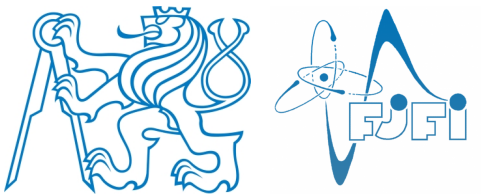
Strukturované testování

Radek Mařík

ČVUT FEL
Katedra telekomunikační techniky, K13132

17. října 2017



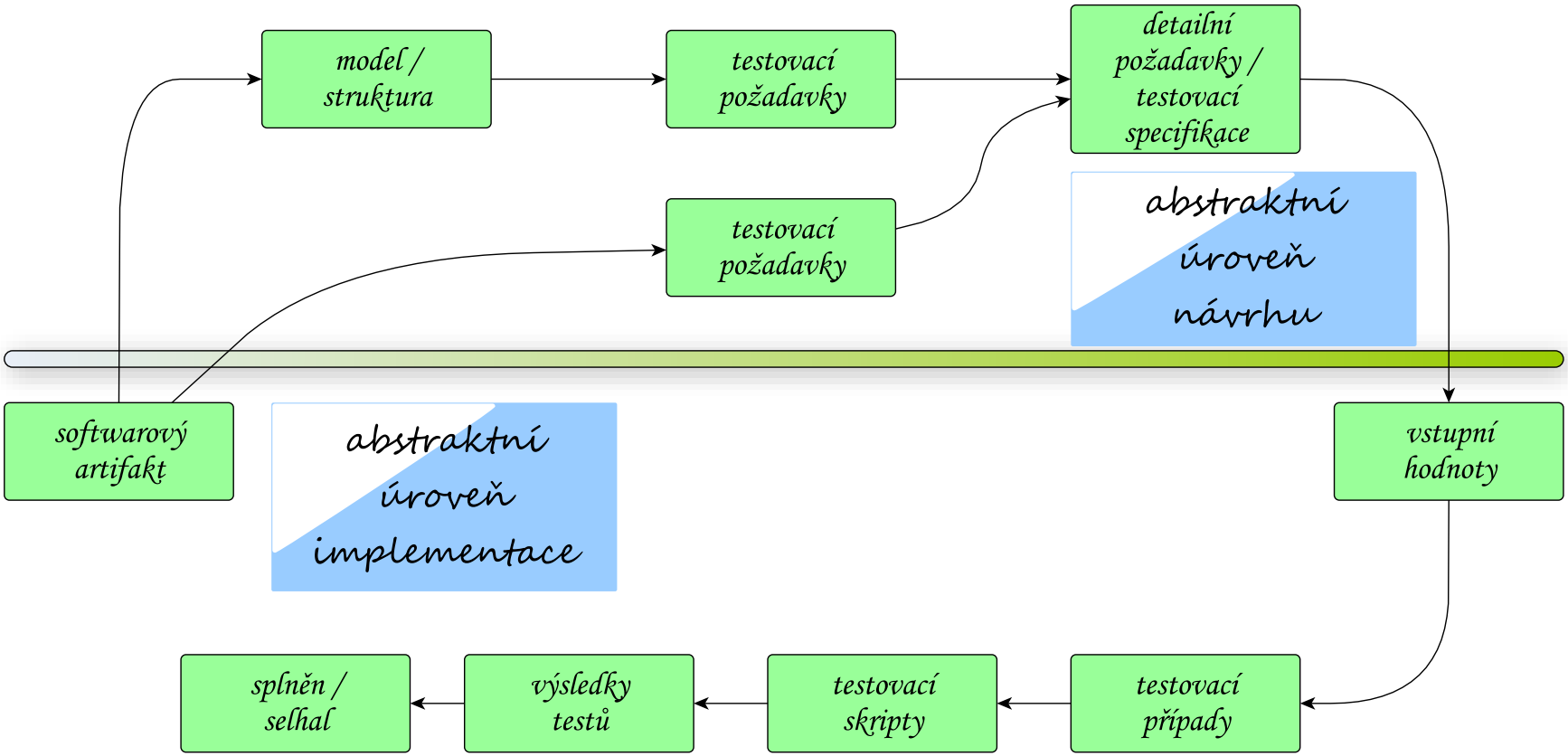
Obsah

1. Návrh testů řízené modelem
   - Principy
2. Testování cest
   - Princip
   - Kritéria pokrytí
3. Použití teorie grafů
   - Úvod
   - Grafové modely softwaru
   - Překlad specifikace do grafu
   - Testování datového toku

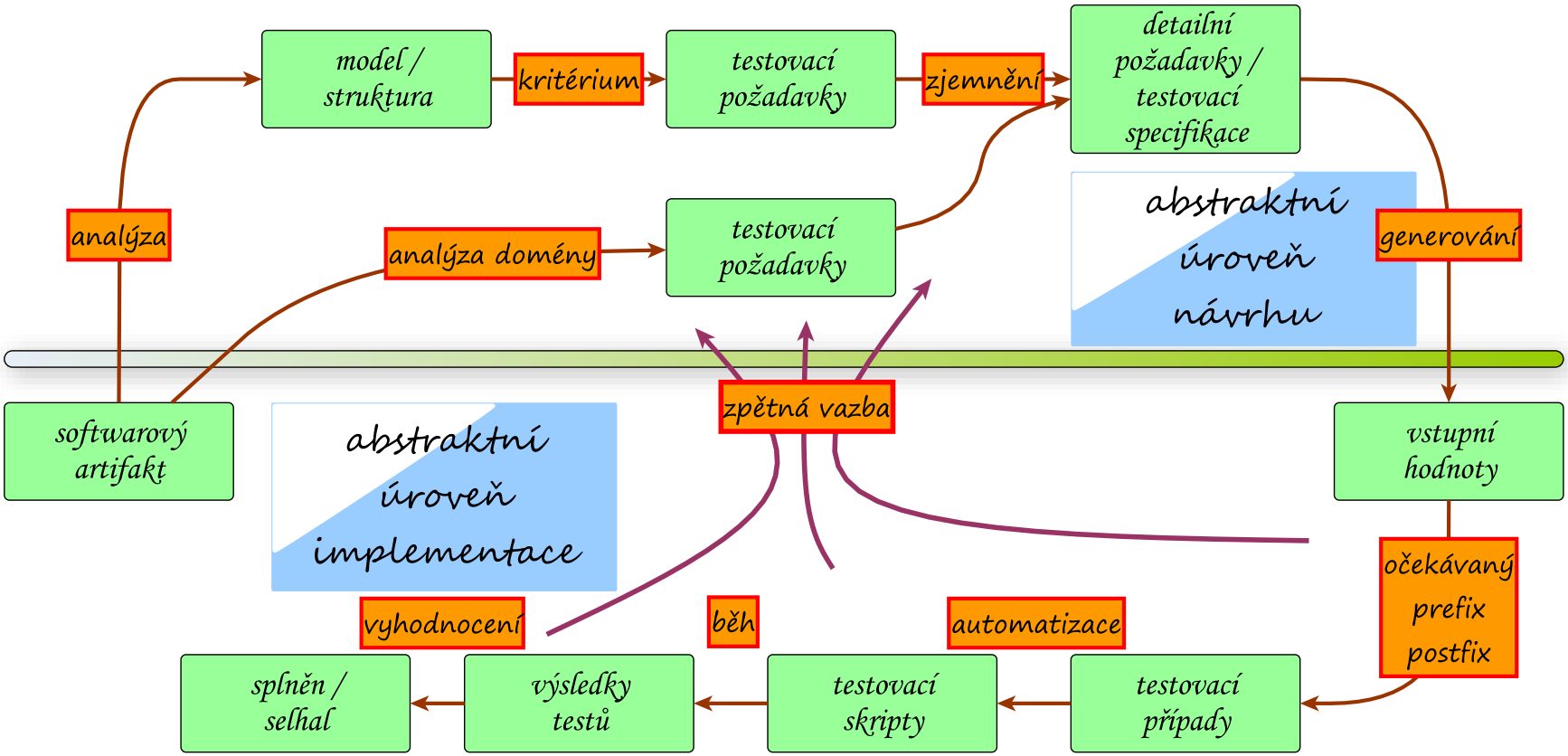
Návrh testů řízené modelem (MDTD) [AO08]

- angl. **M**odel **D**riven **T**est **D**esign
- matematiku provádí jeden návrhář testů
- tradiční testeři a programátoři se soutředí na
  1. nalezení hodnot,
  2. automatizaci testů,
  3. běh testů,
  4. vyhodnocení testů.
- Jako v tradičním inženýrství, inženýr konstruuje modely s pomocí kalkulu, potom předá pokyny tesařům, elektrikářům, technikům, apod.
- Testeři nejsou matematici!

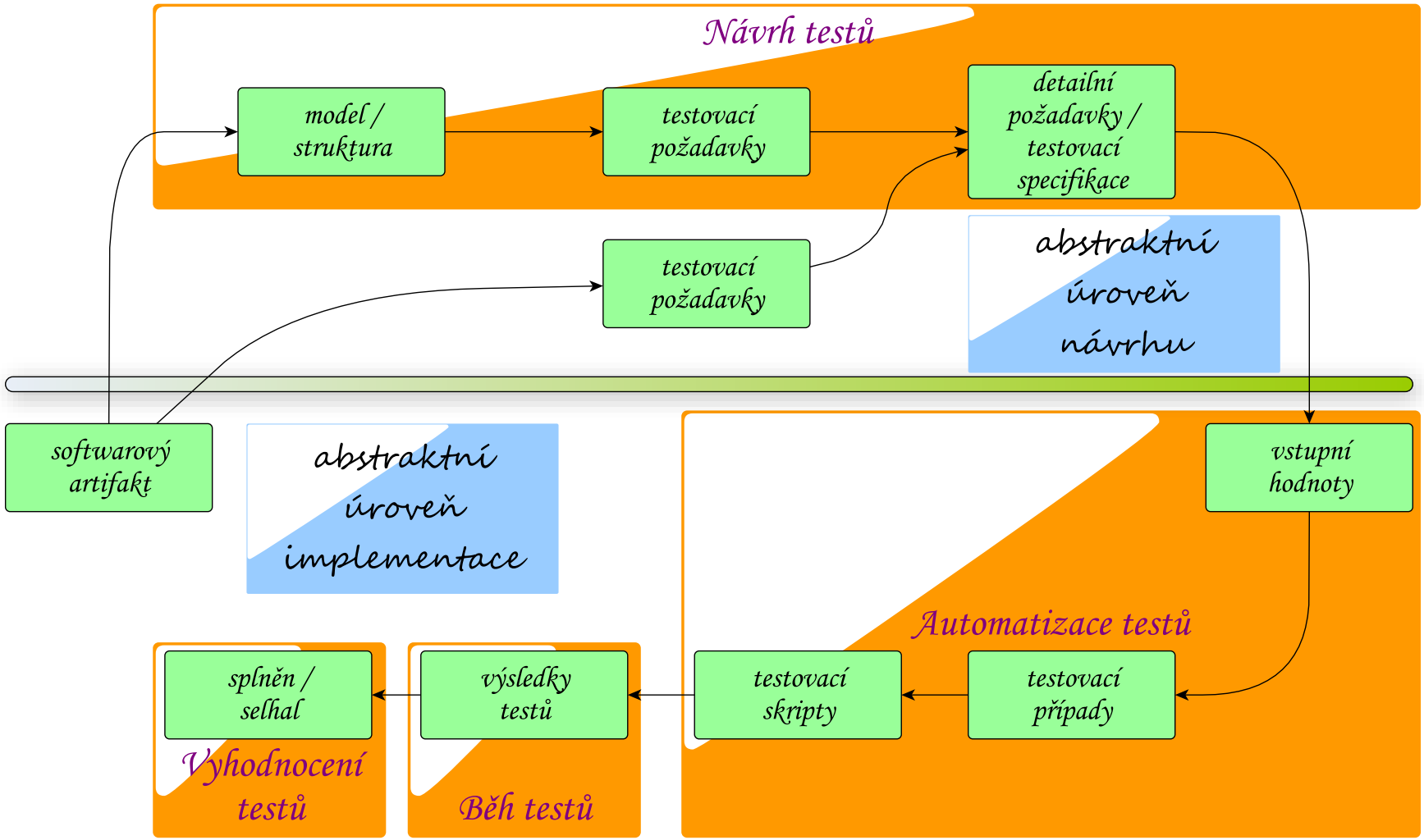
Návrh testů řízené modelem [AO10]



Návrh testů řízené modelem - kroky [AO10]



Návrh testů řízené modelem - aktivity [AO10]

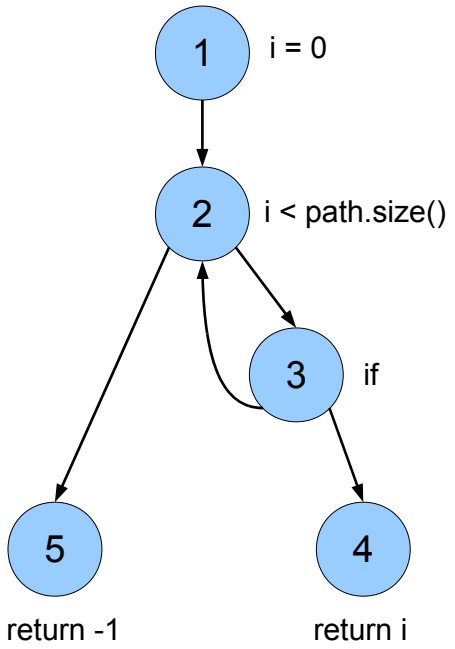


Návrh testů řízené modelem - příklad(1) [AO10]

Softwarový artefakt - Java funkce

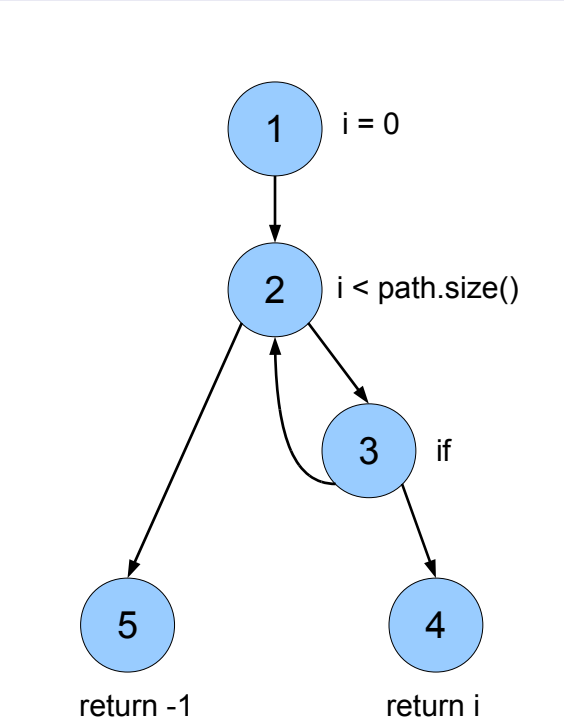
```
/* Return index of node n at
 * the first position it
 * appears,
 * -1 if it is not present.
*/
public int indexOf (Node n)
{
  for (int i=0;
       i < path.size();
       i++)
    if(path.get(i).equals(n))
      return i;
  return -1;
}
```

Řídicí tok



Návrh testů řízené modelem - příklad(2) [AO10]

Řídicí tok



Hrany

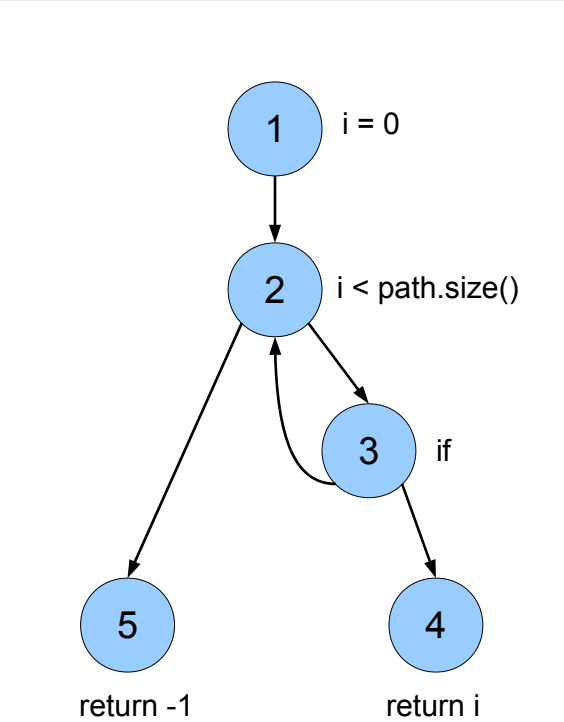
- 1 2
- 2 3
- 3 2
- 3 4
- 2 5

Počáteční uzel: 1

Koncové uzly: 4, 5

Návrh testů řízené modelem - příklad(3) [AO10]

Řídicí tok

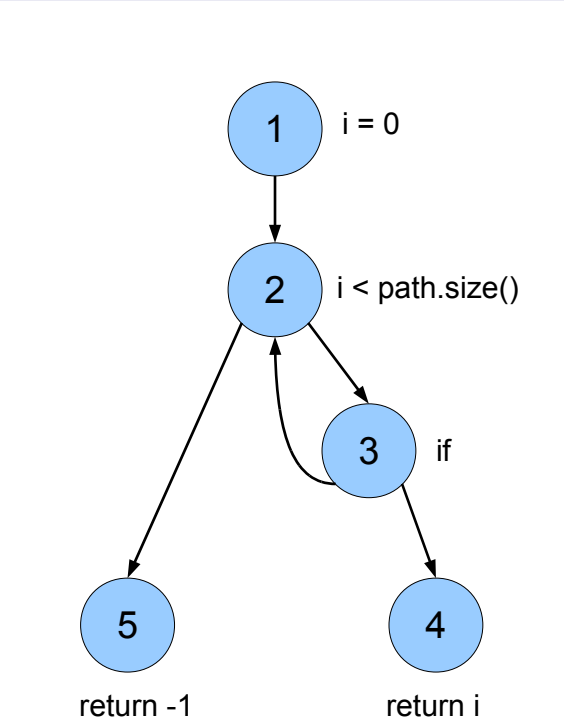


6 požadavků na pokrytí páru hran

1. $[1, 2, 3]$
2. $[1, 2, 5]$
3. $[2, 3, 4]$
4. $[2, 3, 2]$
5. $[3, 2, 3]$
6. $[3, 2, 5]$

Návrh testů řízené modelem - příklad(4) [AO10]

Řídicí tok



Testovací cesty

- $[1, 2, 5]$
- $[1, 2, 3, 2, 5]$
- $[1, 2, 3, 2, 3, 4]$

Model vady&selhání [AO08]

K tomu, aby selhání bylo zpozorováno, je potřeba splnit 3 podmínky:

1. **Dosažitelnost** Místa mající vady musí být dosažitelná.
2. **Infekce** Program se musí dostat do nesprávného stavu.
3. **Propagace** Infekční stav musí ovlivnit nějaký nesprávný výstup programu.

Pozorovatelnost a řiditelnost [AO08]

- **Pozorovatelnost softwaru** Jak snadné je pozorovat chování programu vzhledem k jeho výstupům, ke změnám prostředí, jiného hardwaru a softwarových komponent.
  - Software operující s hardwarem, databázemi, či vzdálenými soubory má nízkou porozovatelnost.
- **Řiditelnost softwaru** Jak je snadné ovládnout program potřebnými vstupy, tj. hodnotami, operacemi, či chováním.
  - Snadné je řídit software vstupy z klávesnice.
  - Hůře se zadávají vstupy ze senzorů nebo distribuovaného softwaru.
  - Abstrakce dat redukuje řiditelnost i pozorovatelnost.

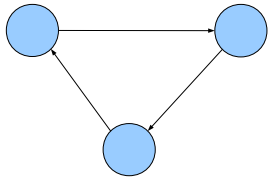
Změny v přístupu k testování [AO08]

- Tradiční testování zdůrazňovalo testovací metody dle **fází**.
  - testování jednotek, komponent, integrační, systémové.
- Moderní metody jsou založeny na **strukturách** a **kritériích**.
  - grafy, logické výrazy, syntax, prostor vstupů.
- **Návrh testů** je de-facto stejný ve všech fázích. Liší se pouze ve
  - způsobu vytvoření **modelů**.
  - výběru **hodnot** a automatizování běhu testů.

Moderní návrh testů [AO08]

- Základní postup.
  1. definuj model softwaru,
  2. nalezni způsob jeho pokrytí.
- **Testovací požadavky**:
  - specifické věci, které musí být splněny nebo pokryty během testování.
- **Testovací kritérium**:
  - množina pravidel a proces, které definují testovací požadavky
  - výběru **hodnot** a automatizování běhu testů.

Možnosti modelování softwaru [AO08]



1. Grafy
2. Logické výrazy
   - (not X or not Y) and A and B
3. Charakterizace vstupních domén:
   - $A : \{0, 1, > 1\}$
   - $B : \{600, 700, 800\}$
   - $C : \{\text{swe}, \text{cs}, \text{isa}, \text{infs}\}$
4. Syntaktické struktury

if $(x > y)$ **then**

$z = x - y;$

else

$z = 2 * x;$

end if

Testování cest [KFN93]

- **Cesta** je sekvence operací, které se provedou od začátku běhu programu do jeho ukončení, tzv. úplná cesta
- **Část cesty** je sekvence operací z jednoho místa programu do jiného místa.
- **Kritéria pokrytí** specifikují třídu cest, které by se měly provést v rámci testování. Typicky redukují množství testů na rozumnou proveditelnou úroveň.
- Testování provedené podle těchto kritérií se nazývá **testování cest**.
- **Critérium adekvátnosti testů** (EN test adequacy criterion) . . . je predikát, který vrací *true* nebo *false* for pár $\langle program, testsuite \rangle$.
  - Obvykle vyjádřeno ve formě pravidla:
    např. „všechny příkazy musí být pokryty“

Testování cest - příklad [KFN93]

```
IF (A>B and C==5)
  THEN do SOMETHING
SET D=5
```

```
(a) A>B  and C==5
       (SOMETHING is done, then D is set to 5)
(b) A>B  and C!=5
       (SOMETHING is not done,  D is set to 5)
(c) A<=B  and C==5
       (SOMETHING is not done, D is set to 5)
(d) A<=B  and C!=5
       (SOMETHING is not done, D is set to 5)
```

Kritéria pokrytí [KFN93]

Pokrytí řádek požaduje provedení každé řádky kódu alespoň jednou. Nejslabší kritérium.

- Tester může pokrýt všechny tři řádky kódu případem (a).

Pokrytí větví znamená, že podmínka každého větvení musí alespoň jednou být pravdivá a alespoň jednou nepravdivá. Toto pokrytí vyžaduje otestování všech řádek a všech větví.

- Tester může použít případ (a) a jakýkoliv další ze zbývajících tří.

Pokrytí podmínek zkontroluje všechny možné způsoby, za kterých daná podmínka (pokrytí i individuálních podmínek) je pravdivá či nepravdivá.

- Vyžaduje všechny čtyři případy.
- Požadováno americkou armádou a letectvím.

Úplné pokrytí cest vyžaduje provedení všech možných různých úplných cest.

Testování složených podmínek

- Mějme výraz $((((a \vee b) \wedge c) \vee d) \wedge e)$

- Potřebujeme 13 testů, abychom pokryli všechny možné kombinace.
- V obecném případě čelíme kombinatorické explozi.
- Nepraktické.

Test	a	b	c	d	e
(1)	True	-	True	-	True
(2)	False	True	True	-	True
(3)	True	-	False	True	True
(4)	False	True	False	True	True
(5)	False	False	-	True	True
(6)	True	-	True	-	False
(7)	False	True	True	-	False
(8)	True	-	False	True	False
(9)	False	True	False	True	False
(10)	False	False	-	True	False
(11)	True	-	False	False	-
(12)	False	True	False	False	-
(13)	False	False	-	False	-

Pokrytí rozhodnutí modifikovaných podmínek

- EN Modified Condition Decision Coverage (MCDC Testing)
- Testování *důležitých kombinací* podmínek, vyhnutí se kombinatorické explozi.
- Kombinace je *důležitá*, jestliže se pro každou základní podmínku prokáže její nezávislý vliv na výsledek rozhodnutí.
- Pro každou základní podmínku C potřebujeme dva testovací případy T_1 a T_2.
- Hodnoty všech vyhodnocených podmínek s vyjímkou C jsou ty samé.
- Složená podmínka se jako celek ohodnotí *True* pro T_1 a *False* pro T_2.
- Dobrá balance mezi hloubkou testování a počtem testů a proto se široce používá.
- Pro N proměnných stačí vytvořit $N + 1$ testovacích případů.

Testování MCDC

- Mějme výraz $((((a \vee b) \wedge c) \vee d) \wedge e)$
- Potřebujeme 6 testů, abychom pokryli MCDC kombinace.
- Podtržené hodnoty nezávisle ovlivňují hodnotu rozhodnutí.
- Ten samý test může pokrývat hodnoty několika základních podmínek.
  - Testovací případ (1) pokrývá hodnotu *True* pro základní podmínky *a*, *c* a *e*.

Test	a	b	c	d	e	Rozhodnutí
(1)	True	-	True	-	True	True
(2)	False	True	True	-	True	True
(3)	True	-	False	True	True	True
(6)	True	-	True	-	False	False
(11)	True	-	False	False	-	False
(13)	False	False	-	False	-	False

Grafy a relace [Bei95]

- Grafy jsou hlavním koncepčním nástrojem testování:
  - grafy toku řízení,
  - grafy toku dat,
  - stromy závislosti volání funkcí,
  - grafy konečných automatů,
  - grafy toku transakcí.
- **Relace:** vybraná asociace mezi objekty,
  - A, B . . . objekty,
  - $\leftrightarrow$, $\leftarrow$, $\rightarrow$ . . . relace,

$$A \rightarrow B$$

  - Příklad: akce A je následovaná akcí B.
- **Uzel:** objekty grafu reprezentované kroužky,
- **Jméno uzlu:** každý uzel má jednoznačnou identitu nebo jméno,
- **Atributy uzlu:** vlastnosti uzlu.
  - Příklady:
    - stav programu,
    - hodnota proměnné.

Hrany grafu [Bei95]

- **Hrana:** šipka nebo čára spojující dva uzly vyjadřující danou relaci mezi těmito uzly.
- **Jméno hrany:** každá hrana má jednoznačnou identitu nebo jméno.
  - pár jmen obou uzlů,
  - speciální jméno,
- **Atributy hran:** vlastnosti hran.
  - Příklady:
    - exekuční čas programu podél specifické cesty,
    - pravděpodobnost provedení dané cesty,
    - fakt o výběru určitého datového objektu.
- **Orientovaná hrana:** používaná pro asymetrické relace.
  - Příklad: A je následováno B.
  - Většina modelů testování používá orientované hrany.
- **Neorientovaná hrana:** hrana označující symetrickou relaci.
- **Paralelní hrany:** dvě či více hran mezi jedním párem uzlů.

Terminologie teorie grafu [Bei95]

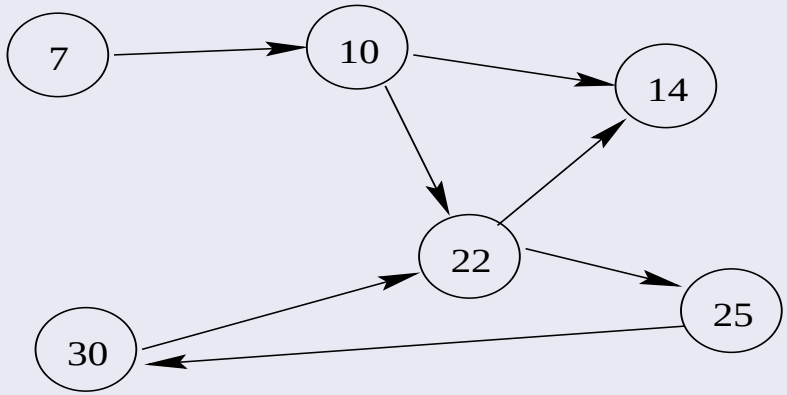
- **Graf:** je kolekce uzlů, jmen uzlů, atributů uzlů, hran, jmen hran, atributů hran a relací mezi uzly.
- **Orientovaný graf:** všechny hrany jsou orientované.
- **Neorientovaný graf:** všechny hrany jsou neorientované.
- **Vstupní hrana:** hrana, která míří do uzlu (hlava šipky).
- **Výstupní hrana:** hrana opouštějící uzel (ocas šipky).
- **Počáteční uzel:** uzel nemající vstupní hrany.
- **Koncový uzel:** uzel nemající výstupní hrany.
- **Uzel větvění:** uzel se dvěma a více výstupními hranami.
  - Příklad: CASE příkaz nebo IF-THEN-ELSE příkaz.
- **Cesta:** sekvence hran spojující dva uzly.
  - Příklad: Při testování chování se pracuje s cestami *modelem*, který popisuje chování softwaru. Takové cesty mohou nebo nemusí korespondovat cestám implementací programu.

Cesty grafu [Bei95]

- **Proveditelná cesta:** cesta, pro níž existují vstupní hodnoty takové, že běh programu bude sledovat danou cestu.
- **Neproveditelná cesta:** cesta, pro níž neexistují množina vstupních hodnot, která by umožňila postupovat programem podél této cesty.
- **Cesta start-stop:** cesta vedoucí od počátečního uzlu do koncového uzlu.
- **Segment cesty:** obvykle cesta, která není cestou start-stop.
- **Jméno cesty:**
  - jména uzlů podél cesty,
  - jména hran podél cesty,
- **Délka cesty:**
  - počet uzlů dané cesty,
  - počet hran dané cesty.
- **Smyčka:** jakákoliv cesta, která navštíví nějaký uzel alespoň dvakrát.
- **Cesta bez smyček:** cesta nemající smyčku.

Reprezentace grafů [Bei95]

Graf



Seznam

7:	10
10:	14, 22
14:	
22:	14, 25
25:	30
30:	22

Matice (tabulka)

	7	10	14	22	25	30
7	.	1	.	.	.	.
10	.	.	1	1	.	.
14	.	.	.	.	.	.
22	.	.	1	.	1	.
25	.	.	.	.	.	1
30	.	.	.	1	.	.

Obecné principy testování [Bei95]

Výstavba modelu (grafu)

1. objekty, o které se zajímáme (uzly).
2. relace, které by měly existovat mezi uzly.
3. které objekty mají vztah s jinými (hrany).
4. vlastnosti asociované s hranami: atributy hran.

Návrh testovacích případů podle modelu

1. definuj graf,
2. definuj relace,
3. navrhni testy pro **pokrytí uzlů** (testy potvrzující přítomnost uzlů).
4. navrhni testy pro **pokrytí hran** (testy potvrzující všechny požadované hrany a žádné jiné).
5. otestuj všechny atributy.
6. navrhni testy smyček.

Metoda hlavních cest [AO08]

1. **Jednoduchá cesta** . . . cesta z n_i do n_j, na které se žádný uzel neobjevuje více jak jedenkrát s vyjímkou, že počáteční a koncový uzel mohou být identické.
2. **Hlavní cesta** . . . cesta z n_i do n_j je hlavní, jestliže je to jednoduchá cesta a není žádnou vlastní podcestou jakékoliv jiné jednoduché cesty (tj. je maximální).
3. **Procházka** . . . Testovací cesta p prochází podcestu q, jestliže q je podcestou p.
4. **Procházka s vedlejšími výlety** . . . Testovací cesta p prochází podcestu q s vedlejšími výlety, jestliže každá *hrana* v q je také v p v tom samém pořadí.
    1. vedlejší výlet s vrací do stejného uzlu.
5. **Procházka s objížďkami** . . . Testovací cesta p prochází podcestu q s objížďkami, jestliže každý *uzel* v q je také v p v tom samém pořadí.
    1. nevrátí se do výchozího uzlu objížďky.

Konstrukce testových cest [AO08]

Procházení nejlepšího úsilí

1. Množina TR_{tour} je podmožina všech testovacích požadavků, pro které lze vytvořit procházku.
2. Množina $TR_{sidetrip}$ je podmožina všech testovacích požadavků, pro které lze vytvořit procházku s vedlejším výletem.
3. Množina T testových cest splňuje procházení nejlepšího úsilí, jestliže pro každou cestu $p \in TR_{tour}$ existuje cesta v T, která prochází p přímo, a pro každou cestu $p \in TR_{sidetrip}$ existuje cesta v T, která prochází p přímo nebo vedlejším výletem.

Nalezení hlavních cest [AO08]

Princip algoritmu

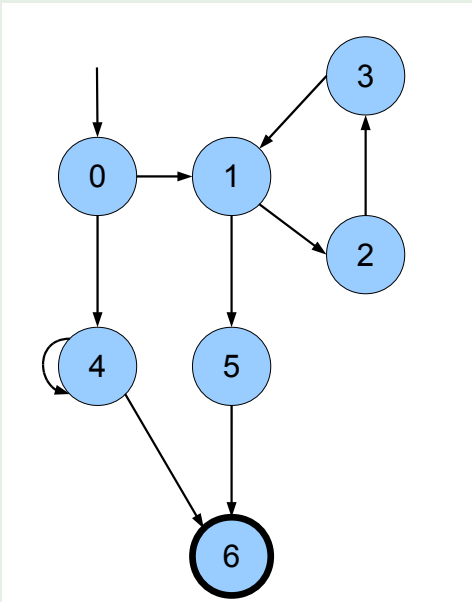
1. Nalezni cesty délky 0 (uzly).
2. Kombinuj cesty délky 0 do cest délky 1 (hrany).
3. Kombinuj cesty délky 1 do cest délky 2.
4. atd.

Označení

1. ! . . . cesta nemůže být prodloužena
2. * . . . cesta tvoří smyčku

Návrh testovacích případů podle modelu - příklad 1 [AO08]

Model



Délka 0

1. [0]
2. [1]
3. [2]
4. [3]
5. [4]
6. [5]
7. [6] !

Délka 1

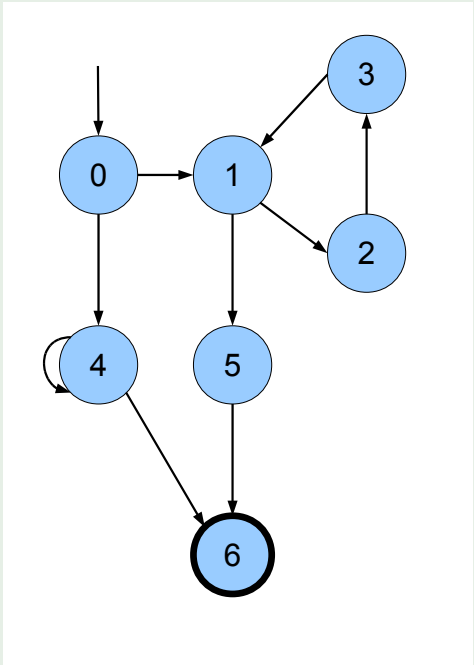
8. [0, 1]
9. [0, 4]
10. [1, 2]
11. [1, 5]
12. [2, 3]
13. [3, 1]
14. [4, 4] *
15. [4, 6] !
16. [5, 6] !

Délka 2

17. [0, 1, 2]
18. [0, 1, 5]
19. [0, 4, 6] !
20. [1, 2, 3]
21. [1, 5, 6] !
22. [2, 3, 1]
23. [3, 1, 2]
24. [3, 1, 5]

Návrh testovacích případů podle modelu - příklad 2 [AO08]

Model



Délka 2

17. $[0, 1, 2]$
18. $[0, 1, 5]$
19. $[0, 4, 6]$!
20. $[1, 2, 3]$
21. $[1, 5, 6]$!
22. $[2, 3, 1]$
23. $[3, 1, 2]$
24. $[3, 1, 5]$

Délka 3

25. $[0, 1, 2, 3]$!
26. $[0, 1, 5, 6]$!
27. $[1, 2, 3, 1]$ *
28. $[2, 3, 1, 2]$ *
29. $[2, 3, 1, 5]$
30. $[3, 1, 2, 3]$ *
31. $[3, 1, 5, 6]$!

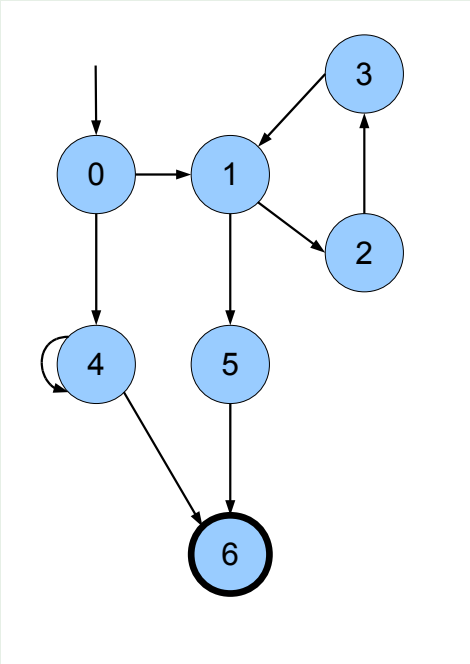
Délka 4

32. $[2, 3, 1, 5, 6]$!

Návrh testovacích případů podle modelu - příklad 3 [AO08]

Po eliminaci podcest jiných jednoduchých cest

Model



Hlavní cesty

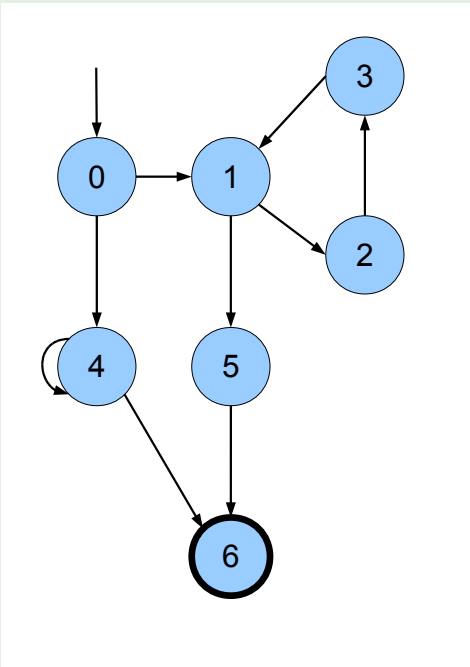
14. $[4, 4]$ *
19. $[0, 4, 6]$!
25. $[0, 1, 2, 3]$!
26. $[0, 1, 5, 6]$!
27. $[1, 2, 3, 1]$ *
28. $[2, 3, 1, 2]$ *
30. $[3, 1, 2, 3]$ *
32. $[2, 3, 1, 5, 6]$!

Návrh testovacích případů podle modelu - příklad 4 [AO08]

Konstrukce testovacích cest

- Existuje spousta algoritmů.
  1. Inženýrský cit - malé soubory dat
  2. Od nejdelších hlavních cest, jejich kombinování do testovacích cest.

Model



Hlavní cesty

14. $[4, 4]$ *
19. $[0, 4, 6]$!
25. $[0, 1, 2, 3]$!
26. $[0, 1, 5, 6]$!
27. $[1, 2, 3, 1]$ *
28. $[2, 3, 1, 2]$ *
30. $[3, 1, 2, 3]$ *
32. $[2, 3, 1, 5, 6]$!

Testovací cesty

1. $25, 27, 32 \rightarrow [0, 1, 2, 3, 1, 5, 6]$
2. $30, 28 \rightarrow [0, 1, 2, 3, 1, 2, 3, 1, 5, 6]$
3. $26 \rightarrow [0, 1, 5, 6]$
4. $19 \rightarrow [0, 4, 6]$
5. $14 \rightarrow [0, 4, 4, 6]$

Predikáty řízení toku [Bei95]

Příklady: výpočet daně z příjmu fyzických osob (smíchané USA a ČR formuláře)

- **logický predikát:** věta nebo výraz, který nabývá logické hodnoty TRUE nebo FALSE.
  - Příklad: Váš rodič Vás může prohlásit za vyživované dítě na přiznání k dani z příjmů fyzických osob.
- **výběrový predikát:** výraz, který nabývá více než dvou hodnot určených k výběru jedné z mnoha variant.
  - Příklad: Zaškrtni pouze políčko - (1) svobodný, (2) ženatý vyplňující společné přiznání, (3) ženatý vyplňující oddělené přiznání, (4) hlava domácnosti, (5) vdovec.
- **složený predikát:** logický výraz zahrnující více logických predikátů spojených spojkami **AND, OR,** nebo **NOT**.

Závislost řídicích predikátů [Bei95]

- **nezávislé predikáty:** dva nebo více predikátů podél cesty jsou nezávislé, pokud jejich pravdivostní hodnoty mohou být nastaveny nezávisle.
- **korelované predikáty:** dva nebo více predikátů podél cesty jsou korelované, pokud pravdivostní hodnota jednoho z nich omezuje pravdivostní hodnoty ostatních predikátů podél cesty.
  - Příklad: daný predikát se objeví dvakrát podél cesty.
- **komplementární segmenty cest:** dva segmenty, pokud ty samé predikáty nacházející se na těchto segmentech mají na jednom segmentu hodnotu TRUE a na druhém FALSE.

Graf toku řízení [Bei95]

- **objekty (uzly):** sekvence kroku zpracování taková, že pokud se zpracuje jeden krok sekvence, budou provedeny i všechny zbývající kroky této sekvence (předpoklad nepřítomnosti pochybení)
  - Příklad: Tax Form 1040 (USA)

    7: (enter) příjmy, výplaty, spropitné.
    8a: (enter) daněné úroky.
    9: (enter) příjmy z dividend
- **relace (hrany):** *je přímo následován čím*
  - Příklad: Tax Form 1040

    7: 8a
    8a: 9

Predikátový uzel [Bei95]

Definice

Uzel s dvěma či více výstupními hranami, každá hrana nesoucí atribut s hodnotou predikátu.

Příklad

Tax Form 1040: Jestliže Vás Váš rodič může prohlásit za vyživované dítě, pak zaškrtněte políčko 33b, jinak políčko 33b nezaškrtávejte.

33b	Váš rodič Vás může prohlásit za závislého (predikátový uzel)
33b'	zaškrtni políčko 33b (procesní uzel)
33b:	33b'(T), 34(F)
33b':	34

Další speciální uzle [Bei95]

- **výběrový uzel:** uzel s více než dvěma výstupními hranami každou ohodnocenou hodnotou výběru.
- **spojovací uzel:** uzel s dvěma a více vstupními hranami.
- Příklad (řádka 34, formulář 1040): Zadej větší z následujících hodnot:
  - buď srážku daně z Rozvahy A řádky 29
  - NEBO standardní srážku podle rodinného stavu.

  Avšak jestliže jste zaškrtl jedno z políček na řádce 33a nebo b, pokračujte v určení standardní srážky podle instrukcí. Jestliže jste zaškrtl políčko 33c, Vaše standardní srážka je nula.
  1. svobodný = $3,800,
  2. ženatý/vdaná vyplňující společně = $6,350,
  3. vdovec =$6,350,
  4. ženatý/vdaná vyplňující odděleně = $3,175,
  5. hlava domácnosti = $5,600.

Příklad výběrového a spojovacího uzlu [Bei95]

34	Standardní nebo položkovaná srážka?
34.1	Rozvaha A, řádka 29
34.2	zaškrtni 33a nebo 33b
34.4	zaškrtnuto 33c?
34.5	vyběr variantu (výběrový uzel)
34.8	srážka = \$6,350 (spojkový uzel)
34:	34.1(položky), 34.2(standard)
34.2:	34.3(T), 34.4(F)
34.4:	34.5(F), 34.6(T)
34.5:	34.7(svobodný), 34.8(společně, vdovec), . . .

Složené predikáty [Bei95]

- Složené predikáty jsou ošidné, neboť skrývají komplexitu.
- Je správné je testovat, protože v nich programátoři dělají často chyby.
- Vždy lze složený predikát expandovat, aby se odhlalila složitost.
- n pomocných predikátoru vede na 2^n větví.
- Musí se uvažovat všech 2^n případů.
- Případy nejsou redudantní, protože složený predikát bude pracovat jenom tehdy, pokud v implementace neobsahuje chybu.
- Příklad: $A\&B$ OR C

A:	B.1(T), B.2(F)
B.1:	C.1(T), C.2(F)
B.2:	C.3(T), C.4(F)
C.1:	TRUE(T,F)
C.2:	TRUE(T), FALSE(F)
C.3:	TRUE(T), FALSE(F)
C.4:	TRUE(T), FALSE(F)

Princip [AO08]

Cíle a terminologie

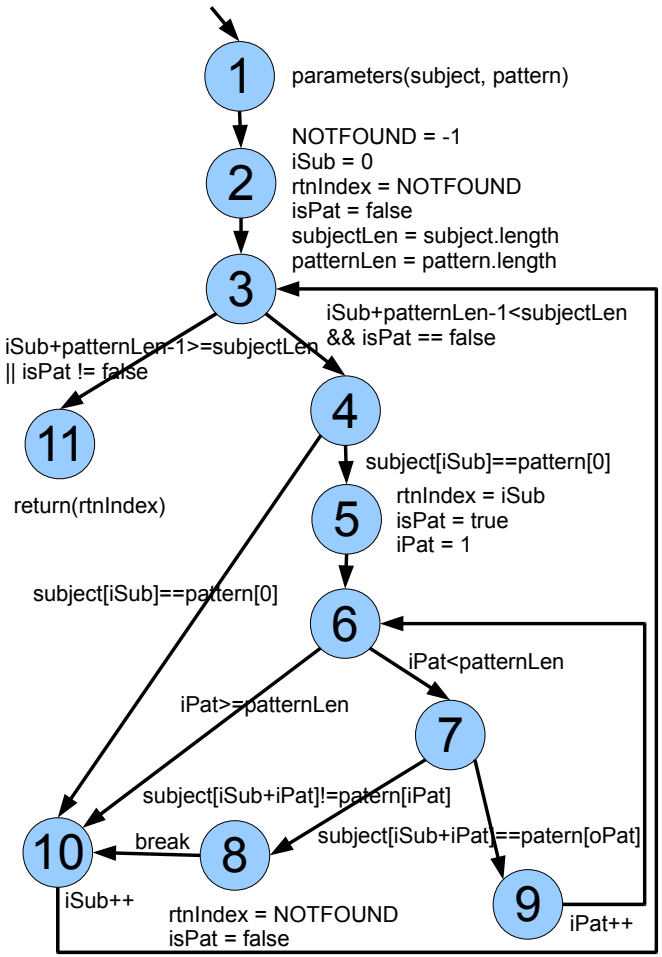
- **Tok datových hodnot:** testy zajišťující, že hodnoty vzniklé na jednom místě jsou použity správně na jiných místech.
- **Definice** *def*: místo, kde je hodnota proměnné uložena do paměti.
- **Užití** *use*: místo, kde se přistupuje k hodnotě proměnné.
- **DU páry** *def* – *use*: asociace určující přenosy hodnot.

Formalizace

- V . . . množina proměnných asociované se softwarovým artefaktem.
- $def(n)$ ($def(e)$) . . . podmnožina množiny proměnných V, které jsou definovány uzlem n (hranou e).
- $use(n)$ ($use(e)$) . . . podmnožina množiny proměnných V, které jsou použity v uzlu n (na hraně e).
  - V programech asociace s uzly.
  - V stavových diagramech i asociace s hranami.

Příklad 1 [AO08]

Model



Množiny $def(n)$ a $use(n)$

$$
\begin{aligned}
def(1) &= \{\text{subject}, \text{pattern}\} \\
def(2) &= \{\text{NOTFOUND}, \text{iSub}, \text{rtnIndex}, \\
&\quad \text{isPat}, \text{subjectLen}, \text{patternLen}\} \\
use(2) &= \{\text{subject}, \text{pattern}\} \\
use(3,11) &= use(3,4) \\
&= \{\text{isPat}, \text{iSub}, \text{subjectLen}, \text{patternLen}\}
\end{aligned}
$$

$$
\begin{aligned}
use(4,10) &= use(4,5) = \{\text{iSub}, \text{subject}, \text{pattern}\} \\
def(5) &= \{\text{rtnIndex}, \text{iPat}, \text{isPat}\} \\
use(5) &= \{\text{iSub}\} \\
use(6,10) &= use(6,7) = \{\text{iPat}, \text{patternLen}\} \\
use(7,8) &= use(7,9) = \{\text{iSub}, \text{subject}, \text{pattern}, \text{iPat}\} \\
def(8) &= \{\text{rtnIndex}, \text{isPat}\} \\
use(8) &= \{\text{NOTFOUND}\} \\
def(9) &= \{\text{iPat}\} \\
use(9) &= \{\text{iPat}\} \\
def(10) &= \{\text{iSub}\} \\
use(10) &= \{\text{iSub}\} \\
use(11) &= \{\text{rtnIndex}\}
\end{aligned}
$$

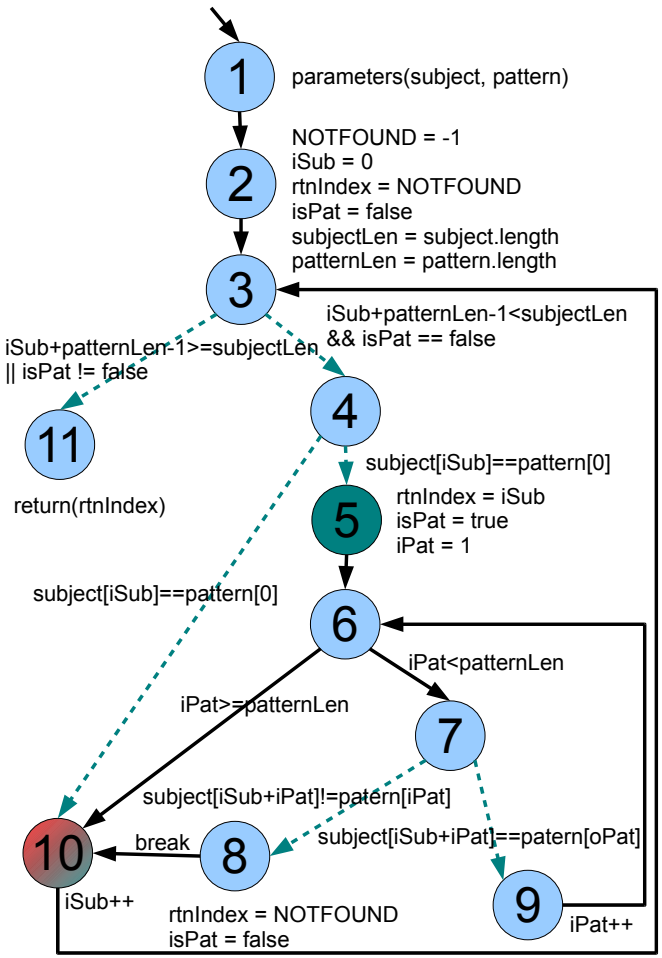
DU testy [AO08]

DU cesty

- Cesta z l_i do l_j je *def*-**čistá** vzhledem k proměnné v, jestliže pro každý uzel n_k a každou hranu e_k na této cestě, $k \neq i$ a $k \neq j$, $v \notin def(n_k)$ a $v \notin def(e_k)$.
  - *def* proměnné v v l_i dosahuje použití v l_j.
- *du*-**cesta:** vzhledem k proměnné v je jednoduchá cesta, která je *def*-čistá vzhledem k v z uzlu n_i, $v \in def(n_i)$, do uzlu n_j, $v \in use(n_j)$.
- $du(n_i, v)$**:** je množina všech *du*-cest vzhledem k proměnné v, která začíná v uzlu n_i.
- $du(n_i, n_j, v)$**:** je množina všech *du*-cest vzhledem k proměnné v, která začíná v uzlu n_i a končí v uzlu n_j.
  - $du(n_i, v) = \cup_{n_j} du(n_i, n_j, v)$

Příklad 2 [AO08]

Model



DU cesty

- uzly: $2, 5, 10$
- hrany: $(3,4), (3,11), (4,5), (4,10), (7,8), (7,9)$
- *def*-cesty a *def*-páry

$du(10, \text{iSub}) = \{[10,3,4], [10,3,4,5], [10,3,4,5,6,7,8],$
$= [10,3,4,5,6,7,9], [10,3,4,5,6,10],$
$= [10,3,4,5,6,7,8,10], [10,3,4,10],$
$= [10,3,11]\}$

$du(10,4,\text{iSub}) = \{[10,3,4]\}$
$du(10,5,\text{iSub}) = \{[10,3,4,5]\}$
$du(10,8,\text{iSub}) = \{[10,,3,4,5,6,7,8]\}$
$du(10,9,\text{iSub}) = \{[10,3,4,5,6,7,9]\}$
$du(10,10,\text{iSub}) = \{[10,3,4,5,6,10], [10,3,4,10],$
$= [10,3,4,5,6,7,8,10]\}$
$du(10,11,\text{iSub}) = \{[10,3,11]\}$

DU kritéria [AO08]

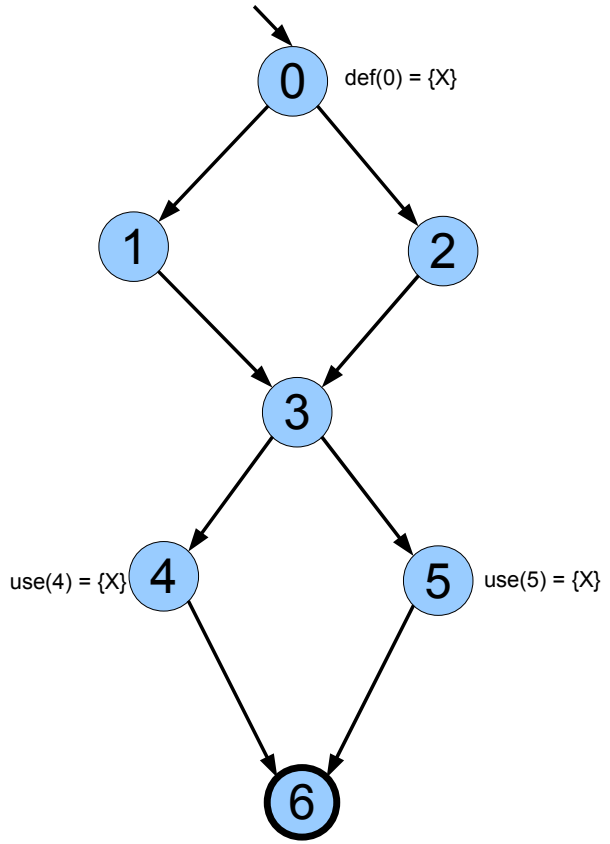
- Vedlejší výlety se vyžadují *def*-čisté vzhledem k dané proměnné.

Hlavní kritéria pokrytí pro datové toky

- **Kritérium všech definic (All-Defs Coverage, ADC)** TR obsahuje nejméně jednu cestu $d \in S$ pro alespoň jednu du-cestu množiny $S = du(n, v)$.
- **Kritérium všech užití (All-Uses Coverage, AUC)** TR obsahuje nejméně jednu cestu $d \in S$ pro každý du-pár množiny $S = du(n_i, n_j, v)$.
- **Kritérium všech du-cest (All-du-Paths Coverage, ADUPC)** TR obsahuje všechny cesty $d \in S$ pro každý du-pár množiny $S = du(n_i, n_j, v)$.

Rozdíly mezi kritérii [AO08]

Model



DU cesty

- Všechny definice:
  1. $\{[0, 1, 3, 4]\}$
- Všechna užití:
  1. $[0, 1, 3, 4]$,
  2. $[0, 1, 3, 5]$
- Všechny *du*-cesty:
  1. $[0, 1, 3, 4]$,
  2. $[0, 1, 3, 5]$,
  3. $[0, 2, 3, 4]$,
  4. $[0, 2, 3, 5]$

Literatura I

Paul Ammann and Jeff Offutt.

Introduction to Software Testing.
Cambridge University Press, Cambridge, UK, first edition, February 2008.
ISBN 0-52188-038-1.

Paul Ammann and Jeff Offutt.

Introduction to software testing, powerpoint slides, August 2010.

Boris Beizer.

Black-Box Testing, Techniques for Functional Testing of Software and Systems.
John Wiley & Sons, Inc., New York, 1995.

Cem Kaner, Jack Falk, and Hung Quoc Nguyen.

Testing Computer Software.
International Thomson Computer Press, second edition, 1993.