# Détection d'objectifs de test polluants pour les critères de flot de données *

Thibault Martin[1] Nikolai Kosmatov[1,2] Virgile Prevosto[1]
Matthieu Lemerre[1]

[1]Université Paris-Saclay, CEA, List, F-91120, Palaiseau, France
firstname.lastname@cea.fr
[2]Thales Research & Technology, Palaiseau, France
nikolaikosmatov@gmail.com

**Contexte** Pour évaluer la qualité d'une suite de tests logiciels, c'est-à-dire s'assurer qu'elle est représentative d'un ensemble de cas d'usage aussi large que possible, de nombreux critères de couverture ont été définis [2]. Parmi ceux-ci, les critères de flot de données, comme all-defs et all-uses, appartiennent aux plus avancés. Ces critères sont définis à partir de la notion de *paire def-use*, composée d'une position où une variable $\mathtt{x}$ est définie et d'une position où $\mathtt{x}$ est utilisée, et telle qu'il existe au moins un chemin reliant la définition à l'utilisation dans le graphe de contrôle du programme sous test. Si un chemin reliant les deux nœuds d'une paire def-use de $x$ ne contient pas de nœud redéfinissant $\mathtt{x}$, on parle d'un chemin sans redéfinition (*def-clear path*).

Comme pour beaucoup d'autres critères, certains objectifs individuels de test sont *polluants* : ils doivent être retirés pour mesurer correctement la couverture de test [3]. On trouve plusieurs types d'objectifs de test polluants pour les critères de flot de données :

- **les objectifs *non applicables*,** où une paire def-use ne peut pas être reliée par un chemin sans redéfinition (i.e. il n'existe aucun chemin permettant d'éventuellement couvrir cet objectif) ;
- **les objectifs *infaisables*,** où une paire def-use peut être reliée par au moins un chemin sans redéfinition, mais aucun de ces chemins n'est activable par un cas de test ;
- **les objectifs *équivalents* (ou *dupliqués*),** qui sont toujours couverts simultanément : il suffit de garder un objectif par classe d'équivalence [1].

*Cette soumission est un résumé étendu d'un article [1], publié à IFM 2020.

1. L'équivalence d'objectifs de test est en effet une relation d'équivalence.

**Problème** Bien que la création d'une liste d'objectifs (candidats) de test pour les critères de flot de données puisse sembler facile, la définition complexe de ces critères, mêlant atteignabilité et chemin sans redéfinition, rend la détection des objectifs polluants difficile. Il est cependant crucial d'éviter de perdre du temps pendant la génération de tests (en essayant de couvrir un objectif polluant) et de mesurer correctement la couverture (en ignorant les objectifs polluants dans le nombre total d'objectifs).

La détection d'objectifs de test polluants pour des critères plus simples a été étudiée précédemment (voir [3,4] pour quelques résultats récents). Cependant l'évaluation et la combinaison de plusieurs techniques d'analyse pour leur détection sur les critères de flot de données —l'objectif de ce travail— n'ont pas été examinées.

**Contributions** Dans ce travail, nous évaluons trois approches pour détecter les objectifs de test polluants pour les critères de flot de données : une analyse de flot de données simple, une analyse de valeurs basée sur l'interprétation abstraite et une analyse de plus faible précondition. Nous avons implanté ces approches dans LTEST [2], une boîte à outils open-source pour le test de programme C [5]. Nous avons ensuite évalué et comparé ces techniques sur différentes études de cas et avons analysé leurs capacités à détecter les objectifs polluants et leurs limites. Nous nous sommes concentrés sur la partie clef des critères de flot de données : les paires def-use. Les capacités de détection que nous avons observées sont différentes de celles des expériences similaires faites précédemment pour d'autres critères.

Enfin, nous proposons une méthode pour une combinaison efficace de plusieurs techniques. Sur la base de nos études de cas, combiner l'analyse statique simple pour détecter à la fois les objectifs non applicables et les objectifs équivalents, avec l'analyse de valeurs pour identifier les objectifs infaisables semble être le meilleur compromis pour une détection rapide et efficace.

**Travaux futurs** Ce travail fournit une comparaison de la capacité de détection des techniques d'analyse. Lors de travaux futurs, il sera nécessaire d'évaluer leurs résultats par rapport au *nombre réel* d'objectifs polluants (ou une sur-approximation calculée en rejouant une suite de tests riche). D'autres améliorations possibles incluent une meilleure prise en charge du langage C (pointeurs et tableaux), l'amélioration des analyses, notamment celle basée sur la plus faible précondition, en ajoutant automatiquement des annotations de code, ainsi que l'extension de notre étude aux objectifs subsumés (c.a.d. impliqués) et à d'autres critères.

*Remerciements.* Ce travail a été partiellement financé par le projet ANR SATOCROSS (grant ANR-18-CE25-0015-01). Nous remercions Sébastien Bardin et les rapporteurs anonymes pour leurs commentaires.

2. disponible sur `https://github.com/ltest-dev/LTest`

## Références


[1] Thibault Martin, Nikolai Kosmatov, Virgile Prevosto, and Matthieu Lemerre. Detection of Polluting Test Objectives for Dataflow Criteria. In *IFM*, 2020.

[2] Paul Ammann and Jeff Offutt. *Introduction to Software Testing*. Cambridge University Press, 2017.

[3] Sébastien Bardin, Mickaël Delahaye, Robin David, Nikolai Kosmatov, Mike Papadakis, Yves Le Traon, and Jean-Yves Marion. Sound and quasi-complete detection of infeasible test requirements. In *ICST*, pages 1–10, 2015.

[4] Michaël Marcozzi, Sébastien Bardin, Nikolai Kosmatov, Mike Papadakis, Virgile Prevosto, and Loïc Correnson. Time to clean your test objectives. In *ICSE*, page 456–467, 2018.

[5] M. Marcozzi, S. Bardin, M. Delahaye, N. Kosmatov, and V. Prevosto. Taming coverage criteria heterogeneity with LTest. In *ICST*, pages 500–507, 2017.