Automated reasoning Assignment 1

Mart Lubbers (s4109503)

October 19, 2015

1 Problem 1

Six trucks have to deliver pallets of obscure building blocks to a magic factory. Every truck has a capacity of 7800 kg and can carry at most eight pallets. In total, the following has to be delivered:

- *Four pallets of nuzzles, each of weight 700 kg.*
- *A number of pallets of prittles, each of weight 800 kg.*
- *Eight pallets of skipples, each of weight 1000 kg.*
- *Ten pallets of crottles, each of weight 1500 kg.*
- *Five pallets of dupples, each of weight 100 kg.*

Prittles and crottles are an explosive combination: they are not allowed to be put in the same truck.
Skipples need to be cooled; only two of the six trucks have facility for cooling skipples. Dupples are very valuable; to distribute the risk of loss no two pallets of dupples may be in the same truck.
Investigate what is the maximum number of pallets of prittles that can be delivered, and show how for that number all pallets may be divided over the six trucks.

1.1 Formal definition

For every truck t_i for $i \in [1 \dots 6]$ in combination with every nuzzle n, prittle p, skipple s, crottle c and dupple d we declare a variable that holds the amount of that type of building block. For truck 1 we thus have the variables: $t_1n, t_1p, t_1s, t_1c, t_1d$.

To lay a contraint on the weight we declare for every truck the following rule.

$$\bigwedge_{i=1}^{T} (t_in * 700 + t_ip * 800 + t_is * 1000 + t_ic * 1500 + t_id * 100 < 7800)$$

To limit the maximum number of pallets in a truck we define.

$$\bigwedge_{i=1}^{T} \left(\left(\sum_{p \in \{n,p,s,c,d\}} t_ip \right) \leq 8 \right)$$

To limit the minimum number of pallets in a truck we define.

$$\bigwedge_{i=1}^{T} \bigwedge_{p \in \{n,p,s,c,d\}} t_ip > 0$$

To describe the number of pallets available we define for every type of pallet a variable that describes the number available if there is a limited number available.

$$num_n = 4 \wedge num_s = 8 \wedge num_c = 10 \wedge num_d = 5$$

To be sure the materials are all delivered we define.

$$\bigwedge_{p \in \{n,s,c,d\}} \left(\left(\sum_{i=1}^{T} t_i p \right) = num_p \right)$$

The first constraint is that prittles and crottles can not be in the same truck. This is easily described with.

$$\bigwedge_{i=1}^{T} (t_i p = 0 \vee t_i c = 0)$$

Skipples need to be cooled and only two trucks have cooling. Therefore we specify.

$$\bigwedge_{i=3}^{T} t_i s = 0$$

Dupples can only be in a truck with a maximum of two.

$$\bigwedge_{i=1}^{T} t_i d \leq 1$$

We can tie this together by putting $\wedge$ symbols between all of the formulas.

1.2 SMT format solution

The formula is easily convertable to SMT format and is listed in Listing 3. A final condition is added with a special variable named `<REP>` that we increment to find the maximum amount of prittles transportable. When running the script in Listing 1 the loop terminates after it echoed 20 meaning that the maximum number of prittles is 20. This iterative solution takes less then 0.1 seconds to calculate.

Listing 1: Iteratively find the largest solution

```
1 i=1
2 while [ $(sed "s/<REP>/$i/g" a1.smt | yices-smt) = "sat" ]
3 do
4   echo $((++i));
5 done
```

1.3 Solution

Truck	Nuzzles	Prittles	Skipples	Crottles	Dupples	Weight	Pallets
1	0	0	4	2	1	7100	7
2	0	3	4	0	1	6500	8
3	0	8	0	0	0	6400	7
4	0	7	0	0	1	5700	8
5	0	0	0	5	1	7600	6
6	4	0	0	3	1	7400	8
total	4	18	8	10	5		

2 Problem 4

Seven integer variables $a_1; a_2; a_3; a_4; a_5; a_6; a_7$ are given, for which the initial value of a_i is i for $i = 1, \ldots, 7$. The following steps are defined: choose i with $1 < i < 7$ and execute

$$a_i := a_{i-1} + a_{i+1}$$

that is, a_i gets the sum of the values of its neighbors and all other values remain unchanged. Show how it is possible that after a number of steps a number of steps a number ≥ 50 occurs at least twice in $a_1; a_2; a_3; a_4; a_5; a_6; a_7$.

2.1 Formal definition

Precondition Say we have I a's denoted as a_i and we have N iterations. ${}_{i_n}a_i$ means variable a_i in iteration n. Iteration 0 is seen as the starting point and can be expressed as in Equation 1

$$(a_1 = 1) \wedge (a_I = I) \wedge \bigwedge_{k=2}^{I-1} ({}_{i_0}a_i = i) \tag{1}$$

Program Every iteration we can choose to do $a_i = a_{i-1} + a_{i+1}$ or nothing. To keep track of what we do we keep a counter c_n for every n that holds either the i if an a_i is chosen or 0 if no action has been taken. Therefore for all iterations we can express this as in Equation 2

$$\bigwedge_{n=1}^{N} \left(\bigvee_{k=2}^{I-1} \left((c_i{=}k) \wedge ({}_{i_n}a_j {=} {}_{i_{n-1}}a_{j-1} + {}_{i_{n-1}}a_{j+1}) \wedge \bigwedge_{j=2}^{I-1} ((j{\neq}k) \wedge ({}_{i_n}a_j {=} {}_{i_{n-1}}a_j)) \right) \vee \bigwedge_{k=2}^{I-1} ({}_{i_n}a_k {=} {}_{i_{n-1}}a_k) \wedge (c_i{=}0) \right) \tag{2}$$

Postcondition Finally the post condition can be described as $a_i > 50$ and some other $a_i = a_j \wedge i \neq j$ for all i. This is expressed in Equation 3

$$\bigvee_{k=2}^{I-1} \left(({}_{i_N}a_k >= 50) \wedge \left(\bigvee_{j=2}^{I-1} ({}_{i_N}a_k = {}_{i_N}a_j) \wedge (k \neq j) \right) \right) \tag{3}$$

Total To tie this all together we just put $\wedge$ in between and that results in:

$$\text{precondition} \ \wedge \ \text{program} \ \wedge \ \text{postcondition}$$

2.2 SMT format solution

Naming the precondition, program and postcondition respectively p_1, p_2, p_3 we can easily convert it to a SMT format. The converting is tedious and takes a lot of time and therefore an automatization script has been created that is visible in the appendices in Listing 4. The script automatically assumes 11 iterations and 7 a_i variables but via command line arguments this is easily extendable. To determine the minimal number of iterations a simple bash script can be made that iteratively increases the iterations as shown in Listing 2. The shortest solution with length 11 is found in around 30 seconds. Finding the smallest solution length incrementally takes around 75 seconds.

Listing 2: Iteratively find the shortest solution

```
1 i=1
2 while [ "$(python a4.py $i | yices-smt)" = "unsat" ]
3 do
4   echo $((++i));
5 done
```

2.3 Solution

The bold cells represent the a_i after applying the function. After ten iterations cell a_2 and a_6 both hold 54 thus satisfying the problem specification.

#	i	a_2	a_3	a_4	a_5	a_6
0	–	2	3	4	5	6
1	4	2	3	**8**	5	6
2	5	2	3	8	**14**	6
3	2	**4**	3	8	14	6
4	4	4	3	**17**	14	6
5	5	4	3	17	**23**	6
6	6	4	3	17	23	**30**
7	5	4	3	17	**47**	30
8	4	4	3	**50**	47	30
9	3	4	**54**	50	47	30
10	6	4	54	50	47	**54**

3 Appendix

Listing 3: a1.smt

```
(benchmark a1.smt
:logic QF_UFLIA
:extrafuns (
  (t1p Int) (t1n Int) (t1s Int) (t1c Int) (t1d Int)
  (t2p Int) (t2n Int) (t2s Int) (t2c Int) (t2d Int)
  (t3p Int) (t3n Int) (t3s Int) (t3c Int) (t3d Int)
  (t4p Int) (t4n Int) (t4s Int) (t4c Int) (t4d Int)
  (t5p Int) (t5n Int) (t5s Int) (t5c Int) (t5d Int)
  (t6p Int) (t6n Int) (t6s Int) (t6c Int) (t6d Int)
)
:formula
(and
  (>= t1p 0) (>= t1n 0) (>= t1s 0) (>= t1c 0) (>= t1d 0)
  (>= t2p 0) (>= t2n 0) (>= t2s 0) (>= t2c 0) (>= t2d 0)
  (>= t3p 0) (>= t3n 0) (>= t3s 0) (>= t3c 0) (>= t3d 0)
  (>= t4p 0) (>= t4n 0) (>= t4s 0) (>= t4c 0) (>= t4d 0)
  (>= t5p 0) (>= t5n 0) (>= t5s 0) (>= t5c 0) (>= t5d 0)
  (>= t6p 0) (>= t6n 0) (>= t6s 0) (>= t6c 0) (>= t6d 0)

  (<= (+ t1p t1n t1s t1c t1d) 8)
  (<= (+ t2p t2n t2s t2c t2d) 8)
  (<= (+ t3p t3n t3s t3c t3d) 8)
  (<= (+ t4p t4n t4s t4c t4d) 8)
  (<= (+ t5p t5n t5s t5c t5d) 8)
  (<= (+ t6p t6n t6s t6c t6d) 8)

  (<= (+ (* t1n 700) (* t1p 800) (* t1s 1000) (* t1c 1500) (* t1d 100)) 7800)
  (<= (+ (* t2n 700) (* t2p 800) (* t2s 1000) (* t2c 1500) (* t2d 100)) 7800)
  (<= (+ (* t3n 700) (* t3p 800) (* t3s 1000) (* t3c 1500) (* t3d 100)) 7800)
  (<= (+ (* t4n 700) (* t4p 800) (* t4s 1000) (* t4c 1500) (* t4d 100)) 7800)
  (<= (+ (* t5n 700) (* t5p 800) (* t5s 1000) (* t5c 1500) (* t5d 100)) 7800)
  (<= (+ (* t6n 700) (* t6p 800) (* t6s 1000) (* t6c 1600) (* t6d 100)) 7800)

  (= (+ t1n t2n t3n t4n t5n t6n) 4)
  (= (+ t1s t2s t3s t4s t5s t6s) 8)
  (= (+ t1c t2c t3c t4c t5c t6c) 10)
  (= (+ t1d t2d t3d t4d t5d t6d) 5)

  (or (= 0 t1p) (= 0 t1c))
  (or (= 0 t2p) (= 0 t2c))
  (or (= 0 t3p) (= 0 t3c))
  (or (= 0 t4p) (= 0 t4c))
  (or (= 0 t5p) (= 0 t5c))
  (or (= 0 t6p) (= 0 t6c))

  (= t3s 0)
  (= t4s 0)
  (= t5s 0)
  (= t6s 0)

  (<= t1d 1)
  (<= t2d 1)
  (<= t3d 1)
  (<= t4d 1)
  (<= t5d 1)
  (<= t6d 1)

  (>= (+ t1p t2p t3p t4p t5p t6p) <REP>)
)
)
```

Listing 4: a4.py

```
#!/usr/bin/env python3
import sys
iterations = int(sys.argv[1]) if len(sys.argv) > 1 else 11
numa = int(sys.argv[2]) if len(sys.argv) > 2 else 7

##Print preamble
print("(benchmark a4.smt")
print(":logic QF_UFLIA")

##Print variables
print(":extrafuns (")
print("(a{} Int)".format(1))
print("(a{} Int)".format(numa))
for i in range(iterations):
    for v in range(2,numa):
        print("(i{}a{} Int) ".format(i,v))
    print("(c{} Int)".format(i))
print(")")

##Print preconditions
print(":formula")
print("(and")
print("(= c0 0)")
print("(= a1 1)")
print("(= a{0} {0})".format(numa, numa))
for i in range(2,numa):
    print("(= i0a{0} {0})".format(i))

##Print iterations
for i in range(1, iterations):
    print("(or")
    for v in range(2, numa):
        print("(and")
        print("(= c{} {})".format(i, v))
        for ov in [k for k in range(2, numa) if k != v]:
            print("(= i{0}a{1} i{2}a{1})".format(i, ov, i-1))
        im1 = 'a1' if v == 2 else 'i{}a{}'.format(i, v-1)
        ip1 = 'a{}'.format(numa) if v == numa-1 else 'i{}a{}'.format(i-1, v+1)
        print("(= i{}a{} (+ {} {}))".format(i, v, im1, ip1))
        print(")")
    print("(and")
    print("(= c{} 0)".format(i))
    for ov in range(2, numa):
        print("(= i{0}a{1} i{2}a{1})".format(i, ov, i-1))
    print(")")
    print(")")


## Post conditions
print("(or")
for v in range(2, numa):
    print("(and (>= i{}a{} 50)".format(iterations-1, v))
    print("(or")
    for ov in [k for k in range(2, numa) if k != v]:
        print("(= i{0}a{1} i{0}a{2})".format(iterations-1, v, ov))
    print("))")
print(")")

## Close the and,benchmark parenthesis
print("))")
```