

Especificação, Modelação e Projecto de Sistemas Embutidos

StateCharts **HandsOn Session**

Paulo Pedreiras, Luís Almeida
{pbrp,lda}@ua.pt



Departamento de Electrónica, Telecomunicações e Informática
Universidade de Aveiro

Captura de máquinas de estados em linguagens de programação sequenciais

Existem duas possibilidades para efectuar a captura de um modelo baseado em máquinas de estados em linguagens de programação sequenciais:

- Utilização de **ferramentas específicas**
 - ◆ Para além do compilador da linguagem-alvo, usa-se uma ferramenta com suporte à linguagem de máquinas de estados
 - Existem ferramentas para **modo gráfico e textual**
 - Algumas suportam **simulação gráfica** (*user friendly*)
 - Podem **gerar código automaticamente** (C, C++, VHDL, ...)
 - ◆ Problemas
 - Suporte à ferramenta adicional: **custos** com licenças, formação, *upgrades*,
 - Em muitos casos o código gerado (ainda?) não é eficiente

Captura de máquinas de estados em linguagens de programação sequenciais

- Uma outra possibilidade consiste na abordagem por **subconjunto de linguagem** (*Language subset approach*)
 - ◆ É ainda usada com muita frequência no desenvolvimento de sistemas embutidos
 - ◆ Define um **conjunto de regras** para **capturar** os **elementos construtivos** das máquinas de estados em elementos construtivos equivalentes da linguagem sequencial pretendida
 - ◆ Usada quer para **implementações** em **software** (e.g., C, C++, Java, ...) quer em linguagens de descrição de **hardware** (e.g., VHDL)
 - ◆ Problemas
 - ◆ **Custo elevado** em termos de capital humano – definir a especificação e efectuar a escrita de código (duas tarefas!)
 - ◆ **Sujeito a erros** - especificação correcta e implementação errada! Factor humano é crítico neste aspecto
 - ◆ **Consistência difícil de manter** – alterações ao projecto têm de ser reflectidas na especificação e na implementação

Language subset approach

Na abordagem por subconjunto de linguagem os passos gerais (adaptado à linguagem “C”) a seguir são:

- **Enumerar** todos os **estados** (#define)
- **Declarar** a variável de estado e **inicializá-la** com o valor correspondente ao **estado inicial**
- Uso de um **único “switch”** para seleccionar o **estado actual**
- Em cada “case” definem-se as **acções**
 - ◆ e.g. up, down, open, timer_start
- Cada “case” verifica as **condições** de transição para calcular o estado seguinte
 - ◆ e.g. if(...) {state = ...;}

Template geral

```
#define S0 0
#define S1 1
...
#define SN N
void StateMachine() {
    int state = S0; // or whatever is the initial state.
    while (1) {
        switch (state) {
            S0:
                // Insert S0's actions here & Insert transitions  $T_i$  leaving S0:
                if(  $T_0$ 's condition is true ) {state =  $T_0$ 's next state; /*actions*/ }
                if(  $T_1$ 's condition is true ) {state =  $T_1$ 's next state; /*actions*/ }
                ...
                if(  $T_m$ 's condition is true ) {state =  $T_m$ 's next state; /*actions*/ }
                break;
            S1:
                // Insert S1's actions here
                // Insert transitions  $T_i$  leaving S1
                break;
            ...
            SN:
                // Insert SN's actions here
                // Insert transitions  $T_i$  leaving SN
                break;
        }
    }
}
```

Exemplo

Controlo de um elevador

```
#define IDLE0
#define GOINGUP1
#define GOINGDN2
#define DOOROPEN3
void UnitControl() {
    int state = IDLE;
    while (1) {
        switch (state) {
            IDLE: up=0; down=0; open=1; timer_start=0;
                if (req==floor) {state = IDLE;}
                if (req > floor) {state = GOINGUP;}
                if (req < floor) {state = GOINGDN;}
                break;
            GOINGUP: up=1; down=0; open=0; timer_start=0;
                if (req > floor) {state = GOINGUP;}
                if (!(req>floor)) {state = DOOROPEN;}
                break;
            GOINGDN: up=0; down=1; open=0; timer_start=0;
                if (req < floor) {state = GOINGDN;}
                if (!(req<floor)) {state = DOOROPEN;}
                break;
            DOOROPEN: up=0; down=0; open=1; timer_start=1;
                if (timer < 10) {state = DOOROPEN;}
                if (!(timer<10)){state = IDLE;}
                break;
        }
    }
}
```

Exercício

Considere um cruzamento regulado por sinais luminosos. Pretende implementar um controlador para este sistema de sinalização:

- **Características do cruzamento**

- 4 estradas ortogonais
- Regulação do transito de viaturas e peões
- Botão de pedido de passagem para peões

- **Trabalho a desenvolver**

- Especificar o controlador usando StateCharts
- Implementar o controlador em linguagem “C” baseando-se na metodologia “*language subset*”
- Sugestões:
 - ◆ Parametrizar todas os aspectos de configuração do sistema (e.g. tempo a verde/estrada, tempo a amarelo, ...)
 - ◆ usar Linux/gcc;
 - ◆ Simplificar! Não são necessários gráficos etc...