On symbolic representations and transformations of sound

The theory of sound-types

Carmine-Emanuele Cella

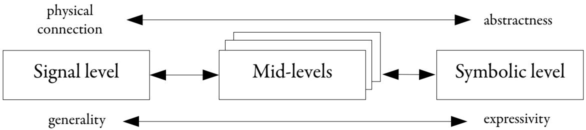
Università di Bologna/IRCAM Paris

December 1, 2011

ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

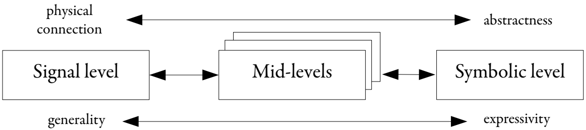
ircam Centre Pompidou

Different representations (1)



- Music, in its final stage of *performance*, can be described in many ways (time-varying signal, symbolic system, etc.).

Different representations (1)



- Music, in its final stage of *performance*, can be described in many ways (time-varying signal, symbolic system, etc.).
- Each approach selects a particular degree of abstraction: the *signal level*, the *symbolic level*, a fixed mixture of both (*mid-levels*).

Different representations (2)

- The signal level is efficient and invertible but has a low degree of abstraction.

Different representations (2)

- The signal level is efficient and invertible but has a low degree of abstraction.
- The symbolic level can define complex relationships between objects but is not invertible and is not *physical*.

Different representations (2)

- The signal level is efficient and invertible but has a low degree of abstraction.
- The symbolic level can define complex relationships between objects but is not invertible and is not *physical*.
- Mid-levels are based on perceptual criteria related to hearing and are in between lower and higher levels.

Different representations (2)

- The signal level is efficient and invertible but has a low degree of abstraction.
- The symbolic level can define complex relationships between objects but is not invertible and is not *physical*.
- Mid-levels are based on perceptual criteria related to hearing and are in between lower and higher levels.
- They have a *fixed degree of abstraction*.

Different representations (2)

- The signal level is efficient and invertible but has a low degree of abstraction.
- The symbolic level can define complex relationships between objects but is not invertible and is not *physical*.
- Mid-levels are based on perceptual criteria related to hearing and are in between lower and higher levels.
- They have a *fixed degree of abstraction*.
- They impose *their own* concepts onto the signal.

Variable abstraction representation (1)

This research aims at creating a representation method for music (the *theory of sound-types*) that fulfills *by-design* the following requirements:

- *Signal-dependent semantics*: the involved concepts of the representation should be inferred from the signal.

Variable abstraction representation (1)

This research aims at creating a representation method for music (the *theory of sound-types*) that fulfills *by-design* the following requirements:

- *Signal-dependent semantics*: the involved concepts of the representation should be inferred from the signal.
- *Scalability*: it should be possible to change the *degree of abstraction* in the representation, ranging from the signal level to the symbolic level.

Variable abstraction representation (2)

- *Weak invertibility*: the representation method should be able to generate the represented signal; the generated signal must not be *waveform*-identical to the original one.

Variable abstraction representation (2)

- *Weak invertibility*: the representation method should be able to generate the represented signal; the generated signal must not be *waveform*-identical to the original one.
- *Generativity*: is the possibility to generate sounds *other* than the original one, according to some parameters in the domain of the representation.

Sound-types: basic ideas (1)

1. A theory to represent sounds by means of *types* and *rules* inferred by some low-level descriptions of signals and subsequent learning stages.

Sound-types: basic ideas (1)

1. A theory to represent sounds by means of *types* and *rules* inferred by some low-level descriptions of signals and subsequent learning stages.
2. It takes inspiration from the *simple type theory* and from λ-calculus.

Sound-types: basic ideas (1)

1. A theory to represent sounds by means of *types* and *rules* inferred by some low-level descriptions of signals and subsequent learning stages.
2. It takes inspiration from the *simple type theory* and from λ-calculus.
3. The types represent classes of equivalences for sounds, while the rules represent transition probabilities that a type is followed by another type.

Sound-types: basic ideas (2)

The decomposition of a signal $x[n]$ into expansion functions is a linear combination of the form:

$$\overset{n}{\vec{x}} = \sum_{k=1}^{K} \alpha_k \, \overset{n}{\vec{g}}_k \ .$$

The coefficients α_k are derived from the analysis stage, while the functions $g_k[n]$ can be determined by the analysis stage or fixed beforehand.

Sound-types: basic ideas (3)

Mathematically, this theory tries to *translate* the original equation:

$$
\begin{aligned}
x[n] &= \sum_{k=1}^{K} \alpha_k g_k[n] \\
&= \alpha_1 g_1[n] + \ldots + \alpha_k g_k[n] \\
&= \beta_1 f_1[n] + \ldots + \beta_j f_j[n] \\
&\vdots \\
&= \omega_1 h_1[n] + \ldots + \omega_t h_t[n]
\end{aligned}
$$

where $\alpha, \beta, \ldots, \omega$ are any kind of weighting coefficients, $g_k, f_j, \ldots, h_t$ are variables belonging to different *types*. All the theory presented above can be realized by defining the **sound-types transform** (STT).

The sound-types transform (1)

- Given a signal $\overset{N}{\vec{x}}$ of length N-samples and a window $\overset{n}{\vec{h}}$ of length n-samples, it is possible to define an **atom** as a windowed chunk of the signal of length n-samples:

$$\overset{n}{\vec{a}} = \overset{n}{\vec{h}} \cdot \overset{n}{\vec{x}} .$$

The sound-types transform (1)

- Given a signal $\overset{N}{\vec{x}}$ of length N-samples and a window $\overset{n}{\vec{h}}$ of length n-samples, it is possible to define an **atom** as a windowed chunk of the signal of length n-samples:

$$\overset{n}{\vec{a}} = \overset{n}{\vec{h}} \cdot \overset{n}{\vec{x}} \; .$$

- A **sound-cluster** as a set of atoms that *lie* in a defined area of a feature space (ie. that share a *similar* set of features):

$$\overset{k_r}{\vec{c}_r} = \{\overset{n}{\vec{a}}_{r,1}, \ldots, \overset{n}{\vec{a}}_{r,k_r}\}.$$

The content of $\overset{k_r}{\vec{c}_r}$ is given by a statistical analysis applied on the feature space.

The sound-types transform (2)

- A **model** $\mathcal{M}_{\overset{N}{\vec{x}}}$ of the signal $\overset{N}{\vec{x}}$ is the defined as the set of the clusters discovered on it:

$$\mathcal{M}_{\overset{N}{\vec{x}}} = \{\overset{k_1}{\vec{c}_1}, \ldots, \overset{k_r}{\vec{c}_r}\}.$$

The sound-types transform (2)

- A **model** $\mathcal{M}_{\overset{N}{\vec{x}}}$ of the signal $\overset{N}{\vec{x}}$ is the defined as the set of the clusters discovered on it:

$$\mathcal{M}_{\overset{N}{\vec{x}}} = \{\overset{k_1}{\vec{c}_1}, \ldots, \overset{k_r}{\vec{c}_r}\}.$$

- The cardinality $|\mathcal{M}_{\overset{N}{\vec{x}}}|$ of the model is also called the **abstraction level** of the analisys; since the number atoms is N/t it is evident that $1 \leq |\mathcal{M}_{\overset{N}{\vec{x}}}| \leq N/t$ with higher abstraction being 1 and lower abstraction being N/t (where t is the overlapping factor used to create atoms).

The sound-types transform (3)

- A sound-cluster has an associate **sound-type** $\overset{n}{\vec{\tau}_r}$, defined as the weighted sum of all the atoms in the sound-cluster where the weights $\overset{k_r}{\vec{\omega}_r}$ are the distances of each atom to the center of the cluster:

$$\overset{n}{\vec{\tau}_r} = \sum_{j=1}^{k_r} \overset{n}{\vec{a}_{r,j}} \cdot \omega_{r,j}$$

with $\omega_{r,j} \in \overset{k_r}{\vec{\omega}_r}$.

The sound-types transform (3)

- A sound-cluster has an associate **sound-type** $\overset{n}{\vec{\tau}}_r$, defined as the weighted sum of all the atoms in the sound-cluster where the weights $\overset{k_r}{\vec{\omega}}_r$ are the distances of each atom to the center of the cluster:

$$\overset{n}{\vec{\tau}}_r = \sum_{j=1}^{k_r} \overset{n}{\vec{a}}_{r,j} \cdot \omega_{r,j}$$

with $\omega_{r,j} \in \overset{k_r}{\vec{\omega}}_r$.

- The set of sound-types in the signal $\overset{N}{\vec{x}}$ is called **dictionary**:

$$\mathcal{D}_{\overset{N}{\vec{x}}} = \{\overset{n}{\vec{\tau}}_1, \ldots, \overset{n}{\vec{\tau}}_r\}.$$

The sound-types transform (4)

- Finally, it is possible to define the *sound-types transform* as a function of time and frequency obtained by multiplying the sound-types in a given dictionary with complex sinusoids:

$$\boxed{\overset{N}{\vec{\Phi}}_{\overset{n}{\vec{k}}} = \sum_{i=0}^{N/t} \overset{n}{\vec{\tau}}_{r,p} \cdot e^{-j \cdot \frac{2 \cdot \pi}{n} \cdot \overset{n}{\vec{k}}}}$$

where $\overset{n}{\vec{k}} = \{f_1, \ldots, f_n\}$ is a vector of frequencies.

The sound-types transform (5)

- The extreme case for $|\mathcal{M}| = N/t$ is interesting: for that abstraction level, each sound-cluster is a singleton made of a single atom and consequently each sound-type reduces to that single atom scaled in amplitude:

$$|\mathcal{M}| = N/t \implies \overset{1}{\vec{c}_r} = \{\overset{n}{\vec{a}_1}\} \implies \overset{n}{\vec{\tau}_r} = \overset{n}{\vec{a}_r} \cdot \omega_{r,1}.$$

The sound-types transform (5)

- The extreme case for $|\mathcal{M}| = N/t$ is interesting: for that abstraction level, each sound-cluster is a singleton made of a single atom and consequently each sound-type reduces to that single atom scaled in amplitude:

$$|\mathcal{M}| = N/t \implies \overset{1}{\vec{c}_r} = \{\overset{n}{\vec{a}_1}\} \implies \overset{n}{\vec{\tau}_r} = \overset{n}{\vec{a}_r} \cdot \omega_{r,1}.$$

- This leads to the important consequence that STT is a **generalization** of STFT:

$$\boxed{\overset{n}{\vec{\tau}_r} = \overset{n}{\vec{a}_r} = \overset{n}{\vec{h}} \cdot \overset{n}{\vec{x}} \implies \sum_{i=0}^{N/t} \overset{n}{\vec{\tau}_{r,p}} \cdot e^{-j \cdot \frac{2\cdot\pi}{n} \cdot \overset{n}{\vec{k}}} = \sum_{i=0}^{N/t} \overset{n}{\vec{h}} \cdot \overset{n}{\vec{x}_{i \cdot t}} \cdot e^{-j \cdot \frac{2\cdot\pi}{n} \cdot \overset{n}{\vec{k}}}}$$

with p defined as above.

Computational criteria

- Some criteria are needed to define types and rules

Computational criteria

- Some criteria are needed to define types and rules
- They can be obtained by means of a twofold process:
  - **types inference**: first, the types involved in the representations are discovered by looking for common entities in a sound
  - **rules inference**: a second stage is needed to discover the rules that link one type to another by means of a sequential analysis

Creation of sound-types (1)

A possible realization is based on low-level descriptors plus classificaiton for types inference and Markov models for rules inference:

- **(atomic decomposition)**: subdivide a sound into small grains of approximately 40 ms called *atoms* or *0-types* overlapping in time and frequency

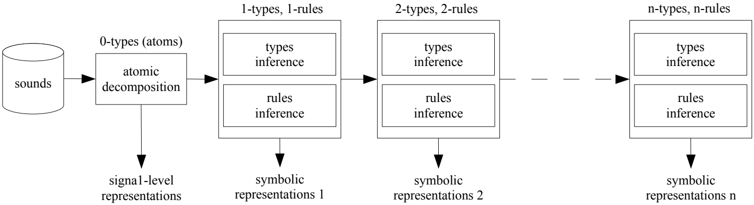
Creation of sound-types (1)

A possible realization is based on low-level descriptors plus classificaiton for types inference and Markov models for rules inference:

- **(atomic decomposition)**: subdivide a sound into small grains of approximately 40 ms called *atoms* or *0-types* overlapping in time and frequency
- **(1-types inference)**: compute a set of low-level descriptors on the atoms obtained in the previous step, project the descriptors in a multi-dimensional space and compute the *clusters*; each cluster will represent a *1-type*
- **(1-rules inference)**: implement a Markov model to describe the sequences of types present in the analysed sound (*1-rules*)

Creation of sound-types (2)

- **(n-types inference)**: compute a set of low-level descriptors on the whole sequences found in the previous step; project again the descriptors and compute the clusters: each cluster will represent a *n-type*
- **(n-rules inference)**: repeat the Markov model until there are no more sequences (*n-rules*).



Sound-types: properties (1)

1. The number of iterations represents the degrees of abstraction

Sound-types: properties (1)

1. The number of iterations represents the degrees of abstraction
2. Discovered types are defined in the time-frequency plane and have increasing time scale

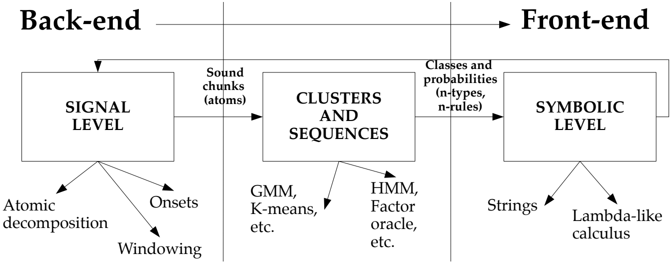
Sound-types: properties (1)

1. The number of iterations represents the degrees of abstraction
2. Discovered types are defined in the time-frequency plane and have increasing time scale
3. The higher the level of a type, the more will be expressive (the less generic)

Sound-types: properties (2)

- *Signal-dependent semantics*: the atoms and the $+$ relation are derived from the signal
- *Scalability*: the possibility to scale over abstraction is implicit to theories of types; we showed how it is possible to translate a representation to another by changing the involved elements and operators
- *Weak invertibility* and *generativity*: there are many possibilities to create a signal back from sound types: pick up randomly an element of each cluster used, pick up the element closest to the center of the cluster or to generate a weighted sum of all the elements of a cluster, etc.

A generalized framework



Clusters (1)

A tool called **Clusters** has been implemented in C++ to analyze sounds and produce quasi-symbolic representations:

- Low-level features analysis of sounds (spectral and temporal) with dimensionality reduction (PCA).

Clusters (1)

A tool called **Clusters** has been implemented in C++ to analyze sounds and produce quasi-symbolic representations:

- Low-level features analysis of sounds (spectral and temporal) with dimensionality reduction (PCA).
- Clusters estimation with K-means and GMM (types).

Clusters (1)

A tool called **Clusters** has been implemented in C++ to analyze sounds and produce quasi-symbolic representations:

- Low-level features analysis of sounds (spectral and temporal) with dimensionality reduction (PCA).
- Clusters estimation with K-means and GMM (types).
- Auto-estimation of number of clusters (gap statistic, BIC).

Clusters (1)

A tool called **Clusters** has been implemented in C++ to analyze sounds and produce quasi-symbolic representations:

- Low-level features analysis of sounds (spectral and temporal) with dimensionality reduction (PCA).
- Clusters estimation with K-means and GMM (types).
- Auto-estimation of number of clusters (gap statistic, BIC).
- Whitin-cluster dispersion for outliers detection.

Clusters (1)

A tool called **Clusters** has been implemented in C++ to analyze sounds and produce quasi-symbolic representations:

- Low-level features analysis of sounds (spectral and temporal) with dimensionality reduction (PCA).
- Clusters estimation with K-means and GMM (types).
- Auto-estimation of number of clusters (gap statistic, BIC).
- Whitin-cluster dispersion for outliers detection.
- Inverse transformations (original sound reconstruction, probabilistic generation and hybridization).

Clusters (1)

A tool called **Clusters** has been implemented in C++ to analyze sounds and produce quasi-symbolic representations:

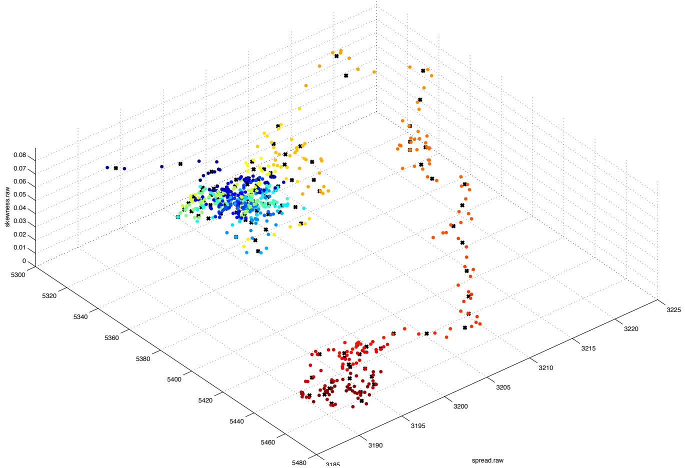
- Low-level features analysis of sounds (spectral and temporal) with dimensionality reduction (PCA).
- Clusters estimation with K-means and GMM (types).
- Auto-estimation of number of clusters (gap statistic, BIC).
- Whitin-cluster dispersion for outliers detection.
- Inverse transformations (original sound reconstruction, probabilistic generation and hybridization).
- Transition probabilities with a Markov model.

Clusters (1)

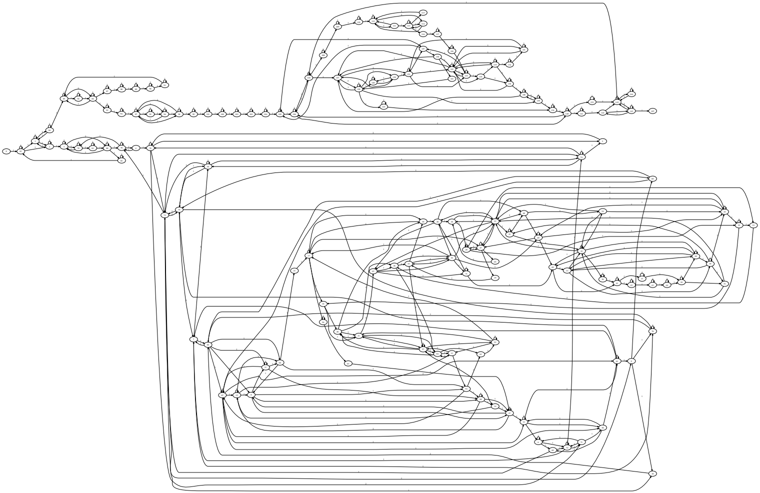
A tool called **Clusters** has been implemented in C++ to analyze sounds and produce quasi-symbolic representations:

- Low-level features analysis of sounds (spectral and temporal) with dimensionality reduction (PCA).
- Clusters estimation with K-means and GMM (types).
- Auto-estimation of number of clusters (gap statistic, BIC).
- Whitin-cluster dispersion for outliers detection.
- Inverse transformations (original sound reconstruction, probabilistic generation and hybridization).
- Transition probabilities with a Markov model.
- Symbolic representation with a simple string.

Clusters (2): types representation (3D case)



Clusters (3): rules representation



Clusters (4): inverse transformation examples 1

Reconstruction

- Original sample 1 (rock band).

Clusters (4): inverse transformation examples 1

Reconstruction

- Original sample 1 (rock band).
- 170 types (10%), 11 mixed features, weighted, taxicab.

Clusters (4): inverse transformation examples 1

Reconstruction

- Original sample 1 (rock band).
- 170 types (10%), 11 mixed features, weighted, taxicab.
- 860 types (50%), 4 spectral features, weighted, taxicab.

Clusters (4): inverse transformation examples 1

Reconstruction

- Original sample 1 (rock band).
- 170 types (10%), 11 mixed features, weighted, taxicab.
- 860 types (50%), 4 spectral features, weighted, taxicab.
- 1600 types (95%), 11 spectral features, weighted, cosim.

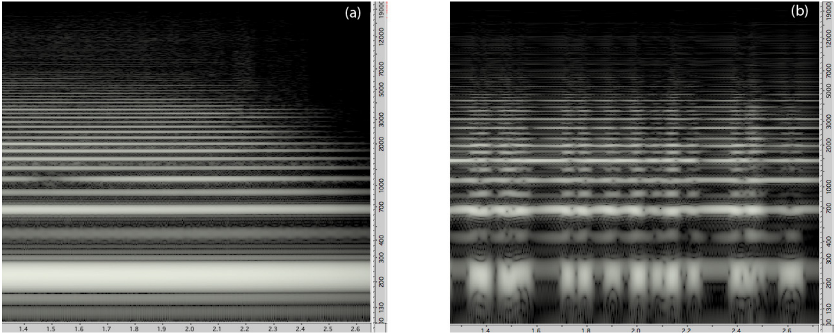
Clusters (4): inverse transformation examples 1

Reconstruction

- Original sample 1 (rock band).
- 170 types (10%), 11 mixed features, weighted, taxicab.
- 860 types (50%), 4 spectral features, weighted, taxicab.
- 1600 types (95%), 11 spectral features, weighted, cosim.
- Original sample 2 (orchestra) and 100% reconstruction.

Clusters (5): bad signal reconstruction

Under a given ratio between types and atoms, the signals are not reconstructed correctly. The following example shows such a situation:



Clusters (6): inverse transformation examples 2

Transformations and generations

- Original sample 3 (lute) and two times slower resynthesis.

Clusters (6): inverse transformation examples 2

Transformations and generations

- Original sample 3 (lute) and two times slower resynthesis.
- Original sample 4 (voice) and one fifth up resynthesis.

Clusters (6): inverse transformation examples 2

Transformations and generations

- Original sample 3 (lute) and two times slower resynthesis.
- Original sample 4 (voice) and one fifth up resynthesis.
- Original sample 5 (bass) and its probabilistic generation.

Clusters (6): inverse transformation examples 2

Transformations and generations

- Original sample 3 (lute) and two times slower resynthesis.
- Original sample 4 (voice) and one fifth up resynthesis.
- Original sample 5 (bass) and its probabilistic generation.
- Original sample 6 (piano) and its probabilistic generation.

Clusters (6): inverse transformation examples 2

Transformations and generations

- Original sample 3 (lute) and two times slower resynthesis.
- Original sample 4 (voice) and one fifth up resynthesis.
- Original sample 5 (bass) and its probabilistic generation.
- Original sample 6 (piano) and its probabilistic generation.
- A probabilistic generation of sample 2 (orchestra).

Clusters (6): inverse transformation examples 2

Transformations and generations

- Original sample 3 (lute) and two times slower resynthesis.
- Original sample 4 (voice) and one fifth up resynthesis.
- Original sample 5 (bass) and its probabilistic generation.
- Original sample 6 (piano) and its probabilistic generation.
- A probabilistic generation of sample 2 (orchestra).
- Hybridization between sample 2 (orchestra) and sample 6 (piano) [mixing].

Clusters (6): inverse transformation examples 2

Transformations and generations

- Original sample 3 (lute) and two times slower resynthesis.
- Original sample 4 (voice) and one fifth up resynthesis.
- Original sample 5 (bass) and its probabilistic generation.
- Original sample 6 (piano) and its probabilistic generation.
- A probabilistic generation of sample 2 (orchestra).
- Hybridization between sample 2 (orchestra) and sample 6 (piano) [mixing].
- Hybridization between sample 2 (orchestra) and sample 6 (piano) [merging].

Clusters (7): current applications I/II

- **Symbolic description**: analysing the created string of labels, it is possible to acquire information of *salient* properties of the sound and represent it in a meaningful way:

$$
\begin{aligned}
x[n] &= 0 + 1 + 2 + 3 + \ldots + \ldots + \ldots + \ldots + n \\
&= \alpha_1 + \ldots + \ldots + \ldots + \alpha_k \\
&= \beta_1 + \ldots + \ldots + \beta_j \\
&\vdots \\
&= \omega_1 + \ldots + \omega_t
\end{aligned}
$$

where $n > k > j > t$ and the types increase their amount of information from one level to another.

Clusters (8): current applications II/II

- **Audio compression**: while not being the main purpose of the approach, it is possible to compress a sound by a given ratio.

Clusters (8): current applications II/II

- **Audio compression**: while not being the main purpose of the approach, it is possible to compress a sound by a given ratio.
- **Time and frequency transformations**: it is possible to perform various transformations such as time-stretch and pitch-shift.

Clusters (8): current applications II/II

- **Audio compression**: while not being the main purpose of the approach, it is possible to compress a sound by a given ratio.
- **Time and frequency transformations**: it is possible to perform various transformations such as time-stretch and pitch-shift.
- **Probabilistic generation**: using the types and their probabilities it's possible to generate sounds *related* to the original, but different.

Clusters (8): current applications II/II

- **Audio compression**: while not being the main purpose of the approach, it is possible to compress a sound by a given ratio.
- **Time and frequency transformations**: it is possible to perform various transformations such as time-stretch and pitch-shift.
- **Probabilistic generation**: using the types and their probabilities it's possible to generate sounds *related* to the original, but different.
- **Hybridization (still experimental)**: using the types of one sound and the probabilities of another one it's possible to create hybrid sounds.

The reduction effect

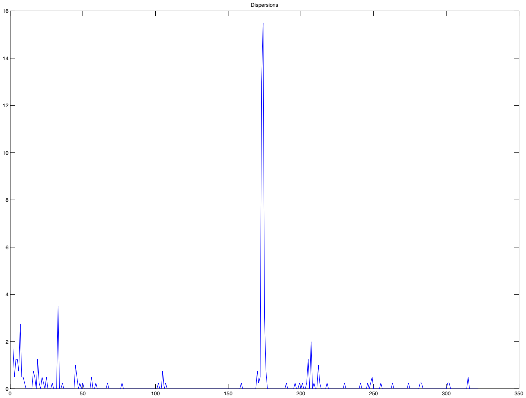
- The smaller the number of clusters (meaning that we reduce the number of clusters, grouping more entities in the same sound-type) the better will be the representation but the worst will be the sound resynthesis.

The reduction effect

- The smaller the number of clusters (meaning that we reduce the number of clusters, grouping more entities in the same sound-type) the better will be the representation but the worst will be the sound resynthesis.
- Sound quality is directly linked to the number of clusters, but if we augment this number we loose the possibility of having a compact representation analysis.

Measure for objective evaluation

Is the *within-cluster dispersion* is a measure for quality?



Future applications (1)

- **Selective transformations**: it should be possible to perform various transformations only on some *selected* types (let's suppose that some discovered types represent the vowels of a singing voice: it should be then possible to operate only on those types selectively).

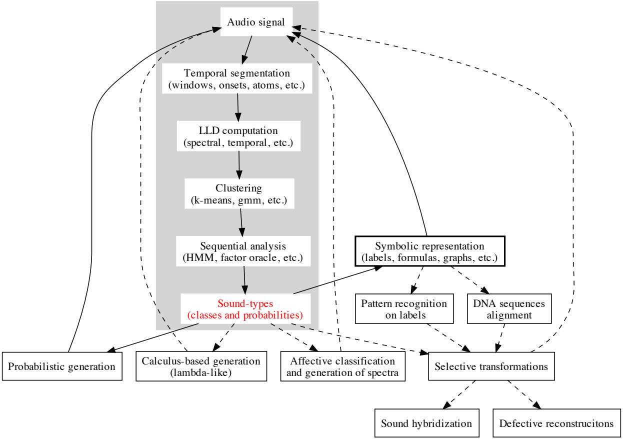
Future applications (1)

- **Selective transformations**: it should be possible to perform various transformations only on some *selected* types (let's suppose that some discovered types represent the vowels of a singing voice: it should be then possible to operate only on those types selectively).
- **Pattern recognition on the representation**: the tool represents a sound using a string of labels; many algorithms could be applied on that string to discover patterns; for example if at a given level the representation is $\alpha\beta\beta\ldots\gamma\delta\delta$, it could be possible to rewrite this with a function of two variables such as $\phi(x, y) = xyy$ thus having $\alpha\beta\beta\ldots\gamma\delta\delta = \phi(\alpha, \beta)\ldots\phi(\gamma, \delta)$.

Future applications (2)

- **Affective classification and generation of spectra**: when a sound has been represented in term of sound-types and probabilities, a supervised labelling could be applied on the discovered elements in order to classify them by means an affective model: some types could be called, for example, *rough* or *sad*. It could be possible, then, to ask the machine to generate similar sounds by means of the discovered probabilities.

Outline of some possible expansions



Any questions?

- C. E. Cella, **Towards a Symbolic Approach to Sound Analysis**, MCM 2009, Yale University - New Haven (CT), Springer.
- C. E. Cella, **Sound-types: a new framework for sound analysis and synthesis**, ICMC 2011, Huddersfield (UK).
- C. E. Cella, **On symbolic representations of music**, PhD dissertation, 2011, University of Bologna.