返信 おすすめ メッセージ 1 / 11

投稿者 : ダイエットパンダ (元のメッセージ) 投稿日時 : 2006/05/28 22:37

題 : 論理プログラミング (Prolog)に挑戦！

動機 :
以前別の掲示板で、第五世代コンピュータの話題に話が及び、
せめて先人の偉業に敬意を表し、話半分でなく実地に手を動かして試してみよう、ということではじめてみます。

目的、目標 :
具体的には、下記の２点が目標（ゴール）です。
(1)先日、これも別の掲示板で話題になった、ゲーデルの不完全性定理にからみ、決定不能な命題（村の理髪師のパラドックスなど）を、Prologの処理系がどう扱うか？
(2)適用する推論ルールをどのように選択するのか？（ルール間の連想機能があるのか？何か優先順位付けがあるのか？あるいは絨毯爆撃的にやるのか？）

手段 :
昔(1982年頃)と違い、いまどき本屋の計算機のコーナーに行っても、Prolog言語や論理プログラミングの本はないかもしれませんが、当時とちがい実際に動く処理系が無料で手に入る時代であるし、私も初心者なので、実地に試していって、道草やら遠回りしながらも、ゴールに近づきたいと思っています。
m(_ _;)Y

最初の返信 前へ 2-11 通を表示 : 総返信数 11 通 次へ 最新の返信

返信 おすすめ メッセージ 2 / 11

投稿者 : ダイエットパンダ 投稿日時 : 2006/05/28 23:24

Prolog処理系の選択 :

とりあえず使えそうなProlog処理系を探してみましたところ、
Gnu-Prologというのがありましたのでこれを使うことにします。そのほかにもあれこれ探したのですが、下記の理由で今回は見送りました。

(1)cu-Prolog
http://www.icot.or.jp/ARCHIVE/Museum/IFS/abst/009-J.html
第五世代コンピュータプロジェクトのアーカイブから、みつけたのですが、
UNIXマシンがターゲットで(Windowsでは動かすのが大変そう)、
私の手持ちのLINUXで、makeしたが、コンパイルエラーとなったので、深入りせず見送り･･･
ソースはぜひ一度読んでみたい。

(2)Visual Prolog (Personal Edition : free)
http://www.visual-prolog.com/vip6/Download/default.htm
ここのページの「Simtel 」または「Tucows」からダウンロード可能のようです。
まちがって商用版をダウンロードしないよう注意！
vip63.zipというファイルが14.3MBくらいとかなりでかい！
zip中身のreadmeを覗くと下記のことがわかりました。
対象OS＝Windows ME/NT/2000/XPとのことであり、しかもグラフィカルなIDE環境で、オブジェクト指向な仕組みをサポートしている。
私としては、Windows/Linux両方で動き、かつ、習得が面倒なGUI/IDEでなく、さくっと動くコマンドラインインタープリター版がほしいので、これも見送り･･･
※当然のことながら、学習用のPersonal Editionなので、使える機能・ライブラリに制限がある。

(3)K-Prolog
http://www.kprolog.com/
個人使用ライセンスでも、媒体なしで8400円とのことで、これもパス･･･
漢字機能や、日本語マニュアルはなかなか魅力的！

(4)The Gnu Prolog
http://pauillac.inria.fr/~diaz/gnu-prolog/
GNU Prolog is a free Prolog compiler with constraint solving over finite domains developed by Daniel Diaz.
「free」です。さすがは、Free Software Foundation！
対象OSも、Redhat系Linux/FreeBSD/Windows/Unix/そのほかいろいろ
で、私はこういうのがいいと思います。
自宅のLinux/Winりょうほうともさくっとインストールできて動きました！
漢字がつかえないし、マニュアルは英語ですが、今回はちょっとためせればいいので、問題なしとします。(英語のマニュアルなんてのは慣れればむつかしくないです、そう思い込んで、頑張ります！(^^)たぶんゆっくり読めば十分判ります！)

ダウンロードは下記参照:
バイナリはこれを参照のこと。
binary RPM (compiled under under ix86 / GNU/Linux).
Win32 auto-install setup (compiled under ix86/windows XP).
他バージョンもあり。
Other versions:

ソースもダウンロード可能:
the main source distribution gprolog-1.2.16.tar.gz.

※次回は、実際にダウンロード、インストールして、簡単なところを動かしてみます。
Y(^o^)Y

返信 0 ポイントのおすすめ メッセージ 3 / 11

投稿日時 : 2006/05/31 0:09

このメッセージは投稿者によって削除されました。

返信 おすすめ メッセージ 4 / 11

投稿者 : ダイエットパンダ 投稿日時 : 2006/06/01 2:58

ちょっと誤記修正しました！（ひとつ前のカキコは削除させていただきましたm(_ _;)m)

一気に、目標(1)の結論を以下に示します。(ほぼ予想通りの結果となりました(^ ^;))
※目標(1)：決定不能な命題（村の理髪師のパラドックスなど）を、Prologの処理系がどう扱うか？

結論：（少なくともGNUの)Prolog処理系は、
①ルールとファクト（事実）の「定義（プログラム読み込み）」の時点で矛盾を検出しない。
②実行しても、矛盾（自分で髭を剃る←→剃らないを無限ループ）を検出しない。

考察：実用上において、うっかり、矛盾するルールやファクト（事実）を定義することもあると思うが、
これを検知できずにシステムリソースを食い尽くすまで走り続けるのはまずい。
しかし、プログラミングの立場から言うと、これは無限ループを引き起こす単なるバグであり、
何がしかの方法で回避しなければならない。

今後の課題：
①システムが矛盾を検知しない以上、プログラミング側にて、矛盾していることを検出する仕組みを
入れられないか？（第五世代のアーカイブのページなどになにか報告があるかもしれない。
ひょっとして、あの推論エンジンは、相反する命題が発生しているのを並列バックグラウンド検知
していたのかも？）
※定理の証明システムに、何か報告があるかもしれない。
※現役の学生・院生の方は、ひょっとして、この手の扱い方をご存知かもしれません。
教えてほしい・・・

以下は調査・実行の記録ですので、興味のある方は、ごらんください。
※使用方法を先に紹介する予定でしたが、結果を先に示してしまいました。後ほど、ダウンロードとインストール、ヘルプの見方、参考ホームページ、などを後追いで記載の予定です。

１．プログラム：下記の通り。

```
jibunde_higesoru(murabito1).        村人１は自分で自分の髭を剃る。
                                      （ファクト：事実）
jibunde_higesoru(tokoya):-          床屋が（自分の）髭を剃るとは→（次行に続く）
    tokoyaga_higesoru(tokoya).        文字通り、床屋は床屋自身の髭を剃るということである。
                                      （床屋に限定したルール）
jibunde_higesoranai(murabito2).     村人２は自分で自分の髭を剃らない。
                                      （ファクト：事実）
jibunde_higesoranai(tokoya):-       床屋が（自分の）髭を剃らないとは、→（次行に続く）
    tokoyaga_higesoranai(tokoya).     文字通り、床屋は床屋自身の髭を剃らないということである。
                                      （床屋に限定したルール）
tokoyaga_higesoru(X):-               床屋がある人Ｘの髭を剃るとは、→（次行に続く）
    jibunde_higesoranai(X).           ある人Ｘは、自分で自分の髭を剃らないということである。
                                      （一般ルール）
tokoyaga_higesoranai(X):-           床屋がある人Ｘの髭を剃らないとは、、→（次行に続く）
    jibunde_higesoru(X).              ある人Ｘは、自分で自分の髭を剃るということである。
                                      （一般ルール）
```

２．プログラムをProlog処理系に読み込む。(下記の通り、正常に実行された。特に、矛盾を検出などしている様子はない。)

```
| ?- ['c:/GNU-Prolog/rihatsushi.pl'].
compiling c:¥GNU-Prolog¥rihatsushi.pl for byte code...
c:¥GNU-Prolog¥rihatsushi.pl compiled, 10 lines read - 1074 bytes written, 15 ms
yes
```

３．プログラムを実行する。具体的に、村人１、村人２、床屋について、床屋が髭を剃るか質問する。

```
| ?- tokoyaga_higesoru(murabito1).     村人１の髭を、床屋が剃るか？
no                                     NO.（剃らない。村人１は自分で自分の髭を剃るから。）
| ?- tokoyaga_higesoru(murabito2).     村人２の髭を、床屋が剃るか？
yes                                    YES.(剃る。村人２は自分で自分の髭を剃らないから。）
| ?- trace.                            ※トレースをONにして、床屋自身のケースを追いかけてみる。
The debugger will first creep -- showing everything (trace)
yes
{trace}
| ?- tokoyaga_higesoru(tokoya).          床屋自身の髭を、床屋が剃るか？
      1    1  Call: tokoyaga_higesoru(tokoya) ?       床屋自身の髭を、床屋が剃るか？
      2    2  Call: jibunde_higesoranai(tokoya) ?     ルールにより、自分の髭を自分で剃らない。
      3    3  Call: tokoyaga_higesoranai(tokoya) ?      従って床屋が床屋の髭をそらない。
      4    4  Call: jibunde_higesoru(tokoya) ?          ルールにより、自分の髭を自分で剃る。
      5    5  Call: tokoyaga_higesoru(tokoya) ?           以下同様・・・
      6    6  Call: jibunde_higesoranai(tokoya) ?
      7    7  Call: tokoyaga_higesoranai(tokoya) ?
      8    8  Call: jibunde_higesoru(tokoya) ?
      9    9  Call: tokoyaga_higesoru(tokoya) ?
     10   10  Call: jibunde_higesoranai(tokoya) ?
                              ・・・（以下同様）・・・
```

※最後は、スタックオーバーフローして、異常終了する。

※トレースをはずして(notrace.)実行すると即座に異常終了する。
以上

「目標(2)適用する推論ルールをどのように選択するのか？」については、上記のように、トレースの使い方がわかったので、これを利用して、ルールの適用優先順位などを調べていきたい。

今回は以上です。ここまで読んでくださったかた、ありがとうございます！m(_ _;)m

返信 おすすめ メッセージ 5 / 11

投稿者：ダイエットパンダ 投稿日時：2006/06/01 23:11

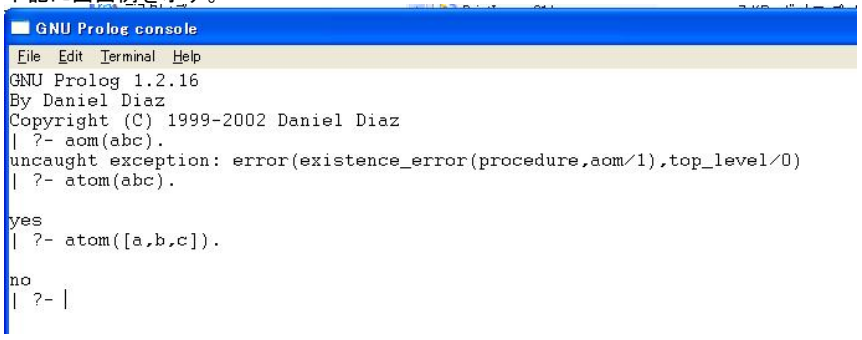
題：GNU-Prologのダウンロード、インストール、起動と終了。
（今回は、Windows版を例にします。）

(1)下記のサイトを、ブラウザで表示させる。
http://pauillac.inria.fr/~diaz/gnu-prolog/

(2)下記の部分の文字をクリックして、GNU-Prolog実行モジュールをダウンロードする。
Win32 auto-install setup (compiled under ix86/windows XP).

(3)setup-gprolog-1.2.16.exeというモジュールがダウンロードできていることを確認したら、それを実行して、自動インストールさせる。途中、下記のような質問がでるが、下記のように応答して先に進む。
・発行元を確認できませんでした～ →実行ボタンをクリック。
・This will install GNU Prolog. Do you wish to continue? →「はい」をクリック。
・Welcome to the GNU Prolog install wizard. →NEXTボタンをクリック。
・Select Destination Directory → C:¥GNU-Prologなどとデフォルト表示されているので、そのまま、NEXTボタンクリック。
・Select Start Menu Folder → GNU Prologなどとデフォルト表示されているので、そのまま、NEXTボタンクリック。
・Select Additional Tasks → Create Desktop iconにチェックがついているので、デスクトップにアイコンを作ってもよいならば、そのままNEXTボタンクリック。
・Ready to install → installボタンをクリック。
・Launch GNU Prolog→チェックをつけたままにして、Finishボタンをクリック。
以上を終わると、下記のようなウィンドウが表示される。

ちょっと、動作させてみる。(画面例は後述。)
atom(abc).と入力(最後のピリオドが必要です。)してリターンキーをうつと、
yesと返ってくる。
※「abc」はこれ以上分割できないアトミックなシンボルなので、atom()と述語に対し、yesを返す。
さらに、
atom([a,b,c]).と入力（最後のピリオドが必要です。）してリターンキーをうつと、
noと返ってくる。
※「[a,b,c]」はリストといわれるもので、のちに別項の例でみるように、分割可能なので、アトミックでないから、noを返す。
なお、まちがえて、aom(abc).などと入力すると、「uncaught exception:～」などと言ってくるので、落ち着いて、入力ミスを確認し、正しく入力しなおす。
下記に画面例を示す。



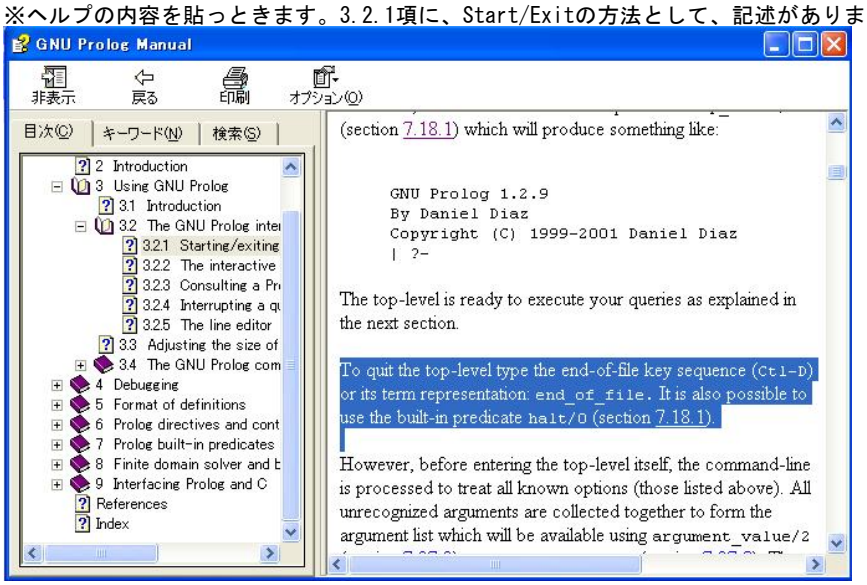
終了するには、以下のいずれかの方法がある。
・end_of_file. と入力してリターンキーを打つ。（最後のピリオド必要です！）
・Fileメニューから、Exitを選ぶ。
・Ctrl+Dを打つ。(Ctrlキーを押しながら、Dを入力する。これは、end_of_file.を入力することに相当する、とhelpに記載がある。)

再度起動するには、以下の方法などがある。
・デスクトップのGNU-Prologのアイコンをクリック起動する。
・C:¥GNU-Prolog¥BIN¥gprolog.exeをクリック起動する。
・Windowsのスタートメニューから、Gnu Prologをたどってそれを選択する。

とりあえず、以上。次回は、簡単な例で、非手続き型言語とはどういうことか、をみてみる。

ここまでよんで下さったかた、ありがとうございましたm(_ _;)m

※ヘルプの内容を貼っときます。3.2.1項に、Start/Exitの方法として、記述があります。



返信 おすすめ メッセージ 6 / 11

投稿者：ダイエットパンダ 投稿日時：2006/06/01 23:41

題：Prologのリスト処理にみる、非手続き型言語の意味。

Gnu-Prologを起動して、下記を入力します。とにかく、最後はピリオド「.」を打ちます！

```
append([a,b],[c,d],X).
```

すると、下記のように応答が返ってきます。

```
X = [a,b,c,d]
yes
```

これは、述語 append(リスト１、リスト２、変数X) の下記の意味を把握すれば、納得できる。
「リスト１とリスト２をつなげた(concatenate)ものが、変数Xの表すものに等しい」
上記の場合は、リスト１，２をつなげた結果である[a,b,c,d]が変数Xにマッチして、
X = [a,b,c,d]を返して、yes（述語が真)を返しているものである。

意味的に考えると、従来の手続き型言語のように、[a,b]と[c,d]をつなげた結果をXに入れろ、
という意味に近く感じられる。
アナロジーで言うと、
２＋２＝□
の□を求めよ、という計算問題のイメージですね！単に、２＋２を計算するだけ、という感じ。

では次に、下記を入力する。

```
append(X,Y,[a,b,c,d]).
```

すると、今度は、下記のように応答を返してくる。
（「;」は自分で入力してリターンキーを打ってください）。

```
X = []
Y = [a,b,c,d] ? ;
X = [a]
Y = [b,c,d] ? ;
X = [a,b]
Y = [c,d] ? ;
X = [a,b,c]
Y = [d] ? ;
X = [a,b,c,d]
Y = [] ? ;
no
```

これは、意味的に考えると、変数 X および Y があって、つなげた結果が [a,b,c,d]になるような
X,Yに何があるか、を自動的に順々に列記している。
ケース１：X=[] とY=[a,b,c,d]のつなげた結果が、[a,b,c,d]
ケース２：X=[a] とY=[b,c,d] のつなげた結果が、[a,b,c,d]
ケース３：X=[a,b,] とY=[c,d] のつなげた結果が、[a,b,c,d]
ケース４：X=[a,b,c] とY=[d] のつなげた結果が、[a,b,c,d]
ケース５：X=[a,b,c,d]とY=[] のつなげた結果が、[a,b,c,d]
（最後はマッチするケースがなくてnoを返していると思われる。ごめんなさいあとでよく調べます。）

この場合、何も手続き（すなわち、アルゴリズム、あるいは、プログラム）を与えていないのに、
述語（あるいは関係、言明といってもいい）append(X,Y,[a,b,c,d])を与えるだけで、
システムが自動的に、可能なケースを応答してくるという、まさに、非手続き型言語の
みごとな例だと私は思いました！

アナロジーで言うと、

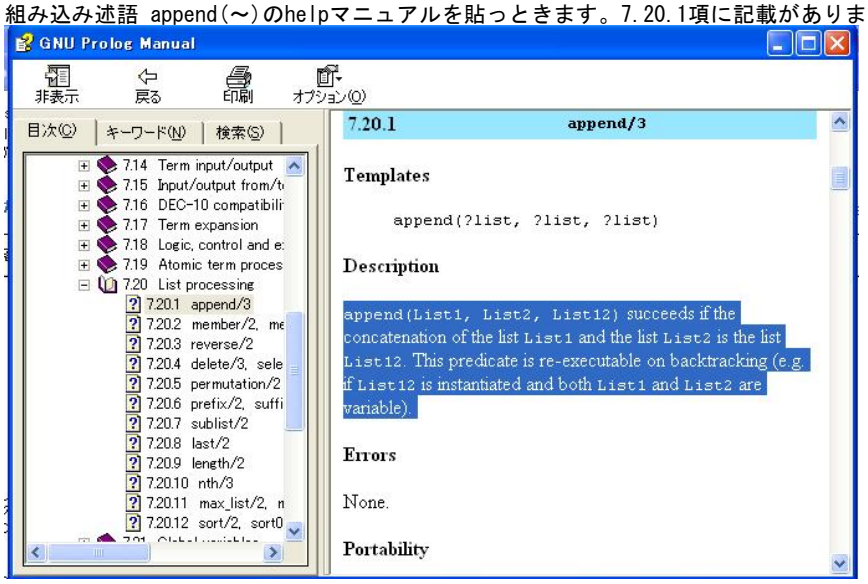
□＋□＝４
となる□の中身を求めよ、という問題に近い感じですね！
余談ですが、イギリスあたりの小学生の問題には、この手の、逆向き計算の問題があるとかいう噂を聞いたことがある・・・日本の小学校の場合は、２＋２＝□という問題をよくみかけます。
どちらも大事ですが、視点を変える、ということも大事だなぁと思います。

実行リストを下記に貼っておきます。

```
Copyright (C) 1999-2002 Daniel Diaz
| ?- append([a,b],[c,d],X).
X = [a,b,c,d]
yes
| ?- append(X,Y,[a,b,c,d]).
X = []
Y = [a,b,c,d] ? ;
X = [a]
Y = [b,c,d] ? ;
X = [a,b]
Y = [c,d] ? ;
X = [a,b,c]
Y = [d] ? ;
X = [a,b,c,d]
Y = [] ? ;
no
| ?-
```

今回は、組み込み述語（システムに最初から定義済みの述語）appendを例にとりましたが、
次回は、自分でPrologプログラムを入力する例を取り上げます。

組み込み述語 append(～)のhelpマニュアルを貼っときます。7.20.1項に記載があります。



返信 おすすめ メッセージ 7 / 11

投稿者：ダイエットパンダ 投稿日時：2006/06/02 0:04

題：参考文献、参考サイト

ちょっと横道にそれます。自宅の本の箱をひっくり返せば見つかると
思っていた、Prologの文法の本が、実は古本屋にうりはらってて、
ありませんでした・・・

それで、参考になるサイトを下記に貼っておきます。
東京大学や京都大学の授業・演習の内容がとても参考になります。
（まだよく読んでなくて、理解もおぼつかないけれど、本気でPrologやろうと思ったら、何をやらねばならないか、の指針を示していると思います。）

情報科学科 情報科学実験I Prolog演習
http://www-tsujii.is.s.u-tokyo.ac.jp/~yoshinag/prolog-enshu/
この中の、下記ページがありがたいです。
講義資料　pdf　の文書、第1回～第7回
http://www-tsujii.is.s.u-tokyo.ac.jp/~yoshinag/prolog-enshu/#resume

Prolog演習、自然言語処理の基礎
http://www.kuis.kyoto-u.ac.jp/isle/le4-lang/lang.html

K-Prolog Compiler Version 5.1　オンラインマニュアル
http://www.kprolog.com/doc/ja/index.html
この中の、Prolog言語（項の定義、プログラム、プログラムの動作）が、ありがたい。
基本文法です。

http://www.kprolog.com/doc/ja/language.html

第五世代コンピュータプロジェクト・アーカイブス
http://www.icot.or.jp/ARCHIVE/HomePage-J.html
この中の、フリーソフトウェアー覧、
http://www.icot.or.jp/ARCHIVE/Museum/IFS/field-J.html
第五世代コンピュータ・プロジェクト 最終評価報告書、
http://www.icot.or.jp/ARCHIVE/Museum/FinalReport/final.html
などが面白いと思う。
評価報告書には、未解決だった群論の問題の一部を自動証明できた、という記載があり、興味深い。

※私も学生時代、宿題にだされたtwo-twisted-semigroupの問題を、手計算で5乗までやって項の表現が
一致する部分をみつけだして無事宿題提出して、周りが舌をまいていたが、こういう問題こそコンピュータ
で解かなければならないと今では思うし、その後の話では、コンピュータで解けたらしい・・・

返信 おすすめ メッセージ 8 / 11

投稿者：ダイエットパンダ 投稿日時：2006/06/02 0:11

補遺：
第五世代のアーカイブスの、注目すべき成果、のところに、未解決の群論の問題の一部を
自動証明したと、記載がありました。
数学などの問題には、結局、エレガントな解答はなく、煩雑な証明の積み重ねではじめて解ける
クラスの問題があるのだろうと思う。

http://www.icot.or.jp/ARCHIVE/Museum/FinalReport/node6.html#SECTION03012000000000000000

返信 おすすめ メッセージ 9 / 11

投稿者：ダイエットパンダ 投稿日時：2006/06/02 22:28

題：プログラムを入力して、走らせる。

簡単な例題を入力して走らせて見る。
例題）三段論法
・人はいずれ死ぬ。　(human(X)→mortal(X))
・ソクラテスは人である。　(human(socrates))
・よって、ソクラテスはいずれ死ぬ。　(mortal(socrates))

方式１：

GNU-Prologのコマンドラインで、下記のように入力する。
（赤い字が自分で入力する部分です。最後のピリオドを忘れずに！）

```
GNU Prolog 1.2.16
By Daniel Diaz
Copyright (C) 1999-2002 Daniel Diaz
| ?- [user].
compiling user for byte code...
mortal(X):-human(X).
human(socrates).
end_of_file.
user compiled, 3 lines read - 340 bytes written, 49265 ms
(1828 ms) yes
| ?- human(socrates).
yes
| ?- mortal(socrates).
yes
| ?- human(nobita).
no
| ?- mortal(nobita).
no
| ?-
```

要は、[user].を入力してから、end_of_file.を入力するまでの間が、プログラム（とファクト事実）であり、その後は、クエリーモード（質問受付モード）に戻る。
human(socrates).は、一応、yesが返ってくるのを確認するためであり、入力しなくてもよい。
要は、mortal(socrates).に対して、yesが返ってくるのをみるのが、目的である。
さらに、nobitaは、まだhumanであると言明してないので、human(nobita).が偽(no)で返り、
よって、mortal(nobita).も、予想通り、noが返ってくる。

方式２：
下記２行を入力したテキストファイル「sandanronpo.pl」を作成し、どこかに置いておく。
（例として、c:¥GNU-Prolog¥sandanronpo.pl　とする。）

```
mortal(X):-human(X).
human(socrates).
```

その後、下記のように入力する。
（赤い字が自分で入力する部分です。最後のピリオドを忘れずに！）

```
GNU Prolog 1.2.16
By Daniel Diaz
Copyright (C) 1999-2002 Daniel Diaz
| ?- ['c:/GNU-Prolog/sandanronpo.pl'].
compiling c:¥GNU-Prolog¥sandanronpo.pl for byte code...
c:¥GNU-Prolog¥sandanronpo.pl compiled, 2 lines read - 393 bytes written, 0 ms
yes
| ?- mortal(socrates).
yes
| ?- mortal(nobita).
no
| ?-
```

普通、Windowsでは、c:¥GNU-Prolog¥sandanronpo.pl　のように、¥を使うが、上記の'～'の
部分は、/ で区切る必要があります。
¥で区切ると、下記のようにエラーになります。

```
| ?- ['c:¥GNU-Prolog¥sandanronpo.pl'].
uncaught exception: error(syntax_error('user_input:4 (char:6) unknown escape
sequence'),read_term/3)
| ?-
```

途中で、間違えたかな、と思ったら、下記のように、Ctrl+Cを打ったあと、aなどでアボートさせて、やり直しができます。

```
| ?- [user].
compiling user for byte code...
human(zocrate      ←なんか変だな、と思って、Ctrlキーを押しながらCキーを打ちます。
Prolog interruption (h for help) ? h      ←ちょっとヘルプを見てから・・・
   a  abort        b  break
   c  continue     e  exit
   d  debug        t  trace
  h/? help
Prolog interruption (h for help) ? a      ← a（アボート）を打ち、あとはやり直す。
execution aborted
| ?-
```

今回は、簡単ですが、これでおしまいです。
あと、まだまだ調べなければいけないことは、and,or,notをどう書けばよいか、が残っています。
cutオペレータ「!」についても、前掲参考資料などを見ながら、調べてみます。
ひと段落ついたら、
目標(2)：適用する推論ルールをどのように選択するのか？（ルール間の連想機能があるのか？何か優先順位付けがあるのか？あるいは絨毯爆撃的にやるのか？）
に着手したいと思います。
m(_ _;)m

返信　おすすめ　メッセージ 10 / 11

投稿者：ダイエットパンダ　投稿日時：2006/06/17 0:46

だいぶ間があきましたが、目標２を再開します。m(_ _;)m
目標(2)：適用する推論ルールをどのように選択するのか？（ルール間の連想機能があるのか？何か優先順位付けがあるのか？あるいは絨毯爆撃的にやるのか？）

下記の例はあまりいい例ではないかもしれないけれど、
「みんな人間(human)なんだ～！」の信念のもとに、やってみます。

プログラム：（ファイル名を、c:¥GNU-Prolog¥human.pl とする。）

```
man(tarou).                  太郎は男である。（事実）
woman(hanako).               花子は女である。（事実）
okama(okama_jirou).          おかま次郎はおかまである。（事実）
onabe(onabe_haruko).         おなべ春子はおなべである。（事実）
hardgei(hardgei_saburou).    ハードゲイ三郎はハードゲイである。（事実）
realgei(realgei_sirou).      リアルゲイ四郎はリアルゲイである。（事実）
rezu(rezu_natsuko).          レズ夏子はレズである。（事実）
newhalf(mr_and_mrs).         ミスターアンドミセスはニューハーフである。（事実）
human(X):-man(X).            Xが男であれば、Xは人間である。（ルール）
human(X):-woman(X).          Xが女であれば、Xは人間である。（ルール）
human(X):-okama(X).          Xがオカマであれば、Xは人間である。（ルール）
human(X):-onabe(X).          Xがおなべであれば、Xは人間である。（ルール）
human(X):-hardgei(X).        Xがハードゲイであれば、Xは人間である。（ルール）
human(X):-realgei(X).        Xがリアルゲイであれば、Xは人間である。（ルール）
human(X):-rezu(X).           Xがレズであれば、Xは人間である。（ルール）
human(X):-newhalf(X).        Xがニューハーフであれば、Xは人間である。
```

プログラムを以下のようにロード後、human(mr_and_mrs).の問を、３回繰り返し、
トレース(trace.)にて、どのように推論規則を適用するのかみてみる。
赤字は自分で入力する部分です。

```
GNU Prolog 1.2.16
By Daniel Diaz
Copyright (C) 1999-2002 Daniel Diaz
| ?- ['c:/GNU-Prolog/human.pl'].
compiling c:¥GNU-Prolog¥human.pl for byte code...
c:¥GNU-Prolog¥human.pl compiled, 16 lines read - 1629 bytes written, 0 ms
(15 ms) yes
| ?- human(mr_and_mrs).
yes
| ?- trace.
The debugger will first creep -- showing everything (trace)
yes
{trace}
| ?- human(mr_and_mrs).
      1    1  Call: human(mr_and_mrs) ?
      2    2  Call: man(mr_and_mrs) ?
```

```
      2    2  Fail: man(mr_and_mrs) ?
      2    2  Call: woman(mr_and_mrs) ?
      2    2  Fail: woman(mr_and_mrs) ?
      2    2  Call: okama(mr_and_mrs) ?
      2    2  Fail: okama(mr_and_mrs) ?
      2    2  Call: onabe(mr_and_mrs) ?
      2    2  Fail: onabe(mr_and_mrs) ?
      2    2  Call: hardgei(mr_and_mrs) ?
      2    2  Fail: hardgei(mr_and_mrs) ?
      2    2  Call: realgei(mr_and_mrs) ?
      2    2  Fail: realgei(mr_and_mrs) ?
      2    2  Call: rezu(mr_and_mrs) ?
      2    2  Fail: rezu(mr_and_mrs) ?
      2    2  Call: newhalf(mr_and_mrs) ?
      2    2  Exit: newhalf(mr_and_mrs) ?
      1    1  Exit: human(mr_and_mrs) ?
(31 ms) yes
{trace}
| ?- human(mr_and_mrs).
      1    1  Call: human(mr_and_mrs) ?
      2    2  Call: man(mr_and_mrs) ?
      2    2  Fail: man(mr_and_mrs) ?
      2    2  Call: woman(mr_and_mrs) ?
      2    2  Fail: woman(mr_and_mrs) ?
      2    2  Call: okama(mr_and_mrs) ?
      2    2  Fail: okama(mr_and_mrs) ?
      2    2  Call: onabe(mr_and_mrs) ?
      2    2  Fail: onabe(mr_and_mrs) ?
      2    2  Call: hardgei(mr_and_mrs) ?
      2    2  Fail: hardgei(mr_and_mrs) ?
      2    2  Call: realgei(mr_and_mrs) ?
      2    2  Fail: realgei(mr_and_mrs) ?
      2    2  Call: rezu(mr_and_mrs) ?
      2    2  Fail: rezu(mr_and_mrs) ?
      2    2  Call: newhalf(mr_and_mrs) ?
      2    2  Exit: newhalf(mr_and_mrs) ?
      1    1  Exit: human(mr_and_mrs) ?
(47 ms) yes
{trace}
| ?- human(mr_and_mrs).
      1    1  Call: human(mr_and_mrs) ?
      2    2  Call: man(mr_and_mrs) ?
      2    2  Fail: man(mr_and_mrs) ?
      2    2  Call: woman(mr_and_mrs) ?
      2    2  Fail: woman(mr_and_mrs) ?
      2    2  Call: okama(mr_and_mrs) ?
      2    2  Fail: okama(mr_and_mrs) ?
      2    2  Call: onabe(mr_and_mrs) ?
      2    2  Fail: onabe(mr_and_mrs) ?
      2    2  Call: hardgei(mr_and_mrs) ?
      2    2  Fail: hardgei(mr_and_mrs) ?
      2    2  Call: realgei(mr_and_mrs) ?
      2    2  Fail: realgei(mr_and_mrs) ?
      2    2  Call: rezu(mr_and_mrs) ?
      2    2  Fail: rezu(mr_and_mrs) ?
      2    2  Call: newhalf(mr_and_mrs) ?
      2    2  Exit: newhalf(mr_and_mrs) ?
      1    1  Exit: human(mr_and_mrs) ?
(31 ms) yes
{trace}
| ?-
```

結局、与えられた推論ルールの順番に適用し、最後に newhalf(mr_and_mrs). にマッチング
してyesを返している。

これが人間だったらどうか？
おそらく、普通の時は、もっとも問い合わせ・回答の累積頻度が多かったものから検索し、
時々、もっとも最近に問い合わせ・回答が何回かあれば、それを優先で(most recently questioned) 即座に回答し、
時間がたつにつれて、また、頻度順位をすこし修正して、頻度順検索にまた戻ると思われるが、
この例のプログラムは、そういったことはお構いなしに、すべてのルールを先頭から評価する。
「最も最近」、とか「最も頻度の多い」、とかの考慮はまったくされていない。

結論：
これをもって、知識推論ベースのシステムに、人間のような考えをすることを期待しては
ならないことが読み取れるであろう。
むしろ、知識を元にした推論を可能な限り適用しているところが計算機らしいところでもあり、
得意分野でもある。

返信 おすすめ メッセージ 11 / 11

投稿者：ダイエットパンダ 投稿日時：2006/06/17 0:56

以上をもって、下記の当初の目標について、GNU-Prologに関しては、
期待通りの結果を得られた。

(1)決定不能な命題（村の理髪師のパラドックスなど）を、Prologの処理系がどう扱うか？
　→無限ループに陥って、スタックオーバーフローとなる。

(2)適用する推論ルールをどのように選択するのか？（ルール間の連想機能があるのか？何か優先順位付けがあるのか？あるいは絨毯爆撃的にやるのか？）
　→与えられたルールを、与えられた順番にマッチングしていく。
　　「最も最近」使ったルールとか、「最も頻繁に」使用するルール」を
　　優先的にマッチング判定するわけではない。

以上により、知識・推論ベースが、いかに人間の思考過程とはなれているか、
そして、むしろどのような問題が得意か、を感覚的に理解していただければ、
本稿の目的を果たしたことになる。従って、知識推論ベースに多くを期待しては
ならず、正しく理解把握して使用することが大事かな、と思います。

昔第五世代開始の頃、友人などから、「最近のコンピュータは人間のように考えるのか？」
の質問を受け、そういうことではないことを説明するのにいつも苦労していたので
、この場を借りて、実例で示したつもりです。
以上で一旦終わりといたします。また何かネタがあれば、調べてみたいと思います。
ありがとうございましたm(_ _;)m

最初の返信 前へ 2-11 通を表示 : 総返信数 11 通 次へ 最新の返信
パンダ的電脳言語考 に戻る 前の話題 次の話題 返信を受信トレイに送信