technische universität dortmund

Fakultät für Informatik
Lehrstuhl XIV – Software Engineering
Prof. Dr. Falk Howar
Bachelorarbeit

Bachelorarbeit

Effiziente Repräsentation und Verarbeitung von Constraintmengen

Ansprechpartner: Simon Dierl (simon.dierl@cs.tu-dortmund.de)

Motivation

In der formalen Verifikation von Software ist es oft nötig, Constraints über Mengen von Variablen – z. B. $(x_1 = x_2) \wedge (x_2 \neq x_3)$ – zu verfolgen und auf Erfüllbarkeit zu prüfen. Dafür ist es essenziell, die Constraints speichereffizient darzustellen und Operationen effizient umzusetzen. Insbesondere für die Analyse von Registerautomaten [6] ist es nötig, Projektion, Erweiterung und Erfüllbarkeitstests effizient durchzuführen. In [5] wurde dazu ein möglicher Ansatz auf Basis von Matrixdarstellungen für die Operatoren $=$ und $\neq$ in der Bibliothek `koral`[1] umgesetzt.

Aufgabe

Im Rahmen der Arbeit sollen Datenstrukturen zur Repräsentation von Constraints und passende Algorithmen zur Projektion auf Teilmengen, Erweiterung und Erfüllbarkeitstests umgesetzt und mittels Benchmarks verglichen werden. Die Implementierung sollte Zusätzlich können Erweiterungen wie Veroderung oder geordnete Körper wie $\mathbb{Q}$ mit Operatoren wie $\leq$ betrachtet werden.

Ziele

Die geplante Arbeit umfasst folgende Schritte und Ziele:

1. Recherche möglicher Ansätze zur Constraintrepräsentation
2. Auswahl der zu verfolgenden Ansätze
3. Umsetzung in einer JVM-Sprache
4. Implementierung von geeigneten Benchmarks
5. Vergleich der Ansätze auf Basis von Benchmarks und theoretischer Betrachtungen

Mögliche Ansätze könnten die Nutzung von SMT-Solvern wie Z3 [4] und cvc5 [2] oder die Nutzung von Union-Find-Deletes [1, 3] sein.

Voraussetzungen

Für die Bachelorarbeit sind gute Programmierkenntnisse sowie Kenntnisse im Bereich der Algorithmen und Datenstrukturen von Nöten.

[1] `https://github.com/tudo-aqua/koral/`

Literatur

[1] Stephen Alstrup, Mikkel Thorup, Inge Li Gørtz, Theis Rauhe, and Uri Zwick. Union-find with constant time deletions. ACM Transactions on Algorithms, 11(1):6:1–6:28, August 2014. doi:10.1145/2636922.

[2] Haniel Barbosa, Clark Barrett, Martin Brain, Gereon Kremer, Hanna Lachnitt, Makai Mann, Abdalrhman Mohamed, Mudathir Mohamed, Aina Niemetz, Andres Nötzli, Alex Ozdemir, Mathias Preiner, Andrew Reynolds, Ying Sheng, Cesare Tinelli, and Yoni Zohar. cvc5: A versatile and industrial-strength SMT solver. In Dana Fisman and Grigore Rosu, editors, Tools and Algorithms for the Construction and Analysis of Systems, volume 13243 of Lecture Notes in Computer Science, pages 415–442, Cham, 2022. Springer International Publishing. TACAS 2022. doi:10.1007/978-3-030-99524-9_24.

[3] Amir Ben-Amram and Simon Yoffe. A simple and efficient union–find–delete algorithm. Theoretical Computer Science, 412(4):487–492, February 2011. doi:10.1016/j.tcs.2010.11.005.

[4] Leonardo de Moura and Nikolaj Bjørner. Z3: An efficient SMT solver. In C. R. Ramakrishnan and Jakob Rehof, editors, Tools and Algorithms for the Construction and Analysis of Systems, volume 4963 of Lecture Notes in Computer Science, pages 337–340, Berlin, Heidelberg, 2008. Springer. TACAS 2008. doi:10.1007/978-3-540-78800-3_24.

[5] Simon Dierl and Falk Howar. REACH on register automata via history independence. In Laura Kovács and Karl Meinke, editors, Tests and Proofs, volume 13361 of Lecture Notes in Computer Science, pages 11–30, Cham, 2022. Springer International Publishing. TAP 2022. doi:10.1007/978-3-031-09827-7_2.

[6] Michael Kaminski and Nissim Francez. Finite-memory automata. Theoretical Computer Science, 134(2):329–363, November 1994. doi:10.1016/0304-3975(94)90242-9.