Caden Kroonenberg

Problem Set 6

EECS 649

2-26-22

6.1

A picture containing calendar

Description automatically generated

6.2

A picture containing diagram

Description automatically generated

B OR C – 12 models

(NOT A) OR (NOT B) OR (NOT C) OR (NOT D) – 15 models

(A IMPLIES B) AND A AND (NOT B) AND C AND D – 0 models (unsatisfiable)

6.3

A piece of paper with writing on it

Description automatically generated with medium confidence

We can observe the equivalence by comparing the truth table values (see above).

6.4

Diagram

Description automatically generated

6.5

|  |  |
| --- | --- |
| (1,2) | (2,2) |
| (1,1) | (2,1) |

a.

2MinesAdj\_[1,1] := true if exactly two adjacent spaces are mines

((X1,2 ^ X2,2) ⊕ (X1,2 ^ X2,1) ⊕ (X2,2 ^ X2,1) ⇔ 2MinesAdj\_[1,1]

b.

A CNF sentence asserting that k of n neighbors contain mines would include XORing terms of the form (X1 ^ X2 ^ … ^ Xk) where each X is a unique neighbor. Observe there are k terms being ANDed.

Ex. (some unique combination of k neighbors)1 ⊕ (some unique combination of k neighbors)2 ⊕ … ⊕ (some combination of k neighbors)n

c.

6.6

6.7

b. I modified the utility, is\_terminal, and print\_out functions to account for the new board size and then updated the test\_boards array to the new board values. This process took around 15 minutes.

c.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Board | Value | Nodes Examined | Terminals Examined | Unique States Encountered | Time Taken |
| 1 | 1 | 0 | 1 | 1 | 0.00011 s |
| 2 | -1 | 0 | 1 | 1 | 0.00042 s |
| 3 | 1 | 3 | 2 | 3 | 0.00012 s |
| 4 | 1 | 52 | 24 | 31 | 0.00062 s |
| 5 | 1 | 41,456 | 21,528 | 1,646 | 0.41115 s |
| 6 | 1 | 42,15,871 | 1,863,541 | 13,667 | 39.287 s |
| 7 |  |  |  |  |  |
| 8 |  |  |  |  |  |
| 9 |  |  |  |  |  |
| 10 |  |  |  |  |  |

d. I modified the minimax\_decision algorithm to initialize alpha and beta values to negative and positive infinity. I also combined the max\_value and min\_value moves to a single function called alphabeta which took the state, a min/max bool, as well as alpha and beta values as parameters. In this function, if maximizing, alpha = max(v, alpha) and break on v >= beta; if minimizing, beta = min(v, beta) and break on v <= alpha. This took around 15 minutes.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Board | Value | Optimal Move | Nodes Examined | Terminals Examined | Unique States Encountered | Time Taken |
| 1 | 1 | - | 0 | 1 | 1 | 0.00011 s |
| 2 | -1 | - | 0 | 1 | 1 | 0.00044 s |
| 3 | 1 | 3 | 3 | 2 | 3 | 0.00013 s |
| 4 | 1 | 0, 3, 12 | 21 | 8 | 20 | 0.0003 s |
| 5 | 1 | 0, 1, 2, 3, 12, 13, 14, 15 | 839 | 367 | 300 | 0.0087 s |
| 6 | 1 | 15 | 9,706 | 3,960 | 1,383 | 0.097 s |
| 7 | 0 | 1, …, 15 | 3,311,292 | 1,250,605 | 30,605 | 31.389 s |
| 8 | 0 | 0, 2, …, 15 | 4,198,437 | 1,562,071 | 37,621 | 40.574 s |
| 9 | 0 | 0 | 11,109,455 | 4,323,860 | 85,690 | 102.749 s |
| 10 (cutoff) | 0 | 0 | 65,959,808 | 26,272,385 | 150,691 | 649.54 s |

e.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Board | Value | Optimal Move | Nodes Examined | Terminals Examined | Unique States Encountered | Time Taken |
| 1 | 1 | - | 0 | 1 | 1 | 0.000088 s |
| 2 | -1 | - | 0 | 1 | 1 | 0.00004 s |
| 3 | 1 | 3 | 3 | 2 | 3 | 0.00011 s |
| 4 | 1 | 0, 3, 12 | 21 | 7 | 20 | 0.00026 s |
| 5 | 1 | 0, 1, 2, 3, 12, 13, 14, 15 | 496 | 133 | 299 | 0.00362 s |
| 6 | 1 | 15 | 2,877 | 568 | 1,382 | 0.017 s |
| 7 | 0 | 1, …, 15 | 65,867 | 7,337 | 30,605 | 0.282 s |
| 8 | 0 | 0, 2, …, 15 | 84,250 | 8,751 | 37,476 | 0.374 s |
| 9 | 0 | 0, …, 4, 6, …, 15 | 170,566 | 14,395 | 76,880 | 0.677 s |
| 10 | 0 | 0, …, 15 | 383,940 | 29,362 | 150,832 | 1.526 s |

The results are quite self-explanatory. Caching drastically decreases the time it takes to evaluate the value of each board. The relative decrease in time taken to evaluate boards is not equal for every board. The first few boards did not receive any noticeable decrease in time taken (the stated decrease was likely due to a difference in my own computer’s resources at the times in which both programs were executed). This is because the cache was empty at this time. It became progressively fuller as the program executed.