D1.1
Specification of the Representation Formalism and of Constraints and Requirements of Reasoning

Authors	Tarek R. Besold, Emilios Cambouropoulos, Danny Arlen de Jesus Gomez Ramirez, Maximos Kaliakatsos-Papakostas, Kai-Uwe Kühnberger, Ewen Maclean, Enric Plaza, Alan Smaill
Reviewers	Marco Schorlemmer

Grant agreement no.	611553
Project acronym	COINVENT - Concept Invention Theory
Date	October 15, 2014
Distribution	PU

The project COINVENT acknowledges the financial support of the Future and Emerging Technologies (FET) programme within the Seventh Framework Programme for Research of the European Commission, under FET-Open Grant number 611553.

Abstract

This deliverable specifies the knowledge representation formalisms used in the mathematics and music domain of the COINVENT project. Furthermore, the reasoning processes for computing the generic space and concept blends are sketched. For mathematics a many-sorted first-order logical language is used to represent mathematical theories. Concept blending operates on these logically specified theories and computes candidates for a generic space as well as candidates for blend spaces. In the music domain, different representation formalisms are specified, each of them of particular interest for different aspects of music. The discussed formalisms are basic chord representations, feature structure representations (of various forms), and hidden Markov models. The representation and the computation of blend spaces are instantiated using a variety of examples.

Keyword list: **Knowledge Representation**, **Reasoning**, **Mathematics**, **Music**

Executive Summary

Concept blending in the sense adopted in the COINVENT project takes two input spaces and attempts to compute a generic space and a blend space, i.e. the latter being a new and independent conceptual space containing a mixture of conceptual information from both input domains. The degree to which such computations are realizable in a formal system depends on the representation of the involved conceptual spaces. This deliverable addresses the issue which representation formalisms can be used for concept blending in the two considered domains mathematics and music.

The two considered domains are rather different from each other. Mathematical objects have clear syntactic descriptions and a precise meaning, there is a clear level of description of mathematical theories, the possibilities to formalize mathematical theories is strongly restricted, and hard constraints can help to distinguish a correct mathematical theory from an incorrect mathematical theory (e.g. consistency), whereas these properties are in general not applicable to the music domain. Musical objects do not have a clear (standard) meaning, there are many levels of abstraction that can be used to describe music (e.g. physical realization, MIDI level, harmonization, motifs, structure of pieces etc.), and pieces of music can be described in many different languages and frameworks. Furthermore, it is difficult to say that hard constraints (like consistency) apply to music which could help to distinguish “correct” pieces of music from “incorrect” ones. As a consequence of these differences the representation formalisms used to work in the intended domains differ significantly from each other in the COINVENT project.

In mathematics, a many-sorted first-order logical language is used to axiomatize the input domains. The generalization of the input domains is formalized in a mild form of higher-order logic (triggered by restricted higher-order anti-unification), and the blend space is again formalized in a many-sorted first-order logical language. The generalization process is computed using Heuristic-Driven Theory Projection (HDTP), whereas the blending process is a colimit construction performed by the HETS framework.

In music, several representation formalisms exist besides each other. On the one hand, there is the need for flexible formalisms dependent on the specific aspect in music that is intended to be modeled, on the other hand, some applications in the music domain require the modeling of uncertainty using some form of probabilistic reasoning. The COINVENT project uses General Chord Type representation (GCT), feature structure representations, and Constrained Hidden Markov Models. It turns out that such representations have their strengths in different applications, e.g. whereas feature structure representations fit to tasks where blending on the chord level is necessary, GCT representations fit to idiom-independent representation, in particular, important in the context of extracting harmonic information from score.

The worked examples specified in this document (and further examples published during the first year of this project) as well as the rudimentary implementations of the system show that the representation formalisms described in this deliverable are very good candidates for the different tasks. A thorough testing and evaluation on more complex examples needs to be conducted during the next project phase.

Contents

1 Introduction

This report summarizes the specification of the representation formalisms used in the domains of mathematics and music of the COINVENT project. Furthermore, it is reported which constraints and requirements for reasoning tasks occur in the framework, by specifying the reasoning processes involved in concept invention tasks. As usual in artificial intelligence, different domains require different representation formalisms, for example, with respect to expressive strength, the possibility to express uncertainty and vagueness, or the mechanisms for drawing inferences, just to mention some of them. Consequently it is very common in AI that different languages are used for applications in different domains of interest [6].

For mathematics and music in the context of concept invention we experienced a very similar situation: whereas large parts of mathematics can be coded in a many-sorted first-order logical language, i.e. the language is rather expressive, crisp, and uses classical logical inference mechanisms, in music a variety of formalisms can be used for the different rather specific musical aspects (e.g. feature structures, Constrained Hidden Markov Models) that are often weaker concerning the expressive strength, but have sometimes additional properties, like the ability to express uncertainty in the case of Constrained Hidden Markov Models. It is important to notice that the sketched pluralistic perspective of representation formalisms does not restrict the possibility to implement concept blending procedures within these formalisms.

The report has the following structure: Section 2 discusses representation aspects of mathematical theories in the light of the computation of concept blends of such theories. In particular, many-sorted first-order logic is specified in Subsection 2.1. Subsection 2.2 gives a general approach of how concept blending in mathematics can be achieved, namely in terms of a two-step process that first, computes a generalized theory (generic space) given two input theories and second, computes a blend space. Subsection 2.3 describes in detail a non-trivial mathematical example of concept blends, namely the complex numbers. Section 3 has a rather similar structure to Section 2, but this time with respect to the domain of music. In Subsection 3.1, representation formalisms for music are introduced (General Chord Type representations, feature structures, and Constrained Hidden Markov Models), whereas in Subsection 3.2 the overall architecture and information extraction from score is discussed. Subsection 3.3 gives a broad overview illustrating how different aspects of music can be represented and how concrete examples of blending chords, blending phrases of pieces, blending chord progressions etc. can be analyzed and computed. Section 4 refers to the Deliverable D8.1 concerning the implementation of the computational system. Finally, Section 5 concludes this report.

2 Representation of and Reasoning in Mathematical Theories to Create New Concepts

2.1 Abstract Representation of Mathematical Theories

The working mathematician specifies mathematical theories almost in every branch of mathematics in terms of axiom systems. The language used for such axiomatiziations (as for underlying definitions of mathematical concepts) is usually a mixture of natural language and a formal (often logical) language. From a cognitive perspective this mixture of two types of languages is rather plausible due to the fact that the combination of the two types of languages results in formalizations that are easier to grasp for cognitive agents in comparison to representations that are exclusively formalized in a logical language.[1]

For computational purposes the usage of a formal language is appropriate. For example, to express the classical continuity condition of a real-valued function we need to write, as one possible representation, the following expression:

$$\forall \varepsilon \in \mathscr{R}(\varepsilon > 0 \rightarrow \exists \delta \in \mathscr{R}(\delta > 0 \wedge \forall x \in \mathscr{R} \forall y \in \mathscr{R}(|x-y| < \delta \rightarrow |f(x) - f(y)| < \varepsilon)))$$

Besides the involved set theoretical relations (e.g. the elementhood relation $\in$), logical operators (e.g. logical connectives like $\wedge$), and the objects of the domain of interest (e.g. the real numbers x and y), there is also the need to represent operations on real numbers (e.g. addition $+$) and relations between real numbers (e.g. the $<$-relation). This can be rather straightforwardly expressed in a logical language $\mathscr{L}$ using a signature containing (additionally to logical and set theoretical symbols) constant symbols, function symbols, and relation symbols of the appropriate arity, i.e. in our example $\mathscr{L} = \{0, 1, -, +, |\cdot|, <, >\}$

Mathematical theories within the COINVENT framework are represented using a many-sorted first-order logic language. This is due to the need for a formalism which allows for expressing conveniently properties of structures of different types (i.e., sorts), offering possibilities to partition the universe into different categories and avoiding to treat it as an entirely homogeneous collection of objects. For an example, from the domain of mathematics, consider the use of points, lines, planes, etc. in geometry.

In most cases, many-sorted logics are based on a set of sorts S in order to generalize the notion of a logical signature, allowing to handle the additional information introduced by the sorts. Many-sorted first-order logic then arises from adding sorts to standard first-order logic, resulting in a formalism which basically maintains all the properties of first-order logic (e.g., coming with the same proof theory).[2] Different from first-order logic, in many-sorted first-order logic arguments of function and predicate symbols may have different sorts, and constant and function symbols also have some sort.

Many-sorted first-order logic is sufficiently expressive to study many interesting aspects of mathematics, such as, for instance, questions relating to certain algebraic structures, real closed fields, or relevant parts of calculus. Although not allowing to address all of mathematics, many-sorted first-order logic seems to offer a good trade-off between expressivity and manageability and feasibility of reasoning.

As a concrete example for the application of a many-sorted first-order logic consider the language used for describing domain theories in the Heuristic-Driven Theory Projection (HDTP)

[1]Using exclusively a logical formalization of mathematical theories together with inference rules to gain mathematical theorems is an old attempt in the history of mathematics. The historically important reference in this respect is Whitehead and Russell's Principia Mathematica [41].

[2]When there are only finitely many sorts in a theory, many-sorted first-order logic can completely be reduced to standard first-order logic: For each sort in the many-sorted theory a unary predicate symbol is added to the single-sorted theory, and an axiom stating that these unary predicates partition the domain of discourse is introduced.

$$\begin{tikzcd}
& G \arrow[dl] \arrow[dr] \arrow[dd] & \\
I_1 \arrow[dr] & & I_2 \arrow[dl] \\
& B &
\end{tikzcd}$$

Figure 1: Goguen's version of concept blending (cf. [12, 13])

framework [38], one of the core modules of the COINVENT architecture: HDTP uses an algebraized version of many-sorted first-oder logic, namely many-sorted term algebras, to define the input conceptual domains. A term algebra requires two ingredients, a *signature* and a set of *variables*.

Definition 1 *A many-sorted signature $\Sigma = \langle Sort, Func \rangle$ is a tuple containing a finite set Sort of* sorts*, and a finite set Func of* function symbols*. An n-ary function symbol $f \in Func$ is specified by $f : s_1 \times s_2 \times \cdots \times s_n \to s$, where $s, s_1, \ldots, s_n \in Sort$. We will consider function symbols of any non-negative arity, and we will use* 0*-ary function symbols to represent* constants.

Definition 2 *Let $\Sigma = \langle Sort, Func \rangle$ be a many-sorted signature, and let $\mathcal{V} = \{x_1 : s_1, x_2 : s_2, \ldots\}$ be an infinite set of sorted variables, where the sorts are chosen from Sort. The set $Term(\Sigma, \mathcal{V})$ and the function $sort : Term(\Sigma, \mathcal{V}) \to Sort$ are defined inductively as follows:*

1. *If $x : s \in \mathcal{V}$, then $x \in Term(\Sigma, \mathcal{V})$ and $sort(x) = s$.*
2. *If $f : s_1 \times s_2 \times \cdots \times s_n \to s$ is a function symbol in Σ, and $t_1, \ldots, t_n \in Term(\Sigma, \mathcal{V})$ with $sort(t_i) = s_i$ for each i, then $f(t_1, \ldots, t_n) \in Term(\Sigma, \mathcal{V})$ with $sort(f(t_1, \ldots, t_n)) = s$.*

The structure $\langle Term(\Sigma, \mathcal{V}), sort \rangle$ is called a many-sorted term algebra *(often suppressing sort).*

HDTP has successfully been applied to a wide range of different examples from mathematics, ranging from modeling complex historical discoveries (such as, for example, Argand's reasoning process giving rise to the notion of the complex plane based on algebraic number theory and vector spaces [24]), through simulating the process of acquiring basic arithmetic and number concepts during children's development [15], to analyzing the mode of operation of analogy-based teaching tools in the mathematics classroom [3].

2.2 Concept Blending in Mathematics for Concept Invention

2.2.1 General Idea

One of the early formal accounts on concept blending, which is especially influential to our approach, is the classical work by Goguen using notions from algebraic specification and category theory (cf. [12, 13]). This version of concept blending can be described by the diagram in Figure 1, where each node stands for a representation an agent has of some concept or conceptual

$$\begin{tikzcd}
 & \textit{Generalization} \arrow[dl] \arrow[dr] & \\
\textit{Source} \arrow[rr, dotted, "\textit{analogical transfer}"] & & \textit{Target}
\end{tikzcd}$$

Figure 2: HDTP's overall approach to creating analogies (cf. [38]).

domain. We will call these representations "conceptual spaces" and in some cases abuse terminology by using the word "concept" to really refer to its representation by the agent. The arrows stand for morphisms, that is, functions that preserve at least part of the internal structure of the related conceptual spaces. The idea is that, given two conceptual spaces I_1 and I_2 as input, we look for a generalization G and then construct a blend space B in such a way as to preserve as many as possible structural alignments between I_1 and I_2 established by the generalization. This may involve taking the functions to B to be *partial*, in that not all the structure from I_1 and I_2 might be mapped to B. In any case, as the blend respects (to the largest possible extent) the relationship between I_1 and I_2, the diagram will commute.

Concept invention in mathematics by concept blending can then be phrased as the following task: given two axiomatizations of two mathematical theories I_1 and I_2, we need first, to compute a generalized theory G of I_1 and I_2 (which codes the commonalities between I_1 and I_2) and second, to compute the blend theory B in a structure preserving way such that new properties hold in B. Ideally, these new properties in B are considered to be (moderately) interesting mathematical properties.

The reasoning process in COINVENT is triggered by the computation of the generalization (generic space). This generalization is computed by HDTP ([38]), a framework for computing analogical relations between two conceptual spaces, each one of them presented as an axiomatization in a (possibly different) many-sorted first-order logic language. As a by-product of establishing an analogy, HDTP provides an explicit generalization of the two spaces which subsequently can be a base for concept creation by abstraction. HDTP proceeds in two phases: in the *mapping phase*, the source and target spaces are compared to find structural commonalities, and a generalized description is created, which subsumes the matching parts of both spaces. In the *transfer phase*, unmatched knowledge in the source space can be mapped to the target space to establish new hypotheses, cf. Figure 2.

For concept invention in the COINVENT project we will only need the mapping mechanism and replace the transfer phase by a new blending algorithm. The mapping is achieved via a generalization process, in which pairs of formulas from the source and target spaces are anti-unified resulting in a generalized theory that reflects common aspects of both spaces. Formulas (or terms) of the two input spaces that are generalized to the same formula (or term) in the generalized theory are considered to be analogically related. The generalized theory can be projected into the original spaces by substitutions which are computed during anti-unification. In the context of this deliverable, we will say that a formula is *covered* by the analogy, if it is in the image of this projection, otherwise it is *uncovered*. In analogy making, the analogical relations are used in the transfer phase to translate additional uncovered knowledge from the source to the target space, while blending combines additional (uncovered) facts from both spaces. Therefore the process of blending can build on the generalization and substitutions that are provided by the analogy engine.

2.2.2 Generalization

The computation of the generalization depends crucially on the anti-unification process in HDTP. First-order anti-unification was introduced by Plotkin [28] in the context of inductive learning. Figure 3 gives several examples for anti-unifications. Terms are generalized resulting in an anti-instance, where differing subterms are replaced by variables[3]. The original terms can be restored

$$\begin{tikzcd}
& X \arrow[dl, "\{X \to a\}"'] \arrow[dr, "\{X \to b\}"] & \\
a & & b
\end{tikzcd}$$

(a)

$$\begin{tikzcd}
& f(X,c) \arrow[dl, "\{X \to a\}"'] \arrow[dr, "\{X \to b\}"] & \\
f(a,c) & & f(b,c)
\end{tikzcd}$$

(b)

$$\begin{tikzcd}
& Y \arrow[dl, "\{Y \to f(a,b)\}"'] \arrow[dr, "\{Y \to g(a,b)\}"] & \\
f(a,b) & & g(a,b)
\end{tikzcd}$$

(c)

Figure 3: Plotkin's first-order anti-unification.

by replacing the new variables by appropriate subterms. This idea can be made more precise by introducing the notion of a substitution:

Definition 3 (Substitution) *Assume a term algebra $Term(\Sigma, \mathcal{V})$ is given. A* substitution *on terms is a partial function $\sigma : \mathcal{V} \to Term(\Sigma, \mathcal{V})$ mapping variables to terms, formally represented by $\sigma = \{x_1 \to t_1, \ldots, x_n \to t_n\}$ (provided $x_i \neq x_j$ for $i, j \in \{1, \ldots, n\}$, $i \neq j$ and sorts of x_i and t_i match). An application of a substitution σ on a term is defined by induction over the term structure:*

- $apply(x, \sigma) = \begin{cases} t & \text{if } x \to t \in \sigma \\ x & \text{otherwise} \end{cases}$
- $apply(f(s_1, \ldots, s_m), \sigma) = f(apply(s_1, \sigma), \ldots, apply(s_m, \sigma))$

We say that a term t' is an instance *of t and t is an* anti-instance *of t', if there is a substitution σ such that $apply(t, \sigma) = t'$. In this case we write $t \xrightarrow{\sigma} t'$ or simply $t \to t'$.*

Using substitutions, generalizations can be defined as follows:

Definition 4 (Generalization) *A generalization for a pair of terms $\langle s, t\rangle$ is a triple $\langle g, \sigma, \tau\rangle$ with a term g and substitutions σ, τ such that $s \xleftarrow{\sigma} g \xrightarrow{\tau} t$.*

Anti-unification aims to find a most specific anti-unifier, normally referred to as least general generalization (*lgg*), i.e. a generalization that is minimal with respect to the instantiation ordering.[4] It has been proven in [28] that for a given pair of terms a first-order generalization always exists and that the *lgg* is unique (up to renaming of variables).

Figure 3 demonstrates how generalizations can induce an analogical relation: in (a), the terms a and b are generalized to X, therefore b can be seen as an analogon to a in the target domain. In (b), the terms a and b are embedded as function arguments in a common context, but still the

[3]Variables introduced by the anti-unification are denoted by capital letters.

[4]This is dual to unification, where a most general unifier (*mgu*) is computed.

same substitutions can be used for generalization and therefore the same analogical relation is established. In (c), the two terms differ with respect to the function symbols. Here, the first-order anti-unification fails to detect the common structure between these terms and generalizes it to X, using rather complex substitutions.

HDTP applies a restricted form of higher-order anti-unification [21] for analogy making. The main problem when dealing with higher-order anti-unification is that generalizations can become arbitrarily complex and may no longer reflect structural commonalities of the original terms. Therefore, extending the set of possible generalizations in a controlled way by introducing a new notion of basic substitution is a natural solution to this problem.

We extend classical first-order terms by introducing variables that can take arguments: for every natural number n we assume an infinite set $\mathcal{V}_n = \{F : s_1 \times \ldots \times s_n \to s, \ldots\}$ of variables with arity n and $s_1, \ldots, s_n, s \in Sort_\Sigma$. Here we explicitly allow the case $n = 0$ with $\mathcal{V}_0$ being the set of first-order variables. In this setting, a *term* is either a first-order or a higher-order term, i.e. an expression of the form $F(t_1, \ldots, t_n)$ with $F : s_1 \times \ldots \times s_n \to s \in \mathcal{V}_n$, terms $t_1, \ldots, t_n \in \mathrm{Term}(\Sigma, \mathcal{V})$, and $\mathrm{sort}_\Sigma(t_i) = s_i$. Analogously to the first-order case shown in figure 4, terms can be anti-unified to a generalization subsuming the specific terms. The basic substitutions given in the following list are applicable in HDTP. These are sufficient for generalizations in analogical reasoning and meet the requirement to only generate less complex anti-instances.

Definition 5 (Basic Substitutions) *We define the following set of basic substitutions:*[5]

1. *A* renaming $\rho^{F,F'}$ *replaces a variable* $F \in \mathcal{V}_n$ *by another variable* $F' \in \mathcal{V}_n$ *of the same argument structure:*
$$F(t_1, \ldots, t_n) \xrightarrow{\rho^{F,F'}} F'(t_1, \ldots, t_n).$$

2. *A* fixation ϕ_c^F *replaces a variable* $F \in \mathcal{V}_n$ *by a function symbol* $f \in \mathcal{C}_n$ *of the same argument structure:*
$$F(t_1, \ldots, t_n) \xrightarrow{\phi_f^F} f(t_1, \ldots, t_n).$$

3. *An* argument insertion $\iota_{G,i}^{F,F'}$ *with* $0 \leq i \leq n$*,* $F \in \mathcal{V}_n$*,* $G \in \mathcal{V}_k$ *with* $k \leq n - i$*, and* $F' \in \mathcal{V}_{n-k+1}$ *is defined by*
$$F(t_1, \ldots, t_n) \xrightarrow{\iota_{G,i}^{F,F'}} F'(t_1, \ldots, t_i, G(t_{i+1}, \ldots, t_{i+k}), t_{i+k+1}, \ldots, t_n).$$

4. *A* permutation $\pi_\alpha^{F,F'}$ *with* $F, F' \in \mathcal{V}_n$ *and bijective* $\alpha : \{1, \ldots, n\} \to \{1, \ldots, n\}$ *rearranges the arguments of a term:*
$$F(t_1, \ldots, t_n) \xrightarrow{\pi_\alpha^{F,F'}} F'(t_{\alpha(1)}, \ldots, t_{\alpha(n)}).$$

Figure 4 gives examples for all basic substitutions. (a) shows an example for *renaming*: the terms in the source and the target domain both contain variables Y and Z which are generalized to variable X. Since variables can represent any possible term, it is irrelevant which variable name is

[5]To improve readability we omit the sortal specifications of the variable symbols, as long as they can be inferred from the context

$$\begin{tikzcd} & f(X) \arrow[dl, "\rho^X_Y"'] \arrow[dr, "\rho^X_Z"] & \\ f(Y) & & f(Z) \end{tikzcd}$$

(a)

$$\begin{tikzcd} & f(X) \arrow[dl, "\phi^X_a"'] \arrow[dr, "\phi^X_b"] & \\ f(a) & & f(b) \end{tikzcd}$$

(b)

$$\begin{tikzcd} & F(a) \arrow[dl, "\phi^F_f"'] \arrow[dr, "\phi^F_g"] & \\ f(a) & & g(a) \end{tikzcd}$$

(c)

$$\begin{tikzcd} & F(a,b,c) \arrow[dl, "\iota^{F,F'}_{X,2}"'] \arrow[dr, "\iota^{F,F''}_{G,1}"] & \\ F'(a,b,X,c) & & F''(a,G(b,c)) \end{tikzcd}$$

(d)

$$\begin{tikzcd} & F(a,b) \arrow[dl] \arrow[dr, "\pi^{F,F'}_\alpha"] & \\ F(a,b) & & F'(b,a) \end{tikzcd}$$

(e)

Figure 4: Examples for all basic substitutions of the restricted higher-order anti-unification.

chosen. It is possible to align a variable of the source domain with a variable in the target domain without any cost. The renaming substitution is only required for formal reasons: it does not lead to a real generalization.

Argument fixation as shown in (b) can be used to replace a variable in the generalization by a symbol of the same argument structure, e.g. $f(X)$ is replaced by $f(a)$ in the source, respectively $f(b)$ in the target, with $X \in \mathcal{V}_0$. Example (c) demonstrates a fixation of a higher-order term $F(a)$ to $f(a)$, respectively $g(a)$. The higher-order variable F has one argument, therefore $F \in \mathcal{V}_1$.

Argument insertion is slightly more complicated: inserting a 0-ary variable X increases the arity of the embedding term by 1. In (d), a variable X is inserted after the second argument on the source side $F(a,b,c) \rightarrow F'(a,b,X,c)$. F' has now four arguments. Inserting a variable $G \in \mathcal{V}_n$ with $n \geq 2$ reduces the arity: On the target side a two-ary variable G is inserted after the first argument $F(a,b,c) \rightarrow F''(a,G(b,c))$. F'' has now only two arguments. This basic substitution is required if a complex structure maps on a less complex structure.

An example for *permutation* is shown in (e). Source and target domain contain terms with an equivalent structure. They differ only with respect to the argument order. $F(a,b)$ serves as generalization for both terms. On the target side, both arguments are permuted and $F(a,b) \rightarrow F'(b,a)$.

For generalizing complex terms, we can successively apply several substitutions: To receive a non-ambiguous set of substitutions we apply the basic substitutions in the order renaming, argument insertion, permutation, and finally fixation. We will call any composition of basic substitutions a (higher-order) substitution and write $t \rightarrow t'$, if there exists a sequence of basic substitutions that transforms t into t'. Again we will call t' an (higher-order) instance of t, and t an (higher-order) anti-instance of t'.

It has been proven in [21] that the application of a basic substitution will never make a term less complex and so the following fact holds:

Fact 1 *For a given term t there are (up to renaming) only finitely many anti-instances (i.e. terms s with $s \rightarrow t$).*

Fact 1 implies that this notion of substitution is a viable tool to compute generalizations in the context of analogy making. It is a real extension of first-order substitution and it is capable of detecting structural commonalities that are ignored by first-order anti-unification. Therefore, this framework fits for the purpose of concept invention by conceptual blending, namely as a trigger for the reasoning process by computing a generalization (generic space) of the two input domains formulated in a many-sorted first-order logical language.

$$\begin{tikzcd}
 & G \arrow[dl, "\cong"'] \arrow[dr, "\cong"] & \\
L \arrow[rr, dotted, "\cong"] \arrow[dr, "\cong"'] & & R \arrow[dl, "\cong"] \\
 & L \cong R &
\end{tikzcd}
\qquad
\begin{tikzcd}
 & \emptyset \arrow[dl] \arrow[dr] & \\
L \arrow[dr] & & R \arrow[dl] \\
 & L \oplus R &
\end{tikzcd}$$

Figure 5: The two extreme cases of input spaces, along with their generalizations and blends.

2.2.3 Blending

In the sketched setting, the blend depends on the portion of the original theories that are covered by the analogy. There are two extreme cases: The first one, shown in the left diagram of Figure 5, occurs when the two input spaces are isomorphic, meaning that there is a bijective morphism that amounts to a simple renaming of signature symbols from the language of L onto the symbols of R.

In that case, all formulae of the theories can be generalized and are completely covered by the analogy found by HDTP, and the resulting blend will be isomorphic to both of them. The other extreme (depicted in the right diagram of Figure 5) is an analogy where no formulae are aligned and therefore the generalized theory is empty, hence, no formulae of the domain theories are covered by the analogy. In this case, a blend can always be obtained by taking the disjoint union of the two input theories (this disjoint union may be inconsistent, however). In practice, neither of these two cases is of real interest (provided there is no interest in knowing whether the disjoint union of two input spaces is consistent). The interesting blends will be cases where only parts of the input theories are covered by the analogy. In fact, one can adjust the blend by changing the generalization, either by removing formulae and thereby reducing the number of mappings and the coverage of the analogy, or by choosing another analogy which associates different formulae. Due to the fact that HDTP outputs a ranked list of different candidates for analogical relations in any case, using another generalization is rather straightforward.

The left theory L and the right theory R can be split into (non-empty) covered parts L^+ and R^+ and uncovered parts L^- and R^-. The covered parts are fully analogical, i.e. basically isomorphic, and make up the core of the blend, which we will call B. The uncovered parts L^- and R^- reflect the idiosyncratic aspects of the domains, which we would ideally want to integrate into the blend B. However, due to the identifications induced by the analogical relation, adding all this to B may result in an inconsistent theory. To preserve consistency, in such cases we may be forced to consider only consistent subsets of this ideal, fully inclusive, blend. We will call a blend of two theories *maximally compressed* if it is consistent and identifies the largest number of corresponding signature symbols. A blend is called *maximally informative* if it includes a maximal consistent subset of the set of potentially merged formulas (obtained by first taking the union of the input theories and then replacing pairs of signature symbols that have been identified by the analogy by only one unified symbol).

We propose the following criteria to select optimal blends: (1) the more informative and the more compressed a blend, the better it is; (2) we are interested in blends that combine both domains, i.e. that contain formulae from L^- and R^- (since otherwise the blend would just be a subtheory of L or R).[6]

It turns out that in the mathematical context the computation of blend spaces is strongly related to a colimit construction, well-known from category theory (cf. Deliverable D3.1). In the worked example (2.3), using HETS for the computation of the blend the mappings from the generalization to the input spaces and the mappings from the input spaces to the blend space are interpreted categorically as theory morphisms. Theory morphisms are intuitively mappings between signatures that preserve the types of the symbols as well as logical consequences. These morphisms are logic independent (i.e. they work without any restrictions for many-sorted first order theories). In Deliverable D3.1, the computational machinery for this process is thoroughly explained.

2.3 Worked Example: Complex Numbers

The complex numbers example is one which historically took many years to be established in mathematics, and is therefore a hard case to model. Nevertheless, we present this work as showing that a rational reconstruction of the modern understanding of complex numbers is possible using the techniques we describe below: given an axiomatization of an order field and an axiomatization of a two-dimensional vector space first, HDTP computes an appropriate generalization (generic space) and second, HETS computes a blend.

2.3.1 Formal Statement of Problem

The complex numbers can be understood as a blend between the geometrical notion of normed real-valued vector space, and the algebraic notion of a field.

- The ordered field axioms for $\mathscr{R}$:
    1. $\mathscr{R}$ with $+_{\mathscr{R}}$, real 0 and unary $-_{\mathscr{R}}$ form a commutative group.
    2. $\mathscr{R}$ without the real 0, with $\times_{\mathscr{R}}$, $\mathbf{1}$ and ${}^{-1_{\mathscr{R}}}$ form a commutative group.
    3. $\times_{\mathscr{R}}$ distributes over $+_{\mathscr{R}}$.
    4. $\leq$ is a total order on $\mathscr{R}$, which respects addition; furthermore, the product of positive numbers is positive.
- The elements of V form a commutative group with identity $\mathbf{0}$ and inverse operation (here as prefix minus).

$$\forall x,y:V \qquad x+y=y+x \tag{1}$$
$$\forall x,y,z:V \qquad (x+y)+z=x+(y+z) \tag{2}$$
$$\forall x:V \qquad x+\mathbf{0}=x \tag{3}$$
$$\forall x:V \qquad x+(-x)=\mathbf{0} \tag{4}$$

[6] Please note that, if L^- or R^- is fully contained in the blend, this is merely a case of analogy (where some parts of the other theory are imported via analogical transfer).

- Interaction between field operations and vector operations. Here the field is the real numbers, operations are written with subscript $\mathscr{R}$; **1** is the field multiplicative identity.

$$\forall x,y:V\ \forall\lambda:\mathscr{R} \qquad \lambda(x+y)=\lambda x+\lambda y \tag{5}$$
$$\forall x:V\ \forall\lambda,\mu:\mathscr{R} \qquad (\lambda+_{\mathscr{R}}\mu)x=\lambda x+\mu x \tag{6}$$
$$\forall x:V\ \forall\lambda,\mu:\mathscr{R} \qquad \lambda(\mu x)=(\lambda\times_{\mathscr{R}}\mu)x \tag{7}$$
$$\forall x:V \qquad \mathbf{1}x=x \tag{8}$$

- Finally, there is a norm on the vector space (giving the size of vectors): $||.||:V\to\mathscr{R}$, with associated axioms.

$$\forall x:V \qquad ||x||\geq 0 \tag{9}$$
$$\forall x:V \qquad ||x||>0\leftrightarrow x\neq 0 \tag{10}$$
$$\forall x:V\ \forall\lambda:\mathscr{R} \qquad ||\lambda x||=|\lambda|\times_{\mathscr{R}}||x|| \tag{11}$$
$$\forall x,y,z:V \qquad ||x+y||\leq ||x||+||y|| \tag{12}$$

On the other side, there are the field axioms, which are already part of the vector space axiomatization. The desired generalized theory linking the axiomatizations will not use that link, but it is an obvious common feature of the two axiomatizations.

For completeness, here is an axiomatization.

- F with $+_{\mathrm{F}}, 0_F, \text{-}_{\mathrm{F}}$ is a commutative group:

$$\forall x,y:F \qquad x+_{\mathrm{F}}y=y+_{\mathrm{F}}x \tag{13}$$
$$\forall x,y,z:F \qquad (x+_{\mathrm{F}}y)+_{\mathrm{F}}z=x+_{\mathrm{F}}(y+_{\mathrm{F}}z) \tag{14}$$
$$\forall x:F \qquad x+_{\mathrm{F}}0_F=x \tag{15}$$
$$\forall x:F \qquad x+_{\mathrm{F}}(\text{-}_{\mathrm{F}}x)=0_F \tag{16}$$

- F apart from 0_F with $\times_{\mathrm{F}}, 1_F, ^{\text{-}1_{\mathrm{F}}}$ is a commutative group.
  Here a subtype $F_{\neq 0}$ of F is used with an obvious definition.

$$\forall x,y:F_{\neq 0} \qquad x\times_{\mathrm{F}}y=y\times_{\mathrm{F}}x \tag{17}$$
$$\forall x,y,z:F_{\neq 0} \qquad (x\times_{\mathrm{F}}y)\times_{\mathrm{F}}z=x\times_{\mathrm{F}}(y\times_{\mathrm{F}}z) \tag{18}$$
$$\forall x:F_{\neq 0} \qquad x\times_{\mathrm{F}}1_F=x \tag{19}$$
$$\forall x:F_{\neq 0} \qquad x\times_{\mathrm{F}}x^{\text{-}1_{\mathrm{F}}}=1_F \tag{20}$$

- Distributivity:

$$\forall x,y,z:F \qquad x\times_{\mathrm{F}}(y+_{\mathrm{F}}z)=(x\times_{\mathrm{F}}y)+_{\mathrm{F}}(x\times_{\mathrm{F}}z) \tag{21}$$

We may now make clear what we are looking for in a blend. First, we are working in first-order sorted logic, and the morphisms should map formulas to formulas while respecting the logical form of our formulas. The generic theory should look for a morphism which maps axioms into axioms (or perhaps theorems) of *both* input theories, such that (parts) of the input theories are instances of the mapped generic theory. And the blend should include aspects of both theories, while respecting the generic aspects.

There are a couple of obvious, but non-creative, answers to the problem as stated.

1. Since the theory of the reals appears in both input spaces, we can take that to be the generic space.

   In this case we get nothing new, the blend is identical to the vector space theory.

2. Much more interestingly, the generic space can identify the addition operation in the theory of the reals with the addition operation on vectors (in both cases these are Abelian groups).

   In this case, the blend says that the operations on vectors should include vector multiplication, following the example of real multiplication.

The latter case we happen to know to be consistent, because the zero-dimensional real vector space is a model (in which there is only one (zero) vector).

This is not taking us where we want to go, so far, because we have not insisted that the vector space is two-dimensional. We can do this, by adding this constraint to the vector space axioms.

Two-dimensional case We now give ourselves products of sorts and pairings directly, and then work with a refinement of normed vector spaces, i.e. one where the axioms, suitably interpreted, hold, as well as the two-dimensionality axioms, again suitably interpreted. So, work with pairs of reals:

$$(x_1,y_1)+(x_2,y_2) =_{def} (x_1 +_{\mathscr{R}} x_2, y_1 +_{\mathscr{R}} y_2) \tag{22}$$
$$\mathbf{0} =_{def} (0,0) \tag{23}$$
$$-(x,y) =_{def} (\text{-}_{\mathscr{R}} x, \text{-}_{\mathscr{R}} y) \tag{24}$$
$$\lambda(x,y) =_{def} (\lambda \times_{\mathscr{R}} x, \lambda \times_{\mathscr{R}} y) \tag{25}$$
$$\|(x,y)\|^2 = ((x \times_{\mathscr{R}} x) +_{\mathscr{R}} (y \times_{\mathscr{R}} y))^2 \tag{26}$$

We can in fact show that, with these conditions, we do indeed have a real (normed) vector space.

Now we can look at the blend of 2-dimensional vector spaces with the real field. Again we can look to see whether the blend is consistent, i.e. is there some multiplication operation over vectors that turns the vectors into a field?

A way to show that the blend here is consistent (assuming the input spaces are) is to provide definitions of multiplication, identity, and inverse in terms of what we already have, satisfying the extra field axioms. Such abbreviational definitions are guaranteed to yield a conservative extension, therefore no new inconsistency will be introduced.

For the case of complex numbers, the standard definitions are at hand.

$$\mathbf{1} =_{def} (1,0) \tag{27}$$
$$\begin{aligned}(x_1,y_1)\times(x_2,y_2) =_{def} ((x_1 \times_{\mathscr{R}} x_2) +_{\mathscr{R}} (\text{-}_{\mathscr{R}}(y_1 \times_{\mathscr{R}} y_2)),\\ (x_1 \times_{\mathscr{R}} y_2) +_{\mathscr{R}} (y_1 \times_{\mathscr{R}} x_2))\end{aligned} \tag{28}$$
$$(x,y)^{-1} =_{def} \|(x,y)\|^{-2_{\mathscr{R}}}(x, \text{-}_{\mathscr{R}} y) \tag{29}$$

(where the inverse square operation is shorthand for the obvious thing).

The preservation of norm under multiplication While the norm operation seems to have played little role so far, we note the importance to the historical development of the role of one of the properties of the usual Euclidean norm.

The following is part of the presentation in [9]:

> The geometrical properties of the complex numbers follow from the fact that they form a composition algebra for the Euclidean norm[7]
>
> $$N(x+iy) = x^2 + y^2$$
>
> which means that
>
> $$N(z_1 z_2) = N(z_1)N(z_2).$$
>
> This entails that multiplication by a fixed (non-zero) number z_0 multiplies all lengths by $\sqrt{N(z_0)}$; ...

Their claim depends on the fact that, in a real vector space, a Euclidean norm with this compositional property uniquely determines the corresponding inner product (see [10] for a justification of the claim).

At any event, this compositionality for norms played an important role in Hamilton's discovery of Hamiltonians, where he says of the property (cf. [17]):

> ...without which consistence being verified, I should have regarded the whole speculation as a failure.

The importance of this property suggests that the blending process should not concentrate entirely on axioms. While our earlier thoughts on blending focused exclusively on operations on sets of axioms, it is quite possible to include other statements such as key theorems in the formalism associated with the initial theory.

In the case of complex numbers, this is the statement

$$||z_1.z_2||^2 = ||z_1||^2.||z_2||^2$$

using a less fussy notation than that above. The centrality of this feature is further emphasized by Argand's discussion of the role of absolute magnitude in his description both of real and complex numbers [2], one of the first expositions of the modern understanding of complex numbers.

The role of key properties associated with a particular theory is reminiscent of Lakatos's characterization of the *hard core* properties of a scientific research programme: features which are central to the identity of the theory and not to be given up, unlike other dispensable features [22].

[7]Conway & Smith here take the norm to be the square of the usual Euclidean distance.

2.3.2 Implementation

In order to automate the process of discovering blends between theories, we employ the HETS system [26]. HETS supports reasoning about logical theories and relationships between them (it also supports reasoning about relationships between different logics, though we make no use of this for present purposes). It will generate proof obligations when claims are made about relationships, such as that everything provable in one theory is provable in another theory, under some translation into different syntax. It is thus well suited to help in organizing and verifying claims about analogical inference.

When HETS is given signature morphisms between theories (a particular case of translations that preserve provability), it can compute a *colimit* which is the categorical equivalent of what we have thus far referred to as a *blend*. In order to compute a blend between two theories, we must first calculate the signature morphisms defining the *generic* space which is common to both theories. To do this we exploit the HDTP software [16, 37] which computes a generalized version of the input theories.

Specification using DOL In order to specify a theory and to further define signature morphisms we use the *DOL* specification language [23]. For our example of complex numbers we use the CASL logic [4] to describe the theories. Initially we define the signature of an ordered field and a two-dimensional vector space. The specification files themselves are too big to show here but can be found in full online in the Ontohub system at `http://ontohub.org/complex-blend/complex_blend`. For the purposes of exposition we show simplified files here. Below is part of a definition of an ordered field, and part of a definition of a vector space:

```
spec Field =
  sort Real
  ops
    0:Real;
    __ + __: Real * Real -> Real;
    - __   : Real -> Real;
    __ * __: Real * Real -> Real
  forall x,y:Real
    . x+0=x    %f_plus_ident%
    . x+y=y+x  %f_com_plus%
    . x+(-x)=0 %f_plus_inv%
end

spec VectorSpace =
  sort Real
  free type Vec ::= pair(r:Real;c:Real)
  ops
    vzero:     Vec;
    0:         Real;
    vmi __:    Vec -> Vec;
    __ vpl __: Vec * Vec -> Vec
  forall x,y:Vec;
    . x vpl vzero = x;     %v_plus_ident%
    . x vpl y = y vpl x    %v_com_plus%
```

$$\begin{tikzcd}
 & Gen \arrow[dl, "?"'] \arrow[dr, "?"] & \\
Ax(NVS2) \arrow[dr, "?"'] & & Ax(F) \arrow[dl, "?"] \\
 & Blend &
\end{tikzcd}$$

Figure 6: Diagram for the theory of two-dimensional vectors.

```
    . x vpl (vmi x) = zero %v_plus_inv%
end
```

This mirrors the exposition given for the definitions of a field and a vector space, introducing a pair type to indicate that we want to develop a theory of two-dimensional vectors. This gives us the following diagram of Figure 6. $Ax(F)$ are the axioms of the real ordered field, and $Ax(NVS2)$ are the axioms of a normed two-dimensional vectors space. The ? correspond to the signature morphisms that will be calculated by HDTP and HETS.

Calculation of Generic Space Since a vector space itself comprises an ordered field, there is a generic space which is calculated using identity morphisms which reproduces the vector space in the blend, as described in Subsection 2.3.1. We are interested here in a generic space together with a morphism which is different to the identity morphism. We exploit the HDTP system to generate the following signature morphism:

$$0 \quad \leftarrow_{G\to f} \quad zero \quad \rightarrow_{G\to v} \quad vzero \tag{30}$$

$$+ \quad \leftarrow_{G\to f} \quad plus \quad \rightarrow_{G\to v} \quad vpl \tag{31}$$

$$- \quad \leftarrow_{G\to f} \quad minus \quad \rightarrow_{G\to v} \quad vmi \tag{32}$$

which generates the following Generic space in HETS:

```
spec Gen =
  sort Generic
  ops
    __ plus __: Generic * Generic -> Generic;
    zero:       Generic;
    minus __:   Generic -> Generic
  forall x,y,z:Generic
    . x plus  zero=x          %gen_plus_ident%
    . x plus y=y plus x       %gen_com_plus%
    . x plus (minus x) = zero %gen_plus_inv%
end
```

We now have a calculated definition of a generic space by which we can calculate a colimit. We need first to indicate in HETS using the DOL language how the signature morphism calculated by HDTP is specified (cf. also Figure 7):

$$\begin{tikzcd}
 & Gen \arrow[dl, "G\to v"'] \arrow[dr, "G\to f"] & \\
Ax(NVS2) \arrow[dr, "?"'] & & Ax(F) \arrow[dl, "?"] \\
 & Blend &
\end{tikzcd}$$

Figure 7: Diagram specifying the signature morphisms calculated by HDTP.

```
view I1: Gen to Field =
  __ plus __ |->  __ + __,
  zero       |->  0,
  minus __   |->  - __

view I2: Gen to VectorSpace =
  __ plus __ |->  __ vpl __,
  zero       |->  vzero,
  minus __   |->  vmi __
```

It now remains to establish the signature morphisms to the computed colimit. The resulting theory is the blend of the ordered real field and the two dimensional vector space.

Calculation of Colimit (Blend) HETS automatically computes the colimit by determining which laws must be associated with symbols in the blend. The symbol `plus` from the Generic theory is now both associated with field addition from the ordered field and vector addition from the two dimensional vector space. The effect of this is that the two dimensional vector space "inherits" the notion of multiplication from the ordered real field. For the complex numbers this is indeed what we want, and we also inherit a definition of multiplicative inverse for vectors.

2.4 Further Worked Examples

Additional worked examples which are both, in the spirit of and predecessors of the complex number example of Subsection 2.2 can be found in the published papers [15] and [24]. The paper [3] provides an explicit modeling and a further example of analogies and analogical reasoning used in classroom situations. General algorithmic aspects of concept blending in the context of mathematical theories can be found in [25] (currently in press).

3 Representation of and Reasoning in Music to Create New Concepts

3.1 Abstract Representation of Musical Aspects

In mathematics, it is rather obvious how to represent mathematical theories: it is natural to represent theories in a logical language in form of axiomatic systems. Music is, however, a domain for which the choice of the representation formalism is less clear. One reason is that music can be analyzed on many different levels: For example, music can be analyzed on the level of its physical realization, on the MIDI level, on the level of motifs and chords of a piece of music, on the level of chord progression structures (harmonic structures), on the level of larger structures of whole pieces of music (e.g. sonata form, rondo form), or on a meta-level where concepts outside of the musical domain are used to describe music (e.g. Chopin's "Raindrop" Etude, Beethoven's "Pathetique" Sonata, Tchaikovsky's "Song of a Lark" piano piece contained in his Op. 37a). Due to the fact that the breadth of the sketched description levels ranges from areas which are usually modeled with subsymbolic representations (e.g. physical level) to areas where highly conceptual and symbolic representations are plausibly used (e.g. meta-level analysis of music), music is a natural candidate for a domain for which the choice of the particular description level constrains the representation formalism.

Consequently, in the COINVENT project different representation formalisms are used for modeling certain aspects of concept invention in music. As a matter of fact it turned out that already on a particularly chosen specific level of analysis, e.g. the harmonization level, a variety of different representation formalisms can be used dependent on whether the representation should cover uncertainty aspects or not, whether the representation should be able to check well-formedness conditions or not etc. Therefore, three representation formalisms are sketched which are used throughout this report: General Chord Type representations (Subsection 3.1.1), Constraint Hidden Markov Models (Subsection 3.1.2), and feature structures (Subsection 3.1.3).

3.1.1 General Chord Type Representations (GCT)

Within the context of the COINVENT research, a new idiom-independent representation of chord types is proposed of chord that is appropriate for encoding tone simultaneities in any harmonic context (such as tonal, modal, jazz, octatonic, atonal). The General Chord Type (GCT) representation, allows the re-arrangement of the notes of a harmonic simultaneity such that abstract idiom-specific types of chords may be derived; this encoding is inspired by the standard roman numeral chord type labeling, but is more general and flexible. Given a consonance-dissonance classification of intervals (that reflects culturally-dependent notions of consonance/dissonance), and a scale, the GCT algorithm finds the maximal subset of notes of a given note simultaneity that contains only consonant intervals; this maximal subset forms the base upon which the chord type is built. The proposed representation is ideal for hierarchic harmonic systems such as the tonal system and its many variations, but adjusts to any other harmonic system such as post-tonal, atonal music, or traditional polyphonic systems.

For the GCT representation the computation of a pitch class simultaneity (chord) and a classification of intervals into consonant/dissonant (binary values) is considered, alongside with an appropriate scale background (i.e. scale with tonic). The GCT algorithm computes for a given multi-tone simultaneity the "optimal" ordering of pitches such that a maximal subset of consonant intervals appears at the "base" of the ordering (left-hand side) in the most compact form. The tonal center (key) is considered as given, therefore the position within the given scale is automatically calculated. Roughly, the input to the algorithm is the following:

1. Consonance vector: The user defines which intervals are consonant/dissonant through a 12-point Boolean vector of consonant (1) or dissonant (0) intervals. For instance, the vector

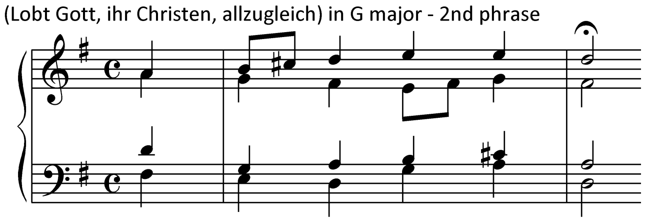
$[1,0,0,1,1,1,0,1,1,1,0,0]$ means that the unison, minor and major third, perfect fourth and fifth, minor and major sixth intervals are consonant – dissonant intervals are the seconds, sevenths and the tritone; this specific vector is referred to in this text as the common-practice consonance vector.

2. Pitch Scale Hierarchy: The pitch hierarchy (if any) is given in the form of scale tones and a tonic (e.g. a D maj scale is given as: $2,[0,2,4,5,7,9,11]$, or an A minor pentatonic scale as: $9,[0,3,5,7,10]$).

3. Input chord: list of MIDI pitch numbers (converted to pc-set).

The GCT algorithm encodes most chord types "correctly" in the standard tonal system. However, the algorithm is undecided in some cases and even makes "mistakes" in other cases. In most instances of multiple encodings, it is suggested that these ideally should be resolved by taking into account other harmonic factors (e.g. bass line, harmonic functions, tonal context, etc). For more details about strategies to resolve these issues, the interested reader is referred to [8]. An example of the GCT analysis-representation is illustrated in Figure 8.

J.S.Bach - Chorale 54 (Lobt Gott, ihr Christen, allzugleich) in G major - 2nd phrase



Roman Numeral Analysis:						
D major	I^6	vii_o^6	I	ii^6	V^7	I
GCT Analysis (tonal major profile)						
2,[0,2,4,5,7,9,11]	0,[0,4,7]	11,[0,3,6]	0,[0,4,7]	2,[0,3,7]	7,[0,4,7,10]	0,[0,4,7]
Pc-Set Analysis (chromatic scale):						
normal orders	[0,4,7]	[0,3,6]	[0,4,7]	[0,3,7]	[0,2,6,9]	[0,4,7]
prime forms	[0,3,7]	[0,3,6]	[0,3,7]	[0,3,7]	[0,3,6,8]	[0,3,7]
GCT Analysis (atonal profile)						
[0,1,2,3,4,5,6,7,8,9,10,11]	2,[0,4,7]	1,[0,3,6]	0,[0,4,7]	4,[0,3,7]	7,[0,2,6,9]	2,[0,4,7]

Figure 8: Chord analysis of a Bach Chorale phrase by means of traditional roman numeral analysis, pc-sets and two versions of the GCT algorithm.

3.1.2 Constrained Hidden Markov Models

Hidden Markov models (HMMs) have been extensively used for the automatic harmonization of a given melody, since their formalization describes the targeted task very well: given a sequence of observed notes (melody), find the most probable (hidden) sequence of chords that is compatible with the observations, according also to a chord transition matrix. In several studies of HMM–based melodic harmonization methodologies, a straightforward distinction is made on the role that some chords play to the composition – mainly the cadence of the phrase. For instance, the cadences of produced harmonizations by the HMM developed in [5] were utilized to evaluate the system's performance, by comparing the cadence patterns that were produced by the system to the ones observed in the dataset.

Several HMM approaches discuss the utilization of some methodological tools to amplify the role of the cadence in the harmonization process. For instance, in [1] and [18] a backwards propagation of the HMM methodology is proposed, i.e. by examining the prior probabilities of the final chord given the final melodic note. The Markov decision process followed in [42] rewards the authentic cadences, thus providing higher probabilities to chord sequences that end with an authentic cadence. In [43] the phrases are divided in tonic, subdominant, dominant, and parallel tonic chords, allowing a trained HMM to acknowledge the positions of cadences, although the selection of chords is performed through a rule–based process. A commercial application utilizing HMM for melodic harmonization is *mySong* [39], which receives the melody by the singing voice of the user, extracts the pitches of the melody and employs an HMM algorithm to provide chords for the melody. According to the HMM approach utilized in mySong, prior probabilities are considered not only for the beginning chord of a piece, but also for the ending one, a fact that further biases the choice of solutions towards ones that incorporate first and final chords that are more often met in the training dataset.

The approach presented in this section is motivated by the research in the aforementioned works, but it is different on a fundamental aspect: it allows the *deterministic* (not probabilistic) insertion of chords at any place in the chord sequence. Such an approach is important since it permits the extension of the "learned" transitions, potentially allowing to build composite harmonization that comprise characteristics from various idioms. To this end, the isolation of the harmony in "strategic" harmonic positions (e.g. the cadence, the beginning or intermediate parts of a phrase) is expected to contribute to the project's perspective.

The chords that "connect" two successive fixed–boundary chord segments are defined by a variation of HMM, the BCHMM. Throughout the development of the BCHMM, a nomenclature relative to the subject under discussion will be followed, i.e. the dataset will comprise musical pieces (more specifically harmonic reductions of pieces), the states will represent chords and the observations will describe melody notes. To this end, the set of possible state–chords will be denoted by $\mathscr{S}$, while the letters C and c will be used for denoting chords. The set of all possible observation–notes will be denoted as $\mathscr{Y}$, while Y and y will be denoting melody notes. Specifically, the capitalized letters will be used to denote statistical variables, while their instantiation variables will be denoted by lower case letters. For example, $P(C_i = c_i)$ denotes the probability that the chord in the i–th position is a c_i chord (where c_i is a specific chord, e.g. a [7, $[0,4,7]$, [10]] chord in GCT form, which is a dominant seventh chord).

An initial set of music phrases is considered which will provide the system with the required statistical background, constituting the training set. Through this dataset the statistics that are induced concern three aspects:

1. The probability for each state (chord) to be a beginning chord. This distribution is computed by examining each beginning chord for each phrase in the dataset and is denoted as $\pi(C_1 = c)$, $c \in \mathscr{S}$.

2. The probability for each state (chord) to be an ending chord. This distribution is computed by examining each ending chord for each phrase in the dataset and is denoted as $\tau(C_T = c)$, $c \in \mathscr{S}$.

3. The probability that each state follows another state, denoted as $P(C_i = c_i | C_{i-1} = c_{i-1})$, c_i, $c_{i-1} \in \mathscr{S}$.
4. The probability of a chord being played over a melody note, denoted as $P(C_i = c_i | Y_i = y_i)$.

These probabilities are related during the computation of the *overall* probability that a certain chord sequence ($C_i = c_i$, $i = 1, 2, \ldots, T$) is applied over an observed melody ($Y_i = y_i$, $i = 1, 2, \ldots, T$). This overall probability is computed by

$$P(C_i = c_i | Y_i = y_i) = P_\pi \, P_\mu \, P_\tau, \tag{33}$$

where

$$P_\pi = \pi(C_1 = c_1) \, P(C_1 = c_1 | Y_1 = y_1), \tag{34}$$

$$P_\mu = \prod_{i=2}^{T} P(C_i = c_i | C_{i-1} = c_{i-1}) \quad P(C_i = c_i | Y_i = y_i), \tag{35}$$

$$P_\tau = \tau(C_T = c_T) \, P(C_T = c_T | Y_T = y_T). \tag{36}$$

An optimal sequence of chords is one that maximizes the overall probability (in Equation 33)[8], by achieving an optimal path of states that yield a maximal combination for the probabilities in all the counterparts (P_π, P_μ and P_τ), typically through the Viterbi [11] algorithm. The probabilities in P_π promote some chords as better solutions to begin the path of chords: the ones that are more often used in the beginning of pieces in the dataset. Similarly, the probabilities in P_τ advance solutions that are more often met as concluding chords. Although the results reported in past works indicate that P_π and P_τ most probably create satisfactory results, these probabilities do not *guarantee* that the more often met beginning and ending chords will be utilized. A similar comment can be made about strategies which focus on constructing satisfactory cadences, by beginning from the end of the phrase to be harmonized and employing the Viterbi algorithm from "right-to-left". Specifically, while the latter approaches have an increased bias towards the cadence part of the phrase, it is again not guaranteed that the cadence or the beginning chord of the phrase will be satisfactory.

Regarding the probabilistic scheme, the process for computing the probability value in Equation 33, incorporates the extraction of the statistical values for $\pi(C_1 = c_1)$ and $\tau(C_T = c_T)$, according to the number of occurrences of each chord as an initial or final chord respectively. For the BCHMM approach however, no statistics are considered for these boundary points, since they *certainly* (with probability 1) include the chords specified by a higher hierarchical level or by a human annotator. To be compatible with the terminology followed hitherto for the presentation of the HMM model, the latter comment can be expressed by modifying the Equations 34 and 36 so that they indicate the chords selected at temporary boundary points between successive checkpoints as certain, while eliminating the probabilities for any other chords to appear. Specifically, if the beginning and ending chords are selected to be α_1 and α_T respectively, the new probabilities that substitute the ones expressed by Equations 34 and 36 are the respective following ones:

$$P'_\pi = \begin{cases} 1, \text{ if } C_1 = \alpha_1 \\ 0, \text{ otherwise} \end{cases} \tag{37}$$

[8]In implementations of HMMs, it is usually the negative log–likelihood that is being minimized, i.e. the logarithm of the expression in Equation 33, since the numbers that are yielded by consecutive multiplications of probabilities (quantities ≤ 0) are difficult to be compared by eye because of their small magnitude.

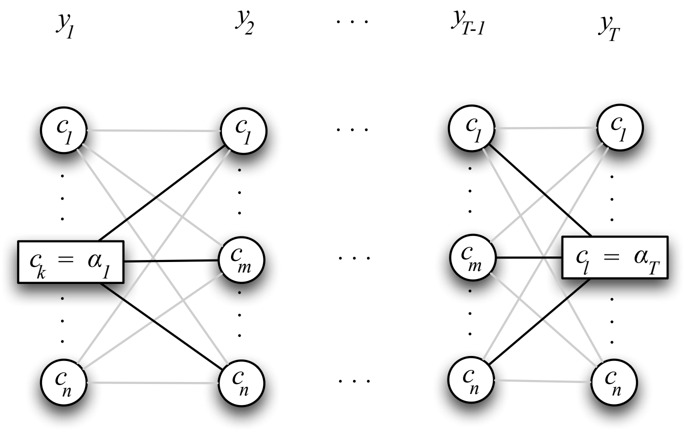


Figure 9: Trellis diagram for the BCHMM. Only transitions from α_1 and to α_T as first and last states respectively are permitted. The intermediate trellis diagram is the same as in a typical HMM.

$$P'_{\tau} = \begin{cases} 1, \text{ if } C_T = \alpha_T \\ 0, \text{ otherwise.} \end{cases} \tag{38}$$

The probability that is therefore optimized is the following:

$$P(C_i = c_i | Y_i = y_i) = P'_{\pi}\ P_{\mu}\ P'_{\tau}, \tag{39}$$

where the factor P_{μ} is the one defined in Equation 35. The employment of the Viterbi algorithm under the constraints imposed by the boundary conditions, as reflected by Equations 37 and 38, assigns zero–value probabilities to all paths, except the ones that begin with α_1 and end with α_T. Figure 9 illustrates the trellis diagram of the Viterbi algorithm under the discussed constraints.

3.1.3 Feature Structures

Idea from Linguistics Feature structures have a long history in linguistics (c.f. [7] for Lexical Functional Grammar and [29] for Head-Driven Phrase Structure Grammar). The classical representation is in form of matrix-like structures. We informally introduce a simple example from linguistics taken from [35]. Consider the German sentence “Ich habe jetzt eine Unfall gesehen” (“Now I have seen an accident”). The sentence is grammatically almost correct except for the agreement between the determiner and the noun in the object NP. “gesehen” calls for an Accusative object, however “Unfall” is Accusative Masculin whereas “eine” is Accusative Feminin. Figure 10 shows the (simplified) corresponding lexical information for “eine” and “Unfall” in a feature structure representation (more precisely a LFG-style representation).

In order to detect the ungrammatical constituent in the above sentence a system needs only to check the feature agreement for those features that need to correspond in a nominal phrase consisting of an indefinite article and a noun. In German these features for which agreement needs to be guaranteed are *gender, number*, and *case* (cf. Figure 10). In case of feature *gender* there is a

$$\begin{array}{cc} \text{“eine”} & \text{“Unfall”} \\ \begin{bmatrix} def: & -- \\ gen: & f \\ num: & sg \\ case: & acc \end{bmatrix} & \begin{bmatrix} pred: & ACCIDENT \\ gen: & m \\ num: & sg \\ case: & acc \end{bmatrix} \end{array}$$

Figure 10: Lexical entries for “eine” and “Unfall”

clash of the values. The possibility to embed smaller feature structures into larger feature structures allows it to uniformly represent words, constituents, and sentences in the sketched matrix-like form. Such matrix-like structures can also be used to represent chords and cord progressions in the music domain.

Formal Basis Feature logic is a well-developed theory. It is not possible to introduce feature logic formally in this report in detail. The reader is referred to [40] for a thorough formal introduction. Just to give the reader a flavor of the formal foundation of feature logic we sketch some basic ideas.[9] We assume that a finite set *Con* of constants (in music e.g. values for a mode like *major* or values of a key like *C#*), a finite set *Feat* of features (e.g. *key, mode, function* etc.), and an infinite set *Var* of variables are given. The denotation of symbols is defined as usual: constants are denoted by letters $a,b,c,\ldots$, features by letters $f,g,h,\ldots$, and variables by letters $x,y,z,\ldots$ The following definition specifies a feature algebra.

Definition 2 *A feature algebra $\mathfrak{A}$ is a pair $\langle D,I\rangle$ where D is a non-empty set and I is an interpretation function defined on constants by $I : Con \to D$ and on features by $I : Feat \to \mathfrak{P}(D\times D)$ such that the following conditions hold:*

(i) *If $\langle a,b\rangle \in I(f)$ and $\langle a,c\rangle \in I(f)$ then $b = c$*
(ii) *If $I(a) = I(b)$ then $a = b$*
(iii) *For all $a \in Con$ there is no $d \in D$ such that $\langle a,d\rangle \in I(f)$*

Condition (i) requires that features are (partial) functions, condition (ii) expresses the unique name assumption for constants, and condition (iii) does not allow the application of features to constants, i.e. constants are considered to be atomic.

Besides the representation of feature structures in matrices it is quite common to represent feature structures as graphs.

Definition 3 *A feature graph is either a graph without edges, i.e. a pair $\langle a,\emptyset\rangle$ where $a \in Con$ is the root of the graph or a graph with edges $\langle x,E\rangle$ where $x \in Var$ is the root of the graph and E is a finite set of (feature–labeled) edges of the form yfs where $y \in Var$, $f \in Feat$, and $s \in Con \cup Var$ such that the following three conditions are satisfied:*

(i) *Edges are uniquely defined: If $yfs \in E$ and $yft \in E$ then $s = t$.*

[9]The following presentation is taken from [35] which is itself based on [40].

(ii) *The graph is connected: If $yfs \in E$ then E contains a path of edges from the root x to y.*
(iii) *The graph is acyclic: If $yfs \in E$ then there is no path of edges leading from s to y.*

Definition 3 requires that a feature graph is a rooted, connected, acyclic, and directed graph.[10] The edges of a feature graph are labeled with features and the nodes with variables or constants. The set of edges E needs to be finite. Subgraphs of a given graph G are defined as usual, namely as a homomorphic embedding of the subgraph into the matrix graph. Definition 4 makes this idea precise.

Definition 4 *Given two graphs G and G' where the root of G is $x \in Con \cup Var$, G is called a subgraph of G' iff there is a homomorphic embedding $h : G \to G'$ such that: $h(x) = x$ and for all edges $yfs \in G : h(yfs) = yfs$.*

Following [40], the collection of all feature graphs can be used to define a new structure called a feature graph algebra. It turns out that this feature graph algebra is itself a feature algebra (compare Fact 6).

Definition 5 *Assume a feature algebra $\mathfrak{A} = \langle D, I \rangle$ is given. A feature graph algebra $\mathscr{F}$ is a pair $\langle \mathbf{D}_{\mathfrak{A}}, I_{\mathfrak{A}} \rangle$ such that:*

(i) *$\mathbf{D}_{\mathfrak{A}}$ is the set of all feature graphs*
(ii) *For $a \in Con$: $I_{\mathfrak{A}}(a) = \langle a, \emptyset \rangle$*
(iii) *$I_{\mathfrak{A}}(f) = \langle G', G \rangle$ iff $xfs \in G'$ where x is the root of G' and G is the maximal subgraph with root s in G'*

Fact 6 **(Smolka)** *A feature graph algebra satisfies the properties of a feature algebra.*

Proof: Compare [40] q.e.d.

Usually feature graphs are considered to be preordered by a subsumption relation $\sqsubseteq$ determining the amount of information coded in the feature graph. The intuition of Definition 7 is to make the concepts of generality and specificity precise in the following sense: if $a \sqsubseteq b$ then a is more general than b (or b is more specific than a).[11] The importance of the subsumption relation $\sqsubseteq$ can be traced back to the fact that $\sqsubseteq$ is used for defining the unification operation on feature graphs. In the framework of amalgams (see below), the amount of information coded in a a feature structure becomes important. Definition 7 specifies a subsumption relation $\sqsubseteq$ on a feature algebra $\mathfrak{A}$.

Definition 7 *Assume $\mathfrak{A} = \langle D, I \rangle$ is a feature algebra. A subsumption preorder $\sqsubseteq$ on $\mathfrak{A}$ is a preorder defined on D satisfying the following property: $a \sqsubseteq b$ iff there is a partial mapping $\phi : D \to D$ such that the following conditions (i) – (iii) hold:*[12]

[10] Alternative definitions of feature graphs allow cyclic graphs as well, compare for example Smolka [40].

[11] In machine learning terms, $A \sqsubseteq B$ means that A is more general than B, while in description logics it has the opposite meaning, since it is seen as "set inclusion" of their interpretations.

[12] According to Definition 2 a feature f is interpreted as a (partial) function. We write $(I(f))(d)$ as the result of mapping $d \in D$ under $I(f)$ in the feature algebra $\mathfrak{A}$.

(i) *For all $c \in Con$ it holds $\phi(I(c)) = I(c)$*
(ii) *If $f \in Feat$, $d \in D$, and $(I(f))(d)$ as well as $\phi(d)$ are defined, then $\phi[(I(f))(d)]$ and $(I(f))(\phi(d))$ are defined such that $(I(f))(\phi(d)) = \phi[(I(f))(d)]$*
(iii) *$\phi(a) = b$*

An Instantiation of the Formal Basis A concrete instantiation of the abstract theory of feature structures are amalgams. An amalgam is a description that combines parts of two other descriptions as a new coherent whole. There are notions that are related to amalgams in addition to conceptual blending, such as merging operation or information fusion. They all have in common that they deal with combining information from more than one "source" into a new integrated and coherent whole; their differences reside on the assumptions they make on the source characteristics and the way in which the combination of the sources takes place.

The notion of amalgams was developed in the context of Case-based Reasoning (CBR) [27], where new problems are solved based on previously solved problems (or cases, residing on a case base). Solving a new problem often requires more than one case from the case base, so their content has to be combined in some way to solve the new problem. The notion of amalgam of two cases (two descriptions of problems and their solutions, or situations and their outcomes) is a proposal to formalize this combinatorial process which produces a new, coherent case.

Formally, the notion of amalgams can be defined in any representation language $\mathscr{L}$ for which a subsumption relation $\sqsubseteq$ between the terms (or descriptions) of $\mathscr{L}$ can be defined. We say that a term ψ_1 subsumes another term ψ_2 ($\psi_1 \sqsubseteq \psi_2$) when ψ_1 is more general than (or equal) to ψ_2. Additionally, we assume that $\mathscr{L}$ contains the infimum element $\perp$ (or "any"), and the supremum element $\top$ (or "none") with respect to the subsumption order.

Next, for any two terms ψ_1 and ψ_2 we can define their *unification*, ($\psi_1 \sqcup \psi_2$), which is the *most general specialization* of two given terms, and their *anti-unification*, defined as the *least general generalization* of two terms, representing the most specific term that subsumes both. Intuitively, a unifier (if it exists) is a term that has all the information in both the original terms, and an anti-unifier is a term that contains only all that is common between two terms. Also, notice that, depending on $\mathscr{L}$, anti-unifier and unifier might be unique or not.

The notion of *amalgam* can be conceived of as a generalization of the notion of unification over terms. The unification of two terms (or descriptions) ψ_a and ψ_b is a new term $\phi = \psi_a \sqcup \psi_b$, called unifier. All that is true for ψ_a or ψ_b is also true for ϕ.; e.g. if ψ_a describes "a red vehicle" and ψ_b describes "a German minivan" then their unification yields the description "a red German minivan." Two terms are not unifiable when they possess contradictory information; for instance "a red French vehicle" is not unifiable with "a blue German minivan". The strict definition of unification means that any two descriptions with only one item with contradictory information cannot be unified.

An *amalgam* of two terms (or descriptions) is a new term that contains *parts from these two terms*. For instance, an amalgam of "a red French vehicle" and "a blue German minivan" is "a red German minivan"; clearly there are always multiple possibilities for amalgams, since "a blue French minivan" is another example of an amalgam. The notion of amalgam, as a form of "partial unification", was formally introduced in [27].

For the purposes of this deliverable, we will introduce a few necessary concepts.

Definition 6 (Amalgam) *The set of* amalgams *of two terms ψ_a and ψ_b is the set of terms such that:*

$$\psi_a \curlyvee \psi_b = \{\phi \in \mathscr{L} | \exists \alpha_a, \alpha_b \in \mathscr{L} : \psi_a \sqcap \psi_b \sqsubseteq \alpha_a \sqsubseteq \psi_a \ \wedge \ \psi_a \sqcap \psi_b \sqsubseteq \alpha_b \sqsubseteq \psi_b \ \wedge \ \phi = \alpha_a \sqcup \alpha_b\}$$

and $\phi \neq \top$ (i.e. $\alpha_a \sqcup \alpha_b$ has a unifer, is not inconsistent).

Thus, an amalgam of two terms ψ_a and ψ_b is a term that has been formed by unifying two generalizations α_a and α_b such that $\psi_a \sqcap \psi_b \sqsubseteq \alpha_a \sqsubseteq \psi_a$ and $\psi_a \sqcap \psi_b \sqsubseteq \alpha_b \sqsubseteq \psi_b$ —i.e. an amalgam is a term resulting from combining some of the information in ψ_a with some of the information from ψ_b. Notice that not all generalizations are taken into account, only those that are less general than $\psi_a \sqcap \psi_b$ (the anti-unification of the inputs); the intuitive reason is that the anti-unification represents what is common or shared between the two inputs and, thus, generalizing more than $\psi_a \sqcap \psi_b$ would eliminate information that is already in both inputs and is compatible. Formally, $\psi_a \curlyvee \psi_b$ denotes the set of all possible amalgams; however, whenever it does not lead to confusion, we will use $\psi_a \curlyvee \psi_b$ to denote one specific amalgam of ψ_a and ψ_b.

The terms α_a and α_b are called the *transfers* or *constituents* of an amalgam $\psi_a \curlyvee \psi_b$. α_a represents all the information from ψ_a which is *transferred* to the amalgam, and α_b is all the information from ψ_b which is transferred into the amalgam.

Usually we are interested only in maximal amalgams of two input terms, i.e. those amalgams that contain maximal parts of their inputs that can be unified into a new coherent description. Formally, an amalgam $\phi \in \psi_a \curlyvee \psi_b$ is maximal if there is no $\phi' \in \psi_a \curlyvee \psi_b$ such that $\phi \sqsubset \phi'$. In other words, if more properties of an input were added the combination would be no longer consistent. The reason why we are interested in maximal amalgams is very simple: consider an amalgam ϕ'' such that $\phi'' \sqsubset \phi$; clearly ϕ'', being more general than ϕ, has less information than ϕ and thus combines less information from the inputs ψ_a and ψ_b. Since ϕ has more information while being consistent, ϕ'' or any amalgam that is a generalization of ϕ, is trivially derived from ϕ by generalization.

Amalgams and conceptual blends are similar in that they combine (parts of) two inputs into a new coherent whole. They diverge however on the assumptions they make about those inputs. Amalgams approach the problem of combining inputs (e.g. ψ_a and ψ_b) from the viewpoint of Artificial Intelligence, assuming some representation language is used to specify these inputs (that an be understood as cases, problems, situations, depending on the context). Although amalgams make very few demands on the representation language (only that some kind of subsumption/generalization relation can be defined among the formulas of the language), they take a very syntactic approach on what the inputs are and what consistency is (since consistency is itself implicitly defined by the way a given language defines unification).

Feature Structures for Coding Chords Feature structures are rather flexible to represent different abstraction levels of music. Whereas features like key, function, mode etc. need to be considered for representing chords, abstract features like mood with values happy and sad can be used for a meta-level description of a chord, a chord sequence, or a whole piece of music. A crucial aspect of modeling musical aspects with feature structures is therefore the choice of the features for representation. Whereas for particular applications COINVENT uses different sets of features to make things simple and to keep as much flexibility as possible, in the following, a possible data structure for the representation of single chords is considered. In order to keep things as simple as possible only tonal chords are considered (i.e. free tonality, atonality, polytonality etc. are not modeled). Intuitively, chords are used that can be interpreted in function theory. Furthermore, we do not stick to a definite set of features, but exemplify the relevant features in some examples. Natural candidates for features used to represent chords are the following ones:

- *key:* Every chord has a feature root key with values C, C#, D, ..., B.
- *mode:* Every chord is in a mode with values *major* and *minor*.
- *func:* Every chord has a function with values *tonic, dominant, subdominant, tonic parallel, double dominant* etc.
- *6:* In a chord, a sixth can be added (sixte ajoutée) with values 0 (no added 6th) or 1 (added 6th).
- *7:* In a chord, a seventh can be added (seventh cord) with values 0 (no added seventh) or 1 (added seventh).
- *9:* In a chord, a ninth can be added (dominant ninth cord) with values 0 (no added ninth) or 1 (added ninth).
- *11:* In a chord, an eleventh can be added (dominant eleventh cord) with values 0 (no added eleventh) or 1 (added eleventh).
- *13:* In a chord, a thirteenth can be added (dominant thirteenth cord) with values 0 (no added thirteenth) or 1 (added thirteenth).
- *sharp:* In a chord several tones can be sharpened. For example, by sharpening the seventh in a dominant seventh cord one gets a major seventh chord, by sharpening the fifth in a dominant seventh chord one gets an augmented seventh chord etc. Values for this feature are subsets of 5#, 7#, 9#, 11#, 13#.
- *flat:* In a chord several tones can be flattened. For example, by flattening the fifth and the seventh in a minor seventh cord one gets a diminished seventh chord, by flattening only the fifth in a minor seventh chord one gets an half-diminished seventh chord etc. Values for this feature are subsets of 5♭, 7♭, 9♭, 11♭, 13♭.
- *Inversion* Dependent on the bass tone in a given chord inversions of this chord can be generated. Values for this feature are 0 (no inversion), 1 (first inversion), 2 (second inversion), 3 (third inversion in case of a seventh chord).
- *add:* In certain musical traditions, composers depart from the arrangement of layers of thirds to generate chords by adding additional tones, e.g. seconds, or fourth. The values for this feature are the keys C, C#, D, ..., B.

In a matrix-like representation, the mentioned features can be represented in a feature structure as depicted in Figure 11 (using flat feature structures). The flat feature representation has the advantage that the resulting structure is rather homogeneous and does not contain deeper embedded feature-value pairs. Nevertheless, several features are dependent on each other for which an

$$\left[\begin{array}{cc}
key: & [C \mid C\# \mid D \mid \ldots \mid B]. \\
mode: & [major \mid minor] \\
func: & [T \mid D \mid S \mid Tp \mid Sp \mid Dp \mid DD \mid SS \mid Dcp \mid t \mid s \mid d \mid tP \mid sP \mid dP\ldots] \\
6: & [0 \mid 1] \\
7: & [0 \mid 1] \\
9: & [0 \mid 1] \\
11: & [0 \mid 1] \\
13: & [0 \mid 1] \\
sharp: & [5\# \mid 7\# \mid 9\# \mid 11\# \mid 13\#] \\
flat: & [5\flat \mid 7\flat \mid 9\flat \mid 11\flat \mid 13\flat] \\
inversion: & [0 \mid 1 \mid 2 \mid 3] \\
add: & [C \mid C\# \mid D \mid \ldots \mid B]
\end{array}\right]$$

Figure 11: A flat feature structure that allows to represent a large variety of tonal music. The *mode* feature is interpreted as the natural minor scale. Notice that functions are dependent on the mode of the scale (i.e. major or minor): For example, a dominant parallel chord (Dp) of a major scale is a minor chord, whereas the dominant parallel (dP) of a minor scale is a major chord. Notice also that there are obvious dependencies between the features, e.g. functions are dependent on the given scale, the sharpening of a seventh is dependent on the value of the feature 7 (if there is no seventh, it cannot be sharpened) etc.

embedded structure, where certain features are collapsed to a more complex feature, is a more natural representation. Figure 12 depicts a possible mild form of a feature structure where certain dependencies are representable in the feature structure. Notice that even here some features are not independent from each other: for example, in a C-major chord, adding with the *add* feature the tone E is redundant, because the tone is already coded in the chord.

Representing chords as feature structures has the advantage that generalizations (i.e. the generic space in the blending model) and blends can be computed rather easily. In the context of single chords, a generalization of two chords can be computed by the HDTP engine by introducing variables for those values of features where there is a disagreement. On the other hand, features of two chords where the values coincide can be copied to the generic space.

Here is a simple exemplification: Consider the example of blending two chords. Take as one input a dominant seventh chord in the second inversion (e.g. in the C-major scale: D - F - G - B) and as the other input chord a double subdominant chord without inversion and without seventh (e.g. in the C-major scale: B♭ - D - F). The task is to compute a generalization and a possible blend chord for the input. Figure 13 shows the corresponding diagram, namely the two input chords, the computed generalization, and a possible blend chord. In the depicted diagram we abstract from many features that are of no interest for the example. The computation of the generalization (i.e. the generic space) is straightforward by introducing variables x, y, and z for corresponding features with different values (compare the computation of generalizations by HDTP in Subsection 2.2.2). The blend space allows various variations of blending the two input chords. Heuristics need to govern the more appropriate candidates for blended chords.

$$\begin{bmatrix} key: & [C \mid C\# \mid D \mid \ldots \mid B] \\ mode-function: & \alpha \begin{bmatrix} major: [T \mid S \mid D \mid Tp \mid Sp \mid Dp \mid DD \mid SS \ldots] \\ minor: [t \mid s \mid d \mid tP \mid sP \mid dP \mid D \mid DD \ldots] \end{bmatrix} \\ 6: & [0 \mid 1] \\ 7: & [0 \mid 1 \mid 7\flat \mid 7\#] \\ 9: & [0 \mid 1 \mid 9\flat \mid 9\#] \\ 11: & [0 \mid 1 \mid 11\flat \mid 11\#] \\ 13: & [0 \mid 1 \mid 13\flat \mid 13\#] \\ sharp: & [5\#] \\ flat: & [5\flat] \\ inversion: & [0 \mid 1 \mid 2 \mid 3] \\ add: & [C \mid C\# \mid D \mid \ldots \mid B] \end{bmatrix}$$

Figure 12: A feature structure that allows to represent certain dependencies between features. The *mode-function* feature is interpreted as functions relative to the natural minor scale.

$$\begin{tikzcd}
 & \begin{bmatrix} mode: & [major] \\ func: & [x] \\ 7: & [y] \\ inversion: & [z] \end{bmatrix} \arrow[dl] \arrow[dr] & \\
\begin{bmatrix} mode: & [major] \\ func: & [D] \\ 7: & [1] \\ inversion: & [2] \end{bmatrix} \arrow[dr] & & \begin{bmatrix} mode: & [major] \\ func: & [SS] \\ 7: & [0] \\ inversion: & [0] \end{bmatrix} \arrow[dl] \\
 & \begin{bmatrix} mode: & [major] \\ func: & [SS] \\ 7: & [1] \\ inversion: & [2] \end{bmatrix} &
\end{tikzcd}$$

Figure 13: Example of a generalization and blend of two chords. Only some interesting features are depicted.

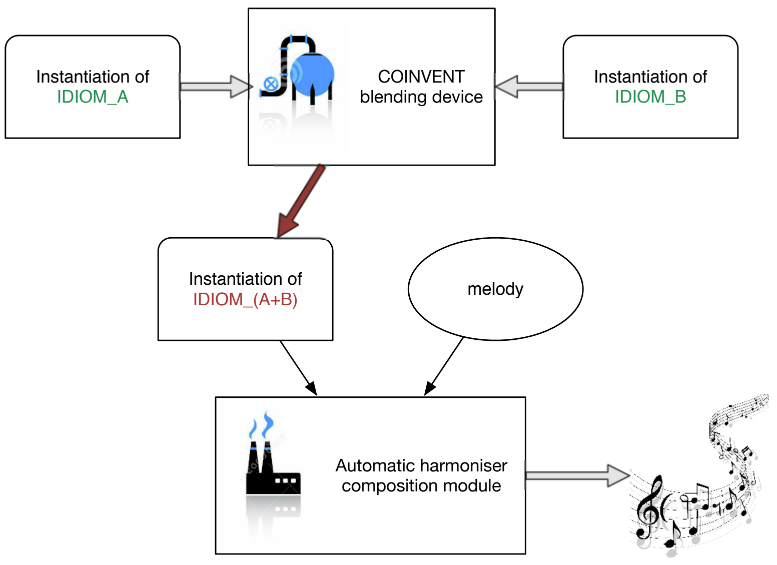


Figure 14: Overview illustration of the COINVENT melodic harmonizer. Conceptual blending takes part in the ontologies that describe harmony, creating blended harmonizations that constitute the descriptive guidelines for the composition module.

3.2 Concept Blending in Harmony for Concept Invention: Overall Architecture and Information Extraction from Score

The development of a melodic harmonizer is designed to encompass two main parts: the *blending* and the *harmonization* modules. In order to give an overall picture of the approach, an architecture is needed that specifies the information flow of the involved modules. The architecture is roughly illustrated in Figure 14. The blending module incorporates all the background knowledge in the form of a harmonic ontology that represents different idioms, as well as certain pieces of each idiom. Harmonic blending concerns the information included in this module. The specific properties of this module are analyzed in Section 3.3. The harmonization module receives information from the blending module, thereby building a harmonization framework of deterministically produced chord constraints.

The architecture requires information in form of a harmonic ontology representing different idioms as well as statistical information about chord sequences. For these two types of information harmonic information needs to be extracted from scores. Ground-truth harmonic information is manually annotated and extracted from pieces of music among a collection of great variety, the *harmonic training* dataset (compare Deliverable D7.1). Manual annotations concern the extraction of key-elements in harmony, from a harmonically reduced version of the examined pieces' phrases. These elements include the required information to describe harmonic concepts on many levels, with a parallel goal to achieve a balance between human-friendly interpretation of data and computationally accessible encoding. Specifically, the harmonic levels that are isolated and represented are the following ones:

1. *Tonality* and tonality changes, which are important for tracking the tonality constitution of

phrases within idioms (e.g. modal or tonal harmony), while also determining the harmonic changes that occur and how often they occur.

2. *Grouping* of phrases for tracing the sub-phrases that include harmonically and melodically coherent parts.

3. Denotation of *harmonic devices*, for isolating areas of harmonically exceptional mechanisms.

The points mentioned above are analytically presented in Section 3.3. On the other hand, building a rich background database with a plethora of different idioms demands the employment of a chord representation formalism that is not limited to representing a subset of the utilized idioms. For instance, the utilization of the alphabetic jazz chord symbols might be adequate for representing jazz, tonal or modal music idioms (among others), however, it imposes serious limitations for the representation of atonal music or for non-Western folk idioms (e.g. the polyphonic songs of Epirus). On the other extreme, "mathematically" inspired representation techniques, like the pitch class sets, can represent any pitch class simultaneity; however, they do not provide any perceptual meaning to the represented harmonic entities. For an idiom-independent representation of automatically extracted information from scores the General Chord Type representation is used (compare Subsection 3.1.1). As mentioned in the respective Subsection the GCT algorithm encodes most chord types correctly. For more information concerning the harmonic training data set the reader is referred to Deliverable D7.1.

3.3 Concept Blending in Harmony for Concept Invention: Representation for the Blending Framework and Worked Examples

Conceptual blending takes place on three levels:

1. *Meta-level* of harmony: This level discusses blending on some aspects of harmony that are described either as general properties of harmony (e.g. the mood) or as harmonic parts that adhere to specific conceptual characteristics (e.g. chromatic voice leading of a particular voice in a chord sequence).

2. *Phrase structure*: Phrase structure refers to the harmonic blocks that succeed one another, including cadences or the ordering of parts that belong to harmonic devices (or harmonic device combinations).

3. *Chord level*: Blending in the chord level yields on the one hand novel chords that preserve some crucial harmonic elements, regarding the relations between notes of successive chords (see the cadence blending example in Section 3.3.1). On the other hand, chord blending allows the invention of intermediate chords for consecutive harmonic parts that belong to different conceptual spaces, providing common-ground solutions for "incompatible" consecutive parts between different CHMMs (see Section 3.3.3).

Additionally, there is a key algorithmic process that has been developed in the context of COINVENT that allows the idiom–dependent probabilistic harmonization of different parts, preserving the harmonic characteristics of a selected idiom. This technique is based on the well-known hidden

Markov models (HMMs, cf. Subsection 3.1.2), with the vital difference that they allow the deterministic insertion of intermediate chords, thus allowing the functionality of the hitherto described blending framework.

3.3.1 Blending on the chord level

A part of the harmonic character that is exposed by a chord transition is due to the underlying relations that exist between some of the notes that compose these chords. A characteristic example that is indicative of this statement, concerns the utilization of the pitch classes in the perfect cadence, which is a harmonic device comprised of two chords (namely the pre-final and final chord) commonly met in tonal music. This transition draws its unique character by the fact that it defines the tonality of a piece, for reasons that are musically explainable, but are beyond the scope of this report. Since this transition suggests a stable tonal environment by strongly implying a tonality (either the piece's or an intermediate tonality), this musical device is utilized as a "cadence", i.e. the ending part of a phrase. What is most characteristic about this cadence, is the third of the pre-final chord that constitutes the leading note to the tonic. There are some additional inter-chord note relations that are also of great importance, e.g. the seventh of the pre-final chord (if we consider the pre-final chord to be a dominant seventh) that leads to the third of the final chord, while some other relations are weak, e.g. the fifth of the pre-final chord could be omitted. The above mentioned example referred to the relations between notes of successive chords that comprise a cadence, while similar information could be utilized to describe relations for any chord transition – not necessarily the final transition (cadence).

The rationale behind blending in the chord transition level is to take advantage of characteristic relations that exist between the involved chords of two *input* transitions, creating novel transitions that encompass the characteristics of both inputs. To this end, to efficiently perform such a blending, a representation scheme is needed that expresses:

1. the relations between *important* notes in each chord,
2. the *important* relations between note successions from the first to the second chord and
3. consistency facts about each input or blendoid space (a fact that is required for performing *consistent* blending, i.e. blendoids that do not include contradictions).

These points will be further examined in the subsequent paragraphs of this section, under the scope of specific examples. The presented examples discuss blending between two input spaces that include typical cadences of different musical idioms: from the *tonal* idiom the *perfect* cadence with a dominant seventh pre-final chord and from the *modal* idiom the *phrygian* cadence. The reason for selecting cadences as examples is that the cadences comprise robust and distinguishing harmonic devices of the idiom they represent, as well as they are by themselves a subject of particular and thorough musicological examination. Figure 15 illustrates examples of cadence, while the two cadences that have been discussed above are the ones depicted as second and fourth.

There are many relations that shape the character of these cadences, the most important of which will be analyzed later in the text. At this point, a simplified version of the cadences is presented, that encompasses only the necessary information for yielding two novel cadences through

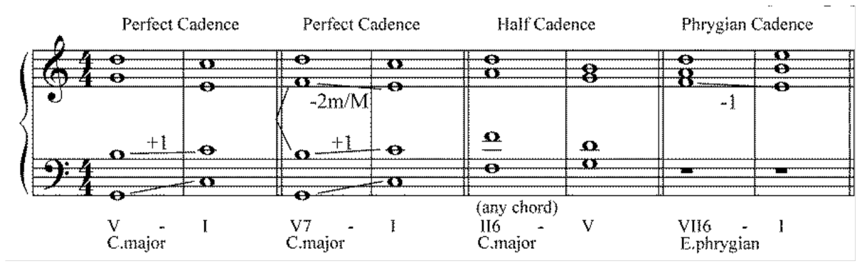


Figure 15: Examples of cadences. The ones involved in subsequent.

conceptual blending. The graphical interpretation of these simplified versions is given in Figure 16 (a) and (b). These cadences are composed of two chords, namely a pre-final and final chord. These chords are considered to have a quality (e.g. major, minor etc.), a root note (the chord's fundamental) and a bass note (which indicates the inversion of the chord). Additionally, several relations between the pre-final and final chords' root and bass notes are considered. These relations are illustrated in the lower "squares" of the graphs in Figure 16. Specifically, `m2`, `m7`, `M6`, `P5` and `unison` indicate a minor second, minor seventh, major sixth, perfect fifth and unison respectively. Figure 16 (c) depicts the generic cadence, which is a cadence that includes a generalized version of all the elements that are included in both (perfect 7 and phrygian) cadences. The generalizations between the generalized elements are named as `topREL`, `bottomREL`, `leftREL` and `rightREL`, indicating relations between the pre-final (prefix `PF`) and final (prefix `F`) chords' root and bass notes, as illustrated by the arrows in the discussed figure. The generalized elements include descriptions of the root (suffix `Root`) and the bass (suffix `Bass`) of the pre-final and the final chords.

The objects (classes) and the relations among them (object properties) are structured in a "generalization sequence", meaning that the application of the generalization (γ) and specialization (ρ) operators function on both objects and relations within the cadence ontology, yielding more general or more specialized cadences. For instance, the cadences in Figure 16 (b) and (c) are specializations of the "generic" cadence in Figure 16 (a). The simple subsumption relations that are considered in Figure 16 and in the discussed amalgams/blends are the following ones:

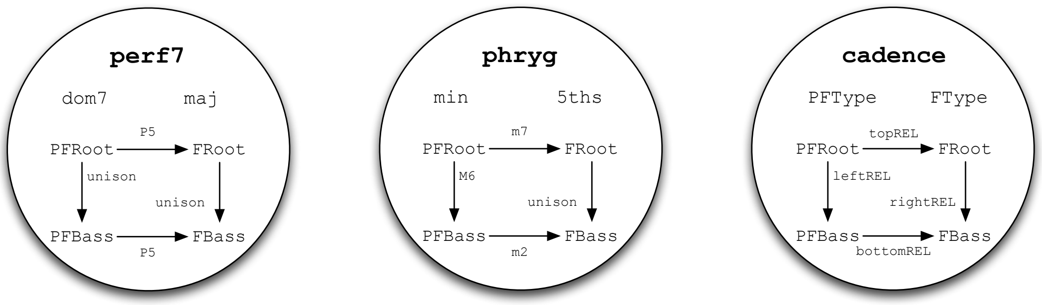
1. `PFType` $\sqsubseteq$ `dom7` or `min`, indicating the general concept of the pre-final chord's type and the specialized types dominant 7-th and minor respectively.

2. `FType` $\sqsubseteq$ `maj` or `5ths`, indicating the general concept of the final chord's type and the specialized types major and 5-ths (root and fifth) respectively.

3. `topREL` $\sqsubseteq$ `P5` or `m7` indicating the general concept of pre-final and final chord root relations and the specific relations of a perfect fifth and a minor seventh interval respectively.

4. Similarly for `bottomREL`, `leftREL` and `rightREL` $\sqsubseteq$ `unison`, `P5`, `m2`, `m7` and `M6`.

As discussed earlier, these relations describe the "square" of relations in a cadence. It is noticed that a square of relations should be consistently assembled, meaning that the relations between every pair of chord elements should not be contradicting. For instance, in the perfect 7 cadence (Figure 16 (a)), the top and bottom relations are the same, a fact that obligates the left



(a) perfect 7 cadence (c) phrygian cadence (c) cadence

Figure 16: Graphical illustration of the formalization followed for representing a cadence (a), a perfect cadence with dominant pre-final chord (b) and a phrygian cadence (c).

and right relations to be the same. It is also noticed that the classes `PFRoot`, `Froot`, `PFBass` and `FBass` are formalized to include no specialization or generalization.

Figure 17 and Figure 18 illustrate the above discussed spaces for two meaningful musical blends that are constructed by the examined ontologies and blending formalizations. Specifically, in both figures the input spaces are named as `perf7` and `phryg`, indicating the concepts of the perfect and phrygian cadences respectively. The intermediate generic spaces of both inputs are called `perf7_gen` and `phryg_gen` respectively. It should be noticed that the intermediate spaces are different in the two presented examples, allowing the desired amalgam to be produced. The generic space is named as `cadence`, which in both examples accommodates the common general information that a cadence incorporates. As illustrated in the discussed figures, the blended spaces, also named as the *colimits* in the category theory nomenclature, produce the `tritone` substitution and the `backdoor` cadences, which are commonly used in jazz music. It should be noted that the information incorporated in the colimits are not only drawn from the input but also from the intermediate spaces (or the generic `cadence` space). This fact allows several inconsistencies to be resolved as discussed in the following paragraph.

3.3.2 Blending on the phrase level of pieces

Grammars constitute an effective methodological approach to represent information that is layered in a hierarchical manner. The structure of a grammar employes rules that describe symbol substitutions from upper to lower hierarchical levels, reaching the lowest level of terminal symbols that represent the grammar's output string. The harmony in phrases of a musical piece or an excerpt present a hierarchically-layered structure that can be efficiently described by grammars. However, the development of harmonic grammars that are accurate enough to capture the basic characteristics of a musical idiom, is a task that requires deep knowledge and ready-made musicological analysis of the harmonic elements and their relation in this idiom. Therefore, the development of musical grammars from scratch is a tedious and time consuming process prone to errors, especially for idioms that are not yet deeply studied.

Previous work on grammars for harmony indicate that harmony in an idiom is a highly specialized characteristic and, therefore, the development of an idiom's grammar requires a deep

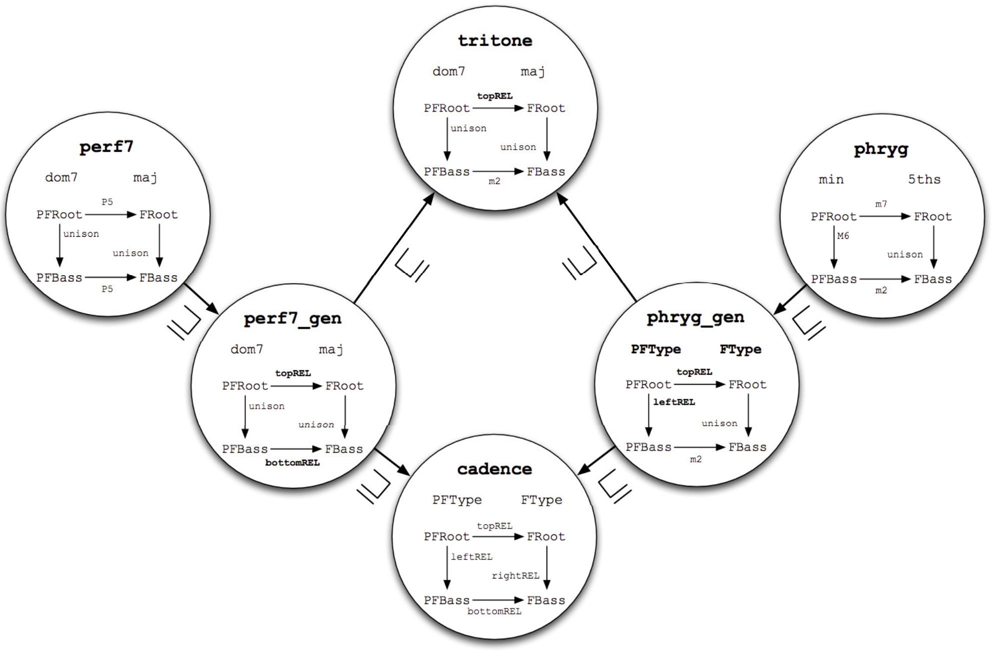


Figure 17: Tritone substitution cadence illustrated according to the core model for conceptual blending.

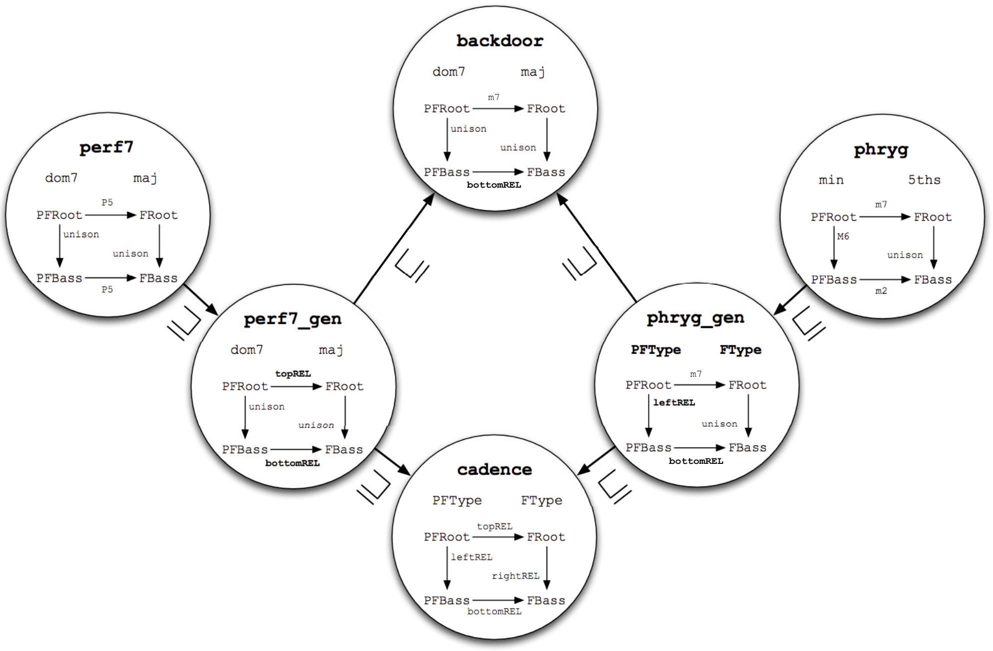


Figure 18: Backdoor cadence illustrated according to the core model for conceptual blending.

musicological analysis of this idiom. For instance, a highly specialized grammar for harmony was presented in [14], which discussed the employment of combinatory categorial grammars to describe the harmony in jazz standards. This grammar allowed to either automatically parse, or generate harmonic parts of the jazz idiom. A large number of rules was included in this grammar, describing highly specialized harmonic devices of the jazz idiom, e.g. special types of cadences like tritone substitution and backdoor cadence. Additionally, the highly specialized nature of harmony is further amplified by the vast complexity of rules that are employed in works that examine the development of grammars for the probably deepest studied musical idiom: the tonal music (e.g. [36, 20]).

Since the presented study incorporates a large dataset of many and diverse idioms, it is practically impossible to hand-craft rules for each studied idiom; this fact is further highlighted by the difficulties to hand-craft rules for tonal or jazz harmony. Therefore, the approach followed in the context of COINVENT includes the development of grammatical rules on a level that can be inferred with automatic means for each piece: the *phrase level*. The phrase level is considered to include general descriptions about the structure of a phrase. A similar approach to describe the phrase level with grammars, has been presented in [36], where rough descriptions of the phrase structure where provided, however again in the context of tonal music. The descriptive elements that are considered for COINVENT are the primary harmonic parts (like ending part and harmony body), as well as special harmonic devices (see Section 3.3.4) that might be incorporated within the primary parts (e.g. harmonic devices that incorporate special voice leading or ready-made chord sets). All these parts will be manually annotated on selected and representative pieces of each idiom on the score, allowing the induction of their grammar by algorithmic means on a later stage.

The basic grammatical rule for constructing a phrase is the following:

$$\mathtt{P} \rightarrow \mathtt{body}_{\text{tonality}=X_2}\mathtt{end}_{\text{tonality}=X_3}, \tag{40}$$

where $\mathtt{P}$ represents the phrase, while $\mathtt{body}_{\text{tonality}=X_2}$ and $\mathtt{end}_{\text{tonality}=X_3}$ represent the phrase's body and ending parts in tonalities X_2 and X_3 respectively – which are not necessarily the same. The primary separation of a phrase into a body and an ending part, as in Equation 40, is based on the distinguishing role that the ending parts (cadences) play in all musical idioms. Different tonalities between different phrase elements are potentially applicable, since in some idioms the cadences prepare a change of key in subsequent phrases.

The $\mathtt{body}_{\text{tonality}=X}$ part is by itself comprised of many potential parts, according to an idiom's special harmonic content. Some examples for expanding the $\mathtt{body}_{\text{tonality}=X}$ part are the following:

$$\begin{aligned} \mathtt{body}_{\text{tonality}=X_1} &\rightarrow \mathtt{body}_{\text{tonality}=X_1}\mathtt{body}_{\text{tonality}=X_2} && (41)\\ &\rightarrow \mathtt{end}_{\text{tonality}=X_1}\mathtt{body}_{\text{tonality}=X_1} && (42)\\ &\rightarrow \mathtt{device}_{\text{tonality}=X_1} && (43)\end{aligned}$$

The expansion rule in Equation 41 indicates a key change within a phrase. Equation 42 represents a harmonic block that includes a cadence before a harmonic new block, describing potential intermediate cadences that occur e.g. before a key change in a phrase. The expansion rule in Equation 43 refers to the utilization of *harmonic devices*, which are described in Section 3.3.4. This rule allows the insertion of special harmonic constructions that include specific harmonic concepts. For instance, the utilization of parallel chromatic chord movement, or the employment of a drone tone in a chord sequence are two harmonic devices. Additionally, the combination of harmonic devices is also realizable, if these devices are not contradicting.

Blending on the phrase level potentially will be examined in the context of *"grammatical blending"*, while its outcome will potentially enable blended phrase structures for different idioms, as well as the combination of characteristic parts (e.g. cadences or harmonic devices) from diverse idioms.

3.3.3 Idiom–dependent probabilistic chord generation

During the last decades, several methodologies have been proposed for the harmonization of a given melody with algorithmic means. Among the most successful are methodologies that incorporate probabilistic mechanisms and statistical learning, since they have the ability to generate harmonies that statistically adhere to the harmonic characteristics of the idiom that the training pieces belong to. In the context of COINVENT, a modified version of the hidden Markov model (HMM) has been examined, namely the *constrained* HMM, which combines the probabilistic HMM framework with additional intermediate fixed–chord constraints. This modified HMM version utilizes the probabilistic idiom-reflecting characteristics, but additionally harnesses any external chord generation processes. For the COINVENT melodic harmonization system, the utilization of this methodology provides on the one hand a robust harmonization basis (the HMM), while on the other hand fixed harmonic constraints referring to other idioms can be injected.

The representation requirements for such a system concern all the algorithmic parts required for the HMM methodology: prior distributions for beginning and ending chords, transition tables of probabilities and observed notes–to–chord selection probabilities (or rules expressed as probabilistic density functions). The CHMM [19] methodology integrates the potential to blend harmonic concepts on different levels than the chord transitions. Such concepts can be programmed by chord generation processes named as the "*harmonic devices*", presented in Section 3.3.4. A future extension of the algorithm will allow the insertion of notes that are "required" to participate in the harmonization, extending the abilities of the system towards generating chord sequences that adhere to voice leading constraints. Although this step appears to be trivial, a number of issues may emerge, involving the absence of chords-states that encompass the desired notes. However, such problems could potentially be resolved by employing concept invention through blending - a consideration that extends conceptual blending from a concept generation mechanism to a solution finding methodology.

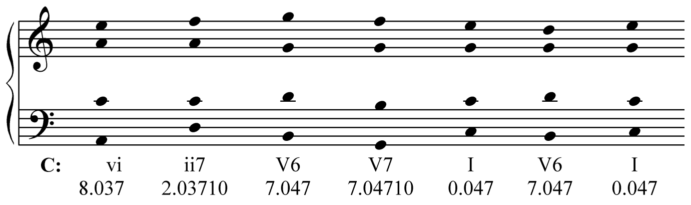
During the "unofficial" evaluation of the presented methodology, several test phrases were harmonized, as well as several anchor point insertion setups were examined. The presented results include some indicative harmonizations that have been produced by the system with different anchor point setups. The utilized dataset comprises a selection of phrases from the "benchmark" chorales of J. S. Bach, specifically some chorales in the major mode.

The experimental process aims to provide indications about the fact that the utilization of the anchor points yields harmonizations that are potentially more "*interesting*" than the ones produced by the typical HMM methodology – depending on the selected anchor points. Therefore, the experimental results expose the ability of the proposed system, as well as the flexibility of the modified HMM scheme towards allowing different – and potentially more interesting – harmonization alternatives, according to the provided anchor points. To this end, the system's evaluation processes mainly address the fact that the proposed methodology is implementable using a relatively small dataset of training pieces.

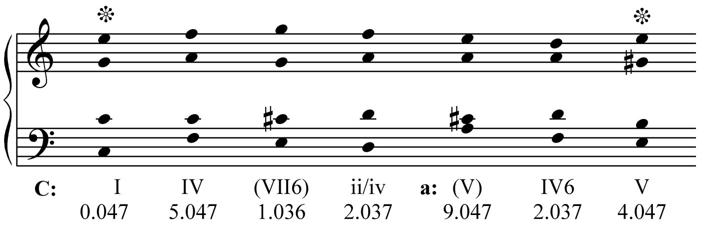
The CHMM methodology in COINVENT addresses the harmonization task within the context of a certain key, thus a full harmonic reduction of phrases is considered as input to the system; the term "phrase" will hereby signify the melody notes and their harmonization, as yielded from the reduction. The phrases of the Bach chorales are divided in two sets according to their key of composition, i.e. in major and minor phrases. Although harmonizations of both modes were tested, the reported results include only major mode phrases. The GCT chords–states that are derived for the major chorales of Bach are 41 and for the minor chorales 38, while many of the major and minor states are overlapping, i.e. exist both in the major and in the minor chorales. Several of these states are redundant since their GCT expression in fact describes chords of the same functionality, e.g. the GCTs [0, $[0,4,7]$, []] and [0, $[0,4]$, []] denote a major chord in the tonic. Additionally, there is a considerable amount of GCT states (around 15 for each mode) that occur only two or three times in the entire dataset. The latter comments indicate that the employment of a GCT clustering technique could group some GCTs according to their harmonic functionality, further reducing the states to approximately 25 for each mode. However, such a grouping methodology is yet to be developed and it is part of ongoing research.

When harmonizing a melody with no constraints, the HMM methodology selects the most probable sequence of chords (hidden states) according to probabilities related to the melody's note to be harmonized and to probabilities related to the transitions between pairs of states. The imposition of fixed–chord constraints is intuitively expected to alter the harmonization "locally", i.e. the CHMM harmonization is expected to be different than the one provided by the typical HMM a few chords before or after a chord that remains fixed – if the selected chord to be fixed is different than the one provided by the HMM. However, the application of chord constraints in some cases provided different harmonizations throughout the entire length of the phrase. The voice leading in the examples presented below was performed by a music expert; an algorithmic process for voice leading is a future research goal. The score examples that are analyzed in the remainder of this section are produced by HMMs or CHMMs that have been trained on the same set of 30 random chorale phrases, which did not include the harmonized phrases.

The example in Figure 19 amplifies the role of anchor chords and specifically the beginning and ending chords of a phrase. In this example, a Bach chorale melody is harmonized with the typical HMM methodology (top) and with anchor boundary (beginning and ending) chords denoted by an asterisk. The boundary chords are the ones utilized by Bach in the genuine chorale. An initial comment concerns the fact that the HMM methodology does not "guarantee" that the beginning and ending (boundary) chords of a melody to be harmonized are identical to the ones that would potentially be utilized by a human composer. Additionally, the role of the boundary chords is crucial: the example in Figure 19 demonstrates that different anchor chords provided an entirely different harmonization. Furthermore, this example shows that the imposition of constraints "forced" the system to follow more "interesting" and unpredictable chord paths, since, the typical HMM methodology utilized more typical and probable chord progressions between V and I chords. The imposition of constraints on the other hand, forced the HMM methodology to establish temporary secondary tonalities, yielding a richer harmonic interpretation of the melodic sequence.



(a) typical HMM



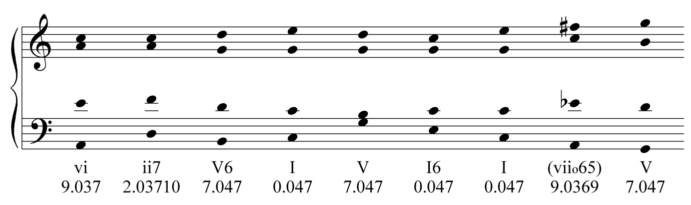
(b) CHMM with boundary anchor chords

Figure 19: (a) The harmonization of a Bach chorale melody with the typical HMM methodology and (b) with constraints on the first and final chords (indicated with an asterisk).

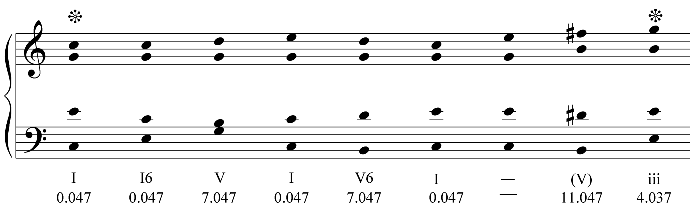
Except from the imposition of boundary chords, the insertion of intermediate chords can also produce interesting results. The example depicted in Figure 20 discusses the harmonization of a Bach chorale in four different versions. Specifically, Figure 20 (a) demonstrates the harmonization produced by the typical HMM methodology, while the harmonization in (b) is produced with constraints on the boundary chords (as indicated by the asterisks). The constraints used in the phrase's boundaries are the ones utilized by Bach in the genuine chorales. The imposition of the boundary constraints does not produce a harmonization that is entirely different regarding the selection of GCT chords (unlike the example shown in Figure 19), however the voice leading that was assigned by the music expert in both phrases is different. The harmonization became more interesting when the music expert indicated the insertion of the diminished chord marked with an asterisk in Figure 20 (c) (fifth chord). This anchor chord changed the harmonization entirely; even when the boundary constraints were alleviated, the harmonization produced by the CHMM system (Figure 20 (d)) was again completely novel. The fact that different constraint conditions produce diverse harmonizations, amplifies the motivation to utilize a "deterministic" chord selection scheme along with the probabilistic HMM framework.

3.3.4 Meta-Level Properties of Music

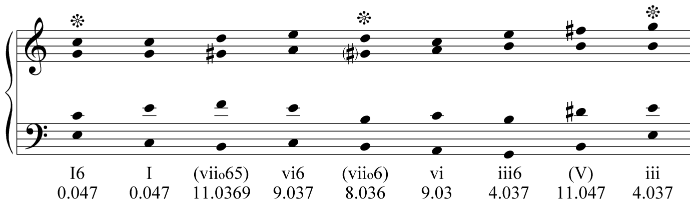
Through the utilization of semantic technologies, the knowledge that is incorporated among large amounts of data becomes potentially understandable both by humans and machine, while at the same time machine reasoning is realizable. Therefore, the development of ontologies that encompass qualitative descriptors of harmony and their interrelations, will allow conceptual blending on a level beyond harmonic objects and structure: at the "*meta*"-level of harmony. Some of these concepts can be expressed according to the emotions or the stylistic references they imply (e.g. happy mood or minimal harmonization). On the other hand, some harmonic concepts are expressed efficiently as specialized "*harmonic devices*", which constitute focused descriptions of specific harmonic content in a musical excerpt, either for vertical or for horizontal harmony (e.g.



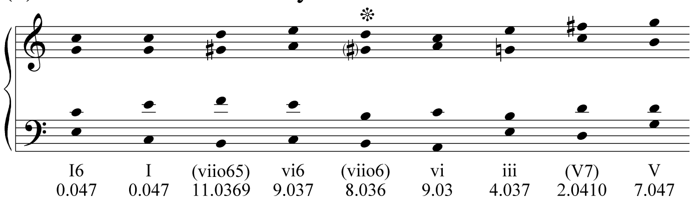
(a) typical HMM



(b) CHMM with boundary anchor chords



(c) CHMM with boundary and intermediate anchor chords



(d) CHMM with an intermediate anchor chord

Figure 20: (a) The harmonization of a Bach chorale melody with the typical HMM methodology and with constraints on (b) the boundary chords, (c) the boundary and one intermediate chord and (d) only one intermediate chord. The fixed intermediate chords selected by a human annotator are indicated on the score with an asterisk.

utilization of chromatic parallel ascending chord movement). Such concepts are incorporated in a semantic context by developing a harmonic ontology that describes relations between them.

The formulation of such a harmonic ontology can be materialized in a manner similar to the one utilized by frameworks that have already been developed for music related tasks. An example of such ontologies is the Music Ontology [32, 34], which hosts information about music publishing on diverse aspects, e.g. from publication and recording dates or instrumentation, to events that might occur at specific time instances within recordings. However, since the discussed harmonic ontology is developed to comprise a set of *composition guidelines* to the underlying melodic harmonization system, its formalization needs to meet more than the *descriptive* standards that the Music Ontology does. For example, if an entry that describes the concept of *mood* in a harmonization is *happy*, then this concept has to map to several related attributes, e.g. prevalent utilization of major chords – which constitutes a form of compositional guideline. Other ontologies that will be

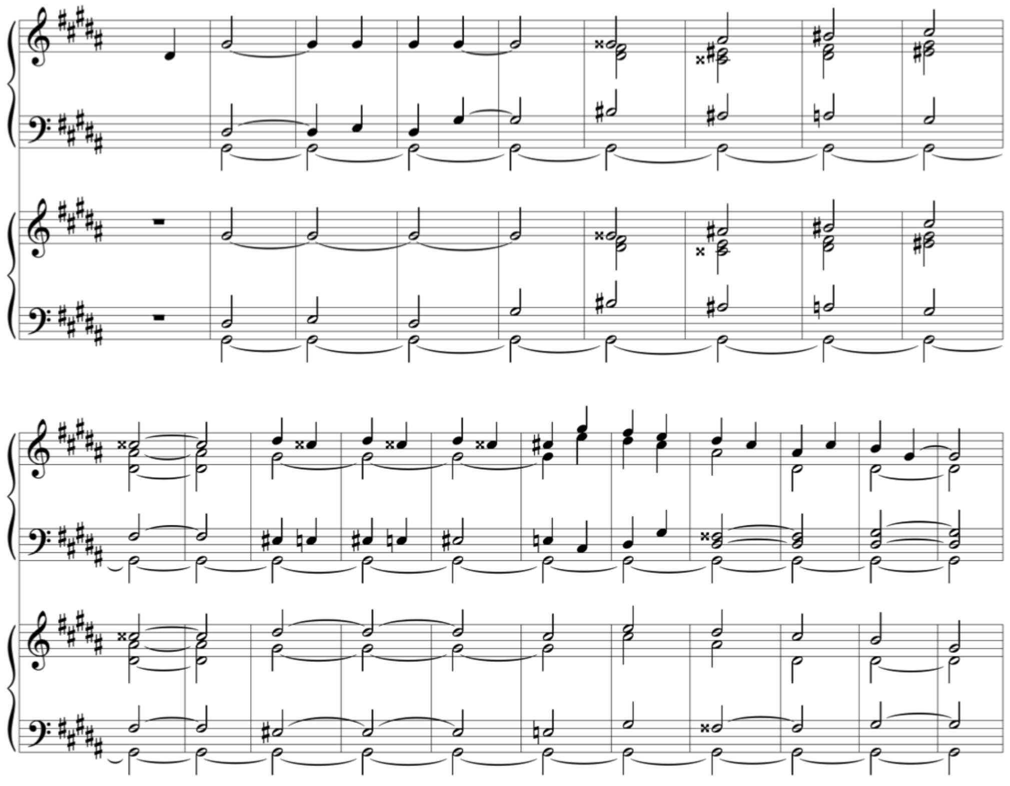


Figure 21: An excerpt of Modest Mussorgsky's "*Il vecchio castello*", where conceptual blending on the harmonic meta-level is realized.

examined as auxiliaries towards the development of the discussed meta-level harmonic ontology, are the Timeline Ontology [33], the Event Ontology [31] and the Chord Ontology [30].

The example examined for describing the meta-level harmonic information concerns an excerpt of Modest Mussorgsky's work entitled "*Il vecchio castello*". In this work Mussorgsky, utilizes blending between a variety of harmonic concepts consolidated in harmonic devices, some of which are combined simultaneously to harmonize parts of the melody.

3.3.5 Blending of Chord Progressions

There is a classical theoretical basis for chord progressions in the tonal system: cadences. In classical music theory, cadences are often used to describe a balance between tensions in a harmonic progression and a resolution of these tensions. Pieces of music can be made more interesting and surprising by using new chord progressions. Figure 22 depicts examples of classical cadences that constitute an important foundation of classical music theory. Here are some properties of such chord progressions:

- It is obvious that the single chords involved in a cadence can be straightforwardly represented using feature structures as described in Subsection 3.1.3.

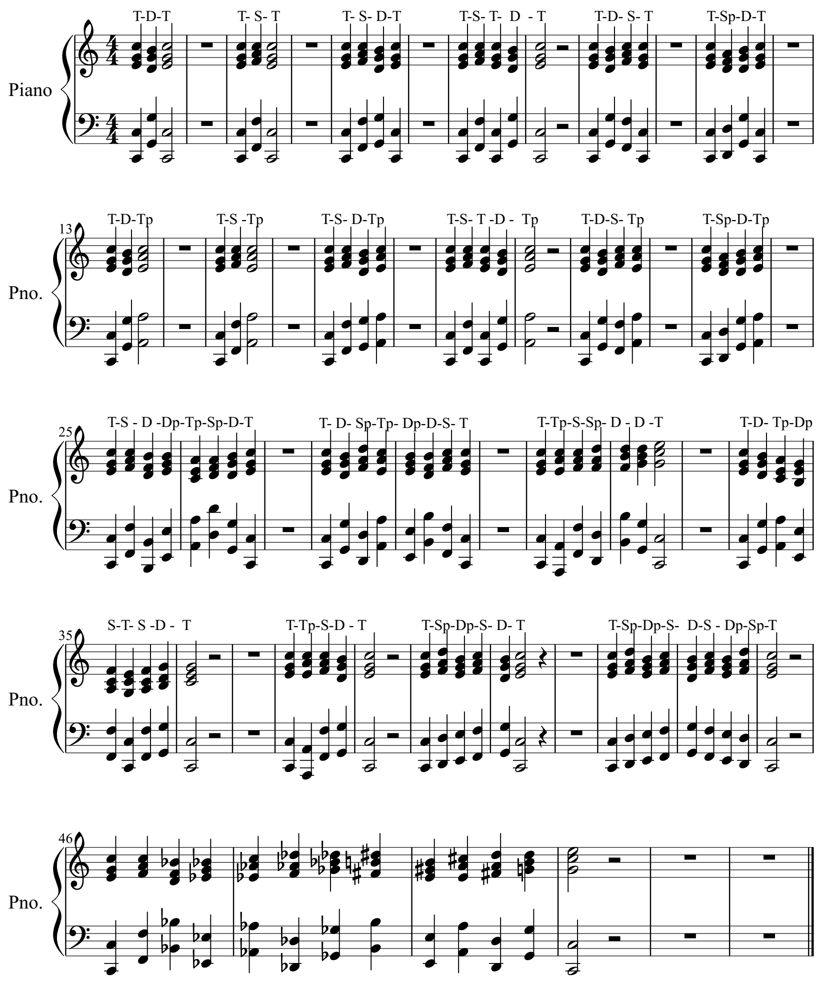


Figure 22: Examples of classical cadences in C major (more generally one could say basic chord progressions in C major). The last cadence is the circle of fifth, where descending fifths (or ascending fourths) constitute the progression. The seventh step in a progression (except the last one) is associated with a dominant chord.

- The basic cadences do not involve features like sevenths, ninths etc.
- Chord progressions occurring in pieces of music are more general than basic cadences and allow in various directions more possibilities for chord progressions.
- Representing a sequence of chords in a feature structure representation requires either the representation of a sequence of features structures or the embedding of feature structures in a matrix feature structure.

Blending chord progressions in this framework can be considered as a blending process of the involved chords of the single sequences in order to achieve a new sequence. It turns out that there are many design choices in a blending process of general chord sequences. In order to keep the complexity of the representation as simple as possible, we exemplify the blending of general chord sequences in non-trivial examples, but we are using an extremely simplified notation of chord sequence, namely as a sequence of functions similar to the presentation in scores of popular music.

Example 1 In the following two input sequences together with the generalization and candidates for blended sequences is listed:

Input 1:	T → Tp → Sp → D7 → T	(in C major: C → a → d → G7 → C)
Input 2:	T → DDD → DD → D → T	(in C major: C → A → D → G → C)
Generalization:	T → x → y → D → T	
Blend 1:	T → Tp → DD → D → T	(in C major: C → a → D → G → C)
Blend 2:	T → DDD → Sp → D7 → T	(in C major: C → A → d → G7 → C)

Blending of the two sequences means first of all to compute a generalization. This can be easily done by introducing variables for all those pairs of chords which do not coincide. In our example, we collapsed the dominant and the dominant seventh chord (fourth chord in the sequences) in the generalization to a dominant chord. The two blended candidates for possible chord sequences emerge naturally from the input. Without doubt the two blends are interesting chord progressions.

Example 2 Again two input sequences are considered, but this time the sequences are very complex and they differ in the length of the sequence. In the following, the first input sequence is the circle of fifths. This sequence is represented by iterated subdominant chords (double subdominant, triple subdominant etc.), respect. iterated dominant chords.

Input 1:	T → S → SS → S3 → S4 → S5 → S6 → D5 → D4 → D3 → DD → D → T
Input 1 (C maj):	C → F → B♭ → E♭ → A♭ → D♭ → G♭ → B → E → A → D → G → C
Input 2:	T → SS → T → D5 →Tp → SS → D5 →T
Input 2 (C maj):	C → B♭ → C → B → a → B♭ → B → C
Generalization:	T → x1 → x2 → x3 → x4 → x5 → x6 → T
Blend 1:	T → S → T → S3 → Tp → SS → D5 →T
Blend 1 (C maj):	C → F → C → E♭ → a → B♭ → B → C
Blend 2:	T → SS → SS → S3 → S4 → SS → S6 → T
Blend 2 (C maj):	C → B♭ → B♭ → E♭ → A♭ → B♭ → G♭ → C
Blend 3:	T → S → SS → D5 →S4 → S5 → D5 →T
Blend 3 (C maj):	C → F → B♭ → B → A♭ → D♭ → B → C

The generated candidates for chord sequences can be considered as rather interesting. In particular, there seems to be a trade-off between the degree of interesting progressions and more conservative progressions that stick more to the original sequences. This could be metaphorically described as a more progressive form of chord progression versus a more conservative form of chord progression. Here are two examples of heuristics which circumscribe these ideas:

- A conservative chord progression starts with the tonic and ends with the tonic whereas a more progressive chord progression does not stick to this constraint.
- A more conservative chord progression attempts to stay in a certain functional region, whereas a more progressive chord progression flips between functional regions.[13]

An important aspect for the next project phase will be the development and evaluation of appropriate heuristics that govern the blending process.

4 Remarks on the Implementation

Some remarks concerning the implementation of the systems in the domain of mathematics are covered in Subsection 2.3. A more complete initial specification of the system has been described in Deliverable D8.1. The reader is referred to this report for more information.

5 Conclusions

This report summarizes the results of the COINVENT project concerning the development of representation formalisms for concept invention by analogy and conceptual blending in the domains of mathematics and music. In particular, representation frameworks were considered that are compatible for computing generalizations and conceptual blends, i.e. formalisms that do not impose severe limitations and constraints to the computation of the blend spaces. The report summarizes the various formalisms, sketches the processes that allow the computation of blend spaces, and gives some hints concerning implementation issues.

In the case of mathematics, it is rather straightforward to represent a particular mathematical theory in form of a system of axioms. The language used to represent axioms is a many-sorted first-order logical language. This suffices to represent non-trivial mathematical theories as shown in Section 2.3 by considering the theory of complex numbers. It was shown how a restricted form of higher-order anti-unification can be used to generalize two input theories and how this generalization can be used as the generic space for conceptual blending. In a second step, concept blending can be performed governed by the principle that a blend should be as informative and as compressed as possible. The rather explicit example of complex numbers was used to illustrate the basic ideas of concept blending and to show how this example is implemented in the DOL framework.

[13]Functional regions could be, for example, the tonic region (T | Tp | tP | Tcp . . .), the subdominant region (S | Sp | sP | SS | S3 | . . .), and the dominant region (D | Dp | dP | DD | . . .). This idea is not new, in [36] one can find a very similar idea.

In the case of music, the situation is more difficult due to the fact that music can be described on quite many different levels of abstraction. Therefore, different representation formalisms are used at the same time representing different aspects of music. Due to the diversity of requirements the report sketches as representation formalisms the general chord type representation (GCT representation), Constrained Hidden Markov Models, and Feature Structures. For each framework the general mechanisms of concept blending are sketched and exemplified using several examples (blending on the chord level, blending on the chord progression level, blending on the phrase level of pieces, meta-level properties of music). Due to the fact that music is formally not as strongly constrained as mathematical theories (e.g. there is no hard consistency concept in music, a concept that is constitutive for mathematics), the choice and ranking of candidates for generalizations and blends is strongly dependent on heuristics. The development and testing of appropriate (if possible cognitively plausible) heuristics as well as the application of the framework to hard problems in mathematics and music is (according to the work package description) a task for the following project period.

References

[1] ALLAN, M., AND WILLIAMS, C. K. I. Harmonising chorales by probabilistic inference. In *Advances in Neural Information Processing Systems 17* (2004), MIT Press, pp. 25–32.

[2] ARGAND, J.-R. Philosophie mathématique. Essai sur une manière de représenter les quantités imaginaires, dans les constructions géométriques. *Annales des Mathématiques pures et appliquées 4* (1813), 133–146.

[3] BESOLD, T., AND KÜHNBERGER, K.-U. Applying ai for modeling and understanding analogy-based classroom teaching tools and techniques. In *KI 2014: Advances in Artificial Intelligence – 37th Annual German Conference on AI* (2014), C. Lutz and M. Thielscher, Eds., LNCS, Springer, pp. 55–61.

[4] BIDOIT, M., AND MOSSES, P. D. *CASL User Manual*. No. 2900. Springer, 2004.

[5] BORREL-JENSEN, N., AND HJORTGAARD DANIELSEN, A. Computer-assisted music composition – a database-backed algorithmic composition system. B.S. Thesis, Department of Computer Science, University of Copenhagen, Copenhagen, Denmark, 2010.

[6] BRACHMAN, R. J., AND LEVESQUE, H. J. *Knowlegde Representation and Reasoning*. The Morgan Kaufmann Series in Artificial Intelligence. Morgan Kaufmann, June 2004.

[7] BRESNAN, J. *Lexical Functional Syntax*. Blackwell, 2001.

[8] CAMBOUROPOULOS, E., KALIAKATSOS-PAPAKOSTAS, M., AND TSOUGRAS, C. An idiom-independent representation of chords for computational music analysis and generation. In *Proceeding of the joint 11th Sound and Music Computing Conference (SMC) and 40th International Computer Music Conference (ICMC)* (2014), ICMC–SMC 2014, p. (to appear).

[9] CONWAY, J. H., AND SMITH, D. A. *On Quaternions and Octonions: Their Geometry, Arithmetic, and Symmetry*. AK Peters, Natick, MA, USA, 2003.

[10] FARAUT, J., AND KORANYI, A. *Analysis on symmetric cones*. Oxford Mathematical Monographs. Oxford University Pres, 1994.

[11] FORNEY, G.D., J. The viterbi algorithm. *Proceedings of the IEEE 61*, 3 (Mar. 1973), 268–278.

[12] GOGUEN, J. An introduction to algebraic semiotics, with application to user interface design. In *Computation for Metaphors, Analogy, and Agents*, vol. 1562 of *Lecture Notes in Computer Science*. Springer, 1999, pp. 242–291.

[13] GOGUEN, J. Mathematical models of cognitive space and time. In *Reasoning and Cognition: Proc. of the Interdisciplinary Conference on Reasoning and Cognition* (2006), D. Andler, Y. Ogawa, M. Okada, and S. Watanabe, Eds., Keio University Press, pp. 125–128.

[14] GRANROTH-WILDING, M., AND STEEDMAN, M. A robust parser-interpreter for jazz chord sequences. *Journal of New Music Research 0*, 0 (2014), 1–20.

[15] GUHE, M., PEASE, A., SMAILL, A., MARTÍNEZ, M., SCHMIDT, M., GUST, H., KÜHNBERGER, K.-U., AND KRUMNACK, U. A computational account of conceptual blending in basic mathematics. *Cognitive Systems Research 12*, 3–4 (2011), 249–265.

[16] GUST, H., KÜHNBERGER, K.-U., AND SCHMIDT, U. Metaphors and heuristic-driven theory projection (HDTP). *Theoretical Computer Science 354* (2006), 98–117.

[17] HAMILTON, W. R. On quaternions: Letter to John T. Graves. *London, Edinburgh and Dublin Philosophical Magazine and Journal of Science xxv* (1844), 489–495.

[18] HANLON, M., AND LEDLIE, T. Cpu bach: An automatic chorale harmonization system. 2002.

[19] KALIAKATSOS-PAPAKOSTAS, M., AND CAMBOUROPOULOS, E. Probabilistic harmonisation with fixed intermediate chord constraints. In *Proceeding of the joint 11th Sound and Music Computing Conference (SMC) and 40th International Computer Music Conference (ICMC)* (2014), ICMC–SMC 2014, p. (to appear).

[20] KOOPS, H. V., MAGALHÃES, J. P., AND DE HAAS, W. B. A functional approach to automatic melody harmonisation. In *Proceedings of the First ACM SIGPLAN Workshop on Functional Art, Music, Modeling &#38; Design* (New York, NY, USA, 2013), FARM '13, ACM, pp. 47–58.

[21] KRUMNACK, U., SCHWERING, A., GUST, H., AND KÜHNBERGER, K.-U. Restricted higher-order anti-unification for analogy making. In *20th Australian Joint Conference on Artificial Intelligence (AI07)* (Gold Coast, Australia, 2007), vol. 4830 of *Lecture Notes of Artificial Intelligence*, Springer.

[22] LAKATOS, I. Falsificationism and the methodology of scientific research programmes. In *Criticism and the Growth of Knowledge*, I. Lakatos and A. Musgrave, Eds. Cambridge, 1974, pp. 91–196.

[23] LANGE, C., KUTZ, O., MOSSAKOWSKI, T., AND GRÜNINGER, M. The distributed ontology language (DOL): Ontology integration and interoperability applied to mathematical formalization. In *AISC/MKM/Calculemus* (2012), pp. 463–467.

[24] MARTINEZ, M., BESOLD, T., ABDEL-FATTAH, A., GUST, H., SCHMIDT, M., U., K., AND KÜHNBERGER, K.-U. *Theoretical Foundations of Artificial General Intelligence*, vol. 4 of *Thinking Machines*. Atlantis - Springer, 2012, ch. Theory Blending as a Framework for Creativity in Systems of General Intelligence, pp. 219–239.

[25] MARTINEZ, M., KRUMNACK, U., SMAILL, A., BESOLD, T., ABDEL-FATTAH, A., SCHMIDT, M., GUST, H., KÜHNBERGER, K.-U., GUHE, M., AND PEASE, A. Algorithmic aspects of theory blending. In *Proceedings of the 12th International Conference on Artificial Intelligence and Symbolic Computation* (in press), G. Aranda-Corral, F. Martín-Mateos, and J. Calmet, Eds., LNAI, Springer.

[26] MOSSAKOWSKI, T., MAEDER, C., AND LÜTTICH, K. The Heterogeneous Tool Set. In *TACAS 2007* (2007), O. Grumberg and M. Huth, Eds., vol. 4424 of *Lecture Notes in Computer Science*, Springer-Verlag Heidelberg, pp. 519–522.

[27] ONTAÑÓN, S., AND PLAZA, E. Amalgams: A formal approach for combining multiple case solutions. In *Proc. ICCBR-10* (2010), vol. 6176 of *LNAI*, Springer, pp. 257–271.

[28] PLOTKIN, G. D. A note on inductive generalization. *Machine Intelligence 5* (1970), 153–163.

[29] POLLARD, C., AND SAG, I. A. *Head-Driven Phrase Structure Grammar*. University of Chicago Press, 1994.

[30] RAIMOND, Y., AND ABDALLAH, S. The Chord Ontology. `http://motools.sourceforge.net/chord_draft_1/chord.html`. Accessed: 2014-08-16.

[31] RAIMOND, Y., AND ABDALLAH, S. The Event Ontology. `http://motools.sourceforge.net/event/event.html`. Accessed: 2014-08-16.

[32] RAIMOND, Y., AND ABDALLAH, S. The Music Ontology. `http://musicontology.com`. Accessed: 2014-08-16.

[33] RAIMOND, Y., AND ABDALLAH, S. The Timeline Ontology. `http://motools.sourceforge.net/timeline/timeline.html`. Accessed: 2014-08-16.

[34] RAIMOND, Y., ABDALLAH, S., SANDLER, M., AND GIASSON, F. The music ontology. In *ISMIR 2007: 8th International Conference on Music Information Retrieval* (Vienna, Austria, September 2007).

[35] REUER, V., AND KÜHNBERGER, K.-U. Feature constraint logic and error detection in icall systems. In *Proceedings LACL'05 Proceedings of the 5th international conference on Logical Aspects of Computational Linguistics* (2005), P. Blache, E. Stabler, J. Busquets, and R. Moot, Eds., vol. 3492 of *LNAI*, Springer, pp. 255–270.

[36] ROHRMEIER, M. Towards a generative syntax of tonal harmony. *Journal of Mathematics and Music 5*, 1 (Mar. 2011), 35–53.

[37] SCHMIDT, M. Restricted higher-order anti-unification for heuristic-driven theory projection. PICS-Report 31-2010, Univ. Osnabrück, Germany, 2010.

[38] SCHMIDT, M., KRUMNACK, U., GUST, H., AND KÜHNBERGER, K.-U. Heuristic-Driven Theory Projection: An Overview. In *Computational Approaches to Analogical Reasoning: Current Trends*, vol. 548 of *Studies in Computational Intelligence*. Springer, 2014, pp. 163–194.

[39] SIMON, I., MORRIS, D., AND BASU, S. Mysong: Automatic accompaniment generation for vocal melodies. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (New York, NY, USA, 2008), CHI '08, ACM, pp. 725–734.

[40] SMOLKA, G. Feature-constraint logic for unification grammars. *Journal of Logic Programming 12*, 1-2 (1992), 51–87.

[41] WHITEHEAD, A. N., AND RUSSELL, B. *Principia Mathematica*, vol. 1-3. Merchant Books, 1910.

[42] YI, L., AND GOLDSMITH, J. Automatic generation of four-part harmony. In *BMA* (2007), K. B. Laskey, S. M. Mahoney, and J. Goldsmith, Eds., vol. 268 of *CEUR Workshop Proceedings*, CEUR-WS.org.

[43] YOGEV, N., AND LERCH, A. A system for automatic audio harmonization, 2008.