Outils logiques et algorithmiques – TD 7 – Correction

Exercice 1

Exercice 2

Cas de base $n = 1$. Un quadrillage de côté 2 auquel on retire une case est couvert par exactement un triomino.

Récurrence On suppose qu'un quadrillage de côté 2^n auquel on retire une case peut toujours être pavé par des triominos. Montrons que c'est également le cas d'un quadrillage de côté 2^{n+1} auquel on retire une case. Un quadrillage de côté 2^{n+1} est formé de quatre carrés de côté 2^n. La case retirée est dans exactement un de ces quatre carrés. On peut retirer une case à chacun des trois autres carrés en plaçant un triomino au centre du quadrillage de côté 2^{n+1}. On obtient alors un triomino et quatre carrés de côté 2^n auxquels une case a été retirée. Par hypothèse de récurrence, ces quatre carrés peuvent être pavés par des triominos donc le quadrillage initial aussi.

Exercice 3 L'étape d'hérédité qui fait passer du cas n au cas $n+1$ n'est correcte que si $n \geq 2$. En effet dans ce cas on regarde l'ensemble des crayons $2..(n+1)$ qui sont tous de la même couleur et de même pour les crayons $1..n$. Tous les crayons ont donc la même couleur que le crayon numéro 2. Mais si il n'y a que deux crayons (cas $n = 1$), l'hypothèse de récurrence dit que le crayon 2 a la même couleur que lui-même et que le crayon 1 a la même couleur que lui-même, mais rien ne permet de conclure que ces deux crayons sont de la même couleur.

Exercice 4 Par récurrence sur ℓ.

— Cas `[]`.

$$\mathtt{count}_2(e, \mathtt{[]}, n) = n = n + 0 = n + \mathtt{count}_1(e, \mathtt{[]})$$

— Cas $x :: \ell$ avec hypothèse de récurrence $\mathtt{count}_2(e, \ell, n) = n + \mathtt{count}_1(e, \ell)$.
Si $x = e$, alors

$$\begin{array}{rcll} \mathtt{count}_2(e, x :: \ell, n) & = & 1 + \mathtt{count}_2(e, \ell, n) & \text{déf. de } \mathtt{count}_2 \\ & = & 1 + (n + \mathtt{count}_1(e, \ell)) & \text{hyp. de récurrence} \\ & = & n + (1 + \mathtt{count}_1(e, \ell)) & \\ & = & n + \mathtt{count}_1(e, x :: \ell) & \text{déf. de } \mathtt{count}_1 \end{array}$$

Si à l'inverse $x \neq e$, alors

$$\begin{array}{rcll} \mathtt{count}_2(e, x :: \ell, n) & = & \mathtt{count}_2(e, \ell, n) & \text{déf. de } \mathtt{count}_2 \\ & = & n + \mathtt{count}_1(e, \ell) & \text{hyp. de récurrence} \\ & = & n + \mathtt{count}_1(e, x :: \ell) & \text{déf. de } \mathtt{count}_1 \end{array}$$

Dans tous les cas

$$\mathtt{count}_2(e, x :: \ell, n) \quad = \quad n + \mathtt{count}_1(e, x :: \ell)$$

Conclusion : la propriété est vraie pour toute liste. *On en déduit que* $\mathtt{count}_2(e, \ell, 0)$ *compte bien le nombre d'occurrences de e dans* ℓ.

Exercice 5

1. Par récurrence sur ℓ.

   — Cas `[]`.

$$\mathtt{longueur(rev([]))} \quad = \quad \mathtt{longueur([])}$$

— Cas $x :: \ell$ avec hypothèse de récurrence sur ℓ.

$$\begin{array}{rcll}
\texttt{longueur}(\texttt{rev}(x :: \ell)) & = & \texttt{longueur}(\texttt{concat}(\texttt{rev}(\ell), x :: \texttt{[]})) & \\
& = & \texttt{longueur}(\texttt{rev}(\ell)) + \texttt{longueur}(x :: \texttt{[]}) & \text{prop. } \texttt{longueur/concat} \\
& = & \texttt{longueur}(\ell) + \texttt{longueur}(x :: \texttt{[]}) & \mathit{HR} \\
& = & \texttt{longueur}(\ell) + 1 & \\
& = & \texttt{longueur}(x :: \ell) &
\end{array}$$

2. Par récurrence sur ℓ_1.

   — Cas [].

$$\begin{array}{rcl}
\texttt{rev}(\texttt{concat}(\texttt{[]}, \ell_2)) & = & \texttt{rev}(\ell_2) \\
& = & \texttt{concat}(\texttt{rev}(\ell_2), \texttt{[]}) \\
& = & \texttt{concat}(\texttt{rev}(\ell_2), \texttt{rev}(\texttt{[]}))
\end{array}$$

   — Cas $x :: \ell$ avec hypothèse de récurrence sur ℓ.

$$\begin{array}{rcll}
\texttt{rev}(\texttt{concat}(x :: \ell, \ell_2)) & = & \texttt{rev}(x :: \texttt{concat}(\ell, \ell_2)) & \\
& = & \texttt{concat}(\texttt{rev}(\texttt{concat}(\ell, \ell_2)), x :: \texttt{[]}) & \\
& = & \texttt{concat}(\texttt{concat}(\texttt{rev}(\ell_2), \texttt{rev}(\ell)), x :: \texttt{[]}) & \mathit{HR} \\
& = & \texttt{concat}(\texttt{rev}(\ell_2), \texttt{concat}(\texttt{rev}(\ell), x :: \texttt{[]})) & \text{prop. } \texttt{concat/concat} \\
& = & \texttt{concat}(\texttt{rev}(\ell_2), \texttt{rev}(x :: \ell)) &
\end{array}$$

3. Par récurrence sur ℓ.

   — Cas [].

$$\begin{array}{rcl}
\texttt{rev}(\texttt{rev}(\texttt{[]})) & = & \texttt{rev}(\texttt{[]}) \\
& = & \texttt{[]}
\end{array}$$

   — Cas $x :: \ell$ avec hypothèse de récurrence sur ℓ.
   Remarquons d'abord que $\texttt{rev}(x :: \texttt{[]}) = \texttt{concat}(\texttt{rev}(\texttt{[]}), x :: \texttt{[]}) = \texttt{concat}(\texttt{[]}, x :: \texttt{[]}) = x :: \texttt{[]}$.

$$\begin{array}{rcll}
\texttt{rev}(\texttt{rev}(x :: \ell)) & = & \texttt{rev}(\texttt{concat}(\texttt{rev}(\ell), x :: \texttt{[]})) & \\
& = & \texttt{concat}(\texttt{rev}(x :: \texttt{[]}), \texttt{rev}(\texttt{rev}(\ell))) & \text{question précédente} \\
& = & \texttt{concat}(x :: \texttt{[]}, \texttt{rev}(\texttt{rev}(\ell))) & \text{remarque} \\
& = & \texttt{concat}(x :: \texttt{[]}, \ell) & \mathit{HR} \\
& = & x :: \texttt{concat}(\texttt{[]}, \ell) & \\
& = & x :: \ell &
\end{array}$$

Exercice 6

1. Problème majeur : appeler à chaque fois concat, dont le coût est proportionnel à la longueur de son premier argument, donne un coût total de rev proportionnel au carré de la longueur de la liste renversée. On pouvait également reprocher à cette version précédente de n'être pas récursive terminale, et donc d'être compilée bien moins efficacement.
2. On donne une unique équation pour rev_rt, et deux équations pour rev_append pour distinguer le cas de la liste vide du cas de la liste non vide.

$$\begin{array}{rcl}
\texttt{rev_rt}(\ell) & = & \texttt{rev_append}(\ell, \texttt{[]})
\end{array}$$

$$\begin{array}{rcl}
\texttt{rev_append}(\texttt{[]}, \ell_2) & = & \ell_2 \\
\texttt{rev_append}(x :: \ell, \ell_2) & = & \texttt{rev_append}(\ell, x :: \ell_2)
\end{array}$$

3. On démontre par récurrence sur ℓ_1 que $\forall \ell_2,\ \texttt{rev\_append}(\ell_1, \ell_2) = \texttt{concat}(\texttt{rev}(\ell_1), \ell_2)$.

— Cas $\texttt{[]}$.

$$\begin{array}{rcl}\texttt{rev\_append}(\texttt{[]}, \ell_2) & = & \ell_2 \\ & = & \texttt{concat}(\texttt{[]}, \ell_2) \\ & = & \texttt{concat}(\texttt{rev}(\texttt{[]}), \ell_2)\end{array}$$

— Cas $x :: \ell$.

$$\begin{array}{rcl}\texttt{rev\_append}(x :: \ell, \ell_2) & = & \texttt{rev\_append}(\ell, x :: \ell_2) \\ & = & \texttt{concat}(\texttt{rev}(\ell), x :: \ell_2) \\ & = & \texttt{concat}(\texttt{rev}(\ell), \texttt{concat}(x :: \texttt{[]}, \ell_2)) \\ & = & \texttt{concat}(\texttt{concat}(\texttt{rev}(\ell), x :: \texttt{[]}), \ell_2) \\ & = & \texttt{concat}(\texttt{rev}(x :: \ell), \ell_2)\end{array}$$

On en déduit donc que $\texttt{rev\_rt}(\ell) = \texttt{rev\_append}(\ell, \texttt{[]}) = \texttt{concat}(\texttt{rev}(\ell), \texttt{[]})$ et on conclut avec la propriété $\texttt{concat}(\ell, \texttt{[]}) = \ell$.