



Prolog

PROgrammation en LOGique

یادداشت‌هایی در زمینه‌ی پرولوگ

محمد علی آستی علی‌بیک

ناصر باقری محمودآبادی

نوع مقاله : علمی - مروری

دانشگاه یزد - ۱۳۹۲

فهرست

- تاریخچه
- نصب و بکارگیری PIE PROLOG
- اولین برنامه به پرولوگ و بررسی آن
- قواعد پرولوگ
- مقایسه عبارتها
- محاسبه
- مقایسه عبارتهای محاسباتی
- توابع کتابخانه‌ای در پرولوگ
- **A Classic Example**
- قوانین بازگشتی
- مثال‌هایی از برنامه‌نویسی در پرولوگ
- اشکال زدایی
- منابع

تاریخچه

پرولوگ، اولین زبان برنامه‌نویسی بر مبنای منطق در سال ۱۹۷۲ در دانشگاه ماری و توسط آلن کالمرار^۱ و فیلیپ راسل^۲ بنا شد.

Prolog مخففی برای "PROgrammation en LOGique" می‌باشد. اساس پرولوگ شامل یک روش برای مشخص کردن گزاره‌های محاسبات گزاره‌ای و تصمیمات محدود است.

برنامه‌نویسی در پرولوگ شامل مشخصات حقیقی در مورد اشیاء و ارتباط آنها و قوانینی که ارتباطات را مشخص می‌کند، است. برنامه‌های پرولوگ مجموعه‌ای از جملات اعلانی در مورد یک مسئله هستند زیرا آنها نحوه محاسبه نتیجه را مشخص نمی‌کند، بلکه ساختار منطقی نتیجه را مشخص می‌کند. پرولوگ با برنامه‌نویسی دستوری و حتی برنامه‌نویسی تابعی در تعریف نحوه محاسبه نتیجه کاملاً متفاوت است. با استفاده از پرولوگ برنامه‌نویسی می‌تواند در یک سطح خیلی خلاصه و کاملاً نزدیک به مشخصات رسمی یک مسأله انجام گیرد. پرولوگ هنوز هم مهمترین زبان برنامه‌نویسی منطقی است.

همچنین از پرولوگ در زمینه‌های هوش مصنوعی مانند:

سیستمهای خبره یا هوشمند (Expert systems)، اثبات نظریه (theorem proving)، طراحی کامپیوتری (CAD)، پردازش زبان طبیعی بطور موفقیت‌آمیزی استفاده شده است. اما در زمینه‌های دیگری مانند سیستم‌های مدیریت پایگاه داده رابطه‌ای یا در آموزش نیز استفاده می‌شود.

John Alan Robinson، یکی از طلایه داران Logic Programming در سال ۱۹۶۵ مقاله‌ای به چاپ رساند که در آن مساله یکسان سازی را به صورت فرمال بیان کرد و سپس الگوریتمی برای یافتن آن ارائه داد. با این کار یک شاخه جدید در علم کامپیوتر به نام منطق محاسباتی ایجاد شد. همچنین کار Robinson مبنای زبان پرولوگ گشت.

پرولوگ یک زبان توصیفی می‌باشد. این بدین معنی است که واقعیتهای و قواعد مورد نیاز را می‌دهیم و آنگاه این پرولوگ است که با بکارگیری استنتاج قیاسی به حل مسئله (یافتن پاسخ برای سوال کاربر) می‌پردازد. البته در غالب مواقع سوال کاربر که تحت عنوان پرس و جو^۳ یا هدف^۴ از آن یاد می‌شود شامل یک و یا چند متغیر است و این بدین معنی است که از موتور استنتاج پرولوگ می‌پرسیم که این متغیرها چه مقادیری می‌توانند به خود بگیرند تا با آنچه در قالب واقعیت‌ها و قواعد داده شده‌اند از لحاظ منطقی درست و سازگار باشند. این در حالی است که در زبان‌های برنامه‌نویسی معمول مانند پاسکال، C و مانند آن که زبانهای رویه‌ای بحساب می‌آیند این برنامه‌نویس است که بایستی

1 Alain Colmerauer

2 Phillipe Roussel

3 query

4 Goal

رویه‌ای تهیه کند که بصورت قدم به قدم به کامپیوتر بگوید که به چه ترتیب مسئله را حل کند.

در این بخش به دنبال آشنایی با مبانی اولیه برنامه نویسی به پرولوگ و شروع به کار سریع با آن می‌باشیم. برای یادگیری پرولوگ نیاز به نصب مفسر مناسب و اجرای برنامه‌ها داریم. برای این منظور می‌توانید از مفسرهای مختلفی استفاده کنید. دو مورد اولیه که در این بخش معرفی می‌شوند عبارتند از:

PIE Prolog و SWI Prolog.

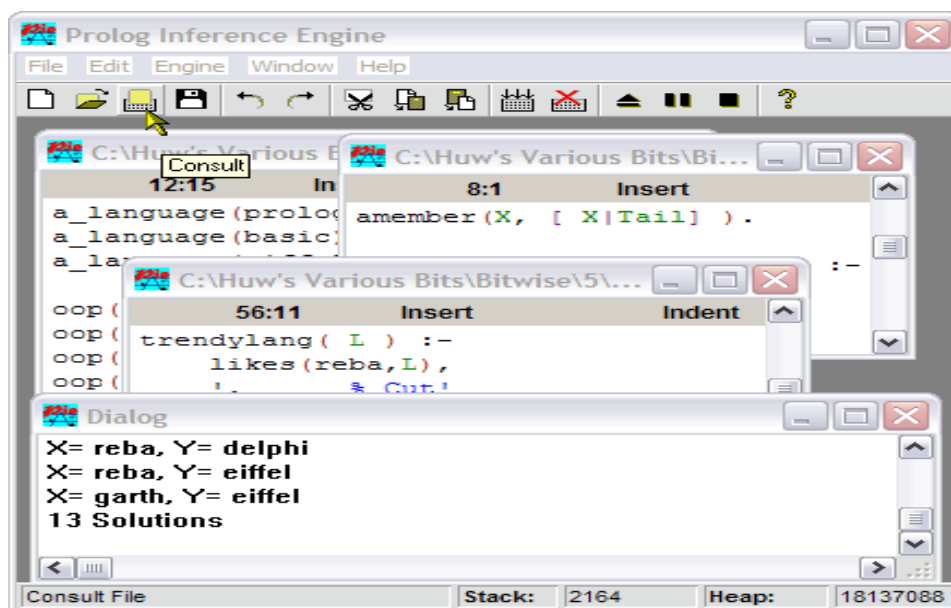
نصب و بکارگیری PIE Prolog

مزیت این مفسر سادگی و سهولت بکارگیری آن می‌باشد. این مفسر پرولوگ یک واسط کاربر چند پنجره‌ای را در اختیار می‌گذارد و ضمناً ویرایشگر آن رنگ‌های متفاوتی را بکار می‌گیرد، این کار برای نمایش هر چه بهتر کد برنامه با توجه نحو آن می‌باشد.

برای دریافت آن می‌توانید به لینک زیر مراجعه کنید.

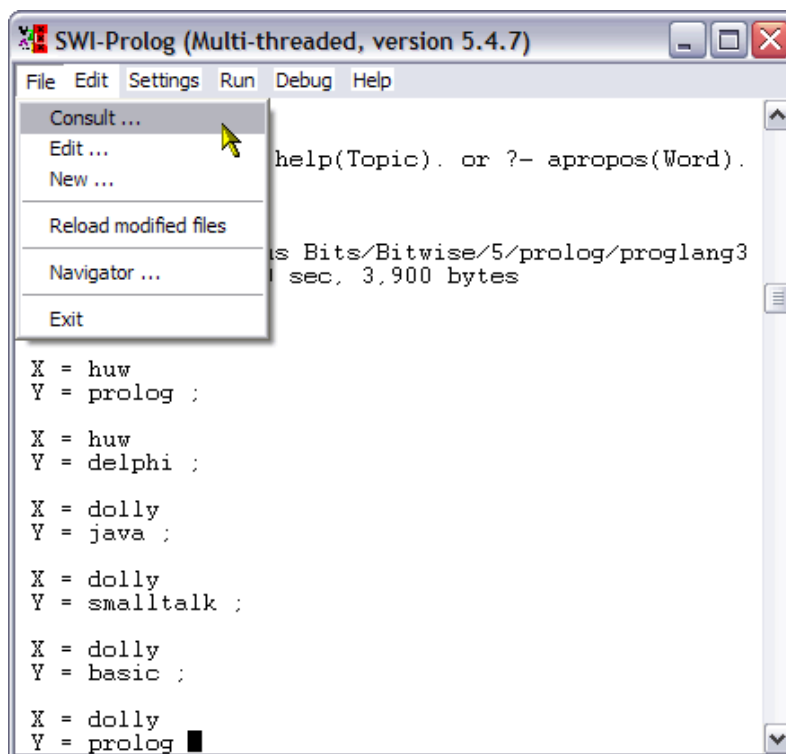
<http://www.bitwisemag.com/apps/pie/pie.zip>

فایل فشرده را در یک زیر فهرست دلخواه باز کنید و بعد از آن کافی است که برنامه `Pie.exe` را اجرا کنید. این مفسر در واقع برنامه‌هایی است که تحت ویژوال پرولوگ نوشته شده است. شکل زیر نمایی از مفسر `PIE` پرولوگ را نشان می‌دهد.



ویژوال پرولوگ یک مفسر و کامپایلر بزرگ با قابلیت‌های منحصر به فردی می باشد. بکارگیری آن نیاز به تجربه و دانش لازم در رابطه با برنامه نویسی تحت ویندوز، برنامه نویسی شیء گرا و مانند آن دارد. برنامه‌هایی که بطور معمول به زبان پرولوگ ارائه می شوند را نمی توان براحتی به همان فرم اولیه‌شان در ویژال پرولوگ اجرا نمود.

چنانکه اشاره شد، **PIE Prolog** برای شروع کار یکی از بهترین انتخابها می باشد. از طرف دیگر از آنجایی که این مفسر تنها تعداد معدودی از ساختارهای زبانی (گزاره‌های تعبیه شده) را شامل می شود و همچنین عدم وجود ابزارهای اشکال زدایی در آن، برای کسانی که قصد دارند قدم های بلندتری در پرولوگ بردارند مفسر توانمندتری مانند **SWI-Prolog** را توصیه می کنیم. بکارگیری **SWI-Prolog** به سادگی و براحتی بکارگیری **PIE Prolog** نمی باشد اما مزایای دیگری دارد. در حالیکه این نسخه فاقد یک محیط و ویرایشگر مجتمع برای ایجاد برنامه می باشد، محدوده وسیعی از گزاره‌های استاندارد را شامل می باشد و همچنین یک محیط گرافیکی مناسب برای اشکال زدایی نیز در اختیار می گذارد. ضمناً می توان با اضافه کردن ملحقاتی به آن، نقص آن در رابطه با سهولت کاربری را رفع نمود.



شکل بالا نمایی از محیط **SWI-Prolog** را نشان می دهد.

Prolog syntax

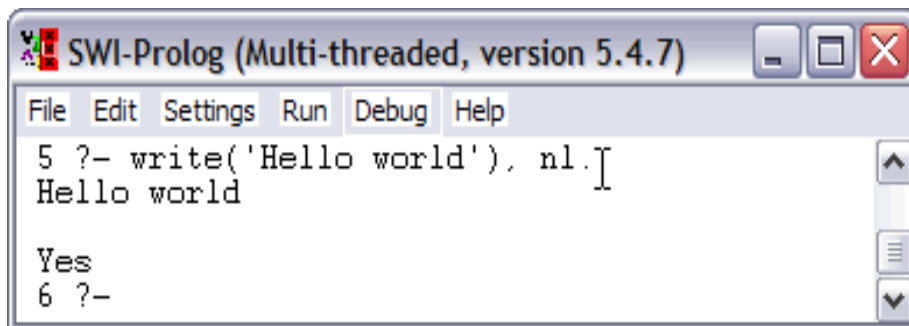
اولین برنامه به پرولوگ و بررسی آن

بهترین طریقه برای درک قابلیت‌های پرولوگ مشاهده‌ی عملکرد آن می‌باشد. در معرفی و بکارگیری یک زبان برنامه نویسی مرسوم است که در غالب موارد اولین برنامه که نوشته می‌شود برنامه‌های است که عبارت **Hello world** را نمایش میدهد. ما نیز در اینجا همین کار را می‌کنیم. در صورتیکه **SWI-Prolog** را بکار می‌گیرید جلوی علامت **اعلان** پرولوگ که بصورت **?-** می‌باشد فرمان زیر را وارد کنید.

```
?- write('Hello world'),nl.
```

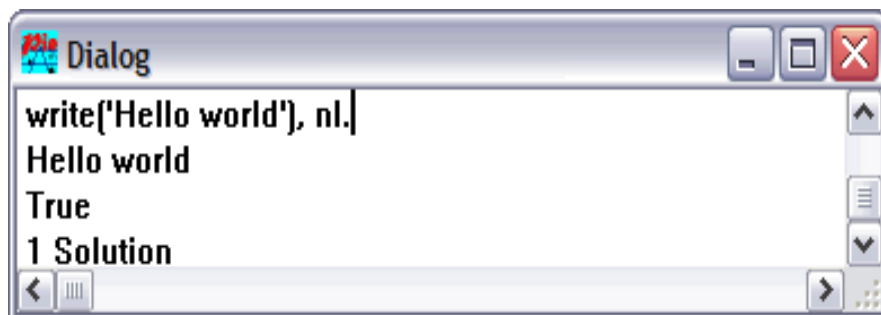
اکثر سیستم‌های پرولوگ مانند **SWI-Prolog** خروجی زیر را می‌دهند.

```
Hello world
Yes
```



توجه کنید، چنانکه نشان داده شده است در فرمان وارد شده، (بجز در رشته **Hello world**) حروف کوچک بکار گرفته شده اند و همچنین در انتها یک نقطه قرار داده شده است. چنانکه در شکل فوق دیده می‌شود، دستور مورد نظر، جلوی علامت اعلان **SWI** پرولوگ و در پنجره اصلی آن وارد شده است. سیستم آنرا در قالب یک پرس و جو می‌بیند، عبارت مورد نظر را چاپ می‌کند و به جهت گزاره استاندارد **nl** که به معنای **new line** می‌باشد، به ابتدای خط بعدی در صفحه خروجی می‌رود. بدنبال آن کلمه **Yes** را در خروجی چاپ نموده است که به معنای آن است که پرس و جوی ما را با موفقیت پاسخ داده است. در نهایت مجدداً علامت اعلان پرولوگ را نشان می‌دهد که به معنای آن است که سیستم آماده دریافت پرس و جوی بعدی است.

در صورتیکه از **PIE Prolog** استفاده می‌کنید فرمان را روی یک خط خالی وارد کنید. در **PIE Prolog** وضعیت بصورت زیر خواهد بود،



در این نسخه از پرولوگ، پرس و جوها در یک خط خالی و در پنجره محاوره وارد می‌شوند. چنانکه می‌بینید هیچ علامت اعلان در اینجا وجود ندارد و در خروجی نیز بجای **Yes**، کلمه **True** را چاپ می‌کند و بدنبال آن تعداد پاسخ‌های یافت شده (در اینجا ۱) را چاپ می‌کند.

در ادامه موارد زیر را بطور مجزا وارد کنید.

```
4 is 2 + 2.
```

```
4 is 2 + 3.
```

برای مورد اول پرولوگ پاسخ **Yes** یا **True** را می‌دهد در حالیکه برای مورد دوم پاسخ **No** را دریافت خواهیم کرد. توجه کنید که در مورد اول از عملگر **is** استفاده کرده‌ایم و نه از **=**. عملگر **is** موجب می‌شود که یک عبارت ریاضی مانند $2 + 2$ ابتدا ارزیابی و محاسبه شود، در حالیکه عملگر **=** در صورتیکه طرفین آن یکسان باشند مقدار **True** را می‌دهد.

بیاید نگاه دقیقتری به برنامه چاپ عبارت **Hello world** بیاندازیم.

```
write('Hello world'),nl.
```

عبارت **Hello world** در قالب یک آرگومان ورودی به گزاره **write/1** ارسال می‌شود. واژه گزاره در پرولوگ در ازای رویه، روتین و یا **قاعده** می‌تواند در نظر گرفته شود، همچنین می‌توان آنرا متناظر با تابع یا زیر برنامه در زبان‌های برنامه نویسی معمول در نظر گرفت.

کلیه گزاره‌های استاندارد که در خود مفسر تعبیه شده‌اند، در هر دو نسخه **SWI-Prolog** و **PIE Prolog** در قسمت **Help** آنها معرفی گردیده و عملکرد هر یک بیان شده است. در کتابها و دیگر منابع آموزشی، گزاره **write** غالباً بصورت **write/1** آورده شده است. در اینجا عدد ۱ به معنای آن است که گزاره مورد نظر فقط یک آرگومان دریافت می‌کند. گاهی نیز گفته می‌شود که **arity** آن گزاره ۱ می‌باشد. گزاره **nl** که برای رفتن به ابتدای خط جدید است بصورت **nl/0** بیان می‌شود که بدین معنی خواهد بود که این گزاره هیچ آرگومانی دریافت نمی‌کند و به بیانی **arity** آن صفر است.

آرگومانها بطور معمول دارای یک نوع مانند عددی، رشته‌ای و غیره می‌باشند. نسخه‌های معمول پرولوگ مانند **PIE** و **SWI** در رابطه با تعیین نوع آرگومانها سختگیر نیستند. این بدین معنی است که گزاره‌ها غالباً می‌توانند آرگومان‌هایی با نوعهای متفاوت را دریافت کنند. بعنوان مثال گزاره `write/1` می‌تواند آرگومان‌هایی از نوع عددی و همچنین رشته‌ای را بپذیرد. این مطلب را با وارد کردن فرمان زیر تجربه کنید.

```
write(5), write( ' plus '), write(8),write(' is ' ),
X is 5+8, write(X), nl.
```

این بار خروجی پرولوگ بصورت زیر خواهد بود،

```
5 plus 8 is 13
X = 13
```

توجه کنید که در بعضی از سیستم‌ها (مانند **SWI**) پس از دریافت پاسخ، بایستی کلید **Enter** را یک مرتبه وارد کنید تا سیستم مجدداً علامت اعلان را نشان دهد. حال بیایید نگاه مجددی به آنچه پرولوگ در خروجی نشان داد بیاندازیم. در خط اول رشته تعیین شده و مقادیر عددی مربوطه را چاپ می‌کند و در خط دوم مقدار آرگومان **X** را می‌دهد. در پرولوگ نام متغیرها با یک حرف بزرگ شروع می‌شود و نام گزاره‌ها همگی با حرف کوچک شروع می‌شوند. اگر آرگومانی با حرف کوچک شروع شود بعنوان یک نماد^۱ در نظر گرفته می‌شود. برای روشن شدن مطلب عبارات زیر را وارد کنید؛ دقت کنید که یک بار آرگومان گزاره‌ی `write` حرف بزرگ **X** و یک بار حرف کوچک **x** است.

```
X is 2, write(X),nl.
X is 2, write(x),nl.
```

در برنامه‌های پرولوگ بایستی به علامتهای نشان‌گذاری توجه کافی داشته باشیم. قبلاً لزوم قرار دادن نقطه در انتهای یک خط را گوشزد کرده بودیم. قرار دادن کاما بین بندها نیز بسیار مهم است و آن‌را به معنای درج «و» بحساب آورید. لذا عبارت زیر به این معنی خواهد بود که جمله `Hello world` را چاپ کن «و» علامت رفتن به ابتدای خط جدید را چاپ کن.

```
write('Hello world'),nl.
```

بطور معمول یک برنامه از ده‌ها، صدها و یا بیشتر خط برنامه تشکیل می‌شود. برنامه‌ها در قالب پروندها ذخیره می‌شوند و در موارد لازم بازیابی و یا اصطلاحاً خوانده می‌شوند. در **PIE** از **File** و بعد **Open** و در **SWI** از **File** و بعد **Edit** را بکار گرفته و برنامه‌های پرولوگ مورد نظرتان را باز کنید.

برنامه‌های پرولوگ معمولاً دارای پسوند **.pro** می‌باشند؛ البته در مواردی نیز پسوند **.pl** بکار گرفته شده

^۱Symbol

است. در اینجا فرض کنید پرونده های بنام `proglang.pl` داریم که شامل موارد زیر می باشد.

```
oop(delphi).
```

```
oop(java).
```

```
oop(smalltalk).
```

```
oop(eiffel).
```

```
likes(huw,prolog).
```

```
likes(huw,delphi).
```

```
likes(dolly,java).
```

```
likes(dolly,smalltalk).
```

```
likes(dolly,basic).
```

```
likes(dolly,prolog).
```

```
likes(tammy,java).
```

```
likes(tammy,eiffel).
```

```
likes(tammy,basic).
```

```
likes(dwight,prolog).
```

```
likes(reba,Language) :-
```

```
    oop(Language),
```

```
    not(likes(dolly,Language)).
```

```
likes(garth,Language) :-
```

```
    likes(reba,Language),
```

```
    likes(tammy,Language).
```

در شروع برنامه یک مجموعه ساده از واقعیت‌ها در رابطه با زبان‌های برنامه نویسی آمده است. چهار واقعیت اول بیان می‌دارند که زبان‌های یاد شده شیء گرا **Object Orientated Programming: OOP** می‌باشند.

`oop(delphi).`

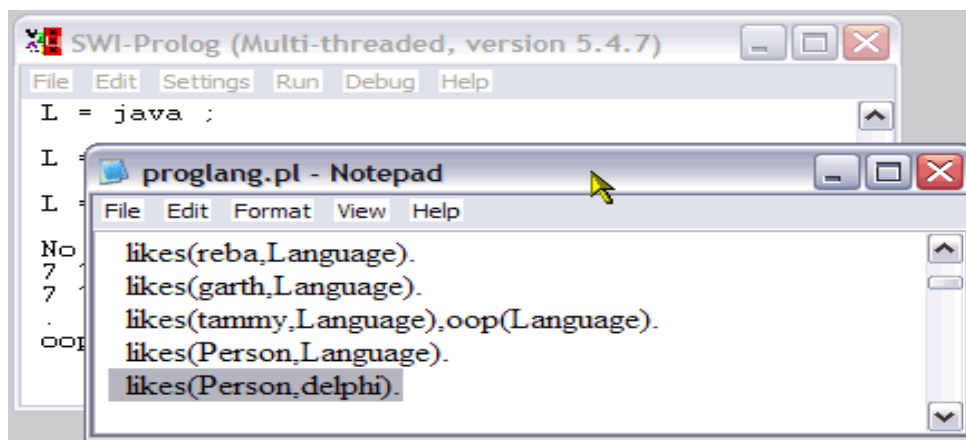
`oop(java).`

`oop(smalltalk).`

`oop(eiffel).`

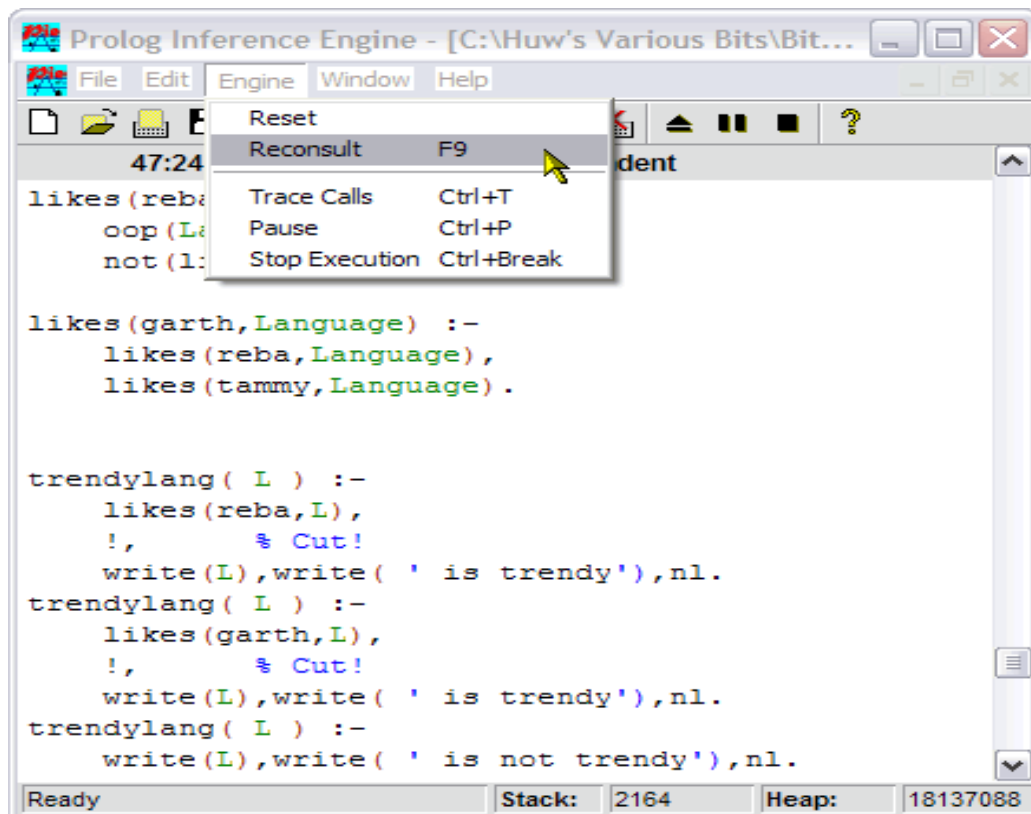
در این رابطه می‌توان پرس و جوهایی را از طریق علامت اعلان پرولوگ وارد کرد. البته قبل از اینکه بتوانید پرس و جویی را وارد کنید بایستی به سیستم پرولوگ بگویید تا کد برنامه را `consult` کند.

توجه داشته باشید که `edit` کردن یک برنامه با `consult` کردن یک برنامه دو چیز متفاوت می‌باشد. در اینجا ما از طریق پنجره اصلی `SWI-Prolog` و از طریق `File` و بعد `Edit` برنامه را که فرض کردیم نام آن `proglang.pl` می‌باشد را فراخوانی می‌کنیم. آنچه اتفاق می‌افتد این است که برنامه مورد نظر توسط برنامه `Notepad` برای ویرایش آورده می‌شود. پس از ویرایش فایل برنامه بایستی از خارج شویم و پس از آن با انتخاب `File` و بعد از آن `Consult` آماده وارد کردن پرس و جوهای مورد نظرمان شویم.



در `PIE` نیز با اندکی تفاوت روش کار به همین صورت می‌باشد. در `PIE` فرض بر این است که برنامه‌های پرولوگ

دارای پسوند **pro** می‌باشند؛ ضمناً این نسخه از پرولوگ یک ویرایشگر نیز در درون خود دارد که با توجه به نحوه برنامه اجزاء آنرا با رنگ‌های مختلف نشان می‌دهد. در اینجا برای رسیدن به **consult** از طریق منوی **Engine** اقدام می‌کنیم. در **PIE** پرولوگ می‌توانیم چند برنامه را در پنجره‌های مجزا باز کنیم و در هر لحظه هر کدام را که خواستیم **consult** کنیم.



اکنون در صورتیکه در **SWI** پرولوگ هستید با وارد کردن **Enter** به اعلان -؟ بروید؛ جلوی این اعلان (و یا در **PIE** روی یک خط خالی در پنجره محاوره) پرس و جوهای زیر را وارد کنید،

```
oop(smalltalk).
oop(basic).
```

خواهید دید که پرس و جوی اول موفق می‌شود (پرولوگ مقدار **Yes** یا **True** را بر می‌گرداند) در حالیکه پرس و جوی دوم شکست می‌خورد (پرولوگ مقدار **no solutions** یا **no** را بر می‌گرداند).

چنین نیست که به پرس و جوهایی محدود باشیم که جواب آنها **Yes/no** باشد؛ بلکه می‌توانیم پرس و جویی وارد کنیم که مثلاً کلیهٔ زبانهای شیء‌گرا را برایمان بیاورد. بیاد آورید که نام یک متغیر با یک حرف بزرگ شروع می‌شود؛ حال روی اعلان پرولوگ مورد زیر را وارد کنید،

oop(Language) .

پرولوگ اولین زبان که در برنامه آنرا شیء‌گرا معرفی کرده بودیم، یعنی **delphi** را به متغیر **Language** می‌دهد. در چنین وضعیتی می‌گوییم متغیر **Language** به **delphi** تعیین مقدار شده^۱ است. پس از آن **PIE** ادامه می‌دهد تا دیگر پاسخ‌های ممکن برای این پرس و جو یعنی **java, smalltalk** و **eiffel** را نیز بیابد. در صورتیکه از **SWI-Prolog** استفاده می‌کنید خواهید دید که فقط اولین پاسخ به فرم زیر نمایش داده خواهد شد،

Language = delphi

ضمناً علامت اعلان یعنی **?** - نیز دیده نمی‌شود. این وضعیت در تبعیت از بسیاری از نسخه‌های پرولوگ می‌باشد که ابتدا فقط یک پاسخ برای پرس و جو می‌یابند و نمایش می‌دهند. البته این گزینه را برای کاربر محفوظ نگه می‌دارند که بتواند ادامه جستجو برای پاسخ‌های ممکن دیگر را درخواست کند. اساساً به همین دلیل است که علامت اعلان نشان داده نمی‌شود (نشان داده شدن علامت اعلان به معنای این است که عمل جستجو و ارائه پاسخ به پرس و جوی قبلی به پایان رسیده است و سیستم آماده دریافت پرس و جوی دیگری است). به هر حال در اینگونه موارد می‌توانیم با وارد کردن **نقطه ویرگول** پاسخ ممکن بعدی را در خواست کنیم. در رابطه با مثال بالا اگر این کار را انجام دهیم پاسخ ممکن بعدی یعنی

Language = java

چاپ می‌شود، در صورتیکه کلید نقطه ویرگول را دو مرتبه دیگر فشار دهیم، نام دو زبان دیگر که بعنوان زبان‌های شیء‌گرا در برنامه مشخص شده بودند در خروجی چاپ می‌شود و نهایتاً علامت اعلان را نشان می‌دهد. اکنون به پنجرهٔ حاوی کد برنامه بروید. در **PIE** پرولوگ می‌توان کد برنامه را در پنجرهٔ ویرایش باز نگه داشت. اگر با **SWI** پرولوگ کار می‌کنید می‌توانید توسط **Notepad** براحتی به کد برنامه دسترسی داشته باشید. در کد برنامه که قبلاً نیز نشان داده شد همچنین دسته‌ای از واقعیت‌ها وجود دارند که بیان می‌دارند هر کس چه زبانی را دوست دارد. این واقعیت‌ها به فرم زیر بیان شده‌اند،

likes(dolly,prolog) .

واقعیت فوق در واقع رابطهٔ بین دو قلم دادهٔ **dolly** و **prolog** را بیان می‌دارد. این معادل این جملهٔ انگلیسی

¹instantiated

است که « `Dolly likes Prolog` ». این موضوع را می توان با وارد کردن پرس و جوی زیر بررسی نمود،

`likes(dolly,prolog)` .

چنانکه انتظار می رود پاسخ سیستم `Yes` یا `True` خواهد بود. در برنامه مان واقعیت های دیگری نیز در رابطه با `dolly` بیان شده است. اینکه او `Smalltalk` ، `Java` و `Basic` نیز دوست دارد.

در رابطه با پرولوگ نیز می بینیم که `Huw` و `Dwight` نیز آنرا دوست دارند. می توانیم با بکارگیری متغیرها پرس و جوهای با انعطاف بیشتری بسازیم و چیزهای مختلف را جستجو کنیم. بعنوان مثال پرس و جوی زیر را وارد کنید (توجه کنید که کلمه `Person` با حرف بزرگ شروع شود و کلمه `prolog` با حرف کوچک).

`likes(Person, prolog)` .

مجدداً خواهید دید که `PIE` همه پاسخ های ممکن را در یک مرحله نمایش می دهد، در حالیکه `SWI` ابتدا فقط مورد زیر را نمایش می دهد،

`Person = huw`

برای اینکه به `SWI` بگویید که به یافتن برای پاسخ های بیشتر ادامه دهد بایستی کلید نقطه ویرگول را فشار دهید. در این صورت دو مورد زیر را نیز نمایش می دهد.

`Person = dolly ;`

`Person = dwight ;`

برای اینکه همه زبان هایی که یک نفر به آنها علاقه مند است را بیابید، کافی است که در پرس و جوی مان نام فرد را بیاوریم و یک متغیر در محل مربوط زبان قرار دهیم، مانند:

`likes(dolly,Language)` .

قواعد در پرولوگ

در واقع برنامه نویسی در پرولوگ چیزی بیشتر از جستجو در لیستی از داده‌ها می‌باشد. توان اصلی پرولوگ متاثر از این است که در پرولوگ می‌توانیم رابطه‌های پیچیده‌ای را بیان کنیم. این رابطه‌ها به فرم **قواعد** بیان می‌شوند. در حالیکه یک **واقعیت** رابطه‌ای را بیان می‌کند که همیشه درست است، یک قاعده تنها در صورتی درست است که شرایط چندی برقرار باشند. لذا **"dolly likes prolog"** یک واقعیت را بیان می‌دارد در حالیکه جمله زیر یک قاعده می‌باشد.

"garth likes any language that both reba and tammy like"

برنامه‌ها از تعریف روال تشکیل شده‌اند؛ یک روال منبعی برای ارزیابی چیزی می‌باشد. در پرولوگ **"-"** به معنای **if** (اگر)؛ **"**، به معنای **and** ؛ و **"**؛"، به معنای **or** می‌باشد.

پرولوگ، فراخوانی‌های در یک پرس‌وجو را به صورت ترتیبی و از چپ به راست، نوشته شده‌اند، ارزیابی می‌کند.

واژگان شروع شده با یک حرف بزرگ با خط زیرین (_)، متغیر هستند.

گزاره‌ها و ثابت‌ها همیشه با یک حرف کوچک یا عدد شروع می‌شوند.

در پرولوگ دو نوع، عبارت داریم؛ یکی قانون‌ها، که دارای علامت **"-"** هستند؛ و دیگری، واقعیت‌ها، که دارای علامت **"**:-" نمی‌باشند.

قانون **"a:-b,c"**؛ یعنی، **"**اگر **b** و **c** هر دو درست باشند، آنگاه، **a**، درست خواهد بود."

هر عبارت، در پرولوگ، با نقطه (.) ختم می‌شود.

تعریف یک گزاره، در پرولوگ، تقریباً، با تعریف یک زیر برنامه برابر است و گزاره‌هایی که در بدنه‌ی شرط می‌آیند، تقریباً با فراخوانی زیر برنامه برابر هستند.

توضیح‌ها به وسیله‌ی پرولوگ پردازش نمی‌شوند؛ به عبارت دیگر، کاری بر روی آنها انجام نمی‌شوند.

مقایسه عبارت‌ها

پرولوگ دارای عملگرهایی برای مقایسه عبارت‌ها می‌باشد .

عملگر "="

این عملگر برای یکسان بودن دو عبارت، می‌باشد مثلاً، در عبارت $X=a$ ، اگر به جای a, X قرار دهیم عبارت، یکسان خواهد شد.

دو عبارت، در پرولوگ ، یکسان هستند، اگر برای هر متغیر موجود در آنها (دو عبارت)، موردی وجود داشته باشد که آنها (دو عبارت) را برابر نماید.

عملگر "=="

اگر بخواهیم بررسی کنیم که آیا دو عبارت باهم ، دقیقاً، برابر هستند یا نه از این عملگر، استفاده می‌نماییم.
مثال‌ها:

?- 3==3.

True

?- 3==X.

True

?- 3==0.

False

?- [a,b]==[a,B].

False

عملگر "/="

برای بررسی اینکه آیا دو عبارت نامساوی هستند یا نه کاربرد دارد. اگر دو عبارت نامساوی داشته باشیم حاصل true و اگر در عبارت باهم مساوی باشند حاصل false خواهد بود.

?- 3\=3.

false

?- 3\=X.

True

محاسبه

برای محاسبه از عملگرهای زیر استفاده می‌کنیم:

- + برای انجام عمل جمع ،
- - برای انجام عمل تفریق
- * برای انجام عمل ضرب
- ** برای انجام عمل به توان رساندن
- / برای انجام عمل تقسیم
- // برای بدست آوردن خارج تقسیم صحیح
- Mod برای بدست آوردن باقیمانده‌ی تقسیم صحیح

نکته : برای وارد کردن پرولوگ به محاسبه‌ی عبارت‌های محاسباتی، مثل جمع، ضرب، تقسیم و غیره، می‌توان از عملگر **is** استفاده کرد.

اگر عبارت زیر را در محیط پرولوگ بنویسیم:

?- Sol = 2+6.  Sol = 2+6

حاصل جمع را دریافت نخواهیم کرد. اما با استفاده از عملگر **is** حاصل جمع را دریافت خواهیم کرد.

?- Sol is 2+6.  Sol = 8

?- Sol is 10/3.  Sol = 3.3333

عملگر is

عملگر **is** موجود در پرولوگ دارای ۲ آرگومان می‌باشد. به دومین آرگومان به چشم یک عبارت محاسباتی، نگاه می‌کند و آن را با آرگومان اول، برابر قرار می‌دهد.

?- A is 2+3*4.
A=15

اولویت اول با * است.

?- 10 is 2*5.
True

?- X+4 is 12.
False

?- X is ,Y is X+2.
X=12
Y=14

نکته: عباراتی مانند (2+2) را می‌توان به صورت پیشوندی یعنی (2,2)+ نیز نوشت.

توجه کنید که **is** را با = اشتباه نگیرید. **X=Y** دارای معنی «آیا X، ممکن است با Y یکسان باشد» است.

?- X = 2+2.
True

?- 4 = (2+2).
False

مقایسه عبارت‌های محاسباتی

عملگر "=="

$E == R$ بررسی می‌کند که آیا مقادیر E و R باهم، برابرند یا نه.

?- $2+3*4 == (2+3)*4$.

False

پرانتهز دارای بالاترین اولویت است

?- $2*4+3 == (2*3)+4$.

True

عملگر "\="

$E \setminus R$ بررسی می‌کند که آیا مقادیر E و R نامساویند؟

?- $2+3*4 \setminus= (2+3)*4$.

True

پرانتهز دارای بالاترین اولویت است.

?- $2*4+3 \setminus= (2*3)+4$.

False

عملگر "<"

$E < R$ بررسی می‌کند که آیا مقدار E کوچکتر از R می‌باشد یا نه؟

?- $5 < 4$.

False

?- $4 < 5$.

True

همچنین عملگرهای ">" برای بررسی بزرگتر بودن، ">=" برای بزرگتر یا مساوی بودن و عملگر "<=" برای بررسی مساوی یا کوچکتر بودن نیز استفاده می‌شود.

نکته: در پرولوگ عملگر "<=" نداریم و در صورت استفاده با خطا مواجه خواهیم شد.

?- 3<=5.

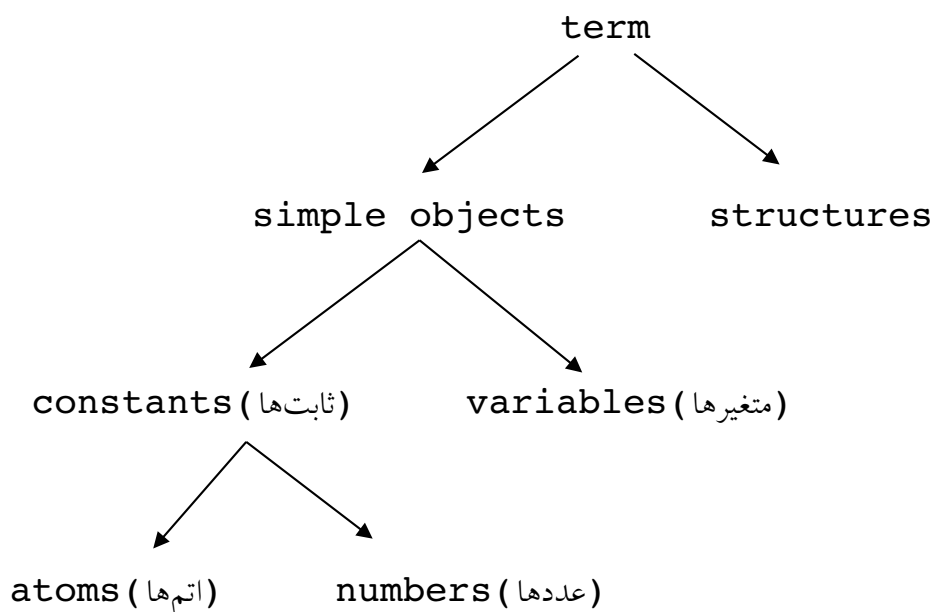
ERROR: Syntaxn error :Operator expected

ERROR: 3

ERROR: **here**

ERROR: <=5

یک برنامه پرولوگ از تعدادی ترم تشکیل یافته است.



Atom: برسه قسم هستند:

- رشته ای از حروف، ارقام و underscore که با یک حرف کوچک شروع می شود:

ali, x_123,y_ _ _ p,ali_Alavi

- رشته ای از حروف ویژه:

-, ?

- رشته ای داخل single quote

Numbers: اعداد می توانند صحیح و یا اعشاری باشند. چون پرولوگ اساساً برای کارهای symbolic و غیر

محاسباتی استفاده می شود زیاد اعداد اعشاری کاربردی ندارند.

Variables: متغیرها رشته‌هایی از حروف، ارقام و **underscore** هستند که با یک حرف **uppercase** شروع می‌شوند.

متغیرهایی که درون پرس‌وجوها هستند، به صورت سور وجودی (\exists) رفتار نمایند و متغیرهای درون شرط‌های برنامه، به صورت سورهای عمومی (\forall)، عمل می‌کنند.

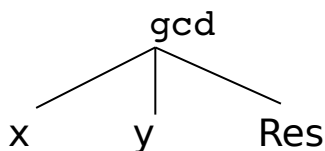
متغیر بی‌نام: در پرولوگ متغیر بی‌نام با **under-score** نمایش داده می‌شود. برای متغیرهایی که تنها در یک جمله به کار برده شده‌اند نیازی به ابداع نام جدید نیست.

hasachild(X) :- parent(X, _).

در رابطه‌ی "بچه‌ای دارد" مهم نیست که بچه چه کسی باشد، بنابراین از متغیر بی‌نام استفاده کردیم.

Structures: تمامی ساختارها در پرولوگ به صورت درخت هستند. این اشیا دارای یک سری اجزاء می‌باشند. این اجزا هم به نوبه خود می‌توانند **structure** باشند.

gcd(X, Y, Res).



بنابراین از نظر پرولوگ نوشتن **a*b** به صورت **prefix** یعنی ***(a,b)** کاملاً مجاز است

یک برنامه بسیار ساده پرولوگ را در نظر می‌گیریم. این برنامه شامل چهار کلاز است که در یک دیتابیس ذخیره شده‌اند.

```

likes( hossein, food).
likes( hossein, icecream).
likes( naeem, icecream).
likes( naeem, hossein).
  
```

?- likes(hossein, X) , likes(naeem, X).

Query که در انتها آمده می‌پرسد که آیا حسین چیزی را دوست دارد که نعیم هم آنرا دوست دارد؟

پرولوگ اولین ترم از **query** (پرس‌وجو) را می‌گیرد یعنی

likes(hossein, x)

و تلاش می‌کند که آنرا با یک کلاز در دیتابیس **unify** کند. پرولوگ با تولید جانشینی $\{X \leftarrow food\}$ موفق می‌شود که دو جمله

`likes(hossein, food)` و `likes(hossein, X)`

را بایکدیگر `unify` (متحد) کند. سپس پرولوگ جانشینی به دست آمده را به تمامی ترم های `query` اعمال می کند. سپس به سراغ دومین ترم می رود، که اکنون

`likes(naeem, food)`

شده است. این جمله با هیچ جمله دیگری در دیتابیس `unify` نمی شود.

بعد از هر شکست، پرولوگ `back track` می کند. یعنی به `unification` قبلی باز می گردد، که در این مورد `unify` کردن

`likes(hossein, X)` و `likes(hossein, food)`

است. اکنون پرولوگ تلاش می کند که اولین ترم `query` را با یک کلاز دیگر در دیتابیس `unify` کند،

`likes(hossein, icecream)`

این دو ترم با جانشینی $\{X \leftarrow \text{icecream}\}$ با یکدیگر `unify` می شوند، که به ترم دوم `query` یعنی

`likes(naeem, X)`

هم اعمال می شود تا

`likes(naeem, icecream)`

را ایجاد کند. ترم جدید با سومین کلاز دیتابیس قابل `unify` شدن است.

بعد از تمام شدن کار با تمامی ترم های `query` پرولوگ جانشینی های به دست آمده را خروجی می دهد، که در اینجا `X=icecream` است.

اصل مشهور Kowalski می گوید که `Algorithm = Logic + Control` هر الگوریتم دو قسمت دارد:

یک توصیف منطقی (`the logic`) و یک شرح از چگونگی اجرای این توصیف (`the control`). یک برنامه نویس منطقی خصوصیات جواب را بیان می کند، اما کنترل را به سیستم زیرین می سپارد. به وضوح این تیپ برنامه نویسی یک سطح بالاتر از برنامه نویسی های دستوری مثل `C` و یا پاسکال است، که در آنها برنامه نویس بایستی خود را درگیر مسائل اجرایی `statement` ها کند. منطق محاسباتی نیز حول و حوش این گونه مباحث می گردد.

توابع کتابخانه‌ای در پرولوج

number(X)

بررسی می‌کند که آیا X، عدد است یا نه.

مثال‌ها:

```
? - number(a).  
false.
```

```
? - number(100).  
true.
```

```
? - number(1000+10).  
false.
```

```
? - number('100').  
false.
```

Integer(X)

بررسی می‌کند که آیا X، عدد صحیح است یا نه.

مثال‌ها:

```
? - integer(a).  
false.
```

```
? - integer(192).  
true.
```

```
? - integer(-32679).  
true.
```

```
? - integer(-40000).  
true.
```

```
? - integer(3.14).  
False.
```

Var(X)

اگر **X**، یک متغیر باشد، **true** را بر می گرداند.

مثال ها :

? – var(j).
false.

? – var(j)
true.

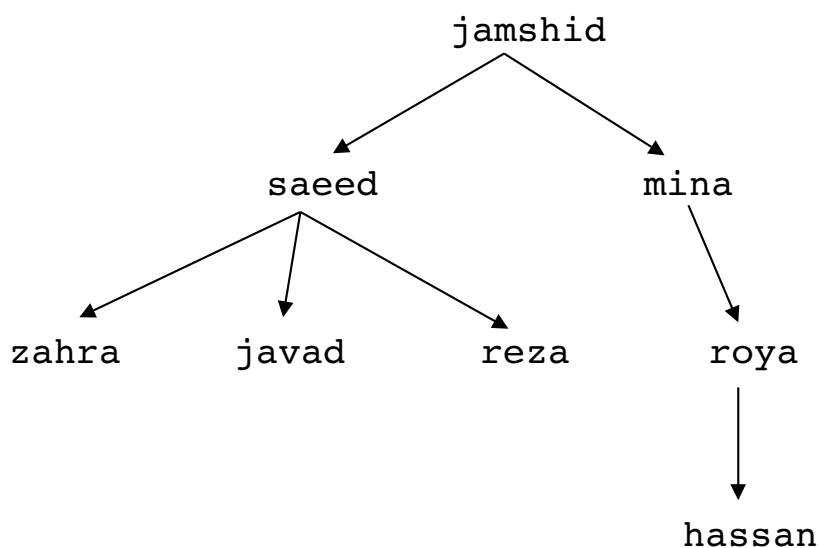
? – var(Soh).
true.

? – var('Soh').
true.

? – var([1,e,j]).
false.

A Classic Example:

یکی از مسائل کلاسیکی که معمولاً در ابتدای آموزش پرولوگ مطرح می کنند مساله شجره نامه خانوادگی می باشد.
برای مثال شجره نامه زیر را در نظر بگیرید:



این شجره نامه را به زبان پرولوگ توصیف می نماییم. برای این کار سه محمول پایه یعنی `male`، `female` و `parent` را اختیار می کنیم و با یک سری `fact` درخت را نمایش می دهیم.

```
Male(jamshid).
male(saeed).
male(javad).
male(reza).
male(hassan).
female(zahra).
female(mina).
female(roya).
parent(jamshid,saeed).
parent(jamshid,mina).
parent(saeed,javad).
parent(saeed,zahra).
parent(saeed,reza).
parent(mina,roya).
parent(roya,hassan).
```

پس از ساختن پایگاه دانش یک سری پرس و جو به سیستم می دهیم.

- Is hassan the parent of saeed?
Query: parent(hassan,saeed).
- Who is saeed's parent?
Query: parent(X,saeed).
- Who were the *children* of saeed?
Query: parent(saeed,X).

اکنون می خواهیم سوالاتی هوشمندانه تر از پرولوگ بپرسیم، از قبیل اینکه :

مادر X چه کسی است ؟

آیا Y خواهر X است؟

عموهای X را نام ببر ...

برای پرسیدن چنین سوالاتی بایستی تعریف مادر، خواهر و یا از این دست را به پرولوگ بدهیم.

یک راه بدیهی این است که برای تمامی افراد موجود در شجره نامه این رابطه را تعریف کنیم، که منطقی به نظر نمی رسد. بنابراین از یک گزاره‌ی منطقی استفاده می کنیم.

$$\forall X, Y : \text{parent}(X, Y) \wedge \text{female}(X) \rightarrow \text{mother}(X, Y)$$

کلاز متناظر در پرولوگ به این صورت خواهد بود:

`mother(X,Y) :- parent(X, Y), female(X).`

این نشان دهنده ی نوع دوم ساختارهایی است که در KB های به زبان پرولوگ می تواند وجود داشته باشد به قسمت سمت چپ (rule). علامت :- اصطلاحاً head و به قسمت سمت راست آن body می گویند.

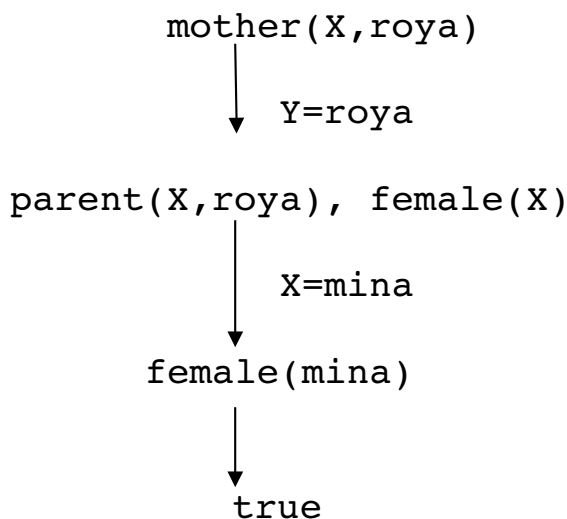
دقت اکید: توجه کنید که رابطه \rightarrow به صورت برعکس و یا **goal-driven** نوشته شده است.

با فرض اضافه شدن این **rule** به KB، فرض کنید که این سوال پرسیده شده است:

`mother(X, roya).`

و به عبارت دیگر: "مادر رویا کیست؟"

برای این پرسش درخت زیر را می توان بنا کرد.



قوانین بازگشتی (recursive rules) :

بازگشت، در هر زبانی، تابعی است که می تواند تا زمانی که به هدف خود برسد، خودش را فراخوانی نماید.
فرض کنید که می خواهیم رابطه predecessor را به KB اضافه کنیم.

```
predecessor(X,Z) :-
    parent(X,Z).
```

قانون فوق می گوید که X از پیشینیان Z است اگر X پدر Z باشد. این قانون تنها تا یک سطح رابطه predecessor را بیان می کند.

```
predecessor(X,Z) :-
    parent(X,Y),
    parent(Y,Z).
```

و این قانون تنها تا دو سطح.

```
predecessor(X,Z) :-
    parent(X,Y1),
    parent(Y1,Y2),
    parent(Y2,Z).
```

این روش پیاده سازی انتها ندارد، و طولانی است. راه بهتر پیاده کردن از روش بازگشتی است، یعنی بگوییم:

```
predecessor(X,Z) :-
    parent(X,Y),
    predecessor(Y,Z).
```

مثالهایی از برنامه نویسی در پرولوگ

تابع فاکتوریل

```
fact(0,1).
fact(N,R):- fact(N1,R1),N is N1+1,R is R1*N.

?- fact(4,R).
R = 24

fact(N,F):- fact1(N,1,F).

fact1(0,F,F).
fact1(N,X,F):- N is M+1, Y is X*N, fact1(M,Y,F).
```

Arithmetic mean

میانگین حسابی

```
?- start.

X= 4.
Y= 6.
R is 5
yes

start:- write('X= '),read(X),
        write('Y= '),read(Y),
        R is (X+Y)/2,
        write('R is '),write(R),nl.
```

دنباله فیبوناچی

The Fibonacci sequence $f(1), f(2), f(3), \dots$ is:

1, 1, 2, 5, 8, 13, 21, 34, 55.....

As you see the definition is easy to grasp:

$$f(1) = f(2) = 1$$

$$f(n) = f(n-2) + f(n-1), \text{ if } n \geq 3$$

Example:

```
?- fib(6,R).
   R = 13
```

```
fib(1,1).
fib(2,1).
fib(N,R):- N >= 3,N1 is N-1,N2 is N-2,
fib(N1,R1),fib(N2,R2),R is R1+R2
```

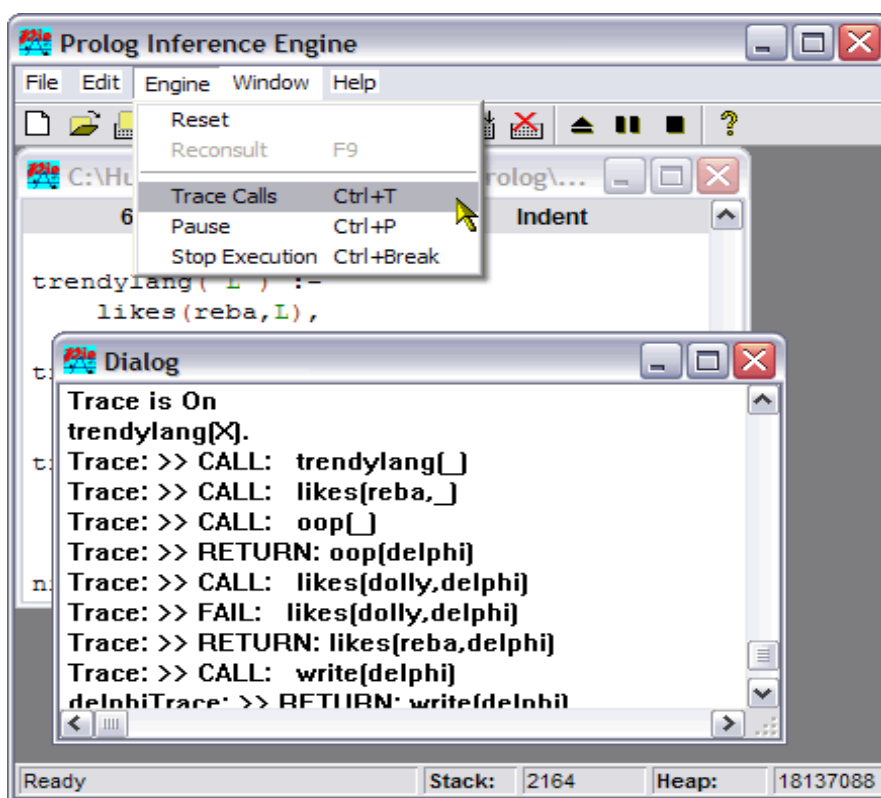
تابع توان

```
?- power(2,3,R).
   R = 8
?- power(2,0,R).
   R = 1
```

```
power(N,0,1):- !.
power(N,K,R):- K1 is K-1,power(N,K1,R1),R is R1*N.
```

اشکال زدایی

در صورتی که موقع اجرای برنامه، نتایج غیر معمول دریافت نمودید امکانی در مفسر پرولوگ وجود دارد که توسط آن می توان اجرای برنامه را بصورت قدم به قدم دنبال نمود. البته این امکان در **PIE** تا حدی اولیه و بدوی می باشد؛ اگر گزینه **Trace Calls** از منوی **Engine** را انتخاب کنید در پنجره محاوره، گزاره هایی که فراخوانی می شوند و همچنین مقادیری که بر می گردانند را می توانید مشاهده کنید

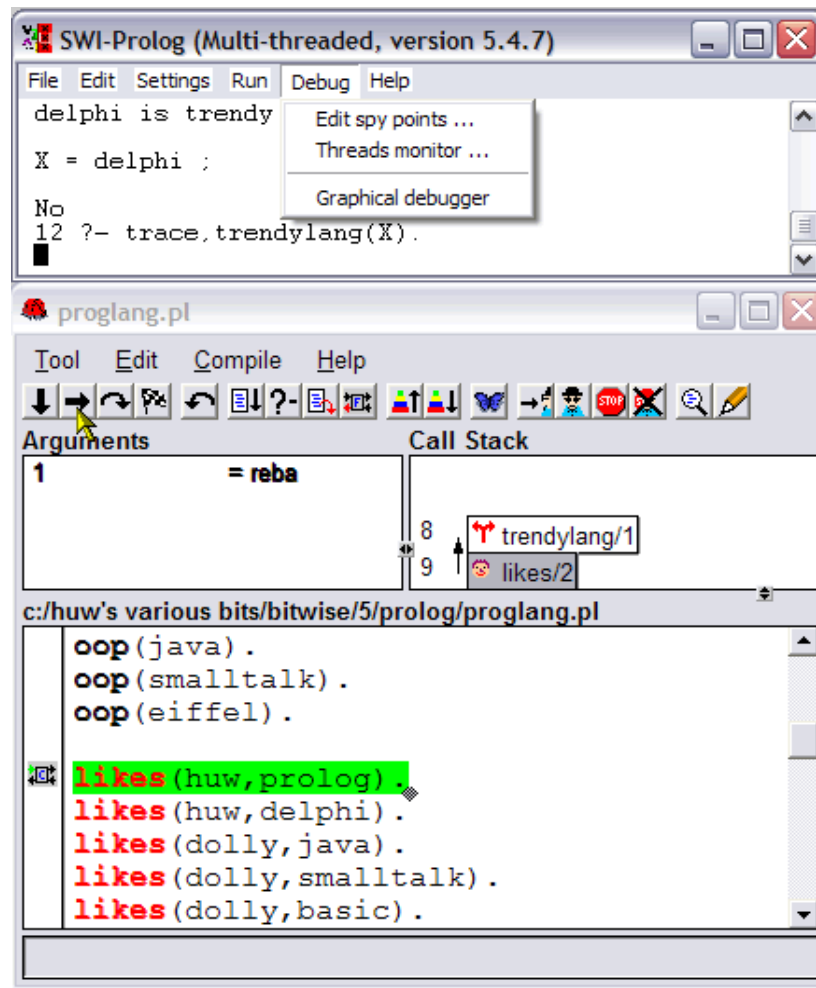


از طرف دیگر **SWI-Prolog** یک اشکالزدای توانمند تعاملی را در اختیار می گذارد. برای بکارگیری این امکان آن را از منوی **Debug** انتخاب کنید و بعنوان مثال عبارت زیر را وارد کنید:

`trace, trendylang(X).`

یک پنجره ظاهر می شود و که در آن هر خط از برنامه که در حال اجرا می باشد برجسته و نمایان نشان داده می شود.

ضمناً از طریق کلیدهایی که در بالای پنجره مورد نظر وجود دارد می توانید عمل دنبال کردن اجرای برنامه را دنبال کنید.



منابع :

1. <http://alain.colmerauer.free.fr/>
2. <http://www.anselm.edu/>
3. <http://www.csupomona.edu/~jrfisher/>
4. <http://www.swi-prolog.org/>
5. <http://www.visual-prolog.com/>

6. کتاب الکترونیکی هوش مصنوعی تألیف دکتر سهراب جلوه‌گر



(تنها عکس موجود از ایشان در سایت شان)

Alain Colmerauer

آلن کالمرار