



ТЕСТВАНЕ С КРАЕН АВТОМАТ

доц. д-р Десислава Петрова-Антонова

# Съдържание

- ❖ Машина с краен брой състояния (краен автомат): базови концепции
- ❖ Тестване с краен автомат
- ❖ Пример: тестване на уеб приложения с краен автомат

# Машина с краен брой състояния: необходимост



# Тестов модел с една или повече фази

## ❖ Модели с една фаза

- Конструирането на модела е фокусирано върху идентифицирането и инициализирането на входа към системата
- Имплицитно се допуска, че изходът може да бъде получен и проверен с тестов оракул

## ❖ Модели с няколко фази

- Наличие на краен брой фази или списъци
- Всяка фаза или списък са уникални
- Крайното решение (резултат) се определя еднозначно от елементите в списъците или фазите в дървовидната структура, през които се преминава
- Информацията за изпълнението се съсредоточава в листата на дървовидната структура
- Многомерни списъци
  - ✓ Всеки избор е точка в многомерно пространство

# Тестов модел с поддръжка на множество състояния

## ❖ Особенности при обработката на информация в софтуерните продукти

- Поведението на софтуера е свързано с повторение на състояния и изпълнението на задачи в цикъл

## ❖ Моделиране на повтарящо се поведение

- Замяна на точките за взимане на решение в списъците или фази на работа със състояния
- Замяна на избора на елемент от даден списък или взимане на решение с преход между състояния
- Осигуряване на връщане назад към предходно състояние
- Наличие на общи под-операции, които изграждат операциите от високо ниво
  - ✓ Спестяване на ресурси от повторно тестване на под-операции

# Краен автомат: базови концепции

## ❖ Елементи на крайните автомати

- Статични елементи
  - ✓ Състояния и преходи (крайно множество)
- Динамични елементи
  - ✓ Входи и изходи (крайно множество)
    - Разделяне с класове на еквивалентност, ако множеството е безкрайно

## ❖ Поведение на крайните автомати

- Във всеки момент от време крайният автомат е в определено състояние или в преход между две състояния
  - ✓ При игнориране на фактора време, машината винаги е в точно определено състояние, наречено текущо състояние
- Детерминиран краен автомат
  - ✓ Изходът и следващото състояние се определят еднозначно от входа и текущото състояние

## ❖ Графично представяне на крайните автомати (модел на Мили)

- Състоянията се представят с възли на граф
- Преходите се представят с насочени дъги между два възела в граф
- Входелите и изходите се представят с теглови коефициенти или анотации на дъгите в графа

# Машины с краен брой състояния: базови концепции



# Машины с краен брой състояния: примери

## ❖ Изпълнение на програма като краен автомат


- Инициализиращо състояние: стартиране на програмата (1)
- Преход към следващо състояние при използване на потребителска функция или изпълнение на израз или процедура (2)
- Многократно извършване на преходи с възможност за повторение на състояния (3)
- Крайно състояние: приключване на изпълнението на програмата (4)
- При преходите между състоянията може да е необходим вход или да се генерира изход (5)

## ❖ Моделиране на ползване на уеб с краен автомат

- (1) Зареждане на страница по подразбиране или потребителски дефинирана страница
- (2) Щракване върху линк или директно зареждане от адресната лента на страница
- (3) Многократно преминаване между страници
- (4) Затваряне на браузъра или спиране на заявките за нови страници
- (5) Преходите могат да се асоциират със заявки и зареждане на страница, съобщения за грешка и др.



# Конструране на машина с краен брой състояния

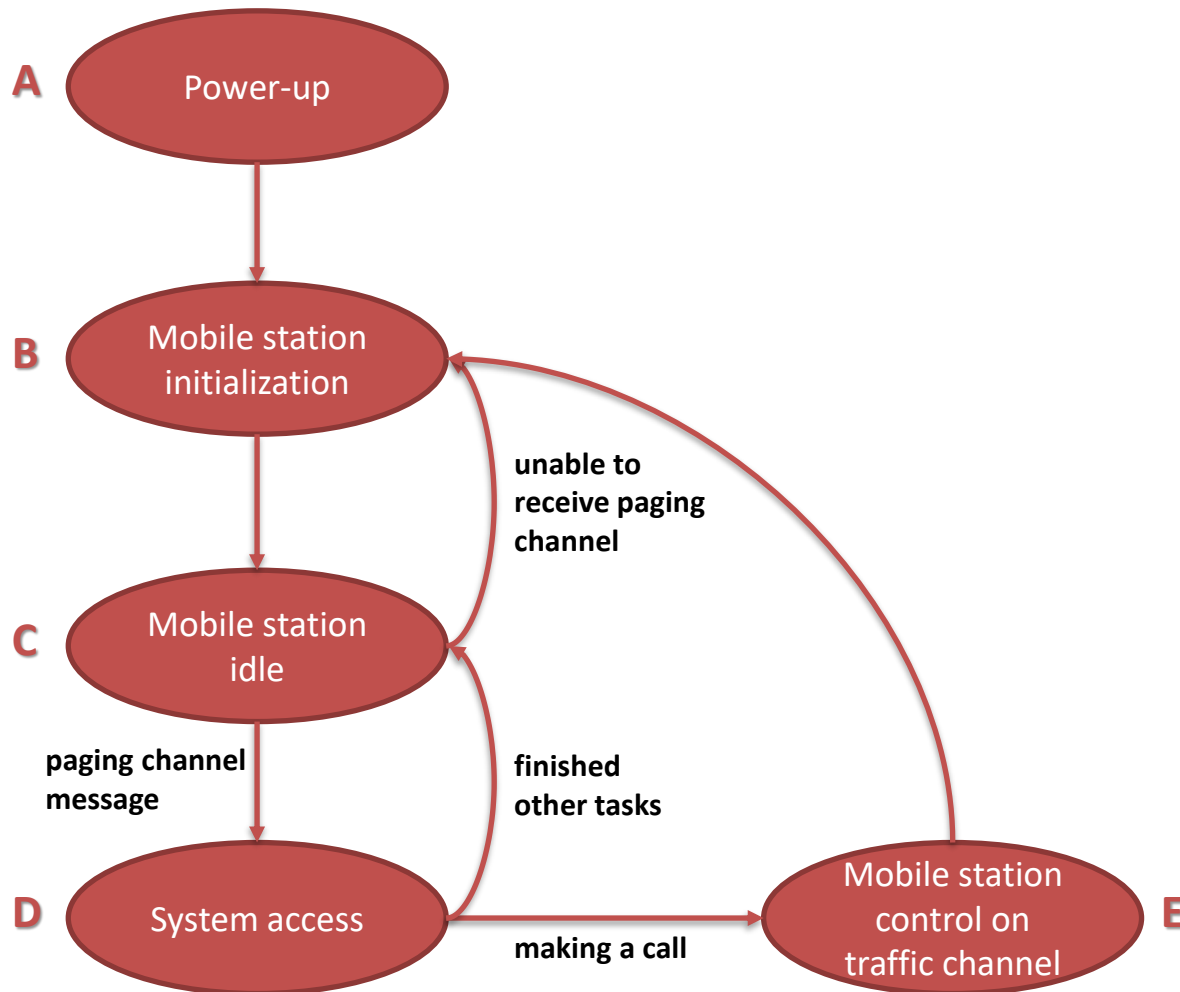


Асоцииране на  
операциите с  
преходи

Асоцииране на  
операциите  
със състояния

# Краен автомат: примери

- ❖ Обработка на разговор при клетъчните комуникационни системи



## Начини за представяне

	A	B	C	D	E
A	na	-/-	na	na	na
B	na	na	-/-	na	na
C	na	noC/-	na	msg/-	na
D	na	na	done/-	na	call/-
E	na	-/-	na	na	na

- ❖ Формално представяне на графа
  - Множество на състоянията: {A, B, C, D, E}
  - Множество на преходите: пример {C, B, “unable to receive paging channel”, –}

- ❖ Таблично представяне

- Състоянията се разписват по редове и колони
- Редовете представят стартови състояния, а колоните – крайните състояния
- Клетките представят входа и изхода за преход от едно състояние в друго
- Клетките, които са празни или маркирани с “na” показват невъзможност за преход между съответните състояния

- ❖ Списъчно представяне, базирано на формалното представяне на граф

- Множеството на състоянията се представя със списък

■ Възможните преходи се представят със списък от вида {C, B, “unable to receive paging channel”, –}



# Откриване на проблеми при с краен автомат

## ❖ Проблеми със състоянията

- Некоректно състояние, определено от некоректно поведение на системата
- Липсващо състояние
- Излишно състояние: наличие на недостижимо състояние или на множество изходни състояния с еднакъв вход

## ❖ Проблеми с преходите

- Липсващ преход
- Излишен преход: наличие на множество преходи до едно и също състояние при един и същ вход
- Некоректен преход, определен от преход към неочаквано състояние или състояние, произвеждащо неочакван изход

## ❖ Проблеми с входовете

- Проблемите с входовете се интерпретират като проблеми със състоянията и преходите
- Разширение с цел прихващане на невалидни входове
  - ✓ Игнориране на невалиден вход посредством запазване на текущото състояние
  - ✓ Диектно прихващане на невалиден вход посредством произвеждане на изход със съобщение за грешка или преход към състояние за прихващане на грешки

## ❖ Проблеми с изходите

- Проблемите с изходите се интерпретират като проблеми с преходите
- Ако преходът произвежда допълнителен или некоректен изход, то той е невалиден

# Конструране на модел

- ❖ Стъпка 1: Определяне на източник на информация и събиране на данни
  - Външна продуктова спецификация или очаквани потребителски сценарии
  - Вътрешна продуктова информация, налична в документите за проектиране на продукта и програмния код
- ❖ Стъпка 2: Конструране на първоначален модел
  - Идентифициране и номериране на състоянията
    - ✓ Използване на вложени или йерархични машини на състоянията в случай на прекалено голям брой състояния
  - Идентифициране на преходите с помощта на входни стойности
    - ✓ Създаване на класове на еквивалентност при много голям или безкраен брой входни стойности
    - ✓ Идентифициране на нови състояния
  - Идентифициране на входно-изходните зависимости
- ❖ Стъпка 3: Подобряване на модела и валидация
  - Идентифициране на липсващи състояния и преходи посредством създаване на контролни списъци, базирани на източниците на информация
  - Идентифициране на излишните преходи и състояния посредством **съпоставяне с информацията в източниците** за създаване на модела или извършване **на анализ за достижимост**
  - Идентифициране на некоректни състояния и преходи

## Дефиниране на тестови сценарии



# Тестване с краен автомат

## ❖ Покритие на състоянията

- Тестовият сценарий е множество от входни стойности, които осигуряват преход от определено начално състояние до определено крайно състояние
- Един тестов сценарий може да осигури пълно покритие на състоянията при наличие на точно едно начално и едно изходно състояние
- При наличие на множество начални и крайни състояния броят на тестовите сценарии нараства

## ❖ Покритие на преходите

- Входелите се класифицират в класове на еквивалентност, от които се генерират тестови сценарии

## ❖ Генериране на входни тестови данни

- Използват се входните стойности, необходими за преход между състоянията
- Прилага се разделяне на входните данни с класове на еквивалентност

## ❖ Изпълнение на тестовите сценарии

- Възможност за съхраняване на “текущо състояние” при стартиране на тестовите
  - ✓ Осигуряване на по-ефективно изпълнение на тестовите, тъй като не винаги се налага стартиране на системата от начално състояние

## ❖ Проверка на резултата

- Използват се изходите, получени след преход към следващо състояние

# Приложения и ограничения

## ❖ Приложения на крайните автомати

- Софтуер с потребителски интерфейс, управляван от менюта
- Системи с ясно идентифицируеми състояния и преходи като системи, работещи в реално време, и управляващи системи
- Обектно-ориентирани системи

## ❖ Ограничения

- Невъзможност за моделиране на голям брой състояния
  - ✓ Използване на йерархични машини с краен брой състояния
- Трудно постигане на покритие при големи системи
  - ✓ Голям брой йерархични крайни автомати
  - ✓ Неравномерно разпределение на проблемите и честотата на използване

## ❖ Възможно решение

- Разширяване на крайния автомат с информация за начина на използване на системата (вериги на Марков)



Тестване на уеб базирани приложения

# CASE STUDY

# Особености на уеб базираните приложения

- ❖ Фокусиране главно върху информацията и документите
  - Традиционният софтуер извършва главно изчислителни операции
- ❖ Смесване на навигацията с изчислителната обработка
  - При традиционния софтуер навигацията и изчислителната обработка са отделени
- ❖ Наличие на множество стартови и изходни точки
  - Традиционният софтуер притежава една стартова точка и ограничен брой изходни точки
- ❖ Голям брой навигационни страници
  - Традиционният софтуер притежава ограничен набор от менюта
- ❖ Необходимост от разнообразна поддръжка за софтуерната архитектура и прилежащата инфраструктура

**Client (Web browser)**

**Web server**

**Middleware**

**Database**

# Какво да тестваме?

## ❖ Дефиниция на уеб повреда

- Неспособност за коректно доставяне на информация или документ (отклонение от очакваното поведение)

## ❖ Източници на повреди

- Хостинг и мрежови повреди
  - ✓ Анализират се със съществуващите техники за системни и мрежови проблеми
- Повреди в браузърите
  - ✓ Повредите се третираат като повреди в софтуерен продукт и се третираат със съществуващите техники за тестване
- Повреди в кода или съдържанието
  - ✓ Тестване на функционалност
  - ✓ Тестване на удобство за употреба (usability)
  - ✓ Тестване на надеждност (reliability)
  - ✓ Уеб компоненти (HTML документ, Java, JavaScript и ActiveX, SGI-Bin скриптове, База от данни, Мултимедийни компоненти)

# Типове тестване при уеб приложенията

## Тестване на специфични аспекти от приложението

Функционално тестване

Тестване за производителност

Тестване на надеждност

Тестване на удобство за употреба

Тестване на натоварването и стрес тестване

Тестване за съвместимост с браузъра

## Тестване за удовлетворяване на потребителските очаквания

Тестване с краен автомат

# Тестване с краен автомат

## ❖ Конструирание на машина на крайните състояния

- Представяне на всяка страница като състояние, при което всяка страница може да бъде начално или крайно състояние
- Представяне на уеб навигацията с преходи
  - ✓ Обикновено преходите се асоциират с хипервръзки предвид непредсказуемостта при директно въвеждане на уеб адрес в браузъра и избор на съхранена страница (bookmarked favorites)
- Входовете се представят като щракване върху хипервръзките
- Изходите се представят като зареждане на страница със съответно съобщение, съдържащо HTML статус, грешка и т.н.

## ❖ Ограничения

- Наличие на голям брой страници, което води до голям брой състояния
- Разпределението на проблемите и използваемостта на различните софтуерни компоненти е неравномерно, което изисква извършване на статистическо тестване, базирано на употреба

