University of Glasgow
Department of Music

Funded by the
Joint Information
Systems Committee

Call: Communications and Information Technology Standards

MuTaTeD! - Music Tagging Type Definition

A project to provide a meta-standard for music mark-up by integrating two existing music standards

(Report Jan 2000)

Dr. Stephen Arnold
Carola Boehm
Dr. Cordelia Hall

University of Glasgow
Department of Music

CONTENTS:

I. Introduction

(by Dr. Stephen Arnold)

"Music" is a term which has to stand for a variety of things. One reads music; one plays music; one listens to music; one studies music; one even "makes" music.

At the end of everything, Music has to have a sonic existence: that is the whole point of it. But to support the sonic manifestation, we have developed ancillary representations which have a number of functions.

The one with a long history is notation. In its earliest forms, when cultural transmission was essentially aural, it captured the barest minimum of information. In no way was it meant to specify every detail of performance, since that kind of information was conveyed via the aural-traditional route which depended on experts so familiar with their musical culture that there was no necessity for the notated to forms even to attempt to do so.

For much of our recorded music history, notated forms of music have not been created as authoritative statements aimed at the archives of posterity. They have been practical documents created to support performance, and they tend not to include details of *how* to perform a work (e.g. details of dynamics, articulation, even instrumentation and tempi variation) but a rather more conceptual representation of the composition in abstract. This is especially understandable when one bears in mind that composers were usually also performers who frequently led performances of their own music, and that there was therefore a shared, unwritten, aural traditional framework which prescribed conventions of performance.

As Music became less localized, more massive in its forces and time-scales, and more individualized and differentiated in style and content, the need for more more highly specified scores and parts became ever more acute, until almost every conceivable nuance of performance was explicitly characterized by graphical or textual signing within the notated version. Think of Mahler's Symphonies. This tendency has continued, although, to this day, there are observable, inevitable (and desirable) deviations between notated and performed manfestations of the work.

With the move to highly specified musical notation has come a scholarly approach to music which accords a privileged place to the notated representations of the work. Within our musical culture, what is meant by the term "music" is, as often as not, the score rather than the performance. Scholars study notated works for many kinds of evidential data relating to the work's history or content, while performers confidently develop their performnces guided only by the notated form of the work.

With the burgeoning of the Information Age, new dimensions for Music are opening up. At the conceptual level, we now conceive of an information space of many inter-related and interleaved dimensions, capable of fulfilling a wide variety of musical functions and needs, not only within purely musical contexts, but also as part of broader multimedia frameworks. The gap between concept and realization is, however, still frustratingly wide. Standards for text, image, networks, metadata, etc., are fields of intense research and commercially oriented activity, raising the prospect of exploting critical masses of content-rich, time-based media over wide-area networks within the framework of , for example, MPEG4, MPEG7, SDIF and SMIL. But the situation regarding standards for musical data is confused, the

confusion arising from the complexity and endless variety of musical data, much of which is "culture specific" rather than genuinely generalizable.

Even though there are as yet no universal, platform-independent, generic, established standards for music representation, music information or tagging, significant steps are now being taken to provide structures for music data and tools for display, searching and manipulation which will undoubtedly prove to be of great value, not only in the rarified realms of academe, but also throughout the "real" commercial worlds of the music and media industries.

The JISC-funded *MuTaTeD!* project is one of several within the fields of music technology, music information and computer music, which are running in the Department of Music at the University of Glasgow. Ours is a step-by-step approach. *MuTaTeD!* attempts to validate through implementation the concept of integrating two existing music-representation standards, namely SMDL (Standard Music Descrpition Language) and NIFF (Notation Interchange File Format). This integration was forseen and advocated in the NIFF specification of 1995.

An application based on this integration should support (i) via SMDL, the representation of Music as a time-structured entity, and (ii) via NIFF, the high-quality display of the notated version. An important outcome of *MuTaTeD!* is the first and only (or so we believe!) SMDL-to-NIFF converter, which allows NIFF-compliant notation editors such as LIME, CPN, IGOR and SmartScore to display SMDL-encoded music data.

The Report which follows is divided into two principal sections: (i*)* a discussion of the challenges of music representation and retrieval; (ii) a detailed introduction to and description of the parsers and compliers used to implement the *MuTaTeD!* prototype. A final section (iii) considers the outcomes of this project and looks forward to future developments in this research area. In particular, we outline *MuTaTeDII* which builds on *MuTaTeD!* by designing and implementing a music-information retrieval system with delivery/access services for encoded music.

We have taken steps to communicate our work to others working in the fields of music encoding, arts and humanities computing and content-based retrieval: some of the more important conferences and meetings are listed below. We have also targetted specialist email ansd news groups, organized and taken part in specialized workshops and been invited to present at others. We shall be co-organizing an important international workshop as part of the International Computer Music Conference in Berlin in August 2000.

We therefore believe that we have attained the goals of the original MuTaTeD! proposal, both in terms of the its technical objectives and of its dissemination objectives.

While we are confident that what has been accomplished within MuTaTeD! is of considerable importance, we recognize that, within the broadest context of music-information representation and retrieval, the steps we have taken, difficult though they have been, are fairly modest in their scope and that considerable further work remains to be done. For this reason, we have embarked on a second phase of MuTaTeD! (MuTaTeD2!), which aims to embed MuTaTeD! outputs in a considerably broader functional context.

II. Achievements and Deliverables

(by the MuTaTeD! Team)

Despite the work being limited to the equivalent of a 4-month FTE project, a significant amount of territory has nevertheless been covered. Principal achievements are:

- proof of concept for the integration of SMDL and NIFF (see Chapter III.8)
- development, revision and integration of a SMDL DTD and a NIFF DTD (see Chapter IV.6 and Appendices A and B)
- the first SMDL to NIFF converter, complementing the Mounce Niff→SMDL converter (Cantate) (ditto)
- expert and user workshops (see Chapter II.1)
- establishment of a small but continuing influence of the MPEG7 standardization process aimed at incorporating compatibility with SMDL. (see Chapter II.1 and Appendix C)
- involvment in the development of the revised SMDL standard, as this will be updated due to changes in HyTime.
- presentations, workshops and invitations to research meetings (Chapter II.1)

1. Conference Papers and Workshops

The results of this works were presented in several large conferences and meetings, national and international:

ARS ELECTRONICA FESTIVAL, Linz, AUSTRIA, September 5 - 8 1999
Meetings of CUIDAD and CIRCUS,
Invited Paper Presentation. Title: "Music Tagging Type Definition".

CIRCUS Meeting, Dublin, Ireland, October 20-22 1999
Invited Paper Presentation. Title: "Metadata and CIRCUS".

DRH99, London, UK, September 12-16 Sept 1999
Paper Presentation: "MuTaTeD!"

Expert and user workshops provided the means of early dissemination of the results:

DRH99, London, UK, September 12-16 Sept 1999
Co-organized Expert Workshop "Music Information Retrieval"

Workshop in Glasgow, 15 December 1999
Organized local workshop "Systems for Music Information Representation and Retrieval"

2. Contact with relevant research groups

Contact with local and external research groups, mailing lists and newsgroups made us aware of other developments in this topic-area and enabled us to become involved in the processes of standardization or creation for SMDL related applications.

Our local research group meets weekly. It is made up of the research assistants, music technology lecturers and academic related staff.

Projects run within the Music Department and relevant to MuTaTeD! are:

Direct software evaluation of Music Notation Packages for an Academic Context
(Project Management: Carola Boehm and Stephen Brandon)
Evaluating music notation packages in the context of a multi-user, multi-computer, networked academic music environment.
Funded by JISC and Universities & Colleges Information System Association (UCISA).
(http://www.pads.ahds.ac.uk/notation_eval.html).

A system for Music-Information Retrieval of Encoded Music (MuTaTeD'II)
(Project Management: Carola Boehm)
The aim of the proposed project is to design and implement a music-information retrieval system with delivery/access services for encoded music. It builds on the output on the MuTaTeD! project. The prototype service will provide a user-friendly, web-based search/browse/query interface to access musical content. Funded by the Library Information Commission and started in 1999.

External Working groups include:

Audio MPEG7 AdHoc group (International scope)
Involvement within the framework of the MPEG7 working group (MPEG ISO/IEC JTC1/SC29 WG11) (http://www.darmstadt.gmd.de/mobile/MPEG7/) to ensure the integration of technologies for information retrieval vital for communities of music libraries and digital music-libraries.

EU-funded working group CIRCUS (European Scope)
Content Integrated Research into Creative User Systems (http://www.circusweb.org), CIRCUS is a working group funded by the EC's Esprit programme. CIRCUS brings together artists, designers, performers, theorists and computer scientists to debate the research issues posed by the next generation of advanced information technology for the creative and performing arts, media and the electronic publishing industry (in its broadest sense). It brings together partner institutions in the UK, Ireland, Finland, France, Germany, Netherlands and Spain. CIRCUS advises the EC on the integration of content and technology in terms of creative pull vs technology push.Glasgow contributes to the subgroup dealing with metadata standards for the creative arts user community.

MusicWeb Connect (European Scope)
Within the european-funded MusicWeb Connect project Glasgow has the reponsibility of technical co-ordination, evaluation and dissemination. It aims to integrate and implement music-specific on-line interactivity and collaborative working tools. Two technologies underpin this project, a) the GUIDO Music Representation Language and Note Server for dynamic music representation and b) "Gentle", the educational application built upon the Hyperwave Information/Knowledge Management System. The collaborating institutions are the Royal Conservatory of Music and Dance Den Haag (Project Co-ordination), Netherlands; Institute for Information Processing, Graz, Austria; Technische Hochschule Darmstadt and Hochschule für Musik Hanover, Germany; and IRCAM, France. Funded by EU Connect Programme and German and Dutch national funding.

3. Additional Dissemination Activities

The success of the international expert workshop organized under the auspices of MuTaTeD! as part of DRH99 has led to a further workshop to be held in the University of Massachusetts to which we have been invited.
We are organizing a further workshop under the auspices of the Internationla Computer Music Conference 2000, where we are also presenting formal papers relating to MuTaTeD!.

SAMURAI (2nd Symposium on Adequate Music Representation Issues), Vancouver, CA
Invited paper. June 16-19 2000

WEDELMUSIC, Firenze, ITALY
Invited as Member of Expert User Group to the first meeting of the EU-funded working group WedelMusic. April 1-2 2000
The objectives of the project are:

- to make possible the distribution of interactive music via the internet managing IPR and integrating audio, symbolic and image formats within a unified model
- to adopt a model for distributing music, taking into account the user needs

Paper (acceptance to be confirmed): "Music Tagging Type Definitions, MuTaTeD I and II : Systems for Music Representation and Re-trieval", Special Session on Sound and Music Technology within the EUROMICRO WORKSHOP on Multimedia and Telecommunication, Maastricht, Netherlands 4-7 September 2000

III. Systems for Music Representation and Retrieval

(by Carola Boehm)

1. Abstract

The project MuTaTeD! validated the concept of integrating two existing music representation standards: SMDL (Standard Music Description Language) and NIFF (Notation Interchange File Format). An application with the combined implementation of these two standards, SMDL and NIFF, supports the representation of music as a time-structured entity in its own "gestalt" as it is conceived and yet provides a standard for high-quality display via the NIFF format. The paper will discuss the general issues surrounding music information representation and music information retrieval, present the result of the UK-JISC funded "proof-of-concept" project: MuTaTeD! (Music Tagging Type Definition) and inform about further development within the UK-Library Information Commission funded MuTaTeD'II project and its MPEG7 involvement.

2. The Broad Context

A serious problem for IT today concerns ***Information Overload***. When, as now, the internet is becoming the platform, the browsers become the operating system, and applications become services[1], services to access music content have to define new methods of storing and distributing time-based data if they are to serve large quantities of high-quality information across wide area networks (WANs). Future information services will need to carry the burden of extensive metadata management, - content searching and manipulation of time-based data - if they are to facilitate intelligent access and efficient delivery to their user communities.

It is now, therefore, quite widely accepted that there needs to be development in three areas (i) ***Information Structure*** (ii) ***Information Representation and*** (iii) ***Information Access.***
In a number of IT sectors, there has been considerable progress in user interfaces, metadata management systems for access, information retrieval, workflow management, and similar areas. But in the area of Music

- the most widely encountered music information structure is midi
- visual information representation over WANs is largely restricted to static images
- music information retrieval is still in its infancy.

While this leaves plenty of scope for researchers, there is a sense that users have still to benefit as fully within the area of Music as they clearly have in other domains.

3. The Music-Specific Context

Accessing and manipulating time-based data over wide-area networks is a research area ready to deliver its first results. For a variety of tasks, such as automatic page-turning and slide-shows, and within a variety of fields, such as multi-media applications and professional video or media companies, there is a requirement for a language that describes time-based media in its structure.

Two main options seem to be possible here:

- handle the media files as they are and synchronize their interaction with each other, or
- use a structured language to represent time-based media files, such as those associated with music, as a time-based medium within its own time-dependent structure.

[1] Kostas Glinos (EU, DG3), "Information Access and Interfaces", presentation for the last call of the 4th framework, 29 September 1997, Brussels.

The first option has yielded solutions that have already been commercially exploited, e.g. interaction of sound and movies, parallel depiction of scrolling text and movies, web slide-sound shows, etc. The second option has so far been neglected, so that the Music community is left without the musical equivalents of, for example, SGML (Standard Generalized Markup Language)[2] and XML (eXtensible Markup Language)[3] and packages to handle them.

"Music Representation Languages" are essentially a means to describe certain musical elements and their time-relation to each other. At the macro level, such a language might represent a sequence of four sound files played one after the other, corresponding to the four movements in a symphony. At the micro level, it might represent single notes on several staffs in their relative "time-positions". Thus, within this model of a "time-based media structure", by heightening the detail of granularity, we conjoin the two options outlined above.

An application with the combined implementation of two standards, SMDL (Standard Music Description Language)[4] and NIFF (Notation Interchange File Format)[5], would support the representation of music as a structured, time-based entity in its own gestalt, - while simultaneously supporting high-quality display via the NIFF format. The publication of the SMDL standard draft in 1995 was recognized in the NIFF specification of the same year when it proposed this very combination for applications.

Among applications which might be founded on this concept are:

- Platform independent file interchange (music software, web)
- Intelligent and dynamic access to structured music (libraries, education)
- Using music-recognition functionality, easy and instant access to a critical mass of music (sheet music sales and loans, music publishers, performers, music and audio distributors)
- Collaborative creation and working processes and the synchronization of time-based media entities of the same and different type (media production industry)

As mentioned above, in order to achieve this, the existence and acceptance of a language that describes time-based media in its time-based structure is required and with it the possibility to represent it in a musical way. The development and adoption of such a language as we envisage will make possible the realization of these objectives.

The basic idea of the MuTaTeD! Project (Music Tagging Type Definition) grew out of these considerations and suggested the following primary project goals:

- the proof of concept of integrating two existing standards, one for music content (SMDL) and one for music score representation (NIFF)
- the use of a standard Meta-DTD for music-tagging languages

[2] SGML, Standard Generalized Markup Language, ISO 8879:1986 of ISO/IEC JTC1/SC34, see http://www2.echo.lu/oii/en/docstand.html#SGML 28/02/00.

[3] XML, eXtensible Markup Language, W3C and IETF, see http://www2.echo.lu/oii/en/docstand.html#XML 28/02/00.

[4] SMDL (Standard Music Description Language) is a HyTime application that conforms to international Standard ISO/IEC 10744. See below.

[5] Notation Interchange File Format, created 1995 in collaboration with several music companies. See below.

4. Music Representation Standards – The Historical Context

Research into Music Representation Standards is not new. There have been many and varied approaches to standardised music representation. The following, historically oriented listing helps show the context for our current efforts.

Independently of (and in parallel with) "conventional" music notation, numerically based systems for encoding musical information have been devised, perhaps governed by an idealistic Pythagorean sense[6] that numerical abstraction and elegance was especially appropriate. But they were also to fulfil a practical need, at a time predating audio recording, to capture musical data as it was being played – a task for which common music notation is not at all well suited[7].

Using typewritten text and early computers, substantial archives of incipits and melodies were developed, giving rise to an ASCII-based notation. These holdings may still be found in many catalogues of music.[8]

With the development of professionally oriented music applications for the computer, new data structures, file formats, and music description standards emerged with almost with every application. The Music Notation software applications, in particular, were forced to create their own underlying music data structures to contain all the information needed to print music. In the absence of established and widely adopted standards, this was inevitable if the applications were to be of any practical use. Surprisingly, perhaps, there was not much impetus to agree a joint standard.

NIFF (Notation Interchange File Format)[9] was the first (and only) commercially-based effort by several companies to agree on an interchange file format for Notation. Although there are now a few NIFF-compliant programs, all the most widely used notation and music programs are still (early 2000) not capable of reading NIFF. Perceptions of their commercial interest appear to lie behind various music companies' failure to follow through with previously announced plans for NIFF implementation. This is difficult for the Music user community to bear, forced as it is to struggle on with MIDI, which cannot and indeed was never intended to support the full extent of the needs of the field.

Recent activity to revive NIFF-related development is largely to the credit of the user community rather than commercial concerns, The MuTaTeD! Project can reasonably be claimed as an important part of this recent push forward.

A positive development can be seen with music tagging languages. The number of text-based music languages, cognate with SGML/XML/SMDL, is rising. A number of groups are pushing for standardisation of a single language to be used over the web. There are developments spinning off from XML under the W3C[10], and developments with SMDL (Standard Music Description Language)[11], backed by MuTaTeD! and their work within MPEG7[12].

---

6 See [Wolf 1963].
7 Ibid. p. 423-437
8 See entries: "DARMS", "Plaine and Easy", "Alma", "Essen", and similar in: {Selfridge Field 1997], "Beyond Midi", Cambridge Massachusetts, CCRMAH 1997.
9 See "Notation Interchange File Format", http://www.musitek.com/niff.html 26/02/00 and Cindy Grande, "NIFF" in [Selfridge-Field 1997] p.491-511.
10 World Wide Web Consortium, http://www.w3.org/
11 ISO/IEC DIS 10743, Standard Music Description Language (SMDL) see http://www2.echo.lu/oii/en/audio.html#SMDL, 26/02/00 and Sloan and Newcomb, "Hytime and SMDL", in: [Selfridge-Field 1997] p.469-489.

As ever, agreement is required so that application developers may safely integrate the standard in their application. Even though SMDL unlike all the other tagging music standards is an ISO[13] standard, it seems so far to have failed the final test of acceptance by application developers.

Below is a summary list of music description standards throughout the last 200 years.

Early Number Notation (19th century and earlier)[14]:
Davantes, Kircher, Avella, Mersenne, Bontempi, Stierlein, Mine, Schulz, Rousseau, Galin-Paris-Cheve, Geisler, Teule, Gatting

19th Century Shorthand[15]:
Sauveur, la Salle, Romanos, World Music Language

Formats for Typewriter and Early Computer Age:
DARMS[16], Plain and Easy[17] and ALMA[18], WMN[19], MECOS[20], ANTOC[21], SPECO[22], NUTOC[23], Essen EsAC[24]

Computer Age: Formats for Notation Software - Binary File Formats
Encore File Format[25], Finale's Enigma[26], Syqyest, Lime's Tilia[27], Nightingale[28], Score's Score[29],

[12] The Moving Picture Experts Group (MPEG) is a working group of ISO/IEC in charge of the development of international standards for compression, decompression, processing, and coded representation of moving pictures, audio and their combination. So far MPEG has produced: MPEG-1 (approved Nov. 1992), MPEG-2 (approved Nov. 1994), MPEG-4 version 1, the standard for multimedia applications (approved Oct. 1998). and is now developing MPEG-7 the content representation standard for multimedia information search, filtering, management and processing (to be approved July 2001). (http://drogo.cselt.stet.it/mpeg/ 11/01/00)

[13] ISO stands for International Organisation for Standardization. The International Organisation for Standardization (ISO) is a worldwide federation of national standards bodies from some 130 countries, one from each country. see http://www.iso.ch/infoe/intro.htm 28/02/00.

[14] See [Wolf 1963].

[15] Ibid. p.419-449.

[16] See Selfridge-Field, Dydo, Hall, Wiering. Towbridge, "Musical Notation Codes (1):DARMS" in: [Selfridge-Field 1997] p.163-216.

[17] See John Howard, Plaine and Easy Code: A code for Music Bibliography, in: [Selfridge-Field 1997] p.362-371.

[18] See [Gould/Longemann 1966].

[19] See [Steinbeck 1982].

[20] Ibid. p.35.

[21] Ibid.

[22] Ibid. p.35 and p394.

[23] Ibid. p.47.

[24] See [Schaffrath 1992] and Schaffrath, "The Essen Associative Code", in: [Selfridge-Field 1997] p.343-359.

[25] See Passport Design, Encore at http://www.gvox.com/ 26/02/00. Passport Design Inc. was bought by G-VOX Entertainment.

[26] Coda Music Software, http://www.codamusic.com/coda/ 26/02/00.

[27] See [Haken/Blostein 1993] and David Cotte, Lippold Haken, "The Lime Tilia Representation", in: [Selfridge-Field 1997] p.283-292.

[28] See Tim Crawford, Donald Byrd, John Bibson, "The Nightingale Notelist", in: Selfridge-Field p.293-314 and Musicware Inc. http://www.musicwareinc.com/prod02.htm 26/02/00.

[29] See Leland Smith, Score in: [Selfridge-Field 1997] p.252-279 and Score (San Andreas Press), http://www.scoremus.com/ 26/02/00.

Computer Age: Formats for Composition/Performance Systems
Midi[30], Mode's SmallMusic[31], Kyma's Kyma[32], Common (Lisp) Music Notation[33], Max Score[34], Csound's Score[35], NeXT's Score format[36], Radio Baton Score File[37], Hush/Hymne/DejaVu[38], MML[39]

Computer Age: Formats for Notation - Text based File Formats
MusicTex[40], Lilyponds Mudela[41], abc[42], GUIDO[43],

Computer Age: Interchange File Formats
NIFF[44]

Computer Age: Analysis and Content Serching
Humdrum's Kern[45], MuseData[46], CPN[47],

WebAge: Formats for Storage, Compression, Delivery
MPEG4[48], SMIL[49]

Web Age: Tagged Music Description Formats
SMDL/SGML[50], MuXML[51], MusicML, MNML[52], MPEG7[53]

Fig.1 Short list of different Music Representations Schemes

[30] See Midi Manufacturers Ass. (MMA), http://www.midi.org 26/02/00 and [Selfridge-Field 1997] p.41-110.
[31] See [Pope 1992].
[32] See [Scaletti 1998].
[33] See Schottstaedt, W. "Common Lisp Music Documentation. Reference manual" ftp://ccrma-ftp.stanford.edu/clm/. 28/02/00 and Schottstaedt, "Common Music Notation", in:[Selfridge-Field 1997] p.217.
[34] See IRCAM/Opcode Max, http://www.opcode.com/products/max/ 26/02/00.
[35] See Vercoe, B. 1991. "CSound Manual and Release Notes" at ftp://media-lab.mit..edu 26/02/00 and David Bainbridge, "Csound", in: [Selfridge-Field 1997] p.111-142.
[36] See D. Jaffe, "The NeXT Score File" in: Selfridge-Field p.146-152 and "NeXT, Handbook and Manuals".
[37] See M. Mathews, "The Radio Baton Conductor Score File", in: [Selfridge-Field 1997] p. 153-158.
[38] See [Ossenbruggen/Eliëns 1995 + 1996]
[39] See Matsushima, in: [Selfridge-Field 1997] p.143-145.
[40] See Werner Icking, "MuTEX, MusicTEX, and MusiXTEX", in: [Selfridge-Field 1997] p.222-231.
[41] See Han Wen Nienhuys, "Lilypond", http://www.cs.uu.nl/people/hanwen/lilypond/index.html, 26/02/00.
[42] See Chris Walshaw, "abc", http://www.gre.ac.uk/~c.walshaw/abc/ 26/02/00.
[43] See Holger H. Hoos , Juergen Kilian, Keith Hamel, "GUIDO", http://www.informatik.tu-darmstadt.de/AFS/GUIDO/ 21/02/00
[44] See "Notation Interchange File Format", http://www.musitek.com/niff.html 26/02/00 and Cindy Grande, "NIFF" in [Selfridge-Field 1997] p. 491-511.
[45] See David Huron, "Humdrum and Kern", in: [Selfridge-Field 1997] , p.375-398.
[46] See Walter B. Hewlett, "MuseData", in [Selfridge-Field 1997] , p.402-445.
[47] See [Maidin 1995].
[48] See ISO/IEC JTC1/SC29 WG11, Moving Picture Expert Group, "ISO/IEC DIS 14496-1 MPEG4", http://www2.echo.lu/oii/en/video.html#MPEG-4, 26/02/00.
[49] See W3C Audio Visual Working Group, "SMIL, Synchronized Multimedia Integration Language", http://www2.echo.lu/oii/en/moving.html#SMIL, 26/02/00.
[50] ISO/IEC DIS 10743, "Standard Music Description Language (SMDL)", http://www2.echo.lu/oii/en/audio.html#SMDL, 26/02/00 and Sloan and Newcomb, "Hytime and SMDL", in: [Selfridge-Field 1997] p. 469-489.
[51] "MuXML", http://tcf.nl/3.0/musicml/index.html, 26/02/00.
[52] "The Musical Notation Markup Language", http://irdu.nus.sg/music/ 26/06/99.
[53] See ISO/IEC JTC1/SC29 WG11, Moving Picture Expert Group, "Multimedia Content Description Interface", http://www.darmstadt.gmd.de/mobile/MPEG7/Objectives.html, 26/02/00.

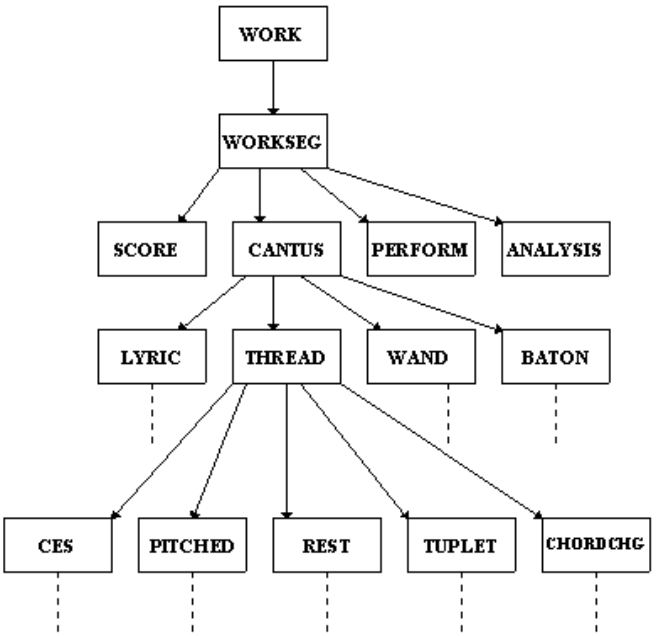
It is hard to understand why it is, that until now no music representation standard has been accepted. A list of problems of developers of music representation standards, which could explain this absence of accepted standards would encompass difficulties of collaborative development efforts as well as difficulties in the intellectual effort itself. Without going into detail, the creation of music standards involves the problematic design and implementation of a) time, b) granularity, c) representation, and d) language functionality. Within the MuTaTeD projects it was decided that rather than implementing yet another language, two existing standards, SMDL and NIFF, were to be taken in order to minimise the problems outlined above.

6. A Short Introduction to SMDL (Standard Music Description Language)

SMDL (Standard Music Description Language) is a HyTime application that conforms to international Standard ISO/IEC 10744.

There are four domains in SMDL:

- logical domain - contains the abstract musical content , described as "the composer's intentions with respect to pitches, rhythms, harmonies, dynamics, tempi, articulations, accents, etc.". It contains any number of `cantus' elements.
- gestural domain - contains any number of performances, each of which specifies how and where components of the logical domains are rendered in a specific performance, as in "the information added by performers".
- visual domain - contains any number of scores, each of which specifies exactly how components of the logical domain is rendered visually, as in "the information added by human editors, engravers, and typesetters".
- analytical domain - comprised of any number of theoretical analyses.



The process of creating an SMDL document instance involves generating a logical domain from a score or a performance, and (optionally) of generating a visual or gestural domain which represents all the correspondences between that score or performance and the logical domain. The relationships between the different domains are created using hyperlinks which may connect chunks of multimedia materials as well.

SMDL can be seen as the XML of music. The Standard Music Description Language (SMDL) is defined (in ISO/IEC Draft International Standard 10743) and builds upon SGML (ISO 8879) and HyTime (ISO/IEC 10744). Defined as "an architecture for the representation of music information, either alone, or in conjunction with text, graphics, or other information needed for publishing or business purposes" it was historically in its development closely related to HyTime[54] and is ex-

[54] For the history of the development of the standard SMDL, see ftp://ftp.techno.com/pub/HyTime/history/history.txt 26/02/00. and/ or http://coral.lili.uni-bielefeld.de/~hnord/hytime/ 26/02/00 and [Newcomb 1991+ 1992].

pected to be published in its revised form with compatibility with XML and HyTime2 in 2000., which was taken up by the US Defense Department. Not being interested in the music-side of things, rather only in the time-based structures, SMDL was created (loosely speaking) by extracting the music side out of HyTime. Nice thing about it is that it is a ISO standard, thus is more secure against any commercial or private pressures in order to change it. But only a few browsers handle it, and these are not available for all platforms. It is unbelievable that there has existed a very well defined standard (700 pages) for about three years, without anybody developing applications. We in Glasgow are still trying to get projects (EU and national) funded to work on platform independent applications using SMDL and NIFF in combination. We would definitely be delighted if an institution like IICM would join us in that effort.

7. A short introduction to NIFF (Notation Interchange File Format)

As Stephen Mounce writes in his introduction to NIFF:

> *"The NIFF project began in February 1994 with a meeting between technical people representing three major music notation programs and three music scanning programs. The group's goal was to define a new standard format for exchange of music notation data, which everyone agreed was long overdue in the industry.*
> *The original companies involved were: Passport Designs (publisher of Encore), San Andreas Press (Score), Coda Music Technology (Finale), Musitek (MidiScan) and TAP Music Systems/MusicWare (NoteScan). The list of advisors has continued to grow over time.*
> *In January of 1995 Coda decided to withdraw from the process (with the intention to publish their own Enigma format). Shortly thereafter, Mark of the Unicorn, Twelve Tone Systems, Opcode Systems, and TAP Music Systems/MusicWare agreed to replace Coda as financial sponsors."*[55]

Thus NIFF is the result of more than two years collaboration between major music software publishers and experts in the field of music notation/representation. NIFF files include graphical object and page layout as well as MIDI performance information.

Up to now, MIDI files have been the de facto standard for exchange of music data between programs. Although this is sufficient for playback, it is inadequate for displaying and printing of music notation. NIFF tried to answer this need with a dedicated notation score representation with its major functionality of being an interchange file format.

To sum up the positive characteristics of NIFF:

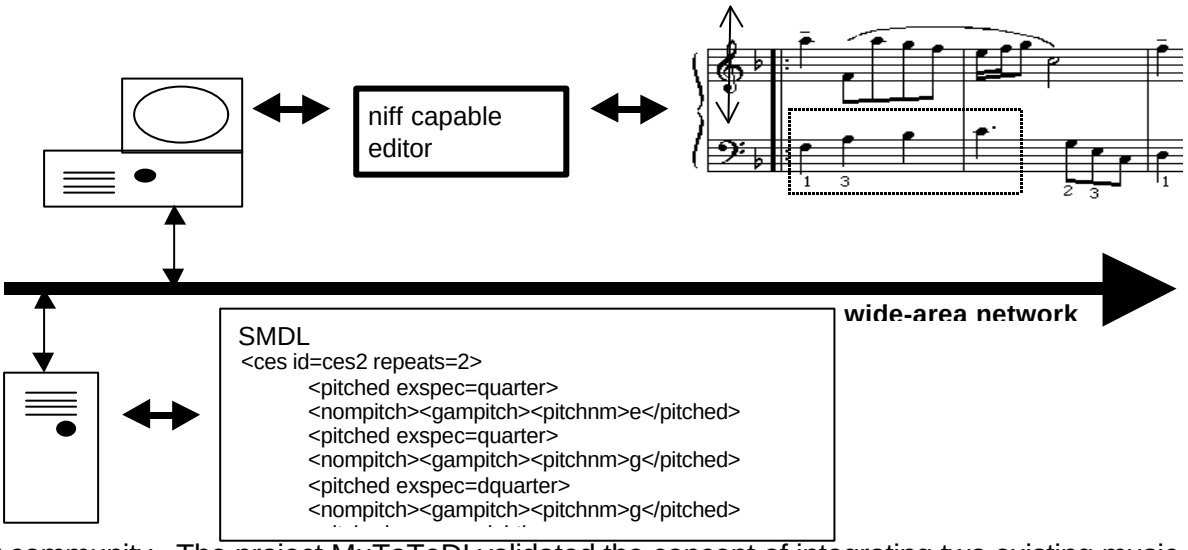
- platform-independent interchange
- intended to preserve a significant amount of visual detail
- allows representation of the most common situations occurring in conventional music
- makes provision for software developers to define their own extensions to handle the more unusual situations
- allows inclusion of Encapsulated PostScript (EPS) files and fonts to allow interchange of features not otherwise defined in the format
- the standard is open and non-proprietary.
- there are Software Developers Kits (SDKs) available

[55] Stephen Mounce (ed), "Niff Homepage", http://www.student.brad.ac.uk/srmounce/niff.html, 1/11/00

Both SMDL and NIFF seemed stable enough to start developing SMDL based and NIFF compatible services for music information retrieval. Although one may be aware of the fact that other standards are much more accepted and hyped within the music industry, and that both NIFF and SMDL are not accepted by it, they still seem to be the only standards powerful enough and close enough to being officially standardized, for developers not to be prone to the danger of working towards moving goalposts. Within this context, plans were made to set up a proof of concept project to validate the concept of integrating these two existing music representation standards.

8. MuTaTeD! – a proof of concept

Objectives of the project were: to integrate SMDL as the Model with NIFF as one possible View, and establish a standard Meta-DTD for music tagging languages, which could be used by the wide



user community. The project MuTaTeD! validated the concept of integrating two existing music representation standards. Additionally, it was to research into the development and integration of a SMDL DTD for the wider music user community and develop a web-application as a demonstrator.(see fig. 5) The work heavily influenced the "Structured Music MPEG7 proposals"[56] which were proposed in order to ensure an SMDL-compliant standard.

Fig - 5 Basic Architecture for the MuTaTeD! project

Due to the time and financial constraints of this project an early decision was taken to use freely available software development environments. For the parser and compiler the Lex/Yacc Parser Technology (lexer, parser, code generator) was chosen and a multipass compiler has been developed with this technology. To create the NIFF output files the NIFF Software Development Kit, also freely available, was used. In addition to this, we could utilize already existing multipass compilers, which convert SMDL to NIFF, developed by the CANTATE[57] project (developer: Steven Mounce). Including our multipass compilers, which also make use of the NIFF SDK we had the following compiler passes:

CANTATE	MuTaTeD!

[56] See [Boehm/Hall 1999].
[57] Cantate, Computer Access to Notation and Text in Music Libraries. http://www.svb.nl/project/cantate/cantate.htm, 11/01/00

2 Pass Compiler: NIFF --> SGML(SMDL)	Multi-Pass compiler: SMDL --> NIFF
- NIFF -> SGML(NIFF) **- SGML(NIFF) -> SGML(SMDL)**	**- SMDL --> SGML(SMDL)** **- SGML(SMDL) --> SGML(NIFF)** **- SGML(NIFF) --> C(NiffSDK) --> C-Code** **- C-Code --> NIFF binary**

The feedback from several large libraries and library projects, which have shown interest in this work, emphasized the value of having the full circle of converters from SMDL to NIFF and NIFF to SMDL available. Although the SMDL-to-NIFF converter as developed still has some restrictions in its functionality, it should be noted that MuTaTeD!, besides having prooved the concept of integrating SMDL and NIFF, it was the gfirst project to build a SMDL to NIFF converter.

8.1. The technical Set-Up

The ideal situation (see Fig. 6), as sketched below, would have been to distribute the multipass converter across server and client whilst using a platform independent client. This design would have had the advantage of client-server distribution in order to maximise the performance for the user. The conversion processes to the final binary NIFF files would be executed on the clients' machines, thus minimising the downloading time and platform dependencies. Only text-based information would be sent to the client, the binary being created within the client-side application.

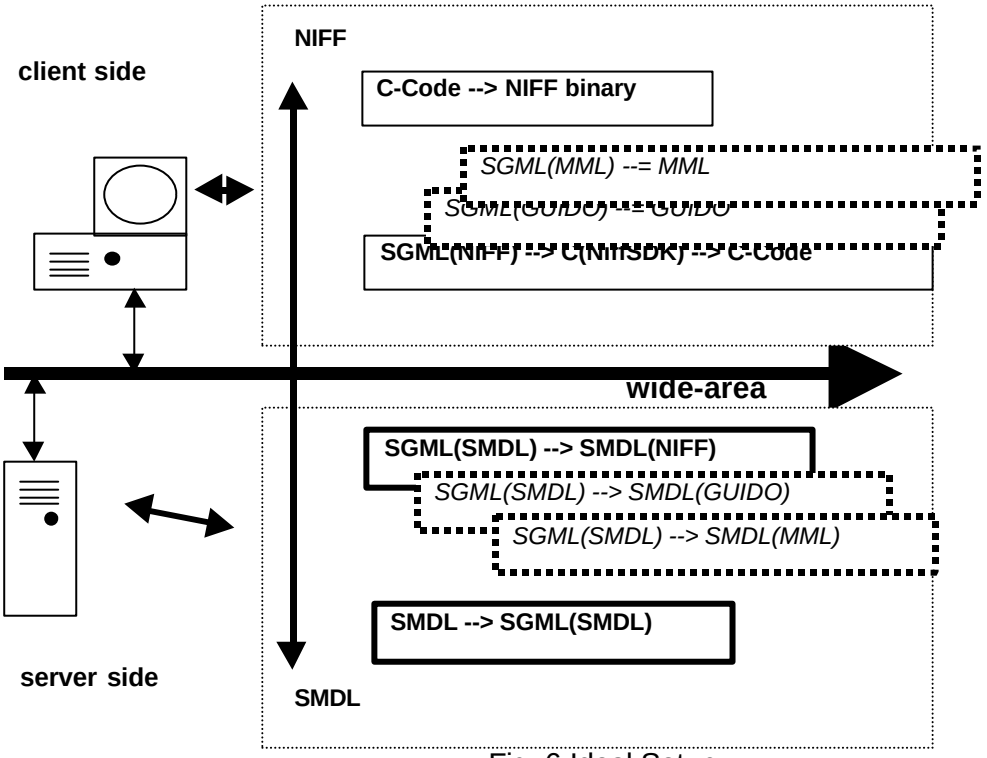
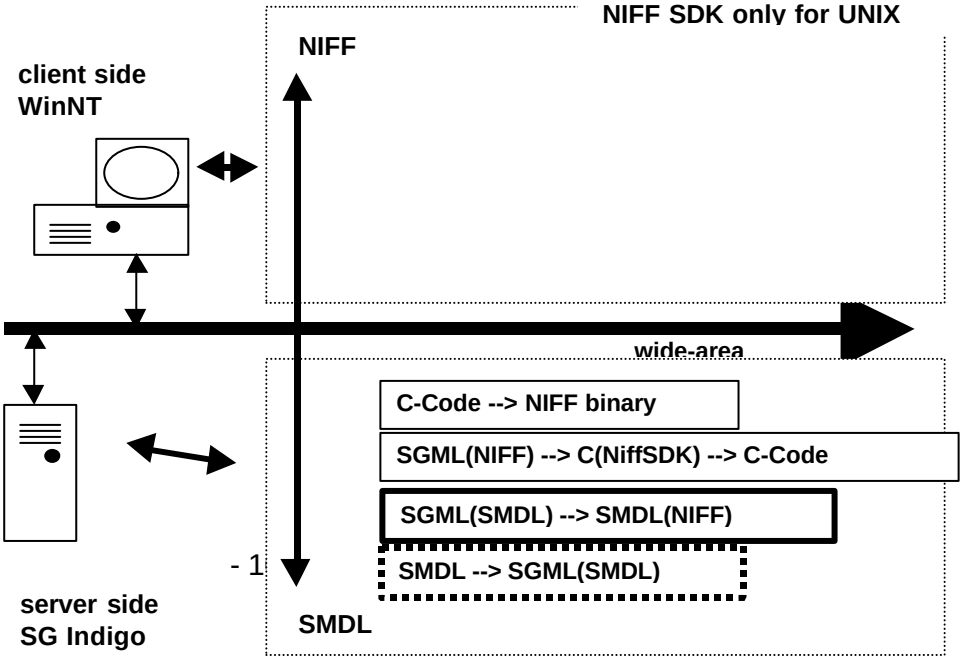


Fig. 6 Ideal Setup

Additionally this would have the advantage of involving separate passes, allowing the modularity to add different client side converters, such



as for instance GUIDO or MML. In this way it could be easily expanded.

Facing the problem that the NIFF SDK was available only for Unix, and not having the time for tedious recompilation procedures of the software tools for WinNT, the decision was made to put all of the passes onto the server side. (see Fig. 7)

Fig. 7 Real Setup

8.2. The MuTaTeD! Multipass Compilers

A compiler contains a *lexer*, a *parser* and a *code generator*. The lexer splits up the program text into what are called 'tokens', tiny pieces of information. These are then examined by the parser, which builds a tree representing the program's structure. If the program can be parsed successfully, then the tree is well-defined and is passed on to the code generator, which walks over the tree and produces target code as it examines each node in the tree. The compiler developed in MuTaTeD! works with two passes. The first translates SMDL into an intermediate form of NIFF, called SGML(NIFF)[58]. The second translates into NIFF binary code. Each of the two passes is written using LEX and YACC. The second pass translates an SGML(NIFF) file to a C program, using the NIFF SDK. This is compiled and executed, resulting in a NIFF binary file that can then be loaded into a NIFF compatible music editor, such as LIME. (see source code excerpt Fig .8 and Fig. 9)

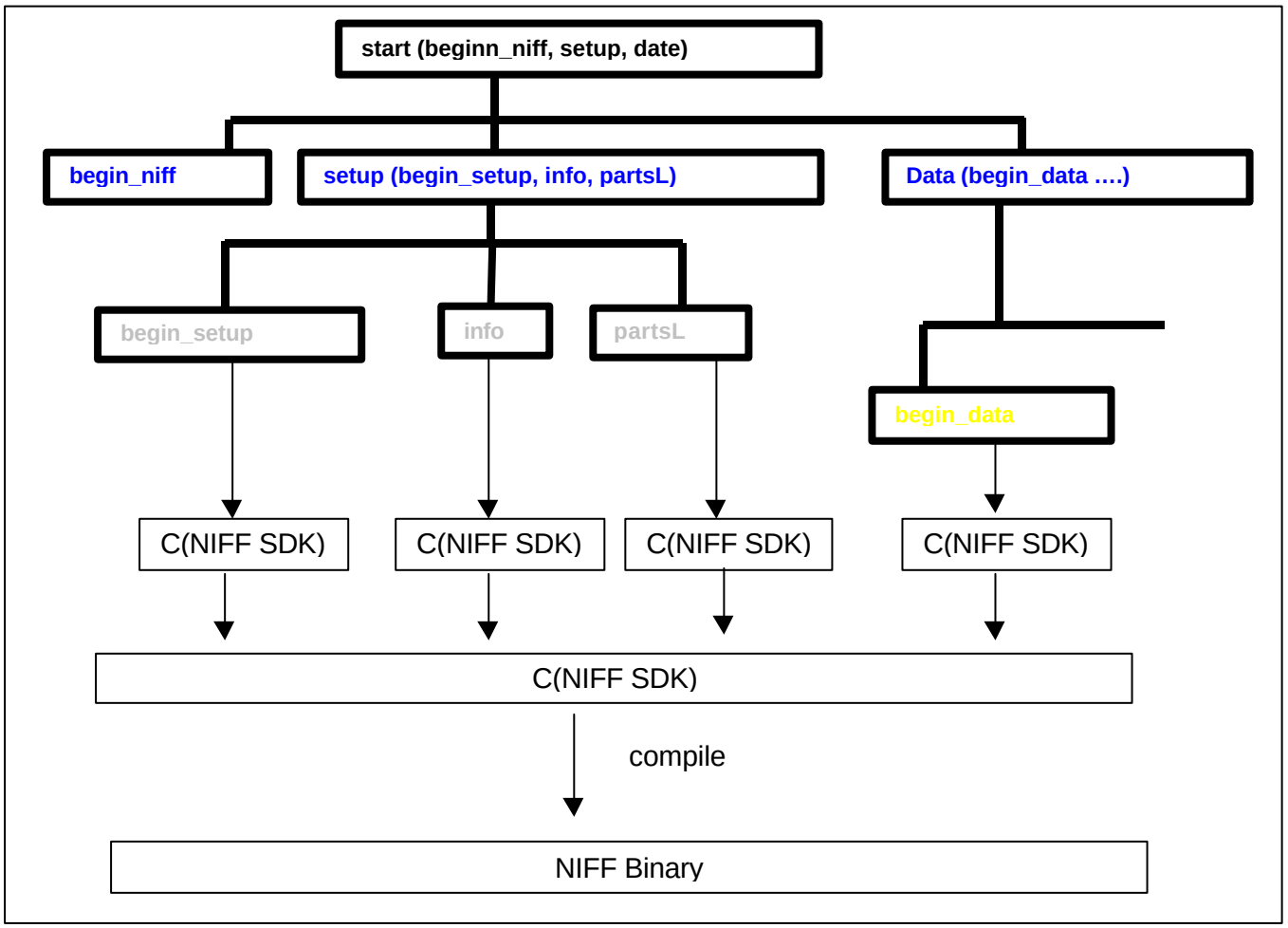


Fig. 8 Multi pass compiler, via Lex/Yacc

How the lex/yacc compilers work can shown in the following examples, deriving from the 2nd Pass, (SGML(NIFF) --> C(NIFF_SDK)):

```
start : NIFF begin_niff setup data END_NIFF end_niff

begin_niff : {printStartFile();
              printf("if (! NIFFIOStartNiff())
              doerror(\"NIFFIOStoreFormNiff failed\");\n");}
end_niff : {printf("NIFFIOEndNiff();\n");  printEndFile(); }

setup : SETUP EHJLQBVWX$LQIR$DUWV/ END_SETUP end_setup
EHJLQBVWX$ : {
printf("if (! NIFFIOStartSetupSection())
doerror(\"NIFFIOStartSetupSection failed\");\n");
          printf("if (! NIFFIOStoreDefaultCLT())
          doerror(\"NIFFIOStoreDefaultCLT failed\");\n");  }
end_setup : { printf("if (! NIFFIOchunkStringTable()) doerror
            (\"NIFFIOStringTable failed\");\n");
            printf("MyWriteStringTable();\n");
            printf("NIFFIOEndSetupSection();\n");}

LQIR : INFO begin_info VERSION INTEGER END_INFO
begin_info : {printf("if (! NIFFIOchunkNiffInfo(version, progtypeNone,
    unitsNoUnits, -1 ,-1)) doerror (\"NIFFIOStoreListSetupSection
    failed\");\n");}

$DUWV/ :  PARTSL begin_partsL part_plus END_PARTSL end_partsL
          begin_partsL : {printf("if (! NIFFIOStartParts()) doerror
    (\"NIFFIOStartParts failed\");\n");}
end_partsL : {printf("NIFFIOEndParts();\n");}

part_plus : part
part_plus part
part : PART begin_part ID IDENTIFIER NAME IDENTIFIER END_PART
begin_part : {printf("if (! NIFFIOchunkPart(0,0,0,1,-1,-1,-1)) doerror
(\"NIFFIOchunkPart failed\");\n");}

data : DATA begin_data pageL_star END_DATA end_data
begin_data : {printf("if (! NIFFIOStartDataSection()) doer-
ror(\"NIFFIOStartDataSection failed\");\n");}
end_data : {printf("NIFFIOEndDataSection();\n");}
    ….etc
    …etc
```

Fig. 8 Source Code 2nd Pass, (SGML(NIFF) --> C(NIFF_SDK))

[58] For future notation within this document, a file written in SGML and using an X DTD is described as being in SGML(X). A file written in C and using functions from the NIFF SDK is referred to as being in C(NIFF SDK).

8.3. Conclusion

Besides proving the concept of using these two standards, the use of the LEX/YACC technology enabled us to implement the converters, but also restricted our flexibility to certain extent. With LEX/YACC we had to hardcode our chosen DTD's into the converters. This is implemented differently in the follow-up project, MuTaTeD'II in which the Groveminder Technology is used. This, besides other advantages, allows for the reading in of different DTDs. NIFF, being binary, also posed some problems of accessing and searching the content. The fact that a compilation procedure has to always precede a reading in, has consequences in the design of intuitive content music retrieval out of NIFF, specifically for displaying music. So although NIFF has proved to be the powerful format, we are thinking of using another text-based representation language for future implementations.

We hope to be able to continue influencing the MPEG7 standardization process to include compatibility with SMDL. Newest developments within the finalization of the SMDL standard will include compatibility with XML and HyTime2.

Finally, MuTaTeD!'s major contribution to the music user community is the first SMDL-to-NIFF converter.

References:

[Boehm/Hall 1999] Carola Boehm, Cordy Hall, " ISO/IEC JTC1/SC29/WG11/ MPEG 98/P620/W2463, MPEG proposal: Description Scheme for description of music content", (Vancouver: MPEG 1999)

[Gould/Longemann 1966] Murray J. Gould and George W. Longemann, "ALMA, Alphameric Language for Music Analysis." Barry S. Brook "Musicology and the Computer, Musicology 1966-200: A practical Program" (New York: The City University of New York Press 1970), p.57.

[Haken/Blostein 1993] Lippold Haken and Dorothea Blostein (Eds), "The Tilia Music Representation: Extensibility, Abstraction, and Notation Contexts for the Lime Music", in: Computer Music Journal, 17:3-Fall, 1993, p.45

[Maidin 1995] Doncha o Maidin, "A Programmer's Environment for Music Analysis", PhD, Cork, IR 1995.

[Newcomb 1991] Newcomb, Steven R. "Standards. Standard Music Description Language Complies with Hypermedia Standard." IEEE Computer 24/7 (July 1991) p.76-79. ISSN: 0018-9162 and

[Newcomb 1992] Newcomb, Steven R.; Newcomb, V. T. "Some Background Information about HyTime." Journal of the Institute of Image Electronics Engineers of Japan 21/5 (October 1992) p.459-467.

[Ossenbruggen/Eliëns 1995] Jacco van Ossenbruggen, Anton Eliëns, "Bringing Music to the Web", Conference Proceedings of the World Wide Web Conference 1995. http://www.cs.vu.nl/~jrvosse/Papers/www95/paper.html 21/02/00

[Ossenbruggen/Eliëns 1996] Anton Eliëns, Martijn van Welie, Jacco van Ossenbruggen and Bastiaan Schönhage, "Jamming (on) the Web", in Proceedings of WWW6, http://www.cs.vu.nl/~jrvosse/Papers/www6/PAPER38.html 21/02/00

[Pope 1992] Pope, S. T. 1992. "The Interim DynaPiano: An Integrated Tool and Instrument for Composers." Computer Music Journal 16(3): p.73-91.

[Scaletti 1998] Scaletti, C. 1989. "The Kyma/Platypus Computer Music Workstation.", Computer Music Journal 13:(2): 23-38.

[Selfridge-Field 1997] Eleanor Selfridge-Field, "Beyond Midi", Cambridge Massachusetts, CCRMAH 1997.

[Schaffrath 1992] Helmut Schaffrath, "The ESAC Databases and MAPPET Software," in: "Computing in Musicology 8" (1992), p.66 and Schaffrath, "The Essen Associative Code", in: Selfridge-Field, p.343-359.

[Steinbeck 1982] Wolfram Steinbeck, "Struktur und Aehnlichkeit, Methoden automatisierter Melodieanalyse", in: Kieler Schriften zur Musikwissenschaft, Band XXV, Baerenreiter Kassel Basel London 1982, p.29 and p.392.

[Wolf 1963] Joahnnes Wolf, Handbuch der Notationskunde II, Hildesheim 1963, p.387-389.

IV. Detailed project report of the implementation of the prototype

(by Dr. Cordelia Hall)

1. Summary

The goal of the project was to provide a 'proof of concept' for translation of SMDL documents into NIFF binary format. The compiler that does this has two passes. The first translates SMDL into an intermediate form of NIFF which we call SGML(NIFF). This form is a text file with SGML like tags. The second translates from this intermediate language into NIFF binary code. Each of the two passes is written using LEX and YACC.

The second pass is a little more complex than the first, because it translates an SGML(NIFF) file to a C program which uses the NIFF SDK. This is then compiled and executed, resulting in a NIFF binary file that can then be loaded into a music editor such as LIME, and browsed.

The next two sections are for interested musicians who need to use the compiler (and are willing to look at files of music in music representation languages) but do not need to modify it. The rest of this document is for programmers who need to maintain and extend the compiler.

2. How to run the compiler

The MuTaTeD! compiler is located in the directory

/usr/people/mutated/mutatedWork/Compiler

and can be run from the directory

/usr/people/mutated/mutatedWork/Compiler/Run

To run it on a file <file>.smdl formatted in SMDL, copy <file>.smdl into the directory above, then go to that directory and type

java Compiler <file>.smdl <file>.nif

The file <file>.nif will then be the binary NIFF file.

The Web demonstrator requires a script in the local cgi-bin directory of the server; this will be added by Carola later.

3. A musician's guide to the MuTaTeD! Compiler

3.1. Twinkle – an extended example

In this section, we look at several versions of Twinkle, a short, well known children's song, so that we can compare the different formats used by the two passes of the compiler.

This first format is SMDL. This initial section describes the HyTime modules needed and those needed for the SMDL DTD.

```
<?HyTime VERSION "ISO/IEC 10744:1992" HYQCNT=32>
<?HyTime MODULE base desctxt>
<?HyTime MODULE measure fcsmdu axismdu>
<?HyTime MODULE locs coordloc multloc relloc>
<?HyTime MODULE links>
<?HyTime MODULE sched splitfcs grpdex exrecon>

<!DOCTYPE work SYSTEM "smdl.dtd"
[<!ENTITY score SYSTEM "twinkle.mus">
 <!ENTITY stress SYSTEM "stress.sgm">
 <!ENTITY pitch SYSTEM "pitchgam.sgm">
 <!ENTITY ficta SYSTEM "fictagam.sgm">
 <!ENTITY length SYSTEM "notedur.sgm">
 <!ENTITY extlists SYSTEM "extlists.sgm">
]>
```

The *work* entity gives the technical description of Twinkle.

```
<work>
<axis id=virtime></axis>

&stress
&pitch
&ficta
&length
&extlists

<!-- this file was created by hand -->
```

The *cantus* represents the logical structure of the music, and it is here that the notes of Twinkle appear:

```
<cantus id=can1 bibinfo=bib1>

<tempobat id=temp1>
<proscope exspec=entire>
<projectr><profun>Moderato</profun></projectr>
</proscope></tempobat>

<extlist id=entire><extent><dimlist><dimspec>
<marklist>1<marklist>-1</extlist>
```

```
<stresuse id=su1 idr=four></stresuse>
```

This thread is Twinkle's single musical line.

```
<thread id=thread1 nominst=piano>
```

The *ces* entity represents a sequence of notes. At present, we also use it to group notes so that each *ces* gives the notes on one staff line.

The key is represented by the attribute *fictagam* . When this value is *none*, then the key is C major. If this value is *flats*<*n*> or *sharps*<*n*>, then the key has the indicated number of flats or sharps.

```
<ces id=ces1 pitchgam=pgam1 fictagam=none>
```

The entity *pitched* represents a note. The attribute *exspec* determines the length of the note. If it is *quarter*, then the note is a quarter note.

The *pitchnm* tag is followed by a letter that indicates the pitch of the note. If it is *c*, then the note is middle C. If there is an accidental, then it will appear after the note, with the following form:

```
<fictadj><nsdist>0<gsdist>n</fictadj>
```

where n is –1 if a flat, 0 if a natural, and 1 if a sharp.

Here are the notes in the SMDL representation of Twinkle:

```
<pitched exspec=quarter>
<nompitch><gampitch><pitchnm>c</pitched>
<pitched exspec=quarter>
<nompitch><gampitch><pitchnm>c</pitched>
<pitched exspec=quarter>
<nompitch><gampitch><pitchnm>g</pitched>
<pitched exspec=quarter>
<nompitch><gampitch><pitchnm>g</pitched>

<pitched exspec=quarter>
<nompitch><gampitch><pitchnm>a</pitched>
<pitched exspec=quarter>
<nompitch><gampitch><pitchnm>a</pitched>
 <pitched exspec=half>
 <nompitch><gampitch><pitchnm>g</pitched>

<pitched exspec=quarter>
<nompitch><gampitch><pitchnm>f</pitched>
<pitched exspec=quarter>
<nompitch><gampitch><pitchnm>f</pitched>
<pitched exspec=quarter>
<nompitch><gampitch><pitchnm>e</pitched>
<pitched exspec=quarter>
<nompitch><gampitch><pitchnm>e</pitched>
```

```
<pitched exspec=quarter>
<nompitch><gampitch><pitchnm>d</pitched>
<pitched exspec=quarter>
<nompitch><gampitch><pitchnm>d</pitched>
 <pitched exspec=half>
 <nompitch><gampitch><pitchnm>c</pitched>

    </ces>

<ces id=ces1 pitchgam=pgam1 fictagam=none>

<pitched exspec=quarter>
<nompitch><gampitch><pitchnm>c</pitched>
<pitched exspec=quarter>
<nompitch><gampitch><pitchnm>c</pitched>
<pitched exspec=quarter>
<nompitch><gampitch><pitchnm>g</pitched>
<pitched exspec=quarter>
<nompitch><gampitch><pitchnm>g</pitched>

<pitched exspec=quarter>
<nompitch><gampitch><pitchnm>a</pitched>
<pitched exspec=quarter>
<nompitch><gampitch><pitchnm>a</pitched>
 <pitched exspec=half>
 <nompitch><gampitch><pitchnm>g</pitched>

<pitched exspec=quarter>
<nompitch><gampitch><pitchnm>f</pitched>
<pitched exspec=quarter>
<nompitch><gampitch><pitchnm>f</pitched>
<pitched exspec=quarter>
<nompitch><gampitch><pitchnm>e</pitched>
<pitched exspec=quarter>
<nompitch><gampitch><pitchnm>e</pitched>

<pitched exspec=quarter>
<nompitch><gampitch><pitchnm>d</pitched>
<pitched exspec=quarter>
<nompitch><gampitch><pitchnm>d</pitched>
 <pitched exspec=half>
 <nompitch><gampitch><pitchnm>c</pitched>

    </ces>

<ces id=ces1 pitchgam=pgam1 fictagam=none>

<pitched exspec=quarter>
<nompitch><gampitch><pitchnm>g</pitched>
<pitched exspec=quarter>
<nompitch><gampitch><pitchnm>g</pitched>
<pitched exspec=quarter>
<nompitch><gampitch><pitchnm>f</pitched>
<pitched exspec=quarter>
```

```
<nompitch><gampitch><pitchnm>f</pitched>

<pitched exspec=quarter>
<nompitch><gampitch><pitchnm>e</pitched>
<pitched exspec=quarter>
<nompitch><gampitch><pitchnm>e</pitched>
 <pitched exspec=half>
 <nompitch><gampitch><pitchnm>d</pitched>

<pitched exspec=quarter>
<nompitch><gampitch><pitchnm>g</pitched>
<pitched exspec=quarter>
<nompitch><gampitch><pitchnm>g</pitched>
<pitched exspec=quarter>
<nompitch><gampitch><pitchnm>f</pitched>
<pitched exspec=quarter>
<nompitch><gampitch><pitchnm>f</pitched>

<pitched exspec=quarter>
<nompitch><gampitch><pitchnm>e</pitched>
<pitched exspec=quarter>
<nompitch><gampitch><pitchnm>e</pitched>
 <pitched exspec=half>
 <nompitch><gampitch><pitchnm>d</pitched>

   </ces>

<ces id=ces1 pitchgam=pgam1 fictagam=none>

<pitched exspec=quarter>
<nompitch><gampitch><pitchnm>c</pitched>
<pitched exspec=quarter>
<nompitch><gampitch><pitchnm>c</pitched>
<pitched exspec=quarter>
<nompitch><gampitch><pitchnm>g</pitched>
<pitched exspec=quarter>
<nompitch><gampitch><pitchnm>g</pitched>

<pitched exspec=quarter>
<nompitch><gampitch><pitchnm>a</pitched>
<pitched exspec=quarter>
<nompitch><gampitch><pitchnm>a</pitched>
 <pitched exspec=half>
 <nompitch><gampitch><pitchnm>g</pitched>

<pitched exspec=quarter>
<nompitch><gampitch><pitchnm>f</pitched>
<pitched exspec=quarter>
<nompitch><gampitch><pitchnm>f</pitched>
<pitched exspec=quarter>
<nompitch><gampitch><pitchnm>e</pitched>
<pitched exspec=quarter>
<nompitch><gampitch><pitchnm>e</pitched>
```

```
      <pitched exspec=quarter>
      <nompitch><gampitch><pitchnm>d</pitched>
      <pitched exspec=quarter>
      <nompitch><gampitch><pitchnm>d</pitched>
       <pitched exspec=half>
       <nompitch><gampitch><pitchnm>c</pitched>

           </ces>
```

Now that we have reached the end of the *thread* and *cantus*, here are the end tags:

```
       </thread>

</cantus>
```

SMDL also provides bibliographic information, and can provide graphic information as well:

```
<bibinfo id=bib1>
<author>Unknown</author>
<uniform>Twinkle</uniform>
<isbd>unknown</isbd></bibinfo>

<nameloc id=graphic1><nmlist>score</nmlist></nameloc>
<score id=link1 linkends="graphic1 can1" extra=A intra=A></score>
</work>
```

3.2. Twinkle – SGMLniff code produced by the first pass of the MuTaTeD! Compiler

The first section of the document is the *setup* section, which defines the part names:

```
<!DOCTYPE niff SYSTEM "intdtd.sgm" []>
<niff>
   <setup>
      <info>6b 1</info>
      <partsL><part><id>p0<name>part0</part></partsL>
   </setup>
```

The other section gives the data (the notes):

```
   <data>
```

The data is organised into pages, each of which contains one or more systems. Each system contains one or more staffs:

```
      <pageL>
           <systemL>
               <staffL>
               <stafhedC></stafhedC>
```

NIFF requires a time slice to appear after each note, giving the sum of the rests and notes read so far in the measure. At the end of each bar, another time slice gives the number of bars read so far.

```
<ts><tstype>2</tstype><top>0<bottom>4</ts>
```

The *clef* entity specifies the clef and the *octave* gives the initial octave used. If it is octave 0, then the octave contains Middle C.

```
<clef><type>treble<octave>0</octave></clef>
```

The *keysig* entity gives a number that represents the key. If that number is 0, the key is C Major. If that number is between 1 and 7, then the key signature has that number of sharps; if it is between 8 and 14, then it is one of the keys with flats (the number of flats is an offset from 7).

```
<keysig>0</keysig>
```

The time signature is given by the top and bottom values:

```
<timesig><top>4</top><bottom>4</bottom></timesig>
```

A note is represented by a *stem* and a *notehead*. A notehead has a *stafstep*, which indicates the pitch as an offset from the lowest line on the staff (which has an offset of 0). In the example here, this note is a Middle C. A notehead also has a *duration*, which is given by a fraction, and a *partid*.

If there is an accidental, then it is specified after the note in a line of the form

```
<accident>n</accident>
```

where n is 2 if a flat, 3 if a natural and 4 if a sharp is required.

```
<stem></stem>
<notehead><stafstep>-2<duration>
 <top>1<bottom>4<tag><partid>p0</partid></notehead>
<ts><tstype>2</tstype><top>1<bottom>4</ts>

<stem></stem>
<notehead><stafstep>-2<duration>
 <top>1<bottom>4<tag><partid>p0</partid></notehead>
<ts><tstype>2</tstype><top>2<bottom>4</ts>

<stem></stem>
<notehead><stafstep>2<duration>
 <top>1<bottom>4<tag><partid>p0</partid></notehead>
<ts><tstype>2</tstype><top>3<bottom>4</ts>

<stem></stem>
<notehead><stafstep>2<duration>
 <top>1<bottom>4<tag><partid>p0</partid></notehead>
<ts><tstype>2</tstype><top>4<bottom>4</ts>
<ts><tstype>1</tstype><top>4<bottom>4</ts>

<stem></stem>
<notehead><stafstep>3<duration>
 <top>1<bottom>4<tag><partid>p0</partid></notehead>
```

```
    <ts><tstype>2</tstype><top>1<bottom>4</ts>

    <stem></stem>
    <notehead><stafstep>3<duration>
     <top>1<bottom>4<tag><partid>p0</partid></notehead>
    <ts><tstype>2</tstype><top>2<bottom>4</ts>

    <stem></stem>
    <notehead><stafstep>2<duration>
     <top>1<bottom>2<tag><partid>p0</partid></notehead>
    <ts><tstype>2</tstype><top>4<bottom>4</ts>
    <ts><tstype>1</tstype><top>8<bottom>4</ts>

    <stem></stem>
    <notehead><stafstep>1<duration>
     <top>1<bottom>4<tag><partid>p0</partid></notehead>
    <ts><tstype>2</tstype><top>1<bottom>4</ts>

    <stem></stem>
    <notehead><stafstep>1<duration>
     <top>1<bottom>4<tag><partid>p0</partid></notehead>
    <ts><tstype>2</tstype><top>2<bottom>4</ts>

    <stem></stem>
    <notehead><stafstep>0<duration>
     <top>1<bottom>4<tag><partid>p0</partid></notehead>
    <ts><tstype>2</tstype><top>3<bottom>4</ts>

    <stem></stem>
    <notehead><stafstep>0<duration>
     <top>1<bottom>4<tag><partid>p0</partid></notehead>
    <ts><tstype>2</tstype><top>4<bottom>4</ts>
    <ts><tstype>1</tstype><top>12<bottom>4</ts>

    <stem></stem>
    <notehead><stafstep>-1<duration>
     <top>1<bottom>4<tag><partid>p0</partid></notehead>
    <ts><tstype>2</tstype><top>1<bottom>4</ts>

    <stem></stem>
    <notehead><stafstep>-1<duration>
     <top>1<bottom>4<tag><partid>p0</partid></notehead>
    <ts><tstype>2</tstype><top>2<bottom>4</ts>

    <stem></stem>
    <notehead><stafstep>-2<duration>
     <top>1<bottom>2<tag><partid>p0</partid></notehead>
    <ts><tstype>2</tstype><top>4<bottom>4</ts>
    <ts><tstype>1</tstype><top>16<bottom>4</ts>

    </staffL>
</systemL>

<systemL>
```

```
<staffL>
<stafhedC></stafhedC>
<ts><tstype>2</tstype><top>0<bottom>4</ts>
<clef><type>treble<octave>0</octave></clef>
<keysig>0</keysig>

<stem></stem>
<notehead><stafstep>-2<duration>
 <top>1<bottom>4<tag><partid>p0</partid></notehead>
<ts><tstype>2</tstype><top>1<bottom>4</ts>

<stem></stem>
<notehead><stafstep>-2<duration>
 <top>1<bottom>4<tag><partid>p0</partid></notehead>
<ts><tstype>2</tstype><top>2<bottom>4</ts>

<stem></stem>
<notehead><stafstep>2<duration>
 <top>1<bottom>4<tag><partid>p0</partid></notehead>
<ts><tstype>2</tstype><top>3<bottom>4</ts>

<stem></stem>
<notehead><stafstep>2<duration>
 <top>1<bottom>4<tag><partid>p0</partid></notehead>
<ts><tstype>2</tstype><top>4<bottom>4</ts>
<ts><tstype>1</tstype><top>20<bottom>4</ts>

<stem></stem>
<notehead><stafstep>3<duration>
 <top>1<bottom>4<tag><partid>p0</partid></notehead>
<ts><tstype>2</tstype><top>1<bottom>4</ts>

<stem></stem>
<notehead><stafstep>3<duration>
 <top>1<bottom>4<tag><partid>p0</partid></notehead>
<ts><tstype>2</tstype><top>2<bottom>4</ts>

<stem></stem>
<notehead><stafstep>2<duration>
 <top>1<bottom>2<tag><partid>p0</partid></notehead>
<ts><tstype>2</tstype><top>4<bottom>4</ts>
<ts><tstype>1</tstype><top>24<bottom>4</ts>

<stem></stem>
<notehead><stafstep>1<duration>
 <top>1<bottom>4<tag><partid>p0</partid></notehead>
<ts><tstype>2</tstype><top>1<bottom>4</ts>

<stem></stem>
<notehead><stafstep>1<duration>
 <top>1<bottom>4<tag><partid>p0</partid></notehead>
<ts><tstype>2</tstype><top>2<bottom>4</ts>

<stem></stem>
```

```
    <notehead><stafstep>0<duration>
     <top>1<bottom>4<tag><partid>p0</partid></notehead>
    <ts><tstype>2</tstype><top>3<bottom>4</ts>

    <stem></stem>
    <notehead><stafstep>0<duration>
     <top>1<bottom>4<tag><partid>p0</partid></notehead>
    <ts><tstype>2</tstype><top>4<bottom>4</ts>
    <ts><tstype>1</tstype><top>28<bottom>4</ts>

    <stem></stem>
    <notehead><stafstep>-1<duration>
     <top>1<bottom>4<tag><partid>p0</partid></notehead>
    <ts><tstype>2</tstype><top>1<bottom>4</ts>

    <stem></stem>
    <notehead><stafstep>-1<duration>
     <top>1<bottom>4<tag><partid>p0</partid></notehead>
    <ts><tstype>2</tstype><top>2<bottom>4</ts>

    <stem></stem>
    <notehead><stafstep>-2<duration>
     <top>1<bottom>2<tag><partid>p0</partid></notehead>
    <ts><tstype>2</tstype><top>4<bottom>4</ts>
    <ts><tstype>1</tstype><top>32<bottom>4</ts>

    </staffL>
</systemL>

<systemL>
    <staffL>
    <stafhedC></stafhedC>

    <ts><tstype>2</tstype><top>0<bottom>4</ts>
    <clef><type>treble<octave>0</octave></clef>
    <keysig>0</keysig>

    <stem></stem>
    <notehead><stafstep>2<duration>
     <top>1<bottom>4<tag><partid>p0</partid></notehead>
    <ts><tstype>2</tstype><top>1<bottom>4</ts>

    <stem></stem>
    <notehead><stafstep>2<duration>
     <top>1<bottom>4<tag><partid>p0</partid></notehead>
    <ts><tstype>2</tstype><top>2<bottom>4</ts>

    <stem></stem>
    <notehead><stafstep>1<duration>
     <top>1<bottom>4<tag><partid>p0</partid></notehead>
    <ts><tstype>2</tstype><top>3<bottom>4</ts>

    <stem></stem>
    <notehead><stafstep>1<duration>
```

```
 <top>1<bottom>4<tag><partid>p0</partid></notehead>
<ts><tstype>2</tstype><top>4<bottom>4</ts>
<ts><tstype>1</tstype><top>36<bottom>4</ts>

<stem></stem>
<notehead><stafstep>0<duration>
 <top>1<bottom>4<tag><partid>p0</partid></notehead>
<ts><tstype>2</tstype><top>1<bottom>4</ts>

<stem></stem>
<notehead><stafstep>0<duration>
 <top>1<bottom>4<tag><partid>p0</partid></notehead>
<ts><tstype>2</tstype><top>2<bottom>4</ts>

<stem></stem>
<notehead><stafstep>-1<duration>
 <top>1<bottom>2<tag><partid>p0</partid></notehead>
<ts><tstype>2</tstype><top>4<bottom>4</ts>
<ts><tstype>1</tstype><top>40<bottom>4</ts>

<stem></stem>
<notehead><stafstep>2<duration>
 <top>1<bottom>4<tag><partid>p0</partid></notehead>
<ts><tstype>2</tstype><top>1<bottom>4</ts>

<stem></stem>
<notehead><stafstep>2<duration>
 <top>1<bottom>4<tag><partid>p0</partid></notehead>
<ts><tstype>2</tstype><top>2<bottom>4</ts>

<stem></stem>
<notehead><stafstep>1<duration>
 <top>1<bottom>4<tag><partid>p0</partid></notehead>
<ts><tstype>2</tstype><top>3<bottom>4</ts>

<stem></stem>
<notehead><stafstep>1<duration>
 <top>1<bottom>4<tag><partid>p0</partid></notehead>
<ts><tstype>2</tstype><top>4<bottom>4</ts>
<ts><tstype>1</tstype><top>44<bottom>4</ts>

<stem></stem>
<notehead><stafstep>0<duration>
 <top>1<bottom>4<tag><partid>p0</partid></notehead>
<ts><tstype>2</tstype><top>1<bottom>4</ts>

<stem></stem>
<notehead><stafstep>0<duration>
 <top>1<bottom>4<tag><partid>p0</partid></notehead>
<ts><tstype>2</tstype><top>2<bottom>4</ts>

<stem></stem>
<notehead><stafstep>-1<duration>
 <top>1<bottom>2<tag><partid>p0</partid></notehead>
```

```
      <ts><tstype>2</tstype><top>4<bottom>4</ts>
      <ts><tstype>1</tstype><top>48<bottom>4</ts>

      </staffL>
   </systemL>
   <systemL>
      <staffL>
      <stafhedC></stafhedC>

      <ts><tstype>2</tstype><top>0<bottom>4</ts>
      <clef><type>treble<octave>0</octave></clef>
      <keysig>0</keysig>

      <stem></stem>
      <notehead><stafstep>-2<duration>
       <top>1<bottom>4<tag><partid>p0</partid></notehead>
      <ts><tstype>2</tstype><top>1<bottom>4</ts>

      <stem></stem>
      <notehead><stafstep>-2<duration>
       <top>1<bottom>4<tag><partid>p0</partid></notehead>
      <ts><tstype>2</tstype><top>2<bottom>4</ts>

      <stem></stem>
      <notehead><stafstep>2<duration>
       <top>1<bottom>4<tag><partid>p0</partid></notehead>
      <ts><tstype>2</tstype><top>3<bottom>4</ts>

      <stem></stem>
      <notehead><stafstep>2<duration>
       <top>1<bottom>4<tag><partid>p0</partid></notehead>
      <ts><tstype>2</tstype><top>4<bottom>4</ts>
      <ts><tstype>1</tstype><top>52<bottom>4</ts>

      <stem></stem>
      <notehead><stafstep>3<duration>
       <top>1<bottom>4<tag><partid>p0</partid></notehead>
      <ts><tstype>2</tstype><top>1<bottom>4</ts>

      <stem></stem>
      <notehead><stafstep>3<duration>
       <top>1<bottom>4<tag><partid>p0</partid></notehead>
      <ts><tstype>2</tstype><top>2<bottom>4</ts>

      <stem></stem>
      <notehead><stafstep>2<duration>
       <top>1<bottom>2<tag><partid>p0</partid></notehead>
      <ts><tstype>2</tstype><top>4<bottom>4</ts>
      <ts><tstype>1</tstype><top>56<bottom>4</ts>

      <stem></stem>
      <notehead><stafstep>1<duration>
       <top>1<bottom>4<tag><partid>p0</partid></notehead>
      <ts><tstype>2</tstype><top>1<bottom>4</ts>
```

```
                <stem></stem>
                <notehead><stafstep>1<duration>
                 <top>1<bottom>4<tag><partid>p0</partid></notehead>
                <ts><tstype>2</tstype><top>2<bottom>4</ts>

                <stem></stem>
                <notehead><stafstep>0<duration>
                 <top>1<bottom>4<tag><partid>p0</partid></notehead>
                <ts><tstype>2</tstype><top>3<bottom>4</ts>

                <stem></stem>
                <notehead><stafstep>0<duration>
                 <top>1<bottom>4<tag><partid>p0</partid></notehead>
                <ts><tstype>2</tstype><top>4<bottom>4</ts>
                <ts><tstype>1</tstype><top>60<bottom>4</ts>

                <stem></stem>
                <notehead><stafstep>-1<duration>
                 <top>1<bottom>4<tag><partid>p0</partid></notehead>
                <ts><tstype>2</tstype><top>1<bottom>4</ts>

                <stem></stem>
                <notehead><stafstep>-1<duration>
                 <top>1<bottom>4<tag><partid>p0</partid></notehead>
                <ts><tstype>2</tstype><top>2<bottom>4</ts>

                <stem></stem>
                <notehead><stafstep>-2<duration>
                 <top>1<bottom>2<tag><partid>p0</partid></notehead>
                <ts><tstype>2</tstype><top>4<bottom>4</ts>
                <ts><tstype>1</tstype><top>64<bottom>4</ts>

                </staffL>
            </systemL>
        </pageL>
    </data>
</niff>
```

3.3. Twinkle – C code produced by the second pass of the MuTaTeD! Compiler

The C files use the NIFF SDK written by Tim Butler and referred to in the first part of this report. Fortunately, the intermediate language, SGMLniff, has a structure that follows NIFF very closely. As the NIFF SDK naturally follows NIFF closely as well, the compiler is a relatively simple one in which the include section is created, then the NIFF chunk calls are inserted and finally, the C code required at the end is created.

```
#include <stdio.h>
#include <string.h>
#include "niffio.h"
```

```
#include "stdcriff.h"
extern void doerror(const char *strMessage);
void MyWriteStringTable(void);
int
main(int argc, char **argv)
{
unsigned char version[8] = "6b";
FILE          *outfp;
NIFFIOFile    *pnf;
NIFFIOStorage   *pstore;
NIFFIOChunkLengthTable *pclt;
RIFFIOSuccess success;
STROFFSET offset;
RATIONAL rat;
outfp = fopen("Test1.nif", "wb");
if (!outfp)
doerror("Can't open Test1.nif for writing");
pstore = NIFFIOStorageNewSTDC(outfp);
if (!pstore)
doerror("Can't create new NIFFIOStorage");
```

Here is where the compiler starts generating code that depends upon the tokens it reads from the SGMLniff file it parses. The **NIFFIO<name>** calls correspond roughly to SGMLniff.

```
if (! NIFFIOStartNiff()) doerror("NIFFIOStoreFormNiff failed");

if (! NIFFIOStartSetupSection()) doerror("NIFFIOStartSetupSection
failed");
if (! NIFFIOStoreDefaultCLT()) doerror("NIFFIOStoreDefaultCLT failed");

if (! NIFFIOchunkNiffInfo(version, progtypeNone, unitsNoUnits, -1 ,-1))
doerror ("NIFFIOStoreListSetupSection failed");

if (! NIFFIOStartParts()) doerror ("NIFFIOStartParts failed");
if (! NIFFIOchunkPart(0,0,0,1,-1,-1,-1)) doerror ("NIFFIOchunkPart
failed");
NIFFIOEndParts();

if (! NIFFIOchunkStringTable()) doerror ("NIFFIOStringTable failed");
MyWriteStringTable();
NIFFIOEndSetupSection();

if (! NIFFIOStartDataSection()) doerror("NIFFIOStartDataSection failed");

if (! NIFFIOStartPage()) doerror("NIFFIOStartPage failed");

if (! NIFFIOStartSystem()) doerror("NIFFIOStartSystem failed");

if (! NIFFIOStartStaff()) doerror("NIFFIOStartStaffL failed");

if (! NIFFIOchunkStaffHeader()) doerror("NIFFIOchunkStaffHeader failed");
```

Durations are represented by rational numbers: the top is the numerator and the bottom is the denominator.

```
rat.numerator = 0;
rat.denominator = 4;
if (! NIFFIOchunkTimeSlice(2,rat)) doerror("NIFFIOchunkTimeSlice
failed");

if (! NIFFIOchunkClef(1,2,0)) doerror("NIFFIOchunkClef failure");

if (! NIFFIOchunkKeySignature(0)) doerror("NIFFIOchunkKeySignature
failure");

if (! NIFFIOchunkTimeSignature(4,4)) doerror("NIFFIOchunkTimeSignature
failed");

if (! NIFFIOchunkStem()) doerror("NIFFIOchunkStem failed");

rat.numerator = 1;
rat.denominator = 4;
if (! NIFFIOchunkNotehead(4,-2,rat)) doerror("NIFFIOchunkNoteHead
failed");

rat.numerator = 1;
rat.denominator = 4;
if (! NIFFIOchunkTimeSlice(2,rat)) doerror("NIFFIOchunkTimeSlice
failed");

if (! NIFFIOchunkStem()) doerror("NIFFIOchunkStem failed");

rat.numerator = 1;
rat.denominator = 4;
if (! NIFFIOchunkNotehead(4,-2,rat)) doerror("NIFFIOchunkNoteHead
failed");

rat.numerator = 2;
rat.denominator = 4;
if (! NIFFIOchunkTimeSlice(2,rat)) doerror("NIFFIOchunkTimeSlice
failed");

if (! NIFFIOchunkStem()) doerror("NIFFIOchunkStem failed");

rat.numerator = 1;
rat.denominator = 4;
if (! NIFFIOchunkNotehead(4,2,rat)) doerror("NIFFIOchunkNoteHead
failed");

rat.numerator = 3;
rat.denominator = 4;
if (! NIFFIOchunkTimeSlice(2,rat)) doerror("NIFFIOchunkTimeSlice
failed");

if (! NIFFIOchunkStem()) doerror("NIFFIOchunkStem failed");
```

```
rat.numerator = 1;
rat.denominator = 4;
if (! NIFFIOchunkNotehead(4,2,rat)) doerror("NIFFIOchunkNoteHead
failed");

rat.numerator = 4;
rat.denominator = 4;
if (! NIFFIOchunkTimeSlice(2,rat)) doerror("NIFFIOchunkTimeSlice
failed");

rat.numerator = 4;
rat.denominator = 4;
if (! NIFFIOchunkTimeSlice(1,rat)) doerror("NIFFIOchunkTimeSlice
failed");

if (! NIFFIOchunkStem()) doerror("NIFFIOchunkStem failed");

rat.numerator = 1;
rat.denominator = 4;
if (! NIFFIOchunkNotehead(4,3,rat)) doerror("NIFFIOchunkNoteHead
failed");

rat.numerator = 1;
rat.denominator = 4;
if (! NIFFIOchunkTimeSlice(2,rat)) doerror("NIFFIOchunkTimeSlice
failed");

if (! NIFFIOchunkStem()) doerror("NIFFIOchunkStem failed");

rat.numerator = 1;
rat.denominator = 4;
if (! NIFFIOchunkNotehead(4,3,rat)) doerror("NIFFIOchunkNoteHead
failed");

rat.numerator = 2;
rat.denominator = 4;
if (! NIFFIOchunkTimeSlice(2,rat)) doerror("NIFFIOchunkTimeSlice
failed");

if (! NIFFIOchunkStem()) doerror("NIFFIOchunkStem failed");

rat.numerator = 1;
rat.denominator = 2;
if (! NIFFIOchunkNotehead(3,2,rat)) doerror("NIFFIOchunkNoteHead
failed");

rat.numerator = 4;
rat.denominator = 4;
if (! NIFFIOchunkTimeSlice(2,rat)) doerror("NIFFIOchunkTimeSlice
failed");

rat.numerator = 8;
rat.denominator = 4;
```

```
if (! NIFFIOchunkTimeSlice(1,rat)) doerror("NIFFIOchunkTimeSlice
failed");

if (! NIFFIOchunkStem()) doerror("NIFFIOchunkStem failed");

rat.numerator = 1;
rat.denominator = 4;
if (! NIFFIOchunkNotehead(4,1,rat)) doerror("NIFFIOchunkNoteHead
failed");

rat.numerator = 1;
rat.denominator = 4;
if (! NIFFIOchunkTimeSlice(2,rat)) doerror("NIFFIOchunkTimeSlice
failed");

if (! NIFFIOchunkStem()) doerror("NIFFIOchunkStem failed");

rat.numerator = 1;
rat.denominator = 4;
if (! NIFFIOchunkNotehead(4,1,rat)) doerror("NIFFIOchunkNoteHead
failed");

rat.numerator = 2;
rat.denominator = 4;
if (! NIFFIOchunkTimeSlice(2,rat)) doerror("NIFFIOchunkTimeSlice
failed");

if (! NIFFIOchunkStem()) doerror("NIFFIOchunkStem failed");

rat.numerator = 1;
rat.denominator = 4;
if (! NIFFIOchunkNotehead(4,0,rat)) doerror("NIFFIOchunkNoteHead
failed");

rat.numerator = 3;
rat.denominator = 4;
if (! NIFFIOchunkTimeSlice(2,rat)) doerror("NIFFIOchunkTimeSlice
failed");

if (! NIFFIOchunkStem()) doerror("NIFFIOchunkStem failed");

rat.numerator = 1;
rat.denominator = 4;
if (! NIFFIOchunkNotehead(4,0,rat)) doerror("NIFFIOchunkNoteHead
failed");

rat.numerator = 4;
rat.denominator = 4;
if (! NIFFIOchunkTimeSlice(2,rat)) doerror("NIFFIOchunkTimeSlice
failed");

rat.numerator = 12;
rat.denominator = 4;
```

```
if (! NIFFIOchunkTimeSlice(1,rat)) doerror("NIFFIOchunkTimeSlice
failed");

if (! NIFFIOchunkStem()) doerror("NIFFIOchunkStem failed");

rat.numerator = 1;
rat.denominator = 4;
if (! NIFFIOchunkNotehead(4,-1,rat)) doerror("NIFFIOchunkNoteHead
failed");

rat.numerator = 1;
rat.denominator = 4;
if (! NIFFIOchunkTimeSlice(2,rat)) doerror("NIFFIOchunkTimeSlice
failed");

if (! NIFFIOchunkStem()) doerror("NIFFIOchunkStem failed");

rat.numerator = 1;
rat.denominator = 4;
if (! NIFFIOchunkNotehead(4,-1,rat)) doerror("NIFFIOchunkNoteHead
failed");

rat.numerator = 2;
rat.denominator = 4;
if (! NIFFIOchunkTimeSlice(2,rat)) doerror("NIFFIOchunkTimeSlice
failed");

if (! NIFFIOchunkStem()) doerror("NIFFIOchunkStem failed");

rat.numerator = 1;
rat.denominator = 2;
if (! NIFFIOchunkNotehead(3,-2,rat)) doerror("NIFFIOchunkNoteHead
failed");

rat.numerator = 4;
rat.denominator = 4;
if (! NIFFIOchunkTimeSlice(2,rat)) doerror("NIFFIOchunkTimeSlice
failed");

rat.numerator = 16;
rat.denominator = 4;
if (! NIFFIOchunkTimeSlice(1,rat)) doerror("NIFFIOchunkTimeSlice
failed");

NIFFIOEndStaff();

NIFFIOEndSystem();

if (! NIFFIOStartSystem()) doerror("NIFFIOStartSystem failed");

if (! NIFFIOStartStaff()) doerror("NIFFIOStartStaffL failed");

if (! NIFFIOchunkStaffHeader()) doerror("NIFFIOchunkStaffHeader failed");
```

```
rat.numerator = 0;
rat.denominator = 4;
if (! NIFFIOchunkTimeSlice(2,rat)) doerror("NIFFIOchunkTimeSlice
failed");

if (! NIFFIOchunkClef(1,2,0)) doerror("NIFFIOchunkClef failure");

if (! NIFFIOchunkKeySignature(0)) doerror("NIFFIOchunkKeySignature
failure");

if (! NIFFIOchunkStem()) doerror("NIFFIOchunkStem failed");

rat.numerator = 1;
rat.denominator = 4;
if (! NIFFIOchunkNotehead(4,-2,rat)) doerror("NIFFIOchunkNoteHead
failed");

rat.numerator = 1;
rat.denominator = 4;
if (! NIFFIOchunkTimeSlice(2,rat)) doerror("NIFFIOchunkTimeSlice
failed");

if (! NIFFIOchunkStem()) doerror("NIFFIOchunkStem failed");

rat.numerator = 1;
rat.denominator = 4;
if (! NIFFIOchunkNotehead(4,-2,rat)) doerror("NIFFIOchunkNoteHead
failed");

rat.numerator = 2;
rat.denominator = 4;
if (! NIFFIOchunkTimeSlice(2,rat)) doerror("NIFFIOchunkTimeSlice
failed");

if (! NIFFIOchunkStem()) doerror("NIFFIOchunkStem failed");

rat.numerator = 1;
rat.denominator = 4;
if (! NIFFIOchunkNotehead(4,2,rat)) doerror("NIFFIOchunkNoteHead
failed");

rat.numerator = 3;
rat.denominator = 4;
if (! NIFFIOchunkTimeSlice(2,rat)) doerror("NIFFIOchunkTimeSlice
failed");

if (! NIFFIOchunkStem()) doerror("NIFFIOchunkStem failed");

rat.numerator = 1;
rat.denominator = 4;
if (! NIFFIOchunkNotehead(4,2,rat)) doerror("NIFFIOchunkNoteHead
failed");

rat.numerator = 4;
```

```
rat.denominator = 4;
if (! NIFFIOchunkTimeSlice(2,rat)) doerror("NIFFIOchunkTimeSlice
failed");

rat.numerator = 20;
rat.denominator = 4;
if (! NIFFIOchunkTimeSlice(1,rat)) doerror("NIFFIOchunkTimeSlice
failed");

if (! NIFFIOchunkStem()) doerror("NIFFIOchunkStem failed");

rat.numerator = 1;
rat.denominator = 4;
if (! NIFFIOchunkNotehead(4,3,rat)) doerror("NIFFIOchunkNoteHead
failed");

rat.numerator = 1;
rat.denominator = 4;
if (! NIFFIOchunkTimeSlice(2,rat)) doerror("NIFFIOchunkTimeSlice
failed");

if (! NIFFIOchunkStem()) doerror("NIFFIOchunkStem failed");

rat.numerator = 1;
rat.denominator = 4;
if (! NIFFIOchunkNotehead(4,3,rat)) doerror("NIFFIOchunkNoteHead
failed");

rat.numerator = 2;
rat.denominator = 4;
if (! NIFFIOchunkTimeSlice(2,rat)) doerror("NIFFIOchunkTimeSlice
failed");

if (! NIFFIOchunkStem()) doerror("NIFFIOchunkStem failed");

rat.numerator = 1;
rat.denominator = 2;
if (! NIFFIOchunkNotehead(3,2,rat)) doerror("NIFFIOchunkNoteHead
failed");

rat.numerator = 4;
rat.denominator = 4;
if (! NIFFIOchunkTimeSlice(2,rat)) doerror("NIFFIOchunkTimeSlice
failed");

rat.numerator = 24;
rat.denominator = 4;
if (! NIFFIOchunkTimeSlice(1,rat)) doerror("NIFFIOchunkTimeSlice
failed");

if (! NIFFIOchunkStem()) doerror("NIFFIOchunkStem failed");

rat.numerator = 1;
rat.denominator = 4;
```

```
if (! NIFFIOchunkNotehead(4,1,rat)) doerror("NIFFIOchunkNoteHead
failed");

rat.numerator = 1;
rat.denominator = 4;
if (! NIFFIOchunkTimeSlice(2,rat)) doerror("NIFFIOchunkTimeSlice
failed");

if (! NIFFIOchunkStem()) doerror("NIFFIOchunkStem failed");

rat.numerator = 1;
rat.denominator = 4;
if (! NIFFIOchunkNotehead(4,1,rat)) doerror("NIFFIOchunkNoteHead
failed");

rat.numerator = 2;
rat.denominator = 4;
if (! NIFFIOchunkTimeSlice(2,rat)) doerror("NIFFIOchunkTimeSlice
failed");

if (! NIFFIOchunkStem()) doerror("NIFFIOchunkStem failed");

rat.numerator = 1;
rat.denominator = 4;
if (! NIFFIOchunkNotehead(4,0,rat)) doerror("NIFFIOchunkNoteHead
failed");

rat.numerator = 3;
rat.denominator = 4;
if (! NIFFIOchunkTimeSlice(2,rat)) doerror("NIFFIOchunkTimeSlice
failed");

if (! NIFFIOchunkStem()) doerror("NIFFIOchunkStem failed");

rat.numerator = 1;
rat.denominator = 4;
if (! NIFFIOchunkNotehead(4,0,rat)) doerror("NIFFIOchunkNoteHead
failed");

rat.numerator = 4;
rat.denominator = 4;
if (! NIFFIOchunkTimeSlice(2,rat)) doerror("NIFFIOchunkTimeSlice
failed");

rat.numerator = 28;
rat.denominator = 4;
if (! NIFFIOchunkTimeSlice(1,rat)) doerror("NIFFIOchunkTimeSlice
failed");

if (! NIFFIOchunkStem()) doerror("NIFFIOchunkStem failed");

rat.numerator = 1;
rat.denominator = 4;
```

```
if (! NIFFIOchunkNotehead(4,-1,rat)) doerror("NIFFIOchunkNoteHead
failed");

rat.numerator = 1;
rat.denominator = 4;
if (! NIFFIOchunkTimeSlice(2,rat)) doerror("NIFFIOchunkTimeSlice
failed");

if (! NIFFIOchunkStem()) doerror("NIFFIOchunkStem failed");

rat.numerator = 1;
rat.denominator = 4;
if (! NIFFIOchunkNotehead(4,-1,rat)) doerror("NIFFIOchunkNoteHead
failed");

rat.numerator = 2;
rat.denominator = 4;
if (! NIFFIOchunkTimeSlice(2,rat)) doerror("NIFFIOchunkTimeSlice
failed");

if (! NIFFIOchunkStem()) doerror("NIFFIOchunkStem failed");

rat.numerator = 1;
rat.denominator = 2;
if (! NIFFIOchunkNotehead(3,-2,rat)) doerror("NIFFIOchunkNoteHead
failed");

rat.numerator = 4;
rat.denominator = 4;
if (! NIFFIOchunkTimeSlice(2,rat)) doerror("NIFFIOchunkTimeSlice
failed");

rat.numerator = 32;
rat.denominator = 4;
if (! NIFFIOchunkTimeSlice(1,rat)) doerror("NIFFIOchunkTimeSlice
failed");

NIFFIOEndStaff();

NIFFIOEndSystem();

if (! NIFFIOStartSystem()) doerror("NIFFIOStartSystem failed");

if (! NIFFIOStartStaff()) doerror("NIFFIOStartStaffL failed");

if (! NIFFIOchunkStaffHeader()) doerror("NIFFIOchunkStaffHeader failed");

rat.numerator = 0;
rat.denominator = 4;
if (! NIFFIOchunkTimeSlice(2,rat)) doerror("NIFFIOchunkTimeSlice
failed");

if (! NIFFIOchunkClef(1,2,0)) doerror("NIFFIOchunkClef failure");
```

```
if (! NIFFIOchunkKeySignature(0)) doerror("NIFFIOchunkKeySignature
failure");

if (! NIFFIOchunkStem()) doerror("NIFFIOchunkStem failed");

rat.numerator = 1;
rat.denominator = 4;
if (! NIFFIOchunkNotehead(4,2,rat)) doerror("NIFFIOchunkNoteHead
failed");

rat.numerator = 1;
rat.denominator = 4;
if (! NIFFIOchunkTimeSlice(2,rat)) doerror("NIFFIOchunkTimeSlice
failed");

if (! NIFFIOchunkStem()) doerror("NIFFIOchunkStem failed");

rat.numerator = 1;
rat.denominator = 4;
if (! NIFFIOchunkNotehead(4,2,rat)) doerror("NIFFIOchunkNoteHead
failed");

rat.numerator = 2;
rat.denominator = 4;
if (! NIFFIOchunkTimeSlice(2,rat)) doerror("NIFFIOchunkTimeSlice
failed");

if (! NIFFIOchunkStem()) doerror("NIFFIOchunkStem failed");

rat.numerator = 1;
rat.denominator = 4;
if (! NIFFIOchunkNotehead(4,1,rat)) doerror("NIFFIOchunkNoteHead
failed");

rat.numerator = 3;
rat.denominator = 4;
if (! NIFFIOchunkTimeSlice(2,rat)) doerror("NIFFIOchunkTimeSlice
failed");

if (! NIFFIOchunkStem()) doerror("NIFFIOchunkStem failed");

rat.numerator = 1;
rat.denominator = 4;
if (! NIFFIOchunkNotehead(4,1,rat)) doerror("NIFFIOchunkNoteHead
failed");

rat.numerator = 4;
rat.denominator = 4;
if (! NIFFIOchunkTimeSlice(2,rat)) doerror("NIFFIOchunkTimeSlice
failed");

rat.numerator = 36;
rat.denominator = 4;
```

```
if (! NIFFIOchunkTimeSlice(1,rat)) doerror("NIFFIOchunkTimeSlice
failed");

if (! NIFFIOchunkStem()) doerror("NIFFIOchunkStem failed");

rat.numerator = 1;
rat.denominator = 4;
if (! NIFFIOchunkNotehead(4,0,rat)) doerror("NIFFIOchunkNoteHead
failed");

rat.numerator = 1;
rat.denominator = 4;
if (! NIFFIOchunkTimeSlice(2,rat)) doerror("NIFFIOchunkTimeSlice
failed");

if (! NIFFIOchunkStem()) doerror("NIFFIOchunkStem failed");

rat.numerator = 1;
rat.denominator = 4;
if (! NIFFIOchunkNotehead(4,0,rat)) doerror("NIFFIOchunkNoteHead
failed");

rat.numerator = 2;
rat.denominator = 4;
if (! NIFFIOchunkTimeSlice(2,rat)) doerror("NIFFIOchunkTimeSlice
failed");

if (! NIFFIOchunkStem()) doerror("NIFFIOchunkStem failed");

rat.numerator = 1;
rat.denominator = 2;
if (! NIFFIOchunkNotehead(3,-1,rat)) doerror("NIFFIOchunkNoteHead
failed");

rat.numerator = 4;
rat.denominator = 4;
if (! NIFFIOchunkTimeSlice(2,rat)) doerror("NIFFIOchunkTimeSlice
failed");

rat.numerator = 40;
rat.denominator = 4;
if (! NIFFIOchunkTimeSlice(1,rat)) doerror("NIFFIOchunkTimeSlice
failed");

if (! NIFFIOchunkStem()) doerror("NIFFIOchunkStem failed");

rat.numerator = 1;
rat.denominator = 4;
if (! NIFFIOchunkNotehead(4,2,rat)) doerror("NIFFIOchunkNoteHead
failed");

rat.numerator = 1;
rat.denominator = 4;
```

```
if (! NIFFIOchunkTimeSlice(2,rat)) doerror("NIFFIOchunkTimeSlice
failed");

if (! NIFFIOchunkStem()) doerror("NIFFIOchunkStem failed");

rat.numerator = 1;
rat.denominator = 4;
if (! NIFFIOchunkNotehead(4,2,rat)) doerror("NIFFIOchunkNoteHead
failed");

rat.numerator = 2;
rat.denominator = 4;
if (! NIFFIOchunkTimeSlice(2,rat)) doerror("NIFFIOchunkTimeSlice
failed");

if (! NIFFIOchunkStem()) doerror("NIFFIOchunkStem failed");

rat.numerator = 1;
rat.denominator = 4;
if (! NIFFIOchunkNotehead(4,1,rat)) doerror("NIFFIOchunkNoteHead
failed");

rat.numerator = 3;
rat.denominator = 4;
if (! NIFFIOchunkTimeSlice(2,rat)) doerror("NIFFIOchunkTimeSlice
failed");

if (! NIFFIOchunkStem()) doerror("NIFFIOchunkStem failed");

rat.numerator = 1;
rat.denominator = 4;
if (! NIFFIOchunkNotehead(4,1,rat)) doerror("NIFFIOchunkNoteHead
failed");

rat.numerator = 4;
rat.denominator = 4;
if (! NIFFIOchunkTimeSlice(2,rat)) doerror("NIFFIOchunkTimeSlice
failed");

rat.numerator = 44;
rat.denominator = 4;
if (! NIFFIOchunkTimeSlice(1,rat)) doerror("NIFFIOchunkTimeSlice
failed");

if (! NIFFIOchunkStem()) doerror("NIFFIOchunkStem failed");

rat.numerator = 1;
rat.denominator = 4;
if (! NIFFIOchunkNotehead(4,0,rat)) doerror("NIFFIOchunkNoteHead
failed");

rat.numerator = 1;
rat.denominator = 4;
```

```
if (! NIFFIOchunkTimeSlice(2,rat)) doerror("NIFFIOchunkTimeSlice
failed");

if (! NIFFIOchunkStem()) doerror("NIFFIOchunkStem failed");

rat.numerator = 1;
rat.denominator = 4;
if (! NIFFIOchunkNotehead(4,0,rat)) doerror("NIFFIOchunkNoteHead
failed");

rat.numerator = 2;
rat.denominator = 4;
if (! NIFFIOchunkTimeSlice(2,rat)) doerror("NIFFIOchunkTimeSlice
failed");

if (! NIFFIOchunkStem()) doerror("NIFFIOchunkStem failed");

rat.numerator = 1;
rat.denominator = 2;
if (! NIFFIOchunkNotehead(3,-1,rat)) doerror("NIFFIOchunkNoteHead
failed");

rat.numerator = 4;
rat.denominator = 4;
if (! NIFFIOchunkTimeSlice(2,rat)) doerror("NIFFIOchunkTimeSlice
failed");

rat.numerator = 48;
rat.denominator = 4;
if (! NIFFIOchunkTimeSlice(1,rat)) doerror("NIFFIOchunkTimeSlice
failed");

NIFFIOEndStaff();

NIFFIOEndSystem();

if (! NIFFIOStartSystem()) doerror("NIFFIOStartSystem failed");

if (! NIFFIOStartStaff()) doerror("NIFFIOStartStaffL failed");

if (! NIFFIOchunkStaffHeader()) doerror("NIFFIOchunkStaffHeader failed");

rat.numerator = 0;
rat.denominator = 4;
if (! NIFFIOchunkTimeSlice(2,rat)) doerror("NIFFIOchunkTimeSlice
failed");

if (! NIFFIOchunkClef(1,2,0)) doerror("NIFFIOchunkClef failure");

if (! NIFFIOchunkKeySignature(0)) doerror("NIFFIOchunkKeySignature
failure");

if (! NIFFIOchunkStem()) doerror("NIFFIOchunkStem failed");
```

```
rat.numerator = 1;
rat.denominator = 4;
if (! NIFFIOchunkNotehead(4,-2,rat)) doerror("NIFFIOchunkNoteHead
failed");

rat.numerator = 1;
rat.denominator = 4;
if (! NIFFIOchunkTimeSlice(2,rat)) doerror("NIFFIOchunkTimeSlice
failed");

if (! NIFFIOchunkStem()) doerror("NIFFIOchunkStem failed");

rat.numerator = 1;
rat.denominator = 4;
if (! NIFFIOchunkNotehead(4,-2,rat)) doerror("NIFFIOchunkNoteHead
failed");

rat.numerator = 2;
rat.denominator = 4;
if (! NIFFIOchunkTimeSlice(2,rat)) doerror("NIFFIOchunkTimeSlice
failed");

if (! NIFFIOchunkStem()) doerror("NIFFIOchunkStem failed");

rat.numerator = 1;
rat.denominator = 4;
if (! NIFFIOchunkNotehead(4,2,rat)) doerror("NIFFIOchunkNoteHead
failed");

rat.numerator = 3;
rat.denominator = 4;
if (! NIFFIOchunkTimeSlice(2,rat)) doerror("NIFFIOchunkTimeSlice
failed");

if (! NIFFIOchunkStem()) doerror("NIFFIOchunkStem failed");

rat.numerator = 1;
rat.denominator = 4;
if (! NIFFIOchunkNotehead(4,2,rat)) doerror("NIFFIOchunkNoteHead
failed");

rat.numerator = 4;
rat.denominator = 4;
if (! NIFFIOchunkTimeSlice(2,rat)) doerror("NIFFIOchunkTimeSlice
failed");

rat.numerator = 52;
rat.denominator = 4;
if (! NIFFIOchunkTimeSlice(1,rat)) doerror("NIFFIOchunkTimeSlice
failed");

if (! NIFFIOchunkStem()) doerror("NIFFIOchunkStem failed");

rat.numerator = 1;
```

```
rat.denominator = 4;
if (! NIFFIOchunkNotehead(4,3,rat)) doerror("NIFFIOchunkNoteHead
failed");

rat.numerator = 1;
rat.denominator = 4;
if (! NIFFIOchunkTimeSlice(2,rat)) doerror("NIFFIOchunkTimeSlice
failed");

if (! NIFFIOchunkStem()) doerror("NIFFIOchunkStem failed");

rat.numerator = 1;
rat.denominator = 4;
if (! NIFFIOchunkNotehead(4,3,rat)) doerror("NIFFIOchunkNoteHead
failed");

rat.numerator = 2;
rat.denominator = 4;
if (! NIFFIOchunkTimeSlice(2,rat)) doerror("NIFFIOchunkTimeSlice
failed");

if (! NIFFIOchunkStem()) doerror("NIFFIOchunkStem failed");

rat.numerator = 1;
rat.denominator = 2;
if (! NIFFIOchunkNotehead(3,2,rat)) doerror("NIFFIOchunkNoteHead
failed");

rat.numerator = 4;
rat.denominator = 4;
if (! NIFFIOchunkTimeSlice(2,rat)) doerror("NIFFIOchunkTimeSlice
failed");

rat.numerator = 56;
rat.denominator = 4;
if (! NIFFIOchunkTimeSlice(1,rat)) doerror("NIFFIOchunkTimeSlice
failed");

if (! NIFFIOchunkStem()) doerror("NIFFIOchunkStem failed");

rat.numerator = 1;
rat.denominator = 4;
if (! NIFFIOchunkNotehead(4,1,rat)) doerror("NIFFIOchunkNoteHead
failed");

rat.numerator = 1;
rat.denominator = 4;
if (! NIFFIOchunkTimeSlice(2,rat)) doerror("NIFFIOchunkTimeSlice
failed");

if (! NIFFIOchunkStem()) doerror("NIFFIOchunkStem failed");

rat.numerator = 1;
rat.denominator = 4;
```

```
if (! NIFFIOchunkNotehead(4,1,rat)) doerror("NIFFIOchunkNoteHead
failed");

rat.numerator = 2;
rat.denominator = 4;
if (! NIFFIOchunkTimeSlice(2,rat)) doerror("NIFFIOchunkTimeSlice
failed");

if (! NIFFIOchunkStem()) doerror("NIFFIOchunkStem failed");

rat.numerator = 1;
rat.denominator = 4;
if (! NIFFIOchunkNotehead(4,0,rat)) doerror("NIFFIOchunkNoteHead
failed");

rat.numerator = 3;
rat.denominator = 4;
if (! NIFFIOchunkTimeSlice(2,rat)) doerror("NIFFIOchunkTimeSlice
failed");

if (! NIFFIOchunkStem()) doerror("NIFFIOchunkStem failed");

rat.numerator = 1;
rat.denominator = 4;
if (! NIFFIOchunkNotehead(4,0,rat)) doerror("NIFFIOchunkNoteHead
failed");

rat.numerator = 4;
rat.denominator = 4;
if (! NIFFIOchunkTimeSlice(2,rat)) doerror("NIFFIOchunkTimeSlice
failed");

rat.numerator = 60;
rat.denominator = 4;
if (! NIFFIOchunkTimeSlice(1,rat)) doerror("NIFFIOchunkTimeSlice
failed");

if (! NIFFIOchunkStem()) doerror("NIFFIOchunkStem failed");

rat.numerator = 1;
rat.denominator = 4;
if (! NIFFIOchunkNotehead(4,-1,rat)) doerror("NIFFIOchunkNoteHead
failed");

rat.numerator = 1;
rat.denominator = 4;
if (! NIFFIOchunkTimeSlice(2,rat)) doerror("NIFFIOchunkTimeSlice
failed");

if (! NIFFIOchunkStem()) doerror("NIFFIOchunkStem failed");

rat.numerator = 1;
rat.denominator = 4;
```

```
if (! NIFFIOchunkNotehead(4,-1,rat)) doerror("NIFFIOchunkNoteHead
failed");

rat.numerator = 2;
rat.denominator = 4;
if (! NIFFIOchunkTimeSlice(2,rat)) doerror("NIFFIOchunkTimeSlice
failed");

if (! NIFFIOchunkStem()) doerror("NIFFIOchunkStem failed");

rat.numerator = 1;
rat.denominator = 2;
if (! NIFFIOchunkNotehead(3,-2,rat)) doerror("NIFFIOchunkNoteHead
failed");

rat.numerator = 4;
rat.denominator = 4;
if (! NIFFIOchunkTimeSlice(2,rat)) doerror("NIFFIOchunkTimeSlice
failed");

rat.numerator = 64;
rat.denominator = 4;
if (! NIFFIOchunkTimeSlice(1,rat)) doerror("NIFFIOchunkTimeSlice
failed");

NIFFIOEndStaff();

NIFFIOEndSystem();

NIFFIOEndPage();

NIFFIOEndDataSection();

NIFFIOEndNiff();
```

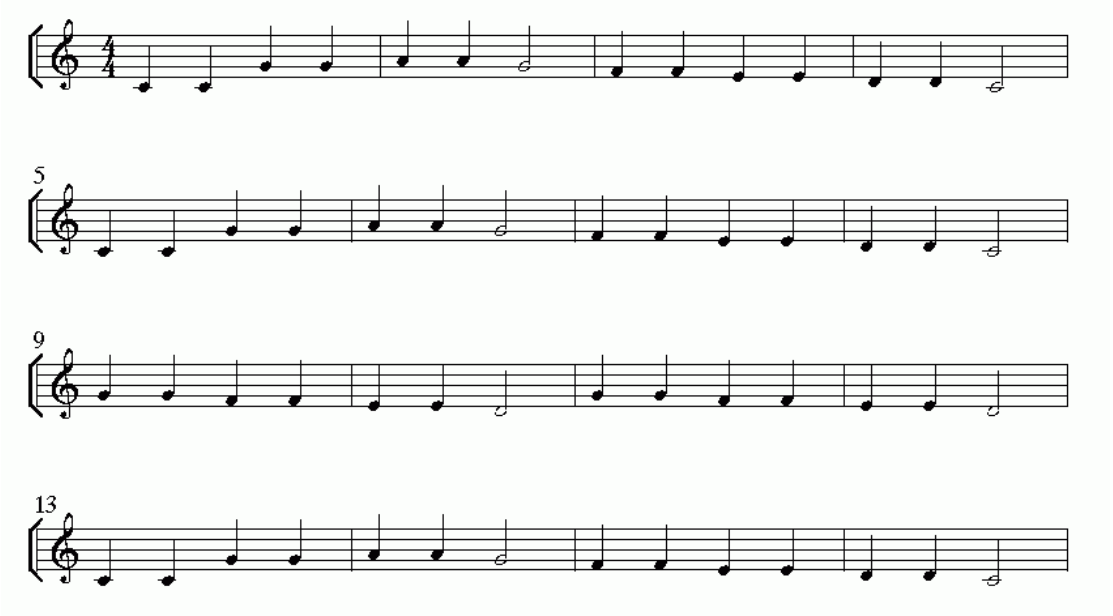
The rest of the C code is generated in one piece by the MuTaTeD compiler.

```
NIFFIOStorageDelete(pstore);
fclose(outfp);
return 0;
}
void
doerror(const char *strMessage)
    {
     fprintf(stderr, "%s\n", strMessage);
       exit(1);
    }
void MyWriteStringTable (void)
{char strings[256][20];
NIFFIOStbl myStbl[20];
short i;
int nWritten;
strcpy(strings[0], "piano");
```

```
strcpy(strings[1], "p");
strcpy(strings[2], "violin");
strcpy(strings[3], "vl");
for (i=0; i < 4; i++) myStbl[i].str = strings[i];
nWritten = NIFFIOStoreStbl(myStbl, 4);
if (nWritten != 4)  {doerror("Couldn't write strings");}
}
```

3.4. Twinkle – in LIME

Finally, here is the image of the binary NIFF file produced in the music editor LIME:



3.5. Limitations of the MuTaTeD Compiler

This compiler was intended to be a 'proof of concept' rather than a full-blown compiler that accepts a large variety of SMDL files. As such, it has significant limitations, and produces good results only for pieces that contain

- A single line using the treble clef
- Octaves 0, 1 and 2
- 4/4 meter
- no repeats

Pieces containing flagged notes do not translate well because there is an ambiguity that has to be resolved. A note containing a flag may be part of a beamed group or have a flag by itself, depending upon decisions to be made by the writer, and given the timespan for this project, this is best left to later work.

The second pass of the compiler is more robust and can handle chorales.

4. A programmer's guide to the implementation

This section is for the programmer who wants to extend the compiler.

There are two versions of the MuTaTeD compiler. The first is in the directory

/usr/people/mutated/mutatedWork/Compiler

and supports the Web Demonstrator. This is a stable version of the compiler that can handle enough of the SMDL language to be a 'proof of concept', but is not a real compiler. The second is under development in the directory

/usr/people/mutated/mutatedWork/Development

Both of these directories have README files that describe their contents.

4.1. Compiler directory - overview

Here is an overview of the Compiler directory and its subdirectories:

Examples_NIFF contains NIFF files produced by compilation of the test files in Examples_SMDL. These are:

Achild.nif
Alabama.nif
Chinese.nif
Larle.nif
Tchinese.nif
Twinkle.nif

Examples_SMDL contains SMDL files that can currently be compiled by the stable version of the compiler. These are:

Achild.smdl
Alabama.smdl
Chinese.smdl
Larle.smdl
Tchinese.smdl
Twinkle.smdl

SGMLniff_NIFF contains the code for the second pass of the compiler.

mk_niffCG – script that builds the object file
niffCG – object file that performs the second pass of the MuTaTeD compiler
niffCG.y – YACC file (parser)
nifflex.l – LEX file (lexical analyser)

SMDL_SGMLniff contains the code for the first pass of the compiler

mk_smdlCG – script that builds the object file
smdlCG – object file that performs the first pass of the MuTaTeD compiler
smdlyacc.y – YACC file (parser)
smdllex.l – LEX file (lexical analyser)

Run contains code that, when executed, will run the compiler (performing two passes).

CompilerScript – script that runs the compiler
Compiler.java – Java program that calls CompilerScript

The NIFF SDK is in the directory

/usr/people/mutated/mutatedWork/ O_niffsdk-1_02

It contains a subdirectory in which examples can be compiled and executed, called

niffsdk1.02/niffio/src/example/Test1

as shown in the script CompilerScript. Once the code generator of the MuTaTed! compiler has finished producing the C file containing calls to the NIFF SDK, that file is placed in the Test1 directory, and the C executable is then placed by the SDK into the directory

niffsdk1.02/niffio/test

This can then be executed with the SDK library by the command

../bin/Test1

Unfortunately, this process still produces a core file, and so that file has to be removed. The binary NIFF file is then

niffsdk1.02/niffio/test/Test1.nif

which is renamed and moved by the CompilerScript script.

5. A programmer's guide to LEX and YACC

First we discuss how a compiler works, then the software that it generally uses. Before this, we define some useful terms.

A *pointer* is a link from one piece of data to another.

A *tree* is a structure that is composed of lots of nodes, each of which has two possible shapes. The first shape belongs to the interior nodes that hold the tree together. An interior node has some data, and two pointers, each of which points to a node. The second shape belongs to a leaf node, which has data but no pointers.

A *stack* is a structure which can be viewed and accessed only from the top. A new value may be put into the stack using a *push* operation. The stack then contains that element on its top. The top element may then be removed using a *pop* operation.

A compiler contains a *lexer*, a *parser* and a *code generator*. The lexer splits up the program text into what are called 'tokens', tiny pieces of information. These are then examined by the parser, which builds a tree representing the program's structure. If the program can be parsed successfully, then the tree is well-defined and is passed on to the code generator, which walks over the tree and produces target code as it examines each node in the tree.

A *regular expression* is an expression that is composed of letters, digits and special symbols. It specifies the structure of the expression being recognised by a lexer.

5.1. A high level view of the structure of a compiler

We now go on to give a more detailed view of a compiler's structure, so that it is easier to understand what the standard software does. This is necessary if you are to alter the compiler.

A *lexer* takes a text file and converts it into a stream of pieces of data called *tokens*. These tokens each have a type. For example, given the file

"This file has 6 words in it!"

a lexer might produce the following tokens:

IDENTIFIER This
IDENTIFIER file
IDENTIFIER has
NUM 6
IDENTIFIER words
IDENTIFIER in
IDENTIFIER it
EXCLAM !

The types are passed on to the *parser*, which uses them to determine whether the file has a correct program. For example, given the file

“4 + (3 + 2 * 9)”

the lexer would produce the tokens

NUM 4
PLUS
LPAREN
NUM 3
PLUS
NUM 2
MUL
NUM 9
RPAREN

and the parser could then check its structure by failing if PLUS and MUL do not have 2 arguments and/or the parentheses do not match. If they do, then the parser could build the tree

```
          PLUS
NUM 4           PLUS
          NUM 3       MUL
                  NUM 2  NUM 9
```

A *code generator* could then traverse the tree and generate the code

4
3
2
9
*
+
+

for a stack machine, which would push all of the arguments on the stack, then pop them and apply the operators as it encountered them on the bottom of the stack. For example, it would push 4,3,2,9, then when it read *, it would know that 2 operands are required, collect them from the top of the stack, perform the multiplication and then push the result. The stack would then have 4,3,11, and the same thing would be done for each of the remaining operators + and +. Code generators usually produce a file that contains machine code such as

```
LDVAL     R1,#0002
LDVAL     R2,#0009
MUL       R3,R1,R2
LDVAL     R4,#0003
```

```
ADD     R5,R3,R4
LDVAL   R6,#0004
ADD     R7,R5,R6
```

Notice the order in which the operations take place. The code generator must traverse the tree nodes in the correct order to produce this code, and later optimise the code by reusing some of the registers R0-R7.

It used to be the case that a programmer writing a compiler would write a lexer, then the parser, then the code-generator – and do the same thing for another compiler later. This is hard work because many languages are very hard to parse, and the most successful algorithms are hard to write without careful study and lots of experience. More important, it duplicates a lot of the programming work, and makes it hard for other programmers to modify the code. Lexing and parsing are well- understood technologies now, so we have software that can do all of the work that compilers need in general, and that can accept high level programming which takes a very short time to do. The result is (usually) a compiler that has few bugs and is easily modified by other programmers who have used this software. One of the best known 'compiler generators' is YACC, which uses LEX to write the lexer and then generates a parser that produces target code as it parses.

LEX

We first describe how a LEX lexer works. A lexer has to recognise each token using a regular expression. Regular expressions are complex and you can read more about them in a book on LEX and YACC (such as *lex & yacc, by Levine, Mason, Brown, O'Reilly & Associates, 1992*). However, we describe a few simple ones here. The expression ab recognises the string "ab". If we want to recognise a string containing lots of repetitions of "ab", we write (ab)*, which accepts a string with any number of "ab"s in it, including none at all. If we use + instead of *, then the string must have at least 1 "ab" in order to be recognised.

Constructing a lexer is fairly straightforward. You first decide what token types you expect the lexer to recognise, then construct regular expressions that recognise them. The lexer will have a line for each expression, which ends with the token type you wish the lexer to return to the parser. The lexer automatically returns the token as well.

Let's consider a simple example. Suppose that we want a lexer for a small expression language that contains numbers, operators and parentheses. We need the following lines in the lexer:

```
[0..9]+     return NUM;
"+"         return PLUS;
"*"         return MUL;
"("         return LPAREN;
")"         return RPAREN;
```

YACC

We now describe how to use a YACC parser generator.

The parser generator contains a series of token types that must be specified in the first section. These are used to form a table that will be given to the lexer, so that it knows what token types are being used.

There are two tasks that are performed by the YACC parser generator. It parses the tokens returned to it, and it generates target code as it parses. We will write a little parser generator that produces the machine code given earlier. Before that, we need to provide you with a little background on parsing.

Parsers are generally written for a class of languages called the *context free* languages. A context free language is one that can be parsed by a context free grammar, a series of rules that specify the set of sentences in the language. If we think only of programming languages, then the set of sentences for a context free grammar is the set of syntactically correct programs.

Here is a grammar for the expression language we have described so far.

```
Start ::= Expression
Expression ::= PLUS Expr Expr
             | MUL Expr Expr

Expr ::= LPAREN Expression RPAREN
       | NUMBER
```

Each *rule* contains the name, followed by '::=', followed by the rule body. Rules are often called *productions*. For example, the first rule is the 'Start' rule, and the second is the 'Expression' rule. The rule body contains one or more clauses, separated by a vertical line. Each clause contains a sequence of words which may be either a token type or the name of a production.

Before looking at code generation, let's look at the parsing process. The parser attempts to match the stream of tokens against the productions. It starts with the Start production, and moves immediately to the Expression production. There, it has two possible clauses that may match. To select a clause, it looks at the token stream. If there is a PLUS token, then it accepts that token and selects the top clause; if there is a MUL token, it does the same for the bottom clause. If it succeeds in accepting all the tokens and it has finished the body of the Start production, then the expression is accepted, otherwise it is rejected by the parser.

We cheated a little by using a very simple grammar which allows the parser to select a rule by looking ahead at just one token. Many grammars require the parser to look ahead *k* tokens. Such a parser is known as an LR(k) parser, and that is close enough to what a YACC parser is for our purposes.

Now, we add code generation. The idea is very simple and easy to use: each time the parser recognises a clause, it has the opportunity to generate some code. Since LEX and YACC use C, we will write a statement in C that performs the code generation for each production in our grammar.

Before we write the C code, we have to consider what possible types of data the lexer could return. This is determined by a record structure declared at the top of the YACC code. We have decided to have a number only, which we will call 'num', so our record structure is

```
%union {
  int num;
}
```

The parser generator will need to generate register numbers (we won't worry about how many registers there are), so that will be done by the C function

```
Int registerNum = 0;
String getRegister () {return registerNum;}
Void nextRegister () {registerNum++;}
```

which appears in the last section of the YACC code.

```
Start ::= Expression

Expression ::= PLUS Expr Expr
                {
                    printf("ADD     %s,%s,%s",nextRegister(),$2,$3);
                    return getRegister();
                };
              | MUL Expr Expr
              {
                    printf("MUL     %s,%s,%s",nextRegister(),$2,$3);
                    return getRegister();
                };

Expr ::= LPAREN Expression RPAREN
                ;
        | NUMBER
                {
                    printf("LDVAL   %s,#%s",nextRegister(),$1.num);
                      return getRegister();
                };
```

Notice that two things are happening for each of the clauses that produce target code. An instruction is printed, and a register value is returned. That register value is the returned value of the clause (and therefore the rule), and is accessed using a $ sign. This is

possible because the parser generator builds an internal tree data structure as it parses, and by the time an entire clause has been recognised, all of the values required to do the code generation are present in that tree.

Let's see what happens when the parser generator looks at the following token stream:

```
MUL
NUMBER 3
LPAREN
PLUS
NUMBER 4
NUMBER 5
RPAREN
```

The first rule is the Start rule, which requires that it looks at the Expression rule. It then selects the MUL clause and parses it. This requires that the Expr production be used, and so the NUMBER clause is finished, causing the first line of target code to be emitted:

```
LDVAL       R0,#3
```

The other Expr in the MUL clause then gets parsed. There are two clauses to select from, and since LPAREN is the next token, the first is chosen. After LPAREN is accepted, the Expression production is then parsed, and the PLUS clause selected.
As the next two numbers are parsed, the code generated is

```
LDVAL       R1,#4
LDVAL       R2,#5
```

Then the PLUS clause is finished, generating the target code

```
ADD         R3,R1,R2
```

and the RPAREN token is accepted. Finally the original MUL clause is finished, generating the code

```
MUL         R4,R0,R3
```

6. The MuTaTeD! Compiler – pass 2 as an example of LEX and YACC

The second pass of the compiler for the MuTaTeD! Project has a more complex task. It must recognise an SGML language expressed by a DTD, and generate C code. The C code is composed of a sequence of calls to the NIFF SDK library, which produces the NIFF binary code. The SGML(NIFF) language is very close in structure to the NIFF language, so the code generation is straightforward.

The source language contains a sequence of tagged expressions. The parser must require that each beginning tag be matched with the end tag for most productions (some are more complex because an end tag isn't required but may appear). Then the parser must output a call to the appropriate NIFF SDK function.

6.1.The lexer for the second pass

The lexer contains a series of pairs of lines each of the form

"<tag>" return BEGIN_TAG
"\<tag>" return END_TAG

Each of these tokens represents the corresponding tag token.

6.2. The parser generator for the second pass

The parser will have productions of the form

Production ::= BEGIN_TAG p1 p2 … pn END_TAG

Roughly speaking, the associated C code will emit a call to the NIFF SDK that starts with TAG and contains arguments provided by $2..$n+1.

6.3. The lexer for the SGML(NIFF) pass

```
%{
/*
 * this is a lexer for sgml(NIFF).
 *
 */
```

The first section is for files that are included because they are useful C libraries:

```
#include <math.h>
#include "y.tab.h"

int printall = 0;
%}
```

This next section gets more space for the finite state machine that LEX generates because we are using a very simple but expensive machine by giving a long series of strings as the regular expressions

```
   /* need more space because lexer is working hard for us */

%e 10000   /* parse tree */
%p 20000   /* positions */
%n 10000   /* states */
%a 10000   /* packed transitions */

%%
```

These next lines contain the regular expressions that specify the tokens accepted by the LEX lexer.

```
[\\t ]+                  /* ignore whitespace */;

"<!DOCTYPE"[a-zA-Z\"\[\]\. ]+">"  ;

"<niff>"   {if (printall == 1) printf("NIFF\n"); return NIFF;}
"</niff>"  {if (printall == 1) printf("END_NIFF\n");
           return END_NIFF;}

"<setup>"   {if (printall == 1) printf("SETUP\n"); return SETUP;}
"</setup>"  {if (printall == 1) printf("END_SETUP\n");
           return END_SETUP;}

"<data>"   {if (printall == 1) printf("DATA\n"); return DATA;}
"</data>"  {if (printall == 1) printf("END_DATA\n");
           return END_DATA;}

"<info>"   {if (printall == 1) printf("INFO\n"); return INFO;}
"</info>"   {if (printall == 1) printf("END_INFO\n");
           return END_INFO;}
```

```
"<partsL>"   {if (printall == 1) printf("PARTSL\n"); return PARTSL;}
"</partsL>"  {if (printall == 1) printf("END_PARTSL\n");
           return END_PARTSL;}

"<part>"   {if (printall == 1) printf("PART\n"); return PART;}
"</part>"  {if (printall == 1) printf("END_PART\n");
           return END_PART;}

"<id>"   {if (printall == 1) printf("ID\n"); return ID;}

"<name>"   {if (printall == 1) printf("NAME\n"); return NAME;}

"<pageL>"   {if (printall == 1) printf("PAGEL\n"); return PAGEL;}
"</pageL>"  {if (printall == 1) printf("END_PAGEL\n");
           return END_PAGEL;}

"<systemL>"   {if (printall == 1) printf("SYSTEML\n");
           return SYSTEML;}
"</systemL>"  {if (printall == 1) printf("END_SYSTEML\n");
           return END_SYSTEML;}

"<partid>"   {if (printall == 1) printf("PARTID\n"); return PARTID;}
"</partid>"  {if (printall == 1) printf("END_PARTID\n");
           return END_PARTID;}

"<tupletds>"   {if (printall == 1) printf("TUPLETDS\n");
           return TUPLETDS;}
"</tupletds>"  {if (printall == 1) printf("END_TUPLETDS\n");
           return END_TUPLETDS;}

"<tag>"   {if (printall == 1) printf("TAG\n"); return TAG;}

"<staffL>"   {if (printall == 1) printf("STAFFL\n"); return STAFFL;}
"</staffL>"  {if (printall == 1) printf("END_STAFFL\n");
           return END_STAFFL;}

"<stafhedC>"   {if (printall == 1) printf("STAFHEDC\n");
           return STAFHEDC;}
"</stafhedC>"  {if (printall == 1) printf("END_STAFHEDC\n");
           return END_STAFHEDC;}

"<mayrepeat>"   {if (printall == 1) printf("REPEAT\n");
           return REPEAT;}
"</mayrepeat>"  {if (printall == 1) printf("END_REPEAT\n");
           return END_REPEAT;}

"<repeatnum>"   {if (printall == 1) printf("REPEATNUM\n");
           return REPEATNUM;}
"</repeatnum>"  {if (printall == 1) printf("END_REPEATNUM\n");
           return END_REPEATNUM;}

"<ts>"   {if (printall == 1) printf("TS\n"); return TS;}
"</ts>"  {if (printall == 1) printf("END_TS\n"); return END_TS;}
```

```
"<tstype>"   {if (printall == 1) printf("TSTYPE\n"); return TSTYPE;}
"</tstype>"  {if (printall == 1) printf("END_TSTYPE\n");
           return END_TSTYPE;}

"<top>"   {if (printall == 1) printf("TOP\n"); return TOP;}
"</top>"  {if (printall == 1) printf("END_TOP\n"); return END_TOP;}

"<bottom>"   {if (printall == 1) printf("BOTTOM\n"); return BOTTOM;}
"</bottom>"  {if (printall == 1) printf("END_BOTTOM\n");
           return END_BOTTOM;}

"<accident>"   {if (printall == 1) printf("ACCIDENT\n");
           return ACCIDENT;}
"</accident>"  {if (printall == 1) printf("END_ACCIDENT\n");
           return END_ACCIDENT;}

"<arpeggio>"   {if (printall == 1) printf("ARPEGGIO\n");
           return ARPEGGIO;}
"</arpeggio>"  {if (printall == 1) printf("END_ARPEGGIO\n");
           return END_ARPEGGIO;}

"<chordsym>"   {if (printall == 1) printf("CHORDSYM\n");
           return CHORDSYM;}
"</chordsym>"  {if (printall == 1) printf("END_CHORDSYM\n");
           return END_CHORDSYM;}

"<clef>"   {if (printall == 1) printf("CLEF\n"); return CLEF;}
"</clef>"  {if (printall == 1) printf("END_CLEF\n");
           return END_CLEF;}

"<type>"   {if (printall == 1) printf("TYPE\n"); return TYPE;}

"<octave>"   {if (printall == 1) printf("OCTAVE\n"); return OCTAVE;}
"</octave>"  {if (printall == 1) printf("END_OCTAVE\n");
           return END_OCTAVE;}

"<keysig>"   {if (printall == 1) printf("KEYSIG\n"); return KEYSIG;}
"</keysig>"  {if (printall == 1) printf("END_KEYSIG\n");
           return END_KEYSIG;}

"<lyric>"   {if (printall == 1) printf("LYRIC\n"); return LYRIC;}
"</lyric>"  {if (printall == 1) printf("END_LYRIC\n");
           return END_LYRIC;}

"<notehead>"   {if (printall == 1) printf("NOTEHEAD\n");
           return NOTEHEAD;}
"</notehead>"  {if (printall == 1) printf("END_NOTEHEAD\n");
           return END_NOTEHEAD;}

"<stafstep>"   {if (printall == 1) printf("STAFSTEP\n");
           return STAFSTEP;}

"<duration>"   {if (printall == 1) printf("DURATION\n");
           return DURATION;}
```

```
"<rest>"   {if (printall == 1) printf("REST\n"); return REST;}
"</rest>"  {if (printall == 1) printf("END_REST\n");
           return END_REST;}

"<stem>"   {if (printall == 1) printf("STEM\n"); return STEM;}
"</stem>"  {if (printall == 1) printf("END_STEM\n");
           return END_STEM;}

"<timesig>"   {if (printall == 1) printf("TIMESIG\n");
           return TIMESIG;}
"</timesig>"  {if (printall == 1) printf("END_TIMESIG\n");
           return END_TIMESIG;}

"<tuplet>"   {if (printall == 1) printf("TUPLET\n"); return TUPLET;}
"</tuplet>"  {if (printall == 1) printf("END_TUPLET\n");
           return END_TUPLET;}
```

The next regular expression matches white space

```
[\t ]+  ;
```

This matches integers

```
([0-9]+|-[0-9]+)  {yylval.num = atoi(yytext); if (printall == 1)
printf("INTEGER\n"); return INTEGER;}
```

This matches the version number

```
[0-9][a-zA-Z]  {if (printall == 1) printf("VERSION");return VERSION;}
```

This matches identifiers that start with a letter and then contains digits and/or letters

```
[a-zA-Z]([0-9]|[a-zA-Z])*   {yylval.str = yytext; if (printall == 1)
printf("IDENTIFIER\n"); return IDENTIFIER;}
```

This matches unexpected characters or a carriage return

```
.|\n    {ECHO; }
```

```
%%

yyerror(const char *msg)
{
printf("%d: %s at '%s'\n", yylineno, msg, yytext);
}
```

6.4. The parser generator for the SGML(NIFF) pass

This is the data structure that determines what values are returned along with the token type. They can either be integers (with the name ‘num’) or strings (with the name ‘str’).

```
%union {
        int num;
        char *str;
}
```

Here are the token declarations that are used to generate the table yytab.h, used by both the lexer and the parser generator.

```
/* here are the tokens */
%token <str> IDENTIFIER
%token <num> INTEGER
%token NIFF END_NIFF
       SETUP END_SETUP
       DATA END_DATA
       INFO END_INFO
       PARTSL END_PARTSL
       PART END_PART
       REPEAT END_REPEAT
       REPEATNUM END_REPEATNUM
       ID
       NAME
       PAGEL END_PAGEL
       SYSTEML END_SYSTEML
       PARTID END_PARTID
       TUPLETDS END_TUPLETDS
       TAG
       STAFFL END_STAFFL
       STAFHEDC END_STAFHEDC
       TS END_TS
       TSTYPE END_TSTYPE
       TOP END_TOP
       BOTTOM END_BOTTOM
       ACCIDENT END_ACCIDENT
       ARPEGGIO END_ARPEGGIO
       CHORDSYM END_CHORDSYM
       CLEF END_CLEF
       TYPE
       OCTAVE END_OCTAVE
       KEYSIG END_KEYSIG
       LYRIC END_LYRIC
       NOTEHEAD END_NOTEHEAD
       STAFSTEP
       DURATION
       REST END_REST
       STEM END_STEM
       TIMESIG END_TIMESIG
       TUPLET END_TUPLET
```

```
        VERSION
```

Here is the grammar for the SGML(NIFF) markup language. Recursive productions are productions that call themselves, either directly or indirectly.

```
%%

/* Grammar for SGML(NIFF) */

/*

Productions with start tags only (that appear in the example bach.niff,
since
the DTD is out of date):

id, name, tag, type, stafstep, duration

Productions that sometimes do not have end tags:

top, bottom

The others always have a start and end tag.

Recursive productions are marked prod_star or prod_plus.

*/

start : NIFF begin_niff setup data END_NIFF end_niff
      ;

/* use this when yacc goes into an infinite loop or does other clever
things!! must also
compile using -t flag. Look at y.output when reduce/reduce errors crop up
*/

debug: {yydebug = 1;}

begin_niff : {printStartFile();
              printf("if (! NIFFIOStartNiff())
doerror(\"NIFFIOStoreFormNiff failed\");\n");
             }
           ;
end_niff : {printf("NIFFIOEndNiff();\n");
            printEndFile();
           }
         ;


setup : SETUP begin_setup info partsL END_SETUP end_setup
      ;
begin_setup : {
               printf("if (! NIFFIOStartSetupSection())
doerror(\"NIFFIOStartSetupSection failed\");\n");
```

```
               printf("if (! NIFFIOStoreDefaultCLT())
doerror(\"NIFFIOStoreDefaultCLT failed\");\n");
              }
             ;
end_setup : {
              printf("if (! NIFFIOchunkStringTable()) doerror
(\"NIFFIOStringTable failed\");\n");
              printf("MyWriteStringTable();\n");
              printf("NIFFIOEndSetupSection();\n");
             }
           ;


info : INFO begin_info VERSION INTEGER END_INFO
     ;
begin_info :
 {printf("if (! NIFFIOchunkNiffInfo(version, progtypeNone, unitsNoUnits,
-1 ,-1)) doerror (\"NIFFIOStoreListSetupSection failed\");\n");}
           ;

partsL : PARTSL begin_partsL part_plus END_PARTSL end_partsL
       ;
begin_partsL : {printf("if (! NIFFIOStartParts()) doerror
(\"NIFFIOStartParts failed\");\n");}
             ;
end_partsL : {printf("NIFFIOEndParts();\n");}
           ;

part_plus : part
          | part part_plus
          ;

part : PART begin_part ID IDENTIFIER NAME IDENTIFIER END_PART
     ;
begin_part : {printf("if (! NIFFIOchunkPart(0,0,0,1,-1,-1,-1)) doerror
(\"NIFFIOchunkPart failed\");\n");}
           ;

data : DATA begin_data pageL_star END_DATA end_data
     ;
begin_data : {printf("if (! NIFFIOStartDataSection())
doerror(\"NIFFIOStartDataSection failed\");\n");}
           ;
end_data : {printf("NIFFIOEndDataSection();\n");}
         ;

pageL_star : pageL pageL_star
           | /* empty */
           ;

pageL : PAGEL begin_pageL plcont_star END_PAGEL end_pageL
      ;
begin_pageL : {printf("if (! NIFFIOStartPage()) doerror(\"NIFFIOStartPage
failed\");\n");}
```

```
            ;
end_pageL : {printf("NIFFIOEndPage();\n");}
          ;

plcont_star_alts : systemL | text

plcont_star : plcont_star_alts plcont_star
            | /* empty */
            ;

text : IDENTIFIER
     ;

systemL : SYSTEML begin_systemL slcont_star END_SYSTEML end_systemL
        ;
begin_systemL : {printf("if (! NIFFIOStartSystem())
doerror(\"NIFFIOStartSystem failed\");\n");}
              ;
end_systemL : {printf("NIFFIOEndSystem();\n");}
            ;

slcont_star_alts : staffL | text

slcont_star : slcont_star slcont_star_alts
            | /* empty */
            ;

anymusC : accident | arpeggio | chordsym | clef /* | figbass */
         | keysig /* | lyric */
         | notehead  | rest  /* | repeat  */
         | stem /* | tagactiv | tempomk */
         /* | text | tie */
         | timesig /* | tuplet*/
         ;

tagcon : partid /* | voiceid | altern | grace | mnid | number | ossia |
silent  */
         | tupletds
         ;

partid : PARTID begin_partid IDENTIFIER END_PARTID end_partid
       ;
begin_partid :
             ;
end_partid :
           ;

/* voiceid : INTEGER         ; */

/* altern : INTEGER       ; */

/* grace : duration       ; */

/* mnid : INTEGER     ; */
```

```
/* number : INTEGER       ; */

/* ossia : ; */

/* silent : ; */

tupletds : TUPLETDS transab transcd END_TUPLETDS
            { printf(">>>ERROR<<< TUPLETDS\n"); }
         ;

transab : duration
        ;

transcd : duration
        ;

alts_ts_anymusC : ts | anymusC
                ;

alt_ts_anymusC_star : alt_ts_anymusC_star alts_ts_anymusC  | /* empty */
                    ;

staffL : STAFFL begin_staffL stafhedC_stafcon_plus end_staffL END_STAFFL
       ;
begin_staffL : {printf("if (! NIFFIOStartStaff())
doerror(\"NIFFIOStartStaffL failed\");\n");}
             ;
end_staffL : {printf("NIFFIOEndStaff();\n");}
           ;

stafhedC_stafcon_plus : stafhedC alt_ts_anymusC_star
stafhedC_stafcon_plus
                      | stafhedC alt_ts_anymusC_star
                      ;

/*
note that stafhedC is supposed to be PCDATA in Mounce's DTD, but there is
the
following line in his bach example:

<stafhedC><tag><partid>p0</partid></stafhedC>

so it is obviously not PCDATA! Will try to reconstruct...
*/


stafhedC : STAFHEDC stafhedC_chunk stafhedC_next END_STAFHEDC
         ;

stafhedC_chunk :
    {printf("if (! NIFFIOchunkStaffHeader())
doerror(\"NIFFIOchunkStaffHeader failed\");\n");}
         ;
```

```
stafhedC_next : TAG tagcon
              | /* empty */
              ;

ts : TS TSTYPE INTEGER END_TSTYPE TOP INTEGER BOTTOM INTEGER END_TS
     {
      printf("rat.numerator = %d;\n",$6);
      printf("rat.denominator = %d;\n",$8);
      printf("if (! NIFFIOchunkTimeSlice(%d,rat))
doerror(\"NIFFIOchunkTimeSlice failed\");\n",$3);
     }
   ;


accident : ACCIDENT INTEGER END_ACCIDENT
     {
      printf("if (! NIFFIOchunkAccidental(%d))
doerror(\"NIFFIOchunkAccidental failure\");\n",
               $2);
     }
         ;

arpeggio : ARPEGGIO INTEGER END_ARPEGGIO
           { printf(">>>ERROR<<< arpeggio %d\n",$2); }
         ;

chordsym : CHORDSYM INTEGER END_CHORDSYM
           { printf(">>>ERROR<<< chordsym %d\n",$2); }
         ;

clef : CLEF TYPE clef_next OCTAVE INTEGER END_OCTAVE END_CLEF
     {
            printf("if (! NIFFIOchunkClef(%d,%d,%d))
doerror(\"NIFFIOchunkClef failure\");\n",
                   getClef(cleftype),getClefStep(cleftype),$5);
     }
    ;

clef_next : IDENTIFIER
     { strcpy(cleftype,$1);}

/* figbass : ; */

keysig : KEYSIG INTEGER END_KEYSIG
         { printf("if (! NIFFIOchunkKeySignature(%d))
doerror(\"NIFFIOchunkKeySignature failure\");\n",
                  $2);
         }
       ;


/* lyric : ; */
```

```
notehead : NOTEHEAD STAFSTEP INTEGER DURATION TOP INTEGER BOTTOM INTEGER
notehead_next END_NOTEHEAD
          {
            printf("rat.numerator = %d;\n",$6);
            printf("rat.denominator = %d;\n",$8);
            printf("if (! NIFFIOchunkNotehead(%d,%d,rat))
doerror(\"NIFFIOchunkNoteHead failed\");\n",
                           noteheadShape($6,$8),$3);
          }
         ;
notehead_next : TAG PARTID IDENTIFIER END_PARTID
              | /* empty */
              ;

duration : DURATION TOP INTEGER BOTTOM INTEGER
            { printf(">>>ERROR<<< duration\n"); }
         ;

/* repeat : ; */

rest : REST STAFSTEP INTEGER DURATION TOP INTEGER BOTTOM INTEGER END_REST
     {
       printf("rat.numerator = %d;\n",$6);
       printf("rat.denominator = %d;\n",$8);
       printf("if (! NIFFIOchunkRest(%d,%d,rat))
doerror(\"NIFFIOchunkRest failed\");\n",
                     restShape($6,$8),$3);
     }
     ;

/* something funny about this! But it appears in
Mounce's bach example, so let's do it this way
*/

stem : STEM END_STEM
       {printf("if (! NIFFIOchunkStem()) doerror(\"NIFFIOchunkStem
failed\");\n");}
     ;

/* tagactiv : ; */

/* tempomk : ; */

/* tie : ; */

timesig : TIMESIG TOP INTEGER END_TOP BOTTOM INTEGER END_BOTTOM
END_TIMESIG
        {
  printf("if (! NIFFIOchunkTimeSignature(%d,%d))
doerror(\"NIFFIOchunkTimeSignature failed\");\n",
           $3,$6);
        }
              ;
```

```
/* tuplet : ; */


%%

void printStartFile ()
{

  /* number of strings in table is first index */
char table [24] [100]
 = {"#include <stdio.h>\n",
    "#include <string.h>\n",
    "#include \"niffio.h\"\n",
    "#include \"stdcriff.h\"\n",
    "extern void doerror(const char *strMessage);\n",
    "void MyWriteStringTable(void);\n",
    "int\n",
    "main(int argc, char **argv)\n",
    "{\n",
    "unsigned char version[8] = \"6b\";\n",
    "FILE          *outfp;\n",
    "NIFFIOFile    *pnf;\n",
    "NIFFIOStorage   *pstore;\n",
    "NIFFIOChunkLengthTable *pclt;\n",
    "RIFFIOSuccess success;\n",
    "STROFFSET offset;\n",
    "RATIONAL rat;\n",
    "outfp = fopen(\"Test1.nif\", \"wb\");\n",
    "if (!outfp)\n",
    "doerror(\"Can't open Test1.nif for writing\");\n",
    "pstore = NIFFIOStorageNewSTDC(outfp);\n",
    "if (!pstore)\n",
    "doerror(\"Can't create new NIFFIOStorage\");\n"
   };



int i;
for (i = 0 ; i < 24 ; i++)
    printf("%s",table[i]);
}

void printEndFile ()
{

  /* number of strings is first index */
char table [23] [100]
 = {"NIFFIOStorageDelete(pstore);\n",
    "fclose(outfp);\n",
    "return 0;\n",
    "}\n",
    "void\n",
    "doerror(const char *strMessage)\n",
    "    {\n",
```

```
    "     fprintf(stderr, \"%s\\n\", strMessage);\n",
    "       exit(1);\n",
    "    }\n",
    "void MyWriteStringTable (void)\n",
    "{char strings[256][20];\n",
    "NIFFIOStbl myStbl[20];\n",
    "short i;\n",
    "int nWritten;\n",
    "strcpy(strings[0], \"piano\");\n",
    "strcpy(strings[1], \"p\");\n",
    "strcpy(strings[2], \"violin\");\n",
    "strcpy(strings[3], \"vl\");\n",
    "for (i=0; i < 4; i++) myStbl[i].str = strings[i];\n",
    "nWritten = NIFFIOStoreStbl(myStbl, 4);\n",
    "if (nWritten != 4)  {doerror(\"Couldn't write strings\");}\n",
    "}\n"
   };

int i;

for (i = 0 ; i < 23 ; i++)
    printf("%s",table[i]);
 }

int noteheadShape (int t, int b)
{
  if (t >= (b / 2)) return 3;
  else if (t >= b) return 2;
  else return 4;
  /* this will probably have to be fixed up! */
}

int restShape (int t, int b)
{
  if (b == 1) return 2;
  else if (b == 2) return 3;
  else if (b == 4) return 4;
  else if (b == 8) return 5;
  else if (b == 16) return 6;
  else if (b == 32) return 7;
  else if (b == 64) return 8;
  else return -1;
}



int getClef (char c[])
{
  if (!strcmp(c,"treble")) return 1;
  else if (!strcmp(c,"bass")) return 2;
  else if (!strcmp(c,"tenor")) return 3;
  else if (!strcmp(c,"percussion")) return 4;
  else if (!strcmp(c,"doubleG")) return 5;
  else if (!strcmp(c,"tab")) return 6;
```

```
  else return 0;
}

int getClefStep (char c[])
{
  if (!strcmp(c,"treble")) return 2;
  else if (!strcmp(c,"bass")) return 6;
  else if (!strcmp(c,"tenor")) return 4;
  else if (!strcmp(c,"percussion")) return 0;
  else if (!strcmp(c,"doubleG")) return 2;
  else if (!strcmp(c,"tab")) return 8;
  else return 0;
}

char cleftype[10];
```

6.5. The first pass of the compiler – LEX and YACC files

The LEX file

```
%{
/*
 * this is a lexer for SMDL.
 *
 */

#include <math.h>
#include "y.tab.h"

int printall = 0;

%}

   /* need more space because lexer is working hard for us */

%e 10000   /* parse tree */
%p 20000   /* positions */
%n 10000   /* states */
%a 10000   /* packed transitions */

%%

"<?HyTime "[!0-9a-zA-Z \.\/:\"=]*">"
                {if (printall) printf("Q_HYTIME\n");return Q_HYTIME ;}

"&"          {if (printall) printf("INCLUDE\n");return INCLUDE ;}

"c"             {if (printall) printf("C_NOTE\n");return C_NOTE;}

"d"             {if (printall) printf("D_NOTE\n");return D_NOTE;}

"e"             {if (printall) printf("E_NOTE\n");return E_NOTE;}

"f"             {if (printall) printf("F_NOTE\n");return F_NOTE;}

"g"             {if (printall) printf("G_NOTE\n");return G_NOTE;}

"a"             {if (printall) printf("A_NOTE\n");return A_NOTE;}

"b"             {if (printall) printf("B_NOTE\n");return B_NOTE;}

"id"           {if (printall) printf("ID\n");return ID ;}

"idr"           {if (printall) printf("IDR\n");return IDR ;}

"bibinfo"           {if (printall) printf("BIBINFO\n");return BIBINFO ;}

"exspec"           {if (printall) printf("EXSPEC\n");return EXSPEC ;}
```

```
"repeats"           {if (printall) printf("REPEATS\n");
                      return REPEATS ;}

"nominst"           {if (printall) printf("NOMINST\n");
                      return NOMINST ;}

"fictagam"         {if (printall) printf("FICTAGAM\n");
                      return FICTAGAM ;}

"pitchgam"         {if (printall) printf("PITCHGAM\n");
                      return PITCHGAM ;}

"dhalf"          {if (printall) printf("DHALF\n");return DHALF ;}

"dquarter"         {if (printall) printf("DQUARTER\n");
                      return DQUARTER ;}

"deighth"         {if (printall) printf("DEIGHTH\n");
                      return DEIGHTH ;}

"half"          {if (printall) printf("HALF\n");return HALF ;}

"quarter"         {if (printall) printf("QUARTER\n");
                      return QUARTER ;}

"eighth"         {if (printall) printf("EIGHTH\n");
                      return EIGHTH ;}

"sixteenth"         {if (printall) printf("SIXTEENTH\n");
                      return SIXTEENTH ;}

"whole"          {if (printall) printf("WHOLE\n");return WHOLE ;}

"<!DOCTYPE "      {if (printall) printf("E_DOCTYPE\n");
                      return E_DOCTYPE;}

"<!ENTITY "      {if (printall) printf("E_ENTITY\n");
                      return E_ENTITY;}

"<work>"             {if (printall) printf("WORK\n"); return WORK;}

"</work>"            {if (printall) printf("END_WORK\n");
                      return END_WORK;}

"<axis "[!0-9a-zA-Z \.\/:\"=]*">"  {if (printall) printf("AXIS\n");
                      return AXIS;}

"</axis>"       {if (printall) printf("END_AXIS\n");
                 return END_AXIS;}

"<cantus"[!0-9a-zA-Z \.\/:\"=]*">" {if (printall) printf("CANTUS\n");
                  return CANTUS;}
```

```
"</cantus>"      {if (printall) printf("END_CANTUS\n");
                 return END_CANTUS;}

"<tempobat"[!0-9a-zA-Z \.\/:\"=]*">"
           {if (printall) printf("TEMPOBAT\n"); return TEMPOBAT;}

"</tempobat>"    {if (printall) printf("END_TEMPBAT\n");
                 return END_TEMPOBAT;}

"<proscope"[!0-9a-zA-Z \.\/:\"=]*">"
           {if (printall) printf("PROSCOPE\n"); return PROSCOPE;}

"</proscope>"    {if (printall) printf("END_PROSCOPE\n");
                 return END_PROSCOPE;}

"<projectr"[!0-9a-zA-Z \.\/:\"=]*">"
           {if (printall) printf("PROJECTR\n"); return PROJECTR;}

"</projectr>"    {if (printall) printf("END_PROJECTR\n");
                 return END_PROJECTR;}

"<profun"[!0-9a-zA-Z \.\/:\"=]*">"
                 {if (printall) printf("PROFUN\n"); return PROFUN;}

"</profun>"      {if (printall) printf("END_PROFUN\n");
                      return END_PROFUN;}


"<extlist"[!0-9a-zA-Z \.\/:\"=]*">"
           {if (printall) printf("EXTLIST\n"); return EXTLIST;}

"</extlist>"      {if (printall) printf("END_EXTLIST\n");
                      return END_EXTLIST;}

"<extent"[!0-9a-zA-Z \.\/:\"=]*">"
           {if (printall) printf("EXTENT\n"); return EXTENT;}
"</extent>"
     {if (printall) printf("END_EXTENT\n"); return END_EXTENT;}

"<dimlist"[!0-9a-zA-Z \.\/:\"=]*">"
           {if (printall) printf("DIMLIST\n"); return DIMLIST;}

"</dimlist>"      {if (printall) printf("END_DIMLIST\n");
            return END_DIMLIST;}

"<dimspec"[!0-9a-zA-Z \.\/:\"=]*">"
           {if (printall) printf("DIMSPEC\n"); return DIMSPEC;}

"</dimspec>"      {if (printall) printf("END_DIMSPEC\n");
                 return END_DIMSPEC;}

"<marklist"[!0-9a-zA-Z \.\/:\"=]*">"
           {if (printall) printf("MARKLIST\n"); return MARKLIST;}
```

```
"</marklist>"     {if (printall) printf("END_MARKLIST\n");
                  return END_MARKLIST;}

"<stresuse"[!0-9a-zA-Z \.\/:\"=]*">"
           {if (printall) printf("STRESUSE\n"); return STRESUSE;}

"</stresuse>"     {if (printall) printf("END_STRESUSE\n");
                  return END_STRESUSE;}

"<thread"          {if (printall) printf("THREAD\n"); return THREAD;}
"</thread>"       {if (printall) printf("END_THREAD\n");
                  return END_THREAD;}

"<ces"             {if (printall) printf("CES\n"); return CES;}
"</ces>"          {if (printall) printf("END_CES\n");
                  return END_CES;}


"<rest"            {if (printall) printf("REST\n"); return REST;}
"</rest>"         {if (printall) printf("END_REST\n");
                  return END_REST;}

"<pitched"        {if (printall) printf("PITCHED\n");
                  return PITCHED;}
"</pitched>"      {if (printall) printf("END_PITCHED\n");
                  return END_PITCHED;}

"<nompitch>"         {if (printall) printf("NOMPITCH\n");
                  return NOMPITCH;}

"<fictadj>"         {if (printall) printf("FICTADJ\n");
                  return FICTADJ;}

"<nsdist>"        {if (printall) printf("NSDIST\n"); return NSDIST;}

"<gsdist>"        {if (printall) printf("GSDIST\n"); return GSDIST;}

"</fictadj>"         {if (printall) printf("END_FICTADJ\n");
                  return END_FICTADJ;}

"<gampitch>"         {if (printall) printf("GAMPITCH\n");
                  return GAMPITCH;}

"<octave"[!0-9a-zA-Z \.\/:\"=]*">"
                  {if (printall) printf("OCTAVE\n");return OCTAVE;}

"</octave>"         {if (printall) printf("END_OCTAVE\n");
                  return END_OCTAVE;}

"<pitchnm>"              {if (printall) printf("PITCHNM\n");
                  return PITCHNM;}

"<lyric"[!0-9a-zA-Z \.\/:\"=]*">"
                  {if (printall) printf("LYRIC\n");return LYRIC;}
```

```
"</lyric>"            {if (printall) printf("END_LYRIC\n");
                      return END_LYRIC;}

"<syllable"[!0-9a-zA-Z \.\/:\"=]*">"
          {if (printall) printf("SYLLABLE\n"); return SYLLABLE;}

"</syllable>"            {if (printall) printf("END_SYLLABLE\n");
                         return END_SYLLABLE;}

"<sylltext>"[a-zA-Z\,\'\- \.]+"</sylltext>"
          {if (printall) printf("TEXT %s\n",yytext); return TEXT;}

"<uniform>"[a-zA-Z\,\'\- \.]+"</uniform>"
     {if (printall) printf("UNIFORM %s\n",yytext); return UNIFORM;}
"</sylltext>"  {if (printall) printf("END_SYLLTEXT\n");
                return END_SYLLTEXT;}

"<bibinfo "[0-9a-zA-Z \=\"\:\/]*">"
          {if (printall) printf("BIBINFO\n"); return BIBINFO;}

"</bibinfo>"
     {if (printall) printf("END_BIBINFO\n"); return END_BIBINFO;}

"<author>"
     {if (printall) printf("AUTHOR\n"); return AUTHOR;}

"</author>"
     {if (printall) printf("END_AUTHOR\n");return END_AUTHOR;}

"<isbd>"
     {if (printall) printf("ISBD\n"); return ISBD;}

"</isbd>"
     {if (printall) printf("END_ISBD\n"); return END_ISBD;}

"<nameloc "[0-9a-zA-Z \=\"\:\/]*">"
     {if (printall) printf("NAMELOC\n"); return NAMELOC;}

"</nameloc>"
     {if (printall) printf("END_NAMELOC\n"); return END_NAMELOC;}

"<nmlist>"
     {if (printall) printf("NMLIST\n");return NMLIST;}

"</nmlist>"
     {if (printall) printf("END_NMLIST\n"); return END_NMLIST;}

"<score "[0-9a-zA-Z \=\"\:\/]*">"
     {if (printall) printf("SCORE\n"); return SCORE;}

"</score>"
     {if (printall) printf("END_SCORE\n");return END_SCORE;}
```

```
"="
     {if (printall) printf("EQUALS\n"); return EQUALS;}

"<!--"[ \(\)\,\'0-9a-zA-Z]*"-->"  ; /* SMDL comment! */

[a-zA-Z]([0-9]|[a-zA-Z])*
     {yylval.str = yytext;
     if (printall) printf("IDENTIFIER\n");return IDENTIFIER;}

([0-9]+|-[0-9]+)
     {yylval.num = atoi(yytext);
     if (printall) printf("INTEGER\n");return INTEGER;}

\"[a-zA-Z]*"."[a-zA-Z]*\"
     {if (printall) printf("QUOTED_ID\n");return QUOTED_ID;}

\"[\/: 0-9a-zA-Z]*\"
     {if (printall) printf("VERSION_ID\n");return VERSION_ID;}

"["              {if (printall) printf("LBRACK\n"); return LBRACK ;}

"]"              {if (printall) printf("RBRACK\n"); return RBRACK ;}

">"              {if (printall) printf("RANGLE\n"); return RANGLE ;}

[\t\n ]+                  /* ignore whitespace */;

.    {ECHO; }

%%
```

The YACC file

```
%union {

        int num;
        char *str;
}


        /* here are the tokens */
%token <str> IDENTIFIER
%token <num> INTEGER
%token DOCTYPE
       HYTIME
       ID IDR
       C_NOTE D_NOTE E_NOTE F_NOTE G_NOTE A_NOTE B_NOTE
       EXSPEC
       START
       REPEATS
       FICTAGAM
       PITCHGAM
       FICTADJ
       NSDIST
```

```
        GSDIST
        END_FICTADJ
        NOMINST
        VERSION_ID
        DHALF DQUARTER QUARTER EIGHTH HALF WHOLE DEIGHTH SIXTEENTH
        RANGLE LANGLE
        RBRACK LBRACK
        ENTITY
        QUOTED_ID
        DOCTYPE_ELT
        WORK END_WORK
        AXIS END_AXIS
        CANTUS END_CANTUS
        TEMPOBAT END_TEMPOBAT
        PROSCOPE END_PROSCOPE
        PROJECTR END_PROJECTR
        DIMLIST END_DIMLIST
        DIMSPEC END_DIMSPEC
        NMLIST
        END_NMLIST
        EXTLIST END_EXTLIST
        EXTENT END_EXTENT
        MARKLIST END_MARKLIST
        STRESUSE END_STRESUSE
        THREAD END_THREAD
        CES END_CES
        INCLUDE Q_DOCTYPE E_ENTITY E_DOCTYPE
        Q_HYTIME EQUALS
        REST END_REST
        PROFUN END_PROFUN
        PITCHED END_PITCHED
        NOMPITCH END_NOMPITCH
        GAMPITCH END_GAMPITCH
        OCTAVE END_OCTAVE
        PITCHNM END_PITCHNM
        LYRIC END_LYRIC
        SYLLABLE END_SYLLABLE
        SYLLTEXT END_SYLLTEXT
        BIBINFO END_BIBINFO
        TEXT
        AUTHOR END_AUTHOR
        UNIFORM END_UNIFORM
        ISBD END_ISBD
        NAMELOC END_NAMELOC
        SCORE END_SCORE

%%

 /* Grammar for SGML(NIFF) */

 /*
 Productions with start tags only (that appear in the example bach.niff,
since
 the DTD is out of date):
```

```
 id, name, tag, type, stafstep, duration

 Productions that sometimes do not have end tags:

 top, bottom

 The others always have a start and end tag.

 Recursive productions are marked prod_star or prod_plus.
 */

start : init hytime_star doctype work
      ;

init : {initialise();}

doctype : E_DOCTYPE IDENTIFIER IDENTIFIER QUOTED_ID
               LBRACK ent_star RBRACK RANGLE
           {
           spaces();
           printf("<!DOCTYPE niff SYSTEM \"intdtd.sgm\" []>\n");
           }

ent_star : E_ENTITY IDENTIFIER IDENTIFIER QUOTED_ID RANGLE ent_star
        | /* empty */
        ;

hytime_star : Q_HYTIME hytime_star
            | /* empty */
            ;

  /* when we figure out what perform and score should have in them, then
*/
  /* we can give the (cantus | perform | score) structure given by the
DTD */
  /* However, the DTD says that perform and score are ANY! */

work : WORK begin_work setup axis include_doctype_star cantus
            bibinfo nameloc score END_WORK end_work

     ;

begin_work : {
             spaces();
             printf("<niff>\n");
             }

setup : {
        spacesIn();
        printf("<setup>\n");

        spacesIn();
        printf("<info>6b 1</info>\n");
```

```
      spacesOut();
         printf("<partsL><part><id>p0<name>part0</part></partsL>\n");

         spacesOut();
         printf("</setup>\n");
         }

axis : AXIS END_AXIS

include_doctype_star : INCLUDE IDENTIFIER include_doctype_star
                     | /* empty */
                     ;

end_work : {
            spaces();
            printf("</niff>\n");
           }

/* this agrees with Mounce examples, but they do not agree with his DTD
*/

cantus : CANTUS begin_cantus
               cantus_choices_star END_CANTUS end_cantus
       ;

/*
acceptable pieces will have one treble line on the piano;
key signature (priority 1)
get in shape for someone else to follow on (documentation within source
code) (2)
(send Carola the document)
write your own SMDL pieces if you cannot get that working after a little
time
else get Mounce software working if possible (90 mins)
 */
begin_cantus : {
               spacesIn();
               printf("<data>\n");
               spacesIn();
               printf("<pageL>\n");
               spacesIn();

              }

end_cantus : {

               spacesOut();
               printf("</pageL>\n");

               spacesOut();
               printf("</data>\n");
             }
```

```
cantus_alts : thread | lyric | tempobat | stresuse | extlist

cantus_choices_star : cantus_alts cantus_choices_star
  /* this last line is required by Mounce examples but not by his DTD */
                      | /* empty */
                      ;

thread : THREAD thread_attributes RANGLE assign_firstCes
              music_events_star END_THREAD
       ;

thread_attributes : ID EQUALS IDENTIFIER thread_attributes
                  | NOMINST EQUALS IDENTIFIER thread_attributes
                  | /* empty */
                  ;

music_events_star : music_events music_events_star
                  | /* empty */
                  ;

music_events : /* tuplet  */
               ces
             | pitched
             | rest
               /* | chordchg */
             ;

assign_firstCes :
                  { firstCes = 1;
                  }

ces_events_star :  ces_events_star ces_events
                  | /* empty */
                ;


ces_events : /* tuplet  */
             pitched
             | rest
               /* | chordchg */
             ;


/* tuplet : ; */

/* chordchg : ; */

ces : CES ces_attributes RANGLE begin_ces ces_events_star END_CES
            end_ces
    ;

ces_attribute :  ID EQUALS IDENTIFIER
               |  PITCHGAM EQUALS IDENTIFIER
               |  REPEATS EQUALS INTEGER
```

```
                /*
                 {
                  spaces();
                  printf("<repeatnum>%d</repeatnum>\n",$3);
                 }
            */

              |  FICTAGAM EQUALS IDENTIFIER
                 {setFictagam($3);
                 }

ces_attributes : ces_attribute ces_attributes
               | /* empty */
               ;

begin_ces :
              {
              spacesIn();
              printf("<systemL>\n");
              spacesIn();
              printf("<staffL>\n");
              spaces();
              printf("<stafhedC></stafhedC>\n");
              spaces();
              printf("<ts><tstype>2</tstype><top>0<bottom>4</ts>\n");
              spaces();
              printf("<clef><type>treble<octave>0</octave></clef>\n");
              octave = 0;
              spaces();
              printf("<keysig>%d</keysig>\n",getFictagam());
              spaces();
              if (firstCes == 1)
                {

printf("<timesig><top>4</top><bottom>4</bottom></timesig>\n");
                firstCes = 0;
                }
              }

music_events_star : music_events_star music_events
                  | /* empty */
                  ;

end_ces :
             {
              spacesOut();
              printf("</staffL>\n");

              spacesOut();
              printf("</systemL>\n");

              }

rest : REST EXSPEC EQUALS setDuration RANGLE END_REST
```

```
       {
        spaces();
        /* we do not know what the staff step is, so it's on G for now */

printf("<rest><stafstep>3<duration><top>%d<bottom>%d</rest>\n",durationTo
p,durationBottom);
        if (topcount == bottomcount)
          {
           spaces();
              printf("<ts><tstype>1</tstype><top>%d<bottom>4</ts>\n",
                    ++barcount*4);
              topcount = 0;
              bottomcount = 1;
            }
        }
     ;


syllable_rest : REST EXSPEC EQUALS syllable_setDuration RANGLE END_REST
     ;

pitched_front : PITCHED EXSPEC EQUALS setDuration RANGLE
             NOMPITCH GAMPITCH maybe_setOctave
              PITCHNM setPitch
          {
        spaces();
          printf("<stem></stem>\n");
          spaces();
          convertPitch(pitch);

printf("<notehead><stafstep>%d<duration><top>%d<bottom>%d<tag><partid>p0<
/partid></notehead>\n",
                realPitch,durationTop,durationBottom);
          }
        ;

pitched : pitched_front maybe_fictadj END_PITCHED
          {
        spaces();

printf("<ts><tstype>2</tstype><top>%d<bottom>%d</ts>\n",topcount,bottomco
unt);
          /* note that more than one bar may have gone by, in which case
this should be a
             while loop - so this is a hack - but we do not know how ties
over bars are represented
             so it may not be a hack */
          if (topcount == bottomcount)
          {
           spaces();
              printf("<ts><tstype>1</tstype><top>%d<bottom>4</ts>\n",
                    ++barcount*4);
              topcount = 0;
              bottomcount = 1;
```

```
            }
          }
        ;

maybe_fictadj : FICTADJ NSDIST INTEGER GSDIST INTEGER END_FICTADJ
               {
                int accidental = 0;
                if ($5 == -1) accidental = 2;
                else if ($5 == 0) accidental = 3;
                else if ($5 == 1) accidental = 4;
                spaces();
                printf("<accident>%d</accident>\n",accidental);
               }
              | /* empty */
              ;

maybe_setOctave : OCTAVE INTEGER END_OCTAVE
                 { octave = $2;}
                | /* empty */
                ;


setDuration : DHALF
            {
             durationTop = 3;
             durationBottom = 4;
             updateCounts();
            }
            | DQUARTER
            {
             durationTop = 3;
             durationBottom = 8;
             updateCounts();
            }
            | DEIGHTH
            {
             durationTop = 3;
             durationBottom = 16;
             updateCounts();
            }
            | HALF
            {
             durationTop = 1;
             durationBottom = 2;
             updateCounts();
            }
            | WHOLE
            {
             durationTop = 1;
             durationBottom = 1;
             updateCounts();
            }
            | QUARTER
            {
```

```
             durationTop = 1;
             durationBottom = 4;
             updateCounts();
            }
            | EIGHTH
            {
             durationTop = 1;
             durationBottom = 8;
             updateCounts();
            }
            | SIXTEENTH
            {
             durationTop = 1;
             durationBottom = 16;
             updateCounts();
            }
            | /* empty */
            ;

syllable_setDuration : DHALF
            | DQUARTER
            | DEIGHTH
            | HALF
            | WHOLE
            | QUARTER
            | EIGHTH
            | SIXTEENTH
            ;

setPitch : C_NOTE
         { pitch = C; }
         | D_NOTE
         { pitch = D;}
         | E_NOTE
         { pitch = E;}
         | F_NOTE
         { pitch = F;}
         | G_NOTE
         { pitch = G;}
         | A_NOTE
         { pitch = A;}
         | B_NOTE
         { pitch = B;}

pitchnm : PITCHNM IDENTIFIER
        ;

tempobat : TEMPOBAT proscope END_TEMPOBAT
         ;

stresuse : STRESUSE END_STRESUSE
         ;

proscope : PROSCOPE projectr END_PROSCOPE
```

```
          ;

projectr : PROJECTR profun END_PROJECTR
          ;

profun : PROFUN IDENTIFIER END_PROFUN
       ;

extlist : EXTLIST extent END_EXTLIST
        ;

maybe_extlist : EXTLIST extent END_EXTLIST
              | /* empty */  ;

extent : EXTENT dimlist maybe_extent_end
       ;

maybe_extent_end : END_EXTENT
                 | /* empty */


dimlist : DIMLIST dimspec maybe_dimlist_end
        ;

maybe_dimlist_end : END_DIMLIST
                  | /* empty */

dimspec : DIMSPEC marklist_plus maybe_dimspec_end
        ;

maybe_dimspec_end : END_DIMSPEC
                  | /* empty */
                  ;

marklist_front : MARKLIST INTEGER maybe_marklist_end

marklist_plus : marklist_front marklist_plus
              | marklist_front
              ;

maybe_marklist_end : END_MARKLIST
                   | /* empty */
                   ;

lyric : LYRIC lyric_attributes syllable_list END_LYRIC
      ;

lyric_attributes : THREAD EQUALS IDENTIFIER lyric_attributes
                 | /* empty */
                 ;

syllable_alts : syllable_rest
              | syllable
```

```
syllable_list : syllable_alts syllable_list
              | syllable_alts
              ;

syllable : SYLLABLE syllable_attributes TEXT END_SYLLABLE
         ;

syllable_attributes : THREAD EQUALS IDENTIFIER syllable_attributes
                    | /* empty */
                    ;

bibinfo : BIBINFO author UNIFORM isbd END_BIBINFO
        ;

author : AUTHOR IDENTIFIER END_AUTHOR
       ;

isbd : ISBD IDENTIFIER END_ISBD
     ;

score : SCORE END_SCORE
      ;

nameloc : NAMELOC nmlist END_NAMELOC
        ;

nmlist : NMLIST IDENTIFIER END_NMLIST
       ;


%%

/* clef constants here */
const TREBLE = 0;
const NOT_A_CLEF = -1000;

/* pitch constants here */
const NOT_A_PITCH = -1000;

/* duration constants here */
const DURATION_TOP_ERROR = -1;
const DURATION_BOTTOM_ERROR = -1;
const NOT_A_DURATION = -1000;

/* octave constants here */
const NOT_AN_OCTAVE = -1000;

/* pitch constants */
const C = 0;
const D = 1;
const E = 2;
const F = 3;
const G = 4;
const A = 5;
```

```
const B = 6;

/* key constants */
const NONE = 0;
const FLATS1 = -1;
const FLATS2 = -2;
const FLATS3 = -3;
const FLATS4 = -4;
const FLATS5 = -5;
const FLATS6 = -6;
const SHARPS1 = 1;
const SHARPS2 = 2;
const SHARPS3 = 3;
const SHARPS4 = 4;
const SHARPS5 = 5;
const SHARPS6 = 6;
const NOT_A_FICTAGAM = -1000;



int topcount, bottomcount, barcount;
int octave;
int pitch;
int fictagam;
int clef;
int durationTop;
int durationBottom;
int sharp;
int flat;
int natural;
int firstCes;
int realPitch;

int spacenum;

void initialise()
{
octave = NOT_AN_OCTAVE;
pitch = NOT_A_PITCH;
fictagam = NONE;
clef = TREBLE;
durationTop = 0;
durationBottom = 1;
topcount = 0;
bottomcount = 1;
barcount = 0;
firstCes = 0;
realPitch = 0;
spacenum = 0;
}

void setFictagam(char *name)
{
if (!(strcmp(name,"none"))) fictagam = 0;
```

```
else if (!(strcmp(name,"flats1"))) fictagam = 8;
else if (!(strcmp(name,"flats2"))) fictagam = 9;
else if (!(strcmp(name,"flats3"))) fictagam = 10;
else if (!(strcmp(name,"flats4"))) fictagam = 11;
else if (!(strcmp(name,"flats5"))) fictagam = 12;
else if (!(strcmp(name,"flats6"))) fictagam = 13;
else if (!(strcmp(name,"flats7"))) fictagam = 14;
else if (!(strcmp(name,"sharps1"))) fictagam = 1;
else if (!(strcmp(name,"sharps2"))) fictagam = 2;
else if (!(strcmp(name,"sharps3"))) fictagam = 3;
else if (!(strcmp(name,"sharps4"))) fictagam = 4;
else if (!(strcmp(name,"sharps5"))) fictagam = 5;
else if (!(strcmp(name,"sharps6"))) fictagam = 6;
else if (!(strcmp(name,"sharps7"))) fictagam = 7;
else fictagam = NONE;
}

int getFictagam() {return fictagam;}

/* todo - make this stuff table driven */

void lookupTrebleClefOct0(int pitch)
{
 realPitch = NONE;
 if (pitch == C)  {realPitch = -2;}
 else if (pitch == D) {realPitch = -1;}
 else if (pitch == E) {realPitch = 0;}
 else if (pitch == F) {realPitch = 1;}
 else if (pitch == G) {realPitch = 2;}
 else if (pitch == A) {realPitch = 3;}
 else /* must be B */ {realPitch = 4;}
}

void lookupTrebleClefOct1(int pitch)
{
 realPitch = NONE;
 if (pitch == C) {realPitch = 5;}
 else if (pitch == D) {realPitch = 6;}
 else if (pitch == E) {realPitch = 7;}
 else if (pitch == F) {realPitch = 8;}
 else if (pitch == G) {realPitch = 9;}
 else if (pitch == A) {realPitch = 10;}
 else /* must be B */ {realPitch = 11;}
}

void lookupTrebleClefOct2(int pitch)
{
 realPitch = NONE;
 if (pitch == C) {realPitch = 12;}
 else if (pitch == D) {realPitch = 13;}
 else if (pitch == E) {realPitch = 14;}
 else if (pitch == F) {realPitch = 15;}
 else if (pitch == G) {realPitch = 16;}
 else if (pitch == A) {realPitch = 17;}
```

```
 else /* must be B */ {realPitch = 18;}
}



/* here we're assuming that there are only proper fractions */

int reduction()
{
int k;
int result = 1;
if (topcount > bottomcount) k = bottomcount; else k = topcount;
while (k > 1)
  {
   if ((topcount % k == 0) && (bottomcount % k == 0))
     { result = k; break;}
   else k--;
  }
return result;
}

void updateCounts()
{
int newbottom = 0;
int gcd = 0;

/* add two fractions - first normalise */
newbottom = bottomcount * durationBottom;
topcount = (topcount * durationBottom) + (durationTop * bottomcount);
bottomcount = newbottom;

gcd = reduction();

topcount = topcount / gcd;
bottomcount = bottomcount / gcd;

/* now deal with the current time sig. Since that is hardwired in for  */
/*   now, we can convert only for that - to be made more general later */

if (bottomcount < 4)
  {
   topcount = topcount * (4 / bottomcount);
   bottomcount = 4;
  }
}



void lookupTrebleClefTable(int oct, int pitch)
{
 if (oct == 0)
    lookupTrebleClefOct0(pitch);
 else
```

```
 if (oct == 1)
    lookupTrebleClefOct1(pitch);
 else
 if (oct == 2)
    lookupTrebleClefOct2(pitch);
 /* else check another octave when this gets more sophisticated */
}

void convertPitch (int pitch)
{
  /* printf("clef=%d; octave=%d; pitch=%d;
realPitch=%d;\n",clef,octave,pitch,realPitch); */
 if (clef == TREBLE)
   lookupTrebleClefTable(octave,pitch);
 /* else check another clef when this gets more sophisticated */
}

void spacesIn ()
{
int i;
spacenum = spacenum + 3;
for (i = 1; i <= spacenum; i++)
    printf(" ");
}

void spacesOut ()
{
int i;
for (i = 1; i <= spacenum; i++)
    printf(" ");
spacenum = spacenum - 3;
}

void spaces ()
{
int i;
for (i = 1; i <= spacenum; i++)
    printf(" ");
}
```

APPENDIX A: SMDL DTD

(by Steven Mounce)

```
<!-- SMDL DTD Authour: S.R.Mounce CANTATE Project -->

<!NOTATION virtime PUBLIC  -- Virtual Time Unit --
      "+//ISO/IEC 10744//NOTATION Virtual Measurement Unit//EN"
>

<!NOTATION SIsecond PUBLIC  -- Real Time --
      "+//ISO/IEC 10744//NOTATION Systeme International second//EN"
>

<!-- not actually part of standard: -->

<!ENTITY % tactvtu "80640">
<!-- The following parameter entity declarations could be used in
     conjunction with the above %tactvtu entity : -->

<!ENTITY % av.cxdm -- could use this as the value of the axisdim
                 attribute of an axis measured in music time.
                 (Maximum possible value given hyqcnt=32.) --
                "4294967295"
>
<!ENTITY % av.cxbg -- could use this as the value of the basegran
                 attribute of threads, batons, and wands.  --
                "vtu"
>
<!ENTITY % av.cxgh -- could use this as the value of the gran2hmu
                 attribute of threads, batons, and wands   --
                "1 1"
>
<!ENTITY % av.cxpg -- could use this as the value of the pls2gran
                 attribute of threads, batons, and wands   --
                "1 %tactvtu"
>

<!ENTITY % av.cxfm -- could use this as the value of the docmdu
                 attribute (of work), the fcsmdu attribute
                 (of cantus), and/or the axismdu attribute
                 (of axis).                                --
                "virtime 1 1"
>



<!ENTITY % SMDLrc -- SMDL resources; things that can appear anywhere --
            "chordgam | pitchgam | mudef | axis | fictagam |
              strestem |nameloc | desctab | extlist | bibinfo |
            text">
```

```
<!ELEMENT axis     -- Axis definition for a finite coordinate space --- O
ANY >

<!ATTLIST axis     HyTime   NAME     axis
              id       ID       #IMPLIED  -- Default: none --
              axismeas -- Measurement domain (SMU) of this axis --
                     -- lextype(NOTATION) --
                     CDATA    #FIXED virtime
              axismdu  -- SMU to MDU ratio that makes the MDU a
                        common denominator of HMUs for all
                            schedules on
                        this axis in all FCS elements that use it.--
                     -- lextype(frac) --
                     CDATA    #FIXED "virtime 1 1"
              axisdim  -- Dimension of this axis in MDUs --
                     -- lextype(number) --
                     CDATA    #FIXED "4294967295"
>

                 <!-- Description Table -->
<!element desctab  -- Descriptive text definition table --
              - O      (desctxt, descdef)+ >
<!attlist desctab  HyTime   NAME     desctab
              id       ID       #REQUIRED >

                 <!-- Descriptive Text -->
<!element desctxt  -- Descriptive text --
              O O      (#PCDATA) -- lextype(words) -->
<!attlist desctxt  HyTime   NAME     desctxt >

            <!-- Descriptive Text Definition -->
<!element descdef  -- Descriptive text definition for exlist --
              O O      (extent) >
<!attlist descdef  HyTime   NAME     descdef >

<!element text - o (#PCDATA) >


<!-- Work Segment Content Model -->
           <!-- use: work, workseg -->
<!ENTITY % m.wksg
           "(workseg+ | (cantus | perform | score)*)"
>

<!ENTITY % bib.att
     "bibinfo       IDREFS  #IMPLIED
      theme         IDREFS  #IMPLIED"
>


<!ELEMENT work
           - O (%m.wksg;) +(%SMDLrc;) >
<!ATTLIST work
           HyTime    NAME    #FIXED HyDoc
```

```
           SMDL      NAME    work
           id       ID      #IMPLIED
           docmdu    -- SMU to MDU ratio that makes the MDU a
                     common denominator of HMUs for all schedules in
                     all FCS that are in a given measurement domain.
                     Constraint: SMU name and ratio for all of the
                     measurement domains used in the document.
                     lextype((NAME,s+,frac),(s+,NAME,s+,frac)*) --
                  CDATA    #FIXED "virspace 1 1" -- Default: "1 1" for
each SMU --
           %bib.att -- bibinfo attributes --
>



<!-- authority attributes -->

<!ENTITY % auth.at
     "authorty   -- reference(s) to the source document(s) or
                 authority(ies) from which this SMDL
                 representation was derived. --
           IDREFS   #IMPLIED"
>

<!-- e.g. reference to NIFF doc -->

                <!-- Bibliographic Information Attributes -->




                <!-- Work Segment -->
<!ELEMENT workseg  -- segment of musical work. --
           - O (%m.wksg;) >
<!ATTLIST workseg
     SMDL    NAME    workseg
     id      ID      #IMPLIED
     class   CDATA   #IMPLIED -- e.g. movement, section, musical number.
                         Default: not specified. --
     endmod  CDATA   "pause"  -- e.g. "15 min. intermission",
                         "await curtain", etc. --

>


<!ELEMENT cantus - O (thread | lyric | tempobat | stresuse)*>
<!ATTLIST cantus
     HyTime          NAME            fcs
     SMDL            NAME            cantus
     %bib.att       -- bibinfo attributes --
     %auth.at      -- authorty attributes --
     norm            -- Normalization according to the
```

```
                    named algorithm
                 (declared as a notation) has been applied. --
              NOTATION       (normalgs) #IMPLIED  -- Default:
                                          not normalized --
              -- HyTime fcs attributes --
     id            ID              #REQUIRED
     fcsname       -- Name of semantic FCS described
                         by the element --
              NAME            #IMPLIED  -- Default: FCS
                                        is unnamed --
     fcsmdu        -- SMU to MDU ratio that makes the MDU a
              common denominator of HMUs for all schedules on
              all axes of FCS in a given measurement domain.--
              -- Constraint: SMU name and ratio for one or
                 more of the domains used in the FCS.  An SMU
                 name can occur only once. --
              -- lextype((NAME,s+,frac),(s+,NAME,s+,frac)*) --
              CDATA           #FIXED "" -- Default:
                                      equal to docmdu --
     axisdefs      -- Definitions of FCS axes --
              -- Constraint: GIs of axis element types --
              -- lextype(GIL) --
              NAMES           #FIXED          virtime

>


<!ENTITY % e.music  -- musical events and event groups --
      "tuplet | ces | pitched | rest | chordchg">

<!-- note that the event element (from HyTime) allows all manner of
     multimedia event types to be added -->

<!ELEMENT thread   -- schedule of music events and event groups. --
             - O (%e.music;)*>
<!ATTLIST thread
      HyTime          NAME            evsched
      SMDL            NAME            thread
      id             ID              #IMPLIED  -- Default: none --
      axisord       -- Order of axes in schedule --
              -- Constraint: GIs of axis element types. Omitted
                 GI means all events in schedule fully occupy
                 the omitted axis --
              -- lextype(GIL) --
              CDATA           #FIXED "" -- Default:
                                        axisdefs in FCS --
      apporder      -- Order of schedule elements is significant to
                 the application and must be preserved --
              (order|disorder) order
      sorted        -- Representation of schedule elements
                         is sorted
                 by order of position on axes of schedule --
              (sorted|unsorted) unsorted
```

```
                -- HyTime schdmeas attributes --
       basegran       -- Base granule for each axis --
                -- lextype(words) --
                CDATA           #FIXED "vtu" -- Default: SMU
                                        for each --
       gran2hmu       -- Granule to HMU ratio for each axis --
                -- Constraint: 1 ratio per axis, or 1 for all --
                -- lextype(fracs) --
                NUMBERS         #FIXED "1 1"
                -- HyTime overrun attributes --
       overrun        -- Handling of dimension that overruns range --
                (error|wrap|trunc|ignore) error
                -- pls2gran attributes --
       pls2gran       -- Pulse to granule ratio for each axis --
                -- Constraint: 1 ratio per axis, or 1 for all --
                -- lextype(fracs) --
                NUMBERS         "1 80640"
       nominst        -- nominal instrument(s) or other performing
                     resource(s) allocated to this thread.    --
                CDATA           #IMPLIED    -- Default:
                                      not specified --


>
                     <!-- Tuplet -->
<!ELEMENT tuplet
             - O (%e.music;)>
<!ATTLIST tuplet
      HyTime            NAME            evgrp
      SMDL              NAME            tuplet
      id               ID              #IMPLIED  -- Default: none --
      grpscope         -- Group scope (nominal extent of group) --
                -- For each axis, the dimension is the lowest first
                   quantum of any group member through the highest
                   last quantum of any group member on that axis.
                   All of a member's extents are included. --
                -- reftype(extent) --
                IDREF           #IMPLIED  -- Default:
                                      to be calculated --
      grpdex           -- Group derived extent specification --
                -- Use for resizing, rearrangement, repetition.
                   Group members are given the derived extents.
                   If nominal extent is also wanted, it must be
                   specified as one of the derived extents. --
                -- Constraint: multiple references concatenated --
                -- reftype(extlist) --
                IDREFS          #IMPLIED  -- Default:
                                      no derivation --
                    -- pls2gran attributes --
      pls2gran         -- Pulse to granule ratio for each axis --
                -- Constraint: 1 ratio per axis, or 1 for all --
                -- lextype(fracs) --
                NUMBERS         "1 1"
```

```
>

                <!-- Cantus Event Sequence -->
<!ELEMENT ces
           - O (%e.music;)*
>

<!ATTLIST ces
      SMDL            NAME            ces
      id              ID              #IMPLIED    -- Default: none --
      ornstyle        -- Ornamentation style: e.g. , period --
                CDATA           #IMPLIED
      pitchgam        -- Current pitch gamut --
                IDREF           #IMPLIED    -- Default: value of
                                       syntactically previous
                                       pitchgam attribute. --
      fictagam        -- Current ficta gamut --
                IDREF           #IMPLIED    -- Default: value of
                                       syntactically previous

                                  fictagam attribute. --
      mudef           -- governing microtuning unit definition --
                IDREF           #IMPLIED    -- Default: value of
                                       syntactically previous
                                       mudef attribute. --
      divisi          -- either allocate simultaneous notes
                            among performing
                   resources, or make everybody play everything. --
                (tutti | divisi)        tutti
                            -- default: everybody
                                          plays everything. --
      arpeggio        -- Either arpeggiate or don't arpeggiate. Applies
only
                   to the onset of the events scheduled to start
                   together, and only when that starting
                      time is the
                   earliest scheduled start time in the entire ces. --
                (rollup | rolldown | noroll)    noroll
                                -- default: no arpeggiation. --
      grace           -- are the contained events grace notes? --
                (grace | nograce)       nograce   -- default: these are
                                             not grace notes. --
      choice          -- Is only one subelement, chosen by the performer,
to
                   be played at any one time, or are all of them to be
                   played? (attribute for handling ossia, 1st/2nd ending
                   etc. --
                (all | one)             all
      choicrit        -- Criteria by which choice should be made. --
                CDATA           #IMPLIED    -- default: no choice needs
                                       to be made (choice=all) --
      conloc          -- ID of schedule, evgrp, or event to be used in
place
                   of, or as content of, this element (see HyTime
```

```
                  International Standard). --
               IDREF          #CONREF
     repeats         -- Number of repetitions beginning at end of last
                  quantum occupied. lextype (unzi). --
               CDATA          1
>

<!element dimspec  -- Dimension specification --
             -- Constraint: Content must resolve to a pair of markers
                (unless marker1 omitted for implied dimref) --
             O O   (marklist, marklist?) >
<!attlist dimspec  HyTime   NAME     dimspec
             id       ID       #IMPLIED  -- Default: none --
>

<!element marklist -- List of axis markers or their equivalents --
             -- Constraint: nested marklists merge into single list --
             -- Constraint: calspec permitted only in calendars --
             O O      (#PCDATA) -- lextype(snzi*) -->
<!attlist marklist HyTime   NAME     marklist
>

<!element dimlist  -- Constraint: derives dimspecs from marker pairs.
                Nested marklist/dimlist merge into single list with
                effect of "((marker1,marker2)|dimspec)*" --
             -- Constraint: calspec permitted only in calendars --
             O O      (dimlist|dimspec
                   |#PCDATA)* -- lextype(snzi*) -->
<!attlist dimlist  HyTime   NAME     dimlist
>


<!element extent   -- Scheduled extent --
             -- Constraint: multiple dimlists concatenated into one
list,
                which must contain one dimspec per axis --
             O O      (dimlist+) >
<!attlist extent   HyTime   NAME     extent
             id       ID       #IMPLIED  -- Default: none --

>


<!element extlist  -- Scheduled extent list --
             -- Constraint: derives dimspecs from marker pairs and
                extents from dimspec sequences (one per axis) and
                interspersed syntactic extents --
             -- Constraint: nested marklist/dimlist/extlist merge into
                single list with effect of
                "(((marker1,marker2)|dimspec)+|extent)+" --
             -- Constraint: calspec permitted only in calendars --
             O O     (extlist|extent|dimlist|dimspec
                   |#PCDATA)* -- lextype(snzi*) -->
<!attlist extlist  HyTime   NAME     extlist
```

```
              id       ID       #IMPLIED  -- Default: none --
              -- added attlist for all-desc below --
              desctxt  -- Descriptive text --
                     -- If specified, its descdef in desctab is treated
                        by HyTime as the element or as its content. --
                     -- lextype(words) --
                     CDATA    #CONREF   -- Default: none --
              desctab  -- Current description tables --
                     -- Constraint: searched in order listed --
                     -- reftype(desctab) --
                     IDREFS   #CURRENT

>



                <!-- Rest -->
<!ELEMENT rest
           - O (extlist?)>
<!ATTLIST rest
     HyTime            NAME            event
     SMDL              NAME            rest
     id              ID              #IMPLIED
     exrecon         IDREF           #IMPLIED
     exspec          IDREFS          #REQUIRED
     pls2gran        NUMBERS         "1 1"
     conloc          IDREF           #CONREF
>




                <!-- Note Event -->
<!ELEMENT pitched
               -- A musically-pitched note. Used in: %e.music; --
               - O (extlist?, nompitch?, vary?)
>

<!ATTLIST pitched
     HyTime            NAME            event
     SMDL              NAME            pitched
     id              ID              #IMPLIED
     exrecon         IDREF           #IMPLIED
     exspec          IDREFS          #REQUIRED
     pls2gran        NUMBERS         "1 1"
     conloc          IDREF           #CONREF
     detune          -- Signed number of MUs to add or subtract --
               -- Default: play in tune (0). lextype(snum).  --
               NMTOKEN         0
     tie             -- Immediate successor to which this event is tied
--
               IDREF           #IMPLIED        -- Default :
```

```
                                        not tied --
>

<!ELEMENT vary          -- Variation in pitch value of note, expressed as
                   a formula. --
                  O O (#PCDATA)
>

<!ATTLIST vary
      HyTime            NAME            sHyTime
      SMDL              NAME            vary
      -- notation        Data content notation. (Actual notation names
                   application-specific). (attribute defined by
                   HyTime)
                  NOTATION          #REQUIRED --
      conloc          -- Formula source (if not in content).  --
                  IDREF           #CONREF
>


                  <!-- Chord change Event -->
<!ELEMENT chordchg      -- Chord change event --
                  - O ( extlist?, (ccschord | figbass)?)?
>

<!ATTLIST chordchg
      HyTime            NAME            event
      SMDL              NAME            chordchg
      id              ID              #IMPLIED
      exrecon         IDREF           #IMPLIED
      exspec          IDREFS          #REQUIRED
      pls2gran        NUMBERS         "1 1"
      conloc          IDREF           #CONREF
      detune          -- Signed number of MUs to add or subtract --
                  -- Default: play in tune (0). lextype(snum).  --
                  NMTOKEN         0
      tie             -- Immediate successor to which this event is tied
--
                  IDREF           #IMPLIED        -- Default :
                                        not tied --
>

                  <!-- Lyric -->
<!ELEMENT lyric         -- Time-ordered sequence of sung syllables.
                   Synchronized with a thread.          --
                  - O (syllable | rest)+
>

<!ATTLIST lyric
      HyTime            NAME            evsched
      SMDL              NAME            lyric
      thread          -- Notes to which lyric is sung --
                  IDREF           #REQUIRED
                  -- HyTime sched attributes  --
```

```
      axisord         -- Order of axes in schedule --
                   -- Constraint: GIs of axis element types. Omitted
                      GI means all events in schedule fully occupy
                      the omitted axis --
                   -- lextype(GIL) --
                   CDATA          #FIXED "" -- Default: axisdefs in FCS --
       apporder       -- Order of schedule elements is significant to
                      the application and must be preserved --
                   (order|disorder) order
       sorted         -- Representation of schedule elements is sorted
                      by order of position on axes of schedule --
                   (sorted|unsorted) unsorted
                   -- HyTime schdmeas attributes --
       basegran       -- Base granule for each axis --
                   -- lextype(words) --
                   CDATA          #FIXED "vtu" -- Default: SMU for each --
       gran2hmu       -- Granule to HMU ratio for each axis --
                   -- Constraint: 1 ratio per axis, or 1 for all --
                   -- lextype(fracs) --
                   NUMBERS        #FIXED "1 1"
                   -- HyTime overrun attributes --
       overrun        -- Handling of dimension that overruns range --
                   (error|wrap|trunc|ignore) error
                   -- pls2gran attributes --
       pls2gran       -- Pulse to granule ratio for each axis --
                   -- Constraint: 1 ratio per axis, or 1 for all --
                   -- lextype(fracs) --
                   NUMBERS        "1 80640"


>

                   <!-- Sung Syllable -->
<!ELEMENT syllable      -- sung syllable  --
                   - O ( extlist?, sylltext?)?
>
<!ATTLIST syllable
      HyTime           NAME           event
      SMDL             NAME           syllable
      id               ID             #IMPLIED
      exrecon          IDREF          #IMPLIED
      exspec           IDREFS         #REQUIRED
      pls2gran         NUMBERS        "1 1"
      conloc           IDREF          #CONREF
>

<!ELEMENT sylltext      -- Syllable text  --
                   - O ( #PCDATA)
>
<!ATTLIST sylltext
      SMDL             NAME           sylltext
>

                   <!-- Stress Pattern Template -->
```

```
<!ELEMENT strestem
                -- Beat cycle definition; dynamic stress pattern.
                   Pointnums in ascending order. --
                - O             (pointnum, stresval*)*
>
<!ATTLIST strestem
     SMDL            NAME            strestem
     id              ID              #IMPLIED
     pointcnt        -- Number of stress points in pattern. --
                NUMBER          #REQUIRED
>

                <!-- Stress Point Number -->
<!ELEMENT pointnum
                -- Stress point receiving agogic or dynamic
                   stresses. Integer from 1 to pointcnt --
                O O             (#PCDATA)
>
<!ATTLIST pointnum
     SMDL            NAME            pointnum
>

                <!-- Stress Type Values -->
<!ELEMENT stresval
                -- Stresses applied to specified stress point --
                - O             (strestxt?, formula?)
>
<!ATTLIST stresval
     SMDL            NAME            stresval
     stresuse        -- ID of stresuse for nested stress pattern. --
                IDREF           #CONREF   -- Default: no nested
                                     pattern. --
>

<!ELEMENT strestxt
                -- Imprecise stress specification: loud, downbeat--
                - O             (#PCDATA)
>
<!ATTLIST strestxt
     SMDL            NAME            strestxt
>

                <!-- Stress Pattern Use -->
<!ELEMENT stresuse
                -- Use of stress pattern in tempo or outer stress
                   pattern. Default: entire tempo of tempo
                   directive. --
                - O             (dimspec | stresdur)?
>
<!ATTLIST stresuse
     SMDL            NAME            stresuse
     id              ID              #REQUIRED
     idr             -- ID of stress pattern --
                IDREF           #REQUIRED
```

```
      stresptr          -- Stress pt # of pattern on which use begins. Not
                           1 only if anacrusis. --
                     NUMBER          1
>

<!ELEMENT stresdur
      -- Duration of one iteration of nested stress pattern.
         Format: NUMBER of points. --
                     - O (#PCDATA)
>

<!ATTLIST stresdur
      SMDL             NAME             stresdur
>



<!ELEMENT tempobat     -- Baton --
                 - O       (proscope+) >
<!ATTLIST tempobat     HyTime    NAME      baton
                        id         ID        #REQUIRED
                     -- HyTime sched attributes  --
      axisord           -- Order of axes in schedule --
                     -- Constraint: GIs of axis element types. Omitted
                        GI means all events in schedule fully occupy
                        the omitted axis --
                     -- lextype(GIL) --
                     CDATA            #FIXED "" -- Default: axisdefs in FCS --
       apporder        -- Order of schedule elements is significant to
                        the application and must be preserved --
                     (order|disorder) order
       sorted          -- Representation of schedule elements is sorted
                        by order of position on axes of schedule --
                     (sorted|unsorted) unsorted
                     -- HyTime schdmeas attributes --
       basegran        -- Base granule for each axis --
                     -- lextype(words) --
                     CDATA            #FIXED "vtu" -- Default: SMU for each --
       gran2hmu        -- Granule to HMU ratio for each axis --
                     -- Constraint: 1 ratio per axis, or 1 for all --
                     -- lextype(fracs) --
                     NUMBERS          #FIXED "1 1"
                     -- HyTime overrun attributes --
       overrun         -- Handling of dimension that overruns range --
                     (error|wrap|trunc|ignore) error
                     -- pls2gran attributes --
       pls2gran        -- Pulse to granule ratio for each axis --
                     -- Constraint: 1 ratio per axis, or 1 for all --
                     -- lextype(fracs) --
                     NUMBERS          "1 80640"


>
```

```
<!ELEMENT proscope -- Projector scope --
              -- Defines the scope of an event projector
                 as an extent of the unprojected schedule --
              - O      (projectr) >
<!ATTLIST proscope HyTime   NAME     proscope
              id       ID       #IMPLIED  -- Default: none --
                    -- exspec attributes --
              exspec   -- Extent specification --
                    -- Constraint: multiple references concatenated --
                    -- reftype(extlist) --
                    IDREFS   #REQUIRED

>


<!ELEMENT projectr -- Event projector --
              -- Defines extents in the projected schedule into which
                 events within the projector scope will be projected --
              -- Constraint: extlists are concatenated into one list --
              - O      (extlist+|profun|scaleref) >

<!ATTLIST projectr HyTime   NAME     projectr
              id       ID       #IMPLIED  -- Default: none --
              strict   -- Strictness of interpretation: e.g., rubato --
                    -- lextype(words) --
                    CDATA    #IMPLIED  -- Default: strict projection --
>


<!ELEMENT profun   -- Projection function --
              - O      (#PCDATA)* >
<!-- was %HyBrid-->
<!ATTLIST profun   HyTime   NAME     profun >


<!ELEMENT scaleref -- Scaling reference --
              -- Instantaneous projection scaling at specified quantum
                 in each axis of aligned original or projected schedule -
-
              -- Constraint: one dimref per axis of baton --
              - O      (dimref+) >
<!ATTLIST scaleref HyTime   NAME     scaleref >



                <!-- PITCH MODEL -->

                <!-- Nominal Pitch -->
<!ELEMENT nompitch
                O O ( gampitch | freqspec | jipit)
>

<!ATTLIST nompitch
     SMDL            NAME            nompitch
```

```
>

                <!-- Gamut-based Pitch -->
<!ELEMENT gampitch -- Named pitch from gamut definition --
              O O      (octave?, pitchnm, fictadj?)
>

<!ATTLIST gampitch
     SMDL            NAME            gampitch
>

                <!-- Octave Offset -->
<!ELEMENT octave   -- Octave offset from named pitch
                 Signed integer: 0 means pitch is in definition
                 octave. Default: no change from octave of
                 syntactically previous pitch.
                 lextype (snum). --
              O O      (#PCDATA)
>

<!ATTLIST octave
     SMDL            NAME            octave
>

                <!-- Fictum Adjustment -->
<!ELEMENT fictadj  -- Fictum (#/b) adjustment from pitchnm: signed
                 integer. Default: no adjustment --
              O O      (gamintvl)
>

<!ATTLIST fictadj
     SMDL            NAME            fictadj
>

                <!-- Frequency Specification -->
<!ELEMENT freqspec -- Frequency specification --
              O O      (hertz | formula)
>

<!ATTLIST freqspec
     SMDL            NAME            freqspec
>

<!ELEMENT hertz    -- Absolute frequency in Hertz (decimal fraction) --
              O O      (#PCDATA) >

<!ATTLIST hertz
     SMDL            NAME            hertz
>


                <!-- Pitch Gamut Definition -->
<!ELEMENT pitchgam -- Pitch gamut definition. If genfreq omitted,
                 no default frequencies generated --
```

```
              - O      (genfreq?, namestep+)
>

<!ENTITY % FRAC    "NUMBERS" -- fraction: numerator, denominator? (=1) --
>
<!ATTLIST pitchgam
      SMDL            NAME            pitchgam
      id              ID              #REQUIRED
      gamutdes        -- Description of gamut --
                 CDATA           #IMPLIED -- Default: undescribed --
      highstep        -- Highest gamut step number: positive integer --
                 NUMBER          #IMPLIED -- Default:
                                      highest gamstep --
      octratio        -- Octave ratio ("1 1" = no octaves) --
                 %FRAC; "2 1"      -- Default: normal octaves --
>

<!-- the formula for an even tempered scale is:
     pitch [n+1] = pitch [n] * ((highstep+1) root of (octave ratio)) -->

                 <!-- Generated Frequency Base -->
<!ELEMENT genfreq  -- Generated frequency base --
              O O      (gamstep, freqspec)
>

<!ATTLIST genfreq
      SMDL            NAME            genfreq
>

                 <!-- Name Step Group -->
<!ELEMENT namestep -- Name step group: pitches based on same name
                  (C, C#, C##). In conventional music: 7
                  namesteps (letter names A-G) --
              - O      (pitchdef+)
>

<!ATTLIST namestep
      SMDL            NAME            namestep
>


                 <!-- Pitch Definition -->
<!ELEMENT pitchdef -- Pitch definition (member of a pitch gamut)
                  If gamstep omitted: next sequential (0 if first)
                  Freqspec overrides generated frequency
                  for gamstep --
              - O      (pitchnm, gamstep?, (freqspec | relfreq)?)
>

<!ATTLIST pitchdef
      SMDL            NAME            pitchdef
>

<!ELEMENT gamstep  -- Gamut step: Non-negative integer (>= 0)
```

```
                Conventional music: 12 gamut steps
                (chromatic scale) --
             O O      (#PCDATA)
>

<!ATTLIST gamstep
     SMDL            NAME            gamstep
>

<!ELEMENT relfreq  -- Relative frequency with respect to another pitch --
             - O      (pitchnm, iratio)
>

<!ATTLIST relfreq
     SMDL            NAME            relfreq
>

                <!-- Pitch Name -->
<!ELEMENT pitchnm  -- Name of pitch in a gamut --
             O O      (#PCDATA) >

<!ATTLIST pitchnm
     SMDL            NAME            pitchnm
>

                <!-- Musica Ficta Gamut -->
<!ELEMENT fictagam      -- Musica ficta gamut definition. --
             - O      (pitchnm, fictadj)+
>

<!ATTLIST fictagam
     SMDL            NAME            fictagam
        id              ID            #IMPLIED
>

                <!-- Interval -->
<!ELEMENT interval -- Scale difference or frequency ratio of two
                pitches --
             O O      (gamintvl | arbintvl | jipitint)
>

<!ATTLIST interval
     SMDL            NAME            interval
>

                <!-- Gamut-based Interval -->
<!ELEMENT gamintvl -- Gamut-based interval: number of name steps and
                gamut steps between two gamut pitch
                definitions --
             O O      (nsdist, gsdist)
>

<!ATTLIST gamintvl
     SMDL            NAME            gamintvl
```

```
>

                <!-- Name Step Distance -->
<!ELEMENT nsdist   -- Name step distance. lextype (snum). --
              O O      (#PCDATA)
>

<!ATTLIST nsdist
      SMDL            NAME            nsdist
>

                <!-- Gamut Step Distance -->
<!ELEMENT gsdist   -- Gamut step distance. lextype (snum) --
              O O      (#PCDATA)
>

<!ATTLIST gsdist
      SMDL            NAME            gsdist
>


                <!-- Arbitrary Interval -->
<!ELEMENT arbintvl -- Arbitrary interval --
              O O      (gsdist | iratio | formula)
>

<!ATTLIST arbintvl
      SMDL            NAME            arbintvl
>

<!ELEMENT iratio   -- Integer ratio interval
                 lextype (snzi, s+, snzi); --
              O O      (#PCDATA)
>

<!ATTLIST iratio
      SMDL            NAME            iratio
>

<!ELEMENT mudef   -- Microtuning unit definition --
              - O      (interval)
>

<!ATTLIST mudef
      SMDL            NAME            mudef
      id              ID              #REQUIRED
      mucount         --  Number of MUs in the interval --
                NUMBER          #REQUIRED
>

                <!-- Just Intonation Pitch -->
<!ELEMENT jipit -- Just intonation nominal pitch --
              O O      ((gampitch | freqspec | jipitref),
                       interval?)
```

```
>

<!ATTLIST jipit
     SMDL            NAME            jipit
     id              ID              #IMPLIED
>


<!ELEMENT jipitref -- Just intonation nominal pitch reference --
              - O      EMPTY
>

<!ATTLIST jipitref
     SMDL            NAME            jipitref
     idr             IDREF           #REQUIRED
>

                <!-- Just Intonation Interval -->
<!ELEMENT jipitint -- Just intonation nominal pitch interval --
             O -    (jipit, jipit)
>

<!ATTLIST jipitint
     SMDL            NAME            jipitint
>

                <!-- Common Chord Symbol -->
<!ELEMENT ccschord      -- common chord symbol chord change --
                - O  ( ( refpitch, ( chordnm | chordbdy))+,
                     bassnote?)
>

<!ATTLIST ccschord
     SMDL            NAME            ccschord
     chordgam        -- Current chord body gamut --
                IDREF           #IMPLIED
                -- Default: no change --
>

<!ELEMENT bassnote      -- bass note (may also be in body of chord).
                   If polychord, interval refers to first
                   refpitch. --
                - O  (gamintvl)
>

<!ATTLIST bassnote
     SMDL            NAME            bassnote
>


                <!-- Figured Bass (Thoroughbass) -->
<!ELEMENT figbass       -- Figured bass (thoroughbass) chord
                   change. --
                - O ( refpitch, figbsint*)
```

```
>

<!ATTLIST figbass
     SMDL            NAME            figbass
     id              ID              #IMPLIED
>

<!ELEMENT figbsint      -- Figured bass (thoroughbass)
                   interval --
                - O ( nsdist, fictadj?)
>

<!ATTLIST figbsint
     SMDL            NAME            figbsint
     sustain -- ID of last figbass over which voice sustains --
                IDREF           #IMPLIED
                -- Default: no sustain --
>

                <!-- Reference Pitch Value -->
<!ELEMENT refpitch      -- Reference pitch on which chord intervals
                   are based. "Root" for CCS chord; "bass"
                   for figured bass. --
                - O ( gampitch)
>

<!ATTLIST refpitch
     SMDL            NAME            refpitch
>
                <!-- Chord Body -->
<!ELEMENT chordbdy -- Intervals of chord with respect to reference
                pitch --
              O O (chordtxt?, gamintvl*)
>

<!ATTLIST chordbdy
     SMDL            NAME            chordbdy
>

<!ELEMENT chordtxt -- Imprecise chord body description (without root
                or bass) --
              - O (#PCDATA)
>

<!ATTLIST chordtxt
     SMDL            NAME            chordtxt
>

                <!-- Chord Body Gamut -->
<!ELEMENT chordgam -- Chord body gamut --
              - O      (chordnm, chordbdy)+ >
<!ATTLIST chordgam
     SMDL            NAME            chordgam
     id              ID              #IMPLIED
```

```
>

<!ELEMENT chordnm  -- Chord body name --
              - O (#PCDATA)
>

<!ATTLIST chordnm
      SMDL            NAME            chordnm
>









<!--  needed? -->
<!ELEMENT perform
              - -   ANY
>
<!ATTLIST perform
      HyTime          NAME            ilink
      SMDL            NAME            "perform"
      anchrole        CDATA   #FIXED  "perform notation #AGG cantus #AGG"
      id              ID              #IMPLIED  -- Default: none --
      linkends        -- Link ends --
                  -- Constraint: one anchor per anchor role. If one
                     is omitted, ilink element is first anchor. --
                  -- Constraint: No HyTime reftype constraints,
                      but application designers can constrain
                      element types with reftype attribute --
                  IDREFS          #REQUIRED
      extra            -- External access traversal rule --
                  -- Constraint: one per anchor or one for all --
                  -- lextype(("E"|"I"|"A"|"N"|"P"),
                          (s+, ("E"|"I"|"A"|"N"|"P"))*) --
                  NAMES           #IMPLIED   -- Default: no HyTime
                                             traversal --
      intra            -- Internal access traversal rule --
                  -- Constraint: one per anchor or one for all --
                  -- lextype(("E"|"I"|"A"|"N"|"P"),
                              (s+, ("E"|"I"|"A"|"N"|"P"))*) --
                  NAMES           #IMPLIED  -- Default: no HyTime
                                            traversal --
      endterms         -- Link end term information --
                  -- Constraint: one per anchor or one for all --
                  -- reftype(HyBrid) --
                  IDREFS          #IMPLIED  -- Default: none --
      aggtrav           -- Traversal of agglink anchors: agg or members --
                   -- Constraint: one per anchor or one for all --
                   -- lextype(("AGG"|"MEM"|"COR"),
                           (s+, ("AGG"|"MEM"|"COR"))*) --
                  NAMES           agg
```

```
                -- HyTime conloc attribute. --
      conloc          -- ID of element to be used in place of, or as
                      content of, this element (see HyTime
                      International Standard). --
                IDREF           #CONREF

>

<!-- link to NIFF, SCORE or bitmap -->

<!ELEMENT score
            - -  ANY
>

<!ATTLIST score
      HyTime          NAME            ilink
      SMDL            NAME            "score"
      anchrole        -- Anchor roles --
                -- Constraint: one per anchor --
                -- lextype((NAME, s+, (RNI, "AGG")?),
                             (s+, NAME, s+, (RNI, "AGG")?)+) --
                CDATA   #FIXED  "score edition #AGG cantus #AGG"
      id              ID              #IMPLIED  -- Default: none --
      linkends        -- Link ends --
                -- Constraint: one anchor per anchor role. If one
                   is omitted, ilink element is first anchor. --
                -- Constraint: No HyTime reftype constraints,
                    but application designers can constrain
                    element types with reftype attribute --
                IDREFS           #REQUIRED
      extra           -- External access traversal rule --
                -- Constraint: one per anchor or one for all --
                -- lextype(("E"|"I"|"A"|"N"|"P"),
                       (s+, ("E"|"I"|"A"|"N"|"P"))*) --
                NAMES            #IMPLIED   -- Default: no HyTime
                                        traversal --
      intra           -- Internal access traversal rule --
                -- Constraint: one per anchor or one for all --
                -- lextype(("E"|"I"|"A"|"N"|"P"),
                            (s+, ("E"|"I"|"A"|"N"|"P"))*) --
                NAMES           #IMPLIED  -- Default: no HyTime
                                      traversal --
      endterms        -- Link end term information --
                -- Constraint: one per anchor or one for all --
                -- reftype(HyBrid) --
                IDREFS          #IMPLIED  -- Default: none --
      aggtrav          -- Traversal of agglink anchors: agg or members --
                -- Constraint: one per anchor or one for all --
                -- lextype(("AGG"|"MEM"|"COR"),
                        (s+, ("AGG"|"MEM"|"COR"))*) --
                NAMES           agg
                -- HyTime conloc attribute. --
      conloc          -- ID of element to be used in place of, or as
                      content of, this element (see HyTime
```

```
                  International Standard). --
               IDREF          #CONREF

>


<!ELEMENT analysis
           - -  ANY
>

<!ATTLIST analysis
     HyTime          NAME           ilink
     SMDL            NAME           "analysis"
     anchrole        -- Anchor roles --
               -- Constraint: one per anchor --
               -- lextype((NAME, s+, (RNI, "AGG")?),
                          (s+, NAME, s+, (RNI, "AGG")?)+) --
               CDATA   #FIXED  "analysis #AGG cantus #AGG"
     id              ID             #IMPLIED  -- Default: none --
     linkends        -- Link ends --
               -- Constraint: one anchor per anchor role. If one
                  is omitted, ilink element is first anchor. --
               -- Constraint: No HyTime reftype constraints,
                   but application designers can constrain
                   element types with reftype attribute --
               IDREFS           #REQUIRED
     extra           -- External access traversal rule --
               -- Constraint: one per anchor or one for all --
               -- lextype(("E"|"I"|"A"|"N"|"P"),
                      (s+, ("E"|"I"|"A"|"N"|"P"))*) --
               NAMES            #IMPLIED   -- Default: no HyTime
                                    traversal --
     intra           -- Internal access traversal rule --
               -- Constraint: one per anchor or one for all --
               -- lextype(("E"|"I"|"A"|"N"|"P"),
                           (s+, ("E"|"I"|"A"|"N"|"P"))*) --
               NAMES           #IMPLIED  -- Default: no HyTime
                                    traversal --
     endterms        -- Link end term information --
               -- Constraint: one per anchor or one for all --
               -- reftype(HyBrid) --
               IDREFS          #IMPLIED  -- Default: none --
     aggtrav          -- Traversal of agglink anchors: agg or members --
                -- Constraint: one per anchor or one for all --
                -- lextype(("AGG"|"MEM"|"COR"),
                       (s+, ("AGG"|"MEM"|"COR"))*) --
               NAMES           agg
               -- HyTime conloc attribute. --
     conloc          -- ID of element to be used in place of, or as
                  content of, this element (see HyTime
                  International Standard). --
               IDREF           #CONREF

>
```

```
<!-- Should it be defined in the same way as as score etc,
     i.e. an ilink to an external file? Or do we
     want this element to contain information in SGML format -
     yes for panorama.
     Recall that %bib.att contains idrefs to bibinfo elements.
     should bibinfo not have an id? Otherwise how does the
     idref refer to it? I added one -->

<!-- Changed ANY content to (#PCDATA) -->
                <!-- Bibliographic Data -->
<!ELEMENT bibinfo
             -- Bibliographic data --
             - O  (themes?,author,uniform,isbd)
>

<!ELEMENT author - o (#PCDATA) >
<!ELEMENT uniform - o (#PCDATA) >
<!ELEMENT isbd - o (#PCDATA) >


<!-- following structure defined for cantate -->

<!-- I have added an id -->
<!ATTLIST bibinfo
     SMDL            NAME            bibinfo
     id              ID              #REQUIRED
>

<!-- is theme idrefs (from bib.att) referring to themes
     elements?If so why no id? I have added one which is implied -->

                <!-- Themes -->
<!ELEMENT themes -- Themes that best identify the work(e.g. incipit) --
            - O  (ilink)*
>
<!ATTLIST themes
     SMDL            NAME            themes
     id              ID              #IMPLIED

>

<!ELEMENT nameloc - o (nmlist*)>
<!ATTLIST nameloc HyTime NAME nameloc
             id     ID   #REQUIRED>
<!ELEMENT nmlist  - o (#PCDATA) --lextype (NAMES) -->
<!ATTLIST nmlist  HyTime NAME nmlist
             nametype (entity|element|unified) #FIXED entity
             obnames (obnames|nobnames) #FIXED nobnames
>
```

APPENDIX B: SGMLniff DTD

(by Steven Mounce)

```
<!-- S.R.Mounce  Project: LIB-CANTATE/4-2016  Last Modified: 31/5/96 -->

<!-- the dtd for the intermediate sgml version of NIFF follows.
   At each stage the full specification is detailed. Recall
   that this dtd assumes the various (list)components are in
   the correct order. Also, it (for the moment) ignores MIDI
   information. Note that it may be worth incorporating the
   RIFF INFO list which contains  comments about the piece
   see p27
   note on tags: any tag can be placed (theoretically) on any
   chunk. It is up to the reading program to interprete (or
   ignore) it. My program will handle only the most basic tags
   (i.e. logical info).
   To check: are the following definitely only graphical?
   staff groupings list (and hence chunks)
   line chunk
   -->



<!ELEMENT niff  - o (setup, data)>
<!-- +tags?-->
<!-- RIFFINFOL is assumed not to be needed (handled/ignored) by
 previous program component.chunkLengthTCH is utilized by the NIFF
 parser, ignored by the IFG.stringTabCH is used and discarded by IFG.
 defValCH,fontDesL,custGraphL are all concerned with fonts and
 graphics. -->

<!ELEMENT setup - o (info, partsL) ><!-- staffgrpL would follow parts
                                      if it is in struc -->

<!ELEMENT info - o (#PCDATA) >

<!-- info content is the version followed by an integer from the
following
      set (neg 1,1,2,3,4,5,6)
 The number denotes the source of the file as follows:
 neg 1 no value/other
 1 engraving program
 2 scanning program
 3 MIDI interpreter
 4 sequencer
 5 research program
 6 educational program
 -->

<!ELEMENT partsL - o (part)+ >

<!ELEMENT part - o (id, name) >
```

```
<!-- is a lyric tag necessary? -->

<!ELEMENT id - o (#PCDATA)>

<!ELEMENT name - o (#PCDATA)>

<!-- not sure if things to do with staffs required, so removed from
structure
    -->

<!-- the id is referred to on Data Section chunks associated with this
part. Assigned sequentially from zero -->


<!ELEMENT staffgpL - o  (staffgpC)+ >
<!ELEMENT staffgpC - o  (grouping) >
<!ELEMENT grouping - o  (#PCDATA) ><!-- 1,2 or 3 -->
<!ATTLIST staffgpC first IDREF #REQUIRED
                   last  IDREF #REQUIRED> <!-- staff id -->
<!-- A staff Grouping chunk describes
some type of symbol at the left end of a series of sequential staves -
such as an initial vertical line. The one present in the setup section
is the default. Is this really necessary logical info for SMDL file?
Remove if not -->



<!--                              DATA                              -->

<!ELEMENT data - o (pageL)* >
<!-- Theoretically there could be none -->

<!ENTITY % plcont "(systemL | text)*">
<!-- is text chunk just graphical? -->
<!ELEMENT pageL - o (%plcont)>
<!ELEMENT text - o (#PCDATA)>
<!-- Specific tags: -->
<!-- ELEMENT pagehedC - o EMPTY
no use at all since height, width tags not needed? -->

<!-- NIFF font symbol Chunk, custom Graphic symbol chunk and line chunk
     are graphical. Is text chunk needed, since there is one for lyrics -
->

<!ENTITY % slcont "(staffL|text)*">

<!ELEMENT systemL - o (%slcont)>

<!-- ELEMENT systemhC - o EMPTY
system header chunk not required -->


<!-- the placement, width, height tags are all to do with graphical
placement -->
```

```
<!ENTITY % anymusC "accident|arpeggio|chordsym|clef|figbass
                                    |keysig|lyric|notehead|rest|repeat
                                    |stem|tagactiv|taginact|tempomk
                                    |text|tie|timesig|tuplet">
<!-- valid music symbol-->

<!ENTITY % tagcon "partid|voiceid|altern|grace|mnid|number|
                                  ossia|silent|tupletds">
<!-- valid tags -->

<!ELEMENT partid - o (#PCDATA)><!-- partid = 'p',int -->
<!ELEMENT voiceid - o (#PCDATA)><!-- an integer -->
<!ELEMENT altern - o (#PCDATA)><!-- an integer -->
<!ELEMENT grace - o (duration)>
<!ELEMENT mnid - o (#PCDATA)><!-- an integer -->
<!ELEMENT number - o (#PCDATA)><!-- an integer -->
<!ELEMENT ossia - o (#PCDATA)><!-- actually empty -->
<!ELEMENT tupletds - o (transab, transcd)>
<!ELEMENT transab o o (duration)>
<!ELEMENT transcd o o (duration)>

<!ELEMENT tag - o (%tagcon)>

<!ENTITY % stafcon "(ts | %anymusC)*">

<!ELEMENT staffL - o (stafhedC, %stafcon) +(tag)>

<!ELEMENT stafhedC - o (#PCDATA)>

<!-- won't parse with empty content -->
<!-- tags: partId, ossia,silent -->

<!ELEMENT ts - o (tstype, top, bottom)>
<!ELEMENT tstype - o (#PCDATA)> <!-- 1(measure start) or 2 (event) -->
<!ELEMENT top - o (#PCDATA)>
<!ELEMENT bottom - o (#PCDATA)><!-- RATIONAL see p34 -->


<!ELEMENT accident - o (#PCDATA) >
<!-- number from 1-9 -->
<!-- Specific tags:        -->

<!ELEMENT arpeggio - o (#PCDATA) >
<!-- applies to preceding chord? Or need refs to specify? -->
<!-- Specific tags:        -->

<!ELEMENT chordsym - o (#PCDATA) >
<!-- is this just graphical or does it have logical fn -->
<!-- Specific tags:        -->

<!ELEMENT clef - o (type,octave) >
<!ELEMENT type - o (#PCDATA) ><!-- treble,bass,alto,tenor -->
<!ELEMENT octave - o (#PCDATA) ><!-- -2,-1,0,1,2 -->
```

```
<!-- will be necessary to determine pitch gamut to use -->
<!-- Specific tags: voiceID?,       -- >

<!ELEMENT figbass - o (#PCDATA) >
<!-- figbass in SMDL -->
<!-- Specific tags:         -->

<!ELEMENT keysig - o (#PCDATA) >
<!-- e.g sharps or flats -->
<!-- Specific tags:         -->

<!ELEMENT lyric  - o (#PCDATA) >
<!-- probably need some more info(time slices etc) -->
<!-- Specific tags:         -->

<!ELEMENT notehead - o (stafstep, duration)>
<!ELEMENT stafstep - o (#PCDATA) >
<!ELEMENT duration - o (top,bottom) >
<!-- Specific tags: partID(if not default), voiceID, gracenote,
                    cuenote?,silent?-->

<!ELEMENT repeat - o (#PCDATA) >
<!-- a logical code telling what type of repeat to do -->
<!-- Specific tags:         -->

<!ELEMENT rest - o (duration) >
<!-- Specific tags:         -->

<!ELEMENT stem - o (#PCDATA) >
<!-- don't think this is required -->
<!-- Specific tags: see draft-->

<!ELEMENT tagactiv - o EMPTY >
<!-- Specific tags: tags which are to be applied to all following
 chunks until deactivated (either separately or together) -->
<!ELEMENT taginact - o EMPTY >

<!ELEMENT tempomk - o (#PCDATA) >
<!-- may not be required: if it is need more detail -->
<!-- Specific tags: -->

<!ELEMENT tie - o EMPTY >
<!-- attribute of pitched in smdl - need ref to event tied to? -->
<!-- Specific tags:? -->

<!ELEMENT timesig - o (top,bottom)>


<!-- Specific tags: -->

<!ELEMENT tuplet - o (#PCDATA) >
<!-- seems to require stems, but do stems always have noteheads?
```

```
PCDATA contains info from tuplet description tag -->
<!-- Specific tags: -->
```

APPENDIX C - Further Documentation

1. Carola Boehm, Cordy Hall, ISO MPEG7 proposal 'Description Scheme for description of music content, *ISO/IEC JTC1/SC29/WG11/ MPEG 98/P620/W2463'* (Vancouver: MPEG 1999)

2. DRH99 Abstract for the Expert Workshop and Paper Presentation held under the auspices of the "Digital Resources in the Humanities Conference", London 1999

3. Announcement of the Glasgow user workshop organized by MuTaTeD!

4. Announcement of the Paper Presentation of MuTaTeD! at the Ars Electronica Festival under the auspices of the CUIDAD EU-funded working-group.

5. Slides for the Presentations at the DRH 99 and Glasgow Workshop