Escola de Computação'2000 - IME-USP

Sistemas de Tempo Real: Abordagem Síncrona

Jean-Marie Farines
Joni da Silva Fraga
Rômulo Silva de Oliveira

LCMI - Laboratório de Controle e Microinformática
DAS - Departamento de Automação e Sistemas
UFSC - Universidade Federal de Santa Catarina

Introdução

- Os *Sistemas de Tempo Real* são sistemas que reagem,
  gerando respostas,
  à estímulos de entrada vindos de seus ambientes

- Estas características os colocam como *Sistemas Reativos* especiais
  que estão submetidos a restrições temporais em suas reações

- O modelo de programação síncrona
  parte do principio que
  o ambiente não interfere com o sistema (ou o programa)
  durante os processamentos das reações

Introdução

- Na recepção do evento de entrada,
  após um eventual cálculo,
  a resposta é considerada emitida simultaneamente à entrada,
  o que caracteriza uma reação como instantânea
- O modelo é dito *síncrono*
  porque as saídas do sistema podem ser vistas
  como sincronizadas com as suas entradas
- A principal conseqüência desta hipótese

  – a *Hipótese Síncrona* –

  é uma simplificação conceptual que facilita a modelagem
  e a análise formal das propriedades do sistema

Hipótese Síncrona

- Parte do pressuposto que
  existe uma máquina suficientemente rápida
  para executar o processamento correspondente à reação
  em tempos não significativos
- A duração do processamento referente a reação,
  se comparado aos tempos relacionados com o ambiente externo,
  é desprezível
- O ambiente externo não evolui durante esse processamento
- A hipótese síncrona também define que
  os eventos ocorridos no sistema
  sejam percebidos instantaneamente
  em diferentes partes do sistema

Hipótese Síncrona

- O modelo de programação síncrono é
  natural do ponto de vista do programador
- Facilita a construção e a compreensão de programas
- Permite a verificação destes

- Ferramentas automáticas possibilitam
  a verificação da correção lógica
  (aspectos temporais viraram aspectos lógicos)
  desses sistemas são associadas à estas linguagens

Hipótese Síncrona

- Um sistema é visto como determinista se
  a mesma seqüência de entradas
  produz sempre a mesma seqüência de saídas
  e não determinista em caso contrário.
- Na abordagem síncrona o sistema é sempre determinista
- Na abordagem convencional, depende do escalonamento

- A hipótese de sincronismo permite ainda que
  programas escritos numa linguagem síncrona
  sejam compilados em autômatos eficientes
- Depois facilmente implementados
  - em linguagens de programação clássicas
  - em hardware

Tipos de Linguagens Síncronas

- Quando as atividades de manuseio de dados
  são importantes e complexas
  enquanto as de controle são reduzidas
- como em aplicações de processamento de sinal
- Mais apropriado seguir um estilo orientado a fluxo de dados
- Como nas linguagens síncronas declarativas
  Lustre e Signal

- Nestas linguagens,
  a reação gera saídas
  a partir da avaliação de um conjunto de equações
  que as definem em função das entradas atuais
  e das entradas prévias (armazenadas).

Tipos de Linguagens Síncronas

- Quando predominam as atividades de controle
  e o manuseio de dados é simples
- Como em aplicações de controle de processos, supervisão de sistemas, protocolos de comunicação, interfaces homem-máquina, drivers de periféricos
- Mais apropriado um estilo orientado ao fluxo de controle
- Como na linguagem síncrona imperativa Esterel
  ou nos autômatos hierárquicos como Statecharts

- Nestas linguagens,
  cada reação corresponde a passagem de uma situação
  à uma nova situação
  em termos de controle

Linguagem Esterel

- A linguagem Esterel é uma linguagem síncrona
- Desenvolvida a partir de 1982
- Por dois laboratórios franceses do INRIA e da ENSMP
- www.esterel.org
- Os princípios básicos
  - *Reatividade*
  - *Sincronismo*
  - *Difusão instantânea*
  - *Determinismo*

Linguagem Esterel

- O modelo é reativo
  - Sistema entra em ação reagindo à presença de estímulos vindos do ambiente em instantes discretos
- As reações sendo instantâneas, atômicas, entradas e saídas se apresentam sincronizadas
- O modelo síncrono não permite uma nova ativação do sistema enquanto o mesmo estiver reagindo ao estímulo atual
- Não há concorrência entre as reações, eliminando assim uma fonte de não determinismo que corresponderia ao entrelaçamento de execuções concorrentes
- A noção de tempo físico é na verdade substituída pela noção de ordem e de simultaneidade entre eventos

Linguagem Esterel

- Considere a especificação informal:
- "*Emite uma saída O, tão logo que as duas entradas A e B tenham ocorrido. Reinicialize este comportamento a cada vez que ocorrer uma entrada R*".

```
module ABRO:
input A, B, R;
output O;
loop
        [await A || await B];
        emit O
each R
end module
```


Linguagem Esterel

- O aumento do número de entradas
  é também facilmente absorvido
  por este estilo de programação
  e o módulo ABCRO conteria apenas esta modificação:

- **loop**
  [await A || await B || await C];
  emit O
  each R

Comandos

- *O atraso*: a construção temporal "**await**" significa a espera por um evento.
  - Quando iniciada, corresponde a uma pausa até o evento ocorrer, instante no qual conclui a operação: "**await A**"
- *A emissão de sinal*: A emissão instantânea de sinal é realizada pela construção "**emit ...**"
  - "**emit O**" corresponde a emissão instantânea do sinal **O**, tão logo a última entrada **A** ou **B** seja recebida.
  - Não pode ocorrer mais de um "**emit**" por instante.

Comandos

- *Seqüência*: "**p;q**" transfere imediatamente o controle a **q**, quando **p** termina
- *Concorrência*: O operador de paralelismo "||" define as construções separadas pelo operador em paralelismo síncrono.
  - A menos da intervenção de algum mecanismo de preempção ou de exceção, a construção termina quando todos seus ramos terminaram.
  - Neste exemplo, "**await A || await B**" termina instantaneamente desde que as duas componentes concluam com as duas entradas **A** e **B** sendo recebidas
- *Aborto ou preempção*: Na construção " **loop p each R**", o corpo **p** é imediatamente inicializado e executa repetidamente até o instante de ocorrência de **R** no qual **p** é abortado e imediatamente reinicializado
  - Esta construção é dita de *aborto ou preempção forte* pois o evento **R** é prioritário sobre o corpo em execução.
  - Se **A**, **B** e **R** ocorrem simultaneamente, **O** não será emitido

Exemplo

- Seja a especificação de um medidor de velocidade descrito informalmente por
  "*Contar o número de centímetros por segundo e difundi-lo a cada segundo como sendo o valor de um sinal* ***Velocidade***".
- Os sinais de entrada do módulo medidor de velocidade
  são gerados a cada centímetro e a cada segundo
- São representados por **Centímetro** e **Segundo**
- Definem cada um, uma unidade de tempo independente
- Caracterizando desta forma que
  o tempo é multiforme no modelo síncrono

Exemplo

- Para simplificar o exemplo
  supõe-se que esses dois sinais não podem ser simultâneos
  - hipótese plausível devido ao ambiente de execução
- A relação de exclusividade de um sinal se representa por **#** numa declaração **relation**

- A difusão do valor da velocidade será feita por um sinal com valor,
  Velocidade,
  que a cada instante além do seu estado
  contém um valor com tipo **integer**

Exemplo

```
module Medidor-Velocidade:
input Centímetro, Segundo;
relation Centímetro # Segundo;
output Velocidade : integer;
loop
    var Distancia := 0 : integer in
        abort
            every Centímetro do
                Distancia := Distancia + 1
            end every
        when Segundo do
                emit Velocidade (Distancia)
        end abort
    end var
end loop
end module
```


Conceito de Tempo

- A noção de tempo físico não existe
- O tempo físico é visto como um sinal entre outros
- Qualquer sinal pode ser considerado para definir uma unidade de tempo independente
- O tempo é dito multiforme,
  qualquer sinal repetido pode ser considerado como definindo sua própria medida de tempo.
- Para representar esta noção, poderia se imaginar uma representação gráfica na forma de vários eixos de tempo com unidades diferentes correspondentes aos diversos sinais repetidos
- No módulo Medidor-Velocidade, as unidades de tempo são **Centímetro** e **Segundo** com eixos de tempo próprios
- A relação entre os mesmos permite que se calcule a velocidade

Outro Exemplo

- Exemplo que descreve o treinamento de um corredor:
- "*Cada manhã, o corredor faz um número fixo de voltas num estádio. A cada volta, ele corre devagar durante 100 metros, depois ele pula a cada passo durante 15 segundos e termina a volta correndo rápido*".
- Determina-se os sinais que corresponderão as unidades de tempo
- Os sinais de entrada são **Manha**, **Volta**, **Metro**, **Passo** e **Segundo**
- Os sinais de saída são **Correr-Devagar**, **Pular** e **Correr-Rápido**
- Relações entre sinais são estabelecidas:
  - **Manha e Segundo** são sincronizados
  - **Volta** e **Metro** são sincronizados
- Os sinais **Correr-Devagar** e **Correr-Rápido**
  serão emitidos de forma contínua
- O sinal de saída **Pular**
  será emitido em reação ao sinal de entrada **Passo**

Outro Exemplo

```
module Corredor
constant Número-Voltas: integer;
input Manha, Volta, Metro, Passo, Segundo;
relation    Manha => Segundo,
  Volta => Metro;
output Correr-Devagar, Pular, Correr-Rápido;
  . . .
```

Outro Exemplo

```
. . .
every Manha do
       abort
              abort
                     abort
                            sustain Correr-Devagar
                     when 100 Metro;
                     abort
                            every Passo do
                                   emit Pular
                            end every
                     when 15 Segundo;
                     sustain Correr-Rápido
              when Volta
       when Número-Voltas Volta
   end every
end module
```


Exemplo do Cinto de Segurança

- Se o motorista liga a chave e não coloca o cinto dentro de 5 segundos
  - Então um alarme bipa por 10 segundos
  - Ou até que a chave seja desligada ou o cinto colocado

```
input key_on, end_timer, key_off, belt_on, reset;
output start_timer, beep;
loop
   abort
      present key_on then
         emit start_timer(5); await end_timer;
         emit start_timer(10);
         abort sustain beep when end_timer;
      end
   when key_off or belt_on
each reset
```

Execução de Tarefas Externas

- Procedimentos externos chamados pela construção "**call**"
  são considerados como instantâneos
- É possível controlar ainda a execução
  de tarefas externas que levam tempo
  usando o mecanismo "**exec**"
- Essas tarefas se comportam como procedimentos a serem executados de forma assíncrona com o programa Esterel
- A forma de assincronismo assim introduzida é restrita para permitir a sincronização com a tarefa apenas quando do término da mesma
- Estas tarefas podem ser também atividades de um objeto real
- "**exec Task (<parâmetros-referência>) (<parâmetros-valores>) return R**"
  - permite a execução de uma tarefa externa
  - **R** é um sinal de retorno que se restringe a um único "**exec**"

Ferramentas

- Linguagem sendo desenvolvida desde 1982 (França)
  por equipes do INRIA e da Ecole des Mines de Sophia-Antipolis
- Atualmente na sua versão 5
  - Compilador (atualmente V5.21) esta sendo disponibilizado gratuitamente no site http://www-sop.inria.fr/meije/esterel/
    para arquiteturas Solaris, Linux, AIX, OSF1 e Windows NT
- Esta versão do compilador permite gerar implementações
  - em software na forma de uma máquina de estados finita
  - em hardware na forma de circuitos
- O código gerado (por exemplo em C) pode ser embutido
  como um núcleo reativo num programa maior
- Da mesma forma, nas implementações de hardware,
  a lista de "*gates*" gerada pode ser embutida em circuitos maiores
- Além do compilador, existe um conjunto de ferramentas disponíveis para desenvolver e para verificar programas em Esterel

Aplicações

- Aviões, automóveis, manufatura
  - Dassault: Controle de aterrizagem
  - Renault
- Protocolos de comunicação
  - AT&T: software em switches
  - Motorola: Circuitos para interface de barramentos
  - Daimler-Benz: Protocolo de comunicação dentro de veículos
- Interface homem-máquina
- Codesign hardware/software

Conclusão

- A abordagem síncrona assume reações instantâneas a eventos extemos
- Esta hipótese de reações instantâneas pode ser entendida como
  - qualquer tempo de resposta é menor que
    o intervalo de tempo mínimo observável no ambiente
- Algumas aplicações de tempo real sustentam esta hipótese
- A utilização de uma linguagem síncrona (Esterel)
  leva à implementações eficientes
  baseadas em autômatos ou em circuitos booleanos
- Aplicações:
  - Sistemas embutidos, protocolos de comunicação
  - Drivers para periféricos, sistemas de supervisão e controle
  - Interfaces homem-máquina