## 9.3. Prédicats du second ordre

En Prolog les arguments d'un prédicat sont considérés comme des données (atomes ou termes composés), ils ne sont pas évalués. En programmation du second ordre les arguments d'un prédicat peuvent correspondre à des prédicats, nous avons rencontré un tel méta-prédicat avec non(P) (à programmer), gProlog proposant /+(P)).

Un autre méta-prédicat (prédéfini) très intéressant est findall (V, But, L) qui réussit si L est la liste des solutions V du problème But.

## Exemple 1: Trouver les enfants de john

```
| ?- findall(X, pere(john, X), L).
L = [john john, caroline]
```

La question pere (john X) fournit toutes les solutions X séparément. Le prédicat findall permet de collecter toutes les solutions d'un problème passé en paramètre.

## Exemple 2: Parcours en largeur d'un graphe

Le parcours en largeur d'un graphe se fait de manière itérative à l'aide d'une file de nœuds à explorer (cf cours d'algorithmique et de graphes). La boucle gérant un tel parcours retire un nœud à explorer de la file et ajoute dans la file les descendants de ce nœud et cela tant qu'il y a des nœuds à explorer. Afin d'éviter de tomber dans un cycle, seuls les nœuds qui n'ont pas déjà été observés (déjà explorés ou en attente d'être explorés) sont introduits dans la file.

En *Prolog* il faut décrire le graphe. Cela peut être réalisé en définissant la base de fait des arcs (exemple vu en TP) :

Nous sommes ici à nouveau dans un cas d'itération traitée en récursivité terminale avec donc en question parcours Largeur ([a],[],R).

MiniTP: Une famille qui aura beaucoup donné à Prolog....

En repartant de  $\mbox{\it w famille Kennedy.pl}$  », collectez avec un seul succès la descendance de rose.

## Rappels cours précédent

- *Prolog* propose des extensions avec les notions de contraintes, de pseudo-prédicat d'entrée-sortie, de méta-prédicat d'introspection et de modification de la base de connaissances
- Les essais successifs dans le modèle *générer-tester* sont un modèle de programmation qui permet par ailleurs de s'appuyer sur un historique avec la récursivité
- L'usage de contraintes permet de gagner en efficacité de l'approche *générer-tester* qui devient alors *contraindre-tester*
- Avec findall, il est possible d'avoir un méta-prédicat réunissant tous les succès en un seul