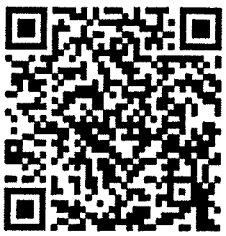
Försättsblad till skriftlig tentamen vid Linköpings universitet

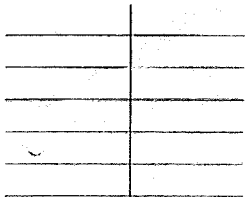


Datum för tentamen	2017-08-17
Sal (1)	TER4(7)
Tid	8-12
Kurskod	TDDD48
Provkod	TEN1
Kursnamn/benämning **Provnamn/benämning**	Automatisk planering Skriftlig tentamen
Institution	IDA
Antal uppgifter som ingår i tentamen	4
Jour/Kursansvarig Ange vem som besöker salen	Jonas Kvarnström
Telefon under skrivtiden	0704-737579
Besöker salen ca klockan	ja
Kursadministratör/kontaktperson (namn + tfnr + mailaddress)	Anna Grabska Eklund, ankn. 2362, anna.grabska.eklund@liu.se
Tillåtna hjälpmedel	inga
Övrigt	
Antal exemplar i påsen	

TDDD48 Automated Planning 2017-08-17

Important Instructions: Read before you begin!

- Though the questions are in English, feel free to answer in Swedish if you prefer! **Det går bra att skriva på svenska!**
- Please, write clearly (block letters if necessary), and use the *lined* side of the paper unless you have a good reason not to! Checked paper works well for math but tends to make text difficult to read...



- *Clear and comprehensible* explanations and motivations are always required. This does not necessarily mean that each answer should be a long essay. What is important is that all the relevant facts are present and clearly explained.
- To see if you have succeeded, turn the tables and **try to misunderstand.** Specifically, after you write an answer, go back and see if you can creatively misinterpret the answer. If so, *clarify, clarify, clarify*!
- A good way to show your knowledge is to write explanations that can be understood by someone *else* who *does not already know* the correct answer.

  If your answer explains a topic well enough that a fellow student could *learn* something new from the answer and/or apply it in practice, you have probably succeeded.

  Conversely, if we ask how HTNs work, saying that "there are methods and operators and you construct a tree using patterns and there can be constraints too, and there's no goal" does not provide much in terms of *useful* information, and certainly will not give you a full score.
- Concrete examples or counterexamples may be useful as part of a motivation. If so, please make sure you include all relevant information about the example you have chosen to use. What is relevant naturally depends on how you use the example.
- When asked to provide examples or illustrate something through a planning problem instance, save time by *keeping the example as small as possible.*

1 General Concepts in Automated Planning

Recall that in standard classical planning problems, no function symbols are allowed, goals constrain only the final state reached by a plan, and preconditions, effects and goals consist of simple sets of positive and negative literals, without disjunction, quantification or conditional effects.

a) Let $P = (O, s_0, g)$ be a classical planning problem. Let $\Pi = \{\pi \mid \pi \text{ is applicable to } s_0\}$ be the set of all action sequences that are executable starting in the initial state s_0. Is Π *always* infinite, *sometimes* infinite or *never* infinite? Motivate and explain why. **(1 point)**

b) Continuing in the context of the previous question, let $S = \{\gamma(s_0, \pi) \mid \pi \in \Pi\}$ be the set of all states that are reachable through executable action sequences starting at s_0. Is S *always* infinite, *sometimes* infinite or *never* infinite? Motivate and explain why.

(Here we have extended the state transition function $\gamma(s, a)$ to operate on sequences π in the natural way, by applying each action in π in the order of execution.) **(1 point)**

c) Let $P_1 = (O, s_0, g_1)$ and $P_2 = (O, s_0, g_2)$ be two classical planning problems that share the same operators and initial state. Let $\pi_1 = [a_1, \ldots, a_n]$ be a solution of length n for P_1, and let $\pi_2 = [b_1, \ldots, b_m]$ be a solution of length m for P_2.

Suppose that the concatenated action sequence $\pi_1 \cdot \pi_2 = [a_1, \ldots, a_n, b_1, \ldots, b_m]$ happens to be *executable* starting at state s_0. Can we then be certain that it is also a *solution* for P_2 (achieves the goals of P_2)? Show clearly that this *is* or *is not* the case. If you want to provide an example or counterexample, try to keep it short and simple. **(2 points)**

2 Search, Search Guidance and Heuristics

Note: A few questions are the same as on the previous exam. This is intended is a test to see to what extent students study and learn from exams they have already participated in. Don't count on this happening regularly...

Given the size of a typical search space, a planner almost always needs some form of search guidance as opposed to searching blindly. Often such guidance takes the shape of a *heuristic function* that evaluates the "quality" of a certain search node, corresponding to one specific choice that can be made at a particular point in the search space. Other methods use *pruning* to completely remove some parts of the search space that can be shown to be uninteresting.

a) The FF (Fast Forward) planner pioneered the use of a new heuristic function $h_{\text{FF}}(s)$. It also introduced *helpful actions*, which can be extracted directly from the computation of $h_{\text{FF}}(s)$.

   - Explain clearly *how* the computation of $h_{\text{FF}}(s)$ yields a set of "helpful actions" as a side effect. The explanation must show in some way *how* these actions are identified – in other words, how they can be distinguished from the "non-helpful" actions.

     (Note: To provide a sufficiently good explanation of helpful actions, you will most likely also have to explain at least some aspects of the computation of $h_{\text{FF}}(s)$ in some detail. However, our focus will not be on the heuristic itself but on whether it is clear how the helpful actions are extracted – and why they are helpful, as discussed below.)

   - Also motivate *why* one could expect this subset of actions to be more "helpful" in a state s than the other actions in the planning domain (S, A, γ).

     (Note: As above, explaining why helpful actions are helpful will probably require explaning some aspects of the computation of the $h_{\text{FF}}(s)$.)

   - Finally, explain how FF uses helpful actions to guide search.

   (4 points)

b) Describe clearly how *dual queues* and *boosted queues* have been applied by Fast Downward to achieve a more balanced use of *preferred successors* (another word for the *helpful actions* pioneered by the FF planner). **(2 points)**

c) Many heuristics work by solving subproblems of the actual problem being solved, and then combining the costs of these subproblems in some way. For example, a single subproblem for the h_2 heuristic is constructed by selecting two literals from the "real" goal, and then (under)estimating the cost of achieving only these two literals. The heuristic then solves one such subproblem for *every* pair of goal literals, and returns the maximum of all calculated costs.

What is the corresponding way in which subproblems are constructed for a *pattern database heuristic* (PDB heuristic)? Describe the general steps and ideas involved. Assume that the problem is already in state-variable representation as opposed to classical representation, so that an initial conversion phase (as described during the lectures) is not necessary. **(2 points)**

3 Planning with Incomplete Information

a) Explain the meaning of, and the differences between, three types of planning: *Fully observable*, *partially observable* and *non-observable*.

Leading questions: Who would observe something? What can or can't this entity observe? When would such observations be *made*? When could they be useful and what would they be used for? **(2 points)**

b) In one of the later lectures, we discussed a number of different planning problems involving incomplete information about the world, including the Stochastic Shortest Path Problem (SSPP).

The Stochastic Shortest Path Problem is formulated in a way that is different from the general MDP (Markov Decision Process) problem – the objective is different, and even the expected type form of plan execution is different. Nevertheless, an SSPP problem instance can *transformed* into a modified problem that is phrased in the correct way to be solved by a general MDP solver.

How do we transform the SSPP problem into an MDP problem in order to guarantee that the resulting solutions terminate properly – that is, which changes must be made to the problem?

Why is this transformation required?

Hint: Consider what the objectives really are in SSPP problem formulations and MDP problem formulations. This is to some extent explicit in the very names of these problem types.

(3 points)

4 Motion Planning

a) When generating a path for a robot, it is important to know whether the robot is *holonomic*.

What is the difference between holonomic and non-holonomic robots? You do not need to explain which one is which, as long as you clearly show what the *difference* is.

Give an example of a holonomic robot and one example of a non-holonomic robot. Again, labeling them correctly is not part of the exam. **(2 points)**

b) Many forms of motion planning build on the use of a *local planner* that knows how to generate *local plans* for a particular type of vehicle. As the local planner is typically quite limited in its ability to connect arbitrary points, there must also be a *higher level planner* that can repeatedly call the local planner. For example, the higher level planner can be a *probabilistic roadmap* (PRM) planner.

- PRM planners begin with a pre-processing phase where the starting point and goal are not known. Explain what the PRM planner does during this phase, how it decides when to stop, and what the resulting output is (the information that is saved to be used later).

  We will not require the description to be perfect in every detail, but it should be sufficiently detailed to demonstrate that you grasp the most important concepts.

- Each time the PRM planner receive a starting point and goal, how does it make use of the result of the pre-processing phase to create a motion plan for this particular problem instance?

(4 points)