HTWK
Hochschule für Technik,
Wirtschaft und Kultur Leipzig

IM Fakultät
Informatik und Medien

Abschlussarbeit
zur Erlangung des akademischen Grades

Bachelor of Science (B.Sc.)

im Studiengang Informatik
der Fakultät Informatik und Medien

Erfüllbarkeit und Optimierung von Pressenplanungsproblemen mit Constraint-Systemen

Julian Bruder
Leipzig, den 21. August 2024

Erstgutachter: Prof. Dr. Johannes Waldmann
Zweitgutachter: Dr. Sebastian Möbius

Die Pressenplanung von Holzbalken aus Bauteilen erfordert eine fein abgestimmte Konfiguration, um die Pressen optimal auszulasten und den Verschnitt der Balken zu minimieren. Dafür hat die Germanedge Solutions GmbH[1], Industriepartner dieser Arbeit, bereits eine unternehmensinterne Heuristik, welche aufgrund ihrer Natur allerdings möglicherweise (optimale) Lösungen nicht berücksichtigt. Im Rahmen dieser Arbeit wird daher eine prototypische Lösung mithilfe der Methoden der Constraint-Programmierung, speziell jener der Satisfiability Modulo Theories (SMT), erarbeitet, welche optimale Lösungen finden soll. Dazu werden die Constraints der Domäne zunächst mithilfe der Prädikatenlogik (FOL) modelliert und anschließend in einer konkreten Repräsentation des Modells implementiert.

Dabei wird ein Programm entwickelt, welches aus den textuellen Eingaben der Pressen- und Bauteilspezifikationen ein Constraint-System erzeugt, das anschließend von einem externen Constraint-Solver gelöst wird. Das Programm wandelt die Lösung des Solvers folglich in eine, zu den bereits existenten Schnittstellen passende, textuelle Ausgabe um. Weiter ist das Programm so strukturiert, dass verschiedene Logiken aus SMT zur Erzeugung des Constraint-Systems genutzt werden können. Dies wird durch Wahl der Programmiersprache Haskell und deren ausdrucksstarkes Typsystem ermöglicht. Als Grundlage zur Kodierung des Pressenplanungsproblems und Interaktion mit externen Solvern wird die Haskell-Bibliothek `Hasmtlib` [Bru24] verwendet und an geeigneten Stellen erweitert.

Das Ziel für die prototypische Lösung des Problems der Pressenplanung ist die korrekte, optimale und bestenfalls schnelle (unter sechs Minuten) Lösung bekannter Testfälle. Da es sich hier um eine prototypische Lösung handelt, werden einige Sonderfälle des Pressenplanungsproblems in dieser Arbeit nicht betrachtet. Am Ende wird die Performance der prototypischen Lösung mit jener der aktuell existenten heuristischen verglichen.

[1] https://www.germanedge.com/, abgerufen am 21.08.2024

Inhaltsverzeichnis

1. Einleitung

Germanedge - Unternehmen (wenige Sätze). Problem? SMT? Wo normalerweise Anwendungen SMT? Wie Anwendung hier? Beispiel mit Visualisierung und Kodierung. Alle Kapitel kurz erwähnen und in einem Satz erklären.

2. Grundlagen der Prädikatenlogik

Erfüllbarkeit, Terme, Atome, Formeln ... [KB08]

3. Problemdomäne Pressenplanung

Einleitung...

3.1. Anforderungen

Kurze Beschreibung der Problemdomäne. Was wird gepresst, wie, welche Schritte, ..., welche Ziele?

3.2. Modellierung

Mathematische Modelle für Erfüllbarkeit des Problems. Warum zwei Modelle? Größe des Problems (Unbekannte, Formeln)?

3.3. Optimierung

Erweiterung der Modelle um Optimierungsziele. OMT, Incremental refinement, ...

4. Satisfiability Modulo Theories

Einleitung...

4.1. Einführung in das Erfüllbarkeitsproblem SMT

Welche Logiken hier relevant? Was zeichnet diese Logiken aus? Wie performant lösen die Solver diese Logiken - Komplexität? SMT-Comp - beste Solver für relevante Kategorien nennen - Lösungsverfahren nur minimal erwähnen.

4.2. Sprachstandard SMTLib2.6

Kodierung prädikatenlogischer Problemen. Assert, ...

4.3. SMT-Kodierung von Anzahl-Constraints

4.4. Optimierung - Optimization Modulo Theories

5. Implementierung in Haskell

Erwähnen, dass Germanedge bereits Lösung hat. Aber: Heuristik - (optimale) Lösungen gehen möglicherweise verloren. Hier: Neuer Ansatz mit Garantie Vollständigkeit und Optimalität. Was ist Haskell und warum Haskell? Was ist Hasmtlib?

5.1. Funktionale Programmierung mit Haskell

5.2. Hasmtlib - Haskell-Bibliothek für SMTLib

Generelle Funktionsweise: Gastsprache - Constraintsprache - Interaktion Solver. Hier müssen besonders der polymorphe Formelbaum (Expr-GADT) und die dependent-like Singletons (SSMTSort) erklärt werden. Wird später gebraucht um zu verstehen, wie Pressenplanungsprobleme flexibel in verschiedenen Logiken kodiert werden (Programm polymorph in SMTSort). Codec erklären.

5.2.1. Ergänzung um Anzahl-Constraints

5.2.2. Ergänzung um Incremental Refinement

5.2.3. Ergänzung um Observable-Sharing

Sprengt das den Rahmen? Hier müsste vorher viel erklärt werden: Repräsentation von Bäumen im Speicher in Haskell (DAG), Auszug Kategorientheorie - Morphismen (hier Paramorphismus), Vergleich zu Tseitin sinnvoll, aber bisher SAT gänzlich unerwähnt. Dann auch die Frage: Warum der Aufwand? Bringt das überhaupt was für den Anwendungsfall Pressenplanung? Wenn nein: Gehört das hier dann überhaupt rein?

5.3. Kodierung des Pressenplanungsproblems

Zwei Modelle erschweren hier möglicherweise Verständnis für den Leser. Für ein Modell entscheiden: Relationales Modell begründend wegwerfen, auf Laufzeitmessung verweisen. Ebenso Unterscheidung Incremental Refinement/OMT, OMT wegwerfen, weil Laufzeit. Geeignete Programmauszüge: Polymorphe Kodierung in Sorte (damit Logik), vereinzelt Datentypen und Constraints analog zu Modellierung. Minimal IO erklären.

5.4. Validierung der Lösung

Warum validieren? Wie validieren? Wenn Modell richtig und auch richtig in SMT kodiert, dann Ergebnis richtig gdw. Solver Spezifikation erfüllt. Erfüllt Solver die Spezifikation? Ja hoffentlich - das müssen wir denen abkaufen - oder Nachweis bspw. per SMT-Comp?

6. Auswertung und Laufzeitmessung

Was haben wir für Testfälle? Woher kommen die Testfälle? Wie wurden die Testfälle ausgewählt? Sind sie repräsentativ für reale Probleme? Vorgehen Laufzeitmessung erwähnen. Maschinen-Specs nennen.

6.1. Vergleich mit realistischem Timeout

Time-Limit 5 Minuten, Vergleich Heuristik Germanedge vs SMT. Nur ein Modell oder beide? - Beide. Auch OMT? - Ja. Auswertung der Ergebnisse.

6.2. Vergleich mit liberalem Timeout

Wer findet bessere Lösung, wenn Timeout sehr liberal (24h?)? Hier werden Hardware-Specs ein Problem. Das muss auf Pool-PCs der HTWK gemacht werden. Darf da ein Binary von der Heuristik laufen? Auswertung der Ergebnisse.

7. Zusammenfassung

Literatur

[Bru24] Julian Bruder. *hasmtlib: A monad for interfacing with external SMT solvers*. Version 2.3.2. Abgerufen am: 20.08.2024. 2024. URL: https://hackage.haskell.org/package/hasmtlib.

[KB08] Uwe Kastens und Hans Kleine Büning. *Modellierung*. 2., überarbeitete und erweiterte Auflage. München: Carl Hanser Verlag GmbH & Co. KG, 2008. DOI: 10.3139/9783446417212. eprint: https://www.hanser-elibrary.com/doi/pdf/10.3139/9783446417212. URL: https://www.hanser-elibrary.com/doi/abs/10.3139/9783446417212.

Appendix

A. Beispiel 1

⋮

B. Beispiel n

Danksagung

Zuallererst möchte ich mich bei Herrn Prof. Dr. Waldmann für das im Wintersemester 2023/2024, in der Vorlesung Constraint-Programmierung geweckte Interesse an jener bedanken. Vielen Dank auch für Ihre zahlreichen Hinweise und Vorschläge während dieser Bachelorarbeit. Weiter möchte ich meinen Dank der Germanedge Solutions GmbH und dort besonders Christoph Schumacher aussprechen, die diese Bachelorarbeit ermöglicht haben. Abschließend auch ein großes Dankeschön an Dr. Sebastian Möbius, welcher mir als Firmenbetreuer jederzeit alle Fragen bestens beantwortet hat.

Eidesstattliche Erklärung

Hiermit versichere ich, dass ich die vorliegende Arbeit selbstständig ohne unzulässige Hilfe Dritter und ohne Benutzung anderer als der angegebenen Quellen und Hilfsmittel verfasst habe. Die den verwendeten Quellen und Hilfsmitteln wörtlich oder sinngemäß entnommenen Stellen sind als solche kenntlich gemacht. Die Arbeit wurde bisher keiner anderen Prüfungsbehörde vorgelegt und auch noch nicht veröffentlicht. Mir sind die strafrechtlichen Konsequenzen einer falschen eidesstattlichen Erklärung bekannt.

Leipzig, den 21. August 2024