9.Моделиране последователностна динамика с FSP-нотация- Резюме

FSP (Finite State Processes) е **формален език за специфициране на конкурентни системи**, използван в комбинация с **LTSA (Labelled Transition System Analyser)**. Той описва **процеси**, които комуникират помежду си чрез **събития** и се представят като **крайни автомати**. FSP е използван основно за **моделиране на конкурентни и синхронизирани системи**, като клиент-сървър, паралелни процеси

Основни концепции в FSP:

Дефиниране на процес: PROCESS_NAME = (action1 -> action2 -> PROCESS_NAME).

Паралелни процеси (||)

Синхронизация на процеси (:: и | оператори)

Последователна програма: **един поток** на управлен. (един процес)

Конкурентна програма: множество потоци на управление (нишки):

- **Извършват множество изчисления паралелно.**
- **Контролират множество външни събития, случващи се едновременно.**

CLIENT = (call -> reply -> continue -> CLIENT).

SERVER=(accept->service->reply->SERVER)/{call/accept}\{service}.

|| CLIENT_SERVER = (CLIENT || SERVER)

Локално взаимодействие: На една машина чрез споделена памет.

Мрежово взаимодействие: Между множество машини чрез мрежови протоколи.

Необходимост от конкурентно програмиране: увеличаване на **производителността** -множество процеси могат да се изпълняват паралелно, използвайки многоядрени процесори; по-добра **отзивчивост**-входно-изходните операции не блокират цялата програма; **по-подходяща структура** -за програми, които управляват множество устройства и събития.

Примери за грешки в конкурентното програмиране:

Therac-25: Комп.машина за лъчелечение, при която грешки в конкурентното програмиране водят до сериозни наранявания.

Mars Rover: Проблеми в взаимодействието на задачите водят до рестартиране на софтуера и загуба на време за изследвания.

Моделиране на конкурентни системи с **LTS/FSP**: модели за представяне на реалния свят и проверка на адекватността на дизайна.

Етикирани преходни системи (Labelled Transition Systems, LTS) - моделите се описват с помощта на **състояния и преходи**-всяко състояние на системата се свързва с възможни действия или събития, а преходите между състоянията са етикетирани с тези действия. Описанието е:1/**текстово - Процеси с крайни състояния (Finite State Processes, FSP-** език за **описание на динамично поведение на системи**, като се дефинират състояния и преходи между тях. 2/Визуализ. и анализиране- **LTSA** Labelled Transition System Analyzer**.**

Процесите и **нишките (**Threads**)**-осн. концепции в многозадачността (concurrent execution) и многопоточността (multithreading).

Процес-независим изпълняващ се **модул**, който има **собствено адресно пространство и ресурси, като памет и регистри**.

Нишка (поток)-по-малка единица на изпълнение в даден процес. Нишките споделят ресурси и адресно пространство с останалите нишки в същия процес.

Конкурентното изпълнение (Concurrent Execution)- множество процеси или нишки се изпълняват "паралелно" в една операционна система- могат да се изпълняват по ред, като операционната система редува между тях, като предоставя време на всяка нишка или процес.

Споделени обекти и интерференция (Shared Objects & Interference(намеса)**):** Когато нишки или процеси имат достъп до **споделени обекти** (напр. глобални променливи или ресурси), може да възникне **интерференция**-едновременното четене и записване върху споделени обекти води до неочаквани или некоректни резултати-трудни за откриване грешки (interference bugs).

Мониторите са синхронизационни механизми, които осигуряват безопасен достъп до споделени ресурси. Те предоставят структури за управление на достъпа до споделени обекти, като осигуряват **взаимно изключване** (**mutual exclusion**) и възможност за **синхронизация на условия** (condition synchronization). **Мониторът** осигурява, че само един поток може да изпълнява даден код в даден момент, когато се използва споделеният ресурс.

Синхронизация на условията - да се осигури, че един поток ще изчака, докато не бъдат изпълнени опр. условия (ресурсът стане наличен).

Задръжка (Deadlock): два или повече потока не могат да напредват поради взаимно изчакване на ресурси, които са задържани от други потоци. Ако поток А държи ресурс X и чака ресурс Y, докато поток Б

държи ресурс Y и чака ресурс X, това води до задръжка, тъй като никой от потоците не може да продължи.

Свойства на безопасността- условия, които гарантират, че системата няма да достигне некоректни или нежелани състояния. Пр: да не се позволява на два потока да извършват операции върху един и същи ресурс по едно и също време, ако това води до грешки.

Свойства на живостта- гарантират, че системата няма да попадне в ситуация, в която не може да извършва напредък. Включва избягване на задръжка и гарантиране, че всяка нишка или процес ще получи възможност за изпълнение след определено време.

Моделно-базирано проектиране (Model-based Design) - процесите и системите се моделират предварително чрез математически модели, преди да се реализират в код

Динамични системи (Dynamic Systems) - чиято **конфигурация и поведение се променят с времето**, въз основа на външни или вътрешни фактори. Динамичността може да се отнася до промени в състоянието на системата или взаимодействията между различни компоненти, които се развиват по време на изпълнението.

Изпращане на съобщения (Message Passing) е техника, използвана за комуникация между процеси или нишки в конкуретни или паралелни системи. Процесите или нишките обменят съобщения- всяка страна изпраща информация към другата по определен начин, обикновено чрез канали за комуникация или съобщителни буфери.

Конкурентен софтуер - включва паралелни процеси или нишки, които могат да се изпълняват едновременно, но в един и същ компютър или разпределена система.

Конкурентните архитектури - дизайна на хардуер и софтуер, които поддържат едновременни операции, като използват **многопроцесорни системи** или **многоядрени процесори**. Включва синхронизация, съвместен достъп до ресурси и управление на паралелни операции.

Времеви системи (Timed Systems)- времето играе важна роля за тяхното поведение. Включват механизми за управление на времето - ограничаване на времето за изпълнение на операции или обработка на събития, които трябва да се случват в точно определен ред. Използват се в вградените системи, мрежовите протоколи и управлението на процеси в реално време.

Програмно верифициране (Program Verification) - процесът на доказване на коректността на програми- математическо доказателство, симулация, верификационни инструменти

Логически свойства (Logical Properties)- математическите х-ки и условия, които трябва да бъдат изпълнени от програмата - логически изрази и правила, които дефинират вярността на програмата. Пр.инварианти, правилата за доказателства и типови системи.