Practical Assignment
Automated Reasoning 2IMF25

prof dr H. Zantema
MF room 6.078
email: h.zantema@tue.nl
August, 2021

The programs to be used

Programs to be used:

- Z3: https://github.com/Z3Prover/z3 .
- Yices: http://yices.csl.sri.com/.

  Both Z3 and Yices are programs for satisfiability modulo theories (SMT). They accept standard SMT format, in particular boolean SAT format.

- NuSMV: http://nusmv.fbk.eu/.

  This is a symbolic model checker based on BDDs

- Prover9 and Mace4: https://www.cs.unm.edu/~mccune/mace4/

  These are tools for predicate and equational logic: Prover9 for giving proofs based on resolution, and Mace4 for finding counterexamples.

Each of the problems should be solved using one of these tools. The tools should do the job: manual modifications of the problems should be avoided.

For further information about the course we refer to
http://www.win.tue.nl/~hzantema/ar.html

The assignment

The practical assignment has to be executed by one or two persons. These groups of at most two persons should be registered in Canvas. The practical assignment consists of two parts.

The results of the assignment have to be described in two reports that should be submitted in PDF via Canvas, preferably less than 20 pages each. For the report on the first part the deadline is **September 29, 2021**, for the report on the second part the deadline is **October 27, 2021**.

For all used formulas an extensive documentation is required, explaining the approach and the overall structure. A generic approach is preferred, since this may result in clearer

descriptions, increasing the confidence in the correctness of the results. Formulas of more than half a page should not be contained in the report, instead the structure of the formula should be explained. From the output of the programs relevant parts should be contained in the report, and observations on computation time should be reported. The answers on the problems should be motivated, where relevant including pictures is appreciated. Every report should contain name, student number and email address of each of the authors. In case of two authors each of them is considered to be responsible for the full text and all results.

Guidelines for grading:

- Clear and generic descriptions are appreciated, both of the formulas themselves and the way they were designed. An example of appreciated style is given at

  `http://www.win.tue.nl/~hzantema/prvb.pdf`.

- For both parts at least 3 out of the 4 solutions should be correct to obtain a 7.
- Not giving a solution at all for one problem is preferred over giving a wrong solution.
- Reasons for obtaining higher than a 7 may be:
  - all problems correctly solved,
  - remarkably clear and structured descriptions,
  - approaches allowing generalizations,
  - original approaches and solutions.

The problems for the first part

For the first part (deadline September 29, 2021) you have to find and describe solutions of the following 4 problems using the indicated programs.

1. 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.

   (a) 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.

   (b) 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.

2. Give a chip design containing two power components and ten regular components satisfying the following constraints:

   - Both the width and the height of the chip is 30.
   - The power components have width 4 and height 3.
   - The sizes of the ten regular components are 4×5, 4×6, 5×20, 6×9, 6×10, 6×11, 7×8, 7×12, 10×10, 10×20, respectively.
   - All components may be turned 90°, but may not overlap.
   - In order to get power, all regular components should directly be connected to a power component, that is, an edge of the component should have a part of length > 0 in common with an edge of the power component.
   - Due to limits on heat production the power components should be not too close: their centres should differ at least 16 in either the x direction or the y direction (or both).

   What if this last distance requirement of 16 is increased to 17? And what if it is increased to 18?

3. Five couples each living in a separate house want to organize a dinner.

   Since all restaurants are closed due to some lock-down, they will do it in their own houses. The dinner will consist of 5 rounds. Due to the 1.5 meter restriction in every house presence of at most 5 people is allowed, by which every round has to be prepared and served in two houses simultaneously, each with the corresponding couple and three guests. Every couple will serve two rounds in their house, and in between the rounds participants may move from one house to another.

   Every two people among the 10 participants meet each other at most 4 times during these 5 rounds. Further there are four desired properties:

   (A) Every two people among the 10 participants meet each other at least once.

   (B) Every two people among the 10 participants meet each other at most 3 times.

   (C) Couples never meet outside their own houses.

   (D) For every house the six guests (three for each of the two rounds) are distinct.

   (a) Show that (A) is possible, both in combination with (C) and (D), but not with both (C) and (D).

   (b) Show that (B) is possible in combination with both (C) and (D).

4. Consider the following program:

   $a := 1;\ b := 1;$
   for $i := 1$ to 10 do
   if ? then $\{a := a + 2b;\ b := b + i\}$ else $\{b := a + b;\ a := a + i\};$
   if $b = 700 + n$ then crash

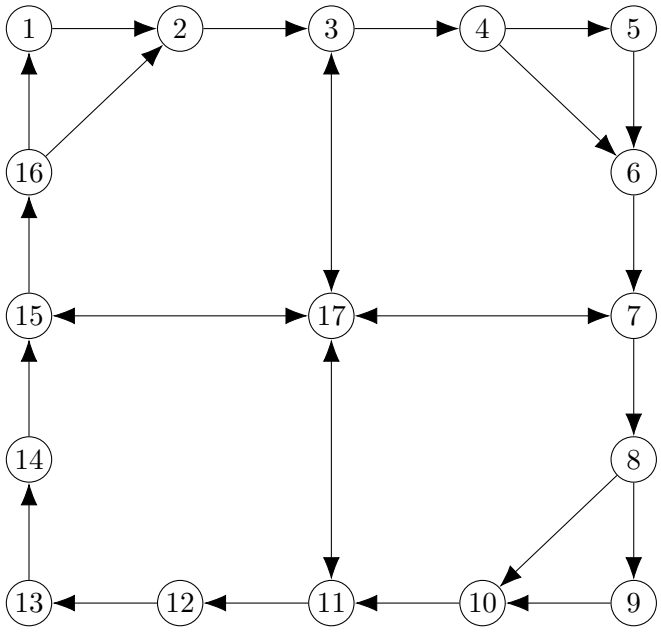
   Here '?' is an unknown test that may yield false or true in any situation. Note that the test on crash is outside the loop, so is only tested at the end.

   Establish for which values of $n = 1, 2, \ldots, 10$ it is safe, that is, will not reach 'crash'. Show for one of the non-safe values of n how $b = 700 + n$ can be reached.

The problems for the second part

For the second part (deadline October 27, 2021) you have to find and describe solutions of the following 4 problems using the indicated programs.

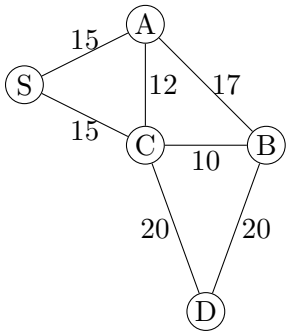
1. Consider the following packet switching network as introduced in the Appendix.



Here every arrow denotes a channel. The double arrows from node 17 to nodes 3, 7, 11, 15 denote pairs of channels: one channel in every direction. Note that there is no channel from node 12 to node 14. In total there are 27 channels. The network has the property that for every two nodes $n \neq n'$ there is a unique shortest path from n to n' of length at most 7. The routing function rout is defined by the property that for every two nodes $n \neq n'$ the channel $\mathsf{rout}(n, n')$ is the first arrow = channel in the shortest path from n to n'. Hence by definition it is correct. An encoding of this routing function is given in `http://www.win.tue.nl/~hzantema/defnetw.txt`. The initial state is defined to be the state in which all channels are free. Investigate whether there is a deadlock reachable from the initial state, for the following choices of the set M of main nodes; in case of a reachable deadlock indicate the obtained deadlock.

(a) $M = \{1, 5, 9, 13\}$;
(b) $M = \{2, 4, 6\}$;
(c) $M = \{1, 3, 5, 15\}$;
(d) $M = \{1, 8, 10\}$;
(e) $M = \{1, 2, 3, 4, 5\}$;
(f) $M = \{11, 12, 15\}$;
(g) $M = \{11, 12, 13, 15\}$;
(h) $M = \{5, 12, 14\}$;
(i) $M = \{5, 11, 14\}$;

2. Four non-self-supporting villages A, B, C and D in the middle of nowhere consume one food package each per time unit. The required food packages are delivered by a truck, having a capacity of 250 food packages. The truck has to pick up its food packages at location S containing an unbounded supply. The locations of this supply location and the villages and the roads between them are shown in the following picture. Here every number indicates a distance, more precisely, the number of time units the truck needs to travel from one village to another, including loading and delivering. The villages only have a limited capacity to store food packages: for A and C this capacity is 110, for B and D it is 160. Initially, the truck is in S and is fully loaded, and in each of the four villages there are 80 food packages.



(a) Show that it is impossible to deliver food packages in such a way that each of the villages consumes one food package per time unit forever.

(b) Show that this is possible if the capacity of the truck is increased to 260 food packages. (Note that a finite graph contains an infinite path starting in a node v if and only if there is a path from v to a node w for which there is a non-empty path from w to itself.)

3. **(a)** Consider the rewrite system consisting of the following rewrite rules

$$\begin{array}{rcl} a(x,y) & \to & a(y,x), \\ a(x,y) & \to & a(p,a(x,x)), \\ a(x,a(y,z)) & \to & a(a(x,y),z), \\ a(x,a(x,x)) & \to & d(x). \end{array}$$

Figure out whether $a(p,p)$ rewrites to $d(d(d(p)))$ in a finite number of steps.

(b) In mathematics, a *monoid* is defined to be a set M with an element $I \in M$ and a binary operator $*$ satisfying

$$x*(y*z) = (x*y)*z, \text{ and } x*I = x, \text{ and } I*x = x,$$

for all $x, y, z \in M$. A monoid is called *commutative* if $x*y = y*x$ for all $x, y \in M$.

(a) Determine whether every monoid in which $x*x = I$ for all $x \in M$, is commutative.

(b) Determine whether every monoid in which $x*(x*x) = I$ for all $x \in M$, is commutative.

(c) Determine whether every monoid in which $(x*x)*(x*x) = I$ for all $x \in M$, is commutative.

For every case where the answer is 'no', give the size of the smallest such non-commutative monoid.

4. Give a precise description of a non-trivial problem of your own choice, and encode this and solve it by one of the given programs.

Here 'trivial problems' include both minor modifications of earlier problems and single solutions of simply specified puzzles like sudokus. In case of doubt please contact the lecturer.

Appendix: Packet switching networks

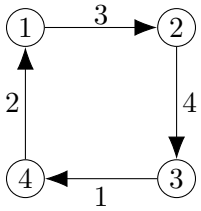
A packet switching network consists of

- a finite set N of *nodes*;
- a finite set C of *channels* c, each having a *source* $\mathsf{source}(c) \in N$ and a *target* $\mathsf{target}(c) \in N$, such that every node $n \in N$ is the source of at least one channel in C;
- a subset M of N of *main nodes* that may send messages and act as destination;
- a *routing function* $\mathsf{rout} : N \times M \to C$ for which $\mathsf{source}(\mathsf{rout}(n,m)) = n$ for every $n \in N$, $m \in M$, $m \neq n$ to decide at which channel to proceed when a message with destination m arrives in node n. If $m = n$ then $\mathsf{rout}(n,m)$ is undefined.

For $m \in M$, $n \in N$, write $\mathsf{next}_m(n) = \mathsf{target}(\mathsf{rout}(n,m))$. Note that $\mathsf{next}_m : N \to N$. A switching network is *correct* if for every $m, m' \in M, m \neq m'$, there exists $k > 0$ such that $\mathsf{next}_m^k(m') = m$. This means that if from node m' a message will be send to m, this message

will arrive at its destination m in k steps. Typically, the routing function will be chosen in such a way that this k will be as small as possible.

A channel may be free, or be occupied by a message. In the latter case it may block access to that channel for other messages. A state from which no step is possible to another state, is called a *deadlock*. For instance, consider the following network: $M = N = \{1, 2, 3, 4\}$, four channels depicted by an arrow from its source to its target, and all four channels are occupied and labeled by the destination of its message.



Then we have a deadlock: every channel wants to switch to a next channel since the destination is different from the target of the channel. But this is blocked for every channel since every next channel is occupied. As every node has exactly one outgoing channel, there is no choice in defining rout.

For a given network we want to figure out whether a deadlock exists, and if so, whether it is reachable. Before considering this in more detail first we have to define how steps are done. We restrict to the simplest setting where this is *asynchronous*, that is, at every moment steps can be done without a central control by a clock. For investigating deadlocks the contents of the data package in the message does not play a role: the contents of a channel is identified by the destination $m \in M$ of the corresponding message. We consider the following steps:

- (send) If node $m' \in M$ wants to send a message to node $m \in M$, $m \neq m'$, then it tries to put this destination into channel $\mathsf{rout}(m', m)$. This is only possible if this channel is free, and as a result this channel is not free any more, but occupied by m.

- (receive) If a channel $c \in C$ contains destination $m \in M$, and $\mathsf{target}(c) = m$, then the message may reach its destination by which c is made free.

- (process) If a channel $c \in C$ contains destination $m \in M$, and $\mathsf{target}(c) \neq m$, then the message may proceed to $c' = \mathsf{rout}(\mathsf{target}(c), m)$. This is only possible if this channel c' is free, and as a result this channel c' is not free any more, but occupied by m, while c is made free.

When allowing these three types of steps (send, receive or process), a typical question to be asked is whether the system allows deadlocks starting from the initial state in which all channels are free.