Ecole Doctorale STIC
Sciences & Technologies de l'Information et de la Communication

ED STIC - Proposition de Sujets de Thèse

pour la campagne d'Allocation de thèses 2011

Titre du sujet :	Explorations des capacités des CSP pour la localisation d'erreurs à partir de contre-exemples
Mention de thèse :	Informatique
HDR Directeur de thèse inscrit à l'ED STIC :	Rueher Michel

Co-encadrant de thèse éventuel :

Nom :	Collavizza
Prénom :	Hélène
Email :	helen@polytech.unice.fr
Téléphone :	04 92 96 51 63

Email de contact pour ce sujet :	michel.rueher@gmail.com
Laboratoire d'accueil :	I3S

Description du sujet :

La localisation des erreurs à l'aide de contre-exemples et de traces d'exécution est un problème critique dans le débogage de logiciels. En effet, quand un programme contient
des erreurs, un model-checker fournit un contre-exemple ou une trace d'exécution qui est souvent longue et difficile à comprendre. L'identification des portions de code qui
contiennent des erreurs est donc en général très coûteuse, même pour des programmeurs expérimentés.

Différentes approches ont été proposées pour aider les programmeurs à cette tâche [BNR03, ReR03, GKL04, SQL05, GBC06, ZGG06, GCK06, GSB07, LiL10, MaM11]. Les
approches actuelles sont principalement basées sur des solveurs booléens qui sont bien adaptées aux model-checkers bornés comme CBMC. Notre environnement de
programmation par contraintes pour la vérification de programmes bornés [CRH10, CVR11],

fournit un modèle plus riche que les solveurs booléens. L'objectif de cette thèse est d'exploiter ce modèle pour concevoir et développer un système basé sur la programmation par contraintes, qui fournit une assistance automatisée à la compréhension et la localisation des erreurs dans les programmes C ou Java.

Dans une première étape, on pourra généraliser l'approche introduite dans [MaM11]: au lieu de calculer le nombre maximum de contraintes qui peuvent être satisfaites, on pourra calculer différents sous-ensembles de contraintes incohérentes (resp. cohérentes). Plus précisément, à l'aide du système de contraintes dérivées de la trace d'un contre-exemple et de la post-condition, on peut calculer:

1. Le IIS (jeu infaisabilité irréductibles) [Chin96, Chin01] pour les contraintes linéaires
2. Les "minimum conflict sets" [Jun04] pour les contraintes du CSP.

Une expérimentation sur des benchmarks standard devra être effectuée pour évaluer le coût et l'intérêt pratique de ces informations. En effet, si le calcul des IIS pour des contraintes linéaires peut être fait à un coût raisonnable, la taille du système de contraintes est cruciale pour le calcul des "minimum conflict sets". Pour réduire l'espace de recherche on pourra utiliser des techniques formelles (e.g., la propagation de la plus faible post condition) ou des heuristiques qu'il faudra définir à l'aide d'expérimentations sur des benchmarks réalistes.

Un environnement de programmation par contraintes permet aussi d'explorer de nouvelles voies, comme la recherche d'un chemin "légèrement" différent de la trace défectueuse, mais qui fournit une réponse correcte, ou les limites supérieures et inférieures pour chaque variable d'entrée d'un chemin défectueux. Enfin, on pourra aussi étudier les connexions entre cette approche et des techniques de diagnostic semi-automatisé pour l'analyse de diagrammes d'entité orientée domaine (FODA) [NaK10].

Le travail proposé nécessite des gouts (et aptitudes) pour la formalisation et pour l'expérimentation.

Références

[BNR03] Thomas Ball, Mayur Naik, and Sriram K. Rajamani. From Symptom to Cause: Localizing Errors in Counterexample Traces. Proc. POPL 2003. ACM Press, pp. 97-105.

[Chin96] John W. Chinneck: An Effective Polynomial-Time Heuristic for the Minimum-Cardinality IIS Set-Covering Problem. Ann. Math. Artif. Intell. 17(1-2): 127-144 (1996).

[Chin01] John W. Chinneck: Fast Heuristics for the Maximum Feasible Subsystem Problem. INFORMS Journal on Computing 13(3): 210-223 (2001)

[CRH10] Hélène Collavizza, Michel Rueher, Pascal Van Hentenryck: CPBPV: a constraint-programming framework for bounded program verification. Constraints 15(2): 238-264 (2010)

[CVR11] Hélène Collavizza, Nguyen Le Vinh, Michel Rueher, Samuel Devulder, Thierry Gueguen. Efficient Constraint-Based Dynamic Strategies For Generating Counterexamples. 26th ACM Symposium On Applied Computing (SAC 2011), Software Verification and Testing Track.

[LiL10] Yongmei Liu, Bing Li: Automated Program Debugging Via Multiple Predicate Switching.

Proc. AAAI 2010, AAAI Press.
[GBC06] Andreas Griesmayer, Roderick Bloem, and Byron Cook. Repair of Boolean Programs with an Application to C. Proc of CAV'06 LNCS 4144, pp. 358-371.
[GCK06] Alex Groce, Sagar Chaki, Daniel Kroening , and Ofer Strichman. Error explanation with distance metrics. International Journal on Software Tools for Technology
(2006) 8(3): 229–247
[GKL04] Alex Groce, Daniel Kroening, and Flavio Lerda. Understanding Counterexamples with explain. Proc. of CAV 2004, LNCS 3114, pp. 453–456, 2004.
[Jun04] Ulrich Junker: QUICKXPLAIN: Preferred Explanations and Relaxations for Over-Constrained Problems. Proc. of AAAI 2004. Pp. 167-172
[GSB07] Andreas Griesmayer, Stefan Staber, and Roderick Bloem. Automated Fault Localization for C Programs. Electronic Notes in Theoretical Computer Science 174 (2007)
95–111.
[MaM11] Manu Jose, Rupak Majumdar. Cause Clue Clauses: Error Localization using Maximum Satisfiability. Proc. of PDLI 11 (32nd ACM SIGPLAN conference on
Programming Language Design and Implementation).
[Nak10] Shin Nakajima . Semi-automated diagnosis of FODA feature diagram. Proceedings of the 2010 ACM Symposium on Applied Computing (SAC '10), pp. 2191-2197.
[ReR03] Manos Renieris, Steven P. Reiss: Fault Localization With Nearest Neighbor Queries. Proc of 18th IEEE International Conference on Automated Software Engineering
(ASE 2003), pp. 30-39.
[SQL05] ShengYu Shen, Ying Qin, Sikun Li: Minimizing Counterexample with Unit Core Extraction and Incremental SAT. Proc of VMCAI 2005, LNCS 3385, pp. 298-312
[ZGG06]Xiangyu Zhang, Neelam Gupta, Rajiv Gupta: Locating faults through automated predicate switching. Proc. ICSE 2006, ACM press, pp. 272-281

URL : http://users.polytech.unice.fr/~rueher/CBEL.pdf

English version:

Constraint-based error localization

Errors localization with counterexamples and execution traces is a critical issue in software debugging. Indeed, when a program P contains errors, a model-checker delivers a
counter example or an execution trace that is often lengthy and difficult to understand. The location of the portions of code that contain errors is therefore often very expensive,
even for experienced programmers.
Different approaches have been proposed to assist the programmer in this task [BNR03, ReR03, GKL04, SQL05, GBC06, ZGG06, GCK06, GSB07, LiL10, MaM11]. Current
approaches are mainly based on SAT solvers, which are well adapted to bounded model-checkers as CBMC. In our constraint-programming environment for bounded program
verification [CRH10, CVR11], we have a richer model that we can try to exploit in the context of this PhD thesis; PhD thesis goal of which is to design and develop a
constraint-based tool providing automated assistance in understanding and localizing errors in C or Java programs.
In a first step, we could generalize the approach introduced in [MaM11]: instead of only

computing the maximum number of constraints that can be satisfied, we can compute different subsets of inconsistent (resp. consistent) constraints. More precisely, from the constraint system derived from the trace of a counter example and the post-condition, we can easily compute:

1. The IIS (irreducible Infeasibility set) [Chin96,Chin01] for the linear constraints
2. Minimum conflict sets [Jun04] for the constraints of the CSP.

Experiments on standard benchmarks and real applications are required to assess the practical relevance of such information. Computing irreducible infeasibility sets (or Maximum Feasible Subsystem) for linear constraints can be done at a reasonable cost. However, the size of the constraint system is a critical issue when computing minimum conflict sets. To reduce the search space we could use formal techniques like weakest post condition propagation. Experiments on realistic benchmarks will also help to define efficient heuristics.

To assist the user we could compute the following information, e.g., searching a "slightly" different path from the faulty path but which provides a correct answer, upper In a second step, we could also investigate the connexions between this approach and semi-automated diagnosis techniques for Feature Oriented Domain Analysis (FODA) diagrams [NaK10].

References

[BNR03] Thomas Ball, Mayur Naik, and Sriram K. Rajamani. From Symptom to Cause: Localizing Errors in Counterexample Traces. Proc. POPL 2003. ACM Press, pp. 97-105.

[Chin96] John W. Chinneck: An Effective Polynomial-Time Heuristic for the Minimum-Cardinality IIS Set-Covering Problem. Ann. Math. Artif. Intell. 17(1-2): 127-144 (1996).

[Chin01] John W. Chinneck: Fast Heuristics for the Maximum Feasible Subsystem Problem. INFORMS Journal on Computing 13(3): 210-223 (2001)

[CRH10] Hélène Collavizza, Michel Rueher, Pascal Van Hentenryck: CPBPV: a constraint-programming framework for bounded program verification. Constraints 15(2): 238-264 (2010)

[CVR11] Hélène Collavizza, Nguyen Le Vinh, Michel Rueher, Samuel Devulder, Thierry Gueguen. Efficient Constraint-Based Dynamic Strategies For Generating Counterexamples. 26th ACM Symposium On Applied Computing (SAC 2011), Software Verification and Testing Track.

[LiL10] Yongmei Liu, Bing Li: Automated Program Debugging Via Multiple Predicate Switching. Proc. AAAI 2010, AAAI Press.

[GBC06] Andreas Griesmayer, Roderick Bloem, and Byron Cook. Repair of Boolean Programs with an Application to C. Proc of CAV'06 LNCS 4144, pp. 358-371.

[GCK06] Alex Groce, Sagar Chaki, Daniel Kroening , and Ofer Strichman. Error explanation with distance metrics. International Journal on Software Tools for Technology (2006) 8(3): 229–247

[GKL04] Alex Groce, Daniel Kroening, and Flavio Lerda. Understanding Counterexamples with explain. Proc. of CAV 2004, LNCS 3114, pp. 453–456, 2004.

[Jun04] Ulrich Junker: QUICKXPLAIN: Preferred Explanations and Relaxations for Over-Constrained Problems. Proc. of AAAI 2004. Pp. 167-172
[GSB07] Andreas Griesmayer, Stefan Staber, and Roderick Bloem. Automated Fault Localization for C Programs. Electronic Notes in Theoretical Computer Science 174 (2007)
95–111.
[MaM11] Manu Jose, Rupak Majumdar. Cause Clue Clauses: Error Localization using Maximum Satisfiability. Proc. of PDLI 11 (32nd ACM SIGPLAN conference on
Programming Language Design and Implementation).
[Nak10] Shin Nakajima . Semi-automated diagnosis of FODA feature diagram. Proceedings of the 2010 ACM Symposium on Applied Computing (SAC '10), pp. 2191-2197.
[ReR03] Manos Renieris, Steven P. Reiss: Fault Localization With Nearest Neighbor Queries. Proc of 18th IEEE International Conference on Automated Software Engineering
(ASE 2003), pp. 30-39.
[SQL05] ShengYu Shen, Ying Qin, Sikun Li: Minimizing Counterexample with Unit Core Extraction and Incremental SAT. Proc of VMCAI 2005, LNCS 3385, pp. 298-312
[ZGG06]Xiangyu Zhang, Neelam Gupta, Rajiv Gupta: Locating faults through automated predicate switching. Proc. ICSE 2006, ACM press, pp. 272-281