7. Машини на състояния – варианти – Резюме

Проблеми при стандартните машини на състояния:

-Големият брой състояния дори при прости системи може да усложни анализа.

-Необходимост от компактно представяне на състояния и действия:

- Използване на **предикати** за описване на състояния.
- Дефиниране на целите чрез промени в източника.
- Фокус върху текстови описания вместо графични за улеснение на кодирането.

Основни варианти на машини на състояния:

А) Състоянията като функция: Всяко **състояние на M е крайна функция от крайно множество Var от променливи** (“identifiers”, “names”) **към (вероятно) безкрайно множество Val от типови стойности** - крайна частична функция: $\mathbf{M = (\{S : Var \nrightarrow Val\}, I, A, d)}$

Пр.:Множеството от състояния на брояч е тотална функция: с функция на преходите: $\{s:\{x\}\rightarrow \text{int } s(x) \geq 0\}$

с функция на преходите**:** $\mathbf{d}_{inc}=\{ (s,a,s'): S \times\{inc\}\times S\ s'(x) = s(x) +1\}$

Функцията на преходите: множество от тройки (s,a,s′); s е началното състояние;а е конкретно действие; s′ е крайното състояние; **Условия** за преход **Φ; Ψ;** v е вектор от променливите на състоянието.

Това е формално описание на действие в система, където:

$d_{action} =\{ (s,a,s'): S\ \{action\}\ x\ S \mid \mathbf{\Phi}\ [s(v)/ v]\ ) \wedge \mathbf{\Psi}\ [s'(v)/ v' , s(v)/ v]\}$

s – начално състояние; **a** – действие; **s'** – ново състояние след изпълнението на действието; **S** – множеството от всички възможни състояния; **Φ (предусловие)** – трябва да бъде изпълнено, за да се извърши действието; **Ψ (следусловие)** – описва връзката между старото състояние **s** и новото състояние **s'**

Може да бъде представена по-кратко чрез **нотация "тип програмиране"**: **pre-post** спецификация: M = (S, I, A, d) :

action

pre **Φ** (v)

post **Ψ** (v, v′)

action (header) ∈ A

Φ и Ψ са pre and post предикати върху вектора v на променливите на състоянието.

Импликация (→) е основната логическа връзка в pre-post спецификациите, защото **постусловията** зависят от **предварителните** условия: **Φ(v)→Ψ(v,v′)**

Конюнкция (∧) може да участва в дефинирането на сложни предварителни условия (или постусловия - едновременно изпълнение на няколко условия.

Б) Действия с аргументи: Преходите могат да зависят от аргументи: inc(i:int) pre i>0 post x′=x+i

Lambda абстракциите - кратко и формално описание на действията

Действия с резултати: допълнителна структура на действията се дефинира, когато **резултатът от действията са необходими на**

външния наблюдател. Преходът на състоянията е случай, който се описва чрез двойка събития:

а) **извикване** (**invocation event**): името на действието и стойностите на неговите входни аргументи или

б) **отговор (response event)**: името на условието за приключване и стойността на неговия резултат.

Използват се специални думи като:

-ok: За нормално приключване.

-result: За върнатата стойност на действието.

-извънредно (exceptional) приключване;

В) Функция на преходите– недетерминизъм: Когато **има повече от едно възможно следващо състояние** при еднакво действие и еднакво изходно състояние, то функцията на преходите трябва да свързва множество от състояния: $d (s_k , action) = \{s_1 ', s_2 ',\ldots, s_n ' \}$

Недетерминизмът - дизюнкция (или) в post състоянието.

inc(i:inc)

pre i > 0

post $x' = x + i \vee x' = 2i$

Г) Съвместно използване на разгледаните случаи

Общият шаблон, прилаган при описание с **pre-post нотация** на всяко действие action в A от M = (S, I, A, d), е:

action(inputs)/ $term_1$ ($output_1$), …, $term_n$ ($output_n$)

pre Φ (v)

post Ψ (v, v′)

където **inputs** е списък на аргументите и техният тип, **$term_i$** - името на i-тото условие за приключване и **$output_i$** е типът и резултатът, съответстващ на **$term_i$** .

Φ и Ψ са **предикати** на състоянието върху вектора v.

• ok – използва се в header за нормално прекъсване

• result – използва се в post-condition за върната стойност

• terminates – използва се в post-condition за стойност на условието за приключване. Стойността е указана в header.

• empty

Други машини на състоянието, често използвани в практиката

A)Детерминистични крайни автомати (DFSA) M = (S, I, F, A, d):

S е крайно (ограничено) множество от възможни състояния;

I, I $\subseteq$ S е множество от нач. състояния (обик. единично- **singleton set**);

F $\subseteq$ S е **крайно** множество от **крайни/последни** състояния;

A е **крайно** множество от действия;

d $\subseteq$ S x A $\nrightarrow$ S е детерминистична част. ф-я за преходи м/у състояния

trace † - пътека, която завършва с последно състояние.

Характеристики: Винаги има едно следващо състояние за всяко входно действие (ако е дефинирано). Пътеката (trace) се нарича завършена, ако завършва в крайно състояние (елемент от F). Езикът на автомата L(M) (множеството от приемливи входни низове) може да бъде безкраен, ако има безкрайно много възможни входове, които довеждат до приемливо състояние.

Б)Крайни изпълнения и безкрайно поведение(finite-trace model) - модел, включващ (само) ограничени пътеки - Това е **модел на изпълнение**, който разглежда **само крайни пътеки**: Машините са с поведение, съставено с **безкрайно множество** от **крайни пътеки**:

-Опростява разсъжденията; Практическа резонност – безкрайното изпълнен. не може да се види; Недостатък – невъзможност да опише deadlocks, fairness. За тази цел се изисква усложняване на структурата на пътеките и поведението

Разлика с DFSA: Докато **DFSA допуска както крайни, така и безкрайни пътеки**, този модел **ограничава** анализа само до **крайни пътеки** (finite traces); Подходящ за системи, където анализираме изпълнение **до определен момент** и не разглеждаме **непрекъснато работещи системи**.

В) Машини с безкраен брой крайни пътеки -Това са **автомати или системи, които имат безкрайно множество от крайни изпълнения**. Разлика с finite-trace model:

- Finite-trace model разглежда отделни крайни пътеки
- Тук говорим за цяла машина, чийто модел на поведение е дефиниран чрез всички крайни пътеки, които може да има.

Пример: Ако DFSA работи върху **краен вход** и приключва, той има **крайна пътека**. Ако разгледаме **всички възможни входове и**

всички изпълними крайни пътеки, тогава говорим за **безкраен брой крайни изпълнения**.

Разсъждения върху МС:

• **Най-важната характеристика на МС е инвариантността**: предикати Q, които са верни за всички достижими състояния(брояч)

1/Индукция върху състоянията от изпълненията**.** Удобна е когато има **рекурсивна структура на домейна:**

Нека има **изпълнение**: $\langle s_0, a_1, s_1, a_2, \ldots, s_{i-1}, a_i, s_i, \ldots\rangle$. За да се докаже, че х-ката Ξ е **инварианта**, необходимо е за всяко изпълнение:

1/ Основен случай: Показва се валидност за началното състояние s_0
2/ Индуктивна стъпка: Приема се валидност за състояние s_{i-1} и се доказва за състояние s_i

2/Предикатът, който формира на състоянията, е по-силен от инвариантността, която се стремим да докажем **P** $\Rightarrow$ Ξ

3/Доказателсво чрез правило върху pre/post спецификацията

1. Показва се, че Ξ е истинен за всички начални състояния

2. За всяко действие Приема се, че

- pre-условието **Φ** е в сила в pre-състоянието,

- инварианта **Ξ** е валидна за pre-състоянието

- post-условието **Ψ** е в сила в pre и post състоянията

-показва се, че **- Ξ е валиден и в post - състоянието** следователно **Ξ е инварианта**