Fonctions sur les ensembles revisitées...

par Michel J Dumontier

Des fonctions utiles sur les ensembles :

Les fonctions que l'on trouve généralement dans la littérature, ne correspondent pas toujours à ce que l'on attend d'elles, ou ne sont pas toujours exactes.

Ainsi on est parfois amené à les réécrire pour son propre usage et pour des applications concrètes.

C'est en lisant l'article de Joseph de Kerf dans le n° 22 des 'Nouvelles d'APL', que j'ai eu l'idée de présenter quelques fonctions sur les ensembles que j'ai dû réécrire.

Nous écrirons des fonctions qui s'appliquent essentiellement à des vecteurs homogènes.

Liste des fonction étudiées : intersection, contenance, sauf (=sans ou complément), union, unique (=nub), différence, égale, identique. (sans oublier l'appartenance qui est une fonction ensembliste mais primitive d'APL)

Conventions:

- Nous suffixerons les noms des fonctions que l'on trouve généralement dans la littérature par la lettre G comme dans INTERG.
- Nous suffixerons les noms des fonctions de Joseph de Kerf par DK comme dans INTERDK.
- Nous suffixerons les noms des fonctions que j'ai dû réécrire par D comme dans INTERD (car INTERG ne suffit pas).
- Les fonctions non suffixées sont aussi des fonctions que j'ai dû écrire, mais je ne sais pas si elles existent déjà dans la littérature (probablement souvent): c'est connu que chacun réécrit ce qui existe déjà sans le savoir!

Remarque: On s'attend à ce que certaines fonctions (exemples: réunion, intersection) soient symétriques (pour A et B étant deux ensembles A inter B doit être égal à B inter A). Un ensemble est une collection d'objets, il n'y a pas forcément une relation d'ordre possible, et l'ensemble (A,B,C) est égal à l'ensemble (B,A,C).

D'autre part, en APL, l'argument gauche a un rôle particulier (déterminant).

Exemple : pour la fonction d'appartenance, la forme du résultat a la forme de l'argument gauche :

Appartenance: (fonction primitive d'APL)

```
      2∈3 4 5
0
      2 3∈ 3 4 5
0 1
```

`1○X` Fonction sinus
`2○X` Fonction cosinus

```
      2!7 Fonction Combinaisons
21
      2.5!5.5 Fonction Béta complète
14.4375
```

Intersection:

Donnons la fonction d'intersection exacte:

```
    ∇ Z←L INTERDK R
[1]   Z←((Z⍳Z)=⍳⍴Z)/Z←(L←,L),R←,R
[2]   Z←((+/Z∘.=L)⌊+/Z∘.=R)/Z
    ∇
```

```
      'CHIQUE' INTERDK 'NICHE'
CHIE
      'NICHE' INTERDK 'CHIQUE'
ICHE
```

Les éléments du résultat sont dans le même ordre que celui de l'argument gauche.

Voici une autre version qui se comporte de la même manière:

```
    ∇ Z←A INTERD B;BB;I;⎕IO;R
[1]   ⎕IO←0 ◇ Z←0↑A
[2]   BO:I←B⍳R←1↑A ◇ →(I=⍴B)/INC ◇ BB←(⍴B)⍴1 ◇ BB[I]←0 ◇ B←BB/B
◇ Z←Z,R
[3]   INC:A←1↓A ◇ →(0≠⍴A)/BO
    ∇
```

Donnons ici la fonction d'intersection que l'on trouve généralement:

```
    ∇ Z←X INTERG Y
[1]   Z←(X∈Y)/X
    ∇
```

Voyons son comportement vis à vis d'une fonction qui donne l'intersection stricte:

```
      'AMSTRAMGRAM' INTERD 'RAMSES'
AMSR
      'AMSTRAMSGRAMMES' INTERD 'RAMSES'
AMSSRE
```

Il y a trois S à gauche, deux à droite: l'intersection est bien deux S.

```
      'AMSTRAMGRAM' INTERG 'RAMSES'
AMSRAMRAM
```

Bien que ce ne soit pas une intersection stricte, la version INTERG peut servir si on veut conserver dans l'argument gauche tous les éléments appartenant à l'argument droit.

Contenance:

Voici ce qu'on trouve généralement pour la contenance, avec le test qui suit:

```
    ∇ Z←X CONTIENTG Y
[1]   Z←∧/Y∈X
    ∇

      'NICHE' CONTIENTG 'CHIEN'
1
      'NICHE' CONTIENTG 'CHIENNE'
1
```

Bien que la deuxième proposition paraisse juste dans la réalité, elle est fausse, à moins de considérer les éléments de l'argument gauche comme génériques: en fait, il manque un N et un E dans l'argument gauche pour former 'CHIENNE'.

Réécrite, la fonction suivante fonctionne plus correctement (en s'inspirant de la fonction DIFFERENCE de Joseph de Kerf):

```
    ∇ Z←L CONTIENTD R;D
[1]   Z←((Z⍳Z)=⍳⍴Z)/Z←(L←,L),R←,R
[2]   Z←((×D)×D←(+/Z∘.=L)-+/Z∘.=R)/Z
[3]   Z←~0∈D≥0
    ∇
      'NICHE' CONTIENTD 'CHIENNE'
0
      'CHIENNE' CONTIENTD 'NICHE'
1
```

(Cette fois-ci, la deuxième proposition ne paraît pas possible dans la réalité, alors que le résultat est vrai!)

Sauf: (peut s'appeler aussi **sans, complément)**

Prenons par exemple la version SANS de l'article cité plus haut et baptisons-la SAUFG, car c'est ainsi qu'on la trouve généralement.
Voici la fonction avec un exemple:

```
    ∇ Z←A SAUFG B
[1]   Z←(~A∈B)/A←,A
    ∇

      'RUSSIA' SAUFG 'U.S.S.R.'
IA
```

On ne tient pas compte dans cette version des éléments (comme ici le point) qui existent dans l'argument droit et pas dans l'argument gauche.
Par contre, SAUFG peut servir comme dans l'exemple suivant :

```
      'U.S.A.' SAUFG '.,-'
USA
```

Union:

La version générale (imparfaite) utilise la fonction ensembliste courante SAUFG.
Voici la fonction UNIONG et un exemple:

```
    ∇ Z←A UNIONG B
[1]   Z←A,B SAUFG A
    ∇

      3 2 1 UNIONG 2 3 4 2
3 2 1 4
```

Ce n'est pas une UNION au sens ensembliste (il manque une fois l'élément 2 dans le résultat).
Donnons maintenant les deux fonctions correctes UNIONDK et UNIOND avec le même exemple:

```
    ∇ Z←L UNIONDK R
[1]   Z←((Z⍳Z)=⍳⍴Z)/Z←(L←,L),R←,R
[2]   Z←((+/Z∘.=L)⌈+/Z∘.=R)/Z
    ∇

    ∇ Z←B UNIOND A;C;I;J;⎕IO
[1]   ⎕IO←0 ⋄ I←0 ⋄ A←,A ⋄ Z←B←,B ⋄ →(0=⍴A)/FIN
[2]   BO:J←B⍳A[I] ⋄ →(J=⍴B)/INC ⋄ C←(⍴B)⍴1 ⋄ C[J]←0 ⋄ B←C/B ⋄
A←(I↑A),(I+1)↓A ⋄
      I←I-1
[3]   INC:→((⍴A)=I←I+1)/FIN ⋄ →((0≠⍴B)∧0≠⍴A)/BO
[4]   FIN:Z←Z,A
    ∇


      3 2 1 UNIONDK 2 3 4 2
3 2 2 1 4
      3 2 1 UNIOND 2 3 4 2
3 2 1 4 2
```

Elles fonctionnent différemment mais elles donnent chacune le même résultat, juste (à une permutation près des éléments, mais ce sont les mêmes bons éléments: dans la version DK, les éléments identiques sont rassemblés; dans la version D, les éléments apparaissent dans l'ordre où on les rencontre dans l'argument gauche puis dans l'argument droit).

Pour reprendre un exemple donné dans ce même article:

```
      'ABRACADABRA' UNIONDK 'YABADABADOE'
AAAAABBRRCDDYOE
      'ABRACADABRA' UNIOND 'YABADABADOE'
ABRACADABRAYDOE
```

Unique:

Ce n'est pas à proprement parler une fonction ensembliste qu'on pourrait baptiser réduction d'un ensemble, mais c'est une fonction auxiliaire très commode que l'on trouve souvent dans la littérature sous le nom de NUB (noyau); on la trouve même sous forme de fonction primitive en J appelée ESSENTIEL. Il y a même mieux en J: sa compagne la fonction CRIBLE DE L'ESSENTIEL, très utile.

Voici NUB avec un exemple sans commentaire, cela va de soi:

```
    ∇ Z←NUB X
[1]   Z←((⍳⍴X)=X⍳X)/X
    ∇
      NUB 'SUISSESSES'
SUIE
```

Différence:

Rappelons la fonction DIFFERENCEDK sur laquelle il n'y a rien à dire et donnons un exemple**:**

```
    ∇ Z←L DIFFERENCEDK R;D
[1]   Z←((Z⍳Z)=⍳⍴Z)/Z←(L←,L),R←,R
[2]   Z←((D>0)×D←(+/Z∘.=L)−+/Z∘.=R)/Z
    ∇
      'ABRACADABRA' DIFFERENCEDK 'ARCHEDENOE'
AAAABBR
```

Voici une fonction qui tient compte des éléments de l'argument gauche et qui donne une valeur réelle de la différence entre les éléments de l'argument gauche et ceux de l'argument droit.

```
    ∇ Z←L DIFFENSD R;D
[1]   Z←((Z⍳Z)=⍳⍴Z)/Z←(L←,L),R←,R
[2]   Z←((×D)×D←(+/Z∘.=L)−+/Z∘.=R)/Z
[3]   (D≠0)/D
    ∇
      'ABRACADABRA' DIFFENSD 'ARCHEDENOE'
4 2 1 ¯1 ¯3 ¯1 ¯1
AAAABBRHEEENO
```

Et voici une dernière version qui utilise la fonction NUB et qui donne au dessus du résultat qui ne comporte qu'une seule occurrence de chaque élément, la valeur de la différence (autant d'éléments que de valeurs de pondération):

```
    ∇ Z←L DIFFENSND R;D
[1]   Z←((Z⍳Z)=⍳⍴Z)/Z←(L←,L),R←,R
[2]   Z←NUB ((×D)×D←(+/Z∘.=L)−+/Z∘.=R)/Z
[3]   (D≠0)/D
    ∇

'ABRACADABRA' DIFFENSND 'ARCHEDENOE'
4 2 1 ¯1 ¯3 ¯1 ¯1
ABRHENO
```

Interprétation: il y a 4 éléments 'A' en excédant dans l'argument gauche par rapport à l'argument droit, 2 éléments 'B' etc... un 'H' manquant, 3 'E' manquant etc.. il y a autant de D, ils se compensent et disparaissent dans la différence! ce qui est normal.

Égale:

On peut aussi avoir besoin de savoir si deux objets sont strictement égaux:

Voici sa définition ainsi que deux exemples pour illustrer son utilisation:

```
    ∇ Z←A EGALE B;D;E
[1]   D←⍴A ◇ E←⍴B ◇→(D=E)/SUI ◇ Z←0 ◇ →0
[2]   SUI:Z←∧/A=B
[3]
    ∇

      F←'CHAT'
      E←'CAT'
      F EGALE 'CHAT'
1
      E EGALE 'CHAT'
0
```

Identique:

Nous dirons que deux ensembles sont identiques s'ils sont composés des mêmes éléments quel que soit le nombre d'occurrences de chaque élément dans chaque ensemble.

Voici la fonction ainsi que deux exemples pour illustrer son utilisation:

```
    ∇ Z←A IDENTIQUE B
[1]   Z←∧/(A∈B),B∈A
    ∇

      'ISO.' IDENTIQUE 'O.S.I.'
1
      ETHANAMIDE←'CHHH-CO-NHH'
      'CHON-' IDENTIQUE ETHANAMIDE
1
```

La fonction étant symétrique, on peut l'utiliser également comme suit :

```
      ETHANAMIDE IDENTIQUE 'CHON-'
1
```