Pour essayer de démontrer la complétude au sens du premier critère, formel, j'ai essayé de commencer par démontrer des résultats plus faibles, des cas particuliers.

Il s'agit des cas particuliers des sur-requêtes et sous-requêtes. En effet, on peut définir une relation d'ordre sur les requêtes et on peut définir une notion de sur-requête en procédant comme suit :

En voyant les requêtes C2QL comme des mots sur l'alphabet

$$\Sigma = \{\sigma_p \,/p \text{ un prédicat}\} \cup \{\pi_\delta \,/\delta \subset \Delta\} \cup \{\text{frag}_\delta \,/\delta \subset \Delta\} \cup \{\text{count}_\delta \,/\delta \subset \Delta\} \\ \cup \{\text{crypt}_\alpha \,/\alpha \in \Delta\} \cup \{\text{decrypt}_\alpha \,/\alpha \in \Delta\} \cup \{\text{join}, \text{defrag}\}$$

on définit la relation d'ordre $\leq$ en disant qu'une requête $u = u_1.\ldots.u_n$ est inférieure à une requête v s'il existe des fonctions $f_1, \ldots, f_{n+1}$ de $\Sigma \cup \text{id}$ telles que $v = f_1.u_1.f_2 \ldots.u_n.f_{n+1}$

et on dit alors que u est une sous-requête de v *et v est une sur-requête de u.

Les cas particuliers que j'ai alors essayé de démontrer sont les suivants :

Lemme 1 *A partir d'une requête Q on peut, en appliquant les lois algébriques dont on dispose, en dériver toutes les sous-requêtes de Q qui soient sémantiquement équivalentes à Q.*

Lemme 2 *A partir d'une requête Q on peut, en appliquant les lois algébriques dont on dispose, en dériver toutes les sur-requêtes de Q qui soient sémantiquement équivalentes à Q.*

Les deux lemmes sont en fait équivalents, car toutes les lois dont on dispose établissent des équivalence, donc "vont dans les deux sens".

Pour démontrer ces deux lemmes (puisqu'ils sont équivalents je choisis d'essayer de démontrer le premier) je voudrais raisonner par récurrence forte sur la différence entre le nombre de fonctions de la requête et le nombre de fonctions de sa sous-requête.

L'initialisation, lorsqu'il y a le même nombre de fonctions, ne pose pas problème : la requête est alors égale à la sous-requête en question.

Pour l'hérédité, je choisit l'une des fonctions qui sont présentes dans la requête initiale mais pas dans la sous-requête et je voudrais donc montrer qu'en appliquant les lois je peux faire disparaître au moins cette fonction (et éventuellement d'autres au même temps, comme ce serait le cas pour la fonction frag) en restant sur-requête de la sous-requête considérée.

Pour ce faire, je voudrais raisonner par disjonction de cas sur la nature de la fonction à faire disparaître :

Pour les sélections Deux sous-cas se présentent.

Soit il n'y a pas d'autres sélections dans la requête, dans lequel cas vu qu'il y a une sous-requête ou cette sélection n'apparaît pas et ou le résultat renvoyé est le même pour toute relation, la sélection porte sur une tautologie et on peut la faire disparaître directement.

Soit il y a d'autres sélections dans la requête, dans lequel cas je me dis (**MAIS JE N'AI PAS RÉUSSI À LE PROUVER**) que le prédicat de la sélection en question doit être une conséquence des prédicats des autres sélections, et en permutant les fonctions pour mettre les différentes sélections côte à côte puis on les fusionnant en une seule puis en les re-séparant et en re-commutant, on doit pouvoir faire disparaître la sélection en question.

C'est ici que j'ai décidé d'abandonner cette approche.