Assignment 11

Wednesday, December 7

This is the last assignment of the semester. These problems require you to use an SMT solver (Z3, CVC4, or CVC5) and a first-order theorem prover (Vampire). The lamr github repository contains Linux binaries in the folder `LAMR/bin`, but if you want to use the software on Windows or macOS, you need to replace these with the appropriate binaries, which you can find online.

Problem 1

The almost square of order n is a rectangle of size $n \times (n+1)$. The almost squares of orders 1 to 3 can fully cover the almost square of order 4. A solution is shown below.

```
1 1 3 3 3
2 2 3 3 3
2 2 3 3 3
2 2 3 3 3
```

In this exercise, we are going to use an SMT solver to determine whether the almost squares of order 1 to n can fully cover the almost square of order m. The encoding uses the theory QF_LIA. The encoding uses $4n$ variables: for the almost square of order `i`, we use variables `xmin_i`, `xmax_i`, `ymin_i`, and `ymax_i`. The variable `xmin_i` (`xmax_i`) denotes the first (last, respectively) row in which the almost square of order `i` is placed. Similarly, the variable `ymin_i` (`ymax_i`) denotes the first (last, respectively) column in which the almost square of order `i` is placed.

The covering of the almost square of order 4 shown above can be expressed using the following assignment to these variables

- `xmin_1 = 1`, `xmax_1 = 2`, `ymin_1 = 4`, `ymax_1 = 4`
- `xmin_2 = 1`, `xmax_2 = 2`, `ymin_2 = 1`, `ymax_2 = 3`
- `xmin_3 = 3`, `xmax_3 = 5`, `ymin_3 = 1`, `ymax_3 = 4`

The code fragment below shows the first part of the encoding used to compute the covering. It shows the declaration of the first variables and the first constraints on those variables.

```
(set-logic QF_LIA)
(declare-const xmin_1 Int)
(declare-const xmax_1 Int)
(declare-const ymin_1 Int)
(declare-const ymax_1 Int)
...
(assert (and (>= xmin_1 1) (<= xmax_1 5)))
(assert (and (>= ymin_1 1) (<= ymax_1 4)))
...
```

Finish the encoding use the following steps:

a) (3 points)

Express constraints that ensure that the almost square of order `i` covers exactly a subgrid of $n \times (n+1)$ or $(n+1) \times n$. The only variables that you can use are `xmin_i`, `xmax_i`, `ymin_i`, and `ymax_i`. Hint: Split the constraint into three parts with one part that enforces the relation between `xmin_i` and `xmax_i`, one part that enforces the relation between `ymin_i` and `ymax_i`, and one part that enforces the relation between all four variables. You will get better results if you express the last part as a conjunction of linear constraints, without using disjunction.

b) (3 points)

For each pair of almost squares, express the constraint that they cannot overlap each other, i.e., there is no cell that is covered by multiple almost squares.

c) (3 points)

Determine a grid assignment showing that the almost squares of orders 1 to 8 can fully cover the almost square of order 15. SMT solvers should be able to quickly solve the intended encoding. Print the grid assignment in a format similar to the grid assignment shown above.

d) (3 points)

Encode the same problem using the theory QF_BV and compare the runtimes between both theories. It is important to use signed bitvectors. The signed bitverctor operations for $<$, $\leq$, $>$, and $\geq$ are `bvslt`, `bvsle`, `bvsgt`, and `bvsge`, respectively. Addition is `bvadd` and subtraction `bvsub`. The declaration of the variables and the initial bounds for these variables are shown below for $n = 3$ and $m = 4$. Note that for bitvectors, CVC4/5 are slow, Z3 a bit faster and Boolector is fastest.

```
(set-logic QF_BV)
(declare-const xmin_1 (_ BitVec 16))
(declare-const xmax_1 (_ BitVec 16))
(declare-const ymin_1 (_ BitVec 16))
(declare-const ymax_1 (_ BitVec 16))
...
(assert (and (bvsge xmin_1 #x0001) (bvsle xmin_1 #x0005)))
(assert (and (bvsge xmax_1 #x0001) (bvsle xmax_1 #x0005)))
(assert (and (bvsge ymin_1 #x0001) (bvsle ymin_1 #x0004)))
(assert (and (bvsge ymax_1 #x0001) (bvsle ymax_1 #x0004)))
...
```

e) Bonus point!

The same encoding can also be used to cover the almost square of order 55 with the almost squares of order 1 to 20. Solving this formula can take minutes. Give it a try.

Problem 2

Use Vampire to verify each of these inferences. As a reality check, also make sure that Vampire cannot refute the hypotheses.

a) (4 points)

Consider these hypotheses:

1. Every honest and industrious person is healthy.
2. No grocer is healthy.
3. Every industrious grocer is honest.
4. Every cyclist is industrious.
5. Every unhealthy cyclist is dishonest.

Show that these imply that no grocer is a cyclist.

b) (4 points)

Consider these hypotheses:

1. Wolves, foxes, birds, caterpillars, and snails are animals, and there are some of each of them.
2. There are some grains, and grains are plants.
3. Every animal either likes to eat all plants or all animals smaller than itself that like to eat some plants.
4. Caterpillars and snails are smaller than birds, which are smaller than foxes, which in turn are smaller than wolves.
5. Wolves do not like to eat foxes or grains, while birds like to eat caterpillars but not snails.
6. Caterpillars and snails like to eat some plants.

Show that these imply that there is an animal that likes to eat an animal that eats grains.