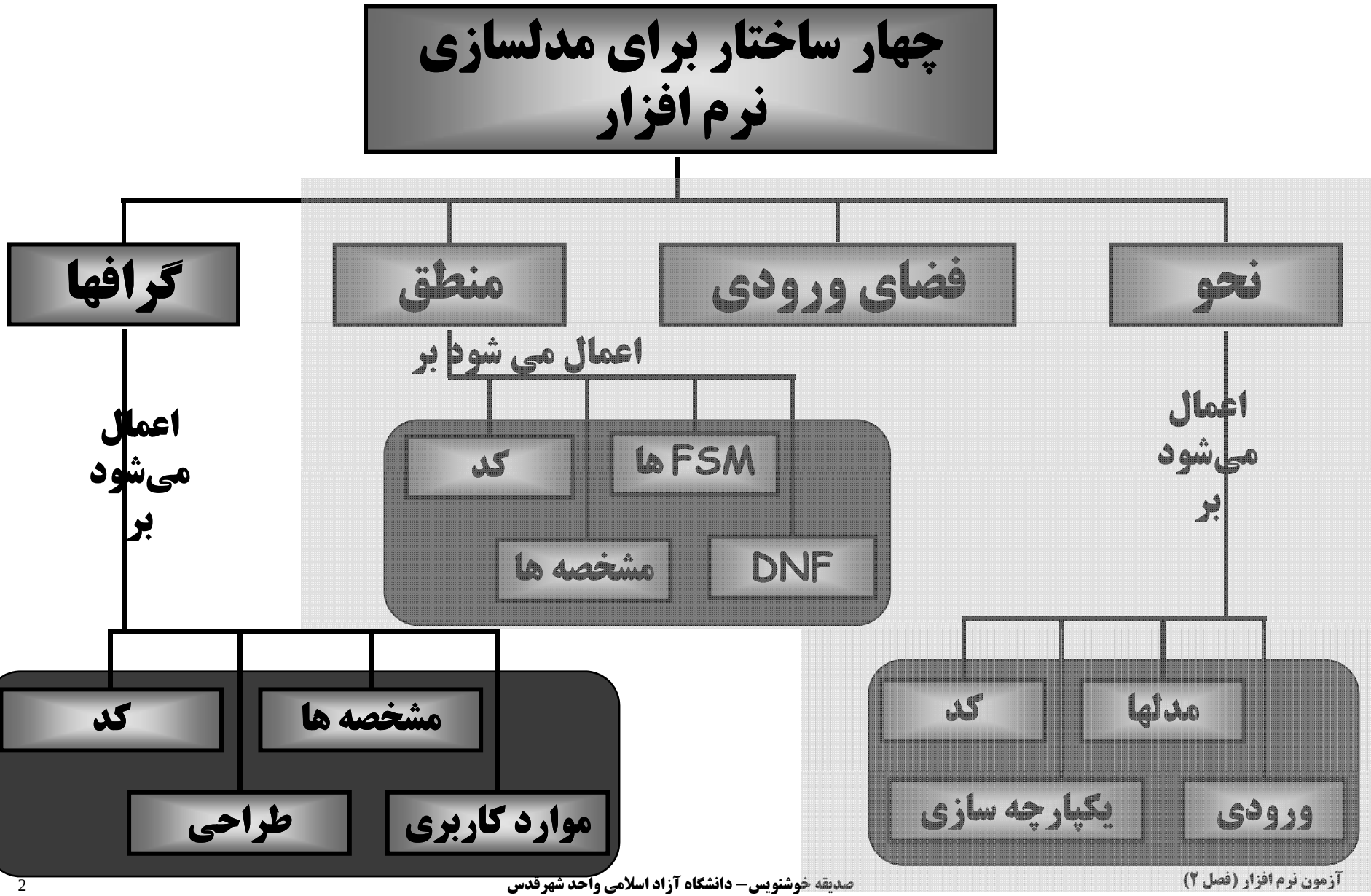
آزمون نرم افزار – بخشهای۲–۱ و ۲–۲

مروری بر معیارهای پوشش گراف

صدیقه خوشنویس
دانشگاه آزاد اسلامی – واحد شهرقدس

فصل ۲ – پوشش گراف



پوشش گراف (بخش ۲–۱)

- گراف رایجترین ساختار مورد استفاده برای تست است.

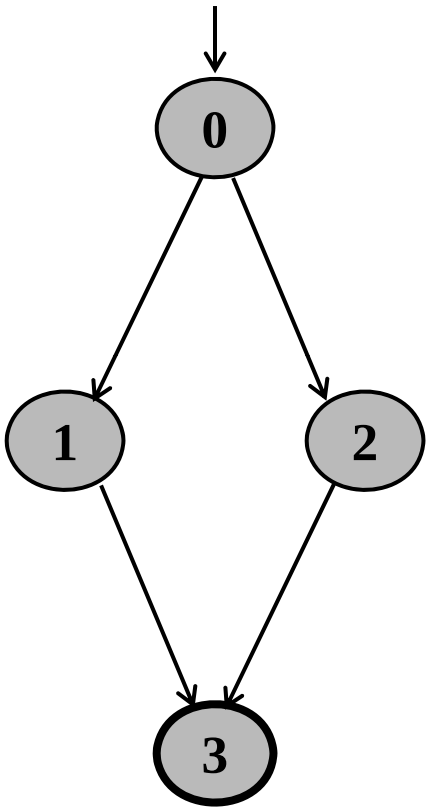
- گراف می تواند بر اساس منابع مختلفی ایجاد شود؛ پس انواع مختلفی از گراف داریم:
  - گراف جریان کنترل (CFG)
  - ساختارهای طراحی
  - ماشین های حالت متناهی (FSM) و نمودارهای حالت (StateChart)
  - موارد کاربری

- معمولاً هدف آن است که گراف ها را به نحوی «پوشش دهیم».

تعریف گراف

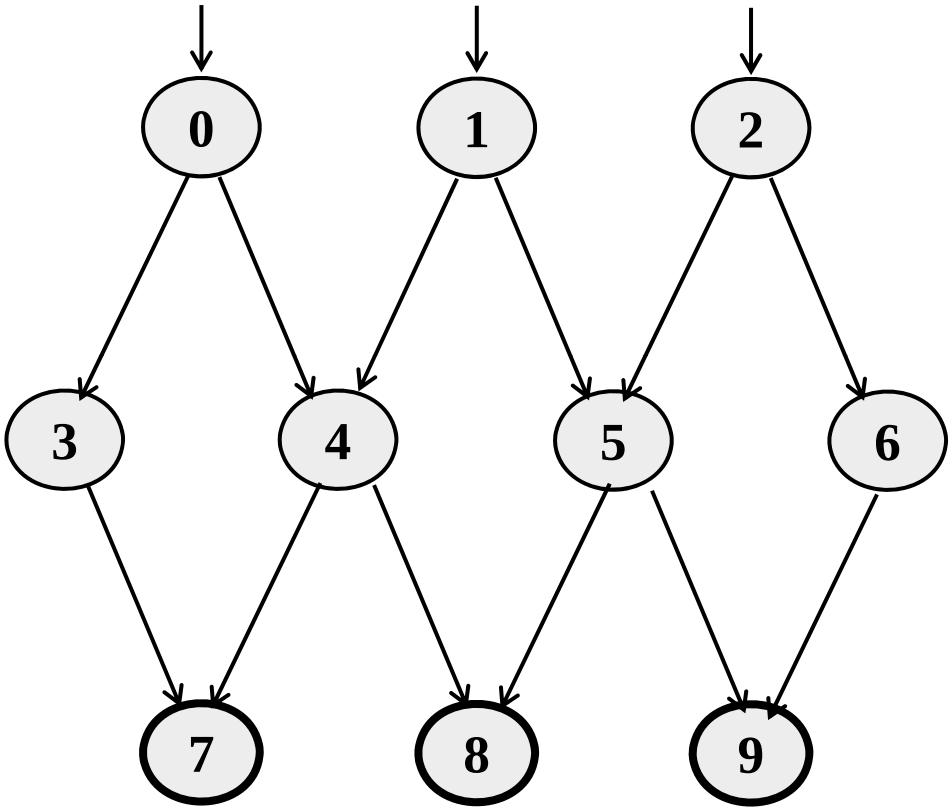
- مجموعه N از تعدادی گره (N نباید تهی باشد)
- مجموعه N_0 از گره های آغازین (N_0 نباید تهی باشد)
- مجموعه N_f از گره های پایانی (N_f نباید تهی باشد)
- مجموعه E از یالها، که هر یال از یک گره به گره دیگر است
  - اگر یال از گره n_i به n_j را به صورت (n_i,n_j) نشان دهیم، به گره **i**، قبلی، و به گره **j**، بعدی می گوییم.

سه مثال از گراف



$N_0 = \{ 0 \}$

$N_f = \{ 3 \}$



$N_0 = \{ 0, 1, 2 \}$

$N_f = \{ 7, 8, 9 \}$

$N_0 = \{ \}$

$N_f = \{ 3 \}$

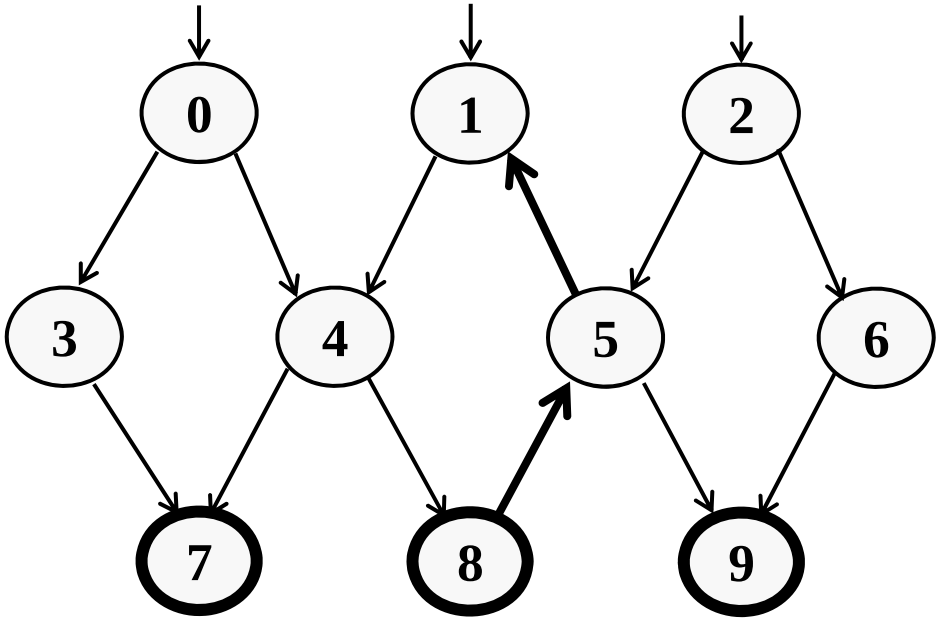
مسیر در گراف

- **مسیر: یک دنباله از گره ها – $[n_1, n_2, \ldots, n_M]$**
  - هر جفت گره متوالی یک یال است
- **طول: تعداد یال ها**
  - یک گره منفرد، مسیری است با طول صفر
- **زیرمسیر: یک زیردنباله از گره ها در مسیر p، یک زیرمسیر از p است**
- **دسترسی از گره n یا Reach(n) : زیرگرافی که می توان از گره n به آنها رسید.**

Reach (0) =
{ 0, 3, 4, 7, 8, 5, 1, 9 }

Reach ({0, 2}) = G

Reach([2,6]) =
{2, 6, 9}



چند مسیر:

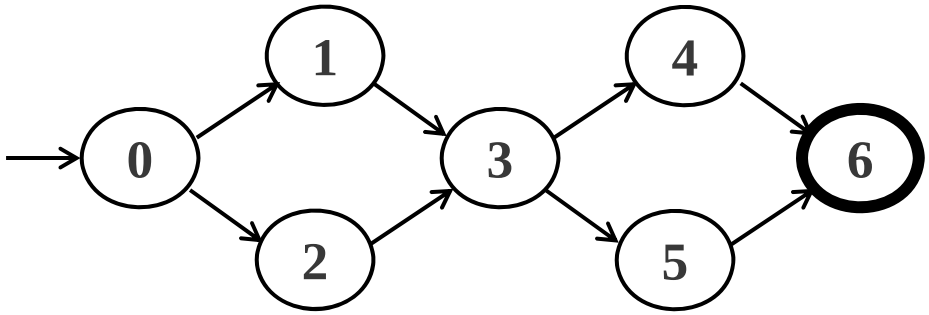
[0, 3, 7]

[1, 4, 8, 5, 1]

[2, 6, 9]

مسیرهای آزمون و گراف SESE

- مسیر آزمون (مسیر تست): مسیری که از یک گره آغازین شروع می شود و به یک گره پایانی ختم می شود.
- مسیرهای آزمون نشان دهنده مسیر اجرای موارد آزمون در برنامه هستند
  - برخی از مسیرهای آزمون می توانند با تست های زیادی اجرا شوند (ممکن = **feasible**)
  - برخی از مسیرهای آزمون با هیچ تستی نمی توانند اجرا شوند (ناممکن = **infeasible**)



برای گراف دو لوزی (double-diamond)
چهار مسیر آزمون داریم:

[0, 1, 3, 4, 6]
[0, 1, 3, 5, 6]
[0, 2, 3, 4, 6]
[0, 2, 3, 5, 6]

ملاقات و پیمایش

- **ملاقات (visit)**: مسیر آزمون p، گره n را ملاقات می کند اگر n درون p باشد.
- مسیر آزمون p، یال e را ملاقات می کند اگر e درون p باشد.

- **پیمایش(tour)**: مسیر آزمون p، زیرمسیر q را پیمایش می کند اگر q زیرمسیر p باشد.

مسیر [0, 1, 3, 4, 6]	
ملاقات می کند گره های:	0, 1, 3, 4, 6
ملاقات می کند یال های:	(0, 1), (1, 3), (3, 4), (4, 6)
پیمایش می کند زیرمسیرهای:	(0, 1, 3), (1, 3, 4), (3, 4, 6), (0, 1, 3, 4), (1, 3, 4, 6)

آزمون ها و مسیرهای آزمون

- **t : یک آزمون**
- **T : مجموعه ای از آزمونها**

- **Path(t) : مسیر آزمونی که توسط آزمون t اجرا می شود.**
- **Path(T) : مجموعه مسیرهای آزمونی که توسط مجموعه آزمون T اجرا می شود.**

آزمایش و پوشش گراف (بخش ۲-۲)

- ما از گراف به صورت زیر در آزمایش نرم افزار استفاده می کنیم:
  - (۱) ایجاد مدلی از نرم افزار به صورت گراف
  - (۲) تعریف نیازمندی آزمون در قالب نیاز به ملاقات یا پیمایش مجموعه های معینی از گره ها، یال ها ، یا زیرمسیرها

- نیازمندی آزمون (Test Requirement)(TR): توصیف این که مسیرهای آزمون باید چه ویژگیهایی داشته باشند.
- معیار آزمون (Test Criterion): قوانینی برای تعریف TRها

آزمایش و پوشش گراف (بخش ۲–۲)

- **برآورده کردن (Satisfaction):**
- **اگر مجموعه نیازمندیهای آزمون = TR**
- **معیار آزمون = C**
- **مجموعه آزمون = T باشد :**
- **مجموعه آزمون T، معیار C را روی گراف G برآورده می کند اگر و فقط اگر برای هر نیازمندی tr در TR، یک مسیر آزمون p در path(T) وجود داشته باشد که نیازمندی آزمون tr را برآورده کند.**

- ***به عبارت ساده تر: این که هر نیازمندی آزمون (tr) را (که بر اساس معیار C تعیین می شوند) بتوان با یک مسیر آزمون (p) برآورده کرد.***
  - ***(بعداً با مثال نشان می دهیم)***

- **معیارهای پوشش ساختاری: تنها در قالب گره ها و یالهای یک گراف تعریف می شود.**
- **معیارهای پوشش جریان داده: باید گراف را با یک سری متغیر حاشیه نویسی کنیم.**

معیارهای پوشش گره و پوشش یال

- پوشش گره (Node Coverage) و پوشش یال (Edge Coverage)
  - ساده ترین معیارهای آزمون هستند
  - نیازمندی آنها آن است که همه گره ها و همه یالها اجرا شوند

پوشش گره (NC) : TR شامل همه گره های قابل دسترس در G است.

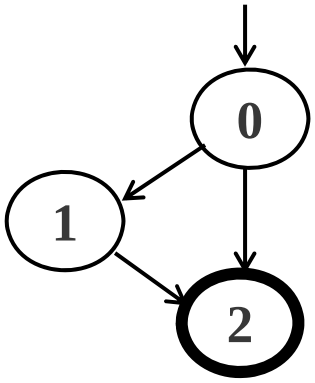
- پوشش یال کمی از پوشش گره قوی تر است:

پوشش یال (EC) : TR شامل همه یال های قابل دسترس در G با طول حداکثر ۱ (شامل طول ۱) است.

- «طول حداکثر ۱» باعث می شود اگر گرافی فقط یک گره (بدون یال) داشت، این معیار آن را هم پوشش دهد.

معیارهای پوشش گره و پوشش یال...

- معیارهای NC و EC فقط وقتی با هم فرق دارند که در گراف دو انشعاب (یکی: یال، و دیگری: یک زیرمسیر دیگر) به یک گره وجود داشته باشد (یعنی یک if-else)



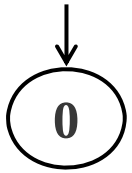
پوشش گره (NC):

نیازمندی آزمون (TR) = { 0, 1, 2 }
مسیر آزمون = [0, 1, 2]

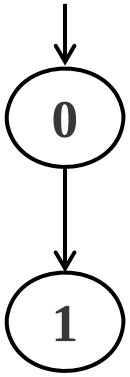
پوشش یال (EC):

نیازمندی آزمون (TR) = { (0,1), (0, 2), (1, 2) }
مسیرهای آزمون = [0, 1, 2]
[0, 2]

مسیرهای با طول ۰ و ۱



- گرافی که تنها یک گره داشته باشد هیچ یالی ندارد.
- اگر پوشش یال، مسیرهای با طول صفر را پوشش ندهد، گراف فوق را نمی‌تواند پوشش دهد.
- اما پوشش یال قوی تر از پوشش گره است و باید آن را «در بر داشته باشد».
- به همین دلیل می‌گوییم در پوشش یال، مسیرهای با طول «حداکثر» ۱ (یعنی با طول ۰ و ۱) باید تست شوند.



- مشابه همین حالت را برای پوشش زوج یال داریم (اسلاید بعد)؛
- که باید مسیرهای با طول حداکثر ۲ (یعنی ۰، ۱ و ۲) را پوشش دهد؛
- تا بتواند گراف تک گره ای یا گراف با دو گره و یک یال را پوشش دهد.

پوشش چندین یال

- پوشش زوج یال (Edge Pair Coverage) نیازمند پوشش یک جفت یال متوالی یا زیرمسیرهایی با طول حداکثر ۲ است (علت طول حداکثر ۲ در اسلاید قبل بیان شد).

پوشش زوج یال (EPC) : TR شامل همه مسیرهای قابل دسترس در G با طول حداکثر ۲ (شامل طول ۲) است.

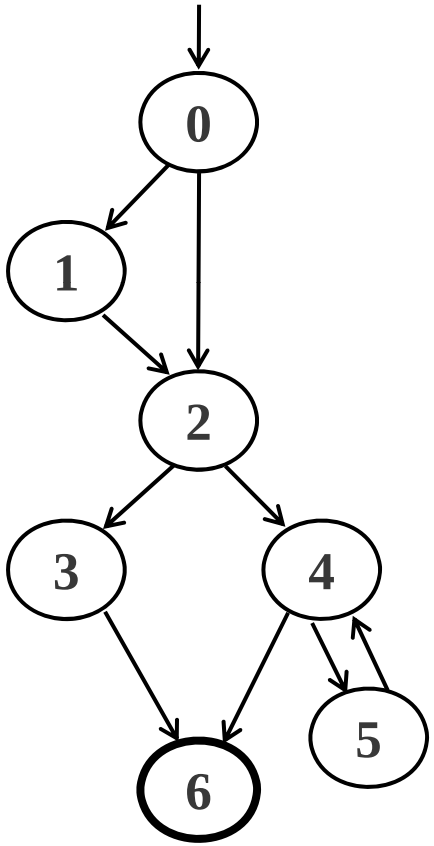
- بسط منطقی این پوشش ما را به پوشش همه مسیرها (با هر طولی) (Complete Path Coverage) می رساند:

پوشش همه مسیرها (CPC) : TR شامل همه مسیرهای G است.

- ولی پوشش CPC در صورتی که گراف دارای حلقه باشد غیرممکن است؛ در این صورت می توانیم مشکل را به این صورت حل کنیم که تصمیم درباره این که کدام مسیرها پوشش داده شوند را به آزمونگر بسپاریم: پوشش مسیر تعیین شده (Specified Path Coverage):

پوشش مسیر تعیین شده (SPC) : TR شامل یک مجموعه S از مسیرهای آزمون است که توسط آزمونگر تعیین می شود.

مثال از پوشش ساختاری



پوشش گره (NC)

TR = { 0, 1, 2, 3, 4, 5, 6 }
Test Paths: [0, 1, 2, 3, 6] [0, 1, 2, 4, 5, 4, 6]

پوشش یال (EC)

TR = { (0,1), (0,2), (1,2), (2,3), (2,4), (3,6), (4,5), (4,6), (5,4) }
Test Paths: [0, 1, 2, 3, 6] [0, 2, 4, 5, 4, 6]

پوشش زوج یال (EPC)

TR = { [0,1,2], [0,2,3], [0,2,4], [1,2,3], [1,2,4], [2,3,6],
[2,4,5], [2,4,6], [4,5,4], [5,4,5], [5,4,6] }
Test Paths: [0, 1, 2, 3, 6] [0, 1, 2, 4, 6] [0, 2, 3, 6]
[0, 2, 4, 5, 4, 5, 4, 6]

پوشش همه مسیرها (CPC)

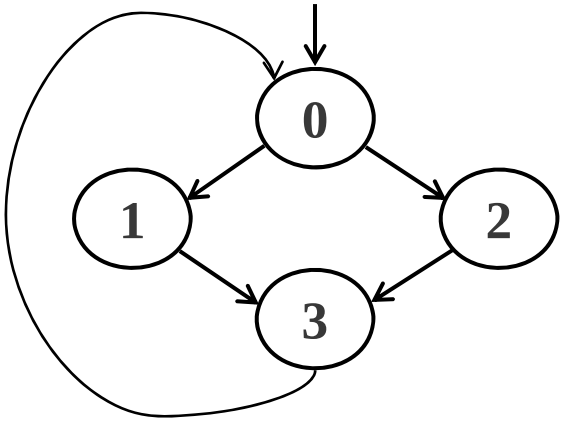
Test Paths: [0, 1, 2, 3, 6] [0, 1, 2, 4, 6] [0, 1, 2, 4, 5, 4, 6]
[0, 1, 2, 4, 5, 4, 5, 4, 6] [0, 1, 2, 4, 5, 4, 5, 4, 5, 4, 6] …

حلقه ها در گراف

- اگر گرافی دارای حلقه باشد، تعداد مسیرهای آن بینهایت خواهد بود.
- پس CPC برای آن ناممکن است.
- SPC نیز رضایت بخش نیست زیرا وابسته به آزمونگر است
- راه حل: معرفی پوشش «مسیر اصلی» (Prime Path Coverage)

مسیرهای ساده و مسیرهای اصلی

- **مسیر ساده (Simple Path): می گوییم یک مسیر، ساده است؛ اگر هیچ گره ای در آن بیش از یک بار دیده نشود.**
  - **استثناء: گره اول و آخر مسیر ساده می تواند یکسان باشد.**
  - **مسیر ساده حلقه داخلی ندارد.**
- **مسیر اصلی (Prime Path): مسیر ساده ای است که زیرمسیر هیچ مسیر ساده دیگری نیست.**



Simple Paths : [0, 1, 3, 0], [0, 2, 3, 0], [1, 3, 0, 1], [2, 3, 0, 2], [3, 0, 1, 3], [3, 0, 2, 3], [1, 3, 0, 2], [2, 3, 0, 1], [0, 1, 3], [0, 2, 3], [1, 3, 0], [2, 3, 0], [3, 0, 1], [3, 0, 2], [0, 1], [0, 2], [1, 3], [2, 3], [3, 0], [0], [1], [2], [3]

Prime Paths : [0, 1, 3, 0], [0, 2, 3, 0], [1, 3, 0, 1], [2, 3, 0, 2], [3, 0, 1, 3], [3, 0, 2, 3], [1, 3, 0, 2], [2, 3, 0, 1]

پوشش مسیر اصلی

- پوشش مسیر اصلی (Prime Path Coverage) یک معیار متناهی است که موجب می شود حلقه ها را هم اجرا کنیم و هم از آنها صرفنظر کنیم.

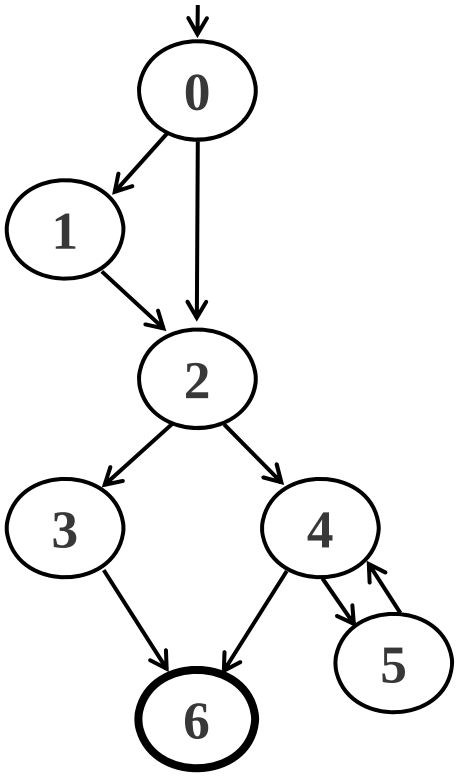
پوشش مسیر اصلی (PPC) : TR شامل همه مسیرهای اصلی گراف G است.

- همه مسیرهای با طول ۰، ۱، ۲ و ... را پیمایش می کند.
- به عبارت دیگر پوشش های گره، یال، زوج یال و .. را در بر دارد.

- نکته: مسیر اصلی و مسیر ساده درون خود حلقه داخلی ندارند. ولی مسیر آزمون می‌تواند درون خود حلقه داخلی داشته باشد.

مثال از پوشش مسیر اصلی

- گراف زیر دارای ۳۸ مسیر ساده است.
- ولی تنها ۹ مسیر اصلی دارد.



Prime Paths

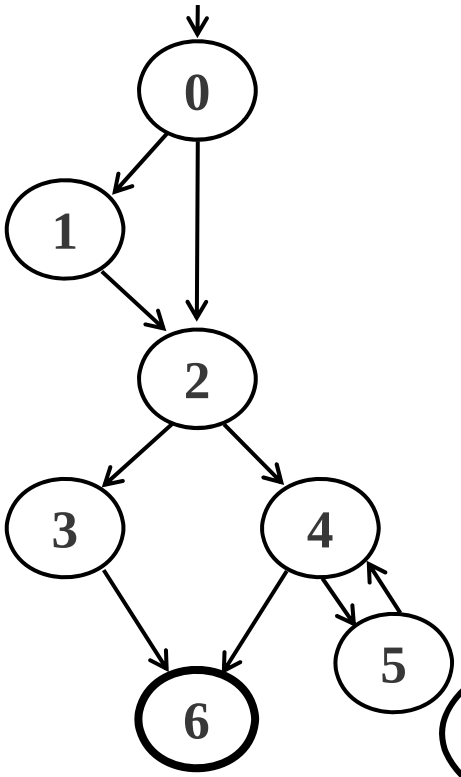
[0, 1, 2, 3, 6]
[0, 1, 2, 4, 5]
[0, 1, 2, 4, 6]
[0, 2, 3, 6]
[0, 2, 4, 5]
[0, 2, 4, 6]
[5, 4, 6]
[4, 5, 4]
[5, 4, 5]

صفر بار اجرای حلقه

یک بار اجرای حلقه

بیش از یک بار اجرای حلقه

مثال از نحوه به دست آوردن مسیرهای اصلی



مسیرهای ساده

طول ۰
[0]
[1]
[2]
[3]
[4]
[5]
[6] !

طول ۱
[0, 1]
[0, 2]
[1, 2]
[2, 3]
[2, 4]
[3, 6] !
[4, 6] !
[4, 5]
[5, 4]

طول ۲
[0, 1, 2]
[0, 2, 3]
[0, 2, 4]
[1, 2, 3]
[1, 2, 4]
[2, 3, 6] !
[2, 4, 6] !
[2, 4, 5] !
[4, 5, 4] *
[5, 4, 6] !
[5, 4, 5] *

طول ۳
[0, 1, 2, 3]
[0, 1, 2, 4]
[0, 2, 3, 6] !
[0, 2, 4, 6] !
[0, 2, 4, 5] !
[1, 2, 3, 6] !
[1, 2, 4, 5] !
[1, 2, 4, 6] !

طول ۴
[0, 1, 2, 3, 6] !
[0, 1, 2, 4, 6] !
[0, 1, 2, 4, 5] !

علامت «!» به معنای انتهای مسیر (برخورد به گره پایانی یا بروز حلقه داخلی در صورت ادامه) است

علامت «*» به معنای ایجاد حلقه در مسیر است (و نمی‌توان دیگر آن را ادامه داد)

مسیرهای اصلی
(زیرمسیر هیچ مسیر ساده دیگری نیستند)

مثال دوم از پوشش مسیر اصلی

مسیرهای ساده

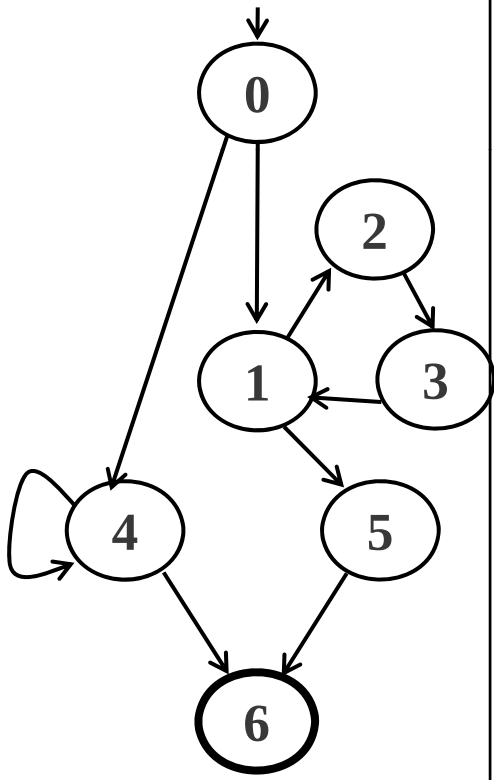
طول ۰
(#:۷)
[0]
[1]
[2]
[3]
[4]
[5]
[6] !

طول ۱
(#:۹)
[0, 1]
[0, 4]
[1, 2]
[1, 5]
[2, 3]
[3, 1]
[4, 4] *
[4, 6] !
[5, 6] !

طول ۲
(#:۸)
[0, 1, 2]
[0, 1, 5]
[0, 4, 6] !
[1, 2, 3]
[1, 5, 6] !
[2, 3, 1]
[3, 1, 2]
[3, 1, 5]

طول ۳ (#:۷)
[0, 1, 2, 3] !
[0, 1, 5, 6] !
[1, 2, 3, 1] *
[2, 3, 1, 2] *
[2, 3, 1, 5]
[3, 1, 2, 3] *
[3, 1, 5, 6] !

طول ۴ (#:۱)
[2, 3, 1, 5, 6] !



مسیرهای اصلی:
[4, 4] *
[0, 4, 6] !
[0, 1, 2, 3] !
[0, 1, 5, 6] !
[1, 2, 3, 1] *
[2, 3, 1, 2] *
[3, 1, 2, 3] *
[2, 3, 1, 5, 6] !

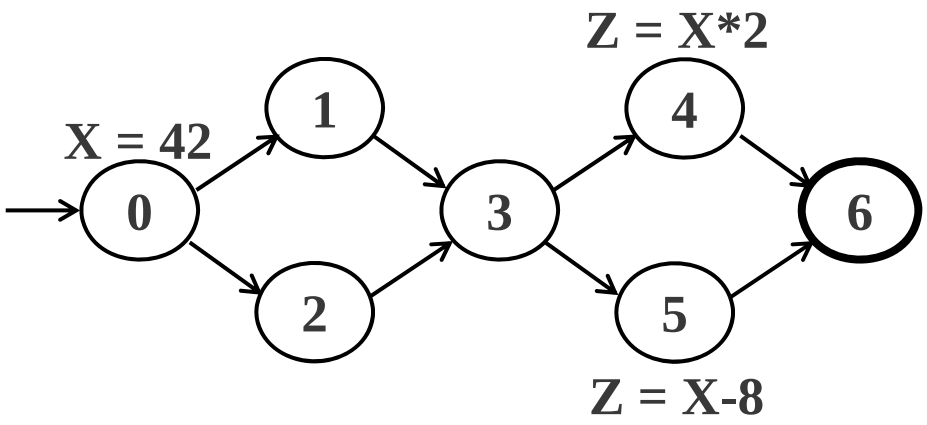
مسیرهای آزمون (از گره آغازین به گره پایانی):
[0, 1, 2, 3, 1, 5, 6]
[0, 1, 2, 3, 1, 2, 3, 1, 5, 6]
[0, 1, 5, 6]
[0, 4, 6]
[0, 4, 4, 6]

معیارهای جریان داده

هدف: سعی در اطمینان از این که مقادیر متغیرها به درستی نوشته و خوانده شده اند.

- **تعریف (def) = محلی از کد که در آن مقدار متغیر در حافظه نوشته می شود**
- **کاربرد (use) = محلی از کد که در آن مقدار متغیر از حافظه خوانده می شود**

- **def(n) یا def(e) = مجموعه متغیرهایی که در گره n یا در یال e تعریف می شوند**
- **use(n) یا use(e) = مجموعه متغیرهایی که در گره n یا در یال e به کار برده میشوند**



Defs: def (0) = {X}
def (4) = {Z}
def (5) = {Z}

Uses: use (4) = {X}
use (5) = {X}

مثال از تعریف و کاربرد

- تعریف (def) = محلی از کد که در آن مقدار متغیر در حافظه نوشته می شود
  - x=1;
  - x+=y;
  - x++;
  - float sin(int x) {.....} [در زمان فراخوانی یک زاویه در پارامتر نوشته می شود]

- کاربرد (use) = محلی از کد که در آن مقدار متغیر از حافظه خوانده می شود
  - if(x>2)…
  - z=x;
  - z=sin(x); [آرگومان خوانده می شود]
  - return x;

- (نکته: اگر پارامتری به صورت call-by-reference تعریف شده باشد (مثلاً float sin(int &x))، آرگومانی که به آن پاس می شود، به دلیل امکان تغییر آن در متد فراخوانی شده، علاوه بر use، دارای def هم هست.)

زوج های DU و مسیرهای DU

- **زوج DU**: زوجی از دو محل (l_i, l_j) که در آن، متغیر v در l_i تعریف (def) شده و در محل l_j به کار می رود (use).

- **عاری از تعریف (def-clear)**: مسیر از l_i به l_j با توجه به متغیر v عاری از تعریف است اگر بین l_i و l_j در گره یا یالی به متغیر مقدار دیگری داده نشود (به عبارت دیگر متغیر v بین مسیر def نشود).

- **دسترسی (Reach)**: اگر مسیر عاری از تعریفی از l_i به l_j با توجه به متغیر v وجود داشته باشد، تعریف v در l_i به کاربرد آن در l_j دسترسی دارد (می رسد).

- **مسیر DU**: یک زیرمسیر ساده از یک تعریف v به یک کاربرد v که با توجه به v عاری از تعریف باشد.

پیمایش مسیرهای DU

- **du (n_i, n_j, v) = مجموعه مسیرهای DU از n_i به n_j**
- **du (n_i, v) = مجموعه مسیرهای DU که از n_i شروع می شوند**

- **می گوییم مسیر آزمون p، زیرمسیر d را با توجه به متغیر v «پیمایش du می کند» اگر p، d را پیمایش کند، و d با توجه به v عاری از تعریف باشد.**

- **سه معیار داریم:**
  - **هر تعریف را به یک کاربرد برسانیم (همه تعاریف = all-defs)**
  - **هر تعریف را به همه کاربردهایش برسانیم (همه کاربردها = all uses)**
  - **همه زوج های تعریف-کاربرد را در نظر بگیریم (all def-uses)**

معیارهای آزمون جریان داده

- معیار اول: باید مطمئن شویم هر **def** به (حداقل) یک **use** می رسد.

پوشش همه تعاریف (ADC = All-Defs Coverage) :
برای هر مجموعه $S=du(n_i,v)$، TR شامل حداقل یک مسیر d در S است.

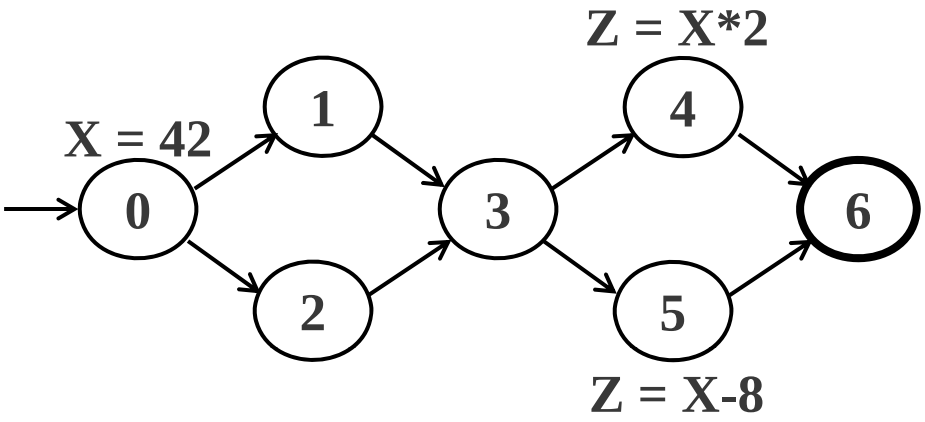
- معیار دوم: باید مطمئن شویم هر **def** به هر **use** ممکن می رسد.

پوشش همه کاربردها (AUC = All-Uses Coverage) :
برای هر مجموعه $S=du(n_i,n_j,v)$، TR شامل حداقل یک مسیر d در S است.

- معیار سوم: باید همه مسیرهای بین **def** ها و **use** ها را پوشش دهیم.

پوشش همه مسیرهای تعریف–کاربرد (ADUPC = All DU Path Coverage) :
برای هر مجموعه $S=du(n_i,n_j,v)$، TR شامل همه مسیرهای d در S است.

مثال از آزمون جریان داده



All-defs TR for *X*
[0, 1, 3, 4]

All-uses TRfor *X*
[0, 1, 3, 4]
[0, 1, 3, 5]

All-du-paths TR for *X*
[0, 1, 3, 4]
[0, 2, 3, 4]
[0, 1, 3, 5]
[0, 2, 3, 5]

مثال TestPat

```
public int pat (char[] subject, char[] pattern)
{
// Post:
//    if pattern is not a substring of subject, return -1
//    else return (zero-based) index where the
//     pattern (first) starts in subject
   final int NOTFOUND = -1;
   int  iSub = 0, rtnIndex = NOTFOUND;
   boolean isPat  = false;
   int subjectLen = subject.length;
   int patternLen = pattern.length;
   while (isPat == false && iSub + patternLen - 1 <
       subjectLen)
   {
     if (subject [iSub] == pattern [0])
     {
       rtnIndex = iSub; // Starting at zero
       isPat = true;
       for (int iPat = 1; iPat < patternLen; iPat ++)
       {
         if (subject[iSub + iPat] != pattern[iPat])
         {
           rtnIndex = NOTFOUND;
           isPat = false;
           break;  // out of "for" loop
         }
       }
     }
     iSub ++;
   }
   return (rtnIndex);
}
}
```

لیست متغیرها:

- subject
- pattern
- NOTFOUND
- iSub
- rtnIndex
- isPat
- subjectLen
- patternLen

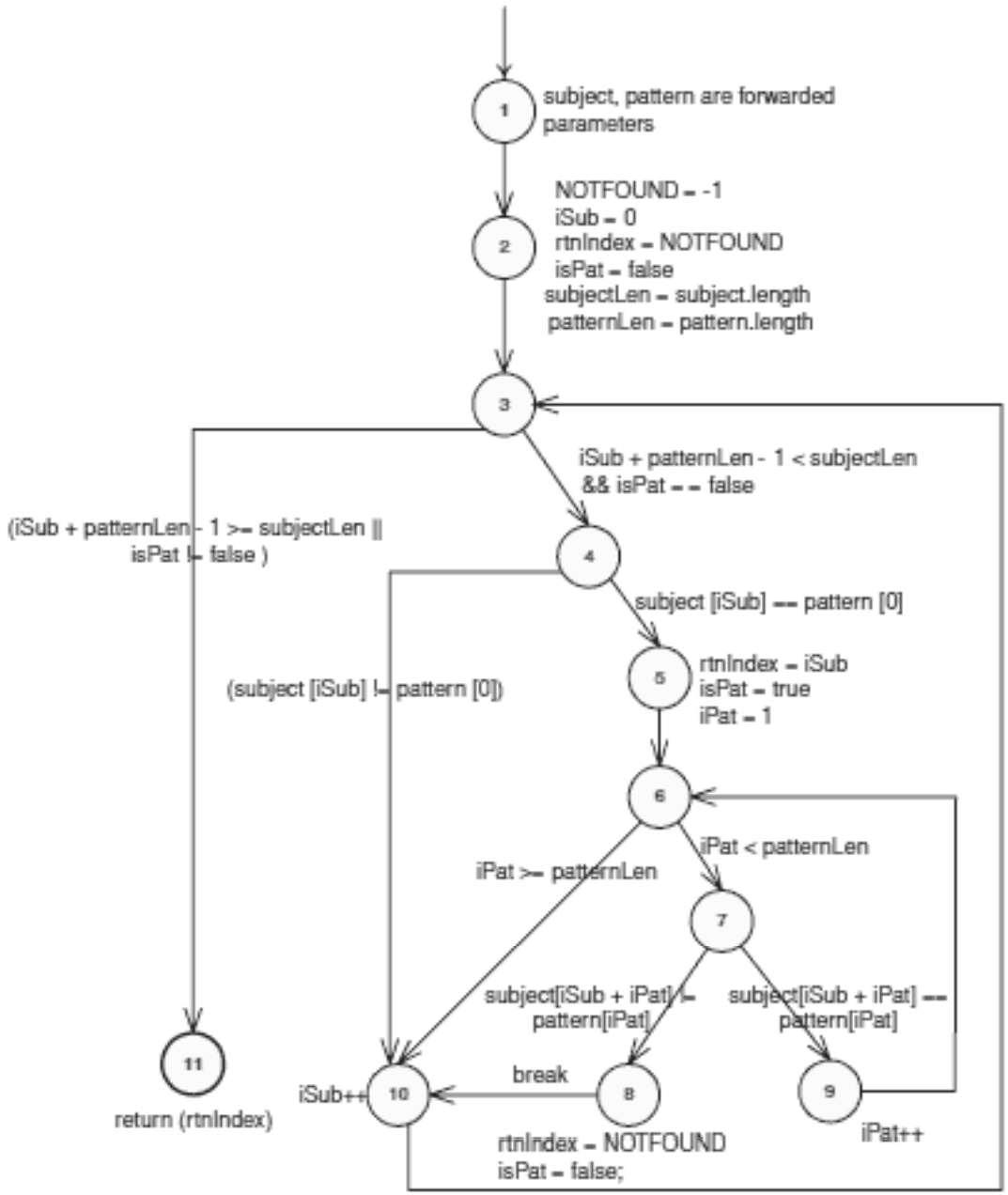


Figure 2.12. A graph showing an example of du-paths.

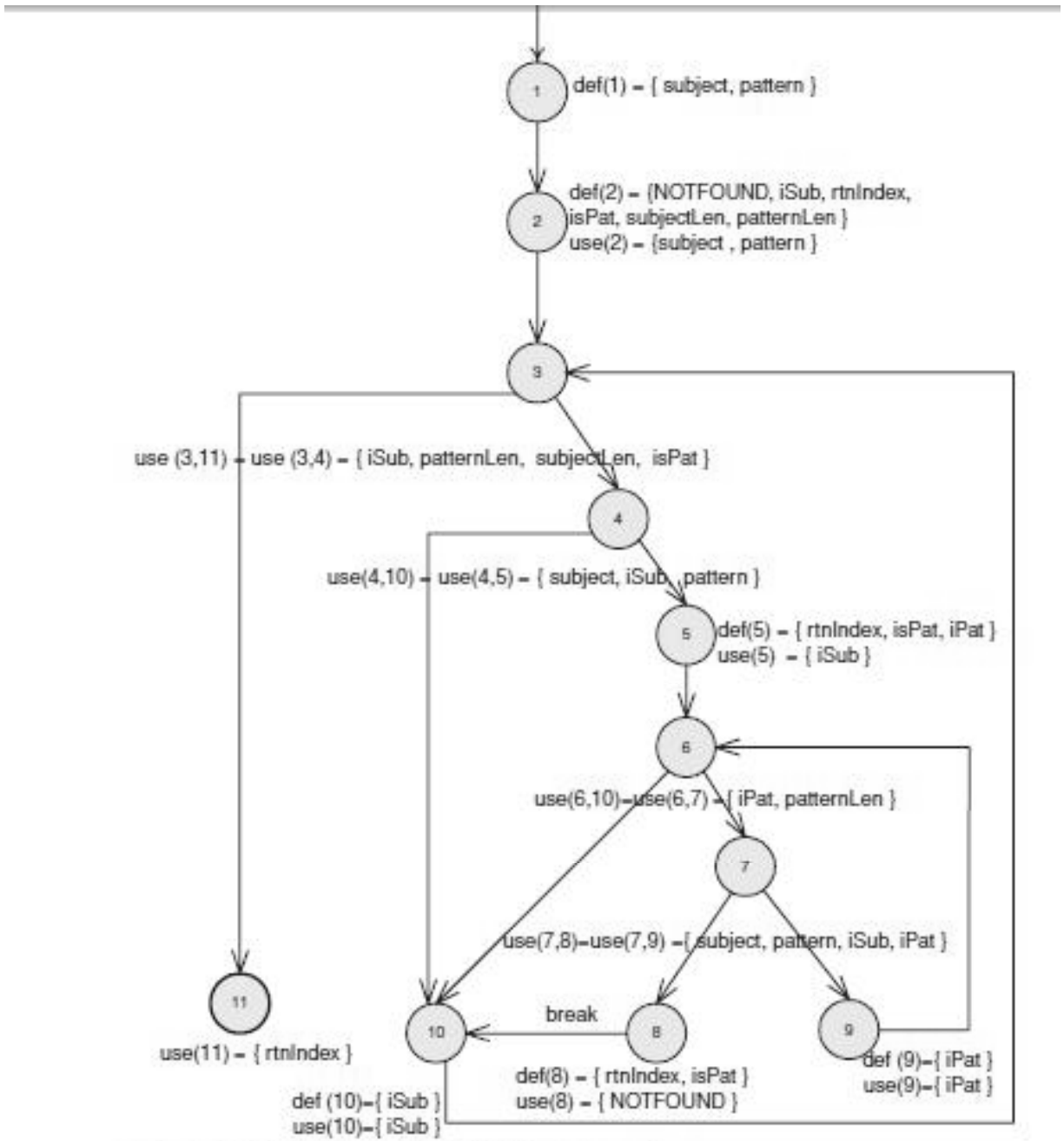
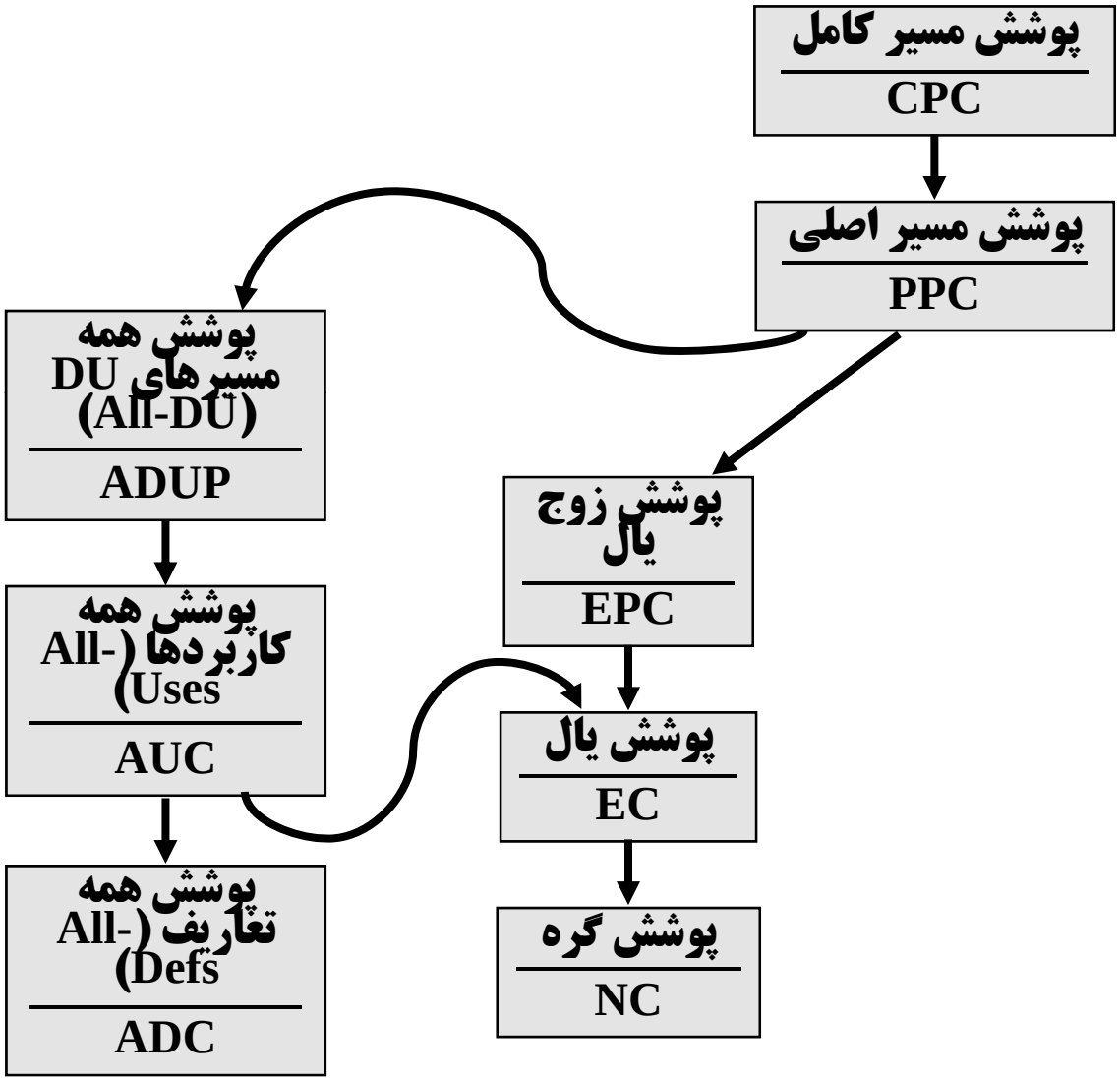


Figure 2.13. Graph showing explicit def and use sets.

رابطه در بر داشتن میان معیارهای مبتنی بر گراف



پایان جلسه دوم