جلسه اول:

**جزوه اول:**

اگر یک زبان را بتوانیم با گرامر ها مدل کنیم که هیچ اگر نشد جز زبان های غیر صوری هستند. در مجموعه ترتیب و تکرار مهم نیست ولی در رشته مهم هست و ترتیب متفاوت با هم متفاوت هستند، هر رشته یا هر چیزی به توان صفر میشود لاندا، اگر در یک رشته تمام حروف یکسان بود تعداد زیر رشته پیشوند و پسوند برابر هستند.

جلسه دوم:

**جزوه دوم:**

هر زبان یک مجموعه هست. تا دقیقه 29 از این بخش دیده شد.

L6 نمایشی از رشته هایی به طول زوج هست چون میتوان به قسمت شکوند حتی اگر تعداد الفبای ای و بی فرد باشد مجموع آنها زوج میشود.

L7 میشود تمام رشته هایی به طور فرد.

L9 همان رشت های آینه هستند و به دو مجموعه میتوان افراز کرد. رشته ای که طول الفباش با هم برابر هست طول آن زوج هست. برای اجتماع دقت کن تکراری ها چون دو بار شمرده میشوند پس باید منهای اشتراک کنیم.

L4 چون تمام رشته ها به طول چهار هست معکوس با خودش برابر هست. برای رشته های زوج l6 هم به همین صورت هست. l9 رشته های آینه ها هست. مکمل با متمم زبان یکی هست از هر کدوم آن میشود استفاده کرد. یک مجموعه به طول n را میتوان به n تا مجموعه افراز کرد. مثلا برای 3 به 3 تا پیمانه میشود برای 4 هم همین هست مثال 4 در جزوه آمده هست.

دقت کن با دو به توان n میتوانی تعداد زیر مجموعه های یک حالت رو بدست بیاری. دقت کن چون مجموعه مادر نامتناهی هست یا خود زبان با متمم زبان نامتناهی هست. مهم هست.

هر زبانی به توان صفر میشود لاندا. برای همین هست که در l\* ما لاندا داریم.

L+ به این معنی نیست که لاندا ندارد بلکه به خود زبان بستگی دارد که لاندا دارد یا نه.

پس تهی استار میشود لاندا چون هر زبانی به توان صفر لاندا دارد.

تو تست بگرد کجا یک چیز داخل l استار هست ولی داخل l نیست اونجا میتونی بگی این 2 تا برابر نیستند اگه گفتن جایی برابر هستند دقیقا دنبال همین بگرد که کجا این 2 تا مجموعه با هم برابر هستند. توی گزینه سوم زبان l تمام مضرب های 3 را حتی در صورت توان های بالا تر پوشش میدهد پس کلا l استار را پوشش میدهد. چرا تو گزینه چهار l+ با سیگما پلاس برابر هست ؟ چون لاندا عضو اون زبان نیست پس برابر هست و لازم نیست بنویسیم سیگما منهای ال استار.

1 ساعت از جلسه سوم دیده شد.

ببین نکته تست اول این هست که ما با الحاق جملات زبان به چیزی نرسیم که تو L نباشد در نتیجه زبان L با L\* برابر نیست. ولی در گزینه 3 چون هر چیزی بسازیم باز بر 3 بخش پذیر هست کلید تست میشود. دقت کن که L زیر مجموعه سیگما استار هست.

عکس روابط بسطار زبان ها درست نیست.

تنها در صورتی دو قسم جمله در تست دوم برقرار هست که طول زوج باشد. و در مجموعه های زوج l با بسطار خودش مساوی هست چونکه ترکیب زوج، زوج میسازد. اگر خواستند برابری سه تا جمله که دوتا مساوی 1 باشد را بدست بیاوریم باید ببینیم جوری میتوان شکست که ضرب اون 2 تا مساوی اون 1 شود اگر نشود یعنی اون زبان با l^\* برابر نیست. کلا خواستی ببینی برابر نیست سعی کن در هم ضرب کنی تا ترکیبی جدیدی بدست بیاد. فقط زبان هایی که l . l e^\* برابر تهی هست که پیشوند نداشته باشد. یعنی هیچ رشته ای پیشوند رشته دیگری نباشد. چون لاندا پیشوند هر رشته ای دیگر هست اگر شرط بالا برقرار باشد یعنی هیچ رشته ای پیشوند رشته دیگر نباشد ان وقت فقط لاندا جز اون زبان هست. گرامر وسیله ای هست برای نمایش دادن یک زبان. گرامر مولد یک زبان هست. دقت کن در زبان وقتی از یک متغیر مشترک استفاده میکنی یعنی میخواهی بگی الفبا وابسته به هم هستند. در خود بازگشتی ها لاندا نقطه بازگشت یا خروج رابطه بازگشتی هست.

تا دقیقه چهل از قسمت چهارم دیده شد.

دو گرامر زمانی معادل اند اگر زبان یکسانی تولید کنند یعنی هر 2 زبان بتوانند رشته های مشترک تولید کنند. واسه هر سطح گرامر زبان جدا بنویس بعد اشتراک کن یعنی یکی رو نوشتی ببر تو بعدی.

گرامر مستقل از متن یا نوع دوم یا آزاد از محتوا که قدرت بیشتری دارند نسبت به منظم بعد میریم سطح بعدی گرامر حساس به متن یا نوع اول که قدرت بیشتری نسبت به مستقل ها داریم و اگر گرامر حساس به متن هم برداریم میشویم بدون محدودیت یا قید و شرط یا نوع صفر یا گرامر دلخواه که هیچ محدودیتی نداریم و خیلی قدرتمند تر هستند. گرامر منظم نام دیگرش با قاعده هست.

**جلسه پنجم:**

گرامر منظم: یا سمت چپ یک متغیر همیشه باید پایانه باشد یا پایانه باشد یا خطی راست یعنی حداکثر کلا یک متغیر داریم اونم در سمت راست پایانه و همین برعکس که میشود خطی چپ یعنی متغیر همیشه یکی و همیشه راست پایانه هست یا پایانه خالی هست. فقط یکی از این 2 شکل باید باشد. دقت کن در هر قاعده باید به این شکل باشد. تو نوشتن اشتقاق ها به قاعده بازگشتی و خروج دقت کن. دقت کن یا باید خطی راست باشد یا خطی چپ هر 2 نمیشود. تعریف گرامر خطی: در سمت راست هر قاعده حداکثر یک متغیر وجود دارد در چپ هم دقیقا یک متغیر. هر گرامر منظم یک گرامر خطی هست ولی برعکس نه. دقت کن در منظم خطی راست یا چپ حداکثر 1 متغیر باید در هر قاعده باشد میتواند دقیقا 1 متغیر خالی باشد. برای تولید زبان a^n b^n را نمیتوان از گرامر منظم استفاده کرد توان این را ندارد پس این یک زبان منظم هم نیست. برای هر گرامر خطی راست یک گرامر خطی چپ وجود دارد و برعکس. وقتی خطی نیست منظم هم نیست.

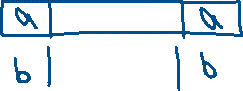
گرامر های حساس به متن: چپ و راست میتوانند هر ترکیبی از متغیر ها و پایانه ها باشند فقط طول سمت راست باید بزرگتر مساوی سمت چپ باشد پس بنابراین در این زبان ما قاعده لاندا نداریم.

گرامر بدون محدودیت: هر ترکیبی از ترمینال ها و متغیر ها در راست و چپ. دقت کن وقتی در سمت چپ aA باشد a حذف نمیشود بلکه باقی میماند و فقط متغیر حذف میشود. هر خطی مستقل از متن هم هست.

جلسه ششم:

دقت کن سیگما استار خودش مجموعه منظم هست ولی خیلی از زیر مجموعه هاش را نمیتواند زبان منظم تولید کند. اگر نا منظم باشد حتما نامحدود یا نامتناهی هست دقت کن.

اگر خواستی رشته هایی بسازی که حتما الفبا شروع و پایانش یکسان هست از قاعده SS استفاده کند. میتونی اینطوری فکر کنی که برای زبان هایی که ترکیب: ab, ab داشته باشیم راهی نیست جز اینکه دو متغیر کنار هم باشند و زبان مستقل از متن هست. دقت کن اگر سر یک رشته a و انتهای رشته b بود بهتر هست از ایده aSb استفاده کنی.



جلسه هفتم:

زبان حساس به متن قاعده لاندا ندارد ولی اگر بتوانیم قاعده لاندا را در نظر نگیریم میتوانیم بگوییم زبان حساس به متن هست. در نوشتن گرامر به وابستگی متغیر ها بهم دقت کن. چرا میگیم مستقل از متن چون جای متغیر کاری نداریم چپ و راست کجاست فقط جاش آن را بگذار. در وابسته به متن ها مثلا به چپ و راست کار دارد و بهش وابسته هست. زبان های خطی بین منظم و مستقل از متن هست. رشته هایی که تعداد ab باید با هم برابر باشد سعی کن از SS استفاده کنی. کلا 2 تا SS همه فرم ها به شکل ab که بخواهد طولشون برابر باشد.

**جلسه هشتم:**

ماشین با خواندن یک علامت به یک حالت میرود یعنی با یک علامت فقط و فقط به یک حالت میرود و تابع انتقال کامل هست یعنی حتما در هر حالت با خواندن هر علامت حرکت دارد و فقط یک حرکت دارد که این نشان دهنده کامل بودن و متناهی بودن هست. حالتی که میرود باید جز مجموعه حالات باشد. تعداد قوانین از اونجا که با خواندن هر علامت به حالت بعدی میرویم میشود تعداد حالا ضربدر تعداد الفبا. حالات نهایی دو تا دایره دارند و اگر رشته را بتواند تولید کند میگوییم این رشته به زبان ما تعلق دارد. از مرتبه o 1 خطی و محدود میتواند بگوید رشته تعلق دارد به زبان یا خیر.

N mod r، r تا عدد هست پس ما هم به r تا حالت در ماشین نیاز داریم واسه این زبان. بعد ببین مساوی با چه عددی هست همون حالت رو نهایی بکن. پس مثلا اون زبان فقط رشته های به مضرب 3 را میپذیرد. دقت کن هر زبانی که در مرتبه o 1 بتوانی حافظه تولید بکنی منظم هست در واقع اونایی که محدود هستند را فقط میپذیرد. دقت کن اگر یک زبان دادند گفتند واسه یک زبان که این زیر رشته توش باشه پیشوند هاش رو بنویس به تعداد پیشوند حالت بگذار. اگر گفتند که این زیر رشته را نداشته باشد تو بیا همون کار رو بکن منتها اون حالت bb را نهایی نکن. یعنی دقیقا عین همی داشتن پیشوند هست فقط حالت آخر نهایی نباشد. ببین اگر گفتند با یک رشته شروع شود تو بیا اول محدود کن که اون مقدار رو بری بعد لوپ بزن رو حالت نهایی. دقت کن اگر گفتند مثلا حاوی این زیر رشته باشد که طول آن n هست تعداد حالات نمیتواند کمتر از n+1 باشد. (n طول رشته مد نظر هست.)

دقت کن ماشین پذیرنده هست و گرامر مولد پس دو ماشین وقتی معادل هستند که زبان یکسانی یا رشته های یکسانی بپذیرند. برای یک زبان منظم بی شمار DFA میتوان طراحی کرد. ببین اگر گفتند تعداد a باید با b ها برابر باشد mod 2 آنها فرض کن هر دو زوج هستند و از مضرب 2 هستند یعنی 2 تا 2 تا تولید میشوند. و برو ببین تعداد حالت ها را بدست بیار مثلا برای ab به 4 حالت نیاز داریم که با هم هر دو برابر باشند و دو تا نهایی هست همونکه هر دو با هم برابر و مقدار آنها زوج هست حالا یا هیچی ندارن یعنی طول رشته صفر و یا طول ab 2 باشد به همین شکل. پس دقت کن وقتی میگویند برابر هست به mod 2 صفر هست یعنی رشته های زوج هست.

نکات: واسه زبان تیپ 10 باید یکی در میان از حروف استفاده کنیم. واسه زبان هایی که گفتند طول رشته باید mod چیزی باشد ببین اون چه عددی هست و از صفر تا اون عدد چند شماره هست به تعداد اون حالت داریم. بعد ببین از اون r تا مساوی کدوم هست همون را بکن نهایی.

**جلسه نهم:**

اگر زبان l منظم باشد آنگاه نات آن لزوما منظم هست. حتما هم DFA هست و دقت کن در اون ماشین جدید حالت های نهایی با هم عوض میشود اونایی که بودن نیستن و اونایی که نبودن حالت نهایی میشوند. دقت کن حالت تله هم به حالت نهایی تبدیل میشود. خانواده زبان های منظم تحت نات بسته هست. تحت اشتراک هم بسته اند یعنی حاصل اشتراک هم منظم هست تحت اجتماع و تفاضل مجموعه ای هم منظم یا بسته هست. در اجتماع اگر یکی از حالت ها نهایی باشد کل اون حالت نهایی هست برخلاف اشتراک که هر دو حالت باید نهایی باشند. تفاضل اونایی هست که حالات نهایی زبان اولی باشد ولی در دومی نباشد. عین همین اثبات برای اشتراک هست. دقت کن پیشوند یک زیر رشته به طول N دارای N+1 پیشوند متمایز میباشد. دقت کن حداقل تعداد حالات N+1 هست. کمینه سازی DFA: رابطه ادغام پذیری یک رابطه هم ارزی هست. و ادغام ناپذیری یک رابطه هم ارزی نیست یعنی اگر الف و ب ادغام ناپذیر بودند و ب با ث ادغام ناپذیر بودند نمیتوانی نتیجه بگیری الف با ث ادغام ناپذیر هست. روش به این صورت هست که ادغام ناپذیر ها را بدست میاوریم بعد اون دسته ای که جز اینها نیستند را با هم ادغام میکنیم. چه حالت هایی با هم ادغام پذیر هستند؟ آنهایی که با صرف یک رشته مشخص هر دو به حالت نهایی بروند یا با صرف یک رشته مشخص هر 2 به حالت نهایی نروند. ادغام ناپذیر: اول اونا که غیر قابل دسترس هستند را حذف کنیم. دوم بریم سراغ اونا که هر نهایی با غیر نهایی ها را زوج تشکیل بدهیم. حالت سوم همون صرف رشته به یک نهایی برسیم و با صرف همون رشته به نهایی نرویم به این میگویند زوج ادغام ناپذیر که علامت میزنیم. بعد هم دقت کن نهایی ها با نهایی باید بررسی شوند و غیر نهایی ها با غیر نهایی. دقت کن اگر شرایطی باشد که دو حالت با یک متغیر به دو حالت مختلف رفتند اگر یکی نهایی یکی غیر نهایی بود که هیچی ولی اگر هر دو نهایی یا غیر نهایی بودند اول بررسی کن ببین ادغام پذیر هستند یا نه اگر بودند اون مبدا هم ادغام پذیر میشود اگر نه که خوب مشخص هست. دقت کن با هر الفبا زبان باید چک شود حالات. NFA میتواند چندین نوع کمینه داشته باشد. حالت سوم خیلی مهم هست دقت کن از دو حالت مبدا با صرف الفبا یکسان به چه حالتی میرویم؟ اگر با یکی نهایی با یکی غیر نهایی آنها میشوند زوج ادغام ناپذیر. هم کلاسی: یعنی هعی داخل چند حالت لوپ بزنند و از آن بیرون نروند با صرف الفبا یکسان. از راه جدولی: محور ایکس را از حالت اول تا حالت یکی به آخری نشان گذاری کن و محور Y را از آخرین حالت به یکی مونده به اولین حالت نشان گذاری کن. اونهایی که در تمام حالات تیک خوردند یا ضربدر نخوردند، میشوند یک کلاس جدا. DFA زبان منظم یکتا نیست (یعنی بی شمار DFA میتوان طراحی کرد) ولی DFA کمینه زبان منظم لزوما یکتا هست.

برای هر زبان منظم تعداد حالات DFA کمینه خود زبان و متمم آن برابر است. اگر DFA کمینه زبان منظم L دارای N1 حالت نهایی و N2 حالت غیر نهایی باشد متمم آن دارای N2 حالت نهایی و N1 حالت غیر نهایی خواهد بود ( یعنی جای نهایی با غیر نهایی عوض شده است.) برای هر زبان منظم لزوما DFA دارای یک حالت نهایی نیست. دقت کن مثلا گفتیم الف و ب با هم ادغام پذیر هستند اگر الف با ث ادغام ناپذیر باشد چون الف و ب با هم زوج تشکیل دادند و ادغام پذیر بودند میگوییم ب با ث هم ادغام ناپذیر هست.

نکات: DFA متمم یک زبان دقیقا مثل خود زبان اصلی هست منتها حالات نهایی به شکل q-f هست یعنی حالاتی نهایی و غیر نهایی با هم عوض میشوند اونایی که نهایی هستند غیر نهایی و برعکس حتی حالات تله. ولی اگر زبان NFA بود برای پذیرنده زبان منظم L اونوقت نمیتوان گفت Q-F برابری میکند با ماشینی که پذیرنده زبان متمم L هست زیرا FA باید قطعی باشد. دقت کن هنگام اشتراک گیری تعداد حالات ضرب دکارتی میشود بعد با اونا بگیر ببین به چه الفبایی حرکت کرده اند دقیقا برو به همان حالت دو تایی یعنی ببین اولی با الفبا اول به کدام حالت رفته دومی با همان الفبا به کدام رفته بعد برو دقیقا به همون حالات خروجی این 2. حداقل تعداد حالات DFA به طول N که دارای N+1 پیشوند متمایز هست دقیقا همین مقدار حداقل تعداد حالات DFA هست. در اجتماع حالات نهایی میشود حالات نهایی اولی بعلاوه دومی منهای مشترک ها در تفاضل هم میشود حالات نهایی که در اولی نهایی هست و در دومی نهایی نباشد. در اشتراک تنها در صورتی حالات نهایی میشود که هر دو تا حالت زبان ها نهایی باشند. دو زبان وقتی با هم معادل هستند که ماشینی که آنها را توصیف میکند معادل باشد یعنی حالات نهایی که اولی میپذیرد دومی هم بپذیرد و حالاتی که اولی نپذیرفت دومی هم نپذیرد. دو نوع زوج ادغام نا پذیر داریم 1. اینکه یکی نهایی و دیگر غیر نهایی باشد. 2. اینکه هر دو با الفبای یکسان به حالات مختلف بروند. تعداد حالات DFA کمینه یک زبان با متمم آن برابر هست.

آنهایی که در جدول باقی ماندند میشوند حالات های جدا که توانستند با هم ادغام شوند.

**جلسه دهم:**

در NFA دیگر تعداد حالات کیو ضربدر الفبا نیست بلکه میتواند 2^Q حالت دارد در صورتی که در DFA ما Q حالت داشتیم از لحاظ تعداد تابع انتقال. حرکت با لاندا هم همین هست منتها لاندا را هم میپذیرد. شرط لازم برای پذیرش این است که رشته باید مصرف شده باشد تا به حالت نهایی برسیم به این معنی نیست که چون با حالت نهایی رسیدیم ماشین اون رشته را میپذیرد. دقت کن ماشین پذیرنده هست یا میگوید بله یا میگوید نه. در NFA عدم قطعیت نداریم چون با یک الفبا به دو حالت میرویم. هر NFA را میتوان به DFA معادل تبدیل کرد. اگر NFA ما N حالت داشت DFA ما 2^N حالت خواهد داشت اگر بدون لاندا باشد البته دقت شود که فقط تعدادی از اون حالات قابل دسترسی هست. اگر یکی از حالات نهایی باشد کل اون حالت نیز نهایی میشود در تبدیل این 2 بهم. تو درختی که میکشی حالت های تکراری را خط بزن دقت هم کن حالت جدید از حرکت اون الفبا به اون حالت ها بدست میاد. بعد تعداد حالات میشود اونایی که در اون درخت ضربدر نخوردند. بسطار لاندا یعنی مجموعه حالاتی که با حرکت لاندا میتوانی به آن برسی. هر حالت بسطار لاندا دارد. تو بسطار لاندا هر حالت خود اون حالت هم وجود دارد. بعد با لاندا حرکت کن ببین تا کجا میتونی برسی. اگر با یک حالت رفتی به حالتی که قبلا بسطار آن را نوشته بودی اجتماع بگیر خودش هم یادت نرود. برای تبدیل لاندا NFA به NFA معمولی اول حرکات لاندا را صبر چون میخواهیم آنها را حذف کنیم، بعد میای میبینی تا آن حالتی که داخل بسطار هم هست چه حروفی مصرف کردی با همون بیا به اون حالت بقیه لاندا ها را حذف کن. دقت کن در روش درختی برای تبدیل به DFA شما خروجی هر بسطار یک حالت میشود. و حالات های بعدی هم دقیقا همان خروجی بسطار ها هستند منتها ببین چه الفبایی دقیقا مصرف شده است، حالت تله با الفبا به خودش میرود فقط. در DFA تنها حالت پذیرش لاندا این هست که حالا شروع نهایی هم باشد مشخص هم هست چون زبان ماشین قبلی با لاندا به حالت نهایی میرفت. دقت کن اول بسطار ها را بدست بیار. دقت کن اگر یک حالت شروعی غیرقابل دسترس باشد اون زبان کلا تهی هست. زبان محدود زیر مجموعه منظم هست. دقت کن تعداد حالات متناهی قطعی دقیقا به انداز 2^N نیست چون خیلی از حالا غیرقابل دسترس نیست و حذف میشود. NFA برای پذیرش سیگما استار با 1 حالت داریم که با کمتر از آن نمیتوان این را پوشش داد بنابراین گزینه 3 رد میشود. دقت کن وارونه ماشین جدیدی که از یک ماشین منظم متنهای بدست میاد لزوما قطعی نیست. دقت کن در وارونه جهت ها تغییر میکند. ماشین حاصل یک NFA میشود حاصل از وارونه نه یک DFA و ما میدانیم یک DFA، NFA نیز هست ولی برعکس این درست نیست، دقت کن توی تست بعدی هعی وارونه بگیر تا گزینه در بیاد علاوه بر توضیحاتی که دادیم. تعداد حالات کمینه هر زبان منظم با کمینه تعداد حالات نات اون زبان برابر هست. ولی تعداد حالات کمینه یک زبان منظم با تعداد حالات کمینه وارونه آن زبان برابر نیست. دقت کن این ربطی به این ندارد که زبان منظم باشد وارونه آن هم منظم هست. هر NFAیی که حالت لاندا نداشته باشد و چند حالت نهایی داشته باشد حتما و حتما میتواند به یک NFA با یک حالت نهایی تبدیل شود و دقت کن بدون لاندا باید باشد. اثبات ساختاری: تمام نهایی را به حالت معمولی تبدیل کن همه حرکات را هم رسم کن عین قبل، یک حالت نهایی هم در نظر بگیر هر یالی به حالت نهایی داشتی به صورت متناظر به حالت نهایی وصل کن حتی اگر در حالت نهایی قبلی به خودش لوپ میزد، دقت کن نباید به قبلی ها لوپ بزند چون با صرف همون رشته ها باید به حالت نهایی برسیم ولی روی حالت نهایی سابق میتواند لوپ زد. دقت کن اینکه تک حالت نهایی باشد هم با گذار لاندا هم بدون لاندا میتوان ساخت هر NFAیی را. دقت کن اگر لاندا داشت ممکن هست بشود ممکن هست نشود. اگر لاندا نداشتیم حتما میشود به تک حالت پذیرش NFA. DFA با یک حالت شروع میشود اگر یکی دادند با چند حالت شروع میشد بیا همرو بکن حالت غیر شروع بعد یک حالت شروع در نظر بگیر با لاندا برو به بقیه و بعدش بیا به DFA تبدیلش کن. پس اگر کلا تک حالتی گفتند بیا و دنبال NFA باش که با لاندا حرکتی ندارد یا نمیپذیرد.

ماشین میلی: ماشین متنهای هست که خروجی آن به حالت فعلی و علامت ورودی وابسته است، برای هر ماشین میلی یک ماشین مور متعادل وجود دارد و برعکس اگر به ازای هر ورودی ممکن رشته خروجی یکسانی تولید کنند. خروجی را در میلی مینویسیم و به حالت جدید میرویم. در مور خروجی در حالات هست نه روی یال ها. یعنی اگر رفتیم داخل فلان حالت اون چیز را بفرست.

نکات: وارون یک ماشین DFA با متمم فرق دارد در وارون فقط جهت فلش عوض میشود ولی متمم حالات نهایی با غیر نهایی عوض میشود. و چون وارون ممکن هست یک NFA بشود از خود زبان که DFA هست پس تعداد حالات حداکثر برابر 2^Q هست که در اینجا Q همان m(L) هست. ولی تعداد حالات متمم با خود زبان برابر هست.

فصل سوم:

عبارات منظم پایه ای را نمیتوان به اجزای کوچکتر شکست، پرانتز برای تعیین اولویت هست که باهاش اولویت عبارات را عوض میکنیم، دقت کن که اگر قید نکرد کران پایین را پس صفر هم هست.

جلسه 12:

اگر زبان محدود باشد میتوان عبارت منظمی نوشت که استار نیاز نداشته باشد و سیگما 2 یعنی تعداد رشته ها به طول 2،

تا دقیقه 30 دیده شد.

ابزار زبان های منظم از لحاظ قدرت برابر هستند. دقت کن در تبدیل ماشین r\*، اون لاندا که مستقیم از حالت 0 به حالت نهایی رفته نشانه r^0 و اون چیزی که از نهایی به اولیه برگشته r^1,r^2 و ... هست. برای تبدیل ماشین متناهی به ماشین متنهای تعمیم یافته ما به حالات نهایی و شروع دست نمی زنیم و فقط حالات میانی را یکی میکنیم.

اثبات ساختاری گرامر: اگر سمت راست گرامر ها را وارون کنی اون گرامر حاصل زبانی هست که اون هم مثلا اگر اولی منظم بود منظم هست یا حتی مستقل از متن بود مستقل از متن هست و زبان های منظم تحت عملگر وارون بسته هستند.

جلسه چهارده:

فصل چهارم:

زبان هایی که تعداد برابر دارند منظم نیستند چون حافظه ای ندارد. زبان های منظم تحت اجتماع نا محدود بسته نیستند و اشتراک نا محدود هم لزوما یک زبان منظم نیست. خارج قسمت راست میشود پیشوند رشته هایی از صورت که رشته های مخرج پسوند آنها هستند. خارج از قسمت چپ دقیقا برعکس همین یعنی پسوند رشته های صورت که رشته های مخرج پیشوند آنها هستند. به آخر رشته ها نگاه کن در راست. در خارج قسمت چپ به اول رشته ها نگاه کن. چپ و راست لزوما با هم برابر نیستند و در بعضی مواقع یک سری رشته ها که در زبان اصلی نیستند در زبان خارج از قسمت هستند. تهی هم میتواند باشد هنگامی که پیشوند صورت با رشته مخرج برابر نباشد. دقت کن کل رشته مخرج را میزنی ها یعنی حتی یک a بود هم تاثیر دارد. اگر زبان مخرج لاندا داشت آنوقت زبان صورت زیر مجموعه خارج قسمت راست و چپ این 2 هست. عکس این درست نیست. یعنی اگر زبان صورت زیر مجموعه زبان مخرج باشد لزوما نمیتوان گفت لاندا عضو زبان مخرج هست. دقت کن وقتی زیر مجموعه محض هست زیر مجموعه مساوی هم هست ولی برعکس درست نیست. منظم ها تحت خارج از قسمت بسته هستند. زبان منظم رو هر چی بود خارج از قسمت منظم هست. در همریختی قرار هست هر رشته روی الفبای سیگما رو معادل کنیم با یک الفبا روی گاما. دقت کن بی شمار میتوانیم تعریف کنیم. هر چیزی که نیاز به حافظه نداشت میتوانی به عنوان پذیرنده به حساب بیاری. اگر یک زبان منظم باشد تحت هر تابع دلخواه لزوما منظم هست. روی عبارت همریختی اعمال میکنی پس چون تونستی واسه اون عبارت همریختی بسازی واسه اون زبان هم میتوانی بسازی پس اون زبان هم منظم هست. اگر تصویر همریختی نا منظم باشد آنگاه خود L لزوما نا منظم هست. اگر تصویر منظم بود نمیتوانی بگویی L لزوما منظم هست. وارون همریختی یعنی میخواهیم برگردیم به زبان اولیه یعنی یک زبانی داریم که تصویر همریختی هست منتها میخواهیم ببینیم چه رشته یا رشته هایی بوده که شده این زبان فعلی. در وارون همریختی باید بزرگترین مجموعه ممکن را در نظر بگیریم.

اون بزرگترین مجموعه ممکن ترکیبش در ذهنت باشد معمولا این شکلی هست. خود زبان زیر مجموعه وارون همریختی این زبان هست و همریختی وارون همریختی هم زیر مجموعه خود زبان هست. اگر L منظم باشد وارون همریختی لزوما منظم هست.

جلسه پانزدهم:

لم تزریق: در یک گراف با n راس طول طولانی ترین مسیر که راس تکراری نبینیم n-1 هست و تعداد گره ها n تا هست یعنی این قدر دیدیم تکراری ندیدیم. این لم برای اثبات نا منظم بودن یک زبان استفاده میشود دقت کن اونجا که گفتیم اندازه XY کوچک تر از N هست دقیقا بخاطر همین نکته ای بود که بالا تر گفتم و وقتی Y بزرگتر مساوی 1 هست یعنی بعد از N ما اولین لوپ یا حلقه را میبینیم. دقت کن که اگر منظم باشد ما میتوانیم به XYZ تقسیم کنیم و این حرکات را بزنیم. و در واقع در این بازی ما باید I را طوری بگیریم که حریف که میگوید منظم هست شکست بخورد. هیچوقت I انتخاب شده 1 نیست چون میشود رشته اولی. آینه ای زوج واسه این رشته ها نمیتوانیم منظم بنویسیم و از خطی استفاده کردیم مثل aSa | bSb | لاندا. دقت کن لم واسه اثبات نا منظم بودن هست دقت کن آینه ای زوج اگر روی تک نمادی باشد میشود رشته هایی به طول زوج که منظم هم هستند پس به صرف اینکه گفتند آینه ای زوج یعنی WW^R نگو که منظم نیست شاید تک حرفی باشد. انتخاب رشته هم در این بازی مهم هست.

برای a^P که در آن P اول باشد زبان متناهی و پذیرنده متنهای نمیتواند این نامتناهی را بپذیرد و نا متناهی میخواهیم. چون اون مجموعه را توانستیم مجزا کنیم به مجموعه های جدا پس اول نیست پس اون زمان منظم نیست. در مجموع ها به متمم آن هم فکر کن شاید کوچک تر شد و توانستی سریعتر حل کنی. کران پایین را ننویسیم یعنی مساوی 0 هست. برای A استار B استار اگر سیگما استار را ازش کم کنیم میشود مجموعه رشته هایی که حداقل یک فرم BA در آن حضور داشته باشد.

جلسه شانزدهم:

اثبات دیگری که برای نا منظم بودن یک مجموعه ابزار مای هیل نرود هست، برای هر رشته میگوید که باید عضو زبان باشند یعنی اگر مثال نقض آوردی تمام هست و منظم نیست. دقت کن اگر x و y از همدیگر غیر قابل تشخیص نباشند یعنی عضو یک کلاس هستند مثلا a با aa یعنی به یک کلاس هم ارزی تعلق دارند. غیر قابل تشخیص بودن یک رابطه هم ارزی و قابل تشخیص بودن رابطه غیر هم ارزی هست چطور میفهمیم غیر قابل تشخیص بودن ؟ XZ عضو زبان باشد ولی YZ نباشد یا برعکس میشود قابل تشخیص بودن و به یک کلاس کاری تعلق ندارند ولی برعکس این باشد یعنی غیر قابل تشخیص هستند. این قضیه میگوید یک زبان منظم است اگر رابطه های هم ارزی دارای تعداد محدودی کلاس هم ارزی باشد یعنی تعداد کلاس های هم ارزی نسبت به زبان منظم L برابر تعداد حالات DFA کمینه هست.

در واقع هر حالت متناظر یک کلاس هم ارزی هست چرا طبق اون DFA چون اگر XZ عضو زبان نباشد حتما و حتما YZ هم عضو زبان نیست چون ما با مصرف رشته به یک حالت یکسان رسیدیم. A+ یعنی همه رشته های این شکلی در یک کلاس هستند مثل aa یا a و دو به دو غیر قابل تشخیص هستند. در واقع تعداد حالات ماشین دو به دو غیر از حالت نهایی با هم غیر قابل تشخیص هستند. و هر عضو از کلاس b با هر عضو از کلاس ab قابل تشخیص هستند یا نسبت به یکدیگر متمایز هستند اما اعضای هر کدام مثلا در کلاس b غیر قابل تشخیص هستند دو به دو. اگر یک کلاسی بود که اعضای آن دو به دو بتوانند متمایز یا قابل تشخیص باشند چون تعداد کلاس های هم ارزی آن نامتناهی میشود پس چون تعداد حالات بینهایت میشود زبان نامنظم هست. اگر منظم باشد تعداد کلاس ها برابر با تعداد حالات DFA کمینه هست. در قبلی از DFA به زبان و کلاس ها رسیدیم ولی در مثال بعدی ما کلاس ها را داریم ولی زبان را نداریم ولی میخواهیم به DFA مورد نظر برسیم که مشخص هست که چون 3 دسته کلاس داریم پس میشود 3 حالت ABC و زبان قطعا منظم هست چون تعداد کلاس ها محدود هست. چون لاندا به کلاس هم ارزی A تعلق دارد پس A حالت شروع ما هست. A همه به a ختم میشوند و B به b و دقت کن همه اعضای این کلاس ها دو به دو غیر قابل تشخیص هستند. ولی اعضای A با اعضای B قابل تشخیص هستند. اون حالات شبیه به هم واسط بین این حالات هستند. چون 3 حالت داریم پس 8 انتخاب برای حالات نهایی داریم یا هر سه تا باشند یا هر سه تا نباشد ( که درست نیست چون این DFA کمینه نبوده پس) و ... ، هر کدام که حالت نهایی بود اون کلاس میشود زبان اون DFA. دقت کن واسه C چون خودش هم مینیمال هست وقتی نهایی باشد متمم هم میتواند نهایی باشد. پس اگر یک زبانی باشد که نیاز به حافظه نامتناهی داشته باشد و پذیرنده متناهی نتواند نیاز را بر طرف کند و اگر بشود براش طراحی کرد از رده زبان های منظم میاد بیرون مشروط به اینکه زبان formal باشد و قوی ترین ماشین بتواند آن را بپذیرد. WW^R تمام آینه های به طول زوج را تولید میکند. ولی فرق دارد WW^RU چون قسمت اول میتواند به لاندا برود و U میتواند هر فرمی از روی AB استار شود پس چون میتوانیم این کار را بکنیم این زبان میشود منظم چون سیگما استار منظم هست. در واقع L1 یک نمایش دیگری از سیگما استار هست حتی اگر رشته به شکل abba باشد هم چون داخل سیگما استار هست پس مشکلی ندارد. ولی اگر بستار استار نباشد و بستار پلاس باشد چون باید ab به تعداد n تا بیاد حتما باید به خاطر بسپاریم چون آینه ای هست و باید واسه اون قسمت هم تولید کنیم پس میشود نا منظم. واسه l6 میایم میگیم اون قسمت تقارن را بده به w بعد اون قسمت هایی که آینه هستند را با u و v بساز. Ww طول زوج را تولید میکند ولی لزوما دلیلی ندارد هر طول زوجی دقیقا برابر باشند. زبان l10 حتما واسه رشته های طول زوج هست پس همان یک نمایش دیگری از زبان منظم هست. دقت کن منظم ها زیر مجموعه محض خطی ها هستند. زبان های نا منظم تحت اجتماع لزوما بسته نیستند یعنی حاصل اجتماع میتواند منظم باشد. تهی چون مجموعه محدود هست و هر مجموعه محدود منظم هست پس تهی هم منظم هست و دقت کن نا منظم تحت اشتراک بسته نیستند. باز هم تاکید میکنم وقتی کف نمینویسیم یعنی بزرگتر مساوی صفر هست. از یک مجموعه نا منظم محدود اضافه کنی یا کسر کنی هنوز هم نا منظم باقی میماند. ولی اگر نا محدود اضافه کنی مثل یک نا منظم دیگه میشود نا منظم. هر زیر مجموعه ای از منظم لزوما منظم نیست مثل a^nb^n. L23 نا منظم هست چونکه باید کوچکترین رشته ها نصف بشود و دسته اول با دسته دوم برابر باشد پس نمیتوان از منظم استفاده کرد، و هر رشته را بتوانی به WW بشکنی که N بار تکرار شود و حداقل 2 باشد. در l24 دیگر کاری به وسط نداریم وسط را میخواد آینه ای باشد یا نباشد را از وسطی ها میسازیم فقط باید با عبارت منظم اول و آخرش را بسازیم که با همدیگر وارون باشند. در زبان l27 چون الفبا دو نمادی هست نا منظم هست ولی تک نمادی بود میشد منظم چون میشد رشته هایی به طول 4 ولی چون دو نمادی هست باید در حافظه بسپاریم با لم تزریق هم میشود.

خواص الگوریتمیک زبان های منظم: همان مسئله تصمیم پذیر هست که اگر الگوریتمی وجود داشته باشد و هر نمونه از مساله را به الگوریتم بدهیم خروجی الگوریتم بله یا نه باشد. به این صورت هست که بخواهیم بفهمیم مثلا یک زبان تهی هست یا نیست این هست که ببینیم با مصرف یک رشته میتوانیم به یکی از حالات پایانی برسیم یا نه یک مسئله دیگر این هست که ببینیم رشته به زبان تعلق دارد یا خیر که از مرتبه تتا n هست که n تعداد حالات هست. مسئله سوم این هست که آیا زبان منظم l نا متناهی هست یا نیست که میبینیم آیا DFA یک سیکل دارد که لوپ بزنیم یا نه البته به این دقت کن که ما میتوانیم لوپ داشته باشیم ولی بینهایت نباشد اصلا میتواند لوپ باشد و زبان تهی شود پس دقت کن به صرف لوپ داشتن نمیگوییم نا متناهی هست. یعنی دوری که در مسیر پذیرش نباشد ارزشی ندارد و بینهایت نیست. همین برای عبارت منظم هم صدق میکند چون قدرت اینها با هم برابر هست. پس اگر استار باشد بیان گر نامتناهی بودن نیست. اگر نامتناهی باشد حتما استار یا پلاس دارد ولی اگر استار یا پلاس داشته باشد لزوما نا متناهی نیست. خواستی ببینی متناهی هست یا نیست عبارت منظم رو به NFA تبدیل کن بعد به DFA بعد میتوانی تشخیص دهی متناهی هست یا نامتناهی. مسئله 4: چه عبارت منظم چه DFA بدن فرقی نداره عبارت منظم رو میتوانیم به DFA تبدیل کنیم و دقت کن اگر دو زبان منظمی که داریم حالات کمینه DFA آنها برابر باشد یعنی معادل هستند. چون DFA کمینه منحصر به فرد هستند راه حل دوم این هست که L1-L2 و برعکس را اجتماع بگیریم DFA بکشیم اگر تهی شد پس این 2 با هم برابر هستند. مسئله 5: (دقت کن هر ابزار توصیف دادند تبدیل به DFA کن) برای حل این ببین L1-L2 تهی باشد و همچنین L2-L1 تهی نباشد یعنی L1 زیر مجموعه محض L2 هست. مسئله 6: رشته های به طول زوج وارون خودش هست مثل سیگما 2، اگر L منظم وارون هم منظم هست این هم از طریق DFA. اگر کمینه ها برابر شد یعنی مجموعه های یکسانی بودند. مسئله 7: ببین به DFA کمینه برسیم که یک حالت داشته باشد و اون هم نهایی باشد یعنی توصیف کننده تمام رشته ها یا سیگما استار هست. عمدتا مسائلی که در حوزه زبان منظم مطرح میشود تصمیم پذیر هستند. مسئله 8: تنها راه برای پذیرش رشته پوش این هست که حالت شروع هم حالت نهایی باشد پس مسئله تصمیم پذیر هست. مسئله 9: آیا دو عبارت منظم رشته مشترکی را توصیف میکنند؟ به DFA تبدیل میکنیم بعد ببینیم اشتراک این دو FA که یک ماشین هست مخالف تهی هست یا نیست اگر نا تهی باشد این دو ماشین حداقل یک رشته مشترک میسازد اگر تهی باشد یعنی هیچ رشته مشترکی را نمی سازند.

**فصل پنجم:**

**جلسه هجدهم:**

ابزار ما و قدرت ما بیشتر میشود در زبان های مستقل از متن. در درخت اشتقاق اگر برگ ها را از چپ به راست ملاقات کنی میشود رشته مورد نظر. برای هر گرامر مستقل از متن برای هر قاعده ای که سمت راست آن یعنی RMD برابر با K باشد طول آن و K بزرگتر از 1 باشد ارتفاع درخت برابری میکند با بین لگاریتم طول رشته w در مبنای k و طول رشته w منهای 1 بر روی k-1. برای بدست آوردن زبان یک گرامر بیا خط به خط بدست بیار که هر خط چه زبانی را تولید میکند چون در نهایت بخشی از زبان را بدست آورده ای. برای a^nb^n ما از گرامر خطی استفاده کردیم در پایین صفحات در صورتی که میتوانستیم از ابزار قوی تر مثل گرامر مستقل از متنی که خطی نباشید استفاده کنیم. گرامر خطی قدرت تولید این زبان و زبانی که در صفحه بعد هست را دارد که منظم نیستند ولی در G3 ما از گرامر مستقل از متن استفاده کرده ایم. برای نوشتن زبان به گرامر هم فقط کافی است از بالاترین لایه مثلا در مثال اول AC را ترتیب دهی کنیم بعد سراغ درونی ترین بخش ها برویم. در زبان بعدی L2 مکمل L2 هست که فرد ها هستند و آنها که متقارن نیستند یا آینه ای نیستند. در زبان مکمل l2 با A اول میایم که اون طول های فرد را بسازیم بعد با B,D میایم اونهایی را میسازیم که زوج هستند ولی آینه ای نیستند یعنی میزاریم به صورت آینه ای لوپ بزنند ولی نقطه بازگشت میشود D که اون آینه و متقارن بودن را خراب کنیم. در L3 میخواهیم آینه باشد پس اول و آخر را میسازیم بعد اون وسط با A یا B تبدیل به فرد بودن میکنیم. وقتی خطی باشد یعنی لزوما مستقل از متن هست. L4 یعنی چه طول زوج باشد چه طول فرد آینه ای باشد. در زبان L5 میخواهیم تمام زبان هایی باشند که زوج آینه ای نیستند یا متقارن نیستند اول میگذاریم متقارن بیاد جلو بعد با aAb یا bAa متقارن بودن را خراب میکنیم با A داریم زوج ها را میسازیم. در L6 میگوییم تمام نا متقارن ها چه زوج چه فرد. یعنی اول میزاریم متقارن بیاد جلو بعد دوباره مثل قبلی خراب میکنیم متقارن ها را با دو قاعده آخری بعد با A هر دوری خواستیم میزنیم. دقت کن حتما لازم نیست وقت بزاری اثبات کنی نا منظم هست وقتی متقارن ها نامنظم هست پس متمم این زبان که میشود همه نا متقارن هم نامنظم هست (منظور L4,L6 هستند). ایده l7 هم به همین شکل هست یعنی اول میزاریم زوج بیاد جلو یعنی دوتایی ولی با 2 قاعده آخر که نقطه بازگشت هست میگذاریم که تبدیل به فرد هستند چون بلاخره یکی باید اضافه بماند برای l7 هم میتوانیم یک گرامر منظم بنویسیم پس نیازی نبود از خطی استفاده کنیم. دقت کن که در زبان l8 اول با قاعده A میایم K را میسازیم بعد با قاعده D میایم M ها را میسازیم. برای قسمت B باید به دو زیر مجموعه بشکنیم چون هیچ حالتی نمیتواند شکل بگیرد که 2 متغیر نداشته باشد پس اول a^nb^n را با D که قبلا ساختیم و با E میایم BC را به توان K میسازیم در واقع داریم K ها را میسازیم. برای موقعی که توان اولی با آخری متفاوت بود ولی خواستی فرم حفظ شوم از مستقل از متن استفاده کن. واسه L9 دقت کن اگر خواستی اول و آخر رشته ها مساوی باشد از قاعده SS استفاده کن دقت کن کلا 4 شکل از L9 بدست میاد یا اول و آخر غیر همسان یا اول و آخر همسان. کلا دو تا SS دیدی داری همسان ها را میسازی. تعداد کل رشته های به طول 4 میشود 2^4 همین برای سایر رشته ها به طول N ولی از اونها ما فقط زوج هایی را میخواهیم که طول ab با هم برابر باشد. در l10 میگوییم که تعداد پسوند های b نباید بیشتر از a شود. یعنی نباید بزاریم b بیاد در آخر پس قاعده aSb باید حذف شود تا تو پسوند ها به مشکل ایجاد شود. در l11 دو نکته دارد اول اینکه باید قاعده bSa را حذف کنیم چرا که تعداد پیشوند های b بیشتر از a نشود و نکته دوم این هست که صحبتی از برابر بودن این 2 نشده است پس میتوانیم حتی تعداد a ها بیشتر شود تا پیشوند بودن حفظ شود. دقت کن زبان های جدید l10 , l11 لزوما با l9 برابر نیستند چون همه رشته هایی که طول ab در آنها برابر باشد را تولید نمیکند. و اجتماع این 2 l9 را تولید نمیکند چرا؟ چون اونهایی اول و آخر همسان دارند را هیچکدام تولید نمیکنند. هر جا فرمی به شکل S->SS بود و این S در تولید رشته نقش داشت مبهم میشود. مثلا اگر از یک متغیر به متغیری رسیدیم که قواعد آن نقطه بازگشت به دیگر متغیر ها یا لاندا نداشت مشخص هست که مفید نیست زیرا انگار در لوپ افتادیم و میتوانیم آنرا حذف کنیم. برای هر زبان منظم میتوان یک گرامر غیر مبهم نوشت چرا بخاطر DFA. برای گرامر های غیر مبهم برای تولید رشته تعداد استفاده اشتقاق چپ و راست با هم برابر هست. و طبیعی هم هست چون در مبهم ها با گام های مختلف به تولید رشته یکسان میتوانی برسی. دقت کن در درخت اشتقاق هر عملی که خواستیم اولویت بالاتری بدهیم در ارتفاع پایین تر آن را قرار میدهیم مثلا در G5 در درخت اولی اولویت ضرب بالا تر هست نسبت به جمع، و اگر ابهام بر طرف نشود از لحاظ معنا جملات غلط میشوند. برای 2 تا جمع یا ضرب یکسان هم میتوانیم اولویت متفاوت بدهیم. در صفحه بعدی میخواهیم بگوییم که E فقط میتواند جمع ها را بسازد و به جمع تجزیه شود و همچنین T فقط میتواند ضرب ها را بسازد و عمق کمتری نسبت به E دارد. E+T یعنی شرکت پذیری جمع در سمت چپ باشد پس فقط E میتواند به جمع تجزیه شود اگر همزمان شرکت پذیری راست و چپ باشد جمله از عرف خارج میشود. T+E میشود شرکت پذیری راست. دقت کن وقتی غیر مبهم شود چون فقط یک درخت برای رشته ها میتوان تولید کرد اولویت ها فریز میشوند. دقت کن برای مبهم بودن گرامر فقط کافی هست یک رشته پیدا کنی که 2 تا درخت متفاوت داشته باشد. و برای تبدیل به یک غیر مبهم واسه G6 میایم اول aS اول a\* را میسازیم بعد با gAتضمین میکنیم که یک g میاد بعد a\* را با A میسازیم. مبهم بودن یا نبودن گرامر تصمیم نا پذیر هست یعنی باید با آزمون و خطا چک کنی. مساله ابهام فقط در زبان های مستقل از متن مطرح هست. اگر قاعده S همزمان بازگشتی چپ و راست داشته باشد و S در تولید رشته مفید باشد لزوما مبهم هست. الحاق منظم و خطی، خطی هست. در قسمت ذاتا مبهم چون نمیتوانیم گرامر مستقل از متن غیر مبهم براش بنویسیم ذاتا مبهم میشود. برای زبان بعدی که ABC هست دقت کن که یک زیر مجموعه ای از این زبان که میشود ABC به توان i که هر 3 با هم برابر هستند اصلا زبان مستقل از متن نیست که بخواهیم براش گرامر بنویسیم و زبان وابسته به متن هست پس اگر زبانی مستقل از متن نباشد زبان ذاتا مبهم هست چون نمیتوان براش گرامر غیر مبهم بنویسیم. دقت کن هر موقع دیدی که مثلا چند متغیر بود مثل ABC ولی دیدی باید به تعداد مساوی از AB تولید کنی و توان آن با C فرق دارد میشود مستقل از متن چون باید دو متغیر کنار همدیگر باشند تا بتوان این زبان را تولید کرد. دقت کن همه این 3 زبانی که   
  
مثال زده شد قرار هست به زیر مجموعه زیر برسد:



چون دنبال ساخت رشته های مشترک هستیم پس اگر مستقل از متن نباشد هر گرامر مستقل از متنی براش بنویسیم مبهم میشود پس میشود ذاتا مبهم. وقتی میگوییم غیر مبهم یعنی پیش فرض مستقل از متن بوده اند. هر زبان مستقل از متن یا ذاتا مبهم هست یا غیر مبهم.

برای هر زبان غیر مبهم لزوما یک گرامر مبهم مولد وجود ندارد مثل زبان تهی که اصلا چیزی نیست که بشه گرامر نوشت. پس گرامری که رشته ای تولید نکند میشود غیر مبهم چون تهی هست و تهی یک زبان منظم هست و محدود هست و دقت کن برای غیر مبهم ها بیشمار میتوانی گرامر تولید کنی. برای هر زبان مستقل از متن غیر مبهم نا تهی حتما میشود یک گرامر مبهم نوشت. زبان های غیر مبهم تحت عملگر وارون بسته هستند یعنی وارون هم غیر مبهم هست. وارون هم میدانی که قسمت راست قواعد را وارون میکنی.

در گرامر های ساده دقت شود که هر تعدادی متغیر بخواهیم میتوانیم بیاوریم ولی آوردن ترمینال ها باید محدود و به شکل گفته شده باشد یعنی زوج های (V,a) فقط یکبار دیده شود. و دقت کن تمام قواعد باید با ترمینال شروع شوند و اگر با متغیر شروع شدند گرامر ساده نیستند. هر گرامر ساده لزوما یک گرامر غیر مبهم هست ولی نمیتوانی بگویی هر گرامر غیر مبهم لزوما گرامر ساده هم هست. و از مرتبه o n هست برای اینکه بفهمیم یک رشته جز یک زبان هست از طریق گرامر یا نه. تعداد قواعد گرامر هم به این صورت هست که حداکثر به اندازه v \* t هست چون اگر بیشتر باشد زوج های (V,a) بیشتر از یکبار دیده میشود.

فصل ششم:

جلسه بیستم:

یک متغیر هست اگر از طریق آن به یک رشته یا ترمینال برسیم و یا به ترکیبی از ترمینال و متغیر برسد، و ترتیب اعمال مهم هست یعنی اول باید جنبه اول اعمال شود بعد جنبه دوم. یعنی اول باید ببینیم که چه قواعدی هستند که به وسیله آنها به یک رشته میرسیم یعنی یک لیست میسازیم اول اونایی را میزاریم که مستقیم به یک رشته میرسد مثل لاندا بعد اون قواعدی را میگذاریم که به وسیله آن میتوانیم به یکی از متغیر های جز لیست جنبه اول برسیم و به همین شکل حاصل منهای کل متغیر ها منهای متغیر های مفید از نظر جنبه اول متغیر های نا مفید بدست میاد و باید دقت کنی که این متغیر ها در هر قاعده ای که بودند باید حذف شوند چون ما به رشته نمیرسیم. بعد باید برویم جنبه دوم را بسازیم که باید به شکل یک گراف بسازیم و برای هر متغیر یک گره در نظر میگیریم و ببینیم از یک متغیر با چه قاعده ای به چه متغیری میرسیم و بعد متغیر های مفید میشود اون حالاتی که قابل دسترس هستند و اونایی که قابل دسترس نباشند حذف میشوند چون غیر مفید هستند و بدون عوض شدن زبان گرامر مورد نظر. دقت کن اگر جنبه دوم را اول اعمال کنی بعد جنبه اول را دوباره باید یکبار دیگر جنبه دوم را اعمال کنی اگر به ترتیب اعمال نکنی. پیچیدگی گرامر میشود سمت راست بعلاوه سمت چپ که سمت چپ معمولا همان 1 هست یعنی واسه هر قاعده اینارو بدست بیار و جمع کن دقت کن مقدار لاندا صفر میشود چون انگار هیچی نیست بعلاوه 0 میکنی انگار. دقت کن با حذف گرامر پیچیدگی کم میشود.

**حذف قواعد لاندا**: دقت کن که زبان نباید در صورت حذف تغییر کند. متغیر پوچ شونده یعنی با تعداد یک حرکت یا بیشتر اون متغیر پوچ شود چون به رشته پوچ رسیده است. و عین اون جنبه اول متغیر ها را بررسی میکنیم که پوچ میشوند یا نه دقت کن اونایی هم که به متغیر های پوچ شونده میروند هم جز لیست پوچی ها بگذار. دقت کن اگر سمت راست یک قاعده n تا متغیر پوچ شونده داشتیم اون قاعده حداکثر به 2^n حالت میتواند قاعده داشته باشد. دقت کن در نهایت باید هر جا لاندا بود آنرا حذف کنی. **دقت کن تکراری ها را نباید بنویسی چون مثلا ممکن هست لاندا را حذف کنی قواعد تکراری تولید میشود که نباید لحاظ کنی**.

ببین حالات در حذف لاندا به این صورت هست که متغیر باشد یا نباشد. با حذف قواعد لاندا سرعت افزایش پیدا میکند. در حالات کلی پیچیدگی افزایش پیدا میکند البته ممکن هم هست کم هم شود. دقت کن در حالات میتواند هیچکدام نباشد و لاندا شود. برای معادل بودن زبان حذف شده از حالات لاندا و زبان اولیه باید قاعده s پریم را اضافه کنیم. غیر انقباضی یعنی طول رشته کاهش پیدا نمیکند.

برای حذف قواعد یکه یا واحد نیز دقیقا به شکل جنبه دوم لاندا عمل میکنیم یعنی اینکه به ازای هر متغیر یک گره میکشیم بعد گراف را ترسیم میکنیم و به طول قواعد یکه ما یال جهت دار خواهیم داشت. دقت کن از یک قاعده به خودش بروی معنایی ندارد. و طی یک زنجیر میتوانیم از یک متغیر به متغیر های سایر برویم و با اعمال این قواعد یکه که میتوانیم به متغیر ها جا به جا شویم اون قواعد یکه را حذف میکنیم. دقت کن دیگر با یکه ها کاری نداریم چون اثر آنها را اعمال کردیم چطوری؟ از روی اینها گراف ساختیم از روی گراف زنجیر ها را ساختیم حالا از روی زنجیر ها میایم قواعد جدید را میسازیم و از یک متغیر به هر متغیری که توانستیم برویم قواعد را لحاظ میکنیم. دقت کن اولش باید یک جدول واحد و غیر واحد بکشیم. دقت کن تکراری ها را هم نباید بنویسی. دقت کن باید همه قواعدی که مال متغیر هایی هست که به آن میتوانیم برویم از یک متغیر را باید لحاظ کنیم دقیقا همان کاری که در گرامر ^G شده است. دقت کن گرامر حاصل شده با هم معادل هستند. با حذف قواعد یکه میشود که سرعت افزایش پیدا کند یعنی مثلا از S به n قبلا طی 4 گام بود در گرامر اولیه ولی الان در یک گام میشود. اگر لاندا از قبل باشد تعداد قواعد لاندا افزایش پیدا میکند. اگر قواعد غیر مفید باشد مثل بازگشتی بدون نقطه خروج باعث افزایش قواعد غیر مفید شود. پیچیدگی میتواند افزایش پیدا کند البته در حالت خاص خیلی خاص کاهش هست.

اگر لاندا نباشد یکه اعمال کنیم که هیچی چون اصلا لاندا نیست که بخواهد تولید شود ولی اگر لاندا داشتیم یکه اعمال کنیم تعداد لاندا ها بیشتر میشود پس باید اول باید لاندا اعمال شود بعد یکه چون اگر لاندا رو حذف کنیم درست هست یکه ایجاد شود ولی مشکلی ندارد ولی دقت کن همه این 2 ممکن هست قواعد نا مفید تولید کنند پس نا مفید رو میگذاریم آخر تا اضافی ها حذف شوند دقت کن اگر به ترتیب اعمال نکنی دوباره مجبوری اعمال کنی تا درست شود. گرامر اگر مبهم باشد لزوما با این ساده سازی ها رفع ابهام نمیشود. دقت کن با رفع ابهام ها کمینه میشود در واقع ساده سازی میشود ولی رفع ابهام نشده است. **حذف بازگشتی چپ**: اول ببین نقطه بازگشت کجاست مثلا در گرامر مثال اول نقطه بازگشت d هست خوب اوکی تک d رو تولید کن حالا اگر میخواهی b بزنی یک متغیر جدید به نام A` تعریف کن و لوپ b را روی آن بزن دقت هم کن تک b هم باشد واسه تولید تکی و در نهایت بازگشتی چپ به راست تبدیل میشود و درخت از راست رشد میکند دقت هم کن اگر در گرامر اولی بازگشتی داشتیم در گرامر معادل هم بازگشتی داریم و خاصیت نامتناهی را حفظ میکند. دقت کن اون ترمینالی که با متغیر میاد و شکل بازگشتی تشکیل میده بستار ستاره ای دارد و لوپ میخورد. دقت کن در محاسباتی ببین اون نقطه بازگشت در چه شرایطی آمده است مثلا در مثال با +و - آمده است خوب همونطوری که بازگشتی چپ حذف میکردی بنویس متغیر پریم جدید تعریف کن بعد تو متغیر پریم حالات مثبت و منفی اون نقطه بازگشت را بساز همین برای بقیه قسمت ها. دقت کن باید به بازگشتی راست تبدیل کنی. در تبدیل بازگشتی چپ غیر مستقیم باید به ازای اون متغیری که قرار هست حذف کنی تمام قواعدش را یکبار لحاظ کنی و حاصل میشود چپ گردی مستقیم بعد باید چپ گردی مستقیم را حذف کنیم. دقت کن در بازگشتی چپ با متغیر پریم اون قواعدی را میسازیم که بازگشتی چپ داشتند.

**جلسه بیست و یکم:**

شکل های نرمال چامسکی و گریباخ، در چامسکی که مستقل از متن هست در سمت راست یا 2 متغیر دارد یا یک ترمینال یعنی لاندا و نا مفید و یکه نباید داشته باشد پس اول نرمال میکنیم اون 3 تا رو میزنیم بعد به چامسکی تبدیل میکنیم. برای تبدیل کردن به چامسکی هم اونایی که دو لیترال هستند رو بگیر یک متغیر بعد دقت هم بکن که اونایی که 3 متغیر شدند در این حالت یکی رو تنها بگیر 2 تای باقی مونده را یک زوج بگیر به یک متغیر اتلاق کن بعد خود اون متغیر قاعده ساخت 2 تای باقی مانده را داشته باشد تا بتوانی چامسکی را بسازی. اگر یک گرامر لاندا و یکه نداشته باشد حتما یک فرم نرمال چامسکی دارد بقیه نکته در جزوه هست. درخت اشتقاق به شکل درخت دودویی هست واسه فرم چامسکی هست و واسه هر رشته که توسط گرامر چامسکی تولید میشود تعداد گام یا مراحل اشتقاق برای تولید رشته 2n-1 هست که n طول رشته مورد نظر هست. و طی n-1 مرحله به رشته ای میرسیم که تماما متغیر هست بعد باید به فرم چامسکی برود که این هم n گام هزینه دارد که تماما ترمینال شود. در چامسکی ابهام روی تعداد گام ها یا مراحل اشتقاق تاثیری ندارد و به طول 2n-1 هست ولی این واسه بقیه صادق نیست چون تعداد گام ها اگر مبهم باشد و چامسکی نباشد متفاوت هست برای تولید رشته یکسان، اگر گرامر مبهم نباشد تعداد گام ها برای تولید رشته یکسان در اشتقاق چپ و راست برابر هست فقط همین. ارتفاع درخت هم مشخص هست یا تک فرزند هست که برگ ترمینال هست یا 2 فرزند هست که فرزندان متغیر هستند و برگ ها برابر هست با تعداد گره ها در سطح ما قبل آخر واسه همین هست لگاریتم + 1 بخاطر سطح آخر بیشترین ارتفاع میشود درخت به شکل اریب و کمترین میشود به شکل پر درخت. برای طول رشته به فرم چامسکی با ارتفاع L حداقل ارتفاع L هست اون هم وقتی که درخت به شکل اریب باشد و حداکثر ارتفاع این هست که تا سطح ما قبل آخر پر باشد و بیشترین تعداد فرزند را داشته باشد با ارتفاع L و میدانیم تعداد سطح آخر با ما قبل آخر یکی هست و تعداد گره در سطح ما قبل آخر برابر هست با 2^L-1 پس حداکثر طول رشته هم همین مقدار میشود. برای هر زبان مستقل از متن ما یک الگوریتم عضویت داریم که بررسی میکند به زبان تعلق دارد یا نه با مرتبه O n^3 که همان مثل برنامه نویسی پویا و ضرب زنجیره ای ماتریس ها هست و به نام CYK هست. دقت کن در جدول پویا که میکشیم قطر اصلی مال رشته های به طول یک هست و قطر بعدی به طول دو همین تا آخر، و دقت کن میگوییم با چه قاعده هایی میتوانیم که اون زیر رشته را تولید کنیم. برای طول های دو بیشتر هم باید بشکنیم ببینیم جواب میدهد یا نه مثلا از 1 تا 3 یکی 11 و یکی 23 که البته جواب نمیدهد اینجا. دقت کن اون رشته ها جایگاه اونها در رشته اصلی هستند و طبق اونا باید بنویسی مثلا از 2 به 4 باید از 22 و بعد 34 بزنی. دقت کن که برای پیدا کردن جواب هر کدام از آن خانه ها از مرتبه O n زمان میبرد چرا که مسئله بازگشتی هست و باید شکست های مختلف را امتحان کنیم اون n^2 اولی هم که هست بخاطر حاصل جمع خانه های از 1 تا n هست. ابهام تاثیری روی زمان الگوریتم عضویت ندارد. شکل نرمال گریباخ: دقت کن که ما هنوز در دسته زبان های مستقل از متن هستیم. هر گرامر ساده یک گرامر گریباخ هست ولی عکس آن درست نیست. در اینجا هم قواعد لاندا و یکه و نا مفید هم داریم و اینها حتما باید حذف شوند تا بتوانیم به گریباخ تبدیل کنیم و در مرحله 4 ام باید بازگشتی چپ را نیز حذف کنیم. یک گام پنجم هم داریم آن هم این هست که اگر بعد از 4 مرحله بالا هنوز قواعدی بود که با متغیر شروع میشدند و به شکل گریباخ نبودند شما بیا با عمل جایگزینی A با قاعده اول خودش این مشکل را حل کن حتی اگر خود قاعده A که میخواهیم جایگزین کنیم باز با متغیر شروع شده بود بیا اول با جایگزینی آن را حل کن بعد برو سراغ قاعده اولیه که مشکل داشت. به مثال اول دقت کن جمله آخر که نقطه بازگشت یکی میاد و بعد از اون اونجایی که بازگشتی داریم میتواند \* بار بیاید. برای چپ گردی باید به راست گردی تبدیل کنیم. برای هر گرامر مستقل از متن که لاندا جز آن نباشد یک فرم گریباخ داریم. چرا عکس گرامر ساده درست نیست چونکه در گریباخ میتوانیم دو قاعده با شروع ترمینال داشته باشیم ولی در گرامر ساده نمیشود. به اندازه طول رشته با مراحل اشتقاق به رشته میرسیم یعنی اگر طول رشته 4 هست به ازای 4 گام به رشته مورد نظر میرسیم که عضو زبان باشد. دقت کن فرم گریباخ A->aB ما میدانیم که B که یک متغیر هست به تعداد استار بار میتواند بیاید.

**فصل هفتم:**

**جلسه بیست و سوم:**

بحث این جلسه روی ماشین های پشته یا PDA ها که پذیرنده زبان های مستقل از متن هستند. ماشین های پشته ای قبلی حافظه محدود داشتند که از مرتبه O 1 بود و به اندازه ثابت حافظه نیاز داشتیم تا بتوانیم براشون پذیرنده طراحی کنیم که میشدند زبان های منظم. این ماشین شبیه همان ماشین قبلی هست فقط حافظه نامحدود بهش اضافه شده است. در واقع الان یک ابزار توصیف در کنار گرامر داریم برای زبان های مستقل از متن که همان ماشین پشته ای هست. دقت کن که الفبای رشته ورودی با الفبای پشته اصلا میتوانند اشتراکی نداشته باشند. Z علامت خالی بودن پشته هست که به بالای پشته هم میتواند اشاره کند. و در تابع گذار دقت کن ما به عنصر بالای پشته هم کار داریم و اینکه در چه حالتی هستیم و با خواندن چه کاراکتری به کدام حالت میرویم و میگوییم حالت چه با خواندن کاراکتر چه بدون خواندن کاراکتر با صرف چه رشته ای از بالای پشته به کدام حالتی میرود و بالای پشته چه خواهد شد. فعلا صحبت روی ماشین پشته ای غیر قطعی یا نامعین هست که NPOA نام دارد بعدا با DPOA هم آشنا میشویم که مدل قطعی هست. دقت کن که رشته را از راست به چپ طبق قرار داد پوش میکنیم بعد از اینکه عنصر سر رشته را پاپ کردیم و مصرف کردیم. لاندا تاثیری ندارد. اگر الفبایی که به آن حالات میرویم لاندا باشد از طول پشته کم میشود چون فقط عنصر بالای پشته پاپ میشود و چیزی پوش نمیشود به عنوان الفبای ورودی. و به واسطه عدم قطعیت به حالات مختلف میرویم. دو حالت پذیرش داریم توسط حالت نهایی و توسط پشته خالی که در پشته خالی هم باید رشته مصرف شده باشد هم پشته خالی باشد حالا در هر حالتی باشد و اصلا حالت نهایی نداریم و منظور توسط پذیرش توسط پشته بوده است. پذیرش به این 2 حالت با هم معادل هستند. وقتی لاندا میگذاریم همان معنی پاپ کردن و چیزی قرار ندادن را میدهد. دقت کن واسه ماشین وقتی حرکت لاندا میزاری که مستقیم به حالت نهایی برسد باید دقت کنی که رشته باید مصرف شود تا به حالت نهایی برسیم پس این غلط میشود و بهتر هست از علامت $ که آخر رشته ها میگذارند بگذاریم که مشخص کنیم که منظورمان این هست به آخر رشته رسیدیم یا حتی رشته پوچ بود میفهمیم که واقعا پوچ بوده است اگر به جای عنصر بالای پشته، یا لاندا بگذاریم یعنی اینکه میخواهیم پشته هم پوچ و خالی باشد و میشد پذیرش رشته توسط پشته خالی. ماشین اولی دارای عدم قطعیت هست. در رشته اولی پذیرش توسط حالات نهایی بوده است. عنصر بالای پشته در واقع صرف میشود. دقت کن رفتار خطی دارد پیچیدگی حافظه ای و ممکن هست کمتر از n هم شود مثلا برای هر 2 a یک a پوش کنیم که خلاصه حداکثر میشود O N. دقت کن اولین چیزی که در قواعد انتقال میاد همون را مصرف میکنیم یعنی وقتی میگیم a,x/z یعنی با صرف a و بالای پشته چی به کجا میرویم. دقت کن ماشین پشته ای بدون گذار لاندا هم میتواند رشته پوچ را بپذیرد یعنی حالت اولیه نهایی باشد پس اگر گفت هموار با گذار لاندا غلط هست. دقت کن عدم قطعیت مثل قبلی نیست تنها در صورتی ماشین پشته ای عدم قطعیت دارد که 1. با صرف دو کاراکتر متمایز و حرف بالای پشته یکسان یکی به حالت نهایی و دیگری به حالت دیگر و یا با صرف دو کاراکتر متمایز و بالای پشته ی یکسان به دو حالت متفاوت با دو عنصر بالای پشته متفاوت برویم. دقت کن در ماشین دوم وقتی میگیم به جای z لاندا بگذار با این کار جلوی پذیرش یک سری از رشته هایی که درست نیست را میگیریم اگر بالای پشته z بماند یک سری از رشته ها که درست نیست را میپذیرد و تنها در صورت مصرف شدن تمام رشته میپذیرد. الفبای پشته با الفبای ورودی میتواند متمایز باشد. در l2 میایم میگیم یک راه حل این هست که با خواندن یک a چهار تا a پوش کن. دقت کن وقتی میگیم لاندا یعنی pop. با افزایش الفبا پشته قدرت بیشتر میشود مثل راه حل سوم. برای l5 میایم حالت آخر را میگیریم که همه a ها و حتی z هم پاپ شود و پشته خالی میشود و ماشین از کار میفتد و پذیرش از طریق پشته خالی خواهد بود.

در زبان l6 دقت کن که وقتی به 2 کاراکتر یکسان میرسی عدم قطعیت دارد چرا چون یا هنوز داخل w هستیم یا معکوس رشته شروع شده است. برای l6 نمیتوانیم ماشین قطعی تعریف کنیم چون وسط رشته جایی که معکوس با اصل رشته جدا میشود نیاز به عدم قطعیت دارد. رده های زبان های خطی دو ابزار داریم که بررسی کنیم هستند یا نیستند یک گرامر خطی هست دو ماشین های پشته ای یک نوبته هست.

ماشین پشته ای قطعی: قانون دوم میگوید که ماشین غیر قطعی یا همان NPDA هست یعنی اگر به این شکل باشد. دقت کن چه با صرف کاراکتر یکسان چه بدون صرف کاراکتر یکسان با الفبای پشته یکسان نباید به دو حالت مختلف برویم اگر بخواهیم که ماشین قطعی باشد. زبان های مستقل از متن قطعی پذیرنده مخصوص به خودشان را دارند که همان ماشین پشته ای قطعی هست زبان های خطی هم ماشین پشته ای مخصوص به خود را دارند که غیر قطعی یک نوبته هست. برای یک نوبته به این فکر کن که بعد از اولین پاپ باز هم نیاز به پوش کردن هست یا نیست اگر نبود یعنی تک نوبته هست. در همین مثال ها L3 زبان مستقل از متن نیست چرا ؟ چون با شروع کاراکتر های b ما دیگر حافظه ای نداریم که مقادیر n را در خاطر داشته باشیم و اون مقدار n صرف شده هست پس زبان های مستقل از متن قدرت تولید همچین رشته هایی را ندارند. مستقل از متن بودن قطعی نسبت به متمم بسته هست. تو برگشت اگر لاندا بگذاریم یعنی نمیخواهیم چیز جدیدی ماشین بپذیرد اگر z بگذاریم یعنی میخواهیم چیز های دیگری بپذیرد مثل m7 ماشین دوم و برای زبان l7 نمیتوانیم ماشین تک نوبته تولید کنیم. اگر تک نوبت بتوانی طراحی کنی صد در صد قطعی هست. زبان l8 هست چرا غیر قطعی هست ؟ چون بعد از خواندن a ها ناچار هستی به دو راه بری یک اینکه به b^n ها بروی یا اینکه به b^2n ها بروی چون دقیقا نمیدانی کجا هستی و چه قدر باید پاپ کنی یعنی یکی یک a پاپ میکند یکی 2 تا پاپ میکند. زبان های مستقل از متن قطعی تحت اجتماع بسته نیستند. منظم محدود باشد خطی هم هست. زبان های خطی تحت اجتماع بسته هستند. زبان های مستقل از متن قطعی اجتماع با منظم قطعا بسته هستند و مستقل از متن قطعی میماند. زبان l9 قطعی است دقت کن در قسمت ماشین اون خط ها که انشعاب پیدا کرده اند بخاطر این هست که اگر a ها پشت سر هم بیاد. دقت کن تا قبل a یازدهم برای 10 تا حالت حساب کردیم ولی از اون به بعد فرم a^nb^n را اجرا میکنیم. دقت کن اگر مجبور باشی در طراحی گرامر به 2 متغیر در یک قاعده بروی دیگر خطی نیست و مستقل از متن غیر خطی هست. عدم قطعیت دارد چرا چون وقتی a ها را خواندی ممکن است b پوچ باشد و یهو بروی سراغ m ها پس نیاز به عدم قطعیت داریم حتما. در زمان l11 هم عدم قطعیت وجود دارد. اما در l12 بر خلاف دو تای قبلی دقیقا میدانیم وقتی a میاد کجا هستیم در قبلی ها نمیدانستیم در b اولی هستیم یا رسیدیم به دومی ها و به 2 مسیر میرسیدیم ولی اینجا نیازی به عدم قطعیت نداریم. غیر خطی را از یک نوبته بودن هم میتوانی برسی. در l13 هم نیازی به عدم قطعیت نیست چون مشخص هست و مانند قبلی ها نیست که ندانی کجا هستیم و اگر b باشد مسیر خودش و d باشد هم مسیر خودش. الحاق خطی با خطی، خطی نیست. در l14 هم نیازی به عدم قطعیت نداری چون بعد از خالی شدن پشته توسط a^nb^n میتوانی سراغ b ها بروی و پوش کنی به این شرط که بالای پشته z باشد یعنی پشته کامل مصرف شده باشد. در l15 دقت کن که کوچکترین رشته اولی که لاندا هست ولی در دومی کوچکترین رشته d هست پس باید یک خط مستقیم از حالت شروع بکشی تا تک d هم بپذیرد. دقت کن به قوانین مشابه هم دقت کن تا عدم قطعیت نباشد. هر DPDA یک DFA هست به طوری که فقط به پشته دست نزنی و بگی z هست. وقتی ارتفاع پشته هیچ موقع کم نمیشود یعنی اصلا پاپ نمیکنی و همواره در حال افزایش باشد یعنی اصلا به پشته نیازی نداشتی واسه همین منظم هست ولی در زمان مستقل از متن ارتفاع اگر افزایش پیدا کند به همان مقدار کاهش هم پیدا میکند.

**جلسه بیست و چهارم:**

برای تبدیل پذیرش توسط پشته خالی به پذیرش توسط حالات نهایی از همه حالات یک حرکت لاندا به یک حالت نهایی جدید اضافه میکنی. حالا برای تبدیل پذیرش توسط حالت نهایی به پذیرش توسط پشته خالی باید با صرف لاندا و با هر چیزی که بالای پشته بود به یک حالت غیر نهایی برویم دقت کن که تعداد قواعد any در آنجا به اندازه الفبای پشته هست تازه باید داخل اون حالت جدید غیر نهایی هم لوپ بزنی تا پشته کاملا خالی شود و حتی z هم پاپ شود. پس پذیرش غیر قطعی در حالات نهایی با پشته خالی برابر هست برای قطعی هم همینطور هست ولی از طریق پذیرش توسط حالت نهایی قطعی یک معادل توسط پشته خالی وجود ندارد تنها در صورتی معادل وجود دارد که prefix free باشد یعنی هیچ رشته ای از زبان پیشوند رشته ای دیگری از زبان نباشد اونوقت میتوان به DPDA پذیرش توسط حالت نهایی را به DPDA پذیرش توسط رشته خالی تبدیل کرد. اگر گفتیم که پذیرش توسط DPDA وجود دارد منظور پذیرش توسط حالات نهایی هست. برای هر زبان مستقل از متن ما یک PDA داریم و این لزوما قطعی نیست و فقط برای مستقل از متن های قطعی DPDA داریم و وقتی میگوییم PDA و چیز دیگری نمیگوییم منظورمان پذیرش توسط حالت نهایی هست چون پذیرش توسط پشته خالی فقط زیر مجموعه ای محضی از زبان های مستقل از متن را حاصل میشود اون زیر مجموعه چی هست؟ prefix free ها هستند. دقت کن زبان های منظم زیر مجموعه محض زبان های قطعی مستقل از متن هستند. یعنی هر زبان منظمی قطعی هست ولی هر قطعی منظم نیست. خطی وجود دارد که قطعی هست و قطعی نیست. زبان های مستقل از متن به 2 بخش ذاتا مبهم و غیر مبهم هست. مدل محاسبه کل زبان های مستقل از متن PDA هست که این ماشین پشته ای قطعی نیست چون فقط واسه مستقل از متن های قطعی ما ماشین قطعی داریم دقت کن که هر DPDA میتواند NPDA باشد ولی برعکس نه. برای خطی هم که میشود همین PDA یک نوبته طراحی کرد. حالا غیر قطعی ها کجا هستند؟ ذاتا مبهم ها با غیر مبهم هایی که قطعی نیستند. پس مستقل از متن یا قطعی هست یا غیر قطعی. هر ذاتا مبهمی قطعا غیر قطعی هست. دقت کن در قضیه بعدی اگر زبان لاندا داشت شما بیا قاعده لاندا را هم اضافه کن تا لاندا را بپذیرد. در تبدیل گرامر گریباخ به NPDA میایم در اولین مرحله با یک حرکت به Q1 میرویم تا شروع گرامر باشد و S بتواند تجزیه شود واسه همین S را بالای پشته میگذاریم. در مرحله بعدی میگوییم که یک سری رشته X را به قوانین انتقال اضافه کن در مرحله سوم هم میگوییم هم با لاندا برو به حالت نهایی هم بالای پشته Z باشد که هر 2 پذیرش را داشته باشیم. دقت کن در تبدیل اگر به ترمینال رسیدی چون چیزی بعدش نیست میگی لاندا بگذار جاش در کجا ؟ در بالای پشته یعنی به نوعی اون متغیر بالای پشته پاپ میشود. برای هر زبان مستقل از متن یک NPDA با 3 حالت وجود دارد چطوری پذیرش با حالت نهایی. دقت کن با پشته خالی منظورمان نیست چون اگر بود در جا باید میگفتیم. با مصرف لاندا و اگر بالای پشته Z باشد میگیم S بیاد اینجوری فقط یکبار S میاد اونم همون اول چون در غیر اینصورت مرتب S میومد. وقتی میگیم PDA منظور NPDA هست. و هر DPDA یک NPDA هست ولی برعکس نه. دقت کن سیپسر را میتوان به لینز تبدیل کرد چطوری مثلا اگر گفتند یکی 1 پاپ کن 3 تا صفر پوش کن خوب کاری ندارد تو بیا از گذار لاندا استفاده کن. دقت کن خروجی تابع انتقال میشود حالت و عنصری که بالای پشته پوش میشود.

ماشین پشته ای تقویت شده: در استاندارد یک پشته داشتیم اما اینجا تعداد پشته ها را زیاد میکنیم. فرقش این هست که دو تا گاما داریم چرا چون بالای دو تا پشته را میخواهیم بدانیم چیست و برای 2 تا هم باید بگوییم پوش یا پاپ کن. قدرتش از استاندارد بیشتر هست و نا مستقل از متن هست. دقت کن از دو پشته به بعد هر چه قدر تعداد پشته زیاد شود تاثیری ندارد و با 2 پشته برابر هست. و دو پشته ای که ماشین تورینگ هست قوی ترین مدل محاسبه هستش.

**جلسه بیست و ششم:**

میخواهیم بدانیم نسبت به چه عملگرهایی زبان های مستقل از متن بسته هستند یا نیستند. سیگما استار منظم هست منظم هم تحت منها منظم هست. دقت کن که مستقل از متن ها تحت اشتراک منظم و تفاضل منظم و الحاق منظم بسته هستند. برای الحاق منظم اگر l1 خطی و l2 منظم باشد حاصل میشود خطی. برای اشتراک هم یکی منظم که DFA هست و خطی هم ماشین پشته ای یک نوبته هست چون تحت اشتراک بسته هستند تحت تفاضل هم بسته هست چون اشتراک میشود L1-!L2 (تعجب منظور نات مکمل هست) که چون اونجا L2 منظم هست و تحت مکمل گیری بسته هست میشود همان اشتراک پس تحت اجتماع هم بسته هستند. دقت کن اشتراک منظم یعنی یک طرف منظم باشد مستقل از متن ها تحت اشتراک معمولی بسته نیستند. مستقل از متن ها تحت مکمل بسته هستند. محدود ها و منظم ها زیر مجموعه زبان های مستقل از متن قطعی هستند. ولی زبان های خطی با مستقل از متن قطعی همپوشانی دارند و زیر مجموعه همدیگر نیستند. یعنی خطی داریم که قطعی نیست و قطعی داریم که خطی نیست. ذاتا مبهم ها زیر مجموعه محض غیر قطعی ها هستند هر ذاتا مبهمی لزوما غیر قطعی است. ولی هر غیر قطعی ذاتا مبهم نیست. خطی ها هم با ذاتا مبهم ها همپوشانی دارند. زبان های مستقل از متن قطعی تحت مکمل بسته هستند ولی کل مستقل از متن ها نسبت به مکمل بسته نیستند پس اگر یک زبانی خودش غیر قطعی بود مکمل آن هم غیر قطعی هست. خطی تحت اجتماع بسته هست. لم تزریق زبان های مستقل از متن: دقت کن به 5 زیر رشته میتوان تقسیم کرد با نقیض میگوییم که زبان مستقل از متن نیست. در لم تزریق زبان های منظم ثابت تزریق تعداد حالات بود و رشته را به DFA میدادیم و یک حالت تکرار میشد ولی اینجا ثابت تزریق تعداد متغیر ها هست یعنی یک متغیر در عمل اشتقاق مرتب تکرار میشود مثلا در مثال اول اون A هست. دقت شود که از طریق لم تزریق مستقل از متن ها میتوانیم اثبات کنیم که زبان نا منظم بودن یک زبان و غیر خطی بودن یک زبان هست. لم تزریق زبان های خطی: مثل مستقل از متن ها میتوان 5 زیر رشته تقسیم کرد. دقت کن از نقیض لم تزریق پی میبریم که زبان خطی هست. دقت کن در مستقل از متن ها فقط یکبار میتوانی اسکن کنی دیگر نمیتوانی برگردی مثلا دوباره n را بدست بیاوری برای زبان l1. دقت کن مستقل از متن فقط یک پشته دارد. در l3 یکبار a را به خاطر میسپاری یا یکبار a هارا در پشته نمیریزی چون میخواهی با b مقایسه کنی. اگر زبان L روی الفبای تک نمادی نامنظم باشد لزوما نا مستقل از متن است. پس هر زبان مستقل از متن روی الفبای تک نمادی لزوما منظم بوده است. دقت کن تا L8 همه زبان های وابسته به متن هستند. در L11 دقت کن بعد از اسکن شدن b باید a را در خاطر داشته باشی پس منظم نیست حتی اگر بزرگتر مساوی صفر باشد هم در حالت خاصی هست و مستقل از متن نیست و حساس به متن هست. اگر گرامر مستقل از متن داری که زبان آن روی الفبا تک نمادی باشد یعنی لزوما منظم بوده است آن زبان. در l13 فرض کن که ما a^nb^n را در پشته پوش کرده ایم و با عدم قطعیت حدس زدیم که w اول را وارد کرده ایم و فقط دومی مانده است تو باید a بخوانی ولی بالای پشته b هست و تو به پایین پشته دسترسی نداری پس PDA برایش نداری در نتیجه مستقل از متن نیست. دقت کن زبانی حساس به متن هست که بدون لاندا باشد. دقت کن از روی ماشین پشته ای یا گرامر یا فرم نرمال چامسکی میتوانیم بفهمیم زبان متناهی هست یا نیست. اگر قواعد نا مفید را حذف کردیم ولی بازم متغیر تکراری داشتیم میتوانیم بگوییم زبان مورد نظر نا متناهی هست. اگر بعد از حذف متغیر تکرار شونده نداشت زبان متناهی بوده است چطوری از اون گرامر اصلاح شده به یک گراف وابستگی میرسیم. در گراف مدل سازی هم که میدانی به تعداد متغیر ها گره داری، یال ها هم میشود اینکه توی قواعد از کدام متغیر به کدام متغیر میرویم و اگر دور داشتیم زبان مورد نظر متناهی هست. دقت کن متغیر تکرار شونده باید مفید باشد. مسئله سوم اگر PDA بود به گرامر تبدیل کن و در گرامر دنبال متغیر های پوچ شونده بگرد تمام متغیر های پوچ شونده را پیدا کن اگر لاندا جز آنها باشد پس تولید میکند. مساله چهارم: الگوریتم عضویت داریم بدون لاندا داریم مثل چامسکی.

مساله پنجم: از گرامر چامسکی مولد زبان L استفاده میکنیم اگر W عضو آن باشد تعداد مراحل اشتقاق برابر با 2n-1 بود پس بیایم تمام اشتقاق ها را لیست کنیم و اگر تعداد گام از 2n-1 بیشتر شد فورا هرس میکنیم چون میدانیم این مسیر رشته به طول n را نمیدهد و تعداد هم محدود هست چون n ثابت هست. تعداد مراحل اشتقاق برای گریباخ برای تولید یک رشته n حرفی n-1 تا است.

**فصل نهم:**

**جلسه بیست و هشتم:**

دقت کن در رشته ورودی بلنک قرار ندارد، بلنک نقش جدا کننده رشته و ابتدا و انتهای رشته را مشخص میکند. دقت کن الفبای نوار بخشی از الفبای رشته ورودی هست و رشته ورودی روی نوار قرار میگیرد و یک سری الفبای بیشتری برای نوار نیاز داریم. در مدل های قبلی head فقط میخواند ولی الان میتواند بنویسد. هد به میزان دلخواه به سمت راست یا چپ حرکت کند و رشته ورودی روی نوار هست و روی هر قسمت از نوار میتوانیم بنویسیم و میتوانیم جلو برویم و بعد میتوانیم بریم عقب یعنی بنویسیم بعد برگردیم عقب پس نوار حافظه نامحدود دارد چون میتوانیم بنویسیم و برگردیم و استفاده کنیم. این مدل قوی ترین مدل هست و قوی تر از این وجود ندارد. خروجی هم روی نوار هست و حافظه دیگری نداریم. گاما به هد اشاره میکند و حالت 1 با خواندن گاما به چپ یا راست میتواند برود تازه میتواند همانجا هم بنویسد در همان سلول. پس 2 تا مولفه داریم اینکه در چه حالتی هستیم و هد به چه الفبایی اشاره میکند. دقت کن همانجایی که هد اشاره میکند را میتوانیم بنویسیم یا دست نزنیم بعد به سمت راست یا چپ حرکت کنیم. اگر به جای الفبای هد همان الفبا را در خروجی بنویسی یعنی فقط خواندی و چیزی ننوشتی. برای توصیف اول زیر رشته سمت چپ را بنویس الحاق کن با حالت جاری یعنی الفبا رشته دیگر نیست و حالت هست و بعد از آن رشته ای میاد که q دارد به آن اشاره میکند و الحاق کن با زیر رشته سمت راست این میشود توصیف لحظه ای. بلنک همان معنی پوچ را میدهد. در نهایت یا halt میکند و بگوید yes or no یا تو لوپ بینهایت بیفتد و چیزی خروجی ندهد. محاسبه: طی چند گام از پیکربندی اولیه به یک پیکربندی برسیم و متوقف شویم در صورتی هست که اون قانون انتقال در اون پیکربندی تعریف نشده باشد. دقت کن برخلاف مدل های قبلی وقتی به حالت نهایی برسیم دیگر نمیتوانیم به حالت دیگری برویم ولی مثلا در سایر حالات برای مصرف رشته به حالات دیگر میرفتیم. دقت هم کن سیگما استار میگذاریم تا بتوانیم لاندا یا پوچ را هم بپذیرد. در تورینگ مصرف شدن معنی ندارد فقط همینکه به حالت نهایی رسید کافی هست میخواهد رشته مصرف شده باشد یا نشده باشد. دقت کن وقتی به حالت نهایی رسیدیم ماشین کاملا halt میشود. کلا مصرف شدن رشته معنی ندارد. دقت کن مثلا در مثال اول حتی اگر بعد از آن bb باشد اهمیتی ندارد چون زیر رشته aa دیده شده است و حق نداریم قانونی تعریف کنیم و توصیف آن هم قبل از bb متوقف میشود. ماشین تورینگ میتواند زبان های تورینگ را توصیف کند و به چیز دیگری نیاز نیست. دقت کن وقتی فرم شکل ها به صورت b\*a باشد حتما وقتی a دیده شد بعد از آن ما یک قانون انتقال بلنک برای حالت نهایی داریم و مهم نیست اول یا آخر رشته باشد چون به حالت نهایی رسیده ایم. برای مثال سوم a^nb^n تو بیا اولین a را که دیدی بخوان جاش x بنویس بعد از a ها بگذر به اولین b که رسیدی بخوان جاش y بگذار بعد بیا تا به چپ ترین x برسی رسیدی a را بخوان و جاش x بگذار بعد دوباره از a ها بگذر بعد از y ها هم بگذر تا به آخرین y برسی بخوان بعد y بگذار. بعد در آخر وقتی رسیدی دیگر به a نمیرسی بعد به y میرسی و تغییر حالت میدهی و کلا باید y ببینی تا به بلنک برسی یعنی به انتها رسیدی و متوقف میشوی در همین مثال قواعد به ترتیب هست. دقت کن در حالت نهایی چپ و راست بودنش مهم نیست. دقت کن وقتی قاعده ای تعریف نشده باشد واسه اون حالت ماشین متوقف میشود. ماشین ما از مرتبه n^2 هست چون هر کدام به تعداد n تا به راست و چپ نیاز داشتند. دقت کن این تایم تابعی از ماشین ما هست اگر ماشین ما متوقف نشود و در لوپ بیفتد که اصلا معنی ندارد.

بهترین مرتبه زمانی n logn هست. دقت کن هر ماشین تورینگ را میتوانی به ماشین تورینگی تبدیل کنی که 2 حالت accept and reject دارا هست. دقت کن چپ و راست مهم نیست مهم این هست که داخل حالت reject یا نهایی بیفتد. دقت شود در q2 نیاز نیست قاعده ای اضافه شود چون صرفا داری مسیر را برمیگردی و اگر لازم بود به بلنک بر بخوری در q1 تیک آنرا زده بودی. واسه q3 هم میتوان گفت مثل a3b4 یا a4b3. مقصد میتواند reject or accept باشد ولی مبدا نمیتواند زیرا از این 2 یالی نباید خارج شود. این ماشین تورینگ پذیرنده بود برای هر زبان فرمال سوری میتوان ماشین تورینگ پذیرنده طراحی کرد. واسه مثال تمرین هم مرتبه زمانی n^2 هست چون باید 2n راست و 2n چپ بری و n تا هم a داریم. دقت کن این 2 حالت، حالت توقف هستند accept and reject.

**جلسه بیست و نهم:**

پیکربندی اولیه: q0 and w. چطوری میتوانیم بفهمیم افتادیم در لوپ بینهایت؟ وقتی به پیکربندی تکراری برویم و مشخص هست که دیگر ماشین متوقف نمیشود. دقت شود که ماشین تورینگ یا محاسبه گر هست و یک تابع f را حساب میکند یا پذیرنده هست که رشته را میپذیرد یا نمیپذیرد. دقت شود که ماشین باید در محاسبه گر ورودی صحیح داشته باشد و جدا کننده باید مشخص باشد بعد جمع انجام شود. دقت کن در محاسبه پذیری هد باید روی اولین کاراکتر رشته خروجی باشد. کپی کردن هم با ماشین محاسبه گر تورینگ هست که میاد یک کپی از رشته را در کنار رشته اصلی قرار میدهد ویژگی مهم این الگوریتم ها این هست که خاتمه پذیر هست حتما و روی حالت نهایی متوقف میشود چگونه؟ از سمت راست برو تا به اولین ضربدر برسی دقت کن قبلش کل رشته را ضربدر زدیم یکبار ضربدر را بردار جاش 1 بنویس بعد برو به سمت راست تا به بلنک برسی رسیدی یک 1 بگذار بعد دوباره همینکار را تکرار کن در واقع بلنک را باید 1 بکنی اگر به سمت چپ حرکت کردی به دنبال ضربدر ولی به بلنک رسیدی یعنی الگوریتم اتمام شده است. از کجا میفهمی همه 1 ها ضربدر شده اند؟ به بلنک برسیم. دقت کن در آخر باید به راست حرکت کنی تا روی اولین کاراکتر رشته متوقف شود. دقت کن پیچیدگی زمانی این ماشین محاسبه گر الگوریتم از مرتبه N^2 هست دقت کن ماشین تورینگ وقتی الگوریتم هست که روی هر ورودی متوقف شود اگر توی لوپ بینهایت بیفتد که دیگر الگوریتم نیست. دقت کن در محاسبه گر ها میخواهیم که هد روی اولین کاراکتر همیشه باشد. خط تیره برای این هست که تعداد قوانین کم شود ممکن هست به جای 0 و 1 بگوید هر الفبایی. برای جمع هم که اگر صفر داشتی باید از چپ حرکت کنی به سمت راست روی کم ارزش ترین متوقف شوی آنرا 1 کنی و برگردی سمت چپ. اگر کم ارزش ترین بیت 1 هست 2 حالت میشود اگر بعد از آن 1 صفر هم در راه بود آنرا 1 کن و به بقیه دست نزن اگر هیچ صفری نبود و به بلنک رسیدی بلنک را 1 کن و روی اولین کاراکتر رشته متوقف شو. مکمل 2: مکمل 1 را با جمع بعلاوه 1 ادغام کن یعنی اونجایی که همه رو مکمل 1 کرد و روی اولین کاراکتر متوقف شد اون حالت بشود حالت اولیه ماشین بعلاوه 1. راه حل دوم از راست حرکت کن یا از چپ حرکت کن تا به کم ارزش ترین بیت برسی بعد به صفر ها دست نزن به اولین 1 هم رسیدی دست نزن از بعد آن همه را برعکس کن. مقایسه کننده هم اول مقایسه میشود بعد هر نتیجه ای شد و هر حالتی متوقف شد حالت توقف آن میشود حالت شروع یکی از اون 3 تا بعد خروجی تولید میشود. دقت کن یک کپی هم میخواهیم از X,Y چون ممکن است مقایسه کننده تغییر داده باشد در ضرب کننده و افزایش دهنده فقط به X نیاز داریم ولی در جمع کننده به هر 2 نیاز داریم. تز تورینگ: تا الان نشده مسئله یافت که توسط ماشین حل شود توسط تورینگ استاندارد حل نشود و بقیه مدل ماشین های تورینگ هم درسته قدرت و انعطاف بیشتری دارند ولی با تورینگ استاندارد قدرت یکسانی دارند. شبیه سازی ماندن: یکی به راست بعد یکی به چپ یا یکی به چپ بعد یکی به راست. حرف نقطه مشخص میکند که هد در اون نوار کجا بوده است. شارپ ها محدوده نوار ها را مشخص میکند. جای A0B میاد C1D قرار میگیرد. یعنی برو تا به شارپ برسی بعد دنبال نقطه بگرد. به تعداد الفبا باید نقطه اضافه کنیم. دقت کن واسه شیفت به راست اون خونه هایی که شیفت پیدا میکند را در خانه های جدید بنویس و مقادیر قبلی آن ها را به خاطر بسپار. A^nb^n را میتوانی توی 2 تا نوار بنویسی بعد به ازای a یک b بره راست هعی ab، ab کن تا مصرف شود. و از مرتبه o n میشود. هر چه ساختار حافظه tm پیچیده تر شود چیزی به قدرت تورینگ اضافه نمیشود چون اگر بشود تز تورینگ زیر سوال میرود. ماشین دو پشته ای با ماشین تورینگ برابر هست. دقت کن هر مدل محاسبه ای که داشته باشیم بتواند محاسبه ای بکند تورینگ هم میتواند بکند این بدیهی هست ولی قسمت اصلی این است که هر کاری که تورینگ میکند اون مدل محاسبه هم بکند. پشته 1 زیر رشته سمت چپ هد و پشته 2 سمت راست زیر رشته هد را مشخص میکند. برای توصیف لحظه ای میتوانی پشته 1 را بنویسی بعد q که حالت فعلی هست را بنویسی بعد پشته 2 را بنویسی. حرکت به سمت راست را چگونه میتوانیم شبیه سازی کنیم؟ از پشته دوم که بالاش همان هد هست پاپ کنیم به پشته اول پوش کنیم اونوقت بالا پشته دوم دقیقا همان چیزی هست که هد به آن اشاره میکند. برعکس همین هم برای حرکت چپ هست. دقت کن 3 تا پشته هم داشته باشیم باز با همان 2 تا پشته هم معادل هست و همه اینها از تز تورینگ پیروی میکنند. هر LBA یا ماشین کراندار خطی هست که یک ماشین تورینگ هست که از لحاظ حافظه مصرفی O N هست. در حالی که N همان طول رشته ورودی هست دقت کن در نوار با استفاده از کروشه که مانند دیوار عمل میکند نمیگذارد ماشین فرا تر از آن برود و به اندازه رشته هست بخاطر همین حافظه مصرفی آن از مرتبه N هست. دقت کن ماشین اواسط این رشته متوقف میشود و دیگر حرکتی نمیکند کلا بعد از حالت نهایی به هیچ حالتی نمیرویم. اگر حتی 2 یا 3 برابر اندازه رشته هم حافظه مصرف کنی باز از مرتبه o n هست مثلا 4n یا 3n دقت کن وقتی میزان ثابت از نوار استفاده کنی یعنی زبان منظم بوده و همان جواب میداده و الکی قوی تر استفاده کردی. LBA از ماشین تورینگ استاندارد ضعیف تر هست چون یک سری زبان ها وجود دارد که دیگر در مرتبه O N قرار نمیگیرند. برای a^nb^nc^n میتوانی از چپ حرکت کنی a دیدی ضربدر یا b دیدی y بگذار یا c دیدی z بگذار همین را لوپ بزن اگر دیگر به چیزی نرسیدی متوقف میشود. دقت کن اگر حتی چند تا حالت نهایی هم داشتیم باز به یک حالت accept میتوانیم کاهش دهیم. دقت کن وقتی به چپ حرکت میکنی باید a ببینی اون وسط نباید b یا c یا حروف دیگری ببینی دقت کن هعی از چپ شروع میکنی میری جلوی a دیدی ضربدر میگذاری میری جلو تا به اولین b برسی بعد y بگذار بعد برو جلو تا به اولین c برسی جای آن z بگذار بعد از c, z, b, y, a عبور میکنی تا به راست ترین x برسی و حرکت میکنی راست تا a بعدی را ضربدر میزاری همین برای بعدی به صورت لوپ. LBA از ماشین پشته ای قوی تر هست. برای a^nb^n میزان حافظه مصرفی o n بود درست هست جفتشون برابر هست ولی در پشته ای فقط بالای پشته را میبینیم ولی در LBA با همین مرتبه مصرفی دست باز تری داریم. LBA ها برای زبان های حساس به متن هستند. زبان هر LBA لزوما یک حساس به متن هستند همچنین گرامر های حساس به متن هم داریم که این 2 ابزار را برای این زبان داریم. از گرامر به LBA معادل و برعکس رسید. زبانی حساس به متن هست که بتوان برای آن LBA طراحی کرد دقت کن لاندا نباید جز این زبان ها باشد. زبان L7 تنها در صورتی میتوان نوشت که تعداد a ها از 2 برابر m ها کمتر یا مساوی باشد و گرنه اصلا نمیتوان نوشت. وقتی زبان های منظم تحت مکمل بسته هستند یعنی زبان های نامنظم هم تحت مکمل بسته هستند. دقت کن نمیتوانی بگویی چون مستقل از متن نیست مکمل آن هم مستقل از متن نیست این غلط هست ما اثبات کردیم تحت مکمل بسته نیستن به استناد تک نمادی کردیم در زبان l8 و L9. زبان های حساس به متن تحت مکمل بسته هستند. ماشین تورینگ غیر قطعی: NTM یا NDTM گفته میشود. در DTM محاسبه به صورت خطی و دنباله ای از پیکربندی ها هست و یا به accept یا reject و یا اینکه در لوپ میفتیم به هر حال خطی هست ولی در NDTM پیکربندی ها به صورت درختی هست چون به واسطه عدم قطعیت به چند حالت میرویم. و در این درخت دقت کن اگر DFS بری جلو و لوپ باشد هیچ موقع به accept نمیرسی و گیر میکنی. و اگر به reject برسی باز نمیتوانی بگی رشته داخل این زبان نیست چون ممکن هست از یک مسیر دیگر به accept برسی با مصرف همین رشته.

دقت کن درخت باید به صورت BFS پیمایش شود و حتما و حتما اگر یک پیکربندی accept وجود داشته باشد حتما میرسد و اگر تمام مسیر های محاسبه یعنی برگ ها reject باشند متوقف میشود ولی اگر لوپ باشد گیر میکند و ماشین متوقف نمیشود. پس دقت کن اگر یک برگ accept باشد حتما بهش میرسیم میخواهد لوپ باشد یا نباشد ولی اگر همه برگ ها reject بودند و لوپ نبود متوقف میشویم ولی اگر لوپ باشد ماشین متوقف نخواهد شد و مرتب داخل پیکربندی تکراری هستیم. هر NTDM میتوان به یک DTM 3 نوار تبدیل کرد و توان محاسباتی این مدل با مدل استاندارد هم فرقی ندارد پس قدرت این 2 ماشین یعنی NTDM با DTM با هم برابر است. ماشین تورینگ قدرت پذیرش تمام زبان های سوری را دارد. **ماشین تورینگ جهانی**: دنبال ماشین چند کاره هستیم یعنی یک کار نباشد یعنی ماشینی داشته باشیم و بهش ماشین تورینگ بدهیم و اجرا کند رشته مشخص را روی آن ماشین و خروجی را بدهد اگر اون ماشین بگوید بله ماشین تورینگ جهانی میگوید بله و اگر گفت نه این هم میگوید نه. توصیف M یعنی کد دودویی توصیف کننده ماشین M همان ماشین ورودی، این کد را کدینگ های مختلفی تولید میکنند. بعد محتوای نوار و حالت فعلی ماشین m را هم داریم. و ماشین تورینگ جهانی میفهمد حالت فعلی و پیکربندی فعلی ماشین چیست و چه قوانینی انتقالی وجود دارد از روی توصیف اون ماشین یا همان کد دودویی و طبق قانون انتقال برای حرکت بعدی میاد حالت فعلی و محتوای نوار را اصلاح میکند تا برود به پیکربندی بعدی تا متوقف شود یا لوپ بیفتد. چگونه هر ماشین تورینگ را به صورت کد دودویی بگذاریم روی نوار؟ با 0 و1 میتوان کد کرد یک روش کدینگ نمونه: هر حالت را با i تا 0 نشان میدهیم و الفبای نوار را با j تا 0 نشان میدهیم. 1 به عنوان جدا کننده است. از کد به قانون انتقال و برعکس میتوانیم برسیم. دقت کن 2 تا 1 و 3 تا 1 هم جدا کننده هست. با 3 تا 1 اومده است جمله را شروع و بسته است. در واقع ماشین تورینگ m تشکیل شده است از تعداد حالات n تعداد الفبا نوار m و code، t تا قوانین انتقال این یک روش کدینگ هست. دقت کن این code ها هم ترتیب قاموسی مینویسی یعنی اول q1 بعد q2. داخل q1 هم باید ترتیب قاموسی باشد یعنی باید مرتب کنی یعنی اول a بعد b بعد بلنک بعد اگر داشتی a,a و داشتی a,b اول میروی سراغ a,a بعد a,b. دقت کن به اندازه شاخص هر الفبا توان میگذاری روی صفر ها مثلا a چون index آن 1 هست ما یک صفر گذاشتیم برای سایر الفبا های روی نوار هم همین. بین هر قانون انتقال 2 تا 11 برای حالتی هم که میرود به اندازه شماره آن توان بگذار روی صفر. اجزای هر انتقال با 1 و بین هر قانون انتقال با 11 جدا شده اند. کد همه ماشین های تورینگ زیر مجموعه اعداد طبیعی هست. دقت کن بعضی کد ها کد ماشین تورینگ نیست مثل 1111 یا 01110. دقت کن کد هر ماشین تورینگ معادل یک عدد طبیعی هست ولی هر عدد طبیعی لزوما یک کد تورینگ نیست. دقت کن مجموعه همه ماشین های تورینگ روی الفبا سیگما یک مجموعه شمارای نامتناهی هست. شمارا یعنی یا محدود هست یا اگر محدود نباشد و نامحدود باشد باید هم اندازه مجموعه اعداد طبیعی باشد، و میدانیم که مجموعه ماشین های تورینگ زیر مجموعه اعداد طبیعی هست و چون خود اعداد طبیعی شمارا هست اینها هم شمارا هستند. برای رشته w هم یک نوعی کد کن بعد با کد ماشین تورینگ الحاق کن. همین برای گرامر و رشته هم میشود و الحاق میکنیم. دقت کن همه اینها توصیف هستند برای اینکه بدهیم به ماشین جهانی و DFA را میتوانیم اجرا کنیم و اگر DFA بگوید بله تورینگ جهانی هم میگوید بله و از روی توصیف DFA میفهمد چه قانون انتقالی باید اجرا شود بعد میرود استیت را اپدیت میکند و هد را هم میبرد جلو روی نوار این DFA و در نهایت یک جا متوقف خواهد شد اگر در فاینال متوقف شد و گفت بله جهانی میگوید بله و رشته به زبان تعلق دارد و برعکس. **ماشین صفی**: در حالی که امکان ذخیره و بازیابی داده از طریق حافظه نامتناهی صف وجود دارد یعنی برخلاف ماشین پشته ای که حافظه نامتناهی آن پشته بود معادل تورینگ هست یعنی از تورینگ به صفی و برعکس رسید یعنی قدرت یکی دارند. دقت کن در نکات جفت حالت اول حافظه ندارند پس منظم میپذیرند فقط. دقت کن در نکته سوم اگر بخواهی یک جا دیگر کپی کنی و ببری چون باید روی اونجا بنویسی ولی نمیتوانی بنویسی پس فقط منظم میپذیرد.

دقت کن نکته بعدی را قبلا داشتی با accept که اضافه میکردی وقتی متوقف میشد در یک حالت نهایی با یک بلنک یا هر حرفی میومد به این حالت در واقع تعریف همان accept میشود یک جورایی.

**جلسه سی‌ام:**

مجموعه های شمارا و ناشمارا: اگر نامحدود بود باید بتوان یک تابع تعریف کرد که مجموعه اعداد طبیعی روی اون تابع 1 به 1 باشد و همچنین پوشا باشد در واقع باید تناظر یک به یک ایجاد کنی پس قطعا شمارا هست با وجود اینکه نامتناهی باشد. دقت کن اون مجموعه ها هم اندازه با مجموعه اعداد طبیعی هستند. در چینش طی نامتناهی گام میتوانیم متوقف شویم و میتوانیم به نوعی بچینیم که با اعداد طبیعی یک چینش متناظر داشته باشیم تا یک مجموعه شمارا باشد واسه سیگما پلاس هم همین شکلی هست. نامحدود متناهی هم اندازه مجموعه اعداد طبیعی هست. یک مجموعه محدود، مجموعه توانی آن ها هم محدود است پس مجموعه توانی آن هم شمارا هست. چینش های نادرست چون نامتناهی هستند یک سری از رشته ها هیچوقت به آنها نمیرسد پس نیاز به چینش های درست داریم. اگر یک مجموعه شمارای نامتناهی باشد مجموعه توانی آنها لزوما ناشمارا هست. روش اثبات به این شکل هست که فرض میکنیم اول یک مجموعه متناظر از مجموعه اعداد طبیعی به مجموعه توانی سیگما استار وجود دارد. هر زبان را در قالب یک رشته نامتناهی روی الفبا صفر و یک نشان میدهیم اون مواردی که بودند 1 و اونهایی که نبودند 0. و برای خلاف اثبات کردن فرض کافی است که یک زبان پیدا کنیم که تصویر هیچ عدد طبیعی نباشد. برای پیدا کردن این زبان روی قطر حرکت میکنیم بیت اول تا آخر را نات میکنیم تا یک زبانی تولید شود که با همه زبان هایی که داریم لااقل روی یک بیت متفاوت هست و تصویر هیچ عدد طبیعی نیست. و دقت شود که مجموعه زبان هایی که ماشین تورینگ برای آنها وجود ندارد قوی ترین مدل محاسبه ما خیلی بیشتر از زبان هایی هست که ماشین تورینگ واسه اونها وجود دارد چون خود مجموعه توانی سیگما استار بسیار بزرگ است. اجتماع زبان های غیر رسمی با صوری میشود مجموعه توانی سیگما استار. مجموعه همه زبان های نامنظم ناشمارا هست ولی وقتی محدودش کنی به سلسله مرتبه سوری ها چون خود سوری ها شمارا هستند هر زیر مجموعه از آن هم شمارا هست. همین برای مستقل از متن ها هم صدق میکند. چون کل فرمال ها شمارا هست پس کل زیر مجموعه از آنها شمارا هست ولی هر چی که بیرون از زبان های سوری باشد ناشمارا هست. هر زبانی که نامتناهی باشد قطعا بیرون از منظم ها هست چون اگر متنهای بود منظم بود. بین خطی ها و مستقل از متن میتوان تناظر یک به یک برقرار کرد. خود سیگما استار شمارای نامتناهی است و هر زیر مجموعه از آن هم شمارا هست پس مستقل از متن ها و خطی ها چون خودشان زیر مجموعه سیگما استار هستند و هم اندازه زبان های طبیعی هستند و شمارا هستند پس میتوان تناظر یک به یک ایجاد کرد. ولی همین کار را برای منظم ها و خطی ها نمیتوان کرد چرا که منظم ها محدود هستند ولی اونها نا محدود هستند و نمیتوان تناظر یک به یک ایجاد کرد. اما اگر منظم نامحدود را میتوان تناظر یک به یک با خطی ایجاد کرد. واسه DFA ها توجه کن که همانطور که تورینگ را کد میکردیم و هر کد معادل یک عدد طبیعی بود برای آنها هم همین هست یعنی معادل هستند یعنی هر DFA با کدینگ معادل یک عدد طبیعی هست اگر چه که برعکس این درست نیست چون ما اعداد طبیعی داریم که معادل آن DFA یا ماشین تورینگی وجود ندارد. همین برای بقیه هم وجود دارد. ماشین تورینگ تصمیم گیرنده یک الگوریتم عضویت هست و تعلق و عدم تعلق رشته به زبان را دارد و هیچ موقع در لوپ گیر نمیکند اگر یک تورینگی به این شکل وجود داشته باشد و بتواند راجب یک زبان تصمیم گیری کند به این زبان بازگشتی گفته میشود. تصمیم گیری در کلاس زبان های بازگشتی است یعنی فقط برای بازگشتی ها الگوریتم عضویت داریم که همان ماشین تورینگ تصمیم گیرنده هست. زبان بازگشتی حتما و حتما یک ماشین تورینگ تصمیم گیرنده دارد. پس چون متوقف میشود الگوریتم دارد. تشخیص دهنده برخلاف تصمیم گیرنده میتواند در لوپ هم بیفتد. هر تصمیم گیرنده میتواند تشخیص دهنده باشد ولی عکس آن درست نیست. **زبان بازگشتی شمارش پذیر**: به اختصار RE هم گفته میشود که اگر یک تورینگ تشخیص دهنده باشد که بتواند زبان مورد نظر را تشخیص دهد. به این زبان، زبان های تشخیص پذیر گفته میشود. دقت کن به اونایی که ماشین تصمیم گیرنده آنها را تشخیص میداد و دارا بودند زبان های تصمیم پذیر گفته میشد. شمارش پذیر ها لزوما به ازای هر ورودی متوقف نمیشود برخلاف قبلی ها چون ممکن هست در لوپ بیفتد برای رشته هایی که داخل زبان نیست اسم دیگر این زبان ها، زبان نیمه تصمیم پذیر یا نیمه حل پذیر گفته میشود چون برای تمام ورودی های داخل زبان حتما متوقف میشود. هر ماشین تصمیم گیرنده یک ماشین تشخیص دهنده هست ولی برعکس آن درست نیست به همین ترتیب هر زبان بازگشتی یک زبان بازگشتی شمارش پذیر هست ولی عکس آن درست نیست چون بازگشتی ماشین تصمیم گیرنده دارد و بازگشتی شمارش پذیر ماشین تشخیص دهند. این 2 رده بازگشتی و بازگشتی شمارش پذیر به جای همان زبان های آزاد یا نوع صفر قرار میگیرند. در این رده LBN های حساس به متن هم قدرت محاسبه این زبان ها را ندارند و ما بازگشتی که حساس به متن نیست را داریم. دقت کن نمودار این صفحه همان نمودار زبان های فرمال هست. اگر یک زبانی فرمال باشد و مکمل آن هم فرمال باشد قطعا تصمیم پذیر بوده است و زبان های تشخیص پذیر مکمل آن ها اصلا فرمال نیست که اگر فرمال بود اون زبان تصمیم پذیر هست. برای اجتماع هم یادت باشد که گام به گام حرکت کنی و به هر 2 ماشین همزمان بدهی اگر گفتند بله که هیچی اگر گفتند نه هم که مشخص هست اگر هم در لوپ بیفتند مشخص هست اگر جدا به جدا بدهی ممکن هست یکی در لوپ بیفتد و نتوانی از دیگری استفاده کنی. برای تصمیم گیرنده دیگر نیازی به موازی اجرا کردن مثل قبلی ها چون حتما متوقف میشوند و الگوریتم عضویت دارند و مثلا برای اشتراک اگر ماشین اول گفت نه که خوب اشتراک به بله هر 2 تا نیاز دارد پس کلا نه ولی اگر اولی گفت بله میرویم سراغ دومی تا دومی هم بگوید بله اگر گفت نه هم که مشخص هست. برای اجتماع اگر ماشین اول گفت بله که اصلا سراغ دومی نمیرویم طبق مفهوم اجتماع اگر گفت نه سراغ دومی میرویم. دقت کن بازگشتی ها زیر مجموعه ای از بازگشتی های شمارش پذیر هستند. برای مکمل هم کافی هست جایی که ماشین تورینگ تصمیم گیرنده گفت بله مکمل همین ماشین بگوید نه و برعکس تا اثبات مکمل برقرار شود یعنی انگار جای accept و reject را عوض کنی برای این زبان. مکمل زبان بازگشتی هم بازگشتی هست. مستقل از متن تحت مکمل بسته نیست پس مکمل آن حتما بازگشتی هست چون بازگشتی ها تحت مکمل بسته هستند اما اینکه حساس به متن هست یا نیست بستگی به حساس به متن ها دارد، حساس به متن ها تحت مکمل بسته هستند پس مکمل زبان مستقل از متن قطعا حساس به متن هست و نیازی به بازگشتی اینا نیست. پس اونایی که تحت مکمل بسته هستند مکمل های آنها هم جز همان نوع ها هستند. و برعکس هم همین هست یعنی چون منظم ها مکمل آنها به نامنظم نمیرسد و برای بقیه هم همینطور پس نامنظم ها زبان های حساس به متن و زبان های غیر بازگشتی تحت مکمل بسته هستند. خطی ها تحت مکمل بسته نیستند یا مستقل از متن قطعی تحت مکمل بسته هستند.

دقت کن اون زبان هایی که خودش و مکملش بازگشتی شمارش پذیر باشند قطعا اون زبان بازگشتی هست و برعکس. اگر بازگشتی باشد قطعا بازگشتی شمارش پذیر هم هست و تصمیم گیرنده تشخیص دهنده هم هست و هر تصمیم پذیر قطعا تشخیص پذیر هم هست واسه همین هست که بازگشتی شمارش پذیر هم هستند. اثبات سطر اول هم به این شکل هست که ورودی را همزمان به L و L بار بدهی و یک گام برو جلو اگر L گفت بله تصمیم گیرنده L میگوید بله اگر L نگوید بله مثلا در لوپ بیفتد حتما و حتما L بار آن میگوید بله و ماشین تورینگ تصمیم گیرنده میگوید خیر. مکمل زبان منظم بازگشتی هم هست چون مشخص هست همه اینها زیر مجموعه هم هستند و همه اینها زیر مجموعه تایپ 0 هستند. زبان هایی که غیر فرمال هستند مکمل آنها هم غیر فرمال هست. غیر رسمی ناشمارا هست و خیلی بیشتر از اعداد طبیعی و فرمال ها هست چرا ناشمارا هست چون دو به توان سیگما استار ناشمارا هست. مکمل هر زبان حساس به متن نا مستقل از متن لزوما نا مستقل از متن نیست بلکه ممکن هست مستقل از متن باشد. دقت کن پذیرش DPDA توسط پشته خالی یک زیر مجموعه محضی از مستقل از متن قطعی هست و اونایی هست که prefix free باشند. برای زبان های تصمیم پذیر از گرامر های بدون محدودیت که واسه تشخیص پذیر ها هستند استفاده میکنیم.

**جلسه سی‌یکم:**

اگر ب که الف به آن کاهش یافته قابل حل باشد الف هم قابل حل هست. درجه سختی الف از ب بیشتر نیست. تصمیم پذیر همان قابل حل بودن هست، decider همان ماشین تورینگ تصمیم گیرنده هست پس خود acceptance DFA یک زبان بازگشتی هست چرا چون تصمیم پذیر هست بخاطر ماشین تورینگ بعد تورینگ اگر DFA گفت بله این هم که تصمیم گیرنده هست میگوید بله و برعکس. برای NFA را میایم به DFA کاهش میدهیم به صورت الگوریتمیک پس چون کاهش دادیم و DFA تصمیم پذیر هست پس NFA هم تصمیم پذیر هست. و سختی NFA از DFA بیشتر نیست. هر مسئله ای را به توان به یک زبان بازگشتی بیان کرد قابل حل هست چون ماشین تصمیم گیرنده دارد. پس هر عبارت منظم هم تصمیم پذیر هست چون به NFA و بعد به DFA کاهش میدهیم. برای گرامر منظم هم همین هست. توصیف زبان های بازگشتی شمارش پذیر یعنی مثلا برای ماشین تورینگ های 1 و2 آیا زبان های آنها با هم برابر هست؟ اشتراک آنها تهی هست؟ یک زبان زیر مجموعه دیگری هست؟ رشته مشترکی را تولید میکنند یا نه و یا توصیف میکنند یا نه جز مسائل تصمیم ناپذیر هستند. نامتناهی بودن یک زبان هم تصمیم ناپذیر هست و توصیف هم میتواند بدهد مثلا ماشین تورینگ زبانش تهی هست یا نامتناهی هست یا نه. یا مثلا گرامر بدون محدودیت بدهد. دقت کن زبان منظم برابر سیگما استار هست یا نه تصمیم پذیر هست این قوانین برای بازگشتی های شمارش پذیر هستند. دقت کن همیشه جای عنوان اون زبان ها میتواند توصیف بدهد مثل ماشین پشته ای واسه مستقل از متن ها. Completeness یعنی زبان مورد نظر برابر سیگما استار باشد که در مستقل از متن ها و بازگشتی شمارش پذیر اینها مسائل تصمیم ناپذیر هستند. دقت کن دلیل اینکه مستقل از متن قطعی کامل هستند یا نه تصمیم پذیر هست چون تحت مکمل بسته هستند و خود زبان با سیگما استار در صورتی برابر هست که L منهای سیگما استار شود تهی شود و سیگما استار منهای L اجتماع آن تهی باشد اولی که مشخص هست چون سیگما استار مجموعه مرجع هست راجب دومی چون حاصل منها میشود مکمل L و مکمل L هم قطعا مستقل از متن قطعی هست و تهی بودن آنها یک مسئله تصمیم پذیر هست.