Модели на софтуерни системи

доц. Олга Георгиева

СУ ФМИ

катедра “Софтуерни технологии”

Модели на софтуерни системи

Лекция 6: Машини на състоянието - основи

Олга Георгиева

СУ, Факултет по Математика и информатика,
Катедра СТ

Z нотацията и машините на състоянието

- **Инвариантата на състоянието се приема да е валидна по време на операциите.**
  - –so don't have to prove as it is maintained
- **Операциите извън предварителните условия водят да “хаос”**
  - –so always want final precondition to be "true"
- **Pre- и post-условията не са разграничени синтактично**
  - –but may be derived
- **Фокусира се върху състояния, а не върху трасета**
- **Входовете, изходите, изключенията са просто нотирани конвенции**

Проблеми на подхода

- Ok за прости машини на състоянието, но ...
- ако системата има безкрайно много състояния?
- Глобалните променливи не са скалируеми
- А параметрите на системата?
- А изключенията?

Основни видове формални методи за спецификация на софтуер

- **Model-based notations**

  Z *(Sprivey, 1989), Vienna Development Method (VDM) (Jones, 1990), B method*

- **Process algebras-based** notations

  Communicating Sequential Processes *(CSP) (Hoare, 1985), CCS (Milner, 1989) and LOTOS (Bjorner, 1987)*

Машина на състоянието (МС) – основна идея

- Машината на състоянието описва идеята, че **движението на системата преминава през множество от състояния, които реализират/отговарят на множество от действия**.

- Две ключови концепции
  - **Състояния** (States)
  - **Действия** (Actions)

- Машината на състоянието трябва да може да формулира:
  - възможните състояния;
  - кое е възможното начално състояние(я);
  - възможните действия;
  - как се променят състоянията, когато има дадено действие;

Машини на състоянието (МС) - представяне

- **Начини за описание на машини на състоянието** (**state machines**)
  - **графично**
  - **като наредени 4-ки**
    - (1) изброяване
    - (2) използване на предикати
  - **чрез специфична нотация**
    - (1) pre- & post-условия
    - (2) вход (input), изход (output), изключения (exceptions)
- **Начини за разсъждение**
  - удовлетворение на инвариант
  - достижимост

МС като инструмент за моделиране

Как се моделира системата чрез **Модели на състоянието**.
Основни теми:

- **МС – основи (State Machine Basics**)

- **МС – вариации (State Machine Variations)**
  моделиране на сложни системи, променливи и действия, недетерминизъм)

- **Разсъждения с МС (Reasoning about State Machines**
  инварианти и ограничения)

- **Графично представяне (Statecharts)**
  специфичен, популярен графичен подход).

Машина на състоянието (МС) – основна идея

• **МС е прост математически модел** – основен и широко разпространен в КН.

• **Цел**: Представянето на сложните машини **с по-прости (абстрактни) машини,** използвайки определени техники за**:**

– *нотация*: стандартна и специфична;
– *абстракция*: ниво на детайлизиране;
– *модуляризация*: композиция и декомпозиция.

• **Не съществува единен (общ) модел за описание** с МС в КН или СИ**.** Прилагат се варианти, съобразени със спецификата на решавания проблем**.**

• **Нивото на точност** в специфицирането зависи от нуждите на описанието и субективното отношение на разработчика.

Машина на състоянието (МС) – основна идея

Както всеки модел, МС разглежда само тези детайли, които са моделирани!

“If a state machine doesn’t let you model the cost of the system, then you won’t be able to reason about how expensive or cheap it will be to build.”

Използване:

- Когато лесно могат да се резюмират и отделят (несвързани) детайли, които да описват определен брой състояния;

- За изследване и оценка на всяка възможност чрез проверка с модел;

- ***Примери***: При описание на *комуникационни* протоколи; сложни *разпределени* алгоритми; *интерфейси*.

Какво е състоянието – илюстративни примери

Всички компютърни системи са машини на състоянието!

- Моментното състояние на системата: *стойности на паметта и регистрите***;**

- Множество от стойности на променливите – *моментните стойности на данните на текущата програма*;

- Управляваща локация/и – *моментно място на програмата в своята последователност на изпълнение*;

- Съдържание на комуникационните канали – *моментното състояние на комуникацията.*

Машина на състоянието: основен модел

- *Определение*:

Машина на състоянието M е наредена четворка (S, A, I, δ), където:

S е множество от възможни състояния (крайно или безкрайно);

I, I $\subseteq$ S е множество от начални състояния (крайно);

A е множество от действия (крайно или безкрайно(!));

$\delta \subseteq$ **S x A x S** е *релация* на преходите.

- Пример:

Car's State Transition Diagram

Ако колата се разглежда като черна кутия, то как ще се формира интерфейсът ѝ' с околната среда?

Множеството **A** понякога се нарича **азбука на M**, а действията - “*събития*”, “*преходи*”, “*етикети*” (LTS),;

Частни случаи:

а) Ако **S** е крайно множество, то **M** е **крайна машина на състоянието;**

б) **I** и/или **A** са безкрайни множества.

в) δ е функция:

Когато δ е **релация, то** δ**: S x A** $\leftrightarrow$ **S**

Когато δ е **функция, то** δ**: S x A** $\rightarrow$ **S**

Допълнения

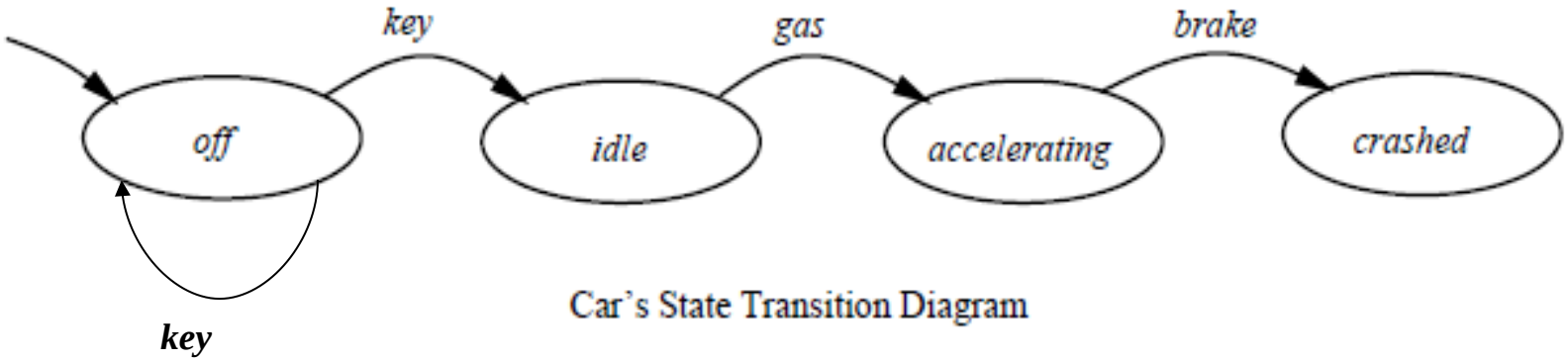
Преход (стъпка): Ако **(s1, a, s2)** е в δ**.** Тогава съществува отбелязана стрелка с етикет **a** от **s1** до **s2;**

Недетерминизъм

Едно действие може да причини непредсказуеми алтернативни преходи от едно и също състояние – недетерминизъм

Когато δ е **релация δ: S x A ↔ S недетерминистична МС**

Когато δ е **функция δ: S x A → S** , то **M** е **детерминистична МС**



Car's State Transition Diagram

Прилагайки определението по-горе дефинирайте МС Car:

Car = ({ …

{

{ …

{ …

)

Машина на състоянието: Основни дефиниции

За машината на състояние **M = (S, I, A, δ) :**

• **Стъпка** в **M:** всяка тройка **(s,a,s')** в δ на **M** ;

• **Изпълнителен фрагмент** е крайна или безкрайна *последователност* (*редица*) на редуващи се състояние и действие $\langle \mathbf{s_0, a_1, s_1, a_2, s_3, \ldots ,} \rangle$, така че за всеки индекс **i,** тройката $\mathbf{(s_i, a_{i+1}, s_{i+1})}$ е стъпка на МС.

• **Изпълнение (execution) е изпълнителен фрагмент,** започващ с начално състояние $\mathbf{s_0}$ на машината **M** .

Примери: Дефинирайте фрагмент или изпълнение са?

⟨ off, key, idle, gas, accelerating, brake, crashed ⟩

⟨ off, key, idle ⟩

⟨ accelerating, brake, crashed ⟩

• За **крайно изпълнение** се дефинира и **крайно състояние** на **M;**

• Състоянието е **достижимо**, ако е крайно състояние при някое крайно изпълнение;

Машина на състоянието: Основни дефиниции

• **Event-based пътека** (trace) (или **action-based пътека**) е последователност от действия на изпълнението.

- ⟨ key, gas, brake ⟩
- ⟨ key ⟩

• **State-based пътека** е последователност от състояния на изпълнение или последователност $\langle \mathbf{s_i} \rangle$ за всяко начално състояние $\mathbf{s_i} \in \mathbf{I}$.

- ⟨ off, idle, accelerating, crashed ⟩
- ⟨ off ⟩

Кой е предпочитаният вид?...

• **Поведението (режим на работа) Beh(M)** на машината **M** е множеството от всички пътеки на **M.**

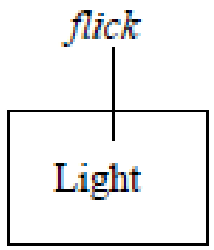
Поведенията (в нашите разглеждания) са **prefix-closed**: а) Празната пътека $\in$ **Beh(M) б)** ако пътека е в **Beh(M),** то всеки префикс на това трасе е в **Beh(M)**.

За всяка МС **M** са възможни *две алтернативни определения* на **Beh(M)**

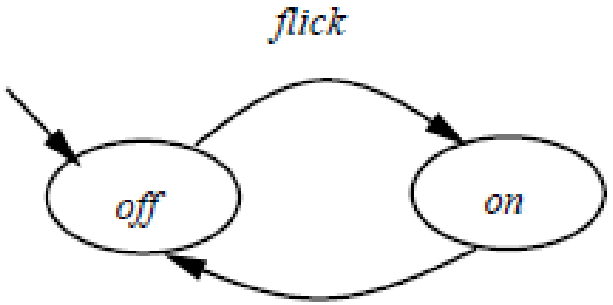
Пр. Beh(Car) =?

Безкрайни машини на състоянието - пример

Може ли крайна МС да има безкрайно изпълнение; а безкрайно поведение?
Може ли безкрайна МС да има крайно поведение?



Light's Interface



Light's State Transition Diagram

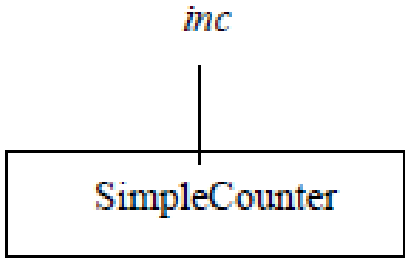
Някои от изпълненията на МС Light:

$$\langle off, flick, on \rangle$$
$$\langle off, flick, on, flick, off \rangle$$
$$\langle off, flick, on, flick, off, flick, on \rangle$$
$$\ldots$$
$$\langle off, flick, on, flick, off, flick, on, flick, off, \ldots, \rangle$$

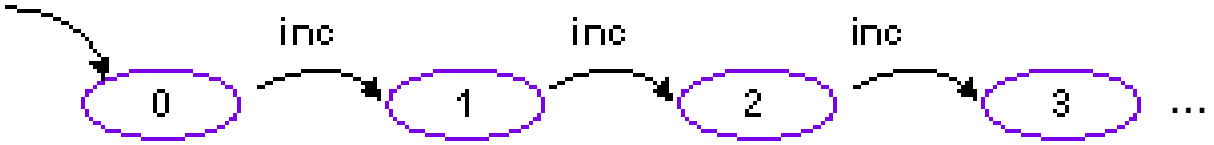
Безкрайни машини на състоянието

• Машините на състоянието може да нямат краен брой състояния т.е. са безкрайни. Защо повечето софтуерни системи са безкрайни?

Пр.: Запишете граф на преходите на Бряч (integer counter)



SimpleCounter`s Interface



Отг.: Възможен запис на машината:

SimpleCounter = (
{0, 1, 2, ...}
{0},
{inc}
{(0,inc,1), (1,inc,2), (2,inc,3), ...}
)

• Но "..." е твърде неточно.

Използване на множества и логика за описание на машини на състоянието:

• Множеството на състоянията: **S** се дефинира по-педантично чрез предикатната логика:

S = { x: Z | x ≥ 0}

и начално състояние

S_0 = { x: Z | x = 0}

• Преходът за ***inc*** може да бъде описан с:

δ_{inc} = **{ (s, a, s'): S x {*inc*} x S | s' = s + 1 }**

Най-общо: $\delta = \bigcup_{a \in A} \delta_a$

• *Пр.:* Компактен запис **SimpleCounter =**

= ({S}, {S_0}, {*inc*},
{ (s, a, s'): S x {inc} x S | s' = s +1 })

Интерфейс и среда

- Много модели на МС се различават само по това какво “наблюдават” в средата.

- Средата също може да бъде моделирана като МС

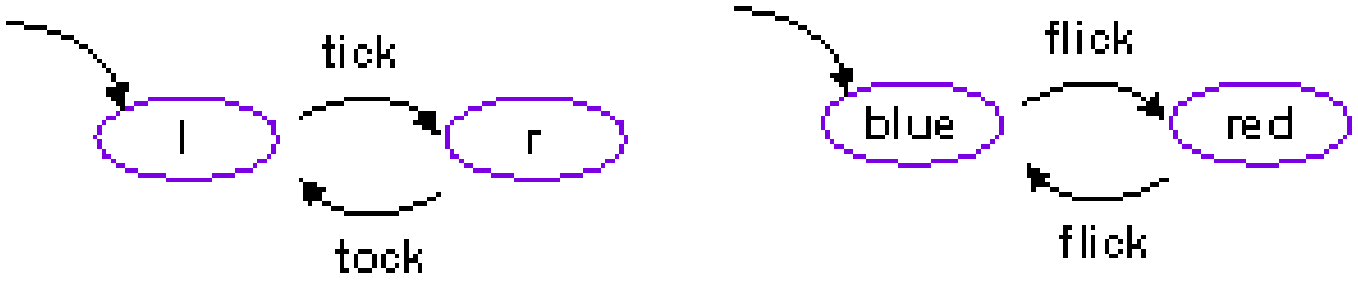
- *Интерфейс на системата:*

- наблюдаемо поведение на взаимодействието на системата със средата:

“Кое е наблюдаемото поведение на системата?” – ТОВА трябва да се моделира.

Входни и изходни действия

• Каква е разликата в тези две машини на състоянието? Какви са действията им? Кои са предпочитаните трасета за тях?



• Машината работи в околна среда като може или **да наблюдава** събитията или самат тя **да предизвиква** събитие.

•Съществува два вида взаимодействия между средата и системата - **входни и изходни действия**. (Пр. Граф на преходите на МС за разпознаване на “010”)

Абстракция

1. Представяме само някое/част от поведението на системата (*пр*. Flick MC, Car MC).

“Може ли средата да наблюдава това различие?”

Пр. “Ябълка, портокал, патладжан” – система за разпознаване.
“ Accelerating” в “Car” модел

2. Решението, какво да бъде обект на абстракция зависи от различни фактори като:

- taste
- what will fit on a page
- what you want to communicate to others
- what you want to reason about
- what can be checked by tools
- experience and practice

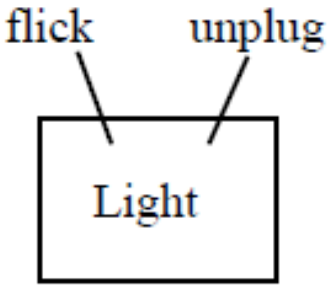
3. Външни и вътрешни действия. Класификация.

Неочаквани действия

Действие, което не може да се случи не е част от интерфейса или е, но не се реализира преход.

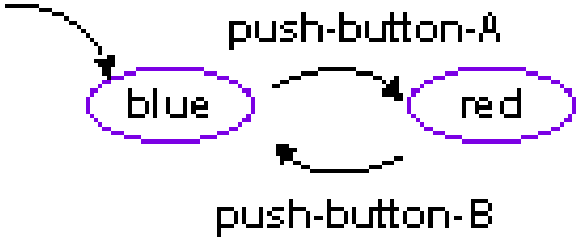
Какво ще се случи, ако натисна бутон **B (също и unplug)**, когато машината е в състояние “**blue**”?

1) Нищо
2) Не е дефинирано – всичко може да се случи
   (състояние “**bottom**” ⊥)
3) Това е **грешка** (и *хаос* може да настъпи).
4) Не може да се случи.



An Unpluggable Light?

Дефинирайте разликата между 2) и 3). Изобразете съответните графи за четирите случая.



Развитие

- Състоянията могат да бъдат огромен брой дори и за много проста система;
- Често се налага по-компактно представяне на състоянията:
  - – Да се опишат като предикат множеството от състояния, от които съществува преход;
  - – Да се опише “целта” чрез промените на източника;
  - – Използване на текст, а не илюстрации (подпомагане на етапа на кодиране);
  - – Фокусиране върху специални аспекти (като пътеки на събития).