The Score Analysis/Re-Synthesis Environment of the "Intelligent Music Workstation"

Goffredo Haus, Alberto Sametti

L.I.M. - Laboratorio di Informatica Musicale
Dipartimento di Scienze dell'Informazione
Università degli Studi di Milano

Introduction

The environment for the analysis/re-synthesis of scores sets up the most abstract level (structural/symbolical) of the "Intelligent Music Workstation" (Pighi et al., 1993). It provides terminal oriented aids for the automatic decomposition of scores, for the synthesis of generative models based on the results of the decomposition, and for the execution of these models. This environment aims at increasing individual productivity on the part of both musicologist and composer. In fact, it allows us to recognize hidden structures in the scores and to use them either to give a more abstract and structural representation of the same scores or to create a variety of scores with structural features (according to the musician) more or less related to those of a given one. The environment consists of three software modules: ScoreSegmenter (for the decomposition), ModelSynth (for the synthesis of the generative models), ScoreSynth (for the execution of the models or for the synthesis of new scores). We also underline that the execution with ScoreSynth of a model generated by ModelSynth, with no changes by the musician, will result in the generation of the starting score which had been decomposed with ScoreSegmenter.

Software Architecture

In this chapter we will analyze the most general software architecture making up the basic context of the central module ModelSynth. As shown in Fig. 1 the

other two modules ScoreSegmenter and Scoresynth are directly linked with it. Here here we briefly describe the peculiarities of the three modules.
ScoreSegmenter aims at the segmentation of musical passages as a first step for their future automatic instrumentation proposed by the computer. It is the focus of the research of the various music objects with which the passage is composed. By "object" we intend those musical fragments that the author has expressed, taking and transforming them according to various musical canons in relation to the historical period and composition form. This is a software prototype which allows us to approach musical texts not as a pure sequence of notes but as an expressive building consisting of some basic elements functionally structured according to a number of different forms. ScoreSegmenter is also essentially an analysis tool. ScoreSynth, on the contrary, is a synthesis tool. It makes available an integrated environment for the creation, debugging and execution of hierarchically organized models of musical scores based on Petri Net formalism. The primary aim of the ScoreSynth module is to put at the musician's disposal a very powerful and effective tool which allows him to consider the composition from an architectural point of view: the "musician" manipulates sequences, functions of transformation and structures. Finally, ModelSynth allows the translation (in terms of Petri Net models achievable by Scoresynth) of the results of the analysis operated by the module ScoreSegmenter. (see Fig. 2)

Functional and I/O specifications

Considering the three modules from a functional viewpoint, we now examine what transformations define and what are their individual fields.

ScoreSegmenter
ScoreSegmenter is able to read a score expressed in traditional music notation and to create a supporting file (called Workingfile), through which it can perform its own analysis, the different themes in SMF format (Standard MIDI File 1.0) and, most important, a textual table containing the results of the researches (Lonati, 1991a) (Lonati, 1991b). This last table, for each theme or fragment found, reports the following information: the voice, the starting note and the final one within the score, the operator (when applied), and the first recognized note of the theme. From now on we will refer to each line of the

table with the term "atom". In order that the ScoreSegmenter tables are significant we have to fix the planning input data of the segmentation so as to carry on exclusive searches into fragments and themes. In this way it is possible to avoid that a sequence of notes, in a certain position in the score, belongs to more than one theme or fragment.

ModelSynth
ModelSynth reads the recognized and stored fragments in SMF format and the textual table produced by the analysis of the ScoreSegmenter; it analyzes them iteratively in order to recognize structures describable by means of Petri Nets; finally, it synthesizes a generative hierarchic model of Petri Nets, achievable by ScoreSynth, having the fragments in SMF format linked to suitable place nodes (knots) of the model.

ScoreSynth
As already said ScoreSynth allows the editing, debugging and execution of models of Petri Nets oriented to the synthesis of MIDI music scores. For a detailed description of the functionalities see Haus and Sametti (1992).

How the module ScoreSegmenter performs the analysis task

A first fundamental aspect of the module consists in the development of algorithms for seeking out occurrences of the music objects or their subparts within the passage (objects that can be recognized by the computer itself or provided by the user). In this regard, we must consider two elements: the attributes of the single note, and the music transformation the note has undergone together with the notes that precede and/or follow it. The attributes are the timing, the accent, the name of the note, the pitch in halftones, and the course of the intervals. In regard to transformations, here we have considered those applied to the attributes of position, or degree, both in the tonal and diatonic scale. These transformations have been realized by means of three different types of algebraic operators and their combinations: precisely, transposition, mirror-inversion, retrogradation operators. As we have already pointed out, these operators realize the correspondent music tonal transformations, when applied to the pitch in halftones, and the real transformations, when applied to the name of the notes. To render flexible this analysis tool we can intervene iteratively to modify significantly the "style" with which the researches are carried on, choosing which attributes must be considered, which tranformations and which variability rate must be applied to the tonal analysis. A second aspect is the real segmentation. We simply focused on the music form of the fugue, generalizing it to the other forms even if it would be opportune to enlarge the algorithm so as to single out the theme of the other forms with more precision and specificity. For singling out the objects we focused not only upon their repetitions because, although obviously a necessary aspect, they are insufficient for music purposes. A theme has in fact a tonal and metric system too (except in modern music) which must be taken into consideration. Thus we make some requests for notes of the hypothetic themes, in relation to the belonging and assertion of the tonalities according to the type of object (thetic, acephalous and anacrustic) and in relation to the exhaustiveness and completeness of the music (as an expression of the metrics). Also for this singling out of themes phase we have arranged a series of parameters which must be specified iteratively in regard both to the valuable elements in the definition of the theme (tonality,metrics, length of phrasing, minimum number of repetitions, etc.), and to the same repetition of the objects.

How the module ModelSynth performs the analysis/re-synthesis task

In substance, ModeSynth performs an operation opposite to the ScoreSegmenter one. In fact, it tries to re-form the structure of the analyzed and atomized music passage. In order not to be a redundant migration, this operation must obviously give some significant features to the final structure of the passage. These features can be summed up in one: representing the informative content of the passage with flexible models (and by "informative" we mean the link between the constituent atoms and their transformations during the process). What we achieve is the analysis of an analysis, in order on the one hand to provide an alternative representation and on the other to extract and code the information for the re-synthesis. The first analysis performed by Modelsynth consists in parsing the table produced by the ScoreSegmenter, generated by the request of segmentation of a passage, using as a support the relative Workingfile. This phase allows the representation in intermediate format of the results contained in the tables. Or it saves all the recognized themes giving a reference code, subdivides all the atoms by voice, rearranges their occurrences according to a time key, and recovers from the Workingfile all the parts (sequences of notes and rests) which we can call "rejects" judged insignificant by the segmentation algorithm but considered the link among the different atoms. Secondly, we have to consider the operators applied to the atoms. Since the operators recognized by the ScoreSegmenter are a narrow subset of those available in ScoreSynth, this operation is immediate. At this point ModelSynth has at its disposal all the information to operate its own analysis, as well as the automatic construction of a Petri Net model in the ScoreSynth format. Correspondent to the starting passage, the latter brings out, if present, the relational constructions and the used transformation functions. Fundamentally, in this phase we can exploit the mechanisms of parametric calls at hierarchic nets disposed by ScoreSynth. These realize the division between the structure of the passage and the themes. The structure is identified by the relations among the themes, their repetitions and their transformations; and it is also codified into a model. The themes only play the role of data. The analysis proceeds considering the voices singularly. The structures we try to recognize are the *simple loops*, the *loops with selection* and the *repetition of specific patterns*. By simple loops we intend the succession of a theme or relative transformation. For example, considering the theme `A` and an operator `T`, we can represent by means of a macro-net a succession as follows

```
T (A) - T (A) - T (A) - T (A)
```

realizing a loop to which the theme, operator and number of the repetitions are passed as parameters. A loop with selection instead occurs when the operator (applied to a same theme) varies. For example, in the succession

```
T (A) - R (A) - I (A)
```

This construction type always uses as a basis a loop-type net. The difference consists in the fact the object of the looping is not a single element, but a macro-subnet to whom the operators to apply at each cycle are passed as parameters. Besides the resolution of the loops, we must also recognize the patterns. These can be sought out at different levels. For example, at the operators level in successions concerning a same theme, as follows:

```
T (A) - R (A) - T (A) -R (A) - T (A) - R (A)
```

At the themes level, as follows

```
T (A) - R (B) - I (C) -T (A) - R (B) - I (C)
```

Or, finally, at the whole nets level. The most significant phases of the ModelSynth analysis can also be summed up through the following steps:

i) creation of an intermediate representation;
ii) pattern recognition on the operators;
iii) recognition of simple loops or with selection ones;
iv) pattern recognition on themes;
v) pattern recognition on whole nets;
vi) go to v).

The analysis process continues only if step v) is able to operate at least one recognition. In the Petri Nets represented in the following figures we show the possible result concerning the most general aspects of the analysis operated by ModelSynth on a passage whose name is `PieceX`. In particular Fig. 3 represents the net of the highest level of the model; Fig. 4 the subnet concerning the involved parts (in this example, three); and Fig. 5 the net `collecting` all the complete themes recognized by ScoreSegmenter. Fig. 6 instead shows how in the model the application of an operator to a theme can be represented: the operator in the line below shows that the transformation operated by it refers to the theme `Theme2`. This simple net could be invoked many times in a model as follows: the generated net of highest level (see Fig. 3), the `PieceX` net (see Fig. 4), the net with all the recognized themes (see Fig. 5), an example of application of a transformation operator to a theme (see Fig. 6).

A Comprehensive Example of Automated Score Modelling

This chapter will take a look at a complete example showing how a PN model is generated. The musical score we consider is the "Invenzione N°1 a due voci per pianoforte" by J.S.Bach. See the complete score in Figure 7.

Here follows the atom tables generated by ScoreSegmenter:

TONALITY

C

RESEARCH OF THE THEME NUMBER 1 FORMED BY:

C_W_1/16 +D_S_1/16 +E_W_1/16 +F_S_1/16 -D_W_1/16 +E_S_1/16 -C_W_1/16 +G_S_1/8 +C_W_1/8 -B_S_1/8 +C_W_1/8 +D_S_1/16

RECUR	VC	MEA	NT(start)	MEA	NT(end)	OPER
1 t	1	1	2 C	2	1 D	RT(0)
2 t	2	7	2 G	8	1 A	RT(7)

There was found in all 2 themes of which:

- 2 Real Transpositions
- 0 Tonal Transpositions
- 0 Real Inversions
- 0 Tonal Inversions

RESEARCH OF THE THEME NUMBER 2 FORMED BY:

C_W_1/16 +D_S_1/16 +E_W_1/16 +F_S_1/16 -D_W_1/16 +E_S_1/16 -C_W_1/16 +G_S_1/8

RECUR	VC	MEA	NT(start)	MEA	NT(end)	OPER
1 t	1	1	2 C	1	9 G	RT(0)
2 t	1	2	2 G	2	9 D	RT(7)
3 t	1	20	2 G	20	9 C	TT(4)
4 t	2	1	3 C	2	1 G	RT(0)
5 t	2	2	5 G	3	1 C	TT(4)
6 t	2	5	2 D	5	9 G	TT(1)
7 t	2	7	2 G	7	9 D	RT(7)
8 t	2	8	2 D	8	9 A	RT(2)

There was found in all 8 themes of which:

- 5 Real Transpositions
- 3 Tonal Transpositions
- 0 Real Inversions
- 0 Tonal Inversions

RESEARCH OF THE THEME NUMBER 3 FORMED BY:

G_W_1/8 -F_S_1/8 +G_W_1/8 -E_S_1/16

RECUR	VC	MEA	NT(start)	MEA	NT(end)	OPER
1 t	1	2	10 G	3	1 E	RT(0)

There was found in all 1 themes of which:

- 1 Real Transpositions
- 0 Tonal Transpositions
- 0 Real Inversions
- 0 Tonal Inversions

RESEARCH OF THE THEME NUMBER 4 FORMED BY:

B_W_1/8 +C_S_1/8 +D_W_1/8 +E_S_1/8

RECUR	VC	MEA	NT(start)	MEA	NT(end)	OPER
1 t	1	11	2 C	11	5 F	TT(1)
2 t	1	11	6 A	12	1 D	TT(6)
3 t	1	12	2 F	12	5 B	TT(4)
4 t	2	3	2 B	3	5 E	RT(0)
5 t	2	3	6 G	4	1 C	TT(5)
6 t	2	4	2 E	4	5 A	TT(3)
7 t	2	5	10 B	6	1 E	RT(0)
8 t	2	5	12 D	6	3 G	TT(2)
9 t	2	19	2 B	19	5 F	RI(9)
10 t	2	19	6 D	20	1 A	TI(0)

There was found in all 10 themes of which:

- 2 Real Transpositions
- 6 Tonal Transpositions
- 1 Real Inversions
- 1 Tonal Inversions

After ModelSynth parsing of the above table, we obtained this atom list. The atoms marked with “No” are the ones discarded because in conflict with one another. If a conflict happens, the longest (referring to the note number in it) or, at least, the first is chosen. In this list we put in evidence the multiple occurrence of the same theme using a framing rectangle.

IND	TYP	OK	REC	THE	VC	MEA	NTS	MEA	NTE
1	T	Yes	1	1	1	1	2	2	1
2	T	**No**	1	2	1	1	2	1	9
3	T	Yes	2	2	1	2	2	2	9
4	T	Yes	1	3	1	2	10	3	1
5	T	Yes	1	4	1	11	2	11	5
6	T	Yes	2	4	1	11	6	12	1
7	T	Yes	3	4	1	12	2	12	5
8	T	Yes	3	2	1	20	2	20	9
9	T	Yes	4	2	2	1	3	2	1
10	T	Yes	5	2	2	2	5	3	1
11	T	Yes	4	4	2	3	2	3	5
12	T	Yes	5	4	2	3	6	4	1
13	T	Yes	6	4	2	4	2	4	5
14	T	Yes	6	2	2	5	2	5	9
15	T	Yes	7	4	2	5	10	6	1
16	T	**No**	8	4	2	5	12	6	3
17	T	**No**	7	2	2	7	2	7	9
18	T	Yes	2	1	2	7	2	8	1
19	T	Yes	8	2	2	8	2	8	9
20	T	Yes	9	4	2	19	2	19	5
21	T	Yes	10	4	2	19	6	20	1

Now we indicate in the score all the significant atoms by theme, ScoreSegmenter reference number, and applied operator (see Figure 8).

We can now see some generated nets included in the generated model: the highest level net `Inv 1/Top` (see Figure 9), the net `Inv 1/Th` with the list of the recognized themes (see Figure 10), the net `Inv 1/Vc` listings all the voices (see Figure 11), the net `Inv 1/Vc1` specifying the first voice (see Figure 12). Note the presence of a rest, associated to the second place, named `Rest`.
To all the place nodes filled with the 'net pattern' we associate a subnet by means of a refinement morphism corresponding to a template net, taken from a library with a list of modifying parameters. Consider for example the segment `Inv 1/Sg1_4` (see Figure 13) and the parameter list for the macro net which corresponds to place `Obj` (see Figure 14).
The entire model hierarchy and dependency levels are shown better within the exploded list in the window of Figure 15.

Future developments

The environment for the analysis/re-synthesis of scores briefly described here has to be considered a prototype for experimental research whose potential for further subtlety in analysis phases is directly proportional to the systematic experimentation we can carry out according to the following primary purposes:
I) generalization of the segmentation capacity of the ScoreSegmenter to the most various music forms; at present, though able to decompose any score, it has more capacity, or better it produces fewer rejects, with passages in fugue or sonata form;
II) extent of the analysis/re-synthesis capacities of ModelSynth for the recognition of related structures extending the current idea of macro-net in such a way that a macro-net can be considered as a parameter of another macro-net; this possibility requires a coherent extension of the ScoreSynth module. In more general terms, we can think of an approach such as the one followed here in the case of the music scores, applied to the case of the multimedial processes, where it is possible to decompose, to organize in generative models and to synthetize processes established by sounds, images and texts.

Acknowledgements

Many thanks to F. Lonati for the design and development of the Score Segmenter program. Moreover, special thanks are due to the L.I.M.-Laboratorio di Informatica Musicale staff for the scientific and technical support provided all throughout our research.

This research has been supported by the Italian National Research Council in the frame of the MUSIC Topic (LRC C4): "INTELLIGENT MUSIC WORKSTATION", Goal C: SISTEMI AVANZATI DI PRODUTTIVITA' INDIVIDUALE, Subproject 7: SISTEMI DI SUPPORTO AL LAVORO INTELLETTUALE, Finalized Project SISTEMI INFORMATICI E CALCOLO PARALLELO.

References

Haus, G.; Sametti, A. (1992). SCORESYNTH: a System for the Synthesis of Music Scores based on Petri Nets and a Music Algebra. In D. Baggi (ed.), *Readings in Computer Generated Music* (pp.53-78). Washington: IEEE Computer Society Press.

Lonati, F. (1991a). *Guida operativa del modulo "ScoreSegmenter"*. Tech. Report M/18. Milan: CNR-PFI2 MUSIC Series.

Lonati, F. (1991b). *Note tecniche del modulo "ScoreSegmenter"*. Tech. Report M/19. Milan: CNR-PFI2 MUSIC Series.

Pighi, I. et al. (1993). *Integrazione dell'architettura e delle specifiche funzionali di alto livello dei moduli della Stazione di Lavoro Musicale Intelligente - Unità Operativa dell'Università degli Studi di Milano*, Tech. Report M/42. Milan: CNR-PFI2 MUSIC Series.

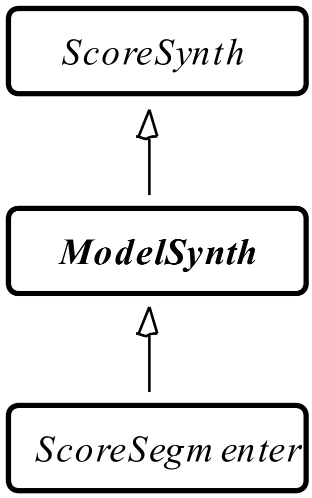

Fig. 1: Software modules architecture.

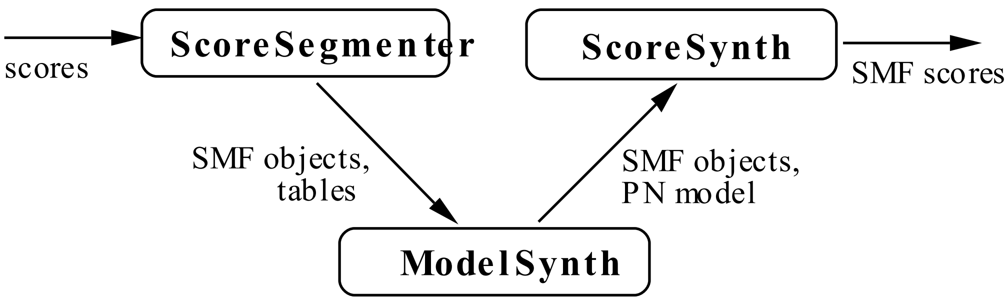

Fig. 2: Input/output flows.

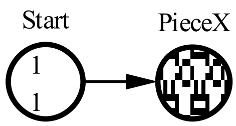

Fig. 3: The highest level net which has been generated.

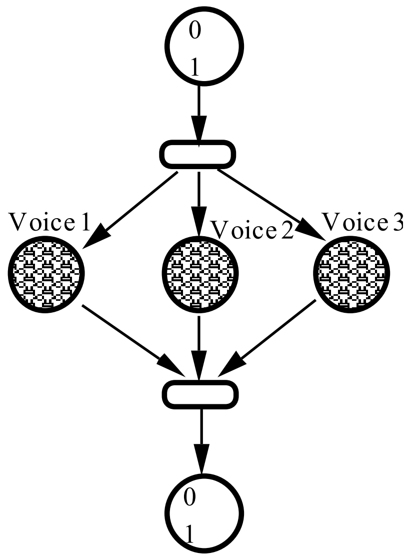

Fig. 4: The net "PieceX".

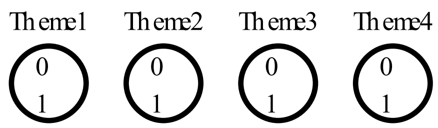

Fig. 5: The net with all the recognized themes.

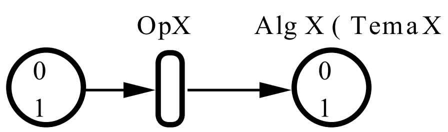

Fig. 6: An example of transposition operator applied to a theme.

INVENZIONE N°1 A DUE VOCI

PER PIANOFORTE

♩ = 92

J. S. Bach


9
10
11
12
cresc. a poco a poco
13
14
15
fp
p
16
17
18

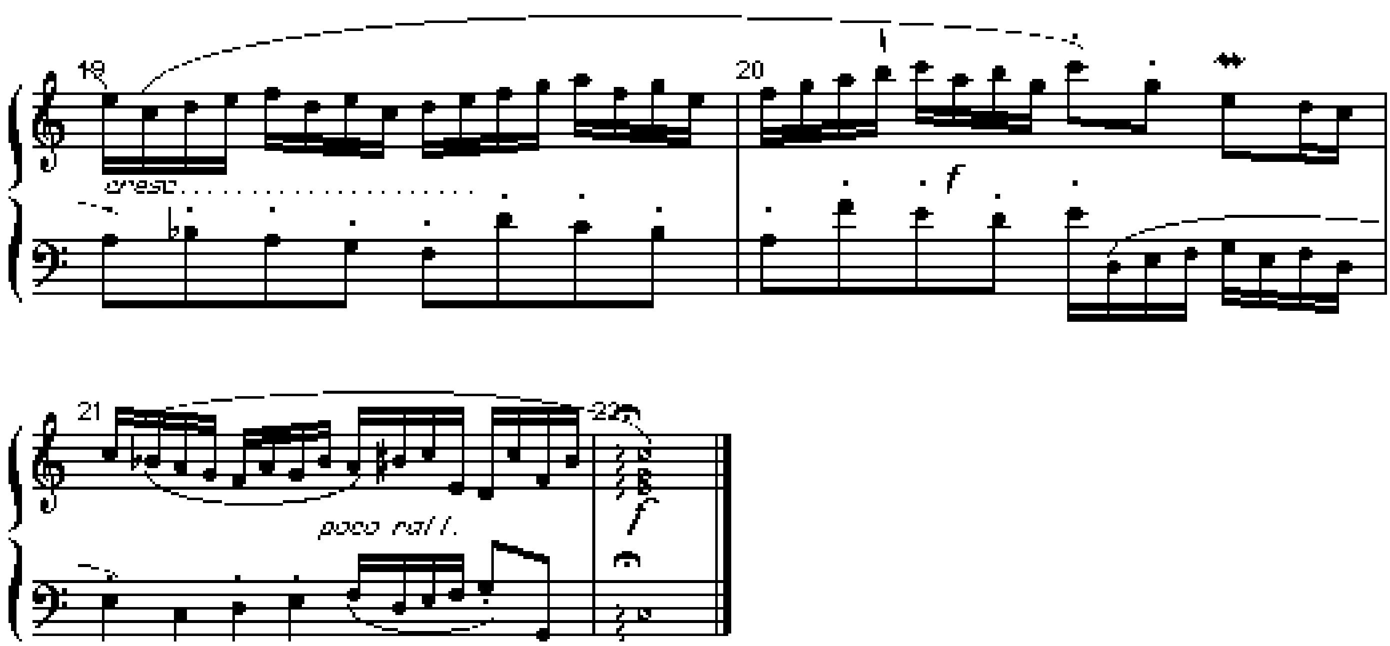

Figure 7: The original score to be analyzed.

INVENZIONE N°1 A DUE VOCI

PER PIANOFORTE

Theme4 / 1 / TT(1)
Theme4 / 2 / TT(6)
Theme4 / 3 / TT(4)
cresc. a poco a poco
fp
p

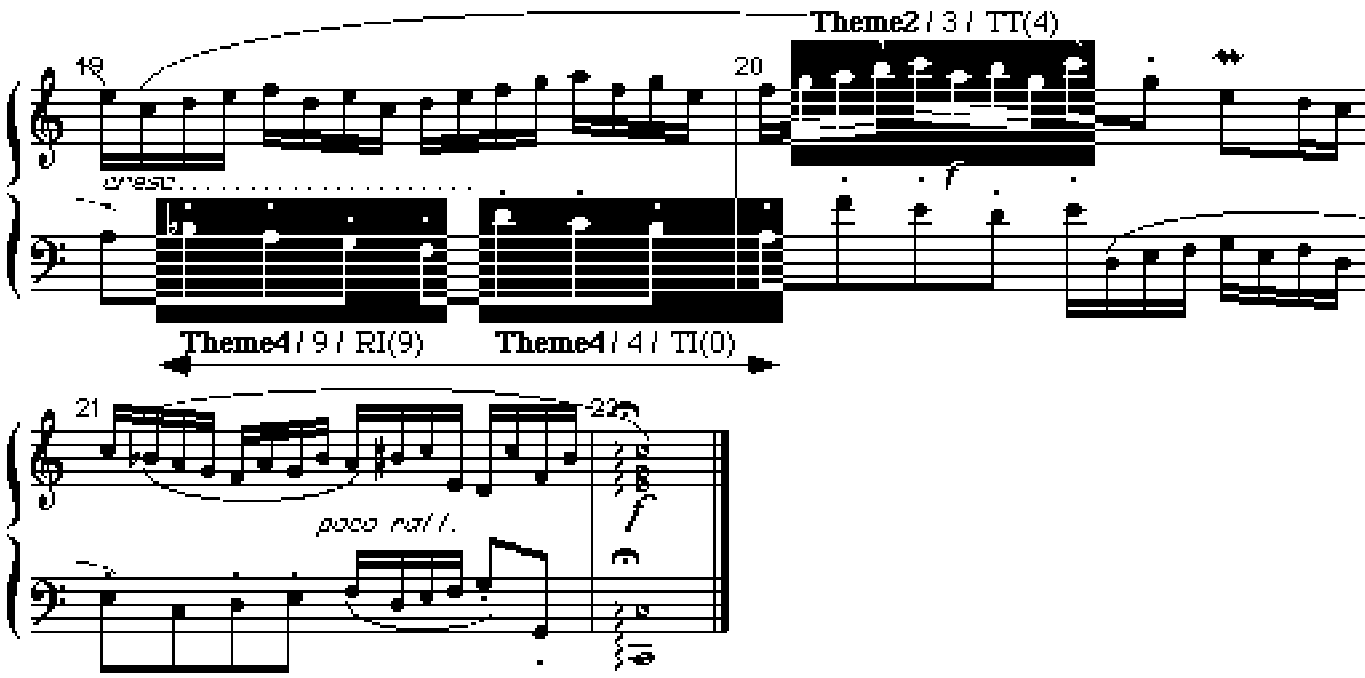

Figure 8: Music objects and operators which have been recognized by the ScoreSegmenter analysis.

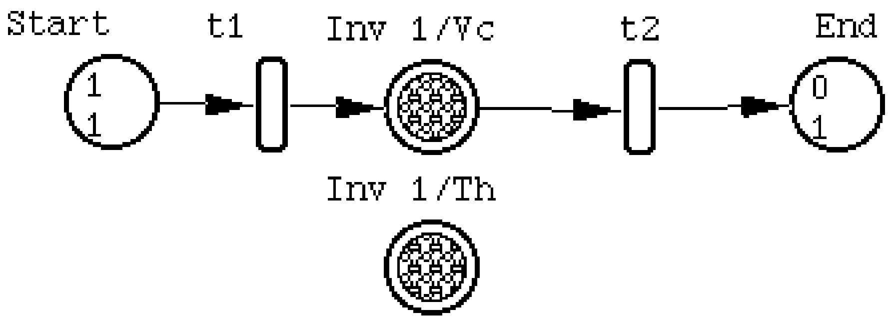

Figure 9: The highest level net `Inv 1/Top`.

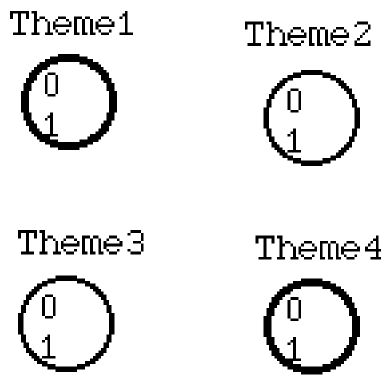

Figure 10: The net `Inv 1/Th` with the list of the recognized themes.

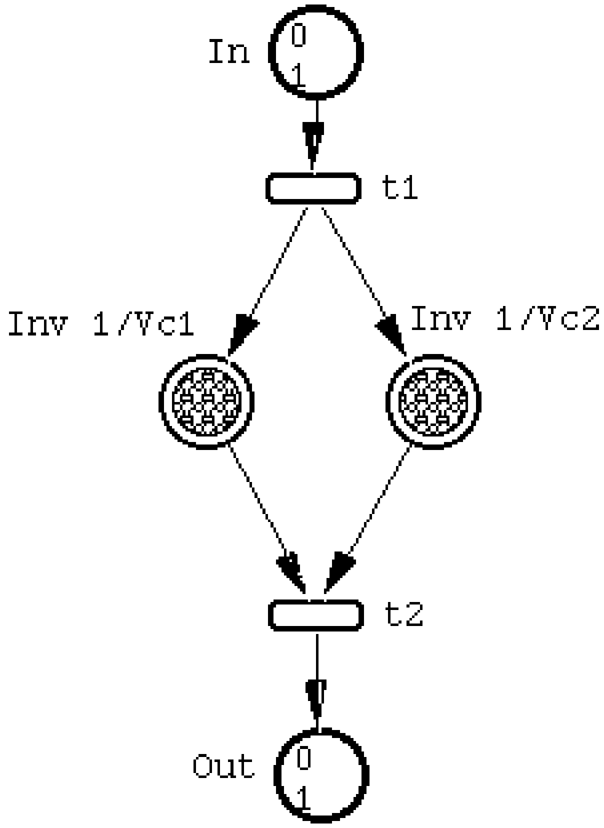

Figure 11: The net `Inv 1/Vc` listings all the voices.

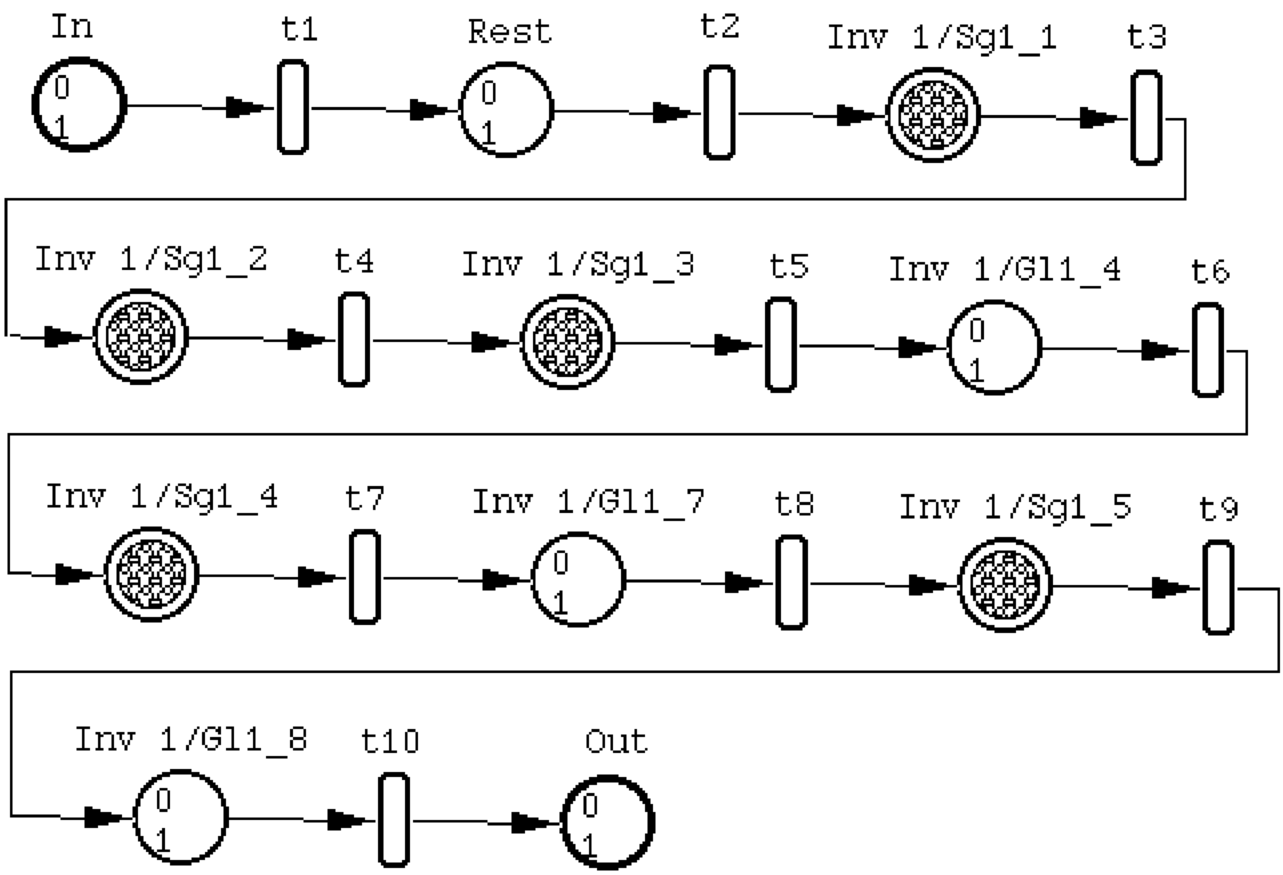

Figure 12: The net `Inv 1/Vc1` that specifies the first voice.

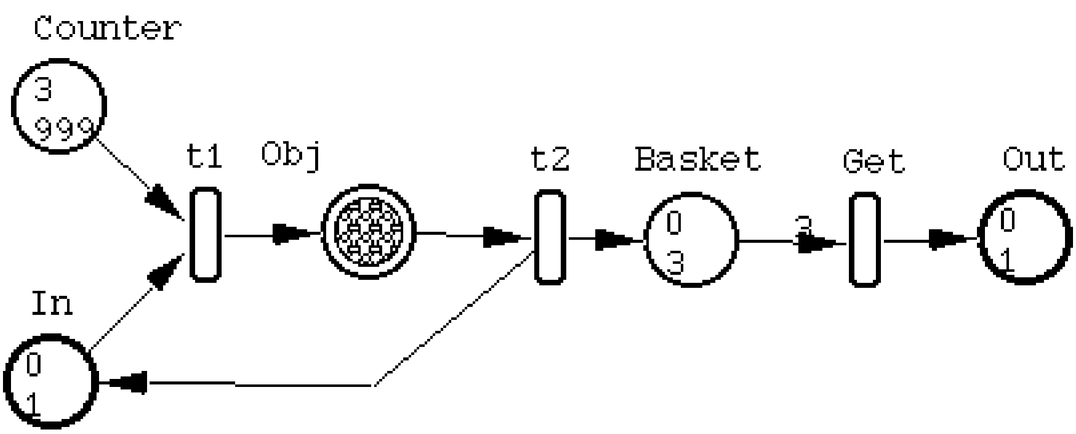

Figure 13: The template net for the segment `Inv 1/Sg1_4`.

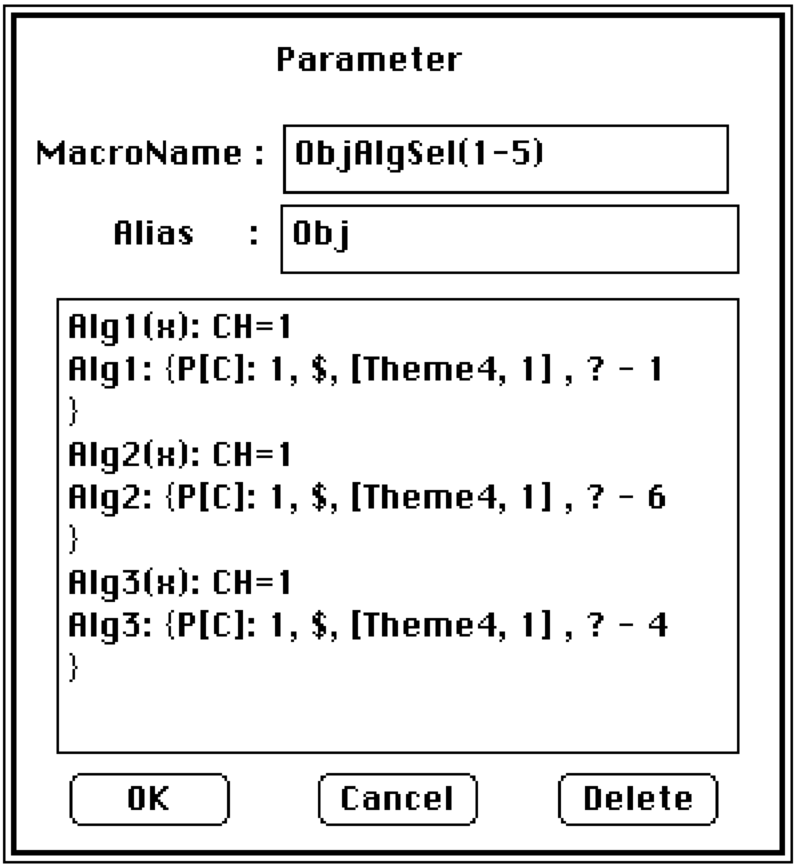

Figure 14: The parameter list for the macro net which corresponds to place `Obj`**.**

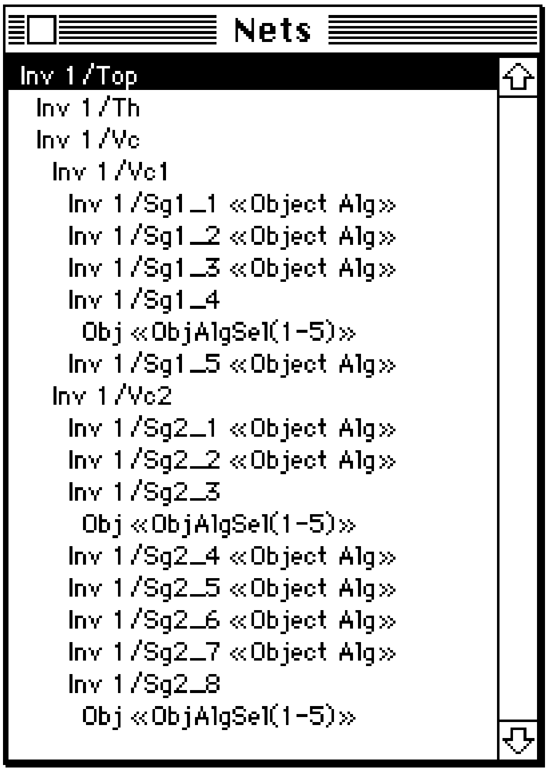

Figure 15: The window which shows the entire model hierarchy and dependency levels.