Exercices: Calcul des prédicats

Lionel Vaux Auclair

Logique
HUGO@Centrale, 2021–2022

Exercice 1. Pour chacune des formules suivantes, trouvez une interprétation qui la valide et une qui la falsifie :

1. $\forall x.\forall y.f(x) = f(y)$;
2. $\forall x.R(x, f(x))$;
3. $\forall x.\exists y.R(x, y)$;
4. $\exists x.\forall y.R(x, y)$.

Exercice 2. Pour chacune des formules suivantes, déterminez s'il s'agit d'une tautologie (avec R un symbole de relation unaire et S un symbole de relation binaire).

1. $\forall x.(\forall y.S(x, y)) \Rightarrow S(x, x)$;
2. $\forall x.\forall y.S(x, y) \Rightarrow S(x, x)$;
3. $\forall x.\exists y.R(y) \Rightarrow R(x)$;
4. $\exists x.\forall y.R(y) \Rightarrow R(x)$.

Exercice 3. On cherche à spécifier un langage de programmation jouet. La trame de notre interprétation est l'ensemble des valeurs que peut prendre une expression dans le langage. On se donne :

— des symboles de constantes : `function`, `type`, `str`, `error` et `undef` ;
— un symbole de fonction : `apply` (binaire : application d'une fonction à un argument),
— des symboles de relations : `=` (binaire : égalité), `hastype` (binaire : le premier argument admet comme type le deuxième argument), `callable` (unaire : on peut appliquer cette valeur à un argument).

On dit d'une valeur qu'elle est :

— définie si elle est différente d'`undef`,
— un type s'il existe une valeur qui l'admet comme type,
— une fonction si elle est de type `function`,
— appelable si elle valide `callable`.

Donnez des formules qui expriment les propriétés suivantes (en supposant chaque fois que les propriétés précédentes sont valides) :

1. `function`, `type`, `error` et `str` sont des types ;
2. `undef` est de type `error` ;
3. chaque valeur admet exactement un type ;
4. en appliquant `type` à une valeur, on obtient son type ;
5. le type d'un type est `type` ;
6. toute fonction peut être appliquée à un argument ;

7. tout type peut être appliqué à un argument ;

8. quand on applique un type à un argument, si le résultat est défini, c'est une valeur de ce type ;

9. si une valeur n'est pas applicable, alors son application à un argument n'est jamais définie.

En supposant que les propriétés précédentes sont validées, montrez que :

10. `type` n'est pas une fonction ;

11. `type` est nécessairement applicable ;

12. si l'application d'une valeur à une autre valeur est définie, la première valeur est applicable.