Heinrich Heine
Universität
Düsseldorf

INSTITUT FÜR INFORMATIK
Softwaretechnik und Programmiersprachen
Universitätsstr. 1 D–40225 Düsseldorf

Entkopplung der Z3 Komponente in ProB mit ZeroMQ

Bachelorarbeit

vorgelegt von

Silas Alexander Kraume

22. Januar 2025

im Studiengang Informatik
zur Erlangung des akademischen Grades

Bachelor of Science (B.Sc.)

Erstgutachter: Prof. Dr. Michael Leuschel
Zweitgutachter: Dr. C. Bolz-Tereick

Selbstständigkeitserklärung

Hiermit versichere ich, die vorliegende Bachelorarbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt zu haben. Alle Stellen, die aus den Quellen entnommen wurden, sind als solche kenntlich gemacht worden. Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

Düsseldorf, den 22. Januar 2025

Silas Alexander Kraume

Zusammenfassung

Fassen Sie hier die Fragestellung, Motivation und Ergebnisse Ihrer Arbeit in wenigen Worten zusammen.

Die Zusammenfassung sollte den Umfang einer Seite nicht überschreiten.

Danksagung

Im Falle, dass Sie Ihrer Arbeit eine Danksagung für Ihre Unterstützer (Familie, Freunde, Betreuer) hinzufügen möchten, können Sie diese hier platzieren.

Dieser Part ist optional und kann im Quelltext auskommentiert werden.

Inhaltsverzeichnis

1 Einführung

1.1 Motivation

blabla

Jedoch birgt der Einsatz des Z3 Solvers auch Herausforderungen, die die Effizienz und Zuverlässigkeit der Anwendung beeinträchtigen können. Ein bekanntes Problem besteht in dem sporadischen Auftreten von Speicherlecks und Segmentation Faults, die sowohl die Stabilität als auch die Nutzbarkeit von ProB's Z3 Interface negativ beeinflussen. Diese technischen Mängel erschweren nicht nur die Durchführung formaler Verifikationen, sondern können auch zu einer zeitraubenden Verwendung der Z3 Solver Komponente sowie Unterbrechung von Arbeitsprozessen führen.

Ein weiterer Mangel liegt in der aktuellen sequentiellen Lösung mehrerer Prädikate. Dieser Ansatz, bei dem die Prädikate nacheinander gelöst werden ist in seiner Natur ressourcenintensiv und zeitaufwändig. Angesichts der steigenden Komplexität formaler Modelle und der wachsenden Nachfrage nach schnellerer Verifikation wird die Limitierung durch die sequentielle Verarbeitung immer offensichtlicher. Eine Parallelisierung der Lösung von Prädikaten könnte hier erhebliche Leistungsverbesserungen bringen, indem moderne Mehrkernarchitekturen effizienter ausgenutzt werden.

Die Kombination dieser Herausforderungen (sporadische technische Instabilitäten und begrenzte Effizienz durch sequentielle Verarbeitung) macht es notwendig, alternative Ansätze oder Verbesserungen für die Integration des Z3 Solvers in ProB zu erforschen. Ziel ist es, sowohl die Zuverlässigkeit als auch die Leistung zu steigern, um den Anforderungen der Nutzer und der immer komplexer werdenden Modelle gerecht zu werden. Diese Problematik bildet die Grundlage und Motivation für die vorliegende Arbeit. Sie zielt darauf ab, die Integration des Z3 Solvers in ProB zu verbessern, indem die bestehende Vorgehensweise verworfen und durch eine neue Architektur ersetzt wird.

1.2 Architekturänderung

bild aus expose

2 Hintergrund

Zur Förderung eines einheitlichen Verständnisses werden in diesem Abschnitt zunächst die erforderlichen Hintergrundinformationen illustriert. Im Folgenden werden die drei zentralen Konzepte behandelt, die für das Verständnis dieser Arbeit von Bedeutung sind: ProB, Z3 und ZeroMQ.

2.1 ProB

[LB03]

2.2 Z3 Solver

[BN24] [MB08]

2.3 ZeroMQ

[Hin13] [S+15]

3 Architekturänderung

3.1 Prolog Datentypen

- atoms string - integers longs (laut dokumentation bis version blabla) - floats doubles - typerefs problem, weil kann alles sein

3.2 Struktur der Nachrichten

- function identifier - status identifier - message

3.3 Interfacefunktionen

porting of all 53 interface function

3.4 Hilfsfunktionen

inbesondere mk_type statemachines

3.4.1 Optimierungen

loops 2 von 4

manchmal $ctx_data -> blabla$ unnötige $ctx - data$

3.5 Server Struktur

long damn switch case threading

3.6 Serveranbindung

server als subprozess starting as needed

3.7 Logging

via sys argv stdout not captureable in sicstus prolog

4 Exceptions

very important. need to be handled properly. need to work with server rep req structure what if multiple errors? need to notify other process

if function that can have an exception (even if handles eaxception) has an exception, return value will be invalid or empty. then the next code segment has exception because of that invalid return value, it is either not cought because not excpected and cant catch, or it is caught and we have 2 exceptions confusingly.

function a calls function b and c with b return. if b has exception and handles it itself, then it will, because sicstus keeps running until return of interface function, return invalid value, and keep running in a. maybe b does not expect an invalid input value and is uncaught, then segfault. or b now also has exception and then we have 2 in sicstus (confusing). additionally now server and prob have to keep req-rep ping-pong system. double exception feed back cannot work, because when ProBreceive exception

4.1 Kontrollfluss

dependency graph of all functions either throw or catch exceptions

5 Build Prozess

makefile build standalone executable etc...

6 Zusätzliche Ergebnisse

6.1 Softlock

endless loop fixed by interrupting on every reset

6.2 Versionsinkompatibilität

makefile hell glibc (OS) incompatible with z3lib.so -> darwin12

7 Leistungsbewertung

everything from data analysis

7.1 Overhead

7.2 IPC vs TCP

8 Zukünftige Arbeit

deinit für dangling zocket und prozess (cleaner)

threading in ProB (eigentlicher Sinn der Arbeit um zu parallelisieren)

9 Konklusion

stuff

Tabellenverzeichnis

Abbildungsverzeichnis

Algorithmenverzeichnis

Quellcodeverzeichnis

Literatur

[BN24] Bjørner, Nikolaj ; Nachmanson, Lev: Arithmetic Solving in Z3. In: Gurfinkel, Arie (Hrsg.) ; Ganesh, Vijay (Hrsg.): *Computer Aided Verification*. Cham : Springer Nature Switzerland, 2024. – ISBN 978–3–031–65627–9, S. 26–41

[Hin13] Hintjens, P: *ZeroMQ: Messaging for Many Applications*. O'Reilly Media, 2013

[LB03] Leuschel, Michael ; Butler, Michael: ProB: A Model Checker for B. In: Araki, Keijiro (Hrsg.) ; Gnesi, Stefania (Hrsg.) ; Mandrioli, Dino (Hrsg.): *FME 2003: Formal Methods*. Berlin, Heidelberg : Springer Berlin Heidelberg, 2003. – ISBN 978–3–540–45236–2, S. 855–874

[MB08] Moura, Leonardo de ; Bjørner, Nikolaj: Z3: An Efficient SMT Solver. In: Ramakrishnan, C. R. (Hrsg.) ; Rehof, Jakob (Hrsg.): *Tools and Algorithms for the Construction and Analysis of Systems*. Berlin, Heidelberg : Springer Berlin Heidelberg, 2008. – ISBN 978–3–540–78800–3, S. 337–340

[S+15] Súster, Martin u. a.: ZeroMQ. In: *Introduction Amy Brown and Greg Wilson* (2015), S. 16