Opérateurs et mots clés du langage B

Expressions ensemblistes

Le B permettant de travailler avec des ensembles abstraits offre donc différentes opérations permettant de les manipuler dans les expressions. Ci-après, en notation ASCII les principales :

/\	Intersection
\/	Union
-	Différence
*	Produit cartésien
:	Appartenance
/:	Non appartenance
<:	Inclusion
<<:	Inclusion stricte
/<<:	Non inclusion stricte

POW	Ensemble des sous-ensembles
POW1	Ensemble des sous-ensembles non vides
FIN	Ensemble des sous-ensembles finis
FIN1	Ensemble des sous-ensembles finis non vides
{}	Ensemble vide
m..n	Intervalle [m,n]

Expressions arithmétiques

3 Le B offre aussi différentes expressions arithmétiques (pour rappel, les seuls types numériques autorisés sont entiers).

+	Addition
-	Soustraction et moins unaire
*	Multiplication
**	Puissance
/	Division entière
mod	Modulo
succ	Successeur (+1)
pred	Prédécesseur (-1)

max	Maximum ensembliste
min	Minimum ensembliste
card	cardinal
PI	Produit généralisé
SIGMA	Somme généralisée

Expressions arithmétiques

- Exemples : avec E={-2,5,-7,8,-3}
- card(E)=5
- max(E)=8
- min(E)=-7
- PI(x).(x : E & x > 0 | x+1)=54
- SIGMA(x).(x : E & x > 0 | x**2)=89
- succ(x)=x+1
- pred(x)=x-1
- 18/4=4
- 18 mod 4=2

Quantificateurs et opérateurs logiques

5

Notation ASCII des quantificateurs existentiels et opérateurs logiques. Attention à ne pas confondre le or logique et le OR du CHOICE ou du CASE

!	Quel que soit ($\forall$)
#	Il existe ($\exists$)
&	Et logique
or	Ou logique
OR	Ou du CHOICE ou du CASE
bool	Conversion d'un prédicat en valeur booléenne

Relations et fonctions

- Notation ASCII des différentes relations et fonctions entre ensembles

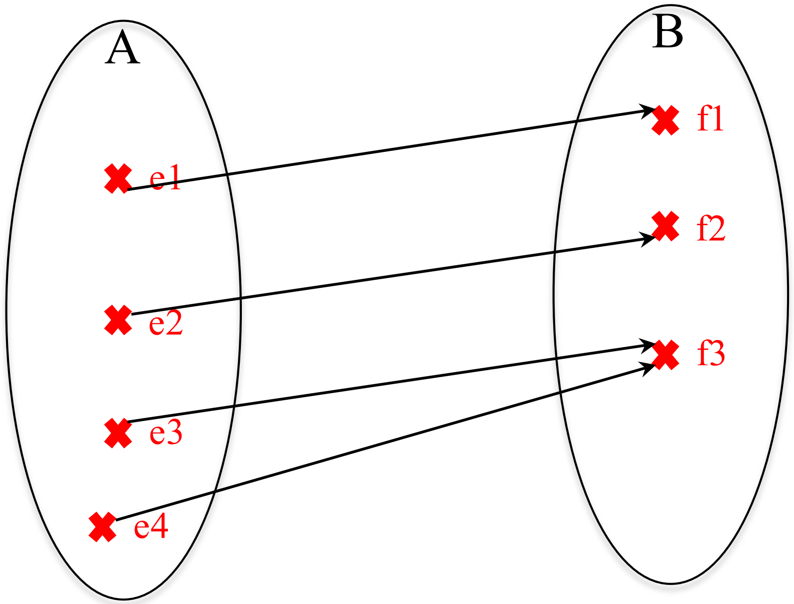
<->	Relations
+->	Fonctions partielles
-->	Fonctions totales
>+>	Injections partielles
>->	Injections totales
+->>	Surjections partielles
-->>	Surjections totales

>->>	Bijection partielle
>+>>	Bijection totale
dom	domaine
ran	codomaine
~	Inverse (R^{-1} noté R~)
R[S]	Image par R d'un sous-ensemble S du domaine

- Pour rappel, une relation est un ensemble donc card(f) est le nombre de couples en relation par f

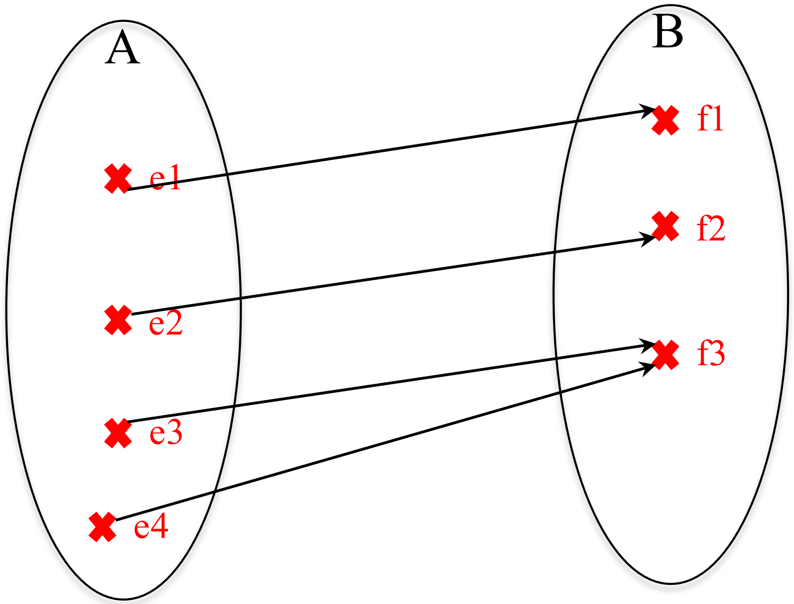
Opérations sur relations et fonctions

- Restriction sur le domaine : <|
- R={(e1|->f1), (e2|->f2),(e3|->f3), (e4|->f3)}
- {e1,e2}<|R={(e1|->f1), (e2|->f2)}



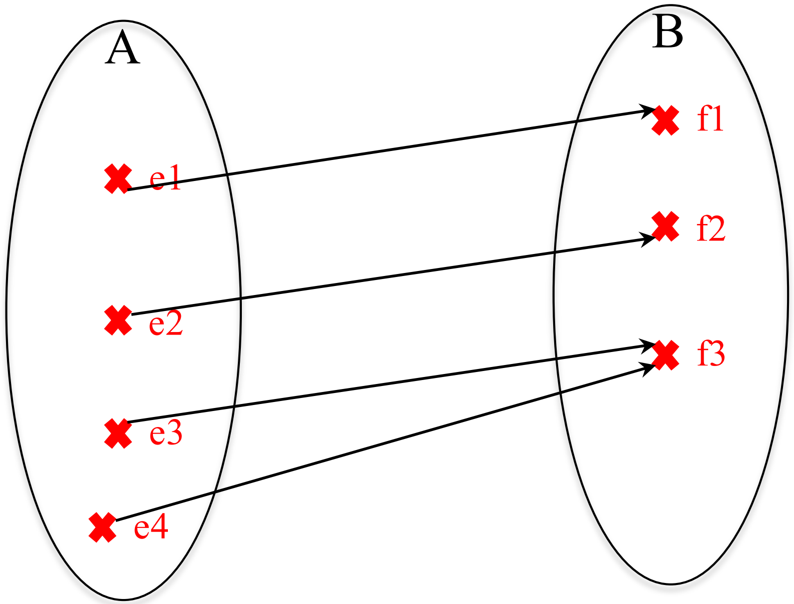
Opérations sur relations et fonctions

- Soustraction sur le domaine : <<|
- R={(e1|->f1), (e2|->f2),(e3|->f3), (e4|->f3)}
- {e3,e4}<<|R={(e1|->f1), (e2|->f2)}



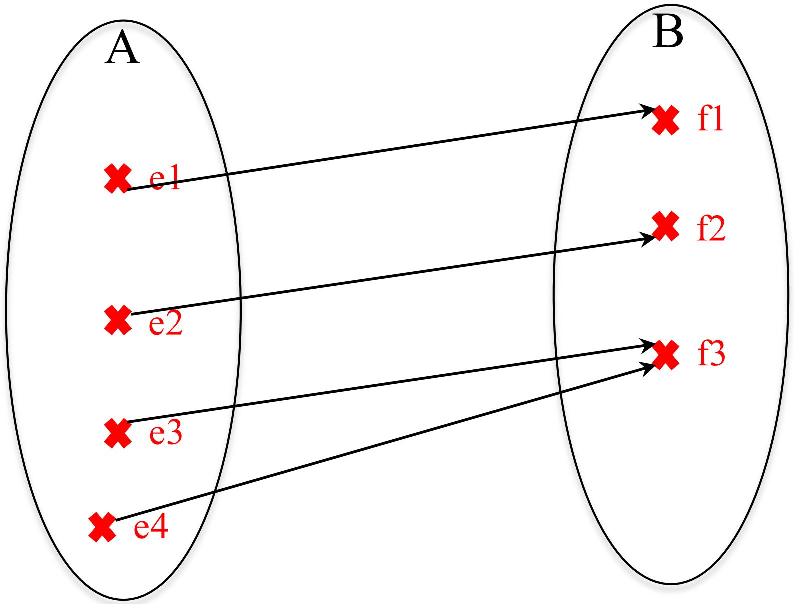
Opérations sur relations et fonctions

- Restriction sur le codomaine : |>
- R={(e1|->f1), (e2|->f2),(e3|->f3), (e4|->f3)}
- R|>{f3}={(e3|->f3), (e4|->f3)}



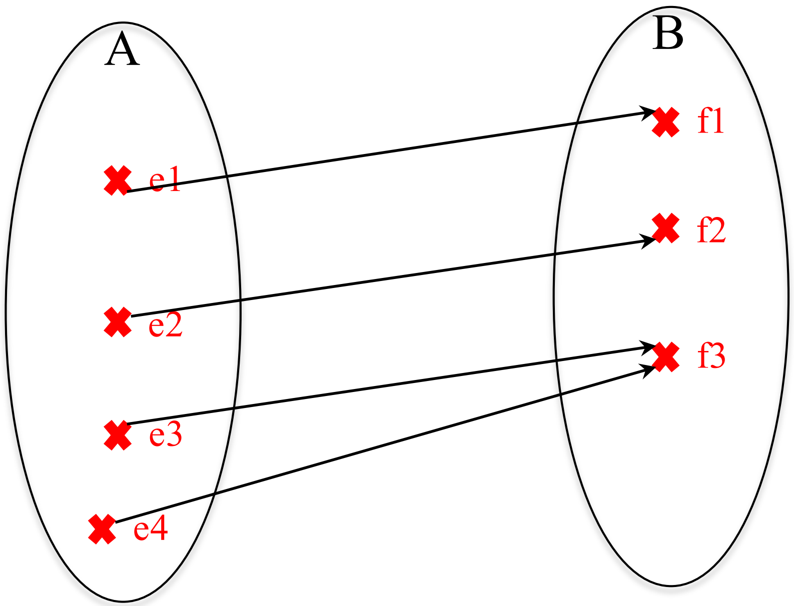
Opérations sur relations et fonctions

- ## Soustraction sur le codomaine : |>>
- R={(e1|->f1), (e2|->f2),(e3|->f3), (e4|->f3)}
- R|>>{f3}={(e1|->f1), (e2|->f2)}



Opérations sur relations et fonctions

- Surcharge d'une fonction : <+
- R={(e1|->f1), (e2|->f2),(e3|->f3), (e4|->f3)}
- R<+{(e1|->f2)} = {(e1|->f2), (e2|->f2),(e3|->f3), (e4|->f3)}
- Peut aussi s'écrire R(e1):=f2



Cas général (pour deux relations) :
R1<+R2 est l'ensemble constitué de R2 et des éléments de R1 dont le premier élément du couple n'appartient pas au domaine de R2.

Opérations sur les relations et fonctions

Notation ASCII des différentes opérations sur les relations et fonctions entre ensembles

<|	Restriction sur le domaine
<<|	Soustraction sur le domaine
|>	Restriction sur le codomaine
|>>	Soustraction sur le codomaine
<+	Surcharge

Composition de relations

- R1 : relation de A vers B et R2 : relation de B vers C
- (R1;R2) est la composition : pour des fonctions c'est la fonction notée en maths R2∘R1 qui à un élément a de A fait correspondre l'élément R2(R1(a)) de C
- Plus généralement
- Pour R1∈A <-> B et R2∈ B <-> C
- (R1;R2)={(a|->c) | (a|->c)∈A×C ∧ ∃b.(b∈B ∧ (a|->b)∈R1 ∧ (b|->c)∈R2}
- On rappelle que ; peut désigner aussi la substitution séquentielle, donc gare à des écritures du genre R3:=R1;R2 ambiguë donc interdite (risque de confusion avec (R3:=R1);R2 certes incorrect car R2 n'est pas une substitution), il faut écrire R3:=(R1;R2)

Itération et fermeture de relations

- Dans un ensemble E on note id(E)={(e|->e)|e∈E}
- Pour une relation R d'un ensemble E dans lui-même on note R^n la relation définie par récurrence par :
  - R^0=Id(E)
  - ∀n>0 R^n=(R;R^{n-1})
- On appelle fermeture transitive de R, et on note R^+ (en ASCII closure1(R)) la plus petite relation transitive contenant R (intuitivement c'est l'union de tous les itérés de R pour n>0)
- On appelle fermeture transitive et réflexive de R et on note R^* (en ASCII closure(R)) la plus petite relation transitive et reflexive contenant R (intuitivement c'est l'union de tous les itérés de R pour n≥0)

Itération et fermeture de relations

- Exemple : R={(1|->3), (2|->1), (2|->2), (3|->3)}
- R^2={(1|->3), (2|->3), (2|->1), (2|->2), (3|->3)}
- closure1(R)=R^2
- closure(R)=
  {(1|->1), (1|->3), (2|->3), (2|->1), (2|->2), (3|->3)}

Produit direct de relations

- Pour R1∈A <-> B et R2∈ A <-> C
- R1⊗R2 (noté en ASCII R1><R2) est la relation de A dans B×C qui met en correspondance a|->(b|->c) où (a|->b) ∈ R1 ∧ (a|->c) ∈ R2
- R1⊗R2={(a|->(b|->c)) | (a|->b) ∈ R1 ∧ (a|->c) ∈ R2}
- Exemple : R1={(1|->5), (1|->7), (2|->3), (3|->4)}
  R2={(1|->2), (2|->4)}
- R1⊗R2 ={(1|->(5|->2), (1|->(7|->2), (2|->(3|->4)}

Produit parallèle de relations

- Pour R1∈A <-> B et R2∈ C <-> D
- R1||R2 est la relation de A×C dans B×D qui met en correspondance (a|->c)|->(b|->d) où (a|->b) ∈ R1 ∧ (c|->d) ∈ R2
- R1||R2={((a|->c)|->(b|->d)) | (a|->b) ∈ R1 ∧ (c|->d) ∈ R2}
- Exemple : R1={(1|->5), (2|->7)}
  R2={(8|->1), (9|->4)}
- R1||R2 ={((1|->8)|->(5|->1)), ((1|->9)|->(5|->4)), ((2|->8)|->(7|->1)), ((2|->9)|->(7|->4))}

Lambda expression

- La lambda expression est une manière abstraite de définir une fonction à l'aide de
  - Un prédicat P définissant le domaine d'une variable x
  - Une expression E(x) dépendant de x
- λx.(P | E(x)) noté en ASCII %x.(P | E(x)) est la fonction définie par l'ensemble des couples (x|->E(x)) où x appartient au domaine défini par P
- Exemple : %x.(x : 1..3 | 2*x)={(1|->2), (2|->4), (3|->6)}

Transformées en fonction / relation

- La transformée en fonction fnc(R) d'une relation R de A dans B est la fonction de A dans POW(B) qui à tout élément a de A associe l'ensemble des éléments de B liés par R à a.
- Exemple :
  - R={(1|->3), (2|->1), (2|->2), (3|->3)}
  - fnc(R)={(1|->{3}), (2|->{1,2}), (3|->{3})}
- La transformée en relation rel (f) d'une fonction f de A dans P(B) est l'ensemble des couples composés d'un élément a de A et des éléments de f(a) : c'est donc l'opération inverse de fnc.
- Exemple :
  - f={(1|->{3}), (2|->{1,2}), (3|->{3})}
  - rel(f)={(1|->3), (2|->1), (2|->2), (3|->3)}

Suites

- Le langage B permet de manipuler des suites (« sequences »). Une suite d'éléments d'un ensemble E est une fonction totale d'un intervalle d'entiers 1..n dans E.
- L'ensemble des suites d'un ensemble E est noté seq(E), l'ensemble des suites non vides seq1(E), l'ensemble des suites injectives est noté iseq(E), l'ensemble des suites injectives non vides est noté iseq1(E) l'ensemble des suites bijectives (permutations) est noté perm(E)
- Exemple : E={e1,e2,e3,e4}
  - [ ] ∈seq(E) (suite vide)
  - [e2,e1,e2,e3] ∈ seq(E) et à seq1(E)
  - [e3,e2,e4] ∈ iseq(E) et à iseq1(E)
  - [e3,e2,e4,e1] ∈ perm(E)

Opérations sur les suites

- size(S) : nombre d'éléments : size([e3,e2,e4,e1])=4
- first(S) : premier élément : first([e3,e2,e4,e1])=e3
- last(S) : dernier élément : last([e3,e2,e4,e1])=e1
- front(S) : S privée de son dernier élément : front([e3,e2,e4,e1])=[e3,e2,e4]
- tail(S) : S privée de son premier élément : tail([e3,e2,e4,e1])=[e2,e4,e1]
- rev(S) : S dans l'ordre inverse : rev([e3,e2,e4,e1])=[e1,e4,e2,e3]

Opérations sur les suites

- ^ : concaténation [e3,e4]^[e1,e2]=[e3,e4,e1,e2]
- -> : insertion d’un élément en tête :
  e7->[e3,e4]=[e7,e3,e4]
- <- : insertion d’un élément en queue :
  [e3,e4]<-e1=[e3,e4,e1]
- /|\ n : restriction aux n éléments en tête
  [e3,e4,e1,e2]/|\ 2 = [e3,e4]
- \|/ n : restriction aux n éléments en queue
  [e3,e4,e1,e2]\|/ 2 = [e1,e2]
- conc : concaténation généralisée :
  conc([e3,e4],[ ],[e6,e5],[e9])=[e3,e4,e6,e5,e9]

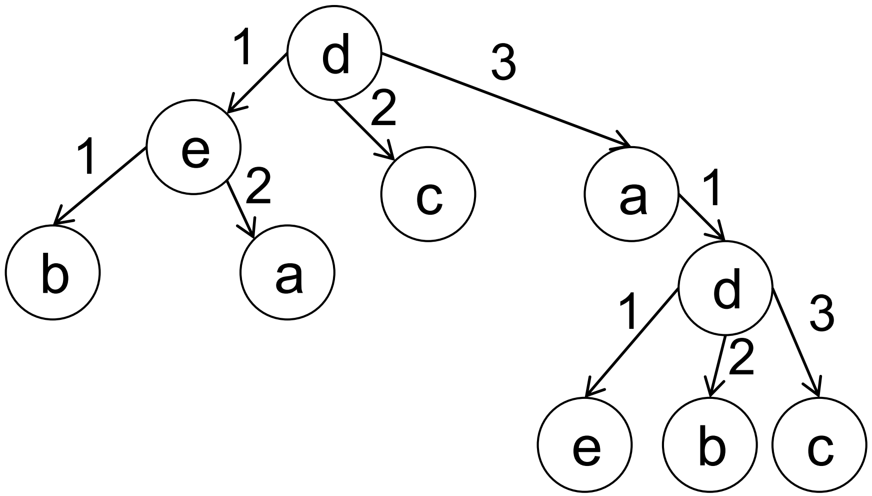
Arbres

- Le B permet également de manipuler des arbres (tree) dont le cas particulier des arbres binaires (btree).
- Si E est un ensemble tree(E) est l'ensemble des arbres décorés d'éléments de E : à chaque nœud est associé un élément de E.
- Les nœuds sont reliés entre eux par des arcs orientés numérotés nommés branches.
- Si une branche va du nœud A au nœud B, le nœud A est dit père de B et le nœud B fils de A.
- Un nœud a exactement un père, à l'exception d'un seul qui a 0 père appelé racine de l'arbre.
- Un arbre possède toujours au moins sa racine (pas d'arbre vide).
- Un nœud peut avoir de 0 à n fils (le nombre de fils s'appelle arité du nœud). L'ordre des fils est significatif. Les nœuds à 0 fils sont appelés feuilles.

Arbres

- Un arbre sur E est modélisé comme une fonction de l'ensemble des suites d'entiers non nuls dans E.
- Ainsi un nœud est repéré par la suite des numéros de branche qui y conduisent ([ ] pour la racine).

Exemple : si E={a,b,c,d,e} et si l'on appelle A l'arbre suivant :



A={([ ]->d),([1]->e),([2]->c),([3]->a),([1,1]->b),([1,2]->a),([3,1]->d),([3,1,1]->e),([3,1,2]->b), ([3,1,3]->c)}