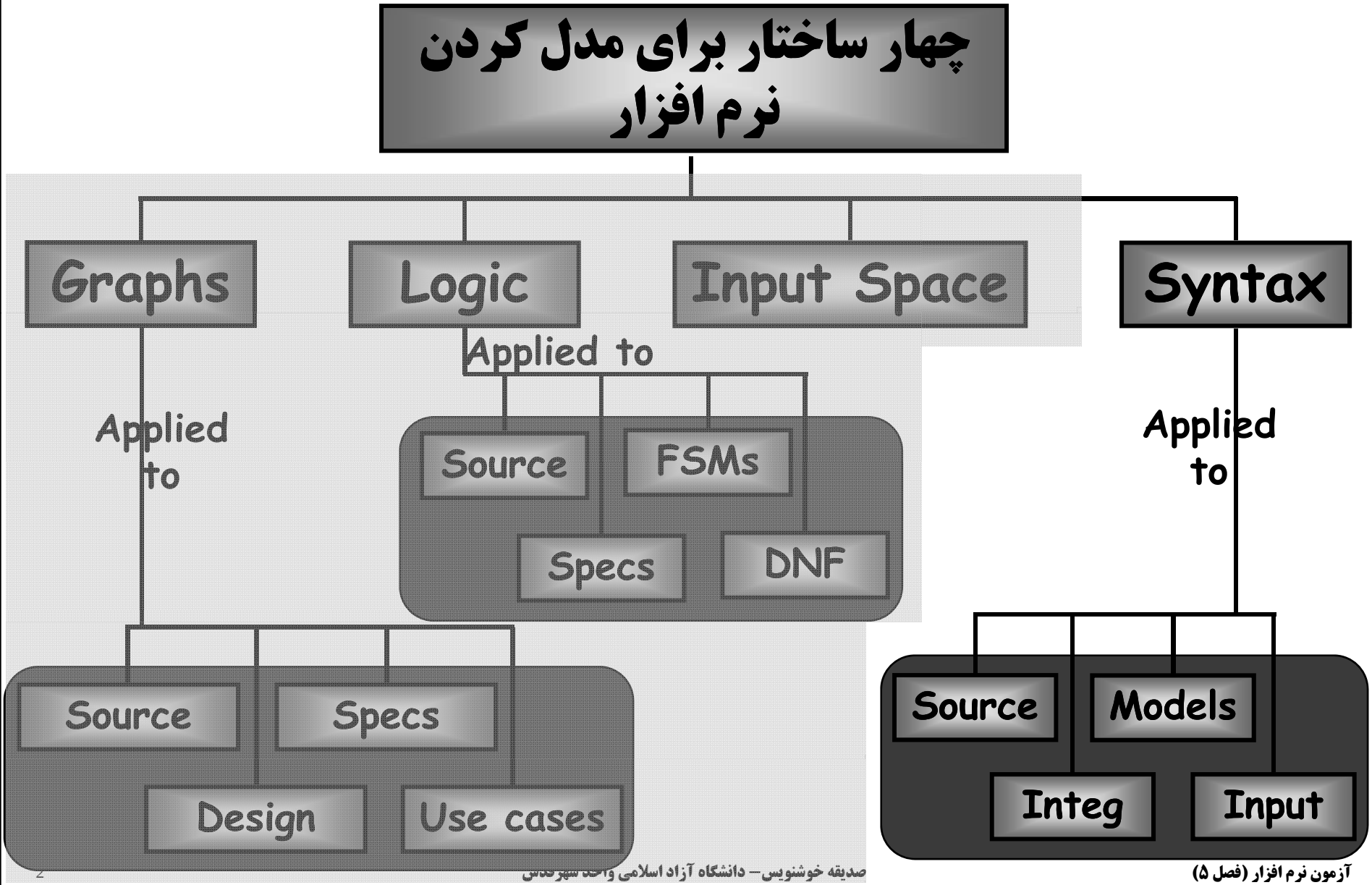
آزمون نرم افزار – فصل۵ بخش ۵–۱

آزمایش مبتنی بر نحو

صدیقه خوشنویس
دانشگاه آزاد اسلامی – واحد شهرقدس

پوشش فضای ورودی



استفاده از نحو (syntax) برای تولید آزمون ها

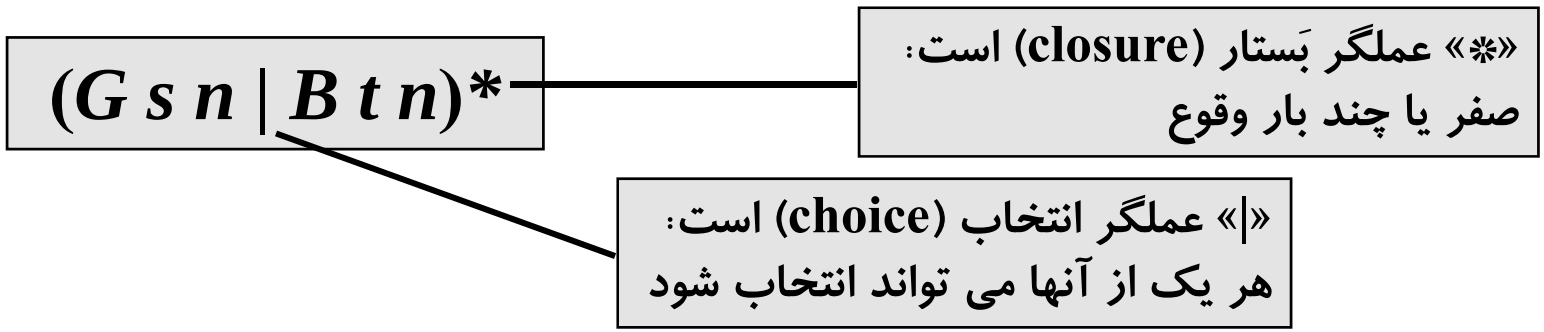
- بسیاری از برنامه ها دارای قواعد دقیق نحوی هستند.
- قواعد نحوی معمولاً به صورت نوعی گرامر (مثل BNF) بیان می شوند.
- توصیفهای نحوی را می توان برای فراورده های مختلفی نوشت؛ مثل:
  - برنامه ها
  - عناصری که باید با هم تجمیع (یکپارچه، متصل) شوند
  - مستندات طراحی
  - توصیفهای ورودیها
- آزمون ها با دو هدف کلی ایجاد می شوند:
  - به نوعی نحو را پوشش دهند.
  - قواعد نحو را نقض کنند! (آزمونهای نامعتبر)

BNF = Backus–Naur Form / Backus Normal Form

معیارهای پوشش گرامر

- مهندسی نرم افزار استفاده عملی زیادی از نظریه آتاماتا می کند:
  - زبانهای برنامه نویسی با BNF تعریف می شوند.
  - رفتار برنامه با ماشینهای حالت متناهی توصیف می شود.
  - ورودیهای مجاز با گرامرها تعریف می شوند.

- یک عبارت منظم (regular expression) ساده به شکل زیر است:



- نشان دهنده هر دنباله ای از "G s n" و "B t n" است.
- "G" و "B" می توانند دستور، متد، یا رویداد باشند...
- "s" ، "t" و "n" می توانند آرگومان، پارامتر، یا مقدار باشند.
- "s" ، "t" و "n" می توانند لیترال، یا مجموعه ای از مقادیر باشند.

(لیترال = نمادی که نشان دهنده یک مقدار ثابت است ≠ متغیر)

استخراج موارد آزمون از گرامر

- وقتی رشته ای که یک قاعده استخراج را برآورده می کند می گوییم «آن رشته در گرامر هست».
- یک مورد آزمون، دنباله ای از رشته ها است که عبارت منظم را برآورده کند.
- فرض کنید "s" ، "t" و "n" عدد باشند:

G 17 08.01.90

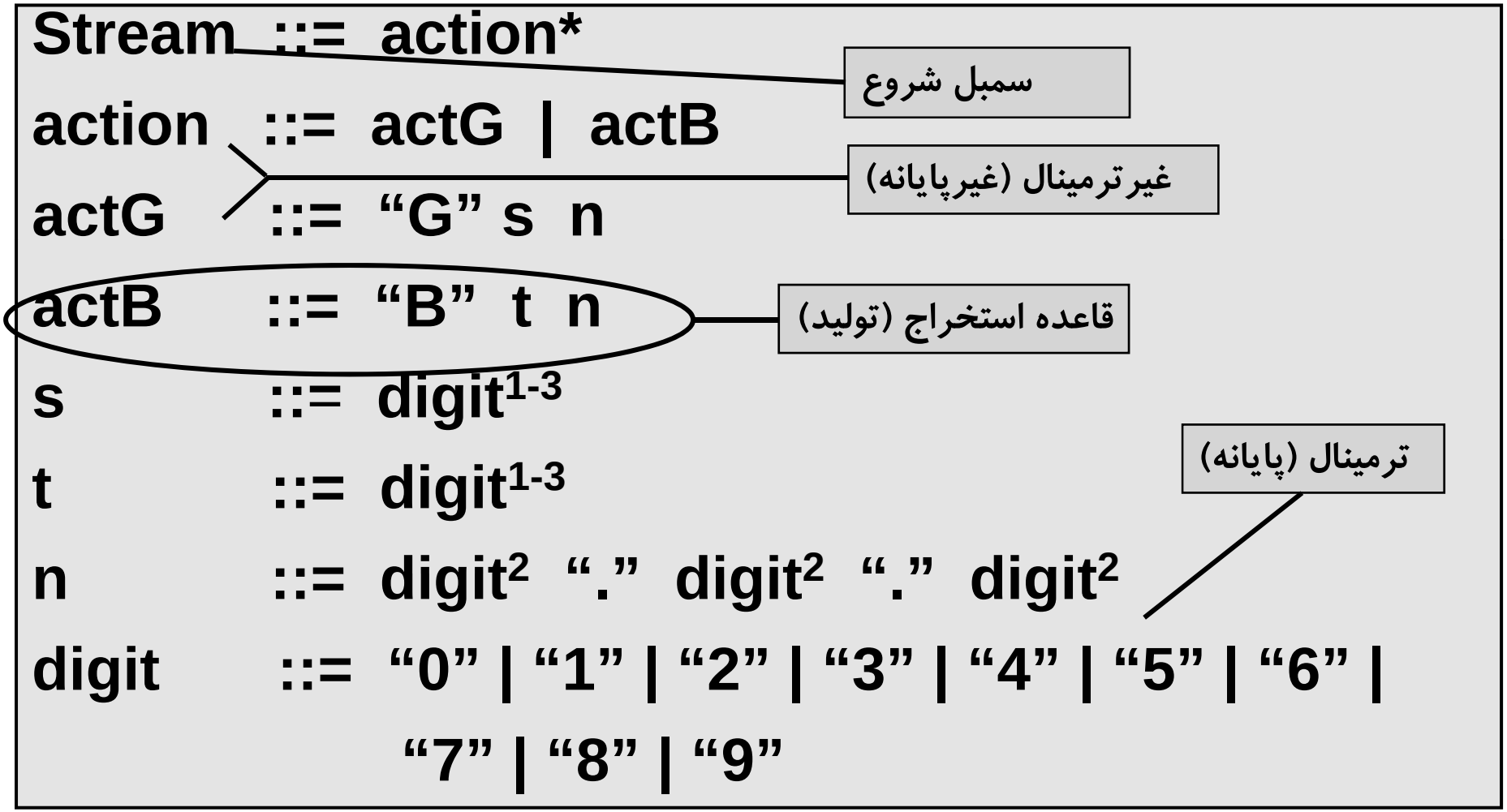
B 13 06.27.94

G 12 11.21.94

B 04 01.09.03

می تواند یک آزمون با چهار قسمت باشد،
یا چهار آزمون مجزا، و ...

گرامرهای BNF



استفاده از گرامرها

Stream ::= action action *

::= actG action*

::= G s n action*

::= G digit$^{1-3}$ digit^{2} . digit^{2} . digit^{2} action*

::= G digitdigit digitdigit.digitdigit.digitdigit action*

::= G 16 08.01.90 action*

...

- شناساگر(recognizer) : تعیین این که آیا یک رشته (یا آزمون) معین در گرامر هست؟
  - به این کار parse گفته می شود
  - برای آن ابزار وجود دارد
  - برنامه ها می توانند از آن برای اعتبارسنجی ورودیها استفاده کنند.
- مولد(Generator): استخراج رشته هایی که در یک گرامر معین هستند.

معیارهای پوشش مبتنی بر نحو

- رایج ترین و ساده ترین راه پوشش: از هر ترمینال و هر قاعده تولید حداقل یک بار استفاده شود.

پوشش سمبل ترمینال (Terminal Symbol Coverage) (TSC):
TR شامل هر سمبل ترمینال t در گرامر G است.

پوشش تولید (ProDuction Coverage) (PDC):
TR شامل هر قاعده تولید p در گرامر G است.

- PDC، TSC را در بر دارد.

- می توان از معیارهای پوشش مبتنی بر گراف روی گرامر استفاده کرد.

معیارهای پوشش مبتنی بر نحو...

- یک معیار دیگر (که عملی هم نیست) عبارت است از استخراج همه رشته های ممکن یک گرامر!!

پوشش استخراج (Derivation Coverage) (DC):
TR شامل هر رشته ممکن است که می تواند از گرامر G استخراج شود.

- تعداد آزمونهای TSC محدود است به تعداد سمبلهای ترمینال.
  - مثلاً در گرامر stream = ۱۳

- تعداد آزمونهای PDC محدود است به تعداد قواعد تولید.
  - مثلاً در گرامر stream = ۱۸

- تعداد آزمونهای DC به جزئیات گرامر وابسته است.
  - مثلاً در گرامر stream = حدود 2,000,000,000 !!

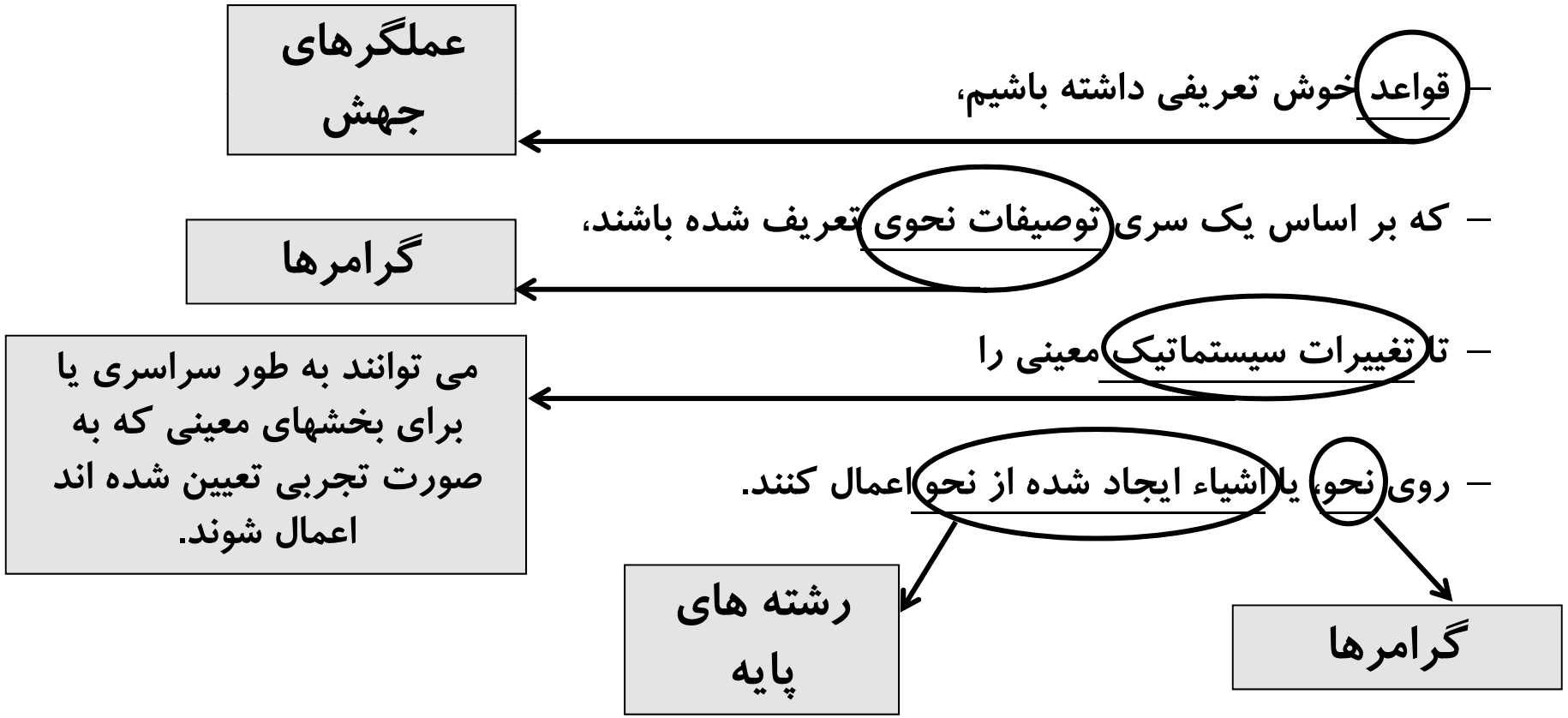
- هر سه پوشش TSC ، PDC و DC شامل آزمونهایی هستند که در گرامر باشند.
  - ولی باید آزمونهایی هم داشته باشیم که در گرامر نباشند!

آزمایش جهش (Mutation Testing)

- گرامرها هم رشته های معتبر را توصیف می کنند و هم رشته های نامعتبر را.
- هر دو نوع رشته ها می توانند به عنوان جهشگر (Mutant) تولید شوند.
- جهشگر، نسخه ای تغییر یافته از یک رشته معتبر است.
  - می تواند خودش، معتبر یا نامعتبر باشد
- جهش مبتنی بر دو چیز است:
  - یک سری عملگر جهش (Mutation Operator)
  - یک سری رشته پایه (Ground String)

جهش یعنی چه؟

- دیدگاه کلی این است:
- وقتی از تحلیل مبتنی بر جهش استفاده می کنیم که:



آزمایش جهش

- رشته پایه: رشته ای در گرامر
- عملگر جهش: قاعده ای که تغییر نحوی رشته های تولید شده از گرامر را مشخص می کند.
- جهشگر (موتانت): نتیجه یک بار اعمال یک عملگر جهش
  - که خود یک رشته است

جهشگرها و رشته های پایه

- نکته کلیدی در آزمایش جهش طراحی عملگرهای جهش خوب است.
  - عملگری که خوب طراحی شود منجر به تولید تست های قدرتمند می شود.

- گاهی رشته های جهشگر، مبتنی بر رشته های پایه هستند.
- گاهی هم مستقیماً از گرامر تولید می شوند.

  - برای ایجاد آزمونهای معتبر از رشته های پایه استفاده می شود.
  - برای ایجاد آزمونهای نامعتبر، نیازی به رشته های پایه نیست.

جهشگرهای نامعتبر

13 17 08.01.90

B 13 06.27

جهشگرهای معتبر

جهشگرها	رشته های پایه
B 17 08.01.90	*G 17 08.01.90*
B 45 06.27.94	*B 13 06.27.94*

پرسشهایی در رابطه با جهش

- **(۱) آیا می توان از بیش از یک عملگر جهش به طور همزمان استفاده کرد؟**
  - **یک رشته جهش یافته می تواند دارای یک بخش جهش یافته باشد یا چند بخش؟**
  - **تقریباً به طور قاطع می توان گفت : خیر! زیرا عملگرها ممکن است با هم تداخل داشته باشند**
  - **تجربه هم به ما می گوید: خیر!**

- **(۲) آیا هر کاربرد ممکن از عملگر جهش را باید در نظر بگیریم؟**
  - **بله، برای آزمون جهش برنامه ها ، این کار ضروری است.**

- **عملگرهای جهش برای زبانهای زیادی وجود دارند:**
  - **زبانهای برنامه نویسی مختلف (فرترن، لیسپ، آدا، C، C++، جاوا و ...)**
  - **زبانهای توصیف (SMV، Z، Object-Z، و توصیفهای جبری)**
  - **زبانهای مدلسازی (مثل نمودارهای Statechart و activity در زبان مدلسازی UML)**
  - **گرامرهای ورودی (XML، SQL و HTML)**

کشتن جهشگرها

- وقتی روی رشته های پایه جهش برای ایجاد رشته های معتبر ایجاد می کنیم، انتظار داریم برنامه در برابر جهشگر نسبت به رشته پایه رفتاری متفاوت داشته باشد.
- وقتی گرامر، زبان برنامه نویسی است:
  - رشته ها = برنامه ها
  - رشته های پایه = برنامه های موجود (از قبل)

- کشتن جهشگر : اگر $m \in M$ جهشگری برای یک اشتقاق **D** باشد و **t** یک آزمون باشد، می‌گوییم **t**، **m** را می کشد اگر و فقط اگر خروجی **t** روی **D** با خروجی آن روی **m** متفاوت باشد.

- **D** ممکن است به صورت لیستی از اشتقاق های دیگر یا به صورت رشته نهایی نمایش داده شود.

معیارهای پوشش مبتنی بر نحو (جهش)

- پوشش عمدتاً بر اساس کشتن جهشگرها تعریف می شود.

پوشش جهش (Mutation Coverage) (MC):
برای هر جهشگر $m \in M$، TR دقیقاً دارای یک نیازمندی است: کشتن m.

- میزان پوشش در آزمایش جهش، برابر است با تعداد جهشگرهای کشته شده.
- به میزان جهشگرهای کشته شده «رتبه جهش» گفته می شود.

معیارهای پوشش مبتنی بر نحو (جهش)...

- وقتی در گرامری جهش ایجاد می کنیم تا رشته های نامعتبر ایجاد شود، هدف آزمایش اجرای جهشگرها است تا ببینیم آیا رفتار درست است یا خیر.
- کار ما این است که فقط عملگرهای جهش را اعمال کنیم.
- دو معیار ساده خواهیم داشت: هر عملگر یا هر قاعده تولید را یک بار استفاده کنیم.

پوشش عملگر جهش (Mutation Operator Coverage) (MOC):
برای هرعملگر جهش، TR دقیقاً دارای یک نیازمندی است: ایجاد یک رشته جهش یافته m که با استفاده از عملگر جهش ایجاد شده باشد.

پوشش تولید جهش (Mutation Production Coverage) (MPC):
برای هرعملگر جهش، TR دارای چندین نیازمندی است: ایجاد یک رشته جهش یافته m که شامل هر قاعده تولیدی باشد که می تواند با استفاده از آن عملگر، جهش یافته باشد.

مثال

گرامر:

```
Stream  ::= action*
action  ::= actG | actB
actG    ::= "G" s  n
actB    ::= "B"  t  n
s       ::= digit[1-3]
t       ::= digit[1-3]
n       ::= digit[2] "." digit[2] "." digit[2]
digit   ::= "0" | "1" | "2" | "3" | "4" | "5" | "6" | "7" | "8" | "9"
```

رشته های پایه

G 17 08.01.90

B 13 06.27.94

عملگرهای جهش

- عوض کردن actG با actB
- جایگزین کردن رقم ها با رقم های دیگر

جهشگرهایی با استفاده از MPC

B 17 08.01.90	*G 13 06.27.94*
G 27 08.01.90	*B 11 06.27.94*
G 37 08.01.90	*B 14 06.27.94*
G 47 08.01.90	*B 15 06.27.94*
G 57 08.01.90	*B 16 06.27.94*
...	...

جهشگرهایی با استفاده از MOC

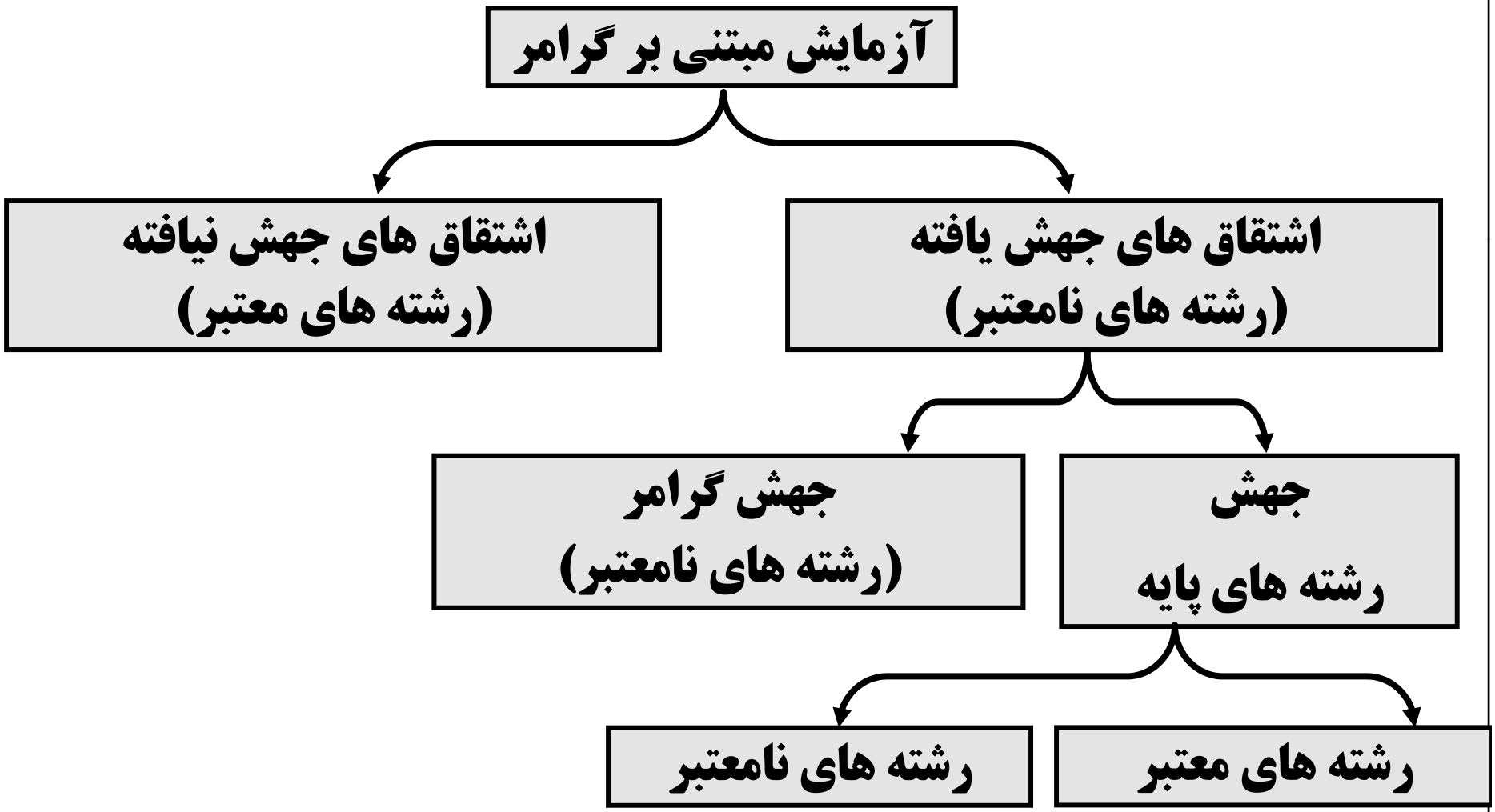
B 17 08.01.90

B 19 06.27.94

آزمایش جهش

- تعداد نیازمندیهای آزمون برای جهش وابسته به دو چیز است:
  - نحو فرآورده ای که در آن جهش ایجاد می کنیم
  - عملگرهای جهش
- آزمایش جهش به صورت دستی بسیار سخت است.
- آزمایش جهش بسیار مؤثر است: استاندارد طلایی آزمایش محسوب می شود.
- آزمایش جهش اغلب برای ارزیابی سایر معیارها مورد استفاده قرار می گیرد.

جهش به عنوان آزمایش مبتنی بر گرامر



پایان جلسه هشتم