NEON.JS: NEUME EDITOR ONLINE

Gregory Burlet gregory.burlet@mail.mcgill.ca
Alastair Porter alastair.porter@mail.mcgill.ca
Andrew Hankinson andrew.hankinson@mail.mcgill.ca
Ichiro Fujinaga ich@music.mcgill.ca

Department of Music Research • Schulich School of Music • **McGill University, CIRMMT** • Montréal, Canada

Overview

Neon.js is a browser-based music notation editor written in JavaScript. The editor can be used to manipulate digitally encoded early musical scores in neume (square-note) notation.

The primary purpose of the editor is to provide a readily accessible interface to correct note pitch and position errors made in the process of optical music recognition (OMR). By being available online, the task of correcting OMR errors can be distributed amongst many people to accelerate the creation of ground-truth data and errorless symbolic music collections.

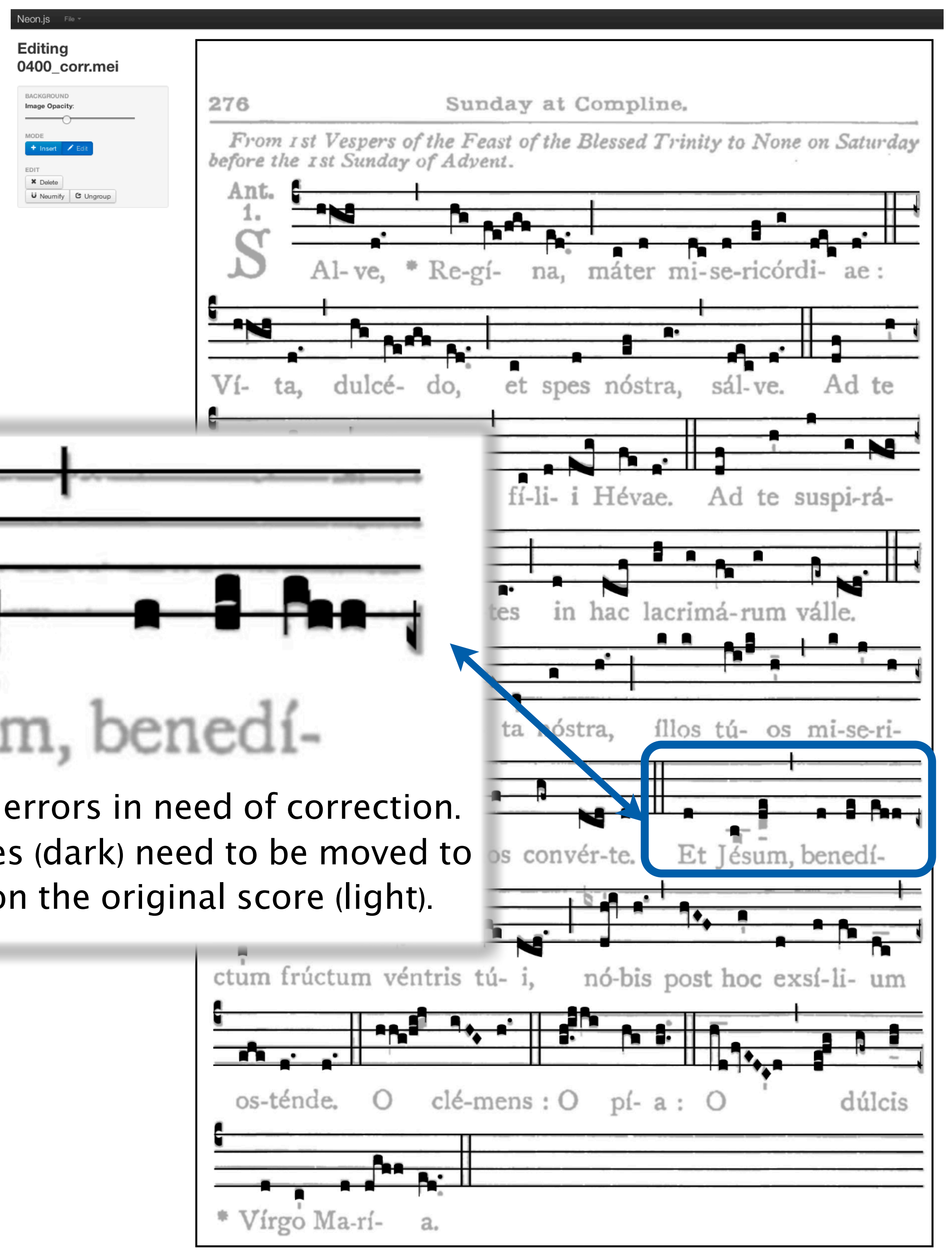
Editing
0400_corr.mei
276
Sunday at Compline.
From 1st Vespers of the Feast of the Blessed Trinity to None on Saturday before the 1st Sunday of Advent.
Ant. 1.
S Al-ve, Re-gí- na, máter mi-se-ricórdi- ae :
Ví- ta, dulcé- do, et spes nóstra, sál-ve. Ad te
fí-li- i Hévae. Ad te suspi-rá-
tes in hac lacrimá-rum válle.
ta nóstra, íllos tú- os mi-se-ri-
os convér-te. Et Jésum, benedí-
ctum frúctum véntris tú- i, nó-bis post hoc exsí-li- um
os-ténde. O clé-mens : O pí- a : O dúlcis
Vírgo Ma-rí- a.

Figure 1. The Neon.js square-note notation editor rendering a page of the *Liber Usualis*.

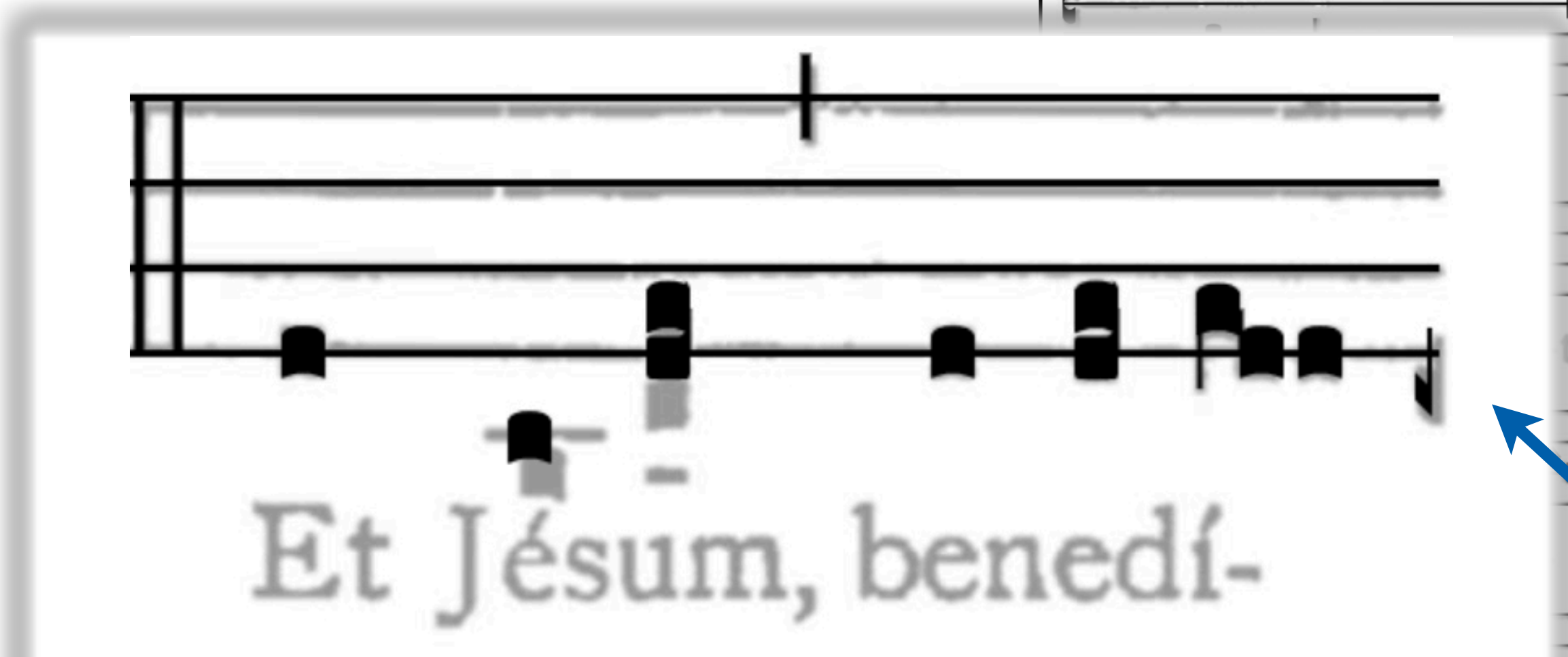

Figure 2. Typical OMR errors in need of correction. Some recognized notes (dark) need to be moved to the same location as on the original score (light).

Neon.js Editor

Notation Encoding

Neon.js uses scores that are encoded in the Music Encoding Initiative (MEI) format. MEI is an XML-based file format for the representation of many music notation formats.

Neon.js uses the Solesmes module, an extension to the MEI core that allows representation of square notes along with other specific practices particular to the notation system used by the monks in Solesmes, France. These practices include divisions (breath marks), episemata (note stresses), and unique neume names.

Notation Rendering

The HTML canvas element is used for rendering musical scores in Neon.js. Images of neumes and ligatures are stored as scalable vector graphics so that the score can be rendered in detail at any zoom level.

MEI files that have been created by OMR contain physical locations on the page of each recognized element. These bounding boxes are used to calculate where to draw musical symbols.

Musical symbols are drawn on top of an image of the original document. The user can adjust the transparency of the background image to show just the rendered notation, or both the background and notation.

Software Architecture

Neon.js has a client-server architecture.

Client:

- Renders the musical score in the browser.
- Transforms user input into AJAX requests that are sent to the server.

Server:

- Receives requests from the client.
- Modifies the underlying MEI file of the score being edited.

Crowdsourcing

Crowdsourcing may be used in an OMR workflow for the purposes of quickly and inexpensively correcting pitch and position errors of notes in digitized musical documents.

The task of correcting pitch and position errors involves dragging the incorrectly recognized notes to match the position of those notes on the source image. A typical correction task is displayed in Figure 2.

By providing an image of the original musical document for reference, a task that would normally require domain-specific musical knowledge can be posed as a comparison problem. This increases the number of possible contributors that can be recruited to perform correction tasks.

Editing Functions

In Neon.js, a neume is represented as a sequence of individual notes called puncta. A selection of puncta may be grouped into larger neume structures using the neumify function.

Neumify

In most cases, the graphical representation of a neume is not a simple concatenation of the selected glyphs. Since each neume is drawn differently, the neume type must be derived.

To derive the neume type:

1. Calculate pitch differences between notes in a selection of neumes.
2. Calculate the melodic contour of the notes (pitch up or down).
3. Search a prefix tree, where the edges represent the direction of movement between two consecutive notes. For notes with the melodic contour *down-up*, the search yields a *porrectus* neume, as in Figure 3.

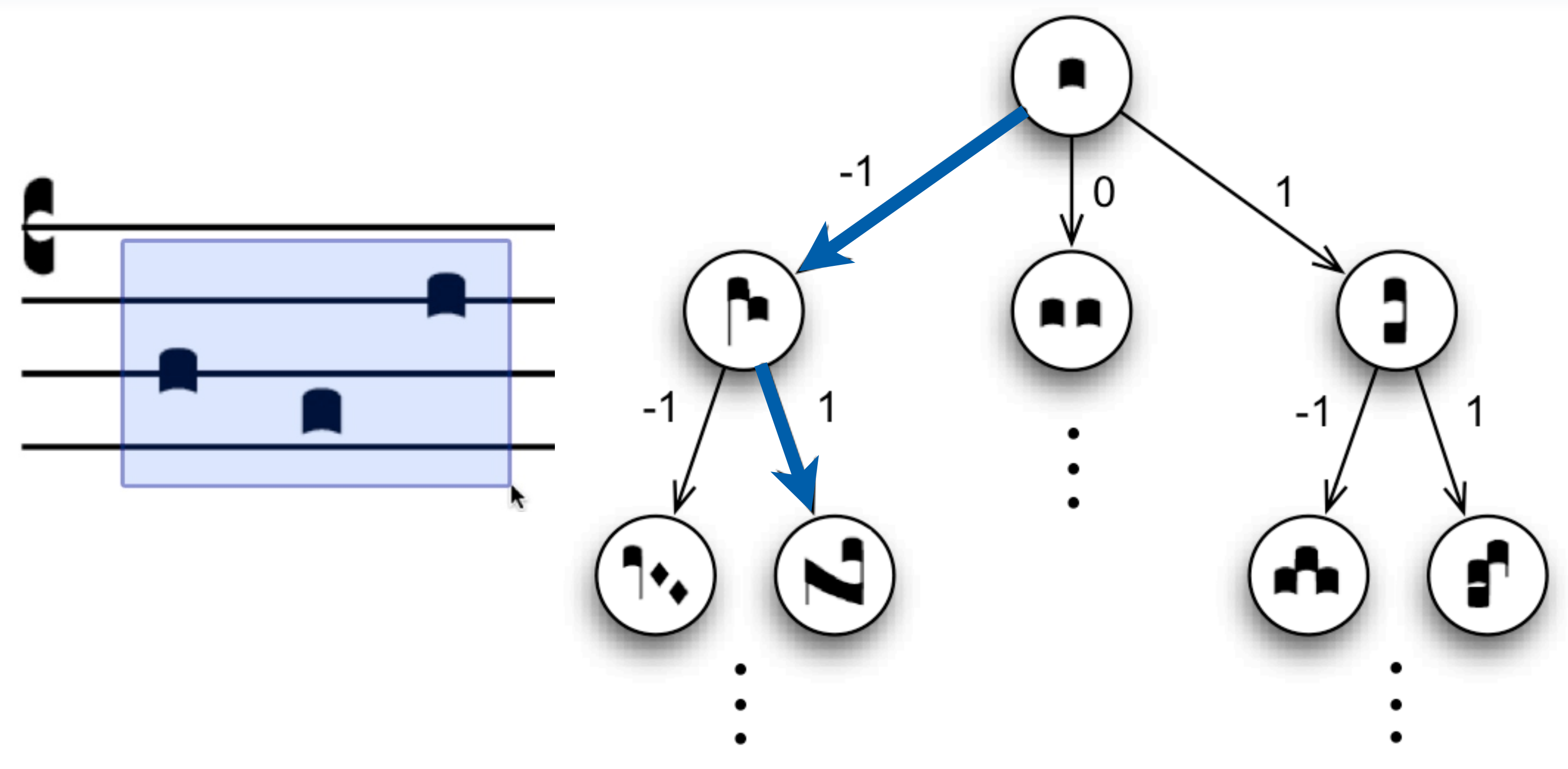

Figure 3. (a) A selection of puncta. (b) Searching a prefix tree to derive the *porrectus* neume type. The bolded arrows reveal the traversal of the tree.

Centre for Interdisciplinary Research in Music Media and Technology • McGill • DDMAL Distributed Digital Music Archives & Libraries Lab • Social Sciences and Humanities Research Council of Canada • Conseil de recherches en sciences humaines du Canada • Canada