Automated Reasoning Assignment 1

Pim Leerkes s1060308 pim.leerkes@ru.nl
Ivo Melse s1088677 ivo.melse@ru.nl

September 2024

1 Delivery

For solving this problem, our approach was to make lists of integer variables for each of the types of cargo, where the indices are the trucks and the values are the number of units of that specific type of cargo present in a truck. In other words, we consider the variables S_i, M_i, G_i, P_i, A_i for $0 \leq i \leq 7$ (i is the index of the truck and the capital letters are the first letter of the name of that particular type of cargo). We then set the right constrains as bounds on these variables.

Initial constraints

To specify that we want four saffron in total distributed over the trucks we express it by requiring that the sum of number of pallets of saffron over all trucks is equal to 4. We also need to specify that the number of saffron is non negative in each truck, and for this we need to use a big conjuction, where we require the number of pallets of saffron in each truck to be bigger or equal to 0:

$$(\sum_{i=0}^{7} S_i) = 4 \wedge \bigwedge_{i=0}^{7} S_i \geq 0$$

For ten mushrooms, ten goats and twenty pears we use the following logical formulas respectively, which are the same as the previous formula, but only the total quantities of those types of cargo are different:

$$(\sum_{i=0}^{7} M_i) = 8 \wedge \bigwedge_{i=0}^{7} M_i \geq 0,$$

$$(\sum_{i=0}^{7} G_i) = 10 \wedge \bigwedge_{i=0}^{7} G_i \geq 0,$$

$$(\sum_{i=0}^{7} P_i) = 20 \wedge \bigwedge_{i=0}^{7} P_i \geq 0.$$

Lastly we also specify that the number of apples should be non negative in each truck. We do not specify anything else for apples yet.

$$\bigwedge_{i=0}^{7} A_i \geq 0.$$

Additional constraints

Aside from what amounts of cargo need to be delivered, there are some additional constraints. These constraints are as follows: We want at most one saffron in each truck, we want mushrooms to only be delivered using trucks with cooling facilities, a truck can carry at most ten units of cargo and lastly a truck can carry no more than 8000kg.

To express that we want at most one pallet of saffron in a truck is simple. Just a big conjuction where we say that the number of pallets of saffron in each truck is less or equal to 1:

$$\bigwedge_{i=0}^{7} S_i \leq 1.$$

Demanding the mushrooms to be cooled is also straightforward to express. Without loss of generality, we fix trucks $0, 1$ and 2 as the trucks containing cooling facilities. Now we just need to say that for the other trucks ($3 \leq i \leq 7$) there can only be exactly 0 pallets of mushrooms:

$$\bigwedge_{i=3}^{7} M_i = 0$$

If we want to make our constraints on the maximum capacity of units of cargo (10) of the trucks, we use a conjuction again over all trucks and for each truck we sum over all types of cargo and set it be less or equal to 10.

$$\bigwedge_{i=0}^{7} S_i + M_i + G_i + P_i + A_i \leq 10$$

To prevent the maximum capacity of weight (8000kg) to be exceeded. We use a very similar expression, only this time we multiply each type of cargo by its weight, and in the end the bound is 8000:

$$\bigwedge_{i=0}^{7} S_i \times 700 + M_i \times 1000 + G_i \times 2500 + P_i \times 400 + A_i \times 400 \leq 8000$$

Solution

In the end we make one conjuction out of all the previously established formulas. The maximum number of pallets of apples that could be delivered in total is: 36. The distributions of all types of cargo that the z3 SAT solver found by using optimization on a is:

Cargo Type	Truck 0	Truck 1	Truck 2	Truck 3	Truck 4	Truck 5	Truck 6	Truck 7
Saffron	0	0	1	0	1	1	0	1
Mushroom	3	3	2	0	0	0	0	0
Goat	1	1	1	2	1	1	2	1
Pears	0	6	5	1	7	0	1	0
Apples	6	0	1	6	1	8	6	8

The goats have eaten the apples

After the goats had eaten all the apples. The only thing that we needed to add to specify that apples and goats can never be in the same truck, i.e, not both A_i and G_i could exceed 0, is a simple logical formula where we again use a big conjuction to let the former hold for all trucks:

$$\bigwedge_{i=0}^{7} \neg(G_i > 0 \land A_i > 0).$$

Updated solution

The maximum number of pallets of apples that could be delivered in total when we require additionally that goats and apples cannot be together in the same truck is: 25. The distribution the SAT solver found is:

Cargo Type	Truck 0	Truck 1	Truck 2	Truck 3	Truck 4	Truck 5	Truck 6	Truck 7
Saffron	1	1	1	0	1	0	0	0
Mushroom	4	4	0	0	0	0	0	0
Goat	0	1	2	2	2	3	0	0
Pears	0	2	5	7	5	1	0	0
Apples	5	0	0	0	0	0	10	10

2 Chip design

We approach this problem by viewing each component (regular and power components) c_i for $0 \leq i < 12$ as a four value tuple, where its four values are: the x coordinate of its top left corner, the y coordinate of its top left corner, its width and its height. We denote this by $c_i^x, c_i^y, c_i^w, c_i^h$ respectively. We will let the indices of the regular components occupy the values 0 to and including 9 and the indices of the power components will be (w.l.o.g) fixed as 10 and 11.

Note: Our coordinate system has (0,0) in the top left corner. Changing y in the positive direction means we move downward.

We define the following auxiliary functions for getting one of corners (top left, top right, bottom left, bottom right respectively) of a component c_i.

$$\begin{aligned} tl(c_i) &= (c_i^x, c_i^y), \\ tr(c_i) &= (c_i^x + c_i^w, c_i^y), \\ bl(c_i) &= (c_i^x, c_i^y + c_i^h), \\ br(c_i) &= (c_i^x + c_i^w, c_i^y + c_i^h). \end{aligned}$$

With these foundations we will start specifying the constraints.

Component sizes

Components should be able to rotate 90 degrees, but they also need to adhere to their size. We specify this by demanding, using a big conjuction, that for all components c_i either its width and height are its two given dimensions in order, or its height and width are (thus allowing for rotation of 90 degrees). Let $s(i)$ be the tuple containing the width and height of component i ($s(i)_1$ refers to the first part of the tuple and $s(i)_2$ the second).

$$\bigwedge_{i=0}^{11} (c_i^w = s(i)_1 \wedge c_i^h = s(i)_2) \vee (c_i^w = s(i)_2 \wedge c_i^h = s(i)_1)$$

Where $s(i)$ is explicitly defined as:
$s(0) = (4,5), s(1) = (4,6), s(2) = (5,20), s(3) = (6,9), s(4) = (6,10), s(5) = (6,11), s(6) = (7,8), s(7) = (7,12), s(8) = (10,10), s(9) = (10,20), s(10) = (4,3), s(11) = (4,3)$.

Components are within the chip boundaries

The total dimensions of the chip are 30×30, and no component can violate this. We express this by saying that for each component, we want the coordinates of its top left corner to be bigger or equal to 0 (otherwise it could be placed outside of the chip). Furthermore we want the bottom right corner, to have its coordinates lesser or equal to 30 (again with the purpose of not letting it be outside the boundaries of the chip). To specify that all these things need to be true at the same time, we use conjunctions. To then specify that it needs to hold for all components, we use a large conjuction as well. We also use our previously defined auxiliary functions to denote the correct corners.

$$\bigwedge_{i=0}^{11} (tl(c_i)_x \geq 0 \wedge tl(c_i)_y \geq 0 \wedge br(c_i)_x \leq 30 \wedge br(c_i)_y \leq 30)$$

Here, $br(c_i)_x, tl(c_i)_x$ and $br(c_i)_y, tl(c_i)_y$ just refer to the first and second elements of the resulting tuple, which are its x and y coordinates respectively.

Components do not overlap

Two components should never overlap. Two components c_i, c_j do not overlap if any of the following holds.

- The x coordinate of the left edge of c_i is bigger or equal to the x coordinate of the right edge of c_j.
- The x coordinate of the right edge of c_i is smaller or equal to the x coordinate of the left edge of c_j.
- The y coordinate of the top edge of c_i is smaller or equal to the y coordinate of the bottom edge of c_j.
- The y coordinate of the bottom edge of c_i is higher or equal to the y coordinate of the top edge of c_j.

We formalize this as such, where for each component we require that at least one of the previous must be true by placing disjunctions between them. To compare each pair of components with each other where the components are unequal we use a big conjuction with the $0 \leq i < j$ notation.

$$\begin{aligned} &\bigwedge_{0 \leq i < j}^{11} \\ (tl(c_i)_x &\geq tr(c_j)_x \vee \\ tr(c_i)_x &\leq tl(c_j)_x \vee \\ tl(c_i)_y &\leq bl(c_j)_y \vee \\ bl(c_i)_y &\geq tr(c_j)_y) \end{aligned}$$

All regular components must touch power components

Detecting touching components is quite verbose. This is why we only explicitly show the case where given components c_i and c_j, c_i is on the right of c_j. The other cases are the same, only change the dimension and direction.

Given that component c_i is to the right of power component c_j, they touch if all of the following hold simultaneously:

- The x-coordinate of the left edge of c_i is equal to the x-coordinate of the right edge of c_j.
- The left edge of c_i and the right edge of c_j obtain the same y value for at least an interval of length greater than 0. This fact is satisfied by using the formula $tr(c_j)_y < bl(c_i)_y < br(c_j)_y$.

The other three cases can be constructed in the same way, but with the corners all rotated the same number of steps (there are 4 different rotations). All four cases in the end need to have disjunctions between them.

$$\begin{aligned}\bigwedge_{i=0}^{11} \bigvee_{j\in\{10,11\}} \\ tl(c_i)_x = tr(c_j)_x \wedge \\ (tr(c_j)_y < bl(c_i)_y < br(c_j)_y) \\ \dots\end{aligned}$$

Distance between power components

To compute the center of a component we define the following function: $center(c_i) = (tl(c_i)_x + c_i^w/2, \quad tl(c_i)_y + c_i^h/2)$. Then we simply enforce center the distance, where d is our chosen constant (which will be either 16, 17 or 18) by using a disjunction.

$$\begin{aligned} center(c_{10})_x - center(c_{11})_x \geq d \vee \\ center(c_{11})_x - center(c_{10})_x \geq d \vee \\ center(c_{10})_y - center(c_{11})_y \geq d \vee \\ center(c_{11})_y - center(c_{10})_y \geq d \end{aligned}$$

Result

Finally we use all the previously established formulas to specify the constrains in one big conjuction formula and we let the sat solver do its work for the different values of d. We got a result up to $d = 17$ (but not $d = 18$), see Figure 1. At the glance of an eye, one can already see that at least all constraints except the last hold. If we look closely, we can see that the last constraint also holds; the y-coordinates of the centres of the power components are exactly 23-6=17 apart.

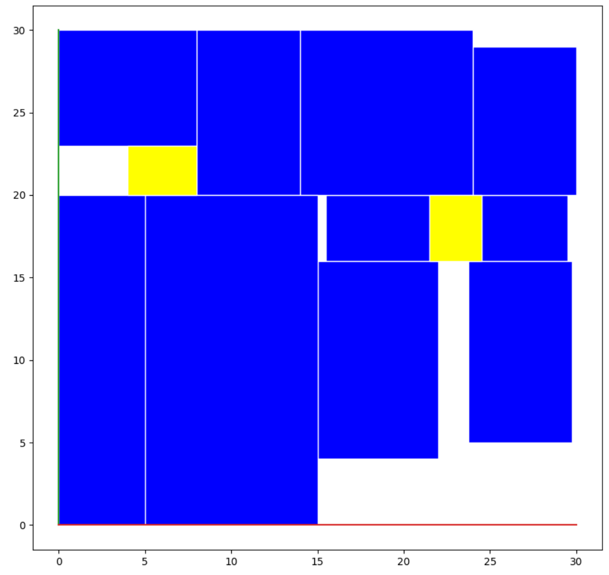

Figure 1: Result for $d = 17$ (which is simultaneously also a solution for $d = 16$). The regular components are blue and the power components are yellow.

3 Dinner

Variables

We start by declaring our variables. Since houses are bound to couples by definition, we treat them as the same thing. For simplicity, we only consider natural numbers.
Note that if we do not specify the range of a number, we assume that it is a natural number (starting at 0).
Couple membership. First of all, there are five couples with 2 people each. We let $Couples = \{p_0c, \ldots p_9c\}$ be a multiset of integers in range $[0, 4]$. The variable p_nc corresponds to person n, and the value of p_nc is the couple of which person n is a member.
People locations. We let $Locations = \{p_nr_mc \mid n < 10,\ m < 5\}$ be a multiset of integers in range $[0, 4]$. The value of p_nr_mc is the house (or couple) where person n is during round m.
Round locations. A single round takes place at the houses of two couples a and b. We let $A = \{r_ma \mid m < 5\}$ and $B = \{r_mb \mid m < 5\}$ be multisets of integers in range $[0, 4]$. Here a and b are the couples that organize round m. For

example, if couple 0 and 1 host round 5, then $r_5a = 0$ and $r_5b = 1$ (or $r_5a = 1$ and $r_5b = 0$).

We enforce the ranges of the aforementioned integers.

$$\bigwedge_{x \in Couples \cup Locations \cup A \cup B} 0 \leq x \leq 4$$

Abbreviations

We use the following auxiliary functions throughout this section.
counts counts how many times a value equal to x occurs in multiset L.
meetings counts how many times persons i and j are in the same house.

$$\begin{aligned} count(x, L) &= \sum_{y \in L} f(x, y) \\ meetings(i, j) &= \sum_{k<5} f(p_i r_k c,\ p_j r_k c) \\ \text{where } f(x, y) &= \begin{cases} 0, & \text{if } x \neq y \\ 1, & \text{if } x = y \end{cases} \end{aligned}$$

Constraints

A couple is two people

We simply enforce this by using our counting function. This assures that every couple i has exactly two members.

$$\bigwedge_{i<5} count(i, Couples) = 2$$

Each round is in two different houses

Trivial. We just need to enforce that $r_ia \neq r_ib$ for all rounds i.

$$\bigwedge_{i<5} r_ia \neq r_ib$$

Five people at each participating house every round

We use an auxiliary function nr, which given a round i and a couple j, calculates the amount of people at j's house during round i.
$nr(i, j) = count(j, \{p_0 r_i c, \ldots, p_9 r_i c\})$.

Then we enforce that for every round i and every couple j, if couple j organizes round i (as either a or b), then there are exactly 5 people present at j's house at round i.

$$\bigwedge_{i<5}\bigwedge_{j<5}((r_i a = j) \rightarrow (nr(i,j) = 5)) \;\wedge\; ((r_i b = j) \rightarrow (nr(i,j) = 5))$$

Every couple hosts two rounds

We use our counting function to enforce that for each couple i, there are exactly two variables in the multiset $\{r_0 a, \ldots r_4 a, r_0 b, \ldots r_4 b\}$ that are equal to i.

$$\bigwedge_{i<5} count(i, \{r_0 a, \ldots r_4 a, r_0 b, \ldots r_4 b\}) = 2$$

Every couple is present at their own organized round

For each round i, couple j, and person k, if round i is held at j, and k is a member of couple j, then person k must be at j's house in round i. The cases for a and b work exactly the same.

$$\bigwedge_{i<5}\bigwedge_{j<5}\bigwedge_{k<10}((r_i a = j \wedge p_k c = j) \rightarrow p_k r_i = j) \wedge ((r_i b = j \wedge p_k c = j) \rightarrow p_k r_i = j)$$

No two participants are in the same house in every round

For every two distinct persons i and j, it is not the case that for every round k, i and j are in the same house.

$$\bigwedge_{i<j<10} \neg(\bigwedge_{k<5} p_i r_k c = p_j r_k c)$$

(A) Every two people meet each other at least once

This requirement can be formalized by simply using our *meetings* function; we assert that the number of meetings between any two distinct persons i and j must be at least 1.

$$\bigwedge_{i<j<10} meetings(i,j) \geq 1$$

(B) Every person meets each other at most three times

This is the exact same principle as A.

$$\bigwedge_{i<j<10} meetings(i,j) \leq 3$$

(C) Couples never meet each other outside their houses

Since it follows from the existing constraints that every couple hosts two rounds at which both hosts have to be present, it follows that C holds if and only if the members of each couple meet each other exactly twice. Therefore it suffices to enforce that any two distinct people who are in the same couple meet each other exactly twice.

$$\bigwedge_{i<j<10} p_i c = p_j c \rightarrow (meetings(i,j) = 2)$$

(D) No person can be a guest in the same house twice

For all persons i and couples j, if i does not belong to couple j (i.e. i would be a guest if they were present in j's house), then i cannot be in j's house more than once.

$$\bigwedge_{i<10} \bigwedge_{j<5} p_i c \neq j \rightarrow (inhouse(i,j) \leq 1)$$

Where $inhouse(i,j) = count(j,\ \{p_i r_0 c, \ldots, p_i r_4 c\})$.
We can get the amount of times that i has been in couple j's house by simply using our count function to count the amount of rounds where i was in j's house

Results

As required, $A \wedge C \wedge D$ is unsatisfiable. Below we present our results for $A \wedge C$, $A \wedge D$, and $B \wedge C \wedge D$.

$A \wedge C$

People and locations per round

	r0	r1	r2	r3	r4
p0	0	2	4	3	0
p1	0	1	3	3	1
p2	0	1	3	4	0
p3	2	2	4	3	0
p4	2	1	3	3	0
p5	2	2	3	4	1
p6	2	2	4	4	1
p7	0	1	4	4	0
p8	0	2	3	3	1
p9	2	1	4	4	1
la	2	2	4	4	0
lb	0	1	3	3	1

People belonging to couples

p0	p1	p2	p3	p4	p5	p6	p7	p8	p9
0	1	0	2	3	2	4	4	3	1

$A \wedge D$

People and locations per round

	r0	r1	r2	r3	r4
p0	2	4	3	0	0
p1	1	3	3	4	0
p2	2	4	1	0	0
p3	2	3	1	4	2
p4	2	4	3	4	0
p5	1	3	3	0	2
p6	1	4	3	4	2
p7	1	4	1	0	2
p8	2	3	1	0	2
p9	1	3	1	4	0
la	1	4	3	0	2
lb	2	3	1	4	0

People belonging to couples

p0	p1	p2	p3	p4	p5	p6	p7	p8	p9
0	3	0	2	4	3	4	1	2	1

$B \wedge C \wedge D$

People and locations per round

	r0	r1	r2	r3	r4
p0	0	2	4	3	2
p1	0	1	0	3	2
p2	4	2	0	1	2
p3	4	2	0	3	3
p4	0	1	4	1	2
p5	0	2	0	1	3
p6	4	2	4	1	3
p7	4	1	4	3	2
p8	0	1	4	3	3
p9	4	1	0	1	3
la	4	1	4	3	3
lb	0	2	0	1	2

People belonging to couples

p0	p1	p2	p3	p4	p5	p6	p7	p8	p9
2	0	2	3	1	0	4	4	3	1

4 Program safety

Part 1

To establish for what values of n the program can never read crash, we formalized the program in the following way. First we need to specify the variables. For the variables in the program a and b we make one in our formalization for each iteration of the program so a_i and b_i for $0 \leq i \leq 10$ (we have the initial values and the 10 iterations of the program). The preconditions of the program are that both a_0 and a_1 are equal to 1, i.e,

$$a_0 = 1 \land b_0 = 1.$$

We formalize the for loop by using for each iteration i the variables that we need in that iteration. This is what it looks like in the end:

$$\bigwedge_{i=1}^{10} (a_i = a_{i-1} + 2 \times b_{i-1} \land b_i = b_{i-1} + i) \lor (a_i = a_{i-1} + i \land b_i = b_{i-1} + a_i).$$

The $\lor$ is there in the middle to represent the φ variable in the program that each iteration can be either equal to true or to false.

Finally the program has a postcondition that is, after the for loop we check the value of b and to know if we indeed reach crash we set

$$b_{10} = 700 + n$$

where we manually set n to different values in order to investigate if the program would be sat or not. If sat then we reach crash, if not we don't crash. When making everything into one big formula by placing conjuctions between them. It turns out that the program is able to reach crash for $n = 1, 3, 6, 7, 9$

Part 2

It is clear that 1 is the smallest value of n such that the program can reach crash. In order to provide the table of the variable values in each iteration, we need to add variables for φ to our formalization. Specifically φ_i for $1 \leq i \leq 10$. We then change the for loop formula from part 1 into the following:

$$\bigwedge_{i=1}^{10} (a_i = a_{i-1} + 2 \times b_{i-1} \land b_i = b_{i-1} + i \land \varphi_i = 1) \lor (a_i = a_{i-1} + i \land b_i = b_{i-1} + a_i \land \varphi_i = 0).$$

The logic here is that if φ is true for a certain iteration i we need to have it so that we φ_i to 1 and vice versa. We now have enough information to provide the table:

iteration	0	1	2	3	4	5	6	7	8	9	10
a	1	3	5	8	38	76	82	89	479	488	1870
b	1	2	7	15	19	24	106	195	203	691	701
φ	-	true	false	false	true	true	false	false	true	false	true