# Conversão de partituras para tablaturas usando algoritmo baseado em autômato adaptativo

Danilo de Jesus da Silva Bellini, Anna Catarina Batista Tavella

***Resumo*—Orfeu é um *software* que foi desenvolvido para auxiliar músicos amadores e profissionais na edição de músicas. Em suas funções, está inclusa a conversão de partitura para tablatura, realizada através de um algoritmo baseado em autômato adaptativo. O autômato é utilizado para trechos sem simultaneidade sonora enquanto que acordes e bicordes são tratados através de uma rotina específica.**

## I. INTRODUÇÃO

O software Orfeu foi desenvolvido com o objetivo de auxiliar músicos amadores e profissionais na edição de músicas. Possui duas interfaces de edição: a de partitura e a de tablatura. A notação em partitura é, em geral, mais completa, por incluir informações sobre ritmos, dinâmicas, articulações e outras características comumente omissas em tablaturas segundo Junior (2007), além de ser uma notação musical historicamente mais importante (BENNETT, 1996), e a mais utilizada até os dias de hoje pelos músicos. A figura-1 a seguir apresenta as duas notações.

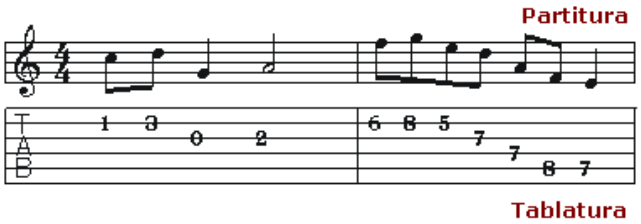


Figura-1: representação em partitura e em tablatura

Toda música no Orfeu é dividida em trilhas, cada uma com propriedades particulares, representando um instrumento musical pertencente à música. A interface visual utiliza apenas uma trilha por vez, exibindo seu conteúdo como partitura ou como tablatura.

A notação em tablatura é bastante utilizada por guitarristas e violonistas, pela maior proximidade com seu instrumento. Na tablatura, a única informação que é sempre apresentada é a altura das notas. Cada número representa uma altura, indicando uma posição ao longo do braço do instrumento (AZEVEDO, 1978), chamado de "casa". Também é comum o uso do termo "traste", principalmente em inglês (*fret*), por ser o nome técnico das barras metálicas que dividem o braço do instrumento em casas. O número zero representa que a corda deve ser tocada solta, ou seja, sem prendê-la a nenhum traste; outro número N indica que a corda deve ser tocada prendendo a ao N-ésimo traste, contando da mão do instrumento ao corpo do mesmo.

Uma funcionalidade do software Orfeu é a conversão entre essas duas formas de representação. Toda nota é sempre representada na partitura e na tablatura simultaneamente, embora suas tablaturas possam ser inválidas, por incluir posições inexistentes no braço do instrumento a ser utilizado. Ao inserir uma nota em alguma das interfaces de edição, é realizada uma conversão dessa informação, porém ignorando o contexto musical presente e, inclusive, a possibilidade de execução. Para contornar esse problema, há no software um recurso adicional de conversão de partitura para tablatura, utilizando-se de adaptatividade (NETO, 1994), em que o contexto musical e a dificuldade de execução são levadas em consideração. O algoritmo utilizado para fornecer esse recurso considera que a trilha a ser convertida já esteja completa, ou seja, que todo o contexto musical já pertença à mesma.

## II. POSSIBILIDADES DE CONVERSÃO DE PARTITURA EM TABLATURA

A estrutura de dados da partitura e da tablatura é basicamente a mesma, mas para este algoritmo, a informação de como os dados são estruturados é irrelevante, importando apenas a maneira de representação das notas e sua ordenação. As notas em partitura são identificadas por um conjunto de três atributos (FAGUNDES, 2004): nome, acidente e oitava; enquanto as notas em tablatura são identificadas por dois atributos: corda e posição (LEAVITT, 2007). Porém, apesar de, aparentemente, as tablaturas serem mais simples de representar, elas exigem o conhecimento prévio do número de cordas do instrumento e da afinação de cada corda, dentro da pauta musical em análise.

Ao fixar uma corda, pode-se facilmente converter cada nota da partitura em uma equivalente na tablatura. Na tablatura, cada nota tem uma posição, um número inteiro que denota a diferença de altura, em semitons, entre a nota representada e a nota da afinação da corda fixada. O processo de obter intervalos em semitons entre duas notas é bastante comum e pode ser encontrado facilmente em livros que contenham os fundamentos da teoria musical (MED, 2001). Quando a nota representada é mais grave que a nota da afinação, e apenas nessa situação, o sinal da posição é negativo. Toda nota possui, portanto, uma posição na corda fixada, porém esse número pode ser negativo ou excessivamente grande. A faixa de valores permitidos é um parâmetro do algoritmo e seu valor usual é [0;24], por ser a faixa completa de valores possíveis na maior parte das guitarras produzidas no mercado. Conforme a afinação das cordas, uma mesma nota pode ser tocada de mais de uma forma.

Embora seja dito que a conversão realizada é de partitura para tablatura, pode-se dizer que esse algoritmo também converte uma tablatura em outra, pois apenas os intervalos, em semitons, são utilizados, e eles estão presentes em ambas as notações.

É dado o nome de "digitação" ao conjunto de posições determinadas pelo algoritmo. O algoritmo não é capaz de converter, de uma forma ideal, qualquer partitura em tablatura. O único critério utilizado foi: em uma melodia, as notas próximas entre si devem possuir as menores diferenças de posição possíveis.

A partir desse critério é possível criar diversos algoritmos de conversão, porém o uso de um algoritmo baseado em autômatos adaptativos mostrou-se adequado para efetuar tal conversão.

## III. Divisão da Trilha

A trilha é dividida em partes para que seja possível a conversão, pois o autômato trabalha apenas com trechos puramente melódicos e contínuos, ou seja, sem pausas e sem simultaneidade de notas; ocorrências como essas servem de separadores entre as partes da trilha, criando três tipos de partes: trechos melódicos, pausa única e bloco com um único agrupamento de notas simultâneas.

As pausas não necessitam de tratamento, visto que não há nota alguma para converter. Elas não são tratadas junto aos trechos melódicos pois, apesar de serem um caso particular de cadeia de entrada nula, o autômato não está preparado para isso.

Os acordes, agrupamento de três ou mais notas simultâneas, e os bicordes, par de notas simultâneas, não são tratados por autômatos adaptativos. Há apenas uma lógica que tenta dispor as notas em cordas que façam a posição das mesmas ficar dentro da faixa de valores permitidos (corda permitida), fornecida como parâmetro. Essa lógica consiste em ordenar as cordas do instrumento da mais aguda até a mais grave e fazer o mesmo com as notas a serem colocadas na tablatura. Cada nota, verificando da mais grave à mais aguda, é colocada na corda permitida de afinação mais grave.

Cada trecho melódico é tratado de forma isolada, utilizando o autômato adaptativo na conversão. O trecho melódico é a entrada do autômato. É designado o nome de "melodia" para esses trechos melódicos.

## IV. Cadeia de Entrada do Autômato Através de n Fitas

Trabalhando com cada melodia, pode-se obter, para cada corda, uma seqüência de números que indicam como a melodia seria tocada no instrumento utilizando apenas essa corda, por exemplo, para um baixo elétrico de quatro cordas com afinação padrão (GDAE), pode-se ter a melodia representada conforme mostrada na tabela-1 (as notas da melodia de exemplo são o início de Greensleaves).

Cada uma das linhas da tabela a seguir é considerada uma fita utilizada pelo autômato, apontadas por um cursor sobre uma única nota, ou seja, sobre uma coluna. O conjunto de todas as fitas do autômato forma a cadeia de entrada do mesmo, ou seja, cada símbolo de entrada é a n-upla formada por cada coluna da tabela de fitas.

Tabela-1: exemplo de representação de melodia

| Afinação | Melodia | A3 | C4 | D4 | E4 | F4 | E4 | D4 |
|---|---|---|---|---|---|---|---|---|
| G3 | | 2 | 5 | 7 | 9 | 10 | 9 | 7 |
| D3 | | 7 | 10 | 12 | 14 | 15 | 14 | 12 |
| A2 | | 12 | 15 | 17 | 19 | 20 | 19 | 17 |
| E2 | | 17 | 20 | 22 | 24 | 25 | 24 | 22 |

Dessa forma o alfabeto de entrada fica definido como todas as n-uplas possíveis para o dado número de cordas. Dentro de cada n-upla pode-se ter como valores qualquer número da faixa de posições permitidas para a tablatura, ou um símbolo que indica que a nota é inválida na corda em questão, o símbolo "E", também utilizado quando não há nota, ou seja, quando o cursor passa a estar fora da fita (antes ou depois da melodia). Nesse último caso descrito, a n-upla contém apenas valores iguais a "E".

Todos os dados de entrada são mantidos imutáveis, e o cursor tem plena liberdade de mover-se tanto para a direita como para a esquerda.

## V. Estados e Transições

O autômato possui um estado para cada corda (indicado por números), um estado de aceitação, um estado de rejeição e um estado de transição (T), com transições conforme constam abaixo no exemplo com três cordas. Cada transição pode ter até três valores. O primeiro valor indica o que será consumido, o segundo valor indica a ação adaptativa que será realizada após efetuar a transição e o terceiro valor indica para que lado o cursor das fitas deve mover-se. Apenas o primeiro valor não pode ser omitido.

O operador unário "~" é usado para indicar um conjunto complementar, ou seja, ~K é o conjunto complementar a K, e K U ~K é o alfabeto de entrada, para qualquer conjunto K pertencente a esse alfabeto. As transições habilitadas apenas por (E,..,E) têm prioridade sobre as demais, por indicar que as fitas terminaram, resolvendo um problema de não-determinismo que ocorre em quase todos os estados.

Quando a cadeia de entrada possuir um elemento que faça parte do conjunto de símbolos de entrada Xc(i), a transição indicada por esse conjunto estará habilitada. Os conjuntos Xc(i) são alterados dinamicamente através das ações adaptativas Bx e Ux, isto é, essas ações inserem e removem transições ao colocar (ou retirar) um símbolo de Xc. Por exemplo, se o símbolo (0,...,0) estiver em Xc(0) então há uma transição do estado 0 para si mesmo, a partir dessa entrada; e ao retirar o símbolo do conjunto Xc(0) essa transição deixa de existir e passa a existir uma do estado 0 para o estado T, com a mesma entrada, caracterizando a adaptatividade (NETO, 1994).

L e R representam o movimento do cursor das fitas para a esquerda e para a direita, respectivamente, baseando-se no fato de que a primeira nota da parte analisada da música está sempre na extremidade esquerda da fita e é sempre a primeira a ser apontada. Transições sem essas indicações não movem o cursor. A figura-2 a seguir mostra um esquema do autômato utilizado pelo algoritmo.

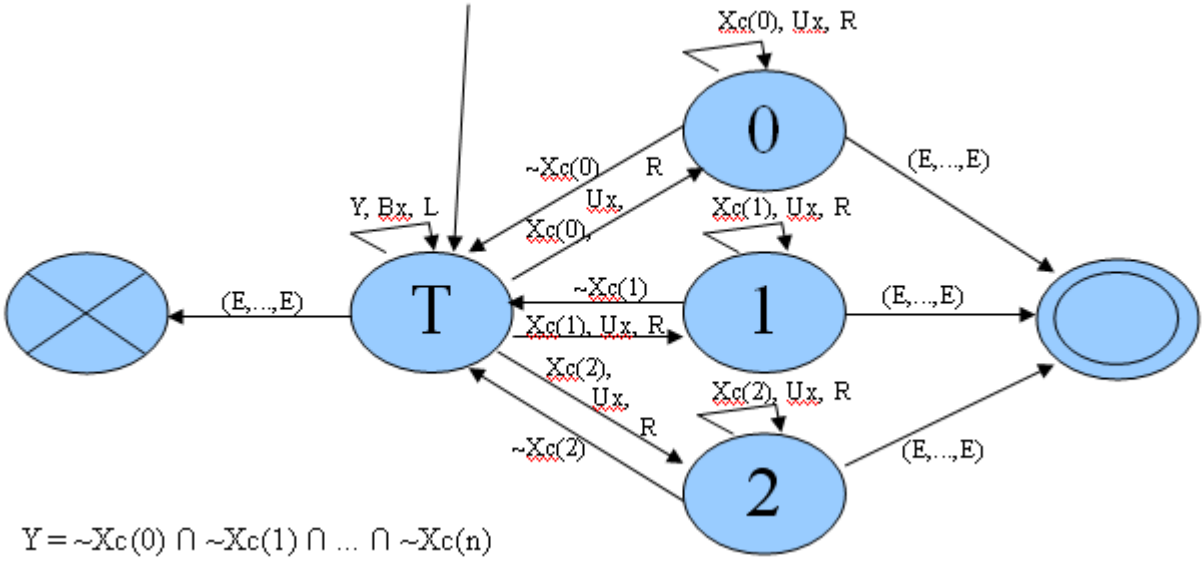


Figura-2: Esquema do autômato

## VI. Lógica do Autômato

O autômato procura manter-se em uma mesma corda pelo maior número possível de notas, realizando uma busca em profundidade na árvore implícita nas possibilidades de combinação de cordas. Um exemplo de execução do autômato pode ser imaginado, a partir dos estados, como sendo:

T→0→0→T→1→1→T→4→T→T→5→T→2→2→Aceitação

Esse processo indica que o autômato iniciou no estado T (sempre utilizado como estado inicial) e já partiu para o estado 0, deixando a nota inicial sobre a corda 0, deixou a nota seguinte na mesma corda e verificou que a nota seguinte não podia estar nessa corda. A partir disso foi ao estado de troca e percebeu que poderia utilizar a corda 1 para essa nota que não podia estar na corda 0, e logo a utilizou, indo ao estado 1. A nota seguinte não apresentou problemas em estar nessa mesma corda, porém seguiu-se uma nota que não podia estar na corda 1, mas podia estar na corda 4 e 5. A escolha inicial foi a corda 4 e o autômato descobre que essa corda é inválida (a transição de T para T é a única que volta o cursor). Escolhendo a corda 5 o autômato continuou a busca normalmente até a aceitação.

Logo Xc(i) representa o conjunto de símbolos que são válidos para o autômato continuar a busca, se aprofundando na i-ésima corda. Para simplificar a visualização, não é necessário visualizar sempre cada n-upla (ou símbolo) de Xc(i) por completo, apenas o i-ésimo elemento da n-upla, ou seja, o número que representa a posição da nota na i-ésima corda. O conjunto Xc(i) conterá esse símbolo se, e somente se, esse número estiver na faixa [`MinX`;`MaxX`]. Dessa forma com apenas dois números inteiros é possível definir todos os conjuntos Xc(i).

Seja a função X(i) que retorna um valor booleano para a i-ésima corda. O significado de X(i) é: se Xc(i) contém a entrada apontada pelo cursor, então retorna verdadeiro, caso contrário retorna falso. Dessa forma X(i) serve como teste para verificar qual é o próximo estado de maneira mais ágil que a verificação individual com cada símbolo de Xc(i). Para realizar isso, X(i) apenas verifica se o valor da posição na i-ésima corda está na faixa [`MinX`;`MaxX`].

Os valores iniciais de `MinX` e `MaxX` a cada execução do autômato devem ser, obviamente, o limite inferior e superior da faixa de posições permitida na tablatura, respectivamente.

Há uma indeterminação no estado T, pois X(i) pode ser válido para mais de um valor de i. Nesse caso a varredura é feita do menor valor de i até o maior valor de i, conforme exemplificado. Isso justifica que sempre o segundo estado é o 0.

As ações adaptativas Ux (*Update* X) e Bx (*Backup* X) servem basicamente para atualizar os valores de MinX e `MaxX`. Dessa forma os conjuntos Xc(i) serão alterados, e conseqüentemente a função X(i). A maneira como `MinX` e `MaxX` são alterados é descrita adiante.

Cada transição com R atribui o número da corda como saída, mas cada transição com L indica que essa saída era incorreta, "retirando" esse número da saída. Isso pode ser feito imaginando-se uma fita adicional "de saída", com números que indicam a fita escolhida para aquela nota. A tabela-2 abaixo apresenta os números na fita de saída.

Tabela-2: Números da fita de saída

| Afinação / Melodia | A3 | C4 | D4 | E4 | F4 | E4 | D4 |
|---|---|---|---|---|---|---|---|
| G3 (Fita 0) | 2 | 5 | 7 | 9 | 10 | 9 | 7 |
| D3 (Fita 1) | 7 | 10 | 12 | 14 | 15 | 14 | 12 |
| A2 (Fita 2) | 12 | 15 | 17 | 19 | 20 | 19 | 17 |
| E2 (Fita 3) | 17 | 20 | 22 | 24 | 25 | 24 | 22 |
| Saída | 2 | 1 | 1 | 0 | 0 | 0 | 1 |

## VII. Algoritmo

Todo o núcleo do funcionamento desse algoritmo está nas ações adaptativas que atualizam o `MinX` e o `MaxX`. Essas ações

visam a obedecer ao critério previamente estabelecido de que as distâncias entre as posições devem ser as menores possíveis.

Diferenças de três posições na tablatura em um trecho de poucas notas serão consideradas sempre possíveis de serem executadas, pois supondo que o músico tenha quatro dedos, todas as notas são possíveis de serem mapeadas em um dedo a partir do número da posição, e, portanto, esse é um bom número inicial para o algoritmo.

O algoritmo basicamente repete a execução do autômato iterativamente para cada valor de faixa de distâncias (ou diferença de posições). O valor é incrementado a cada iteração. O algoritmo, visto em alto nível, em uma pseudo-linguagem:

```
Repetir para cada melodia
 Faixa de distâncias := 3
  Enquanto (Executa autômato) não retorna aceitação
   Faixa de distâncias := Faixa de distâncias + 1
  Impõe fita saída: cordas utilizadas para a melodia
 Até acabar a música
```

Essa faixa de distâncias é importante para o autômato, pois é o menor valor que `MaxX – MinX` pode assumir, ou seja, é o menor tamanho do intervalo `[MinX;MaxX]`.

## VIII. AÇÕES ADAPTATIVAS

Nem sempre é necessário fazer a melodia inteira ficar na faixa de distâncias descrita. Por exemplo, se há uma melodia com vinte notas, a primeira nota não precisa estar na mesma região que a vigésima, ou seja, não é interessante forçar que a melodia inteira esteja sempre dentro da faixa de distâncias.

Seja W a janela que contém o conjunto de valores numéricos das últimas posições utilizadas como saída (e.g. no exemplo da Greensleaves seria 12, 10, 12, 9, 10, 9, 12). O tamanho dessa janela indica quantos números serão utilizados. Para isso é necessário uma pilha que armazene todos os valores já utilizados. Seja `fretStack` essa pilha. A janela sempre se resume a indicar os elementos do topo dessa pilha.

O tamanho da janela não pode ser demasiado pequeno, pois tornaria irrelevante a importância da faixa de distâncias. Se o tamanho da janela for menor do que 4, praticamente qualquer entrada tornar-se-á válida, em particular a escala cromática ficaria toda na mesma corda (0,1,2,3,4,5,6,...). Se o tamanho da janela for quatro, esse valor continua pequeno demais para garantir que a digitação será boa caso haja repetição de números, como a seqüência (0,1,1,2,3,3,4,5,5). No caso em que há uma repetição de todas notas, comum na presença do chamado "tremolo picking"(TURNER and WHITE, 1988), o primeiro valor a garantir a separação para outra corda é 7, pois (0,0,1,1,2,2,3,3) tem oito elementos (portanto o primeiro zero é descartado) e o elemento seguinte não poderá ser 4 caso a faixa de distância seja a inicial, 3. Valores maiores são um exagero na maior parte dos casos, pois essa janela de tamanho N obriga que qualquer conjunto de N notas adjacentes esteja dentro da faixa de distâncias. Sete notas é o suficiente para haver uma mudança de contexto e um pequeno deslocamento na região da melodia. Dentro da janela há um valor máximo e mínimo, sejam esses `MaxW` e `MinW`, respectivamente.

A ação Ux inicialmente empilha o novo valor de posição em `fretStack` para garantir que esse valor não será perdido e esteja na janela quando necessário. A ação Bx faz o inverso, descartando o valor do topo dessa pilha.

As ações Ux e Bx atualizam as variáveis `MinX` e `MaxX` e, conseqüentemente, a função X(i), indicando nova a faixa de valores permitidos para a posição da nota. Essa faixa pode ser facilmente calculada tendo todos os valores da janela de posições usadas, bastando possuir o valor máximo e mínimo dentro da janela. O valor de `MinX` e `MaxX` passam a ser os limites da janela, acrescidos de quanto falta para completar a faixa de distância em ambos os sentidos (diminuir `MinW` e aumentar `MaxW`), de forma que garanta que o próximo valor, estando nessa faixa, é incapaz de tornar a nova amplitude da janela (`MaxW - MinW`) maior que a faixa de distâncias:

```
MinX = MinW – FaixaDistâncias + (MaxW – MinW)
MaxX = MaxW + FaixaDistâncias – (MaxW – MinW)
```

Obviamente para uma janela vazia, `MinX` e `MaxX` serão iguais aos limites da faixa de valores permitidos para as posições, e após a aplicação das fórmula acima deve-se sempre saturar os valores de tais variáveis nessa faixa de valores permitidos.

## IX. CONCLUSÃO

O algoritmo funciona bem para melodias, conjunto de notas sucessivas e totalmente sem simultaneidade sonora, sem pausas muito curtas (pausas são consideradas curtas quando têm duração inferior a uma colcheia). Entretanto, se houver grandes discrepâncias de altura (três oitavas, por exemplo) o algoritmo é lento. É interessante limitá-lo a poucos trastes quando isso for possível.

Embora não haja ganhos consideráveis de *performance* ou até mesmo de resultado utilizando o algoritmo aqui descrito na versão inicial em relação aos algoritmos convencionais, outros pontos podem ser explorados para melhorá-lo, como por exemplo o tratamento de notas múltiplas (bicordes e acordes) ou o tratamento de ornamentos existentes. Com tais melhorias, esperamos obter resultados melhores e mais abrangentes.

## REFERÊNCIAS


[1] J. Junior, *O básico de guitarra*, Guitar X. Acesso em 17/12/2007. Disponível em: http://www.guitarx.com.br/index.asp?url=library_html/basico_guitarra/basico_da_guitarra%206_10.htm.

[2] R. Bennett, *Uma breve história da música*, Ed. Jorge Zahar Editor Ltda, 1996, 80p.

[3] F. Azevedo, *Método para Guitarra*, Ed. Bruno Quaino Editores, 1978, 16p.

[4] J. J. Neto, *Adaptive Automata for Context -Sensitive Language*. SIGPLAN NOTICES, Vol. 29, n. 9, pp. 115-124, September, 1994.

[5] M. D. Fagundes, *Teoria da Música*, Ed. Keyboard, 2004, 183p.

[6] W. Leavitt, *Método Moderno para Guitarra*, 1a Ed., Vol. 1, Ed. Irmãos Vitale, 2007, 128p.

[7] B. Med, *Teoria da Música*, 4a ed., Ed. Musimed, 2001, 420p.

[8] G. Turner, B. White, *Progressive Lead Guitar*, USA, Ed. Koala Publications, 1988, 94p.

[9] H. Pistori. *Tecnologia Adaptativa em Engenharia de Computacão: Estad. da Arte e Aplicações*. Tese (Doutorado) — Universidade de São Paulo.