (c-bar, c-min, and c-max), and the likelihood of the states is comparable with that of the template (c-beats and c-prob). A record, named index-failures, is kept of how many times a generated state fails to meet the criteria. When a threshold c-failures is exceeded at any state index, the penultimate state is removed and generation continues. If the value of c- failures is exceeded for the first state, a message is returned.

This function returns the states as well as the datapoints, and also index-failures and i, the total number of iterations.

generate-beat-spacing-forcing<-

Started, last checked	12/10/2010, 12/10/2010
Location	Generating beat MNN spacing backwards
Calls	beat-spacing-states, checklist<-p, choose-one, choose-one-with-beat, geom-mean-likelihood-of-states<-, index-1st-sublist-item>=, my-last, nth-list-of-lists, segments-strict, sort-dataset-asc, states2datapoints-by-lookup<-, translate-datapoints-to-last-ontime
Called by	
Comments/see also	Consider subdividing into several functions. See also generate-beat-MNN-spacing<-, generate-beat-spacing-forcing->.

Example:

```
(progn
  (setq
   final-states
   (read-from-file
    (concatenate
     'string
     *MCStylistic-Oct2010-example-files-path*
     "/Racchman-Oct2010 example"
     "/internal-final-states.txt")))
  (setq
   stm<-
   (read-from-file
    (concatenate
     'string
     *MCStylistic-Oct2010-example-files-path*
     "/Racchman-Oct2010 example"
     "/transition-matrix<-.txt")))
  (setq no-ontimes< 2)
  (setq
   dataset-all
   (read-from-file
    "/Users/tec69/Open/Music/Datasets/C-24-3-ed.txt"))
```

```
  (setq
   dataset-template
   (subseq dataset-all 0 120))
  (setq
   *rs* #.(CCL::INITIALIZE-RANDOM-STATE 48164 60796))
  (setq
   datapoints-from-next-interval
   '((5/2 63 62 1/2 0) (5/2 72 67 1/2 0)
     (3 34 45 1 1) (3 46 52 1 1) (3 65 63 1/2 0)
     (3 74 68 1/2 0) (15/4 65 63 1/4 0)))
  (setq
   next-state-context-pair
   (first
    (beat-spacing-states
     datapoints-from-next-interval "No information"
     3 1 3)))
  (setq generation-interval '(1 3))
  (setq
   pattern-region
   '((1 51 55) (1 55 57) (1 61 61) (1 63 62) (2 51 55)
     (2 55 57) (2 61 61) (5/2 73 68)))
  "Data loaded and variables set.")
(generate-beat-spacing-forcing<-
 final-states stm<- no-ontimes< dataset-template
 generation-interval pattern-region
 next-state-context-pair (list "originalp")
 3 10 4 19 12 12 12 .15)
--> ((((7/2 (9))
       (NIL NIL "No information"
        ((5/2 63 62 1/2 0 3 0)
         (5/2 72 67 1/2 0 3 1))))
      ((3 (9))
       (-2 -1 "C-6-3"
        ((101 71 66 1/2 0 203/2 424)
         (101 80 71 1/2 0 203/2 425)))))
     ((2 65 63 1/2 0) (2 74 68 1/2 0)
      (5/2 63 62 1/2 0) (5/2 72 67 1/2 0))
     2 (0 0))
```

This function appears to be very simiar to the function generate-beat-MNN-spacing<-. The difference is that there are some extra arguments here, which allow for using either external or internal initial/final states, and for using information from a discovered pattern or previous/next state to further guide the generation, hence ‘forcing’.

geom-mean-likelihood-of-states<-

Started, last checked	12/10/2010, 12/10/2010
Location	Generating beat MNN spacing backwards
Calls	geom-mean-likelihood-of-subset<-, max-item, min-item, nth-list-of-lists, orthogonal-projection-not-unique-equalp
Called by	generate-beat-MNN-spacing<-, generate-beat-spacing-forcing<-
Comments/see also	

Example:

```
(setq
 ontime-state-points-pairs
 '((27
    ((27 42 49 1 1 28 0) (27 70 65 1/2 0 55/2 1)))
   (55/2 ((27 42 49 1 1 28 0)))
   (111/4
    ((27 42 49 1 1 28 0) (111/4 67 64 1/4 0 28 2)
     (111/4 79 71 1/4 0 28 3)))
   (28
    ((28 54 56 1 1 29 4) (28 58 58 1 1 29 5)
     (28 64 62 1 1 29 6) (28 66 63 2 0 30 7)
     (28 78 70 2 0 30 8)))
   (29
    ((29 54 56 1 1 30 9) (29 58 58 1 1 30 10)
     (29 64 62 1 1 30 11) (28 66 63 2 0 30 7)
     (28 78 70 2 0 30 8)))
   (30
    ((30 47 52 1 1 31 12) (30 74 68 1/3 0 91/3 13)))
   (91/3
    ((30 47 52 1 1 31 12) (91/3 76 69 1/3 0 92/3 14)))
   (92/3
    ((30 47 52 1 1 31 12) (92/3 74 68 1/3 0 31 15)))
   (31
    ((31 54 56 1 1 32 16) (31 62 61 1 1 32 17)
```

```
    (31 73 67 1 0 32 18)))
   (32
    ((32 54 56 1 1 33 19) (32 62 61 1 1 33 20)
     (32 71 66 1 0 33 21)))
   (33
    ((33 42 49 1 1 34 22) (33 69 65 1/2 0 67/2 23)))
   (67/2 ((33 42 49 1 1 34 22)))
   (135/4
    ((33 42 49 1 1 34 22) (135/4 67 64 1/4 0 34 24)))
   (34
    ((34 54 56 1 1 35 25) (34 57 58 1 1 35 26)
     (34 61 60 1 1 35 27))) (35 NIL)))
(setq
 dataset
 '((27 42 49 1 1) (27 70 65 1/2 0) (111/4 67 64 1/4 0)
   (111/4 79 71 1/4 0) (28 54 56 1 1) (28 58 58 1 1)
   (28 64 62 1 1) (28 66 63 2 0) (28 78 70 2 0)
   (29 54 56 1 1) (29 58 58 1 1) (29 64 62 1 1)
   (30 47 52 1 1) (30 74 68 1/3 0) (91/3 76 69 1/3 0)
   (92/3 74 68 1/3 0) (31 54 56 1 1) (31 62 61 1 1)
   (31 73 67 1 0) (32 54 56 1 1) (32 62 61 1 1)
   (32 71 66 1 0) (33 42 49 1 1) (33 69 65 1/2 0)
   (135/4 67 64 1/4 0) (34 54 56 1 1) (34 57 58 1 1)
   (34 61 60 1 1)))
(geom-mean-likelihood-of-states<-
 ontime-state-points-pairs dataset 3)
--> ((27 0.07142857) (55/2 1/14) (111/4 0.062499996)
     (28 0.10958345) (29 0.09672784) (30 0.11785113)
     (91/3 0.083333336) (92/3 0.11785113)
     (31 0.1514267) (32 0.13999122) (33 0.16666667)
     (67/2 1/6) (135/4 0.16666667) (34 0.3333333))
```

Applies the function geom-mean-likelihood-of-subset<- recursively to the first argument.

geom-mean-likelihood-of-subset<-

Started, last checked	12/10/2010, 12/10/2010
Location	Generating beat MNN spacing backwards
Calls	empirical-mass, index-1st-sublist-item>=, index-1st-sublist-item>, likelihood-of-subset
Called by	comparable-likelihood-profilep, geom-mean-likelihood-of-states
Comments/see also	

Example:

```
(setq
 subset
 '((27 42 49 1 1 28 0) (111/4 67 64 1/4 0 28 2)
   (111/4 79 71 1/4 0 28 3)))
(setq
 subset-palette
 (orthogonal-projection-not-unique-equalp
  subset '(0 1)))
(setq first-subset-ontime 27)
(setq last-subset-ontime 111/4)
(setq
 dataset
 '((27 42 49 1 1) (27 70 65 1/2 0) (111/4 67 64 1/4 0)
   (111/4 79 71 1/4 0) (28 54 56 1 1) (28 58 58 1 1)
   (28 64 62 1 1) (28 66 63 2 0) (28 78 70 2 0)
   (29 54 56 1 1) (29 58 58 1 1) (29 64 62 1 1)
   (30 47 52 1 1) (30 74 68 1/3 0) (91/3 76 69 1/3 0)
   (92/3 74 68 1/3 0) (31 54 56 1 1) (31 62 61 1 1)
   (31 73 67 1 0) (32 54 56 1 1) (32 62 61 1 1)
   (32 71 66 1 0) (33 42 49 1 1) (33 69 65 1/2 0)
   (135/4 67 64 1/4 0) (34 54 56 1 1) (34 57 58 1 1)
   (34 61 60 1 1)))
(setq
 dataset-palette
 (orthogonal-projection-not-unique-equalp
  dataset '(0 1)))
(setq
 ontimes-list (nth-list-of-lists 0 dataset))
(setq c-beats 4)
```

```
(geom-mean-likelihood-of-subset<-
 subset subset-palette first-subset-ontime
 last-subset-ontime dataset-palette ontimes-list
 c-beats)
--> 0.052631576
```

This function is similar to the function geom-mean-likelihood-of-subset, the difference being we look forward to form the empirical distribution, rather than backward. The first argument to this function, called subset, is a point set. Both in the scenario of likelihood calculation for an original excerpt and for a generated passage, the point set is a segment of the music. The argument subset-palette consists of a (listed) list of MIDI note numbers from the subset. Note: first-subset-ontime is not necessarily the ontime of the first datapoint, as they will have been sorted by MIDI note number. The variable dataset-palette is analogous, ontimes-list is a list of ontimes from the dataset. The threshold c-beats determines how far forward we look to form the empirical distribution. The output of this function is the geometric mean of the likelihood of the subset (that is, a product of the individual empirical probabilities of the constituent MIDI note numbers.

translate-datapoints-to-last-offtime

Started, last checked	12/10/2010, 12/10/2010
Location	Generating beat MNN spacing backwards
Calls	constant-vector, translation
Called by	
Comments/see also	Deprecated.

Example:

```
(translate-datapoints-to-last-offtime
 30
 '((0 44 51 1 1) (0 48 53 1 1) (0 56 58 1 0)
   (2 44 51 1 1) (3 48 53 5 1) (6 56 58 1 0)))
--> ((22 44 51 1 1) (22 48 53 1 1) (22 56 58 1 0)
     (24 44 51 1 1) (25 48 53 5 1) (28 56 58 1 0))
```

This function takes two arguments: a time t and a list of datapoints. It calculates the maximum offtime of the datapoints, and translates the datapoints, such that the maximum offtime is now t.

translate-datapoints-to-last-ontime

Started, last checked	12/10/2010, 12/10/2010
Location	Generating beat MNN spacing backwards
Calls	constant-vector, my-last, translation
Called by	generate-beat-MNN-spacing->, generate-beat-spacing-forcing->
Comments/see also	translate-datapoints-to-first-ontime

Example:

```
(translate-datapoints-to-last-ontime
 34 0
 '((0 44 51 1 1) (0 48 53 1 1) (0 56 58 1 0)
   (2 44 51 1 1) (3 48 53 1 1) (6 56 58 1 0)))
--> ((28 44 51 1 1) (28 48 53 1 1) (28 56 58 1 0)
     (30 44 51 1 1) (31 48 53 1 1) (34 56 58 1 0)).
```

This function takes three arguments: an ontime, an ontime index and a list of datapoints (assumed to be sorted in lexicographical order). It translates these datapoints such that the ontime of the last datapoint equals the first argument.

4.6.10 Generating beat MNN spacing for&back

The main function here is called generate-beat-MNN-spacing<->. Given initial and final states, forwards- and backwards-running state-transition matrices, and a template dataset, it generates datapoints (among other output) that conform to various criteria, which can be specified using the optional arguments. The idea is to join forwards- and backwards-generated phrases, choosing whichever pair leads to a phrase whose mean deviation from the template likelihood profile is minimal.

The criteria are things like: not too many consecutive states from the same source, the range is comparable with that of the template, and the likelihood of the states is comparable with that of the template.

generate-beat-MNN-spacing<->

Started, last checked	13/10/2010, 13/10/2010
Location	Generating beat MNN spacing for&back
Calls	generate-forwards-or-backwards-no-failure, my-last, segments-strict, unite-datapoints
Called by	
Comments/see also	Consider simplifying/renaming generating functions.

Example: see Stylistic composition with Racchman-Oct2010 (Sec. 3.4, especially lines 498-502).

This function unites several forwards- and backwards running realisations of Markov models built on the arguments initial-states, stm->, final-states, and stm<-. It is constrained by a template (the argument dataset-template) and various parameters: like not too many consecutive states from the same source (c-sources), the range is comparable with that of the template (c-bar, c-min, and c-max), and the likelihood of the states is comparable with that of the template (c-beats and c-prob).

The numbers of forwards- and backwards- realisations generated are determined by the arguments c-forwards and c-backwards respectively. The output is a list of three hash tables (one containing the forwards candidates, one the backwards candidates, and one the united candidates). If c-forwards $= m$ and c-backwards $= n$, then the number of united candidates is $3mn$.

generate-beat-spacing-forced<->

Started, last checked	13/10/2010, 13/10/2010
Location	Generating beat MNN spacing for&back
Calls	generate-forced<->no-failure, generate-forwards-or-backwards-no-failure, segments-strict, unite-datapoints
Called by	
Comments/see also	Consider simplifying/renaming generating functions.

Example:

```
(progn
  (setq generation-interval '(12 24))
```

```
(setq terminal->p nil)
(setq terminal<-p nil)
(setq
 external-initial-states
 (read-from-file
  (concatenate
   'string
   *MCStylistic-Oct2010-example-files-path*
   "/Racchmaninof-Oct2010 example"
   "/initial-states.txt")))
(setq
 internal-initial-states
 (read-from-file
  (concatenate
   'string
   *MCStylistic-Oct2010-example-files-path*
   "/Racchmaninof-Oct2010 example"
   "/internal-initial-states.txt")))
(setq
 stm->
 (read-from-file
  (concatenate
   'string
   *MCStylistic-Oct2010-example-files-path*
   "/Racchmaninof-Oct2010 example"
   "/transition-matrix.txt")))
(setq
 external-final-states
 (read-from-file
  (concatenate
   'string
   *MCStylistic-Oct2010-example-files-path*
   "/Racchmaninof-Oct2010 example"
   "/final-states.txt")))
(setq
 internal-final-states
 (read-from-file
  (concatenate
   'string
   *MCStylistic-Oct2010-example-files-path*
   "/Racchmaninof-Oct2010 example"
```

```
    "/internal-final-states.txt")))
  (setq
   stm<-
   (read-from-file
    (concatenate
     'string
     *MCStylistic-Oct2010-example-files-path*
     "/Racchmaninof-Oct2010 example"
     "/transition-matrix<-.txt")))
  (setq
   dataset-all
   (read-from-file
    "/Users/tec69/Open/Music/Datasets/C-56-1-ed.txt"))
  (setq
   dataset-template
   (subseq dataset-all 0 132))
  (setq
   pattern-region
   '((12 60 60) (12 64 62) (25/2 57 58) (51/4 64 62)
     (13 52 55) (13 64 62) (14 54 56) (29/2 62 61)
     (15 55 57) (15 59 59) (63/4 64 62) (16 43 50)
     (16 50 54) (16 66 63) (17 67 64) (18 57 58)
     (18 60 60) (18 64 62) (75/4 66 63) (19 43 50)
     (19 50 54) (19 67 64) (20 66 63) (20 69 65)
     (21 59 59) (21 67 64) (21 71 66) (87/4 74 68)
     (22 43 50) (22 50 54) (22 74 68) (23 62 61)
     (24 60 60) (24 64 62)))
  (setq state-context-pair-> nil)
  (setq state-context-pair<- nil)
  (setq
   checklist
   (list "originalp" "mean&rangep" "likelihoodp"))
  (setq
   *rs* #.(CCL::INITIALIZE-RANDOM-STATE 6086 61144))
  (setq time-a (get-internal-real-time))
  (setq
   output
   (generate-beat-spacing-forced<->
    generation-interval terminal->p terminal<-p
    external-initial-states internal-initial-states
    stm-> external-final-states internal-final-states
```

```
    stm<- dataset-template pattern-region
    state-context-pair-> state-context-pair<-
    checklist 3 10 4 19 12 12 12 0.2 3 3))
  (setq time-b (get-internal-real-time))
  (float
   (/
    (- time-b time-a)
    internal-time-units-per-second)))
--> 15.091431 seconds.
(setq
 output-datapoints
 (gethash
  '"united,2,3,backwards-dominant" (third output)))
(saveit
 (concatenate
  'string
  *MCStylistic-Oct2010-example-files-path*
  "/united,2,3,backwards-dominant.mid")
 (modify-to-check-dataset
  (translation
   output-datapoints
   (list
    (- 0 (first (first output-datapoints)))
    0 0 0 0)) 900))
```

This function is similar to the function generate-beat-MNN-spacing<->. The difference is that there are some extra arguments here, which allow for using either external or internal initial/final states, and for using information from a discovered pattern or previous/next state to further guide the generation.

The function unites several forwards- and backwards running realisations of Markov models built on the arguments initial-states, stm-¿, final-states, and stm¡-. It is constrained by a template (the argument dataset-template) and various parameters: like not too many consecutive states from the same source (c-sources), the range is comparable with that of the template (c-bar, c-min, and c-max), and the likelihood of the states is comparable with that of the template (c-beats and c-prob).

The numbers of forwards- and backwards- realisations generated are determined by the arguments c-forwards and c-backwards respectively. The output is a list of three hash tables (one containing the forwards candidates, one the backwards candidates, and one the united candidates). If c-forwards $= m$ and c-backwards $= n$, then the number of united candidates is $3mn$.

generate-forced<->no-failure

Started, last checked	13/10/2010, 13/10/2010
Location	Generating beat MNN spacing for&back
Calls	generate-beat-spacing-forcing->, generate-beat-spacing-forcing<-, segments-strict
Called by	generate-beat-spacing-forced<->
Comments/see also	Consider simplifying/renaming generating functions.

Example:

```
(progn
  (setq
   internal-states->
   (read-from-file
    (concatenate
     'string
     *MCStylistic-Oct2010-example-files-path*
     "/Racchmaninof-Oct2010 example"
     "/internal-initial-states.txt")))
  (setq
   internal-states<-
   (read-from-file
    (concatenate
     'string
     *MCStylistic-Oct2010-example-files-path*
     "/Racchmaninof-Oct2010 example"
     "/internal-final-states.txt")))
  (setq
   stm->
   (read-from-file
    (concatenate
     'string
     *MCStylistic-Oct2010-example-files-path*
     "/Racchmaninof-Oct2010 example"
     "/transition-matrix.txt")))
  (setq
   stm<-
   (read-from-file
```

```
    (concatenate
     'string
     *MCStylistic-Oct2010-example-files-path*
     "/Racchmaninof-Oct2010 example"
     "/transition-matrix<-.txt")))
  (setq no-ontimes> 29)
  (setq
   dataset-all
   (read-from-file
    "/Users/tec69/Open/Music/Datasets/C-17-1-ed.txt"))
  (setq
   dataset-template
   (subseq dataset-all 0 250))
  (setq generation-interval '(25 29))
  (setq
   pattern-region
   '((25 60 60) (25 67 64) (25 70 66) (25 76 69)
     (25 82 73) (53/2 60 60) (53/2 67 64) (53/2 70 66)
     (53/2 76 69) (53/2 81 72) (27 60 60) (27 67 64)
     (27 70 66) (27 76 69) (27 78 70) (111/4 60 60)
     (111/4 67 64) (111/4 70 66) (111/4 76 69)
     (111/4 79 71) (28 60 60) (28 67 64) (28 70 66)
     (28 76 69) (28 81 72) (29 60 60) (29 67 64)
     (29 70 66) (29 76 69) (29 79 71)))
  (setq
   checklist
   (list "originalp" "mean&rangep" "likelihoodp"))
  (setq
   *rs* #.(CCL::INITIALIZE-RANDOM-STATE 6086 61144))
  (setq
   output
   (generate-forced<->no-failure
    "forwards" nil nil internal-states-> stm->
    no-ontimes> dataset-template generation-interval
    pattern-region nil checklist 3 10 4 19 12 12 12
    0.2))
  (first output))
--> 0.335408 seconds
(if (listp (fifth output))
  (saveit
   (concatenate
```

```
    'string
    *MCStylistic-Oct2010-example-files-path*
    "/test.mid")
   (modify-to-check-dataset
    (translation
     (fifth output)
     (list
      (- 0 (first (first (fifth output))))
      0 0 0 0)) 900)))
```

This function is similar to the function generate-forwards-or-backwards-no-failure. The difference is that this can take a pattern or a previous or next state as extra constraints. This is necessary when generating a passage according to a template. It generates states (and realisations of those states), taking initial (or final) states and a forwards- (or backwards-) running stm as arguments. The direction must be specified as the first argument, so that the appropriate generating function is called. Depending on the values of the parameters, a call to a generating function can fail to produce a generated passage, in which case this function runs again, until a passage has been generated, hence 'no failure'.

generate-forwards-or-backwards-no-failure

Started, last checked	13/10/2010, 13/10/2010
Location	Generating beat MNN spacing for&back
Calls	generate-beat-MNN-spacing->, generate-beat-MNN-spacing<-, segments-strict
Called by	generate-beat-MNN-spacing<->, generate-beat-spacing-forced<->
Comments/see also	Consider simplifying/renaming generating functions.

Example:

```
(progn
  (setq
   initial-states
   (read-from-file
    (concatenate
     'string
```

```
    *MCStylistic-Oct2010-example-files-path*
    "/Racchmaninof-Oct2010 example"
    "/initial-states.txt")))
 (setq
  stm
  (read-from-file
   (concatenate
    'string
    *MCStylistic-Oct2010-example-files-path*
    "/Racchmaninof-Oct2010 example"
    "/transition-matrix.txt")))
 (setq no-ontimes> 24)
 (setq
  dataset-all
  (read-from-file
   "/Users/tec69/Open/Music/Datasets/C-59-3-ed.txt"))
 (setq
  dataset-template (subseq dataset-all 48 184))
 (setq
  checklist
  (list "originalp" "mean&rangep" "likelihoodp"))
 (setq
  *rs* (CCL::INITIALIZE-RANDOM-STATE 2249 23752))
 (setq
  output
  (generate-forwards-or-backwards-no-failure
   "forwards" initial-states stm no-ontimes>
   dataset-template checklist 3 10 4 19 12 12 12
   0.2))
 (first output))
--> 0.048711 seconds.
(if (listp (fifth output))
 (saveit
  (concatenate
   'string
   *MCStylistic-Oct2010-example-files-path*
   "/test.mid")
  (modify-to-check-dataset
   (translation
    (fifth output)
    (list
```

```
    (- 0 (first (first (fifth output))))
    0 0 0 0)) 900)))
```

This function generates states (and realisations of those states), taking initial (or final) states and a forwards- (or backwards-) running stm as arguments. The direction must be specified as the first argument, so that the appropriate generating function is called. Depending on the values of the parameters, a call to a generating function can fail to produce a generated passage, in which case this function runs again, until a passage has been generated, hence 'no failure'.

keys-of-states-in-transition-matrix

Started, last checked	13/10/2010, 13/10/2010
Location	Generating beat MNN spacing for&back
Calls	state-in-transition-matrixp
Called by	most-plausible-join
Comments/see also	

Example:

```
(setq A (make-hash-table :test #'equal))
(setf
 (gethash '"cand,1,1,superimpose" A)
 '((3/4 60 60 1 0) (3/4 64 62 1 0) (3/4 67 64 1 0)))
(setf
 (gethash '"cand,1,2,superimpose" A)
 '((0 60 60 1 0) (3/4 64 62 1 0) (2 67 64 1 0)))
(setf
 (gethash '"cand,2,1,superimpose" A)
 '((3/4 60 60 1 0) (3/4 64 62 1 0) (3/4 69 65 1 0)))
(setf
 (gethash '"cand,2,2,superimpose" A)
 '((3/4 60 60 1 0) (3/4 64 62 1 0) (3/4 72 67 1 0)))
(setq
 dataset-keys
 '("cand,1,1,superimpose" "cand,1,2,superimpose"
   "cand,2,1,superimpose" "cand,2,2,superimpose"))
(setq ontime 3/4)
(setq
 stm
```

```
 '(((7/4 (4 5)) "etc") ((7/4 (4 3)) "etc")
   ((7/4 (4 3 17)) "etc")))
(keys-of-states-in-transition-matrix
 A dataset-keys ontime stm "beat-spacing-states" 3 3
 1)
--> ("cand,1,1,superimpose" "cand,2,1,superimpose")
```

This function takes a hash table consisting of datasets and a list of keys for that hash table as its first two arguments. The function state-in- transition-matrixp is applied to the dataset associated with each key, and if it is in the state- transition matrix, this key is included in the returned list. This function can be used to check that the composer actually wrote the chords that are created at the bisection.

min-max-abs-diffs-for-likelihood-profiles

Started, last checked	13/10/2010, 13/10/2010
Location	Generating beat MNN spacing for&back
Calls	abs-differences-for-curves-at-points, geom-mean-likelihood-of-states, max-item, min-argmin, nth-list-of-lists segments-strict
Called by	most-plausible-join
Comments/see also	

Example:

```
(setq A (make-hash-table :test #'equal))
(setf
 (gethash '"cand,1,1,superimpose" A)
 '((0 60 60 3/4 0) (0 64 62 3/4 0) (0 67 64 3/4 0)
   (3/4 60 60 1 0) (3/4 64 62 1 0) (3/4 67 64 1 0)
   (7/4 60 60 1 0) (7/4 64 62 1 0) (7/4 67 64 1 0)))
(setf
 (gethash '"cand,1,2,superimpose" A)
 '((0 60 60 1 0) (3/4 64 62 1 0) (2 67 64 1 0)))
(setf
 (gethash '"cand,2,1,superimpose" A)
 '((0 61 60 3/4 0) (0 65 62 3/4 0) (0 68 64 3/4 0)
   (3/4 60 60 1 0) (3/4 64 62 1 0) (3/4 67 64 1 0)
   (7/4 62 61 1 0) (7/4 66 63 1 0) (7/4 69 65 1 0)))
(setf
```

```
 (gethash '"cand,2,2,superimpose" A)
 '((3/4 60 60 1 0) (3/4 64 62 1 0) (3/4 72 67 1 0)))
(setq
 dataset-keys
 '("cand,1,1,superimpose" "cand,1,2,superimpose"
   "cand,2,1,superimpose" "cand,2,2,superimpose"))
(setq
 template-dataset
 '((0 60 60 3/4 0) (0 64 62 3/4 0) (0 67 64 3/4 0)
   (3/4 59 59 1 0) (3/4 62 61 1 0) (3/4 67 64 1 0)
   (7/4 60 60 1 0) (7/4 64 62 1 0) (7/4 67 64 1 0)))
(setq c-beats 12)
(min-max-abs-diffs-for-likelihood-profiles
 A dataset-keys template-dataset c-beats)
--> "cand,1,1,superimpose"
```

This function takes a hash table consisting of datasets and a list of keys for that hash table as its first two arguments. Its third argument is the dataset for a template. The idea is to compare each of the likelihood profiles for the datasets associated with the keys with the likelihood profile of the template dataset, using the function abs- differences-for-curves-at-points. Each comparison will produce a maximal difference. The key of the dataset that produces the minimum of the maximal differences is returned, and the intuition is that this will be the most plausible dataset, compared with the template.

most-plausible-join

Started, last checked	13/10/2010, 13/10/2010
Location	Generating beat MNN spacing for&back
Calls	disp-ht-key, keys-of-states-in-transition-matrix, min-max-abs-diffs-for-likelihood-profiles
Called by	
Comments/see also	

Example:

```
(setq A (make-hash-table :test #'equal))
(setf
 (gethash '"cand,1,1,superimpose" A)
```

```
 '((0 60 60 3/4 0) (0 64 62 3/4 0) (0 67 64 3/4 0)
   (3/4 60 60 1 0) (3/4 64 62 1 0) (3/4 67 64 1 0)
   (7/4 60 60 1 0) (7/4 64 62 1 0) (7/4 67 64 1 0)))
(setf
 (gethash '"cand,1,2,superimpose" A)
 '((0 60 60 1 0) (3/4 64 62 1 0) (2 67 64 1 0)))
(setf
 (gethash '"cand,2,1,superimpose" A)
 '((0 61 60 3/4 0) (0 65 62 3/4 0) (0 68 64 3/4 0)
   (3/4 60 60 1 0) (3/4 64 62 1 0) (3/4 67 64 1 0)
   (7/4 62 61 1 0) (7/4 66 63 1 0) (7/4 69 65 1 0)))
(setf
 (gethash '"cand,2,2,superimpose" A)
 '((3/4 60 60 1 0) (3/4 64 62 1 0) (3/4 72 67 1 0)))
(setq
 template-dataset
 '((0 60 60 3/4 0) (0 64 62 3/4 0) (0 67 64 3/4 0)
   (3/4 59 59 1 0) (3/4 62 61 1 0) (3/4 67 64 1 0)
   (7/4 60 60 1 0) (7/4 64 62 1 0) (7/4 67 64 1 0)))
(setq
 stm
 '(((7/4 (4 5)) "etc") ((1 (4 3)) "etc")
   ((11/4 (4 3 17)) "etc")))
(setq c-beats 12)
(most-plausible-join A 3/4 template-dataset stm 3 3 1)
--> "cand,1,1,superimpose"
```

This function applies the function keys-of- states-in-transition-matrix, followed by the function min-max-abs-diffs-for-likelihood-profiles, to determine which dataset, out of several candidates, is the most plausible fit with the template dataset.

remove-coincident-datapoints

Started, last checked	13/10/2010, 13/10/2010
Location	Generating beat MNN spacing for&back
Calls	remove-datapoints-coincident-with-datapoint
Called by	unite-datapoints
Comments/see also	

Example:

```
(remove-coincident-datapoints
 '((12 64 61 1) (14 63 62 1) (31/2 65 63 1/2))
 '((9 60 60 1) (10 64 62 3) (13 63 62 1)
   (13 72 67 5) (15 65 63 1) (16 65 63 2)) 1 3)
--> ((9 60 60 1) (13 63 62 1) (13 72 67 5)
     (16 65 63 2))
```

This function removes any datapoints (second argument) that sound at the same time as datapoints provided as the first argument.

remove-datapoints-coincident-with-datapoint

Started, last checked	13/10/2010, 13/10/2010
Location	Generating beat MNN spacing for&back
Calls	append-offtimes, my-last
Called by	remove-coincident-datapoints
Comments/see also	

Example:

```
(remove-datapoints-coincident-with-datapoint
 '(12 64 61 1)
 '((9 60 60 1) (10 64 62 3) (13 63 62 1)
   (13 72 67 5) (15 65 63 1) (16 65 63 2)) 1 3)
--> ((9 60 60 1) (13 63 62 1) (13 72 67 5)
     (15 65 63 1) (16 65 63 2))
```

This function removes any datapoints (second argument) that sound at the same time as a datapoint provided as the first argument.

remove-datapoints-with-nth-item<

Started, last checked	13/10/2010, 13/10/2010
Location	Generating beat MNN spacing for&back
Calls	index-1st-sublist-item>=, nth-list-of-lists
Called by	unite-datapoints
Comments/see also	

Example:

```
(remove-datapoints-with-nth-item<
 '((9 60) (10 64) (13 63) (13 72) (15 65) (16 65)) 10
 0)
--> ((10 64) (13 63) (13 72) (15 65) (16 65))
```

This function removes any datapoints whose nth-items are less than the second argument. Datapoints are assumed to be in lexicographic order.

remove-datapoints-with-nth-item<=

Started, last checked	13/10/2010, 13/10/2010
Location	Generating beat MNN spacing for&back
Calls	index-1st-sublist-item>, nth-list-of-lists
Called by	unite-datapoints
Comments/see also	

Example:

```
(remove-datapoints-with-nth-item<=
 '((9 60) (10 64) (13 63) (13 72) (15 65) (16 65)) 10
 0)
--> ((13 63) (13 72) (15 65) (16 65))
```

This function removes any datapoints whose nth-items are less than or equal to the second argument. Datapoints are assumed to be in lexicographic order.

remove-datapoints-with-nth-item>

Started, last checked	13/10/2010, 13/10/2010
Location	Generating beat MNN spacing for&back
Calls	index-1st-sublist-item>, nth-list-of-lists
Called by	unite-datapoints
Comments/see also	

Example:

```
(remove-datapoints-with-nth-item>
 '((9 60) (10 64) (13 63) (13 72) (15 65) (16 65)) 15
 0)
--> ((9 60) (10 64) (13 63) (13 72) (15 65))
```

This function removes any datapoints whose nth-items are greater than the second argument. Datapoints are assumed to be in lexicographic order.

remove-datapoints-with-nth-item>=

Started, last checked	13/10/2010, 13/10/2010
Location	Generating beat MNN spacing for&back
Calls	index-1st-sublist-item>=, nth-list-of-lists
Called by	unite-datapoints
Comments/see also	

Example:

```
(remove-datapoints-with-nth-item>=
 '((9 60) (10 64) (13 63) (13 72) (15 65) (16 65)) 15
 0)
--> ((9 60) (10 64) (13 63) (13 72))
```

This function removes any datapoints whose nth-items are greater than or equal to the second argument. Datapoints are assumed to be in lexicographic order.

state-in-transition-matrixp

Started, last checked	13/10/2010, 13/10/2010
Location	Generating beat MNN spacing for&back
Calls	beat-spacing-states, nth-list-of-lists, segments-strict, spacing-holding-states
Called by	keys-of-states-in-transition-matrix
Comments/see also	

Example:

```
(state-in-transition-matrixp
 '((0 60 60 1 0) (0 64 62 1 0) (0 67 64 1 0))
 0
 '(((7/4 (4 5)) "etc") ((1 (4 3)) "etc")
   ((11/4 (4 3 17)) "etc"))
 "beat-spacing-states" 3 3 1)
--> T
```

This function checks a state, which exists at a specified ontime in a given dataset, for membership in a state-transition matrix. If it is a member, T is returned, and NIL otherwise.

unite-datapoints

Started, last checked	13/10/2010, 13/10/2010
Location	Generating beat MNN spacing for&back
Calls	remove-coincident-datapoints, remove-datapoints-with-nth-item<, remove-datapoints-with-nth-item<=, remove-datapoints-with-nth-item>=, remove-datapoints-with-nth-item>, sort-dataset-asc
Called by	generate-beat-MNN-spacing<->, generate-beat-spacing-forced<->
Comments/see also	

Example:

```
(unite-datapoints
 '((9 60 60 1) (10 64 62 2) (13 63 62 1) (14 60 60 2)
   (15 65 63 2))
 '((27/2 60 60 1/2) (14 60 60 1) (14 63 62 1)
   (31/2 65 63 1/2) (16 64 62 1) (17 59 59 1))
 14 "superimpose")
--> ((9 60 60 1) (10 64 62 2) (13 63 62 1)
     (14 60 60 2) (14 63 62 1) (31/2 65 63 1/2)
     (16 64 62 1) (17 59 59 1))
```

This function unites two sets of datapoints. The third argument, join-at, is the ontime at which they are united (specified to avoid overhanging notes from each set sounding during the other), and the fourth argument, join-by, gives the option of superimposing the sets, or letting the first or second set take precedence.

4.6.11 Generating with patterns preliminaries

Most of these functions process lists that represent interval endpoints (interval in the mathematical sense). Others scan hash tables (typically a patterns hash) for instance in order to find the indices of elements with certain properties. There are also some custom merge-sort functions.

car<

Started, last checked	22/10/2010, 22/10/2010
Location	Generating with patterns preliminaries
Calls	
Called by	merge-sort-by-car<
Comments/see also	Consider changing location.

Example:

```
(car< '(1 "sprout") '(1.1 "purple"))
--> T
```

This function returns T if the first element of the first argument is less than the first element of the second argument, and NIL otherwise.

car>

Started, last checked	22/10/2010, 22/10/2010
Location	Generating with patterns preliminaries
Calls	
Called by	merge-sort-by-car>
Comments/see also	Consider changing location.

Example:

```
(car> '(1 "sprout") '(1.1 "purple"))
--> NIL
```

This function returns T if the first element of the first argument is greater than the first element of the second argument, and NIL otherwise.

generate-intervals

Started, last checked	22/10/2010, 22/10/2010
Location	Generating with patterns preliminaries
Calls	interval-intersectionsp, interval-subsetsp
Called by	
Comments/see also	

Example:

```
(setq existing-intervals '((11 17) (20 26) (26 30)))
(setq floor-ontime 1)
(setq ceiling-ontime 40)
(generate-intervals
 floor-ontime ceiling-ontime existing-intervals)
--> ((1 11) (17 20) (30 40))
```

If $L = \{[a_1, b_1), [a_2, b_2), \ldots, [a_l, b_l)\}$ is a list of non-overlapping intervals, with each interval a subset of $[a, b)$, then this function returns endpoints of the non-overlapping intervals $M = \{[c_1, d_1), [c_2, d_2), \ldots, [c_m, d_m)\}$ such that $L \cup M = [a, b)$.

indices-of-max-subset-score

Started, last checked	22/10/2010, 22/10/2010
Location	Generating with patterns preliminaries
Calls	merge-sort-by-car>, merge-sort-by-vector<vector-car, my-last, unaddressed-patterns-subset-scores
Called by	
Comments/see also	

Example:

```
(setq
 patterns-hash
 (read-from-file-balanced-hash-table
  (concatenate
   'string
   *MCStylistic-Oct2010-example-files-path*
   "/patterns-hash.txt")))
(indices-of-max-subset-score patterns-hash)
--> (3 4)
```

This function takes a patterns-hash (a list of discovered translational equivalence classes, each with corresponding attributes) and returns the indices of the pattern that has the highest subset score. The first index is for the TEC, the second is for the occurrence.

interval-intersectionp

Started, last checked	22/10/2010, 22/10/2010
Location	Generating with patterns preliminaries
Calls	
Called by	interval-intersectionsp
Comments/see also	

Example:

```
(interval-intersectionp '(7 9) '(3 7.1))
--> T.
```

This function returns T if its two arguments, endpoints of the intervals $X = [a, b]$ and $Y = [c, d]$, are such that $X \cap Y \neq \emptyset$, and NIL otherwise.

interval-intersectionsp

Started, last checked	22/10/2010, 22/10/2010
Location	Generating with patterns preliminaries
Calls	interval-intersectionp
Called by	generate-intervals
Comments/see also	

Example:

```
(interval-intersectionsp
 '(7 9) '((1 2) (2.2 2.4) (2 2.5) (3 7.1) (1 1)))
--> 3
```

This function has two arguments: the endpoints of a single interval $X = [a, b]$, and the endpoints of a list of intervals $L = \{[a_1, b_1], [a_2, b_2], \ldots, [a_l, b_l]\}$. The function returns the minimum value of i such that $[a, b] \cap [a_i, b_i] \neq \emptyset$, or NIL if no such i exists.

interval-subsetp

Started, last checked	22/10/2010, 22/10/2010
Location	Generating with patterns preliminaries
Calls	
Called by	interval-subsetsp
Comments/see also	

Example:

```
(interval-subsetp '(7 9) '(7 10))
--> T
```

This function returns T if its two arguments, endpoints of the intervals $X = [a, b]$ and $Y = [c, d]$, are such that $X \subseteq Y$, and NIL otherwise.

interval-subsetsp

Started, last checked	22/10/2010, 22/10/2010
Location	Generating with patterns preliminaries
Calls	interval-subsetp
Called by	generate-intervals
Comments/see also	

Example:

```
(interval-subsetsp
 '(7 9) '((1 2) (2.2 2.4) (2 8.5) (4 9.1) (1 1)))
--> T
```

This function has two arguments: the endpoints of a single interval $X = [a, b]$, and the endpoints of a list of intervals $L = \{[a_1, b_1], [a_2, b_2], \ldots, [a_l, b_l]\}$. The function returns T if there exists $1 \leq i \leq l$ such that $[a, b] \subseteq [a_i, b_i]$, or NIL if no such i exists.

merge-sort-by-car<

Started, last checked	22/10/2010, 22/10/2010
Location	Generating with patterns preliminaries
Calls	car<
Called by	
Comments/see also	Consider changing location.

Example:

```
(merge-sort-by-car<
 '((2 "b") (6 "j") (0 "a") (3 "i") (6 "h")))
--> ((0 "a") (2 "b") (3 "i") (6 "j") (6 "h"))
```

This function performs an ascending merge sort on a list of lists, using the first element of each list to determine position.

merge-sort-by-car>

Started, last checked	22/10/2010, 22/10/2010
Location	Generating with patterns preliminaries
Calls	car>
Called by	indices-of-max-subset-score
Comments/see also	Consider changing location.

Example:

```
(merge-sort-by-car>
 '((2 "b") (6 "j") (0 "a") (3 "i") (6 "h")))
--> ((6 "j") (6 "h") (3 "i") (2 "b") (0 "a"))
```

This function performs a descending merge sort on a list of lists, using the first element of each list to determine position.

merge-sort-by-vector<vector-car

Started, last checked	22/10/2010, 22/10/2010
Location	Generating with patterns preliminaries
Calls	vector<vector-car
Called by	indices-of-max-subset-score
Comments/see also	Consider changing location. See also vector<vector.

Example:

```
(merge-sort-by-vector<vector-car
 '(((1 8.968646) (0 0)) ((0 8.957496) (1 0))
   ((0 8.167285) (2 0)) ((2 3.8855853 (3 4)))))
--> (((0 8.167285) (2 0)) ((0 8.957496) (1 0))
     ((1 8.968646) (0 0)) ((2 3.8855853 (3 4))))
```

This function performs an ascending merge sort on a list of lists, using the lexicographic order of first elements to determine position.

unaddressed-patterns-subset-scores

Started, last checked	22/10/2010, 22/10/2010
Location	Generating with patterns preliminaries
Calls	pair-off-lists
Called by	indices-of-max-subset-score
Comments/see also	

Example:

```
(setq
 patterns-hash
 (read-from-file-balanced-hash-table
  (concatenate
   'string
   *MCStylistic-Oct2010-example-files-path*
   "/patterns-hash.txt")))
(unaddressed-patterns-subset-scores patterns-hash)
--> ((((1 (0 0)) (1 (0 1)) (1 (0 2)) (1 (0 3)))
      8.968646)
     (((0 (1 0)) (0 (1 1))) 8.957496)
     (((0 (2 0)) (0 (2 1))) 8.167285)
     (((1 (3 0)) (1 (3 1)) (1 (3 2)) (1 (3 3))
       (2 (3 4)) (0 (3 5)) (1 (3 6)) (2 (3 7))
       (2 (3 8)) (1 (3 9)) (2 (3 10))) 3.8855853)).
```

This function scans a patterns-hash for unaddressed patterns. If a pattern is unaddressed, its subset score and indices will appear in the output, as well as its pattern importance rating.

4.6.12 Pattern inheritance preliminaries

These functions filter the results of applying SIACT to an excerpt. The result is a list consisting of hash tables, where each hash table consists of the keys: index, name, cardinality, occurrences, MTP vectors, rating, compactness, expected occurrences, compression ratio, pattern, region, and translators. The most important function in this file is called prepare-for-pattern- inheritance.

indices-of-patterns-equalp-trans&intersect

Started, last checked	20/10/2010, 20/10/2010
Location	Pattern inheritance preliminaries
Calls	intersection-multidimensional
Called by	remove-patterns-equalp-trans&intersect
Comments/see also	

Example:

```
(setq
 patterns-hash
 (read-from-file-balanced-hash-table
  (concatenate
   'string
   "/Applications/CCL/Lisp documentation"
   "/Example files/patterns-hash.txt")))
(indices-of-patterns-equalp-trans&intersect
 (gethash '"pattern" (first patterns-hash))
 (gethash '"translators" (first patterns-hash))
 (rest patterns-hash))
--> (0 3 4).
```

This function takes information about a pattern from the first hash table in a list of hash tables. It then compares this with each pattern in the rest of the list. If a pair of patterns have the same translation vectors and their first occurrences have intersecting datapoints, then the second pattern in the pair will have to be removed. To this end, the index of the second pattern is returned.

prepare-for-pattern-inheritance

Started, last checked	20/10/2010, 20/10/2010
Location	Pattern inheritance preliminaries
Calls	evaluate-variables-of-patterns2hash, remove-overlapping-translators-of-patterns, remove-patterns-equalp-trans&intersect, remove-patterns-shorter-than, subset-scores-of-patterns+, translate-patterns-to-1st-occurrences
Called by	
Comments/see also	

Example:

```
(progn
  (setq
   SIACT-output
   (read-from-file
    (concatenate
     'string
     "/Applications/CCL/Lisp documentation"
     "/Example files/SIACT-output.txt")))
  (setq
   dataset-all
   (read-from-file
    "/Users/tec69/Open/Music/Datasets/C-68-1-ed.txt"))
  (setq dataset-mini (subseq dataset-all 0 350))
  (setq
   projected-dataset
   (orthogonal-projection-unique-equalp
    dataset-mini '(1 1 1 0 0)))
  "Yes!")
(setq
 patterns-hash
 (prepare-for-pattern-inheritance
  SIACT-output projected-dataset 1))
--> gives a hash table called patterns-hash.
(write-to-file-balanced-hash-table
 patterns-hash
 (concatenate
  'string
  "/Applications/CCL/Lisp documentation"
  "/Example files/patterns-hash2.txt"))
```

This function applies functions that prepare the output of SIACT run on a dataset for Markov-chain Monte Carlo generation with pattern inheritance.

remove-overlapping-translators

Started, last checked	20/10/2010, 20/10/2010
Location	Pattern inheritance preliminaries
Calls	
Called by	remove-overlapping-translators-of-patterns
Comments/see also	

Example:

```
(remove-overlapping-translators
 3 '((0 0 0) (1 2 3) (4 0 0) (5 0 0) (7 0 0) (8 2 1)))
--> ((0 0 0) (4 0 0) (7 0 0))
```

This function takes the duration of a pattern and its translators as arguments, and returns a list of those translators that do not produce overlapping patterns (in the sense of the argument pattern-duration).

remove-overlapping-translators-of-patterns

Started, last checked	20/10/2010, 20/10/2010
Location	Pattern inheritance preliminaries
Calls	remove-overlapping-translators
Called by	prepare-for-pattern-inheritance
Comments/see also	

Example:

```
(setq
 patterns-hash
 (read-from-file-balanced-hash-table
  (concatenate
   'string
   "/Applications/CCL/Lisp documentation"
   "/Example files/patterns-hash.txt")))
(setq
 patterns-hash
 (remove-overlapping-translators-of-patterns
  patterns-hash))
--> gives a hash table called patterns-hash.
```

This function applies the function remove-overlapping-translators recursively to a list consisting of hash tables. Each hash table contains information about a discovered pattern, as returned by the function evaluate-variables-of-patterns2hash. The output is an updated hash table.

remove-patterns-equalp-trans&intersect

Started, last checked	20/10/2010, 20/10/2010
Location	Pattern inheritance preliminaries
Calls	indices-of-patterns-equalp-trans&intersect, remove-nth-list
Called by	prepare-for-pattern-inheritance
Comments/see also	

Example:

```
(setq
 patterns-hash
 (read-from-file-balanced-hash-table
  (concatenate
   'string
   "/Applications/CCL/Lisp documentation"
   "/Example files/patterns-hash.txt")))
(setq
 patterns-hash
 (remove-patterns-equalp-trans&intersect
  patterns-hash))
--> gives a hash table called patterns-hash.
```

This function applies the function indices-of-patterns-equalp-trans&intersect recursively. The result is that the lower-rating of any pair of patterns is removed if the two patterns have the same translation vectors and their first occurrences have intersecting datapoints. It is assumed that each pattern has already been arranged so that its first translation vector is the zero vector.

remove-patterns-shorter-than

Started, last checked	20/10/2010, 20/10/2010
Location	Pattern inheritance preliminaries
Calls	
Called by	prepare-for-pattern-inheritance
Comments/see also	

Example:

```
(setq
```

```
 lil-pattern&sources
 '((((1/2 72 67 1/2) (1 76 69 1/2) (3/2 79 71 1/2)
     (2 84 74 2) (5/2 67 64 1/2) (3 64 62 1/2)
     (7/2 60 60 1/2))
    16/23 (140 5 0) 1 (104 -5 0) 4/5 (96 -5 0))
   (((1/2 72 67 1/2) (3/2 79 71 1/2))
    1 (130 0 0) 2/3 (100 0 1/2))))
(remove-patterns-shorter-than lil-pattern&sources 3)
--> ((((1/2 72 67 1/2) (1 76 69 1/2)
       (3/2 79 71 1/2) (2 84 74 2) (5/2 67 64 1/2)
       (3 64 62 1/2) (7/2 60 60 1/2))
      16/23 (140 5 0) 1 (104 -5 0) 4/5 (96 -5 0)))
```

Let a be the floor of the first ontime and b be the ceiling of the last offtime of a pattern. If this is less than the optional variable duration-threshold, then this pattern will not appear in the output of this function.

subset-score-of-pattern

Started, last checked	20/10/2010, 20/10/2010
Location	Pattern inheritance preliminaries
Calls	fibonacci-list, my-last, subset-multidimensional, translations
Called by	subset-scores-of-patterns+
Comments/see also	

Example:

```
(setq
 patterns-hash
 (read-from-file-balanced-hash-table
  (concatenate
   'string
   "/Applications/CCL/Lisp documentation"
   "/Example files/patterns-hash.txt")))
(subset-score-of-pattern
 (gethash '"pattern" (nth 6 patterns-hash))
 6 patterns-hash)
--> '(2 2 1)
```

This function takes a pattern as its first argument, called the probe pattern, and a hash table of patterns as its second argument. It counts and returns the number of patterns in the hash table (including translations) of which the probe pattern is a subset.

subset-scores-of-patterns+

Started, last checked	20/10/2010, 20/10/2010
Location	Pattern inheritance preliminaries
Calls	first-n-naturals, subset-score-of-pattern, translations
Called by	subset-scores-of-patterns+
Comments/see also	

Example:

```
(setq
 patterns-hash
 (read-from-file-balanced-hash-table
  (concatenate
   'string
   "/Applications/CCL/Lisp documentation"
   "/Example files/patterns-hash.txt")))
(setq
 patterns-hash
 (subset-scores-of-patterns+ patterns-hash))
--> gives a hash table called patterns-hash.
```

This function applies the function subset- score-of-pattern to each pattern (including translations) listed in a hash table of patterns. It also creates inheritance indices (for example the first occurrence of the highest-rating pattern is labelled $P_{0,0}$) and a variable called inheritance addressed, set to "No" by default, but will revert to "Yes" when patterns are incorporated into the generated passage. This function is the last step in preparing a hash table of patterns for generation with pattern inheritance.

translate-pattern-to-1st-occurrence

Started, last checked	20/10/2010, 20/10/2010
Location	Pattern inheritance preliminaries
Calls	constant-vector, multiply-list-by-constant, translation, vector<vector-t-or-nil
Called by	translate-patterns-to-1st-occurrences
Comments/see also	

Example:

```
(setq
 pattern
 '((1/2 72 67 1/2) (1 76 69 1/2) (3/2 79 71 1/2)
   (2 84 74 2) (5/2 67 64 1/2) (3 64 62 1/2)
   (7/2 60 60 1/2)))
(setq translators '((-1/2 0 0 0) (0 0 0 0) (3 2 1 0)))
(translate-pattern-to-1st-occurrence
 pattern translators)
--> (((0 72 67 1/2) (1/2 76 69 1/2) (1 79 71 1/2)
      (3/2 84 74 2) (2 67 64 1/2) (5/2 64 62 1/2)
      (3 60 60 1/2))
     ((0 0 0 0) (1/2 0 0 0) (7/2 2 1 0)))
```

Sometimes an occurrence of a pattern is found, other than the first occurrence in a piece. This function takes such instances and rearranges the pattern and the translators, so it is the first occurrence which is displayed.

translate-patterns-to-1st-occurrences

Started, last checked	20/10/2010, 20/10/2010
Location	Pattern inheritance preliminaries
Calls	translate-pattern-to-1st-occurrence
Called by	prepare-for-pattern-inheritance
Comments/see also	

Example:

```
(setq
 patterns-hash
 (read-from-file-balanced-hash-table
```

```
  (concatenate
   'string
   "/Applications/CCL/Lisp documentation"
   "/Example files/patterns-hash.txt")))
(setq
 patterns-hash
 (translate-patterns-to-1st-occurrences
  patterns-hash))
--> gives a hash table called pattern-hash.
```

This function applies the function translate-pattern-to-1st-occurrence recursively to a list consisting of hash tables. Each hash table contains information about a discovered pattern, as returned by the function evaluate-variables-of-patterns2hash. The output is an updated hash table.

4.6.13 Generating with patterns

The main function here is generate-beat-spacing<->pattern-inheritance, at the heart of the model named Racchmaninof-Oct2010 (standing for RAndom Constrained CHain of Markovian Nodes with INheritance Of Form).

generate-beat-spacing<->pattern-inheritance

Started, last checked	25/10/2010, 25/10/2010
Location	Generating with patterns
Calls	generate-beat-spacing-for-intervals, generate-intervals, indices-of-max-subset-score, merge-sort-by-vector<vector-car, my-last, nth-list-of-lists, translate-to-other-occurrences, translation
Called by	
Comments/see also	

Example: see Stylistic composition with Racchmaninof-Oct2010 (Sec. 3.5).

This function is at the heart of the model named Racchmaninof-Oct2010 (standing for RAndom Constrained CHain of Markovian Nodes with INheritance Of Form). It takes nine mandatory arguments and twenty-two optional arguments. The mandatory arguments are initial-state and state-transition lists, and also information pertaining to a so-called *template with patterns*.

The optional arguments are mainly for controlling various criteria like: not too many consecutive states from the same source, the range must be comparable with that of the template, and the likelihood of the states must be comparable with that of the template.

generate-beat-spacing-for-interval

Started, last checked	25/10/2010, 25/10/2010
Location	Generating with patterns
Calls	beat-spacing-states, beat-spacing-states<-, generate-beat-spacing-forced<->, most-plausible-join, my-last, nth-list-of-lists, remove-datapoints-with-nth-item<, remove-datapoints-with-nth-item>
Called by	generate-beat-spacing-for-intervals
Comments/see also	

Example:

```
(progn
  (setq
   external-initial-states
   (read-from-file
    (concatenate
     'string
     *MCStylistic-Oct2010-example-files-path*
     "/Racchmaninof-Oct2010 example"
     "/initial-states.txt")))
  (setq
   internal-initial-states
   (read-from-file
    (concatenate
     'string
     *MCStylistic-Oct2010-example-files-path*
     "/Racchmaninof-Oct2010 example"
     "/internal-initial-states.txt")))
  (setq
   stm->
   (read-from-file
    (concatenate
     'string
```

```
    *MCStylistic-Oct2010-example-files-path*
    "/Racchmaninof-Oct2010 example"
    "/transition-matrix.txt")))
  (setq
   external-final-states
   (read-from-file
    (concatenate
     'string
     *MCStylistic-Oct2010-example-files-path*
     "/Racchmaninof-Oct2010 example"
     "/final-states.txt")))
  (setq
   internal-final-states
   (read-from-file
    (concatenate
     'string
     *MCStylistic-Oct2010-example-files-path*
     "/Racchmaninof-Oct2010 example"
     "/internal-final-states.txt")))
  (setq
   stm<-
   (read-from-file
    (concatenate
     'string
     *MCStylistic-Oct2010-example-files-path*
     "/Racchmaninof-Oct2010 example"
     "/transition-matrix<-.txt")))
  (setq
   dataset-all
   (read-from-file
    (concatenate
     'string
     *MCStylistic-Oct2010-data-path*
     "/Dataset/C-56-1-ed.txt")))
  (setq
   dataset-template
   (subseq dataset-all 0 132))
  "Data imported.")
(progn
  (setq generation-interval '(12 24))
  (setq
```

```
 whole-piece-interval
 (list
  (floor (first (first dataset-all)))
  (ceiling (first (my-last dataset-all)))))
(setq A (make-hash-table :test #'equal))
(setf
 (gethash
  '"united-candidates,1,1,superimpose" A)
 '((9 70 66 3 0) (9 79 71 1/2 0) (19/2 74 68 3/2 0)
   (10 55 57 1 1) (10 62 61 1 1) (11 55 57 1 1)
   (11 62 61 1 1) (12 38 47 1 1) (12 69 65 1/2 0)))
(setq B (make-hash-table :test #'equal))
(setf
 (gethash
  '"united-candidates,2,3,forwards-dominant" B)
 '((24 38 47 1 1) (49/2 66 63 1/2 0) (25 50 54 1 1)
   (25 57 58 1 1) (25 60 60 1 1) (25 62 61 1 0)
   (26 50 54 1 1) (26 57 58 1 1) (26 60 60 1 1)
   (26 62 61 1 1) (26 66 63 1 0) (26 70 66 3/4 0)
   (107/4 69 65 1/4 0) (27 43 50 1 1)
   (27 67 64 2 0)))
(setq
 interval-output-pairs
 (list
  (list
   (list 9 12)
   (list
    "united-candidates,1,1,superimpose"
    (list nil nil A)))
  (list
   (list 24 27)
   (list
    "united-candidates,2,3,forwards-dominant"
    (list nil nil B)))))
(setq
 pattern-region
 '((12 60 60) (12 64 62) (25/2 57 58) (51/4 64 62)
   (13 52 55) (13 64 62) (14 54 56) (29/2 62 61)
   (15 55 57) (15 59 59) (63/4 64 62) (16 43 50)
   (16 50 54) (16 66 63) (17 67 64) (18 57 58)
   (18 60 60) (18 64 62) (75/4 66 63) (19 43 50)
```

```
     (19 50 54) (19 67 64) (20 66 63) (20 69 65)
     (21 59 59) (21 67 64) (21 71 66) (87/4 74 68)
     (22 43 50) (22 50 54) (22 74 68) (23 62 61)
     (24 60 60) (24 64 62)))
  "Argument instances defined.")
(progn
  (setq
   checklist
   (list "originalp" "mean&rangep" "likelihoodp"))
  (setq beats-in-bar 3) (setq c-failures 10)
  (setq c-sources 4) (setq c-bar 48) (setq c-min 38)
  (setq c-max 38) (setq c-beats 38) (setq c-prob 1)
  (setq c-forwards 3) (setq c-backwards 3)
  (setq
   *rs* #.(CCL::INITIALIZE-RANDOM-STATE 56302 14832))
  "Parameters set.")
(progn
  (setq
   interval-output-pair
   (generate-beat-spacing-for-interval
    generation-interval whole-piece-interval
    interval-output-pairs external-initial-states
    internal-initial-states stm->
    external-final-states internal-final-states stm<-
    dataset-template pattern-region checklist
    beats-in-bar c-failures c-sources c-bar c-min
    c-max c-beats c-prob c-forwards c-backwards))
--> ((12 24)
     ("united,2,1,backwards-dominant"
      (#<HASH-TABLE
       :TEST EQUAL size 3/60 #x30004340CE3D>
       #<HASH-TABLE
       :TEST EQUAL size 3/60 #x30004340C82D>
       #<HASH-TABLE
       :TEST EQUAL size 27/60 #x30004340C21D>)))
```

This function generates material for a specified time interval, by calling the function generate-beat-spacing-forced<->.

generate-beat-spacing-for-intervals

Started, last checked	25/10/2010, 25/10/2010
Location	Generating with patterns
Calls	generate-beat-spacing-for-interval
Called by	generate-beat-spacing<->pattern-inheritance
Comments/see also	

Example: see example for generate-beat-spacing-for-interval.

This function applies the function generate-beat-spacing-for-interval to each member of a list called `interval-output-pairs`.

interval-output-pairs2dataset

Started, last checked	25/10/2010, 15/5/2015
Location	Generating with patterns
Calls	unite-datapoints
Called by	generate-beat-spacing<->pattern-inheritance
Comments/see also	Updated with an optional argument to allow specification of the location of a point set within the output part of an interval-output list.

Example: see example for unite-datapoints.

This function applies the function unite-datapoints, to convert the output for various intervals into a dataset.

translate-to-other-occurrence

Started, last checked	25/10/2010, 25/10/2010
Location	Generating with patterns
Calls	translation
Called by	translate-to-other-occurrences
Comments/see also	

Example:

```
(translate-to-other-occurrence
 nil '(36 12 7) '((12 24))
 (make-hash-table :test #'equal)
 (translation
  '((12 38 47 1/2 1) (49/4 64 62 1/4 0)
    (25/2 50 54 1 1) (25/2 55 57 1 1) (25/2 58 59 1 1)
    (25/2 62 61 1 0)) '(36 12 7 0 0)))
--> (((48 60)
      ("translated material"
       (NIL NIL
        #<HASH-TABLE :TEST EQUAL
        size 1/60 #x3000436FF23D>))))
```

This function takes a list of interval- output pairs as its argument (from one iteration of generate-beat-spacing¡-¿pattern-inheritance) Its second argument is a translation vector, by which each of the output datasets must be translated.

translate-to-other-occurrences

Started, last checked	25/10/2010, 25/10/2010
Location	Generating with patterns
Calls	my-last, nth-list-of-lists, remove-nth, subtract-list-from-each-list, translate-to-other-occurrence
Called by	generate-beat-spacing<->pattern-inheritance
Comments/see also	

Example: see example for translate-to-other-occurrence.

This function applies the function translate-to-other-occurrence to each member of a list called `translators`. It first determines whether the location to which material will be translated has been addressed.

4.6.14 Realising states

These functions are used to convert states generated using random generation Markov chains (RGMC) into datapoints. A lot of the functions have similar versions in the file markov-compose.lisp.

create-MIDI&morphetic-numbers

Started, last checked	1/9/2010, 1/9/2010
Location	Realising states
Calls	add-to-list, nth-list-of-lists
Called by	states2datapoints-by-lookup
Comments/see also	create-MIDI-note-numbers, create-MIDI&morphetic-numbers<-

Example:

```
(setq
 states
 '(((1 (12 7 5 4))
    (NIL NIL "C-63-1"
         ((0 35 45 1/2 1 1/2 0) (0 47 52 1/2 1 1/2 1)
          (0 54 56 1/2 0 1/2 2) (0 59 59 1/2 0 1/2 3)
          (0 63 61 1/2 0 1/2 4))))
   ((3/2 "rest")
    (NIL NIL "C-63-1" NIL))
   ((7/4 NIL)
    (-6 -3 "C-63-1"
        ((831/4 56 57 1/4 0 208 701))))
   ((2 (12 14 6 3))
    (-26 -15 "C-63-1"
         ((208 30 42 1 1 209 702)
          (208 42 49 1 1 209 703)
          (208 56 57 1 0 209 704)
          (208 62 60 1 0 209 705)
          (208 65 62 1 0 209 706))))
   ((3 (12 12 9 3))
    (0 0 "C-63-1"
       ((209 30 42 1 1 210 707)
        (209 42 49 1 1 210 708)
        (209 54 56 1 0 210 709)
        (209 63 61 1 0 210 710)
        (209 66 63 1 0 210 711))))
   ((1 (12 16 6 4))
    (0 0 "C-63-1"
       ((216 30 42 1 1 217 740)
        (216 42 49 1 1 217 741)
```

```
        (216 58 58 1 0 217 742)
        (216 64 62 1 0 217 743)
        (216 68 64 1 0 217 744))))
   ((2 (5 4 3))
    (19 11 "C-63-1"
        ((43 56 57 1 1 44 195) (43 61 60 1 1 44 196)
         (43 65 62 1/2 0 87/2 197)
         (43 68 64 1 1 44 198))))
   ((5/2 (5 4 8 4))
    (0 0 "C-63-1"
       ((25 54 56 1 1 26 101) (25 59 59 1 1 26 102)
        (25 63 61 1 1 26 103)
        (51/2 71 66 1/2 0 26 105)
        (51/2 75 68 1/2 0 26 106))))
   ((3 (3))
    (19 11 "C-63-1"
        ((230 73 67 3/4 0 923/4 809)
         (230 76 69 3/4 0 923/4 810))))
   ((7/2 (4))
    (3 2 "C-67-1"
       ((41/2 72 67 1/2 0 21 94)
        (41/2 76 69 1/2 0 21 95))))))
(create-MIDI&morphetic-numbers states)
--> (((1 (60 72 79 84 88) (60 67 71 74 76))
      (NIL NIL "C-63-1"
           ((0 35 45 1/2 1 1/2 0)
            (0 47 52 1/2 1 1/2 1)
            (0 54 56 1/2 0 1/2 2)
            (0 59 59 1/2 0 1/2 3)
            (0 63 61 1/2 0 1/2 4))))
     ((3/2 NIL NIL)
      (NIL NIL "C-63-1" NIL))
     ((7/4 (54) (57))
      (-6 -3 "C-63-1"
          ((831/4 56 57 1/4 0 208 701))))
     ((2 (28 40 54 60 63) (42 49 57 60 62))
      (-26 -15 "C-63-1"
           ((208 30 42 1 1 209 702)
            (208 42 49 1 1 209 703)
            (208 56 57 1 0 209 704)
            (208 62 60 1 0 209 705)
```

```
       (208 65 62 1 0 209 706))))
((3 (28 40 52 61 64) (42 49 56 61 63))
 (0 0 "C-63-1"
    ((209 30 42 1 1 210 707)
     (209 42 49 1 1 210 708)
     (209 54 56 1 0 210 709)
     (209 63 61 1 0 210 710)
     (209 66 63 1 0 210 711))))
((1 (28 40 56 62 66) (42 49 58 62 64))
 (0 0 "C-63-1"
    ((216 30 42 1 1 217 740)
     (216 42 49 1 1 217 741)
     (216 58 58 1 0 217 742)
     (216 64 62 1 0 217 743)
     (216 68 64 1 0 217 744))))
((2 (47 52 56 59) (53 56 58 60))
 (19 11 "C-63-1"
     ((43 56 57 1 1 44 195) (43 61 60 1 1 44 196)
      (43 65 62 1/2 0 87/2 197)
      (43 68 64 1 1 44 198))))
((5/2 (47 52 56 64 68) (53 56 58 63 65))
 (0 0 "C-63-1"
    ((25 54 56 1 1 26 101) (25 59 59 1 1 26 102)
     (25 63 61 1 1 26 103)
     (51/2 71 66 1/2 0 26 105)
     (51/2 75 68 1/2 0 26 106))))
((3 (66 69) (64 66))
 (19 11 "C-63-1"
     ((230 73 67 3/4 0 923/4 809)
      (230 76 69 3/4 0 923/4 810))))
((7/2 (69 73) (66 68))
 (3 2 "C-67-1"
    ((41/2 72 67 1/2 0 21 94)
     (41/2 76 69 1/2 0 21 95)))))
```

This function is meant to take generated states and realise MIDI note numbers and morphetic pitch numbers for each state.

half-state2datapoints-by-lookup

Started, last checked	1/9/2010, 17/8/2014
Location	Realising states
Calls	state-note2datapoint-by-lookup
Called by	states2datapoints-by-lookup
Comments/see also	half-state2datapoints

Example:

```
(setq
 half-states
 '(((1 (60 72 79 84 88) (60 67 71 74 76))
    (NIL NIL "C-63-1"
         ((0 35 45 1/2 1 1/2 0) (0 47 52 1/2 1 1/2 1)
          (0 54 56 1/2 0 1/2 2) (0 59 59 1/2 0 1/2 3)
          (0 63 61 1/2 0 1/2 4))))
   ((3/2 NIL NIL)
    (NIL NIL "C-63-1" NIL))
   ((7/4 (54) (57))
    (-6 -3 "C-63-1"
        ((831/4 56 57 1/4 0 208 701))))
   ((2 (28 40 54 60 63) (42 49 57 60 62))
    (-26 -15 "C-63-1"
         ((208 30 42 1 1 209 702)
          (208 42 49 1 1 209 703)
          (208 56 57 1 0 209 704)
          (208 62 60 1 0 209 705)
          (208 65 62 1 0 209 706))))
   ((3 (28 40 52 61 64) (42 49 56 61 63))
    (0 0 "C-63-1"
       ((209 30 42 1 1 210 707)
        (209 42 49 1 1 210 708)
        (209 54 56 1 0 210 709)
        (209 63 61 1 0 210 710)
        (209 66 63 1 0 210 711))))
   ((1 (28 40 56 62 66) (42 49 58 62 64))
    (0 0 "C-63-1"
       ((216 30 42 1 1 217 740)
        (216 42 49 1 1 217 741)
        (216 58 58 1 0 217 742)
```

```
        (216 64 62 1 0 217 743)
        (216 68 64 1 0 217 744))))
   ((2 (47 52 56 59) (53 56 58 60))
    (19 11 "C-63-1"
        ((43 56 57 1 1 44 195) (43 61 60 1 1 44 196)
         (43 65 62 1/2 0 87/2 197)
         (43 68 64 1 1 44 198))))
   ((5/2 (47 52 56 64 68) (53 56 58 63 65))
    (0 0 "C-63-1"
       ((25 54 56 1 1 26 101) (25 59 59 1 1 26 102)
        (25 63 61 1 1 26 103)
        (51/2 71 66 1/2 0 26 105)
        (51/2 75 68 1/2 0 26 106))))
   ((3 (66 69) (64 66))
    (19 11 "C-63-1"
        ((230 73 67 3/4 0 923/4 809)
         (230 76 69 3/4 0 923/4 810))))
   ((7/2 (69 73) (66 68))
    (3 2 "C-67-1"
       ((41/2 72 67 1/2 0 21 94)
        (41/2 76 69 1/2 0 21 95))))))
(half-state2datapoints-by-lookup
 0 half-states
 '(500 500 500 500 500) '(0 500 1000 1500 2000 2500))
--> ((0 60 60 500 1) (0 72 67 500 1) (0 79 71 500 0)
     (0 84 74 500 0) (0 88 76 500 0))
```

This function increments over the ith note of a half-state, $i = 0, 1, \ldots, n-1$. The function state-note2datapoint-by-lookup is applied.

index-of-offtime-by-lookup

Started, last checked	1/9/2010, 17/8/2014
Location	Realising states
Calls	nth-list-of-lists
Called by	state-note2datapoint-by-lookup
Comments/see also	index-of-offtime

Example:

```
(index-of-offtime-by-lookup
```

```
 3 28
 '(((1 (60 72 79 84 88) (60 67 71 74 76))
      (NIL NIL "C-63-1"
           ((0 35 45 1/2 1 1/2 0)
            (0 47 52 1/2 1 1/2 1)
            (0 54 56 1/2 0 1/2 2)
            (0 59 59 1/2 0 1/2 3)
            (0 63 61 1/2 0 1/2 4))))
     ((3/2 NIL NIL)
      (NIL NIL "C-63-1" NIL))
     ((7/4 (54) (57))
      (-6 -3 "C-63-1"
          ((831/4 56 57 1/4 0 208 701))))
     ((2 (28 40 54 60 63) (42 49 57 60 62))
      (-26 -15 "C-63-1"
           ((208 30 42 2 1 209 702)
            (208 42 49 1 1 209 703)
            (208 56 57 1 0 209 704)
            (208 62 60 1 0 209 705)
            (208 65 62 1 0 209 706))))
     ((3 (28 40 52 61 64) (42 49 56 61 63))
      (0 0 "C-63-1"
         ((209 30 42 1 1 210 707)
          (209 42 49 1 1 210 708)
          (209 54 56 1 0 210 709)
          (209 63 61 1 0 210 710)
          (209 66 63 1 0 210 711)))))
 '(1/2 1/4 1/4 1) 3)
--> 4
```

Given a starting index, a note-index and some half-states to search through, this function returns the index of the half-state where the note- number in question comes to an end.

state-durations-by-beat

Started, last checked	1/9/2010, 23/11/2014
Location	Realising states
Calls	min-item, nth-list-of-lists
Called by	states2datapoints-by-lookup
Comments/see also	state-durations

Example:

```
(setq
 states
 '(((1 (12 7 5 4))
    (NIL NIL "C-63-1"
         ((0 35 45 1/2 1 1/2 0) (0 47 52 1/2 1 1/2 1)
          (0 54 56 1/2 0 1/2 2) (0 59 59 1/2 0 1/2 3)
          (0 63 61 1/2 0 1/2 4))))
   ((3/2 "rest")
    (NIL NIL "C-63-1" NIL))
   ((7/4 NIL)
    (-6 -3 "C-63-1"
        ((831/4 56 57 1/4 0 208 701))))
   ((2 (12 14 6 3))
    (-26 -15 "C-63-1"
         ((208 30 42 1 1 209 702)
          (208 42 49 1 1 209 703)
          (208 56 57 1 0 209 704)
          (208 62 60 1 0 209 705)
          (208 65 62 1 0 209 706))))
   ((3 (12 12 9 3))
    (0 0 "C-63-1"
       ((209 30 42 1 1 210 707)
        (209 42 49 1 1 210 708)
        (209 54 56 1 0 210 709)
        (209 63 61 1 0 210 710)
        (209 66 63 1 0 210 711))))
   ((1 (12 16 6 4))
    (0 0 "C-63-1"
       ((216 30 42 1 1 217 740)
        (216 42 49 1 1 217 741)
        (216 58 58 1 0 217 742)
        (216 64 62 1 0 217 743)
        (216 68 64 1 0 217 744))))
   ((2 (5 4 3))
    (19 11 "C-63-1"
        ((43 56 57 1 1 44 195) (43 61 60 1 1 44 196)
         (43 65 62 1/2 0 87/2 197)
         (43 68 64 1 1 44 198))))
   ((5/2 (5 4 8 4))
```

```
    (0 0 "C-63-1"
       ((25 54 56 1 1 26 101) (25 59 59 1 1 26 102)
        (25 63 61 1 1 26 103)
        (51/2 71 66 1/2 0 26 105)
        (51/2 75 68 1/2 0 26 106))))
   ((3 (3))
    (19 11 "C-63-1"
        ((230 73 67 3/4 0 923/4 809)
         (230 76 69 3/4 0 923/4 810))))
   ((7/2 (4))
    (3 2 "C-67-1"
       ((41/2 72 67 1/2 0 21 94)
        (41/2 76 69 1/2 0 21 95))))))
(state-durations-by-beat states 3)
--> (1/2 1/4 1/4 1 1 1 1/2 1/2 1/2)
```

This function takes a list of states as its argument, and returns a list containing the duration of each state. It is a little more involved than the function state-durations, because of the nature of the beat-spacing states.

state-note2datapoint-by-lookup

Started, last checked	1/9/2010, 17/8/2014
Location	Realising states
Calls	index-of-offtime-by-lookup
Called by	half-state2datapoints-by-lookup
Comments/see also	state-note2datapoint

Example:

```
(setq
 half-states
 '(((1 (60 72 79 84 88) (60 67 71 74 76))
      (NIL NIL "C-63-1"
           ((0 35 45 1/2 1 1/2 0)
            (0 47 52 1/2 1 1/2 1)
            (0 54 56 1/2 0 1/2 2)
            (0 59 59 1/2 0 1/2 3)
            (0 63 61 1/2 0 1/2 4))))
     ((3/2 NIL NIL)
      (NIL NIL "C-63-1" NIL))
```

```
((7/4 (54) (57))
 (-6 -3 "C-63-1"
     ((831/4 56 57 1/4 0 208 701))))
((2 (28 40 54 60 63) (42 49 57 60 62))
 (-26 -15 "C-63-1"
      ((208 30 42 1 1 209 702)
       (208 42 49 1 1 209 703)
       (208 56 57 1 0 209 704)
       (208 62 60 1 0 209 705)
       (208 65 62 1 0 209 706))))
((3 (28 40 52 61 64) (42 49 56 61 63))
 (0 0 "C-63-1"
    ((209 30 42 1 1 210 707)
     (209 42 49 1 1 210 708)
     (209 54 56 1 0 210 709)
     (209 63 61 1 0 210 710)
     (209 66 63 1 0 210 711))))
((1 (28 40 56 62 66) (42 49 58 62 64))
 (0 0 "C-63-1"
    ((216 30 42 1 1 217 740)
     (216 42 49 1 1 217 741)
     (216 58 58 1 0 217 742)
     (216 64 62 1 0 217 743)
     (216 68 64 1 0 217 744))))
((2 (47 52 56 59) (53 56 58 60))
 (19 11 "C-63-1"
     ((43 56 57 1 1 44 195) (43 61 60 1 1 44 196)
      (43 65 62 1/2 0 87/2 197)
      (43 68 64 1 1 44 198))))
((5/2 (47 52 56 64 68) (53 56 58 63 65))
 (0 0 "C-63-1"
    ((25 54 56 1 1 26 101) (25 59 59 1 1 26 102)
     (25 63 61 1 1 26 103)
     (51/2 71 66 1/2 0 26 105)
     (51/2 75 68 1/2 0 26 106))))
((3 (66 69) (64 66))
 (19 11 "C-63-1"
     ((230 73 67 3/4 0 923/4 809)
      (230 76 69 3/4 0 923/4 810))))
((7/2 (69 73) (66 68))
 (3 2 "C-67-1"
```

```
         ((41/2 72 67 1/2 0 21 94)
          (41/2 76 69 1/2 0 21 95))))))
(state-note2datapoint-by-lookup
 4 0 half-states '(1/2 1/4 1/4 1 1 1 1/2 1/2 1/2 1/2)
 '(0 1/2 3/4 1 2 3 4 9/2 5 11/2 6))
--> (0 88 76 1/2 0)
```

The ith note of the jth half-state is transformed into a so-called 'datapoint', meaning we find its ontime (the jth element of the partition points), its MIDI note number, morphetic pitch number, duration, and voice.

states2datapoints-by-lookup

Started, last checked	1/9/2010, 1/9/2010
Location	Realising states
Calls	create-MIDI&morphetic-numbers, fibonacci-list, half-state2datapoints-by-lookup, state-durations-by-beat
Called by	
Comments/see also	states2datapoints, states2datapoints-by-lookup<-

Example:

```
(setq
 states
 '(((1 (12 7 5 4))
    (NIL NIL "C-63-1"
         ((0 35 45 1/2 1 1/2 0) (0 47 52 1/2 1 1/2 1)
          (0 54 56 1/2 0 1/2 2) (0 59 59 1/2 0 1/2 3)
          (0 63 61 1/2 0 1/2 4))))
   ((3/2 "rest")
    (NIL NIL "C-63-1" NIL))
   ((7/4 NIL)
    (-6 -3 "C-63-1"
        ((831/4 56 57 1/4 0 208 701))))
   ((2 (12 14 6 3))
    (-26 -15 "C-63-1"
         ((208 30 42 1 1 209 702)
          (208 42 49 1 1 209 703)
```

```
          (208 56 57 1 0 209 704)
          (208 62 60 1 0 209 705)
          (208 65 62 1 0 209 706))))
   ((3 (12 12 9 3))
    (0 0 "C-63-1"
       ((209 30 42 1 1 210 707)
        (209 42 49 1 1 210 708)
        (209 54 56 1 0 210 709)
        (209 63 61 1 0 210 710)
        (209 66 63 1 0 210 711))))
   ((1 (12 16 6 4))
    (0 0 "C-63-1"
       ((216 30 42 1 1 217 740)
        (216 42 49 1 1 217 741)
        (216 58 58 1 0 217 742)
        (216 64 62 1 0 217 743)
        (216 68 64 1 0 217 744))))
   ((2 (5 4 3))
    (19 11 "C-63-1"
        ((43 56 57 1 1 44 195) (43 61 60 1 1 44 196)
         (43 65 62 1/2 0 87/2 197)
         (43 68 64 1 1 44 198))))
   ((5/2 (5 4 8 4))
    (0 0 "C-63-1"
       ((25 54 56 1 1 26 101) (25 59 59 1 1 26 102)
        (25 63 61 1 1 26 103)
        (51/2 71 66 1/2 0 26 105)
        (51/2 75 68 1/2 0 26 106))))
   ((3 (3))
    (19 11 "C-63-1"
        ((230 73 67 3/4 0 923/4 809)
         (230 76 69 3/4 0 923/4 810))))
   ((7/2 (4))
    (3 2 "C-67-1"
       ((41/2 72 67 1/2 0 21 94)
        (41/2 76 69 1/2 0 21 95))))))
(states2datapoints-by-lookup states 3 60 60)
--> ((0 60 60 1/2 1) (0 72 67 1/2 1) (0 79 71 1/2 0)
     (0 84 74 1/2 0) (0 88 76 1/2 0) (3/4 54 57 1/4 0)
     (1 28 42 1 1) (1 40 49 1 1) (1 54 57 1 0)
     (1 60 60 1 0) (1 63 62 1 0) (2 28 42 1 1)
```

```
(2 40 49 1 1) (2 52 56 1 0) (2 61 61 1 0)
(2 64 63 1 0) (3 28 42 1 1) (3 40 49 1 1)
(3 56 58 1 0) (3 62 62 1 0) (3 66 64 1 0)
(4 47 53 1 1) (4 52 56 1 1) (4 56 58 1/2 0)
(4 59 60 1/2 1) (9/2 56 58 1/2 1)
(9/2 64 63 1/2 0) (9/2 68 65 1/2 0)
(5 66 64 1/2 0) (5 69 66 1 0)
(11/2 73 68 1/2 0))
```

This function applies the function half-state2datapoint-by-lookup recursively to a list of states.

4.6.15 Realising states backwards

These functions are used to convert states generated using a backwards running random generation Markov chain (RGMC) into datapoints. The functions have similar versions in the files realising-states.lisp and markov-compose.lisp.

create-MIDI&morphetic-numbers<-

Started, last checked	1/9/2010, 1/9/2010
Location	Realising states backwards
Calls	add-to-list, nth-list-of-lists
Called by	states2datapoints-by-lookup<-
Comments/see also	create-MIDI&morphetic-numbers, create-MIDI-note-numbers

Example:

```
(setq
 states<-
 '(((3 NIL) (NIL NIL "C-6-2" ((285 61 60 3 0 288 6))))
   ((2 (5 3 9))
    (17 10 "C-6-2"
     ((58 56 57 1 1 59 219) (58 61 60 1 1 59 220)
      (58 64 62 1 1 59 221)
      (58 73 67 3/2 0 119/2 222))))
   ((3/2 (3 9 14))
    (7 4 "C-7-2"
     ((120 50 54 1 1 121 351) (120 53 56 1 1 121 352)
```

```
      (120 62 61 1 1 121 353)
      (241/2 76 69 1/2 0 121 355))))
   ((1 (3 9 15))
    (0 0 "C-7-2"
     ((120 50 54 1 1 121 351) (120 53 56 1 1 121 352)
      (120 62 61 1 1 121 353)
      (120 77 70 1/2 0 241/2 354))))
   ((7/2 (3 6 15))
    (-2 -1 "C-7-2"
     ((119 52 55 1 1 120 346) (119 55 57 1 1 120 347)
      (119 61 60 1 1 120 348)
      (239/2 76 69 1/2 0 120 350))))))
(create-MIDI&morphetic-numbers<- states<- 57 58)
--> (((7/2 (35 38 44 59) (45 47 50 59))
      (-2 -1 "C-7-2"
       ((119 52 55 1 1 120 346)
        (119 55 57 1 1 120 347)
        (119 61 60 1 1 120 348)
        (239/2 76 69 1/2 0 120 350))))
     ((1 (33 36 45 60) (44 46 51 60))
      (0 0 "C-7-2"
       ((120 50 54 1 1 121 351)
        (120 53 56 1 1 121 352)
        (120 62 61 1 1 121 353)
        (120 77 70 1/2 0 241/2 354))))
     ((3/2 (33 36 45 59) (44 46 51 59))
      (7 4 "C-7-2"
       ((120 50 54 1 1 121 351)
        (120 53 56 1 1 121 352)
        (120 62 61 1 1 121 353)
        (241/2 76 69 1/2 0 121 355))))
     ((2 (40 45 48 57) (48 51 53 58))
      (17 10 "C-6-2"
       ((58 56 57 1 1 59 219) (58 61 60 1 1 59 220)
        (58 64 62 1 1 59 221)
        (58 73 67 3/2 0 119/2 222))))
     ((3 (57) (58))
      (NIL NIL "C-6-2" ((285 61 60 3 0 288 6)))))
```

This function is similar to the function create-MIDI&morphetic-numbers. The difference is that states generated by a backwards-running Markov model are supplied as the argument, and the intervals between bass notes refer to states X_n and X_{n+1}, rather than X_n and X_{n-1}, so the unpacking proceeds differently. The states are reversed into temporal order at the end.

states2datapoints-by-lookup<-

Started, last checked	1/9/2010, 1/9/2010
Location	Realising states backwards
Calls	create-MIDI&morphetic-numbers<-, fibonacci-list, half-state2datapoints-by-lookup, state-durations-by-beat
Called by	
Comments/see also	states2datapoints, states2datapoints-by-lookup

Example:

```
(setq
 states<-
 '(((3 NIL) (NIL NIL "C-6-2" ((285 61 60 3 0 288 6))))
   ((2 (5 3 9))
    (17 10 "C-6-2"
     ((58 56 57 1 1 59 219) (58 61 60 1 1 59 220)
      (58 64 62 1 1 59 221)
      (58 73 67 3/2 0 119/2 222))))
   ((3/2 (3 9 14))
    (7 4 "C-7-2"
     ((120 50 54 1 1 121 351) (120 53 56 1 1 121 352)
      (120 62 61 1 1 121 353)
      (241/2 76 69 1/2 0 121 355))))
   ((1 (3 9 15))
    (0 0 "C-7-2"
     ((120 50 54 1 1 121 351) (120 53 56 1 1 121 352)
      (120 62 61 1 1 121 353)
      (120 77 70 1/2 0 241/2 354))))
   ((7/2 (3 6 15))
    (-2 -1 "C-7-2"
     ((119 52 55 1 1 120 346) (119 55 57 1 1 120 347)
      (119 61 60 1 1 120 348)
      (239/2 76 69 1/2 0 120 350))))))
```

```
(states2datapoints-by-lookup<- states<- 3 57 58)
--> ((0 35 45 1/2 1) (0 38 47 1/2 1) (0 44 50 1/2 1)
     (0 59 59 1/2 0) (1/2 33 44 1 1) (1/2 36 46 1 1)
     (1/2 45 51 2 1) (1/2 60 60 1/2 0) (1 59 59 1/2 0)
     (3/2 40 48 1 1) (3/2 48 53 1 1) (3/2 57 58 4 0))
```

This function is very similar to the function states2datapoints-by-lookup. It applies the function half-state2datapoint-by-lookup recursively to a list of states generated by a backward-running Markov model.

4.7 Harmony and metre

4.7.1 Ontimes signatures

These lisp functions help with converting between ontimes and bar/beat numbers, when time signatures change according to the variable time-sigs-with- ontimes, which is a list of time signatures in a piece. In this list of lists: the first item of a list is a bar number; the second item is the upper number of the time signature in that bar (specifying number of beats per bar); the third item is the lower number of the time signature (specifying the division used to count time, with 4 for crotchet, etc.); the fourth item is the corresponding ontime.

append-ontimes-to-time-signatures

Started, last checked	15/3/2010, 15/3/2010
Location	Ontimes signatures
Calls	my-last
Called by	
Comments/see also	kern-file2ontimes-signatures

Example:

```
(append-ontimes-to-time-signatures
 '((1 2 4) (2 3 8) (4 3 4) (7 5 8)))
--> ((1 2 4 0) (2 3 8 2) (4 3 4 5) (7 5 8 14))
```

This function appends ontimes to rows of the time-signature table.

bar&beat-number-of-ontime

Started, last checked	15/3/2010, 10/6/2015
Location	Ontimes signatures
Calls	row-of-max-ontime<=ontime-arg
Called by	bar-beat-ontimes
Comments/see also	

Example:

```
(bar&beat-number-of-ontime
 10 '((1 2 4 0) (2 3 8 2) (4 3 4 5) (7 5 8 14)))
--> (5 3 10)
```

Given an ontime and a time-signature table (with ontimes appended), this function returns the bar number and beat number of that ontime. Beat numbers are expressed in crotchets counting from one (see the second and third examples above).

bar-beat-ontimes

Started, last checked	15/3/2010, 15/3/2010
Location	Ontimes signatures
Calls	bar&beat-number-of-ontime
Called by	bar-beat-ontimes
Comments/see also	

Example:

```
(bar-beat-ontimes
 0 1 '((1 2 4 0) (2 3 8 2) (4 3 4 5) (7 5 8 14)))
--> ((1 1 0) (1 2 1) (2 1 2) (2 2 3) (3 3/2 4)
     (4 1 5) (4 2 6) (4 3 7) (5 1 8) (5 2 9)
     (5 3 10) (6 1 11) (6 2 12) (6 3 13) (7 1 14)
     (7 2 15))
```

Given an ontime start, a subdivision of the crotchet beat and a time-signature table (with ontimes appended, this function returns the bar and beat numbers that will be displayed at the foot of the MIDI table, and the corresponding ontimes.

increment-by-x-n-times

Started, last checked	15/3/2010, 15/3/2010
Location	Ontimes signatures
Calls	my-last
Called by	bar-beat-ontimes
Comments/see also	

Example:

```
(increment-by-x-n-times 1/2 3 7.5)
--> (7.5 8.0 8.5 9.0)
```

Adds the first argument to the third (default zero), and continues to do so until the second argument is exceeded.

ontime-of-bar&beat-number

Started, last checked	15/3/2010, 15/3/2010
Location	Ontimes signatures
Calls	my-last
Called by	bar-beat-ontimes
Comments/see also	

Example:

```
(ontime-of-bar&beat-number
 5 2 '((1 2 4 0) (2 3 8 2) (4 3 4 5) (8 5 8 17)))
--> (5 2 9)
```

Given a bar and beat number, and a time-signature table (with ontimes appended, this function returns the ontime of that bar and beat number.

row-of-max-bar<=ontime-bar

Started, last checked	15/3/2010, 15/3/2010
Location	Ontimes signatures
Calls	
Called by	ontime-of-bar&beat-number
Comments/see also	

Example:

```
(row-of-max-bar<=ontime-bar
 4 '((1 2 4 0) (2 3 8 2) (4 3 4 5) (8 5 8 17)))
--> (4 3 4 5)
```

Returns the row (in a list of time signatures) of the maximal bar number less than the bar number argument.

row-of-max-ontime<=ontime-arg

Started, last checked	15/3/2010, 2/1/2015
Location	Ontimes signatures
Calls	
Called by	bar&beat-number-of-ontime
Comments/see also	

Example:

```
(row-of-max-ontime<=ontime-arg
 7 '((1 2 4 0) (2 3 8 2) (4 3 4 5) (5 5 8 8)))
--> (4 3 4 5)
```

Returns the row (in a list of time signatures) of the maximal ontime less than the ontime argument.

2/1/2015. Added handling of negative ontimes (e.g., representing an anacrusis).

4.7.2 Chord labelling

An implementation of the HarmAn algorithm as described by Pardo and Birmingham (2002), as well as an extension of this algorithm to provide functional-harmonic analysis.

a-list-in-b-list

Started, last checked	19/4/2011, 19/4/2011
Location	Chord labelling
Calls	
Called by	score-segment-against-template
Comments/see also	Could be improved with use of the function count.

Example:

```
(a-list-in-b-list '(0 48 3 5 2 3) '(5 3 48))
--> 4
```

This function takes two lists as arguments. It returns the number of elements in the first list that are contained in the second list.

cadence-time-intervals

Started, last checked	16/5/2014, 16/5/2014
Location	Chord labelling
Calls	append-list, append-ontimes-to-time-signatures, bar&beat-number-of-ontime, firstn-list, HarmAn->roman, nth-list, nth-list-of-lists, positions, replace-all, row-of-max-ontime<=ontime-arg
Called by	
Comments/see also	

Example:

```
(setq
 path&name
 (merge-pathnames
  (make-pathname
   :directory '(:relative "C@merata2014" "misc")
   :name "dowland_denmark_galliard" :type "krn")
  *MCStylistic-MonthYear-data-path*))
(setq question-string "perfect cadence")
(setq
 point-set (kern-file2dataset-by-col path&name))
(setq
 ontimes-signatures
 (kern-file2ontimes-signatures path&name))
(cadence-time-intervals
 question-string point-set ontimes-signatures)
--> ((8 12))
```

This function uses the function HarmAn->roman to create a Roman numeral labelling for an input point set. Bigrams in this labelling are used to identify certain types of cadence queries, which can be specified as the first argument. Time windows in which the cadences occur are returned.

chord-index2MNN-mod12&class

Started, last checked	19/4/2011, 19/4/2011
Location	Chord labelling
Calls	
Called by	labelled-listed-segments2datapoints
Comments/see also	

Example:

```
(chord-index2MNN-mod12&class 37)
--> (1 3)
```

This function takes an index in the variable *chord-templates-p&b&min7ths* as argument. It can be seen from this variable that there are six different classes of chord template (0, major triad; 1, dom7; 2, minor triad; 3, fully diminished 7th; 4, half diminished 7th; 5, diminished triad; 6, minor 7th). All classes but one have unambiguous roots. For example, if the index is 4, we know that the 5th (5 = 4 + 1) element of the list is (5 9 0), and that this is a major triad with root F. This function converts the index into a pair consisting of root (MIDI note number modulo 12) and class (as listed above). The ambiguity of a fully diminished 7th chord can be resloved by context, using another function.

chord-time-intervals

Started, last checked	16/6/2015, 25/6/2015
Location	Chord labelling
Calls	
Called by	??
Comments/see also	

Example:

```
(setq
 point-set
 '((0 50 54 1/2 3) (1/2 57 58 1/2 3) (1 53 56 1/2 3)
   (3/2 57 58 1/2 3) (2 50 54 1 3) (2 69 65 1 2)
   (5/2 53 56 1/2 3) (3 55 57 1/2 3) (3 70 66 3/2 2)
   (7/2 58 59 1/2 3) (4 52 55 1/2 3) (9/2 55 57 1/2 3)
   (9/2 69 65 1/2 2) (5 49 53 1/2 3) (5 67 64 1/4 2)
   (21/4 70 66 1/4 2) (11/2 52 55 1/2 1)
```

```
   (11/2 69 65 1/2 1) (6 49 53 1/2 3) (6 67 64 1 2)))
(setq question-string "chord of D minor")
(chord-time-intervals question-string point-set)
--> ((5/2 3))
(setq question-string "chord C#3, G4")
(chord-time-intervals question-string point-set)
--> ((5 21/4) (6 13/2))
(setq question-string "chord E, A")
(chord-time-intervals question-string point-set)
--> ((11/2 6))
(setq question-string "quaver note chord")
(chord-time-intervals question-string point-set)
--> ((2 5/2) (9/2 5) (11/2 6))
(setq
 question-string "quaver note chord in the left hand")
(chord-time-intervals question-string point-set)
--> ((11/2 6))

(setq
 point-set
 '((15 60 60 1/2 1) (15 64 62 1/2 1) (15 72 67 1/2 0)
   (31/2 60 60 1/2 1) (31/2 64 62 1/2 1)
   (31/2 76 69 1/4 0) (63/4 74 68 1/4 0)
   (16 60 60 1/2 1) (16 64 62 1/2 1) (16 76 69 1/2 0)
   (33/2 60 60 1/2 1) (33/2 64 62 1/2 1)
   (33/2 79 71 1/4 0) (67/4 77 70 1/4 0)
   (17 60 60 1/2 1) (17 64 62 1/2 1) (17 79 71 1/2 0)
   (35/2 60 60 1/2 1) (35/2 64 62 1/2 1)
   (35/2 82 73 1/4 0) (71/4 81 72 1/4 0)
   (18 53 56 1/2 1) (18 60 60 1/2 1) (18 64 62 1/2 1)
   (18 81 72 2 0) (37/2 53 56 1/2 1) (37/2 60 60 1/2 1)
   (37/2 64 62 1/2 1) (19 53 56 1/2 1)))
(setq
 question-string "sixteenth note chord Bb, C, E")
(chord-time-intervals question-string point-set)
--> ((35/2 71/4))
```

This function takes a natural language query as its first argument and a point-set representation of a music excerpt as its second argument. The third (optional) argument consists of staff and clef names. The function parses the query for mention of a chord (e.g., 'C major chord', 'chord of D minor', 'sixteenth note chord Bb, C, E') and then extracts instances of this chord from the point set, returning the time intervals at which they occur. Chord names (e.g., C minor) are mapped to chord templates (e.g., C, E♭, G) using the variable *chord-name-template-assoc-p&b&min7ths*.

The function can handle pitches and pitch classes, as well as restricting extracted chords to particular durations or searching on duration alone. If the excerpt contains pitches/pitch classes specified in the query as well as some extras, then such chords will still be returned. If a chord occurs over several segments (because other notes come and go), then several adjoining time windows will be returned.

HarmAn->

Started, last checked	19/4/2011, 19/4/2011
Location	Chord labelling
Calls	HarmAn->labelling, labelled-listed-segments2datapoints, resolve-dim7s, segments-strict
Called by	labelled-listed-segments2datapoints
Comments/see also	

Example:

```
(HarmAn->
 '((15 54 56 1 3) (15 62 61 1/2 2) (15 69 65 1 1)
   (15 74 68 1 0) (31/2 60 60 1/2 2) (16 55 57 1 3)
   (16 59 59 1 2) (16 67 64 1 1) (16 74 68 1 0)
   (17 57 58 1 3) (17 60 60 1 2) (17 66 63 1 1)
   (17 74 68 1 0) (18 59 59 1 3) (18 62 61 1 2)
   (18 67 64 1 1) (18 74 68 1 0) (19 52 55 1 3)
   (19 64 62 1 2) (19 67 64 1 1) (19 71 66 1/2 0)
   (39/2 72 67 1/2 0) (20 47 52 1 3) (20 62 61 1 2)
   (20 67 64 1 1) (20 74 68 1 0) (21 48 53 1 3)
   (21 64 62 1 2) (21 69 65 1/2 1) (21 72 67 1/2 0)
   (43/2 67 64 1/2 1) (43/2 71 66 1/2 0)
   (22 50 54 1 3) (22 57 58 1 2) (22 66 63 1 1)
   (22 69 65 1 0)))
--> ((15 2 1 1 8) (16 7 0 1 4) (17 2 1 1 4)
     (18 7 0 1 4) (19 4 2 1/2 4) (39/2 0 0 1/2 4)
     (20 7 0 1 4) (21 9 2 1/2 4) (43/2 0 0 1/2 2)
     (22 2 0 1 4))
```

This function is an implementation of the forwards-running HarmAn algorithm described by Pardo and Birmingham (2002). The format of the output is a chord dataset, where the first dimension is ontime, the second dimension is the MIDI note number modulo 12 of the root of the chord, the third dimension is the class of the chord (0, major triad; 1, dom7; 2, minor triad; 3, fully diminished 7th; 4, half diminished 7th; 5, diminished triad; 6, minor 7th), the fourth dimension is the duration of the chord, and the fifth dimension contains the score as assigned by the HarmAn algorithm (a large weight suggests that the chord was labelled unambiguously—a small weight suggests otherwise).

HarmAn->labelling

Started, last checked	19/4/2011, 19/4/2011
Location	Chord labelling
Calls	max-argmax-of-segment-scores, max-argmax-of-segments-score, my-last, nth-list-of-lists
Called by	HarmAn->, HarmAn->roman
Comments/see also	

Example:

```
(HarmAn->labelling
 '((0 ((0 60 60 2) (0 64 62 2) (0 72 67 1)))
   (1 ((0 60 60 2) (0 64 62 2) (1 74 68 1/2)))
   (3/2 ((0 60 60 2) (0 64 62 2) (3/2 76 69 1/2)))
   (2 ((2 59 59 1) (2 65 63 1) (2 79 71 1)))
   (3 ((3 60 60 1) (3 64 62 1) (3 79 71 1)))))
--> ((((0 ((0 60 60 2) (0 64 62 2) (0 72 67 1)))
       (1 ((0 60 60 2) (0 64 62 2) (1 74 68 1/2)))
       (3/2
        ((0 60 60 2) (0 64 62 2) (3/2 76 69 1/2))))
      (6 0))
     (((2 ((2 59 59 1) (2 65 63 1) (2 79 71 1))))
      (2 19))
     (((3 ((3 60 60 1) (3 64 62 1) (3 79 71 1))))
      (3 0)))
```

This function is a partial implementation of the forwards-running HarmAn algorithm described by Pardo and Birmingham (2002). It is partial in the sense that further functions are required to produce chord datapoints rather than labelled segments (see the function labelled-listed-segments2datapoints) and resolve the ambiguity of diminished 7ths (see the function resolve-dim-7s).

HarmAn->roman

Started, last checked	7/5/2014, 7/5/2014
Location	Chord labelling
Calls	centre-dataset, constant-vector, fifth-steps-mode, HarmAn->labelling, min-argmin, my-last, nth-list-of-lists, orthogonal-projection-unique-equalp, resolve-dim7s-roman, restrict-point-set-to-MNN-mod12, segments-strict
Called by	
Comments/see also	

Example:

```
(HarmAn->roman
 '((0 51 55 1/2 3) (0 58 59 1 2) (0 63 62 1 1)
   (0 67 64 1 0) (1/2 50 54 1/2 3) (1 48 53 1/2 3)
   (1 60 60 1 2) (1 63 62 1 1) (1 68 65 1 0)
   (3/2 51 55 1/2 3) (2 50 54 1/2 3) (2 53 56 1 2)
   (2 65 63 1/2 1) (2 70 66 1 0) (5/2 48 53 1/2 3)
   (5/2 63 62 1/2 1) (3 46 52 1/2 3) (3 55 57 1 2)
   (3 62 61 1 1) (3 70 66 1 0) (7/2 50 54 1/2 3)
   (4 48 53 1/2 3) (4 55 57 1 2) (4 63 62 1 0)
   (4 63 62 1 1) (9/2 46 52 1/2 3) (5 44 51 1/2 3)
   (5 60 60 1/2 2) (5 63 62 1/2 1) (5 65 63 1 0)
   (11/2 46 52 1/2 3) (11/2 58 59 1/2 2)
   (11/2 62 61 1/2 1) (6 39 48 2 3) (6 58 59 2 2)
   (6 63 62 2 1) (6 67 64 2 0))
 *chord-templates-p&b&min7ths*)
--> (("I" (0 1)) ("IVb" (1 2)) ("Vb" (2 5/2))
     ("II7c" (5/2 3)) ("iiib" (3 4)) ("vi7d" (4 5))
     ("ii7b" (5 11/2)) ("V" (11/2 6)) ("I" (6 8))).
```

This function segments and labels chords in some input piece of music. The algorithm is based on an implementation of HarmAn by Pardo and Birmingham (2002). HarmAn compares input triples of ontimes, MIDI note numbers, and durations to predefined chord templates, and performs segmentation and segment labelling on this basis. The labels are absolute, for instance (15 2 1 1 8) means that a chord begins on ontime 15, has root 2 modulo 12 (i.e., D), is of type 1 (dom7 chord), lasts 1 beat, and was assigned to this chord template with strength 8.

While useful, this output does not provide a functional-harmonic analysis. I programmed some extra steps to estimate the overall key of the input piece, using the Krumhansl-Schmuckler key-finding algorithm Krumhansl (1990), and then to caluclate relative (or functional) harmonic labels by combining the estimate of overall key with the absolute labels output by HarmAn->. For instance, if the overall key is G major, and HarmAn-> output the label D dom7, then my code would convert this to V7. I have taken care to make sure the labelling of diminished 7th chords is correct. The overall program is referred to as HarmAn->roman. It does not handle secondary keys, but might be adapted to do so using a slice through a keyscape Sapp (2005).

labelled-listed-segments2datapoints

Started, last checked	19/4/2011, 19/4/2011
Location	Chord labelling
Calls	HarmAn->labelling, labelled-listed-segments2datapoints, resolve-dim7s, segments-strict
Called by	labelled-listed-segments2datapoints
Comments/see also	

Example:

```
(labelled-listed-segments2datapoints
 '((((22
      ((22 50 54 1 3 23 97) (22 57 58 1 2 23 98)
       (22 66 63 1 1 23 99) (22 69 65 1 0 23 100)))
     (23
      ((23 47 52 1 3 24 101) (23 59 59 1 2 24 102)
       (23 66 63 1 1 24 103) (23 74 68 1 0 24 104))))
    (8 74))
   (((24
      ((24 52 55 1/2 3 49/2 105)
       (24 59 59 1/2 2 49/2 106)
       (24 67 64 1/2 1 49/2 107)
```

```
      (24 72 67 1 0 25 108))))
   (2 0))
  (((49/2
     ((49/2 54 56 1/2 3 25 109)
      (49/2 57 58 1/2 2 25 110)
      (49/2 69 65 1 1 51/2 111)
      (24 72 67 1 0 25 108))))
   (4 57))
  (((25
     ((25 55 57 1 3 26 112) (25 59 59 1/2 2 51/2 113)
      (49/2 69 65 1 1 51/2 111)
      (25 71 66 1 0 26 114)))
    (51/2
     ((25 55 57 1 3 26 112) (51/2 60 60 1/2 2 26 115)
      (51/2 67 64 1 1 53/2 116)
      (25 71 66 1 0 26 114)))
    (26
     ((26 50 54 1 3 27 117) (26 62 61 1 2 27 118)
      (51/2 67 64 1 1 53/2 116)
      (26 69 65 1 0 27 119)))
    (53/2
     ((26 50 54 1 3 27 117) (26 62 61 1 2 27 118)
      (53/2 64 62 1/2 1 27 120)
      (26 69 65 1 0 27 119))))
   (8 67))
  (((27
     ((27 51 54 1 3 28 121) (27 60 60 1 2 28 122)
      (27 66 63 1 1 28 123) (27 69 65 1 0 28 124))))
   (4 36))
  (((28
     ((28 52 55 1/2 3 57/2 125) (28 59 59 1 2 29 126)
      (28 67 64 2 1 30 128)
      (28 67 64 1/2 0 57/2 127))))
   (4 28))
  (((57/2
     ((57/2 54 56 1/2 3 29 129) (28 59 59 1 2 29 126)
      (28 67 64 2 1 30 128)
      (57/2 69 65 1/2 0 29 130))))
   (1 23))
  (((29
     ((29 55 57 1/2 3 59/2 131) (29 64 62 1 2 30 132)
```

```
       (28 67 64 2 1 30 128)
       (29 71 66 1/2 0 59/2 133))))
    (4 28))
   (((59/2
      ((59/2 57 58 1/2 3 30 134) (29 64 62 1 2 30 132)
       (28 67 64 2 1 30 128)
       (59/2 72 67 1/2 0 30 135))))
    (4 72))
   (((30
      ((30 59 59 1 3 31 136) (30 62 61 1 2 31 137)
       (30 66 63 1 1 31 138) (30 74 68 1 0 31 139)))
     (31 NIL))
    (4 35))))
--> ((22 11 6 2 8) (24 0 0 1/2 2) (49/2 6 5 1/2 4)
     (25 4 6 2 8) (27 0 3 1 4) (28 4 2 1/2 4)
     (57/2 11 1 1/2 1) (29 4 2 1/2 4) (59/2 9 6 1/2 4)
     (30 11 2 1 4))
```

This function takes labelled listed segments as an argument. It converts these to datapoints where the first dimension is ontime, the second dimension is the MIDI note number modulo 12 of the root of the chord, the third dimension is the class of the chord (0, major triad; 1, dom7; 2, minor triad; 3, fully diminished 7th; 4, half diminished 7th; 5, diminished triad; 6, minor 7th), the fourth dimension is the duration of the chord, and the fifth dimension contains the score as assigned by the HarmAn algorithm (a large weight suggests that the chord was labelled unambiguously—a small weight suggests otherwise).

max-argmax-of-segment-scores

Started, last checked	19/4/2011, 19/4/2011
Location	Chord labelling
Calls	max-argmax, score-segment-against-template
Called by	HarmAn->labelling, minimal-segment-scores
Comments/see also	

Example:

```
(max-argmax-of-segment-scores '(0 3 5 7 5))
```

```
--> (2 17)
```

This function takes a list of MIDI note numbers modulo 12 as its only argument. It scores this list using the function score-segment-against- template, for each chord template in the variable *chord-template*, and returns the index of the chord that produces the maximum score. If there is a tie, the index of the first such chord is returned.

max-argmax-of-segments-score

Started, last checked	19/4/2011, 19/4/2011
Location	Chord labelling
Calls	max-argmax, score-segment-against-template, segments2MNNs-mod12
Called by	HarmAn->labelling
Comments/see also	

Example:

```
(max-argmax-of-segments-score
 '((0 ((0 60 60 2) (0 64 62 2) (0 72 67 1)))
   (1 ((0 60 60 2) (0 64 62 2) (1 74 68 1/2)))
   (3/2 ((0 60 60 2) (0 64 62 2) (3/2 76 69 1/2)))))
--> (6 0)
```

This function a list of segments its only argument. The segments are appended for scoring purposes. A score is given according to the function score-segment-against-template, for each chord template in the variable chord-templates, and the index of the chord that produces the maximum score is returned, as well as the maximum score. If there is a tie, the index of the first such chord is returned.

minimal-segment-scores

Started, last checked	19/4/2011, 19/4/2011
Location	Chord labelling
Calls	max-argmax-of-segment-scores, nth-list-of-lists
Called by	
Comments/see also	Deprecated.

Example:

```
(minimal-segment-scores
 '((0 ((0 60 60 2) (0 64 62 2) (0 72 67 1)))
   (1 ((0 60 60 2) (0 64 62 2) (1 74 68 1/2)))
   (3/2 ((0 60 60 2) (0 64 62 2) (3/2 76 69 1/2)))
   (2 ((2 59 59 1) (2 65 63 1) (2 79 71 1)))
   (3 ((3 60 60 1) (3 64 62 1) (3 79 71 1)))))
--> ((0 ((0 60 60 2) (0 64 62 2) (0 72 67 1)) (2 0))
     (1 ((0 60 60 2) (0 64 62 2) (1 74 68 1/2)) (0 0))
     (3/2
      ((0 60 60 2) (0 64 62 2) (3/2 76 69 1/2)) (2 0))
     (2 ((2 59 59 1) (2 65 63 1) (2 79 71 1)) (2 19))
     (3 ((3 60 60 1) (3 64 62 1) (3 79 71 1)) (3 0)))
```

This function takes a list of MIDI note numbers modulo 12 as its only argument. It scores this list using the function score-segment-against-template, for each chord template in the variable *chord-template*, and returns the index of the chord that produces the maximum score. If there is a tie, the index of the first such chord is returned.

MNN-mod12&class2chord-index

Started, last checked	19/4/2011, 19/4/2011
Location	Chord labelling
Calls	
Called by	
Comments/see also	

Example:

```
(MNN-mod12&class2chord-index '(10 4))
--> 49
(MNN-mod12&class2chord-index '(11 6))
--> 74
```

This function takes a pair consisting of root (MIDI note number modulo 12) and chord class (listed above) as argument. It converts this pair into the index in the variable *chord-templates-p&b&min7ths*. The situation is complicated slightly by the fourth category (of fully-diminished 7th chords), which contains only 3 chords.

resolve-dim7s

Started, last checked	19/4/2011, 19/4/2011
Location	Chord labelling
Calls	
Called by	HarmAn->
Comments/see also	resolve-dim7s-roman

Example:

```
(resolve-dim7s
 '((25 4 6 2 8) (27 0 3 1 4) (28 4 2 1/2 4)))
--> ((25 4 6 2 8) (27 3 3 1 4) (28 4 2 1/2 4))
```

This function takes a chord dataset as an argument, where usually the first dimension is ontime, the second dimension is the MIDI note number modulo 12 of the root of the chord, the third dimension is the class of the chord (0, major triad; 1, dom7; 2, minor triad; 3, fully diminished 7th; 4, half diminished 7th; 5, diminished triad; 6, minor 7th), the fourth dimension is the duration of the chord, and the fifth dimension contains the score as assigned by the HarmAn algorithm (a large weight suggests that the chord was labelled unambiguously—a small weight suggests otherwise). The function searches for any chord datapoints of class 3 (fully diminished). If it finds such a chord datapoint, it looks at the MIDI note number modulo 12 of the subsequent chord, y. If $y - 1 \mod 12$ is a member of the previous chord, then $y - 1 \mod 12$ becomes its root. Otherwise the root is unchanged. Thus, this function resolves the spelling of ambiguous fully diminished 7th chords.

resolve-dim7s-roman

Started, last checked	7/5/2014, 7/5/2014
Location	Chord labelling
Calls	max-argmax, restrict-point-set-to-MNN-mod12
Called by	HarmAn->roman
Comments/see also	resolve-dim7s

Example:

```
(setq
 point-set2
 '((16 -3 -2 2) (17 3 2 1) (17 6 3 1) (17 12 7 4/3)))
```

```
(setq template-MNNs-mod12 '(0 3 6 9))
(setq MNN-mod12-of-lowest-note 9)
(setq fifth-steps-mode '(6 5))
(resolve-dim7s-roman
 point-set2 template-MNNs-mod12
 MNN-mod12-of-lowest-note fifth-steps-mode)
--> "#vio7b".

(setq
 point-set2
 '((19 -5 -3 2) (20 1 1 1) (20 4 2 1) (20 10 6 4/3)))
(setq template-MNNs-mod12 '(1 4 7 10))
(setq MNN-mod12-of-lowest-note 7)
(setq fifth-steps-mode '(6 5))
(resolve-dim7s-roman
 point-set2 template-MNNs-mod12
 MNN-mod12-of-lowest-note
 fifth-steps-mode)
--> "vo7b".
```

This function returns a label for spelling a diminished 7th chord. The morphetic pitch numbers passed to the function (in the third column of each list in the input point set) are central to this spelling process. In the above examples they are expressed relative to a tonic that has MIDI note number 0 and morphetic pitch number 0.

restrict-point-set-to-MNN-mod12

Started, last checked	7/5/2014, 7/5/2014
Location	Chord labelling
Calls	
Called by	HarmAn->roman
Comments/see also	restrict-dataset-in-nth-to-xs

Example:

```
(restrict-point-set-to-MNN-mod12
 '((1 -8 -5 1) (1 -1 -1 1) (1 2 1 1) (1 8 4 1/2)
   (3/2 9 5 1/2) (2 -8 -5 1) (2 -4 -3 1) (2 2 1 1)
   (2 11 6 3/4) (11/4 5 3 1/4)) '(4 8 11 2))
--> ((1 -8 -5 1) (1 -1 -1 1) (1 2 1 1) (1 8 4 1/2)
```

```
    (2 -8 -5 1) (2 -4 -3 1) (2 2 1 1) (2 11 6 3/4)).
```

This function returns only the points from the input point set whose MIDI note numbers belong to the second argument (a list of MIDI note numbers modulo 12).

score-segment-against-template

Started, last checked	19/4/2011, 19/4/2011
Location	Chord labelling
Calls	a-list-in-b-list
Called by	max-argmax-of-segment-scores
Comments/see also	

Example:

```
(score-segment-against-template '(0 1 5 7 5) '(5 9 0))
--> 0
```

This function takes two lists as arguments. The first is a list of MIDI note numbers modulo 12, and the second is a chord template. Three quantities are calcuated: P, the number of MNNs that are members of the chord template; N, the number of MNNs that are not members of the chord template; M, the number of elements of the chord template that are not members of the list of MNNs. The value of $P - (M + N)$ is returned.

segments2MNNs-mod12

Started, last checked	19/4/2011, 19/4/2011
Location	Chord labelling
Calls	mod-list, nth-list-of-lists
Called by	max-argmax-of-segment-scores
Comments/see also	

Example:

```
(segments2MNNs-mod12
 '((0 ((0 60 60 2) (0 64 62 2) (0 72 67 1)))
   (1 ((0 60 60 2) (0 64 62 2) (1 74 68 1/2)))
   (3/2 ((0 60 60 2) (0 64 62 2) (3/2 76 69 1/2)))))
--> (0 4 0 0 4 2 0 4 4)
```

This function takes a list of segments its only argument. The MIDI note numbers of each segment are mapped to modulo 12 and appended into one list.

triad-time-intervals

Started, last checked	19/4/2011, 19/4/2011
Location	Chord labelling
Calls	HarmAn->roman
Called by	**??**
Comments/see also	

Example:

```
(setq
 question-string
 "subdominant triad in first inversion")
(setq
 point-set
 '((0 51 55 1/2 3) (0 58 59 1 2) (0 63 62 1 1)
   (0 67 64 1 0) (1/2 50 54 1/2 3) (1 48 53 1/2 3)
   (1 60 60 1 2) (1 63 62 1 1) (1 68 65 1 0)
   (3/2 51 55 1/2 3) (2 50 54 1/2 3) (2 53 56 1 2)
   (2 65 63 1/2 1) (2 70 66 1 0) (5/2 48 53 1/2 3)
   (5/2 63 62 1/2 1) (3 46 52 1/2 3) (3 55 57 1 2)
   (3 62 61 1 1) (3 70 66 1 0) (7/2 50 54 1/2 3)
   (4 48 53 1/2 3) (4 55 57 1 2) (4 63 62 1 0)
   (4 63 62 1 1) (9/2 46 52 1/2 3) (5 44 51 1/2 3)
   (5 60 60 1/2 2) (5 63 62 1/2 1) (5 65 63 1 0)
   (11/2 46 52 1/2 3) (11/2 58 59 1/2 2)
   (11/2 62 61 1/2 1) (6 39 48 2 3) (6 58 59 2 2)
   (6 63 62 2 1) (6 67 64 2 0)))
(triad-time-intervals question-string point-set)
--> ((1 2))
```

This function takes a string and a point set as its compulsory arguments, where the string may refer to a triad. It returns the time intervals in the point set where the triad occurs.

triad-inversion-time-intervals

Started, last checked	19/4/2011, 19/4/2011
Location	Chord labelling
Calls	HarmAn->roman
Called by	??
Comments/see also	

Example:

```
(setq
 question-string
 "triad in first inversion")
(setq
 point-set
 '((0 51 55 1/2 3) (0 58 59 1 2) (0 63 62 1 1)
   (0 67 64 1 0) (1/2 50 54 1/2 3) (1 48 53 1/2 3)
   (1 60 60 1 2) (1 63 62 1 1) (1 68 65 1 0)
   (3/2 51 55 1/2 3) (2 50 54 1/2 3) (2 53 56 1 2)
   (2 65 63 1/2 1) (2 70 66 1 0) (5/2 48 53 1/2 3)
   (5/2 63 62 1/2 1) (3 46 52 1/2 3) (3 55 57 1 2)
   (3 62 61 1 1) (3 70 66 1 0) (7/2 50 54 1/2 3)
   (4 48 53 1/2 3) (4 55 57 1 2) (4 63 62 1 0)
   (4 63 62 1 1) (9/2 46 52 1/2 3) (5 44 51 1/2 3)
   (5 60 60 1/2 2) (5 63 62 1/2 1) (5 65 63 1 0)
   (11/2 46 52 1/2 3) (11/2 58 59 1/2 2)
   (11/2 62 61 1/2 1) (6 39 48 2 3) (6 58 59 2 2)
   (6 63 62 2 1) (6 67 64 2 0)))
(triad-inversion-time-intervals
 question-string point-set)
--> ((1 2) (2 5/2) (3 4) (5 11/2))
```

This function takes a string and a point set as its compulsory arguments, where the string may refer to a type of triad inversion. It returns the time intervals in the point set where the type of triad inversion occurs.

4.7.3 Inner metric analysis

Implementation of Inner Metric Analysis as described by Volk (2008). Inner Metric Analysis is based the concept of a local metre, which is defined as a set of ‘equally spaced onsets…[that] contains at least three onsets and is maximal, meaning that it is not a subset of any other subset consisting of equally distanced onsets' (Volk, 2008, p. 261).

I noticed a relationship between local metres and maximal translatable patterns, which is exploited in this implementation. As the latter are coded for datasets of dimension greater than or equal to one (not just the set of onsets), it is possible to calculate local metres in more than one dimension using the functions below.

It should be noted that this implementation requires improvement, as some of the output metres are not local metres: consider for example, the metres ((8)(2)3(2)) and ((4)(2)7(2)). The first is a subset of the second, so the first is not a local metre. But both will be output here, because they are generated by different MTPs (MTP of 8 and 4 respectively).

expand-local-metre

Started, last checked	28/4/2011, 28/4/2011
Location	Inner metric analysis
Calls	add-two-lists, multiply-list-by-constant
Called by	general-metric-weights
Comments/see also	

Example:

```
(expand-local-metre '((4) (2) 7 (2)))
--> '((2) (6) (10) (14) (18) (22) (26) (30))
```

This function expands a local metre (Volk, 2008) into a list of member ontimes. The format of the local metre is: period vector (or d in Volk's 2008); first member (or s); length in the MTP (or k); a list of phases (or ph). In this explanation, the term onsets has been used, but in fact this code generalises to multiple-dimension local metres.

general-metric-weight

Started, last checked	28/4/2011, 28/4/2011
Location	Inner metric analysis
Calls	fibonacci-list, my-last
Called by	general-metric-weights
Comments/see also	

Example:

```
(setq
 expanded-local-metres
 '(((0) (1) (2)) ((8) (9) (10)) ((16) (17) (18))
   ((24) (25) (26)) ((6) (8) (10)) ((14) (16) (18))
   ((22) (24) (26))
   ((2) (6) (10) (14) (18) (22) (26) (30))
   ((2) (8) (14)) ((10) (16) (22))
   ((18) (24) (30)) ((2) (9) (16)) ((10) (17) (24))
   ((0) (8) (16) (24)) ((1) (9) (17) (25))
   ((2) (10) (18) (26)) ((6) (14) (22) (30))
   ((0) (9) (18)) ((8) (17) (26)) ((6) (16) (26))
   ((2) (14) (26)) ((6) (18) (30)) ((2) (16) (30))))
(general-metric-weight '(9) expanded-local-metres)
--> 21
(general-metric-weight '(10) expanded-local-metres)
--> 74
```

This function calculates the general metric weight (Volk, 2008) of an onset in a set of onsets. It is the sum over all local metres (of minimum length specifiable by l) of which the onset is a member, and the summand is κ^p, where κ is the length of the local metre in question, and $p = 2$ is a parameter. In this explanation, the term onsets has been used, but in fact this code generalises to multiple- dimension local metres.

general-metric-weights

Started, last checked	28/4/2011, 28/4/2011
Location	Inner metric analysis
Calls	general-metric-weight, local-metres-via-SIA, orthogonal-projection-unique-equalp, read-from-file
Called by	
Comments/see also	

Example:

```
Dataset defined as in the function local-metres-via-SIA.
(setq
 fpath
 (merge-pathnames
  (make-pathname
```

```
   :directory '(:relative "Example files"))
  *MCStylistic-Aug2013-functions-path*))
(setq fname "schubertOp94No4")
(general-metric-weights dataset fpath fname)
--> (((0) (1) (2) (6) (8) (9) (10) (14) (16) (17) (18)
      (22) (24) (25) (26) (30))
     (17 13 78 70 25 21 74 70 33 21 78 66 25 13 78 70))
```

This function calculates the general metric weights (Volk, 2008) of onsets in a dataset. It returns two lists: one containing the onsets, and the other the corresponding metric weights. Please see the functions local-metres-via-SIA and general-metric-weight for more details. The default parameters of minimum length $l = 2$ and exponent $p = 2$ are as in Volk (2008). In this explanation, the term onsets has been used, but in fact this code generalises to multiple-dimension local metres.

local-metres-via-SIA

Started, last checked	28/4/2011, 28/4/2011
Location	Inner metric analysis
Calls	orthogonal-projection-unique-equalp, read-from-file, SIA-reflected-merge-sort, vector-MTP-pairs2local-metres, write-to-file
Called by	general-metric-weights
Comments/see also	

Example:

```
(setq
 dataset
 '((0 49 54 2 0) (0 56 58 2 0) (0 61 61 2 0)
   (0 65 63 1 1) (1 68 65 1 1) (2 49 54 4 0)
   (2 56 58 4 0) (2 61 61 4 0) (2 66 64 4 1)
   (2 70 66 4 1) (6 49 54 2 0) (6 56 58 2 0)
   (6 61 61 2 0) (6 65 63 2 1) (6 68 65 2 1)
   (8 49 54 2 0) (8 56 58 2 0) (8 61 61 2 0)
   (8 65 63 1 1) (9 68 65 1 1) (10 49 54 4 0)
   (10 56 58 4 0) (10 61 61 4 0) (10 66 64 4 1)
   (10 70 66 4 1) (10 73 68 4 1) (14 49 54 2 0)
   (14 56 58 2 0) (14 61 61 2 0) (14 65 63 2 1)
   (14 68 65 2 1) (16 56 58 2 0) (16 60 60 2 0)
```

```
   (16 65 63 1 1) (16 68 65 1 1) (17 62 61 1 1)
   (18 56 58 4 0) (18 60 60 4 0) (18 63 62 4 1)
   (18 66 64 4 1) (22 56 58 2 0) (22 60 60 2 0)
   (22 66 64 2 1) (22 75 69 2 1) (24 49 54 2 0)
   (24 56 58 2 0) (24 61 61 2 0) (24 68 65 1 1)
   (24 77 70 1 1) (25 66 64 1 1) (25 75 69 1 1)
   (26 49 54 4 0) (26 56 58 4 0) (26 61 61 4 0)
   (26 65 63 4 1) (26 73 68 4 1) (30 49 54 2 0)
   (30 56 58 2 0) (30 61 61 2 0) (30 65 63 2 1)
   (30 68 65 2 1)))
(setq
 fpath
 (merge-pathnames
  (make-pathname
   :directory '(:relative "Example files"))
  *MCStylistic-Aug2013-functions-path*))
(setq fname "schubertOp94No4")
(local-metres-via-SIA dataset fpath fname)
--> text file containing local metres in specified
location:
(((1) (0) 2 (0)) ((1) (8) 2 (0)) ((1) (16) 2 (0))
 ((1) (24) 2 (0)) ((2) (6) 2 (0)) ((2) (14) 2 (0))
 ((2) (22) 2 (0)) ((4) (2) 7 (2)) ((6) (2) 2 (2))
 ((6) (10) 2 (4)) ((6) (18) 2 (0)) ((7) (2) 2 (2))
 ((7) (10) 2 (3)) ((8) (0) 3 (0)) ((8) (1) 3 (1))
 ((8) (2) 3 (2)) ((8) (6) 3 (6)) ((9) (0) 2 (0))
 ((9) (8) 2 (8)) ((10) (6) 2 (6)) ((12) (2) 2 (2))
 ((12) (6) 2 (6)) ((14) (2) 2 (2)))
```

This function returns a list of local metres (Volk, 2008) for a dataset under a specified projection. Each local metre that has a length k of greater than or equal to the optional variable min-length is returned as a list in the format: period vector (or d in Volk's 2008 paper); first member (or s); length in the MTP (or k); a list of phases (or ph). A relationship between local metres and maximal translatable patterns was observed, hence the use of SIA.

NB Comparison with Volk (2008) suggests that four local metres are missing from the Schubert example: ((8) (2) 3 (2)), ((8) (6) 3 (6)), ((12) (2) 2 (2)), and ((12) (6) 2 (6)).

normalise-metric-weights-by-quartiles

Started, last checked	28/4/2011, 28/4/2011
Location	Inner metric analysis
Calls	orthogonal-projection-unique-equalp, read-from-file, SIA-reflected-merge-sort, vector-MTP-pairs2local-metres, write-to-file
Called by	general-metric-weights
Comments/see also	

Example:

```
(setq
 metric-weights
 '(17 13 78 70 25 21 74 70 33 21 78 66 25 13 78 70))
(normalise-metric-weights-by-quartiles metric-weights)
--> (1 1 4 3 2 1 4 3 2 1 4 3 2 1 4 3)
```

This function takes a list (of metric weights) as input. It calculates the lower, median, and upper quartiles of the list. Then it outputs a list of the same length as the argument, where each metric weight has been assigned a number 1-4, indicating whether it is in the min-lower, lower-median, median-upper, or median- max quartile.

vector-MTP-pair2local-metres

Started, last checked	28/4/2011, 28/4/2011
Location	Inner metric analysis
Calls	add-two-lists, vector-MTP-pair2local-metres
Called by	vector-MTP-pairs2local-metres
Comments/see also	

Example:

```
(vector-MTP-pair2local-metres
 '((2) (0) (6) (7) (8) (9) (11) (14) (16) (22) (24)))
--> '(((2) (6) 2 (0)) ((2) (7) 3 (1)) ((2) (14) 2 (0))
      ((2) (22) 2 (0)))
(vector-MTP-pair2local-metres
 '((2 1) (0 54) (6 60) (8 61) (14 60) (16 61) (22 60)
   (24 61)))
```

```
--> '(((2 1) (6 60) 2 (0 0)) ((2 1) (14 60) 2 (0 0))
      ((2 1) (22 60) 2 (0 0))
```

This function takes a vector-MTP pair as its only mandatory argument. It parses the MTP for local metres. Each local metre that has a length of greater than or equal to the optional variable min-length is returned as a list in the format: period vector (or d in Volk's 2008 paper); first member (or s); length in the MTP (or k); a list of phases (or ph).

vector-MTP-pairs2local-metres

Started, last checked	28/4/2011, 28/4/2011
Location	Inner metric analysis
Calls	vector-MTP-pair2local-metres
Called by	local-metres-via-SIA
Comments/see also	

Example:

```
(vector-MTP-pairs2local-metres
 '(((2) (0) (6) (7) (8) (9) (11) (14) (16) (22) (24))
   ((6) (0) (2) (8) (10) (16) (18) (24))))
--> '(((2) (6) 2 (0)) ((2) (7) 3 (1)) ((2) (14) 2 (0))
      ((2) (22) 2 (0)) ((6) (2) 2 (2))
      ((6) (10) 2 (4)) ((6) (18) 2 (0)))
```

This function applies the function vector- MTP-pair2local-metres recursively to members of a list. The output is a flat list, in the sense that successive applications are appended.

4.7.4 Keyscape

An implementation of the calculation of keyscapes as described by Sapp (2005), based on key profiles described by Krumhansl and Kessler (1982) and Aarden (2003). Not currently able to depict the calculated keyscapes. The function 4.7.4 is an implementation of the Krumhansl-Schmuckler key-finding algorithm (Krumhansl, 1990).

accumulate-to-weighted-mass

Started, last checked	26/4/2011, 26/4/2011
Location	Keyscape
Calls	my-last
Called by	present-to-weighted-mass
Comments/see also	

Example:

```
(accumulate-to-weighted-mass
 '(4 0 1) '((4 0) 7) '(((0 4) 3) ((4 0) 7)))
--> (((4 0) 8) ((0 4) 3))
```

This function takes three arguments: a datapoint d (the last dimension of which is a weighting); an element (to be updated) of the emerging empirical probability mass function p; p itself. The weighting of the observation is added to the existing mass.

add-to-weighted-mass

Started, last checked	26/4/2011, 26/4/2011
Location	Keyscape
Calls	my-last
Called by	present-to-weighted-mass
Comments/see also	

Example:

```
(add-to-weighted-mass '(3 1 6) '(((0 4) 3) ((4 0) 7)))
--> (((6 72) 1/3) ((4 0.1) 2/3))
```

This function takes two arguments: a datapoint d (the last dimension of which is a weighting), and an emerging empirical probability mass function p. The observation and its weighting is is added to the existing mass.

datapoints-sounding-between

Started, last checked	26/4/2011, 26/4/2011
Location	Keyscape
Calls	my-last
Called by	keyscape-list
Comments/see also	points-sounding-at

Example:

```
(datapoints-sounding-between
 '((5/2 72 66 1/2 0 3) (3 42 49 3 1 6)
   (3 74 68 1/3 0 10/3) (10/3 76 69 1/3 0 11/3)
   (11/3 74 68 1/3 0 4) (4 57 58 1 1 5)
   (4 61 60 1 1 5)) 3 4)
--> ((3 42 49 3 1 4 1) (3 74 68 1/3 0 10/3 1/3)
     (10/3 76 69 1/3 0 11/3 1/3)
     (11/3 74 68 1/3 0 4 1/3))
```

A list of datapoints with offtimes appended is the first argument to this function. The second argument is the ontime of a window, a, and the third argument is the offtime of the same window, b. A datapoint appears in the output of the function if it sounds during the window [a b). The amount of time for which it sounds in [a b) is appended.

dataset2pcs-norm-tonic

Started, last checked	26/4/2011, 26/4/2011
Location	Keyscape
Calls	key-correlations, max-argmax, nth-list-of-lists
Called by	
Comments/see also	

Example:

```
(dataset2pcs-norm-tonic
 '((3 42 49 3 1) (3 74 68 1/3 0) (10/3 76 69 1/3 0)
   (11/3 74 68 1/3 0)))
--> (7 3 5 3)
```

This function estimates the key of the input dataset. It subtracts the tonic pitch class from each input MIDI note number, and outputs the answer modulo twelve.

fifth-steps-mode

Started, last checked	2/1/2013, 2/1/2013
Location	Keyscape
Calls	key-correlations, max-argmax, nth-list-of-lists
Called by	beat-rel-MNN-states
Comments/see also	

Example:

```
(setq
 relevant-datapoints
 '((-1 72 67 7/4 0) (0 55 57 1 1) (0 61 61 1 1)
   (0 64 63 1 1) (3/4 70 66 1/4 0) (1 56 58 1 1)
   (1 60 60 2 1) (1 63 62 2 1) (1 68 65 1/2 0)
   (3/2 70 66 1/2 0) (2 51 55 1 1) (2 72 67 7/4 0)
   (3 55 57 1 1) (3 61 61 1 1) (3 64 63 1 1)
   (15/4 70 66 1/4 0) (4 56 58 1 1) (4 60 60 2 1)
   (4 63 62 2 1) (4 68 65 1/2 0) (9/2 77 70 1/2 0)
   (5 51 55 1 1) (5 75 69 1 0) (6 55 57 1 1)
   (6 61 61 1 1) (6 64 63 1 1) (6 72 67 3/4 0)
   (27/4 70 66 1/4 0) (7 56 58 1 1) (7 60 60 2 1)
   (7 63 62 2 1) (7 68 65 1/2 0) (15/2 70 66 1/2 0)
   (8 51 55 1 1) (8 72 67 1 0) (9 56 58 3/4 1)
   (9 61 61 3 1) (9 63 62 3 0) (39/4 55 57 1/4 1)
   (10 53 56 1/2 1) (21/2 55 57 1/2 1) (11 51 55 1 1)
   (12 55 57 1 1) (12 61 61 1 1) (12 64 63 1 1)
   (12 72 67 3/4 0)))
(fifth-steps-mode relevant-datapoints)
--> (-4 0)
```

This function returns the key of input datapoints, in the format of steps on the cycle of fifths (e.g., -1 for F major) and mode (e.g., 5 for Aeolian).

key-correlations

Started, last checked	26/4/2011, 26/4/2011
Location	Keyscape
Calls	weighted-empirical-mass, weighted-mass2key-profile
Called by	dataset2pcs-norm-tonic, normalised-key-correlations
Comments/see also	

Example:

```
(setq
 relevant-datapoints
 '((3 42 49 3 1 4 1) (3 74 68 1/3 0 10/3 1/3)
   (10/3 76 69 1/3 0 11/3 1/3)
   (11/3 74 68 1/3 0 4 1/3)))
(key-correlations relevant-datapoints)
--> (("C major" 0.0056350203) ("Db major" -0.16437216)
     ("D major" 0.6365436) ("Eb major" -0.41964883)
     ("E major" 0.1384299) ("F major" -0.30088827)
     ("Gb major" 0.040827263) ("G major" 0.18202147)
     ("Ab major" -0.51650715) ("A major" 0.2044552)
     ("Bb major" -0.10270603) ("B major" 0.29621017)
     ("C minor" -0.23124762) ("C# minor" 0.14107251)
     ("D minor" 0.03225725) ("Eb minor" 0.058523033)
     ("E minor" 0.30459806) ("F minor" -0.51036686)
     ("F# minor" 0.25775552) ("G minor" -0.067237906)
     ("G# minor" -0.2650179) ("A minor" 0.039761756)
     ("Bb minor" -0.41389754) ("B minor" 0.65379965))
```

This function takes datapoints as its only default argument. These datapoints have already had offtimes appended, and have passed a test for membership of the time window [a b). Their duration within [a b) is given as the final dimension. The weighted empirical mass of these datapoints is calculated, using the projection on to MIDI note number mod 12, weighted by duration within [a b). This mass is converted to a key profile (vector), and the pairwise correlations between this probe profile and various other profiles contained in the optional argument key-profiles are returned.

normalised-key-correlations

Started, last checked	26/4/2011, 26/4/2011
Location	Keyscape
Calls	add-to-list, fibonacci-list, key-correlations, multiply-list-by-constant, my-last, nth-list-of-lists
Called by	keyscape-list
Comments/see also	

Example:

```
(setq
 relevant-datapoints
 '((3 42 49 3 1 4 1) (3 74 68 1/3 0 10/3 1/3)
   (10/3 76 69 1/3 0 11/3 1/3)
   (11/3 74 68 1/3 0 4 1/3)))
(normalised-key-correlations relevant-datapoints)
--> (("C major" 0.04212125) ("Db major" 0.028406756)
     ("D major" 0.09301668) ("Eb major" 0.007813568)
     ("E major" 0.052833818) ("F major" 0.017393991)
     ("Gb major" 0.04496021) ("G major" 0.056350354)
     ("Ab major" 0.0) ("A major" 0.058160085)
     ("Bb major" 0.033381365) ("B major" 0.065561965)
     ("C minor" 0.02301191) ("C# minor" 0.053046998)
     ("D minor" 0.044268865) ("Eb minor" 0.04638773)
     ("E minor" 0.06623862) ("F minor" 4.9533776E-4)
     ("F# minor" 0.06245983) ("G minor" 0.036242582)
     ("G# minor" 0.020287657) ("A minor" 0.044874255)
     ("Bb minor" 0.008277524)
     ("B minor" 0.094408736))
```

The output of the function key-correlations is converted to a probability vector.

keyscape-list

Started, last checked	26/4/2011, 26/4/2011
Location	Keyscape
Calls	append-offtimes, datapoints-sounding-between, max-item, normalised-key-correlations, nth-list-of-lists
Called by	
Comments/see also	

Example:

```
(setq
 dataset
 '((-1 61 60 1 0) (0 30 42 1 1) (0 66 63 3/2 0)
   (1 49 53 1 1) (1 57 58 1 1) (1 61 60 1 1)
   (3/2 68 64 1/2 0) (2 49 53 1 1) (2 57 58 1 1)
   (2 61 60 1 1) (2 69 65 1/2 0) (5/2 72 66 1/2 0)
   (3 42 49 1 1) (3 74 68 1/3 0) (10/3 76 69 1/3 0)
   (11/3 74 68 1/3 0) (4 57 58 1 1) (4 61 60 1 1)
   (4 66 63 1 1) (4 73 67 1 0) (5 57 58 1 1)
   (5 61 60 1 1) (5 66 63 1 1) (5 78 70 1 0)
   (6 54 56 1 1) (6 73 67 1/3 0) (19/3 74 68 1/3 0)
   (20/3 73 67 1/3 0) (7 59 59 1 1) (7 62 61 1 1)
   (7 66 63 1 1) (7 71 66 1 0) (8 59 59 1 1)
   (8 62 61 1 1) (8 66 63 1 1) (8 78 70 1 0)))
(keyscape-list dataset *Aarden-key-profiles* 3 1 3)
--> ((-3 3
      ((-1 61 60 1 0 0 1))
      (("C major" 0.0024341724) ...) (0.10747497 6))
     (0 3
      ((0 30 42 1 1 1 1) (0 66 63 3/2 0 3/2 3/2)
       (1 49 53 1 1 2 1) (1 57 58 1 1 2 1)
       (1 61 60 1 1 2 1) (3/2 68 64 1/2 0 2 1/2)
       (2 49 53 1 1 3 1) (2 57 58 1 1 3 1)
       (2 61 60 1 1 3 1) (2 69 65 1/2 0 5/2 1/2)
       (5/2 72 66 1/2 0 3 1/2))
      (("C major" 0.010985555) ...) (0.09890273 18))
     (3 3
      ((3 42 49 1 1 4 1) (3 74 68 1/3 0 10/3 1/3)
       (10/3 76 69 1/3 0 11/3 1/3)
```

```
  (11/3 74 68 1/3 0 4 1/3) (4 57 58 1 1 5 1)
  (4 61 60 1 1 5 1) (4 66 63 1 1 5 1)
  (4 73 67 1 0 5 1) (5 57 58 1 1 6 1)
  (5 61 60 1 1 6 1) (5 66 63 1 1 6 1)
  (5 78 70 1 0 6 1))
 (("C major" 0.015585414) ...) (0.09317962 18))
(6 3
 ((6 54 56 1 1 7 1) (6 73 67 1/3 0 19/3 1/3)
  (19/3 74 68 1/3 0 20/3 1/3)
  (20/3 73 67 1/3 0 7 1/3) (7 59 59 1 1 8 1)
  (7 62 61 1 1 8 1) (7 66 63 1 1 8 1)
  (7 71 66 1 0 8 1) (8 59 59 1 1 9 1)
  (8 62 61 1 1 9 1) (8 66 63 1 1 9 1)
  (8 78 70 1 0 9 1))
 (("C major" 0.024908373) ...) (0.09740843 23))
(-3 4
 ((-1 61 60 1 0 0 1) (0 30 42 1 1 1 1)
  (0 66 63 3/2 0 3/2 1))
 (("C major" 0.002813767) ...) (0.09659196 18))
(0 4 ((0 30 42 1 1 1 1) ...)
 (("C major" 0.015838815) ...) (0.09398684 18))
(3 4 ((3 42 49 1 1 4 1) ...)
 (("C major" 0.014643886) ...) (0.09295517 18))
(-3 5 ((-1 61 60 1 0 0 1) ...)
 (("C major" 0.005957871) ...) (0.09737477 18))
(0 5 ((0 30 42 1 1 1 1) ...)
 (("C major" 0.014239526) ...) (0.09530763 18))
(3 5 ((3 42 49 1 1 4 1) ...)
 (("C major" 0.018060159) ...) (0.088710114 23))
(-3 6 ((-1 61 60 1 0 0 1) ...)
 (("C major" 0.009841155) ...) (0.099177815 18))
(0 6 ((0 30 42 1 1 1 1) ...)
 (("C major" 0.013386026) ...) (0.096429521 18))
(3 6 ((3 42 49 1 1 4 1) ...)
 (("C major" 0.019494385) ...) (0.09097412 23))
(-3 7 ((-1 61 60 1 0 0 1) ...)
 (("C major" 0.014419735) ...) (0.09467592 18))
(0 7 ((0 30 42 1 1 1 1) ...)
 (("C major" 0.013051211) ...) (0.09554448 18))
(-3 8 ((-1 61 60 1 0 0 1) ...)
 (("C major" 0.013291403) ...) (0.09573281 18))
```

```
    (0 8 ((0 30 42 1 1 1 1) ...)
     (("C major" 0.015427576) ...) (0.0922521 18))
    (-3 9 ((-1 61 60 1 0 0 1) ...)
     (("C major" 0.012677101) ...) (0.09622239 18))
    (0 9 ((0 30 42 1 1 1 1) ...)
     (("C major" 0.01631804) ...) (0.09062016 18))
    (-3 10 ((-1 61 60 1 0 0 1) ...)
     (("C major" 0.012418479) ...) (0.09584213 18))
    (-3 11 ((-1 61 60 1 0 0 1) ...)
     (("C major" 0.014786912) ...) (0.09264141 18))
    (-3 12 ((-1 61 60 1 0 0 1) ...)
     (("C major" 0.015724108) ...) (0.09101982 18)))
```

Implementation of keyscapes as described by Sapp (2005).

points-sounding-at

Started, last checked	14/1/2015, 14/1/2015
Location	Keyscape
Calls	
Called by	**??**, **??**, **??**
Comments/see also	datapoints-sounding-between

Example:

```
(points-sounding-at
 '((5/2 42 49 3 1 11/2) (5/2 72 66 1/2 0 3)
   (3 74 68 1/3 0 10/3) (10/3 76 69 1/3 0 11/3)
   (11/3 74 68 1/3 0 4) (4 57 58 1 1 5)
   (4 61 60 1 1 5)) 3)
--> ((5/2 42 49 3 1 11/2) (3 74 68 1/3 0 10/3)).
```

A list of points with offtimes appended is the first argument to this function. The second argument is a time, t. A point appears in the output of the function if it sounds at t, meaning its ontime x and offtime y satisfy $x \leq t$ and $y > t$.

present-to-weighted-mass

Started, last checked	26/4/2011, 26/4/2011
Location	Keyscape
Calls	accumulate-to-weighted-mass, add-to-weighted-mass
Called by	weighted-empirical-mass
Comments/see also	

Example:

```
(present-to-weighted-mass '(0 4 3) '(((4 0) 7)))
--> (((0 4) 3) ((4 0) 7))
```

This function takes two arguments: a datapoint d (the last dimension of which contains a weighting), and an unnormalised empirical probability mass function p which is in the process of being calculated. If d is new to the empirical mass, it is added with mass given by its weight, and if it already forms part of the mass, then this component is increased by its weight.

weighted-empirical-mass

Started, last checked	26/4/2011, 26/4/2011
Location	Keyscape
Calls	present-to-weighted-mass
Called by	key-correlations
Comments/see also	

Example:

```
(weighted-empirical-mass
 '((4 0 5) (4 0 2) (0 4 3)) '())
--> (((0 4) 3) ((4 0) 7))
```

This function returns the weighted empirical probability mass function p for a dataset listed d1*, d2*,..., dN*, where the last dimension of each datapoint is the weighting.

weighted-mass2key-profile

Started, last checked	26/4/2011, 26/4/2011
Location	Keyscape
Calls	
Called by	key-correlations
Comments/see also	

Example:

```
(weighted-mass2key-profile '(((0) 3) ((7) 4)))
--> (3 0 0 0 0 0 0 4 0 0 0 0)
```

This function converts an unnormalised probability mass function to a twelve-point vector, with the weight from the mass function corresponding to the ith MIDI note number mod 12 appearing as the ith element of the vector.

4.7.5 Neo-Riemannian operations

The functions below are implementations of neo-Riemannian operations (Lewin, 2007; Cohn, 1998). These include leading-note, relative, and parallel operations). At present they do not take correct pitch spelling into account. For instance, the operations 'LPL' and 'PLP' map C major to G♯ minor and A♭ minor respectively, but both will be represented by pitch class 8.

chord-pair2NRO-string

Started, last checked	24/2/2013, 24/2/2013
Location	Neo-Riemannian operations
Calls	
Called by	chord-pairs2NRO-strings
Comments/see also	chord-pairs2NRO-strings, NRO-string2chord-pairs, NRO-strings2chord-pairs

Example:

```
(chord-pair2NRO-string '(3 2) '(4 0))
--> "LRL"
(chord-pair2NRO-string '(7 0) '(2 0))
--> "LR"
```

```
(chord-pair2NRO-string '(3 2) '(10 2))
--> "RL"
(chord-pair2NRO-string '(0 0) '(5 2))
--> "RLP"
```

This function takes two chord pairs as arguments. It returns the neo-Riemannian operation(s) that transform the first chord into the second chord, as a string. It will return the shortest such string for this transformation, and if there are two of equal length, it will return one arbitrarily. The format of each input chord is a two-element list, with first element for the MIDI note number modulo 12 of the root of the chord, and the second element the class of the chord (0, major triad; 2, minor triad).

chord-pairs2NRO-strings

Started, last checked	24/2/2013, 24/2/2013
Location	Neo-Riemannian operations
Calls	chord-pairs2NRO-strings
Called by	
Comments/see also	chord-pair2NRO-string, NRO-string2chord-pairs, NRO-strings2chord-pairs

Example:

```
(chord-pairs2NRO-strings '((3 2) (4 0) (7 0) (2 0)))
--> ("LRL" "PR" "LR")
```

This function takes a list of chord pairs as its argument (first element for the MIDI note number modulo 12 of the root of the chord, and the second element the class of the chord: 0, major triad; 2, minor triad). It returns a list of strings of length $n - 1$, where n is the length of the input list. The strings are the neo-Riemannian operation(s) that transform chord i into chord $i + 1$. It will return the shortest such string for each transformation, and if there are two of equal length, it will return one arbitrarily. The format of each input chord is a two-element list, with first element for the MIDI note number modulo 12 of the root of the chord, and the second element the class of the chord (0, major triad; 2, minor triad).

NRO-L

Started, last checked	24/2/2013, 24/2/2013
Location	Neo-Riemannian operations
Calls	
Called by	NRO-string2chord-pairs
Comments/see also	NRO-P, NRO-R

Example:

```
(NRO-L '(3 2))
--> (11 0)
(NRO-L '(11 0))
--> (3 2)
```

This function applies the neo-Riemannian leading-note operation to an input chord pair, represented as a two-element list, with first element for the MIDI note number modulo 12 of the root of the chord, and the second element the class of the chord (0, major triad; 2, minor triad).

NRO-P

Started, last checked	24/2/2013, 24/2/2013
Location	Neo-Riemannian operations
Calls	
Called by	NRO-string2chord-pairs
Comments/see also	NRO-L, NRO-R

Example:

```
(NRO-P '(3 2))
--> (3 0)
(NRO-P '(3 0))
--> (3 2)
```

This function applies the neo-Riemannian parallel operation to an input chord pair, represented as a two-element list, with first element for the MIDI note number modulo 12 of the root of the chord, and the second element the class of the chord (0, major triad; 2, minor triad).

NRO-R

Started, last checked	24/2/2013, 24/2/2013
Location	Neo-Riemannian operations
Calls	
Called by	NRO-string2chord-pairs
Comments/see also	NRO-L, NRO-P

Example:

```
(NRO-R '(3 2))
--> (6 0)
(NRO-R '(6 0))
--> (3 2)
```

This function applies the neo-Riemannian relative operation to an input chord pair, represented as a two-element list, with first element for the MIDI note number modulo 12 of the root of the chord, and the second element the class of the chord (0, major triad; 2, minor triad).

NRO-string2chord-pairs

Started, last checked	24/2/2013, 24/2/2013
Location	Neo-Riemannian operations
Calls	NRO-L, NRO-P, NRO-R
Called by	NRO-strings2chord-pairs
Comments/see also	chord-pair2NRO-string, chord-pairs2NRO-strings, NRO-strings2chord-pairs

Example:

```
(NRO-string2chord-pairs "PRP" '(0 0))
--> ((0 0) (0 2) (3 0) (3 2))
```

This function applies the neo-Riemannian operations specified as a string in the first argument to an input chord pair (optional second argument), represented as a two-element list, with first element for the MIDI note number modulo 12 of the root of the chord, and the second element the class of the chord (0, major triad; 2, minor triad). The first element of the string is applied first. The first, all intermediate, and last chords are returned.

NRO-strings2chord-pairs

Started, last checked	24/2/2013, 24/2/2013
Location	Neo-Riemannian operations
Calls	my-last, NRO-string2chord-pairs
Called by	
Comments/see also	chord-pair2NRO-string, chord-pairs2NRO-strings, NRO-string2chord-pairs

Example:

```
(NRO-strings2chord-pairs '("PRP" "LR" "PRPL") '(1 0))
--> ((1 0) (4 2) (9 2) (10 2))
(NRO-strings2chord-pairs '("R" "L" "LRLR") '(0 0))
--> ((0 0) (9 2) (5 0) (7 0))
```

This function applies the neo-Riemannian operations specified as strings in the first argument to an input chord pair (optional second argument), represented as a two-element list, with first element for the MIDI note number modulo 12 of the root of the chord, and the second element the class of the chord (0, major triad; 2, minor triad). The first element of the string is applied first. There is a choice (optional third argument) between returning all intermediary chords, or just the final result of each NRO (default).

4.8 Query staff notation

Code written for participating in the C@merata task of MediaEval 2014, in which a natural language noun phrase (question) and staff notation in MusicXML format are provided, and the task is to output time windows that correspond to whatever is specified in the question, e.g., “a quaver, then a major third”.

4.8.1 C@merata processing

Top-level functions for the MediaEval 2014 C@merata task.

Stravinsqi-Jun2014

Started, last checked	17/6/2014, 17/6/2014
Location	C@merata processing
Calls	append-ontimes-to-time-signatures, articulation&event-time-intervals, c@merata2014-question-file2question-string, c@merata2014-write-answer, cadence-time-intervals, cross-check-compound-questions, duration-time-intervals, duration&pitch-class-time-intervals, followed-by-splitter, harmonic-interval-of-a, index-item-1st-doesnt-occur, kern-file2dataset-by-col, kern-file2ontimes-signatures, kern-file2tie-set-by-col, nadir-apex-time-intervals, pitch-class-time-intervals, melodic-interval-of-a, rest-duration-time-intervals, staves-info2staff&clef-names, texture-time-intervals, tied&event-time-intervals, triad-time-intervals, triad-inversion-time-intervals, word-time-intervals, word&event-time-intervals
Called by	
Comments/see also	

Example:

```
(setq question-number "004")
(setq
 question-path&name
 (merge-pathnames
  (make-pathname
   :directory
   '(:relative
```

```
     "C@merata2014" "training_v1")
   :name "training_v1" :type "xml")
  *MCStylistic-MonthYear-data-path*))
(setq
 notation-path
 (merge-pathnames
  (make-pathname
   :directory
   '(:relative
     "C@merata2014" "training_v1"))
  *MCStylistic-MonthYear-data-path*))
(setq notation-name "f3")
(Stravinsqi-Jun2014
 question-number question-path&name
 notation-path notation-name)
--> (wrote the following to text file)
  <question number="004" music_file="f3.xml" divisions="4">
    <text>"F# followed by a major sixth"</text>
    <answer>
      <passage start_beats="3" start_beat_type="8"
               end_beats="3" end_beat_type="8"
               start_divisions="4" end_divisions="4"
               start_bar="2" start_offset="6"
               end_bar="2" end_offset="6" />
    </answer>
  </question>
```

This function reads a question string from an xml file, loads a piece/excerpt of music to which the question refers, analyses the music so as to answer the question, and writes the answer to a different xml results file.

c@merata2014-question-file2question-string

Started, last checked	17/6/2014, 17/6/2014
Location	C@merata processing
Calls	read-from-file-arbitrary, replace-all
Called by	Stravinsqi-Jun2014
Comments/see also	

Example:

```
(setq question-number "004")
(setq
 question-path&name
 (merge-pathnames
  (make-pathname
   :directory
   '(:relative
     "C@merata2014" "training_v1")
   :name "training_v1" :type "xml")
  *MCStylistic-MonthYear-data-path*))
(setq notation-name "f3")
(c@merata-question-file2question-string
 question-number question-path&name notation-name)
--> ("harmonic interval of a minor sixth" 4)
```

This function retrieves a natural-language question string from an xml file, and also retrieves the value stored in the divisions field.

c@merata2014-write-answer

Started, last checked	17/6/2014, 17/6/2014
Location	C@merata processing
Calls	bar&beat-number-of-ontime, row-of-max-ontime<=ontime-arg
Called by	Stravinsqi-Jun2014
Comments/see also	

Example:

```
(setq
 question-string
 "F# followed two crotchets later by a G")
(setq division 1)
(setq time-intervals '((4 8) (8 10)))
(setq ontimes-signatures '((1 4 4 0) (5 3 8 16)))
(setq question-number "004")
(setq
 notation-path
 (merge-pathnames
  (make-pathname
   :directory
```

```
   '(:relative
     "C@merata2014" "training_v1"))
  *MCStylistic-MonthYear-data-path*))
(setq notation-name "f3")
(c@merata2014-write-answer
 question-string division time-intervals
 ontimes-signatures question-number
 notation-path notation-name)
--> (wrote the following to text file)
  <question number="004" music_file="f3.xml" divisions="1">
    <text>"F# followed two crotchets later by a G"</text>
    <answer>
      <passage start_beats="4" start_beat_type="4"
               end_beats="4" end_beat_type="4"
               start_divisions="1" end_divisions="1"
               start_bar="2" start_offset="1"
               end_bar="2" end_offset="4" />
    </answer>
    <answer>
      <passage start_beats="4" start_beat_type="4"
               end_beats="4" end_beat_type="4"
               start_divisions="1" end_divisions="1"
               start_bar="3" start_offset="1"
               end_bar="3" end_offset="2" />
    </answer>
  </question>
```

This function takes a list of time intervals as its main argument and writes them to an xml-style file, conforming to the MediaEval 2014 C@amerata standard.

cross-check-compound-question

Started, last checked	17/6/2014, 17/6/2014
Location	C@merata processing
Calls	bar&beat-number-of-ontime, my-last, ontime-of-bar&beat-number
Called by	cross-check-compound-questions
Comments/see also	

Example:

```
; questions, ontimes-signatures, and ans-list as in
; cross-check-compound-questions
(setq iq 0)
(setq ia 2)
(setq it 0)
(cross-check-compound-question
 questions ontimes-signatures ans-list na iq ia it)
--> ((0 2 0) ((1 0 1)))
```

This function is a helper function for cross-check-compound-questions. It takes a time-interval answer for question component i (where the question is of the compound variety) and looks at the time offset specified for question component $i + 1$ to calculate the target ontime at which an event should occur. It has to be quite subtle because this offset could be expressed as a quantity of note values or bar numbers.

cross-check-compound-questions

Started, last checked	17/6/2014, 17/6/2014
Location	C@merata processing
Calls	cross-check-compound-question, matching-pursuit
Called by	Stravinsqi-Jun2014
Comments/see also	

Example:

```
(setq
 questions
 '(("F#" nil) ("crotchet" 2 "bars")
   ("perfect 5th" 3)))
(setq ontimes-signatures '((1 4 4 0) (5 3 8 16)))
(setq
 ans-list
 '(
   (; ans-fn1 - duration-fn
    ; q1 - pitch
    nil
    (; q2 - duration
     (0 1) (1 2) (8 9))
    (; q3 - harmonic interval
```

```
     (1 7) (2 4)))
   (; ans-fn2 melodic-interval-fn
    ; q1 - pitch
    nil
    ; q2 - duration
    nil
    (; q3 - harmonic interval
     (4 7) (12 14)))
   (; ans-fn3 - pitch-fn
    (; q1 - pitch
     (0 2) (2 5))
    (; q2 - duration
     (8 9) (12 14) (14 15))
    ; q3 - harmonic interval
    nil)
   (; ans-fn4 - harmonic-interval-fn
    ; q1 - pitch
    nil
    (; q2 - duration
     (0 12) (12 14) (14 15))
    (; q3 - harmonic interval
     (4 7) (11 13) (14 15)))))
; (0 2) (8 9) (12 14) Candidate time intervals
; (0 2 0) (1 0 2) (2 3 1) Candidate indices in ans-list
(cross-check-compound-questions
 questions ontimes-signatures ans-list)
--> ((0 13))

(setq
 questions
 '(("F" NIL) ("crotchet" 0) ("perfect fifth" 3)))
(setq
 ans-list
 '(
   (; ans-fn1 - duration-fn
    (; q1 - pitch
     (1 2) (7 8) (9 10))
    (; q2 - duration
     (0 1) (2 3) (8 9))
    (; q3 - harmonic interval
     (1 7) (2 4)))
```

```
   (; ans-fn2 melodic-interval-fn
    (; q1 - pitch
     (0 2))
    ; q2 - duration
    nil
    (; q3 - harmonic interval
     (4 7) (12 14)))
   (; ans-fn3 - pitch-fn
    (; q1 - pitch
     (0 2) (6 8))
    (; q2 - duration
     (8 9) (12 14) (14 15))
    ; q3 - harmonic interval
    nil)
   (; ans-fn4 - harmonic-interval-fn
    ; q1 - pitch
    nil
    (; q2 - duration
     (0 12) (12 14) (14 15))
    (; q3 - harmonic interval
     (5 7) (11 13) (14 15)))))
; (0 2) (2 3) (6 8) Candidate time intervals
; (0 2 0) (1 0 1) (2 3 0) Candidate indices in ans-list
(cross-check-compound-questions
 questions ontimes-signatures ans-list)
--> ((1 7) (7 13) (0 7) (0 7) (6 13))
```

This function looks across an answer list for individual components that could be combined to form answers to compound questions. The answer list is ordered first by answer function, then by question number, then by time interval answering a question.

matching-pursuit

Started, last checked	17/6/2014, 17/6/2014
Location	C@merata processing
Calls	bar&beat-number-of-ontime, ontime-of-bar&beat-number
Called by	cross-check-compound-questions
Comments/see also	

Example:

```
(setq
 assoc-list
 '(((0 0 0) NIL)
   ((0 0 1) ((1 0 2) (1 2 0)))
   ((0 0 2) ((1 0 2) (1 2 0)))
   ((0 1 0) ((1 0 2) (1 2 0)))
   ((0 2 0) ((1 0 2) (1 2 0)))
   ((0 2 1) NIL)
   ((1 0 0) NIL)
   ((1 0 1) ((2 1 0) (2 3 0)))
   ((1 0 2) ((2 3 1)))
   ((1 2 0) ((2 3 1)))
   ((1 2 1) NIL)
   ((1 2 2) NIL)
   ((1 3 0) NIL)
   ((1 3 1) NIL)
   ((1 3 2) NIL)))
(matching-pursuit assoc-list '(0 2 0))
--> ((0 2 0) (1 0 2) (2 3 1))
```

This function 'pursues' a probe list across an assoc list, returning subsequent entries so long as the chain of probes continues. For instance, in the example, list `(1 0 2)` is in the list indexed by `(0 2 0)`, so we return the latter then look it up and see `(2 3 1)` is in the list indexed by `(0 2 0)`, so we return it etc.

I am concerned that this function does not do everything it ought. For example, how would `((0 2 0) (1 0 2) (2 3 1))` emerge: it is a legitimate chain but I do not see how it is returned. More testing required!

4.8.2 Kern to staff features

The functions below will parse a kern file (http://kern.ccarh.org/) and identify various aspects of it. For instance, kern-file2ontimes-signatures will identify all the time signature changes in a kern file and convert them to a list of bar numbers where they occur, and what they consist of.

kern2clef-changes

Started, last checked	17/6/2014, 17/6/2014
Location	Kern to staff features
Calls	kern-anacrusis-correction, kern-col2staff-changes, kern-rows2col-preserving-clefs, read-from-file-arbitrary, sort-dataset-asc, staves-info2staves-variable-robust, tab-separated-string2list, test-all-true
Called by	
Comments/see also	

Example:

```
(setq
 path&name
 (merge-pathnames
  (make-pathname
   :directory '(:relative "Kern")
   :name "C-41-1-ed" :type "txt")
  *MCStylistic-MonthYear-data-path*))
(firstn 5 (kern2clef-changes path&name))
--> ((0 "clefF4" 1) (0 "clefG2" 0) (121 "clefG2" 1)
     (126 "clefF4" 1) (127 "clefG2" 1)
     (132 "clefF4" 1) (133 "clefG2" 1))
```

This function parses a kern file and returns a list consisting of triples: the first element in each triple is ontime; the second is a string specifying the clef change that occurs at that point in time; the third is the staff number to which the clef change belongs.

kern-col2staff-changes

Started, last checked	17/6/2014, 17/6/2014
Location	Kern to staff features
Calls	constant-vector, parse-kern-spaced-notes
Called by	kern2clef-changes
Comments/see also	

Example:

```
(setq
 a-list
 '(NIL NIL NIL (("*clefG2")) NIL NIL (("16r"))
   (("16B" "16c")) (("16d")) (("16e")) (("*clefGv2"))
   (("16f")) (("16d")) (("16e")) (("16c"))))
(kern-col2staff-changes a-list 0 (list 0))
--> ((0 "clefG2") (1 "clefGv2"))
```

This function plays a similar role as the function kern-col2dataset in the function kern-file2dataset-by-col. It keeps a running total of ontime in the spine of a kern file, and uses this to populate any encountered clef changes with the appropriate ontime where the clef change first takes effect.

kern-file2ontimes-signatures

Started, last checked	9/5/2014, 9/5/2014
Location	Kern to staff features
Calls	tab-separated-string2list
Called by	
Comments/see also	append-ontimes-to-time-signatures

Example:

```
(setq
 kern-rows-sep
 '(("**kern" "**kern" "**dynam" "**kern")
   ("*staff3" "*staff2" "*staff2" "*staff1")
   ("*clefF4" "*clefG2" "*clefG2" "*clefG2")
   ("*k[f#c#g#]" "*k[f#c#g#]" "*k[f#c#g#]"
    "*k[f#c#g#]")
   ("*M6/8" "*M6/8" "*M6/8" "*M6/8")
   ("8r" "8a/" "f" "8r")
   ("=1" "=1" "=1" "=1")
   ("2.r" "4.dd\\" "." "2.r")
   ("." "4.d/" "." ".")
   ("=2" "=2" "=2" "=2")
   ("2.r" "4.g#/" "." "2.r")
   ("." "4r" "." ".")
   ("." "8g#/" "." ".")
   ("=3" "=3" "=3" "=3")
   ("*M4/4" "*M4/4" "*M4/4" "*M4/4")
```

```
   ("4BB\\ 4c#\\" "4a/" "." "4g#\\ 4ff#\\")
   ("4Cn\\ 4B-\\" "4a/" "." "4fn/ 4dd/")
   ("2BB\\ 2cn\\" "2a/" "." "2gn\\ 2ffn\\")
   ("=4" "=4" "=4" "=4")
   ("1r" "1r" "." "1r")
   ("==" "==" "==" "==")
   ("*-" "*-" "*-" "*-")))
(kern-file2ontimes-signatures "blah" kern-rows-sep)
--> ((1 6 8) (3 4 4))
```

This function parses a kern file looking for lines beginning with "*M", which denote changes in time signature. Apart from the first such instance, which it is assumed belongs to bar 1 of a piece, it then looks immediately before these lines to parse the bar numbers at which time signature changes occur. Call the function append-ontimes-to-time-signatures afterwards to include ontimes as well.

kern-rows2col-preserving-clefs

Started, last checked	17/6/2014, 17/6/2014
Location	Kern to staff features
Calls	fibonacci-list, nth-list-of-lists, space-bar-separated-string2list, recognised-spine-commandp, tab-separated-string2list, update-staves-variable
Called by	kern2clef-changes
Comments/see also	

Example:

```
(setq
 rows
 '("**kern **kern"
   "*staff2 *staff1"
   "=1-=1-"
   "*clefF4 *clefG2"
   "*k[] *k[]"
   "*M4/4 *M4/4"
   "2r 16r"
   ". 16B/LL 16c/LL"
```

```
   ". 16d/"
   ". 16e/JJ"
   "* *clefGv2"
   ". 16f/LL"
   ". 16d/"
   ". 16e/"
   ". 16c/JJ"))
(kern-rows2col-preserving-clefs rows 1 '((2 1) (1 1)))
--> (NIL NIL NIL (("*clefG2")) NIL NIL (("16r"))
     (("16B" "16c")) (("16d")) (("16e"))
     (("*clefGv2")) (("16f")) (("16d")) (("16e"))
     (("16c")))
```

This function plays a similar role as the function kern-rows2col in the function kern-file2dataset-by-col. It keeps information relating to clefs as well as note information, and removes everything else.

kern-rows-sep2time-sigs&idx

Started, last checked	1/5/2014, 1/5/2014
Location	Kern to staff features
Calls	
Called by	kern-file2ontimes-signatures
Comments/see also	

Example:

```
(setq
 kern-rows-sep
 '(("*M6/8" "*M6/8" "*M6/8" "*M6/8")
   ("8r" "8a/" "f" "8r")
   ("=1" "=1" "=1" "=1")
   ("2.r" "4.dd\\" "." "2.r")
   ("." "4.d/" "." ".")
   ("=2" "=2" "=2" "=2")
   ("*M4/4" "*M4/4" "*M4/4" "*M4/4")
   ("4BB\\ 4c#\\" "4a/" "." "4g#\\ 4ff#\\")
   ("4Cn\\ 4B-\\" "4a/" "." "4fn/ 4dd/")
   ("2BB\\ 2cn\\" "2a/" "." "2gn\\ 2ffn\\")
   ("==" "==" "==" "==")
   ("*-" "*-" "*-" "*-")))
```

```
(kern-rows-sep2time-sigs&idx kern-rows-sep)
--> (((6 8) 0) ((4 4) 6))
```

This function parses rows from a kern file that have already been converted from tab-separated text into lists of strings. It looks for lists that begin with the substring "*M", which specifies a change in time signature, converts the rest of such strings to the upper and lower number of the following time signature (for instance "*M8/8" maps to 6 and 8), and returns this in a list along with the index of the row.

staves-info2staff&clef-names

Started, last checked	17/6/2014, 17/6/2014
Location	Kern to staff features
Calls	nth-list, positions, read-from-file-arbitrary, replace-all, tab-separated-string2list
Called by	Stravinsqi-Jun2014
Comments/see also	

Example:

```
(staves-info2staff&clef-names
 '("!!!COM: Chopin, Frederic"
   "**kern **kern **dynam"
   "*thru *thru *thru"
   "*staff2 *staff1 *staff1/2"
   "*>A *>A *>A"
   "*clefF4 *clefG2 *clefG2"))
--> (("piano left hand" "bass clef")
     ("piano right hand" "treble clef"))
(staves-info2staff&clef-names
 '("!!!COM: Beethoven, Ludwig van"
   "!!!CDT: 1770///-1827///"
   "**kern **dynam"
   "*Ipiano *Ipiano"
   "*clefG2 *clefG2"
   "*k[b-] *k[b-]"
   "*F: *F:" "*M3/4 *M3/4" "*MM40 *MM40"
   "8.c/L ." "16c/Jk ." "=1 =1" "* *"))
--> (("piano right hand" "treble clef"))
(staves-info2staff&clef-names
```

```
 '("!!!AGN: chorale"
   "**kern **kern **kern **kern"
   "*ICvox *ICvox *ICvox *ICvox"
   "*Ibass *Itenor *Ialto *Isoprn"
   "*I\"Bass *I\"Tenor *I\"Alto *I\"Soprano"
   "*>[A,A,B] *>[A,A,B] *>[A,A,B] *>[A,A,B]"
   "*>norep[A,B] *>norep[A,B] *>norep[A,B] *>norep[A,B]"
   "*>A *>A *>A *>A"
   "*clefF4 *clefGv2 *clefG2 *clefG2"
   "*k[f#] *k[f#] *k[f#] *k[f#]"))
--> (("bass" "bass clef") ("tenor" "tenor clef")
     ("alto" "treble clef") ("soprn" "treble clef"))
(staves-info2staff&clef-names
 '("**kern **kern"
   "*staff2 *staff1"
   "=1-=1-"
   "*clefF4 *clefG2"
   "*k[f#c#g#d#] *k[f#c#g#d#]"
   "*M3/8 *M3/8"))
--> (("piano left hand" "bass clef")
     ("piano right hand" "treble clef"))
(staves-info2staff&clef-names
 '("**kern **text **kern **text **kern **text **kern **text"
   "*staff4 *staff4 *staff3 *staff3 *staff2 *staff2 *staff1 *staff1"
   "=1-=1-=1-=1-=1-=1-=1-=1-"
   "*clefF4 * *clefGv2 * *clefG2 * *clefG2 *"
   "*k[b-e-] * *k[b-e-] * *k[b-e-] * *k[b-e-] *"
   "*M4/4 * *M4/4 * *M4/4 * *M4/4 *"
   "1BB-place. 2B-\ place. 2f/ place. 2b-\ place"))
--> (("staff4" "bass clef") ("staff3" "tenor clef")
     ("staff2" "treble clef")
     ("staff1" "treble clef"))
```

This function parses rows supplied from a kern file and returns a list of string pairs: the first element in each pair is a label for a staff in the score. If none is provided, "staff$x$" will appear; the second element is a label for the clef type with which the staff begins (e.g., "bass clef"). This is useful because often users refer to "bass clef" when they mean the lower part of the keyboard staff, for instance.

4.8.3 Pitches intervals durations

The functions below will parse a kern file (http://kern.ccarh.org) and convert it to a dataset. The main function is kern-file2dataset. Occasionally there are conflicts between kern's relative encoding and the timewise parsing function. These have been resolved by the function kern-file2dataset-by-col.

articulation&event-time-intervals

Started, last checked	10/6/2014, 10/6/2014
Location	Pitches intervals durations
Calls	duration-time-intervals, duration&pitch-class-time-intervals, intersection-multidimensional, pitch-class-time-intervals, replace-all, string-separated-string2list
Called by	Stravinsqi-Jun2014
Comments/see also	word&event-time-intervals

Example:

```
(setq question-string "fermata on an F")
(setq
 artic-set
 '((3/2 59 59 1/2 2 NIL NIL NIL)
   (2 50 54 1 3 NIL NIL NIL) (2 57 58 1 2 NIL NIL NIL)
   (2 65 63 1 0 (";") NIL NIL)
   (2 66 63 1 1 NIL NIL NIL)))
(articulation&event-time-intervals
 question-string artic-set)
--> ((2 3))
```

This function looks for expressive markings in the articulation dimension of an articulation point set and events specified in the question string. It returns time intervals corresponding to notes that are set to the expressive marking specified in the question string and that instantiate the specified event.

duration&pitch-class-time-intervals

Started, last checked	16/5/2014, 16/5/2014
Location	Pitches intervals durations
Calls	duration-string2numeric, modify-question-by-staff-restriction, my-last-string, pitch&octave2MIDI-morphetic-pair, replace-all, restrict-dataset-in-nth-to-xs
Called by	
Comments/see also	

Example:

```
(duration&pitch-class-time-intervals
 "dotted minim C sharp"
 '((0 37 46 1 1) (1/3 68 64 1/3 0) (2/3 66 63 1/3 0)
   (1 49 53 3 1) (1 56 57 1 1) (1 59 59 1 1)
   (1 65 62 1/2 0) (3/2 66 63 1/2 0)
   (2 49 53 3 1) (2 50 54 3 1)))
--> ((1 4) (2 5))
```

This function returns (ontime, offtime) pairs of points (notes) that have the duration and pitch class specified by the first string argument. Durations can be in the format "dotted minim" or "dotted half note", for instance. Pitc classes can be in the format The function does not look for dotted notes in the case of the word dotted, but adds one half of the value to the corresponding note type and looks for the numeric value.

duration-string2numeric

Started, last checked	16/5/2014, 16/5/2014
Location	Pitches intervals durations
Calls	
Called by	duration&pitch-class-time-intervals, duration-time-intervals
Comments/see also	

Example:

```
(duration-string2numeric "four hemidemisemiquavers")
```

```
--> 1/16
(duration-string2numeric "dotted quaver")
--> 3/4
(duration-string2numeric "dotted yeah")
--> NIL
```

This function converts a duration expressed in string format into a numeric format.

duration-time-intervals

Started, last checked	16/5/2014, 16/5/2014
Location	Pitches intervals durations
Calls	duration-string2numeric, modify-question-by-staff-restriction, restrict-dataset-in-nth-to-xs
Called by	
Comments/see also	

Example:

```
(duration-string2numeric "four hemidemisemiquavers")
--> 1/16
(duration-string2numeric "dotted quaver")
--> 3/4
(duration-string2numeric "dotted yeah")
--> NIL
```

This function converts a duration expressed in string format into a numeric format.

harmonic-interval-of-a

Started, last checked	1/5/2014, 1/5/2014
Location	Pitches intervals durations
Calls	harmonic-interval-segments2raw-times, interval-string2MNN-MPN-mods, maximal-translatable-pattern, orthogonal-projection-not-unique-equalp, replace-all, segments-strict
Called by	Stravinsqi-Jun2014,
Comments/see also	melodic-interval-of-a

Example:

```
(harmonic-interval-of-a "third"
 '((0 60 60 3 0) (2 63 62 1 0) (5 63 62 1 0)
   (5 67 64 1/2 0)))
--> '((2 3) (5 11/2))
(harmonic-interval-of-a "major 3rd"
 '((0 60 60 3 0) (2 63 62 1 0) (5 63 62 1 0)
   (5 67 64 1/2 0)))
--> ((5 11/2))
```

The first argument is a string; the second is a point set. The function returns a list of raw ontime-offtime pairs during which the harmonic interval specified by the string is sounding. If an ontime-offtime pair is (a, b), it should be noted that the interval sounds in the interval $[a, b)$.

One of the training questions mentioned simultaneous intervals. I will need to write a function that looks for the word "simultaneous", splits the string into the requested intervals, calculates ontime-offtime pairs for each interval, then finds the intersection of these.

harmonic-interval-segments2raw-times

Started, last checked	1/5/2014, 1/5/2014
Location	Pitches intervals durations
Calls	
Called by	labelled-listed-segments2datapoints
Comments/see also	

Example:

```
(harmonic-interval-segments2raw-times
 '(-1 0 1/3 2/3 1 3/2 2 11/4 3)
 '(NIL NIL NIL NIL ((56 57)) ((56 57)) NIL ((59 59))
   NIL))
--> '((1 2) (11/4 3))
```

The first argument is a sequence of ontimes; the second is a list of MNN-MPN pairs of the same length as the first argument. The function unites consecutive windows and returns a list of time windows in which there are non-null items.

interval-string2MNN-MPN-mods

Started, last checked	17/6/2014, 17/6/2014
Location	Pitches intervals durations
Calls	
Called by	harmonic-interval-of-a, melodic-interval-of-a
Comments/see also	

Example:

```
(interval-string2MNN-MPN-mods "perfect 5th")
--> ((7 4))
```

This function converts a string representation of a harmonic or melodic interval to a list of pairs of MIDI note and morphetic pitch numbers modulo twelve and seven respectively. For instance, a perfect fifth is the interval of 7 MNN and 5 MPN.

melodic-interval-of-a

Started, last checked	17/6/2014, 17/6/2014
Location	Pitches intervals durations
Calls	append-list, dataset-restricted-to-m-in-nth, interval-string2MNN-MPN-mods, modify-question-by-staff-restriction, nth-list-of-lists, pairs-forming-melodic-interval-of, replace-all, split-point-set-by-staff
Called by	Stravinsqi-Jun2014
Comments/see also	harmonic-interval-of-a

Example:

```
(setq
 point-set
 '((0 52 55 1/2 1) (1/4 76 69 1/2 0)
   (1/2 54 56 1/2 1) (3/4 75 68 1/2 0)
   (1 56 57 1/2 1) (5/4 74 68 1/2 0)
   (3/2 57 58 1/2 1) (7/4 73 67 1/2 0)
   (2 59 59 1/2 1) (9/4 71 66 1/2 0)
   (5/2 61 60 1/2 1) (11/4 69 65 1/2 0)
   (3 63 61 1/2 1) (13/4 68 64 1/2 0)
```

```
   (7/2 64 62 1/4 1) (15/4 63 61 1/4 1)
   (15/4 66 63 1/2 0) (4 61 60 1/4 1)
   (17/4 59 59 1/4 1) (17/4 68 64 1/8 0)
   (35/8 69 65 1/8 0) (9/2 64 62 1/2 1)
   (9/2 68 64 1/4 0) (19/4 71 66 1/8 0)
   (39/8 69 65 1/8 0) (5 52 55 1/2 1)
   (5 71 66 1/4 0)))
(setq question-string "melodic fourth")
(melodic-interval-of-a question-string point-set)
--> ((17/4 9/2 5))
(setq
 question-string
 "perfect melodic fourth in the bass clef")
(melodic-interval-of-a question-string point-set)
--> ((17/4 9/2 5))
(setq
 question-string "melodic 4th in the treble clef")
(melodic-interval-of-a question-string point-set)
--> nil
(setq
 question-string "4th in the bass clef")
(melodic-interval-of-a question-string point-set)
--> nil
(setq
 question-string "melodic minor 2nd")
(melodic-interval-of-a question-string point-set)
--> ((1/4 3/4 5/4) (5/4 7/4 9/4) (11/4 13/4 15/4)
     (17/4 35/8 9/2) (35/8 9/2 19/4) (1 3/2 2)
     (3 7/2 15/4) (7/2 15/4 4))
(setq
 question-string "rising melodic minor 2nd")
(melodic-interval-of-a question-string point-set)
--> ((17/4 35/8 9/2) (1 3/2 2) (3 7/2 15/4))
(setq
 question-string "melodic descending minor 2nd")
(melodic-interval-of-a question-string point-set)
--> ((1/4 3/4 5/4) (5/4 7/4 9/4) (11/4 13/4 15/4)
     (35/8 9/2 19/4) (7/2 15/4 4))
(setq
 question-string
 "melodic rising minor 2nd in the left hand")
```

```
(melodic-interval-of-a question-string point-set)
--> ((1 3/2 2) (3 7/2 15/4))
(melodic-interval-of-a
 "octave leap" '((0 60 60 1 0) (2 72 67 1 0)))
--> ((0 2 3))
```

The first argument is a string; the second is a point set. The function returns a list of raw ontime1-ontime2-offtime2 triples subtended by the melodic interval specified by the string. The task description suggests that a melodic interval pertains from the ontime of the first note to the offtime of the second note. This causes problems for identifying consecutive melodic intervals, however, because for instance, in the melody C-D-E, technically the second rising melodic second (D-E) begins before the first rising melodic second (C-D) ends. Thus the ontime of the second note is output as well, so that this can be used to identify consecutive intervals if required.

The training questions mention that melodic intervals can only occur 'within staff' (unlike harmonic intervals), so this is how the function has been implemented. It also handles requests to restrict returned results to particular staves, whereas the function harmonic-interval-of-a does not at present.

modify-question-by-staff-restriction

Started, last checked	16/6/2014, 17/6/2014
Location	Pitches intervals durations
Calls	remove-staff-restriction-from-q-string
Called by	duration&pitch-class-time-intervals, duration-time-intervals, melodic-interval-of-a, nadir-apex-time-intervals, pitch-class-time-intervals, rest-duration-time-intervals
Comments/see also	

Example:

```
(setq
 question-string "melodic minor 2nd in the bass clef")
(setq
 staff&clef-names
 '(("piano left hand" "bass clef")
```

```
   ("piano right hand" "treble clef")))
(modify-question-by-staff-restriction
 question-string staff&clef-names)
--> ("melodic minor 2nd" (1))
(setq question-string "melodic minor 2nd")
(modify-question-by-staff-restriction
 question-string staff&clef-names)
--> ("melodic minor 2nd" NIL)
(setq
 question-string
 "perfect fifth between bass and alto voices")
(setq
 staff&clef-names
 '(("Bass" "bass clef") ("Tenor" "tenor clef")
   ("Alto" "treble clef") ("Soprano II" "treble clef")
   ("Soprano I" "treble clef")))
(modify-question-by-staff-restriction
 question-string staff&clef-names)
--> ("perfect fifth" (4 2))
```

This function modifies a question string according to the presence of a substring that restricts a question to a particular staff or voice. The numerical index of the relevant staff or voice is identified (recall that the left-most spines in a parsed kern file have the highest staff numbers) and returned, along with the modified question string (modified to have the substring removed for ease of subsequent processing).

nadir-apex-time-intervals

Started, last checked	11/5/2014, 11/5/2014
Location	Pitches intervals durations
Calls	dataset-restricted-to-m-in-nth, max-nth-argmax, min-nth-argmin, modify-question-by-staff-restriction, replace-all
Called by	Stravinsqi-Jun2014
Comments/see also	

Example:

```
(nadir-apex-time-intervals
```

```
 "nadir in Soprano I voice"
 '((91 67 64 1 0) (92 66 63 3/2 0) (187/2 64 62 1/2 0)
   (94 62 61 1 0) (95 62 61 1 0) (96 74 68 1 0)
   (97 74 68 1 0) (98 75 69 3 0))
 '(("Soprano II" "treble clef")
   ("Soprano I" "treble clef")))
--> ((94 95))
(nadir-apex-time-intervals
 "melodic interval of a second"
 '((91 67 64 1 0) (92 66 63 3/2 0) (98 75 69 3 0))
 '(("Soprano II" "treble clef")
   ("Soprano I" "treble clef")))
--> nil
```

This function locates the lowest- or highest- sounding note, usually in a specified part or voice.

pairs-forming-melodic-interval-of

Started, last checked	7/6/2014, 17/6/2014
Location	Pitches intervals durations
Calls	index-1st-sublist-item>, multiply-list-by-constant, nth-list, nth-list-of-lists, restrict-dataset-in-nth-to-xs, subtract-two-lists
Called by	melodic-interval-of-a
Comments/see also	Could be optimised.

Example:

```
(setq
 point-set
 '((1/4 76 69 1/2 0) (3/4 75 68 1/2 0)
   (5/4 73 67 1/2 0)))
(setq MNN-MPN-mods '((1 1) (2 1)))
(pairs-forming-melodic-interval-of
 point-set MNN-MPN-mods "down")
--> (((1/4 76 69 1/2 0) (3/4 75 68 1/2 0))
     ((3/4 75 68 1/2 0) (5/4 73 67 1/2 0)))
(setq
 point-set
```

```
 '((1/4 76 69 1/2 0) (3/4 75 68 1/2 0)
   (3/4 77 70 1/2 0)
   (5/4 74 68 1/2 0) (7/4 73 67 1/2 0)
   (9/4 71 66 1/2 0) (11/4 69 65 1/2 0)
   (13/4 68 64 1/2 0) (15/4 66 63 1/2 0)
   (17/4 68 64 1/8 0) (35/8 69 65 1/8 0)))
(setq MNN-MPN-mods '((1 1)))
(pairs-forming-melodic-interval-of
 point-set MNN-MPN-mods "either")
--> (((1/4 76 69 1/2 0) (3/4 75 68 1/2 0))
     ((1/4 76 69 1/2 0) (3/4 77 70 1/2 0))
     ((5/4 74 68 1/2 0) (7/4 73 67 1/2 0))
     ((11/4 69 65 1/2 0) (13/4 68 64 1/2 0))
     ((17/4 68 64 1/8 0) (35/8 69 65 1/8 0)))
(pairs-forming-melodic-interval-of
 point-set MNN-MPN-mods "up")
--> (((1/4 76 69 1/2 0) (3/4 77 70 1/2 0))
     ((17/4 68 64 1/8 0) (35/8 69 65 1/8 0))))
(pairs-forming-melodic-interval-of
 point-set MNN-MPN-mods "down")
--> (((1/4 76 69 1/2 0) (3/4 75 68 1/2 0))
     ((5/4 74 68 1/2 0) (7/4 73 67 1/2 0))
     ((11/4 69 65 1/2 0) (13/4 68 64 1/2 0)))
```

This function takes a point set as its first argument and a list of pairs of MIDI note and morphetic pitch numbers (mod twelve and seven respectively) as its second argument. It returns pairs of points that give the melodic interval (rising or falling) specified by the MNN-MPN pairs. The interval between the point pair is strictly melodic, meaning that if the first point in the pair has ontime x and the second point in the pair has ontime y, there can be no other point with ontime $z < y$ (although $z = y$ is permissible).

pitch-class-time-intervals

Started, last checked	1/5/2014, 1/5/2014
Location	Pitches intervals durations
Calls	dataset-restricted-to-m-in-nth, modify-question-by-staff-restriction, my-last-string, pitch&octave2MIDI-morphetic-pair, replace-all, restrict-dataset-in-nth-to-xs
Called by	Stravinsqi-Jun2014
Comments/see also	

Example:

```
(pitch-class-time-intervals
 "F sharp"
 '((-1 66 63 4/3 0) (0 37 46 1 1) (1/3 68 64 1/3 0)
   (2/3 66 63 1/3 0) (1 49 53 1 1) (1 56 57 1 1)
   (1 59 59 1 1) (1 65 62 1/2 0) (3/2 66 63 1/2 0)
   (2 49 53 1 1)))
--> ((-1 1/3) (2/3 1) (3/2 2))
```

This function returns (ontime, offtime) pairs of points (notes) that have the pitch class specified by the first string argument It can be in the format "G double flat" or "Gbb", for instance.

remove-staff-restriction-from-q-string

Started, last checked	16/6/2014, 16/6/2014
Location	Pitches intervals durations
Calls	replace-all
Called by	modify-question-by-staff-restriction
Comments/see also	word&event-time-intervals

Example:

```
(setq question-string "fermata on an F")
(setq
 artic-set
 '((3/2 59 59 1/2 2 NIL NIL NIL)
   (2 50 54 1 3 NIL NIL NIL) (2 57 58 1 2 NIL NIL NIL)
   (2 65 63 1 0 (";") NIL NIL)
```

```
   (2 66 63 1 1 NIL NIL NIL)))
(articulation&event-time-intervals
 question-string artic-set)
--> ((2 3))
```

This function looks for expressive markings in the articulation dimension of an articulation point set and events specified in the question string. It returns time intervals corresponding to notes that are set to the expressive marking specified in the question string and that instantiate the specified event.

tied&event-time-intervals

Started, last checked	17/6/2014, 17/6/2014
Location	Pitches intervals durations
Calls	dataset-restricted-to-m-in-nth, duration-time-intervals, duration&pitch-class-time-intervals, pitch-class-time-intervals, replace-all
Called by	Stravinsqi-Jun2014
Comments/see also	articulation&event-time-intervals, word-time-intervals

Example:

```
(setq question-string "[] F sharp crotchet")
(setq
 tie-set
 '((0 66 63 1 0 "[") (1 66 63 1 0 "[]")
   (2 66 63 1 0 "]") (3 67 64 1 0 "[")
   (4 67 64 1 0 "[]") (5 67 64 1 0 "]")))
(tied&event-time-intervals question-string tie-set)
--> ((1 2))
```

This function looks for ties in the corresponding dimension of an unresolved-tie point set. It returns time intervals of tied notes that also instantiate some other musical event, such as a duration or pitch.

word-time-intervals

Started, last checked	1/5/2014, 1/5/2014
Location	Pitches intervals durations
Calls	sort-dataset-asc, string-separated-string2list
Called by	Stravinsqi-Jun2014
Comments/see also	word&event-time-intervals

Example:

```
(setq question-string "the word \"we\"")
(setq
 artic-set
 '((0 55 57 1 4 NIL NIL ("Sing"))
   (0 62 61 1 3 NIL NIL ("Sing"))
   (1 55 57 1 4 NIL NIL ("we"))
   (1 59 59 2 3 NIL NIL ("we"))))
(word-time-intervals question-string artic-set)
--> ((1 3) (1 2))
```

This function looks for words in the lyrics index of an articulation point set. It returns time intervals corresponding to notes that are set to the word specified in the question string. The use of the functions reverse and sort-dataset-asc could be improved.

word&event-time-intervals

Started, last checked	10/6/2014, 10/6/2014
Location	Pitches intervals durations
Calls	duration-time-intervals, duration&pitch-class-time-intervals, intersection-multidimensional, pitch-class-time-intervals, replace-all, string-separated-string2list
Called by	Stravinsqi-Jun2014
Comments/see also	articulation&event-time-intervals, word-time-intervals

Example:

```
(setq
 artic-set
```

```
 '((0 46 52 4 3 NIL NIL ("might."))
   (0 58 59 2 2 NIL NIL ("might."))
   (3 58 59 1 3 NIL NIL ("Be-"))
   (3 70 66 1 2 NIL NIL ("Be-"))
   (4 69 65 1 1 NIL NIL ("-hold"))
   (4 72 67 1 0 NIL NIL ("-hold"))))
(setq
 question-string "word &quot;Be-&quot; on a B flat")
(word&event-time-intervals question-string artic-set)
--> ((3 4))
(setq
 question-string "word &quot;Ja&quot; on an A flat")
(word&event-time-intervals question-string artic-set)
--> nil
(setq
 question-string "Bb on the word &quot;Be-&quot;")
(word&event-time-intervals question-string artic-set)
--> ((3 4))
```

This function looks for words in the lyrics dimension of an articulation point set and events specified in the question string. It returns time intervals corresponding to notes that are set to the word specified in the question string and that instantiate the specified event.

4.8.4 Texture

The function texture-from-kern will parse a kern file, convert it to a point-set representation in which notes appear as points in pitch-time space, and identify different types of musical texture in the point set (e.g., monophonic, homophonic, melody with accompaniment, polyphonic, or contrapuntal). These textures are output as a list of quads: the first element is the beginning of a time window; the second is the end of a time window; the third is a texture string (from the options above) or nil, and the fourth is a value in $[0, 1]$ expressing the confidence with which the texture label has been assigned.

The functions were coded hastily and require further testing. For examples of different textures, see the source code.

texture-from-kern

Started, last checked	9/6/2014, 9/6/2014
Location	Texture
Calls	append-offtimes, append-ontimes-to-time-signatures, bar&beat-number-of-ontime, datapoints-sounding-between, increment-by-x-n-times, kern-file2dataset-by-col, kern-file2lyrics-tf, kern-file2ontimes-signatures, max-item, min-item, my-last, nth-list-of-lists, ontime-of-bar&beat-number, texture-whole-point-set
Called by	Stravinsqi-Jun2014
Comments/see also	

Example:

```
(setq
 path&name
 (merge-pathnames
  (make-pathname
   :directory '(:relative "C@merata2014" "misc")
   :name "dowland_denmark_galliard" :type "krn")
  *MCStylistic-MonthYear-data-path*))
(setq win-size 4)
(setq hop-size 1)
(setq mono-thresh .95)
(setq homo-thresh .8)
(setq macc-thresh .8)
(setq
 ontimes-signatures
 (append-ontimes-to-time-signatures
  (kern-file2ontimes-signatures path&name)))
(setq
 point-set
 (kern-file2dataset-by-col path&name))
--> ((0 38 47 1 1) (0 57 58 1 0) (0 62 61 1 0)
     (0 69 65 1 0) (1 50 54 2 1) (1 57 58 2 0)
     (1 62 61 2 0) (1 69 65 1 0) (2 69 65 1 0)
```

```
     (3 38 47 3 1) (3 57 58 3 0) (3 69 65 1 0)
     (4 67 64 1/2 0) (9/2 66 63 1/2 0)
     (5 64 62 1/2 0) (11/2 67 64 1/2 0) (6 38 47 2 1)
     (6 57 58 2 0) (6 66 63 1 0) (7 64 62 1/2 0)
     (15/2 62 61 1/2 0) (8 45 51 1 1) (8 64 62 1 0)
     (9 50 54 1 1) (9 57 58 3 0) (9 62 61 3 0)
     (10 38 47 2 1))
(texture-from-kern
 path&name win-size hop-size mono-thresh homo-thresh
 macc-thresh ontimes-signatures point-set)
--> ((0 12 "melody with accompaniment" 0.8333333))

(setq
 path&name
 (merge-pathnames
  (make-pathname
   :directory
   '(:relative "C@merata2014" "training_v1")
   :name "f1" :type "krn")
  *MCStylistic-MonthYear-data-path*))
(texture-from-kern path&name)
--> ((0 36 "homophonic" 0.89666665)
     (36 64 "polyphonic" 1.0))
```

This function parses a kern file, converts it to a point-set representation in which notes appear as points in pitch-time space, and identifies different types of musical texture in the point set (e.g., monophonic, homophonic, melody with accompaniment, polyphonic, or contrapuntal). These textures are output as a list of quads: the first element is the beginning of a time window; the second is the end of a time window; the third is a texture string (from the options above) or nil, and the fourth is a value in $[0, 1]$ expressing the confidence with which the texture label has been assigned.

The function works by passing windows of specifiable size (four bars) and hop (one size) to the function texture-whole-point-set. Windows with contiguous labels are elided. The thresholds for textural labels can be set also.

The point set in the second example comes from bars 23-30 of Morley's 'April is in my mistress' face' (1594). Please see above for more example point sets.

texture-time-intervals

Started, last checked	9/6/2014, 9/6/2014
Location	Texture
Calls	
Called by	Stravinsqi-Jun2014
Comments/see also	

Example:

```
(setq question-string "polyphony")
(setq
 texture-point-set
 '((0 36 "homophonic" 0.89666665)
   (36 64 "polyphonic" 1.0)))
(texture-time-intervals
 question-string texture-point-set)
--> ((36 64))
```

This function identifies the time intervals in some texture point set that correspond to the contents of a question string, and returns those time intervals.

texture-whole-point-set

Started, last checked	9/6/2014, 9/6/2014
Location	Texture
Calls	add-to-nth, constant-vector, dataset-restricted-to-m-in-nth, intersection-multidimensional, max-item, nth-list-of-lists, orthogonal-projection-unique-equalp, set-difference-multidimensional-sorted-asc, sky-line-clipped, test-all-true
Called by	texture-from-kern
Comments/see also	

Example:

```
(setq
 point-set
 '((32 50 54 4 3) (32 62 61 2 2) (32 66 63 4 1)
   (32 69 65 1 0) (33 74 68 1 0) (34 74 68 1 0)
```

```
   (35 62 61 1 2) (35 74 68 1 0) (36 62 61 1 2)
   (36 77 70 4 0) (37 50 54 1 3) (37 62 61 1 2)
   (38 50 54 1 3) (38 65 63 4 2) (39 50 54 1 3)
   (40 53 56 4 3) (41 69 65 1 1) (41 72 67 1 0)
   (42 65 63 2 2) (42 69 65 1 1) (42 72 67 1 0)
   (43 69 65 1 1) (43 72 67 1 0) (44 60 60 4 2)
   (44 72 67 2 1) (44 75 69 2 0) (45 48 53 1 3)
   (46 48 53 1 3) (46 67 64 2 1) (46 75 69 2 0)
   (47 48 53 1 3) (48 51 55 3 3) (48 67 64 2 1)
   (48 70 66 1 0) (49 67 64 1 0) (50 55 57 2 2)
   (50 67 64 2 1) (50 70 66 1 0) (51 51 55 1 3)
   (51 72 67 1 0) (52 46 52 3 3) (52 58 59 2 2)
   (52 65 63 1 1) (52 74 68 8 0) (53 62 61 1 1)
   (54 58 59 2 2) (54 67 64 3 1) (55 48 53 1 3)
   (56 50 54 4 3) (56 57 58 4 2) (57 66 63 1/2 1)
   (115/2 64 62 1/2 1) (58 66 63 2 1) (60 55 57 2 2)
   (60 67 64 1 1) (60 71 66 4 0) (61 67 64 1 1)
   (62 67 64 1 1) (63 55 57 1 2) (63 67 64 1 1)))
(texture-whole-point-set point-set)
--> ("polyphonic" 1)
```

This function takes a point set as its only mandatory argument. Optional arguments include a logical to indicate whether the point set represents music set to words (default true) and thresholds between 0 and 1 for the proportion of conformant notes that must be surpassed in order to declare a texture monophonic, homophonic, melody with accompaniment, etc. The other possible textures are polyphonic (with words), contrapuntal (without words), or no texture defined (nil).

The point set in the example comes from bars 23-30 of Morley's 'April is in my mistress' face' (1594). Please see the source code for more example point sets.

4.8.5 Analytic string manipulations

The functions below are for converting string-based representations of quantity into numeric representations, and for splitting question strings into n components.

append-list-of-lists

Started, last checked	16/6/2014, 16/6/2014
Location	Analytic string manipulations
Calls	
Called by	followed-by-splitter
Comments/see also	append-list

Example:

```
(append-list-of-lists
 '(("yes" 0)
   (("crotchet" 0) ("crotchet" 0) ("crotchet" 0))
   ("no" 4)))
--> (("yes" 0) ("crotchet" 0) ("crotchet" 0)
     ("crotchet" 0) ("no" 4))
(append-list-of-lists '(("yes" 0)))
--> (("yes" 0))
(append-list-of-lists '(("yes" 0) ("no" 0)))
--> (("yes" 0) ("no" 0))
(append-list-of-lists
 '((("crotchet" 0) ("crotchet" 0))))
--> (("crotchet" 0) ("crotchet" 0))
```

In a list this function identifies elements that are lists of lists, and removes one structural level from these lists.

consecutive-question2list

Started, last checked	16/6/2014, 16/6/2014
Location	Analytic string manipulations
Calls	number-string2numberless-string, number-string2numeric
Called by	followed-by-splitter
Comments/see also	

Example:

```
(consecutive-question2list "three ascending seconds")
--> (("ascending second" 0) ("ascending second" 0)
     ("ascending second" 0))
```

```
(consecutive-question2list "three crotchets in a row")
--> (("crotchet" 0) ("crotchet" 0) ("crotchet" 0))
(consecutive-question2list "two consecutive kittens")
--> (("kitten" 0) ("kitten" 0))
(consecutive-question2list "perfect cadence")
--> ("perfect cadence" 0)
(consecutive-question2list "semiquaver rest")
--> ("semiquaver rest" 0)
(consecutive-question2list
 '("consecutive fifths" NIL))
--> (("fifth" 0) ("fifth" 0))
(consecutive-question2list "two fifths")
--> (("fifth" 0) ("fifth" 0))
```

This function turns a question string containing 'consecutive', 'in a row', or some implicit reference to consecutive events such as 'three crotchets' into the appropriate number of question strings. A call to this function is embedded in the function followed-by-splitter.

edit-out-duration-of-question-string

Started, last checked	11/6/2015, 11/6/2015
Location	Analytic string manipulations
Calls	
Called by	??, ??
Comments/see also	

Example:

```
(edit-out-duration-of-question-string
 "dotted crotchet C4")
--> "C4"
(edit-out-duration-of-question-string
 "C4 dotted crotchet in the oboe")
--> "C4   in the oboe"
```

This function, new for Stravinsqi-Jun2015, removes any mention of a musical duration from an input string, returning whatever remains of the string. As the second example shows, it could be improved by removing extra white spaces, which (might) otherwise lead to errors in subsequent processing.

followed-by-splitter

Started, last checked	16/6/2014, 16/6/2014
Location	Analytic string manipulations
Calls	concat-strings, consecutive-question2list, modify-by-later, my-last, ??, replace-all,space-bar-separated-string2list, string-separated-string2list, tied-question2list
Called by	Stravinsqi-Jun2014
Comments/see also	

Example:

```
(followed-by-splitter
 "C followed by Eb in the bass clef")
--> (("C" 0) ("by Eb in the bass clef" 0))
(followed-by-splitter
 "F# followed two crotchets later by a G")
--> (("F#" 0) ("G" 2))
(followed-by-splitter
 "F#, then two quavers later a G")
--> (("F#" 0) ("G" 1))
(followed-by-splitter "F# then two bars later a G")
--> (("F#" 0) ("G" 2 "bars"))
(followed-by-splitter "F# then a G")
--> (("F#" 0) ("G" 0))
(followed-by-splitter
 (concatenate
  'string "three ascending seconds followed by a fall"
  " of a third"))
--> (("ascending second" 0) ("ascending second" 0)
     ("ascending second" 0) ("fall of third" 0))
(followed-by-splitter
 (concatenate
  'string "two falling seconds followed three"
  " quarter notes later by a rise of a third"))
--> (("falling second" 0) ("falling second" 0)
     ("rise of third" 3))
(followed-by-splitter "consecutive fifths")
--> (("fifth" 0) ("fifth" 0))
(followed-by-splitter
```

```
(concatenate
 'string "two falling seconds followed three"
 " quarter notes later by a crotchet tied with a"
 " minim"))
```

If a question string refers to a sequence of events (e.g., ‘F then two crotchets later a G’), this function splits it into two separate questions, also returning the time relation that must pertain between the two events.

modify-by-later

Started, last checked	16/6/2014, 16/6/2014
Location	Analytic string manipulations
Calls	number-string2numeric, number&note2time-interval, replace-all
Called by	followed-by-splitter
Comments/see also	

Example:

```
(modify-by-later "two crotchets later by a G")
--> ("G" 2)
(modify-by-later "four eighth notes later by a G")
--> ("G" 2)
(modify-by-later "two measures later by a G")
--> ("G" 2 "bars")
(modify-by-later "F natural")
--> ("F natural" NIL)
(modify-by-later "four consecutive quavers")
--> ("four consecutive quavers" NIL)
(modify-by-later "four quavers in a row")
--> ("four quavers in a row" NIL)
```

This function edits a question string that refers to a subsequent musical event taking place a specified amount of time later. The question string is cleaned of the “later” reference and the amount of time later is returned (measured in crotchet beats). If it is a number of “bars later” then this is not possible to measure in crotchet beats unless the beginning ontime/bar and time signature (plus changes) are known. So in this case the number of bars is returned, plus the string “bars” for further processing.

my-last-string

Started, last checked	16/6/2014, 16/6/2014
Location	Analytic string manipulations
Calls	
Called by	duration&pitch-class-time-intervals, pitch-class-time-intervals
Comments/see also	my-last

Example:

```
(followed-by-splitter
 "C followed by Eb in the bass clef")
--> (("C" NIL) ("by Eb in the bass clef" NIL))
(followed-by-splitter
 "F# followed two crotchets later by a G")
--> (("F#" NIL) ("G" 2))
(followed-by-splitter
 "F#, then two quavers later a G")
--> (("F#" NIL) ("G" 1))
(followed-by-splitter
 "F# then two bars later a G")
--> (("F#" NIL) ("G" 2 "bars"))
```

If a question string refers to a sequence of events (e.g., "F then two crotchets later a G"), this function splits it into two separate questions, also returning the time relation that must pertain between the two events.

number&note2time-interval

Started, last checked	16/6/2014, 16/6/2014
Location	Analytic string manipulations
Calls	duration-string2numeric, number-string2numeric
Called by	modify-by-later
Comments/see also	

Example:

```
(number&note2time-interval "two crotchets")
--> 2
(number&note2time-interval "two semiquavers")
```

```
--> 1/2
(number&note2time-interval
 "four sixteenth notes")
--> 1
(number&note2time-interval "eighth note")
--> 1/2
(number&note2time-interval "quarter note")
--> 1
(number&note2time-interval
 "three dotted half notes")
--> 9
(number&note2time-interval
 "two consecutive crotchets")
--> nil
```

This function converts a string-based representations of quantity (expressed as a number and a note value) into the numeric representations measured in crotchet beats.

number-string2numberless-string

Started, last checked	16/6/2014, 16/6/2014
Location	Analytic string manipulations
Calls	min-item, replace-once
Called by	consecutive-question2list
Comments/see also	number-string2numeric

Example:

```
(number-string2numberless-string "six quarter notes")
--> "quarter notes"
(number-string2numberless-string "seventh")
--> "seventh"
```

This function returns a revised version of the input string, removing any numeric quantity that appears at the front.

number-string2numeric

Started, last checked	16/6/2014, 16/6/2014
Location	Analytic string manipulations
Calls	min-item
Called by	modify-by-later
Comments/see also	number-string2numberless-string

Example:

```
(number-string2numeric "two crotchets")
--> 2
(number-string2numeric "quarter note")
--> nil
(number-string2numeric "eight crotchets")
--> 8
(number-string2numeric "sixteenth note")
--> nil
(number-string2numeric "six quarter notes")
--> 6
(number-string2numeric "nine consecutive crotchets")
--> "consecutive"
(number-string2numeric "nine crotchets")
--> 9
(number-string2numeric "nine crotchets in a row")
--> "consecutive"
(number-string2numeric "three bars")
--> 3
```

This function returns a natural number, a string, or nil, representing the quantity mentioned at the beginning of the input string argument. The most common use case for this function will be a string containing a quantity of note or bar values that signify a time interval (e.g., two crotchets later). If, however, a number of consecutive events is referred to (e.g., two crotchets in a row), this must be recognised and the string "consecutive" returned instead.

pitch-class-sequential-expression2list

Started, last checked	16/6/2014, 16/6/2014
Location	Analytic string manipulations
Calls	positions-char
Called by	followed-by-splitter
Comments/see also	consecutive-question2list

Example:

```
(pitch-class-sequential-expression2list
 (list "A sharp B flat Db" 2 "bars"))
--> (("A sharp" 2 "bars") ("B flat" 0) ("Db" 0))
(pitch-class-sequential-expression2list
 (list "asecnding G B D" 1))
--> (("G" 1) ("B" 0) ("D" 0))
(pitch-class-sequential-expression2list
 (list "A sharp crotchet B flat Db" 2))
--> (("A sharp crotchet" 2) ("B flat" 0) ("Db" 0))
(pitch-class-sequential-expression2list
 (list "ascending A sharp crotchet" 2))
--> ("ascending A sharp crotchet" 2)
(pitch-class-sequential-expression2list
 (list "quarter note F in the Alto" 0))
--> ("quarter note F in the Alto" 0)
```

This function identifies pitch classes in a question string, such as might be provided sequentially (e.g., C E♭ G). It splits the question into separate elements consisting of these pitch classes (and possibly other events).

tied-question2list

Started, last checked	17/6/2014, 17/6/2014
Location	Analytic string manipulations
Calls	my-last, string-separated-string2list
Called by	followed-by-splitter
Comments/see also	

Example:

```
(setq
```

```
 question-string&time-interval
 (list
  (concatenate
   'string
   "dotted crotchet tied with a crotchet tied with a "
   "quaver") 2 "bars"))
(tied-question2list question-string&time-interval)
--> (("[ dotted crotchet" 2 "bars") ("[] crotchet" 0)
     ("] quaver" 0))
(setq
 question-string&time-interval
 (list
  (concatenate
   'string "dotted crotchet tied with a crotchet") 1))
(tied-question2list question-string&time-interval)
--> (("[ dotted crotchet" 1) ("] crotchet" 0))
(setq
 question-string&time-interval
 (list
  (concatenate
   'string "dotted crotchet") 1))
(tied-question2list question-string&time-interval)
--> ("dotted crotchet" 1)
```

This function unpacks an expression such as 'crotchet tied with a quaver' into the constituents 'crotchet tied forward' and 'quaver tied back'.

4.8.6 Artic dynam lyrics utilities

The functions here are designed to assist with processing articulation, dynamics, and lyrics information in kern files.

articulation-points

Started, last checked	7/6/2014, 28/5/2015
Location	Analytic string manipulations
Calls	
Called by	articulation&event-time-intervals, kern-file2phrase-boundary-states
Comments/see also	

Example:

```
(setq question-string "fermata")
(setq
 artic-set
 '((3/2 59 59 1/2 2 NIL NIL NIL)
   (2 50 54 1 3 NIL NIL NIL) (2 57 58 1 2 NIL NIL NIL)
   (2 65 63 1 0 (";") NIL NIL)
   (2 66 63 1 1 NIL NIL NIL)))
(articulation-points question-string artic-set)
--> (("" "") ";" ((2 65 63 1 0 (";") NIL NIL))).
```

This function looks for expressive markings in the articulation dimension of an articulation point set. It returns points (notes) to which the expressive marking specified in the question string applies.

REFERENCES

Bret J. Aarden. *Dynamic melodic expectancy.* PhD thesis, School of Music, Ohio State University, 2003.

Andreas Arzt, Sebastian Böck, and Gerhard Widmer. Fast identification of piece and score position via symbolic fingerprinting. In Fabien Gouyon, Perfecto Herrera, Luis Gustavo Martin, and Meinard Müller, editors, *Proceedings of the International Symposium on Music Information Retrieval*, pages 433–438, Porto, 2012. International Society for Music Information Retrieval. Retrieved 15 January, 2013 from http://www.cp.jku.at/research/papers/Arzt_etal_ISMIR_2012.pdf.

Richard Cohn. An introduction to neo-riemannian theory: a survey and historical perspective. *Journal of Music Theory*, 42(2):167–180, 1998.

Tom Collins. *Improved methods for pattern discovery in music, with applications in automated stylistic composition.* PhD thesis, Faculty of Mathematics, Computing and Technology, The Open University, 2011.

Tom Collins. Stravinsqi/De Montfort University at the MediaEval 2014 C@merata task. In *Proceedings of the MediaEval 2014 Workshop*, page 6 pages, Barcelona, Spain, 2014.

Tom Collins and Christian Coulon. FreshJam: suggesting continuations of melodic fragments in a specific style. In *Proceedings of the International Workshop on Musical Metacreation*, pages 73–75, Palo Alto, CA, 2012. AAAI Press.

Tom Collins, Robin Laney, Alistair Willis, and Paul H. Garthwaite. Developing and evaluating computational models of musical style. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing.* In press.

Tom Collins, Jeremy Thurlow, Robin Laney, Alistair Willis, and Paul H. Garthwaite. A comparative evaluation of algorithms for discovering translational patterns in baroque keyboard works. In J. Stephen Downie and Remco Veltkamp, editors, *Proceedings of the International Symposium on Music Information Retrieval*, pages 3–8, Utrecht, 2010. International Society for Music Information Retrieval.

Tom Collins, Robin Laney, Alistair Willis, and Paul H. Garthwaite. Modelling pattern importance in chopin's mazurkas. *Music Perception*, 28(4): 387–414, 2011.

Darrell Conklin and Mathieu Bergeron. Feature set patterns in music. *Computer Music Journal*, 32(1):60–70, 2008.

David Cope. *Experiments in musical intelligence*. The Computer Music and Digital Audio Series. A-R Editions, Madison, WI, 1996.

David Cope. *Virtual music: computer synthesis of musical style*. MIT Press, Cambridge, MA, 2001. (Includes essays by Douglas Hofstadter, Eleanor Selfridge-Field, Bernard Greenberg, Steve Larson, Jonathan Berger, and Daniel Dennett).

David Cope. *Computer models of musical creativity*. MIT Press, Cambridge, MA, 2005.

Tuomas Eerola and Adrian C. North. Expectancy-based model of melodic complexity. In Chris Woods, Geoff Luck, Renaud Brochard, Fred Seddon, and John A. Sloboda, editors, *Proceedings of the International Conference on Music Perception and Cognition*, page 7 pages, Keele, UK, 2000. Department of Psychology, Keele University. Retrieved 12 June, 2010 from http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.110.3186.

Jamie Forth and Geraint A. Wiggins. An approach for identifying salient repetition in multidimensional representations of polyphonic music. In Joseph Chan, Jackie Daykin, and M. Sohel Rahman, editors, *London algorithmics 2008: Theory and practice*, Texts in Algorithmics, pages 44–58. College Publications, London, UK, 2009.

Carol L. Krumhansl. *Cognitive foundations of musical pitch*. Oxford University Press, New York, NY, 1990.

Carol L. Krumhansl and Edward J. Kessler. Tracing the dynamic changes in perceived tonal organization in a spatial representation of musical keys. *Psychological Review*, 89(4):334–368, 1982.

David Lewin. *Generalized interval systems and transformations*. Oxford University Press, Oxford, UK, 2007. (Original work published 1987 by Yale University Press, New Haven).

David Meredith. Point-set algorithms for pattern discovery and pattern matching in music. In Tim Crawford and Remco C. Veltkamp, editors, *Proceedings of the Dagstuhl Seminar on Content-Based Retrieval*, page 23 pages, Dagstuhl, Germany, 2006. IBFI. Retrieved 10 August, 2009 from http://drops.dagstuhl.de/opus/volltexte/2006/652/pdf/06171 .MeredithDavid.Paper.652.pdf.

David Meredith, Kjell Lemström, and Geraint A. Wiggins. Algorithms for discovering repeated patterns in multidimensional representations of polyphonic music. *Journal of New Music Research*, 31(4):321–345, 2002.

David Meredith, Kjell Lemström, and Geraint A. Wiggins. Algorithms for discovering repeated patterns in multidimensional representations of polyphonic music. In *Cambridge Music Processing Colloquium*, page 11 pages, Cambridge, UK, 2003. Department of Engineering, University of Cambridge. Retrieved 10 August, 2009 from http://www.titanmusic.com/papers/public/cmpc2003.pdf.

Ignacy J. Paderewski, editor. *Fryderyk Chopin: Complete works*, volume 10. Instytut Fryderyka Chopina, Warsaw, 1953.

Bryan Pardo and William P. Birmingham. Algorithms for chordal analysis. *Computer Music Journal*, 26(2):27–49, 2002.

Marcus T. Pearce and Geraint A. Wiggins. Evaluating cognitive models of musical composition. In Amílcar Cardoso and Geraint A. Wiggins, editors, *Proceedings of the International Joint Workshop on Computational Creativity*, pages 73–80, London, UK, 2007. Goldsmiths, University of London.

Curtis Roads. *The computer music tutorial*. MIT Press, Cambridge, MA, 1996.

Craig Stuart Sapp. Visual hierarchical key analysis. *ACM Computers in Entertainment*, 3(4):19 pages, 2005.

Peter Seibel. *Practical Common Lisp*. Apress, Berkeley, CA, 2005.

Esko Ukkonen, Kjell Lemström, and Veli Mäkinen. Geometric algorithms for transposition invariant content-based music retrieval. In Holger H.

Hoos and David Bainbridge, editors, *Proceedings of the International Symposium on Music Information Retrieval*, pages 193–199, Baltimore, MD, 2003. International Society for Music Information Retrieval. Retrieved 15 February, 2010 from http://ismir2003.ismir.net/papers/Ukkonen.pdf.

Anja Volk. The study of syncopation using inner metric analysis: Linking theoretical and experimental analysis of metre in music. *Journal of New Music Research*, 37(4):259–273, 2008.

Paul von Hippel. Redefining pitch proximity: Tessitura and mobility as constraints on melodic intervals. *Music Perception*, 17(3):315–327, 2000.