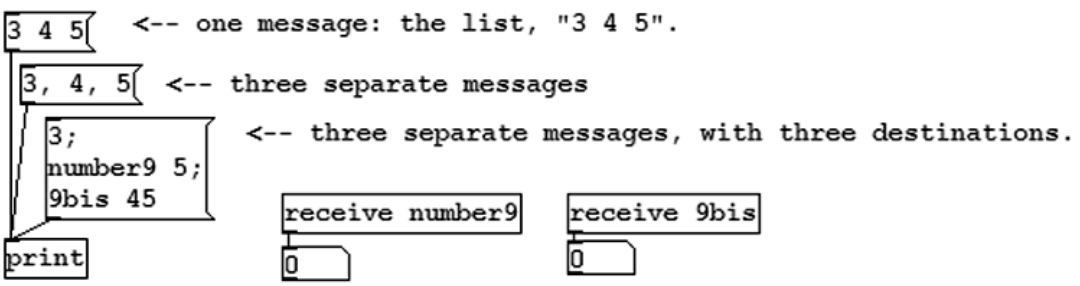
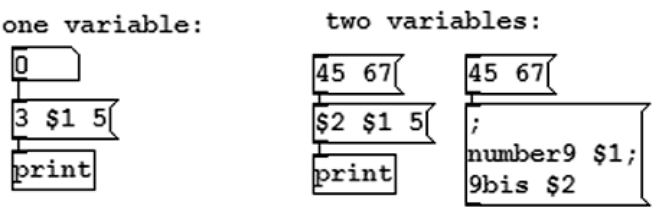
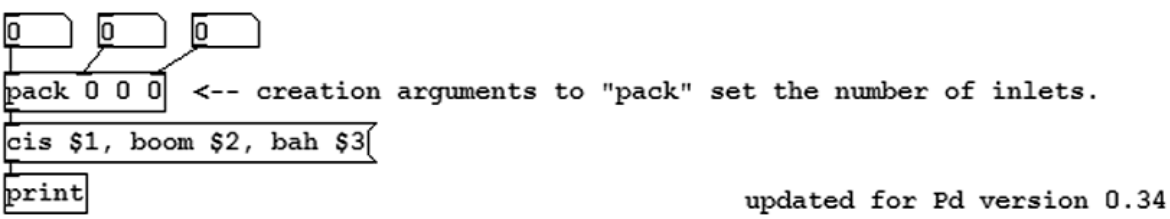
In addition to using semicolons to separate messages, you can use commas, which continue a stream of messages to the same destination. Thus:



You can use "$1", etc., as variables in messages. Send the message box a list whose elements supply the values. A number is just a list with one element.



But to really exploit the possibilities using multiple variables, you will need the "pack" object to get two or more values into the same message:



updated for Pd version 0.34

Music as a Formal Language

Bryan Jurish

Abstract

The main focus of this paper is the characterization of generic musical structure in terms of the apparatus of formal language theory. It is argued that musical competence falls into the same class as natural language with respect to strong generative capacity – the class of *mildly context-sensitive languages* described by Joshi (1985).

1 Introduction

The main focus of this paper is the characterization of generic musical structure in terms of the apparatus of formal language theory. Section 2 briefly describes some relevant aspects of formal language theory. Section 3 introduces that class of mildly context-sensitive languages assumed to contain the natural (spoken) languages, and presents an argument that musical competence also falls into this class.

2 A Brief Tour of Formal Language Theory

Alphabets and Strings

A formal language is simply a set L of (finite) strings[1] over some finite character alphabet Σ: $L \subseteq \Sigma^*$. Candidates for alphabetic characters in the domain of music include rhythmic, tonal, and timbral quanta. Such an inventory of quanta might be extended by additional "empty elements" to encode continuous aspects of musical structure within the context of a finite alphabet, much as the symbolic linguist's toolbox has been extended to include *traces, empty operators*, and other theoretical objects which cannot be directly observed in surface strings.

[1] Traditionally, formal languages are defined in terms of the *free monoid* $\langle \Sigma^*, \circ, \varepsilon \rangle$, where $\circ$ is the concatenation operator and ε represents the empty string.

Grammars and Automata

Any "interesting" language being infinite, it has proved useful to characterize formal languages by means of some well-defined finite specification. Traditionally, both grammars and automata have been used for this purpose. The most general notion of a grammar is usually defined as a 4-tuple $G = \langle V, \Sigma, P, S \rangle$ where:

- V is a finite alphabet, partitioned into disjoint subsets N (nonterminal alphabet) and Σ (terminal alphabet),
- $P \subset V^* \times V^*$ is a finite set of *productions* or *rewrite* rules, usually written as $x \rightarrow y \in P$ for $(x, y) \in P$.
- $S \in N$ is the distinguished start symbol.

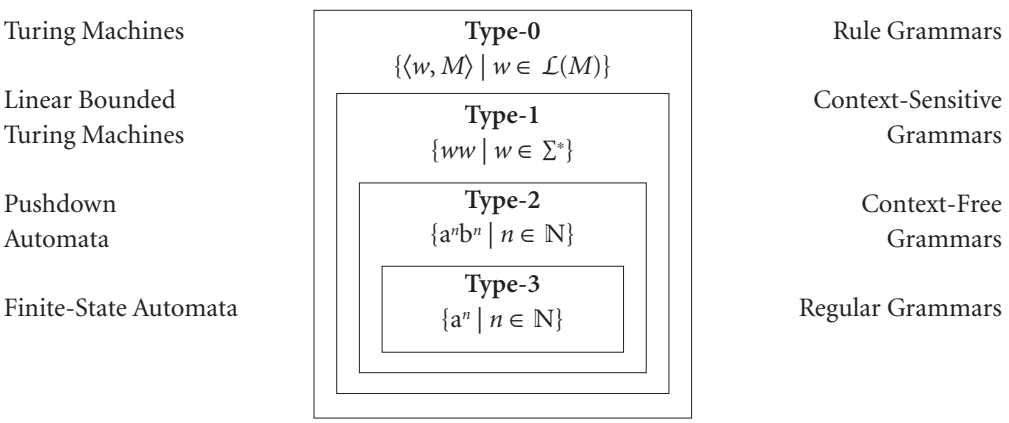


Figure 1: The Chomsky hierarchy of weak generative capacity

A grammar G may be used to specify the formal language $L = \mathcal{L}(G)$ generated by G by means of the *direct derivability* relation $\Rightarrow_G$ for G: $uxv \Rightarrow_G uyv$
iff $u, v, x, y \in V^*$ and $x \rightarrow y \in P$; defining[2] $\mathcal{L}(G) = \{w \in \Sigma^* \mid S \Rightarrow_G^* w\}$.

Varying restrictions on the format of a grammar's rules give rise to different species of grammar, and has strong implications concerning the computational complexity associated with common language-related tasks such as parsing or generation.

[2] Here, $\Rightarrow_G^*$ is the reflexive and transitive closure of the direct derivability relation $\Rightarrow_G$ for G.

Generative Capacity

The weak generative capacity of a grammar formalism is just the set of string languages which can be described in terms of generation by that formalism. A grammar formalism's *strong generative capacity* on the other hand can be identified with the set of *complete derivations* in that formalism, thus capturing the structural properties encoded by a grammar in addition to its string output. The inclusion hierarchy of traditional grammar formalisms (Chomsky, 1957; Hopcroft and Ullman, 1969) is graphically depicted in Figure 1.

- **Type-3 (Regular Languages)**
  A notational variant of *regular expressions* such as those used by many UNIX utilities, the type-3 languages (regular grammars, finite-state automata) are computationally unproblematic: they are deterministically parseable (in linear time), and support a number of useful algebraic operations including union, concatenation, closure, and even complement and intersection. The major drawback of type-3 languages is their limited generative capacity[3].
- **Type-2 (Context-Free Languages)**
  Type-2 languages exhibit structures which can be described by trees, such as balanced parentheses or "mirror" structures. Most programming languages belong to the deterministically parseable subset of the type-2 languages. The type-2 languages in general are parseable in $O(n^3)$ time, but carry with them a number of sticky problems related to *ambiguity*, and are not closed under complement or intersection.

  Certain aspects of musical structure place musical languages at least in this category – an example is the "mirrored" melodic run in Figure 2(a).

Figure 2: Context-free (a) and non-context-free (b) tone dependencies

- **Type-1 (Context-Sensitive Languages)**
  The context-sensitive languages include the *copy language* (see Figure 1) among others. They are decidable, but are computationally quite complex in the general case. In some cases however, the adequate analysis of musical languages appears to require more structure than a type-2 grammar can provide, as suggested by Figure 2(b).

[3] As anyone who has tried to use `sed`(1) to automate tricky corrections in *C* code knows, regular expressions cannot perform balanced parentheses matching to arbitrary depths.

- **Type-0 (Recursively Enumerable Languages)**
  Type-0 languages are those recognized by some Turing machine (equivalently, those produced by some unrestricted rule grammar), and are usually taken to be identical to the (image of the) set of computable functions. Type-0 formalisms are very powerful, but famously intractable (Turing, 1936).

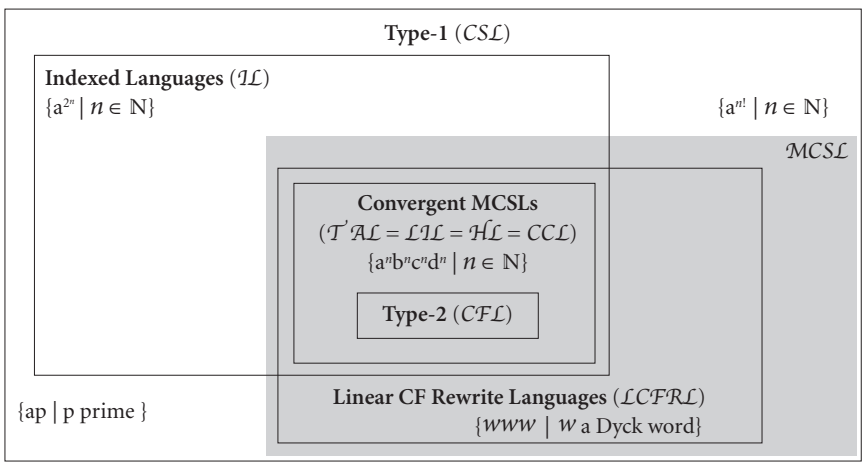


Figure 3: Generative capacity of some (mildly) context sensitive formalisms

3 Mild Context-Sensitivity

Natural languages are known to exhibit some phenomena the description of which goes beyond the weak generative capacity of context-free grammars (Huybregts, 1984; Shieber, 1985). Most prominent among these are *counting dependencies* (of order $m \in \mathbb{N}$) of the form $\{x_1^n x_2^n \ldots x_m^n \mid n \in \mathbb{N}\}$: contextfree (CF) grammars can encode such dependencies only for $m \leq 2$. This is easy to see informally when context-free analysis trees are considered: tree structures cannot have "tangling branches", and thus cannot encode such dependencies.

Joshi (1985) characterized the class of languages to which natural languages are expected to belong as that of the *mildly context-sensitive* languages (MCSL), the inclusion relations between some well-known examples of which are graphically depicted in Figure 3. This class of formal languages is informally identified by the properties of *constant growth*, *polynomial parsing complexity*, and *bounded cross-serial dependencies*, which are discussed individually in the following sections.

Constant Growth

Every known natural (spoken) language has a length-ordering on well-formed sentences for which there is a finite upper bound on the number of words which separate the sentences with respect to that ordering. The condition on constant growth excludes languages such as $\{\mathbf{a}^{2^n}\}$ from the class of MCSLs, thus eliminating Aho's (1968) Indexed Grammars as a candidate formalism.

Although I lack extensive knowledge of the empirical data in the case of musical languages, I think it safe to surmise that these also display the constant growth property. Taking melody again as an example, this means that for a musical language L there must exist a number $j \in \mathbb{N}$ such that for all melodies[4] $M_1, M_2 \in L$ with $|M_1| > |M_2|$ where there is no $M_3 \in L$ with $|M_1| > |M_3| > |M_2|$, then $|M_1| - |M_2| \leq j$. Informally, this amounts to the claim that whenever a melody can be extended (and possibly altered) to produce a longer melody, the maximum length of the shortest such extended melody is fixed by the musical system in question.

Polynomial Parsing Time

Essentially a constraint on computational tractability, MCSL membership for a given input string must be decidable in polynomial time with respect to the length of that string. Rephrased in terms of generation systems, an output string of length n should be derivable in $O(n^k)$ for some $k \in \mathbb{N}$ specific to the grammar (weak form) or MCSL subfamily (strong form) in question.

For practical purposes, it is certainly desirable to require some computational complexity constraint on any formalism used to describe musical structure. Concrete modeling proposals known to me (Xenakis, 1971; Lerdahl and Jackendoff, 1983; Steedman, 1984, 1996; Baker, 1989; Leman, 1989; Bel and Kippen, 1992; Bod, 2001, 2002) do in fact all fulfill this criterion.

Bounded Cross-Serial Dependencies

This condition is clearly motivated by the counting dependency evidence placing natural language beyond the domain of the context-free languages. It requires a limit $m \in \mathbb{N}$ on the number of cross-serial structural dependencies encodable by a given grammar (original, weak form), or grammar formalism (strong form).

It is not at all immediately clear whether such a restriction is desirable for musical languages, especially if phenomena such as theme and variation are to be encoded as real cross-serial

[4] Here, $|M|$ represents the length of M: $|x_1 \ldots x_n| = n$ for $x_1, \ldots, x_n \in \Sigma$.

dependencies rather than pure chance similarities. At the very least, we should reject the strong form of this criterion for musical languages. Such a weak (original) form of Joshi's restriction is compatible with the formal properties of contemporary theories for the description of natural (spoken) languages (Stabler, 1999; Michaelis, 2001).

Common equivalent mildly context-sensitive grammar formalisms include the *Tree Adjoining Languages* ($\mathcal{TAL}$) (Joshi, 1985, 1987), the *Linear Indexed Languages* ($\mathcal{LIL}$) (Gazdar, 1988), the *Head Languages* ($\mathcal{HL}$) (Pollard, 1984), and a restricted form of the *Combinatory Categorial Languages* ($\mathcal{CCL}$) (Steedman, 1990). This family of languages is distinguished by a hard limit of $m \leq 4$ counting dependencies.

Stronger formalisms which provide room for more counting dependencies on a per-grammar basis include the *Divided Index Languages* ($\mathcal{DIL}$) (Staudacher, 1993) and the *Linear Context-Free Rewrite Languages* ($\mathcal{LCFRL}$) (Seki et al., 1991).[5] Michaelis (1999, 2001) showed that the latter are equivalent to Stabler's (Stabler, 1997) formalization of the *Minimalist Grammars* proposed by Chomsky (1995) as a vehicle for the description of natural language.

4 Conclusions and Perspectives

Adopting the working hypothesis that musical structure can be expressed symbolically in terms of a finite alphabet of structural quanta, it becomes possible to model a musical system as a formal language, or set of strings. The generative capacity required for the characterization of musical languages must lie beyond that of the context-free languages, in order to adequately encode the cross-serial dependencies displayed by musical phenomena such as theme and variation. Empirical arguments were presented that musical languages exhibit the characteristic properties of the mildly context-sensitive languages, to which natural (spoken) languages are also assumed to belong.

One property considered important for musical languages which was not discussed above is the possibility of their incremental generation and analysis, also known as the "improvisation property". While empirical evidence exists that spoken language also displays this property, it is often unclear how it may be efficiently implemented for a strong language family such as the MCSLs.

Real-time musical processing environments such as Pd (Puckette, 1996, 2004) oriented toward generation and synthesis are usually Turing-complete (type0), thus opting for strong generative capacity and leaving issues of computational complexity within the user's discretion.

[5] The context-sensitive "pattern grammars" used by Bel and Kippen (1992) to model musical structure appear at first glance to fall into the same category, but a proof of this hypothesis is beyond the scope of the current work.

Analytically motivated approaches (Xenakis, 1971; Steedman, 1984; Bod, 2002) often tend toward the opposite extreme of low computational complexity at the cost of generative capacity.

One way to try and unify these disparate approaches is to use a powerful (but complex) processing tool such as Pd as a vehicle for the instantiation, manipulation, and interpretation of instances of a weaker (but generally efficient) generative mechanism. The most obvious candidate for such a weak embedded mechanism which also exhibits the improvisation property and which is familiar to many users is that of *regular expressions*, a notational variant of the type-3 languages. An abstract C library for regular expressions and weighted finite state machines with a Pd interface is currently under development (Jurish, 2004).

References

A. V. Aho. Indexed grammars. *Journal of the Association for Computing Machinery*, 15:647-671, 1968.

M. Baker. A computational approach to modeling musical grouping structure. *Contemporary Music Review*, 4:311-325, 1989.

B. Bel and J. Kippen. Bol processor grammars. In O. Laske, M. Balaban, and K. Ebcioglu, editors, *Understanding Music with AI – Perspectives on Music Cognition*, pages 366-401. MIT Press, Cambridge, MA, 1992.

R. Bod. A memory-based model for music analysis: Challenging the Gestalt principles. *Journal of New Music Research*, 31(1):26-36, 2001.

R. Bod. A unified model of structural organization in language and music. *Journal of Artificial Intelligence Research*, 17:289-308, 2002.

N. Chomsky. *Syntactic Structures*. Mouton and Co., The Hague, 1957.

N. Chomsky. *The Minimalist Program*. MIT Press, Cambridge, MA, 1995.

G. Gazdar. Applicability of indexed grammars to natural languages. In U. Reyle and C. Rohrer, editors, *Natural Language Parsing and Linguistic Theories*, pages 69-94. D. Reidel, Dordrecht, 1988.

J. E. Hopcroft and J. D. Ullman. *Formal Languages and Their Relation to Automata*. Addison-Wesley, Reading, MA, 1969.

R. Huybregts. The weak inadequacy of context-free phrase structure grammars. In G. J. de Haan, M. Trommelen, and W. Zonneveld, editors, *Van Periferie naar Kern*, pages 81-99. Foris, Dordrecht, 1984.

A. K. Joshi. Tree adjoining grammars: How much context-sensitivity is necessary for characterizing structural descriptions. In D. Dowty, L. Kartunnen, and A. Zwicky, editors, *Natural Language Processing: Theoretical, Computational, and Psychological Perspectives*, pages 206-250. Cambridge University Press, New York, NY, 1985.

A. K. Joshi. An introduction to tree adjoining grammars. In A. Manaster-Ramer, editor, *Mathematics of Language*, pages 87-114. John Benjamins, Amsterdam, 1987.

B. Jurish. gfsm finite-state machine library. work in progress, 2004.
URL http://www.ling.uni-potsdam.de/~moocow/projects/gfsm .

M. Leman. Adaptive dynamics of musical listening. *Contemporary Music Review*, 4:347-362, 1989.

F. Lerdahl and R. Jackendoff. *A Generative Theory of Tonal Music*. MIT Press, Cambridge, MA, 1983.

J. Michaelis. *On Formal Properties of minimalist Grammars*, volume 13 of *Linguistics in Potsdam*. Univeristät Potsdam, Potsdam, Germany, 2001.

J. Michaelis. Derivational minimalism is mildly context-sensitive. *In Linguistics in Potsdam* (*LiP*), volume 5, pages 179-198. Universität Potsdam, Potsdam, Germany, 1999.

C. POLLARD. *Generalized Phrase Structure Grammars, Head Grammars, and Natural Language*. PhD thesis, Stanford University, Stanford, CA, 1984.

M. PUCKETTE. *Pd: real-time music and multimedia environment (version 0.37-4)*, 2004. <http://crca.ucsd.edu/~msp/software.html>

M. PUCKETTE. Pure Data: another integrated computer music environment. In *Proceedings of the Second Intercollege Computer Music Concerts*, pages 37-41, Tachikawa, Japan, 1996.

H. SEKI, T. MATSUMURA, M. FUJII, AND T. KASAMI. On multiple context-free grammars. *Theoretical Computer Science*, 88:191-229, 1991.

S. M. SHIEBER. Evidence against the context-freeness of natural language. *Linguistics and Philosophy*, 8:333-343, 1985.

E. STABLER. Derivational minimalism. In C. Retoré, editor, *Logical Aspects of Computational Linguistics (LACL '96)*, volume 1328 of *Lecture Notes in Artificial Intelligence*, pages 68-95. Springer, Berlin, Heidelberg, 1997.

E. STABLER. Remnant movement and complexity. In G. Bouma, G.-J. M. Kruijff, E. Hinrichs, and R. T. Oehrle, editors, *Constraints and Resources in Natural Language Syntax and Semantics*, pages 299-326. CSLI Publications, Stanford, CA, 1999.

P. STAUDACHER. New frontiers beyond context-freeness: DI grammars and DI automata. *EACL Proceedings*, pages 358-367, 1993.

M. STEEDMAN. A generative grammar for jazz chord sequences. *Music Perception*, 2:52-77, 1984.

M. STEEDMAN. Gapping as constituent coordination. *Linguistics and Philosophy*, 13:207-263, 1990.

M. STEEDMAN. The blues and the abstract truth: Music and mental models. In *Mental Models in Cognitive Science*, pages 305-318. Erlbaum, Mahwah, NJ, 1996.

A. M. TURING. On computable numbers with an application to the Entscheidungsproblem. *Proceedings of the London Mathematical Society*, 2(42): 230-265, 1936.

I. XENAKIS. *Formalized Music*. Indiana University Press, Bloomington, 1971.