Listes en prolog

Les **listes** sont une structure de données fondamentale en Prolog. Elles sont très souvent utilisées en programmation. Une liste est une séquence finie d'éléments.

- Exemples de listes en Prolog :
  - `[mia, vincent, jules, yolanda]`
  - `[mia, criminel(honeybunny), X, 2, mia]`
  - `[]`
  - `[mia, [vincent, jules], [butch, ami(butch)]]`
  - `[[], dead(z), [2, [b,c]], [], Z, [2, [b,c]]]`
- Tous les types d'éléments de Prolog peuvent être des éléments d'une liste (termes simples, complexes, listes, …).
- La longueur d'une liste est le nombre d'éléments qu'elle contient.
- Les éléments d'une liste sont entourés de crochets `[A,B,…]`.
- Il existe une liste spéciale : la **liste vide** `[]`.

Tête et Queue

Une liste non vide peut être vue comme étant composée de deux parties :

- La **tête** est le premier élément.
- La **queue** est le reste de la liste lorsqu'on enlève le premier élément. La queue d'une liste est toujours une liste.

Exemple 1 :

```
[mia, vincent, jules, yolanda]
```

- Tête : `mia`
- Queue : `[vincent, jules, yolanda]`

Exemple 2 :

```
[[], mort(z), [2, [b,c]], [], Z, [2, [b,c]]]
```

- Tête : `[]`
- Queue : `[mort(z), [2, [b,c]], [], Z, [2, [b,c]]]`

Exemple 3 :

```
[mort(z)]
```

- Tête : `mort(z)`
- Queue : `[]`

La liste vide joue un rôle important dans les prédicats récursifs pour manipuler les listes en Prolog. Pour Prolog, `[ ]` est une liste spéciale, simple, sans structure interne. La liste vide n'a ni queue ni tête.

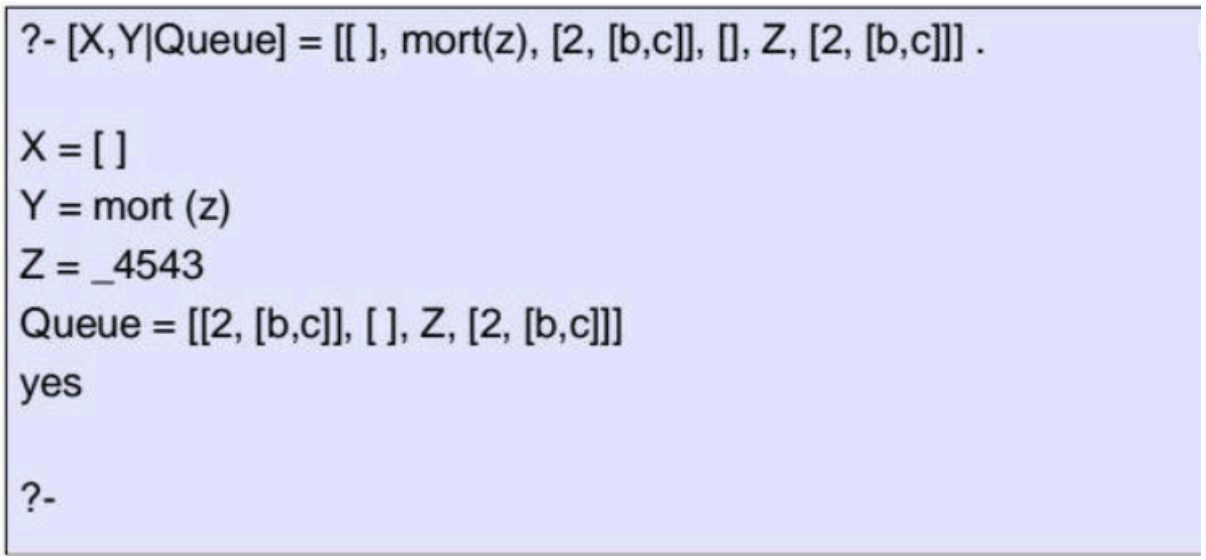
L'opérateur prédéfini `|`

Prolog dispose d'un opérateur prédéfini pour décomposer les listes en sa tête et sa queue : `|`. L'opérateur `|` est un outil indispensable pour écrire des prédicats qui manipulent des listes.

Exemple:

```
?- [Tete|Queue] = [mia, vincent, jules, yolanda].
Tete = mia
Queue = [vincent,jules,yolanda]
yes
```

Cet exemple montre une tentative d'unification où `[X|Y]` est essayé d'être unifié avec une liste vide `[ ]`, ce qui échoue car une liste vide ne peut pas être décomposée en une tête et une queue.



Cet exemple illustre la décomposition d'une liste en tête, deuxième élément, et queue en Prolog. On utilise `[X,Y|Queue]` pour extraire le premier élément en `X`, le deuxième en `Y`, et le reste de la liste en `Queue`.

Variables anonymes _

Supposons que nous nous intéressions au 2ème et au 4ème éléments d'une liste. Le moyen le plus simple d'obtenir l'information désirée est d'utiliser des variables anonymes.

- Exemple :

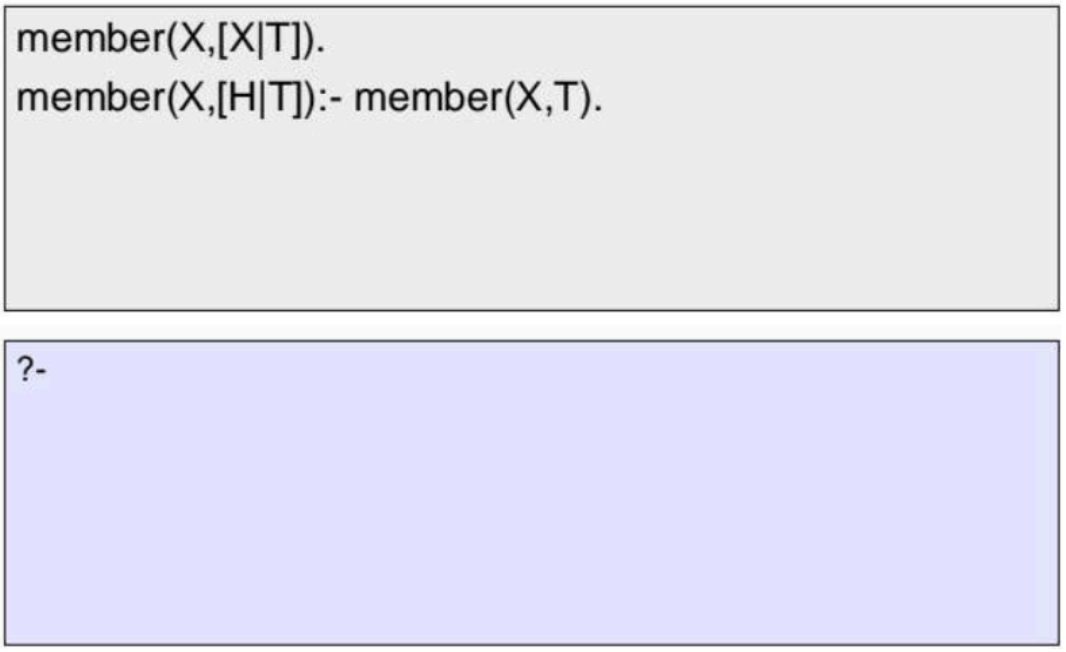
```
?- [ _,X2, _,X4|_ ] = [mia, vincent, marsellus, jody, yolanda].
X2 = vincent
X4 = jody
yes
```

- Le **souligné** `_` est la **variable anonyme**.
- Chaque occurrence de la variable anonyme est indépendante : elles peuvent être liées à des éléments différents.
- Elle est utilisée lorsqu'une variable est nécessaire mais que son instantiation ne nous intéresse pas.

Membre

Ce prédicat est généralement appelé `member/2`. Il sert à déterminer si un élément `X` appartient à une liste `L`.

```
member(X,[X|T]).
member(X,[H|T]): - member(X,T).
```



La première règle `member(X, [X|T]).` vérifie si `X` est la tête de la liste. Si c'est le cas, `X` est membre de la liste. La deuxième règle `member(X, [H|T]) :- member(X, T).` vérifie si `X` est membre de la queue de la liste. Si `X` n'est pas la tête, alors on vérifie récursivement si `X` est membre du reste de la liste (`T`).

Voici des exemples d'utilisation :

```
?- member(yolanda,[yolanda,trudy,vincent,jules]).
true.

?- member(vincent,[yolanda,trudy,vincent,jules]).
true.

?- member(zed,[yolanda,trudy,vincent,jules]).
false.
```

```
member(X,[X|T]).
member(X,[H|T]):- member(X,T).
```

```
?- member(zed,[yolanda,trudy,vincent,jules]).
no
?-
```

En utilisant une variable pour la recherche :

```
?- member(X,[yolanda,trudy,vincent,jules]).
X = yolanda ;
X = trudy ;
X = vincent ;
X = jules ;
false.
```

```
member(X,[X|T]).
member(X,[H|T]):- member(X,T).
```

```
?- member(X,[yolanda,trudy,vincent,jules]).
X = yolanda;
X = trudy;
X = vincent;
X = jules;
no
```

Une réécriture possible de `member/2` est la suivante :

Le prédicat `member/2` fonctionne par récursivité pour parcourir une liste :

- Traiter la tête, puis
- Faire récursivement la même chose à la queue

Exemple: `a2b/2`

Le prédicat `a2b/2` prend deux listes comme arguments et réussit si :

- Le premier argument est une liste de `a`, et
- Le deuxième argument est une liste de `b` de la même longueur.

Exemples :

```
?- a2b([a,a,a,a],[b,b,b,b]).
true.

?- a2b([a,a,a,a],[b,b,b]).
false.

?- a2b([a,c,a,a],[b,b,b,t]).
false.
```

1-Définition de `a2b/2` :

Cas de base : la liste vide

```
a2b([],[]).
```

2-Cas récursif :

```
a2b([a|L1],[b|L2]): - a2b(L1,L2).
```

3-Code complet:

```
a2b([],[]).
a2b([a|L1],[b|L2]): - a2b(L1,L2).
```

```
a2b([],[]).
a2b([a|L1],[b|L2]):- a2b(L1,L2).
```

```
?- a2b([a,a,a],[b,b,b]).
yes
?-
```

```
a2b([],[]).
a2b([a|L1],[b|L2]):- a2b(L1,L2).
```

```
?- a2b([a,a,a,a],[b,b,b]).
no
?-
```

Références :

SWI-Prolog. (s. d.). https://www.swi-prolog.org/

Learn ProLog Now ! (s. d.). https://lpn.swi-prolog.org/lpnpage.php?pageid=top

PROLOG – Programmation Logique

Traduction : Yannick Estève
Cours dispensé à **CERI – Avignon Université**
[Lien si disponible]