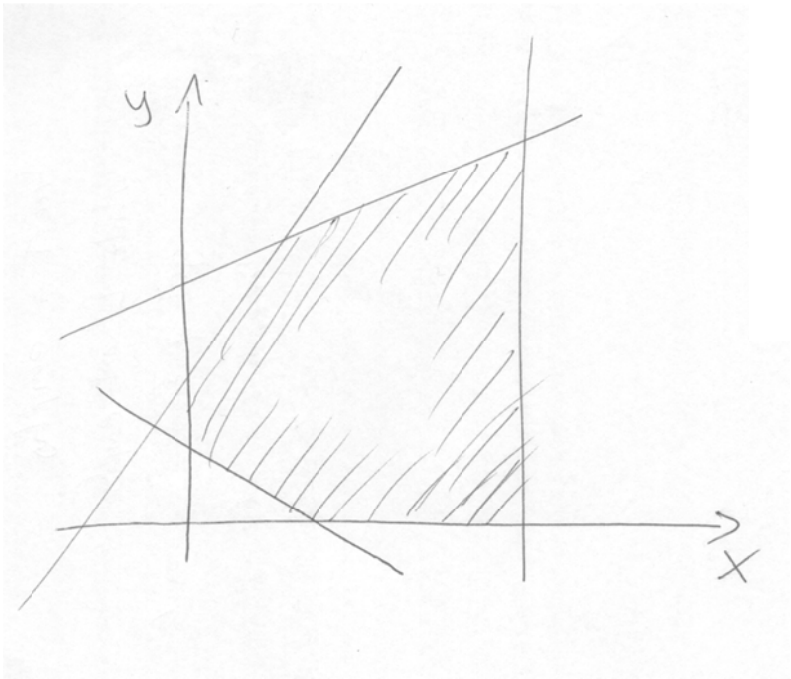
DECO

Discrete Event Control and Optimization

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

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.

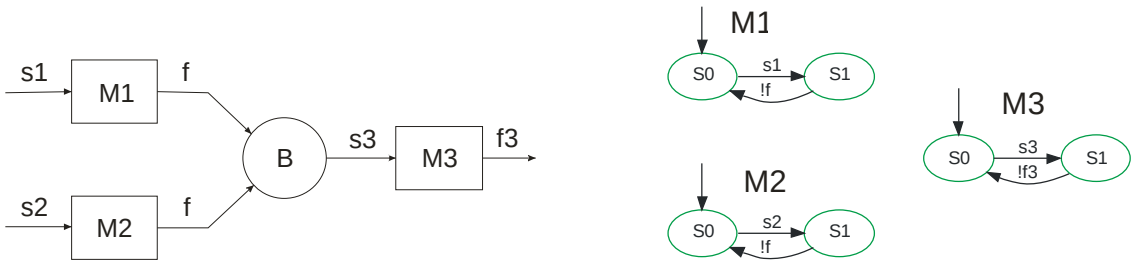
In total the exam comprises 25 credits. For the grades 3, 4 and 5, is respectively required 10, 15 and 20 credits.

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. Compositional Verification and Modular Synthesis

We have a small manufacturing system as shown below. M1, M2 and M3 are machines, and B is a buffer. The plant can be modeled by the simple automata given to the right below. The s_i events (i = 1,2,3) are controllable, while the f and f_3 events are uncontrollable. Notice that M1 and M2 output their products in synchrony into B. Thus, this can be regarded as an assembly operation, which adds a single product to the buffer.



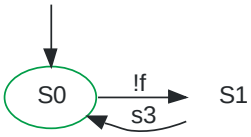
a) Give a specification for the buffer to hold one work piece, never to over- or underflow, and to always end up empty. (1p)

b) Use the abstraction-based compositional method to verify whether the system is *nonblocking* or not. Be sure to explain in detail the procedure. (2p)

c) Use the abstraction-based compositional method to verify whether the system is *controllable* or not. Be sure to explain in detail the procedure. (3p)

d) Using modular synthesis, calculate a supervisor for the non-abstracted system. (2p)

The specification B can look like this:



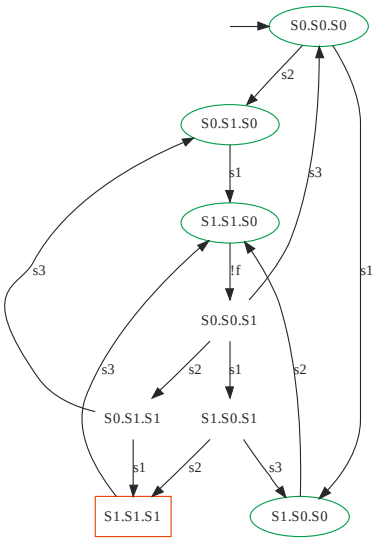
We can start by abstracting M3, noting that !f3 is a local uncontrollable event. This abstraction is valid both for verification and synthesis, and it makes M3 self-loop only so it can be ignored.

For compositional verification, we can merge states that are connected by local events, whether those events are uncontrollable or not. Doing this for the M1, M2, and B creates self-loop only automata with their respective single states marked. Thus, the system is obviously nonblocking.

For compositional controllability verification, before abstracting it we have to "plantify" the specification, and mark all states except the dump state, which looks like this:

Merging the marked states of the plantified B would create a non-deterministic automaton, so we cannot do that, even though they are connected by the local event s3 (well, we can actually, but that's another story). Still, since M1 and M2 are self-loop only, we can easily determine that the dump state is reachable, and hence the system is not *controllable.*

Modular synthesis is straightforwardly done by only considering M1, M2 and B. The result looks like this (the forbidden state can be removed, of course):



Task 2. Linear Programming

For the LP-problem given below, all constants, a, b, m_1 and m_2 are larger than 0, with $m_2 > m_1$.

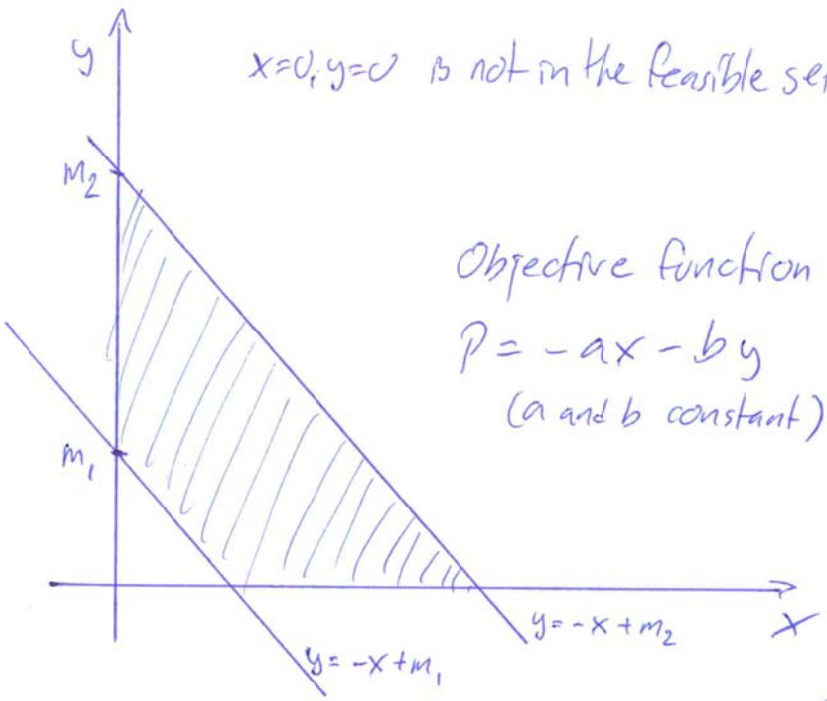
$$\min\ p = -ax - by$$
$$s.t \quad x + y \geq m_1$$
$$x + y \leq m_2$$
$$x, y \geq 0$$

a) Formulate an LP problem in standard form for the given constraints and objective function. (1p)

b) Assuming that $b \geq a$, solve the problem by the Simplex method. (4p)

c) What would happen if $b < a$? You do not have to solve for that case, just explain. (2p)

d) What would happen if $m_2 < m_1$? Explain. (1p)



$x=0, y=0$ is not in the feasible set!

Objective function

$P = -ax - by$

(a and b constant)

$$\min P = -ax - by$$

s.t. $y \geq -x + m_1$ $\quad$ $y + x - s = m_1$

$y \leq -x + m_2$ $\quad$ $y + x + t = m_2$

With slack and surplus variables

Rewrite

$s = -m_1 + x + y$

$t = m_2 - x - y$

At $x=0, y=0$, $s = -m_1$

Not Good!

Rearrange

$y = m_1 - x + s$

$t = m_2 - x - (m_1 - x + s) = (m_2 - m_1) - s$

$P = -ax - b(m_1 - x + s) = -bm_1 + (b-a)x - bs$

At the origin ($x=0, s=0$) y and t are positive. GOOD!

As we see, $P^* \leq -bm_1$

20140524-ctd

Let's solve it

$$y = m_1 - x + s$$

$$t = (m_2 - m_1) - s$$

$$p = -b m_1 + (b-a)x - b s$$

Are we done? NO, since increasing s decreases p

Let $x = 0$, then

$$y = m_1 + s \geq 0$$

$$t = (m_2 - m_1) - s \geq 0 \Rightarrow s \leq (m_2 - m_1)$$

Rewrite

$$s = (m_2 - m_1) - t$$

$$y = m_1 - x + (m_2 - m_1) - t = m_2 - x - t$$

$$p = -b m_1 + (b-a)x - b(m_2 - m_1) + bt =$$

$$= -b m_2 + (b-a)x + bt$$

Are we done? If $b \geq a$, then yes!

(If $b < a$, then NO!!)

Optimum

$$p^* = -b m_2$$

$$s^* = m_2 - m_1$$

$$y^* = m_2$$

$$x^* = 0$$

$$t^* = 0$$

Note that the constraints have the slope -1

If $b > a$, then p has slope $k < -1$

If $b < a$, then p has slope $k > -1$

In this case, obviously the optimum is at $x^* = 0$, $y^* = m_2$

In the other case, $x^* = m_2$, $y^* = 0$

If $m_2 < m_1$ *as we see from the figure above, the constraints would be conflicting and there would exist no solution.*

Task 3. Integer Linear Programming Theory

The Simplex algorithm in tableau form works on a matrix looking like

$$\left[\begin{array}{c|c} A & b \\ \hline c^T & -p \end{array}\right]$$

where *A*, *b*, and *c* are the respective parameter matrix and vectors. Using the tableau to solve the LP problem will result in the optimal objective function value *p* negated in the lower right hand corner.

Explain why this value turns up negated. (3p)

The optimal objective function value turns up negated, since what the tableau expresses is really the problem formulated as

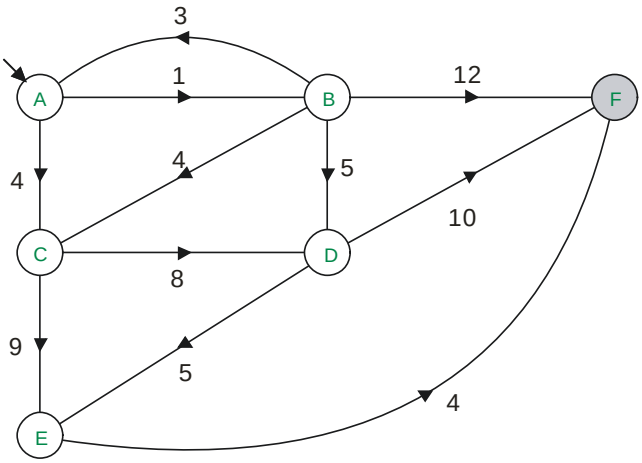
$$\begin{aligned} &\min p \\ s.t \quad &Ax - b = 0 \\ &c^T x - p = 0 \\ &x \geq 0 \end{aligned}$$

The core of this formulation can be expressed in matrix form as

$$\left[\begin{array}{c|c} A & b \\ \hline c^T & p \end{array}\right]\left[\begin{array}{c} x \\ \hline -1 \end{array}\right] = 0$$

Here we now see why the optimal objective function value ends up negated.

Task 4. Discrete Optimization



Above is given a graph, with costs on the edges.

a) Using Dijkstra's algorithm, find the least cost path through the graph. (2p)

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

c) Explain why Dijkstra's algorithm and A* require all edge costs to be non-negative. (1p)

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. For b) you need to define a good estimate.

The optimal path is A-B-F.

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[1,A] C[4,A]	A[0,10,-]	B[1,9,A] C[4,10,A]
B[1,A]	C[4,A] F[13,B] D[6,B]	B[1,9,A]	F[13,0,B]: C[4,10,A]: D[6,8,B]
C[4,A]	D[6,B] F[13,B] E[13,C]	F[13,0,B]	D[6,8,B]: C[4,10,A]
D[6,B]	E[11,D] F[13,B]		
E[11,D]	F[13,B]		
F[13,B]			

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 monotonic and not over-estimating, and this example 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).*

If edge costs were allowed to be negative, it would not hold that the cost of going from A to B is less or equal to the cost of going from A to B to C. This would have as consequence that the optimal solution to a sub-problem would not necessarily be part of the optimal solution to the global problem. The algorithms rely on this "dynamic programming" property. Also, if we had negative costs, going around indefinitely in a loop would allow arbitrarily good solutions, the algorithms would not terminate.