Prévention du déréférencement de références nulles dans un langage à objets

Jean-Sébastien Gélinas, Étienne M. Gagnon, Jean Privat

Université du Québec à Montréal
gelinas.jean-sebastien@courrier.uqam.ca, {gagnon.etienne_m,privat.jean}@uqam.ca

Résumé. Le déréférencement de références nulles est une erreur de programmation courante dans les langages à objets. Pour la prévenir, certaines approches garantissent statiquement son absence à l'aide de systèmes de types ou d'annotations mais réduisent l'expressivité du langage. D'autres approches analysent plutôt le code source pour identifier les erreurs potentielles, mais peuvent trouver des faux-positifs et ne garantissent pas l'absence d'erreurs à l'exécution. Dans cet article, nous proposons une approche offrant la garantie statique d'absence d'erreur de déréférencement dans une grande portion du code. L'approche consiste en un système de types statiques simple, des vérifications dynamiques et un opérateur de test dynamique. En plus de préserver l'expressivité du langage, notre approche limite la zone de danger à la construction des instances et permet la détection précoce des erreurs à l'exécution. Nos mesures expérimentales démontrent une grande étendue des garanties statiques et l'efficacité de la détection précoce des erreurs.

1 Introduction

Le déréférencement d'une référence nulle est une erreur de programmation fréquente dans les langages à objets. Normalement cette erreur cause l'arrêt prématuré du logiciel en cours d'exécution. Certains langages permettent de récupérer cette erreur avec un système d'exception (`try`/`catch` en JAVA et C#), mais ceci constitue une mesure extraordinaire pour les logiciels qui ne peuvent se permettre d'arrêter. En fait, le problème réel n'est souvent pas dû au déréférencement lui-même ; il est plutôt dû au fait que la variable (ou l'attribut) déréférencé *contient* la valeur `null` à ce moment. Le code erroné se situe donc normalement ailleurs, à un endroit potentiellement lointain où on a soit oublié d'initialiser la variable, soit stocké erronément `null` dans la variable.

Dans cet article, nous présentons le résultat de notre recherche d'une approche à utiliser dans un nouveau langage de programmation à objets pour prévenir le déréférencement de références nulles sans sacrifier, pour autant, l'expressivité du langage.

Il pourrait sembler simple de modifier un système de types similaire à celui des langages JAVA et C# pour indiquer, à même le type, si `null` est une valeur permise ou non. Malheureusement, la construction des objets cause problème car il n'y a généralement pas de valeur non-nulle appropriée à stocker par défaut dans un attribut de type non-nul lors de l'allocation de