ADRIANO BARATÈ
LUCA A. LUDOVICO
GIANCARLO VERCELLESI

RETI DI PETRI, MUSICA E MULTIMEDIALITÀ

L.I.M. - Laboratorio di Informatica Musicale
DI*Co* - Dipartimento di Informatica e Comunicazione
Università degli Studi di Milano
Via Comelico, 39/41
I-20135 Milano (Italy)

INDICE

CAPITOLO 1

LE RETI DI PETRI

1.1 INTRODUZIONE

La descrizione e l'elaborazione della musica richiedono strumenti formali adatti alla rappresentazione di iterazioni, concorrenza, ordinamento, gerarchia, causalità, temporizzazione, sincronismo, non determinismo [HR88].

La notazione tradizionale è destinata alla comunicazione immediata dell'informazione durante l'esecuzione del pezzo. I simboli vengono scelti e organizzati nella partitura in accordo con le tecniche strumentali delle parti. Osserviamo che il livello di rappresentazione all'interno della partitura è più dettagliato che nell'attività compositiva. Infatti, se la notazione su accollature e pentagrammi è costituita da sequenze parallele di note, le strutture compositive si nascondono piuttosto all'interno di tali sequenze.

Le Reti di Petri, che verranno ora introdotte e definite, rappresentano uno strumento in grado di descrivere ed elaborare oggetti musicali all'interno di ambienti di analisi, di composizione e di esecuzione. Dopo aver illustrato le basi teoriche e le convenzioni grafiche proprie delle Reti di Petri, mostreremo come le strutture musicali possano essere evidenziate e rielaborate tramite tale tipo astratto di rappresentazione.

La presente trattazione, rivolta soprattutto ai musicisti e finalizzata all'utilizzo degli strumenti software sviluppati presso il L.I.M., presenta dapprima in modo informale le Reti di Petri. Saranno poi esposte alcune definizioni formali sull'argomento, il cui scopo è consolidare le nozioni acquisite in maniera intuitiva. Passeremo infine a considerare le molteplici applicazioni delle Reti di Petri in ambito multimediale.

Il lettore più interessato potrà comunque trovare trattazioni più rigorose nell'ampia bibliografia dedicata all'argomento. In particolare, i rudimenti della teoria sulle Reti di Petri sono esposti in modo chiaro in [Pet76], [Pet81], [Rei00]. I restanti riferimenti bibliografici sono dedicati ad aspetti particolari delle Reti in ambito musicale o multimediale, e nel corso della trattazione si farà ampio ricorso ad essi.

1.2 DEFINIZIONI DI BASE

Una Rete di Petri è un modello astratto e formale atto a rappresentare la dinamica di un sistema che esibisce attività asincrone e concorrenti [Sam88].

Le Reti di Petri (d'ora in avanti chiamate PN, abbreviazione di *Petri Net*) consistono di un insieme di oggetti elementari: *posti*, *transizioni* ed *archi*, rappresentati graficamente da cerchi, rettangoli e linee orientate. I posti e le transizioni vengono anche detti *nodi*. Il nodo di partenza di un arco si dice *nodo di entrata*, mentre quello d'arrivo *nodo di uscita*.

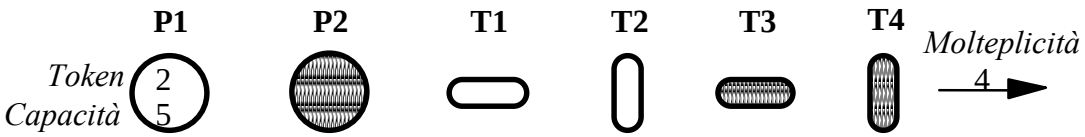

Fig. 1. Simboli grafici usati per le PN

La Fig. 1 mostra i simboli che vengono solitamente utilizzati nelle rappresentazioni grafiche delle PN. Da sinistra a destra, vengono indicati rispettivamente: un posto semplice di nome **P1**, con attributi che indicano il numero di *token* presenti (definiti più avanti) e la capacità del posto; una sottorete di nome **P2**; una transizione orizzontale semplice di nome **T1**; una transizione verticale semplice di nome **T2**; una transizione orizzontale **T3** che rappresenta una sottorete; una transizione verticale **T4** che rappresenta una sottorete; e infine un arco con peso (o molteplicità) 4.

Questi elementi si combinano tra loro a formare una PN, con l'accortezza di rispettare alcuni vincoli che verranno elencati (e definiti formalmente) più avanti.

In riferimento alla Fig. 2: **P1**, **P2** e **P3** sono posti, **T1**, **T2** e **T3** transizioni. **P1** è un posto di ingresso delle transizioni **T1** e **T2**; **T3** è una transizione d'uscita di **P2** e **P3**.

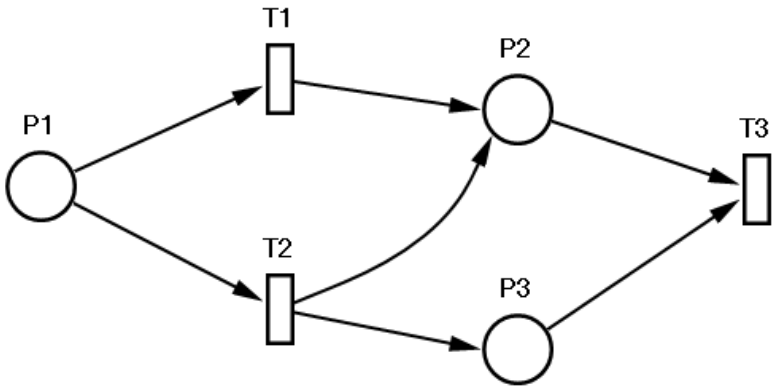

Fig. 2. Esempio di PN

Un primo vincolo delle PN è il seguente: un arco può connettere solo nodi di tipo diverso. Dunque, gli archi collegano un posto e una transizione o una transizione e un posto (come mostrato in Fig. 2), ma mai nodi omonimi: gli archi non connettono mai due posti o due transizioni.

Due nodi possono essere connessi da più di un arco dello stesso orientamento. Si tratta della situazione sintetizzata in Fig. 3, ove tra **P1** e **T1** dovrebbero essere tracciati 5 archi. Una scorciatoia per indicare che esistono *n* archi tra due nodi consiste nel tracciare un unico arco e associargli un numero positivo, detto *peso* o *molteplicità*.

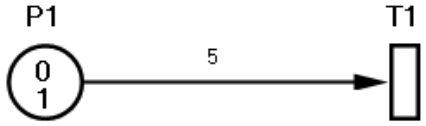

Fig. 3. Arco di peso 5

Un elemento proprio delle PN che consente loro di automodificarsi ed evolvere è il concetto di *marca* o *marcatura*: ad ogni posto viene associato un numero di marche presenti (In Fig. 4 il posto **P1** ha inizialmente 2 marche, **P2** ne ha solo 1). Il vocabolo inglese utilizzato per contrassegnare le marche è *token* (letteralmente, gettone). Talvolta il formalismo grafico consiste nel disegnare all'interno del posto dei veri e propri cerchi anneriti che rappresentano i gettoni. Ciò conferisce immediatezza alla rappresentazione, ma diventa scomodo nel caso di numerose marche o di trattazione informatica del problema. Nella presente trattazione preferiamo attenerci alle convenzioni di Fig. 1.

Le marche possono essere spostate di posto in posto seguendo determinate regole (*firing rules*), e il sistema in tal modo evolve. Le regole che governano tale evoluzione dinamica sono le seguenti.

- Se tutti i posti di ingresso di una transizione hanno un numero di marche maggiore o uguale al peso dei rispettivi archi in ingresso, la transizione si dice *abilitata allo scatto*.
- Se una transizione è abilitata allo scatto, l'esecuzione dello scatto toglierà dai posti in ingresso un numero di marche pari al peso dell'arco in ingresso ed aggiungerà ad ogni posto in uscita tante marche quanto è il peso dell'arco in uscita (si veda Fig. 4).

L'esecuzione di una rete è costituita da scatti di transizioni. Dopo l'avvio del processo, gli scatti si susseguono finché nessuna transizione è più abilitata a scattare, momento in cui l'esecuzione cessa.

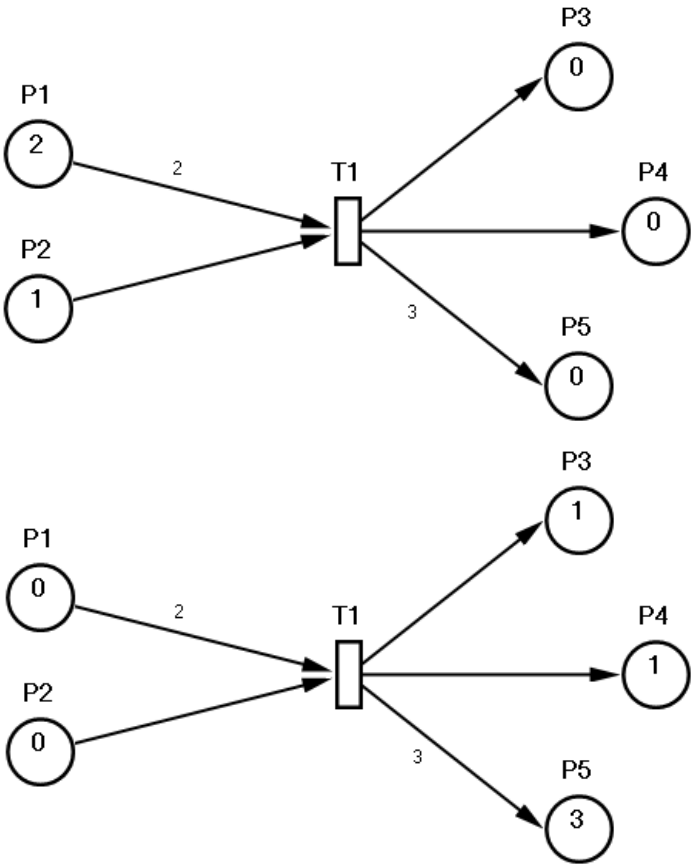

Fig. 4. Scatto di una transizione

Osserviamo che se una transizione è abilitata non significa che debba scattare per forza. A questo proposito si veda la Fig. 5: entrambe le transizioni sono abilitate, ma in ingresso non ci sono abbastanza marche per farle scattare tutte e due. In questo caso le transizioni vengono dette *in alternativa*, e soltanto una di esse scatterà. Inoltre, in assenza di temporizzazione (un concetto assente nelle PN senza estensioni), non è determinabile il momento in cui una transizione abilitata allo scatto effettivamente ha luogo. In altre parole, il fatto che una transizione sia abilitata allo scatto non implica che scatti immediatamente.

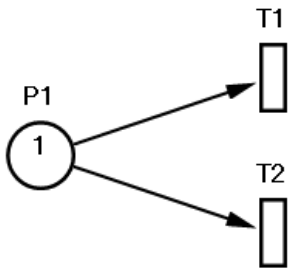

Fig. 5. Transizioni in alternativa

1.3 DEFINIZIONI FORMALI DI PN

Presentiamo ora alcune definizioni formali sul concetto di PN. Come presto vedremo, esistono diversi tipi di PN: la definizione seguente è quella di PN generale [Rei00].

*Una **PN** è una tripla:*

$$\mathbf{PN = (P, T, A)}$$

dove P è detto insieme dei posti, T è detto insieme delle transizioni ed A è detto insieme degli archi. Inoltre devono valere le proprietà:

1. $P \cap T = \varnothing$
2. $P \cup T \neq \varnothing$
3. $A \subseteq (P \times T) \cup (T \times P)$
4. $dom(A) \cup ran(A) = P \cup T$, *dove*
   $dom(A) = \{x \in P \cup T : (x,y) \in A \text{ per qualche } y \in P \cup T\}$
   $ran(A) = \{y \in P \cup T : (x,y) \in A \text{ per qualche } x \in P \cup T\}$

La 1 afferma che l'insieme dei posti **P** e l'insieme delle transizioni **T** sono disgiunti. La 2 impone alla rete di non essere vuota. La 3 afferma che un arco **A** connette soltanto nodi di tipo diverso. La 4, infine, stabilisce che non devono esistere nodi isolati, ovvero senza alcun arco entrante e uscente.

Il tipo di PN più adatto alla codifica e trattazione dell'informazione multimediale incorpora però alcune estensioni: la definizione seguente, tratta da [Des00], meglio si adatta ai nostri scopi.

*Una **PN P/T** è una tupla:*

$$\mathbf{PN = (P, T, A, c, p, m_0)}$$

dove (P, T, A) è una PN generale, con

P : insieme dei posti, non vuoto, finito

T : insieme delle transizioni, non vuoto, finito

$A \subseteq (P \times T) \cup (T \times P)$: *insieme degli archi*

$c : P \rightarrow \{1, 2, 3, ...\}$: *capacità dei posti*

$w : A \rightarrow \{1, 2, 3, ...\}$: *peso degli archi*

$m_0 : P \rightarrow \{0, 1, 2, ...\}$: *marcatura iniziale dei posti*

(deve essere: $\forall\ p \in P : m_0(p) \leq c(p)$)

Un nodo (sia esso un posto o una transizione) può avere più di un nodo del tipo opposto in ingresso o in uscita. Questo fa sì che più archi possano unire un posto ad una transizione o viceversa. Il numero di tali archi dà la *molteplicità* o *peso* di un arco.

1.4 ESTENSIONI

Il modello di rete fin qui descritto è chiamato *Posti/Transizioni*, ed incorpora alcune modifiche rispetto alla definizione di PN generale. Nei successivi paragrafi verranno presentate le estensioni di diretto interesse.

1.4.1 CAPACITÀ

La *capacità* è una proprietà dei posti che definisce il numero massimo di marche ammesse per ogni nodo. Questa caratteristica aggiunge una nuova condizione per l'abilitazione di una transizione: una transizione non è abilitata se la marcatura di almeno un nodo di uscita supera la sua capacità dopo lo scatto della transizione.

L'introduzione della capacità ha come conseguenza anche l'aggiunta di un nuovo caso di non determinismo, denominato *conflitto*, che si presenta quando due o più transizioni sono abilitate e lo scatto di una inibisce lo scatto delle altre. Questa inibizione è data dal fatto che lo scatto di tutte le transizioni porterebbe nei posti di uscita un numero di marche superiore rispetto alla capacità dei suddetti posti. La Fig. 6 mostra un esempio di conflitto.

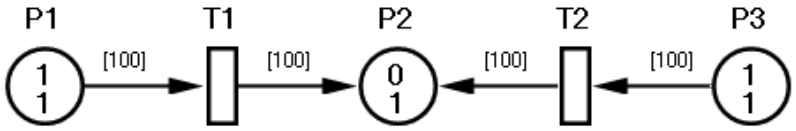

Fig. 6. Esempio di conflitto

1.4.2 RAFFINAMENTI

I *raffinamenti* rappresentano un tipo di morfismo. La teoria riguardante i morfismi è estesa e complessa, ed esula dagli scopi della presente trattazione. In questa sede ci concentreremo dunque unicamente sui raffinamenti.

I raffinamenti sono utili per descrivere modelli complessi di PN tramite PN semplici e strutture gerarchiche. Un raffinamento, che qui viene chiamato *sottorete*, è una PN che dà una descrizione più dettagliata di un nodo dal più alto livello di astrazione. Questo nodo viene chiamato *nodo padre*, mentre la sottorete viene detta *rete figlia*.

L'esempio in Fig. 7 illustra un caso di raffinamento. La parte alta della figura rappresenta l'intera rete, senza raffinamenti. La parte bassa mostra come la rete si trasforma da un punto di vista

grafico introducendo il concetto di raffinamento. Compare ora un posto **P2**, che sottende la rete figlia mostrata a margine.

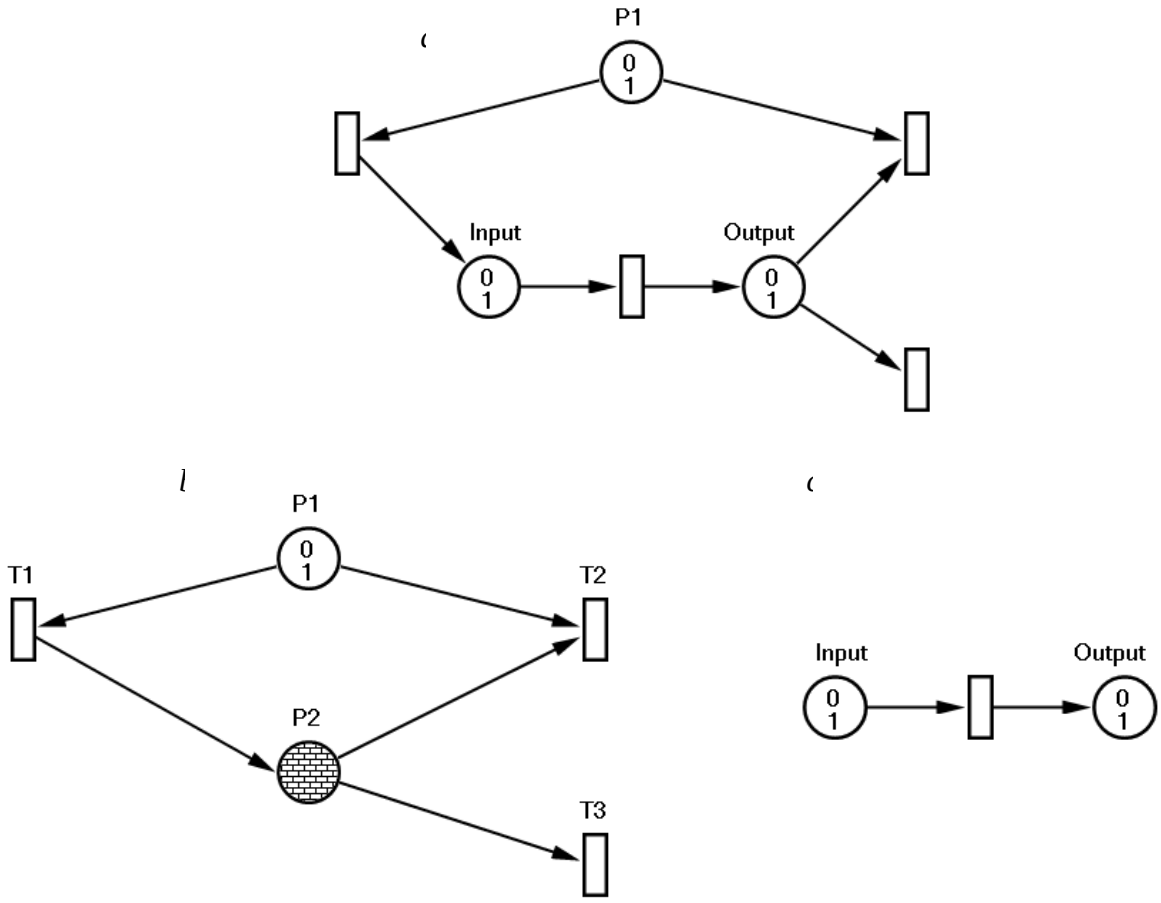

Fig. 7. Concetto di raffinamento

Se si sceglie di definire la sottorete associata ad un posto, allora la rete figlia deve avere due posti speciali: il *posto di input* e il *posto di output*; gli archi di ingresso del posto padre sono gli archi di ingresso della rete figlia; gli archi di uscita del posto padre sono gli archi di uscita della rete figlia. Un discorso analogo vale per le transizioni.

1.4.3 TEMPORIZZAZIONE

Le PN sono particolarmente adatte a descrivere processi concorrenti e controllare la loro eventuale sincronizzazione. Analizzando il processo da un punto di vista temporale, quando una transizione è abilitata allo scatto, la durata di tale scatto è supposta istantanea. Inoltre, come già detto, non è determinabile l'istante in cui lo scatto (abilitato) avrà luogo.

Non ci sono misure di tempo all'interno delle PN: è il comportamento della rete che implicitamente determina la temporizzazione, ed è la struttura della rete a definire la sequenza degli

scatti. Osserviamo inoltre che la sequenza degli scatti può cambiare in diverse esecuzioni della rete: si possono avere più transizioni abilitate nello stesso istante e non si può sapere quale di queste scatterà per prima.

Al fine di descrivere la durata di eventi temporizzati all'interno delle PN, è possibile associare degli intervalli o durate temporali alle transizioni o ai posti. Il discorso sarà approfondito maggiormente nel prossimo capitolo, dedicato alle applicazioni musicali.

1.5 STRUTTURE SIGNIFICATIVE

Esistono alcune strutture elementari utili nella progettazione delle PN e consolidate in letteratura. Esse sono: la *sequenza*, l'*alimentazione congiunta* **(**o *parallelismo* o *split*), l'*alimentazione alternativa* **(**o *non determinismo*), la *congiunzione* (o *asincronismo* o *joint*) e la *fusione* (o *sincronismo*). Si veda a questo proposito la Fig. 8.

Tutte le strutture elementari considerate possono essere collegate per formare reti più complesse.

Un caso particolarmente interessante è quello dell'*alimentazione alternativa*, in quanto introduce il *non determinismo*. In presenza di questa struttura, è possibile comunque configurare la sequenza degli scatti utilizzando una funzione probabilistica: si associa ad ogni arco un peso probabilistico; tale valore relativo rispetto alla somma totale dei pesi probabilistici degli archi in conflitto indica la probabilità di scelta. Chiariamo con un esempio questo concetto. Si consideri una PN con i seguenti archi:

- Arco 1 (A1): Peso probabilistico 5
- Arco 2 (A2): Peso probabilistico 10
- Arco 3 (A3): Peso probabilistico 300

Nel caso in cui l'applicazione debba effettuare un scelta non deterministica tra A1, A2 e A3, l'arco A1 avrà 5 probabilità su 315 di essere scelto (1,6%), A2 ne avrà 10 su 315 (3,2%), e A3 ne avrà 300 su 315 (95,2%).

Se, in un passaggio successivo dell'esecuzione, gli archi che indicano le transizioni in conflitto sono soltanto A1 e A2, le probabilità cambiano dinamicamente e diventano: 5 su 15 (33,3%) per A1 e 10 su 15 (66,7%) per A2.

Un caso particolare si verifica qualora il peso probabilistico di un arco è pari a 0; in tale situazione, la transizione ad esso associata scatterà se e soltanto se non esiste nessun altro arco abilitante con probabilità maggiore di 0.

Nelle nostre convenzioni grafiche, il peso probabilistico di un arco è indicato da un numero posto in corrispondenza dell'arco stesso tra parentesi quadre. Si veda a questo proposito la Fig. 6 sulle transizioni in conflitto.

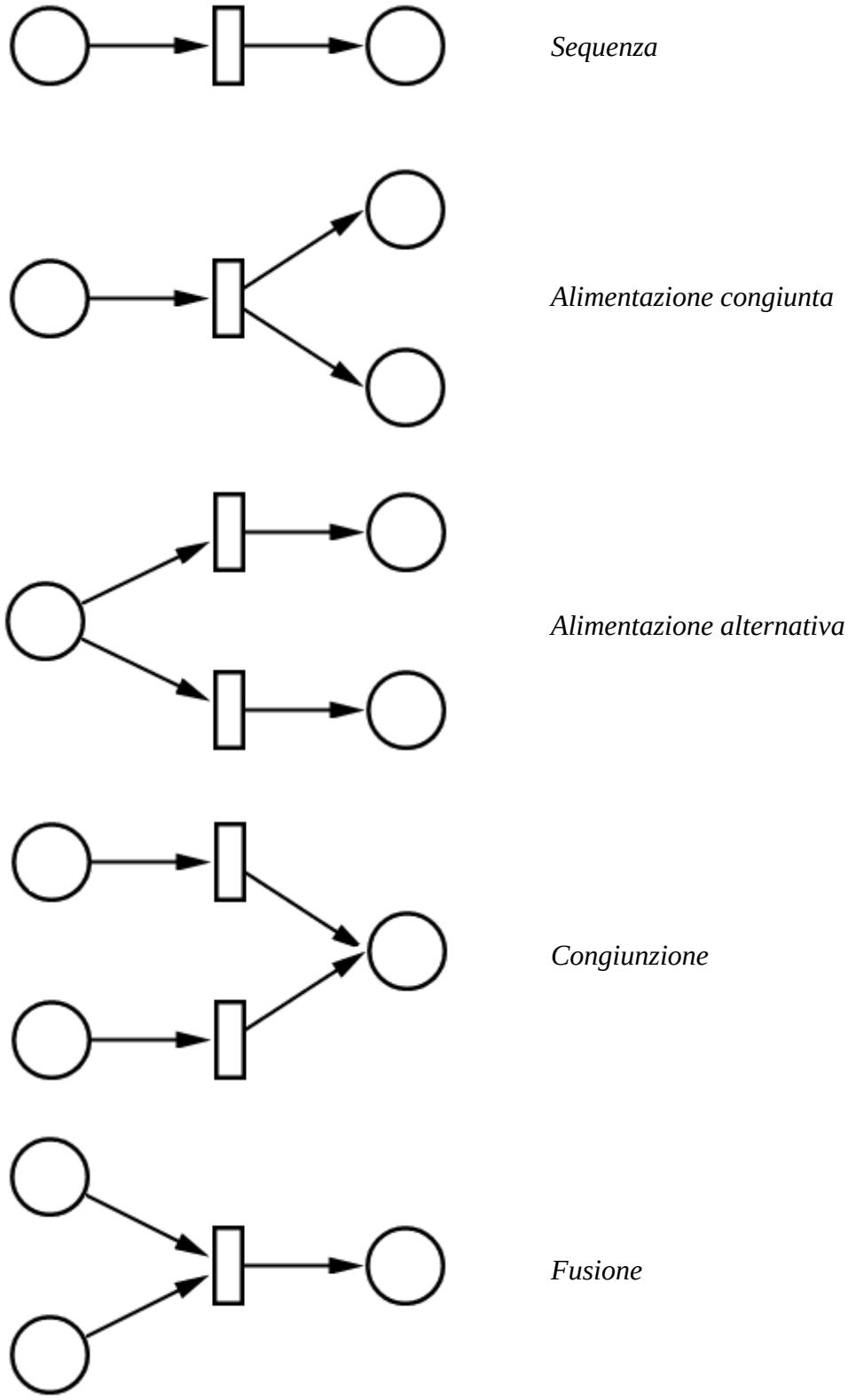

Fig. 8. Strutture elementari delle PN

Una struttura decisamente importante per rappresentare dipendenze logico-casuali è la *wait* (attesa), in cui si utilizza un oggetto per controllare il verificarsi di un oggetto musicale.

Leggermente più complicata è la struttura *loop* (ciclo ripetitivo), usata per ripetere un determinato numero di volte una sottostruttura. In Fig. 9 ne sono mostrate due possibili implementazioni; nella seconda, viene utilizzato un oggetto di controllo contatore che ha la funzione di contenere un numero di marche pari al numero di cicli da eseguire.

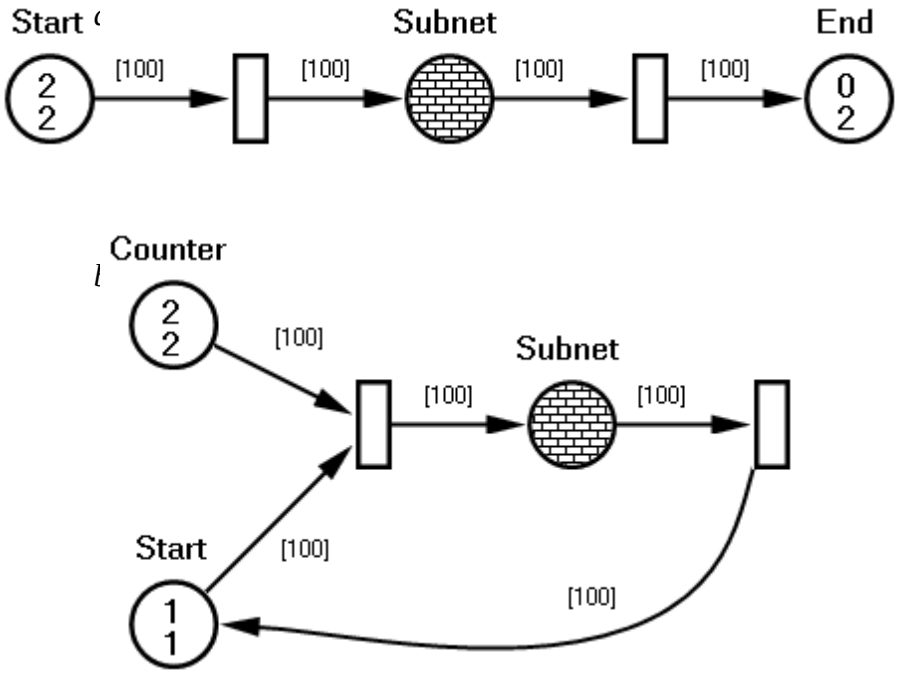

Fig. 9. Due implementazioni della struttura *loop*

1.6 UN ESEMPIO DI PN: SELETTORE DI POSTI

Per illustrare praticamente il funzionamento di una PN forniremo ora un esempio di rete in cui verrà posta particolare attenzione all'utilizzo del peso probabilistico nelle situazioni di alternativa tra transizioni, realizzando una PN che, ponendo nel posto Selettore un numero di marche *n* fra 1 e 4, seleziona tra 4 posti (MX1, MX2, MX2 e MX4) quello numero *n*.

La PN è visibile in Fig. 10.

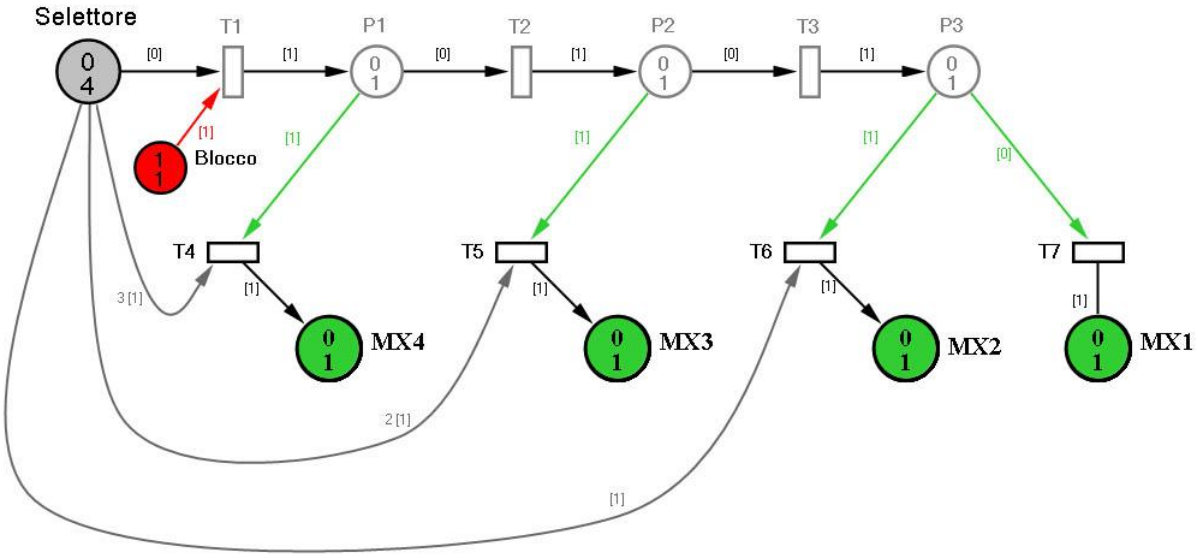

Fig. 10. PN Selettore di posti

Prima di illustrare il funzionamento della rete, ricordiamo che un arco con peso probabilistico 0 farà scattare la transizione associata soltanto quando non è in alternativa/conflitto con altri archi con peso maggiore di 0. L'arco quindi potrà far scattare la transizione soltanto: a) quando è l'unico arco della rete che possa far scattare una transizione, oppure b) quando sono in alternativa/conflitto soltanto transizioni con archi di peso probabilistico 0, nel qual caso la transizione che scatta sarà scelta casualmente[1].

Vediamo il funzionamento della rete con il numero variabile di marche poste inizialmente nel posto Selettore:

- 4 marche: all'inizio dell'esecuzione l'unica transizione che può scattare è **T1**, visto che le altre transizioni hanno almeno un posto in ingresso senza marche. La transizione scatta anche se l'arco da Selettore a **T1** ha peso probabilistico 0, visto che non ci sono altri archi in alternativa/conflitto. Ora P1 ha una marca e 3 marche sono rimaste in Selettore. Questo fa sì

[1] In verità nell'algoritmo di scelta viene selezionato in questo caso il primo arco tra quelli in alternativa/conflitto e viene eseguita la transizione corrispondente.

che esista un alternativa tra lo scatto di **T2** e **T4**. Considerando però che l'arco che collega P1 a **T2** ha peso 0, verrà scelta **T4** che, scattando, attiverà il posto MX4.

- 3 marche: dopo lo scatto iniziale di **T1 (**come visto nel caso precedente), **T4** non può scattare perché l'arco che proviene da Selettore ha peso in marche 3 mentre il posto in ingresso ha soltanto 2 marche. L'unica transizione abilitata è **T2** che, scattando, pone una marca in P2. Ora si verifica una situazione di alternativa analoga al punto precedente, con **T5** che, avendo peso probabilistico maggiore di 0, scatterà attivando MX3.
- 2 marche: il funzionamento iniziale della rete è analogo ai punti precedenti: riprendiamo quindi dallo stato illustrato in Fig. 11. A questo punto entrano in gioco il posto Blocco: se esso non fosse presente in questo stato scatterebbero le transizioni **T3** e **T1**, che hanno entrambe in ingresso archi con peso probabilistico 0, ma che non sono in alternativa/conflitto con altre. Questo comporterebbe la mancanza di un'altra marca in Selettore e l'inibizione quindi di **T6**. Inserendo invece il posto Blocco, **T1** può scattare solo all'inizio dell'esecuzione, e, partendo dalla situazione in Fig. 11, scatta **T3**, poi **T6**, attivando il posto MX2.
- 1 marca: nella situazione finale del punto precedente P3 conteneva una marca. In questo caso **T6** non può scattare perché Selettore è vuoto: viene attivato il posto MX1.

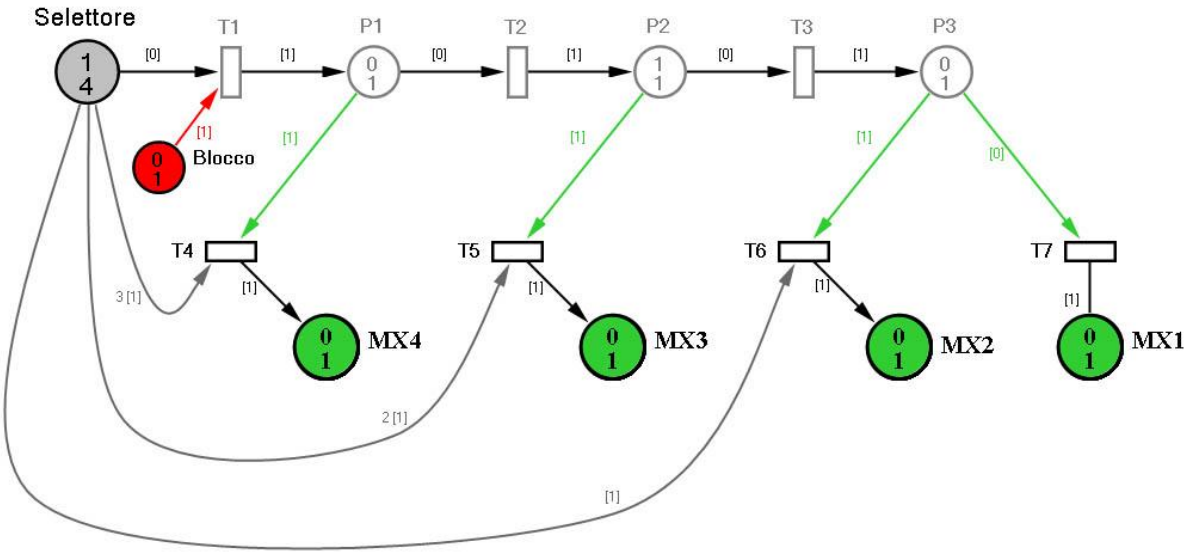

Fig. 11. PN Selettore di posti durante l'esecuzione

Naturalmente la rete descritta può essere estesa ad un selettore fra un numero di posti qualsiasi, soltanto aggiungendo alla struttura altri elementi in modo simile a quelli esistenti.

CAPITOLO 2

RETI DI PETRI E MUSICA

2.1 MOTIVAZIONI

Come evidenziato in [CHZ86], uno degli scopi principali della ricerca nel campo dell'Informatica musicale è la definizione di linguaggi adatti alla descrizione di entità musicali. Tra le caratteristiche che tali linguaggi devono necessariamente avere, ricordiamo:

- affinità con il modo di ragionare del compositore;
- supporto di diversi livelli di rappresentazione dell'informazione musicale;
- meccanismi di morfismo tra i vari livelli di rappresentazione;
- strutture di elaborazione concorrente;
- operatori per elaborare e trasformare le entità musicali;
- alta flessibilità.

Come vedremo nei prossimi paragrafi, presso il Laboratorio di Informatica Musicale è stato indagato il formalismo grafico delle Reti di Petri, che presenta numerose caratteristiche interessanti per la descrizione e l'elaborazione della musica. Secondo [HS92], le PN rappresentano un modo per descrivere, elaborare e sintetizzare le strutture musicali addirittura più flessibile della notazione tradizionale su rigo. Di certo, esse si prestano a descrivere aspetti della musica che ben difficilmente trovano un'adeguata rappresentazione nella scrittura tradizionale, e questo aspetto sta diventando particolarmente rilevante nella musica contemporanea (si pensi alla musica elettronica, a quella concreta, alla computer-music e via dicendo…)

Presso il Laboratorio di Informatica Musicale, le PN vengono utilizzate fin dai prima anni '80 come strumento di base per la descrizione e l'elaborazione della musica. Tale strumento formale è apparso adeguato [HS92], dato che:

- richiede l'uso di pochi simboli;
- presenta una forma grafica di notazione;
- consente descrizioni gerarchizzate;
- permette la descrizione e l'elaborazione di oggetti musicali;
- consente di caratterizzare il tempo;
- supporta modelli deterministici e non deterministici;
- consente macro-definizioni per strutture comuni.

2.2 GLI OGGETTI MUSICALI

Le PN possono essere validamente usate come linguaggio per descrivere l'interazione concorrente tra oggetti. In questo caso si tratterà di oggetti musicali, locuzione che richiede una definizione.

Per *oggetto musicale* intendiamo qualunque cosa abbia un significato musicale e possa essere pensata come un'entità unitaria, sia essa semplice o complessa, astratta o concreta. Gli oggetti musicali sono caratterizzati da un nome e da relazioni con altri oggetti musicali.

Un'elaborazione musicale può dunque essere descritta in termini di evoluzione di uno o più oggetti musicali. L'identificazione percettiva di tali oggetti non è né semplice né univoca: due ascoltatori diversi, anche a parità di competenza, possono avere una percezione diversa della stessa esecuzione. Dato un pezzo musicale, quanti e quali sono gli oggetti musicali? Ancora una volta, la risposta non può essere univoca. Da un lato, un processo musicale può essere descritto a basso livello come sequenza di campioni sonori; all'estremo opposto, troviamo descrizioni con alto grado di astrazione, in cui si identificano strutture di organizzazione dei suoni o dei simboli in partitura. La determinazione delle relazioni tra entità elementari, svolta con il fine di dar loro unità ed identità, porta all'identificazione di oggetti musicali. Dunque la musica può essere definita come evoluzione di oggetti musicali che interagiscono tra loro. Nella reciproca interazione, gli oggetti possono comparire o sparire, modificarsi o ripetersi,… La definizione degli oggetti e delle trasformazioni deriva da un processo di astrazione a sua volta non univoco, dipendendo dal punto di vista in cui ci si pone nei confronti del processo musicale. Ad esempio, in [HS92] vengono considerati oggetti musicali di tipo nota, meta-evento e messaggio MIDI.

A scopo esemplificativo, consideriamo due casi molto diversi di identificazione nella partitura di oggetti musicali. Un oggetto musicale può essere una ricorrente formula melodica di accompagnamento (ad esempio, il *basso albertino*); le sue trasformazioni possono essere di natura ritmica e melodica, e possono trarre giustificazione (o a loro volta influenzare) le altre voci della composizione. Sempre considerando la partitura ma da un punto di vista radicalmente diverso, un oggetto musicale può anche essere una macro-sezione, come nel caso del *refrain* di un *rondò*; e in tal caso le trasformazioni sono i processi che portano alla riproposizione, letterale o variata, del ritornello nelle sue diverse occorrenze.

Per comprendere la natura dell'interazione tra oggetti, possiamo utilizzare i concetti di *gerarchia*, *concorrenza* e *causalità* definiti in [DH82] e solo accennati in questa trattazione.

Evidenziamo infine che l'introduzione del concetto di oggetto musicale enfatizza la natura concorrente e distribuita di un processo musicale, che può essere visto ora come il risultato della cooperazione di diversi attori che comunicano tramite messaggi.

2.3 OGGETTI MUSICALI E RETI DI PETRI

Una volta definito cosa sia un oggetto musicale, adottiamo le PN come linguaggio per descrivere il flusso temporale di uno o più oggetti musicali e le reciproche interazioni

Gli oggetti musicali in generale saranno situati nei posti di una PN, mentre le transizioni della PN rappresenteranno regole di trasformazione, che nel nostro caso sono implementate da operatori musicali.

La stessa informazione musicale può essere descritta tramite PN a diversi livelli di astrazione per mezzo di approcci di modellazione alternativi. Secondo [HS92], modellizzare la musica con PN rende naturale la descrizione del suo livello strutturale come un ambiente multilivello al cui interno gli oggetti musicali possono fluire in modo concorrente e interattivo.

Gli oggetti musicali possono essere elaborati e modificati dalla PN; gli stessi parametri della PN permettono di creare istanze di tali oggetti che si modificano in accordo al comportamento della rete in esecuzione.

Un modello di PN con una particolare marcatura iniziale può rappresentare un'intera famiglia di partiture; una determinata esecuzione del modello sintetizza una partitura specifica. Cambiando la marcatura iniziale, varia anche la famiglia di partiture che possono essere prodotte dall'esecuzione del modello. La generazione di un'unica partitura a partire da una certa marcatura iniziale è in realtà un caso particolare che si verifica quando il modello è totalmente deterministico.

Vediamo ora qualche esempio concreto di PN che coinvolga oggetti musicali e trasformazioni. Ad esempio, le cinque strutture elementari mostrate in Fig. 12 trovano un'immediata interpretazione musicale:

- la *sequenza* serve a descrivere un flusso sequenziale di oggetti;
- l'*alimentazione congiunta* rappresenta la generazione di due o più oggetti partendo da uno solo;
- l'*alimentazione alternativa* può descrivere casi musicali che implicano scelte non deterministiche;
- la *congiunzione* rappresenta un collegamento o confronto (in senso logico) tra oggetti differenti;
- la *fusione*, infine, descrive l'unione di due oggetti a formarne uno solo.

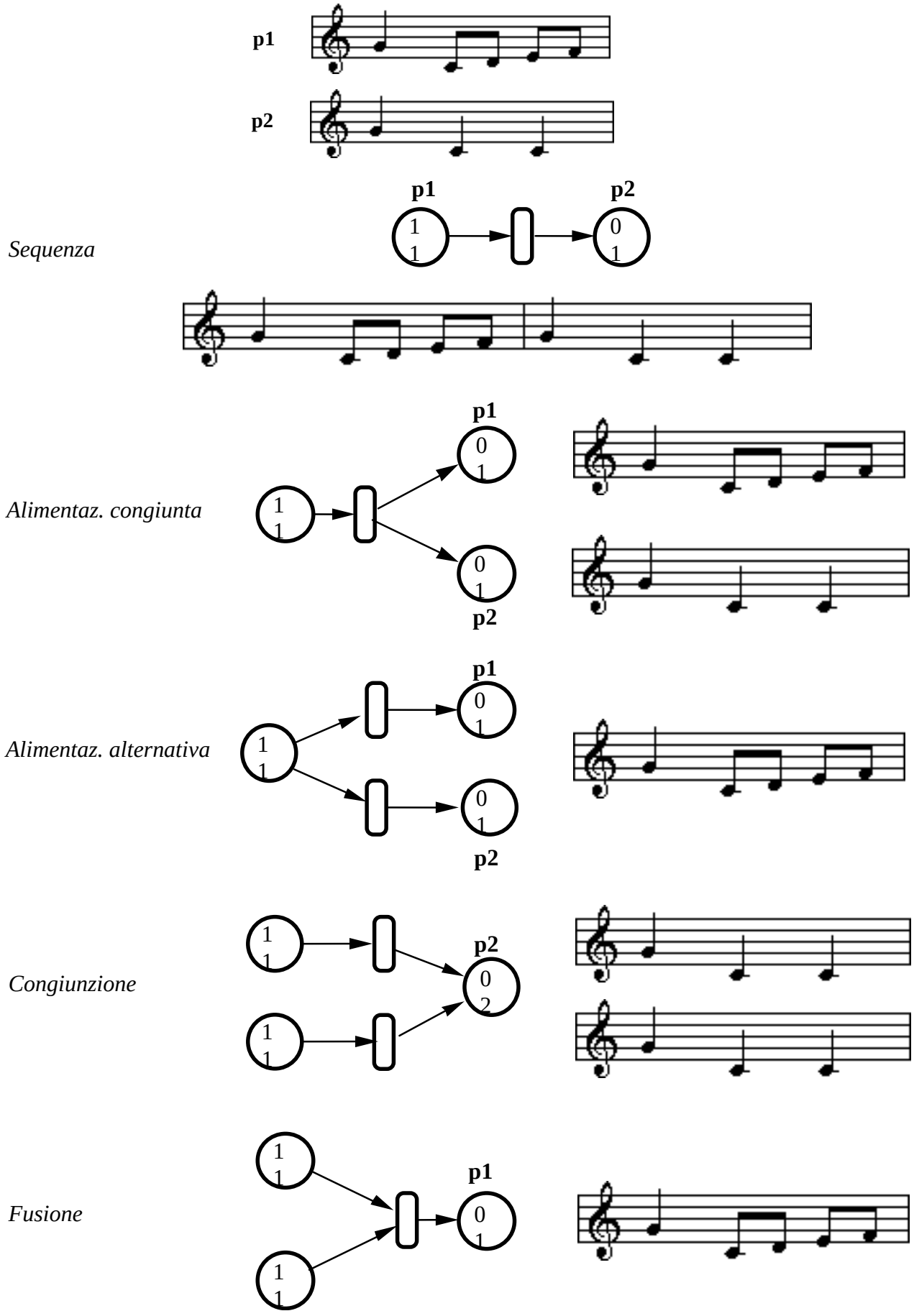

Fig. 12. Strutture elementari ed esempi di implementazione su frammenti di partitura

La Fig. 12 mostra le potenzialità delle PN nella strutturazione di un brano musicale. Ad esempio, la struttura della sequenza non fa altro che accodare due oggetti musicali (in questo caso, si trattava di due frammenti di partitura). Ma alle transizioni è possibile associare delle vere e proprie trasformazioni del materiale musicale. Si tratta in questo caso di processi che in qualche modo alterano gli oggetti musicali. Per citare le tipologie di trasformazione più comuni, ricordiamo: la trasposizione, l'aumentazione/diminuzione, la retrogradazione, l'inversione speculare… Ciascuna di esse manipola l'oggetto musicale che si trova nel posto di ingresso e lo propone trasformato al posto di uscita.

Parlando di oggetti musicali e Reti di Petri, sovviene un'interessante considerazione. Sulla base delle definizioni proposte finora, un qualsiasi processo musicale può essere descritto in termini di un *concetto metaforico* (per noi il concetto di oggetto musicale) e di un *linguaggio formale* (nel nostro caso si tratta del formalismo grafico delle Reti di Petri). Le interazioni tra oggetti musicali vengono controllate dalle marche presenti nella rete.

Lo scopo della coppia [metafora, linguaggio] è la descrizione formale di un processo musicale, che in altre parole è un'analisi volta a migliorare la comprensione del processo. E' opportuno sottolineare che i due elementi evidenziati, ossia la metafora e il linguaggio, mantengono l'indipendenza dal punto di vista logico.

2.4 IL TEMPO

Tutti gli oggetti musicali, a qualsiasi livello di astrazione, sono temporizzati. Il tempo rappresenta una delle dimensioni irrinunciabili della musica, e trova esplicazione nel concetto di ritmo. Ricordiamo che il ritmo non è associato unicamente alla durata (valore) delle note, ma anche alla disposizione degli accenti, alla successione delle armonie, alla scansione temporale in frasi, episodi, sezioni, cadenze e via dicendo… Possiamo vedere il tempo come un oggetto indipendente con cui ogni altro oggetto deve relazionarsi.

Gli oggetti musicali agiscono nel tempo in modo concorrente, comunicando e sincronizzandosi in particolari istanti sull'asse dei tempi.

Quando passiamo alla descrizione formale del linguaggio musicale, il modello adottato deve necessariamente assicurare due condizioni:

- la *consistenza temporale*, ossia l'esistenza di un unico asse temporale;
- la *risoluzione temporale*, ossia la distanza minima tra due punti distinti dell'asse temporale.

Come detto in precedenza, la struttura di base delle PN può essere adeguatamente estesa per modellizzare i processi temporizzati. L'esecuzione della rete diventa un processo non solo distribuito ma anche temporizzato.

Nel caso specifico delle PN applicate alla musica, molto spesso gli oggetti musicali sono proprio frammenti di partitura. L'estensione delle PN da noi considerata abilita la transizione in uscita non appena termina l'esecuzione del frammento musicale.

2.5 LE PN COME STRUMENTO ANALITICO-DESCRITTIVO

In molti lavori del L.I.M. (Laboratorio di Informatica Musicale) le Reti di Petri sono state utilizzate per descrivere ed analizzare composizioni appartenenti alla letteratura tradizionale. In particolare, ricordiamo l'applicazione delle PN a:

- J.S. Bach, *Canone perpetuo dall'Offerta musicale*, [Hau84]
- J.S. Bach, *Fuga XXIII dal I libro del Clavicembalo ben temperato*, [CHZ86]
- J.S. Bach, *Invenzione a 2 voci n° 1*, [HS92]
- M. Ravel, *Bolero*, [HR93]
- I. Stravinskij, *La sagra della primavera*, [DH93] [DH96]

Quando le PN vengono utilizzate come strumento di analisi, la loro applicazione porta a una migliore comprensione del pensiero compositivo dell'autore. Lo sforzo di identificare oggetti musicali e reciproche interazioni consente una visualizzazione incisiva ed originale del percorso musicale, che si concretizza nel formalismo grafico delle PN stesse.

La descrizione della partitura può avvenire a diversi gradi di astrazione. Prendendo come esempio il Bolero [HR93], la Fig. 13 evidenzia la struttura di controllo ad alto livello dell'intero pezzo. Quest'ultimo si compone di quattro strutture cicliche in crescendo e di un finale. Il primo scatto nella rete dà inizio alla prima struttura ciclica, e in seguito ogni ciclo produce la sequenza di base associata al posto 2B1+2B2 (che poi altro non è che la ripetizione alternata di due strutture in modo maggiore e due strutture in modo minore); dopo tre cicli si arriva alla quarta iterazione e la struttura ciclica viene interrotta dalla mancanza di *token*, mentre è abilitata allo scatto la transizione che conduce al finale.

Precisiamo che in Fig. 13 sono stati adottati dei formalismi grafici particolari cui finora non si è fatto cenno, quali l'arco inibitore posto in uscita a CNT.

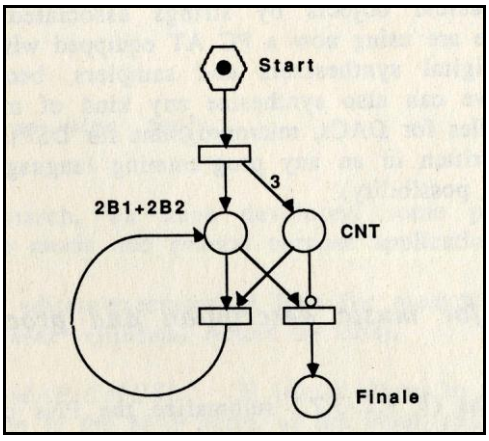

Fig. 13. La macrostruttura del Bolero di Ravel

2.6 LE PN COME STRUMENTO COMPOSITIVO

La descrizione di oggetti musicali, sia estratti di partiture sia oggetti speciali definiti per la sintesi, può essere in realtà il primo passo per la trasformazione e rielaborazione delle strutture musicali identificate.

In pratica, esistono numerosissime possibili applicazioni delle PN al pensiero compositivo. Della prima abbiamo accennato: si tratta di rielaborare le strutture identificate in qualcosa di esistente per dar vita a una nuova composizione. Ciò può avvenire ad esempio mantenendo inalterata la macrostruttura ma cambiando il materiale sonoro all'interno dei posti (applicare il processo di Fig. 13 a frammenti di un minuetto di Mozart); oppure cambiando la marcatura iniziale, il che produce una diversa istanza di esecuzione della stessa rete; o ancora cambiando il significato musicale delle transizioni (trasposizioni, retrogradazioni, diminuzioni, ecc.). E queste sono solo alcune delle possibilità offerte dall'applicazione delle PN al pensiero musicale.

Riteniamo interessante evidenziare alcuni brani concepiti esplicitamente per PN, quali le "Musiche per poche parti" di G. Haus. In questo caso, l'idea compositiva non trae spunto da partiture pre-esistenti, bensì da una struttura e del materiale sonoro originale, che viene proposto e rielaborato sulla base della struttura stessa.

Sottolineiamo nuovamente gli strumenti principali che le PN offrono al compositore: iterazione, concorrenza, ordinamento, sincronizzazione, gerarchia. Tutto questo può essere applicato a materiale sonoro (partiture tradizionali, suoni di sintesi, controlli MIDI,…) per realizzare il proprio pensiero musicale.

2.7 STORIA DELLE PN AL L.I.M.

Fin dai primi anni ’80 presso il L.I.M. ci si occupa dell’applicazione delle PN all’ambito musicale con fini di analisi e di rielaborazione.

I primi passi in questo senso si devono alle pubblicazioni di G. Degli Antoni e G. Haus, che hanno investigato la possibilità di descrivere processi musicali mediante differenti approcci formali, impostando una base concettuale per la descrizione mediante Reti di Petri della causalità nei processi musicali [DH82][DH85].

In collaborazione con il musicologo A. Rodriguez, G. Haus ha applicato tali concetti alla descrizione di processi musicali temporizzati mediante Reti di Petri e loro specifiche estensioni [HR88]. A tale scopo si sono usati come "banco di prova" testi musicali complessi, come mostrato in [HR93]. In quest'ultimo lavoro è stato definito ed estensivamente applicato un particolare tipo di Reti di Petri gerarchiche, con capacità, molteplicità e macrodefinizioni attualizzabili mediante lista di parametri formali.

G. Haus e A. De Matteis hanno poi messo in luce i limiti di questo approccio, basando le argomentazioni su opportuni controesempi [DH93] [DH96].

L'applicazione musicale delle Reti di Petri ha rivestito un ruolo centrale nelle ricerche presso il L.I.M. Questa attività è stata ed è tuttora oggetto di continuo confronto con altri ricercatori stranieri che seguono approcci più o meno affini; in particolare, si è rivelata significativa la collaborazione con il Computer Audio Research Laboratory dell'Università della California in San Diego per l'integrazione del loro sistema software CMusic con il nostro esecutore di Reti di Petri musicali e il coordinamento con S. Pope (dal 1990 *editor* del Computer Music Journal e responsabile del laboratorio di Computer Music della Rank Xerox a Palo Alto). Pope ha anch'egli sperimentato le Reti di Petri per applicazioni musicali in ambiente Smalltalk. I diversi approcci si sono reciprocamente influenzati e sono divenuti complessivamente un punto di riferimento per quanto concerne la ricerca sugli strumenti formali per la descrizione musicale.

In questo ambito di ricerca sono stati realizzati alcuni prototipi software, in diverse versioni corrispondenti a diversi linguaggi di *output*; la caratteristica comune è l’accettare come linguaggio di ingresso classi di partiture musicali descritte mediante Reti di Petri e produrre, mediante l'esecuzione dei modelli di reti, partiture nei linguaggi operativi tipici di alcuni sistemi per la sintesi digitale del suono. I più rilevanti prototipi realizzati sono:

- MAP/CMusic, che genera partiture a livello operativo-sonoro in linguaggio CMusic [CHZ86]
- MAP/MCL, che produce partiture a livello simbolico in linguaggio MCL per la workstation musicale CMI Fairlight [CHIZ86]

- MAP/Mac che genera partiture a livello esecutivo MIDI
- SCORESYNTH che crea partiture a livello esecutivo in Standard MIDI File Format 1.0 [HS91][HS92][HS94]. Il sistema SCORESYNTH unisce un'algebra specifica per la elaborazione di oggetti musicali (nozione definita in [DH82] e [DH85]) alla descrizione della gerarchia, della concorrenza e della sincronizzazione dei processi musicali mediante Reti di Petri.

Un discorso a parte riguarda il sistema PETREX [Hau88a][Hau88b] che, pur non essendo specializzato per la sintesi di testi musicali ma piuttosto per la simulazione di sistemi reali complessi e per l'integrazione tra più sistemi informativi, è comunque in grado di generare testi in un qualsivoglia linguaggio formale sulla base delle regole di produzione associate agli scatti delle transizioni delle Reti di Petri. PETREX è un editor/esecutore di modelli formali basati su Reti di Petri gerarchiche, automodificantisi e colorate. Tale software permette di sperimentare modelli che simulano il comportamento di sistemi reali complessi, integrare diversi sistemi informativi attraverso appositi canali logici, sintetizzare testi coerenti con le regole di produzione che possono essere associate alle transizioni delle reti.

G. Haus, T. Belletti e G. Galizia hanno progettato un sistema di authoring multimediale, MediaSynth, basato sull'approccio dei modelli gerarchici di Reti di Petri modificate, un modello precedentemente studiato e applicato in campo musicale [BGH95].

In particolare i software ScoreSynth e MediaSynth sono stati recentemente re-implementati per l’ambiente Microsoft Windows rispettivamente a opera di A. Baratè [Bar04] e S. Spagliardi [Spa04]. Tali applicazioni rappresentano per il L.I.M. lo stato dell’arte sulle Reti di Petri applicate al campo della musica e della multimedialità.

2.8 IL FUNZIONAMENTO DI SCORESYNTH

La nuova versione di ScoreSynth è passata dall'utilizzo del formato MIDI per la rappresentazione degli oggetti musicali, al nuovo formato MX, basato su tecnologia XML, sviluppato all'interno del L.I.M. per consentire una rappresentazione multilivello dell'informazione musicale.

Vediamo ora brevemente quali sono le funzionalità del software, partendo dall'interfaccia:

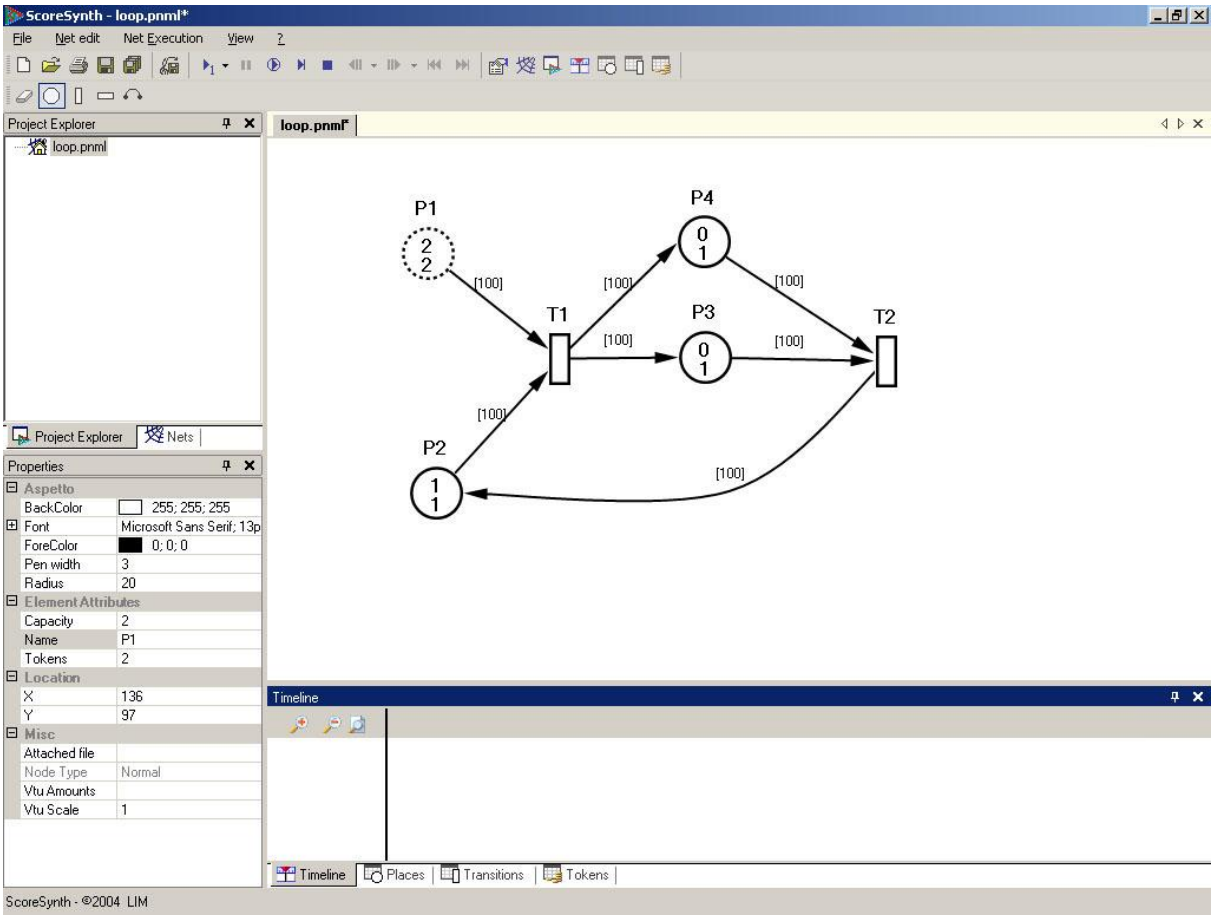

Fig. 14. Interfaccia di ScoreSynth

La zona centrale della finestra d'interfaccia offre un'area per il disegno delle PN, offrendo la possibilità di lavorare con più finestre aperte (standard MDI - Multiple Document Interface); in questo modo è per esempio possibile lavorare con una PN costituita anche da sottonodi, e avere contemporaneamente aperte tutte le sottoreti.

L'utilizzo del tasto destro all'interno dell'area di disegno permette di visualizzare menu contestuali che facilitano la selezione dei tools disponibili e di alcune opzioni basilari della rete.

Nella parte sinistra ed in basso sono invece presenti una serie di finestre mobili, che possono essere ancorate ai bordi della finestra principale, spostate, chiuse o venire chiuse e ripristinate automaticamente al passaggio del mouse.

Project Explorer. ScoreSynth lavora in termini di progetto: ogni progetto è costituito da una serie di PN, visualizzate in questa finestra; occorre definire la rete principale da eseguire.

Nets. In questa finestra viene presentata una struttura ad albero che rappresenta la gerarchia di reti e sottoreti contenute nel progetto.

Properties. Visualizza le proprietà dell'oggetto selezionato o della PN in caso non sia selezionato nulla. In questa finestra è possibile modificare le proprietà grafiche dei nodi, definire in una sottorete i nodi di input/output, associare oggetti musicali ai nodi, ecc.

Places. Riepilogo dei posti con indicazione del nome, marche, capacità, PN di appartenenza, file associato. E' possibile scegliere se mostrare solo i posti della rete visualizzata correntemente oppure di tutte le reti del progetto. Durante l'esecuzione temporizzata oppure passo-passo di una PN è possibile, con l'ausilio di questa finestra e delle successive, mantenere monitorato lo stato della rete per facilitare operazioni di debug.

Transitions. Riepilogo delle transizioni con indicazione del nome, della rete di appartenenza e del file associato. Come per i posti è possibile scegliere il tipo di visualizzazione.

Tokens. Riepilogo delle marche presenti nelle reti.

Timeline. Finestra che mostra le tracce che costituiscono l'esecuzione della PN principale. Ad ogni esecuzione dell'oggetto associato ad un posto della rete viene inserito un rettangolo che rappresenta, con la sua dimensione orizzontale, la lunghezza temporale dell'oggetto stesso. In questo modo si può avere la chiara visualizzazione delle sovrapposizioni e giustapposizioni tra gli oggetti musicali eseguiti.

Nella parte superiore dell'interfaccia trova posto la barra dei menu e quella degli strumenti, che duplica le voci corrispondenti alle operazioni più usate.

Una PN creata in ScoreSynth può essere eseguita secondo diverse modalità:

- **Step-by-Step.** Esecuzione passo-passo; ad ogni passo nella rete visualizzata vengono aggiornati i token dei posti e i contenuti delle finestre di riepilogo.
- **Timed Play.** Esecuzione temporizzata: è possibile scegliere il tempo che intercorre fra i passi dell'esecuzione, consentendo una visione rallentata dell'evoluzione della rete.
- **Play to End.** Esecuzione completa, senza pause tra un passo ed il successivo.

E' poi possibile durante l'esecuzione "navigare" all'interno dei passi già effettuati.

Per un'analisi approfondita dell'interfaccia e delle sue funzionalità rimandiamo al manuale utente di ScoreSynth.

2.9 UN ESEMPIO APPLICATIVO

Vediamo ora un esempio di utilizzo di ScoreSynth per la creazione di un canone: verrà illustrato il procedimento di creazione di 3 versioni di PN ed il loro debug, per evidenziare gli strumenti che l'applicazione mette a disposizione di un musicista.

In Fig. vediamo un canone a 4 voci, in cui ogni voce ripete 2 volte il tema (diviso in 4 parti) prima di terminare. Ogni voce entra al termine dell'esposizione della prima parte della voce precedente.

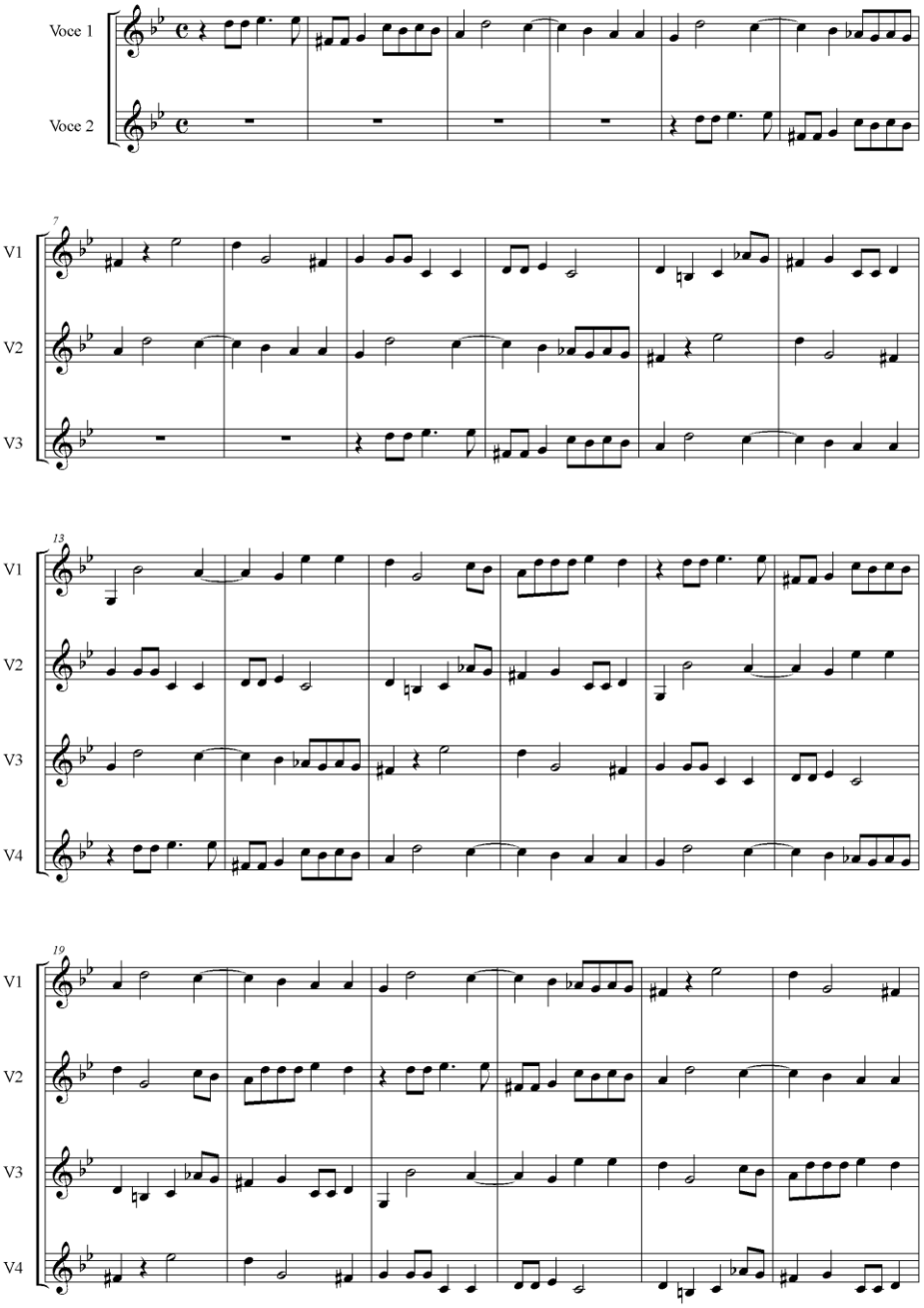

Fig. 15 (continua). Canone di esempio

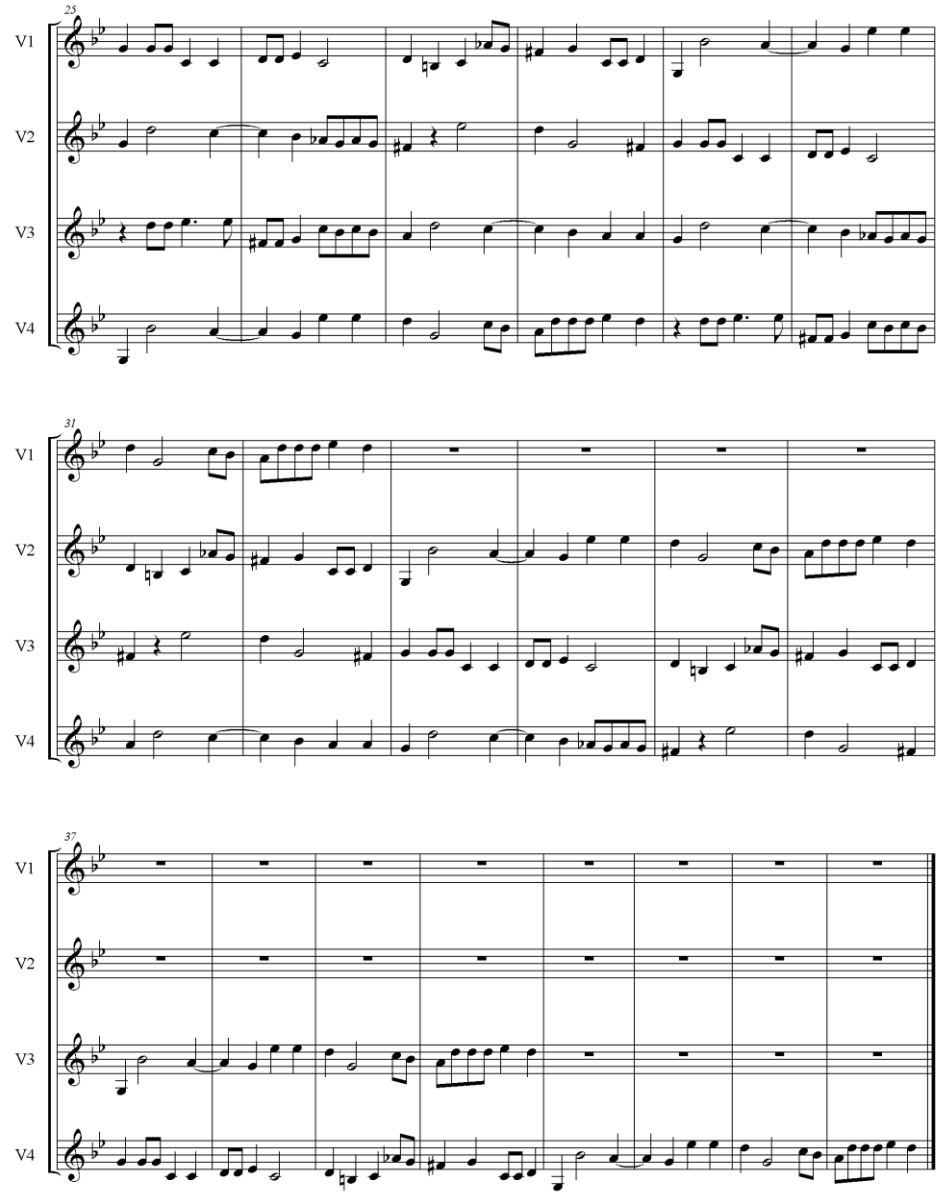

Fig. 15. Canone di esempio

Per creare una PN che rappresenti il canone supponiamo in prima istanza di aver codificato in MX sia l'intero tema ripetuto 2 volte che una pausa di 4 battute, intercorrente tra l'inizio di una voce e quella successiva. In quest'ottica, quando viene eseguito il modello PN, occorrerà che un token abiliti l'esecuzione del tema e della pausa relativa alla voce successiva; questo è implementato dalla rete in Fig. 16, dove il posto Tema ha capacità 4 per contenere i token relativi alle 4 voci che interverranno. Inserendo poi una transizione tra i posti Pausa e Tema, verrà avviata la successiva esecuzione del tema.

A questo punto occorre inserire un meccanismo che consenta di ripetere per ognuna della tre voci successive alla prima il tragitto Pausa->Tema. Questo viene implementato in Fig. 17: allo

scatto di T1 viene avviato il Tema (della prima voce) e vengono passati 3 token al posto Contatore, che, attraverso T2, fanno partire a mano a mano le esposizioni successive del tema.

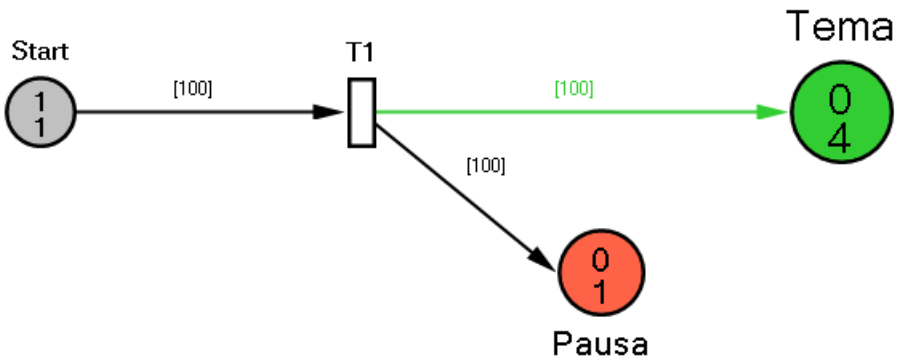

Fig. 16. Esecuzione base Tema-Pausa

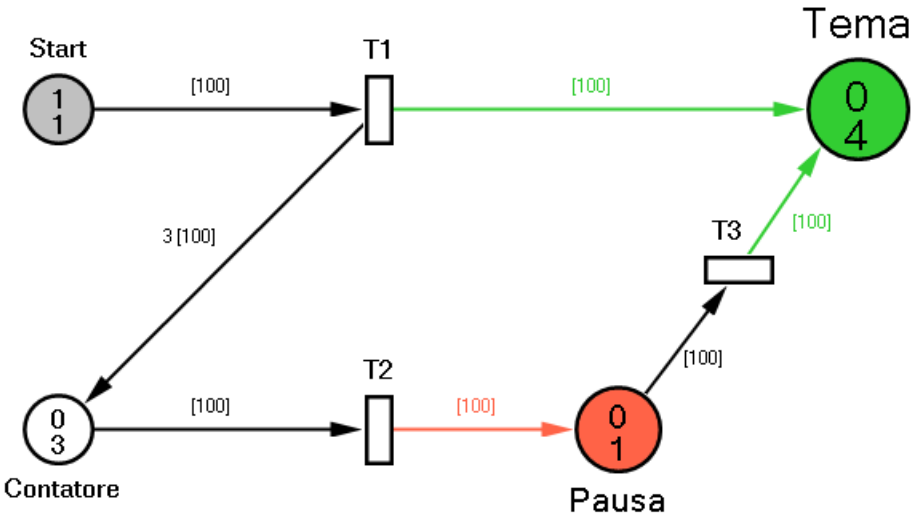

Fig. 17. Canone (prima versione)

Provando ad eseguire la PN creata in ScoreSynth è possibile visionare nella finestra Timeline la struttura delle voci, verificando la corrispondenza con la partitura originale.

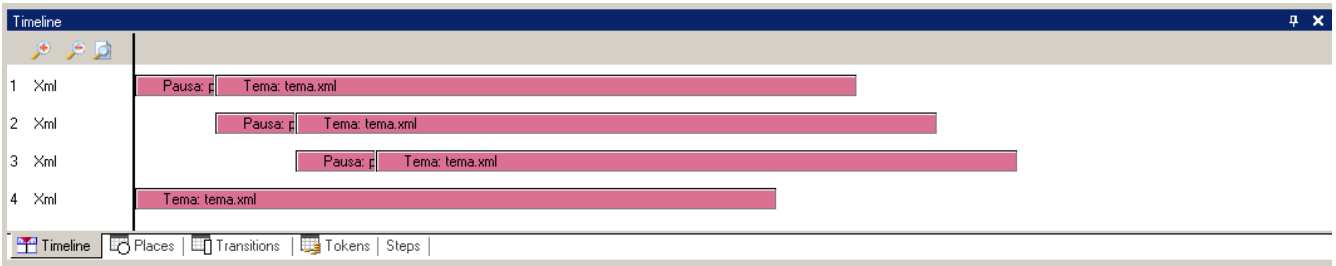

Fig. 18. Timeline dell'esecuzione del canone

Vediamo ora come rendere più evidente dal punto di vista dell'analisi musicale la struttura del canone mostrato come esempio. Vogliamo rappresentare con 4 MX diversi le 4 parti del tema, e non contemplare la presenza di una codifica MX della pausa intercorrente tra le entrate delle diverse voci. Le parti si susseguono una dopo l'altra, dando vita alla PN in Fig. 19, in cui Tema*n* indica la parte *n*-esima del tema.

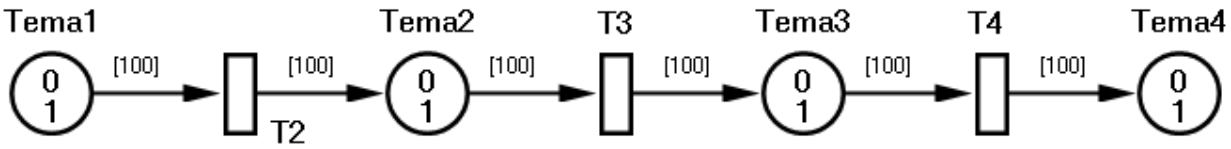

Fig. 19. Scomposizione in parti del tema

Visto che ogni voce successiva inizia quando quella immediatamente precedente ha concluso la propria prima parte, si dovrà aggiungere un posto che inserisca in Tema1 un token quando esso finisce la sua esecuzione precedente, ottenendo per le altre voci lo sfasamento voluto. Provando ad eseguire passo-passo la PN è possibile vedere il tragitto che i tokens percorrono, passando dal posto Tema1 al Tema4, eseguendo ogni frammento MX associato; la rete e la timeline all'ultimo passo dell'esecuzione sono visibili in Fig. 20.

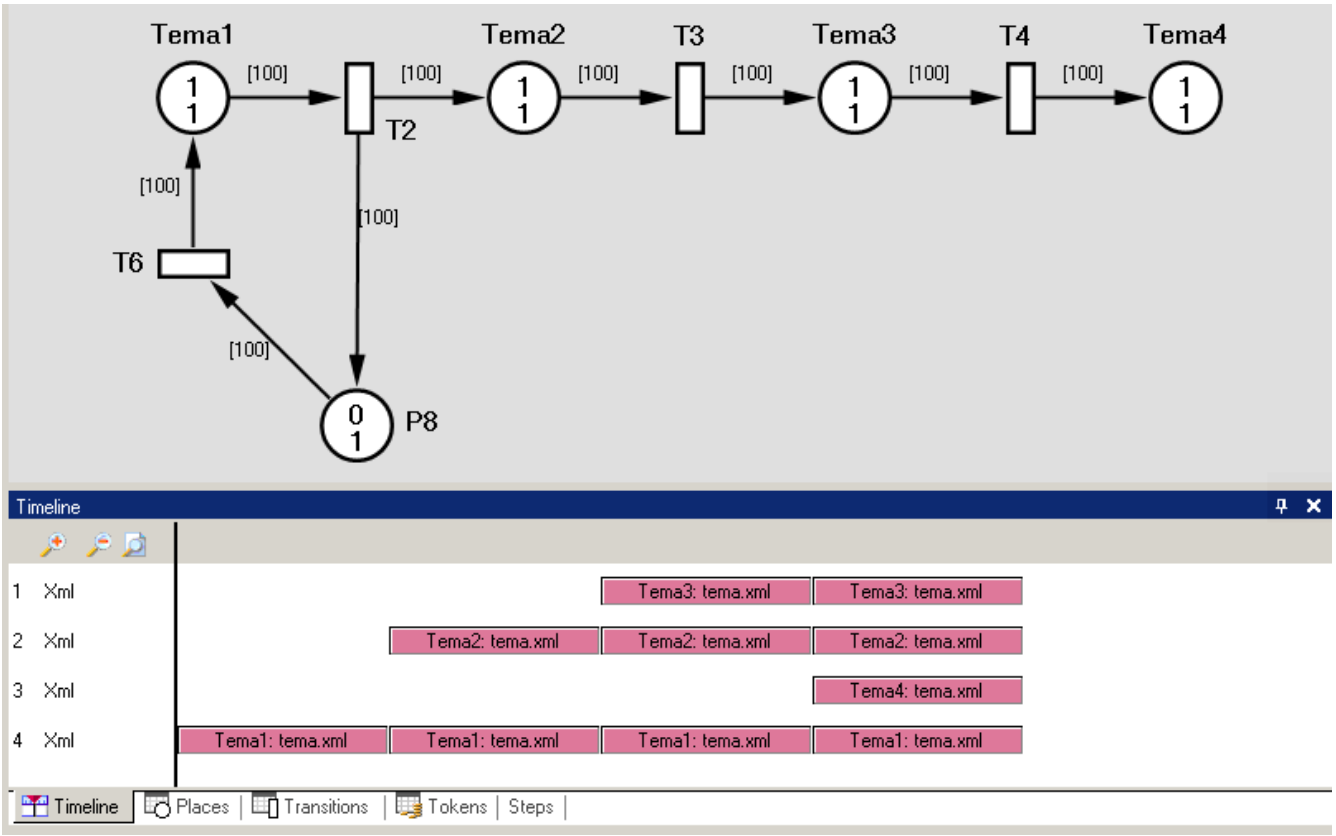

Fig. 20. Ultimo passo della PN e timeline corrispondente

Osservando il comportamento della PN si evidenziano alcuni problemi:

1. il percorso dei token viene bloccato dalla capacità di Tema4, che deve quindi essere aumentata per ospitare gli 8 token corrispondenti al tema completo formato da 4 parti e proposto 2 volte per ogni voce.
2. il posto P8 continua a ricevere e riscaricare token a Tema1, mentre ogni parte del tema deve essere ripetuta 8 volte e poi uscire dall'esecuzione: occorre limitare lo scatto di T6 a 8 volte.

La rete in forma finale è presentata in Fig. 21, mentre in Fig. 22 è rappresentata la rete alla fine dell'esecuzione con la sua timeline; eseguendo la rete passo-passo è possibile osservare come i token lasciano a mano a mano le parti del tema e si accumulano in Tema4.

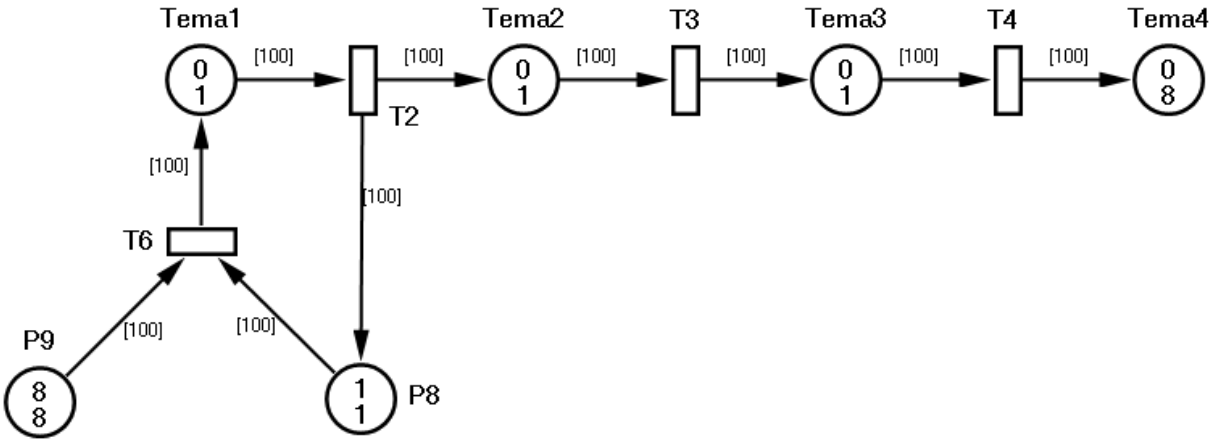

Fig. 21. Canone (seconda versione)

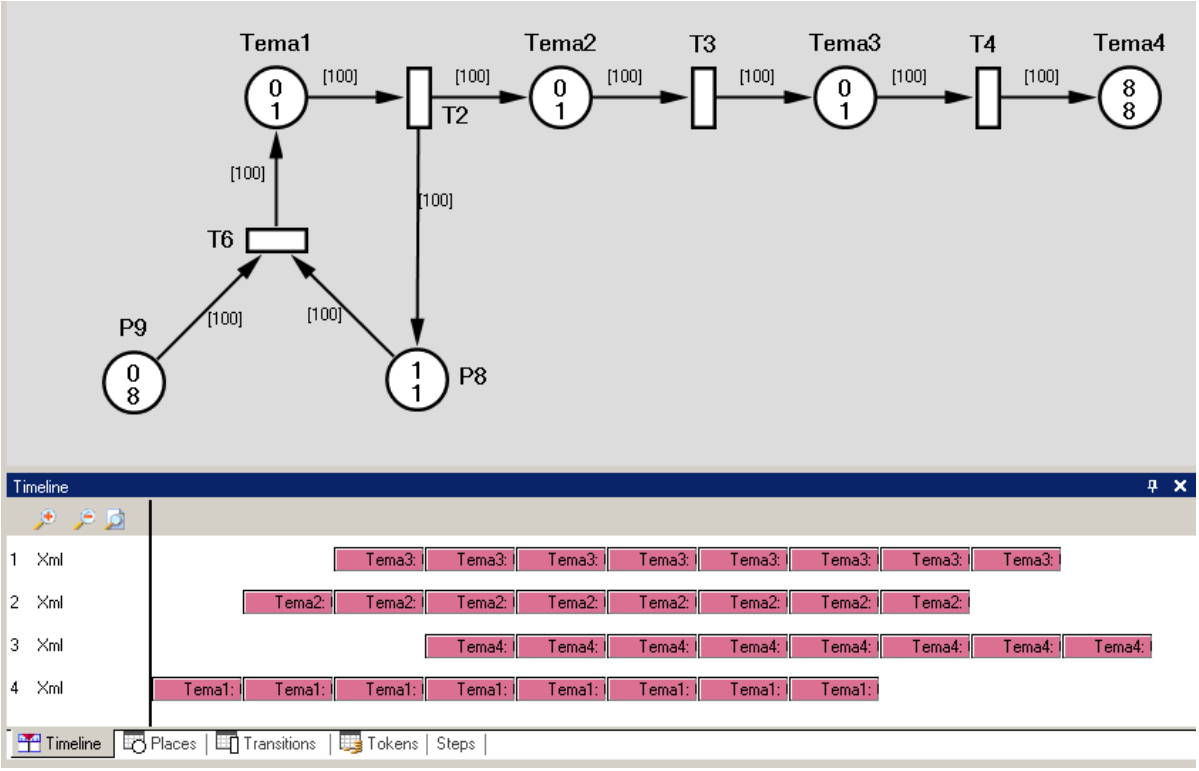

Fig. 22. Ultimo passo del canone con relativa timeline

Come ulteriore esempio vogliamo ora sfruttare le capacità non deterministiche della PN, creando una rete la cui esecuzione crei una composizione con la stessa macrostruttura dell'esempio musicale originale, ma nella quale la successione delle singole parti di ogni esposizione del tema sia casuale. Per esempio, mantenendo la nomenclatura precedente, la prima voce potrà essere costituita dalle parti Tema2 → Tema1 → Tema3 → Tema4, e, nella sua ripetizione, Tema1 → Tema4 → Tema2 → Tema3.

Per iniziare illustriamo in Fig. 23 la struttura classica di scelta che consente, usata come sottorete, di scegliere ad ogni suo utilizzo una delle possibili parti del nostro tema.

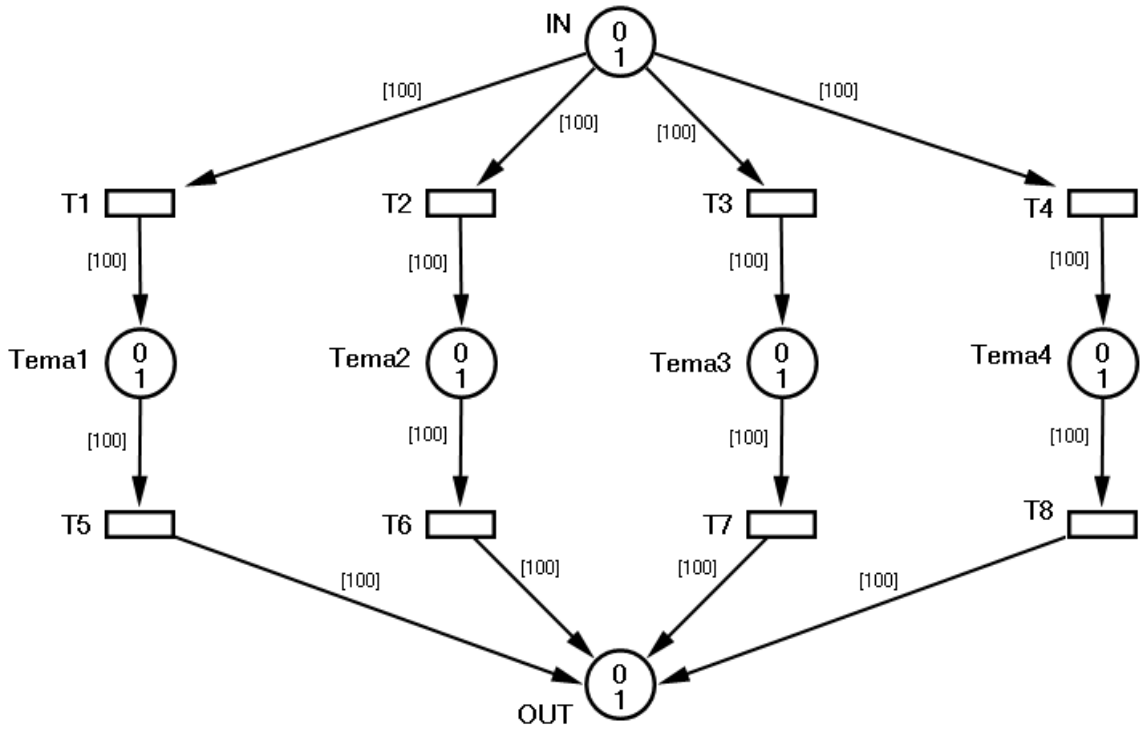

Fig. 23. Sottorete di scelta

Quello che noi vogliamo è però che non si ripeta, in due esecuzioni successive della sottorete, lo stesso frammento di tema. Utilizzando dei posti che blocchino le parti già eseguite in una scelta precedente risolviamo il problema, producendo la rete in Fig. 24.

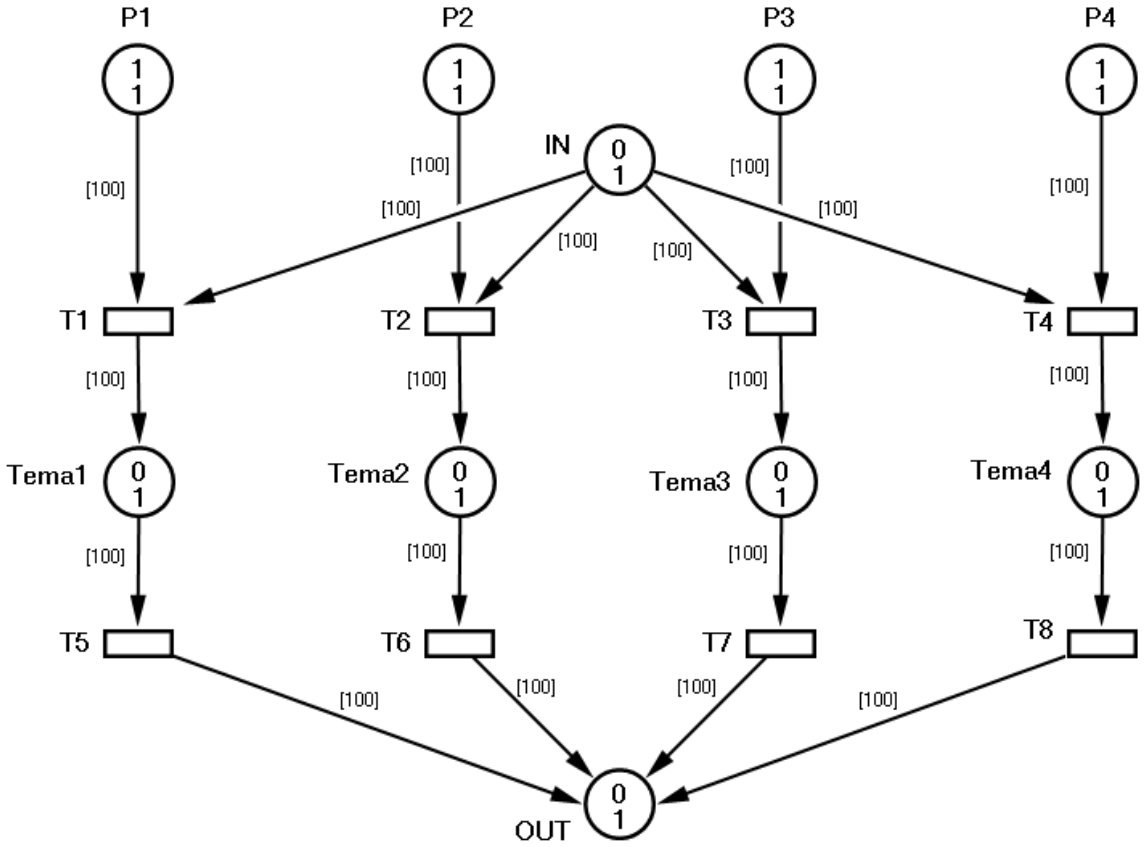

Fig. 24. Sottorete di scelta modificata

Un ulteriore problema che si presenta è che nel nostro esempio musicale il tema viene eseguito 2 volte; questo comporta che nella sottorete presentata, dopo la quarta chiamata, i posti P1, P2, P3 e P4 (che avranno 0 token) non permetteranno lo scatto di nessuna transizione. Occorre quindi che dopo la quarta chiamata vengano reinseriti 4 token nei posti P1...P4, consentendo così una o più ripetizioni del tema: la PN risultante è rappresentata in Fig. 25, con la colorazione delle varie parti per una migliore comprensione delle loro differenti funzioni.

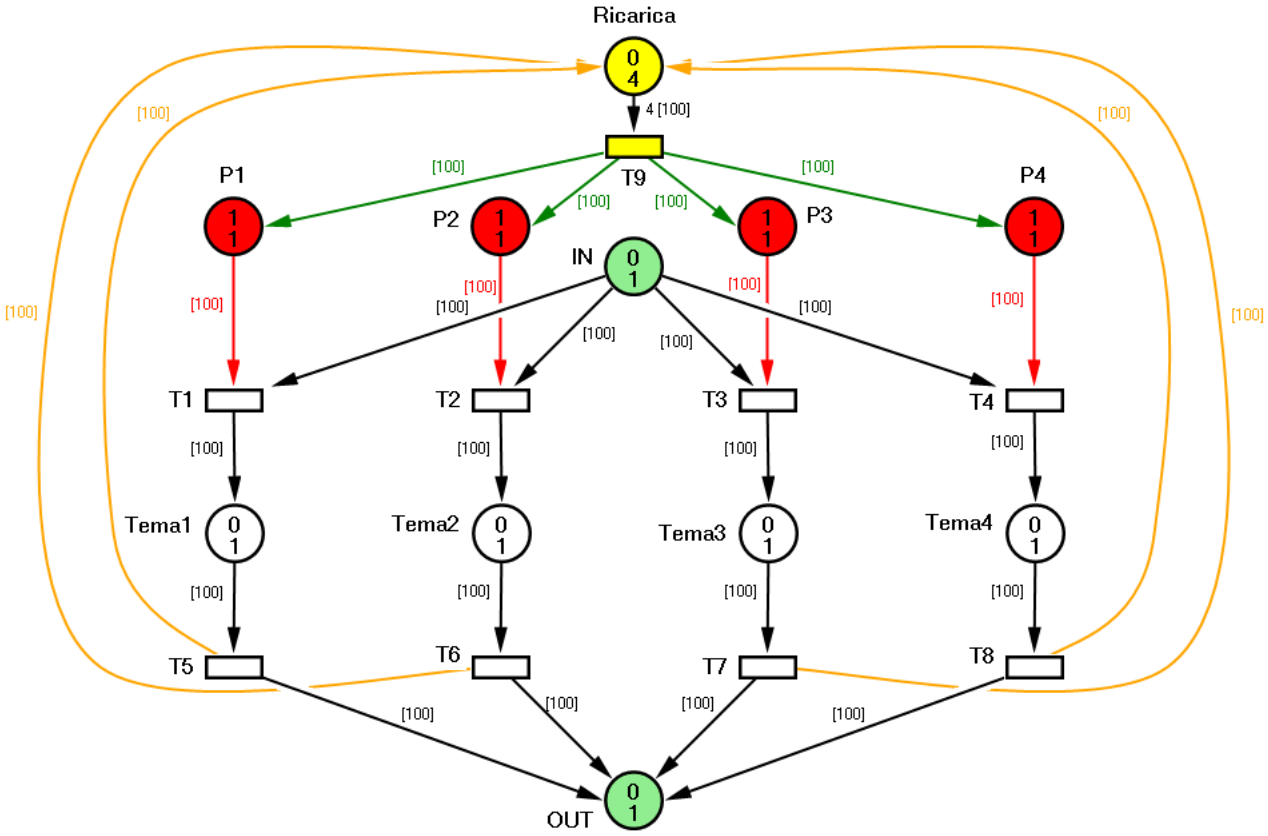

Fig. 25. Sottorete in versione finale

Ora non resta che utilizzare la sottorete creata in una PN completa che sintetizzi il "canone non deterministico"; la rete finale della precedente soluzione si adatta anche a questo scopo, con la differenza che invece di associare ad ogni nodo una parte del tema verrà associata una voce, costituita dalla sottorete appena creata (Fig. 26).

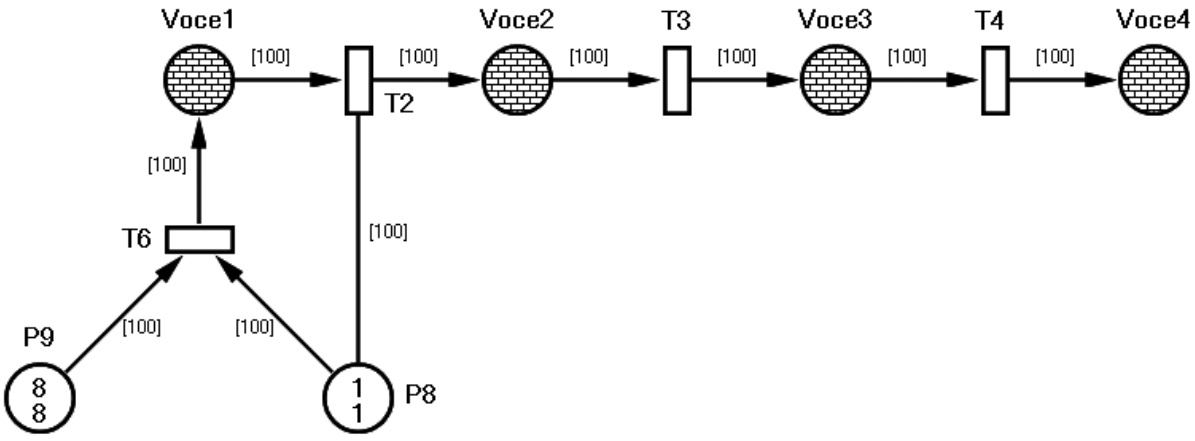

Fig. 26. Canone (terza versione)

Per facilitare la comprensione del comportamento della rete durante il debug sono state associate ai posti Voce1...Voce4 4 diverse sottoreti, in cui ad ogni posto che esegue una parte del tema è stato dato il nome Tema*n*_*m*, dove *n* indica la voce e *m* indica la parte del tema.

Eseguendo la PN in ScoreSynth si evidenzia un ultimo problema: come visibile nella sottorete Voce4 in Fig. 27 e nella timeline finale in Fig. 28, alcuni token vengono bloccati perchè non esistono posti in uscita in grado di ospitarli; questo viene risolto semplicemente cambiando in 4 la capacità del posto OUT nella sottorete Voce4. La timeline corretta che evidenzia la struttura del canone e le scelte operate casualmente nelle parti costituenti i temi è visibile in Fig. 29.

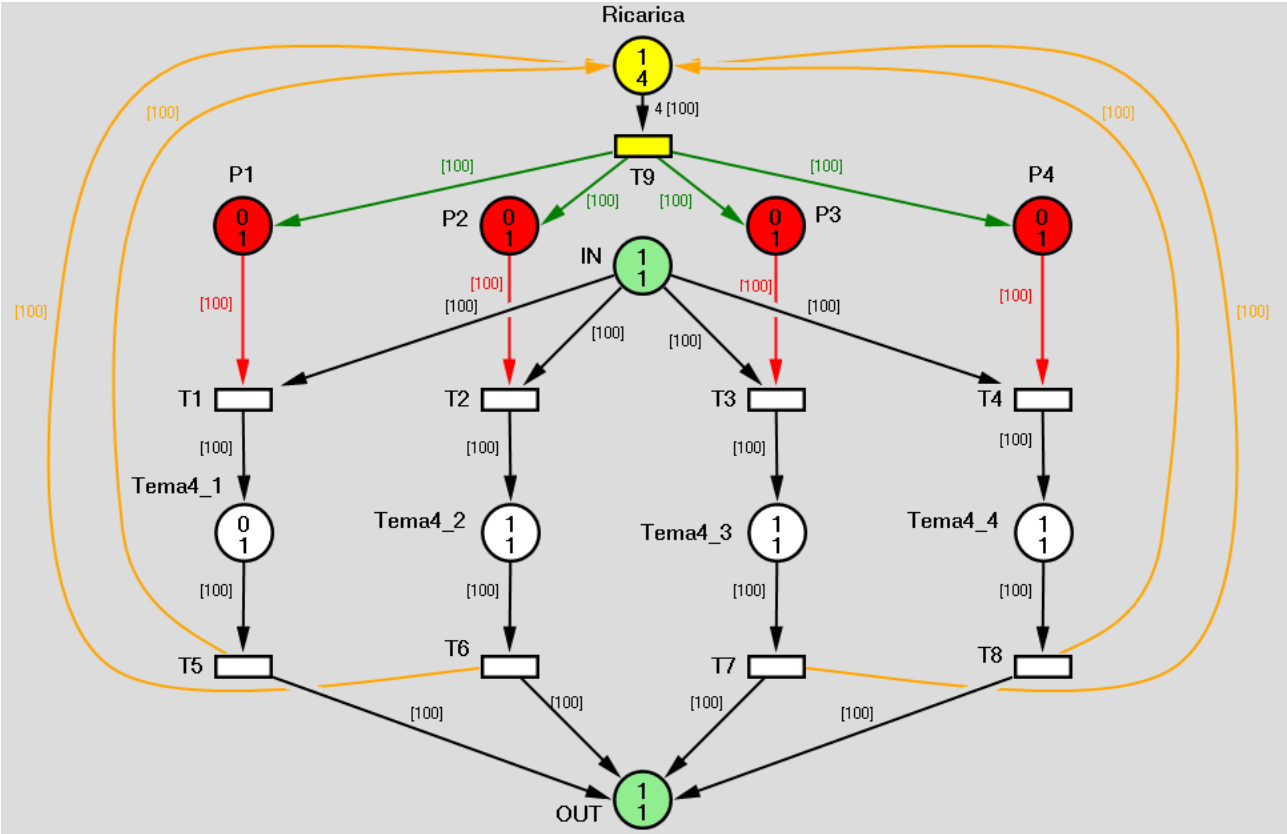

Fig. 27. Sottorete bloccata

I tre approcci mostrati in questo esempio applicativo fanno capire che non esiste sempre una PN "corretta" per modellare una particolare struttura musicale, ma un insieme di possibili soluzioni. Si è inoltre visto come alcune strutture si ripresentano in più punti dei modelli, come i posti che bloccano lo scatto delle transizioni dopo *n* volte (P9 dell'ultima rete principale, o P1...P4 nella sottorete), o l'intera struttura del canone, utilizzata sia per rappresentare le varie parti del tema (nella seconda soluzione) che le diverse voci (nell'ultima).

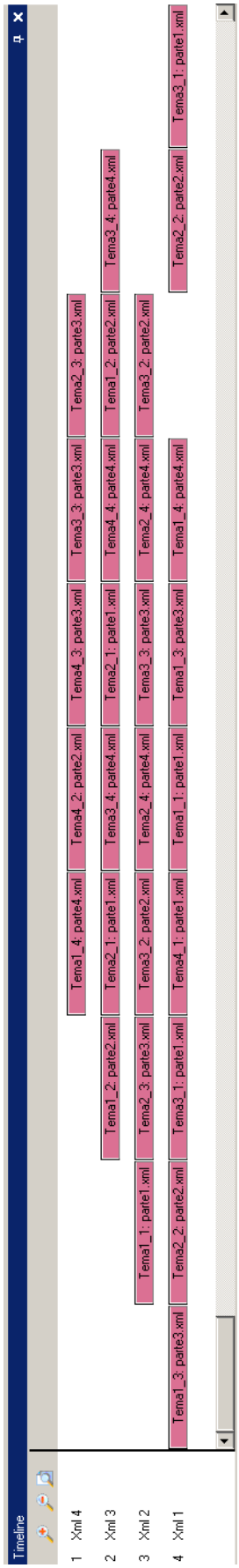

Fig. 28. Timeline con sottorete bloccata

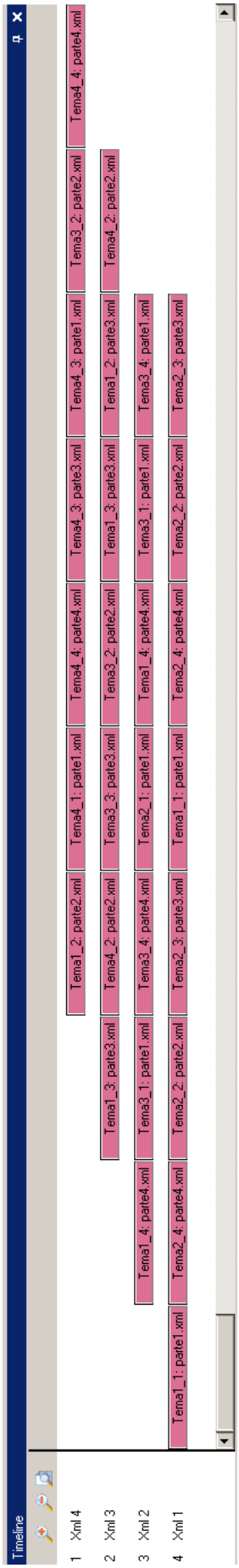

Fig. 29. Timeline con sottorete modificata

CAPITOLO 3

RETI DI PETRI E MULTIMEDIA

3.1 MOTIVAZIONI

Come già detto in precedenza, l'idea di applicare le Reti di Petri alla musica nasce dall'esigenza di voler ampliare il concetto di spartito, al fine sfruttare le potenzialità ed i vantaggi offerti sia dall'audio digitale che dai formalismi delle Reti di Petri, creando così una sorta di spartito multimediale.

Visti i risultati positivi ottenuti nell'ambito musicale, è risultato piuttosto naturale pensare a quali potessero essere i vantaggi introdotti da un tale formalismo applicato alle più generiche opere multimediali, ossia a materiali contententi non solo audio ma anche video ed, eventualmente, immagini e testo.

Ciò ha indotto il L.I.M. (Laboratorio di Informatica Musicale, Università Statale di Milano) a proseguire le proprie ricerche in questo senso, concluse con lo sviluppo di un prototipo SW - MediaSynth - che si propone come strumento generativo di opere multimediali mediante l'uso di PN. Questa applicazione, implementata sia per piattaforme Macintosh [BGH95] che per piattaforme Win32 [Spa04], permette di assegnare *oggetti multimediali* ai posti della rete, ed associare alle transizioni degli algoritmi di trasformazione che permettano di modificare i contenuti dei vari media; l'esecuzione della rete dà come risultato un oggetto multimediale finale contenente l'intera opera multimediale.

Nel prosieguo di questo capitolo, verrà per prima cosa introdotto il concetto di *Authoring System* e di oggetto multimediale; successivamente verrà data una descrizione formale della struttura di PN multimediale, evidenziandone le analogie con le reti musicali. Infine, verranno forniti una serie di esempi volti a mostrare il funzionamento ed i possibili vantaggi forniti da tale sistema, non solo in ambiti prettamente musicali, ma anche in settori industriali quali la cinematografia o la pubblicità

3.2 SISTEMI AUTORE (AUTHORING SYSTEM)

Prima di inziare a parlare dettagliatamente di Reti di Petri multimediali, è necessario introdurre il concetto, molto importante, di *Authoring System* [Den95]. Con questo termine si indicano tutti quei sistemi che permettono di assemblare media diversi quali audio, video, testo, immagini, grafica, in documenti multimediali ove le relazioni tra gli oggetti assemblati hanno una certa logica e una certa sincronizzazione temporale. Inoltre, la maggior parte dei sistemi autore mette a disposizione un linguaggio di *scripting* per creare applicazioni più sofisticate.

Di norma, l'utilizzo dei sistemi autore non richiede una conoscenza tecnica sofisticata e le applicazioni generate sono caratterizzate dal contenere testo, grafica e audio. Tale facilità di sviluppo introduce il notevole vantaggio di una maggiore velocità nella creazione di un'opera multimediale.

Analizziamo ora le caratteristiche principali di un sistema autore. In primo luogo, esso fornisce delle funzioni per il reperimento e la definizione dell'ubicazione dei media; tali risorse possono essere sia interne all'applicazione che esterne e collegate all'opera tramite un *path* locale o un URL. In secondo luogo, i sistemi autore forniscono un'interfaccia grafica per la creazione delle relazione tra i vari oggetti multimediali a cui possono essere associati algoritmi attivabili in funzione di un particolare evento (ad es. 'OnMouseClick'). Generalmente, questi algoritmi sono scritti o con un linguaggio di script interno all'*authoring system*, o con linguaggi di programmazione "compilati" (ad es. C/C++, Java, ecc.), direttamente inglobati nell'opera tramite librerie. Infine, gli Authoring System forniscono una piattaforma SW che rende l'esecuzione delle opere multimediali create indipendenti dal sistema operativo e dalla piattaforma HW utilizzata in fase di sviluppo. Tale piattaforma prevede, tra le altre cose, un insieme di *codec* per l'*encoding* ed il *decoding* dei vari media (ad es. JPG, MP3, AAC, ecc.) impiegati.

Generalmente, in ogni sistema autore è presente una barra di pulsanti che permette di importare oggetti multimediali, creare le relazioni tra di essi ed associare eventuali algoritmi. Da tale barra è poi possibile eseguire l'opera o esportarla in un formato standard per la rappresentazione audio/video (ad es. MPEG, ecc.).

Un altro aspetto di fondamentale importanza da considerare è il concetto di tempo. Di norma, le informazioni sono strutturate sotto forma di griglia, spesso indicata con il termine *timeline,* dove le righe contengono i diversi oggetti multimediali e le colonne rappresentano i vari "istanti" temporali; due oggetti che si trovano sulla stessa colonna vengono eseguiti contemporaneamente.

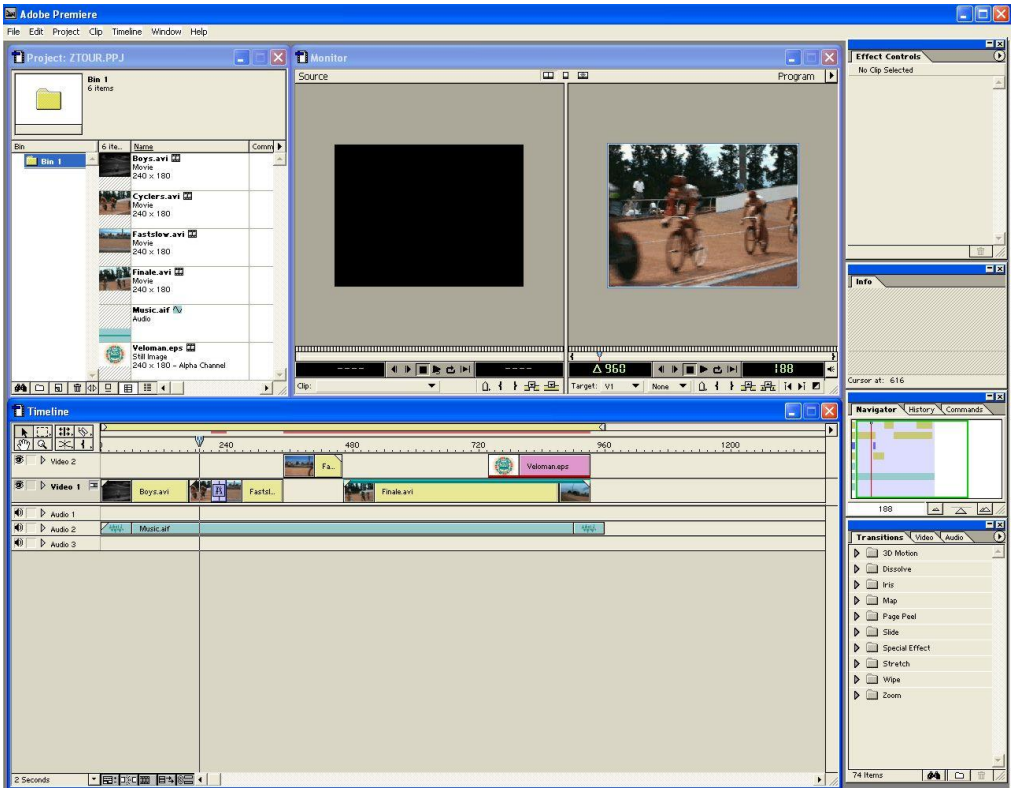

Fig. 30: Esempio di ambiente di Authoring (Adobe Premiere)

Un'ultima caratteristica che i moderni sistemi autore forniscono è la possibilità di creare applicazioni basate su architettura Client/Server. Una tale funzione permette la creazione di opere multimediali provviste di funzioni per lo *streaming* e fortemente integrabili all'interno di applicazioni Web.

3.3 GLI OGGETTI MULTIMEDIALI

Come già accennato in precedenza, ai posti di una PN multimediale sono associati degli oggetti multimediali; in questo paragrafo ne daremo una definizione precisa.

Per *oggetto multimediale* intendiamo un qualsiasi clip audio, frammento video, immagine o testo, codificato secondo uno dei svariati standard esistenti [Ver04b]. Tali oggetti possono contenere, per esempio, *ciak* cinematografici (per esempio, in formato WMV), sfondi fissi (per esempio, in formato JPG), *loop* musicali, basi strumentali, effetti speciali, ecc. (per esempio, in formato non compresso WAV o compresso MP3).

L'insieme delle codifiche che l'applicazione è in grado di supportare dipende strettamente dal tipo di piattaforma su cui essa si appoggia. Per esempio, il prototipo *MediaSynth Win32* [Spa04] sfrutta DirectX 9.0, perciò, l'insieme dei formati utilizzabili nell'applicazione sarà l'insieme (o un suo sottoinsieme) di ciò che tale tecnologia fornisce:

- *Formati di file*:
  - Windows Media™ Audio (WMA) [1]
  - Windows Media™ Video (WMV) [1]
  - Advanced Systems Format (ASF) [1]
  - Motion Picture Experts Group (MPEG)
  - Audio-Video Interleaved (AVI)
  - QuickTime (versione 2 e precedenti)
  - WAV
  - AIFF
  - AU
  - SND
  - MIDI

- *Formati di compressione*:
  - Windows Media Video [2]
  - ISO MPEG-4 video version 1.0 [1]
  - Microsoft MPEG-4 version 3 [1]
  - Sipro Labs ACELP [1]
  - Windows Media Audio [1]
  - MPEG Audio Layer-3 (MP3) (solo decompressione)
  - Digital Video (DV)
  - MPEG-1
  - MJPEG
  - Cinepak

[2] Le applicazioni DirectX devono usare il Windows Media™ Format SDK per supportare questo formato. Microsoft non fornisce un decoder MPEG-2. Vari decoder MPEG-2 hardware e software compatibili con DirectShow sono resi disponibili da terze parti.

Un’osservazione va fatta sull’oggetto multimediale finale ottenuto dall’esecuzione della Rete di Petri: esso conterrà al suo interno una sequenza *ordinata* di diversi media, quindi, il formato utilizzato per la sua rappresentazione dovrà poter codificare contemporaneamente dati mediali di natura diversa (audio, video, ecc.) e le informazioni di sincronizzazione tra di essi. Esempi di tali codifiche sono l’AVI (Audio-Video Interleaved) o l’MP4 [Ver04a].

3.4 RETI DI PETRI MULTIMEDIALI

Le *Reti di Petri multimediali* forniscono un approccio grafico, semplice ed intuitivo per la creazione di opere multimediali. Esse rappresentano una naturale estensione delle Reti di Petri musicali, e possono essere viste come un particolare *sistema autore*, provvisto però di formalismi del tutto diversi da quelli presenti nei prodotti attualmente in commercio. Di fatto, permettono di sfruttare tutti i vantaggi di semplicità forniti da un *authoring system* ed i benefici introdotti dall'impiego del formalismo delle Reti di Petri (non-determinismo, asincronismo, parallelismo, ecc.)

In una PN multimediale, ai posti vengono associati degli oggetti multimediali, mentre alle transizioni sono associati opportuni algoritmi di manipolazione dei media. Perciò, come accade per un'elaborazione musicale, anche un'opera multimediale può essere descritta in termini di evoluzione di uno o più oggetti multimediali, opportunamente trasformati da algoritmi associati alle transizioni.

Partendo dai concetti introdotti con le PN musicali (Cap. 2), analizziamo le caratteristiche di quelle multimediali, evidenziandone analogie e differenze:

- ai posti non vengono associati frammenti musicali ma generici oggetti multimediali contenenti audio vocale, audio musicale, video o, eventualmente, testo e immagini;
- gli algoritmi associati alle transizioni sono riferiti al media e non al frammento musicale. Perciò, non si parla più ad esempio di trasposizione o retrogradazione ma, se consideriamo il caso di media audio, di algoritmi di *editing* classico o diretto su formati compressi [Ver04c] (*fade*, *pitch shifting*, *time stretching*, filtri, EQ, ecc.)
- per la composizione di opere multimediali ci si avvale delle medesime strutture di PN impiegate nelle reti musicali, già ampiamente descritte nei par. 1.5 e 2.3 (Fig. 12);
- la gestione della temporizzazione in fase di esecuzione è identica a quella impiegata nelle PN musicali (par. 2.4). Ciò che cambia è il tempo di esecuzione del posto, che in questo caso dipende strettamente dalla durata dell'oggetto multimediale e non musicale. In altre parole, si verifica lo scatto di una transizione solo quando è terminata l'esecuzione dell'oggetto multimediale associato al posto.

Il risultato finale ottenuto dall'esecuzione della PN è una sequenza di media di natura diversa, temporalmente ordinati all'interno di una *timeline* (Fig. 31), struttura su cui si basano gli attuali sistemi autori. E' dunque possibile collocare il formalismo delle PN per la descrizione dell'informazione multimediale ad un livello più astratto, gerarchicamente superiore rispetto a

quello fornito dagli attuali sistemi autore, in quanto basato sulla *struttura* dell'opera e non sulla *sequenza ordinata* di media.

Perciò, come accade per l'informazione musicale, modellizzare un'opera multimediale con PN rende naturale la descrizione del suo livello strutturale come un ambiente multilivello al cui interno gli oggetti multimediali possono fluire in modo concorrente e interattivo.

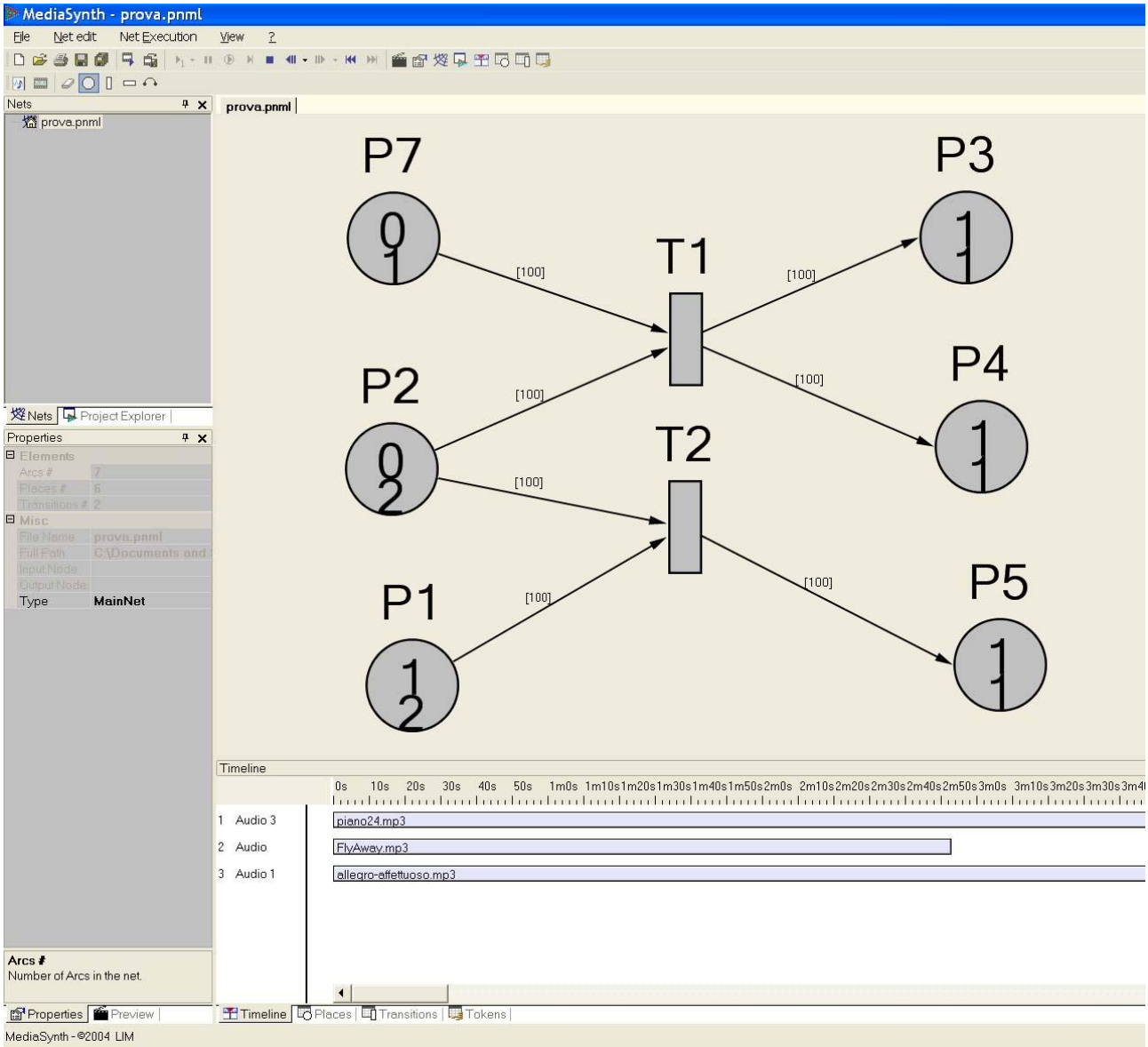

Fig. 31: Esempio d'esecuzione di una rete di petri multimediale in MediaSynth [Spa04]; si può vedere la Rete di Petri (in centro) e la relativa timeline creata (in basso a destra)

3.5 APPLICAZIONI DELLE RETI DI PETRI MULTIMEDIALI

In questo paragrafo analizziamo alcuni esempi concreti ove sia possibile sfruttare le potenzialità offerte dalle Reti di Petri multimediali. Nel prossimo paragrafo invece, verrà mostrato come creare, passo-passo, una sequenza di trailers cinematografici sfruttando molte delle caratteristiche fornite dalle PN.

Si supponga per esempio di dover costruire una serie di spot pubblicitari che differiscano solo in alcune parti (la colonna sonora, il filmato, alcuni effetti sul video o sull'audio). In tal caso è necessario costruire una sola volta la struttura della rete e modificare via via il o i parametri che differenziano le varie sequenze da generare (oggetti multimediali, numero delle marche, ecc.) in funzione di ciò che si vuole creare. In questo modo si possono ottenere una serie di spot pubblicitari con medesimo modello generativo ma con differenti contenuti.

Anche la creazione di filmati audio-video può essere eseguita efficientemente tramite le Reti di Petri multimediali. Si pensi ad esempio ai trailer cinematografici: per pubblicizzare nuovi film in uscita al cinema, vengono sempre realizzati diversi spot con durate e contenuti differenti in funzione del contesto in cui sono presentati. Generalmente vengono scelte e messe in sequenza diverse scene del film che ne evidenziano i momenti salienti, a cui vengono associate opportune basi vocali e/o musicali.

Infine, le PN possono introdurre notevoli vantaggi nello sviluppo di opere multimediali come videoclip musicali, cortometraggi televisivi o jingle radiofonici.

3.6 REALIZZAZIONE DI TRAILER CINEMATOGRAFICI

Per la realizzazione di trailer cinematografici, è stato creato un modello di PN composto da una rete principale e da una sottorete, la quale viene eseguita per un certo numero di volte (nel nostro esempio, questo valore è pari a 3). Tale sottorete, essendo non deterministica, permette di ottenere filmati di uguale durata, ma con scene (oggetti multimediali) differenti, il tutto senza dover operare nessun tipo di variazione sulla struttura della rete stessa. Indubbiamente un bel risparmio di tempo e di lavoro!

Con dei semplici ampliamenti sarebbe possibile anche inserire l'audio corrispondente ad ogni scena del trailer ed una colonna sonora della durata dell'intero video.

3.6.1 RETE PRINCIPALE

Fra le strutture elementari di PN, quella che permette di eseguire un numero prefissato di volte un posto è il *loop* (par. 1.5).

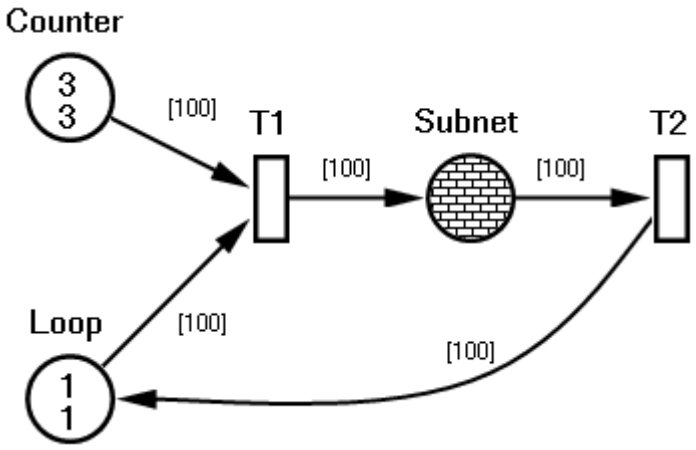

Fig. 32: Struttura di un loop

Il numero di *token* presenti nel posto *Counter* determina il numero di esecuzioni del ciclo. A ogni esecuzione, il numero di marche in *Counter* diminuisce di 1: quando *Counter* non contiene più *token* l'esecuzione si ferma, in quanto la transizione **T1** non risulta più abilitata.

Generalmente i trailer cominciano e finiscono con uno spezzone particolare (che può contenere il logo della casa cinematografica, il nome dei film, il cast, ecc.). È quindi necessario modificare la rete in modo che sia possibile inserire un filmato iniziale ed uno finale prefissati.

Per aggiungere un filmato iniziale basta inserire prima di *Counter* un posto *Clip Start* che contenga tale filmato. Dopo l'esecuzione di *Clip Start* viene avviata la struttura *Loop* che inserisce nel trailer il numero di scene volute, scelte nella sottorete contenuta in *Subnet*.

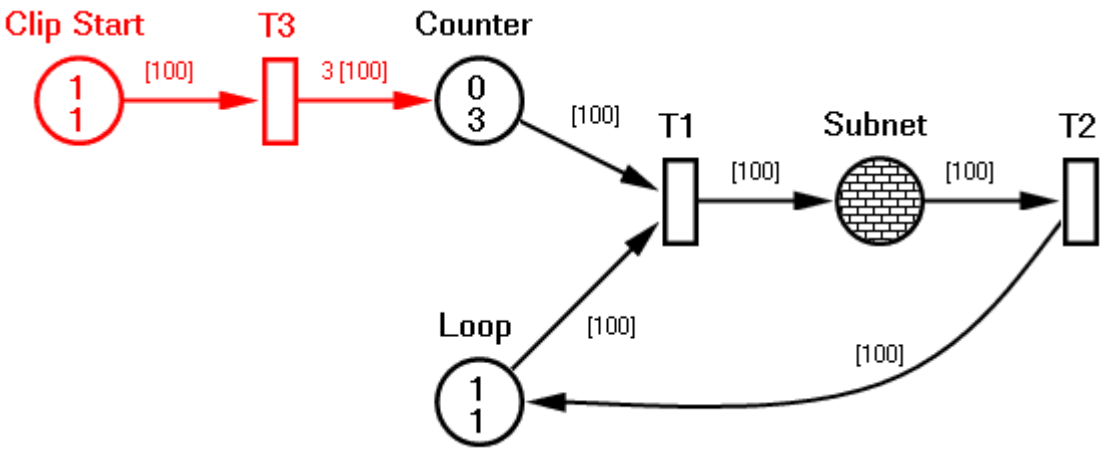

Fig. 33: Inserimento del filmato iniziale

Cambiando il peso dell'arco fra **T3** e *Counter*, e la capacità di *Counter*, si può variare il numero di esecuzioni della struttura loop. Nell'esempio in Fig. 33 il ciclo viene eseguito tre volte.

Per aggiungere un filmato finale si può sfruttare il peso probabilistico degli archi: al posto *Loop* si collega una transizione **T4** con un arco con peso probabilistico 0. In uscita a **T4** si aggiunge un posto *Clip End* che contiene il filmato finale.

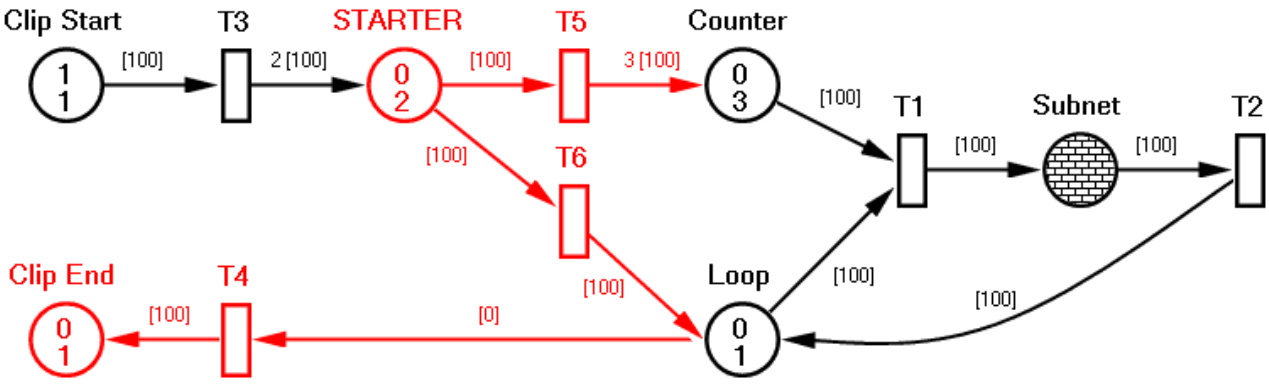

Fig. 34: Inserimento del filmato finale

Affinché la rete funzioni nel modo voluto è necessario aggiungere un nodo *STARTER* che abiliti il ciclo. Senza questo accorgimento, infatti, il token nel posto *Loop* farebbe scattare la transizione **T4** al primo passo di esecuzione.

Il peso probabilistico 0 dell'arco fra *Loop* e **T4** garantisce che la transizione **T4** non scatti se la transizione **T1** è abilitata, ovvero finché ci sono dei token nel posto *Counter*, ma scatti solo quando il ciclo è stato eseguito per il numero di volte prefissato.

3.6.2 SOTTORETE

Ad ogni ciclo viene eseguita la sottorete referenziata dal nodo *Subnet*. Questa sottorete deve garantire che venga aggiunta alla *timeline* una scena a caso, diversa ad ogni esecuzione. Per implementare questo requisito si utilizzano due strutture elementari delle Reti di Petri messe in parallelo: la *fusione* (per la non duplicazione delle scene) e l'*alimentazione alternativa* (per il non determinismo).

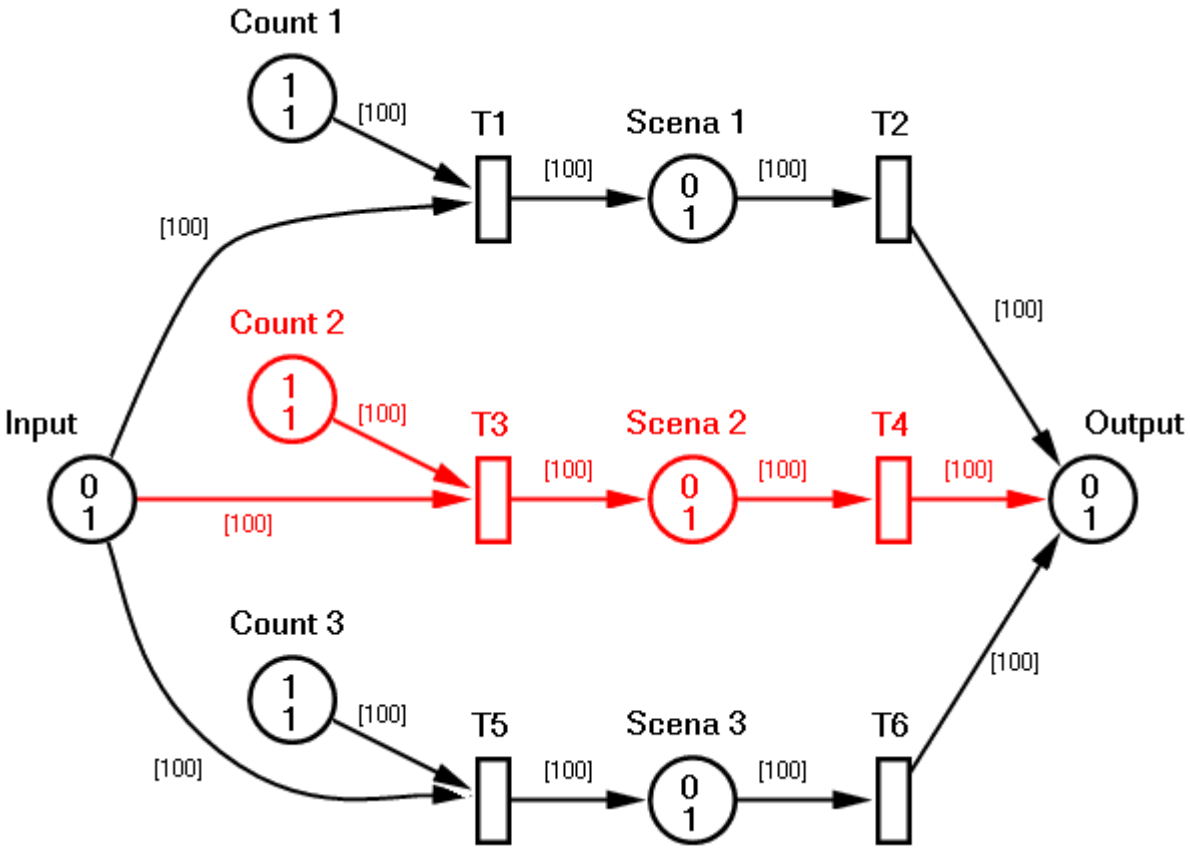

Fig. 35: Sottorete (non deterministica) per la scelta delle scene

Quando un *token* raggiunge il posto *Input*, si ha una situazione di alternativa e l'algoritmo di esecuzione sceglie casualmente quale delle transizioni abilitate (**T1**, **T3**, e **T5**) far scattare. Lo scatto della transizione scelta porta il *token* nel posto contenente una delle scene (*Scena 1*, *Scena 2* o *Scena 3*) e azzera il relativo posto *Count*. In questo modo si inibisce lo scatto della stessa transizione in una successiva esecuzione della sottorete.

L’esecuzione continua con l’inserimento del filmato scelto in *timeline* e con lo spostamento del *token* nel posto *Output* e successivamente nella rete principale.

Risulta evidente che le possibilità di generare filmati sempre diversi aumentano proporzionalmente col numero di strutture elementari inserite nella sottorete.

Variando i pesi probabilistici degli archi uscenti dal nodo *Input* si può anche fare in modo che certe scene siano scelte con più probabilità di altre.

3.6.3 RISULTATO DELL’ESECUZIONE

Il risultato dell’esecuzione della rete è una sequenza di clip video che di cui solo il primo e l’ultimo sono noti a priori. La parte centrale contiene infatti un numero di scene pari al peso dell’arco tra **T5** e *Counter*, scelte a caso fra tutte le scene disponibili e il cui ordine può variare per ogni esecuzione.

Il valore di *Counter* non deve mai essere superiore al numero di scene disponibili nella sottorete, altrimenti l’esecuzione si blocca prima di raggiungere il posto *Clip End*.

Di seguito sono riportati alcuni esempi di esecuzione della rete (il valore di *Counter* è riportato per ogni esempio).

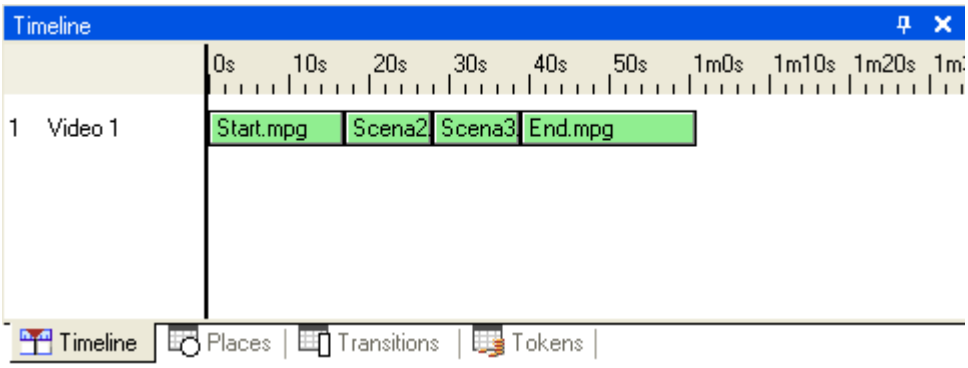

Fig. 36: Esecuzione della rete (Counter = 2)

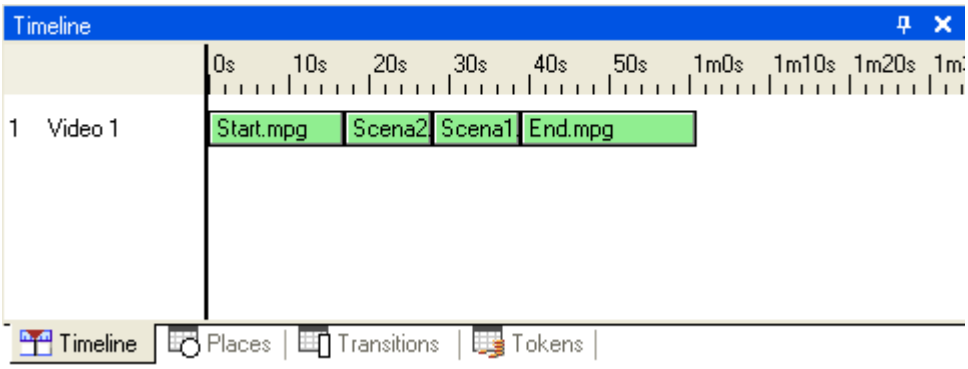

Fig. 37: Esecuzione della rete (Counter = 2)

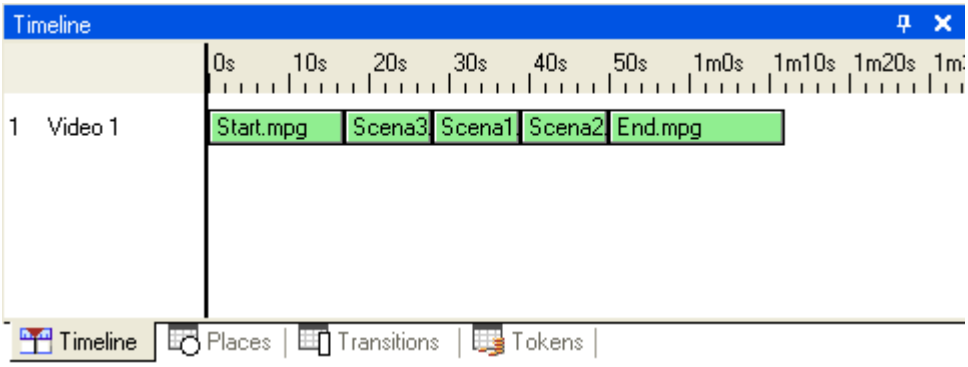

Fig. 38: Esecuzione della rete (Counter = 3)

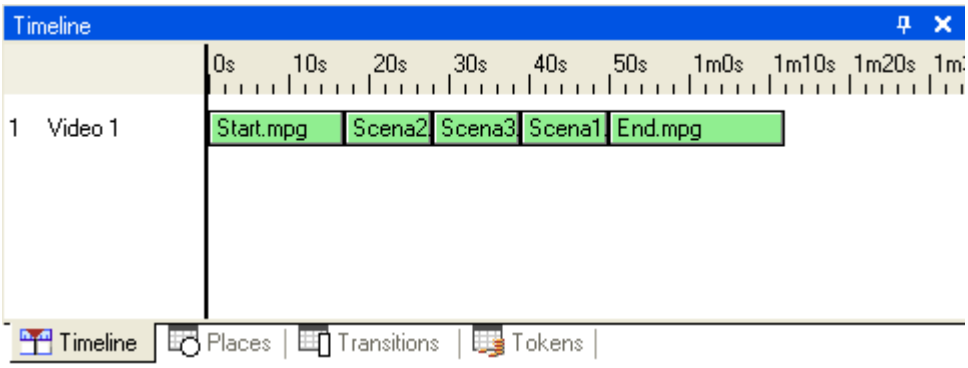

Fig. 39: Esecuzione della rete (Counter = 3)

BIBLIOGRAFIA

[Bar04] Baratè, A., *Un sistema per la generazione di documenti musicali XML basato su Reti di Petri*, Tesi di Laurea in Scienze dell'Informazione, Università di Milano, 2004

[BGH95] Belletti, T., Galizia, G., e Haus, G., *MediaSynth: a System for the Synthesis of Multimedia Scores based on Petri Nets*, Atti del 11° Colloquio di Informatica Musicale, pp. 107-110, Università di Bologna, 1995

[CHIZ86] Camurri, A., Haus, G., Iacomini, G., e Zaccaria, R., *Il sistema MAP per il controllo del CMI Fairlight*, Atti del VI Colloquio di Informatica Musicale (1985), Università di Napoli, pp. 209-225, UNICOPLI, Milano, 1987

[CHZ86] Camurri, A., Haus, G., e Zaccaria, R., *Describing and Performing Musical Processes by Means of Petri Nets*, Interface, Vol. 15, N.1, pp.1-25, Swets & Zeitlinger B.V., Amsterdam, 1986

[Den95] Dennis, B. M., e Arrison, M. A., *Technical Issues in Hypermedia Authoring System*, Compcon '95 "Technologies for the Information Superhighway", Digest of Papers, 5-9 Pages 373 – 378, Div. of Computer Science, California Univ., Berkeley, CA, USA, March 1995

[Des00] Desel, J., *Place/Transition Nets*, 21st International Conference on Application and Theory of Petri Nets, Introductive Tutorial, Catholic University in Eichstätt, 2000

[DH82] Degli Antoni, G., e Haus, G., *Music and Causality*, Proceedings 1982 International Computer Music Conference, pp.279-296, La Biennale, Venezia, Computer Music Association Publ., San Francisco, 1983

[DH85] Degli Antoni, G., e Haus, G., *Netz-Reprasentationen von Musikstucken* in *Musik-Psychologie. Ein Handbuch in Schlusselbegriffen*, Ed. Bruhn/Oerter/Rosing, pp.141-148, Urban & Schwarzenberg Publ., Monaco, Germania, 1985

[DH93] De Matteis, A., e Haus, G., *Formalizzazione di strutture generative all'interno de La*

sagra della primavera, Atti del 10° Colloquio di Informatica Musicale, pp. 48-54, Università di Milano, 1993

[DH96] De Matteis, A., e Haus, G., *Formalization of Generative Structures within Stravinsky's The Rite of Spring*, Journal of New Music Research, Vol.25, N.1, pp. 47-76, Swets & Zeitlinger B.V., Amsterdam, 1996

[Hau84] Haus, G., *Elementi di Informatica Musicale*, Gruppo editoriale Jackson, Milano, 1984

[Hau88a] Haus, G., *PETREX Reference Manual*, LIM-DSI Internal Report, 60pp., Università degli Studi, Milano, 1988

[Hau88b] Haus, G., *PETREX Tutorial Manual*, LIM-DSI Internal Report, 65pp., Università degli Studi, Milano, 1988

[HR88] Haus, G., e Rodriguez, A., *Music Description and Processing by Petri Nets*, Rapporto interno D.S.I., L.I.M. Università degli Studi, Milano, 1988

[HR93] Haus, G., e Rodriguez, A., *Formal Music Representation; a Case Study: the Model of Ravel's Bolero by Petri Nets* in *Music Processing*, G. Haus Editor, Computer Music and Digital Audio Series, pp.165-232, A-R Editions, Madison, WI, 1993

[HS91] Haus, G., e Sametti, A., *Un sistema per la sintesi di partiture MIDI mediante esecuzione di modelli formali*, Atti del 9° Colloquio di Informatica Musicale, pp. 164-177, Università di Genova, 1991

[HS92] Haus, G., e Sametti, A., *ScoreSynth: A System of Music Scorse Based on Petri Nets and a Music Algebra* in Baggi, D., *Readings in Computer-Generated Music*, IEEE Computer Society Press, Los Alamitos, California, 1992

[HS94] Haus, G., e Sametti, A., *Modelling and Generating Musical Scores by Petri Nets*, Languages of Design, Vol. 2, N. 1, pp. 7-24, Elsevier Publ., Amsterdam, 1994

[Pet76] Petri, C. A., *General Net Theory*, Proceedings of the Joint IBM & Newcastle upon Tyne Seminar on Computer Systems Design, 1976 **J.**

[Pet81] Peterson, J. L., *Petri Net Theory and the Modeling of Systems*, Prentice Hall, New Jersey, 1981

[Rei00] Reisig, W., *An Informal Introduction to Petri Nets*, 21st International Conference on Application and Theory of Petri Nets, Introductive Tutorial, Humboldt University of Berlin, 2000

[Spa04] Spagliardi, S., *Modelli e processi software per la generazione automatica di processi multimediali in ambiente Windows*, Tesi di Laurea in Informatica, Università di Milano, 2004

[Ver04a] Vercellesi, G., *MP4 - AAC Overview*, dispense del corso di Informatica Applicata alla Musica, Dipartimento di Informatica e Comunicazione, Università degli Studi di Milano, 2004

[Ver04b] Vercellesi, G., *MPEG / Audio Tutorial*, dispense del corso di Informatica Applicata alla Musica, Dipartimento di Informatica e Comunicazione, Università degli Studi di Milano, 2004

[Ver04c] Vercellesi, G., *Manipolazione Diretta di Formati Audio Compressi MP3*, dispense del corso di Informatica Applicata alla Musica, Dipartimento di Informatica e Comunicazione, Università degli Studi di Milano, 2004