**ct-bess** N queens fol fix dbcc8bc 37 seconds ago

**1** contributor

228 lines (180 sloc) 3.31 KB

# Block Sort Readme

## Compiling:

This program was developed on Ubuntu 16.04 xenial using g++ 5.4.0

C++11 is required.

Please use your local machine or a server like `linux.tamu.edu` via ssh.

To compile run one of the following commands:

```
g++ -o main main.cpp -std=c++11
```

```
make
```

## Running:

To run the program type `./main` with or without problem flags.

To specify which problem to run, add the name of the problem to the next runtime argument.

You can run the program in your terminal by typing some of the following examples:

```
./main
```

```
./main test
```

```
./main 1a
```

> I **highly** recommend running the `test` file, because it is carefully crafted to work as intended. Careful running `falseTest` though, it will generate the empty clause but will take a long time before it truly decides `falseTest` is unsatisfiable.

## Output:

When and if a solution is found a corresponding output file is generated in `problemSet-SolutionTraces/`

For example: `./main test` produces `problemSet-SolutionTraces/test`

> **Note:** The program substitutes satisfied clauses with `( * )`. Meaning stared clauses are true, and empty clauses `( )` are false.

## 1d. FOL KB of the map of Australia

```
adjacent( WA, NT )
adjacent( WA, SA )

adjacent( NT, WA )
adjacent( NT, SA )
adjacent( NT, Q )

adjacent( Q, NT )
adjacent( Q, SA )
adjacent( Q, NSW )

adjacent( NSW, SA )
adjacent( NSW, V )

adjacent( V, SA )
adjacent( V, NSW )

adjacent( SA, WA )
adjacent( SA, NT )
adjacent( SA, Q )
adjacent( SA, NSW )
adjacent( SA, V )

adjacent( T )

color( SA, r )
color( WA, b )
color( NT, g )
color( Q, b )
color( NSW, g )
color( V, b )
color( T, r )
```

## 5. FOL KB for N-Queens Problem

Where `N = even integer`

`Queen( i, j )` means there is a queen in that `( row, col )`.

> No queen is implicitly defined by not claiming a queen is in that space.

```
Queen( 1, 2 )
Queen( 2, 4 )
Queen( 3, 6 )
...
Queen( N/2, N )

Queen( N/2 + 1, 1 )
Queen( N/2 + 2, 3 )
Queen( N/2 + 3, 5 )
...
Queen( N/2 + N/2 - 1, N - 3 )
Queen( N, N - 1 )
```

## Example Trace

From: `test.cnf`

```
0: (-A v -B )
1: ( A v  B )
2: (-B v  D )
3: (-A v  D )
4: (-B v -C )
5: ( B v  C )
6: (-C v -A )
7: (-C v -B )
8: ( A v  B v  C )

Model: {  }
Pure Symbol: true D
0: (-A v -B )
1: ( A v  B )
2: ( * )
3: ( * )
4: (-B v -C )
5: ( B v  C )
6: (-C v -A )
7: (-C v -B )
8: ( A v  B v  C )

Model: {  D  }
Chose: true A
0: (-B )
1: ( * )
2: ( * )
3: ( * )
4: (-B v -C )
5: ( B v  C )
6: (-C )
7: (-C v -B )
8: ( * )

Model: {  D  A  }
Unit Clause: false B
0: ( * )
1: ( * )
2: ( * )
3: ( * )
4: ( * )
5: ( C )
6: (-C )
7: ( * )
8: ( * )

Model: {  D  A -B  }
Unit Clause: true C
0: ( * )
1: ( * )
2: ( * )
3: ( * )
4: ( * )
5: ( * )
6: ( )
7: ( * )
8: ( * )

Model: {  D  A -B  C  }
Back-tracking...
Chose: false A
0: ( * )
1: ( B )
2: (-B v  D )
3: ( * )
4: (-B v -C )
5: ( B v  C )
6: ( * )
7: (-C v -B )
```

```
8: ( B v  C )

Model: { -A  }
Pure Symbol: true D
0: ( * )
1: ( B )
2: ( * )
3: ( * )
4: (-B v -C )
5: ( B v  C )
6: ( * )
7: (-C v -B )
8: ( B v  C )

Model: { -A  D  }
Unit Clause: true B
0: ( * )
1: ( * )
2: ( * )
3: ( * )
4: (-C )
5: ( * )
6: ( * )
7: (-C )
8: ( * )

Model: { -A  D  B  }
Pure Symbol: false C
0: ( * )
1: ( * )
2: ( * )
3: ( * )
4: ( * )
5: ( * )
6: ( * )
7: ( * )
8: ( * )

Model: { -A  D  B -C  }
The logic is satisfiable
Model: { -A  D  B -C  }
```