

question 4. On utilise la commande *fst*, qui est déjà implémentée en Mini-ML, pour extraire la première composante du couple.

question 5. Le typage des expressions permet d'effectuer des opérations légales et bien typées. Cela permet aussi d'obtenir le résultat voulu (et non un résultat incohérent si le programme termine).

question 6. Supprimée

question 7. Si on a pu donner dans les premières questions des règles de typage et une sémantique à petit pas de *pred*, il n'existe cependant pas d'implémentation en Mini-ML de cette fonction. En effet, cela supposerait que nous avons à notre disposition des opérateurs d'addition et/ou de soustraction et de comparaison autres que l'égalité, ce qui n'est pas le cas ici.

question 8.

question 9. Règle concernée :

$$\frac{t \text{ value} \quad u \text{ value} \quad t \neq u}{(t = u) \hookrightarrow \text{false}}$$

\neq signifie ici que t et u ont des valeurs différentes ou des types différents (?). On a besoin de ce symbole car sinon, on remplacerait $t \neq u$ par ce qu'on veut démontrer, c'est-à-dire $(t = u) \hookrightarrow \text{false}$, et donc obtenir un arbre de dérivation infini, ce qui est absurde.

question 10. Supprimée.

question 11. Le ML étant un langage fonctionnel, on a besoin de pouvoir définir des fonctions. On doit donc implémenter une structure de donnée qui prend en argument des paramètres, un corps et renvoie un résultat, et ceci ne peut pas être codé directement à partir des opérations données.

question 12.

question 13. La règle de typage de $\&\&$ est la suivante :

$$\frac{\Gamma \vdash u : A \quad \Gamma \vdash v : A}{\Gamma \vdash u \&\& v : \mathbf{bool}}$$

La règle de typage de $+$ est similaire à celle de *plus*, définie en question 2.

question 14. L'instruction *let rec... in...* peut inclure des programmes récursifs qui ne vont pas terminer, et donc des arbres de syntaxe abstraite infinis. L'exclure permet d'éviter ce phénomène.

question 15. On utilise ici une option pour signifier qu'il peut éventuellement y avoir un symbole de type *ty*, mais ce n'est pas nécessaire pour la compilation. L'intérêt d'utiliser le type *ty option ref* et qu'on manipule directement des pointeurs, et des objets mutables, ce qui peut-être utile si on envisage un α -renommage pas exemple.

question 16.

question 17. Faite.

question 18. Faite.

question 19.

question 20.