Introdução a Faust

Gilmar Dias André J. Bianchi

22/11/2012

A linguagem

FAUST - Functional AUdio STream

- É uma linguagem de especificação (de *processadores de sinais*).
- Compilada: FAUST -> C++ -> binários executáveis.
- Opera no nível amostral.
- Semântica simples e bem definida.
- Orientada a diagramas de bloco utilizando programação funcional.
- Pode ser utilizado em diversas “arquiteturas”.
- Descreve um processador de sinais.

IDE Experimental - FaustWorks

Requisitos:

- Faust
- Qt4

Download:

- http://faust.grame.fr/

Instalação:

```
$ qmake-qt4
$ make
```

Estrutura de um programa

Expressões:

- Bloco primitivo.
- Combinação de expressões via operadores de composição.
- Representa um diagrama de blocos.

Definições:

- Identifiação de expressões:

  ```
  nome_do_bloco = expressão ;
  ```

- O nome do bloco deve começar com letra, pode ter símbolos e '_'.

Estrutura de um programa

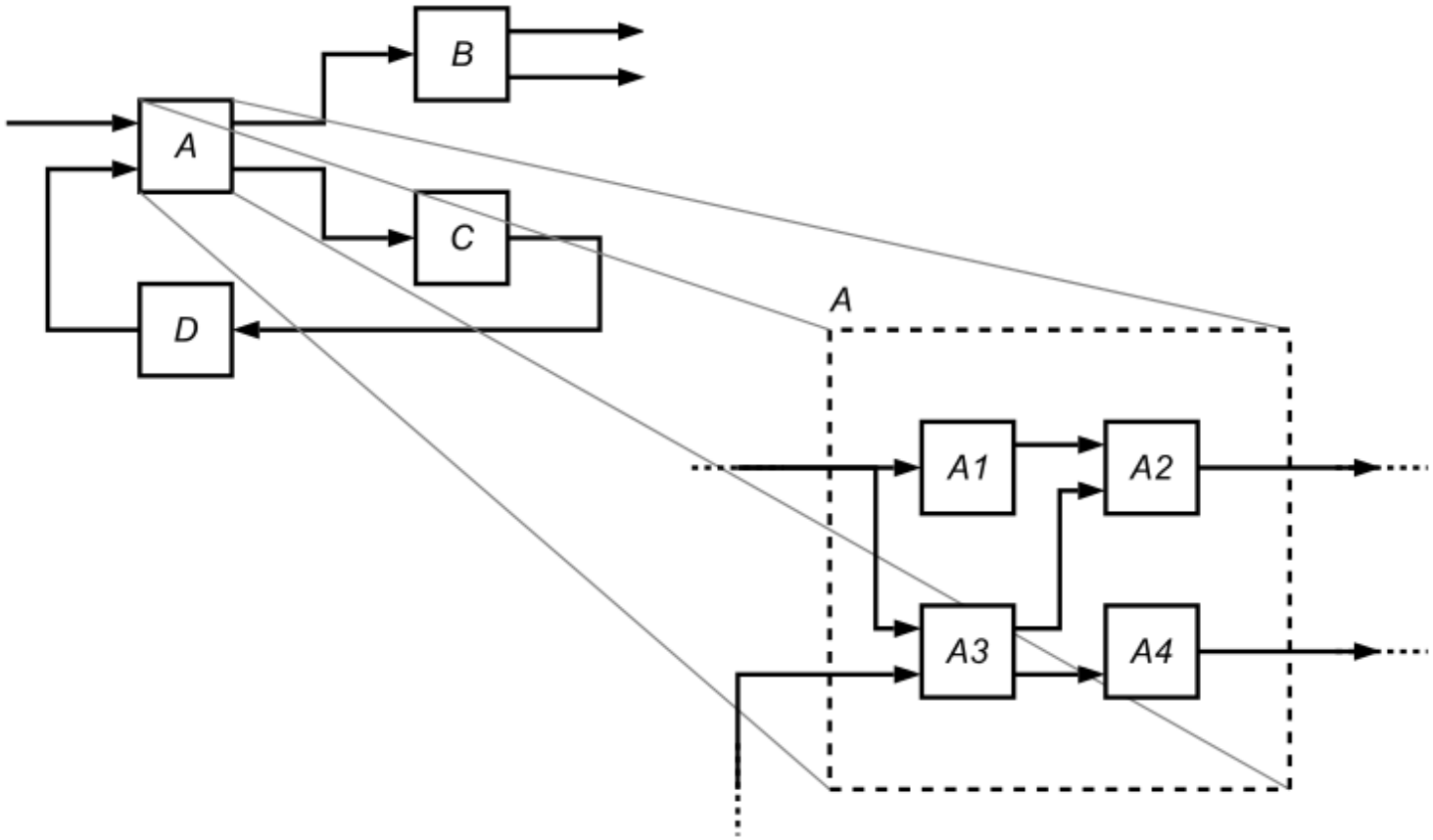


Figure: Definições em um diagrama de blocos

Estrutura de um programa

Programa em Faust: sequência desordenada de definições.

- Apenas uma definição é obrigatória:

  `process`

Comentários:

- Como em C/C++

  `//, /* ... */`

Estrutura de um programa

Exemplo:

```
1  declare name "noise";
2  declare copyright "(C)GRAME 2006";
3
4  import("music.lib");
5
6  // noise level controlled by a slider
7  process = noise * vslider("volume", 0, 0, 1, 0.1);
```

A linguagem - Primitivas

Blocos de conexão (plug boxes):

- Identidade (_).

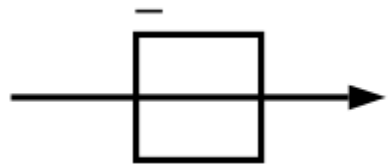


Figure: Bloco identidade

- Corte (!).

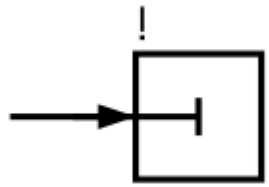


Figure: Bloco corte

A linguagem - Primitivas

Operadores matemáticos:

- Aritméticos (`+`, `-`, `*`, `/` e `%`).

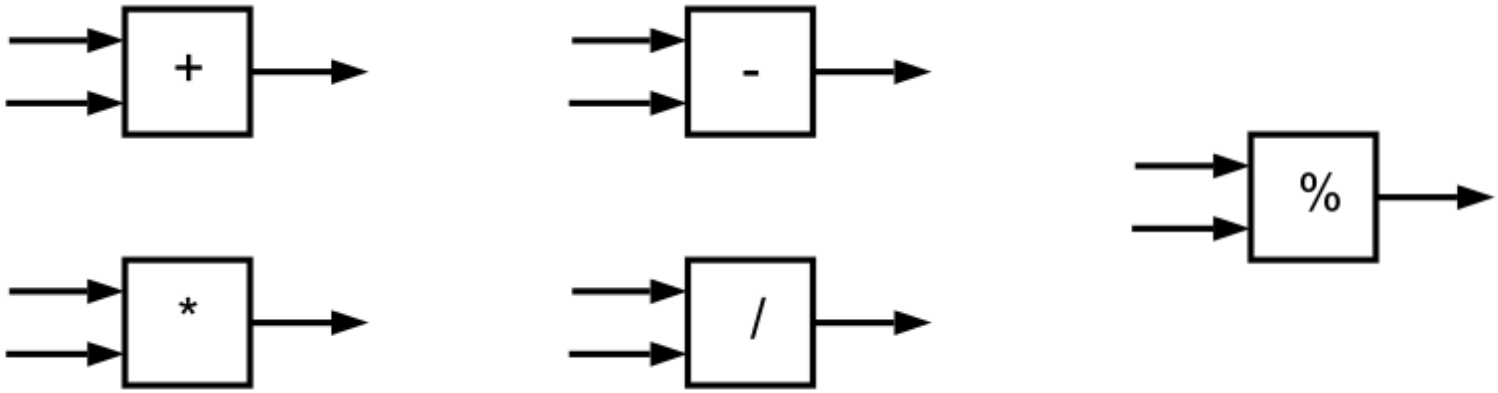
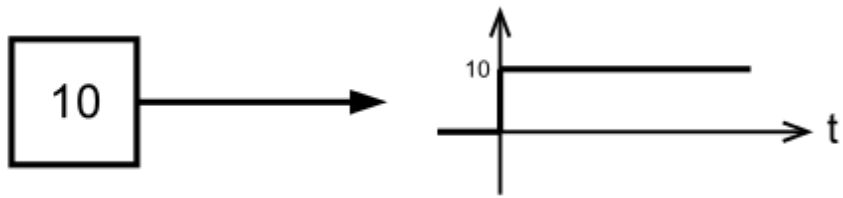


Figure: Blocos de operadores aritiméticos

- Comparação (`<=`, `>=`, `<`, `>`, `==` e `!=`).

A linguagem - Primitivas

- Operadores bit a bit (`<<`, `>>`, `&`, `|` e `^`).
- Constantes:
  - Literais em C++
  - int: '#', float: '#.#'



A linguagem - Primitivas

- Conversões de tipo em Faust (casting):

Figure: Conversões de tipo em Faust

- Funções estrangeiras:

```
ffunction(prototipo, include, biblioteca)
```

Primitivas - Memórias

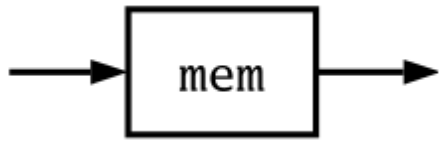
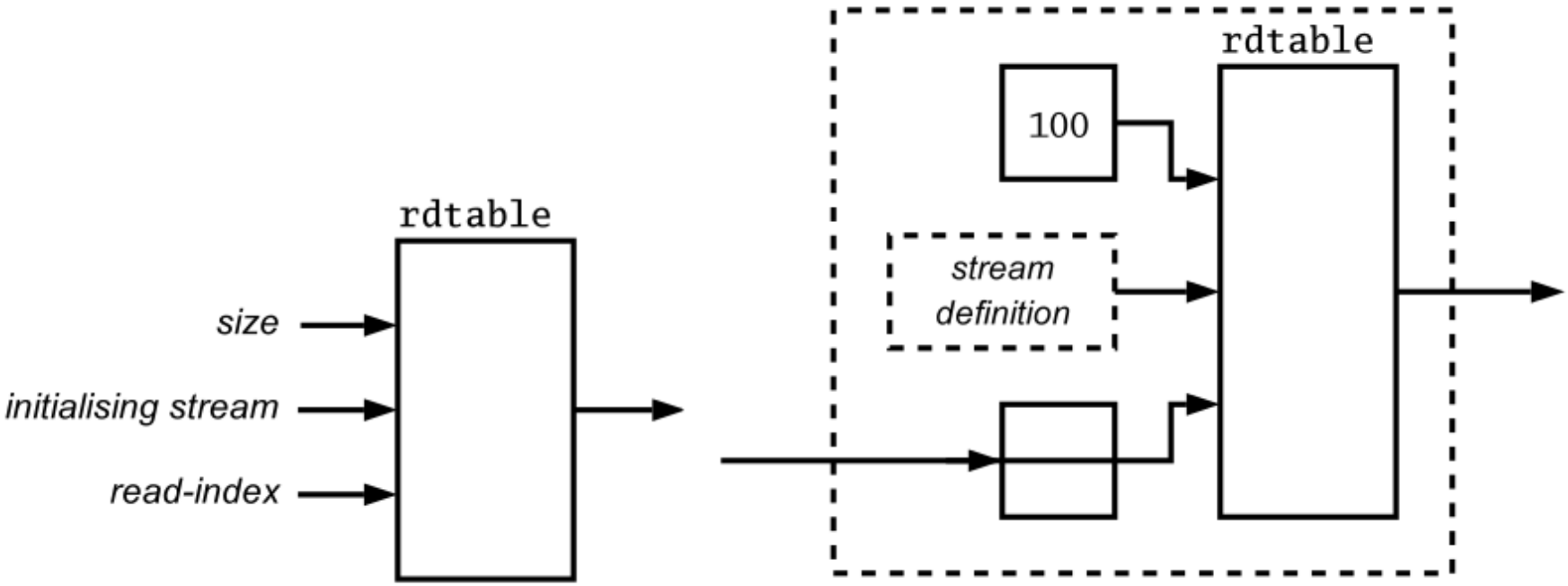


Figure: Atraso simples

Primitivas - Memórias

- Tabela somente de leitura



Primitivas - Memórias

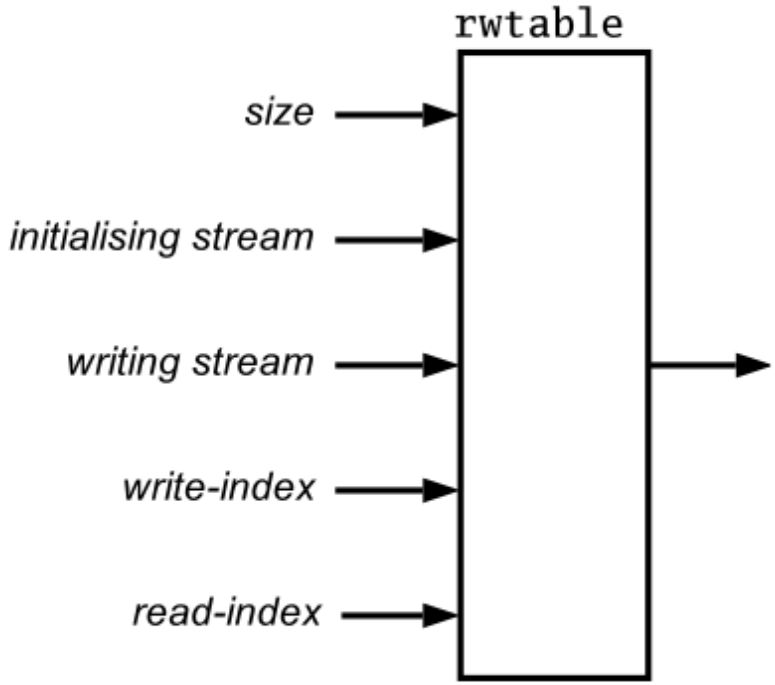


Figure: Tabela de leitura e escrita

A linguagem - Primitivas

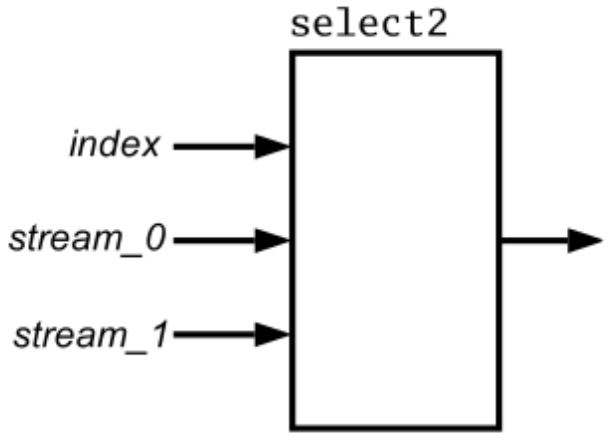


Figure: Chaves de seleção: select2 e select3

Primitivas - Interface gráfica com o usuário

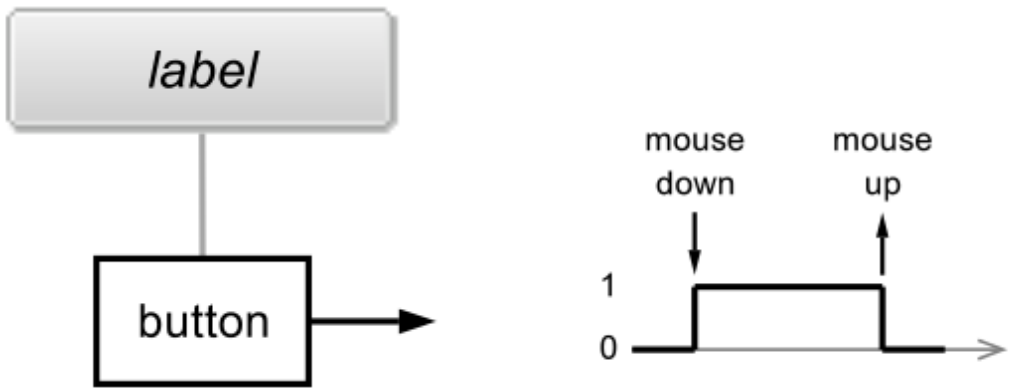


Figure: Botão - `button("label")`

Primitivas - Interface gráfica com o usuário

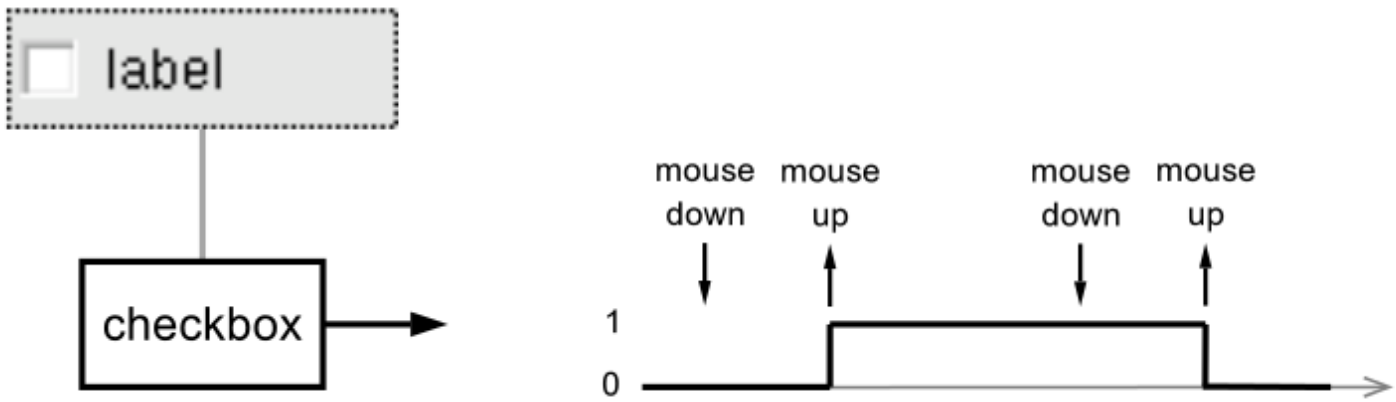


Figure: Checkbox: `checkbox("label")`

Primitivas - Interface gráfica com o usuário

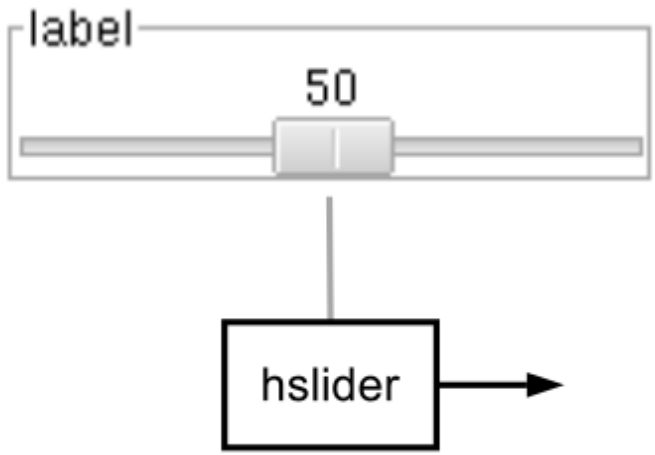


Figure: Sliders: `hslider("label", inicio, fim, max, passo)`

Primitivas - Interface gráfica com o usuário

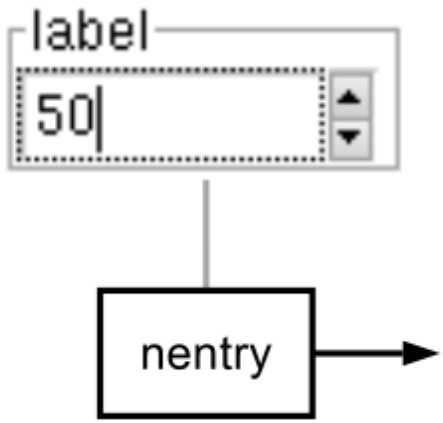


Figure: Entrada numérica: `nentry("label", inicio, fim, max, passo)`

Primitivas - Interface gráfica com o usuário

- Grupos hgroup, vgroup, tgroup

Figure:

```
tgroup("Sliders", (A,B,C))
```

A linguagem - Operadores de composição

- Serial (:)

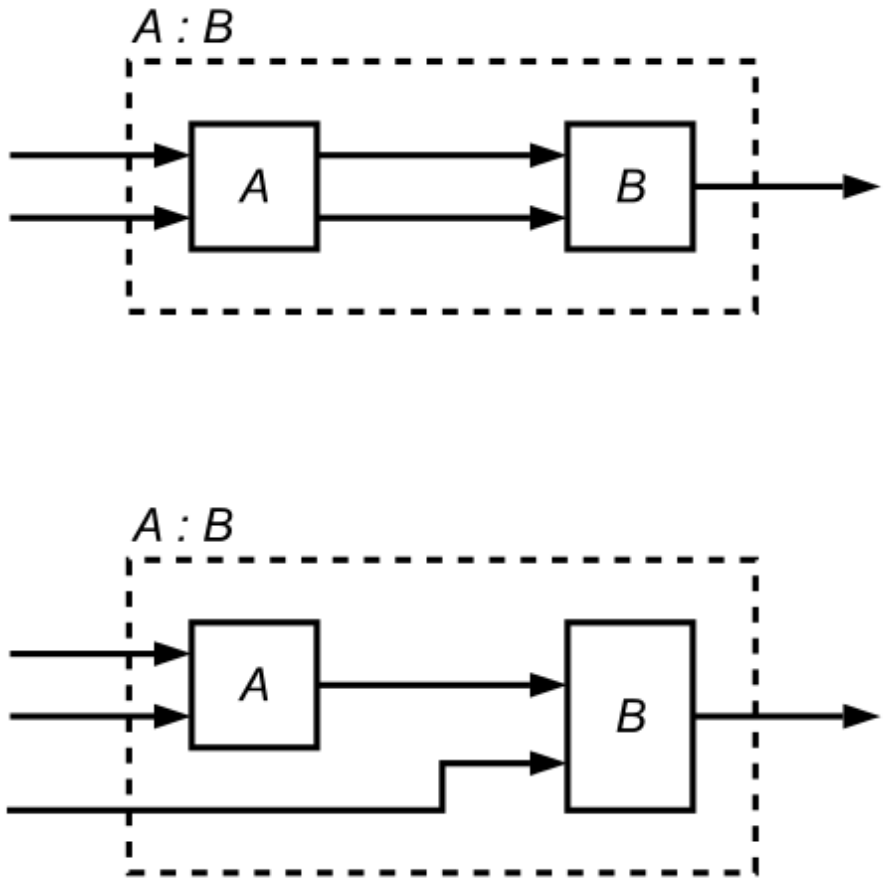


Figure: Operador serial

A linguagem - Operadores de composição

- Serial (:)

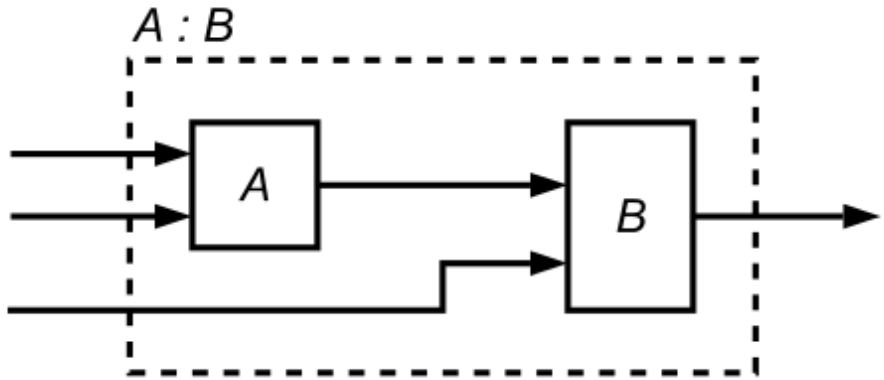


Figure: Operador serial

A linguagem - Operadores de composição

- Serial (:)

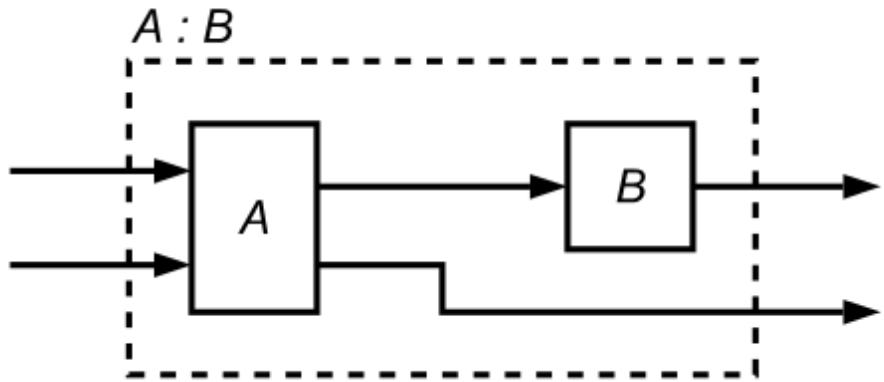


Figure: Operador serial

A linguagem - Operadores de composição

- Paralelo (,)

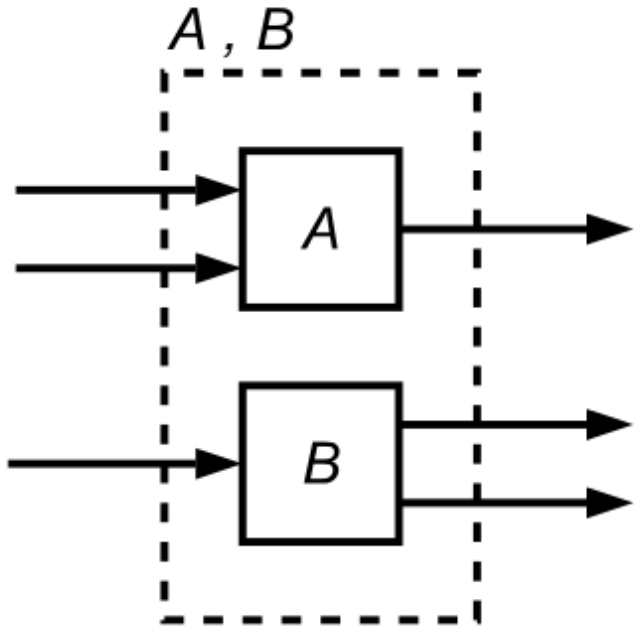


Figure: Operador paralelo

A linguagem - Operadores de composição

- Split ($<:$)

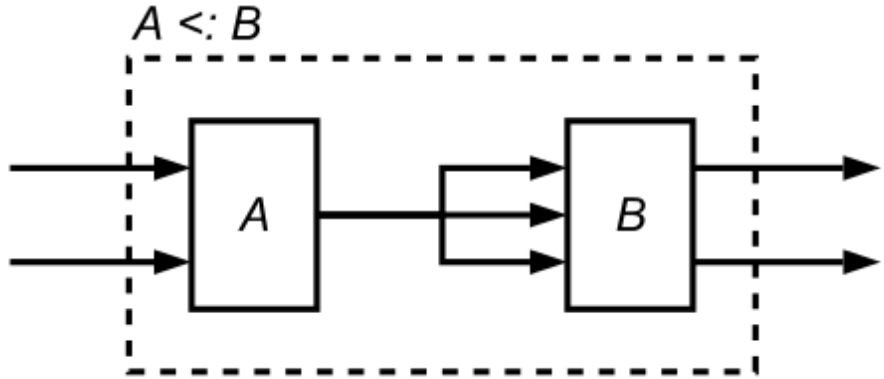


Figure: Operador split

A linguagem - Operadores de composição

- Split ($<:$)

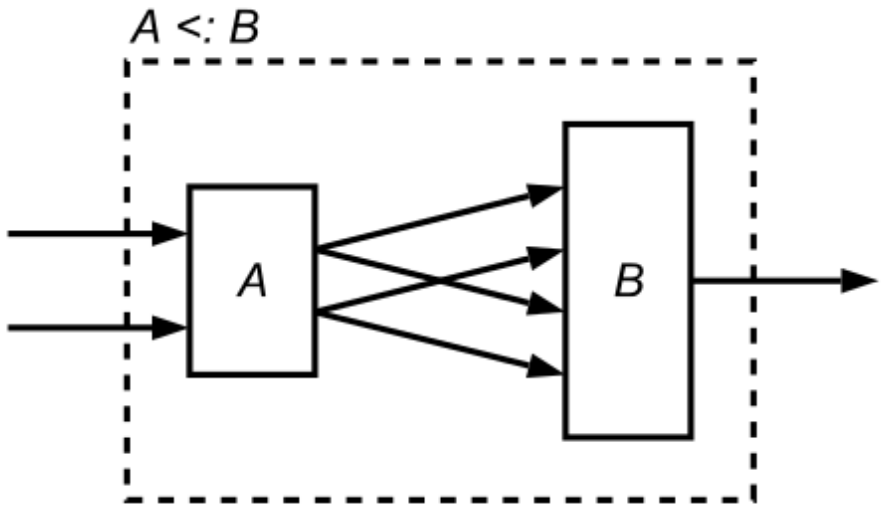


Figure: Operador split

A linguagem - Operadores de composição

- Split ($<:$)

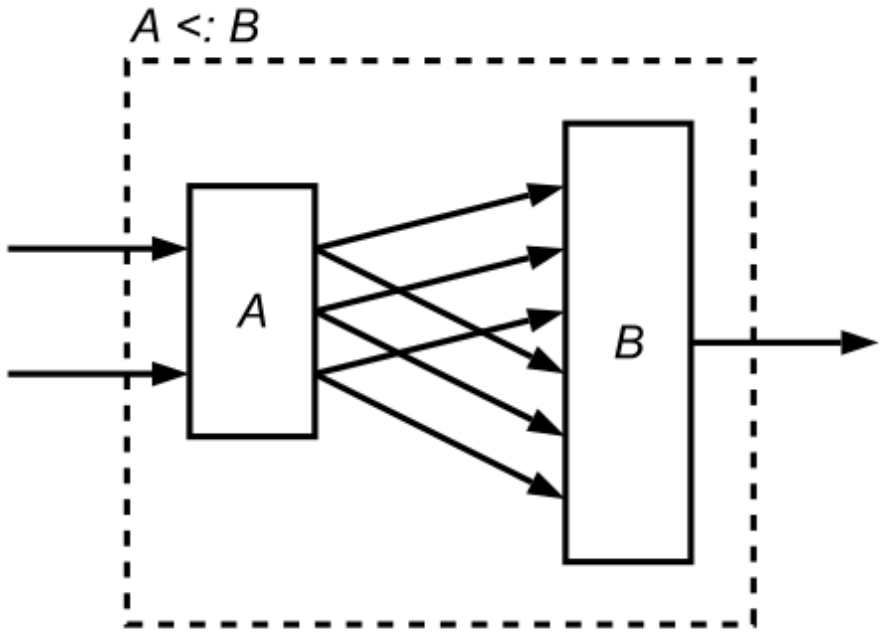


Figure: Operador split

A linguagem - Operadores de composição

- Merge ($:>$)

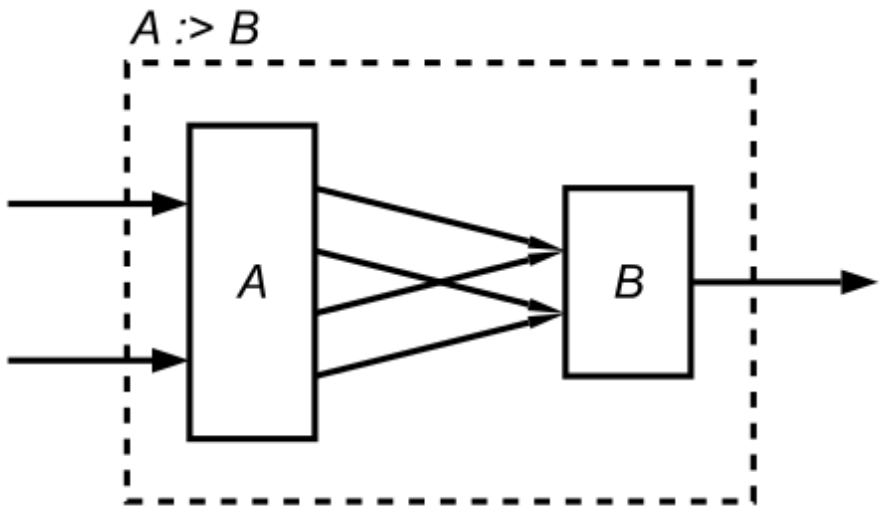


Figure: Operador merge

A linguagem - Operadores de composição

- Recursivo (~)

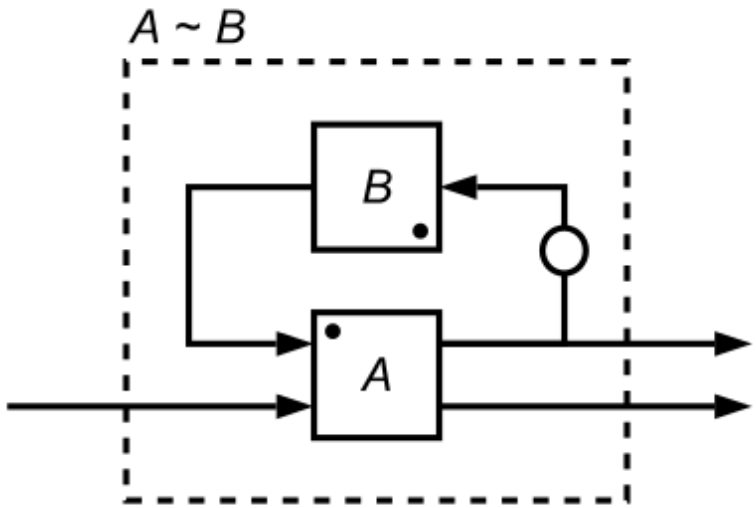


Figure: Operador recursivo

Operadores de composição

- Recursivo (~)

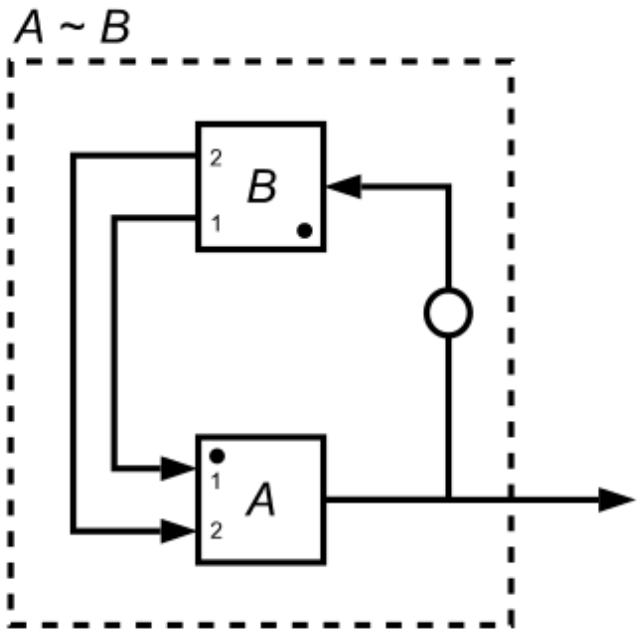


Figure: Operador recursivo

Precedência de operadores

Priority	Symbol	Name
3	~	recursive
2	,	parallel
1	:, <:, :>	serial, split, merge
0	+, −, *, /, ...	arithmetic operators

Figure: Precedência

Precedência de operadores

Symbol	Name	Associativity
~	recursive	Left
,	parallel	Right
:, <:, :>	serial, split, merge	Right

Figure: Precedência

A linguagem - Abstrações

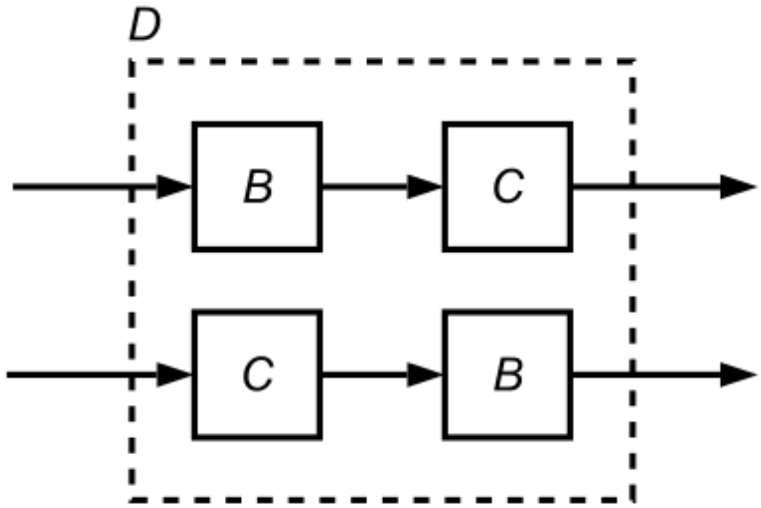


Figure: Abstrações

```
A(x,y) = (x,y):(y,x)
B = ...
C = ...
D = A(B,C)
```

Açúcar sintático

Operadores matemáticos podem ser usados de forma mais simples:

```
A/B  = (A,B) : /
/(A) = (_,A) : /
```

Exemplos

Exemplo 1

Onda quadrada

Onda quadrada: blocos básicos

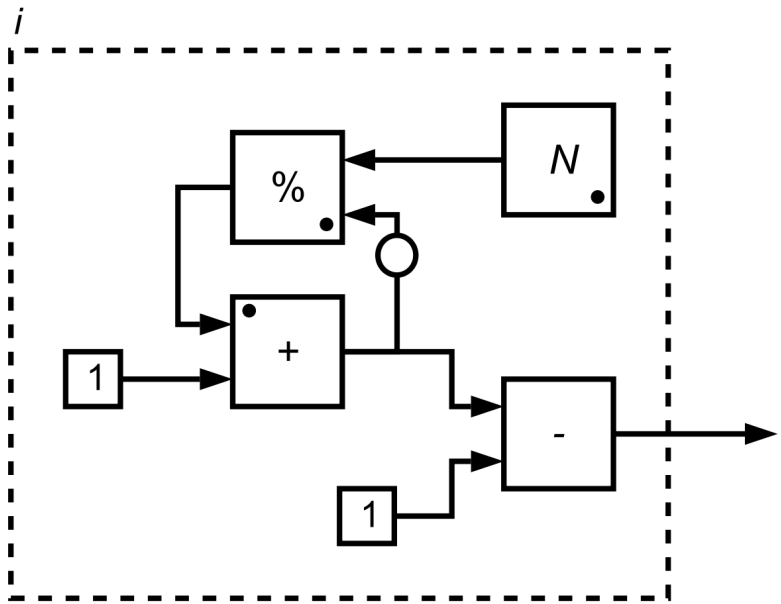


Figure: Um contador de 0 até N-1

Onda quadrada: blocos básicos

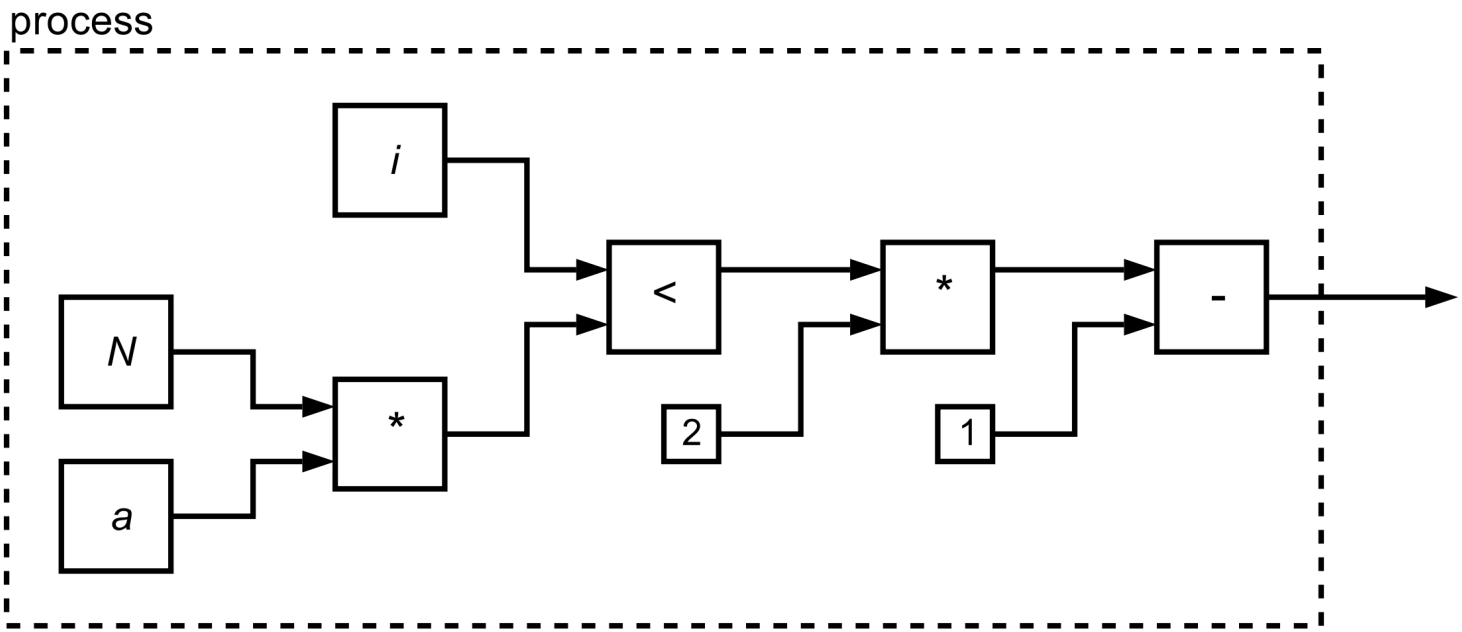


Figure: Gerador de onda quadrada.

Onda quadrada: sliders

Elementos de interface gráfica:

```
vslider(str, cur, min, max, step)
hslider(str, cur, min, max, step)
```

Onda quadrada: código

```
1  //      A square wave oscillator
2  //-----------------------------------
3  T = hslider("Period",1,0.1,100.,0.1); // Period (ms)
4  N = 44100./1000.*T:int; // The period in samples
5  a = hslider("Cyclic ratio",0.5,0,1,0.1); // Cyclic ratio
6  i = +(1)~%(N):-(1); // 0,1,2...,n
7  process = i,N*a : < : *(2) : -(1) ;
```

Onda quadrada: demo

Demonstração:

```
./square -cyclicratio 0.5 -n 44100 -period 100
```

Exemplo 2: Onda senoidal

Onda senoidal: blocos básicos

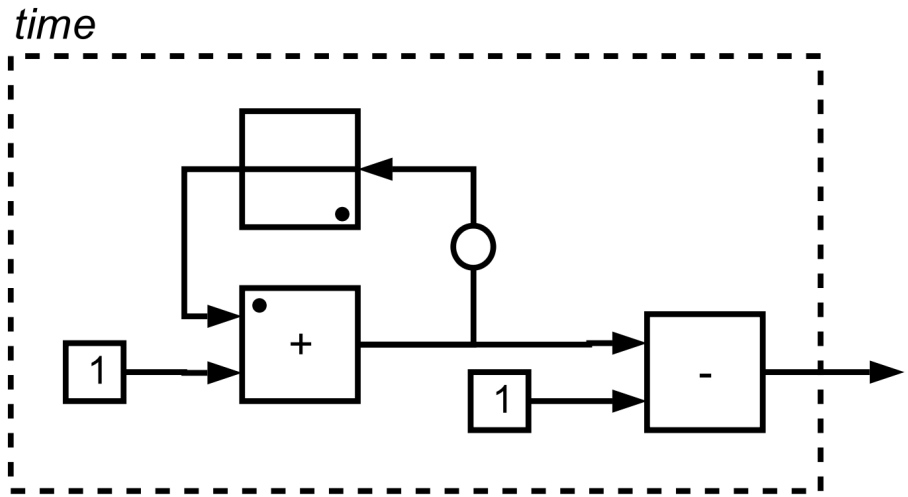


Figure: Tempo discreto: 0, 1, 2, 3, ...

Onda senoidal: blocos básicos

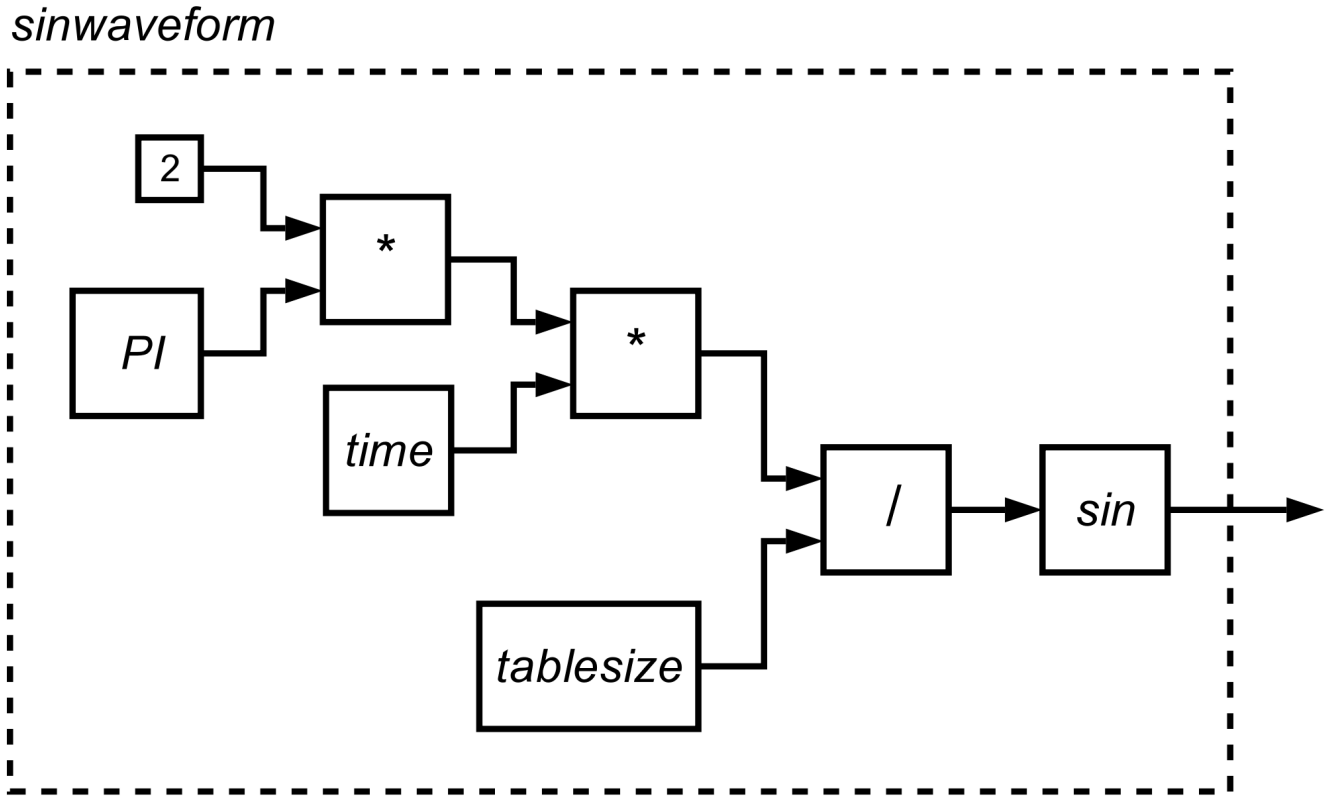


Figure: Forma de onda senoidal

Onda senoidal: blocos básicos

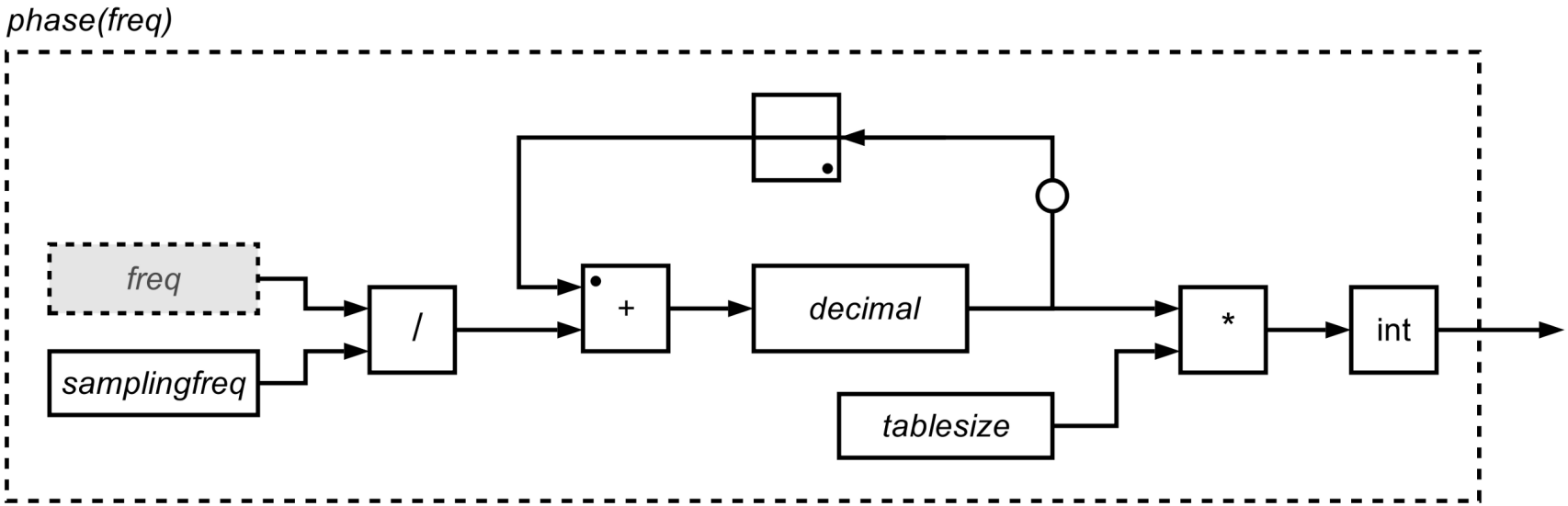


Figure: Cálculo da fase

Onda senoidal: leitura de tabela

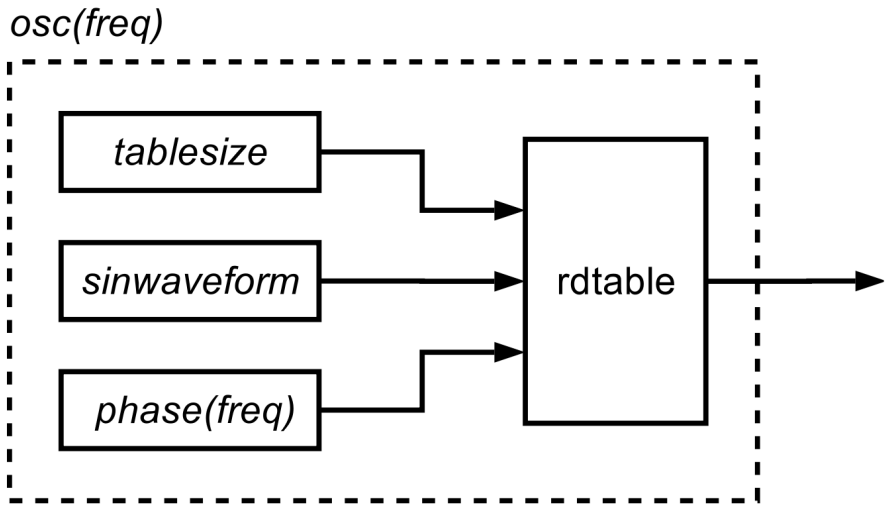


Figure: Forma de onda senoidal

Onda senoidal: código

```
1  //-----------------------------------------------
2  //       Sinusoidal Oscillator
3  //-----------------------------------------------
4  // Mathematical functions & constants
5  //-------------------------------------
6  sin   = ffunction(float sin (float), <math.h>, "");
7  floor = ffunction(float floor (float), <math.h>, "");
8  PI    = 3.14159265358979323385;
```

Onda senoidal: código

```
1  // Oscillator definition
2  //-------------------------
3  tablesize    = 40000 ;
4  samplingfreq = 44100. ;
5  time         = (+(1)~_) - 1; // 0,1,2,3,...
6  sinwaveform = time*(2*PI)/tablesize : sin;
7  decimal      =  _ <: -(floor);
8  phase(freq) = freq/samplingfreq :
9                    (+ : decimal) ~ _ : *(tablesize) : int ;
10 osc(freq)   = phase(freq) : rdtable(tablesize,sinwaveform);
```

Onda senoidal: código

```
1  // User interface
2  //-----------------
3  vol = hslider("volume", 0, 0, 1, 0.001);
4  freq = hslider("freq", 400, 0, 15000, 0.1);
5  //-----------------
6  process = osc(freq) * vol;
```

Onda senoidal: demo

Demonstração:

```
./sineosc -freq 440 -n 44100 -volume 1
```

Exemplo 3: Gerador de ruído aleatório

Gerador de ruído aleatório: código

```
1  //-----------------------------------------------
2  //           Two noises compared
3  //-----------------------------------------------
4
5  RANDMAX = 2147483647;
6
7  random1 = ffunction(int random (), <stdlib.h>, "");
8  noise1 = (random1 << 1) * (1.0/RANDMAX);
9
10 random2 = (*(1103515245)+12345) ~ _ ;
11 noise2 = random2 * (1.0/RANDMAX);
12
13 compare(a,b) = (a*(1-button("Switch")) +
14                                   b*button("Switch"));
15
16 process = compare(noise1, noise2) *
17           hslider("volume", 0, 0, 1, 0.01) <: _,_  ;
```

Gerador de ruído aleatório: demo

Demonstração: comparação de gráficos e som.

Exemplo 4: Karplus Strong

Karplus Strong: demo

Demonstração: tosco e bem feito.

Exemplo 5: Freeverb

Freeverb: código

```
1  monoReverb(fb1, fb2, damp, spread)
2      = _ <:  comb(combtuningL1+spread, fb1, damp),
3              comb(combtuningL2+spread, fb1, damp),
4              comb(combtuningL3+spread, fb1, damp),
5              comb(combtuningL4+spread, fb1, damp),
6              comb(combtuningL5+spread, fb1, damp),
7              comb(combtuningL6+spread, fb1, damp),
8              comb(combtuningL7+spread, fb1, damp),
9              comb(combtuningL8+spread, fb1, damp)
10         +>
11             allpass (allpasstuningL1+spread, fb2)
12         :   allpass (allpasstuningL2+spread, fb2)
13         :   allpass (allpasstuningL3+spread, fb2)
14         :   allpass (allpasstuningL4+spread, fb2)
15         ;
```

Freeverb: código

```
1  stereoReverb(fb1, fb2, damp, spread)
2      = + <:  monoReverb(fb1, fb2, damp, 0), monoReverb(fb1,
```

Freeverb: demo

Demonstração: FaustWorks e LADSPA.

Conclusão

Outros exemplos

- capture.
- dbmeter.
- echo.
- noise.
- multiband-filter.

Outras arquiteturas

- Alsa + {GTK,QT}
- Csound
- iPhone
- jack {GTK,QT}
- Ladspa
- lv2
- Max MSP
- Octave
- OSS
- Pure Data
- VST
- . . . e muitas outras.

Fim!

Obrigado pela atenção!

- Faust: http://faust.grame.fr/
- Contato: {grodias,ajb}@ime.usp.br
- Grupo de Computação Musical do IME/USP: http://compmus.ime.usp.br/