

– TD3 OCAML –

Relations

Les relations sont une généralisation des fonctions qui constituent le fondement théorique de nombreux logiciels informatiques. Par exemple, une relation peut définir un ensemble de liens dans un réseau: $a \rightarrow b$ se code par (a, b) . b est une image de a et a est un antécédent de b . Une relation sera représentée par une liste triée de couples qui permet d'avoir des algorithmes plus performants. L'objet de ce TD est de définir des fonctions de base manipulant les relations. Par exemple, une relation serait décrite ainsi: $[(1,2);(2,3);(2,4);(3,1);(4,4)]$.

1. Programmer la fonction `mém c r` qui cherche si un couple $c = (x, y)$ fait partie de la relation.
2. Programmer la fonction `img r` qui donne les images de r .
3. Faites la fonction `is_fun r` qui teste si une relation est une fonction. Une relation est une fonction si chaque antécédent est unique.
4. Programmer la fonction `union r1 r2` réalisant l'union des deux relations.
5. Programmer la fonction `inter r1 r2` réalisant l'intersection des relations.
6. Tri d'une liste par le tri fusion. Ce tri consiste à séparer une liste en 2 sous-listes de taille égale approximativement et ce de manière récursive, puis ensuite d'en faire une fusion triée. Pour les relations la fusion correspond à une union triée. Le principe de récursion est le suivant : *le tri fusion correspond à l'union de deux sous listes triées à partir de la liste initiale.*
 - Faites une fonction `split l` qui retourne un couple de 2 listes de taille approximativement égale.
 - Faîte la fonction `sortset l` qui prend une liste et réalise le tri fusion pour avoir en sortie un ensemble trié (liste triée avec une seule occurrence des éléments) en utilisant le tri fusion. EXEMPLE
`sortset [1;2;4;0;45;17;17;8;17];; -: int list = [0; 1; 2; 4; 8; 17; 45]`
7. Programmer la fonction `invert r` qui retourne l'inverse d'une relation. (sous forme de liste triée).
8. Faites une fonction `linkit x l` qui crée une relation r avec un élément x et une liste l d'éléments correspondant à l'image de x dans r . si $l = [a_1, .. a_n]$ on obtient: $[(x, a_1), ..., (x, a_n)]$.
9. Faites la fonction `composition` qui compose 2 relations. $(a, b) \circ (b, c) = (a, c)$
10. Faites la fonction `identity r` qui construit une relation d'identité des éléments de la relation. Par exemple, `identity [(1,2);(1,3);(1,4);(2,3);(2,4);(3,5);(3,6);(5,6)]` retourne:
`[(1, 1); (2, 2); (3, 3); (4, 4); (5, 5); (6, 6)]`
11. Faites une fonction `reachable r x y` qui teste s'il existe un chemin de x à y dans la relation r , c'est à dire si l'on peut atteindre y à partir de x en traversant les relations élémentaires.
12. La fermeture transitive d'une relation consiste à relier des éléments indirectement reliés. Par exemple, la fermeture transitive de $\{1 \rightarrow 2, 2 \rightarrow 3, 3 \rightarrow 4\}$ est $\{1 \rightarrow 2, 1 \rightarrow 3, 1 \rightarrow 4, 2 \rightarrow 3, 2 \rightarrow 4, 3 \rightarrow 4\}$. Faites la fonction `closure` réalisant cette fermeture.