Sumário

- Introdução
- Estruturas típicas
  - Sequência
  - Concorrência
  - Sincronização/partilha de recursos
- Hierarquia
- Validação / verificação
- Implementação em plataformas genéricas

Algoritmo de interpretação

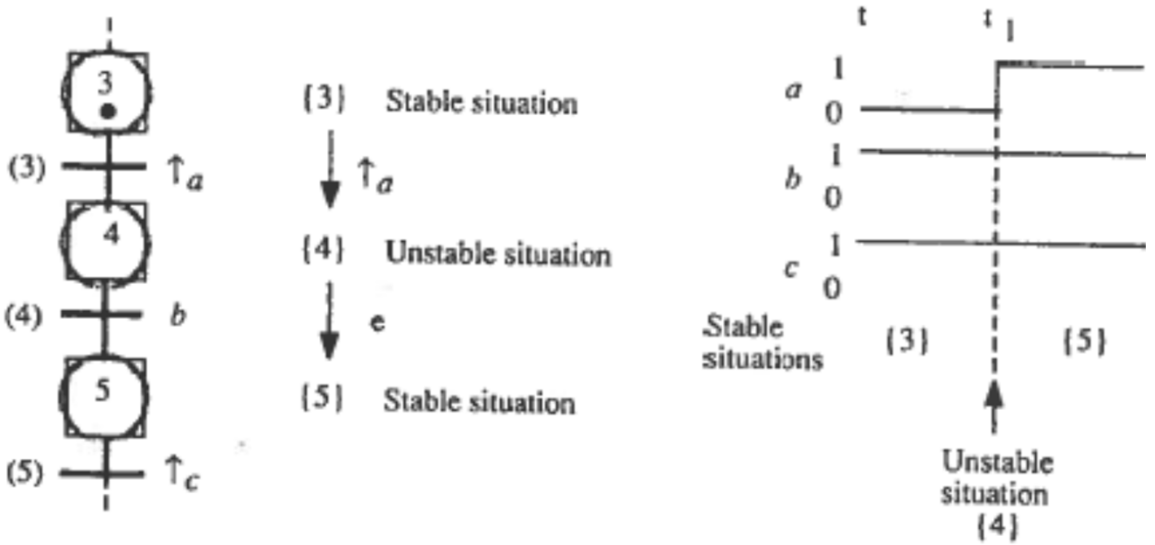
— É necessário ter um algoritmo de interpretação do Grafcet que não suscite ambiguidades de interpretação dos vários intervenientes:

  — Dois utilizadores com a mesma sequência temporal das entradas, devem obter a mesma sequência temporal das saídas

— Partindo de uma situação estável, passando por um conjunto de situações instáveis, onde fica um tempo infinitamente curto (instantâneo), até atingir uma nova situação estável

  — **Situação = conjunto de etapas que estão ativas**

— Exemplo de uma situação instável: **disparo iterado** (sequências de disparos consecutivos)



Algoritmo de interpretação

FEUP

— Considerar que a recetividade (predicado) associada a cada transição **Ri** pode ser escrita como **Ri=Ci.Ei**

  — **Ci** é uma condição boleana e **Ei** é um evento. Exemplo:

    —R1=a*b ⇔ (a*b)***e** , em que **e :** evento (↑) que ocorre sempre (uma abstração)

    —R2=↑a ⇔ 1*↑a

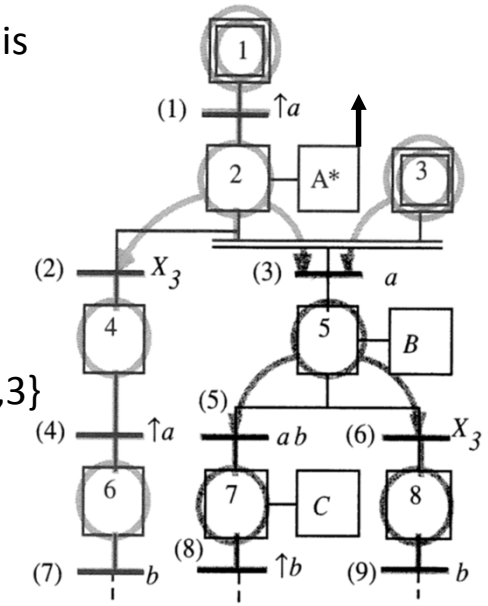
  — Com este conceito, <u>o disparo de uma transição está sempre associado à ocorrência de um evento:</u>

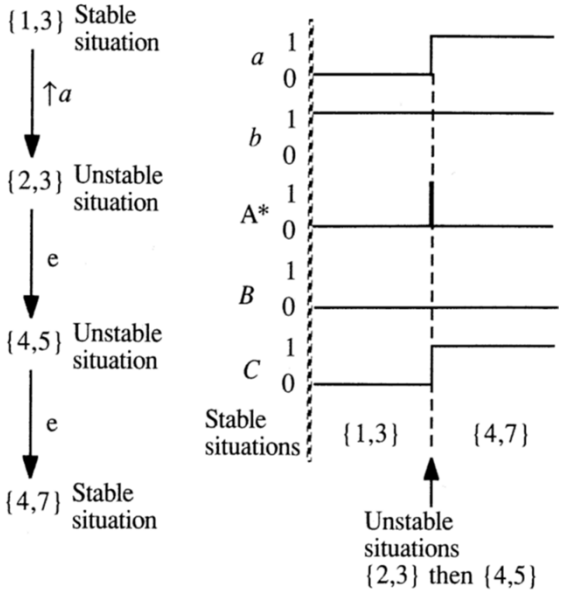
    —<u>Externo OU que ocorre sempre **(e)**</u>

1. Inicialização : ativar as etapas iniciais. Executar as ações impulsionais associadas. Ir para o passo 5.
2. Se ocorrer um evento externo Ei, determinar o conjunto de transições T1 que podem ser disparadas com este evento. Se T1≠Ø ir para o passo 3, caso contrário modificar, se necessário, as ações condicionais associadas com as etapas ativas. Ir para o passo 2
3. Disparar as transições que podem ser disparadas. Se depois do disparo simultâneo destas transições a situação ficar inalterada ir para o passo 6.
4. Executar as ações impulsionais associadas às etapas que foram ativadas no passo 3
5. Determinar o conjunto de transições T2 que podem ser disparadas quando ocorre o evento **e (i.e. transições não associadas e eventos externos).** Se T2≠Ø ir para o passo 3.
6. Após atingir uma situação estável:
   a) Determinar o conjunto de ações contínuas A0 que devem ser desactivadas
   b) Determinar o conjunto de ações contínuas A1 que devem ser activadas
   c) Colocar a 0 todas as ações de A0. Colocar a 1 todas as ações de A1
   d) Ir para o passo 2

Algoritmo de interpretação: exemplo

- P1: situação {1,3}. Sem ações impulsionais
- P5: (1) habilitada, mas não pode ser disparada por **e**. T2=Ø
- P6: situação estável {1,3}, A1=A2=Ø
- P2: quando ↑a ocorre, T1={(1)}
- P3: disparo de (1). Atingida a situação {2,3}
- P4: A ação impulsional A* é executada
- P5: habilitada a transição (2). Atingida a situação {2,3} e a=1, logo T2={(2),(3)}
- P3: disparo simultâneo de (2) e (3). Atingida a situação {4,5}
- P4: Sem ações impulsionais em {4,5}
- P5: habilitadas as transições (4), (5) e (6). A transição (5) é disparável na ocorrência de **e** dado que a*b=1. A transição (4) não é disparável na ocorrência de **e**, mas sim em a↑. A condição associada a (6) não permite o disparo. Logo T2={(5)}





- P3: disparo de (5). Atingida a situação {4,7}
- P4: sem ações impulsionais
- P5: T2=Ø
- P6: Situação estável {4,7}. A0=0, A1={C}. Logo C=1
- P2: à espera de um evento.
- …

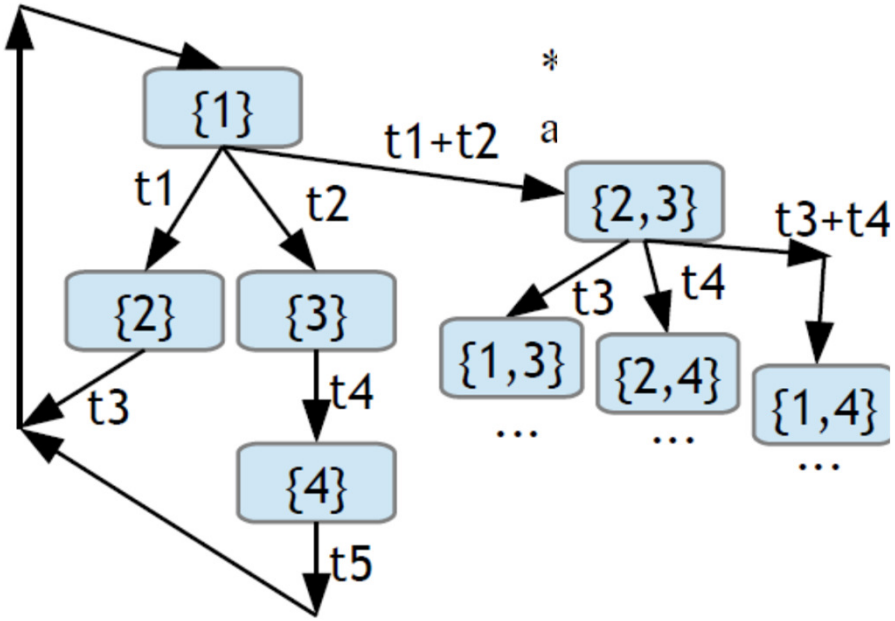
Grafo das situações acessíveis : introdução

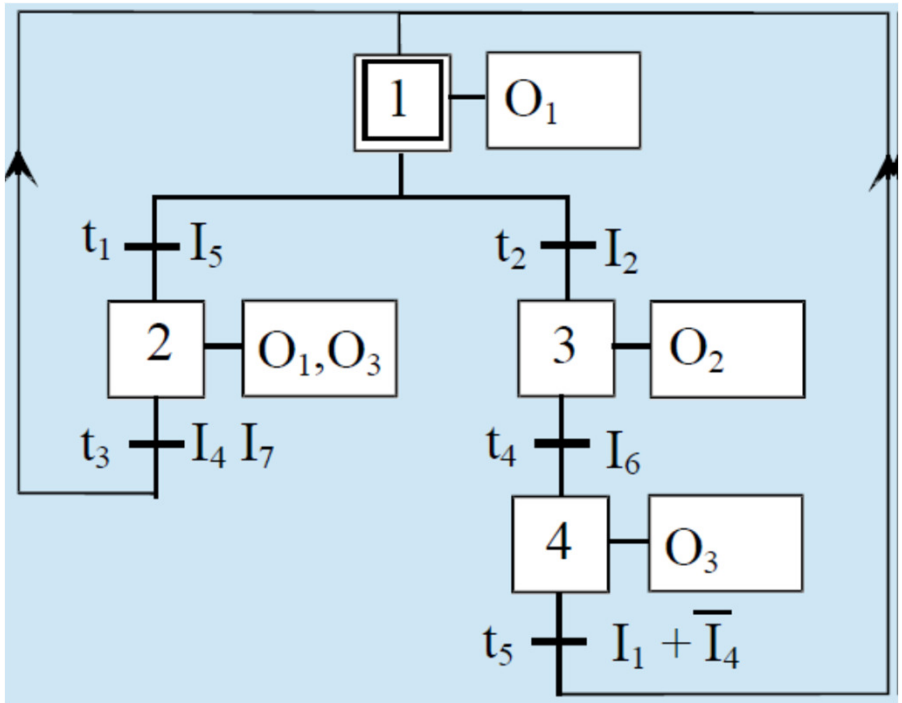
— Depois de construído o Grafcet é necessário verificar se a solução proposta evolui de acordo com o que foi especificado.

— A técnica mais simples consiste na determinação do **Grafo das Situações Acessíveis**

  — Partir da situação inicial

  — Para cada situação, analisar todas as combinações de transições que podem disparar e que originam uma nova situação.





Grafo das situações acessíveis : contexto

FEUP

- O problema torna-se complexo de analisar porque o disparo das transições pode depender de variáveis externas (ex. sensores) cujo o comportamento não é conhecido à partida.

- Uma solução é obter um Grafcet **autónomo** em que se ignora o comportamento particular das variáveis externas, mas em que se **considera o comportamento da estrutura do Grafcet e das variáveis internas.**

  - Obtêm-se assim um grafo que contêm todos os comportamentos possíveis.
  - O grafo "real" (ie. que considera as variáveis externas) será um subconjunto do anterior.
  - Vantagens
    - Deteção de situações de **bloqueio** e/ou **conflito**
  - Desvantagens
    - crescente complexidade de análise quando há muitas etapas ativas em simultâneo e/ou muitas combinações de transições que podem disparar no mesmo instante

    ⇒ **EXPLOSÃO DO ESPAÇO DE ESTADOS**

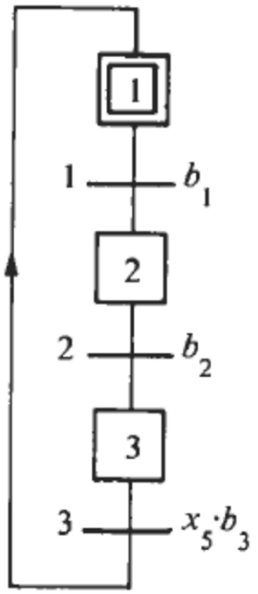
    ⇒ **Necessidade de ferramentas de análise automática : Model Checking !**

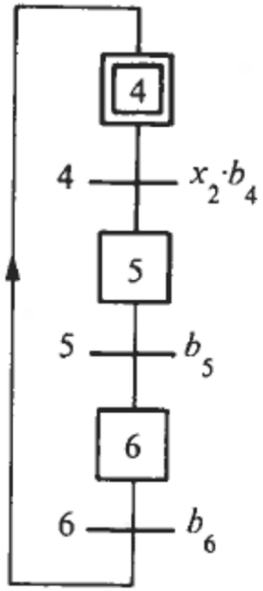
Grafo das situações acessíveis: exemplo

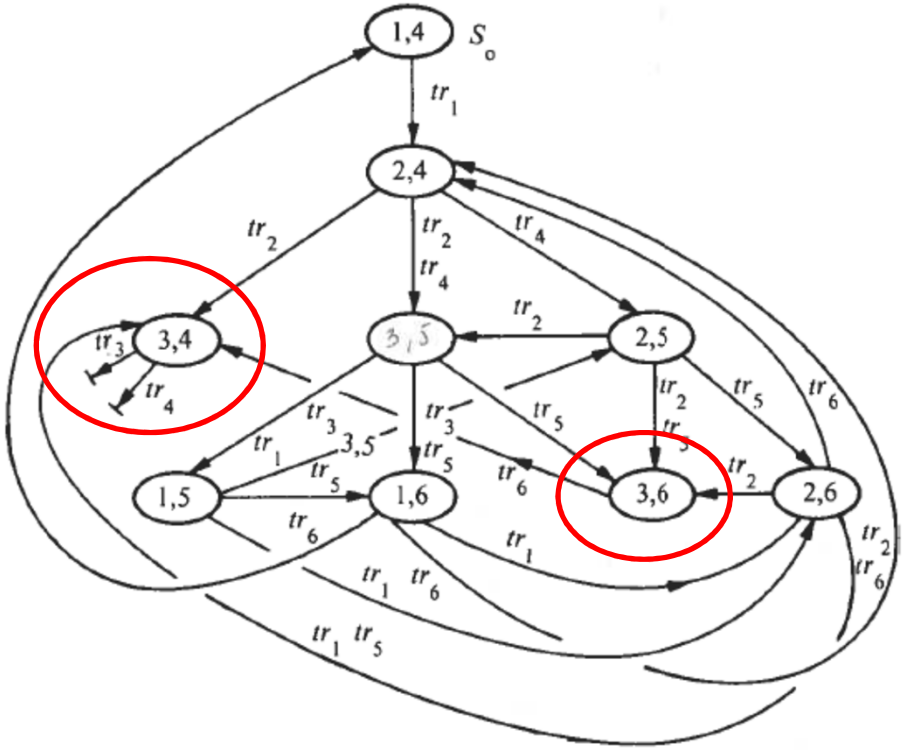
— Há 2 casos de bloqueio (deadlock).

  — Na pratica o bloqueio pode não existir devido ao comportamento das variáveis externas

  — É necessário reformular o modelo para evitar situações deste tipo.







Sumário

- Introdução
- Estruturas típicas
  - Sequência
  - Concorrência
  - Sincronização/partilha de recursos
- Hierarquia
- Validação / verificação
- Implementação em plataformas genéricas

Implementação numa plataforma genérica

FEUP

- É possível implementar o Grafcet em plataformas que não suportem 'de raiz' esta linguagem
- Existem 2 métodos que permitem uma implementação simples:
  - **Método Assíncrono**:
    - Simples de implementar, mas que pode falhar em situações ditas 'instáveis'.
  - **Método Síncrono**:
    - Mais complexo, mas que funciona em todas as situações.
    - Se existirem várias transições ativas no mesmo instante, vão ser disparadas no mesmo instante sem conflito
    - Sequências de disparos consecutivos (que envolvam ativar/desativar etapas) são implementadas corretamente.

Implementação numa plataforma genérica

FEUP

— **Algoritmo do Método Assíncrono**:

1. Verificar se uma transição pode disparar. Se sim, desactivar as etapas anteriores e activar as posteriores
2. Repetir o ponto (1) para todas as transições
3. Executar as acções das etapas que estão activas
4. Voltar a (1)

— **Algoritmo do Método Síncrono**.

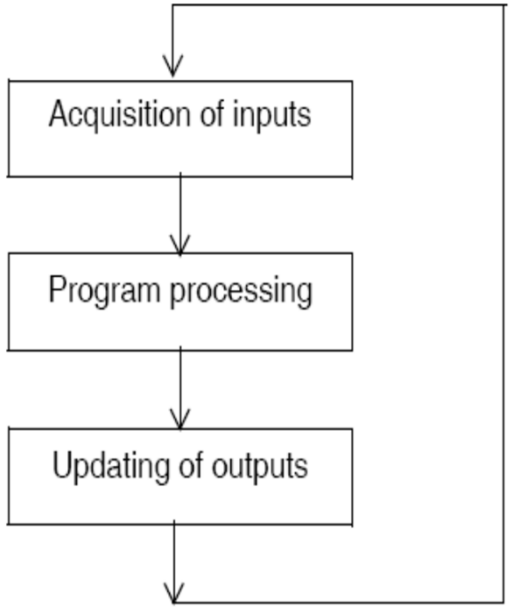
1. Verificar se uma transição pode disparar. Se sim, armazenar esta informação
2. Repetir o ponto (1) para todas as transições
3. Para todas as transições que podem disparar;
   - Desativar todas as etapas anteriores
4. Para todas as transições que podem disparar;
   - Ativar todas as etapas posteriores
5. Executar as ações das etapas que estão ativas.
6. Voltar a (1)

Implementação numa plataforma genérica

— Vamos admitir que temos à disposição uma plataforma com um ambiente de desenvolvimento em C

— E que se pretende um modelo de execução semelhante a um autómato

```
#include "…"

main()
{
        inicialização();

        while(1)
        {
                leitura_das_entradas();
                execução_do_programa();
                actualização_das_saídas();
        }
}
```



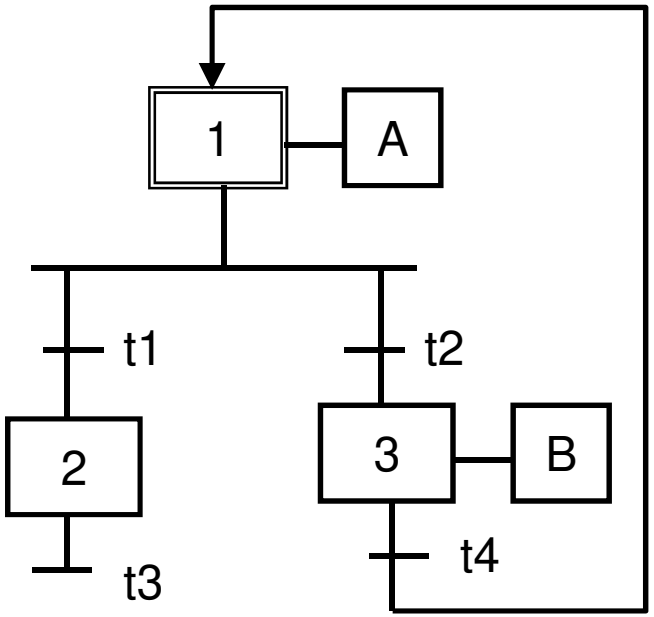
Implementação numa plataforma genérica

FEUP

- Vamos admitir que apenas uma etapa do Grafect está activa de cada vez.
  - Basta uma variável (ex. inteiro) para indicar qual a etapa que está ativa

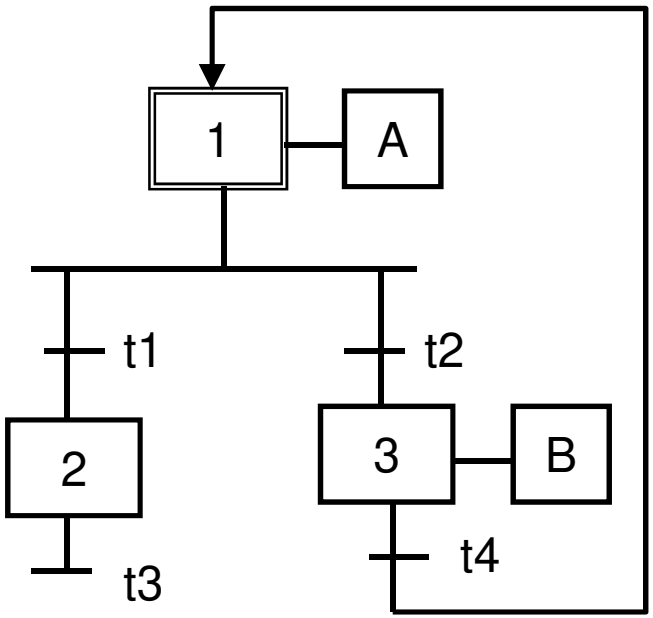
- A implementação do algoritmo assíncrono é simples:
  - Para todas as transições verificar se podem ser 'disparadas' ou não
    - Se SIM, desactivar as etapas imediatamente anteriores e activar as imediatamente seguinte;
    - Se Não, não fazer nada
  - Após disparar todas as transições disparáveis, verificar quais as etapas activas e executar as respectivas acções.

Implementação numa plataforma genérica – exemplo (1)




```
int et; //etapa activa
…
inicializacao()
{
  et = 1;
  …
}
…
}
```

Implementação numa plataforma genérica – exemplo (2)



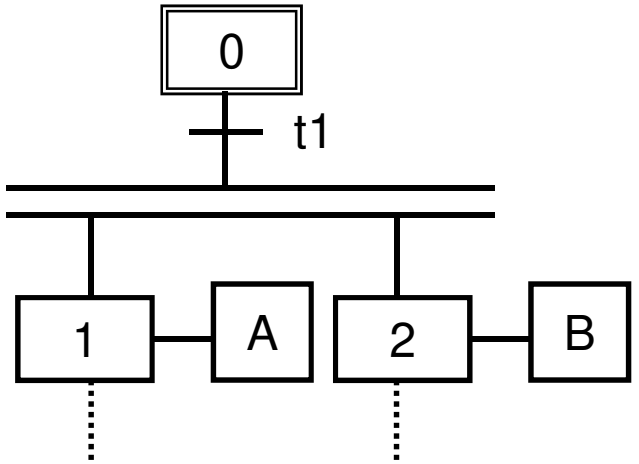

```
execucao_do_programa()
{
   // teste das transições
   if (et==1 && t1==true)
         et = 2;

   if (et==1 && t2==true)
         et = 3;

   if (et==2 && t3==true)
         et = 0; // poço

   if (et==3 && t4==true)
         etapa = 1;

   // execução das acções
   if (et==1) then A=true;
   else A=false;

   if (et==3) then B=true;
   else B=false;
}
```

Implementação numa plataforma genérica – exemplo (3)

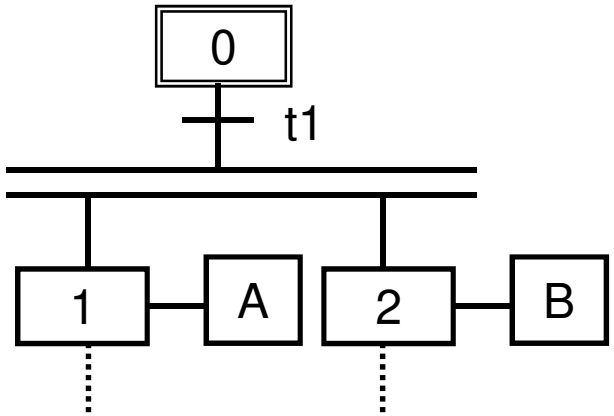
— Se existir mais do que uma etapa ativa em cada instante a solução é do mesmo tipo, basta apenas utilizar uma estrutura de dados que permita armazenar o estado de todas as etapas:

  — Exemplo: utilizar um array de boleanos

```
#define N_ET 10; //numero de etapas
bit etapa[N_ET]; //etapas activas
…
inicializacao()
{
  et[0] = true;
  et[1] = et[2]=…=et[9]=false;
  …
}
```



Implementação numa plataforma genérica – exemplo (4)



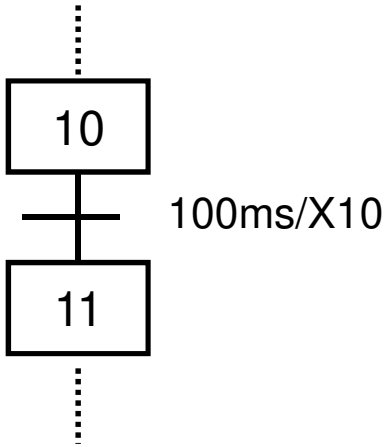

```
…
execucao_do_programa()
{
  // testes das transições
  if (et[0]==true && t1==true)
  {
        et[0] = false;
        et[1] = true;
        et[2] = true;
  }
  …
  // execução das acções
  if (et[1]) then A=true;
  else A=false;

  if (et[3]) then B=true;
  else B=false;
  …
}
```

Implementação numa plataforma genérica – exemplo (5)

— E se existirem acções mais complexas, como por exemplo:

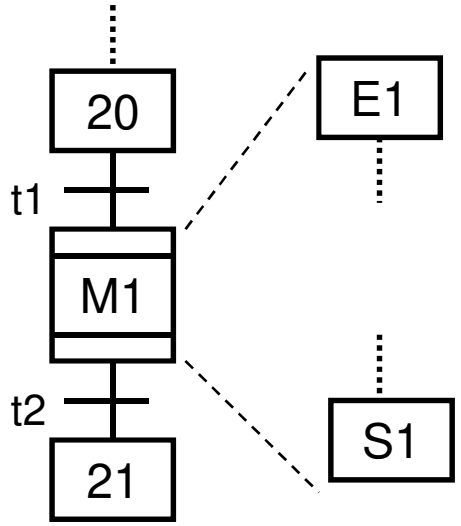
  — Temporizadores, Múltiplos grafcets, Macro-ações e Macro-etapas

— Também é possível implementar todos estes aspectos, basta ter uma abordagem bem estruturada do problema




```
bit timer(st, pv)  //timer, st activo ao flanco
…
execucao_do_programa()
{
 bit t1=false;

 if (et[10] && t1==true)
       et[11]=true;
 …
 if (et[10]) then
       t1=timer(true, 100);    // inicia timer
 else
       t1= timer(false, 100);  // pára o timer
 …
}
```

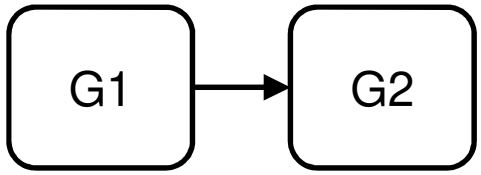
Implementação numa plataforma genérica – exemplo (6)




```
executa_macro_etapa_M1()
{
  …
}
```

```
execucao_do_programa()
{
  // execução de macro-etapas

  bit M1=false; // M1 (macro-etapa)
        …
        if (et[20] && t1==true)
        {
                et[E1]=true;
                M1=true;
        }

        if (M1==true)
        executa_macro_etapa_M1();

        if (et[S1] && t2==true)
        {
                M1=0;
                et[21]=true;
        }
        …
}
```

Implementação numa plataforma genérica – exemplo (7)

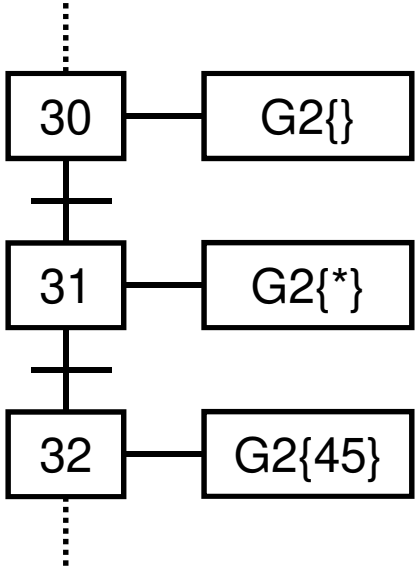



```
execucao_do_programa()
{
  // execução de múltiplos grafcets
  // ordem directa da prioridade

        executa_G1();
        executa_G2();
        …
}
```

Implementação numa plataforma genérica – exemplo (8)

— A implementação de macro-acções pode ser realizada de forma simples

```
execucao_do_programa()
{
  // Reset de um grafcet
  if (G1_et[30])
        G2_et[]=0;
  …
  // Congelar grafcet
  if (G1_et[31])
       forcar_G2 = true;
  …
  // Forçar estado
  if (G1_et[32])
       G2_et[]=0;
       G2_et[45]=true;
  …
}
```




```
executa_G2()
{
  …
  // testes das transições
  if (… && forcar_G2==false)
        …
}
```

Bibliografia

— "The GRAFCET Specification Language", P. Portugal, A. Carvalho, The Industrial Information Technology Handbook, 2004 (resumo alargado do Grafcet)

— "Petri Nets and Grafect", R. David, H. Halla, Prentice Hall, 1992 (na biblioteca)

  — Capítulo : "Grafcet" (com indicação das seções de leitura obrigatória)