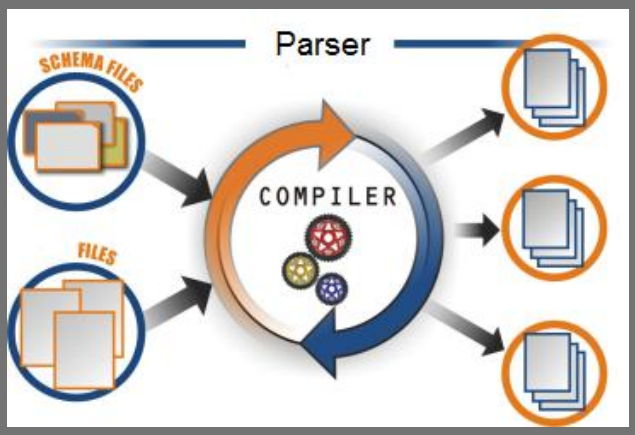
2010

Compiler T3

Parsing Table Prognosticator M

Help

نمودارهای تغییر حالت برای نمایش تجزیه کننده های پیشگو مناسب اند . تجزیه کننده های پیشگو می توانند برای گرامرهای LL(1) نوشته شوند، زیرا قانون تولید مناسب برای غیر پایانه می تواند فقط با نگاه کردن به نماد ورودی فعلی انتخاب شود . ساختارهای جریان کنترل، به همراه کلمه های کلیدی متمایز کننده ی آنها، معمولاً قیدهای LL(1) را برآورده می کنند.



Behzad Khosravifar
COMPILER T3
Behzad.khosravifar@Gmail.com
B.B
6/5/2010

INTRODUCTION

پیشگو، یعنی، تجزیه کننده های بازگشتی – کاهشی، که به عقبگرد نیاز ندارند، می توانند برای دسته ای از گرامرها به نام LL(1) ساخته شوند. اولین "L" در LL(1) به معنای پویش ورودی از چپ (Left) و دومین "L" برای تولید چپ ترین اشتقاق، و "۱" به معنای یک نماد ورودی پیش نگر در هر مرحله است تا تجزیه کننده بتواند تصمیم گیری کند

گرامرهای دسته LL(1) به اندازه کافی غنی هستند تا اغلب ساختارهای برنامه نویسی را پوشش دهند، ک رچه نوشتن گرامر مناسب برای زبان مبداء نیاز به دقت فراوانی دارد. به عنوان مثال، هیچ گرامر بازگشتی چپ یا مبهم نمی تواند LL(1) باشد.

در زیر یک مثال از نحوه ی تولید جدول تجزیه پیشگو برای گرامرهای ورودی بیان شده است:

نکته

اولین گرامر ورودی به عنوان قانون شماره ۱ و متغیر شروع تجزیه در نظر گرفته می شود.

1. E -> TR
2. R -> +TR
3. R -> -TR
4. R -> ε
5. T -> FP
6. P -> ×FP
7. P -> /FP
8. P -> ε
9. F -> id
10. F -> num
11. F -> (E)

تمام حروف و یا متغیر های به کار رفته در سمت چپ هر قانون به عنوان یک غیر پایانه در نظر گرفته می شود؛ البته تکرار حرف یک غیر پایانه در سمت چپ گرامرهای مختلف همان یک غیر پایانه با چندین قانون است . برای مثال گرامر های زیر:

$A \rightarrow \alpha\beta B$

$A \rightarrow \varepsilon$

معادل گرامر $A \rightarrow \alpha\beta B \mid \varepsilon$ می باشد که در آن A یک غیر پایانه شناخته شده است.

با توجه به قوانین ورودی ابتدا باید جدول First (non-Terminal) تمام غیر پایانه ها را بدست آورد:

First (E) = First (TR) =

= First (T) + if (First (T) have ε) then First (R)

First یک غیر پایانه برابر First کل قوانین تولیدی اش می باشد که در صورت وجود ε در هر یک از غیر پایانه ها عمل First غیر پایانه ی بعدی هم اضافه می شود . مثلاً در مثال فوق اگر First (T) دارای ε باشد به دنبال متغیر بعدی می رویم؛ که در صورت غیر پایانه بودن متغیر First آن متغیر و در صورت پایانه بودن خود پایانه را جزء First محسوب کرده، و دیگر ادامه نمی دهیم.

اگر یک غیرپایانه چند قانون داشته باشد ، اجتماع First تمام قوانین مربوط به آن را به عنوان First اش محصوب می کنیم.

در زیر تابع First برای تمام غیر پایانه های گرامرها اجرا شده و نتیجه ی آن مشاهده می شود

First (E) = { id, num, (}
First (R) = { +, -, ? }
First (T) = { id, num, (}
First (P) = { ×, /, ? }
First (F) = { id, num, (}

همان طور که مشاهده می کنید First غیر پایانه ها بدست آمده و حال باید به سراغ Follow غیر پایانه ها برویم.

برای بدست آوردن Follow باید لیست پویایی برای هر غیر پایانه داشته باشیم زیرا هر لحظه امکان اضافه شدن توکنی به لیست وجود دارد . سپس تمام قوانین را توکن به توکن خوانده و Follow اش را حساب کرده و به لیستش اضافه کنیم . در صورت رسیدن به محاسبه ی Follow یک توکن غیر پایانه به متغیرهای بع د از خود نگاه می کنیم و First شان را به عنوان Follow در نظر می گیریم.

A → $\alpha\beta BCDEF$

Follow (α) = پایانه ها را حساب نمی کنیم

پس به توکن بعدی می رویم:

Follow (β) =

First (BCDEF) + if (First (BCDEF) have ε) then + Follow (A)

در زیر Follow تمام غیر پایانه ها نمایش داده شده است:

Follow (E) = { $,) }
Follow (R) = { $,) }
Follow (T) = { +, -, $,) }
Follow (P) = { +, -, $,) }
Follow (F) = { ×, /, +, -, $,) }

حال نوبت به محاسبه جدول تجزیه پیشگو می رسد.

برای محاسبه ی این جدول به اطلاعات هر دوی First و Follow نیازمندیم.

ابتدا باید بدانیم که هر دستور عامل تولید چه توکن های پایانه ای است؛ و برای این کار شماره دستور غیرپایانه را به پایانه های حاصل از First توکن های سمت راست ربط می دهیم و اگر در بین این پایانه ها ε هم وجود داشت آن ε را حذف کرده و بجای آن توکن های پایانه ای حاصل از Follow غیر پایانه مربوط به دستور را اضافه می کنیم. و بقیه پایانه هایی که ربطی به غیر پایانه ندارند را با رابطه ی "خطا" نشان می دهیم.

و در آخر جدول تجزیه پیشگو را برای گرامر مثال فوق مشاهده می کنید:

در هنگام شروع بدست آوردن Follow غیر پایانه ها ابتدا علامت $ را به Follow غیر پایانه ی قانون اول اضافه میکنیم . برای مثال در گرامرهای ذکر شده داریم:

Follow (E) = { $, . . .

نکته ی دوم این است که در هنگام بدست آوردن Follow غیرپایانه ی جاری اگر First تمام متغیرهای بعد از خود دارای ε باشند و یا هیچ متغیری بعد از آن وجود نداشته باشد، آنگاه Follow غیر پایانه ی سمت چپ قانون را هم به Follow غیرپایانه ی جاری اضافه می کنیم.

در صورت مشاهده یک پایانه در ادامه ی یک غیرپایانه، First تمام غیر پایانه ها را تا آن پایانه + خود پایانه را به عنوان Follow آن غیر پایانه جاری اضافه می کنیم و به متغیرهای بعد از پایانه برای محاسبه اش توجه نمی کنیم.

غیر پایانه ها	+	-	×	/	id	num	(	)	$
E	خطا	خطا	خطا	خطا	1	1	1	خطا	خطا
R	2	3	خطا	خطا	خطا	خطا	خطا	4	4
T	خطا	خطا	خطا	خطا	5	5	5	خطا	خطا
P	8	8	6	7	خطا	خطا	خطا	8	8
F	خطا	خطا	خطا	خطا	9	10	11	خطا	خطا

Workmanship Program

در برنامه ی Compiler T3 تمام مراحل بالا به طور دقیق اجرا می شود. تنها نکته ی مهم در این برنامه طریقه ی وارد کردن گرامرها بصورت صحیح می باشد. برای وارد کردن هر قانون یک قالب عمومی وجود دارد که باید رعایت شود، در غیر این صورت برنامه آن قانون را به عنوان یک خطای گرامری شناخته و از اجرای کامپایلر جلوگیری می کند. ابتدا باید به چند نکته از این قالب توجه کرد:

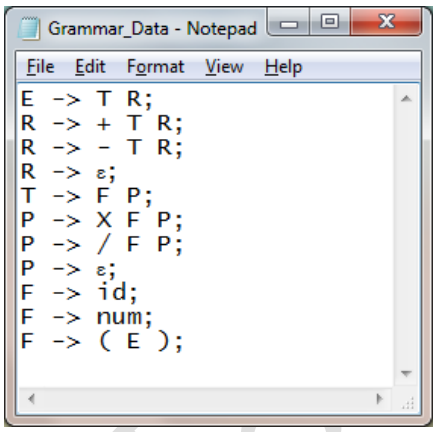
- برای شناسایی پایانه ها و یا غیر پایانه ها از هم باید هنگام وارد کردن گرامر بین آن ها فاصله وجود داشته باشد.
- فلش ربط دهنده ی غیر پایانه به قانون اش به صورت -> می باشد. در ابتدا و انتهای فلش فاصله وجود دارد.
- سمت چپ فلش در هر گرامر جایگاه فقط و فقط یک غیر پایانه می باشد A -> ...;
- غیر پایانه ها و پایانه ها می توانند چند حرفی باشند. ولی حرف اول باید از حروف انگلیسی باشد. A1b -> a0 a2 ab ...;
- حرف اول در غیر پایانه ها فقط و فقط حروف بزرگ انگلیسی می تواند باشد؛ ولی حرف اول و یا دیگر حروف پایانه ها نیز می توانند از حروف بزرگ انگلیسی باشند. A1 -> + Ba C;
- زبان گرامری که این برنامه پشتیبانی می کند LL(1) می باشد بنابراین همان طور که گفته شد نباید دارای ابهام و بازگشتی چپ باشد، زیرا آنگاه دارای این شرط نیست و خروجی برنامه نادرست می باشد
- در انتهای هر قانون برای مشخص کردن اتمام گرامر مربوطه از سمیکالون ; استفاده می کنیم. S -> + T R;
- در سمت راست هر قانون در صورت وجود فضای خالی بین آخرین توکن و سمیکالون; باعث بروز خطا خواهد شد، زیرا فضای خالی به مفهوم وجود توکن بعدی می باشد. A -> α β B C D E F;
- اگر وجود یک فضای خالی space اگر موردی نداشته باشد آنگاه چندین فضای خالی هم موردی نخواهد داشت.
- برای وارد کردن علامت تهی و یا همان اپسیلون ε در گرامرها باید از کلید مقرر شده در قسمت بالای برنامه استفاده کرد.

و در آخر می توان از مترجم هوشمند متن گرامری مفروض استفاده کرد؛ بطوری که می توانید بعد از اتمام نوشتن یک بند از گرامر در همان لحظه درستی و یا عدم درستی گرامرهای نوشته شده را با توجه به اعلان بالای برنامه متوجه شوید.

برای راحتی تایپ و صرفه جویی در وقت می توان از اعمال بارگذاری و بازیابی از فایل های نوشتاری استفاده کرد. ولی باید توجه کرد که در هنگام نوشتن گرامرها در فایل Text (*.txt) نباید شماره ی دستور و یا سطر را نوشت.

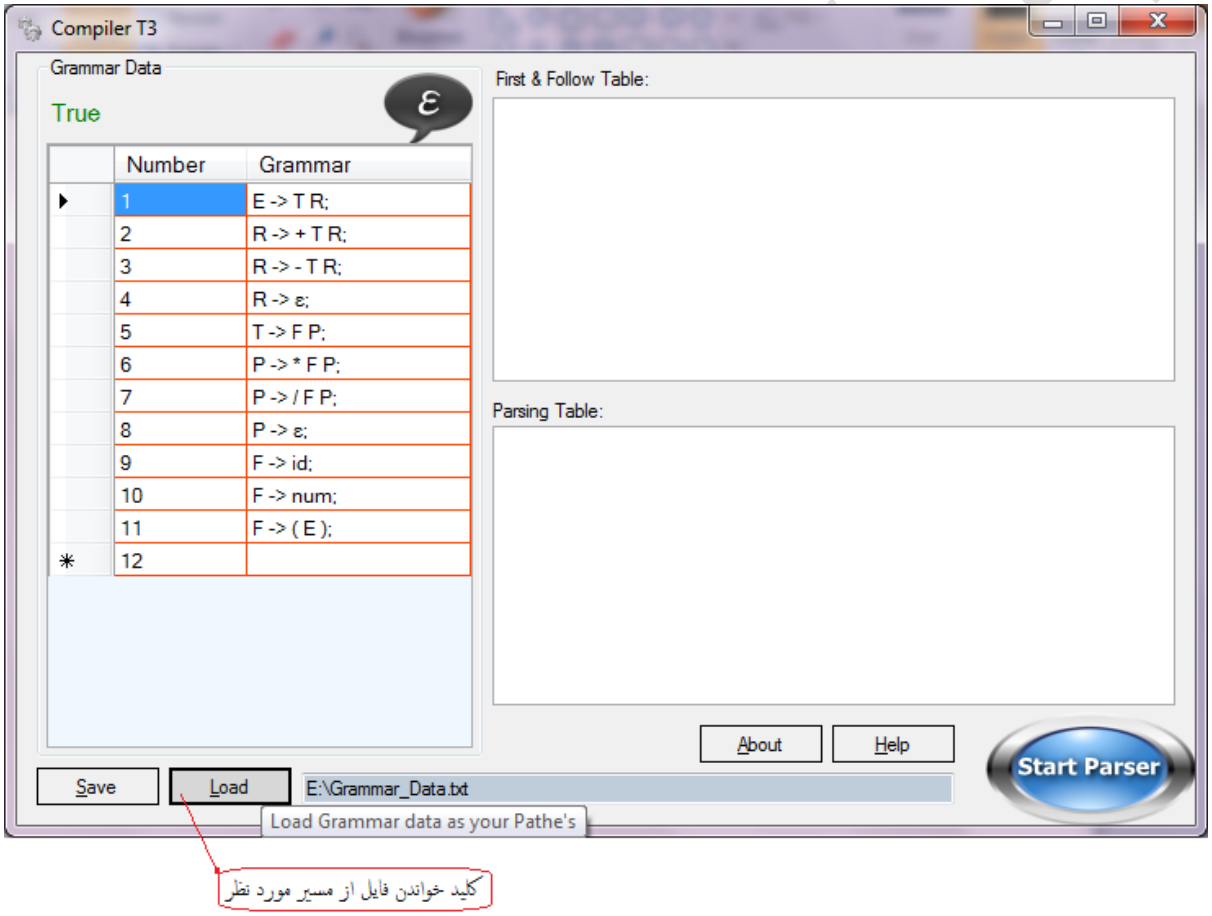
برای مثال ما یک فایل تکست حاوی گرامر مثال ذکر شده را به برنامه می دهیم

محتوای فایل ورودی :



همان طور که مشاهده می شود فایل تکست حاوی گرامر مثال توسط برنامه Notepad خود ویندوز باز شده است که محتویات آن قابل رویت می باشد. و ما مسیر این فایل را توسط کلید Load به برنامه وارد می کنیم. و همین کار را می توانستیم به صورت دستی هم انجام دهیم و تک تک قوانین را به برنامه وارد کنیم.

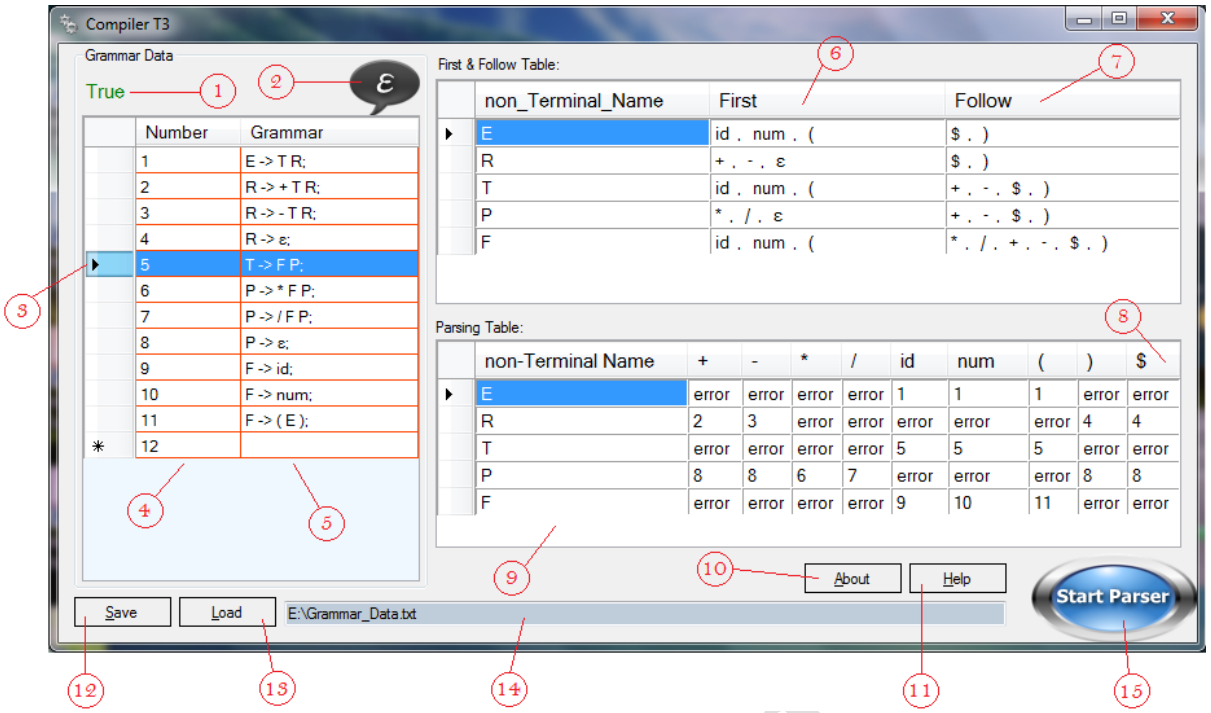
ورود اطلاعات به برنامه از فایل نوشتاری :



و همین اطلاعات را می توان با زدن کلید Save در داخل یک فایل نوشتاری ذخیره کرد تا در آینده بتوان همان‌فایل را Load کرد.

در صورت درست بودن گرامرهای ورودی کلید Start Parser فعال می شود که با کلیک بر آن می توانید جدول تجزیه پیشگو را مشاهده کنید. علاوه بر جدول تجزیه ، حاصل توابع First , Follow هم نمایش داده می شود.

در تصویر بعد تمام قسمت های برنامه به صورت شماره گذاری شده توضیح داده خواهد شد.



۱) نشان دهنده درستی یا عدم درستی گرامرهای وارده در هر لحظه از ورود اطلاعات و یا تغییر اطلاعات

۲) برای تایپ حرف اپسیلون ε در هنگام وارد کردن قوانین به صورت دستی و یا در هنگام تغییر دادن آنها.

۳) انتخاب رکورد جاری برای نمایش و یا حذف آن رکورد، که با فشار دادن کلید Delete در آن هنگام سطر انتخابی را پاک می کند.

۴) این ستون مربوط به شماره ی قوانین می باشد و به صورت خودکار وارد می شود و حتی در موقع حذف رکوردی ترتیب آن حفظ شده و تمام رکوردها را به بالا شیفت می دهد

۵) این ستون مربوط به گرامرهای ورودی می باشد. و می توان آن را هم بصورت دستی و هم بصورت فایل نوشتاری وارد کرد.

۶) این ستون مربوط به پایانه های حاصل از خروجی تابع First برای هر غیر پایانه می باشد.

۷) این ستون نشان گر پایانه های موجود در Follow هر غیر پایانه است.

۸) ستون های مربوط به همه پایانه ها در جدول تجزیه پیشگو ، که هر کدام از این ستون ها برای یک پایانه ی منحصربه فرد می باشد

۹) جدول تجزیه پیشگو با گرامرهای ورودیه غیر بازگشتی چپ.

۱۰) درباره ی برنامه و ارتباط با سازنده برنامه ی Compiler T3.

۱۱) راهنمای استفاده از برنامه.

۱۲) کلید ذخیره اطلاعات گرامرهای ورودی در مسیری که شما به آن می دهید

۱۳) کلید بازیابی اطلاعات گرامرها از مسیری که شما وارد می کنید.

۱۴) نمایش مسیر ذخیره و یا بازیابی اطلاعات گرامرها.

۱۵) کلید شروع پردازش کامپایلر و نمایش جدول تجزیه ی پیشگو در خروجی.

این برنامه توسط آقای بهزاد خسروی فر تهیه و تولید شده است. و هدف از آن فقط استفاده ی آموزشی دارد.

هر گونه استفاده ی تجاری از این برنامه سو استفاده از حق تولید کننده محسوب می شود. ولی استفاده ی آموزشی آن آزاد می باشد.

Programmer Name's :	Behzad Khosravifar
Program Name's:	Compiler T3
Respective Branch :	Compilers
Programmer Branch :	Software Engineering
Respective Master Name :	Mr. S.Sobhani
Credit Date :	7 jun(6) 2010 - ۱۷ خرداد ۱۳۸۹
Program Platform's :	.NET Framework 3.5
Program Language's :	C#.NET 3.5
Email :	Behzad.khosravifar@gmail.com
Email :	Behzadkh@hotmail.com
Web Site :	www.Azerbaycan.ir
Web Site :	www.Unixe.co.cc
Mobile :	+98 9149149202
University :	UCNA – University College of Nabi Akram

I HOPE YOU ENJOY USING THESE PROGRAMS