NOMBRE: Benjamín Farías Valdés

N.ALUMNO: 17642531

Pontificia Universidad Católica de Chile
Escuela de Ingeniería
Departamento de Ciencias de la Computación

IIC2213 — Lógica para Ciencias de la Computación — 1' 2021

Tarea 5

Algoritmos

Para la tarea se utilizaron los siguientes 2 algoritmos:

1. **Búsqueda Exhaustiva (Fuerza Bruta):** Se generan todas las combinaciones posibles de valores binarios para cada proposición (sin un orden particular), evaluándolas en la fórmula hasta encontrar una solución (en cuyo caso es satisfacible) o bien terminar y concluir que no es satisfacible.

2. **Custom DPLL:** Corresponde a una implementación del algoritmo **DPLL utilizando Backtracking,** pero aplicando las ideas de DPLL en forma de podas y heurísticas que encontré personalmente interesantes. En cada asignación de valor a una proposición, decidí aplicar una **evaluación parcial** de la fórmula (omitiendo las variables no asignadas), para así poder buscar soluciones prematuras o podar en caso de que alguna cláusula se volviera insatisfacible por ese camino. En cuanto a la heurística, decidí basarme en el **Unit Propagation de DPLL** y priorizar las proposiciones que se encontrasen solas en alguna cláusula que aún no fuese verdadera (y por lo tanto se puede deducir el valor de dichas proposiciones y encontrar más rápido inconsistencias). También incluí una **métrica de polaridad** de las proposiciones, que permite determinar si una proposición aparece predominantemente en la fórmula de forma directa o con su negación. Usando este valor de polaridad, elegí probar el valor de la proposición con un 0 primero si es que esta tenía polaridad negativa, y con un 1 si tenía polaridad positiva, ya que de esta forma se logra hacer verdaderas a la mayor cantidad de cláusulas dentro de la fórmula.

Resultados

A continuación se tiene un resumen de los resultados, con los tiempos medidos en **segundos:**

Tests/Algoritmos	Búsqueda Exhaustiva	Custom DPLL
20 SAT	Avg: 1.08 Min: 0.001 Max: 3.03	Avg: 0.03 Min: 0.004 Max: 0.09
50 SAT	Avg: INF Min: INF Max: INF	Avg: 10.72 Min: 0.03 Max: 30.02
50 UNSAT	Avg: INF Min: INF Max: INF	Avg: 28.62 Min: 10.74 Max: 54.97

Análisis

Para los tests satisfacibles de 20 proposiciones, se tiene que el algoritmo de fuerza bruta funciona bastante bien, demorándose alrededor de **1 segundo en promedio,** a pesar de que depende de la suerte que tenga al probar combinaciones. El algoritmo custom DPLL es **aprox. 36 veces más rápido en promedio,** lo cuál se debe a que su heurística y podas permiten llegar a una solución sin pasar por tantos estados. En el peor caso, el algoritmo de fuerza bruta debe evaluar $2^{20} = 1048576$ combinaciones, lo que es bastante razonable para la velocidad de una CPU moderna, razón por la que no hay una diferencia tan grande entre ambos algoritmos en este caso.

Para los tests satisfacibles de 50 proposiciones, el algoritmo de fuerza bruta **no fue capaz de terminar en un tiempo razonable** (fue detenido tras 1 hora de ejecución), lo que es bastante esperable dado que ahora el peor caso implica revisar $2^{50} = 1125899906842624$ combinaciones, lo que se escapa de la capacidad computacional actual (a excepción de computadores especializados en alto rendimiento). Por otro lado, custom DPLL **rinde de manera espectacular, demorándose menos de 11 segundos en promedio.** Aquí se nota la clara ventaja de realizar podas para descartar millones de combinaciones ingenuas, así como de realizar una búsqueda dirigida por una heurística que permita encontrar podas frecuente y rápidamente, reduciendo así la cantidad de estados totales a revisar de **forma drástica.**

Finalmente, para los tests **NO satisfacibles** de 50 proposiciones, como es de esperarse, el algoritmo de fuerza bruta no logra terminar, ya que como no es capaz de podar debe revisar exactamente las 2^{50} combinaciones. En cuanto a custom DPLL, logra un rendimiento muy eficiente, aunque **claramente menor a los tests satisfacibles,** ya que no encontrará ninguna solución y deberá descartar mediante podas tantos estados como pueda para determinar que cada fórmula es finalmente imposible de hacer verdad.

Como conclusión se tiene que el algoritmo custom DPLL es **muy apropiado para el problema SAT** en lógica para ciencias de la computación, y puede ser mejorado mediante una investigación y experimentación más profunda aplicando el estado del arte.