Técnicas de aprimoramento de um sistema de acompanhamento musical

Roberto Piassi Passos Bodo

Instituto de Matemática e Estatística
Universidade de São Paulo

12 de Novembro de 2013

Acompanhamento musical

Acompanhamento musical

- escutar a performance de um músico;

Acompanhamento musical

- escutar a performance de um músico;
- comparar os eventos da entrada com os eventos de uma partitura;

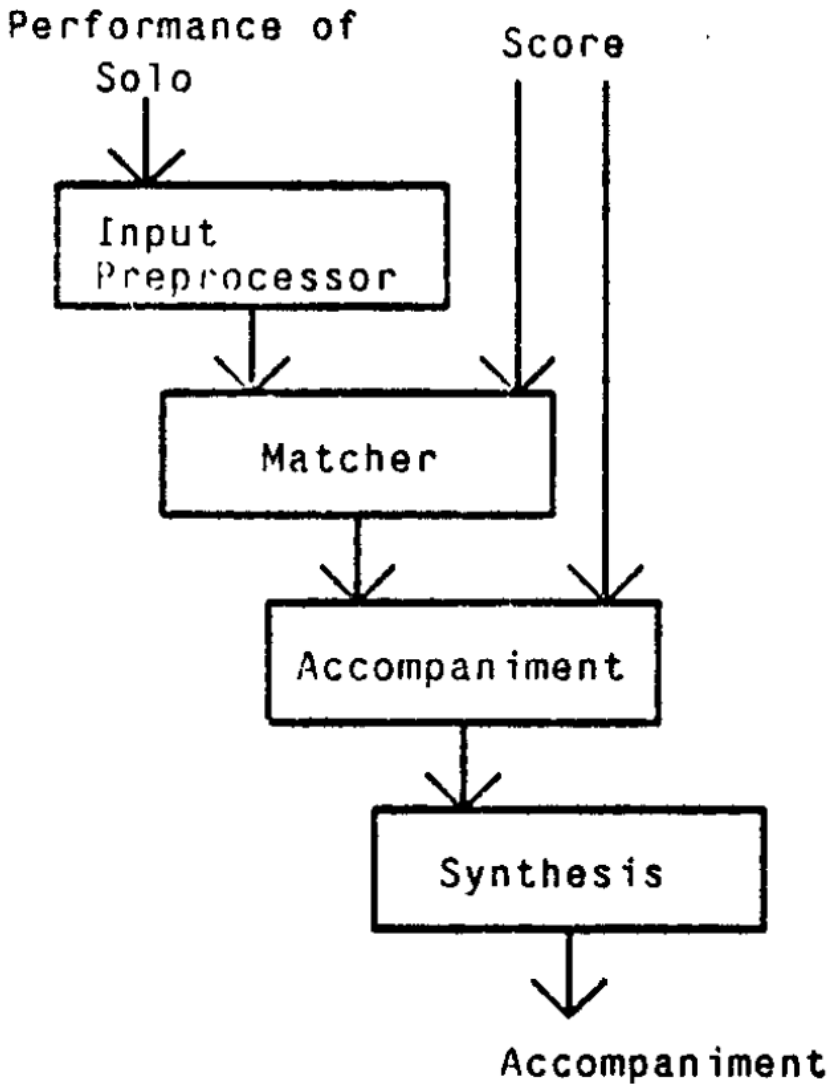
Acompanhamento musical

- escutar a performance de um músico;
- comparar os eventos da entrada com os eventos de uma partitura;
- inferir um andamento;

Acompanhamento musical

- escutar a performance de um músico;
- comparar os eventos da entrada com os eventos de uma partitura;
- inferir um andamento;
- tocar o acompanhamento apropriado.

Sistema de acompanhamento musical



Casamento monofônico

Características dos eventos:

Casamento monofônico

Características dos eventos:

- são trivialmente comparados;

Casamento monofônico

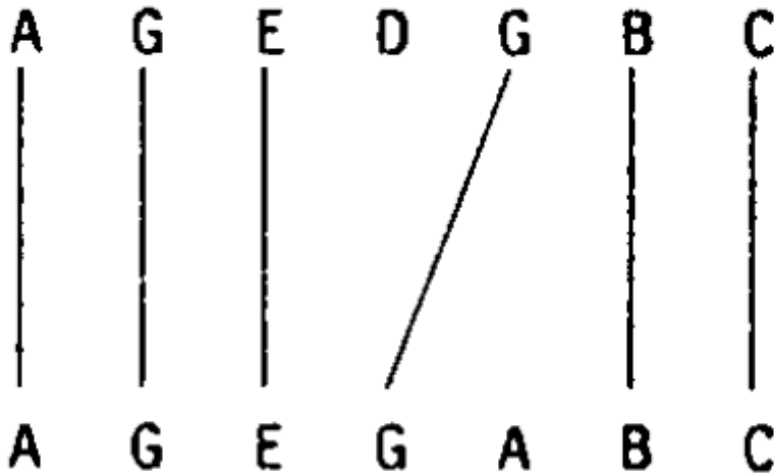
Características dos eventos:

- são trivialmente comparados;
- são totalmente ordenados.

Casamento monofônico

Características dos eventos:

- são trivialmente comparados;
- são totalmente ordenados.



Casamento polifônico

Características dos eventos:

Casamento polifônico

Características dos eventos:

- são mais difíceis de comparar;

Casamento polifônico

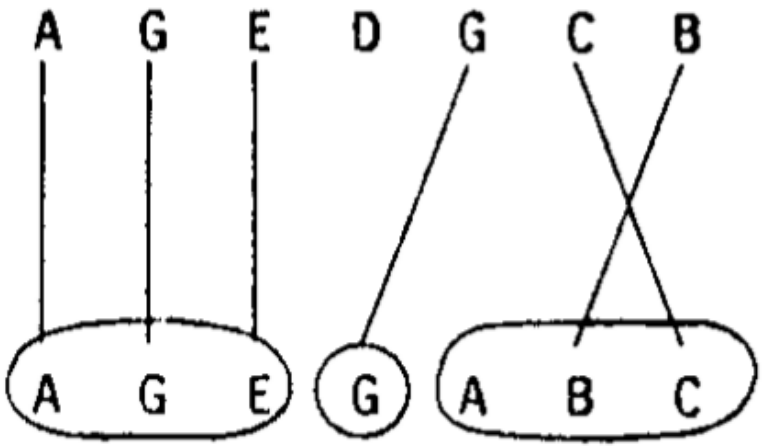
Características dos eventos:

- são mais difíceis de comparar;
- são parcialmente ordenados.

Casamento polifônico

Características dos eventos:

- são mais difíceis de comparar;
- são parcialmente ordenados.



Eventos compostos

Para cada nota recebida da performance:

Eventos compostos

Para cada nota recebida da performance:

- se o intervalo de tempo em relação à nota anterior for menor do que epsilon (100ms, por exemplo), adicionamos no mesmo evento composto;

Eventos compostos

Para cada nota recebida da performance:

- se o intervalo de tempo em relação à nota anterior for menor do que epsilon (100ms, por exemplo), adicionamos no mesmo evento composto;
- caso contrário, criamos um novo evento composto.

Eventos compostos

Para cada nota recebida da performance:

- se o intervalo de tempo em relação à nota anterior for menor do que epsilon (100ms, por exemplo), adicionamos no mesmo evento composto;
- caso contrário, criamos um novo evento composto.

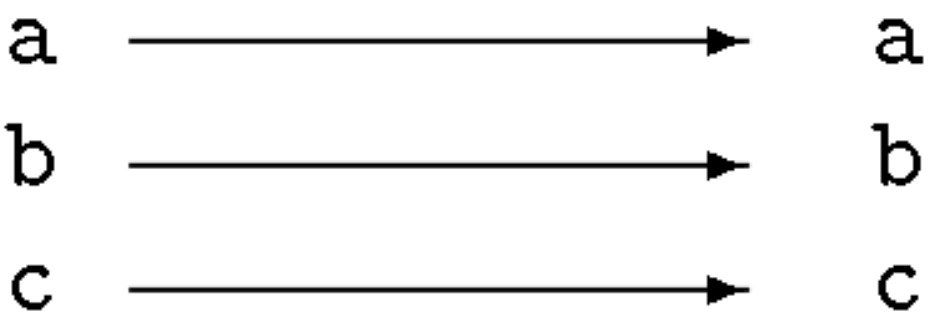
Com isso, reduzimos o problema do acompanhamento polifônico para o monofônico.

1. Ornamentos

Em um cenário ideal, temos uma bijeção entre a performance e a partitura.

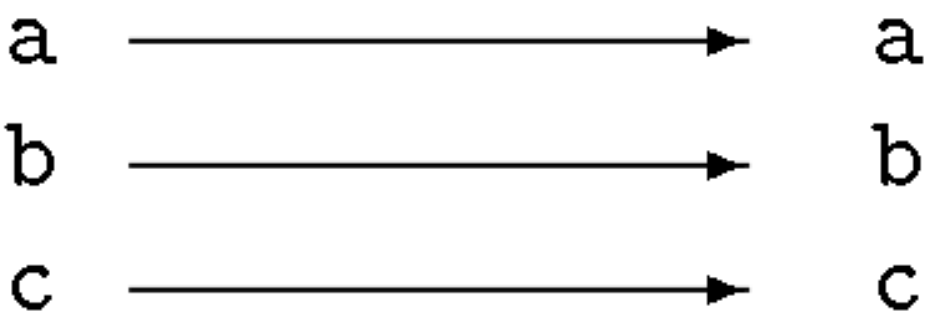
1. Ornamentos

Em um cenário ideal, temos uma bijeção entre a performance e a partitura.



1. Ornamentos

Em um cenário ideal, temos uma bijeção entre a performance e a partitura.



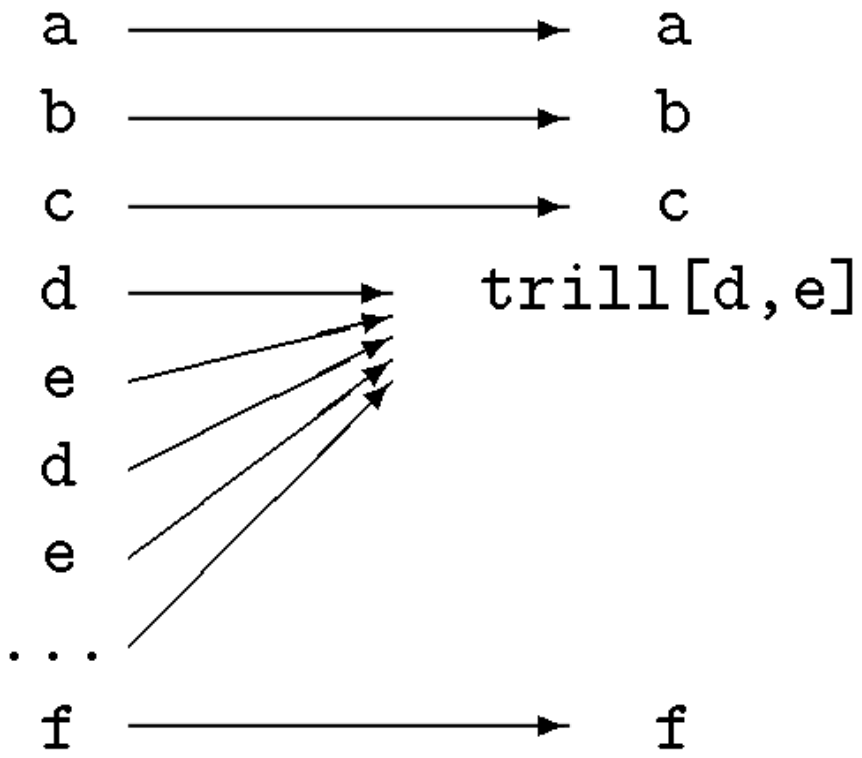
Porém, com alguns ornamentos nem sempre conseguimos fazer esse mapeamento.

1. Ornamentos

Problema:
Em um trinado, por exemplo, não sabemos quantas notas o músico irá tocar.

1. Ornamentos

Problema:
Em um trinado, por exemplo, não sabemos quantas notas o músico irá tocar.



1. Ornamentos

Solução sugerida:

1. Ornamentos

Solução sugerida:

- Input Preprocessor será agora um autômato finito com dois estados;

1. Ornamentos

Solução sugerida:

- Input Preprocessor será agora um autômato finito com dois estados;
  - o primeiro (inicial) terá o comportamento padrão;

1. Ornamentos

Solução sugerida:

- Input Preprocessor será agora um autômato finito com dois estados;
  - o primeiro (inicial) terá o comportamento padrão;
  - o segundo irá lidar ornamentos.

1. Ornamentos

Solução sugerida:

- Input Preprocessor será agora um autômato finito com dois estados;
  - o primeiro (inicial) terá o comportamento padrão;
  - o segundo irá lidar ornamentos.
- o Matcher relata um ornamento na partitura para o Input Preprocessor;

1. Ornamentos

Solução sugerida:

- Input Preprocessor será agora um autômato finito com dois estados;
  - o primeiro (inicial) terá o comportamento padrão;
  - o segundo irá lidar ornamentos.
- o Matcher relata um ornamento na partitura para o Input Preprocessor;
- o Input Preprocessor sai do estado inicial e só volta quando:

1. Ornamentos

Solução sugerida:

- Input Preprocessor será agora um autômato finito com dois estados;
  - o primeiro (inicial) terá o comportamento padrão;
  - o segundo irá lidar ornamentos.
- o Matcher relata um ornamento na partitura para o Input Preprocessor;
- o Input Preprocessor sai do estado inicial e só volta quando:
  - detecta uma nota mais longa;

1. Ornamentos

Solução sugerida:

- Input Preprocessor será agora um autômato finito com dois estados;
  - o primeiro (inicial) terá o comportamento padrão;
  - o segundo irá lidar ornamentos.
- o Matcher relata um ornamento na partitura para o Input Preprocessor;
- o Input Preprocessor sai do estado inicial e só volta quando:
  - detecta uma nota mais longa;
  - recebe a nota esperada da partitura após o ornamento;

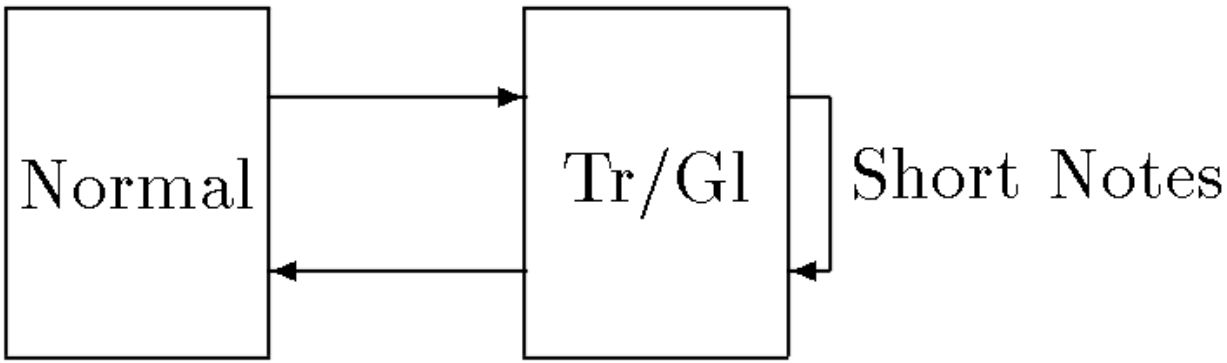
1. Ornamentos

Solução sugerida:

- Input Preprocessor será agora um autômato finito com dois estados;
  - o primeiro (inicial) terá o comportamento padrão;
  - o segundo irá lidar ornamentos.
- o Matcher relata um ornamento na partitura para o Input Preprocessor;
- o Input Preprocessor sai do estado inicial e só volta quando:
  - detecta uma nota mais longa;
  - recebe a nota esperada da partitura após o ornamento;
  - ultrapassa um tempo limite.

1. Ornamentos

Reading Special Symbol



Long Note `or`
Time Out `or`
Expected Note

2. Equivalência de oitavas

Motivação:

2. Equivalência de oitavas

Motivação:

- o solista precisa tocar as notas da partitura de forma mais regrada para uma boa evolução do acompanhamento;

2. Equivalência de oitavas

Motivação:

- o solista precisa tocar as notas da partitura de forma mais regrada para uma boa evolução do acompanhamento;
- algum músico pode desejar tocar de forma mais livre em seu instrumento;

2. Equivalência de oitavas

Motivação:

- o solista precisa tocar as notas da partitura de forma mais regrada para uma boa evolução do acompanhamento;
- algum músico pode desejar tocar de forma mais livre em seu instrumento;
- também pode querer fazer acordes abertos quando não especificados.

2. Equivalência de oitavas

Solução sugerida:

2. Equivalência de oitavas

Solução sugerida:

- utilizar um vetor de bits para representar um evento composto;

2. Equivalência de oitavas

Solução sugerida:

- utilizar um vetor de bits para representar um evento composto;
- o bit menos significativo corresponde a C, o próximo bit corresponde a C# e assim por diante;

2. Equivalência de oitavas

Solução sugerida:

- utilizar um vetor de bits para representar um evento composto;
- o bit menos significativo corresponde a C, o próximo bit corresponde a C# e assim por diante;
- pitch_class = midi_number % 12.

2. Equivalência de oitavas

Solução sugerida:

- utilizar um vetor de bits para representar um evento composto;
- o bit menos significativo corresponde a C, o próximo bit corresponde a C# e assim por diante;
- pitch_class = midi_number % 12.

Vantagens extras:

2. Equivalência de oitavas

Solução sugerida:

- utilizar um vetor de bits para representar um evento composto;
- o bit menos significativo corresponde a C, o próximo bit corresponde a C# e assim por diante;
- pitch_class = midi_number % 12.

Vantagens extras:

- menor consumo de memória (eventos de 16 para 2 bytes);

2. Equivalência de oitavas

Solução sugerida:

- utilizar um vetor de bits para representar um evento composto;
- o bit menos significativo corresponde a C, o próximo bit corresponde a C# e assim por diante;
- pitch_class = midi_number % 12.

Vantagens extras:

- menor consumo de memória (eventos de 16 para 2 bytes);
- melhor performance (operações *bitwise*).

2. Equivalência de oitavas

```
(performance) 0010 1010 0001
(score)       0010 1001 0001
                              EOR
(result1)     0000 0011 0000
(score)       0010 1001 0001
                              AND
(result2)     0000 0001 0000
```

3. Paralelismo no Matcher

O Matcher considera apenas um subconjunto da partitura.

3. Paralelismo no Matcher

O Matcher considera apenas um subconjunto da partitura.

Problemas:

3. Paralelismo no Matcher

O Matcher considera apenas um subconjunto da partitura.

Problemas:

- o músico pode avançar para fora da janela.

3. Paralelismo no Matcher

O Matcher considera apenas um subconjunto da partitura.

Problemas:

- o músico pode avançar para fora da janela.
  - ele pulou uma parte da partitura?

3. Paralelismo no Matcher

O Matcher considera apenas um subconjunto da partitura.

Problemas:

- o músico pode avançar para fora da janela.
  - ele pulou uma parte da partitura?
  - temos um casamento errado devido a uma nota extra?

3. Paralelismo no Matcher

O Matcher considera apenas um subconjunto da partitura.

Problemas:

- o músico pode avançar para fora da janela.
  - ele pulou uma parte da partitura?
  - temos um casamento errado devido a uma nota extra?
  - devemos aumentar o tamanho da janela?

3. Paralelismo no Matcher

O Matcher considera apenas um subconjunto da partitura.

Problemas:

- o músico pode avançar para fora da janela.
  - ele pulou uma parte da partitura?
  - temos um casamento errado devido a uma nota extra?
  - devemos aumentar o tamanho da janela?
- o músico pode parar de tocar;

3. Paralelismo no Matcher

O Matcher considera apenas um subconjunto da partitura.

Problemas:

- o músico pode avançar para fora da janela.
  - ele pulou uma parte da partitura?
  - temos um casamento errado devido a uma nota extra?
  - devemos aumentar o tamanho da janela?
- o músico pode parar de tocar;
  - ele vai voltar no ponto de onde parou?

3. Paralelismo no Matcher

O Matcher considera apenas um subconjunto da partitura.

Problemas:

- o músico pode avançar para fora da janela.
  - ele pulou uma parte da partitura?
  - temos um casamento errado devido a uma nota extra?
  - devemos aumentar o tamanho da janela?
- o músico pode parar de tocar;
  - ele vai voltar no ponto de onde parou?
  - ele vai voltar no ponto onde o Accompanist chegou?

3. Paralelismo no Matcher

Solução sugerida:

3. Paralelismo no Matcher

Solução sugerida:

- transformar o Matcher em um objeto;

3. Paralelismo no Matcher

Solução sugerida:

- transformar o Matcher em um objeto;
- criar uma nova instância quando tivermos dúvidas de onde o músico está;

3. Paralelismo no Matcher

Solução sugerida:

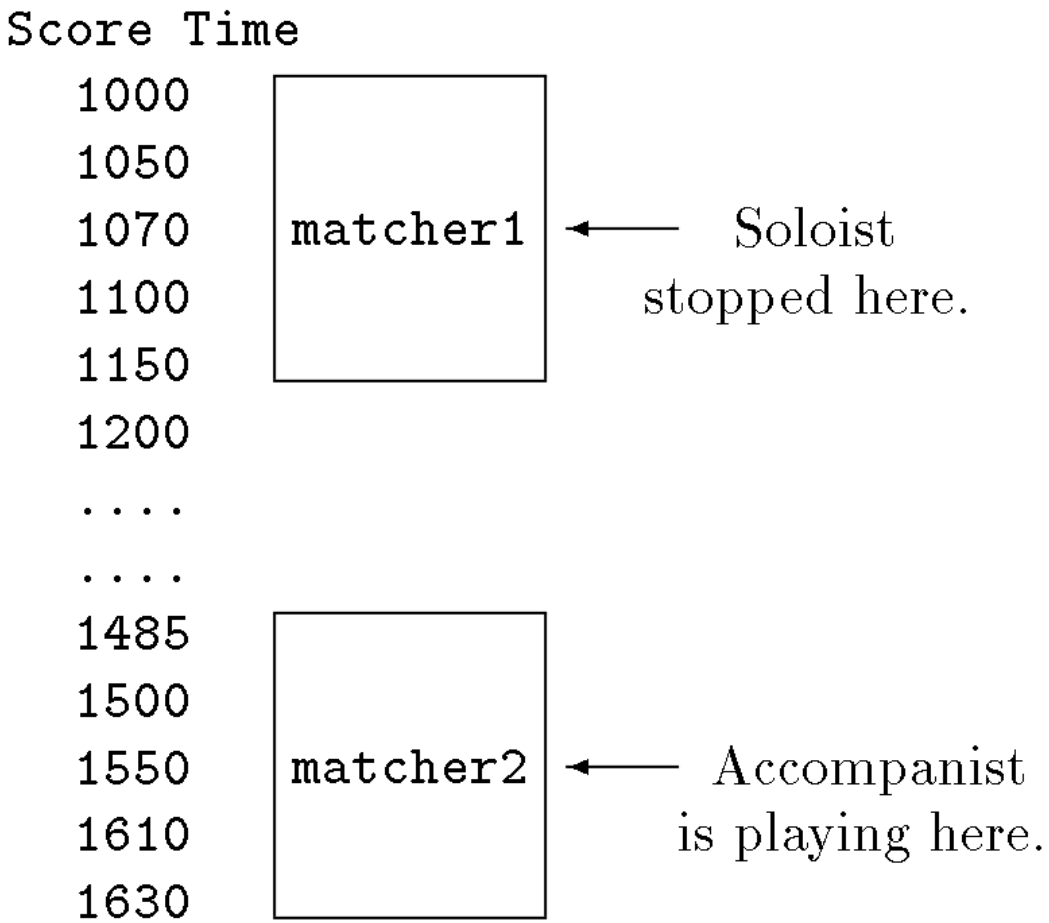
- transformar o Matcher em um objeto;
- criar uma nova instância quando tivermos dúvidas de onde o músico está;
- centralizamos cada janela nas posições alternativas;

3. Paralelismo no Matcher

Solução sugerida:

- transformar o Matcher em um objeto;
- criar uma nova instância quando tivermos dúvidas de onde o músico está;
- centralizamos cada janela nas posições alternativas;
- descartamos um dos Matchers quando o outro encontra um casamento confiável.

3. Paralelismo no Matcher



4. Accompanist

Nas implementações anteriores:

4. Accompanist

Nas implementações anteriores:

- a partitura é uma sequência de eventos *note-on/note-off*;

4. Accompanist

Nas implementações anteriores:

- a partitura é uma sequência de eventos *note-on/note-off*;
- na ausência de casamentos o Accompanist toca cada evento quando seu tempo for alcançado;

4. Accompanist

Nas implementações anteriores:

- a partitura é uma sequência de eventos *note-on/note-off*;
- na ausência de casamentos o Accompanist toca cada evento quando seu tempo for alcançado;
- mantemos um ponteiro na partitura para saber até onde foi tocada;

4. Accompanist

Possibilidades:

4. Accompanist

Possibilidades:

- o Accompanist sabe onde o músico está na partitura;

4. Accompanist

Possibilidades:

- o Accompanist sabe onde o músico está na partitura;
  - ele está adiantado em relação ao músico?

4. Accompanist

Possibilidades:

- o Accompanist sabe onde o músico está na partitura;
  - ele está adiantado em relação ao músico?
  - ele está atrasado em relação ao músico?

4. Accompanist

Possibilidades:

- o Accompanist sabe onde o músico está na partitura;
  - ele está adiantado em relação ao músico?
  - ele está atrasado em relação ao músico?
- o Accompanist descobre que a posição do solista estava errada.

4. Accompanist

Possibilidades:

- o Accompanist sabe onde o músico está na partitura;
  - ele está adiantado em relação ao músico?
  - ele está atrasado em relação ao músico?
- o Accompanist descobre que a posição do solista estava errada.
  - ele deve pular na partitura para o ponto correto?

4. Accompanist

Possibilidades:

- o Accompanist sabe onde o músico está na partitura;
  - ele está adiantado em relação ao músico?
  - ele está atrasado em relação ao músico?
- o Accompanist descobre que a posição do solista estava errada.
  - ele deve pular na partitura para o ponto correto?
  - o que fazer com notas que já estavam soando?

4. Accompanist

Possibilidades:

- o Accompanist sabe onde o músico está na partitura;
  - ele está adiantado em relação ao músico?
  - ele está atrasado em relação ao músico?
- o Accompanist descobre que a posição do solista estava errada.
  - ele deve pular na partitura para o ponto correto?
  - o que fazer com notas que já estavam soando?
  - o que fazer com notas que deveriam estar soando?

4. Accompanist

Algoritmo mais robusto:

4. Accompanist

Algoritmo mais robusto:

- o Accompanist recebe 3 informações do Matcher.

4. Accompanist

Algoritmo mais robusto:

- o Accompanist recebe 3 informações do Matcher.
  - o tempo virtual do evento casado;

4. Accompanist

Algoritmo mais robusto:

- o Accompanist recebe 3 informações do Matcher.
  - o tempo virtual do evento casado;
  - uma indicação se o evento passado foi o próximo da sequência em relação ao último casado;

4. Accompanist

Algoritmo mais robusto:

- o Accompanist recebe 3 informações do Matcher.
  - o tempo virtual do evento casado;
  - uma indicação se o evento passado foi o próximo da sequência em relação ao último casado;
  - o tempo virtual do próximo evento da partitura.

4. Accompanist

Algoritmo mais robusto:
A cada relato de casamento, o Accompanist chama duas funções:

4. Accompanist

Algoritmo mais robusto:
A cada relato de casamento, o Accompanist chama duas funções:

- *change_virtual_time()*;
  calcula o próximo valor do relógio virtual e muda as notas que estão soando

4. Accompanist

Algoritmo mais robusto:
A cada relato de casamento, o Accompanist chama duas funções:

- *change_virtual_time()*;
  calcula o próximo valor do relógio virtual e muda as notas que estão soando
- *adjust_clock_speed()*.
  muda o andamento do acompanhamento para refletir o da performance solo

4. Accompanist: *change_virtual_time()*

Primeira decisão:

4. Accompanist: *change_virtual_time()*

Primeira decisão:

if (*delta_virtual_time* $<$ *min_delta*){ // 100ms, por exemplo
    // a mudança no relógio virtual é descartada.
}

4. Accompanist: *change_virtual_time()*

Primeira decisão:

if (*delta_virtual_time* $<$ *min_delta*){ // 100ms, por exemplo
    // a mudança no relógio virtual é descartada.
}

Observações:

4. Accompanist: *change_virtual_time()*

Primeira decisão:

if (*delta_virtual_time* $<$ *min_delta*){ // 100ms, por exemplo
    // a mudança no relógio virtual é descartada.
}

Observações:

- resulta em uma leitura mais suave do acompanhamento;

4. Accompanist: *change_virtual_time()*

Primeira decisão:

if (*delta_virtual_time* $<$ *min_delta*){ // 100ms, por exemplo
    // a mudança no relógio virtual é descartada.
}

Observações:

- resulta em uma leitura mais suave do acompanhamento;
- se o solista quiser realmente mudar de andamento a magnitude será muito maior.

4. Accompanist: *change_virtual_time()*

Quando a mudança de tempo solicitada é realmente processada temos 2 possibilidades:

4. Accompanist: *change_virtual_time()*

Quando a mudança de tempo solicitada é realmente processada temos 2 possibilidades:

1. o casamento está na sequência;

4. Accompanist: *change_virtual_time()*

Quando a mudança de tempo solicitada é realmente processada temos 2 possibilidades:

1. o casamento está na sequência;
   o Accompanist adquire mais confiança em relação à posição do músico na partitura

4. Accompanist: *change_virtual_time()*

Quando a mudança de tempo solicitada é realmente processada temos 2 possibilidades:

1. o casamento está na sequência;
   o Accompanist adquire mais confiança em relação à posição do músico na partitura
2. o casamento não está na sequência.

4. Accompanist: *change_virtual_time()*

Quando a mudança de tempo solicitada é realmente processada temos 2 possibilidades:

1. o casamento está na sequência;
   o Accompanist adquire mais confiança em relação à posição do músico na partitura
2. o casamento não está na sequência.
   a noção que o Accompanist tinha da posição do solista pode estar errada

4. Accompanist: *change_virtual_time()*

Quando o casamento está na sequência temos 2 casos:

4. Accompanist: *change_virtual_time()*

Quando o casamento está na sequência temos 2 casos:

1. o Accompanist está atrasado;

4. Accompanist: *change_virtual_time()*

Quando o casamento está na sequência temos 2 casos:

1. o Accompanist está atrasado;
   rapidamente toca os eventos pendentes para alcançar o músico

4. Accompanist: *change_virtual_time()*

Quando o casamento está na sequência temos 2 casos:

1. o Accompanist está atrasado;
   rapidamente toca os eventos pendentes para alcançar o músico
2. o Accompanist está adiantado.

4. Accompanist: *change_virtual_time()*

Quando o casamento está na sequência temos 2 casos:

1. o Accompanist está atrasado;
   rapidamente toca os eventos pendentes para alcançar o músico
2. o Accompanist está adiantado.
   continua a tocar as notas que já começou (respeitando as durações) e espera o solista alcançar o acompanhamento naquele ponto

4. Accompanist: *change_virtual_time()*

A heurística "na sequência"apresenta um problema quando o músico faz uma mudança drástica no tempo.

4. Accompanist: *change_virtual_time()*

A heurística "na sequência"apresenta um problema quando o músico faz uma mudança drástica no tempo.

Verificação extra:
if (*delta_virtual_time* $>$ *max_delta*){ // 2s, por exemplo
    // Accompanist pula diretamente para o novo ponto
}

4. Accompanist: *change_virtual_time()*

Quando o casamento não está na sequência:

4. Accompanist: *change_virtual_time()*

Quando o casamento não está na sequência:

- sincronizamos com o músico imediatamente;

4. Accompanist: *change_virtual_time()*

Quando o casamento não está na sequência:

- sincronizamos com o músico imediatamente;
- tocamos as notas que deveriam estar soando naquele ponto da partitura;

4. Accompanist: *change_virtual_time()*

Quando o casamento não está na sequência:

- sincronizamos com o músico imediatamente;
- tocamos as notas que deveriam estar soando naquele ponto da partitura;
  - pré-processar a partitura para calcular quantas notas deveriam estar soando para cada evento da partitura;

4. Accompanist: *change_virtual_time()*

Quando o casamento não está na sequência:

- sincronizamos com o músico imediatamente;
- tocamos as notas que deveriam estar soando naquele ponto da partitura;
  - pré-processar a partitura para calcular quantas notas deveriam estar soando para cada evento da partitura;
  - ler a partitura de trás para frente a partir do tempo virtual atual e interpretar n eventos note-on.

4. Accompanist: *change_virtual_time()*

Quando o casamento não está na sequência:

- sincronizamos com o músico imediatamente;
- tocamos as notas que deveriam estar soando naquele ponto da partitura;
  - pré-processar a partitura para calcular quantas notas deveriam estar soando para cada evento da partitura;
  - ler a partitura de trás para frente a partir do tempo virtual atual e interpretar n eventos note-on.
- ajustar o ponteiro de evolução do acompanhamento.

4. Accompanist: *adjust_clock_speed()*

Tarefa: manter o valor da velocidade do relógio virtual.

4. Accompanist: *adjust_clock_speed()*

Tarefa: manter o valor da velocidade do relógio virtual.

- v = velocidade na qual o acompanhamento está sendo tocado (em relação ao que está definido na partitura)

4. Accompanist: *adjust_clock_speed()*

Tarefa: manter o valor da velocidade do relógio virtual.

- v = velocidade na qual o acompanhamento está sendo tocado (em relação ao que está definido na partitura)
- v = velocidade na qual o solo está sendo tocado (em relação ao que está definido na partitura)

4. Accompanist: *adjust_clock_speed()*

Tarefa: manter o valor da velocidade do relógio virtual.

- v = velocidade na qual o acompanhamento está sendo tocado (em relação ao que está definido na partitura)

- v = velocidade na qual o solo está sendo tocado (em relação ao que está definido na partitura)

A cada novo relato do Matcher temos um novo ponto no gráfico de tempo real vs. virtual.

4. Accompanist: *adjust_clock_speed()*

Primeira abordagem: $v = \frac{deta\_tempo\_virtual}{deta\_tempo\_real}$

Resultado: respostas instantâneas na execução do andamento.

Problema: performances reais possuem variações locais de velocidade que não indicam variações no andamento.

4. Accompanist: *adjust_clock_speed()*

Segunda abordagem: interpolação dos últimos n pontos.

Resultado: execução mais suave do acompanhamento.

Problema (insignificante): custo computacional.

4. Accompanist: *adjust_clock_speed()*

Heurística 1:

4. Accompanist: *adjust_clock_speed()*

Heurística 1:
Quando ocorre um match fora de sequência, precisamos descartar a tabela atual.

4. Accompanist: *adjust_clock_speed()*

Heurística 1:
Quando ocorre um match fora de sequência, precisamos descartar a tabela atual.
Problema:

4. Accompanist: *adjust_clock_speed()*

Heurística 1:
Quando ocorre um match fora de sequência, precisamos descartar a tabela atual.
Problema:

- a velocidade não é ajustada enquanto a tabela não está cheia;

4. Accompanist: *adjust_clock_speed()*

Heurística 1:
Quando ocorre um match fora de sequência, precisamos descartar a tabela atual.
Problema:

- a velocidade não é ajustada enquanto a tabela não está cheia;
- a tabela pode demorar muito para ser preenchida;

4. Accompanist: *adjust_clock_speed()*

Heurística 1:
Quando ocorre um match fora de sequência, precisamos descartar a tabela atual.
Problema:

- a velocidade não é ajustada enquanto a tabela não está cheia;
- a tabela pode demorar muito para ser preenchida;
- o acompanhamento pode ultrapassar o músico.

4. Accompanist: *adjust_clock_speed()*

Heurística 1:
Quando ocorre um match fora de sequência, precisamos descartar a tabela atual.
Problema:

- a velocidade não é ajustada enquanto a tabela não está cheia;
- a tabela pode demorar muito para ser preenchida;
- o acompanhamento pode ultrapassar o músico.

Solução: iremos atualizar a velocidade se a diferença entre o primeiro e o último ponto for maior que um limite (1s, por exemplo).

4. Accompanist: *adjust_clock_speed()*

Heurística 2:

4. Accompanist: *adjust_clock_speed()*

Heurística 2:
Quando o músico toca muito rápido, as notas desse trecho não são boas indicadoras de andamento.

4. Accompanist: *adjust_clock_speed()*

Heurística 2:
Quando o músico toca muito rápido, as notas desse trecho não são boas indicadoras de andamento.
Solução: não inserimos um valor novo se a diferença entre ele e o último ponto for menor do que um limite (200ms, por exemplo).

4. Accompanist: *adjust_clock_speed()*

Heurística 2:
Quando o músico toca muito rápido, as notas desse trecho não são boas indicadoras de andamento.
Solução: não inserimos um valor novo se a diferença entre ele e o último ponto for menor do que um limite (200ms, por exemplo).

Heurística 3:

4. Accompanist: *adjust_clock_speed()*

Heurística 2:
Quando o músico toca muito rápido, as notas desse trecho não são boas indicadoras de andamento.
Solução: não inserimos um valor novo se a diferença entre ele e o último ponto for menor do que um limite (200ms, por exemplo).

Heurística 3:
Se o Macther não relatar mais nenhum casamento o Accompanist pode terminar o acompanhamento sozinho.

4. Accompanist: *adjust_clock_speed()*

Heurística 2:
Quando o músico toca muito rápido, as notas desse trecho não são boas indicadoras de andamento.
Solução: não inserimos um valor novo se a diferença entre ele e o último ponto for menor do que um limite (200ms, por exemplo).

Heurística 3:
Se o Macther não relatar mais nenhum casamento o Accompanist pode terminar o acompanhamento sozinho.
Solução ("detecção de fuga"): o Accompanist para de tocar depois de um período (2s, por exemplo) após o tempo do próximo evento esperado.

5. Atrasos nos relatos do Matcher

5. Atrasos nos relatos do Matcher

- o Accompanist tem alta confiança no Matcher;

5. Atrasos nos relatos do Matcher

- o Accompanist tem alta confiança no Matcher;
- o Matcher relata um casamento toda vez que um novo evento gera um melhor casamento geral;

5. Atrasos nos relatos do Matcher

- o Accompanist tem alta confiança no Matcher;
- o Matcher relata um casamento toda vez que um novo evento gera um melhor casamento geral;

Problema: o Input Preprocessor pode deixar passar alguns ornamentos gerando casamentos errados.

5. Atrasos nos relatos do Matcher

Solução sugerida:

5. Atrasos nos relatos do Matcher

Solução sugerida:

- o Matcher deve atrasar relatos de notas não consecutivas;

5. Atrasos nos relatos do Matcher

Solução sugerida:

- o Matcher deve atrasar relatos de notas não consecutivas;
- se a nota esperada aparecer (dentro de um tempo máximo permitido de atraso), o relato anterior é cancelado;

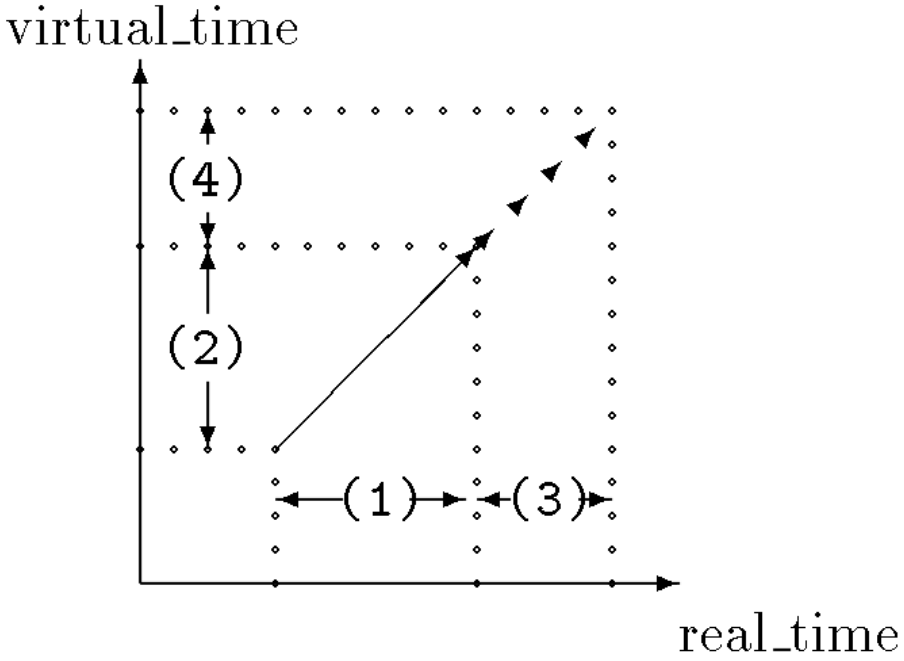
5. Atrasos nos relatos do Matcher

Solução sugerida:

- o Matcher deve atrasar relatos de notas não consecutivas;
- se a nota esperada aparecer (dentro de um tempo máximo permitido de atraso), o relato anterior é cancelado;
- caso contrário, ele é enviado para o Accompanist.

5. Atrasos nos relatos do Matcher

Compensação no tempo virtual:



(1) $\Delta real\_time$
(2) $\Delta virtual\_time$
(3) N
(4) $virtual\_time\_delay$

That's all folks!

The End! =)