Open access • Proceedings Article • DOI:10.1109/ACSD.2013.21

Prototyping a Concurrency Model — Source link

Benjamin Morandi, Mischael Schill, Sebastian Nanz, Bertrand Meyer

Institutions: ETH Zurich

Published on: 08 Jul 2013 - International Conference on Application of Concurrency to System Design

Topics: Non-lock concurrency control, Concurrent object-oriented programming, Multiversion concurrency control, Isolation (database systems) and Optimistic concurrency control

Prototyping a Concurrency Model

Diss. ETH No. 21822

Prototyping a Concurrency Model

A thesis submitted to attain the degree of

DOCTOR OF SCIENCES of ETH ZURICH

(Dr. sc. ETH Zurich)

presented by

BENJAMIN MORANDI

Master of Science ETH in Computer Science
born on July 7th, 1983
citizen of Rossa (GR), Switzerland

accepted on the recommendation of

Prof. Dr. Bertrand Meyer, examiner
Prof. Dr. José Meseguer, co-examiner
Prof. Dr. Jayadev Misra, co-examiner

2014

Abstract

Many novel programming models for concurrency have been proposed in the wake of the multicore computing paradigm shift. These models aim to raise the level of abstraction for expressing concurrency and synchronization in a program, thereby helping programmers avoid programming errors. This goal, however, causes the semantics of the models to become ever more complex and increases the risk of design flaws. Such flaws can have costly consequences if they are discovered after compiler and runtime support has been developed. It is therefore beneficial to verify the models beforehand.

This thesis proposes to prototype concurrency models using executable formal specifications. The prototype is useful from the beginning to the end of a model development. Initially, developers can use the prototype to test and correct the core of the model. As the development continues, developers can expand the prototype iteratively. For each extension, they enhance the specification, test the extension against the rest of the model, and apply corrections where necessary. Once the development is completed, the prototype serves as a reference.

This thesis applies the prototyping method to SCOOP, an object-oriented concurrency model. It demonstrates how the method facilitates the process of finding and resolving flaws in SCOOP and two extensions. In particular, it applies the method to extend SCOOP with (1) an exception mechanism to handle exceptions resulting from asynchronous calls and (2) a mechanism for fast and safe data sharing, reducing execution time by several orders of magnitude on data-intensive parallel programs. This effort results in 16 clarifications across various aspects, all included in a comprehensive executable formal specification in Maude.

This thesis also presents new SCOOP-specific performance metrics and a technique to compute them from event traces. Having a verified concurrency model does not guarantee that programmers write efficient concurrent programs. It is hence necessary to provide performance analysis tools that assist programmers in their task. Since SCOOP differs considerably from established models, reusing existing performance metrics is not an option. Instead, the new metrics are specifically designed for SCOOP. A case study on optimizing a robot control software

demonstrates the usefulness of these metrics.

As a result of this thesis, SCOOP has an executable formal specification for future SCOOP developers, new mechanisms for exception handling and data sharing, as well as SCOOP-specific performance metrics. Having demonstrated the benefits of the method on an extensive concurrency model, we believe that the method can also benefit other models - new or mature.

Zusammenfassung

Im Zuge der zunehmenden Verbreitung von Mehrkernprozessoren wurden viele Programmiermodelle für Nebenläufigkeit vorgeschlagen. Diese Modelle helfen Programmierern Fehler zu vermeiden, indem sie Abstraktionen einführen, um Nebenläufigkeit und Synchronisation auszudrücken. Diese Abstraktionen erhöhen aber auch die Komplexität der Modelle, was das Risiko von Designfehlern erhöht. Solche Fehler können teure Konsequenzen haben, falls sie erst nach der Entwicklung des Compilers und der Laufzeit gefunden werden. Es ist daher vorteilhaft, das Modell vorweg zu verifizieren.

Diese Dissertation schlägt vor, Nebenläufigkeitsmodelle mittels ausführbaren formalen Spezifikationen zu entwickeln. Die Spezifikation eines Modells dient als Prototyp und ist nützlich vom Anfang bis zum Ende der Entwicklung. Zu Beginn können Entwickler den Prototyp verwenden, um den Kern des Modells zu testen und zu korrigieren. Später können Entwickler den Prototyp iterativ weiterentwickeln. Für jede Erweiterung ergänzen sie die Spezifikation, testen die Erweiterung in der Gesamtheit und bringen gegebenenfalls Korrekturen an. Sobald die Entwicklung des Modells abgeschlossen ist, dient der Prototyp als Referenz.

Diese Dissertation wendet die Prototyping Methode auf ein objektorientiertes Nebenläufigkeitsmodell namens SCOOP an. Sie zeigt wie die Methode hilft, Fehler in SCOOP und zwei Erweiterungen zu finden und zu korrigieren. Im Besonderen erweitert sie SCOOP unter Einsatz der Methode mit (1) einem neuen Mechanismus zur Behandlung von Ausnahmen, die aus asynchronen Aufrufen resultieren, und (2) einem Mechanismus zur schnellen und sicheren Datenteilung, welcher die Ausführungszeit von Programmen mit vielen Datenzugriffen um mehrere Grössenordnungen reduziert. Dieser Aufwand mündet in 16 Klarstellungen von verschiedenen Aspekten, die alle in einer umfassenden ausführbaren formalen Spezifikation in Maude enthalten sind.

Diese Dissertation präsentiert zusätzlich SCOOP-spezifische Leistungsmetriken und eine Technik, um diese aus Ereignisprotokollen zu berechnen. Ein verifiziertes Nebenläufigkeitsmodell garantiert nicht, dass Programmierer effiziente nebenläufige Programme schreiben. Es ist daher wichtig den Programmierern Lei-

stungsanalysetools zur Verfügung zu stellen, die sie bei ihrer Arbeit unterstützen. Da sich SCOOP grundlegend von etablierten Modellen unterscheidet, ist es nicht angebracht, existierende Leistungsmetriken zu verwenden. Deshalb sind die neuen Metriken gezielt an SCOOP angepasst. Eine Fallstudie über die Optimierung einer Robotersteuerung demonstriert den Nutzwert dieser Metriken.

Durch diese Dissertation erhält SCOOP eine ausführbare formale Spezifikation für zukünftige SCOOP Entwickler, neue Mechanismen für die Ausnahmebehandlung und Datenteilung, sowie SCOOP-spezifische Leistungsmetriken. Da die Methode bei der Anwendung auf ein umfangreiches Nebenläufigkeitsmodell vorteilhaft ist, sind wir zuversichtlich, dass die Prototyping Methode auch bei der Entwicklung von anderen Modellen, neu oder ausgereift, von Vorteil sein kann.