Report for assignments for Automated Reasoning

Authors:
Dibran Dokter s1047390 dibran.dokter@ru.nl
Sijmen van Bommel s1038820

Problem 1: Pallets

Eight trucks have to deliver pallets of obscure building blocks to a magic factory. Every truck has a capacity of 8000 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 400 kg.
- Eight pallets of skipples, each of weight 1000 kg.
- Ten pallets of crottles, each of weight 2500 kg.
- Twenty pallets of dupples, each of weight 200 kg.

Skipples need to be cooled; only three of the eight trucks have facility for cooling skipples.
Nuzzles are very valuable: to distribute the risk of loss no two pallets of nuzzles may be in the same truck.
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 start by defining a double list *trucks* for the number of trucks (*ntrucks*) and the number of different pallets (*npallettypes*).
This leads to a $n \times k$ list of integers where n is the number of trucks.
Where the integer value $trucks_{i,j}$ is the number of pallets of type j in truck i.

Besides this list we have a list for the weights for the different types of pallets, this list is used as a lookup table.

$$weightlookup = [nuzzlesweight, prittlesweight, skipplesweight, crottlesweight, dupplesweight]$$

which for this specific case is:

$$weightlookup = [700, 400, 1000, 2500, 200]$$

We define another list for the number of pallets that are required for the different types of pallets.

$$itemtypes = [nuzzlesreq, prittlesreq, skipplesreq, crottelsreq, dupplesreq]$$

which for this specific instance is:

$$itemtypes = [4, 0, 8, 10, 20]$$

After this we can start defining the different constraints on the solution.

All the number of pallets on the trucks must be non-negative:

$$nonnegative = \bigwedge_{i=1}^{ntrucks} \bigwedge_{j=1}^{itemtypes} (trucks_{i,j} \geq 0).$$

Constrain the weight of the pallets per truck by looking up the weight of the itemtype and multiplying by the number of pallets of that type for every type and then constraining the sum per truck:

$$weight = \bigwedge_{i=1}^{ntrucks} (\sum_{j=1}^{itemtypes} (weighttable_j \cdot trucks_{i,j}) \leq maxweight.$$

Constrain the capacities of the pallets per truck by summing the number of items per truck:

$$capacities = \bigwedge_{i=1}^{ntrucks} \sum_{j=1}^{itemtypes} (trucks_{i,j}) <= capacity$$

For each pallet type, require at least the given number of pallets, except for the prittles, example for nuzzles:

$$nuzzles = \bigwedge_{i=1}^{ntrucks} \sum(trucks_{i,nuzzles}) >= nuzzlesrequired$$

This can be generalized as follows:

$$items = \bigwedge_{i=1}^{trucks} \sum_{j=1}^{itemtypes} (trucks_{i,j}) >= itemsrequired_j$$

After defining these constraints the basic case without the cooling or distribution of nuzzles is satisfied.

Then we define the further constraints as follows. Skipples need to be cooled; only three of the eight trucks have facility for cooling skipples:

$$ncooledtrucks = 3 \tag{1}$$

$$cooledskipples = \bigwedge_{i=ncooledtrucks+1}^{notcooledtrucks} (trucks[i][nuzzles] == 0) \tag{2}$$

Nuzzles are very valuable: to distribute the risk of loss no two pallets of nuzzles may be in the same truck:

$$valuablenuzzles = \bigwedge i = 1^{ntrucks}(trucks_{i,nuzzles} < 2)$$

Now we have all constraints, the total formula consists of the conjunction of all these constraints:

$$nonnegative \wedge capacities \wedge items \wedge cooledskipples \wedge valuablenuzzles$$

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.

After defining these constraints we optimise for the number of prittles by using the z3 Optimizer, and we define the following requirement:

$$\bigwedge_{i=1}^{ntrucks} maximize(trucks_{i,prittles})$$

Applying `z3 trucks.py` yields the following result within half a second:

```
[0, 5, 1, 2, 0]
[0, 4, 3, 1, 0]
[0, 0, 4, 0, 4]
[1, 0, 0, 2, 5]
[1, 0, 0, 2, 5]
[1, 5, 0, 1, 1]
[0, 8, 0, 0, 0]
[1, 0, 0, 2, 5]
```

which contains:
$5 + 4 + 0 + 0 + 0 + 5 + 8 + 0 = 22$ prittles

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.

Prittles and crottles are an explosive combination: they are not allowed to be put in the same truck:

$$\bigwedge_{i=1}^{ntrucks} \neg(trucks_{i,prittles} > 0 \wedge trucks_{i,crottles} > 0)$$

we restrict the model that not both prittles and crottles can have a capacity of more than zero for each truck.

Again applying `z3 trucks.py` yields the following result within 4 seconds:

```
[0, 0, 2, 2, 4]
[1, 0, 2, 2, 1]
[0, 4, 4, 0, 0]
[1, 0, 0, 2, 5]
[0, 8, 0, 0, 0]
[0, 8, 0, 0, 0]
[1, 0, 0, 2, 5]
[1, 0, 0, 2, 5]
```

This solution has $4 + 8 + 8 = 20$ prittles

Generalization

Our solution can be generalized by giving a different number for $ntrucks$ and $npallettypesanddefiningadif$

Problem 2: Chip Design

Problem 3: Dinner

Solution:

We start by defining a datastructure for the dinner schedule, for this we use a number of *rounds*, *houses* and *people*. where $rounds = 5, houses = 5, people = 10$
This leads to a list of $n \times m \times k$ of booleans where n is the number of rounds, m is the number of houses and k is the number of people.

Which leads to the following output, where we get 5 of these arrays, one for every round.

```
[1, 1, 1, 0, 0, 1, 0, 1, 0, 0]
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[0, 0, 0, 1, 1, 0, 1, 0, 1, 1]
```

Where the row is a house, and everywhere where there is a 1 in the column that person is there.

After we define this structure we use a calculation to find the house for a certain person, this is defined as follows:

$personToHouseIndex = \lfloor p/(\lfloor people/houses \rfloor) \rfloor$

After defining the datastructure we start defining the constraints for the problem.

First we define some basic constraints.

Generalization