Linguagens Síncronas

Sistemas Reativos

Noemi Rodriguez

Departamento de Informática
PUC-Rio

2019

Sistemas Reativos

- concorrência
- requisitos de temporização
- determinismo
- confiabilidade

Linguagens Síncronas

- primitivas que facilitem expressar a reação a eventos
- sequência do programa estabelecida por entrada de eventos
- tempo tratado como outros eventos

- Reação de programa a eventos é instantânea.

Linguagens Síncronas

- implementação pode garantir propriedades desejadas
- tipicamente usadas para implementar *parte* de sistema complexo
- idéias de comunicação com ambiente
  - restrições limitam uso para uso computacional geral

exemplos

- Esterel
- Céu

Esterel

- comunicação com ambiente: sinais e sensores
  - broadcast de sinais com ou sem valores
- sinais - entrada e saída
- sensores - entrada
- programa: sequência de ticks lógicos
  - programa reage a número arbitrário de eventos ocorridos

- Frédéric Boussinot and Robert De Simone. 1991. The Esterel language. Proceedings of the IEEE 79, 9, 1293–1304.

Céu

- comunicação com ambiente: eventoa
- concorrência modelada por *trilhas* de programas
- comando `await` para programa
  - avanço no tempo
- programa: sequência discreta de reações a eventos do ambiente

- F. Sant'anna, R. Ierusalimschy, N. Rodriguez, S. Rossetto, and A. Branco. 2017. The Design and Implementation of the Synchronous Language CÉU. ACM Trans on Embedded Computer Systems. 16(4). 2017.
- `http://www.ceu-lang.org`

Céu: exemplo

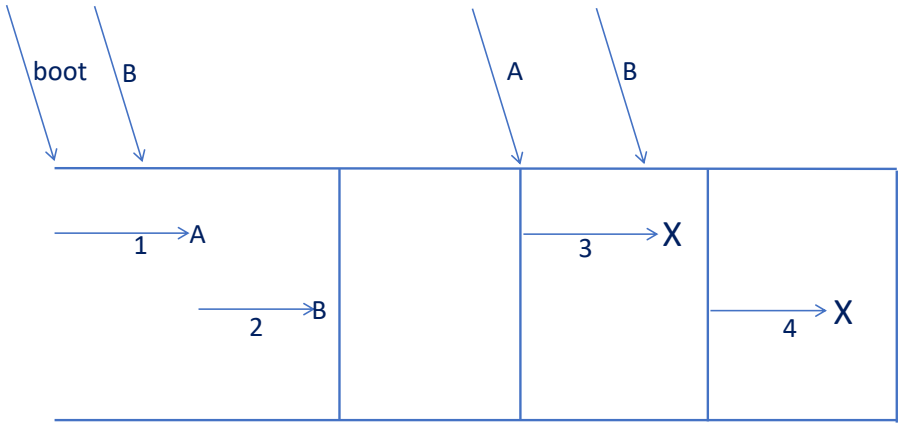
```
par/or do
  loop do
    emit led0(on);
    await 1s;
    emit led0(off);
    await 1s;
  end
with
  await key1pressed;
end
emit led0(on);
```

Céu: execução de programa

1. Programa inicialmente reage ao evento “boot” em uma única trilha.
2. Trilhas ativas executam até terminar ou executarem um `await`.
   - sem paralelismo real
   - tempo limitado
3. Programa vai para estado ocioso.
4. Na ocorrência de algum evento, ambiente de execução acorda todas as trilhas que estejam esperando este evento. Volta ao passo 2.

Céu: execução programa

```
par/and do
   <...> //1
   await A;
   <...> //3
with
   <...> //2
   await B;
   <...> //4
end
```



- eventos só são tratados se programa está a espera deles

Céu: programação reativa estruturada

- estado agora de volta ao ponto do código em que estamos!
- uso de variáveis globais decresce bastante

Céu: *sampling* e *timeouts*

tarefas periódicas:

```
par/and do
  emit tempreq;
  t = await temp;
  ...
with
  await 2s;
end
```

timeout:

```
par/or do
  msg = await receive;
  ...
with
  await 1s;
end
```

Céu: verificações

- loops sempre têm que conter uma chamada a `await`
- trilhas concorrentes não podem ter acessos conflitantes a uma variável
  - trilhas concorrentes: segmentos em paralelo que reagem ao mesmo evento

```
1  input void A, B;
2  var int x = 1;
3  par/and do
4      await A;
5      x = x + 1;
6  with
7      await B;
8      x = x * 2;
9  end
```

[a] Accesses to x are safe

```
1  input void A;
2  var int y = 1;
3  par/and do
4      await A;
5      y = y + 1;
6  with
7      await A;
8      y = y * 2;
9  end
```

[b] Accesses to y are unsafe

Céu: timers

- ambiente mantém "tempo externo" e desconta atrasos em novas esperas

```
1  var int v;
2  await 10ms;
3  v = 1;
4  await 1ms;
5  v = 2;
6
7
8
9  .
```

```
1  par/or do
2      await 10ms;
3      <...>          // any non-awa
4      await  1ms;
5      v = 1;
6  with
7      await 12ms;
8      v = 2;
9  end
```