Das SAT-Problem

Sebastian GANDLER
Sebastian.Gandler@student.uibk.ac.at
Universität Innsbruck

1 Einleitung

In diesem Dokument geht es um das sogenannte SAT-Problem. Das erste Kapitel gibt eine kurze Einführung zum Thema Aussagenlogik und Erfüllbarkeitsbeweis. Im zweiten Kapitel wird über die Komplexität von Algorithmen gesprochen. Im Anschluss werden wichtige Algorithmen zum Lösen von SAT-Problemen erläutert. Zu guter Letzt werden in den letzten beiden Kapitel über das eigentliche SAT-Problem und zudem praxisnahe Anwendungen angeführt.

2 Begriffserklärung

Um Verständnisprobleme im Vorhinein zu vermeiden, werden in diesem Kapitel die Grundlagen der Aussagenlogik und deren Erfüllbarkeitsbeweis näher erläutert.

2.1 Aussagenlogik

$F = (x1 \wedge (\neg(x2 \vee (x3 \vee x4))))$ Aus dieser Formel ist gut ersichtlich, dass sogenannte logische Operatoren UND ($\wedge$), ODER ($\vee$) sowie der Negation ($\neg$) zur Beschreibung verwendet werden um Aussagen in Boolesche Formeln zu verpacken.

2.1.1 Beispiel

Sei die Formel $F = x1 \vee (x2 \wedge \neg x3)$ sowie die Belegung A mit $A(x1) = 0$, $A(x2) = 1$ und $A(x3) = 0$ gegeben. Der Wahrheitswert von F lässt sich wie

folgt bestimmen:

$$\begin{aligned} A(F) &= A(A(x1) \vee A(A(x2) \wedge A(\neg x3))) && (1)\\ &= A(0 \vee A(1 \wedge 1)) && (2)\\ &= A(0 \vee 1) && (3)\\ &= 1 && (4) \end{aligned}$$

Für eine ausführlichere Beschreibung dieses Beispiels verweisen wir auf [4]

Definition 2.1. ***(Erfüllbarkeit)***
Eine Formel F der Aussagenlogik ist genau dann erfüllbar, wenn eine Belegung A mit A(F) = 1 existiert, sonst ist F unerfüllbar.

Definition 2.2. ***(Konjunktive Normalform)***
Eine Formel F der Aussagenlogik ist genau dann in konjunktiver Normalform, wenn sie aus der Konjunktion von Klauseln besteht.[3]

3 Komplexität von Algorithmen

Bevor wir uns noch weiter in die Materie begeben, sollten wir uns zuerst der Komplexität von Algorithmen widmen. Unter dem Begriff Komplexität versteht man den Ressourcenverbrauch eines Algorithmus zur Lösung eines entscheidbaren Problems. Um die Schwierigkeit solcher Probleme und deren Mächtigkeit zu unterteilen, verwendet man sogenannte Komplexitätsklassen.

4 Algorithmen

In diesem Kapitel zeigen wir wichtige Algorithmen, welche das Grundgerüst vieler moderner SAT-Solver bilden.

4.1 Davis-Putnam (DP) Algorithmus

Im Jahre 1960 wurde von Davis und Putnam ein wichtiger Schritt zum Lösen von Erfüllbarkeitsproblemen der Aussagenlogik vorgestellt. Der DP-Algorithmus löst das Erfüllbarkeitsproblem indem es in jedem Durchlauf durch Fallunterscheidung eine Variable eliminiert. [5] Bevor das geschieht, wird die gegebene Klauselmenge bereinigt, tautologische Klauseln werden gestrichen.

4.2 Davis-Putnam-Logemann-Loveland (DPLL) Algorithmus

Der DPLL Algorithmus ist einer der bekanntesten und weit verbreitetsten Methoden zum Lösen von SAT. Dieser Algorithmus setzt im Gegensatz

zu Davis-Putnam auf das sogenannte Backtracking-Prinzip. Hierbei wird schrittweise eine erfüllende Variablenbelegung konstruiert und die gegebene Formel dabei vereinfacht. [5] Zur Vereinfachung der Formel wird ein Fallunterscheidungslemma genutzt.

5 SAT-Problem

Definition 5.1. ***(SAT-Problem)***
Sei eine Formel F der Aussagenlogik in konjunktiver Normalform gegeben. Die zu beantwortende Fragestellung lautet: ist F erfüllbar, das heisst, existiert eine Belegung A für die in F enthaltenen Variablen, so dass A(F) = 1 gilt?

In dieser Definition ist die Beschränkung auf konjunktiver Normalform nicht zwingend notwendig, es ist dadurch begründet, dass alle gängigen SAT-Verfahren auf Basis in Abschnitt 4.2 vorgestellten Davis-Logemann-Loveland Algorithmus aufbauen.

Der kanadische Wissenschaftler Stephen Arthur Cook entwarf im Jahre 1971 einen Algorithmus, mit dessen Hilfe das Berechnungsverfahren nichtdeterminitscher Turingmaschinen mittels Aussagenlogik simuliert werden kann. Zudem bewies Cook, dass das SAT-Problem zur Klasse der NP-vollständigen Probleme gehört. [2] Die bislang ungelöste Frage ist jedoch, ob jedes NP-vollständige Problem mit einer deterministischen Turingmaschine in polynomieller Zeit gelöst werden kann.

5.1 P-NP Problem

Derzeit gibt es noch keinen Beweis, aber die Vermutung liegt nahe, dass P ungleich NP ist. Könnte man zeigen, dass ein Problem aus NP in Polynomialzeit auf einer deterministischen Turingmaschine lösbar ist, dann würden die Klassen P und NP zusammenfallen, auf Grund der NP-Vollständigkeit. Sollte dies passieren, könnte jedes beliebige Problem aus NP durch Polynomialzeitreduktion darauf reduziert und somit in deterministischer Polynomialzeit gelöst werden. [2]

Satz von Stephen A. Cook

> "Das Erfüllbarkeitsproblem der Aussagenlogik (SAT) ist NP-vollständig."

6 Anwendungsgebiete von SAT-Algorithmen

Aufgrund des enormen Leistungssprungs moderner SAT-Algorithmen in den letzten Jahren konnten sie vermehrt an praktischer Relevanz gewinnen. Das

Anwendungsgebiet der SAT-Algorithmen ist weitreichend, weshalb wir in diesen Kapitel nur ein bekanntes Verfahren beschreiben.

6.1 Bounded Model Checking

Seit der Veröffentlichung 1999, gewann das BMC Verfahren in der Industrie laufend an Bedeutung. BMC ist derzeit konkurrenzlos, wenn es um das Auffinden von logischen Fehlern in komplexen Systemen geht,weshalb es als weit bessere Ergänzung zu Binary Decision Diagrams(BDD) [1] angesehen werden kann. Das Besondere an diesem Verfahren ist, dass bei einem erfolgreichen Fall auch ein Gegenbeispiel aufzeigt wird. Anwendung findet man unter anderem in der künstlichen Intelligenz(Planning as Satisfiability) oder in den Teilgebieten des Schaltkreisentwurfs.

7 Schlussfolgerung

Zusammenfassend lässt sich sagen, dass es zukünftig wichtig ist, an neuen Lösungsmöglichkeiten zu arbeiten und zu forschen, da viele Problemstellungen noch nicht in polynomieller Zeit lösbar sind. In diesem Dokument wurde versucht einen kurzen Überblick über das SAT-Problem selbst sowie über gängige praktische Anwendungen einzugehen. Für eine genauere Behandlung dieses Themas, reicht das Ausmaß dieser Arbeit leider nicht. Genauere Informationen zur Thematik sind unter [2] [3] nachzulesen.

Literatur

[1] Edmund Clarke, Armin Biere, Richard Raimi, and Yunshan Zhu. Bounded model checking using satisfiability solving. *Formal methods in system design*, 19(1):7–34, 2001.

[2] Stephen A. Cook. The complexity of theorem-proving procedures. In *Proceedings of the Third Annual ACM Symposium on Theory of Computing*, STOC '71, pages 151–158, New York, NY, USA, 1971. ACM.

[3] Michael Huth and Mark Ryan. *Logic in Computer Science: Modelling and reasoning about systems.* Cambridge University Press, 2004.

[4] Tobias Schubert. *SAT-Algorithmen und Systemaspekte: vom Mikroprozessor zum parallelen System.* PhD thesis, University of Freiburg, 2008.

[5] Philipp Sieweck. Sat solving mit gpu unterstützung. 2012.