

MIF04 – GESTION DE DONNÉES POUR LE WEB

TD – Algèbres de collections

Exercice 1 : Typage En supposant les types suivants pour les *builtins* :

- $(>) : int \rightarrow int \rightarrow bool$
- $(+) : int \rightarrow int \rightarrow int$
- $sum : arr(int) \rightarrow int$

Donner le type des expressions suivantes :

1. $\lambda r. Map_{\lambda x. (\{A:x.A, D:x.B+x.C\})} (Filter_{\lambda x. (x.A > x.C)}(r))$
2. $\lambda r. Agg_{sum, \lambda x. (x.A), B} (Filter_{\lambda x. (x.C > 0)}(r))$
3. $\lambda r. Map_{\lambda x. (\{D:x.left.A, E:x.right.C\})} (Join_{\lambda y. \lambda z. (y.B=z.B)}(r)(r))$

Exercice 2 : Sous-typage

1. Expliquer, en vous appuyant éventuellement sur des exemples, le fonctionnement des règles de sous-types des records (AddField) et (SubField).
2. Expliquer en quoi la règle de sous-type (AddField) est compatible avec le système de types (i.e. on n'accède pas à des champs de records qui n'existent pas).

Exercice 3 : Algèbre en SQL On suppose une relation $R(A, B, C)$ représentée par une collection R de type $\langle A : int, B : int, C : int \rangle$. Pour chacune des requêtes d'algèbre de collection suivantes, donner un équivalent en SQL.

1. $Map_{\lambda x. (\{A:x.A, D:x.B+x.C\})} (Filter_{\lambda x. (x.A > x.C)}(R))$
2. $Agg_{sum, \lambda x. (x.A), B} (Filter_{\lambda x. (x.C > 0)}(R))$
3. $Map_{\lambda x. (\{D:x.left.A, E:x.right.C\})} (Join_{\lambda y. \lambda z. (y.B=z.B)}(R)(R))$

Exercice 4 : SQL en Algèbre On suppose une relation $R(A, B, C)$ représentée par une collection R de type $\langle A : int, B : int, C : int \rangle$. Pour chacune des requêtes SQL suivantes, donner un équivalent en algèbre de collections, en Javascript puis en MongoDB aggregation pipeline.

1.

```
SELECT *
FROM R
WHERE R.A = R.B + R.C
```

2.

```
SELECT SUM(A) as D, B, C
FROM R
GROUP BY B, C
ORDER BY B, C
```

3.

```
SELECT R.A, R.C, R2.D
FROM R
JOIN (SELECT count(R.A) as D, C
      FROM R
      GROUP BY C) R2
ON R.C = R2.C
```

4.

```
SELECT R1.C, SUM(R1.A)+COUNT(R2.B) as D
FROM R R1 JOIN R R2 ON R1.C = R2.C
GROUP BY R1.C
HAVING R1.C < COUNT(R1.B)
```

Typage des fonctions

$$\text{(App)} \frac{\Gamma \vdash f : \tau \rightarrow \tau' \quad \Gamma \vdash d : \tau}{\Gamma \vdash f(d) : \tau'}$$

$$\text{(Lambda)} \frac{\Gamma[x : \tau] \vdash d : \tau'}{\Gamma \vdash \lambda x. d : \tau \rightarrow \tau'}$$

$$\text{(Var)} \frac{}{\Gamma \vdash x : \tau} \text{ si } x : \tau \in \Gamma$$

Typage des records, des listes

$$\text{(Field)} \frac{\Gamma \vdash d : \langle a : \tau \rangle}{\Gamma \vdash d.a : \tau}$$

$$\text{(Record)} \frac{\Gamma \vdash d_1 : \tau_1 \quad \dots \quad \Gamma \vdash d_n : \tau_n}{\Gamma \vdash \{a_1 : d_1, \dots, a_n : d_n\} : \langle a_1 : \tau_1, \dots, a_n : \tau_n \rangle}$$

$$\text{(Singleton)} \frac{\Gamma \vdash d : \tau}{\Gamma \vdash [d] : \text{arr}(\tau)}$$

$$\text{(Empty)} \frac{}{\Gamma \vdash [] : \text{arr}(\tau)}$$

Types des constantes

$$\text{(Const)} \frac{}{\Gamma \vdash c : \tau}$$

en prenant τ et c comme suit :

- type *int* : 1 , 2 , ...
- type *float* : 1.0 , 0.3 , 10.42 , ...
- type *string* : "truc" , ...

Sous-typage

$$\text{(Sous-typage)} \frac{\Gamma \vdash d : \tau \quad \tau \preceq \tau'}{\Gamma \vdash d : \tau'}$$

$$\text{(Refl)} \frac{}{\tau \preceq \tau}$$

$$\text{(Trans)} \frac{\tau \preceq \tau' \quad \tau' \preceq \tau''}{\tau \preceq \tau''}$$

$$\text{(AddField)} \frac{}{\langle a_1 : \tau_1, \dots, a_k : \tau_k, a_{k+1} : \tau_{k+1} \rangle \preceq \langle a_1 : \tau_1, \dots, a_k : \tau_k \rangle}$$

$$\text{(SubField)} \frac{\tau_k \preceq \tau'_k}{\langle a_1 : \tau_1, \dots, a_k : \tau_k \rangle \preceq \langle a_1 : \tau_1, \dots, a_k : \tau'_k \rangle}$$

$$\text{(SubList)} \frac{\tau \preceq \tau'}{\text{arr}(\tau) \preceq \text{arr}(\tau')}$$

$$\text{(SubFunc)} \frac{\tau_1 \succeq \tau'_1 \quad \tau_2 \preceq \tau'_2}{\tau_1 \rightarrow \tau_2 \preceq \tau'_1 \rightarrow \tau'_2}$$

FIGURE 1 – Typage

$Map_f : arr(\tau) \rightarrow arr(\tau')$ avec $f : \tau \rightarrow \tau'$	$Agg_{f,g,a} : arr(\tau)$ $\rightarrow arr(\langle key : \tau_a, value : \tau'' \rangle)$ avec $f : arr(\tau') \rightarrow \tau''$, $g : \tau \rightarrow \tau'$ et $\tau \preceq \langle a : \tau_a \rangle$
$Filter_f : arr(\tau) \rightarrow arr(\tau)$ avec $f : \tau \rightarrow bool$	$FlatMap_f : arr(\tau) \rightarrow arr(\tau')$ avec $f : \tau \rightarrow arr(\tau')$
$Join_f : arr(\tau_1) \rightarrow arr(\tau_2)$ $\rightarrow arr(\langle left : \tau_1, right : \tau_2 \rangle)$ avec $f : \tau_1 \rightarrow \tau_2 \rightarrow bool$	$Sort_f : arr(\tau) \rightarrow arr(\tau)$ avec $f : \tau \rightarrow \tau \rightarrow bool$

FIGURE 2 – Types des opérateurs de l'algèbre

En MongoDB, on peut appeler le framework d'agrégation via `db.collection.aggregate(...)`
Voici quelques *stages* utilisables dans le pipeline :

\$project spec : ensemble de spécifications de champs. Permet d'ajouter, supprimer ou changer la valeurs de certains champs.

\$match spec : conditions sur les champs (c.f. query Mongo). Filtre les éléments du pipeline à la façon d'une requête MongoDB classique.

\$lookup spec :

```
{
  from: <collection to join>,
  localField: <field from the input documents>,
  foreignField: <field from the documents of the "from" collection>,
  as: <output array field>
}
```

Effectue une jointure avec la collection spécifiée, mais en combinant chaque docuemnt de la collection courante avec le tableau des documents correspondants.

\$unwind spec : chemin de champs avec \$ au début. Produit un document pour chaque valeur du tableau contenu dans le champ spécifié en recopiant le reste du document.

\$group spec :

```
{
  _id: <expression>, // Group By Expression
  <field1>: { <accumulator1> : <expression1> },
  ...
}
```

Regroupe les documents par valeur décrite dans `_id` et calcule pour chaque champ la valeur agrégée décrite par l'accumulateur auquel on passe les valeurs calculées par l'expression.

\$sort spec : ensemble de champs avec le ordre de tri (1 ou -1)

Réf : <https://docs.mongodb.com/manual/reference/operator/aggregation-pipeline/>

FIGURE 3 – Agrégation pipeline en MongoDB