	G2 (98.0)	C3 (130.8)	G3 (196.0)	C4 (261.6)	G4 (392.0)	C5 (523.3)	G5 (784.0)	C6 (1046.5)
10ms	–	**139.1**	193.9	258.5	392.0	525.0	787.5	1052.1
15ms	99.0	131.0	195.1	261.2	394.5	524.1	786.4	1052.3
20ms	98.9	131.4	195.1	261.8	393.7	523.5	785.6	1052.3
30ms	98.9	131.4	195.2	261.8	393.7	523.9	785.6	1052.1

Table 3: Results of PPPT algorithm (incorrect results for 10ms C3; not enough samples to run 10ms G2 case).

the 10-millisecond segment of G2 because the fundamental period of that note is 227.0 samples and the number of samples in the segment is 223 (at a sampling rate of 22255Hz). The algorithm also converges to an incorrect fundamental frequency for the 10-millisecond segment of C3. Furthermore, note that the frequency estimates generated by this algorithm are closer to the actual frequencies than the two frequency-domain algorithms.

Summary

This paper reports experiments that show how the accuracy of the FFR described in Brown (1992) is affected by different window widths. It then proposes a new FFR algorithm with higher accuracy for narrow analysis windows. It also describes a modification to the PPPT algorithm of Cook et al. for improved operation in real time.

References

Amuedo, J. (1985). Periodicity estimation by hypothesis-directed search. In *Proc. of ICASSP '85*, (Tampa, Florida, May 1985), 395–398.

Brown, J.C. (1991). Calculation of a constant Q spectral transform. *J. Acoust. Soc. Am. v. 89*, no. 1, (Jan. 1991), pp. 425–434.

Brown, J.C. (1992). Musical fundamental frequency tracking using a pattern recognition method. *J. Acoust. Soc. Am. v. 92*, no. 3, (Sep. 1992), pp. 1394–1402.

Brown, J.C. and Puckette, M.S. (1992). An efficient algorithm for the calculation of a constant Q transform. *J. Acoust. Soc. Am. v. 92*, no. 5, (Nov. 1992), pp. 2698–2701.

Brown, J.C. and Puckette, M.S. (1993). A high resolution fundamental frequency determination based on phase changes of the Fourier transform. *J. Acoust. Soc. Am. v. 94*, no. 2, Pt. 1, (Aug. 1993), pp. 662–667.

Cook, P.R., Morrill, D., and Smith, J.O. (1993). A MIDI control and performance system for brass instruments. In *Proc. of ICMC*, (Tokyo, Japan, 1993), 130–133.

Doval, B. and Rodet, X. (1991a). Fundamental frequency estimation using a new harmonic matching method. In *Proc. of ICMC*, (Montreal, Cananda, 1991), 555–558.

Doval, B. and Rodet, X. (1991b). Estimation of fundamental frequency of musical sound signals. In *Proc. of ICASSP '91*, (Toronto, Canada, May 1991), 3657–3660.

Kuhn, W.B. (1990). A real-time pitch recognition algorithm for music applications. *Computer Music Journal, v. 14*, no. 3, (Fall 1990), pp. 60–71.

Lane, J.E. (1990). Pitch detection using a tunable IIR filter. *Computer Music Journal, v. 14*, no. 3, (Fall 1990), pp. 46–59.

Ney, H. (1982). A time warping approach to fundamental period estimation. *IEEE Trans. on Systems, Man, and Cybernetics, v. SMC-12*, no. 3, (May/June 1982), pp. 383–388.

Pearson, E.R.S. and Wilson, R.G. (1990). Musical event detection from audio signals within a multiresolution framework. In *Proc. of ICMC*, (Glasgow, 1990), 156–158.

A Linguagem *SOM-A* para Síntese Aditiva

ALUIZIO ARCELA
Laboratório de Processamento Espectral
Departamento de Ciência da Computação
Universidade de Brasília
Brasília, DF — CEP 70910-900
e-mail: arcela@lpe1.cic.unb.br

Resumo

Descrição formal da sintaxe, do universo semântico, dos operadores e de alguns aspectos de implementação da linguagem *SOM-A*. Com o interpretador desta linguagem, que é voltada exclusivamente para a síntese aditiva de sinais musicais, executam-se partituras polifônicas associadas a orquestras, para as quais se definem, uma a uma, as componentes espectrais em todos os seus parâmetros, isto é, ordem de freqüência, ângulo inicial de fase, curva de envoltória, e grau de estereofonia.

HISTÓRICO

Um primeiro esboço para a linguagem *SOM-A* surgiu em 1986 a partir de algumas experiências com o programa Music V (Mathews et. al. 1969). Tais experiências giravam em torno da tentativa de se interpretarem composições algorítmicas baseadas nas árvores de tempos (Arcela 1986), as quais, na maioria das vezes por força do modelo, possuíam uma grande quantidade de componentes espectrais, e eram caracterizadas, do ponto de vista da escrita de eventos e mecanismos espectrais, por não haver nelas, ao menos aparentemente, qualquer possibilidade ou indício de serem interpretadas por um processo que não fosse o da síntese aditiva. Além disso, registrava-se nessas composições a exigência de uma certa projeção acústica estereofônica, segundo a qual um subconjunto bem definido do universo de componentes espectrais deveria soar em apenas um dos canais, enquanto as demais componentes soariam no outro canal, exigência esta que trazia uma ligeira alteração na arquitetura do instrumento mínimo necessário à síntese aditiva padrão, isto é, a que se registra na literatura, como em (Moorer 1977). As estruturas de dados estáticas da implementação FORTRAN do sistema Music V permitiam apenas a execução de instrumentos possuídores de relativamente poucas componentes. E como havia a intenção de se perfazer uma interpretação plena dessas estruturas musicais em todas as suas partes, sem que houvesse qualquer truncamento no conjunto de componentes, a idéia que restava era investir na definição de uma linguagem que pudesse aceitar um instrumento de qualquer tamanho, e que tivesse uma única especialidade: a síntese aditiva. E assim, com apenas 5 operadores, e com uma sintaxe semelhante a de LISP — os instrumentos e os eventos espectrais devem ser escritos rigorosamente na forma de expressões simbólicas —, surgiu a linguagem *SOM-A*, assumindo a simplicidade como sua característica maior.

Como não podia ser de outra forma, a primeira implementação ocorreu em LISP (Nogueira Filho 1988), que era sem dúvida a atitude mais correta e natural, levando-se em conta a natureza das estruturas de dados escolhidas para representar os elementos de *SOM-A*. Naquela época, o que de melhor havia para o sistema operacional MSDOS era o interpretador muLISP87 (Mulisp 1987), um ambiente satisfatório em muitos aspectos, mas oferecendo como obstáculo crucial à implementação de programas voltados para a síntese de sinais de áudio justamente a sua inevitável execução de operações aritméticas totalmente por software. Às vezes, para se interpretar um trecho de 10s, dependendo da complexidade orquestral, eram gastas entre 10 e 20 horas de processamento em intel286 a 10 MHz. Há o caso de uma peça de 1'20" contendo instrumentos de até 76 componentes espectrais para a qual foram necessários 25 dias ininterruptos de processamento! Mesmo assim, a implementação de *SOM-A* em LISP para pequenas máquinas assumiu uma importância capital no desenvolvimento da linguagem, pois, se não consegue executar peças de conteúdo espectral denso em um prazo razoável, ela vem cumprindo o papel de uma especificação

viva desta linguagem para síntese aditiva, a partir da qual são levadas a cabo outras implementações através de linguagens mais ágeis no tocante a operações aritméticas, como a implementação em C++ para MS-Windows (Castro 1994), que apresenta um bom desempenho — a tal peça de 25 dias é aí concluída em cerca de 2 horas em processador intel486 a 66MHz —, além de oferecer uma interface MS-Windows de boa funcionalidade em diversos itens, como o acompanhamento visual das formas de onda à medida que o sinal vai sendo produzido, a manipulação de três formatos para arquivos de amostras (wav, voc, str), e o acionamento de placas de som (atualmente, USE e SoundBlaster 16) ao final do processo ou em seguida a uma interrupção do usuário. Está em curso o transporte desta versão em C para uma máquina Sun (Meireles et. al.). Vale ainda registrar que, entre o trabalho inicial feito com muLISP e este último em Borland C, houve uma implementação cuidadosa em Sun Common LISP para ambiente UNIX (Ramalho 1991), cujo resultado apresentou um desempenho bastante superior ao que se consegue com o LISP para DOS, principalmente em termos de interface e velocidade. Havia, contudo, uma expectativa por um rendimento bem melhor, levando-se em conta que em Common LISP as variáveis numéricas podem ser tipificadas em ponto flutuante, o que torna as operações aritméticas envolvidas em *SOM-A* totalmente executáveis por hardware.

Em todas esses projetos foram adotadas técnicas adequadas para gerenciamento de memória virtual para que as orquestras possuidoras de um grande número de componentes não tivessem problemas de espaço na máquina a ponto de se ver impedida a execução de uma dada partitura. O emprego desta técnica segue o já mencionado preceito de implementação *SOM-A*, qual seja, o de não haver limites para a orquestra quanto ao maior número possível de unidades-H, nem quanto ao número de notas simultâneas endereçadas a essa orquestra, a não ser os últimos limites de espaço impostos pela máquina física que abriga uma implementação da linguagem.

OS ELEMENTOS

Unidade-H é a denominação do mecanismo responsável pela geração de uma componente espectral no contexto da linguagem *SOM-A*. Especificamente, unidade-H é o algoritmo composto por um oscilador senoidal a tabela, cuja amplitude é controlada por um gerador de envoltória, e cuja entrada de freqüência é multiplicada por um valor denominado ordem de freqüência, conforme ilustra a Figura 1. A saída de uma unidade-H poderá ser dirigida a um canal de áudio x, ou a um canal y, dependendo do valor do parâmetro de estereofonia. Construir um instrumento musical em *SOM-A* significa especificar um agrupamento de várias dessas unidades em configuração de síntese aditiva, conforme a Figura 2, de modo que todas as saídas parciais serão somadas para cada canal de áudio, e haverá uma única entrada de freqüência e uma única entrada de amplitude.

Constroi-se um **instrumento *SOM-A*** a partir de um número qualquer de componentes, todas elas possuindo um comportamento espectral bem definido e individualizado. Para se executar uma nota nesse instrumento, são

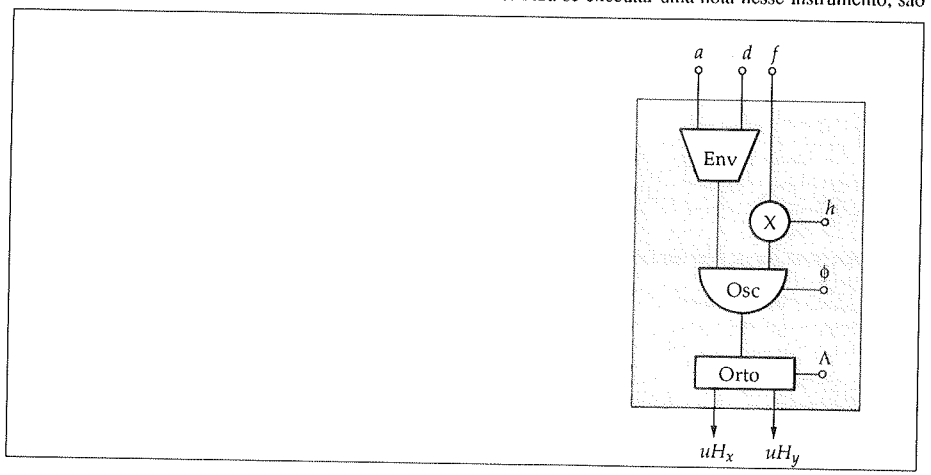

Fig. 1 A unidade-H

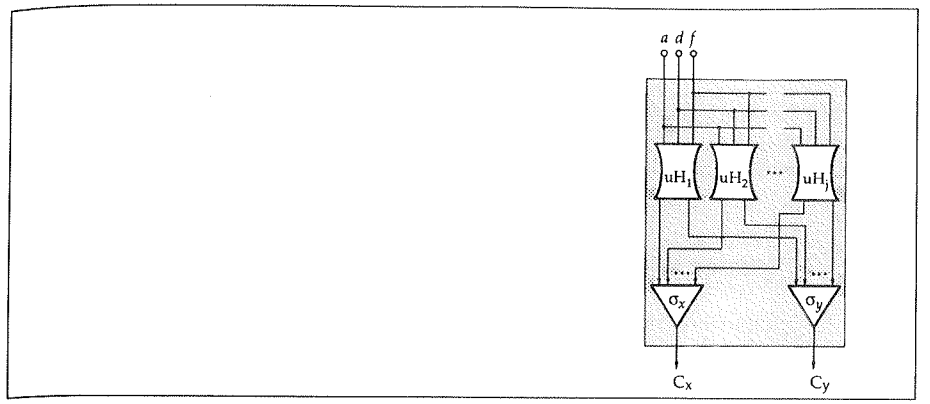

Fig. 2 Um instrumento SOM-A

utilizados na partitura comandos de notas que, na sua apresentação mais simples, fazem menção a toda a população de unidades-H do instrumento, porém, quando escrito de uma forma mais específica, solicita ao instrumento que acione seletivamente um bem definido subconjunto de unidades. Esta capacidade singular de fracionar o instrumento em grupos menores concede à linguagem a noção e, por conseguinte, o mecanismo do **subinstrumento**, noção esta útil em composição musical que explora a elaboração sistemática ou algorítmica de timbres. A ação do subinstrumento é eficaz especialmente na obtenção de timbres correlatos, mas também possibilita o emprego de uma certa "economia" de componentes espectrais no espaço tímbrico global de uma eventual peça.

SINTAXE

SOM-A é um interpretador de programas espectrais que possuem pelos menos duas partes: uma orquestra feita de unidades-H, que é o primeiro bloco sintático, e uma seqüência de notas. Estes programas são denominados *cartas espectrais*, e constam de arquivos de caracteres representando uma seqüência de instruções capazes de criar instrumentos, atribuir valores de lapso e de taxa de amostragem, bem como executar nos instrumentos um aglomerado de notas, sejam elas simultâneas ou não, segundo uma ordenação cronológica e uma normalização de amplitudes realizadas previamente.

1. Operador *VAL*

É sempre o primeiro comando de uma carta, como um cabeçalho, devendo ser aplicado sob a forma:

(VAL t_1 t_2 F_a A T N L)

Sua função é atribuir valores ao lapso de tempo t_1 t_2 em segundos, à taxa de amostragem F_a em *cps*, aos 3 modificadores *A T N* de interpretação de notas, e ao número de pontos máximo *L* para as envoltórias. Estes parâmetros são assim definidos:

- A *Andamento*: valor que multiplica (com exceção dos tempos t_1 t_2 estabelecidos por *VAL*) todos os tempos e durações, quais sejam, (1) a vigência dos instrumentos; (2) os parâmetros t_a t_b do operador *EXE*; (3) o instante inicial e (4) a duração (*d*) da nota.
- T *Transposição:* valor que multiplica as freqüências (*f*) das notas, sem, contudo alterar as ordens de freqüência dos instrumentos.
- N *Norma:* valor que multiplica as amplitudes (*a*) das notas.
- L *Limite de envoltória:* valor limite para as abscissas das envoltórias.

Exemplo: a instrução (VAL 0 10 12000) solicita que a carta seja interpretada de 0–10 *s* a uma taxa de amostragem

de 12000 *cps*. Como os modificadores não estão explicitados, todos eles assumem o valor 1, enquanto que o limite de envoltória valerá 511 por default.

2. Operador *INS*

Define um instrumento na íntegra, através de uma expressão da forma:

(**INS** <vigência> <nome-ins> <unidades-H>)

para o qual:

<vigência> é o instante a partir do qual o instrumento <nome> deixa de existir.

<nome-ins> é uma cadeia de caracteres possuindo obrigatoriamente um literal como primeiro caractere.

<unidades-H> é uma lista de sublistas $((h_1\ \phi_1\ e_1\ \Lambda_1)\ (h_2\ \phi_2\ e_2\ \Lambda_2)\ \ldots\ (h_k\ \phi_k\ e_k\ \Lambda_k))$ que determinam os valores das partes funcionais de cada unidade-H do instrumento, quais sejam:

h_i *ordem de freqüência* da i-ésima unidade: número real positivo.

ϕ_i *ângulo de fase*: valor expresso am graus.

e_i *envoltória*: Lista de pares de coordenadas (*ordenada* × *abscissa*) sendo a ordenada um valor real compatível com os limites numéricos da implementação, e a abscissa um número inteiro compreendido entre zero e L. Estes pares possibilitam a especificação de uma forma geométrica através de até $L + 1$ amostras de uma curva que modulará em amplitude a respectiva componente espectral e que se ajustará à duração da nota.

Λ_i *grau de estereofonia*. Valor entre 0 e 1 que corresponde ao balanço de saída da componente com relação aos canais estereofônicos x, y. Se for 0, a saída estará totalmente em x.

3. Operador *EXE*

A aplicação deste operador na forma abaixo submete uma sequência de eventos espectrais <<notas>> entre dois instantes aos respectivos instrumentos da orquestra:

(EXE t_a t_b)

<<notas>>

(STP)

Em que <<notas>> é uma seqüência de notas <nota$_1$> <nota$_2$> ... <nota$_n$>. Diversas seções (EXE t_a t_b) <<notas>> (STP) podem compor uma carta, uma vez que o interpretador *SOM-A* entende o operador *EXE* da seguinte maneira: "execute todas as notas da seqüência abaixo até que o operador (STP) ocorra, ou o instante de execução t seja maior ou igual a t_b". Na situação mais comum, t_a é zero e t_b é a duração da carta. Ainda que se possa utilizar um mesmo instrumento para diferentes vozes superpostas ou não, uma das imposições do interpretador *EXE* é que as notas estejam ordenadas segundo o seu segundo parâmetro, isto é, devem aparecer na carta em ordem cronológica.

<nota> é uma ordem de execução de um certo instrumento previamente definido na orquestra, de modo que ela será tocada a partir de um certo instante, por um certo tempo, a uma dada freqüência, a uma dada intensidade, sendo escrita como uma lista de 5 elementos:

<nota> = (<instrumento>
<instante inicial>
<duração>
<freqüência em cps>
<amplitude>)

O instrumento solicitado pela nota pode ser escrito como um átomo ou como uma lista, isto é:

<instrumento> = <nome-ins> | <lista definidora de um sub-instrumento>

No caso de ser um átomo, a nota estará exigindo a ação plena do instrumento. De outro modo, isto é, se o instrumento for representado por uma lista, a nota estará clamando por um subconjunto das unidades-H de um determinado

instrumento, o qual passará a ser considerado como um **macro-instrumento**. A seleção de unidades-H originária desse macro-instrumento é o que se denomina **subinstrumento**, que é assim definido:

<subinstrumento> = (<nome do macro-instrumento>
<(n_1 n_2 ... n_j)>
<vigência do subinstrumento>)

sendo o primeiro elemento é um átomo, representando um macro-instrumento, o segundo é uma lista de números representando unidades-H selecionadas desse macro-instrumento. Cada número representando a posição da unidade-H na seqüência <unidades-H>. O terceiro é opcional. Trata-se da vigência do subinstrumento criado com a nota. Quando um subinstrumento é criado para atender apenas a uma nota, não há necessidade de se especificar a vigência; *SOM-A* se encarregará de calcular este valor.

4. Operador *STP*

V. operador *EXE*.

5. Operador *FIM*

A aplicação do operador (FIM) encerra a interpretação da carta.

DUAS CARTAS

Na primeira carta abaixo, a primeira nota solicita um subinstrumento derivado de I1, de tal maneira que apenas a sua primeira componente será ativada (10x10=100 Hz), soando inteiramente no canal x. Embora esta primeira nota dure apenas 0.2 s, a vigência do subinstrumento foi fixada em 0.8 em razão da quarta nota vir a utilizar este mesmo subinstrumento entre os instantes 0.6 e 0.8 s. A segunda nota solicita a segunda componente totalmente no canal y. A terceira faz soar 80% da terceira componente no canal x e 20% no canal y. A quarta é semelhante à primeira. A quinta nota solicita um subinstrumento com duas unidades-H de I1, justamente a primeira e a segunda; e a penúltima nota solicita todas as unidades-H de I1, o que corresponde a se utilizar I1 por um todo, sendo o resultado equivalente à ação da sétima e última nota.

```
(VAL 0 1.6 11025 1 1 1 925)
(INS 1.6 I1
   (10   0     ((0 0) (30000 385) (4000 343) (0 925)) 0)
   (12   90    ((0 0) (7500 385) (0 925)) 1)
   (15   180   ((0 0) (3500 385) (0 925)) 0.2))
(EXE 0 1.6)
   ((I1   (1) 0.8)  0.0 0.2 10 1)
   ((I1   (2) 0.4)  0.2 0.2 10 1)
   ((I1   (3))      0.4 0.2 10 1)
   ((I1   (1))      0.6 0.2 10 1)
   ((I1   (1 2))    0.8 0.2 10 0.6)
   ((I1   (1 2 3))  1.0 0.2 10 0.4)
   (I1              1.4 0.2 10 0.25)
(STP)
(FIM)
```

A carta abaixo é um extrato de 1s da composição algorítmica T3OLD.CAR, ilustrando (1) a ortoestereofonia estrita, (2) a ocorrência de valores negativos nas envoltórias, e (3) valores não inteiros — sendo alguns menores do que a unidade — para as ordens de freqüência.

```
(VAL 0  1.0  44100  0.0222222 1 1.25 720)
(INS 24 A21
(1 40.08 ((0 0) (-92 7) (-5134 14) (908 21) (0 720)) 0)
(0.7033 40.55 ((0 0) (2166 9) (-2875 18) (0 720)) 0)
(0.5717 39.91 ((0 0) (1845 10) (0 720)) 0)
(0.9335 41.10 ((0 0) (-3696 8) (0 720)) 0)
)
```

```
(INS 24 A22
(1 43.06 ((0 0) (-1771 12) (-5355 24) (-3304 36) (0 720)) 1)
(1.3289 43.79 ((0 0) (833 11) (-2751 23) (0 720)) 1)
(0.0605 136.43 ((0 0) (4322 14) (0 720)) 1)
(0.245 43.57 ((0 0) (-3707 13) (0 720)) 1)
)
(INS 6 A23
(1 44.79 ((0 0) (5224 9) (0 720)) 0)
(1.4084 44.27 ((0 0) (6886 7) (0 720)) 0)
(1.5638 45.01 ((0 0) (4274 6) (0 720)) 0)
)
(INS 6 A24
(1 45.36 ((0 0) (2950 12) (0 720)) 1)
(0.4581 44.55 ((0 0) (-10723 13) (0 720)) 1)
(2.4536 44.80 ((0 0) (2710 10) (0 720)) 1)
)
(INS 41 A25
(1 29.92 ((0 0) (-778 46) (-428 91) (20 137) (0 720)) 0)
(1.532 40.61 ((0 0) (-575 10) (-225 20) (223 31) (0 720)) 0)
(1.9605 55.89 ((0 0) (-738 1) (-388 2) (60 3) (0 720)) 0)
)
(INS 41 A26
(1 135.47 ((0 0) (1317 93) (740 186) (264 279) (0 720)) 1)
(0.2397 139.32 ((0 0) (1202 111) (625 223) (150 334) (0 720)) 1)
(0.7683 40.93 ((0 0) (972 93) (395 185) (-80 278) (0 720)) 1)
)
(EXE 0 41)
  ((A24 (2 3)) 0    6  2166.678 0.8322)
  ((A23 (2 3)) 0    6  7127.652 0.8322)
  ((A22 (4))   6    6  2826.039 0.8845)
  ((A21 (4))   6    6  8340.816 0.8845)
  ((A22 (3))   12   6  2826.039 0.8845)
  ((A21 (3))   12   6  8340.816 0.8845)
  ((A22 (2))   18   6  2826.039 0.8845)
  ((A21 (2))   18   6  8340.816 0.8845)
  (A26         23   18 3982.623 0.8666)
  (A25         23   18 7822.052 0.8666)
(STP)
(FIM)
```

REFERÊNCIAS

ARCELA, A. 1986. "Time-trees: the inner organization of intervals." *Proceedings of the International Computer Music Conference*, Haia.

CASTRO, R.R.F. 1994. Implementação de *SOM-A* em Borland C^{++} para o ambiente MS-Windows, *Relatório Técnico LPE-9402*, Universidade de Brasília.

MATHEWS, M.V. et. al. 1969. *The technology of computer music*. Cambridge, Mass.: The MIT Press.

MEIRELES, A., GIOIA, O.G. e CASTRO, R.R.F. 1993. *SOM-A* em C para SUN SparcStation, *Relatório Técnico LPE-9303*, Universidade de Brasília.

MOORER, J.A. 1977. "Signal processing aspects of computer music—A survey." *Proceedings of the IEEE* 65(8): 1108-1137.

NOGUEIRA FILHO, V. 1988. Síntese aditiva modular — uma máquina espectral programável. *Dissertação de mestrado*, Universidade de Brasília.

RAMALHO, G.L. 1991. *SOM-A* em Sun Common LISP para o ambiente Open Windows. *Relatório Técnico LPE-9105*, Universidade de Brasília.

MULISP 1987. *muLISP-87, LISP Language Programming Environment*. SoftWarehouse Inc., Honolulu, Hawaii.

Processador de Efeitos em Sinais Digitais de Áudio

MÁRCIO DA COSTA PEREIRA BRANDÃO
CARLOS AUGUSTO JORGE LOUREIRO
TÚLIO DA COSTA ZANNON
Laboratório de Processamento Espectral
Departamento de Ciência da Computação
Universidade de Brasília - Brasília, DF
CEP 70910-900 BRASIL

RESUMO

Um sistema para o processamento de sinais digitais de áudio por meio de estruturas geradas através da interligação de blocos básicos é aqui descrito. Podem ser utilizados diversos tipos de blocos básicos tais como somadores, multiplicadores, osciladores e blocos de retardo. As estruturas estão descritas por arquivos de configuração que contém a forma de interligação, juntamente com os parâmetros de funcionamento destes blocos. O objetivo principal deste trabalho é a construção de estruturas que correspondam a sistemas de ambientação de áudio, tais como filtros, reverberadores e câmaras de eco. Como é possível arbitrar-se as interligações entre os blocos, as configurações disponíveis não são limitadas como no caso da maioria dos processadores de efeitos implementados em 'hardware'. Além disto, o sistema foi idealizado de tal forma que seja fácil a criação de novos blocos através da simples incorporação de novas funções.

1. Introdução

O objetivo deste trabalho é descrever o desenvolvimento de um processador de efeitos para os sinais digitais de áudio produzidos no Laboratório de Processamento Espectral (LPE) da Universidade de Brasília. Estes sinais são gerados pelo programa de síntese ortoestereofônica SOM-A (Arcela, 1989), em conjunto com ferramentas de composição algorítmica que se fundamentam na Teoria da Árvore de Tempos (Arcela, 1984, 1986, 1991).

Através de configurações arbitrárias de interligações entre suas unidades básicas, sinais de áudio armazenados em arquivos são transformados pelo sistema, possibilitando a obtenção dos mais variados tipos de efeitos nos arquivos de áudio de saída. Como o sistema foi idealizado visando o processamento por software, sem o apoio de DSP's, não é possível a sua operação em tempo-real (Lobão, Martinelli, 1992). Esta limitação em parte é compensada pela possibilidade de sua utilização em diferentes plataformas que não disponham de recursos adicionais de hardware.

2. Estrutura Interna

A estrutura interna do processador de efeitos é composta por buffers de dados, buffers de operações e blocos básicos, como mostra a Figura 1. O sistema foi implementado em torno de uma estrutura totalmente baseada no uso de buffers para que seja possível a utilização de configurações arbitrárias para as interligações entre os blocos, além de possibilitar expansões futuras com facilidade.