# TV-PR: Theme and Variations Planner/Realizer

By

Richard Wesley Todd, Jr.

Submitted to the
Office of Honors Programs and Academic Scholarships
Texas A&M University
in partial fulfillment of the requirements for

1998-99 UNIVERSITY UNDERGRADUATE RESEARCH FELLOWS PROGRAM

Submitted on April 15, 1999

Approved as to style and content by:

Dr. John Yen (Faculty Advisor)
Department of Computer Science

Susanna Finnell, Executive Director
Honors Programs and Academic Scholarships

Fellows Group: Engineering I

Abstract

**TV-PR Theme and Variations, Planner/Realizer**

Richard Todd, Advised by Dr. John Yen
University Undergraduate Fellow, 1998-1999, Texas A&M University,
Department of Computer Science

Computer generated music sounds too random and disconnected. Even when the melody and harmony are very pleasant (a task that computers are increasingly adept at performing), one doesn't have to listen long before realizing that a computer wrote the piece. Leading researchers in the field agree that continuity in computer music deserves more attention at this point.

My goal is to improve the overall form and coherence in automated music generation. The main focus of my work will be text planning techniques from the natural language generation field. These text planners guide the text generators and help computers form not only well-formed sentences, but also coherent paragraphs. Currently, there is no computer music system that uses a similar planning scheme. I will adapt the text planner-generator model to computer music, and design a working system that generates a plan for a set of variations on a user-given theme.

# TV-PR: Theme and Variations, Planner-Realizer

**(a Computer-Generated Music Project)**

Richard Todd, Dept. of Computer Science

Dr. John Yen, Advisor

## Introduction

This paper discusses TV-PR, a Theme and Variations Planner/Realizer. It is a computer program, and it takes computer-generated music in a new direction. The Planner/Realizer paradigm originated in natural language research, where it is used to help shape paragraphs of text into a coherent form. As will be shown, computer-generated music today lacks the sophistication of complete coherence. By applying the Planner/Realizer idea to music, TV-PR hopes to solve this problem.

There are two types of 'correctness' in music. In this paper I'll call them horizontal and vertical correctness. Vertical correctness refers to the quality of the sound at any instant in time (the name incidentally being derived from looking at a vertical column of notes in the score). Horizontal correctness refers to the quality of the music as it changes (derived from looking across the page).

The majority of work in computer music so far has been primarily vertical, that is, correctness of individual chords. The goal is to make sure every local combination of notes sounds good. Most go as far as to consider the sequence of vertical events that make up a chord sequence. However, as will be shown in the discussion below, just knowing that the music sounds good at every instance is not equivalent to knowing that

the piece sounds good as a whole. The logical flow of the piece, the style, and the variety among other things determine the quality of the whole as well.

**EMI - Experiments in Musical Intelligence**

At this point I'd like to discuss one of the most successful computer music projects to date. The project is called EMI, and its output has been featured on CDs and numerous reviews. Perhaps most telling is that it has been on the winning side of contests where humans try to guess if pieces were Mozart or EMI [Cope 1996].

Surprisingly, code originally written to create lexically correct haiku poems forms the base of the EMI program [Cope 1987]. EMI creates music by locating stylistically interesting points in existing works and incorporating these into generic, 'context-free' music [Cope 1991 & 1992]. This recombinant form of creation can be convincing for a few measures, but in terms of large-scale form becomes disjointed and incoherent [Smoliar 1994]. Cope says himself that "continuity deserves the most attention at this juncture," as computer music programs tend to lump musically correct phrases together with no interest in their potential connectivity [Cope 1991].

EMI works by comparing several similar works in the same style, and locating similar musical structures. The result of this extensive pattern-matching operation is a set of what EMI calls 'signatures,' the essence of the style under consideration. However, the user must alter the input music considerably to facilitate the process.

Specifically, the user must make the pieces analyzed look as similar as possible, without changing the music. It's matters of representation that humans can deal with easily, but which fools computers easily. For instance, the input pieces need to all be in

the same key. Otherwise functionally equivalent phrases will not match due to different key-centers. Also, the time signatures need to match. The user might need to re-notate one piece so that the time signature matches the other. Also, one piece in fast tempo and long notes might be converted to a slower tempo with faster notes. The sound of the altered piece must be identical to the original (change of key notwithstanding), and if the changes are done correctly the number of valid signatures grows considerably. The great success of the EMI program in practice makes the trouble of inputting the music worth it.

While the signatures and recombination of existing music produces good results at the phrase level, the large scale structure in EMI is defined by the user as well. Thus the large-scale structure problem is eliminated by human intervention in EMI.

## The Problem at Hand

EMI is representative of most (if not all) current music programs in that they either force the user to impose a large-scale form on their output, or they ignore the issue altogether. Many times this is a matter of the algorithm involved. Many simple algorithms (such as the one that created the sample in Appendix A) simply produce output until some stop point, which has everything to do with the end of a numerical sequence and nothing to do with musical meaning.

TV-PR will address this issue by using a planner to describe the entire piece from an abstract level before any music is written. The planner will create a framework that facilitates a logical, stylistically consistent output. All methods employed and the major theoretical influences on the project are explored below.

**Goals for TV-PR**

I have three major goals for TV-PR: Independence from style, simplicity, and the reuse of existing algorithms. The first is the most difficult to accomplish, as so much music theory is steeped in the style from which it is formulated. For studying a certain style of music, this is no hindrance, but to teach a computer to handle any input one must separate style from other aspects deemed universal. TV-PR endeavors to do just this, with heavy influence from the Schillinger System of Music Composition (discussed later). Simplicity flows freely from the Blackboard framework (a problem-solving paradigm used in artificial intelligence) on which TV-PR is built. The rather large problem at hand is fragmented fairly efficiently into agents which act individually (but cooperatively) to form the whole solution. Finally, it will be shown in the discussion of the algorithms employed that several existing algorithms are integrated into the planner. Not only does this simplify the problem, but it also sped development up greatly.

**Why Theme and Variations Form?**

There are many reasons to choose Theme and Variations form for the first incarnation of a music generator. First, it allows TV-PR to avoid generating an original melody. As a result, it also doesn't need to be able to evaluate its theme in terms of 'good' and 'bad.' Also, restricting ourselves to one theme eliminates an entire dimension of the planning task, greatly simplifying the problem. A method for generalizing the system for multiple themes will be given later. Yet another advantage to Theme and Variations form is the lack of complex transitions between the individual variations. Indeed, by far the most common transition between two variations is a couple seconds of silence. At first glance transitions may seem trivial, but any composer will whole-

heartedly disagree with that assertion! Doing away with the need for transitional music greatly simplifies the planning task.

**Additional Simplifications**

In addition to the simplifications provided by the theme and variations form, two more design choices helped to reduce the problem early on: music only for solo piano, and manual voice separation.

At first glance it may seem almost trivial to make a computer write for many instruments, given that it can write for one. It turns out that many human composers have had considerable difficulty doing just this! Writing music for an ensemble or orchestra adds many dimensions to the composition problem, and in turn its solution space. The program would need information including: the range of the instruments, their relative loudness and their various idiosyncrasies (for instance violinists can play two or three notes at once, but not *any* set of two or three notes -- and they can't play them as cleanly or as fast). More difficult still, the computer would need some way to characterize each instrument's sounds, and be able to put together these sounds with the goal of producing a particular effect. This is a very abstract and complicated problem, not to be solved anytime soon.

By restricting input (and future output) to the piano, things are considerably simpler. The piano is very close to an abstract music instrument, by which I mean that it is often suitable for playing any music (written for piano or not). Reasons for this claim include its very large range, relatively uniform sound throughout its range (compare this to a flute, which warbles on low tones and becomes shrill in the upper registers), and

wide array of musical effects (from percussive notes to flowing melodies). In addition, synthesizers generally produce high-quality piano sounds, making piano a good choice for computer-music output.

The other great design simplification is a bit of music pre-processing by the user. This is similar to the work the user must do to add to the sample library in EMI (discussed above). Specifically, it falls upon the user to separate the voices among their functional groups. By this I mean that the main voice always be designated voice 1, and each background voice be uniquely numbered. Thus TV-PR always knows what the foreground melody will be.

TV-PR uses the channel information in the MIDI input to determine to which voice a note belongs. I chose the channel because it is readily accessible to users from most MIDI editing programs. Also, since only piano input/output is allowed, the actual channel information is superfluous.

## Existing Formal Music Theories

### Lerdahl and Jackendoff

In an influential book titled A Generative Theory of Tonal Music, a set of theories about many aspects of music are developed. The end result is several sets of rules governing the way music is put together. Most every part of the theory involves building a binary tree structure showing the grouping of notes at various levels. The book additionally showed how a reduction of the material akin to Schenkerian analysis might be performed with similar rules.

Each rule set had two sections: Well-Formedness rules and Preference rules. The first are hard constraints that basically define the structure being built. The Preference rules act as tie breakers when it becomes uncertain which of two groupings is correct. The preference rules are not of the easy kind to put into a computer. For example: They advise to prefer structures which could be construed as parallel over other likely structures [Lerdahl & Jackendoff 1983].

The work of Lerdahl and Jackendoff is important because 1) it takes an approach that somewhat parallels Schenkerian analysis, which is very popular 2) while not strictly formal, the strict structures and algorithmic approach to building them are more-or-less computer friendly. In fact, the theories have been built into a Prolog program for explaining music [Widmer 1992].

**Eugene Narmour's IR Model**

Eugene Narmour approaches the problem of musical grouping from a psychological angle. His The Analysis and Cognition of Melodic Complexity has numerous examples showing that it is plausible that the brain functions as Narmour says it does. Further, the model he puts forth achieves great stylistic independence, being tested on a huge variety of material.

The essence of Narmour's theories stems directly from its name, the Implication-Realization (IR) model. Its one premise is that humans experience music through expectations. If the music implies something, and then that thing happens (it is Realized), then we are pleased. If the implied action does not happen, we are dismayed. According to the IR model, it is this push and pull of expectation and fulfillment that

drives music. And the relationships are delightfully simple and familiar. For instance, notes going a small interval tend to keep going in the same direction. Notes taking a large interval imply a reversal (that is, large jumps want to turn around!). With just a few of these relationships, any series of notes can be discussed in terms of their implications.

Though it sounds overly simple, the books and the evidence therein are very convincing. Plus, the basic structures involved are easily fed into computers. Dr. Markwin Van Der Berg's Delta Framework software also performs the basic IR categorizations as a non-essential feature (seemingly for fun). Unfortunately, as with Lerdahl & Jackendoff, to really implement the full theory (which includes the ways the basic structures build high-level chains) on a computer one has to deal with much harder issues. For instance, a strong reversal in the direction and interval of the line can produce a sense of closure, provided (among other things) that it is not in the middle of a harmonic progression. Teaching the computer what it means to be inside a harmonic progression is not trivial at all.

Still, the theory is on strong psychological ground and is a unique and interesting look at musical grouping. The Prolog implementation of Lerdahl & Jackendoff was considering switching to an IR implementation [Widmer 1992].

**Joseph Schillinger**

Joseph Schillinger was an award-winning Russian composer and teacher. After defecting from the Soviet Union in 1928, he set up permanent residence in New York, where he tutored and advised such notables as Gershwin, Benny Goodman, Glen Miller, Tommy Dorsey, John Cage, and Earl Brown. Though largely forgotten today, Schillinger

left behind a huge (1,640 pages) compendium of music theory based entirely in mathematics. Not only did it touch upon all areas of composition, it was style-free. Schillinger described the workings of American jazz and swing on one page and gives examples of Beethoven and Wagner on the next -- all in reference to the same theory! This stands in stark contrast to conventional, subjective theories culled exclusively from common-practice music.

One might ask, then, why Schillinger has fallen into such obscurity. Some claim it's the mathematical nature of the work. Another possibility could be that even the mathematically inclined are turned off by the theory because Schillinger's terminology is confusing and out-of-sync with mainstream mathematics. Schillinger's evangelical stance and references to the "mistakes" of the classical masters most likely also damaged the credibility of the theory.

The appeal of the Schillinger System of Musical composition for computer-based music is clear from the first page. The mathematics and graphs on which Schillinger's theory is based translate readily into code. Schillinger was disgusted by composers who worked by trial-and-error, but he did not intend to reduce composing to a mechanical process: "My system does not circumscribe the composer's freedom, but merely points out the methodological way to arrive at a decision. Any decision which results in a harmonic relation is fully acceptable." [Schillinger 1978]

Though mentioning the whole of the topics covered in the Schillinger System would be time and space prohibitive, TV-PR clearly shows Schillinger's influence throughout. Here are some of the most important concepts and ideas:

**Idea 1: Interference of Periodic Waves**

This concept forms the basis of the entire Schillinger System. It has a variety of uses. For instance, interference can create duration patterns for groups of notes, relative time spent presenting music in certain formats (such as types of inversions), and much more. Once a composer gets used to thinking in terms of periodic waves (called 'generators' from now on), it becomes second nature.

An example may be instructive at this point. An example for note durations follows. Mathematically inclined people will recognize this procedure as analogous to producing a composite wave from its Fourier series. Start with a periodic note attack, say 1 attack per 2 time units. I'll call this a monomial generator. Alone, it produces a rhythmic pattern 2 + 2 + 2 + 2 + etc. That is, a continuous set of attacks, each lasting two time units. The length of a time unit is not important at the moment...it could be any constant duration.

Now add another monomial generator to the mix. This one produces an attack every three time units (so it generates duration units 3 + 3 + etc.). By starting both generators at the same time, and noting the distance between every attack, a new pattern emerges. This new pattern is a generator itself, and is called the resultant, r3-2. The easiest way to find it (though not computer friendly) is to draw the two generators one below the other on graph paper and drop vertical lines from every attack:

| | Units of Time (* indicates a note attack) | | | | | |
|---|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | 5 | 6 |
| Generator 3 | * | | | * | | |
| Generator 2 | * | | * | | * | |
| Resultant | * | | * | * | * | |

For r3-2 we get 2 + 1 + 1 + 2. It can be shown that every rhythmic pattern possible can be generated with this procedure (more than 2 generators can be used at once). A little math shows that generator 'A' will attack 'B' times, and vice-versa. Thus in the above example, generator 3 attacks 2 times and generator 2 attacks 3 times.

Look at r3-2 again: 2+1+1+2. Once a time unit is chosen this can be directly translated into a series of notes such as 1/4 - 1/8 - 1/8 - 1/4 or 1/8 -1/16 - 1/16 - 1/8. The list of numbers is just the proportions of the unit. Note also that it is symmetric: (2+1) + (1+2). All of the resultants have this property. For example r4-3 is (3+1+2)+(2+1+3).

**Idea 2: Concept of Balanced/Unbalanced**

The second important idea from Schillinger's work involves an analysis of the intervals created by generators. The concept is simple: Like intervals create balance, and unlike intervals create imbalance. The less similar the intervals, the more unbalanced the system becomes.

Though simple, this idea applies throughout TV-PR. Consider a rising set of notes with pitches [C C# D D#], which has the intervallic relationship [+1 +1 +1]. This

has a standard deviation of 0, and is thus a predictable and balanced rise. Now assume this same set of four notes has durations [1/8 1/4 3/8 1]. There is quite a jump from an eight note to an eighth note (C) to a whole note (D#), and these notes are rhythmically unbalanced. Similar judgements could be made on the other properties of the notes (such as their volume).

[Berry 1987] argues that one of the keys to understanding music is to see it as a pendulum moving from balanced to unbalanced areas. Using Schillinger's definition of balance, computers can identify this shift easily (with respect to the discrete measurable values of the notes).

**Idea 3: Expansion and Contraction**

Like the ideas of Balanced and Unbalanced above, expansion and contraction apply to all aspects of music, and at all levels. For the purposes of this discussion, I apply it only to the pitch of the notes in a phrase. There are two types: Geometric and Tonal.

Geometric expansion/contraction of a melody is easy to understand. Pick a pitch for use as a key-center. Ideally this choice would have some rationale from the melody itself (later in this paper I describe the method TV-PR uses). Then, label all the pitches in the melody in terms of semitone distance from center. Thus for center C, D is 2 and E is 4 and so on. Now multiply all of these by some factor. Natural number factors such as 2, 3, and 4 geometrically expand the melody. Fractional factors like 1/2, 1/3, and 1/4 contract it. Note that while a melody may always be expanded, contraction is not always possible with a 12-note chromatic scale (what tone is 1/2 of 3 semitones?)

Tonal Expansion of pitch is a bit different, in that only steps of a certain scale are used. The scale chosen has an effect on the outcome, but the only requirement is that it must contain all of the melody notes. Thus multiple scales can apply to the same melody for the purposes of tonal expansion/contraction.

The easiest procedure for Tonal Expansion is:

1) Rewrite the scale, choosing every other note until all notes are chosen. Thus, the major scale CDEFGAB (1 octave) would become CEGBDFA (2 octave span).
2) Now, to expand, number the melody notes according to the unaltered scale. To contract, number the notes on the expanded scale.
3) Then the corresponding notes from the other scale form the expansion/contraction.

A second expansion involves taking the original scale and taking every third note. Note that expanding the first expansion gives the third expansion--*not* the second!

Neither geometric nor tonal expansions/contractions change the overall contour of a musical characteristic. Instead, the contour is merely amplified or attenuated. Schillinger points out that expanding a melody with either of these methods tends to 'modernize' music. Thus Mozart expanded once sounds more like Debussy. Expanding twice sounds even more modern.

## Preliminary Work

### 'Particle Variations'

Based on the success of the mathematical Schillinger system, I became attracted to the idea of 'art imitates math.' The result was a failed effort at producing natural changes to melodic lines. Because the basis for the model was forces on a particle, I dubbed the work 'Particle Variations.' Basically, the notes of the theme were given masses and charges, and positions relative to each other proportional to their playing time. Then a particle (with its own mass, charge, and initial velocity) is sent into the theme's space. The trace of the particle becomes the melodic variation.

I thought that with proper extra forces throughout the space I could force the particle to behave and produce a new curve. Ideally the new curve would be similar to the theme's but varied in some way wholly dependent on the initial conditions.

Unfortunately this did not occur. When the particle didn't fly off into infinity, it oscillated regularly around the theme. Perhaps this is good for one type of (fake-sounding to say the least) melodic variation, but it is hardly a complete solution. Appendix B has an example of the output.

### Pure Schillinger Variations

The second preliminary work I did leading up to TV-PR was a few sets of variations (on pencil and paper), following Schillinger's System to the letter when possible. The goal here was to use no human intuition, to make no choice that could not be traced with some programmable logic. As the computer would do, I first planned the

set in detail. I was immediately impressed with the ease with which these plans jumped onto paper.

When I tried to make a decent realization of these plans, however, I came into considerable difficulty. In the end, both the power and the potential failure of Schillinger's System is its great generality. Within Schillinger's System, any combination of notes is possible. Schillinger only asks that there be some underlying logic to it. The failure of this approach is that it requires an overlying intelligence that holds it to the desired style.

Oddly enough the resulting music sounds a lot like purely random composition. Though an underlying logic exists, the individual sections are similar to existing styles to a large enough degree that the whole seems scattered. TV-PR tries to avoid this by simultaneously forcing Schillingerian logic onto the piece and also putting controls on which combinations and sequences of elements are acceptable.

## TV-PR Dissected

### The Natural Language Analogy

The similarities between music and language run deep and are well documented. As mentioned above, the correlation between the two is so tight that music generation algorithms have been based on algorithms for generating poetry [Cope 1987]. Of particular interest for many are generative grammars, which can make structurally correct

sentences out of a few words and rules. Adapting grammars to music is commonplace in computer music research.

Unfortunately, the focus of the majority of these adapted techniques is the sentence. This is the root of the continuity problem in computer-generated music. An approach that focuses only on making coherent sentences (no matter how effective) will almost surely fail to make coherent paragraphs. In the same way, a list of coherent musical statements does not necessarily make a coherent composition.

**Content Planning**

In the realm of natural language generation, the large-scale coherence issue has received a great deal of attention. One popular approach (used by systems such as TEXT, KDS, and Penman [Meteer 1992]) is depicted in figure 1. The generation process is split into two components, which I will call the Content Planner and the Realizer. Their roles in text generation are usually along these lines:

- Content Planner – Decide what information to express. Decide the order in which the information will be expressed. Choose appropriate content to for the current goals of speech (intended effect, conciseness, etc.).
- Realizer – Organize the individual sentences according to grammatical rules. Choose actual words based on the content decisions of the planner.

The content planner addresses the continuity problem because the relationships between the sentences are logically arranged and accounted for. In addition, this

framework allows a language generator to have a goal that it wants to accomplish across a paragraph (such as changing the reader's opinion, or padding bad news with preliminary statements). Usually all of the planning takes place before any of the sentences are written, so the two parts are somewhat independent in most implementations.

**A Music Content Planner**

Despite the similar techniques underlying music and text generation, I am unaware of an existing music program to implement a content planner. Given the success of planners for language, a music planner would be a logical step to take toward large-scale coherence. Outlining a music framework with a planner is simple:

- Music Content Planner – Decide how many themes to use. Decide an appropriate order in which to express the themes. Choose appropriate music properties to accomplish musical goals.
- Music Realizer – Produce each musical fragment according to the planner's description and locally sound music construction.

The first two goals of the content planner are straightforward. The meaning of 'appropriate music properties' may not be as clear, though. An example would be helpful here.

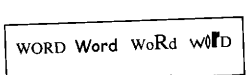
A composer decides to write a minuet. Because of the conventions of minuet writing, she must choose 2 themes in the order AABABA (repeats written out). This is

far from a complete plan, though. How loud is theme A? In what register is its melody? Is the rhythm of the harmony smooth or jerky? These are the types of qualities that need to be defined in the planning process.

Note also that each instance of theme A could have a completely different set of properties. This implies that planning continuity is a two-dimensional problem: the properties of each theme must be coherent, and their collective properties as the themes change over time must also be coherent.

**New Issues to Face**

There are a few roadblocks to using the planning framework for music that aren't issues for language, however. The first and most difficult of these problems concerns the meaning of a musical fragment. Written words have one or more definite meanings and connotations (both context-dependent). Groups of music notes also have meanings and connotations, but they are not well defined or categorized. Also it would seem likely that the interpretation depends much more on the interpreter than is the case with language. What is the meaning of a certain combination of music notes in one setting and how does it change in another? There is too little information to definitively answer this question (though generalities can be made, as will be discussed).




Another aspect of this problem can be described:

Each of these is easily recognized as the same word, and understood to stand for the same concept. Sets of the same four music notes with different properties (volume, timbre, etc.) could be recognized as fundamentally different, especially in context. Unfortunately, the music theory literature on this subject is sparse (except for admissions that the conundrum exists -- for instance [Williams 1997]).

Yet another issue to be addressed is the concept of musical rules. A language realizer creates well-formed sentences according to a finite set of accepted models. There is no direct analogy for music. For one thing, different musical styles can be (and often are) compared to different languages, each with its own set of guidelines.

Beyond that, even if we choose a language and painstakingly define it as a model of 'correctness,' we may have a conflict of interest with the goal of creating art. Certainly a human composer who painstakingly copies the style of another isn't praised or considered artistic for doing so. The ideal for an artist is to have a language of one's own. This makes qualifying music as 'good' or 'bad' difficult since comparing it to a model seems self-defeating (to a degree), while evaluation without comparison seems impossible. How can a realizer know if it is doing satisfactory work? Further discussion of the difficulties presented by the realizer follows the description of the planner.

**Blackboard Paradigm**

Music, like language, can be understood on several conceptual levels at once. One might think of an entire symphony and describe it in an abstract sense. Perhaps a particular symphony is conventional, long, for large orchestra, and overall very dissonant. Then on a deeper level, the first movement of the symphony could be described as moody, with a strong rhythmic drive. At deeper levels one might describe the qualities of

the sections of the movement, then the phrases in those sections, on down to the actual notes.

A great model for problems that can be broken into levels like this is the Blackboard model. The idea is to create agents (called knowledge sources) that each have a skill relevant to solving the problem at hand. All the agents monitor the Blackboard (a shared resource initially holds a description of the problem). Whenever a knowledge source notices that it has something to contribute, it adds a partial solution to the blackboard. Ideally, this will trigger other knowledge sources to act and move this partial solution further. Blackboard models also scale up to multiple-processor machines well, due to the independent behavior of the knowledge sources.

The appeal of the Blackboard system in the case of a music planner is that the plan evolves on multiple conceptual levels at once. The DIOGENES natural language generator utilizes a blackboard model for this flexibility [Meteer 1992]. An important (but difficult) task is to keep track of how local choices affect the piece of music as a whole. A Blackboard implementation facilitates this by creating knowledge sources that watch the solution on one level and update the solution on another.

The diagram on this page shows the Blackboard model for TV-PR. Each deeper level has more specific information regarding qualities of the music over time. The design is also such that the system can give a synopsis of its plan at various levels of detail; it only discusses items of the appropriate depth in the blackboard.

**V-PR's Blackboard Model**

Level 1: Abstract
Contains:
1) The Mood of the variation.
2) The designation of which stylistic tendencies to follow.
3) The overall deviation of this variation from the theme.

Level 2: Semi-Abstract
Contains:
1) Any stereotype to follow (such as a waltz feel)
2) The major variation strategy (such as varying the rhythm).
3) The phrase structure of this variation (such as ABA etc).

Level 3: Semi-Refined
Contains:
1) The Tempo, Rhythm, Volume, Texture, etc. distributions.
2) The measure of dissonance

Level 4: Refined
Contains:
1) The harmonic and textural map of the variation
2) The motifs (rhythmic and intervallic) allotted to this variation.

Level 5: Music
Contains:
1) The MIDI input data.
2) The MIDI output data.

Level 6: Decomposition
Contains:
1) A breakdown of the phrasing in the input.
2) A pool of the musical resources identified in the input (motifs, notable features, etc.)

In the majority of Blackboard implementations, the solution migrates from level to level in one direction. For example, consider a speech recognition Blackboard that identifies sounds, then phonemes, then words, then phrases, and finally sentences (the output). One interesting feature of TV-PR's Blackboard is that the input is on the same conceptual level as the output (on Level 5). In fact, the input forms the first part of the output! Instead of a general flow from one end of the Blackboard to the other, the solution path is in three disjoint steps.

First, the input is analyzed and decomposed into a set of musical resources on Level 6. These resources represent the material used to build the variations, and include elements such as the salient melodic features, the overall volume, etc. Second, a plan is

formulated across all of the upper conceptual levels (Levels 1 through 4). This process is discussed in detail below. Finally, the realizer knowledge sources generate the actual music from the plan.

## Overall Solution Process

The next section of this paper describes the steps TV-PR takes to create a plan. There are 3 steps:

1) Decompose the MIDI input

2) Generate an Abstract Plan

3) Generate a Concrete Plan

### Decompose the MIDI input

The first step in the planning process is decomposing the MIDI input. The goal here is to select what I call the 'salient features' of the input theme. As the discussion below will explain, some features are easier to extract than others are. Each feature is extracted by a knowledge source, and becomes an entry on the Blackboard. The features TV-PR tries to capture are:

- Key
- Mode
- Volume
- Harshness
- Voice Separation
- Phrasing
- Texture
- Rhythmic Feel
- Harmony
- Syncopation
- Mood

- Stylistic Conventions

**Key**

TV-PR's notion of key (like many of its notions) is very simplistic. Since TV-PR was designed to be style-independent, I avoided implementing some concept of a traditional tonal key. Instead, I borrowed the idea from Schillinger that the key center of a phrase is the pitch class with the most weight. In the early versions of TV-PR weight was defined by total time audible. Later I got better results by scaling the time weight by the volume of the note:

weight = (time * (volume/MAX_VOLUME))

In the event of a tie one of the centers is chosen at random. A possible future improvement might be to adjust the weight of a note based on its metric emphasis.

Of course this method will sometimes disagree with the conventional choice for key, but it works regardless of style and may actually be a better indicator of the phrase's nature anyway. For instance, if a phrase has C major as the first and last chords, and the penultimate chord is a G7, one might immediately claim that the phrase is in C. However, what if the pitch with the most emphasis and playing time is F? TV-PR says this is an overriding factor.

In addition, TV-PR records the mode of the input. Instead of trying to fit a traditional mode to the input, the program takes a more liberal view of musical modes. The three choices are Major, Minor, or Neither. The overriding factors are the types of third, sixth, and seventh degrees in the scale (relative to the chosen key-center). Thus a piece in the dorian mode would be classified Minor by the analysis. Atonal works will ideally be called Neither. I feel this is a good compromise between the more exact

designations possible for tonal works and their less-appropriate nature for more modern pieces.

**Volume**

TV-PR treats volume simply as well. MIDI notes in the input file all have a volume level (called the velocity). It is an integer between 0 and 127. The analysis on volume in the input MIDI file is done by simple statistics. The average and the standard deviation of the volume are taken.

**Harshness**

The explanation here is long, and deservedly so. Harshness is one of the most important aspects of music in TV-PR. It is interesting, being considered both a basic element of a sound (like its volume), and a subjective phenomenon (like the mood of a piece). From the beginning, I planned to use fuzzy sets to determine the consonance of a set of notes. The basis of this approach is the natural overtone series produced by acoustic instruments, and the approach is extremely versatile.

As described in [Vidyamurthy & Chakrapani 1992], a note can be represented as its degree of participation in producing all possible pitches. The motivation for such a model comes from acoustics. To illustrate, when middle C is struck on a piano, more than just that C note sounds. Also produced are the C above it, and the G above that, and the C above that, etc. in a set pattern. These extra notes are called *partials*, and each successive partial is generally softer than the last. The relative intensity of the base note and its partials is different from instrument to instrument (more on this later).

Incidentally, the pattern of notes that the partials normally take is a simple one. To find the frequency of the Nth note in the pattern, multiply the base frequency by N. Note that by this equation, partials of most notes become very high in pitch very early in the series. As a result, TV-PR need not use many partials (after all, the humans can't generally hear pitches higher than 30 kHz).

TV-PR uses the base note and its first 15 partials. This suffices for close notes (since higher partials are very soft and more than 4 octaves above the original note). Using more memory and computation time, this amount can be extended at will. The reasons one might want to do this will become evident during the following discussion.

For now, though, an example might clarify the setup. A middle C will be mapped onto the set of all pitches such that:

1) the note has membership in note C, C+12 semitones, +19 semitones, +24, +28, +31, +34, +36, +38, +40, +41, +43, +45, +46, +47, and +48 semitones. (these notes correspond to the closest note to the frequency of the actual partial--more on the difference between frequencies and notes below)
2) the base note has the most membership (read 'a higher value'), and it decreases exponentially as the partials get higher.

Purists might complain that the notes in these sets are only equal-temperament approximations of the actual partial frequencies. While it is true that the frequencies of equal-tempered notes do not exactly match the partial frequencies, they are close enough to be identified as equal. Also, the original research with this system produced good results, so I see no reason to complicate things.

Using this model, [Vidyamurthy & Chakrapani 1992] outline how one can compute the consonance of a set of tones. Two operations are involved:

1) Find the intersection of the fuzzy sets (the notes). For this application the intersection is defined as the minimum value of the two for each pitch value. If note 1 has 0.50 at pitch 3, and note 2 has 0.00 at pitch 3, then the intersection has 0.00 at pitch 3, and so on. Note that the intersection has the same structure as a note--it is simply the interference of two notes.
2) Sum the membership values of this intersection across all pitches. This is the degree of consonance.

A little experimentation shows that this system produces results consistent with reality. The most consonant interval is the unison (a note played with itself has no dissonance). In accordance, this system gives a unison the maximum possible consonance value. It may be instructive to see how. The two notes are identical, and thus have the same membership in the set of all notes. Thus in step one, taking the intersection leaves us with a set equal to the original note. Now step two sums the membership values of the intersection. Since any other intersection must have membership values less than the original note, then the sum for our unison must be the maximum possible consonance.

By making test runs of many intervals (including those spanning multiple octaves), I found mostly-predictable results. [Vidyamurthy & Chakrapani 1992] showed similar results. Note also that the system is easily extended to handle more than two notes. Thus the consonance of a whole group of tones may be judged.

The system (as described) is not perfect, though. For instance, note that Dissonance is not simply the inverse of consonance, despite their common use as antonyms. At first I considered this an error, but introspection and research now convinces me otherwise.

One example will suffice: Perceived dissonance decreases with octave separation. For instance a second is less dissonant when reformulated as a ninth (try it on a piano if this assertion seems suspect). However, in the current fuzzy-set system octave separation tends to give *less* consonance. That means that an inverse notion of dissonance tends toward *more* dissonance with octave separation! This is not the desired result! My answer to this paradox is to do away with the semantics of the issue, and form a new viewpoint on note interaction. In it, consonance and dissonance aren't inverses at all. With octave separation, intervals become both less consonant *and* less dissonant (read 'the notes are less related'). From this viewpoint closeness gives notes a potential to be harmonious or clash, and distance does reduces this potential.

The above description is the limit of the fuzzy-set method's utility in TV-PR. As an aside I'd like to allude to its extreme versatility here. As stated earlier, different instruments have different sonic profiles (not only with respect to the strengths of their partials but also with the way these partials change in strength over time). The fuzzy-set system can use this information to accurately combine notes from different instruments. The code simply uses the fuzzy set for the appropriate instruments for the intersection step. This way, if an instrument has weak upper partials then these will contribute less to the result. The end result is a program able to decide if minor 2nds sound better between

two violins or a violin with a flute. Information such as this would be invaluable to a program trying to orchestrate new (or even existing!) music.

**Phrasing**

The object is to take a string of notes and determine where the phrase boundaries are. Here I used an algorithm called LBDM, or Local Boundary Detection Method. It is based on an article by Emilios Cambouropoulos ([Cambouropoulos 1996]), and is surprisingly simple to grasp. The algorithm basically looks at a number of raw parameters on the notes, and either says "this parameter changed," or "this parameter has not changed." The different parameters have weights so that some matter more than others, but in its simplest form the algorithm simply counts up the votes for each notes as the potential boundary.

At that point TV-PR has a string of numbers. How can this information be used to determine the phrase boundaries? Based on the previous paragraph, one might jump at the maximum value in the list. But there could be three or even five phrases in the input! And on top of this, the global maximum may not actually be a cutting point between phrases! If a human were to draw the lines, (s)he might get a sense of what the local maximums are for each section of the list, and then choose large numbers that fall on more or less regular intervals. Preference for regularity hearkens back to Lerdahl and Jackendoff's rules [Lerdahl & Jackendoff 1983]. Of course TV-PR would only ignore the maximum if it is very close to the 'better,' regularly spaced value nearest to it.

The fact is that these complications only get worse as more cases get examined. Because of this difficulty I have temporarily dropped the idea of TV-PR breaking the input into phrases.

However, the LBDM method is still invaluable to TV-PR in another form. All of the local maxima are used to cut of the phrases into motifs (short strings of notes that carry more meaning than they do individually). Every identified motif is broken into its rhythmic and intervallic components and stored in what I call the musical resource pool.

What is this good for? TV-PR already knows the notes and contour of the input melody in its entirety, by virtue of having MIDI input. With a decomposition into a pool of note combinations which, by definition, carry a meaning, TV-PR can project this meaning onto the rest of the phrase. Not only is the idea a familiar one in music, but the simplest procedure for accomplishing this in a realization of the piece also bears resemblance to a human composer's activity. Later in this paper, it will be shown that the pool of resources culled from the theme is the key to the Concrete Plan generated by TV-PR.

**Texture**

Texture refers to the number of active voices. [Berry 1987] makes the distinction between voices acting together and voices acting independently. For example, if there are three voices at some moment but they have near identical rhythm and direction, then the texture is really 1. TV-PR also notes this distinction, saving the number of voices as RAW_TEXTURE and the amount of coordination between the voices in the raw texture as NOTE_COORD.

Raw texture, like Volume, is treated as a statistic. Three entries are placed on the blackboard: The minimum, maximum, and standard deviation.

**Harmonic Map**

TV-PR creates a map of the harmony used in the input. Four pieces of information are kept about each chord: the root, the degree, the bass note, and the duration. The degree of the chord refers to the number of unique pitch classes in the chord. Thus a G major chord would have degree 3, and a G7 would have degree 4. Note that this also works as a naive judge of the tension in the chord (even in non-triadic music).

For generality's sake I neglected several pieces of information only relevant in the tonal world. Only one concession to tonality is made: if the pitch classes can be arranged in a triadic configuration, then I record the root. Otherwise, the root is given the same value as the bass note.

**Rhythmic Syncopation**

This is just a check for strong notes that fall on weak beats. Of course this doesn't completely capture the meaning of 'syncopation,' but the information is very useful.

The input MIDI file must have its notes framed in the proper time signature at all times for this to be accurate. For example, it is common to frame a whole piece in duple 6/8 even though it really has a few measures of 3/4. This makes no difference to the listener, and is easier for the performer to read. TV-PR, however, uses the time signature to determine which beats are strong and which are weak (a la [Lerdahl & Jackendoff

????]). Fuzzy membership values are used to map a note's location in a measure to its metric strength. This makes the algorithm more robust when the input MIDI notes aren't exactly on the beats (for instance, if it contains *rubato* effects).

**Mood**

Mood is a very high-level concept, and difficult to ascertain from a set of notes. The largest problem is its inherent subjectivity. Features such as mood and harmonic style (discussed below) are just one step away from assigning a meaning to the notes. But, as [Williams 1997] says, "Since the same [specific musical device] can produce a very different effect at the same point in two pieces...giving it any further label might only impose a programme." In other words, since two people may associate the same segment of music with two different moods, assigning a mood might be the wrong way to go. Perhaps one day computer programs will be sophisticated enough to produce an opinion as valid that of a human. This won't be possible without advances in cognitive psychology, though.

TV-PR determines a mood based on a subset of the same rules used to plan (see the planner section below). For example,

IF Rhythm[JERKY] THEN (Mood[EXCITING] OR Mood[SURPRISING])

would be one of the rules. Points are accumulated based on the fuzzy degree of confidence in the antecedent. So if the rhythmic feel were jerky with a .85 confidence,

then Mood[EXCITING] and Mood[SURPRISING] both gain .85 points. After each rule has fired, the mood with the most 'points' wins.

**Harmonic Style**

Harmonic Style is another abstract, subjective aspect of the music. For this reason, the judgement of Harmonic Style enjoys the least success of all the analysis done by TV-PR. Basically, the knowledge source looks at the average Harshness, the Key

## Generating the Abstract Plan

**Art vs. Optimality**

This section describes the operation of the Abstract Content Planner. As described above, the Planner is made up of several knowledge sources working in the upper conceptual levels of the Blackboard. They each work independently to produce one facet of the musical plan (the actual elements making up a plan will be described shortly).

The planning process has been formulated as a fuzzy constraint satisfaction problem. The planning knowledge sources in effect search a large solution space for valid combinations of plan elements. Plan elements denote the themes and the musical qualities discussed above. Each active plan element has a presence on the blackboard, and an associated strength rating. Thus, for example, two moods could be present with one dominating the other. A few examples of these qualities are given in the box at the top of the next page for clarity:

**Example Planning Elements**

Mood Elements: CALM_MOOD, EXCITED_MOOD, etc.

Style Designation: CLASSICAL_STYLE, TWENTYCENT_STYLE, etc.

Feel elements: TEMPO[0-999] in beats per minute
AVG_TEXTURE[1-20], MAX_TEXTURE[1-20] in number of voices.

Unfortunately a standard search will not do for a musical (or any artistic) plan. In fact, there are many obstacles to overcome in terms of 'artistic' searches through a solution space. The algorithms in common use (and the optimizations that make them tractable) are geared toward finding the best solution. To mimic creativity, that's the last thing we want. We want to find any decent solution and find a different one every time we start over.

Another point to note is that any 'optimal' solution to an artistic problem is a result of program logic, not art. That is why there isn't one perfect novel, and why no painters worry about finding the optimal solution to the bowl of fruit problem. In other words, there is no perfect painting, but a computer might derive one from a set of rules in a knowledge base. That's not as much an artistic achievement as it is proof that the set of rules is too strict or focused on certain stylistic considerations.

In an attempt to overcome this obstacle the following guidelines were followed in constraining the combinations of planning elements:

1) Identify the bad combinations. There are fewer bad choices than good ones, so there will be fewer constraints to process.

2) Only stop combinations of elements that are always bad. For instance, Volume(Loud) and Volume(Soft) can clearly never coexist in the same musical fragment. Use constraints to eliminate this combination. Try to avoid

style-motivated constraints (such as the common practice tendency for dissonant notes to resolve downward) at all costs, as this limits the potential output of the system. Style is handled elsewhere.

3) Differentiate between amounts of error in breaking one of the constraint rules. Some mistakes are more critical than others.

Following the guidelines above, a set of bad combinations of elements was compiled. Each of these has an associated error, which accumulates as bad choices are added. An example would be:

```
CALM_MOOD * (TEMPO[FAST] + RHYTHM[JERKY]) ; 3.
```

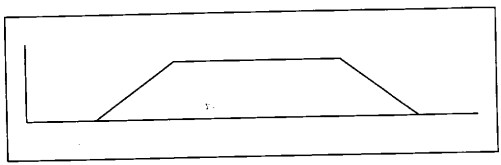
where * is a logical AND and + denotes logical OR. This mistake has an error strength of 3.

**Constraint Capabilities**

TV-PR has the ability to read and apply constraints in the form given above, but it has much more capability than is implied by the example given. With TV-PR, constraints are:

- Fuzzy. For the given example, Tempo[FAST] and Rhythm[JERKY] are fuzzy membership sets. Other examples include Tempo[VERY_SLOW],

Tempo[MODERATE], Tempo[VERY_FAST], and Rhthm[SMOOTH]. All membership sets in the Constraint solver take the form:



because the computations are faster. Triangular sets can be computed by making the middle section come to one point. Nonlinear (S-shaped) membership sets are used for metric emphasis decisions in TV-PR, but not in the constraints or preferences.

- Able to reference relative variations: Note that the example above only applies to one variation. This used to be the limit of TV-PR's capability, but it soon became apparent that the planner needs to reason about adjacent variations as well. So we can have:

  ( EXCITING_MOOD[o(-2)] * EXCITING_MOOD[o(-1)] *

  EXCITING_MOOD[] ); 300.

  meaning "Allowing any degree of exciting mood for three variations in a row entails an error of 300 on the third variation."

- Able to reference the theme: Eventually it also became clear that variations would need to make reference to the theme's characteristics. This is mainly to keep tabs on just how different we are from the theme. For example:

(CLASSICAL_STYLE[t()] * TWENTYCENT_STYLE[] *

DEVIATION_FROM_ORIG[LOW] ) ; 50.

meaning "Given that the current variation's deviation from the theme is to be low, then it shouldn't be in the modern harmonic style if the theme is in the classical harmonic style."

A sample from the fuzzy constraints is given in Appendix C.

**Implementing Style**

One might wonder how well a system without stylistic considerations can perform. Wouldn't the chances that the resulting music sounds bad to our ears be overwhelming? What if music in a certain style is the desired output? Shouldn't there be a way to reinforce good selections?

To handle these considerations, another set of rules was produced. These represent conventional preferences that are in agreement with a specific style, and called 'proven choices.' These 'proven choices' have associated points, which are accumulated in the same way as error points. Thus, at any moment the Blackboard has information about how much the solution is out-of-bounds (error) and how much the solution is conventional (proven choices).

Each style of music has its own set of conventional preferences, and one or more of these can be used in any planning process as desired. Without proven choices, the system only avoids patently bad selections (deemed to never be reasonable).

(CLASSICAL_STYLE[t()] * TWENTYCENT_STYLE[] *

DEVIATION_FROM_ORIG[LOW] ) ; 50.

meaning "Given that the current variation's deviation from the theme is to be low, then it shouldn't be in the modern harmonic style if the theme is in the classical harmonic style."

A sample from the fuzzy constraints is given in Appendix C.

**Implementing Style**

One might wonder how well a system without stylistic considerations can perform. Wouldn't the chances that the resulting music sounds bad to our ears be overwhelming? What if music in a certain style is the desired output? Shouldn't there be a way to reinforce good selections?

To handle these considerations, another set of rules was produced. These represent conventional preferences that are in agreement with a specific style, and called 'proven choices.' These 'proven choices' have associated points, which are accumulated in the same way as error points. Thus, at any moment the Blackboard has information about how much the solution is out-of-bounds (error) and how much the solution is conventional (proven choices).

Each style of music has its own set of conventional preferences, and one or more of these can be used in any planning process as desired. Without proven choices, the system only avoids patently bad selections (deemed to never be reasonable).

Note that, in the natural language analogy, these proven choice rules profile the reader of the text. Many natural language generators (such as PAULINE [Hovy 1988]) take into account the profile of the reader. This information is used to, among other uses:

1) Avoid telling the reader something he or she already knows.

2) Avoid insulting the reader (or, rarely, the opposite).

3) Produce text at the desired reading level.

The musical preferences represent an audience that likes to hear music with certain specifications. Profiles for many styles of music can be generated and used to guide the planner and, eventually, the realizer.

**Searching with Personality**

Now the actual constraint satisfaction process is described. As discussed above, we are looking for an artistic solution. So instead of choosing an 'optimal' order to select the variables for instantiation, they are chosen in random order. Instead of trying the optimal choice for the values (such as the choice that constrains the future choices least), this is also a random process.

One might worry that a problem with so many variables (an average of 20 per variation—with up to 32 variations) cannot be solved in reasonable time by choosing solution points at random. The system isn't overly constrained, so a random choice has a reasonable chance of hitting a good spot. Also, the choices aren't completely random, due to the searching mechanism described presently. The motivation for the search mechanism implemented can be conveyed easily with these four realistic composition situations:

1) A composer sets out to write a piece of music with the idea that the volume will slowly increase from near silence at the beginning to the loudest notes possible at the end. Everything else is undecided, but the composer plans to build the entire piece on this idea.

2) A certain composer is known for writing music having a very logical thematic structure. Paradoxically, this composer also changes key at seemingly arbitrary times.

3) A composer who works for hours on end to find the perfect part to fit the partially written music.

4) A composer who gives up quickly if an appropriate part doesn't spring to mind.

These are just a few realistic examples of composing behavior. They all show that a totally random search for a plan doesn't capture the essence of creativity in art much better than a methodical blind search. While it's not always best to model the natural phenomenon, for the purpose of creating music, the composer makes a good model.

To humanize the search, a set of personalities that the solver will use was created. Each variable in the planning problem has an associated searcher with its own personality. The description should make it clear how the personalities affect the planning process

There are three facets to each personality:

1) Work Ethic – the amount of work the personality will put into the search.

2) Conventionality – the personality's liking for conventional choices.

3) Scholarship – the personality's commitment to correctness.

Further, these three facets each have three major subtypes:

**Facet 1: Work Ethic**

THE IMPROVISER: This represents the artist that feels that art should flow naturally. If a good solution isn't found within a few random tries, it gives up. Control is released to the scheduler so another variable can be instantiated. This variable will try again later.

STUBBORN: This personality will try a many attempts to find a random solution before giving up. Once it does give up, it behaves just as the Improviser above.

METHODICAL: This represents the artist who feels that there must be a set of values that fits in the current partial solution. If Methodical can't find a good solution in a few tries, it methodically searches the space until one is found (breadth-first).

**Facet 2: Conventionality**

THE TRADITIONALIST: This is the artist who doesn't want to make waves. It prefers selections that are 'proven choices.' Thus any valid combination that does not improve the current solution's 'proven choice' points is rejected (if another solution can be found).

THE INNOVATOR: This innovator is the artist that wants to do things a little different than the norm, but not stray too far away. Innovator prefers a mix of 'proven choices' and original combinations.

THE ORIGINAL: The Original is the artist that refuses to walk the beaten path. Any solution that improves the 'proven choice' points is rejected.

**Facet 3: Scholarship**

THE PERFECTIONIST: This is the artist that refuses to settle for anything less than perfect. It rejects all choices that increase the total error of the current solution. If an error-free choice exists, the Perfectionist will wait for it.

THE SCHOLAR: Correctness is important to this artist, but not to the extent of the perfectionist. This is the artist that believes that sometimes a little error actually improves the product.

THE DELINQUENT: The Delinquent doesn't take errors into account. It makes a choice and stands by it. Of course there is a maximum error built into the program. Should the Delinquent choose something that pushes the error level too high, the program logic still pulls the plug.

Note that some combination of these traits mimics the behavior in each of the real-life composing situations given earlier. There are 3 x 3 x 3 = 27 possible behavior patterns, for a wide variety of possibilities. One might wonder if it's not better to use the same personality for all of the variables in the planning process. The answer is no, as

artists often treat different aspects of their work very differently (and with different levels of proficiency).

When the planning process begins, each variable search's knowledge source is given a personality. It takes on one behavior for each facet. The only restriction is that there can be only one Delinquent per plan, as the amount of backtracking would increase a great deal otherwise.

So, just like a real artist, we might have a program that's a Stubborn Perfectionist Innovator when it comes to his choice of goals, but a Delinquent Methodical Traditionalist when it comes to mapping out the textures.

**A Sample Run Through the Planning Process**

First an attribute is selected, such as the volume. Then a personality is chosen at random to be used for this attribute. Now a mathematically logical pattern (like those prescribed in Schillinger's works) is chosen, and a set of attribute values is mapped onto this pattern.

At this point the prospective entries are placed on the Blackboard, and the new total error and conventionality is computed for the variation set. If these values do not match the acceptable states for the current personality, then a the prospective entries are removed, and the process repeats. If the number of tries exceeds the current personality's patience, then another attribute is chosen. The planner will come back to the failure later. If the same attribute has failed in this way many times, then the planner assumes that a good value cannot be found and starts backtracking. In the worst case, the planner

backtracks until the troublesome attribute is the only instantiated variable, and it almost certainly succeeds then.

The system would be considerably enhanced if more types of symmetric patterns could be generated. As of this writing, TV-PR only uses two types of pattern generators (both prescribed by Schillinger's System).

## Generating the Concrete Plan

At this point in the design process, we have 1) a pool of musical resources, and 2) an abstract musical plan. Creating a concrete plan is as simple as assigning the appropriate resources to the variations in the abstract plan. Or is it?

The problem is, what is the best way to distribute the resources? Keeping my goal of simplicity in mind, I tried two elementary approaches: First-Come First Serve and Round Robin distribution.

### FCFS

This scheme looks at each variation in order and assigns resources to it. A variation with a goal of Exciting_Mood might get assigned to unbalanced rhythms and intervals. A variation with a goal of calm might get very balanced intervals, and rhythms containing longer notes. We don't reuse a resource until all similar resources have been used.

**RR**

This scheme distributes one resource to a variation and moves on. It continues until all variations have enough resources or resources run out (whichever happens second). In practice, this algorithm produces slightly better results because each variation has a better chance of getting it's best match than with FCFS. In FCFS distribution, the first variation gets

**Sample Output of a Concrete Plan by TV-PR**

A couple sample pages from a concrete plan generated by TV-PR are given in Appendix D. On examination, a major design point should become clear: TV-PR's output is in HTML format. Thus plans produced by the program are both easy to browse locally and ready for immediate publication on the World Wide Web. Also, the format makes the output interactive and easy to navigate around.

## Why Realization Isn't Possible (Yet)

This paper has outlined the architecture and implementation of a music planner. To make a fully functional music-writing program, the complimentary realizer must be developed. This is reserved for future work, as several large roadblocks come between a concrete plan and sheet music. As I stated above, the realizer's job is to create locally good music according to the given plan. Though many locally-coherent music generators exist, none of the algorithms involved lend themselves easily to goal-oriented writing.

**Disparity Between Representation and Reality**

In fact there is little information available about accomplishing goals like those put forth in TV-PR. Too much information is lost in multiple abstractions. The reality of music is sound waves travelling through air, causing vibrations in the ear drum. A copy of these sound waves captures all the relevant information about the sound. This is why multiple CD players can produce the same sound from a single CD. What happens in the nerves and brain after the sound enters the ear is not completely understood. In this understanding, though, lies vital information about how music is perceived by its audience.

Unfortunately, TV-PR and many other music programs don't deal with sound waves. Instead, TV-PR deals with MIDI information, an abstraction from the sound to discrete events like notes, instruments, and pitch bends. During playback, the synthesizer replaces the timbre information lost in this abstraction. This is why multiple synthesizers often produce dramatically different results when fed the same MIDI data.

Even more unfortunate is that music theory, the human world's compendium of musical understanding, is based on yet another abstraction -- musical notation. Music notation is similar to MIDI data, but is often even less specific with regard to dynamics and rhythm. In fact, an interesting way to look at a score is seeing it as an incomplete composition. Parameters like the physical qualities of the room and the instruments, the temperature and pressure of the air, the conductor, and the players themselves actually complete it at performance time.

So we have a situation in which the only direct link to the human experience is sound, the direct connection to the computer-composition world is MIDI (or similar), and

the only connection to music theory is music notation. The essence of the realization problem is the absence of an algorithmic way to transform one representation into another without human intervention.

Our specific case involves taking a specification that partially maps the characteristics of a piece in music notation, and generating a good MIDI or sound-wave representation of a performance. Even within this task, there is the problem of the 'proper' form of MIDI file. For instance, a rigid MIDI file with notes entered by hand (or heavily quantized performance) will accommodate algorithms based in traditional music theory. But any actual performance will differ significantly from the score. If the theory works on the score, but the public's actual perception of a piece is based on performance, how does a computer program cope with the disparity? EMI's answer is to take sheet music in and put sheet music out. Then human musicians interpret and perform the music. Without more insight into the nature of music, this seems to be the best answer.

**An Even Bigger Issue**

In the end, even solving the representation disparity between sound and MIDI may not help much. Another roadblock to coherent computer-generated music is that there seems to be a certain (presumably large) array of basic information necessary to compose music. Music generators exclusively use either music theory or a random process to put notes together. However, people with no music training can write wonderful music with neither of these crutches. All computer programs lack this innate sense of "sounds good" and "sounds bad." With this simple judgment, a computer

program could write music simply by testing its output until it produces something pleasing.

Some steps in this direction were taken in [ ] where genetic algorithms are trained (under supervision) to like a certain set of chords, and the trained set forms the "ear" for a prospective piece. The program then tests potential combinations of notes with the "ear" to produce pleasing music. The results are interesting. The approach requires new training for each piece of music, but is more feasible than a more general arrangement. In the long run, though, the completely successful music generators may only be possible by extensively training them from sound samples.

Note that the preceding statement is not meant to reduce the utility of a planner like the one developed here. A system able to listen and judge prospective music could make automatic realization of such a plan possible.

## Future Work

There are several steps left to take before I'll consider the planner of TV-PR complete. Many of these tasks have been described or alluded to in the descriptions above.

### Accommodate a Wider Variety of Input

Some of the analysis assumes a melody and accompaniment style in the input. This works well in other settings as long as there is always 1 principle voice (and the user has properly kept this voice processed as voice 1). Still, in future versions of TV-PR, a more general framework would be better.

**Multiple Themes**

Variation Sets based on two or more themes, though not as common as single-theme variations, are written from time to time. The most common format for these sets is interleaving variations (Theme 1, Theme 2, Theme 1, etc.). Since this format reduces to generating two sets of variations and alternating the parts, the existing planner would work well as is. In reality, though, the music is not as rigid as this solution would be. Some interplay between the musical resources of the themes would need to be facilitated in order to produce realistic variations. Consider the common practice of framing Theme 1 in the style of Theme 2, for example.

**Multiple Instruments**

Early in this work, the planning problem was greatly simplified by restricting the input (and eventually, the output) to piano music. As described above, the system to discriminate harshness is already capable of handling multiple instruments (provided that realistic sonic profiles can be developed). The MIDI files also carry instrument mappings readily. The likely problems stem from the obligatory expansion of the planner to provide instrument-oriented goals. The orchestral planner might call for a soft version of the melody with high sustain, or for certain parts to either blend in or stand out. This effectively adds another dimension to the planning problem, and increases the complexity of the constraints and preferences considerably. (My relatively small experience in orchestration would probably also be a burden here).

### Automatic Separation of Voices

Of course, the planner would be more useable and accessible if MIDI files could be input unmodified. The two major advantages would be eliminating the tedium of separating voices, and servicing users without MIDI editing capability.

### Separation of Phrases

Perhaps the advance worth the most attention now for TV-PR is the ability to reliably separate musical phrases. The problems with the current algorithm are described in the section about motif selection. With a more robust phrase separation algorithm, not only would the planner be more expressive, but motif selection would benefit as well.

The phrasing problem is interesting in that several possible steps toward a solution come to mind, but none of these steps are currently feasible to implement. LBDM, the phrase-separator on which my attempts were based ([Cambouropoulos 1996]) acknowledges that the system needs to somehow incorporate information about parallelism and repetition. Without such capability, relatively simple melodies (such as *Frère Jacques*) can produce bad results. I should note here that LBDM performs quite well on melodies not relying on repetition or other abstract grouping to define their structure.

### Interaction between the Abstract and Concrete Planner

Though it would seriously impact the running time, a more flexible (and intelligent) arrangement would be for the planner to decide on the proper number of variations to generate. There are two possible directions to do:

1) Every time the concrete planner runs out of resources, the abstract planner must take some action. Variations may be shaved off the middle (to preserve symmetry), or perhaps the planner would have to start over and produce fewer sections. Every time the concrete planner has too many resources left over, the abstract planner would have to add new variations in the middle (to preserve symmetry) or start over. One problem with this approach is that the location of the climaxes changes relative to the length of the entire piece. Steps would have to be taken to ensure the climaxes don't move too far from their mark.

2) A far better solution (and far more difficult to implement) would be to rewrite the knowledge sources so that the concrete and abstract planners run in parallel. In this system, the concrete planner would approximate the distribution of the musical resources. If all indications yell failure, then the number of variations is adjusted and the process starts over. Assuming the predictions could be made accurate, this solution would eliminate a lot of wasted effort by the abstract planner.

**Explanation System**

Given that a good realizer for other variation planner is likely years off, one important enhancement would be a system able to explain the reasons for its choices. This would help the user who feels that one of TV-PR's choices are suspect. Unfortunately, the planner as written is not very capable of providing this information.

Explanation is easy to implement in a system that uses production rules (such as an expert system), but TV-PR is largely random to promote variety.

Since the planner simple throws random values at the preferences until it is satisfied, the only explanation has to be "I chose it because it was the first choice not seriously violating my constraints." Preferable would be, "I chose Am for the third variation because it is related to the main key (being the parallel minor of A), and because a traditional choice for a variation in two voice counterpoint is a minor key."

Still, effort to retro-fit TV-PR with an explanation system would be well-worth it from the human user's perspective. Novices may learn from it while composing, and masters may want the plan justified.

**Standard Resource Pool**

Though we'd be on shaky ground with the goal of style independence, the fact remains that composers use information from outside the theme to construct variations. Every style has certain clichés that help to define it. The Alberti bass line, ubiquitous in the Classical era, makes a perfect example. A program like EMI would no doubt pick up on these with a large enough database, but TV-PR currently only uses information found within the theme. A simple set of rhythms and intervals (no doubt connected to the harmonic style to put a time period on it) would expand TV-PR's horizons considerably.

## References


Berry, Wallace. 1987. Structural Functions in Music. Dover Publications, Inc.: New York.

Cambouropoulos, Emilios. 1996. "A formal theory for the discovery of local boundaries in a melodic surface." Proceedings of the Troisièmes Journèes d;Informatique Musicale. Universitè de Caen: Caen.

Cope, David. 1987. "An Expert System for Computer-Assisted Composition." Computer Music Journal 11(4): 30-46.

Cope, David. 1991. Computers and Musical Style. A-R Editions, Inc.: Madison, Wisconsin.

Cope, David. 1992. "Computer Modeling of Musical Intelligence in EMI." Computer Music Journal 16(2): 69-83.

Hovy, Eduard H. 1988. Generating Natural Language under Pragmatic Constraints. Lawrence Erlbaum Associates: New Jersey.

Kamenetsky, S., Hill, D., and Trehub, S. 1997. "Effect of Tempo and Dynamics On the Perception of Emotion in Music." Psychology of Music 25: 149-160.

Lerdahl, Fred, and Jackendoff, Ray. 1983. A Generative Theory of Tonal Music. MIT Press: Cambridge, Massachusetts.

Meteer, Marie. 1992. Expressibility and the Problem of Efficient Text Planning. Pinter Publishers: London.

Narmour, Eugene. 1992. The Analysis and Cognition of Melodic Complexity. University of Chicago Press: Chicago, Illinois.

Quinn, Ian. 1997. "Fuzzy Extensions to the Theory of Contour." Music Theory Spectrum 19(2): 248-263.

Schillinger, Joseph. 1978. The Schillinger System of Musical Composition. Da Capo Press: New York.

Shank, Roger, Kass, Alex, and Riesbeck, Christopher. 1994. Inside Case-Based Explanation. Lawrence Erlbaum Associates: New Jersey.

Smith, Allan. 1997. "Cumulative Method of Quantifying Tonal Consonance." Music Perception 15(2): 183.

Smoliar, Stephen. 1994. "Computers Compose Music, But Do We Listen?" Music Theory Online 0(6):(Web Page) <http://boethius.music.ucsb.edu/mto/issues/mto.94.0.6/mto.94.0.6.smoliar.art>

Vidyamurthy, G. and Chakrapani, Jaishankar. 1992. "Cognition of Tonal Centers: A Fuzzy Approach." Computer Music Journal 16(2): 45-50.

Widmer, Gerhard. 1992. "Qualitative Perception Modeling and Intelligent Musical Learning." Computer Music Journal 16(2): 51-66.

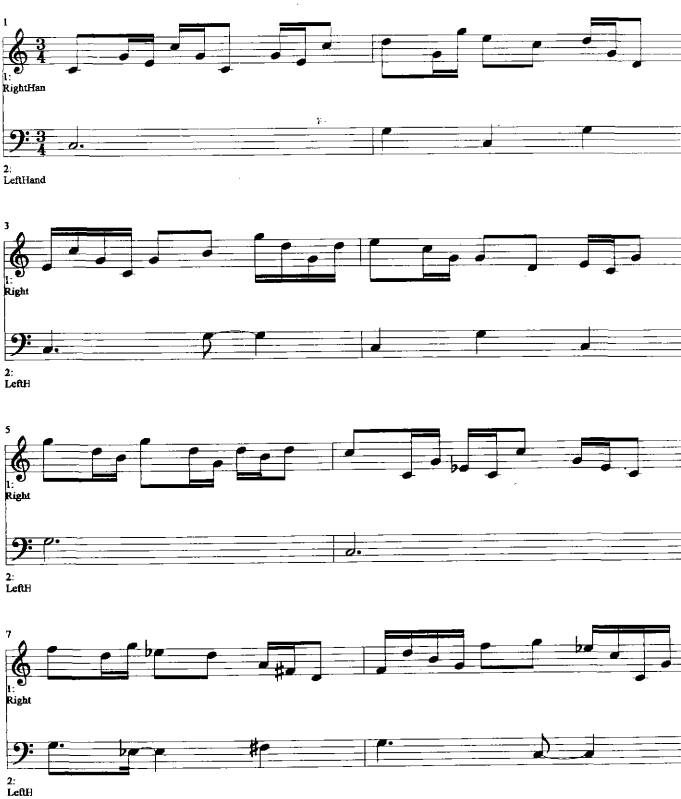
Williams, Peter. 1997. The Chromatic Fourth During Four Centuries of Music. Clarendon Press. Oxford.

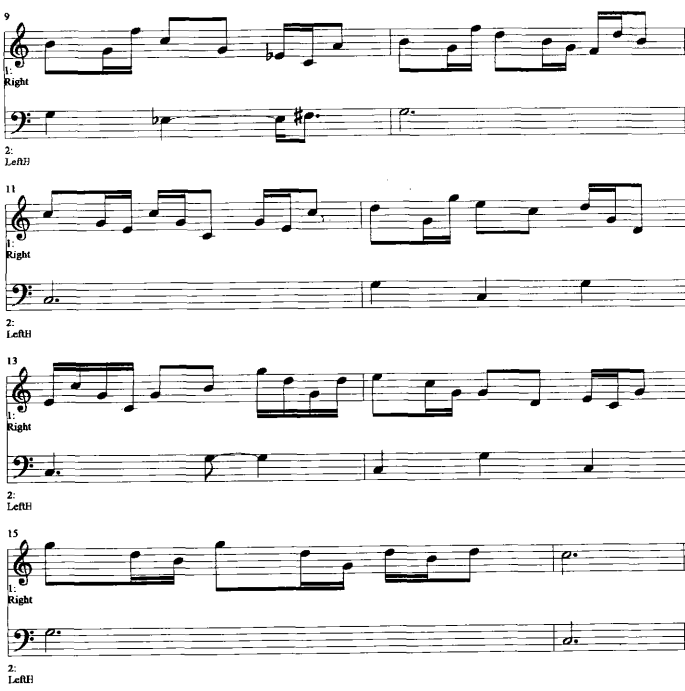
# Appendix A: Sample Composition Composed by Algorithm using Schillinger Resultants

# Resultant of Overlapping Generators

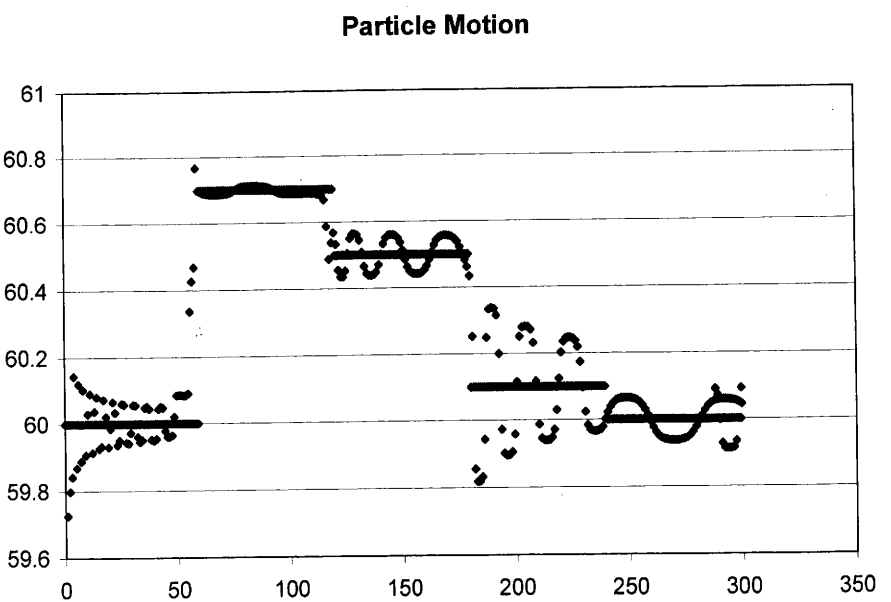
(Following Schillinger's Methodology)

1998




9
1:
Right
2:
Left
11
1:
Right
2:
Left
13
1:
Right
2:
Left
15
1:
Right
2:
Left

## Appendix B: Sample from 'Particle Variations'


Particle Motion
61
60.8
60.6
60.4
60.2
60
59.8
59.6
0
50
100
150
200
250
300
350

# Appendix C: Sample of Constraint Rule Base

```
// ****************************************
// RULE 1
// softer moods don't mix with other moods well
// ****************************************
( (
    CALM_MOOD[v(30,500)] + SOFT_MOOD[v(30,500)] + RELAXED_MOOD[v(30,500)]
) *
(
    MOODY_MOOD[v(30,500)] + SURPRISING_MOOD[v(30,500)] +
    DISORIENTED_MOOD[v(30,500)] + EXCITING_MOOD[v(30,500)]
) ) ; 50.

// ****************************************
// RULE 2
// don't have a calmer mood during a climax...
// ****************************************

( CLIMAX[] *
(
    CALM_MOOD[v(60,500)] + SOFT_MOOD[v(60,500)] + RELAXED_MOOD[v(60,500)]
) ) ; 80.          // high penalty for softer moods....

( CLIMAX[] *
(
    CALM_MOOD[v(10,80)] + SOFT_MOOD[v(10,80)] + RELAXED_MOOD[v(10,80)]
) ) ; 55.     // lesser penalty for lesser soft moods...

// ****************************************
// RULE SET 3
// can't be incompatible types at the same time
// ****************************************
(    MARCH_TYPE[] * WALTZ_TYPE[] ); 100.

( COUNTERPOINT_TYPE[] * (MELODYACCOMP_TYPE[] + BLOCKCHORD_TYPE[] ) ) ; 100.

(    WALTZ_TYPE[] * BLOCKCHORD_TYPE[] ); 100.

      // MORE???

// ****************************************
// RULE SET 4
// penalties with wierd combinations with types
// ****************************************

(    WALTZ_TYPE[] * TEMPO_FEEL[v(-20,80)] ) ; 40. // too slow

(     WALTZ_TYPE[] * TEMPO_FEEL[v(81,100)] ); 10. // on the slow side

(    WALTZ_TYPE[] * TEMPO_FEEL[v(220,250)] ); 10. // on the fast side

(    WALTZ_TYPE[] * TEMPO_FEEL[v(220,500)] ); 40. // very fast

(    MARCH_TYPE[] * TEMPO_FEEL[v(220,500)] ); 40. // too fast for a march

(    MARCH_TYPE[] * RHYTHM_FEEL_H[v(75,500)] ); 50. // too jerky for a march

(   VOLUME_FEEL[v(-20,20)]  ); 50. // too soft
(   VOLUME_FEEL[v(0,55)] ); 23. // pretty soft..
( TEMPO_FEEL[v(-20,40)] ); 50. // too slow...
( CALM_MOOD[v(-20,60)] ); 50. // not enough
( SOFT_MOOD[v(-20,60)] ); 50. // not enough
( RELAXED_MOOD[v(-20,60)] ); 50. // not enough
( SURPRISING_MOOD[v(-20,60)] ); 50. // not enough
( DISORIENTED_MOOD[v(-20,60)] ); 50. // not enough
( MOODY_MOOD[v(-20,60)] ); 50.
( EXCITING_MOOD[v(-20,60)] ); 50.

( DEVIATION_FROM_ORIG[v(-20,25)] ); 25. // not enough

( CLASSICAL_STYLE[] * TWENTYCENT_STYLE[] ); 400. // NOT to Happen
```

## Appendix D: Sample output plan from TV-PR

# Variations

Variations on G:\DOCUMENTS\SCHIL3.MID

1. Variation 1 (the given theme)
2. Variation 2
3. Variation 3
4. Variation 4
5. Variation 5
6. Variation 6
7. Variation 7
8. Variation 8
9. Variation 9
10. Variation 10
11. Variation 11

# Variation 2

- The variation uses G as the key center.
- The variation strategy is, to a moderate degree, time expansion.
- The variation has, to a high degree, a romantic feel.
- The variation has a tempo around 103 bpm.
- There is, to a moderate degree, an exciting feeling.
- There is, to a high degree, a disorienting feeling.
- The variation exhibits continuity to a very high degree.
- The variation has a maximum texture of 4 voices.
- The variation has a mimimum texture of 3 voices.
- This variation has aspects of melody and accompaniment writing, to a high degree.
- The variation feels conventional (with regard to the current style) to a high degree.
- The variation predominantly uses the minor mode.
- The variation exhibits rhythmic instability in the melody to a low degree.
- The variation exhibits rhythmic instability in the harmony to a low degree.
- The variation exhibits intervallic instability in the harmony to a low degree.
- The variation exhibits intervallic instability in the melody to a moderate degree.
- It deviates from the theme to a moderate degree.
- The variation exhibits rhythmic syncopation to a low degree.
- The variation exhibits harshness to a very high degree.

- I recommend emphasizing the following motif, which is rhythmic.
- I recommend emphasizing the following motif, which is intervallic.