

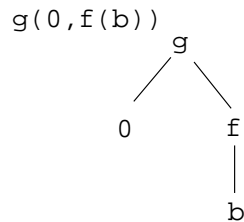
記号処理

Prolog はデータとして、数値だけではなく、項を扱うことができる。データは構造を持ったオブジェクトであり（文字列を使って表現されるが）、「文字列を前から見ていく」という概念はない。

データ構造（述語が引数とするもののデータ構造であってプログラム自身の構造ではないことに注意）

- 変数 (variable) は大文字または '_' で始まる文字列である。
- 定数 (atom) は小文字で始まる文字列である。
- 構造 (structure) は $\text{func}(T_1, \dots, T_n)$ の形をしている。ただし、 func はファンクタと呼ばれる演算子、 T_1, \dots, T_n は項である。
- 項 (term, データオブジェクト) は変数か定数か構造である。

構造の木表現



注意：項はデータ，すなわち引数として扱うものなので，ゴールと混同しないこと。

ゴールは $\text{pred}(T_1, \dots, T_n)$ の形をしている。ただし、 pred は述語記号（関係）、 T_1, \dots, T_n は項である。リストも $\text{list}(H, T)$ という構造を持つ項の 1 つと考えてよい。

このようなデータを扱うときは

- 木構造を根ノードから順にノードを 1 つずつ処理し、それよりも下の木構造に対して再帰をかけ（cf. リストの場合に頭部を処理し尾部には再帰をかけたのと同様）
- 最後に葉ノード（ベースケース）に至ったらその処理をして終了する

という形のプログラムになる。

テストデータが複雑な形なので、タイプミスのないようプログラム中に

```
test(X) :- subst(f(f(b)), X).
```

と定義しておくなど工夫せよ。

最初は単純なデータでテストしてから複雑なデータでテストするようにせよ。また、単純なデータを使ってエラーの原因を特定すること。

練習問題

- 以下の文法 (\mathcal{G}_1) になかったもののみがデータとして与えられるとする。このとき、項 Term がアトム a を含むかどうかを判定する述語 `ocr_a(Term)` のプログラムを作成せよ。たとえば `ocr_a(f(f(a)))` は成功し、`ocr_a(f(f(b)))` は失敗する。 \mathcal{G}_1 はプログラムが対象とする言語であり、プログラミング言語と混同しないように注意。**(\mathcal{G}_1 をプログラムの一部として記述するのは間違いである。)**

(\mathcal{G}_1)
`Term ::= Alphabet | f(Term)`
`Alphabet ::= a | b`

- 文法 (\mathcal{G}_1) になかったもののみがデータとして与えられるとする。項 T1 に出現するアトム a をすべて c に書き換えた結果が項 T2 であるという関係を表す述語 `subst(T1,T2)` のプログラムを作成せよ。たとえば、`subst(f(f(a)), X)` は `X = f(f(c))` となって成功し、`subst(f(f(b)), X)` は `X = f(f(b))` となって (書き換えなしで) 成功する。
- 文法 (\mathcal{G}_1) になかったもののみがデータとして与えられるとする。項 Term に含まれるファンクタの数が N であるという関係を表す述語 `n_functors(Term,N)` のプログラムを作成せよ。たとえば `n_functors(f(f(f(a))), X)` は `X = 3` となって成功する。
- 文法 (\mathcal{G}_2) になかったもののみがデータとして与えられるとする。項 T1 に出現するアトム a をすべて c に書き換えた結果が項 T2 であるという関係を表す述語 `subst2(T1,T2)` のプログラムを作成せよ。たとえば、`subst2(f(g(1, f(a))), X)` は `X = f(g(1, f(c)))` となって成功する。

(\mathcal{G}_2)
`Term ::= Alphabet | Digit | f(Term) | g(Term,Term)`
`Alphabet ::= a | b`
`Digit ::= 0 | 1`

演習問題 (r6)

* のついている問題はオプションなのでできる者のみ解答せよ。

- 文法 (\mathcal{G}_2) になかったもののみがデータとして与えられるとする。項 T1 に出現するアトム A をすべて項 B に書き換えた結果が項 T2 であるという関係を表す述語 `subst_atom(A,B,T1,T2)` のプログラムを作成せよ。たとえば、`subst_atom(a, g(b,b), g(a, f(1)), X)` は `X = g(g(b,b), f(1))` となって成功する。
- 文法 (\mathcal{G}_2) になかったもののみがデータとして与えられるとする。与えられたアトム A が与えられた項 Term に出現する回数が C であるという関係を表す述語 `count_ocr(A,Term,C)` のプログラムを作成せよ。たとえば `count_ocr(a, f(g(a, f(a))), C)`, `count_ocr(b, f(g(a, f(a))), C)`, `count_ocr(a, f(g(b, f(a))), C)` はそれぞれ `C = 2, 0, 1` となって成功する。

(Hint)
`count_ocr(A,A,1).`
`% ベースケースが一致すれば 1 回となって終了`
`count_ocr(A,f(T),C) :- count_ocr(A,T,C).`
`% f(T) と T における A の出現回数の関係は ?`
`count_ocr(A,g(T1,T2),C) :- count_ocr(A,T1,C1), count_ocr(A,T2,C2), C is C1+C2.`
`% g(T1,T2) と T1,T2 それぞれにおける A の出現回数の関係は ?`
`count_ocr(A,B,0).`
`% ベースケースが一致しなければ 0 回となって終了`

- 文法 (\mathcal{G}_2) になかったもののみがデータとして与えられるとする。項 T1 に出現するファンクタ f をすべて h に書き換えた結果が項 T2 であるという関係を表す述語 `subst_functor(T1,T2)` のプログラムを作成せよ。たとえば `subst_functor(f(g(a, f(a))), T2)`, `subst_functor(f(g(a, f(a))), T2)`, `subst_functor(f(g(b, f(a))), T2)` はそれぞれ `T2 = h(g(a, h(a)))`, `T2 = h(g(a, h(a)))`, `T2 = h(g(b, h(a)))` となって成功する。

- (4) 文法 (\mathcal{G}_2) になかったもののみがデータとして与えられるとする．与えられた項 Term に含まれるファンクタの数が N であるという関係を表す述語 `n_of_functors(Term,N)` のプログラムを作成せよ．項以外のものは入力されないと仮定してよい．
たとえば `n_of_functors(f(g(a, f(b))), X)` は `X = 3` となって成功する．
- (5) 文法 (\mathcal{G}_2) になかったもののみがデータとして与えられるとする．与えられたアトム A が与えられた項 Term に出現するかどうかを判定する述語 `ocr_check(A,Term)` のプログラムを作成せよ．`ocr_check` のみを用いて (他の述語を使わず) 再帰的に定義すること．
たとえば `ocr_check(a, f(g(a, f(a))))`, `ocr_check(b, f(g(a, f(a))))`,
`ocr_check(a, f(g(b, f(a))))` はそれぞれ `true`, `false`, `true` となる．
- (6)* 文法 (\mathcal{G}_3) になかったもののみがデータとして与えられるとする．与えられた項 (Term) に含まれる部分項のリストが L であるという関係を表す述語 `list_of_subterms(Term,L)` のプログラムを組み込み述語 `append` (text pp.69-70 `conc` と同じ) を使って作成せよ．ただしリストは重複を許すとする．また、項以外のものは入力されないと仮定してよい．
たとえば `list_of_subterms(f(g(a, f(b))), X)` は
`X = [f(g(a, f(b))), g(a, f(b)), a, f(b), b]` となって成功する．

(\mathcal{G}_3)

```
Term ::= Alphabet | f(Term) | g(Term,Term)
Alphabet ::= a | b
```

- (7) 練習問題 4 の解答例プログラムの (i) 各節の論理的意味および
(ii) `?- subst2(f(g(1, f(a))), X)` を実行したときの実行過程を示せ．(i) については**命題の形になっていること**，すなわち，引数への入出力を書くのではなく、「C である」「A かつ B ならば C である」のように記述すること．第 1 回の資料「Prolog の実行過程」，第 2 回資料「sum の実行過程の補足」などを参考に「ゴール」「実行」「単一化 (ユニフィケーション)」という用語をすべて用いて段階的に説明せよ．(**トレースを貼り付けてはいけない．**)