1 Auswertung der Implementierung

In diesem Kapitel wird die Richtigkeit des SAT-Solvers mit dem PicoSAT-Solver verglichen. Nach dem Vergleich wird eine Auswertungen der Zeiten für das Lösen von SAT-Problemen erstellt, wobei die Implementierung mit zChaff und MiniSAT verglichen wird.

1.1 Vergleich mit PicoSAT

Um auf die Richtigkeit der Implementierung zu Prüfen, wird der SAT-Solver gegen den PicoSAT-Solver geprüft. Der PicoSAT-Solver wurde hierfür ausgewählt, weil dieser eine einfache Anbindung zu Haskell besitzt. Zur Generieren der Tests wird das Haskellpaket Test.QuickCheck und die Unterkomponente Test.QuickCheck.Monadic verwendet.

Listing 1.1: Code zum Vergleich der Richtigkeit der Implementierung

```
1  prop_picoSATcomparison :: [[NonZero Int]] -> Property
2  prop_picoSATcomparison cl = withMaxSuccess 10000 $ monadicIO $ do
3    let clauses = coerce cl
4    picoSol <- run (PicoSAT.solve clauses)
5    let cdclSol = cdcl (map (map fromIntegral) clauses) False False
6    assert ( case (picoSol, cdclSol) of
7               (PicoSAT.Unsatisfiable, UNSAT) -> True
8               (PicoSAT.Unknown, _)           -> False
9               (PicoSAT.Solution _, SAT _)    -> True
10              _                              -> False )
```

Die Funktion erstellt zufällig 10000 verschiedene Probleme, wobei die SAT-Solver die generierten Probleme auf ihre Ergebnisse vergleichsweise prüfen. Dabei werden die SAT-Instanzen auf vier verschiedene Fälle verglichen. Wenn beide Solver entsprechende Ergebnisse mit UNSAT für ein Problem melden, so ist der Test für diese eine Instanz erfolgreich. Dies ist auch der Fall, wenn PicoSAT Solution [Integer] und die Implementierung SAT TupleList als Ergebnis ausrechnen. Falls PicoSAT ein Unknown Ergebnis kalkuliert oder keine der beiden ersten genannten Fälle in einem SAT-Problem ausgerechnet werden, so wird der Test als fehlgeschlagen ausgewertet. Wenn ein einziger Test dabei fehlschlägt, so wird der ganze Test abgebrochen und meldet eine fehlerhafte Implementierung für den entsprechenden Fall.

Mit der Eingabe des Befehls „stack test“ in der Kommando-Zeile innerhalb des Implementierungsordners wird der Test gestartet. Die derzeitige Version der Implementierung kann die generierten Test ohne Probleme ausführen, womit die Richtigkeit des Solvers gewährleistet ist.

1.2 Performancevergleich mit anderen SAT-Solver

Der Performancevergleich der Solver wird innerhalb einer virtuellen Maschine (VM) durchgeführt. Die Spezifikation für die VM hierbei sind folgende:

- Betriebsystem: Ubuntu (64-bit)
- Speicherort: Hard Disk Drive (HDD)
- Hauptspeicher 2048 MB

Die VM läuft auf einem Laptop mit den folgenden Spezifikationen:

- Prozessor: Intel(R) Core(TM) i5-8250U CPU @1.60GHz 1.80GHz
- RAM: 8,00 GB
- Systemtyp: 64-Bit-Betriebssystem, x64-basierter Prozessor
- Betriebssystem: Windows 10 Home
- Version: 20H2

In dem Vergleich werden Benchmarks[1] von „Construction and Analysis of Distributed Processes" (CADP) verwendet.
Die Ausführung der Solver wird gestoppt, wenn nach 5 Minuten[2] keine Ergebnisse für die entsprechende Probleminstanz vorliegen. Zum Vergleich werden zChaff 2007 .3.12 (64-bit)[3] und MiniSAT 2.2.1-5build2_amd64[4] herangezogen. Die Implementierung wird in dieser Arbeit als „Impl“ abgekürzt. In dem Vergleich werden die benötigte Zeit zum Lösen der Probleme, die Anzahl der Neustarts, Entscheidungen und der gelernten Klauseln geprüft. Für die Messung der Laufzeit wird in Impl das Haskell-Packet timeit 2.0 verwendet und alle Zeiten werden 2 Stellen nach dem Komma gerundet.
[5]

[1] Link zur Benchmark: https://cadp.inria.fr/resources/vlsat/
[2] (höhere Zeit vielleicht?)
[3] http://www.princeton.edu/~chaff/zchaff.html
[4] Installation mithilfe von „apt-get install minisat“
[5] Sollen weniger Benchmarks verwendet werden? Bin mir nicht sicher ob 8 zu viel sind.

Data	Impl	zChaff	MiniSAT
Zeit in Sekunden	0.00	0.00	0.00
Anzahl der Neustarts	0	-	0
Entscheidungen	0	1	1
Anzahl der gelernten Klauseln	0	0	0

Tabelle 1.1: Benchmark vlsat_1_10_17

Data	Impl	zChaff	MiniSAT
Zeit in Sekunden	0.01	0.00	0.00
Anzahl der Neustarts	0	-	0
Entscheidungen	1	3	
Anzahl der gelernten Klauseln	0	0	

Tabelle 1.2: Benchmark vlsat_1_54_270

Data	Impl	zChaff	MiniSAT
Zeit in Sekunden			
Anzahl der Neustarts		-	
Entscheidungen			
Anzahl der gelernten Klauseln			

Tabelle 1.3: Benchmark vlsat_1_90_316

Data	Impl	zChaff	MiniSAT
Zeit in Sekunden			
Anzahl der Neustarts		-	
Entscheidungen			
Anzahl der gelernten Klauseln			

Tabelle 1.4: Benchmark vlsat_1_102_481

Data	Impl	zChaff	MiniSAT
Zeit in Sekunden			
Anzahl der Neustarts		-	
Entscheidungen			
Anzahl der gelernten Klauseln			

Tabelle 1.5: Benchmark vlsat_1_210_1275

Data	Impl	zChaff	MiniSAT
Zeit in Sekunden			
Anzahl der Neustarts		-	
Entscheidungen			
Anzahl der gelernten Klauseln			

Tabelle 1.6: Benchmark vlsat_1_222_1477

Data	Impl	zChaff	MiniSAT
Zeit in Sekunden			
Anzahl der Neustarts		-	
Entscheidungen			
Anzahl der gelernten Klauseln			

Tabelle 1.7: Benchmark vlsat_1_228_3437

Data	Impl	zChaff	MiniSAT
Zeit in Sekunden			
Anzahl der Neustarts		-	
Entscheidungen			
Anzahl der gelernten Klauseln			

Tabelle 1.8: Benchmark vlsat_1_240_1624

Hier fängt dann Vergleich an. Wie kann die Implementierung verbessert werden in hinsicht der Performance etc....