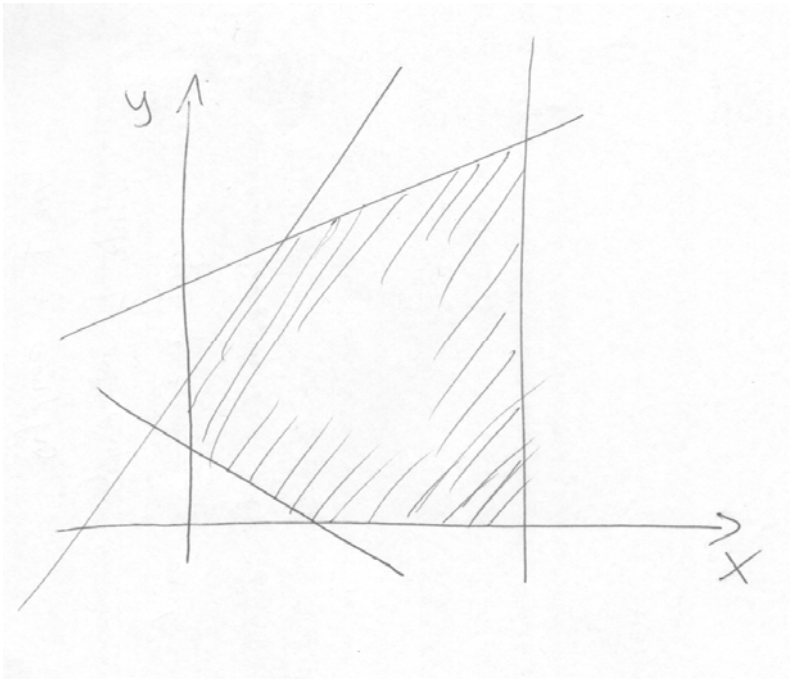
DECO

Discrete Event Control and Optimization

Exam SSY 220, Wednesday, January 15, 14:00-18:00, M

Teacher: Martin Fabian, (772) 3716

Time when teacher present: 15:00, 17:00



Solutions and answers should be complete, written in English and be unambiguous and well motivated. In the case of ambiguously formulated exam tasks, the suggested solution with possible assumptions must be motivated. The examiner retains the right to accept or decline the rationality of assumptions and motivations.

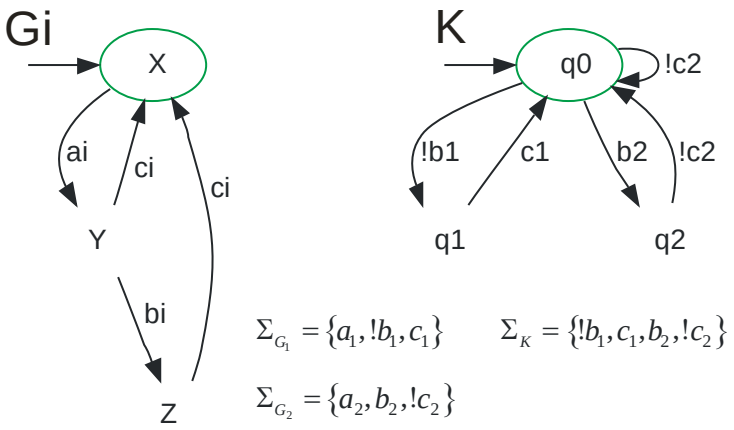
In total the exam comprises 25 credits. For the grades 3, 4 and 5, is respectively required 10, 15 and 20 credits.

Solutions will be announced on the course web-page on the first week-day after the exam date. Exam results are announced through Chalmers' administrative routines. The corrected exams are open for review seven work days after the exam, 12:30 – 13:30 at the department.

Aids: None.

Task 1. Supervisory Control Theory

Controllability is an important property in the Supervisory Control Theory; the supervisor must be controllable with respect to the plant and the uncontrollable events. Below is a specification K and a plant component G_i of which there are two (i = 1,2), and the uncontrollable events are !b1, and !c2 (note that the plant components are not identical in this respect). By a clever trick called *plantify* we can remove the distinction between plant and specification while still being able to synthesize a proper supervisor.



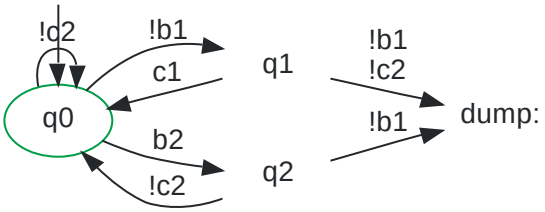
$\Sigma_{G_1} = \{a_1, !b_1, c_1\}$ $\quad \Sigma_K = \{!b_1, c_1, b_2, !c_2\}$

$\Sigma_{G_2} = \{a_2, b_2, !c_2\}$

a) Plantify the given specification K above. (2p)

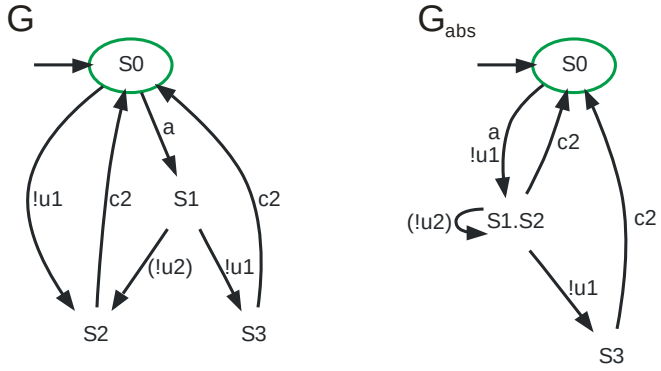
b) Is K controllable with respect to the G_i (i = 1,2) and the uncontrollable events !b1, and !c2? Explain. (2p)

K is not controllable. After a1.!b1.a2, the plant $G_1 \| G_2$ can do !c2 but the specification does not agree. We see this in the plantified version of K below, where !b1.!c2 leads to the dump state.



Task 2. Abstractions for Compositional Verification

Emilia was given the task to calculate a supervisor by compositional synthesis. One of the automata looked like the one to the left below. Emilia happily abstracted away the local uncontrollable event (!u2) into the automaton on the right.



a) Is the abstraction valid for controllability verification? Explain. (3p)

b) Is the abstraction valid for non-blocking verification? Explain. (3p)

Answering these questions requires finding some test (plant or specification) that does not include the local event (!u2), such that it gives different results with G and G_{abs}. And different results means controllable and uncontrollable, and blocking and non-blocking, respectively. Note that to assess validity for non-blocking verification, the test must fail due to blocking, not due to uncontrollability, and vice versa. So, different tests are needed to assess the two properties.

The abstraction is not valid for controllability verification. We can see this if we consider a test with the language (!u1.c2). This is controllable with G but uncontrollable with G_{abs}.*

The abstraction is also not valid for non-blocking verification. We can see this if we consider a test which blocks the controllable event a and has the language (!u1.!u1.c2). This is non-blocking with G_{abs} but blocking with G.*

Task 3. Linear Programming

A company can produce two products, A and B. Producing one unit of A takes two hours, and one unit of B takes three hours. There are 40 hours of production time available per week. However, by paying a fixed cost of €1200, an additional eight hours can be used. If more than 10 units of A are produced, at least five units of B must be produced. The profit for each unit of A is €200, and for each unit of B €400.

Formulate a linear programming problem that maximizes the profit for the company. (5p)

Decision variables:

x_A and x_B, number of units produced of the respective products, A and B.

y_1 is 1 if additional time is used, else 0.

y_2 is 1 if more than 10 units of A are produced, else 0.

$$\max 200x_A + 400x_B - 1200y_1$$
$$\text{s.t. } 2x_A + 3x_B \le 40 + 8y_1$$
$$My_2 \ge x_A - 10$$
$$x_B \ge 5y_2$$
$$x_A, x_B \ge 0, \text{integer}$$
$$y_1, y_2 \in \{0,1\}$$

As usual, M is a "large enough" positive constant. It is necessary to multiply y_2 with M, since y_2 is 0-1, while x_A - 10 can be much larger than 1. The expression is to say that y_2 must be 1 when $x_A > 10$.

Task 4. Linear Programming

We have the following linear programming problem:

$$\begin{aligned} \min \quad & -x_1 - x_2 - x_3 - x_4 + x_5 & \\ \text{s.t.} \quad & x_1 + x_3 & \le 3 \\ & x_1 + 2x_3 & \le 4 \\ & x_2 + x_4 & \le 4 \\ & x_2 + x_5 & \le 3 \\ & & \\ & x_1, x_2, x_3, x_4, x_5 & \ge 0 \end{aligned}$$

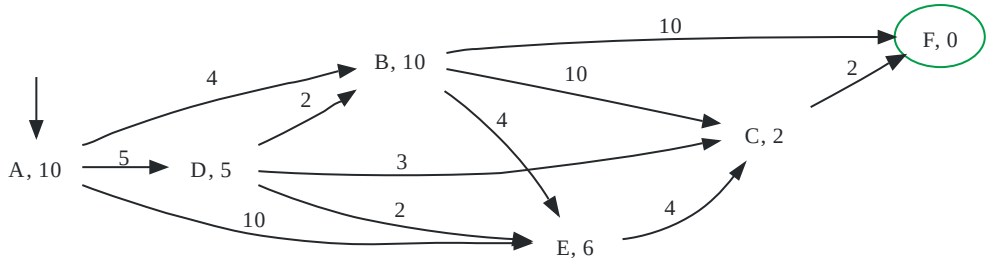
a) Solve it. (3p)

b) If we require the x_i to be integer, would the solution in a) also be a solution to this IP-problem? Motivate. (1p)

From the constraints we can see that this problem is actually two distinct sub-problems, one involving x_1 and x_3, one involving x_2, x_4, and x_5. These can be solved individually. Of course, since x_5 increases *the objective function, it must be 0. The two problems are then two-dimensional, and can hence be solved graphically; see the solution below.*

The solution is integer, and we know that if the LP-solution is integer, then it is also the IP-solution.

Task 5. Discrete Optimization



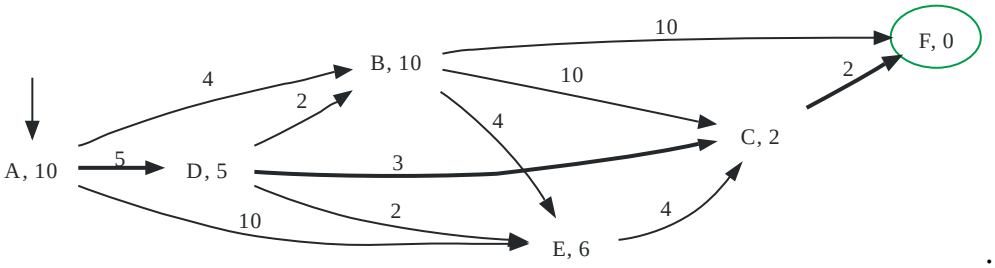
Above is given a graph, with weights on the edges, and for each node an estimate of the cost to reach the goal node *F*.

a) Using Dijkstra's algorithm, find the least cost path through the graph. (3p)

b) Using the A* algorithm, find the least cost path through the graph. (3p)

In both cases, show on each iteration which node is taken out from and put in to the queue, and also what the queue looks like.

The optimal path is marked by thick lines. It is the same for both algorithms, of course.



The workings of Dijkstra's algorithm is shown to the left, below, and A is to the right.*

Dijkstra's Algorithm		*A**	
A[0,-]	B[4,A] D[5,A] E[10,A]	A[0,10,-]	D[5,5,A] B[4,10,A] E[10,6,A]
B[4,A]	D[5,A] C[14,B] E[8,B] F[14,B]	D[5,5,A]	C[8,2,D] B[4,10,A] E[7,6,D]
D[5,A]	E[7,D] F[14,B] C[8,D]	C[8,2,D]	F[10,0,C] B[4,10,A] E[7,6,D]
E[7,D]	C[8,D] F[14,B]	F[10,0,C]	E[7,6,D] B[4,10,A]
C[8,D]	F[10,C]		
F[10,C]			

The first element within the brackets is the current cost of the node, the last element is the current parent, and the middle element is the estimate.

Note that the given estimates are actually the true optimal values from the node to the goal. Naturally, such estimates fulfill the requirements we place on estimates, monotonic and not over-estimating, and this example also shows that the tighter the estimate, the better A performs. Here it goes straight for the goal, never diverging from the optimal path (which Dikstra's algo does).*

$$\min \quad -x_1 - x_2 - x_3 - x_4 + x_5$$

$$\begin{aligned} \text{s.t.} \quad x_1 + x_3 &\leq 3 \\ x_1 + 2x_3 &\leq 4 \\ x_2 + x_4 &\leq 4 \\ x_2 &\leq 3 \\ x_5 &\leq 7 \\ x_i &\geq 0 \end{aligned}$$

From the constraints we can see that this is actually three distinct sub-problems, one involving x_1 and x_3, one involving x_2 and x_4, and one involving only x_5. Of course, since x_5 increases the obj fcn, it must be zero. The other two problems can each be solved graphically.

optimum = -7