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

1.2 Architekturänderung

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]

ProB: A Versatile Tool for Formal Methods in Software Engineering Introduction In modern software engineering, ensuring the correctness and reliability of systems—particularly in safety-critical domains such as aviation, automotive, and healthcare—is paramount. Formal methods play a crucial role in achieving this goal by providing mathematically rigorous ways to model, verify, and validate software and systems. Among the tools available for formal methods, ProB, developed by Michael Leuschel and his team, stands out as a versatile and robust model checker, animator, and constraint solver.

What is ProB? ProB is a formal verification tool initially designed for the B-Method, a formal method used for specifying, designing, and coding software systems. Over time, ProB has expanded its capabilities to support a range of formal languages, including Event-B, CSP (Communicating Sequential Processes), TLA+, and Z. Its primary functions include model checking, animation of specifications, and solving constraints defined within these models. ProB is widely used in both academic and industrial settings, proving its utility in teaching, research, and real-world system development.

Key Features of ProB

Model Checking ProB enables the automatic verification of formal models against specified properties. For example, it checks whether invariants are maintained, events are deadlock-free, or other user-defined conditions are satisfied. This ensures that the model adheres to its intended behavior, highlighting errors that might otherwise remain undetected.

Animation One of ProB's unique features is its ability to animate formal models. This allows users to explore system behavior interactively, making it easier to understand complex models and identify potential design flaws early in the development process. Animation is

particularly beneficial for stakeholders who may not have expertise in formal methods, as it bridges the gap between abstract specifications and tangible system behavior.

Constraint Solving ProB incorporates a powerful constraint solver, allowing users to find solutions to complex constraints specified in formal models. This is useful in scenarios where proving the existence of a particular state or verifying a certain property is required.

Wide Language Support and Integration Although ProB was initially developed for the B-Method, it now supports a variety of formal languages. Additionally, it integrates seamlessly with tools like Rodin, a popular platform for developing Event-B models. This makes ProB an indispensable part of a broader formal methods ecosystem.

Usability and Accessibility ProB is available as a standalone application, a command-line tool, and even as a web interface. This flexibility ensures that users can leverage its capabilities across various platforms and workflows. Its graphical interface further simplifies the process of visualizing and analyzing models.

Applications of ProB

ProB has been successfully applied in diverse fields, including software engineering, systems engineering, and hardware design. Notable applications include:

Safety-Critical Systems: ProB is used to verify the correctness of systems in domains where failure can have catastrophic consequences, such as railway signaling systems, medical devices, and avionics software. Teaching and Education: The tool serves as a valuable resource for introducing students to formal methods. Its animation capabilities make abstract concepts more accessible to learners. Industry Use: Companies employ ProB to ensure compliance with industry standards and regulations, particularly in scenarios where formal methods are mandated. Advantages and Limitations

The advantages of ProB include its ease of use, comprehensive feature set, and support for multiple formal languages. Its ability to integrate with other tools, such as Rodin, and its powerful visualization and constraint-solving capabilities make it a preferred choice for many practitioners and researchers.

However, like any tool, ProB has limitations. Model checking can suffer from state-space explosion, especially in large systems with complex interactions. While ProB includes optimizations to mitigate this, users must still design their models thoughtfully. Additionally, expertise in formal methods is often required to fully leverage ProB's capabilities, which can pose a barrier for newcomers.

Conclusion

ProB exemplifies the power and potential of formal methods in building reliable and robust systems. Its ability to animate, verify, and validate formal models makes it a cornerstone in the toolkit of engineers and researchers alike. As software and systems become increasingly complex, tools like ProB will continue to play a vital role in ensuring that technology meets

the highest standards of safety and correctness.

2.2 Z3 Solver

[BN24] [MB08]

Z3 Solver: Revolutionizing Automated Reasoning The Z3 Solver, developed by Microsoft Research, stands as one of the most influential tools in the realm of automated reasoning and formal verification. As a powerful Satisfiability Modulo Theories (SMT) solver, Z3 enables users to address complex logical problems by efficiently determining the satisfiability of logical formulas under various theories. Its versatility and performance have made it indispensable in domains ranging from software verification and program synthesis to artificial intelligence and operations research.

At its core, Z3 addresses problems expressed in propositional logic and extends this capability to incorporate richer theories such as linear and nonlinear arithmetic, arrays, bit-vectors, uninterpreted functions, and more. This flexibility allows Z3 to model and reason about a wide variety of systems, ranging from hardware circuits to intricate software applications. For instance, developers use Z3 to detect bugs in programs, prove correctness properties, or automatically generate code snippets that satisfy specified constraints.

One of the key strengths of Z3 lies in its robust architecture and optimization techniques. It employs cutting-edge algorithms, such as conflict-driven clause learning and theory-specific decision procedures, to solve problems that might otherwise be computationally intractable. Its support for quantifiers further expands its applicability to problems involving universally or existentially quantified variables, a common requirement in formal verification tasks.

Z3's open-source nature has significantly contributed to its adoption and evolution. Available on platforms like GitHub, it offers bindings for popular programming languages such as Python, C++, and .NET, making it accessible to a broad range of users. Researchers and engineers alike integrate Z3 into their tools and workflows to automate reasoning tasks, analyze constraints, and verify properties of systems efficiently. Its extensive documentation, active community, and integration with development environments have further cemented its role as a go-to solver.

The impact of Z3 extends beyond academia into industry, where it is employed to verify critical systems in finance, healthcare, and autonomous systems. Its ability to reason about complex models has also made it invaluable in artificial intelligence, where constraint-solving is central to tasks such as planning and scheduling.

In conclusion, the Z3 Solver exemplifies the potential of automated reasoning tools to transform the way we approach logical and computational challenges. By providing a scalable, flexible, and user-friendly platform, Z3 has empowered researchers and practitioners to tackle problems of unprecedented complexity, pushing the boundaries of what is possible in computation and formal reasoning. Its ongoing development and widespread adoption

ensure that Z3 will remain a cornerstone of automated reasoning for years to come.

2.3 ZeroMQ

[Hin13] [S+15]

The ZeroMQ Library: A Powerful Tool for Scalable Messaging ZeroMQ, often referred to as "ØMQ,ïs a versatile and high-performance messaging library that serves as a foundation for building scalable and distributed systems. Designed to abstract the complexities of socket programming, ZeroMQ allows developers to focus on designing robust communication architectures rather than worrying about low-level networking details. Its combination of flexibility, speed, and simplicity has made it a favorite among developers building real-time systems, distributed applications, and microservices.

At its core, ZeroMQ operates as an asynchronous messaging library. Unlike traditional socket programming, which requires intricate handling of connections, message framing, and error conditions, ZeroMQ simplifies these tasks by providing a high-level API. Developers can choose from a variety of communication patterns tailored to specific use cases, such as publish/subscribe, request/reply, and push/pull. For instance, a chat application might use the publish/subscribe pattern to broadcast messages to multiple subscribers, while a task distribution system might leverage the push/pull pattern to distribute workloads across a pool of worker nodes.

One of ZeroMQ's standout features is its performance. Built with efficiency in mind, it minimizes overhead and latency, making it ideal for applications requiring rapid data exchange. This performance advantage is achieved through features like non-blocking I/O, zero-copy message transfers, and optimized protocols. Furthermore, ZeroMQ supports multiple transport mechanisms, including TCP, IPC (inter-process communication), in-process communication, and multicast, allowing it to adapt seamlessly to different networking environments.

ZeroMQ is also highly portable, with bindings available for numerous programming languages, including C, C++, Python, Java, and Go. This wide language support enables interoperability across different components of a system. For example, a Python-based data ingestion service can easily communicate with a Java-based analytics engine using ZeroMQ as the common messaging layer. Additionally, ZeroMQ's compatibility with multiple platforms ensures that developers can deploy it on everything from small embedded systems to large cloud-based infrastructures.

Another strength of ZeroMQ lies in its ability to handle dynamic scaling. Unlike traditional message brokers that act as centralized servers, ZeroMQ operates in a peer-to-peer fashion, enabling decentralized architectures. This eliminates single points of failure and allows the system to grow organically as new nodes join. Such scalability is particularly beneficial for applications in fields like IoT, financial trading, and distributed machine learning, where

workloads and data volumes can fluctuate unpredictably.

Despite its many advantages, ZeroMQ is not without challenges. It is a lightweight library, not a full-fledged message broker, which means it lacks certain enterprise features such as message persistence, monitoring, and security layers out of the box. Developers often need to implement or integrate these features themselves, which may add complexity to some projects. However, this trade-off is often acceptable given ZeroMQ's unmatched speed and flexibility.

In conclusion, ZeroMQ is a powerful and adaptable messaging library that excels in scenarios demanding high performance and scalability. Its simplicity, combined with support for diverse communication patterns and transport mechanisms, empowers developers to build robust distributed systems efficiently. While it may require additional effort to address certain advanced requirements, the benefits it offers make ZeroMQ a cornerstone of modern messaging and communication infrastructures.

deinit für dangling zocket und prozess (cleaner)

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

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