Real-time Music Composition through P-timed Petri Nets

Adriano Baratè, Goffredo Haus, Luca A. Ludovico
Dipartimento di Informatica (DI)
Università degli Studi di Milano
Via Comelico 39, 20135 Milano, Italy
{barate, haus, ludovico}@di.unimi.it

ABSTRACT

This paper introduces a new real-time concept of reconfigurable P-timed Petri nets. Our goal is to provide a formal model to build and modify a net on the fly. In the first part of the article, the original P-timed extensions are summarized. Then we define an endomorphism that alters the original Petri net in real time; for instance one can change the number of tokens or the net structure. The endomorphism is applied to Music Petri nets, showing how this new approach can be effective in real-time synthesis of music. The final case study provides a practical application by illustrating the real-time creation of a simple piano loop.

1. INTRODUCTION

The present work discusses an innovative approach to the concept of real-time modification in reconfigurable P-timed Petri nets.

Modifiable Petri nets have been already explored in a number of scientific papers, such as [1], [2], and [3]. In those cases, the main aim was to define the evolution of model properties with respect to net modifications, but the aspects related to real time were not relevant for the discussion. On the contrary, our approach takes advantage from a real-time interaction with Petri nets structure.

The first part of the paper concerns the basic theory of Petri nets. In Section 2, the original P-timed extensions are summarized, and some new features are introduced. Section 3 addresses the specific case of real-time modifications.

The aim of the second part is applying Petri nets to the music composition field.

The relationship between Petri nets and music has been explored in a number of previous scientific works. One of the milestones is [4], where the authors define how to describe and process music through Petri nets. In [5] an early software tool for the synthesis of music scores through Petri nets is presented. More recent works address the applicability of this formal tool to music analysis [6] and composition [7].

In the mentioned approaches real-time modifications [1] of the net are not supported. For analysis purposes, this is not a relevant limitation. Depending on its characteristics, an existing composition could be either easy (e.g. canons and fugues) or hard (e.g. free-jazz improvisations) to describe through Petri nets; however, in neither case the resulting Petri net requires on-the-fly adjustments.

As regards composition through Petri nets, the process can follow different approaches. A composer can conceive the structure of the whole piece *a priori*, so that real-time modifications are not required. On the other hand the composer can adopt techniques aiming at a continuous manipulation of existing fragments.

This approach is commonly accepted in some specific music styles. For example, *Minimalism* [8] is a form of experimental music strongly based on the gradual transformation of music fragments and on the reiteration of musical phrases or smaller units (e.g. figures, motifs, etc.). In this case, a Petri net could be employed to encode and mutually link smaller music entities, thus providing the basic pattern of the piece, while on-the-fly modifications could be easily applied in order to obtain gradual transformations.

Analogous processes can be applied to a more traditional context, too. For instance, Arnold Schönberg tried to approach this matter systematically in [9], where he described how to transform music entities and how to build complex structures from simpler ones. Also Heinrich Schenker in many theoretical works revealed his interest towards structures and their modifications. In [10] he states that “the act of tonal composition depends on the composer’s sense of the fundamental structure”, and “the secret of balance in music ultimately lies in the constant awareness of the transformation levels and the motion from foreground to background or the reverse”. Finally, let us cite the research by Fred Lerdahl and Ray Jackendoff about a generative theory of tonal music [11], where the concepts of *rhythmic structure*, *grouping structure*, *metrical structure* and their interconnections are detailed.

Modifiable Petri nets are fit for modelling dynamic behaviour, thus allowing the composer to modify the net on the fly, namely during the performance of the music piece. Music Petri nets will be formally discussed in Section 4, whereas the modifications supported by our model will be detailed in Section 5.

[1] Here for *real-time modifications* we mean those changes that can occur during the performance of the piece.

2. P-TIMED PETRI NETS WITH PROBABILISTIC ARC WEIGHTS

In the classical Petri net theory, places and transitions have no tempo parameters associated. When a token arrives at a place, it will be immediately ready to be transferred to outgoing transitions. The duration of a transition firing is equal to zero. For performance evaluation, a number of extensions has been introduced, associating timings with various elements of a Petri net [12] [13] [14]. Common implementations are *deterministic* and *stochastic* Petri nets [15] [16]. In the stochastic case time is modelled through probability distributions, in deterministic Petri nets time is directly associated with places, transitions and/or tokens.

In this work we use P-timed deterministic Petri nets [17], where a timing parameter is associated with places. Using this extension, when one or more tokens arrive at a given place, they are reserved for a specified interval, and only after this time lapse the outgoing transitions are enabled to (eventually) take the tokens.

Now we provide the definition of P-timed Petri nets.

Definition 1. A P-timed Petri net is a 8-tuple

$\mathbf{PN} = (\mathbf{P}, \mathbf{T}, \mathbf{A}, \mathbf{c}, \mathbf{w}_t, \mathbf{m}_0, \mathbf{w}_p, \tau)$, where:

1. $\mathbf{P}$ is a finite set of **places**
2. $\mathbf{T}$ is a finite set of **transitions**
3. $\mathbf{A}$ is a finite set of **arcs**
4. $\mathbf{c} : \mathbf{P} \to \mathbb{N}$ is the **capacity** of places
5. $\mathbf{m}_0 : \mathbf{P} \to \mathbb{N}_0$ is the **initial marking** of places
6. $\mathbf{P} \cap \mathbf{T} = \varnothing$
7. $\mathbf{P} \cup \mathbf{T} \neq \varnothing$
8. $\mathbf{A} \subseteq (\mathbf{P} \times \mathbf{T}) \cup (\mathbf{T} \times \mathbf{P})$
9. $dom(\mathbf{A}) \cup ran(\mathbf{A}) = \mathbf{P} \cup \mathbf{T}$, where
   $dom(\mathbf{A}) = \{x \in \mathbf{P} \cup \mathbf{T} \mid \exists y \in \mathbf{P} \cup \mathbf{T}, (x, y) \in \mathbf{A}\}$
   $ran(\mathbf{A}) = \{y \in \mathbf{P} \cup \mathbf{T} \mid \exists x \in \mathbf{P} \cup \mathbf{T}, (x, y) \in \mathbf{A}\}$
10. $\forall p \in \mathbf{P} : \mathbf{m}_0(p) \leq \mathbf{c}(p)$
11. $\forall (x, y) \in \mathbf{A} : (y, x) \notin \mathbf{A}$
12. $\mathbf{w}_t : \mathbf{A} \to \mathbb{N}$ is the arcs' **tokens weight**
13. $\mathbf{w}_p : \mathbf{A} \to \mathbb{N}_0$ is the arcs' **probabilistic weight**
14. $\tau : \mathbf{P} \to \mathbb{R}_0^+$ is the **timing** associated with places

In this definition items ranging from 1 to 5 define the nomenclature of the model, while the following items specify the common requirements in order that:

1. a node must be either a place or a transition;
2. the net must contain at least a node;
3. an arc must connect a place to a transition;
4. all nodes must be connected to other nodes with an arc;
5. every marking must be equal to or lower than the corresponding initial capacity;
6. if an arc connects a place to a transition, the same transition cannot be connected to the same place in the opposite direction by another arc.

The last three items introduce a nomenclature not common in all types of Petri nets:

1. the classical definition of *arc weight*, here called *tokens weight* to distinguish it from the definition that follows;
2. a new weight associated with arcs, that serves in alternative or conflict situations, to control the probability of chosing a particular transition for firing (as considered in Definition 2);
3. a number that specifies the time to wait before the tokens present in a place can be considered free to leave that place.

The corresponding firing rule must consider both the time associated with single places and the probabilistic weights of arcs:

Definition 2. Let $\mathbf{PN}$ be a P-timed Petri net, with its components denoted by
$\mathbf{P_{PN}}, \mathbf{T_{PN}}, \mathbf{A_{PN}}, \mathbf{c_{PN}}, \mathbf{w}_{t\mathbf{PN}}, \mathbf{m}_{0\mathbf{PN}}, \mathbf{w}_{p\mathbf{PN}}, \tau_{\mathbf{PN}}$.

1. $\mathbf{m}_t : \mathbf{P_{PN}} \to \mathbb{N}_0$ is called a **marking** at time t of places iff

   $$\forall p \in \mathbf{P_{PN}}\text{: } \mathbf{m}_t(p) \leq \mathbf{c_{PN}}(p),$$

   and $t - t_0(p) \geqslant \tau(p)$, where $t_0(p)$ is the time of the previous marking change.
   For the sake of clarity, in the following let $\mathbf{m}$ be a marking of $\mathbf{PN}$.
2. $\mathbf{IN}(n) = \{x \in \mathbf{P} \cup \mathbf{T} : (x, n) \in \mathbf{A_{PN}}\}$ is the set of **input nodes** of n, where $n \in \mathbf{P} \cup \mathbf{T}$
3. $\mathbf{OUT}(n) = \{y \in \mathbf{P} \cup \mathbf{T} : (n, y) \in \mathbf{A_{PN}}\}$ is the set of **output nodes** of n, where $n \in \mathbf{P} \cup \mathbf{T}$
4. A transition $t \in \mathbf{T_{PN}}$ is **enabled** iff
   $\forall p \in \mathbf{IN}(t)$: $\mathbf{m}(p) \geq \mathbf{w}_{t\mathbf{PN}}(p, t)$
   $\forall p \in \mathbf{OUT}(t)$: $\mathbf{m}(p) \leq \mathbf{c_{PN}}(p) - \mathbf{w}_{t\mathbf{PN}}(t, p)$
5. An enabled transition t may **fire**, changing the current marking $\mathbf{m}$ in $\mathbf{m}'$ such that $\forall p \in \mathbf{P_{PN}}$:
   $$\mathbf{m}'(p) = \begin{cases} \mathbf{m}(p) - \mathbf{w}_{t\mathbf{PN}}(p, t) & \iff p \in \mathbf{IN}(t) \\ \mathbf{m}(p) + \mathbf{w}_{t\mathbf{PN}}(t, p) & \iff p \in \mathbf{OUT}(t) \\ \mathbf{m}(p) \text{ otherwise} \end{cases}$$
6. $\mathbf{E}(\mathbf{m}) = \{t \in \mathbf{T_{PN}} \mid t \text{ is enabled}\}$ is the set of enabled transitions

7. $\mathbf{tw}_p(t) = \sum_{x=t \vee y=t} \mathrm{w}_p[(x,y)]$ is the **total probabilistic weight** of t

8. $\big(\exists \bar{t} \in \mathbf{E}(\mathbf{m}) \mid \mathbf{tw}_p(\bar{t}) > 0\big) \implies \forall t_i \in \mathbf{E}(\mathbf{m})$: $P(t_i) = \dfrac{\mathbf{tw}_p(t_i)}{\sum_{t \in \mathbf{E}(\mathbf{m})} \mathbf{tw}_p(t)}$ is the probability of choosing t_i as the first transition to fire

9. $\big(\forall \bar{t} \in \mathbf{E}(\mathbf{m}) : \mathbf{tw}_p(\bar{t}) = 0\big) \implies \forall t_i \in \mathbf{E}(\mathbf{m})$: $P(t_i) = \dfrac{1}{\|\mathbf{E}(\mathbf{m})\|}$ is the probability of choosing t_i as the first transition to fire

While the first concepts of the previous definition are commonly accepted, the last three items involve the new concept of *probabilistic weight*. The definitions state that, when many transitions are enabled to fire, the probability of choosing one specific transition can be calculated as follows:

- If all the enabled transitions have all the input/output arcs with probabilistic weights equal to 0, the first transition to fire is chosen randomly;
- If at least one of the enabled transitions has a probabilistic weight of the input/output arcs greater than 0, the probability of choosing one specific transition to fire is the sum of its input/output probabilistic weights divided by the sum of all the input/output probabilistic weights of all the enabled transitions.

An example of probabilistic weight is presented in Figure 1. In this net there are three transitions and an input place with only one token. **T1**, **T2**, and **T3** are enabled but as alternatives. The three incoming arcs of the transitions have different probabilistic weights, represented by numbers inside square brackets, while the outgoing arcs have the same probabilistic weight equal to 0 (omitted by convention). In this case the probabilities to fire are:

$$P(\mathbf{T1}) = \frac{2+0}{2+0+3+0+100+0} = \frac{2}{105} = 1.9\%$$

$$P(\mathbf{T2}) = \frac{3+0}{2+0+3+0+100+0} = \frac{3}{105} = 2.9\%$$

$$P(\mathbf{T3}) = \frac{100+0}{2+0+3+0+100+0} = \frac{100}{105} = 95.2\%$$

It must be noted that the probability of choosing one of the firing transitions dynamically changes with the evolution of the net. In the previous example, let us consider the net as a part of a more complex one, with tokens arriving many times at place **P1**; case by case not all the transitions could be enabled: if **P4** has reached its capacity, it cannot be considered for firing, and the probabilities of choosing either **T1** or **T2** change respectively to $\frac{2}{5} = 40\%$ and to $\frac{3}{5} = 60\%$.

The probabilistic weight is used in this work when: *i)* two or more transitions are in alternative, but *ii)* we want to follow a specific path until a particular event occurs.[2] To accomplish this, the last transition to fire must have all the input/output arcs with probabilistic weights equal to 0. An implementation of a loop structure that uses this concept is presented in Figure 2. In this model the arc connecting **InP** to **InT** has a probabilistic weight set to 1. From the point of view of **InP**, there is an alternative situation every step of the loop, since it has to choose what transition can fire between **InT** and **ExitT**. The probabilistic weight resolves this non-deterministic situation, always choosing **InT** instead of **ExitT**. When the loop process is completed, the **Counter** place is empty, and **InT** is no more enabled. Only in this case, since no other transition is enabled, **ExitT** fires – even if it has an associated arc with probabilistic weight of zero – thus exiting the loop.

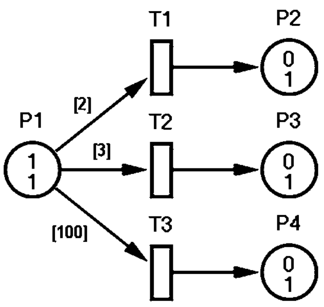

Figure 1. An example of probabilistic arc weights.

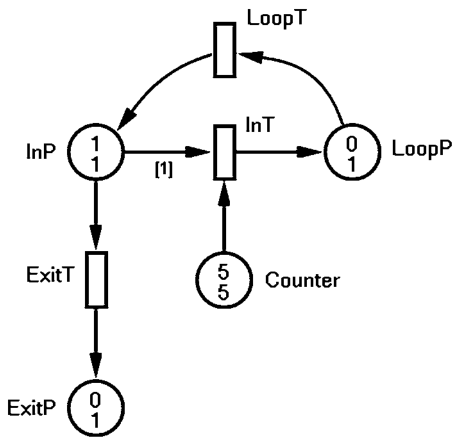

Figure 2. An example of use of probabilistic arc weights of value 0.

3. REAL-TIME MODIFICATIONS

This section focuses on how Petri nets can be used in a real-time environment by changing net parameters on the fly, i.e. when a model is being executed. All supported modifications to Petri nets affect the model itself and the firing rule from a theoretical point of view.

[2] For this goal, transition priority could be used instead of probabilistic weight, but this concept is far more versatile. For example, in other cases probabilistic weight could be employed to implement a non-deterministic net.

Now we will list the possible real-time modifications [3] of a Petri net:

- modification of place marking (PM): when adding or subtracting a number of tokens in a place, the place capacity is automatically incremented - if needed - to contain the new number of tokens;
- modification of place capacity (PC): the marking is automatically decremented if the specified capacity is less than the number of tokens contained in the place;
- modification of arcs' tokens weight (ATW);
- modification of arcs' probabilistic weight (APW);
- modification of the set of places (PP): if a place is added, by default the new place has marking equal to 0, capacity equal to 1, and timing equal to 0; if a place is removed, all its input/output arcs are removed too;
- modification of the set of transitions (TT): if a transition is removed, all its input/output arcs are removed too;
- modification of the set of arcs (AA): if an arc is added, by default the new arc has a tokens weight equal to 1 and a probabilistic weight equal to 0.

Modifications can happen at any time, and they must be considered atomic. When a modification occurs, the transition firing rule must be instantly applied, as the new parameters could have created new firing conditions. The possible modifications of the net occur in zero time.

Since a real-time Petri net can be constructed from scratch, conditions 7 and 9 of Definition 2 must not be considered during the real-time modification of parameters. Thus, a Petri net can either be empty, or have unconnected nodes.

After these considerations, we are ready to provide a formal definition of real-time modifications:

Definition 3. Let $\mathbf{PN}$ be a P-timed Petri net, with its components at a certain time denoted by
$\mathbf{P}, \mathbf{T}, \mathbf{A}, \mathbf{c}, \mathbf{w}_t, \mathbf{m}_0, \mathbf{w}_p, \tau$.
A **real-time Petri net modification** is an endomorphism $\mathbf{RTM} \in \{\mathbf{PM}, \mathbf{PC}, \mathbf{ATW}, \mathbf{APW}, \mathbf{PP}, \mathbf{TT}, \mathbf{AA}\}$ of $\mathbf{PN}$ where

1. $\mathbf{PM}(\mathbf{PN},p,k) = (\mathbf{P}, \mathbf{T}, \mathbf{A}, \mathbf{c}', \mathbf{w}_t, \mathbf{m}'_0, \mathbf{w}_p, \tau)$, where $\mathbf{m}'_0(p) = k$, and $\mathbf{c}'(p) = max\,(\mathbf{c}(p), k)$

2. $\mathbf{PC}(\mathbf{PN},p,l) = (\mathbf{P}, \mathbf{T}, \mathbf{A}, \mathbf{c}', \mathbf{w}_t, \mathbf{m}'_0, \mathbf{w}_p, \tau)$, where $\mathbf{c}'(p) = l$, and $\mathbf{m}'_0(p) = min\,(\mathbf{m}_0(p), l)$

3. $\mathbf{ATW}(\mathbf{PN},a,m) = (\mathbf{P}, \mathbf{T}, \mathbf{A}, \mathbf{c}, \mathbf{w}'_t, \mathbf{m}_0, \mathbf{w}_p, \tau)$, where $\mathbf{w}'_t(a) = m$

4. $\mathbf{APW}(\mathbf{PN},a,n) = (\mathbf{P}, \mathbf{T}, \mathbf{A}, \mathbf{c}, \mathbf{w}_t, \mathbf{m}_0, \mathbf{w}'_p, \tau)$, where $\mathbf{w}'_p(a) = n$

5. $\mathbf{PP}(\mathbf{PN},p) = (\mathbf{P}', \mathbf{T}, \mathbf{A}', \mathbf{c}', \mathbf{w}_t, \mathbf{m}'_0, \mathbf{w}_p, \tau')$, where
   (a) if p $\notin \mathbf{P}$
      i. $\mathbf{P}' = \mathbf{P} \cup \{p\}$
      ii. $\mathbf{A}' = \mathbf{A}$
      iii. $\mathbf{c}'(p) = 1$
      iv. $\mathbf{m}'_0(p) = 0$
      v. $\tau'(p) = 0$
   (b) if p $\in \mathbf{P}$
      i. $\mathbf{P}' = \mathbf{P} \setminus \{p\}$
      ii. $\mathbf{A}' = \mathbf{A} \setminus \{ \bigcup_{x=p \vee y=p} (x, y)\}$
      iii. $\mathbf{c}' = \mathbf{c}$
      iv. $\mathbf{m}'_0 = \mathbf{m}_0$
      v. $\tau' = \tau$

6. $\mathbf{TT}(\mathbf{PN},t) = (\mathbf{P}, \mathbf{T}', \mathbf{A}', \mathbf{c}, \mathbf{w}_t, \mathbf{m}_0, \mathbf{w}_p, \tau)$, where
   (a) if t $\notin \mathbf{T}$
      i. $\mathbf{T}' = \mathbf{T} \cup \{t\}$
      ii. $\mathbf{A}' = \mathbf{A}$
   (b) if t $\in \mathbf{T}$
      i. $\mathbf{T}' = \mathbf{T} \setminus \{t\}$
      ii. $\mathbf{A}' = \mathbf{A} \setminus \{ \bigcup_{x=t \vee y=t} (x, y)\}$

7. $\mathbf{AA}(\mathbf{PN},(x,y)) = (\mathbf{P}, \mathbf{T}, \mathbf{A}', \mathbf{c}, \mathbf{w}'_t, \mathbf{m}_0, \mathbf{w}'_p, \tau)$, where
   (a) if $(x, y) \notin \mathbf{A}$
      i. $\mathbf{A}' = \mathbf{A} \cup \{(x, y)\}$
      ii. $\mathbf{w}'_t[(x, y)] = 1$
      iii. $\mathbf{w}'_p[(x, y)] = 0$
   (b) if $(x, y) \in \mathbf{A}$
      i. $\mathbf{A}' = \mathbf{A} \setminus \{(x, y)\}$
      ii. $\mathbf{w}'_t = \mathbf{w}_t$
      iii. $\mathbf{w}'_p = \mathbf{w}_p$

for every value of the parameters $p, t, (x, y), k, l, m, n$, varying in the following sets: $p \in \mathbf{P}$; $t \in \mathbf{T}$; $(x, y) \in \mathbf{A}$; $k, n \in \mathbb{N}_0$; $l, m \in \mathbb{N}$.

As regards the notation adopted in the previous definition, please note that each modification step creates a new net that can be seen as the original one for a possible further modification.

[3] Please note that all the modifications are done when a net is executing, and not at design time.

4. MUSIC PETRI NETS

At LIM (Laboratorio di Informatica Musicale) of the Università degli Studi of Milan, Petri nets have been applied to the music field since 1982. In particular, early papers [18] investigated the possibility of describing causality in music processes through the formal approach of Petri nets, while only recent studies focus on music creation. Different applications of this formalism to music analysis do lead to contradictory results depending on the repertoire. On one side Ravel's *Bolero* has been well modelled as in [19], but on the other some limitations became evident with a complex work such as Stravinsky's *Rite of Spring* [20].

In this section we summarize our approach on using P-timed Petri nets in music contexts, partly derived from past applications. The first definitions to be clarified are the concepts of *music objects* and *music algorithms*.

A music object is anything carrying a music meaning, including a note, a sequence of notes, rests, device-control commands. It must be clear that at this level of abstraction implementations of music objects are not important: e.g. sequences of notes can be expressed in terms of MIDI commands, MP3 files, textual representations, and so on; but for the goals of this work, we do not care.

While music objects represent music entities of some kind, a music algorithm is whatever function applicable to such objects. Music algorithms include not only well-known transformations of music fragments, such as transposition, retrogradation, inversion, but also loudness control, instrument change, complex mathematical functions.

In our model *music objects* can be associated with places and *music algorithms* to transitions. A particular parameter – set place by place – indicates if the associated music object has to be played, or only transferred to the output transitions. The following rules apply when a Music Petri net is executed:

- When a place **P** receives *n* tokens from an input transition **T**:
  - If **P** has an associated music object **MO** of duration t^* and the *playing* parameter is set:
    - *n* simultaneous executions of **MO** are played and $\tau(p) = t^*$ (i.e. while playing, the new tokens cannot be considered for firing);
    - after the end of the performance (i.e. when $t - t_0(p) \geqslant \tau(p)$), the *n* tokens are free to leave **P**;
    - **MO** is passed to output transitions.
  - If **P** has an associated music object **MO** and the *playing* parameter is not set:
    - the *n* tokens are free to leave **P**;
    - **MO** is passed to output transitions.
  - If the place has no associated music objects:
    - If **T** has a music object **MO** of duration t^* in output:
      - **MO** is retrieved from **T**;
      - *n* simultaneous executions of **MO** are played and $\tau(p) = t^*$ (i.e. while playing, the new tokens cannot be considered for firing);
      - after the end of the performance (i.e. when $t - t_0(p) \geqslant \tau(p)$), the *n* tokens are free to leave **P**;
      - **MO** is passed to output transitions.
    - If **T** has no music objects in output:
      - the *n* tokens are free to leave **P**.
- When a transition **T** fires and receives n_1, n_2, ...,n_m tokens from m input places **P**$_1$,**P**$_2$, ...,**P**$_m$, possibly containing music objects **MO**$_1$,**MO**$_2$, ...,**MO**$_m$:
  - if T has an associated music algorithm **MA**, it is applied to input music objects;
  - the *k* non-empty input music objects, modified by **MA,** are mixed, thus obtaining a new music object **MO**;
  - **MO** is ready to be passed to all the outgoing places.

Further details on Music Petri nets can be found in [6] and [7].

5. REAL-TIME MUSIC PETRI NETS

In the field of Music Petri nets, real-time modifications generate changes in the produced music. In this section we show how this is accomplished. Let us focus on some of the modification types introduced in the Section 3 and on the changes related to music objects and music algorithms:

- modification of place marking (PM): if the place does not contain an associated music object, nothing happens in terms of music performance. If a music object is present and *n* tokens are added, *n* new istances of that object are played, while if *m* tokens are subtracted, *m* current playing music objects are stopped;
- modification of place capacity (PC): if the capacity is decremented, the considerations about place marking modifications must be applied;
- modification of the set of places (PP): if a playing place is removed, the execution stops and the set of enabled transitions is evaluated again;
- modification of associated music objects (MO): if the place has $n > 0$ tokens, the current music-object performances, if present, are stopped, and *n* new instances of the new music object are executed;
- modification of associated music algorithms (MA).

In general terms, the only immediate effect of these modifications is the performance of *n* new music objects, whereas most changes modify the net structure but their effects are produced while the net execution is running.

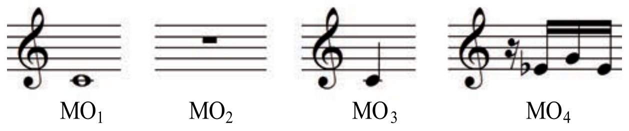

Figure 3. The original fragments MO_1 and MO_2, and the derived objects MO_3 and MO_4.

By using this new paradigm a composer/performer can create music in real-time, by mixing and altering a set of pre-prepared music objects. In this manner, one can concentrate on the structure of the music piece, at a higher level of abstraction in respect of notes and rests. Basically this leads to a wide set of possibilities ranging between two opposite approaches:

- a complex Music Petri net model is constructed in advance and the performer changes some of its characteristics while music is playing;
- a Music Petri net is built from scratch, starting with an empty model and creating a complex model step by step.

6. CASE STUDY: A PIANO LOOP

In this section we will describe a case study addressing real-time modifications of Petri nets applied to music scores. In particular, the creation of a simple piano loop will be discussed.

A relevant aspect is the intentionally basic toolkit employed in this example. It is limited as regards:

- the number of starting music objects, including only MO_1, i.e. a whole note, and MO_2, i.e. a whole rest (see Figure 3);
- the number of music algorithms, embracing only 3 melodic operators (MA_{m1}: "transpose one minor third up", MA_{m2}: "transpose one perfect fifth up", MA_{m3}: "transpose one octave down") and 1 rhythmical operator (MA_r: "divide-by-4").

Having a limited number of elements to manage is important for an easy interaction with the interface. Yet this toolkit proves to be sufficiently complete to provide both flexibility and variety to the final result. In fact such basic elements can be used to build Petri nets that are more and more complex, supporting multiple voices melodically and rhythmically independent. Moreover, a number of modifications can occur in real time, so that the user can build a (potentially) complex music performance by applying (potentially) simple processes.

The following example will illustrate such concepts by showing some of the modification techniques introduced in Section 5.

First, we want to create some basic patterns starting from the previous ones. This process can be performed in real time as well as in a setup phase. Now let us consider the former case, but in the following the latter will be addressed too.

The rhythmical value of MO_1 is reduced from a whole to a quarter note by applying MA_r, thus obtaining MO_3. Another basic pattern, namely MO_4, can be built from a suitable combination of MO_1 and MO_2. The desired rhythmic pattern is a quadruplet made by 1 rest followed by 3 notes, whose total duration is a quarter note. Both MO_1 and MO_2 undergo a double application of MA_r, so that they finally represent a sixteenth note and a sixteenth rest. As regards the pitch of the former fragment, it is transposed up by MA_{m1} to obtain both the first note and the third note, and by MA_{m2} for the second note. The quadruplet will be identified as fragment MO_4, which in terms of Petri nets represents a subnet. Needless to say, in our approach also subnets can be dynamically modified, as illustrated below. Also MO_3 and MO_4 are shown in Figure 3.

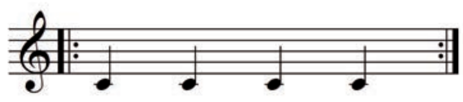

Figure 4. The 4-times repetition of music fragment MO_3.

Now we will describe a real-time compositional process based on the fragments previously prepared. The first step consists in the 4-times repetition of MO_3, with no further melodic nor rhythmical modification. Figure 4 shows the music score referring to an undefined number of iterations, and Figure 5 presents the corresponding Petri net. The number of tokens in **P1** allows to control the number of repetitions for the whole structure: in this case it is set to 3. **P1** lets the user achieve a basic on-the-fly modification of the net, even if its topology does not change: by adding tokens, the number of repetitions will be increased accordingly.

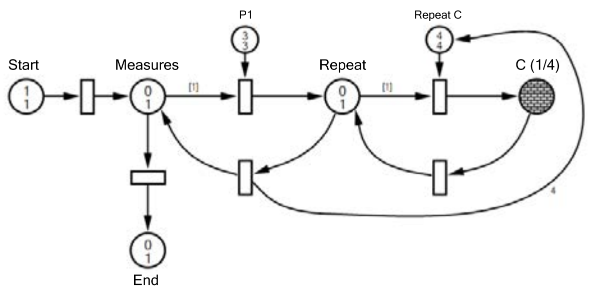

Figure 5. The Petri net for 3 iterations of the 4-times repetition of music fragment MO_3.

In order to understand the graphical representation of the net, it is worth to recall some conventions exposed in [6]. Places can present three different background colors: white for empty places, i.e. places with no MO assigned; solid gray for places containing MOs; gray pattern for subnets. For instance, Figure 5 presents a subnet that subsumes the fragment shown in Figure 6.

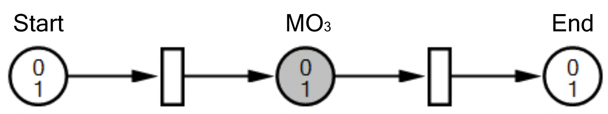

Figure 6. Single performance of music object MO_3.

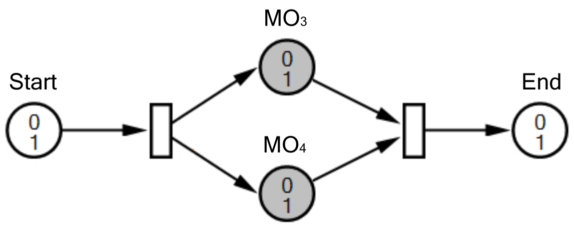

Figure 7. Concurrent performance of music object MO_3 and MO_4.

In our scenario, the execution of the net has started and we want to add other voices on the fly. This kind of interaction is very simple to be achieved even in a real-time performance environment. For instance, the subnet shown in Figure 6 is modified by connecting also the fragment MO_4, as shown in Figure 7. Similarly, the musician can add a lower voice made of whole notes, namely fragments obtained by transposing MO_1 one octave below through MA_{m3}. In this case, the composer has changed net topology on the fly. What we have described results into the net of Figure 8, and the corresponding music score is provided in Figure 9.

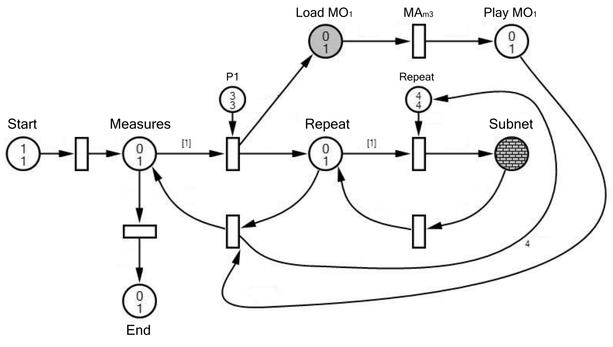

Figure 8. The Petri net for 3 iterations of the enriched music fragment.

Another kind of modification that can occur involves the behaviour of music algorithms. For example, let us redefine the meaning of MA_{m3} at each loop iteration: at step 1 MO_1 is transposed an octave below, at step 2 a major ninth below, and at step 3 a minor tenth below. The resulting score is provided in Figure 10.

Finally, let us change the music content of MO_2 in real time, for instance changing the rest with an F-pitched note during the third loop iteration. The result is shown in Figure 11.

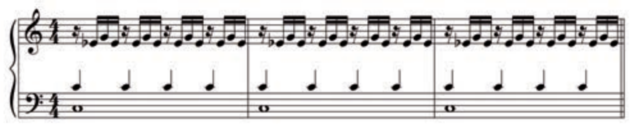

Figure 9. The resulting music score for Petri net in Figure 8.

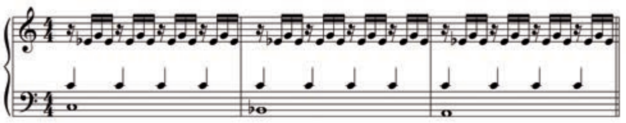

Figure 10. The effect of real-time MA_{m3} redefinitions.

7. CONCLUSIONS

In this work modifiable Petri nets have been introduced from a formal point of view and then applied to music composition. Real-time modifications of Petri nets can occur at different level, influencing not only place marking but even their topology. Needless to say, the possibility to interact with net structure, coupled to a number of already known features (concurrent processes, probabilistic weights, etc.), provides a user with a powerful tool to modify the model on the fly. In the music field, this is a relevant feature for composers who manipulate music information.

The final case study has briefly shown some of the possible modifications that are easy to be achieved in a real-time environment.

As regards future works, since Petri nets are a formalism usually far from the way of thinking of a traditional composer, software tools should be designed and developed to implement a musician-oriented interface.

8. REFERENCES

[1] J. Barros and L. Gomes, “Net model composition and modification by net operations: a pragmatic approach,” in *Industrial Informatics, 2004. INDIN’04. 2004 2nd IEEE International Conference on.* IEEE, 2004, pp. 309–314.

[2] H. Ehrig and J. Padberg, “Graph grammars and Petri net transformations,” *Lectures on Concurrency and Petri Nets*, pp. 65–86, 2004.

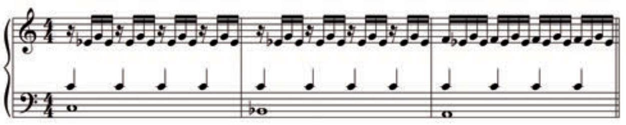

Figure 11. The effect of a real-time MO_2 redefinition at measure 3.

[3] H. Ehrig, K. Hoffmann, J. Padberg, U. Prange, and C. Ermel, "Independence of net transformations and token firing in reconfigurable place/transition systems," *Petri Nets and Other Models of Concurrency–ICATPN 2007*, pp. 104–123, 2007.

[4] G. Haus and A. Rodriguez, "Music description and processing by Petri nets," *Advances in Petri Nets 1988*, pp. 175–199, 1988.

[5] G. Haus and A. Sametti, "Scoresynth: a system for the synthesis of music scores based on Petri nets and a music algebra," *Computer*, vol. 24, no. 7, pp. 56–60, 1991.

[6] A. Baratè, G. Haus, and L. Ludovico, "Music analysis and modeling through Petri nets," *Computer Music Modeling and Retrieval*, pp. 201–218, 2006.

[7] A. Baratè, G. Haus, and L. Ludovico, "Petri nets applicability to music analysis and composition," in *Proceedings of the International Computer Music Conference'07 (ICMC 2007)*, 2007, pp. 97–100.

[8] K. Potter, *Four musical minimalists: La Monte Young, Terry Riley, Steve Reich, Philip Glass*. Cambridge University Press, 2002, vol. 11.

[9] A. Schoenberg, G. Strang, and L. Stein, *Fundamentals of musical composition*. Faber & Faber, 1967.

[10] H. Schenker and O. Jonas, "Der freie Satz," 1956.

[11] F. Lerdahl and R. Jackendoff, *A generative theory of tonal music*. MIT Press, 1996.

[12] C. Ramchandani, "Analysis of asynchronous concurrent systems by timed Petri nets," Massachusetts Institute of Technology, Tech. Rep., 1974.

[13] J. Wang, *Timed Petri nets: Theory and application*. Kluwer Academic Publishers Norwell, 1998, vol. 39.

[14] W. Zuberek, "Timed Petri nets and preliminary performance evaluation," in *Proceedings of the 7th annual symposium on Computer Architecture*. ACM, 1980, pp. 88–96.

[15] D. Kartson, G. Balbo, S. Donatelli, G. Franceschinis, and G. Conte, *Modelling with generalized stochastic Petri nets*. John Wiley & Sons, Inc., 1994.

[16] M. K. Molloy, "Performance analysis using stochastic petri nets," *Computers, IEEE Transactions on*, vol. 100, no. 9, pp. 913–917, 1982.

[17] J. Coolahan Jr and N. Roussopoulos, "Timing requirements for time-driven systems using augmented Petri nets," *IEEE Transactions on Software Engineering*, vol. SE-9, no. 5, pp. 603–616, 1983.

[18] G. Degli Antoni and G. Haus, "Music and causality," in *Proceedings of 1982 International Computer Music Conference*, 1983, pp. 279–296.

[19] G. Haus and A. Rodriguez, "Formal music representation; a case study: the model of Ravel's Bolero by Petri nets," *Music Processing. Computer Music and Digital Audio Series*, pp. 165–232, 1993.

[20] A. De Matteis and G. Haus, "Formalization of generative structures within Stravinsky's "The rite of spring"," *Journal of New Music Research*, vol. 25, no. 1, pp. 47–76, 1996.