Music Learning with Massive Open Online Courses (MOOCs)
L. Steels (Ed.)

doi:10.3233/978-1-61499-593-7-119

Chapter 9. Steps Towards Intelligent MOOCs

A case study for learning counterpoint

Katrien BEULS [a] Johan LOECKX [a]
[a] *VUB Artificial Intelligence Lab, Brussels, Belgium*

Abstract. Despite their overwhelming success, present-day Massive Open Online Courses are far removed from the student modelling capacities displayed by earlier Intelligent Tutoring Systems. Being mere content delivery tools, MOOCs typically lack a thorough assessment module as well as tools for personalising the learner's track. When learning music, particularly, these two properties are indispensable. This chapter surveys suggestions made by experts in the field of AI in education today towards the incorporation of ITS tools and techniques into MOOCs. Yet, more traditional student models and tutoring modules are not without shortcomings themselves and the real challenge lies in making active models of both the tutor and the student, which can be used to predict future learning tracks and set the right challenges. Agent-based tutoring systems offer an attractive framework for building such active tutor/student models. The proposed concepts are illustrated in the domain of music composition. A tutoring system has been implemented to teach students the craft of *counterpoint*, a commonly used strategy for learning polyphonic music composition. It is based on the theory of *flow* to keep students motivated and optimize learning.

Keywords. Adaptive Learning, Intelligent Tutoring Systems, MOOCs, counterpoint, music learning, student models, tutoring strategies, agent-based tutoring systems, music tutoring, online learning, theory of flow

1. Introduction

Since their appearance in 2011, Massive Open Online Courses (MOOCs) have become omnipresent in today's higher education landscape. Yet, although their rise is recent and their popularity large, the ideas that support these courses have been around for multiple decades (see [1] in this Volume). The first "teaching machine" was introduced in the fifties by the behaviorist B. F. Skinner in the form of an incremental mechanical system that would reward students for correction responses to questions [2]. The idea was later reinforced by the famous two-sigma problem that could show that student achievement in classroom interaction differs greatly from results obtained from individual tutoring [3]. If we transfer this idea into today's globally connected age, to what extent do participants in Massive Open Online Courses experience individual tutoring? Surely, in terms

of evaluation, "in classes of 100,000 students, or more, instructors, no matter how many assistants they might have, are not going to be able to do the grading" [4].

MOOCs are mere content delivering tools today, replacing traditional university lectures, more than tools for assisting teachers in traditional classroom education. Despite their overwhelming success in terms of student numbers that these courses reach, the current first-generation MOOCs have two main shortcomings that are often mentioned by experts: (i) automatic assessment does not go beyond regular expression matching in simple self test questions at the end of each lecture segment and (ii) every learner follows the same learning path through the lectures, lacking any personalized tutoring. These two shortcomings are also reflected in the high number of dropouts (90-95%), which is often attributed to challenges similar to distance learning, such as time management. However, a comparative study showed that "MOOC students learned a bit more than students in a traditional university course, but less than students taught with an interactive engagement pedagogy" [5]. In sum, MOOCs as they are today are very useful in blended learning settings, where a human teacher incorporates MOOC material into their own lectures but currently less efficient in stand-alone education.

Two main paths are typically put forward to escape this deadlock situation in the online setting. First, by constructing knowledge in a collaborative way and by assessing each other's work, students learn from each other. This kind of learning is sometimes referred to as collaborative or peer learning and found especially interesting in problem-solving kinds of domains, like design or music composition. Second, online courses form a testbed for adaptive learning techniques by means of intelligent automated tools. Indeed, Intelligent Tutoring Systems have focused for decades on building exactly such systems that perform automatic assessment based on extensive domain knowledge and student models. A student model can be defined as the set of beliefs that a tutor has about a student. These beliefs include the knowledge and skills of the student in the target domain, his learning preferences and other attributes. They can be inferred based on a student's observable behavior: through his answers, actions or the results that he obtains.

Indeed, one would assume that such personalized education becomes crucial in big groups of learners. The large-scale data available in MOOCs hosts a huge potential for machine learning techniques to extract and generalize over learner patterns and offer individual learning tracks. This chapter tries to bridge the apparent gap between earlier intelligent tutoring systems (ITSs) and the (seemingly) abrupt rise of video-based MOOCs and argues for the need of an **active tutor and student model**, something which can only be achieved within an agent-based architecture.

This chapter is organised as follows. First, the current state of individual tutoring in MOOCS is reviewed. Next, a brief history on Intelligent Tutoring Systems is given to make the reader familiar with the basic terminology and architecture. The main contribution of this chapter, active tutor and student models, will be introduced in a fourth section and its capabilities demonstrated in the domain of music—more specifically counterpoint tutoring, a compositional technique. Finally, conclusions are drawn.

2. Individual tutoring in MOOCs

How well do MOOCs score in terms of facilities for individual tutoring? Many course designers have argued already that the real advantage of using MOOCs lies in their value

in so-called *flipped classrooms* or hybrid education, where a regular lecturer relies on MOOCs only as content delivery for his course and uses it to spend more time on individual tutoring and discussions on the subject matter in physical interactions with the students in the classroom. A MOOC is then rather seen as one way of learning, which allows students to connect and collaborate by engaging in the learning process actively. Daphne Koller reported a higher-than-usual attendance in her Stanford courses that are taught this way: "We can focus precious classroom time on more interactive problem-solving activities that achieve deeper understanding—and foster creativity" [6]. Such studies point to the importance of teachers as individual mentors who can debug students' thinking and "honestly be enthusiastic when they excel" [7].

Apart from improving the quality of face-to-face time in lectures, the real question is whether MOOCs can be used as stand-alone tools in distance education in the way intelligent tutoring systems were thought to function? A systematic comparative study by Judy Kay and her colleagues at the University of Sydney across a sample of major massive open online course platforms revealed that "all of the systems currently have only rudimentary facilities to capture learner activity data for analysis" [8], meaning that offering opportunities for individual learning paths is not on the agenda of current MOOC designers. The student can see "rather simple information about their marks and progress" [id.]. The researchers therefore rightfully conclude by saying that "here is a place where there is exciting potential to introduce Artificial Intelligence & Education (AIED) tools and techniques into MOOCs" [8].

MOOCs certainly offer new opportunities for individual tutoring when they are used in blended learning settings where teachers can inspect every student's individual progress on certain exercises and intervene if needed by offering targeted feedback on the components the student is struggling with or teaming up stronger with weaker peers [9]. Still, automatized individual tutoring with richer evaluation models to measure student engagement are not yet fully realized. This is not the least the case in learning music, a discipline that involves many different skills that cannot be assessed with simple multiple choice questions and in which the personal nature of a learner's track is primordial. Yet, the effective use of online and intelligent technology has been underresearched in music as well [10]. Their absence has contributed to the general feeling of disappointment towards MOOCs [11]. To understand how this aspect could be improved, the following section situates MOOCs within the larger history of Intelligent Tutoring Systems, in which individual tutoring through the use of student models plays a prominent role.

3. A brief history of Intelligent Tutoring Systems

Although today a well-established concept, Intelligent Tutoring Systems (ITSs) have gone a long way since the first breakthroughs in the early seventies that incorporated AI techniques into programmed instructions. These early advances allowed for (i) alternative representations of content, (ii) alternative paths through material and (iii) alternative means of interaction. Much of the research into expert systems turned out to be useful for representing expert (tutor) knowledge and building student models. In the eighties, and still very much so today, the main research questions of the field of ITS could be formulated as follows [12]:

- What is the nature of knowledge, and how is it represented?

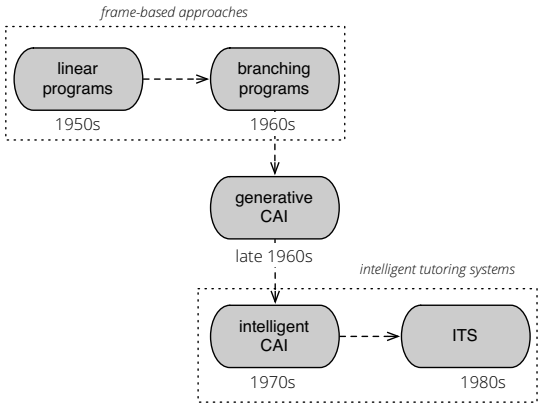


Figure 1. The history of Intelligent Tutoring Systems

.

- How can an individual student be helped to learn?
- Which styles of teaching interaction are effective, and when should they be used?
- What misconceptions do learners have?

3.1. From frame-based approaches to ITSs

The first attempts to build tutoring systems were all frame-based, where most frames contained simple questions (fill the gap exercises, selecting the correct answer, etc.). Such tutoring systems proceeded to present the next frame regardless of the accuracy of the student's response. It was therefore nothing more than a programmed textbook, completely lacking any individualization. In the 1960s, Crowder tried to overcome this major shortcoming as he introduced the notion of branching programs. Although still having only a number of fixed frames, these programs no longer ignored student's responses but the system could comment on a student's response and use it to choose the next frame [13].

Generative Computer-Assisted Instruction (CAI) was launched in the late 1960s. The idea of generative CAI was that a computer could generate teaching material automatically. One of the main advantages was that memory usage could be considerably reduced by this technique, since the frames did not have to be saved as such. However, this approach remained restricted to drill-type exercises, in which the learner model consisted of nothing more but an integer. Uhr and his collaborators [14] implemented a series of systems which auto-generated problems in vocabulary recall and arithmetic, two domains that presumably require drill and practice types of exercises. The sophistication in their systems was situated in the task-selection mechanism, which ensured the exercise level to be adapted to the student's overall performance.

It was Jaime Carbonell's mission to put Artificial Intelligence into CAI, meaning that the computer should have a representation of what is being taught, to whom and how [15, 16]. He developed SCHOLAR, a tutoring system for teaching Latin-American geography. SCHOLAR helped students enhance their knowledge by (i) solving problems

at a certain level or by (ii) involving them in discussions with the computer in a more interactive way.

Although there is no sharp boundary, in the 1980s intelligent CAI was replaced by Intelligent Tutoring Systems (ITS), which try to extend the domain of applicability, power and accuracy of CAI systems [17, 18, 19, 20]. Figure 1 summarises the different steps in the history of Computer-Aided Instruction until the arrival of ITS. Examples of early ITS include the Pittsburgh Urban Math Project (PUMP) algebra tutor [21] and the SHERLOCK control panel [22], used to train Air Force techniques to diagnose problems that might occur. Another early system is GUIDON [17, 23], which was the first intelligent tutor based on an expert system. GUIDON was also the first program to teach medical knowledge.

3.2. The general ITS architecture

Current Intelligent Tutoring Systems have a standard architecture with a number of components that are each responsible for a specific function. The components can best be explained according to the knowledge type they encode, which results in the following four types:

1. **Domain knowledge** (how experts perform in the domain): definitions, processes or skills needed to multiply numbers (e.g. the AnimalWatch tutor), generate algebra equations (e.g. PAT tutor), etc.;
2. **Student knowledge** (how to reason about student knowledge): stereotypic student knowledge of the domain and information about current student (time spent on problems, hints requested, correct answers, preferred learning style, etc.);
3. **Tutoring knowledge** (encoding reasoning about the feedback): either derived from empirical observations of teachers or enabled by technology (simulations, animated characters);
4. **Communication knowledge**: includes graphical user interfaces, animated agents, dialogue mechanisms.

The domain knowledge module (expert knowledge), the student model module (student knowledge) and the tutoring module (tutoring knowledge) are interconnected in the main architecture of an ITS. Communication knowledge is incorporated by a user interface module that mediates between the student input and the tutoring module (Figure 2). Because the communication knowledge is often included in the tutoring module, the remainder of this section discusses the three main interconnected modules in the ITS architecture: expert knowledge, student knowledge and tutoring knowledge.

3.2.1. Expert knowledge

Domain models interact very closely with the student model: they are the first step in representing the expert knowledge. They can generally be divided into three categories of complexity: (i) problem solving (mathematics problems, Newtonian mechanics), (ii) analytic and unverifiable domains (ethics, law) and (iii) design domains (architecture, music composition). There are two main axes in the classification of domain models: a first one ranging from simple to complex and a second one ranging from well-structured to ill-structured. Category 1 models represent expert knowledge in the field of arithmetic

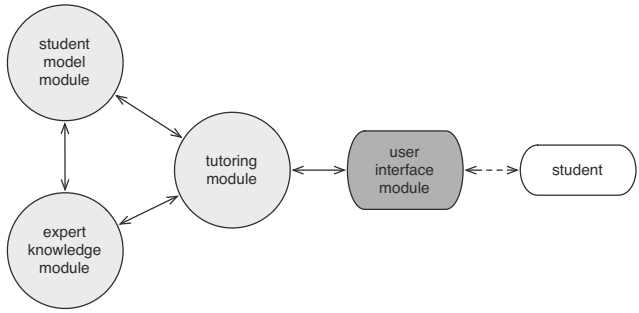


Figure 2. The basic architecture of an Intelligent Tutoring System consists of three main modules: expert knowledge, a student model and a tutoring module.

and other well-defined domains (well-structured, simple). Category 3 represents the other side of the axes: complex and ill-structured domains such as the knowledge needed to build an ITS for architecture tutoring. Finally, Category 2 contains qualitative representations of expert knowledge for fields such as language or music, which are halfway on both axes [24].

Learning music involves a whole range of learning activities: from music history over mastering an instrument to interpreting a piece and composing in a particular style, requiring different kinds of cognitive skills. While more academically oriented courses could be assessed and taught in a more traditional way, teaching students how to compose requires advanced tutoring and close guidance. In an online tutoring system, the models that represent this knowledge thus need to reflect this knowledge closely. To teach students the craft of classical composition, for example, a common approach is to teach the *practice of counterpoint*, which consists of a set of do's and don'ts in polyphonic music writing. It could be classified as Category 1 due to the formal and well-defined nature of rules that constrain harmony and melody. Still, just checking student's work for correctness is unsatisfactory given the complexity of the task and modelling the knowledge including all student misconceptions, is a difficult and time-consuming task that often needs to be carried out by hand.

3.2.2. Student knowledge

A student model can be defined as the set of beliefs that a tutor has about a student. These beliefs include the knowledge and skills of the student in the target domain, his learning preferences and other attributes. They can be inferred based on a student's observable behaviour: through his answers, actions or the results that he obtains. Traditional ITSs keep track of a student's performance based on a series of pre-set learning objectives, such as a range of grammatical phenomena in the target language or vocabulary items covering the learning situations that the student has selected.

To improve the student modelling enterprise, some tutoring systems allow their students to inspect and control the student model [25]. Student models with this property are called *open learner models* (OLMs). They can contain simple overviews of knowledge (such as a skill meter) or a more detailed representation of knowledge, concepts etc. Park Woolf [24] lists several motivations for the use of open learner models, such as (i) the student has the right of access to and control over his personal information;

(ii) the student can potentially correct the learner model; (iii) the frequent asymmetric relationship between the student and the tutor can be resolved; and (iv) OLMs stimulate reflective learning in the student.

Researchers in ITS tend to classify their student models according to three main dimensions. The first one covers the input that the system receives, while the remaining two are structural properties of the student profile. Van Lehn [26] refers to them as bandwidth, target knowledge type and the differences between student and expert:

1. *Bandwidth* refers to the amount and quality of the input that the diagnosis component receives about what the student is doing or saying. From this input, the tutor must infer what the student is thinking and believing [26].
2. *Target knowledge type*. Because a good student model can in practice solve the same problems as a real student would be able to solve, it can be used to actively predict the real student's answer. To solve these problems the model needs "*some kind of interpretation process that applies knowledge in the student model to the problem*" [27]. Depending on whether we are dealing with procedural or declarative knowledge, a different interpretation process is required.
3. *Student-expert differences*. The knowledge of a student is usually regarded as the background knowledge of a student modeling system. Student knowledge always needs to be understood in relation to an expert model that can provide explanations on the correct way(s) to solve a problem. To compare student and expert or tutor knowledge, most ITSs claim to use the same knowledge representation language for both [26]. However, reality is often different. Due to economy and other implementation issues, the student model is often a copy of the expert model plus a collection of differences: missing concepts (knowledge that the student does not yet have) and misconceptions (knowledge that the student has that the tutor does not).

3.2.3. Tutoring knowledge

A tutoring model has two main functions, which are mirrored in the basic tasks of instruction, namely *to stimulate and evaluate learning*. ITS resarch has mainly addressed these functions separately [28] and sometimes together as in the ASSISTment system [29, 30, 31], which combines 'assistance' and 'assessment'. A tutoring model thus needs to decide on *when* and *how* to intervene and it is responsible for content planning of what to teach next. The question of when and how to assist the learner is "the fundamental dilemma of tutoring" [32, 33, 34].

Assisting and tutoring the learner can further be divided into two sub-functions [34]: "cognitive diagnosis, defined as the detection of the sources of errors, and the selection of tutoring or remediation strategies". Recent developments in automatically learning the learner's affective states [35, 36, 37] have increased the complexity of reasoning about optimal tutoring decisions.

Tutor's decisions are often reflected in the different forms of interaction that the tutor has with the learner. Typical forms of interaction include socratic dialogs, hints, feedback from the system, etc. A human teacher typically uses six types of feedback [38, 39, 40]: explicit correction, recasts, clarification requests, metalinguistic feedback, elicitation, repetition or any combination of these. These interactions usually occur through the user interface module, that connects the student with the tutoring module (see Figure 2). The

user interface often includes a dialogue system for interacting with the student. This type of conversational interaction is particularly useful when the learner's answer is incomplete. Because tutors usually have an approximate sense of what a student knows, it "appears to be sufficient to provide productive dialogue moves that lead to significant learning gains in the student" [41].

4. Introducing active tutor and student models

4.1. State-of-the-art

Intelligent Tutoring Systems today work with static student models that keep track of a student's performance on certain predefined knowledge domains by counting scores and comparing these to averages. If the technique of a student model is to be included to make MOOCs more intelligent, we should consider a more dynamic student model that can actually function as a real model of the actual student and predict future behavior.

In Artificial Intelligence, a promising way to introduce such dynamic models is to make use of agents that can take on the role of learners and tutors. Such agents are autonomous entities that pursue their own goals and learn according to the outcome of its own or other agents' actions. Indeed, multi-agent systems have sometimes been considered as good candidates for building basic Intelligent Tutoring Systems infrastructures as they fulfil all the necessary requirements [42]:

(i) they are made of different interconnected, complex components;
(ii) they provide multiple, different and complementary services;
(iii) each of their components is functionally autonomous; and
(iv) they are equipped with specific knowledge structure and reasoning mechanisms.

Agents are thus often decomposed by their function in the teaching and learning process, with for instance one evaluation agent, one modeling agent, one recording agent, one student agent, etc. [43]. Moreover, the usefulness of agent technology in intelligent education systems is their contribution to make these systems adaptive, able to learn and dynamic by providing dynamic adaptation of domain knowledge and of behaviour of individual learners ([44], cited by [43]). Pedagogical agent-based systems are often used to monitor a particular project and enhance communication between members of a group [45]. Some researchers have designed agents for every course unit [46] or assigned a new agent to a specific learning topic [47].

4.2. Active student & tutor agents

We suggest to abandon such distributed systems in favour of a more holistic agent-based tutoring system with only two agents: one that models a tutor and one that simulates a learner. The latter can then function as *an active student model* that can run and try out solutions in parallel to predict a student's behaviour. These predictions can then be used to set the right challenge level, select the next exercise and suggest corrections to the real student. We will reuse concepts and findings of a recent PhD dissertation on language tutoring with such an agent-based architecture [48]. Similarly, the two agents that form the backbone of an agent-based counterpoint tutoring system are the music agent (ex-

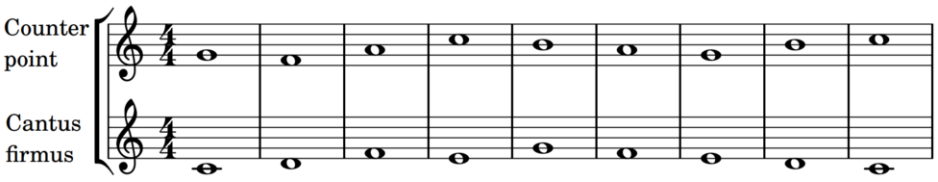


Figure 3. Example of a correct first species counterpoint piece. Given the cantus firmus (CF), a student is asked to compose the counterpoint voice, that should obey harmonic, melodic and motion constraints.

pert composer) and the student agent (music pupil). Both agents share the same architecture: a set of music rules, a processing engine and a meta-level architecture to capture inconsistencies in the student's composition of the counterpoint voice. Yet, although the components of the teacher and student agent are homologous, the realization of these components is not identical. The teacher agent represents an ideal composer in counterpoint whose musical skills also allow him to correct erroneous utterances of others. The student agent does not yet master all the counterpoint rules that are needed to be fully expressive in his compositions.

A music agent can be extended with a tutoring strategies component and a student profile component to become a fully-fledged tutor agent. These components personalize the tutoring process by keeping essential information about the student that is constantly being updated so that tutoring can be personalized to better fit the motivations of the individual student. This approach is particularly interesting in the domain of music, where learner's paths are very personal and the rules to be learnt exhibit complex interactions so that keeping track of which rules a student masters, is crucial.

5. Illustration in the domain of music

5.1. The study of counterpoint

The study of counterpoint attempts to express the properties of melodious polyphonic music, by investigating how the individual voices are formed and interact with each other. It has been a very important historical effort to capture the style of Renaissance music in which the independence of voices and harmonious polyphony is central. Still today, however, it is an indispensable tool to teach students the craft of classical music composition. A *counterpoint exercise* consists of a given *cantus firmus* (CF) or monophonic melody, on which a student has to compose a second melody to (called the *counterpoint voice*, CP). Of course, not all possibilities of notes are allowed (in this case a random melody would suffice), and here come the counterpoint rules into play. Figure 3 gives an example solution of such an exercise. The composed melody, should obey:

(a) harmonic or vertical (spanning two voices),
(b) melodic or horizontal (concerning one voice) and
(c) motion rules (relative movement of two voices).

As these rule are interacting in different ways, solving a counterpoint exercise can be seen as solving a complex sudoko. One discriminates between hard constraints (absolute "no-go's") and soft constraint (discouraged). Besides constraints, there are also guidelines

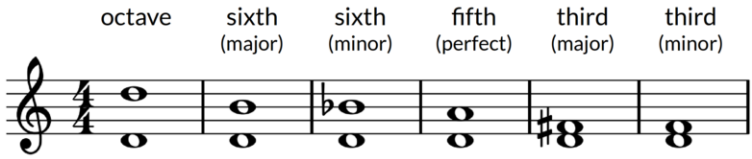


Figure 4. The study of counterpoint is centred around "rules" that limit the compositional freedom of a composer. They are intended to maximise the independence of voices while meanwhile preserving the harmonious nature. The most basic rule of counterpoint, pictured above, specifies which harmonic intervals are allowed and which are forbidden.

for creating 'better sounding' melodies, for example that a melody should be smooth and consist of the least number of skips available. The rules and guidelines collectively define the polyphonic musical style of the Renaissance era (1400-1600 AD). Because the 'freedom' of compositional expression has been limited on purpose in counterpoint, it helps pupils to cope with the many and complex interactions that occur constantly in polyphonic music.

Figure 4 depicts a basic harmonic rule in standard musical notation, that says that the possible *harmonic interval* (that is, distance between two notes with regards to pitch) are restricted: only octaves, minor/major sixths, perfect fifths and minor/major thirds are allowed. Of course there are many more and more complex rules that involve the two voices and movement of the melodic lines. The difficulty of counterpoint lies in the fact that the rules often conflict with each other so that solving one error leads to violation of another rule. It leads us too far to explain all this in detail and we refer readers interested to know more about the musical aspects of counterpoint to excellent literature on the topic [49, 50].

Also from a computational point of view, counterpoint is interesting as the "rules" to be learnt are quite formalised and the interactions not too complex. For this reason, it has been an interesting case study for computational representations in the past, using formalisms in the domain of feature spaces, fuzzy logic or constraint programming [51, 52, 53]. Though these approaches yield satisfying results, they lack musicality, explanatory power and mechanisms to engage them in tutoring activities as the musical knowledge can not be exploited.

In the following sections we will outline the basic operation of our proposed tutoring system for teaching counterpoint. We will look in more detail at the two-agent architecture, at the musical grammar to analyse a given composition and at how the tutor agent diagnoses errors and repairs them.

5.2. Tutoring based on the theory of "flow"

Figure 5 depicts the overall process of the proposed Intelligent Counterpoint Tutoring System, based on the concept of *flow*. Flow theory, introduced by Mihalyi Csikszentmihalyi [54], describes the experience of intrinsically motivated people when they are deeply engaged in an activity. During this time, people are in such a state of extreme concentration that they forget about time and the world around them. It is believed that learning is optimized in this situation and the primary goal of the process pictured in Figure 5 is thus to keep students in a state of flow. One of the groundbreaking aspects

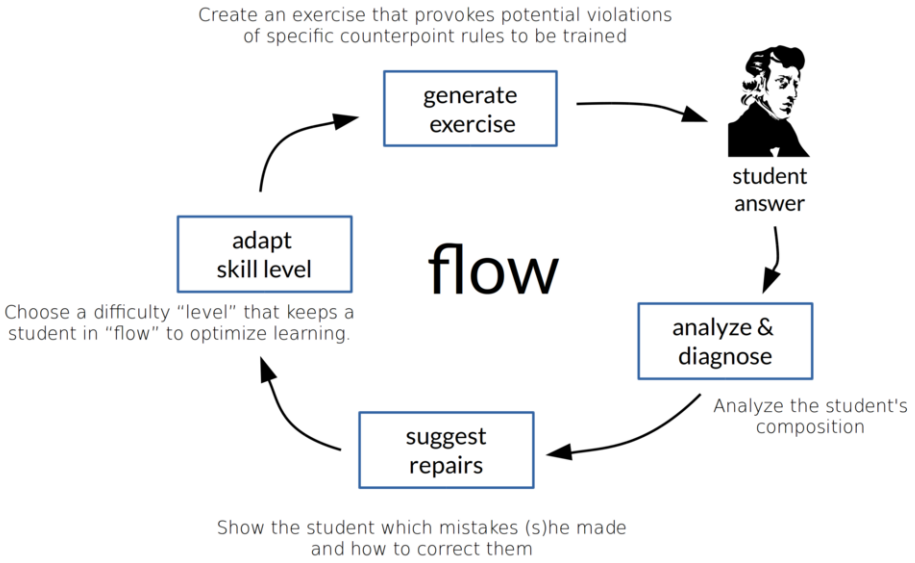


Figure 5. The challenge level is adapted to match the student's capabilities and skills to keep them in a state of flow and optimize learning. In order to achieve this, the tutoring agent should be well aware of what knowledge a student masters by keeping the student model up-to-date. The virtual tutor analyses the composition of a student and can suggest repairs when needed.

of this theory is the fact that the distinction between teacher and learner is diffused and students are put central to the learning process [55].

Flow occurs when a person perceives the challenges and the skills brought to it, as both balanced and (slightly) above average. For example, if an exercise is too difficult, a student may become anxious or frustrated which not only hampers learning but is also detrimental to motivation. If, on the other hand, the exercises that are presented are too easy, a student may become bored and lose interest. Finding this balance is one of the challenges of our tutor agent. As a consequence, it is essential that the tutor continuously gauges the student's skill level and keeps the student model in sync by tracking in detail which knowledge the student masters and which not.

In our specific case, the student model is an active one and corresponds to the (executable) set of counterpoint rules that the student masters. Because this knowledge can be instantiated, the tutor agent can *simulate* which response(s) a student will/might give. If there are any discrepancies observed, the student model is adapted to reflect accordingly to match the student's skill set. When a student provides an answer to an exercise, the expert tutor (who masters all counterpoint rules), analyses the piece and diagnoses which rules have been violated and suggests repairs. We will now look in more detail into each of these steps, starting with the musical analysis when a student submits the solution to an exercise.

5.3. Analysing the student's answer

To encode musical knowledge, we have opted for a grammar-based representation of counterpoint rules. Figure 6 shows an example initial structure before analysis. To build this grammar for musical composition we use the Fluid Construction Grammar frame-

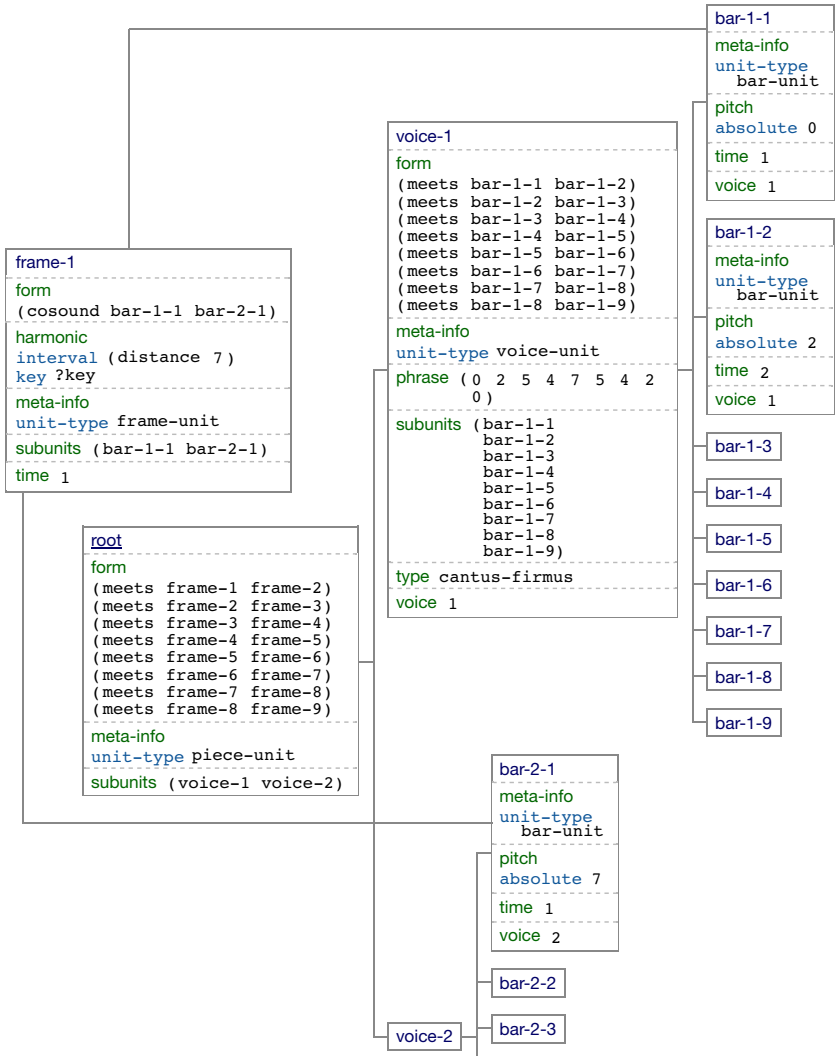
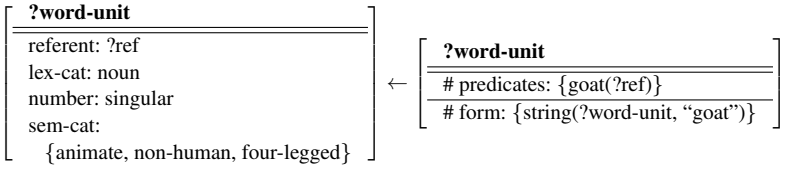


Figure 6. Partial initial transient structure for the example counterpoint exercise with selected voice, frame and bar units expanded. Bar units have two parent units: the voice to which they belong (horizontal view) as well as the frame (vertical view).

work, a computational formalism that is based on the theory of Construction Grammar within linguistics [56, 57]. In construction grammar, the main data structure is a construction, which is a mapping between meaning and form through semantic and syntactic categorisations. Constructions are implemented as feature structures with a conditional and a contributing pole. In grammatical processing, constructions contribute to a transient feature structure that is being built up from scratch, starting from a conceptualization in formulation and from an utterance in comprehension. Information that is in the conditional pole functions as a requirement that needs to be satisfied before a construction can contribute new features or hierarchy to the transient feature structure.

For example, a lexical construction for “goat” has the following conditional slots

(on the right-hand side of the arrow): semantic predicates (conditions in formulation) and strings (conditions in comprehension). In formulation, this construction will contribute the "goat" form and features such as `referent`, `lex-cat`, `number` and `sem-cat`. In comprehension, the construction contributes the meaning predicates as well as the same contributing features on construction's the left-hand side.

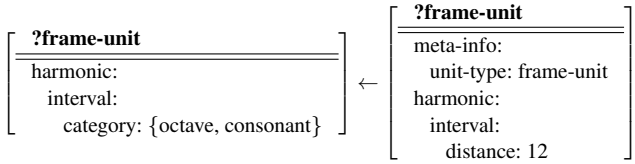


Similarly, musical rules can be represented as constructions. For the moment, we will ignore the controversial topic of 'meaning of meaning' and consider all counterpoint rules as being purely syntactic. Now instead of features such as `form`, `referent` or `sem-cat`, a construction contains information about the `pitch`, `interval`, `motion`, etc. of a note or a musical phrase. Instead of an utterance, the grammar engine will now receive a musical piece with two voices, the first one being the cantus firmus and the remaining one the counterpoint voices. For simplicity sake, we now only consider *first species* counterpoint exercises, also called *note against note*, in which only one note is played at the same time as the cantus firmus, as shown in Figure 3. In this simple case, a melody can be represented by a straightforward list of notes , in which tone pitches are represented by a single number from 0 to 12, representing the relative pitch in a chromatic scale (an octave contains 12 semitones in Western music). The list-based representation of the piece discussed above, is included here:

```
((0 2 5 4 7 5 4 2 0)
 (7 5 9 12 11 9 7 11 12))
```

Figure 6 shows the initial transient structure when analysing the example piece. There are melodic units such as `voice-1` containing the full phrase of the cantus firmus and harmonic units such as `frame-1` for the cosounding bars 1-1 and 2-1 with an interval distance of 7 semi-tones (see pitch features in respective bar units). It is on this initial feature structure that individual constructions will work.

Constructions can then focus on any part of the transient structure: certain bar units within the same voice, a frame unit connecting two bars vertically, etc. or even units that have not been built yet such as a melodic motion unit, combining all adjacent notes within a single motion movement (see Figure 7). Similar to the lexical construction shown above, a harmonic construction such as the octave construction consists of a conditional pole with features that have to be present (here a frame unit with harmonic interval distance of 12) and a contributing pole (harmonic interval categories octave and consonant).



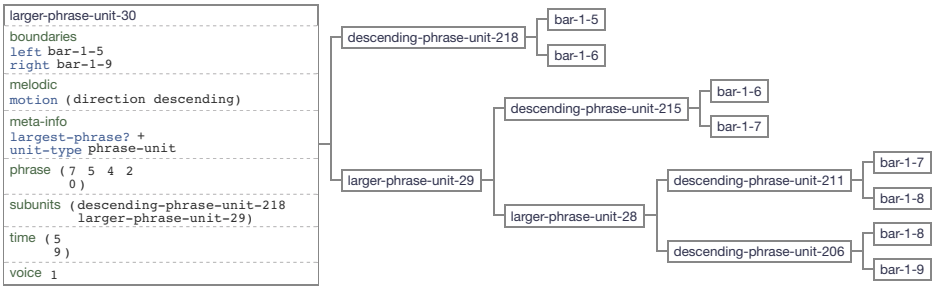


Figure 7. A descending motion phrase ranging from bar 1-5 to bar 1-9 of the example piece. Only the uppermost unit is fully expanded here. It groups a low-level descending phrase unit and a higher order phrase unit. The motion construction thus always looks at two units with the same melodic motion direction and unites them into a phrase unit, containing the full phrase, time information and the direction.

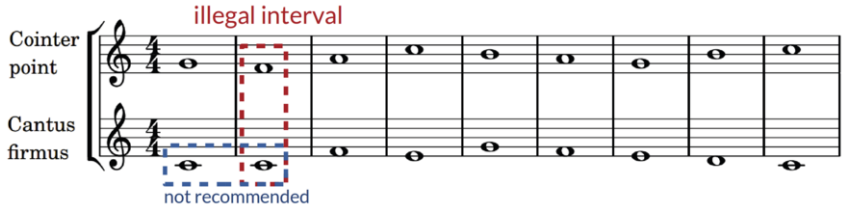


Figure 8. Example of a first species counterpoint piece with a violation of the "legal intervals" rule in the second bar.

The current grammar contains 29 constructions, divided into harmonic, melodic and harmonic-melodic ones. As it goes beyond the scope of this paper to describe the full grammar in detail, we'll simply make an inventory of the constructions of the tutor grammar that are used to analyse musical pieces.

5.4. Diagnosing the student's answer with regard to counterpoint rules

Once constructions have done their work to analyse the musical piece and build a transient structure that highlights relations between individual notes on a melodic and a harmonic level, the tutor uses a set of diagnostics that contain specific counterpoint rules to diagnose particular deviations. This section highlights some examples of such diagnostics for three types of counterpoint rules: harmonic, melodic and harmonic-melodic (motion). All diagnostics make use of the final transient structure that results from the application of the musical constructions. A diagnostic will either return one or more problems or nothing at all. The problems will then trigger repair strategies that launch a search process to find a satisfying solution for every problem. Let us consider the following erroneous counterpoint piece, with its musical notation shown in Figure 8.

```
((7 5 9 12 11 9 7 11 12)
(0 0 5 4 7 5 4 2 0))
```

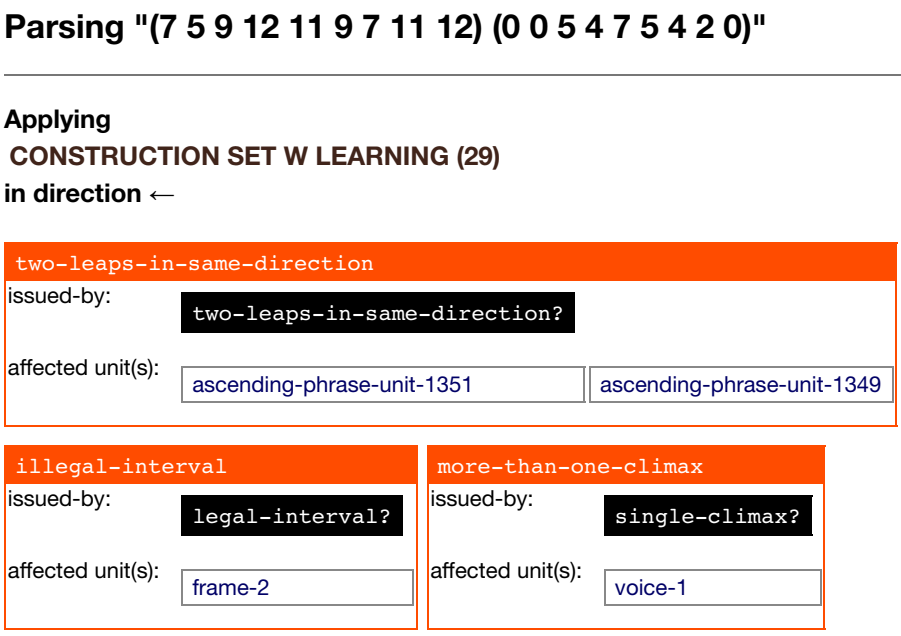


Figure 9. Problems diagnosed for the example melody: two illegal harmonic intervals in frames 2 and 4 and one melodic problem at the end where two skips follow each other in a descending motion (4→2→0).

5.4.1. Harmonic rules

1. **Legal intervals**
   This diagnostic will check all frames (vertical cuts in the piece) to see if each of them is either an octave, perfect fifth, minor/major third or minor/major sixth. Unisons are only allowed in the first bar. There are constructions for each of these intervals as well as for the illegal ones which leave behind a harmonic `interval-category` feature. This feature is a list such as {`third, major-third, consonant`}. All the diagnostic has to check here is the presence of the `consonant` value. In the example melody introduced above, there are two illegal intervals located in frame 2 (perfect fourth) and frame 4 (minor sixth), which is signalled by FCG diagnostics in Figure 8b.

5.4.2. Melodic rules

1. **Big leaps**
   No leaps bigger than a minor sixth (distance 8) are allowed, nor are tritones (distance 6). This diagnostic goes through every phrase unit and checks its melodic interval feature for its distance. If it is bigger than 8, it will diagnose a `leap-bigger-than-minor-sixth` problem. If it is equal to 6, it diagnoses a `tritone` problem.
2. **Exposed tritones**
   A motion where the begin and end note form a tritone is forbidden. To diagnose this, we need to find the largest motion unit (marked with the feature `largest-unit` set to `+` and then inspect the distance between the first and last

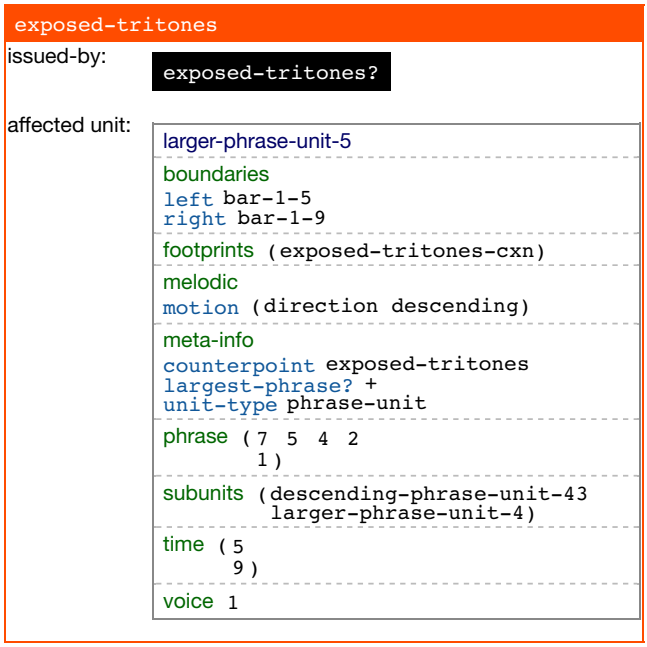
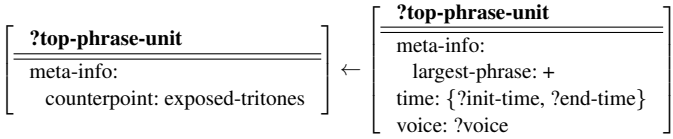
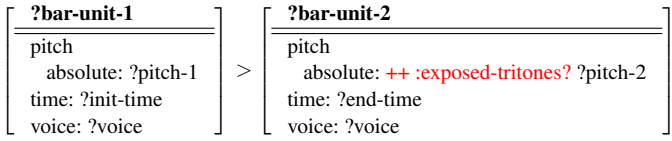


Figure 10. The descending phrase unit contains an exposed tritone, with the difference between start and end note equal to 6 semi-tones. The diagnostic has left a counterpoint feature in the unit's meta info.

note-unit. Instead of manually going through the transient structure, we instantiate a diagnostic construction and check whether it can apply[1]. If it does, the problem is present. We include the diagnostic `exposed-tritones-cxn` here below. Figure 10 shows the problem box in the web interface. Section 5.5 will show in more details how this problem could be repaired.





3. **Climax**
   A single climax is preferred, located somewhere in the middle of the counterpoint melody (when it is the highest voice), not at the beginning or the end. This rule requires two steps: first of all there can be only one global climax (marked with

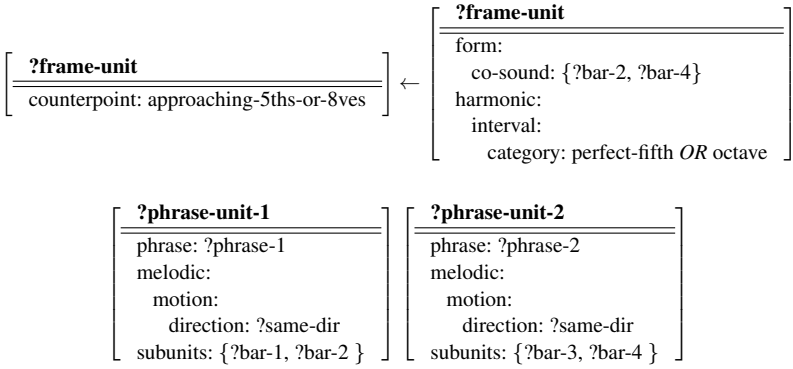
[1]The absolute pitch distance is calculated by means of an expansion operator (indicated by the ++) that is called during the matching process. If no tritone is found, the diagnostic construction cannot apply.

the feature `(global +)`. Second, this note should be situated around the middle of the melody. The diagnostic currently defines middle as the length of the melody plus and minus one. For each of these steps a problem can be signalled: `more-than-one-climax` or `single-climax-not-centralized`.

4. **Voice crossing**
   The counterpoint voice should not cross the cantus firmus. As there is a construction that analyses a melody for possible crossings and leaves a feature if there has been one (or multiple), this diagnostic simply has to check the presence of such a feature.

5.4.3. Motion rules

1. **Parallel perfect consonances**
   Thanks to the motion constructions (detecting parallel, similar, contrary or oblique motion in two melodies) and the melodic constructions that mark perfect fifths or octaves, it has become straightforward to signal the presence of a parallel perfect consonance. This diagnostic looks for a parallel motion phrase that contains a perfect consonance feature.
2. **Approaching fifths or octaves**
   A fifth or octave harmonic interval may not be approached by similar motion. This diagnostic again consists of a construction that looks as follows:



5.5. Suggesting pedagogically sound repairs

Simply indicating whether a counterpoint piece satisfies or breaks particular counterpoint rules without providing clues how to improve a solution, is insufficient in a complex learning domain as music composition. Insightful feedback in the form of "repair strategies" can hence improve learning considerably. Repair strategies formulate ways to solve *individual* problems that were diagnosed by the tutor agent. Solving an individual problem locally, however, can introduce new problems at different points in the melody. Repairing counterpoint violations thus turns into a local search problem that scores repairs and fixes (solutions suggested by a repair strategy).

The main purpose of repairing counterpoint violations is not just to find a solution (or fix) that satisfies all the rules, but one that provides *insight* to the student on how

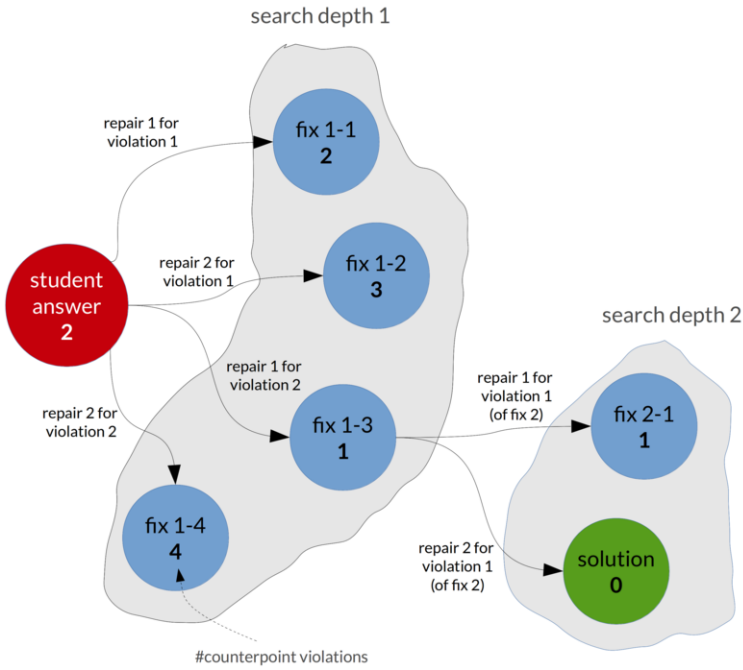


Figure 11. A breadth-first search is applied to find the least number of repairs needed to fix a student's piece. The number of counterpoint rule violations is indicated inside the node. First, all fixes (resulting from a repair) at search depth 1 are explored; next the fixes at depth 2 are explored, ordered by number of violations in ascending order.

to repair his/her mistakes. Therefore, it is important that the tutor provides the "easiest" way to repair its solution. Easy in this sense is the solution where the fewest additional problems were introduced while repairing, i.e. errors that were not made by the student.

Figure 11 illustrates the search process that the tutor runs through to find the best set of fixes. A student piece with two violations, is analysed and all possible fixes (that is, resulting pieces after a repair) are listed together with the remaining number of violations for each. Next, fixes at search depth two are explored and the process is repeated. The tutor follows thus a breath-first search strategy. A simple heuristic of ranking by remaining rule violations is used to choose the order of exploring the nodes at a particular depth. This approach is *guaranteed* to find the *easiest way* to repair the student's piece. Because the original exercise originates from a correct counterpoint with rule violations introduced, the search depth can be limited in practice. For example, when a student has to create a counterpoint voice from scratch, the search depth can be limited as there is no real instructional advantage of proposing many fixes to change a very bad piece into a good one.

It goes beyond the scope of this paper to explain the full details of the tutor's repair strategies. However, we briefly discuss a single problem with its possible repair strategies in what follows. The problem at stake is `exposed tritones` (see above). Figure 12 illustrates the problem and points to three possible repairs:

1. Altering the starting or ending note of the motion phrase.

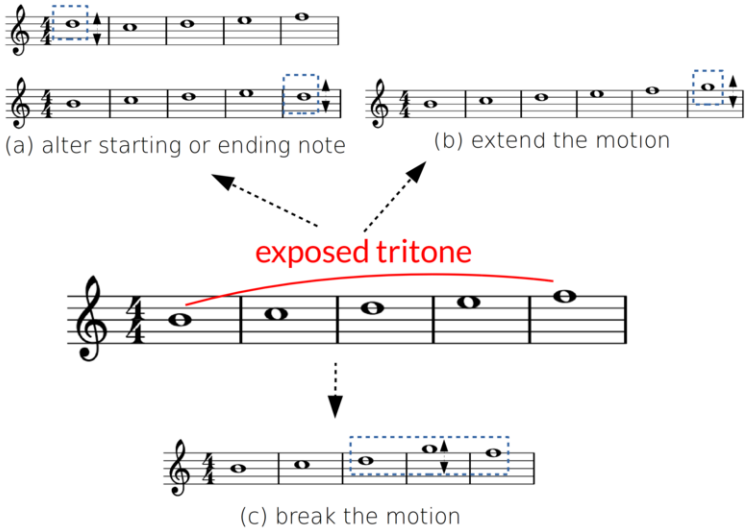


Figure 12. Possible repair strategies for an exposed tritone. The starting or ending note can be changed—with or without breaking the motion (a). The motion can also be extended so the tritone is resolved (b). Another solution is to break the motion within by altering the one-before-last note (c).

2. Extending the phrase so that the tritone is not exposed any more.
3. Breaking the motion somewhere in the middle of the phrase.

These repair strategies are relatively generic and can introduce valid solutions for other problems at the phrase level too (e.g. approaching fifths). They all work on the affected unit(s) for which the problem was diagnosed but also have access to the full transient structure of the musical analysis as they might have to modify other units (corresponding to notes, melodic phrases, harmonic frames or complete voices) in order to solve the counterpoint violation.

For instance, the first strategy to change the initial or ending note of a phrase (given by the problem) yields multiple possible fixes. The exact number is dependent on the range of notes the initial note can be changed into. For now, both the first and the last note of the phrase can be increased or decreased by maximally 3 degrees, resulting in twelve possible fixes for this first repair strategy. To select a valid fix, we rank them based on the number of new problems they introduce into the piece and first pursue the one with the fewest.

5.6. Updating the student model

Finally, the tutor agent has immediate access to the student model so that he can scrutinize the actual state of the agent's construction inventory and learning strategies. It is only then that he can properly align the student model to the real student that is being coached. The student model is aligned after every interaction that the student has with the tutor agent. When the student was successful in the current task, the tutoring strategies will indicate how to update the student agent's constructions that correspond to the task. Indeed, as particular potential rule violation were elicited by the exercise design, the tutor can increase its confidence that the student truly masters these counterpoint rules. Vice

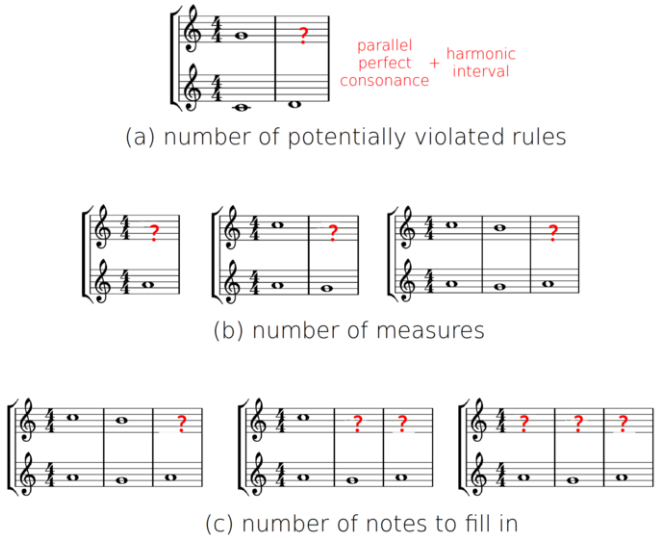


Figure 13. The difficulty level of counterpoint exercises can be controlled in a fine-grained manner. The tutor agent gradually increases the number of potentially violated rules, number of measures and number of notes to fill in, according to the skill level of the student.

versa, when a student's solution is diagnosed with errors, the student model is updated accordingly by lowering the confidence that the student masters the rule.

Also, when there was a failure or mismatch between the goals of the tutor agent and the real student (this can happen specifically when a new difficulty (counterpoint rule) is introduced by the tutor), the student agent proves useful to verify whether the mismatch could have been predicted based on the student model or not. Because the construction inventories have the same architecture in the tutor and the student agent, this symmetry can be used to learn about possible gaps or inconsistencies in the student's grammar.

5.7. Creating counterpoint exercises

The student model informs the tutor about the exercise complexity a pupil can handle and which counterpoint rules (s)he masters. To keep students in flow, fine-grained control over the challenge level is crucial to keep it in balance with the skill level. For this reason, we propose the fine-tuning of three parameters as shown in Figure 13:

(a) the number of measures (longer pieces can introduce more conflicts),
(b) the number of notes to fill in (more degrees of freedom), and
(c) the rules that are potentially violated in the exercise.

The automatic generation of exercises with such a fine-grained particular challenge level, however, has not yet been fully operationalised. In a first stage, we have implemented control measures (a) and (b). In the future, we plan to implement (c) in the following manner. Using a database of correct counterpoint pieces of varying length that is constantly updated, a new piece is generated by altering random notes to create a piece that will probably violate certain rules. Analysis of this piece tells the tutor which rules have been violated exactly. At this point, the tutor can decide to present this piece to the stu-

dent if it aligns with the particular difficulties the student struggles with (according to the student model), or store it for later use and create a new exercise. Clearly, this approach is far from perfect and needs further refinement.

6. Conclusions

We can now speak of at least two generations of ITS research, and we are currently at the dawn of a third one, which will probably be much more revolutionary. The first generation spans roughly from 1970 until 1990, a period of thirty years in which more than 40 systems were released. This early generation was powered by the booming of Artificial Intelligence, a field that was seeking applications for its technologies. The second generation, ranging from roughly 1990 until today has formulated the scientific foundations of the field [58]. Also implementations of real systems in schools realized, thanks to new spread of digital technologies in traditional education. The third generation massively scales the potential of Computer-Assisted Instruction to online video-based courses that are freely accessible to thousands of students: the so-called MOOCs. Although the first hype of MOOCs in the years 2012-2013 had set high hopes onto the disrupting nature of these courses on the traditional education landscape, they failed to accomplish them.

One reason for their failure that this chapter has put forward is the lack of individual tutoring they offer (when used in a distance education setting), which is mainly due to the absence of a student model and tutoring strategies. By building a bridge to earlier techniques found in Intelligent Tutoring Systems and enhancing these by making use of a truly predictive student model in the form of an active autonomous agent that simulates the actual learner.

In this chapter, we have outlined a tutoring system based on the theory of flow. Flow theory describes the experience of intrinsically motivated people when they are deeply engaged in an activity. It is believed that learning is optimized when students are in a state of flow. This situation happens when the challenge level is kept in balance with the skill level of the student to avoid boredom and anxiety. For this reason, an architecture based on active tutor and student agents is proposed. As the student agent carefully track the pupil's skill level and can simulate its answers, appropriate exercises can be presented to the student that keeps her/him in a state of flow.

The proposed methodology has been tested in the domain of music, more precisely the Study of Counterpoint, still today an effective instructional tool to teach students the craft of polyphonic music composition. An operational grammar of music and counterpoint as well as a tutor and student agent has been implemented in Fluid Construction Grammar (FCG). A core component of the virtual tutor is the in-depth musical analysis of the student's piece and the possibility to suggest repairs that correct the mistakes. This way, the student does not only get feedback on the correctness of an answer (right/wrong), but also insight into his/her mistakes and how to solve them, a mechanism central to any learning process. Though the creation of counterpoint exercises that elicit specific counterpoint violations is still suboptimal, the proposed work opens up the way to a stand-alone online counterpoint tutor.

References

[1] L. Steels, "The coming of moocs," in *Learning about music in the age of MOOCS* (L. Steels, ed.), Amsterdam: IOS Press, 2015.

[2] B. F. Skinner, *Cumulative record*. New York: Appleton Century Crofts, 1961.

[3] B. Bloom, "The 2 sigma problem: The search for methods of group instruction as effective as one-to-one tutoring," *Educational researcher*, vol. 13, pp. 3–16, 1984.

[4] G. Ritzer, "The rise of the prosuming machines." Blog article, March 2014.

[5] K. Colvin, J. Champaign, A. Liu, Q. Zhou, C. Fredericks, and D. Pritchard, "Learning in an introductory physics mooc: All cohorts learn equally, including an on-campus class," *The International Review of Research in Open and Distance Learning*, vol. 15, no. 4, 2014.

[6] D. Koller, "Death knell for the lecture: Technology as a passport to personalized education," *New York Times*, 5 December 2011.

[7] F. G. Martin, "Will massive open online courses change how we teach?," *Commun. ACM*, vol. 55, pp. 26–28, Aug. 2012.

[8] J. Kay, P. Reimann, E. Diebold, and B. Kummerfeld, "Moocs: So many learners, so much potential ...," *IEEE Intelligent Systems*, vol. 28, no. 3, pp. 70–77, 2013.

[9] S. Kahn, "Let's use video to reinvent education." TED Talk, March 2011.

[10] J. Bowman, *Online Learning in Music: Foundations, Frameworks, and Practices*. Oxford University Press, 2014.

[11] A. Agarwal, "Why massive open online courses (still) matter." TED Talk, June 2013.

[12] J. A. Self, "Bypassing the intractable problem of student modeling," in *Intelligent Tutoring Systems: At the crossroads of Artificial Intelligence and Education* (C. Frasson and G. Gauthier, eds.), Norwood, NJ: Ablex, 1988.

[13] N. A. Crowder, "On the difference between linear and intrinsic programing," in *Programs, teachers, and machines* (A. G. Grazia and D. A. Sohn, eds.), pp. 77–85, New York, NY: Bantam Books, 1964.

[14] L. Uhr, "Teaching machine programs that generate problems as a function of interaction with students," in *Proceedings of the 24th National Conference,*, pp. 125–134, 1969.

[15] J. Carbonell, *Mixed-initiative man-computer instructional dialogue*. PhD thesis, MIT, Cambridge, MA, 1970.

[16] J. Carbonell, "Ai in cai: An artificial intelligence approach to computer aided instruction," *IEEE Transactions on Man-Machine Systems*, vol. 11, pp. 190–202, 1970.

[17] W. J. Clancey, *Transfer of Rule-based Expertise through a Tutorial Dialogue*. PhD thesis, Stanford University, Stanford, CA, 1979.

[18] J. A. Self, "Student models in computer aided instruction," *International journal of man-machine studies*, vol. 6, pp. 261–276, 1974.

[19] J. A. Self, "Concept teaching," *Artificial Intelligence*, vol. 9, pp. 197–221, 1977.

[20] J. A. Self, "A perspective on intelligent computer-aided instruction," *Journal of Computer Assisted Learning*, vol. 1, pp. 159–166, 1985.

[21] J. Anderson, A. Corbett, K. Koedinger, and R. Pelletier, "Cognitive tutors: Lessons learned," *The journal of the learning sciences*, vol. 4, no. 2, pp. 167–207, 1995.

[22] A. Lesgold, S. Lajoie, M. Bunzo, and G. Eggan, "Sherlock: A coached practice environment for an electronics troubleshooting job," *Computer Assisted Instruction and Intelligent Tutoring Systems*, pp. 201–238, 1992.

[23] W. J. Clancey, *Knowledge-based Tutoring: The Guidon Program*. Cambridge, MA: MIT Press, 1987.

[24] B. Park Woolf, *Building Intelligent Interactive Tutors*. Burlington, MA: Morgan Kaufmann, 2008.

[25] R. Cook and J. Kay, *The justified user model: a viewable, explained user model*. Basser Department of Computer Science, University of Sydney, 1994.

[26] K. VanLehn, "Student Modeling," in *Foundations of Intelligent Tutoring Systems* (M. Polson and J. Richardson, eds.), pp. 55–78, Hillsdale, NJ: Erlbaum, 1988.

[27] M. Polson and J. Richardson, *Foundations of Intelligent Tutoring Systems*. Interacting with Computers Series, Taylor & Francis, 2013.

[28] K. VanLehn, "Intelligent tutoring systems for continuous, embedded assessment," in *The future of assessment: Shaping teaching and learning* (C. Dwyer, ed.), Lawrence Erlbaum Associates, 2007.

[29] L. Razzaq, J. Patvarczki, S. Almeida, M. Vartak, M. Feng, N. Heffernan, and K. Koedinger, "The assistment builder: Supporting the life cycle of tutoring system content creation," *Learning Technologies, IEEE Transactions on*, vol. 2, no. 2, pp. 157–166, 2009.

[30] L. Razzaq and N. Heffernan, "Open content authoring tools," *Advances in Intelligent Tutoring Systems*, pp. 407–420, 2010.

[31] T. Turner, M. Macasek, G. Nuzzo-Jones, N. Heffernan, and K. Koedinger, "The assistment builder: A rapid development tool for its," in *Proceedings of the 12th Annual Conference on Artificial Intelligence in Education*, pp. 929–931, 2005.

[32] J. Beck, K. Chang, J. Mostow, and A. Corbett, "Does help help? introducing the bayesian evaluation and assessment methodology," in *Intelligent Tutoring Systems*, pp. 383–394, Springer, 2008.

[33] L. Razzaq and N. Heffernan, "To tutor or not to tutor: That is the question," in *Proceedings of the Conference on Artificial Intelligence in Education*, pp. 457–464, 2009.

[34] J. Bourdeau and M. Grandbastien, "Modeling tutoring knowledge," in *Advances in Intelligent Tutoring Systems* (R. Nkambou, J. Bourdeau, and R. Mizoguchi, eds.), pp. 123–143, Springer, 2010.

[35] I. Arroyo, D. G. Cooper, W. Burleson, B. P. Woolf, K. Muldner, and R. Christopherson, "Emotion sensors go to school," in *Proceeding of the 2009 conference on Artificial Intelligence in Education, July 6th-10th, Brighton, UK, IOS Press*, pp. 17–24, 2009.

[36] J. Walonoski and N. Heffernan, "Detection and analysis of off-task gaming behavior in intelligent tutoring systems," in *Intelligent Tutoring Systems*, pp. 382–391, Springer, 2006.

[37] B. Woolf, I. Arroyo, D. Cooper, W. Burleson, and K. Muldner, "Affective tutors: Automatic detection of and response to student emotion," *Advances in Intelligent Tutoring Systems*, pp. 207–227, 2010.

[38] A. Ferreira, J. D. Moore, and C. Mellish, "A Study of Feedback Strategies in Foreign Language Classrooms and Tutorials with Implications for Intelligent Computer-Assisted Language Learning Systems," *International Journal of Artificial Intelligence in Education*, vol. 17, pp. 389–422, 2007.

[39] R. Lyster and L. Ranta, "Corrective Feedback and Learner Uptake," *Studies in Second Language Acquisition*, vol. 19, pp. 37–66, Mar. 1997.

[40] I. Panova and R. Lyster, "Patterns of corrective feedback and uptake in an adult ESL classroom," *Tesol Quarterly*, vol. 36, no. 4, pp. 573–595, 2002.

[41] A. C. Graesser, P. Chipman, B. C. Haynes, and A. Olney, "AutoTutor: an intelligent tutoring system with mixed-initiative dialogue," *IEEE Transactions on Education*, vol. 48, Nov. 2005.

[42] R. Nkambou, J. Bourdeau, and P. V., "Building Intelligent Tutoring Systems: An Overview," in *Advances in Intelligent Tutoring Systems* (R. Nkambou, J. Bourdeau, and R. Mizoguchi, eds.), Studies in Computational Intelligence, ch. 18, pp. 361–375, Berlin: Springer, 2010.

[43] S. Sun, M. Joy, and N. Griffiths, "The use of learning objects and learning styles in a multi-agent education system," *Journal of Interactive Learning Research*, vol. 18, no. 3, pp. 381–398, 2007.

[44] M. A. Razek, C. Frasson, and M. Kaltenbach, "Toward more cooperative intelligent distance learning environments," *Software Agents Cooperation Human Activity*, 2002.

[45] M. D. Beer and J. Whatley, "A multi-agent architecture to support synchronous collaborative learning in an international environment," in *Proceedings of the first international joint conference on Autonomous agents and multiagent systems: part 1*, pp. 505–506, ACM, 2002.

[46] Y. Shang, H. Shi, and S.-S. Chen, "An intelligent distributed environment for active learning," *Journal on Educational Resources in Computing (JERIC)*, vol. 1, no. 2es, p. 4, 2001.

[47] M. Boicu, G. Tecuci, B. Stanescu, D. Marcu, M. Barbulescu, and C. Boicu, "Design principles for learning agents," in *Proceedings of AAAI-2004 Workshop on Intelligent Agent Architectures: Combining the Strengths of Software Engineering and Cognitive Systems*, pp. 26–33, 2004.

[48] K. Beuls, *Towards an agent-based tutoring system for Spanish verb conjugation*. PhD thesis, Vrije Universiteit Brussel, November 2013.

[49] J. J. Fux, *The study of counterpoint from Johann Joseph Fux's Gradus ad Parnassum*. No. 277, WW Norton & Company, 1965.

[50] P. Hindemith, *The Craft of Musical Composition*, vol. 2. Schott & Co Ltd, 1984.

[51] A. E. Yilmaz and Z. Telatar, "Note-against-note two-voice counterpoint by means of fuzzy logic," *Knowledge-Based Systems*, vol. 23, no. 3, pp. 256–266, 2010.

[52] T. Anders and E. R. Miranda, "Constraint programming systems for modeling music theories and composition," *ACM Computing Surveys (CSUR)*, vol. 43, no. 4, p. 30, 2011.

[53] D. Herremans and K. Sörensen, "Composing fifth species counterpoint music with a variable neighborhood search algorithm," *Expert systems with applications*, vol. 40, no. 16, pp. 6427–6437, 2013.

[54] M. Csikszentmihalyi, *Beyond bordedom and anxiety: experiencing flow in work and play*. Cambridge University Press, 1978.

[55] L. Steels, "The architecture of flow," in *A Learning Zone of One's Own* (M. Tokoro and L. Steels, eds.), pp. 137–149, IOS Press, 2004.

[56] L. Steels, *Design patterns in fluid construction grammar*, vol. 11. John Benjamins Publishing, 2011.

[57] L. Steels, "Basics of fluid construction grammar," *Constructions and Frames*, 2015.

[58] J. A. Self, "Theoretical foundations for intelligent tutoring systems," *Journal of Artificial Intelligence in Education*, vol. 1, no. 4, pp. 3–14, 1990.