

10. Процеси и нишки -Резюме

Процес: Единица на последователно изпълнение, която включва действия в определен ред.

Нишка (Thread): Лека версия на процес, която споделя памет с други нишки в рамките на един процес.

Комплексните системи се структурират като набор от по-прости дейности, всяка представена като **последователен процес**.

Процесите могат да се припокриват или да бъдат конкурентни, за да отразяват паралелизма в реалния свят или за управление на комуникации и други устройства.

Крайни автомати (Finite State Processes - FSP): Моделират процесите като поредици от действия.

- превключвател: SWITCH = (on -> off -> SWITCH)

Системи с етикетиран преход (Labelled Transition Systems- LTS):

Графично представяне на FSP, което показва състояния и преходи.

Моделите се описват с помощта на **крайни състояния**, известни като **LTS**. Тези системи се представят по два начина:

-Текстов алгебричен вариант (FSP - Finite State Processes): чрез математически запис. Използва се за точно дефиниране на преходите между състоянията. Подходящ за прецизно моделиране на процеси

-Графичен вариант (LTS): Визуално предст. с-мата като граф: **състоянията** са представени като **възли**; **Преходите** между състоянията - като **стрелки с етикети**; Позволява по-лесно разбиране и визуализация на системата

Процесът -изпълнение на последователна програма. Модел: **крайна машина на състоянията(КМС)**, която преминава от едно състояние в друго чрез изпълнение на поредица от атомарни действия.

-Всяко състояние описва **текущото положение на изпълнението**

-Преходите между състоянията стават чрез **прости, неделими (атомарни) действия**

-Моделът позволява точно проследяване на логиката на изпълнение

-Крайнният брой състояния осигурява възможн. за пълен анализ

FSP- префикс на действие: Когато **х** е действие, а **Р** е процес, **префиксът (х-> Р)** описва процес, който:

1. Първоначално извършва действието **х**
2. После се държи точно както е описан от процеса **Р**

ONESHOT = (once -> STOP) - КМС (терминиращ процес). Извършва действието "once" и спира (STOP)

- Действията започват с малки букви
- Процесите започват с главни букви

Префиксирането е да се моделира последователност от действия, където всяко следващо действие зависи от предишното.

Модел на процес с **алтернативни действия**: **(х-> Р | у-> Q)**

-Първоначално процесът може да извърши или действие **х**, или **у**

-Първото извършено действие определя по-нататъшното поведение:

- Ако първото действие е **х**, процесът продължава като **Р**

- Ако първото действие е y , процесът продължава като Q

Моделира **недетерминистично** поведение. Първоначален избор между две възможни действия. Различно продължение в зависимост от избраното първо действие

FSP модел на машината с четирите цветни бутона: Само жълтият бутон предизвиква действие "candy". Моделът е недетерминистичен - може да се избере произволен бутон при всяко състояние. След натискане на бутон, машината се връща в същото състояние:

**FAULTY = ({red,blue,green}-> FAULTY
|yellow -> candy -> FAULTY).**

Буфер- приема стойности в диапазона 0 до 3 и след това ги извежда:

**BUFF = (in[i:0..3]->out[i]-> BUFF) or
BUFF(N=3) = (in[i:0..N]->out[i]-> BUFF).**

Индексите - за дефиниране на локални процеси и действия, които позволяват гъвкаво моделиране чрез параметризация.

Индексни процеси и действия: Локалните процеси с индекси са еквивалентни на дефиниране на **отделен процес за всяка стойност на индекса** т.е. можем да използваме диапазони и индекси, за да моделираме различни вариации на процесите и техните действия.

Декларации на константи и диапазони:

const $N = 1$ (N е константа със стойност 1)

range $T = 0..N$ (T е диапазон от 0 до N (в този случай 0..1))

range $R = 0..2*N$ (R е диапазон от 0 до $2*N$ (в този случай 0..2))

SUM = (in[a:T][b:T] -> TOTAL[a+b]),

TOTAL[s:R] = (out[s] -> SUM).

SUM: Процесът очаква две входни действия in[a][b], където a и b са стойности от диапазона T (0..1). След като получи тези стойности, изчислява тяхната сума a+b и преминава към състояние TOTAL[a+b].

TOTAL[s:R]: системата изпълнява действието out[s], където s е стойност от диапазона R (0..2), след което се връща към SUM.

Guarded actions (охранявани действия) са механизъм в **FSP**, който използва условия (guards), за да управлява кои действия могат да бъдат изпълнени в даден момент- позволява моделирането на поведението на системата да бъде обвързано с тек. състояние и зададени условия.

when B x -> P | y -> Q -Ако условието B е **истинно** (true), тогава действията x и y са **достъпни** за избор. Ако условието B е **неистинно** (false), тогава действието x **не може да бъде избрано**. В този случай само y остава допустимо, ако няма друга охрана.

(COUNT):

COUNT (N=3) = COUNT[0],

COUNT[i:0..N] = (when(i<N) inc->COUNT[i+1]
|when(i>0) dec->COUNT[i-1])).

Деклариране на брояч: COUNT представлява брояч, който може да се увеличава (inc) или намалява (dec). Нач. състояние е COUNT[0].

Състояния на брояча: Броячът може да има стойности i в диапазона 0..N, където N е зададена константа (в примера N=3).

Охранявани действия: $\text{when}(i < N) \text{ inc} \rightarrow \text{COUNT}[i+1]$: Действието inc (увеличаване) е достъпно само ако текущата стойност на брояча i е по-малка от максималната стойност N . След увеличаването броячът преминава в състояние $\text{COUNT}[i+1]$. $\text{when}(i > 0) \text{ dec} \rightarrow \text{COUNT}[i-1]$: Действието dec (намаляване) е достъпно само ако тек. ст-ст на брояча $i > 0$. След намаляване броячът преминава в $\text{COUNT}[i-1]$.

Process Alphabets (Азбука на процеса): множеството от действия, в които процесът може да участва. Тези действия дефинират взаимодействията на процеса с външната среда. **Азбука на процеса-списъкът с действия**, които са част от дефиницията на процеса.

Имплицитно дефиниране: Азбуката на процеса **не се дефинира ръчно**, а автоматично се извежда от действията, които са описани в дефиницията на процеса.

Извеждане на азбуката: Азбуката на даден процес може да бъде визуализирана с помощта на **LTSA** в прозореца "Alphabet"

Process Alphabet Extension (Разширение на азбуката на процеса): позволява да се добавят допълнителни действия към **имплицитната азбука** на процеса, които иначе не са включени в неговата дефиниция. По подразбиране, азбуката на процеса се състои от действията, дефинирани в него. **Разширението на азбуката** позволява **ръчно добавяне** на допълнителни действия, които не са изрично описани в логиката на процеса- с оператора $+\{\}$.

$\text{WRITER} = (\text{write}[1] \rightarrow \text{write}[3] \rightarrow \text{WRITER}) + \{\text{write}[0..3]\}.$

Имплицитна азбука: Без разширение, азбуката на процеса включва само действията, дефинирани в логиката на процеса: $\text{write}[1]$, $\text{write}[3]$

Разширение на азбуката: С добавянето на $\{write[0..3]\}$, азбуката на процеса се разширява така, че да включва **всички действия в диапазона $write[0]$ до $write[3]$** , дори ако някои от тях не са дефинирани в логиката на процеса.

FILTER в FSP - обработва стойности v в диапазона **0 до 5**:

$FILTER = (in[v:0..5] \rightarrow DECIDE[v]),$

$DECIDE[v:0..5] = (when (v \leq 2) out[v] \rightarrow FILTER \mid when (v > 2) FILTER).$

Моделиране на процеси като крайни автомати(КА)

Процесът е единица на конкурентност, която представлява изпълнението на дадена програма. Процесът е независима единица, която може да се изпълнява паралелно с други процеси.

LTS (Labelled Transition Systems): моделиране на процеси като **крайни автомати**, където: 1/Състоянията представляват различни етапи на процеса. 2/Преходите между състоянията са последователности от **атомарни действия**.

FSP : Език за специфициране на процеси с помощта на:

- **Префикс “ \rightarrow ”:** Определя следващото действие.
- **Избор “ \mid ”:** Позволява избор между няколко действия.
- **Рекурсия:** Позволява повтарящо се поведение.

Нишки в Java: Нишките (threads) се използват за имплементиране на процеси. Нишката е лека единица на изпълнение, която може да се създава, стартира и спира динамично. Thread Lifecycle:

-CREATED (Създадена): създадена, но все още не е стартирана.

-RUNNING (Изпълнява се): Нишката изпълнява задачата си.

-RUNNABLE (Готова за изпълнение): Нишката е готова за изпълнение, но изчаква ресурс от процесора.

-NON-RUNNABLE (Неизпълнима): Нишката временно е в изчакване

-TERMINATED (Прекратена): Нишката е приключила изпълнението си и не може да бъде рестартирана.