Improving Optical Music Recognition by Means of Abductive Constraint Logic Programming

Miguel Ferrand[1,⋆], João Alexandre Leite[2,⋆⋆], and Amilcar Cardoso[1]

[1] Centro de Informática e Sistemas, Universidade de Coimbra (CISUC), Polo II, Pinhal de Marrocos, 3030 Coimbra, Portugal
`mferrand|amilcar@dei.uc.pt`

[2] Centro de Inteligência Artificial (CENTRIA), Departamento de Informática, Universidade Nova de Lisboa, 2825 Monte da Caparica, Portugal
`jleite@di.fct.unl.pt`

Abstract. In this paper we propose a hybrid system that bridges the gap between traditional image processing methods, used for low-level object recognition, and abductive constraint logic programming used for high-level musical interpretation. *Optical Music Recognition* (OMR) is the automatic recognition of a scanned page of printed music. All such systems are evaluated by their rate of successful recognition; therefore a reliable OMR program should be able to detect and eventually correct its own recognition errors. Since we are interested in dealing with polyphonic music, some additional complexity is introduced as several concurrent voices and simultaneous musical events may occur. In RIEM, the OMR system we are developing, when events are inaccurately recognized they will generate inconsistencies in the process of voice separation. Furthermore if some events are missing a consistent voice separation may not even be possible.

In this work we propose an improved architecture for RIEM to allow the system to hypothesize on possible missing events, to overcome the major failure in the voice assignment due to minor recognition failures. We formalize the process of voice assignment and present a practical implementation using *Abductive Constraint Logic Programming.*

Once we abduce these missing events and know where to look for them in the original score image, we may provide the proper feedback to the recognition algorithms, relaxing the recognition thresholds gradually, until some minimum quality recognition criteria is reached.

Keywords: Abduction, Constraints, Optical Music Recognition, Knowledge Based Recognition

1 Introduction and Motivation

Optical Music Recognition (OMR) is the automatic recognition of a scanned page of printed music. All such systems are evaluated by their rate of successful

⋆ The author would like to thank CISUC for supporting his research

⋆⋆ Partially supported by PRAXIS XXI scholarship no. BD/13514/97.

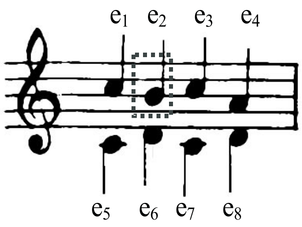

Fig. 1. Score sample

recognition; therefore, a reliable OMR program should be able to detect and eventually correct its own recognition errors. In this work we propose an architecture for an OMR system to deal with uncertain and missing information. We also present a practical implementation of such a system using *Abductive Constraint Logic Programming.*

The difficulties presented to an OMR system are common to most image recognition systems [2]: poor quality of the original, undesirable segmentation resulting from the image scanning process and loss of detail resulting from several processing phases. In the particular case of OMR the high density of musical information creates additional segmentation problems because many graphical objects intersect others or touch where, syntactically, they should not. Other difficulties arise also from the inconsistent nature of music writing (e.g. identical symbols with slight variations in size can have different meanings).

Since we are interested in dealing with polyphonic music, some additional complexity is introduced as several concurrent voices and simultaneous music events may occur. In RIEM [14], the OMR system we are developing, a scheduling algorithm was used to find assignments of events to voices [9]. If events are inaccurately recognized they will generate inconsistencies in the process of voice separation. Furthermore if some events are missing (i.e. symbols not recognized) a voice separation[1] may not be achieved. In the example of Fig. 1, representing an excerpt of a music score with two voices and eight events, one possible separation of voices would be to assign events e_1, e_2, e_3 and e_4 to one voice and e_5, e_6, e_7 and e_8 to another voice. If, for example, the recognition of event e_2 fails, it is no longer possible to perform a voice separation because, according to standard music notation[2], the duration of all voices must be equal.

To overcome these problems some approaches have been presented, making use of music knowledge to enhance the process of recognition. In [4] and [5], a grammar describing musical notation is proposed to control the recognition process particularly the decomposition and labeling of objects. A solution is presented for the problem of voice reconstruction in polyphonic scores and a simplified

[1] In this work we refer to voice separation as the process of syntactical reconstruction of the music score. No semantical music knowledge is involved in this process.

[2] refers to "classical" music notation

prototype has also been implemented, however limited to two voices per staff, and unable to go back to the image score to find unrecognized information.

The absence of high-level musical knowledge during the first phases of recognition leads to a loss of resources in searching for symbols where they have no chance to exist. Therefore an OMR system architecture should allow feedback of high-level knowledge, obtained during the interpretation phase, to low-level data during the recognition [15].

Most knowledge-based approaches have mentioned the need to detect recognition errors but they seldom explain concise methods for repairing or compensating these errors, in particular when objects are not recognized.

In this work we propose an architecture to allow an OMR system to hypothesize on possible missing events, to overcome a failure in the voice assignment process due to minor recognition failures. For this purpose, we will use *Abductive Constraint Logic Programming* (ACLP) [11] [12][13] which is a system that integrates, into a single framework, both the paradigms of *Abductive Logic Programming* (ALP) and *Constraint Logic Programming* (CLP). Both ALP and CLP can be viewed as forms of hypothetical reasoning where a set of hypotheses (abducibles in ALP and constraints in CLP) is determined in order to satisfy a goal. This paradigm will be mainly used to reason about missing events due to any of the reasons mentioned above. The main advantages of ACLP reside on the fact that it permits the expressive power of a high-level declarative representation of problems in a way close to their natural specification (inherited from ALP) while preserving the efficiency of a specialized constraint solver. The preservation of both ALP and CLP identities is of great importance when dealing with problems such as OMR where on one side we have high-level descriptions of syntactic properties of music notation, better suited for ALP, while on the other side we have low-level information, such as spatial coordinates, better dealt with by CLP.

In the example of Fig. 1, we expect the system to abduce a missing event (e_2) while at the same time restrict its spatial coordinates to be situated between those of e_1 and e_3. This way we have the notion of abduced event, with high-level semantic meaning, encapsulating the low-level spatial information.

Once we abduce these events and know where to look for them, in the original score image, we may provide the proper feedback to the recognition algorithms. During this iterative process we may relax the recognition thresholds, gradually, until some minimum quality recognition criteria is reached.

The remainder of this paper is structured as follows: in Sect.2 we briefly describe the ACLP framework; in Sect.3 we propose an OMR architecture; in Sect.4 we describe the image pre-processing and recognition modules; in Sect.5 we formalize the process of interpretation and present an implementation using ACLP; in Sect.6 we conclude and elaborate on future developments.

2 The ACLP Framework

This description closely follows that of [12]. The ACLP framework consists in the integration of two forms of hypothetical reasoning, namely Abductive and Constraint Logic Programming. For a detailed description the reader is referred to [13].

2.1 The Language of ACLP

Given an underlying framework of CLP($\mathcal{R}$), we define:

Definition 1 (Abductive theory or program). *An* ***abductive theory or program*** *in ACLP is a triple* $\langle P, A, IC \rangle$ *where:*

- *P is a constraint logic program in CLP($\mathcal{R}$) consisting of rules of the form* $p_0(t_0) \leftarrow c_1(u_1), ..., c_n(u_n) \,||\, p_1(t_1), ..., p_m(t_m)$[3] *where* p_i *are predicate symbols,* c_i *are constraints in the domain* $\mathcal{R}$ *and* u_i, t_i *are terms of* $\mathcal{R}$*.*
- *A is a set of abducible predicates, different from the constraints in* $\mathcal{R}$*.*
- *IC is a set of integrity constraints, which are first order formulae over the language of CLP($\mathcal{R}$).* $\diamond$

Definition 2 (Goal). *A* ***goal****, G, has the same form as the body of a program rule whose variables are understood as existentially quantified.* $\diamond$

An ACLP theory or program contains three types of predicates: (i) ordinary predicates as in standard LP, (ii) constraint predicates as in CLP and (iii) abducible predicates as in ALP. The abducible predicates are normally not defined in the program and any knowledge about them is represented either explicitly or implicitly in the integrity constraints IC.

The ACLP system allows for the use of Negation as Failure (NAF), handling it through abductive interpretation as proposed in [8].

The abducibles are seen as high-level answer holders for goals (or queries) to the program carrying their solutions.

Definition 3 (Answer). *An* ***answer****,* Δ*, for a goal, G, is a set of assumptions of the form:*

- $ab(d)$*, where* $ab \in A$ *and* $d \in$ *domain of* $\mathcal{R}$*.*
- $\exists X\, (ab_1(X), ..., ab_n(X), C(X))$*, where* $ab_1, ..., ab_n \in A$ *and* $C(X)$ *is a set of CLP($\mathcal{R}$) constraints.* $\diamond$

The integrity constraints express high-level properties that must hold by any solution (set of abducible assumptions) of a goal for this to be accepted.

[3] The symbol || is used to separate the constraint conditions from the program predicate conditions in the conjunction of the body of the rule.

2.2 Declarative Non-monotonic Semantics of ACLP

The (non-monotonic) semantics of ACLP is inherited from that of ALP and abduction. An answer for a goal G is correct if it forms an abductive explanation for G.

Definition 4 (Solution). *Given a theory $\langle P, A, IC \rangle$ and a goal G, an answer Δ is a* ***solution*** *of G iff there exists at least one consistent grounding of Δ (in $\mathcal{R}$) and for any such grounding, Δ_g:*

- *$P \cup \Delta_g$ entails G_g, and*
- *$P \cup \Delta_g$ satisfies the IC.*

where G_g denotes a corresponding grounding of the goal G. $\diamond$

For the details of the corresponding grounding and a formal semantics of the integrity constraints the reader is referred to [13]. Informally integrity constraints are sentences that must be entailed by the program together with the abductive hypotheses ($P \cup \Delta_g$) for Δ_g to be a valid set of hypotheses.

Example 1. Consider the following ACLP theory and goal G:

$$\begin{array}{ll} P = \{p(X) \leftarrow X > 2 || q(X), a(X) & IC = \{\neg(X > 8 || a(X))\} \\ \quad\;\; q(X) \leftarrow X > 4, X < 10 || [\,]\} & A \;\; = \{a\} \\ G = p(X) & \end{array}$$

a solution of G is $\Delta = \exists X \, (a(X), X > 4, X \leq 8)$. $\diamond$

2.3 Implementation

The ACLP system is built, as a meta-interpreter, on top of the ECLiPSe language [7] for Constraint Logic Programming interfacing appropriately the non-monotonic reasoning of abduction with the specialized constraint solving of the CLP language.

Once a theory $\langle P, A, IC \rangle$ is loaded, the program is executed by calling at the ECLiPSe level: $aclp - solve(goal, initial - hypotheses, output - variable)$ where:

- *goal* is an ordinary ECLiPSe goal,
- *initial − hypotheses* is a list of abducible hypotheses, and
- *output − variable* is an ECLiPSe variable.

The *output − variable* returns a list of abducible hypotheses, with their domain variables instantiated to specific values in their domain, containing the *initial − hypotheses* and which is a solution of the *goal*. Normally, the list of initial-hypotheses is empty.

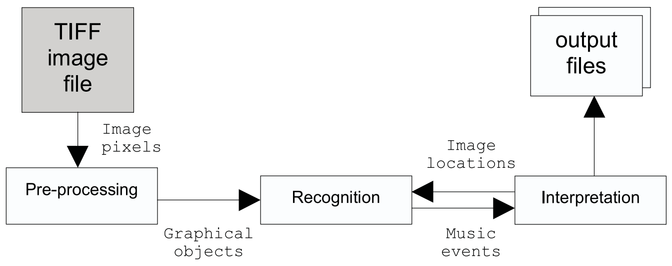

Fig. 2. RIEM - three processing modules

3 System Architecture

In this section we describe the improved architecture of RIEM (a previous version can be found in[14]), an *Optical Music Recognition* system. RIEM has a layered architecture consisting of three main processing modules: *Image Pre-processing*, *Recognition* and *Interpretation* (see Fig.2).

The pre-processing module handles all image low-level processing, transforming image pixels into symbolic data. It reads a scanned page of printed music in the form of a TIFF image file and performs some basic filtering in the image. Follows the detection and removal of the score staff lines to allow the extraction of all remaining graphical objects. These objects are then decomposed and stored as lists of graphical primitives, before being handed to the recognition module.

All information generated by the pre-processing module will be trusted in further modules.

The recognition module transforms graphical objects into meaningful musical events and symbols. Recognition is accomplished through object feature analysis and graphical primitive analysis. In RIEM recognition algorithms make use of thresholds that are functions of the score image features. This eliminates the need for magic numbers hard coded in the recognition procedures and allows the algorithms to cope with some variation in the print quality and style of the music scores.

The interpretation module is responsible for the syntactic reconstruction of the music score. It attempts a coherent voice separation for a given set of recognized events, based on music notation rules. During the process of voice separation the interpretation module may detect some missing events resulting from symbols not recognized. In such cases, information on the hypothetical location of those events is sent back to the recognition module, that will attempt a looser recognition in the specified locations. This process repeats until some limit is reached concerning the relaxation of the recognition parameters.

At the output of the interpretation module a MIDI standard file and a graphical music-publishing file may be generated.

4 Image Pre-processing and Recognition

Follows a more detailed description of the pre-processing and recognition modules of RIEM.

4.1 Image Pre-processing

The scanned music page, stored in a TIFF file, is read by the system and a run-length encoding is performed on the image to eliminate redundancy and therefore increase the performance of further processing stages. Some of the noise contained in the original scanned image is also filtered at this time.

Staff-line detection Staff lines are determinant to reference and interpret all musical symbols. Because staff-lines overlap most all other graphical elements, their presence interferes with the recognition of the information contained in a music page. In RIEM, staff line detection and reconstruction is accomplished by detecting and matching contiguous horizontal line segments in the image. Our method is analogous to the one described in [3] and can cope with some curvature of the staff lines, a frequent case in scanned music pages.

After the exact location of staff lines is known, they are removed from the image in order to isolate all remaining musical symbols. This process is not yet perfect since, in some cases, it is difficult to distinguish which parts of a staff line intersecting an object belong to the object. As a result, some object parts may be inadvertently removed causing additional object segmentation. Also, some unremoved staff line segments may be left attached to objects, introducing slight changes in its shapes.

At this point we are able to determine two important values that will play the role of constants throughout the process. These are the average spacing between contiguous staff lines (`SL_SPACING`) and the average staff line width (`SL_WIDTH`). The first one is a reference for the size of all musical symbols in the score while the second is a measurement of the thickness of some detail features and may also be used as a tolerance value.

Most threshold values used in the recognition algorithms depend on these values. Since they are redefined for every new score page, we make the recognition algorithms independent of variations in music print styles and sizes.

Symbol extraction and classification Contour extraction is performed for all image objects and all contours are decomposed and stored as lists of graphical primitives such as line segments, arcs and junctions. Objects are then classified according to size, proportions, and number and type of compound primitives.

4.2 Recognition

According to the previous classification, graphical objects are handled in dedicated recognition algorithms. Recognition is based on graphical primitive inspection and reconstruction, to find music object components (i.e. note heads, stems, beams...) and also on feature analysis, to acquire attributes for the recognized objects (i.e. white or black notes, number of beams...).

Recognition algorithms incorporate numerical thresholds that are a function of the two constants `SL_ SPACING` and `SL_WIDTH` and also a selectivity index R. Initially the value of R is set to 1 which corresponds to maximum selectivity of the algorithms. This value may be further decreased, after each feedback cycle, to relax the recognition algorithms when re-inspecting particular areas of the image.

Two types of objects may be recognized: non-temporal and temporal objects.

Non-temporal objects are usually expression markings or modifiers of the events pitch, and they are not handled until the whole process of recognition and interpretation is terminated. For details on this matter, the reader is referred to [14].

The recognition module generates events for all temporal objects and these are stored in a structure associated with its corresponding staff. All generated events are uniquely identified and labeled with the corresponding type (note, rest, accidental,...). Some additional attributes are also stored with the event expressing temporal and spatial characteristics.

Spatial attributes include the event cartesian coordinates in the image. These coordinates provide the necessary information to pre-process events in the interpretation module, where important temporal relations will be inferred, based on the relative positioning of events. This spatial event analysis is based on proximity comparisons, and makes use of the two constant values determined in the pre-processing module.

Pitch information is not included with the events because it can also be inferred from its coordinates with respect to the already-known location of the staff lines.

The temporal attributes are the events duration, and will be used later in the interpretation module, for the syntactic reconstruction of the score.

In the previous sub-section we have seen that uncertainty is present in the process of recognition resulting mainly from image deterioration and object segmentation. In RIEM, when object features are ambiguously recognized, and they determine the duration of an event, the recognition algorithms may, in some cases, generate a list of possible durations for the corresponding event.

5 Interpretation

In order to cope with polyphony, a correct voice separation is important because event synchronism is mandatory for the generation of event-based music file formats, like MIDI.

This voice separation is undertaken in the interpretation module of RIEM, based on the syntax of music notation and on topological relations between events.

In this section we describe, in greater detail, the problem of voice separation and event sequencing, together with its implementation within the interpretation

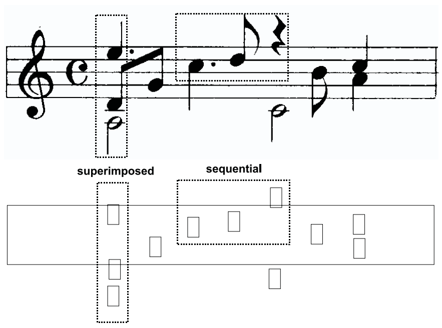

Fig. 3. Original score sample and event diagram showing superimposed and sequential events

module. We start with a general description of the problem, together with the formalization of some needed concepts. The section is concluded with a general description of the implementation using *Abductive Constraint Logic Programming.*

5.1 Voice Separation and Event Sequencing

Music events are represented by graphical symbols that may appear sequentially or superimposed in a music score. Such events can, in a simplified form (i.e. ignoring some attributes not needed at this point), be defined in the following manner:

Definition 5 (Event). *An* ***event*** *e is a triple $\langle X, Y, Dur \rangle$ where:*

- *$X(e) = X$ and $Y(e) = Y$ represent the events relative spatial coordinates;*
- *$Dur_e(e) = Dur$ represent the events duration.* ◇

The relative spatial locations of these events translate into important temporal relations: superimposed events express simultaneity and they are known to have the same starting moment. Sequentially layered events express temporal precedence, thus having distinct starting moments.

These relations are inferred from domain specific spatial analysis (as depicted in Figure 3) and may be determined for any pair of events. Formally we have:

Definition 6 (Precedence). *Let e_1 and e_2 be two events. Event e_1 is said to* ***precede*** *event e_2, denoted by* $\mathbf{precede}(\mathbf{e}_1, \mathbf{e}_2)$*, iff $X(e_1) < X(e_2)$.* ◇

Definition 7 (Simultaneity). *Let e_1 and e_2 be two events. Events e_1 and e_2 are said to be* ***simultaneous****, denoted by* $\mathbf{simultaneous}(\mathbf{e}_1, \mathbf{e}_2)$*, iff $X(e_2) = X(e_1)$.* ◇

A monophonic line or voice, of a music piece, can be represented by a voice (sequence) of related events.

Definition 8 (Voice). *A* ***voice*** *is a finite set of events* $V = \{e_1, ..., e_n\}$ *such that:*

$$\forall_{e_i,e_j \in V, i \neq j}, precede(e_i, e_j) \vee precede(e_j, e_i).\diamond \tag{1}$$

Associated with each voice we have the notion of its total and partial durations, given by the following definitions:

Definition 9 (Voice Duration). *Let* $V = \{e_1, ..., e_n\}$ *be a voice. The* ***voice duration****, denoted by* $Dur(V)$ *is given by:*

$$Dur(V) = \sum_n Dur_e(e_n).\diamond \tag{2}$$

The voice duration results from adding the duration of all events that constitute it.

Definition 10 (Voice Partial Duration). *Let* $V = \{e_1, ..., e_n\}$ *be a voice. The* ***voice partial duration*** *(up to* $e_k \in V$*), denoted by* $Dur_p(e_k, V)$*, is given by:*

$$Dur_p(e_k, V) = \sum_{\substack{e_p \in V \\ precede(e_p, e_k)}} Dur_e(e_p).\diamond \tag{3}$$

The voice partial duration (up to a given event) is the sum of the durations of all events that precede the given one.

In polyphonic music pieces several voices may exist concurrently. All musical events are assigned to an unique voice. These assignments of events to concurrent voices must obey several spatial and temporal properties. Such assignment is defined as follows:

Definition 11 (Voice assignement). *Let* E *be a finite set of events. A* ***voice assignment*** *of* E *is a finite set* $\mathcal{V} = \{V_1, ..., V_n\}$*, where* $V_1, ..., V_n$ *are voices, such that:*

$$\forall_{V_i,V_j \in \mathcal{V}}, Dur(V_i) = Dur(V_j). \tag{4}$$

$$\bigcup_n V_n = E. \tag{5}$$

$$\forall_{V_i,V_j \in \mathcal{V}}, V_i \cap V_j = \emptyset. \tag{6}$$

$$\forall_{e_i \in V_m, e_j \in V_n}, simultaneous(e_i, e_j) \iff Dur_p(e_i, V_m) = Dur_p(e_j, V_n). \tag{7}$$

The number of voices in the assignment is given by n. $\diamond$

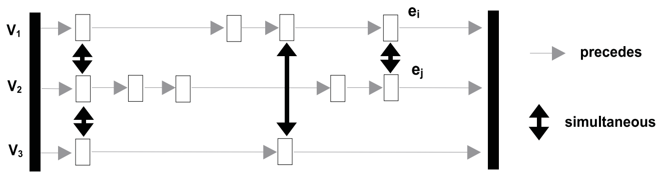

Fig. 4. 3-voice assignment for music excerpt of Fig.3

In the previous definition, (4) imposes that all voices must have the same total duration; (5) imposes that all events must be assigned to a voice; (6) imposes that an event cannot be assigned to more than one voice and (7) means that simultaneous events serve as synchrony checkpoints, i.e. the partial duration, up to two simultaneous events, of the voices to which those events are assigned must be equal; conversely, if the partial duration of two voices is equal, their next events must be simultaneous.

Based on the previous definition, we set forth the class of sets of events for which it is possible to have at least one voice assignment:

Definition 12 (Coherent set of events). *Let E be a finite set of events. E is* ***coherent*** *iff there exists at least one voice assignment of E.* ◇

An example of a voice assignment is represented in Fig.4.

From the previous definitions it follows:

Proposition 1. *Let E be a coherent finite set of events. For any two voice assignments of E, $\mathcal{V}_i$ and $\mathcal{V}_j$, we have that the number of voices of $\mathcal{V}_i$ is the same as the number of voices of $\mathcal{V}_j$.* ◇

This proposition establishes that the number of voices in an assignment is a characteristic of the set of events that produced it. We can then define:

Definition 13. *Let E be a coherent finite set of events. We define $Voices(E)$ as the number of voices of any of the voice assignments of E.* ◇

It is important to mention at this point that these definitions, although set forth for small cells of a music piece (e.g. a measure or line), may be extended to represent and relate a full music piece. Whole measures may be treated as events and may be related by preceding relations. In multi-part pieces several superimposed staffs can be viewed as streams and may be related by simultaneity relations (i.e. as suggested by extended measure bars across staves). This hierarchical nature of the representation enables us to partition large music pieces, by establishing additional constraints between the resulting smaller excepts.

In a previous version of the RIEM system [9], a scheduling algorithm was used to find assignments of events to voices. This algorithm succeeds only if the set of events is coherent. Although a perfect recognition algorithm produces, from a piece of music written according to standard music notation, a coherent set of events, such a recognition rate is usually not the case. This way, the success of the scheduling algorithm cannot be guaranteed.

Usually, what is present at the input of the interpretation module is a non coherent set of events mostly due to unrecognized objects. We must then envisage a process to hypothesize on the missing events. This goal is twofold: on one side, if we are able to produce a set of hypotheses on the unrecognized events, we can feed them back to the recognition module and relax the algorithms there; on the other side we can use these hypotheses to produce a voice assignment that even though may not be a completely accurate representation of the music being recognized, it is definitely better than not have any.

The problem can be synthesized in the following way: given a set of events E, produced by the recognition module, find (abduce) a set of events A such that $E \cup A$ is coherent, i.e. so that a voice assignment can be made. The following proposition guarantees that a solution A always exist:

Proposition 2. *For every finite set of events E, there is at least one finite set of events A such that $E \cup A$ is coherent.* $\diamond$

As is typical in hypothetical reasoning systems, strengthened by the good performance rate of the recognition module of RIEM, we are interested in minimal solutions.

Next we show how this can be accomplished by using Abductive Constraint Logic Programming (ACLP).

5.2 Implementation

As seen before, ACLP presents a declarative and flexible framework to solve the hypothetical reasoning problem at hand.

The main components of the ACLP theory that allows us to abduce the missing events (in practice, what is abduced are the assignments) and perform the voice assignments, $\langle P, A, IC \rangle$, are[4]:

P : The Constraint Logic Program, P, contains:

– rules establishing the necessary spatial relationships (precedence and simultaneity) between events, as in Defs. 6 and 7:

$$\begin{aligned} precede(Id_1, Id_2) \leftarrow {} & X_2 > (X_1 + C_0), \\ & event(Id_1, X_1, \_, \_), event(Id_2, X_2, \_, \_). \end{aligned}$$

$$\begin{aligned} simultaneous(Id_1, Id_2) \leftarrow {} & X_2 \geq (X_1 - C_0), X_2 \leq (X_1 + C_0), \\ & event(Id_1, X_1, \_, \_), event(Id_2, X_2, \_, \_). \end{aligned}$$

[4] For clarity, these rules do not follow the exact syntax of the ACLP meta-interpreter. Also some implementation restrictions of the ACLP meta-interpreter, such as the requesite that rules of P that depend on abducibles must be unfolded into the constraints in IC, have not been followed to make the presentation more understandable.

where the positive constant C_0 is introduced to allow for a margin within which events are still considered simultaneous;
- rules defining the predicates:

$$voice\_duration(Voice, Duration)$$
$$partial\_duration(Voice, Event, Duration)$$

as in Defs. 9 and 10;
- a set of facts, representing the events outputted by the recognition module, of the form:

$$event(Id, X, Y, DurList).$$

where *DurList* is a list of possible durations for the event.
- a rule creating events for every abduced assignment:

$$event(Id, X, Y, Dur) \leftarrow assign(Id, \_, X, Y, Dur).$$

A : There is only one abducible predicate which corresponds to the assignments of events to voices, i.e. $A = \{assign/5\}$.

IC : The set of integrity constraints, IC, contains:

- an IC establishing that simultaneous events cannot be assigned to the same voice, according to (1):

$$\begin{aligned}\perp \leftarrow\ & Id_1 \neq Id_2, simultaneous(Id_1, Id_2),\\ & assign(Id_1, Voice_1, \_, \_, \_), assign(Id_2, Voice_1, \_, \_, \_).\end{aligned}$$

- an IC establishing that the duration of all voices must be equal according to (4):

$$\begin{aligned}\perp \leftarrow\ & Dur_1 \neq Dur_2,\\ & voice\_duration(Voice_1, Dur_1), voice\_duration(Voice_2, Dur_2).\end{aligned}$$

- an IC establishing that every event must have an assignment according to (5). Furthermore, these assignments must have a duration that belongs to the list of possible durations of the event:

$$\begin{aligned}\perp \leftarrow\ & Dur :: DurList,\\ & event(Id, \_, \_, DurList), not\, assign(Id, \_, \_, \_, Dur).\end{aligned}$$

- an IC establishing that the same event cannot be assigned to different voices according to (6):

$$\begin{aligned}\perp \leftarrow\ & Voice_1 \neq Voice_2,\\ & assign(Id, Voice_1, \_, \_, \_), assign(Id, Voice_2, \_, \_, \_).\end{aligned}$$

– an IC establishing that the partial durations, up to simultaneous events, of two voices must be equal according to (7):

$$\begin{aligned}&\perp \leftarrow Dur_1 \neq Dur_2, simultaneous(Id_1, Id_2),\\ &\quad assign(Id_1, Voice_1, \_, \_, \_), partial\_duration(Voice_1, Id_1, Dur_1),\\ &\quad assign(Id_2, Voice_2, \_, \_, \_), partial\_duration(Voice_2, Id_2, Dur_2).\end{aligned}$$

The module is called with the ACLP meta-predicate $aclp - solve(true, [], A)$, returning in A, the set of abduced assignments.

Example 2. In the example of Fig.1, with event e_2 not being recognized, the input of the Interpretation module would be the following set of events E (for clarity, the Id's of the events have been set according to Fig.1):

$$\begin{aligned}E = \{&event(e_1, 42, 130, [1]), event(e_3, 130, 132, [1]), event(e_4, 182, 151, [1]),\\ &event(e_5, 38, 200, [1]), event(e_6, 85, 189, [1]), event(e_7, 129, 201, [1]),\\ &event(e_8, 180, 190, [1])\}\end{aligned}$$

with $C_0 = 20$. Δ would be a solution provided by the interpretation module:

$$\begin{aligned}\Delta = \exists &X_1, X_2, X_3, X_4, X_5(assign(e_1, X_1, 42, 130, 1), assign(e_3, X_1, 130, 132, 1),\\ &assign(e_4, X_1, 182, 151, 1), assign(e_5, X_2, 38, 200, 1), assign(e_6, X_2, 85, 189, 1),\\ &assign(e_7, X_2, 129, 201, 1), assign(e_8, X_2, 180, 190, 1),\\ &assign(e_2, X_1, X_3, X_4, X_5), X_1 \neq X_2, X_3 \geq 65, X_3 \leq 105, X_5 = 1\end{aligned}$$

representing the abduction of 8 assignments, where $assign(e_2, X_1, X_3, X_4, X_5)$ corresponds to the assignment of the event that is missing from E. ◇

6 Conclusions and Future Work

In this paper we proposed a hybrid system that bridges the gap between traditional object recognition methods, used for low-level image processing, and abductive constraint logic programming used for high-level musical interpretation.

It is desirable that OMR systems are capable of detecting and correcting its own recognition errors. For this, we propose an architecture to achieve a coherent voice separation in the presence of uncertain or missing event information. Our model deals with polyphonic scores with no limit on the number of voices.

We have formalized the process of voice separation setting forth some properties that, among other things, guarantee that a solution is always provided. We then use ACLP to perform voice separation, achieve the necessary voice assignments, while at the same time adbucing information about missing events, that is determinant to provide feedback to the recognition algorithms.

The ability to perform hypothetical reasoning, characteristic of ACLP, together with the use of feedback in our architecture, is a significant improvement over the previous system. In a forthcoming paper, we will report the practical results of our architecture's implementation.

As we relax the recognition algorithms, we increase the level of uncertainty, therefore the use of three-valued abduction [6] could be advantageous. Also, new developments related to the use of tabling techniques to perform adbuction [1], appear to be very promising, and could improve the performance of the system.

References

1. J. J. Alferes, L. M. Pereira, T. Swift. *Well-founded Abduction via Tabled Dual Programs.* Technical Report, Dept. de Informática, New University of Lisbon, 1999.
2. H.S. Baird, H. Bunke, K. Yamamoto. *Structured Document Image Analysis.* Springer Verlag, 1992.
3. N.P. Carter. *Automatic Recognition of Printed Music in the Context of Electronic Publishing.* PhD Thesis, University of Surrey, February 1989.
4. B. Coüasnon, P. Brisset, I. Stéphan. *Using Logic Programming languages For Optical Music Recognition.* In Proceedings of the Third International Conference on The Practical Application of Prolog, Paris, France, 1995.
5. B. Coüasnon and B. Retif. *Using A Grammar For A Reliable Full Score Recognition System.* In Proceedings of the International Computer MusicConference, Banff, Canada, September 1995.
6. C. V. Damásio, L. M. Pereira, *Abduction on 3 valued Extended Logic Programs*, In V. W. Marek, A. Nerode and M. Trusczynski, editors, Logic Programming and Non-Monotonic Reasoning, Proc. of 3rd International Conf., LPNMR'95, Lecture Notes in Artificial Intelligence 928, pp. 29-42, Springer-Verlag, 1995.
7. *ECLiPSe User manual.* ECRC, Munich, Germany, 1994.
8. K. Eshghi, R. A. Kowalski. *Abduction Compared with Negation by Failure.* Proceedings of the 6th International Conference on Logic Programming, ICLP89, Lisbon, MIT Press, 1989.
9. M. Ferrand, A. Cardoso. *Scheduling to Reduce Uncertainty in Syntactical Music Structures.* Proceedings of the XIVth Brazilian Symposium on Artificial Intelligence, Brazil, 1998.
10. A. Kakas, R. Kowalski, F. Toni. *The Role of Abduction in Logic Programming.* Handbook of Logic in Artificial Intelligence and Logic Programming 5, pages 235-324, D.M. Gabbay, C.J. Hogger and J.A. Robinson eds., Oxford University Press, 1998.
11. A. Kakas, A. Michael. *Integrating abductive and constraint logic programming.* Proceedings of the Twelfth International Conference on Logic Programming, Tokyo 1995.
12. A. Kakas, C. Mourlas. *ACLP: Flexible Solutions to Complex Problems.* Proceedings of Logic Programming and Non-monotonic Reasoning, 1997.
13. A. Kakas, A. Michael, C. Mourlas. *Abductive Constraint Logic Programming.* Technical Report, University of Cyprus, 1998.
14. J. A. Leite, M. Ferrand, A. Cardoso. *RIEM - A System for Recognition and Interpretation of Music Writing* (in portuguese). Internal Report RI-DEI-001-98, Dept. Engenharia Informática, Universidade de Coimbra, 1998.
15. M.V. Stuckelberg, C. Pellegrini, M. Hilario. *An Architecture for Musical Score Recognition using High-Level Domain Knowledge.* Proceedings of the Fourth International Conferences on Document Analysis and Recognition, IEEE Computer Society, Ulm, 1997.