Practical Assignment Automated Reasoning 2IW15

Irfan Nur Afif
student number: 1035476
email: irfan.nur.afif@.student.tue.nl

Problem: Pallets delivery

1. 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 eight trucks.
2. Do the same, with the extra information that prittles and crottles are an explosive combination: they are not allowed to be put in the same truck.

Solution:

We introduce five sets of variables for every type of pallets: a for nuzzles, b for prittles, c for skipples, d for crottles, e for dupples. For every variable $k \in \{a, b, c, d, e\}$, we divide it again into $k_i, i = 1, \cdots 8$ (according to the number of trucks) representing the number of pallets k in truck i.

For a truck, it has maximum capacity of 8 pallets. This is expressed by the formula

$$a_i + b_i + c_i + d_i + e_i \leq 8, \text{ for } i = 1, 2, \cdots 8$$

Each pallets k has a weight w_k kg. Each truck also has a maximum weight of 8,000 kg. This is expressed by the formula:

$$w_a a_i + w_b b_i + w_c c_i + w_d d_i + w_e e_i \leq 8000, \text{ for } i = 1, 2, \cdots 8$$

where $w_a = 800$ (corresponds to nuzzles weight in kg), $w_b = 1100$, $w_c = 1000$, $w_d = 2500$, $w_3 = 200$. For each pallets, there are a number of items that needs to be delivered by trucks. This is represented by the formula

$$\sum_{i=1}^{8} a_i \leq p_a \wedge \sum_{i=1}^{8} b_i \leq p_b \wedge \sum_{i=1}^{8} c_i \leq p_c \wedge \sum_{i=1}^{8} d_i \leq p_d \wedge \sum_{i=1}^{8} e_i \leq p_e$$

where $p_a = 4$ (corresponds to number of nuzzles), p_b is unknown, $p_c = 8$, $p_d = 10$, $p_e = 12$. There are also two additional constraints. The first is that there are only three trucks that can carry the skipples. The way we represents this is by putting the first three trucks as the trucks that has cooling facility (thus it can carry skipples). Since the ordering of the trucks is not constrained in answering the problem, thus we can express it by stating truck 4 until 8 can't carry skipples, which is

$$\bigwedge_{i=4,\cdots 8} c_i = 0$$

The second constraint is no two pallets of nuzzles may be in the same truck. This means that each trucks can only carry a maximum of 1 nuzzle. This can be represented by this formula

$$\bigwedge_{i=1,\cdots 8} a_i \leq 1$$

Last but not least, for every variables, it has to be greater than 0 because the minuimum number of a pallet in the truck is 0.

$$(\bigwedge_{i=1,\cdots 8} a_i \geq 0) \wedge (\bigwedge_{i=1,\cdots 8} b_i \geq 0) \wedge (\bigwedge_{i=1,\cdots 8} c_i \geq 0) \wedge (\bigwedge_{i=1,\cdots 8} d_i \geq 0) \wedge (\bigwedge_{i=1,\cdots 8} e_i \geq 0)$$

Finally, we can solve the maximum number of p_b by trying the biggest integer numbers from the term `(= (+ b1 b2 b3 b4 b5 b6 b7 b8)` p_b`)` that yields sat results. We know the upper limit of p_b candidate. If each of the 8 trucks can carry at most 8 pallets, there has to be at most 64 pallets. Thus, the upper limit of the candidate is $64 - p_a - p_c - p_d - p_e = 22$. We try $p_b = 22$ but the yices-smt solver returns an unsat result. But, when we try $p_b = 21$, the result is sat. Thus, the answer for the first question is 21. Here is the code to solve the first question: (full code can be seen at: TODO)

```
(benchmark test.smt
:logic QF_UFLIA
:extrafuns
((a1 Int) (a2 Int) (a3 Int) (a4 Int) (a5 Int) (a6 Int) (a7 Int) (a8 Int)
(b1 Int) (b2 Int) (b3 Int) (b4 Int) (b5 Int) (b6 Int) (b7 Int) (b8 Int)
(c1 Int) (c2 Int) (c3 Int) (c4 Int) (c5 Int) (c6 Int) (c7 Int) (c8 Int)
(d1 Int) (d2 Int) (d3 Int) (d4 Int) (d5 Int) (d6 Int) (d7 Int) (d8 Int)
(e1 Int) (e2 Int) (e3 Int) (e4 Int) (e5 Int) (e6 Int) (e7 Int) (e8 Int)
)
:formula
( and
(<= (+ (* 800 a1) (* 1100 b1) (* 1000 c1) (* 2500 d1) (* 200 e1)) 8000)
(<= (+ (* 800 a2) (* 1100 b2) (* 1000 c2) (* 2500 d2) (* 200 e2)) 8000)
(<= (+ (* 800 a3) (* 1100 b3) (* 1000 c3) (* 2500 d3) (* 200 e3)) 8000)
(<= (+ (* 800 a4) (* 1100 b4) (* 1000 c4) (* 2500 d4) (* 200 e4)) 8000)
(<= (+ (* 800 a5) (* 1100 b5) (* 1000 c5) (* 2500 d5) (* 200 e5)) 8000)
(<= (+ (* 800 a6) (* 1100 b6) (* 1000 c6) (* 2500 d6) (* 200 e6)) 8000)
(<= (+ (* 800 a7) (* 1100 b7) (* 1000 c7) (* 2500 d7) (* 200 e7)) 8000)
(<= (+ (* 800 a8) (* 1100 b8) (* 1000 c8) (* 2500 d8) (* 200 e8)) 8000)
(<= (+ a1 b1 c1 d1 e1) 8)
(<= (+ a2 b2 c2 d2 e2) 8)
(<= (+ a3 b3 c3 d3 e3) 8)
(<= (+ a4 b4 c4 d4 e4) 8)
(<= (+ a5 b5 c5 d5 e5) 8)
```

```
(<= (+ a6 b6 c6 d6 e6) 8)
(<= (+ a7 b7 c7 d7 e7) 8)
(<= (+ a8 b8 c8 d8 e8) 8)
(<= (+ (* 800 a2) (* 1100 b2) (* 1000 c2) (* 2500 d2) (* 200 e2)) 8000)
(<= (+ (* 800 a3) (* 1100 b3) (* 1000 c3) (* 2500 d3) (* 200 e3)) 8000)
(<= (+ (* 800 a4) (* 1100 b4) (* 1000 c4) (* 2500 d4) (* 200 e4)) 8000)
(<= (+ (* 800 a5) (* 1100 b5) (* 1000 c5) (* 2500 d5) (* 200 e5)) 8000)
(<= (+ (* 800 a6) (* 1100 b6) (* 1000 c6) (* 2500 d6) (* 200 e6)) 8000)
(<= (+ (* 800 a7) (* 1100 b7) (* 1000 c7) (* 2500 d7) (* 200 e7)) 8000)
(<= (+ (* 800 a8) (* 1100 b8) (* 1000 c8) (* 2500 d8) (* 200 e8)) 8000)
(= (+ a1 a2 a3 a4 a5 a6 a7 a8) 4)
;main!!!!
(= (+ b1 b2 b3 b4 b5 b6 b7 b8) 21)
(= (+ c1 c2 c3 c4 c5 c6 c7 c8) 8)
(= (+ d1 d2 d3 d4 d5 d6 d7 d8) 10)
(= (+ e1 e2 e3 e4 e5 e6 e7 e8) 20)
(>= a1 0)
(>= a2 0)
(>= a3 0)
(>= a4 0)
(>= a5 0)
(>= a6 0)
(>= a7 0)
(>= a8 0)
(>= b1 0)
(>= b2 0)
(>= b3 0)
(>= b4 0)
(>= b5 0)
(>= b6 0)
(>= b7 0)
(>= b8 0)
(>= c1 0)
(>= c2 0)
(>= c3 0)
(>= c4 0)
(>= c5 0)
(>= c6 0)
(>= c7 0)
(>= c8 0)
(>= d1 0)
(>= d2 0)
(>= d3 0)
(>= d4 0)
(>= d5 0)
(>= d6 0)
(>= d7 0)
(>= d8 0)
(>= e1 0)
(>= e2 0)
(>= e3 0)
```

```
(>= e4 0)
(>= e5 0)
(>= e6 0)
(>= e7 0)
(>= e8 0)
;only three of the eight trucks have facility for cooling skipples, assume t1 2 3
(= c4 0)
(= c5 0)
(= c6 0)
(= c7 0)
(= c8 0)
;no two pallets of nuzzles may be in the same truck
(<= a1 1)
(<= a2 1)
(<= a3 1)
(<= a4 1)
(<= a5 1)
(<= a6 1)
(<= a7 1)
(<= a8 1)
)
)
```

Applying `yices-smt -m no1a.smt` yields the following result within a fraction of a second: `sat`

```
(= c6 0)
(= e8 4)
(= a4 1)
(= a6 0)
(= a5 1)
(= c3 1)
(= b5 1)
(= c5 0)
(= e1 0)
(= c7 0)
(= e4 4)
(= b2 3)
(= b4 1)
(= c2 2)
(= d5 2)
(= d1 0)
(= d2 1)
(= e6 1)
(= e2 1)
(= e3 2)
(= a1 1)
(= b6 7)
(= c4 0)
(= c8 0)
(= b7 2)
(= e5 4)
(= a8 0)
```

```
(= d6 0)
(= d7 2)
(= a3 1)
(= d3 1)
(= d4 2)
(= e7 4)
(= b8 2)
(= d8 2)
(= c1 5)
(= a2 0)
(= a7 0)
(= b1 2)
(= b3 3)
```

Expressed in a picture this yields: TODO

In answering second question, there is an additional constraint that we have to implement. Prittles and crottles are an explosive combination: they are not allowed to be put in the same truck. This means that in a truck, the amount of prittles or crottles has to be zero (or both). Which means

$$\bigwedge_{i=1,\cdots 8} (b_i = 0 \vee d_i = 0)$$

Implementing the second part is just adding some additional line to the first part as follows

```
;prittles and crottles are an explosive combination:  they are not allowed to be put in
the same truck.
(or (= b1 0) (= d1 0))
(or (= b2 0) (= d2 0))
(or (= b3 0) (= d3 0))
(or (= b4 0) (= d4 0))
(or (= b5 0) (= d5 0))
(or (= b6 0) (= d6 0))
(or (= b7 0) (= d7 0))
(or (= b8 0) (= d8 0))
```

Again, we have to find the biggest value of p_b using the same rule as the first part, starting from $p_b = 22$. The biggest p_b that we yielding sat result is 19. The output of `yices-smt -m no1a.smt` yields the following result within a fraction of a second

```
sat
(= c6 0)
(= e8 1)
(= a4 0)
(= a6 1)
(= a5 0)
(= c3 2)
(= b5 0)
(= c5 0)
(= e1 4)
```

```
(= c7 0)
(= e4 1)
(= b2 0)
(= b4 7)
(= c2 5)
(= d5 3)
(= d1 2)
(= d2 1)
(= e6 1)
(= e2 2)
(= e3 4)
(= a1 1)
(= b6 6)
(= c4 0)
(= c8 0)
(= b7 0)
(= e5 2)
(= a8 1)
(= d6 0)
(= d7 2)
(= a3 0)
(= d3 2)
(= d4 0)
(= e7 5)
(= b8 6)
(= d8 0)
(= c1 1)
(= a2 0)
(= a7 1)
(= b1 0)
(= b3 0)
```

Expressed in a picture this yields: TODO

We check that indeed there are eight queens for which no two are on the same row, column or diagonal.

Remark:

Our formula contains some redundancy: the requirement that every row contains exactly one queen implies that there are exactly n queens in total. By expressing that every column contains at least one queen, from this property one concludes that also every column contains at most one queen. We chose for this redundancy for keeping the symmetry in the solution, and also following the general strategy that introducing redundancy is often good for efficiency.

Generalization:

As we generalized our approach for n rather than 8, it is interesting to see what happens for other values of n. For $n > 10$ we have to take care of the notation: if we keep the notation then it is not clear whether `p111` represents $p_{1,11}$ or $p_{11,1}$. This is solved by putting an extra symbol between the two numbers.

For $n = 3$ the resulting formula is unsatisfiable, showing that there is no solution. For $n = 4, 5, 6, \ldots$ the formula is satisfiable, by which is a solution of the problem is found. Efficiency is not a problem: for $n = 60$ there are 3600 variables and the formula consists of over 350,000 clauses, but still `yices` succeeds in finding a solution within a few seconds.

Problem: Designing chip

Give a chip design containing two power components and ten regular components satisfying the constraints. What if this last distance requirement of 18 is increased to 20? And what if it increased to 22?

Solution:

Let's say the wide of the chip is w and its height is h. We can imagine it as 2 dimension plane that spans from $(0, 0)$ to (w, h). We have k normal component $N = n_1, n_2, \cdots n_k$ and l power component $P = p_1, p_2, \cdots p_l$. Each a component has wide w_a and height h_a. Let x_a, y_a be the bottom-left coordinate of component a in the chip. We also introduce variable z_a for every component a, which is an integer that has value of 0 or 1. If z_a equals 0 then component a is not rotated 90^0, that means it still has w_a as its wide and h_a as its height. If z_a is 1, then a is rotated 90^0, thus it has a wide of h_a and the height is w_a. This is because in the constraint, there is a rule that a component may be rotated. To simplify the representation we will number every component from 1 to $k + l$, where the first k components represents the normal components, and component $k+1, \cdots l$ are power components.

The first rule that we should make is to limit the value of z to strictly 0 and 1. This can be formulated as follows:

$$\bigwedge_{i=1,\cdots(k+l)} (z_i \geq 0 \wedge z_i \leq 1)$$

The second rule is that a component may only be placed within the chip size. Since a component may have a z-value of 0 or 1, we need to build 2 cases. This can be formulated as follows:

$$\bigwedge_{i=1,\cdots(k+l)} \Bigg(w_i \geq 0 \wedge h_i \geq 0 \wedge$$

$$\Bigg((x_i + w_i \leq w \wedge y_i + h_i \leq h \wedge z_i = 0 \Bigg) \vee \Bigg(x_i + h_i \leq w \wedge y_i + w_i \leq h \wedge z_i = 1) \Bigg) \Bigg)$$

There are 3 main terms in the conjuction rule above. The first and second are clear: x_i and y_i of the i-th component has to be greater than 0. Then the third clauses, we split it into disjuction of two cases: the first is when $z_i = 0$ and the second case is when $z_i = 1$. We choose disjuction because it is a representaion of a choice/possibility. In the first cases, there are 3 disjunction clauses: first is a clause that stated $x_i + w_i \leq w$ (which means the bottom right corner of the component cannot be greater than chip size), second clause is $y_i + h_i \leq h$ (which means the top right corner of the component cannot be greater than chip size) and the last clause is $z_i = 0$ which means we don't rotate the component. Similarly in the second case, the three clauses are similar. Since this is the case where we do the rotation, then we have $x_i + h_i \leq w \wedge y_i + w_i \leq h \wedge z_i = 1$ (the wide of the rotated component is the height of the unrotated component, and also applies to the height).

The next problem is stating the non-overlap constraint. This can be expressed in this formula:

$$\bigwedge_{i=1,\cdots k+l; j=i+1,\cdots k+l} \Bigg(\bigg(\Big(x_i + w_i \leq x_j \vee x_j + w_j \leq x_i \vee y_i + h_i \leq y_j \vee y_j + h_j \leq y_i \Big) \wedge \Big(z_i = 0 \Big) \wedge \Big(z_j = 0 \Big) \bigg)$$

$$\vee \bigg(\Big(x_i + w_i \leq x_j \vee x_j + h_j \leq x_i \vee y_i + h_i \leq y_j \vee y_j + w_j \leq y_i \Big) \wedge \Big(z_i = 0 \Big) \wedge \Big(z_j = 1 \Big) \bigg)$$

$$\vee \bigg(\Big(x_i + h_i \leq x_j \vee x_j + w_j \leq x_i \vee y_i + w_i \leq y_j \vee y_j + h_j \leq y_i \Big) \wedge \Big(z_i = 1 \Big) \wedge \Big(z_j = 0 \Big) \bigg)$$

$$\vee \bigg(\Big(x_i + h_i \leq x_j \vee x_j + h_j \leq x_i \vee y_i + w_i \leq y_j \vee y_j + w_j \leq y_i \Big) \wedge \Big(z_i = 1 \Big) \wedge \Big(z_j = 1 \Big) \bigg) \Bigg)$$

We take two distinct (in terms of component number, not size) pairs of every components in the chips. For every pair (i, j), we do a disjunction of 4 cases:

1. Component i and j are not rotated ($z_i = 0 \wedge z_j = 0$). In this case there are 4 possibilities:

    (a) Component i is located on the left of component j, that is $x_i + w_i \leq x_j$.

    (b) Component i is located on the right of component j, that is $x_j + w_j \leq x_i$.

    (c) Component i is located on the bottom of component j, that is $y_i + h_i \leq y_j$.

    (d) Component i is located on the upper of component j, that is $y_j + h_j \leq y_i$.

2. Component i is not rotated and j is rotated ($z_i = 0 \wedge z_j = 1$). Similar with the first case, but now we swap the value of w_j and h_j because j is rotated.

3. Component i is rotated and j is not rotated ($z_i = 1 \wedge z_j = 0$). Similar with the first case, but now we swap the value of w_i and h_i because i is rotated.

4. Component i and j are rotated ($z_i = 1 \wedge z_j = 1$). Similar with the first case, but now we swap the value of w_i and h_i also w_j and h_j because i and j are rotated.

The next constraint is "all regular components should directly be connected to a power component, that is, an edge of the component should have at least one point in common with an edge of the power component". This can be formulated as follows.

$$\bigwedge_{i=1,\cdots k} \left(\bigvee_{j=k+1,\cdots l} \Big((x_i = x_j) \vee (y_i = y_j) \Big) \right)$$

For each normal component (numbered from 1 to k), we compare each of its edge to the edge of the power component (numbered from $k+1$ to l). We only need one point in common with power component for every normal component, thus a disjuction for every comparison with power component is sufficient. In each comparison, we just need to check whether it shares the x-value ($x_i = x_j$) or y-value ($y_i = y_j$).

In this problem we have $w = 30$, $h = 30$, $k = 10$, $l = 2$

Problem: Jobs scheduling

Find a solution of this scheduling problem for which the total running time is minimal

Solution:

We introduce 12 variables for each job: $t_1, t_2, \cdots, t_{12}$ representing the starting time of job $1, 2, \cdots, 12$ respectively. Each of the job has duration $d_1, d_2, ...d_{12}$. Most of the constrain in this problem is in form "Job x may only start if jobs y_1 and y_2 and ... y_n have been finished." This means that the starting time of job x is always greater or equal than the time job $y_1, y_2, \cdots y_n$ finishes which is $t_{y_1} + d_{y_1}, t_{y_2} + d_{y_2}, \cdots t_{y_n} + d_{y_n}$ respectively. This statement can be formulated as follows:

$$\bigwedge_{i=1,\cdots n} t_x \geq t_{y_i} + d_{y_i}$$

In this problem, we have $d_1, d_2, \cdots d_{12}$ equal to $6, 7, \cdots 12$ respectively. For each of the x we construct like above (TODO)

Another form of the constraint is "Job x may not start earlier than job $y_1, y_2, \cdots y_n$." It means that the starting time of job x has to be greater than or equal to the starting time of job $y_1, y_2, \cdots y_n$. This statement can be formulated as follows:

$$\bigwedge_{i=1,\cdots n} t_x \geq t_{y_i}$$

The last form of the constraint is "Jobs $y_1, y_2, \cdots y_n$ require a special equipment of which only one copy is available, so no two of these jobs may run at the same time".

Which means: for every pair of job $(a, b) \in \{(y_1, y_2), (y_1, y_3), \cdots (y_{n-1}, y_n)\}$ Job a has to be finished before job b started or job a has to be started after job b finished. This statement can be formulated as follows:

Let $Y = \{(y_1, y_2), (y_1, y_3), \cdots (y_{n-1}, y_n)\}$
$\forall_{(a,b)\in Y}\ ((t_a + d_a \leq t_b) \vee (t_a \geq t_b + d_b))$

$$\forall_{a,b} \bigwedge_{i=1,\cdots n} t_x \geq t_{y_i}$$

Then, we have to make sure for every $t_1, t_2, \cdots t_12$ has to be greater than 0 because our starting time is $t = 0$. For having the smallest total job time, we have to get the biggest starting job time from the smt-solver and try to change that number into smaller number until it reaches the smallest number that still yields sat result. Here is the initial code to solve the job scheduling problem:

```
(benchmark test.smt
:logic QF_UFLRA
:extrafuns
( (t1 Real) (t2 Real) (t3 Real) (t4 Real) (t5 Real) (t6 Real) (t7 Real) (t8 Real) (t9
Real) (t10 Real) (t11 Real) (t12 Real)
)
:formula(and
(>= t1 0)
(>= t2 0)
(>= t3 0)
(>= t4 0)
(>= t5 0)
(>= t6 0)
(>= t7 0)
(>= t8 0)
(>= t9 0)
(>= t10 0)
(>= t11 0)
(>= t12 0)
(>= t3 (+ t1 6))
(>= t3 (+ t2 7))
(>= t5 (+ t3 8))
(>= t5 (+ t4 9))
(>= t7 (+ t3 8))
(>= t7 (+ t4 9))
(>= t7 (+ t6 11))
(>= t8 t5)
(>= t9 (+ t5 10))
(>= t9 (+ t8 13))
(>= t11 (+ t10 15))
(>= t12 (+ t9 14))
(>= t12 (+ t11 16))
(or (<= (+ t5 10) t7) (>= t5 (+ t7 12)))
(or (<= (+ t5 10) t10) (>= t5 (+ t10 15)))
(or (<= (+ t7 12) t10) (>= t7 (+ t10 15)))
)
```

```
)
```

From the initial result, we see that t_{12} produces the biggest starting time, thus it is always likely the last job to run. We have $t_{12} = 54$ from the initial result. We try to minimize t_{12} by adding `(= t12 x)` where x is the desired starting time. Based on our experiment, $x = 42$ is the smallest number that still yields sat result. Thus, the minimal total job time is $42 + d_{12} = 42 + 17 = 59$. (Full final code can be seen at: TODO)

Applying `yices-smt -m no3.smt` yields the following result within a fraction of a second:

```
sat
(= t8 15)
(= t1 1)
(= t4 0)
(= t6 14)
(= t3 7)
(= t10 0)
(= t5 15)
(= t12 42)
(= t2 0)
(= t7 25)
(= t9 28)
(= t11 15)
```

Expressed in a picture this yields: TODO