1. Concetti generali

Per concetti generali, intendo la metodologia di base per la composizione con Opusmodus, le connessioni tra i parametri della musica e la produzione del testo-codice in questo ambiente informatico.

1.1. Astrazione dei parametri

La filosofia di base in Opusmodus è trattare la musica come un'entità parametrica in cui durate, altezze, dinamiche (velocità) ed espressioni (attributi) possano essere trattate in modo autonomo e con propri specifici parametri.

Questa filosofia di trattare ogni aspetto della sintassi musicale separatamente consente una grande flessibilità e potenza nel fornire un'ampia gamma di possibilità correlate strettamente al DNA stesso del materiale trattato. In questo modo si possono apportare variazioni alle altezze mantenendo intatta la struttura della durate e, viceversa, apportare variazioni alle durate mantenendo intatta la struttura delle altezze. È anche possibile utilizzare funzioni che vincolano i parametri fra loro, dove un parametro interferisce nell'output dell'altro.

1.2. OMN: I quattro elementi

(citazione da *Introduzione al Linguaggio OMN*)

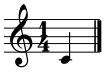
<durata>
OMN è stato creato pensando di porre come primo elemento il TEMPO. Dopotutto noi possiamo far musica anche senza una precisa intonazione delle note. Se ci accingiamo a compilare un testo OMN per comporre musica, in primo luogo necessitiamo di un manuale di riferimento che ci sia di aiuto mentre apprendiamo i fondamenti del linguaggio OMN. Pertanto abbiamo voluto completare questa introduzione con un dizionario speciale dei termini articolato nei quattro principali elementi che compongono il linguaggio OMN. Innanzitutto occorrono alcune ridefinizioni preliminari. TEMPO infatti è una nozione troppo generica che occorre ridefinire in ter-

mini di ritmo e durata: quando vogliamo parlare di come si organizza il ritmo in notazione musicale di solito lo descriviamo come una commistione di simboli che identificano differenti durate. Perciò nel vocabolario di OMN si è deciso di utilizzare il termine DURATA quale definizione generale nel far riferimento al tempo.

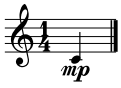
```
(q)
```

<altezza>

Il secondo elemento del linguaggio OMN è l'ALTEZZA. Sebbene un qualsiasi pezzo di musica possa essere definito come "una durata in un segmento di tempo", un brano musicale comincia ad esistere come entità musicale specifica se e solo quando al suono prodotto viene attribuita una determinata altezza.

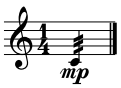
```
(q c4)
```

<intensità>

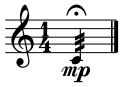
Il terzo elemento costitutivo del linguaggio OMN è l'INTENSITÀ. La tradizionale notazione su pentagramma dispone di un insieme di simboli di uso comune che sono ricavati dalla prima lettera della parola in lingua italiana che identifica il grado di intensità di una o più note a seguire. In OMN sono stati stabiliti 12 livelli di intensità cui corrispondono altrettanti specifici simboli, da `ppppp` a `fffff`. In più OMN include molti altri simboli che però possono essere classificati soltanto come "dinamiche" generiche perché ad essi non corrisponde nessun preciso valore (come dato numerico).

```
(q c4 mp)
```

<attributo>

Il quarto elemento del linguaggio OMN è l'ATTRIBUTO. Il numero dei simboli iconici e le parole utilizzati per descrivere l'espressività in musica è

molto vasto: tenuto, staccato, legato, trillo, corona, ecc. Oltretutto molti strumenti, in particolare quelli della famiglia degli archi, posseggono un proprio vocabolario specifico di termini tecnico-espressivi: pizzicato, sul ponticello, flautando, ecc. È da notare che questi termini possono essere inclusi in un testo OMN e, se il vostro campionatore dispone di una libreria di effetti per gli archi, queste istruzioni di espressività possono essere ascoltate direttamente.

```
(q c4 mp trem)
```

Infine c'è la SIMULTANEITÀ che rende possibile assemblare assieme gli attributi e questo si può fare semplicemente col segno '+'.

```
(q c4 mp trem+fermata)
```

(fine della citazione da *Introduzione al Linguaggio OMN*)

1.3. Operazioni sulle liste

Opusmodus opera in ambiente LISP. In ambiente LISP la manipolazione delle liste è un concetto basilare, (LISP significa **LIS**te **P**rocessate). Poiché tutti gli elementi del linguaggio OMN sono codificati in liste ogni manipolazione di liste rientra nel concetto stesso del sistema. Per manipolare le liste ci sono funzioni cosiddette primitive (fondamentali) del LISP, che appartengono alla sintassi basilare di questo linguaggio di programmazione: **`car`**, **`cdr`**, **`caar`**, **`cddr`**, **`eval`**, **`apply`**, **`append`**, **`butlast`**, **`mapcar`**, **`max`**, **`min`**, **`reverse`**, ecc.

Accanto alle primitive del LISP ci sono molte altre funzioni di Opusmodus dedicate alle manipolazioni delle liste, funzioni che agiscono sui parametri specifici del linguaggio OMN: altezze, lunghezze, velocità e attributi sono tutti definiti come liste.

Le liste sono dappertutto!

Se non sai manipolare le liste, non puoi fare musica con Opusmodus!

1.4. Calcolo combinatorio

Opusmodus dispone di strumenti molto utili per lavorare con il calcolo combinatorio, quali le combinazioni e le permutazioni.

Nell'esempio successivo, la funzione **`combination`** fornisce l'elenco con tutte le combinazioni di `3` elementi non ripetitive degli elementi di un universo di **`6`** elementi (`0 1 2 3 4 5`):

```
(combination 3 '(0 1 2 3 4 5))
=> (0 1 2) (0 1 3) (0 1 4) (0 1 5) (0 2 3) (0 2 4) (0 2 5)
   (0 3 4) (0 3 5) (0 4 5) (1 2 3) (1 2 4) (1 2 5) (1 3 4)
   (1 3 5) (1 4 5) (2 3 4) (2 3 5) (2 4 5) (3 4 5) (5 0 1)
   (5 0 2) (5 0 3) (5 0 4) (5 1 2) (5 1 3) (5 1 4) (5 2 3)
   (5 2 4) (5 3 4) (4 5 0) (4 5 1) (4 5 2) (4 5 3) (4 0 1)
   (4 0 2) (4 0 3) (4 1 2) (4 1 3) (4 2 3) (3 4 0) (3 4 1)
   (3 4 2) (3 5 0) (3 5 1) (3 5 2) (3 0 1) (3 0 2) (3 1 2)
   (2 3 0) (2 3 1) (2 4 0) (2 4 1) (2 5 0) (2 5 1) (2 0 1)
   (1 2 0) (1 3 0) (1 4 0) (1 5 0)
```

Utilizzando la funzione **`get-count`** con l'opzione `:sum` uguale **`t`**, otteniamo il totale delle combinazioni possibili della lista, pari a `60`:

```
(get-count (combination 3 '(0 1 2 3 4 5)) :sum t)
=> 60
```

Possiamo ottenere le permutazioni di un insieme come `(0 1 2)` utilizzando la funzione **`permute`**:

```
(permute '(0 1 2))
=> ((0 1 2) (0 2 1) (1 0 2) (1 2 0) (2 0 1) (2 1 0))
```

1.5. Numeri Forte

Nel 1973, Allen Forte (1926-2014) costruì la summa di tutte le strutture armoniche disponibili nel sistema temperato (FORTE, 1973, p.179-181). La classificazione di tutti gli accordi esistenti nel sistema temperato è contenuta in sole due pagine. L'enorme numero di permutazioni possibile è

sioni), in 352 qualità uniche (tenendo conto della forma originale e della forma invertita) e 200 diversi vettori d'intervallo, venendosi così a costituire un totale di 4096 raccolte di altezze (considerando le trasposizioni degli insiemi indicati). Forte ha anche messo in ordine gli insiemi per cardinalità, cioè per numero di elementi distinti in un insieme definito. Per ogni studioso che ha svolto ricerche approfondite sulle combinazioni di altezze, questa nomenclatura tassonomica ha contribuito a mantenere un ordine per cardinalità distinte, dalle triadi agli insiemi più ampi, fino alla totalità dei dodici suoni. Grazie alla nomenclatura tassonomica degli insiemi di altezze sviluppata da Forte è stato possibile capire le proprietà degli accordi (intesi come classi di insiemi) e la loro equivalenza per inversione, il concetto di forma prima ed infine valore di contenuto dell'intervallo. La teoria di Forte oggi rappresenta lo standard nella teoria degli insiemi musicali, comunemente usata da molti teorici della musica e compositori. Questa teoria è totalmente inclusa fra gli strumenti di Opusmodus.

Accanto a Forte, Elliott Carter (1908-2012), nel suo Harmony Book (2002), ha svolto uno studio esaustivo ed equivalente sulle combinazioni tra gli insiemi e i loro complementi. Carter ha usato una propria convenzione di denominazione degli insiemi, ma alla fine ha deciso di collegare la sua particolare nomenclatura e quella usata da Forte (CARTER, 2002, pp. 23-26).

In Opusmodus ci sono molte funzioni connesse alla teoria degli insiemi ed è possibile utilizzare i numeri di Forte per operazioni sulle classi di insiemi. Ecco alcuni esempi della funzione **pcs**:

```
(pcs '6-Z10)
=> (0 1 3 4 5 7)

(pcs '9-7 :pitch)
=> (c4 cs4 d4 eb4 e4 f4 g4 gs4 bb4)

(pcs '9-7 :vector)
=> (6 7 7 6 7 3)
```

particolarmente interessante è la funzione **pcs-sub-sets** per trovare eventuali sotto-insiemi di un insieme:

```
(pcs-sub-sets 4 (pcs '7-35) :forte)
=> (4-8 4-10 4-11 4-11b 4-13 4-13b 4-14 4-14b 4-16 4-16b 4-20
    4-21 4-22 4-22b 4-23 4-26 4-27 4-27b 4-z29 4-z29b)
```

seguente, la funzione **pcs-super-sets** mostra tutti gli insiemi che contengono una triade maggiore come sottoinsieme, cioè i super-insiemi della triade maggiore (numero Forte 3-11b).

```
(pcs-super-sets 7 (pcs '3-11b) :forte)
=> (7-2 7-2b 7-3 7-3b 7-4 7-4b 7-5b 7-6 7-6b 7-7 7-7b 7-8 7-9
    7-9b 7-10 7-10b 7-11 7-11b 7-z12 7-13 7-13b 7-14 7-14b 7-15
    7-16 7-16b 7-z17 7-z18 7-z18b 7-19 7-19b 7-20 7-20b 7-21
    7-21b 7-22 7-23 7-23b 7-24 7-24b 7-25 7-25b 7-26 7-26b 7-27
    7-27b 7-28 7-28b 7-29 7-29b 7-30 7-30b 7-31 7-31b 7-32 7-32b
    7-33 7-34 7-35 7-z36 7-z36b 7-z37 7-z38 7-z38b)
```