Algorithm 4 - Producing rewritings

Input: A query Q and a list of lists of CSDs I.

Output: A set of rewritings R that matches with the query and fulfill the user preferences.

```
function $ProduceRewritings(Q, I)$
$R \leftarrow \emptyset$
$\mathcal{T}_{\text{init}}[\![\ \mathcal{A}gg(Q)\ ]\!]$
$p \leftarrow I.next()$
while $p \neq \emptyset$ and $\mathcal{T}_{\text{cond}}[\![\ \mathcal{A}gg(Q)\ ]\!]$ do
    if $isRewriting(Q, p)$ then
        $R \leftarrow R \cup Rewriting(p)$
        $\mathcal{T}_{\text{inc}}[\![\ \mathcal{A}gg(Q)\ ]\!]$
    end if
    $p \leftarrow I.next()$
end while
return $R$
end function
```

Algorithm 5 - Validating a combination of CSDs

Input: A query Q and a set of candidate services descriptions p.

Output: A boolean value. *True*, if the set p is a rewriting of the query. *False*, otherwise.

```
function $isRewriting(Q, p)$
let $p = \{CSD_1, CSD_2, ..., CSD_k\}$
if (a) The number of elements in the union $CSD_1.G_1 \cup CSD_2.G_2, ..., \cup CSD_k.G_k$
   is equal to the number of abstract services in $Q$
   (b) The intersection $CSD_1.G_1 \cap CSD_2.G_2, ..., \cap CSD_k.G_k$ is empty then
    return $true$
end if
return $false$
end function
```