باسمه تعالی

پرولوگ

Prolog Tool: SWI Prolog http://www.swi-prolog.org
Recommended resources:

- Computational Intelligence: A Logical Approach - David Poole http://www.cs.ubc.ca/spider/poole/ci.html
- Prolog Programming For Artificial Intelligence : Ivan Bratko
- Prolog Programming A First Course : Paul Brna (online)

پرولوگ در سال 1972 در دانشگاه مارسی و توسط Alain Colmerauer و همکاران ابداع شد. Prolog مخففی برای "PROgrammation en LOGique" می باشد. مستندات اولیه در زمینه پرولوگ همگی به زبان فرانسه هستند. پس از مدتی در گوشه و کنار دنیا مخصوصاً در اروپا و ژاپن زبان پرولوگ طرفدارانی پیدا کرد. گروهی که پرولوگ را ساختند اساساً یک گروه تحقیقاتی برای natural language understanding برای زبان فرانسه بودند. قصد و هدف اصلی آنها ساختن سیستمی بود که بتواند چنین کاری را انجام بدهد:

- User

Cats kill mice.
Tom is a cat who does not like mice that eat cheese.
Jerry is a mouse who eats cheese.
Max is not a mouse.
What does Tom do?

- Computer

Tom does not like mice that eat cheese.
Tom kills mice.

- User

What does Jerry eat?

- Computer

Cheese.

پرولوگ را یک زبان very high level می دانند.

Japan's new generation computer tech(ICOT) department has decided to adopt PROLOG as the official kernel language of the fifth generation computer system(FGCS) project which started in 1981. The primary goal of the FGCS project is to replace the traditional Von Neumann computers by smarter ones capable of reasoning, learning, associating, making inferences and decisions, and understanding speech, written text and pictures.

همچنین از پرولوگ در زمینه های زیر استفاده شده است:
Expert systems, natural language understanding, theorem proving, deductive DB, CAD Tool Design, compiler writing
به طور کلی پرولوگ بر روی دوبنیان ریاضی بنا شده است:
(pattern matching) unification و (backtracking) search آقای Woo ادعا می کند که به طور متوسط 55 الی 70 درصد زمان اجرای برنامه های به زبان پرولوگ مربوط به unification می شود.

Function Free First-order Logic

یک Well-formed formula در این منطق شامل موارد زیر می شود:

1- A set of constant symbols: { a, b, c,..}
2- A set of variable symbols: { X, Y, Z,..}
3- A set of predicate symbols : { f, g, h,..}
4- The connectives: $\wedge, \neg$
5- The quantifiers: $\exists, \forall$
6- The punctuations: (,),.

در این منطق یک سری قوانین استنتاجی و نیز مفهوم تساوی وجود دارند.
مثالی از چند WFF:

$\forall x, \exists y : y > x$
$\exists y, \forall x : y > x$
$\forall x : ((\exists y : 6 * y = x) \rightarrow (\exists y : 3 * y = x))$
If a number x is divisible by 6, then it is also divisible by 3.

تعریف term : به هر جزء تشکیل دهنده منطق یک ترم می گوییم مثل a ، f(g(h(a),a) . ترم ها را با حروف {s,t,u,..} نشان می دهیم.
تعریف substitution : یک substitution یک تابع از متغیرها به term ها است. substitution را معمولاً با نمادهای $\{\alpha, \tau, \Theta, ...\}$ نمایش می دهند. substitution سیگما را که در آن $\sigma(x)$ برابر g(h(a),a) بوده و $\sigma(y)$ برابر h(a) است می توان به صورت یک سری binding در داخل براکت نوشت. $\{x \leftarrow g(h(a), a), y \leftarrow h(a)\}$
برای مثال اگر در جمله $\forall x \exists y : likes(x, y)$ از substitution ی به صورت $\{x \leftarrow Ali, y \leftarrow Apple\}$ استفاده شود حاصل likes(Ali, Apple) خواهد بود.
بدیهی است که substitution ها می توانند با یکدیگر ترکیب بشوند، مثل $\sigma\theta(t)$. این تابع شرکت پذیر است $(\sigma\theta)\tau(s) = \sigma(\theta\tau)(s)$ ولی در همه حالات جابه جایی پذیر نیست.
Unification: دو ترم s و t اصطلاحاً unifiable هستند اگر یک جانشینی σوجود داشته باشد به طوری که $\sigma(s) = \sigma(t)$.

مثال 1- دو ترم f(X,Y) و f(g(Y,a),h(a)) یکسان پذیر هستند.
ومی توان برای آنها جانشینی زیر را در نظر گرفت: $\{X \leftarrow g(h(a), a), Y \leftarrow h(a)\}$

مثال 2- دوترم f(X,a,b) و f(c,Y,X) یکسان پذیر نمی باشند.
مثال 3- دو ترم X و f(X) یکسان پذیر نمی باشند ولی infinitely unifable هستند.
$\sigma = \{x \leftarrow f(f(f(...)))\}$

John Alan Robinson ، یکی از طلایه داران Logic Programming در سال 1965 مقاله ای به چاپ رساند که در آن مساله یکسان سازی را به صورت فرمال بیان کرد وسپس 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.
یک برنامه بسیار ساده پرولوگ را در نظر می گیریم. این برنامه شامل چهار کلاز است که در یک دیتابیس ذخیره شده اند.

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

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

Query که در انتها آمده می پرسد که آیا حسین چیزی را دوست دارد که نعیم هم آنرا دوست دارد؟ پرولوگ اولین ترم از query را می گیرد likes(hossein, X) و تلاش می کند که آنرا بایک کلاز در دیتابیس unify کند. پرولوگ با تولید جانشینی $\{X \leftarrow food\}$ موفق می شودکه دو جمله likes(hossein, food) و likes(hossein, X) را بایکدیگر unify 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 backtrack می کند. یعنی به unification قبلی باز می گردد، که در این مورد unify کردن likes(hossein, X) با likes(hossein, food) است. اکنون پرولوگ تلاش می کند که اولین ترم query رابا یک کلاز دیگر در دیتابیس unify کند، که likes(hossein, icecream) است. این دو ترم با جانشینی $\{X \leftarrow icecream\}$ 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 این جا X=icecream است.

The motivation for logic programming is to let programmers describe *what* they want separately from *how* to get it.

اصل مشهور Kowalski می گوید که Algorithm = Logic + Control. هر الگوریتم دو قسمت دارد: یک توصیف منطقی (the logic) و یک شرح ازچگونگی 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 اجرایي statement ها کند. منطق محاسباتی نیز حول و حوش این گونه مباحث می گردد.